LINUX CLUSTER POSSIBILITIES IN 3-D PHOTO QUALITY IMAGING AND ANIMATION

Arjun Jain

5th Semester CSE

Room No. 231

Chamundi Block

R.V. College of Engineering

Bangalore-59

9886260870

arjun@rvce.ac.in

November 15, 2003

1 ABSTRACT

We present the PC cluster - Blob built in the Department of Computer Science and Engineering, Bangalore. The structure of the cluster is described and the performance is evaluated by rendering benchmark Persistence of Vision 3-D images.

1.1 BACKGROUND

With the increasing needs of market there is a high demand for High-end computing systems. One of the major drawbacks of these High-performance systems is their enormous cost. Moreover such systems are not scalable and therefore not suitable for a growing industry with ever growing computing needs. The requirement is a Lowcost, High-Performance system which is Scalable. In such a scenario, Cluster technology aptly fits in. A Linux cluster is a high performance low cost parallel computing engine capable of delivering on big scientific and engineering problems. It is basically Loosely Coupled network of Linux servers functioning as a single parallel machine. The basic philosophy being able to harness the computational power of many as such low performing machines when used together. Clusters have been found to deliver the performance of a Conventional supercomputer at low cost.

1.2 TECHNOLOGY

GNU/Linux (`http://www.gnu.org`) is used as the operating system for the cluster. PVM 3.1 (`http://www.netlib.org/`) was used as the Virtual Machine. Since Paralell Virtual Machine has an extensive library that is compatible with ANSI C and C++ for message passing with slave timeout feature, PVM was used here. POV-RAY 3.5c (`http://www.povray.org`)

was used as the Image Rendering software to render photo realistic 3-d images. The load balancing was done by a custom made patch developed-PVM-Pov which renders images on the cluster. PVM-Pov divides the image to be rendered into blocks, spawns the slaves and the rendered images are send back to the master where they are integrated generate the final image.

1.3 DESIGN

Ray-Tracing is perhaps one of the most processor consuming tasks for the industry today. Huge Vectors and MPPs are employed to render photo-quality images. Also with increasing hardware comes increase in price. Rendering of complex images on a cluster is a unique concept wherein the image is divided into number of sub images (by the custom made patch) as objects and layers. Each of these layers in turn are then spawned to the slaves by the master. The slaves then process their load and their rendered share is sent back to the master. The master then integrates all the layers and objects received from the slaves and thus the final image is rendered.

1.4 RESULTS

The results obtained by rendering images on the cluster have been more than satisfactory. The statistics generated (described in detailed later) have been very encouraging as described for one of our test benchmarks benchmark-3-5c.pov are as follows: a single node workstation rendered

the image at 1024x768 resolution in 59 minutes and 23 seconds. The same image rendered on our Cluster with 7 nodes took 10 minutes and 2 seconds which implies an efficiency of 81.12%. Thus cluster developed is an industry standard Beowulf Class Linux cluster built using off the shelf regular PCs.

1.5 CONCLUSION

Each 10 second motion picture quality animation consists of at least 800 frames. One frame of the POV-Ray benchmark benchmark.pov at a resolution of 1024 x 768 took 59.23 mins on a high end system AMD 1800+ with 128 MB DDR Ram . Thus a similar 10 second animation will take only 32 days 18 hours and 40 minutes approximately. Thus the only possible alternatives in this field of high quality image and graphics rendering is either an inexpensive and scalable Cluster or an expensive Vector.

2 INTRODUCTION

Computational image rendering is widely used for generating images and graphics for the motion pictures industry and bio-modelling. From the mathematical point Ray-tracing is a rendering technique that calculates an image of a scene by shooting rays into the scene. The scene is built from shapes, light sources, a camera, materials, special features, etc. For every pixel in the final image one or more viewing rays are shot into the scene and tested for intersection with any of the objects in the scene.

Viewing rays originate from the viewer, represented by the camera, and pass through the viewing window (representing the final image).Every time an object is hit, the color of the surface at that point is calculated. For this purpose the amount of light coming from any light source in the scene is determined to tell whether the surface point lies in shadow or not. If the surface is reflective or translucent new rays are set up and traced in order to determine the contribution of the reflected and refracted light to the final surface color, but in order to achieve results comparable with those generated by conventional techniques experimental we need supercomputers power or parallel computers cluster. In this way such tasks are solved during the last decade of 20th century. Clusters based on PCs running Linux have become the cheapest supercomputers in the academic and commercial field. We created such a cluster in The Department of Computer Science, R.V. College of Engineering (CSE, RVCE). The clusters performance have been tested by generating many such test images.

3 PERSONAL COMPUTER CLUSTER

With the power and low prices of today's PCs and the availability of 100 Mb/s Ethernet interconnect, it makes possible to combine them to build High- Performance-Computing and Parallel Computing environment. Today, there is a wide range of switches available, ranging from 8 to over 100 ports, some with one or more Gigabit modules that lets you build large systems by interconnecting such switches or by using them

with a Gigabit switch. Switches have become inexpensive enough, so there is not much reason to build your network by using cheap hubs or by connecting the nodes directly in a hypercube network. The local area PC cluster was made in The Department of Computer Science, RVCE for image processing and rendering. PC cluster of 7 nodes: all of them dual AMD 1.8 G Hz processors; all have 128 MB RAM, a 40 GB disk drives. Machines have been installed with the standard RedHat 9.0 and WinXP(parallel cluster runs in Linux OS at this time). All PCs are assumed to boot from a their own hard drive and have a fast Ethernet network (100 Mb/s) connection to a switch controlling the private cluster network. The suggested range of addresses for a private network is from 192.168.100.1 to 192.168.100.20. Nobody is able to connect directly to a compute node from outside this network anyway. This keeps normal traffic from interfering with inter-node communication and vice versa.

Thus the total memory 896 MB; total disk space 280 GB. We use cus-

tom made PVM-Pov Cluster software - a high-performance parallel image rendering environment for workstation and PC clusters [2]. PVM-Pov uses the PVM 3.1 high performance communication library a dedicated communication library for cluster computing (it allows using many types of networks). Communication library used is faster than usual TCP protocol. In addition a trunking is possible (installation and use of up to 4 NICs). Under the PVM virtual machines users are not aware whether or not a system is a cluster of single/multi-processor computers or a cluster of clusters. PVM system has check point function. It allows restart function and improves a reliability of the system. Available compilers are gcc and g++ compilers C, C++ and HPF. The cluster differs from the network of workstations in security,application software, administration, booting and file systems. Application software uses underlying message passing system like Parallel Virtual Machine (PVM). There are many ways to express parallelism, but message passing is the more effective and more modern.

For administering the cluster we have login node with a keyboard, monitor and mouse. Other nodes can be headless (no keyboard, mouse, or monitor) but in our way there are all machines like login node, because our cluster is made on the basis all equal class. Any machine can be logged into from the login node using secure shell(ssh) and can be administered.

4 DESIGN

Image generation using Ray-tracing is a is a rendering technique that calculates an image of a scene by shooting rays into the scene. The scene is built from shapes, light sources, a camera, materials, special features, etc.

For every pixel in the final image one or more viewing rays are shot into the scene and tested for intersection with any of the objects in the scene. Viewing rays originate from the viewer, represented by the camera, and pass through the viewing window (representing the final image).

Every time an object is hit, the color of the surface at that point is calculated. For this purpose the amount of light coming from any light source in the scene is determined to tell whether the surface point lies in shadow or not. If the surface is reflective or translucent new rays are set up and traced in order to determine the contribution of the reflected and refracted light to the final surface color.

Now this is a very 'heavy' and time-taking job for the processor. For every pixel atleast three to four rays are shot and tested for intersection. Each ray has its equation and each equation has to be solved for intersection. Each pixel needs to be rendered and simple ray tracing can be done for a uniprocessor system using the following algorithm.

```
Select center of projection and window on viewplane
for(each scan line in image){
        for(each pixel in scanline){
                determine ray from center of projection through pixel;
```

```
                for(each object in the scene){
                        if(object is intersected and is closest considered thus
                                record intersection and object name;
                }
                set pixel's color to that at closest object intersection;
        }
}
```

Now, instead of rendering the entire image on the same system we here use a clustered system in the following way: the image is broken up into still smaller blocks. Each slave is to then render one or more of these blocks independently and send the rendered image back to the master where all the blocks are integrated to form the final image. This can be explained using the following algorithm.

This is an algorithm to add 75 integers on a clustered system with 3 slave nodes and one master node.

```
/* Algorithm for the master program */
initialize the array 'items'.

/* send data to the slaves */
for i = 0 to 3
    Send items[25*i] to items[25*(i+1)-1] to slave Pi
end for

/* collect the results from the slaves */
```

```
for i = 0 to 3
    Receive the result from slave Pi in result[i]
end for

/* calculate the final result */
sum = 0
for i = 0 to 3
    sum = sum + result[i]
end for

print sum
```

The algorithm for the slave can be written as follows.

```
/* Algorithm for the slave program */

Receive 25 elements from the master in some array say 'items'

/* calculate intermediate result */
sum = 0
for i = 0 to 24
    sum = sum + items[i]
end for
send 'sum' as the intermediate result to the master
```

Here each slave is given the task of adding 1/3 of the total no of integers and the final result of these 25 addition is sent to the master

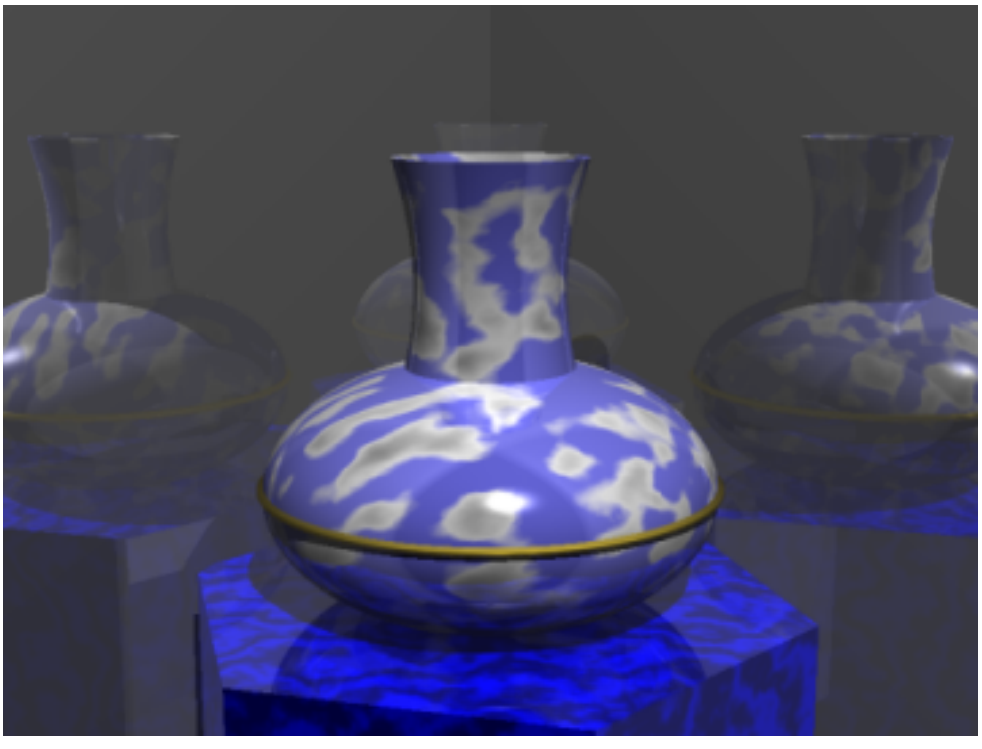
where it adds the 3 results from the slaves to get the final result of the addition of 75 integers. Thus the job is done in paralell.

5 RESULTS

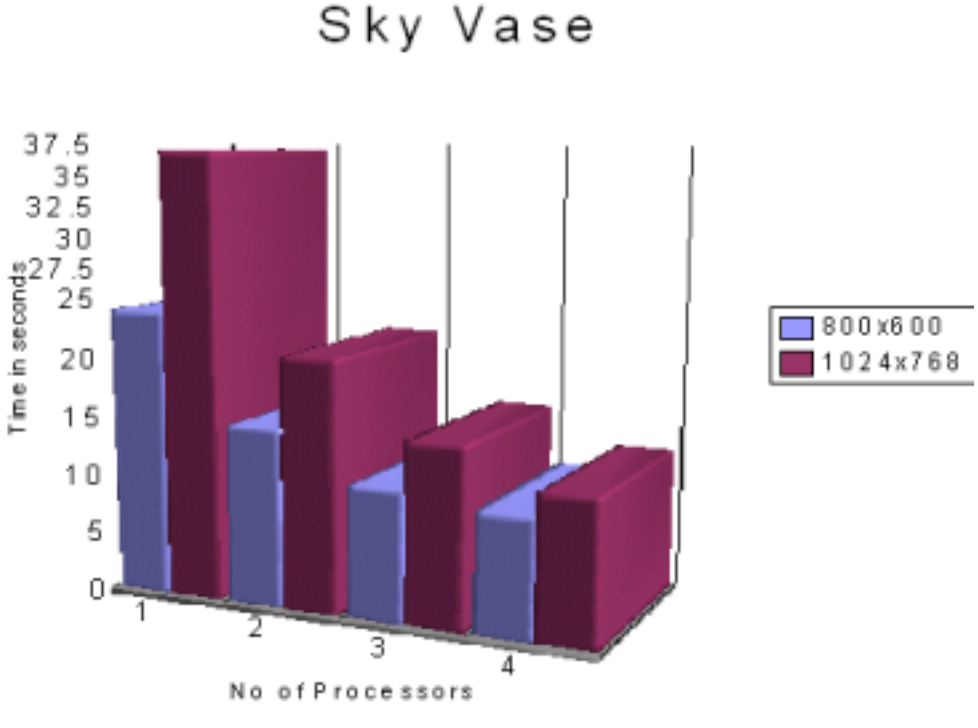
The algorithms described and developed were used to generate many benchmark 3-D images such as the famous skyvase.pov, chess.pov and blob.pov. All the experiments under consideration here were carried on clusters with 4, 3, 2, and single processor systems.

5.1 SKYVASE.POV

The benchmark SKYVASE rendered at 1024x768 resolution is shown bellow:



This image takes approximately 14 seconds in a quard processor system and 17 in the tri processor system.



In a uniprocessor system the same takes 38 seconds and in a bi processor system 20 seconds thus with a degradation of only 2% to 14%.

5.2 CHESS.POV

This is a more complex nad *heavier* image than the Skyvase. The benchmark CHESS 2c rendered at 10024x768 resolution is shown bellow:

This image takes approximately 150 seconds in a quard processor system and 210 in the tri processor system.

In a uniprocessor system the same takes 580 seconds and in a bi processor system 350 seconds thus with a degradation of only 12% to 14%.

6 REFRENCES

1.PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing, Al Geist, Adam Beguelin, Jack Dongarra, Robert Manchek, Weicheng Jiang and Vaidy Sunderam, MIT Press. Available at http://www.netlib.org/

2.MPI: The Complete Reference, Marc Snir, Steve Otto, Steven Huss-Lederman, David Waker and Jack Dongarra, MIT Press. Available at http://www.netlib.org/.

3.RS/6000 SP: Practical MPI Programming,Yukiya Aoyama and Jan Nakano, International Techical Support Organization, IBM Corporation,

http://www.redbooks.ibm.com/.

4.A Beginner's Guide to PVM Parallel Virtual Machine, Clay Breshears and Asim YarKhan, Joint Institute of Computational Science, University of Tennessee, USA. www-jics.cs.utk.edu/PVM/pvm/_guide.html.

5.PVM: An Introduction to Parallel Virtual Machine, Emily Angerer Crawford, Office of Information Technology, High Performance Computing, www.hpc.gatech.edu/seminar/pvm.html.

6.www.povray.org - POVRAY sources for UNIX

7.PVM home http://www.epm.ornl.gov/pvm/pvm_home.html

8. PVMPOV from sunsite.unc.edu