COMMUNICATION COST OF A MATRIX PRODUCT ON SUPER-HYPERCUBE ARCHITECTURE

Maryam Amiripour and Hamid Abachi
Department of Electrical and Computer Systems Engineering
Monash University
Australia
E-mail: maryam.amiripour@eng.monash.edu.au

KEYWORDS
Communication Cost, Hypercube, Super Hypercube, Dynastatic, Matrix Product

ABSTRACT

Processor allocation and the task scheduling technique in parallel processing systems play a significant role in improving the performance of a message-passing architecture. Adapting the right algorithms and further improvements in areas such as time complexity, execution time, speed up and synchronization mechanisms undoubtedly facilitates implementation of advanced applications on a parallel processing system. These applications include but are not limited to DNA computing, artificial immune systems and optical computing to name a few. This paper highlights the communication cost related to a Super-Hypercube topology for being a subclass of traditional Hypercube architecture.

Furthermore, a particular reference is made to the mathematical modeling of Hypercube and Super-Hypercube architectures. Finally, graphical presentations are carried out based on mathematical calculations to address the advantage of Super-Hypercube topology.

INTRODUCTION

Parallel processing systems are commonly applied in areas such as military, space, signal processing, image processing and pattern recognition that require high computational power. These parallel processing systems could be implemented to solve many engineering problems that suffer from lack of high reliability, performance, flexibility and compatibility, availability, portability, and low in cost and size.

Hypercube architectures perform well for a large range of problems. It is well suited for both general-purpose and special-purpose applications. They are mainly used in matrix operations, sorting, signal and image processing where extensive data processing is required (Walker 1998**.** Figure 1 illustrates the general form of this architecture.

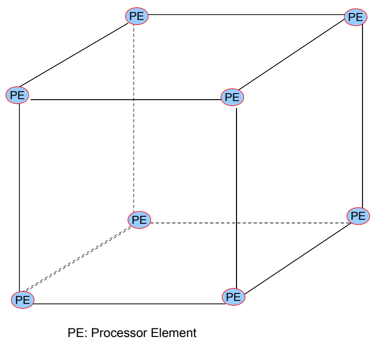

Figure 1: 3 D Hypercube Architecture

In Hypercube architectures when communication between two indirectly connected Processing Elements (PEs) is required, the message has to cross one or more hyper-planes and go through intermediate PEs before reaching its destination. The PEs involved are required to compute and handle message-passing, which reduces the overall computational power and performance. In addition, if one of the intermediate PEs is faulty or busy performing tasks, there will be a significant downtime in communication between the source and destination PEs.

In order to overcome Hypercube limitations such as routing and expandability, an enhanced version of the Hypercube architecture namely Super-Hypercube was developed (Abachi 1997). This architecture includes applying a Router (R) to the basic Hypercube. This router acts as a crossbar switch, which can provide a communication path between two indirect PEs. Its usage in conjunction with SGI (Silicon Graphics Inc.) products relieves the processor of the routing task and provides more efficient computing activities. Figure 2 shows the basic principle of this architecture.
.
As reported in (Grama et al. 2003), interconnection networks can be classified as either dynamic or static. The former interconnection is designed by using switches to connect PEs together. On the other hand, the latter deals with the networks consist of point-to-point communication links among PEs. In the Super-Hypercube topology, the adjacent PEs are directly connected together without use of the Router (R) and indirect PEs are connected together through a Router (R).

Proceedings of the 2008 High Performance Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

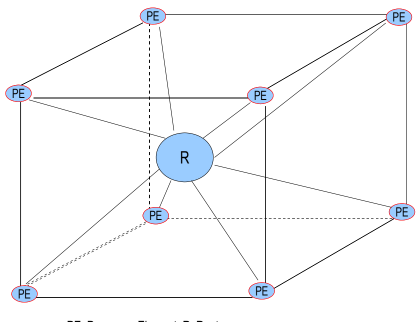

Figure 2: Super- Hypercube Architecture

The Super-Hypercube outlined in this paper uses combination of both categories (dynamic and static). In this paper it is coined as **Dyna**mic-**Static** (**Dynastatic**) interconnection.

The aim of this research is to identify the advantages of communication cost for Dynastatic Super-Hypercube architecture when is compared with the traditional Hypercube topology.

MATRIX MULTIPLICATION ON DISTRIBUTED MEMORY SYSTEMS

In many applications, matrix multiplication involves dealing with different sizes (squares vs. rectangular) and may include the communication cost. The size of the matrix can significantly impact on the performance of parallel matrix multiplication algorithm (Dongarra et al 2007).
This section outlines the general mathematical model for square matrix multiplication.

Basic Concepts, Definitions and Assumptions

Let A and B be matrices of size $m \times n$ and $n \times p$ respectively. The product of A and B is a matrix of size $m \times p$ which denoted by AB and is given by:
$(\mathrm{AB})_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj} = a_{i1}b_{1j} + \cdots + a_{n1}b_{1n}$
for each pair i and j with $1 \leq i \leq m$ and $1 \leq j \leq p$.

For the purpose of this paper, we perform a matrix multiplication on a DMS which is more favorable than shared memory. In doing so, we consider the following definitions (Li 2007):

Definition 1: In order to construct our mathematical we consider that a DMS can support one-to-one communication in $T_{o-oc}(P)$ time unit. For this purpose, a fast and scalable parallel matrix algorithm is required.

Definition 2: We assume that a DMS consists of P PEs $\{p_0, p_1, \cdots, p_{p-1}\}$ with their own local memory$\{\mathrm{m}_0, m_1, \cdots, m_{p-1}\}$. In addition, we consider that PEs have the capability of communicating with each other through message-passing scheme. Moreover, the computation and communication are globally synchronized into steps. That is to say, a step is either a computation step or a communication step. In former, each PE has a capability of performing a local logic/ arithmetic operation or in worse scenario is idle and it utilises constant amount of time.
In latter, PEs could communicate with one another bio-directionally via an interconnection network. In this case, a communication step can be mathematically expressed as:
$((\Psi(0), w_0), (\Psi(1), w_1), \cdots, (\Psi(p-1), w_{p-1}))$ where $0 \leq i \leq p$. This results in PE p_j sending a value w_j to PE $\Psi(P_J)$ and Ψ is a mapping
$\Psi: \{0, 1, \cdots, p-1\} \rightarrow \{-1, 0, 1, \cdots, p-1\}$.

Definition 3: If PE p_j doesn't send any messages during the communication step, then $\Psi(j) = -1$ and w_j is undefined.
However, in a practical situation, there is at most one j such that $\Psi(j) = i$. This implies each PE can maximum receive one message in one-to-one communication step. This also reviles that based on definition 1, the DMS supports the above communication step in $T_{o-oc}(P)$ time frame.
From a practical application point of view, in the busiest communication step, every PE sends a message to another P processor and $(\Psi(0), (\Psi(1), \cdots, (\Psi(p-1))$is a permutation of $(0, 1, 2, \cdots, p-1)$.

Definition 4: Based on the above definitions and assumptions, if a computation step and the communication step in performing a parallel task on a DMS, are T_{cps} and T_{cms} respectively, then the time complexity of performing parallel tasks can be presented as: $O(T_{cps} + T_{cms}T_{o-oc}(P))$.

Furthermore, if the number of PEs in parallel processing system is less than the required sub-tasks, then the time complexity can be shown as:

$$O(\frac{T_{sq}(N)}{p} + T_{par-com}(N, P)) \qquad (1)$$

where N is the problem size, P is the number of PEs available, $T_{sq}(N)$ is the time complexity of the best sequential algorithm, and $T_{par-com}(N, P)$ is the overall communication overhead of a parallel computation.
From an algorithmic point of view, a DMS is characterized by the function $T_{o-oc}(P)$ which measures the communication capability of the interconnection network.

According to (Coppersmith and Winograd 1990), the fastest sequential algorithm for matrix multiplication

has the time complexity of $O(N^{\delta})$ where currently the best value for δ is 2.3755.
Based on these definitions, we try to find out the best time of running this sequential algorithm in parallel form on Hypercube and Super-Hypercube.

MATRIX MULTIPLICATION ON HYPERCUBE AND SUPER-HYPERCUBE

In multiplying two $N \times N$matrices where the number of PEs is less than the number of sub-tasks , i.e. $1 \leq P \leq N^{\delta}$, we assume that l is an integer such that $l^{\delta} \leq P$ i.e. $l = \lfloor P^{\frac{1}{\delta}} \rfloor$ which has the matrices of sub-matrices $\frac{N}{l}$ (i.e., all the matrices $A = (A_{ij}), B = (B_{ij})$ and $C = (C_{ij})$ are partitioned to sub-matrices of size $\frac{N}{l} \times \frac{N}{l}$). Therefore, one can conclude that, in terms of computation time, if we multiply $\frac{N}{l} \times \frac{N}{l}$ matrix by $\frac{N}{l} \times \frac{N}{l}$ matrix sequentially on P PEs it will take $\frac{N^{\delta}}{P}$ units of time.

Calculation of the Communication Time for Matrix Multiplication on Hypercube

As we know the identification of each PE in h_{hp} dimensional Hypercube is based on their binary representation. The set of PEs which are distance d from one PE to another in Hypercube is showed by C_d and it includes $\binom{h_{hp}}{d}$PEs. Since Hypercube is a symmetrical architecture, so any algorithm which is written for any PE can be converted to an identical algorithm for PE $n \in \{1, 2, ...2^{h_{hp}-1}\}$by binary product of all PE i.d's referenced in any specific PE algorithm with n.

We assume that it takes $t_{trm} m_{len} + t_{st-t}$ time for a PE to send a message of length m to a neighbor, where t_{trm} represents the transfer rate of a message and t_{st-t} the time for start up and termination.

According to (F. Stout and Wagar. 1990), the fastest possible time for one PE in h_{hp} dimensional Hypercube to send a message to an arbitrary PE with distance d is:

$$T_{(comm)_{hp}} = \begin{cases} \frac{t_{trm}}{h_{hp}} m_{len} + 2\sqrt{\frac{(d+1)t_{st-t}t_{trm}}{h_{hp}}} m_{len}^{\frac{1}{2}} \\ +(d+1)t_{st-t} \\ \text{for } d \leq (h_{hp} - 1) \\ \frac{t_t rm}{h_{hp}} m_{len} + 2\sqrt{\frac{(h_{hp}-1)t_{st-t}t_{trm}}{h_{hp}}} m_{len}^{\frac{1}{2}} \\ +(h_{hp} - 1)t_{st-t} \\ \text{for } d = h_{hp} \end{cases}$$

(2)

In this scenario, we assume that all PEs can communicate to one another simultaneously.
So, when multiplying two matrices of size $N \times N$on a h_{hp}-dimensional Hypercube by applying sub-matrices of size$\frac{N}{l} \times \frac{N}{l}$, each PE can broadcast the message of length$\frac{N^2}{P^{\frac{2}{\delta}}}$.

For calculating the communication time in the Hypercube, we consider the worse case scenario. This simply implies that if we intend to send a message of length $\frac{N^2}{P^{\frac{2}{\delta}}}$ from any PE to the farthest PE ($d = \log P_{hp}$), that would include the communication time for all the PEs within this range.

Therefore, the time takes to multiply two $N \times N$ matrices in forms of sub-matrices on a Hypercube is:

$$T_{(par-comp)_{hp}} = T_{(comp)_{hp}} + T_{(comm)_{hp}}$$

This results in:

$$T_{(par-comp)_{hp}} = \frac{N^{\alpha}}{P_{hp}} + \frac{t_{trm}}{\log p_{hp}}\left(\frac{N^2}{P_{hp}^{\frac{2}{\delta}}}\right) + 2\frac{\sqrt{(log P_{hp}-1)t_{st-t}t_{trm}}}{log P_{hp}}\left(\frac{N}{P_{hp}^{\frac{1}{\delta}}}\right) + (\log P_{hp} - 1)t_{st_t} \qquad (3)$$

where P_{hp} denotes the number of PEs in Hypercube.

Calculation of the Communication Time for Matrix Multiplication on Super- Hypercube

Now we are in the position to expand the above methodology to cover the Super-Hypercube architecture. This means for the case of Super-Hypercube:

$$T_{(par-comp)_{shp}} = T_{(comp)_{shp}} + T_{(comm)_{shp}}$$

By including the Router (R) in the middle of Hypercube, we have provided a direct connection between any two PEs in Hypercube. Therefore, all PEs in Super- Hypercube are in equal distance to one another.

Moreover, the required time to multiply two $N \times N$ matrices in form of sub-matrices on a Super-Hypercube is:

$$T_{(par-comp)_{shp}} = \frac{N^{\delta}}{P_{shp}} + \frac{t_{trm}}{\log P_{shp}}\left(\frac{N^2}{P_{shp}^{\frac{2}{\delta}}}\right) + 2\frac{\sqrt{2t_{st-t}t_{trm}}}{\log P_{shp}}\left(\frac{N}{P_{shp}^{\frac{1}{\delta}}}\right) + 2t_{st-t}$$

(4)

where P_{shp} is the number of PEs in Super-Hypercube. In deriving equation (4) we assumed that $P_{shp} = P_{hp}$.

COMPARISON

Figure 4 shows the graphical presentation of communication cost with variable number of PEs and Figure 5 presents the graphical presentation of the communication cost with variable matrix size. In evaluating these results we have assumed that $t_{trm} = t_{st-t} = 1$.

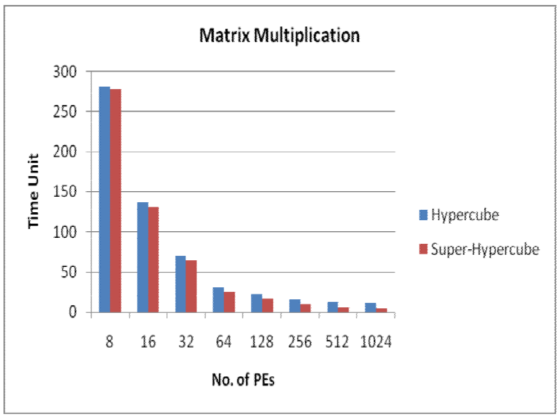

Figure 4. Graphical Presentation of communication cost for Matrix Multiplication with variable number of PEs.

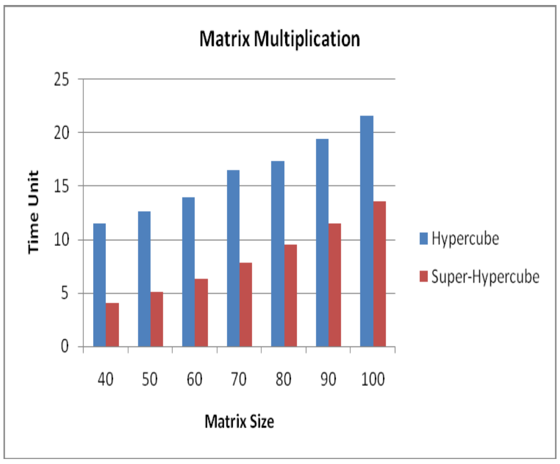

Figure 5. Graphical Presentation of communication cost for Martix Multiplication with variable matrix size.

By providing a direct path between any two indirect nodes through a Router (R), the communication time of a matrix product is sigificanytly shorter in Super-Hypercube compare with the Hypercube archticture. This is evident by analyzing Figure 4 and Figure 5 respectively. This accomplishment has played an important part towards improving the overall operation time and hence the performance of the message-passing architecture. This is clearly evident in Figure 4, when the number of PEs exceeds 256.

CONCLUSION

This paper has addressed the communication cost of a matrix product on message-passing architectures. The mathematical modeling for matrix multiplication on Hypercube and Super-Hypercube architectures was derived and numerical results for both architectures were presented. The existence of a Router (R) in a traditional Hypercube which results in having the Super-Hypercube has significantly improved the overall performance of the system. As a further work, we intend to propose an enhanced version of Super-Hypercube architecture. Then we will deliver the general formula for parallel time computation. This will be complementing the mathematical calculations and simulation carried out for this architecture. Furthermore, to validate this analysis, the most appropriate architecture will be chosen and compared with our findings to support our claims.

REFRENCES

Abachi, H and A.L, Walker. 1997. "Simulation Modeling of Fault-Tolerant Hypercube, Super-Hypercube and Torus Networks" *Proceeding of 12th International Conference on Computers and Their Applications (ISCA)*, Arizona, U.S.A, 50-53 (March).

Amiripour, M.; H. Abachi; and R. Lee. 2007. "Total System Cost and Average Routing Distance Analysis of Master-Slave Super-Super-Hypercube 4-Cube Message-Passing Architecture*" The International Journal of Computer and Information Science (IJCIS)*, Vol 10, No 2, 269-279 (June).

Coppersmith, D and S. Winograd. 1990. "Matrix Multiplication via Arithmetic Progressions." *J. Symbolic Computation*, Vol 9, 251-280.

Dongarra, J; J.F. Pineau; Y. Robert; Zh. Shi; and F. Vivien. 2007 " Revisiting Matrix Product on Mater-Worker Platform."IEEE Proceding on Parallel and Distributed Processing Symposium,(IPDPS 2007) ,1-8, (March)

F. Stout, Q. and B. Wagar. 1990. "Intensive Hypercube Communication: Prearranged Communication in Link-Bound Machines." *Journal of Parallel and Distributed Computing 10*, 167-181**.**

Grama, A.; A. Gupta; G. Karypis; and V. Kumar. 2003. "*Introduction to Parallel Computing*." Addison Wesley, U.S.A.

MARYAM AMIRIPOUR received her B.A. in Mathematics from Al-Zahra University in Iran in 1999. That was followed by a Post Graduate Diploma in Information and System Management form Queensland University in Australia in 2001.She is currently pursuing her PhD degree in Department of Electrical and Computer Systems Engineering at Monash University in Australia. Her area of research includes hardware design, modeling and simulation of advance parallel processing systems. The main parameters of her investigation include evaluation of performance, reliability, speed and cost analysis of massively parallel processing systems. She has a number of referred journal and conference papers in these areas. Her e-mail address is: maryam.amiripour@eng.monash.edu.au.

HAMID ABACHI received his Ph.D. degree in Computer Engineering from University College Cardiff in Wales, Britain, in 1981. He has twenty five years of teaching, research and administrative experiences in international universities around the world. He is currently an Associate Professor in the Department of Electrical and Computer Systems Engineering at Monash University in Australia. He has more than 95 referred international publications including Journal and conference papers. He has served as a member of international program committee to more than 72 international conferences around the world. On a number of occasions has acted as the conference chairman and on many occasions as the session chairman at international conferences. He has also participated in the plenary sessions at sessions at international conferences. He has been a co-recipient of the John Madsen Medal for his best Journal paper in the discipline of Electrical Engineering form the Institution of Engineers Australia (IEAust) in 2002, plus receiving a number of best paper awards in international conferences. In addition he has been invited as a keynote speaker at four international conferences. He is a Fellow of The Institution of Engineering and Technology (the IET, formerly IEE) in Britain, a Fellow of the Institution of Engineers in Australia (IEAust) and a Senior Member of IEEE in the USA. His research interests include design and simulation of parallel processing systems, modeling of advanced computer architectures, application of distributed multimedia computing in advanced Engineering Education. His e-mail address is: hamid.abachi@eng.monash.edu.au.