A STUDY OF COMPARISON OF TWO PARALLEL ARCHITECTURES FOR MATRIX MULTIPLICATION

Dr. Girish Kumar*

Ankit Kumar**

ABSTRACT

In this paper, efforts are made to study the parallel architectures and the topology associated with parallel architectures. Further a parametric study is conducted for solving the matrix multiplication problem by using two different parallel architectures and it has been observed that matrix size and behavior is dependent on the complexity of the architecture being used for carrying the multiplication.

Keywords: PRAM Model, Scalable, Speed Up, Network Topologies, Hypercube, 2-D MESH

* Associate.Prof., BPIBS, GNCT, Shakarpur,Delhi-110092

** Asst. Prof. ,SIET, Gangoh, Saharanpur-247341

1. INTRODUCTION

Parallel processing is information processing that emphasizes the concurrent manipulation of data elements belonging to one or more process solving a single problem. A parallel computer is a multiple processor computer capable of parallel processing.

The throughput of a device is the number of results it produces per unit time. There are many ways to improve the throughput of a device. The speed at which the device operates can be increased, or the concurrency the number of operations that are being performed at any one time can be increased.

Pipelining and data parallelism are two ways to increase the concurrency of a computation. (13).

Data parallelism is the use of multiple functional units to apply the same operation simultaneously to elements of a data set. A k-fold increase in the number of functional units leads to a k-fold increase in the throughput of the system, if there is no overhead associated with the increase in parallelism.

Speedup is the ratio between the time needed for the most efficient sequential algorithm to perform a computation and the time needed to perform the same computation on a machine incorporating pipelining and/or parallelism.

##Associate.Prof.,BPIBS,GNCT,Shakarpur,Delhi-110092

#Asst. Pof.,SIET,Gangoh,Saharanpur-247341

1.1. CONTROL PARALLELISM:-

Our discussion has focused on data parallel and pipelined algorithms. Pipelining is actually a special case of a more general class of parallel algorithms, called control parallel algorithms. In contrast to data parallelism, in which parallelism is achieved by applying a single operation to a data set, control parallelism is achieved by applying different operations to different data elements simultaneously.

Most realistic problems can exploit both data parallelism and control parallelism. Realistic problems also have some precedence relations between different tasks. For example, consider the problem of performing an estate's weekly landscape maintenance as quickly as possible (Fig 1-4). Suppose four chores must be performed: moving the lawn, edging the lawn, checking the sprinklers, and weeding the flower beds. With the exception of checking the sprinklers which is easily performed by a single person, each of the remaining chores can be done more quickly by multiple workers. Increasing the lawn mowing speed by creating a lawn mowing team and assigning each team member a portion of the lawn is an example of data parallelism. Since there is no reason why the flower beds cannot be weeded at the same time the lawn is being mowed, we can assign another team to the weeding. Concurrent weeding and lawn mowing is an example of control parallelism. Precedence relations exist between checking the sprinklers and the three other tasks, since all of the other tasks must be completed before the Sprinklers are tested.

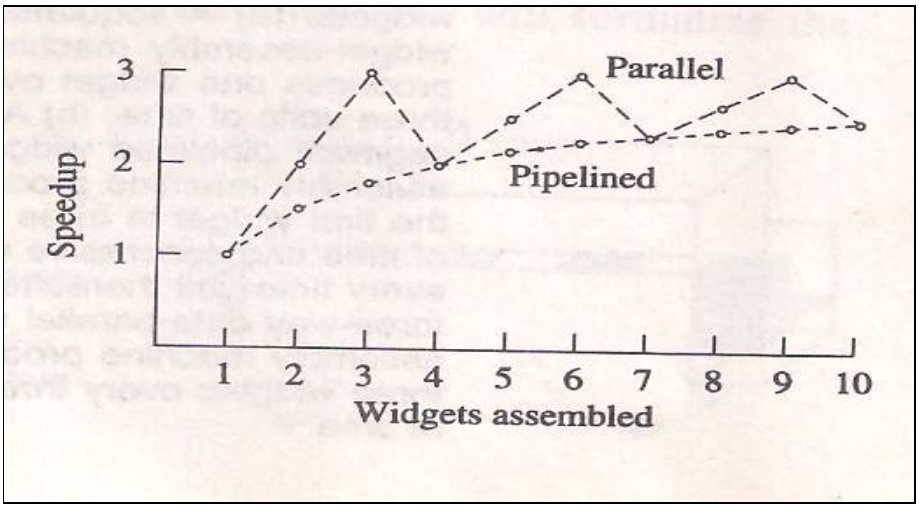

Fig.1-3 shows Speedup achieved by pipelined and parallel widget assembly machines.

2. Processor Topologies

2.1 MESH NETWORK

In a mesh network, the nodes are arranged into a q-dimensional lattice. Communication is allowed only between neighboring nodes; hence interior nodes communicate with 2q other processors. Figure 2-1a illustrates a two dimensional (2-D) mesh. Some variants of the mesh model allow wrap around connections between processors on the edge of the mesh. These connections can connect processors in the same row or column (Fig. 2-1b) or adjacent rows or columns (Figs 2-1c).

Let's evaluate the mesh network according to our four criteria. We assume that the mesh has no wrap around connections. The diameter of a q-dimensional mesh with k^q nodes is $q(k - 1)$. Hence, from a theoretical point of view, mesh networks have the disadvantage that data routing requirements often prevent the development of polylogarithmic time parallel algorithms. In practice, however, some computer architects would rather implement fewer, faster links than more, slower links (11).

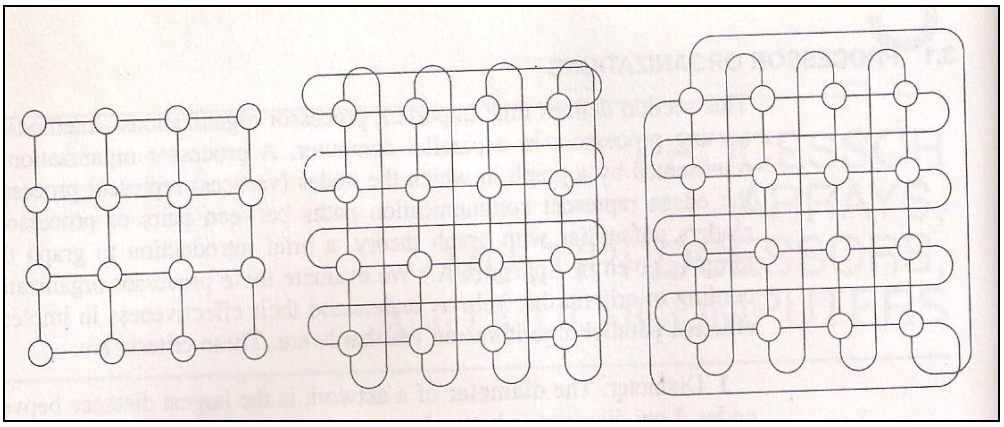

Figure2-1 Shows Two Dimensional Meshes

(a) Mesh with no wrap-around connections. (b) Mesh with wrap-around connections between processors in same row or column. (c) Mesh with wrap-around connections between processors in adjacent rows or columns.

2.2 HYPERCUBE (CUBE-CONNECTED) NETWORKS

A cube connected network, also called a binary n-cube network, is a butterfly with its columns collapsed into single nodes. Formally, this network consists of 2^k nodes forming a k-dimensional hypercube. The nodes are labeled 0,1…..,.,$2^k - 1$; two nodes are adjacent if their labels differ in exactly one bit position. A four dimensional hypercube is shown in figure2-2 (12).

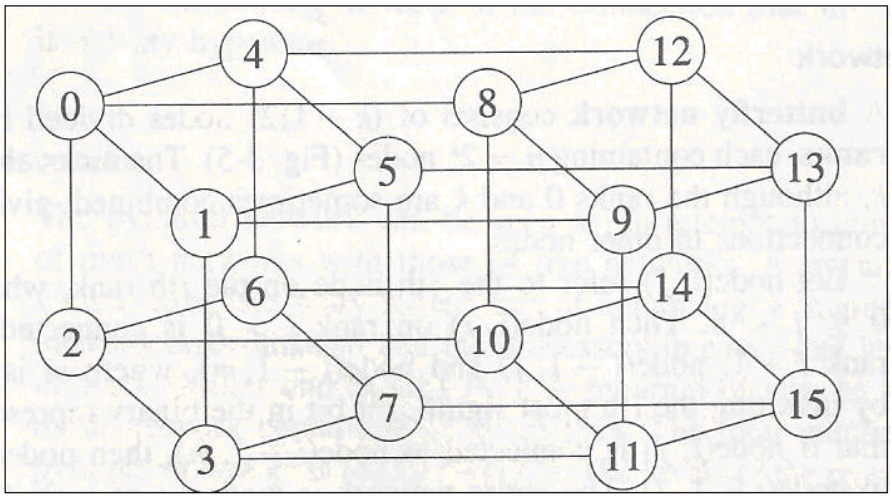

Figure 2-2 A four dimensional (16 node) hypercube.

3. MATRIX MULTIPLICATION ALGORITHMS

3.1 MATRIX MULTIPLICATION ON HYPERCUBE MODEL

Parameter q { Matrix size is 2^q x 2^q }
Global l
Local a, b, c, s, t
Begin

{Phase 1: Boradcast matrices A and B}
For l ← 3q – 1 downto 2q do
For all Pm, where BIT(m,l) = 1 do
t ← BIT.COMPLEMENT(m,l)
a< = [t]a
b< = [t]b

endfor
endfor
for l <- q – 1 downto 0 do
for all Pm ,where BIT(m,l)≠BIT (m,2q+l) do
t <- BIT.COMPLEMENT (m,l)

```
                a< = [t]a
        endfor
endfor
   For l <-2q – 1 downto q do
   For all Pm where BIT(m,l)≠BIT (m,q+l) do
                t <- BIT.COMPLEMENT(m,l)
                 b< = [t]b
endfor
endfor
{Phase 2: Do the multiplications in parallel}
         For all Pm do
              c <- a x b
endfor
      {Phase 3: Sum the productrs}
          For l <- 2q to 3q – 1 do
        for all Pm do
              t <- BIT.COMPLEMENT(m,l)
                 s = [t]c
                  c <-c + s
   endfor
  endfor
end
```

The two new functions, BIT and BIT COMPLEMENT, are used by this algorithm. Function BIT, passed integer arguments m and l, and returns the value of the lth bit in the binary representation of m. Function BIT.COMPLEMENT, passed integer arguments m and l, and returns the value of the integer formed by complementing the value of bit l in the binary representation of m.

The first for loop requires 2q data routing steps. The last three for loops require q data routing steps each. Hence a total of 5q data routing steps are sufficient to multiply two matrices on the hypercube SIMD model. The algorithm also uses one multiplication step and q addition steps. Clearly, the complexity of matrix multiplication on the hypercube SIMD model is $\theta(q) = \theta(\log n)$, given n^3 processing elements (19).

3.2 MATRIX MULTIPLICATION ON 2-D MESH SIMD

```
Global n          {Dimension of matrices}
       k
Local  a, b, c

Begin
    {Stagger matrices}
          For k ← 1 to n – 1 do
          For all P(i, j) where 1 ≤ i, j ≤ n do
            If i > k then
            a< = east (a)
           endif
           if j > k then
            b< = south (b)
                    endif
                     endfor
                endfor


          {Compute dot products}
          For all P(i,j) where 1 ≤ i, j ≤ n do
           c ←  a x b
           endfor
           for k ← 1 to n-1 do
           for all P(i,j)where 1 ≤ i, j ≤ n do
           a = east (a)
           b = south (b)
           c ← c + a x b
```

(10) has shown that multiplication of two n x n matrices on the 2-D mesh SIMD model requires Θ(n) data routing steps.

Matrix multiplication on the 2-D mesh SIMD model requires Ω or for large values of n, approximately s ≥ 0.35n, data routing steps.

4. CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

1) Hypercube SIMD model requires $n^3=2^{3q}$ processors to multiply nxn matrices (matrix size is $2^q x 2^q$).while 2D MESH SIMD model requires n^2 processors to multiply the same.

2) HYPERCUBE model requires θ (log n) time to multiply the matrices while 2D MESH SIMD requires θ (n) time.

3) If we compare the complexity of parallel algorithm on both architectures then HYPERCUBE model is more scalable.

An algorithm is known to be scalable if its cost is in the same complexity class as an optimal sequential algorithm. A sequential algorithm takes θ (n3) time to multiply two nxn matrices.

So, a parallel algorithm on hypercube architecture is more scalable than on 2D MESH architecture.

4.2 FUTURE WORK

Finally we conclude that a parallel algorithm of matrix multiplication is more scalable on hypercube architecture in comparison to mesh architecture. In future, our aim is to develop such a architecture which is the combination of more than one architectures (like mesh and hypercube both.) which can give less complexity. The other aspect may be to develop such a parallel algorithm

which suggests the suitable architecture on which it gives less complexity in comparison to others.

REFERENCES

1.Afrati . F.C.H Papadimitriou, and G. Papageorgiou 1985. The complexity of cubical graphs. Information and Control, vol 66, pp 53-60.

2.Aho,A.,J.Hopcroft,and J.Ullman.1974.The design and analysis of computer algorithms.

3.Akl,S.G.1989.The design and analysis of parallel algorithms.

4.Amdahl,G.1967.Validity of the single processor approach to achieving large scale computing capabilities.

5.Bokhari S. 1981. On the mapping problem. IEEE Transaction on computers, vol c-30, no. 3 pp 207-214

6.Coffman E. Jr. and R. Graham. 1972. Optimal scheduling for two processor systems. Acta Informatica , vol 1 pp 200-213.

7.Cybenko G. D .W krumme and K.N venkataraman 1986. Hypercube embedding is NP complete. In proceeding of the first conference on hypercube multiprocessors, SIAM press, Philadelphia, PA, PP 148-157.

8.Dekel E. D Nassimi and S. sahni 1981. Parallel matrix and graph algorithm. SIAM journal on computing, vol 10, no 4, Nov pp 657-675.

9.Fortune,s.,and J.Wyllie.parallelism in random access machines.

10.Gentleman W.M 1978. Some complexity results for matrix computations on parallel computers. Journal of the ACM, vol 25, no 1, jan pp 112-115.

11.Havel I., and J. moravek. 1972 B-valuation of graphs. Czechoslovak Mathematical journal, vol 22 pp 338-351.

12.Livingston M., and Q.F Stout 1987. Embedding in hypercubes. In proceedings of the sixth international conference on Mathematical modeling.

13.Moravec H.P 1979. Fully interconnected multiple computers with pipelined sorting nets. IEEE transactions on computers, vol c-28, no. 10, oct pp 795-801.