

# Matching Algorithms and Architecture in Hierarchical Shared-Memory Multiprocessor (HSM) Systems

Ashfaq Khokhar and Michel Dubois

Department of Electrical Engineering-Systems  
University of Southern California  
Los Angeles, CA 90089-2562  
dubois@priam.usc.edu

## Abstract

In this paper, we map several interprocessor communication and linear algebra algorithms on a memory coherent hierarchical shared-memory multiprocessor (HSM) system and their communication complexities are evaluated. The results show that the hierarchical architecture is ill-suited to algorithms exhibiting no temporal locality on data accesses or to the algorithms with point-to-point communication.

## 1 Introduction

With the advent of VLSI, massively parallel processing systems with thousands of processing elements are feasible today. The design of shared-memory multiprocessors that can be scaled up for large number of processors is a topic of active research. The advantage of shared memory systems over distributed memory systems is that the programmer's model of a shared-memory machine is limited to a set of threads sharing a common memory. Therefore, shared-memory systems must efficiently support parallel multithreaded applications.

Several shared memory systems with large number of processors are being designed using hierarchy of processors in clusters. Examples include the CEDAR [6], the Data Diffusion Machine [4], and the Encore Giga-max [10]. An HSM- $i$  is a Hierarchical Shared-memory Multiprocessor with  $i$  levels. An HSM-1 is a cluster of  $Q_1$  processors. In general, an HSM- $i$  is obtained by connecting  $Q_i$  HSM- $(i-1)$  systems,  $i = 1, 2, \dots$ .

The question addressed in this paper is whether multithreaded algorithms can be mapped effectively on an HSM- $i$ ,  $i = 1, 2, \dots$ . Rather than designing algorithms specifically for HSM's, we investigate straightforward parallel implementations of several interprocessor communication algorithms, as well as of several linear algebra algorithms on a two-level architecture (HSM-2). A parallel algorithm is said to be "well-matched" to the hierarchical architecture if the memory traffic, in terms of complexity, is the same in each interconnection at all levels of the hierarchy. Unless an algorithm is well-matched to the architecture it does not run efficiently on a system with very large number of processors as the interconnections at the highest

levels become major bottlenecks.

We start by giving an overview of the architecture and the coherence protocol in Section 2. In Section 3 and Section 4, we analyze the communication complexities of the basic communication algorithms and of the linear algebra algorithms on an HSM. We conclude in Section 5 with a discussion of our results<sup>1</sup>.

## 2 Hierarchical Shared-memory Multiprocessor (HSM) System

In this paper, we restrict our analysis to an HSM-2, i.e. we are assuming a hierarchy of two levels only such that  $Q_1 = P$  and  $Q_2 = Q$ . However, results obtained for HSM-2 can easily be generalized to HSM- $i$ , for  $i > 2$ . Figure 1 shows the architecture, which consists of the following components.

- A large number of processors organized into  $Q$  clusters and  $P$  processors per cluster.
- A main memory shared by all clusters.
- A global cache shared by processors in each cluster.
- A local cache associated with each processor.
- A local interconnection network (LIN) connecting the processors in each cluster to their global cache.
- A global interconnection network (GIN) connecting the global caches of all clusters to the main memory.

*Intracluster* communication (i.e. communication among processors in the same cluster) is accomplished through the LIN. *Intercluster* communication is accomplished through the GIN.

In order to reduce the effect of memory access delays, multiple copies of each data may exist in different global and local caches. In general, if there is a copy of

<sup>1</sup>In this paper we briefly outline our results. The detailed research work can be found in [5].

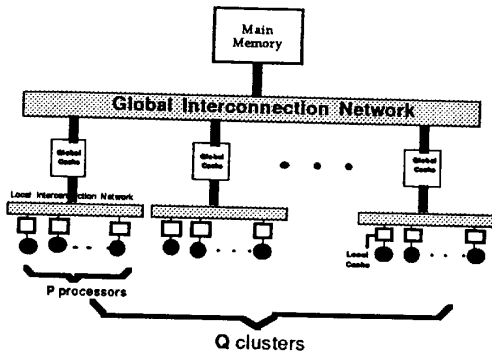


Figure 1: Hierarchical cluster-oriented shared-memory multiprocessor.

data in any local cache of a cluster, then a copy must also exist in the global cache of the cluster. Copies of data are loaded in caches on demand, i.e. at the time the data are referenced.

If a piece of data is not found in the local cache, the caches of the cluster are searched (one LIN access). If it is not in any cache of the cluster, the main memory is accessed (one GIN access). The main memory contains one copy of every addressable data; however the copy in main memory may not be up-to-date, in which case the global and local caches of remote clusters are searched (another LIN access). Once the data is found, a copy is brought to the global cache of the requesting cluster, and any further reference to that particular data by the processors belonging to the same cluster is serviced without accessing the main memory. Whenever a piece of data is updated in a local cache, all of its copies in the local caches of its cluster and in the caches of other clusters must be invalidated. For all algorithms, unless otherwise mentioned, we assume that all data are in the main memory and that all caches are empty at the start of the algorithm.

To avoid communication bottlenecks, the bandwidth of the GIN must be able to support the inter-cluster traffic without saturating; similarly, the bandwidth of each LIN must be sufficient for the intracluster traffic. We say that the bandwidth requirements of the target algorithms must match the available bandwidths at the global and local levels of the architecture.

We will assume that all the LINs and the GINs have exactly the same bandwidth (as in [10]). Therefore the total bandwidth available for intracluster communication is  $Q$  times the total bandwidth for intercluster communication.

### 3 Basic Communication Algorithms

This section is dedicated to the analysis of complexities for a few basic communication algorithms including broadcasting and accumulation. These algorithms are at the core of many important numerical algorithms [1]. For all the algorithms in this Section we assume that copies of the values to broadcast or to

accumulate are already in cache before the algorithm starts.

#### 3.1 Single Node Broadcast:

A single node broadcast consists in communicating the value of a variable from a given processor (source PE) to all other processors (destination PEs) [1].

**Lemma 3.1** *Given  $N = PQ$  nodes distributed over  $PQ$  processors such that there is one node per processor, a single node broadcast can be achieved in  $O(PQ)$  intracluster communication steps and  $O(Q)$  intercluster communication steps in HSM-2.  $\square$*

#### 3.2 Multinode Broadcast:

Multinode broadcast is a generalized version of a single node broadcast simultaneously initiated by every node. The algorithm for this task is similar to single node broadcast, but it is simultaneously executed by every PE [1].

**Proposition 3.1** *For an  $N = PQ$  node graph, multinode broadcast can be carried out in  $O(PQ)^2$  intracluster communication steps and  $O(PQ^2)$  intercluster communication steps in HSM-2.  $\square$*

#### 3.3 Single Node Accumulation:

Single node accumulation is defined as transmitting the value of variable from every PE to one particular PE in the system [1]. During transmission, values are combined. A single node accumulation is required, for example, in the algorithm that adds  $QP$  numbers each stored in a different PE's memory. There are  $Q(P-1)$  source PEs and only one destination PE. This task is the reverse of single node broadcast, and has same complexity.

**Lemma 3.2** *Given  $N = PQ$  nodes, distributed over  $PQ$  processors such that there is one node per processor, single node accumulation can be achieved in  $O(PQ)$  intracluster communication steps and  $O(Q)$  intercluster communication steps in HSM-2.  $\square$*

#### 3.4 Multinode Accumulation:

Multinode accumulation can be seen as the reverse of multinode broadcast or as single node accumulation initiated by every PE in the system [1]. The algorithm for this task is similar to single node accumulation, however, it is simultaneously executed by every PE.

**Proposition 3.2** *For an  $N = PQ$  node graph, multinode accumulation can be carried out in  $O(PQ)^2$  intracluster communication steps and  $O(PQ^2)$  intercluster communication steps in HSM-2.  $\square$*

## 4 Numerical Algorithms

This section deals with several numerical algorithms, including FFT, and their implementation on an HSM-2. The bandwidth requirements for inter- and intracluster communication are derived for the following algorithms: matrix-vector multiplication, matrix-matrix multiplication, Gaussian elimination, QR decomposition, LU decomposition, and FFT. The complexity results are derived for fine-grain data mapping

only, i.e. one data element per processor. However, the results of the analysis based on these implementations are also valid for coarse-grain data mappings, i.e. one block of data per processor.

#### 4.1 Matrix-Vector Multiplication:

We consider the calculation of a matrix-vector product  $Ax$  on HSM-2 with  $PQ$  processors, where  $A$  is an  $N \times N$  matrix and  $x$  is an  $N$ -dimensional vector. In this algorithm, all data are read-only, except for the result. This problem does not map well on an HSM because there is no temporal locality of accesses to the matrix elements.

##### Case - 1: $PQ = N$

**Lemma 4.1** Given an  $N \times N$  matrix and  $N$ -dimensional vector, such that  $PQ = N$ , a matrix-vector product can be accomplished in  $O(PQN)$  intracluster and  $O(PQN)$  intercluster communication steps in HSM-2.  $\square$

##### Case - 2: $PQ = N^2$

**Lemma 4.2** Given an  $N \times N$  matrix and an  $N$ -dimensional vector, such that  $PQ = N^2$ , a matrix-vector product can be accomplished in  $O(PQ)$  intracluster and  $O(PQ)$  intercluster communication steps in HSM-2.  $\square$

#### 4.2 Matrix-Matrix Multiplication:

##### Case - 1: $PQ = N$

**Lemma 4.3** Matrix-matrix multiplication of two  $N \times N$  matrices can be achieved in  $O(PQN^2)$  intracluster and  $O(PQN)$  intercluster communication steps in HSM-2, where  $PQ = N$ .  $\square$

##### Case - 2: $PQ = N^2$

**Lemma 4.4** Matrix-matrix multiplication of two  $N \times N$  matrices can be achieved in  $O(PQN)$  intracluster and  $O(PQN)$  intercluster communication steps in HSM-2, where  $PQ = N^2$ .  $\square$

#### 4.3 Fast Fourier Transform (FFT):

FFT is an algorithm for computing the Discrete Fourier Transform (DFT) [7, 2]. Two algorithms for the one dimensional FFT of  $N$  data items are considered, the non-shuffling and the shuffling FFTs [11]. In both cases the vector of  $N$  data points is divided into  $PQ$  chunks containing  $N/PQ$  consecutive items.

##### 4.3.1 Non-shuffling FFT

The non-shuffling FFT for  $N$  data points can be represented by a butterfly graph with  $\log_2 N$  stages.

**Lemma 4.5** Non-shuffling FFT of  $N > PQ$  points in HSM-2 can be computed in  $O(PQ \log_2 \bar{P})$  intracluster, and  $O(PQ \log_2 Q)$  intercluster communication steps.  $\square$

##### 4.3.2 Shuffling FFT

In the shuffling FFT, computations of partial FFTs alternate with shuffling stages in which data are passed among processors. The total number of butterfly/shuffling stages is:  $2 \lceil \frac{\log_2 N}{\log_2 \frac{N}{PQ}} \rceil$ . We only consider one butterfly/shuffle stage in the algorithm.

**Lemma 4.6** Shuffling FFT of  $N \geq PQ$  points in HSM-2 can be computed in  $O(PQ \log_2 P)$  intracluster, and  $O(N)$  intercluster communication steps.  $\square$

#### 4.4 Gaussian Elimination:

Gaussian elimination is a method to solve a system of linear equations, for example,  $Ax = b$ , where  $A$  is the coefficient matrix and  $x$  and  $b$  are vectors. In Gaussian elimination, matrix  $A$  is first reduced to upper triangular form using forward-elimination and then the system is solved using back-substitution. Both back-substitution and forward-elimination are equally complex computation-wise and also have similar communication patterns [9].

**Lemma 4.7** Given a system of  $N$  linear equations of  $N$  variables, Gaussian elimination can be accomplished in  $O(PQ)^2$  intracluster communication and  $O(PQ^2)$  intercluster communication steps in HSM-2, such that  $PQ = N$ .  $\square$

#### 4.5 QR Decomposition:

QR decomposition is another method to solve systems of linear equations. To be applicable, the columns of  $A$  must be independent. To solve the system  $Ax = b$ , matrix  $A$  is transformed into two matrices  $Q$  and  $R$ , such that  $A = QR$  where  $R$  is invertible and  $Q$  is orthogonal. After this transformation, the system can be written as  $QRx = b$ . In order to solve the system, we have to solve  $Rx = Q^t b$  ( $Q^{-1} = Q^t$  since  $Q$  is an orthonormal matrix).

**Lemma 4.8** Given a system of  $N$  equations of  $N$  variables such that  $PQ = N$ , its solution can be evaluated using the QR decomposition method in  $O(PQ)^2$  intracluster and  $O(PQ^2)$  intercluster communication steps in HSM-2.  $\square$

#### 4.6 LU Decomposition:

LU decomposition consists of decomposing the coefficient matrix  $A$  into  $L$  and  $U$  matrices. The split of matrix  $A$  into matrices  $L$  and  $U$  is equivalent to forward-elimination [9]. Once  $L$  and  $U$  are found, two triangular systems can be solved by back-substitution—as described earlier.

**Lemma 4.9** A system of  $N$  linear equations with  $m$  variables can be solved in  $O(PQ)$  intercluster and  $O(PQ^2)$  intracluster communication steps using LU decomposition in HSM-2.  $\square$

Problem	Intercluster	Intracuster	Ratio Intercluster Intracuster
Single Node Broadcast	$O(Q)$	$O(PQ)$	$O(P)$
Multi Node Broadcast	$O(PQ^2)$	$O(PQ)^2$	$O(P)$
Single Node Accumulation	$O(Q)$	$O(PQ)$	$O(P)$
Multi Node Accumulation	$O(PQ^2)$	$O(PQ)^2$	$O(P)$

Interprocessor Communication Algorithms

Problem	Intercluster	Intracuster	Ratio Intercluster Intracuster
Matrix - Vector Multiplication	$PQ=N$	$O(PQN)$	$O(1)$
	$PQ=N^2$	$O(QN)$	$O(P)$
Matrix - Matrix Multiplication	$PQ=N$	$O(QN^2)$	$O(P)$
	$PQ=N^2$	$O(PQN)$	$O(1)$
Gaussian Elimination	$O(PQ^2)$	$O(PQ)^2$	$O(P)$
QR Decomposition	$O(PQ^2)$	$O(PQ)^2$	$O(P)$
LU Decomposition	$O(PQ^2)$	$O(PQ)^2$	$O(P)$
Fast Fourier Transform	Non-Shuffling	$O(N \log Q)$	$\frac{O(\log P)}{O(\log Q)}$
	Shuffling	$O(N)$	$O(\log P)$

Numerical Algorithms

Table 1: Comparison of intracuster and intercluster communication complexities.

## 5 Conclusions

The results obtained in this paper are summarized in Table 1. For many algorithms analyzed in this paper, the amount of intercluster communication is  $1/P$ th of the amount of communication in each cluster. Therefore, up to  $P$  clusters can be connected resulting in a system of  $P^2$  processors. Since the arguments developed in this paper can be applied to any number of levels, an HSM-i with approximately  $P^i$  processors would be effective for that class of algorithms.

Unfortunately, there are significant algorithms in the table for which the amount of intercluster traffic is  $Q$  times the amount of traffic in each cluster. This implies that the interconnection among  $Q$  clusters in an HSM needs  $Q$  times the bandwidth of the interconnection among processors within each cluster. Algorithms exhibiting no temporal locality of data accesses—such as matrix-matrix multiplication— or algorithms with point to point communication— such as FFTs— fall in this later category.

Whether this is a requirement for the algorithm or for the problem, is an open question. More work should be devoted to the design of algorithms for HSMs if such architectures are going to be viable.

## References

- [1] Dimitri P. Bertsekas and John N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice Hall, 1989.
- [2] Laxmi N. Bhuyan and Dharma P. Agarwal, *Performance Analysis of FFT Algorithms on Multiprocessor Systems*, IEEE Transactions on Software Engineering, July 1983, pp. 512-521.
- [3] Steve Chen, *Multiprocessing Linear Algebra Algorithms on the CRAY-XMP2: Experiences with small Granularity*, Journal of Parallel and Distributed Computing, August 1984.
- [4] Erik Hagersten, Seif Haridi, and David H. D. Warren, *The Cache Coherence Protocol of the Data Diffusion Machine*, in Cache and Interconnect Architecture in Multiprocessors, Michel Dubois and Shreekanth Thakkar, editors, Kluwer Academic Publishers, 1990.
- [5] Ashfaq Khokhar and Michel Dubois, *Maching Algorithms and Architectures in Hierarchical-Shared Memory Multiprocessor Systems*, Tech Report # 92-1, University of Southern California, Los Angeles, Dec. 1991.
- [6] J. Konicek et al., *The Organization of the Cedar System*, Proceedings of International Conference on Parallel Processing, 1991, pp. I-49 - I-56.
- [7] R. N. Mahapatra, V. Ashok Kumar, and B. N. Chatterji, *Performance of Parallel FFT Algorithm on Multiprocessors*, International Conference on Parallel Processing, 1990, vol. III, pp. 368-369.
- [8] W. Miranker, *A Survey of Parallelism in Numerical Analysis*, SIAM, October 1971.
- [9] Gilbert Strang, *Linear Algebra and its Applications*, Academic Press, 1988.
- [10] Andrew W. Wilson Jr., *Hierarchical Cache/Bus Architecture for Shared Memory Multiprocessors*, The 14th International Symposium on Computer Architecture, 1987, pp. 244-252.
- [11] Digital Signal Processing Committee, *Programs for Digital Signal Processing*, IEEE Press, 1979.