A Review of Matrix multiplication in Multicore Processor using Interconnection Network

ChandrikaKarmakar
Research scholor
CCEMRaipur, Chhattisgarh, India
Email:chandrika4190@gmail.com

Prof. Dr.P.Udaya Kumar
CCEM Raipur, Chhattisgarh, India
Email:uday.uday08@gmail.com

Abstract

One of the most important constraints of today's architectures for data-intensive applications is the limited bandwidth due to the memory-processor communication bottleneck. This significantly impacts performance and energy. Interconnection network is an important component of a parallel computer. Hypercube Interconnection network provides a strong reliable network among processors. Interconnection networks play a central role in determining the overall performance of the multiprocessor systems. The interconnection network is placed between various devices in the multiprocessor network. Parallel machines break a single problem down into parallel tasks that are performed concurrently, reducing significantly the application processing time.The estimation of time taken of matrix multiplication by hypercube interconnection network in multi-core processors will be calculated. For achieving parallelism OMP parallel programming model which performs parallelism in shared memory environment will be used.

Keywords— *Interconnection network, Hypercube Interconnection, Matrix multiplication, parallel computing.*

I. INTRODUCTION

Interconnection network is an important component of a parallel computer. Interconnection networks are used to interconnect various processors and memories with each other.In Parallel Execution, execute the program more than one task, with each task being able to execute the same or different statement at the same time. The estimation of time taken of matrix multiplication by hypercube interconnection network in multi-core processors will be calculated. The Hypercube Interconnection network provides a strong reliable network among processors. In hypercube, each node/processor is arranged in the form of cube with binary address.

Matrix multiplication is a key function of many data intensive applications such as data recognition, data mining, The designs of high-performance processor architectures are moving toward the integration of a large number of multiple processing cores on a single chip. The performance scalability of such chips requires a solid interconnection network architecture and its behaviouralevaluation should begin in the design and verification stage.

Matrix multiplication is such a central operation in many numerical algorithms, much work has been invested in making matrix multiplication algorithms efficient. Applications of matrix multiplication in computational problems are found in many fields including scientific computing and pattern recognition and in seemingly unrelated problems such counting the paths through a graph. Many different algorithms have been designed for multiplying matrices on different types of hardware, including parallel systems, where the computational work is spread over multiple processors. The definition of matrix multiplication is that if C = AB for an n × m matrix A and an m × p matrix B, then C is an n × p matrix with entries.

$$\begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

A multi-core processor is a single computing component with two or more independent actual processing units (called "cores"), which are units that read and execute program instructions. The instructions are ordinary CPU instructions (such as add, move data, and branch), but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs amenable to parallel computing.

Multi-core processor ,used in high-performance or parallel computing as a low-latency interconnection though they could be implemented on top of a packet switching network. The improvement in performance gained by the use of a multi core processor depends very much on the software algorithms used and their implementation.

Hypercube topology consists of connections of the nodes to form cubes. The nodes are also connected to the nodes on the other cubes. This network consists of connections of the nodes to form cubes. The nodes are also connected to the nodes on the other cubes.

An n-cube network, also called hypercube, consists of $N=2^n$ nodes; n is called the dimension of the n-cube network. When the node addresses are considered as the corners of an n- dimensional cube, the network connects each node to its N neighbors. In an n -cube, individual nodes are uniquely identified by n-bit addresses ranging from 0 to N-1.

Parallel computing: In Parallel Execution, execute the program more than one task, with each task being able to execute the same or different statement at the same time. A parallel computer is a "Collection of processing elements that communicate and co-operate to solve large problems fast".

The estimation of time taken of matrix multiplication by hypercube interconnection network in multi-core processors will be calculated. Parallel computing is an evolution of serial computing that attempts to emulate what has always been the state of affairs in the natural world: many complex, interrelated events happening at the same time, yet within a sequence.

Computation:

a) To be run on a single computer having a single Central Processing Unit (CPU);

b) A problem is broken into a discrete series of instructions.

c) Instructions are executed one after another.

d) Only one instruction may execute at any moment in time.

In Parallel computers, Dual core is an example of parallel computer. These computers provide us hardware level parallelism. But when we simply work on these type of systems they work like a uni - processor system means at a time only single core or CPU are working and other core or CPU are in a idle state at that time. With the increased use of multiprocessor and multi-core systems in embedded applications, software design considerations now include methods to distribute the software functionality across these computing resources.

Multicore and multiprocessor architectures to see how they influence software design decisions and also see how different configurations of the cores and processors may affect the design. The principal focus will be on the core and processor interfaces to the memory system with particular emphasis on the on-chip cache.

II. LITERATURE SURVEY

A. Parallel Matrix Multiplication on Memristor-Based Computation-in-Memory Architecture

One of the most important constraints of today's architectures for data-intensive applications is the limited bandwidth due to the memory-processor communication bottleneck. This significantly impacts performance and energy. For instance, the energy consumption share of communication and memory. Access may exceed 80%. Recently, the concept of Computation-in- Memory (CIM) was proposed, which is based on the integration of storage and computation in the same physical location using a Crossbar topology and non-volatile resistive-switching memristor technology.

The experimental results show that, depending on the matrix size, CIM architecture exhibits several orders of magnitude higher performance in total execution time and two orders of magnitude better in total energy consumption than the multicore-based on the shared memory architecture.

Method Used: CIM architecture

Table of matrix multiplication computational complexity

Algorithm	Time complexity	Multipliers	Adders
Sequential MM	$\mathcal{O}(n^3)$	1	1
1D Parallel MM	$\mathcal{O}(n^2)$	n	n
2D Parallel MM	$\mathcal{O}(n)$	n^2	n^2
3D Parallel MM	$\mathcal{O}(log_2 n)$	n^3	n^3 - n^2

Algorithm 1: 3D Parallel Matrix Multiplication Algorithm

```
Input: A, B, _ A and B are m-by-p and p-by-n
Output: C = A · B where C is m-by-n
1: procedure MATRIX MULTIPLY(INT m, INT n, INT p)
2: for all i, j, k ∈( 1 to m, n, p ) do
3: Cijk= Aik× Bkj
4: for all i, j ∈( 1 to m, n) do
5: for s = 1 to log2 (p) do
6: for all kodd= 1 to (p/2s−1) do
7: Cijkodd= Cijkodd+ Cij(kodd+2s−1)
8: Cij= Cij1
```

Algorithm 2: Row Major-order Data Mapping Algorithm

```
Input: m, n, p _ input matrices are m-by-n and n-by-p
Output: LISTS _ two-dimensional array of coordinate lists
1: procedure MATRIXMAPPING(INT m, INT n, INT p)
2: LISTS = new vector<INT PAIR>* m * p
3: INT ORDER = ceil(log4(n))
4: for i= 0 to m − 1 do
5: for j = 0 to p − 1 do
6: RECURSIVE(LISTS[i][j], ORDER, 2*i, 2*j)
7: return LISTS
```

Conclusion: We show the capability of memristor-based CIM architecture in exploiting the maximum amount of parallel matrix multiplication algorithm while minimizing their communicationcost via our communication-efficient mapping scheme. The results clearly show that CIM architecture has the potential to outperform traditional shared-memory multicore architecture.

B. Statistical Definition of an Algorithm in PRAM Model & Analysis of 2 × 2 Matrix Multiplication in 2n Processors Using Different Networks

The mathematical bound is not sufficient to compute the time complexity of an algorithm in parallel model. As we discuss in our previous paper that there are several factor that affect the Complexity of an algorithm in parallel model. In general we define a statistical bound for parallel model and it derivedFrom the defining of statistical bound for sequential algorithm.

The PRAM model is suitable for study of statistical bound; in general, the number of processors on a real parallel architecture is limited. There are several definitions provide for the parallel algorithm complexity.

The statistical bunds for an algorithm in PRAM model. In this section we demonstrate the statistical bound by considering an algorithm of two 2 × 2 matrix multiplications (A, B) and to compute the sum of the multiplied, four elements. The resulting product matrix C = A × B, comprehend eight multiplications and seven additions.

The algorithm executed over 2n number of processors where n = 0, 1, 2, 3, 4. We are Analyzing the complexity of algorithm by compelling on different interconnections

Networks.In Parallel Random Access Machine Model (PRAM). We present a concept of statistical analysis of 2 × 2 matrix multiplication and its summation will be executed in 2n processors.It shows conceptually how a communication delays taking an important role in parallel time complexity.

Method Used: PRAM models (Parallel Random Access Machine Model)

Table comparison between mathematical and statistical bound

Mathematical Bounds	Statistical Bounds
Operations are counted.	Operations are Weighted.
A separate bound is provided for a specific operation type	Weighting permits collective considerations for determining the bound
Theoretical derivable	They are conceptual
They may be unrealistic at times	Guaranteed to be realistic
They are system independent.	They can only system invariant
They are ideal for analyzing worst case behavior (as the bounds have guarantee)	They are suitable for average case
They are exact	They are exact provided only they are system invariant

Mathematical Analysis of the algorithm executed nnumber steps over p number processors and having communication delay d.

then

Tp(n) = □(n\p +d)

If communication delay is negligible then the complexity is

Tp(n)= □ (n/p)

And when n=p then Tp(n) = □(1) which is not true

we can conclude that when Interconnection networkchanges the time complexity also changed. In all cases when communication delays are vary.

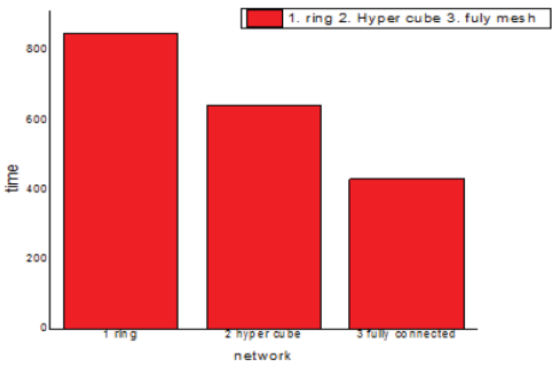


Figure 9. Comparision bar diagrams for different networks

Conclusion: In this paper we have outlined a new approach to the statistical analysis of parallel algorithm in PRAM, and defining of several concepts. It seems that when processors increase the complexity of algorithm its increase or decrease depends upon the number of communication delays.

For further research this approach can also be applied for several algorithms. The conceptual results from the PRAM models is ”worst case” results where nothing is known on the type of problem at hand; better results may be obtained with more knowledge by considering various types of PRAM models and communication networks.

C. *An Optical Multi-Mesh Hypercube: A Scalable Optical Interconnection Network for Massively Parallel Computing.*

A new interconnection network for massively parallel computing is introduced. This network is called an Optical Multi-Mesh Hypercube (OMMH) network. The OMMH integrates positive features of both hypercube (small diameter, high connectivity, symmetry, simple control and routing, fault tolerance, etc.) and mesh (constant node degree and scalability) topologies and at the same time circumvents their limitations (e.g., the lack of scalability of hypercubes, and the large diameter of meshes).

The OMMH can maintain a constant node degree regardless of the increase in the network size. In addition, the flexibility of the OMMH network makes it well suited for optical implementations. This paper presents the OMMH topology, analyzes its architectural properties and potentials for massively parallel computing, and compares it to the hypercube. Moreover, it also presents a three-dimensional optical design methodology based on free- IT has become very clear that significant improvements in computer performance in the future can only be achieved through exploitation of parallelism at all machine design levels. On the architectural side, communication among the elements of a high-performance computing system is recognized as the limiting and decisive factor indetermining the performance and cost of the system.

In recent years, there have been considerable efforts in the design of interconnection networks for parallel computers. Two of the most popular point-to-point interconnection networks for parallel computers today are the binary n-cube, also called the hypercube, and the mesh interconnection networks.

Method Used: OMMH network (Optical Multi-Mesh Hypercube)

OMMH network

with the wrap-around mesh can be described as follows:

Rule 1

ommh,,(z,j,k) = ((**i**+ 1)mod **1 , j , k**)

ommh,,(i,j,k) = ((I + i- 1)mod 1.j.k)

ommhm3(i,j,k) = (i, (j + 1)mod m , k)

ommhm4(i , jk,) = (i ,(m+ j - 1)mod m, k)

ommh cd (i , j ,5 -1 . . . kd+lkdkd-1 ' . ' k o) =

(Z , j , k,-l ' ' . kd+lkdkd-l ... ko), ford = 0,1, ..', 72- 1,

whereknPl . ' . kd+lkdk&l . . ' kois

representation of integer k.

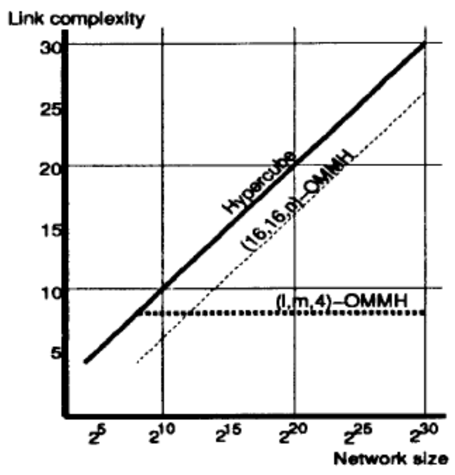
a binary

The first four interconnection functions, **ommh**, **ommh,,** ,

ommh m3, and **ommh,** , are for the four-nearest-neighbor connections

including wrap-around connections and **ommhCd,** for

d = 0, 1, . . . ,n - 1, determines the hypercube interconnection.



Conclusion: To overcome the lack of scalability in the regular hypercubenetworks, a new interconnection network topology, called anOptical Multi-Mesh Hypercube, is presented. The proposedNetwork is a combination of hypercube and mesh topologies.

The analysis and simulation results show that the new interconnection network is very scalable, meaning the configuration of the existing nodes is relatively insensitive to the growth of the network size, and more efficient in terms of communication. It is also shown that the new interconnection network is highly fault-tolerant. Any faulty node **or** link can be bypassed by only two additional hops with little modification of the fault-free routing scheme.

Due to the concurrent existence of multiple meshes and hypercubes, the new network provides a great

Architectural support for parallel processing and distributed computing. In addition, a wide body of parallel algorithms that have been designed for the hypercube and the mesh interconnection are readily implementable on the proposed network.

D. Global Commutative and Associative Reduction Operations in Faulty SlMDHypercubes.

The hypercube is a regular and symmetric structure that has proved very popular for parallel processing applications. Several hypercube based machines are commercially available which include machines from Intel, Ncube, CM-2, and Paralex. An important area of research is to obtain efficient algorithms for parallel processing applications that operate gracefully even when some of the nodes of the cube fail.

Such algorithms are very useful in missioncritical environments including medicine and space exploration.

We consider the problem of computing a global commutative and associative operation, also known as semi-group operation, (such as addition and multiplication) on a faulty hypercube. In particular, we study the problem of performing such an operation in an n-dimensional SlMD hypercube, Q, with up to *n* - **1** node and/or link faults.

In an SIMD hypercube, during a communication step, nodes can exchange information with their neighbors only across a specific dimension. d, of *n* dimensions, depending on where the faults are located.

An important and useful property of this dimension ordering is the following:

If the n-cube **IS** partitioned into k-sub cubes using the first *k* dimensions of this ordering, namely d,, d2, ... d,for any *2 2 k< n,* then each k-sub cube in the partition contains at most *k*- 1 faults. We use this result to develop algorithms for global sum. These algorithms use *3n-2, n*+ 3 log *n*+ *3* log log*n,* and *n*+ log *n*+ 4 log log*n*+ O(log loglog*n)* time steps, respectively.

An important area of research is to obtain efficient algorithms for parallel processing applications that operate gracefully even when some of the nodes of the cube fail. Such algorithms are very useful in mission critical environments including medicine and space exploration.

Computing.

A global commutative and associative operation, also known as semi-group operation, (such as addition and multiplication) on a faulty hypercube. In particular, we study the problem of performing such an operation in an n-dimensional SlMD hypercube, Q, with up to *n* - **1** node and/or link faults.

In an SIMD hypercube, during a communication step, nodes can exchange information with their neighbors only across a specific dimension. d, of *n* dimensions, depending on where the faults are located.

Method Used: SIMD hypercube

Algorithm: The dimensions of Q, can be ordered as (d,, d,, . . ., d,) suck

that for every k, 2 5 k 5 n, every subcube induced by the dimensions

(d,, d,, . .., dk) contains at most k - 1 faulty nodes.

PROOFA. ssume that the fault set F is ordered arbitrarily, as F = (f,,

f,, . . ., fn-l). We will first prove another fact using induction:

that there is an ordering (zl, *i2,* , **zn-,)** of some ***IZ*** - 2 dimensionssuch that, for every 1,1 *5 1 5 n* - 2, no two faulty nodesamong *f,,* . . ., $+, agree in their values of the]-bit vector in positions *I,,* . .,*I,*

Define the Jth dimension *z*ias follows

a) Choose *i1* to be any dimension in which *f* ,disagrees

with*f,*.

b)For*I* **t** 2, if fi+l agrees with some earlier faulty node *fi*

(I 5 J) in all the dimensions il, ***i2,*** ., *zi-*,* then choose any

dmension*t* m whchh+l disagrees mth*fi,* and let *I,* = *t.*

c) If there is no such *fi,* then choose for z, any dimension

that is not already chosen.

The base case, *I* = 2, is easy to verify. To show the inductive

step, note that when choosing *if,* if h+l disagrees with all of *f,,*

f,, ..., ***4*** m the *(1* - 1)-bit vectors induced by bit positions zl,

. , i,-,, then we can choose any dimension for ***il***that is not

already chosen, and our inductive hypothesis will be preserved.

On the other hand, if there is some earlier fi with

which*f,,,* agrees on these *1* - 1 bits, then there is at most one

such*f,,* by the inductive hypothesis, and we only need to

choose some dimension for *z*ion which ***A+,*** and *fi* disagree.

We now construct the ordering required by the theorem by

essentially using the reverse ordering of the *zIs.* We choose *d,*

and *d,* to be the two dimensions that were not chosen in the

above procedure, and for *I 2* 3, d, = in-i+l.

We report on the basic concept of multi-core processors. In this article also we survey the most important aspects of challenge in this method. However, before multi-core processors the performance increase from generation to generation was easy to see, an increase in frequency.

This model broke when the high frequencies caused processors to run at speeds that caused increased power consumption and heat dissipation at detrimental levels. Adding multiple cores within a processor gave the solution of running at lower frequencies, but added interesting new problems.

The technical problem of having multiple cores on a CPU is MEMORY management. Though CPU speed increases exponentially with multi cores upto 8 cores but the accessing of memory from DRAM and other levels of memory hierarchy could not match up with processor speed resulting in MEMORY WALL which is greatest overhead to performance.

References

[1] AdibHaron, Jintao Yu, RazvanNane, MottaqiallahTaouil, Said Hamdioui, Koen Bertels "Parallel Matrix Multiplication on Memristor Based Computation-in-Memory Architecture" 978-1-5090-2088-1/16/$31.00 ©2016 IEEE. PP.759-766.

[2]S.K.Panigrah, and S.Chakraborty, 2Statistical Definition of an Algorithm in PRAM Model & Analysis of 2 × 2 Matrix Multiplication in 2n Processors Using Different Network Souvenir of the 2014 IEEE International Advance computing Conference, IACC 2014, pp. 717-724.

[3]C. S. Raghavendra and M. A. Sridhar, "Global commutative and associative reduction operations in faulty SIMD hypercubes," in IEEE Transactions on Computers, vol. 45, pp. 495-498, April 1996.

[4]J. Zhou, Z. Wu, S. Yang, and K. Yuan, "Symmetric property and reliability of balanced hypercube," IEEE Transactions on Computers, vol. 64(3), 2013, pp. 876–881.

[5] R. A. Van De Geijn and J. Watts, "SUMMA: Scalable universal matrix multiplication algorithm," *CCPE*, pp. 255–274, 1997.

[6]J Bruck, R Cypher, and D Soroker, "Tolerating Faults in HypercubesUsmgSubcubeParthoning," IEEE Trans Computers, vol 41,no 5, pp 599-605, May 1992.

[7]B Becker and H U Simon, "How Robust Is the n-Cube?," Information and computation, vol 77, pp 162-178,1988.

[8]P Ramanathan and K G Shin, "Reliable Broadcasting in Hypercube Multicomputers," IEEE Trans Computers, vol 37, no 12, pp1,654-1,657, Dec 1988.