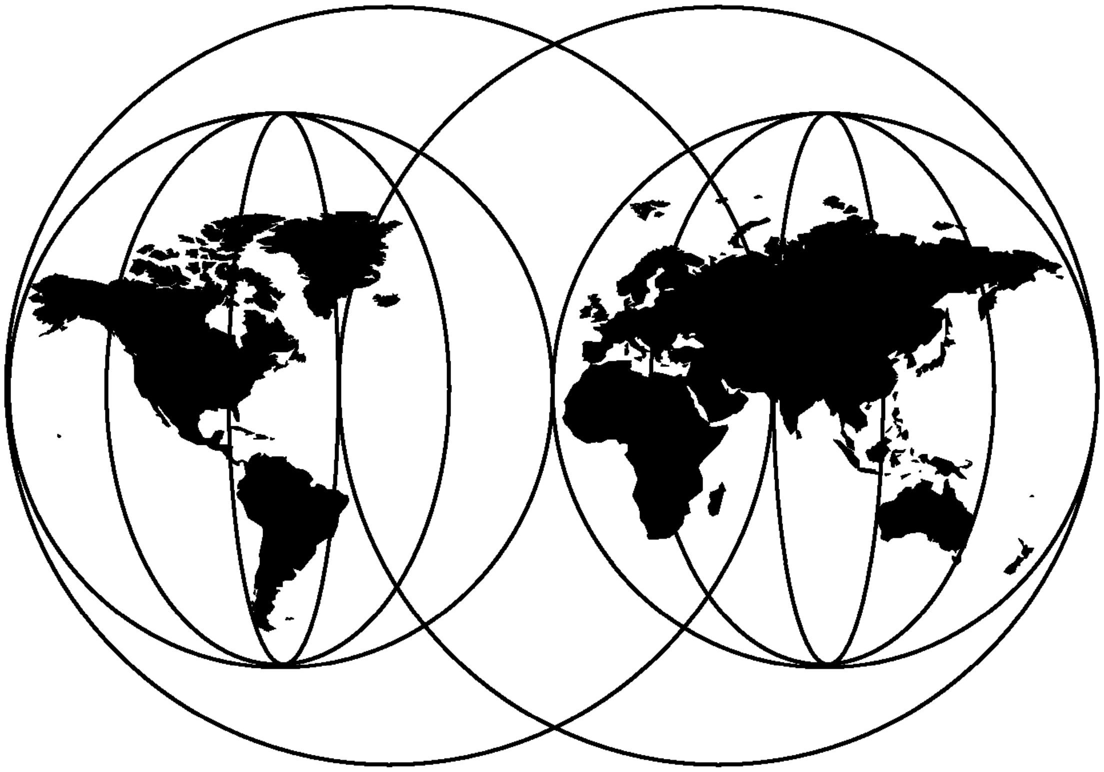
Image and Workflow Library: Capacity Planning and Performance Tuning for VisualInfo and Digital Library Servers

Mike Ebbers, Gordon Campbell, Peter Carter,
Cataldo Mega, Andy Mitchell, Romil Turakhia



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-4974-01

International Technical Support Organization

SG24-4974-01

Image and Workflow Library: Capacity Planning and Performance Tuning for VisualInfo and Digital Library Servers

May 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix E, “Special Notices” on page 227.

Second Edition (May 1999)

This edition applies to Version 2, Release 3 of the licensed program IBM eDMSuite ImagePlus VisualInfo, product number 580n-AAR, Version 2, Release 4 of the licensed program IBM DB2 Digital Library, product number 580n-AAR, and Version 4, Release 1 of the licensed program IBM VisualInfo for OS/400, product number 5733-A18.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Contents

Preface

This redbook tells you how to perform capacity planning and performance tuning for VisualInfo and Digital Library servers. We have gathered information from a wide variety of sources, including the product developers, field professionals, and data bases and publications on these topics. This edition applies to IBM eDMSuite ImagePlus VisualInfo, Version 2 Release 3, IBM DB2 Digital Library, Version 2 Release 4, and IBM VisualInfo for OS/400, Version 4 Release 1.

As product support experts and developers visit customers in the field, they find many of the same issues over and over. Many of these could be prevented with the correct procedures in place. This redbook is written to help minimize the outages from known procedural problems and errors.

An architectural overview of VisualInfo and Digital Library is provided in chapters 1-3. Information on capacity planning for servers is provided in chapters 4-7. Performance tuning tips for servers are detailed in chapters 8-14. Scenarios from customer installations are described in chapters 15-17.

This redbook is written for people responsible for architecting and tuning a solution using VisualInfo or Digital Library servers.

You can read it from cover to cover or use it for reference.

Some knowledge of VisualInfo and or Digital Library is assumed.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working together with the developers of VisualInfo and Digital Library at the Santa Teresa Lab, San Jose, California and at the IBM Software Solution s Development Lab in Research Triangle Park, North Carolina.

The advisor for this Redbook:

Mike Ebbers is a Senior Software Specialist at the International Technical Support Organization, Poughkeepsie Center. He organizes skills transfer projects on Lotus Domino for S/390, workflow and imaging. He has worked for IBM for 25 years in technical positions: 10 in the field, 10 in education and 5 with the ITSO.

The authors of this redbook:

Gordon Campbell is a Senior I/T Specialist for the Image Competency Center in Gaithersburg, MD, USA. He supports VisualInfo and Digital Library tools involving sizing and performance, and has authored and teaches the VisualInfo API class. He has 20 years experience with IBM.

Peter Carter is a Certified Consulting I/T Specialist with the e-Document Management Solutions unit of the IBM Global Services group in Toronto, Canada. He joined IBM Canada in 1996 but has been working with ImagePlus and VisualInfo on the AS/400 since 1990.

Cataldo Mega is a Solutions Architect at the EMEA Digital Library Competency Centre at the IBM Laboratory Boeblingen, Germany. He joined IBM in 1986 and since 1990 he has worked in the EMEA Technical Support Center for High End Systems such as ES/9000 and SP2.

Andy Mitchell is a Senior I/T Specialist for IBM Global Services in South Africa. Over the past 5 years he has been the lead specialist on several VisualInfo and Digital Library implementations. He has more than 17 years experience with IBM.

Romil Turakhia is the Founder/Director of Software Services and Overseas Operations of Automated Workflow Pvt. Ltd., India & Singapore. This company has been an IBM business partner since its inception, before which Romil was actively involved in providing enterprise-wide solutions, and also provided mentorship on the Web for Microsoft Products.

Acknowledgements

The authors of the first edition of this redbook were:

Steve Buzby
Tiina Pihlasviita

Special thanks to **Kiyoshi Shintani** for hosting three residents at Santa Teresa lab and to **Ken Nelson** for hosting a resident at the Research Triangle Park lab.

Thanks also to the following people for their invaluable contributions to this project:

Kevin Barker, Research Triangle Park Lab
Will Belknap, Santa Teresa Lab
Joe Brownsmith, Santa Teresa Lab
Su-Jin Chan, Santa Teresa Lab
Jim Lin, Santa Teresa Lab
Jim Reimer, Santa Teresa Lab
Phillip Sanchez, Santa Teresa Lab
Mayank Shah, Santa Teresa Lab
Basuki Soetarman, Santa Teresa Lab
Robert Summers, Santa Teresa Lab
Sundar Veliah, Santa Teresa Lab
Ron Yorita, Santa Teresa Lab
All those who contributed to the UK Swat Team Database

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 241 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Part 1. Architecture and Design

- VisualInfo
- Digital Library
- AS/400
- ContentConnect

Chapter 1. VisualInfo Architecture—an Overview

This redbook helps you size and tune a VisualInfo or Digital Library system. We have collected tips and techniques from all over the world, taken from years of experience. Some are general and some pertain to a specific platform.

We hope you enjoy using this redbook. We trust that it will save you much time and effort.

1.1 Introducing VisualInfo

A VisualInfo solution offers custom document processing for both large and small organizations. It allows capture, storage, and retrieval of documents online and provides document, folder, and workflow management capabilities. VisualInfo also provides extensive data integrity and security.

It provides access to document processing and library management from a departmental to an enterprise-wide basis. Departments located anywhere across an enterprise can access their own documents as well as enterprise documents. The system also provides for distributed placement of documents and other data on a variety of LAN-based and host-based servers, while maintaining seamless user access. The system consists of one or more clients connected to a library server and one or more object servers. The components can consist of OS/2 or Windows clients connected to OS/2, Windows NT, AIX, MVS/ESA or OS/390 servers, where the objects are stored and indexed.

The library server stores the indexing and retrieval information for each of the documents. It interacts with the client to receive and direct requests for documents residing on any of the object servers. The object servers are the actual repositories for the stored document images. They serve as the file room for the system.

With VisualInfo, a customized document management solution can be developed that includes library and information processing capabilities for multiple media types. Image, workflow, and other applications can be created to automate and gain control of the information the enterprise processes each day, and which increase productivity, lower storage costs, and improve customer service.

Figure 1 on page 4 shows an overview of the VisualInfo system, and we will now introduce its components.

1.2 Library Server

The library server uses a relational database to manage digital objects and provide data integrity. The library server maintains index information and controls access to objects that are stored on one or more object servers. Library servers that run on OS/2, AIX, and Windows NT can use DB2, DB2 UDB, or the Oracle database. Library servers that run on MVS/ESA use only DB2 to manage the library contents. A VisualInfo system has one library server.

All requests for information and services are channeled through the library server. Its function is to build SQL requests for DB2 and transmit the results from the SQL queries. Library server machines typically have heavy CPU loads. The library server performs the following functions:

- It manages library data.
- It maintains index information.
- It controls access to documents stored in the object server.

For instance, the library server directs requests from the library client to the appropriate object server to do the following:

- Store, retrieve, and update documents stored in object servers
- Update and query the indexes and descriptive information stored in the library catalog

The client application supports one library server in a given system session. However, custom applications can be developed to reference multiple library servers using the folder manager or the library client interface.

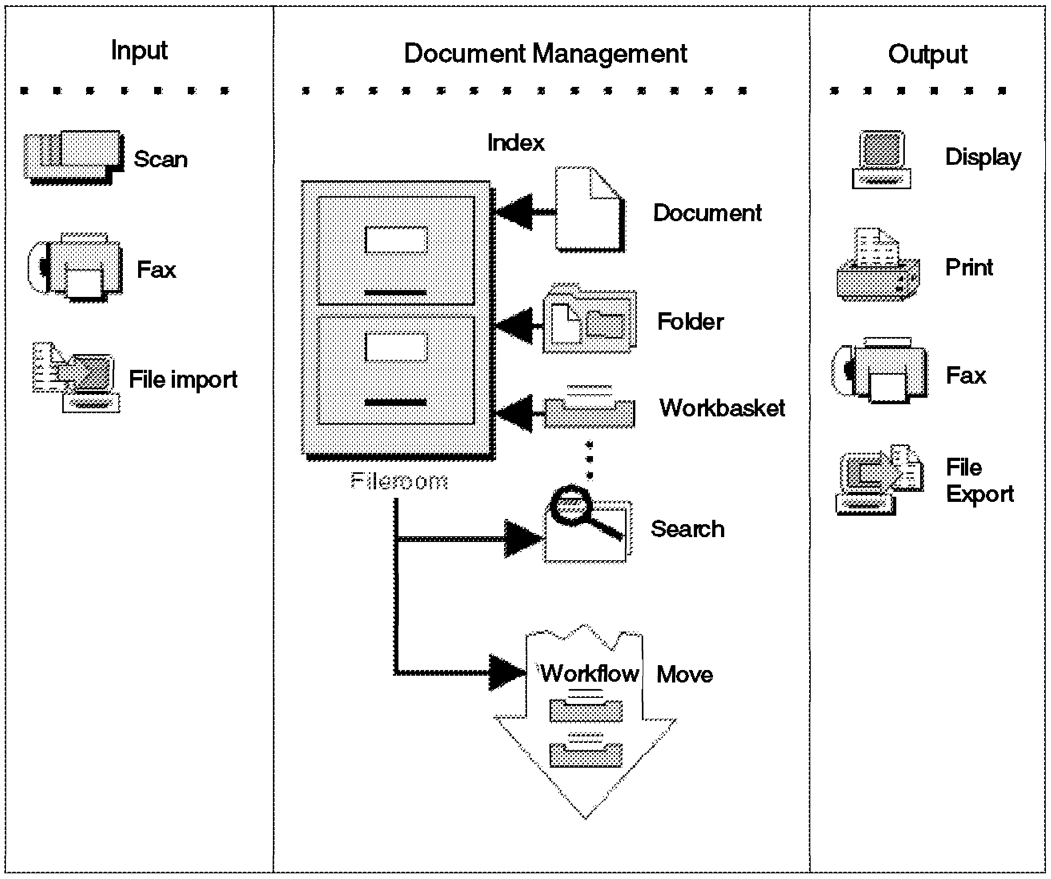


Figure 1. VisualInfo Overview

1.3 Object Server

Each object server is associated with one library server and maintains the documents stored in the library. The object server receives requests from the library server and communicates with the library client to complete requests from your application. Object servers natively support attachment of both direct access storage devices (DASD), more commonly known as disks) and a range of optical storage devices. When used with the IBM Advanced Distributed Storage Manager (ADSM), then it also supports all devices supported by this system such as Tape Library Systems. Object servers also provide system-managed storage (SMS) to automatically move documents from one storage medium to another, such as from DASD to optical disk. A VisualInfo system can have one or more object servers for maximum configuration flexibility and performance, and objects can be automatically moved between these servers as needed by your application. The object server uses a DB2 or Oracle database, and ADSM for SMS.

On the AIX and NT operating systems, this is also known as the LAN-Based Object Server (LBOS), and on MVS, this is known as the Host-Based Object Server (HBOS).

1.3.1 LAN-Based Object Server Functions

The LAN-based object server, or LBOS, runs with the OS/2, NT and AIX operating systems. (For an introduction to HBOS, see 1.3.8, “Object Server Processes on MVS” on page 8.) The LBOS manages several object storage areas, including the staging area, archive drive, and optical storage. The archive drive is the first permanent storage area for an object. The object server machines often have heavy I/O loads, since they store and retrieve large objects.

The object server has to retrieve the object from an input workstation, which may be a scanning workstation, file server, or client workstation. The object server has to send the object to a client when a valid copy does not exist in the client′s local cache (as previously mentioned, the local cache applies *only* for OS/2 clients). The background tasks used to copy the objects from one storage location to another and to delete unneeded objects may affect the object server′s ability to service online application requests. The process times for each are modeled using an IBM internal VisualInfo sizing tool, and are presented throughout this book. The VISizer is introduced in Chapter 4, “ How to Plan for Capacity” on page 33.

1.3.2 Staging Area

The staging area is a critical component of the object server. It is defined at installation time. The staging area must be large enough to hold all objects that are stored and opened (retrieved) in the time frame that the de-stage and purge processes are dormant to prevent the object server from entering a quiesce state. When an object is first stored, the object server gets it directly from the client and writes it into the staging area. When first stored in the staging area, the object/file has an attribute of “r” for readonly to prevent inadvertent deletion or changing of the file. The entry in the object server′s BASE_OBJECTS table has a status of S, meaning the object resides in the staging area and a path of 0.

When you request an object from the object server, you are given a copy of the object that was written to the staging area. If the object has already been de-staged or migrated to an archive drive, a copy is written to the staging area, after the object is delivered to the client. This copy is deleted by the purger according to the last reference date of the file on a FIFO (first-in, first-out) basis.

The management of the staging area is controlled by the de-stager process, which moves new and modified objects to their first SMS device, and the purger process, which deletes objects that have been de-staged. Parameters that specify the size of the staging area, the time between startup of the de-stager and purger processes, and the percentage of used space to reclaim are set through the system administration application.

Note: It is advantageous to have a staging area large enough that stage/de-stage cycles do not occur during prime shift. However, if the system were to fail, then be aware that you should plan to allow for enough time to run CHKDSK (or the object server recovery routine for NT and AIX) on the staging area during a re-boot (which may take multiple hours on, for instance, a RAID device).

1.3.3 De-Stager

Objects in the staging area waiting to be archived have the following attributes:

- Obj_status of S in BASE_OBJECTS
- Action_date of 12/31/9999 in BASE_OBJECTS
- File attribute of r in the staging area for AIX, NT or OS/2

When the de-stager runs, the object/file is renamed and copied to an archive area (file directory) named LBOSDATA that is resident on a volume (drive letter) defined for the object′s storage class. The specific volume is determined by the storage policy assigned to the collection ID for the object. The object′s row in the BASE_OBJECTS table is updated to reflect the volume ID, file name, path, and action date. The status changes from S to A (for archived). The de-stager removes the readonly file attribute. On AIX, the de-stager changes to permission field to "rw."

1.3.4 Purger

Once the de-stager archives the object to the archive drive and resets the file attribute of R, the file is eligible for purging. Copies of the objects remaining in the staging area are only erased from the staging area when certain space limitations, as defined by the system administrator, are reached. The purger deletes the file from the staging area using a FIFO algorithm based on the last reference date. This causes the oldest files to be deleted first. The purger continues to delete files until the free space percentage threshold is reached or its allowed time to run is expired. The user has no control over which objects in the staging area are purged.

Note that if purge thresholds are reached during a de-stage cycle, the purger automatically starts (even if disabled).

1.3.5 Migrator

Once the objects have been de-staged, they are eligible to be migrated to another storage drive or media. The migration of objects is managed through the SMS items called *management class* and *storage class*. The migrator moves the object/files from one storage class (DASD, for instance) to the next storage class (for example, OPTICAL) based on the time values specified for the object′s management class. It is not based on frequency of use. Once an object has been moved to the new storage class, the BASE_OBJECTS table is updated with the new location information, and the previous copy of the object is deleted.

The object server components are shown in Figure 2.

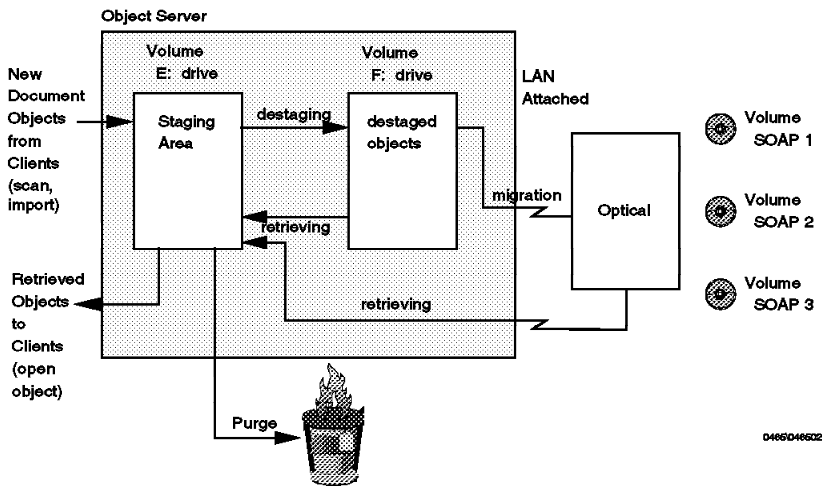


Figure 2. Object Server Components

1.3.6 LAN Cache and Remote Servers

Many large installations of VisualInfo require a large central library server, a large central object server, and several remote distributed object servers located near the users. The remote object servers improve retrieval response time for the remote workgroup. Objects are stored permanently in the central object server. When a remote retrieve request is received, a copy of the object is sent to the remote object server. It is kept in the staging area for later retrieval requests by that workgroup. This is beneficial in a distributed system, where the object retrieval may be costly in response time over a wide area network.

1.3.6.1 What Is LAN Cache

On OS/2, Windows NT and AIX object servers, objects can be cached to improve performance where multiple users need local access to the same objects. The objects reside in the staging area of the object servers where they are cached.

1.3.6.2 Enabling LAN Cache

To enable the LAN cache feature, the global flag in the library server must be set to on and the local flag for each LBOS must be on. The global flag is accessed from the library server configuration icon when you are logged on to the system administration application on the second page of the library server configuration General Tab dialog. To set the local flag for LAN cache, open the **Object Server Assignment** icon, select the desired LBOS, and select **LAN Cache Enable** box.

1.3.6.3 LAN Cache Process Flow

The retrieve function issued from the client workstation initiates the request that an object is brought to the client workstation, or to pre-fetch an object into the local object server staging area and make it available for processing. If the LBOS LAN cache does not have a cache copy, the copy is loaded as follows:

- The client sends a request to the library server for an object.
- The library server sends a cache retrieval request to cache LBOS. If the object is cached, the cache object server sends the object to the client. The LBOS returns a positive request to the library server and the process ends.
- If the object is not located in the cache object server, the library server sends a retrieval request to the archive object server.
- The archive object server sends the object to the client.
- The archive object server returns to the library server.
- The library server returns to client to complete the request.
- The library server sends a cache copy request to the archive object server.
- The archive object server sends a copy to the cache object server.
- The cache object server sends an acknowledgement of the request to the library server.

The object server knows if an object is a cache copy by examining the object name. Cache object names have a timestamp at the end of the file name. The timestamp part of the name is used by the object server to determine if it is a valid cached object. The cached objects are kept in the LBOS staging area.

1.3.7 LBOS Remote Migration

It is possible to define a remote LBOS or HBOS and have objects migrate according to SMS definitions from their current object server to a remote object server.

The remote migration occurs based on the object′s storage management policy, or because a move order is received from the library server.

1.3.8 Object Server Processes on MVS

The Host-Based Object Server (HBOS) runs in an MVS/ESA environment under control of CICS and uses the IBM Object Access Method (OAM) to allow System Managed Storage (SMS) to store and retrieve the objects. OAM maintains the object directory database and the object DASD database (32 K and 4 K table). The TSO-based ISMF panels are used to define the SMS items and object transition cycle times. Running the object storage management cycle migrates the objects based on the management class, storage class, and ACS routine determinations. There is no staging area concept in MVS, unlike the LBOS. Objects are retrieved directly from the permanent location, either the DB2 4 K or 32 K table, from optical, or from tape.

1.3.9 Object Server Processes on AIX and NT

The AIX-based object server functions are similar to the OS/2 server, except the SMS functions can be provided by the ADSTAR Distributed Storage Manager (ADSM).

1.4 Client Components

VisualInfo includes a choice of configurable client applications that provide a complete end-user interface for the document management system. The system also includes a wide variety of application programming interfaces (APIs) that allow the use of business applications with the VisualInfo system and the available fax processing, workflow, and document capture complementary offerings.

User exits provide points where you can create application-specific processing routines to customize your VisualInfo system; see Figure 3.

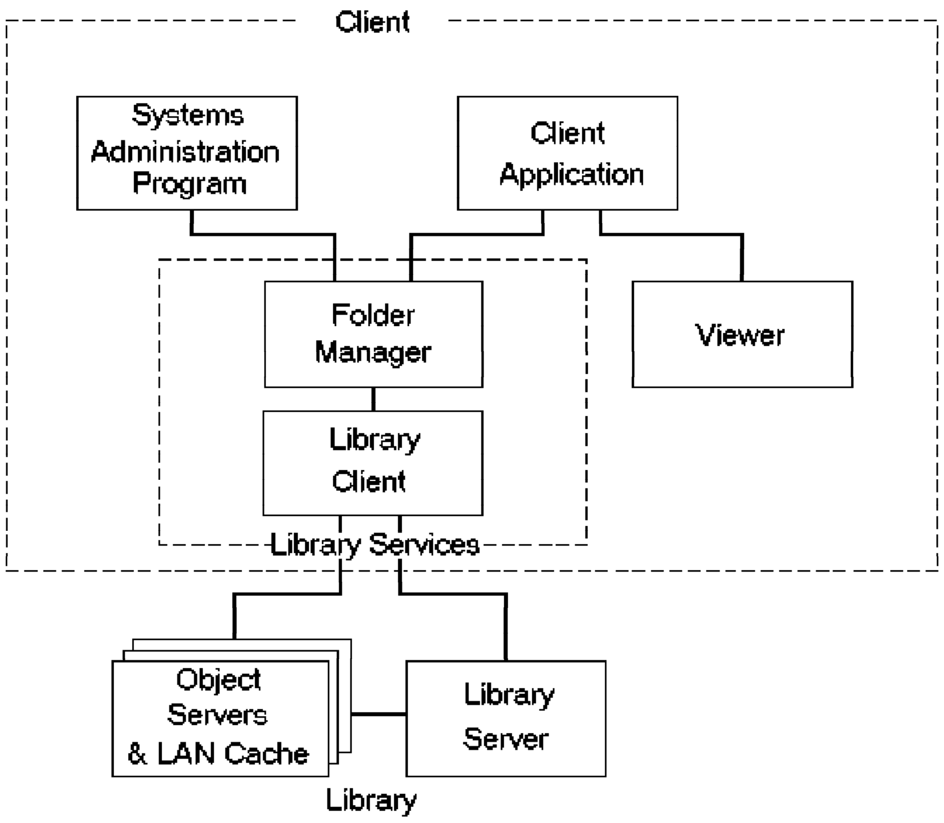


Figure 3. Client Components

1.4.1 Folder Manager

The folder manager provides a document management programming interface that uses the library client to access information stored on the library server and object servers.

The folder manager provides a data model for managing online documents and folders much as one would with paper documents. It submits API requests to the library client to perform library functions for the client application. The library client then interacts with the library server and the object server independent of the platform that the servers are running on. That way, you can expand the VisualInfo system without having to change the applications.

Your system can access these folder manager services through the client application. It is possible also to interface directly with the VisualInfo system through the folder manager APIs to create a customized document-management, workflow, or other business application.

Depending on the level of access to documents, you can perform the following operations using the folder manager:

- Store a document
- Index a document or folder
- Process a document or folder
- Retrieve a document or folder

1.4.2 Library Client

As an alternative to using the folder manager, the library client (a lower-level interface that allows the flexibility to create a custom data model) can be used. It lets applications directly connect with the library server and object servers in a VisualInfo system through APIs. These library API calls are used if you decide not to use the folder manager, or if you want to mix the two sets of APIs in a customized application. Compared to folder manager APIs, the library client APIs provide access to a more detailed level of underlying product functions. In certain cases, such as to take advantage of different performance characteristics, it is necessary to use the library client APIs for services not available through the folder manager APIs.

1.4.3 Client Applications

The VisualInfo Windows client applications provide a complete image- and document-processing system that can be customized to meet the needs of the enterprise. The client applications interact with all the VisualInfo components through an end-user interface. The client applications do not require additional programming. They integrate the features of the VisualInfo system, including the folder manager services, into one application. They can, however, be extended through user exits.

The client applications allow you to input documents by scanning or importing files. Then, documents and the folders containing the documents can be managed in an online file room. Through the search facilities of the client applications, you can find documents and folders and start them into a desired workflow. By retrieving documents from the object server, you can browse, display, print, or export them into files onto local disks.

The client applications provide APIs to integrate folder, workflow, and document management with your existing information systems. Custom software and other applications can easily be integrated.

1.4.4 System Administration Programs

The system administration programs are powerful end-user interfaces that let you configure the VisualInfo system. One of these is Java-based and can reside anywhere in your VisualInfo network, provided you have installed the latest CSD. The other runs on an NT or OS/2 workstation. It is used to configure resources such as user, groups, privilege sets, access control lists, index classes, work baskets and workflow.

1.5 Understanding VisualInfo Communication Flow

A VisualInfo domain consists of a single library server and its connected object servers and clients. An object server communicates with only one library server. The communication between these three components occurs in a *triangle* hierarchy. The communication flow between client workstations, their assigned library server, and the available object servers follows a sequence that is shown in Figure 4.

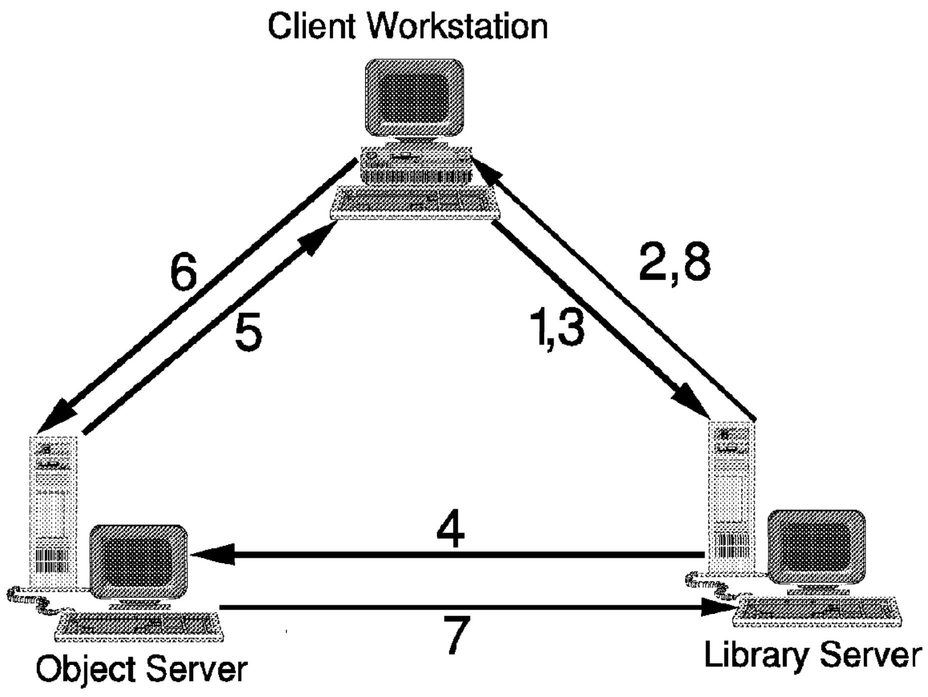


Figure 4. VisualInfo Communications Flow

1. The client workstation does an open object (document) request. It looks for the object in its local cache (OS/2 client only) and if found, passes the timestamp of the object in the cache in the request to the assigned library server. This process is done to allow the client workstation to check for a valid copy of the object stored in its workstation local cache.
2. The library server receives the request and validates the timestamp of the object passed from the workstation client. If the workstation cache copy is valid, the library returns a response and the process finishes here.
3. If the object is not found in the local cache, the client workstation sends a notification to the library server saying that it does not have the object.
4. The library server sends an object request to the appropriate object server that owns the object. The client workstation ID that is requesting the object is also sent.
5. The object server initiates the communication link with the client workstation and sends the object.
6. The client workstation notifies the object server that the object was received successfully.
7. The object server notifies the library server that the operation was completed successfully.
8. The library server finishes the object request process.
9. A client application opens the object and the user is able to view the object.

Chapter 2. Digital Library Architecture—an Overview

IBM has recognized the importance of protecting valuable assets such as rare manuscripts and scrolls, large photo and high-quality image collections and audio-visual material such as recordings and films.

To address this requirement, IBM developed the Digital Library product, which evolved from the digital document management system, VisualInfo. You will notice a similarity to VisualInfo, as described in the previous chapter. But the aim of Digital Library is broader than just document management. Instead, it enables users to manage large amounts of information and many types of media, to conduct searches that would otherwise be impossible, and to perform rapid information distribution and retrieval.

IBM′s Digital Library is a comprehensive solutions framework that addresses the important needs of large organizations. It maximizes the value of multimedia assets and allows quicker and more refined access. It is built on top of an RDBMS and provides a completely integrated service framework that combines all the required functions of multimedia information capture, storage, management, search, retrieval and distribution into a single architecture; see Figure 5.

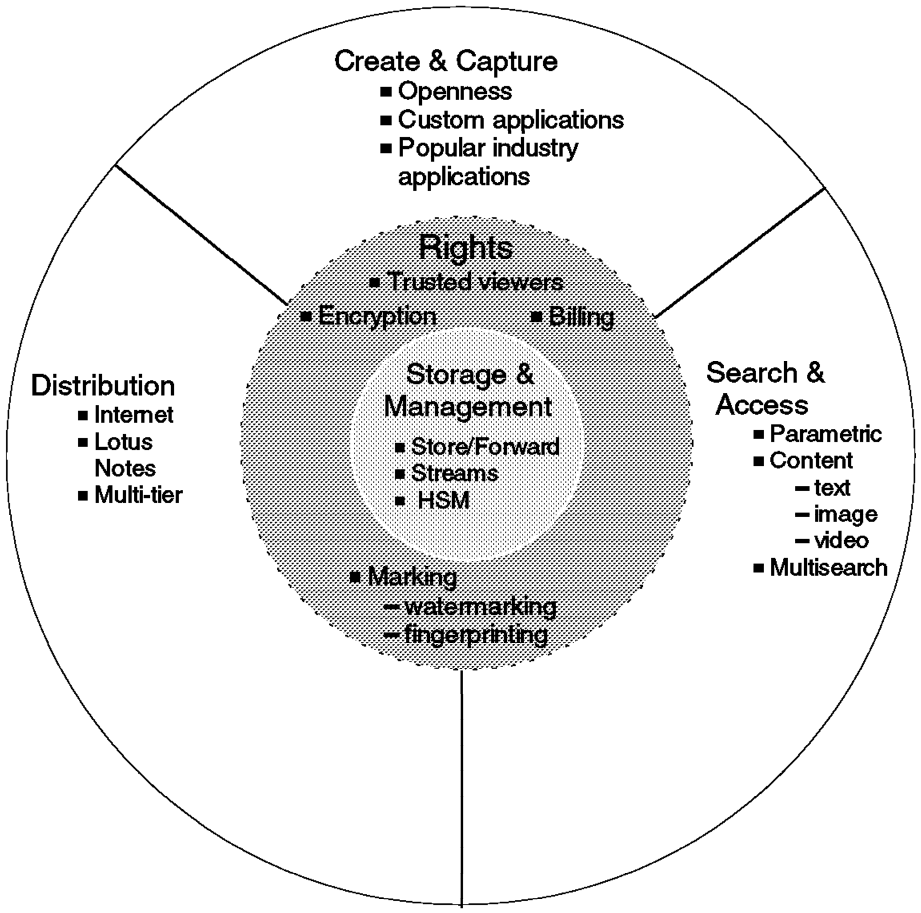


Figure 5. Digital Library Key Functions

2.1 Digital Library Functions

To enable the collection, organization, management, protection, and distribution of content, a Digital Library must address many key functions. The following five basic functions form the foundation of IBM′s Digital Library product, as illustrated in Figure 5 on page 13.

1. Create and Capture

   This is the process of converting conventional assets to digital form and loading them into Digital Library. This should address the conversion of scanned images to appropriate formats with compression where necessary,

2. Storage and Management

   This includes the creation of thumbnails, if required, and the import of legacy metadata into Digital Library.

   This is the process of organizing and maintaining digital assets for availability to users whenever required. It provides an open strategy that supports a variety of platforms, database technologies, and operating systems. The solution may be scaled with the growth of the collection.

3. Search and Access

   With vast quantities of information being stored, it is imperative that users be able to find what they need quickly and efficiently. With this function, parametric searching and free text searching is available for image, video and text collections. It allows the user to access data from a variety of client workstations with a user interface that can be adapted as new applications for Digital Library are added.

4. Distribution

   This function deals with delivering content to the users. IBM Digital Library distributes information through any existing network, including client/server environments, commercial online services, the Internet, and corporate intranets. Data distribution is supported using two paradigms: The *push-model* where data is streamed to the client workstation for playback, and the *pull-model*, where a replica of the data object is copied onto the client′s local disk.

5. Rights Management

   This function of Digital Library helps protect the valuable intellectual property contained in the library from unauthorized access and abuse. It overcomes the old paradigm of protecting assets by making them less accessible, and affords greater access without increased risk.

2.2 Digital Library Extended Architecture

Departing from the base VisualInfo client-server architectural model, the extended architecture of Digital Library enhances and expands all components of the base architecture.

- The library server basic parametric search capabilities are associated and combined with free-text and image search engines (TextMiner and QBIC).
- To the traditional store-and-forward data distribution model of the object servers has been added the new data streaming model of the audio-video server.

- A complete new middle-tier server called Grand Portal, has been introduced to support multi-search facilities to Web-enabled clients accessing Digital Library.

The core of Digital Library is the newly enhanced library server, which manages the catalog information (metadata), locates stored objects using a variety of advanced search technologies, provides secured access to the objects held in the collection, and communicates with the various object servers. The digital assets themselves are stored in one or more object servers which can be distributed across an enterprise network to provide user access. Regardless of the implementation, digital assets are transferred directly from the object server to the client, completing the final side of the triangular architecture.

The client, middle-tier server, library server and object server can be implemented across several machines and platforms, or these functions can all run on one platform. The best architecture for any solution is based on a variety of factors including the location, number, and nature of the users accessing the system.

2.2.1 Library Server

Conceptually, all information about data or metadata is stored in the library server. Thus the conventional VisualInfo library server + text server + image server can now be called the library server.

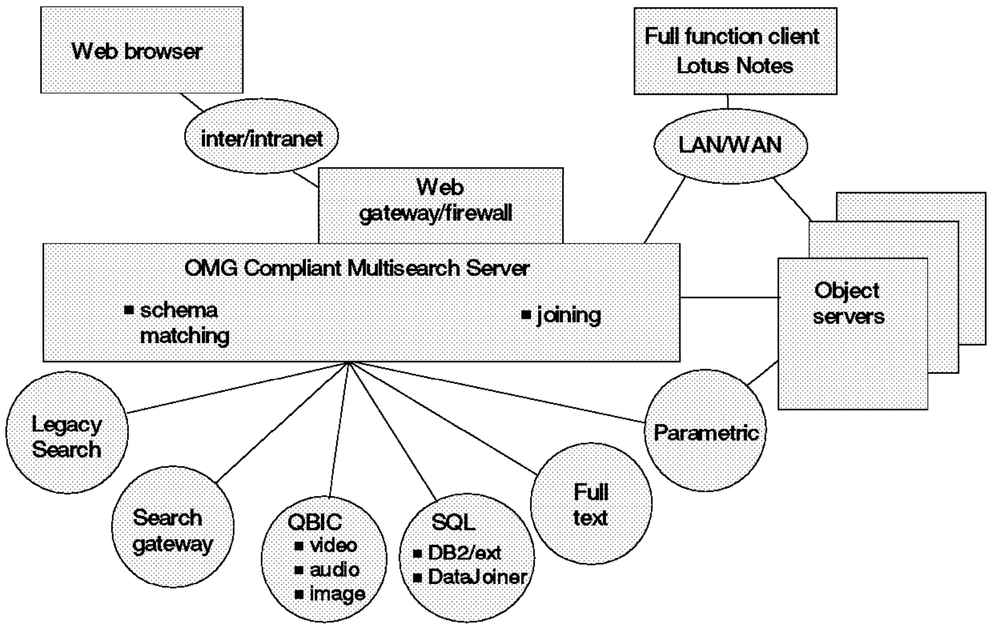


Figure 6. Digital Library Extended Architecture

Physically, this library server has extensions. The conventional library server, as we know it from VisualInfo, contains metadata, such that it only supports parametric searches. The new extended library server stores metadata for free-text searching in the text servers, and for image searching, into image servers.

This is the heart of Digital Library; all system- and user-defined metadata, as well as the access control to the digital assets, are managed here. Only one library server is required in any Digital Library solution. The library server works with either Oracle or DB2 on AIX and Windows NT.

2.2.2 Text Search Server

This is an optional component that runs alongside the library server and provides very advanced free-text search capabilities. It can be combined with parametric searching to provide powerful combined search capabilities.

Text search allows you to automatically index, search, and retrieve documents based on a full text search. Users can locate documents by searching for words or phrases, abbreviations or acronyms and even proper names.

In a typical LAN environment, a text search installation comprises of one or more servers and several clients. The text search server program can be installed on a machine together with other Digital Library components. The text search client resides on client workstations and provides access to the server. Text search supports both single-byte and double-byte character sets, and runs on AIX, Windows 95 and NT.

In addition to the server and client components, text search uses dictionaries to support the linguistic processing of documents in different languages during indexing and retrieval. Dictionaries must be installed on the server and at each client workstation, to achieve satisfactory performance. Dictionaries installed at the client workstation are used for extended matching when applications retrieve the matches of a previous search for later highlighting. Dictionaries installed at the server workstation are used by the search service while indexing and searching.

Digital Library support for multi-search facilities can be used for the following:

Combined query	Search within a given datastore, using one or a combination of supported query types.
Parametric query	Queries requiring an exact match on the condition specified in the query predicate and the data values stored in the datastore.
Text query	Queries on the content of text fields for approximate match with the given text search expression; for example, the existence (or non-existence) of certain phrases or word-stems. Each search type is supported by one or more search-engines.
Query refinement	Search on the results of previous search.

2.2.3 Image Search Server

Query By Image Content (QBIC) is a technology that allows image searching based on image color, texture, and layout characteristics. For example, QBIC searching would allow a user to select a photo (for example, a white farm house with green grass and a large dark tree on one side) and search for all images with similar color layouts or compositions. This search would likely return several similar houses with large trees nearby.

2.2.3.1 QBIC Overview

The QBIC server or image server provides the capability to query collections of images by content including such properties as average color, color percentages, color layout, and texture occurring in the images. This capability of performing queries by image content complements traditional queries which use image filenames as the basis of queries. The QBIC server is a search engine for *digital imagery*. QBIC does not actually store the images; it merely indexes the images stored in Digital Library to support searches on those images.

The use of QBIC server involves two major steps:

1. Catalog creation: A preprocessing step to compute numeric features and store this metadata in a QBIC catalog.
2. Image query: A run-time step that uses the computed features to find images similar to a query specification.

QBIC indexes are organized in a two-level structure, QBIC database and QBIC catalog. This two-level structure provides a flexible way for organizing QBIC indexes. A QBIC server has one or more QBIC databases. Each QBIC database is a named collection of QBIC catalogs. A QBIC catalog holds the data about the visual features of images. The example in Figure 7 shows a QBIC server with two databases and two catalogs per database. You create a QBIC catalog for images that you want to make available for searching by content.

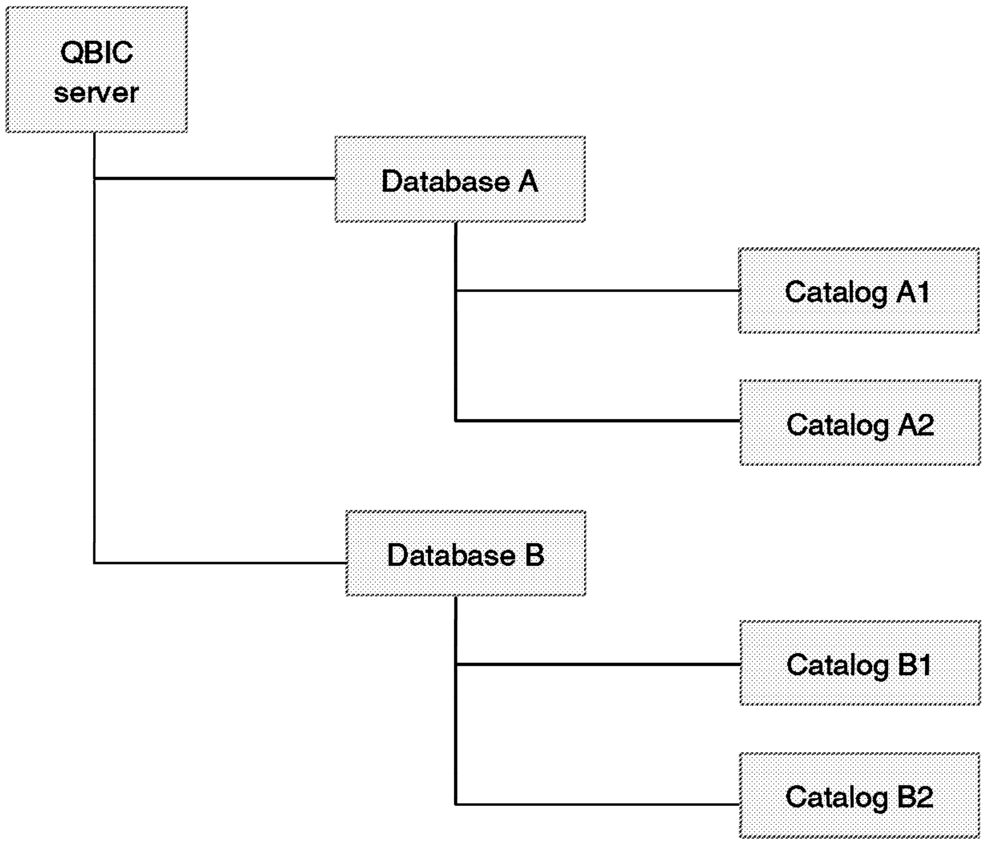


Figure 7. QBIC Server with Databases and Catalogs

When you create a QBIC catalog, you identify the features for which you want the QBIC server to analyze, store, and later query data. You can also add or drop features from a QBIC catalog after the catalog is created.

2.2.3.2 Features of QBIC

A QBIC database is a named collection of catalogs. A QBIC catalog holds data about the visual features of images. QBIC features are:

- *Average color* is the sum of the color values for all pixels in an image/the number of pixels in the image.
- *Histogram color* measures the percentages of color distribution of an image.
- *Positional color* is the average color value for the pixels in a specified area in an image.
- *Texture* refers to the average texture of an image. It measures the coarseness, contrast, and directionality of an image. *Coarseness* indicates the size of repeating items in an image. *Contrast* identifies the brightness variations in an image. *Directionality* indicates whether a direction predominates in an image.

To make an image available for searching by content, you catalog the image. When you *catalog* an image, the QBIC server analyzes the image by computing the feature values for the image, and stores the values in a QBIC catalog.

When you search for an image by content against a specific QBIC catalog as the target, your query identifies a feature for the search (such as average color), and the value of the feature (such as RGB values or an example image) as the source. The QBIC server computes the feature value of the source and compares it to the cataloged feature values for the target images. It then computes a score that indicates how similar the feature values of the target images are to the source. The QBIC server returns the images whose features are most similar to the source.

2.2.4 Object Server

Both the library server and the object server manage the collection items and assure referential integrity between the catalog data and the digital media content itself. Every Digital Library solution has one or more object servers. The object server can reside on the same server machine as the library server. Object servers are available for AIX, NT, OS/2 and MVS.

Users store and retrieve digital objects in the object server through requests routed by the library server. It is the repository for objects stored in the system. The object server efficiently and automatically manages storage resources, based on the storage management entities (such as volumes) that are defined through the Digital Library system administration program. The object server supports attachment of DASD, ADSM, and other storage devices. A Digital Library system can have many object servers distributed across networks to provide user access. Object servers run on AIX and Windows NT.

They can use IBM DB2 UDB (or the previous version of DB2, which is also supported) or the Oracle database to manage the library contents. A Digital Library system has one or more object servers. The object server also works with the System Administration program to efficiently manage storage resources. This function permits the system administrator to specify how long documents reside on one media type before migrating them to another. After the system administrator defines migration policies, the object server manages storage automatically. Object servers can use IBM DB2 UDB (or the earlier version of DB2) or the Oracle database to manage their contents.

2.2.5 Digital Library Systems Management

Digital Library has a complete suite of database, user, and system administration tools. Some of the features supported by the Digital Library management system are:

- Server monitoring and control (library server, object servers, text search server)
- User management (access privileges and passwords)
- Network communications administration (for DL entities)
- Object storage management strategies for each object server
- Database administration and optimization

2.2.6 Hierarchical Storage Management

Usually when dealing with large collections of data the use of a hierarchy of storage devices is advisable, to move most the data out to the most convenient storage medium, which in most cases is the magnetic tape. In those cases, IBM ADSM must be used for auxiliary device storage and system backups.

2.2.7 Digital Library Middle-Tier Server (SDK)

Clients can consist of custom-developed Java applications, compiled C++ programs, or HTML Web-based user interfaces. The current trend is to have a Java or HTML-based Web interface that can be accessed from any client platform that can run a Web browser. The Base Digital Library product includes a native Windows client, along with HTML and Java sample client applications.

Note: This information is included as an architectural overview. However, at the time of publication, performance and capacity information were not available and are not addressed in this redbook.

The client toolkit consists of:

- Client APIs
- Rights management (Visible and Invisible Watermarking, the Cryptolope technology)
- Object-oriented (OO) APIs and Java Applets and the Internet Application Toolkit
- Dynamic Page Builder (with Net.Data)
- Client for Windows

The Digital Library client toolkits may be used to build Internet or desktop client applications that access the data and objects managed by Digital Library. The Client for Windows, also part of the client toolkit, provides a user interface to the Digital Library system and its contents from Windows desktops. The client toolkit may also be used to develop Web-based and desktop applications that combine object catalog and text searches in a single query. A Java applet that comes with Digital Library demonstrates multi-search capability. APIs are provided for Java, C++, C, and ActiveX, and sample applications are included with Digital Library. Client applications can be developed for a variety of platforms, including AIX, Windows NT, Windows 95 and Macintosh. (On AIX and Macintosh, only the client APIs are available.)

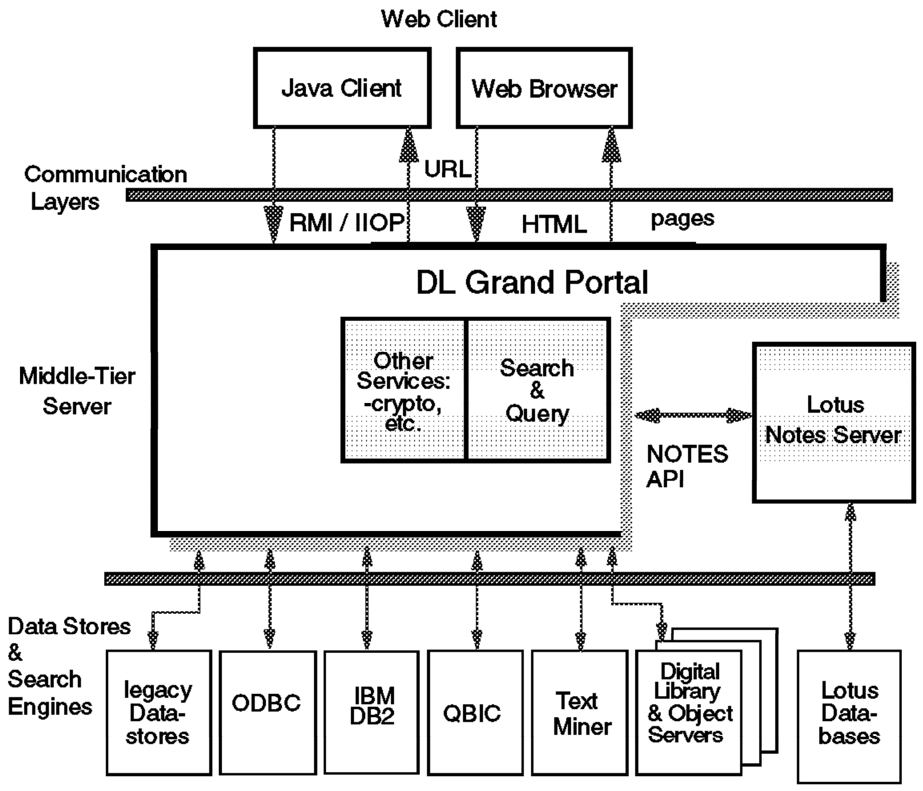


Figure 8. IBM Digital Library

The Grand Portal Gateway is used to perform the following functions in Digital Library where access occurs via browsers (either Netscape or Internet Explorer 3.0 or later) and through TCP/IP. A Web server (Apache, Netscape FastTrack, Sun JavaServer, Microsoft IIS, Lotus GO Webserver) receives an HTTP request for a search of a document or an object, translates the HTTP request into Digital Library-compatible requests, extracts the desired objects from the Digital Library library server and object server, adds any HTML information required and provides the document (or object) with appropriate HTTP to the requester.

Using a Web Macro Processor such as Net.Data, it is simple to write interactive Web applications by using macros to add logic, variables, program calls and dynamic reports to HTML. Net.Data macros are coded in a proprietary Web macro language along with HTML. Net.Data runs on the Web server as a common gateway interface (CGI) program. The URL that invokes it also indicates which Net.Data macro to process. When Net.Data finishes processing the macro, the resulting HTML is sent to the Web server and is passed on to the Web client, where it is displayed on the browser. The Live Connection feature of Net.Data is used to keep persistent database connections in order to improve user request throughput.

2.3 VideoCharger Server Key Features

Note: This information is included as an architectural overview. However, at the time of publication, performance and capacity information were not available and are not addressed in this redbook.

There are three parts needed to make Digital Library and VideoCharger work together, namely the VideoCharger, the "middleware code" which we call Media Manager and a DL object server enabled for video. VideoCharger Version 2.0 is available on AIX and NT. The Media Manager middleware code is packaged together with VideoCharger and runs on AIX or NT. Video-enabled NT and AIX object servers will ship with the next Digital Library release. With this new release, the need for a special object server will be dropped; all DL object servers will be enabled for video without the extra cost.

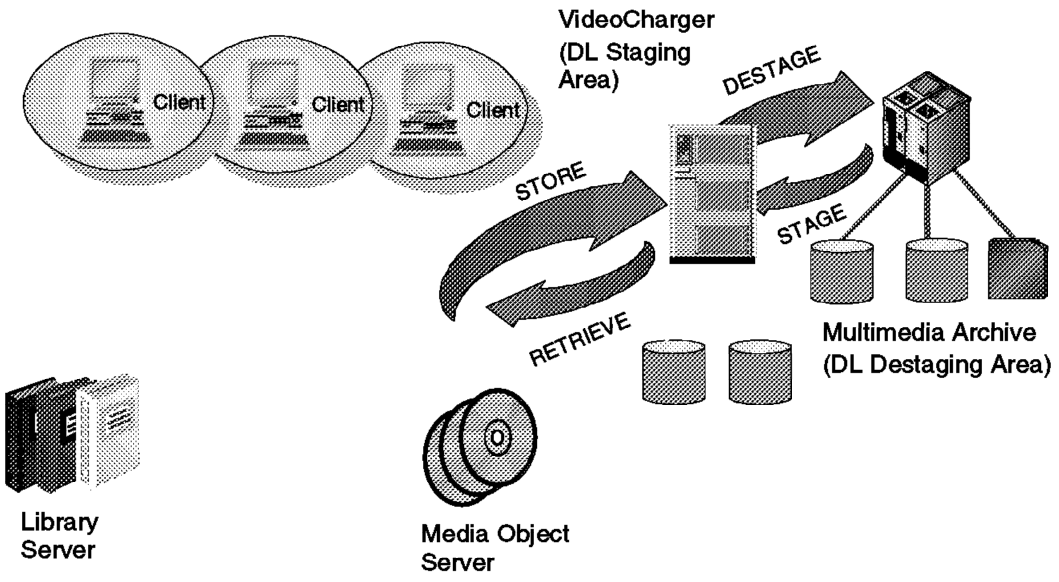


Figure 9. VideoCharger System

IBM VideoCharger is a solution for the delivery of continuous time media (audio and video) to Internet- or intranet-connected clients. The video is ″streamed″ (delivered in real-time) and therefore does not require that the file be downloaded and saved before being played by the client software. In video parlance, the video is ″pushed″ by the server over the network to the client. This approach is different from most file servers today where the data is ″pulled″ by the client issuing successive reads to the server. The ″push″ architecture is similar to a broadcast environment where a video stream is started by a play command and will continue until stopped.

IBM VideoCharger provides the following features.

2.3.1 Scalable Servers

The IBM VideoCharger on NT offers a simple, single-box solution with easy installation and configuration. While not as scalable as the VideoCharger on the AIX platform, it offers all the same functional capabilities. Also, VideoCharger on NT offers embedded encoder support, allowing for real-time encoding with IP multicast; see 2.3.4, "IP Multicast and Real-Time Encoding" on page 23.

The AIX solution can grow from a low-end server through a high-end parallel processor. VideoCharger allows you to expand to video server complexes by clustering RS/6000 processors as your demand for video streams increases. Support for both small computer system interface (SCSI) and serial storage architecture (SSA) devices is provided. SSA offers higher data throughput (80 MBps) and built-in fault tolerance, and expands up to 873 GB for each SSA adapter.

2.3.2 Support for Many Media Formats

IBM VideoCharger offers a solution for both the Internet and intranet environments. For Internet clients, especially the home Internet users which are typically connected via slower network connections, VideoCharger will support the delivery of low bit rate (LBR) video.

LBR video is based on industry standard H.263 video and G.723 audio from the video conferencing industry. This technology allows audio and video to be served to home Internet users connected with 28.8 Kb modems. Using approximately 16 Kb/s, the LBR video offers 8 KHz 16-bit PCM audio and 160X120 video at 7.5 frames per second. The LBR video can be encoded at higher quality rates to provide higher resolution or more frames per second for those clients which are connected via ISDN modems, cable modems or an intranet network.

For the intranet environment, VideoCharger provides support for higher quality (and higher bit rate) videos. In this environment, both MPEG-1 and MPEG-2 content are supported at rates up to 8 Mb/s. Near CD quality single-channel audio is supported at 24 KHz. This flexible support for higher quality video allows a multitude of applications to be enhanced with video in the intranet environment.

VideoCharger includes trancoders that can convert either Windows AVI or WAV files into low bit rate format. The compressed data is stored in a unique file format (.IBA) optimized for UDP protocol.

2.3.3 Support for Industry Standard Protocols

In addition to supporting industry standard file formats for audio and video, VideoCharger supports the popular Internet and World Wide Web (WWW) protocols including IP and HyperText Transport Protocol (HTTP). This allows the product to be used with industry standard applications such as HTML Web browsers. It also allows the product to be used on a wide variety of network types including LANs (local area networks such as Ethernet, Token-Ring, FDDI), WANs (wide area networks such as T1, E1, T3, E3) and Asynchronous Transfer Mode (ATM).

VideoCharger also uses the latest Internet protocols. These include:

- TCP protocol for content-serving through firewalls
- Real-time Transport Protocol (RTP) for the delivery of the media stream
- ReSerVation Protocol (RSVP) for acquiring Quality of Service (QoS) on the network (in AIX server only)
- Real-time Streaming Protocol (RTSP), a communications protocol for control and delivery of real-time media over the Internet
- Path MTU Discovery

- IP multicast for broadcasting audio or video.

The use of these protocols allows the solution to be as efficient as possible for delivering video in today′s network infrastructure. They require no changes to the existing infrastructure, but will take advantage of new hardware (such as RSVP-enabled routers) as they are added to the network.

2.3.4 IP Multicast and Real-Time Encoding

The IP multicast feature, in particular, allows VideoCharger to be used as a broadcast-type server in the Internet environment. This allows a single audio or video stream to be sent to multiple people, thus reducing the bandwidth requirements on the network.

In addition, VideoCharger on Windows NT offers embedded encoder support. This allows an MPEG encoder to be installed on the server and have VideoCharger directly control the encoder for functions such as real-time IP Multicast and real-time IP Multicast with live recording of the same stream. This is a very efficient yet powerful method of providing a broadcast of a live event, while recording it for later re-broadcast with minimal network load.

2.3.5 Multimedia File System

At the heart of the VideoCharger Server for AIX is a higher performance multimedia file system. The file system incorporates techniques such as wide striping, real-time disk scheduling, large block-size transfers, data replication, and automatic disk calibration to provide reliable delivery of audio and video data. On Windows NT, your multimedia assets are placed in the NT File System (NTFS) for storage and retrieval. NTFS provides options for configuring multiple disks into a striping group, thus taking advantage of all the performance benefits of striping.

2.3.6 Multimedia Archive

The IBM 3466 Network Storage Manager can act as a video/multimedia archive for storage management and recall of video objects produced by VideoCharger. As a video archive, the facilities of ADSM are made available for management of large, medium, and small video objects. The IBM 3466 can store over 13 TB of uncompressed data. Together with communication options such as ATM, FDDI, Ethernet 10-100 Mb/s and Token-Ring and automated tape handling, the IBM 3466 can provide a combination of broad archive flexibility, vast storage access and high availability of video assets.

2.3.7 Content and System Management

IBM VideoCharger comes with a comprehensive Web-based administration and configuration facility with the capability of content and system management including loading, searches, and queries. The combination of VideoCharger with IBM Digital Library Version 2 provides an integrated digital media management system for all media assets (image and text, and audio and video). IBM Digital Library addresses many of the issues relating to storage management, search and access, and rights management.

Chapter 3. Overview of Other Architectures

Let us take a look at two other architectures: the VisualInfo system for the AS/400 and the ContentConnect product.

3.1 VisualInfo for AS/400 Server Architecture

VisualInfo for AS/400 architecture is the same as the classic VisualInfo model. A domain consists of a single library server and its connected objects servers and clients.

3.1.1 Understanding the VisualInfo Model

The physical implementation provides for the library server and the primary object server to reside on the same AS/400. Additional secondary AS/400 object servers can be included in the domain and, like classic VisualInfo architecture, the secondary servers communicate only with one library server; see Figure 10.

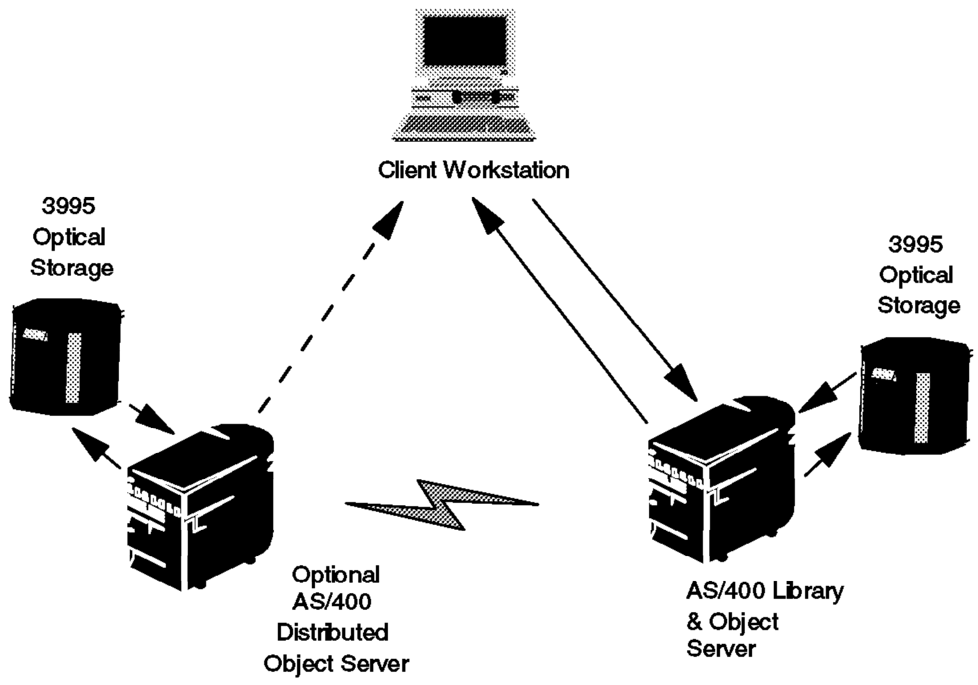

Client Workstation
3995 Optical Storage
3995 Optical Storage
Optional AS/400 Distributed Object Server
AS/400 Library & Object Server

Figure 10. VisualInfo for AS/400 Topology

3.1.1.1 Library Server

The AS/400 library server runs DB2/400 as the database, which is included in the operating system. All client requests are processed by the library server, which contains the VisualInfo application database and the program objects. These program objects communicate with the object servers and manage the SMS functions for VisualInfo.

3.1.1.2 Object Server

The primary object server is installed on the same AS/400 server as the library server. The object server data base has a choice of file systems to use for storage of the image files. Originally, the Document Library file system (QDLS) was the only file system choice for image storage. However, the Integrated File System (IFS) has become the preferred file system for image files, due mainly to performance and the strategic nature of this file system within the AS/400 architecture.

3.1.1.3 Staging and De-staging in VI/400

Because of the integrated nature of the VisualInfo for AS/400 architecture, staging and de-staging are performed as part of the library server functions. The staging area is combined with the permanent disk storage (DASD) area. When an object is stored in VisualInfo, the object is written to the AS/400 DASD. Based on the VI configuration, a request may be made to copy the object to optical storage, and a request generated to remove the object from DASD after the specified length of time.

A library server function will copy the selected objects to optical. Another library server function will remove the object from AS/400 DASD when the specified interval has expired.

When a client requests an object that currently resides on DASD, it is displayed from DASD directly, because there is no separate staging area. If the object resides only on optical, it is fetched to DASD (staged) and then displayed. If the document has a limited life on DASD, based on the VI configuration, a request is generated to remove the object from DASD after the specified length of time.

This staging/de-staging function can occur many times during the life of the object. For performance reasons, it is desirable to have the document reside on DASD when it is most frequently requested.

3.1.1.4 Optical Object Servers

The optical object servers included in the architecture can either be attached directly to the AS/400 (C4X models), or through the local area network (C2X models). The directly attached optical server hardware is managed by the AS/400 object server and the OS/400 operating system. The LAN-attached server hardware is managed by the library server and the LAN controller operating system.

3.1.2 AS/400 Architecture Differences

While we will not discuss the AS/400 platform architecture in detail, there are several concepts which are different and should be mentioned.

3.1.2.1 Memory Management

The AS/400 architecture uses a single-level storage concept which treats all storage as one large memory space. A machine component determines whether the referenced memory address is in real memory or auxiliary storage (DASD), and it swaps pages into and out of memory as required. The application sees only an unlimited amount of memory.

The need to allocate private memory for an application or process is eliminated in most cases, because the operating system lets each program work as though it has unlimited memory. In situations where a file is accessed very frequently, you can choose to have it cached for better performance.

Allocating ″swapspace″ is not necessary on an AS/400 system, because the single-level storage architecture lets the operating system manage memory and disk as a single entity.

3.1.2.2 Database Management

DB2/400 is not a separate program product, but is built into the operating system. Therefore, all tables on the AS/400 are part of the database, and there is no need to import or export tables.

Management of table or file space is not needed because the AS/400 handles this by dynamically extending file size as needed.

Balancing disk space is performed by the AS/400 operating system. The storage manager will distribute files evenly across all disk arms within the storage pool.

While there are some database administrative functions, normally there is no management of the database required.

3.2 ContentConnect Architecture

The ContentConnect product plays an important role in IBM′s enterprise document management solutions. It provides a ″federated search″ capability across multiple document repositories, without the need for common database architectures, and performs the searches without programming. See Figure 11 for the general topology of the product.

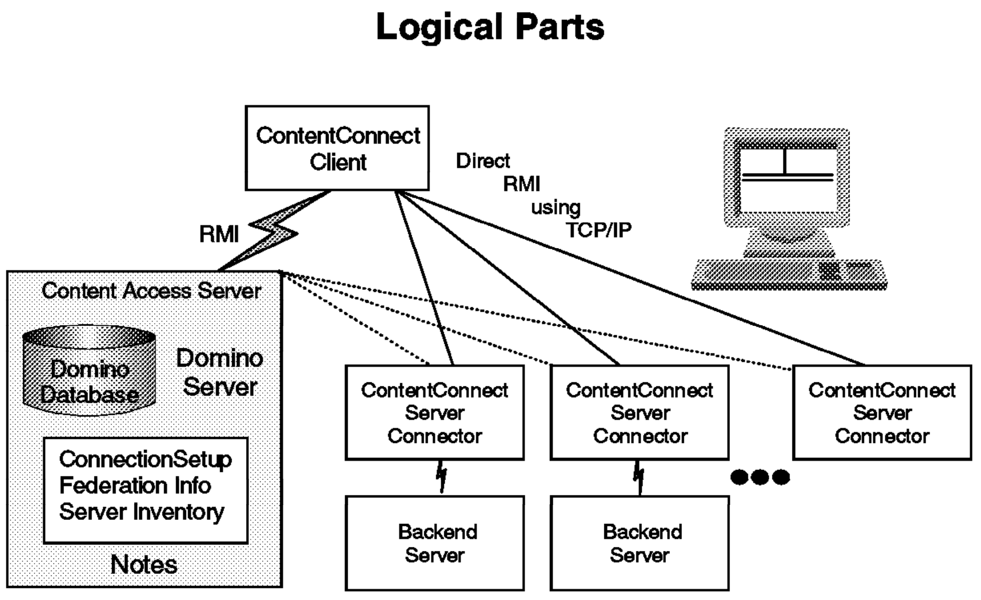


Figure 11. ContentConnect General Topology

The ContentConnect architecture is key to its ability to federate (combine results of searches into meaningful results) searches. ContentConnect is written in

Java, and consists of a client, an access server and one or more server connectors.

Currently, ConnectConnect searches the following server backends:

- VisualInfo
- ImagePlus for OS/390
- VisualInfo for AS/400
- OnDemand
- Domino.Doc

3.2.1 ContentConnect Components

The components of ContentConnect are described in the following sections.

3.2.1.1 The Client

The ContentConnect client comes in several ″flavors″, made possible by Java development. The client is shipped as a standalone client, running in a Java Virtual Machine (JVM), as an applet running in a browser, or from a Notes client. The current version of the client does not include a full Java viewer, so the VI and OnDemand viewers are shipped and installed with the product.

3.2.1.2 The Access Server

The Access Server is the ″heart″ of the ContentConnect product, even though it is used at definition time, and once for each logon. Because of this design, this server does not become a bottleneck in performance.

The Access Server consists of three components:

- Setup database (a Notes application)
- Server Inventory program
- Federation Info program

It is not a gateway, therefore logons to the document servers do not have to go through this server. The Access Server is simply used to register backend servers to search, and as a place to store the search templates and mapping information.

The Setup database is a Notes application which allows a non-programmer to define which servers are to be accessed, also mapping the disparate index fields in each backend database to a common display definition.

The server inventory program is run once, at initial setup time or whenever the backend database changes, to update the setup data base. The server inventory program logs on to each backend and builds index class/keyfield inventory for the application, and stores the information in the Notes application database.

The Administration function creates the search templates which map each column in the template to a data element in the backend data base. This architecture allows the connection of almost any backend repository of data, without the programming other federated searches require.

The ContentConnect client connects to the Access Server at first logon, to get the templates and server list to which the user is authorized. This is then stored on the client for the session, or until the client requests a refresh.

3.2.1.3 Server Pools

Server Pools provide an alternative to the direct connection of each client to each backend server. The server pool architecture serves as middle-tier server complex, and eliminates the need to install a server application programming interface (API) set on each client machine. With the server pool approach, the API set resides on the middle tier, providing easier software management and a thinner client; see Figure 12. Server pools communicate with Remote Method Invocation (RMI).

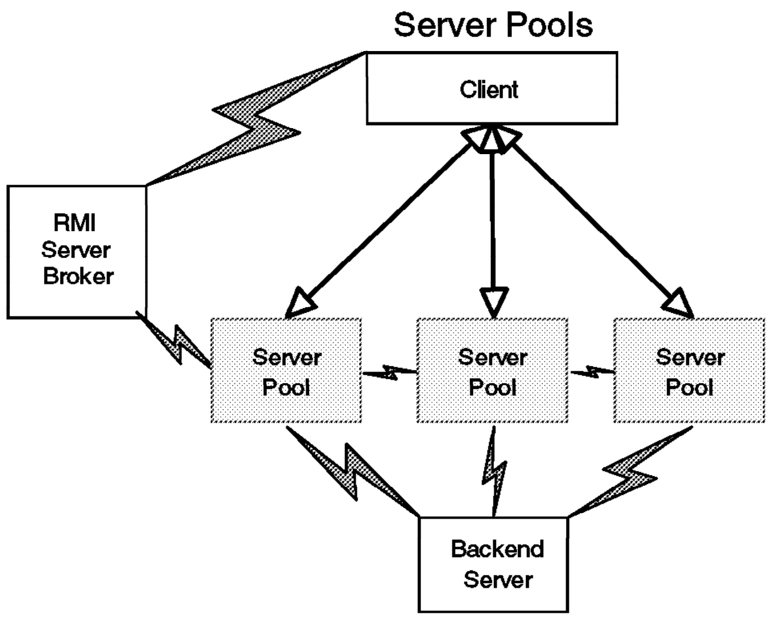


Figure 12. Server Pool Topology

As you can see from Figure 12, there are two server pool components:

- Broker Server
- Server Pool members

The Broker Server is the main server provider; there is only one in the pool. This server makes itself available via the RMI Registry, and clients connect to the Broker Server. The broker obtains server connections from pool members and returns a connection back to each client.

The Broker Server provides a server connection only if pool members cannot (for example, if the connection is busy). The client receives the server connection and communicates with it.

The Server Pool Member registers with the Broker server instead of RMI. There can be many pool members, and these pool members only communicate with the broker server. The broker will request a server connection for the client. The server pool member will return the server connection to broker. Once the connection is returned, the client only communicates with the server connection that was returned.

3.2.1.4 Server Connectors

Server Connectors are the bridge between the ContentConnect client and the backend repositories of documents. The connector makes possible the connections to the various databases, and each connector is unique for the backend repository to which it connects.

The server connectors serve as interfaces between the client and the client API set that is used to search the repository. In VisualInfo, this is the VI client API set. If the connections are made using RMI and the server pool connections, the server connectors serve as an interface between the RMI Server Connection and the client API set.

The server connectors can be installed on each client, or in RMI, on the broker server.

3.2.1.5 Performance Considerations

There is a great deal of flexibility in how you connect your clients to the backend servers, and with this flexibility comes performance trade-offs. We review performance implications in Chapter 13, “VisualInfo for AS/400 Performance Tuning” on page 153.

Part 2. Capacity Planning

- How to perform capacity planning
- Document management considerations
- Specific platforms
- Digital Library scenarios

Chapter 4. How to Plan for Capacity

The objective of carrying out a capacity planning project is to accurately estimate the current workload and its rate of growth to help the customer in planning and purchasing decisions for a VisualInfo system that meets current and future business requirements. A successful capacity planning project lies in using the following steps:

- Understanding the current environment
- Characterizing the workload
- Estimating storage requirements
- Performing benchmark measurements
- Forecasting future workload and system performance

Planning a VisualInfo or Digital Library System can be a difficult task, with many things to consider. A company should know how they are currently processing their documents and data, and they may have some idea of what they want to accomplish with an imaging system. The task is to develop a model that best represents their data flow, using objects, documents, and folders. You will need to understand their business process and user needs well enough to include in that model how best to index the content for efficient searching, retrieving and routing.

To automate some of the capacity planning process, a tool called VISizer is available to IBM representatives. It is written and maintained by the IBM Image and Digital Library Competency Center in Gaithersburg, MD. IBM employees can retrieve it from this Web site:

```
http://compcntr.washington.ibm.com/tools.htm
```

Of course, use of the VISizer should not be done in a vacuum. Sufficient analysis before and after a VISizer exercise needs to be done to ensure that the results match the real-world environment you are planning for. Therefore, the author of the VISizer has included this chapter in the redbook to let you prepare for use of this tool. The questions that are asked in the VISizer have been put into words to explain the philosophy and flow of the tool.

4.1 Understanding the Current Environment

A thorough understanding of the current environment is a prerequisite for any capacity planning. At minimum, the following items should be analyzed:

- Document types and counts

  Documents should be grouped and categorized by type (for example, invoices or claims). Also, scanning resolution and the resulting file size must be known.

- Document arrival rate

  When, how, and where documents enter the system needs to be analyzed. While annual document volumes are often used as a benchmark, it is dangerous to apply these statistics without knowledge of document arrival peaks and valleys. When analyzing asymmetric document arrival patterns, it is most critical to identify the peak arrival rate and size the system accordingly.

- Document usage

  When, how, and where each kind of document is used must also be well understood. For example, a document may go through a business process by following a workflow or by stepping through VisualInfo workbaskets. At each workbasket, this document is viewed and perhaps 40% of the time, annotations may need to be added to it. This example may require more system resources than a different example where the volumes are the same but fewer views and annotations are needed.

4.1.1 VisualInfo Objects, Items, and Index Classes

A VisualInfo object can best be thought of as simply a file on a hard disk. It must be affiliated with a VisualInfo Item, and has a unique part number within the item. It has a size expressed in bytes, KBytes, Megs, or larger. It has a data format, such as ASCII, TIFF, MO:DCA, HTML, or one of many other data formats. This data format is often called the *content class* of the object. The file for the object resides on the object servers of the system. To retrieve an object involves knowing the item it is a part of, and the unique part number within the item.

A VisualInfo item can best be thought of as container for objects. It is given a unique item ID when it is created in the system, which will be used forever when referencing the item. It needs to be in an *Index Class*, and will have key field values for the key fields that are defined in the index class. There are two types of VisualInfo items:

- A *document* item is often used to represent what a customer would consider a document. It can have zero, one, or more objects in it. The Client Application considers a base object to be the scanned image, possibly a note object (an ASCII file), and possibly an object that contains the history of the document. All these objects are considered affiliated with the document, and together they make up the document.
- A *folder* item is a container for document items and other folders. You may think of a folder as the logical equivalent of a folder in a file cabinet containing documents that have some sort of relation to each other. For instance, a university may want to keep student records on a VisualInfo system. Each student may have a single folder that would be theirs, and documents about that student would be placed inside. An administrator working with the student would search for the folder, then open it up for the relevant documents about that student.

Index classes can be thought of as the type of document or folder. Every document and folder item are in an index class and acquire the attributes that were given to the index class when it was created. For instance, an insurance company might create an index class for their Claims Processing form. It may have key fields for claim number, policy number, policy owner′s name, and date received. Every claim form scanned into the system would be in this index class and have specific, even unique, values for each of the fields. Working the claim could involve searching the system for the form with a particular number.

4.1.2 Developing the Data Model

When sizing the system, you first need to determine what items and index classes are going to make up the system. This can be the most difficult task in sizing the system. You need to become aware of how the customer currently processes its documents, and how it wants to conduct its business with imaging. You need to develop the data model that will define the documents and describe what objects the documents will have. If folders are to be used, you need to determine what they represent to the business, and how they fit into the process.

When choosing key fields for an index class, your primary concerns are selecting fields that will be searched on, and fields that will distinguish one document from another when a list of search results are displayed. Remember that the attribute value table will have a column for every key field defined to the index class, and will have a row for every item placed in the index class. Avoid having an attribute for every thing you can think of. While some customers will say they want everything about an item searchable and in this table, this will lead to increased search times, and wasted space on the library server.

4.2 Characterizing the Workload

Once the environment has been analyzed and understood from the previous step, we can move to characterizing the workload generated by our environment.

The first step in the characterization process is to identify the basic component of the workload. The basic component refers to a generic unit of work that arrives at the system from an external source. For VisualInfo, this basic unit is a Library Server Function Request (LSFR). An LSFR is a request sent from the client to the server. An LSFR starts with a connect order, followed by one or more other orders, and ends with an end transaction order. Most LSFRs have only one order between the connect and end transaction orders. In 4.2.2, “Estimating the Number of LSFRs” on page 37, we show how to estimate the number of LSFRs generated by a given workload.

But first we will discuss the characteristics of server workload in more detail.

4.2.1 Estimating the Load on the Library Server

Every action performed by a client talking to a library server uses some small amount of the resources of the library server. The key to estimating the total load on the server is to estimate the total number of actions the client community must perform to support the customer′s business requirements. For instance, if the customer creates ten thousand documents per day, that is ten thousand store requests on the library server. If the same customer retrieves each document five times, that is fifty thousand retrieve requests. The number of searches, folder opens, workbasket routes, and so forth can be determined by asking appropriate questions of the customer in their terms.

With a data model that properly represents the customer′s data and business process, you can effectively size the system. The following questions deal with numbers and sizes of documents, folders, and index classes, and the amount of stress placed on the servers and workstations. The entire goal of the sizing exercise is to effectively estimate the following:

1. Storage requirements for the objects on DASD, optical, or other media

2. DASD requirements for the databases, prerequisite software, and other requirements on the servers
3. Processing power of the servers, along with memory recommendations
4. Number of clients needed to support the workload

You may also have documents that vary greatly in their size and other characteristics. You may want to answer each of the following questions by document type or index class.

1. How many documents and folders will be stored in the system?

   This is the most important question to answer, as it affects all four of the estimates. The number can be expressed in two ways: either the customer knows how many items they currently have in residence, or they know how many new ones they plan to receive each day for processing. For new items per day, you will need to determine for how many years you want to estimate the system, or how many years objects will reside in the system. This will tell you how much long-term media, such as optical, you will need. If your customer plans to keep documents “forever,” you should choose a length of time you want to size the system for, knowing that more storage resources will be needed for continued processing beyond this point.

2. What is the total size of the documents?

   This would be the sum of the size all of the objects in the document. It has the greatest effect on the amount of final media storage needed, and the network requirements needed to transmit the objects when they are stored or retrieved.

3. How many objects will be in the items?

   This affects the size of the library server database, the SBTPARTS table, and the size of the BASE_OBJECTS table on the LAN-based object servers. For VisualInfo implementations, typically a document item will have a base object as the scanned image, and possibly a note and annotation object also in the document.

   Digital Library documents will often have many different objects that are somehow related to each other. For instance, a television commercial may be a document item, but for objects it may have the video clip, audio clip, thumbnail of a representative frame, text part that is the transcript of the audio, a descriptive note part, and an index part that lists all the other parts, for a total of six objects for each commercial

4. What is the sum of the lengths of the key fields in the index class?

   This affects the size of the library server database. Each index class has an Attribute Value Table (AVT) that will have a column for each key field, and a row for every item placed in this index class. It is this table that is searched when retrieving items by attribute values. See 4.1.1, “VisualInfo Objects, Items, and Index Classes” on page 34 for recommendations on choosing appropriate fields.

5. How many days will the objects reside on DASD?

   Along with number of objects and size per objects, this question estimates the amount of DASD required for storage. Typically, VisualInfo implementations assume that new documents will reside on DASD for a set number of days while they are actively being worked on and often retrieved. Then they are migrated off to slower media, such as optical, for archiving purposes, where they are less likely to be retrieved, but are still needed for auditing purposes.

6. How many daily retrieves of items will be done?

   Customers should know how many retrieve requests are expected per day, or that each document will be retrieved, on average, a certain number of times for it to be ″fully processed.″ If, for instance, a claims form is retrieved five times during in its lifetime, the daily retrieves done by the system will be five times the number stored per day.

7. How many ad hoc searches for documents and folders will be done per day?

   Searches for items are done on the AVT tables of the index classes. The customer should know, for instance, how many calls they get at their help desk, or how many clerks they have doing processing. Each search places a stress on the library server, and may result in a document or folder being retrieved from the results list.

8. If you are using the workbaskets and workflow, you need to determine the following:

   - How many workbaskets will the document be routed through?

     Each route places a certain amount of stress on the library server. Also, it could be assumed that each route would also result in one item retrieve; why else would it be routed?

   - How many days will items reside in the workbasket or workflow?

     This affects the size of the library server database. The information concerning items in workbaskets and workflows is kept in the SBTWIPITEMS table. There is one row in this table for every item that has been placed in any workbasket or workflow.

9. How many hours each day is the system going to be available for input, processing, and object migration?

   The other questions are all phrased in terms of “How many... ...per day?” If the workday is ten, sixteen, or twen ty-four hours long, a smaller processor could be used on the library server than for a standard eight-hour day. You may want to size for a single hour of the peak workload, if one can be determined.

One of the bottlenecks in the system is migrating objects to optical or slower media off-shift. Increasing the number of hours allowed for off-shift may allow for reducing the number of object servers otherwise recommended.

4.2.2 Estimating the Number of LSFRs

You may not have used units such as LSFRs before. However, you can describe your workload in terms of end-user transactions such as scanning a document and indexing a document. Therefore, your capacity planner needs to understand the business process and how to translate various business processes into VisualInfo application functions. For example, the input process for an imaging system may consist of the following steps:

1. Scan a document (three LSFRs).
2. Add the document to a workbasket (two LSFRs).
3. Open document for indexing (four LSFRs).
4. Index/save document (four LSFRs).

5. Add to an existing folder (13 LSFRs).

This adds up to 26 LSFRs per document. We show how to use this information in 4.4, " Studying Benchmark Measurements" on page 41. There are other ways to collect LSFRs, however.

If the customer plans to use the client application that comes with VisualInfo, they can ask their IBM representative about an IBM internal tool called VisualInfo Sizer, which can be used to calculate the number of LSFRs generated. Results from these tools are used throughout this redbook. Refer to page 33 for information on how to obtain this tool.

In Table 1, we have listed the main client application functions and the number of LSFRs that are required to perform them.

Table 1. Client Application LSFRs by Function

Function	**LSFRs**
Create Document	3
Add to Workbasket	2
Open Document from Local Cache	4
Open Document from Object Server or LAN Cache	4
Close Document	1
Index/Save Document	4
Add to New Folder	20
Add to Folder	13
Basic Search	7
Open Folder	8
Start workflow	4
Complete workflow	10
Get next in workbasket	3
Route to next workbasket	3

If the user writes a custom application, Folder Manager traces on the client side can be used to estimate the number of LSFRs as it is shown in several chapters beginning with Chapter 15, "Customer Scenario #1" on page 175. For those cases where the customer does not have a working VisualInfo system, the number of folder manager API calls can be estimated or requested from the application developer. Most VisualInfo folder manager calls generate one LSFR per API call.

A similar process must be done to size the object server CPU and disks. Disk/optical storage size, rather than speed, often drives the choice of storage medium.

Following this method, we can add the LSFRs for a given hour (or our designated time period). A peak hour is often the best time period to cover, since that is when the most capacity is needed. The hourly LSFR rate should be 80% of the maximum supported for a given library server configuration to allow for peak processing hours. Time should be allocated for the object server (OS/2 and AIX) de-stager, migrator, and purger processes to occur without impacting the interactive requests.

4.2.2.1 Sample Workload

You may wonder what the LSFR load of your system is. You can compare your workload to the following sample system, or make your own calculations based on the method shown.

In Table 2, we can calculate the number of LSFRs that a system generates if:

- It receives 3000 documents per day into 600 new folders.
- Each folder is opened twice (on average).
- Each folder is put in a workflow containing three workbaskets.
- Each document is viewed three times (on average).
- Five hundred basic ad hoc searches are executed to verify index information.

Table 2. Number of LSFRs Generated by a Typical Client User

Function	**Daily #**	**Daily LSFRs**	**Hourly LSFRs (8-Hour Day)**
Document stores	3000	9000	1125
Add document to new folder	600	12 000	1500
Add document to existing folder	2400	31 200	3575
Folder opens (2 x 600)	1200	9600	1200
Document opens	9000	36 000	4500
Start workflow	600	2400	300
Route to next workbasket	600	1800	225
End workflow	600	6000	750
Basic search	500	3500	450
Total		111 500	13 938

A total of 13 938 LSFRs per hour are processed by the library server, assuming a steady arrival rate.

4.2.2.2 Activity of Users

It has also been measured that a normal client workstation can generate 750 LSFRs in one hour doing the mix of functions shown in Table 2. This user would be called a high activity (or heads-down) user. To determine the number of users supported, divide the LSFRs by the number of LSFRs generated by the user type:

- High activity user - 13 938/750 = 18 users
- Medium activity user - 13 938/560 = 24 users
- Low activity user - 13 938/375 = 37 users

Therefore, you can roughly estimate the number of users that your system can support by calculating the number of LSFRs per shift and dividing by the activity level as previously shown.

4.2.3 Capturing Trace Information

If you have a working VisualInfo system available, you can trace data and collect statistics to measure the current workload. The Windows client contains application traces for scanning and the client application. The OS/2 client contains command files to obtain system information. The MVS and AIX library server have tools to gather statistics.

4.3 Estimating Storage Requirements

Now we turn to the step of estimating how much disk storage you will need on each server.

4.3.1 Estimating Requirements for Object Storage

Estimating the requirements for object storage is very straightforward for LAN-based object servers (OS/2, NT, and AIX). Multiply the number of documents you expect by the size of the documents. To calculate the amount of DASD storage needed, multiply together the number of documents per day, size per document, and number of days you want them to stay on DASD. For long-term object storage, such as optical, use the number of years you want to project out times number of days per year instead of the days on DASD.

For MVS host-based object storage, you still need to know just the number of documents, size per document, and days on DASD number, but because the Host-Based Object Server stores the objects in a DB2 database by cutting the object up into 32 K segments, the DASD storage calculations are more difficult.

4.3.2 Estimating Library Server Database Size

The size of the library server database grows as the number of documents and folders grows. To estimate its size, you need to know the number of documents or folders you expect to have in the system, the number that will be in a workbasket or workflow at a given time, the number of folders you expect each item to be in, the number of objects you expect to have, the total length of the key fields of an index class, and the number of items that will be in each of the index classes.

Each document and folder will have:

- One row in the SBTITEMS table.
- One row in the SBTLINKS table for every folder it is placed into.
- One row in the SBTPARTS table for every object the document has.
- One row in the SBTWIPITEMS table if it is placed in a workbasket or workflow.
- One row in the Attribute Value Table of the index class it is in.

Calculations are based on the *DB2 Planning Guide* which states that the size of a table is the number of rows times the following: ((8 + SizeOfTheRow) * 1.5)+(((8 * NumberOfIndexs) + LengthOfIndexs) * 2.0).

The tables that grow as the number of documents grow are shown in Table 3 on page 41.

Table 3. UDB Tables That Grow

Table Name	RowLength	IndexLength /Number	Factor	Number of Rows
SBTITEMS	100	32/2	258	Number of items
SBTPARTS	91	28/1	221	Number of objects
SBTLINKS	82	94/3	371	Number of folder adds
SBTWIPTABLE	92	39/2	260	Number of items in workbaskets or workflows
SBTSUSPWIPTABLE	80	18/1	184	Number of suspended items
AVT tables	See Note 1			
AVT for TextSearch	101	16/1	212	TextParts (DL Only)

Notes:

1. The factor in Table 3 is the row length and index information run through the formula. It is effectively the number of bytes used for each row of the table. The table size would be the table factor times the number of rows.
2. The row length of an AVT table is the sum of the length of the key fields the index class has plus sixteen for the ItemID.

The library server has many tables other than the ones mentioned, but they do not grow in size as the number of documents grows, and their size becomes less significant for planning.

4.3.3 Estimating LBOS Database Size

The BASE_OBJECTS table on OS/2, NT and AIX object servers is the one table that grows as the number of objects stored on the server grows. Calculating its size uses the same formula as the library server tables, with a row length of 121, and a factor of 326.

The object server also has many other tables, dealing primarily with SMS strategy, but they do not grow in size as the number of objects grows, and their size becomes less significant for planning.

4.4 Studying Benchmark Measurements

Once the workload has been characterized, the next step is to conduct an experiment to measure the throughput capacity of the proposed system under this workload. Fortunately, we do not need to carry out this step since there has been much work performed on gathering performance information for VisualInfo. This information is contained in several documents. These documents are packaged with the internal VisualInfo Sizer tool, which is available to IBM representatives on an image tools disk. There are also some redbooks which are helpful. See Appendix F, “Related Publications” on page 229.

The capacity for each of the different library server platforms is shown in Table 4 on page 42. This table shows the number of high-activity users and maximum number of LSFRs supported, while still maintaining an acceptable level of connect time (or queue time) for the specified library server configuration.

A high-activity user generates 750 LSFRs per hour; a medium-activity user generates 75% of the work of a high-activity user, or 560 LSFRs per hour; and a low-activity user generates 50% of the work of a high-activity user, or 375 LSFRs per hour.

Any number of users or LSFRs higher than those stated may result in longer response times, which are listed in Table 4.

Table 4. Performance Summary

Server Type	**Number of High Activity Users**	**LSFR/hour**
OS/2 (PC350)	96	72,000
AIX (R24)	160	120,000
MVS/ESA (3090 6-way)	192	144,324

4.5 Forecasting Future Workload and System Performance

What if you have a working VisualInfo system that has acceptable performance? You plan to deploy VisualInfo throughout your company during the next two years. Will your response time change? Can your current library server handle this load? If not, what should you upgrade to? These are typical questions that need to be answered by the system capacity planner. Here we note two methods to address these.

One method that is often used in predicting system performance under future workloads is linear extrapolation. Linear extrapolation is quick and easy; however, this assumes that the CPU is the gating factor for system throughput. Tests show that there is a linear relationship between LSFR/hour and processor speed as measured by SPECint92 benchmark. Of course, this assumes there are no other bottlenecks or constraints within the system, such as memory. We recommend that you use this method if your system is not highly utilized (for example, peak CPU utilization is less than 50%). As system utilization increases, bottlenecks appear, congestion increases, and performance degrades at a fast rate and linear approximations become invalid.

For AIX machines, we recommend using the Relative Online Transaction Processing Performance (Relative OLTP Performance) rather than SPECint92.

Another method for predicting future system performance is benchmarking, as we discussed in 4.4, " Studying Benchmark Measurements" on page 41. Benchmarking produces more accurate results and often uncovers system bottlenecks; however, it is more expensive and time consuming. Benchmarks are widely accepted because they deal with real systems. When conducting a benchmark, it is essential to make sure that the simulated workload is as close as possible to the real world workload.

The VisualInfo Sizer tool is also available to IBM representatives to help their customers with capacity planning.

4.6 Related Reading

See Appendix F, “Related Publications” on page 229 for additional publications and redbooks. Two redbooks of particular interest in this discussion are:

1. *Image and Workflow Library: Best Practices for VisualInfo*, SG24-4792.

   This is an excellent reference for planning and installing the VisualInfo components. Included are examples of the actual installation, including recommendations and descriptions for the installation parameters.

   Chapter 9 contains capacity planning and performance expectations when running VisualInfo Servers in an MVS/ESA environment.

   Appendix B contains VisualInfo MVS/ESA Server measurements in a simulated environment. The configuration contained a VI Library Server and Object Server running in the same MVS/ESA image.

2. *IBM ImagePlus VisualInfo Library and Object Servers for MVS/ESA Planning Guide*, GG24-4452.

   Chapter 16, “Customer Scenario #2” on page 183 contains capacity planning and performance expectations when running VisualInfo servers in an MVS/ESA environment.

   Appendix C, “Trace APIs and LSFR Results from MVS/ESA with OS/2 Client” on page 221 contains VisualInfo MVS/ESA server measurements in a simulated environment. The configuration contained a VisualInfo library server and object server running in the same MVS/ESA image.

Chapter 5. Considerations in a Document Management System

In addition to choosing and tuning an operating system platform, it is important to understand some of the characteristics of the application subsystem you will be using. In this chapter we continue our discussion of a document management system with a high-level overview of some of its components.

Over the past few years, document imaging has evolved from stand-alone electronic filing systems to distributed client-server systems that span a variety of platforms and networks. At the same time, imaging is being deployed in mission-critical applications where system performance and reliability are of utmost importance. As a result, customers are now asking how to size a document management system and predict its performance up front before they install it.

As one may imagine, many components come into play when analyzing the performance of such a system. Among them are:

- Document size and volume
- Document capture and index rate
- Document retrieval rate
- Workflow/Workgroup interactions
- Display subsystem
- Storage subsystem
- Hardware and software configurations
- Network configuration

IBM VisualInfo is an example of a document management solution that provides enterprise-wide access to document processing and library management. The system consists of one or more clients connected to one library server and one or more object servers. The servers run under OS/2, Windows NT, AIX, or MVS. Network protocols can be either SNA or TCP/IP.

In this chapter, we present some of the issues that you should consider in analyzing the performance of system components. Secondly, we cover some methods that can be used to monitor resource usage and analyze different components that comprise the end-to-end response time.

5.1 Capture Subsystem Considerations

The document capture subsystem refers to the components used to enter the documents into the imaging system. This is one of the most critical subsystems, since scanning documents accurately, reliably, and quickly is at the core of the ability to perform work from the digitized image.

5.1.1 Scanning

Scanners are used to capture and translate the paper document into digital data or image. This ″cost of capture″ includes the scanning of the source documents and indexing the images into the system. The index information includes at least one unique primary key and secondary keys that may be used to search and locate the object once it is stored into the system. Some solutions include auto-indexing the documents by using a bar code or by performing some type of optical character recognition (OCR) on the key fields.

Scanner performance is measured in pages per minute (ppm), and in impressions per minute (ipm) for duplex scanners, which is the number of pages per minute times two. There are desktop scanners (10-20 ppm), mid-range scanners (40-60 ppm), and high volume scanners (80-120 ppm) on the market today.

All scanners have *rated speeds*, the speed at which the scanner mechanism can capture the documents. However, the rated scanner speed provided by the manufacture can only be achieved under ideal conditions. Actual scanner throughput is considerably lower than the rated scanner speed. The only way to determine the actual scanner speed in your environment is to run a test using sample documents.

Some documents may need to be scanned in gray scale, which is slower than bitonal (black and white) scanning. Many documents can be scanned in at a resolution of 200 dpi but, for OCR or forms recognition, scan resolutions of 300 dpi or more are usually required. Depending on the scanner, scanning at 200 dpi can be twice as fast as scanning at 300 dpi. Selecting gray scale or bitonal scanning and the resolution (dpi) has a dramatic impact on the performance of the entire imaging system. For example, an 8.5x11 document captured at 200 dpi has a file size of 467 500 bytes, while the same document captured at 300 dpi is 1 051 875 bytes (uncompressed).

5.1.2 Compression

Another factor that affects the size of the generated file is the compression algorithm used to compress the bitmap file. Compression algorithms are classified as either lossless or lossy. An image viewed after *lossless* compression is identical to the way it was before being compressed. An image viewed after *lossy* compression is different from before it was compressed because some information has been lost. Commonly used compression formats include CCITT Group III, Group IV, and JPEG. Compressed image size is typically one-tenth of the original or less. Table 5 shows the file size resulting from scanning one page using TIFF6 format for different DPI and gray scale values.

Table 5. File Size after Compression (K Bytes)

DPI	**100**	**200**	**400**
Bitonal (B/W)	23.7	42.6	92.1
256-level gray scale	104.2	292.5	796.8

5.2 Storage Subsystem Considerations

As you can see, documents stored as digitized images require considerably more space than documents stored using conventional coding such as ASCII text or data. The average 5000-character textual page (80 characters per line and 60 lines per page) stored conventionally requires about 5000 bytes (one for one). With elementary blank space compression, this can be reduced to approximately 2000 bytes per page. The same document page when captured digitally with a scanner and compressed for storage requires 25 times as much space, or about 50 000 bytes.

A document management system must support online, near-line, and offline document archiving to multiple storage media. Storage media are grouped into three main classes: solid state, magnetic, and optical. Solid state or random access memory (RAM) is the most expensive and the fastest. RAM access time is measured in nanoseconds, however, they are limited in capacity. Magnetic media can be magnetic tape or magnetic disk. Magnetic disks are by far the most common storage devices in use today. The disks vary in capacity from a few hundred megabytes to gigabytes. Depending on the format, magnetic tapes can deliver capacities between 1 GB and 50 GB per tape. The third kind of storage is optical disk.

5.3 Storage Management and Caching Considerations

Since most imaging systems have multi-year retention for some images while other images never get deleted, the amount of storage used over time can grow to several terabytes. Imaging systems requiring large amounts of storage also require active storage management or systems managed storage (SMS). The entire storage system must be optimized to provide the maximum trade off between capacity and performance. One of the ways to accomplish this is to use high-speed magnetic devices in either a pre-fetching or caching capacity. Pre-fetching images requires the ability to predict what is utilized during the processing period. These images are then physically moved or copied from a slower, high-capacity medium to a high-speed, online storage subsystem where all retrieval requests are directed.

Another means of managing imaging files is to use a high-speed magnetic subsystem as a cache that stores frequently used images. Unlike a pre-fetch, which involves a batch transfer of files from one source to another, the cache acts as a buffer to store only frequently used images for an extended period. Several studies show that in many work environments, the probability that the same document is referenced several times within a short period is fairly high. For a client-server application such as VisualInfo where the documents or objects are stored on the server, this creates a fair amount of traffic every time a document is retrieved. This loads the network and increases the response time, in particular, for those environments where the clients and the servers are connected over a wide area network (WAN).

For these environments, object caching is effective in decreasing the load on the network and dramatically improving client response time. The client applications have versions that run on either the OS/2 or Windows operating systems. A custom client can also be coded from a supplied toolkit.

5.4 Network Considerations

Adding a distributed imaging system to an existing network can congest the network, causing long delays in response times. Present network capacity and traffic must be understood before adding the imaging traffic. An imaging system has sources (scanners) and destinations (user workstations). Each image has a measurable size, typically between 30 KB and 120 KB per image. Sources send images to destinations at a certain rate.

For example, high-speed scanners can have an effective throughput above 100 pages per minute, which is nearly two images per second. If each image is 100 KB, that is already a load of 1.6 Mbps compared to a theoretical Ethernet capacity of 10/100 Mbps. It is feasible that each of these image files may need to be transported over the network as many as five times or more in the process of scanning, indexing and viewing; thus it can be seen that the network could easily become overloaded.

There are solutions to network capacity problems. One frequently used strategy is to isolate all the activity related to scanning and image processing to its own sub-network. For some cases, imaging may need to be implemented on its own separate network in order not to affect current applications. Also, there is a need to evaluate peak network requirements by analyzing the network traffic generated by scanning at the peak processing periods. Normally during a pilot phase of an image project, a sub-network can be installed that isolates the imaging traffic to determine sample transaction rates and network traffic without impacting the current network traffic and response time.

If peak processing network requirements exceed available network capacity, corrective measures must be taken. Alternatives may include scanning during off-shift hours, migrating the objects to remote object servers during non-primetime hours, increasing the network line speed, adding object servers, or spreading the user workload over longer periods of the day.

5.5 Throughput and Response Time Considerations

The two most important metrics that characterize the performance of a client-server system such as VisualInfo are throughput capacity and response time. Throughput capacity numbers indicate how big or how fast a server machine is needed for a particular workload. Throughput capacity is usually measured as the number of functions or requests completed in a unit of time. LSFRs per hour are the units of capacity.

The second performance metric is response time, which is the elapsed time from pressing the Enter key or selecting the **OK** push button to the time a response has been displayed so the user can do useful work. The response time interval includes the client workstation processing time, network transmission time, and server processing time.

For example, one customer might be interested in displaying a single document. Another customer may need to display several documents before they can do useful work. It is desirable to configure your system so that response time remains stable until the server reaches maximum capacity, or until the network becomes a bottleneck.

Contributions of every subsystem to the overall end-to-end response times must be clearly understood. End-to-end response time can be divided into three components: client time, network time, and server time. Figure 13 on page 49 shows a timing diagram of a typical transaction. First, the user presses the Enter key to view a particular document. The client application processes the end-user request (preprocessing time) and it builds and sends the request to the server. The server receives and processes the request. The response is sent back to the client workstation. The workstation may follow with zero or more requests to the server until it has enough information to service a user′s request. After the last response is received, the client workstation needs to format all response blocks to the user, such as opening a window and displaying an image in it.

Figure 13 on page 49 shows that different timing components within each subsystem can be added. This information must be known in order to be able to suggest improvements to reduce response times.

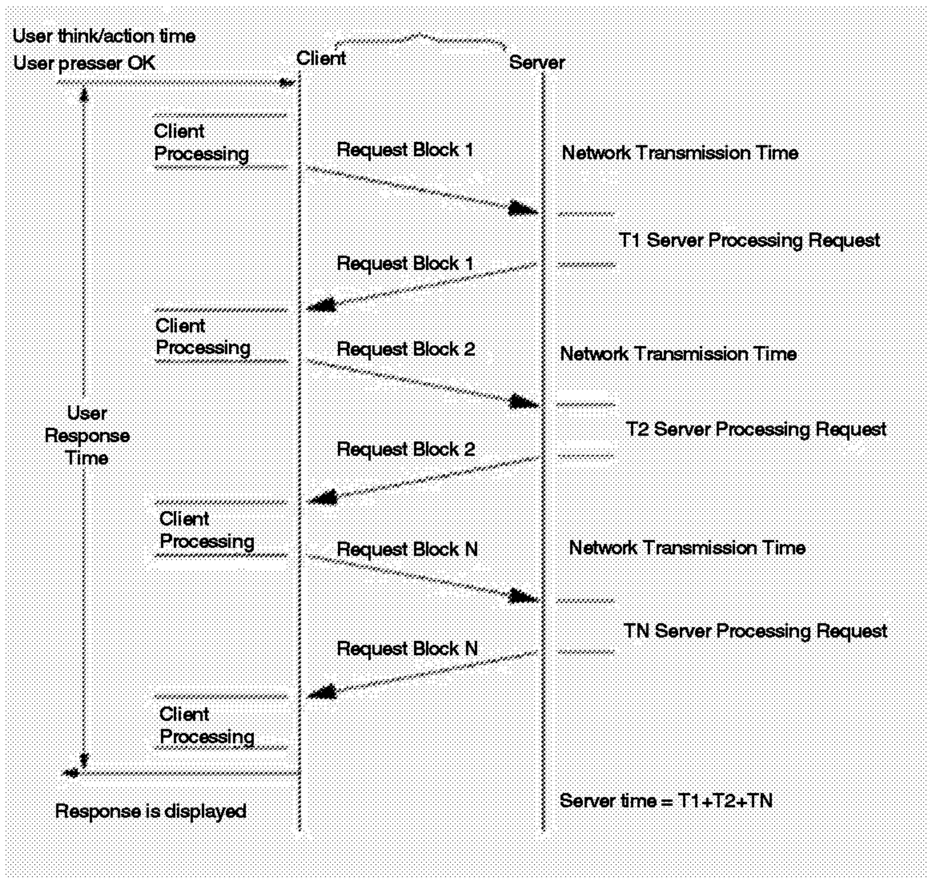


Figure 13. Timing Diagram of a Typical Function

The response time experienced by the user would be the overall sum of the three process times: client time, network time, and server time. The number of request blocks issued is n. The number of request blocks and response blocks is two times n, or t, and each requires a transmission over the network.

```
Response time = Client Process time + Network Time  + Server Process Time
              =  (C1+C2+..+Cn)      + (N1+N2+..+Nt) + (T1+T2+..+Tn)
```

For example, Figure 14 on page 50 shows contributions of the client and the combined library server and object server to the response time for various functions in a directly connected LAN environment. For most functions, the time spent in the client is much greater than the time spent in the server. Therefore, for similar environments and given that throughput capacity of the servers is adequate, upgrading the client machine with a faster one (processor speed, memory, disk, and display subsystem) decreases response time more than if the server is upgraded. Note that for most functions, response time was under two seconds. It should be noted that these measurements were taken for an unloaded system and network times were negligible compared to other times. For example, the time to transmit 20 000 bytes over a 16 Mbps token-ring is 0.01 seconds.

If the same measurements were taken in a WAN environment where the client and the server are connected over a 56 Kbps link, network times cannot be ignored.

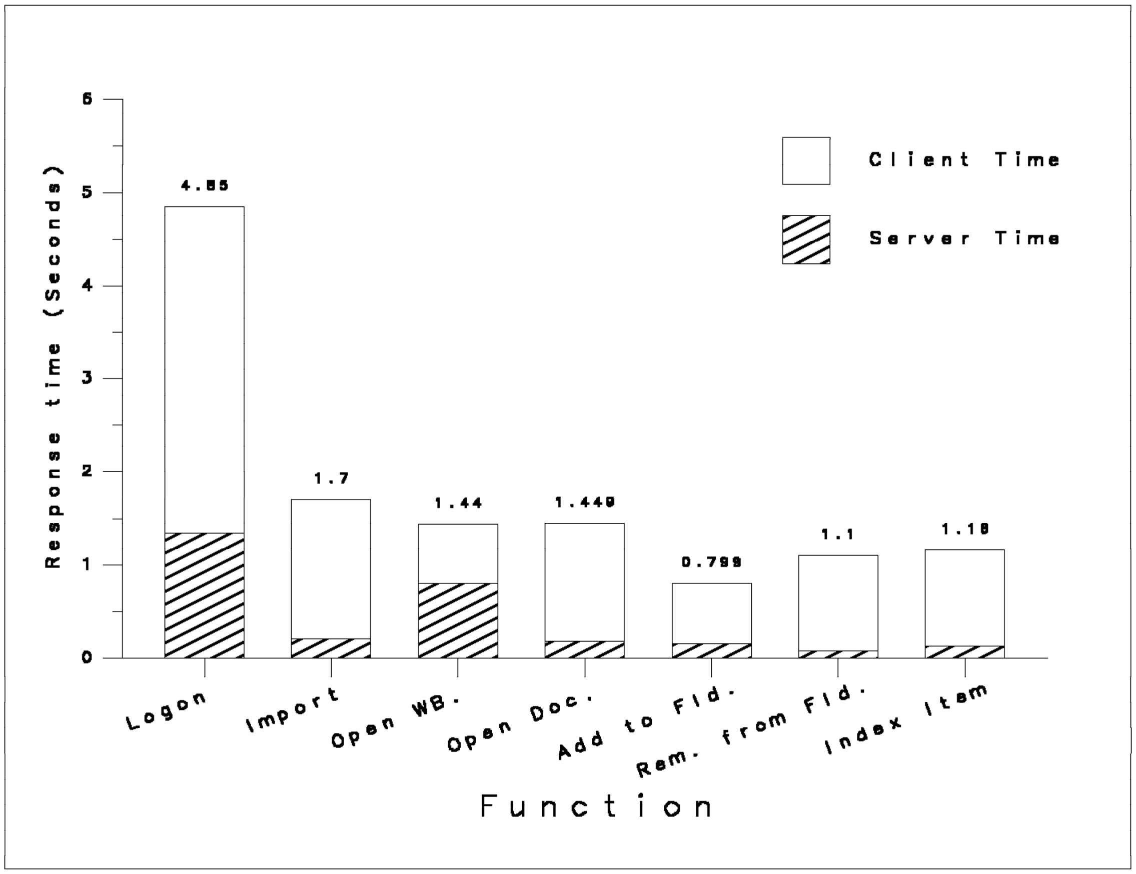


Figure 14. Response Time Apportionment

5.6 Remote Library Server Response Times

One of the little-known facts about VisualInfo is that large amounts of data are transferred between the library server and the client workstation for many function requests. In fact these data transfers can be staggeringly large.

Intuitively people often believe that large data transfers only occur between the object server and the client as objects are uploaded or downloaded. While this is true, it is also a fact that large quantities of data are transferred between the library server and the client. These transfers are so large as to require significant bandwidth between client and library server if the client workstations are located remotely from the library server, even though the object server may be LAN-attached to the clients.

For example, the following figures were measured on a customer′s system:

Retrieve index	8,132 bytes
Specific document search	30,128 bytes
Import coded data	34,344 bytes

Retrieve image	14,493 bytes
Open workbasket	439,099 bytes
Store document	7,372 bytes
Route document	1,444 bytes

The figures quoted are not for any one VisualInfo API, but were the number of bytes transferred between client and library server to perform a business function. As such, they represent the total bytes transferred by several API calls, but the transfers needed to be completed rapidly to give good response times.

On a dedicated 2 megabit link, the Open Workbasket function would take 2 seconds of line time without allowing for any queuing. On a 128 kilobit link, this would rise to nearly 16 seconds, again assuming no queuing and ignoring the impact on other users. Clearly any plans to have client workstations located remotely from the library server must be planned carefully, with a full assessment of the bandwidth needed to support realistic response times. Bandwidth of less than 512 kilobits per second is unlikely to be adequate in anything other than very special situations, and often much more will be required.

Chapter 6. Capacity Planning Tips for VisualInfo Platforms

In this chapter we discuss tips and techniques for capacity planning on each server platform for which VisualInfo is available.

6.1 Client Workstation

On your VisualInfo system, you can trace data and collect statistics to measure the current workload. In this section, we discuss trace and statistics data captured within VisualInfo.

The Windows client contains application traces for scanning and the client application. The OS/2 client contains command files to obtain system information. For a customer scenario on capacity planning with Windows, see Chapter 15, "Customer Scenario #1" on page 175. For customer scenario on capacity planning with OS/2, see Chapter 16, "Customer Scenario #2" on page 183 and Chapter 17, "Customer Scenario #3" on page 195.

6.1.1 Client Trace for Windows

This trace facility is available from the IBM support center for the Windows client in VisualInfo version 2.2.

The Windows client trace is enabled by settings in two files. File \FRNROOT\LOG\LOGSRC.INI contains a list of trace functions. The folder manager calls are traced in options VIINTRFC and VIITEM. These options should be enabled for tracing. Edit file \WINDOWS\VIC.INI and add the following statements:

```
[Trace]
FolderManager_Log=Yes
```

Start the client application, perform the work to be traced, log off the client, and close the client application window. The trace file is located in \FRNROOT\LOG\VIC.LOG. The APIs are listed along with the approximate time in milliseconds to complete. Trace entries for APIs that connect to the library show times greater than 0 milliseconds.

6.1.2 Client Trace for OS/2

The folder manager trace command file for OS/2 initiates a VisualInfo application, which can be either the standard VisualInfo Client application or a user-written application. The OS/2 trace command file, named FRNODBIM.CMD, produces a client application trace, image services trace, class library exception log, and folder manager trace. The command file creates a single file, FRNROOT\LOG\FRNDBIMG.ZIP, that contains all the trace files. The folder manager trace file is in a file named FMDUMP.LOG and produces a list of the APIs processed during the application execution.

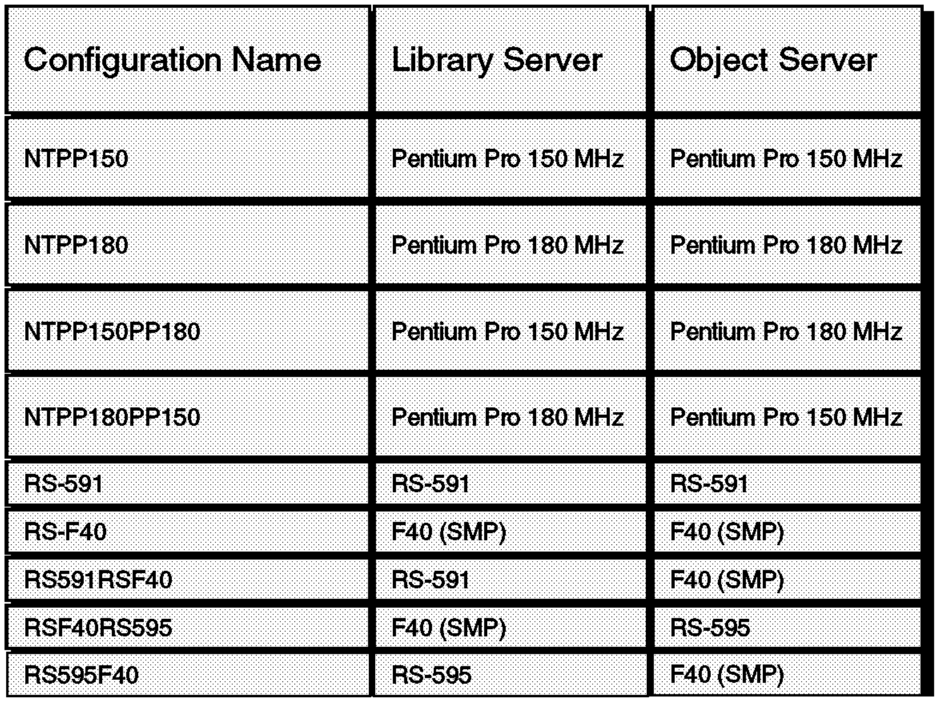
Start the client application by executing FRNODBIM.CMD, perform the work to be traced, log off the client, and close the client application window. If a custom application has been developed, change the name of the file that is executed in file FRNODBIM.CMD to match. The API trace file is located in \FRNROOT\LOG\FMDUMP.LOG. By examining the APIs in the trace along with the timestamps, you can determine which APIs interact with the library server. If there is a minimal time lapse between the start and end entries, the request was satisfied by the information stored in the local cache.

6.2 AIX System

You may have heard that the VI 2.3 AIX servers give better performance than previous releases. This is true and, in addition, using the same hardware configuration, we also found at least 26% throughput improvement on an RS6000 model 591 machine. This was because of substantial improvement in interprocess communication, in particular the shared memory and semaphores.

Note: For a customer secenario using AIX servers, see Chapter 15, "Customer Scenario #1" on page 175.

The following graphs demonstrate the considerations when planning for a typical VisualInfo system. These graphs were created after using some of the performance data from the appendixes in this redbook. They are meant to provide rule-of-thumb lookup graphs. Figure 15 summarizes the various configurations used for the performance tests. This data may also be used as a key to understanding the x-axis references to configurations in the two graphs that follow:



Configuration Name	Library Server	Object Server
NTPP150	Pentium Pro 150 MHz	Pentium Pro 150 MHz
NTPP180	Pentium Pro 180 MHz	Pentium Pro 180 MHz
NTPP150PP180	Pentium Pro 150 MHz	Pentium Pro 180 MHz
NTPP180PP150	Pentium Pro 180 MHz	Pentium Pro 150 MHz
RS-591	RS-591	RS-591
RS-F40	F40 (SMP)	F40 (SMP)
RS591RSF40	RS-591	F40 (SMP)
RSF40RS595	F40 (SMP)	RS-595
RS595F40	RS-595	F40 (SMP)

Figure 15. System Configurations Used for Testing

Figure 16 on page 55 gives an idea of the throughput that can be expected under test conditions from various configurations, in LSFRs per hour.

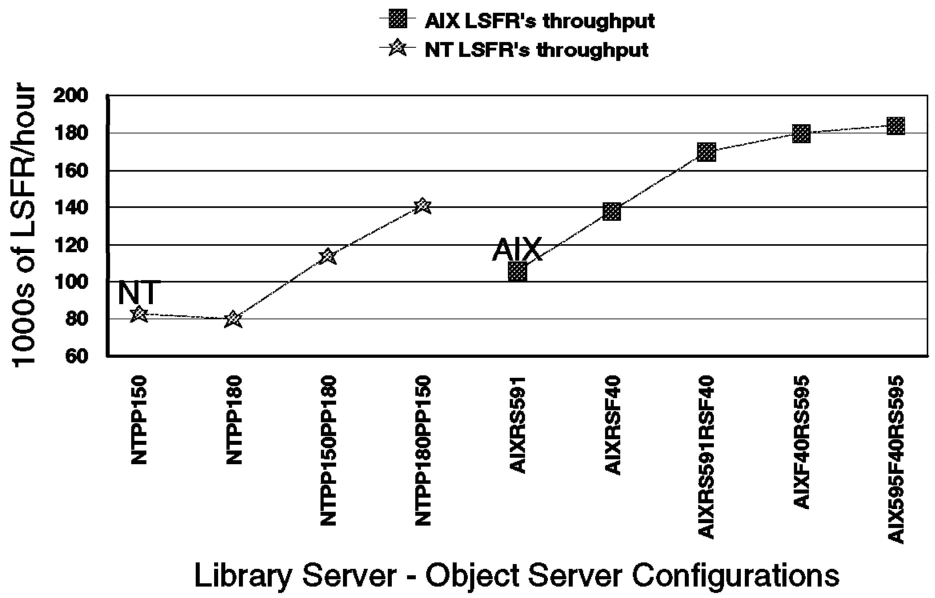


Figure 16. Summary of Transactions and Throughput

Figure 17 gives an idea of the number of concurrent users that the various configurations can scale up to, illustrating the various different types of users.

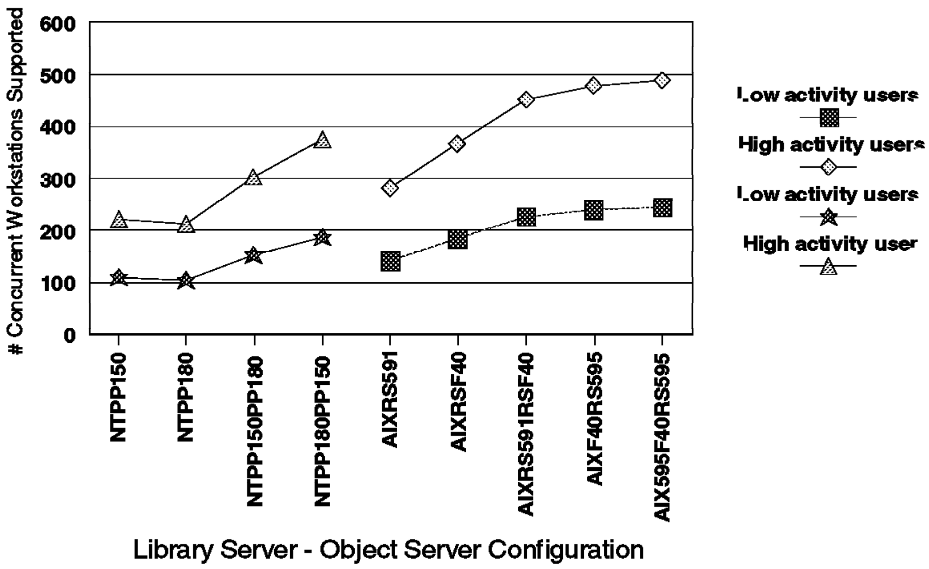


Figure 17. Summary of Concurrent Users

6.2.1 AIX Statistics

The AIX library server provides statistics. The statistics produced include the total request blocks processed and, for each library order code, the number of times of each order code was executed and the average elapsed library server time in milliseconds. The same statistics are summarized by patron ID. Figure 18 on page 57 contains a subset of the AIX Library Server statistics.

```
Begin listing of Library Server statistics:

Number of request blocks processed = 1035;
Total time (milliseconds) to process request blocks = 115260;
Number of processors active = 5;

Order timing information:

 Order CHECKIN_ORDER_CL:
  4294963702 milliseconds to process 31 orders;
  Average processing time = 138547216 milliseconds;
 Order CHECKOUT_ORDER_CL:
  35170 milliseconds to process 34 orders;
  Average processing time = 1034 milliseconds;
 Order CONNECT_ORDER_CL:
  4294860978 milliseconds to process 1029 orders;
  Average processing time = 4173820 milliseconds;
 Order ENDTRANS_ORDER_CL:
  4294912114 milliseconds to process 1023 orders;
  Average processing time = 4198350 milliseconds;
 Order DQUERY_ORDER_CL:
  109942 milliseconds to process 82 orders;
  Average processing time = 1340 milliseconds;
 Order RETRIEVE_ORDER_CL:
  70356 milliseconds to process 42 orders;
  Average processing time = 1675 milliseconds;
 Order STORE_ORDER_CL:
  10798 milliseconds to process 32 orders;
  Average processing time = 337 milliseconds;
 Order STOREITEM_ORDER_CL:
  15536 milliseconds to process 31 orders;
  Average processing time = 501 milliseconds;
 Order DEFWORKSLIP_ORDER_CL:
  4294944870 milliseconds to process 28 orders;
  Average processing time = 153390888 milliseconds;

Patron name = STEVE
 Number of requests blocks processed = 97;
 Total time (milliseconds) to process request blocks = 4294813286;

 Patron order timing information:

 Order CONNECT_ORDER_CL:
  4294917340 milliseconds to process 95 orders;
  Average processing time = 45209656 milliseconds;
 Order ENDTRANS_ORDER_CL:
  4294884422 milliseconds to process 95 orders;
  Average processing time = 45209309 milliseconds;
 Order DQUERY_ORDER_CL:
  39436 milliseconds to process 9 orders;
  Average processing time = 4381 milliseconds;
 Order RETRIEVE_ORDER_CL:
  5890 milliseconds to process 1 orders;
  Average processing time = 5890 milliseconds;
 Order STORE_ORDER_CL:
  4294962890 milliseconds to process 1 orders;
  Average processing time = 4294962890 milliseconds;
 Order STOREITEM_ORDER_CL:
  5480 milliseconds to process 3 orders;
  Average processing time = 1826 milliseconds;
```

Figure 18. AIX Statistics

6.2.2 AIX as a Web Server

For information on sizing an AIX system as a Web server, see Appendix A, "RS/6000 Web Server Sizing Guide" on page 203.

6.3 MVS System

The MVS library server has two CICS transactions, FRN6 and FRN7, that print library server statistics. The statistics produced include the library order code, the number of each order code executed and the average elapsed library server time in milliseconds.

Note: For a customer scenario using MVS servers, see Chapter 16, "Customer Scenario #2" on page 183.

6.3.1 Recording Statistics

To begin recording the MVS statistics, use transaction FRN5 to capture the statistics:

```
FRN5 SET TAKE_TASK_STATS 1
FRN5 SET TAKE_SYSTEM_STATS 1
```

To produce a report of the statistics, use either transaction FRN6 to produce a snapshot or FRN7 to produce a report at specified intervals. Use FRN6 to display the statistics on a CICS screen. The parameter INIT/NOINIT can be used to reset the counters to zero after each report is generated.

```
FRN6 FRNT DISPLAY NOINIT
```

Transaction FRN7 is a long-running transaction. A sample of output from FRN7 is in Figure 19 on page 59. The interval was set to 30 minutes and the counters were initialized to zero after each snapshot.

6.3.2 Key Order Codes

Key order codes used to determine the LSFRs executed are:

- 1034 (CONNECT_ORDER_CL) and 1134 (ENDTRANS_ORDER_CL). The number of transactions for these two should always be the same.
- Order code 1356 is a static query request (SQUERY_ORDER_CL) and is used frequently to receive information regarding workbaskets, index classes, and user privileges.
- Order code 1512 (RETRIEVE_ORDER_CL) is a retrieve object request and may generate two object server requests.
- Order code 1612 (STORE_ORDER_CL) is a store object request and generates two object server requests.

The description for each order code is in Appendix D, "MVS Library Server Statistics (FRN6) Output" on page 225.

When a client issues a request, the request is encapsulated into a request block that begins with a connect order. It is important to be able to match the workload performed on the client with the system statistics.

In Figure 19 on page 59, 16070 LSFRs were executed, which included 22389 static queries, 807 retrieve requests, and 428 store requests. The average object server time for a retrieve is 2.7 seconds, and for a store, it is 0.9 seconds.

```
*********************************************
  System Statistics for Data Base : VIMVS
         At: Thu Jan 30 16:18:51 1997
*********************************************
 Order     Number     Avg. Elapsed      Avg. Elapsed
  Code   of Orders    LS Time (MS)      OS Time (MS)
  1012       905           12
  1023       905           21
  1034     16070           11
  1112       324           69
  1134     16070           12
  1212       424           15
  1223      1813           50
  1356     22389           16
  1367      1072           79
  1423       552           24
  1512       807         1349              2725
  1612       428          761               886
  1954       763           23
  1965      8154           61
  2012       463           67
  2023       520           29
  2034        32           21
```

Figure 19. MVS Library Server Statistics from Transaction FRN7

6.3.3 MVS Statistics by Transaction Type

FRN6 output statistics are shown in Appendix D, “MVS Library Server Statistics (FRN6) Output” on page 225. The order codes and counts listed for several transactions are included. The counts are cumulative. The description for each order code is in Appendix D, “MVS Library Server Statistics (FRN6) Output” on page 225.

If the OS time seems large, it is because on the first request the object server must retrieve the object. On the next requests, the object is cached (in the workstation local cache or in LAN cache), and the object server is not accessed. So for the object server only, the original retrieval time is available to display in the report. But this caching lowers the average retrieval time for the library server,

6.4 AS/400 System

VisualInfo for AS/400 capacity planning follows the same process that the other VisualInfo and Digital Library platforms follow. The size and characteristics of the server platform will differ, but the logical data model is essentially the same. You must understand your customer′s business needs, and develop a model which satisfies those needs.

If you are new to capacity planning for VisualInfo systems, or if you would like more background information on capacity planning, read the sections in the beginning of this chapter on capacity planning for VisualInfo. In addition, we list several other publications which may assist you in the task of capacity planning.

6.4.1 Estimating Object Storage Requirements

The process for estimating the amount of storage required for the objects (images) is the same for VisualInfo for AS/400 as the other VI and Digital Library platforms.

Follow these steps to estimate DASD storage requirements for objects:

1. Calculate average document size = average size of a page * average number of pages.
2. Calculate number of documents input daily = number of documents scanned and imported * average document size (1).
3. Calculate DASD required for documents = number of documents input daily (2) * average number of days document is stored on DASD.

   Note: If the number of days on DASD is different for different types of documents, you should perform this calculation separately for each document that has a different number of days on DASD, to be as accurate in your estimate as possible.

Calculate the amount of optical storage required for long term storage, using the following formula:

Optical storage = DASD requirement (3) * number of years * number of days per year.

6.4.2 Understanding Server Capacity

Capacity planning for a VisualInfo for AS/400 server will be similar to planning capacity for any AS/400 client/server application.

AS/400 client/server applications share similar architectures with other client/server applications which include the following components of response time:

- Client processing: The VisualInfo client application is responsible for requesting the data and images from the server, and formatting the information for presentation.
- Application server processing: The VI/400 server programs receive and process the client requests.
- Database server processing: DB2/400 is integrated into the machine, and it manages the library and object server database, also processing database requests (indexing, searches) that come from the client.
- Communications processing: Whether clients are connected by local area network or by analog connection, the communications processing functions contribute to the overall response time between client and server.

6.4.3 Server vs. System Models

The current AS/400 models are split into two different types: the “system” (traditional) models and the “server” models. The server models are optimized for client/server applications like VisualInfo for AS/400. Therefore, if the line of business (LOB) applications that run with VI/400 are client/server applications, or if VI/400 is running on its own machine, the server models provide excellent performance at a low price. If LOB applications are traditional AS/400 “green screen" applications, the system models have better performance characteristics.

6.4.4 ImagePlus/400 Imagesizer

The Image Competency Center in Gaithersburg has a sizer for ImagePlus/400. This sizer is in the process of being updated for the additional functionality provided by VisualInfo for AS/400. The current version of the sizer can provide a better estimate of server requirements than our guideline in this chapter. However, the estimate derived from the sizer should be verified before being used as the final word on the sizing requirements of a given server.

6.4.5 Additional Redbooks and Reference Material

The following redbooks provide additional information on capacity planning for an AS/400 server:

- *AS/400 Server Capacity Planning*, SG24-2159
- *AS/400 Performance Management*, SG24-4735

In addition, the AS/400 product manuals provide information on tuning and capacity planning on an AS/400 server.

Chapter 7. Scenarios for Digital Library Applications

Here are two scenarios to help you plan a Digital Library system. We begin with an image archive system using the Web, and we follow up with a digital video management system. Both of these descriptions should give you a feeling for how to plan your Digital Library system for capacity and performance.

7.1 A Web-Enabled High Quality Image Archive System

Heritage collections of digitized manuscripts, paintings, pictures and artifacts lead to the first implementations for digital libraries. The considerations in this section are based on customers′ requirements for High Quality Image Archives, implemented according to the extended functions and services of IBM′s Digital Library and presented through the new Web Services Infrastructure, the Grand Portal.

The information that we outlined in about the considerations for a document management system is also valid for an image archive solution. But, in addition, when the Internet becomes the communication vehicle, sizing the network and the middle-tier server becomes a key issue. Therefore, major focus must be put in the design of this system component.

The most important consideration is the maximum number of clients that the system has to support and the type of work that they have to accomplish. Answering this question will help you decide how many clients a single Digital Library server workstation can support. This will help determine the number of workstations the system needs in order to support the expected work load.

When you perform a customer requirements collection and analysis project, the following questions should be raised:

- Is this system going to be available through the Internet or intranet?
- What is the potential demand for access to this site?
- What is the speed of your connection to the Internet/intranet?
- How many requests will the system be serving (h,d,y)?
- What is the typical document type being searched?
- Are these multi-part documents?
- What is the average file size of the retrieved document/part?
- Are there parts which will be streamed out?
- Will the Web server be generating dynamic data for access?

The remaining sections of this redbook will help you understand how to use the answers to these questions to design a successful Web-based image archive solution.

7.1.1 The Scanning and Capturing Workstations

Standard digitization tasks can be performed with any existing capture workstation capable of running the scanner and appropriate software (for instance, Adobe Photoshop). The scanner can be any device that produces Photoshop-compatible images of sufficient quality to meet the needs of the customer. Digital Library does not specify a scanner or scanning software. The only requirement is that the scanning process produce a TIFF image of suitable quality, resolution and size to meet the customers′ requirement. Since all scanning solutions are capable of producing TIFF images, the customer is free to use an existing scanner and need not worry about special requirements when contracting out the scanning work. If digital cameras become appropriate input devices for this application, and if the images are not already in TIFF format, they can be easily converted using image processing tools such as Adobe Photoshop.

The scanned TIFF image which is directly produced by the scanning process is of the highest quality and usually of maximum resolution. Bear in mind that the meaning of this statement is dependant to a very large extent on the business model. The term “high quality” may have different meanings for people from different domains. Most users accessing the Digital Library over a network would not want to access this image since it requires high bandwidth and a client computer with large amounts of memory. Therefore, a Digital Library import facility would, for example, create four secondary images called “derivatives” from the TIFF image by using image conversion utilities which create the four derivatives: *full, screen, thumbnail* and *zoom*. Such utilities come with the DB2 UDB Image extender of the Digital Library watermarking libraries (see the IBM picture conversion utility any2any.)

A full image is a digitally compressed version of the original TIFF image and is the largest and best quality of all the derivatives. For browsing and quick views of images, a smaller screen size representation of the full image, called a “screen image,” and also a much smaller derivative image called a “thumbnail” would be used. Another derivative, called a “zoom image,” is a 2X magnification of the screen image. Based on the application requirements, one could store all images or only those derivatives.

7.1.2 Data Filtering and Compression

In the process of creating the derivatives, the image can be reduced in size, sharpened, rotated, the color space converted, compressed to JPEG or GIF format, and a watermark may be applied. All of these capabilities are controlled by a variety of parameters to allow the image processing to be tailored to the individual needs of each customer. Furthermore, these functions are designed to work in a batch load process to minimize the time required to process and load the assets into Digital Library.

The actual level of JPEG compression that is acceptable will need to be determined through experimentation; however, in our experience 50:1 compression can be achieved without noticeable loss with most images. Beyond this, it is a function of the image content and interpretation of the viewer as to what is acceptable.

The result of the capture process are the four processed derivative images and a file with one record for each processed TIFF image that contains all the relevant data about the derivatives so that they can be quickly loaded into the Digital Library.

7.1.3 Storage Subsystem Consideration

As one would imagine, only the derivative images are stored in the Digital Library′s online storage for interactive search and browse. The original TIFF image would be stored in the less expensive near-online storage or shelved offline. Again this depends on the business model chosen. Assume our Web-enabled High Quality Picture Archive contains pictures of an original average size of 6″x14″ and we have to store 50,000 of them. A simplistic disk storage computation would give us the figures shown in Table 6.

Table 6. High Quality Image Storage Calculations

Item	**Calculation**	**Total**
Original photo (average 6″x14″)	n/a	n/a
Scanned TIFF image	6i x 300dpi x 14i x 300dpi x 24bpp	23 MB
“Full” image (50:1 JPEG)	23 MB/50	450 K
“Zoom” image (150 dpi JPEG)	6i x 150dpi x 14i x 150dpi x 24bpp / 50	113 K
“Screen” image (75 dpi JPEG)	6i x 75dpi x 14i x 75dpi x 24bpp / 50	28 K
“Thumbnail” image	200p x 200p x 24bpp / 50	2.5 K
Metadata	50 rows x 80 columns	4 K
Total online storage per record	Sum of above w/o original	~ 600 KB
Total storage per record	Sum of above + original	~ 24 MB
Total storage for all records	50,000 x 24 MB	1.20 TB

7.1.4 The Data Model

To make possible the management and retrieval of stored images from the IBM Digital Library, some indexing information must be attached to the images by which they can be identified. In terms of Digital Library this is done by creating one or more index classes. It is advisable to define as few index classes as possible. A complex data model places a heavier load on the system and is more difficult to maintain. Once the data model is developed, the data injection and the indexing is done during the loading process. For better and enhanced search performance, any or all of the attributes in the index class may be included in the Text Server search index, which will index the text information contained in those fields for free-text searches. By doing so one would overcome the penalties of full table scans when doing wild card searches on attribute fields in the underlying data base. Usually, a free-text search is 10x faster than a SQL string compare on the same test data. Another advantage of using the Text Search engine is that semantical processing of the text data may be done by applying linguistic features to get faster and better results. Normally, a large amount of text data is better searched through the Text Search engine than through the relational database.

7.1.5 The Loader

Content data injection into the Digital Library is a loader work process, where the scanning workstations stage their high quality images onto an intermediate middle-tier server for further processing. The Digital Library import facility as a next step would create the four derivatives (*full, screen, thumbnail* and *zoom* images) from the original TIFF image using the image conversion utilities.

In general, loading is performed in three steps:

1. A new item is created in the Library Server and the image files are loaded as content item parts into the object server.
2. The attribute data is imported into the index class from the attributes file and loaded on the library server. Any fields tagged to be indexed for the full-text search are indexed at this time.
3. These items are placed into a workbasket for additional processing.

Typically a large number of items would be stored in the Digital Library at once. As such the Loader must be designed to perform a batch operation by reading loading instructions from external loader files.

The important thing to consider here is the network bandwidth that this data injection process requires. For example, having 5 scanning workstations uploading large image files onto the server at the same time can easily lead to network congestion. Therefore caution must be used when designing the overall system interconnection network layout. Any networked machine can connect to the Digital Library, browse the contents and view or print images. So it is mandatory to subdivide the network into segments of adequate throughput capacity to serve the designated audience as expected. Consider using switching Fast-Ethernet or ATM backbones.

7.1.6 Searching/Browsing

Searching and browsing of the picture collection by end users is accomplished by a standard Web browser using a series of Web pages to perform searches and view results. Access to the Web interface is available on the Digital Library middle-tier server known as the “Grand Portal,” which is configured as an extension of a Web server (Apache, Netscape FastTrack, Java WebServer, Lotus Go Web, IIS) using Net.Data, serverlets or CGI scripts.

Searching is done using the Digital Library combined search facilities, where parametric, free-text and image search are supported by the use of Text Miner and QBIC. Text Miner supports Boolean, free text and hybrid queries. Words entered in the text area on the Simple Search page are used to form a free text query. A free text query consists of single words that do not need to be adjacent to each other in a document. The Text Miner index is scanned and items containing any of these words are returned. All of the results are displayed on the Results Gallery page. This is known as a *hybrid query*. Phrase searches are available and can be quite complex.

Attribute searching is typically limited to a subset of the index class. This is because several of the attributes, such as screen_image_size and others, are for Web display purposes only.

Today the most popular search paradigm used is “navigation of the infospace.” At a given time from the results list page, the user can click a “thumbnail” or “title” to further zoom into the image or into new information. For example, from the Attributes frame, the user could click **Zoom** to bring up a new browser window with just the zoom size image for the item contained within it. This allows the user to view the screen and zoom images at the same time.

Table 7 shows a summary of the requirements that you will need to support searching and browing.

Table 7. Summary of Searching/Browsing Requirements

Disk storage calculations	**Calculation**	**Total**
Total online storage per record	Sum of above w/o original	~ 600 KB
Total storage per record	Sum of above + original	~ 24 MB
Total storage for all records	50,000 x 24 MB	size = 21.20 TB
Search/Report Requirements		
Attribute search	Yes	Standard on advanced search.
Free text search	Yes	Standard with text-server.
Thumbnail images with results	Yes	Standard.
Full size images available	Yes	Thumbnail, screen, 2 x screen and full all available.
Results can be tagged	Yes	Can be implemented with Digital Library file folders and some changes to the search/results interface.
Printing of results (Color, 8x14)	Yes	Standard with browser. Additional, more sophisticated reporting can be developed.
Printing of high original	Yes	By print request submission.
Records can be “archived”	Yes	Can be implemented with an “archive” field in the index class, or a new index class for “archived” records.
Records can be “purged”	Yes	Standard.
Distribution/Access		
Concurrent users	Yes	Standard with 10-user license.
Networked access (intranet)	Yes	Standard.
Web access (Internet)	Yes	Standard.

7.1.7 Network Considerations and Sizing Factors

When sizing a Web server, the most important consideration is the size of the target audience.

7.1.7.1 Bandwidth

In working with a customer to size the Web solution, it is important to understand the implications of the speed of the networking connection to the Web server. Often, many potential Web content providers are very focused on the value of "hits per day."

The level of traffic that a particular Web server can support will be dependent on the server type, the content accessed on the server and the speed of the connection of the server to the Internet environment.

An Internet service provider will deliver a connection of defined speed; three of the most common speeds are:

- ISDN (128 Kbps)
- T1 (1.544 Mbps)
- T3 (45 Mbps)

For an intranet environment, common LAN speeds are 10 Mbps (over Ethernet) and 100 Mbps (over fast Ethernet or FDDI). Figure 20 shows the interrelationship between the average Web transaction size, the speed of the networking topology, and the maximum theoretical "hits per second."

As the average Web transaction size increases, the maximum number of transactions decreases. Sites that plan on being mostly text-based will have average transactions sizes around 1 to 5 KB. Most well-designed sites with a mix of text and graphics intended for access by modem users are in the 10 KB per transaction size, while sites with a substantial portion of multimedia content can exceed 100 KB per transaction.

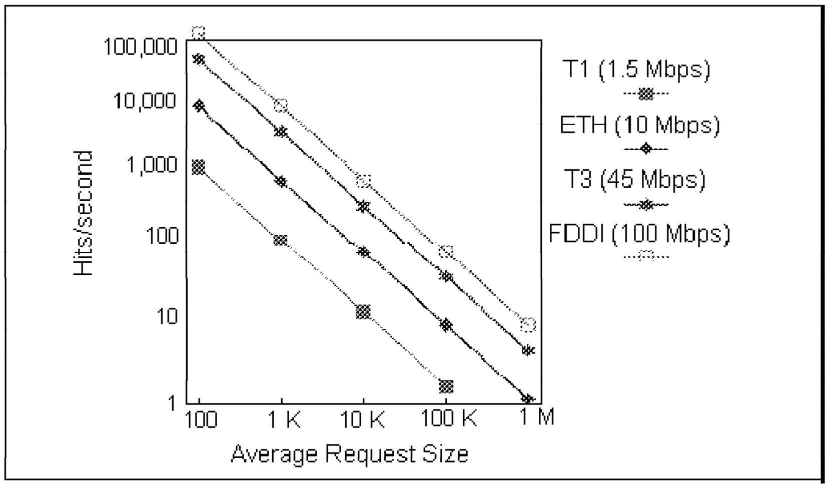


Figure 20. Comparison of Media Capacity

Figure 20 shows the relationship between the speed of the network, the average request size and the maximum theoretical "hits per second." To translate this into a number of hits for an 8-hour peak usage period, multiply the hits per second by 30,000.

7.1.7.2 Content Type

The size of the Web content is important in looking at the resources required for a server, indicating the necessary data storage requirements. Additionally, when the content on the Web server is dynamically generated, substantial processing resources may be required. Dynamic content on a Web site can be generated in many ways, from a simple counter that displays the number of hits that a page has received, to a system that uses analysis of user clicks to tailor the information that the user sees at the site. To gain some insight into the impact of simple content generation on performance, a discussion and simulation of the dynamic workload model should be considered.

7.1.7.3 Number of Clients

The number of simultaneous users of a site is very challenging to characterize. Unlike other types of client-server architectures, the weight of an individual client on the Web server is quite small and short-lived. Connections to a Web server are traditionally stateless sessions that begin with an open from the client, a request for data, the server reply with data, and the session close. Depending on the speed of the network connection, the size of the data requested and the server load, this session can last from tenths of seconds to tens of seconds. The degradation of performance with increasing client load is best seen through average latency of data throughput, and must be therefore continuously be monitored.

7.1.8 Availability Considerations and Issues

In this section we discuss some considerations for scalability and availability.

7.1.8.1 Scalability

Scalability in VisualInfo/Digital Library systems is achieved by specializing the role of the archive system nodes and by distributing the upcoming system workload as uniformly as possible among the available nodes. Starting with a proposed configuration, you can expect a linearly scalable Digital Library hardware/software infrastructure, by just adding more nodes or mass storage subsystems to satisfy the following as required:

- I/O bandwidth
- Processor workload demand
- Communication facilities

7.1.8.2 Availability and Recovery

By making a redundant system design (with no single point of failure) and through the use of HACMP, high availability of the VisualInfo/Digital Library system may be achieved. In fact, a VisualInfo HACMP installation has been tried at a customer′s site.

There is no standard backup strategy defined for VisualInfo/Digital Library, nor does the product include backup software. Instead, all backup tools and strategies supported by the underlying database and operating system can be used, including automated "off-hours" backups and backup of ADSM. Recovery times are a function of the loss and the backup strategy and tools used. We suggest ADSM as a vehicle for a consolidated backup system, as it supports backup of data from servers and PCs.

7.2 Digital Video Asset Management System

In this section we help to size and quantify a Video Asset Management System solution based on IBM Digital Library. We focus mainly on the capacity and performance planning steps. We worked from a set of domain-specific requirements with consideration for storage capacity, data throughput (I/O and network subsystems) and user transaction load.

We give a general discussion of the system design in order to outline the context in which such a "digital video archive" will be embedded. The presented design sketch is based on an analysis of requirements collected from different European broadcast companies and their newsroom systems.

7.2.1 The EBU-News System Business Model

One source of daily TV news of the broadcast industry in Europe is the European Broadcast Union EBU-News System (Euros). Many broadcasters record the incoming news on tapes and store them for later usage on shelves in their video archive. The in-house news production is developed by a number of editors who use VCRs to preview the available news material. They create Edit Decision Lists of news clips for their daily production, that are broadcast later. All of these processes are usually based on analog techniques. During the past 3 to 4 years, due to the availability of compute power and storage capacity at affordable prices, the trend toward the digital domain is growing.

Using this scenario, we refer to such a Digital Video Asset Management System as being the workplace of TV program editors for their daily news production work, as well as the domain of the supporting IT personnel.

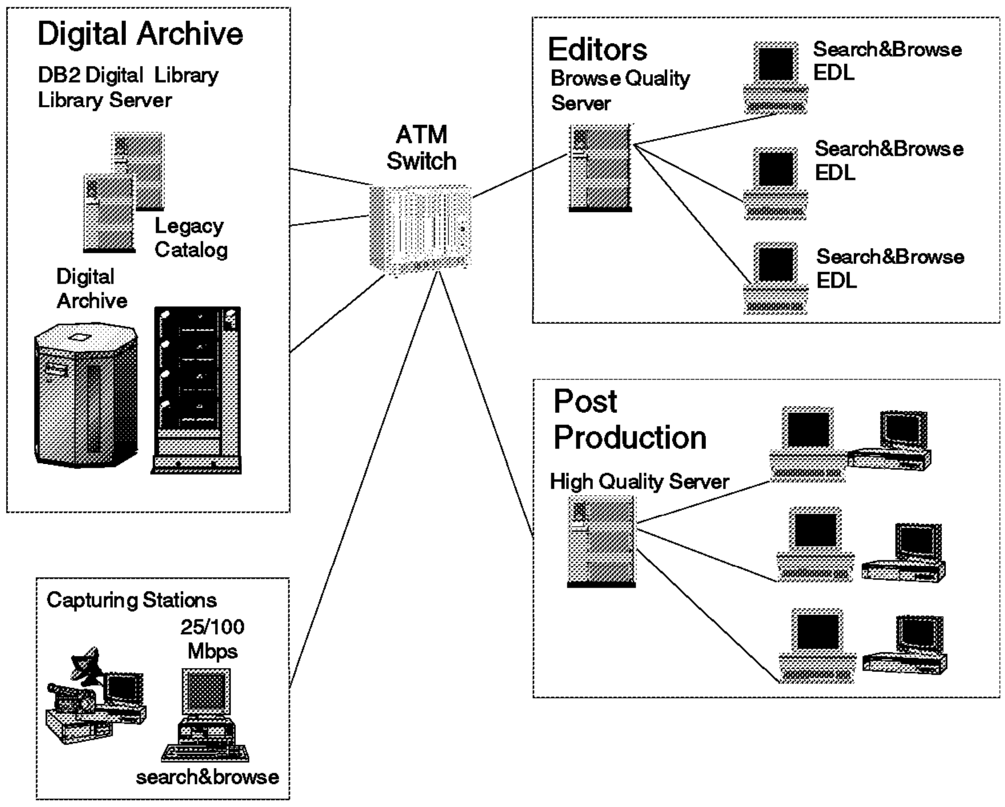


Figure 21. Sketch of a Newsroom System

Assume that this group of editors are our “digital archive customers” These customers are equipped with specific PC-based workplaces, all having multimedia capabilities and network access. We want to give them access to a central server, where the incoming news events are captured, encoded (compressed), logged, indexed, and made available for search and browse.

Their major requirement is: ″The effective online search of the news material for program selection and build, as well as the browsing of the low quality video material and pre-selection at the editor′s workstation″. The buzzwords are “server-based browsing,” which allows simultaneous multiclient access of the digital video material. This is one of the major advantages that digital video has over analog video.

This scenario is also valid for those cases where video material is produced by some other means, like a video edit station such as AVID Media Composer or NewsCutter.

It is important to note that the video material is not only digitized, but different replicas must also be created. These replicas must be encoded into different video formats and at different compression rates.

A key factor in the solution is that all work processes (implemented to support the implied business model) should always deal with metadata and browse quality high compressed video material, except for the cases of:

- Post-production
- Program transmission
- Long-term archiving

In most cases, little of the actual data should be physically moved. As you can see from Figure 21 on page 70, the different work processes also need different specialized servers that can be accessed through an interconnecting network with adequate throughput and bandwidth.

Table 8 gives a rough picture of the storage and bandwidth requirements of video assets compressed at different bit rates.

Table 8. Storage and Bandwidth Requirements of Video Assets

Encoding Format	**Description/Characteristics**	**1h (approx.)**
Uncompressed video	270 Mbps ~ 27 MB/s	~ 100 GB
DVC-Pro 4:2:2	High-quality material at 50 Mbps	~ 20 TB
MPEG2 4:2:2	High-quality material at 50 Mbps	~ 20 TB
Broadcast-quality material	MPEG2 4:1:0 at 15 Mbps (20:1 compression)	~ 7 GB
Broadcast-quality material	MPEG2 / 2 at 8 Mbps	~ 3.5 GB
Browse-quality material	MPEG1 ~ 1.5 Mbps	~ 0.7 GB

7.2.2 Overview of the Solutions Components

The Video Asset Management System is often the core part of the production infrastructure and its work processes. As such, it is important to have the complete system in mind when designing each single component with its defined interfaces for importing and exporting multimedia material.

Another important aspect is the indexing and logging of the video material. With the new search technologies, different indexing and search strategies can be adopted by using sophisticated search engines such as: Text Search Engines, Query By Image Content Engines, Query By Video Content Engines, Query By Audio Content. Therefore some effort must go into developing a data model in order to efficiently support the production chain.

The trend today is to turn some of the content data into metadata. For example, if there is a script associated with video material, this script can be used to "text index" the video. In a later phase, a "text search engine" can be used to search and retrieve the video asset based on the information given by the script.

Also, if "automatic shot change detection" and "key-frame extraction" techniques are adopted, than you can use the key frames to build a results picture gallery. This allows you to select and browse individual scenes or feed a "Query By Image Content" engine for image-indexing the video. In both cases, the key frames become metadata that can be used to search for video assets in the archive. The query technique is called "Query by Video Content."

The same logic also holds for audio. The trick is to reduce the audio, video, and image content into descriptive text data that can be indexed by a text search engine.

A word must be said about the project management aspects, when introducing such a "digital video archive solution" into an existing production infrastructure. Before the real project, you usually start with a pilot to do some preliminary investigation and to gain experience with the new techniques and technologies. Especially important is the investigation of how current production processes can be integrated and extended with the new technology.

Figure 22 on page 73 shows a schematic view of the involved system components. As you can see, the central controlling element is the IBM Digital Library. At the bottom (or backend) is the storage management system, which can be accessed only through the Library Management System.

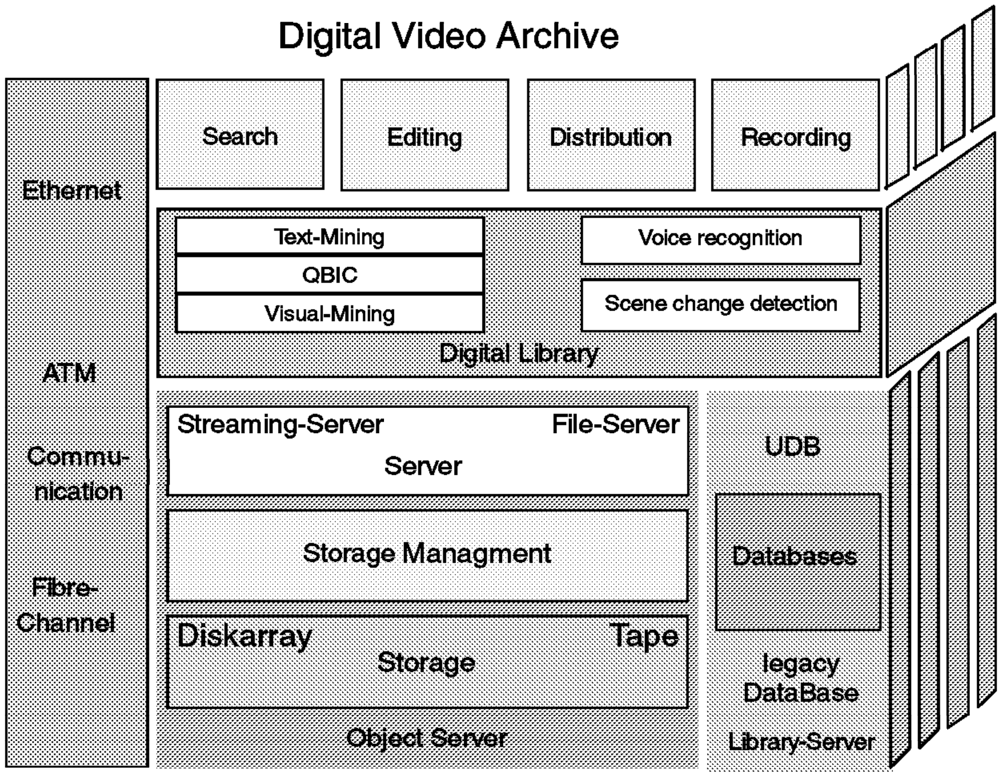


Figure 22. Digital Video Archive System

At the top or client side, you can find the single application components according to the single production processes:

- Capturing and indexing
- Editing
- Searching
- Distributing

The backbone of the entire system is the communication facilities that provide the connection between the single components. The communication infrastructure must be able to handle the communication requests from the single workplaces with the right semantics and adequate bandwidth.

7.2.3 Components of a Digital Video Archive System

The main system components are:

- Legacy Catalog System (every broadcaster has one!)
- Asset Management System based on IBM Digital Library
- Hierarchical Storage Management Subsystem
  - Workspaces

    The Archive will have different storage areas with different requirements, such as:

    - Studio production storage area
    - Dailies staging area

        - Long-term hierarchical storage subsystem

- Online server as required (such as Pluto Airstation, Louth Automation System, and others)
- LAN-based Browse Video Server
- Disk workspaces (staging areas for the video material)
- Automated tape archive as required
- Capturing and Recording Stations (SDI, SDDI, MPEG2, DVC-Pro, MPEG1, and others)
- Editing/Postproduction Stations (AVID Media Composer, NewsCutter, and others)
- Transmission Stations for distribution of the video material
- Network infrastructure

7.2.3.1 Storage and Throughput Considerations

Table 9 and Table 10 list some typical storage and throughput characteristics for digital data.

Table 9. Storage Needs

Item	**Description/Characteristics**	**Amount of Data**
Amount of video material	2230+970 min	~ 53.3 h
High-quality material	DVC-Pro 4:2:2 at 50 Mbps	~ 1.2 TB
Broadcast-quality material	MPEG2 ML&MP at 15 Mbps (20:1 compression)	~ 360 GB
Browse-quality material	MPEG1 or MPEG2 at 2Mbps	~ 57 GB
Hierarchical storage	Not considered here, but part of the design since it is key to the system	
Total	Storage needs	~ 1.7 TB

Table 10. Network Bandwidth Needs

Item	**Description / Characteristics**	**Rate of Data**
10 x search and browse WS	20 x 1.5 Mbps network bandwidth	30 Mbps
3 x transmission WS	3 x 15 Mbps network bandwidth	45 Mbps
2 x recording WS	2 x 1.5 Mbps network bandwidth	3 Mbps
2 x high-quality recording	DVC-Pro 4:2:2 at 50 Mbs	100 Mbps
Total	Network bandwidth needs	178 Mbps

7.2.4 Client Workstations

In this section we discuss the various types of client workstations which could make up this system.

7.2.4.1 Recording Station

This workstation will encode the original video material into an MPEG1 format at 1.5 Mbps using the IBM VideoCharger Version 2.0 for Windows NT using a Futuretel PrimeView realtime encoder card. During the encoding process, the digital video stream can be multicast "live," recorded onto disk, or multicast and recorded. It can then be transferred directly (staged) onto the primary Video Server (in this case a VideoCharger for AIX on a RS/6000 workstation) in order to be archived.

If other encoding formats are required, then the use of high-quality encoder systems should be considered. For example, the Vela encoder comes with the ability to do realtime ftp onto an AIX VideoCharger. This is the ability to encode from various video sources (analog and digital) at different compression rates and to transfer the encoded file in near-realtime. On request, the video stream is multicast in quasi-realtime (with a delay of < 10 sec) into the intra-subnetwork to a predefined audience.

This workstation will also provide the pre-indexing and logging, as well as the cuts of the material into single file segments as necessary. Indexing can be done using "automatic shot change detection" and "key-frame extraction," together with parametric and text data that can be keyed in manually. The key frames, along with the SMPTE Time Code (TC), provide the ability to jump into the video clip at the right position during browsing.

If desired, using a voice recognition module (IBM ViaVoice), you could also generate a transcript of the audio track to be used as text index metadata for research. A Digital Library client has to be installed on this system and an import facility for both the metadata and the content data must be developed. Most of the integration effort here is the implementation of the data model and the storage hierarchy through custom loader facilities; that is, loading of the metadata into the library and the search engines used, and loading of the video data onto the respective storage servers.

Table 11 lists the hardware and software requirements for the recording station.

Table 11 (Page 1 of 2). Hardware/Software Requirements

Component	**Description/Characteristics**
Hardware requirements	IBM-compatible Pentium II at 266 Mhz, 128 MB of main memory and dedicated SCSI attached disks, preferably configured with RAID striping for streaming. IDE drives should only be used if you plan on very small number of streams. Network adapter cards—Micro Channel or Peripheral Component Interconnect (PCI), a FutureTel Prime View MPEG1 realtime encoder adapter.

Table 11 (Page 2 of 2). Hardware/Software Requirements

Component	**Description/Characteristics**
Software requirements	VideoCharger Server for NT, Windows NT 4.0 with Service Pack 3, NTFS. For the Web server: Microsoft IIS for NT Servers or Peer Web Services (for NT workstations), Version 3 Latest FutureTel device drivers from FutureTel′s Web site (http://www.futuretel.com), Digital Library Version 2 with specified CSD.

7.2.4.2 Video Editing or Production System

The choice of workstation is up to the customer. Whatever system is chosen (for example, the AVID NewsCutter or MediaComposer) is OK. The type of effort that has to be done for the video archive is the integration work into the archive itself.

A Digital Library client has to be installed onto this system, and a check in/check out semantic has to be developed and implemented. Possible available options to import the video material are:

1. Use the IBM Digital Library support to load AVID OMF files.
2. Use VideoCharger staging/destaging facilities to upload at constant bit rate file transfer from capturing workstation to the video archive.

Care must be taken during the design of the subnetwork, because video staging/destaging will put a lot of load onto the network. The production systems must be isolated from the editor′s workstation in order to avoid network congestion.

7.2.4.3 Search and Browsing Station

This workplace is based on generic workstations which have intranet access to the Web-enabled digital video archive through an access control system. It provides support for the research of video assets using all available search engines and technologies.

If a legacy catalog exists, then part of the existing schema must be mapped onto the new one that must be developed. The new data model must complement and enhance the old one in order to support the new systems requirements.

Searching becomes a federated search, where a query must be first split into sub-queries and later submitted to the individual search engines. The single result lists in turn must be intersected and consolidated into one consistent hit list.

Very important is the availability of an efficient and appealing graphical user interface (GUI), especially for the user acceptance of the system.

The GUI is used not only for searching but also for the browsing of the low-quality video material. If necessary, these clients must also be supported though a wide area network (WAN).

Browsing videos is done with the IBM VideoCharger Player for Windows 95 and Windows NT that supports two viewing modes for video:

1. The standalone “helper” video viewer with a graphical user interface with a flexible display layout, control customization, and improved display status and statistics. This supports a draggable screen, video segment selection for repeated play, and changing video selections without reloading the player. The player also provides extensive controls, including fast forward, stop, rewind and auto replay for audio and fast forward, stop, pause and play video display control.
2. Plug-in viewing supports streaming video from an HTML page. The “plug-in” executes under the control of the browser and does not require an extra window for displaying the video. Mouse buttons provide player control.

A set of C APIs are provided to allow the development of customized applications and players. Table 12 lists the hardware and software requirements for the search and browsing station.

Table 12. Hardware/Software Requirements

Component	**Description/Characteristics**
Hardware requirements	IBM PC-compatible Pentium (R) 233 MHz, MMX, 64 MB RAM, 2GB HD, (for MPEG1 software decoding), 6MB disk space, audio adapter supported by ActiveMovie, network adapter or modem supported by Windows 95 or Windows NT TCP/IP (28.8 Kbps or higher).
Software requirements	Windows 95 or Windows NT operating system, VideoCharger Player for Windows 95 and Windows NT, HTML Web browser, Netscape Navigator 3.0 or later, Microsoft (R) Internet Explorer 3.0 or later

7.2.4.4 Video Asset Management System

IBM Digital Library will become the asset management system. Both the IBM Video Charger and IBM Video Charger MultiMedia Archive are fully integrated into Digital Library and as such they are under tight control. Table 13 list the hardware and software requirements for the video asset management system.

Table 13. HW/SW Requirements for the Video Asset Management System

Component	**Description/Characteristics**
DL Library Server	RS/6000 Mod F40, SMP, 1024MB I/O Subsys: SCSI UW 4x4 GBDisk Network: FastEthernet Graphic Adapter: any Monitor: any OS: AIX 4.3
DL Object Server	RS/6000 Mod F40, SMP, 1024MB I/O Subsys: SCSI UW 4x4 GBDisk Network: FastEthernet Graphic Adapter: any Monitor: any OS: AIX 4.3

7.2.4.5 Hierarchical File Management

Non-streaming data will be managed by standard Digital Library object servers which also integrate the IBM ADSM product. The use of an Automated Tape Archive, for example based on the IBM 3494/IBM Magstar, is suggested. For streaming data, the storage hierarchy is implemented by the Video Charger and the Media Archive (based on IBM′s Netstore product). In both cases the online cache should use the IBM SSA High Performance Storage Subsystem. For storage, IBM RAID systems are used. They are hot-swappable and allow storing up to 144 Gbytes per RAID array. That is up to 163 hours of browse-quality video material. The system is scalable up to 50 MPEG1 streams and is based on proven IBM RS/6000 technology. Note that 2 to 3 arrays can sustain up to 100 Mbps rough data, that is delivered using either ATM-155 or Fast-Ethernet-Adapters.

7.2.4.6 Online Distribution and Browse Server

VideoCharger Server supports Multimedia Archive access for the retrieval and playback of off-line assets, as well as the storage of assets on Multimedia Archive. VideoCharger Server can also be used with IBM Digital Library Version 2.1 for streaming video and audio objects over the network.

Given the listed application requirements, the browse-quality Video Server would be based on Video Charger Version 2.0 for AIX running on a RS/6000 SMP stand-alone system or in an RS/6000 SP node because of its ability to scale, as the need to support many LAN- or WAN-based multimedia applications arises.

Video and audio can therefore be delivered to clients attached to the Internet or an intranet. Many types of Constant Bit Rate (CBR) streaming data, including any combination of the following data streams can be delivered:

- Low Bit Rate (LBR) and audio-only, low bit rate format down to 28.8 Kbps to support modem access through the Internet
- LBR, Mid Band and High Bit Rate (HBR) video/audio
- WAV
- AVI
- MPEG1
- MPEG2 up to 8 Mbps

Table 14 describes a VideoCharger configuration.

Table 14 (Page 1 of 2). VideoCharger Configuration

Component	**Description/Characteristics**
Video Charger V2.0 for AIX. The recommended minimum hardware configuration is:	RS/6000 POWER, POWER2, or PowerPC uniprocessor, SMP, or SP system. 128 MB memory for all applications. Dedicated disk and disk controller (SCSI or SSA) for the multimedia file. SCSI-2 Fast/Wide or SSA disk drives for best performance. Network adapter cards—Micro Channel or Peripheral Component Interconnect (PCI). Industry Standard Architecture (ISA) is not recommended.

Table 14 (Page 2 of 2). VideoCharger Configuration	
Component	**Description/Characteristics**
The recommended minimum hardware configuration is:	AIX Version 4.2.1 server code HTTP server software (such as IBM Internet Connection) DCE 2.1 Client software (included with AIX 4.2.1) *Optional Software:* Digital Library Version 2 with specified CSD
Media Archive	IBM Video Charger Media Archive, IBM Netstore 1x 3494 ATL Library 4x 3590 Magstar Tape Drives

Network Infrastructure: Standard IBM Fast Ethernet or ATM Switches and adapter cards.

A structural view of the Digital Video Archive infrastructure is given in Figure 23.

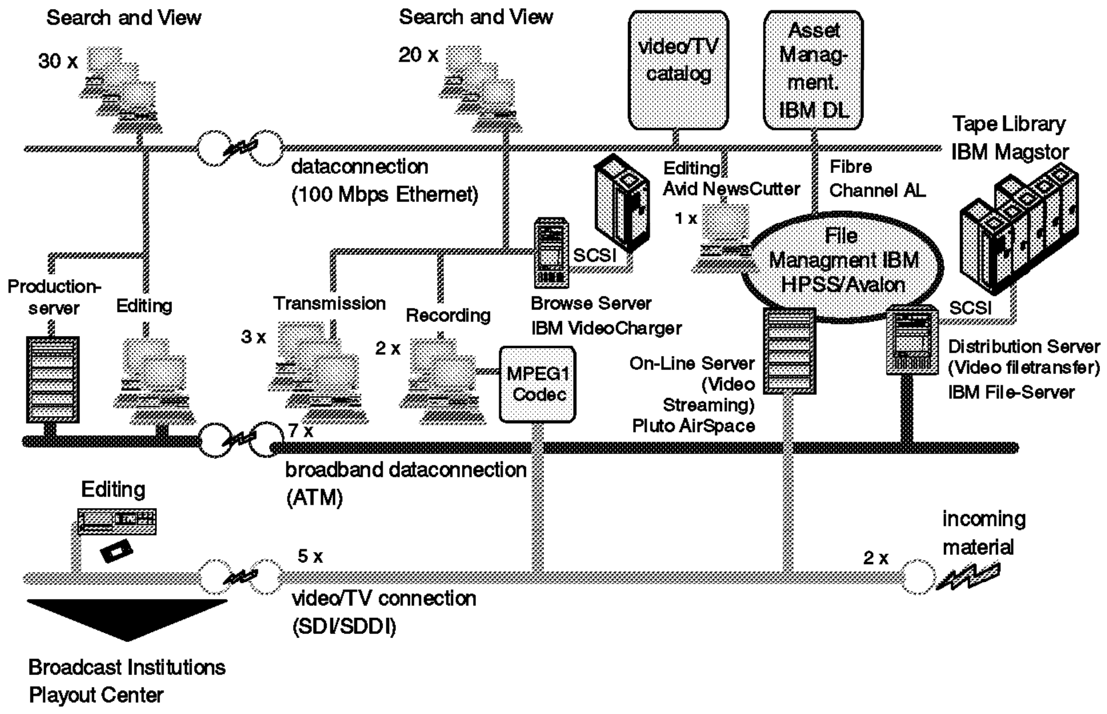

Search and View
30 x
Search and View
20 x
video/TV catalog
Asset Manag-ment. IBM DL
Tape Library IBM Magstor
dataconnection (100 Mbps Ethernet)
Editing Avid NewsCutter
1 x
Fibre Channel AL
File Managment IBM HPSS/Avalon
SCSI
Production-server
Editing
Transmission
3 x
Recording
2 x
Browse Server IBM VideoCharger
SCSI
MPEG1 Codec
On-Line Server (Video Streaming) Pluto AirSpace
Distribution Server (Video filetransfer) IBM File-Server
7 x
broadband dataconnection (ATM)
Editing
incoming material
5 x
2 x
video/TV connection (SDI/SDDI)
Broadcast Institutions Playout Center

Figure 23. A Typical Broadcast News and TV Archive Scenario

7.2.5 Summary of Solution Components Needed

Table 15 on page 80 contains a summary of the single system components with a description and task characteristics

Table 15. Single System Components

Component	Description/Characteristics	Implemented by
Asset Management System	Central catalog server of the Video Asset Management System. Central access point to the digital archive.	DL Library server
Browsing server, Distribution server	Browsing server to "stream" out the video assets to the requesting clients. Full integration with DL; access only due to the DL Library server.	Video Charger, Media Manager
Video archive	Long-term video archive for the high- and low-quality video material on HSM.	VideoCharger Media Archive, Netstore
Legacy catalog	If it exists, then it has to be integrated into the search servers.	TV catalog
Digitizing station	Recording, cutting in single news contributions. Indexing of the broadcast Videostreams (such as the EBU News Preview Systems).	Capturing station
Editor's work station	S&R Station, standard client machine for search and browse of the corresponding video material.	Search and Retrieve station (S&R)

7.2.5.1 Summary of Components

Note that the single system components are chosen based on the input data given for the resource computation. To size the system in a realistic way, a detailed requirements analysis has to be done.

Table 16 (Page 1 of 2). Sizing the System

Component	Description/Characteristics	Size /Throughput
Media Asset Management System		
DL Library server	RS/6000 Mod F40, SMP, 1024MB, I/O Subsys: SCSI UW 4x4 GBDisk Network: FastEthernet Graphic Adapter: any Monitor: any OS: AIX 4.3	12 MB 100 Mbps
DL Object server	RS/6000 Mod F40, SMP, 1024 MB, I/O Subsys: SCSI UW 4x4 GBDisk Network: FastEthernet Graphic Adapter: any Monitor: any OS: AIX 4.3	12 MB 100 Mbps
Digital Video Archive		
VideoCharger V2.0	RS/6000 Mod F50, SMP, 1024 MB, I/O Subsys: SCSI UW 4x4 GBDisk Network: 3xFastEthernet or ATM-155Mbps Graphic Adapter: any Monitor: any OS: AIX 4.3	12 MB 300 Mbps

Table 16 (Page 2 of 2). Sizing the System

Component	Description/Characteristics	Size /Throughput
Online Storage	IBM 7133 SSA RAID Mod 600 4x 7133 Filesystem throughput	144 GB 570 GB 300 Mbps
Media Archive	IBM Video Charger Media Archive, IBM Netstore 1x 3494 ATL Library 4x 3590 Magstar Tape Drives	13 TB 8 MB
Client Workstations		
Capturing Station	PC Pentium II, 333 MHz, 128MB RAM I/O Subsys: SCSI UW 4x4 GBDisk Network: PCI FastEthernet Graphic Adapter: any Encoder: FuturetelPrimeView (MPEG1) Monitor: any OS: WinNT 4.0	
Search and Retrieve Station	PC Pentium 233 MHz, MMX, 64 MB RAM, 2GB HD, Ethernet, WinNT/Win95, 17″ Monitor	
Network Infrastructure		
Network	IBM Fast Ethernet switching hubs or an IBM ATM Switch	

Part 3. Performance Tuning

- General tips
- ADSM tips
- Platform-specific tips

Chapter 8. General Performance Tips

In this chapter we have collected tips and techniques for enhancing the performance of your Digital Library / VisualInfo system and related components. In subsequent chapters, we cover platform-specific techniques.

8.1 Tips from the Product Developers

This section contains tuning and performance tips that the team gathered from sources at the IBM Santa Teresa Development Laboratory (STL) and around the world.

Performance tuning for any product addresses the stage when an installed system experiences a need for improved performance; for example, the response times have lengthened. For VisualInfo and Digital Library, this is a major concern. Tuning will always be required when the system gets loaded with more objects and users. Begin your performance planning early so that you do not run into problems later on.

We have outlined tips and methods for improving performance. These tips are gathered from the labs and from people in the field who have actually gone through these exercises.

Tip: If you are designing your own application, you should consider the possibility of writing your programs to perform functions in parallel.

8.1.1 Introduction

VisualInfo/Digital Library makes available a very rich set of functionality and it is tempting to use this to the fullest. However, this could have a detrimental effect on end-user response times and server resource consumption. High functionality usually carries a penalty in performance impact, particularly in the areas of workbaskets, workflow, foldering structures and multiple index classes. In some cases the penalty may be acceptable, in others it may not.

Tip: Design with performance in mind. It is very difficult to change a poorly performing design after it has been implemented.

If either the library server or the object server is remote from the client machines, make sure there is adequate network capacity. This applies to the library server and the object server. The library server can send a half megabyte or more of data to the client to open a workbasket. It is not unusual to see data transfers of 10-20 kilobytes between the client and the library server. Having the required bandwidth available is essential and may be very expensive in countries where the telecommunications providers are either a monopoly or government controlled, leading to customer dissatisfaction if the requirement has not been anticipated.

8.1.2 Systems Management

VisualInfo/Digital Library are not always treated as serious mission-critical systems. Many VisualInfo/Digital Library systems are being installed with very little attention to testing and systems management. This is changing, and now an increasing number of customers consider their VisualInfo/Digital Library system to be their most critical system.

However, it is not uncommon to come across a situation where the low cost of the hardware for PC-based client server systems leads to an attitude that no systems management is necessary. Most times this is an oversight, arising from lack of experience with mission-critical client-server systems.

If the company′s business is going to depend on the VisualInfo/Digital Library system, then it is important that adequate attention is given to backup and recovery (of all hardware components: processor, DASD, optical, network) testing and change control. Just because it is possible to make changes easily on the production system does not mean that this is where they should be first implemented. An adequate test environment coupled with good change management is essential and needs effort and time to establish. All these factors affect the availability of the system, and therefore, in the customers′ perception, the performance of the system.

Equally important is the need to thoroughly test the VisualInfo/Digital Library system after development and before going live. Storage management, backup and recovery, optical install and backup, and performance all need testing thoroughly. Enough time should be allocated to rectify any problems that appear.

8.1.3 Network Considerations for Remote Systems

Most people are aware that a system which has an object server remote from the clients and connected by a WAN line instead of a LAN may perform more slowly. However, this situation is also true of a system where the library server is remote from the clients.

Clients transmit and receive large volumes of data between themselves and the library server. In addition, an end-user response may involve a number of message pairs being exchanged between the client and the library server.

As an example, a search of an index class which resulted in a list of 50 items being returned involved 50 kilobytes of data being transmitted to the library server and 150 kilobytes being received from it. There were 12 message pairs involved, all for one response. The end-user response time was significant, approximately 21 seconds. These volumes of data are not unusual. In fact it would be unusual for less than about 10-20 kilobytes to be exchanged per end-user response.

The performance documents produced by the Santa Teresa lab for Digital Library (which also apply to VisualInfo) gave some figures for data transmitted between client and server for some APIs. The volumes are well above what might be expected. The problem is exacerbated at times (in the client application) by unnecessary APIs being issued. APAR fixes are available for some situations. You should discuss this with your IBM support representative, who may check with RETAIN for further assistance.

Therefore, it is obvious that a reasonably high speed link is required between VisualInfo/Digital Library clients and the library server, in addition to anything required for object server traffic. A 64 kilobit link would require 25 seconds of line time alone (versus less than a second for a 16 megabit Token-Ring) for the previous example, assuming no queuing because of other traffic.

In addition, where a TP link is involved, there are other factors to be considered. Most TP links are optimized for 3270-type traffic, where a single message pair is required for each end-user response. In these situations, the delays in the network are normally not significant. These delays (attention delays, polling delays and node response times) may typically be of the order of 0.4 to 0.6 seconds per message pair. When an end-user response involves 12 message pairs, these delays can mean an additional 5 to 7 seconds in response time, in addition to the time taken for the data to be transmitted.

It is usually not possible to do much about the amount of data transmitted between the client and the library server, but it may be possible to tune the TP network, although this may involve higher utilizations for some of the resources (such as NCPs) and higher speed links than the networking people may be happy with. In addition, it may be possible to use data compression on the links, but this may involve higher processing loads on the client workstations or other complications.

Summary: If you are using remote communications, make sure your network is tuned for client-server traffic and can deliver the response times you expect.

8.1.4 Basic Terminology

We would like to clarify certain terms, which sound similar but have entirely different meanings. These are database terms and VisualInfo/Digital Library terms which tend to get confused. This is because the terminology used by VisualInfo/Digital Library in the library server resembles the underlying relational database.

8.1.4.1 Index Class

An *index class* is a user-defined data structure that contains the catalog data of objects stored in the object server. To the underlying database manager, each Index Class is a relational database table.

Internally, all index classes are assigned an eight-character name AVTnnnnn, where nnnnn can range from 00001 to 99999. Therefore, index classes are also called *AVT tables*.

AVT00001 to AVT00007 are internal to VisualInfo/Digital Library. User-defined index classes begin from AVT00008. As each new index class is created, the numeric part of the table name is incremented by one and assigned to the newly created index class.

8.1.4.2 Index Field

An *index field* is one of the attributes that constitutes the Index Class data structure. An index class contains one or more index fields. To DB2, an index field is a column in the corresponding AVT table.

Internally, each index field is assigned a 14-bytes long alphanumeric name ATTRIBUTEnnnnn, where nnnnn range from 00001 to 9999.

ATTRIBUTE00001 to ATTRIBUTE00043 are reserved for internal use. User-defined index fields begin from ATTRIBUTE00044. An index field is also known as a *key field*.

8.1.4.3 Database Index

In a relational database, an index is created from a column or columns in a database table. In other words, a column in a database table can be "assigned" as a database index. The index is usually created by the database administrator for improving query performance. If a column is frequently used in the search queries, it is usually a good candidate for creating a database index out of it.

8.1.5 Performance Tuning

On a VisualInfo/Digital Library server, performance tuning can be done at the following three levels:

1. Operating system tuning
2. Database tuning
3. Subsystem parameters (library server and object server)

We shall go into details on each of these three levels for the library server and the object server.

Notes:

1. The terminology and examples used are mostly from the workstation-based DB2. However, since all RDBMS have evolved from the same relational model, the techniques used for tuning the DB2 database should also be applicable to Oracle-based VisualInfo/Digital Library installations and to the mainframe-based VisualInfo/Digital Library which is based on DB2/MVS. The differences are at the terminology and command syntax level, but not the techniques.
2. Performance bottlenecks can be caused by inadequate or improperly sized hardware giving inadequate processor capacity or an insufficient amount of system memory. Throughout this section, we assume that the system is properly configured to support the user workload, and we limit our discussion to performance enhancements that can be made by tuning existing hardware. It could be that upgrading or migrating from a particular platform to another may improve performance, but those options are handled here.

8.2 Tuning Your Library Server

Now we examine some tips and techniques for tuning the library server, using our three topic areas: operating system, database and application.

8.2.1 Operating System Tuning

The operating system-level tuning can be divided into the following four areas: CPU, I/O, memory and communications. As mentioned previously, if proper capacity planning were done before acquiring hardware, CPU and system memory should not cause any performance problem. So operating system-level tuning for the library server involves mostly communications tuning and I/O tuning.

8.2.1.1 Communications Tuning

Usually, the default communications adapter setting is sufficient for the library server running on the system. However on AIX, during peak load hour, the TCP/IP transmit queue may overflow. Hence, it is recommended that the TCP/IP transmit queue size be set to the maximum allowed by the system.

The steps involved in tuning the AIX TCP/IP transmit queue size are as follows:

```
Smitty --> Devices
--> Communication
--> Token Ring Adapter (or the adapter used)
--> Adapter
--> Change / Show Characteristics of a .. Adapter
--> <select the correct adapter>
--> Change transmit queue size to 160
--> Press <Enter> to execute the change
```

8.2.1.2 I/O Tuning

Usually, I/O tuning involves distributing I/O activities to as many physical disks as possible, and placing the most active file system on the fastest physical device. We limit our discussion to the file systems that are directly related to the library server.

Distributing I/O: Distributing I/O involves detecting the file systems generating most of the I/O activities and placing each of the file systems on separate physical disks (if multiple disks exist on the system). If the system has multiple I/O adapters, then I/O performance can be further improved by distributing the active file systems among the adapters.

The library server has two major file systems which generate most of the I/O activities. These two file systems are the *database logs* and the *database tables*. By default, DB2 places both database logs and database tables on the same file system. For better performance, these two file systems should be placed on separate physical disks or I/O adapters. Changing the DB2 log path requires updating the library server database configuration file. The steps involved in changing this path are covered in 8.2.2, "Database Tuning."

Placing Active Files on the Fastest DASD: If multiple types of DASD exist on the system, I/O performance can be improved by placing the most active file systems on the fastest DASD. With proper database tuning, the most active file system in the library server should be the file system defined for database logging. Therefore, the log file system should be placed on the fastest physical disk(s).

8.2.2 Database Tuning

This is probably one the most important tuning areas for VisualInfo/Digital Library. In this section, we discuss several database tuning areas that have a significant impact on library server performance.

Attention

Customer Experience:

Here is an experience from a customer who was having problems which resulted in system outages. They were reorganizing the library server data base each night, but they overlooked several important elements.

- Default database indexes were used for each VisualInfo index class. These defaults are not usually created for optimum client query performance.
- In the `reorg` command, rather than select the option **reorganize by primary key**, they used the option **free up space**, which did not reorganize.
- They had not used the output of the `reorg check` command to verify results. (However, if they had done the other two items, this step would not be needed.)

Once they fixed these three elements, the system outages went away.

Important: Database tuning described in this section can permanently change the database. Therefore it is very important to take a full backup of the databases (both library server and object server) before attempting any tuning technique described in this section.

8.2.2.1 Tuning DB2 Database Buffer Pool

The buffer pool is probably the single most important database tuning area for DB2.

DB2 data and index tables reside on DASD, and are divided into 4 K pages. Data/index pages must be brought into memory before DB2 can process them. Accessing data pages from DASD is performance-expensive and can take several thousands of CPU cycles, while accessing data already in memory takes only a few CPU cycles. DB2 buffer pool is a memory block reserved for DB2 to cache data pages and index pages. The size of buffer pool is determined by the database parameter, BUFFPAGE. Each BUFFPAGE is 4 KB in size and the default for this parameter is set at 1000. The goal of buffer pool tuning is to increase the DB2 *buffer pool hit ratio*. The buffer pool hit ratio is the percentage of time that the database manager did not need to load a page from disk to service a page request.

Buffer pool tuning involves monitoring the buffer pool hit ratio and increasing the parameter BUFFPAGE, if the ratio is low.

8.2.2.2 Monitoring the DB2 Buffer Pool Hit Ratio

Monitoring the DB2 buffer pool hit ratio requires collecting buffer pool usage statistics. The steps for collecting the DB2 buffer pool statistics are as follows:

1. Connect to the library server database using the following DB2 command: db2 connect to <library server name>
2. Turn on the buffer pool monitor: db2 update monitor switches using bufferpool on
3. Clear the buffer pool statistic counter: db2 reset monitor for db <library server name>

4. Generate a database snapshot report:
   db2 get snapshot for db on <library server name> > snapshot.rpt

5. When step 4 is completed, a database snapshot report is generated, and the report is saved in a file called snapshot.rpt.

Buffer pool statistics can be found in the middle of the report. To calculate the buffer pool hit ratio, we need the following four statistics from the snapshot report:

- Buffer pool data logical reads - call this *A*
- Buffer pool data physical reads - call this *B*
- Buffer pool index logical reads - call this *C*
- Buffer pool index physical reads - call this *D*

Item A and item C are the number of data and index pages that DB2 did not have to load from disk, because they are already cached in the buffer pool. The condition is called *buffer pool hit*. Item B and item D are the logical reads, which are, respectively, A and C plus the number of data and index pages which DB2 had to load from disk, because they were either never paged in or were paged out or replaced by other data pages.

The buffer pool hit ratio is derived from the following formula:

```
buffer pool hit ratio = (1-(B+D)/(A+C)) * 100%
```

When the sum of buffer pool data and index physical reads (B+D) approaches 0, the buffer pool hit ratio will approach 100%. The goal of buffer pool tuning is to tune the size of buffer pool, so that the bufferpool data and index physical reads will be as close to zero (0) as possible. If the bufferpool hit ratio is substantially less than 100%, then the database configuration parameter, BUFFPAGE, needs to be increased.

The steps for changing the BUFFPAGE are as follows:

1. db2 connect to <library server name>
2. db2 update database config for <library server name> using buffpage nnnn, where nnnn is the new number of buffpage to be set
3. Stop library server.
4. Run the library server bind utility. On NT and OS/2, the Library Server bind utility is found in the Utility folder. On AIX, the bind utility is the shell script, frnbind.lib, under $FRNROOT directory.

Note: The new BUFFPAGE setting will not become effective until the library server is stopped, the databases are rebound, and all the applications connected to the library server database are disconnected.

The following command may be used to force disconnection of all applications:
db2 force application all

Note: Defining a large bufferpool size without enough physical memory on the system can cause the system to become memory-bound. If the library server is the only major application running on the system, the bufferpool size can be as big as 50% of the physical memory. For example, a system with 512 MB of memory can have BUFFPAGE set to 60,000 pages, which is roughly 240 MB. Any setting beyond 50% of physical memory is not recommended.

8.2.2.3 Creating Database Indexes.

If any of the index fields are frequently used in search requests against the library server, those index fields should be used to create database indexes. Database indexes can be created either through the system administration client, or through a direct DB2 SQL command.

For details on the steps involved in creating a database index through the system administration client, refer to *Image Plus VisualInfo System Administration Guide*, SC31-7774.

Attention

Customer Experience:

Here is an experience from a customer who was having performance problems on an AIX server. (This tip applies to all platforms.) They had not created database indexes, so each document access resulted in a table scan (looking sequentially through the table for the requested entry). Once they implemented these indexes, access time dropped from over 30 seconds to just a few seconds.

They had to follow these one-time procedures:

- Instead of reorganizing the tables by primary index (ITEMID), they changed the reorg utility to physically reorganize the database tables by the index that is most frequently used in forming sequential search queries.
- They ran the AIX runstats utility.
- They regenerated the VI DLL files. This caused the static queries to become updated with the new INDEXNAME values. Then they ran the library server bind utility.

This customer also increased the BUFFPAGE parameter to about 50% of the RAM on the system. (For a heavily loaded system, up to 75% would not be unusual). Therefore, it is important to have a lot of RAM.

The steps involved in generating a database index through DB2 SQL commands are as follows:

1. db2 connect to <library server name>
2. db2 ″create index <index_name> on <fully-qualified table name>(<column-name>)″

   For example, if a database index SSN for the index field, social security number (ATTRIBUTE00046) is to be created, then the command syntax would be: db2 ″create index SSN libuser.AVT00009 (ATTRIBUTE00046)″ where libuser is the library server user ID.

   Again, note that the index field, Social Security Number, is known as ATTRIBUTE00046 to DB2.

When a database index is created, a DB2 *index table* is created. An index table is a table with two columns. The first column is the index itself and the second column is a pointer to the physical location of the record that contains the particular index value. If a search request provides a specific value, such as SSN=′123-45-6789′, then DB2 can locate the record associated with the given

Social Security Number by using the index table alone, a procedure that is known as *index search*. Index search is very efficient for the following two reasons:

1. An index table is usually small enough to fit in to the DB2 bufferpool, so index searches involve very little, if any, I/O.
2. Index searching is done using a binary search algorithm; therefore only a very small portion of the index table needs to be scanned to locate the value. From the pointer stored with the index value, the data page which contains the record can be brought into memory in one I/O cycle, if it is not already there.

In contrast, without a database index, the same request has to go through a table scan. In a table scan, DB2 may need to read the entire table. Since data pages must be fetched into DB2 memory buffer from disk before DB2 can process it, a table scan against a large table can take several minutes (or even hours) to complete. When a table is small enough to fit in DB2 bufferpool, a table scan may not cause significant performance problems. However, as the size of the table grows, it is less likely for the entire table to fit in the bufferpool, and therefore users will experience gradual performance degradation because of the increased number of I/O cycles needed to complete the search.

The failure to create a database index for the index fields frequently used in searches probably constitutes most performance PMRs reported by customers.

8.2.2.4 Creating an Index for Every Field

In some installations, all index fields of an index class are used in some searches. In these cases, a frequently asked question is: can a database index be created for every index field? The answer is: it depends. The creation of a database index is not “free.” Each database index causes an index table to be created, and each additional index table will incur the following costs:

- Additional disk space is needed to store the index tables. When paged into the DB2 memory buffer, each index table competes with other database tables and index tables for the DB2 memory buffer.
- Every update (including insert, delete and update) to the table will be applied by DB2 to *every* index table.

If the index class is fairly static, (many read only accesses and very few updates), then only the first type of cost is involved. The benefit of creating additional database indexes usually outweighs the cost of needing more disk space and DB2 memory buffer. However, if there are a substantial number of updates to the index class, then the existence of many index tables can substantially slow down the table update response time. The decision now becomes a trade-off between read response time and update response time.

Fortunately, database index creation is not irreversible. Additional database indexes can be created to avoid table scan, and when the response time for updates becomes unacceptable, then the DB2 drop index statement can be used to remove selective indexes.

8.2.2.5 Regenerating the DLLs

After creating additional database indexes, the DLLs which contain the static SQLs for accessing the index classes need to be regenerated. Regenerating DLLs forces DB2 to use the newly created database indexes to access the tables. The steps needed to regenerate the DLLs are as follows:

1. Shut down the library server.
2. Remove the old DLLs from the DLL directory. On AIX, the directory path is $FRNLOCAL/<library server name>/dll. On Windows NT, the directory path is $FRNROOT/<library server name>/dll.
3. db2 connect to <library server name>
4. db2 ″update sbtclassdefs set dllstatus=-1 where dllstatus=0″
5. db2 ″update sbtviewdefs set dllstatus=1″
6. Start the library server and the new DLLs are regenerated as part of the library server startup.

8.2.2.6 Determine Which Index to Use for Reorganizing

If a database table has more than one database index defined, you need to choose which database index to reorganize the table by. For the library server, if a database index is created for an index class as described, the index class will have at least two database indexes, the default primary index, ITEMID, plus the indexes created as described.

The VisualInfo/Digital Library System Administration clients provide optimization utilities for scheduling a routine table reorg. The database indexes that used for reorg are picked up from one of the library server tables, SBTSBTABLES. By default, each index class is reorganized using the default index, ITEMID. If an index class needs to be reorganized by other index, the SBTSBTABLES needs to be modified, or a DB2 reorg command need to be issued separately.

SBTSBTABLES is a system table, which contains an entry for each table that library server owns. The database index to reorganize each table is maintained in the table under the column INDEXNAME. By default this column has a value ′NONE′ for each index class. This value is substituted with the DB2 primary key when the reorg utility is called. For the library server, ITEMID is the default primary key field for all the index classes. Therefore, unless the column is modified, all Index Classes are physically reorganized in the order of ITEMID.

To reorganize an index class using an index other than ITEMID, the SBTSBTABLES needs to be updated. The steps to do that are as follows:

If we want AVT00009 to be reorganized by Social Security Number, and the index name for Social Security Number is SSN, then the steps involved in updating the SBTSBTABLES are as follows:

1. db2 connect to <library server name>
2. db2 ″update SBTSBTABLES set indexname=′SSN′ where tablename=′AVT00009′″

If you do not know what indexes are defined for the table, DB2 system table syscat.indexes can be queried to find all the indexes defined for a table, such as AVT00009: db2 ″select indname from syscat.indexes where tabname=′AVT00009′″

Alternatively, you can write your own reorg utility by calling the DB2 reorg command directly instead of using the utility provided by VisualInfo/Digital Library.

8.2.2.7 Reorganizing a Table

Performing a table reorg, updating table statistics, and rebinding should be done at regular intervals, or when performance begins to degrade.

The physical order of table records can have a significant impact on database performance. Choosing a database index for the DB2 reorg utility to improve performance requires some understanding of how the DB2 bufferpool works, whether the index is non-unique, or whether wild card search queries exist. Usually, index fields used for wild card searches are the top candidates to be considered for use in a database reorg. Using the previous example index class, AVT00009, a wild card search might be caused by a user needing to find all records in AVT00009 with the last name beginning with “SMI.” Using the VisualInfo/Digital Library client application basic search, the entry field next to last name would be filled in with “SMI*,” making this a wild card search.

A wild card search can result in a substantial amount of I/O because many data pages contain matching records which may need to be paged into the DB2 bufferpool. This type of search can affect the performance of the overall system, because the memory buffer could be replaced by the data pages which meet the wild card search. Then the data pages for other queries have to be paged out and later paged in when needed. This could substantially degrade the performance for other queries.

To avoid performance degradation, the index field frequently used in the wild card search should be defined as a database index. Also, the table (the VisualInfo/Digital Library Index Class) should be physically organized in the order of the index field. If the table is not organized by the index field, each matching DB2 record could reside on a separate data page, and potentially result in a separate I/O for each matching record. If there are 2,000 records matching the search request, that would be 2000 I/O cycles, which could cause the system to become I/O-bound, at least temporarily. If the index class is organized by the index field, these 2000 records may reside in a few consecutive data pages, and with DB2 sequential pre-fetch, only a few I/O cycles are needed to bring every matching record into the bufferpool.

Performance tuning often involves trading off between alternatives. If the same index class has more than one index field used for wild card searches, then (since a table can only be physically organized by one database index) the benefits of using either one needs to be carefully analyzed and the performance must be closely monitored. You may want to try to organize the table by one index first and then by the other index after a few days and so on for every potential candidate, to compare the difference in performance.

If there is no index field involved in wild card search, then the index field defined as non-unique should be considered for database reorg. Searches on non-unique index fields are similar to wild card searches, because the records matching the search conditions could be located in multiple data pages and require multiple I/O cycles to bring them all into the DB2 bufferpool. Reorg by the index will cluster all records with the same value in a few consecutive pages and thus can take advantage of DB2′s sequential pre-fetch to reduce the amount of I/O and therefore improve the response time.

If most searches are of ′equal′ type, such as SSN=′134-34-4567′, and the index field is unique, where no two rows are allowed to have the same value, then it does not matter which index to reorg the table with. At most, only two I/Os are needed to bring the datapage into memory, one for the index page once the other for the actual data page.

Once a reorg index is identified, the next step is to decide how often to reorg the tables. The reorg utility is expensive to run and it could take hours to run if a table is large. Also while a table is being reorganized, it may not be available for access. Therefore, to improve system availability, determining the frequency of running a reorg utility is important. DB2 provides a utility called REORGCHK which checks the cluster ratio of a table against each database index defined and recommends whether the reorg utility is needed.

The steps for running reorgchk are as follows:

1. db2 connect to <library server name>
2. db2 reorgchk update statistics on table all

Step 2 generates a report which contains statistics such as cluster ratio, table sizes for each table against each database index defined and so on. The value under column heading F4 in the report is the cluster ratio of the index; when the value in this column is below 80%, the reorg column would have two asterisks (**), indicating a reorg of the table is recommended.

Note: Since only one index can be used for a table reorg, normally the low cluster ratio for all the database indexes may be ignored, except the one identified for reorg.

The reorg of a table can be done either through the Systems Administration utility or a direct DB2 call. To reorg all tables belonging to the library server, it may be easier to use the Systems Administration utility. However, if only one or two tables need to be reorganized, using the DB2 reorg command may be more efficient. The steps for running the DB2 reorg utility for a specific table such as AVT00009 using the LAST_NAME index are as follows:

1. db2 connect to <library server name>
2. db2 reorg table <libuser_id>.AVT00009 index LAST_NAME where libuser_id is the library server user ID.

   The ID is required, since reorg requires a fully qualified table name to be provided.

To aid the DB2 optimizer in choosing an access path for optimal performance, DB2 stores crucial table statistics, such as cluster ratios and table sizes, in its system catalog. The system catalogs do not get updated until the DB2 runstats utility is called. Therefore, table reorg *must be followed* by running the DB2 RUNSTATS utility.

Updating of table statistics can be done either through the system administration client or a direct DB2 call. For the steps involved in updating table statistics through systems administration, refer to *ImagePlus VisualInfo System Administration Guide*, SC31-7774.

The steps to update table statistics through a direct DB2 call are as follows:

1. db2 connect to <library server name>

2. db2 reorgchk update statistics on table all

or

1. db2 runstats on table libuser_id.AVT00009

Note: The reorgchk command with update statistics option is equivalent to DB2 RUNSTATS. As specified in the example, reorgchk can update table statistics for all tables in one command, while RUNSTATS requires one RUNSTATS command per table.

8.2.2.8 Spread I/O by Changing the DB2 Log Path

By default, DB2 assigns both database table and database logs to the same file system. It is recommended that the library server database tables and database logs should be assigned to two different file systems, preferably on separate disks. The steps to change the DB2 logs to a different file system (log path) are as follows:

1. db2 connect to <library server name>
2. db2 update database config using newlogpath <new_log_path> where <new_log_path> is the filesystem created specifically for database logs.

This change will not take effect until all applications are disconnected from DB2. This can be done as follows:

1. Stop the library server.
2. Issue a db2 force application all command.

8.2.3 Tuning of Library Server Parameters

Here are some tips for tuning the library server.

8.2.3.1 Setting Search to case_sensitive

The case_sensitive search setting is probably one of the most important library server tuning parameters. The default setting for this parameter is case_insensitive and is set in the default library server configuration file, IBMCONFIG. The default setting for this parameter will not cause any performance problems, if all the searches are against numeric index fields. However, if a search is against an alphabetic or alphanumeric index field, the case insensitive setting will cause a table scan.

As we mentioned previously, the performance of table scan is proportional to the size of the table. If the performance of a table scan is not acceptable, the parameter *must be set* to case sensitive. For steps to change the search to case sensitive, refer to *ImagePlus VisualInfo System Administration Guide*, SC31-7444.

Once the search is set to case sensitive, the case of the search value provided must be exactly the same as that stored in the library server database; if not, the search will fail. If maintaining the case of the index field is not important, you can change the index value in the table to upper case and inform the user to enter upper case index values for all future searches. DB2 provides a built-in function, TRANSLATE, which can update a specific column with one single SQL call. For example, the steps for translating ′Last_name′ in index class AVT00009 to upper case are as follows:

1. db2 connect to <library server name>
2. db2 ″update AVT00009 set ATTRIBUTE00046=translate(ATTRIBUTE00046)″

3. db2 commit

8.2.3.2 Setting Number of Library Server Child Processes

Library server child processes are processes that handle VisualInfo/Digital Library client requests. The number of child processes, also called the number of “pies” in older releases of VisualInfo, can have significant impact on the library server performance. If the system is not bounded by memory, then the higher the number of child processes, the greater the number of concurrent users the library server can handle. *Concurrent users* here means users that issue requests to the library server simultaneously, in other words, the users that press **<ENTER>** at the same time. If there are more concurrent users than the number of child processes, some users will have to wait until one of the child processes free up. The users whose requests are queued will experience slower response time.

The default number of child processes is set to 5 in the default library server configuration file, IBMCONFIG. The situation at your installation may require this number to be changed. The steps involved in changing the setting are as follows:

1. Start the library server.
2. Bring up the system administration client.
3. Logon with the user ID FRNADMIN, or any user ID with system administration privileges. If your library server is already running with a customized configuration file, then double-click on the current configuration file and skip steps 4 to 6. If the library server is running with the default configuration, IBMCONFIG, then go to next step.
4. Click the plus sign (**+**) next to the library server to display all options under library server.
5. Right-click the first displayed option **Configurations**, and click **new** to make a new copy from the default configuration file. Note that the default file, IBMCONFIG, can be viewed but not modified; therefore, this step is needed.
6. In the displayed window New Configuration Properties, enter a new configuration name.
7. Update the entry field next to Maximum processes to the desired number of child process setting. This entry can be found in the middle of the page, and by default the entry should contain the number 5.
8. Click **OK** to save the new configuration file.
9. Update the library server start-up file to use the new library server configuration. The configuration file that the library server used at start-up time is set in the environment variable, FRNDBCFG.

On AIX, one way to update this setting is to edit the file $FRNLOCAL/frnsetup.lib and change FRNDBCFG=IBMCONFIG to FRNDBCFG=newconfig, where “newconfig” is the new configuration name entered in step 5.

On Windows NT, one way to change it is through the library server properties window. To reveal the properties window follow the steps outlined below:

1. Right-click **Start** and select **Open All Users**.
2. Right-click **Programs** and select **Open**.
3. Right-click the VisualInfo or Digital Library folder and select **open**.

4. Right-click on the library server icon and select **Properties**
5. On the tab Shortcut on the library server Properties window (in the middle of the page), the entry field next to the Target contains the default IBMCONFIG. Replace IBMCONFIG with the new configuration name.

The following points need to be considered before changing the number of child processes setting:

- The number of child processes setting is used at the time of the library server initialization to determine how many child processes to pre-start. In other words, updating this number during run time does not become effective until the library server is restarted. Therefore, to ensure adequate system performance, it is important to understand the workload, especially the peak workload, and to configure the number of child processes accordingly. We recommend set the number of child processes to 30, if the library server machine has at least 256 MB of memory.
- Each child process is connected to a DB2 agent at the time of the library server initialization. The maximum number of DB2 agents is determined by the MAXAPPLS parameter in the library server database configuration file. The default MAXAPPLS is set at 40, which should be sufficient if the number of child processes is less than or equal to 38. If you plan to configure more than 38 child processes, then the MAXAPPLS parameter should be set higher accordingly.

  Note: You should allow at least two extra DB2 agents for DB2 ad hoc queries and database optimization utilities. The following command sequence can be used to modify the MAXAPPLS setting:

  1. db2 connect to <*library server name*>
  2. db2 update db cfg for <*library server name*> using maxappls nnn where nnn is the new MAXAPPLS setting, which should at least be equal to the number of child processes + 2.

8.2.3.3 Setting FRNHEAPSIZE

FRNHEAPSIZE determines the size of shared memory segment that library server processes can use during run time for interprocess communication. Each VisualInfo/Digital Library component has its own FRNHEAPSIZE setting.

The setting for this parameter has very little impact on performance. However, when the number of child processes increases and the number of concurrent users keeps every child process busy at the same time, the library server could crash because of lack of dynamic storage. Therefore, if the system has enough memory and the number of child process is set at 30, the FRNHEAPSIZE for library server should be set to at least 10240.

On AIX, this parameter is set in $FRNLOCAL/frnsetup.lib. On Windows NT, it is set in the $FRNROOT\frnnstls.bat file. You can edit the corresponding file directly to change the setting.

8.2.3.4 VisualInfo—Balancing Folders and Index Classes

VisualInfo provides a very comprehensive database model which any customer can customize extensively to meet their own requirements. Unfortunately, in the absence of any guidelines, this capability can lead to systems being designed which use all the functions available to provide what are perceived to be elegant business solutions. The downside of this approach can often be a poorly performing system.

As a basic principle, the number of index classes used should be kept to an absolute minimum. (This is the experience that we have heard from people in the field. On the other hand, the labs say that, theoretically, there could be any number of index classes and this should not make any difference). Each additional index class involves additional DB/2 tables and potentially much more complex application coding. Searches may be done across multiple index classes which inevitably leads to table scans and long response times. We have seen small systems (less than 100 users) with as many as 50 index classes defined. This number is generally excessive and could be avoided at the initial design stage.

Often a high number of index classes arises because of a decision to have an index class for each type of document contained in the system—this is the intuitive approach. A better approach is to create a new index class when the document types are too different to group them into the same index class.

Besides to limiting the number of index classes, another good design principal is to limit the complexity of the folder structure. A manual filing system will inevitably have a single-level folder structure; rarely will there be folders within folders. Given the facilities of VisualInfo, it is often tempting to create quite complex folder structures with, for example, policy folders contained within customer folders. Again this approach seems elegant from the application design perspective, but it does introduce significant complexity in the application for handling the logical units of work which arise in maintaining these structures. More complex structures complicate the problem even further.

Unless there are really significant reasons, only a single-level folder structure should be considered and you might even consider whether to use folders at all. It may be possible to collect documents into pseudo-folders by using a column of the document index class to hold a clustering key.

8.3 Tuning Your Object Server

Now we discuss techniques to tune the object server.

8.3.1 Operating System Tuning

VisualInfo/Digital Library Object Server stores objects as flat files in the filesystems; therefore, I/O tuning is essential for object server performance. Tuning the object server I/O requires understanding of the filesystems that belong to object server and how object server generates I/O to each of the filesystems.

8.3.1.1 Distributing I/O

VisualInfo/Digital Library Object server owns the following four major filesystems:

- Staging filesystem
- Destaging filesystem
- DB2 database
- DB2 database logs

These four filesystems can become active concurrently, so if there are enough hard disk drives in the system, these four filesystems should each be placed on its own physical disks. At the minimum, Staging and Destaging filesystems should be separated into different physical disk drives. If these two filesytems are placed on the same set of physical disks, multiple processes can generate I/O requests against the same physical disks, creating unnecessary I/O bottlenecks on the system.

8.3.2 Database Tuning

Object server database tuning is not as complicated as the library server. In general, the default database indexes defined for object server database tables are adequate.

8.3.2.1 Reorganizing the BASE_OBJECTS Table

If the BASE_OBJECTS table is reorganized when the cluster ratio for the primary index OBJ_OBJECTNAME is less than 80%, it will improve performance.

To check the cluster ratio for this table, follow the steps below:

1. db2 connect to <object server name>
2. db2 reorgchk update statistics on table <obj_userid>.BASE_OBJECTS, where <obj_userid> is the object server administrator user ID.

Note: By default, BASE_OBJECTS table has two database indexes defined; the index which needs to have high cluster ratio is the primary index. On the REORGCHK output, this index should have a creator name SYSIBM and index name that begins with SQL, followed by a 15-digit number.

When the cluster ratio of this table is less than 80%, the table needs to be reorganized, table statistics need to be updated, and finally the packages need to be rebound.

Table reorg can be done either through the VisualInfo/Digital Library System Administration database utilities or through direct DB2 calls. To reorg all tables belonging to the Object Server, it is easier to use the System Administration utility. The VisualInfo/Digital Library System Administration utility uses the database table SBTSBTABLE to determine how to reorg the tables that belong to the object server. In particular, the INDEXNAME column in the SBTSBTABLE defines by which database index the object server table will be reorganized. The value in this column should contain NONE for all the rows. This value is substituted with the DB2 primary key when the reorg utility is called.

In some installations, the INDEXNAME column contains a null value; thus, the database utility will not run properly. So it is important to check the value in this table. The step for checking the SBTSBTABLE is as follows:

1. db2 connect to <object server name>
2. db2 ″select tablename, indexname from sbtsbtables″

If the indexname columns are blank, then the table needs to be updated for the VisualInfo/Digital Library database utilities to work properly. This problem is fixed in the latest CSD. If the CSD has not been installed as yet, the following command sequence to update the table may be used:

1. db2 connect to <object server name>
2. db2 ″update sbtsbtables set indexname=′NONE′″

If only the BASE_OBJECTS table needs to be reorged, it is more efficient to use the DB2 reorg command. The steps for running the DB2 reorg utility for BASE_OBJECTS table are as follows:

1. db2 connect to <object server name>
2. db2 reorg table <objuser_id>.BASE_OBJECTS index SQLnnnnnnnnnnnnnnn, where objuser_id is the Object Server user ID. The ID is required, because reorg requires a fully qualified table name to be provided and SQLnnnnnnnnnnnnnnn is the primary index of BASE_OBJECTS table.

The primary index of the BASE_OBJECTS table can be found in the syscat.indexes table using the following commands:

1. db2 connect to <object server name>
2. db2 ″select indname from syscat.indexes where tabname=′BASE_OBJECTS′″

To aid the DB2 optimizer choose an access path for optimal performance, DB2 stores crucial table statistics such as cluster ratios and table sizes in its system catalog. The system catalogs do not get updated until the DB2 runstats utility is called. Therefore, table reorg *must be followed* by running the DB2 RUNSTATS utility.

Updating of table statistics can be done either through the system administration client or a direct DB2 call. For the steps involved in updating table statistics through systems administration, refer to *ImagePlus VisualInfo System Administration Guide*, SC31-7774.

The steps to update table statistics through direct DB2 calls are as follows:

1. db2 connect to <object server name>
2. db2 runstats on table objbuser_id.BASE_OBJECTS

Finally, for DB2 to use the updated statistics to access the tables, the object server bind utility must be run. On Windows NT and OS/2, the object server bind utility can be found in the VisualInfo/Digital Library Utilities folder. On AIX, the bind utility is the shell script frnbind.obj found under the $FRNROOT directory.

If the System Administration Optimization utility is called, then DB2 RUNSTATS utility and package rebind are automatically called. For more information on the optimization utility, refer to *ImagePlus VisualInfo System Administration Guide*, SC31-7774.

There are other tables in the object server database too, but the sizes of these are usually very small and the content fairly static; therefore, tuning these tables is not essential.

8.3.2.2 Increasing the BUFFPAGE Parameter

The same understanding of the buffer pool is required for this as mentioned previously for the library server. For detailed steps on instructions and implications, refer to 8.2.2.1, “Tuning DB2 Database Buffer Pool” on page 90.

8.3.2.3 Assigning DB2 logs to a Separate Disk

The same understanding of the I/O is required for this as mentioned previously for the library server. For detailed steps on instructions and implications, refer to 8.2.2.8, "Spread I/O by Changing the DB2 Log Path" on page 97.

8.3.3 Tuning of Object Server Parameters

There are four major activities in object server operation that compete for system resources. These four activities are:

- Storing/retrieving objects
- Destaging objects from staging area to destaging area
- Purging objects from staging area after the objects are destaged
- Migrating objects from destaging area to the next level of storage hierarchy

Tuning the object server requires some understanding of how each of these activities consume system resources. Storing and retrieving are handled by the object server child processes. Destaging of objects is handled by a single destager process, purging of objects is managed by a single purger process, and two migrator processes handle migration requests.

On AIX, the process name for each of these object server components is: frnxlbpr, frnxlbde, frnxlbpu, and frnxlbmg. On NT, the process (thread) names are frnnlbpr, frnnlbde, frnnlbpu and frnnlbmg. These process/thread names may be used to monitor how each process/thread consumes CPU resource.

The number of object server child processes is a tunable parameter. The steps for changing the number of child processes are documented in *ImagePlus VisualInfo System Administration Guide*, SC31-7774. Like the library server, the object server child processes are pre-started at object server startup time and cannot be changed dynamically. However, the destager, purger and migrator can all be disabled after the object server has been initialized. When the destager, purger and migrators are active at the same time, they can jointly consume a substantial amount of CPU resource and could cause the system to become CPU-bound.

8.3.3.1 Tuning Object Storing and Retrieving

A store request causes the object to be written to the staging area, and a retrieve request can generate a read request against the staging area (if the object is already in staging area) or a write to staging area (if the object has been destaged or migrated and purged from staging area). Many concurrent users issuing store or retrieve commands simultaneously can cause the staging disk to be almost 100% busy.

8.3.3.2 Isolating Staging Area

The staging filesystem will be the single busiest filesystem in the object server. Not only are the store/retrieve requests constantly reading from/writing to the staging area, but when the destager and the purger are activated, additional reading and writing is performed on the same filesystem. Therefore, the I/O performance on the staging filesystem should be closely monitored and action should be taken when it becomes a bottleneck. If possible, you should define a dedicated filesystem for staging only. With a dedicated filesystem, it is easier to use an AIX operating system command such as `df` or the NT `dir` command to determine how full the filesystem becomes and to adjust system activities accordingly.

8.3.3.3 Disabling Destager, Purger and Migrator During Peak

Each of these three processes, when activated, can consume a large portion of the CPU resource, leaving very few CPU cycles for storing/retrieving.

The commands for disabling the destager, purger and migrators on AIX are:

- `frncmd <object server name> destager disable`
- `frncmd <object server name> purger disable`
- `frncmd <object server name> migrator disable`

Replace `disable` with `START` or `ENABLE` to activate the process immediately to or enable the process for activation according to the schedule setting in the object server configuration file.

To operate in this setting, the staging filesystem must be large enough to be able to contain all the objects which may be required to remain in the staging area (the daily input volumes, and in some cases, the daily request volumes from users) until the destager, purger and migrators are enabled. If for some reason these processes cannot be disabled during peak workload hours, then a symmetric multi-processor (SMP) with two or more CPUs can be used to ease potential CPU contention.

Note: The VisualInfo/Digital Library Object Server running on an SMP machine requires the latest CSD to be applied.

8.3.3.4 Keeping Frequently Accessed Objects on DASD

Retrievals from optical are significantly slower than from DASD, therefore, frequently accessed objects should be kept on DASD and the optical device should be used for archiving objects that are rarely accessed.

8.3.3.5 Setting the Number of Child Processes

Most of the VisualInfo/Digital Library requests involve only the library server. Only the store/retrieve requests involve the object server, so the object server child processes should be substantially less that the number of library server child processes. The general rule of thumb recommended by the lab is that the object server′s number of child processes should be two-thirds those of the library server.

8.3.3.6 Setting vol_numbuckets in base_volumes Table

In general, accessing files from a large filesystem is not as efficient as accessing files from a smaller filesystem. Vol_numbuckets in the base volumes table defines the number of buckets, or paths, used to store objects on the volume. By default, this column is set to one. With this setting, all objects stored in the volume are under the same directory path. If there are a larger number of objects/files stored in the volume, the search for objects in this file system by the object server can be slow.

If the vol_numbuckets is greater than 1, an equal number of subdirectory paths will be created and the objects will be spread equally across these subdirectories. The number of buckets can be changed only through direct DB2 calls as follows:

1. db2 connect to <object server name>

2. db2 ″update base_volumes set vol_numbuckets = nn″ where nn is the number of subdirectories to be created.

Note: The latest CSD (at least October 1998) must be installed for the multiple buckets to work properly.

8.3.3.7 Tuning the Staging Area

You can use the following techniques to ensure that your staging area provides the performance you need.

Avoid Filling the Staging Area: When the staging area becomes 100% full, the object server stops accepting requests and the entire system can hang.

Avoid Too Many Files: Most object server operations require a directory search to be made. A large number of files in the staging area slows down a directory search. In addition, if an object server recovery is needed after an abnormal object server shutdown, it could take hours. During the recovery process, the object server is not available for processing requests, so keeping a large set of files in the staging area could mean many hours of object server down time.

For example, our test shows that with around 21000 100 KB files in the staging area, recovery could take more than an hour. The recovery time increases exponentially with the number of files in the staging area.

Allow One Day′s Worth of Objects: You should have a staging area large enough to contain all the objects that will be accessed in a single day. This means that all the new objects that need to be stored and old objects that will be retrieved from destaged or migrated areas should ideally reside in the staging area for the length of the working day at least. With a large enough staging filesystem, the destaging, purging and migrating of objects can be postponed until off-hours.

However, the more the files in the staging area, the longer it will take for the object server recovery to run if an abnormal termination occurs. Therefore, both the storage/retrieval response time and the tolerable object server down time should be considered to derive an acceptable object destaging/migration strategy. For example, if the user can tolerate less than one hour of object server down time, then destaging and purging should be scheduled to maintain a staging area with no more than 20000 files at anytime, regardless of the daily volumes.

Maintain an Object Store Rate: If the staging area is not large enough to contain all the files to be accessed in a working day, then try to maintain an object store rate that does not outrun the destager.

The object store rate is around 20000 100 KB files per hour.

Note: The object server configuration file contains several parameters that affect the destager, purger and migrator performance. The parameters and default setting are as follows:

parameters	**default**
purger cycle	5 minutes
migrator cycle	30 minutes

destager cycle	15 minutes
stager batch size	50
migrator batch size	50

The first three parameters determine how long the three processes wait in between processing of objects. The batch size determines the number of files to process per batch. These five parameters can be changed to meet the processing requirement of each installation. Refer to *ImagePlus VisualInfo System Administration Guide*, SC31-7444 for more detail. For example, when the staging area is close to 100% full, and you want to increase the destager and purger rate, then set the purger and destager cycle to 1 minute and the stager batch size to 1000. This should improve the destaging and purging performance.

As mentioned previously, a good way to ensure that all object server processes can obtain necessary CPU resource when needed is to install the object server on an SMP machine (the latest CSD needs to be applied).

8.4 Tuning Your Client

Now let us discuss ways to tune the client system.

8.4.1 Perfomance Tuning for Large Objects

The following tuning parameters must be adjusted when importing and exporting large objects on a client workstation:

- The FRNCOMP environment variable must be set to CLIENT.
- The FRNCOMPID environment variable must be set to DAEMON.
- The FRNHEAPSIZE environment variable must be 4 times the size of the object divided by 4096, because the unit of FRNHEAPSIZE is 4 KBbytes.

All three of these environment variables must be set.

For example, importing/exporting a 50 meg object requires the settings:

```
FRNCOMP=CLIENT
FRNCOMPID=DAEMON
FRNHEAPSIZE=50000
```

The following system time-out values must be adjusted to handle a time-out during import/export.

FRNFMTIMEOUT environment variable is the minute base to control the time-out value on the Folder Manager layer. The library server and object server time-out value is adjusted using system administration client tools. As a general rule, the FRNFMTIMEOUT value should be set to the object size divided by 3000000. Since it takes about one second to transfer 500 KB and the unit of FRNTIMEOUT is minutes, then 500 KB times 60 seconds is 3000000.

8.4.2 Control the Number of Items in a Table of Contents

By default, Table of Contents windows display up to 100 items. If an index class or search results contain more than 100 items, only the first 100 are listed. You can control the number of items that display in a Table of Contents by setting a parameter in your Client for Windows .ini file.

The steps to do this are as follows:

1. Edit the VIC.INI file, located on your workstation in the \Windows subdirectory. If you are running NT, the VIC.INI file is in the \WINNT subdirectory.
2. Add the following:
   - Preferences
   - MaxTOC=xxx where *xxx* is any positive integer. If you set MaxTOC=0, your Table of Contents lists all items.

Tables of Contents on your workstation will now include up to the number of items that you assigned to MaxTOC.

The default value for MaxTOC is 100. There is no upper limit on MaxTOC, but the higher you set it, the slower your system′s performance will be when you retrieve a Table of Contents.

8.4.2.1 Creating a Suitable Number of Multiple Image Files

Scanning in Windows creates a mix of multiple image files that each have multiple pages. Scan′s default action creates the first image file with from 1 to 5 pages, and any subsequent image files with from 1 to 15 pages. Here are a couple of examples to demonstrate:

```
Batch size = 3 pages
Files created = SCAN0000 (3 pages)

Batch size = 14 pages
Files created = SCAN0000 (5 pages), SCAN0001 (9 pages)

Batch size = 42 pages
Files created = SCAN0000 (5 pages), SCAN0001 (15 pages),
SCAN0002 (15 pages), SCAN0003 (7 pages)
```

This default behavior can be modified by a knowledgable user by modifying their VIC.INI file as follows:

```
Scan
PagesInFirstFile=9
PagesInRemainingFiles=50
```

If you were to set both variables to 1, each SCANnnnn file would contain only one page. If PagesInFirstFile is set to zero, only one file will be created. If PagesInFirstFile is greater than zero and PagesInRemainingFiles is zero, there will be a maximum of two files created, the first one having from 1 to the value of PagesInFirstFile pages, and the second file containing all the rest.

This feature was implemented to optimize object server-to-client transactions. The object server performs best when one very large object is transferred. The client, however, could display the first page while the subsequent files are being loaded. This behavior allows us to create a happy medium, according to the particular situation at hand.

8.4.2.2 Increasing MTU Size on Client

If you are using Windows NT clients on a token-ring, consider increasing the MTU size to 2000 bytes for a 4 Mbps token-ring and 4000 bytes for a 16 Mbps token-ring. If you are using Windows NT clients over Ethernet, use the default MTU size of 1500 bytes.

8.4.2.3 Making a System-Assigned Workbasket

If a workbasket contains a large number of items, consider making it a system-assigned workbasket.

8.4.2.4 Disable Workbasket Count Interval

If the workbasket contains a large number of items, keep the item count interval set high, or turn it off. When the workbasket count is refreshed, the library server performs SQL queries to determine the number of items in each workbasket. The number of LSFRs for each workbasket is 3 + (number of items in workbasket / 50 + 1). This number of LSFRs is executed for each client workstation in each specified refresh time interval.

Disable the item count when the TOC of workbaskets is displayed.

8.4.2.5 Changing the Session Timeout Parameter

For high activity users, set the session timeout (SESSION_TIMEOUT) to 3000 milliseconds or 3 seconds. Set it to 0 for all other users. The session timeout value determines how long the conversation with the server is to remain open after the server responds to the client request.

8.4.2.6 Set the ROWLIMIT Value in the CNTL Table

Set the ROWLIMIT value in the CNTL table to reflect your search requirements, if different from the default value of 5000.

8.5 Network and LAN

LAN server performance must be considered in an environment where there is a heavy load on the LAN, or where large files are being transferred between server and requester. A number of changes should be made to the object server for the OS/2 IBMLAN.INI file and on the optical server.

- MAXTHREADS

  The parameter MAXTHREADS defines the number of threads within a requester that can be used to handle simultaneous network requests. This is most likely to occur when multiple applications are simultaneously making requests. This parameter must be altered in concert with both MAXCMDS and NUMWORKBUF. If MAXTHREADS is changed, the individual values for MAXCMDS and NUMWORKBUF should be greater than or equal to MAXTHREADS.

	Default	Recommended
MAXTHREADS	10	20
MAXCMDS	16	32
NUMWORKBUF	15	30

- SESSTIMEOUT

This parameter specifies the time, in seconds, that the requester waits before disconnecting a session from the server that is not responding to a request. If the server workload is extremely heavy, it may be necessary to increase this value.

```
                 Default  Recommended
SESSTIMEOUT        45        300
```

- WRKHEURISTICS

  The parameter WRKHEURISTICS enables the raw read and write server message block (SMB) protocol that allows transfer across the LAN without the SMB header. The protocol transfers large files directly between server memory and the workstation cache. Change bits 11, 12, and 13 from the default value of 1 to 0.

Consider the performance of the network. Avoid going through bridges and routers if possible. If you are using a 16/4 token-ring adapter, be sure it is moving data at 16 Mbps and not 4 Mbps.

8.6 What Happens During a Logon

Sometimes, we need to know what processing is involved in a logon to VisualInfo. This may be in order to solve a connectivity problem or to understand the performance of the logon process.

Ip2Logon is an API used by client applications to log on to the server. It comprises the steps necessary to connect to the configuration server, as well as the logon to the library server.

SimLibLogon is the API which connects a client to a library server, SimLibLogon consists of three stages:

- Attach: This involves no order codes being passed, but contains one message pair between client and library. If the library is set to “Reject incoming requests,” it is the attach that fails. It may be that the attach is also used by the Windows client when **Library Server Connection Verification** is selected (to be checked).
- Connect/Endtrans: This involves a simple request block being sent to the library, which retrieves no data. The user ID and encrypted password are contained in the Connect order. (There is an open item under investigation where the library rejected the logon attempt at this stage, so check the item status if you experience this condition.)
- The final stage is the central purpose of the SimLibLogon API. It returns the required data to the folder manager cache. (There are two types of calls to SimLibLogon; the usual one does the work to populate the folder manager cache.) It includes a request block containing the following orders:

```
Order 1034 Length 286 CONNECT_ORDER_CL
Order 1356 Length 50 SQUERY_ORDER_CL
Order 1356 Length 43 SQUERY_ORDER_CL
Order 1356 Length 43 SQUERY_ORDER_CL
Order 1356 Length 147 SQUERY_ORDER_CL
Order 1356 Length 50 SQUERY_ORDER_CL
Order 1356 Length 50 SQUERY_ORDER_CL
Order 1356 Length 53 SQUERY_ORDER_CL
Order 1356 Length 31 SQUERY_ORDER_CL
```

```
                Order 1965 Length 43 VIEWSEARCH_ORDER_CL
                Order 1965 Length 43 VIEWSEARCH_ORDER_CL
                Order 1356 Length 31 SQUERY_ORDER_CL
                Order 1965 Length 36 VIEWSEARCH_ORDER_CL
                Order 1134 Length 11 ENDTRANS_ORDER_CL
```

As can be seen, most of these requests are Static Queries, while the rest are ViewSearches (queries on AVTs). They each retrieve data from some of the library tables as follows:

```
Static Query 14 SBTVIEWDEFS,SBTVIEWATTRS,SBTATTRDEFS,
                SBTCLASSATTRS,SBTNLSKEYWORDS
Static Query 12 SBTPRIVILEGES,SBTPATRONS
Static Query 45 SBTPATRONS
Static Query 19 SBTCHECKEDOUT
Static Query 3  SBTCLASSDEFS,SBTNLSKEYWORDS,SBTACCESSLISTCODES,
                SBTOBJECTSERVER,SBTCOLLNAME,SBTPATRONS
Static Query 4  SBTVIEWDEFS,SBTNLSKEYWORDS,SBTACCESSCODES
Static Query 18 SBTCNTL,SBTACCESSCODES
Static Query 46 SBTACCESSCODES,SBTITEMS
ViewSearch      AVT00002,SBTITEMS
ViewSearch      AVT00003,SBTITEMS
Static Query 47 AVT00003,SBTLINKS
ViewSearch      AVT00004
```

Notes:

1. AVT00002 is the table defining workbaskets.
2. AVT00003 defines workflows.
3. AVT00004 defines the properties of index classes.

Where more than one table name is given, a join is performed to get the result table.

A communications trace (either TCP/IP or CM/2) can be taken of the logon process if it is not performing well. It can be analyzed using a program which will give the actual execution times of each order in the request block, and this will show how the overall logon time is made up.

In general, note that some of the queries shown require a join with the SBTITEMS table. If the library server database has not been reorganized since installation, it may be that the join is not optimized for the size of table that SBTITEMS has now become. Reorganization of the table causes the plan for the static query to be rebound and the DB2 optimizer should then ensure an index search rather than a table scan for the items table.

We have seen a problem on an AIX Library server where, despite this, the optimizer still causes a table scan of SBTITEMS. This was fixed in service pack U449586 for DB2/6000.

8.7 Table Scan Performance Diagnostics

This section deals with the Table Scan Performance problem diagnosis—what data to collect and how to interpret the result. Performance problem diagnosis is a big topic. Complete coverage of the subject is beyond the scope of this section, so we focus only on a specific topic: table scan performance problem diagnosis.

A complete discussion of VisualInfo/Digital Library performance problem diagnosis is an extensive topic. Not all of it is documentable science; some falls more in the domain of knowledge and experience. The purpose of this section is to provide a high-level introduction regarding the general steps toward diagnosing performance problems caused by an undertuned database.

8.7.1 Gathering System Resource Usage Statistics

When a performance problem is discovered, the first step should always be to get a snapshot of the system resource usage statistics, especially statistics on CPU, I/O, and memory. Although resource usage statistics do not help resolve a performance problem, they may help narrow the problem scope and determine which tools to use. Tools for collecting system resource usage statistics exist on all the operating system platforms supported by VisualInfo/Digital Library:

- For OS/2, there is SPM/2.
- Windows NT, there is Performance Monitor.
- For AIX, there are vmstat and iostat.

8.7.2 Interpreting the Statistics

This section demonstrates how to use the AIX commands `vmstat` and `iostat` to detect table scan related performance problems.

Note: The first statistics from `vmstat 5` and `iostat 5` are a summary since the last system reboot, and should be discarded.

Following is an edited output on AIX from the command `vmstat 5`.

The command causes the average system resource usage statistics within the last 5 seconds to be printed, as shown in Figure 24.

```
kthr   memory       page                      cpu
----   ----------- ------------------------- -----------
r  b   avm    fre  re  pi  po  .......        us sy id  wa

0  0   25875  3334      0   0                  3  5 83  10
0  0   26121  2745      0   0                 10  9 76   6
0  0   26219  2196      0   0                  6  7 35  52
0  0   26577  2366      0   0                 11  8  2  79
0  0   26798  2324      0   0                  5 12  0  82
0  0   27173  2024      0   0                 13 19  0  68
2  0   27751  2327      0   0                  5  5  9  81
1  0   27993  2500      0   0                  5  4 10  82
```

Figure 24. Output of vmstat 5 Command

This output from `vmstat` is a typical library server system statistics pattern caused by DB2 going through a table scan. The 0 under the pi and po columns indicate that there is no page-in and page-out, which implies that the system is not limited by memory. The us and sy columns under CPU indicate that the total consumed by the user and the system is around 30% of the available CPU power. The column wa indicates that there are outstanding I/O requests while the CPU is idle.

The high number in the wa column implies that there is a large amount of I/0 activity and the high I/O lasts for more than 30 seconds. If there are no other activities going on in the system at that time, we can make the assumption that the performance problem could be caused by DB2 going through a table scan. To double-check that our assumption is correct, we can recreate the problem and issue iostat 5 to find the disk which is causing the high I/O.

Figure 25 shows an edited iostat 5 output.

```
disks:  % tm_act     kbps

hdisk0     0.0      0.0
hdisk1     0.8      4.8
hdisk2    81.4    240.8
```

Figure 25. Output of iostat 5 Command

8.7.3 Queries Which Cause Table Scan

The % tm_act column indicates that the hdisk2 was 81.4% busy during the last 5 second interval. The iostat 5 command during a DB2 table scan should display this pattern of statistics for several consecutive 5 second intervals. If hdisk2 is where the database filesystem resides, then we are fairly sure that table scan is what is causing all the high I/O activity in the system.

If you know which library server search request caused the table scan, you can proceed to 8.7.4, “Fixing a Table Scan Problem” on page 115 to resolve the issue. However, in a production system there might be many users issuing different requests to library server, and pinpointing which request causes the table scan may not be easy. When this is the situation, the DB2 event monitor tool can be quite useful.

There are two command sequences. The first one captures the event monitor trace, and should be invoked before attempting to recreate the problem. The second set of command sequences should be invoked after the problem is successfully recreated. For ease of use, you can put these two sets of command sequences into two separate command files. The command sequence for capturing the event monitor trace is shown in Figure 26.

```
db2 connect to <Library Server Name>
db2 create event monitor evm1 for database, statements write to file
     '<directory path>'
db2 set event monitor evm1 state = 1
```

Figure 26. Capturing Event Monitor Trace

Note: <Library Server Name> is your library server database name, while <directory path> is a valid directory for which the db2 instance has read/write permission.

The formatting command sequence set is shown in Figure 27.

```
db2 connect to <Library Server Name>
db2 set event monitor evm1 state = 0
db2evmon -db <Library Server Name> -evm evm1 > evm1.out
db2 drop event monitor evm1
```

Figure 27. Event Monitor Formatting

The output is written to emv1.out. On AIX, you can issue a command against evm1.out such as:

```
   grep ′Rows read:′ evm1.out ]more
```

This statement will quickly identify whether there is any table scanning. If a table scan exists, the number that follows Rows read: will be close to the number of rows in the associated database table. The output file evm1.out also contains a beginning and ending timestamp, CPU usage time and so on for each SQL executed. The CPU cost of each SQL and the elapsed time of each SQL can be examined to determine which SQL is causing a performance problem. You can then find the package name and section number from evm1.out and use the db2expln tool to examine the access plan and tune the SQL.

A typical table scan entry looks like Figure 28.

```
163) Statement Event ...
Application Id: *LOCAL.DB2.971009165957
Sequence number: 0001
Statement Identification
Statement Type: Static
Operation: Fetch
Section number: 14
Application creator: USERID
Package name: FRNV0092
Cursor Name: TCURSOR8V1
Timestamps
Start Time: 10-14-1997 14:08:05.240992
Stop Time: 10-14-1997 14:09:04.326047

CPU times
User CPU time: 0.000000 seconds
System CPU time: 0.000000 seconds
Statement Activity
Fetch Count: 0
Sorts: 0
Total sort time: 0
Sort overflows: 0
Rows read: 962553
Rows written: 0
Internal rows deleted: 0
Internal rows updated: 0
Internal rows inserted: 0
```

Figure 28. Table Scan Entry

The fifth line from the bottom, Rows read: 962553, indicates how many rows DB2 has to read to process the particular query. The package name and the section number of the package which causes the table scan is recorded on line 7 and on line 9. You need to use the db2expln tool to find the exact SQL statement which caused the table scan. The syntax is:

```
 ′scale=0.7′.
db2expln -d <Library Server Name> -p FRNV0092 -c libuser -s  14 -o /tmp/exp.92
```

The output from db2expln is saved in exp.92, which gives the syntax of the query and the DB2 access plan for the query. See Figure 29 on page 114.

```
Section = 14:
SQL Statement:

  SELECT ICV00010.ITEMID:
  FROM ICV00010, SBTITEMS:
  WHERE ((SBTITEMS.SEMTYPE = :svar_12____1_2 ) AND
          (TRANSLATE(ATTRIBUTE00050) = :hvar_12____1_3 )) AND
          (SBTITEMS.ITEMID = ICV00010.ITEMID)
  FOR FETCH ONLY

Access Table Name = USERID.AVT00010  ID = 57
&#186;  #Columns = 2
&#186;  Relation Scan
&#186;  &#186;  Prefetch: Eligible
&#186;  Lock Intents:
&#186;  &#186;  Table: Intent Share
&#186;  &#186;  Row  : Share
&#186;  Sargable Predicate(s)
&#186;  &#186;  #Predicates = 1
&#186;  Create/Insert Into Sorted Temp Table  ID = t1
&#186;  &#186;  #Columns = 1
&#186;  &#186;  #Sort Key Columns = 1
&#186;  &#186;  Sortheap Allocation Parameters
&#186;  &#186;  &#186;  #Rows     = 1569
&#186;  &#186;  &#186;  Row Width = 20
&#186;  &#186;  Piped:
Sorted Temp Table Completion  ID = t1
Access Temp Table  ID = t1
&#186;  #Columns = 1
&#186;  Relation Scan
&#186;  &#186;  Prefetch: Eligible
Nested Loop Join
&#186;  Access Table Name = USERID.SBTITEMS  ID = 30
&#186;  &#186;  #Columns = 2
&#186;  &#186;  Single Record
&#186;  &#186;  Index Scan:  Name = SYSIBM.SQL960620093033090  ID = 1
&#186;  &#186;  &#186;  #Key Columns = 1
&#186;  &#186;  &#186;  Data Prefetch: Eligible
&#186;  &#186;  &#186;  Index Prefetch: Eligible
&#186;  &#186;  Lock Intents
&#186;  &#186;  &#186;  Table: Intent Share
&#186;  &#186;  &#186;  Row  : Share
&#186;  &#186;  Sargable Predicate(s)
&#186;  &#186;  &#186;  #Predicates = 1
End of Section
```

Figure 29. Output from db2expln

This DB2 explain output shows that the table scan was against index class AVT00010, and the table scan was because of the second line of the WHERE clause which requires a translate of ATTRIBUTE0050. This table scan problem is caused by a case insensitive setting in the library server configuration file and the fact that ATTRIBUTE0050 is defined as an alphanumeric field.

8.7.4 Fixing a Table Scan Problem

Once the cause of the table scan is identified, the fix is almost trivial. For the previous table scan problem, follow the instructions in the performance tuning section to modify the case_insensitive setting. Rebinding the DLLs after this should generally solve the problem.

If the problem is caused by a missing database index, then follow the instruction in 8.2.2.3, " Creating Database Indexes." on page 92.

8.7.5 Checking for DB2 Locking

Sometimes the displayed system statistics may show the system is not constrained by CPU, I/O or memory. If that is the case, then it is likely that excessive DB2 locking may be the cause. To check whether locking is the problem, use the following command sequence:

```
db2 connect to <Library Server Name>
db2 update monitor switches using lock on
db2 get snapshots for locks on <Library Server Name>
```

The output from the last command will reveal how many locks are held by each of the library server child processes, table name of lock and other helpful information. Detecting a locking problem is easy. For details on fixing this problem, see the DB2 documentation and manuals.

Chapter 9. Using ADSM

This chapter discusses the use of ADSM as an archive/retrieval system, and how to tune when using it.

9.1 VisualInfo/Digital Library Interfacing with ADSM

This section covers aspects of how the ADSM interface currently works. We include it here to give you a better understanding of the overall system. In addition, we have included performance information.

9.1.1 ADSM Database File Aggregation

ADSM, when linked to its standard client, groups individual files which are being backed up together into a bulk object called an *aggregate file*. It then deals with the file as a whole, and keeps a single entry in its database for the aggregate file. This helps improve performance.

With the way VisualInfo/Digital Library uses the ADSM API, no such grouping of VisualInfo/Digital Library objects is feasible. Thus each VisualInfo/Digital Library object has an individual entry in the ADSM database. ADSM performance is not optimum when dealing with relatively small individual objects of around 100 K average, as would be a typical VisualInfo object; the usual aggregate file is megabytes in size. This would probably work well with a Digital Library-type of object, however.

9.1.2 ADSM Filespaces

A VisualInfo/Digital Library collection corresponds to an index class category of documents (such as claims). Objects in a collection can be migrated to an ADSM volume which defines the ADSM management class and the storage pool to be used. ADSM groups objects from a single client (such as VisualInfo/Digital Library) that are stored with the same management class into a filespace.

9.1.3 APIs Used

VisualInfo/Digital Library uses the backup/archive APIs of ADSM. However, the archive/retrieve APIs are designed for hierarchical storage management and that appears to more closely correspond to what is required for VisualInfo/Digital Library.

9.1.4 Should ADSM-Controlled DASD Be Used

One way of using ADSM is for VisualInfo/Digital Library to migrate to an ADSM-controlled disk and let ADSM do storage pool migration from disk to optical. This migration is space-based, with a potential disadvantage for VisualInfo/Digital Library.

The disadvantage is that, once the ADSM space threshold is exceeded, all objects in all filespaces for the client using the most space are migrated in a group, regardless of the date stored. Since, for a VisualInfo/Digital Library system, all filespaces have the same client node (the object server name), all VisualInfo/Digital Library objects will be migrated together. This means there is an unpredictable period that an object will stay on disk: some objects could be migrated to optical on the first day. If ADSM disk caching is not used, then there would be only the optical copy available, leading to slow retrieval for these recent objects which would typically be active within a workflow.

As an alternative, if ADSM disk storage is to be used with VisualInfo/Digital Library, then disk caching should be considered. This allows files to remain on disk, available for retrieval after optical migration. When no more objects can fit in the disk storage pool, cached objects are deleted on a least recently retrieved basis. Note that ADSM deletes individual files when caching is used, rather than working with complete filespaces, as ADSM migration does.

This seems useful for VisualInfo/Digital Library, but there is performance overhead when storing objects to disk. However, there is a decrease in retrieval time, which should be more helpful for VisualInfo/Digital Library.

9.1.5 Writing Directly to Optical Without Using ADSM

The VisualInfo/Digital Library destager or migrator passes objects to ADSM to be stored on optical. ADSM will buffer the arriving data, ensuring that the optical is kept writing at optimal speed. However, as both the VisualInfo/Digital Library destager or migrator are single-threaded, only a single optical drive can be used for concurrently writing data. Note that you could define some combination of destager, migrator and ADSM DASD to write to optical for different index classes. This is more complex, but could be an option.

The following statement was received from VisualInfo/Digital Library development in response to a request for some help in sizing a specific system.

″On AIX, the VisualInfo/Digital Library object server can migrate roughly 63 100 KB objects per minute to ADSM Optical. 20,000 documents will take roughly 5 hours to migrate. However, customers can set up the object server to migrate to a small ADSM-managed DASD and let ADSM handle the migration to optical. This setup should reduce the total migration time down to a couple of hours.″

9.1.6 Design of ADSM Filespaces for Performance

The performance of the ADSM database will degrade if there are too many entries in a filespace. Thus a production-volume VisualInfo/Digital Library system may need to sub-divide its data so it is split into multiple ADSM filespaces. This can be done at first by allocating different collections and ADSM volumes to different index classes. If, over time, the number of objects in a given category accumulates too much, the following method could be adopted:

In the VisualInfo/Digital Library system administration, mark the ADSM volume full and allocate a new ADSM volume name. Create a new ADSM Management class pointing at the same storage pool. Subsequently stored objects will follow the same hierarchical storage policy, but will be entered in a new ADSM filespace, enhancing the ADSM database performance.

9.1.7 Choosing a Tape Device

IBM supplies a fast tape library, the IBM 3575, which is based on the robotics from the 3995. It is specially designed to give fast response for random read requests, as required for VisualInfo/Digital Library and other HSM applications. It should permit an object retrieval in around 20 seconds, with a mount rate of 360/hour. As such, it should give comparable performance with a 3995, but with a far lower cost/GB.

9.1.8 Operational Monitoring

You need to monitor both the amount of free space in the ADSM database and the ADSM storage pools, and allocate more space before it fills up. For performance reasons, monitor the number of entries in the VisualInfo/Digital Library filespaces. See 9.1.6, "Design of ADSM Filespaces for Performance" on page 118 for how to create fresh file spaces for VisualInfo/Digital Library.

9.1.9 Problem Determination

Using the ADSM V3 standard client, it is possible to view the data stored by the API client with the same node. Thus we can logon to the ADSM client using the VisualInfo/Digital Library object server node name and access the VisualInfo/Digital Library files stored within ADSM.

9.2 ADSM Performance Tuning

In 9.1, "VisualInfo/Digital Library Interfacing with ADSM" on page 117, we describe briefly how ADSM works with the object server. You should understand that section, as well as ADSM product documentation, before using these tips.

Note: If you need any help or assistance in tuning ADSM, ask your IBM representative, who can refer to MKTTOOLS on HONE, where there is an ADSM Performance and Tuning Guide and some ADSM Performance reports.

9.2.1 Destage from Object Server to ADSM DASD

When using ADSM for destaging/migration, it is important to prevent it from becoming a bottleneck at a later point in time. Many installations choose to destage or migrate to ADSM-managed optical. However, destaging or migrating to ADSM-managed optical is significantly slower than to the ADSM-managed DASD. Therefore, a simple solution to overcome the optical performance is defining a small ADSM-managed DASD, configuring the object server to destage or migrate into this area, and allowing ADSM to handle the migration between unlike devices, such as optical and tape. Using ADSM for migration means that the object server migrator will not have to run at all, or less often, leaving more CPU and I/O cycles for other object server activities.

9.2.2 Maxsessions Setting on ADSM

ADSM needs a session for every concurrent retrieve from VisualInfo/Digital Library (the number of VisualInfo/Digital Library Object Server processes), plus two for the destager and VisualInfo/Digital Library migrator. The default on ADSM/NT is 25, which should be adequate. This may be changed according to requirements.

9.2.3 Defining DBBUFFPOOL on ADSM

It is important to define ADSM's buffer pool to be large enough (say 32 Mb). ADSM provides a cache hit rate statistic using the command `Q DB F=D`. This should be at least 98%. If this is lower, then a larger buffer pool needs to be defined.

9.2.4 ADSM Communications Tuning

Often, the ADSM server will be on the VisualInfo/Digital Library Object Server machine, with no other ADSM clients. In this case, ADSM communications should use Shared Memory (AIX), or Named Pipes (NT) to improve performance.

Chapter 10. Tips for the Workstation Platforms

This chapter contains tips specific to the Windows and OS/2 platforms.

10.1 Tips for Windows Platforms

Here are some tips for the Windows operating systems.

10.1.1 Use Windows NT

1. Do not plan to use the VisualInfo Windows 3.1 client.

   You will almost inevitably encounter memory problems, which are common to Windows 3.1. Windows 3.1 is just not a suitable platform for building a robust client-server solution. Microsoft has advised people against using Windows 3.1 for client-server systems.

2. Do not plan to use the VisualInfo Windows 95 client.

   It is not a strategic operating system and it also suffers from some of the same problems as Windows 3.1. We have seen people almost run out of memory resources with Windows 95. If you have a choice, go for Windows NT, and persuade your customer to make a Windows NT decision also. It will save you and the customer a lot of trouble in the long run.

10.1.2 UDB on Windows NT

Windows NT is well known for using lots of hardware resources, particularly memory. 256 MB should be considered a starting point for a production library server, with 512 MB as more realistic for a system with 100 users and above. DB2 can make significant use of large amounts of memory for buffer pool usage, and it is likely to complain as buffer pool sizes are increased in systems with less than 512 MB. It is always easier to buy memory when equipment is first purchased rather than to upgrade at a later date—start big. Generally AIX systems will require more. 1 GB would be a reasonable starting point for an AIX library server.

10.1.3 Setting Up to Create Large Objects

Under Windows NT, the size of PAGEFILE.SYS should be set to a minimum of 120 MB (for a 15 MB object). This helps to prevent virtual memory errors in the operating system.

To change the minimum size of PAGEFILE.SYS:

1. Open the Control Panel and select **System**.
2. Click **Performance**.
3. Click **Virtual Memory**.
4. Increase the Paging File Initial Size to 120 MB. (For NT servers, the paging file size should already be set to a minimum of 300 MB.)

You can monitor your virtual memory use by selecting the Administrative Tools group.

1. Open the Performance Monitor.

2. Click **Edit/Add to Chart**.
3. Under Object, select **Process**.
4. Under Instance, select the program which is running.
5. Under Counter, select virtual bytes.
6. Click **Add**.

If the graph reaches your minimum page file size, you have reached the limit for your system. You should increase the page file size.

10.1.4 Increasing FRNHEAPSIZE

In NT, the FRNHEAPSIZE needs to be increased. The default is set to 512 pages, which is usually not enough for high-end server support (over 100 concurrent users). This parameter is set in the batch file frnnstls.bat in the FRNROOT directory.

10.1.5 Reorganizing Database Tables

In NT, a reorganization of major tables is required when there are substantial inserts/deletes performed against the database tables. The major tables in this include:

- SBTITEMS
- SBTPARTS
- SBTLINKS
- Index classes (AVT00..) TABLES
- SBTEVENTS (POSSIBLE)
- SBTWIPITEMS (POSSIBLE)
- SBTWIPSUSPITEMS (POSSIBLE)

10.1.6 Entering Host Name

During the network table generation, the user is prompted to enter the host name. The user may enter the symbolic name of the server, the fully-qualified host name, or the actual IP address. We recommend using the symbolic name of the server that is known to the network domain name server. If the network does not have a domain name server, add the symbolic name of the server to the hosts file with its associated IP address.

Note: This recommendation is for VisualInfo Version 2.1. For Versions 2.2 and later, the user may use either address format.

10.1.7 Scanning Resolution

Windows clients store the scanned document on the client workstation hard disk until it is saved. Make sure you have enough disk space if you are scanning large documents. The recommended scan resolution is 200 dpi or lower.

10.1.8 MTU Size for Token-Ring

If you are using Windows NT clients on a token-ring, consider increasing the MTU size to 2000 bytes for a 4 Mbps token-ring and 4000 bytes for a 16 Mbps token-ring. If you are using Windows NT clients over Ethernet, use the default MTU size of 1500 bytes.

10.2 Tips for the Client Application

These tips apply to the client application code which is supplied with both the Windows and OS/2 version of VisualInfo.

10.2.1 Fileroom Searching

Use case-sensitive searches against character index fields. Case-insensitive searches force a relational table scan. The performance of relational scans is proportional to the size of the table and degrades significantly with growth in the number of rows.

10.2.2 System-Assigned Workbasket

If the workbasket contains a large number of items, consider making it a system-assigned workbasket.

10.2.3 Workbasket Count Interval

If the workbasket contains a large number of items, keep the item count interval set high or turn it off. When the workbasket count is refreshed, the library server performs several SQL queries to determine the number of items in each workbasket. The approximate number of LSFRs for each workbasket is three plus (number of items in workbasket / 50 + 1). This number of LSFRs is executed for each client workstation in each specified refresh time interval.

Disable the item count when the TOC of workbaskets is displayed.

10.3 Tips for OS/2 Platforms

Note: For more details on this topic, see *Image and Workflow Library: Best Practices for VisualInfo*, SG24-4792.

This is an excellent reference for planning and installing the VisualInfo components. Included are examples of the actual installation, including recommendations and descriptions for the installation parameters.

10.3.1 Allocate a Large Swapper File

Allocate a large swapper file at system initialization. Monitor the size swapper files on machines after all applications are started. Allocate the swapper file of proportional size. This reduces disk fragmentation.

10.3.2 Communications Manager/2

For SNA protocols, configure the network as a connection network.

10.3.3 DB2/2 System Parameters

If many documents are being added or removed from the system, run the database reorganization utility often to help improve response time. If 10 or more documents are being added and removed daily, the reorganization utility should be scheduled to run nightly. Otherwise, running these utilities weekly should be sufficient. Database utilities should not be run when active users are on the system.

Try to keep DB2/2 log files, the library server database, and the OS/2 SWAPPER.DAT on separate physical drives. Do the same for the object server workstation.

Addition of database indexes results in improved search response times. It is better to create multiple indexes than to create a single index with multiple key fields.

10.3.4 Library Server Database

BUFFPAGE is the RAM allocation to store DB2/2 table data. Each unit specified allocates one 4 K page of memory. A number around 250 (1MB of RAM) should be a starting point. Increase this number as sufficient free RAM allows.

For more information on the BUFFPAGE parameter, see 8.2.2.1, "Tuning DB2 Database Buffer Pool" on page 90.

SORTHEAP is the RAM allocation to perform sort operations of DB2/2 table data. Each unit specified allocates one 4 K page of memory. A number around 36 (1441 KB of RAM) should be a starting point. Increase this number as sufficient free RAM allows.

10.3.5 Client Cache

For users on OS/2 clients who will view the same document multiple times, increase the size of the local cache specified in the file FRNCACHE.CTL located in the \FRNROOT\LSTMGR\CTL directory. The default size is 10 MB, and can be increased to 100 MB to enhance performance. This space is used to keep a copy of the object after the first access to the object server.

For the average page size of 50 K bytes, 100 MB allows approximately the last 2000 pages viewed to be stored in the client′s local cache. When retrieving objects, this list is searched first to determine if the object is present. The probability of reuse needs to be considered before enlarging the local cache size. The longer the list, the longer the search takes if the object is not present.

Note: Do not use this for workstations that use a document only once, such as scanning or re-indexing workstations.

The other parameter in the FRNCACHE.CTL file is the Storage Reclamation Termination Percentage. This specifies the amount of free space left after space reclamation. For client workstations that re-use objects from the cache, consider setting this value to 50%.

Changing both of these values may improve performance temporarily and require monitoring to determine the best settings.

For workstations where the probability of reuse is near zero, the local cache function can be disabled by renaming certain file names. The VisualInfo installation readme file named /FRNROOT/README.OS2 contains directions on how to disable the local cache.

Chapter 11. AIX Performance Tips

Most of the tips and techniques that can be used to optimize performance in the AIX environment are discussed in preceding chapters. However, 11.2, “AIX Disk and I/O Subsystem” on page 128 contains some discussion as to how you can optimize the disk I/O on an AIX system. This section has been extracted from *Monitoring and Managing IBM SSA Disk Subsystems*, SG24-5251 and is included because of the possible impact these suggestions could have on VisualInfo/Digital Library. For other redbooks with related information, refer to: `http://www.redbooks.ibm.com`

11.1 AIX-Based Servers

First, we provide some specific techniques from the field for tuning your library and object servers.

11.1.1 AIX Library Server

Keep the DB2/6000 files and the DB2/6000 library server database on separate physical drives.

Reorganize major tables frequently until the system reaches a steady state. Use the Library Server Re-Org Utility.

Set the ROWLIMIT value in the CNTL table to accurately reflect the desired search results.

11.1.2 AIX Object Server

Avoid filling the staging area. When this happens, the object server goes into the quiesce state and refuses all requests. It does not resume until the staging percent stop is reached. If your staging area is large, the quiesce state can last for hours. An ideal environment is one where de-staging never occurs during prime working hours. The staging area should be large enough to hold all objects to be stored during a single day plus objects that are retrieved during the day.

Another alternative to avoid de-staging during the day is to store objects after the work day completes and de-stage and migrate during off hours. This approach prepares your work for the following day. Do not migrate objects during peak periods of activity.

Isolate the staging area on a physical separate drive and SCSI ID. Keep the DB2/6000 files and the DB2/6000 object server database on separate physical drives.

Reorganize the BASE_OBJECTS table frequently until the system reaches a steady state. Use the object server re-org utility.

11.1.2.1 Checking BUFFPAGE

The following command will tell you the value that BUFFPAGE is currently set to for the VI server:

```
db2 get db cfg for <your server db name> |grep BUFFPAGE
```

In addition, if you increase the BUFFPAGE substantially, the performance should be significantly better and you should see a lot less I/O activity on your system.

11.1.2.2 AIX as a Web Server

For information on sizing an AIX system as a Web server, see Appendix A, “RS/6000 Web Server Sizing Guide” on page 203.

11.2 AIX Disk and I/O Subsystem

In this section, we discuss how to improve performance and provide some basic rules. However, data integrity is most important and this should not be compromised in order to improve performance.

I/O performance depends on many different factors. These include the hardware, the configuration of the hardware, the operating system with its settings, the application itself and how it uses the system resources. Any of these values can be a bottleneck for system performance. It is important to remember that removing one bottleneck will almost inevitably lead to another one elsewhere. A balanced system design is the ultimate goal of all performance tuning.

11.2.1 Performance on disk level

There is usually more than one disk drive installed in any given system. If all your data is on one disk, the performance cannot be better than the Media Data Rate in Table 17. For random operations with small block sizes, the seek and latency times are very important.

Table 17 (Page 1 of 2). SSA Disk Parameter

Disk Name	**Starfire 1100**	**Starfire 2200**	**Starfire 4320**	**Scorpion 4500**	**Scorpion 9100**	**Sailfin 9100**	**Marlin 18000**	**Thresher 9100**
Formatted Capacity	1.1	2.2	4.5	4.5	9.1	9.1	18	9.1
Media Data Rate (Banded) in MBps	9.59-12.58	9.6-12.58	9.59-12.58	10.2-15.42	10.2-15.42	11.5-22.4	11.5-22.4	16.16-25.6
Sustained data rate (MBps)	—	—	—	9.2	9.2	—	—	—
Average read (ms)	6.9	7.5	8.0	7.5	8.5	6.5	7.5	6.3
Average write (ms)	—	—	9.5	—	—	—	—	—
Track-to-track read (ms)	—	—	0.5	0.5	0.5	0.7	0.7	0.7
Rotational Speed	7200	7200	7200	7200	7200	7200	7200	10020
Latency (Average)	4.17	4.17	4.17	4.17	4.17	4.17	4.17	2.99
Buffer size (KB)	—	—	512	512	512	1024	1024	1024

Table 17 (Page 2 of 2). SSA Disk Parameter

Disk Name	Starfire 1100	Starfire 2200	Starfire 4320	Scorpion 4500	Scorpion 9100	Sailfin 9100	Marlin 18000	Thresher 9100
Interface Transfer Rate (max)	20	20	20 MB	80 MB	80 MB	160	160	160

Notes:

1. The Starfire disks are older and cannot be ordered.
2. Further information about the disks can be found on the World Wide Web at:

   `http://www.storage.ibm.com/hardsoft/diskdrdl.htm/`

For sequential applications with large block sizes, the media data rate is the most important factor, while for online transaction types of workload which typically use small block sizes with lots of small seeks, the seek and latency times of the disks are the most important factors. However, if you can stripe the data over more than one disk, you will usually obtain better performance than by having all the data on one disk. If you use AIX disk mirroring, you get better read performance than by using non-mirrored disks.

11.2.1.1 Tuning

All disks have a memory buffer on the disk. Normally this cache is used as read cache. Some operating systems (but not AIX) allow you to use this memory as write cache. This way you get better write performance, but you must use this option carefully. The cache is not protected, so in the event of power failure or certain error conditions on the SSA loop, you can lose data.

Disk performance is also affected by what area on the disk (inner, middle, outer) the data resides. The sustained data rate from a disk is greatest on the outside diameter of the disk (the lowest logical block addresses on the disk). Thus data that needs to be read or written at a high data rate is best situated at low LBAs. The access time of a disk is best midway between the outer and inner edges of the disk. Data that is accessed frequently should be placed in this region. There are several tools on the different platforms to see the location of the data or file systems on the disk.

On AIX, you can use `xlvm`. To change the area expectations of a logical volume on AIX, you can use `smit chlvl`. After this it is necessary to reorganize the volumegroup of the logical volume by `smit reorgvg`. The parameter POSITION on physical volume is an expectation that not all data can be in one area.

11.2.1.2 Performance Implications of Disk Mirroring

If mirroring is being used with non-RAID disks and Mirror Write Consistency is on (as it is by default), you may want to locate the copies in the outer region of the disk, since the Mirror Write Consistency information is always written in Cylinder 0. This region corresponds to the lower logical block addresses near 0. From a performance standpoint, mirroring write operations is costly, mirroring with Write-Verify is costlier still (extra disk rotation per write), and mirroring with both Write Verify and Mirror Write Consistency is costliest of all (disk rotation plus a seek to Cylinder 0).

To avoid confusion, we should point out that although an `lslv` command will usually show Mirror Write Consistency to be on for non-mirrored logical volumes, no actual processing is incurred unless the COPIES value is greater than one. On the other hand, Write Verify defaults to off, since it does have meaning (and cost) for non-mirrored logical volumes.

It should be remembered that, for read operations, mirroring can have significant performance advantages. This is because the LVM has a choice of two disks from which it can read the desired data and it can therefore choose whichever disk has the smaller queue or shortest seek time.

These comments are not specific to SSA, as SCSI disks see the same effects.

The block size that you use depends upon your application. Large blocks are more effective than smaller blocks for transferring large amounts of data. But if your application mainly reads and writes small amounts of data, it makes no sense to use large block sizes. Small block sizes are more effective if high I/O rates are required.

queue_depth: The main parameter that can be changed is the queue_depth for the hdisks. However it is not recommended to change the queue depth parameter, as the system default is probably optimum for most environments.

On AIX you can see the value by the following command:

```
lsattr -El hdiskn -a queue_depth
```

To change this value, the filesystems on this disk must be unmounted and the hdisk set to a status of *defined*. If it is necessary to change this value then:

- unmount / *filesystem_name* (unmount the file systems)
- rmdev -l hdisk*n* (set the hdisk from available to defined)
- chdev -l hdisk*n* -a “queue_depth=x” (x is the new queue depth)
- cfgmgr (run configuration manager to make disk available)
- mount / *filesystem_name* (mount the file systems from step 1)

11.2.2 Performance at Adapter Level

In this section we discuss how you can improve performance by configuring your loops and selecting the right types of adapter.

11.2.2.1 Number of Disks in the Subsystem

In any disk subsystem, the larger the number of independent actuators per Mbyte, the smaller is the impact of seek time. This effect is more pronounced as the number of start I/Os per second increases, the transfer length of each operation decreases, and the size of the seek increases. When approaching limiting cases, it is better to use a larger number of small capacity drives (for example 4N x 4.5 Gbyte drives) rather than a few high capacity drives (N x 18 Gbyte). However be sure to split the access. See Table 17 on page 128.

If your I/O mix consists of many long sequential transfers, then 6 to 12 disk per SSA loop is a good choice. In the limit, this number of files can sustain sufficient data throughput to saturate the bandwidth available to the adapter at micro channel (at approximately 35 Mbps). The maximum sustained data rate of about 35 MBps can be achieved with as few as 6 drives, provided the transfers are large sequential blocks of data. The 6 drive load was measured with 196 KB block sizes. This requires using raw I/O to non-JFS logical volumes, or raw I/O to hdisks.

The more typical situation is using the journaled filesystem (JFS). The default block size transferred with JFS is 4 KB.

Figure 30 shows the operations per second on numbers of disks.

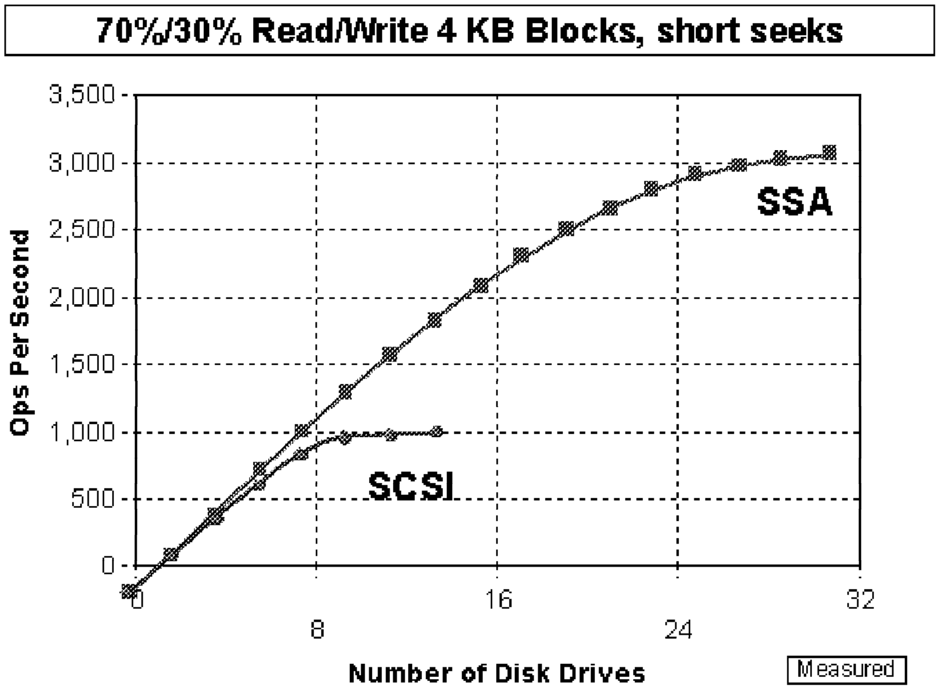


Figure 30. Performance Chart 1

If your I/O mix consists of many short transfers distributed with random seeks across the file, then more files can be comfortably handled by a single adapter. Two loops of 16 or 24 disks would be appropriate (making a total of 32 or 48 disks per adapter).

11.2.2.2 Frame Multiplexing

The SSA adapter card is limited at any one instant in time to transferring data over 4 duplex links (2 per port). So there is a physical limit of 8 data transfer operations at any one instant. However, because the adapter data bandwidth greatly exceeds the bandwidth available from a single drive, this is not really relevant when considering the number of drives that an adapter or host can support.

Transfers from each disk are broken up into 128 byte frames. Each frame is individually addressed. It is possible to interleave frames for different transfers, so that the frames from many different data transfers can be intermixed and sent down the same link one after the other. This is called *frame multiplexing*. The SSA hardware automatically manages this, deciding according to well-defined rules when to insert a new frame, or when to allow existing traffic to flow through. Similarly, the hardware automatically interprets the frame address, and de-multiplexes the data stream, separating it out into its individual logical transfers.

The number of data streams that SSA can support on a port in theory is very large. The SSA adapters under consideration here support 10 data transfers plus two control message frame transfers per port. Hence, 40 data transfers could be streaming between an adapter card and the SSA network at any point in time.

This flexibility in the SSA architecture means it can be quite difficult to state categorically the exact number of disks that an adapter can keep busy, since this is often highly dependent on the particular workload being executed at any particular time.

11.2.3 Basic Configuration Rules

Every link in the loop has a 20 MBps read and a 20 MBps write wire. On each adapter are two links, therefore a adapter can perform 40 MBps reads and 40 MBps writes per loop.

11.2.3.1 Fairness Algorithm

To ensure that no one device dominates the loop, SSA implements a fairness algorithm. Simply stated, the algorithm involves the circulation of tokens (called SAT tokens, for satisfaction). The tokens circulate in both directions around a loop, or from end to end on a string. The principle of the fairness algorithm is that each node will, over time, accumulate a queue of I/O, either data or commands and responses, which it wants to transmit. Typically, if a node is not passing messages from the adjacent node on one side to the adjacent node on the other side, it sends its I/O on the appropriate link, which would be quiet. If the node is busy passing traffic from one side to the other and still wants to transmit I/O, it queues the I/O. In theory, this poor node could be stuck passing traffic for other nodes and be unable to transmit its own.

Enter the SAT token. One token circulates in each direction around the loop (or along the string). Simply stated, if the node receives a SAT token and has a queue of I/O waiting to transmit, it is allowed to transmit the I/O at that time. This transmission is allowed up to a level at which the node is considered “satisfied.” At this point, the node must again wait until either until the link is quiet or it receives another SAT token.

Thus, in theory, you will want to place the disks with the lowest I/O nearest the adapter, and the disks with the highest I/O furthest from the adapter. The increase in latency, for placing disks further from the adapter, is only about 5 microseconds per hop. So even if a disk is 20 hops from the adapter, the increased latency is only 0.1 ms. However, the SSA loop must be heavily loaded for disk placement within the loop to matter.

In a loop with a single initiator, it is more important to balance the I/O load across both halves of the loop than to worry about the position of the disk within the loop. In a loop with multiple initiators, you should place the disks adjacent to the adapter that controls them.

In Figure 31 on page 133, disks 0 to 7 are using port A1 and disks 8 to 15 are using port A2. If the loop is broken, disks will be automatically take the only possible way to the adapter. The performance impact can be 0 if the link between disk 7 and disk 8 is broken, and it can be up to 50% if one of the links to the adapter is broken.

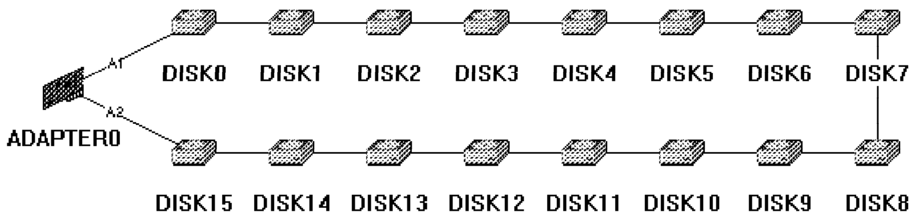


Figure 31. Performance Sample 1

To get better performance, you can change the configuration to Figure 32.

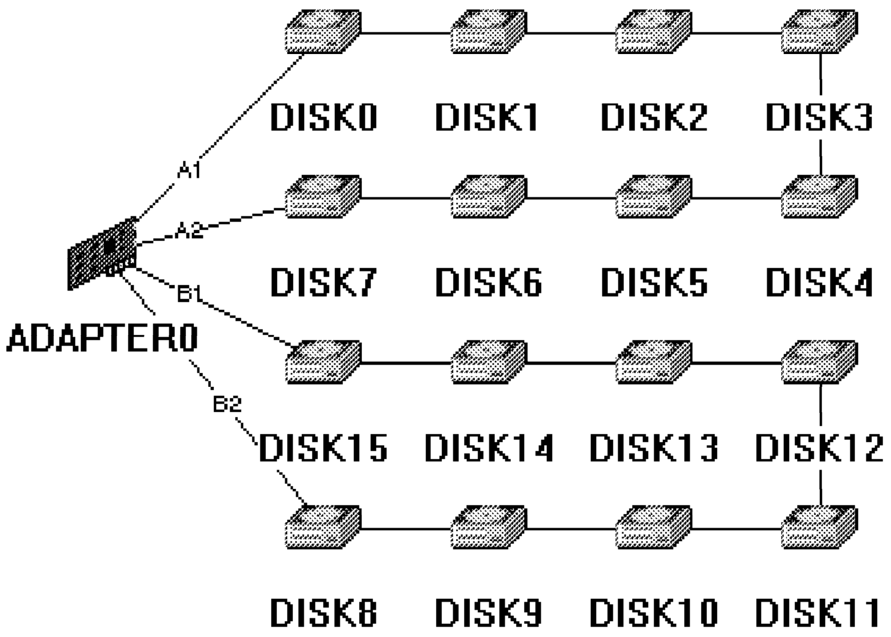


Figure 32. Performance Sample 2

In this configuration, you use four links. Therefore the adapter can read and write 80 MBps from the disks, so you get a theoretical bandwidth from 5 MBps for each disk at the same time. (In Figure 31, it was 2.5 MBps). You have two separate loops. If one loop breaks, you have between 50 and 100% performance on the defect loop and full performance on the other loop.

11.2.3.2 Number of Adapters

The next bottleneck can be the adapter. The throughput from an SSA Microchannel adapter to a system is not higher than 35 MBps and from an SSA PCI adapter, it can be 60 MBps. For higher data rates, you need more adapters.

How many adapters you can use in your system depends on slots in your system and other system specifications. In Table 18 on page 134, you can see how many adapters of each type you can use.

Table 18. Comparison of Adapters

Feature Code	**6214**	**6215**	**6216**	**6217**	**6218**	**6219**	**4012**	**4011**	**4003**	**7190-100**	**7190-200**
Operating System	AIX	AIX	AIX	AIX	AIX	AIX	Win NT	Win NT, OS/2, Netwar, DOS	Solaris 2.4, Solaris 45 2.5.1	Sun, HP, AIX, NT, other	Sun, HP, AIX, NT, other
Adapter Description	Classic	Enhncd	Enhncd	RAID5	RAID5	Enhncd	—	—	Classic	extern box	extern box
Bus	MCA	PCI	MCA	MCA	PCI	MCA	PCI	PCI	SBus	SCSI -FW	SCSI -UW
HW RAID Types	n/a	5	5	5	5	5	0,1,5	0,1,5	n/a	n/a	n/a
Adapter / System	4	4	4	4	4	4	3	3	2	SCSI	SCSI
Loops	2	2	2	2	2	2	2	2	1	1	1
Disks per Loop	48	48	48	48	48	48	22	48	48	48	48
Maximal JBOD Multi-Attach	2	2	8	1	1	2*	2	1	4*	4	16
Maximal RAID Multi-Attach	n/a	1 Raid5	0	1 Raid5	1 Raid5	1 Raid5	2 Raid1	1	n/a	n/a	n/a
Read Cache	n/a	32 MB	n/a	8 MB	8 MB	32 MB	—	8 MB	—	—	—
Fast Write Cache (optional)	n/a	4 MB	n/a	n/a	n/a	4 MB	—	—	n/a	—	—
FW Feature Code	n/a	#6222	n/a	n/a	n/a	#6222	—	—	n/a	—	—
Intermix with	6216	6219	6214	n/a	n/a	6215	—	—	n/a	7190	7190
HACMP Qualified	Yes	Yes	Yes	No	No	Yes	n/a	n/a	n/a	Yes	Yes
Target Mode Support	No	No	No	No	No	Yes	—	—	n/a	—	—
IO Operations/sec Non-Raid	3000	3000	3000	3000	3000	3000	—	3000	3000	1900	3000
IO Operations/sec Raid 5	n/a	3000	n/a	3000	3000	3000	2000 Raid1	—	n/a	n/a	n/a
IO Operations/sec	n/a	1000	n/a	1000	1000	1000	—	1000	n/a	n/a	n/a
MBytes/sec (R/W) non-Raid	35/35	35/35	35/35	35/35	35/35	35/35	—	60/40	35/35	18	35
MBytes/sec (R/W) Raid 5	n/a	29/13	n/a	29/7	29/29	29/13	60/Raid1	35/7	n/a	n/a	n/a
Microcode Level	2401	1801	2402	2904	3700	1801	—	—	—	—	—
Adapter Type	4-D	4-N	4-G	4-I	4-J	4-M	—	—	—	—	—
Diagnose & Maintenance	diag 732smit	diag 735smit	diag 738smit	diag 741smit	diag 744smit	diag 747smit	SSA RSM	SSA RSM	ssau ssacf	—	—
Management Layouts	StorX maymap	StorX maymap	StorX maymap	StorX maymap	StorX maymap	StorX maymap	SSA RSM	SSA RSM	—	—	—

A good solution for both performance and availability is to use two adapters in the same loop on the same system, but not all adapters allow this. If you do this, it is better for performance to put disks between those adapters as shown in Figure 33 on page 135.

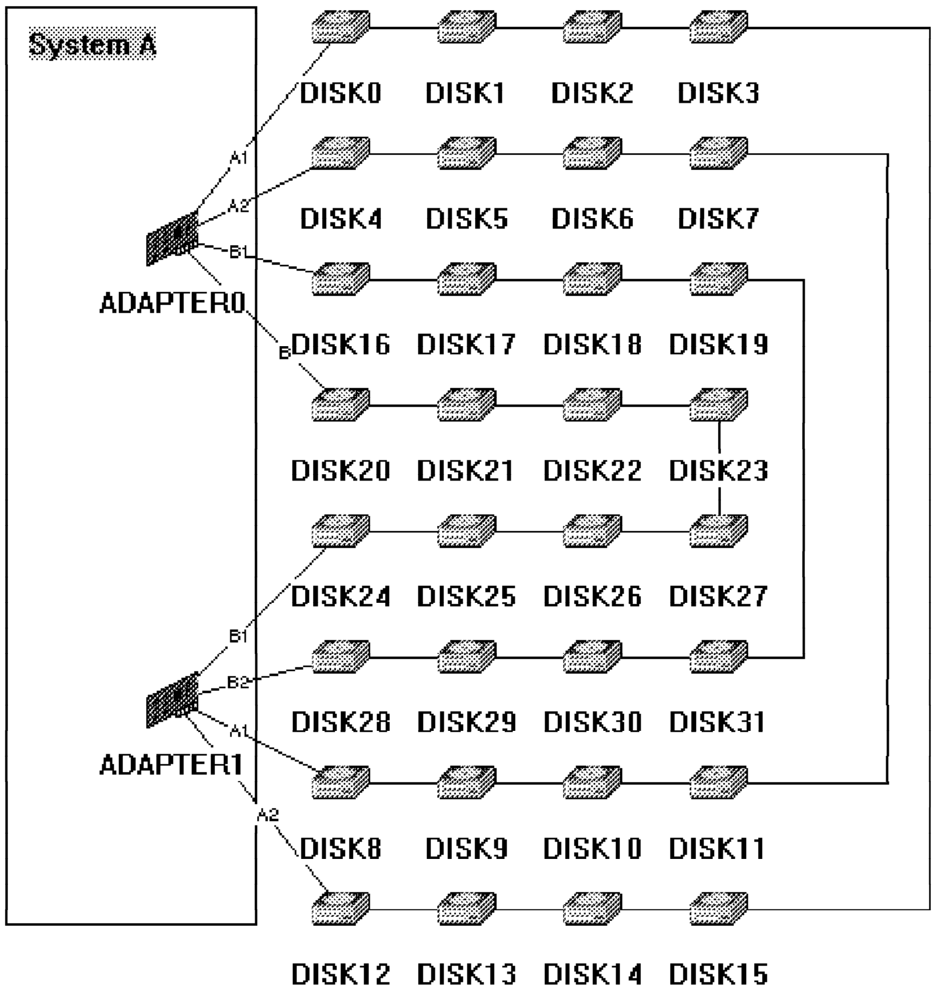


Figure 33. Performance Sample 3

In this configuration, four disks are using one port of the adapter. This is a theoretical bandwidth of 5 MBps for read and 5 MBps for write for each disk at the same time. If one adapter fails, the disks are still available with up to 50% performance impact. In this configuration the throughput from disk to system can be 70 MBps on Microchannel. Figure 34 on page 136 shows a direct connection to the adapter.

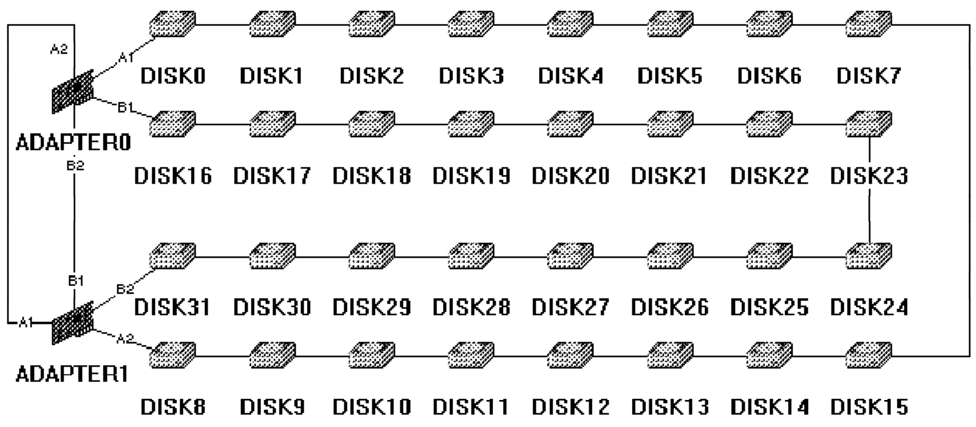


Figure 34. Performance Sample 4

This configuration is not as efficient. The availability and throughput from the adapter to the system is like that of Figure 33 on page 135. However, only one link per loop per adapter is used by the disks, so you get only a theoretical bandwidth of 2.5 MBps for read and 2.5 MBps for write for each disk at the same time.

Using three adapters in one system, as in Figure 35 on page 137, provides a good solution.

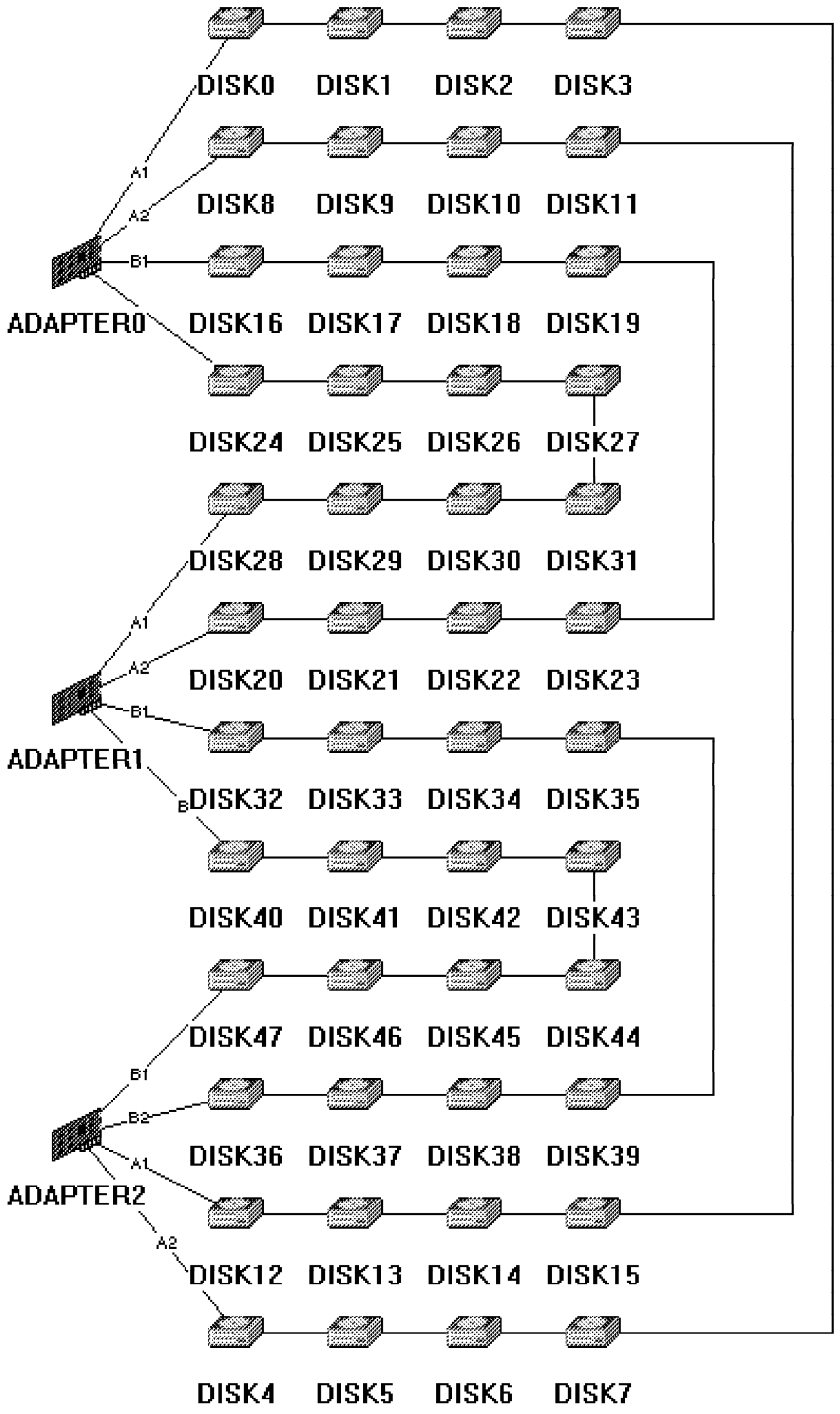


Figure 35. Performance Sample 5

If you use multi-system attached SSA configurations, the same rules apply.

11.2.4 Adapter Types

In this section, we discuss the usage of adapters with RAID hardware. The RAID array is configured on the adapter and not by the operating system. An overview of the differences between the RAID levels can be found in Table 19.

Table 19. Overview of RAID

Raid Level	**Common Name**	**Description**	**Disks Req′d**	**Data Availability**	**Data Transfer Capacity**	**I/O Request Rate**
0	Striping	Data distributed across the disks in the array. No redundant information provided.	N	Lower than single disk.	Very high.	Very high for both read and write.
1	Mirroring	All data replicated on N separate disks. N is most commonly 2.	2N, 3N	Higher than RAID level 3,5 and single disk	Higher than single disk for read, similar to single disk for write	Up to N times that of a single disk for read, less than a single disk for write
5	Raid5	Data sectors are distributed as with disk striping, redundant information is interspersed with user data.	N+1	Much higher than a single disk; comparable to RAID 3	Similar to disk striping for read; lower than single disk for write	Similar to disk striping for read; generally lower than single disk for write

11.2.4.1 Non-RAID

Non-RAID configurations are also called Just a Bunch of Disk (JBOD) and are supported by all types of adapters.

11.2.4.2 RAID 0

Adapters that support RAID 0 are only available for Intel-based systems. Select RAID 0 for applications that would benefit from the increased performance capabilities of this RAID level. However, because data redundancy is not provided, do not use RAID 0 for mission-critical applications that require high availability.

11.2.4.3 RAID 1

Adapters that support RAID 1 are only available for Intel-based systems. Select RAID 1 for applications where data availability is a key concern, and which have high levels of write operations, such as transaction files, and where cost is not a major concern.

11.2.4.4 RAID 5

RAID 5 is supported by several AIX adapters (see Table 18 on page 134) and also by adapters for Intel-based systems. Select RAID 5 for applications that manipulate small amounts of data, such as transaction processing applications. Since all the disk heads move independently (to satisfying multiple requests) it is appropriate for multi-user applications. Also, the selection of RAID 5 is a good compromise between performance, data protection and costs.

11.2.4.5 Number of Components in a RAID 5 Array

In a RAID 5 array, when read requests map onto separate disks within the array, they can be processed in parallel. This striping effect means that arrays with a large number of components can offer better read performance than those with small numbers of components. To offset this, the larger the number of components in a given array, the longer the rebuild time required when a component fails, and the poorer the performance of the array while the component is missing. On the other hand, the larger the number of components in a single array, the lower the cost of providing parity data. Most users will prefer to use relatively small arrays to conserve performance. We recommend 3 to 8 disks.

Other RAID tips include:

- Use spare disks.
- Define several arrays in a loop or on one adapter.
- It is also possible and better for performance to span RAID arrays over the loops on one adapter.

11.2.4.6 Use of Fast Write Cache

Fast write cache can significantly improve the response time for write operations. However care must be taken not to flood the cache with write requests faster than the rate at which the cache can destage its data. Fast write cache can adversely affect the maximum I/O rate, however, since additional processing is required in the adapter card to determine if the data that is being transferred is in the cache or not.

Fast write cache typically provides significant advantages in specialized work loads, for example copying a database onto a new set of disks. If the fast write cache is spread over multiple adapters, this can multiply the benefit.

11.2.4.7 Fiber Extender

The use of fiber extenders reduce the throughput per loop over distance. At a distance of 2.4 km, the performance can be reduced to 12 MBps. Following are recommendations for the usage of RAID adapters.

Non-RAID Operation

- Distribute the disks evenly over the available SSA loops.
- If possible, distribute read and write data evenly throughout the SSA loops.
- For high operations/sec rate applications, use logical volumes made up of small disks. The adapter can sustain a 70/30 mix of read/write operations, with 4 Kbyte transfer lengths, at up to 3000 operations per second. This I/O rate can be produced by relatively few disks, say 32. Additional disks can be attached to the adapter to provide additional connectivity and storage; however, this will not increase the overall rate of the subsystem. A second adapter should be installed if higher I/O rates are required and the load balanced between the two adapters.
- When using Logical Volumes with mirroring, arrange to have the mirror copies on different disks on different loops.

RAID 5 Operation

- For transaction processing applications, where a large number of small unrelated I/O requests are made:
  - Use smaller (1 or 2 GB) disks rather than larger disks (4 GB).
  - If a database log file is being used, place this in a separate non-RAID disk, and mirror it, if required.
  - Increase the queue_depth attribute on each hdisk from the default of 3 to 2N or 3N for a N+1. (This maximizes the chance of reading data from each component of the array in parallel.)
- For data-intensive applications, where a small number of large, possibly related, I/O requests are made, it is tempting to use a large number of components in each array (to maximize the effect of striping).

  However, when the array is operating with one component missing, it must read every other component to reconstruct the missing data. This means that when reconstructing data, performance reduces as the number of components in the array increases. It is better to limit the number of components in the array to a relatively small number to make, say (4+P).

Having selected a particular SSA RAID configuration it is important not to neglect the conventional logical volume manager (LVM) tuning techniques (for example, spreading a Logical Volume across multiple disks, using striped Logical Volumes, multiple JFS logs). As always, it is also important to avoid file system fragmentation.

Chapter 12. OS/390 Scaling and Tuning

In this chapter, we present techniques for performance on the MVS platform.

12.1 OS/390 Scalability

ImagePlus VisualInfo (VI) 2.3 for OS/390 enables multiple Library Server CICS regions offering potential improvements in scalability. Some of the following points may seem obvious, but will help to level-set expectations appropriately:

1. The single Library Server CICS region restriction which existed prior to VI 2.3 is being removed. Within the CICS address space, there is a heavily utilized MVS dispatchable unit known as the Quasi-Reentrant (QR) Task Control Block (TCB). If a VI installation is running on a multi-processor environment, it is possible for the QR-TCB to become saturated (close to 100% utilized), even when the overall system CPU utilization is very low (for example, below 50%).

   This means that the VI Library Server is running at capacity and is unable to fully exploit the processing capabilities of the environment. Therefore, monitoring the CICS Library Server QR-TCB is essential.

   If the QR-TCB is not near capacity, but throughput is constrained, then the recommendation is to tune the environment *before* adding more Library Server regions. Adding more regions to a constrained environment where the QR-TCB is not the bottleneck will only exacerbate existing performance problems.

2. There must be CPU capacity available on the system if the desired end result is more throughput by adding CICS Library Server Regions on a single MVS image. Moving to VI 2.3 will *not* reduce the CPU cost per transaction. It enables the potential for increased VI capacity by removing the single CICS Library Server restriction.

   If your total system CPU utilization with VI 2.1 is near capacity, there will not be much additional throughput from adding more CICS Library Server Regions.

3. On a single MVS image, the most favorable environment for increasing capacity by adding additional CICS Library Server regions is a Central Electronic Complex (CEC) with a large number of CPUs. The minimum number of CPUs is highly dependent upon the VI workload, the non-VI workload, and environmental factors. It is probably safe to assume the CEC must have at least three or more CPUs available to realize scalability benefits from additional Library Server Regions.

4. When adding additional CICS Library Server regions, expect to invest in tuning the environment. Some areas to look at include:

   - I/O Subsystem
   - DB2 log
   - DB2 bufferpool sizes and assignments
   - DB2/VI indexes (recommend type 2)
   - DB2 locking contention
   - DB2 stored procedure parameters

- DB2/VI bind parameters
- CICS/DB2 RCT

VI 2.1 had a built-in constraint with the single Library Server region restriction. VI 2.3 removes this restriction, but with any environment, removing one bottleneck does not guarantee higher capacity. Eliminating one bottleneck often exposes another constraint which may be at least as problematic, so be prepared for performance monitoring and tuning activity.

5. Going from 1 to 2 CICS Library Server regions will not yield 2X capacity. Expect to get no more than 1.6X times the capacity when adding a second Library Server region to a well-tuned system.

12.2 Resources for Tuning

The following sections contain a collection of hints and tips for VisualInfo MVS server performance tuning. However, each environment and workload is different, and therefore there is no guarantee that these techniques will improve the performance in all environments.

The Host-Based Library Server (HBLS) and the Host-Based Object Server (HBOS) are CICS/DB2 applications. As such, the general CICS and DB2 monitoring/tuning methodologies apply. Both CICS and DB2 have a wealth of documentation on this subject. Additionally, the HBOS uses an IBM product, Object Access Method (OAM). Refer to the following publications for further information.

DB2 for OS/390 V5 Administration Guide Volume 2, SC26-8957
CICS TS for OS/390 CICS Performance Guide, SC33-1699
DB2 for OS/390 V5 Performance Topics, SC24-2213
DB2 for OS/390 Application Design Guidelines for High Performance, SG24-2233
CICS/ESA-DB2 Interface Guide, SG24-4536
Object Access Method Planning, Installation, and Storage Administration Guide for Object Support, SC26-4918
MVS Initialization and Tuning Reference, SC28-1752

Every shop has its favorite set of performance monitoring tools. The monitoring tools used here may be considered the “least common denominator.” All of these tools except DB2PM come with the program products (CICS, VisualInfo, MVS/ESA or OS/390) used in a Host Based VisualInfo environment.

- Resource Measurement Facility (RMF)
- DB2 Performance Monitor (DB2PM)
- CICS Statistics
- CICS transactions
  - VI: FRN6 transaction
  - CICS/DB2 attach: DSNC transaction
- VTAM display commands
- MVS catalog address space display commands

12.3 Tuning Suggestions For Both HBLS and HBOS

1. **Suggestion -** Use the CICS VTAM High Performance Option (HPO). This will decrease the CPU cost per transaction.

   How to implement - Specify HPO=YES in the CICS SIT macro. This parameter can only be set via the SIT, it cannot be set using CICS SYSIN overrides. You may also need to increase the RAPOOL parameter.

   How to monitor - Use the VTAM statistics report from CICS statistics. If the "Times at RPL maximum" is non-zero, increase the RAPOOL SIT parameter. Use RMF to measure the CICS region CPU consumption per transaction with HPO=NO vs. HPO=YES.

2. **Suggestion -** Turn off the CICS internal trace. This will decrease the CPU cost per transaction.

   How to implement - In the SIT or the CICS SYSIN overrides, specify INTTR=NO.

   How to monitor - Use RMF to measure the CICS region CPU consumption per transaction with INTTR=YES vs INTTR=NO.

3. **Suggestion -** Use MRO/XM or MRO/XCF as the CICS Multi-region communication between Library Server TORs and AORs, rather than CICS VTAM ISC. You can use MRO/XM for communication between regions within a single MVS image and MRO/XCF for regions that are on different members of a sysplex. If the CICS regions are not within a sysplex, you must use VTAM ISC. Using MRO/XM or MRO/XCF will reduce the CPU cost per transaction compared with CICS VTAM ISC. (Note: for HBLS to HBOS, the only supported Multi-region communication protocol is VTAM ISC.)

   How to implement - On the CICS Connection definition, specify:

   ```
   ACCESSMETHOD(XM)
   PROTOCOL(LU61)
   ```

   How to monitor - Use RMF to measure the CICS region CPU consumption per transaction with VTAM ISC vs. MRO connections.

4. **Suggestion -** For RCT tuning, use protected entry threads and specify sufficient threads. This will reduce CPU cost per transaction and improve response time.

   How to implement - In the RCT, make sure there is a DSNCRCT macro for plan FRNMFRNI and plan CBRIDBS with TYPE=ENTRY. Specify reasonable values for THRDS, THRDA and THRDM. We suggest you use THRDS=THRDA=THRDM. Be sure the value specified for THRDMAX on the TYPE=INIT macro is large enough to support the entry threads (too small a value will result in unnecessary thread termination and creation).

   How to monitor - Use the DSNC DISP STAT transaction to check for values in the W/P (Wait Pool) column. If there is a large number, increase the RCT values accordingly. From the DB2PM accounting report, check that the number of DEALLOCATIONs is smaller than the #OCCURRANCEs. Also, a large number of RESIGNONs is an indicator of thread reuse.

5. **Suggestion -** For the DB2 Stored Procedure Address Space, specify a sufficient number of TCBs for concurrent processing of stored procedures and make the VI stored procedure resident.

How to implement - In the startup JCL for the DB2 stored procedure address space, set NUMTCB appropriately (monitor the stored procedure address space to determine if this value needs to be increased).

In SYSIBM.SYSPROCEDURES, update the row for frnmsp01 so that the column STAYRESIDENT='Y'.

```
  UPDATE SYSIBM.SYSPROCEDURES SET STAYRESIDENT='Y' WHERE
     PROCEDURE = 'FRNMSP01';
```

Or modify the install job:

```
  ?FRN?.SFRNINS1(FRNDBLDX)
  the STAYRESIDENT value is ' ' which defaults to 'N',
  change this to 'Y'
```

How to monitor - Use the DB2 -DISPLAY PROCEDURE command to monitor frnmsp01. If there is excessive queuing, then increase NUMTCB to a larger value. The address space must be stopped and started for the new value to take effect.

Use RMF to monitor the device where the stored procedure library resides (look at the STEPLIB DD card in the DB2 SPAS startup JCL). If there is a high I/O rate to this device, it may be caused by reloading the stored procedure.

6. **Suggestion -** Use type 2 indexes for the library server and for OAM. Use this suggestion in conjunction with suggestion 7 on page 145.

   How to implement - For existing environments, check sysibm.sysindexes by running the following query:

```
SELECT CREATOR, NAME,
  FROM SYSIBM.SYSINDEXES
  WHERE INDEXTYPE <> '2'
  AND   TBNAME LIKE 'FRN%';
```

Change these indexes to type 2:

```
ALTER INDEX xxx.xxx SET INDEXTYPE = '2';
```

If you are installing the Library server or OAM, add the clause “type 2” to the create index DDL.

```
For the Library Server modify jobs:
  ?FRN?.SFRNINS1(FRNDBCRT)
  ?FRN?.SFRNINS1(FRNDBCRX)
  ?FRN?.SFRNINS1(FRNDBCR2)
  ?FRN?.SFRNINS1(FRNDBCR3)

For the Object Server (OAM) modify jobs:
  SYS1.SAMPLIB(CBRISQLX)
  SYS1.SAMPLIB(CBRISQLY)
  SYS1.SAMPLIB(CBRISQLn) -- Where n is for each group
```

7. **Suggestion -** Bind plans and packages using RELEASE(DEALLOCATE) and CURRENTDATA(NO). For the best results, make sure the indexes are type 2 (see suggestion 6). This will allow DB2 to use lock avoidance which will reduce CPU costs, and in some cases it will improve response times. Secondly, this will avoid releasing and requiring resources at each commit (or sync point), which will also reduce CPU costs. The best results will occur when there is a high degree of thread reuse. (Note: using RELEASE(DEALLOCATE) in a thread reuse environment will increase the EDM pool utilization. Monitor and adjust the EDM pool accordingly.)

   How to implement - Add the following options to the bind statements:

```
   RELEASE(DEALLOCATE)
   CURRENTDATA(NO)

For the Library Server modify jobs:
   ?FRN?.SFRNINS1(FRNPKBD1)
   ?FRN?.SFRNINS1(FRNPKBD2)
   ?FRN?.SFRNINS1(FRNPLBND)
   modify the skeleton templates (refer to the FRNBATCH, FRNICJCL
   and FRNSQJCL DD cards in the Library Server CICS startup JCL)

For the Object Server (OAM) modify jobs:
   SYS1.SAMPLIB(CBRABIND)
   SYS1.SAMPLIB(CBRHBIND)
   SYS1.SAMPLIB(CBRIBIND)
```

   How to monitor - Use the DB2PM accounting report. The average “lock request” value should be much lower after converting to type 2 indexes and rebinding with these bind options.

8. **Suggestion -** Use the DB2 5.1 PREFORMAT option for tables that grow rapidly. For high insert environments, this will reduce I/O and reduce lock suspensions, which in turn can improve response times.

   How to implement - This is a DB2 V5.1 utility enhancement. Specify the keyword “PREFORMAT” when using the load or reorg utility. When the utility is run, the primary extent will be preformatted (so it will take longer for the utility to run). You must make sure the primary allocation is large enough to minimize secondary extents.

   How to monitor - Turn on db2 performance trace class 6 and 7 (we recommend using GTF as the destination):

```
s gtf    (this proc will be different on each installation)
-sta trace(perfm) class(6,7) dest(gtf)
```

   Reduce the trace by running DB2PM (you may limit the interval by specifying appropriate FROM and TO timestamps).

```
DB2PM LOCKING
  (TRACE(LEVEL(SUSPENSION),
        FROM(,00:01:00),
        TO(,23:59:00) ))
```

   If there is DB2 extend locks contention, there will be entries with lock resource type X′09′:

```
--- L O C K   R E S O U R C E ---
EVENT    TYPE      NAME                    EVENT SPECIFIC DATA
-------- --------- -------------------------------------------------
LOCK     X'09'     DB  =266                DURATION=MANUAL   STATE=X
SUSPEND            OB  =6                  ORIG.RSN=LATCH CONTENTION
                                                    HASH    =X'0000070A'
```

To determine what object is experiencing the extends, run the following query and reorg or re-load the object with the ″PREFORMAT″ option.

```
SELECT NAME, DBID, OBID
   FROM SYSIBM.SYSINDEXES
   WHERE OBID = 6
   AND   DBID = 266;
SELECT NAME, DBID, OBID
   FROM SYSIBM.SYSTABLES
   WHERE OBID = 6
   AND   DBID = 266;
SELECT NAME, DBID, OBID, PSID
   FROM SYSIBM.SYSTABLESPACE
   WHERE (OBID = 6
   OR    PSID = 6)
   AND   DBID = 266;
```

Use the DB2PM statistics report to monitor EDM pool usage. If the "% PAGES IN USE" is high or the hit ratio is low, increase the pool size.

9. **Suggestion -** Put the DB2 log datasets behind a controller with caching. This will improve DB2 logging performance and can improve response time.

   How to implement - On some of the newer controllers, write I/Os will be reasonably fast. For older 3990 controllers, run the IDCAMS utility to enable caching. (See *DFSMS/MVS: DFSMSdfp Storage Administration Reference*, SC26-4920 for more information.)

```
//CACHEON   JOB  MSGLEVEL=(1,1),REGION=4096K,MSGCLASS=A
//GO        EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=*
//SYSIN     DD   *
  SETCACHE VOLUME(xxxxxx) UNIT(3390) SUBSYSTEM ON
  SETCACHE VOLUME(xxxxxx) UNIT(3390) DEVICE ON
  SETCACHE VOLUME(xxxxxx) UNIT(3390) DASDFASTWRITE ON
  SETCACHE VOLUME(xxxxxx) UNIT(3390) CACHEFASTWRITE ON
  SETCACHE VOLUME(xxxxxx) UNIT(3390) NVS ON
```

   How to monitor - Use RMF and check the Avg I/O response time for the device with the DB2 log. The response times should be less than 10ms. The DB2PM accounting report class 3 "ser.task switch" times should be smaller with these settings.

10. **Suggestion -** Specify IOQ=PRIORITY and set the IOP for CICS regions higher than the DB2 address space. This will put CICS application DB2 synchronous read I/Os at a higher priority than the DB2 asynchronous write I/Os. In general, this will keep application response times more consistent.

    How to implement - In the SYS1.PARMLIB(IEAIPSxx) member that is used in your installation specify IOQ=PRTY. Specify an IOP value for the DB2 PGN and CICS PGN so that CICS has a higher value. For example:

```
PGN=35,(DMN=35,DP=F80,IOP=F74)     /* CICS                 */
PGN=46,(DMN=46,DP=F90,IOP=F73)     /* DB2                  */
```

The general recommendation is that the DB2 dispatching priority should be higher than CICS, but that the IOP for DB2 should be lower than CICS for Online Transaction processing environments. The standard IBM recommended dispatching priority is:

a. VTAM
b. IRLM
c. DB2
d. CICS

How to monitor - Use the DB2PM accounting report. The class 3 “SYNCHRON. I/O” time should be smaller when CICS has a higher IOP. Also, in the highlights section of the accounting report, the “SYNC I/O AVG.” time should be smaller when CICS has a higher IOP.

11. **Suggestion -** Use DB2 Row Level Locking (RLL) for tables that have update contention. (Note: RLL may degrade query performance against tables vs. page level locking.)

    How to implement - Small tables with high concurrent update activity, like the FRNCHECKEDOUT table, are candidates for RLL. Run the following query to determine the locksize for various tablespaces:

    ```
    SELECT NAME, LOCKRULE
       FROM SYSIBM.SYSTABLESPACE
       WHERE DBNAME = ′xxxxxxx′
       ORDER BY NAME;
    ```

    Using the following monitoring technique, determine if there is a high degree of contention for a tablespace with locksize of PAGE or ANY. Alter the locksize to row:

    ```
    ALTER TABLESPACE xxxxxxx.yyyyyyy LOCKSIZE ROW;
    ```

    How to monitor - Use the same monitoring technique as described for suggestion 10 on page 146. Look at the DB2PM lock suspension report to determine if there are many lock suspensions for a particular tablespace. Also, look at the DB2PM accounting report “LOCK/LATCH” class three times to see if the times are large. Alter the lock size for the tablespace experiencing the most lock contention. Check the DB2PM accounting report and the lock suspension report to verify if there has been improvement.

12. **Suggestion -** Spread high activity tablespaces and the DB2 logs onto different devices. This will reduce I/O contention and improve response times.

    How to implement - Each environment is different, so it is necessary to monitor your workload. The folder manager workload used for VI performance laboratory evaluation shows the high activity tablespaces, indexes and datasets are:

    - Link indexes
    - Item indexes
    - Item tablespace

- DB2 catalog
- DB2 logs

These are candidates for spreading out to different devices. Datasets can be moved using the following job:

```
//MOVEDS   JOB   MSGCLASS=A,MSGLEVEL=(1,1),REGION=4096K
//MOVE     EXEC  PGM=ADRDSSU,REGION=5M
//SYSPRINT DD    SYSOUT=*
//SYSIN DD *
    COPY OUTDY(VIO012) DS(INC(    -
    xxx.yyyyy.zzzzz.*.*  )) -
    DELETE CAT
```

How to monitor - Use RMF and look for devices with high % DEV UTIL. Determine if there are DB2 tablespaces or datasets which can be moved.

12.4 Tuning Suggestions for HBOS

1. **Suggestion -** Put the OAM collection entries in an MVS catalog that is not on a shared UCB and be sure it has SHAREOPTIONS(3 3). If the catalog is on a shared UCB and the catalog does not have SHAREOPTIONS(3 3), there will be I/Os to the catalog and the VVDS for each OAM store and retrieve.

   How to implement - Look at the attributes of the catalog with the OAM collection entries and determine the volume where the catalog resides.

   ```
   LISTCAT EN('xxx.xxxx')         // xxx.xxxx is the OAM collection name
   LISTCAT EN('yyy.yyyy') ALL     // yyy.yyyy is the catalog
   ```

   If the SHAREOPTIONS are not (3 3), determine if the catalog is on a shared UCB. If it is, then *do not* change the SHAREOPTIONS! Determine if the catalog is actually shared. If it is *not* shared, then alter the SHAREOPTIONS.

   ```
           ALTER 'yyy.yyyy' SHAREOPTIONS(3 3)
   ```

   How to monitor - Use RMF to check the device activity where the catalog resides.

2. **Suggestion -** Enable the Virtual Lookaside Facility for the catalog with the OAM collection. The virtual lookaside facility will buffer information in its address space and can reduce I/Os to the catalog. Reducing I/Os will improve OAM response times.

   How to implement - Create or modify a SYS1.PARMLIB(COFVLFxx) member so that it has the following entries:

   ```
   CLASS NAME(CSVLLA)
   EMAJ(LLA)

   CLASS NAME(IGGCAS)      /* CATALOG DATA SPACE  */
   EMAJ(yyy.yyyyy)        /* your OAM catalog    */
   MAXVIRT(4096)
   ```

   If your SYS1.PARMLIB(COMMNDxx) does not start VLF, then add:

```
COM=′S VLF,SUB=MSTR,NN=xx′
```

How to monitor - Use RMF to check the device activity where the catalog resides. After setting VLF for the catalog, check the I/O activity. Also, the following command may be entered to check the VLF hit ratio:

```
F CATALOG,REPORT,VLF(SYS1.NATIVE.MASTER.CATALOG.LABA70)
IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
IEC359I CATALOG REPORT OUTPUT 712
**CAS*************************************CATALOG DATA SPACE CACHE*
*    HIT%  RECORDS  SEARCHES   HITS   DELETES  SHARING  INVALID   *
*******************************************************************
*  yyy.yyyyy                                                      *
*    050%  000000AD 000435B6 00021B26 0000000F 00000000 00000000  *
*******************************************************************
*    HIT%  RECORDS  SEARCHES   HITS   DELETES  SHARING  INVALID   *
**CAS*************************************CATALOG DATA SPACE CACHE*
IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED
```

3. **Suggestion -** Put the OAM MVS catalog behind a cache controller. This will improve I/O response times and result in faster OAM response times.

   How to implement - See suggestion 9 on page 146.

   How to monitor - See suggestion 9 on page 146.

4. **Suggestion -** Make the Object Server CICS region steplib datasets LLA-eligible. If you are seeing high I/O rates to the devices where the OAM CICS region steplib datasets reside, this will eliminate steplib I/O. The result is faster OAM response times.

   How to implement - Create or modify a SYS1.PARMLIB(CSVLLAxx) member so that it has the following entries for the OAM CICS region steplib datasets. For example:

```
LIBRARIES(-LNKLST-)
LIBRARIES(CICS410.SDFHAUTH)
LIBRARIES(CEEV1R50.SCEERUN)
LIBRARIES(DSN510.SDSNLOAD)
FREEZE(-LNKLST-)
FREEZE(CICS410.SDFHAUTH)
FREEZE(CEEV1R50.SCEERUN)
FREEZE(DSN510.SDSNLOAD)
```

   If your SYS1.PARMLIB(COMMNDxx) does not start VLF, then add:

```
COM=′S LLA,SUB=MSTR,LLA=xx′
```

   How to monitor - Use RMF to check the device activity where the catalog resides. After setting LLA for the datasets, check the I/O activity.

12.5 MVS System Parameters/Performance

Review settings in the following MVS libraries to ensure the best values are being used based on the environment.

- SYS1.PARMLIB(IEFSSNxx)
- SYS1.PARMLIB(IEAICSxx)
- SYS1.PARMLIB(ERBRMFxx)

Online monitors can cause extra overhead with their data gathering.

Make sure that dispatching priority of the following address spaces are in the order of:

1. VTAM
2. IRLM
3. DB2
4. CICS

Use FRNMHBCS to "seed" the FRNCHECKEDOUT table.

Fine-tune the Session_Timeout parameter to reduce session creation:

- Longer for "heads-down" users.
- Too long a value may hold too many resources.
- Too short may not allow for session rates.

Be careful how many index classes you create because each one generates additional DB2 tables and rows. If an index class is large, you should provide a separate table space for it. When creating an index class, create the JCL into a dataset. Verify it first and submit it manually.

When choosing between TCP/IP or SNA communications, SNA generally provides better performance (especially since CICS is used). If TCP/IP is installed, use these two parameters as guidelines:

```
TCP/IP max sockets, MX=200,
TCP/IP backlog queue, QL=100
```

12.6 CICS

Monitor the utilization of DSA for the library server and object server CICS regions in conjunction with the values of MXT and AMXT.

Use CEDA in the CICS library server region to ensure that the trace flag is OFF for transaction FRNI and for transaction FRNO in the object server region. Eliminate journaling if the data is not being used.

RMFMON (TSO) TRX for CICS regions gives transactions reports.

Set DPMODE = LOW in the RCT.

12.7 DB2/MVS System Parameters

To minimize the deadlock detention time, the parameter DEADLOC in procedure SYS1.PROCLIB(IRLMPROC) should be set to (4,4) instead of (15,4). The lower the number, the sooner DB2 breaks the deadlock. This increases throughput although, the deadlocks still occur.

The DB2 DSNZPARM parameter log load should be a high number (>100 000). This parameter is used by DB2 in determining when to synchronize the database records.

Place the following library server DB2 tables in separate DASD volumes and schedule regular REORG/RUNSTATS/BIND:

- ITEM table/index
- Checkout table/index
- Work-In-Progress table/index
- No-Index Class table/index

Isolate the DB2 log on a single volume. Place the DB2 tables of the library server and OAM on different DASD volumes. Spread out image objects evenly through the use of collection name.

Limit the number of users accessing the same workbasket. For high-activity workbaskets, try to limit the number of users per workbasket to 10.

For DB2 *prior* to V4.1:

- Reduce the DB2 deadlock detection cycle time if deadlocks occur frequently.
- Start with eight dedicated DB2 threads to the library server FRNI transaction. Increase the threads gradually to balance transaction response time with deadlocks.
- Start with eight dedicated DB2 threads to the object server FRNO transaction.

Using DB2 V4.1:

- Implement type 2 index and row-level locking.
- Use DSNC DISPLAY STAT to monitor and adjust thread usage.

12.8 Library Server (HBLS)

Delete unneeded entries in the SBTEVENTS table. This table contains events logged by system reorganization utility, event code (1002), and for history logging such as ″check object into workflow,″ event code 1000. The event codes are listed in \FRNROOT\INCLUDE\FRNPFI2.H for event codes one to 25. To reduce the number of events logged, set FRNFMEVENTS=0 to eliminate the logging of workbasket and workflow-related events.

The default session timeout is three seconds. For an occasional user, set session timeout to 0. For heavy users, it may be beneficial to increase the session timeout.

Create a separate library server for each major application.

Chapter 13. VisualInfo for AS/400 Performance Tuning

This chapter describes how to tune VisualInfo on an OS/400 operating system.

13.1 Workstation Configuration

The VisualInfo client is responsible for the presentation functions of the application, and therefore it is an important component. Although there are minimum requirements for the client, informal testing and customer experience have shown that minimum configurations may not always provide good performance in all environments.

Even just a couple of years ago, hardware costs were enough to make us carefully consider a small processor or memory upgrade. However, as hardware costs continue to drop, and the cost of our people who use these systems continues to climb, the hardware cost differential between an adequate PC and a good PC just does not seem to be worth worrying about.

13.1.1 Processor Speed

The faster the processor of the client, the more quickly it can present the information it receives from the client. In general, the Pentium-class processors with speeds of 166 MHz and above will generally provide better throughput.

If the workstation is performing only VI client functions, the speed of the processor can be slower than if the workstation is concurrently performing other application tasks.

13.1.2 Memory Requirements

The recommended memory configuration is 64 MB. Memory in the client can make a significant impact on performance. While this may seem obvious, the following real-world example drives home this fact.

At one customer site, it took 17 seconds to print a four-page MO:DCA document on a Windows 95 client with 32 MB memory. When the customer upgraded the client to 64 MB memory, the same document printed in 4 seconds.

This improvement is consistent with what is generally expected when you add memory to any computer. If the processor is memory constrained, nothing will run well. Memory is getting less expensive every day, so it may be a wise use of money to add some extra memory.

In the Windows NT operating system, you can easily find out how much memory is being used. Open the **Task Manager** by clicking the right mouse button on the task bar. Select the **Performance** tab, and the charts show your CPU and memory usage. If the amount of memory used is more than the physical memory in the workstation, the processor will be paging into and out of memory, which will slow performance.

If the workstation will be used for other applications in addition to VI/DL, it is important to check memory usage when these applications are running as well, to get an accurate picture of what the user will experience.

Windows NT also provides a performance monitor as an administrative tool. To check that out, select **Start**, **Programs**, **Administrative Tools (common)**, and **Performance Monitor**.

13.1.3 Operating System

The Windows operating system that you use will also have an impact on the amount of memory required. Windows 95 generally needs less memory than Windows NT, so you should take this into account when planning workstation configurations.

13.1.4 Disk Space

The general rules for disk space on the client apply. There should be enough room for memory paging space, as well as work space for the client application to store temporary data. Of course, you must also include any space required by any other applications that run on that workstation

13.2 Server Tuning Tips

Here are some techniques for tuning the server.

13.2.1 Logon

Logging on the VisualInfo for AS/400 is a heavy processing activity. When a user logs on to VisualInfo/400, the server sends a number of lists back to be cached in the client. At logon, a search (SQL query) returns index class information and attributes for all index classes, content classes, privilege strings for all tables, a list of all workbaskets, a list of all workflow processes, and so forth.

This information is intended to save the client from having to search the server tables every time one of these items is needed. However, this can make the logon activity appear to be very slow to the user.

The only way to “tune” this activity is to control the size of the tables to be returned to the client. Therefore, any unnecessary index classes, privilege strings, and processes that are not required could be deleted from the system, and this would have a positive impact on the logon performance. Of course, this is all relative. Deleting one process from a set of 100 would not make a large difference to the logon, but deleting 20 of those 100 processes would.

Another way to improve this performance is to have the user stay logged on. As long as the cost of concurrent licenses is not an issue, this will provide the most consistent benefit.

13.2.2 Search

A key component of VI/400 is the search capability. Both Basic and Advanced Search can be used to select items from the VI/400 library server database, and there are a number of things that affect the performance of the searches.

Every search that is requested runs a DB2/400 dynamic SQL query across the library server database. Therefore, whatever can be done to improve database query will improve the VI/400 searches. Note that a search against all index classes does not include the FLRCLASS index class (this is used for legacy WAF and there are no entries in EKD0312 for this index class). There is also no search on DOCCLASS (all documents in EKD0310).

The VI/400 library server database supports up to 8 user-defined key fields for each index class. To support the search capability, there is a view across the physical data organized by each of the key fields on the index class. This view is an AS/400 logical file. However, these views are on the primary key only, so that a search using more than one key might still have to search a large number of records.

If there were an access path that contained each combination of key fields, the SQL query optimizer would be able to choose the access path that matched the key fields used in the search. However, this would require more than 40,000 views, and the system overhead to support these views would negate any query savings.

The best approach to improving the search performance is to build only those access paths that are most frequently used. If a large number of searches always uses the same three key fields in the same order, having an access path with those key fields would improve those searches. This approach would be useful for any highly used combination of key fields searched.

The problem is how to find out which ones are used. The solution, provided in all RISC-based AS/400 operating systems, is a command called STRDBMON (Start Database Monitor). This command will log all file accesses for jobs that you request, and indicate whether an access path was used, and if not, make recommendations for access paths that would improve access.

This command can be run for one or all jobs, and will report its findings. Based on the number of times it built an access path, you can determine which access paths would be most useful.

13.2.3 Reorganizing Physical Files

Another potential area for performance improvement is in file reorganization. As records are added and deleted from various tables in the database, the space occupied by the deleted records becomes unusable. Reorganizing the tables will reclaim that space and improve file access times.

For installations migrating from ImagePlus WAF, there is a change in the way the database behaves. In WAF, document records were never removed from the EKD0310 file. If a document was deleted, the object was removed from the object server, but the library server index entry remained. In VisualInfo for AS/400, this is different. If you delete an item from VI/400, the object is removed from the object server, and the index entry is removed from the library server. This means that the EKD0310 file now contains deleted records, and it is now a candidate for file reorganization.

13.2.4 Excess Logical Files

In any database, the number of indexes built over a table will impact performance to some degree. This is because when a table is to be updated, all indexes built over that table must be checked by the database manager to see if they will be affected by the change. If an index is affected, it will be locked until the update is performed. This means that the more indexes you have over a table, the longer the locks whenever the table is updated.

Therefore, if you are creating extra indexes over the VI/400 tables, be aware that this incurs an overhead, and you should carefully consider the impact.

VisualInfo for AS/400 allows VI client applications to access all items in the database, including documents and folders created by WAF. This allows a migration for existing customers of WAF to VI/400 without losing any documents or folders that already exist, but this does not allow a WAF application to access any documents or folders created by VI/400.

This is implemented using logical views of the key document and folder tables (EKD0310, EKD0315, EKD0350). There is one view that VI/400 uses, and contains all table rows. There is another view that WAF uses, but this view contains rows that reference only WAF objects.

13.2.5 TCP and APPC Protocol Settings

The VI Client can use either TCP/IP or APPC protocols. There are some tuning items which have been found to improve the transmission time.

When you use APPC, you can define the mode to use for the sessions. Tests that were run in the VI/400 lab indicated that using APPC mode QPCSUPP (defined by Client Access/400) reduces the transmission time approximately 50% over the #INTER mode.

If you are using PCOMM for the APPC connection between the client, there are some parameters that can affect transmission throughput. Tweaking PCOMM parameters cut the time by almost 70%! In lab tests some parameters on the LAN device configuration were altered, changing the ACK delay from 100 to 30 ms, POLL timeout from 3000 to 500, and ACK timeout from 1000 to 500. Under MODE definitions, the Default RU size was changed to 16384.

Figure 36 shows the measured results of these tuning changes, when the SimLibReadObject API was used to read a 500 K document:

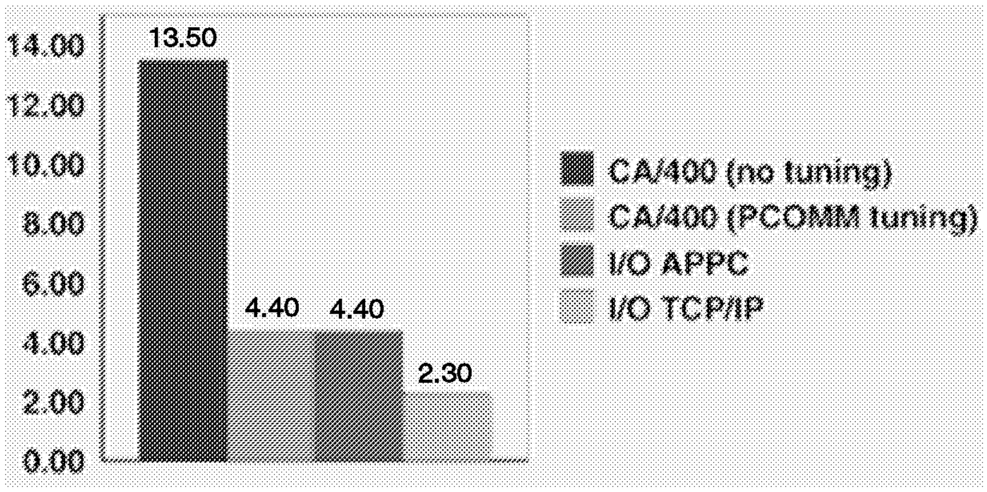


Figure 36. 500K Document Read Throughput

13.2.6 Maximum Transmission Unit

When you create a TCP route, one of the parameters is the Maximum Transmission Unit (MTU) size. This parameter is like a block size, and controls the size of the transmission block. The default is to use the MTU of the TCP interface definition. The interface definition also has a default MTU which defaults to the maximum frame size on the line description (LIND). The default on the Token-Ring line description is 1994 bytes, but the maximum is 16393 bytes. If you accept defaults, as many AS/400 users do, the MTU will be only 2 K. When sending image objects from the object server to the client, a larger transmission unit is better, because it will reduce the number of line turnarounds required. For transmitting images, an MTU set to 16393 should improve performance.

13.2.7 IFS versus QDLS

The VI/400 Object Server can store image objects in a number of file systems. The initial file system was the Document Library System (QDLS). This file system was originally designed for OfficeVision/400 and the storage of its documents, along with associated indexing information (author, description, last changed by). The indexing information associated with this file system adds some overhead to storing documents in QDLS. In the case of VI/400 documents, this indexing information is not used.

A newer file system, added to the AS/400 architecture in Version 3.1 of OS/400, is the Integrated File System (IFS). This PC-like file structure is the base structure upon which all file systems on the AS/400 are built. The developers in Rochester have indicated that this file system is strategic and will continue to be enhanced.

The question has been asked by many existing VI/400 customers about which file system is faster. In tests run by the VI/400 developers, they found a 3% measurable difference between IFS and QDLS. This is not really significant.

13.2.8 General AS/400 Configuration and Tuning

In general, a well-tuned AS/400 will provide good performance, as evidenced by the measurements indicated. Following are three redbooks which provide some guidance toward AS/400 performance tuning:

- *AS/400 Communication Performance Investigation*, SG24-4895
- *AS/400 Server Capacity Planning*, SG24-2159
- *AS/400 Performance Management*, SG24-4735

13.2.9 Separate Listener Jobs

The TCP configuration for VI/400 includes a listener job in the Qserver subsystem that listens for requests from clients. The default TCP port is 31015, and this listener job immediately spawns a new job and passes the client to the spawned job.

While there is no performance information that indicates a bottleneck in the listener function, it is possible to have more than one listener job, each one listening on a different port, and then configure the VI client workstations to use different ports. This would help balance the load of clients, and might improve performance if you have a large number of clients.

The other benefit to different socket numbers would be to isolate problems in a listener job to just that socket, and the clients attached to that socket. If you want redundancy for communications, you could configure more than one server subsystem, and have a different listener job and port assigned to each subsystem. You could then set up the VI client workstations with an entry in the FRNOLINT table for each TCP port, and on alternate clients, select one or the other as the default server.

In effect, each server subsystem would appear to the client as a different server, and if one subsystem or listener job failed, the user could logon to another server subsystem by selecting a different system name from the FRNOLINT table.

13.2.10 Shared File Opens

The AS/400 architecture provides for sharing a file open among all programs in a job. A shared open is approximately twelve times (12X) faster than a “full” open. If the main tables used were opened at logon, there should be a performance benefit, although the lab has not done any testing to confirm this.

The assumption is that the most benefit would be gained by sharing the open of the EKD0310, EKD0312, EKD0315, EKD0350 tables. This effect could be tested by opening these files in the LOGON user exit on the server. This should be tested to ensure that the opens do not conflict with programs which might not be expecting the shared open.

13.2.11 API Resource Utilization

Following is a list of APIs which incur somewhat higher overhead than others. This is useful to help understand what activities are likely to take longer, or use more system resources.

- Logon
- Scan and Import
- Open workbasket
- Add to new folder
- Start process
- End process
- Search

Open workbasket and search are examples of APIs which produce variable workloads, and variable response time. The open workbasket API response time will be directly proportional to the number of items in the workbasket: the more items, the longer it will take to open, because the API is retrieving information about each of the items in the basket.

Search takes a variable amount of time as well. As we discussed earlier, the search function constructs a dynamic SQL query for the search criteria, and depending on the query, there may or may not be an access path available. If not, a table scan is required, and depending on how many entries in the table, the time to complete the query can vary widely. As we discussed, if searches seem to be taking a long time, the Database Monitor can make suggestions for creating an access path for frequently used searches.

13.2.12 Logging

VI/400 APIs can log a variety of information. You can use the API logging to assist with problem determination, as well as assist in performance analysis.

Logging with the VI/400 APIs takes place at either the client or the server, depending on where the API call originates.

On the client, the logging information is written to the file VI400.LOG, which is stored in the root directory for the client APIs. For example, if you installed the client and APIs in the directory FRNROOT, that is where you will find the log. Each time you start the client, the log is overwritten, so you need to copy the file if you want to keep the information in it.

On the AS/400 server, the log is written as a physical file VI400.LOG, and is written to the job′s QTEMP library. This means that it will also disappear when the job ends. If you are running in batch, you need to copy or move the file to another library before the job ends.

There are several logging variables, but the two that seem to have the most applicability for our purposes are the VI400_LOG_PERFORMANCE and VI400_LOG_ALL environment variables.

VI400_LOG_PERFORMANCE is used to log the performance information for each API that is called. When you turn performance logging on, it records start and stop time, as well as bytes transmitted and received. This can help in understanding how long it took a particular API to execute. This logging does not seem to have a performance impact on the running of the API, since it is logging such a small amount of data.

VI400_LOG_ALL is most often used when debugging a problem. When you turn all logging on, all information, including send and receive buffers, are logged. Using this level of logging can have an impact on performance, so it is not recommended for use all of the time. If there is a performance bottleneck that cannot be resolved with logging performance data (above), full logging may be required.

Setting the log levels can be done in a number of ways. On the client, the variable can be set in the registry so that every time you log on, this environment variable is set. You can also set it in a command window, and then call the client or the APIs. This setting is only valid for that command window, and disappears when the window is closed. To set a logging variable in a command window, use the following format:

```
SET VI400_LOG_ALL=Y
```

On the server, the environment variable is set with a command, ADDENVVAR. It can be changed with the CHGENVVAR command as well. To set the performance logging variable, execute this command:

```
ADDENVVAR ENVVAR(VI400_LOG_PERFORMANCE) VALUE(true)
```

To turn off performance logging, execute this command:

```
CHGENVVAR ENVVAR(VI400_LOG_PERFORMANCE) VALUE(*NULL)
```

Because this variable is set using an AS/400 command, you can imbed this in a batch program as well.

13.2.13 Optical Storage Management Performance

The VI/400 object server can store documents of two different types of optical library servers. Optical libraries can either be attached directly to the AS/400 server, and managed by the OS/400 operating system, or attached via LAN, and managed by the optical library controller, which runs OS/2 Warp.

The directly-attached libraries are easier to set up and manage, but the LAN libraries provide the ability to provide optical storage to any user on the LAN, not just to VI/400 users.

The Image Competency Center (ICC) ran a number of tests to compare the performance of the direct-attached versus LAN-attached libraries. The following charts show the results for optical store. The test were run with ImagePlus/400 Version 2.4 on an AS/400 model 320, with processor feature 2050. VI/400 uses the same batch processes to store objects on optical, so the comparative results in this test are still valid for VI/400.

This chart shows the number of 50 KB documents stored per hour using two of the models of optical libraries. The 3995-143 is a direct-attached model, while the 3995-123 is a LAN-attached model. The test was run using both one- and two-store processor job(s) on the AS/400.

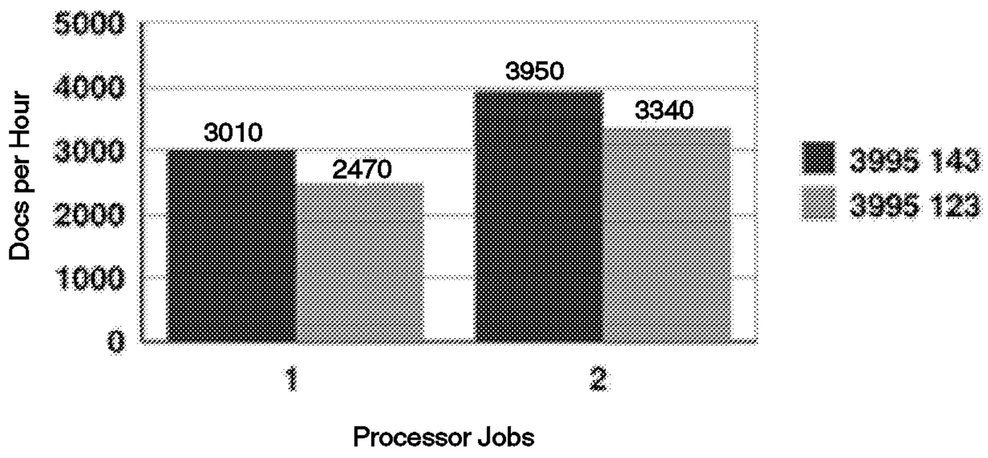


Figure 37. Optical Store Performance. AS/400 320-2050

As you can see from Figure 37, the direct-attached model 143 stored more 50 K documents per hour than the LAN model 123. However, as you can see from Figure 38 on page 161, the AS/400 CPU was utilized more heavily with the direct-attached model.

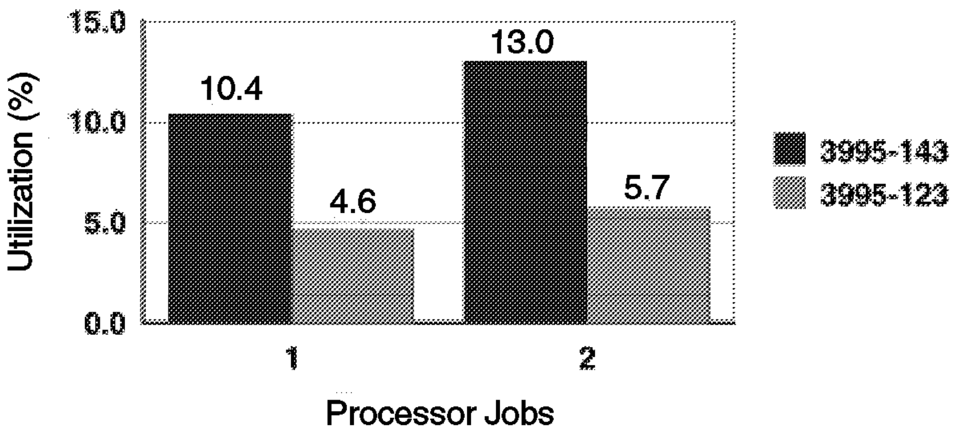


Figure 38. CPU Utilization for Storage Test. AS/400 320-2050

Optical retrieval shows a somewhat different story. As you can see from Figure 39, the LAN-attached library can retrieve 50 K documents faster than the direct-attached library.

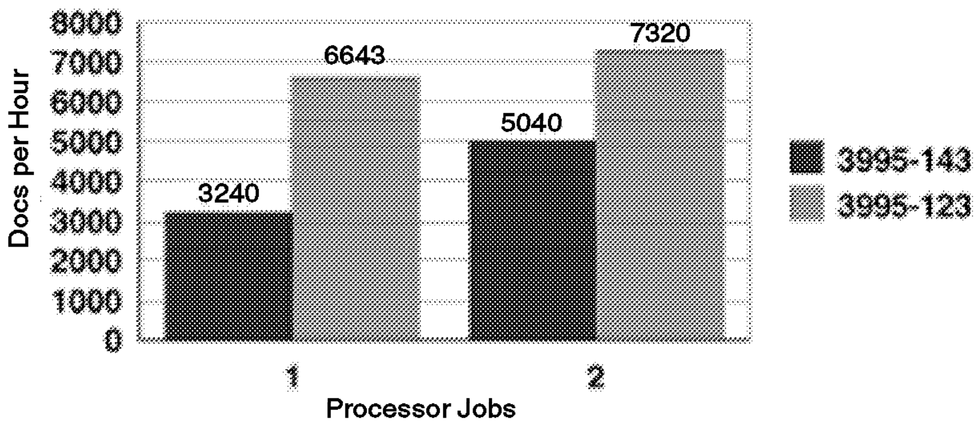


Figure 39. Optical Retrieval Performance -- Docs per Hour. AS/400 320-2050

The CPU utilization is also either the same or less on a LAN-attached library, as shown in Figure 40 on page 162.

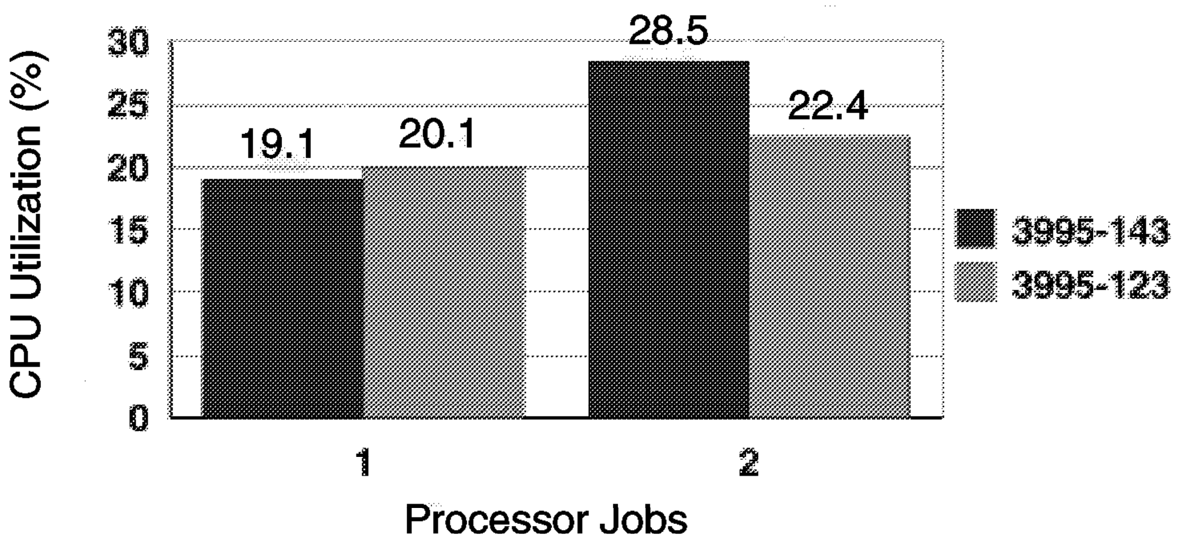


Figure 40. Optical Retrieval Performance -- CPU Utilization. AS/400 320-2050

In general, the biggest throughput improvements in both store and retrieve processes are found when you go from one processor job to two jobs. After the second job, the throughput improvement is not as substantial.

13.2.14 Subsystem Memory

The general recommendation for performance tuning on an AS/400 server is to let the auto tuner built into the operating system do most of the work for you. The way to do this is to set the system value QPFRADJ to automatically tune (option 2 or 3). Thereafter, memory pools that are shared will be tuned on a regular basis by the auto tuner, and gives good performance in most situations. However, auto tune does take some time to take effect. Therefore, if you have a specific cutover at end-of-day from interactive (client) work to batch processing, it is better to move the memory manually to the batch subsystem (or in an scheduled job), and then let the auto tuner adjust accordingly.

13.2.15 AS/400 Expert Cache

The AS/400 has an expert caching function that can be used to keep frequently used database table pages in memory to improve performance. This caching function allows the operating system to determine what should be kept in a memory pool, based on the usage characteristics of the data. To implement expert cache, use the Work with Shared Pools command (WRKSHRPOOL) and change the pools associated with the subsystems in which you are running the server jobs (probably QSERVER). Set the paging option to *CALC, which will let the operating system determine what to keep in memory. Note that you should not select this option if your AS/400 is memory-constrained.

13.2.16 VI/400 Security Implications for Performance

Security configuration can also affect performance, because authority restrictions control what is returned from an API request.

If a user has no read authority, an open will show only those items with explicit read authority. However, a route workbasket request will show all workbaskets whether the user is authorized to read the workbasket or not

Even if you have no authority to search items, the search dialog is enabled on the menu. However, when selecting the index class to search, the user is shown a list of only those index classes to which they are authorized.

This kind of authority-based filtering is important to understand, because a user′s authority can account for perceived variations in performance from one user to another. A user who is authorized to a few items will generally get better performance, because the server will only return items to which the user is authorized, thereby possibly saving a significant amount of data transfer.

Chapter 14. ContentConnect Performance Tuning Tips

This information on ContentConnect tuning was gathered during a period of testing performed in the development laboratory in Research Triangle Park (Raleigh) NC with the ContentConnect development team. While the testing performed was not exhaustive, it did provide some excellent insight into how the product might perform in customer situations. It also provided us with an opportunity to explore the architecture′s tuning capabilities.

14.1 ContentConnect Performance Configuration Options

The ContentConnect architecture is very flexible, allowing many choices in the configuration. As with any architecture, each of these configuration options has a cost associated with it. It is up to the individual customer to determine where his optimum cost/benefit lies.

14.1.1 Local vs. RMI

The first option that has an impact on performance is the choice of connecting the client, either directly to the Server Connector or through Java Remote Method Invocation (RMI).

Local connection means that the there is no RMI server broker managing the connection to the backend server. As you will see from Figure 41 on page 166, local connection is faster.

The cost of choosing this approach is twofold. First, all the backend APIs (VI, OnDemand and so on) must reside on the client workstation. This means that there is additional space required on that client, and that the processing power of the client could become a bottleneck. The second cost is in the maintenance of that workstation. Every time updates are provided to the APIs, each and every client must be updated.

This contrasts with the RMI server broker approach, where the APIs only reside on the server broker machine, not on any of the clients. This allows simpler software maintenance, and smaller clients.

In Figure 41 on page 166, we show the average response time recorded when we ran a comparison between local connection and RMI connection.

14.1.2 Configurations

The personal computers used in these tests for the client and the server broker workstation were IBM PC300GL machines, with 300 MHz Pentium processors, and 160 Mbytes of memory. There was sufficient disk space on each of the machines.

The AS/400 used in these test was a model 510, with the 2144 processor feature, and 512 Mbytes of main memory. DASD usage was approximately 50% of 16 GB available. This machine has a CPW rating of approximately 110.

Note: CPW is a comparative rating that the AS/400 division uses to compare AS/400 processors. For example, a processor with a rating of 200 has approximately twice the capacity for the CPW workload as a processor with a rating of 100.

In each of the following tests, we simulated the following user activities:

1. Log on to the VI/400 server.
2. Execute 10 searches.
    a. Maximum search results = 50.
    b. Maximum items to retrieve = 9.
3. Log off.
4. Execute steps 1 to 3 two more times (for a total of three loops).

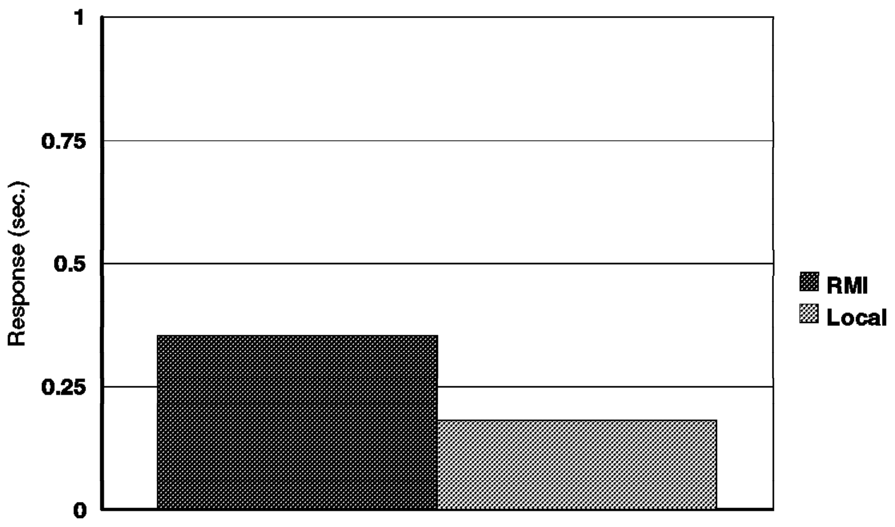


Figure 41. RMI vs. Local Connection

As you can see from Figure 41, local connection is approximately twice as fast as an RMI server broker connection. However, in real terms, this represents only 1/5 of one second slower than the local connection. It seems clear that in a large environment, the extra complexity of an RMI connection would likely be offset by the simplified maintenance of using the RMI broker server architecture.

14.1.3 Applet in Browser

The ContentConnect client can also run as a Java applet within a browser. This approach provides the simplest approach from a maintenance perspective. The applet can reside on the client, and that is all that runs on the client.

The disadvantage of this approach is that performance will not be as good as the full client application. There will be intermediate translations of image formats to a format supported by the browser (in addition to any overhead associated with the browser itself).

We did not perform any tests for performance of the browser approach.

14.1.4 Content Access Server

The Content Access Server is the administration server which contains the Domino database of search templates and server inventory. This server is accessed whenever a server inventory is to be updated, and also the first time a user logs onto the ContentConnect client.

When the user logs on, the server inventory and search templates are cached in the client, and there is no further connection to the Content Access Server. This server is *not* a bottleneck. It has a very low CPU utilization, and could be located on one of the other servers. We did not perform any performance tests on the Content Access Server.

14.2 Federated Search Performance

ContentConnect is a federated search client, allowing you to search multiple backend servers concurrently and then present the search results in one list to the client. One of the questions we had was about the performance of the federated search compared with individual searches.

To perform this test, we used the same clients and servers previously outlined, and added a VisualInfo AIX Library Server to the ContentConnect server inventory.

This test was set up to simulate the following user actions:

1. Log on to the server.
2. Perform 10 searches.
    a. Each search would retrieve a maximum of 50 items.
    b. Each search would not retrieve any items (hit list only).
3. Log off the server.
4. Repeat steps 1 through 3 three times.

The results show the average results for each of the servers (VI/400 and VI AIX), as well as the elapsed time for the combined searches. The search engine will execute the searches in parallel, and return search results to the list as they are returned from the server. Therefore, if one server is taking longer than another, you are not stuck waiting for the slow server to finish before the other search(es) can begin.

You can see from Figure 42 on page 168 the advantage you get from a federated search. The VI/400 searches averaged .29 of a second by themselves. The VI AIX searches averaged .333 of a second by themselves. If the searches were run consecutively, the total time to return the search results would be .623 of a second.

However, because ContentConnect performed the searches concurrently, the total elapsed time to return the search results from both searches was only .387 of a second, only slightly longer than the longest search (VI AIX).

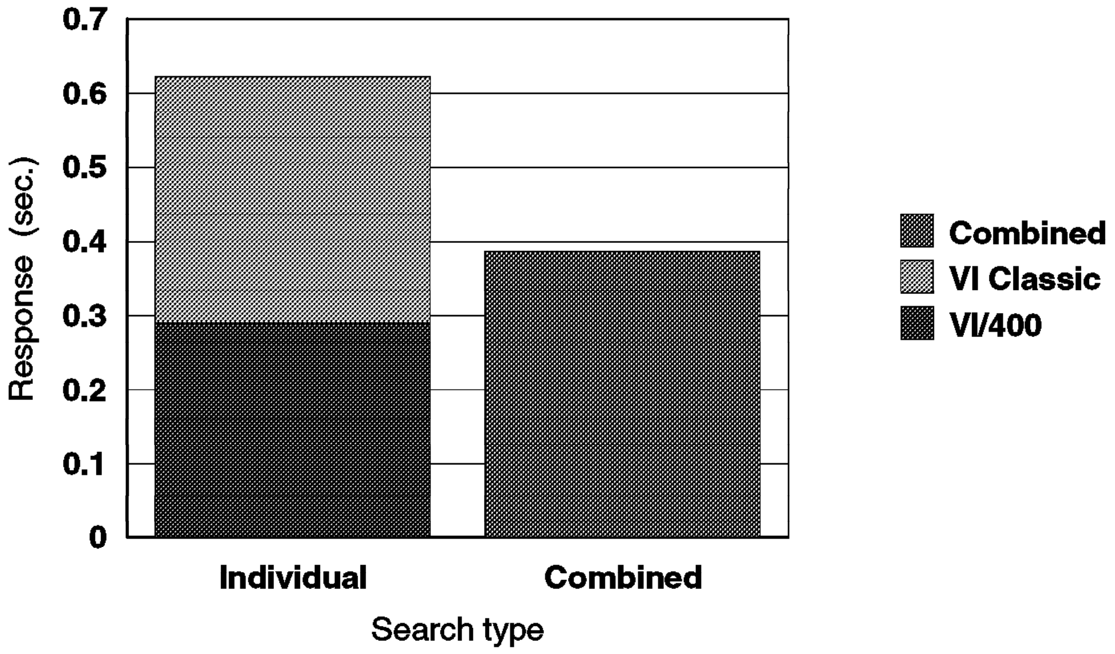


Figure 42. Individual Searches vs. Combined

14.3 Search Performance and Capacity

We also ran a number of tests to determine capacity numbers for client searches. In these tests, we connected to the AS/400 server that we used in the other tests (Model 510), and we used the same PC client and middle-tier RMI servers.

The first series of tests were designed to get a sense of the growing demands placed on the VI/400 server as we added concurrent users through ContentConnect. While the ContentConnect client was used in these tests, it is reasonable to assume that the VI client should give equal or better performance and capacity numbers.

This test was set up to simulate the following user actions:

1. Log on to the server.
2. Perform 3 searches per minute.
   a. Each search would retrieve a maximum of 4 list items.
   b. Each search would retrieve 1 item.
3. Log off the server.
4. Repeat steps 1 through 3 concurrently for the number of users listed in Figure 43 on page 169.

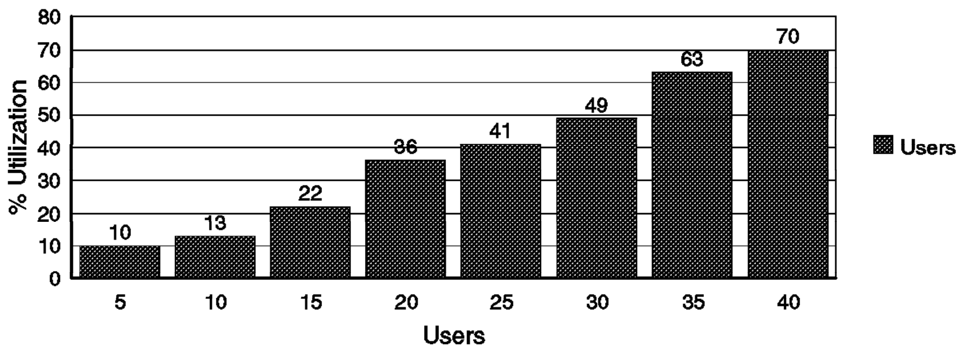


Figure 43. Heavy Users by CPU Utilization

As you can see from Figure 43, there is a steady and predictable increase in the VI/400 server CPU utilization as we add concurrent users to the system.

On this particular server, we reached the recommended maximum utilization at approximately 40 heavy users. This workload would be representative of heads-down data entry users. To get a sense of capacity for a larger number of users, you could use the AS/400 CPW number for this machine (110), and compare to the various ratings for other processors.

We also performed similar tests for other representative workloads. We determined an "interrupted user" workload of the following:

1. Log on to the server.
2. Perform 1 search per minute.
   a. Each search would retrieve a maximum of 4 list items.
   b. Each search would retrieve 1 item.
3. Log off the server.
4. Repeat steps 1 through 3 concurrently for the number of users listed in Figure 44 on page 170.

We also determined a casual user to have the following representative workload:

1. Log on to the server.
2. Perform 1 search every 3 minutes.
   a. Each search would retrieve a maximum of 4 list items.
   b. Each search would retrieve 1 item.
3. Log off the server.
4. Repeat steps 1 through 3 concurrently for the number of users listed in Figure 44 on page 170.

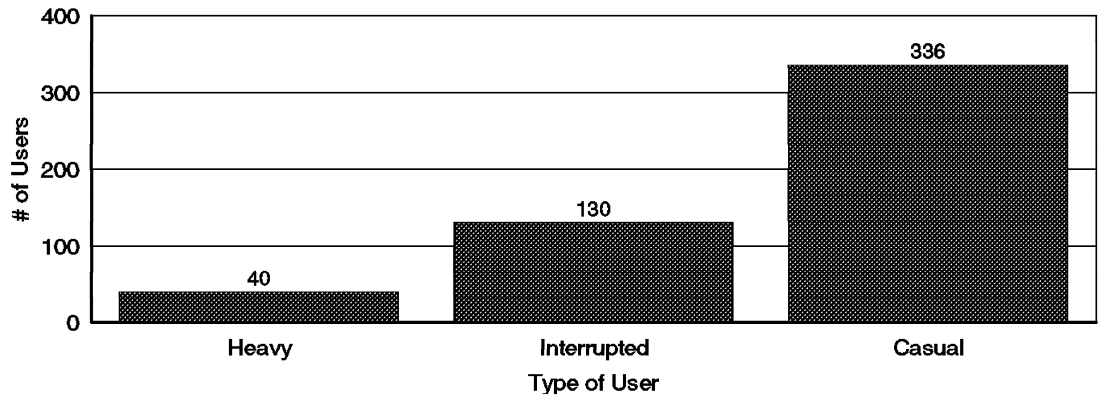


Figure 44. Number of Users at 70% Utilization

As you can see from Figure 43 on page 169 and Figure 44, as you change the workload mix, you can really change the number of users who can get good response time with the server (< 1 second).

Since the AS/400 server products now provide capacities into the thousands of CPWs, you can obtain significant capacity for additional users by increasing the processor.

14.4 Server Pool Performance

The RMI server broker architecture supports multiple server pools processing connections to the backend servers.

There are two kinds of server providers, the broker server and the server pool members. The server pool members serve as a middle-tier server complex. Each server pool member is a separate Java virtual machine (JVM).

The RMI server broker assigns client connection requests to the server pools that are active on a round-robin basis, up to the limit assigned.

As part of our testing, we ran the same test using one server pool member, and then using a second server pool member. The test was set up to simulate the following user actions:

1. Log on to the VI/400 server.
2. Execute 40 searches.
    a. Maximum search results = 50.
    b. Maximum items to retrieve = 0.
3. Log off.
4. Execute steps 1 to 3 two more times (for a total of three loops).

Figure 45 on page 171 shows the results response time measured during those tests.

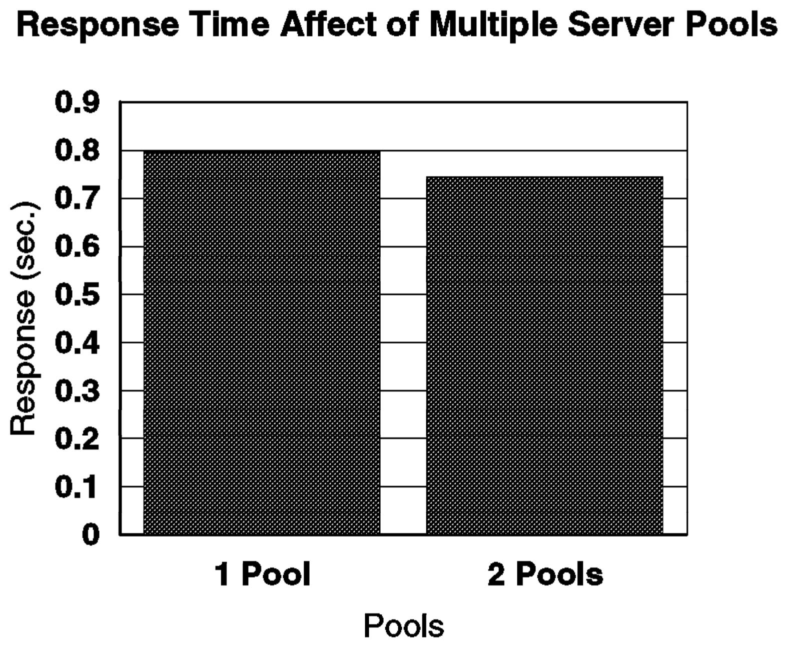


Figure 45. Response Time Effect of Multiple Server Pools

We also tried to find out if there was a limit to the number of sessions (threads) in a server pool member. To test this, we ran four tests, each one simulating the following user actions:

1. Log on to the VI/400 server.
2. Execute 40 searches.
    a. Maximum search results = 50.
    b. Maximum items to retrieve = 0.
3. Log off.
4. Repeat steps 1 to 3 two more times (for a total of three loops).

We ran this test four times, once with 5, 10, 15 and then 20 sessions in a pool, and measured the results as shown in Figure 46 on page 172.

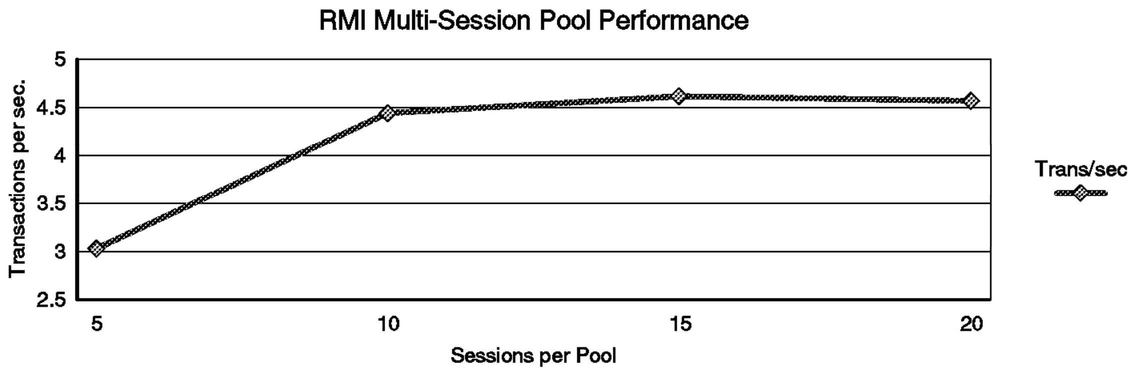


Figure 46. RMI Multi-Session Pool Performance

Based on these results, we can see that a server pool seems to have a limit of approximately 10 active sessions (threads) before the server pool member hits the "knee of the curve." At this point, adding extra sessions to the server pool will provide limited, if not decreasing throughput improvement. At the approximate 10-session threshold, the PC on which it runs uses all its available CPU. Even though the CPU on the PC maxes out, there is still CPU available.

Based on this test, we determined that there is no substantial difference in response time performance as you add a second server pool, as long as you do not run more than 10 to 12 sessions in the server pool. If you exceed the capacity of the pool, performance suffers substantially. Therefore, adding additional pools adds extra capacity for concurrent sessions.

If you add another server pool to the same machine, the CPU will usually drop enough to easily accommodate the additional sessions. Therefore, the bottleneck appears to be in the Java Virtual Machine, not the PC itself. There seems to be some thrashing when it gets busy that affects the CPU itself, although we did not do any tests to isolate the reason for the degradation.

14.4.1 Summary

What we found in these tests is that there are a number of variables that affect the performance, response time and throughput of ContentConnect clients. The architecture provides a variety of ways to configure your system. With some basic tuning rules, you can provide a very high-performing federated search client to a large number of users.

Part 4. Customer Scenarios

- AIX
- MVS/ESA
- OS/2

Chapter 15. Customer Scenario #1

This chapter consists of the first of several scenarios from actual customers. Each chapter describes the application, type of users and their workload, the hardware, software and network configuration. We also give a more detailed description of the most frequently executed transactions in each application, along with folder manager trace information about them. The trace results are compared with the estimates about performance and capacity and a basis for growth is outlined.

The characteristics of customer scenario #1 are shown in Table 20.

Table 20. Customer Scenario #1

Number	Library Server	Object Server	Clients (# - type)	Notes
1 - Accounts Payable	AIX	AIX	200 - Windows	

15.1 An Accounts Payable Scenario

This customer plans to implement an accounts payable application in their accounting department. The purchase orders, invoices, and payment checks are the main documents and are scanned into the system.

The application utilizes an AIX library server, and two AIX object servers. The objects are stored on DASD for 45 days and then are migrated to optical. The VisualInfo windows client is installed on 200 workstations located at a central location.

15.1.1 Overview of the Application

When a purchase order (PO) is issued in the purchasing department, some information may be handwritten on the form and sent off to be prepared for scanning. The vendor is notified of the PO number for reference on their invoice. The index information for the PO is the purchase order number and vendor number. After indexing, the purchase order is filed in the file room until the goods are received. The receiving clerk adds a note to the purchase order.

When the invoice is received in the accounting department, it is scanned into the system and placed in the to-be-indexed workbasket. It is indexed using the invoice number and vendor number and a folder is created. There is an average of four purchase orders per invoice. The invoice folder is added to the AP workflow.

The first workbasket is for reconciliation. A file room search is performed by PO number to find the purchase orders. The purchase orders are added to the folder. If a purchase order is not found, or found but does not have a note from the receiving clerk, the invoice is moved to the exception workbasket. If any comments are attached to a purchase order, such as partial order received or damaged goods, the invoice is moved to the claims workbasket.

After reconciliation, the invoice is moved to the accounting workbasket. The accounting information and the amount to pay are determined and entered into the AP ledger database. The invoice is then moved to the check approval workbasket. When the check is printed, it is verified against the invoice information Then it is scanned in and added to the invoice folder. The printed check is mailed to the vendor. The invoice is removed from the workflow and filed.

The key information on the documents is the vendor number, invoice number, and the purchase order number. During the scan process, the purchase order number and vendor number are barcoded on the purchase orders and the invoice number is hand keyed.

15.1.2 Users

Most of the users are in the accounting department. They use only the workbasket functions for processing the invoices. Some users are in the claims department; they use workbasket and file room searches to retrieve the POs and invoices.

15.1.3 Hardware

The client workstations are:

- IBM P350
- 32 MB of memory
- Fixed Disk: 810 MB

The scanners used are Bell and Howell 6338A.

The AIX library server machine is a model R24. The AIX object server machines are model C20s.

15.1.4 Software

The customer is using VisualInfo V2.2.

The client workstation software configuration is:

- Windows 95
- VisualInfo V2.2
- TCP/IP

Server software configuration is:

- AIX V4.2
- VisualInfo V2.2
- DB2/6000
- ADSM V2.1 for the object server

15.1.5 Network Topology

The workstations are connected through the 10 Mbps Ethernet LAN.

15.1.6 Transactions

The main application is the accounts payable workflow, which allows the user to:

- Get the next invoice.
- Search by purchase order number.
- Add to a folder.
- Route to a workbasket.
- Remove from the workflow.

A folder manager trace was run on a few of the transactions performed during the working day. Each trace starts with a logon to VisualInfo. The following sections describe these main transactions.

15.1.6.1 Scanning

The documents are scanned as simplex (one-sided) documents. The batches are identified as either barcoded or non-barcoded. The scanners are rated at 42 pages per minute. There are a total of three scan workstations in the system. Each user can scan approximately 2400 documents in an hour.

There are 8000 purchase orders, 2000 invoices, and 2000 checks scanned per day.

The invoices are saved to the to-be-indexed workbasket. The purchase orders are filed in the fileroom.

15.1.6.2 Index Processing

During the indexing application for each batch of documents, the index information is verified and any index information that was misread is corrected.

15.1.6.3 File Room Search and Retrieve

When an invoice is received, a search is made for the purchase order and the purchase order is moved into the folder.

15.1.6.4 Workflow Functions

The main workflow is the accounts payable workflow, and it contains the reconciliation, accounting, and check approval workbaskets.

15.1.7 Data Captured

15.1.7.1 Sample Trace Information

The client application trace was produced with the folder trace enabled. Results from the Folder Manager trace are in the file VIC.LOG. It lists all the APIs processed during trace. Looking at the response time for each API call, we can see which calls interacted with the library server and which calls found all of the data in the client′s cache.

It appeared from the trace that during the logon procedure, which was executed in the beginning of each trace, the following APIs call the library server:

- SimLibLogon
- Ip2ListClasses
- SimLibSearch
- Ip2ListContentClasses

In this trace, the total number of library server calls during logon and initiation is five.

The APIs used to add an item to the to-be-indexed workbasket are:

- SimLibCreateItem
- SimLibGetItemType
- SimLibCatalogObject
- Ip2AddWorkBasketItem

In each of the following procedures, some of the API calls were executed several times, depending on the steps taken during that trace. Every time workbaskets, folders, or documents are listed, one or more of the following calls are executed:

- SimLibGetItemType
- SimLibGetTOC
- SimLibOpenItemAttr
- SimLibOpenObject

Storing: Storing the document involved the following APIs interacting with the library server:

- SimLibCreateItem
- SimLibCreateObject
- SimLibOpenObject
- SimLibCloseObject
- Ip2AddWorkBasketItem
- SimLibGetItemSnapshot
- Ip2ListHistory

In this trace, the total number of library server calls for storing a two-page document was six API calls.

Workbasket Functions: In this trace, opening a workbasket and a document in it produced the following library server calls:

- SimLibGetItemType
- SimLibGetTOC
- SimLibGetTOCData
- Ip2ListHistory
- SimLibGetItemAffiliatedTOC
- SimLibOpenObject

The total number of calls to the library server was eight.

Adding the document into a new folder and moving it to a new workbasket involves the following library server API calls:

- SimLibGetTOC
- SimLibGetTOCData
- SimLibOpenItemAttr
- SimLibOpenObject
- SimLilGetItemAffiliatedTOC
- SimLibCreateItem
- SimLibGetItemType
- SimLibGetItemInfo
- SimLibGetItemSnapshot
- SimLibSaveAttr

- SimLibAddFolderItem
- SimLibCloseAttr

The total number of calls to add a new folder was 20.

Searching for one document:

- SimLibSearch
- SimLibGetTOCData
- SimLibOpenItemAttr
- SimLibOpenObject
- SimLibGetItemAffiliatedTOC
- SimLibGetItemSnapshot

The total number of calls was eight.

Workflow Functions: Using the VisualInfo workflow functions to start workflow, add to workbasket, change workflow, and complete workflow uses the following APIs:

- SimLibGetTOC
- SimLibGetTOCData
- SimLibOpenItemAttr
- SimLibOpenObject
- SimLibGetItemAffliatedTOC
- SimLibGetItemSnapshot
- Ip2StartWorkflow
- Ip2GetNextWorkBasketInWorkFlow
- Ip2RouteWipItem
- Ip2GetWorkBasketItemPriority
- Ip2ChangeWorkflow
- Ip2setWorkBasketItemPriority
- Ip2CompleteWorkFlow

The total number of calls varies, depending on the function. Starting the workflow for a document uses three LSFRs. Adding to a workbasket uses six LSFRs. Changing the workflow for a document uses four LSFRs. Completing or removing the workflow for a document uses nine LSFRs.

15.1.7.2 Type and Number of LSFRs

The results from the application trace are listed in Table 21. In this table, the trace results are compared to: estimated number of LSFRs from lab results or as suggested in Table 22 on page 180 or in the Sizer tool.

Table 21 (Page 1 of 2). Trace Results from Client Application/Estimated LSFRs

Function	Trace results	LSFRs
Logon	5	5
Scan	6	6
Index	28	4
Open Workbasket	3	3
Open Document	5	5
Add to Folder	21	14
Add to Workbasket	6	2

Table 21 (Page 2 of 2). Trace Results from Client Application/Estimated LSFRs

Function	Trace results	LSFRs
Start Workflow	3	3
Remove from Workflow	9	

This is an estimate of the average transactions executed when processing each invoice:

- Open workbasket
- Open document
- Add document to folder
- Route to workbasket
- Remove from workflow
- Other (comment, status, and so on; approximately five LSFRs)

Besides the previously mentioned searches, each user performs approximately 20 ad hoc searches.

In Table 22, we show the number of LSFRs in the client application by function.

Table 22. Client Application LSFRs by Function

Function	LSFRs
Create Document	3
Add to Workbasket	2
Open Document from Local Cache	4
Open Document from Object Server or LAN Cache	4
Close Document	1
Index/Save Document	4
Add to New Folder	20
Add to Folder	13
Basic Search	7
Open Folder	8
Start workflow	4
Complete workflow	10
Get next in workbasket	3
Route to next workbasket	3

15.1.7.3 VisualInfo Sizer Tool

Using the Sizer tool gave the following result:

```
Largest hourly LSFRs based on workday      148943
```

This is a little higher than the RS/6000 capacity of 120000 LSFRs per hour.

15.1.7.4 Response Time

Table 23 shows the response times for the functions from tracing this application. The times are in seconds and are arrived at by adding the time it took to execute each API. The actual end-user time is higher.

Table 23. Response Times from Folder Manager Trace

Function	**Response Time (in seconds)**
Logon	4
Scan/Import	3
Open Workbasket	5
Open Document	6
Index	2
Search	4
Start Workflow	2
Add to Folder	4
Add to Workbasket	2
Remove from Workflow	4

15.1.8 Conclusions

It appears that this application transaction mix exceeds the current supported VisualInfo Server configuration. These transaction numbers are for peak periods, and the average workload is 80% of what is modeled here.

Chapter 16. Customer Scenario #2

This chapter consists of the second of several scenarios from actual customers. Each chapter describes the application, type of users and their workload, the hardware, software and network configuration. We also give a more detailed description of the most frequently executed transactions in each application along with folder manager trace information about them. The trace results are compared with the estimates about performance and capacity and a basis for growth is outlined.

The characteristics of this customer scenario are shown in Table 24.

Table 24. Customer Scenario #2

Number	Library Server	Object Server	Clients (# - type)	Notes
2 - Social Security Benefits	MVS/ESA	MVS/ESA	230 - OS/2	Using VIBB/Rexx application

16.1 A Social Security Benefits Scenario

This section describes an insurance company′s VisualInfo application. The company specializes in Social Security benefits and pensions. In this scenario, the library server and the object server are located in an MVS host. The client application is running on OS/2 workstations. This VisualInfo-based customized pilot application is used in 230 of the 5000 workstations in the enterprise. The workstations are connected to the host through an ATM network. VisualInfo workstations are located in two sites in five local offices.

16.1.1 Overview of the Application

When a customer wants to make a claim for a benefit (or pension), they fill out an application form for that benefit. An application form can consist of one page or several pages. Other information such as medical records can be attached to the form. A customer can either send the claim to the company through the mail or bring it to a local office personally. This application is designed to replace the transfer and filing of these paper forms in the insurance company.

When a customer sends a claim for a benefit to the company, it is first scanned in as a document. The key information on each document is the Social Security number. With the Social Security number, the name of the customer is searched from the main database using CICS with HLLAPI. The document is then indexed using the Social Security number and the type of benefit (pension, disability, family allowance, and so on) and stored in a workbasket.

In this application, the VisualInfo workflow is not used. The claims of each customer are always handled in the local office closest to their residence. In each local office, the workbaskets are organized depending on how the local office is organized. In some local offices, all the claims of one customer are handled by the same work group. The customers are ordered in alphabetical order by their last name. In other local offices, a work group handles only special kinds of benefit claims for all the customers.

In both cases, each work group has its own workbasket. When a user starts processing a claim, the document is moved from the work group′s workbasket to a personal workbasket, and a folder is created using the benefit type as an index. The user can also add comments to the document or to the folder. Each customer has as many folders as there are claims for different benefits. While a folder or a document is being processed, it is kept in the workbasket of the user handling it. When a claim for a benefit is handled, the folder including the documents is stamped as being resolved.

16.1.2 Users

All of the users belong to a work group. Depending on the organization of the local office, each group handles either all of the claims of one customer or the same types of claims from all of the customers. Each user is either taking care of direct customer service or doing back-office duty.

Some work groups only go through the mail and scan the documents in. After scanning, the document is stored in the correct workbasket of the work group responsible for handling it. If a customer brings the claim personally to the local office, the user taking care of them scans in the document and stores it directly to the right workbasket. The user can also add comments to the document, create a folder for the document, or store the document in an existing folder.

There are a total of 25 scanner workstations in the system. Each day, the users scan and store approximately 30 pages (six documents at five pages each) in an hour.

Users in other work groups go through the workbasket of the group, choose a document or a folder for handling, and move it to their personal workbaskets. They can also scan documents while serving customers.

Each user handles about two claims in an hour. Besides processing the scanned documents, they have a CICS connection to the host for viewing, updating, and storing other information.

16.1.3 Hardware

The client workstation is a PC330 with:

- Pentium Processor, 133 Mhz or PentiumPro Processor, 200 Mhz
- 32 MB of memory
- Fixed Disk: 1.2 GB or 1.6 GB
- Cache: 10 MB

The scanners used are Fujitsu Scanpartner EOs.

The host server machine is a 9672-RX4.

16.1.4 Software

The customer is using VisualInfo Version 1.2. The customized application program is coded with VisPro REXX using VisualInfo Building Blocks.

The library server and object server are in the host.

The software configuration for the clients is:

- OS/2 V2.11
- VisualInfo V1.2.
- Communications Manager/2 V1.11
- VisPro REXX V3.0
- VIBB V1.95.11

Host software configuration is:

- OS/390 R2
- VisualInfo V1.2.
- DB2 4.1
- CICS 5.1

16.1.5 Network Topology

The workstations are connected to the host through an Asynchronous Transfer Mode (ATM) network. The VisualInfo workstations are located in two different sites. At each site, there are about 100 workstations.

At the remote locations, the workstations are connected through IBM 8282 concentrators to IBM 8260 hubs, which, in turn, are connected to the ATM network provided by the local telecom.

In the data center, the connection is made from the host to the ATM network through 3746 front-end processors, a token-ring LAN, IBM 8281 ATM bridge, and an 8260 hub with a LAN Emulation Server. The network topology is shown in Figure 47 on page 186.

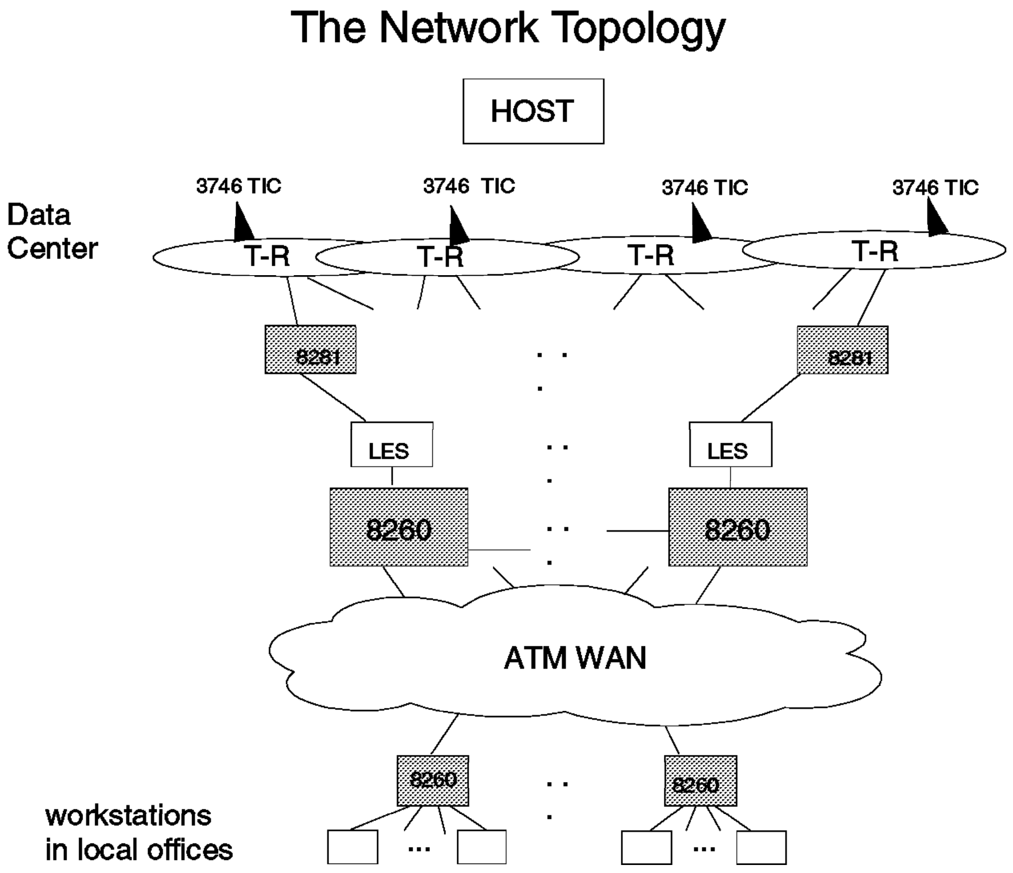


Figure 47. Network Topology

All together there are 2000 workstations in eight sites that are connected to the host through ATM at the time of this writing. The remaining 3000 workstations are connected through SDLC and will soon be using ATM as well.

Of the 2000 workstations connected through ATM, 230 are using the VisualInfo application at the moment. This is about 5% of the total number of workstations.

16.1.6 Transactions

After logging on to VisualInfo and checking the environment, the application asks for the local office to be used. This is the local office where the user is working. After that, the application opens a main window where the following actions can be chosen:

- Scanning
- Workbaskets
- Search by person
- Search by benefit
- Search from archive

We ran a folder manager trace on a few of the most typical transactions or procedures a user does during a working day using this application. Each trace starts with logging on to VisualInfo, performing some checkups, and choosing the local office where the document is to be handled. The following sections describe the main transactions.

16.1.6.1 Scanning

One of the main transactions is scanning a document. When a document is scanned, the name of the customer is searched from a host database using CICS with HLLAPI. The search key is the Social Security number. After that, the document is indexed by the Social Security number and the type of benefit, and stored in a workbasket.

16.1.6.2 Workbasket and Search by Person

Another typical procedure is to open the workbasket and choose a document to be handled. In this trace, we opened first a workbasket and then a six-page document in it. We then viewed the pages of the document one by one, forward and backward. We rotated and zoomed one of them. Then we returned to the main window and made a search by person to get all of the folders for the customer in question. We opened one of these folders, looked at the pages in it, and finally moved it to another workbasket for further handling by another user.

16.1.6.3 Functions in Personal Workbasket

A user can perform certain functions on each document or folder in a workbasket. During this trace, we performed the following functions:

- Comment: We wrote a comment and attached it to a document.
- Status: We viewed the current status of a document including some history information.
- Print: We printed the document with front page.
- Index: We viewed an index listing all of the documents in one folder.
- Archive: We viewed an archive telling where the original paper documents are located.
- Hold/Release: Through a search based on a person (Social Security number), we put a document on hold (for example, waiting for more information from the customer) and then released it.
- Divide: We divided a six-page document into two documents, indexed the new document with new indexing information, and stored it.
- Stamp: We stamped a folder including a claim for benefit as being resolved.

16.1.6.4 Searching

In this application, a search can also be performed on the type of benefit. We did a benefit search listing all of the solved claims in a local office.

Lastly, we did a search listing a customer′s archived documents.

16.1.7 Data Captured

In this section, we explain the results from Folder Manager trace in this customized application. The type and number of LSFRs from the trace is compared to the average estimated number of LSFRs and to the number of LSFRs that the VisualInfo client application is issuing. The trace results are compared to VisualInfo Sizer tool results and to the statistics from the library server and the object server. Also, the response time for each function in the trace is listed.

16.1.7.1 Sample Trace Information

The client application trace was produced with FRNODBIM.CMD. Results from the folder manager trace are in the file FMDUMP.LOG. It lists all of the APIs processed during trace. Looking at the time elapsed between the start and end of each API call, we can see which calls interacted with the library server and which calls found all of the data in the client′s cache. A sample FMDUMP.LOG from tracing the first transaction is in Appendix B, “Output from the Client Trace for OS/2” on page 217.

The logon procedure was executed at the beginning of each trace. From these traces, we can see that the following APIs call the library server:

- SimLibLogon
- Ip2ListContentClasses
- SimLibGetItemType
- SimLibGetTOC
- Ip2GetWorkFlowInfo

Besides SimLibLogon, VisualInfo Building Blocks (VIBB) used in this application issue a call to Ip2ListContentClasses during the VIBB logon procedure VILOGON. The remaining calls are made selecting the local office and establishing the environment for working with its information using VIBB call VIGETTOC. More information about API calls used by VIBB is found in *VisualInfo Building Blocks for REXX and OS/2 Command Line*, GG24-4500.

In these traces, the total number of library server requests during logon and initiation was 13. Ip2GetWorkFlowInfo was called nine times because there were nine workbaskets.

Also other APIs were executed during the logon procedure (see the sample trace in Appendix B, “Output from the Client Trace for OS/2” on page 217). The time elapsed executing these calls was close to zero (less than four-hundredths of a second), since the data for them had been brought from library server to client cache during previous calls.

Some of the API calls were executed several times, depending on the steps taken during that trace. Every time workbaskets, folders, or documents were listed, one or more of the following calls were executed:

- SimLibGetItemType
- SimLibGetTOC
- SimLibGetItemSnapShot
- Ip2ListHistory

The number of SimLibGetTOCData calls depends on the number of documents being read. In this application, 50 documents were read with one library server request.

The following list contains all of the API calls that interacted with the library server during these traces. A complete listing of the APIs and total number of calls to the library server in each trace is in Appendix C, “Trace APIs and LSFR Results from MVS/ESA with OS/2 Client” on page 221.

- SimLibAddFolderItem
- SimLibCloseAttr
- SimLibCloseObject
- SimLibCreateItem

- SimLibCreateObject
- SimLibDeleteItem
- SimLibGetItemAffiliatedTOC
- SimLibGetItemSnapShot
- SimLibGetItemType
- SimLibGetItemXref
- SimLibGetTOC
- SimLibGetTOCData
- SimLibLogon
- SimLibOpenItemAttr
- SimLibOpenObject
- SimLibSearch
- SimLibWriteObject
- Ip2ActivateItem
- Ip2AddWorkBasketItem
- Ip2GetWorkFlowInfo
- Ip2ListContentClasses
- Ip2ListHistory
- Ip2RemoveWorkbasketItem
- Ip2SuspendItem
- Ip2TOCCount

For more information about these APIs, see *ImagePlus VisualInfo Application Programming Reference Volume 2*, GC31-7774

16.1.7.2 Type and Number of LSFRs

The results from the customer application trace are listed in Table 25. In this table, the trace results (LSFRs) are compared to an estimated number of LSFRs as suggested in Table 1 on page 38 or in the VisualInfo Sizer tool or from lab results.

In application, the actual number of calls varies, depending on the steps taken to proceed with each function and when counting the calls is started and stopped. For example, the first OPEN DOCUMENT after logon produces more APIs than the others. The number of library server calls in OPEN WORKBASKET depends on the number of documents in that workbasket. The basic number of calls is three, and it increases by one for every 50 documents. Studying the APIs in the trace, the functions being executed and the total number of API calls during each transaction can be counted.

Table 25. Trace Results from Customer Application / Estimated LSFRs

Function	Trace results	LSFRs (Sizer)
Logon	2	2
Scan	4	6
Store (Index)	3	4
Open Workbasket	6-12	min. 3
Open Document	4	4
Search	3-7	7
Add to New Folder	19	20
Route to Workbasket	10	10
Functions on Document	4-10	

The scanner workstation users scan and store about eight documents in an hour issuing 56 LSFRs. The total amount of scanning and storing from 25 scanner workstation users is 1400 LSFRs in an hour.

The other users are estimated to process about two claims in an hour daily. The actual number is 1.875 if each user is aiming to handle 15 documents daily. The amount of LSFRs processed for each claim varies a great deal. This is a rough estimate about the average transactions made processing each claim:

- Open Workbasket
- Open Document
- Add Document to Folder
- Route to Workbasket
- Other (Comment, Status, and so on; approximately five LSFRs)

Besides the preceding transactions, each user does approximately 20 ad hoc searches.

These users each produce approximately 95 LSFRs in an hour. Total number from 205 users is 19475 LSFRs in an hour.

The total estimated number from all users including all processing is 20875 LSFRs in an hour, giving an average of 91 LSFRs an hour per user.

A low-activity user is defined as generating 375 LSFRs. The number of LSFRs generated by these users is only 24% of that. However, their workload (and also the computer′s workload) includes a great deal of other processing besides image processing.

16.1.7.3 Customized Application versus VisualInfo Client Application

The trace results from the customer application were compared to the number of LSFRs produced by the VisualInfo client application tracing similar transactions. Results from the VisualInfo client application trace are in Table 26. These results are from traces run in the lab using an OS/2 client, OS/2 Library Server and OS/2 Object Server with no other workload.

Table 26. VisualInfo Client Application Trace Results with LSFRs

Function	**Trace results**	**LSFRs**
Logon	2	2
Scan/Import	6	6
Store (Index)	9	12
Open Workbasket	6	7
Open Document	16	Max. 16
Search	4	7
Add to New Folder	14	20
Route to Workbasket	6	11
Functions on document	2-5	3-6

Note: The Item count for workbaskets should be disabled. If it is enabled, two library server API calls are made for every workbasket.

Comparing the results in Table 25 on page 189 with Table 26, it can be seen that the customized client application produces fewer API calls in most of the functions. In addition, the number of library server API calls can be reduced in customized applications by paying attention to the workflow of the application. Studying the Folder Manager trace results of the application can be of great help.

16.1.7.4 VisualInfo Sizer versus Trace Results

Using the VisualInfo Sizer tool produced the following result:

```
Largest hourly LSFRs based on workday       18,606 (OS/2 Client)
```

The difference between the Sizer result and trace results is 12% of the Sizer result. This is quite tolerable because both are estimated values. Even though the Sizer tool labels its results as the largest hourly LSFRs, this figure is based on average numbers of documents being processed in the system. The LSFRs calculated from the trace results are based on rough estimates and assumptions.

16.1.7.5 FRN7 versus Trace Results

We also ran statistics from an actual work day with FRN7. The statistics were gathered every 30 minutes and the counters were initialized every time. When an average number of LSFRs in an hour was calculated from the statistic, the result was 20 239 LSFRs. The difference from VisualInfo Sizer tool results is less than 9% and the difference from trace results is 3%.

Note that this was the average number. The number of LSFRs during the highest peak hour was 30877 LSFRs. When there are 230 users in the system, this is 134 LSFRs per user in an hour. Based on the figure given in Table 27, this means that the total number of users (using MVS/ESA 3090 6-way with 58.6% CPU utilization) can be 1077 even with activity as high as this.

Table 27. Performance Summary

Server Type	Number of High Activity Users	LSFR/hour
OS/2 (PC350)	96	72,000
AIX (R24)	160	120,000
MVS/ESA (3090 6-way)	192	144,324

16.1.7.6 Response Time

Table 28 lists the response times for the functions in this application based on the Folder Manager trace.

Table 28 (Page 1 of 2). Response Times from Folder Manager Trace

Function	Response Time (seconds)
Logon	16
Scan	77
Store (Index)	49
Open Workbasket	8
Open Document	7

Table 28 (Page 2 of 2). Response Times from Folder Manager Trace

Function	**Response Time (seconds)**
Search	8
Add to New Folder	59
Route to Workbasket	48
Functions on document	8

The response times presented in Table 28 on page 191 are end-to-end response times consisting of:

- The total time elapsed tracing one function
- Client processing time

The total elapsed time was calculated from the trace as being the time elapsed between the start of the first API call to the end of last API call. Since this trace was run from the client, this time includes both server time and network time. The statistics gathered with FRN7 show that the total average retrieve time from the server is about five seconds and the total average store time is about two seconds.

The portion of client processing keeps constant depending on the client machine′s capacity. Here we assumed it to be about one second (see Figure 14 on page 50).

16.1.8 Conclusions

When the results from tracing the application transactions and the statistics run during a typical working day are compared to the estimates given by the Sizer tool, the difference seems to be 8-12%. This tool gives a fairly good estimate of the average workload, even for customized applications. Adding 20-30% of capacity to the amounts given by the Sizer tools should provide a safe basis for planning your system.

Using statistics such as FRN6 or FRN7 gives a good view of the workload of a production system. The statistics show what is actually being done and what are the peak hours and numbers. Based on these results, a plan for growth (how many users the current system holds, scalability) can be laid out.

However, there are also other factors to consider. Even though in this system the number of users working with this workload can be increased to 1077 while still providing good throughput and response times, other components might not support this. For example, in this case the CICS library server limits the scalability of VisualInfo, although the CPU might have the processing power.

In a production environment, the VisualInfo client application does not necessarily provide the best possible way of processing a customer′s specific application requirements. A customized application may be needed. The folder manager trace is a good way to find out if the customized application is working efficiently in terms of optimizing calls to the library server and object server.

The measured response times using this application vary from seven seconds (opening a workbasket or folder/document) to one minute (scanning, indexing, and storing a document). This is quite acceptable since the volumes are not that high, and handling each document also involves other processing besides image processing.

The current ATM network has been tested with heavy image workload (sending and retrieving images to/from host to client) up to a load of 8 Mbps. The present load varying from 10 Kbps to a few hundred Kbps does not give any noticeable delay in response times.

The total number of calls varies, depending on the function. Starting the workflow for a document uses three LSFRs. Adding to a workbasket uses six LSFRs. Changing the workflow for a document uses four LSFRs. Completing or removing the workflow for a document uses nine LSFRs.

16.1.8.1 Type and Number of LSFRs

The results from the application trace are listed in Table 29. In this table, the trace results are compared to estimated number of LSFRs from lab results or as suggested in Table 1 on page 38 or in the Sizer tool.

Table 29. Trace Results from Client Application/Estimated LSFRs

Function	Trace results	LSFRs
Logon	5	5
Scan	6	6
Index	28	4
Open Workbasket	3	3
Open Document	5	5
Add to Folder	21	14
Add to Workbasket	6	2
Start Workflow	3	3
Remove from Workflow	9	

This is an estimate of the average transactions made processing each invoice:

- Open workbasket
- Open document
- Add document to folder
- Route to workbasket
- Remove from workflow
- Other (comment, status, and so on; approximately five LSFRs)

Besides the previously mentioned searches, each user does approximately 20 ad hoc searches.

16.1.8.2 VisualInfo Sizer Tool

Using the Sizer tool gave the following result:

```
Largest hourly LSFRs based on workday      148943
```

This is a little higher than the RS/6000 capacity of 120000 LSFRs per hour.

16.1.8.3 Response Time

Table 23 on page 181 shows the response times for the functions from tracing this application. The times are in seconds and are arrived at by adding the time it took to execute each API. The actual end user time is higher.

Table 30. Response Times from Folder Manager Trace

Function	**Response Time (in seconds)**
Logon	4
Scan/Import	3
Open Workbasket	5
Open Document	6
Index	2
Search	4
Start Workflow	2
Add to Folder	4
Add to Workbasket	2
Remove from Workflow	4

16.1.9 Conclusions

It appears that this application transaction mix exceeds the current supported VisualInfo Server configuration. These transaction numbers are for peak periods, and the average workload is 80% of what is modeled here.

Chapter 17. Customer Scenario #3

This chapter consists of the third of several scenarios from actual customers. Each chapter describes the application, type of users and their workload, the hardware, software and network configuration. We also give a more detailed description of the most frequently executed transactions in each application along with folder manager trace information about them. The trace results are compared with the estimates about performance and capacity, and a basis for growth is outlined.

The characteristics of this customer scenario are shown in Table 31.

Table 31. Customer Scenario #3

Number	Library Server	Object Server	Clients (# - type)	Notes
3 - Legal Archive Application	OS/2	OS/2 (combined with Lotus Notes)	22 - OS/2	Lotus Notes Clients

17.1 A Legal Archive Application

This section describes a VisualInfo application for an insurance company′s legal department. The application is done with Lotus Notes. VisualInfo is used in the background through VisualInfo High Level Programming Interface (VHLPI). In this scenario, the library server and the object server are located on the same OS/2 server. The client application for VisualInfo is run on OS/2 workstations, but Lotus Notes clients are running under WINOS2. There are 22 users currently on this system. The workstations are connected to the server with a token-ring LAN network using TCP/IP.

17.1.1 Overview of the Application

This application is designed to file data into two Lotus Notes databases: the lawsuit database and the detective database. The lawsuit database keeps records on all the lawsuits. The application provides the means to search lawsuits with different criteria, and to follow their progress and status. Also, output reports on lawsuits can be printed. The detective database holds information on each case being investigated and the suggested solutions to them.

The application is coded with Lotus Notes. The image type documents are scanned or imported and viewed using VisualInfo in the background. All of the VisualInfo functions used (logon, user privileges, searches, and so on) are handled through Lotus Notes. Each image document is indexed using a Lotus Notes document ID. This links the related images to a Lotus Notes document. Documents are not stored in any folders or workbaskets, and no workflow is used.

17.1.2 Users

There is one scanner in the system. About 30000 to 50000 pages are scanned with it per year. Scanned pages per document can vary from five to 100.

Users of the application search for preliminary cases in the databases to help them resolve new cases. Each user does approximately 10 searches to Lotus Notes in one day. About 20 searches are made to VisualInfo from Lotus Notes in one day by each user.

17.1.3 Hardware

The client workstation is a PC365 with:

- PentiumPro Processor, 180 MHz
- 48 MB of memory
- Fixed Disk: 1.6 GB
- Cache: 10 MB

The scanner used is a Fujitsu M3093.

The server is a PC Server 520 with:

- Pentium Processor, 166 MHz
- 96 MB of memory
- Fixed Disk: 6 GB RAID level 5

17.1.4 Software

The customer is using VisualInfo Version 2.2. The program integrating VisualInfo with Lotus Notes is coded in REXX using VHLPI.

The software configuration for the clients is:

- OS/2 Warp Connect V3.0
- TCP/IP (in OS/2)
- REXX (in OS/2)
- Lotus Notes Windows Client V4.1
- VHLPI from VisualInfo version 2.2

The software configuration of the server is:

- OS/2 Warp Connect V3.0
- TCP/IP (in OS/2)
- REXX (in OS/2)
- DB2/2 V2.11
- Lotus Notes V4.1
- VHLPI from VisualInfo version 2.2

The library server and object server are on the same workstation.

17.1.5 Network Topology

There are 22 workstations connected to one server machine with a 16 Mbps token-ring LAN using TCP/IP. There are no connections outside of this LAN network.

17.1.6 Transactions

A folder manager trace was taken with the most typical transactions in this application. Each transaction is described in the following sections.

17.1.6.1 Scanning

First, in the application, a document is created in Lotus Notes. Some data and key fields are filled in. If there are images belonging to this document, the scanning of the images (for example, court orders) is started. The scanned images are stored and indexed with a Lotus Notes document ID. For backup purposes, they are also indexed by the document type and date. Each document gets a lifetime by Lotus Notes.

17.1.6.2 Importing

Importing an image to a Lotus Notes document is done the same way the scanning is done with the exception that the image is created from an existing file (for example, a text file from a word processor).

17.1.6.3 Viewing

Documents are searched from Lotus Notes. If there are scanned or imported images stored in VisualInfo with that document, they are searched also. The search is done by document ID. Notes and comments can be added to these documents.

Periodically a batch program is run, looking for the images whose lifetime has been exceeded and deleting them from VisualInfo.

17.1.7 Data Captured

This section includes a brief description of the trace results, the type and number of LSFRs, response times, and a comparison to the estimates from the VisualInfo Sizer tool and the VisualInfo Client Application. A more thorough description of how to analyze the trace is given in 16.1, “A Social Security Benefits Scenario” on page 183.

17.1.7.1 Sample Trace Information

The trace was produced with FRNODBIM.CMD, which gives the results of Folder Manager trace in file FMDUMP.LOG. The API calls to the Folder Manager made from this application were:

- SimLibCatalogObject
- SimLibCloseAttr
- SimLibCreateItem
- SimLibDeleteItem
- SimLibGetItemAffiliatedTOC
- SimLibGetItemXref
- SimLibGetTOC
- SimLibGetTOCData
- SimLibLogon
- SimLibOpenItemAttr

- SimLibOpenObject
- SimLibSearch
- SimLibStoreNewObject
- Ip2GetContentClassHandlers
- Ip2GetTOCUpdates
- Ip2ListContentClasses

Many of the API calls made in this customized application are the same as in the VIBB coded application (customer scenario 1). However, the calls in this application seem to be almost equal to the calls made by the standard VisualInfo client application. Estimates based on the VisualInfo client application can, therefore, be assumed to give quite accurate figures about performance in this application.

17.1.7.2 Type and Number of LSFRs

The LSFRs from the trace results are compared to the estimates in Table 32. The estimates are from Table 1 on page 38, the Sizer tool, and lab tests.

Table 32. Trace Results from Customer Application/Estimated LSFRs

Function	**Trace results**	**LSFRs**
Logon	2	2
Scan/Import	3	6
Store (Index)	11	4
Open Document	2-4	4
Search	Min. 4	7
Note/Comment	4	

The number of LSFRs for scanning/importing, indexing, and storing documents are 462 LSFRs per hour. Searching and opening a document adds up to 176 LSFRs per hour. Adding a comment or a note to a document and storing it involves 44 LSFRs per hour.

The total amount produced by all users is 682 LSFRs per hour. This is approximately 31 LSFRs per hour per user, which means that VisualInfo workload is low in this environment, about 4% of a high activity workload. In this system, VisualInfo is simply used for archiving the images without using any folders, workbaskets, or workflow.

17.1.7.3 Customized Application versus VisualInfo Client Application

The comparison of the trace results from this application to the trace results from VisualInfo client application is shown in Table 33. The client application trace was taken in the lab also using OS/2 Client and OS/2 Library and Object Server.

Table 33 (Page 1 of 2). VisualInfo Client Application Trace Results with LSFRs

Function	**Trace results**	**LSFRs**
Logon	2	2
Scan/Import	3	6
Store (Index)	11	12

Table 33 (Page 2 of 2). VisualInfo Client Application Trace Results with LSFRs

Function	**Trace results**	**LSFRs**
Open Document	2-4	Max. 16
Search	Min. 4	7
Note/Comment	4	3-6

Even though almost the same API calls were issued from this application and a VisualInfo client application processing similar functions, the customized application seemed to produce fewer calls than the VisualInfo client application.

17.1.7.4 Trace Results versus the VisualInfo Sizer tool

The result from using the Sizer tool gave the following:

```
Largest hourly LSFRs based on workday      664 (OS/2 Client)
```

The difference between the Sizer result and the trace results is only 3%. The API calls to the library server produced by this application are close to the APIs issued by the VisualInfo client application. Therefore the Sizer tool, based on the client application, gives a quite accurate estimate in this case.

From Table 4 on page 42, we calculate the number of users this server can service to be 2322 users. This means, of course, that the workload consists only of image processing.

17.1.7.5 Response Time

Table 34 shows the response times for the functions from tracing this application.

Table 34. Response Times from Folder Manager Trace

Function	**Response Times**
Logon	11
Scan/Import	67/29
Store (Index)	31
Open Document	10
Search	11
Note/Comment	5

The response times in this table are end-to-end response times including client processing, server, and network time. During this trace, only two users were logged on. The network time in this 16 Mbps token-ring LAN is low even with all 22 users working at the same time. The client processing time is assumed to be about one second in all cases.

17.1.8 Conclusions

Since the Library Server API calls made by this application are so close to those made by VisualInfo client application, the VisualInfo Sizer tool is a good method of estimating capacity in this environment. The difference between the trace results and the Sizer results was only 3%. Looking at the total number of LSFRs the server can handle providing a good throughput, the number of users in this system can be 2322. This is true if there were no other applications and the amount of input and processed documents stays the same. However, this is probably not the case. The use of the VisualInfo Sizer tool is recommended to estimate the values for your situation.

In a small, fast token-ring LAN environment such as this, the network time is minimal. Even though the workload increases heavily, the portion of network time remains low.

Part 5. Appendixes

- RS/6000 Web Server Sizing Guide
- OS/2 Client Trace Output
- MVS/ESA Trace Output
- MVS Library Server Statistics (FRN6) Output

Appendix A. RS/6000 Web Server Sizing Guide

Welcome to the IBM RS/6000 Web Server sizing guide. The purpose of this guide is to help you determine the right size Web server for your customer. The information in this document represents a set of guidelines that can be used to approximate the size of a server. There is also a brief comparison of IBM RS/6000 servers to other servers that have published WebStone numbers.

Disclaimer: This document is being retained on the RS/6000 Web Site for reference purposes. The performance numbers in this paper reflect results on the listed hardware models and software versions. They do *not* represent the best possible performance with recently announced RS/6000 hardware and Web Server software.

A.1 Overview

At the time this document was written the SPECweb benchmark was not available. This guide has been prepared with results from the WebStone benchmark. WebStone is a proposed benchmark standard from Silicon Graphics, Inc. More specific information on WebStone, some performance white papers, and the benchmark itself can be obtained from:

`http://www.sgi.com/Products/WebFORCE/WebStone.`

Future versions of this sizing guide will be based on relevant data from the SPECweb benchmark.

While every effort has been made to include as much information as possible, this guide is only one of many resources available to assist you in developing IBM Web server solutions.

A.1.1 The Right Sizing Questions to Ask

Asking the right questions is a big step in providing the right solution.

A.1.1.1 Questions From Your Client

- How big an RS/6000 do I need for a Web server?
- How many “hits” per day can an RS/6000 handle?
- What is the maximum number of clients I can support on an RS/6000?

In order to best answer these questions, you need to ask your client some questions. The rest of this document will help you understand how to take the answers and design a successful Web solution.

A.1.1.2 Questions For Your Client

1. Is this web server going to be an Internet or intranet server?
2. What is the potential demand for access to this site?
3. What is the speed of your connection to the Internet?
4. How many pages will you be serving?
5. What is the average file size of pages you are serving?
6. Will the Web server be generating data for access?

A.2 Know the Target Environment

When sizing a Web server, the most important consideration is the size of the target audience.

A.2.1 Internet

The Internet has been growing at a phenomenal pace. It connects each new user with a vast amount of information covering every interest worldwide from classic cars to politics to investments. Organizations put their web servers on the Internet to make their products and information accessible by a global audience. Sizing a web server for the Internet can be a very difficult task. The Internet includes millions of interconnected individuals who are navigating from one web server to the next in search of information that has value to them. It is hard to estimate how popular a site may become.

Take the case of an IBM Web server that was set up to follow the chess match between Deep Blue, an IBM SP supercomputer, and world chess champion Gary Kasparov.

The original project plan for the site was to accommodate around 200,000 accesses per day. On day one of the match, the uniprocessor Web server registered over 5 million hits during the first game. During that game, the server was very difficult to reach, and many users were frustrated. Some hard work and 48 hours later, the games were served by the original uniprocessor system and six SP2 nodes, all mirroring the data from the match. This increased the data serving capability above demand, and got the Internet chess community back online for the later games. In fact, as the site became more popular due to the improved access, the hit counts again began to rise, prompting the addition of three additional SP2 nodes.

A.2.2 Intranets

Intranets are private nets that use the same standards and protocols of the public Internet. Intranets are rapidly deploying internal Web sites as the new network-centered corporate computing platform. An Intranet Web site dissolves departmental, geographic, and technical boundaries by creating a universal way to connect people to people and people to information.

Sizing a web server for an intranet is considerably easier than sizing one for the Internet. The total number of potential users can be determined more accurately by using the total number of employees in the relevant department or the entire company.

A.3 Sizing Factors

Let′s look at some key sizing factors.

A.3.1 Bandwidth

In working with a customer to size the Web solution, it is important to understand the implications of the speed of the networking connection to the Web server. More often than not, many potential Web content providers are very focused on the vague ″hits per day″ quantity. The level of traffic that a particular Web server can support will be dependent on the server type, the content accessed on the server and the speed of the connection of the server to the Internet environment.

An Internet service provider will deliver a connection at a defined speed. Three of the most common speeds are: ISDN (128 kbps), T1(1.544 Mbps), and T3 (45 Mbps). For an Intranet environment, common LAN speeds are 10 Mbps (over Ethernet) and 100 Mbps (over fast Ethernet or FDDI). Figure 48 shows the inter-relationship between the average Web transaction size, the speed of the networking topology, and the maximum theoretical ′hits per second′.

As the average Web transaction size increases, the maximum number of transactions decreases. Sites that plan on being mostly text based will have average transactions sizes around 1-5 KB, most well-designed sites with a mix of text and graphics intended for access by modem users are in the 10 KB per transaction size and sites with a substantial portion of multimedia content can exceed 100 KB per transaction.

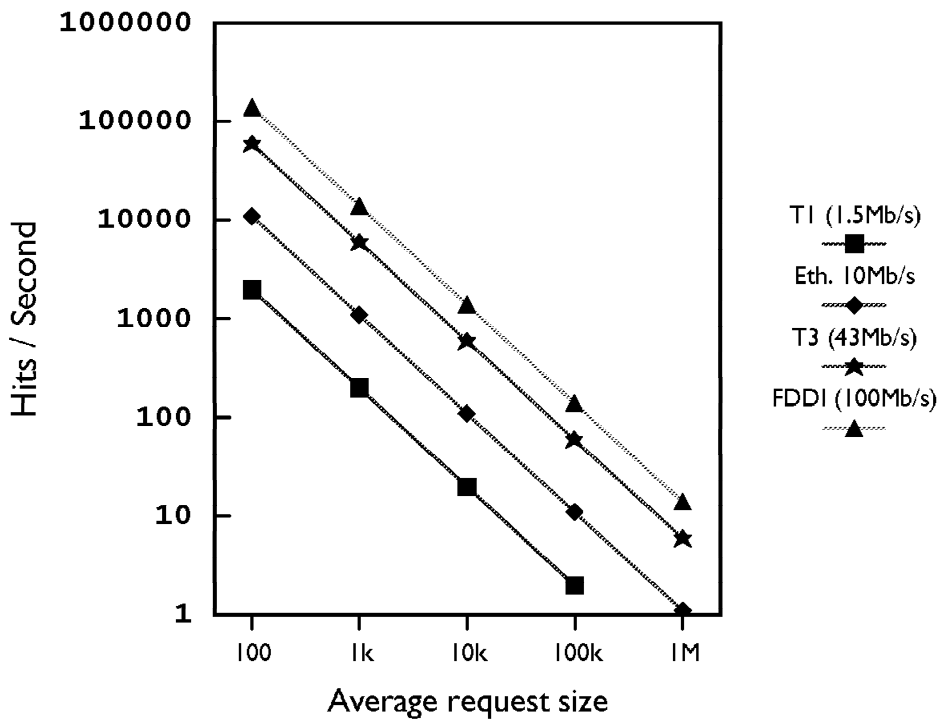


Figure 48. Hits vs. Network Speed. The relationship between the speed of the network, the average request size and the maximum theoretical ″hits per second″.

To translate this data into a number of hits for an approximately 8-hour peak usage period, multiply the hits per second by 30,000.

A.3.2 Content Type

The physical size of the Web content is important in looking at the resources required for a server, indicating the necessary data storage requirements. Additionally, when the content on the Web server is dynamically generated, substantial processing resources may be required. Dynamic content on a Web site can be generated in many ways, from a simple counter that displays the number of hits that a page has received to a system that uses analysis of user clicks to tailor the information (and advertisements in some cases) that the user sees at the site. To gain some insight into the impact of simple content generation on performance, we will discuss results from WebStone-1.1 dynamic workload models.

A.3.3 Number of Clients

The number of simultaneous users of a site is very challenging to characterize. Unlike other types of client-server architectures, the weight of an individual client on the Web server is quite small and short-lived. Connections to a Web server are traditionally stateless sessions that begin with an open from the client, a request for data, the server reply with data, and the session close. Depending on the speed of the network connection, the size of the data requested and the server load, this session can last from tenths to tens of seconds. The degradation of performance with increasing client load is best seen through the WebStone-1.1 average latency data.

A.4 WebStone Benchmarks

The WebStone benchmark is a simple benchmark that has been used to characterize uniprocessor Web server performance at the entry level. The benchmark uses a cluster of systems to query a Web server for a series of pages defined in a workload. The benchmark allows an adjustable number of simulated clients to connect to the server under test.

A.4.1 Performance Overview

Results reported include the total number of HTTP connections per second, the average latency, the errors/second, a load scaling factor and the throughput from the server under test in Mbps. For comparison, Figure 49 on page 207 and Figure 50 on page 207 summarize the uniprocessor results with an IBM 43P, a SGI Challenge S, and a Sun Netra i20 running the WebStone 1.0 benchmark.

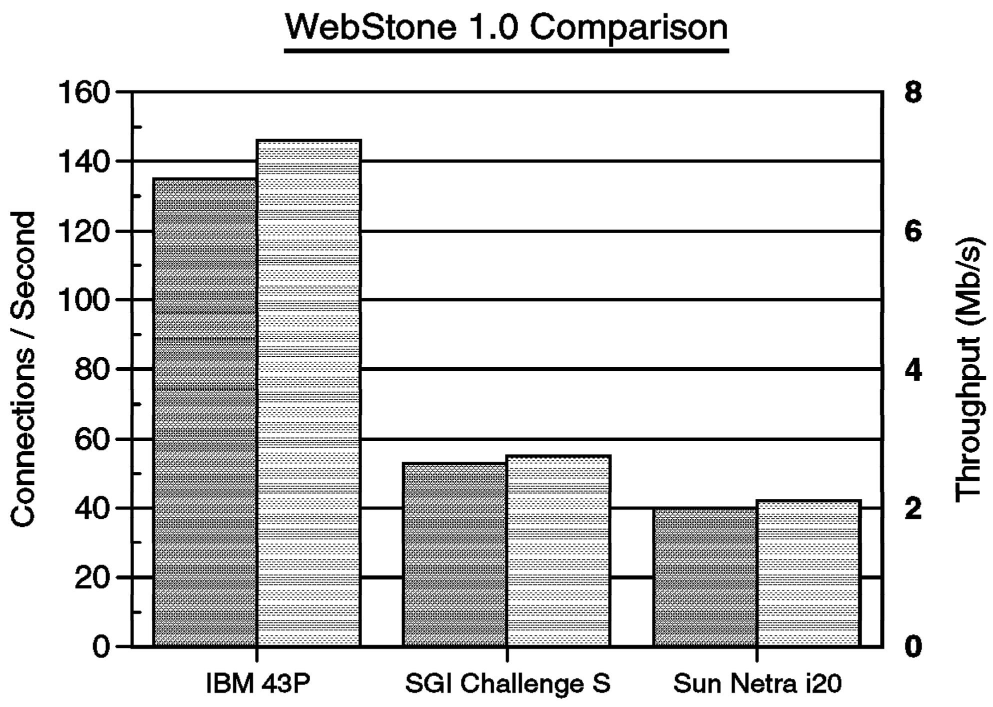


Figure 49. WebStone Throughput. Summary of major results with WebStone 1.0, bars on the left and right for each platform are for left and right axes, respectively.

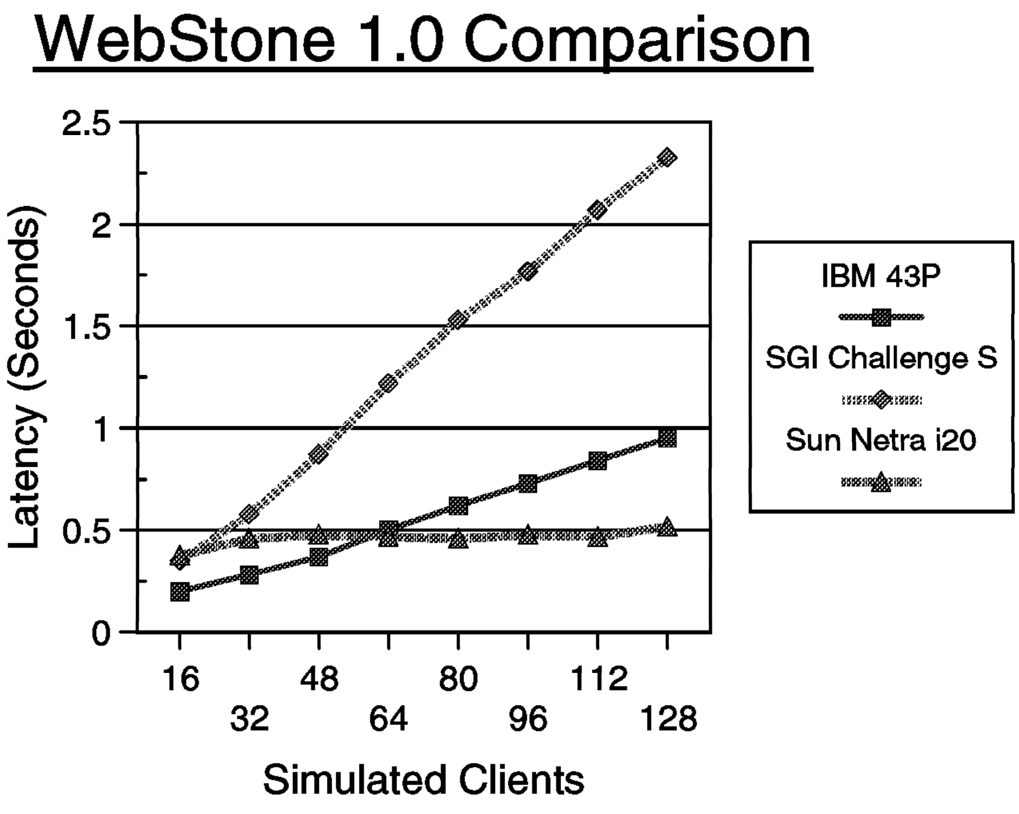


Figure 50. WebStone Latency

The IBM 43P is substantially faster than the SGI and Sun entry Web offerings. The SGI and Sun performance data we re generated by Silicon Graphics, Inc. Figure 50 is an interesting illustration of how not to handle a high volume of Web traffic. The SGI system handles all requests to the server. Due to speed difference between the 43P and the SGI, it has a much higher latency at 128 simulated clients. The SGI benchmark of the Sun system observed that the Sun system simply turned away connections as it became saturated, thereby maintaining a low latency on the connections acknowledged. Data source on 12/15/1995:

`http://www.sgi.com/Products/WebFORCE/We bStone/sun-ss20/sun-ss20.html`

A.4.2 Test Configuration

Configuration files (both the test environment and workload) for the above benchmarks are shown in Figure 51 and Figure 52 on page 209.

```
# BENCHMARK PARAMETERS -- EDIT THESE AS REQUIRED
ITERATIONS="3"
MINCLIENTS="16"
MAXCLIENTS="160"
CLIENTINCR="16"
TIMEPERRUN="10"

# SERVER PARAMETERS -- EDIT AS REQUIRED
SERVER="www43p"
PORTNO=80
SERVERINFO=lscfg
OSTUNINGFILES=
WEBSERVERDIR="/home/netscape"
WEBDOCDIR="$WEBSERVERDIR/docs"
WEBSERVERTUNINGFILES=

# WE NEED AN ACCOUNT WITH A FIXED PASSWORD, SO WE CAN REXEC
# THE WEBSTONE CLIENTS
CLIENTS="sp1n13 sp1n14 sp1n15 sp1n16"
CLIENTACCOUNT=root
CLIENTPASSWORD=root
CLIENTINFO=lscfg
TMPDIR=/tmp
```

Figure 51. Test Bed. Defines parameters for test including number of clients and equipment for the test.

```
# Silicon Surf model: pages and files to be tested for.
8
40 2
/file2k.html
/file3k.html
25 2
/file1k.html
/file5k.html
15 2
/file4k.html
/file6k.html
5 1
/file7k.html
4 4
/file8k.html
/file9k.html
/file10k.html
/file11k.html
4 5
/file12k.html
/file14k.html
/file15k.html
/file17k.html
/file18k.html
6 1
/file33k.html
1 1
/file200k.html
```

Figure 52. File Set. Defines the file workload; each section of .html files is a "page" to be downloaded.

A.4.3 Web Server Configurations

These test configurations and a dynamic page content configuration were run on the following Web server hardware system to look at the scaling properties of RS/6000 hardware in this application environment.

A.4.3.1 Hardware

RS/6000 43P-133 (7248-133) - Single PowerPC 604 Processor

- 133 MHz PowerPC 604 (4.72 SPECint95, 3.76 SPECfp95)
- 512 KB synchronous L2 cache
- 128 MB Memory
- 2.2 GB Disk
- Integrated SCSI and Ethernet controllers

RS/6000 590 - Power 2 Architecture

- 66 MHz POWER2 (3.33 SPECint95, 10.4 SPECfp95)
- 256 KB synchronous L2 cache
- 128 MB Memory
- 2 GB Disk
- Microchannel Ethernet controller

- Microchannel FDDI controller

RS/6000 J30 - SMP PowerPC 601, 75 MHz

- 4way PowerPC 601 (78.2 SPECint_base_rate95, 89.3 SPECfp_base_rate95)
- 32 KB synchronous L1 cache / processor
- 1 MB synchronous L2 cache / processor
- 128 MB Memory
- 4.5 GB Disk
- Microchannel Ethernet controller
- Microchannel FDDI controller

RS/6000 SP2 - Parallel Processor

- SP 66 Thin 2 nodes (39H equivalent) (3.42 SPECint95, 10.2 SPECfp95)
- 2MB L2 cache/node
- Memory 128 MB/node
- Disk 4 GB/node
- Microchannel FDDI controller

A.4.3.2 Operating System

AIX 4.1.4 + (PTF 443072)

A.4.3.3 Server Software

Netscape Commerce Server Version 1.12 (NCS)

A.4.3.4 Tuning

Maximum processes per user set to 256

DNS reverse lookup off

Netscape MaxProcs 200 (UNI, SMP); 128 (total on all SP2 nodes, for example, 64/node on 2 server configuration)

A.4.3.5 Web Client Configuration

Four Nodes on a RS/6000 SP1+ drove the Web client processes. Each node (370 equivalent) was equipped with 128 MB RAM, 1 GB local disk and microchannel Ethernet and FDDI adapters. For the SP2 benchmarks, two 39H and one 590 nodes were used as Web client drivers over FDDI.

A.5 Results of Static Tests

A major portion of the content on the Web is static. This includes both images and textual data. The CPU resources required to serve such data are minimal. The IBM RS/6000 product line has a large performance range from entry systems to highly parallel processing configuration and this range in performance is observable in static content WebStone benchmarks only when the correct processor and network topology combination are achieved. Figure 53 on page 211 illustrates the range of HTTP connection rates with Web servers based from the entry 43P through four SP2 nodes used in parallel.

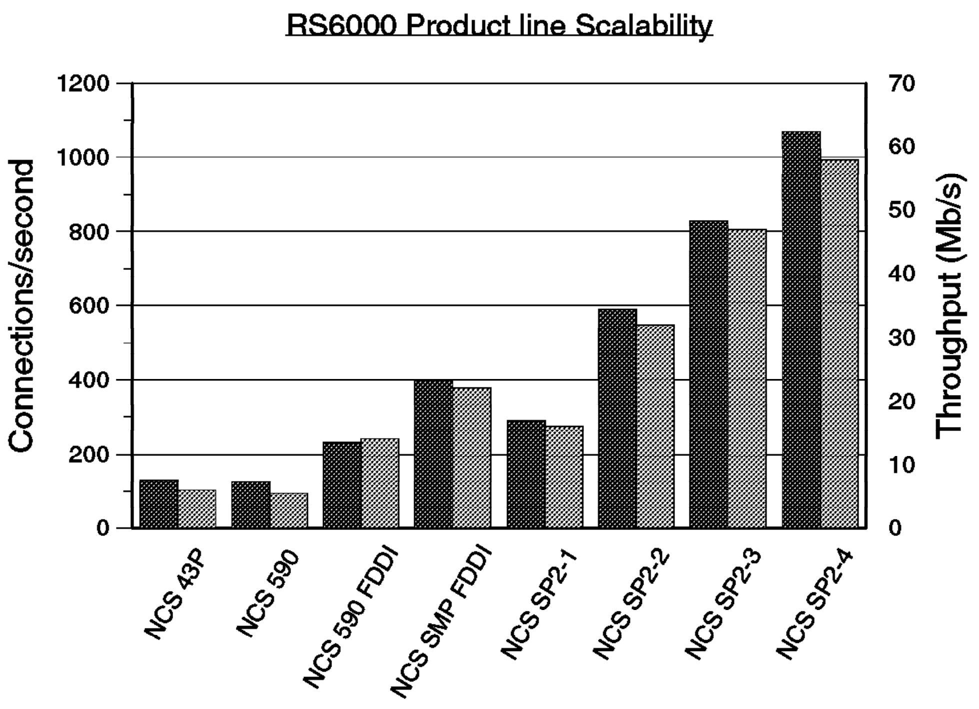


Figure 53. Static Data, RS/6000 Product Line. Static workload results with the WebStone 1.1 benchmark.

A typical HTTP connection consists of a client open, client request, server header and data response and connection shutdown. The average response size is approximately 7 Kb. With FDDI, the bottleneck is at the CPU performance level for the midrange to high-end solutions.

As Figure 53 shows, a site that has static data and relatively small networking bandwidth capabilities needs no more than a properly configured entry uniprocessor system. At Ethernet networking speeds, even the 43P is network-bound and not CPU-bound. As sites expand and anticipate the need to accommodate T3 (45 Mbps), speeds or for Intranet (LAN-based) sites that require a database or other application on the Web server, a midrange uniprocessor, SMP or SP2 system is called for.

A.6 Results of Dynamic Tests

When a Web server responds to users in a more dynamic way, we see a much stronger case for increased computing power at the server. Unfortunately, there has been very little work that characterizes the exact impact of various types of dynamic workload on HTTP performance. With WebStone-1.1, we are able to observe the effects of dynamic workload in a very simple way. Instead of returning an existing file, the Web server being tested generates a file of random characters. Figure 54 on page 212 shows throughput in Mbps over a range of RS/6000 hardware using both 25% and 50% dynamically generated page content.

Performance bottlenecks in this benchmark shift from the network speed to the processing speed of the Web server. In some configurations, there are still situations where the performance is network-bound. For example, comparing the results for the 590 with Ethernet and FDDI, we see that the output is network bound at 25% dynamic content generation and CPU bound at 50%. The SMP system is heavily bound by the speed of the Ethernet network at 25% dynamic page content.

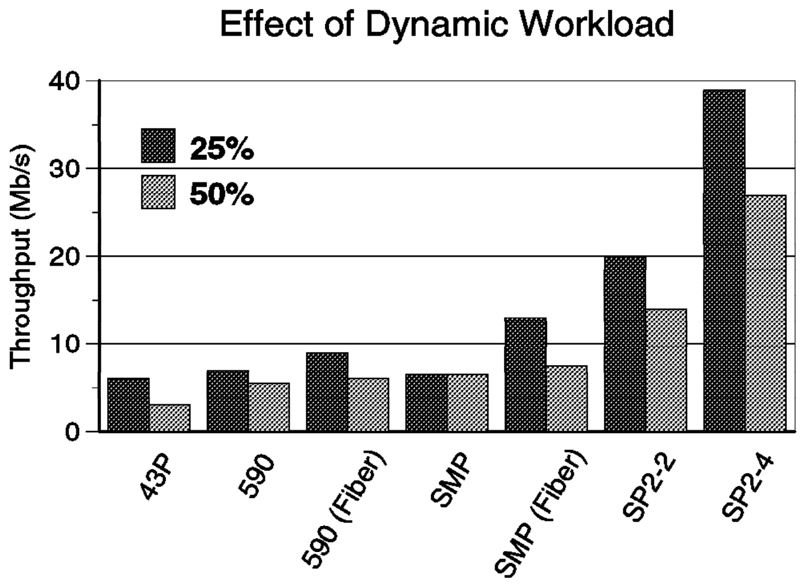


Figure 54. Dynamic Workload, RS/6000 line. Throughput in Mbps as a function of processor, network topology, and percentage of workload dynamically generated . The left-hand bar for each configuration is at 25% dynamic page content and the right-hand bar is at 50% dynamic page content.

A.7 Results of Client Workload

As the physical number of clients increases, one expects overall response time from the server to increase. Over the client range available for testing (WebStone 1 only produces statistics for up to 256 clients), we see reasonable behavior in server latency with increasing numbers of clients. In Figure 55 on page 213 and Figure 56 on page 214, we see the combined effects of dynamic workload and client quantity.

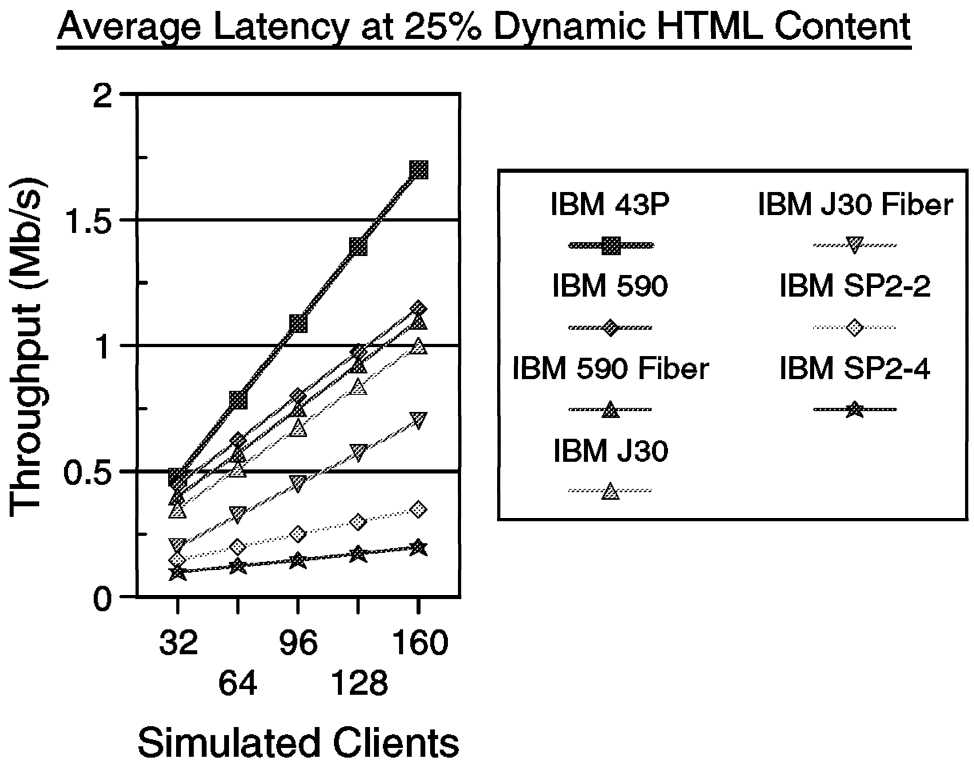


Figure 55. 25% Dynamic HTML. Latency in seconds vs. number of simulated clients.

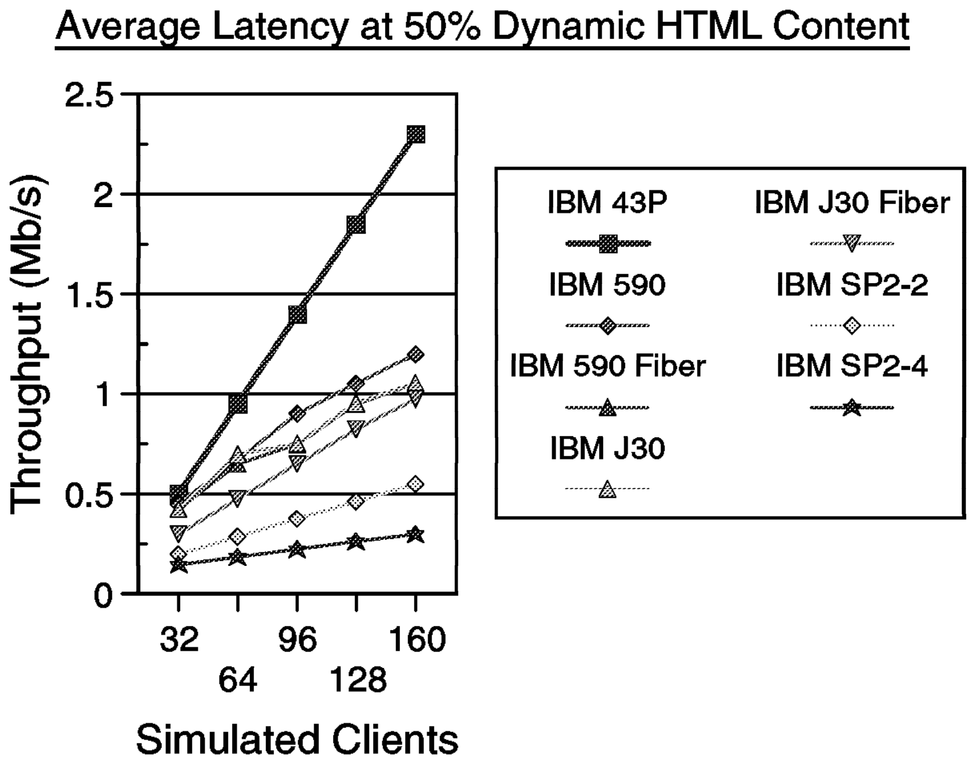


Figure 56. 50% Dynamic HTML. Latency in seconds vs. number of simulated clients.

Figure 55 on page 213 is for 25% dynamic content and Figure 55 on page 213 is for 50% dynamic content. Solutions that will perform best in a real-world, large client workload environment have lower slopes in these graphs.

Looking at Figure 55 on page 213 and Figure 56, we see the combined effects of dynamic workload and we see that there is a close to linear relationship between number of clients and server latency over the client range displayed. The slope of these lines is a strong indicator of the scalability of the solution. At 160 clients and 50% dynamic content, the 43P has an average latency of 2.3 seconds, approaching the upper range of acceptable delay. Higher performance solutions have lower slopes, indicating a more graceful degradation under heavy workload. In general, these workloads reflect moderate traffic rates.

Consider a vendor site with an average of 64 KB data per Web page (a mix of text, small icons, and a couple of small pictures on each page). If a typical visitor accesses 10 pages at the site, then during an 8 hour period 32,000 visitors could be accommodated with a 43P at 50% dynamic content workload. We have successfully served data to over 500 simulated simultaneous clients using a 590 over FDDI with 128 MB of RAM.

A.8 Recommendations and Conclusions

In general, Internet sites that are connected by T1 lines and Ethernet-LAN connected Intranet sites with largely static data are adequately served by a entry uniprocessor system with adequate disk storage for the content provided. It is important to have enough RAM to accommodate both the HTTP server processes

(64 MB was adequate in our testing) and for file caching of page content that resides on disk.

Sites with high-bandwidth connections to the Internet and Intranet sites that can utilize FDDI will benefit from midrange and SMP solutions. Sites that will generate significant Web content in response to user actions or potential E-Commerce sites should consider such systems even if they are connected by T1 lines to the Internet or Ethernet-LAN to the Intranet.

The best possible scale-up with increasing client workload is accomplished through the use of IBM SP systems, or clusters of uniprocessor systems. This solution depends upon the usage of Round-Robin Domain Name Service to direct clients to multiple nodes running mirrors of the Web site. In addition, to get the linear scale-up shown in this section, care must be taken to make sure that each node has an adequate dedicated connection to the Internet. In the data here, 4 nodes of the SP2 would be well suited to filling the full band width of a T3 line for distribution of static Web content. An SP solution brings strengths of better management and faster intra-node communication of shared data. For the Intranet, the common practice of resource consolidation into SP systems creates a natural location for an intranet Web server.

Appendix B. Output from the Client Trace for OS/2

This is a sample of the folder manager trace that is produced in the OS/2 client by the command file FRNODBIM.CMD. It contains the API calls for logon, scanning, indexing and storing a document in a workbasket, and logoff.

The trace format and entries may change without notice.

```
SS 00000002 13:27:19.67 SIMLIBLOGON
E  00000002 13:27:29.52 SIMLIBLOGON                      Returned 0
SS 00000001 13:27:29.91 SIMLIBLISTCLASSES
E  00000001 13:27:29.91 SIMLIBLISTCLASSES                Returned 0
SS 00000001 13:27:29.91 SIMLIBGETCLASSINFO
E  00000001 13:27:29.91 SIMLIBGETCLASSINFO               Returned 0
SS 00000001 13:27:29.91 SIMLIBGETCLASSINFO
E  00000001 13:27:29.91 SIMLIBGETCLASSINFO               Returned 0
SS 00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.91 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS
E  00000001 13:27:29.94 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000001 13:27:29.97 SIMLIBLISTCLASSES
E  00000001 13:27:29.97 SIMLIBLISTCLASSES                Returned 0
SS 00000001 13:27:29.97 SIMLIBGETCLASSINFO
E  00000001 13:27:29.97 SIMLIBGETCLASSINFO               Returned 0
SS 00000001 13:27:29.99 IP2LISTATTRS
E  00000001 13:27:29.99 IP2LISTATTRS                     Returned 0
SS 00000001 13:27:29.99 IP2LISTCONTENTCLASSES
E  00000001 13:27:30.20 IP2LISTCONTENTCLASSES            Returned 0
SS 00000001 13:27:30.20 IP2LISTWORKBASKETS
E  00000001 13:27:30.20 IP2LISTWORKBASKETS               Returned 0
```

```
SS 00000001 13:27:30.20 IP2LISTWORKFLOWS
E  00000001 13:27:30.20 IP2LISTWORKFLOWS                 Returned 0
SS 00000001 13:27:31.33 IP2GETLIBSESSIONINFO
E  00000001 13:27:31.36 IP2GETLIBSESSIONINFO             Returned 0
SS 00000005 13:27:31.91 IP2LISTWORKBASKETS
E  00000005 13:27:31.91 IP2LISTWORKBASKETS               Returned 0
SS 00000005 13:27:31.91 IP2GETWORKBASKETINFO
E  00000005 13:27:31.94 IP2GETWORKBASKETINFO             Returned 0
SS 00000005 13:27:31.97 IP2LISTWORKFLOWS
E  00000005 13:27:31.97 IP2LISTWORKFLOWS                 Returned 0
SS 00000005 13:27:31.97 IP2GETWORKFLOWINFO
E  00000005 13:27:31.97 IP2GETWORKFLOWINFO               Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSES
E  00000005 13:27:34.73 SIMLIBLISTCLASSES                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS             Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
```

```
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS
E  00000005 13:27:34.73 SIMLIBLISTCLASSVIEWS                Returned 0
SS 00000001 13:27:38.45 SIMLIBGETITEMTYPE
E  00000001 13:27:38.64 SIMLIBGETITEMTYPE                   Returned 0
SS 00000001 13:27:38.64 SIMLIBGETTOC
E  00000001 13:27:38.96 SIMLIBGETTOC                        Returned 0
SS 00000001 13:27:38.96 IP2CLOSETOC
E  00000001 13:27:38.96 IP2CLOSETOC                         Returned 0
SS 00000001 13:27:45.61 IP2GETWORKFLOWINFO
E  00000001 13:27:45.83 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:27:45.83 IP2GETWORKFLOWINFO
E  00000001 13:27:45.97 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:27:45.97 IP2GETWORKFLOWINFO
E  00000001 13:27:46.09 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:27:46.09 IP2GETWORKFLOWINFO
E  00000001 13:27:46.25 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:27:46.25 IP2GETWORKFLOWINFO
E  00000001 13:27:46.42 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:27:46.42 IP2GETWORKFLOWINFO
E  00000001 13:27:46.58 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:27:46.58 IP2GETWORKFLOWINFO
E  00000001 13:27:46.77 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:27:46.77 IP2GETWORKFLOWINFO
E  00000001 13:27:46.91 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:27:46.91 IP2GETWORKFLOWINFO
E  00000001 13:27:47.03 IP2GETWORKFLOWINFO                  Returned 0
SS 00000001 13:28:00.28 SIMLIBGETCLASSINFO
E  00000001 13:28:00.28 SIMLIBGETCLASSINFO                  Returned 0
SS 00000001 13:28:00.28 SIMLIBGETATTRINFO
E  00000001 13:28:00.28 SIMLIBGETATTRINFO                   Returned 0
SS 00000001 13:28:00.28 SIMLIBGETATTRINFO
E  00000001 13:28:00.28 SIMLIBGETATTRINFO                   Returned 0
SS 00000001 13:28:00.28 SIMLIBGETATTRINFO
E  00000001 13:28:00.28 SIMLIBGETATTRINFO                   Returned 0
SS 00000001 13:29:12.79 SIMLIBGETATTRINFO
E  00000001 13:29:12.82 SIMLIBGETATTRINFO                   Returned 0
SS 00000001 13:29:12.82 SIMLIBGETATTRINFO
E  00000001 13:29:12.82 SIMLIBGETATTRINFO                   Returned 0
SS 00000001 13:29:12.82 SIMLIBGETATTRINFO
E  00000001 13:29:12.82 SIMLIBGETATTRINFO                   Returned 0
SS 00000001 13:29:12.82 SIMLIBGETATTRINFO
E  00000001 13:29:12.82 SIMLIBGETATTRINFO                   Returned 0
```

```
SS 00000001 13:29:12.82 SIMLIBGETATTRINFO
E  00000001 13:29:12.82 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:12.82 SIMLIBGETATTRINFO
E  00000001 13:29:12.82 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:12.82 SIMLIBGETATTRINFO
E  00000001 13:29:12.82 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:12.82 SIMLIBGETATTRINFO
E  00000001 13:29:12.82 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:12.86 SIMLIBCREATEITEM
E  00000001 13:29:13.07 SIMLIBCREATEITEM                     Returned 0
SS 00000001 13:29:13.07 SIMLIBCREATEOBJECT
E  00000001 13:29:13.75 SIMLIBCREATEOBJECT                   Returned 0
SS 00000001 13:29:13.88 SIMLIBOPENOBJECT
E  00000001 13:29:14.18 SIMLIBOPENOBJECT                     Returned 0
SS 00000001 13:29:14.46 SIMLIBWRITEOBJECT
E  00000001 13:29:14.46 SIMLIBWRITEOBJECT                    Returned 0
SS 00000001 13:29:14.46 SIMLIBWRITEOBJECT
E  00000001 13:29:14.49 SIMLIBWRITEOBJECT                    Returned 0
SS 00000001 13:29:14.49 SIMLIBCLOSEOBJECT
E  00000001 13:29:16.45 SIMLIBCLOSEOBJECT                    Returned 0
SS 00000001 13:29:16.84 IP2ADDWORKBASKETITEM
E  00000001 13:29:17.12 IP2ADDWORKBASKETITEM                 Returned 0
SS 00000001 13:29:19.27 SIMLIBGETCLASSINFO
E  00000001 13:29:19.27 SIMLIBGETCLASSINFO                   Returned 0
SS 00000001 13:29:19.27 SIMLIBGETATTRINFO
E  00000001 13:29:19.27 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:19.27 SIMLIBGETATTRINFO
E  00000001 13:29:19.27 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:19.27 SIMLIBGETATTRINFO
E  00000001 13:29:19.27 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:41.68 SIMLIBGETITEMTYPE
E  00000001 13:29:41.78 SIMLIBGETITEMTYPE                    Returned 0
SS 00000001 13:29:41.78 SIMLIBGETITEMSNAPSHOT
E  00000001 13:29:41.99 SIMLIBGETITEMSNAPSHOT                Returned 0
SS 00000001 13:29:42.00 SIMLIBGETCLASSINFO
E  00000001 13:29:42.00 SIMLIBGETCLASSINFO                   Returned 0
SS 00000001 13:29:42.00 SIMLIBGETATTRINFO
E  00000001 13:29:42.00 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:42.00 SIMLIBGETATTRINFO
E  00000001 13:29:42.00 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:42.00 SIMLIBGETATTRINFO
E  00000001 13:29:42.00 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:42.00 SIMLIBGETATTRINFO
E  00000001 13:29:42.00 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:42.00 SIMLIBGETATTRINFO
E  00000001 13:29:42.00 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:42.00 SIMLIBGETATTRINFO
E  00000001 13:29:42.00 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:42.00 SIMLIBGETATTRINFO
E  00000001 13:29:42.00 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:42.00 SIMLIBGETATTRINFO
E  00000001 13:29:42.00 SIMLIBGETATTRINFO                    Returned 0
SS 00000001 13:29:42.48 IP2LISTHISTORY
E  00000001 13:29:42.93 IP2LISTHISTORY                       Returned 0
SS 00000001 13:30:04.64 SIMLIBLOGOFF
```

Appendix C. Trace APIs and LSFR Results from MVS/ESA with OS/2 Client

The following trace results are from the customized application described in Chapter 16, “Customer Scenario #2” on page 183. In this application, the VisualInfo building blocks were used. The type and number of the library server calls tracing each transaction are listed here.

C.1 Scanning

Scanning, indexing, and storing the document involved the following APIs interacting with the library server:

- SimLibCreateItem
- SimLibCreateObject
- SimLibOpenObject
- SimLibCloseObject
- Ip2AddWorkBasketItem
- SimLibGetItemSnapshot
- Ip2ListHistory

Tracing the first transaction, the total number of library server calls scanning, indexing, and storing a two-page document was seven API calls. Each API was called once.

C.2 Workbasket and Search by Person

The trace results from the second transaction gave the following information:

In this trace, opening a user′s personal workbasket and a document in it produced the following library server calls:

- SimLibGetItemType
- SimLibGetTOC
- SimLibGetTOCData
- Ip2ListHistory
- SimLibGetItemAffiliatedTOC
- SimLibOpenObject
- SimLibSearch
- SimLibDeleteItem

The total number of calls to the library server was 16.

Listing all the documents of one customer made the following library server API calls:

- SimLibSearch
- SimLibGetItemType
- SimLibGetTOC
- SimLibGetTOCData
- SimLibDeleteItem

The total number of calls was seven.

Adding the document into a new folder and moving it to a new workbasket involves the following library server API calls:

- SimLibGetItemType
- SimLibGetTOC
- SimLibGetItemAffiliatedTOC
- SimLibOpenObject
- SimLibGetItemSnapshot
- Ip2ListHistory
- Ip2ActivateItem
- Ip2RemoveWorkbasketItem
- SimLibOpenItemAttr
- SimLibCloseAttr
- SimLibCreateItem
- SimLibAddFolderItem
- Ip2AddWorkbasketItem
- SimLibGetItemXref
- SimLibSearch
- SimLibGetTOCData
- SimLibDeleteItem

The total number of calls was 33.

C.3 Functions in Personal Workbasket

The API calls to the library server by each function during tracing the third transaction are in the following list.

Adding a comment to one document:

- Ip2ListHistory
- SimLibGetItemAffiliatedTOC
- SimLibCreateObject
- SimLibWriteObject
- SimLibCloseObject
- SimLibOpenItemAttr
- SimLibCloseAttr
- SimLibGetItemType
- SimLibGetItemSnapshot

The total number of calls was 10.

Viewing the status of one document:

- Ip2ListHistory
- SimLibGetTOC
- SimLibGetItemType
- SimLibGetItemAffiliatedTOC
- SimLibOpenObject

The total number of calls was five.

Printing a document:

- SimLibGetItemType
- Ip2TOCCount
- SimLibGetTOC

- SimLibGetTOCData

The total number of calls was five.

Viewing at an index:

- SimLibGetItemType
- Ip2TOCCount
- SimLibGetTOC
- SimLibGetTOCData

The total number of calls was five.

Viewing at an archive:

- SimLibSearch
- SimLibGetItemType
- SimLibGetTOC
- SimLibGetTOCData

The total number of calls was five.

Putting a document on hold:

- Ip2ListHistory
- Ip2ActivateItem
- Ip2SuspendItem
- SimLibOpenItemAttr
- SimLibCloseAttr
- SimLibGetItemType
- SimLibGetItemSnapshot

The total number of calls was seven.

Releasing the document:

- Ip2ListHistory
- Ip2ActivateItem
- SimLibGetItemType
- SimLibGetItemSnapshot

The total number of calls was four.

Dividing one document into two, indexing and saving the new document:

- Ip2ListHistory
- SimLibGetItemType
- SimLibGetItemAffiliatedTOC
- SimLibOpenObject
- SimLibCloseObject
- SimLibCreateItem
- SimLibCreateObject
- Ip2AddWorkbasketItem
- SimLibDeleteItem
- SimLibGetItemSnapshot
- Ip2ActivateItem
- Ips2RemoveWorkbasketItem
- SimLibOpenItemAttr
- SimLibCloseAttr

- SimLibSearch
- SimLibGetTOC
- SimLibGetTOCData
- SimLibAddFolderItem
- SimLibGetItemXref

The total number of calls was 43.

Stamping:

- Ip2ListHistory
- SimLibOpenItemAttr
- SimLibCloseAttr
- Ip2ActivateItem
- SimLibGetItemType
- Ip2RemoveWorkbasketItem

The total number of calls was eight.

C.4 Searching

The API calls to the library server in the fourth trace from a search based on benefit and customer′s archive were the following:

Search by benefit:

- SimLibSearch
- SimLibGetItemType
- SimLibGetItemSnapshot

The total number of calls was three.

Search customer′s archived documents:

- SimLibSearch
- SimLibGetItemType
- SimLibGetTOC
- SimLibGetTOCData

The total number of calls was four.

Appendix D. MVS Library Server Statistics (FRN6) Output

This appendix lists, by transaction description, the number of library server orders executed. The data was collected using the FRN6 call provided with VisualInfo/MVS. The numbers are cumulative.

```
After Client Application Logon:
 Order      LS Order    Avg Elapsed      Avg Elapsed
 Code          Count   LS Time (MS)     OS Time (MS)

 1034 (connect)    2            55
 1134 (endtran)    2             7
 1356 (squery)     9            42
 1965 (viewsearch) 3            33

After Click on Basic Search:
 Order      LS Order    Avg Elapsed      Avg Elapsed
 Code          Count   LS Time (MS)     OS Time (MS)

 1034              8            25
 1134              8             6
 1356             11            36
 1965              9            35

After Fileroom Search against class Claim with no items returned:
 Order      LS Order    Avg Elapsed      Avg Elapsed
 Code          Count   LS Time (MS)     OS Time (MS)

 1034              9            26
 1134              9             6
 1356             11            36
 1965             10            47

After Fileroom Search against class Benefits Document (AVT00009) with
list of 14 items returned:
 Order      LS Order    Avg Elapsed      Avg Elapsed
 Code          Count   LS Time (MS)     OS Time (MS)

 1023 (checkout)   1           400
 1034             18            16
 1134             18            23
 1223 (link)      16            33
 1356             77            12
 1367 (dquery)     1           830
 1954 (setattrib)  1           750
 1965 (viewsearch)30            79
 2012 (storeitem)  1           180

After opening a document from the list created above:
 Order      LS Order    Avg Elapsed      Avg Elapsed
 Code          Count   LS Time (MS)     OS Time (MS)

 1023 (checkout)   2           395
 1034             25            13
 1134             25            93
 1223 (link)      16            33
 1356             79            13
 1367              1           830
 1512 (retrieve)   2          1575             920
 1954              1           750
 1965             33            76
 2012              1           180

After closing the document and returning to the list:
 Order      LS Order    Avg Elapsed      Avg Elapsed
 Code          Count   LS Time (MS)     OS Time (MS)

 1012 (checkin)    1           460
 1023 (checkout)   2           395
 1034             29            17
 1134             29            82
 1223 (link)      16            33
 1356 (squery)    81            18
 1367 (dquery)     1           830
 1512 (retrieve)   2          1575             920
 1954 (setattrib)  1           750
 1965 (viewsearch)34            85
 2012 (storeitem)  1           180

After closing the list of documents:
 Order      LS Order    Avg Elapsed      Avg Elapsed
 Code          Count   LS Time (MS)     OS Time (MS)

 1012 (checkin)    2           235
 1023 (checkout)   2           395
 1034             34            15
 1112 (disgard)    1           940
 1134             34            74
 1223 (link)      16            33
 1356 (squery)    85            19
 1367 (dquery)     2           455
 1512 (retrieve)   2          1575             920
 1954 (setattrib)  1           750
 1965 (viewsearch)34            85
 2012 (storeitem)  1           180

After Logoff:
 Order      LS Order    Avg Elapsed      Avg Elapsed
 Code          Count   LS Time (MS)     OS Time (MS)

 1012 (checkin)    2           235
 1023 (checkout)   2           395
 1034 (connect)   35            18
 1112 (discard)    1           940
 1134 (endtrans)  35            72
 1223 (link)      16            33
 1356 (squery)    85            19
 1367 (dquery)     2           455
 1512 (retrieve)   2          1575             920
 1954 (setattrib)  1           750
 1965 (viewsearch)34            85
 2012 (storeitem)  1           180
```

Appendix E. Special Notices

This publication is intended to help IBM marketing and support personnel more accurately predict VisualInfo or Digital Library server capacity planning requirements and end user response times for a given set of application requirements.

The information in this publication is not intended as the specification of any programming interfaces that are provided by VisualInfo and Digital Library. See the PUBLICATIONS section of the IBM Programming Announcement for VisualInfo and Digital Library for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Appendix F. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

F.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 231.

- *IBM ImagePlus VisualInfo Library and Object Servers for MVS/ESA Planning Guide*, GG24-4452
- *Image and Workflow Library: Best Practices for VisualInfo*, SG24-4792
- *DB2 for OS/390 V5 Performance Topics*, SG24-2213
- *VisualInfo Building Blocks for REXX and OS/2 Command Line*, GG24-4500
- *AS/400 Communication Performance Investigation*, SG24-4895
- *AS/400 Server Capacity Planning*, SG24-2159
- *AS/400 Performance Management*, SG24-4735
- *CICS/ESA-DB2 Interface Guide*, SG24-4536
- *DB2 for OS/390 Application Design Guidelines for High Performance*, SG24-2233

F.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs:

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbook	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037

F.3 Other Publications

These publications are also relevant as further information sources:

- *ImagePlus VisualInfo Planning and Installation Guide*, GC31-7772
- *ImagePlus VisualInfo System Administration Guide*, SC31-7774
- *DB2 for OS/390 V5 Administration Guide Volume 2*, SC26-8957
- *CICS TS for OS/390 CICS Performance Guide*, SC33-1699
- *DFSMS/MVS Object Access Method (OAM) Planning, Installation, and Storage Administration Guide for Object Support*, SC26-4918

F.4 Publications Available on the Internet

These publications are also relevant as further information sources and are available in HTML format only on the ImagePlus Home Page at: `http://www.software.ibm.com/is/image/index.htm`

- *ImagePlus VisualInfo Application Programming Guide for OS/2*, SC31-9059
- *ImagePlus VisualInfo Application Programming Reference*, (three volumes), SC31-9061, SC31-9062, SC31-9063

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download or order hardcopy/CD-ROMs redbooks from the redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders via e-mail including information from the redbook fax order form to:

In United States:	e-mail address: usib6fpl@ibmmail.com
Outside North America:	Contact information is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the ″How to Order″ section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl/

This information was current at the time of publication, but is continually subject to change. The latest information for customers may be found at `http://www.redbooks.ibm.com/` and for IBM employees at `http://w3.itso.ibm.com/`.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may also view redbook, residency and workshop announcements at `http://inews.ibm.com/`.

IBM Redbook Fax Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

- Invoice to customer number ____________________
- Credit card number ____________________

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

Note: For more information about terms, refer to the product publications found in Appendix F, “Related Publications” on page 229.

A

API. See *application program interface.*

APPC. Advanced program-to-program communication. A commonly used protocol for various network environments, such as Internet, Host communications, and LANs. FlowMark uses either APPC or TCP/IP. See also *TCP/IP.*

application program interface (API). An interface provided by the workflow manager that enables programs to be started, processes to be controlled, and operators to work with data containers.

B

base product. The product that provides the functionality required for the operation, for example, VisualInfo, Lotus Notes. This is the product called via the Service Broker Manager.

benchmark. An executable model of a workload. Benchmarks are executed in controlled environments so that measurements can be obtained of throughput, response times, and resource consumption. The measures can then be interpreted to determine the capacity of a system or, if detailed enough, can be used to develop, calibrate, and validate system models.

bit. The smallest representation of data, a 1 or 0 (on or off).

byte. A character of data, usually 8 bits.

C

cache. A storage area which can be accessed more quickly than the usual storage media. It is relatively small, so only the most recently used data is kept there.

capacity. The workload that a system can sustain and still meet response time objectives.

CAPI. Common Application Programming Interfaces. The VisualInfo primary programming interfaces.

CIF. Common Interchange File. The CIF specifies attribute names needed to import documents and folders to the VisualInfo system from other platforms.

CIU file. Common Interchange Unit file.

client. Clients access shared network resources or functionality provided by a server.

COLD. Computer Output to Laser Disc.

CPU. Central Processing Unit.

CUA. Common User Access.

D

DASD. Direct Access Storage Device. A device in which access time is effectively independent of the location of the data.

DB2/2. IBM Database 2 for OS/2, a relational database manager.

DDE. Dynamic data exchange. A OS/2 or MS Windows feature that enables data exchange between applications.

DLL. Dynamic link library. A module containing a routine that is linked at load time or runtime.

document. A transmission medium for, or depository of, information such as a report or an invoice. A VisualInfo document consists of a base part and the item information, notes with content class, annotation with content class, history and MGDS data (used with OCR). It is indexed by system-defined and user-defined attributes. A document can be contained in one or more folders.

document flow. The flow of documents through an organization. This can be through a document management system in digitized form or in hardcopy form.

document storage. A physical location where hard-copy documents are stored.

E

EDI. See *electronic data interchange.*

EDM. Electronic document management.

electronic data interchange (EDI). A computer-to-computer connection between two companies. The transaction flow for an EDI transaction is regulated through international protocol.

F

FAT. OS/2 or DOS file allocation table.

FaxRouter/2. A local area network (LAN)-based solution that provides the capability to send, receive, archive, display, distribute, print, and delete faxes from a user′s Windows or OS/2 workstation.

FlowMark. An IBM program product that manages and controls the execution of a process path or workflow.

folder. An item that can contain other folders or documents. In the VisualInfo system, a folder can be indexed by system-defined and user-defined attributes (key field information).

G

GUI. Graphical user interface.

H

HLLAPI. High-level language application program interface. HLLAPI is used by application programs for host communication, such as 3270 or 5250 screen formats.

HPFS. OS/2 high performance file system. A format for hard drives, like file allocation table (FAT).

I

IBM. International Business Machines Corporation.

icon. A picture that represents a program or function which can be started by the end user.

ICPF. IBM ImagePlus Capture Facility. A high-volume document capture subsystem which includes barcode recognition.

index class. A user-defined group of information used to store and retrieve an item or set of items. An index class identifies the type of key fields, automatic processing requirements, and storage requirement for a document or folder.

information. Facts or objects that have meaning to human beings; as opposed to *data*, which requires context and interpretation in order to become *information*. Information is formatted data. Business objects that are produced by or moved between activities and systems using information flows.

in-process server. A OLE server that is implemented as a dynamic link library is called an in-process server. It runs in the client′s address space.

IOCA. Image Object Content Architecture. A collection of constructs used to interchange and present images.

IPFO. IBM Intelligent Forms Facility. A product which supplements ICPF and supports OCR.

IT. Information Technology. The hardware, software, services and facilities which a company uses to store, process and retrieve data and information.

ITSO. International Technical Support Organization.

IWPM/2. IBM ImagePlus Workstation Program/2.

K

KB. Kilobytes (1,024 bytes of storage), usually in a disk or memory capacity environment.

Kbps. Kilobits per second (1,024 bits of data per second), usually in a data transmission environment.

L

LAN. See *local area network*.

line of business (LOB). A family of products and services having common characteristics.

LN:DI. Lotus Notes: Document Imaging support.

LOB. See *line of business*.

local area network (LAN). A system which connects workstations in the same location. It is able to transfer data at a high rate of speed, usually millions of bits per second.

LSFR. Library Server Function Request. A request block sent by the client to the library server to perform a piece of work.

M

MB. Megabytes (1,048,576 bytes of storage), usually in a disk or memory capacity environment.

Mbps. Megabits per second (1,048,576 bits of data per second), usually in a data transmission environment.

MHz. Megahertz. A measure of the speed of a workstation processor.

measurements. Information that is obtained by an instrument. Examples are CPU utilization, disk utilization, I/O per second, LSFRs per hour, response time per transaction (gathered by a test tool or by a person with a stop watch) and so on.

MGDS. Machine Generated Data Stream.

MO:DCA. Mixed Object Document Content Architecture. An IBM architecture developed to allow the interchange of object data among applications within the interchange environment and among environments.

model. This term can include predictions and projections determined by any means. Customarily, it is applied to detailed algorithms that take anticipated workload as input, rely on good measures/assumptions of resource consumption per unit of work for each significant system component, and produce as output the estimated throughput and response time of the system. Examples are: simulation models built using SNAPSHOT, GPSS, SIMULA, QGERT; analytic models such as queueing models like BEST1, or those built using any ordinary procedural language; other mathematical models, such as systems of differential equations.

MQSeries. Message Queue Series. A communications layer which establishes a connection between two systems. With MQSeries, messages can be sent and received through queues.

ms. Millisecond, or one-thousandth of a second.

O

OCR. Optical Character Recognition.

OLE. Object Linking and Embedding.

out-of-process server. An OLE server running in its own process space is called out-of-process server. This kind of OLE server is implemented as an executable file.

P

patch code. Industry standard pattern of horizontal lines; You can use patch codes to indicate the beginning of a new folder or document.

platform. The operating system environment in which a program runs. FlowMark is a distributed, cross-platform application, which can run on OS/2, AIX, and Windows.

PPDS. Page Printer Data Stream.

predictions/projections. Estimates made of workload, resource consumption, or response time as opposed to measurements. Projections are usually simple scalars from known (measured) values. Estimates in general can be wild guesses or output from a simulation or analytic model. They are not measurements.

procedure. A series of steps or activities required to perform a process.

program. A computer-based application that supports the work to be done in an activity. Program activities reference executable programs using the logical names associated with the programs.

R

RAM. Random access memory. The location where data is stored while being processed by the CPU.

report. Formatted text and graphics, usually generated by a system.

report layout. The design and specifications for the format of a printed report. See also *screen layout*.

repository. An organized, shared collection of data or information that supports business process re-engineering, application development, or business or systems management. It is usually automated and is implemented as a database.

resource consumption. The execution time of a particular resource. This is generally given for a particular unit of work. Examples would be: number of LS CPU seconds per LSFR, ms of staging disk busy time per object stored, and so on. These values are either estimated from known CPU path lengths, or more often, calculated from measured percent utilization during a known workload. For example, if a CPU is 50% busy doing 100 LSFRs per second, then the resource consumption of an average LSFR is 5 ms.

response time. The elapsed time to execute a unit of work. The scope of a response time may be time in a particular component, or from the time the end user presses Enter until the keyboard is available for further input. Response time includes the resource consumption time plus delays, or wait times, resulting from resource contention/queueing. Response time objectives or requirements are the maximum times that can be tolerated.

Response time estimates or predictions are guesses as to what response times can be delivered by a particular system under a particular workload. Response time measurements are actual response times obtained from a particular system executing a particular workload. Examples are: seconds required to store a document, ms of elapsed time on the Library Server, and so on.

REXX. Restructured extended executor language. A procedures language.

S

SB. See *Service Broker Architecture*.

SBM. See *Service Broker Manager*.

screen layout. The design and specifications of the image that the user sees on the screen of a system. See also *report layout*.

server. A system that is designed to share data with client applications. Servers and clients are often connected via a network, or may be simply located on the same computer.

service. A collection of related features that respond to requests for specific activities or that yield information. The services are accessed through a consistent and published interface of the service that encapsulates its implementation.

In the Service Broker Architecture, a service DLL contains service functions which interface with the base product. A service function receives the user data from the Service Broker Manager and calls the appropriate product APIs to perform the work. The results are returned via the Service Broker Manager to the service requester, and then back to the user application.

Service Broker Architecture. The Service Broker Architecture is designed to allow users of workflow systems or other applications to work with multiple tools in multiple interactions without the need to reload the tool each time or perform multiple logons to server sessions. The aim is to allow all required sessions and tools to be available during the work session without the users needing to be aware of the application execution or logic.

Service Broker Manager. A FlowMark component that controls the operation of service broker, sessions. This includes the interaction between service requester and services, between service broker and services, and also the intialization of the service brokers and services.

service requester. A service requester is the interface to the user application. The user application calls the service requester APIs to request the base product to perform some work. The service requester formats the user data and issues a request to the service broker, which invokes the appropriate service function via service threads.

service thread. One or more service threads are started for each service. Each thread receives information from the service requester to call the respective service function. When the function has been called, the thread returns information to the service requester.

skill. An ability, proficiency, expertness of a person that comes from training, practice, and experience.

subject expert. A specialist in a particular area of expertise, such as workflow or document management.

symbol. A graphical representation of an object or thing, which may be abstract in nature; for example, a line with an arrow is the symbol for a data or information flow.

system. A set of processes with a common aim that act on data or information using input and producing output. Usually used in the context of *information system* or *data processing system*.

system development. The design, code, test, implementation, and maintenance of an information system. Can also denote a business function, which performs systems development.

T

TCP/IP. Transmission control protocol/Internet protocol. A commonly used protocol for various network environments, such as Internet, Host communications, and LANs. FlowMark uses either TCP/IP or APPC. See also *APPC*.

throughput. See *workload*. While used interchangeably with workload, throughput often refers to measurements of workload during a benchmark.

TIFF. Tag Image File Format.

TOC. Table Of Contents.

top-level process. In FlowMark, a process that is started from a user′s process list or from an application program.

trigger. An event or condition that start or ends an activity, a process, or process path.

U

unit of measure. A standard dimension, extent, or quantity, such as days, hours, or minutes, dollars of cents, or meters or centimeters. A unit of measure is used for measurement purposes.

V

VHLPI. VisualInfo high-level programming interface. The service broker for VisualInfo. It can be used to integrate FlowMark with VisualInfo for document management connectivity and services.

VisualBasic. A programming language under MS Windows.

VisualInfo. An IBM product for document and image management.

W

WAN. See *wide area network*.

WF. See *workflow*.

wide area network (WAN). A system which connects a client and a server in different locations. Transfer rate is slower than a LAN, usually thousands of bits per second.

workflow. A sequence of activities (units of work). A movement of units of work through a process. In process-based applications, it can be part of a control flow.

workload. The amount of work done per unit of time, or the number of users that will be using the system concurrently (and the amount of work they can each do per unit of time). Examples are: documents processed per hour, LSFRs per hour, Folder Manager APIs per hour. Examples from outside of the VI application would include: keystrokes per second, transactions per second, SQL calls per second, checks written per day, visits to an automated teller per hour, and so on.

workload capture. Tracing the work that an end user is doing. This ranges from getting general end-user statements about “documents processed per day” to the actual work of processing each document. It is also used to find out what the productivity of a single user can be (that is the number of units of work that a single user can impose per unit of time), and can be invaluable in finding bugs in an application that cause unexpected extra work.

Index

ITSO Redbook Evaluation

Image and Workflow Library: Capacity Planning and Performance Tuning
SG24-4974-01

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and Fax it to: USA International Access Code + 1 914 432 8264 or:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
__Customer __Business Partner __Solution Developer __IBM employee
__None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____________

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:
____

____

____

____

What other redbooks would you like to see published?
____

____

____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**
____

____

____

____

____

SG24-4974-01
Printed in the U.S.A.

Image and Workflow Library: Capacity Planning and Performance Tuning
for VisualInfo and Digital Library Servers

SG24-4974-01