CS546 “Parallel and Distributed Processing”
Homework 3

Submission:
Due by 11:59pm of 10/05/2016
Total points 100 - Late penalty: 10% penalty for each day late
Please upload your assignment on Blackboard with the following name:
CS546_SectionNumber_LastName_FirstName_HW3.
Please do NOT email your assignment to the instructor and/or TA!

1. (10 points) 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: For a true shared-memory computer such as EREW PRAM with p processors and a shared memory with b words, each of the p processors in the ensemble can access any of the memory words, provided that a word is not accessed by more than one processor simultaneously. To ensure such connectivity, the total number of switches must be Θ(pb). For a reasonable memory size, constructing a switching network of this complexity is very expensive. Thus a true shared-memory computer is impossible to realize in practice.

2. (10 points) A d-dimensional hypercube graph, also called the d-cube graph and commonly denoted Q_n, is the graph whose vertices are the 2^k symbols $e_1, \ldots, e_d$ where $e_i = 0$ or 1 and two vertices are adjacent if the symbols differ in exactly one coordinate (see more here).
   A cycle in a graph is defined as a path originating and terminating at the same node. The length of a cycle is the number of edges in the cycle. Show that there are no odd-length cycles in a $d$dimensional hypercube.

Ans: Consider a cycle A1, A2, . . . , Ak in a hypercube. As we travel from node Ai to Ai+1, the number of ones in the processor label (that is, the parity) must change. Since A1 = Ak, the number of parity changes must be even. Therefore, there can be no cycles of odd length in a hypercube.

3. (10 points) 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 dotproduct based matrix-vector product?

$$\begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} L_{1,1} & 0 & 0 \\ L_{2,1} & L_{2,2} & 0 \\ L_{3,1} & L_{3,2} & L_{3,3} \end{pmatrix} \cdot \begin{pmatrix} U_{1,1} & U_{1,2} & U_{1,3} \\ 0 & U_{2,2} & U_{2,3} \\ 0 & 0 & U_{3,3} \end{pmatrix}$$

1: $A_{1,1} \rightarrow L_{1,1}U_{1,1}$	6: $A_{2,2} = A_{2,2} - L_{2,1}U_{1,2}$	11: $L_{3,2} = A_{3,2}U_{2,2}^{-1}$
2: $L_{2,1} = A_{2,1}U_{1,1}^{-1}$	7: $A_{3,2} = A_{3,2} - L_{3,1}U_{1,2}$	12: $U_{2,3} = L_{2,2}^{-1}A_{2,3}$
3: $L_{3,1} = A_{3,1}U_{1,1}^{-1}$	8: $A_{2,3} = A_{2,3} - L_{2,1}U_{1,3}$	13: $A_{3,3} = A_{3,3} - L_{3,2}U_{2,3}$
4: $U_{1,2} = L_{1,1}^{-1}A_{1,2}$	9: $A_{3,3} = A_{3,3} - L_{3,1}U_{1,3}$	14: $A_{3,3} \rightarrow L_{3,3}U_{3,3}$
5: $U_{1,3} = L_{1,1}^{-1}A_{1,3}$	10: $A_{2,2} \rightarrow L_{2,2}U_{2,2}$	

Ans: The best strategy is when the vector y is in the cache. This means that for each iteration of the loop, only 1 cache line must be fetched (for the matrix). Since, each iteration involves 2 FLOPs. Thus, peak performance for a cache line of 4 words is = 8 FLOPs/ cache line fetch = 8 FLOPs/ 100 ns = 80 MFLOPS.

Here, assuming the multiplication algorithm is the following:

```
for(i  0; i < SIZE; i++)
    for(j = 0; j < SIZE; j++)
        for(k = 0; k < SIZE; k++)
            z[i][j] += x[i][k] * y[k][j];
```

Then, 5 cache lines will have to fetched, one for the matrix X and 4 for the matrix Y (since the access is performed in a column-major fashion). Thus, the peak performance = 8 FLOPs/500 ns = 16 MFLOPS.

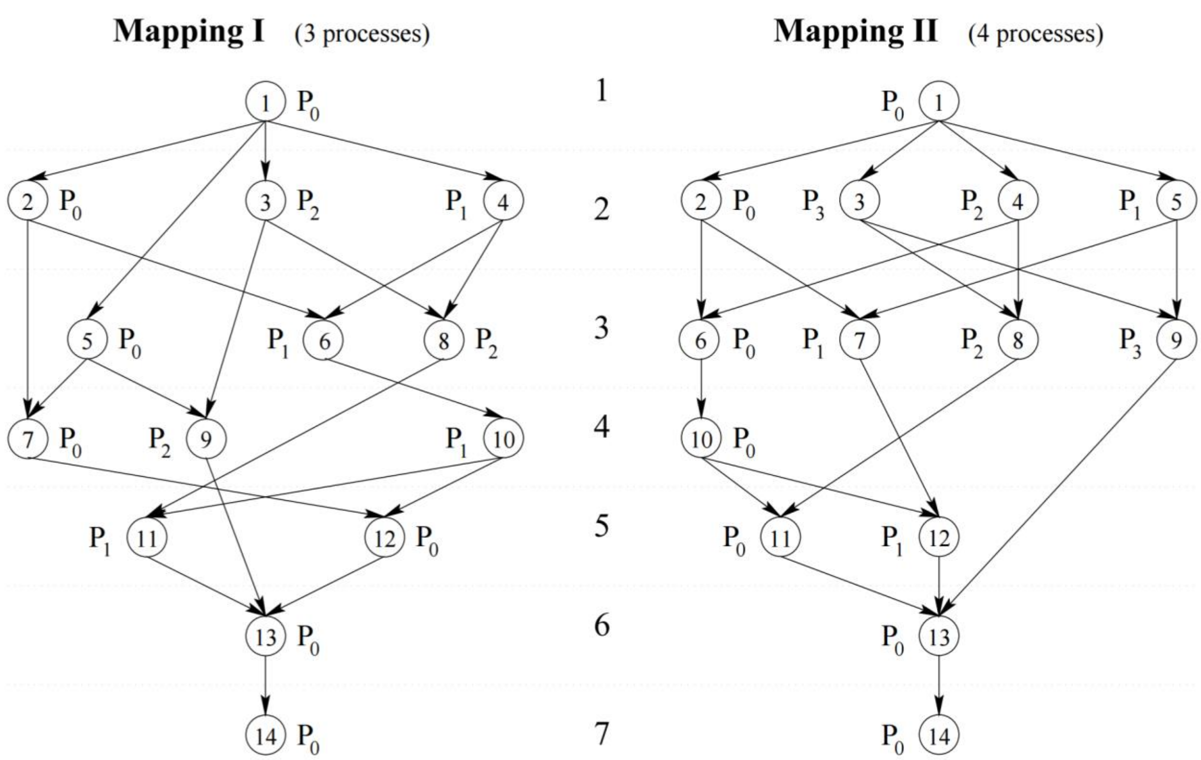
4. (10 points) Enumerate the critical paths in the decomposition of LU factorization shown in the following figure (textbook figure 3.27).

Ans 1,2,6,10,11,13,14, 1,2,6,10,12,13,14, 1,4,6,10,11,13,14, 1,4,6,10,12,13,14

5. (20 points) 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.
   b) Four processes

   Prove informally that your mapping is the best possible mapping for three processes.

Ans:



Using three processes achieves the maximum possible speedup of 2.

6. (20 points) In class we saw the Parallel Parition LU (PPT) algorithm for solving tridiagonal linear systems. 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 `ith` node, where `0 ? i ? p-1`.
   b) Use the LU decomposition method to solve

   $$A_i[\tilde{x}^{(i)}, v^{(i)}, w^{(i)}] = [d^{(i)}, a_{im}e_0, c_{(i+1)m-1}e_{m-1}]$$

   All computations can be executed in parallel and independently on p processors.
   c) Send $\tilde{x}_0^{(i)}, \tilde{x}_{m-1}^{(i)}, v_0^{(i)}, v_{m-1}^{(i)}, w_0^{(i)}, w_{m-1}^{(i)}$ from the `ith` 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

   $$\Delta x^{(i)} = [v^{(i)}, w^{(i)}]\begin{bmatrix} y_{2i-1} \\ y_{2i} \end{bmatrix}$$

   $$x^{(i)} = \tilde{x}^{(i)} - \Delta x^{(i)}$$

   You are asked to provide an analysis of the algorithm regarding:
   a) Computation cost with and without pivoting
   b) Communication cost

Ans : If n= order of each system

p is the number of processors

α - the communication start time

β - the transmission time

τ - the computing speed

Without pivoting

Step a) per processor sends data to 4 nodes hence complexity is p(α+4β) : O(p)

Step b) O($(n/p)^3$)

Step c) per processor sends data to 4 nodes hence complexity is p(α+4β): O(p)

Step d) O($(n/p)^3$)

Step e) O($(n/p)^3$)

Step f) Getting all result on root node : per processor sends data to 4 nodes hence complexity is p(α+4β) : O(p)

Computation cost with pivoting= O($(n/p)^3$)

Communication cost = O(p)

With pivoting

Step a) per processor sends data to 4 nodes hence complexity is p(α+4β) : O(p)

Step b) O((n/p))

Step c) per processor sends data to 4 nodes hence complexity is p(α+4β) : O(p)

Step d) O((n/p))

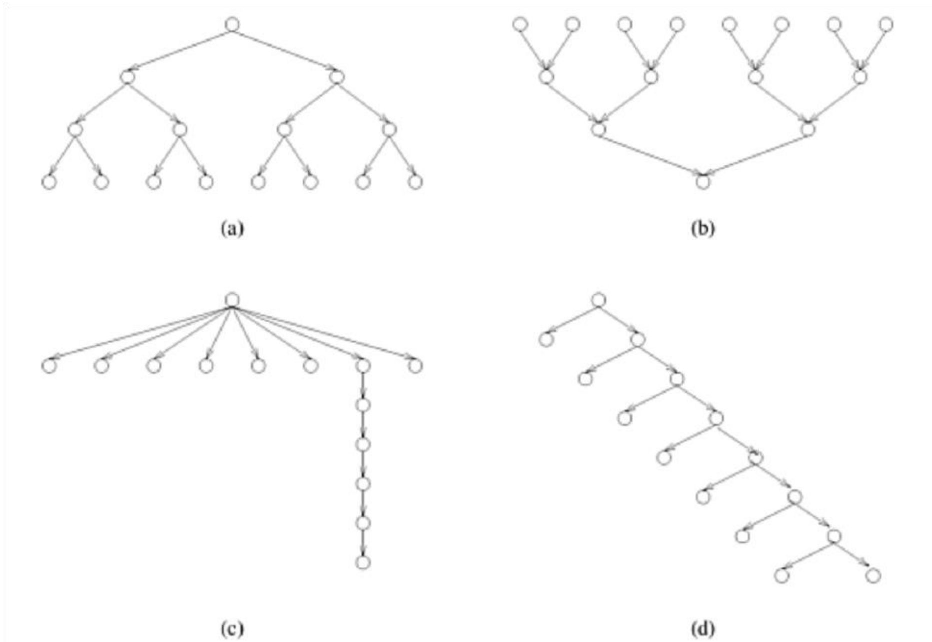
Step e) O((n/p))

Step f) per processor sends data to 4 nodes hence complexity is p(α+4β) : O(p)

Computation cost with pivoting= O((n/p))

Communication cost = O(p)

7. (20 points) For the task graphs given in the following figure, 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.



Ans: We assume eacg node to be of unit weight.

1. (a) 8, (b) 8, (c) 8, (d) 8.
2. (a) 4, (b) 4, (c) 7, (d) 8.
3. (a) 15/4, (b) 15/4, (c) 2, (d) 15/8.
4. (a) 8, (b) 8, (c) 3, (d) 2.
5. Number of parallel processes limited to 2: (a) 15/8, (b) 15/8, (c) 7/4, (d) 15/8.

Number of parallel processes limited to 4: (a) 3, (b) 3, (c) 2, (d) 15/8.

Number of parallel processes limited to 8: (a) 15/4, (b) 15/4, (c) 2, (d) 15/8