Computer Modelling & New Technologies, 2003, Volume 7, No.2, 7-15
Transport and Telecommunication Institute, Lomonosov Str.1, Riga, LV-1019, Latvia

CLUSTER APPROACH TO HIGH PERFORMANCE COMPUTING

A. KUZMIN

Institute of Solid State Physics, University of Latvia, Kengaraga Street 8, LV-1063 Riga, Latvia
E-mail: a.kuzmin@cfi.lu.lv

The High Performance Computing (HPC) allows scientists and engineers to deal with very complex problems using fast computer hardware and specialized software. Since often these problems require hundreds or even thousands of processor hours to complete, an approach, based on the use of supercomputers, has been traditionally adopted. Recent tremendous increase in a speed of PC-type computers opens relatively cheap and scalable solution for HPC using cluster technologies. Here we discuss basics of cluster technology and present an example of cluster solution, implemented at the Institute of Solid State Physics of the University of Latvia within the Latvian SuperCluster (LASC) project.
Keywords: high performance computing, cluster computing

1. Introduction

Development of new materials and production processes, based on high-technologies, requires a solution of increasingly complex computational problems. However, even as computer power, data storage, and communication speed continue to improve exponentially, available computational resources are often failing to keep up with what users demand of them. Therefore high-performance computing (HPC) infrastructure becomes a critical resource for research and development as well as for many business applications. Traditionally the HPC applications were oriented on the use of high-end computer systems - so-called "supercomputers". Before considering the amazing progress in this field, some attention should be paid to the classification of existing computer architectures.

SISD (Single Instruction stream, Single Data stream) type computers. These are the conventional systems that contain one central processing unit (CPU) and hence can accommodate one instruction stream that is executed serially. Nowadays many large mainframes may have more than one CPU but each of these execute instruction streams that are unrelated. Therefore, such systems still should be regarded as a set of SISD machines acting on different data spaces. Examples of SISD machines are for instance most workstations like those of DEC, IBM, Hewlett-Packard, and Sun Microsystems as well as most personal computers.

SIMD (Single Instruction stream, Multiple Data stream) type computers. Such systems often have a large number of processing units that all may execute the same instruction on different data in lock-step. Thus, a single instruction manipulates many data in parallel. Examples of SIMD machines are the CPP DAP Gamma II and the Alenia Quadrics.

Vector processors, a subclass of the SIMD systems. Vector processors act on arrays of similar data rather than on single data items using specially structured CPUs. When data can be manipulated by these vector units, results can be delivered with a rate of one, two and, in special cases, of three per clock cycle (a clock cycle being defined as the basic internal unit of time for the system). So, vector processors execute on their data in an almost parallel way but only when executing in vector mode. In this case they are several times faster than when executing in conventional scalar mode. For practical purposes vector processors are therefore mostly regarded as SIMD machines. Examples of such systems are Cray 1 and Hitachi S3600.

MIMD (Multiple Instruction stream, Multiple Data stream) type computers. These machines execute several instruction streams in parallel on different data. The difference with the multi-processor SISD machines mentioned above lies in the fact that the instructions and data are related because they represent different parts of the same task to be executed. So, MIMD systems may run many sub-tasks in parallel in order to shorten the time-to-solution for the main task to be executed. There is a large variety of MIMD systems like a four-processor NEC SX-5 and a thousand processor SGI/Cray T3E supercomputers.

Besides above mentioned classification, another important distinction between classes of computing systems can be done according to the type of memory access (Figure 1).

Shared memory (SM) systems have multiple CPUs all of which share the same address space. This means that the knowledge of where data is stored is of no concern to the user as there is only one memory accessed by all CPUs on an equal basis. Shared memory systems can be both SIMD or MIMD. Single-CPU vector processors can be regarded as an example of the former, while the multi-CPU models of these machines are examples of the latter.

Distributed memory (DM) systems. In this case each CPU has its own associated memory. The CPUs are connected by some network and may exchange data between their respective memories when required. In contrast to shared memory machines the user must be aware of the location of the data in the local memories and will have to move or distribute these data explicitly when needed. Again, distributed memory systems may be either SIMD or MIMD.

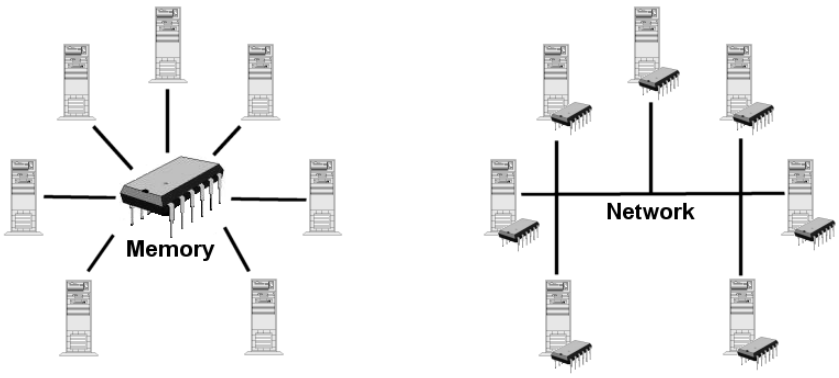

Figure 1. Shared (left) and distributed (right) memory computer architectures

To understand better the current situation in the field of HPC systems and a place of cluster-type computers among them, some brief overview of supercomputers history will be given below.

An important break-through in the field of HPC systems came up in the late 1970s, when the Cray-1 and soon CDC Cyber 203/205 systems, both based on vector technology, were built. These supercomputers were able to achieve unprecedented performances for certain applications, being more than one order of magnitude faster than other available computing systems. In particular, the Cray-1 system [1] boasted that time a world-record speed of 160 million floating-point operations per second (MFlops). It was equipped with an 8 megabyte main memory and priced $8.8 million. A range of first supercomputers applications was typically limited by ones having regular, easily vectorisable data structures and being very demanding in terms of floating point performance. Some examples include mechanical engineering, fluid dynamics and cryptography tasks. The use of vector computers by a broader community was initially limited by the lack of programming tools and vectorising compilers, so that the applications had to be hand coded and optimised for a specific computer system. However, commercial software packages became available in the 1980s for vector computers, pushing up their industrial use. At this time, the first multiprocessor supercomputer Cray X-MP was developed and achieved the performance of 500 MFlops.

Strong limitation for further scalability of vector computers was their shared-memory architecture. Therefore, massive parallel processing (MPP) systems using distributed-memory were introduced by the end of the 1980s. The main advantage of such systems is the possibility to divide a complex job into several parts, which are executed in parallel by several processors each having dedicated memory (Figure 1). The communication between the parts of the main job occurs within the framework of the so-called message-passing paradigm, which was standardised in the message-passing interface (MPI). The message-passing paradigm is flexible enough to support a variety of applications and is also well adapted to the MPP architecture. During last years, a tremendous improvement in the performance of standard workstation processors led to their use in the MPP supercomputers, resulting in significantly lowered price/performance ratios.

Traditionally, conventional MPP supercomputers are oriented on the very high-end of performance. As a result, they are relatively expensive and require special and also expensive maintenance support. To meet the requirements of the lower and medium market segments, the symmetric multiprocessing (SMP) systems were introduced in the early 1990s to address commercial users with applications such as databases, scheduling tasks in telecommunications industry, data mining and manufacturing.

Better understanding of applications and algorithms as well as a significant improvement in the communication network technologies and processors speed led to emerging of new class of systems, called *clusters of SMP* or *networks of workstations* (NOW), which are able to compete in performance with MPPs and have excellent price/performance ratios for special applications types. On practice, clustering technology can be used for any arbitrary group of computers, allowing to build homogeneous or heterogeneous systems. Even bigger performance can be achieved by combining groups of clusters into HyperCluster or even Grid-type system [2], which will be briefly considered in the last section.

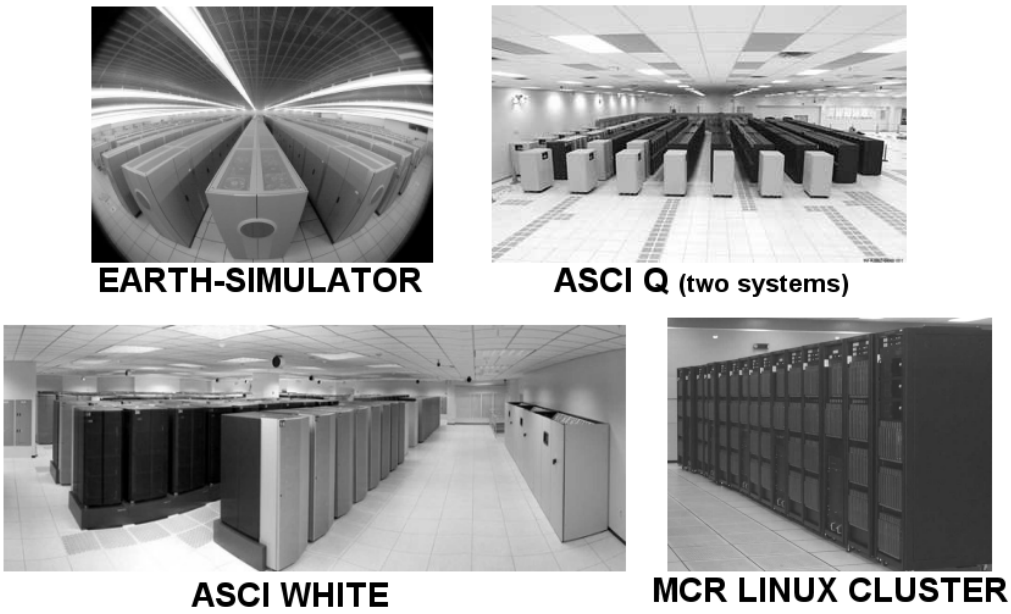

Figure 2. Top five supercomputer systems (November 2002) [6]

It is worth to note, that by the end of 2002, the most powerful existing HPC systems (Figure 2) have performance in the range from 3 to 36 TFlops. The top five supercomputer systems include Earth-Simulator (35.86 TFlops, 5120 processors), installed by NEC in 2002 [3]; two ASCI Q systems (7.72 TFlops, 4096 processors), built by Hewlett-Packard in 2002 and based on the AlphaServer SC computer systems [4]; ASCI White (7.23 TFlops, 8192 processors), installed by IBM in 2000 [4]; and, a pleasant surprise, MCR Linux Cluster (5.69 TFlops, 2304 Xeon 2.4 GHz processors), built by Linux NetworX in 2002 for Lawrence Livermore National Laboratory (USA) [5]. According to the recent TOP500 Supercomputers List from November 2002 [6], cluster based systems represent 18.6% from all supercomputers, and most of them (about 60%) use Intel's processors. Finally, one should note that application range of modern supercomputers is very wide and addresses mainly industrial, research and academic fields. The covered areas are related to telecommunications, weather and climate research/forecasting, financial risk analysis, car crash analysis, databases and information services, manufacturing, geophysics, computational chemistry and biology, pharmaceutics, aerospace industry, electronics and much more.

In this paper we will introduce basics of cluster technology and consider an example of the HPC cluster, built at the Institute of Solid State Physics (ISSP) of the University of Latvia.

2. Basics of Cluster Computing

Cluster computing refer to technologies that allow multiple computers, called cluster nodes, to work together with the aim to solve common computing problems. A generic cluster architecture is shown in Figure 3. Each node can be a single or multiprocessor computer, such as a PC, workstation or SMP server, equipped with its own memory, I/O devices and operating system. The cluster, having similar nodes, is called homogeneous, otherwise - heterogeneous. The nodes are usually interconnected by local area network (LAN) based on one of the following technologies: Ethernet, Fast Ethernet, Gigabit Ethernet, Myrinet [7], Quadrics Network (QsNet) [8], InfiniBand communication fabric [9], Scalable Coherent Interface (SCI) [10], Virtual Interface Architecture (VIA) [11] or Memory Channel [12]. The

speed of network technology is characterised by a bandwidth and a latency. Bandwidth means how much information can be sent through a particular network connection, and latency is defined as the time it takes for a networking device to process a data frame. A comparison of these two parameters for above mentioned network technologies is given in Table 1. Note that a higher network speed is usually associated with a higher price of related equipment. To improve further cluster performance, different network topologies can be implemented in each particular case. Moreover, channel bonding technology can be used in the case of the Ethernet-type networking to double the network bandwidth. To realise this technology, two network interface cards (NICs) should be installed in each node, and two network switches should be used, one for each channel, to form two separate virtual networks. The optimal choice of the network type is dictated by demands on speed and volume of data exchange between several parts of the application software, running on different nodes.

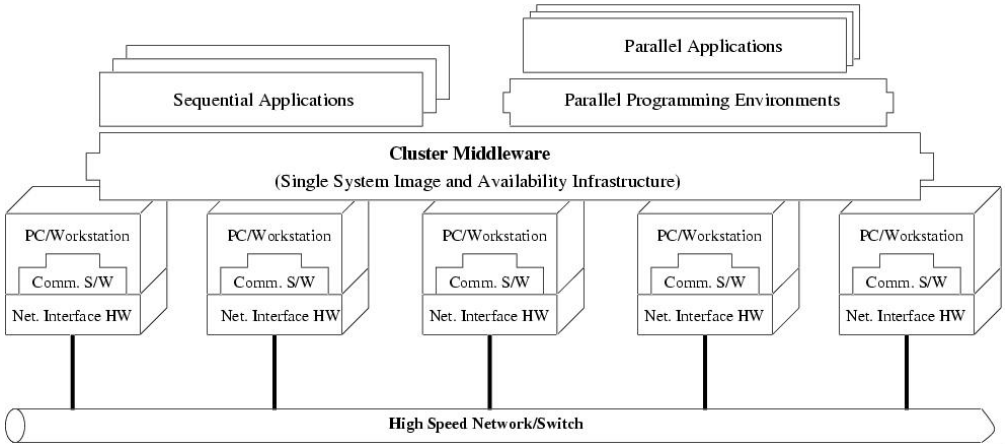

Figure 3. Typical cluster architecture [16]

TABLE 1. Comparison of network technologies used in cluster systems. Note that the latency of Ethernet devices depends strongly on a particular switch realisation.

Network Technology	Bandwidth (MByte/sec)	Latency (μsec/packet)
Ethernet	1.25	-
Fast Ethernet	12.5	158
Gigabit Ethernet	125	33
Myrinet	245	6
QsNet	340	2
InfiniBand	320, 1280 and 3840	<10
SCI	400	1.5
VIA	150	8
Memory Channel	100	3

Various operating systems, including Linux, Solaris and Windows, can be used to manage the nodes. However, in order for the clusters to be able to pool their computing resources, special cluster-enabled applications must be written using clustering libraries or a system level *middleware* [13] should be used. The most popular clustering libraries are PVM (Parallel Virtual Machine) [14] and MPI (Message Passing Interface) [15]; both are very mature and work well. By using PVM or MPI, programmers can design applications that can span across an entire cluster's computing resources rather than being confined to the resources of a single machine. For many applications, PVM and MPI allow computing problems to be solved at a rate that scales almost linearly in relation to the number of processors in the cluster.

The cluster architecture is usually optimised for *High Performance Computing* or *High Availability Computing*. The choice of the architecture is dictated by the type of an application and available budget. A combination of both approaches is utilised in some cases, resulting in a highly reliable system, characterized by a very high performance. The principal difference between these two approaches consists of that in the HPC case, each node in the cluster executes a part of the common job,

whereas in the second case, several nodes perform or are ready to perform the same job and, thus, are able to substitute each other in a case of failure.

High availability (HA) clusters are used in mission critical applications to have constant availability of services to end-users through multiple instances of one or more applications on many computing nodes. Such systems found their application as Web servers, e-commerce engines or database servers. HA clusters use redundancy to ensure that a service remains running, so that even when a server fails or must go offline for service, the other servers pick up the load. The system optimised for maximum availability should not have any single point of failure, thus requiring a specific architecture (Figure 4). Two types of HA clusters can be distinguished - shared nothing architecture and shared disk architecture. In the first case, each computing node is using dedicated storage, whereas the second type of HA cluster shares common storage resources, interconnected by Storage Area Network (SAN). The operation of HA cluster requires normally special software, which is able to recognize the occurred problem and transparently migrate the job to another node.

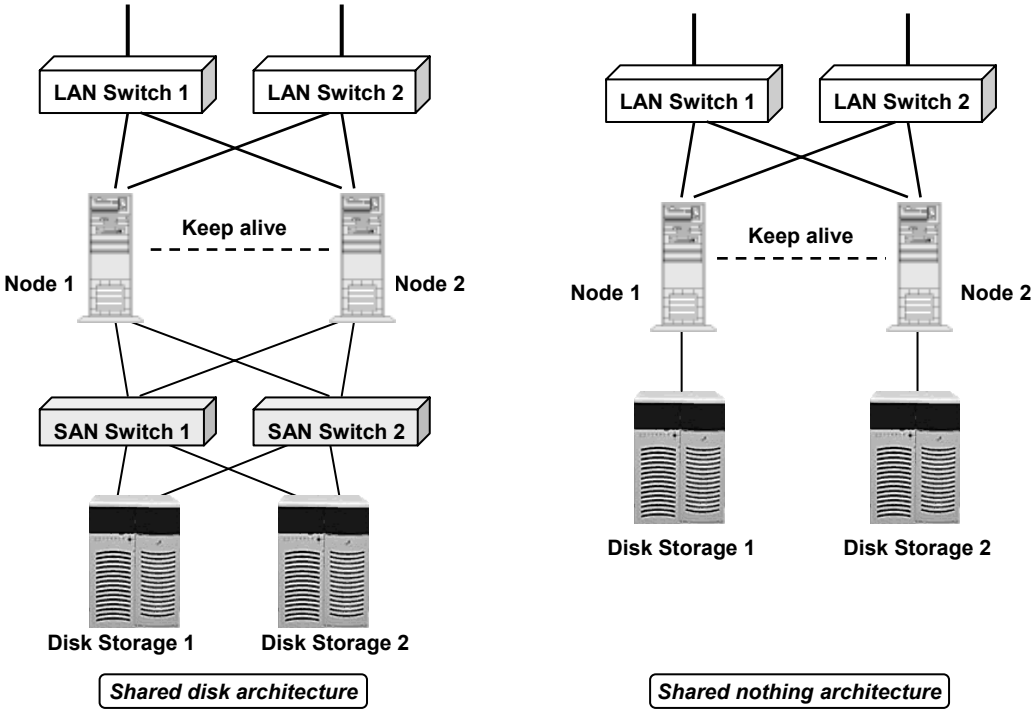

Figure 4. High availability cluster with no single point of failure

HPC clusters are built to improve processing throughput in order to handle multiple jobs of various sizes and types or to increase performance. The most common HPC clusters are used to shorten turnaround times on compute-intensive problems by running the job on multiple nodes at the same time or when the problem is just too big for a single system. This is often the case in scientific, design analysis and research computing, where the HPC cluster is built purely to obtain maximum performance during the solution of a single, very large problem. Such HPC clusters utilise parallelised software that breaks down the problem into smaller parts, which are dispatched across a network of interconnected systems that concurrently process each small part and then communicate with each other using message-passing libraries to coordinate and synchronize their results. The Beowulf-type cluster [17], which will be described in the next section, is an example of the HPC system. Beowulf system is the cluster which is built primarily out of commodity hardware components, is running a free-software operating system like Linux or FreeBSD and is interconnected by a private high-speed network. However, some Linux clusters, which are built for high availability instead of speed, are not Beowulfs.

While Beowulf clusters are extremely powerful, they are not for everyone. The primary drawback of Beowulf clusters is that they require specially designed software in order to take advantage of cluster resources. This is generally not a problem for those in the scientific and research communities who are used to writing their own special purpose applications since they can use PVM or MPI libraries

to create cluster-aware applications. However, many potential users of the cluster technologies would like to have some kind of performance benefit using standard applications. Since such applications have not been written with the use of PVM or MPI libraries, such users simply cannot take advantage of a cluster. This problem has been limited the use of cluster technologies to a small group of users for years. Recently, a new technology, called openMosix [18], appears that allows standard applications to take advantage of clustering without being rewritten or even recompiled.

OpenMosix is a "patch" to the standard Linux kernel, that adds clustering abilities and allows any standard Linux process to take advantage of a cluster's resources. OpenMosix uses adaptive load-balancing techniques and allows processes running on one node in the cluster to migrate transparently to another node where they can execute faster. Because openMosix is completely transparent to all running programs, the process that has been migrated does not even know that it is running on another remote node. This transparency means that no special programming is required to take advantage of openMosix's load-balancing technology. In fact, a default openMosix installation will migrate processes to the "best" node automatically. This makes openMosix a clustering solution that can provide an immediate benefit for many applications.

A cluster of Linux computers running openMosix can be considered as a large virtual SMP system with some exclusions. The CPUs on a "real" SMP system can exchange data very fast, but with openMosix, the speed at which nodes can communicate with one another is determined by the speed of the network. Besides, openMosix does not currently offer support for allowing multiple cooperating threads to be separated from one another. Also, like an SMP system, openMosix cannot execute a single process on multiple physical CPUs at the same time. This means that openMosix will be not able to speed up a single process/program, except to migrate it to a node where it can execute most efficiently. At the same time, openMosix can migrate most standard Linux processes between nodes and, thus, allows for extremely scalable parallel execution at the process level. Besides, if an application forks many child processes then openMosix will be able to migrate each one of these processes to an appropriate node in the cluster. Thus, openMosix provides a number of benefits over traditional multiprocessor systems.

The openMosix technology can work in both homogeneous and heterogeneous environments, thus allowing to build clusters, consisting of tens or even hundreds of nodes, using inexpensive PC hardware as well as a bunch of high-end multi-processor systems. The use of openMosix together with new Intel's Hyper-Threading technology, available with the last generation of Intel Xeon processors, allows additional improving of performance for threaded applications. Also existing MPI/PVM programs can benefit from openMosix technology.

4. Latvian SuperCluster (LASC) as an Example of Cluster System

In this sections we will describe an example of the Beowulf-type cluster (Figure 5), called LASC (Latvian SuperCluster), which was installed at the ISSP during 2002 [19]. The main goal of the LASC project is to secure researchers with an access to HPC system, able to help in a solution of modern physical problems by numerical simulations. The need for such installation is determined by a complexity of tasks appearing now within fundamental and applied research projects in solid state physics and materials science. Currently, the main use of the cluster is devoted to quantum chemistry calculations, Monte-Carlo modelling and x-ray absorption spectra analysis.

A specific sort of physical problems and a limitation of financial resources forced a careful choice of hardware and software components for the cluster construction. At the planning stage, several criteria have been taken into consideration to optimize the cluster configuration:

- high speed for numerical calculations,
- large amount of physical memory, able to accumulate huge arrays appearing in quantum chemistry calculations,
- reliable hardware with a possibility of upgrading and expanding,
- high speed networking between cluster nodes,
- easy maintenance and administration,
- secure access on pre-registration basis,
- lowest total cost.

As one can note, the last requirement is somewhat controversial to others therefore a compromise have to be done. The "best" solution in the spring 2002, conforming mostly to specified

requirements, was a cluster of SMP servers, equipped with two Intel Pentium-III CPUs and interconnected through the Fast Ethernet switch. Therefore, the present cluster configuration (Table 2), adopted according to the available budget, consists of five nodes (one front-end node plus four computational nodes), which are Compaq ProLiant ML350 G2 servers. Only the front-end node has a connection with the rest of the world (Internet) using one of the Network Interface Cards (NICs). The other NICs on all nodes are connected to a HP ProCurve 2312 Fast Ethernet switch. The total resources available to the users are 10 CPUs, having a peak power about 13 GFlops, 20 GB of physical memory (RAM) and 336 GB of storage space on UATA-100 IDE/SCSI hard disks (expandable up to 3.3 TB).

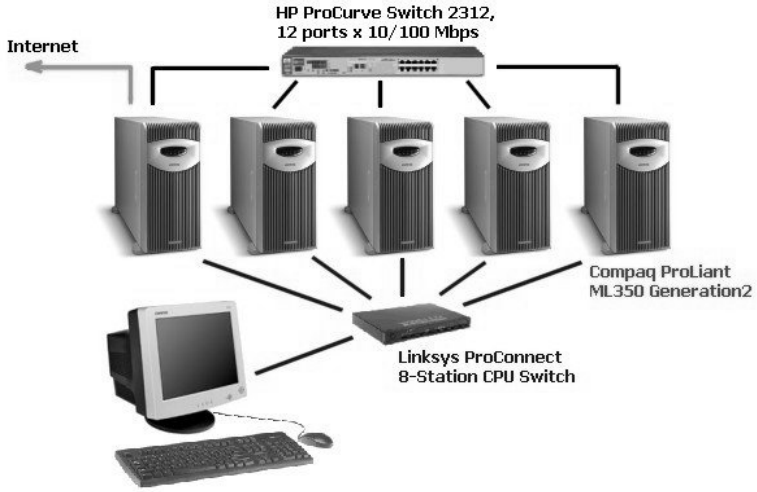

Figure 5. LASC structure. The APC Smart-UPS are note shown

TABLE 2. Technical details of Latvian SuperCluster.

1. Front-end (master) node - Compaq ProLiant ML350 G2 server

- Two Intel Pentium® lll 1.26 GHz FC-PGA2 processors with 512-KB second level ECC cache
- ServerWorks LE 3.0 Chipset with 133-MHz Front Side Bus
- 4 GB PC133-MHz Registered ECC SDRAM memory (RAM)
- Compaq NC3163 Fast Ethernet 10/100 Mbit/s NIC (embedded) PCI 10/100
- Intel® PRO/1000 XT 64bit PCI Server Adapter (8490XT)
- IBM Deskstar 60.0 GB 60GXP (7200 rpm) UATA100-IDE Hard Drive
- Maxtor 120 GB DiamondMaxPlus9 (7200 rpm) UATA133-IDE Hard Drive
- Promise Technology Ultra100 TX2 PCI Controller Card
- Compaq 36.4 GB (10000 rpm) Hot Plug Wide Ultra3SCSI Hard Drive
- 40x IDE CD-ROM Drive
- 1.44 MB Floppy Drive
- Integrated ATI RAGE XL Video Controller with 8-MB SDRAM Video Memory
- Integrated Dual Channel Wide Ultra3 SCSI Adapter

2. Four computational nodes - Compaq ProLiant ML350 G2 servers

- Two Intel Pentium® lll 1.26 GHz FC-PGA2 processors with 512-KB second level ECC cache
- ServerWorks LE 3.0 Chipset with 133-MHz Front Side Bus
- 4 GB PC133-MHz Registered ECC SDRAM memory (RAM)
- Compaq NC3163 Fast Ethernet 10/100 Mbit/s NIC (embedded) PCI 10/100
- IBM Deskstar 60.0 GB 60GXP (7200 rpm) UATA100-IDE Hard Drive
- Promise Technology Ultra100 TX2 PCI Controller Card
- 40x IDE CD-ROM Drive
- 1.44 MB Floppy Drive
- Integrated ATI RAGE XL Video Controller with 8-MB SDRAM Video Memory
- Integrated Dual Channel Wide Ultra3 SCSI Adapter

3. Five APC Smart-UPS 620VA (one for each node)

4. Hewlett Packard ProCurve Switch 2312, 12 x 10/100 Mbps

5. Lynksys ProConnect 8-Station CPU KVM Switch

6. Monitor 17" Compaq V7550 (0.25, 1600x1200 max, Flat Screen), Keyboard, Mouse

All main software, installed on the LASC cluster, is of open source type and is available for free under GNU public license. The nodes are running under the Red Hat Linux operating system on the private subnetwork, protected from the Internet by a firewall. The cluster is accessible only via secure shell interface (SSH2) and, additionally, the authorisation by the client's IP address is performed. For users convenience, all information related to the cluster use and its real time status is available on-line from the Web [19].

The storage disks space in the cluster is shared among all the nodes via Network File System (NFS), so that all data as well as home directories of the users are accessible on all nodes in a similar way. The users applications can run in interactive mode, in background or in batch mode. At present time, to launch an application on the particular node, the user should login to that node first or use an automatic job submitting system, called CLRUN. The CLRUN system allows (i) to submit a job from any node to any node for execution and (ii) to display the load for every node and all user's processes on each nodes. In the future, we plan to install openMOSIX load balancing system to improve the overall cluster performance and to simplify its use. The programming environment available to the users consists of the GNU set of compilers (C/C++/Fortran/Pascal) and the Intel's C++/Fortran compilers, which have better degree of optimization and able to automatically parallelise a user's code on SMP systems. To use the entire cluster's computing resources within user's applications, the popular clustering libraries such as PVM and MPI are available.

The LASC system represents a classical example of the "home-made" Beowulf cluster. Similar systems can be easily build for other than scientific needs with possibly even lower price. As an example, standard off-the-shelf computers, equipped with much smaller memory and thus being much cheaper, can be used as nodes in a cluster for multimedia encoding applications, such as sound MPEG-3 or video MPEG-4 processing, and for images rendering to create stunning three-dimensional graphics.

5. Future trends - Grid computing

As computer networks become cheaper and faster, a new computing paradigm, called the Grid [2], has evolved. The Grid is a large system of computing resources that performs tasks and provides to users a single point of access, commonly based on the World Wide Web interface, to these distributed resources. Users consider the Grid as a single computational resource. Resource management software, frequently referenced as *middleware*, accepts jobs submitted by users and schedules them for execution on appropriate systems in the Grid, based upon resource management policies. Users can submit thousands of jobs at a time without being concerned about where they run. The Grid may scale from single systems to supercomputer-class compute farms that utilise thousands of processors. Depending on the type of applications, the interconnection between the Grid parts can be performed using dedicated high-speed networks or the Internet.

By providing scalable, secure, high-performance mechanisms for discovering and negotiating access to remote resources, the Grid promises to make it possible for scientific collaborations to share resources on an unprecedented scale, and for geographically distributed groups to work together in ways that were previously impossible. Several examples of new applications that benefit from using Grid technology constitute a coupling of advanced scientific instrumentation or desktop computers with remote supercomputers; collaborative design of complex systems via high-bandwidth access to shared resources; ultra-large virtual supercomputers constructed to solve problems too large to fit on any single computer; rapid, large-scale parametric studies, e.g. Monte-Carlo simulations, in which a single program is run many times in order to explore a multidimensional parameter space.

The Grid technology is currently under intensive development. Major Grid projects include NASA’s Information Power Grid, two NSF Grid projects (NCSA Alliance’s Virtual Machine Room and NPACI), the European DataGrid Project and the ASCI Distributed Resource Management project. Also first Grid tools are already available for developers. The Globus Toolkit [20] represents one such example and includes a set of services and software libraries to support Grids and Grid applications.

Acknowledgements

The author is grateful to the Institute of Solid State Physics, University of Latvia, and the EC Excellence Centre of Advanced Material Research and Technology for providing a financial support for the LASC cluster installation. This work was also partially supported by the Latvian Government Research Grants No. 01.0807 and 01.0821.

References

[1] http://www.cray.com/
[2] http://www.gridcomputingplanet.com/
[3] http://www.es.jamstec.go.jp/
[4] http://www.llnl.gov/asci/platforms/
[5] http://www.llnl.gov/linux/mcr/
[6] http://www.top500.org/
[7] http://www.myrinet.com/
[8] http://www.quadrics.com/
[9] http://www.infinibandta.org/
[10] http://hsi.web.cern.ch/HSI/sci/sci.html
[11] http://www.vidf.org/
[12] http://www.hp.com/techservers/systems/symc.html
[13] http://www.cs.wisc.edu/condor/
[14] http://www.epm.ornl.gov/pvm/pvm_home.html
[15] http://www-unix.mcs.anl.gov/mpi/
[16] Buyya R. (ed.) (1999) High Performance Cluster Computing: Systems and Architectures, Prentice Hall, New York.
[17] http://www.beowulf.org/
[18] http://openmosix.sourceforge.net/
[19] http://www.cfi.lu.lv/lasc/
[20] http://www.globus.org/

Received on the 25th of December 2003