Eastern University, Sri Lanka

Department of Mathematics

Special Degree Examination in Computer Science - 2013/2014

CS406: Parallel Computing

- Answer ALL questions. (This paper has 6 questions on 6 pages.)
- Time allowed: **Three Hours.**
- Start a new page for each question.
- Allocate your time wisely, the point value of each part shown in square brackets.

Efficient mechanisms for routing a message to its destination are critical to the performance of parallel system.

(a) One commonly used routing technique in a *hypercube* is called *E-cube routing.*

Explain clearly, how the E-cube routing mechanism sends a message from 1000 to 0101 in a four-dimensional hypercube. [20%]

(b) The *omega network* is an interconnection network composed of $\log p$ stages, which connects p processing nodes to p memory banks.

i. At each stage of an omega network, a *perfect shuffle* interconnection pattern feeds into a set of $p/2$ switches or switching nodes.

(α) Each switch has two connection modes: *straight-through* and *cross-over.* ***Illustrate*** these two modes. [10%]

(β) ***Sketch*** a schematic diagram of an omega network with <u>four</u> processing nodes and <u>four</u> memory banks. [30%]

ii. Consider the omega network with four processing nodes and four memory banks. ***Explain*** clearly, how the routing mechanism determines the route from the source 10_2 to the destination 11_2. [20%]

iii. By considering a suitable example, ***show*** that the omega network is a blocking network. [20%]

2. An abstraction used to express the *dependencies* among *tasks* and their relat
execution is known as a *task-dependency graph.*

The following table shows a relational database of students' details:

ID	CD	Class	Year	Med
S1001	3	1st	2011	Eng
S1002	4	2nd-Up	2012	Tam
S1003	3	2nd-Low	2010	Eng
S1004	4	1st	2012	Eng
S1005	4	2nd-Low	2012	Sin
S1006	4	2nd-Low	2010	Eng
S1007	3	2nd-Up	2011	Tam
S1008	3	1st	2011	Sin
S1009	4	1st	2012	Eng
S1010	3	2nd-Up	2012	Eng

Where,
ID - Students' ID number
CD - Duration of the course (i
Med - Medium

Consider the following query for the above given relational database:

```
(Med=Eng) AND (CD=4) AND ( (Class=1st) OR (Class=2nd-Up
```

(a) i. For this query, ***construct*** the most efficient task dependency grap
ii. ***Determine*** the maximum degree of concurrency and the average
currency for your graph obtained in part (a.i).
iii. Without considering the average degree of concurrency, ***prove*** th
(obtained in part (a.i)) is the most efficient one.

(b) ***Propose*** an efficient process-task mapping for your graph obtained
Explain your mapping clearly.

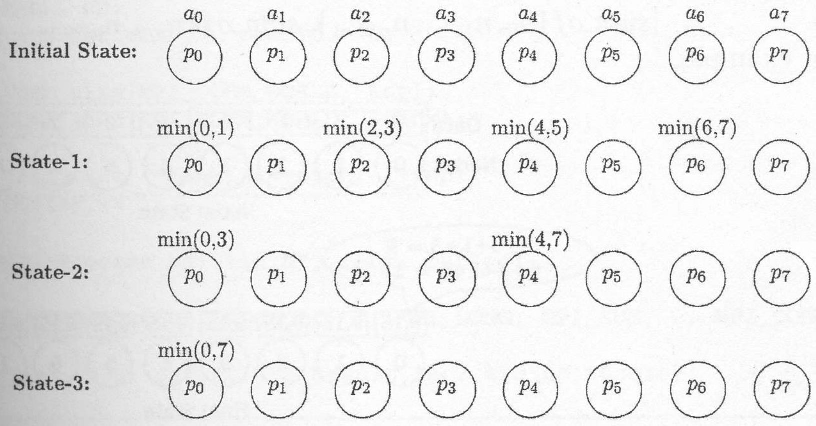
(a) ***Define*** what is meant by a *cost-optimal system* for a problem, and ***deduce*** that a cost-optimal parallel system has an *efficiency* of $\theta(1)$. [15

(b) ***State*** clearly, what is meant by *scalablity* of a parallel algorithm. [5

(c) Consider a problem of finding minimum among the list of n numbers $(a_0, a_1, ...a_{n-1})$, using p processing elements.

The figure shown below illustrates a procedure for finding the minimum in the list of 8 numbers using 8 processing elements in three steps. The minimum is determined by propagating partial solutions up a logical binary tree of processing elements.

The processing elements are labelled $p_0, p_1, ...p_7$, each processing element is initially assigned one of the numbers, $min(i, j)$ denotes the minimum of numbers $(a_i, a_{i+1}, ...a_j)$.



i. ***Determine*** the *parallel runtime* T_P. [20

ii. Suppose the *serial runtime* is $\theta(n)$, ***show*** that the above method is not cost optimal. [20

iii. ***Devise*** a cost optimal method to find the minimum of n numbers on p processing elements such that $p < n$, and ***show*** that your method is cost optimal.
Discuss the scalablity of the method. [40

4. (a) *All-to-all broadcast* is used in several important parallel algorithms.

   i. Clearly ***illustrate***, how the *all-to-all broadcast* can be performed on

   hypercube.

   ii. ***Give*** an algorithm for implementing *all-to-all broadcast* on a *d-di*

   percube.

   (b) Consider a d-dimensional p-node hypercube with one number on each n

   node P_i holds the number n_i.

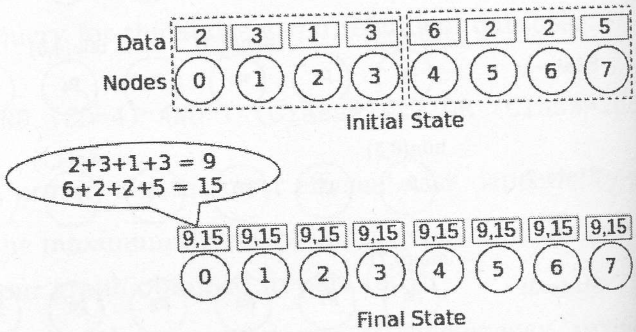
   The task is to find an identical pair of values by all the p nodes, such th

   will be the sum of numbers from each $(d-1)$-dimensional sub-hypercu

   That is, every node P_i will have an identical pair

   $$[sum\_of\{n_0, n_1, ..., n_{p/2-1}\}, sum\_of\{n_{p/2}, n_{p/2+1}, ..., n_{p-1}\}]$$

   For example,



   Construct a parallel algorithm to perform the above mentioned task.

   Hint: Carefully observe the communication pattern illustrated in part (a

   ify the algorithm obtained in part (a.ii).

(a) A message exchanging problem is shown below:

```
P0                          P1

send(&a, 1, 1);             send(&a, 1, 0);
receive(&b, 1, 1);          receive(&b, 1, 0);
```

If the send and receive operations are implemented using a *blocking non-buffered* protocol, there is a possibility for a deadlock. ***Explain*** clearly, how bufferes can be used to avoid this problem. [25%]

(b) Consider the following code segment:

```
void CS(long int stop){
// Assume that "stop" is a non negative integer

    int rank, np;
    long int buffer = 0, result = 0;

    MPI_Comm_size(MPI_COMM_WORLD, &np);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    for (long int i=rank; i<=stop; i+=np){
        buffer += i;
    }
    printf("Process %d: %ld\n", rank, buffer);

    MPI_Reduce(&buffer, &result, 1, MPI_LONG, MPI_SUM, 0, MPI_COMM_WORLD);

    if (rank == 0) printf("\n Result is : %ld\n",result);
}
```

i. If the number of processes in "MPI_COMM_WORLD" is 3, ***trace out*** the final result for the function call `CS(5)`. ***State*** the purpose of this code. [35%]

ii. ***Modify*** this code to calculate the sum of any range of consecutive natural numbers from a "`start`" number to a "`stop`" number. [40%]

For example, if "`start = 5`" and "`stop = 12`", then

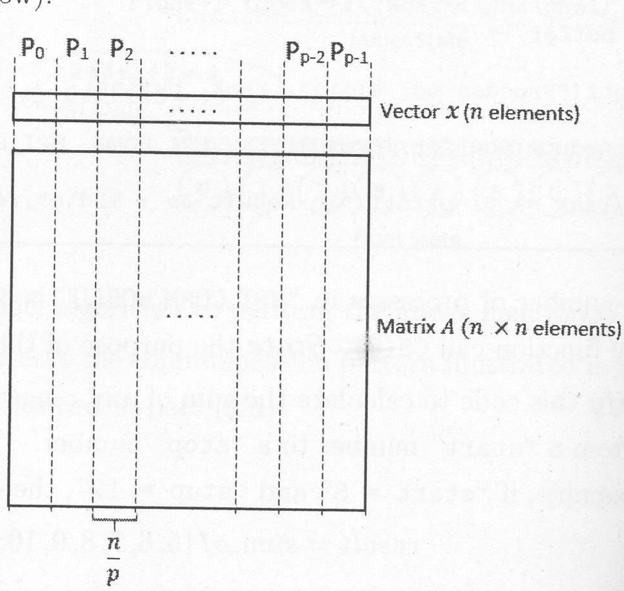
$$result = sum\_of\{5, 6, 7, 8, 9, 10, 11, 12\}$$

6. (a) Consider the following code segment:

```
//assume that the "root" has a matrix "A" of size r*c, and r is divisible b
//where (r = number of rows) and (c = number of columns)

MPI_Comm_size( MPI_COMM_WORLD, &np );
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

int nelts = c * r/np;
int local_A[r/np][c];
int local_x[r/np];

MPI_Scatter(A, nelts, MPI_INT, local_A, nelts, MPI_INT, root, MPI_COMM_WORL
```

Suppose the **root** has a vector x of r elements, and you want to solve the matrix-problem $Ax = y$. **Write** the remaining part of the code segment to solve th parallally by np processors, then obtain the final result y in the node "**root**".

(b) Consider a problem in which a matrix A of size $n \times n$ and a vector x of n el have been distributed among p number of processes such that each process ge consecutive columns of A and the elements of vector x that correspond to these co (see the figure below).



Assume that $n \times (n/p)$ elements of A and (n/p) elements of x are stored in and "*local_x*" respectively.

Write the remaining part of the code segment given below to solve the matr problem $Ax = y$, and obtain the resultant vector y entirely in P_0.

```
MPI_Comm_size(comm, &npes);
nlocal = n/npes;
int local_y[n];
```