HW3

1. Why is it difficult to construct a true shared memory computer? What is the minimum number of switches for connecting p processors to a shared memory with b words (where each word can be accessed independently)?

Ans: EREW PRAM is the true shared-memory computer, in which no two processors are allowed to read or write the same shared memory cell simultaneously, And EREW PRAM with p processors and a shared memory with b words where each of the p processors in a group can access any of the memory words but each word can be accessed independently.

To assure such connectivity, the total number of switches for connecting p processors to a shared memory with b words is O(pb). And constructing a switching network with this complexity is very expensive for reasonable memory. So, it's very difficult to construct a true shared-memory computer.

2. Ans:

Given that a dimensional hypercube graph ,also called the d-cube graph and commonly denoted, also called the d-cube graph and commonly denoted Qn, is the graph whose vertices are the 2^k symbols $e1, \ldots, ed$ where $ei = 0$ or 1 and two vertices are adjacent if the symbols differ in exactly one coordinate. A cycle in a graph is defined as a path originating and terminating at the same node. And the length of a cycle is the number of edges in the cycle.

We need to Show that there are no odd length cycles in a d-dimensional hypercube.

Let us consider a cycle including a node A. where A=A1, A2, A3……Ad=A. and we have to travel from node A(i) to A(i+1),$1 \leq i \leq d-1$,and the parity changes.

And we know that cycle in a graph is defined as a path originating and terminating at the same node. Therefore, A1=An, and there must be an even number of changes or n must be odd or length of cycle is even.

And any two given nodes of d-cube there is always path between them, a labelling of d-cube one way reaching node y from node x is to modify the bits of the label of x one at a time in order to transform binary bits label of x into binary bits label of y. Each time one bit is changes, so that we have crossed edge. This provides a simple way to construct a path of length atmost d between any two vertices of d-cube.

The vertex set is v =(e2). Each vertex v=(v1,v2,…,vd), $vi \in e2$ has a parity $p(v):=\sum vi \in e2(i=1 \text{to } d)$,Where by two vertices connected by an edge have different parity. Recalling that the diameter of a graph is the maximum distance between any two nodes in a graph, it follows that the length of any cycle has to be even.

3 . Now consider the problem of multiplying a dense matrix with a vector using a two loop Dot-product formulation. The matrix is of dimension 4K x 4K. (Each row of the matrix takes 16 KB of storage.) What is the peak achievable performance of this technique using a two loop dot product?

Ans: It is given (Text book question 2.2)that Level 1 cache is 32 KB , DRAM is 512 MB and Processor operating at 1 GHz.

- ⇨ Time per cycle =1/ (1* l0^9).

The latency to L1 cache is one cycle and the latency to DRAM is 100 cycles.

- ⇨ Time for 100 cycles = 100 / (1* 10^9).

The line size of cache is four words.

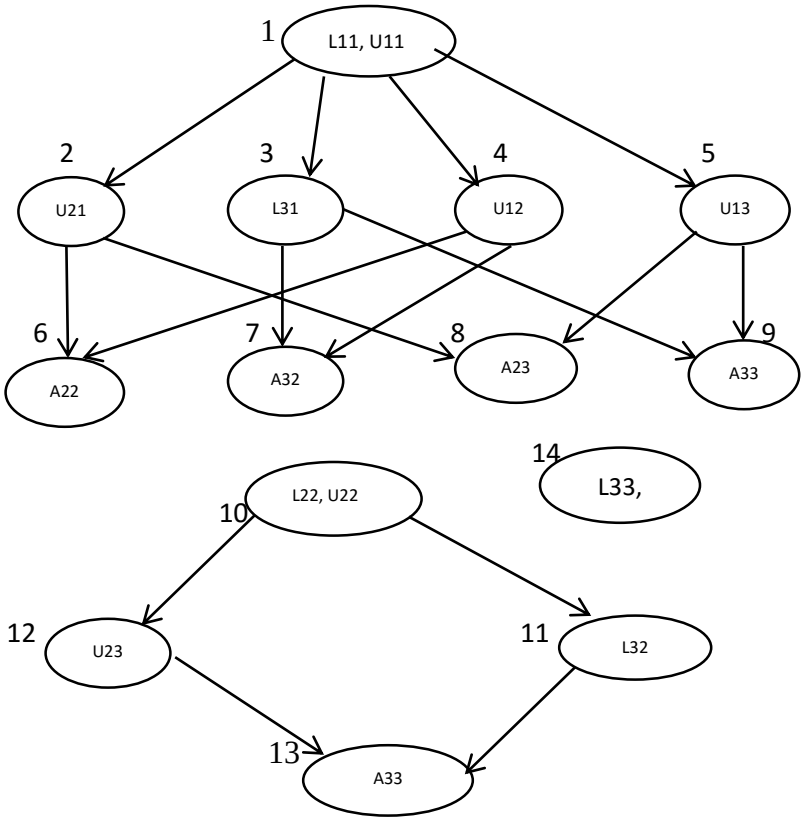
- ⇨ For one dot product, the number of words fetches by the processor is four Therefore, 4*4/ (4 * 100/ (1* 10^9)) = 4O MFLOPS.

To multiply a dense matrix with a vector using a two-loop dot-product formulation, the dimension of a matrix is 4K x 4K where Each row of the matrix takes 16KB of storage.

- ⇨ The peak achievable performance of this technique using a two loop dot product is 16/ (16*100/ 1 * 10^9) = 10 MFLOPS

4 Ans): we need to enumerate the critical paths in the decomposition of LU factorization (textbook figure 3.27)

Task dependency Graph:

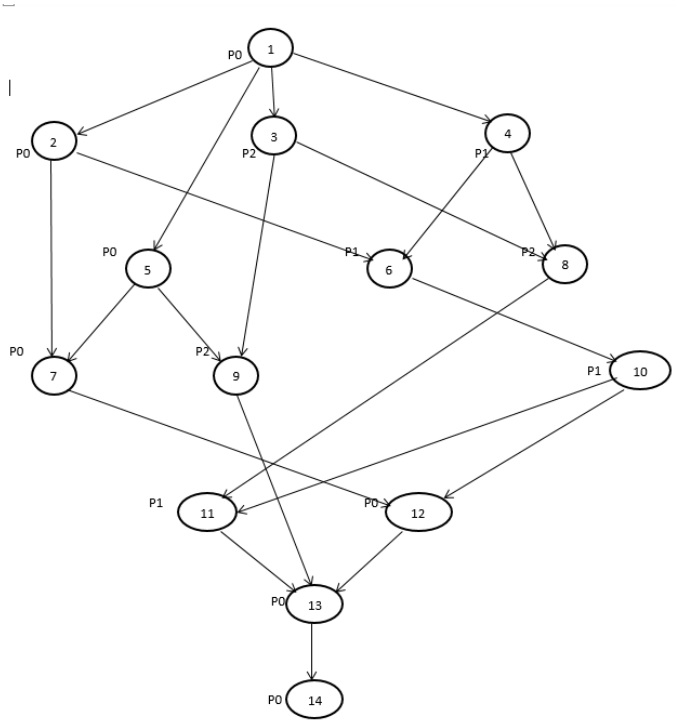


Enumeration: We have Enumeration of 11 critical paths,

1) 1, 2,6
2) 1,2,8
3) 1,3,7
4) 1,3,9
5) 1,4,6
6) 1,4,7
7) 1,5,8
8) 1,5,9
9) 10,11,13
10) 10,12,13
11) 14

5 Ans): we need to show an efficient mapping of the task dependency graph of the decomposition shown in the above figure (textbook figure 3.27) onto:

a) Three processes.
Task dependency Graph:



1.) Initially 1, 10, 14 must occur to start, 1 has four children, each with two dependencies. 10 has two children, having one dependency together. 14 has no dependencies.

2.) Next we will do 2, 3, 4 since they each have the most dependencies, at two each.

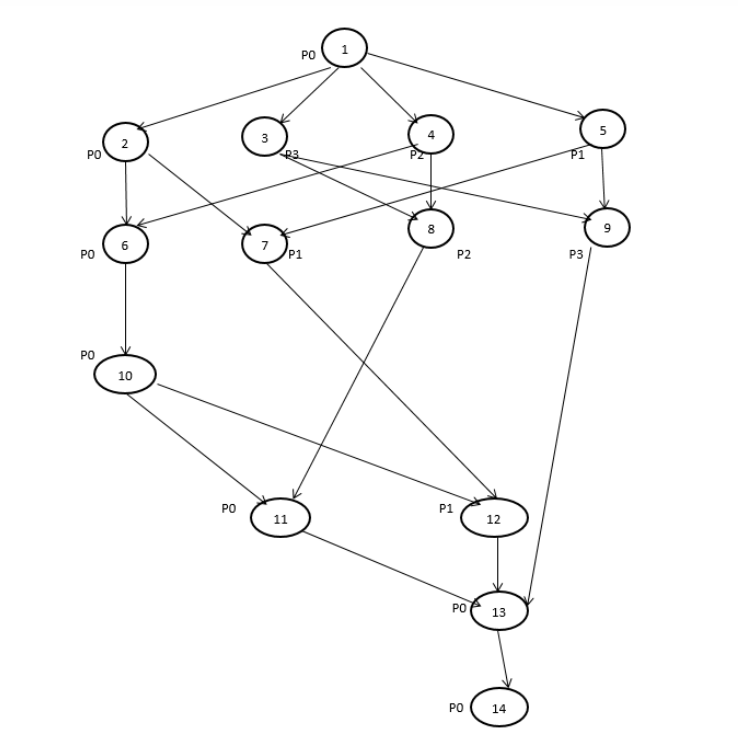
3.) Now, we will do 5,6,8, since they have a dependency,5 has two and do 6, 7 to complete two paths.

4.) Now, we will do 7,9, since they have a dependency, and do 7,9 to complete two paths.

5.) Now, we will do 11, 12 since they have a dependency and 13 to complete other path

b) Four processes

Task dependency Graph:



1.) Initially 1, 10, 14 must occur to start, with one idle processor. 1 has four children, each with two dependencies. 10 has two children, having one dependency together. 14 has no dependencies.

2.)next we will do 2, 3, 4, 5 since they each have the most dependencies, at two each.

3.) Now, we will do 11, 12 since they have a dependency, and do 6, 7 to complete two paths.

4.) Do 8, 9 to complete two paths, and 13 to complete other path.

We need to prove informally that mapping is the best possible mapping for three processes.

For Task distribution on 3 processes:

=> Maximum speedup is 14/7=2.

Efficiency = Speedup/no.of processes

=2/3

For Task distribution on 4 processes:

=> Maximum speedup is 14/7=2.

Efficiency = Speedup/no.of processes

=2/4

- ⇨ The efficiency of Task distribution on 3 processes is greater than the Task distribution on 4 processes. Here, I have optimized the mapping on 3 processes.so,we can say that mapping on 3 processes is best compared to mapping on 4 processes.

6 Ans): Given,

Using p processors, the PPT algorithm to solve Ax = d, consists of the following steps:

a) Allocate Ai, d(i) and elements aim , c(i+1)m-1 to the i th node, where 0 ? i ? p-1.

b) Use the LU decomposition method to solve the Ai .All computations can be executed in parallel and independently on p processors'

c) Send X0, Xm-1, V0,Vm-1 W0,Wm-1 from the i th node to the other nodes 0 ? i ? p-1.

d) Use the LU decomposition method to solve Zy = h on all nodes.

e) Compute in parallel on p processors.

Now, we need to provide an analysis of the algorithm regarding: Computation cost with and without pivoting and Communication cost.

In all above five operations, each processor communicates only with its two neighboring Processors. Circuit Switching and wormhole routing have reduced communication costs, it remains an intrinsic overhead in parallel computing. Moreover, the improvement in communication capabilities is low compared to dynamic improvement in processing speed. And time to communicate with nearest neighbors is linear with problem size in distributed memory computer.

Let S be the number of bytes transferred and the transfer time to communicate with neighbor can be expressed as $\alpha+S\beta$, where α is the fixed startup time and β is the incremental transmission per byte. And assume 4 bytes are used as a real number, step c and step d take $\alpha+8\beta$ and $\alpha+4\beta$ time respectively. Number of parallel operations is $17n/p-4$ and communication time is $2(\alpha+6\beta)$.

As the number of processes increases, we need to find the scalability of how the algorithm will perform when the problem size is scaled linearly with number of processes.

Let the T(p,w) be the execution time for solving a system with w work on p processes., amount of work are scaled up by factor N.,execution remains unchanged.

T(N*p,N*w)= T(p,w)

The LU decomposition method for tridiagonal systems was chosen sequential algorithm. It takes 8n-7 floating point operations. As size w increases N times w'

W'=N*8n=8n'

n'=N*n',

Let T(comp) represent the unit of a computation operation normalized to the communication time.

Time required to solve PDT algorithm with p processes is (17 n/p-4)T(comp)+ 2(α+6β).

T(N*p,N*w)= (17 n'/N*p-4)T(comp)+ 2(α+6β).By substituting w' and n',we ill get T(p,w).

Therefore, algorithm is perfectly scalable. These arguments are applied to other systems with same result.

Therefore, Computation cost with pivoting is (17 n/p-14)T(comp),and without pivoting ((17 n/p)+16p-45)T(comp).

And Communication cost is 2(α+6Tβ)

7 Ans): For the task graphs given in the following figure, we need to determine the following:

a). Maximum degree of concurrency.

b). Critical path length.

c). Maximum achievable speedup over one process assuming that an arbitrarily large number of processes is available

d) The minimum number of processes needed to obtain the maximum possible speedup.

e) The maximum achievable speedup if the number of processes is limited to 2, 4, and 8.

Graph A)

a) 8

b) 4

c) Speedup= serial/parallel

= 15/4

= 3.75

d) 8

e) 2 Processes: 15/8=1.875

4 Processes: 15/5=3

8 Processes: 15/4=3.75

Graph B)

a) 8

b) 4

c) Speedup= serial/parallel

= 15/4

= 3.75

d) 8

e) 2 Processes: 15/8=1.875

4 Processes: 15/5=3

8 Processes: 15/4=3.75

Graph C)

a) 8

b) 7

c) Speedup= serial/parallel

= 14/7

= 2

d) 8

e) 2 Processes: 14/8=1.75

4 Processes: 14/7=2

8 Processes: 14/7 =2

Graph D)

a) 2 with eager scheduling

b) 8

c) Speedup= serial/parallel

= 15/8

= 1.875

d) 2

e) 2 Processes: 15/8=1.875

4 Processes: 15/8=1.875

8 Processes: 15/8= 1.875