Eastern University, Sri Lanka

Department of Mathematics

Special Degree Examination in Computer Science -

2009 / 2010 (September / October 2011)

CS 406: Parallel Computing

Answer all questions

This paper has 4 questions and 4 pages

e allowed: Three Hours

(a) ***Explain*** briefly how pipelining of instructions enables faster execution.

(b) ***State*** clearly what you understand by each of the following terms:

i. true data dependency,　　ii. resource dependency

iii. branch dependency,　　iv. dynamic instruction issue

(c) With the aid of suitable examples, ***explain*** briefly how issues that may arise in each of the cases in (b) above can be solved.

(d) With regard to memory hierarchy, ***define*** each of the following terms clearly:

i. latency　　ii. bandwidth　　iii. cache line

(e) Consider a system with a level 1 cache (L1-cache) of 32 KB and DRAM of 512 MB with the processor operating at 2 GHz. The latency to L1-cache is one cycle and the latency to DRAM is 100 cycles. In each memory cycle, the processor fetches four words - that is, cache line size is four words. Assume an optimal cache placement policy, where necessary.

i. ***What*** is the DRAM latency in seconds?

ii. ***Determine*** the peak achievable performance of a dot product of two vectors of 32-bit (1 word) integers?

iii. Now consider the following code for multiplying a dense matrix of 5120×5120 elements with a vector of 5120 elements, where each element of the matrix as well as of the vector is a 32-bit (1 word) integer.

```
for (i = 0; i < dim; i++)
   for (j = 0; i < dim; j++)
      c[i] += a[i][j] * b[j];
```

(α) If matrix is laid out in a column-major fashion, ***what*** is the peak achievable performance? (assume that the entire vector is already cached.)

(β) ***Show*** how you would restructure the above code segment to remove stridden access so that the performance can be improved.

(γ) ***Determine*** the peak achievable performance for your restructured code segment.

2. (a) With the aid of suitable diagrams, clearly ***describe*** the SIMD architecture
MIMD architectures. ***State*** their advantages and disadvantages.

(b) ***State*** what you understand by *parallel random access machine* (PRAM).

(c) Clearly ***distinguish*** the following PRAM models:

i. EREW-PRAM   ii. CREW-PRAM   iii. ERCW-PRAM   iii. CRC

(d) ***State*** clearly what is meant by *multistage interconnection networks.*

(e) A commonly used multistage connection network is the *omega network*. This consists of $\log p$ stages, where p is the number of inputs (processing nodes) the number of outputs (memory banks). Each stage of the omega network co an interconnection pattern that connects p inputs and p outputs; a link exists input i and output j satisfying

$$j = \begin{cases} 2i & \text{for } 0 \leq i \leq p/2 - 1 \\ 2i + 1 - p & \text{for } p/2 \leq i \leq p - 1 \end{cases}$$

i. ***Draw*** a diagram to show an omega network for eight processors and eight banks.

ii. Briefly ***explain*** how the routing of data in an omega network is accomplis

(f) The *butterfly network* is another multistage connection network composed of log (as the omega network). In a butterfly network, each switching node i at a l connected to the node i at level $l + 1$ and to a switching node j at level $l + 1$ su it differs from i only at the l^{th} most significant bit. That is, switching node i a is connected to element j at level $l + 1$ for $j = i$ and for $j = i \oplus (2^{\log p - l})$.

Draw a diagram to show a butterfly network for eight processors and eight banks.

(a) ***Define*** the terms *parallel runtime*, *speedup*, and *efficiency* of a parallel algorithm.

(b) ***State*** clearly what you understand by the following terms:

i. task-dependency graph, ii. maximum degree of concurrency,

iii. critical path length

(c) Consider the following scenario:

Suppose n candidates appear in an examination, each of whom answers all k questions. Let the time to correct an answer book be kp, where p be the time to correct one question.

i. Suppose k teachers decided to independently and simultaneously do the correction. The answer books are equally distributed among the k teachers in time kq, where q be the distribution time for each teacher. Each teacher corrects all the questions in their answer books.

(α) ***Determine*** the time to correct all the answer books by k teachers.

(β) ***Determine*** the speedup and efficiency due to the *data parallelism*.

ii. Consider an alternative way:

The answer books are kept as one pile, and k teachers decided to co-operatively correct one question each: The first teacher corrects answer to Q_1 of the first answer book and passes it to the second teacher who starts correcting Q_2. After correcting Q_2, the second teacher passes the book to the third teacher to correct Q_3, and so on. After passing the book to the second teacher, the first teacher takes a new book from the pile, and passes it to the second teacher after correcting Q_1.

In this way, from the k^{th} answer book onwards all the teachers are busy correcting. All are busy until after the first teacher finishes correcting Q_1 in the last answer book. The work would be complete when the last teacher finishes correcting Q_k of the last book.

Determine the speedup and efficiency due to pipeline processing.

iii. ***Compare*** the speed-up times determined in part(i.β) and part(ii) above.

iv. ***Draw*** separate task-dependency graphs for both cases described in parts (i) and (ii). ***Compare*** the maximum degree of concurrency and critical path lengths.

4. (a) With the aid of diagrams, clearly ***distinguish*** the three collective communic erations:

   i. `MPI_Scatter` ii. `MPI_Gather` ii. `MPI_Reduce`

   (b) Clearly ***explain*** how the *scatter* and *gather* are performed on *a hypercube* topo ***determine*** the parallel runtime in each case.

   (c) Clearly ***distinguish*** the *block-cyclic distribution* and *cyclic distribution* of an n among p processes.

   (d) Consider a matrix A of size $n \times n$. Initially, the process 0 has a vector b of si all of n columns of the matrix A are distributed among n processes, 1 column ***Write*** a single MPI program to accomplish the following tasks:

   i. Distribute the vector b among the n processes such that the process i re element $b[i]$;

   ii. Calculate a vector *localY* in each process such that

   $$localY = elementb \times localA$$

   where *localA* is the column of elements and the *elementb* is the element of b from process 0.

   iii. Calculate a vector y in the process 0 such that $y_i = \sum_{p=0}^{n-1} localY_i[p]$ for $0 \leq i$

   That is, the i^{th} element of the vector y is the summation of all the i^{th} eleme *localY* of all the processes.