Parallel Computing
COMP 633  Fall 2018
Written Assignment #3: Sample Solutions

Assigned: Thu Nov 15
Due: Tue Dec 4 (at start of class)

1. [7] Let $H[1:n]$ be an array of integer values in the range $1..k$, with that $1 \leq k \leq n$. We want to find the most frequently occurring value $m$ in $H$, i.e. the mode of $H$. For example, with $n = 8, k = 4$, and $H = [1, 3, 1, 3, 4, 3, 1]$, we should find $m = 3$. For simplicity you can assume that the mode is unique.

(a) Verify that the sequential time complexity for this problem is $\Theta(n)$.

(b) Describe an efficient parallel BSP algorithm for this problem using $p$ processors assuming the condition $n = kp$ with $k \geq p$ and give its BSP cost. The input $H$ is distributed evenly over the processors, so initially each processor holds $k = n/p$ input values (assume $p$ divides $n$ evenly). The result $m$ should be available in the first processor on termination.

**Sample Solution**

(a) To understand whether a solution is efficient, we need to determine the sequential time complexity of the problem. In general, finding the mode of a collection of values requires $\Omega(n \log n)$ time. However, our mode problem is restricted, as values in $H[1:n]$ are drawn from the range $1..k$ and we are given $k \leq n$. Thus we can histogram the values in $H$ using an auxiliary array of size $k$ and determine the mode in time $O(k + n) = O(n)$. This is also a lower bound since we have to examine all $n$ input values. Thus the sequential time complexity for our problem is $\Theta(n)$.

(b) Under the BSP model $H$ is distributed evenly across processors with each processor holding $n/p$ values. Following the strategy of the sequential algorithm, we can histogram the occurrences of values in $H$ and identify the value with maximum occurrence count. We do this in three supersteps as follows.

1. Each processor histograms its $n/p$ local values across $k$ bins. Next, this histogram is partitioned into $p$ groups each with $k/p$ bins and a total exchange of the groups among the $p$ processors is performed so processor $i$ receives counts of values in group $i$ from each processor. The BSP cost for this step is

   $O(n/p) + p(k/p)g + L = O(n/p) + k \cdot g + L = O(n/p) + O(n/p) \cdot g + L$

2. Each processor receives $k/p$ mode candidates and $p$ counts for each candidate. These counts are summed to create $k/p$ elements of the complete histogram in each processor. Each processor selects the value with the maximum count in its group and sends it to processor $1$. Since $p = n/k < n/p$, the BSP cost for this step is $O(n/p) + O(n/p) \cdot g + L$

3. Processor 1 examines the $p$ candidates for the mode received from the previous step to find $m$, the mode of $H$. Since $p = n/k < n/p$, the BSP cost is $O(n/p) + L$.

Thus the total BSP cost is $O(n/p) + O(n/p) \cdot g + 3 \cdot L$. The BSP algorithm is work-efficient since the work term is $O(n/p)$. The algorithm’s communication overhead depends on the network since
\[ \lim_{n \to \infty} \frac{O(n/p) \cdot g + 3 \cdot L}{O(n/p)} \approx g \] (assuming the BSP implementation and the sequential implementation perform similar total work, which is reasonable in this case).

2. [10] The Discrete Fourier Transform (DFT) of a sequence of complex values \( X[0:n-1] \) and \( n = 2^r \) yields complex values \( Y[0:n-1] \) such that
\[
Y_i = \sum_{0 \leq k < n} X_k \omega^{ki} \quad \text{where} \quad \omega = e^{2\pi i / n}.
\]
The radix-2 FFT to compute \( Y[0:n-1] \) (in bit-reversed index order) can be expressed as a W-T model algorithm with \( S(n) = O(\log n) \) and \( W(n) = O(n \log n) \) as follows

```plaintext
forall i \in 0:n-1 do
    Y[i] := X[i]
end for
for m := 0 to r-1 do
    forall i \in 0:n-1 do
        let \((b_0 \ldots b_{m-1} b_m b_{m+1} \ldots b_{r-1})\) be the binary representation of \( i \)
        int j := \((b_0 \ldots b_{m-1} 0 b_{m+1} \ldots b_{r-1})\)
        int k := \((b_0 \ldots b_{m-1} 1 b_{m+1} \ldots b_{r-1})\)
        int h := \((b_n b_{m-1} \ldots b_0 0 \ldots 0)\)
        Y[i] := Y[j] + Y[k] \cdot \omega^h
    end for
end for
```

(a) Construct an algorithm where \( n = p \) with BSP cost \( O(\log p)(1 + g + L) \).
(b) Construct an algorithm where \( n = 2^r \) and \( r \geq 2 \log p \) with BSP cost
\[
O(1) \left( \frac{n \log n}{p} + \frac{n \cdot g + L}{p} \right)
\]
In both cases \( X \) and \( Y \) should be distributed evenly over processors.

**SAMPLE SOLUTION**

(a) Observe the FFT communication pattern is naturally suited to processors logically arranged in a boolean hypercube because in each step of the algorithm we compare values in \( Y \) whose index differs in the same digit position when viewed as a binary value. If we distribute \( n = p = 2^k \) values of \( Y \) in processor index order then in iteration \( 0 \leq m \leq r - 1 \), one of \( y_j \) and \( y_k \) is at processor \( i \), and the other is at processor \( nb_{r-m}(i) \), where \( nb_d(i) \) is the label of the processor across dimension \( d \), i.e. the processor whose index differs from \( i \) only in bit \( d \).

Consequently, if we vary the dimension \( j \) from \( r \) down to 1 on successive iterations, the communication pattern is a simple exchange of \( y \) values across dimension \( j \), followed by an update of the local \( y \).

Here is the program for processor \( i \):
for $j := r$ downto 1 do

\[
y' := \text{value of } y \text{ at } nb(i) \quad \text{SS1}
\]

if $(i^{(i)} = 1)$ then $y, y' := y', y$ endif

\[
h := \text{bitrev}(i \text{ div } 2^{i-1})2^{i-1} \quad \text{SS2}
\]

$y := y + y'^h$

end do

Each iteration of this loop communicates a single value in one superstep, and performs a constant amount of work in another superstep. Since we are alternating communication and computation supersteps on successive iterations, we can combine the computation on iteration $j$ with the communication on iteration $j + 1$, and perform one additional superstep at the end. Since $y$ holds a complex value, we should treat it as a two word transfer to yield a BSP cost of $O(1) + 2g + L$ per superstep. For a total of $\lg p$ iterations, the BSP cost is $O(\lg p)(1 + g + L)$.

(b) For the case $n > p$, we can have an array $Y$ with $n/p$ values in each processor. We can continue to treat the problem as if arranged on a boolean hypercube of dimension $\lg n$, where $\lg p$ dimensions are embedded across processors and $\lg n - \lg p$ dimensions are embedded within the array $Y$ at each processor. Then we can apply a communication optimization similar in strategy that was used to improve the communication efficiency of bitonic sort and the multiscan in radix sort.

Suppose we have $p = 2^q$ processors and $n = 2^k2^q$ values with $k \geq q$, which we view as arranged in a Boolean hypercube of degree $k + q$. If we assume a cyclic decomposition of input values across processors (i.e. successive values in the input are distributed across successive processors), then the first $k$ iterations of the algorithm (communicating values along the highest $k$ dimensions of the hypercube) can be performed entirely locally within each processor, while the last $q = \lg p$ iterations require communication across processors. Since the cost of each such communication between processors is $(n/p)(2g + L)$, this would yield an overall communication cost of $(\lg p)(n/p)(2g + L)$ which is too large.

We can improve the communication cost as follows.

Superstep (1) performs $k$ iterations locally within each processor followed by a transposition of the data from cyclic distribution into block distribution. This will embed the lowest $q$ dimensions entirely within processor memory.

Superstep (2) then applies the last $q$ iterations of the algorithm locally.

The transposition is a total exchange operation in which the $i^{th}$ group of $n/p^2$ values must be moved to processor $i$, hence is an $h$-relation of size $n/p$. On termination the result is equally distributed across processors in a block distribution (the first $n/p$ values are in the first processor, the second $n/p$ values are in the second processor, etc.).

The complete algorithm has two supersteps. The first has cost $k(n/p) + 2(n/p)g + L$. The second has cost $q(n/p) + L$. The total BSP cost is thus

\[
(k + q)(n/p) + 2(n/p)g + 2L = (n \lg n)/p + 2(n/p)g + 2L
\]

and satisfies our target bound in part (b) of the question. For an additional cost of $2(n/p)g + 2L$ we can accept input values distributed evenly across processors in any fashion, so we can relax our assumption of input data arranged in a cyclic distribution while retaining the asymptotic BSP cost.