CSE 6220
Introduction to High Performance Computing
Spring 2020 - Midterm II Solutions

Note:

- The exam has 5 questions for a total of 25 points. Please make sure that you have all the questions.
- Clearly state any assumptions you make in the solution of a problem.

Run-time of Communication Primitives and Sorting:

1. *Broadcast, Reduce, AllReduce*: $O((\tau + \mu m) \log p)$
2. *Scatter, Gather, AllGather*: $O(\tau \log p + \mu m p)$
3. *All-to-All*: $O(\tau p + \mu m p)$
4. *Many-to-Many*: $O(\tau p + \mu(r + s))$, where s and r denote the maximum total outgoing communication size and maximum total incoming communication size per processor, respectively.
5. *Bitonic Sort*: Comp: $O\left(\frac{n}{p} \log \frac{n}{p} + \frac{n}{p} \log^2 p\right)$ Comm: $O\left(\tau \log^2 p + \mu \frac{n}{p} \log^2 p\right)$
6. *Sample Sort*: Comp: $O\left(\frac{n \log n}{p} + p \log^2 p\right)$ Comm: $O\left(\tau p + \mu\left(\frac{n}{p} + p \log^2 p\right)\right)$

Questions:

1. (5 points) Consider an n node tree with nodes $v_0, v_1, \ldots, v_{n-1}$ that are distributed among p processors using block decomposition, i.e., P_i contains nodes $v_{i\frac{n}{p}}, v_{i\frac{n}{p}+1}, \ldots, v_{(i+1)\frac{n}{p}-1}$, for $0 \leq i < p$. Let's assume the maximum number of children per node is bounded by c. Each node also contains the indices at which its parent and children are stored. For each of the following operations, which communication primitive will you use and what is the runtime?

   (a) Each node in the tree has $O(1)$ sized data that should be sent to its parent.

   (b) Each node in the tree has $O(1)$ sized data that should be sent to all its children.

   Solution:

   Use `many-to-many` communication primitive in both cases. Each processor has $\frac{n}{p}$ tree nodes. Let S and R denote the maximum possible size of a message that would be sent and received by a processor.

   (a) $S \leq \frac{n}{p}$, $R \leq c\frac{n}{p}$

   (b) $S \leq c\frac{n}{p}$, $R \leq \frac{n}{p}$

In both cases, runtime would be the same:

$$\text{Hypercubic Permutation} = O\left(\tau \log p + \mu \frac{n}{p} \log p\right)$$

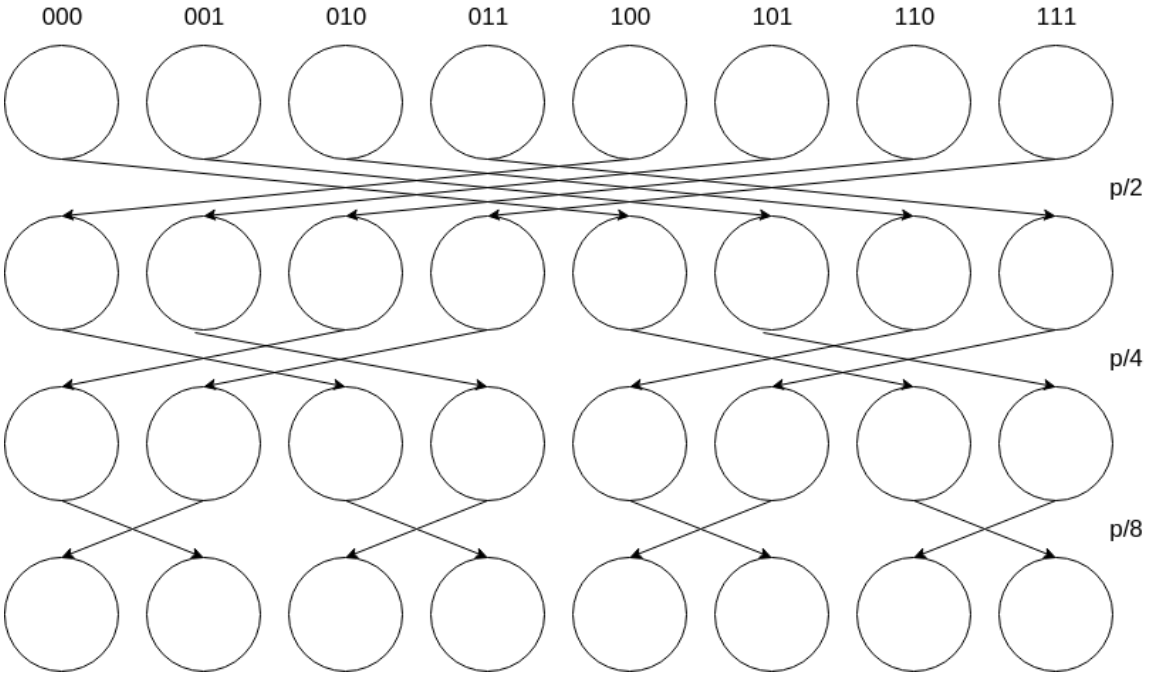
$$\text{Arbitrary Permutation} = O\left(\tau p + \mu \frac{n}{p}\right)$$

2. (5 points) Given a reduction operator `op`, a new reduction communicative primitive is defined as follows. On a p processor system, each processor has an array of size p. As a result of executing the primitive, processor with rank i should have the reduction of the i^{th} elements of all the arrays, using the operator `op`. Design an efficient algorithm for this primitive and compute its run-time. To simplify your analysis, assume p is a power of two.

   Solution:

   Let the array on each processor be A_i. This problem can be solved in $\log p$ iterations as follows:

   (a) Divide the p processors into two groups such that P_0 to $P_{p/2-1}$ belong to the left group and $P_{p/2}$ to P_{p-1} belong to the right group.
   (b) Every P_i in the right group sends $A_i[0, \frac{p}{2} - 1]$ to P_j in the left group, where $j = i - \frac{p}{2}$.
   (c) Similarly, every P_j in the left group sends $A_i[\frac{p}{2}, p - 1]$ to P_i in the right group, where $i = j + \frac{p}{2}$.
   (d) Every processor computes reduction of the received elemet with its local A_i using `op`.
   (e) The left and right groups represent subproblems on $p/2$ processors using their respective parts of A (i.e., $A_i[0, \frac{p}{2} - 1]$ and $A_i[\frac{p}{2}, p - 1]$). Repeat the above steps on each group separately until each processor has one element.



   The run time complexity is given by

- Computation cost $= O(p/2 + p/4 + \ldots 1) = O(p)$
- Communication cost $= O(\tau \log p + \mu(p/2 + p/4 + \ldots 1) = O(\tau \log p + \mu p)$

3. (6 points)

   (a) (3 points) Draw an 8-element bitonic sorting circuit to sort elements in **non-increasing** order. Use horizontal lines for the numbers and vertical lines to denote comparison operations. Label the comparison operations as $\uparrow$ or $\downarrow$ where the direction of the arrowhead indicates the destination of the smaller element. Illustrate how the input

   $$(10, 14, 9, 6, 17, 3, 8, 4)$$

   is sorted using the diagram.

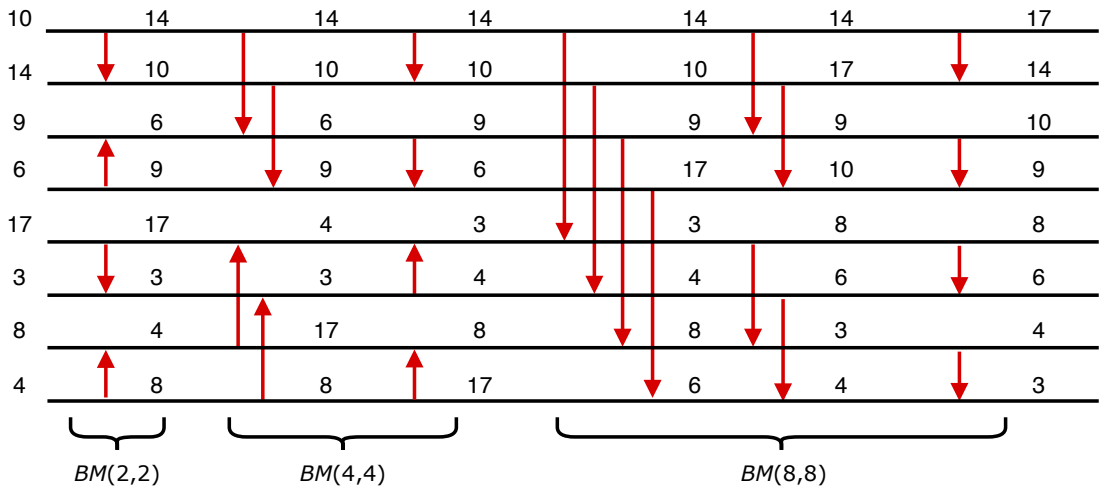
   (b) (3 points) Assume you have 3 processors and following input

   $$(10, 14, 9, 6, 17, 3, 16, 8, 4, 13, 5, 11, 15, 1, 12, 2, 7, 18)$$

   is evenly distributed to processors using block decomposition. Illustrate the steps of the sample sort algorithm and sort this input in **non-decreasing** order.

Solution:

(a) Bitonic Sort Diagram:



(b) We have $n = 18$ and $p = 3$, so each processor will start with $n/p = 6$ elements. The local splitters are chosen to partition the local arrays into p parts, so each part will have 2 elements.

	Processor 0	Processor 1	Processor 2
Initial data	10, 14, 9, 6, 17, 3	16, 8, 4, 13, 5, 11	15, 1, 12, 2, 7, 18
Locally sort and select local splitters	3, **6**; 9, **10**; 14, 17	4, **5**; 8, **11**; 13, 16	1, **2**; 7, **12**; 15, 18
Sort candidate splitters and select global splitters	2, **5**	6, **10**	11, 12
Split on global splitters	1, 2, 3, 4, 5	6, 7, 8, 9, 10	11, 12, 13, 14, 15, 16, 17, 18

4. (4 points) Give an algorithm to merge two sorted sequences of lengths m and n, respectively. You may assume that the input is an array of length $m + n$ with one sequence followed by the other, distributed across processors such that each processor has a subarray of size $\frac{m+n}{p}$. What are the computation and communication times for your algorithm? (You may assume the use of any permutation-style communication, not necessarily hypercubic.)

   Solution:

   Without loss of generality, we can assume that $m \geq n$. Reversing the second sorted sequence results in a bitonic sequence of length $m + n$. To reverse the second sorted sequence, each processor need to communicate with at most 2 processors and the communication time will be $O(\tau + \mu(m+n)/p)$. Bitonic merging of a sequence of length $m+n$ using p processors takes $O((\tau + \mu(m+n)/p) \log p)$ communication time and $O((n+m)/p \log p)$ computation time.

5. (5 points) Consider the embedding of a p-leaf complete binary tree onto p-processors using hypercubic permutations as discussed in class. In this embedding, we assigned the leaves of the tree to target processor ranks according to the binary code ordering. Suppose that we change this algorithm such that leaf j is mapped to processor $(b + j) \mod p$, for a given b, such that $1 \leq b < p$. We still map an internal node of the tree to the processor to which its left child is mapped. Show that in this mapping, two processors that are connected in the tree topology may be assigned to processors in the target topology whose ranks differ in more than one bit position. (**Hint:** Just show a counterexample, and clearly state parameter choices.)

   Solution:
   One simple counterexample is given below with $p = 4$ and $b = 1$. In this embedding, processors 01 and 10 are connected (as are 11 and 00) but they differ in more than 1 bit.