

Adaptive Resource Relocation in Virtualized Heterogeneous Clusters



A thesis submitted for the degree of
Doctor of Philosophy
of
The Australian National University

Muhammad Atif
November 2010

I declare that the work in this thesis is entirely my own and that to the best of my knowledge it does not contain any materials previously published or written by another person except where otherwise indicated.

Muhammad Atif
November 2010

This thesis is dedicated to the loving memory of my father, who taught me to believe in myself and have a positive attitude towards life.

I also dedicate the thesis to my mother.

Acknowledgements

I would like to express appreciation for Dr. Peter Strazdins for his guidance and advise during my doctoral research for the past three and a half years. As my supervisor, he guided me to remain in focus towards achieving the goal. His observations and comments helped me to broaden the knowledge in the research area and beyond. He also was instrumental in establishing the overall direction of the research and steer me towards the goals and milestones of my research work.

I would like to thank my advisors Dr. Alistair Rendell and Dr. Eric McCreath for their useful comments, suggestions and encouragement throughout my research work. They extended all the possible support during my research work, which was not only at the academic level but also at the personal level.

My special gratitude for Computer Systems Research Group at the department of computer science, ANU. These 'geeks' have a great tradition of supporting each other in every possible way. Their constant support, discussions and useful suggestions acted as catalysts to my research output. I would like to especially thank Mr. Ahmed El Zein from the ANU supercomputer facility for helping me in setting up the compute farm.

Appreciation for Alexander Technology and Platform Computing for donating the hardware, especially Mr. Richard Alexander for his vision that enabled this research work.

My very special thanks to National University of Sciences and Technology (NUST), Pakistan for providing me a scholarship for the first three years of my candidature at the Australian National University.

I sincerely thank Department of Computer Science, ANU for providing me tuition fee and living allowance scholarship for the rest of my candidature. This enabled me to remain in focus to the research and not to worry about financial constraints. Especially Dr. Peter Christian, who in his capacity as Associate Dean (Higher Degree Research) always extended his full support. I would take this opportunity to thank all the administration and technical staff at the department of computer science. During the three and a half years of research, I never faced any administration issue.

To my wife Bushra, and our children Rameen and Zain, thank you for being there for me throughout the candidature. Especially Bushra for giving me strength and support during the tough times. I would also like to thank my mother for remembering me in her prayers.

Abstract

Cluster computing has recently gone through an evolution from single processor systems to multicore/multi-socket systems. This has resulted in lowering the cost/performance ratio of the compute machines. Compute farms that host these machines tend to become heterogeneous over time due to incremental extensions, hardware upgrades and/or nodes being purchased for users with particular needs. This heterogeneity is not surprising given the wide range of processor, memory and network technologies that become available and the relatively small price difference between these various options. Different CPU architectures, memory capacities, communication and I/O interfaces of the participating compute nodes present many challenges to job scheduling and often result in under or over utilization of the compute resources. In general, it is not feasible for the application programmers to specifically optimize their programs for such a set of differing compute nodes, due to the difficulty and time-intensiveness of such a task.

The trend of heterogeneous compute farms has coincided with resurgence in the virtualization technology. Virtualization technology is receiving widespread adoption, mainly due to the benefits of server consolidation and isolation, load balancing, security and fault tolerance. Virtualization has also generated considerable interest in the High Performance Computing (HPC) community, due to the resulting high availability, fault tolerance, cluster partitioning and accommodation of conflicting user requirements. However, the HPC community is still wary of the potential overheads associated with virtualization, as it results in slower network communications and disk I/O, which need to be addressed.

The live migration feature, available to most virtualization technologies, can be leveraged to improve the throughput of a heterogeneous compute farm (HC) used for HPC applications. For this we mitigated the slow network communication in Xen; an open source virtual machine monitor. We present a detailed analysis of the communication framework of Xen and propose communication configurations that give 50% improvement over the conventional Xen network configuration. From a detailed study of the migration facility in Xen, we propose an improvement in the live migration facility specifically targeting HPC applications. This optimization gives around 50% improvement over the default migration facility of Xen.

In this thesis, we also investigate resource scheduling in heterogeneous compute farm with the perspective of dynamic resource re-mapping. Our approach is to profile each job in the compute farm at runtime, and propose a better resource mapping compared to the initial allocation. We then migrate the job(s) to the best-suited homogeneous sub-cluster to improve overall throughput of the

HC. For this, we develop a novel heterogeneity and virtualization-aware profiling framework, which is able to predict the CPU and communication characteristics of high performance scientific applications.

Our experimental setup consists of a 32 node virtualized cluster with machines having CPU frequency between 2 GHz to 3 GHz. The framework was tested with a subset of NAS parallel benchmarks with the workload generated from Dror Feitelson's workload archives. The prediction accuracy of our performance estimation model is over 80%. The framework implementation is lightweight, with an overhead of 3%. Our experiments show that we are able to improve the throughput of the compute farm by 25% and the time saved by the HC with our framework is over 30%. The framework can be readily extended to HCs supporting a cloud computing environment.

Contents

Acknowledgements	v
Abstract	vii
I Introduction	1
1 Introduction	3
1.1 Motivation	3
1.2 Research Objectives	6
1.3 Research Goals	6
1.4 Methodology	7
1.5 Contributions	8
1.6 Thesis Outline	10
II Background	13
2 Heterogeneous Cluster Scheduling	15
2.1 Heterogeneous Clusters	15
2.2 Programming Issues in Heterogeneous Clusters	16
2.3 Approaches to Parallel Application Scheduling in an HC	17
2.3.1 Performance Modeling and Estimation	17
2.3.2 Heterogeneity-aware Schedulers	19
2.3.3 Heterogeneity-aware Applications	22
2.4 Approaches to Job scheduling	24
2.4.1 Batch Processing	24
2.4.2 Gang Scheduling	26
2.5 Scheduling Solutions	27
2.5.1 Maui Scheduler	28
2.5.2 The N1 Grid Engine	29
2.6 Chapter Summary	29
3 Virtualization in Cluster Computing	31
3.1 Overview of Virtualization	31
3.2 Challenges in x86 Virtualization	33
3.3 Xen Architecture	35
3.3.1 Xen Hypervisor	36

3.3.2	Domain 0	36
3.3.3	Domain U	36
3.3.4	Management Software	37
3.4	Xen's Bridged Network Architecture	37
3.5	Live Migration Basics	38
3.6	Use of Virtualization in HPC	40
3.6.1	Use of HPC and Xen in Cloud Computing	43
3.7	Chapter Summary	44
III	Infrastructure Improvement	47
4	Communication Infrastructure Improvement	49
4.1	Related Work	49
4.1.1	Intra-domain Communication Improvement	50
4.2	Network Configurations	51
4.3	MPI Implementation Differences	54
4.4	Results	54
4.4.1	Experimental Setup	55
4.4.2	Application Level Benchmarks	55
4.4.3	Inter-domain Communication	56
4.4.4	Intra-domain Communication	59
4.4.5	Inter-Intra Node Communication Mix	62
4.5	Discussion and Summary	67
5	Live Migration Improvement	69
5.1	Related Work	69
5.2	Experimental Platform	71
5.3	Xen Migration Statistics	72
5.4	Optimization of the Live Migration Routine	74
5.4.1	Comparison of Migration Techniques	75
5.4.2	Migration on 1×n Cluster Configurations	75
5.4.3	Migration on 2×n Cluster Configurations	79
5.5	Summary	81
IV	Design, Implementation and Evaluation	83
6	Design and Implementation of a Resource Relocation Framework	85
6.1	Motivation	85
6.2	Related Work	86
6.3	Profiling-Based Execution Time Model	87

6.3.1	Assumptions	88
6.3.2	Computational Model	88
6.3.3	Communication Characterization of Parallel Programs	89
6.3.4	Memory Utilization	92
6.3.5	Predicted Execution Time	92
6.3.6	Migration Prediction	93
6.3.7	Migration Decisions	95
6.4	Framework Implementation	97
6.4.1	Virtualization-aware Scheduler	101
6.5	ARRIVE-F Accuracy and Overheads	102
6.5.1	Experimental Setup	102
6.5.2	Applications	102
6.5.3	Comparison of Predicted and Actual Execution Times	103
6.5.4	Framework Overheads	108
6.6	Chapter Summary	109
7	Evaluation of ARRIVE-F	111
7.1	Live Migration Experiments	111
7.1.1	Computation Requirement Migration	112
7.1.2	Computational Requirement Migration 2	114
7.1.3	Communication Requirement Migration	115
7.1.4	Migration due to Insufficient Memory	116
7.2	Compute Farm Throughput	116
7.2.1	Experiment Number 1	119
7.2.2	Experiment Number 2	123
7.2.3	Experiment Number 3	125
7.3	Sensitivity Analysis	127
7.3.1	Different values of τ	128
7.3.2	Migration Reevaluation	128
7.3.3	Migration Threshold	129
7.3.4	Migration β	130
7.4	Discussion on ARRIVE-F Migration Decisions	130
7.5	Chapter Summary	136
8	Conclusions and Future Work	137
8.1	Infrastructure Improvement	137
8.2	ARRIVE-Framework	139
8.3	Future Directions	141

V	Appendices	147
A	Appendix A: Lublin-Feitelson Job Streams for the Experiments	149
A.1	Introduction	150
A.2	Job Data - Experiment 1	150
A.3	Job Data - Experiment Number 2	157
A.4	Job Data - Experiment Number 3	162
B	Comparison of Various Intra-domain Communication Mechanisms.	167
B.1	Introduction	168
B.1.1	Latency Results	169
B.1.2	Bandwidth Results	170
C	Appendix C: The Database Schema	175
D	Appendix D: ARRIVE-F Statistics and Screenshots	177
D.1	Lines of Code	178
D.2	Dependencies	178
D.3	ARRIVE-F Screenshot	178
	Bibliography	181

List of Figures

1.1	ARRIVE-F: Adaptive Resource Relocation In Virtualized Environments framework block diagram.	9
3.1	Privilege rings in the x86 architecture [courtesy [21]].	34
3.2	Xen environment block diagram [courtesy [33]].	35
3.3	Xen bridge architecture.	38
3.4	Stages in default Xen migration [courtesy: [46]].	39
4.1	Various network configurations available to Xen VMs.	52
4.2	OSU 2 pair bandwidth benchmark	60
4.3	OSU 3 pair bandwidth benchmark	60
4.4	OSU 4 pair bandwidth benchmark	61
4.5	OSU bandwidth benchmark (1-pair, co-located VMs)	61
4.6	OSU bandwidth benchmark (2-pair, co-located VMs)	62
4.7	NAS parallel benchmarks on a 1×4 cluster.	63
4.8	NAS parallel benchmarks on a 2×4 cluster	63
4.9	NAS parallel benchmarks, Class A on a 8×1 cluster configuration	65
4.10	NAS parallel benchmarks, Class A on a 8×2 cluster	66
5.1	Different Cluster Configurations.	71
5.2	Migration of a single VM from a VMM in a 1×4 configuration	73
5.3	Migration of a 131072 Page VM on a 1×n cluster conf.	76
5.4	Effects of different migration routines on a migrating node and its ‘source’ VMM during the execution of FT.B.4 benchmark.	77
5.5	Effects of different migration routines on a migrating node and its ‘source’ VMM during the execution of CG.B.4 benchmark.	79
5.6	Migration of two VMs from a single VMM on a 2×4 cluster configuration.	80
5.7	Migration of two VMs (131072 Pages each) on a 2×4 cluster configuration.	80
6.1	Pictorial view of heterogeneous compute farm.	86
6.2	Migration decision scenario 1 - Migration of equal nodes.	96
6.3	Migration decision scenario 2 - Migration to free nodes.	96
6.4	Migration decision scenario 3 - Migration with the free nodes.	97
6.5	Migration decision scenario 4 - Migration with multiple jobs.	97
6.6	ARRIVE-F block view.	98

6.7	Migration Sequence of a cluster exchange between two parallel applications.	100
6.8	Overheads of the profiling framework and daemons.	108
7.1	Gantt chart of the first 2000 seconds of experiment number 1	124
7.2	ARRIVE-F not migrating a 2 process job to make way for 8 process job.	133
7.3	Serial vs parallel migration of two jobs spanning multiple VMMs . .	134
7.4	Impact of parallel migration	135
7.5	Serial vs parallel migration of multiple jobs spanning multiple VMMs	135
B.1	Latency results of various intra-domain communication mechanisms in Xen.	170
B.2	Bandwidth of various Xen intra-domain communication mechanisms.	173
B.3	Bidirectional bandwidth of various Xen intra-domain communication mechanisms.	173
C.1	Extended entity-relationship diagram of the ARRIVE-F database. . .	176
D.1	ARRIVE-F dashboard.	179

List of Tables

4.1	Summary of latency benchmark (μ Sec) at 1 byte	57
4.2	Summary of latency benchmark (MB/Sec) at 4 MB	58
4.3	Avg. bandwidth benchmark (MB/Sec) for message size $\geq 4K$	58
4.4	Avg. bi-bandwidth benchmark (MB/Sec) for message size $\geq 4K$	59
4.5	Avg. bandwidth benchmark (MB/Sec) for message size $\geq 4K$	62
5.1	Avg. migration statistics of an idle virtual machine.	72
5.2	Average statistics of migrating a 512 MB (131072 Page) VM on a 1×4 cluster configuration.	73
5.3	Optimized migration statistics of a 131072 Page VM on a 1×4 cluster configuration.	75
6.1	Heterogeneous compute farm	102
6.2	Computational model accuracy (predicted vs actual)	104
6.3	Penalty in CPU cycles	105
6.4	Calculation of time estimate (seconds) based on computation model $\tau = 50$ seconds	105
6.5	Communication model accuracy (predicted vs actual)	106
6.6	Communication calculations for $\tau = 50$ seconds	107
7.1	Computational requirement migration	112
7.2	Calculation of time estimate based on computation model $\tau = 50$ seconds	113
7.3	Computational requirement migration	114
7.4	Calculation of time estimate based on computation model $\tau = 50$ seconds	115
7.5	Communication requirement migration	116
7.6	Memory requirement migration	116
7.7	Base-run vs migration run	120
7.8	Job allocation in base run	120
7.9	Experiment 1: Job migration 1	120
7.10	Experiment 1: Job migration 2	121
7.11	Experiment 1: Job Migration 3	122
7.12	Experiment 2: Base-run vs migration run	123
7.13	Experiment 2: Job migration 1	125
7.14	Experiment 3: Base-run vs migration run	126
7.15	Experiment 3: Job migration 1	126
7.16	Experiment 3: Job migration 2	127

7.17	Experiment 3: Job migration 3	127
7.18	Effect of changing the value of τ	128
7.19	Effect of different pooling intervals τ	128
7.20	Effect of the migration threshold	129
7.21	Migration β	130
7.22	ARRIVE-F not migrating a 2 process job... [Job Queue]	132
A.1	Experiment 1: List of jobs	156
A.2	Experiment 2: List of jobs	161
A.3	Experiment 3: List of jobs	166
B.1	Comparison of intra-domain communication latency μ seconds	169
B.2	Comparison of intra-domain communication bandwidth	171
B.3	Comparison of intra-domain bidirectional bandwidth	172
D.1	Lines of code	178