

THE AUSTRALIAN NATIONAL UNIVERSITY
First Semester Examination – June 2003

COMP4300

Parallel Systems

Study Period: 15 minutes

Time Allowed: 2 hours

Permitted Materials: Calculator

Exam questions total 100 marks.

Answer ALL parts of ALL questions.

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 [2 marks]

Explain the differences between synchronous and blocking communications.

Question 2 [2 marks]

Page based distributed shared memory systems are prone to poor performance due to false aliasing. What is false aliasing and why does it lead to poor performance?

Question 3 [2 marks]

In the context of parallel computer architectures what is COMA?

Question 4 [2 marks]

How does task granularity affect Amdahl's law?

Question 5 [3 marks]

What is the difference between minimal, deterministic and adaptive routing?

Question 6 [4 marks]

In pthreads what is the difference between a condition variable and a lock? (If it helps draw a cartoon to illustrate your answer!) Give one example of how each might be used.

Question 7 [4 marks]

In the context of parallel searching algorithms what is meant by i) acceleration anomaly, ii) deceleration anomaly, iii) detrimental anomaly. Give a brief indication of how these can arise.

Question 8 [5 marks]

Dijkstra's termination detection algorithm involves passing a token that is coloured either white or black around a ring of processors. Explain how this algorithm works and outline under what situations such a termination algorithm is mandatory.

Question 9 [6 marks]

A shared memory multiprocessor has the basic synchronization instructions **lock** and **unlock**. The operation **lock(L)** waits until a Boolean data item L is clear and then sets it and proceeds, while **unlock(L)** clears the lock. Using these basic locks a COMP4300 student has implemented a barrier synchronization routine according to the following pseudocode:

```
procedure barrier(n);
private last;
shared n, count, b,c;
begin
  lock(c);
  count := count + 1;
  last := (count == n);
  unlock(c);
  if last then unlock(b)
  else begin lock(b); unlock(b); end
  lock(c);
  count := count - 1;
  if count = 0 then lock(b);
  unlock(c);
end barrier;
```

The parameter n is the number of processes to cooperate in the barrier, and initial conditions are: c-unlocked, b-locked, and count=0. The barrier behaves **incorrectly** when five processes execute the following code:

```
phase1();
barrier(5);
phase2();
barrier(5);
phase3();
```

Describe what is wrong and detail how you would fix it.

Question 10 [7 marks]

The following are four MPI collective operations

```
MPI_Barrier
MPI_Bcast
MPI_Reduce
MPI_Allreduce
```

State briefly what you think each function does, and how you would expect its cost to scale with respect to the number of parallel processes (P) involved and, when applicable, its input parameters

Question 11 [8 marks]

The following C program is designed to compute:

$$\text{sum} = \sum_{i=1}^n i$$

```
#include <stdio.h>
#include <omp.h>
int main(void){
    int i,j,n,sum,iam,nthrd;
    n=100;
    sum=0;
    i=1;
    #pragma omp parallel default(none) shared(iam,i,j) \
        firstprivate(n) private(nthrd) reduction(+:sum)
    {
        iam = omp_get_thread_num();
        nthrd=omp_get_num_threads();
        do {
            sum += i+iam;
        #pragma omp atomic
            i+=nthrd;
        } while (i <= n);
    }
    printf("Sum from 1 to %i = %i \n",n,sum);
    return 0;
}
```

The code works correctly with 1 OpenMP thread but fails when the number of threads is set to 2 or greater.

- Explain what is wrong with the above code.
- Provide a working version of this code that removes any irrelevant statements.
- Comment on the likely scalability of your code.

Question 12 [25 marks]

Any symmetric positive definite matrix A can be written as $A = U^T U$ where U is strictly upper triangular (i.e. only elements U_{ij} where $j > i$ are non zero). One algorithm for evaluating U is known as Cholesky factorization. Cholesky factorization is very closely related to the Gaussian elimination algorithm discussed in lectures (but has the advantage of never requiring pivoting). Pseudocode for Cholesky factorization is shown below:

```
procedure cholesky(A);
begin
  for k := 0 to n-1 do
    begin
      A[k,k] :=  $\sqrt{A[k,k]}$ ;
      for j := k + 1 to n - 1 do
        A[k,j] := A[k,j]/A[k,k];
      for i := k + 1 to n - 1 do
        for j := i to n - 1 do
          A[i,j] := A[i,j] - A[k,i] × A[k,j];
        endfor;
      endfor;
    end cholesky;
```

- (a) What is the computational cost of the above algorithm when implemented on a sequential processor. Justify your answer.
- (b) Parallelise the above code for a distributed memory parallel computer using MPI. Give a detailed description of your algorithm including, where necessary, pseudocode.
- (c) Critically analyze your parallel algorithm. Include a detailed discussion of the computation and communication costs and how they scale with the number of parallel processes.

Question 13 [30 marks]

Three programming models that are applicable to distributed memory parallel computers are i) Message Passing, e.g. MPI, ii) Global Arrays (GA) or the closely related Distributed Data Interface (DDI), iii) software Distributed Shared Memory (DSM), e.g. Linda. Compare and contrast these three models. For each paradigm include in your answer:

- a brief description of what it is
- its advantages
- its disadvantages
- an example of an application for which it is well suited

Which of the above paradigms would you recommend to a parallel programming newbie? Justify your answer.

