

THE AUSTRALIAN NATIONAL UNIVERSITY
First Semester Examination – June 2004

COMP3320/COMP6464

High Performance Scientific Computing

Study Period: 15 minutes

Time Allowed: 3 hours

Permitted Materials: Calculator

Exam questions total 100 marks.

COMP3320 Answer Questions 1-7

COMP6464 Answer Questions 1-6 and 8

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

Question 1 [10 marks]

Fundamentals of Numerical Computing

- (a) Finite precision floating point addition and multiplication are commutative, but not associative. Explain what is meant by this. [2 marks]
- (b) Explain the difference between truncation error and rounding error. [2 marks]
- (c) Machine represented floating point number systems are finite and discrete. The number of normalized floating point numbers can be written as

$$2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$$

where β is the base, t is the precision, U is the maximum value of the exponent and L is the minimum value of the exponent. For $\beta = 2$, $t = 3$, $U = 1$ and $L = -1$ there are 25 normalized floating point numbers, 12 are positive, 12 are negative and 1 value represents zero. For this system:

- (i) What are the numerical values of the 12 positive floating point numbers?
- (ii) What is the underflow limit and what is the overflow limit?
- (iii) What are subnormal numbers and where would they go?

[6 marks]

Question 2 [10 marks]

Performance Modeling and Measurement

(a) For most applications it is best to have small number of very powerful processors rather than a larger number of less powerful processors. Why? Can you suggest a class of applications for which this may not be true? [3 marks]

(b) Consider a multi-user computer system with a (minimum) time slice interval of $t_s = 0.01s$. There is a moderate ‘process load’ on this system. A test program is to be used to time a computation; the program uses a high resolution wall timer (having overhead and resolution of much less than t_s).

Briefly explain how the following can improve the accuracy of the timings:

(i) Ensuring $0.5t_s \leq t_I < t_s$, where t_I is the timing interval of the computation

(ii) Taking several consecutive timings of the test program and then taking the smallest time as the “true” value.

[3 marks]

(c) When running your application program you have used hardware performance counters to measure instructions per cycle (IPC).

(i) If your measured value is high what does this tell you about the interaction of your program with the underlying hardware?

(ii) On an UltraSPARC platform what would you consider to be a high (or low) value anyway?

(iii) Why is it difficult to draw conclusions between instruction counts obtained on fundamentally different computer architectures (eg. UltraSPARC with Pentium IV)?

[4 marks]

Question 3 [15 marks]

Parallelism

(a) Dependences inhibit parallel operations.

- (i) Use the following four lines of C code to illustrate what is meant by a true dependence, an anti-dependence and an output dependence.

```
/*S1*/ Y = X / C;  
/*S2*/ X = X + C;  
/*S3*/ Z = Y + C;  
/*S4*/ Y = C - Y;
```

- (ii) Re-write the above code to eliminate all output and antidependences by renaming variables.

[5 marks]

(b) The following piece of code computes the new temperature (`tnew`) on a Cartesian 2-D grid by taking the average of the current temperatures (`told`) of the surrounding four grid points. It then prints out the maximum difference between the new and old temperatures (`mxdiff`). This process is repeated for 1000 iterations (`iter`).

```
for(iter=0; iter<1000; iter++){  
    mxdiff = 0.0;  
    for (j = 1; j < 19; j++){  
        for (i = 1; i < 19; i++){  
            tnew[j][i] = 0.25*(told[j+1][i] + told[j-1][i] +  
                               told[j][i+1] + told[j][i-1] );  
            tdiff = fabs(told[j][i] - tnew[j][i]);  
            mxdiff = (mxdiff < tdiff) ? tdiff : mxdiff;  
        }  
    }  
    printf(" iteration %d convergence %lf\n",iter,mxdiff);  
    for (j = 0; j < 20; j++){  
        for (i = 0; i < 20; i++){  
            told[j][i] = tnew[j][i];  
        }  
    }  
}
```

Show how you would parallelise the above code using OpenMP. You are not expected to reproduce the exact syntax of the OpenMP directives, but are required to make the intent of your directives clear. (You should note that while OpenMP includes a `reduction` clause, `maximum` is not a supported operation for this clause in the C implementation of OpenMP).

[10 marks]

Question 4 [15 marks]

Scientific Applications

- (a) The total interaction energy for a group of three or more atoms is approximated by the following:

$$E = E_2 + E_3$$

where

$$E_2 = \sum_{ij} \left(\frac{A}{r_{ij}^{12}} + \frac{B}{r_{ij}^6} \right)$$

and

$$E_3 = \sum_{ijk} C (\theta_{ijk} - \theta_{ijk}^0)^2$$

In the above expressions r_{ij} is the distance between atoms i and j (note the summation excludes $i = j$), θ_{ijk} is the current angle at atom j defined by the intersection of lines $i - j$ and $j - k$ (note the summation excludes $i = j$ or $i = k$ or $j = k$), θ_{ijk}^0 is some reference value for the same angle, and A, B and C are constants.

- (i) Provide pseudo code to illustrate how you would evaluate E_2 . (Assume there are `natom` atoms, and that the coordinates of the atoms are stored in an array of the form `coord[natom][3]`, where the first index corresponds to the atom and the second index corresponds to the `x`, `y` or `z` Cartesian coordinate).
- (ii) Discuss how you would go about evaluating E_3 . Note that the value of θ_{ijk} depends on the current coordinates of atoms i , j and k , but θ_{ijk}^0 does not depend on the current coordinates of these atoms. The $natom^3$ values of θ_{ijk}^0 are read once as input at the start of the simulation and stored in an array (`theta[natom][natom][natom]`).
- (iii) How would you expect the total CPU time required to evaluate E to scale as a function of the total number of atoms?
- (iv) How does the memory requirement of your code scale with the total number of atoms?

[15 marks]

Question 5 [20 marks]

High Performance Computer Architecture

- (a) The typical memory hierarchy of a modern processor includes registers, cache, main memory and disk storage. Contrast these different levels of storage. Your discussion should include typical size, access time, bandwidth and management of. [5 marks]
- (b) What is a cache line conflict? Illustrate your answer by outlining a small code fragment that will exhibit a cache line conflict on a computer with a 16KB 2-way set associate cache that has a 32byte cache line size. [5 marks]
- (c) Over time there has been a tendency for all microprocessors to increase the number of stages in their pipeline. Why? What is the potential disadvantage of long pipelines and how have microprocessors designers sought to minimize this disadvantage? [4 marks]
- (d) On a shared memory parallel computer the words $x[0]$ and $x[1]$ map to the same cache line. At time T_0 both processors P1 and P2 issue read requests bringing this cache line into their respective caches. In subsequent time steps the following instructions are issued:

Time	P1	P2
0	read $x[0]$	read $x[1]$
1	write $x[0]$	
2		read $x[1]$
3	write $x[0]$	
4		write $x[1]$
5	read $x[1]$	

- (i) On each processor what happens to the cache line containing $x[0]$ and $x[1]$ at the various time steps.
- (ii) The above example shows one (or more) misses due to false sharing. What is a false sharing miss, and when do such misses occur in the above access sequence?

[6 marks]

Question 6 [20 marks]

Performance Programming

- (a) You are writing a new 3D game that you hope will earn you fame and fortune. You are currently working on a function to blank the screen buffer before drawing the next frame. The screen you are working with has a 640×480 array of pixels. The machine you are working on has a 64KB direct mapped data cache with 4-byte lines. The C structures you are using are:

```
1 struct pixel {
2     char r;
3     char g;
4     char b;
5     char a;
6 };
7
8 struct pixel buffer[480][640];
9 int i, j;
10 int *iptr;
```

Assume the following:

- `sizeof(char) == 1` and `sizeof(int) == 4`
- `buffer` begins at memory address 0
- The cache is initially empty
- The only memory accesses are to the entries of the array `buffer`, ie. variables `i`, `j` and `iptr` are stored in registers

- (i) What percentage of writes in the following code will miss in the cache? Clearly show how you derive your answer.

```
1 for (j = 0; j < 640; j++){
2     for (i = 0; i < 480; i++){
3         buffer[i][j].r = 0;
4         buffer[i][j].g = 0;
5         buffer[i][j].b = 0;
6         buffer[i][j].a = 0;
7     }
8 }
```

- (ii) What percentage of writes in the following code will miss in the cache? Clearly show how you derive your answer.

```
1 iptr = (int *) buffer;
2 for (; iptr < ((int *)buffer + 640*480); iptr++)*iptr = 0;
```

[6 marks]

- (b) Your friend has written the following routine to perform matrix transpositions of square matrices

```
1 void transpose(double *dst, double *src, int dim)
2 {
3     int i, j;
4     for (i = 0; i < dim; i++)
5         for (j = 0; j < dim; j++)
6             dst[j*dim + i] = src[i*dim + j];
7
8 }
```

For a small fee you agree to optimise this routine for performance on a modern RISC processor. You should assume that the computer has a 64KB direct mapped level 1 data cache with a 32 byte cache line size and `sizeof(double) == 8`. (You can ignore level 2 cache effects). The virtual memory system has an 8KB page size and there is a 64 entry TLB.

- (i) How much would you charge your friend to perform this work? To justify your fee give a detailed discussion of the issues that are involved and what you would do. As required provide snippets of code to illustrate the modifications you propose to undertake. (10 marks)
- (ii) Your colleague has heard of cache-oblivious algorithms and wonders if such methods are applicable in this case. What are cache oblivious algorithms and how might they be applied to the above code? (4 marks)

[14 marks]

COMP3320 STUDENTS ONLY

Question 7 [10 marks]

(a) Java for computational science:

- (i) From a computational science perspective give two potential benefits AND two current limitations of Java.
- (ii) When computational scientists made early attempts to use Java they found it to be very slow. Why was this? What recent advances have meant that the performance difference between native C or Fortran code and the equivalent Java code is no longer as great?

[4 marks]

(b) Distributed memory parallel computing: In message passing the time required to perform a communication is often modeled by the following equation

$$t_{comm}(n) = \alpha + \beta n$$

- (i) Explain what each of the terms in this equation mean.
- (ii) What are typical values for α and β on current generation parallel computers?
- (iii) Beowulf clusters are a popular message passing environment. What factors have contributed to the success of Beowulf clusters?

[6 marks]

COMP6464 STUDENTS ONLY

Question 8 [10 marks]

- (a) Overheads due to synchronisation and loop scheduling are an important factor in determining the performance of shared memory parallel programs. Outline how you would go about measuring the overhead of the following directive:

```
#pragma omp for schedule(type[,chunk])
```

as a function of different scheduling types and chunk sizes. [5 marks]

(b) Image processing

- (i) Give two different examples of how discrete Fourier transforms are used in image processing.
- (ii) How does the cost of performing a 2-D Fourier transform vary with the size of the data (eg. for an image of size $N \times M$ pixels)
- (iii) Why is it advantageous to have images for which the dimensions are powers of 2?

[5 marks]