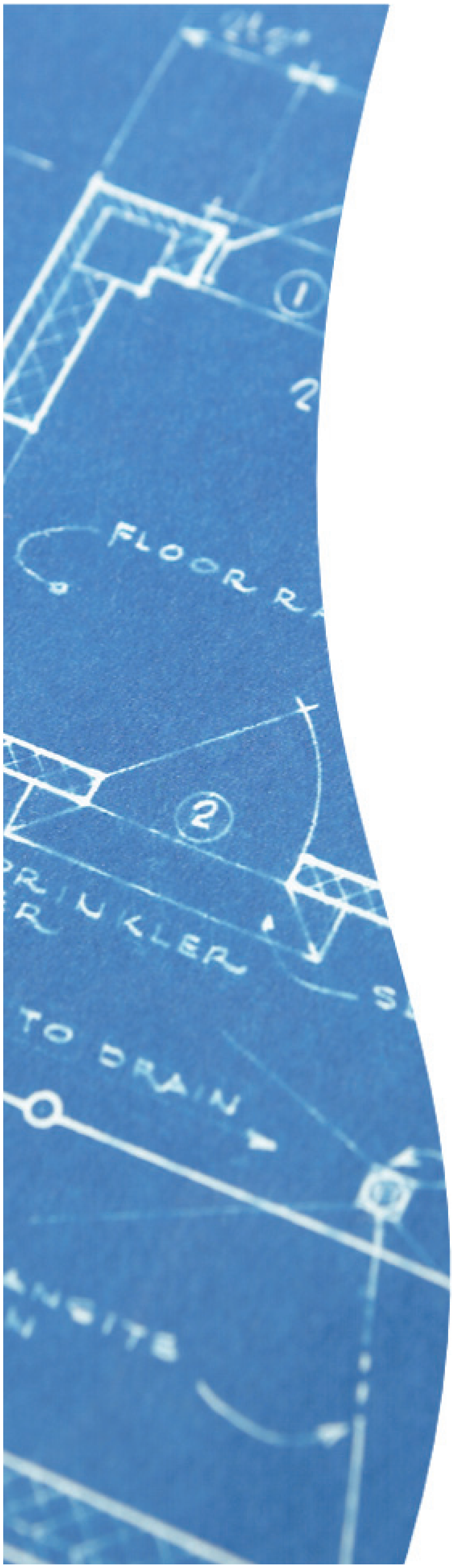

TOKYO INSTITUTE OF TECHNOLOGY SUPERCOMPUTER GRID ARCHITECTURE AND PERFORMANCE OVERVIEW

Nobu Hashizume, Sun Microsystems

Sun BluePrints™ On-Line — February 2007

Part No 820-0831-10
Revision 1.0, 2/19/07
Edition: February 2007

Table of Contents

Chapter 1
Introduction

One of the world's leading technical institutes, the Tokyo Institute of Technology (Tokyo Tech) created the fastest supercomputer in Asia, and one of the largest outside of the United States. Using Sun x64 servers and data servers deployed in a grid architecture, Tokyo Tech built a cost-effective, flexible supercomputer that meets the demands of compute- and data-intensive applications. Built in just 35 days, the TSUBAME grid includes hundreds of systems incorporating thousands of processor cores and terabytes of memory, and delivers 47.38 trillion[1] floating-point operations per second (TeraFLOPS) of sustained LINPACK benchmark performance and 1.1 petabyte of storage to users running common off-the-shelf applications. Based on the deployment architecture, the grid is expected to reach 100 TeraFLOPS in the future.

This Sun BluePrints™ article provides an overview of the Tokyo Tech grid, named TSUBAME. The first in a series of Sun BluePrints articles on the TSUBAME grid, this document discusses the requirements and overall system architecture of the grid, as well as the tuning performed to achieve high LINPACK benchmark performance results.

Note – High performance computing environments, like the TSUBAME grid, constantly grow and change. The latest system configuration information and performance characteristics of the TSUBAME grid can be found on the TOP500 Supercomputer Sites Web site located at *http://www.top500.org*, or the Web site of the Global Scientific Information and Computing Center at the Tokyo Institute of Technology located at *http://www.gsic.titech.ac.jp/index.html.en*

Supercomputing Requirements

Tokyo Tech set out to build the largest, and most flexible, supercomputer in Japan. With a variety of users providing input into the size and functionality of the system, the new supercomputing campus grid infrastructure had several key requirements.

- *Versatile and high performance systems*
  Groups focused on large-scale, high-performance distributed parallel computing, as well as those interested in shared memory computing, required a mix of 32- and 64-bit systems that could run the Linux operating system version 2.4 or later, and be capable of providing over 1,200 SPECint® 2000 (peak) and 1,200 SPECfp® 2000 (peak) performance per CPU, combining for over 20,000 SPECfp®_rate2000 (peak) and 36 TeraFLOPS sustained LINPACK performance across the system. A theoretical peak total CPU performance of 40 TeraFLOPS was

1.TOP500 Supercomputing Sites, November 2006, http://www.top500.org/lists/2006/11 desired. At least half of the servers had to support the x86 instruction set. Each server in the grid had to incorporate at least eight CPUs and 16 GB of shared access memory, with over half the servers incorporating 32 GB of memory, and total grid memory of 5 TB or more. At least one compute server had to contain 64 GB of shared access memory.

- *Distributed parallel computing environment*
  The system had to provide a distributed parallel computing environment with support for the Pthread libraries, Message Passing Interface (MPI) version 1.1 or later, and OpenMP resources and technologies. This environment, combined with additional software tools, had to provide effective job execution management for both interactive and batch jobs.
- *Massive storage capacity*
  With a wide range of researchers throughout the university accessing the system, as well as collaborators all over the world, data storage was a key concern. Over a petabyte of physical storage capacity was required, with a Mean Time Between Failure (MTBF) for data loss across the entire system of 1,000 years, and an average not ready (ANR) of less then 0.4 percent. A parallel file system with a total RAID I/O transfer rate of 5 GB/second was needed to support over 1,000 Network File System (NFS) mount points, or equivalent remote file system capabilities, along with fast parallel file systems like Lustre.
- *Wide accessibility*
  Not content with sheer size, Tokyo Tech was looking to bring supercomputing to everyday use. Unlike traditional, monolithic systems based on proprietary solutions that service the needs of the few, the new supercomputing architecture had to be able to run commercial off-the-shelf and open source applications, including structural analysis applications like ABAQUS and MSC/NASTRAN, computational chemistry tools like Amber and Gaussian, and statistical analysis packages like SAS, Matlab, and Mathematica. In all, over 20 commercial applications from independent software vendors (ISVs), and six open source applications, had to be able to run well on the new system.
- *Short development and deployment cycle*
  Many supercomputing environments are based on proprietary systems, and are designed for dedicated use. Typically, such systems take years to develop and deploy. Tokyo Tech wanted to take a fresh approach and reduce the development and deployment time to months.

Chapter 2
The TSUBAME Supercomputer Grid Architecture

Unlike dedicated supercomputers based on proprietary architectures, the TSUBAME supercomputer grid utilizes standard off-the-shelf hardware components to create a high performance computing (HPC) cluster solution. A single Sun Fire™ x64 system architecture is used across 655 servers to power three different types of clusters within the grid. All systems in the grid are interconnected via InfiniBand technology, and are capable of accessing 1.1 petabyte (PB) of hard disk storage in parallel. Incorporating technology from ClearSpeed Technology, Inc., ClusterFS, and Voltaire, as well as the Sun N1™ System Manager and Sun N1 Grid Engine software, the TSUMBAME grid runs the Linux operating system to deliver applications to users and speed scientific algorithms and data processing.

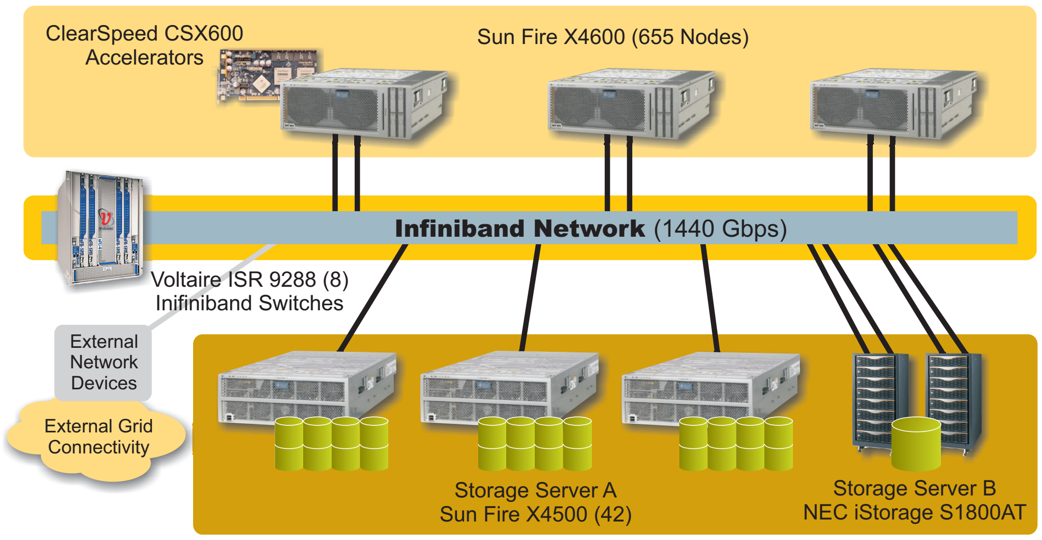
ClearSpeed CSX600 Accelerators
Sun Fire X4600 (655 Nodes)
Infiniband Network (1440 Gbps)
Voltaire ISR 9288 (8) Inifiniband Switches
External Network Devices
External Grid Connectivity
Storage Server A Sun Fire X4500 (42)
Storage Server B NEC iStorage S1800AT

Figure 2-1. The TSUBAME supercomputer grid system architecture

- *Sun Fire X4600 servers*

  In total, the TSUBAME grid consists of 655 Sun Fire X4600 servers running SuSE Linux Enterprise Server 9 SP3 configured into capacity, capability, and shared memory clusters. Together, these systems provide users access to 10,480 high-performance, Next-Generation AMD Opteron processor cores and 21 TB of memory. Each Sun Fire X4600 server incorporates two PCI-Express 4x double data rate (DDR) InfiniBand host adapters for connection to the network.

  The Sun Fire X4600 server is a fast, scalable, and energy efficient 16-way x64 server in a compact 4RU form factor. Indeed, this rackmount server scales from four to eight sockets, simply by adding modular processor boards. This innovative design enables Sun Fire X4600 systems to be upgraded and scaled to next generation processors and memory without disrupting the existing software and network environment. Sun Fire X4600 servers feature single- or dual-core Next-Generation AMD Opteron™ processors, and the ability to natively run the Solaris™ Operating System (OS), Linux, or Windows environments. As a result, these servers give organizations the flexibility to consolidate many applications onto a powerful, industry standard platform. Sun Fire X4600 servers support up to 64 GB of DDR-400 memory with error correcting code (ECC) protection and chipkill mechanisms.

- *Sun Fire X4500 servers*
  Forty-two high-performance Sun Fire X4500 servers running RedHat Enterprise Linux 4 provide storage for the TSUBAME grid. These high density data servers each incorporate 48 direct attached, hot-swappable 500 GB SATA drives, for a total storage capacity of 1 PB. Each Sun Fire X4500 server also includes one PCI-X 4x DDR InfiniBand host adapter.

  The Sun Fire X4500 server integrates high-performance AMD Opteron processors and massive data storage. By integrating these technologies, the Sun Fire X4500 server delivers high storage density and fast throughput rates. These systems deliver four-way x64 server performance and up to 24 TB of direct attached, hot-pluggable SATA disk drives in a 4U form factor, with 1 GB/second throughput from disks to network and 2 GB/second throughput to memory. Sun Fire X4500 servers support up to 16 GB of DDR-400 memory with ECC and chipkill memory technology. Two PCI-X slots and Quad Gigabit Ethernet are provided. All Sun Fire X4500 servers run the RedHat Enterprise Linux 4 operating system, and use standard RedHat Enterprise Linux RAID features to provide software RAID functionality.

- *Voltaire Grid Director ISR9288*
  All Sun Fire X4600 compute systems and Sun Fire X4500 data servers are connected to an InfiniBand network through eight Voltaire Grid Director ISR9288 high-speed InfiniBand switches. Each switch provides 20 Gbps bidirectional bandwidth for up to 288 InfiniBand ports in a single 14U chassis, enabling 1,352 server and storage links. Up to 11.52 Tbps full bisectional switch bandwidth in a fat-tree architecture is possible, with less than 420 nanoseconds of latency between any two ports. As a result, Voltaire ISR9288 switches can be interconnected to form large clusters consisting of thousands of servers.

- *ClearSpeed CSX600*
  In the TSUBAME grid, 360 compute servers are configured with a ClearSpeed Advance accelerator board for added floating-point performance for HPC algorithms. The accelerator board combines two CSX600 processors in a PCI-X form factor and delivers 96 GFlops theoretical peak performance and 50 GFlops sustained double-precision matrix multiply (DGEMM of BLAS) performance while averaging 25 Watts power consumption. Table 2-1 describes the theoretical peak performance of the different server configurations as well as the overall grid architecture.

Table 2-1. TSUBAME supercomputer grid theorectical peak performance

Servers	Quantity	Configuration	Theorectical Peak Performance (TeraFLOPS)
Sun Fire X4600 server	295	No accelerator card	22.7
Sun Fire X4600 server	360	ClearSpeed Advance accelerator card	62.2
Total	655	Mixed	84.9

Network Architecture

The InfiniBand connectivity schema is designed to provide the TSUBAME grid with an optimum and balanced network load, maximum availability, and high performance. All Sun Fire X4500 and Sun Fire X4600 servers are connected to one of the six edge InfiniBand switches. These six switches are in turn connected to two Voltaire Grid Director ISR9288 core switches (Figure 2-2). There are 24 links between each pair of edge and core switches, resulting in a blocking ratio of 5:1 and a maximum of nine hops between any two nodes. Multiple paths are available through the core switch, fostering high availability.

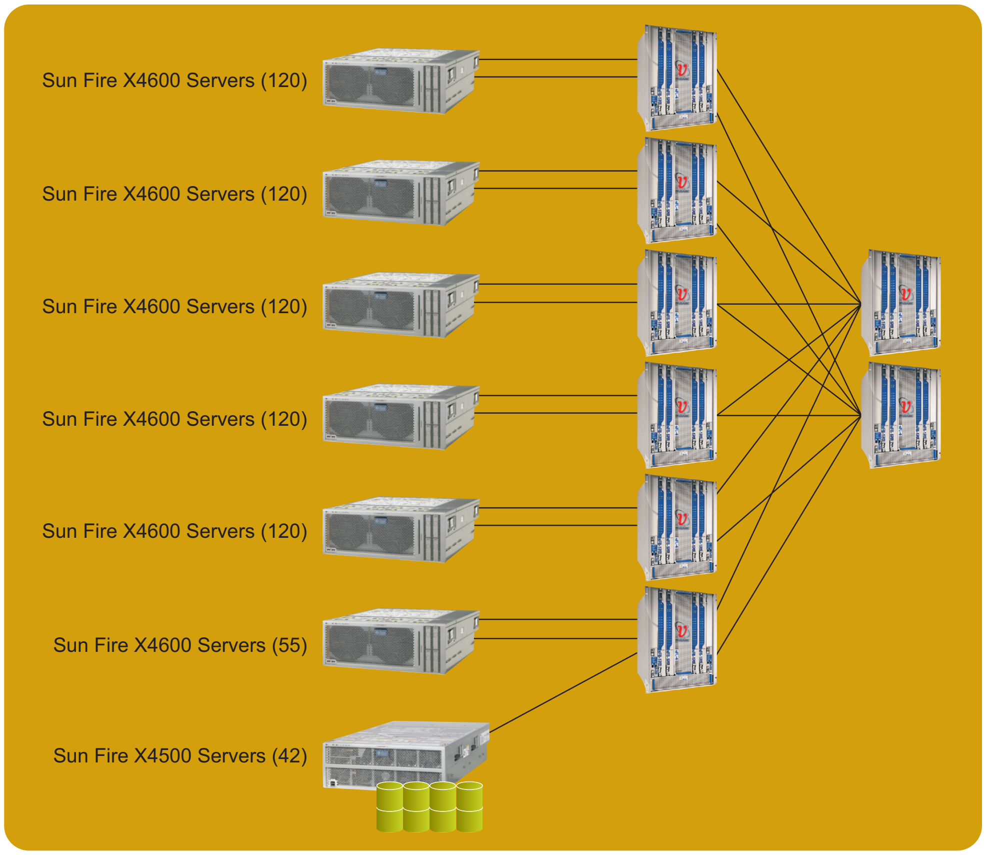
Sun Fire X4600 Servers (120)
Sun Fire X4600 Servers (120)
Sun Fire X4600 Servers (120)
Sun Fire X4600 Servers (120)
Sun Fire X4600 Servers (120)
Sun Fire X4600 Servers (55)
Sun Fire X4500 Servers (42)

Figure 2-2. The TSUBAME supercomputer grid network architecture

Each Sun Fire X4600 server incorporates two PCI-Express 4x double data rate (DDR) InfiniBand host adapters for connection to the network. Both connections are utilized to avoid congestion in the InfiniBand fabric and speed communication for large remote direct memory access (RDMA) messages. It is important to note that each InfiniBand host adapter has a limited amount of on-board memory for Queue Pair (QP) resources. While a single host adapter can provide up to 8,000 QPs, a single MPI program that utilizes all Sun Fire X4600 compute servers can result in over 10,000 MPI jobs on the grid. A single host adapter can provide suitable I/O performance for up to eight CPU cores. However, two or more host adapters are needed to satisfy bandwidth requirements for large symmetric multiprocessing (SMP) servers with more than eight cores. As a result, utilizing two host adapters in each server decreases InfiniBand fabric congestion.

In addition, it is possible for many routes to use the same port. For example, Voltaire MPI uses large RDMA messages for inter-node communication. Using only a single host adapter in a server could affect inter-node bandwidth performance.

Server and Switch Connectivity

Each InfiniBand host adapter installed in Sun Fire X4600 compute servers is connected to a different switch line board (sLB). As a result, each link is connected to one of the 24 InfiniScale-III chips on the 12 line board module (two chips per module), providing optimum edge switch distribution (Figure 2-3).

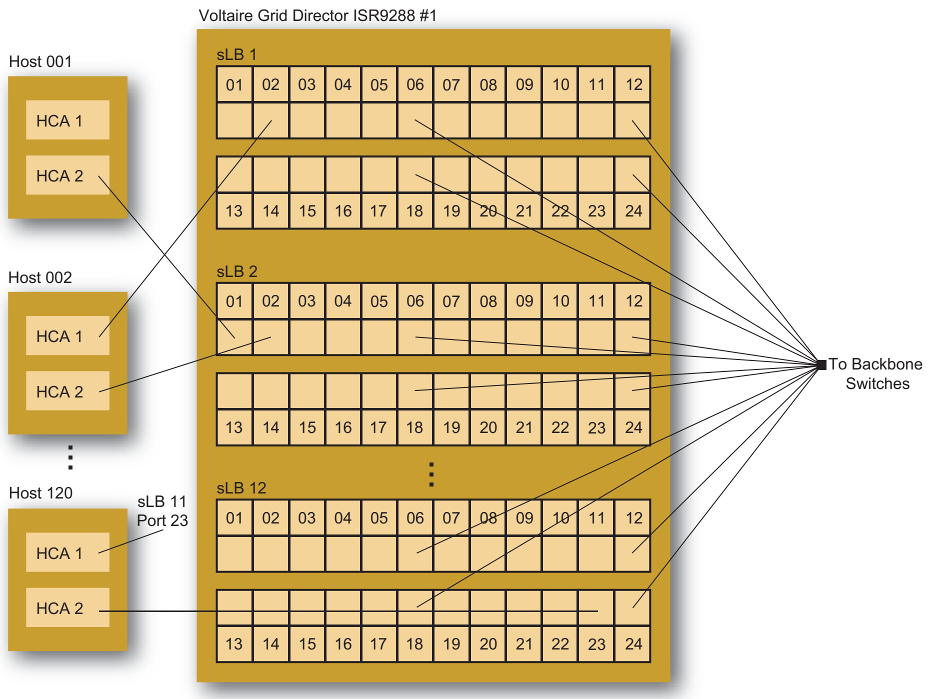

Figure 2-3. InfiniBand connections between servers in the grid and the edge switches

All edge switches are connected to two core, or backbone, switches. Two connections between each edge switch and each core switch provide multiple connectivity paths. This redundancy fosters greater sLB module availability and performance (Figure 2-4).

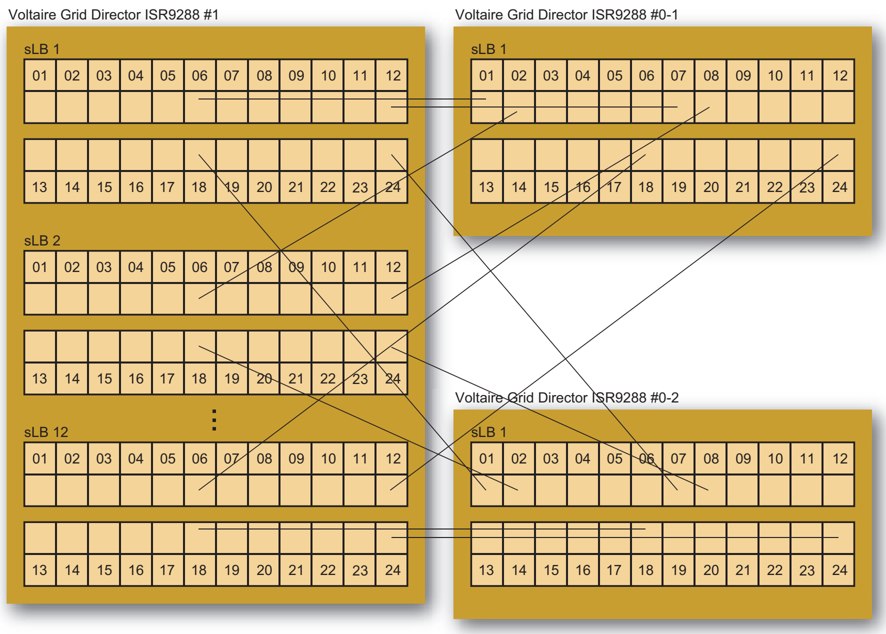
Voltaire Grid Director ISR9288 #1
Voltaire Grid Director ISR9288 #0-1
Voltaire Grid Director ISR9288 #0-2
sLB 1
sLB 2
sLB 12
sLB 1
sLB 1
01 02 03 04 05 06 07 08 09 10 11 12
13 14 15 16 17 18 19 20 21 22 23 24

Figure 2-4. InfiniBand connections between the edge and core switches

Software Architecture

The TSUBAME grid software architecture includes a wide variety of software packages that run on the compute and data servers and work together to make the grid widely accessible to users.

Compute Server Software

All Sun Fire X4600 servers in the TSUBAME grid run the SuSE Linux Enterprise Server 9 SP3 environment, as well as several software tools that aid grid deployment and operation (Figure 2-5).

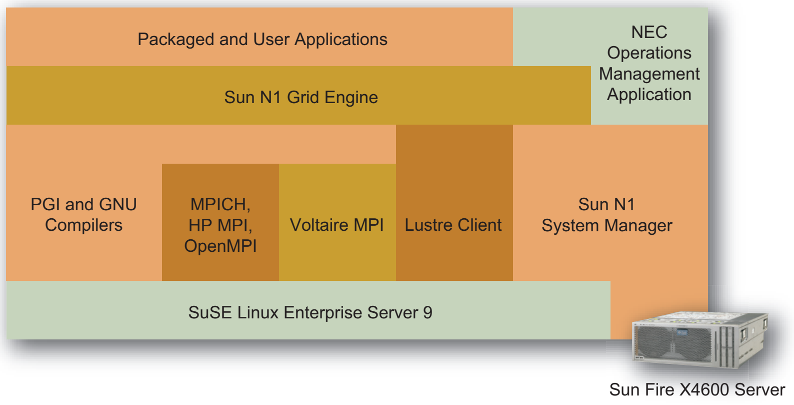

Figure 2-5. The software stack that runs on all Sun Fire X4600 compute servers in the TSUBAME grid

- *Sun N1 Grid Engine 6.0 software*

  Unlike most dedicated cluster environments, the TSUBAME grid was designed to make vast computing and storage resources easily accessible to a wide range of users running off-the-shelf applications. The Sun N1 Grid Engine software makes this possible by managing how jobs are allocated to systems in the grid — without users needing to know the underlying details of where jobs run.

  The Sun N1 Grid Engine software provides policy-based workload management and dynamic provisioning of application workloads. The software is used to create a grid of networked systems and give users access to these compute resources. In the TSUBAME grid, an instance of the software runs on a Sun Fire X4100 management server and directs job traffic to Sun N1 Grid Engine execution nodes running on Sun Fire X4600 compute servers within the grid. Computing tasks and jobs are distributed throughout the grid to Sun Fire X4600 compute servers based on resource requirements, user requests, and administrative and managerial policies. Usage accounting data is accumulated and stored to enable Tokyo Tech to determine which resources were used for a particular job and user. The administrative and management interfaces provided by the software are used to take care of the grid.

  By using the Sun N1 Grid Engine software, the physical systems that comprise the TSUBAME grid can be viewed logically (Figure 2-6). Users log in to the grid via login nodes that are load balanced via a round robin policy. Sessions are then transferred to an interactive node by the Sun N1 Grid Engine software. It is on these interactive servers that users create and submit jobs, and compile and run applications. Batch nodes are available for batch job processing, as well as applications like Matlab and Mathematica. Jobs from these systems run only on execution nodes. Management nodes take care of licensing, backup and restore operations, high availability network file system (HA-NFS) access, resource accounting, and more.

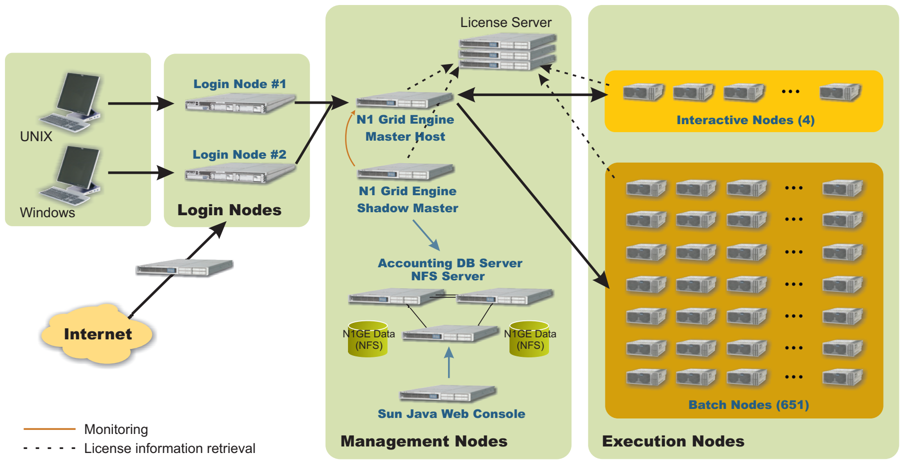

Figure 2-6. The Sun N1 Grid Engine software manages how jobs are distributed throughout the grid

- *Sun N1 System Manager 1.3 software*
  The TSUBAME grid environment utilizes a large number of horizontally scalable servers to generate massive compute power. As the number of systems in the grid grows, the task of managing the system infrastructure throughout its life cycle, and ensuring systems perform at desired levels, rises in complexity. To help this effort, the TSUBAME grid utilizes the Sun N1 System Manager software to handle life cycle management functions. With this software, administrators discover, provision, monitor, update, and manage the hundreds of Sun servers in the TSUBAME grid from a single console over a dedicated 100 Mbps Ethernet network. Note that data and applications do not utilize the Ethernet network at any time.

- *Lustre File System client software*
  Lustre File System client software provides access to the parallel file systems running on Sun Fire X4500 data servers distributed throughout the TSUBAME grid architecture. With this software, the compute servers in the grid are able to communicate with storage systems in parallel, helping to speed the amount of scientific data being processed.

- *Operations management application*
  NEC developed a custom management application for the TSUBAME grid that provides Network File System (NFS) access to user home directories located on NEC iStorage S1800AT systems with approximately 96 TB of storage using RAID-6 technology. The software gives administrators the ability to add, modify, and delete users, and calculate charges for time used based on the accounting log of the Sun N1 Grid Engine software.

- *Developer tools*
  Distributed-memory programming paradigms enable developers to attain peak performance from applications and use clusters as high performance computing platforms. Typically implemented using the *messaging-passing* model, distributed-memory applications perform well when designed with the clustered environment in mind. Programmers need to design applications to take advantage of multiple nodes, coordinate the data between them, utilize message-passing paradigms, and debug distributed code—tasks and techniques that ensure applications attain peak performance.

  As applications move to the grid, they may need modification. To help this effort, a variety of Message Passing Interface (MPI) tools are installed to aid custom and commercial application portability, such as Voltaire MPI, MPICH, OpenMPI, and HP-MPI. Some of these tools utilize the IP over InfiniBand (IPoIB) protocol rather than native InfiniBand protocols. In addition, Voltaire IBHOST enables applications to employ MPI communication over the InfiniBand network. Based on MVAPICH, the Voltaire implementation includes several enhancements for the TSUBAME grid, including support for two InfiniBand host adapters in a single system, shared receive queue, and adaptive FASTPATH.

Portland Group (PGI) 6.1 and GNU (gcc) scalar and parallel compilers are installed on all compute nodes in the cluster for access by developers.

Data Server Software

All data in the TSUBAME grid is stored on 42 Sun Fire X4500 data servers that run the RedHat Enterprise Linux 4 operating environment and are distributed throughout the grid architecture. Access to data stored on these systems is provided by the Lustre File system, an object-based file system designed for high-performance and availability. In the TSUBAME grid, the Lustre File System uses storage devices to manage data objects that are distributed among Sun Fire X4500 data servers in the grid architecture. File system metadata is stored on other servers, enabling routine file operations to be performed without impacting data storage operations. Mirrored copies of a transaction journal, including changes to file system metadata, ensure data availability and integrity in the event of a hardware or software failure.

Because Sun Fire X4500 servers do not include a hardware RAID controller, software RAID technology is typically made available via the software RAID capabilities of the Linux operating system. The TSUBAME grid uses these Linux RAID capabilities on all Sun Fire X4500 servers, with all servers aggregated into parallel I/O by the Lustre File System.

All Sun Fire X4500 servers in the TSUBAME grid also run the Sun N1 System Manager software.

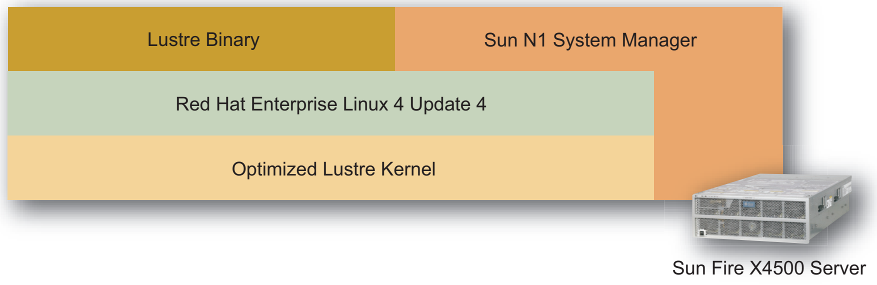

Figure 2-7. The software stack that runs on all Sun Fire X4500 data servers in the TSUBAME grid

Chapter 3
Performance Overview

Tokyo Tech required the new supercomputer grid to provide at least 36.29 TeraFLOPS. With the Sun solution, the TSUBAME debuted with a performance of 38.18 TeraFLOPS on LINPACK performance as measured by the TOP500 Supercomputers Site[1]. This chapter describes the tuning performed to achieve these results.

High Performance LINPACK Configuration and Tuning

LINPACK is a collection of linear algebra subroutines to perform matrix factorization, solving linear systems of various forms, solutions to least-squares problems, and other common linear algebra operations. High Performance LINPACK (HPL) is a version of LINPACK that utilizes the Message Passing Interface to solve a dense system of linear equations using double-precision (64-bit) arithmetic. HPL was used to test the performance of the TSUBAME supercomputer grid. HPL requires access to the Message Passing Interface, as well as the Basic Linear Algebra Subprograms (BLAS) library, to perform tasks.

More information on HPL, including download capabilities, can be found at *http://www.netlib.org/linpack*.

GotoBLAS Library

The Basic Linear Algebra Subprograms (BLAS) library is a set of routines for performing scalar, vector, vector-vector, matrix-vector, and matrix-matrix operations. While many BLAS implementations are available, the fast GotoBLAS library was used to test the LINPACK performance of the TSUBAME grid. The standard GotoBLAS library creates up to two threads for a given process. Since Sun Fire X4600 servers include 16 processor cores, it was believed the ability to create 16 threads for each process could help reduce MPI communication between nodes and improve LINPACK performance. As a result, Kazushige Goto provided a modified version of the GotoBLAS library that is capable of creating 16 threads.

In addition, a version of the library that uses `HUGETLB(HUGEPAGE)` in the `dgemm()` routine was also provided, as its use can help reduce translation look-aside buffer (TLB) cache misses. Initial testing efforts focused on trying to use `HUGETLB` for the main matrix of the HPL test. However, use of this parameter for the MPI send and receive buffers interacted with the Linux kernel and drivers, hampering its use. As a result, the

1.TOP500 Supercomputing Sites, June 2006, *http://www.top500.org/lists/2006/06*. Higher performance have already been achieved. See the TOP500 Supercomting Sites Web site located at *http://www.top500.org*, or the Web site of the Global Scientific Information and Computing Center at the Tokyo Institute of Technology at *http://www.gsic.titech.ac.jp/index.html.en* for the latest configuration and perfomance characteristics of the TSUBAME grid.

GotoBLAS library utilized in this benchmarking effort only uses `HUGETLB` for the computational buffer in the `dgemm()` routine. Because this routine requires 32 MB of memory for each thread, the following Linux system settings were used.

1. Set the `HUGEPAGES` parameter in the */boot/grub/menu.lst* file as a boot parameter.

```
hugepages=300
```

2. Set the maximum size of a shared memory segment to 3,355,443,200 bytes.

```
# echo 3355443200 > /proc/sys/kernel/shmmax
```

3. Set the maximum number of shared memory pages for the system to 3,355,443,200.

```
# echo 3355443200 > /proc/sys/kernel/shmall
```

Message Passing Interface

Voltaire IBHOST 3.5.5 was used to enable applications to use MPI communication over the InfiniBand network. Based on MVAPICH, Voltaire IBHOST includes several enhancements for the TSUBAME grid:

- Support for two InfiniBand host adapters in a single system
- Awareness of CPU and memory affinity in the MPI library
- Shared receive queue
- Adaptive FASTPATH
- Logging functionality

High Performance LINPACK Block Size

HPL includes a parameter, `NB`, that indicates the block size for data distribution and computational granularity. In general, a small `NB` value provides a good load balance for data distribution. However, a larger `NB` value can help computational performance. In addition, the `NB` value depends on the block size of the `dgemm()` routine, making the right value dependent on the implementation of the BLAS library in use.

The GotoBLAS library is optimized for a block size of 256 in the `dgemm()` routine, suggesting that setting `NB` to 256 should optimize HPL performance. In practice, message passing overhead, internal `dgemm()` routine blocking, and other factors impact performance, and using the theoretical optimum value does not yield the best results. Figure 3-1 describes HPL performance on the TSUBAME grid using several `NB` values. All tests were conducted on one, two, and four Sun Fire X4600 servers with eight 2.4 GHz AMD Opteron processors and 32 GB of memory. All CPU cores in the servers were used in each test. Table 3-1 summarizes the `N` values for the systems tested.

Table 3-1. HPL N value settings

Number of Servers	Configuration
1	46080
2	65520
4	92640

As illustrated in Figure 3-1, an NB value of 240 produced the fastest result on configurations with one, two, and four Sun Fire X4600 servers.

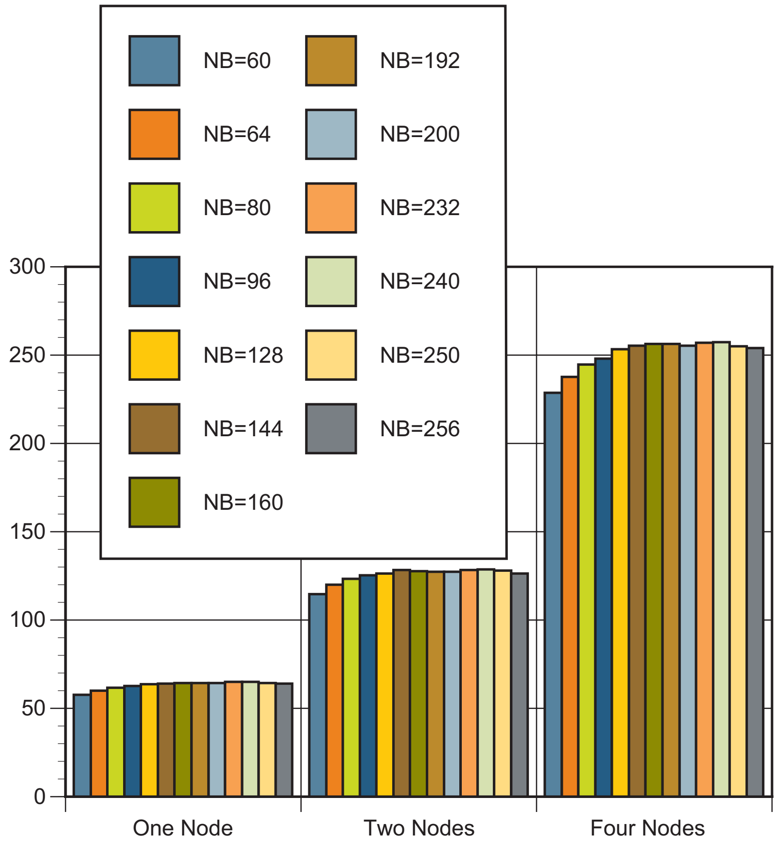

Figure 3-1. TSUBAME grid HPL performance results using different NB values

NUMA Awareness

The Sun Fire X4600 server employs a fat symmetric multiprocessing (SMP), non-uniform memory access (NUMA) system architecture (Figure 3-2). In such an architecture, CPU and memory affinity is a key factor for improving performance. As can been seen in the Hypertransport routing in the system architecture diagram, a worst case scenario requires three Hypertransport hops to access the memory of the other CPU. By default, the SuSE Linux Enterprise Server 9 environment pays no attention to Hypertransport routing, and assign physical memory from CPU 0 to CPU 7 in numerical order, causing the worst possible memory access conditions.

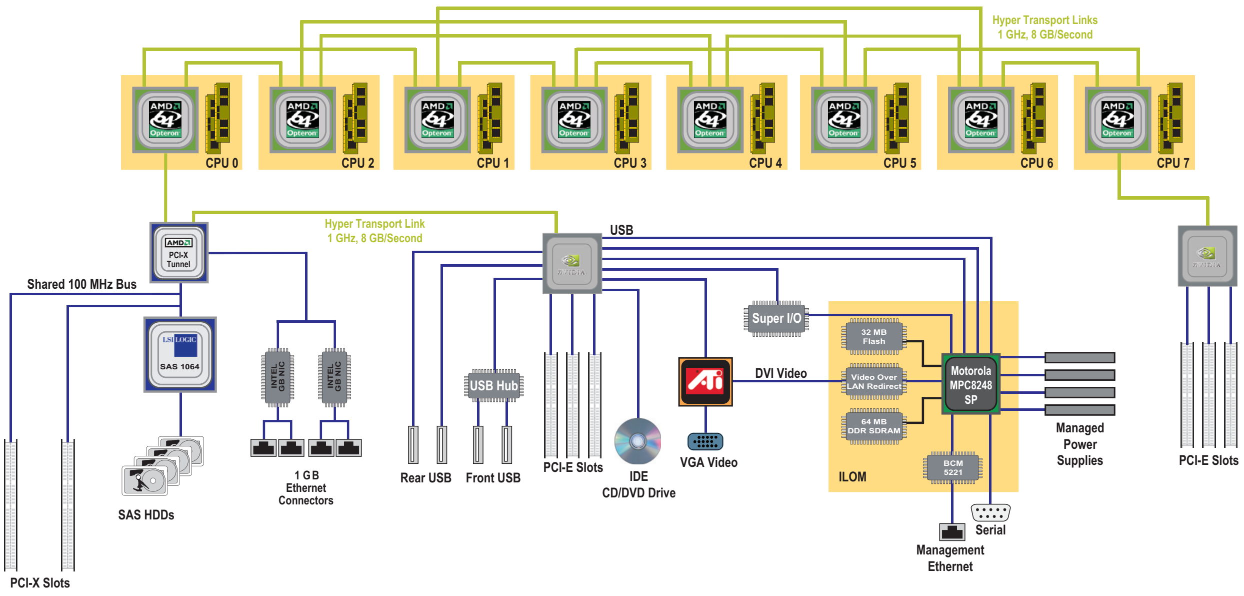

Figure 3-2. The Sun Fire X4600 system architecture

Assigning processes and threads to CPUs so that the hop ring for Hypertransport 1 utilizes CPUs 0, 2, 4, 6, 7, 5, 3, 1 (in that order) helps achieve good memory locality. The Linux `numactl` or `taskset` command can be used for this purpose. In HPL, main matrix memory is allocated by the main program thread in the local memory of the CPU to which each process is bound. HPL does not access the memory of other CPUs. However, it is possible for every computational thread to access distant memory even if CPU affinity is employed.

As a result, it is important to use the `numactl` or `taskset` command to configure CPU and memory affinity for applications which are CPU or memory intensive. Figure 3-3 illustrates the impact of using the `taskset` command to invoke HPL processes. All tests were conducted on one, two, and four Sun Fire X4600 servers with eight 2.4 GHz AMD Opteron processors and 32 GB of memory. All CPU cores in the servers were used in each test. Table 3-2 summarizes the `N` values for the systems tested.

Table 3-2. HPL `N` value settings

Number of Servers	Configuration
1	46080
2	65520
4	92640

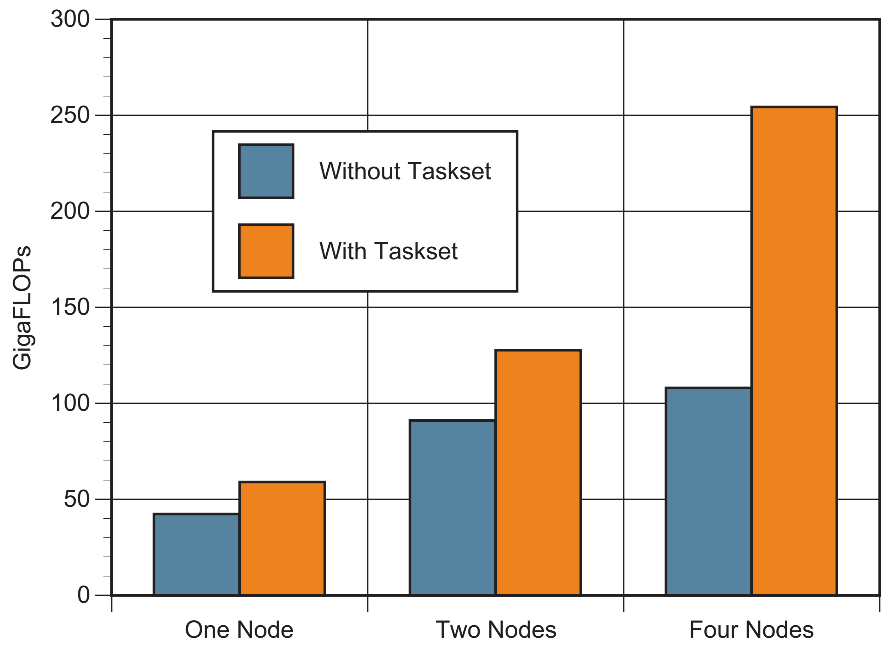

Figure 3-3. Performance results with, and without, using the taskset command

In addition, it is important to consider the combination of processes and threads when running HPL on Sun Fire X4600 servers. The GotoBLAS library can create worker threads by setting an environment variable. A variety of factors affect the choice for the number of threads for each HPL process. A high thread count help reduce the amount of message passing related to MPI communication, as more threads reduces the number of HPL processes on each server. If 16 threads are used on a given Sun Fire X4600 server, no message passing should occur within the node. As a result, theoretically one process per node with 16 threads in the process should result in optimal HPL performance in the TSUBAME cluster configuration. In practice, eight processes per node with two threads per process produced the best HPL performance.

For TSUBAME grid benchmark testing, the `taskset` command was used to set CPU and memory affinity for HPL execution, as shown the script below. This script specifies two threads per process configuration, and can be modified for other thread values. The script is invoked by specifying the number of threads:

```
# mpirun_ssh -np 16 -hostfile hostfile ./xrun
# !/bin/ksh
cd /home/hashi/hpl/bin/G4
export OMP_NUM_THREADS=2
export GOTO_NUM_THREADS=$OMP_NUM_THREADS
rank =‘expr $MPIRUN_RANK % 8‘
case $RANK in
0)
cpulist=”4,5”
;;
1)
cpulist=”8,9”
;;
2)
cpulist=”12,13”
’’
3)
cpulist=”14,15”
;;
4)
cpulist=”10,11”
;;
5)
cpulist=”6,7”
;;
6)
cpulist=”2,3”
;;
7)
cpulist=”0,1”
;;
esac
taskset -c $cpulist ./xhpl $*
```

Figure 3-4 describes HPL performance on the TSUBAME grid using several process and thread combinations. All tests were conducted on one, two, and four Sun Fire X4600 servers with eight 2.4 GHz AMD Opteron processors and 32 GB of memory. All CPU cores in the servers were used in each test. Table 3-3 summarizes the `N` values for the systems tested.

Table 3-3. HPL N value settings

Number of Servers	Configuration
1	46080
2	65520
4	92640

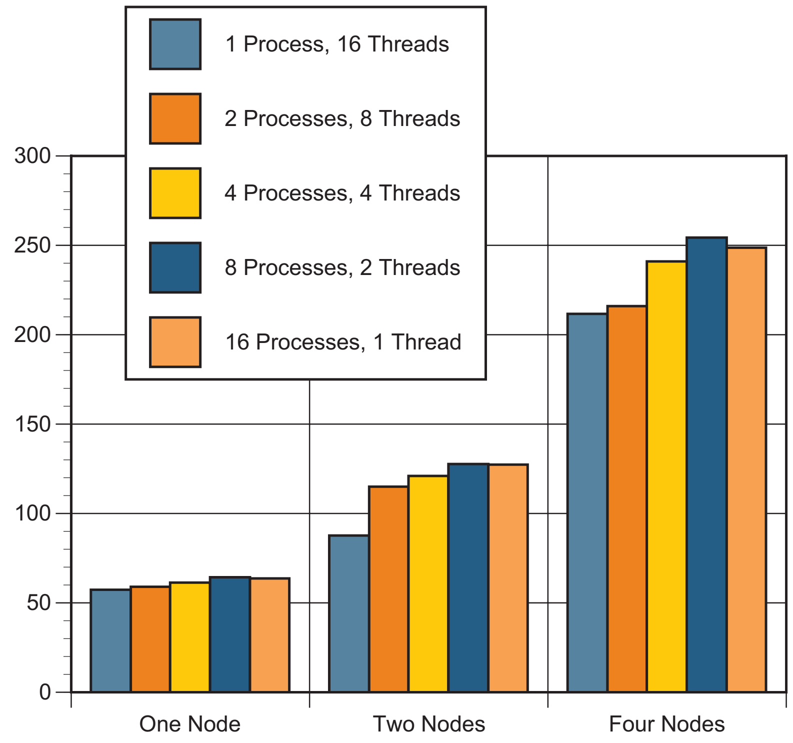

Figure 3-4. Process versus threads comparison

High Performance LINPACK Panel Broadcast

Once the panel factorization is computed in HPL, the panel is broadcast to the other process columns as shown in Figure 3-5.

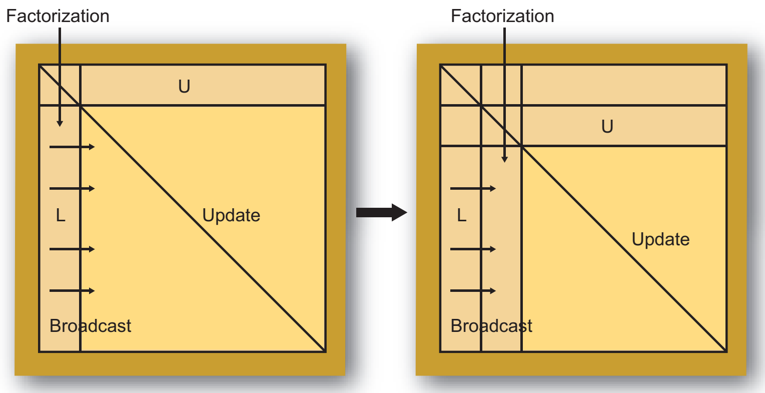

Figure 3-5. HPL LU factorization

HPL provides six broadcast algorithms. Benchmark tests show the `Increasing-2-ring-modified` algorithm produces better performance than other broadcast algorithms on several HPC cluster configurations. Figure 3-6 depicts the performance characteristics of the six HPL broadcast algorithms on one, two, and four Sun Fire X4600 servers configured with eight 2.4 GHz AMD Opteron processors and 32 GB of memory. All CPU cores in the servers were used in each test, and the `taskset` command was used to set two threads per process. Table 3-4 summarizes the `N` values for the systems tested.

Table 3-4. HPL `N` value settings

Number of Servers	Configuration
1	46080
2	65520
4	92640

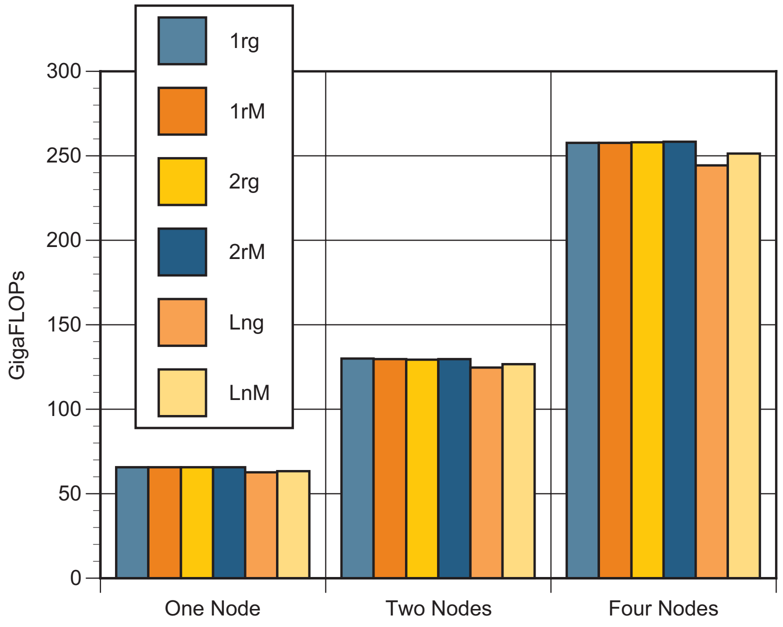

Figure 3-6. HPL broadcast algorithm performance comparison

As illustrated in Figure 3-6, there is appreciable difference between the `Increasing-ring` (`1rg`), `Increasing-ring-modified` (`1rM`), `Increasing-2-ring` (`2rg`), and `Increasing-2-ring-modified` (`2rM`) algorithms. In fact, the `Increasing-2-ring-modified` algorithm was used to measure up to 512 nodes, and the `Increasing-ring` and `Increasing-ring-modified` algorithms were reconfirmed on the entire 648 nodes in the TSUBAME grid. As is evidenced in Figure 3-7, the `Increasing-ring` algorithm produced the fastest results. Theoretically, the `Increasing-ring-modified` algorithm reduces the broadcast as compared to the `Increasing-ring` algorithm, however the `Increasing-ring` algorithm can balance the broadcast data transfer on a large cluster configuration with a fat edge switch, as is deployed in the TSUBAME grid.

Figure 3-7 details the performance results of the three broadcast algorithms on the entire 648 Sun Fire X4600 compute servers in the TSUBAME grid. All CPU cores in the servers were used in each test, and the `taskset()` command was used to set two threads per process. The `N` value was set to 441360 in all tests.

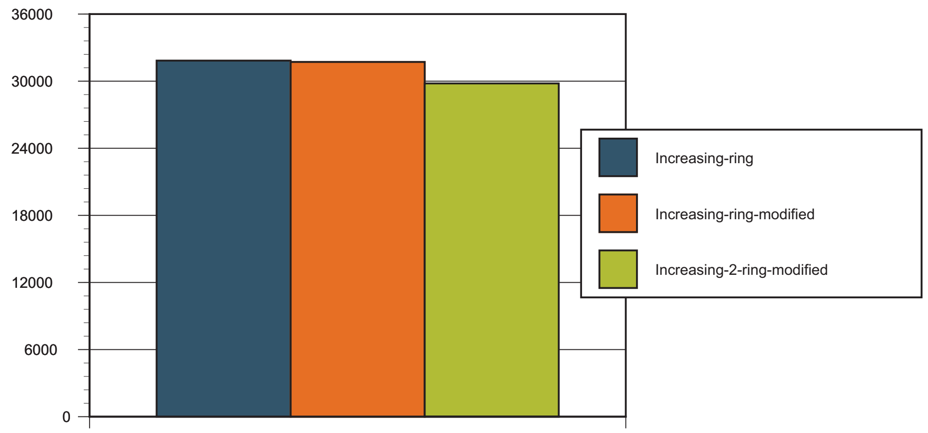

Figure 3-7. HPL broadcast algorithm performance comparison

High Performance LINPACK PxQ Settings

In HPL, data is distributed onto a two dimensional PxQ grid of processes. First, the HPL NxN matrix is logically partitioned into NBxNB blocks as depicted in Figure 3-8.

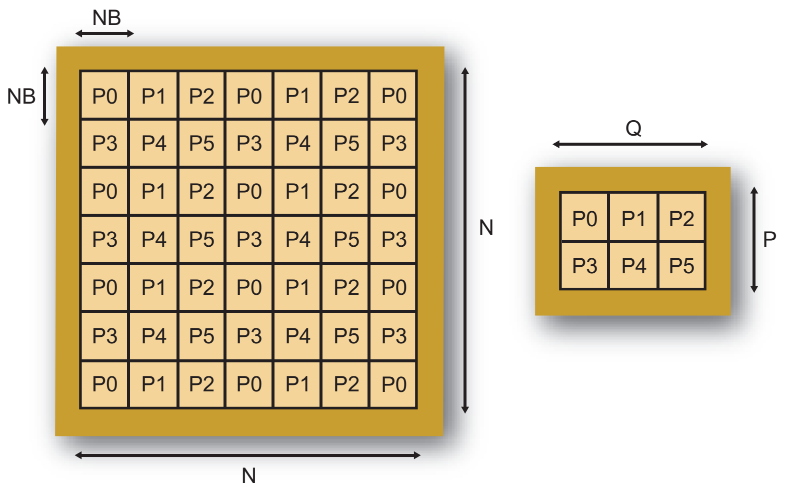

Figure 3-8. HPL two-dimensional block-cycle partition

In theory, the best performance should be achieved when P and Q are approximately equal, with Q having a slightly higher value than P. When the number of processors available is a multiple of 32, this practice can be applied. However, the success of this approach can depend on the physical interconnect network as well. The TSUBAME grid contains 655 servers with 10,480 processor cores. Therefore, PxQ ideally should be 40x131 with two threads per HPL process when using all servers in the grid for LINPACK testing. However, a PxQ value of 40x131 causes congestion on the interconnect network, impacting communication. However, when 648 servers incorporating 10,368 processor cores is used instead, a PxQ value of 72x72 or 36x144 with two threads per HPL process can be used to achieve better results. Indeed, these values provide a good balance for HPL partitioning and computation. Figure 3-9 depicts the performance results for several PxQ settings on 648 Sun Fire X4600 servers with a smaller HPL N value (441360) to reduce execution time. While theory suggests a PxQ value of 72x72 or 64x81 should produce a superior result, actual testing revealed the best performance was achieved with a PxQ value of 36x144.

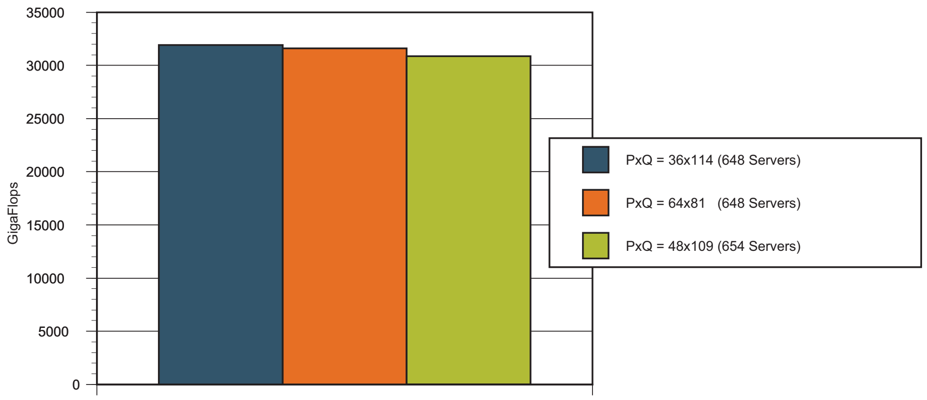

Figure 3-9. Performance comparison of PxQ settings on 648 servers in the grid

Note a test using the PxQ value of 72x72 is not included in Figure 3-9, as a PxQ value of 36x144 performs better than other settings on 512 servers (Figure 3-10). In these tests, the HPL N value was 1048320, larger than the value used for the tests described by Figure 3-9. In addition, approximately 50 percent of memory was used during the performance of these tests.

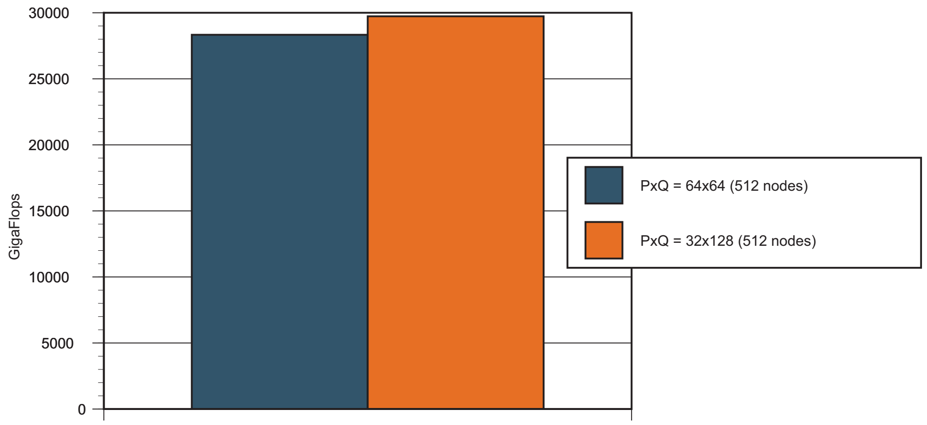

Figure 3-10. Performance comparison of PxQ settings on 512 servers in the grid

Keep in mind the TSUBAME grid utilizes a fat InfiniBand edge switch configuration, and fat compute servers. In this architecture, panel broadcasting is essentially horizontal passing. All broadcasts in a single node use shared memory, and almost all broadcasts between nodes are performed within a single InfiniBand edge switch. This configuration helps reduce the overhead associated with message passing. A larger Q value helps as well.

High Performance LINPACK N Value Settings

HPL algorithms play a key role in performance. For example, while using a large `N` value helps improve performance, it also causes longer execution times. As a result, there may be little value in measuring the impact of larger `N` values. As a result, basic tuning of the large TSUBAME cluster configuration used a smaller `N` value.

The number of floating-point operations can be calculated by:

$$\text{Number of floating-point operations} = \frac{(2 \times N \times N \times N)}{(3 + 2 \times N \times N)}$$

Estimated execution time can be calculated by:

$$\text{Estimated execution time} = \frac{\text{Number of floating-point operations}}{\text{Target Flops}}$$

The amount of memory to use for HPL testing can be estimated as follows:

$$\text{Amount of memory (bytes)} = N \times N \times 8 + 2 \times N \times 8 + N \times 4$$

The HPL `N` value should be divided by the `NB` and `Q` values for improved partitioning. The amount of HPL memory needed, including the memory associated with the MPI and InfiniBand software stack, should be less than the amount of physical memory included in the server. It is important to mind the available memory on each CPU node. Keep in mind the available memory on CPU node 0 is less than other CPU nodes in the current SuSE Linux Enterprise Server NUMA implementation, as the operating environment always allocates memory for the kernel from the CPU 0 node. For example, if HPL is executing with two threads per HPL process and CPU node 0 experiences a memory overflow condition, the HPL process on CPU node 0 must access another CPU's memory, impacting performance.

Figure 3-11 depicts the performance characteristics of several HPL `N` value settings on one, two, and four Sun Fire X4600 servers configured with eight 2.4 GHz AMD Opteron processors and 32 GB of memory. In the one node configuration, using 80 percent of memory yielded the fastest results. In two and four node configurations, using 75 percent of memory produced the fastest results. These results are explained by the fact that the single node configuration used less memory for MPI and InfiniBand communication, while the two and four node configurations accessed more memory for network communication. These results indicate that when HPL is urn on a very large cluster, it is important to give MPI and InfiniBand access to more memory. Similarly, the memory needed for MPI and InfiniBand communication impacts the size of HPL `N` values.

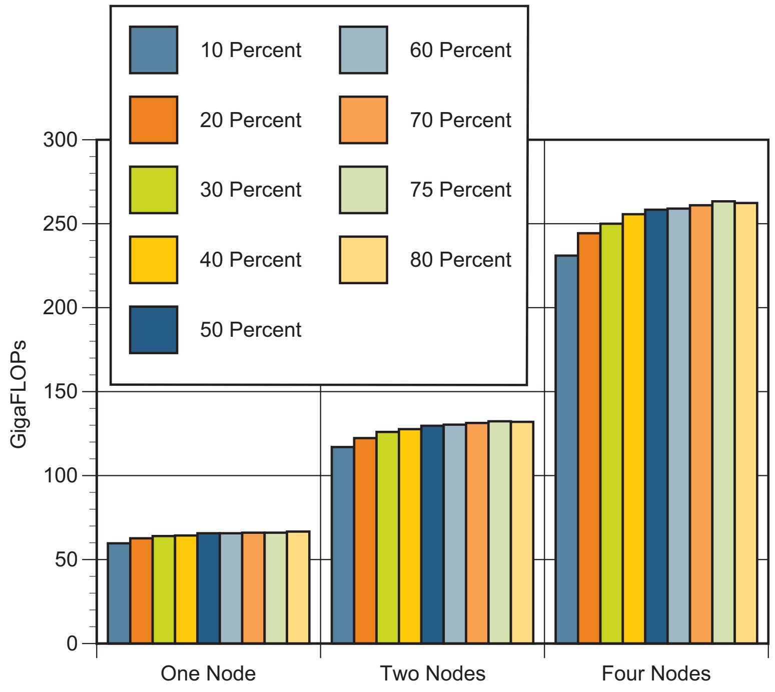

Figure 3-11. Performance comparison using several HPL N values

Other High Performance LINPACK Parameters

Additional HPL parameters can play a role in grid performance. While not analyzed in detail, each value was adopted in the TSUBAME grid based on historical and experimental performance, and theoretical reasoning. Table 3-5 lists the additional HPL parameters and their values used for TSUBAME grid LINPACK benchmark testing.

Table 3-5. Additional HPL parameters and values

Parameter	Value	Parameter	Value
`PMAP`	Row-major process mapping	`SWAP`	Mix (threshold=240)
`PFACT`	Right	`L1`	Transposed form
`NBMIN`	4	`U`	Transposed form
`NDIV`	2	`EQUIL`	Yes
`RFACT`	Right	`ALIGN`	8 double-precision words
`Depth`	1		

Message Passing Interface Parameters

Several Message Passing Interface parameters played a role in tuning the TSUBAME supercomputer grid.

- `VIADEV_USE_SHM_RNDV`
  The `VIADEV_USE_SHM_RNDV` MPI parameter is used by the Voltaire MPI library. The `VIADEV_USE_SHM_RNDV` parameter turns enables and disables use of the Rendezvous protocol for SMP intra-node communication in shared memory for messages that exceed `SMP_EAGERSIZE`. The default value is `VIADEV_USE_SHM_RNDV=0`, indicating Rendezvous messages should use the InfiniBand host adapter rather than shared memory. During TSUBAME grid LINPACK benchmark testing, all message passing for intra-node communication used shared memory, with `VIADEV_USE_SHM_RNDV=1`, to reduce the latency for intra-node communication.

- `SMPI_LENGTH_QUEUE`
  The `SMPI_LENGTH_QUEUE` MPI parameter is used by the Voltaire MPI library. The `SMPI_LENGTH_QUEUE` parameter sets the size of the shared buffer between every two processes on the same server. A larger value may allow more communication without waiting for flow control. This parameter is specified in MB. TSUBAME grid LINPACK benchmark testing used the default value of 2 MB.

- `SMP_EAGERSIZE`
  The `SMP_EAGERSIZE` MPI parameter is used by the Voltaire MPI library. The `SMP_EAGERSIZE` parameter changes the switch point from the Eager protocol to the Rendezvous protocol for SMP intra-node communication. The default value is 512 KB for an x64 architecture. TSUBAME grid LINPACK benchmark testing used a value of `SMP_EAGERSIZE=650`.

- `VIADEV_HCA_DISTRIBUTE`
  The `VIADEV_HCA_DISTRIBUTE` MPI parameter is used by the Voltaire MPI library for dual port and host adapter support. The `VIADEV_HCA_DISTRIBUTE` parameter specifies the method for distributing ranks on active ports and host adapters. The default value is 1, and indicates round-robin distribution on all active ports and host adapters. Ranks are be assigned as follows: rank 1 on port1, rank 2 on port2, rank 3 on port1, rank 4 on port 2, and so on. During TSUBAME grid LINPACK benchmark testing, a setting of `VIADEV_HCA_DISTRIBUTE=2` was used, indicating distribution should be done with sections. In this scheme, the first half of all ranks on the same node are assigned to port or host adapter 1, with the remainder of all ranks on the same node assigned to port or host adapter 2. This method was chosen because rank N will send a message to rank N+1 as a broadcast algorithm. If using a round-robin scheme, all communication between the ranks must use two host adapters, potentially impacting data transfers. The use of sections can alleviate this problem by enabling a single communication to use two host adapters on the same node.

  It is important to note the `VIADEV_HCA_DISTRIBUTE=2` parameter indicates how to assign a MPI rank (process) to dual InfiniBand host adapters for intra-node communication. While this value was used for Linpack testing, its use did not improve Linpack performance. Therefore, shared memory was used for all intra-node communication in the TSUBAME grid, and the VIADEV_HCA_DISTRIBUTE parameter was rendered irrelevant. In addition, if InfiniBand host adapters are used for intra-node communication, section mechanisms could provide better performance than round-robin policies.

FASTPATH

In the Voltaire MPI library, the FASTPATH buffer is allocated for all connections per HPL process by default. As a result, it uses a large amount of memory yet the HPL process sends only a few ranks. The VIADEV_NUM_RDMA_BUFFER parameter is the mechanism for setting the RDMA FASTPATH per connection. The default value is 32. Setting the VIADEV_NUM_RDMA_BUFFER parameter to zero disables the buffer, which can help reduce memory usage for MPI and InfiniBand communication but increases latency. Using an adaptive FASTPATH technique, users can check the actual connections of each rank via a log, and set the VIADEV_ADAPTIVE_RDMA_LIMIT parameter to be the number of actual connections. The VIADEV_ADAPTIVE_RDMA_LIMIT parameter reflects the number of connections that can be upgraded to use the RDMA buffers for the Eager protocol.

In addition, the VIADEV_ADAPTIVE_RDMA_THRESHOLD parameter or VIADEV_ADAPTIVE_PRE_ALLOCATED_PATHS parameter should be set. The VIADEV_ADAPTIVE_RDMA_THRESHOLD parameter defines the number of Eager protocol packets the to send to the recipient before allocating the RDMA buffer for this connection. The the VIADEV_ADAPTIVE_PRE_ALLOCATED_PATHS parameter defines the number of paths to be preallocated and registered for adaptive FASTPATH. This value is equal to the VIADEV_NUM_RDMA_BUFFER vbufs parameter. All FASTPATH options are enabled when the VIADEV_ENABLE_ADAPTIVE_FAST_PATH parameter is set to 1.

Registered Memory

The Voltaire software provides two options related to registered (pin-down) memory on the host adapters.

- VIADEV_MAX_TOTAL_REG_MEM_SIZE
  The VIADEV_MAX_TOTAL_REG_MEM_SIZE parameter is the maximum registered memory size, in MB. The default value is 350.
- VIADEV_NDREG_ENTRIES
  The VIADEV_NDREG_ENTRIES parameter is the number of registered elements in the lazy-de-register. The default value is 1000.

  Using larger values for both parameters helps push more packets into registered memory, improving communication performance. However, finding the optimal setting is challenging as experimenting with multiple combinations of these parameters is difficult, if not impossible. It is clear the default values are too small when running HPL on a large cluster like the TSUBAME grid. As a result, HPL benchmark testing use values of `VIADEV_MAX_TOTAL_REG_MEMSIZE=2000` and `VIADEV_NDREG_ENTRIES=16000` were used in the final testing effort that resulting in grid performance of 38.18 TeraFLOPS. Figure 3-12 shows the difference between two different memory settings on 512 Sun Fire X4600 servers in the grid. For this test, `VIADEV_MAX_TOTAL_REG_MEMSIZE=500` and `VIADEV_NDREG_ENTRIES=5000`. were used. An HPL `N` value of 414480 was used, with only 7.81 percent of memory used for HPL.

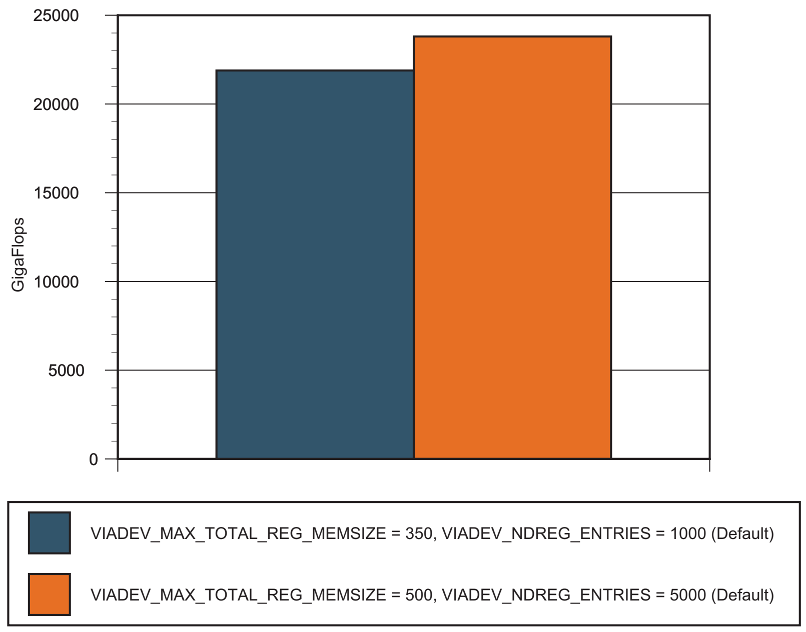

Figure 3-12. Performance comparison using two registered memory settings on 512 nodes

Affinity for the Message Passing Interface

The Voltaire software provides options for setting CPU and memory affinity in MPI functions. When the `VIADEV_ENABLE_AFFINITY` parameter is set to 1, CPU affinity is enabled. This parameter sets CPU affinity for every rank in the same node. When the `VIADEV_ENABLE_MAFFINITY` parameter is set to 1, memory allocation based on memory affinity is enabled. It is important to set both the `VIADEV_SPECIFIC_AFFINITY=1` and `VIADEV_ID=AFFINITY` parameters for all ranks in same node as follows. Note that 99 is a separator between ranks. All other numbers represent CPU numbers. This setting should be equal to the script used to invoke HPL earlier in this document.

```
VIADEV_ID_AFFINITY = 4,5,99,8,9,99,12,13,99,14,15,99,10,11,99,6,7,99,2,3,99,0,1
```

Shared Receive Queue

The Voltaire software provides shared receive queue (SRQ) options for reducing the memory used by MPI and InfiniBand communication. The queue is disabled by default, and was not used for TSUBAME grid LINPACK benchmark testing.

The shared receive queue is enabled by setting the `VIADEV_SRQ_ENABLE` parameter to 1. However, the `VIADEV_SRQ_CON_LIMIT` parameter determines whether the shared receive queue should be used. The default for this parameter is 500 processes. If `NP` is greater than or equal to `VIADEV_SRQ_CON_LIMIT`, the shared receive queue is created. Note that `VIADEV_SRQ_SIZE` is the size of the shared receive queue, and the default value is 512. The `VIADEV_SRQ_LIMIT` parameter sets the lower limit for the shared receive queue, with a default value of 30. The MPI library re-posts receive items into the shared receive queue using an asynchronous event if the number of posted requests in the shared receive queue is smaller than this limit.

LINPACK Performance Results

Table 3-6 details the LINPACK performance results for the TSUBAME supercomputer grid. Multiple cluster sizes were tested, with a 648 node grid incorporating 10,368 processors and 21 TB of memory producing the fastest results at 38.18 TeraFLOPS.

Table 3-6. TSUBAME supercomputer grid LINPACK performance results for multiple cluster sizes

Nodes	Node Configuration	P	Q	Threads	CPUs	Nmax	Rmax	Rpeak	Efficiency
1	2.6 GHz, 8 CPUs, 16 cores, 64 GB memory	2	4	2	16	58320	72.24	83.20	86.83%
1	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	2	4	2	16	58320	67.06	76.80	87.32%
2	2.6 GHz, 8 CPUs, 16 cores, 64 GB memory	4	4	2	32	80400	142.70	166.40	85.76%
2	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	4	4	2	32	80400	132.00	153.60	85.94%
4	2.6 GHz, 8 CPUs, 16 cores, 64 GB memory	4	8	2	64	114000	284.80	332.80	85.58%
4	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	4	8	2	64	114000	263.80	307.20	85.87%
8	2.6 GHz, 8 CPUs, 16 cores, 64 GB memory	8	8	2	128	160080	564.40	665.60	84.80%
8	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	8	8	2	128	160080	524.70	614.40	85.40%
16	2.6 GHz, 8 CPUs, 16 cores, 64 GB memory	8	16	2	256	220800	1121.00	1331.20	84.21%
16	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	8	16	2	256	220800	1042.00	1228.80	84.80%

Nodes	Node Configuration	P	Q	Threads	CPUs	Nmax	Rmax	Rpeak	Efficiency
32	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	16	16	2	512	321120	2054.00	2457.60	83.58%
64	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	16	32	2	1024	454080	4052.00	4915.20	82.44%
96	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	16	48	2	1536	556080	6011.00	7372.80	81.53%
128	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	32	32	2	2048	620400	7690.00	9830.40	78.23%
256	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	32	64	2	4096	741600	14700.00	19660.80	74.77%
512	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	32	128	2	8192	1149120	30200.00	39321.60	76.80%
578	2.4 GHz, 8 CPUs, 16 cores, 32 GB memory	34	136	2	9248	995520	33380.00	44390.40	75.20%
648	2.6 GHz, 8 CPUs, 16 cores, 64 GB memory (16 nodes) 2.4 GHz, 8 CPUs, 16 cores, 32 GB memory (632 nodes)	36	144	2	10368	1334160	38180.00	49868.80	76.56%

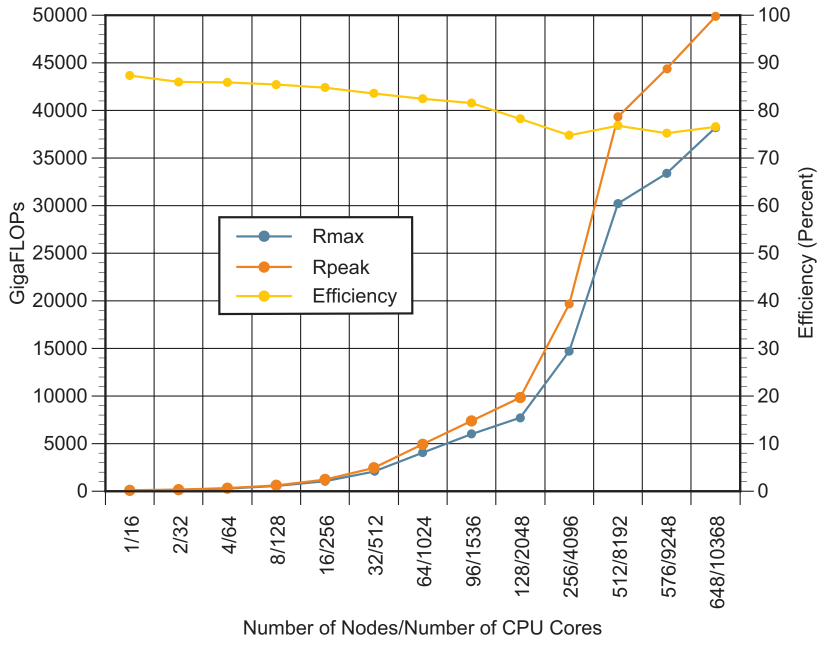

Figure 3-13. TSUBAME supercomputer grid performance history

Summary

Creating a fast supercomputer can be a time-consuming task. By taking an innovative approach that employs Sun x64 servers and data servers in a grid architecture, Sun and Tokyo Tech built a cost-effective, flexible supercomputer that meets the demands of compute- and data-intensive applications. This creative approach utilizes hundreds of systems incorporating thousands of processor cores and terabytes of memory to deliver the performance needed by users running common off-the-shelf applications. Unlike traditional and proprietary solutions, this powerful grid was deployed in only 35 days, testifying to the effectiveness of configuring open systems based on industry standards that run powerful grid software into a fast and reliable grid that can grow and adapt to changing user demands.

Chapter 4
References

About the Author

Nobu Hashizume is a Senior Technical Specialist in Sun's Professional Services organization. Since joining Sun in 1993, Nobu has worked in a variety of consulting roles aimed at helping customers create the most effective computing environments. In addition to working on operating system and compiler technologies, Nobu worked in Sun's World-Wide Customer Benchmarking Center and helped customers tune high performance computing applications for Sun systems. Over the last 13 years, Nobu has consulted on a variety of customer challenges, including working with customers on in-house earthquake data gathering and transfer, the Tokyo Institute of Technology TSUBAME grid, and other HPC projects.

Acknowledgements

The author thanks the following for their advice and contributions to this article:

- Professor Satoshi Matsuoka of the Tokyo Institute of Technology, for his advice on TSUBAME grid architecture and performance.
- Kazushige Goto of the University of Texas at Austin, for providing a fast version of the GotoBLAS library and advice on Linpack tuning.
- Yiftah Shahar of Voltaire, Inc., for his advice on MPI and InfiniBand tuning.
- Brian Whitney, for sharing his knowledge of HPC benchmarks and providing advice on Linpack tuning.
- Ron Larson, Hans Joraandstad, Barbara Perz, Paulde Nijs, and Eduardo Pavon Gonzalez for supporting TSUBAME benchmark efforts.
- Marshall Choy, for providing insight into the Sun x64 server architecture.
- Junichi Koike, for leading the TSUBAME grid architecture design effort and providing invaluable input to this article.
- Toshihiro Kujiraoka, for leading the TSUBAME grid project and providing tremendous support throughout Linpack testing efforts.
- All other individuals involved in the TSUBAME grid project.

References

TOP500 Supercomputer Sites
`http://www.top500.org`

LINPACK
`http://www.netlib.org/linpack`

HPL—A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers
A. Petitet, R. C. Whaley, J. Dongarra, A. Cleary
`http://www.netlib.org/benchmark/hpl`

Netlib Repository at UTK and ORNL
`http://www.netlib.org`

GOTO BLAS
Kazushige Goto
`http://www.tacc.utexas.edu/general/staff/goto`

Texas Advanced Computing Center
`http://www.tacc.utexas.edu`

AMD Core Math Library (ACML)
`http://developer.amd.com/acml.jsp`

Automatically Tuned Linear Algebra Software (ATLAS)
`http://math-atlas.sourceforge.net`

MPI Over InfiniBand Project, Ohio State University
`http://nowlab.cse.ohio-state.edu/projects/mpi-iba`

Ordering Sun Documents

The SunDocs^{SM} program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is
`http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:
`http://www.sun.com/blueprints/online.html`