Spring'16: CS415/515 Parallel Programming 5/25/16

Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu); Class: MW 2:00-3:50pm @ FAB 40-07; Office Hr: MW 1-2pm & by appt.

Lab 9: Practice Problems

1. (Data Partition) The following code is to be mapped to a message-passing machine. Consider two partition choices for the array `a`:

   (a) partition only the first dimension; and

   (b) partition only the second dimension.

   Both vectors `b` and `x` are aligned with the second dimension of `a`, however they can be distributed with any of the columns of `a`.

   ```
   for (j = n-1; j >= 0; j--) {
     x[j] = b[j] / a[j][j];
     for (i = 0; i < j; i++) {
       b[i] -= x[j] * a[i][j];
       a[i][j] = 0;
   }
   ```

   For each of the partition/distribution combination, decide what form of communication (e.g. shifts, broadcast, etc.) will be needed for the loop nest. Which choice do you think is better?

2. (Chapel) Consider the following Chapel program, `test.chpl`:

   ```
   use BlockDist;
   const D = {0..3};
   const MD = D dmapped Block(D);
   var a: [D] int = 0;
   var b: [MD] int = 0;

   writeln("Case 1:");
   forall i in D {
     writeln(i, ": ", here.id, ", ", a[i].locale.id, ", ", b[i].locale.id);
   }
   writeln("Case 2:");
   forall i in MD {
     writeln(i, ": ", here.id, ", ", a[i].locale.id, ", ", b[i].locale.id);
   }
   ```

   When the program is executed with the command

   ```
   linux> ./test -nl 4
   ```

   What gets printed?

3. (Bitonic Sort)

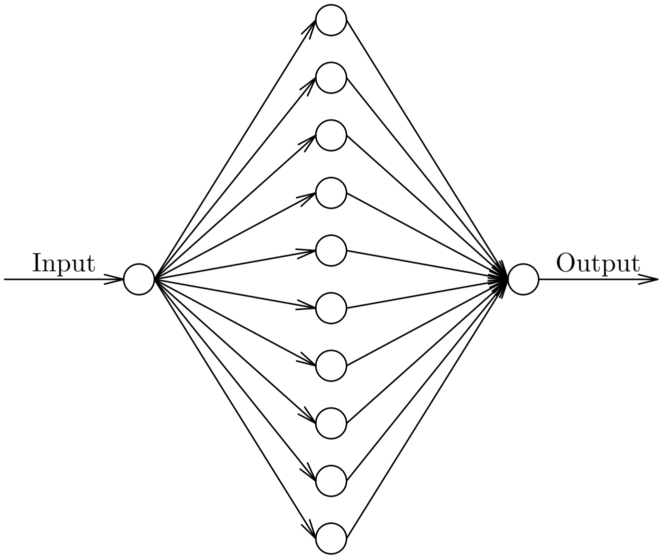
   (a) Which of the following sequences are bitonic sequences?

   i. `3,6,4,2` ; ii. `4,6,5,1,3` ; iii. `6,2,3,6,9,7` ; iv. `1,3,6,4,7,9`

   (b) Using the bitonic sort algorithm, show the steps of sorting the bitonic sequence, `3,8,9,6,5,4,1,2`,

4. (External Sorting) In external sorting, data needs to be read in from disk and write back to disk. A round-trip from and to disk is called a `pass`. In lecture, we showed a statement *"Most (external sorting) algorithms use only two passes."* Are two passes *always* sufficient for sorting? Prove or disprove.

5. (Speedup) The task graph shown here represents an application. Each bubble represents an inherently sequential task. There are 12 tasks: an input task, 10 computation tasks, and an output task. Each of the 12 tasks can be accomplished in 1 unit of time on one processor. The input task must complete before any computational tasks begin. Likewise, all 10 computational tasks must complete before the output task begins.



   (a) What is the maximum speedup that can be achieved if this problem is solved on two processors?

   (b) What is an upper bound on the speed up that can be achieved if this problem is solved with parallel processors?

   (c) What is the smallest number of processors sufficient to achieve the speedup given in part (b)?

6. (Performance Analysis) Consider the following code:

```
for (j=0; j<m; j++)
  forall (i in 1:n)
    x[i] = 2 * (x[i-1] + x[i+1]);
```

Consider a parallel machine with p processors. Assume that the vector `x` is partitioned over the processors with a simple block strategy. Also assume that computing the new value for an element costs one unit of time and communicating an element to a neighboring processor costs 50 units of time. (Count sends and receives as separate operations.)

   (a) Give an expression for the sequential execution time for the program.

   (b) Give an expression for the parallel execution time for the program.

   (c) Give an expression for the speedup as a function of n and p.

   (d) What is the maximum speedup possible if $n = 2,000$?

   (e) Assume the value of elements-per-processor is fixed at 200, calculate the fix-memory speedup; express it as a function of p.