Lecture 8: Designing Parallel Algorithms
Abhinav Bhavele, Department of Computer Science
Announcements

• Assignment 1 has been released and due on October 5
• Project descriptions are due on September 28
• Quiz on September 30, due on October 1
Writing parallel programs

- Decide the serial algorithm first
- Data: how to distribute data among threads/processes?
  - Data locality: assignment of data to specific processes to minimize data movement
- Computation: how to divide work among threads/processes?
- Figure out how often communication is needed
Prefix sum

• Calculate partial sums of elements in array
• Also called a “scan” sometimes

\[
pSum[0] = A[0]
\]

\[
\text{for}(i=1; \ i<N; \ i++) \ { }
\text{pSum}[i] = pSum[i-1] + A[i]
\]
Parallel prefix sum

| 2 | 8 | 3 | 5 | 7 | 4 | 1 | 6 |
Parallel prefix sum

\[
\begin{array}{cccccccc}
2 & 8 & 3 & 5 & 7 & 4 & 1 & 6 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
2 & 10 & 11 & 8 & 12 & 11 & 5 & 7 \\
\end{array}
\]
Parallel prefix sum

2 8 3 5 7 4 1 6

2 10 11 8 12 11 5 7

2 10 13 18 23 19 17 18
Parallel prefix sum
In practice

- You have $N$ numbers and $P$ processes, $N \gg P$
- Assign a $N/P$ block to each process
  - Do calculation for the blocks on each process locally
- Then do parallel algorithm with partial prefix sums
Parallel Sorting

- Sorting is used in many HPC codes
- For example, figuring out which particles/atoms are within a cutoff radius
- Two broad categories of parallel sorting algorithms:
  - Merge-based
  - Splitter-based
Review QuickSort

- Choose a pivot element from the unsorted list
- Move all elements < pivot before the pivot and all elements > pivot after the pivot
- Recursively apply this to the sublists before and after pivot
Sample Sort

- Generalization of QuickSort
- Instead of selecting one pivot, we select $s-1$ samples randomly
  - This provides us with $s-1$ “splitters”
- Once sorted, these $s-1$ splitters create $s$ buckets
- Keys are then placed in the appropriate bucket
- Call sample sort or quick sort recursively
Parallel Sample Sort

- Assumption: keys are distributed across all processors in the beginning
- Sample s keys randomly from each process
- Bring all keys s * p keys to one process
  - select p-1 splitters from this sorted sample
- Send all splitters to all processes
- Processes exchange data based on buckets
- Call some fast sorting algorithm locally