Deanship of Graduate Studies
Al-Quds University

P^2O^2CM: Portable Parallel Object Oriented Computing Model

Hana Hirbawi

M.Sc Thesis

Jerusalem-Palestine

2004

P^2O^2CM: Portable Parallel Object Oriented Computing Model

By

Hana Abdel-Salam Saleh Hirbawi

B.Sc.: Al-Quds Open University
Jerusalem

Supervisor: Dr. Badie Sartawi

A Thesis Submitted in Partial Fulfillment of Requirement for the Degree of Master of
Computer Science

Computer Science Department
Al-Quds University

August, 2004

Department of Computer Science

College of Science and Technology

Deanship of Graduate Studies

P^2O^2CM: Portable Parallel Object Oriented Computing Model

By

Student Name: Hana Hirbawi

Registration Number: 20011311

Supervisor: Dr. Badie Sartawi

Co-Supervisor: -

Master thesis submitted and accepted, Date:

The names and signatures of the examining committee members are as follows:

1- Dr. Badie Sartawi Head of Committee

2- Dr. Rashid Jayousi Internal Examiner

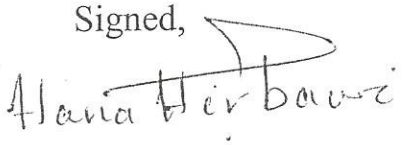
3- Dr. Khaled Khanfar External Examiner

Al-Quds University
2004

Declaration:

I certify that this thesis submitted for the degree of Master is the result of my own research, except where otherwise acknowledged, and that this thesis (or any part of the same) has not been submitted for a higher degree to any other university or institution.

Signed,



Hana Hirbawi

August 27, 2004

Acknowledgments

I would like to thank Dr. Badie Sartawi for all his invaluable assistance and guidance, without which, this work would never have come to fruition.

I would also like to thank Dr. Rashid Jayousi and Dr. Ghassan al-Qaimari for their assistance.

Finally I thank my family and friends especially my parents and my husband for their patient, support and encouragement.

Abstract

Distributed systems consisting of powerful workstations and high-speed interconnection networks are economical alternative to special purpose supercomputers. Networks of workstations that are comprised of independent, general purposes computers are available everywhere and have much unused computing power to be exploited. Utilizing these resources for parallel computing is becoming a good idea with the existence of many developed tools and systems that make these workstations appears as a single virtual machine. Heterogeneity, high latency communication, fault tolerance and dynamic load balancing are issues that should be addressed in these systems to exploit parallelism. Developing parallel programs in such systems requires expertise not only in parallel programming issues but also in distributed computing. We present a portable object-oriented parallel computing model based on Java and using JPVM message passing interface. We provide an implementation of the model in a library called P^2O^2CM. It provides a transparent way of parallelizing algorithms with load balancing to enhance resource utilization. P^2O^2CM liberates the users from the complexities of distributed systems and parallel programming, thus the programmer have only to learn the interface to the library instead of learning parallel programming. We have chosen graph algorithms to test our model and ran the prototype on Ethernet based network of Win2K computers and achieved appreciable speedup and efficiency. We also solved large instances of problems, which could not be solved on a single computer for reasons of speed or memory constraints.

ملخص البحث

تعتبرالانظمة الموزعة التي تتكون من أجهزة حاسوب متصلة مع بعضها بواسطة شبكات الحاسوب بديلا للحواسيب المصممة للبرمجة المتوازية. تتوفر شبكات الحاسوب التي تتكون من أجهزة الحاسوب المتصلة مع بعضها في جميع المؤسسات التعليمية وغيرها. استغلال هذه الموارد في البرمجة المتوازية يساعد في تنفيذ البرامج بوقت أقل، بالاضافة الى امكانية تنفيذ البرامج التي تحتاج الى ذاكرة ذات سعة عالية، خاصة مع توفر الادوات التي تساعد في الاستفادة من هذه الموارد. استخدام هذه الموارد في البرمجة المتوازية يحتاج الى الالمام بالبرمجة المتوازية بالاضافة الى معرفة كيفية برمجة الانظمة الموزعة من حيث الاختلاف في تمثيل البيانات في الحواسيب ذات الهيكليات المختلفة وتوزيع العمل على الحواسيب المتصلة بشكل يضمن استغلالها في البرمجة المتوازية. نقدم في هذا البحث نموذج حوسبة متوازية يتضمن تطوير اجراءات مبرمجة باستخدام لغة البرمجة جافا والتي تساعد المبرمجين في كتابة برامج يتم تنفيذها على شبكات الحاسوب بشكل متوازي. تناول البحث ايضا تنفيذ للنموذج على شكل مكتبة من الاجراءات المبرمجة. تزود هذه المكتبة للمبرمج طريقة للبرمجة المتوازية بدون معرفة سابقة بطرق البرمجة المتوازية أو المعرفة بخصائص أو برمجة الأنظمة الموزعة. الاجراءات التي توفرها المكتبة يمكن تنفيذها على هيكليات حواسيب مختلفة بدون اعادة ترجمتها. يتطلب النموذج أن يتعلم المبرمج المستخدم للاجراءات المتوفرة في المكتبة كيفية استخدام المكتبة في كتابة البرامج بدلاً من تعلم البرمجة المتوازية باستخدام الانظمة الموزع ة. لقد تمكنا من تنفيذ بعض البرامج باستخدام المكتبة على شبكة من الحواسيب المتصلة والتي تعمل بنظام التشغيل ويندوز وحصلنا على نتائج أسرع مقارنة مع تنفيذ نفس البرامج على جهاز حاسوب واحد، بالا ضافة الى تنفيذ برامج لم نتمكن من تنفيذها على جهاز واحد بسبب سعة الذاكرة.

TABLE OF CONTENTS

CHAPTER 1 1

INTRODUCTION 1

1.1 Research Objectives 4
1.1.1 Transparent Parallelization 4
1.1.2 Resource Utilization 5
1.1.3 Code reusability 6
1.1.4 Extensibility 7
1.1.5 Code portability 8

1.2 Motivations 8

CHAPTER 2 12

BACKGROUND AND LITERATURE REVIEW 12

2.1 Basic Concepts of Parallelism 12

2.2 Terminology of Parallel Software 15
2.2.1 Message Passing 17
2.2.2 Transfers Through Shared Memory 17
2.2.3 Direct Remote-Memory Access 18

2.3 Distributed and Parallel Computing 19

2.4 Heterogeneous Parallel Computing 21

2.5 Networked Computers as a Multicomputer Platform 22

2.6 The Object Oriented Programming Paradigm 24
2.6.1 Classes 24
2.6.2 Inheritance 25

2.7 Graph Algorithms 25
2.7.1 Depth First Search and Bread First Search Algorithms 27
2.7.2 Minimum Spanning Tree: *Prims Algorithm* 28
2.7.3 Shortest Paths: *Dijkstra Algorithm* 29
2.7.4 Connected Components 30

CHAPTER 3 32

MODELS OF PARALLEL COMPUTING 32

3.1 Parallel Model Computing Properties 34

3.2 Overview of Parallel Programming Models 37

3.3 Message Passing Parallel Computing 39

3.4 Overview of Message Passing Models 44
3.4.1 Message Passing Interface MPI 44
3.4.2 Parallel Virtual Machine (PVM) 45
3.4.3 Java Parallel Virtual Machine (JPVM) 49

3.5 Previous Work on Parallel Libraries 51
3.5.1 PPBB-Lib 52
3.5.2 ZRAM 52
3.5.3 ParSA 53

CHAPTER 4 58

DESIGN AND REQUIREMENTS OF GENERAL PURPOSE PARALLEL COMPUTING MODEL 58

4.1 Requirements of Parallel Computing Model 58
4.1.1 Reusability 58
4.1.2 Efficiency 59
4.1.3 Portability 60

4.2 Object Oriented Design of The Library 61
4.2.1 Extensibility 63
4.2.2 Reusability 64

4.3 Architecture of The Library 65
4.3.1 Message Passing Interface 65
4.3.2 Kernel Services 66
4.3.3 Virtual Machine 68
4.3.4 Load Balancing Function 70
4.3.5 Master and Worker Services 71
4.3.6 Graph Algorithm Layer 72
4.3.7 Application Interface 73

CHAPTER 5 76

IMPLEMENTATION 76

5.1 Implementation Decisions 76

5.2 System Architecture 79

5.3 Implementation of The Core Data Structure 81
5.3.1 Graph Class 81
5.3.2 Queue Class 81
5.3.3 List Class 83

5.4 Implementation of The General Services 83
5.4.1 Master Class 83
5.4.2 Worker Class 85
5.4.3 Dynamic Balancer Class 86
5.4.4 Kernel Class 89
5.4.5 Shortestpath Class 90
5.4.6 Termination Detection Implementation 91

5.5 Limitations of the Library Implementations 91

CHAPTER 6 93

PERFORMANCE EVALUATION 93

CHAPTER 7 103

CONCLUSION AND FUTURE WORK 103

7.1 Summery Of Work 103

7.2 Future Enhancements 105

7.3 Future Research Directions 105

REFERENCES 107

APPENDIX A 109

THE LIBRARY PACKAGE 109

APPENDIX B 117

TABLES OF PERFORMANCE RESULTS 117

INDEX OF FIGURES

Figure 2.1: Distributed memory MIMD and shared memory MIMD...14

Figure 3.1: Master-slave "worker" structure.....42

Figure 3.2: Communications in PVM.....46

Figure 4.1: P^2O^2CM architecture.....67

Figure 4.2: The structure of P^2O^2CM library.....75

Figure 5.1: Full P^2O^2CM package structure with sub-packages.....82

Figure 5.2: Distributed work pool for dynamic load balancing.....88

Figure 6.1: Partitioning the adjacency matrix A among processors.....94

Figure 6.2: Execution time for graph size ranges from 10 to 3000.....96

Figure 6.3: Execution time for graph size ranges from 10 to 800.....97

Figure 6.4: Observed speed up on 1,2,4, and 8 workstations.....98

Figure 6.5: Observed speed up for 1,4,8,16 workstations.....98

Figure 6.6: Observed speed up for 1,4,8,16 processors.....99

Figure 6.7: Observed efficiency for graph size from 10 to 2000.....102

Figure A1: The Kernel class UML.....109

Figure A2: The Dynamic Balancer UML.....110

Figure A3: The Graph class UML.....111

Figure A4: The Queue class UML.....112

Figure A5: The shortest Path class UML.....113

Figure A6: The List class UML.....114

Figure A7: The Worker class UML.....115

Figure A8: The Master class UML.....116

INDEX OF TABLES

Table 1: Execution time for graph size ranges from 10 t0 5000.....118

Table 2: Execution time on 1, 2, 4, 8 workstations.....118

Table 3: Execution time on 1, 2, 4, 8, 16 workstations.....119

Table 4: Speed up on 1, 2, 4, 8, 16 workstations.....120

Table 5: Efficiency on 1,2,4,8,16 workstations.....120

Chapter 1

Introduction

Parallel computing is about 20 years old, since then parallel computing has solved complex problems and high-performance applications in traditional areas in science, engineering and application areas such as artificial intelligence and finance. However, special-purpose massively parallel machines[1] have proven to be expensive to build, difficult to use, and could not take advantage of improving technologies by replacing higher speed processors and interconnection networks.

Distributed systems consisting of powerful workstations and high-speed interconnection networks are economical alternative to special purpose supercomputers. On the other hand, the availability of powerful desktop computers and high-speed connectivity has placed a potential distributed parallel processing environments. Network of workstations that are compromised of independent, general purpose computers are available everywhere and have much unused computing power to be exploited.

---

[1]The definition of a massively parallel computer is generally used to describe a SIMD machine with over 1000 processors. These machines are fine grained and involve very high-speed communications. By contrast, the networked workstation paradigm being used for the PVM would not be a massively parallel model.

Utilizing these resources for parallel computing is becoming a good idea with the existence of many developed tools that makes these workstations appear as a single virtual machine. Heterogeneity, high latency communication, fault tolerance and load balancing are issues that should be addressed in these systems to exploit parallelism. Developing efficient parallel programs to be executed on such system requires expertise not only in parallel programming but also in distributed computing, fault tolerance and load balancing techniques. Writing libraries that provide support for these issues liberate users from the complexities of distributed systems, thus the programmers have only to learn the interface to these libraries instead of learning parallel and distributed programming.

Architecture-independent message passing tools have been developed to allow transparent use of networks of workstations for parallel computing. Models for message passing include systems such as MPI, PVM, JPVM, javaPVM, Parmacs and P4 (Sundarem and Geist 1999). These systems connect a heterogeneous collection of computers and making them appear like a large distributed memory computer.

Using object oriented languages, especially Java language is gaining growing interest for implementing complex science and engineering applications. Due to the excellent support offered by Java (Horstmann

and Cornell 2000) for programming on distributed platforms, Java is considered an excellent language for developing complex applications to be executed in parallel on a network of workstations especially after the development of JIT compiler (Ishizaki et al. 2003).

Despite these successes, parallel computing has not become a major methodology in computer science. We have chosen to examine the state of parallel computing on network of workstations by implementing a portable object oriented parallel computing model using Java and JPVM to create a virtual machine for parallel computing on which the problem is broken into smaller units of work and distributed throughout the processing nodes of the virtual machine with dynamic load balancing that increases processor utilization. Our model called P^2O^2CM is easy to program, has a software development methodology based on client server paradigm, architecture independent and guarantee performance. We have implemented several graph algorithms using the model. Doing so addresses both software and development issues concerning these areas, hardware and performance issues.

Graphs can be used to model a wide variety of structures and relationships. Most applications of graphs can be reduced to standard graph properties and using well-known algorithms. These include depth