Engi 8893 / 9869 Assignment 1

T.S. Norvell

Due 2006 Feb 10, 10:30 AM

Solutions should be typed. Diagrams may be hand-drawn neatly.

As much as appropriate, use the design notation from Andrews's text.

Q0 [10]. The book shows a distributed solution to the matrix multiplication algorithm that uses N processes. Suppose that you have N^2 processes. Each process has a very small local memory[1]. How could you use distributed peers to solve the matrix multiplication problem? Design the algorithm. In what topology would you suggest connecting the processors and how would you distribute processes to processors?

(Optional extra. Suppose you have N processors and can arrange them how you like. Compare your algorithm to the one in section 1.8 in terms of time complexity and communication complexity.)

Q1 [15]. Consider Peterson's tie breaker algorithm. The time complexity is hard to calculate, since there is no way to tell how long a process might be in its spin loop when there is another process in or trying to get into the critical section. If there is no contention (i.e. if the process spends no time in its spin loop), the 2 process version is $\Theta(1)$ time. Of course it is also $\Theta(1)$ space.

- [5] (a) For the N process version, what are the time and space complexities with-respect to N, assuming that there is no contention.
- [10] (b) Design an algorithm for N process mutual exclusion that is based on multiple rounds of the 2 process tie breaker algorithm, but that uses $O(\log N)$ time and $O(N)$ space, assuming no contention.

Q2 [10 / 15].

(a) [10] Design a parallel, shared-memory version of the quick-sort algorithm. Don't use recursive parallelism in your final algorithm.

(b) [5] (9869 only) Suppose the length of the array is N. If there are $\Theta(\log N)$ processors, is the best-case time for your algorithm $\Theta(N)$? Explain why or why not.

Q3 [10]. Run-length. Given an array of N values, calculate, for each item, the number of contiguous, subsequent items that have the same value. Design the algorithm to run in $\Theta(\log N)$ time on N processors

In:	a	a	a	b	b	a	b	b
Out:	2	1	0	1	0	0	1	0

Bonus [10]. Given a deterministic finite state recognizer, with the transition table represented so that the next transition takes $\Theta(1)$ time to calculate. Find an $O(|Q| \log N)$ time parallel algorithm to recognize a string of length N (represented as an array), where the machine has $|Q|$ states. You may use up to N processors.

[1] The amount of local memory your algorithm requires for each process should not be a function of N.