Apply Cluster and Grid Computing on Parallel 3D Rendering

Chao-Tung Yang  Chuan-Lin Lai
Department of Computer Science and Information Engineering
Tunghai University
Taichung, 407 Taiwan, R.O.C.
Tel: +886-4-23590121 Ext. 3279  Fax: +886-4-23591567
E-mail: ctyang@mail.thu.edu.tw

Abstract
A cluster is a collection of independent and cheap machines, used together as a supercomputer to provide a solution. In this paper, a PC cluster consists of one master node and nine diskless slave nodes (10 processors), is proposed and built for parallel rendering purpose. The system architecture and benchmark performances of this cluster are also presented. Internet computing and Grid technologies promise to change the way we tackle complex problems. They will enable large-scale aggregation and sharing of computational, data and other resources across institutional boundaries. And harnessing these new technologies effectively will transform scientific disciplines ranging from high-energy physics to the life sciences. Also, in this paper, we construct two heterogeneous PC clusters for parallel rendering purpose and install Linux Red Hat 9 on each PC cluster. Then, these clusters are set to the different subnet. Therefore, we use the grid middleware - Globus ToolKit, to connect these two clusters to form a grid computing environment on multiple Linux PC clusters. We also install the SUN Grid Engine, to manage and monitor incoming or outgoing computing job and schedule the job to achieve high performance computing and high CPU utilization. The system architecture and benchmark performances of this cluster are also presented.

Keywords: PC cluster, Parallel rendering, Cluster computing, Grid Computing, Speedup.

1. Introduction
Extraordinary technological improvements over the past few years in areas such as microprocessors, memory, buses, networks, and software have made it possible to assemble groups of inexpensive personal computers and/or workstations into a cost effective system that functions in concert and possesses tremendous processing power. Cluster computing is not new, but in company with other technical capabilities, particularly in the area of networking, this class of machines is becoming a high-performance platform for parallel and distributed applications [4, 5, 9, 10, 11, 12, 13, 14].

Grid computing, most simply stated, is distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. The standardization of communications between heterogeneous systems created the Internet explosion. The emerging standardization for sharing resources, along with the availability of higher bandwidth, are driving a possibly equally large evolutionary step in grid computing [1, 2, 3, 4, 5, 6, 14].

Rendering is a technique for generating a graphical image from a mathematical model of a two or three-dimensional object or scene. A common method of rendering is ray tracing. Ray tracing is a technique used in computer graphics to create realistic images by calculating the paths taken by rays of light entering the observer’s eye at different angles. Ray tracing is an ideal application for parallel processing since there are many pixels, each of whose values are independent and can be calculated in parallel. The Persistence of Vision Ray Tracer (POV-Ray) is an all-round 3-dimensional ray tracing software package [7, 8]. It takes input information and simulates the way light interacts with the objects defined to create 3D pictures and animations. In addition to the ray tracing process, newer versions of POV can also use a variant of the process known as radiosity (sophisticated lighting) to add greater realism to scenes, particularly those that use diffuse light POVRay can simulate many atmospheric and volumetric effects (such as smoke and haze).

In this paper, a cluster with diskless clients was built. The system architecture and benchmark performances of the cluster are presented. In order to measure the performance of our cluster, the parallel ray-tracing problem is illustrated and the experimental result is also demonstrated on our Linux PC cluster. The experimental results show that the highest speedup is 6.55 for PVMPOV, when the total numbers of processor are 10 on cluster. Then, we use MPIPOV parallel ray-tracing techniques to examine the Grid Computing system. The experimental results show that the highest speedups are
obtained for MPI-POV, when the total number of processors is 8, by creating 8 tasks on Grid system. The results of this study will make theoretical and technical contributions to the design of a message passing program on Grid Computing system.

2. Background

2.1. Cluster Computing

A Beowulf cluster uses multi-computer architecture, as depicted in Figure 1. It features a parallel computing system that usually consists of one or more master nodes and one or more compute nodes, or cluster nodes, interconnected via widely available network interconnects. All of the nodes in a typical Beowulf cluster are commodity systems—PCs, workstations, or servers—running commodity software such as Linux.

From a user’s perspective, a Beowulf cluster appears as a Massively Parallel Processor (MPP) system. The most common methods of using the system are to access the master node either directly or through Telnet or remote login from personal workstations. Once on the master node, users can prepare and compile their parallel applications, and also spawn jobs on a desired number of compute nodes in the cluster. Applications must be written in parallel style and use the message-passing programming model. Jobs of a parallel application are spawned on compute nodes, which work collaboratively until finishing the application. During the execution, compute nodes use standard message-passing middleware, such as Message Passing Interface (MPI) and Parallel Virtual Machine (PVM), to exchange information [4, 5, 6, 9, 10, 11, 12, 13].

![Figure 1: Logic view of a Beowulf cluster.](image)

2.2. Grid Computing

Grid computing requires the use of software that can divide and farm out pieces of a program to as many as several thousand computers. Grid computing can be thought of as distributed and large-scale cluster computing and as a form of network-distributed parallel processing. It can be confined to the network of computer workstations within a corporation or it can be a public collaboration. The establishment, management, and exploitation of dynamic, cross-organizational sharing relationships require new technology. This technology is Grid architecture and supporting software protocols and middleware [1, 2, 4, 5, 14].

Another key technology in the development of grid networks is the set of middleware applications that allows resources to communicate across organizations using a wide variety of hardware and operating systems. The Globus Toolkit [1, 2] is a set of tools useful for building a grid. Its strength is a good security model, with a provision for hierarchically collecting data about the grid, as well as the basic facilities for implementing a simple, yet world-spanning grid.

2.2.1. Globus Toolkit

The Globus Project [2] provides software tools that make it easier to build computational grids and grid-based applications. These tools are collectively called The Globus Toolkit. The Globus Toolkit is used by many organizations to build computational grids that can support their applications. Globus has a layered architecture (see Figure 2) that includes the fabric, connectivity, resource, collective and application layers.

![Figure 2: The Globus Grid architecture, where protocols, services and APIs occur at each level](image)

2.2.2. MPICH-G2

MPICH-G2 [6] is a grid-enabled implementation of the MPI v1.1 standard. That is, using services from the Globus Toolkit® (e.g., job startup, security); MPICH-G2 allows you to couple multiple machines, potentially of different architectures, to run MPI applications. MPICH-G2 automatically converts data in messages sent between machines of different architectures and supports multi-protocol communication by automatically selecting TCP for inter-machine messaging and (where available) vendor-supplied MPI for intra-machine messaging. Existing parallel programs written for MPI can be executed over the Globus infrastructure just after recompilation [6].

2.2.3. SUN Grid Engine

Sun Grid Engine is new generation distributed resource management software which dynamically matches users’ hardware and software requirements to
the available (heterogeneous) resources in the network, according to policies usually defined by management [3]. Sun Grid Engine acts as the central nervous system of a cluster of networked computers. Via so-called daemons, the Grid Engine Master supervises all resources in the network to allow full control and achieve optimum utilization of the resources available.

Sun Grid Engine aggregates the compute power available in dedicated compute farms, networked servers and desktop workstations, and presents a single access point to users needing compute cycles. This is accomplished by distributing computational workload to available systems, simultaneously increasing the productivity of machines and application licenses while maximizing the number of jobs that can be completed.

3. Performance Evaluation

We have run PVMOV [7, 8] on our 10-processors testbed and have had amazing results, respectively. With the cluster configured, runs the following commands to begin the ray tracing and generates the image files as shown in Figure 2.

$ ./pvmpov +iskyvase.pov +w800 +h600 +nt10 -L/home/ct/pvmpov3_1g_2/povray31/include
$ ./pvmpov +ipawns.pov +w800 +h600 +nt10 -L/home/ct/pvmpov3_1g_2/povray31/include
$ ./pvmpov +ifish13.pov +w800 +h600 +nt10 -L/home/ct/pvmpov3_1g_2/povray31/include

This is the benchmark option command-line with the exception of the +nw and +nh switches, which are specific to PVMOV and define the size of image each of the slaves, will be working on. The +nt switch is specific to the number of tasks will be running. For example, +nt10 will start 10 tasks, one for each processor.

The messages on the screen should show that slaves were successfully started. When completed, PVMOV will display the slave statistics as well as the total render time. In case of Skyvase model, by using single processor mode for processing 1600×1280 image, the render time was 88 seconds. Using out the diskless cluster (10 processors) further reduced the time to 17 seconds. The execution times for the different Povray model (Skyvase, Pawns, and Fish13) on the cluster were shown in Figure 3, respectively. The corresponding speedups of different problem size by varying the number of task (option: +nt) was shown in Figure 5.

In case of Skyvase model, by using single processor mode for processing 1600×1280 image, the render time was 4652 seconds. Using out the Grid Computing sys-
tem (8 processors) further reduced the time to 647 seconds. The execution times for the different POVray model (Skyvase, Pawns, and Fish13) on the Grid were shown in Figures 7, 8, and 9, respectively. The highest speedup was obtained about (1600×1280) for Chesss2 model by using our Grid Computing system with 8 processors, respectively.

5. Conclusions
In this paper, a cluster with diskless clients was built. The system architecture and benchmark performances of the cluster are presented. In order to measure the performance of our cluster, the parallel ray-tracing problem is illustrated and the experimental result is also demonstrated on our Linux PC cluster. The experimental results show that the highest speedup is 6.55 for PVMPOV, when the total numbers of processor are 10 on cluster. Then, we use MPIPOV parallel ray-tracing techniques to examine the Grid Computing system. The experimental results show that the highest speedups are obtained for MPIPOV, when the total number of processors is 8, by creating 8 tasks on Grid system. The results of this study will make theoretical and technical contributions to the design of a message passing program on Grid Computing system.

References