Comparing Performance of Algorithms on Diverse Clusters

Jon Ludwig
Department of Computer Science
Rochester Institute of Technology
Rochester, NY, 14623
Email: jal2656@rit.edu

Brad Israel
Department of Computer Science
Rochester Institute of Technology
Rochester, NY, 14623
Email: bdi8241@rit.edu

Gregor von Laszewski
Department of Computer Science
Rochester Institute of Technology
Rochester, NY, 14623
Email: gregor@rit.edu

***Abstract*—Our project attempts to setup an easily accessible and extensible platform for analyzing a grid's computational performance. A Java web service is used to both collect and serve results and a standalone Java application is used, by the end user, to gather individual system and grid system information and benchmark results. The gathered data is finally shared through the web service's display interface, so that others can compare grids together to find the optimal one for their problem.**

I. Introduction

The overall goal of this project is to develop a platform that makes it easy to perform a comparative analysis of various benchmark algorithms on a selection of different grids and view the consolidated results. The hope is, that the end users will not only share the performance data for the grids that they have access to, but also use our web interface to find the best grid for their unique problem. One of the main design goals for our project is to make the project as easily extensible as possible, so that other colleagues developing web projects can utilize our web services. We have designed the system with flexibility in mind so new benchmarks can easily be added to improve the comparison data, and the benchmarks can easily be run by the end user on many different cluster configurations. More of our design will be discussed in section II and a description of the benchmarks we used and their results will be shown in sections III and IV, respectively.

II. Project Layout and Expansion

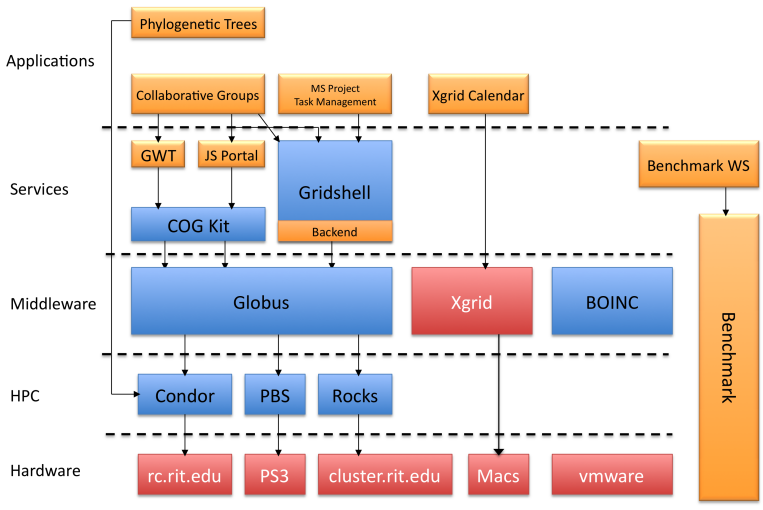

Fig. 1. Projects Overview

Our project, as seen in Figure 1, spans most of the different layers of what we have come to call *cyber infrastructure*. It

is split into two parts, the benchmark script and web services. Our benchmark script executes in the lower layers, which are more closely tied to the various levels of hardware and the topologies of the networks. After the benchmark script tests performance at the lower levels, the gathered information is passed on to our web service, which processes the information into a format that can be easily used by other users.

The general assumptions for our project are that the system that the web server is running on must have an external address or some way for the scripts to access the web services and send grid information. For installation purposes the machine must also have Java 1.6 and Maven 2.x installed and configured. More details on the installation procedure can be found in our project's source directory [5]. In order to run the benchmark script package in a grid environment, the assumption is that the JAR file has been downloaded to the head node of the cluster, or somewhere that is able to connect to each of the individual machines. Each of the machines must also have SSH service and Java installed and configured. The end user needs only to configure the script by editing the *configuration.xml* file. This file informs the script about the grid configuration and other information, such as contact information. The script is run like any other executable JAR package. More details on the configuration and usage of the benchmark script can be found on the default webpage of the web server and or in the benchmark script package.

Since our project spans many of the layers, it was also designed to be an example for most of the other projects in the class. The whole project is written into a Maven project, that can be downloaded and installed in about two commands, which makes it very easy for others to get our entire project running without having to read excessive amounts of documentation. Maven also allows other people to easily modify the project and add their projects in as modules that all share the same web server and libraries, which will make project integration and collaboration much easier. The documentation for how to integrate projects into our Maven project can be found in the Cyberaide project repository [5]. There is also an unpublished academic paper [6] that was done for a different course, but goes into the details of how the Cyberaide Web Projects are setup using Maven.

III. Benchmarks

The benchmark script is flexible enough to run almost any type of command as a benchmark. An internal timing mechanism is used, so the benchmark is not required to format its output in any particular way or report its status to the benchmark script. One requirement is that the benchmark must not be interactive, that is it should not block on user input from the standard input stream. Several benchmarks were considered to demonstrate our project, and ultimately the usefulness user of a benchmark result to a particular user will depend on what type of project the user is planning to run on the grid or cluster.

A. System Benchmarks

There are many different ways to benchmark an individual system. Computational speed is typically the most common metric that is measured by a benchmark. Although, there are other important factors that may influence the completion time of a project, depending on the nature of the work of the project. For example, a task which is heavily I/O bound will benefit little from a system with great amounts of CPU power but an underpowered I/O system.

We chose to use the *NAS Parallel Benchmark*, or *NPB*, developed at NASA [3] as our demonstration benchmark. We chose this benchmark suite because it includes a number of benchmarks which are representative of a common problem solved with grids, *computational fluid dynamics* or *CFD*. NPB

version 3.0 includes benchmarks which are mainly focused on manipulating matrices, solving linear equations, and performing Fourier transforms. The benchmarks are CPU bound and test several important conditions that affect computational speed. For example, the **MG** benchmark implements a multigrid algorithm to solve 3 dimensional scalar Poisson equations. This benchmark tests performance by performing data operations which exhibit good locality, and therefore typically good cache hit rate, and operations which exhibit poor locality. Architectures with different types and amounts of cache will likely show a difference in benchmark performance. Thus it is important to consider the benchmark which correlates well with the type of problem you will be solving.

The NPB is implemented in both *High Performance Fortran* (*HPF*) and Java. We chose to use the Java version so that the benchmark would not need to be recompiled for different architectures and most systems have a JVM which is able to execute the benchmark without important any additional libraries or code. However, if a user desired the HPF version of the benchmark they could easily substitute it for the Java version by configuring the benchmark script appropriately.

The benchmark script is capable of running multiple benchmarks consecutively, and in our demonstration we show an example running four benchmarks from the NPB benchmark suite.

B. Grid Benchmarks

Another reason we chose the NPB benchmark suite from NASA is that there is a corresponding set of benchmarks based on NPB tailored for execution in a grid environment. This grid benchmark suite is called *NAS Grid Benchmarks* or *NGB* [11]. There are two implementations of NGB of interest: a purely Java implementation which uses Java's RMI mechanism for communication, and a Globus 2.4 implementation. We hope that future work with this project will allow us to experiment with these grid-wide benchmarks. Choosing a benchmark suite with related system and grid-wide implementations will allow users to compare the effects of the grid toolkit on performance. For example, measuring the difference between running the system benchmarks on all the machines in a grid and running the grid-wide benchmark which utilizes a grid toolkit such as Globus, will give an estimate of the amount of overhead that the grid toolkit has imposed on the computation time.

IV. Results

We executed the demonstration benchmarks described in section III using our project on a selection of cluster systems. The benchmarks produced results which are compatible with the computing resources for each cluster. The results of the benchmarks are automatically uploaded to the web service, and can then be viewed with the web client, as shown in Figure 2.

Running the benchmarks on a selection of machines in the Computer Science Department at RIT provided the results seen in Table I. It is clear that there is some variation in benchmark time even on systems which are practically identical. This can be contributed to load from other users on the system. Especially the benchmark measurements on *holly*, a popular machine, even it has more aggregate CPU power, its benchmark measurements are worse. If the load on the machine at the time the benchmark is executed is typical, then the benchmark gives a good estimate of how well a job will perform under typical load. This is especially useful for shared machines such as these.

This paper will be updated with the results of running the benchmarks on the lab machines.

Machine Name	Number of Processors	Speed of Processors	MG	FT	IS	BT
holly	8	1.3 GHz	106ms	41ms	177ms	188ms
queeg	4	750 MHz	11ms	29ms	75ms	235ms
california	2	2.2 GHz	34ms	36ms	29ms	47ms
florida	2	2.2 GHz	43ms	41ms	70ms	56ms
idaho	2	2.2 GHz	67ms	39ms	39ms	84ms
illinois	2	2.2 GHz	28ms	38ms	43ms	64ms
indiana	2	2.2 GHz	43ms	33ms	35ms	47ms
kansas	2	2.2 GHz	29ms	34ms	39ms	106ms
maine	2	2.2 GHz	45ms	39ms	45ms	79ms
newyork	2	2.2 GHz	53ms	60ms	33ms	61ms
vermont	2	2.2 GHz	36ms	29ms	40ms	42ms

TABLE I
BENCHMARK RESULTS ON MACHINES IN THE COMPUTER SCIENCE DEPARTMENT

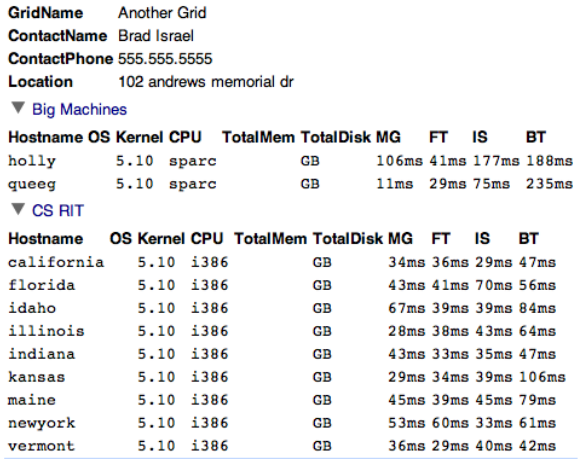

Fig. 2. Example of the client interface

V. FUTURE WORK

R.I.T's short quarter system left our project with quite a few desired features that would improve our project's ease of use, abilities, and comparative value. Some of these improvements may be integrated into the project for the completion of future courses or by others at a future date.

A. Database

The integration of a relational database server like MySQL [8] or PostgreSQL [9] would improve the speed and functionality of the back-end of our project and would be needed to accomplish most of the future work laid out in this section. Our project currently uses an XML file to store and display the submitted grid data, but this is not the most efficient way to do this. Using a real database would allow the user to just send the changes or updates of their grid infrastructure instead of having to re-run all of the benchmarks and send the entire XML file. This increase in communication efficiency would decrease the amount of load put on the web server, which would be desirable for the web site to act as a real-time status display, as described in subsection V-E.

B. Security Issues

Our project does not currently implements any security mechanisms to prevent the insertion of malicious data, falsification of benchmark results, or overwriting of other people's submitted data. This would be a major concern if this project were going to be used on a large scale. Our current assumption for the project is very academic, meaning that we assume that users are going to "play nicely" to benefit the users that are trying to find the best grid for their project. Unfortunately, this never really works in real situations, so we are going to discuss future security improvements that can be made to our project.

The first mechanism we would need is a separation of guest and member access on the website. The guest should be able to view all the grid and system information and the members

should be able to create, modify, or delete their own submitted information. This would be done through a simple registration system on the website. New members would sign up and be given a randomly generated code that they could place in their grid configuration file. Then when the member runs the benchmark script, the code can be verified by the web service before processing and updating the webpage. This is very similar to the way that Google handles svn access on Google Code. The guest users would still be able to see all the data, but wouldn't be able to modify it, and other members could change their grid data, but wouldn't have the correct code to change other members' data.

Now that there is some information being sent from the client to the web service that the user wouldn't want others to see, SSL would need to be configured on the web server. For the Tomcat web server that is currently being used for our project, the setup procedure for SSL is well documented [1]. Using SSL, in this case, would be a secure enough way to send the member's code to the web server without a more complicated challenge-response password system.

The other security problem that will have to be dealt with is a harder one. The project should be able to prevent the user from submitting false or malicious data back to the web service. For malicious data, like SQL injection attacks [12], there are documented ways to "scrub" the data for SQL symbols and keywords that shouldn't appear in any input. For faked benchmark data, however, there is not really an easy way to determine if the user edited the results before sending them back to the web service. For this we would recommend some sort of reputation service, where users of the grid could rate the grid based on their real world experiences, sort of like an E-bay sellers rating. More research would have to be done in this area to find a suitable answer to the problem.

C. Class Benchmarks

Part of the design goals of our project was to make it extensible, so that new benchmarks could be easily added into the project and used to gather better comparison data. Other than the benchmarks we are currently using, as described in section III, we thought it would be good to include a couple of the projects that our colleagues are working on this quarter as benchmarks in a future version. The main projects we had in mind for this would be Joel Lathrop's project [7] on Folkman and Ramsey numbers and Andrew Brown's project [2] on phylogenetic trees. Joel's project has the advantage that it was written to support a few different architectures, including the Playstation 3 cluster, so it would be fairly easy to integrate into our project as a benchmark. With help from Joel, we're sure that we could find a smaller test graph and time how long it takes to run through his algorithm on different grids. Andrew's project seems to be a little more specific to one architecture and language, but it might be possible to use pieces of it in the same way as we would for Joel's. Using custom benchmarks like these would give a more accurate picture of the performance of a grid for solving real world problems.

D. Java Web Start

Java Web Start or JNLP (Java Network Launching Protocol) technology has been around for a while and allows Java applications to be configured and launched from a website onto a local computer [10]. This technology offers a few improvements to our project. It can simplify the installation process while still working on most common platforms like Linux, OS X, and Windows. Our project would have to setup the webpage so that an end user could click on a link to start running the application and configure the application for their grid. JNLP would then be able to download all the needed

benchmark files and run the application on the user's system. This would cut down on the amount of documentation our users would have to read before being able to configure and run the application on their grid.

JNLP can also increase the security of the Java application because it is run in a *sandbox* environment that would not have complete access to the end user's system. This could be an advantage for system administrators that were wary of our application's ability to access important system files and send them back to the web server. It would also have the advantage of making sure that the application that the user ran would always be the most up-to-date version available. For our project this means that the users have the ability to seamlessly run the latest version of the application, which could have more or better benchmarks, without having to reconfigure or save their configuration files.

E. Ganglia Plugin

One of the more interesting expansion areas for our project was to not only collect performance data for different grids, but also to keep track of continuously updated system status. This could be accomplished by writing a plugin for the Ganglia [4] tool, which is already widely used for grid status information. The plugin for this would take some of the information on the individual systems and send it to our web service, which would then be able to display it in an informative way. A possible scenario for this would be if 10 nodes in a 1000 node grid had a power outage, ganglia would notice and tell our web service, which would in turn then update the user interface. This conglomeration of the status of multiple grids would be useful for anyone looking for the best grid to run their project on. The user would not only be able to see the performance metrics of the systems, but also the real-time status of the systems. If the user saw that all the systems in a fast grid were under heavy load, they could opt for a slightly slower grid with less load.

The plugin could also keep our website data up-to-date with the current configuration of a grid without having the administrator re-run our benchmarking application. When integrated into Ganglia, it could automatically remove individual systems from our grid data if they have been offline for a certain number of days or even run the benchmarks and gather system information for any new nodes that come online. This integration into Ganglia would be beneficial to end users who have previously setup Ganglia for their grid and want our website to have the latest data automatically.

VI. Conclusion

Although there are many areas that can be improved in our project, we feel that the initial goals that we set out to accomplish have been completed and more. We have also put in the effort required to make the integration and modifications of our project easier for future students who may want to do so. Hopefully those future students will research and develop some of the ideas we discussed in section V and find new and better ways to extend our project into something that is well polished and even more useful on a large scale.

References

[1] Apache Software Foundation. Apache Tomcat 6.0 - SSL Configuration HOW-TO. Webpage, 2006. Available from World Wide Web: http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html.

[2] A. D. Brown and G. von Laszewski. Parallel programming for grids: Phylogenetic trees. Grid Seminar I Paper, 2008.

[3] M. Frumkin, M. Schultz, H. Jin, and J. Yan. Implementation of the nas parallel benchmarks in java. Technical Report NAS-02-009, NASA Advanced Supercomputing, 2002.

[4] Ganglia. Ganglia monitoring system. Webpage, 2007. Available from World Wide Web: http://ganglia.info/.

[5] Grid Seminar Contributors. Cyberaide webprojects. Webpage, 2008. Available from World Wide Web: http://cyberaide.googlecode.com/svn/trunk/code/cyberaide-webprojects/.

[6] B. Israel. Maven2 for web projects. Development Tools Term Paper, 2008.

[7] J. Lathrop and G. von Laszewski. Solving folkman and ramsey numbers on the playstation 3. Grid Seminar I Paper, 2008.

[8] MySQL AB. Mysql. Webpage, 2008. Available from World Wide Web: http://www.mysql.com/.

[9] PostgreSQL Global Development Group. Postgresql. Webpage, 2008. Available from World Wide Web: http://www.postgresql.org/.

[10] Sun Microsystems, Inc. Jdk 5.0 java web start-related apis and developer guides. Webpage, 2004. Available from World Wide Web: http://java.sun.com/j2se/1.5.0/docs/guide/javaws/.

[11] R. Van der Wijngaart and M. Frumkin. Nas grid benchmarks version 1.0. Technical Report NAS-02-005, NASA Ames Research Center, 2002.

[12] Wikipedia. Sql injection. Webpage, May 2008. Available from World Wide Web: http://en.wikipedia.org/w/index.php?title=SQL_injection&oldid=213109312.