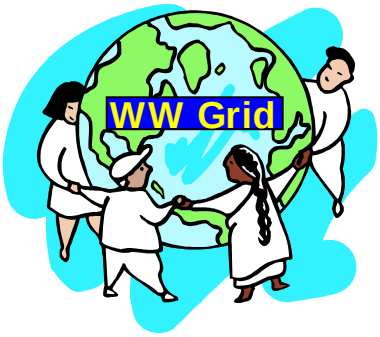
Economic-based Distributed Resource Management and Scheduling for Grid Computing



by

Rajkumar Buyya
B.E. (Mysore Univ.) and M.E. (Bangalore Univ.)
http://www.buyya.com

A thesis submitted in fulfillment of the requirements for the Degree of

Doctor of Philosophy

School of Computer Science and Software Engineering
Monash University, Melbourne, Australia

April 12, 2002

Table of Contents

Table of Figures ..... **vi**

Abstract ..... **x**

Declaration ..... **xi**

Acknowledgements ..... **xii**

Chapter 1 Introduction **1**

1.1 Inspiration for Computational Grids 1
1.2 Economic-based Grid Resource Management and Scheduling 2
1.2.1 Assessing Wants and Needs 3
1.2.2 The Nimrod-G Grid Resource Broker 3
1.2.3 GridSim Toolkit and Economic Grid Broker Simulator 5
1.3 Contributions 5
1.4 Organization 7
1.5 Acknowledgements 7

Chapter 2 Grid Technologies and Resource Management Systems **9**

2.1 Introduction 9
2.2 Major Technological Milestones: Enabling Grid and P2P Computing 10
2.3 Grid Computing Environments 12
2.3.1 Resource Management Challenges 12
2.3.2 Grid Components 13
2.3.3 Grid Computing Projects 14
2.4 Resource Management Systems Taxonomy 16
2.5 Mapping Taxonomy to Some Grid Resource Management Systems 17
2.5.1 AppLeS: A Network Enabled Scheduler 18
2.5.2 Condor: Cycle Stealing Technology for High Throughput Computing 18
2.5.3 Data Grid 19
2.5.4 Globus: A Toolkit for Grid Computing 19
2.5.5 Javelin 20
2.5.6 Legion: A Grid Operating System 20
2.5.7 MOL: Metacomputing Online Kernel 21
2.5.8 NetSolve: A Network Enabled Computational Kernel 22
2.5.9 Ninf: A Network Enabled Server 22
2.5.10 PUNCH: The Purdue University Network Computing Hubs 22
2.5.11 Nimrod-G Grid Resource Broker 22
2.6 Summary and Comments 23

Chapter 3 Service-Oriented Grid Architecture for Distributed Computational Economies **24**

3.1 Introduction 24
3.2 Computational Economy and its Benefits 26
3.3 Requirements for Economic-based Grid Systems 27
3.4 Grid Architecture for Computational Economy (GRACE) 28
3.4.1 Grid Resource Broker (GRB) 29
3.4.2 GRACE Framework—Leveraging Globus Tools 29
3.4.3 Grid Open Trading Protocols and Deal Template 31

3.4.4 Pricing, Accounting, and Payment Mechanisms ... 33
3.5 Related Work ... 35
3.6 Economic Models in the Context of GRACE Framework ... 38
3.6.1 Commodity Market (Flat or Supply-and-Demand Driven Pricing) Model ... 38
3.6.2 Posted Price Model ... 40
3.6.3 Bargaining Model ... 40
3.6.4 Tender/Contract-Net Model ... 41
3.6.5 Auction Model ... 42
3.6.6 Bid-based Proportional Resource Sharing Model ... 44
3.6.7 Cooperative Bartering Model ... 45
3.6.8 Monopoly/Oligopoly ... 45
3.6.9 Other Influences on Market Prices ... 46
3.7 Summary and Conclusion ... 46

Chapter 4 The Nimrod-G Grid Resource Broker and Economic Scheduling Algorithms ... 47

4.1 Introduction ... 47
4.2 The Nimrod-G Resource Broker: An Economic based Grid Scheduler ... 48
4.2.1 Objectives and goals ... 48
4.2.2 Services and End Users ... 49
4.2.3 Architecture ... 50
4.2.4 Nimrod-G Clients ... 51
4.2.5 The Nimrod-G Grid Resource Broker ... 52
4.3 Scheduling and Computational Economy ... 54
4.4 Scheduling Algorithms ... 54
4.5 Implementation Issues and Technologies Used ... 56
4.6 Scheduling Evaluation on Nimrod-G Simulated Test Queues ... 58
4.7 Scheduling Experiments on the World-Wide Grid ... 60
4.7.1 The World-Wide Grid (WWG) Testbed ... 61
4.7.2 Cost Optimisation Scheduling – Australian Peak and Off-peak Times ... 62
4.7.3 Cost and Time Optimisation Scheduling using Local and Remote Resources ... 67
4.7.4 Large Scale Scheduling with Cost and Time Optimisation ... 70
4.8 Summary and Comments ... 79

Chapter 5 GridSim: A Toolkit for Modeling and Simulation of Grid Resource Management and Scheduling ... 81

5.1 Introduction ... 81
5.2 Related Work ... 82
5.3 GridSim: Grid Modeling and Simulation Toolkit ... 83
5.3.1 Features ... 83
5.3.2 System Architecture ... 83
5.3.3 SimJava Discrete Event Model ... 84
5.3.4 GridSim Entities ... 85
5.3.5 Application Model ... 86
5.3.6 Interaction Protocols Model ... 87
5.3.7 Resource Model – Simulating Multitasking and Multiprocessing ... 89
5.3.8 GridSim Java Package Design ... 94
5.4 Building Simulations with GridSim ... 97
5.5 Summary and Comments ... 99

Chapter 6 Scheduling Simulations ... 101

6.1 Economic Grid Resource Broker Simulation ... 101
6.1.1 Broker Architecture ... 101
6.1.2 Determining the Deadline and Budget ... 103
6.1.3 Scheduling Algorithms ... 104
6.2 Simulation Experiment Setup ... 105
6.2.1 Resource Modeling ... 105
6.2.2 Application Modeling ... 106

6.3 Deadline and Budget Constrained Cost Optimisation Scheduling 107
6.3.1 Scheduling Experiments with a Single User 107
6.3.2 Scheduling Experiments with Multiple Competing Users 113
6.4 Deadline and Budget Constrained Time Optimisation Scheduling 116
6.5 Comparing the Cost and Time Optimisation Scheduling 121
6.6 DBC Cost-Time Optimisation Scheduling 123
6.6.1 Experiment Setup 123
6.6.2 Scheduling Experiments with Cost and Cost-Time Optimisation Strategies 124
6.7 Summary and Conclusion 131

Chapter 7 The Virtual Laboratory: Enabling Drug Design on the Grid 132
7.1 Introduction 132
7.2 Operational Model 133
7.3 Architecture – The Software Stack 134
7.3.1 Docking Code 135
7.3.2 Nimrod-G Tools 136
7.3.3 Chemical Database Management and Intelligent Access Tools 137
7.4 Application Composition 138
7.5 Scheduling Experimentations 142
7.6 Related Work 147
7.7 Summary and Conclusion 147

Chapter 8 Conclusions and Future Directions 149
8.1 Summary 149
8.2 Conclusions 149
8.3 Future Directions 151
8.3.1 Supporting Different Application Models 151
8.3.2 Supporting Different Economic Models 151
8.3.3 Accounting 152
8.3.4 Enhancing GridSim to Support QoS based Resource Entities 152
8.3.5 Wide-Area Data-Intensive Programming and Scheduling Framework 152

Appendix A Tools automate computer sharing 153

Appendix B Toolset teams computers to design drugs 156

Bibliography 159

Table of Figures

Figure 1.1: Volta demonstrates the battery for Napoleon I at the French National Institute, Paris, in 1801. The painting (by N. Cianfanelli, 1841) is from the Zoological Section of "La Specula" at the National History Museum, Florence University, Italy. 1
Figure 1.2: Nimrod-G Grid resource broker. 4
Figure 2.1: Towards Grid computing: A conceptual view. 10
Figure 2.2: Major milestones in networking and computing technologies from the year 1960 onwards. 11
Figure 2.3: A high-level view of the Grid and interaction between its entities. 12
Figure 2.4: A layered Grid architecture and components. 14
Figure 3.1: A generic Grid architecture for computational economy. 28
Figure 3.2: GRACE framework realization within Globus context. 30
Figure 3.3: GRACE Open Trading Protocols. 31
Figure 3.4: A finite state representation of resource trading (for bargain model). 32
Figure 3.5: An Interaction between GSP resource management components. 33
Figure 3.6: Interaction between GSPs and users in a commodity market Grid for resource trading. 39
Figure 3.7: Posted price model and resource trading in a computational market environment. 40
Figure 3.8: Brokers bargaining for lower access price for minimizing computational cost. 41
Figure 3.9: Tender/ContractNet model for resource trading. 42
Figure 3.10: Auctions using external auctioneer. 43
Figure 3.11: Auctions using their own Auctioneer. 44
Figure 3.12: Market-based proportional resource sharing. 45
Figure 4.1: QoS based resource management: intersection of economic, scheduling, and Grid worlds..... 49
Figure 4.2: A layered architecture of Nimrod-G system. 50
Figure 4.3: A Snapshot of Nimrod-G Execution Monitoring and Steering Client. 51
Figure 4.4: Active Sheet – Spreadsheet processing on the Grid using the Nimrod-G broker. 52
Figure 4.5: The Flow actions in the Nimrod-G runtime environment. 52
Figure 4.6: High level steps for adaptive scheduling used in the Nimrod-G broker. 55
Figure 4.7: DBC Cost-optimisation scheduling algorithm behaviour for various budgets. 59
Figure 4.8: Time optimisation scheduling algorithm behaviour for various budgets. 60
Figure 4.9: Conservative time optimisation scheduling algorithm behavior for different budgets. 60
Figure 4.10: The World Wide Grid (WWG) testbed. 61
Figure 4.11: Nimrod-G parameter sweep processing specification. 63
Figure 4.12: Computational scheduling during Australian peak time (US off-peak time). 64
Figure 4.13: Computational scheduling during Australian off-peak time (US peak time). 64
Figure 4.14: Number of resources in use during Australian peak time scheduling experiment..... 65

Figure 4.15: Cost of resources in use at Australian peak time scheduling experiment. 66
Figure 4.16: Number of resources in use at Australian off-peak time scheduling experiment. 66
Figure 4.17: Cost of resources in use at Australian off-peak time scheduling experiment. 67
Figure 4.18: Resource selection in deadline and budget constrained time optimization scheduling. 69
Figure 4.19: Resource selection in deadline and budget constrained cost optimization scheduling. 69
Figure 4.20: Nimrod-G parameter sweep processing specification. 70
Figure 4.21: A snapshot of the Nimrod-G monitor during “optimize for time” scheduling experiment. 72
Figure 4.22: A snapshot of the Nimrod-G monitor during “optimize for cost” scheduling experiment. 72
Figure 4.23: No. of jobs in execution on different Grid resources during DBC time optimization scheduling. 73
Figure 4.24: Total No. of jobs in execution on Grid during DBC time optimization scheduling. 74
Figure 4.25: No. of jobs processed on different Grid resources during DBC time optimization scheduling. 74
Figure 4.26: Total no. of jobs processed on Grid resources during DBC time optimization scheduling. 75
Figure 4.27: The amount spent on different Grid resources during DBC time optimization scheduling. 75
Figure 4.28: The total amount spent on Grid during DBC time optimization scheduling. 76
Figure 4.29: No. of jobs in execution on different Grid resources during DBC cost optimization scheduling. 77
Figure 4.30: Total No. of jobs in execution on Grid during DBC cost optimization scheduling. 77
Figure 4.31: No. of jobs processed on different Grid resources during DBC cost optimization scheduling. 78
Figure 4.32: Total No. of jobs processed on Grid during DBC cost optimization scheduling. 78
Figure 4.33: The amount spent on different Grid resources during DBC cost optimization scheduling. 79
Figure 4.34: The total amount spent on Grid during DBC cost optimization scheduling. 79
Figure 5.1: A modular architecture for GridSim platform and components. 84
Figure 5.2: A flow diagram in GridSim based simulations. 85
Figure 5.3: Entity communication model via its Input and Output entities. 87
Figure 5.4: An event diagram for interaction between a time-shared resource and other entities. 88
Figure 5.5: An event diagram for interaction between a space-shared resource and other entities. 89
Figure 5.6: An event handler for simulating time-shared resource scheduling. 90
Figure 5.7: PE share allocation to Gridlet in time-shared GridSim resource. 91
Figure 5.8: Modeling time-shared multitasking and multiprocessing based on an event scheme. 91
Figure 5.9: An event handler for simulating space-shared resource scheduling. 92
Figure 5.10: PE allocation to the Gridlets in space-shared GridSim resource. 93
Figure 5.11: Modeling space-shared multiprocessing based on an event scheme. 93
Figure 5.12: A class hierarchy diagram of GridSim package. 95
Figure 5.13: Global tags in GridSim package. 96
Figure 5.14: A sample code segment for creating Grid resource and user entities in GridSim. 98
Figure 5.15: The Gridlet method in GridSim. 98
Figure 5.16: A sample code segment for creating a Grid resource broker in GridSim. 99
Figure 6.1: Economic Grid resource broker architecture and its interaction with other entities. 102
Figure 6.2: A class hierarchy diagram of Grid broker using the gridsim package. 103
Figure 6.3: Deadline and budget constrained (DBC) scheduling with cost-optimization. 106
Figure 6.4: No. of Gridlets processed for different budget limits with a fixed deadline for each. 107

Figure 6.5: No. of Gridlets processed for different deadline limits with a fixed budget for each. 108
Figure 6.6: Deadline time utilized for processing Gridlets for different values of deadline and budget. ... 108
Figure 6.7: Budget spent for processing Gridlets for different values of deadline and budget. 109
Figure 6.8: Gridlets processed on resources for different budget values with short deadline. 109
Figure 6.9: Gridlets processed on resources for different budget values with medium deadline. 110
Figure 6.10: Gridlets processed on resources for different budget values with long deadline. 110
Figure 6.11: Trace of No. of Gridlets processed for a short deadline and high budget constraints. 111
Figure 6.12: Trace of budget spent for short deadline and high budget constraints. 111
Figure 6.13: Trace of No. of Gridlets processed for a medium deadline and low budget constraints. 112
Figure 6.14: Trace of No. of Gridlets processed for a long deadline and low budget constraints. 112
Figure 6.15: Trace of the no. of Gridlets committed for a short deadline and high budget constraints. 113
Figure 6.16: Trace of the no. of Gridlets committed for a medium deadline and high budget constraints. 113
Figure 6.17: No. of Gridlets processed for each user when a varying number of users competing for resources. 114
Figure 6.18: The average time at which the user experiment is terminated when a number of users are competing for resources. 115
Figure 6.19: The average budget spent by each user for processing Gridlets. 115
Figure 6.20: No. of Gridlets processed for each user with varying number of users competing for resources. 116
Figure 6.21: The average time at which the user experiment is terminated with varying number of users competing for resources. 117
Figure 6.22: The average budget spent by each user for processing Gridlets. 117
Figure 6.23: Deadline and budget constrained (DBC) time optimisation scheduling algorithm. 118
Figure 6.24: No. of Gridlets processed for different budget and deadline limits. 119
Figure 6.25: The time spent in processing Gridlets using the DBC time optimisation. 119
Figure 6.26: The budget spent in processing Gridlets using the DBC time optimisation. 120
Figure 6.27: Selection of different resources for processing Gridlets for different budget limits. 120
Figure 6.28: The budget spent in processing Gridlets on different resources for different budgets. 121
Figure 6.29: The time spent in processing application jobs using time cost and time optimisation scheduling algorithms given different budget limits. 122
Figure 6.30: The budget spent in processing application jobs using time cost and time optimisation scheduling algorithms given different budget limits. 122
Figure 6.31: Deadline and budget constrained (DBC) scheduling with cost-time optimisation. 123
Figure 6.32: The number of Gridlets processed, time, and budget spent for different deadline and time limits when scheduled using the cost and cost-time optimisation algorithms. 125
Figure 6.33: The number of Gridlets processed and resources selected for different budget values with a long deadline value when scheduled using the cost and cost-time optimisation algorithms. 126
Figure 6.34: The number of Gridlets processed and resources selected for different deadline values with a large budget when scheduled using the cost and cost-time optimisation algorithms. 127
Figure 6.35: The time spent for processing application jobs for different deadline constraints with a large budget when scheduled using the cost and cost-time optimisation algorithms. 127
Figure 6.36: The budget spent for processing application jobs for different deadline constraints with a large budget when scheduled using the cost and cost-time optimisation algorithms. 128
Figure 6.37: Trace of No. of Gridlets processed on resources for a medium deadline and high budget constraints when scheduled using the cost optimisation strategy. 129

Figure 6.38: Trace of No. of Gridlets processed on resources for a medium deadline and high budget constraints when scheduling using the cost-time optimisation strategy..... 129
Figure 6.39: Trace of the number of Gridlets committed to resources for a medium deadline and high budget constraints when scheduled using the cost optimisation strategy. 130
Figure 6.40: Trace of the number of Gridlets committed to resources for a medium deadline and high budget constraints when scheduled using the cost-time optimisation strategy. 130
Figure 7.1: X-ray crystal structure of a target protein receptor and small molecules to be docked. 133
Figure 7.2: Resource brokering architecture for screening molecules on distributed resources. 134
Figure 7.3: Layered architecture of Virtual Laboratory for drug design. 135
Figure 7.4: Relation between key programs in the *dock* suite. 136
Figure 7.5: Deployment of Virtual Laboratory components at runtime and their interaction. 137
Figure 7.6: Protocols for Interaction between the CDB clients and the server. 138
Figure 7.7: A configuration input file for docking application. 139
Figure 7.8: Parameterisation of a configuration input file 140
Figure 7.9: A plan file defining parameters type and their values. 140
Figure 7.10: Task definition of docking jobs. 141
Figure 7.11: Parameterisation of script for extracting molecule from CDB. 141
Figure 7.12: Static and Dynamic Input Files of Docking program. 143
Figure 7.13: A snapshot of the Nimrod-G monitor during “Optimize for Time” scheduling. 143
Figure 7.14: A snapshot of the Nimrod-G monitor during “Optimize for Cost” scheduling. 143
Figure 7.15: No. of jobs processed on Grid resources during DBC time optimization scheduling. 144
Figure 7.16: The amount spent on resources during DBC time optimization scheduling. 145
Figure 7.17: No. of jobs in execution on Grid resources during DBC time optimization scheduling. 145
Figure 7.18: No. of jobs processed on Grid resources during DBC Cost optimization scheduling. 146
Figure 7.19: The amount spent on resources during DBC Cost optimization scheduling. 146
Figure 7.20: No. of jobs in execution on Grid resources during DBC Cost optimization scheduling. 147

Economic-based Distributed Resource Management and Scheduling for Grid Computing

Doctoral Candidate: Rajkumar Buyya
Doctoral Advisor: Professor David Abramson

School of Computer Science and Software Engineering
Monash University, Melbourne, Australia

Abstract

Grid computing, emerging as a new paradigm for next-generation computing, enables the sharing, selection, and aggregation of geographically distributed heterogeneous resources for solving large-scale problems in science, engineering, and commerce. The resources in the Grid are heterogeneous and geographically distributed. Availability, usage and cost policies vary depending on the particular user, time, priorities and goals. The management of resources and application scheduling in such a large-scale distributed environment is a complex task. This thesis proposes a distributed computational economy as an effective metaphor for the management of resources and application scheduling. It proposes an architectural framework that supports resource trading and quality of services based scheduling. It enables the regulation of supply and demand for resources; provides an incentive for resource owners to participate in the Grid; and motivates the users to trade-off between deadline, budget, and the required level of quality-of-service. The thesis demonstrates the capability of economic-based systems for wide-area parallel and distributed computing by developing users' quality-of-service requirements-based scheduling strategies, algorithms, and systems. It demonstrates their effectiveness by performing scheduling experiments on the World-Wide Grid for solving parameter sweep—task and data parallel—applications.

Thesis Contributions: To support the thesis that an economic-based Grid resource management and scheduling system can deliver significant value to users, resource providers and consumers, compared to traditional approaches, we have:

- identified the key requirements that an economic-based Grid system needs to support,
- developed a distributed computational economy framework called the **Gr**id **A**rchitecture for **C**omputational **E**conomy (GRACE), which is generic enough to accommodate different economic models and maps well onto the architecture of wide-area distributed systems,
- designed deadline and budget constrained scheduling algorithms with four different strategies: cost, time, conservative-time, and cost-time optimisations,
- developed a Grid resource broker called Nimrod-G that supports deadline and budget constrained algorithms for scheduling parameter sweep applications on the Grid,
- developed a Grid simulation toolkit, called GridSim, that supports discrete-event based simulation of Grid environments to allow repeatable performance evaluation under different scenarios,
- evaluated the performance of deadline and scheduling algorithms through a series of simulations by varying the number of users, deadlines, budgets, and optimisation strategies and simulating geographically distributed Grid resources, and
- demonstrated the effectiveness and application of Grid technologies for solving real-world problems such as molecular modelling for drug design on the WWG (World-Wide Grid) testbed.

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text, and a list of references is given.

Rajkumar Buyya
April 10, 2002

Acknowledgements

I am grateful to numerous local and global “peers” who have contributed towards shaping this thesis.

At the outset, I would like to express my appreciation to Professor David Abramson for his advice during my doctoral research endeavor for the past three years. As my supervisor, he has constantly forced me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth. I thank him for providing me with the opportunity to work with a talented team of researchers.

I would like to thank my former supervisor Professor Clemens Szyperski who helped me to start my doctoral research in Australia (Queensland University of Technology) with a smile. Surprisingly, within six months of my candidature, he announced his plan for joining Microsoft Research in the United States, which triggered my migration to Monash University in search of a new advisor. This migration resulted in going beyond clusters: building technologies for world-wide distributed computing and demonstrating their suitability for real applications. This happens to coincide with my original theme for PhD research dreamt of in the early 1990s!

I am grateful to Jonathan Giddy (Distributed Systems Technology Centre, Melbourne) for contributing significantly towards the implementation of a prototype Nimrod-G system based on the architecture and scheduling policies proposed and investigated as part of our distributed computational economy research. He has helped me greatly and been a source of knowledge with real world programming skills, specifically in Python!

I would like to express my sincere thanks to Manzur Murshed (Monash University, Gippsland Campus) for generously sharing his time and knowledge in our cooperative work on building a foundation infrastructure for grid simulation. He has played a major role in making me understand the concept of discrete-event simulation.

I thank Kim Branson (Walter and Eliza Hall Institute for Medical Research) for sharing information on the molecular docking application and data model that enabled me to develop a virtual laboratory environment. It helped me in demonstrating the suitability of our Grid technologies for enabling drug design application processing on the Grid.

I would like to thank Ian Foster (Argonne National Laboratory, USA) for his critical comments and suggestions during the initial phases of thesis work.

I am grateful to Mark Baker (University of Portsmouth, UK), Domenico Laforenza (CNR, Italy), Heinz Stockinger (CERN, Switzerland), Klaus Krauter (Redfern Networks, Australia), Muthucumaru Maheswaran (University of Manitoba, Canada), Steve Chapin (Syracuse University, USA), and David DiNucci (Elepar, USA) for sharing their knowledge that influenced the work and resulted in cooperative joint publications.

I greatly appreciate Rob Gray (Distributed Systems Technology Centre) for his generous time in proof reading. I would like to express my special thanks to him for being a great soul. He is always ready to help with a smile.

I thank Omer Rana (Cardiff University), Franck Cappello (University of Paris-Sud), Marcelo Pasin (Federal University of Santa Maria), Shoaib Ali Burq (University of Melbourne), Srikumar Venugopal (University of Melbourne), Ajith Abraham (Monash University), Jagan Kommineni (Monash University), Rajanikanth Batchu (Mississippi State University), Slavisa Garic (DSTC), Andrew Hunter (Monash University), Rohit Patil, Sohel Khan (Icode Software), Abhishek Bhagat (Digital India), and Kay and Mike Higginbottom (AnEyeForDetail.net) for their comments and proof reading some of the chapters.

I thank John Crossley (Monash University), Jack Dongarra (University of Tennessee, Knoxville),

Wolfgang Gentzsch (Sun Microsystems), Francine Berman (University of California, San Diego), Geoffrey Fox (Indiana University), Toni Cortes (Universitat Politecnica de Catalunya, Barcelona), Hai Jin (University of Southern California, Los Angeles), Savithri S (Motorola India Electronics Ltd., Bangalore), Sudharshan Vazhkudai, Spyros Lalis (FORTH, Greece), Jim Basney (University of Wisconsin), Colin Gan (University of Minnesota), Nirav Kapadia (Purdue University), John Karpovich (Virginia University), Andre Merzky (ZIB, Berlin), Jarek Nabrzyski (Poznan Supercomputing Centre), Hidemoto Nakada (Tokyo Institute of Technology), Michael Neary (UCSB), Joern Gehring (Paderborn Center for Parallel Computing, Germany), Achim Streit (Paderborn), Henri Casanova (UCSD), Lance Norskog (Enron), Scott Jackson (Pacific Northwest National Laboratory), and Craig Lee (Aerospace Corporation) for their comments.

I thank Andrzej Goscinski (Deakin University) and Andrew Wendelborn (University of Adelaide) for their detailed critique on this thesis.

I would like to thank colleagues and organisations for sharing their resources that have been utilised to create a World-Wide Grid (WWG) testbed. The organizations whose resources I have used in scheduling experiments reported in this thesis are: Monash University (Melbourne, Australia), Victorian Partnership for Advanced Computing (Melbourne, Australia), Argonne National Laboratories (Chicago, USA), University of Southern California's Information Sciences Institute (Los Angeles, USA), Tokyo Institute of Technology (Tokyo, Japan), National Institute of Advanced Industrial Science and Technology (Tsukuba, Japan), University of Lecce (Italy), and CNUCE-Institute of the Italian National Research Council (Pisa, Italy), Zuse Institute Berlin (Berlin, Germany), Charles University, (Prague, Czech Republic), University of Portsmouth (UK), and University of Manchester (UK).

I acknowledge the Australian Government, Queensland University of Technology, Monash University, Cooperative Research Centre for Enterprise Distributed Systems Technology (DSTC), Distributed Systems and Software Engineering Centre, and the IEEE Computer Society for providing scholarships to pursue doctoral studies.

I would like to thank administrative and technical staff members of the school who have been kind enough to advise and help in their respective roles. I thank Thomas Peachey for making life fun while working.

Last, but not least, I would like to dedicate this thesis to my family, my wife Smrithi and daughter Soumya, for their love, patience, and understanding—they allowed me to spend most of the time on this thesis. For the past six months, my daughter kept on reminding me: "Appaja, go to office and write your thesis"!

Rajkumar Buyya

April 10, 2002, Melbourne, Australia