Parallel Programming, Algorithms and Architecture
Midterm Exam
November 2, 2000
Take Home, Open Notes and Reference Texts
Due Tuesday November 7, 2000
beginning of class

Name:

Total:
Problem 1 (20 points)

(a) 10 pts. Suppose you have a 10 stage pipelined adder, how many cycles would it take to add of 1000 pairs of numbers? What is the computational rate (operations per cycle) achieved? How close is it to the limiting performance of the pipeline?

(b) 10 pts. Suppose the workload of your code spends 90% of the time in vectorizable operations. A vendor proposes that you upgrade to a machine that includes a vector unit that is 100 times faster than its scalar unit (which performs the same as your current scalar unit). Relative to the price of your current system, what is the most you would agree to pay for the new machine? Justify your answer.
Problem 2 (20 points)

Consider the following sequential code segments and determine the data dependences that would hinder parallel execution. The parallel code is to be consistent with the sequential code, i.e., it is to produce the same results in all of the variables. Remember we discussed true dependences where a statement producing data must be executed before a statement consuming that data, antidependences where a statement reading data must execute before another statement updates the data. Identify all such dependences and any other types that would prevent the parallelization of the code and its producing sequentially consistent results.

(i)  
S1: \[ A = B + D \]  
S2: \[ C = A \times 3 \]  
S3: \[ A = A + C \]  

(ii)  
do \text{i1} = 1,10  
do \text{i2}=1,10  
do \text{i3} = 1,10  
S1: \[ a(\text{i1}, \text{i2}, \text{i3}) = b(\text{i1}-1, \text{i2}, \text{i3}) \times c(\text{i1}, \text{i2}) + d \times e \]  
S2: \[ b(\text{i1}, \text{i2}, \text{i3}) = a(\text{i1}, \text{i2}-1, \text{i3}) \times f(\text{i1}, \text{i2}) \]  
\end do  
\end do  
\end do  

(iii)  
do \text{i}=1,n  
S1: \[ a(\text{i}) = b(\text{i}) \times c(\text{i}) \]  
do \text{j}=1,n  
S2: \[ d(\text{i}, \text{j}) = a(\text{i}-1) + e(\text{i}, \text{j}-1) \]  
S3: \[ e(\text{i}, \text{j}) = d(\text{i}, \text{j}-1) + f \]  
\end do  
\end do  
\end do  
S4: \[ g(\text{i}, \text{k}) = h(\text{i}-5)+1 \]  
\end do  
S5: \[ h(\text{i}) = \sqrt{a(\text{i}-2)} \]  
\end do
Problem 3 (20 points)

Consider the matrix vector product \( y \leftarrow Ax \) where \( A \) is a square matrix of order \( n \). Recall there are two main approaches. The first uses dotproducts, i.e.,

\[
\eta_i = \sum_{j=1}^{n} a_{i,j} \xi_j
\]

where \( y \) consists of \( \eta_i, i = 1, \ldots, n \), \( x \) consists of \( \xi_i, i = 1, \ldots, n \) and \( A \) consists of \( a_{i,j} i = 1, \ldots, n \) and \( j = 1, \ldots, n \). The second uses a series of vector triads

\[
y = \xi_1 a_1 + \xi_2 a_2 + \cdots + \xi_n a_n
\]

where \( a_i \) is the \( i \)th column of \( A \).

Now suppose \( A \) is also a sparse matrix, i.e., most of the elements are zero, and it is stored using the compressed row format data structure described below.

Let \( nz \) be the total number of nonzero elements in \( A \). Let \( A(1:~nz) \) be an array in which the nonzeros of \( A \) are grouped together so that elements of the same row are contiguous. No assumption can be made about the order of the elements within each row, i.e., they cannot be assumed to be sorted by column index. No assumption can be made about the order of the rows themselves, i.e., the group of contiguous elements for row \( j \) might appear before or after the group of contiguous elements for row \( i \) regardless of the relationship of \( i \) and \( j \). Let \( \text{CPTRS}(1:nz) \) be an integer array where \( \text{CPTRS}(i) \) contains the column index for the nonzero element stored in \( A(i) \). Finally we assume that we have two arrays \( \text{RS}(1:n) \) and \( \text{RE}(1:n) \) such that \( A(\text{RS}(i)) \) is the first element in row \( i \) and \( A(\text{RE}(i)) \) is the last element in row \( i \), i.e., \( \text{RS}(i) \) points to the beginning of row \( i \) in \( A(1:nz) \) and \( \text{RE}(i) \) points to the end of row \( i \) in \( A(1:nz) \).

**Example:** For the matrix

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 2 \\
5 & 2 & 3 & 0 & 0 \\
0 & 0 & 1 & 0 & 3 \\
7 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 8
\end{bmatrix}
\]

one possible layout in the data structure is
• \( A(1:10) = (1, 3, 5, 2, 3, 2, 1, 7, 3, 8) \)
• \( \text{CPRTR}(1:10) = (3, 5, 1, 2, 3, 5, 1, 1, 4, 5) \)
• \( \text{RS}(1:5) = (6, 3, 1, 8, 10) \)
• \( \text{RE}(1:5) = (7, 5, 2, 9, 10) \).

(It is one possible version due to the arbitrary ordering of elements within a row and the rows within \( A \).)

1. Describe how you would implement \( y \leftarrow Ax \) on a parallel shared memory architecture. This data structure should allow a simple synchronization strategy, e.g., doalls only. Describe your implementation using the Cedar Fortran constructs of the notes.

2. On the same architecture, describe how you would implement \( y \leftarrow A^T x \) where \( A^T \) denotes the transpose of \( A \). If \( B = A^T \) then the \( i, j \) element of \( B \) is the same as the \( j, i \) element of \( A \). Assume that you must use the data structures given above. Justify your choice of synchronization strategies. Hint: consider carefully the use of one or more critical sections.
Problem 4

Consider the matrix vector product $y \leftarrow Ax$ where $A$ is a square matrix of order $n$. Suppose you are to implement this on a distributed memory processor. The partitioning of the data is given below. You do not have to worry about how the data got that way. Your solution should start from the assumed data positioning.

A block column partitioning of $A$ is assumed. Each processor has $k = n/p$ contiguous columns in its memory. The vector $x$ is not distributed, all $n$ of its elements are stored in processor 0. The solution $y$ is to end up distributed with each element stored in the processor that contains the column of $A$ with the same index, i.e., each processor has $n/p$ contiguous elements of $y$. So processor 0 has columns 0 through $n/p$, and the elements of $y$ with these same indices, and all of $x$. Processors 1 to $p - 1$ have similar portions of $A$ and $y$ and none of $x$.

Describe how you would implement this primitive using MPI subroutines for message passing. You do not have to write the complete code (unless you want to) but you should be very specific about what subroutines you use, when you use them, and what they accomplish.

Use as many global communication primitives as possible as these tend to reveal patterns in the communication that can be exploited for efficiency’s sake. Recall that we discussed: broadcast, scatter, gather, allgather, and alltoall. A subset of these will be very useful for this problem.
Problem 5

Suppose you are responsible for designing the protocol to control access to a single I/O resource on a distributed memory machine. We assume that any process can perform I/O but only one can perform it at any particular time. If the machine used a shared memory paradigm this access could be controlled by implementing a critical section using, say, locks. However, the distributed memory paradigm has no such support. The I/O resource itself is not a process involved in the MPI message passing activity, i.e., you should only worry about the MPI processes involved in the protocol. Communication between the I/O resource and the process that is allowed to use it will be done via special I/O routines that are not relevant here.

- Describe a strategy that would allow the processes to control their access to the I/O resource while respecting the above critical section restriction.

- How would you expect the performance to change as the number of processes involved increases?