Benchmark and Performance Analysis of TurboBLAST on IBM® xSeries™ Server Cluster

Ruzhu Chen, Clarisse Taaffe-Hedglin, Nathan Willard, Andrew H. Sherman

ruzhuchen@us.ibm.com

January 27, 2002

Abstract

TurboBLAST of Turbogenomics, Inc, is a software program that provides a high performance, remotely accessible BLAST service based on multiple executions of the unmodified NCBI BLAST wrapper *blastall* program. TurboBLAST outperforms the NCBI BLAST by performing parallel similarity searches on multiple machines (cluster). This article presents the benchmark results of TurboBLAST obtained on an IBM® xSeries™ (x330) server cluster. The performance is analyzed over the size of the input queries and the length of each query, as well as the performance effect of the number of alignments returning to the output file. Inputs of sequences with different sizes and lengths were prepared and used as benchmarking criteria. The results showed linear speedup in terms of elapsed time of up to 116 processors for extra long query, and up to 32 processors for short query, regardless of the size of the input file. The speedup, however, was affected by the number of database sequences to show alignments and one-line descriptions (*i.e.*, **tblastall** options **-b** and **-v** values) returning in the outputs.

Introduction

What is *BLAST*?

New molecular biology techniques, such as genomic sequencing as well as PCR (polymerase chain reaction), microarray, and EST (expression sequence tag), allow us to obtain thousands of genomic sequences daily in one lab. These novel sequences need to be elucidated quickly. BLAST (Basic Local Alignment Search Tool) is a set of powerful similarity search tools to identify novel DNA or protein sequences by matching with previously characterized genes and proteins presented in genomic or protein databases. BLAST results could give the biologists both functional and structural information of the novel DNA or protein sequences.

BLAST was primarily developed by NCBI (National Center for Biotechnology Information). Since its release in 1990, the BLAST programs have been designed for speed, with a minimal sacrifice of sensitivity to distant sequence relationships. The scores assigned in a BLAST search have a well-defined statistical interpretation, making real matches easier to distinguish from random background hits. BLAST uses a heuristic algorithm which seeks local as opposed to global alignments and is therefore able to detect relationships among sequences which share only isolated regions of similarity [Altschul *et al.*, 1990].

The BLAST was designed to execute in a standalone machine. In order to use machine resources extensively on multiprocessors computers, several parallel mechanisms were used for BLAST. POSIX pthread-version BLAST (such as the newly NCBI BLAST-2.2) can effectively scale to a number of processors for long sequence query but less scalability for the short sequence query. Using a parallelism scheme, which partitions large input and database to small inputs and databases to form several individual BLAST runs, then gathers the results, a BLAST run can be scaled up to a large number of processors regardless of the length of the sequence query (such as the Perl script BLAST wrapper and the SGI's HT-BLAST) [Camp, Cofer and Gomperts, 1998]. However, the speedup of BLAST is limited to one machine. Therefore, to use BLAST in a clusters-type of computer system, new parallel schemes needed to be developed and implemented. TurboBLAST developed by Turbogenomics, Inc is one such a multi-machine BLAST system.

What is TurboBLAST?

TurboBLAST is a parallel BLAST system designed to run on a cluster of commodity server machines, PC's, or workstations (see the *TurboBLAST User's Guide*). TurboBLAST provides a high performance, remotely accessible BLAST service based on the use of multiple executions of the unmodified NCBI **blastall** program on more than one machines (cluster). TurboBLAST achieves high performance for large numbers of BLAST searches in two ways: by use of batch queuing, and by splitting an individual BLAST search request into multiple parts by dividing up both the set of input (query) sequences and the databases against which the search is to be conducted. In addition, TurboBLAST manages some of the chores of related to updating genomic databases in a multi-machine environment.

TurboBLAST outperforms the NCBI or HT-BLAST (SGI) by searching multiple machines (workers), while achieving similar performance in a single machine with multiple processors. There are three components in TurboBLAST: clients, workers, and a master. The coordination of these three subsystems is outlined in Figure 1. The speedup of the sequence search relies on the number of worker nodes and the number of processors within each node.

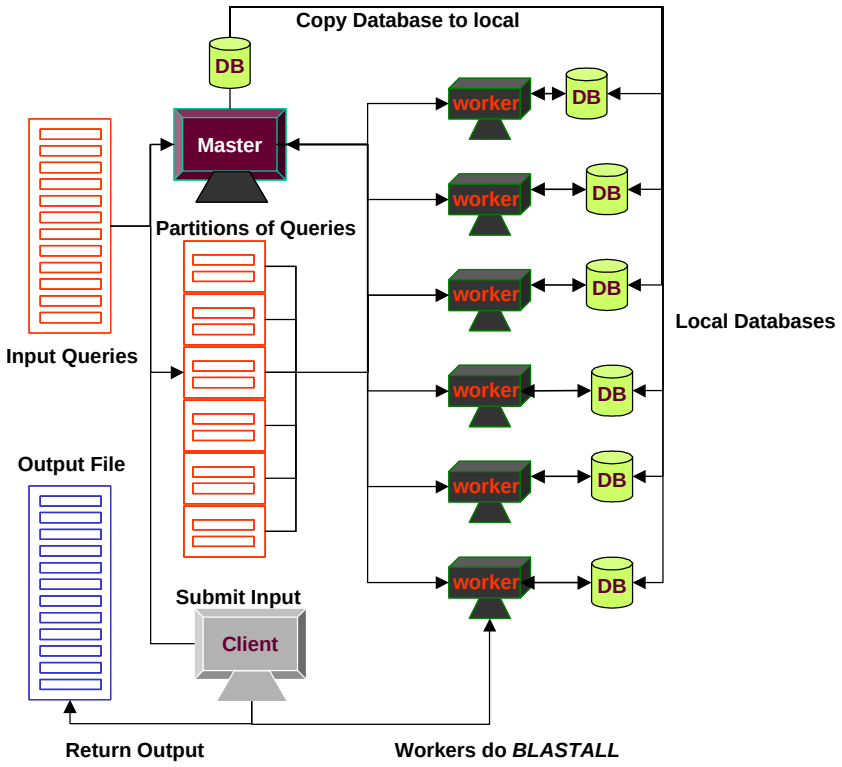

Figure 1. An overview of the three TurboBLAST subsystems. **Master**: coordinates the various parts of the TurboBLAST. A single instance of the Master handles the multiple workers and clients. The coordination is managed by the Paradise program. **Workers**: pick up jobs from master and perform the actual BLAST using standard NCBI BLAST program. **Client(s)**: a user machine submitting **tblastall**. The program **tblastall** is a replacement of **blastall** (NCBI BLAST wrapper). It executes the BLAST search using a collection of workers. The client communicates with the master and retrieves the results from workers.

What is IBM xSeries Server?

IBM xSeries server is an Intel-based platform that includes IBM's X-Architecture™ technology. X-Architecture technology is an evolving blueprint that is drawn from the vast enterprise server heritage of IBM. xSeries engineers take the technologies that have already revolutionized larger IBM systems and bring them to the Intel-based platform. X-Architecture technology pulls together many of the best features of the other systems:

- Availability characteristics achieved by zSeries™
- Scalability of the pSeries™.
- Solution relationships and self-maintaining capabilities (i.e., auto-tuning, auto-configuration) of the iSeries™

The xSeries 330 Server Cluster (Cluster1300) was used in TurboBLAST benchmarking and performance analysis. The hardware and system configurations are described as follows:

System:	Linux lcan (*xSeries 330*)
Processor:	Intel Pentium® III @ 996 MHz
Number of processors:	2 /each node
Memory:	1 GB /each worker node
Network Interface used	Myrinet switch
Operating system:	Red Hat Linux 7.1
Python:	Python 2.1
Java®:	JDK/JRE 1.3.1
Kernel:	2.4.2

TurboBLAST Installation and Configurations

Prerequisite Software Installation

Java:

Java JDK/JRE 1.3.1.2 was downloaded from Sun Microsystem's® Java homepage (http://java.sun.com/j2se). IBM JVM is recommended for the future use.

Python:

Python 2.1 from http://www.python.com was downloaded and installed in the user home directory.

Paradise:

6.2_R was provided by Turbogenomics, Inc. and installed in the same directory as TurboBLAST's.

Installation and Configurations of TurboBLAST

TurboBLAST version 1.1 was installed in directory */vol/lifesci/rchen* on Linux cluster *lcan*. The configuration of the TurboBLAST was modified in the following configuration files:

1). In "jpiranha.properties" file:
set policy.idle=60;
set policy.advance=1.9
set policy.retreat=3.0
set policy.advanceCheck=30
set policy.retreatCheck=5

2). In "tblast.conf" file:
set "worker =0" : # no worker in master machine
set tmpdir = /bench1 # the default /tmp is too small. /bench1 has more than 2GB space
set javaMaxheap = 256 javaMaxheaparg = -Xmx256m.

Program Execution

A number of workers (*tblastworker*) were started after the master (*tblastmaster*) was running. To determine whether the workers were ready or not, the command **listworkers -l** was issued. Once the workers were ready, the client script could be started to execute the **tblastall** search. The client script for this benchmark measurement included **tblastall** such as:

#!/bin/ksh

time python [TBLAST dir]/tblastall ***-b*** *#* ***-v*** *#* ***-p*** *blastn(p)* ***-d*** *database(s)* ***-i*** *inputfile* ***-o*** *output*

where options **-b** and **-v** are described below:
-v: database sequences to show one-line descriptions in output;
-b: Number of database sequences to show alignments in output.
#: the value are either 1 or 25.

Databases

- Human Chromosomes 16 and 19 (hs_chr16.fa and hs_chr19.fa), 72 and 52 MB in size. Downloaded Date: 07-20-2001 from NCBI Genomic database
- Swiss_pro, 21 MB in size, Protein database. Downloaded Date: 02-2001

Input Queries

Extra-long query:
Drosophila gnome sequences (*drosoph.nt*), 2340 queries, more than 10,000 letters/query.
Long query:
Protein sequences, 2275 queries, ~1000 letters/query;
Protein sequences, 792 queries, ~1000 letters/query.
Short query:
Protein sequences, 2293 queries, ~500 letters/query;
Protein sequences, 899 queries, ~500 letters/query;
Protein sequences, 962 queries, 300 to 500 letters/query.

The extra-long query was run against Human Chromosome 16 and 19 databases, while the long and short queries were executed against the Swiss_pro database.

Results

The query sequences were categorized into three types on the basis of the length of the queries in order to better understand the TurboBLAST performance. The query inputs were also classified as large size and medium size which consisted of >2000 queries and <1000 queries, respectively. Inputs with long and short queries were tested with **tblastall** options –**v** and **–b** of values 1 and 25, while extra-long query input was tested with **tblastall –v** and **–b** setting at 1 since the extra-long queries took too much time to complete the benchmark run and the scalability of extra-long query input was less concerned. To test the effect of **tblastall** options **–v** and **–b** on the performance, the **–b** and **–v** values of 100 and 250 were tested. The benchmark results of all the inputs were summarized in Table 1 and 2.

Table 1: Benchmark results of large size input consisting of 2340 extra-long (>10,000) queries and small size input consisting of 962 very short (300 t0 500) queries. A total of 58 workers were used.

No. of Processors	Extra-long Query (2340 queries)	Short Query (medium size input)	
		-b & -v 100	-b & -v 250
	Elapsed Time (Seconds)	Elapsed Time (Seconds)	Elapsed Time (Seconds)
1	60423.00	2011.25	2242.20
2	34020.80	1070.31	1190.96
4	11062.00	563.15	601.49
8	4446.00	308.33	414.02
16	2486.25	263.23	416.80
32	1399.69	251.00	396.75
64	923.04	242.24	393.00
96	629.58	254.19	395.73
116	567.84	242.15	392.80

Table 2: Benchmark results of inputs (2275 and 792 queries, respectively) consisting of long (1000) queries and inputs (2293 and 899 queries, respectively) consisting of short (500) queries. Note that the elapsed time unit is seconds.

No. of Processors	Elapsed Time 48	Elapsed Time 32	Elapsed Time 16	Elapsed Time 8	Elapsed Time 4	Elapsed Time 2	Elapsed Time 1
Large-long (2275)							
-v –b 1	306.32	339.21	588.80	1130.69	2230.3	4439.25	8815.44
-v –b 25	444.97	462.46	688.00	1261.30	2512.68	4939.46	9872.32
Large-short (2293)							
-v –b 1	113.40	143.57	275.73	507.34	992.85	1965.05	3932.86
-v –b 25	146.60	161.39	305.72	558.61	1103.38	2181.58	4369.43
Medium-long (792)							
-v –b 1	164.56	167.31	251.62	429.22	820.82	1570.29	3129.15
-v –b 25	260.56	260.61	296.64	495.41	938.15	1796.04	3610.92
Medium-short (899)							
-v –b 1	56.19	68.01	113.10	206.31	393.09	775.42	1543.79
-v –b 25	64.33	73.97	125.08	224.29	432.12	847.80	1711.48

Performance Analysis

Length of Queries vs. Benchmark Performance

Three types of query input files were prepared and tested, based on the length of the queries. Benchmark results showed that the performance of TurboBLAST on the x330 Linux Sever Cluster was linearly sped up when the query sequences were over 10,000 nucleotides in length and a total of 2340 queries in the input file (**Figure 2**). For inputs containing queries ranging from 1000 residues and 500 residues, the performance scaled up to 32 processors **(Figures 3** and **4**). These results indicated that the TurboBLAST could effectively execute on at least 32 processors, and more effectively used beyond 32 processors for the longer sequence matching. One reason for the poor scaling on the high end of this testing (over 32 CPUs) is that many processors were sitting idle for the shorter queries, as there were fewer tasks than the processors. One solution would be to reduce the tasksize by setting "com.turbogenomics.Turboblast.simpleTaskManager.tasksize" in the configurations.

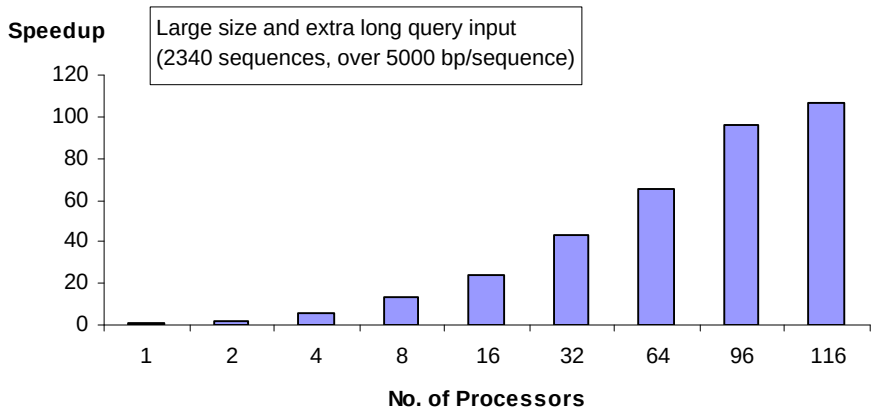

Figure 2. The performance of x330 Linux Server Cluster running the TurboBLAST using the extra-long query input with 2340 queries. The benchmark results show the linear scale up to 116 processors or 58 workers. The **tblastall** was run against two genomic databases and the **–b –v** values were set to 1.

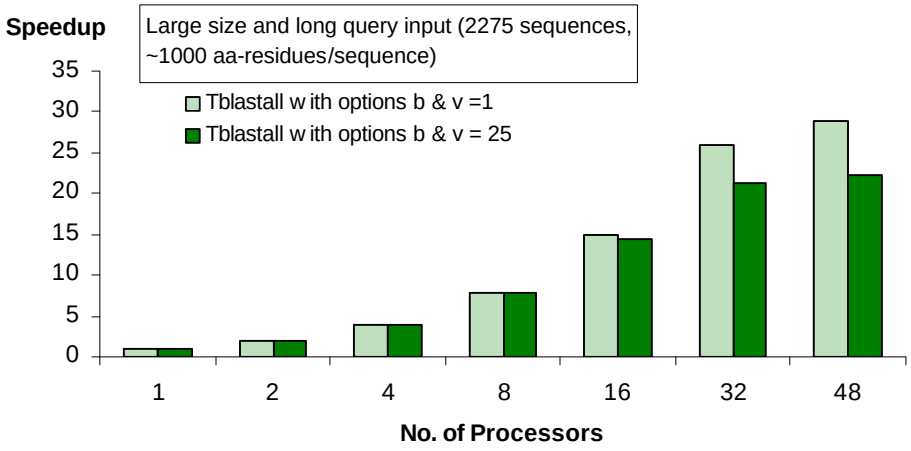

Figure 3. The performance of the x330 Linux Server Cluster running the TurboBLAST using the long query (1000 letters/query) input with 2275 queries. The benchmark results show the linear scalability to 32 processors when the **tblastall** was run against the Swiss_pro database and the **–b –v** values were set to 1 and 25, respectively.

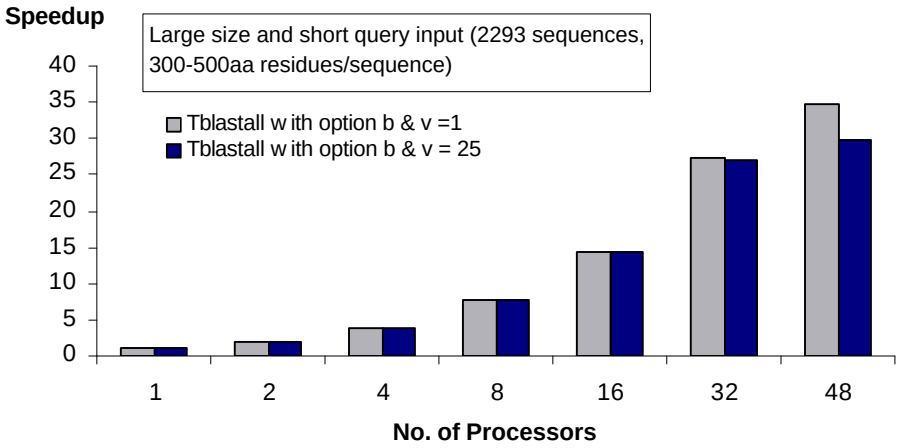

Figure 4. The performance of x330 Linux Server Cluster running the TurboBLAST using the short query (300-500 letters / query) input with 2293 queries. The benchmark results show the linear scalability to 32 processors when the **tblastall** was run against the Swiss_pro database and the **–b –v** values were set to 1 and 25, respectively.

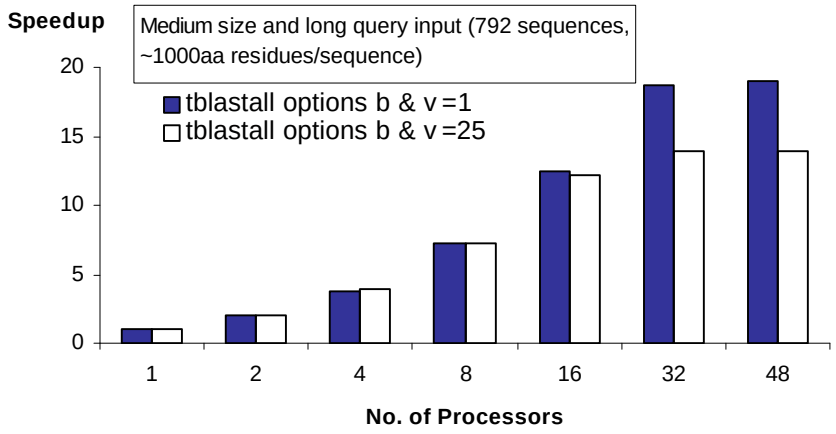

Figure 5. The performance of x330 Linux Server Cluster running the TurboBLAST using the long query (1000 letters / query) input with 792 queries. The benchmark results show the linear scalability to 16 processors when the **tblastall** was run against the Swiss_pro database and the **–b –v** values were set to 1 and 25, respectively.

Numbers of Queries in an Input File vs. Benchmark Performance

The performance of TurboBLAST on the IBM x330 Linux Server Cluster might depend on the numbers of queries in an input file (the size of input). Two inputs, large size (2000 queries) and medium size (1000 queries), were used for benchmarking. The benchmark results were compared

in **Figures 3** to **6**. The speedups for the medium cases were less than those for large cases regardless of the length of the queries (long and short queries). This may be partially due to the increase of overhead along with the increase of processors in the **tblastall** process. More merging time for output was observed when the numbers of processors were included in the process (data recorded in the log file *tblastall.log*).

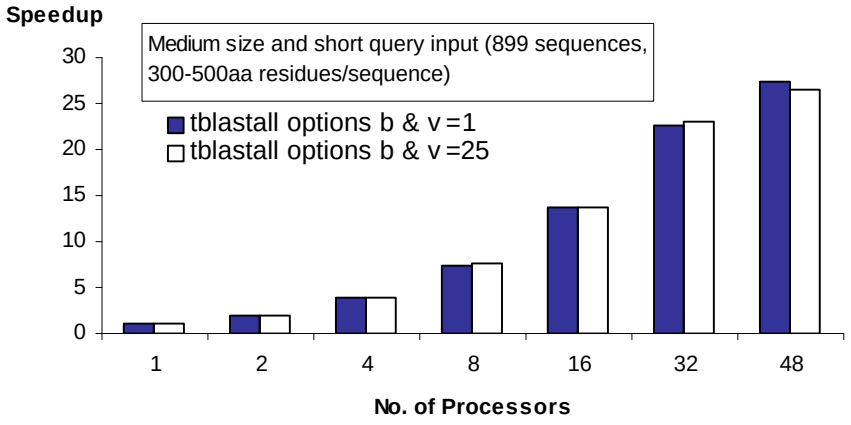

Figure 6. The performance of x330 Linux Server Cluster running the TurboBLAST using the short query (300-500 letters/query) input with 899 queries. The benchmark results show the linear scalability to 16 processors when the **tblastall** was run against the Swiss_pro database and the **–b –v** values were set to 1 and 25, respectively.

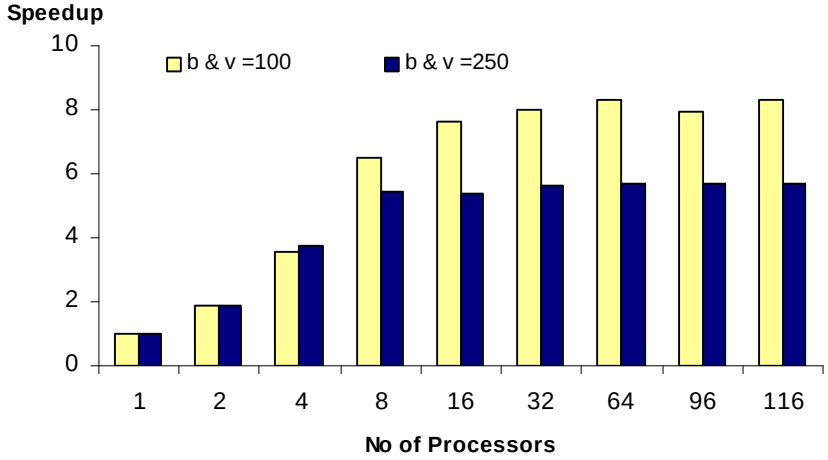

Figure 7. The performance of x330 Linux Server Cluster running the TurboBLAST using the short query (300-500 letters/query) input with 962 queries. The benchmark results show the linear scalability only to 4 processors when the **tblastall** was run against the Swiss_pro database and the **–b –v** values were set to 100 and 250, respectively. The higher **–b –v** values, the worse performance was obtained.

TBLASTALL Options (-b –v) vs. Benchmark Performance

The setting values of **tblastall** options **–b** and **–v** dramatically affected the performance of TurboBLAST on the Linux Server Cluster when their values were set over 100 (**Figure 7**). With **–b** and **–v** values setting to 100 or 250, the scalability could not succeed beyond 4 processors. This is because the overhead time required for the output merging ~~was~~ dominated. Since the **–b** and **–v** values (i.e., the number of database sequences to show alignments and online descriptions) setting over 100 make no sense for the BLAST outputs, it was suggested that these configurations be reduced below 50 in order to achieve the best performance.

Conclusion

The benchmark results have shown that the performance of TurboBLAST can effectively scale on IBM x330 Linux Server Cluster (**Figure 8)**. It is a great advantage that the TurboBLAST system runs on the multiple machines environment, where the current NCBI BLAST and other BLAST programs such as SGI's HT-BLAST could only run on single machine environment. We conclude that the TurboBLAST available on the IBM xSeries Server Cluster could provide definite benefit to genomic and proteomics research.

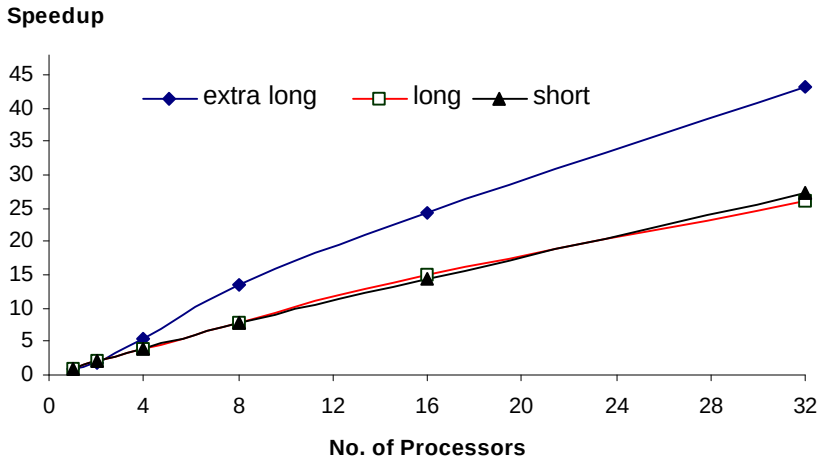

Figure 8. Comparison of performance of a x330 Linux Server Cluster running the TurboBLAST with queries of different length. The benchmark results show the linear scalability to 32 processors when the input size is over 2000 queries. Note that the longer the query is, the better performance the TurboBLAST has.

Acknowledgment

I would like to thank all the benchmark teams in Poughkeepsie, New York who gave us system and technical supports whenever it needed. Usha Reddy from IBM life sciences team helped us to prepare some of the benchmark inputs. Tzy-hwa K Tzeng and others from IBM life sciences and Linux teams make useful discussion and advice.

References

Altschul, S., Gish W., Miller W., Myers EW., and Lipman D., 1990: Basic Local Alignment Search Tool. *Journal of Molecular Biology* 215:403-410.

Camp N., Cofer H., and Gomperts R., 1998: High-Throughput BLAST. *SGI White Paper.* Turbogenomic, Inc: 2001: TurboBLAST User Guide v-1.1. New Haven, CT 06510.

Special Notices

Any performance data contained in this document was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Trademarks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM, e(logo), iSeries, pSeries, X-Architecture, xSeries, zSeries

IBM Trademarks information can be found at: http://www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SPEC, SPECjbb, SPECint, SPECfp, SPECweb, and SPECsfs are trademarks of the Standard Performance Evaluation Corporation and information can be found at: http://www.spec.org.

TPC, TPC-R, TPC-H, TPC-C, and TPC-W are trademarks of the Transaction Processing Performance Council. Information can be found at: http://www.tpc.org.

Other trademarks are the property of their respective owners.

Notes on Benchmarks and Values

The benchmarks and values shown here were derived using particular, well configured, development-level computer systems. Unless otherwise indicated for a system, the values were derived using 32-bit applications and external cache, if external cache is supported on the system. All benchmark values are provided "AS IS" and no warranties or guarantees are expressed or implied by IBM. Actual system performance may vary and is dependent upon many factors including system hardware configuration and software design and configuration. Buyers should consult other sources of information to evaluate the performance of systems they are considering buying and should consider conducting application oriented testing. For additional information about the benchmarks, values and systems tested, contact your local IBM office or IBM authorized reseller or access the following on the Web:

TPC	http://www.tpc.org
GPC	http://www.spec.org/gpc
SPEC	http://www.spec.org
Pro/E	http://www.proe.com
Linpack	http://www.netlib.no/netlib/benchmark/performance.ps
Notesbench Mail	http://www.notesbench.org
VolanoMark	http://www.volano.com
Fluent	http://www.fluent.com

Unless otherwise indicated for a system, the performance benchmarks were conducted using AIX V4.2.1 or 4.3, IBM C Set++ for AIX/6000 V4.1.0.1, and AIX XL FORTRAN V5.1.0.0 with optimization where the compilers were used in the benchmark tests. The preprocessors used in the benchmark tests include KAP 3.2 for FORTRAN and KAP/C 1.4.2 from Kuck & Associates and VAST-2 v4.01X8 from Pacific-Sierra Research. The preprocessors were purchased separately from these vendors.

The following SPEC and Linpack benchmarks reflect the performance of the microprocessor, memory architecture, and compiler of the tested system:

- SPECint95 - SPEC component-level benchmark that measures integer performance. Result is the geometric mean of eight tests that comprise the CINT95 benchmark suite. All of these are written in the C language. SPECint_base95 is the result of the same tests as CINT95 with a maximum of four compiler flags that must be used in all eight tests.

- SPECint_rate95 - Geometric average of the eight SPEC rates from the SPEC integer tests (CINT95). SPECint_base_rate95 is the result of the same tests as CINT95 with a maximum of four compiler flags that must be used in all eight tests.

- SPECfp95 - SPEC component-level benchmark that measures floating-point performance. Result is the geometric mean of ten tests, all written in FORTRAN, that are included in the CFP95 benchmark suite. SPECfp_base95 is the result of the same tests as CFP95 with a maximum of four compiler flags that must be used in all ten tests.

- SPECfp_rate95 - Geometric average of the ten SPEC rates from SPEC floating-point tests (CFP95). SPECfp_base_rate95 is the result of the same tests as CFP95 with a maximum of four compiler flags that must be used in all ten tests.

- SPECint2000 - New SPEC component-level benchmark that measures integer performance. Result is the geometric mean of twelve tests that comprise the CINT2000 benchmark suite. All of these are written in C language except for one which is in C++. SPECint_base2000 is the result of the same tests as CINT2000 with a maximum of four compiler options that must be used in all twelve tests.

- SPECint_rate2000 - Geometric average of the twelve SPEC rates from the SPEC integer tests (CINT2000). SPECint_base_rate2000 is the result of the same tests as CINT2000 with a maximum of four compiler options that must be used in all twelve tests.

- SPECfp2000 - New SPEC component-level benchmark that measures floating-point performance. Result is the geometric mean of fourteen tests, all written in FORTRAN and C languages, that are included in the CFP2000 benchmark suite. SPECfp_base2000 is the result of the same tests as CFP2000 with a maximum of four compiler options that must be used in all fourteen tests.

- SPECfp_rate2000 - Geometric average of the fourteen SPEC rates from SPEC floating-point tests (CFP2000). SPEC_base_rate2000 is the result of the same tests as CFP2000 with a maximum of four compiler options that must be used in all fourteen tests.

- SPECweb96 - Maximum number of Hypertext Transfer Protocol (HTTP) operations per second achieved on the SPECweb96 benchmark without significant degradation of response time. The Web server software is ZEUS v.1.1 from Zeus Technology Ltd.

- SPECweb99 - Number of conforming, simultaneous connections the Web server can support using a predefined workload. The SPECweb99 test harnass emulates clients sending the HTTP requests in the workload over slow Internet connections to the Web server. The Web server softwre is Zeus from Zeus Technology Ltd.

- LINPACK DP (Double Precision) - n=100 is the array size. The results are measured in megaflops (MFLOPS).

- LINPACK SP (Single Precision) - n=100 is the array size. The results are measured in MFLOPS.

- LINPACK TPP (Toward Peak Performance) - n=1,000 is the array size. The results are measured in MFLOPS.

- LINPACK HPC (Highly Parallel Computing) - solve largest system of linear equations possible. The results are measured in GFLOPS.

VolanoMark is a 100% Pure Java server benchmark characterized by long-lasting network connections and high thread counts. In this context, long-lasting means the connections last several minutes or longer, rather than just a few seconds. The VolanoMark benchmark creates client connections in groups of 20 and measures how long it takes for the clients to take turns broadcasting their messages to the group. At the end of the test, it reports a score as the average number of messages transferred by the server per second.

VolanoMark 2.1.2 local performance test measures throughput in messages per second. The final score is the average of the best two out of three results.

The following SPEC benchmark reflects the performance of the microprocessor, memory subsystem, disk subsystem, network subsystem:

- SPECsfs97_R1 - the SPECsfs97_R1 (or SPEC SFS 3.0) benchmark consists of two separate workloads, one for NFS V2 and one for NFS V3, which report two distinct metrics, SPECsfs97_R1.v2 and SPECsfs97_R1.v3, respectively. The metrics consist of a throughput component and an overall response time measure. The throughput (measured in operations per second) is the primary component used when comparing SFS performance between systems. The overall response time (average response time per operation) is a measure of how quickly the server responds to NFS operation requests over the range of tested throughput loads.

The following Transaction Processing Performance Council (TPC) benchmarks reflect the performance of the microprocessor, memory subsystem, disk subsystem, and some portions of the network:

- tpmC - TPC Benchmark C throughput measured as the average number of transactions processed per minute during a valid TPC-C configuration run of at least twenty minutes.

- $/tpmC - TPC Benchmark C price/performance ratio reflects the estimated five year total cost of ownership for system hardware, software, and maintenance and is determined by dividing such estimated total cost by the tpmC for the system.

- QppH is the power metric of TPC-H and is based on a geometric mean of the 17 TPC-H queries, the insert test, and the delete test. It measures the ability of the system to give a single user the best possible response time by harnessing all available resources. QppH is scaled based on database size from 30GB to 1TB.

- QthH is the throughput metric of TPC-H and is a classical throughput measurement characterizing the ability of the system to support a multiuser workload in a balanced way. A number of query users is chosen, each of which must execute the full set of 17 queries in a different order. In the background, there is an update stream running a series of insert/delete operations. QthH is scaled based on the database size from 30GB to 1TB.

- $/QphH is the price/performance metric for the TPC-H benchmark where QphD is the geometric mean of QppH and QthH. The price is the five-year cost of ownership for the tested configuration and includes maintenance and software support.

The following graphics benchmarks reflect the performance of the microprocessor, memory subsystem, and graphics adapter:

- SPECxpc results - Xmark93 is the weighted geometric mean of 447 tests executed in the x11perf suite and is an indicator of 2D graphics performance in an X environment. Larger values indicate better performance.

- SPECplb results (graPHIGS) - PLBwire93 and PLBsurf93 are geometric means of literal and optimized Picture Level Benchmark (PLB) tests for 3D wireframe and 3D surface tests, respectively. The benchmark and tests were developed by the Graphics Performance Characterization (GPC) Committee. The results shown used the graPHIGS API. Larger values indicate better performance.

- SPECopc results - CDRS-03, CDRS-04, DX-03, DX-04, DX-05, DRV-04, DRV-05, DRV-06, Light-01, Light-02, Light-02, AWadvs-01, AWadvs-02, Awadvs-03, and ProCDRS-02 are weighted geometric means of individual viewset metrics. The viewsets were developed by ISVs (independent software vendors) with the assistance of OPC (OpenGL Performance Characterization) member companies. Larger values indicate better performance.

The following graphics benchmarks reflect the performance of the microprocessor, memory subsystem, graphics adapter, and disk subsystem:

Bench95 and Bench97 Pro/E results - Bench95 and Bench97 Pro/E benchmarks have been developed by Texas Instruments to measure UNIX® and Windows NT® workstations in a comparable real-world environment. Results shown are in minutes. Lower numbers indicate better performance.

The NotesBench Mail workload simulates users reading and sending mail. A simulated user will execute a prescribed set of functions 4 times per hour and will generate mail traffic about every 90 minutes. Performance metrics are:

- NotesMark - transactions/minute (TPM).
- NotesBench users - number of client (user) sessions being simulated by the NotesBench workload.
- $/NotesMark - ratio of total system cost divided by the NotesMark (TPM) achieved on the Mail workload.
- $/User - ratio of total system cost divided by the number of client sessions successfully simulated for the Mail NotesBench workload measured.

Total system cost is the price of the server under test to the customer, including hardware, operating system, and Domino Server licenses.