

Abstract

Parallel processing is being used more and more to solve complex problems in the computing world. Common to all parallel processing is the exchange of data in between various processes of computing. One of the main paradigms used for this is message passing. Message passing can be a serious source of overheads for the efficiency of parallel algorithms. The topology, or layout, or the way computers are physically interconnected with each other, dramatically affects the overhead created by message passing.

It is believed that parallel algorithms designed with communication schemes of parallelization for specific network topologies, out perform the more general equivalents when performing on their network of predilection. To develop these algorithms, more research is required on the effects of topology on message passing overheads.

NTS – Network Topology Simulator is a tool designed to help study the effects of topology on parallel algorithms. NTS simulates a network topology and is able to run a standard PVM (Parallel Virtual Machine) program on this simulated network. A PVM parallel algorithm requires only minor changes to be able to run on NTS and experience the effects of the various simulated topologies.

Tools, such as EPPA (Experimental Parallel Performance Analysis) or the internal performance recording facilities of NTS, can then be used to investigate the effects of the simulated topology on the parallel algorithm.

Acknowledgements

I would like to thank my promoter Erik Dirkx, for getting me interested into parallel algorithms with such a good class. My assistant Jan Lemeire, without whom this thesis would be not, as he is the one who proposed changing my thesis subject to continuing the AOM project, for which I am thankful. Min Zhang and especially Olivier Thonnard, my companions through the first design of this project, without them AOM could never have been realized and therefore this second incarnation NTS would never have been possible. Olivier, thank you for making such a good and robust job at the router code and bringing us your light on the routing issues. I would like to mention my friends Marek Suliga and Andy Hill, both for helping me through Linux, testing, helping in times of need, teaching me, while having a great time. I would like to thank my girlfriend Clara Stynen for time and understanding. To one and all, thanks.

Table of Contents

	Page
Abstract	i
Acknowledgements	ii
Table of Contents	iii
Table of Figures	viii
Table of Tables	x
1 Introduction	1
1.1 Parallel Processing	1
1.1.1 Massively Parallel Processors	1
1.1.2 Distributed Computing	1
1.1.3 Communicating Processes	2
1.2 Parallel Study	2
1.2.1 Scalability	2
1.2.2 Speedup	3
1.2.3 Efficiency	3
1.2.4 Coefficient of Parallization	4
1.3 Topology	6
1.3.1 Full Interconnect	6
1.3.2 Static Networks	6
1.4 Why NTS	9
2 Theory	12
2.1 PVM	12
2.2 Design Patterns	13
2.2.1 Creational Patterns	13
2.2.2 Structural Patterns	13
2.2.3 Behavioral Patterns	14
2.2.4 Elements of a Design Pattern	14

2.2.5	Observer Pattern	14
2.2.6	Singleton Pattern	16
2.2.7	Iterator Pattern	17
2.2.8	Façade Pattern	18
2.3	Qt	19
2.4	Graphs	20
2.5	AOM	21
3	<u>Design</u>	22
3.1	Aim of the project	22
3.2	Goals of NTS	22
3.3	Principals Behind NTS	23
3.3.1	Development Language	23
3.3.2	Deadlocks	23
3.3.3	Equivalency with PVM	23
3.3.4	Routing	23
3.3.5	Dynamic Update Network	24
3.3.6	Parallel Processing Network	24
3.4	Definitions	25
3.5	ntsRouter	25
3.5.1	Message Forwarding Logging	25
3.5.2	Routing Protocol	26
3.5.3	Router Example	28
3.5.4	Message Processing	29
3.5.5	ntsRouter Options	29
3.6	ntsLibrary	29
3.6.1	Connecting	30
3.6.2	Settings	30
3.6.3	Message Forwarding	31
3.6.4	Receiving Forwarded Messages	31
3.6.6	Sequence	31

3.6.7	Integer Combination	32
3.6.8	Lookup Table	33
3.6.9	Overloaded vs. Forwarded	34
3.7	ntsMaster	34
3.7.1	Creation	34
3.7.2	The Running Network	35
3.7.3	Dynamic Rebuilding	35
3.7.4	Master Design	35
4	Using NTS	38
4.1	Installing	38
4.2.1	Getting NTS	38
4.2.2	Installing	38
4.2.3	Notes	38
4.2.4	Requirements	39
4.3	Converting a PVM Application	39
4.4	Restrictions of Use	39
4.5	Running NTS Simulation	40
4.6	ntsMaster	40
4.6.1	The Main Window	40
4.6.2	Network Building	41
4.6.3	Main Graphic View	42
4.6.4	NTS Log Display	42
4.6.5	Lists View	43
4.6.7	Properties View	43
4.6.8	Log View	43
4.6.9	Message Log	43
4.6.10	Path View	44
4.6.11	Time Matrix	44
4.6.12	Player	44
4.6.13	Save/Load	45

4.6.14	Auto Building	45
4.6.15	Setting Routers to Silent	45
4.6.16	All Bi-Directional	46
4.7	Known Bugs	46
4.7.1	PVM_Config	46
5	Implementation	47
5.1	ntsRouter	47
5.1.1	The Router Object	47
5.1.2	The Controller	50
5.2	ntsLibrary	51
5.2.1	Settings	52
5.2.2	Forwarding Functions	52
5.2.3	Sending Messages	52
5.2.4	Receiving Messages	53
5.2.5	Disconnecting	53
5.3	ntsMaster	53
5.3.1	ntsNetwork Package	55
5.3.2	Controller Package	56
5.3.3	Graphics	58
5.3.4	Views	61
5.3.5	Player	61
5.3.6	Threads	61
5.3.7	File Format	61
6	Conclusion	63
6.1	Further Development	64
6.1.1	Listening To Any Incoming Message	64
6.1.2	Efficiency	65
6.1.3	Robustness	66
6.1.4	Extensions	66

7	<u>References</u>	67
8	<u>Appendix</u>	68
8.1	antGraph	68
8.2	UML Diagrams	69

Table of Figures

	Page
Figure 1: Speedup versus the number of processing elements for adding a list of numbers	4
Figure 2: Efficiency vs. number of processors	4
Figure 3: A completely non-blocking crossbar network connecting p processors to b memory banks	6
Figure 4: Removing links from a full interconnect network to make a static forward network	7
Figure 5: Construction of hypercubes from hypercubes of lower dimension.	8
Figure 6: Two and three-dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.	8
Figure 7: The layers of NTS	10
Figure 8: Representation of a (dynamic) 3-by-3, 2-D mesh with wraparound	11
Figure 9: Example of many windows showing different views of the same data	15
Figure 10: The Observer Pattern Structure	15
Figure 11: The singleton pattern structure	17
Figure 12: Iterator Pattern structure	18
Figure 13: Façade Pattern Structure	19
Figure 14: Detail of network shown in Figure 8	28
Figure 15: The AOM layer keeps the send and receive sequence for each TID to which a message was sent/a message was received from and checks the two to make sure they are the same for reordering	32
Figure 16: Integer Combination size limitation	33
Figure 17: The main packages of the ntsMaster and their connections	36
Figure 18: ntsMaster Main Window	41
Figure 19: ntsMaster Network Toolbar	41
Figure 20: ntsMaster Add Router Popup	41
Figure 21: ntsMaster Main window showing extra views	44

Figure 22: ntsRouter packages	47
Figure 23: Classes view of the Router Package	49
Figure 24: ntsLibrary packages	51
Figure 25: ntsLibSettings class	52
Figure 26: The main packages of the ntsMaster and their connections	54
Figure 27: ntsMaster, ntsNetwork Package / data structure classes	54
Figure 28: The forwarding process of adding a new Router	57
Figure 29: Instantiation and connection of the Message List	57
Figure 30: Illustration of flow of data for a selection	58
Figure 31: ntsMaster Graphical Package	59
Figure 32: save file format	62
Figure 33: antGraph Package	68
Figure 34: ntsPlayer	69
Figure 35: ntsNetwork Package	70
Figure 36: ntsGraphics Package	71
Figure 37: ntsRouter, Router Package	72

Table of Tables

	Page
Table 1: Example of a routing table (Router#1)	29
Table 2: Description of the “Router” class	48
Table 3: ntsRouter messages	50 - 51