Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

Simulations of Pinhole Imaging for AXAF: Distributed Processing Using the MPI Standard

D. Nguyen and B. Hillberg

Smithsonian Astrophysical Observatory, 60 Garden St., Cambridge, MA 02138

Abstract. The pinhole simulation program is a computationally and memory intensive task. A conventional sequential approach would limit the size and complexity of such a problem when investigated in the framework of the SAO *AXAF* simulation system. A parallel version was developed instead, to enable distributed processing on a cluster of workstations. The program makes use of the Message Passing Interface (MPI) standard for parallel processing, implemented as an API (Application Programming Interface) to the Local Area Multicomputer (LAM) programming environment developed at the Ohio Supercomputer Center.

1. Introduction

As part of our efforts to support the *AXAF* program, the SAO *AXAF* Mission Support Team has developed a software suite to simulate *AXAF* images generated by the flight mirror assembly (Jerius 1994). One of the tasks of this system is to simulate pinhole imaging of the X-ray source.

2. Pinhole Simulation in Sequential Mode

The task of the pinhole program is to tabulate the weight of the photons detected through a pinhole. The weight of a photon represents the probability of finding the photon at a given position. Photons are generated with weights equal to 1 at the source. Weights are reduced at every reflection point along their paths toward the detector. The condition for a photon to successfully pass through a pinhole is:

$$(x - x_o)^2 + (y - y_o)^2 < r_o^2 \tag{1}$$

where (x,y) is the photon position in the plane of the pinhole, relative to the pinhole of radius r_o centered at (x_o,y_o). In order to simulate a two dimensional scan at the focal plane for pinholes of various radii, the pinholes are laid out on a cubic lattice. The pinholes on a rectangular grid simulate a two dimensional scan, the stack of rectangular grids represents the pinholes of various radii.

A naive approach in writing the pinhole program would require

$$O(N * X * Y * Z) \tag{2}$$

operations to finish. N is the number of incident photons, X and Y are the number of grid sites along the x and y axes of the rectangular grid, and Z is the number of radii to be calculated. An efficient program should minimize the number of times equation (1) has to be executed. The layout of the pinholes on a cubic lattice is used for this purpose, since in this case the photon stream only needs to be read once. Further reduction in execution time can be achieved by realizing that the pinholes of different radii are concentric. In the current implementation, the program requires

$$O(N * Overlap^2 * log_2(Z) * (log_2(X) + log_2(Y))) \tag{3}$$

operations. N, X, Y, and Z are as defined above, and *Overlap* is the number of overlapping pinholes in x and y directions. The drawback of the cubic lattice layout approach is that the three dimensional array that holds the photon weights can be prohibitively large for some class of problems; e.g., a 4000 by 4000 grid size would require 128 MB of memory.

Verification of the software was done as follows: a spatially uniform distribution of photons was generated with the weight of each photon set to unity. The photons were traced to the plane of the pinholes. The number of photons which make it through a pinhole must be equal to the density of the incident photon beam times the area of the pinhole.

3. Parallel Processing

These efficiency improvements achieve optimum speed for the pinhole simulation program running on a given machine. This, however, is still not satisfactory given the large volume of data that needs to be simulated. The next step was to consider parallel processing across workstations on a local area network (LAN).

Distributed multicomputing on a network-connected workstations provides a cost-effective environment for high performance scientific computing. Software packages exist that support parallel processing on workstation clusters by managing the communications and data traffic in a way transparent to the application. These packages provide sets of Application Programming Interfaces (API) for various languages so that their functions can be called from an application. Examples of such packages are Express, PARMACS, PVM, and P4. Although these packages have very similar functionalities, their APIs are very different. The result of non-standard APIs is that third party software becomes specific to a given package, and cannot be used with other packages.

4. The MPI Standard

A standardization process for a message passing system was initiated at the "Workshop on Standards for Message Passing in a Distributed Memory Environment," held in Williamsburg, Virginia in 1992 November. The Message Passing Interface (MPI) forum consists of researchers from government laboratories, universities, and industry, along with vendors of concurrent computers. The MPI standard was derived from the best features of its predecessors, rather than adopting one of the existing systems. The MPI standard includes: Point-to-point communication, Collective operations, Process groups, Communication context, Process Topologies, Bindings for FORTRAN 77 and C, Environment Management, and Inquiry and Profiling Interfaces.

The pinhole program makes use of the MPI standard for parallel processing implemented as an API to the Local Area Multicomputer (LAM) programming environment developed at the Ohio Supercomputer Center (Burns 1989). LAM is a distributed memory MIMD programming and operating environment for heterogeneous UNIX computers on a network.

5. Pinhole Simulation in Parallel Mode

The pinhole program can be completely parallelized by using data decomposition. More than one instance of the program can run, each on a different machine, analyzing a different subset of the data. One process (the master) initiates all the other processes (the slaves), and, at the end, collects all the results. In the master-slave computing paradigm, each slave communicates with the master, but there is no inter-slave communication.

The expected speedup from the sequential process for p number of slaves is (Almasi & Gottlieb, 1994):

$$speedup = \frac{p * r_1}{(r_1 + p^2)} \tag{4}$$

$$r_1 = \frac{T_{sequential}}{T_{communication1}} \tag{5}$$

where $T_{sequential}$ is the time needed on one machine and $T_{communication1}$ is the time needed to communicate with one slave. For a given r_1 the maximum speedup occurs at $p = \sqrt{r_1}$. The maximum speedup is therefore $0.5\sqrt{r_1}$.

The pinhole program was modified and executed on a system of one master and ten slave processes, each running on a different workstation. The results were consistent with equations (4) and (5), i.e., r_1 was measured to be about 100, so for ten slaves a factor of five in speed was gained.

6. Summary

MPI enables the writing of portable high-performance libraries for distributed-memory machines, easing the burden for application programmers.

References

Almasi, G. & Gottlieb, A. 1994, Highly Parallel Computing, 2nd ed., (Redwood City, Benjamin/Cummings)

Burns, G. 1989, in Proceedings of the Fourth Conference on HyperCubes, Concurrent Computers, and Applications (Los Altos, Golden Gate Enterprises)

Jerius, D., Freeman, M., Gaetz, T., Hughes, J. P., & Podgorski, W. 1995, this volume, p. **??**

Message Passing Interface Forum 1995, MPI: A message passing Interface Standard, Technical Report (Knoxville, University of Tennessee), in press