# Impact of New Sub-Classes of Hypercube Topology on Execution Time of Matrix Multiplication

First M. Amiripour, Second H. Abachi

*Abstract*—Today's advanced research areas such as DNA computing, different branches of nanotechnology, immune cell system and optical computing require extensive data processing. Therefore, parallel processing systems with sophisticated hardware and software platforms are widely used. Furthermore, implementing the right algorithm which affects the overall execution time is a challenging task. This paper presents the principle of a massively parallel processing system based on Master-Slave Super-Super Hypercube 4-Cube ($MS^3H4$-Cube) topology which could be easily implemented by using SGI products. Moreover, it is verified that the execution time of a matrix product is shorter when is applied on Master- Slave Super-Hypercube (MSSHP) compared with Hypercube (HP) topology.

***Keywords*—**Architecture, Execution Time, Master-Slave, Matrix Product, Super-Hypercube

## I. 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. HP 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 [1].

In HP 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 HP limitations such as routing and expandability, a sub-class of the HP architecture namely Super-Hypercube (SHP) is used [2]. This architecture includes applying a Router (R) to the basic HP. This router acts as a crossbar switch, which can provide a communication path between 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. Another configuration could be to include one extra PE (Master-Processor) which through a Master-Router (MR) is connected to the remaining PEs (Slave-Processors). This configuration called Master-Slave Super-Hypercube (MSSHP) as shown in Figure 1. As far as the interconnectivity of the proposed architecture is concerned, this architecture falls into a new interconnection category.

As reported in [3], 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 proposed topology, the adjacent PEs are directly connected together without use of the Router (R) and indirect PEs are connected together through a Router (R).

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

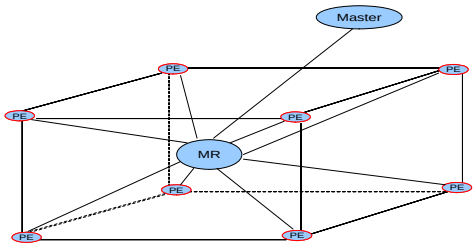


Fig. 1 Master-Slave Super-Hypercube architecture

## II. Description of $MS^3H4$-Cube Architecture

The basic building blocks of this architecture as shown in Figure 2 and previously reported in [4], is based on the SHP architecture.

Under this arrangement, each processing element in each Super-Super-hypercube which contains the Router$R_{11}$, is itself a SHP with the Router$R_{111}$.

Manuscript received Jov 6, 2007: Revised version received Mar 12, 2008.

F. M. Amiripour is with Monash University , Department of Electrical and Computer Systems Engineering, Victoria 3800, Australia. ( phone: +613 99053824; fax:+613 99053454; e-mail: maryam.amiripour@ eng.monash.edu.au).

S. H. Abachi is with Monash University , Department of Electrical and Computer Systems Engineering, Victoria 3800, Australia. ( phone: +613 99053824; fax:+613 99053454; e-mail: hamid.abachi@ eng.monash.edu.au).

The processing element with the Router $R_{111}$ is called satellite slave. In general, the overall control and management of the system is carried out by a Master-Processor. For this reason the overall architecture is called Master-Slave Super-Super-Hypercube n-cube architecture (MSSSHn-Cube), or further abbreviated as $MS^3Hn-Cube$. Figure 2 illustrates the interconnection of two Super-Super-Hypercube which results in construction of $MS^3H4-Cube$ topology. However, upon availability of technology and hardware capability this could be extended to $n$ Super-Super-Hypercube arrangement.

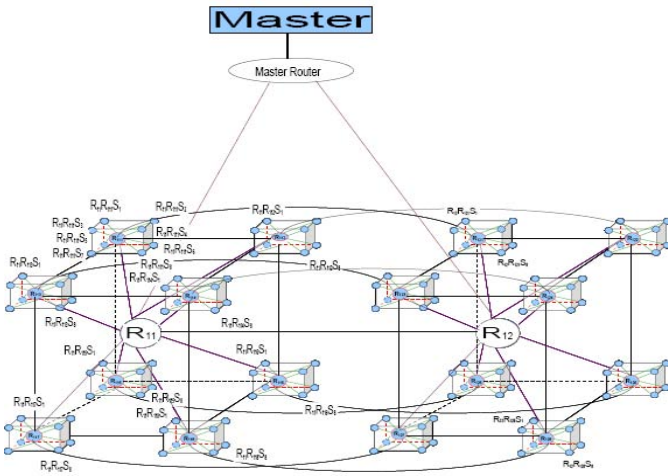


Fig. 2 $MS^3H4-Cube$ architecture

### A. Operation of $MS^3H4-Cube$ Architecture

As it was reported in [5], Master-Slave tasking is a simple yet widely used technique to execute independent task under the centeralised supervision of a Master-Processor. the operation of $MS^3H4-Cube$ architecture in a massively parallel processing system as reported in [6] can best be explained as follows.

The main role of the Master-Processor is the task allocation and overall management and control of the system. To achieve this goal, undoubtedly, the Master-Processor needs to posses faster processing capability and additional memory capacity to be able to have full control of the system management. Once the main task is divided into multiple sub-tasks, then it is placed in the main memory of the Master-Processor. At this point there are two possibilities that one can consider:

1- Either to use the routers as crossbar switches, without any processing capabilities, to direct the packages from source to destinations and vice-versa. Therefore, for transmitting a subtask from the Master-Processor to Slave 1 in the satellite slave configuration that contains Router $R_{111}$, first the subtask would find its way through $R_{11}$ to $R_{111}$ and then would reach to the processing element $R_{11}R_{111}S_1$. In this context, one could provide direct connections between the Master-Processor and routers $R_{111}, R_{112}, \cdots, R_{118}, R_{121}, R_{122}, \cdots, R_{128}$ which will bypass router $R_{11}$.

2- To incorporate some processing capabilities and memory facilities within each router $(R_{11}, R_{111}, \cdots, R_{11}, R_{118}, R_{12}, R_{121}, \cdots, R_{12}, R_{128})$. In this scenario, the subtask could be saved in the memory of each router (for example first in $R_{11}$ and then in $R_{111}$), before reaching its final destination in $R_{11}R_{111}S_1$. In reality one can consider the routers as co-Master-Processors in this arrangement.

The advantage and disadvantage of each approach can be explained as follows. In the first case although data transmission and computation is faster, the overall fault-tolerance of the system is lower if there is a fault in the Master-Processor. On the other hand, the second approach may seem a bit involved and slower than the first case but if there is a hardware or software fault within the Master-Processor, the overall system is not subject to a catastrophic failure. This is due to the fact that routers have memory and processing capabilities and they in turn can act as co-Master-Processor within their own SHP arrangement. The transfer of data and information can take place through a direct connection that is provided for this purpose. So far as the multiprocessor scheduling which is a challenging problem in the real time system theory is concerned, one has to consider two main strategies. These are: partitioning strategy and global strategy.

In former scenario a task is allocated to a processor and is executed by that processor. The latter deals with the case when at any instance a task can be executed on any processor or even be preempted and moved to a different processor before is completed [7]. In the proposed architecture, the former scheme is adopted for the task allocation methodology.

In operation of the proposed architecture after the task is divided into a number of subtasks, they are allocated through $R_{11}$ and $R_{111}$ to each satellite slave processor for execution. After completion of execution of each subtask, a copy of the result is saved in the memory of each co-Master-Processor (assuming that $R_{11}$ is level one and $R_{111}$ is level two co-Master-Processors) plus the memory of the main Master-Processor. This precaution is taken to minimize a catastrophic failure that could result if the Master-Processor failed.

Upon completion of each subtask, the satellite slave processor would request allocation of the next available subtask by sending an interrupt request signal through appropriate channels to the Master-Processor. This procedure will continue until all the subtasks are executed and results are collected by the Master-Processor. In the case when the subtasks are independent of one another and the execution time is the same, then that results in simultaneous interrupt request signals arriving at the co-Master-Processors and subsequently at the Master-Processor. In order to resolve the conflict resulting from initiation of simultaneous interrupt requests, the system designer through either hardware or software arrangement can assign priority to different satellite slave processors where they will receive services according to their priorities. Of course implementation of software priority scheme would be more convenient than a hardware approach. This is due to the fact that rearranging priority through software means reprogramming of the control register

belonging to the interrupt priority controller device, whereas the hardware approach needs rearranging connection of priority signals to different satellite slave processors. The latter naturally is more time consuming and hence more inconvenient. It is worth noting that while satellite slave processors are engaged in the execution of subtasks, co-Master-Processors as well as the Master-Processor can be involved in the supervisory and control of the overall system. In addition, a precautionary approach could be implemented that includes random execution of subtasks and its comparison with results obtained by the satellite slave processors. Nevertheless, in the event of any discrepancy, the warning arrangement which is incorporated within the control section of the Master-Processor would generate a warning signal indicating either a hardware malfunctioning or a software failure in the system which contributed towards this discrepancy.

Incorporation of the former approach would improve the control and management procedures whereas the implementation of the latter would undoubtedly enhance the reliability of such a dedicated message- passing architecture.

Furthermore, implementation of Artificial Intelligence and Expert System within the operating system of Master-Processor would significantly enhance the performance of the overall system.

### *B. The Building Blocks of MS3H4-Cube Architecture*

The building block of each processing element and the router proposed for this topology are based on the technology developed by Silicon Graphics Inc (SGI)[8]. It is believed that this product provides a suitable test-bed for this investigation. Although SGI has many products that are specifically designed to be used in multiprocessor environment, the most recent and suitable architectures (Altix 3700Bx2 series) has been chosen to be used in the design of the building blocks of MS3H4-Cube architecture [3].

## III. MATRIX PRODUCT ON DISTRIBUTED MEMORY SYSTEMS (DMS)

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.

This section outlines the general mathematical model for square matrix multiplication.

### *A. 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 $1 \le i \le m$ and $1 \le j \le p$.

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 [9]:

**Definition 1***:* In order to construct our mathematical model, 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 $0 \le i \le p$ . This results in PE $p_j$ sending a value $w_j$ to PE $\Psi(P_J)$ and $\Psi$ is a mapping

$\Psi : \{0, 1, \cdots, p-1\} \to \{-1, 0, 1, \cdots, p-1\}$.

**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 that 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 unit.

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 $(0, 1, 2, ...p-1)$.

**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 execution time 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 execution time can be shown as:

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

where $N$ is the problem size, $P$ is the number of PEs available, $T_{sq}(N)$ is the execution time 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 [10], the fastest sequential algorithm for matrix multiplication has the execution time of $O(N^{\delta})$ where currently the best value for $\delta$ 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.

## IV. MATRIX PRODUCT ON HP AND SHP

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.

### A. Computation of the Communication Time for Matrix Product on HP and SHP

As we know the identification of each PE in $h_{hp}$ dimensional HP is based on their binary representation. The set of PEs which are distance $d$ from one PE to another in HP is showed by $C_d$ and it includes $\binom{h_{hp}}{d}$PEs. Since HP 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.

### B. Analysis of Mathematical Modeling

According to [11], the fastest possible time for one PE in $h_{hp}$ dimensional HP to send a message to an arbitrary PE with distance $d$ is

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

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 HP 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 HP, 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 HP is:

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

This results in:

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

where $P_{hp}$ denotes the number of PEs in HP.
Now we are in the position to expand the above methodology to cover the SHP architecture. This means for the case of SHP:

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

By including the Router (R) in the middle of HP, we have provided a direct connection between any two PEs in HP. Therefore, all PEs in SHP 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 SHP is:

$$T_{(par-comp)_{shp}} = \frac{N^{\delta}}{P_{shp}} + \frac{t_{trm}}{\log P_{shp}}(\frac{N^2}{P_{shp}^{\frac{2}{\delta}}}) + 2\frac{\sqrt{2t_{st-t}t_{trm}}}{\log P_{shp}}(\frac{N}{P_{shp}^{\frac{1}{\delta}}}) + 2t_{st-t} \quad (4)$$

Where $P_{shp}$ is the number of PEs in SHP. In driving equation (4) we assumed that $P_{shp} = P_{hp}$

### C. Execution Time of Matrix Product on HP Architecture

In HP architecture when processing is carried out in parallel, it is best to assume that one of the processors acts as the Master-Processor which sends the message of size $\frac{N^{\delta}}{P^{\frac{2}{\delta}}}$ to each slave processor for processing purpose. In this case, the Slave-Processors after receiving the allocated subtasks perform the inner product of sub-matrices and upon their completion will send the results back to the Master-Processor for summation. This process includes scatter and gather which means the Master-Processor sends a piece of data of equal size to all the Slave-Processors and then all the Slave-Processors send k pieces of data of the same length to the Master-Processor [12]. Therefore, the time that it takes a message of length $\frac{N^{\delta}}{P^{\frac{2}{\delta}}}$ to be sent from the Master-Processor to a single Slave Processor is:

$$T(comm)_{(Master-SingleSlave)_{hp}} = t_{st-t} + \frac{N^{\delta}}{P^{\frac{2}{\delta}}} t_{trm}$$

And the time that it takes the massage to be received by a single Slave-Processor is:

$$T(comm)_{(SingleSlave-Master)_{hp}} = t_{st-t} + \frac{N^{\delta}}{P^{\frac{2}{\delta}}} t_{trm}$$

Therefore, sending and receiving a message to and from a single slave processor takes the following unit of times:

$$T(comm)_{(SingleSlave-Master)_{hp}} + T(comm)_{(Master-SingleSlave)_{hp}} = 2[t_{st-t} + \frac{N^{\delta}}{P^{\frac{2}{\delta}}} t_{trm}]$$

Finally, the total time to send a message of length $\frac{N^{\delta}}{P^{\frac{2}{\delta}}}$ to all the Slave-Processors and the time that it takes to receive it from them is:

$$T(comm)_{(Master-Slave)_{hp}} + T(comm)_{(Slave-Master)_{hp}} = 2\sum_{i=1}^{\log P_{hp}}(t_{st-t}) + (2^{\log P_{hp}} - 1)\frac{N^{\delta}}{P^{\frac{2}{\delta}}} t_{trm} = 2(\log P_{hp} t_{st-t} + (2^{\log P_{hp}} - 1))\frac{N^{\delta}}{P^{\frac{2}{\delta}}} t_{trm}$$

Further substitution and simplification results in having the parallel time execution of a matrix product on HP as:

$$\begin{aligned} T_{(par-comp)_{hp}} &= \frac{N^{\delta}}{P_{hp}} + \frac{t_{trm}}{\log P_{hp}}(\frac{N^2}{P_{hp}^{\frac{2}{\delta}}}) + 2\frac{\sqrt{(\log P_{hp}-1)t_{st-t}t_{trm}}}{\log P_{hp}}(\frac{N}{P_{hp}^{\frac{1}{\delta}}}) \\ &+ (\log P_{hp} - 1)t_{st-t} + 2\log P_{hp} t_{st-t} \\ &+ 2(2^{\log P_{hp}} - 1)\frac{N^2}{P_{hp}^{\frac{2}{\delta}}} t_{trm} \\ &= \frac{N^{\delta}}{P_{hp}} + \frac{2(2^{\log P_{hp}}-1)}{\log P_{hp}}(\frac{N^2}{P_{hp}^{\frac{2}{\delta}}})t_{trm} \\ &+ \frac{2\sqrt{(\log P_{hp}-1)t_{st-t}t_{trm}}}{\log P_{hp}}(\frac{N}{P_{hp}^{\frac{1}{\delta}}}) \\ &+ (3\log P_{hp} - 1)t_{st-t} \end{aligned} \quad (5)$$

## V. EXECUTION TIME OF MATRIX PRODUCT ON MSSHP

As we know the identification of each PE in $h_{hp}$ dimensional HP is based on their binary representation. The set of PEs which are distance $d$ from one PE to another in HP is showed by $C_d$ and it includes $\binom{h_{hp}}{d}$PEs. Since HP 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.

By further expansion, we can conclude that, the parallel time computation of MSSHP can be presented as:

$$\begin{aligned} T_{(par-comp)_{msshp}} = &\sum_{i=1}^{P_{shp}} T_{(comm)_{master-slave(p_i)}} + T_{(comp)_{shp}} + \\ &T_{(comm)_{shp}} + \sum_{i=1}^{P_{shp}} T_{(comm)_{slave(p_i)-master}} \end{aligned} \quad (6)$$

To derive the above expression, we assume that:

- the Master-Processor has the processing capability ;
- the allocation of sub-tasks by the Master-Processor to each Slave-Processor is carried out one at a time;
- the Slave-Processors start to process after receiving entire message and can not communicate with one another or with the Master-Processor before completing the execution of the current subtask;
- the utlised platform is homogenous; and
- the result of execution of each subtask is return to the Master-Processor for further processing.

This model is then extended to derive the analytical modeling of task scheduling for the MSSHP architecture.

$$\begin{aligned} T_{(par-comp)_{msshp}} &= \frac{N^{\delta}}{P_{shp}} + \frac{t_{trm}}{\log P_{shp}}(\frac{N^2}{P_{shp}^{\frac{2}{\delta}}}) \\ &+ \frac{2\sqrt{2t_{st-t}t_{trm}}}{\log P_{shp}}(\frac{N}{P_{shp}^{\frac{1}{\delta}}}) + 2t_{st-t} + 2t_{st-t} \\ &+ 2(2^{\log P_{shp}} - 1)\frac{N^2}{P_{shp}^{\frac{2}{\delta}}} t_{trm} \\ &= \frac{N^{\delta}}{P_{shp}} + \frac{2(2^{\log P_{shp}}-1)}{\log P_{shp}}(\frac{N^2}{P_{shp}^{\frac{2}{\delta}}})t_{trm} \\ &+ 2\frac{\sqrt{2t_{st-t}t_{trm}}}{\log P_{shp}}(\frac{N}{P_{shp}^{\frac{1}{\delta}}}) + 4t_{st-t} \end{aligned} \quad (7)$$

At this stage we use the existing mathematical models derived so far to demonstrate the graphical presentations of the execution time of matrix product on both HP and MSSHP architecture. Figure 3 illustrates the graphical presentation for Matrix Multiplication with a variable number of PEs on HP and MSSHP. However Figure 4 provides the graphical presentation for Matrix Multiplication with variable Matrix Sizes on HP and MSSHP.

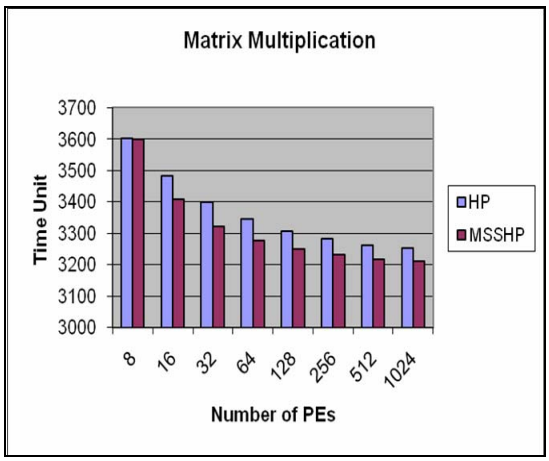


Fig. 3 graphical presentation for matrix product with variable number of PEs on HP and MSSHP

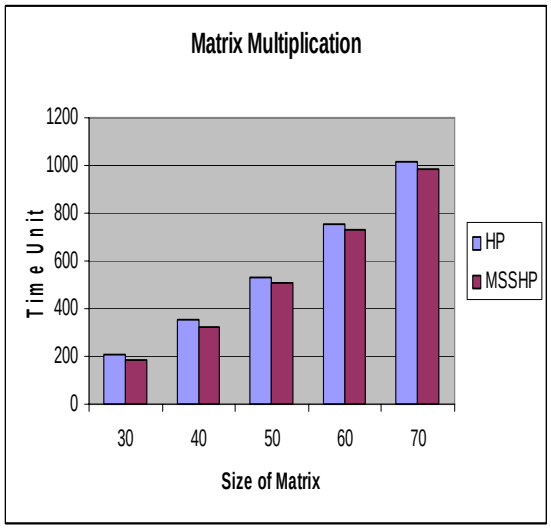


Fig. 4 graphical presentation for matrix product with variable matrix sizes on HP and MSSHP

## VI. PROCEDURE FOR CALCULATING THE EXECUTION TIME OF MATRIX MULTIPLICATION ON MS$^3$H4-CUBE

As indicated earlier, the role of the router can either be considered as being a cross bar switch for providing communication path between two indirect nodes, or it can have some processing capability and could be considered as a co-Master-Processor. In the first case, the execution time can have the general form of:

$$T(par - comp)_{MS^3H-4Cube} = \sum_{j=1}^{2}\sum_{i=1}^{8} T(comm)_{Master-Slave(R_{1j}R_{1ji}S_i)} + T(comm)_{ssh-4cube} + T(comp)_{ssh-4cube} + \sum_{j=1}^{2}\sum_{i=1}^{8} T(comm)_{Slave(R_{1j}R_{1ji}S_i)-Master}$$

Moreover, in case of router acting as the co-Master-Processor, the general form of the execution time takes the general form of:

$$T(Par - Comp)_{MS^3H4_{Cube}} = \sum_{j=1}^{2} T(comm)_{Master-Router(R_{ij})} + \sum_{j=1}^{2}\sum_{i=1}^{8} T(comm)_{Router(R_{1j})-Router(R_{1ji})} + \sum_{j=1}^{2}\sum_{i=1}^{8} T(comm)_{Router(R_{1ji})-Slave(S_i)} + T(comp)_{MS^3H4-Cube} + \sum_{j=1}^{2}\sum_{i=1}^{8} T(comm)_{Slave(S_i)-Router(R_{1ji})} + \sum_{j=1}^{2}\sum_{i=1}^{8} T(comm)_{Router(R_{1ji})-Router(R_{1j})} + \sum_{j=1}^{2} T(comm)_{Router(R_{ij})-Master}$$

However, the full description and further expansion of these mathematical expressions will be revisited in our future work.

.

## VII. CONCLUSION

This paper has addressed the impact of new sub-classes of HP topology on execution time of matrix multiplication. To achieve this, we have highlighted the concept of MSSHP architecture which has led to the introduction of a topology entitled MS$^3$H 4-Cube architecture. Furthermore, the communication time and the overall execution time of matrix product on HP and consequently on MSSHP architecture have been outlined. This has provided the foundation to proceed with driving a similar mathematical modeling applicable to MS$^3$H 4-Cube architecture.

We have concluded that the completion time of matrix multiplication when we vary the number of PEs or the size of matrix is shorter for MSSHP when has been compared to that of HP architecture.

This claim can be justified by considering the effect of having a router in the HP topology which results in offering an optimized communication path. Consequently, this improves the final expression for the communication cost and overall time execution of a matrix chain product. Lack of this inclusion is highly noticeable, when one experiences either a busy or faulty intermediate node between two indirect nodes in a HP topology. Of course in this scenario, the performance of the massively parallel processing system will be significantly degraded.

As far as the future work is concerned, we intend to expand the expressions for the communication cost and the execution time of a matrix product on the MS$^3$H 4-Cube topology. This then leads to arriving at an appropriate task scheduling for the

$MS^3H$4-Cube architecture. To validate this analysis, we intend to compare our findings with benchmark results using SGI products. Further to this justification, we will choose the most appropriate architecture in literature and will compare our findings to support our claims.

## REFERENCES

[1] J. Walker, "Performance, Reliability and Cost Analysis of Message Passing Architecture." *Master of Engineering Thesis*" Department of Electrical and Computer Systems Engineering Monash University, Feb 1998.

[2] H. Abachi and J. Walker. "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 1997

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

[4] Amiripour, M.; H. Abachi; and R. Lee. "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, pp. 269-279 June 2007.

[5] Beaumont, O; A. Legrand and Y. Robert, "The Master-Slave Paradigm with Heterogonous Processors" *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, No 9, pp. 897-908, Sep 2003.

[6] Amiripour. M, Abachi. H, "Average Routing Distance Analysis and Comparison of Master-Slave Super-Super-Hypercube 4-Cube Topology with different Message Passing Architectures" *6th IEEE/ ACIS International Conference on Computer and Information Science (ICIS )*, Australia, pp. 622-628 July 2007.

[7] Lo'pez. J et al. "Minimum and Maximum Utilisation Bounds for Multiprocessor Rate Monotonic Scheduling" *IEEE Transactions on Parallel and Distributed Systems*, Vol 15, no. 7, pp.642-653, July. 2004.

[8] Sillicon Graphics Inc, "Hardware: End-User, Altix 3700 Bx2" System Overview, Chapter 3, U.S.A, 2004, PP.1-6.

[9] Li, K. "Analysis of Parallel Algorithms for Matrix Chain Product and Matrix Powers on Distributed Memory Systems." *IEEE Transaction on Parallel and Distributed Systems*, Vol. 18, No. 7 July 2007.

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

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

[12] Y. Saad, and M.H. Schultz, "Data Communication in Hypercube." *Journal of Parallel and Distributed Computing"* Vol. 6, pp. 115-135, 1989.