Matching Algorithms And Architecture In Hierarchical Shared-Memory Multiprocessor (HSM) Systems

Ashfaq Khokhar and Michel Dubois

CENG Technical Report 92-01

Department of Electrical Engineering – Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4475

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

Abstract

In this paper, an analysis of memory traffic at various levels of the hierarchy in a broad class of memory-coherent hierarchical shared-memory multiprocessor systems is presented. A parallel algorithm is said to be "well-matched" to the hierarchical architecture if the memory traffic 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.

Several interprocessor communication and linear algebra algorithms are mapped 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 novel VLSI techniques, massively parallel processing systems with thousands of processing elements are feasible now. 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 is that the programmer's model of a shared-memory machine is limited to a set of threads sharing a common memory. Shared-memory systems must efficiently support parallel multithreaded applications.

One major problem with large-scale shared memory multiprocessors is the difficulty of maintaining the UMA (Uniform Memory Access) property in hardware; UMA implies that all accesses to shared-memory have the same delay, regardless of the requesting processor, or the memory location accessed. The reason is that the distance–and therefore the communication delay–between any couple of processors inevitably increases with the number of processors in the system.

One way to reduce the impact of the interprocessor communication overhead is to associate a cache with each processor. Since most of the processor's shared-memory accesses are completed by the cache, the delay of accessing shared memory is limited to cache misses, which are quite rare. Nonetheless, if processors are very fast the miss delay can be intolerable in very large-scale shared-memory systems.

In NUMA (Non-Uniform Memory Access) systems, some cache(s) are faster to reach from any one processor than other caches. One well-known way to achieve this is through hierarchical clustering of processors, to produce a hierarchy of memory access delays.

Several shared memory systems with large number of processors are being designed using hierarchy of processors in clusters. Examples include the CEDAR [8], the Data Diffusion Machine [5], and the Encore Gigamax [13]. 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, $i = 1, 2, \cdots$.

The question addressed in this paper is whether multithreaded algorithms can be mapped effectively on an HSM-i, $i = 1, 2, \cdots$. 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).

The algorithms are carefully mapped onto the architecture and their communication requirements at different levels of the hierarchy are evaluated. A parallel algorithm is said to be "well-matched" to the hierarchical architecture if the memory traffic 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 Sections 3 and 4, an analysis of the communication complexities of the basic communication algorithms and that of linear algebra algorithms is presented. We conclude in Section 5 with a discussion of our results.

2 Hierarchical Shared-memory Multiprocessor (HSM) System

This section contains a detailed description of the HSM system considered in the paper. In general, an HSM system may have i levels; an i-level HSM is denoted by HSM-i. We will 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.

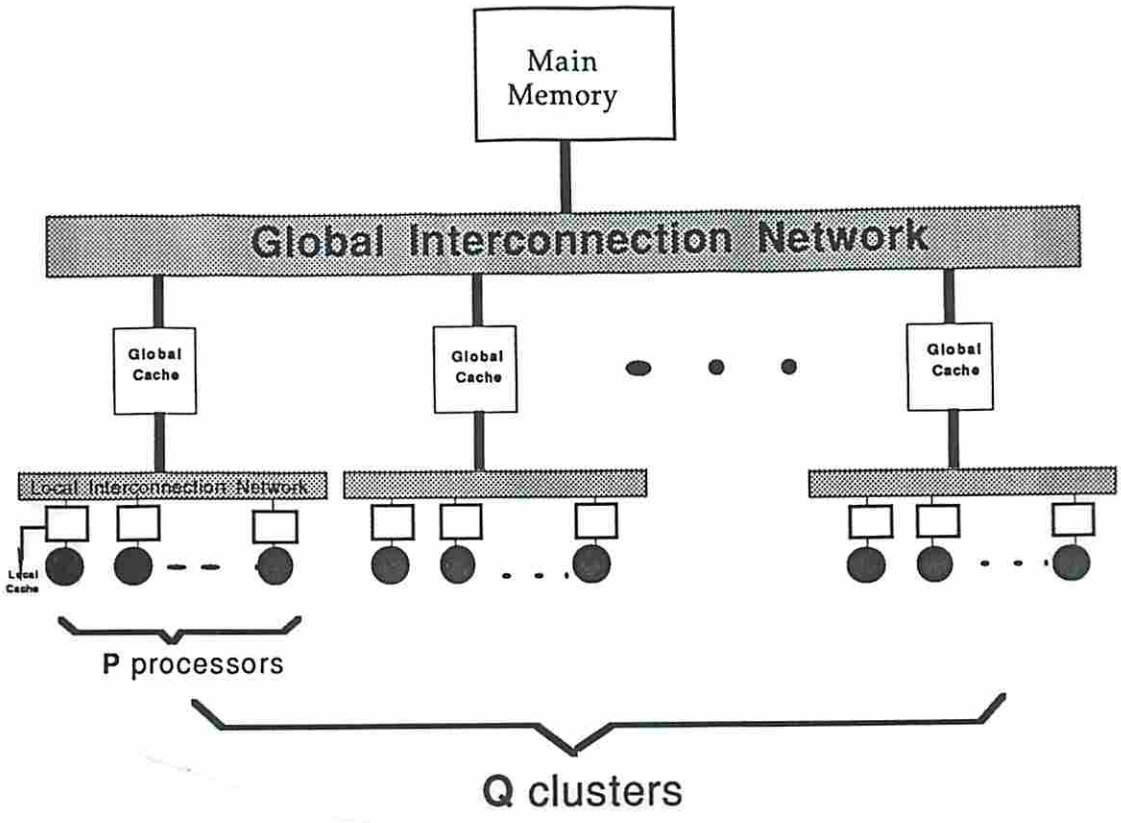

Figure 1: Hierarchical cluster-oriented shared-memory multiprocessor.

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 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 intercluster 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.

If the bandwidth of each interconnection must increase with the level in the hierarchy, large scale multiprocessors with many levels will be impossible to build. On the other hand, if the bandwidth of each interconnection remains constant at all levels the number of levels can be unlimited. We will assume that all the LINs and the GINs have exactly the same bandwidth (as in [13]). 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 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.* □

Algorithm:

1. The (P-1) PEs in the same cluster read the variable by accessing the local cache of the source PE through the bus.
2. One PE from all other clusters reads the data from the local cache of the source PE, a copy is loaded in the global cache of its own cluster.
3. All the PEs read the desired variable from their respective global cache.

Communication Complexity:

- Total number of intracluster transfers =
  $(P-1)$ – in step 1
  $+ (Q-1)$ – in step 2
  $+ P(Q-1)$ –in step 3
- Total number of intercluster transfers = $(Q-1)$ – in the second step.
- Ratio of intracluster vs intercluster communication = $O(P)$.

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.* □

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, therefore, has same complexity. However the algorithm is different.

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.* □

Algorithm:

1. In each cluster, the values are combined and the result is stored in the global cache of each cluster (resulting in PQ intracluster transfers).
2. The values in the Q global caches are combined in the destination PE (resulting in $(Q-1)$ intercluster and $2(Q-1)$ intracluster transfers).

Communication Complexity:

- Total number of intracluster transfers =
  $+ PQ$ - in step 1
  $+ 2(Q-1)$ - in step 2

- Total number of intercluster transfers = $(Q-1)$ - in the second step.
- Ratio of intracluster vs intercluster communication = $O(P)$.

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.* □

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. We consider implementations for fine-grain data mapping, i.e. one data element per processor. However, the results of the analysis based on this implementation apply directly to 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 on 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.* □

Algorithm:
Each processor computes a component of the resultant vector.

1. A processor in each cluster reads the vector from the main memory.
2. All other processors in a cluster read the vector from the global cache of the cluster.
3. Each processor reads one row of matrix A from the main memory.
4. Each processor computes its component.

Communication Complexity :

- Total number of intracluster transfers =
  QN – in step 1
  $+ Q(P-1)N$ – in step 2
  $+ PQN$ – in step 3
- Total number of intercluster transfers =
  QN – in step 1
  $+ PQN$ – in step 3
- Ratio of intracluster vs intercluster communication: $O(1)$.

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.* □

Algorithm: (assume $P = Q = N$)

1. A processor i in each cluster reads element i of the vector from the main memory, $0 < i \leq P$.
2. Each processor i in cluster j reads the (i,j)th element of the matrix A, $0 < i \leq P$ and $0 < j \leq Q$.
3. Each processor computes its product.
4. One processor from every cluster computes the sum of the partial products over its cluster.

Communication Complexity :

- Total number of intracluster transfers =
  QN – in step 1
  $+ N^2$ – in step 2
  $+ Q(P-1)$ – in step 4

- Total number of intercluster transfers =
  QN – in step 1
  $+ N^2$ – in step 2
- Ratio of intracluster vs intercluster communication: $O(1)$.

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 $PQ = N$.* □

In the following algorithm, for the product of two matrices A and B of size $N \times N$, each processor computes one row of the resultant matrix C.

Algorithm:

1. A processor in each cluster reads the matrix B from the main memory.
2. Other processors in each cluster load a copy of the matrix B.
3. Each processor i in cluster j accesses the row i of the matrix A, where $0 < i \leq P$ and $0 < j \leq Q$.
4. Each processor i computes its respective row of the resultant matrix C, $0 < i \leq PQ$.

Communication Complexity :

- Total number of intracluster transfers =
  PQN^2 – in steps 1 and 2
  $+ PQN$ – in step 3
- Total number of intercluster transfers =
  QN^2 – in step 1
  $+ PQN$ – in step 3

- Ratio of intracluster vs intercluster communication = $O(P)$

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 $PQ = N^2$.* □

In the following algorithm, for the product of two matrices A and B of size $N \times N$, each processor computes one element of a row of the resultant matrix C.

Algorithm:

1. A processor in cluster i reads row i of the matrix A from the main memory, $0 < i \leq Q$.
2. Other processors in the cluster load a copy of the row.
3. Each processor i in cluster j accesses the column i of the matrix B, where $0 < i \leq P$ and $0 < j \leq Q$.
4. Processor i in cluster j computes $C(i, j)$, where $0 < i \leq P$ and $0 < j \leq Q$.

Communication Complexity :

- Total number of intracluster transfers =
  PQN – in steps 1 and 2
  $+ PQN$ – in step 3
- Total number of intercluster transfers =
  QN – in step 1
  $+ PQN$ – in step 3
- Ratio of intracluster vs intercluster communication = $O(1)$

4.3 Fast Fourier Transform (FFT):

FFT is an algorithm for computing the Discrete Fourier Transform (DFT) [10, 2].

Definition 1:

The DFT of a sequence $a_0, \ldots, a_{n-1}$ is the sequence $b_0, \ldots, b_{n-1}$, where for $0 \leq j < N$

$$b_j = \sum_{i=0}^{n-1} a_i w^{ij}$$

Here, w is a primitive N^{th} root of unity. That is, $w^n = 1$, but $w^i \neq 1$. Two algorithms for the one dimensional FFT of N data items are considered, the non-shuffling and the shuffling FFTs [14]. 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. The graph is shown in Fig. 2.

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

Algorithm:

1. Each processor computes the FFT for its chunk of N/PQ points in $\log_2 \frac{N}{PQ}$ butterfly stages.
2. Each cluster computes the FFT for its chunk of N/Q points in $\log_2 P$ stages (intracluster communication only).
3. Finally the FFT of N points is computed, using the partial results from each cluster, in $\log_2 Q$ butterfly stages.

Communication Complexity:

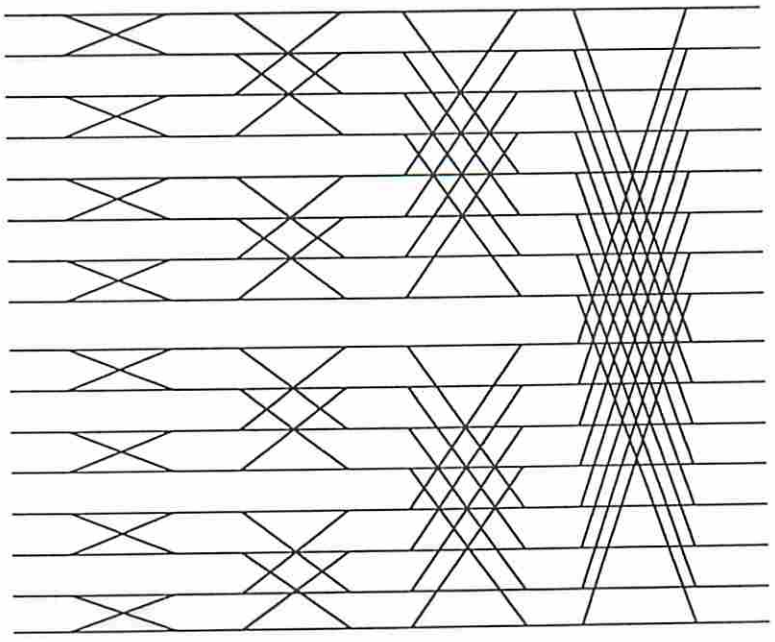

Figure 2: Data flow diagram of a 16 point non-shuffling FFT algorithm.

- Total number of intracluster transfers =
  N – in step 1
  $+ (\frac{N}{Q} \log_2 P)Q$ – in step 2
  $+ N \log_2 Q$ – in step 3.
- Total number of intercluster transfers =
  N – in step 1
  $+ N \log_2 Q$ – in step 3.
- Ratio of intracluster vs intercluster communication = $O(\frac{\log P}{\log Q})$.

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: $2\lceil \frac{\log_2 N}{\log_2 \frac{N}{PQ}} \rceil$. The data flow graph is shown in Fig. 3. 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.* □

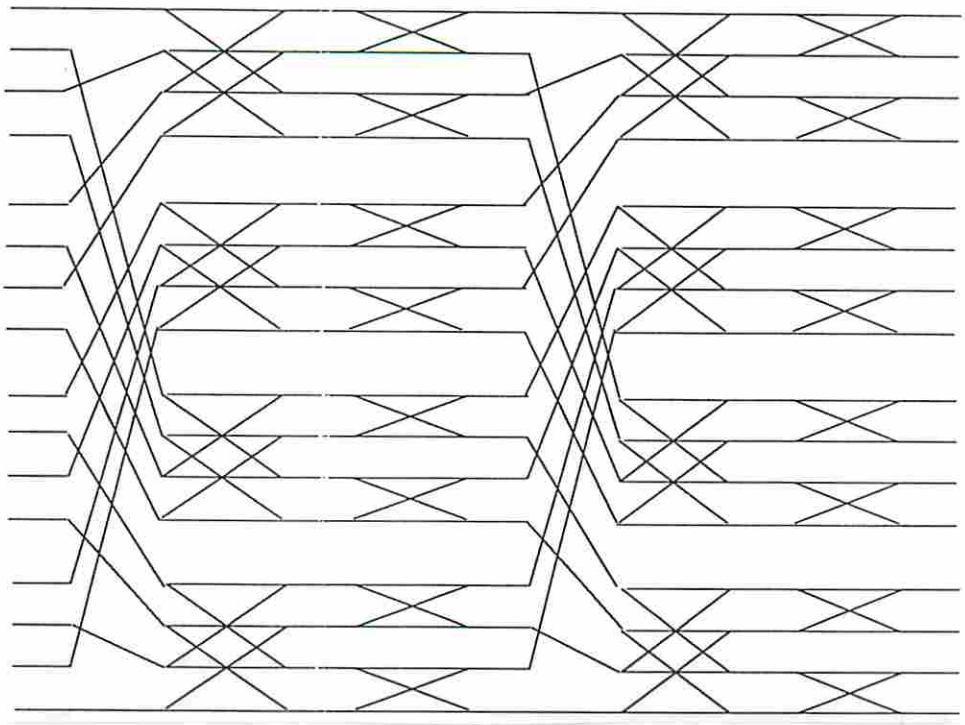

Figure 3: Date flow diagram of a 16 point shuffling FFT algorithm.

Algorithm: (for one butterfly/shuffle stage)

1. Compute partial FFT in each processor ($\log_2 \frac{N}{PQ}$ steps).
2. Compute partial FFT in each cluster ($\log_2 P$ steps).
3. Shuffle across the clusters (1 step).

Communication Complexity:

- Total number of intracluster transfers =
  N – in step 1
  $+ (\frac{N}{Q} \log_2 P)Q$ – in step 2
  $+ N$ – in step 3.
- Total number of intercluster transfers =
  N – in step 1
  $+ N$ – in step 3.
- Ratio of intracluster vs intercluster communication = $O(\log_2 P)$.

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 is solved using back-substitution. Therefore, this procedure comprises of two steps: forward-elimination, and back-substitution. Both back-substitution and forward-elimination are equally complex computation-wise and also have similar communication patterns [12]. We will analyze communication complexities for back-substitution only. Each processor is assigned to compute one variable.

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 $PQ = N$.* □

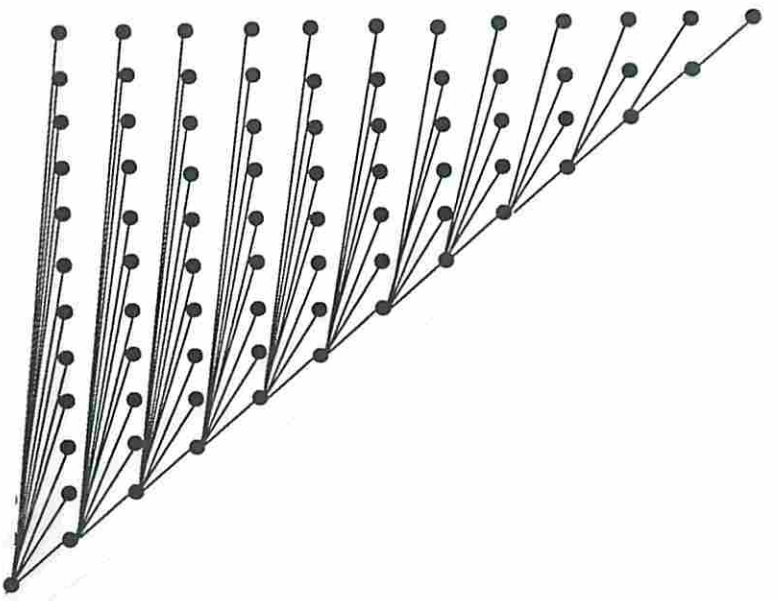

Figure 4: Data flow graph for Back-substitution.

Algorithm:

1. During iteration i each processor tries to evaluate its assigned variable. If the processor is successful, it broadcasts the value to the remaining $N - i$ processors who are waiting on this value.

2. Step 1 is repeated $N - 1$ times.

Communication Complexity:

In an upper triangular system, the value of the Nth variable of an N variable equation system is available at the beginning. This value is broadcast to all other processors. Then in each step, the processor which is able to evaluate its variable using the broadcast value, computes the value and then broadcast it to the the remaining processors. This procedure is repeated N times and after N iterations the system is solved. The data flow graph is given in Fig. 4.

- Total number of intracluster transfers = $Q^2 \sum_{i=1}^{P-1} P - (P - i) = O(PQ)^2$.

- Total number of intercluster transfers = $(P - 1) \sum_{i=1}^{Q-1} (Q - i) = O(PQ^2)$.

- Ratio of interacluster vs intercluster communication = $O(P)$.

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 $Rx = Q^t b$. As Q is an orthonormal matrix hence, $Q^{-1} = Q^t$.

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

Algorithm:

1. Convert A into Q and R matrices.
2. Compute the transpose for matrix Q
   Matrix Transpose:
   (At the end of the first phase, processor i contains row i of matrix Q. The transpose of matrix Q is being stored in matrix B.)
   (a) Processor i reads element i from the local cache of processor j and stores it at $B(i,j)$, $0 < i \leq PQ$, $0 < j \leq PQ$.
3. Multiply Q^t i.e. matrix B by vector b.
4. Solve the simplified system using back-substitution.

Communication Complexity:
The conversion of matrix A into Q and R is simultaneous and is achieved using the Gram-Schmidt process. The Gram-Schmidt process is equivalent to back-substitution as described in [12].

- Total number of intracluster transfers =
  $O(PQ)^2$ – back-substitution in step 1
  $+ PQ(P-1)$ – in step 2 (matrix transpose)
  $+O(PQ)$ – matrix-vector multiplication in step 3
  $+ O(PQ)^2$ – back-substitution in step 4.
- Total number of intercluster transfers =
  $O(PQ^2)$ – back-substitution in step 1
  $+ PQ(N-P)$ in step 2 (matrix transpose)
  $+ O(PQ)$ – matrix-vector multiplication in step 3
  $+ O(PQ^2)$ – back-substitution in step 4.
- Ratio of intracluster vs intercluster communication = $O(1)$.

4.6 LU Decomposition:

LU decomposition consists of decomposing coefficient matrix A into L and U matrices. The system $Ax = b$ is split into two triangular systems:

$$\text{first into } Lc = b, \text{ and then into } Ux = c$$

The split of matrix A into matrices L and U is equivalent to forward-elimination [12]. 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 $O(PQ)^2$ intercluster and $O(PQ^2)$ intracluster communication steps using LU decomposition in HSM-2.* □

Algorithm:

1. Transform matrix A into L and U matrices using forward-elimination.
2. Solve the triangular systems using back-substitution.

Communication Complexity:
LU decomposition is equivalent to Gaussian Elimination in terms of its complexity. All the steps involved in carrying out LU decomposition have already been described earlier in the analysis of the Gaussian Elimination algorithm.

- Total number of intracluster transfers =
  $O(PQ)^2$ – forward-elimination in step 1
  $+ O(PQ)^2$ – back-substitution in step 2.
- Total number of intercluster transfers =
  $O(PQ^2)$ – forward-elimination in step 1
  $O(PQ^2)$ – back-substitution in step 2.
- Ratio of intracluster vs intercluster communication = $O(P)$.

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. The effectiveness of such an architecture is obvious from the analysis: one can use as many as Q^i processors with i levels of hierarchy with the same bandwidth for the network connecting clusters as for the network within a clusters without any bottleneck problem.

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.

[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] Beauregard Fraleigh, *Linear Algebra*, Edison Wesley.

[5] 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 Shreekant Thakkar, editors, Kluwer Academic Publishers, 1990.

[6] Kai Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, McGraw Hill, pp. 370-372.

[7] Ju-wook Jang and Wojtek Przytula, *Trade-offs in mapping FFT computations onto fixed size mesh processor arrays*, International Parallel Processing Symposium, 1991.

[8] J. Konicek at al., *The Organization of the Cedar System,* Proceedings of International Conference on Parallel Processing, 1991, pp. I-49 - I-56.

[9] David Kuck, Duncan Lawrie, and Ahmed Sameh, *High Speed Computer and Algorithm Organization*, Academic Press, Inc. 1977.

[10] R. N. Mahapatra, V. Ashok Kumar, and B. N. Chatterji, *Performance of Parallel FFT Algorithm on Multiprocessors*, Inetnational Conference on Parallel Processing, 1990, vol. III, pp. 368-369.

[11] W. Miranker, *A Survey of Parallelism in Numerical Analysis*, SIAM, October 1971.

[12] Gilbert Strang, *Linear Algebra and its Applications*, Academic Press, Inc. 1988.

[13] Andrew W. Wilson Jr., *Hierarchical Cache/Bus Architecture for Shared Memory Multiprocessors*, The 14th International Symposium on Computer Architecture, 1987, pp. 244-252.

[14] Digital Signal Processing Committee, *Programs for Digital Signal Processing*, IEEE Press, 1979.

Problem	Intercluster	Intracluster	Ratio $\frac{\text{Intracluster}}{\text{Intercluster}}$
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 Communicatioon Algorithms

Problem		Intercluster	Intracluster	Ratio $\frac{\text{Intracluster}}{\text{Intercluster}}$
Matrix - Vector Multiplication	$PQ=N$	O(PQN)	O(PQN)	O(1)
	$PQ=N^2$	O(QN)	O(PQN)	O(P)
Matrix - Matrix Multiplication	$PQ=N$	$O(QN^2)$	$O(PQN^2)$	O(P)
	$PQ=N^2$	O(PQN)	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(Nlog Q)	O(Nlog P)	$\frac{O(\log P)}{O(\log P)}$
	Shuffling	O(N)	O(Nlog P)	O(log P)

Numerical Algorithms

Table 1: Comparison of intracluster and intercluster communication complexities.