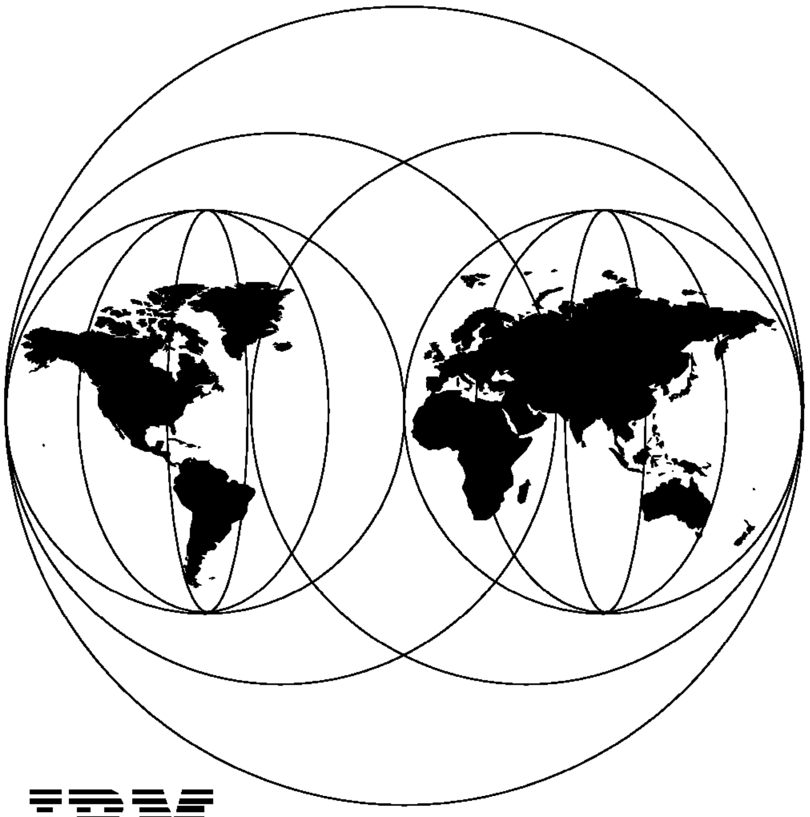
SG24-4836-00

CICS for AIX and CICS System Manager for AIX: Experiences with a Large RISC System/6000 Scalable POWERParallel (SP) System

October 1996



International Technical Support Organization
San Jose Center

International Technical Support Organization

SG24-4836-00

CICS for AIX and CICS System Manager for AIX: Experiences with a Large RISC System/6000 Scalable POWERParallel (SP) System

October 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, “Special Notices” on page 49.

First Edition (October 1996)

This edition applies to Version 4 of Transaction Server for AIX for use with AIX 4.1.4.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. 471 Building 80-E2
650 Harry Road
San Jose, California 95120-6099

Contents

Figures

Tables

Preface

This book documents the construction of a large configuration of Transaction Server for AIX software that was built across 144 nodes of a RISC System/6000 Scalable POWERParallel (SP) system at the Maui High Performance Computing Center in Maui, Hawaii. The configuration was built specifically to demonstrate to customers attending the IBM Asia Pacific “Energize Your Enterprise” software conference that large, CICS for AIX configurations can be built, run, and managed with CICS Systems Manager for AIX (CICS SM).

This book describes the demonstration, the process used to build the 144-node system, and the results obtained. The CICS SM workload management component is covered in detail. A number of recommendations are provided for those people who are implementing CICS for AIX and CICS SM in a RISC System/6000 SP system.

This book is written for:

Customers	To demonstrate that large configurations of Transaction Server for AIX have been built and throughput in terms of thousands of transactions per second can be sustained and managed
Systems Engineers	To provide a practical example of how one workload-managed CICS for AIX based transaction processing system was set up. A number of recommendations for the implementation of such systems are also provided to optimize the performance obtained.

A working knowledge of the following is assumed:

- CICS for AIX
- CICS SM
- RISC System/6000 SP

(64 pages)

How This Redbook Is Organized

This book is organized as follows:

- “Introduction”

  This chapter briefly describes the reasons for building the demonstration and writing this book.

- “The Objectives”

  This chapter describes the objectives of building the demonstration.

- “Overview of the RISC System/6000 SP”

  This chapter provides an overview of a RISC System/6000 SP system. It is intended to provide a common understanding of the environment facilities that the system supports.

- “The Environment”

This chapter provides a description of the hardware and software used in this configuration. For the hardware, the configuration of the nodes is described. For the software, the code versions used are documented.

- "The Application"

  This chapter provides a brief description of the DebitCredit application that was used in the demonstration to generate processor and disk load in the CICS for AIX servers. The chapter also explains how the standard version of DebitCredit was modified for use in this demonstration.

- "CICS SM Workload Management Configuration"

  This chapter describes the CICS SM workload management configuration that was used to control the routing of the simulated workload.

- "Maximizing Use of the High-Performance Switch"

  Full exploitation of the high performance switch in a RISC System/6000 SP system is crucial to obtaining maximum benefit from the system. This chapter examines the communication between the components of the configuration and explains how traffic can be directed to run over the switch.

- "Planning and Preparation"

  This chapter describes the early work that was performed to prepare for construction of the demonstration on the RISC System/6000 SP system.

- "Configuring the RISC System/6000 SP"

  This chapter describes how the configuration was built on the RISC System/6000 SP system. The 144-node system was built in three stages. The stages and the steps within them are covered in detail.

- "Measuring and Demonstrating"

  This chapter describes the process that was used to obtain the demonstration results. We also discuss the configuration that was demonstrated to customers.

- "Recommendations"

  As a result of building the configuration a number of recommendations are made across each of the major areas that the demonstration covered: CICS for AIX, CICS SM, the RISC System/6000 SP system, and DCE. This chapter details those recommendations.

- Appendix A, "Sample Data-Dependent Routing Program"

  This appendix contains a copy of the CICS SM data-dependent routing exit used in the demonstration.

- Appendix B, "Software Installation"

  This appendix contains a list of the scripts used to perform software installation, and a list of the software installed on each node type.

- Appendix C, "Reconfiguring DCE"

  This appendix presents the work required to move from a full DCE configuration to a DCE lite configuration.

The Team That Wrote This Redbook

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Guido De Simoni is a Project Leader at the International Technical Support Organization, San Jose Center. He writes extensively and teaches IBM classes worldwide on all areas of distributed OLTP. Before joining the ITSO three years ago, Guido worked in the AIX Competency Center in Italy as a Connectivity Advisor.

Tim Dunn is a Performance Analyst at the Hursley Laboratory in England. He has eight years of experience in the performance tuning of CICS for AIX and CICS for MVS/ESA systems. He has also worked extensively with CICS System Manager for AIX, focusing on the workload management component.

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

Phil Conway, IBM Asia Pacific

for his work in negotiating the necessary time and resources with the Maui High Performance Computing Center.

Keith Edwards, IBM US

for his work in helping to build and run the configuration

Jeff Tennenbaum, IBM US

for his work in helping to build the demonstration as well as his problem determination skills

Glenn Wightwick, IBM Australia

for his work in the early planning stages when we were developing the installation scripts, as well as his contribution of “Overview of the RISC System/6000 SP” on page 5 of this book.

All of the people at the Maui High Performance Computing Center involved in making the Transaction Server for AIX demonstration happen, but especially:

Chris Chisolm
Mike Ida
Dave Jackson
Margaret Lewis
Candice Shirley
Lynette Wissenger

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 -- Heading ′EVAL′ unknown -- to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Home Pages at the following URLs:

For Internet users	`http://www.redbooks.ibm.com`
For IBM Intranet users	`http://w3.itso.ibm.com/redbooks`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Introduction

In July 1996 IBM Asia Pacific held a software conference entitled "Energize Your Enterprise." More than 150 representatives of IBM′s leading customers from Asia Pacific attended the conference in Maui, Hawaii. During the conference a number of demonstrations were given at the Maui High Performance Computing Center (MHPCC) to illustrate how customers can use IBM software to implement effective and robust business solutions that will enable their company IT infrastructure to grow in step with their business needs. The demonstrations included one for Transaction Server for AIX to illustrate CICS for AIX and CICS System Manager for AIX (CICS SM) in use on a RISC System/6000 Scalable POWERParallel (SP) system.

The MHPCC contains one of the world′s large parallel computers, a 384-node RISC System/6000 SP system. The MHPCC was established on Maui to support the United States Air Force Maui Space Surveillance Site. The primary customer group at the MHPCC consists of U.S. Department of Defense researchers. The MHPCC also serves other government missions and business and academic customers and supports a wide variety of scientific, research, and commercial projects. For more information about the MHPCC, see its home page on the World Wide Web at *http://www.mhpcc.edu*.

The Transaction Server for AIX demonstration was built specifically for customers attending the conference. In building the demonstration, significant experience was gained in the development of such configurations. To disseminate that information, the demonstration work is documented in this book.

In this book we show the major steps taken in order to build the configuration. We hope that many of the principles we followed will be of use to others who are responsible for implementing similar systems.

This book does not illustrate all of the individual steps undertaken and commands issued to build the system. It provides an overview of the installation and configuration process, which we believe is the more important aspect. A list of commands would not be meaningful for other installations because the system requirements would most likely differ from ours. Understanding the sequence of operations and the principles on which we based our configuration is of value, however.

The scripts and programs that were used to build the demonstration are available as a SupportPac. The SupportPac is intended to be used in conjunction with an IBM Account Team or Systems Engineer and as such can be ordered only through IBM internal channels. Details of the SupportPac contents are available on the World Wide Web at
http://www.hursley.ibm.com/cics/txppacs/txpsum.html

Some of the more involved aspects of the demonstration, such as CICS SM Workload Management and its use of data-dependent routing, are covered in some level of detail to assist you in building a similar system.

Probably the greatest strength of the book is that it documents an actual implementation. It is written from experience for a user who understands the concepts of CICS for AIX, CICS SM, Distributed Computing Environment (DCE), and a RISC System/6000 SP. Because the book assumes that you are familiar with these products, little product explanation is provided. A novice in the use of these products should read *Addressing OLTP Solutions with CICS: The Transaction Server for AIX*, SG24-4752-00, to gain the necessary background information.

The book ends with a number of recommendations to assist you in your implementation of a CICS for AIX and CICS SM configuration on a RISC System/6000 SP.

The Objectives

The objectives of the project was to build, measure, and demonstrate a configuration consisting of a large number of CICS for AIX servers using CICS SM Workload Management to distribute the incoming work from a large number of simulated users over the pool of available servers. User activity was simulated through the use of multiple instances of a program that used the CICS external call interface (ECI). The entire demonstration, both clients and servers, was run on a RISC System/6000 SP system.

Such a system was constructed to show that:

- CICS for AIX works well on a RISC System/6000 SP. The RISC System/6000 SP is an emerging hardware platform on which CICS for AIX is capable of running. The RISC System/6000 SP enabled a very large number of CICS for AIX servers to be built within the one machine. There were no inherent bottlenecks, no control region to act as a constraining influence. It was possible to direct the traffic between CICS clients and servers so that it made use of the high bandwidth connection between the nodes in a RISC System/6000 SP.
- CICS for AIX enables large configurations to be built. By running a medium-weight transaction, we were able to demonstrate transaction throughput of thousands of transactions per second, thus proving that CICS for AIX can process high-volume throughput and is well ahead of today′s production transaction rates.
- A large CICS for AIX configuration can be managed. With large systems, management becomes increasingly important, as they require an increasing number of people to administer them.

It is also important to understand what we did not intend to demonstrate; namely:

- Use of a parallel database. Our demonstration was of CICS for AIX and CICS SM, not database technology.
- A complex application. We chose for the demonstration a version of the DebitCredit application, a well-known application or workload that is widely understood. For more information about the implementation of the DebitCredit application, see “CICS SM Workload Management Configuration” on page 13. We recognize that a DebitCredit application is not particularly typical of a customer environment. However, the actual application was not an issue in this demonstration; the ability to perform workload management over a large number of servers was the issue.

Overview of the RISC System/6000 SP

The RISC System/6000 SP computer is a general-purpose, scalable, parallel system based on a distributed memory, message-passing architecture. The system consists of a number of processor nodes connected by means of a high-performance switch. The processor nodes are based on generic RISC System/6000 processors using POWER2 microprocessors and/or PowerPC symmetric multiprocessors (SMPs). The RISC System/6000 SP system can scale from a single processor up to a system containing 512 processor nodes. Each processor node has its own memory, disk, and peripherals such as network interfaces, tape drives, or external disk and executes its own copy of the AIX operating system.

Nodes for the RISC System/6000 SP are installed in frames. A frame can contain 16 thin nodes, 8 wide nodes, or four 4 nodes. Thin nodes are based on RISC System/6000 390 and 39H systems, wide nodes are based on RISC System/6000 590 and 591 systems, and high nodes are based on RISC System/6000 R40, J40, or G40 systems. The various thin, wide, and high nodes can support different amounts of memory, level 2 cache, disk, and expansion slots.

Within a RISC System/6000 SP system, nodes are connected to a RISC System/6000 control workstation through an Ethernet network. Such a network may consist of a single Ethernet or multiple Ethernet networks (depending on the size of the RISC System/6000 SP system) and is generally referred to as the SP Ethernet. The SP Ethernet is used for system management purposes such as node installation. Each frame is also connected to the control workstation through an RS-232 link, which provides hardware monitoring and control functions.

In addition to the SP Ethernet, nodes in a RISC System/6000 SP are also connected through the high-performance switch. The switch provides a low-latency, high-bandwidth interconnection network between nodes. Each node connects to the switch through a microchannel adapter. Processes running on a node can communicate with other processes on other nodes through standard TCP/IP protocols or by utilizing a lower overhead message-passing protocol. Because many software subsystems (such as DCE) are based on TCP/IP sockets, existing middleware layers can automatically take advantage of the increased bandwidth and lower latency of the switch network when running on a RISC System/6000 SP system. Figure 1 on page 6 illustrates the major components of a RISC System/6000 SP and the way in which they are interconnected.

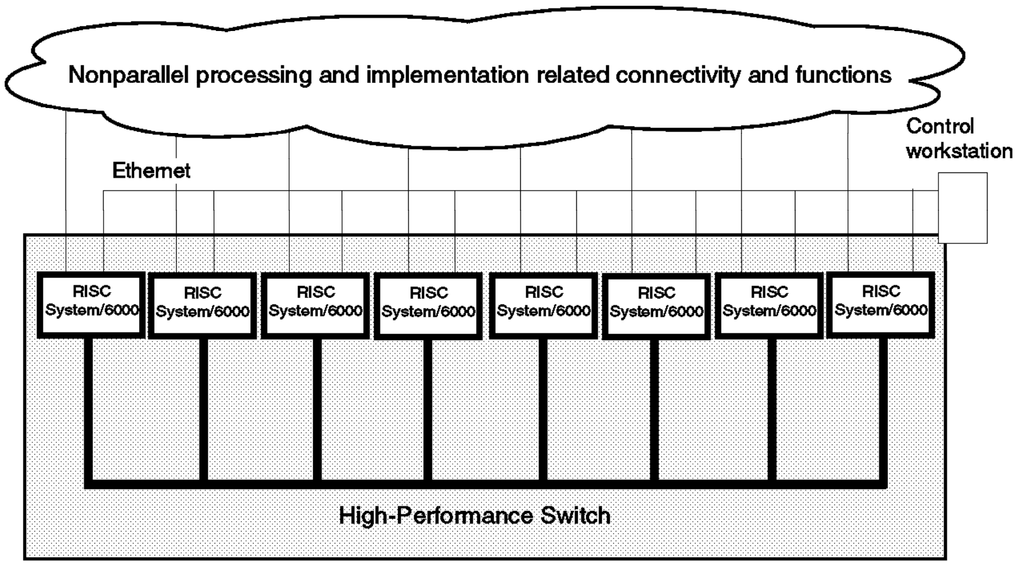


Figure 1. RISC System/6000 SP: Major Components

To communicate with users or applications external to the RISC System/6000 SP, additional communication adapters are typically installed in various nodes of the RISC System/6000 SP system. These nodes may be specific gateway nodes with asynchronous transfer mode (ATM), Fiber Distributed Data Interface (FDDI), token-ring, or Ethernet interfaces. In some SP systems, all nodes may contain external connections.

Given that a typical RISC System/6000 SP system may have three or more TCP/IP interfaces (SP Ethernet, high-performance switch, and external interfaces), it is clearly important that TCP/IP traffic be routed over the appropriate network. This routing is usually achieved by ensuring that each subsystem selects the appropriate network interface.

Because the RISC System/6000 SP provides a broad range of processing and communications capability, it can support a wide range of applications, including:

- Parallel message-passing jobs
- Distributed client-server systems
- Parallel databases
- Transaction processing

The Environment

The configuration was built and demonstrated on a part of the large RISC System/6000 SP processor installed at the MHPCC.

Initial machine time was obtained on two stand-alone RISC System/6000 machines; subsequent time was obtained on 144 nodes of the 384-node machine. The initial machines were a RISC System/6000 Model 250 and a RISC System/6000 Model 370. These machines were used to develop and test the installation and workload configuration scripts without any impact on the RISC System/6000 SP, which is a production system. Once we began machine time on the RISC System/6000 SP, we were able to build and run the demonstration on our allocated nodes, while other MHPCC users continued to run their work on the other nodes, thus illustrating the power and flexibility of the RISC System/6000 SP environment.

Hardware

All 144 nodes that we used on the RISC System/6000 SP consisted of POWER2 processors with 128 MB of memory and two 1 GB disks. Each of the nodes was equivalent to a RISC System/6000 Model 390 processor.

Software

The following levels of software were used:

- AIX Version 4.1.4.
- Transaction Server for AIX Version 4. The following components were used:
  - CICS for AIX Version 2.1.1.
  - Encina Version 2.2.
  - CICS System Manager for AIX Version 1.0.
- DB2 for AIX Version 2.1.1
- AIX DCE Version 2.1

The Application

A key aim in building the demonstration was to show management of work across the available CICS for AIX servers. To do that we needed an application or workload. We used a version of DebitCredit, an application that is not necessarily representative of a particular industry sector or type of installation but well understood.

The terms *application* and *workload* are used interchangeably in the industry. We use the term *application* in this book where it helps to avoid confusion with CICS SM Workload Management.

DebitCredit

DebitCredit represents a simple banking application in which customers visit branches of a bank and credit or debit sums of money to their account. Each branch of the bank has 10 tellers associated with it. Balances are maintained for the account holder, the teller, and the branch. In addition, an audit history is maintained of debit/credit activity. The balances are maintained in the following tables:

- Account table—contains each customer′s balance
- Teller table—contains each teller′s balance
- Branch table—contains the balance for each branch
- History table—the audit trail.

There is a single transaction type in this application, with all transactions performing the same operations, namely, reading and updating the account, teller, and branch tables and writing a record to the history table.

CICS ECI Version

Typically, DebitCredit is implemented as a screen-based application. Terminal simulators that drive the application simulate the activity of users entering data on the screen interface. The program that performs the logic of the application is unaware that the data is supplied by a terminal simulator, and not a user.

Today, CICS applications are increasingly being written to make use of the CICS ECI, an interface that enables applications to be designed with the business logic on the CICS server and the presentation logic on the workstation-based CICS Clients. The CICS ECI enables a non-CICS client application to call a CICS application, synchronously or asynchronously, as a subroutine. The client application communicates with the CICS server program through the CICS communication area (COMMAREA). The program typically populates the COMMAREA with data accessed from files or databases, and the data is then returned to the client for manipulation or display. A great benefit of the CICS ECI support is that any program on the workstation can access facilities on the CICS for AIX server.

To reflect the increasingly widespread use of the CICS ECI, DebitCredit was rewritten to produce a CICS ECI-based version. This involved writing a CICS ECI application in addition to producing a modified version of the program that ran in the CICS for AIX region. The two programs communicated through a CICS

COMMAREA. In the more traditional implementation, communication to and from the CICS application running in the CICS for AIX region would be through the use of CICS basic mapping support (BMS) send and receive commands.

The CICS ECI application was written to issue repeated requests to a server. It was written as a multithreaded application to enable multiple users to be simulated with a single program. The multithreaded client uses less memory than multiple single-threaded clients. For a limited amount of memory we were able to simulate more users than would have otherwise been possible.

Figure 2 illustrates the application.

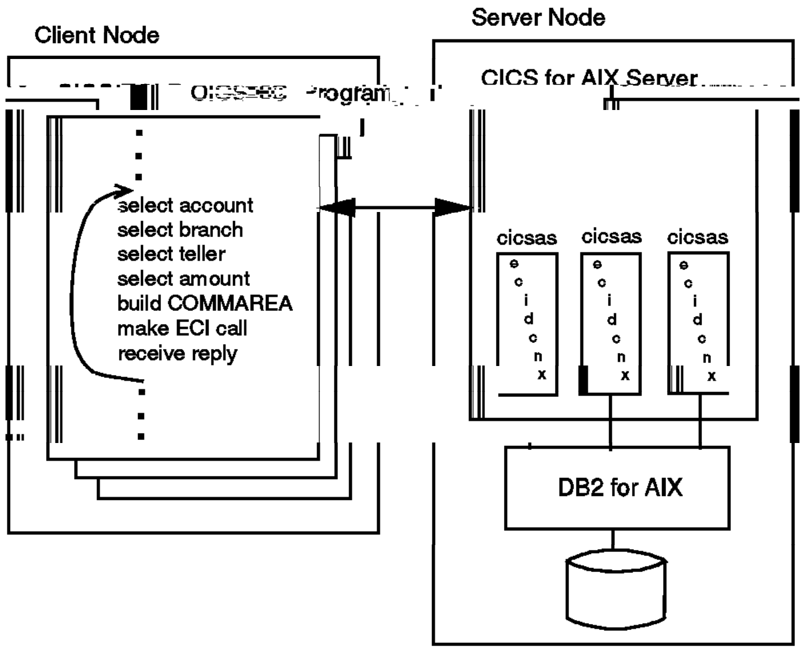


Figure 2. Overview of the Application

The workload processing commenced with the CICS ECI application selecting an account, teller, branch number, and amount at random. These details were then copied to the COMMAREA as was the workload partition name. See "Two Level Routing" on page 14 for more details of workload partitions. The partition name was specified in the ECI_PARTITION environment variable, which was set in the CICS ECI program′s environment before invocation. A synchronous CICS ECI call was then made to a CICS application program running in a CICS for AIX region. On invocation the CICS application in the CICS for AIX region read the CICS COMMAREA; accessed and updated the relevant account, teller, and branch rows held within the DB2 for AIX database; and finally passed the results back to the CICS ECI program through the CICS COMMAREA.

When a CICS ECI program issues a program call to a CICS for AIX region, by default, the called program will be run under a transaction identifier of CPMI. This is the transaction name of the CICS mirror transaction. Allowing the name to default in this manner makes it difficult to tell whether a transaction with a name of CPMI is a workload transaction or some other request because all other requests coming in from an ECI application as well as work from other connected CICS servers also use this same transaction identifier. To distinguish those transactions associated with the workload from all other CPMI-related activity, the transaction identifier under which the workload ran was changed to BECI. Two changes were performed:

- In the CICS ECI program, the eci_transid field, which is used to describe the ECI call in the ECI_PARMS structure, was set to a value of BECI. When the CICS ECI call is received by a CICS for AIX, region it will be run under this transaction identifier.
- Within the CICS for AIX region, a transaction of BECI was defined with a first program of DFHMIRS, the same program that is invoked with CPMI.

With these changes, DFHMIRS was still executed when the CICS for AIX server received an ECI request, but it was now possible to attribute that portion of its usage that was due to the workload.

CICS SM Workload Management Configuration

Although it would have been possible to have statically routed all of the work generated by the CICS ECI programs across the 100 CICS for AIX regions, it would have been a highly rigid process. If we had subsequently wanted more or fewer CICS for AIX regions, it would have been necessary to manually rebalance the work over the changed number of servers. Such rebalancing makes the process of static routing inflexible and labor intensive. If a CICS for AIX region had failed, the users sending work to that region would have been unable to work until either the region was recovered or they had been redirected to another working region.

Workload management as provided by CICS SM Workload Management provides a means of dynamically distributing work over a pool of available servers with the minimum of intervention. If a CICS for AIX region should fail, CICS SM Workload Management detects the failure. The work that is routed to that now failed region will be routed to the remaining regions capable of supporting that type of work. This rebalancing takes place without manual intervention, thus helping to provide significantly improved application availability. Similarly it is possible to add regions and for work to be routed to all regions, including the new addition.

For this demonstration, we performed the workload management of CICS ECI requests from 10,000 simulated users. The clients were spread over 42 nodes of the RISC System/6000 SP. For the workload management to function we needed a workload manager implementation like that shown in Figure 3 on page 14.

The workload management component made routing decisions on the basis of data that was available to it about the CICS regions and resources. That data was held in two types of cache, global and local. Both caches were similar in structure. On plex activation, details of the newly active configuration were passed from the SMappl to the workload management application (WLMAppl), and through the WLMAppl to the SMappl interface (WSI). The WLMAppl added the information that was passed to it to the global cache. There was one global cache on the management node, and a local cache on each of the client nodes. A local cache holds a subset of the data that is held in the global cache. The information held in a particular local cache reflects the requests made by the CICS ECI programs running on that node. In this demonstration, all CICS ECI programs requested the same CICS application program, ecidcnx. In a more typical environment, there would be many requested programs, and not all local caches would necessarily contain the same information because not all users would necessarily be running the same applications.

In addition to the local cache on each node, there was also a workload cache manager (WCM) and a workload management client. The WCM is essentially responsible for managing the local cache. The workload management client is responsible for making the routing decisions. Each WCM communicated with the WLMAppl, using TCP/IP sockets to transfer data from the global to the local cache. When the first CICS ECI request was issued on each client node the workload management client tried to make a routing decision based on the information available in the local cache. As there was no entry in the local cache for that program, a request was made to the WCM to obtain that data. As a result the WCM issued a data request to the WLMAppl. The WLMAppl examined its global cache and passed the required data back to the WCM to store in the local cache. Had the data not been present in the global cache, the WLMAppl would have forwarded a request to the SMAppl.

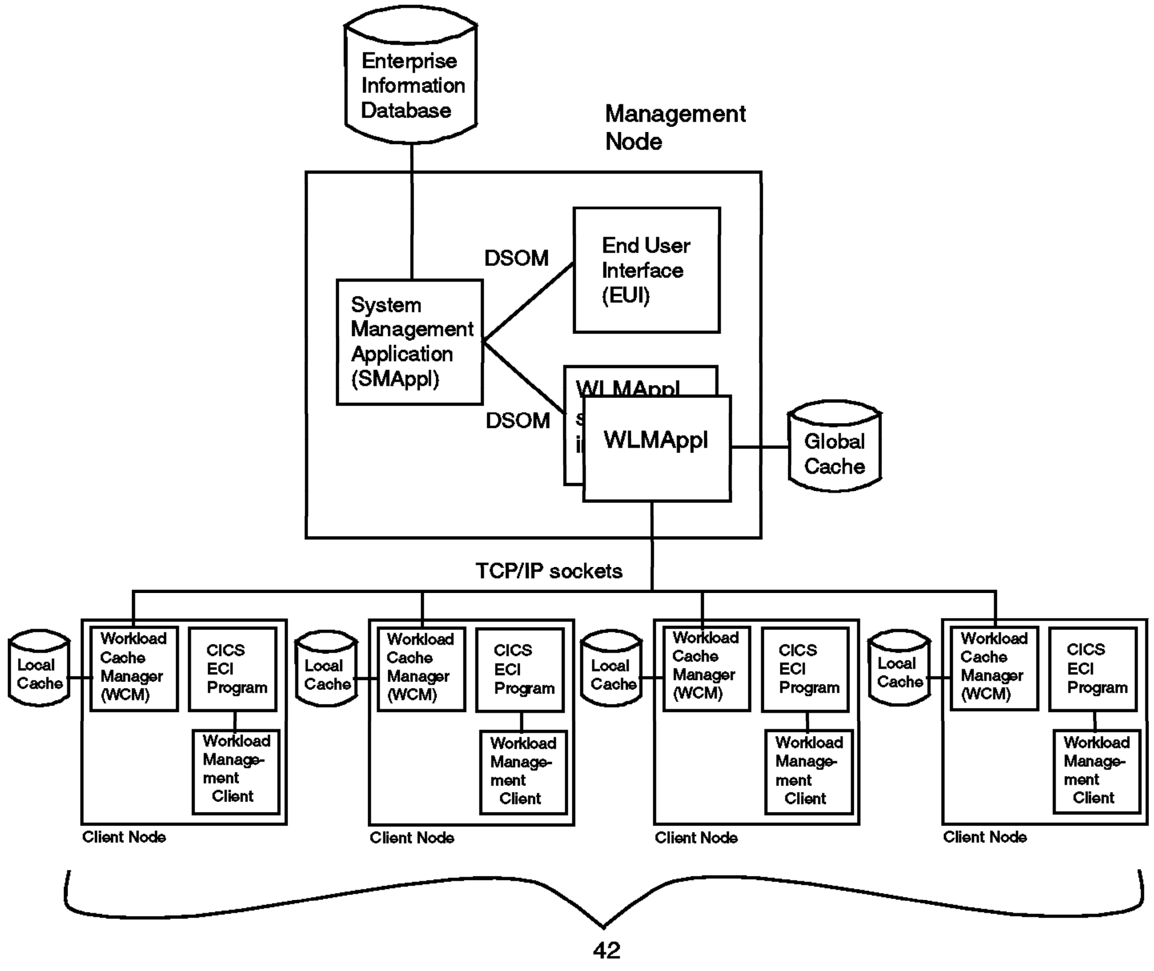

Enterprise Information Database
Management Node
DSOM
End User Interface (EUI)
System Management Application (SMAppl)
DSOM
WLMAppl
Global Cache
TCP/IP sockets
Local Cache
Workload Cache Manager (WCM)
CICS ECI Program
Workload Management Client
Client Node
42

Figure 3. Workload Management Components and Interconnection

Two Level Routing

Attempting to balance work evenly over a pool of 100 servers is a very difficult and potentially impossible task, especially with a large population of users (we were simulating 10,000 users), because the workload management client in each node makes routing decisions based solely on the information available to the node on which it runs. There is no global view of the health of all of the servers. In such a situation, it is easy for the workload management clients on two separate nodes to route a significant number of their users to the same small group of servers and so overload the servers. As a result the clients see increasingly poor response times. In reaction, future requests will be routed elsewhere, leaving this previously overloaded group of servers underutilized, possibly empty. The same effect is likely to be seen for the newly selected servers. This hunting or ripple results from having a large pool of servers with a large population of users who are free to move between servers. A much improved method of routing is to form groups of servers or partitions. With this approach a partition is first selected, and then within the partition the most suitable server is chosen to actually route the request to. This approach restricts the ability of clients to wander yet still retains flexibility as a number of regions are available to route work to within each partition. This is the approach that is possible with the data-dependent routing facility that CICS SM offers.

We implemented data-dependent routing to control the distribution of work over the pool of 100 servers. We formed 13 partitions and allocated each server to one and only one partition. We had 12 partitions of eight servers, and 1 partition of four servers. The partitions were implemented by performing a workload separation on the CICS ECI program that formed our workload. Figure 4 provides an overview of CICS SM data-dependent routing.

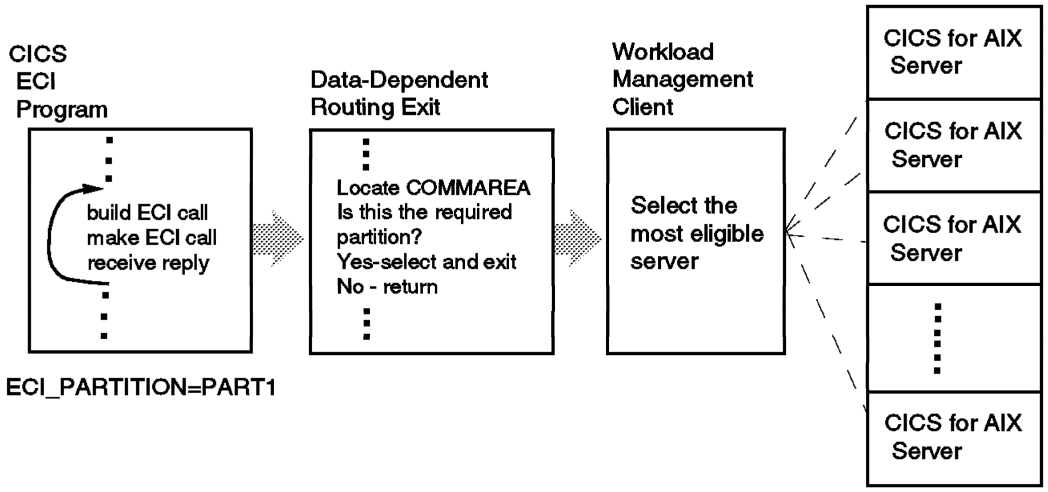


Figure 4. CICS SM Data-Dependent Routing

Here is how the data-dependent routing worked:

- On invocation each CICS ECI program was passed the name of a partition that it should use through the ECI_PARTITION environment variable. We allocated the partition name on a round robin basis. For the total list of CICS ECI programs that was run, we looped around assigning each a partition number from 1 to 13, then 1 to 13, and so on until each ECI program was assigned a partition number. The partition name could just as easily have been derived from the result of running a hashing algorithm on an account number or by selecting a geography code.
- The CICS ECI program copied the partition name to the COMMAREA that was used in the CICS ECI call.
- The CICS ECI call was made.
- CICS SM Workload Management intercepted the call, recognized that the call was for a separated resource, and invoked the data-dependent routing exit for each of the partitions in which the program was available, or until the return code from the exit indicated that no more calls were required. In other words, the partition name with which the exit was called matched the partition name specified in the CICS ECI program′s COMMAREA.
- Once a partition was chosen, the workload management client applied the workload management algorithm to the servers in the selected partition and chose the most suitable server.
- The CICS ECI call was then issued to the selected server.

A copy of the data-dependent routing exit that we used is supplied in Appendix A, “Sample Data-Dependent Routing Program” on page 41. Another, more comprehensive, sample can be found in the /usr/lpp/cicssm/samples/wuexmp.c file.

Maximizing Use of the High-Performance Switch

To maximize the value of running on an RISC System/6000 SP processor it is necessary to send as much traffic over the switch as possible. The switch has a bidirectional bandwidth of approximately 80 MB per second (40 MB in each direction concurrently), whereas the SP Ethernet has a bidirectional bandwidth of only 1.2 MB per second.

In deciding which traffic was eligible to be routed over the switch, we had to first examine all intercomponent communication that could take place in the demonstration. From that list we then focused on those components that had intermachine communication and for which it would be possible to control the communication route.

Intercomponent Communication

The interactions that took place between the major components in the demonstration are listed below. In a more complex environment, where there is perhaps external communication outside the RISC System/6000 SP, or where there is CICS for AIX region to CICS for AIX region communication, for example, the list would be more extensive.

- DCE cell directory services (CDS) and security
- CICS client to CICS server
- CICS server to CICS client (only when the servers becomes busy)
- CICS server to Encina Structured File Server (SFS)
- CICS server to DB2 for AIX
- CICS SM SMappl to CICS SM resource controller
- CICS SM WLMappl to CICS SM WCM
- CICS ECI client to CICS SM workload management client
- Network file system (NFS) file access by configuration scripts or programs, for example.

Having identified the interactions, we were then able to look at the communication method used and see whether it was feasible to route it over the switch. Table 1 summarizes the characteristics of these different types of communication.

Table 1 (Page 1 of 2). Summary of Component Intercommunication Characteristics

Interaction	Communication Method	Transport Mechanism	Frequency of Traffic (1)
DCE CDS and security	DCE remote procedure call (RPC)	SP Ethernet or switch	L (2)
CICS client to CICS server	DCE RPC	SP Ethernet or switch	H
CICS server to CICS client (3)	DCE RPC	Switch	L
CICS server to Encina SFS	Pipes or shared memory	Intramachine	M
CICS Server to DB2 for AIX	Program call	Intramachine	H
CICS SM SMappl to CICS SM resource controller	SOM	SP Ethernet or switch	M
CICS SM WLMappll to CICS SM WCM	TCP/IP sockets	SP Ethernet or switch	L
CICS ECI application to workload management client	Program call	Intramachine	H

Table 1 (Page 2 of 2). Summary of Component Intercommunication Characteristics

Interaction	Communication Method	Transport Mechanism	Frequency of Traffic (1)
NFS file access	DCE RPC	SP Ethernet or switch	M

Note: (1) *H* denotes high frequency of traffic; *M*, medium frequency; and *L*, low frequency. (2) This communication is marked as *L* because the traffic occurs only at startup and does not run for the entire workload. The DCE server can be heavily loaded at startup, however. (3) This type of traffic takes place only when the server is heavily loaded and the CICS transaction scheduler goes into asynchronous scheduling mode.

In Table 1 on page 17, the interaction of most interest to us was the CICS client to CICS server communication. The DCE CDS and security and the NFS file access interactions were also of interest but less important. In section "Enabling Communication over the High-Performance Switch" we show how we configured the traffic to use the switch rather than the SP Ethernet.

Had either the Encina SFS or DB2 for AIX system been located on a separate node from the CICS for AIX region on the RISC System/6000 SP, they may well have deserved closer examination. But as the communication between them and the CICS for AIX region was intramachine, use of the SP Ethernet rather than the switch was not an issue in this case.

Enabling Communication over the High-Performance Switch

To ensure that components communicated over the switch, we generally had to use the host name that corresponded to the switch address of the node (the exception being DCE). Below we examine the implications of that approach for each communication in Table 1 on page 17 that was eligible to use the switch.

DCE Services

In configuring DCE on the DCE server and client nodes we used the host names corresponding to the SP Ethernet addresses of the nodes, although we still managed to get the traffic to flow over the switch by using the DCE environment variable, RPC_UNSUPPORTED_NETIFS, which was set to a list of all nonswitch interfaces. The presence of this environment variable causes those servers that register their IP addresses in the DCE CDS to register only addresses for interfaces that are not listed in RPC_UNSUPPORTED_NETIFS. On the nodes on which we were working, there were up to four network interfaces:

- en0, an SP Ethernet connection
- en1, an SP Ethernet connection
- fi0, an ATM connection
- css0, the switch connection

We used RPC_UNSUPPORTED_NETIFS=en0,en1,fi0, which prohibited traffic over all interfaces but the switch. When a client sought the address of a server by issuing a request to the DCE CDS clerk, the address returned was the switch address of the server. The traffic to that server thus traveled over the switch.

It was *essential* to have RPC_UNSUPPORTED_NETIFS present in the environment in which DCE was configured in order for all communication to travel over the switch. Failure to have RPC_UNSUPPORTED_NETIFS present in this environment meant that subsequent use of RPC_UNSUPPORTED_NETIFS had no effect. When DCE was configured without RPC_UNSUPPORTED_NETIFS present, we found that traffic flowed over both the SP Ethernet and the switch. We had to reconfigure the DCE cell, ensuring that RPC_UNSUPPORTED_NETIFS was present.

The easiest way of ensuring that RPC_UNSUPPORTED_NETIFS was present in the environment of all correct processes was to place it in the /etc/environment file on each of the nodes.

CICS Client to CICS Server

As DCE RPC was the communication method for CICS client to CICS server communication, the transport mechanism over which the communication flowed was controlled by how DCE RPC was configured. As we discuss in "DCE Services" on page 18, this was achieved through the use of the DCE environment variable, RPC_UNSUPPORTED_NETIFS. The list of CICS-related servers that register addresses with the DCE CDS include:

- cicsrl
- cicsterm (when asynchronous transaction scheduling is active)
- CICS ECI program (when asynchronous transaction scheduling is active)
- Encina SFS

By ensuring that RPC_UNSUPPORTED_NETIFS was present in the /etc/environment file, no further action was required to ensure that the CICS-related DCE traffic traveled over the switch.

CICS SM SMappl to Resource Controller

We did not configure the CICS SM SMappl to resource controller communication to travel over the switch. The volume of messages was low in our environment. In a more dynamic environment where CICS for AIX regions are started and stopped more frequently or resources being enabled or disabled, we would recommend configuring this traffic to travel over the switch. Use the switch address of the nodes in the CICS SM configuration file (/var/cicssm/repos/cicssm.config).

CICS SM WLMAppl to WCM

We did not configure the CICS SM WLMAppl to CICS SM WCM communication to travel over the switch. The volume of messages was low because there were few changes in the state of the CICS servers and the configured resources. In a more dynamic environment, we would recommend configuring this traffic to travel over the switch. Use the switch address of the nodes in the CICS SM configuration file (/var/cicssm/repos/cicssm.config).

NFS File Access

When a file was mounted as an NFS file system, we ensured that we specified the switch host name of the NFS server. This we could do because the NFS server was located on the RISC System/6000 SP.

On the RISC System/6000 SP we also used the automount daemon (amd) to mount our common file system. This file system contains the scripts and programs required to install and configure the workload. The switch host name of the file owning system was specified to amd. If the system on which such a file system is located is not on the RISC System/6000 SP, you have to use an interface other than the switch.

Verifying Use of the High-Performance Switch

Having made the necessary changes to route traffic over the switch, we verified that most of the network traffic was indeed traveling over the switch. We used the netstat -I css0 5 command, where css0 is the network interface for the switch connection. This command displayed the network traffic on the switch connection as well as the total network traffic for the node. The majority of the traffic for the node should travel over css0 (the switch connection).

With this verification we could see that our initial configuration of DCE was incorrect and that DCE RPCs were traveling over both the SP Ethernet and the switch. We were then able to check the DCE configuration and identify the problem. The DCE check was performed on both a client and server node. Because all of the clients were configured with the same script, if one was functioning correctly, the others should be. Similarly with the server nodes.

Planning and Preparation

The application, its installation, and configuration were already well understood from previous work. However, we were still some way from being able to proceed immediately with work on the RISC System/6000 SP. We spent the majority of the first week of our time on site at the MHPCC, understanding what was currently installed on the RISC System/6000 SP, looking at how the RISC System/6000 SP was configured, and determining how we could implement the demonstration in this unfamiliar environment. We went though several stages:

- Familiarization, understanding the MHPCC method of operation, documenting our requirements
- Redesigning the configuration, adapting to work within the current configuration
- Developing installation scripts, developing and refining scripts to assist with product installation

Once we completed these stages, we were in a position to commence work on the RISC System/6000 SP (see "Configuring the RISC System/6000 SP" on page 29).

Familiarization

As the environment in which the demonstration was to be implemented was alien to us, and the systems administrators at MHPCC were unfamiliar with CICS for AIX, CICS SM, and DB2 for AIX, a two-step familiarization process was required: one to understand the environment in which we were about to begin work, and one for the MHPCC staff to understand at an implementation level what it was that we were about to build on their system.

MHPCC has traditionally supported very different user requirements from those we had. Users typically run batch processing work. The nodes of the RISC System/6000 SP are assigned to job classes in which jobs run for varying periods of time, the length of time being set by the job class. We elected to take machine time on the RISC System/6000 SP as a series of slots rather than in one single block. To accommodate us, the job queues for the requested nodes were drained for the period of our machine slot.

Usually a user does not know or care on which nodes a submitted job ran. The submitted job would create the necessary environment each time, copying data as required at initialization and saving it before job termination. For us it was necessary to always work on the same group of nodes so that the CICS for AIX regions and DB2 for AIX instances were already allocated and available. (The process of CICS for AIX region and DB2 for AIX instance setup is more involved than the straightforward copying of data or initialization of structures.) It was also necessary to determine how to build the demonstration with the minimum of effort and disruption to the existing MHPCC system and users. We also had to bear in mind, that once the demonstration had been given, we would have to remove everything that we had put in place, leaving the RISC System/6000 SP environment with the minimum of change. This is unusual, as typically the products would be installed and remain in use.

The space available for software installation, databases, and logs was very limited, at around 500 MB. The situation was made more acute by the fact that another demonstration was also being built on the same series of nodes for the same conference.

Redesigning the Configuration

Our traditional approach with this workload was to install all of the components required for each of the products that we would use in the demonstration and install this code on each of the nodes. In effect we had a single package. This approach meant that it was possible for a node to function as any type of node, be it a cell server, a management node, a client node, or a server node. Our traditional approach gave greater flexibility but was likely to be the cause of two problems this time. First there would be a shortage of disk space, and second, with 144 nodes to install, the installation time was likely to be significantly increased. The approach was changed to install the minimum amount of code that was required on each node so that the required function could be performed. The functions identified were:

- Client
- Server
- Management
- Cell server

There was merit in this approach. By reducing the number of functions performed on a node, it became easier and quicker to install and customize the necessary software for a particular node type because not all customization had to be performed on all of the nodes. It became more involved to switch the function of a node, but the setup process up was speeded up. Below we look in more detail at each of the four functions.

Client

The client function simulated user activity. In a production environment, such activity would be generated by the users of the system. For the demonstration this activity was achieved through running multiple instances of a CICS ECI program. In project development the use of such a program can be very useful for application and stress testing.

The client function was implemented on a node of the RISC System/6000 SP. This made the node a *client node*. Each of the client nodes was configured identically. There were 42 such nodes. Figure 5 on page 23 shows the composition of a client node. The client function included the CICS SM Workload Management components required to allow the routing of CICS ECI requests. There were two components:

- A workload cache manager. The workload cache manager interfaces with the CICS SM workload management server to ensure that the required information is available in the local cache that is associated with the workload cache manager. This information details which CICS for AIX servers support a requested resource, such as a program or transaction, to which a client is about to route.
- A workload management client. This is a CICS SM-provided CICS exit. Different exits are provided for the routing of different types of work, such as

CICS EPI requests or dynamic transaction routing (DTR). The supplied CICS exit intercepted each ECI request, searched the local workload cache manager to see which CICS for AIX regions supported the requested program, and then applied a workload balancing algorithm to determine which of the available systems was the best to route the work to. Finally, the workload management client updated the workload cache manager′s cache to reflect the routing decision that had been made.

For a more detailed discussion of CICS SM Workload Management, see *IBM CICS System Manager for AIX Product Overview*, GC33-1592-00.

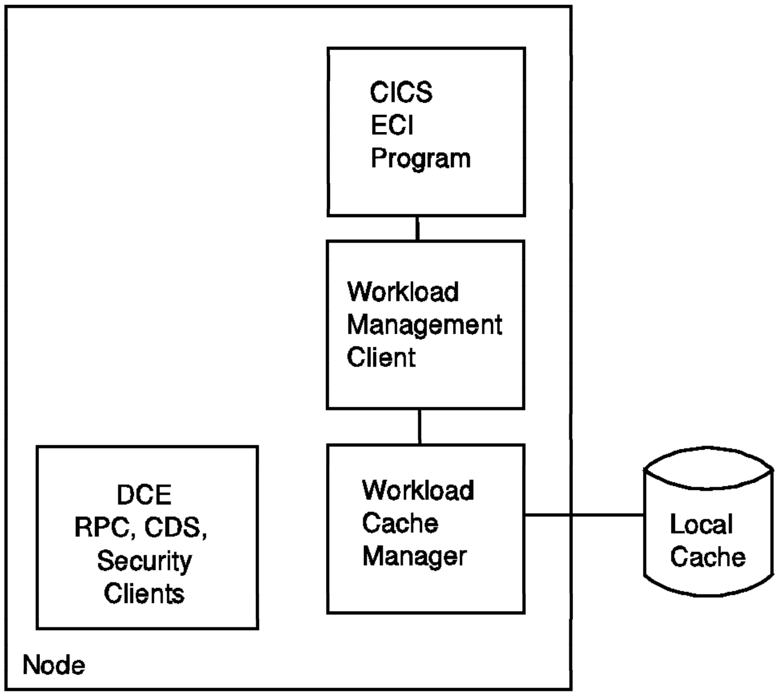


Figure 5. A Client Node

A client node had multiple instances of the CICS ECI program (see "The Application" on page 9 for more details) running, typically around 10 to 15. There was one instance of the workload cache manager, and one instance of the workload management client. The workload management client was invoked multiple times. The client node also had the DCE RPC, CDS, and security clients configured.

Server

The server function processed incoming requests to run workload transactions. A server consisted of:

- A CICS for AIX region
- An Encina SFS
- A DB2 for AIX system
- The CICS SM components required for a managed node
- The DCE RPC, CDS, and security clients

There was no special significance in the fact that we used an Encina SFS rather than DB2 for AIX for CICS file control.

Each ECI request from the multiple instances of the CICS ECI program running on the client nodes resulted in a DebitCredit (see "The Application" on page 9 for more details) transaction being run on a CICS for AIX server. All application data was stored in a DB2 for AIX database. The database was accessed from the CICS application program through the non-XA interface between the CICS for AIX server and DB2 for AIX. This type of connection allows a single agent optimization to occur at syncpoint processing, that is, when the updates that transactions have performed must be either committed or backed out. The use of the single agent optimization leads to a performance improvement when compared with the use of the full XA interface under identical circumstances. It was possible for us to use the non-XA single agent optimization because updates to application data occurred only under one resource manager, namely, DB2 for AIX. Had any CICS recoverable resources, such as a CICS recoverable queue where file control was implemented in an Encina SFS, been updated, a second resource manager would have been involved, and the optimization could not have been used without compromising the integrity of the data.

The server function was implemented on a node of the RISC System/6000 SP. This made the node a *server node*. Figure 6 shows the composition of a server node. Each of the server nodes was configured identically. There were 100 such nodes. As there was one CICS for AIX region per node, each region was given the same name as the host name. Similarly with the Encina SFS. Each DB2 system used the instance name of inst1.

All databases on each of the nodes were replicates. All were created with the same process. However, after creation, no attempt was made to synchronize the updates. The databases on each of the servers thus fell rapidly out of synchronization. Our sole focus was to illustrate the feasibility of a large number of CICS for AIX servers with effective workload management using CICS SM Workload Management.

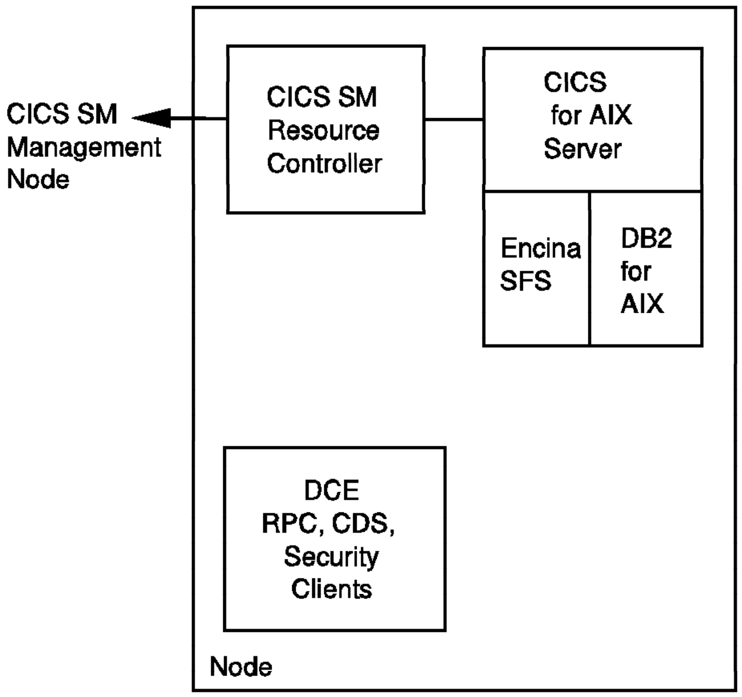


Figure 6. A Server Node

Management

With CICS SM the management function provided a single point of control for the CICS configuration. There were two aspects to the management of the configuration:

- Systems management. This is the starting and stopping of CICS servers and the enabling and disabling of resources.
- Workload management. This is the distribution of work over the available servers.

The management function was implemented on one node of the RISC System/6000 SP. This made the node a *management node*. Figure 7 shows the composition of the management node.

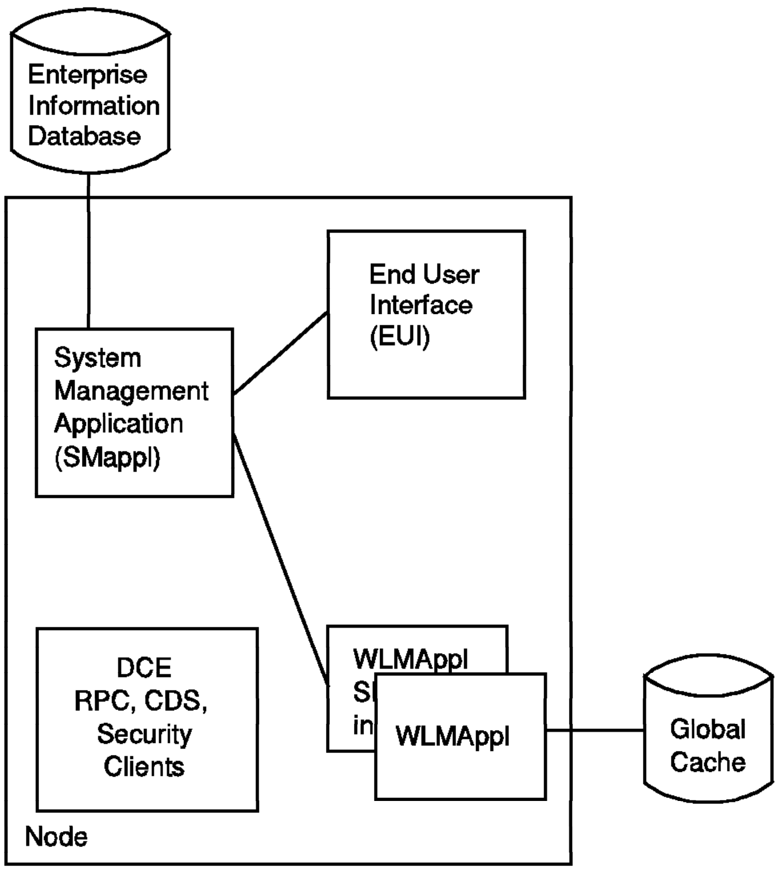

Enterprise Information Database
End User Interface (EUI)
System Management Application (SMappl)
DCE RPC, CDS, Security Clients
WLMAppl
WLMAppl
Global Cache
Node

Figure 7. The Management Node

The management node consisted of the following components:

- System management application (SMappl)
- End user interface (EUI)
- Workload manager application to SMappl interface (WSI)
- Workload manager application (WLMAppl)
- The global cache associated with the WLMAppl

- The DCE RPC, CDS and security clients[1]

Cell Server

The cell server function provided the services of a DCE cell server. The DCE server provided an authentication service through the Security server and a server location service through the CDS.

The cell server function was implemented on a node of the RISC System/6000 SP. This made the node a *cell server node*. Figure 8 shows the composition of the cell server node.

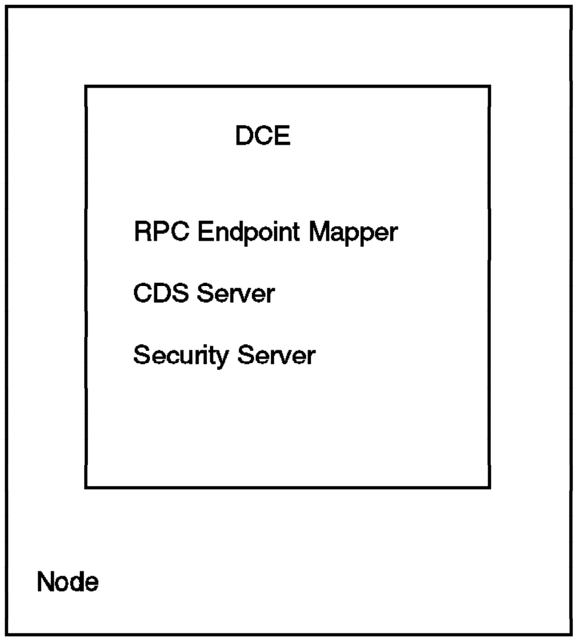


Figure 8. The Cell Server Node

Developing Installation Scripts

Whereas previously we had installed all software on each of the systems, it was now necessary to determine exactly what was required for each of the node types that had been identified. Only software required for run-time support of the various components was installed. In order to perform the compilation of CICS C- and SQL-based programs, it was necessary to install the application development components on at least one system. They were installed on the stand-alone systems only. All program compilations were performed on this machine only, and the load module was copied to the required node. During the course of our investigation we also identified the directories (such as /usr/lpp/cics) that were created as part of the installation process. By using this approach it was possible to remove the file system used to hold the code once the product had been deleted. This saved having to significantly increase the size of /usr for the software installation, only to leave it with a large amount of free space that could not be reclaimed.

[1] CICS SM currently requires DCE to be configured on nodes on which it is located.

We developed Korn shell scripts to install each of the software components, creating and mounting any necessary file systems as well as checking for the installation of any software prerequisites. This work took place on two machines that were not on the RISC System/6000 SP, the aim being to have a complete set of scripts that had been developed and comprehensively tested before ever beginning work on the RISC System/6000 SP.

Once we had developed the installation scripts and installed the software components, we built and verified the workload. We then deinstalled all newly installed software and performed a complete installation and workload setup sequence to verify the scripts.

At this point we had developed and tested a series of scripts to perform product installation and workload setup for each of the nodes types. We were now in a position to proceed to the configuration of the RISC System/6000 SP.

Configuration Summary

Figure 9 shows the overall configuration. The figure illustrates the node allocation and the node interconnection. Each of the nodes has two network connections. The first is to the SP Ethernet, and the second is to the switch. The control workstation is also connected to the SP Ethernet. Thus in the control workstation we have a central administration point from which it is possible to route commands to all or a subset of the nodes. This facility makes it easy to route commands to a particular frame of nodes, all clients, the servers, or any subset of these groups of nodes.

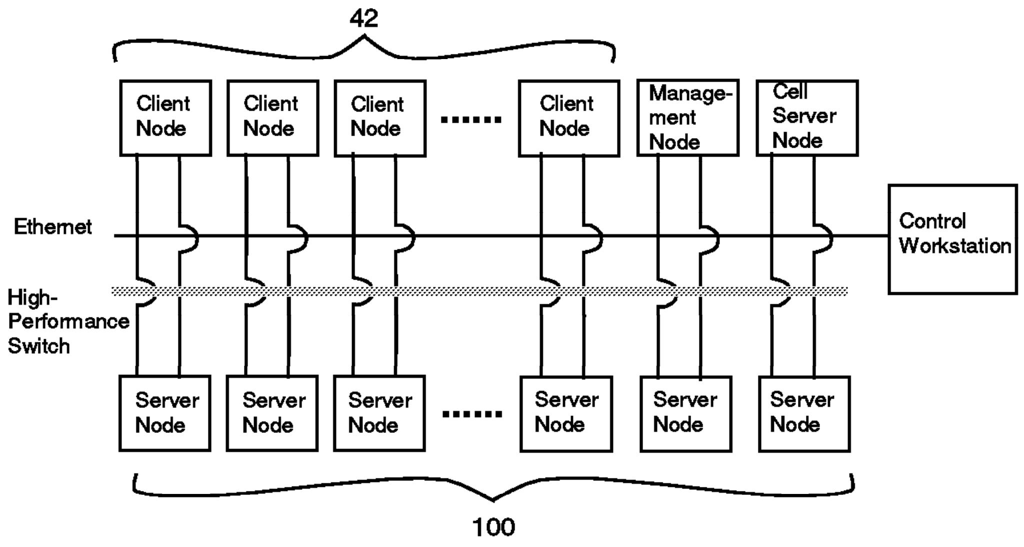


Figure 9. RISC System/6000 SP Node Configuration

Configuring the RISC System/6000 SP

Once we completed the preparation work (see "Planning and Preparation" on page 21), we could begin to build the demonstration on the RISC System/6000 SP. The method that we followed was to first install the software and perform the necessary configuration on a small number of nodes; we chose 16.

Once the first 16 nodes had been installed and successfully configured, the next step was to take one machine slot to install all the necessary code across the remaining 128 nodes. The configuration of the 128 nodes would then take place in steadily increasing numbers, working with 32, 64, 96, and finally all 128 nodes. We believed that it would be too complicated to both install and configure all 128 nodes in one slot. Changes were required to the installation technique in order for us to be able to install across so many nodes concurrently. The details of how to install across multiple nodes concurrently are presented in "Installing the Next 128 Nodes" on page 32.

In building the demonstration on the RISC System/6000 SP our approach was to choose the method that allowed the most rapid implementation because we were buying machine time by the hour. By having programs and scripts to check the health of the CICS for AIX regions and DB2 for AIX instances, we had built in error checking that would help us identify any errors that might creep in during such rapid deployment of a system.

Let us now look in more detail at how the 144 nodes were configured.

Preparation on the RISC System/6000 SP

Before we could begin work on the RISC System/6000 SP, we had to perform a number of administrative tasks:

- Confirm the names of the CICS for AIX, Encina SFS, and DB2 for AIX systems
- Allocate the cics, cicssm, SFS_SERV, and inst1 userids as NIS and SP users
- Allocate the cics and cicssm groups as NIS groups and SP groups
- Reserve a port number in /etc/services for the CICS SM WLMAppl to WCM communication on all 144 nodes
- Add the RPC_UNSUPPORTED_NETIFS=en0,en1,fi0 environment variable to the /etc/environment file on all 144 nodes
- Create an amd entry for the common file system that held the scripts and programs required to perform code installation and customization.

The userid of SFS_SERV was selected as the userid for the Encina SFS on all of the server nodes. An alternative approach would have been to have used a different userid for each of the Encina SFS systems. This would have been an administrative burden with 100 userids to be allocated. The home directory of the SFS_SERV userid had to be different on each node. The home directory name is of the form /var/cics_servers/SSD/cics/sfs/hostname, where hostname is the name of the node. The home directory was correctly set by placing an entry into /etc/passwd on each of the server nodes, overriding the home directory portion of the userid definition. This override was placed before the :+ entry in /etc/passwd file, which marks the start of the inclusion of the NIS userids.

The First 16 Nodes

The first group of 16 nodes was allocated as follows:

- 1 node as a DCE server
- 1 node as a management point
- 4 nodes for CICS clients
- 10 nodes for CICS servers

Here is the sequence of operations that we performed to build a running configuration on the first 16 nodes:

1. Created working collectives for the client nodes, server nodes, and DCE client nodes. The DCE client nodes were the sum of the server and the client nodes.
2. Issued a dsh command to run the script that installed the dce server code on the node allocated to be the cell server. The command was directed to the one node by using the -w flag of the dsh command.
3. Issued a dsh command to run the script that installed the CICS SM management code on the node allocated to be the management node. Again, the command was directed to the one node by using the -w flag of the dsh command.
4. Issued a dsh command to run the scripts that installed the required code on the server nodes
5. Issued a dsh command to run the scripts that installed the required code on the client nodes

   The client and server installations were run concurrently because only one copy of the installp program was running on a node at a time.
6. On the DCE cell server node, manually configured the DCE server. Only the security and CDS servers were configured. Distributed Time Service was not configured as the nodes of the RISC System/6000 SP are already kept in close time synchronization using the Network Time Protocol daemon that is supplied with the AIX Parallel System Support Program (PSSP).
7. On the DCE server node ran a script to perform an administration-only configuration to add each of the DCE client nodes. This approach to DCE configuration was chosen because it avoid having to enter the DCE cell administrator′s password on each of the client nodes. The administration-only configuration was originally implemented to assist with the configuration of remote nodes. It also works well in a RISC System/6000 SP environment where there are potentially many nodes to configure.
8. Issued a dsh command to run the script that would perform the local DCE configuration on each of the client nodes. This script also ran the cicssetupclients script to enable the node as a CICS client node.
9. On the management node performed CICS SM customization:
   - Edited the CICS SM configuration file
   - Added those users permitted to start a CICS SM EUI
   - Added the nodes that had CICS SM resource controllers—that is, the server nodes
   - Added an entry for a WLMAppl

10. Configured CICS SM using the cicssm configure command
11. Started the CICS SM EUI
12. Created the configuration, system template, required workload programs, and transactions
13. Made models from the templates
14. Configured the workload programs on the system model
15. Made clones of the system models—one for each server node
16. Configured the workload separation on the ecidcnx program
17. Created an Encina SFS server on each of the server nodes with a name the same as the host name, that is, the value of the uname -n command, and a shortname of SFS_SERV
18. NFS exported the /var/cicssm/repos file system from the CICS SM management node to all of the client and server nodes
19. Mounted the /var/cicssm/repos file system on each of the client and server nodes. The system was mounted so that the same implementation repository could be shared among the server nodes and the management node and the workload management definitions could be shared among the client nodes and the management node.
20. Ran the CICS SM configure command on each of the server nodes
21. Verified and activated the CICS SM plex. At this point CICS SM allocated the CICS regions on each of the 10 servers.
22. Issued a dsh command to start the CICS SM WCM on each of the client nodes
23. Issued a dsh command to perform the DB2 for AIX customization and database creation on each of the servers.
24. Issued a dsh command to run a script to do the following:
    - Place an entry in the CICS for AIX region environment file for the DB2 instance.
    - Copy the DB2 switch load file to the region bin directory.
25. Using CICS SM, stopped and cold started the CICS for AIX regions to pick up the change to the region environment file

    It was at this point that we discovered a problem with the CICS SM SMappl to CICS SM resource controller communication. The problem caused the CICS SM EUI to hang when a CICS SM resource controller was stopped and restarted. As a result of this problem, we realized that it would be infeasible to use this procedure to build all of the necessary CICS for AIX regions and that a circumvention was required given that we were committed to producing a large demonstration in a finite time. The circumvention was to export one of the CICS SM created CICS for AIX regions using the cicsexport command and to import this region definition on each of the new servers as they were added to the complex. Because of the flexible and dynamic nature of the CICS SM Workload Management component, we were able to continue with the demonstration. Ideally the health monitor information is passed back from the managed node to the SMappl and into the workload management component so that routing decisions can take account of the state of the health of the region. Although the health information was not available, the workload management component could still recognize when a region was either not performing well or down and then route work accordingly.

Note: The SMappl to resource controller communication problem has since been fixed.

26. Issued a dsh command to each server node to run a script that would invoke a CICS ECI program that ran a single database query against the region on that node. This program ensured that the database on each node was opened before transactions were run.

27. Started the measurement script suite to run multiple CICS ECI programs from each of the client nodes against the newly created CICS for AIX regions and so verify that:

    - Work was successfully being routed from each of the client nodes using CICS SM WLM routing
    - Transactions were running in each of the servers

Note: We issued the dsh command from the control workstation of the RISC System/6000 SP only after the working collective was set appropriately.

At this point we had a successful 16-node implementation of the workload.

Installing the Next 128 Nodes

Even with the file system containing the installation images mounted over the switch, we believed that the load generated would be too great to perform a large number of parallel installations. To have the installations proceed as quickly as possible, we decided to copy the installation images out to each of the 128 nodes and run the installation from a local directory. Here is the sequence of installation events:

1. Took a copy of the installation images required for the client and server nodes. This included all of the necessary CICS for AIX, CICS SM, DB2 for AIX, and DCE code.

2. Tried to compress the existing installation images before copying them out to each of the nodes. We achieved less than a 10% compression rate. The compressed file size was still too large, given that we wanted to copy it out to 128 nodes in as short as time as possible and with minimum disruption to other users of the system.

3. Removed as much as possible from the installation image directories. We removed items that were not essential, such as PostScript files and language sets.

4. Wrote the installation images as a single file, using the tar command, and compressed them before they were copied to a local directory on one of the 128 nodes

5. Uncompressed the compressed file in the local directory and restored it to its original directory structure, using the tar command

6. Ran the installation scripts, using the modified installation images to ensure that it was still possible to create both a client and server node

7. Copied the installation images to the remaining 127 nodes, using sample AIX code for a program called mcp (mcp is shipped in /usr/lpp/ppe.poe/samples/mpi/mcp.c), which provides a means of rapidly copying large volumes of data over the switch from one node to another

8. Created two new working collectives, one for the new servers and one for the new clients

9. Issued a dsh command to run the script to install the server images on the new server nodes

10. Issued a dsh command to run the script to install the client images on the new client nodes

The testing of the reduced size installation images and the copying of the images out to the 127 systems were done before the start of a machine slot. In a 4-hour machine slot, it was possible to install all of the required code on 127 nodes.

At this point we had the necessary software installed on all 144 nodes; 16 nodes were completely configured and operational.

Configuring the Next 128 Nodes

The configuration of the newly installed 128 nodes started with the configuration of 32 nodes. The additional nodes were taken in the ratio of one client to two servers. This ratio was determined by the application that we were running. Approximately one client node full of CICS ECI programs was capable of driving two server nodes running the DebitCredit application. By taking the nodes in this manner we were always able to test both the new client and new server nodes because we always had sufficien clients to drive the servers.

In the first slot in our customization of the newly installed nodes, we achieved the following:

1. Created the required clones on the CICS SM management node. Although we were no longer using the systems management component, we had to keep the CICS SM workload management component up to date so that when the plex was activated CICS SM Workload Management contained data in the workload management global cache to indicate that the newly created CICS for AIX regions were available to route work to.

2. Configured DCE on the new clients and new servers, using a script that was run across all required nodes by means of the dsh command

3. Created an Encina SFS server on all of the new server nodes

4. Performed the database setup and population on the new servers

5. Created and customized the CICS for AIX region on each of the new servers. This customization included updating the region environment file and copying the switch load file to the region bin directory.

6. Started the CICS SM WCM on each of the new client nodes

7. Started DB2 for AIX, Encina SFS, and the CICS for AIX region on each of the previously established servers

8. Ran the database initialization program on each of the server nodes

9. Activated the CICS SM plex. This made the workload management information available for the WCMs and hence facilitated the routing of ECI requests.

The configuration of these first 32 nodes from the 128 proceeded very smoothly, giving us great confidence that the installation process had gone well. Consequently we decided to immediately configure the remaining 96 nodes in a single step, so completing the configuration of the 128 nodes. We used the same procedure as described above. This gave us a total of 144 configured nodes.

Measuring and Demonstrating

With the 144-node complex built, we were able to proceed with some measurements to demonstrate that work could be managed by balancing across 100 servers once we had verified that the configuration was running successfully. We also had to prepare a demonstration for customers attending the conference.

A series of Korn shell scripts enabled us to take measurements over a variable number of nodes. Using these scripts in conjunction with a suitably coded CICS ECI program, it was possible to control the length of time that the CICS ECI client program would run. All CICS ECI client programs ran for a specified initialization period to allow for terminal sign on and for the system to become stable. The CICS ECI client program then ran for a specified number of slots. Each slot was of the same duration. The CICS ECI client program was written so as to report statistics on the number of transactions that had been run along with a summary of transaction response times for each of the slots. At the end of a measurement run, the statistics were collected and summarized to produce an overall transaction rate.

Verifying the Configuration

Once the configuration had been built, we had to verify that all of the components functioned correctly. We used a series of short measurements. From previous measurements taken with the initial 16-node configuration, we had an expectation of the overall system throughput. Thus we could judge whether the current throughput was adequate or there was a problem. Additionally a number of scripts were written to search the CICS for AIX region console message file and CSMT message log for error or abend messages. We ran these scripts periodically across all server nodes, using the dsh command to check that there were no problems. We had to adopt this approach since it is not feasible to monitor 100 servers by hand.

In taking the early throughput measurements, we realized that there was a prolonged initialization period and overhead before the clients could start fully functioning. Part of this overhead was the DCE authentication that each CICS ECI program had to perform. With several hundred CICS ECI programs trying to concurrently authenticate, the delays became significant as the DCE server processed the requests as quickly as it could. This was an artificial situation, because in no enterprise would 10,000 users concurrently start work. For the demonstration, our principal aim was to demonstrate the effectiveness of CICS SM Workload Management. Any delays in getting the system up and running were to be avoided, especially as we were paying for machine time by the hour. Accordingly we decided to assist the initialization process and opt for a reduced DCE configuration, known as a *non-DCE* or *DCE lite* configuration. *DCE lite* is the more accurate name because DCE RPC is still used; the DCE CDS and security are not, however.

The configuration work required to remove DCE CDS and Security is described in Appendix C, “Reconfiguring DCE” on page 47. With the use of DCE lite, the client initialization time was reduced. The CICS client to server traffic continued to travel over the switch as DCE RPC was configured to use the switch.

Running With 100 Servers

The 100-server run consisted of 100 server nodes, 42 client nodes, and 10,000 simulated users. During the measurement period of 1 hour, this configuration processed around 14 million transactions, with a sustained peak over a 10-minute period of 6289 transactions per second.

In analyzing this run, we found evidence of memory constraint on the client nodes. This constraint caused a number of client processes to become lost, and hence throughput was reduced. We were severely time constrained, which prevented a rerun of this particular measurement. Despite the limitations of this run, we had still demonstrated the ability to balance work over 100 CICS for AIX regions with CICS SM Workload Management. We had achieved our objective.

The Demonstration

The long initialization required for the system to become stable when running with a high number of servers and users meant that it was not feasible to demonstrate the 100-server system to customers attending the conference. The schedule was such that we gave the demonstration four times in one day to different groups of customers. To work with something more manageable, we demonstrated a 48-node configuration consisting of:

- 1 CICS SM management node
- 15 client nodes
- 32 server nodes

It is very difficult to adequately demonstrate transactions running in a system. Our approach was to use the xmperf tool to capture a number of system metrics on the server nodes and display them using the 3DMON tool. This allowed us to clearly demonstrate that there was CPU activity on each of the 32 server nodes. As an illustration of the dynamic nature of CICS SM Workload Management, we forcibly closed five CICS for AIX regions during the course of the demonstration. The audience saw the corresponding rise in CPU utilization of the remaining server nodes as the work was redistributed over the reduced number of servers. The redistribution was performed without manual intervention.

Recommendations

As a result of the experience we accumulated, there are a number of recommendations we would make to anyone either implementing or running a CICS for AIX configuration on a RISC System/6000 SP. The recommendations are divided into the following categories:

- CICS for AIX
- CICS SM
- RISC System/6000 SP
- DCE

Let us now look at the recommendations.

CICS for AIX

- Initially try to configure each node with one CICS for AIX region and either an Encina SFS or a DB2 for AIX system, depending on how you want to configure CICS file control.
- Where possible, locate the CICS for AIX clients inside the RISC System/6000 SP. (This is often not possible, however.)
- If there is CICS for AIX region to CICS for AIX region communication with the RISC System/6000 SP, ensure that traffic flows over the switch. Configure both CICS for AIX regions to use CICS family TCP/IP and ensure that the host name given for the remote system in both the listener definition (LD) and communication definition (CD) stanza files is the switch address of the host on which the remote CICS for AIX region runs.
- Consider making applications available in multiple CICS for AIX regions. This availability, in conjunction with CICS SM Workload Management, can help to significantly increase application availability in the event of an outage in one CICS for AIX region.
- Look at the use of specialized nodes. Consider the following two examples:
  - If a node is going to perform only the function of a client, why install all the server code as well?
  - If is likely that application development will proceed on only a small number of nodes and perhaps not on the RISC System/6000 SP at all, why install all application development software on these nodes?

  When there are many machines or nodes on which software must be installed, the implementation of specific node types can lead to easier installation and configuration.
- Where possible, uniquely identify incoming work; that is, write each CICS ECI application to use a different transaction identifier instead of the CPMI identifier. When work can be uniquely identified in this manner, it makes it is much easier to identify problems and monitor the performance of such work.

CICS SM

- Run the CICS SM management function on its own node where possible or alternatively on an underutilized node to ensure that plenty of CPU is available to process plex activations.
- For more than 8 to 10 servers in a plex that is CICS SM workload managed, look at the use of two-level data-dependent routing. Using the default approach results in an increasing overhead for the client as there are more and more system clones to be scored by the workload management component. Each time a routing decision is made, each of the eligible system clones has to be scored by the workload management client before selecting the most suitable. The overhead rises linearly with an increasing number of servers from which to choose.
- When performing workload management with a multithreaded CICS ECI program, ensure that the number of threads in use multiplied by the number of regions visited does not exceed the thread limit of a CICS ECI program. The thread limit is currently set to 16. If the number of threads multiplied by the number of regions visited exceeds 16, you are likely to see increased CPU usage for both the CICS ECI program and any CICS for AIX regions to which work has been routed. This increased usage is due largely to the implementation of the CICS ECI support, which is implemented in such a way that a terminal is installed in the CICS for AIX region to which a CICS ECI request is made. If a free terminal is not installed in the CICS for AIX region at the point at which the CICS ECI request is made from the CICS ECI program, the supplied CICS ECI code will cause a terminal to be installed. As the limit is 16, if a seventeenth request is made from the CICS ECI program to a CICS for AIX region that does not have a free terminal installed by this CICS ECI program, one of the previous 16 terminals will be uninstalled. A terminal will then be installed in the new CICS for AIX region. This install and uninstall activity can create a significant overhead on the CICS for AIX regions if allowed to proceed in any volume. As an illustration, if a CICS ECI program could potentially route work to any one of five CICS for AIX regions, the maximum number of suggested threads to use would be 3.

RISC System/6000 SP

- Direct as much traffic over the switch as possible. This includes:
  - Client to server traffic where possible. If the clients are located outside the RISC System/6000 SP, consider directing the traffic to a number of wide nodes that have external network connections. From that point the traffic should be directed to flow over the switch once inside the RISC System/6000 SP.
  - File mounts
  - Any relevant intercomponent traffic, such as DB2 for AIX client to DB2 for AIX server traffic when the DB2 for AIX server is located on a separate node. The DB2 for AIX client portion would be located on the same node as the CICS for AIX region.
- Maximize the use of the dsh command. The command allows work to proceed in parallel and reduces the elapsed time required to perform a function across multiple nodes.

- Look closely at the code that has to be installed on each node. Not all software necessarily has to be installed on all nodes. You may want to install all software on one or two nodes so that application development can proceed. Excess software beyond that needed to actually run the system takes additional time to install and consumes disk space.
- Initially work on the minimum number of nodes. This could as small as one or two. Use these nodes to check out scripts and ensure that all works as expected before moving to a larger number of nodes. If you run commands that are incomplete or incorrect in some way, for *n* nodes you have *n* times to correct the problem. Sometimes this may require logging on to the node manually. It becomes very easy to spend a lot of time cleaning up after a mistake.
- Where there is significant traffic into and out of the RISC System/6000 SP, consider these options: If you have thin nodes, install wide nodes. If you have wide nodes, install network adapters. The SP Ethernet has a very limited bandwidth, so do not be constrained by it.
- Use systematic naming conventions for all nodes. The node name should include only the host name, not the domain name. Make the host names associated with the SP Ethernet and switch addresses very similar in format; for example, fr17n01 for the SP Ethernet address, and fr17s01 for the switch address. With this approach, it is obvious which interface is being used. If you were to use colors, it is not intuitive that red means the switch interface.
- Use logical naming conventions. Consider naming nodes by their frame and slot number. That might not be a very imaginative approach, but it is logical and sensible. It also makes it very easy to locate a node and understand quickly whether a problem affects a portion of a frame or the whole frame. Names based on a theme, such as place names, do not allow resource location to be made as easily or rapidly.

DCE

- Locate the DCE server within the RISC System/6000 SP. If this is not feasible, ensure that there is good communication to the DCE server. Do not rely on the SP Ethernet as the communications path from the collection of CICS for AIX regions in the RISC System/6000 SP to the DCE server, which is located outside.
- Consider running the DCE server on its own node within the RISC System/6000 SP. Whether it is a good idea or not will depend largely on the level of activity that takes place. If there are many users who regularly authenticate, you may want to allocate the cell server on its own node. This was something that we had to do. Our environment was atypical, however, because in most installations user authentications gradually increase in number; there is not a large surge, as there was with our measurements.
- Configure DCE traffic to flow over the switch. Ensure that RPC_UNSUPPORTED_NETIFS is set to the appropriate value for your environment and that it is present in the environment in which the DCE server and clients are configured. Also ensure that RPC_UNSUPPORTED_NETIFS is present in the environment of all those servers that will register their addresses with the DCE CDS. This is most easily achieved by adding RPC_UNSUPPORTED_NETIFS to the /etc/environment file on each node.

- If DCE is configured to run over the switch, and there is DCE RPC communication from machines outside the RISC System/6000 SP to servers within the RISC System/6000 SP, ensure that the switch addresses of the nodes are available outside the RISC System/6000 SP. If you do not do this, it will not be possible to locate the servers within the RISC System/6000 SP. This addressing problem is most easily solved by adding an entry in the name server for the switch address for each node of the RISC System/6000 SP.

Appendix A. Sample Data-Dependent Routing Program

```
/* ------------------------------ BGH ---------------------------------------
//
// File : wuexmp.c
// ---------------
//
// Description : Basic WLM Client 'C' User Exit
// -------------
//
//              Build with:
//
//                 make -f wlm_make bhgwlmexit
//
// This file implements the WLM Client User Exit in the simplest way possible.
// Called successively for each partition, this function will compare the
// name of the partition with a name held in the PARTITION_NAME environment
// variable. If they match, the current partition is given the "choice"
// value of one and partition scanning is terminated. Otherwise wait for the
// next one to come along.
//
//
// Author : Mike Taylor
// --------------------
//
//  Extended by Keith Edwards and Tim Dunn to reference the COMMAREA of
//  the CICS ECI call and look for the Partition name. This is then compared
//  with the current partition name and if this is the required partition
//  then we issue a return code of 1, to say no more calls please.
//  This program could just as easily be looking at the account code for
//  example.
//
//  Date: 15 July 1996
//
// ------------------------------ EBGH ------------------------------------*/

#include<stdlib.h>

enum exit_style_type {DPL, DTR, ECI, EPI, CICSTERM, CICSTELD};


struct CommArea
{
    char               Account[4] ;           /*   4 bytes  */
    char               Branch[4] ;            /*   4 bytes - alignmen*/
    char               Teller[4] ;            /*   4 bytes  */
    char               Amount[4] ;            /*   4 bytes  */
    char               AccountBalance[4] ;    /*   4 bytes  */
                                              /*  20 Subtotal         */
    char               Partition[12]   ;      /* 12 bytes   */
    char               Padding[168]    ;      /* 168 bytes  */
                                              /* 200 bytes total      */
} ;


int bh_wlm_user_exit
(
  enum exit_style_type   exit_style,
  int                    partition_number,
  const char            *resource_model_name,
  const char            *separation_name,
  const char            *partition_name,
  const char            *partition_data,
  const char            *data_ptr_1,
  int                    data_len_1,
  const char            *data_ptr_2,
  int                    data_len_2,
  const char            *data_ptr_3,
  int                    data_len_3,
  const char            *data_ptr_4,
  int                    data_len_4,
  int                   *choice
)

{

  char                partition_from_exit[12] ;
  char                partition_to_use[12] ;
  int                  return_code;

  strncpy(partition_to_use , data_ptr_1+20 , 12) ;
  if ( 0 == partition_to_use )
  {
    /* error - environment variable not set */
    printf("Environment variable PARTITION_NAME is not set\n");
  }

  /* compare the partion name in the environment variable with
     name of this partition to see if this is the one to use */
  if ( 0 == strcmp(partition_to_use,partition_name) )
  {
    /* yes, this is the one to use */
    *choice = 1;
    return_code = 1; /* no more calls thankyou */
  }
  else
  {
    /* no, not the one */
    *choice = 0;
    return_code = 0; /* try another please */
  }

  return return_code;
}
```

Appendix B. Software Installation

For software installation a number of Korn shell scripts were written. These scripts were responsible for installing a software component or a group of components. The appropriate scripts would then be run on a client, a server, or both depending on the components that were required on that node type. In the section that follow, we briefly look at the scripts and the software that was installed as a result of running them.

Software Installation Scripts

The Korn shell scripts that were written to perform the installation of groups of software largely corresponded to the node type. The script names that were created are listed below. Only the script names are given for the sake of brevity. The names do give a good indication of the installation work that each script performed. Each script performed the installation of multiple components. Exactly which components were installed on each of the node types is given in "Installed Software by Node Type" on page 44.

- install_cics_client
- install_cics_server
- install_cicssm_server
- install_compat_net
- install_db2_server
- install_dce_client
- install_dce_server
- install_encina
- install_som

Note: The install_compat_net script was used to install a bos compatibility fix, which was required for CICS SM Workload Management. The fix is required for compatibility between AIX V3 and AIX V4.

Having high-level scripts that perform the installation of multiple components simplified the process of installing software over many systems. During the installation process the required scripts would be run on each node.

Table 2 indicates which scripts were run on which node type.

Table 2 (Page 1 of 2). Summary of Installation Scripts and Nodes Types

Script	Client	Server	Management	Cell Server
install_cics_client	X			
install_cics_server		X		
install_cicssm_server			X	
install_compat_net	X	X	X	X
install_db2_server		X		
install_dce_client	X	X	X	
install_dce_server				X
install_encina		X		

Table 2 (Page 2 of 2). Summary of Installation Scripts and Nodes Types

Script	Client	Server	Management	Cell Server
install_som		X	X	

Installed Software by Node Type

This section lists the software installed by node type. The minimum necessary software was installed on each of the nodes.

Client Node

```
bos.compat.net
cics.base.rte
cics.client.rte
cics.msg.en_US.base
cicssm.rte
cicssm.wlmclient
cicssm.wlmenabler
cicssm.msg.en_US.rte
dce.client.core.rte
dce.client.core.rte.admin
dce.client.core.rte.cds
dce.client.core.rte.config
dce.client.core.rte.rpc
dce.client.core.rte.security
dce.client.core.rte.time
dce.client.core.rte.zones
dce.compat.client.core.smit
dce.msg.en_US.client.core.rte
dce.msg.en_US.compat.client.core.smit
dce.pthreads.rte
dce.msg.en_US.pthreads.rte
```

Server Node

```
bos.compat.net
cics.base.rte
cics.server.rte
cics.server.rte.ivp
cics.client.rte
cics.base.adt
cics.server.adt
cics.client.adt
cics.msg.en_US.base
cicssm.rte
cicssm.rc
cicssm.smenabler
cicssm.msg.en_US.smenabler
cicssm.msg.en_US.rte
dce.client.core.rte
dce.client.core.rte.admin
dce.client.core.rte.cds
dce.client.core.rte.config
dce.client.core.rte.rpc
dce.client.core.rte.security
dce.client.core.rte.time
dce.client.core.rte.zones
dce.compat.client.core.smit
dce.msg.en_US.client.core.rte
dce.msg.en_US.compat.client.core.smit
dce.pthreads.rte
dce.msg.en_US.pthreads.rte
```

Management Node

```
bos.compat.net
cicssm.rte
cicssm.rc
cicssm.smenabler
cicssm.adt
cicssm.eui.icons
cicssm.eui.rte
cicssm.msg.en_US.eui
cicssm.msg.en_US.rte
cicssm.msg.en_US.smenabler
cicssm.smappl
cicssm.wlmappl
cicssm.wlmclient
cicssm.wlmenabler
cicssm.wlmserver
somwg.rte
somwg.somuc
som.ir
somwg.util
somwg.somr
somwg.somp
somwg.dsom
somtk.somuc
somtk.somr
somtk.somp
somtk.rte
somtk.ir
somtk.util
somtk.dsom
somtk.comp
dce.client.core.rte
dce.client.core.rte.admin
dce.client.core.rte.cds
dce.client.core.rte.config
dce.client.core.rte.rpc
dce.client.core.rte.security
dce.client.core.rte.time
dce.client.core.rte.zones
dce.compat.client.core.smit
dce.msg.en_US.client.core.rte
dce.msg.en_US.compat.client.core.smit
dce.pthreads.rte
dce.msg.en_US.pthreads.rte
```

Cell Server Node

```
dce.client.core.rte
dce.client.core.rte.admin
dce.client.core.rte.cds
dce.client.core.rte.config
dce.client.core.rte.rpc
dce.client.core.rte.security
dce.client.core.rte.time
dce.client.core.rte.zones
dce.compat.client.core.smit
dce.msg.en_US.client.core.rte
dce.msg.en_US.compat.client.core.smit
dce.msg.en_US.compat.client.core.smit
dce.pthreads.rte
dce.msg.en_US.pthreads.rte
```

Appendix C. Reconfiguring DCE

In the full configuration, DCE RPC, CDS, and security were configured. In the DCE lite configuration, only DCE RPC was configured. The additional information required for clients to locate servers is generated as a part of the reconfiguration process.

Reconfiguration was required on both the client and server nodes. The reconfiguration work required was different for both nodes types. Korn shell scripts were written to perform the necessary processing. This gave us the ability to run the reconfiguration in parallel when the scripts were invoked by using the dsh command. The reconfiguration of DCE from a full configuration to a DCE lite configuration proceeded very quickly and smoothly. Once we were ready to perform the reconfiguration, the work took approximately 30 minutes for the 100 server and 42 client nodes.

Client Reconfiguration

The processing required on the client nodes was minimal. The first step was to remove the existing DCE configuration, using the rmdce command. We configured DCE RPC, using the cicscp create dce command. The environment variable RPC_UNSUPPORTED_NETIFS was already present in /etc/environment and hence present at the time at which DCE RPC was configured.

A new environment variable, CICS_HOSTS, was added to the environment of the CICS ECI programs. CICS_HOSTS was set to a list of host names that could contain the region being requested. When a CICS client attempted to connect to a CICS for AIX region, the list of host names was searched until the CICS for AIX region was found on one of the hosts.

Server Reconfiguration

The steps required to reconfigure the server nodes were as follows:

1. Stop the CICS for AIX region.
2. Stop the Encina SFS server.
3. Remove the existing DCE configuration on the node, using rmdce.
4. Configure DCE RPC, using the cicscp create dce command. The RPC_UNSUPPORTED_NETIFS environment variable was already present in /etc/environment and hence present at the time at which DCE RPC was configured.
5. Change the Authentication name service in the CICS for AIX region RD stanza file to CICS.
6. Change the Nameservice and ProtectionLevel attributes in the SFS server definition to be none. We performed these changes, using the cicsupdateclass command.
7. Create a binding file to allow the SFS to locate the CICS for AIX region. We created the file by copying the required information into a file called *server_bindings* in the /var/cics_servers directory.

Full information on configuring to run in a DCE lite environment is provided in the *CICS on Open Systems Administration Reference*, SC33-1563.

Appendix D. Special Notices

This publication is intended to assist systems administrators and IBM systems engineers in planning and configuring a CICS for AIX and CICS SM configuration on a RISC System/6000 SP system. It illustrates the work that was involved in creating a very large CICS for AIX and CICS SM configuration. The information in this publication is not intended as the specification of any programming interfaces that are provided by Transaction Server for AIX or AIX. See the PUBLICATIONS section of the IBM Programming Announcement for Transaction Server for AIX 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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 performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

AIX	AIXwindows
C Set ++	CICS
CICS/ESA	CICS/MVS
Enterprise Systems Architecture/390	ESA/390
IBM	MVS/ESA
NetView	Personal System/2
PS/2	RACF
RISC System/6000	RS/6000
ThinkPad	

Computer Associates	CA-ACF2, CA-TOP SECRET
Open Software Foundation	DCE
Transarc	Encina

Appendix E. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

International Technical Support Organization Publications

For information on ordering these ITSO publications see "How To Get ITSO Redbooks" on page 53.

- *Addressing OLTP Solutions with CICS: The Transaction Server for AIX*, SG24-4752.

Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	**Subscription Number**	**Collection Kit Number**
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

Other Publications

These publications are also relevant as further information sources:

- *IBM CICS System Manager for AIX Product Overview*, GC33-1592
- *CICS Family: Client/Server Programming*, SC33-1435-01
- *CICS on Open Systems Administration Reference*, SC33-1563-01.

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL `http://www.redbooks.ibm.com`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type `GOPHER.WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **ITSO4USA category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

- **Telephone orders**

- **Mail Orders** — send orders to:

- **Fax** — send orders to:

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to `softwareshop@vnet.ibm.com`

- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to `announce@we-19wareote` tohops(serv(leavstser-maij35 T 0.0blank).6.2 Tm 0 Tc 07o)Tj 7 23/F5 64

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

- **Please put me on the mailing list for updated versions of the IBM Redbook Catalog.**

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.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

Glossary

A

abend. Abnormal ending of transaction.

API. Application programming interface. A set of calling conventions defining how a service is invoked through a software package.

APPC. Advanced Program-to-Program Communication. An implementation of SNA′s LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

application manager. A CICS run-time process that manages a pool of application servers.

application server. A CICS run-time process that executes CICS tasks.

application unit of work. A set of actions within an application that the designer chooses to regard as an entity. It is up to the designer to decide how, if at all, an application should be subdivided into application units of work, and whether any application unit of work will consist of one, or many, logical units of work (LUWs). Typically, but not exclusively, an application unit of work corresponds to a CICS transaction.

asynchronous. Without regular time relationship; unexpected or unpredictable with respect to the execution of a program instruction. See *synchronous*.

B

business process. An entity-handling activity that is of limited duration, defined in scope, and set by business goals and policies, not by organization or implementation.

C

CEMT. A CICS-supplied transaction that invokes all master terminal functions. These functions include inquiring about and changing the value of parameters used by CICS, altering the status of system resources, and terminating tasks.

cicsprnt. A CICS-supplied program that provides the ability to define a printer terminal on the client workstation. The program enables CICS applications running on the server to direct output to the client-attached printer. It is supplied with the CICS Clients.

cicsterm. A CICS-supplied program that provides 3270 emulation and enables connection to a CICS region. It is provided as part of the CICS Clients.

CICS. Customer Information Control System (CICS). A distributed online transaction processing system—an online system controller and some utilities that are capable of supporting a network of many terminals. The CICS family of products provides a range of application platforms on many operating system platforms.

client. As in client-server computing, the application that makes requests to the server and, often, deals with the interaction necessary with the user.

client-server computing. A form of distributed processing, in which the task required to be processed is accomplished by a client portion that requests services and a server portion that fulfills those requests. The client and server remain transparent to each other in terms of location and platform. See *client*, *distributed processing*, and *server*.

commit. An action that an application takes to make permanent the changes it has made to CICS resources.

conversational. Communication model where two distributed applications exchange information by way of a conversation. Typically one application starts (or allocates) the conversation, sends some data, and allows the other application to send some data. Both applications continue in turn until one decides to finish (or deallocate). The conversational model is a synchronous form of communication.

cooperative processing. The process by which a single application is divided between two or more hardware platforms. Very often the term is used to reflect a tightly coupled relationship between the parts of the application.

D

database. (1) A collection of interrelated data stored together with controlled redundancy according to a scheme to serve one or more applications. (2) All data files stored in the system. (3) A set of data stored together and managed by a database management system.

DCE. Distributed Computing Environment. Adopted by the computer industry as a de facto standard for distributed computing. DCE allows computers from a variety of vendors to communicate transparently and share resources such as computing power, files, printers, and other objects in the network.

design. The process of composing a software blueprint, showing how to build from the requirements specification document. Design often includes module decompositions, data structure definitions, file format definitions, and important algorithm descriptions.

distributed processing. Distributed processing is an application or systems model in which function and data can be distributed across multiple computing resources connected on a local area network or wide area network. See *client-server* computing.

DPL. Distributed program link. Provides a remote procedure call (RPC)-like mechanism by function shipping EXEC CICS LINK commands. In CICS/6000, DPL allows the linked-to program to issue unsupported CICS/6000 function shipping calls, such as to DB2 and DL/I, and yields performance improvements for transactions that read many records from remote files.

DTP. Distributed transaction processing. Type of intercommunication in CICS. The processing is distributed between transactions that communicate synchronously with one another over intersystem links. DTP enables a CICS application program to initiate transaction processing in a system that supports LU 6.2 and resides in the same or a different processor.

E

EUI. End-user interface. Used interchangeably with GUI. See *GUI*.

ECI. External call interface. An application programming interface (API) that enables a non-CICS client application to call a CICS program as a subroutine. The client application communicates with the server CICS program, using a data area called a COMMAREA.

Encina. Enterprise computing in a new age. A set of DCE-based products from Transarc Corporation that are available on the RISC System/6000. The Encina family of online transaction processing products includes:

- Encina Toolkit Executive
- Encina Server
- Encina Structured File Server (SFS)
- Encina Peer-to-Peer Communication Executive (PPC).

EPI. External presentation interface. An application programming interface (API) that allows a non-CICS application program to appear to the CICS system as one or more standard 3270 terminals. The non-CICS application can start CICS transactions and send and receive standard 3270 data streams to those transactions.

environment. The collective hardware and software configuration of a system.

F

file server. A centrally located computer that acts as a storehouse of data and applications for numerous users of a local area network.

G

GUI. Graphical user interface. A style of user interface that replaces the character-based screen with an all-points-addressable, high-resolution graphics screen. Windows display multiple applications at the same time and allow user input by means of a keyboard or a pointing device such as a mouse, pen, or trackball.

H

heterogeneous network. A local or wide area network comprising hardware and software from different vendors, usually implementing multiple protocols, operating systems, and applications.

high-performance switch. A high-bandwidth network within a RISC System/6000 SP that interconnects the nodes.

host. (1) In a computer network, a computer providing services such as computation, database access, and network control functions. (2) The primary or controlling computer in a multiple computer installation.

I

instance. An implementation of a database user that enables databases to be allocated.

intercommunication. Communication between separate systems by means of Systems Network Architecture (SNA), Transmission Control Protocol/Internet Protocol (TCP/IP), and Network Basic Input/Output System (NetBIOS) networking facilities.

interoperability. The ability to interconnect systems from different manufacturers and have them work together to satisfy a business requirement. Some examples of requirements are message interchange between systems, and sharing of resources, such as data, between applications running on different hardware and software platforms.

L

LU type 6.2 (LU 6.2). Type of logical unit used for CICS intersystem communication (ISC). The LU 6.2 architecture supports CICS host to system-level products and CICS host to device-level products. APPC is the protocol boundary of the LU 6.2 architecture.

LUW. Logical unit of work. An update that durably transforms a resource from one consistent state to another consistent state. A sequence of processing actions (for example, database changes) that must be completed before any of the individual actions can be regarded as committed. When changes are committed (by successful completion of the LUW and recording of the syncpoint on the system log), they do not have to be backed out after a subsequent error within the task or region. The end of an LUW is marked in a transaction by a sync point that is issued by either the user program or the CICS server, at the end of task. If there are no user sync points, the entire task is an LUW.

M

macro. An instruction that when executed causes the execution of a predefined sequence of instructions in the source language in which the macro is embedded. The predefined sequence can be modified by parameters in the macro.

messaging. A communications model whereby the distributed applications communicate by sending messages to each other. A message is typically a short packet of information that does not necessarily require a reply. Messaging implements asynchronous communications.

middleware. Middleware is a set of services that allows distributed applications to interoperate on a local area network or wide area network. It shields the developer or end user from the system complexity and enables delivery of service requests or responses transparently across computing resources.

O

object. A program or a group of data that can behave like a thing in the real world.

OLTP. Online transaction processing. A style of computing that supports interactive applications in which requests submitted by terminal users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. An online transaction processing system supervises the

A year later ISC gave SQL formal international standard status.

stored procedures. Facility for storing procedural code associated with relational database management systems (RDBMSs) that enforces the procedure′s use during any database operation.

synchronous. (1) Pertaining to two or more processes that depend on the occurrence of a specific event such as a common timing signal. (2) Occurring with a regular or predictable time relationship.

sync point. A logical point in execution of an application program where the changes made to the databases by the program are consistent and complete and can be committed to the database. The output, which has been held up to that point, is sent to its destination, the input is removed from the message queues, and the database updates are made available to other applications. When a program terminates abnormally, CICS recovery and restart facilities do not back out updates made before the last complete sync point.

T

test. Testing involves checking each individual module built during the implementation phase, then integrating the modules into a single program structure. The program as a whole is then tested to ensure that it performs as designed.

thread of control. Inside the X/Open DTP model, the concept that associates resource manager work with the global transaction. Routines in the XA interface that manage the association between a thread of control and transactions must be called from the same thread.

transaction. A unit of processing (consisting of one or more application programs) initiated by a single request. A transaction can require the initiation of one or more tasks for its execution.

transaction branch. A part of the work in support of a global transaction for which the transaction manager and the resource manager engage in a commitment protocol coordinated with, but separate from, that for other branches.

transaction manager. Provides the function to begin, end, commit, and roll back transactions.

transaction monitor. Provides a total environment for transactional applications. In addition to transaction manager functions, provides services to aid development, execution, and operation of transaction applications.

transaction processing. A style of computing that supports interactive applications in which requests submitted by users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. A transaction processing system supervises the sharing of resources for processing multiple transactions at the same time.

two-phase commit. For a database, a protocol that is used to ensure uniform transaction commit or abort in a distributed data environment between two or more participants. The protocol consists of two phases: the first to reach a common decision, and the second to implement the decision.

TX. Within the DTP model adopted by X/Open, the interface among the application or transaction monitor and the transaction manager.

W

workstation. A configuration of input and output equipment at which an operator works. A terminal or microcomputer, usually one that is connected to a mainframe or a network, at which a user can perform applications.

X

XA. Within the DTP model adopted by X/Open, the interface between the transaction manager and resource managers.

List of Abbreviations

AIX	Advance Interactive Executive
ATM	asynchronous transfer mode
BMS	basic mapping support
CDS	cell directory services
CICS	Customer Control Information System
CICS SM	CICS System Manager for AIX
COMMAREA	communication area
DB2	DATABASE 2
DCE	Distributed Computing Environment
DTR	dynamic transaction routing
ECI	External Call Interface
EUI	End User Interface
FDDI	Fiber Distributed Data Interface
HPS	high-performance switch
IBM	International Business Machines Corporation
IT	information technology
LD	listener definition
MHPCC	Maui High Performance Computing Center
PSSP	AIX Parallel System Support Programs
OLTP	online transaction processing
RPC	remote procedure call
SMP	symmetric multiprocessor
SFS	Encina Structured File Server
SMAppl	system management application
WCM	workload cache manager
WLMAppl	workload management application

Printed in U.S.A.

SG24-4836-00