PARALLEL COMPUTING, EXERCISE SESSION 4

27TH OF DECEMBER, 2014

Exercise 1. Consider a 'master-slave' algorithm, in which the master processor first distributes work to the slaves, and at the end collects and processes the results produced by the slaves. Assume that there is no parallel overhead. Assume that the master's task takes a factor s of the total execution time T_{seq} before the work can be done in parallel.

(1) Calculate the speedup and the parallel efficiency for:
   - $p = 20$, and $s = 0.1$ or $s = 0.001$,
   - $p = 1000$, and $s = 0.1$ or $s = 0.001$.

(2) Suppose that for the same algorithm you measure a speedup S of 3.8 on 4 processors. Which speedup do you expect for $p = 64$ and for $p = 128$ processors?

(3) What is the maximum speedup for $s = 0.1$ as $p \to \infty$? What if $s = 0.001$?

Exercise 2. Recall the parallel three-superstep BSP dot-product algorithm as well as its tree-based counterpart. For each of the two algorithms,

(1) determine the overhead function and parallel efficiency.

(2) how does the algorithm scale with the number of processors, assuming g, l, n are constant?

(3) how does the algorithm scale with the number of processors, assuming g, n are constant and l is linear in p?

(4) on the Cray T3E computer, for $p = 64$, we have $g = 78$ and $l = 1825$. What is the parallel efficiency for $n = 1000$? What for $n = 5 \cdot 10^6$? Which algorithm performs best and why?

Exercise 3. Consider 2D and 3D square grids, with M the number of grid points. Consider a simple stencil operation (2D: five point stencil; 3D: seven point stencil). Consider p processors and 1D, 2D, and 3D blockwise partitionings. Derive formulas for the parallel runtime (T_p) and the parallel enfficiency (E). Discuss the dependence on M, p, and $n = M/p$ (the number of points per processor).

Exercise 4. The sieve of Eratosthenes is an algorithm for finding the prime numbers contained in $\{2, 3, \ldots, n\}$:

```
1. allocate an array X of length n
2. set all entries to 1 (true)
3. let p=2
4. while there is a 1 in X, do
5.    set all multiples of p in X to 0 (false)
6.    let p be the index of the first 1 in X with index larger than the current p
7. end while, done (the entries in X marked true are primes)
```

Try to understand this sequential algorithm and address the following tasks:

(1) For a given p selected at lines 3 or 6, is it useful at line 5 to cross out (set to 0) multiples of p smaller than p^2?

(2) Given the above, if p is larger than $\sqrt{n}$, does it make sense to continue the while loop 4–7?

(3) Consider cyclic and block partitionings of the data for parallelisation purposes. Modify the sequential algorithm to obtain a parallel sieve for each of these distributions; write these algorithms in BSP-style pseudo-code, and incorporate the optimisations the above allude to.

(4) Try to find the advantages and disadvantages of using cyclic and block distributions. Which one is more suitable? Hint: work out small examples.

(5) Try to find an expression of the parallel overhead in terms of n, p, g, l of your chosen algorithm.

Exercise 5. The parallel bubble-sort algorithm has parallel runtime $T_p = \Theta(n/p \log(n/p) + n + ng + 2pl)$. How does this compare to the runtime of the parallel quicksort algorithm presented last week, e.g., what is the asymptotic difference in performance?