Distributed Systems Middleware
A Framework for Parallel and Distributed Computing on Heterogeneous Systems

By
Jameela Al-Jaroodi

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Hong Jiang

Lincoln, Nebraska

August, 2004

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

DISSERTATION TITLE

Distributed Systems Middleware - A Framework for Parallel and Distributed Computing on Heterogeneous Systems

BY

Jameela Al-Jaroodi

SUPERVISORY COMMITTEE:

	Approved	Date
Signature		6/22/04
Typed Name	Dr. Hong Jiang	
Signature		6/22/04
Typed Name	Dr. Sharad Seth	
Signature		6/22/04
Typed Name	Dr. Steve Goddard	
Signature		6/22/04
Typed Name	Dr. Xiao Cheng Zeng	
Signature		6/22/04
Typed Name	Dr. Stephen Scott	
Signature		
Typed Name		

University of Nebraska | GRADUATE COLLEGE

Distributed Systems Middleware

A Framework for Parallel and Distributed Computing on Heterogeneous Systems

Jameela Al-Jaroodi, Ph.D.
University of Nebraska-Lincoln, 2004

Adviser: Hong Jiang

Distributed heterogeneous systems and large clusters provide a great opportunity and, at the same time, pose a great challenge for high performance parallel and distributed applications. While these systems posses a vast reservoir of resources by virtue of scales and diversity, it is this very diversity and heterogeneity (in architectures, configurations and operating environments) that can severely restrict the efficient and simultaneous utilization of the available resources. In this dissertation, we investigate a variety of issues in this area and provide a framework for a multi-layered middleware approach, called **Delmon**, to facilitate efficient utilization of the rich resources existing in such heterogeneous and distributed environments. Middleware solutions provide the missing links between the vast resources and the application domain in a way that simplifies development, provides robust and reliable access to resources, and helps optimize resource utilization.

My research work on Delmon, a distributed systems middleware framework, comprises several stages: (1) Studying the current parallel programming models (which include the message-passing model, the distributed shared memory/object model,

distributed/parallel multithreading, and seamless parallelization) and identifying the general requirements of each model. (2) Separating the programming model requirements from the general run-time support requirements and identifying the general middleware functions necessary to support heterogeneous systems. (3) Based on the first two steps, designing the general framework for the three-layer Delmon (which includes a layer for parallel/distributed tools and programming models, a layer for the run-time environment, and a layer for the resource-dependant services). (4) Designing the self-organized agent-based run-time environment. (5) Designing the object-passing model and implementing the prototype in Java (the Java Object-Passing Interface (JOPI)) then experimentally evaluating its performance. (6) Extending an analytical performance model to evaluate the performance of parallel applications on heterogeneous systems.

Delmon provides a well-defined software architecture for developing parallel and distributed tools, programming models, and applications. It provides the critical link between the vast resources and the application domain, while reducing the complexity of the middleware itself. Delmon simplifies development, provides robust and uniform access to resources, helps optimize resource utilization, and facilitates the generation of stable distributed software. From a software engineering point-of-view, such a layered middleware approach and the separation of concerns improve the development and management of parallel and distributed tools and programming models in many ways. The quantitative and qualitative evaluation of Delmon using analytical and experimental methods demonstrates the benefits and good performance of the framework and its prototype implementation.

This is for my grandmother, Zainab, who passed away before seeing it to completion. May GOD bless her with his mercy.

Acknowledgements

First and most importantly, I would like to extend my warmest thanks and deepest appreciations to the one person who made all of this possible. He is my colleague throughout my undergraduate and graduate studies, class mate, research partner, best friend, husband, the father of my children, and soon-to-be colleague in my post-Ph.D. career, Nader Mohamed. Without Nader's encouragement, love and tremendous support in school and at home, I would have never made it this far. So Nader, Thank you, and thank GOD for you, "you are truly the wind beneath my wings". I would like to also thank my sons Hashim and Qassim, who have been the sunshine in my life and the driving force that propels me forward. Their innocence, energy, and enthusiasm have been a source of my inspiration and empowerment for me to be as active and energetic as they are and to be successful so that I would become their role model. In addition, I would like to thank my wonderful and loving father Mohamed Ali Al-Jaroodi, my great uncles Dr. Hassan Al-Jaroodi and Saeed Al-Jaroodi, and all my family and extended family members so much for believing in me and encouraging me to pursue my dreams.

My research advisor, Dr. Hong Jiang, deserves the profound appreciation and extreme gratitude. Over the past four years, He was my academic advisor, my research advisor and colleague, and my friend. His unlimited encouragement, valuable advice and guidance, and his wisdom allowed me to focus my research in the right direction, make successful strides towards a successful completion of my degree, and embark on a rewarding research experience. In addition, I would like to thank Dr. David Swanson for

his constant involvement in this research, boundless support, and indispensable guidance and assistance. I would also like to extend my sincere gratefulness for the readers of my dissertation, Dr. Steve Goddard and Dr. Sharad Seth, for their thorough reviews of the manuscript and their thoughtful and constructive advices and comments. I really appreciate all their help and support. Further more, I would like to thank Dr. Steve Scott and Dr. Xiao Cheng Zeng, who agreed to serve as members of the supervisory committee and were very generous with their support and encouragement.

For their financial support, I would like to extend my gratitude first to the University of Bahrain for the full study scholarship they provided to earn my doctoral degree. I would also like to thank my advisor for arranging to support my research through the Secure Distributed infrastructure grant (an NSF (National Science Foundation) supported project) and a UNL academic priority grant (PRISM). I am very grateful for all these organizations for their support. In addition I would like to extend my sincere appreciation to Donna McCarthy for her invaluable support and guidance not just as the secretary of SDI, but also as a true and wonderful friend.

More thanks are due to the Department of Computer Science and Engineering (CSE) and its chairman, Prof. Richard Sincovec, the research computing facility (RCF) and its members, and the SDI group members. They all were wonderful supporters and a great help throughout this venture. And finally I would also like to extend my thanks to the different departments at UNL such as the graduate school, admissions, international affairs, and many others for all the services they make available for students like me to pursue their educational goals and earn their degrees.

My heartfelt gratefulness to every one and every organization that participated in making this dream come true and I apologize if I failed to mention anyone deserving of my thanks.

Thank you,

Jameela

Table of Contents

List of Figures

List of Tables

List of Publications

1. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "JOPI: A Java Object-Passing Interface," accepted in Concurrency and Computation: Practice and Experience – special issue of 2002 Java Grande/ISCOPE Conference, Editor: Geoffrey Fox. 2004.

2. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "Middleware Infrastructure for Parallel and Distributed Programming Models on Heterogeneous Systems," in IEEE Transactions on Parallel and Distributed Systems – Special Issue on Middleware, Guest Editors: R. Guerraoui and W. Zwaenepoel, vol. 14, no. 11, pp. 1100-1111, November 2003.

3. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "An Overview of Parallel and Distributed Java for Heterogeneous Systems: Approaches and Open Issues," accepted in Parallel and Distributed Computing Practices - Special Issue on Algorithms, Systems and Tools for High Performance Computing on Heterogeneous Networks, Guest Editors: Frédéric Desprez, Eric Fleury, Alexey Lastovetsky, and Alexey Kalinov, 2003

4. J. Al-Jaroodi, N. Mohamed, and H. Jiang, " Distributed Systems Middleware Architecture from a Software Engineering Perspective," in proc. IEEE International Conference on Information Reuse and Integration (IRI'03), Session on Software Stability, Las Vegas, Nevada, October 2003.

5. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "Modeling Parallel Applications Performance On Heterogeneous Systems," in proc. IPDPS 2003,

Workshop on Advances in Parallel and Distributed Computational Models, Nice France, April 2003.

6. N. Mohamed, J. Al-Jaroodi, H. Jiang, and D. Swanson, "JOPI: A Java Object-Passing Interface," in proc. ACM Java Grande-ISCOPE (International Symposium on Computing in Object-Oriented Parallel Environments) Conference (JGI2002), Seattle, Washington, November 2002 (Acceptance rate 32%).

7. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "An Agent-Based Infrastructure for Parallel Java on Heterogeneous Clusters," in proc. 4th IEEE International Conference on Cluster Computing (CLUSTER2002), Chicago, Illinois, September 2002 (1/3 of submitted papers accepted).

8. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "Agent-Based Parallel Java for Heterogeneous Systems", in proc. 15th International Conference on Parallel and Distributed Computing Systems (PDCS2002), Louisville, Kentucky, September 2002.

9. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "A Comparative Study of Parallel and Distributed Java Projects for Heterogeneous Systems", in proc. IPDPS 2002, 4th International Workshop on Java for Parallel and Distributed Computing, Ft Lauderdale, Florida, April 2002.

10. J. Al-Jaroodi and N. Mohamed, "JOPI: Java Object-Passing Interface: Administrator's Guide," Technical Report TR04-05-01, May 2004, Computer Science and Engineering, University of Nebraska-Lincoln

11. J. Al-Jaroodi and N. Mohamed, "JOPI: Java Object-Passing Interface: User's Guide," Technical Report TR04-05-02, May 2004, Computer Science and Engineering, University of Nebraska-Lincoln

12. H. Jiang, D. Swanson, N. Mohamed, and J. Al-Jaroodi. "High Performance Distributed Java," poster, Diversity of Informatics EPSCoR Conference, April 4-5, 2002, Lincoln, Nebraska.

13. H. Jiang, D. Swanson, J. Al-Jaroodi, N Mohamed, "Agent-Based High Performance Computing for Heterogeneous Systems," poster, Workshop on Secure Distributed Information and Bioinformatics Research Laboratories, University of Nebraska-Lincoln, Nebraska, November 2001.

14. J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "Agent-Based Parallel Computing in Java - Proof of Concept," Technical Report TR-UNL-CSE-2001-1004, 2001, Computer Science and Engineering, University of Nebraska-Lincoln.

Chapter 1 Introduction

The ever-increasing need for high performance computing has led to great advances in the area of distributed and parallel computing. Currently, with the advancement of technology and high performance processors becoming a common commodity, clusters [16], the Grid [45], and networked heterogeneous systems are becoming capable of providing processing powers comparable to or exceeding those provided by special purpose multi-processor systems for a fraction of the cost. Thus, it is essential to have software that can exploit such systems by means of transparent and efficient mechanisms to utilize the multiple resources available. Such a co-existence of various hardware and software components that are in many cases inherently incompatible with one another imposes a problem to the applications trying to utilize them. For example, an application written in C for parallel execution on a shared memory multi-processor cannot be executed on a distributed cluster directly. Moreover, many systems and application environments have their own unique features that require different access methods.

On the other hand, many applications currently require tremendous amounts of processing power, memory, storage space and other resources. Examples of computer- and resource-intensive applications include scientific applications such as the n-body and linear algebra problems; data-mining and information-retrieval applications where many components and databases exist on heterogeneous systems and require intensive

processing to get the desired results; cryptanalysis algorithms; computer simulations and visualization problems; and GIS (Geographical Information Systems) applications. All these applications and many more could benefit greatly from the existence of an infrastructure that supports these applications across multiple heterogeneous platforms.

Heterogeneity, physical distribution and scale of such systems pose a great challenge to the systems developers and significantly complicate the design of suitable and efficient tools and programming models. Currently, there is no single software that provides all the necessary services and is mature enough to support systems of such magnitude, in terms of management, development, and deployment. Consequently, the vast resources in large clusters, heterogeneous systems, and even the Grid are not efficiently utilized due to the high costs and complexity of developing efficient tools and applications that can fully utilize them. However, to facilitate such features, many tools, services and programming models have been introduced. For example, MPI [113] has become the most common parallel programming tool for clusters and has also been expanded to handle parallel programming on the Grid (MPICH-G) [47] and other environments. In addition, many tools and services are in use and/or under investigation to facilitate system setup, configuration, management, maintenance, and control. Other tools for user application management and scheduling have also been devised to enhance system usability and fairness. At the cluster level, there is also a lot of research underway to find ways to simplify the use and management of clusters and to provide a single system image for the users. Most of this research spans multiple areas such as distributed operating systems, software engineering, parallel and distributed computing, networking, and middleware.

1.1 Basic Concepts

In the context of this dissertation, several terms will be used that may otherwise have different meanings depending on the context they are used in. Therefore, before proceeding further, we will discuss the general meanings of these terms as we use them in the context of this dissertation (we may also use some additional terminology according to their general known definition).

Cluster: A cluster in this context refers to Beowulf clusters, which consist of multiple nodes (with mostly one or two processor per node) closely connected by a switched network such as Ethernet or Myrinet [86]. Clusters provide high computing power in a distributed memory architecture. In general, most clusters have fully connected nodes and a single point of access from the outside world (usually through the *head node*).

Heterogeneous Cluster: The word "heterogeneous" means "of varying (dissimilar) characteristics". A heterogeneous cluster is a collection of high performance machines connected via some type of networks such as LANs (Local Area Networks) or WANs (Wide Area Networks). The machines involved may be of similar or different architectures. For example, a distributed system of multiple clusters, a few multi-processor machines, and a number of high performance servers can be connected to form a heterogeneous cluster. Another example is a cluster of clusters where multiple clusters (of similar architecture) are connected together, forming a larger cluster.

Heterogeneous Systems: In some contexts, researchers refer to systems that contain similar machines using the same operating environment as heterogeneous if the load on the machines or the memory utilization varies from one machine to another. For example,

a network of workstations (NOW) supporting multiple users and applications is considered heterogeneous. However, others consider this NOW as homogeneous in terms of machine capabilities and operating environment. In this dissertation, we have a broad definition for heterogeneity, which refers to machines that not just vary in load and utilization, but also in architecture, number of processors per machine or node, operating environment, and communication infrastructure (e.g., heterogeneous clusters). One example of such structure is the collection of machines and resources available on a university campus, where there are many LANs, clusters, shared-memory multi-processor machines, application servers, etc.

Parallel and Distributed Applications: Parallel and distributed applications require the utilization of multiple processors (or multiple machines). However, there are some qualitative and quantitative differences between parallel and distributed applications. The former aims at high performance computing by parallelizing compute- and communication-intensive jobs and concurrently executing multiple tasks within each job to strive for linear speedup. Thus, parallel applications are best suited for multi-processor machines or tightly coupled distributed systems such as clusters. The latter, on the other hand, aims at concurrently executing multiple jobs and relatively independent tasks within each job in a distributed environment. As a result, distributed applications are usually service-oriented that include, for example, client/server and web service applications, or task-oriented such as information collection, analysis and dissemination. Nevertheless, both have similar operational requirements, such as remote access, communication, and resource management to support concurrency.

Parallel Programming Models: A programming model defines the different primitives (APIs, Advance Programming Interfaces), or programming constructs, available for the application developer. In parallel programming, a programming model provides a set of APIs that allow the parallel application developer to define the parallelization techniques used and build the applications accordingly. An example of such models is the message-passing model, which provides APIs to exchange data and to organize and control the parallel processes in the applications.

Middleware: By definition, middleware is a layer between the applications and the system or operating environment. Middleware abstracts many of the system details into a well-defined set of APIs that simplifies the developer's task of writing applications for the system. In addition, middleware also provides abstracted communication mechanisms at different levels. For example, CORBA is an application-level middleware facilitating object-communication between independent applications, while Sockets represent a network-level middleware facilitating communications between tasks within a distributed or parallel application. Similarly, a message-passing model can also be considered a special type of middleware that abstracts the communication and coordination mechanisms for developers of parallel applications. In a broad manner, parallel and distributed programming models can be considered as specialized types of middleware.

Distributed Systems Middleware: This qualified middleware in this context relates to the layer (or layers) that handles services at the system level rather than the applications or communications levels. This middleware provides the necessary functionality to access and utilize the systems resources efficiently, while reducing the application development efforts. Since this middleware is intended for distributed systems