Modeling Parallel Applications Performance on Heterogeneous Systems

Jameela Al-Jaroodi, Nader Mohamed, Hong Jiang and David Swanson
Department of Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE 68588-0115
[jaljaroo, nmohamed, jiang, dswanson@cse.unl.edu]

Abstract

The current technologies have made it possible to execute parallel applications across heterogeneous platforms. However, the performance models available do not provide adequate methods to calculate, compare and predict the applications performance on these platforms. In this paper, we discuss an enhanced performance evaluation model for parallel applications on heterogeneous systems. In our analysis, we include machines of different architectures, specifications and operating environments. We also discuss the enabling technologies that facilitate such heterogeneous applications. The model is then validated through experimental measurements using an agent-based parallel Java system, which facilitates simultaneous utilization of heterogeneous systems for parallel applications. The model provides good evaluation metrics that allow developers to assess and compare the parallel heterogeneous applications performances.

Key Words

Parallel applications, heterogeneous systems, cluster, and performance model.

1. Introduction

High performance, parallel and distributed applications are becoming increasingly resource-intensive, requiring high speed processors, large memory capacity, huge and fast storage systems, and fast, reliable interconnections. However, most applications are confined within a single architecture due to the machine dependant nature of the development environments. Recently some effort has been invested in providing parallel programming capabilities that can simultaneously span multiple heterogeneous platforms. Such environments could provide efficient parallel executions by allowing heterogeneous applications to be matched to suitable heterogeneous platforms. One strong direction in developing these environments is using Java, thus enabling the utilization of heterogeneous systems in executing parallel applications. The success of such efforts has also led to the need for some formal analytical models to evaluate and compare the performance of applications utilizing these systems.

The heterogeneity of a system can be defined in different ways depending on the varying characteristics. In one scenario, for example, a network of workstations (NOW) may be considered heterogeneous due to the varying load on each machine, thus the available resources on each workstation are not always the same. In a second scenario, a heterogeneous system is a collection of different machines with varying architectures, different number of processors and operating environments. In this paper, heterogeneity refers to the second scenario.

Execution of parallel applications on heterogeneous systems can benefit from a suitable model that measures and evaluates their to manage and schedule resources more effectively. The existing performance models addressing this issue are mostly restricted to heterogeneous systems that contain similar (homogeneous) components, but with varying loads on each component. In addition, almost all models assume single-thread machines that execute exactly one task of the application at any given time (see Section 3). This paper presents an enhanced model for measurements that accommodates heterogeneous systems with varying machine configurations and different numbers of processors per machine.

To enable heterogeneous parallel applications, portable and machine independent systems and development tools are needed. Java is considered a very suitable programming language for such applications because Java is machine independent. Therefore, Java code can be compiled once and the resulting bytecode can be executed on any machine with any operating system without any changes. Therefore, much effort has been put into providing Java-based parallel systems [1]. In this paper, we introduce an example agent-based system for parallel Java applications [2]. This system uses a Java object-passing interface (JOPI) [9] and allows simultaneous execution of the application parallel processes on heterogeneous machines. We will use JOPI

for our experiments to evaluate and validate the enhanced performance model.

This paper will first discuss the enabling technology including the agent-based run-time environment and JOPI in section 2. Section 3 describes some of the measurement metrics for parallel applications and the enhancements added to accommodate heterogeneous applications performance evaluation. In section 4, we will present the experimental evaluations using some benchmark applications. Finally, Section 5 concludes the paper with some remarks about current and future work.

2. Heterogeneous Systems

In the context of this paper, we define a heterogeneous system (HS) as a collection of machines of varying architectures, number of processors and operating environments that are connected via a local area network (LAN). However, the prototype system and analysis introduced later in the paper apply to heterogeneous systems connected via other types of networks. An example heterogeneous system is a collection of a multi-node Linux-based cluster connected to an IRIX-based multi-processor machine (e.g. SGI origin 2000) and a collection of Windows-based servers. Some research has been conducted to study and analyze heterogeneous applications performance, but most of it is based on theoretical analysis and considering heterogeneity in terms of varying loads on similar machines [15, 5, 10, 14]. This may be attributed to the fact that, at the time, the available technology did not provide an easy way to deploy and execute applications on varying platforms simultaneously. However, some research has been done proposing models and techniques to implement parallel applications for heterogeneous networks such as in [3].

Although many development tools and languages can be used for such parallel applications and environments, currently, Java's machine independence provides the perfect development tool for parallel and distributed applications that span heterogeneous platforms. Standard Java technologies such as Java virtual machine (JVM) [13] and JINI [7] provide a variety of features to develop and implement distributed Java applications. However, there are some key features lacking in JVM when directly applied as an underlying infrastructure for constructing and executing parallel Java applications on heterogeneous systems. These missing features, needed by different parallel Java programming models, include:

- *Loading user programs onto the remote JVMs of the heterogeneous system nodes,*
- *Managing the heterogeneous system resources,*
- *Scheduling user jobs,*
- *Security,*
- *Job and thread naming, and*
- *User commands and collective operations*

A number of research groups have worked on providing parallel capabilities in Java [1], which is portable by nature. However, there is little work done on the utilization of heterogeneous platforms to execute parallel applications and analyze their performance. This section discusses an example Java-based environment that facilitates executing parallel Java applications on heterogeneous systems. The main advantages of utilizing heterogeneous systems are:

1. Achieving high performance results at a relatively low cost compared to multiprocessor parallel machines.
2. Easier expandability and upgrade of the hardware by adding more machines that are not necessarily the same as the existing ones or replacing existing machines by newer ones.
3. Allowing for better utilization of the different architectures by scheduling application components on the machines most suitable for them. For example, if some sections of the application are tightly coupled and require frequent communications, these sections can be scheduled on an MPP or SMP machine, while other sections that are relatively independent can be scheduled on a cluster of workstations in the system.

We implemented an example parallel Java system that supports these requirements. The system utilizes objects for information (data and code) exchange and can simultaneously span multiple heterogeneous platforms. The next sub-sections will discuss this system and Section 4 will show the experiments done using it.

2.1 An Agent-Based Infrastructure

The agent-based system prototype system, discussed in details in [2], is designed to satisfy the requirements for heterogeneous parallel applications support. The system provides a pure Java infrastructure based on a distributed memory model. This makes the system portable, secure and capable of handling different programming models on heterogeneous systems. The system also supports multi-threaded execution on multi-processor machines. The system also has a number of components that collectively provide middleware services for a high performance Java environment on clusters and heterogeneous networked systems such as the software agents and client services.

The main functions of the system are to deploy, schedule, and support the execution of the Java code, in addition to managing, controlling, monitoring, and scheduling the available resources on clusters or on a collection of heterogeneous networked systems. Java serializable objects [13] were used as a communication unit among the system's components where each object represents a specific type of request or reply. The client services and environment APIs provide commands for users to interact with the environment. The system also provides features for remote deployment, job monitoring and control, job and thread naming and security.

2.2 The Java Object-Passing Interface

The agent-based infrastructure is capable of supporting different parallel and distributed programming models. For example, implementing distributed shared object API's for Java (work in progress) and Java object-passing interface (JOPI), which is discussed in more details in [9]. In addition, distributed applications can utilize this infrastructure to facilitate their operation. JOPI was implemented on top of the services provided by this infrastructure. JOPI is an object-passing interface with primitives similar to those in MPI, where information is exchanged by means of objects instead of messages. Using JOPI, users can initiate point-to-point communications in synchronous and asynchronous modes, group communications and synchronization, in addition to a other supporting functions and attributes.

JOPI utilizes most of the features provided by the agent-based infrastructure, including scheduling mechanisms, deployment and execution of user classes, control of user threads and available resources, and the security mechanisms. Writing parallel programs using JOPI is generally simpler than using C and MPI, mainly due to the object-oriented nature of Java, thus the user can define a class for solving the sequential version first and test it, then use it as the main object in the parallel version. A few additional methods and classes can be added to the original classes to facilitate the decomposition and distribution of the problem, exchange of intermediate results, and reconstruction of the solution. Finally, the main class will implement the parallelization policy such as deciding on the size of sub-problem, load balancing, etc. Passing objects in JOPI preserves the object-oriented advantages in Java and simplifies the parallelization process and the information exchange mechanisms.

3. The Performance Model

With the technology available to develop and execute heterogeneous parallel applications on heterogeneous systems, it is necessary to provide a model that will provide a better understanding of the performance of such applications. There has been a long debate on what is the best method to represent the performance of a parallel application. Generally, the speedup and efficiency metrics are considered adequate given the appropriate description of how exactly they are measured and reported. In some cases, however, speedup can be misleading because it may give a distorted picture. The one metrics that almost all agree on is the elapsed time (response time) [4]. Elapsed time is directly measured for the application and it encapsulates all affecting attributes such as system time and communication time. In addition, it represents what the user can relate to as a performance measure. However, it is not useful when trying to predict the performance of the application with different settings such as larger input sets or more processors. In addition, Most measurements and analyses are done at the application level rather than the lower level to provide a practical view of how parallel applications will perform on a heterogeneous system and what the user can expect to achieve with different system configurations or algorithms.

The other issue that needs to be considered is the type of environment used to execute the parallel application. Most of the models and metrics, developed earlier, are based on the homogeneous environment, thus, not accounting for the possible differences between the participating machines. For example, speedup is measured as the elapsed time at one processor T_1 divided by the time at p processors T_p. In this case, if some of the processors are slower than others, the effect will not be evident. Some research has been conducted to consider the heterogeneity of parallel applications and some basic metrics were defined in [15, 5, 10, 14].

In [15], the authors introduced a model to measure and evaluate parallel applications performance on heterogeneous networks. However, the heterogeneity was defined based on the different loads on the participating machines. Therefore, the model assumed a cluster of similar machines connected by a uniform network and each machine can execute exactly one task (process) at any given point of time. However, current technology and development tools allow for utilizing heterogeneous systems, which these models do not consider. The remainder of this section briefly discusses the model introduced in [15] and extends it analytically to accommodate for heterogeneous systems, where participating machines have multiple processors, different architectures and varying CPU/Memory/IO performances. In addition, the enhanced model incorporates machines that execute multiple tasks simultaneously.

The power of a machine M_i is defined by the amount of work the machine can complete in unit time. The power weight $W_i(A)$ of machine M_i with application A is given as the amount of work M_i can complete relative to the fastest available machine in the system. Thus, the power weight of a machine can be calculated by either equation (1) or (2) as follows

$$W_i(A) = \frac{S_i(A)}{\max_{j=1}^{m}\{S_j(A)\}} \quad \text{.....(1)}$$

$$W_i(A) = \frac{\min_{j=1}^{m}\{T(A, M_j)\}}{T(A, M_i)} \quad \text{.....(2)}$$

where

- $S_i(A)$ is the speed of the machine (MIPS or FLOPS) executing application A.
- $T(A, M_i)$ is the elapsed execution time for application A on machine M_i.

- m is the total number of machines in the heterogeneous system *HS*.

Using the Power weight of the machines provides a more accurate measure of the effect each machine will have on the system, thus giving a clear view of how the system performs. Using this definition [15] derives the speedup *SP*, efficiency *E*, and the system heterogeneity *H* definitions such that

$$SP = \frac{\min_{j=1}^{m}\{T(A, M_j)\}}{T(A, HS)} \text{.....(3)}$$

$$E = \frac{SP}{\sum_{j=1}^{m} W_j(A)} \text{.....(4)}$$

$$H = \frac{\sum_{j=1}^{m}(1 - W_j(A))}{m} \text{.....(5)}$$

Based on the above model, we introduce a number of assumptions to extend the model for a truly heterogeneous system. The major difference is that any machine can execute multiple parallel tasks simultaneously. The original model assumes that each machine in the system executes a single task at any point of time; therefore, the power weight can be calculated using the machine's speed. However, in the current situation, each machine can have one or more processors and can simultaneously execute multiple tasks of the application.

1. The power factor calculations remain the same using the elapsed time of the application on a single processor of that machine. However, it is also possible to consider the total number of processors used on that machine as one, thus, taking the power factor based on the execution time using all the processors. Nevertheless, this will lead to having to recalculate the power weight of all machines every time the number of processors used changes.
2. Machines are dedicated, thus no owner compute time is involved. However, other effects on time such as system and idle time and communication costs are considered in the model as part of total elapsed time.
3. The measurements are application dependant to a limited extent. In many cases the effect of the application remain negligible as long as the application and its data fit in main memory during execution, thus no page swapping is involved. Without the overhead of memory paging, applications will behave similarly.
4. To calculate the power weights of machines we rely on the execution time of the application, not the speed of the machine. The time gives a more accurate measure of the machine's overall performance because it accounts for all affecting attributes such as actual CPU time, idle time and I/O time.
5. The model is applied at the application level to reflect the user's perception of the application performance. In addition, this simplifies the model by incorporating all time considerations within a single measurement.

This form provides a measure of the relative machine powers with respect to the parallel application used. When multiple processes execute on the machine, the elapsed time will include processing time and additional overhead imposed by both the system and communication. Thus, the efficiency calculations need to be adjusted to accommodate for the varying number of processors on each machine. Similarly, calculating the heterogeneity of the system requires accounting for the number of processor used on each machine.

$$E = \frac{SP}{\sum_{j=1}^{m} W_j * n_j} \text{.....(6)}$$

$$H = \frac{\sum_{j=1}^{m}(n_j * (1 - W_j(A)))}{m} \text{.....(7)}$$

where n_j is the number of processors (or tasks) used in machine M_j. By multiplying the number of processors n_j by the power weight W_j of the machine M_j, we can represent the expected power weight of that machine using the n_j processors rather than a single processor. Based on equation (6), the power weight of a machine M_j is its weight using a single processor. The product $n_j * W_j$ gives an estimate of the power weight when n_j processors are used to execute n_j tasks of the application. Therefore, this estimate does not account for the communication overhead imposed between the n_j processors within the machine M_j. However, the overhead is relatively very small because the processors reside in a single machine and within close proximity of each other.

Equation (7) represents the true effect of having machines with different configurations and computing power, hence providing a representative evaluation metric for applications performance. Using equation (7), it is also possible to reverse the calculations to predict the execution time of the application if part of the configurations has changed. Given the efficiency and power weights of the current configuration, the speedup can be calculated from equation (6). Using the speedup determined and equation (3), *T(A, HS)* for the new configuration can be calculated. This value is an approximate value of the expected performance with the new configuration. The efficiency value used can either be the same as the current configuration efficiency or be an estimated value of the next possible efficiency using the available measurements. The efficiency can be estimated by extrapolating the efficiency graph against the number of processors used. This is achievable by calculating the efficiency values for a number of measurements, plotting the results and employing a curve fitting technique on the plot. A simpler linear extrapolation can also be used, but this will give a less accurate estimate.

Similarly, the expected elapsed time for the application with a different input size can be estimated. The model provides a uniform method to measure and evaluate parallel application performance on a heterogeneous system. The model relies on the measured elapsed time to calculate the power weight of the machines. Using the power weights and elapsed time of the parallel application, speedup, efficiency and heterogeneity can be calculated. Moreover, the calculations can be reversed to estimate the elapsed time for other problem settings such as larger data sets or different system configurations. However, one shortcoming is that it may require recalculating all the power weights of the machines if the fastest machine was removed or replaced by a faster one.

4. Model Validation

Some benchmark applications were written to evaluate the performance of the heterogeneous system using JOPI. All experiments used standard JVM sdk 1.3.1 and were executed on homogeneous and heterogeneous clusters. The platforms used are listed in table 1.

Name	Platform Description
CSNT	3 CPUs, Intel x86 700MHz, 1.5GB RAM. OS: Windows 2000 advanced server
RCF	Origin 2000, 32 processors, 250 MHz, 4MB cache, 8GB RAM. OS: IRIX 6.5.13
SH	Cluster, 24 nodes, dual 1.2 MHz AthlonMP, 256KB cache, 1GB RAM per node. OS:Linux

Table 1: A list of Machines used in the experiments

4.1 Traveling Salesman Problem (TSP)

The algorithm is based on branch-and-bound search [8]. This problem required using many of the JOPI primitives to implement an efficient load-balanced solution. Broadcast was used to distribute the original problem object to processes and used by processes to broadcast the minimum tour value found. This allows other processes to update their minimum value to speedup their search. In addition, asynchronous communication is used by the processes to report their results to the master process, while continuing to process other parts. The results obtained (Figure 1) show good speedup measures with growing number of processors and fixed problem size (22 cities).

The TSP was also executed on a heterogeneous cluster consisting of nodes from SH and CSNT. In this case the initial calculations of the power weight shows that SH is the fastest machine and CSNT has around .94 power relative to SH. TSP was executed using different

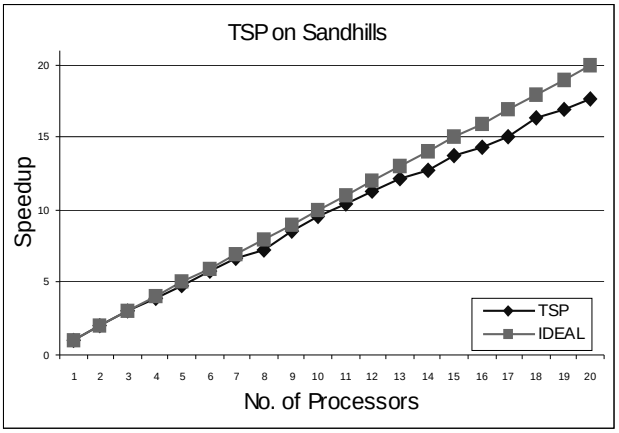

Figure 1. Speedup results for TSP (22 cities) on a homogeneous cluster.

configurations of heterogeneous processors from SH and CSNT (see table 2). The results show that the efficiency at four processors is 64%, which is low because of the communication overhead, but when more of SH (the faster machine) processors are added, the efficiency increases.
Using the information in table 2, we can estimate the elapsed time for the application if the configurations change. For example, assume we need to estimate the time for the 6-2 combination, then we first estimate the expected efficiency. This is done either by using the efficiency from the 4-2 combination (77.8%) or by estimating the value by methods like curve fitting. For illustration, we will use the available value, thus equation (7) gives the speedup for the new configuration to be 6.1323. Using equation (3), T(A, HS) is calculated to be 5806912.415ms. Compared to the experimental result, the percentage error is only 3.1 percent.

No. of Processors			**Time (sec)**	**Speedup**	**Efficiency (%)**
Total	**SH**	**CSNT**			
1	1	0	3560.9729	1	100
1	0	1	37839.885	0.941063	100
2	0	2	18765.547	**1.897612**	**97.7615**
2	2	0	18068.563	**1.970811**	**98.5406**
4	2	2	14112.931	**2.523199**	**64.9953**
6	4	2	7782.309	**4.575728**	**77.7904**
8	6	2	5633.393	**6.321187**	**80.1965**
PW	**1**	**0.9411**			

Table 2. Performance measurement for TSP on a heterogeneous system.

4.2 Matrix Multiplication

A dense matrix multiplication (MM) algorithm [6] is used with load balancing mechanism and synchronous

point-to-point communication. The first matrix is divided into blocks of rows and the second is divided into blocks of columns. Each process gets a block of rows and a block of columns and when work is done, the process sends its results to the master process and takes new blocks to compute. Here, a matrix of size 1800x1800 floating numbers was used, with a stripe size of 300 rows or columns. The initial results, on Sandhills (SH), RCF and CSNT are illustrated in Figure 2.

The second part of the experiment was conducted using the MM on a heterogeneous cluster consisting of processors form RCF, SH, and CSNT (see Table 3).

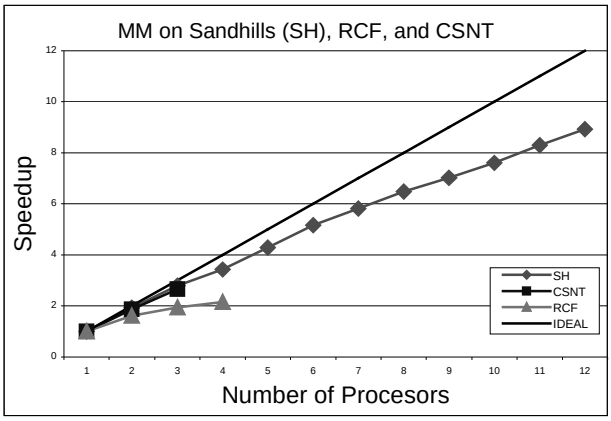

Figure 2: Speedup for the MM on homogeneous clusters.

The results reported were derived from the time measurements of the application (see Figure 3) using the model introduced in Section 3. The power weight PW of the machines show that SH has the best response time thus is taken as the reference for PW calculations using equation (2) for RCF and CSNT. In addition, the speedup was calculated using equation (3); however, from the results it is evident that speedup alone does not provide an accurate representation of the heterogeneity of the system. The efficiency, on the other hand, was calculated using equation (7) and it takes into account the differences in machine performances and gives a better representation of the performance.

By comparing with the experiment on SH alone, the efficiency results show how the different speeds of machines in the heterogeneous cluster affected the outcome. For example, at six processors on SH we get 86% efficiency, while using three processors in SH and three from CSNT, which has similar speed as SH, we achieved 81%. However, at 12 processors in SH we achieved 74% efficiency as opposed to 63% using six processors from SH, three from CSNT and 3 from RCF, which is much slower than the other two machines. In addition, if we compare the efficiency calculated here with the efficiency calculated by dividing the speedup by the total number of processors used (regardless of their contribution) we find that it becomes lower for systems containing more slow machines than fast ones. For example, for the same configuration mentioned above, the direct efficiency (speedup / No. of processors) is 51%, which is lower than the one calculated by equation (7). Moreover, if we consider a configuration with machines of similar power weights, the efficiency is similar in both cases. As shown in this experiment, the measurement metrics provided by the model for heterogeneous parallel applications resulted in a better understanding and more accurate representation of the results.

No. Processors				**Elapsed Time (sec)**	**Relative Speedup**	**Efficiency (%)**
Total	**SH**	**RCF**	**CSNT**			
1	1	0	0	280.83	0.94429	1
1	0	1	0	807.65	0.32835	1
1	0	0	1	265.19	1	1
3	3	0	0	100.93	**2.62734**	**92.7448**
3	0	3	0	416.49	**0.63673**	**64.6401**
3	0	0	3	100.24	**2.64566**	**88.1888**
6	3	0	3	55.708	**4.76032**	**81.612**
12	6	3	3	42.992	**6.16831**	**63.9152**
16	13	0	3	34.591	**7.66639**	**50.1866**
PW	**0.94**	**0.33**	**1**			

Table 3 Performance measurement for MM on a heterogeneous system.

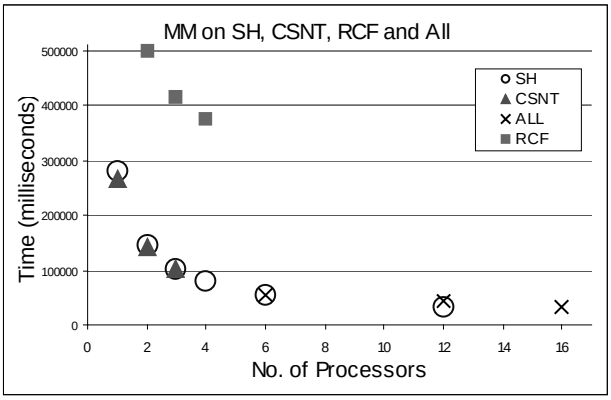

Figure 3. Elapsed time for MM on homogeneous clusters and heterogeneous systems**.**

4.3 Discussion

As shown by the experiments above, the proposed model gives a better representation and understanding of the heterogeneous parallel applications. The efficiency

and heterogeneity of the system can be calculated in a way that reflects the true configurations of the system. In addition, the experiments show how the model fits with the experimentally measured values. Finally, the model can also be used to estimate the possible elapsed time for an application with different data sets or different system configurations. This information would be essential in helping decide what machines and how many need to be used to achieve the desired levels of efficiency and response time. Moreover, the existence of the agent-based parallel Java enabled these experiments and provided a valuable opportunity to validate the proposed model.

5. Conclusion

The model introduced in this paper extends some available metrics to evaluate the performance of heterogeneous parallel applications. The extensions mainly accommodate the varying platforms and operating environments used and the possibility of having multiple tasks of the parallel application on each machine. Using the enhanced metrics, it is possible to calculate the speedup and efficiency of the application and further estimate the performance for a different problem size or different heterogeneous system configurations. Generally, this model is essential because current technology allows for the simultaneous utilization of heterogeneous systems to execute parallel applications. In this paper, we discussed an example for such a system, the agent-based infrastructure and JOPI, which were used to develop the experiments and evaluate the introduced model. The results show that heterogeneous systems can provide ample computing power without the need to upgrade or change the available systems. In addition, the performance model provides a means of evaluating these applications and representing the results in a way representative of the heterogeneous nature of the environment. However, some automated measurement and profiling tools may be helpful in determining performance values. This is left for future investigation.

6. Acknowledgement

This project was partially supported by a National Science Foundation grant (EPS-0091900) and a Nebraska University Foundation grant (26-0511-0019), for which we are grateful. We would also like to thank other members of the secure distributed information (SDI) group [12] and the research computing facility (RCF) [11] at the University of Nebraska-Lincoln for their continuous help and support.

References

[1] Al-Jaroodi, J., Mohamed, N., Jiang, H. and Swanson, D., "A Comparative Study of Parallel and Distributed Java Projects for Heterogeneous Systems", in Workshop on Java for Parallel and Distributed Computing at IPDPS 2002, Ft Lauderdale, Florida, 2002.

[2] Al-Jaroodi, J., Mohamed, N., Jiang, H. and Swanson, D., "An Agent-Based Infrastructure for Parallel Java on Heterogeneous Clusters", in proceedings of IEEE International Conference on Cluster Computing (CLUSTER'02), Chicago, Illinois, September 23 – 26, 2002.

[3] A. A. Aly, A. S. Elmaghraby, and K. A. Kamel, Parallel Programming on top of Networks of Heterogeneous Workstations (NHW), Proceedings of The International Society for Computers and Their Applications, International Conference on Computer Applications in Industry and Engineering (CAINE-98), Las Vegas, Nevada., W. Perrizo (editor), pp: 115-118.

[4] Crowl, L., "How to Measure, Present, and Compare Parallel Performance", IEEE Parallel and Distributed Technology, Spring 1994, pp 9-25.

[5] Donaldson, V., Berman, F. and Paturi, R., "Program Speedup in Heterogeneous Computing Network", Journal of Parallel and Distributed Computing 21, 316-322, 1994.

[6] Gunnels, J., Lin, C., Morrow, G. and Geijn, R., "Analysis of a Class of Parallel Matrix Multiplication Algorithms", Technical paper, 1998, http://www.cs.utexas.edu/users/plapack/papers/ipps98/ipps98.html

[7] The Jini Community http://www.jini.org/

[8] Karp, R. and Zhang, Y., "Randomized Parallel Algorithms for Backtrack Search and Branch-and-Bound Computation", Journal of the ACM, V. 40, 3, July 1993.

[9] Mohamed, N., Al-Jaroodi, J., Jiang, H. and Swanson, D., "JOPI: A Java Object-Passing Interface", in proceedings of the Joint ACM Java Grande-ISCOPE (International Symposium on Computing in Object-Oriented Parallel Environments) Conference (JGI2002), Seattle, Washington, November 2002.

[10] Post, E. and Goosen, H., "Evaluating the Parallel Performance of a Heterogeneous System", in proceedings of HPC Asia 2001, Australia, September 2001.

[11] Research Computing Facility at UNL home page. http://rcf.unl.edu

[12] Secure Distributed Information Group at UNL. http://rcf.unl.edu/~sdi

[13] Sun Java web page http://java.sun.com

[14] Xiao, L., Zhang X. and Qu, Y., "Effective Load sharing on Heterogeneous Networks of Workstations", in proceedings of IPDPS 2000, Mexico, May 2000.

[15] Zhang, X. and Yan, Y., "Modeling and Characterizing Parallel Computing Performance on Heterogeneous Networks of Workstations", in proceedings of the 7th IEEE Symposium on Parallel and Distributed Processing (SPDPS'95), 1995.