CSS497 Undergraduate Research

Performance Evaluation of Agent Teamwork

3/31/2008
CSS497 Final Report
Timothy Chuang

Contents

Introduction

I spent the last two quarters working as an undergraduate research assistant for Professor Munehiro Fukuda on the on-going enhancement and evaluation of Agent Teamwork project. This final report documents the work I completed, the process of framework installation and debugging, my experience and lessons I learned.

Agent Teamwork is a high through-put grid computing middleware system that employs mobile agents for job dispatching. To support the on-going enhancement project of Agent Teamwork, my task was to evaluate performance of Agent Teamwork against two contemporary alternatives.

Although there is no single framework that provides an exact match of Agent Teamwork's functionalities, Condor is a good target platform due to its fault-tolerant features. Globus, on the other hand, provides no fault-tolerance features, but serves as a good performance benchmark due to its popularity.

For the purpose of performance evaluation, I built Condor reference platform on the machines in Professor's Distributed Systems Laboratory (DSL), and maintained Globus reference platform which was installed by a former student and attempted to conduct performance evaluation.

However, certain problems arose during the project and as a result, Condor was dropped from the project, and Globus performance evaluation has not been completed. I will go into detail on the problems I encountered.

I will begin by providing a brief overview of the project and then review the installation of the reference platform and parallel applications used in the performance evaluation.

Project Overview

This section provides an overview of the project

- ☐ Establish reference platform
  - ■ Condor Installation
  - ■ PVM installation
- ☐ Implement parallel applications
  Several parallel applications were implemented on three different frameworks for the purpose of performance evaluation. The following applications were coded in C++/PVM, C++/MPICH and Java/MPIJava.
  - ■ Matrix Multiplication
  - ■ Wave2D Simulation
  - ■ Mandelbrot Set Simulation
  - ■ Distributed Grep
- ☐ Check previous Globus installation status
- ☐ Conduct performance evaluation

Reference Platform Installation

The installation of Condor and PVM has been covered in detail in my intermediate report. Therefore, I will simply summarize the process in this section.

Condor and PVM

The installation of Condor was met with many challenges, both technical and logistical. However, after working with UWB's Linux system adminstrators, Mrs. Meryll Larkin and Mr. David Grimmer, I was able to get Condor to run on all machines in the Distrubted Systems Laboratory (DSL).   Due to version incompatibility, current Condor installation only runs on machines in DSL.

The installation of PVM went rather smoothly, with some minor issues with connection between hosts. After working with Mrs. Larkin to determine the cause, it appears that PVM requires UDP port to be open. Currently, PVM runs on all machines in DSL (Medusa and mnodes, Priam, Perseus, Tarvos, and Io) and all machines in the UW1-320-lab.

Implmentations of parallel applications

In order to evaluate performance, I implemented and ported several parallel applications to different frameworks and languages. I attempted to preserve the basic syntax of original java applications in order to create a fair environment for performance evaluation. This section contains implementation detail of different versions of parallel applications used in performance evaluation. Since the PVM versions of those parallel applications have already been covered in detail in my intermediate report, the PVM section will only provide a summary of those applications.

Condor – C++/PVM

The original applications, Matrix Multiplication, Mandelbrot and Wave2D, were written in Java using MPI-java framework. My first task was to port those applications to C++ and PVM. The basic syntax was preserved with minor changes due to the fundamental differences between Java /C++ and MPI/PVM. I also implemented a very simple distributed grep application to measure the file application performance.

Globus - C++/MPICH

In order to run the same applications on Globus, I had to port the existing applications over to C++/MPICH.   The process involved changing the C++ code I had written with PVM framework to MPICH framework.

Agent Teamwork – Java/MPI-Java

The Agent Teamwork versions of those parallel applications were modified to take advantage of Agent Teamwork's fault tolerance features that include check-pointing and recovery in the event of a server crash.

Matrix Multiplication

Globus

There are several differences between MPI-java and MPICH that I had to overcome during the porting process. MPICH has the same syntax as PVM for send and receive. Copying of an array cannot be done by appointing an offset in the send and receive functions. The same work around was applied in C++/MPICH version by creating a temporary array that holds the elements that are to be sent to slave nodes.

Agent Teamwork

This version of Matrix Multiplication uses the exact same syntax of the original Java version. The program was modified to take advantage of Agent Teamwork's fault tolerance features. The program check-points itself once every 1,000 iterations within the main computation loop.

Mandelbrot

Globus

Due to the fact that C++ does not provide a similar API for image manipulation, such functionality was dropped from this version of Mandelbrot. Array dimensions are hard-coded and require further modification if the user wishes to increase the program's capacity.

Agent Teamwork

Buffered Image class is not a serialization class. In order to incorporate Agent Teamwork's fault tolerance features, all objects within the class must be serializable. As a result, the functionality for displaying a jpeg image as output was disabled. The program was then modified to incoporate Agent Teamwork's checkpointing and resumption features. The main computation loop check-points itself once every 1,000 iterations.

Wave2D

Globus

Wave2D also utilizes Java's abstract windowing toolkit class to create a real time graphical display of the result. This functionality was dropped from this version. Array dimensions are hard-coded and require further modification if the user wishes to increase the program's capacity.

Agent Teamwork

The syntax of this version remains the same as the original Java/MPI-java version. The real time graphical display functionality was disabled for performance evaluation purpose.

Distributed Grep

Globus

There is virtually no difference between this version of distributed grep and C++/PVM version.

Agent Teamwork

The basic syntax of this version remains the same as its C++ counterpart, with the exception of Agent Teamwork's own fault-tolerance features.

Globus Status

As of 3/26/08, I verified that Globus can be run as globus account user. Due to logistical challenges the installation imposed, it was difficult to check system status. The help of a root account user was required in order to create valid certificates and run several root-level binaries that came with the installation of Globus. With the help of Mrs. Larkin, the files that required root-access were modified so that globus account user can execute those binaries without root-access. A working certificate has also been signed by Mrs. Larkin and Globus installation should be in good condition and ready for performance evaluation.

This section documents details of Globus platform.

Important Directories

Mpich – contains mpich-g2 binaries and examples
Globus – this is a symbolic link to globus actual installation location. All system level binaries can be found inside.

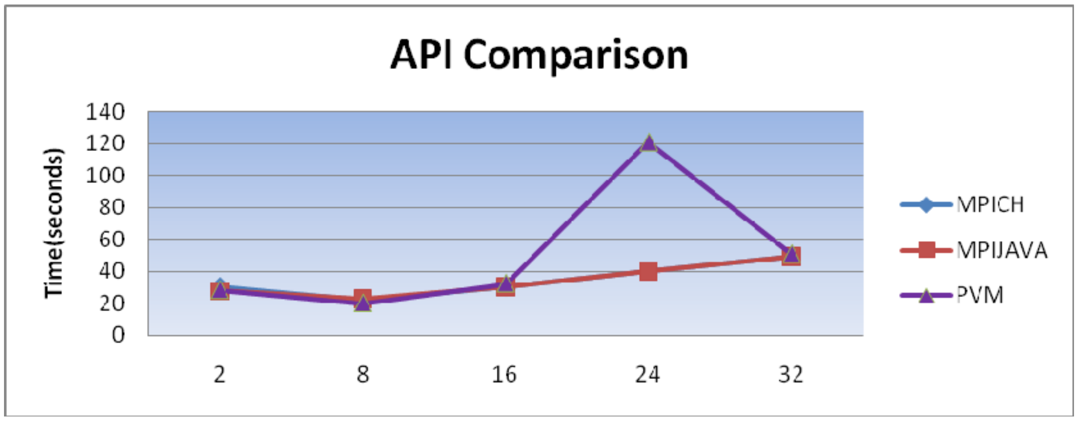
I obtained most of Globus job execution details from http://www3.niu.edu/mpi/ which contains useful documentation on MPICH-G2.

Performance Evaluation

About half way into the project, I learned from the Condor team that PVM is no longer supported by Condor and there was no reliable way to run PVM parallel application using Condor. Therefore, Condor was dropped from the project. Globus test required the help of a root-account user and I was not able to perform full evaluation in time for this report. As a result, this section focuses mainly on the evaluation of Agent Teamwork's fault-tolerance performance.

Framework Evaluation

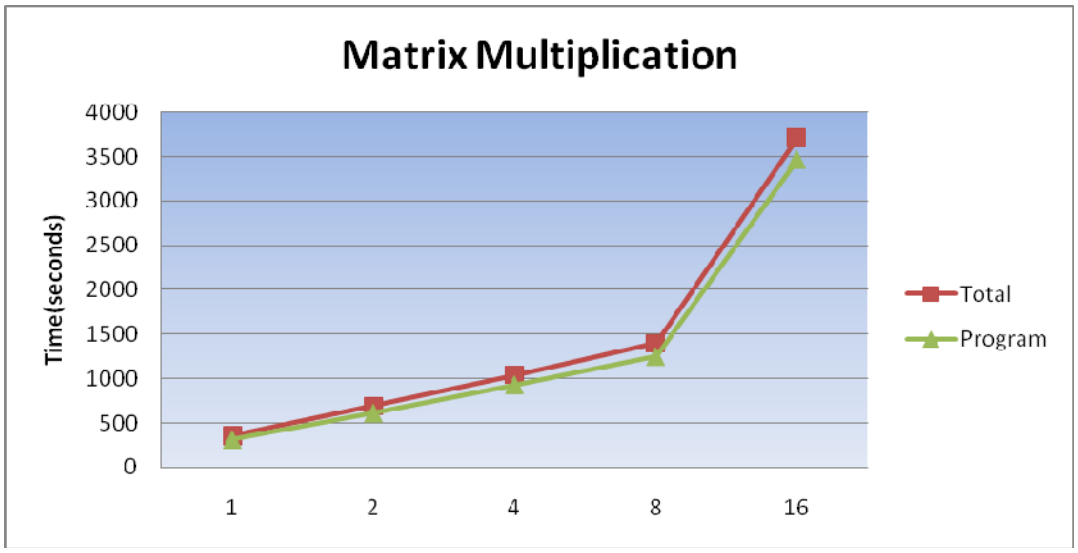
To get a good understanding of the impact of various API on the performance evaluation, performace tests were conducted using the Matrix Multiplcation program.



The result showed that all three APIs have comparable performance. PVM exihibited abnormal behavior with 24 copmuting nodes, but the rest of the results is very similar to the other two APIs.

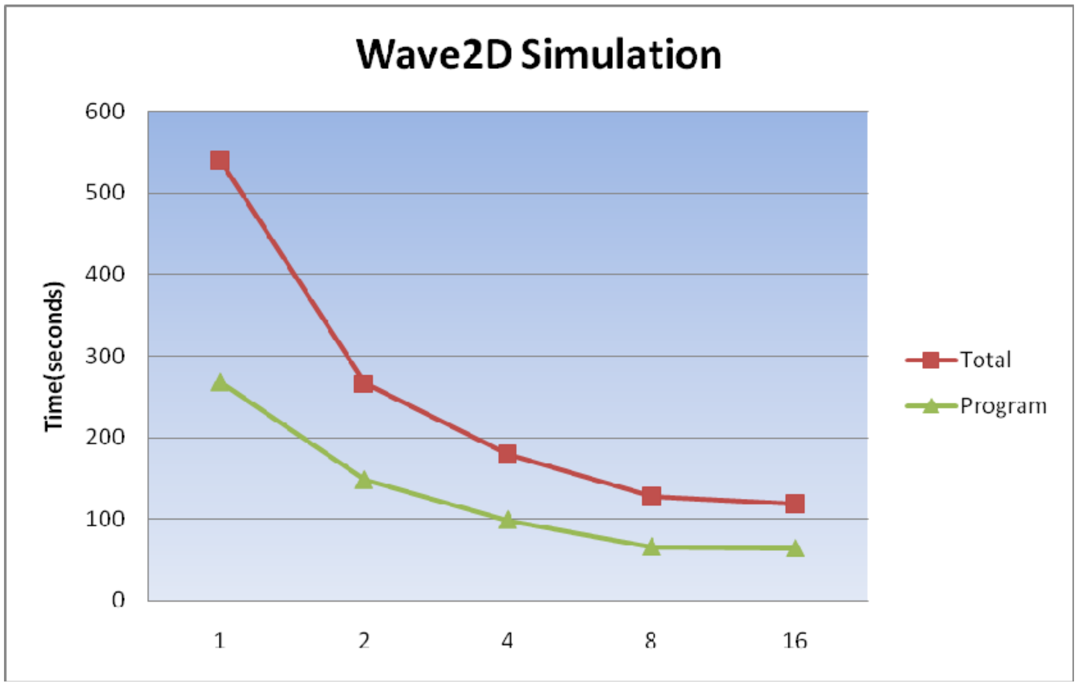
Matrix Multiplication

Matrix size = 1000x1000
It is really difficult to go any higher than 1,000,000 elements, since the master program instantiates three 1,000,000 element arrays.



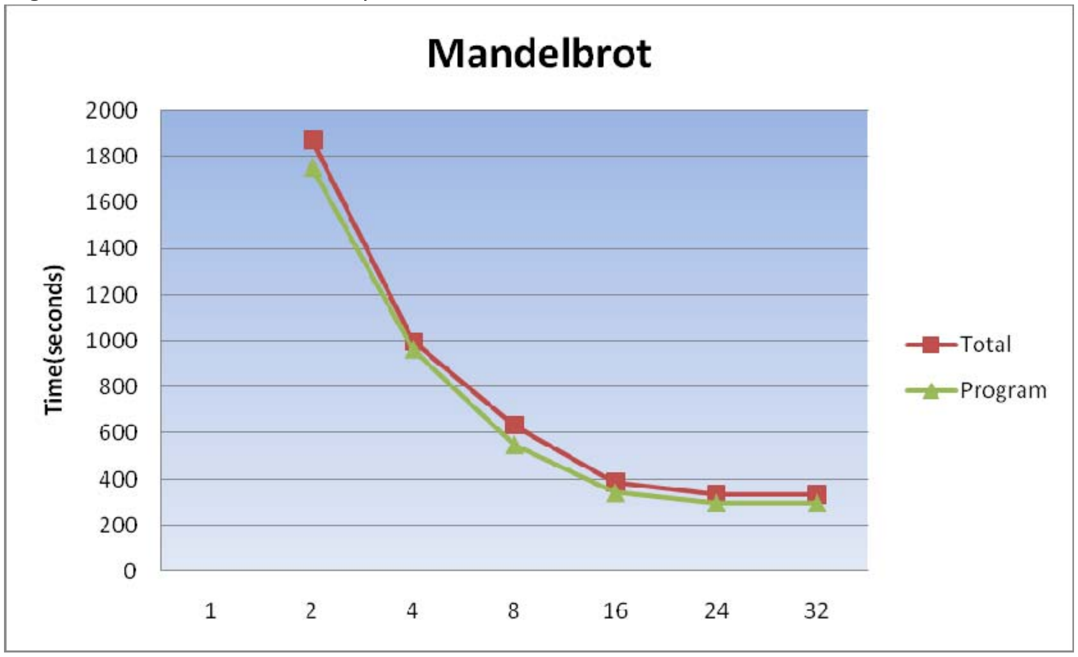
Wave2D

Size = 1500, time = 1500, interval = 500,000


Wave2D Simulation
Time(seconds)
600
500
400
300
200
100
0
1
2
4
8
16
Total
Program

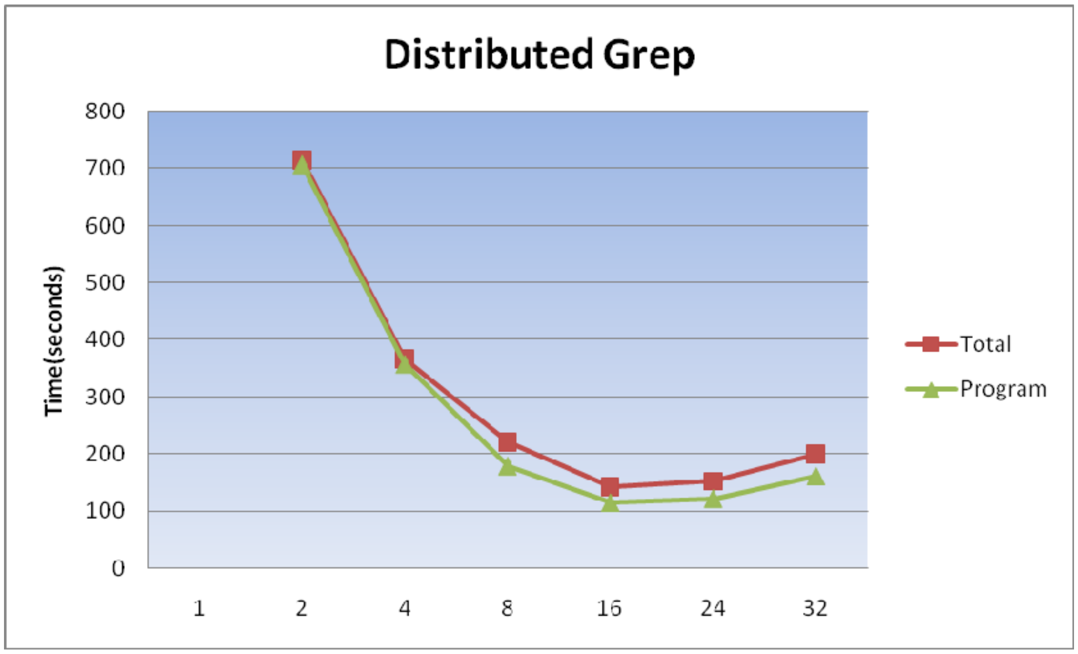
Mandelbrot

Arguments = 0, 0, 0, xRes 1000, yRes 1000, maxIteration 150,000



Distributed Grep

Textfile size = 546,319,436 bytes



Conclusion

My goal was to evaluate Agent Teamwork's performance against contemporary alternatives such as Condor and Globus. However, due to unforeseen circumstances and difficulties, Condor was dropped from the project, and Globus execution was delayed due to complications in the way Globus was set up – most accesses require root-account.

My work showed that Agent Teamwork provides a fully fault-tolerant execution environment. The performance hit from check-pointing is within tolerable margin, usually 10%, and it is up to the programmer to decide how fault-tolerant his/her application is to be, by deciding how often the program's execution state should be backed up, which in turn, slightly affects the performance of the application.

Future Work

Performance evaluation will be continued to take Globus performance into account to give a better sense of the performance of Agent Teamwork's job-dispatch, resumption and checkpointing features. I hope to conduct performance evaluation as time allows, and include the result as appendix to this report in the future.

Raw Data

This section contains raw data obtained during performance evaluation process. Certain cells are marked as NA due to the performance suffering from saturation and are omitted.

Mandelbrot

	Time (seconds)					
	run 1		run 2		run 3	
Node#	Total	Program	Total	Program	Total	Program
1	NA	NA	NA	NA	NA	NA
2	1875.123	1754.323	1820.84	1753.88	1901.002	1825.403
4	998.253	964.354	1020.45	940.644	1012.543	966.043
8	635.506	549.567	644.555	551.35	639.08	548.094
16	387.761	341.523	400.008	360.221	392.943	349.01
24	336.593	297.512	330.001	300.422	331.049	299.403
32	334.504	296.5	315.405	281.401	320.455	282.111

Distributed Grep

	Time (seconds)					
	run 1		run 2		run 3	
Node#	Total	Program	Total	Program	Total	Program
1	NA	NA	NA	NA	NA	NA
2	715.879	705.893	720.889	711.98	717.443	706.11
4	366.765	357.499	370.004	366.57	381.39	370.001
8	220.245	179.837	215.034	174.555	218.333	176.451
16	142.133	115.86	141.033	117.809	144.693	116.913
24	152.241	123.067	149.894	120.987	151.089	121.78
32	200.229	162.6	199.23	160.331	201.478	163.004

Wave2D

	Time (seconds)					
	run 1		run 2		run 3	
Node#	Total	Program	Total	Program	Total	Program
1	443.914	267.63	534.916	267.211	540.144	268.001
2	270.923	148.483	265.524	147.099	266.041	148.041
4	180.211	99.25	180.038	97.145	180.016	98.91
8	146.567	69.262	129.467	73.031	127.885	66.044
16	116.812	73.021	120.058	66.053	119.011	64.476
24	NA	NA	NA	NA	NA	NA
32	NA	NA	NA	NA	NA	NA

Matrix Multiplication

	Time(seconds)					
	run 1		run 2		run 3	
Node#	Total	Program	Total	Program	Total	Program
1	356.338	318.489	361.774	321.889	360.399	322.884
2	689.598	615.658	690.002	617.044	689.439	614.966
4	1041.735	933.63	1030.038	921.981	1049.455	930.54
8	1397.822	1251.16	1395.085	1254.094	1401.043	1260.45
16	3711.09	3462.45	3701.001	3504.54	3705.049	3498.847
24	NA	NA	NA	NA	NA	NA
32	NA	NA	NA	NA	NA	NA