IMPLEMENTATION AND PERFORMANCE ANALYSIS OF MPI CLUSTER SYSTEM IN DISTRIBUTED RENDERING

Erick Irawadi Alwi[1], Teguh Bharata Adji[2], Sujoko Sumaryono[3]

[1] Dept. of Informatics Engineering, Faculty of Computer Science, Muslim University Of Indonesia (UMI)
Jln. Urip Sumoharjo KM 5 Makassar 90231 INDONESIA

[2,3] Dept.of Electrical Engineering, Faculty of Engineering, Gadjah Mada University (UGM)
Jln. Grafika 2 Yogyakarta 55281 INDONESIA

E-mail: [1]erick.alwi@umi.ac.id, [2]adji@mti.ugm.ac.id, [3]sujoko@mti.ugm.ac.id

***Abstract*—Graphic image is compulsory in delivering information and communication in this information technology era. Graphic applications are used widely to manipulate images in advertising industries and in animation movie productions. In order to manipulate numerous and complex images, it is required long period of time and high-performance computers. However, computers with high-performance specification are still very expensive these days. This condition triggers the development of distributed processing technique using the implementation of computer cluster which does not require very advanced specifications. A computer cluster is a computer system that employs many computers (PCs) which are connected within a computer network and work simultaneously in parallel for solving given computational processes.**

MPICH2 cluster system was implemented in cloud computing infrastructure in the laboratory. A performance analysis was conducted on how the system handle parallel distributed processes using POV-Ray software. The performance of the system was then tested, using execution time, efficiency rate, and speedup as the parameters.

This research used 4 sample images tested in MPICH cluster system for rendering. After speedup and efficiency rate analyses were conducted, it is then possible to conclude that the produced average speedup is 2.68 and the average efficiency rate in rendering the four 3D images is 92%. Those results show that the cluster system of the distributed rendering is efficient in executing the computational processes.

Keywords: Distributed rendering, Computer Cluster, Parallel computing, MPICH2, Cloud Computing

I. INTRODUCTION

Graphic image is compulsory in delivering information and communication in this information technology era. Graphic applications are used widely to manipulate images in advertising industries and in animation movie productions. In order to manipulate numerous and complex images, it is required long period of time and high-performance computers. However, computers with high-performance specification are still very expensive these days. This condition triggers the development of distributed processing technique using the implementation of computer cluster which does not require very advanced specifications.

Computer cluster is a computer system that employs many computers (PCs) which are connected within a computer network and work simultaneously in parallel for solving given computational processes [9].

The cluster system has been developing so fast and has now been used by various institutions. The main reason is because of the advantages promised by the cluster system, such as low in cost but the computational performance is comparable to that of mainframe computers [11].

One possible way to implement the parallel computing concept is in the cluster system. Basically, a cluster system implements the concept how to run a single process by collaborating many software and hardware that are connected to that particular cluster system. Distributed rendering, in general, can be described as a process that employs the advantages of parallel computing in order to execute its rendering processes.

II. LITERATURE REVIEW

Parallel computing is one of the most interesting technologies since the invention of electronic computers in the years of 1940s. Breakthroughs in the parallel processing keep increasing and gain a significant place besides other technologies since Renaissance Era (1950s), Mainframe Era (1960s), Minis Era (1970s), PC Era (1980s), and Parallel Computer Era (1990s). The influence of the development of other technologies and of how technology changes people's perceptions toward computer, it is then possible to understand the importance of the parallel computing. The core of parallel computing are its hardware, software, and applications. Parallel processing is an information processing method that puts more emphasize on how to manipulate the average of data elements against one or more processes of certain problem. Therefore, parallel computing can be defined as computers with many processors that are capable of executing parallel processing [12].

Parallelism has been acknowledged as the solution in making the computer works even much faster. It may seem that

978-1-5090-5548-7/16/$31.00 ©2016 IEEE

the limit of electronic devices in increasing the computer speed has not been found. However, it is a fact that each electronics component has its own physical limitation in increasing its performance. Sooner or later, such limit will be met and parallelism will be the only solution [5].

A parallel program needs a coordination system when one task depends on the other tasks. There are two types of coordination in the parallel computing: asynchronous and synchronous ones. The synchronous coordination requires all tasks to be undertaken at the same time while setting aside the reality of dependency among tasks. The asynchronous coordination utilizes locking mechanism in coordinating the processors without having to be running at the same time.

A. Cluster

Cluster is an integration of a group of data with similar correlation or characteristic. It means that PCs and servers (and other devices with operating systems) are integrated into a single computational unit. In addition to the usage of computer networks, the fundamental characteristic of clusters is how to utilize such configuration to solve problems. The other important characteristic is how to handle a certain task simultaneously using the configuration within the cluster. That what makes cluster system different from ordinary network mechanism that runs client-server configuration. Even though sometimes it may seem confusing to differentiate between the two, they are different. The cluster puts more focus on the integration of operating system and hardware in order to fully utilize resources available of each node, namely processors, memories, and hard disks [15]. There are two dominant types of cluster, which are:

a) High Performance Computing (HPC). Generally, HPC clustering type focuses on how to accelerate certain computation process, thus the task is executed faster. A good example of this clustering type is openMosix.
b) High Availability (HA). In general, this clustering type is aimed to ensure that every program that runs on it is running continuously, even though one of its nodes is crashed or down. One very common example of this clustering type is Apache web server, which is embedded with a redirector, so that when one of its servers is down, another server can directly be back up the work.

B. Message Passing Interface

Message passing interface (MPI) is an independent language in communication protocol used by parallel programs in the computer cluster [10]. MPI is a standard in message passing-based parallel computing that is implemented in several parallel programming environments due to its independent characters. One of its implementation in computer cluster networks is MPICH2 application.

MPICH2 is a framework that implements MPI according to MPI-1 and MPI-2 standards [11]. MPICH2 is developed by Mathematics and Computer Science Division in Argonne national laboratory.

MPICH2 is the further development of MPICH1 that used MPI1 for message passing library. MPICH2 is able to implement both MPI1 and MPI2 as the development of MPI1. MPI stands for Message Passing Interface while CH is derived from *Chameleon*. The portability layer that was used by previous MPICH in order to provide portability for message-passing systems.

C. MPICH2 Features

a) A full implementation of MPI1 and MPI2, including the limitation of message passing.
b) Supporting MPMD (*Multiple Program Multiple Data*) programs and *cluster* with different platforms.
c) C++ standard binding in MPI1 is available in MPI1 functions.
d) Providing Fortran 77 and Fortran 90 bindings, including both types of 'mpif.h' and MPI module.
e) The open source version of MPICH is available on Windows NT.
f) Supporting many types of environments, including SMP clusters and very large parallel computers.
g) Following many target recommendation of GNU in case of building and installing, including VPATH.
h) Parts of MPI2 support:
   – Most parts of MPI-IO are supported by the ROMIO implementation.
   – MPI_INIT_THREAD (only MPI_THREAD_SINGLE and MPI_THREAD _FUNNELLED).
   – Various types of MPI_Info and MPI_Datatype.
i) MPICH also provides components of parallel programming environments, including:
   – Tools to trace and to record logs, based on interface profiles.
   – MPI, including changeable log file formats (SLOG). Tools for parallel performance visualization (upshot and jump shot).
   – Extensive performance and value correctness testing.

D. Parallel Performance Parameters

In this research, the performance analysis is based on the parallel processing performance in executing distributed rendering within a cluster system network. The goal of parallel processing is to solve speed and memory capacity constraints. As in computation within a single processor, the parallel computing performance is also influenced by programming techniques, computer architectures, or both. Two important performance parameters in evaluating parallel system are speedup and efficiency [3].

a) Speedup

Speedup is the comparison between the fastest execution time of sequential algorithm and that of parallel algorithm, or execution time of parallel algorithm within a single processor and that in a certain number of processors. Mathematically, speed up can be defined with the following Equation (2.1):

$$S(n) = \frac{T(1)}{T(n)} \quad \text{..... (2.1)}$$

where,
$S(n)$: *speedup*
$T(1)$: execution time on a single processor
$T(n)$: execution time on n processors

b) Efficiency

In measuring the performance of a parallel system, efficiency also needs to be considered in addition to speedup. Mathematically efficiency can be defined by Equation (2.2) as follows:

$$E(n) = \frac{S(n)}{n} \quad \text{..... (2.2)}$$

$E(n)$: efficiency
$S(n)$: speedup
n : number of processors

E. Rendering

Rendering is a process of utilizing software in order to produce images from certain models created previously. The m model used to provide digital image is a description of certain three dimensional object. Moreover, several information that are important for rendering purpose are point of view of the camera, geometry, lightning, surface characteristic, and rendering algorithm that will produce two dimensional images.

III. RESEARCH METHODOLOGY

This research was conducted under experimental method by implementing a cluster system using MPICH2 middle-ware. It employed a descriptive analysis method which was undertaken by administering direct investigation and by analyzing the observation result over the occurrences within the system. This descriptive analysis approach was used to mine the information from MPICH2 cluster system implementation.

A. Research Tools

In this research, an MPICH2 tool was used to carry out the distributed rendering process in the cluster system. Whereas the hardware and operating system specifications used were as follows:

1. Hardware

Hardware that were used as master node and slave nodes in building the cluster network system were those with following specifications:

a) Three computers as Nova-Computers and Nova-volume with following specifications:
- Intel Xeon Quad Core E5430 processor (2,66 GHz, 4 Core, VT, 64bit)
- 8 GB DDR3 PC 5300 Memory
- 1 TB SCSI HDD
- VGA Intel on-board
- 2 NIC 100/1000 Gigabit Ethernets

b) One server as Nova-computer, Nova-Network, Nova-Volume, and Nova-Scheduler.
- Intel i7 Quad Core E5430 processor (2,66 GHz, 8 Core, VT, 64bit)
- 8 GB DDR3 PC 5300 Memory
- 1 TB SCSI HDD
- VGA NVidia 1024 Mb
- 2 NIC 100/1000 Gigabit Ethernet

c) Five Personal Computers (PCs) Virtual Machines with following specifications:
- QEMU Virtual CPU
- 1 GB DDR2 RAM Memory
- 70 GB HDD

d) One Single PC for comparison purposes, with following specification:
- Intel Core i7-2600K 3,4 GHz processor
- 4 GB DDR2 RAM Memory
- 640 GB HDD
- LAN Card

e) One laptop with following specifications:
- Intel core 2 duo T6600 processor
- 4 GB DDR3 Memory
- 320 GB HDD
- VGA NVIDIA GeForce G 105M

2. Software

a) Linux Operating System with Ubuntu Distro, Linux Ubuntu 10.04—Lucid Lynx 32 bit.
b) Network file system (NFS)
c) OpenSSH server (slave node)
d) OpenSSH client (master node)
e) XRDP (remote desktop)
f) MPICH2
g) POV-Ray 3.5

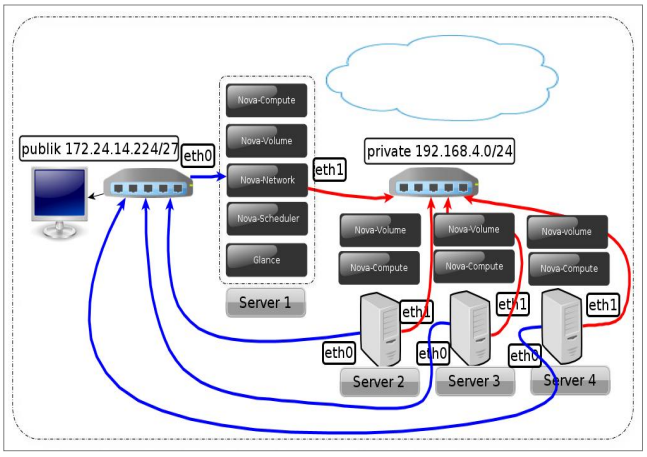

Fig. 3. 1 Cloud computing topology using *OpenStack*, as provided in MTI laboratory

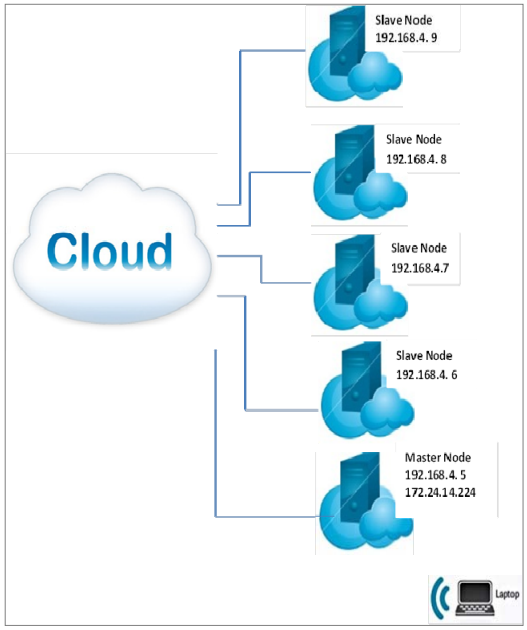

Fig. 3. 2 Architectural design of cluster system

B. Cluster System Design

This research was undertaken using the steps described previously, the first was implementing a cluster system. The cluster system was established in a virtualization provided in the cloud, then the cluster system architectural design with virtual computer provided by the cloud is represented in Fig. 3.2.

As shown by Fig. 3.2, the cluster system consisted of five virtual computers that had been provided by the cloud service, which included one master computer and four slave computers for MPI cluster system (MPICH2). Fig. 3.2 also shows one laptop for remote controlling to all virtual computers.

Virtual computers in the system were divided into two categories, master and slave computers. Master computer was responsible for delivering tasks to slave computers while slave computers were in charge of receiving and completing those given tasks. The master computer was also responsible for executing some particular tasks. The execution results from slave computers were then returned back to the master computer. In carrying out such mechanism, it was required also to run SSH and NFS services as well as the cluster system in order to integrate all nodes within the cluster.

POV-Ray application was used for rendering process, which was also integrated in the clustering system, so that the rendering process was turned into distributed rendering. The tasks were then spit into several pieces and disseminated to all nodes for further processing. This POV-Ray application was installed in all nodes and the distributed rendering process was the object for measurement, in terms of its speedup time and efficiency rate.

The network in the cluster system was set up using *Star* logical topology. It was very advantageous for further development, in which it was possible to add or reduce the number of nodes to/from the running system without having to disturb existing network activities.

IV. RESULT AND DISCUSSION

A. Testing and Analysis

The goal of testing design of the cluster distributed rendering in this research was to test its ability in executing computational processes. Parameters used within the research were speedup, efficiency, and execution time. The cluster testing was undertaken using MPICH2 for tasks delivery to the slave nodes.

The testing was conducted by running the rendering process on four sample images with various qualities, according to previously given resolution. The first rendering was conducted with one node, then for each following rendering testing process, one additional node is registered.

From the result of previous tests, it is then possible to recapitulate the data into a table for further analysis, as shown by the following Table 4.1.

Table 4. 1 The data recapitulation of testing results

Parameter	Testing	Images			
		Makeegg	Chess	Plants_demo	Glass_demo
Execution Time (second)	I	124	1256	1217	34660
	II	59	620	633	23118
	III	38	414	424	14958
	IV	28	320	321	12227
	V	27	300	315	11596
Speedup	I	1	1	1	1
	II	2,102	2,026	1,923	1,499
	III	3,263	3,034	2,870	2,317
	IV	4,429	3,925	3,791	2,835
	V	4,593	4,187	3,863	2,989
Efficiency	I	1	1	1	1
	II	1,051	1,013	0,961	0,750
	III	1,088	1,011	0,957	0,772
	IV	1,107	0,981	0,948	0,709
	V	0,919	0,837	0,773	0,598

Where:
Testing I : 1 Node in the Distributed Rendering system
Testing II : 2 Nodes in the Distributed Rendering system
Testing III : 3 Nodes in the Distributed Rendering system
Testing IV : 4 Nodes in the Distributed Rendering system
Testing V : 5 Nodes in Distributed Rendering system

From obtained data in Table 4.1, graphical representations are then created for speedup and efficiency analysis of the distributed rendering system. The representations are shown by Fig. 4.1 and Fig. 4.2.

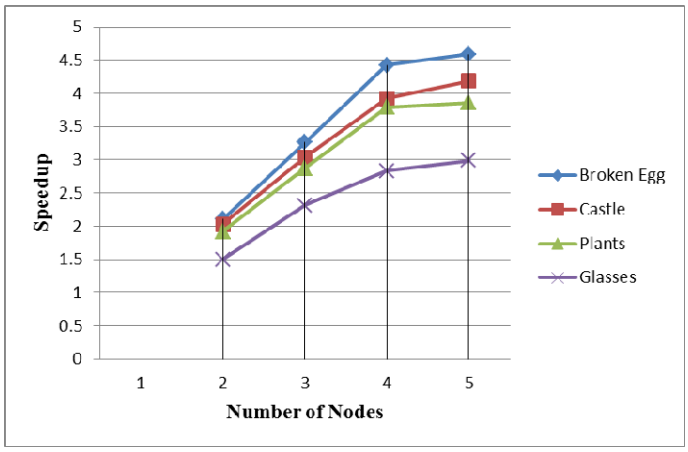

Fig 4. 1 Graphics of speedup value at four tested images

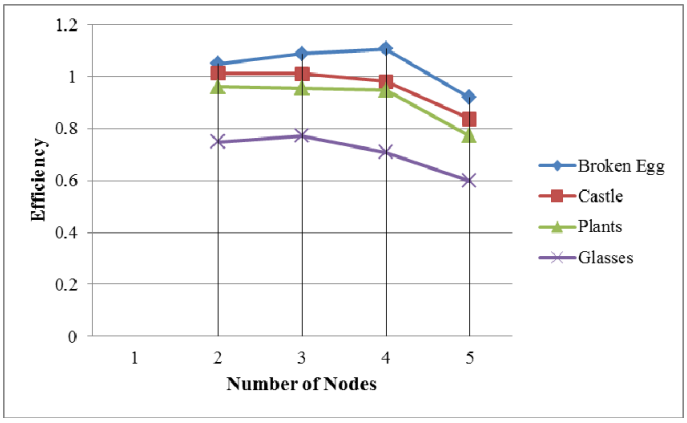

Fig. 4. 2 Graphics of efficiency rates on four tested images

The next testing was to compare the execution time using the proposed distributed rendering system and using single Core i7-2600K PC (quad core) in rendering the four given images. The result of the testing is shown at Table 4.2 and the graphical representation of such comparison is visualized on Fig. 4.3.

Table 4. 2 The data of the comparison result on rendering time from four 3D tested images between single core i7 PC and four nodes distributed rendering

Node	Rendering time			
	Broken Egg	**Castle**	**Plants**	**Glasses**
Single PC core i7	108	1132	993	34660
4 *node distributed rendering*	59	620	633	23118

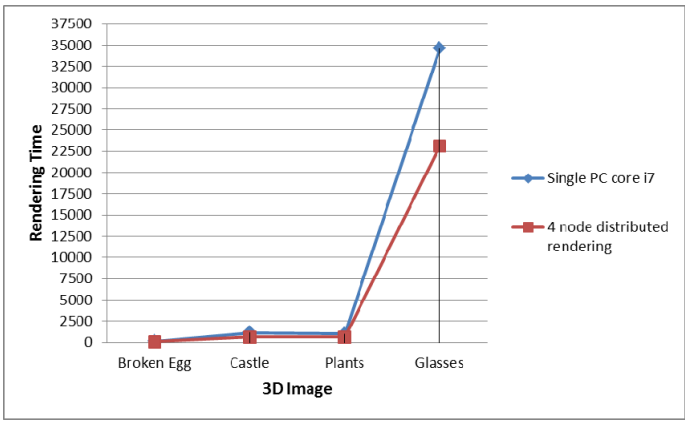

Fig 4. 3 The graphical comparison of 4-node distributed system and a single Core i7 PC

According to the results shown by Fig. 4.1, Fig. 4.2, Fig. 4.3, Table 4.1, and Table 4.3, it is therefore possible to derive following analysis:

1. The speedup increased significantly within one node and two nodes distributed rendering, which were 1.886 and 2.871. However, when the distributed rendering system had five nodes, the speedup increase was not significant, only 3.908 in average, which was quite far from its ideal value, 5-time-fold increase. This was due to the communication from slave nodes to master node that took place for every single running node.
2. After the testing on the cluster system, the average efficiency rate of those four 3D images was as high as 0.924 or about 92%. The average speedup from the four 3D images reached up to 2.68 times. Those results show that the proposed cluster system of distributed rendering is efficient in executing computational processes.
3. There were some cases in which the efficiency rate went beyond 1 (100%). It was because of the non-ideal speedup time or speedup which was in super liner speedup state (S(n)>n), in which the produced speedup time was greater than the number of processors. Such phenomenon might also happen as the result of the existence of the large cache space on hardware in physical cloud infrastructure, in one node within the cluster system.
4. In general, the speedup would not increase in linear to the number of processors. It would, however, tend to decrease up to a certain particular saturated point.
5. The non-ideal speedup time and efficiency rate occurred as the result of the existence of overhead in the parallel system. This phenomenon occurs in all types of parallel system. The tendency of the decreasing speedup and efficiency rate follows the Amdahl Law.
6. Overhead was a part of communication cost, which simply means the time period required to send and to receive the data.
7. As the number of processors grew (larger number of nodes), then communication cost increased. This was the cause of the efficiency drop for introduction of any additional nodes to the system.
8. The different execution time for each tested image was

due to some other factors that influence the results of rendering processes, such as shading, texture shading, contour mapping, blurring, shadowing, smooth shadowing, reflection, and transparency.

9. According to the comparison of the testing result, the execution time of single core i7 2600K PC was faster than that of four nodes distributed rendering system. The reason for such phenomenon was the nonexistence of frame passing process of the rendering result from one computer node to the other computer in single Core i7 2600K PC, which played important role in data communication.
10. However, the distributed cluster system has disadvantage of expensive communication cost (large overhead) because of its network usage as the communication interface.

V. CONCLUSION AND SUGGESTION

A. Conclusion

1. A distributed rendering cluster system is built in a virtualization provided by Infrastructure as a Service (IaaS) cloud computing services where such services can provide computing infrastructure in the form of storage media, processors, memories, and operating systems according to desired needs.
2. The advantage of a distributed cluster system for rendering purpose is its ability to run simultaneously in parallel, in which the computational process is split or distributed into several nodes in order to gain faster computational process. It is also very good at retrieving back the computational results from cluster nodes in order to visualize the results.
3. The performance of distributed rendering cluster system built is quite efficient in the process of computing. This is indicated by the efficiency rate reached up to 92% and its speedup value is 2.68 times more in the fourth test of 3D images.

B. Suggestion

1. It is suggested that for further research there should be analysis on data communication, particularly the overhead from the cluster system for rendering requirement.

REFERENCES

[1] Amdahl, G.M, “Validity of the Single Processor Approach to Achieving Large-Scale Computing Capability,” Proc. AFIPS Conference, pp 483-485, 1967.

[2] Barney, B, “Introduction to Parallel Computing,” Jan 2009. [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/. [Accessed Sept 20, 2011].

[3] Eager, D.L, Zoharjan and Lazowska, E.D, “Speedup versus Efficiency in Parallel Systems,” IEEE Transactions On Computers, vol. 38, no. 3, March 1989.

[4] Flynn, M, "Some Computer Organizations and Their Effectiveness". IEEE Trans. Comput. C-21: 948, 1972.

[5] Fatullah, F., Mutiara, A. B., Yulianto, C, “Analisa Kinerja *Cluster* Linux Dengan Pustaka MPICH Terhadap Perkalian Matriks: Proceedings, komputer dan Sistem Intelijen (KOMMIT2004) Auditorium Universitas Gunadarma, Jakarta, 2004.

[6] Gustafson, J.L, “Reevaluating Amdahl’s Law,” Comm. ACM, 31 (5): 532-533, 1988.

[7] Gonz´alez-Morcillo, C; Weiss, G; Vallejo, D; Jim´enez1y, L; Fdez-Sorribes, J A, “3D *Distributed Rendering* and Optimization using Free Software,” Austria: Software Competence Center,2010.

[8] Graham, R.L., Shipman, G.M., Barrett, B., Castain, R.H., Bosilca, G., and Lumsdaine, A, “Open MPI: A High-Performance, Heterogeneous MPI,” In Proceedings of CLUSTER, 2006.

[9] Gozali, F. Lagusto, D, “Analisis unjuk kerja komputasi *Distributed* shared memory pada sistem *cluster* komputer personal,” JETri, vol 4, no 2, pp 25-44, ISSN 1412-0372, Feb 2005.

[10] Kurniawan, Adi, Pengujian sistem *cluster*ing pada redhat linux 9 dan openmosix menggunakan rangkaian seri RLC. Yogyakarta : Andi Publisher, 2007.

[11] Lee, Changhun, “*Distributed* Shared Memory,” doctoral dissertation, Control Information Systems Lab., School of Electrical Engr.and Computer Science, Seoul National University, Seoul, 2002.

[12] Laksono, A. D., Mutiara, A. B., Heruseto, B, “Analisis perbandingan antara *cluster* openmosix dengan MPI terhadap aplikasi *rendering* POV-RAY” Proceedings. Komputer dan Sistem Intelijen (KOMMIT2004) Auditorium Universitas Gunadarma, Jakarta, Aug 2004.

[13] P. Dian, “Analisis kinerja sistem *cluster* pada proses *distributed rendering* menggunakan blender,” M.S thesis. Gadjah Mada Univ., Indonesia, 2011.

[14] Purnoma, Membangun Virtual PC Dengan Virtualbox.Yogyakarta: Andi Publisher, 2010.

[15] Putri, F. M. Dewi, “Perbandingan Kinerja *Cluster* OpenMosix dengan Disk dan Tanpa Disk: Proceedings, Komputer dan Sistem Intelijen (KOMMIT2004) Auditorium Universitas Gunadarma, Jakarta, Aug 2004.

[16] Ratnawati, Mirna, Pembangunan Objek 3D dan Transformasi Terhadapnya Dengan Menggunakan POV-RAY. Fakultas Ilmu Komputer Universitas Gunadarma, Jakarta, 2007

[17] T. Gilles, “Broken egg (POV-Ray),” 20 Feb 2003.[Online]. Available: http://www.oyonale.com/modeles.php?lang=en&page=26. [Accessed Des.20,2011].

[18] T. Gilles, “Castle tower (POV-Ray),” 6 May 2004. [Online]. Available: http://www.oyonale.com/modeles.php?lang=en&page=34. [Accessed Des. 20,2011].

[19] T. Gilles, “Plants and butterfly (POV-Ray),” 6 May 2004. [Online]. Available: http://www.oyonale.com/modeles.php?lang=en&page=34. [Accessed Des. 20,2011].

[20] T. Gilles, “Glasses,pitcher,dice (POV-Ray),” 6 May 2004. [Online]. Available: http://www.oyonale.com/modeles.php?lang=en&page=34. [Accessed Des. 20,2011].

[21] Wilkinson, B and Allen, M, Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computer. New Jersey : Prentice Hall, Inc, 2005.

[22] Winarno, I., Network File System (NFS) + Samba server. Politeknik Elektronika Negeri Surabaya Institut Teknologi Sepuluh November-PENS ITS Surabaya, 2008.

[23] Y. Arvy, “Implementasi parallel processing pada ray tracing engine menggunakan POV-Ray berbasis MPI.” M.S Thesis. Sepuluh Nopember Institute of Tecnology., Indonesia, 2008.