An Introduction to Parallel Cluster Computing using PVM for Computer Modeling and Simulation of Engineering Problems

Presented by Valerie Spencer
Mentored by Jim Kohl

Oak Ridge National Laboratory RAM Internship
June 11 – August 17, 2001

Contents **Page**

List of Figures **Page**

Acknowledgements

I appreciate Dr. Z. T. Deng’s decision of selecting me to be a part of the Summer 2001 RAM Program. I appreciate Dr. Jim Kohl for guiding my research, dedicating time to discuss my findings, and facilitating resources from the Computer Science and Mathematics Division. This research was performed under the Research Alliance for Minorities Program administered through the Computer Science and Mathematics Division, Oak Ridge National Laboratory. This Program is sponsored by the Mathematical, Information, and Computational Sciences Division; Office of Advanced Scientific Computing Research; U.S. Department of Energy. I would also like to acknowledge Mrs. Debbie McCoy for organizing the RAM Program that allowed me to spend the summer gaining significant information at ORNL. Other acknowledgements go out to Debbie Flanagan, Stephen Scott, and Cheryl Hamby. This work has been authored by a contractor of the U.S. Government at Oak Ridge National Laboratory managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

Abstract

An investigation has been conducted regarding the ability of clustered personal computers to improve the performance of executing software simulations for solving engineering problems. The power and utility of personal computers continues to grow exponentially through advances in computing capabilities such as newer microprocessors, advances in microchip technologies, electronic packaging, and cost effective gigabyte-size hard drive capacity. Many engineering problems require significant computing power. Therefore, the computation has to be done by high-performance computer systems that cost millions of dollars and need gigabytes of memory to complete the task. Alternately, it is feasible to provide adequate computing in the form of clustered personal computers. This method cuts the cost and size by linking (clustering) personal computers together across a network. Clusters also have the advantage that they can be used as stand-alone computers when they are not operating as a parallel computer. Parallel computing software to exploit clusters is available for computer operating systems like Unix, Windows NT, or Linux. This project concentrates on the use of Windows NT, and the Parallel Virtual Machine (PVM) system to solve an engineering dynamics problem in Fortran.

Introduction

There are two parts to the PVM system. The *daemon* is a special purpose process that runs on behalf of the system to handle all the incoming and outgoing message communication. It is represented by "pvmd3" or "pvmd" and any user with a valid login id can install and execute this on a machine. The other part of the system is a library of routines that enables parallel tasks on the computers to interact.

In order to use PVM in the computation of a problem, the problem must be able to be broken down into several tasks. This is known as parallelism, which is done in two ways. One way is “functional” and is accomplished by breaking the application into different tasks that perform different functions. The other way is “data” and this is done by having several similar tasks each solve one part of the data. PVM is a system that can be used for one of these methods or a combination of both.

C, C++, and Fortran are all languages that can be used to write PVM codes. This project is done using the Fortran language. Fortran language binders are carried out as subroutines instead of functions. This is because some compilers cannot combine Fortran functions with C functions. Therefore, Fortran applications (programs) have to be linked to the PVM Fortran library, the standard PVM library and the C socket library using these links libfpvm3.lib, libpvm3.lib, and wsock32.lib.

PVM codes are written following two main programming models. Using the master/worker model, the master task creates all other tasks that are designed to work on the problem, and then coordinates the sending of initial data to each task, and collects results from each task. However, in the hostless model the initial task spawns off copies of itself as tasks and then starts working on its portion of the problem while the created tasks immediately begin working on their portion. Input and output requirements are often handled by individual tasks, but it may be beneficial to have some results collected by a single task, especially if those results need to be broadcast back out to all the tasks.

In order to execute a program under PVM, the user adds calls to the PVM library routines that spawn off tasks to the other machines within the user's virtual machine and allow tasks to send and receive data.

Method

The problem to solve was a 3-D spring/mass system. Utilizing the ideas and concepts learned in my dynamics course, we applied the equations for spring force, velocity, and displacement such as:

$\text{Force} = K_{spring}(DX)$

$X = X_o + V_o\, DT + [a(DT)^2]/2$ where a = Force/mass

Therefore, we came up with the equations:

(1) Ftmp = (E0 - E) * Springs(I, J)

and

(2a) Fx(I) = Fx(I) + Ftmp * (X(I) - X(J)) / E
(2b) Fy(I) = Fy(I) + Ftmp * (Y(I) - Y(J)) / E
(2c) Fz(I) = Fz(I) + Ftmp * (Z(I) - Z(J)) / E

(2d) Fx(J) = Fx(J) + Ftmp * (X(J) - X(I)) / E
(2e) Fy(J) = Fy(J) + Ftmp * (Y(J) - Y(I)) / E
(2f) Fz(J) = Fz(J) + Ftmp * (Z(J) - Z(I)) / E

and

(3a) X(I) = X(I) + (Vx(I) * DT) + ((Fx(I) * DT**2)/(2 * Mass))
(3b) Y(I) = Y(I) + (Vy(I) * DT) + ((Fy(I) * DT**2)/(2 * Mass))
(3c) Z(I) = Z(I) + (Vz(I) * DT) + ((Fz(I) * DT**2)/(2 * Mass))

and

(4a) Vx(I) = Vx(I) + (Fx(I) * DT / Mass)
(4b) Vy(I) = Vy(I) + (Fy(I) * DT / Mass)
(4c) Vz(I) = Vz(I) + (Fz(I) * DT / Mass)

First, using equation (1) the spring force attached to any two masses is calculated and then is used to calculate the vector forces of mass 1 (2a-c) and mass 2 (2d-f). The forces accumulated in equations (2a-c) are then used to update all mass locations (3a-c) and velocities (4a-c) in vector form as the system vibrates back and forth in space. No boundary conditions are assumed. Before the velocities are updated (4a-c), they are also used in equations (3a-c). DT is the change in time, which is a constant value, as are mass and E0 (original extension of spring). Figure 1 shows a diagram of the program flow.

Results and Discussion

Figure 2 shows the performance of the average time in seconds that it took to run a sequential version of the program versus a parallel version of the program for the given number of masses. Only a single desktop PC was used and the number of computation iterations was set at five hundred. Each parallel run was set up to have one master task and two worker tasks. The results of the parallel program were poor because when a parallel program is run on a single computer, the overhead of running multiple processes parallel will always make the "parallel" code run slower than the sequential code.

Figure 3 shows the results after running the sequential program on a laptop and the desktop. The laptop, which is much slower than the desktop, has a huge effect on the performance.

In order to collect more efficient and correct data, we need to run the parallel code in a more balanced parallel environment. Figure 4 shows the results clustering the PC with the laptop to create our parallel environment. The PC, which obviously has more computing power, is slowed down by the laptop when running in parallel. A proper, balanced parallel cluster should give the desired results that when given a large and complex problem, clustering PCs together will compute much faster than a single computer. For the current configuration, the imbalance in computing power, plus the cost of parallel communication overhead, makes the sequential program faster.

Conclusion

This research has proved that clustering personal computers together can provide adequate computing power for large engineering problems. It has also proved that computers within the cluster can be used as stand alone computers because the desktop and laptop in this cluster served other purposes such as preparing a paper, this Power Point presentation, and a poster presentation. However, there are several optimizations that should be done to this project to improve the parallel performance. Avoiding unnecessary message packing and reducing the message sizes are two approaches that I plan to explore in the future upon my potential return to ORNL next summer.

References

Geist, Al, Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1994), *PVM: Parallel Virtual Machine*, London, Massachusetts: MIT Press.

Nyhoff, L., and Leestma, S. (1988), *Fortran 77 for Engineers and Scientists* (4th ed.), New York: Macmillan.

Resnick, R., and Halliday, David. (1977), *Physics: Part One* (3rd ed.), New York: John Wiley & Sons.

Appendix of Figures

Figure 1:
Program Flow

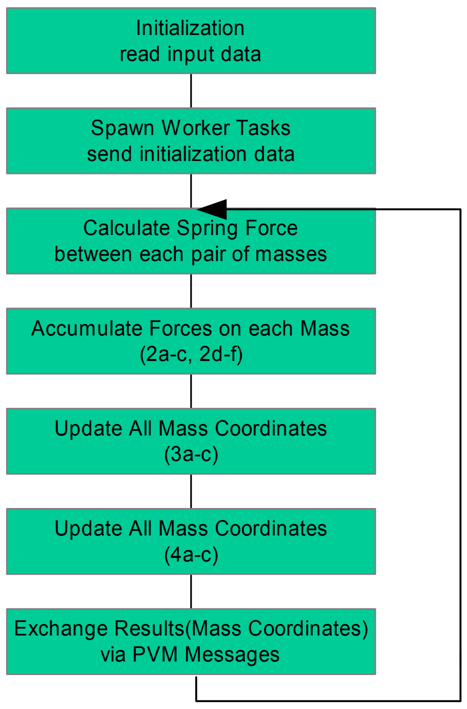

Initialization
read input data
Spawn Worker Tasks
send initialization data
Calculate Spring Force
between each pair of masses
Accumulate Forces on each Mass
(2a-c, 2d-f)
Update All Mass Coordinates
(3a-c)
Update All Mass Coordinates
(4a-c)
Exchange Results(Mass Coordinates)
via PVM Messages

Figure 2:
Performance of Sequential Program vs.
Parallel Program on Desktop Only

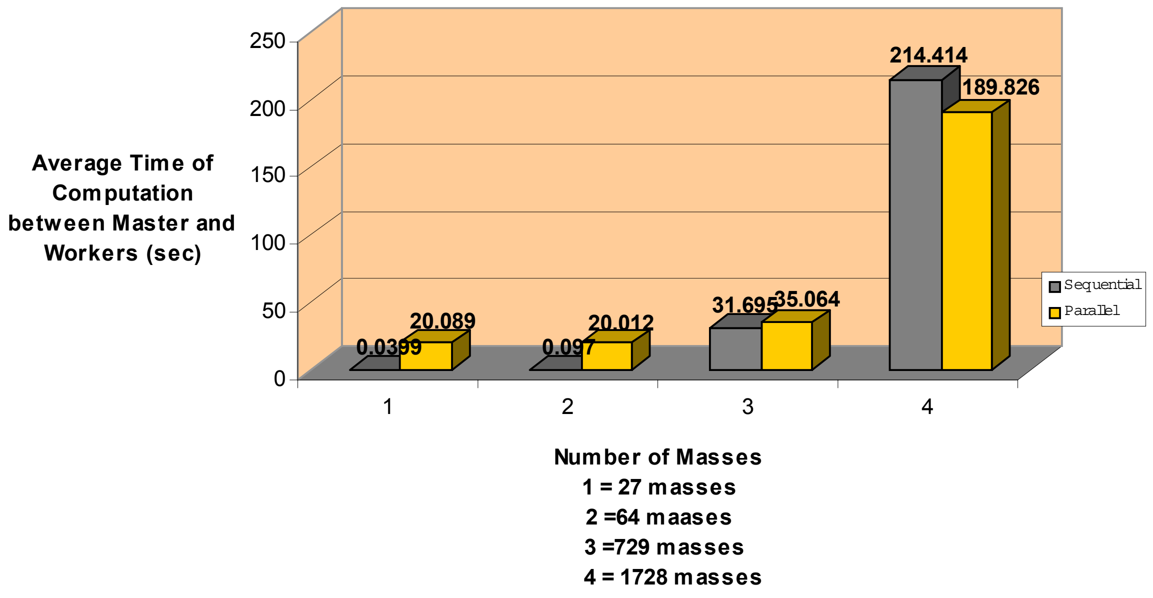


Figure 3:
Performance of Sequential Program on the Laptop vs. the Desktop

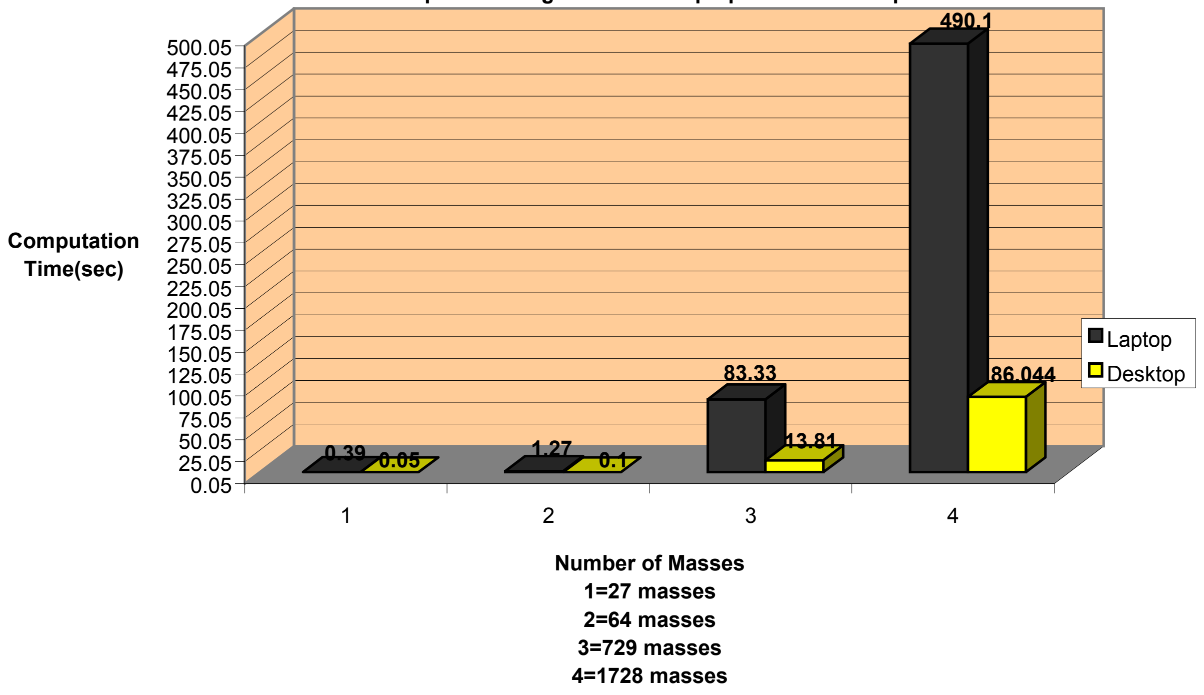
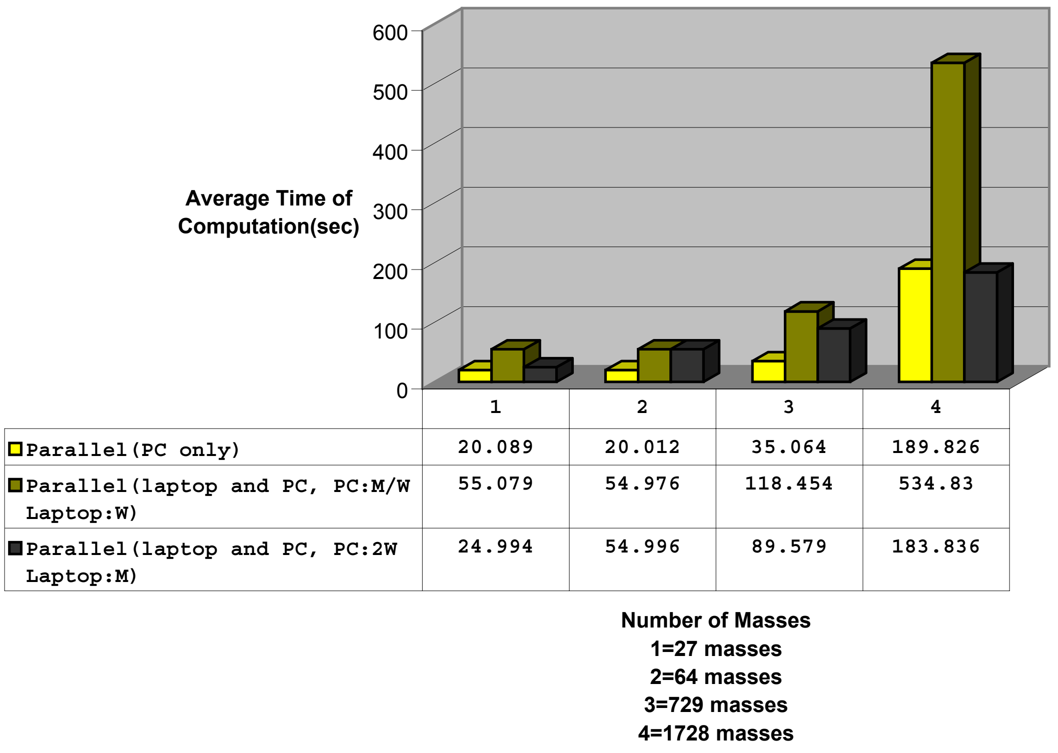


Figure 4:
Performance of Parallel Program in a Parallel Environment



	1	2	3	4
🟨 Parallel(PC only)	20.089	20.012	35.064	189.826
🟩 Parallel(laptop and PC, PC:M/W Laptop:W)	55.079	54.976	118.454	534.83
⬛ Parallel(laptop and PC, PC:2W Laptop:M)	24.994	54.996	89.579	183.836