Proc. 4st Int. Conf. "Information Technologies and Management", April 11-12, 2006 (Information Systems Management Institute, Riga, Latvia, 2006) pp. - .

LOAD-BALANCING TECHNOLOGY IN CLUSTER COMPUTING

ALEXEI KUZMIN

Institute of Solid State Physics of Latvian University
Kengaraga Street 8, LV-1010, Riga, Latvia
e-mail: a.kuzmin@cfi.lu.lv

ABSTRACT

Cluster computing becomes very popular approach used to provide solutions to problems that require significant computational power. An efficient use of cluster requires load balancing technology that optimizes cluster resources utilization. In this work we will describe a solution, which is used on Latvian SuperCluster (LASC) at the Institute of Solid State Physics of the University of Latvia. [*Keywords: cluster computing*, *load balancing*]

1. INTRODUCTION

Cluster computing refers to technologies that allow multiple computers, called cluster nodes, to work together with the aim to solve common computing problems [1,2]. A generic cluster architecture is shown in Figure 1. Each node can be a single or multiprocessor computer, such as a PC, workstation or multiprocessors server, equipped with its own memory, I/O devices and operating system. The nodes are interconnected by high speed local area network (LAN) such as, for example, Fast Ethernet or Gigabit Ethernet. To work as a single system, the cluster uses special software, called "cluster middleware" [1,2]. Note that cluster can be used for both sequential and parallel applications. However, to run parallel programs, the parallel programming environment (PPE) is required [1,2]. We will address these points in more details below.

Clusters became originally attractive with the availability of high performance commodity computers and message-passing software, which plays the role of PPE. Today most high-performance systems, often called "supercomputers", utilize cluster architecture. There have been totally 360 clusters in the recent Top500 Supercomputers list (as of November 2005) [3]. The first four places have been occupied by BlueGene/L, BGW, ASCI Purple and Columbia clusters (Figure 2), having performances up to several hundreds teraflops. It is worth to note that most (371) cluster systems in the Top500 list work under Linux operating system, that significantly reduces operation costs as well as increases compatibility between different installations.

An example of high-performance computing (HPC) cluster in Latvia is Latvian SuperCluster (LASC) [4,5,6], which has been installed at the Institute of Solid State Physics (ISSP) of the Latvian University (LU) in 2002. The current architecture of LASC is shown in Figure 3. It is used for theoretical simulations in the fields of solid state physics and quantum chemistry by about 20 users, having different software demands, that makes the question of optimization of cluster resources utilization very actual.

Proc. 4st Int. Conf. "Information Technologies and Management", April 11-12, 2006 (Information Systems Management Institute, Riga, Latvia, 2006) pp. - .

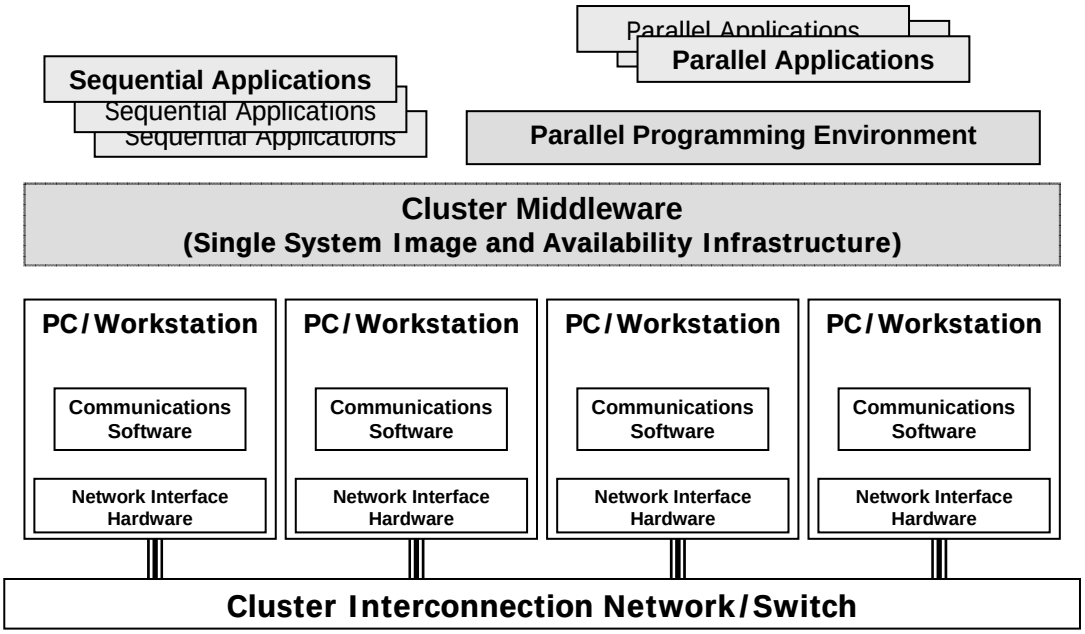
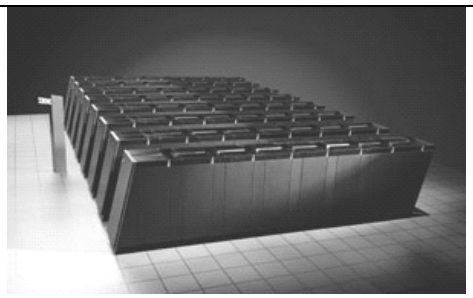
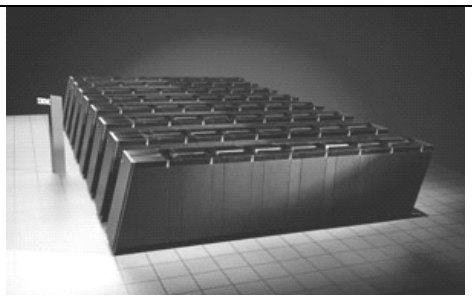


FIGURE 1. TYPICAL CLUSTER ARCHITECTURE [1].



BlueGene/L
IBM 0.7 GHz PowerPC 440
280.6 TFlops, 131072 processors



BlueGene @ Watson (BGW)
IBM 0.7 GHz PowerPC 440
91.29 TFlops, 40960 processors

ASCI Purple
IBM 1.9 GHz Power5 575
63.39 TFlops, 10240 processors

Columbia
SGI Altix 1.5 GHz
51.87 TFlops, 10160 processors

FIGURE 2. TOP SUPERCOMPUTER SYSTEMS IN NOVEMBER 2005 [3].

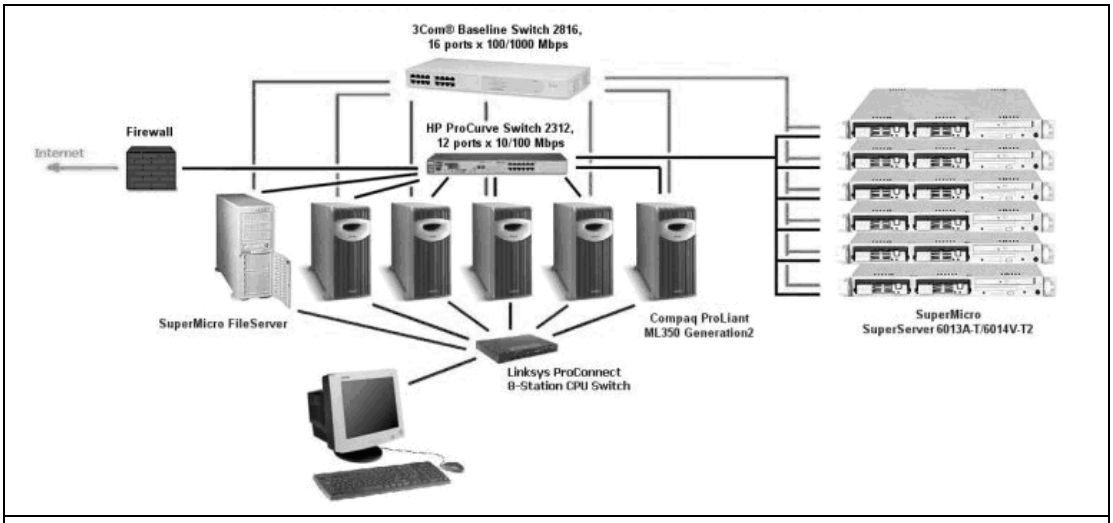


FIGURE 3. LATVIAN SUPERCLUSTER (LASC) ARCHITECTURE IN 2006 [6]. THE CLUSTER IS OPERATING UNDER REDHAT LINUX 9.0 WITH OPENMOSIX AND CONSISTS OF 22 CPUS (PIII,P4-XEON) INTERCONNECTED BY FAST AND GIGABIT ETHERNET NETWORKS. THEORETICAL PEAK PERFORMANCE IS ABOUT 48 GFLOPS. TOTAL CLUSTER RESOURCES ARE 44 GB OF OPERATING MEMORY (RAM) AND 3.0 TB OF HARD DISK SPACE.

In this paper we will review currently available technologies for cluster resources management and discuss our experience with load balancing technology, based on openMosix [2,7].

2. CLUSTER SOFTWARE TOOLS AND MIDDLEWARE

A cluster computer can be operated under different operating systems, including Linux, Unix and Windows [1,2,8]. However, to simplify its utilization for users and to optimize the use of cluster resources, a special cluster software tools as parallel programming environment (PPE), resource management and scheduling (RMS) system or middleware must be used. Examples of PPE are clustering libraries, as MPI (Message Passing Interface) [9,10] and PVM (Parallel Virtual Machine) [11,12] , or any other interfaces/libraries based on RSH (remote shell) and SSH (secure shell) technology. It is important to note that the use of PPE requires special design of applications by programmers, and the resources allocation is controlled by the user at the start-up time. This can result in a non-homogeneous load on cluster nodes especially in multi-users environment. To overcome this problem, the RMS system, e.g. openPBS, Condor or Libra [2,13], can be used to automatically control the load distribution among cluster nodes at the start-up time. Usually the RMS system looks like a batch system allowing to a user to submit a job through Web-interface. After job submission, the RMS system dispatches the job to the optimal cluster node(s) and returns the results of job execution back to the user via Web–interface. The drawback of RMS systems is the need of job description preparation and an inability to control dynamically (during execution) the cluster resources.

Significantly better solution can be achieved through the use of the “middleware” technology [1]. The middleware layer resides between the operating system and user-level environment and consists of essentially two sublayers (single system image (SSI) and system availability) of software infrastructure. The main goal of the middleware is to guarantee transparency, scalability and availability of cluster resources. Transparency means that the middleware offers a single system image (SSI) view of a cluster system, so that users can use a single entry point (one node) to access full cluster resources without thinking about the complexity of the cluster architecture. Scalability guarantees easy modification of cluster, i.e. addition/removal of new nodes, with automatic load re-distribution. Finally, availability

means automatic recovery from failures by using checkpointing and fault tolerant technologies as well as handling consistency of data upon replication between nodes.

SSI is the illusion, created by software or hardware, that presents a collection of resources as one, more powerful resource. The SSI offers several benefits such as transparent use of system resources, transparent process migration and load balancing across nodes, improved reliability and higher availability, improved system response time and performance, simplified system management, reduction in the risk of operator errors and no need to be aware of the underlying system architecture to use cluster nodes effectively. SSI can be implemented at different levels: (i) application and subsystem level, (ii) operating system kernel level and (iii) hardware level. The first type of SSI realization includes batch-type systems and system management (RSH, SSH), special libraries (MPI, PVM) or software (Parallel Oracle [14], Sun Cluster [15]). The hardware realizations include symmetric-multi-processors (SMP) techniques. Finally, the SSI can be incorporated into the operating system kernel as, for example, in Solaris-MC [15] and openMOSIX.

OpenMosix, a Multicomputer Operating System for Unix, is a Linux kernel extension for single-system image (SSI) clustering [2,7]. It uses adaptive load balancing techniques and allows processes running at one node in the cluster to migrate transparently to another node where they can execute faster. OpenMOSIX technology [2,7] is based on pre-emptive process migration mechanism and adaptive resources (processor, operating memory, network) allocation policy. One should note that openMosix cannot execute a single process on multiple physical CPUs at the same time, therefore 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. Moreover, openMosix allows a creation of automatically configurable cluster with dynamic architecture, having a variable number of computational nodes. This allows for a temporary increase of cluster computation power using idle laboratory computers, connected to the cluster.

TABLE 1. LIST O F OPENMOSIX LIVECD/DVD DISTRIBUTIONS AVAILABLE ON THE WEB.

Linux distribution	**Brief description**	**Internet address**
ClusterKnoppix	full openMosix Cluster with Knoppix and XFree	http://bofh.be/clusterknoppix/
Quantian	DVD for mathematical/scientific workstations	http://dirk.eddelbuettel.com/quantian.html
BCCD	for cluster computing education	http://bccd.cs.uni.edu/
Clusterix	Morphix-like distribution	http://clusterix.livecd.net/
Dynebolic	for multimedia production	http://bccd.cs.uni.edu/
CHAOS	CD based mini distribution	http://midnightcode.org/projects/chaos/
PlumpOS	CD based mini distribution	http://plumpos.sourceforge.net/
eucaristOS	floppy based distribution	http://eucaristos.sourceforge.net/
GoMF	floppy based distribution	http://gomf.sourceforge.net/
openMosixLOAF	floppy based distribution	http://openmosixloaf.sourceforge.net/

To take an advantage of openMosix technology, potential users can install modified Linux kernels on the stationary cluster nodes or use so-called “LiveCD” technology, which allows a temporary conversion

of a group of interconnected computers into a cluster. Today several openMosix-powered distributions (see Table 1) are readily available on the Web, allowing for a simple cluster configuration process on the fly. Each distribution is supposed to be used for a particular goal and therefore could contain some related software. In short, openMosix-based distributions can be divided into two groups: (1) full distributions, which include graphical Windows-type interface as XFree and possibly other cluster-oriented software, and (2) CD or floppy based mini distributions, which can be used just to start-up a node in a cluster.

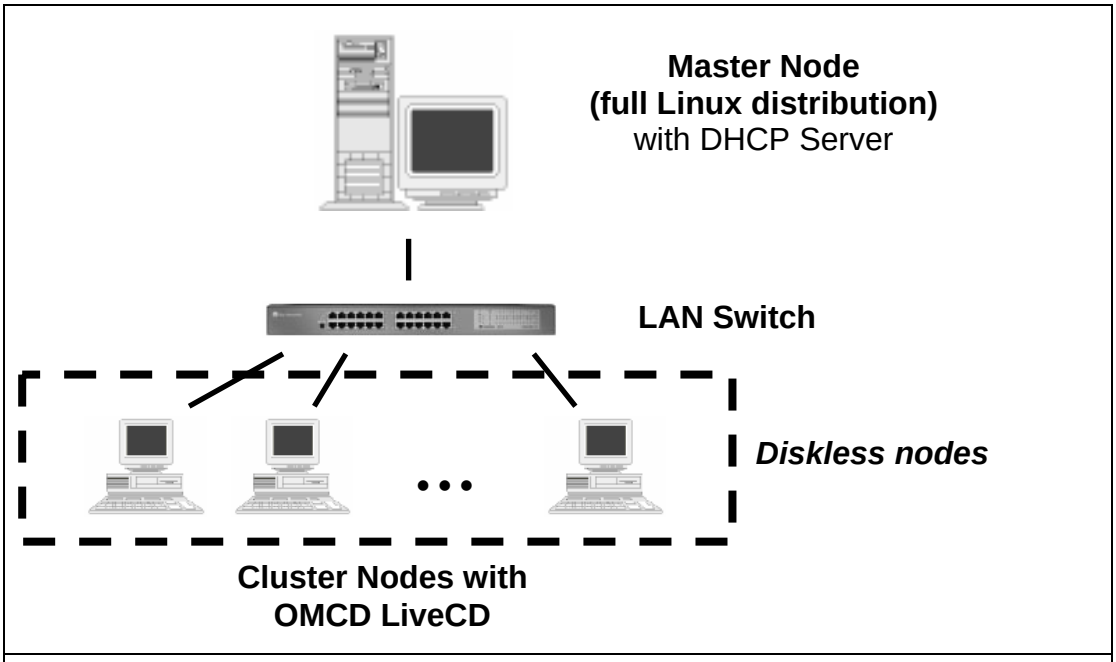


FIGURE 4. OPENMOSIX CLUSTER SET-UP WITH OMCD LIVECD [6].

Another LiveCD distribution, based on Linux RedHat 9.0 and being compatible with the LASC, has been prepared at ISSP and is called OMCD [6]. It can be used for dynamic expansion of the LASC system with new nodes or instant openMosix-type cluster construction using, for example, computing resources available in the computer classes. In the last case, a possible cluster set-up is shown in Figure 4, where a group of diskless nodes is connected to the central ("Master") node through a network switch. The openMosix cluster can be configured using two ways: (1) manually by providing each cluster node with a list of all nodes IP addresses/names, or (2) automatically via the use of autodiscovery daemon, called "omdiscd" [2,7]. Also an important point is that all nodes in a cluster must use the same openMosix version. The OMCD distribution uses autodiscovery approach together with automatic allocation of the nodes IP addresses (through the DHCP server technology), so that no administrator intervention is required during cluster set-up. To use such cluster, the user(s) must connect to and run their applications on the central node. During execution time, the openMosix will take care to distribute all running processes among available computational nodes (including the central node) and will continuously try to maintain homogeneous load at all nodes. After program termination, the results will be automatically returned to the central node. Thus, this technology is completely transparent for users and does not require any modification of the user programs as well as a preparation of any additional job description files.

Proc. 4st Int. Conf. "Information Technologies and Management", April 11-12, 2006 (Information Systems Management Institute, Riga, Latvia, 2006) pp. - .

3. CONCLUSIONS

Load balancing technology, based on openMosix [2,7], is easy to use and allows to optimize automatically the utilization of cluster resources. Linux cluster with openMosix is reliable platform for high-performance computing with dynamically variable resources. It can be recommended for the use in educational environment, including teaching process, as well as for instant construction of temporary clusters.

ACKNOWLEDGEMENT

This work was supported in part by the Latvian Government Research Grants No. 05.1717.

REFERENCES

[1] Buyya R. (ed.) (1999) *High Performance Cluster Computing: Architectures and Systems.* Prentice Hall PTR.
[2] Sloan J.D. (2004) *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI.* O'Reilly.
[3] See http://www.top500.org/
[4] Kuzmin A. (2003) Cluster approach to high performance computing. *Computer Modelling and New Technologies* **7,** 7-15.
[5] Kuzmin A. (2004) *Latvian SuperCluster - LASC: recent developments*. In Proc. 2nd Int. Conf. "Information technologies and management 2004", Information Systems Management Institute, Riga, Latvia.
[6] See http://www.cfi.lu.lv/lasc
[7] See http://openmosix.sourceforge.net/
[8] Sterling T. (2003) *Beowulf Cluster Computing with Windows (Scientific and Engineering Computation)*. Wiley.
[9] See http://www-unix.mcs.anl.gov/mpi/
[10] Шпаковский Г.И., Серикова Н.В. (2002) *Программирование для многопроцессорных систем в стандарте MPI*. БГУ, Минск.
[11] See http://www.epm.ornl.gov/pvm/pvm_home.html
[12] Немнюгин С., Стасик О. (2002) *Параллельное программирование для многопроцессорных вычислительных систем*. БХВ, Санкт-Петербург.
[13] Buyya R. (ed.) (1999) *High Performance Cluster Computing: Programming and Applications*. Prentice Hall PTR.
[14] Mahapatra T., Mishra S. (2000) *Oracle Parallel Processing*. O'Reilly.
[15] See http://www.sun.com/