# 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

**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. This work has been authored by 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.**

# About Myself

- ✓ Alabama A&M University, Junior
- ✓ Mechanical Engineering, G.P.A – 3.74
- ✓ B.S. - 2003
- ✓ Graduate School – Georgia Tech, M.S.
    - Material Science
    - Computer Science

# Introduction

The power and utility of personal computers are growing exponentially through advances in computing capabilities such as:

- ✓ newer microprocessors
- ✓ advances in microchip technologies
- ✓ electronic packaging
- ✓ cost effective gigabyte-size hard drive capacity

# Existing Problem

Many engineering problems require significant computing power. Therefore, the computation has to be done by high-performance computer systems that can cost millions of dollars and need significant memory to complete the task.

# Why Use Parallel Computing

It is feasible to provide adequate computing in the form of clustered personal computers.

- ✓ cuts the cost and size by linking (clustering) personal computers together across a network
- ✓ they can be used as stand-alone computers when they are not operating as a parallel computer

# Parallel Computing

Parallel computing software to exploit the clusters is available for several computer operating systems

- ✓ Unix
- ✓ Windows NT
- ✓ 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.

# Two Part PVM System

- ✓ *daemon* is a special purpose process that runs on behalf of the system to handle all the incoming and outgoing messages; it is represented by “pvmd3” or “pvmd” and any user with a valid login id can install and execute this on a machine

- ✓ *library* of routines that allows the computers to interact “in parallel” (resource and task management “addhost” and “spawn”, pack and send messages…)

# Using PVM

The problem must be able to be broken down into several tasks

- ✓ functional parallelism - breaking the application into different tasks that perform different functions
- ✓ data parallelism - having several similar tasks that are the same solve over different parts of the data

# Writing PVM Code

Using Fortran language

- ✓ Fortran language binders are carried out as subroutines instead of functions
- ✓ Fortran applications have to be linked to three libraries, the PVM Fortran; the standard PVM; and the C sockets, using these links:
  - ✓ libfpvm3.lib
  - ✓ libpvm3.lib
  - ✓ wsock32.lib

# Two Models of PVM Codes

- ✓ master/worker model - the master task creates all other tasks that are designed to work on the problem, then coordinates the input of initial data to each task, and collects the output of results from each task

- ✓ 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

[We used master/worker model for this project]

# Project Description

- ✓ Write a program to calculate force, velocity, and coordinates of masses in a given spring-mass system as it vibrates back and forth in space

- ✓ Written in Fortran, developed in Microsoft Developer Studio, and can run from either the PVM console or MS-DOS command prompt

- ✓ Single-Program Multiple-Data (SPMD) programming model is used, also called “Data Parallelism” which means all tasks are the same but each one computes on a different subset of the data

# Basic Applied Equations

Equations for spring force, velocity, and displacement

- ✓ Force = $K_{spring}(\Delta X)$
- ✓ $X = X_o + V_o \Delta T + [a(\Delta T)^2]/2$ where a = Force/mass

# Equations Used

✓ Calculate Spring Force

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

(E0 = original extension of spring)

✓ Calculate Vector Force to Both Masses

(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

# More Equations

- ✓ Update All Mass Coordinates

(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))

- ✓ Update All Mass Velocities

(4a) Vx(I) = Vx(I) + ( Fx(I) * DT / Mass )

(4b) Vy(I) = Vy(I) + ( Fy(I) * DT / Mass )

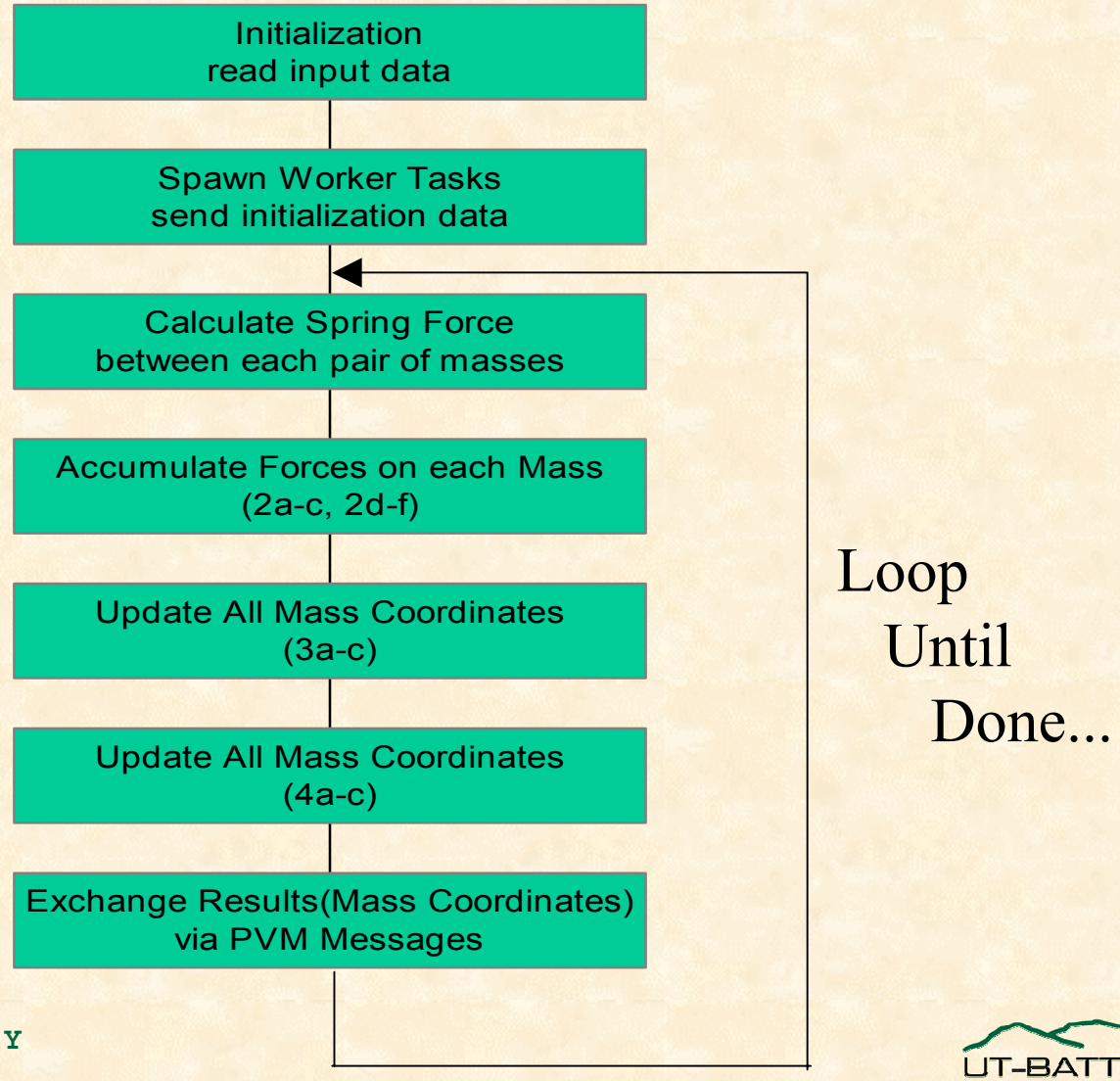
(4c) Vz(I) = Vz(I) + ( Fz(I) * DT / Mass )

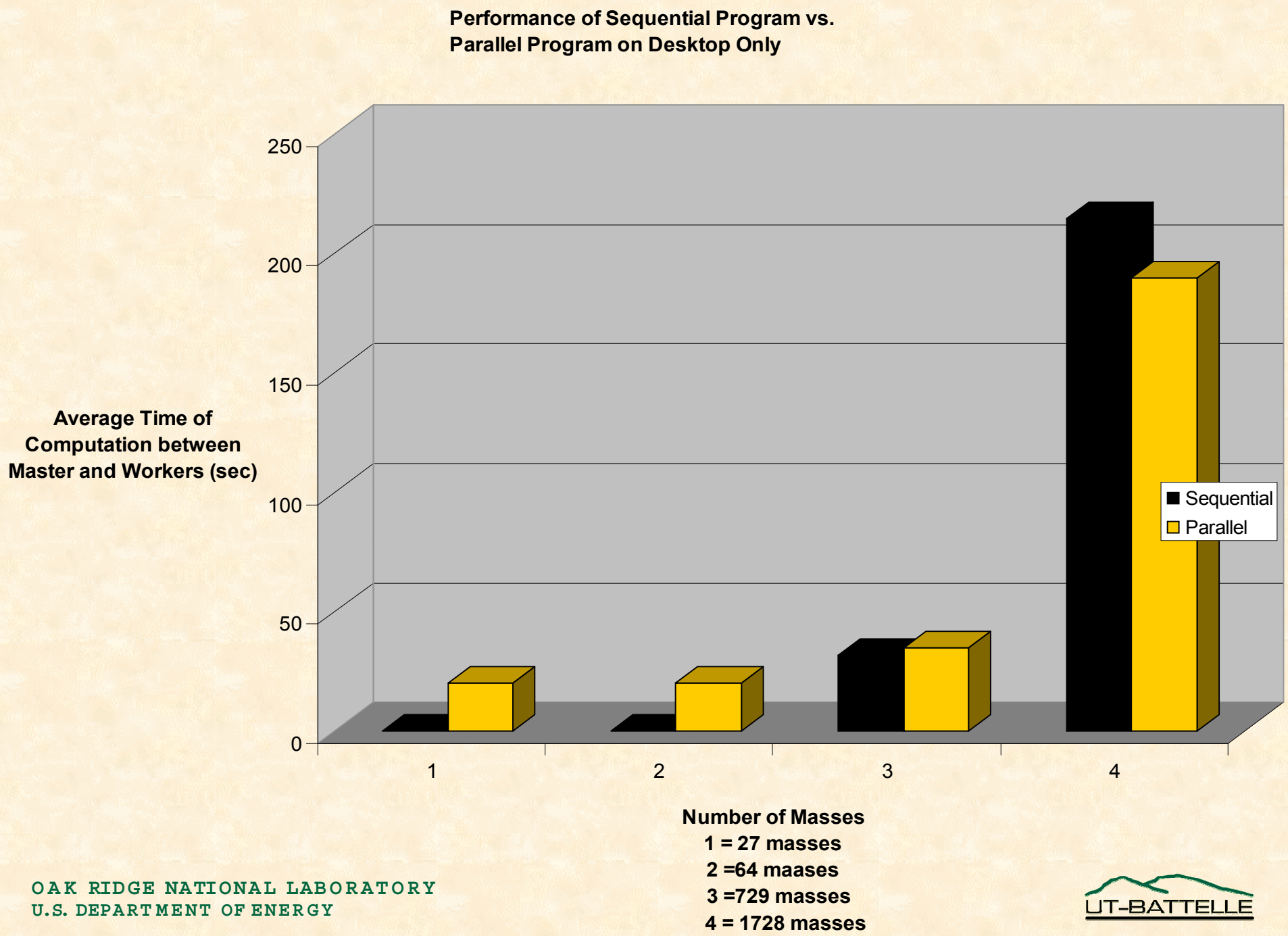
(DT = change in time)

# Method

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). These forces 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. We assume no boundary conditions. Before the velocities are updated (4a-c), they are used in equations (3a-c). DT is the change in time, which is a constant value like Mass and E0 (original extension of spring).

# Program Flow

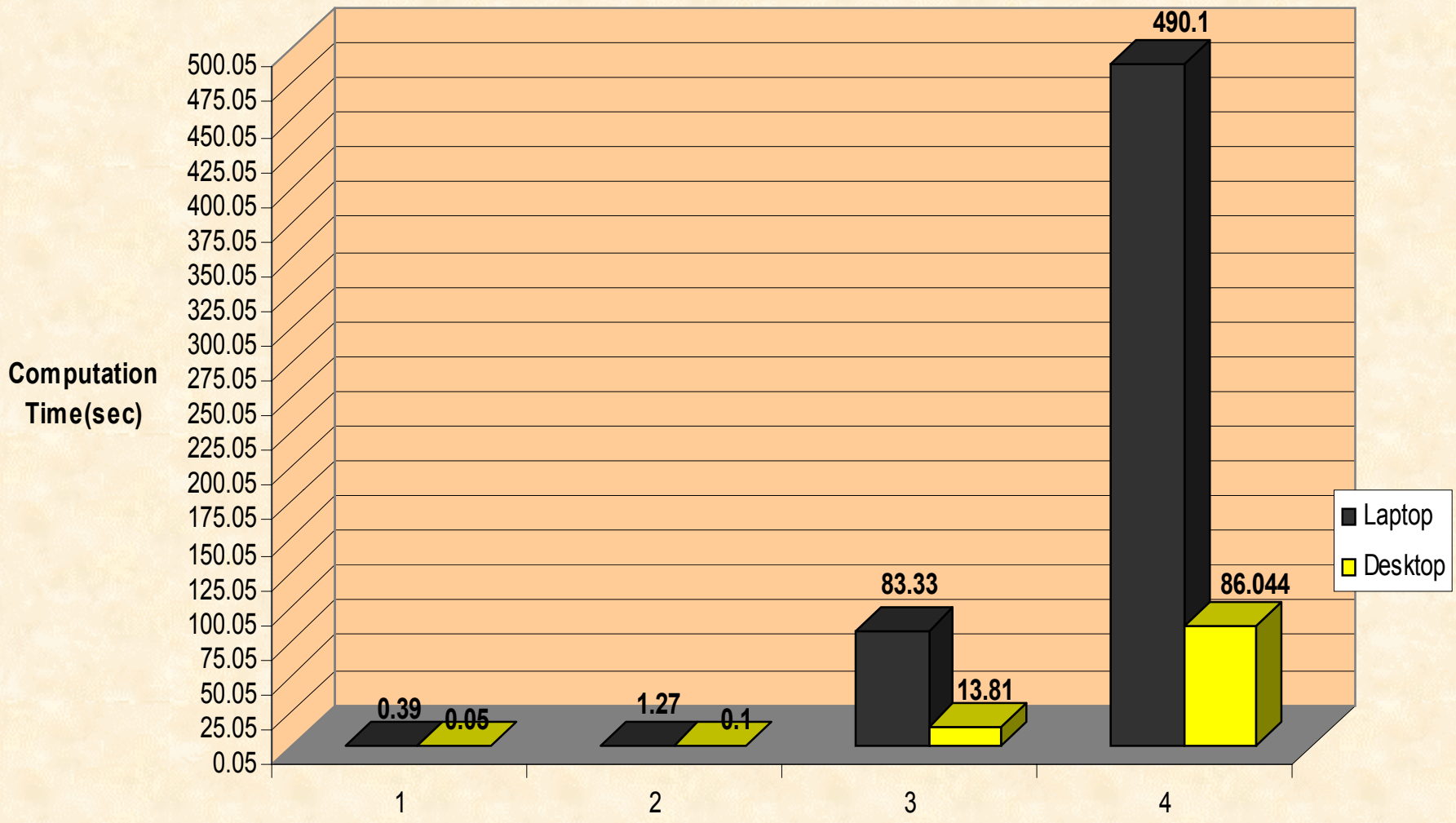



Performance of Sequential Program vs.
Parallel Program on Desktop Only
Average Time of Computation between Master and Workers (sec)
250
200
150
100
50
0
1
2
3
4
Sequential
Parallel
Number of Masses
1 = 27 masses
2 =64 maases
3 =729 masses
4 = 1728 masses

# Importance of the Parallel Environment

In order to collect more efficient and correct data, we need to run the parallel code in a real parallel environment. When a parallel program is ran on a single computer, the overhead of running multiple processes parallel will always make the "parallel" code look slower than the sequential code. Our parallel environment will be created by clustering a PC with a laptop. The PC which obviously has more computing power will be given more data to compute. This should give the desired results that when given a large and complex problem, clustering PCs together will compute much faster than a single computer.

## Performance of Sequential Program on the Laptop vs. the Desktop




Number of Masses

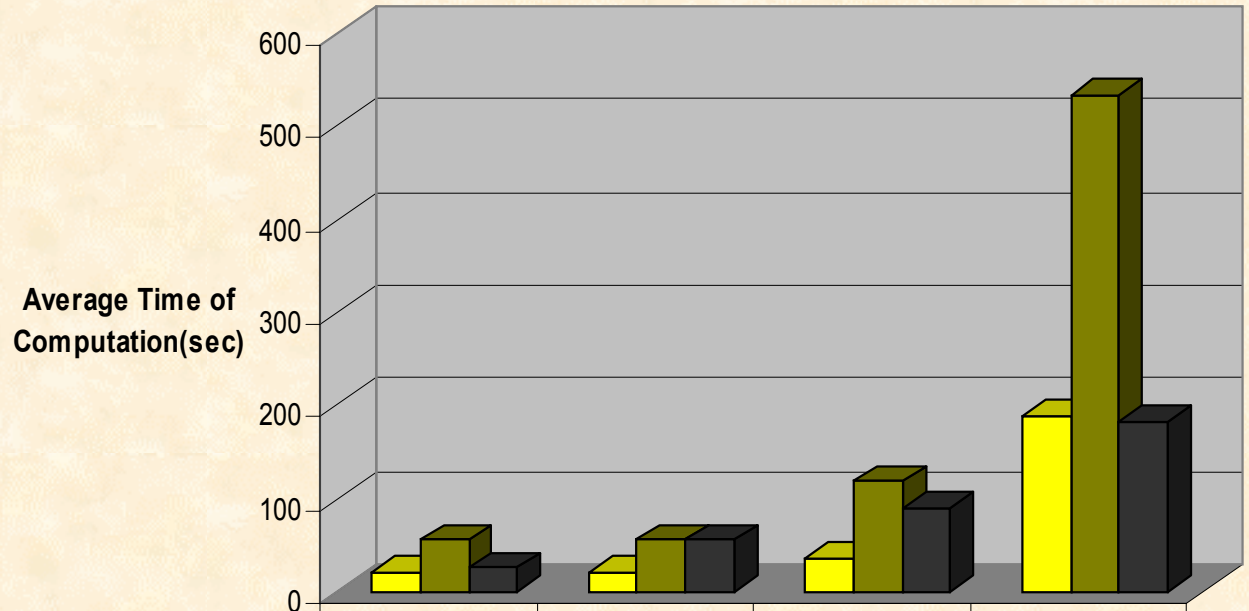
1=27 masses

2=64 masses

3=729 masses

4=1728 masses

## 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 |

**Number of Masses**

**1=27 masses**

**2=64 masses**

**3=729 masses**

**4=1728 masses**

# Optimization of Performance

Upon my potential return to ORNL next summer, two optimizations to this project I plan to explore:

- ✓ Avoid unnecessary message packing
  - ✓ **Pack redundant messages once, send to all**
- ✓ Reduce message size
  - ✓ **Pack minimal necessary message data for each recipient**

# Conclusion

This research has shown that clustering personal computers together can provide adequate computing power for large engineering problems. It has also proven 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.

# Acknowledgements

I appreciate Dr. Z. T. Deng's decision of selecting me to be apart 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. 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.