TRANSFORMING COMPUTER LAB INTO A MINI HPC CLUSTER

Jovan Janevski, Dragan Jovanovski, Josip Kolic, Jasna Veljanovska, Elvis Imeroski, Leonid Djinevski, Sime Arsenovski
FON University
Skopje, Macedonia

ABSTRACT

In almost any education institution there are computer labs that are available to students during work hours. However, during off hours, these labs are underutilized machines which contain significant computing performance. In this paper we are present the cluster infrastructure that we implemented for utilizing one of the computer labs at FON University. Additionally we implemented a simple micro-benchmark in order to present the computing capabilities of our cluster.

I. INTRODUCTION

Computer cluster is a set of connected computers that work together and can be view as a single system. Components of the cluster are usually interconnected between them with fast connections.

Recently, influenced of fast information technology development, the number of computer clusters is growing. This is based on the need for computational resources on one hand, and the hardware that is becoming more affordable.

There are several implementations of computer clusters for different operation systems and environments. For GNU/Linux there is various of cluster software which support GNU/Linux for application clustering like Beowulf [1][2], distcc [3][4] and MPICH Linux Virtual Server [5]. MOSIX [6][7], openMosix [8][9], Kerrighed [10][11], Open SSI [12] are clusters implemented into the kernel that provide automatic process migration between the homogeneous nodes. Microsoft Windows Compute Cluster Server 2003 [13][14] which is based on Windows Server have some parts from High Performance computing like the Job Scheduler, MSMPI library and management tools.
In this paper we present an implementation of a cluster build over a commonly equipped computer science laboratory. During work hours, the lab computers are used for education purposes, while off hours the lab computers form a cluster that is used for high performance computing (HPC). We propose node management scenarios for the presented cluster, and we perform a micro-benchmark in order to evaluate the performance and limits of our computing infrastructure.

This paper is organized as follows: Section 2 present a short overview of our cluster infrastructure. In the same section we propose and present our implementation of power management scenarios. A short overview of benchmarking tools is presented in Section 3, followed by the description of our micro-benchmark based on the matrix multiplication algorithm. The methodologies and hardware platform used are described in Section 4. The obtained results are discussed in Section 5. A conclusion is presented in Section 6.

II. INFRASTRUCTURE

The HPC cluster is organized as presented in Fig. 1. Job scheduling is performed by the portable batch system (PBS) implementation called TORQUE Resource Manager [15], and maintained and commercially supported by [16]. Cluster Resources, Inc.TM, and includes contributions from NCSA, OSC, USC, U.S. Department of Energy, Sandia, PNNL, University of Buffalo, Teragrid and many other HPC institutions. TORQUE offers improved usability and scalability over previous derivatives, and has continued to grow based on the value it provides no cost. The TORQUE Resource Manager is a distributed resource manager providing control over batch jobs and distributed compute nodes. TORQUE is built on the principles of the Portable Batch System, also known as PBS, from which there are two versions OpenPBS (open-source but no longer maintained) and PBS Pro, a paid for PBS software. TORQUE is also open source and derives from OpenPBS, with currently many years of development separating TORQUE from OpenPBS. TORQUE is currently one of the better ways to manager cluster job queues, especially given that Sun Grid Engine is no longer free.

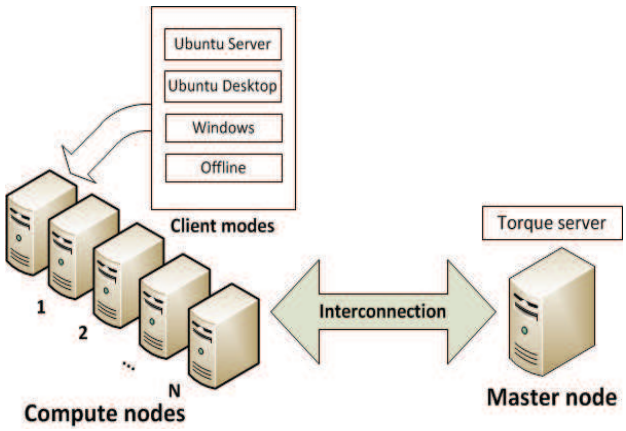

Figure 1 PBS Infrastructure

All machines in the cluster are interconnected with fast connections. Fig. 1 depicts the states in which the compute nodes can be found: active, passive, offline and busy.

A. Power Management Technique

The purpose of the other nodes in the cluster is solely as compute nodes. The active state is installed with Ubuntu Desktop 12.04 and the passive compute node is installed with Windows. The first scenario TORQUE server provides LAN

wake up command. Many devices export native wake-up control. In particular, modern Ethernet Network Interface Cards (NIC) support Wake-on-LAN (WOL) and their drivers export that function via eth-tool. The second scenario TORQUE server provides remote command to shutdown a certain Compute nodes, and the third scenario TORQUE server gives the command to restart the Windows Compute nodes in the Linux Operating System.

B. Portable Batch System

Job scheduling is performed by the portable batch system (PBS) implementation called TORQUE Resource Manager. Management and synchronization of resources is performed by WebDAV (Web Distributed Authoring and Versioning) [17], which is a distributed file system based on the HTTP protocol that consists of a set of methods, headers and content types characteristic to HTTP/1.1.

Our cluster requires sharing of the same directory subtree structure, which means having a networking file system is a must. For the convenience and maximum efficiency WebDAV is installed on the server node, and the shared directory is mounted on the client nodes. As a network file sharing solution WebDAV is chosen primarily due to its standardized nature. The mounting of the davfs2 is done through Uniform Resource Locator. Let's start from nomenclatural hierarchy - the client is set to look for the IP of a specific domain name - which is the server. This is done through the DNS protocol. The DNS protocol is the foundation of logical nomenclature of the internet as we know it. Thus it's well documented, tested and implemented everywhere making it a standardized naming system. Furthermore, the client has the IP address on the networking layer. From the URL on transport layer the client obtains the port too, in standard solution this port is either 80 for plaintext or 443 for encrypted connections both ports are common. Then it uses extended HTTP protocol to establish file structure, modify file structure and read/write/execute files.

Using well defined, DNS protocol, common ports that are usually open on firewalls, and standardized HTTP protocol, WebDAV is prevalent among other solutions, which require non-standard firewall open ports, or require additional name resolution protocol as WINS.

III. Benchmarking the Cluster

Benchmark is a standard or a set of standards used as a point of reference for evaluating performance or level of quality.
The HPC Challenge Benchmark is a set of benchmarks whose role is to test many attributes which are essential for the performance of HPC computers. This benchmark consists of seven tests. High Performance Linpack (HPL) [18] is a software package that solves linear system of equations of order *n*. It uses implementations such as MPI (Message Passing Interface) [19] and BLAS (Basic Linear Algebra Subprograms) [20]. The HPL package is used for testing and timing program to quantify the accuracy of the solution as well as the time it took to compute that solution. DGEMM [21] and PTRANS [22] measure the floating-point execution rate of double precision real matrix-matrix multiplication and the total communication capacity. STREAM benchmark [23] is used for the measurement of computer memory bandwidth from ordinary user programs.

Another widely adopted benchmark is the Standard Performance Evaluation Cooperation [24] is formed to establish and maintain a standardized set of benchmarks that can be applied to the high performance computers. System Performance Evaluation Cooperative is used for measuring and evaluating workstation performance. There are few releases of SPEC benchmarks: SPEC89, SPEC92, SPEC95, SPEC2000 and SPEC2010. The first set of ten benchmark programs is released in 1989. It had about 150,000 lines of source code. Later, in 1992, it was released SPEC92. Until 2010, SPEC released three updated benchmark suites, each of them describing a standard reference machine and a set of program to run and formulas for computing the scores.

A. Matrix Multiplication Micro-benchmark

Matrix multiplication is one of the most commonly used algorithms in computing, thus we decided to utilize this algorithm in order to micro-benchmark our HPC cluster. The product matrix multiplication algorithm takes two matrices A_{mxl} and B_{pxn} as input data, and produces an output matrix $C = A * B$ as it is defined in (1). It is very important to state that both input matrices can be multiplied only when the number of columns in the first matrix equals the number of rows in the second matrix p.

$$C_{ij} = \sum_{i=1}^{m} \sum_{k=1}^{l} \sum_{j=1}^{n} a_{ik} b_{kj} \quad (1)$$

where a_{ik}, b_{kj} and c_{ij} are corresponding elements of the matrices A, B and C, for $i = 0..m-1$, $k = 0..l-1$ and $j = 0..n-1$. For simplicity we use square matrices for A, B and C, thus equation (1) transforms into equations (2)

$$C_{ij} = \sum a_{ik} b_{kj} \quad (2)$$

where a_{ik}, b_{kj} and c_{ij} are corresponding elements of the matrices A, B and C, for all $i,j,k=0..N-1$.

IV. Testing Methodologies

A Our hardware platform consists of 24 nodes, where each node consists of a dual-core Intel Pentium E5300 processor at 2.6GHz, with 32KB of L1 cache memory, 2MB of L2 cache memory, 4GB of RAM memory, and 500GB of hard drive storage. The nodes are interconnected with model network adapters, in a 100Mbit Ethernet network. The rest of the

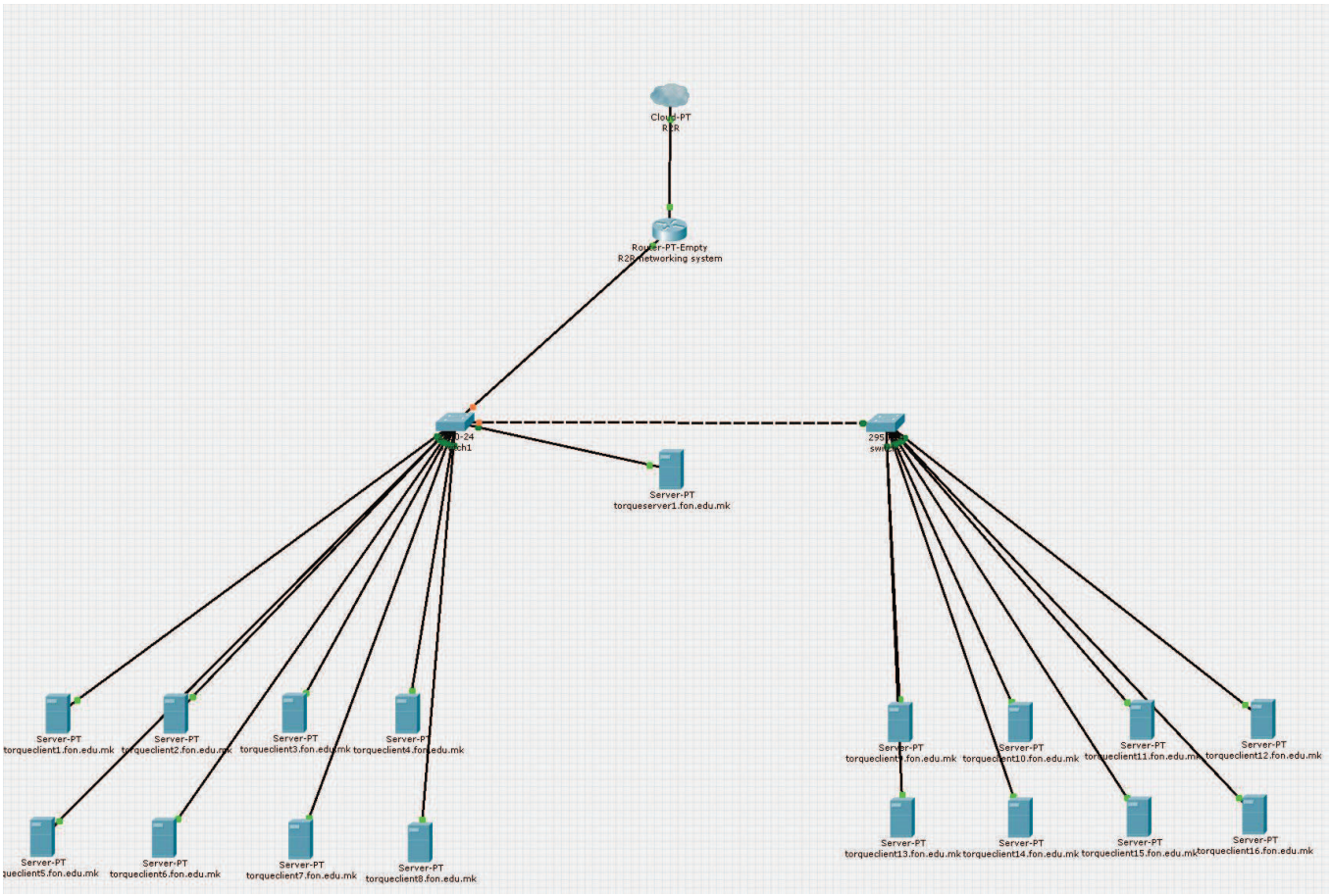

Figure 2 Normalized speed for both approaches for the case $wA = hB = 32$

network specification is presented in Table 1, while the network topology is depicted in Fig 2. The server node and the client nodes are connected physically through cables to the network switches. The two network switches are connected to each other via another cable (local pipe). The second switch uplink is connected with the first switch. The first switch uplink is connected to the R2R network system. The system Layer2 communication is done via Ethernet frames. All machines in the current configuration belong to the same Ethernet broadcast domain. The machines are assigned static IP addresses via DHCP protocol from the R2R networking system. Also static DNS entry is created - A pointer. The network communication is done within the local subnet, with the rest of the networks and the internet via the R2R networking system.

Jobs are submitted on the server and the server assigns the jobs to the nodes. On the server node among the other things there are Torque Server installed and Apache with WebDAV configured. On the client nodes among the other things there are Torque Client installed and davfs2 mount client. In an equivalent way, the file synchronization is done through WebDAV. On transport layer, the server node listens on TCP port 15001, the clients on TCP 15002 and 15003. WebDAV uses standard HTTP TCP 80 port, DHCP UDP 67 and 68 and DNS TCP/UDP 53.

For message passing we use an open-source implementation of OpenMPI version 1.4.3 for distributed communications between nodes in the cluster.

Table 1:

Part	Name/specs	Quantity
Layer1	cat5e 15m cable	30
Layer2	Netgear generic 10/100Mbps full duplex IEEE802.3 unmanaged switch	2
Layer3	R2R network system	N/A

We obtain the performance results by executing 10 iteration of our micro-benchmark for each test. For the software scalability we run the benchmark at full hardware infrastructure (all 24 nodes online), for the following data requirements of matrix size N = 64, 128, 256, 512, 1024, 2048. Regarding the hardware scalability, we perform our tests for a single data requirement of matrix size N = 1024, while varying the number of used processors.

V. RESULTS

In this section we present the results obtained from running our micro-benchmark on our cluster. We have chosen 5 cases for matrix sizes of N = 512, 1024, 2048, 3072, and 4096. For each matrix size N, two sets of tests were performed. In the first set of tests, one processor is available per node, thus we obtain the scalability up to 18 processors as presented in Fig 3.

The second set of tests, two processors were available per node, thus the scalability is up to 36 processors as is charted in Fig 4. It is clear from both charts that the cluster resources are underutilized, which is because of the slow memory transfers over the slow 100Mbps network.

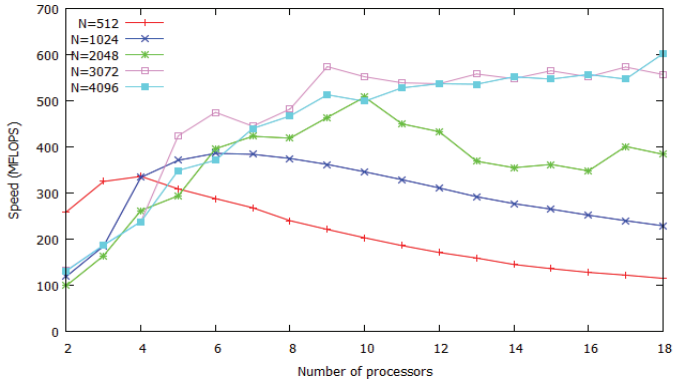

Figure 3 Scalability of the cluster for one processor per node

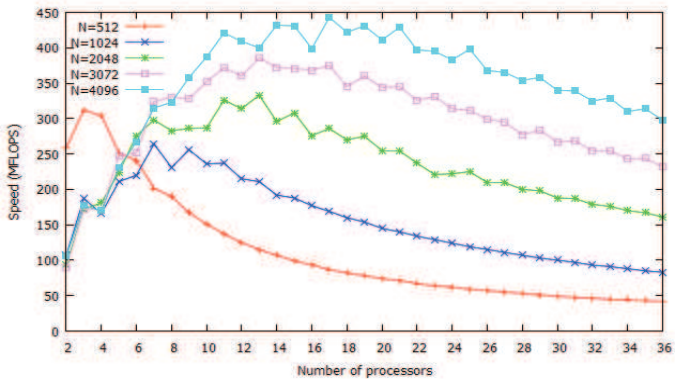

Figure 4 Cluster scalability for two processors per node

VI. CONCLUSION

In this paper we propose a cluster infrastructure for utilizing the resources of a computer lab FON University. A micro-benchmark was implemented and the cluster performance was analysed. The results show that the available cluster resources are underutilized. The bottleneck is the communication time increases with the increase of computer nodes and processors. As future work, we plan to upgrade our network infrastructure to Gigabit Ethernet. Additionally, we plan to implement and analyses the performance of BLAS LAPACK, and other fast linear algebra libraries, and the CUDA Toolkit in order to additionally utilize the GPGPU resources.

REFERENCES

[1] T. Sterling, et al. "*How to build a Beowulf: A Guide to the Implementation and Application of PC Clusters*". MIT press, 1999.

[2] J.R.R Tolkien, "*Beowulf: the Monsters and the Critics*", Oxford University Press, 1936.

[3] Martin Pool, *"distcc, a fast free distributed compiler"*, 2003.

[4] J. Hall, *"Distributed computing with distcc"*, Linux Journal 2007.

[5] J. Mack, "*LVS-HOWTO*" homepage, Retrived February 2013 http://www.austintek.com/LVS/LVS-HOWTO/.

[6] G.B. Amnon, R. Wheeler. "*MOSIX: An integrated multiprocessor UNIX"*, Mobility, ACM Press/Addison-Wesley Publishing Co., 1999.

[7] G.B. Amnon, O. La'adan. "*The MOSIX multicomputer operating system for high performance cluster computing*", Future Generation Computer Systems 13.4 (1998): 361-372.

[8] K. Buytaert, "*The OpenMosix HOWTO*", The Linux Documentation Project (2002).

[9] J.M. Meehan, A. Wynne. "*Load balancing experiments in openmosix*", Computers and Their Applications (2006): 314-319.

[10] C. Morin, et al., "*Kerrighed: a single system image cluster operating system for high performance computing"*, Euro-Par 2003 Parallel Processing (2003): 1291-1294.

[11] R. Lottiaux, et al., "*Openmosix, openssi and kerrighed: A comparative study*", Cluster Computing and the Grid, CCGrid 2005. IEEE International Symposium on. Vol. 2. IEEE, 2005.

[12] P. Osinski, E. Niewiadomska-Szynkiewicz. "*Comparative Study of Single System Image Clusters*", Prace Naukowe Politechniki Warszawskiej. Elektronika 169 (2009): 145-154.

[13] Russe, Charlie. "Deploying and managing microsoft windows compute cluster server 2003." httlx//technet. microsof t. com/en—us/library/cc720097. aspx (2005).

[14] Russel, Charlie. "Overview of Microsoft Windows Compute Cluster Server 2003." White Paper, Microsoft Cooperation (2005).

[15] Nakada, Hidemoto and Takefusa, Proc. 3rd International Workshop on Grid Computing and Applications, An advance reservation based computation resource manager for global scheduling, 2007.

[16] Nakada, Hidemoto and Takefusa, Atsuko and Ookubo, Computer and Information Technology, 2006. CIT'06. The Sixth IEEE International Conference on, 65—65, IEEE, 2006.

[17] Goland, Yaron and Whitehead, E and Faizi, Asad and Carter, Steve and Jensen, Del. HTTP Extensions for Distributed Authoring – WEBDAV, RFC 2518, IETF, Feb (1999).

[18] Jack J. Dongarra, Piotr Luszczek, and Antoine Petitet. The LINPACK benchmark: Past, present, and future. Concurrency and Computation: Practice and Experience, 15:1–18, 2003.

[19] MPI Forum. MPI: A Message-Passing Interface Standard. International Journal of Supercomputer Applications 8(3/4), 165–416 (1994).

[20] S. Blackford, G. Corliss, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, C. Hu, W. Kahan, L. Kaufmann, B. Kearfott, F. Krogh, X. Li, Z. Maany, A. Petitet, R. Pozo, L. Remington, WALSTER, W., C. Whaley, V. Wolff, J. Gudenberg, A. Lumsdaine, Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard, International Journal of High Performance Computing 15, 3–4, 2001.

[21] Jack J. Dongarra, J. Du Croz, Iain S. Duff, and S. Hammarling. Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms. ACM Transactions on Mathematical Software, 16:1–17, March 1990.

[22] Daisuke Takahashi and Yasumasa Kanada. Highperformance radix-2, 3 and 5 parallel 1-D complex FFT algorithms for distributed-memory parallelcomputers. The Journal of Supercomputing, 15(2):207–228, 2000.

[23] John McCalpin. STREAM: Sustainable Memory Bandwidth in High Performance Computers. (http://www.cs.virginia.edu/stream/).

[24] Dixit, Kaivalya M. "The SPEC benchmarks." Parallel Computing 17.10 (1991): 1195-1209.