CSC/ECE 506: Architecture of Parallel Computers
Problem Set 2
Due Wednesday, June 30, 2004

Problems 1, 2, and 3 will be graded. There are 60 points on these problems. Note: You must do all the problems, even the non-graded ones. If you do not do some of them, half as many points as they are worth will be subtracted from your score on the graded problems.

Problem 1. (25 points) Given the need to multiply two matrices together to yield a third:

\[ P = M \times N \]

Where,

- M consists of 100 rows and 50 columns
- N consists of 50 rows and 200 columns

(2 points each)

a) Using Tichy’s \( n^2 \log n \) algorithm, how many processors would be needed in order to perform this calculation, compared to how many processors needed if his \( n \log n \) algorithm were implemented instead?

b) How do the communication steps for each of the matrix rows of M & N differ between the two algorithms in computing the result matrix P?

c) In this example, how many communication steps are needed using the \( n^2 \log n \) algorithm versus using \( n \log n \)?

d) Since both algorithms implement a sum reduction or fanin reduction to sum their inner products, the difference between the two is clearly the mechanism or algorithm used to calculate these inner products. Describe these differences and any pros or cons in terms of communication vs. processors it might have.

e) If we had another matrix instead of M, say matrix Z that was 128 rows by 50 columns. How many iterations would be necessary in order to complete this particular column using the fanout tree method? That is if we seeded the first column of N into the first row of Z, how many iterations are necessary before the \( l \times m \) multiplications can be computed in parallel for their inner products.

f) Which processors take part in computing \( P[50, 100] \) using \( n^2 \log n \) algorithm?

g) Which processors take part in computing \( P[50, 100] \) using \( n \log n \) algorithm?

h) We’ve seen that although we’re able to achieve more parallelism using the \( n \log n \) algorithm it is still not that efficient. Show why this is so by determining the number of processors implemented during the first broadcast or column N to matrix M.

i) Using Tichy’s other algorithm, \( \log n \), how many processors would be needed to compute M x N? Is this realistic and why or why not?

j) Where does the \( \log n \) time come from?

Problem 2. (15 points) Consider a four-processor bus-based multiprocessor using the Illinois MESI protocol. Each processor executes a test&set* lock to gain access to a null critical section. Assume the test&lock instruction always goes on the bus and it takes the same time as a normal read transaction. The initial condition is such that processor 1 has the lock and the processors 2, 3 and 4 are spinning on their caches waiting for the lock to be released. Every processor gets the lock once and then exits the program. Considering only the bus transactions related to the lock-unlock operations:

a. What is the least number of transactions executed to get from the initial to the final state?

b. What is the worst case number of transactions?

c. Repeat parts (a) and (b) assuming the Dragon protocol.

* See link for a description: [http://www.cs.virginia.edu/~son/cs414.f03/q3.txt](http://www.cs.virginia.edu/~son/cs414.f03/q3.txt)
Problem 3. (20 points) This question concerns the parallel algorithms for multiplying two arrays on a machine with \( N \) processors. Suppose we multiply two \( 8 \times 8 \) matrices \( A \) and \( B \) on a machine with 64 processors. Let the rows and columns each be numbered 0 through 7.

a. (3 points) At which processor is row 6 of \( A \) in the initial configuration? Which processors is it broadcast to?

b. (2 points) At which processor will row 4 of \( A \) be multiplied by column 2 of \( B \)?

c. (2 points) At which processors will the elements of row \( i \) of matrix \( C \) be calculated?

d. (6 points) Using an efficient algorithm, how long will it take to collect the elements of a row of \( C \) in the same processor that originally held the corresponding row of \( A \)? Assume that: routing operations take one cycle each, routing operations can be performed by all of the processors at the same time, but no processor can send more than one element at a time, nor can it receive more than one element at a time.

e. (5 points) If routing operations, multiplications, and additions all require one cycle each, how many cycles are required to multiply the two arrays? In addition to the assumptions of part (d), assume that the matrices start out in their "initial configurations" as specified in the notes, and the result matrix \( C \) must end up at the same processors that originally held \( A \).

f. (2 points) Based on your answer to (e), write an expression for the number of cycles it would take to multiply two \( 16 \times 16 \) matrices on a 256-processor array. (You need not perform the arithmetic to evaluate your expression).

Problem 4. (20 points) Consider the following memory hierarchy:

\[
\begin{array}{c|c|c}
\text{Processor} & \text{Cache} & \text{Memory} \\
\end{array}
\]

(a) For the given memory hierarchy, what are the hit and miss rates in the cache and the memory if the processor executes a total of 1,000,000 memory references, and 945,000 of which hit in the cache and 45,000 of which hit in the main memory?

(b) If the memory size is 4000MB, and the cache is of size 64KB, and the cache has lines of size 128B each, and is 4-way set-associative, then

(i) The cache contains how many numbers of lines, sets and tags?

(ii) The address used in this system contains total how many bits, and each tag in the cache contains how many bits?

(iii) If the cache is write-through, how many bits are required for each tag if an LRU replacement policy is used? What if the cache is write-back?

(c) The cache has a hit rate of 95% and a cache hit latency of 5 ns. The main memory takes 100 ns to return the first word (32 bits) of the line, and 10 ns to return each subsequent word.

(i) What is the cache miss time for this cache? (Assume that the cache waits until the line has been fetched into the cache and then re-executes the memory operation, resulting in a cache hit. Neglect the time required to write the line into the cache once it has been fetched from the memory. Also, assume that the cache takes the same amount of time to detect a miss has occurred as to handle a cache hit.)

(ii) If doubling the cache line length reduces the miss ratio to 3 percent, does it reduces the average memory access time? (Remember: Average access time for each level in memory hierarchy contains the time taken because of hit + time taken because of miss)
(d) Similar to the 1st bit of the question, the processor executes 1,000,000 memory references. The cache has a miss rate of 7%, of which 1/4 are compulsory misses or cold misses, 1/4 are capacity misses, and 1/2 are conflict misses.

(i) If the only change you are allowed to make to the cache is to increase the associativity, what is the maximum number of misses that you can hope to eliminate?

(ii) If you are allowed to both increase the cache size and increase the associativity, what are the maximum number of misses that you can hope to eliminate?

Problem 5. (20 points) Write the code for an average_columns function for the data-parallel model of lecture 8.

The function should divide up the data so that each processor operates on columns of data. The average of each column of data should be written into the corresponding element of the 1-dimensional array column_averages_array. The calling function is already written for you.

```c
int nprocs;
int row_count;
int col_count;

double **A;
double *column_averages_array;

procedure main()
begin
  read(nprocs);
  read(row_count);
  read(col_count);
  A <= G_MALLOC( row_count * col_count );
  column_averages_array <= G_MALLOC( col_count );
  read(A);
  average_columns(A, column_averages_array, row_count, col_count);
  print_1d_array( "Column Averages:
", column_averages_array );
end

procedure average_columns(A, col_avgs, row_count, col_count)
begin
  double **A, double *col_avgs, int row_count, int col_count;
begin
  *** your code here. ***
end
```