AIN SHAMS UNIVERSITY
FACULTY OF COMPUTER & INFORMATION SCIENCES
DEPARTMENT OF SCIENTIFIC COMPUTING

Deployment of Distributed Applications in a High Performance Computer Environment

A Thesis Submitted to the Department of Scientific Computing, Faculty of Computer and Information Sciences, Ain Shams University, in Partial Fulfilment of the Requirements for the Doctorate Degree of Computer and Information Sciences

By

Amal Said Khalifa

M.Sc. Degree in Scientific Computing (2004).
Lecturer Assistant, Department of Scientific Computing,
Faculty of Computer and Information Sciences, Ain Shams University.

Supervised by

Prof. Dr. Mohammed Fahmy Tolba
Professor, Department of Scientific Computing,
Faculty of Computer and Information Sciences,
Ain Shams University

Prof. Dr. Reda Ammar
Head of Computer Science and Engineering Department,
University of Connecticut – USA

Prof. Dr. Mohammed Essam Khalifa
Dean of Faculty of Computer and Information Sciences,
Ain Shams University

March, 2009

ACKNOWLEDGMENTS

I am deeply indebted to, Prof. Reda Ammar, for his endless help, and support during our two years stay in USA. His advice and insight have been invaluable throughout my entire time of research. Our weekly meetings gave me academic guidance and provided the milestones that helped to develop my research skill.

I want to express my deep gratitude to Prof. Dr. Mohammed Fahmy Tolba for his emotional and academic support either in Egypt or in USA. His continuous contact with me and my external supervisor gave me a really strong push toward achieving my research goals.

I would like to extend my thanks to Essam Khalifa for his continuous encouragement, understanding and support. He stood by me throughout all the difficulties I faced during the last few years. I won't forget to also thank Prof. Dr. Tahany Fergany, who gave me a lot of her time and effort. I cannot possibly thank her enough.

Finally I am really grateful to my husband for his deep caring and very strong encouragement that helped me in my "Mission Impossible" especially with three kids at home. Furthermore, this dissertation would have never been possible without the support of my family. I would like to thank my mother and my sister "Amira" for taking care of my kids during the very busy and difficult time of work finalization. In addition, I am deeply grateful for my father. When I was a kid, he didn't hesitate to answer any of my extensive questions about the world around me. He encouraged me to read, learn and express myself. Thank you for your genes, Dad!!

Abstract

Heterogeneous computing (HC) environment is the coordinated use of different types of machines, networks, and interfaces to maximize the ability to solve computationally intensive problems. Usually theses applications consist of various components that have different computational requirements. As machine architectures become more advanced, the extent to which a given task can exploit a given architectural feature depends on how well the task's computational requirements match the machine's advanced capabilities. This variation in tasks needs as well as machine capabilities has created a very strong need for developing Mapping/scheduling techniques especially for the HC community. In fact, the applicability and strength of HC systems are derived from their ability to match computing needs to appropriate resources.

The mapping problem can be stated shortly as: deciding on which task should be moved to where and when, to optimize some system performance criteria. Mapping problems are known to be NP-Complete except under a few special situations. The existing heuristics for mapping tasks in HC systems works either statically or dynamically. This distinction is based on the time at which the mapping decisions are made. In contrast to static techniques where the complete set of tasks to be mapped is known a priori and the mapping is done off-line, in dynamic mapping methods the machine allocation process is done at run time. Although the principal advantage of the static mapping is its simplicity, it fails to adjust to changes in the system state. A dynamic scheme is needed because the arrival times of the tasks may be random and some machines in the suite may go off-line and new machines may come on-line.

This research proposes, describes, compares, and discusses a number of mapping algorithms that can be used for dynamically assigning tasks to machines in a general HC system. In the HC environment considered here, the tasks are assumed to be independent, i.e., no communications between the tasks are needed. This scenario is likely to be present, for instance, when many independent users submit their jobs to a collection of shared computational resources. Furthermore, some of the algorithms

investigated in this study are preemptive, and assume that the tasks have no deadlines or priorities associated with them.

In general, dynamic mapping heuristics can be grouped into two categories: immediate mode and batch mode. The algorithms investigated in this study cover these two categories. Actually, in the immediate mode, a task is mapped onto a machine as soon as it arrives at the mapper. However, in batch mode, tasks are not mapped onto the machines as they arrive; instead they are collected into a set that is examined for mapping at prescheduled times called mapping events. The independent set of tasks that is considered for mapping at the mapping events is called a meta-task. While immediate mode heuristics consider a task for mapping only once, batch mode heuristics consider a task for mapping at each mapping event until the task begins execution.

Six new heuristics, four for batch mode and two for immediate mode, are introduced as part of this research. For the heuristics discussed here, maximization of throughput is the primary objective, because this performance measure is the most common one in production oriented environments. To achieve this objective, the batch mode heuristics considered task migration as well as resource utilization. On the other hand, the immediate mode heuristics considered, to varying degrees and in different ways, improving performance by applying the concept of load balancing.

Simulation studies were performed to compare the performance of these heuristics with some existing ones. In total, six batch mode heuristics and three immediate mode heuristics are examined. The extensive experiments were carried on under a variety of system heterogeneity modes and different task arrival rates. Furthermore, the trade-offs among and between immediate mode and batch mode heuristics was studied experimentally. The experimental results helped to reveal which algorithm to use in a given heterogeneous environment.

Table of Contents

Abstract ..... *ii*

Table of Contents ..... *iv*

List of Figures ..... *viii*

List of Tables ..... *xii*

Chapter 1: Introduction *1*

1.1 Motivation and problem Statement **2**

1.2 Research Objectives **3**

1.3 Thesis Overview **3**

Chapter 2: Background Overview *5*

2.1 Distributed Systems **6**

2.2 Heterogeneous Computing Systems **7**

2.3 Resource Management Systems **10**

2.4 Introduction to the Mapping Problem **12**

2.5 Modelling the Mapping Problem **14**

2.5.1 Modelling Processors: 15

2.5.2 Modelling Processes: 15

2.5.3. Modelling Task Execution Time: 15

2.5.4. Modelling Cost: 17

2.6 A Taxonomy of Mapping Approaches **18**

Chapter 3: Literature Survey *21*

3.1 Introduction **22**

3.2 Static Mapping **23**

3.3 Dynamic Mapping **26**

3.3.1 Immediate Mode Techniques: 27

3.3.2 Batch Mode Techniques: 28

3.4 Mapping Multiple DAGs **31**

Chapter 4: Dynamic Batch-mode Deployment Algorithms *35*

4.1 Overview **36**

4.2 Assumptions and Notations **37**

4.3 The *Mig*Min-min algorithm **39**

4.3.1 Overview 39

4.3.2 The Mapping Strategy 40

4.3.3 Complexity Analysis 43

4.4 The *Utl*Min-min algorithm **43**

4.4.1 Overview 43

4.4.2 The Mapping Strategy 44

4.4.3 Complexity Analysis 48

4.5 The *Mig*Max-min algorithm **49**

4.5.1 Overview 49

4.5.2 The Mapping Strategy 50

4.6 The *Utl*Max-min algorithm **53**

Chapter 5: Immediate-mode Load Balancing Algorithms *56*

5.1 An Overview **57**

5.2 The *MLB* algorithm **58**

5.2.1 Overview 58

5.2.2 The Mapping Strategy 59

5.2.3 Complexity Analysis 62

5.3 The *LLI* algorithm **62**

5.3.1 Overview 62

5.3.2 The Mapping Strategy 62

5.4 A Case Study **65**

Chapter 6: The Simulation Model ***70***

6.1 Introduction **71**

6.2 Simulating Task Arrivals **72**

6.3 Modelling Classes of Heterogeneity **74**

6.4 Maintaining System State **80**

6.5 Managing Simulation Time **83**

Chapter 7: Performance Evaluation & Analysis ***88***

7.1 Introduction **89**

7.2 The Min-mins Comparison **90**

7.2.1 Inconsistent Environment 91

7.2.2 Semi-Consistent Environments 95

7.2.3 Consistent Environments 100

7.3 The Max-mins Comparison **103**

7.3.1 Inconsistent Environments 103

7.3.2 Semi-consistent Environments 106

7.3.3 Consistent Environments 108

7.4 Immediate Mode Comparisons 110
7.4.1 Inconsistent Environments 111
7.4.3 Semi-consistent Environments 113
7.4.3 Consistent Environments 114
7.5 An Overall Comparative Study 116
7.5.1 Inconsistent Environments 116
7.5.2 Semi-consistent Environments 118
7.5.3 Consistent Environments 119
7.6 Summary and Remarks 121

Conclusions 124
Future Work 126
Bibliography 127

List of Figures

Figure 2.1: Architecture of a distributed syste **7**
Figure 2.2: A conceptual model for automatic assignment of tasks to machines in a HC system environment **9**
Figure 2.3: High-level functional architecture of the of MSHN. **12**
Figure 2.4: Mapping versus Scheduling in a distributed environment **13**
Figure 2.5: Execution time possibilities **16**
Figure 2.6: A hierarchical classification of mapping techniques **19**
Figure 3.1: The process of executing a parallel / distributed application **22**
Figure 3.2: An Example of task DAG **31**
Figure 4.1: The *Mig*Min-min Algorithm **42**
Figure 4.2: The UtlMin-min Algorithm **48**
Figure 4.3: The *Mig*Max-min Algorithm **52**
Figure 4.4: The *Utl*Max-min Algorithm **55**
Figure 5.1: The main steps of the *MLB* algorithm **61**
Figure 5.2: The main steps of the *LLI* algorithm **64**
Figure 5.3: Mapping the example tasks using the MCT heuristic **66**
Figure 5.4: Mapping the example tasks using the MET heuristic **67**
Figure 5.5: Mapping the example tasks using the OLB heuristic **67**
Figure 5.6: Mapping the example tasks using the MLB heuristic **68**
Figure 5.7: Mapping the example tasks using the *LLI* heuristic **69**
Figure 6.1: An algorithm for generating Poisson random arrival times with mean λ **73**
Figure 6.2: An example of a randomly generated ETC matrix representing an *inconsistent* LoLo HC environment **77**
Figure 6.3: An example of a randomly generated ETC matrix representing a *semi-consistent* LoLo HC environment **77**
Figure 6.4: An example of a randomly generated ETC matrix representing a *consistent* LoLo HC environment **77**
Figure 6.5: An algorithm for randomly generating different classes of the ETC matrix **79**
Figure 6.6: A proposed data structure to maintain state of all machines in a simulated HC suite **80**
Figure 6.7: An example of a machine's state data during the

simulation **82**
Figure 6.8: How the Simulator composes a metatask each time the clock triggers a mapping event in a batch-mode mapping **85**
Figure 6.9: Mapping the arriving tasks in an immediate mode **86**
Figure 7.1: Comparisons of the normalized *Makespan* of the Min-min algorithms for 500 independent tasks arriving at λ_l to Inconsistent HC systems **92**
Figure 7.2: Comparisons of the *System Utilization* of the Min-min algorithms for 500 independent tasks arriving at λ_l to Inconsistent HC systems **93**
Figure 7.3: Comparisons of the normalized *Makespan* of the Min-mins algorithms for 500 independent tasks arriving at λ_h to Inconsistent HC systems **94**
Figure 7.4: Comparisons of the System Utilization of the Min-min algorithms for 500 independent tasks arriving at λh to Inconsistent HC systems **95**
Figure 7.5: Comparisons of the normalized Makespan of the Min-mins algorithms for 500 independent tasks arriving at λ_l to Semi-consistent HC systems **97**
Figure 7.6: Comparisons of the *System Utilization* of the Min-min algorithms for 500 independent tasks arriving at λ_l to Semi-consistent HC systems **97**
Figure 7.7: Comparisons of the normalized *Makespan* of the Min-mins algorithms for 500 independent tasks arriving at λ_h to Semi-consistent HC systems **99**
Figure 7.8: Comparisons of the *System Utilization* of the Min-min algorithms for 500 independent tasks arriving at λ_h to Semi-consistent HC systems **99**
Figure 7.9: Comparisons of the normalized *Makespan* of the Min-min algorithms for 500 independent tasks arriving at λ_l to Consistent HC systems **101**
Figure 7.10: Comparisons of the *System Utilization* of the Min-min algorithms for 500 independent tasks arriving at λ_l to Consistent HC systems **101**
Figure 7.11: Comparisons of the normalized *Makespan* of the Min-min algorithms for 500 independent tasks arriving at λ_h to Consistent HC systems **102**

Figure 7.12: Comparisons of the *System Utilization* of the Min-min algorithms for 500 independent tasks arriving at λ_h to Consistent HC systems **102**
Figure 7.13: Comparisons of the normalized *Makespan* of the Max-min algorithms for 500 independent tasks arriving at λ_l to Inconsistent HC systems **104**
Figure 7.14: Comparisons of the *System Utilization* of the Max-min algorithms for 500 independent tasks arriving at λ_l to Inconsistent HC systems **105**
Figure 7.15: Comparisons of the normalized *Makespan* of the Max-min algorithms for 500 independent tasks arriving at λ_h to Inconsistent HC systems **105**
Figure 7.16: Comparisons of the *System Utilization* of the Max-min algorithms for 500 independent tasks arriving at λ_h to Inconsistent HC systems **106**
Figure 7.17: Comparisons of the normalized *Makespan* of the Max-min algorithms for 500 independent tasks arriving at λ_l to Semi-consistent HC systems **107**
Figure 7.18: Comparisons of the *System Utilization* of the Max-min algorithms for 500 independent tasks arriving at λ_l to Semi-consistent HC systems **108**
Figure 7.19: Comparisons of the normalized *Makespan* of the Max-min algorithms for 500 independent tasks arriving at λ_l to Consistent HC systems **109**
Figure 7.20: Comparisons of the *System Utilization* of the Max-min algorithms for 500 independent tasks arriving at λ_l to Consistent HC systems **110**
Figure 7.21: Comparisons of the normalized *Makespan* of the Immediate-mode algorithms for 500 independent tasks arriving at λ_l to Inconsistent HC systems **112**
Figure 7.22: Comparisons of the *System Balance* of the Immediate-mode algorithms for 500 independent arriving to Inconsistent HC systems **112**
Figure 7.23: Comparisons of the normalized *Makespan* of the Immediate-mode algorithms for 500 independent tasks arriving to Semi-consistent HC systems **113**
Figure 7.24: Comparisons of the *System Balance* of the Immediate-mode algorithms for 500 independent tasks arriving to Semi-consistent HC systems **114**

Figure 7.25: Comparisons of the normalized *Makespan* of the Immediate-mode algorithms for 500 independent tasks arriving to Consistent HC systems **115**
Figure 7.26: Comparisons of the *System Balance* of the Immediate-mode algorithms for 500 independent tasks arriving to Consistent HC systems **115**
Figure 7.27: Comparisons of the normalized *Makespan* of all of the proposed algorithms under different Inconsistent HC systems considering low task arrival **117**
Figure 7.28: Comparisons of the normalized *Makespan* of all of the proposed algorithms under different Inconsistent HC systems considering high task arrival **117**
Figure 7.29: Comparisons of the normalized *Makespan* of all of the proposed algorithms under different Semi-consistent HC systems considering low task arrival **118**
Figure 7.30: Comparisons of the normalized *Makespan* of all of the proposed algorithms under different Semi-consistent HC systems considering high task arrival **119**
Figure 7.31: Comparisons of the normalized *Makespan* of all of the proposed algorithms under different Consistent HC systems considering low task arrival **120**
Figure 7.32: Comparisons of the normalized *Makespan* of all of the proposed algorithms under different Consistent HC systems considering high task arrival **121**

List of Tables

Table 5.1: The expected ready times for the machine in the example HC system 65
Table 5.2: The expected Execution Time for the example tasks 65
Table 5.3: The expected times at which the example tasks will arrive during the simulation time 65
Table 5.4: A summary of the experimental results of mapping the example tasks using different algorithms 69
Table 7.1: Comparisons of the average number of migrations taken by the Min-min algorithms in Inconsistent HC environments considering low task arrival rate 93
Table 7.2: Comparisons of the average number of migrations taken by the Min-min algorithms in Inconsistent HC environments considering High task arrival rate 95
Table 7.3: Comparisons of the average number of migrations taken by the Min-min algorithms in Semi-consistent HC environments considering low task arrival rate 98
Table 7.4: Comparisons of the average number of migrations taken by the Min-min algorithms in Semi-consistent HC environments considering high task arrival rate 98
Table 7.5: Comparisons of the average number of migrations taken by the Min-min algorithms in Consistent HC environments considering low task arrival rate 100
Table 7.6: Comparisons of the average number of migrations taken by the Min-min algorithms in Consistent HC environments considering high task arrival rate 103
Table 7.7: Comparisons of the average number of migrations taken by the Max-min algorithms in Inconsistent HC environments considering low task arrival rate 104
Table 7.8: Comparisons of the average number of migrations taken by the Max-min algorithms in Inconsistent HC environments considering high task arrival rate 106
Table 7.9: Comparisons of the average number of migrations taken by the Max-min algorithms in Semi-consistent HC environments considering low task arrival rate 108

Table 7.10: Comparisons of the average number of migrations taken by the Max-min algorithms in Consistent HC environments considering low task arrival rate 110
Table 7.11: A check list showing which algorithm to use in different HC environments considering low arrivals 123
Table 7.12: A check list summary of which algorithm to use in different HC environments considering high arrivals 123