Scientific
Computation
Series

Multicomputer Networks
Message-Based Parallel Processing

Daniel A. Reed and
Richard M. Fujimoto

The MIT Press

Multicomputer Networks: Message-Based Parallel Processing

Daniel A. Reed and Richard M. Fujimoto

The MIT Press
Cambridge, Massachusetts
London, England

Publisher's Note

This format is intended to reduce the cost of publishing certain works in book form and to shorten the gap between editorial preparation and final publication. Detailed editing and composition have been avoided by photographing the text of this book directly from the author's prepared copy.

Second printing, 1988

This book was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Reed, Daniel A.
Multicomputer networks.

(MIT Press series in scientific computation)
Includes index.
1. Computer networks. 2. Parallel processing (Electronic computers) I. Fujimoto, Richard M. II. Title. III. Series.
TK5105.5.R44 1987 004'.35 87-22833
ISBN 0-262-18129-0

Series Foreword

It is often the case that the periods of rapid evolution in the physical sciences occur when there is a timely confluence of technological advances and improved experimental techniques. Many physicists, computer scientists, chemists, biologists, and mathematicians have said that such a period of rapid change is now underway. We are currently undergoing a radical transformation in the way we view the boundaries of experimental science. It has become increasingly clear that the use of large-scale computation and mathematical modeling is now one of the most important tools in the scientific and engineering laboratory. We have passed the point where the computer is viewed as just a device for tabulating and correlating experimental data; we now regard it as a primary vehicle for testing theories for which no practical experimental apparatus can be built. NASA scientists speak of "numerical" wind tunnels, and physicists use supercomputers to see what happens during the birth of a universe.

The major technological change accompanying this new view of experimental science is a blossoming of new approaches to computer architecture and algorithm design. By exploiting the natural parallelism in scientific applications, new computer designs show the promise of being able to solve problems whose solutions were considered unreachable a few years ago. When coupled with the current biennial doubling of memory capacity, supercomputers are on their way to becoming the laboratories of much of modern science.

With the advent of fast, powerful microprocessors, a new branch of the computer industry has emerged. By using large numbers of these cheap processors, each connected to a large private memory, it is possible to build a computing system with very impressive potential performance. If the processors are connected to each other so that they can exchange messages in a reasonably efficient manner and if the programmer can decompose his computation into a large system of communicating processes, such a multicomputer network can be a powerful supercomputer.

Fujimoto and Reed have written the first comprehensive treatment of the architecture and performance modeling of this family of nonshared-memory MIMD computing systems. Their book begins with a survey of the basic architectural ideas behind the current generation of hypercube systems but quickly moves into the main contribution of the volume: a foundation for the analytical modeling and rigorous simulation of multicomputer networks. Along the way they give a treatment of the VLSI constraints on network nodes and a survey of the issues confronted in designing operating systems for multicomputer networks. The volume concludes with a detailed performance analysis of the current crop of commercial systems.

While it is very exciting to see a book on a topic of such great current interest to so many scientists, it is even more important to see a volume that can set the standard for analytical study of these systems. This volume is required reading not only for students wishing to learn about the current family of systems but also for the architects designing the next generation of hypercube machines.

Dennis B. Gannon

Preface

The recent appearance of powerful single-chip processors and inexpensive memory has renewed interest in the message passing paradigm for parallel computation. Much of this interest can be traced to the construction of the CalTech Cosmic Cube, a group of Intel 8086/8087 chips with associated memory connected as a D-dimensional **hypercube**.

Following the success of the Cosmic Cube, four companies (Intel, Ametek, Ncube, and Floating Point Systems) quickly began delivery of commercial hypercubes. These hypercube systems are but one member of a broader class of machines, called **multicomputer networks**, that consist of a large number of interconnected computing nodes that asynchronously cooperate via message passing to execute the tasks of parallel programs. Each network node, fabricated as a small number of VLSI chips, contains a processor, a local memory, and (optionally) a communication controller capable of routing messages without delaying the computation processor.

Multicomputer networks pose several important and challenging problems in network topology selection, communication hardware design, operating systems, fault tolerance, and algorithm design. This monograph summarizes recent results in each of these areas, with the following emphases.

- *Analytic Models of Interconnection Networks.* Although the hypercube topology has many attractive features, many other network interconnection topologies are not only possible, but are often preferable for certain algorithm classes. Chapter two analyzes the effectiveness of several metrics for topology evaluation and presents a new metric based on asymptotic bound analysis. The chapter concludes with a comparison of several proposed network topologies.

- *VLSI Constraints and Communication.* Whether each node of a multicomputer network is implemented as a single VLSI component or as a printed circuit board, packaging constraints limit the number of connections that can be made to the node, placing an upper bound on the total I/O bandwidth available for communication links. As more links are added to each node, less bandwidth is available for each one. However, increasing the number of links connected to each node will usually reduce the mean number of hops required to reach a particular destination. Chapter three examines the balance between link bandwidth and mean hop count as the number of links to each node varies.

- *Communication Paradigms and Hardware Support.* Because multicomputer networks are limited in extent to a few cabinets, rather than a geographic area, they require hardware, rather than software, support for message transport, routing, buffer

management, and flow control. Chapter four evaluates design alternatives for each of these issues and presents one possible hardware realization.

• *Multicomputer Operating Systems.* There are two primary approaches to parallel processing and task scheduling on a multicomputer network. In the first, all parallel tasks in a computation are known *a priori* and are mapped onto the network nodes before the computation is initiated. The tasks remain on their assigned nodes throughout the entire computation. In the second, the mapping of tasks onto network nodes is done dynamically. Here, a parallel computation is defined by a dynamically created task precedence graph where new tasks are initiated and existing tasks terminate as the computation unfolds. Although static tasks can be scheduled at compile time by a single processor, dynamically created tasks must be assigned to network nodes by a distributed scheduling algorithm executing on the network. Chapter five summarizes approaches to the static scheduling problem, analyzes the feasibility of dynamic task scheduling, and reviews the current state of the art.

No examination of message passing systems would be complete without a discussion of potential application algorithms. Parallel discrete event simulation represents one extreme of the application spectrum where the task interaction pattern can be very irregular and change greatly over the program lifetime. At the opposite extreme, iterative partial differential equations (PDE) solvers typically iterate over a regular grid with nearest neighbor communication among parallel tasks.[1]

• *Applications: Distributed Simulation.* Simulation of complex systems imposes extraordinary computational requirements. War games with battlefield management, functional simulation of integrated circuits and Monte Carlo simulation of many particle systems often require hundreds or thousands of hours on the fastest computer systems that are currently available. Several approaches to discrete event simulation based on parallel execution and message passing paradigms have been developed. Chapter six surveys the state of the art in distributed simulation and discusses the performance ramifications for multicomputer networks.

• *Applications: Partial Differential Equations.* Existing hypercubes have already been widely used as testbeds for iterative, partial differential equations solvers. However, there are many ways to partition the iteration grid into parallel tasks. For example, on the Intel iPSC hypercube, domain partitioning by strips has been used even though strip

[1]More sophisticated PDE solvers exist, however, that exhibit complex, time dependent behavior.

partitions are provably non-optimal when data transfer is used as a metric. The reason: the high message startup costs on the iPSC favor minimization of message *count* rather than the amount of data transferred. Chapter seven analyzes the interaction of problem partitioning and architectural parameters and presents a performance model that identifies the appropriate partitioning for a given multicomputer network.

- *Commercial Hypercubes: A Performance Analysis.* Four hypercube families are now commercially available: the Intel iPSC, Ametek System/14, Ncube/ten, and FPS T Series. Moreover, several groups are actively constructing research prototypes, notably the JPL Mark-III at CalTech and NASA Jet Propulsion Laboratory. Chapter eight evaluates the hardware designs and system software of the four commercial hypercubes and the JPL Mark-III. In addition, chapter eight presents a comparative performance study using a set of processor and communication benchmarks.

Acknowledgments

We would be remiss not to acknowledge the colleagues and students whose insights, enthusiasm and suggestions shaped this work during its formative stages. In the early years, Herbert Schwetman and Carlo Séquin directed the dissertation research presented in chapters two, three and four. Their guidance was invaluable; we owe them a great debt.

Many of the benchmarks in chapter eight were conducted by Dirk Grunwald while visiting the NASA Jet Propulsion Laboratory. The evaluations of competing hypercube architectures and insights into their performance limitations are his. His eagerness to hypothesize and experiment embody the scientific spirit.

Members of the Concurrent Computation Project at CalTech/JPL, particularly John Fanslow and Herb Madan, cheerfully offered advice and provided access to the JPL Mark-III. Jack Dongarra provided access to the Intel iPSC at Argonne National Laboratory. David Poplawski and Brenda Helminen of Michigan Technological University graciously provided both the benchmark data and the FPS T Series example program discussed in chapter eight.

The students in the *Picasso* research group were a continual source of inspiration and delight. Without them this would not have been possible. To Dirk Grunwald, David Bradley, Bobby Nazief, Chong Kim and Balkrishna Ramkumar, our undying gratitude. Finally, students from the Computer Science Departments at the University of Utah and the University of Illinois at Urbana-Champaign provided many valuable comments on drafts of the text while they were used for classes.

During this writing Daniel Reed was supported in part by the National Science Foundation under grants NSF DCR 84-17948, NSF DCI 86-05082 and an NSF Presidential Young Investigator Award, and by the National Aeronautics and Space Administration under grants NASA NAG-1-613 and NASA NAG-1-595. Richard Fujimoto was supported by the Office of Naval Research under contract N00014-87-K-0184 and a University of Utah Research Grant. Both Dr. Reed and Dr. Fujimoto were also supported by Faculty Development Awards from International Business Machines.

Daniel A. Reed
Richard M. Fujimoto

July 1987

Contents

	Series Foreword	xiii
	Preface	xv
1	**Introduction**	1
1.1	Multicomputer Networks: A Definition	2
1.2	A Parallel Computer Systems Comparison	5
	1.2.1 Pipelined Vector Processors	5
	1.2.2 Shared Memory MIMD Multiprocessors	7
	1.2.2.1 The Sequent Balance 21000	8
	1.2.2.2 Large Scale Multiprocessors	8
	1.2.3 SIMD Computers	10
	1.2.4 Systolic Arrays	12
	1.2.5 Data Flow Computers	13
	1.2.6 Multicomputer Networks	15
1.3	Multicomputer History	16
	1.3.1 Early Projects	16
	1.3.2 The Hypercube Topology	17
	1.3.3 The Cosmic Cube and Its Descendants	19
1.4	Multicomputer Building Blocks	19
	1.4.1 The Inmos Transputer	20
	1.4.2 The Torus Routing Chip	20
1.5	Outline of Text	23
	1.5.1 Communication Systems	24
	1.5.2 Multicomputer Operating Systems	25
	1.5.3 Multicomputer Applications	26
	1.5.4 Performance of Existing Machines	26
2	**Analytic Models of Interconnection Networks**	29
2.1	Definitions	30
	2.1.1 Network Connectivity	31
	2.1.2 Network Diameter	33
	2.1.3 Mean Internode Distance	33

2.1.3.1 Uniform Message Routing - Symmetric Interconnections 35
2.1.3.2 Sphere of Locality Message Routing - Symmetric Interconnections 36
2.1.3.3 Decreasing Probability Message Routing - Symmetric Interconnections 37
2.1.3.4 Uniform Message Routing - Asymmetric Interconnections 39
2.1.4 Visit Ratios 40
2.1.5 Expansion Increments 41
2.2 Single Stage Interconnection Networks 42
2.3 Analyzing the Torus: An Example 46
2.3.1 The Binary Hypercube: A Special Case 52
2.4 Analysis of Single Stage Interconnection Networks 52
2.4.1 Implications of Network Size 53
2.4.2 Expansion Increment Comparison 53
2.4.3 Network Connectivity Comparison 56
2.4.4 Network Diameter Comparison 56
2.4.5 Mean Internode Distance Comparison 56
2.5 Communication Link Visit Ratios 60
2.5.1 Feasible Computation Quanta 62
2.5.2 Network Selection 66
2.6 Nodes with Limited Communication Bandwidth 68
2.7 A Comparison Based on Rankings 69
2.8 Network Performance at Finite Workloads 69
2.8.1 Asymptotic Bound Analysis 70
2.8.2 Product Form Queueing Networks 71
2.8.3 Balanced Job Bound Analysis 74
2.8.3.1 Approximate Intersection Points 76
2.9 Summary 77

3 VLSI Constraints and the Optimal Number of Ports 80
3.1 VLSI Constraints 81
3.2 Virtual Cut-Through 82

3.3 Analytic Studies 83
3.3.1 Assumptions 83
3.3.2 Model I: Cluster Nodes 85
3.3.2.1 Queueing Model 86
3.3.2.2 Delay 90
3.3.2.3 Bandwidth 91
3.3.3 Model II: Fixed Sized Networks 94
3.3.3.1 Queueing Model 95
3.3.3.2 Delay 99
3.3.3.3 Bandwidth 103
3.3.4 Summary of Analytic Results 104
3.4 Simulation Studies 107
3.4.1 Assumptions 107
3.4.2 The Application Programs 108
3.4.2.1 Barnwell Filter Program (global SISO, 12 tasks) 111
3.4.2.2 Block I/O Filter Program (local SISO, 23 tasks) 111
3.4.2.3 Block State Filter Program (local SISO, 20 tasks) 112
3.4.2.4 FFT Program (local PIPO, 32 tasks) 113
3.4.2.5 LU Decomposition (global PIPO, 15 tasks) 114
3.4.2.6 Artificial Traffic Loads (global PIPO, 12 tasks) 116
3.4.2.7 Characterization of the Application Programs 116
3.4.3 Issues Under Investigation 119
3.4.4 Simulation Results on Cluster Node Networks 120
3.4.4.1 Fully Connected Networks 121
3.4.4.2 Full-Ring Tree Networks 122
3.4.4.3 Butterfly Networks 123
3.4.4.4 Ring Networks 123
3.4.4.5 Conclusions for Cluster Node Networks 127
3.4.5 Simulation Results on Fixed Sized Networks 128
3.4.5.1 Lattice Topologies 128
3.4.5.2 Tree Topologies 130
3.4.5.3 De Bruijn Networks 131
3.5 Summary 134

4 Communication Paradigms and Hardware Support 138
4.1 Transport Mechanisms 139
4.2 A Virtual Circuit Based Communication System 142
4.2.1 Virtual Circuits 142
4.2.2 Virtual Channels 143
4.2.3 Translation Tables 144
4.2.4 Switch Architecture 145
4.3 Routing 148
4.4 Buffer Management 152
4.5 Flow Control 157
4.5.1 A Send / Acknowledge Protocol 158
4.5.2 Remote Buffer Management 162
4.6 Evaluation of Design Parameters 164
4.6.1 Number of Virtual Channels 164
4.6.2 Amount of Buffer Space 165
4.6.2.1 Buffer Space: Deadlock Considerations 166
4.6.2.2 Buffer Space: Performance Considerations 169
4.7 Complexity of the Communication Circuitry 172
4.8 Summary 174

5 Multicomputer Network Operating Systems 177
5.1 Overview 179
5.2 Scheduling Static Tasks 183
5.2.1 A Formal Model of Static Task Scheduling 183
5.2.2 Static Scheduling Via Integer Programming 186
5.2.3 Static Scheduling Via Clustering 188
5.2.3.1 Initialization 190
5.2.3.2 Iterative Task Assignment 190
5.2.4 Static Scheduling Via Simulated Annealing 192
5.2.4.1 A Sample Problem 194
5.2.4.2 Minimization Criterion 196
5.2.4.3 Algorithm Implementation 197
5.2.4.4 Experiments 197

5.2.4.5 General Observations 198
5.3 Scheduling Dynamic Tasks 198
5.3.1 A Feasibility Study of Dynamic Scheduling 200
5.3.1.1 Task Precedence Graphs 201
5.3.1.2 Simulation Methodology 202
5.3.1.3 Simulation Experiments 205
5.3.1.4 General Observations 212
5.3.2 Dynamic Scheduling Via Gradient Planes 213
5.3.2.1 Creating Gradient Planes 214
5.3.2.2 The Gradient Plane Algorithm 214
5.3.2.3 Simulation Studies 217
5.3.2.4 General Observations 222
5.3.3 Dynamic Scheduling Via Waves 224
5.3.3.1 Management Hierarchies 224
5.3.3.2 Wave Scheduling Algorithm 225
5.3.3.3 Analysis of Wave Scheduling 228
5.4 Fault Tolerance 230
5.4.1 Recovery Methods 232
5.4.2 Summary 234
5.5 Future Directions in Operating Systems 234

6 Applications: Distributed Simulation 239
6.1 Simulation of Discrete Systems 239
6.1.1 The Causality Principle 240
6.1.2 What is Distributed Simulation? 241
6.1.3 Why is Distributed Simulation Hard? 243
6.1.4 An Example 245
6.1.5 Overview of Distributed Simulation Strategies 247
6.2 The Deadlock Avoidance Approach 248
6.2.1 The Selection Phase 252
6.2.2 The Computation Phase 252
6.2.3 The I/O Phase 252
6.2.4 Initialization and Termination 253

6.3 The Deadlock Detection and Recovery Approach 253
6.3.1 The Simulation Phase 253
6.3.2 The Dijkstra Scholten Algorithm (DSA) 254
6.3.3 The Chandy Misra Algorithm (CMA) 256
6.3.4 Deadlock Recovery 258
6.3.5 Simulation Using Deadlock Detection 260
6.4 The Time Warp Mechanism 261
6.4.1 Assumptions 261
6.4.2 Logical Processes 261
6.4.3 Error Detection and Roll Back 262
6.4.4 Anti-Messages 263
6.4.5 Global Virtual Time 264
6.4.6 Flow Control 265
6.4.7 I/O and Runtime Errors 265
6.4.8 Termination Detection 265
6.5 Summary 266

7 Applications: Partial Differential Equations 268
7.1 Solution Techniques 270
7.2 Grid Partitions 273
7.2.1 Related Work 274
7.2.2. Five Point Stencil 275
7.2.2.1 Rectangular Partitions 275
7.2.2.2 Triangular Partitions 276
7.2.2.3 Hexagonal Partitions 277
7.2.3 Nine Point Stencil 278
7.2.3.1 Rectangular Partitions 279
7.2.3.2 Triangular Partitions 280
7.2.3.3 Hexagonal Partitions 280
7.2.4 Other Stencils 281
7.3 Computation/Communication Ratios 282
7.4 Optimal Pairs of Stencil and Partition 283
7.5 Performance Models 287

7.5.1 Message Passing Analysis 292
7.5.2 Performance Prediction 295
7.5.3 Experimental Validation 298
7.6 Summary 303

8 Commercial Hypercubes: A Performance Analysis 305
8.1 Hypercube Architectures 307
8.1.1 Intel iPSC 307
8.1.1.1 Hardware Organization 307
8.1.1.2 Software Organization 311
8.1.2 Ametek System/14 317
8.1.2.1 Hardware Organization 317
8.1.2.2 Software Organization 319
8.1.3 JPL Mark-III 324
8.1.3.1 Hardware Organization 325
8.1.3.2 Software Organization 327
8.1.4 Ncube/ten 328
8.1.4.1 Hardware Organization 328
8.1.4.2 Software Organization 330
8.1.5 FPS T Series 330
8.1.5.1 Hardware Organization 331
8.1.5.2 Software Organization 334
8.2 Hypercube Performance Analysis 335
8.2.1 Performance Study I (Intel, Ametek, JPL) 335
8.2.1.1 Test Environment 336
8.2.1.2 Processor Benchmarks 338
8.2.1.3 Simple Communication Benchmarks 341
8.2.1.4 Synthetic Communication Benchmarks 347
8.2.1.5 Temporal Locality 348
8.2.1.6 Spatial Locality 349
8.2.1.7 Experimental Results 350
8.2.2 Performance Study II (FPS T Series) 352
8.2.2.1 Processor Benchmarks 353

8.2.2.2 Communication Benchmarks 356
8.2.2.3 FPS T Series System Balance 359
8.2.3 Performance Study III (Intel iPSC/VX) 361
8.3 Hypercube System Comparisons 362
8.3.1 Hardware 363
8.3.2 Software 364
8.3.3 Summary 370
8.4 Future Directions 370

Index 378