Eastern University, Sri Lanka

Department of Mathematics

Special Degree Examination in Computer Science 2010/2011 (March 2014)

CS 406: Parallel Computing

Answer all questions

This paper has 6 questions in a total of 4 pages

ime allowed: Three Hours

1. (a) ***Define*** the following terms: *latency*, and *cache hit ratio*. [

(b) i. Consider a program attempts to solve a problem instance of size W in a single processor system with the processor operating at 2 GHz. The latency to the cache is two cycles and the latency to the DRAM is 200 cycles.

Assume that the program has a cache hit ratio of 85%.

(α) ***Calculate*** the DRAM latency and the cache latency in seconds. [

(β) If the computation is memory bound and performs one FLOP for every two memory access, ***determine*** the computation rate in MFLOPS. [2

ii. Suppose, the same problem instance of size W has been taken for the execution on a two-processor parallel system with the above mentioned processor and the DRAM. Assume that the two processors are effectively executing half of the problem instance (i.e., size $W/2$).

Now the cache hit ratio is 92%, 6% of the remaining data comes from local DRAM, and the other 2% comes from the remote DRAM with a latency of 400ns.

(α) ***Determine*** the overall computation rate in MFLOPS. [2

(β) ***Calculate*** the *speedup*. [1

(γ) There is an anomaly in the *speedup* -***what*** is it? ***Describe*** the cause for this anomaly. [2

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

(b) The image smoothing works as replacing the value of every pixel in an image by the average of the gray levels in the neighbourhood defined by a filter mask.

Suppose you are requested to devise an algorithm to smooth an $n \times n$ RGB image using a 5×5 filter mask.

i. ***Determine*** the serial runtime.

ii. If you have a parallel system with p processing elements, clearly ***describe*** a parallel algorithm to smooth an $n \times n$ RGB image by the 5×5 filter mask.

iii. Let each multiply-add takes time t_c, the per-word transfer takes time t_w, and the startup time for the data transfer is t_s, and assume that every pixel represents one word of RGB data. ***Determine*** the parallel runtime and the efficiency of your method given for part (b.ii).

3. (a) Clearly ***describe*** what is meant by *cost-optimal system.*

(b) Consider the problem of adding N numbers by P students $[S_1, S_2, \dots, S_P]$, sitting on a straight line. Initially, the numbers to be added are divided equally among all students, and at the end of the computation, the first student finds the sum of all the numbers. Suppose the students calculate sums on the blackboard so that everyone can see the others results as soon as they are ready.

i. ***Suggest*** an optimal parallel approach to find the total sum.

Hint: Think of this as a situation where anybody can communicate with anybody and there is a better way than S_1 adds all the P local sums.

ii. ***Show*** that your approach given in part (i) is cost optimal.

(a) With the aid of diagrams, ***explain*** how the *one-to-all broadcast* and *all-to-all broadcast* are performed on a *hypercube topology.* [25%]

(b) All-reduce operation is a communication operation, which is identical to performing an all-to-one reduction followed by a one-to-all broadcast of the result.

i. ***Show*** that there is a faster way to perform all-reduce on a d-dimensional hypercube by using the communication pattern of all-to-all broadcast. [30%]

ii. ***Write*** pseudo code to implement all-reduce operation on a d-dimensional hypercube. [27%]

iii. ***Determine*** the total communication time for the all-reduce algorithm given in part (b.ii). [18%]

Consider the following MPI code:

```
#include "mpi.h"
#include <stdio.h>
void main (int argc, char *argv[]) {
     int v, nv, pr, np, right, left, other, sum, i;
     MPI_Status  st;

     MPI_Init(&argc, &argv);
     MPI_Comm_rank(MPI_COMM_WORLD, &pr);
     MPI_Comm_size(MPI_COMM_WORLD, &np);
     right = pr + 1;
     if (right == np) right = 0;
     left = pr - 1;
     if (left == -1) left = np-1;

     sum = 0;
     value = pr;
     for( i = 0; i < np; i++) {
          MPI_ISend(&v, 1, MPI_INT, right, 1, MPI_COMM_WORLD);
          MPI_Recv(&nv, 1, MPI_INT, left, 1, MPI_COMM_WORLD, &st);
          if (nv <= pr) sum = sum + nv;
          v = nv;
     }
     printf ("Node number %d:\t Sum = %d\n", pr, sum);
     MPI_Finalize();
     return;
}
```

[Question 5 continues on the next page]

(a) ***Write*** a possible output of each process.

(b) ***Explain*** clearly what will happen if we replace `MPI_ISend` with `MPI_Send`.

(c) ***Explain*** how you would modify this code, and use `MPI_Send`, rather than `MPI_ISend`.

(d) ***Modify*** the original code to print the result of the following function in node 0:

$0^2 + 1^2 + 3^2 + 6^2 + 10^2 + ... + \{0 + 1 + 2 + ... + (np - 1)\}^2$

6. (a) ***Write*** the syntax and ***describe*** each of the following MPI functions:

   i. `MPI_Send` ii. `MPI_Receive` iii. `MPI_Sendrecv_replace`

   (b) A ring network of p processes work as follows:

   - Every process sends a message to its neighbor in a cyclic fashion.
   - At start, the message content is the ranks of the processes, thereafter every process sends the message to the next what it has received recently.
   - This will be carried for $(p - 1)$ times.
   - The message tag is the receiver's *rank* number.
   - Each receiver prints out the step number and the received element.

   i. Briefly ***discuss*** about the possibility for a *deadlock* situation.

   ii. ***Write*** a *safe* MPI code segment to implement the problem described above using `MPI_Send` and `MPI_Recv`.

   iii. ***Show*** how you would modify your code to use `MPI_Sendrecv_replace` rather than `MPI_Send`.