Boston University
College of Liberal Arts

Computer Science Department
111 Cummington Street, MCS 138
Boston, Massachusetts 02215
Tel: (617) 353-8919 / Fax: (617) 353-6457

CLA CS-551

Parallel Computing
Models, Languages, and Architectures

Sample Homework Problems

Problem 1 (UNITY)
Define the rank of an element *A[j]* in an integer array *A[0..N]* as the number of items in *A[0..N]* smaller than *A[j]* plus the number of items in *A[0..N]* equal to *A[j]* with subscripts smaller than j.

- ◇ Without using quantified expressions, write a Unity program to set the element *R[j]* of an integer array *R[0..N]* equal to the rank of *A[j]* in the array A. Solve the problem assuming:
  a. A sequential architecture.
  b. A synchronous shared memory SIMD architecture with N processors.
  c. A synchronous shared memory SIMD architecture with N/k processors, where $k \in \mathcal{Z}$, $k > 1$ is the virtual processing ratio.
  d. A synchronous shared memory architecture with N^2 processors.
- ◇ In each one of the above cases, estimate the time complexity (assume that each state change is equivalent to a unit of time).

Problem 2 (Shared Memory MIMD / BBN-Buterfly)
One technique to print all the prime numbers smaller than a given bound, M, can be sketched as follows:

1 Generate a *big enough* list of integers.
2 Eliminate any list element that is divisible by another one with a smaller index.
3 When no more elimination is possible, print out uneliminated integers $< M$.

Write (no real implementation required) an *efficient* Butterfly program that prints all prime numbers less than some given upper bound, M, using the aforementioned technique. Comment on the efficiency of your program.

Hint: Use a generator to spawn processes that would do the elimination. Refer to the examples in the Butterfly Plus (Uniform System) manual for sample code.

Problem 3 (CM-2/C-Paris)
We all know that to compute the product of two matrices, we do:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} * B_{kj}$$

To do this on the CM-2, storing one element per processor, is tricky because we have to get the appropriate elements to the appropriate processors. Doing so using general communication might prove to be overly expensive.

Another way of doing matrix multiplication is using Pipelining. To explain this technique, consider the following: ($\oplus$ and $\otimes$ denote element-wise addition and multiplication)

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{22} & A_{23} & A_{21} \\ A_{33} & A_{31} & A_{32} \end{pmatrix} \otimes \begin{pmatrix} B_{11} & B_{22} & B_{33} \\ B_{21} & B_{32} & B_{13} \\ B_{31} & B_{12} & B_{23} \end{pmatrix}$$

$$\oplus$$

$$\begin{pmatrix} A_{12} & A_{13} & A_{11} \\ A_{23} & A_{21} & A_{22} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \otimes \begin{pmatrix} B_{21} & B_{32} & B_{13} \\ B_{31} & B_{12} & B_{23} \\ B_{11} & B_{22} & B_{33} \end{pmatrix}$$

$$\oplus$$

$$\begin{pmatrix} A_{13} & A_{11} & A_{12} \\ A_{21} & A_{22} & A_{23} \\ A_{32} & A_{33} & A_{31} \end{pmatrix} \otimes \begin{pmatrix} B_{31} & B_{12} & B_{23} \\ B_{11} & B_{22} & B_{33} \\ B_{21} & B_{32} & B_{13} \end{pmatrix}$$

In the above computation, the elements of the 3 by 3 arrays A and B were initially arranged so that successive circular shifts (shifting up for A and left for B) resulted in the elements of A and B be aligned at each processing element (PE) so that the cumulative sum for the matrix multiplication can be computed in C, which remains static with respect to the PEs.

a. Write a C-Paris program to do matrix multiplication storing A_{ij}, B_{ij}, and C_{ij} in the same processor. Your program should use *sends* (or *gets*) to move the data around.

b. Write a C-Paris program to do matrix multiplication using the pipelined approach described before.

c. Show that your programs work (that might be tricky with 4096-element arrays!)

d. Provide some timing comparison. Comment.

Hint: Assume that A and B are square matrices.. If you need a more challenging problem, try solving part (b) *efficiently* for compatible matrices.

Problem 4 (CM-2 / C*)
If two numbers x and y are picked at random from a uniform distribution $[0-1]$, then the probability that on the X-Y plane, the point (x, y) falls inside the unit circle is given by: $\frac{\pi}{4}$ This experiment is an example of a Monte-Carlo simulation and can be used to estimate the value of the irrational constant π to any degree of accuracy. Write a C* program to find an accurate representation for π using Monte-Carlo simulation.

Problem 5 (CM-2 / C*)
The Connection Machine is to be used to perform a simulation to estimate the probability that an accelerated particle aimed at a thin sheet of metal will be reflected, deflected, or absorbed. The experiment works as follows:

- ◇ Initially, the particle is aimed at an angle perpendicular to the sheet.
- ◇ Each step of the simulation concerns itself with the journey of the particle as it bounces from one atom to another.
- ◇ At any given iteration, the distance traveled, which follows an exponential distribution, can be calculated using the equation:

  $$d = \frac{1}{\lambda} \cdot \ln\left(\frac{1}{1 - \texttt{rnd()}}\right)$$

  where, `rnd()` is a uniformly-distributed random value and λ is a constant that depends on the metal under test ($\frac{1}{\lambda}$ is the average distance traveled in one iteration).
- ◇ As a result of hitting an atom at the end of a given iteration, the particle's trajectory direction is changed by an angle θ. The rotation angle in radians is a random variable that can be generated using the equation:

  $$\theta = \pi \left[\left(\frac{1}{K} \sum_{i=1}^{K} r_i \right) - \frac{1}{2} \right]$$

  where, r_i, $1 \leq i \leq K$ are uniformly distributed random variables and K is constant that depends on the metal under test.
- ◇ After each iteration, if the perpendicular distance traveled by the particle through the metal exceeds the thickness D of the sheet then the particle is considered to have been deflected. If the perpendicular distance traveled is negative then the particle is considered to have been reflected. The particle is considerd to have been absorbed if the total distance traveled (in any direction) is greater than $L \times D$.

Write a C* program to simulate the described experiment. A total number of at least 32,000 particles must be simulated. Your solution should assign a particle to each PE and simulate an array of particles in a SIMD fashion. Your program should maximize the utilization of the connection machine.

Problem 6 (Maspar / MPL)
Given a Maspar-like computer with functions similar to those defined in class for the MPL language, write a pseudo-MPL procedure to compute efficiently the sum of a plural variable x. You can use either nearest neighbor communication or the circuit switching global router to communicate. Assume that the machine has $N = 2^k$ processing elements that can be arranged in a linear or rectangular shape, with direct connections to immediate neighbors, and that the cost of a one-hop communication with a neighbor is c. Also, assume that the circuit switching global router can handle $\frac{N}{2^r}$ random connections at any point in time (this is an over-simplification). Assume that the cost of establishing such connections is $s \times c$ ($s > 1$) and that the transfer time can be neglected. What is the cost of invoking your procedure?

Clarify any assumptions you make in solving your problems.