

THE AUSTRALIAN NATIONAL UNIVERSITY
First Semester Examination – June 2004

COMP4300
Parallel Systems

Study Period: 15 minutes

Time Allowed: 3 hours

Permitted Materials: Calculator

The exam will be out of 75.

Answer any THREE questions.

Answer questions 1+2+3 your maximum mark will be 70/75

Answer questions 1+2+4 your maximum mark will be 75/75

Answer questions 1+3+4 your maximum mark will be 75/75

Answer questions 2+3+4 your maximum mark will be 80/75

Any mark over 75 will be taken as full marks in the exam

Point form is recommended for ‘descriptive’ questions.

*Clarity and conciseness in answers will be highly valued;
marks may be lost for supplying irrelevant information in
answers.*

Question 1 [20 marks]

The openMP parallel programming model includes the following four directives for work-sharing and synchronization:

```
#pragma omp for schedule(static, chunksize)
for(i=0;i<N;i++){ do something! }
```

```
#pragma omp for schedule(dynamic, chunksize)
for(i=0;i<N;i++){ do something! }
```

```
#pragma omp barrier
```

```
#pragma omp critical
{critical stuff}
```

- (a) Briefly outline what each of the above directives does. [4 marks]
- (b) Nothing comes for free! Associated with each of the above constructs there is an overhead. Outline how you would set about measuring the overhead of each of the above directives. [16 marks]

Question 2 [25 marks]

You are writing an image processing program, where the image is represented as a 2D array of pixels. The basic iteration in this computation looks like:

```
for i = 1 to 1022
  for j = 1 to 1022
    newA[i,j] = (A[i,j-1] + A[i-1,j] + A[i,j+1] + A[i+1,j])/4;
```

Assume both **A** and **newA** are 1024×1024 matrices of four-byte single-precision floating-point numbers stored in row-major order such that (eg) **A**[i,j] and **A**[i,j+1] are at consecutive addresses in memory as are elements **A**[0,1023] and **A**[1,0]. You should assume that **A** starts at memory location 0 while **newA** is stored immediately after **A** (ie. at memory location $1024 \times 1024 \times 4$). You are writing this code for a 32-processor machine that uses a bus and a snoopy cache coherency protocol. The cache on each processor is 32KB with a cache line size of 64bytes.

- (a) Briefly outline what is meant by cache coherency and how cache coherency is implemented using snooping. [4 marks]
- (b) You first try allocating 32 rows of the matrix to each processor in an interleaved assignment. Discuss this from the perspective of bus traffic and cache coherency. [5 marks]
- (c) Repeat the above but this time assign 32 contiguous rows of the matrix to each processor. Is this a better parallelization scheme? [5 marks]
- (d) Repeat the above but this time assign 32 contiguous columns instead of rows to each processor. Is this a better parallelization scheme? [5 marks]
- (e) On closer inspection of your hardware you note that the cache is direct-mapped. This will impact on performance. Explain why this is and any differences between its effect on the three different algorithms. Is it possible that for this problem making slight modifications to the data layout would avoid the limitations of a direct mapped cache? [6 marks]

Question 3 [25 marks]

The following C routine is designed to iterative solve a system of linear equations of the form

$$Ax = b$$

where A is an $n \times n$ matrix of known values, b is an n element vector of known values, and x is the n element vector that you wish to determine.

```
void solve(int max_iter, double threshold, int n,
           double a[], double b[], double x[], double new_x[])
{
    int i, iter, j;
    double sum, diff, max_diff;

    for (i = 0; i < n; i++) x[i] = b[i];

    for (iter = 0; iter < max_iter; iter++){
        for (i = 0; i < n; i++){
            sum = -a[i*n+i]*x[i];
            for (j = 0; j < n; j++){
                sum = sum + a[i*n+j]*x[j];
            }
            new_x[i] = (b[i] - sum)/ a[i*n+i];
        }
        max_diff=(x[0]-new_x[0])*(x[0]-new_x[0]);
        for (i = 0; i < n; i++) {
            diff=(x[i]-new_x[i])*(x[i]-new_x[i]);
            max_diff = (max_diff > diff) ? max_diff : diff;
            x[i] = new_x[i];
        }
        if (max_diff < threshold ) break;
    }
    printf("Final error %20.12f\nTotal Iterations %5d\n",max_diff, iter);
}
```

The code uses the Jacobi approach where

$$x_i = \frac{1}{a_{i,i}} [b_i - \sum_{j \neq i} a_{i,j} x_j]$$

- (a) Discuss how you would parallelize the above code for a distributed memory parallel computer using MPI. In your code all $O(n^2)$ data structures must be stored in a distributed fashion, but you are permitted to replicate all $O(n)$ or smaller data structures. Your discussion is expected to include a justification for your design decisions. **[20 marks]**
- (b) Discuss how you would expect your code to perform as a function of number of processors. **[5 marks]**

Question 4 [30 marks]

As an author of the *mympi* program your attention is drawn to a paper by Rolf Rabeneifner on the “Optimization of Collective Reduction Operations” to be presented at the 2004 International Conference on Computational Science in Krakow that you are currently attending. As a result you cancel your sightseeing tour of Wawel Castle and endure a day at the conference! From the talk there appears to be some merit to this new scheme, but due to jet lag you were not able to follow all the details. You resolve to look at the paper on your return to Australia. When you come to do this, however, you find that the pdf file has been corrupted and several of the key performance formula are missing from pages 4 and 5.

- (a) The final performance characteristics of the allreduce (and reduce) implementations are still given. Missing, however, are the key terms used to derive the final formula. These missing terms are labelled as T1-T5. Derive expressions for T1-T5 showing clearly how you obtain these terms. Then show how they add together to give the total value for the non-power-of-two allreduce (Tfinal). (You should note that a one way communication is modeled as $\alpha_{uni} + n\beta_{uni}$ while a two-way exchange of data is also modeled via a single expression $\alpha + n\beta$. The ratio of $\alpha_{uni}/\alpha (= f_\alpha)$ can range from 0.5 to 1.0 depending on whether the communication infrastructure supports full duplex or not.) [10 marks]
- (b) Sketch code to implement figure 1. (I am not looking for syntactically correct code, but the essence of what you do must be clearly demonstrated.) [15 marks]
- (c) Provide a few comments on how you would expect this algorithm to perform on an Ethernet based cluster of workstations. (ie. do you think that it will be superior to the rather poor global reduction operations currently programmed in *mympi*!!!). [5 marks]
-