Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Concurrent Programming for Numerical Applications

Date: Friday 28th January 2011
Time: 14:00 - 16:00

Please answer any TWO Questions from the Four questions provided

This is an OPEN book examination

The use of electronic calculators is permitted provided they are not programmable and do not store text.

[PTO]
Question 1.
Consider the following fragments of code that perform some simple numerical linear algebra computations (vector axpy operation (linked vector addition and vector scaling), and the multiplication of two lower triangular matrices:

\[ a = \alpha x + y \]
\[ B = L \ast M, \]

where \( \alpha \) is a scalar, \( a, x, y \) are vectors of length \( n \) (\( n \) can be assumed to be large) and \( B, L, M \) are \( n \times n \), lower triangular, matrices. (A lower triangular matrix is one in which all the elements above the diagonal are zero, \( L_{ij} = 0, i < j \).)

i) The following FORTRAN code initialises the vectors \( b, c \) and implements the vector axpy operation

```fortran
C C vector initialisation C
C DO i=1,n
 x(i) = rand()
 y(i) = rand()
END DO C
C C vector axpy C
C DO i=1,n
 a(i) = alpha*x(i) + y(i)
END DO C
```

Identify, without reference to any particular parallel architecture, the nature of any parallel work in the loops above. [3 marks]

ii) One implementation (implementation A) parallelises the second loop (the axpy operation) by including the OMP pragma

```fortran
!$omp parallel do schedule(static)
```

immediately before the second DO statement, and a second implementation (implementation B) parallelises both loops by including the same pragma before each of the DO statements.

The performance (in seconds) of these two implementations on 1 – 8 cores of chronos (a 16-core AMD Opteron-based server) (these timings exclude the initialisation loop in each case) is as follows:

(Question 1 continues on the following page)
(Question 1 continues from the previous page)

<table>
<thead>
<tr>
<th>No of Cores</th>
<th>Implementation A</th>
<th>Implementation B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4594</td>
<td>0.4597</td>
</tr>
<tr>
<td>2</td>
<td>0.5907</td>
<td>0.2308</td>
</tr>
<tr>
<td>3</td>
<td>0.4054</td>
<td>0.1554</td>
</tr>
<tr>
<td>4</td>
<td>0.3638</td>
<td>0.1182</td>
</tr>
<tr>
<td>6</td>
<td>0.2724</td>
<td>0.08048</td>
</tr>
<tr>
<td>8</td>
<td>0.2451</td>
<td>0.06411</td>
</tr>
</tbody>
</table>

Explain these results in terms of parallel overheads. [7 marks]

iii) The following FORTRAN code fragment calculates the lower triangular matrix product:

```fortran
C
C Matrix multiplication
C
DO j = 1, n
   DO i=j, n
      B(i,j) = 0.0
      DO k = j, i
         B(i,j) = B(i,j) + L(i,k) * M(k,j)
      END DO
   END DO
END DO
```

Identify the nature (and limitations) of any parallel work in the above calculation as it is written. Suggest a parallel implementation of the above calculation – clearly identify all the potential overheads and include consideration of the initialisation of the arrays L, M. [10 marks]
Question 2.

a) The following table shows the output of a profiling tool for a simple program:

<table>
<thead>
<tr>
<th>Routine</th>
<th>Exclusive time (sec.)</th>
<th>Inclusive time (sec.)</th>
<th>Number of calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>bessel</td>
<td>110.0</td>
<td>110.0</td>
<td>50000000</td>
</tr>
<tr>
<td>hankel</td>
<td>30.0</td>
<td>30.0</td>
<td>12000000</td>
</tr>
<tr>
<td>Solve</td>
<td>10.0</td>
<td>130.0</td>
<td>1</td>
</tr>
<tr>
<td>Init</td>
<td>5.0</td>
<td>25.0</td>
<td>1</td>
</tr>
<tr>
<td>Main</td>
<td>5.0</td>
<td>160.0</td>
<td>1</td>
</tr>
</tbody>
</table>

Exclusive time is the time spent in the given routine alone, inclusive time is the time spent in the given routine and all its descendants.

1) Identify the routines which do not call any other routines, and sketch a feasible call tree. [2 marks]

2) The routines bessel and hankel are part of a library which implements special mathematical functions, and their execution times are independent of the values of their arguments. Calculate the expected effect on the total execution time of the program resulting from the following optimisations (in each case, briefly comment on the likely cost in terms of programmer effort):

   i) Replacing the given library with an optimised version of the library in which the cost of calling bessel is reduced by a factor of 1.1, and the cost of calling hankel is reduced by a factor of 1.5. [4 marks]

   ii) Forcing the compiler to inline bessel and hankel, assuming that the overhead of a routine call is $10^{-7}$ seconds (0.1 microseconds). [4 marks]

   iii) Implementing a different overall algorithm which requires the same number of calls to hankel but 10 times fewer calls to bessel. [4 marks]
b) In OpenMP, the PARALLEL DO construct has a number of scheduling options including:

- Static (with specified chunk size),
- Dynamic (with specified chunk size),
- Guided.

In the following code fragment, suggest a suitable choice of scheduling option, and chunksize, for the PARALLEL DO directive, giving reasons for your choice. Assume that the target architecture is chronos (a 16-core AMD Opteron-based server), and do not consider any restructuring of the code.

i) 

```fortran
!$omp PARALLEL DO
  DO J=1, 100000
    IF (ISSQUARE(J)) THEN
      CALL SUB(A(J))
    END IF
  END DO
!$omp END PARALLEL DO
```

**Here ISSQUARE is a logical function returning .TRUE. if and only if its argument is a perfect square, and SUB is a subroutine whose execution time is independent of its argument, and is significantly greater that the execution time of ISSQUARE.**

[6 marks]
Question 3.

a) Consider the computation represented by the following fragment of FORTRAN code:

```fortran
integer i, j, n
real error, eps
real a(n,n), f(n,n)

C iteration loop
while (abs(error) .gt. tol) do
   do j = 2, n-1
      do i = 2, n-1
         a(i,j) = 0.25*(a(i,j-1)+a(i-1,j)+a(i+1,j)+a(i,j+1))
         & + a(i,j) + f(i,j)
      enddo
   enddo
   error = . . .
end while
```

For a $p^2$–processor message-passing implementation of this algorithm, describe

i) A block-row partitioning of the data,

ii) A block-column partitioning of the data,

iii) A block partitioning of the data, with rectangular blocks of dimensions $q \times r$, where $q \times r = n^2 / p^2$, and $q, r \leq n/2$.

Derive the volume of communication per processor implied by each data distribution, and show that, for $p \geq 2$, the volume of communication is minimised when we choose to partition $a$ and $f$ into square blocks of dimensions $n/p \times n/p$. [8 marks]

b) Identify whether the following fragments of FORTRAN code can be executed in parallel (possibly after suitable transformations). Justify your answers, using diagrams where appropriate:

1) Code fragment 1

```fortran
DO i = 1, n
   a(2*i) = i
   b(i) = a(6*i-1)
END DO
```

[3 marks]

2) Code fragment 2

```fortran
DO i = 1, n
   a(i) = b(i)
   c(i) = a(i+1)
   d(i) = c(i) + 1
END DO
```

[3 marks]

(Question 3 continues on the following page)
c) The following fragment of FORTRAN code is given:

```fortran
k = 1000
do i = 2, 999
   do j = 1, 1000
      k = k+1
      if (i .ge. j) then
         c(i,j) = 0.0
         do m = j, j+1
            c(i,j) = c(i,j) + a(m)*b(k+1000*(m-j))
         enddo
      endif
   enddo
enddo
```

i) Transform the above code into a semantically equivalent form where the outer loop is parallelisable. [3 marks]

ii) How would you map the transformed parallel code onto the processors so as to reduce load imbalance? [3 marks]
Question 4.
(a) Consider Strassen’s Algorithm for the calculation of the matrix product
\[ C = A \times B \]
where A, B, and C are \( N \times N \) matrices. Assuming that \( N \) is even, divide each of A, B and C into the \( 2 \times 2 \) block matrices
\[
A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix},
\]
where the blocks \( A_{ij}, B_{ij} \) and \( C_{ij} \) are \( \frac{N}{2} \times \frac{N}{2} \) matrices. Given the intermediate matrices
\[
P_1 = (A_{11} + A_{22})(B_{11} + B_{22}),
\]
\[
P_2 = (A_{12} + A_{22})B_{11},
\]
\[
P_3 = A_{11}(B_{12} - B_{22}),
\]
\[
P_4 = A_{22}(B_{21} - B_{11}),
\]
\[
P_5 = (A_{11} + A_{12})B_{22},
\]
\[
P_6 = (A_{21} - A_{11})(B_{11} + B_{12}),
\]
\[
P_7 = (A_{12} - A_{22})(B_{21} + B_{22}),
\]
show that the blocks of \( C \) are given by
\[
C_{11} = P_1 + P_4 - P_5 + P_7,
\]
\[
C_{12} = P_3 + P_5,
\]
\[
C_{21} = P_2 + P_4,
\]
\[
C_{22} = P_1 + P_3 - P_2 + P_6.
\]

[4 marks]

(b) Considering floating point operations only, show that conventional matrix multiplication of two \( N \times N \) matrices requires \( 2N^3 - N^2 \) flops (floating point operations), whilst Strassen’s algorithm, with one level of recursion, requires approximately \( \frac{7}{4} N^3 + \frac{11}{4} N^2 \) flops. [6 marks]

(c) Given that \( N = 2^n \), where \( n \) is a positive integer, describe a recursive version of Strassen’s algorithm. Show that Strassen’s algorithm is more efficient than the conventional algorithm for matrices of dimensions \( 16 \times 16 \), but not for matrices of dimensions \( 8 \times 8 \). Hence, show that it is optimal to recur Strassen’s algorithm until the submatrices have dimensions \( 8 \times 8 \). [6 marks]

(d) Describe briefly how you would implement Strassen’s algorithm on a parallel machine such as chronos, and note any major obstacles to good parallel performance. [4 marks]