Performance Enhancement of Bus-Based Parallel/Distributed Systems

by

Sibabrata Ray

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor David Klarner

and Professor Hong Jiang

Lincoln, Nebraska

August, 1995

UMI

300 North Zeeb Road
Ann Arbor, MI 48103

DISSERTATION TITLE

Performance Enhancement of Bus-Based Parallel/Distributed Systems

BY

Sibabrata Ray

SUPERVISORY COMMITTEE:

	APPROVED	DATE
Signature	David A. Klarner	
Typed Name	Prof. David Klarner (Chair)	
Signature		
Typed Name	Prof. Hong Jiang (Co-Chair)	
Signature	Joseph Leung	June 5, 1995
Typed Name	Prof. Joseph Leung	
Signature		June 1, 1995
Typed Name	Prof. Jean-Camille Birget	
Signature	Sarit Mukherjee	05/08/95
Typed Name	Prof. Sarit Mukherjee	
Signature	Earl S. Kramer	5/25/95
Typed Name	Prof. Earl Kramer	

GRADUATE COLLEGE
UNIVERSITY OF NEBRASKA

Performance Enhancement of Bus-Based Parallel/Distributed Systems

Sibabrata Ray, Ph.D.

University of Nebraska, 1995

Advisors: David Klarner and Hong Jiang

In a tightly coupled environment reusing slots in the traditional sense is not a lucrative option because each erasure node will introduce a few bits delay. For an on board system, where each slot is only a few bits long, such delay is intolerable. We have developed a method for reusing part of the bandwidth available from spatial/spectral bandwidth extension without introducing any buffering delay. Further, we have provided a polynomial time algorithm to optimally reconfigure the bus under the design constraints for any given traffic pattern. We have found that the reconfigured bus substantially outperforms the traditional slotted bus in most practical scenarios.

In a dual-bus local area network the need for erasure nodes introduces a few bits of delay for each added erasure node, thus it is not profitable to make all nodes erasure nodes. This fact gives rise to the problem of placing a minimum number of erasure nodes on the network so that the throughput gain is maximum. This thesis gives the first polynomial time algorithm to solve the problem for all traffic conditions.

The second part of the thesis considers partitioning and mapping task graphs for distributed shared memory systems (which includes bus based systems). The problem in its general form is known to be NP-complete and it is important to come up with simple but effective and fast heuristics. We have considered the

tree task graphs which arise from many important programming techniques such as divide-and-conquer, branch-and-bound, etc. Further, we have proposed a polynomial time algorithm for data allocation in cacheless bus-based multiprocessor systems.

ACKNOWLEDGMENT

I would like to thank my advisor Prof. Hong Jiang for all the help he provided me, both at professional and personal level. Without him I would not be able to complete this dissertation.

Special thanks goes to Prof. Sarit Mukherjee. I really enjoyed working with him. Thanks also to Prof. Birget for all the stimulating discussions. I would like to thank Prof. David Klarner, Prof. Joseph Y-T. Leung and Prof. Earl Kramer for serving on my committee.

Finally, many thanks to my wife Keka and my four year old Raka. I am indebted to them for all my happiness during the years of frustration through which every Ph. D. student goes.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The invention of digital computers has changed the methodology used in science and engineering research drastically. Indeed, over the past decades computing has become an essential tool for advancing various science and engineering disciplines. Advances in computer science and engineering have also helped to solve many heretofore "intractable" problems. However, the advancements in science and engineering are bringing up new problems everyday and the demand for more powerful computers is increasing. An example from [70] will illustrate the historical trend very nicely. Two decades ago, when fluid dynamicists were simulating flows in one dimension, CDC 7600 machines from Control Data Corporation could perform very accurate computations in a single dimension of space and time. During the last decade, researchers began to use CRAY-1 machines to perform accurate computations of two-dimensional time-dependent fluid flow. The CRAY-1 is 10 times faster in comparison to The CDC 7600 with twice the memory. Today's computers are 100 times faster in comparison to the CRAY-1 and have 1,000 times more memory. Computational fluid dynamicists now routinely perform computational experiments that rival the complexity of laboratory experiments. Such

historical anecdotes are abundant in the literature [54, 14].

Researchers in the fields of scientific and engineering computation have had the extremely good fortune to see their capabilities increase by more than an order of magnitude each decade for the last few decades. However, as the device characteristics are fast approaching their physical limits it is becoming harder and harder to sustain this pace of growth. Therefore, computer scientists are trying to develop new innovative hardware and software design methodologies to sustain the growth rate in the face of the physical limits. Most scientific and engineering applications can be divided into small subtasks in such a way that some of those subtasks may be executed concurrently. The need to exploit this inherent parallelism of the scientific and engineering applications, coupled with the advent of cost effective, reliable and versatile VLSI/ULSI technology, has driven parallel systems from the realm of theoretical studies and speculations to that of commercial product and applications.

Parallel and distributed systems have expanded the horizon of computational disciplines beyond the imagination of the scientists of the previous generation. However, if history is any indication, no amount of computational power is going to satisfy the need of the new generation of scientists and engineers. Hence, the performance of existing parallel and distributed systems needs to be enhanced continuously. Researchers continue to mount their effort in this direction and their successes have upgraded computation from a supporting process in theoretical and experimental investigations to a main tool for advancing science and technology.

The effort to develop more and more powerful parallel and distributed systems has been mainly channeled into three directions, namely, architectural, algorithmic and application systems. Researchers working in the architectural domain are

continuing to design new and innovative architectural components to increase the raw computational power. On the other hand, researchers working in algorithmic and application systems domain continue to invent better algorithms and better system software to harness and properly utilize the raw power. In recent years, noticeable inventions have been made in all three domains.

Some of the recent success of computer architects includes the invention of RISC architecture [55], multiprocessor cache memory [8, 71, 47, 52], new innovative network technology such as wormhole routing [11], and different latency hiding techniques such as prefetching and non-block send/receive. For a list of inventions in the other two domains we refer the reader to [44, 9].

In spite of all the success mentioned, the performance of parallel and distributed systems needs to be enhanced further. In this dissertation we propose methods to enhance the performance of bus-based parallel and distributed systems both in architectural as well as in application systems domain. Bus-based systems are used in both shared memory architecture and LAN/MAN based network of workstations (NOW). In the next section, the architectures of both shared memory systems and networks of workstations are described. In the application systems domain we have considered applications which give rise to tree and linear task graphs. The concept of task graphs is discussed later in detail. One novel feature of the dissertation needs to be mentioned here. Usually performance evaluation techniques are used to predict the performance of a system already designed. In this dissertation performance evaluation techniques are integrated into performance enhancement.

1.2 Architecture of Parallel Systems

A parallel system consists of many processors working in cooperation to solve one or more problems. An application task is divided among the processors according to some scheduling strategy. The processors work on the subtasks assigned to them and may need to communicate with other processors from time to time depending on the need of the task. In general, computing systems are divided in four large architectural classes [22]:

1. Single instruction, single data architecture (SISD)
2. Single instruction, multiple data architecture (SIMD)
3. Multiple instruction, single data architecture (MISD)
4. Multiple instruction, multiple data architecture (MIMD)

Among the four classes SIMD and MIMD have been most extensively studied in the parallel processing community, because of their natural suitability for parallel processing systems. SIMD machines apply a single instruction to many data simultaneously whereas in MIMD machines many instruction streams are executed on independent data sets simultaneously. In this dissertation we are mostly concerned with the MIMD architecture.

MIMD machines are further classified on the basis of the mechanism used to establish interprocessor communication. The shared memory architecture and the message passing architecture are two paradigms of MIMD machines. In a shared memory architecture all processors share a single global logical address space and anything written in the memory by one processor may be read or modified by any other processor. On the other hand, for the message passing architecture,

processors have to send/receive messages explicitly. In this dissertation we shall confine ourselves mostly to the shared memory paradigm.

The concept of shared memory multiprocessors has become very popular in the last several years. Both academia and industry have shown a keen interest in this particular architectural paradigm and, in addition to several research prototypes like RP3 of IBM [56], Cedar at University of Illinois [23], TRAC at University of Texas [63], Dash at Stanford and Hector at University of Toronto, many shared memory systems are available on the market. Examples of commercial shared memory systems include the Sequent Balance/Symmetry, the Encore Multimax, the Alliant FX series, the BBN Butterfly, the Kendall Square series. Due to the vast popularity enjoyed by the shared memory systems, it is of utmost importance to continue to enhance the performance of shared memory systems.

1.2.1 Shared Memory Systems

A shared memory multiprocessor consists of several processing elements (PEs) and memory modules (MMs). The communication between the PEs and MMs is provided by the interconnection network (IN). Logically all the memory modules together form a single address space and each processing element can access the whole address space. The processors communicate and synchronize through the shared memory. The architectural organization of the multiprocessor depends on the organization of the interconnection network.

Shared memory systems may be divided into two classes.

1. Tightly coupled or true shared memory (Figure 1.1) systems, and

2. Loosely coupled or distributed shared memory (Figure 1.2) systems.

In a tightly coupled shared memory system processors are connected to one

side of the network and the memory modules are connected to the other side. When a processor accesses a memory location it has to use the interconnection network. In a loosely coupled shared memory system each processor has its own memory module. A processor does not need to go through the interconnection network to access its own memory module. However, to access other memory modules it has to go through the interconnection network.

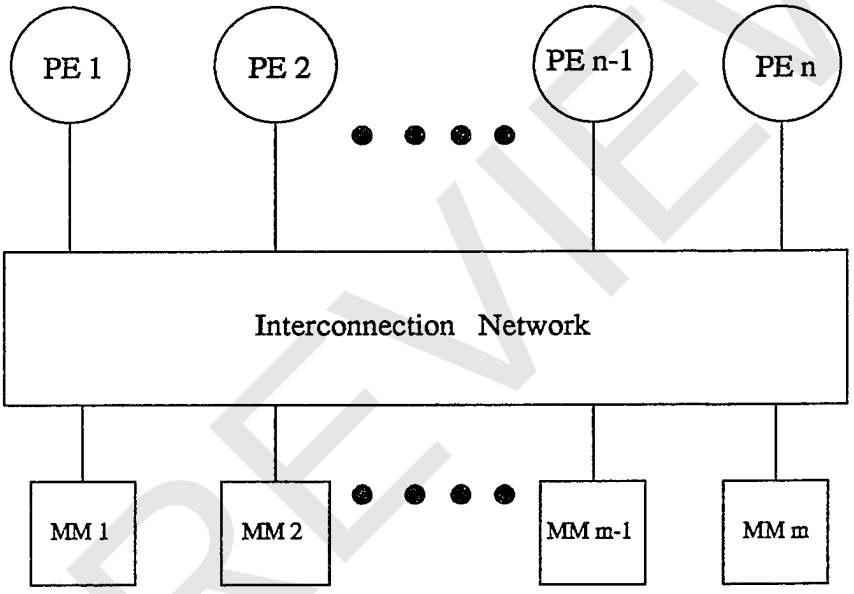

Figure 1.1: Tightly coupled shared memory system

An interconnection network provides the coordination necessary for the system to operate effectively. It carries requests from one point to the other and operates on address mapping. That is, a request is generated with one or more destinations and the routing is done by the destination address. Routing of multiple requests may or may not be done in parallel.

An interconnection network may be *synchronous* or *asynchronous*. It may be *packet switched*, *circuit switched* or *wormhole-routed*, and *centralized* or *decentral-*

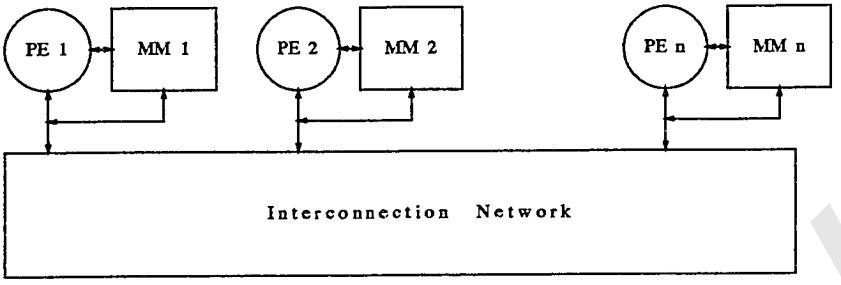

Figure 1.2: Loosely coupled shared memory system

ized, depending on its operational characteristics. Architecturally, interconnection networks are divided in two classes:

1. Static, and

2. dynamic.

For a static interconnection network the network is fixed and does not change its configuration. The connections between the processors are fixed and the processors route the request through the proper connection. However, for dynamic interconnection networks the network topology is dynamic and may be changed by setting the switches properly. Dynamic interconnection networks may be classified into three categories: 1) Crossbar, 2) shared bus and 3) multistage networks.

Crossbar

A crossbar interconnection network is a fully connected non-blocking network. In other words, any permutation of memory accesses may be routed using a crossbar switch parallelly. Therefore, all memory accesses may proceed simultaneously as long as there is no memory conflict among the set of processors. A crossbar switch is shown in Figure 1.3.

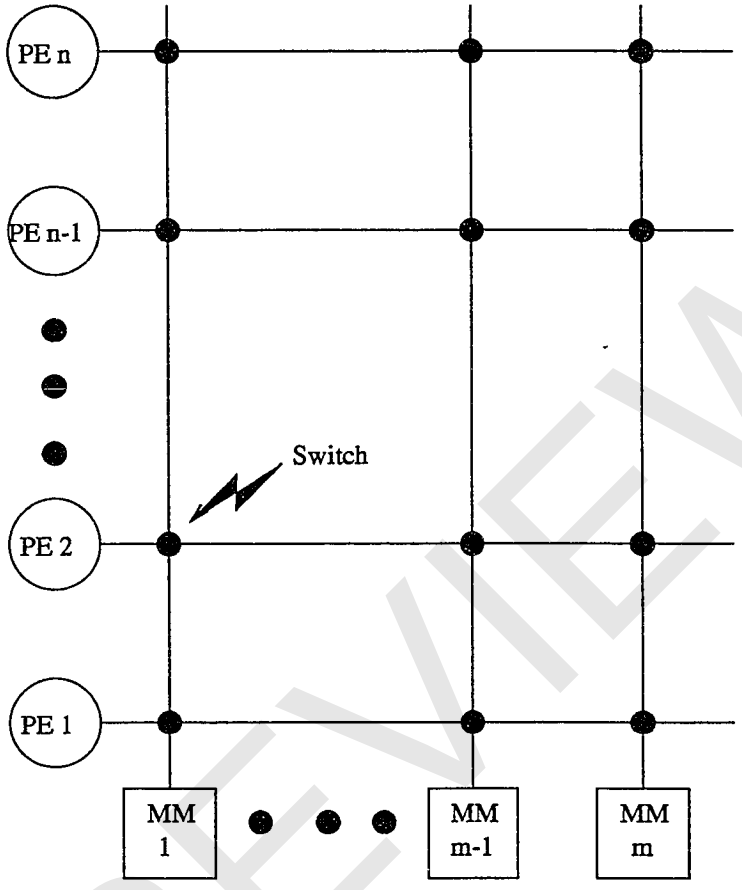

Figure 1.3: Crossbar interconnection network

The high flexibility provided by a crossbar switch comes at a very high price: it needs to have $O(NM)$ switching elements, where n and m are the numbers of PEs and MMs respectively. Therefore, a crossbar becomes prohibitively costly for large systems. During the random mode of execution, memory conflicts reduce the effective bandwidth of a crossbar switch. Therefore, crossbars are considered useful only for fairly small systems.

Shared-Bus

A shared bus is a cable (electrical or optical) through which several modules communicate. All modules located along the shared bus try to gain access to it via some bus-controller or arbiter. If more than one module tries to gain access at the same time, the arbiter/controller grants access to one of them according to some preset priority rule. The shared bus is probably the simplest interconnection network. Furthermore, the shared bus is a natural medium for broadcasting. It is especially suitable for implementing snoopy cache coherence protocols. Due to its simplicity and other beneficial features it has been implemented by several multiprocessor computer manufacturers. However, a shared bus can provide very limited bandwidth only and therefore, creates a bottleneck for fine-grain parallel processing.

To alleviate the problems arising from the bandwidth limitation of the traditional shared bus, several bandwidth expansion strategies have been proposed. They include bandwidth expansion along three dimensions:

1. **Temporal dimension.** Bandwidth expansion in the temporal dimension involves pipelining the signal over the bus. This way several processors may access the bus at the same time. Hence the temporal bandwidth expansion results in both increased bandwidth and decreased latency.

2. **Spatial dimension.** Bandwidth expansion in the spatial dimension involves using multiple buses instead of one bus.

3. **Spectral dimension.** For buses built on a frequency oriented medium (for example an optical fiber), each node may transmit in more than one frequency (possibly in bit parallel fashion). Expanding the bandwidth of

a bus by transmitting in more than one frequency is called expanding the bandwidth in spectral dimension.

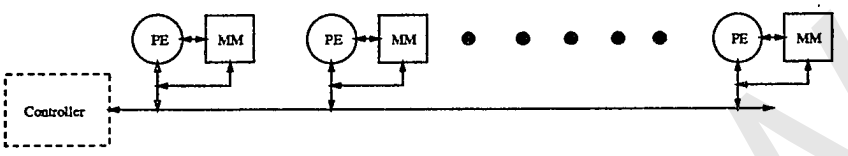

Figure 1.4: A shared-bus based system

All the bandwidth expansion strategies described retain all the beneficial features of the conventional bus and also provide the system with higher bandwidth. Therefore, larger systems with high performance can be built on the bandwidth expanded bus structures.

Multistage Interconnection Networks (MIN)

An $N \times N$ multistage interconnection network is capable of connecting N modules to N modules, where N is usually a power of some integer. If $N = c^n$ then the multistage interconnection network is built using $c \times c$ crossbar switches in n levels. Each switch has c inputs and c outputs. A packet arriving at one input is routed to some output depending on $\log c$ control bits. Many different MIN's have been designed and are available commercially. An example of a commercially available MIN based multiprocessor is the BBN's Butterfly.

In Figure 1.6 an 8×8 delta network built using 2×2 crossbar switches is shown. MIN's are classified in two broad categories depending on their ability to permute requests: 1) Blocking and 2) non-blocking. In a non-blocking MIN any permutation can be realized. However, in a blocking MIN only a part of all possible permutations can be realized in parallel.