Cluster Computing and Its Applications at HPCL

Sissades Tongsima Narongsak Pimpunchat Ekasit Kijsipongse
Sitichai Laoweerakul Royol Chitradon

Abstract

At the High Performance Computing Laboratory (HPCL), cluster computing (also known as *Beowulf* cluster) systems are used to reduce the work load of the existing supercomputers. The scope of the cluster system in this paper is to build the system which can efficiently handle parallel and sequential jobs. Furthermore, this system is designed to house the large disk capacity ($> 10^{12}$ bytes or 1 Terra bytes) by using the software RAID (Redundancy Array of Inexpensive Disk). Using RAID, we can improve the performance of the read and write I/O bottlenecks. Here, we present both architectural aspect and software that make up this cluster computing system. The I/O performance issue will also be discussed.

1 Introduction

The computer market trend exhibits the price-performance ratio of the commodity computer and network hardware has been significantly improved over the past 10 years. A performance of a single off-the-shelf CPU is now catching up with those high end CPUs, e.g., in terms of clock speed, one could expect a processor running at or above 1 GHz in the near future. This demonstrates the possible market share between using parallel computational systems built from commercial commodity of the shelf (COTS) components and buying CPU time on costly Supercomputers. Furthermore, technology trends have shown that microprocessors have become smaller and more powerful as the time goes by. Figure 1 supports this claim by showing that the performance figures of COTS processors is converging to those of supercomputers.

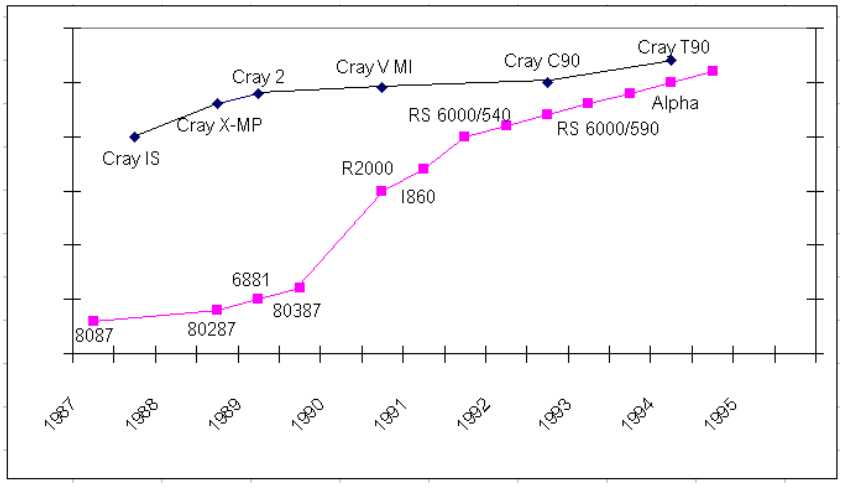

Figure 1: Microprocessors have become smaller and more powerful (source: excerpt from a slide presented by Dr. Dongarra—High Performance Computing and Trends, August 18th 1999)

Observe that the DEC Alpha chip was approaching the performance of Cray T90 by the year 1994. Among the COTS competitors, Alpha chips from DEC seem to outrun other vendors' chips, such as AMD or Intel. Figure 2 portrays that these vendors are catching up the leader of COTS

group. The deficit of 50% performance gap is cut down to less than 3% by 1999 as shown by the *Delta* plots in the figure. With this motivation, many research efforts are aiming at building an in-house cluster computing system to support the need of using CPU times on Supercomputer in their projects.

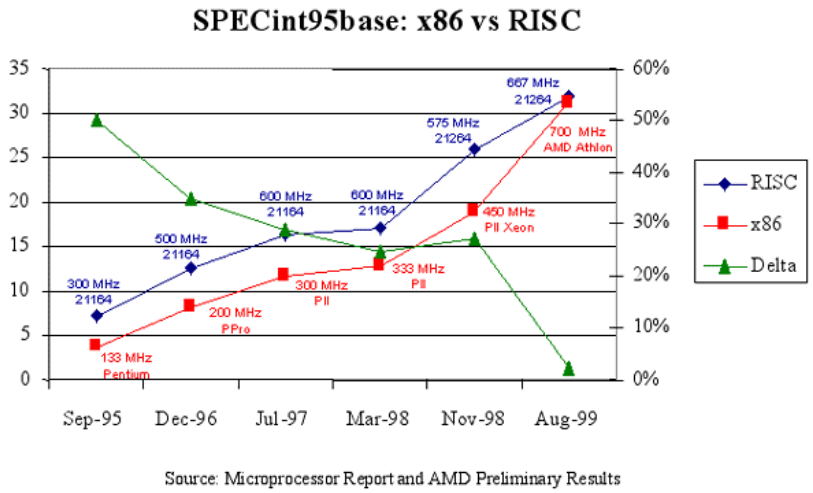

Figure 2: Performance comparison between x86 compatible processors and the RISC series (Alpha chips)

Cluster computing [3] is much similar to the notion of multicomputer where there are two or more computers connected via interconnection network cloud (see Figure 3) as presented in many literatures [1, 2]. Each computer in this system, called a *node*, has its own CPU and memory. The concept of cluster computing has been adopted and implemented by many research organizations where the size (number of nodes) of the system ranges from 2 to more than 512 nodes reflecting the high degree of scalability. This class of computer can be regarded as having one big Supercomputer instead of viewing it as a network of workstations (NOW) where many users own and operate at the consoles (see Figure 4). Users will greatly benefit from using this *monolithic* perspective (single machine) as opposed to stealing the CPU times of other workstations at night when nobody's logged on to complete their parallel jobs.

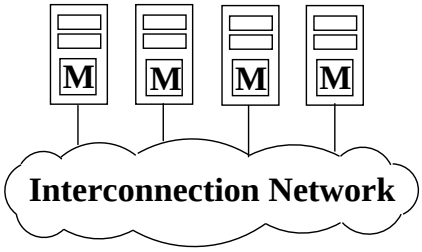

Figure 3: Multicomputer architecture

Computing environment and applications

The target operating system used in the cluster computing machine is free UNIX operating system, called *Linux*. Like other PCs running Linux, this cluster system does not contain any custom hardware component that Linux does not support. The Symmetric Multiprocessor (SMP) feature is recognized by most Linux distributions. To form a parallel computer, the parallel computing

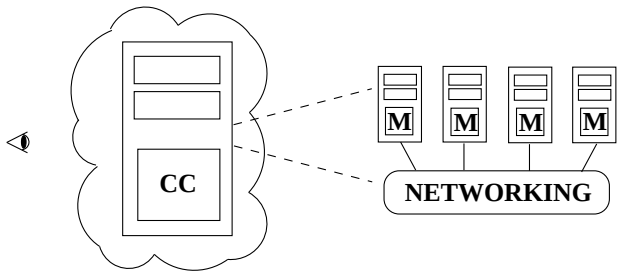

Figure 4: Cluster computing system from a user's view

environment need to be set up. Freely available message passing libraries, e.g. PVM, MPICH and LAM, could be installed to create a usable parallel virtual supercomputer. Furthermore, for numerical computation purposes, parallel numerical libraries, e.g. PETSc, can be provided. The above message passing libraries may require users to have a deep understanding in parallelizing their applications. Doing so will help improve the performance of running parallel applications on the cluster computing system. Unfortunately, training the users to achieve such a goal is costly.

Like many typical supercomputers, this cluster computing system can provide services to many users. We can use the queueing management software, e.g. Distributed Queueing System (DQS), to manage jobs submitted by different users. The parallelization scale for this top management software is coarse-grained when comparing with those message passing libraries. This software suite can bridge the gap between using the existing sequential codes and writing the new parallel codes. In other words, running sequential programs on the cluster computing system can still provide the parallelism (two or more programs can be run in parallel when using the queueing management software).

Applications running on this system can be categorized (but not restricted to) as three major groups: *Data Warehousing*, *Geographical Information System* (GIS), and *Computational Scientific applications*. The applications in the first group include database management system such as PostgreSQL, MySQL and others. For the second group, we will focus on using GRASS (Geographic Resources Analysis and Support System) which allows users to analyze the GIS data. Both GRASS and PostgreSQL can be used to solve complex GIS problems. Finally, the last group may include some scientific applications such as Gaussian and other Quantum Chemistry applications.

1.1 Hardware description

The proposed system comprises of four PCs wired by the *Gigabit Ethernet* technology. This system contains a large number of SCSI hard disk adding up to one Terra byte (OTB) capacity which is the concentration of this system. From this point on, we will refer to the proposed system as the OTB server or just OTB for brevity. Two out of four PCs will be used as the file server where each of which serves 500 Gigabyte of disk space. The OTB server will use 2 dual-pentium III 500MHz Xeon computers to govern the one Terra byte worth of data. The other two computers (using dual-pentium III 500MHz) will serve as the computing nodes. Put another way, these computing nodes are responsible for most computation intensive jobs besides accessing data from the hard disks. Furthermore, these nodes will be used as gateways connecting OTB to the outside world.

The preliminary plan is to get 20 of 50GB Ultra2 SCSI hard disks. Each file server will control 10 of the hard disks. Software RAID is used to setup the one Terra byte of the OTB's secondary storage. Such a storage can be used to store a huge database like GIS, image map, remote sensing and others.

1.2 Detail specification

The following are the list of main components used in our proposed project. It should be noted here that the specification of each component could be altered in order to tune the performance

of the final system.

Component	Specification	Compatibility	Quantity
CPU	Intel Pentium III Xeon 500MHz	yes	4
CPU	Intel Pentium III 500MHz	yes	4
Main-board (File server)	Intel GX chipset for Dual Pentium III Xeon with Ultra2 SCSI Dual Ch.	yes	2
Main-board (Compute node)	Intel BX chipset for Dual Pentium III with Ultra2 SCSI	yes	2
Tape Backup	DLT tape 80 GB	yes	2
NIC	Gigabit Ethernet	yes (fiber optic)	12
NIC	10/100 Ethernet	yes	2
Hard disk	50 GB Ultra2 SCSI	yes	20

1.3 Summary

The purpose of this system is not to replace any existing high performance computing system. It rather serves as an economical auxiliary system which is able to perform decent parallel computation using either *message passing* technology (carefully hand-crafted parallel codes) or numerical *parallel libraries* as well as a reliable web/database server with a large secondary storage.

2 Software on OTB

The term *software* on OTB (one Terra byte system) encompasses a long list of software packages which are used to make the OTB be the OTB. These software packages can be categorized into four groups, Administrative, Development, Service, and Application. The following elaborates the list of software packages (majority) for the OTB version 0.9b.

Administrative : The packages belonging to this group include the operating system and its utilities which allow a sysadmin efficiently governs the system.

Development : Software in this category is those for developing in-house software running on OTB such as message passing libraries, parallel libraries for numerical applications and others. It also includes some libraries which are needed for compiling source codes.

Service : Software allowing users to access the system is likely to fit in this category. Software like `ftp`, `telnet`, web server, DBMS are the examples of this group.

Application : The last group characterizes those applications running on OTB. These applications include both sequential and parallel applications such as GRASS, and other GIS related applications.

Table 1 shows the package listing of software running on the OTB system. Observe that this table corresponds with the release number *0.9beta* which is the prerelease version of this system. Before the major version 1.0 is announced, some of the versions of these packages may be changed (either upgrading or downgrading the software to improve the system stability).

3 Software RAID and its experiments

The RAID (Redundant Array of Inexpensive (Independent) Disks) technology is popularly used to glue many independent disks to make up a large image of the single disk drive [4, 5, 6]. RAID has been characterized into many types according to its functionalities, e.g., RAID-0 uses disk striping[1],

[1]data is partitioned in to small chunks and they are distributed among the physical disks. This way the data can be retrieved or written in parallel from the disks. However, RAID-0 does not honor fault tolerant issue. This makes read/write in RAID-0 operate very fast comparing with other modes.

Administrative Group		Development Group		Service Group		Application Group	
Linux	Mandrake-6.01 Kernel-2.2.12 SMP (kernel)	Message Passing	PVM MPICH	Networking	Telnet FTP	Software	GRASS4 Gaussian
		Parallel Lib	PETSc	WWW server	Apache-1.3	Xwindows	XFree86-3.3.1
Security	SSH (secure) TCP Wrapper	Database	PostgreSQL-6.5			Browser	Netscape-4.7
				Unix/Dos	Samba		
Job manager	DQS-3.2.7 CRON/AT	Java related	JDK2	Mail	sendmail-8.9		
		Software Engr.	CVS				
System Manager	NIS/NFS RAID,NAT	Debugger	ddd				

Table 1: Package listing of software on One Terra Byte cluster computing system

RAID-1 uses for mirroring the data[2] etc. To implement RAID one can either use software or special hardware to create the system. We have chosen to use the software RAID in our system.

To demonstrate the the stability and performance issues of RAID, we have performed the experiment to study how RAID can improve performance of the I/O system, namely read and write operations. We also present how the number of disks affects the performance of RAID. Note that the performance of the I/O system in this experiment is measured by the benchmark tool, called *bonnie*. It performs a series of reading and writing tests on a file of known size (2 Gigabytes). For each test, bonnie reports the bytes processed per elapsed second, per CPU second, and the percent CPU usage (user and system). The following are the specifications of the experimented system.

Hardware

- CPU: 2xIntel Pentium II Xeon 450MHz Cache L2 512KB
- Memory: 128MB
- Hard disk: SEAGATE Model ST318275LW Size 18GB
- BUS: PCI 32bits 33MHz
- I/O: SCSI 2 channel

Software

- OS: Linux Mandrake 6.0 with Thai Extension from NECTEC
- Kernel Version: 2.2.11 with software RAID-19990824 patch
- Swap space size: 256MB

3.1 Testing methodology

1. Use benchmark tool, called "bonnie", to measure the speed of reading and writing data to and from a single disk (NON-RAID). The size of data is fixed at 2000MB to ensure that it is large enough so that the speed of I/O does not get dominated by the OS cache. Two bonnie processes have been run simultaneously to simulate multi-user environment.
2. Starting with the number of disks equals 2, create RAID-0 such that the odd-numbered disks attach to channel I (SCSI channel) and the even-numbered disks attach to channel II.
3. Measure the speed of reading and writing using bonnie.
4. Increase the number of disks in RAID then goto step 2.

3.2 Bonnie the benchmarker

1. **Sequential Output**:
   (a) *Per-character*: The file is written using the `putc()` standard I/O macro. The loop that does the writing should be small enough to fit into any reasonable I-cache. The CPU overhead here is that required to do the stdio code plus the OS file space allocation.
   (b) *Block*: The file is created using `write(2)`. The CPU overhead should be just the OS file space allocation.

[2] RAID-1 is used for mirroring data. Two identical copies of data will be stored in the disks in order to improve the reliability of the I/O. This helps improve the speed of reading operation. Nonetheless, extra disk space must be allocated for this purpose and integrity checking shall be performed when writing occurs.

(c) *Rewrite*: Each BUFSIZ (buffer size) of the file is read with `read(2)`, set to dirty, and is rewritten with `write(2)`, requiring an `lseek(2)`. Since no space allocation is done, and the I/O is well-localized, this should test the effectiveness of the file system cache and the speed of data transfer.

2. **Sequential Input**:

   (a) *Per-character*: The file is read using the `getc()` stdio macro. Once again, the inner loop is small. This should exercise only stdio and sequential input.

   (b) *Block*: The file is read using `read(2)`. This should be a very pure test of sequential input performance.

3. **Random Seeks**: This test runs SeekProcCount processes in parallel, doing a total of 4000 `lseek()`s to locations in the file specified by `random()` in BSD systems, `drand48()` on SySV systems. In each case, the block is read with `read(2)`. In 10% of cases, it is set to dirty and written back with `write(2)`.

3.3 Results

The following are the dumped output (plain text) from running bonnie against the tested while varying the number of hard disks.

```
1 disks (NON-RAID)
              -------Sequential Output-------- ---Sequential Input-- --Random--
              -Per Char- --Block--- -Rewrite-- -Per Char- --Block--- --Seeks---
            MB K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU  /sec %CPU
          2000  9370 90.9 13150 18.7  2682  4.4  6192 64.9  6283  3.8  35.4  0.4

2 disks (RAID)
              -------Sequential Output-------- ---Sequential Input-- --Random--
              -Per Char- --Block--- -Rewrite-- -Per Char- --Block--- --Seeks---
            MB K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU  /sec %CPU
          2000  9712 94.2 24754 37.2  4959  9.2  6818 72.8 11882  7.6  80.0  0.9

3 disks (RAID)
              -------Sequential Output-------- ---Sequential Input-- --Random--
              -Per Char- --Block--- -Rewrite-- -Per Char- --Block--- --Seeks---
            MB K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU  /sec %CPU
          2000  9513 93.5 34172 23.6  5851 11.1  6885 70.0 14421  7.9  95.2  1.4

4 disks (RAID)
              -------Sequential Output-------- ---Sequential Input-- --Random--
              -Per Char- --Block--- -Rewrite-- -Per Char- --Block--- --Seeks---
            MB K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU  /sec %CPU
          2000  9800 95.0 36241 55.4  6332 12.2  7559 79.8 15368  9.5 141.0  1.1
```

To show the trend of the above experiments, some of data produced by bonnie were plotted as displayed in Figures 5–6. The number of disks used in these experiments are plotted against the transfer rate (read or write) in Kbytes. Figure 5 presents throughput of reading and writing data in block operation while Figure 6 shows throughput of reading and writing data per character.

3.4 Summary

By using RAID, the I/O system has a significant performance improvement over not using it. As the number of disks in RAID grows the speed of reading, writing and seeking data tends to increase. Nonetheless, for characters I/O, the benefit of using RAID is not as high as using RAID for block

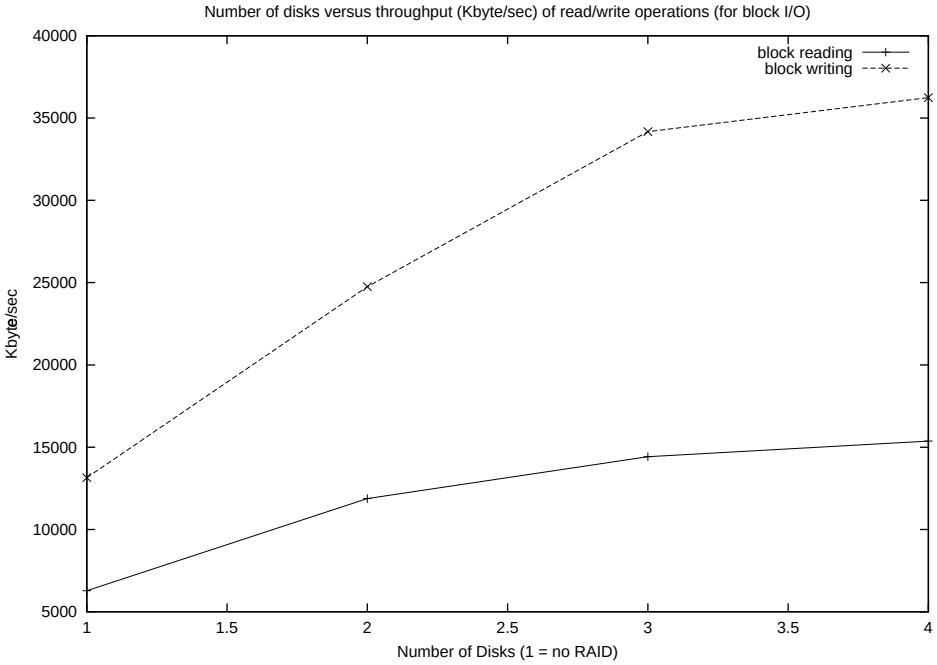

Figure 5: Sequential block I/O operations

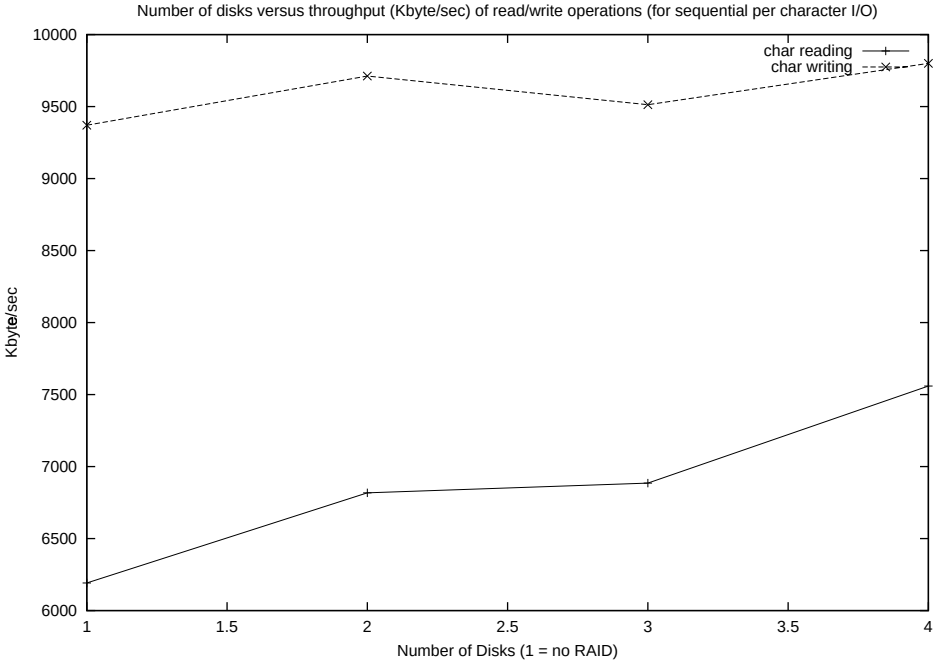

Figure 6: Sequential character I/O operations

I/O. This is because much of the work done in read or write character I/O operations is lost in the `putc()` and `getc()` loops.

4 Conclusion

Our vision of creating a system which has a very good price-performance ratio and capable of dealing with tasks that require a lot of data storage is done with the combination of *Beowulf* class concept and the *RAID* technology. The designed system uses the commodity of the shelf computers along with a large number of independent hard drives. The target applications for our proposed system ranges from regular scientific computations to applications which require a large storage like data warehousing and/or geographical information system applications. Not only can the proposed system execute program written for parallel machines, but also it can handle multiple sequential jobs effectively using the managing software to evenly distribute the system load. In terms of data storage, using RAID helps improve the performance of I/O read-write operations. This is due to how RAID take advantage of the physical arrangement of the system hard drives. In other words, the I/O operations can be perform in parallel rather than in sequential as appeared in a single hard disk.

References

[1] Ian Foster. *Designing and Building Parallel Programs: Concept and Tools for Parallel Software Engineering*. Addison-Wesley Publishing Co., 1994.

[2] Kai Hwang. *Advance Computer Architecture: Parallelism, scalability, programmability*. McGraw-Hill Publishing Co., 1993.

[3] High Performance Computing Laboratory, National Electronics, and Computer Technology Center. Cluster computing. URL: http://cluster1.hpcc.nectec.or.th.

[4] Jakob Stergaard. Another linux software RAID HOWTO. URL: http://ostenfeld.dk/ jacob/Software-RAID.HOWTO.

[5] Linas Vepstas. Linux software RAID HOWTO. URL: http://www.linuxdoc.org/mini/Software-RAID.html.

[6] Linas Vepstas. Raid solutions for linux. URL: http://linas.org/linux/raid.html.