Parallel Sorting
Merge Sort & Bubble Sort

CS370 Parallel Processing
Spring 2014
Sorting

• Arrange elements of a list into certain order
• Make data become easier to access
• Speed up other operations such as searching and merging
• Many sorting algorithms with different time and space complexities
Parallel Sorting

Design methodology

• Based on an existing sequential sort algorithm
  – Try to utilize all resources available
  – Possible to turn a poor sequential algorithm into a reasonable parallel algorithm (bubble sort and parallel bubble sort)

• Completely new approach
  – New algorithm from scratch
  – Harder to develop
  – Sometimes yield better solution
Bubble Sort

- One of the straight-forward sorting methods
  - Cycles through the list
  - Compares consecutive elements and swaps them if necessary
  - Stops when no more out of order pair

- Slow & inefficient

- Average performance is $O(n^2)$
Parallel Bubble Sort
Odd-even Transposition Sort

• Compare all pairs in the list in parallel
• Alternate between odd and even phases
Parallel Bubble Sort

• When to stop?
• Shared flag, sorted, initialized to true at beginning of each iteration (2 phases), if any processor perform swap, sorted = false
Parallel Bubble Sort Complexity

• Sequential bubble sort, $O(n^2)$
• Parallel bubble sort? (if we have unlimited # of processors)
Parallel Bubble Sort Complexity

- Sequential bubble sort, $O(n^2)$
- Parallel bubble sort?
- Do $n-1$ comparisons for each iteration => $O(n)$
- Seq. Quicksort is $O(n \log n)$

![Diagram of Parallel Bubble Sort](image-url)
Parallel Bubble Sort Example

• How many steps does it take to sort the following sequence from least to greatest using the Parallel Bubble Sort? How does the sequence look like after 2 cycles?

• 4,3,1,2
Parallel Bubble Sort Example

Step 1

Step 2

Step 3

Step 4
Divide and Conquer

• Dividing problem into sub-problems
• Division usually done through recursion
• Solutions from sub-problems then combined to give solution to the original problem
“Divide”
“Conquer”
Merge Sort

- Collects sorted list onto one processor
- Merges elements as they come together
- Simple tree structure
- Parallelism is limited when near the root
Example
Merge Sort Complexity

\[ T(n) = \begin{cases} 
  b & \text{n = 1} \\
  2T \left( \frac{n}{2} \right) + bn & \text{n > 1}
\end{cases} \]

Solve the recurrence relation, (CS331)

\[ T(n) = O(n \log n) \]
Parallel Merge Sort

• Parallelize processing of sub-problems
• Max parallelization achieved with one processor per node (at each layer/height)
Parallel Merge Sort Example

• Perform Merge Sort on the following list of elements. Given 2 processors, P0 & P1, which processor is responsible for which comparison?

• 4,3,2,1
Parallel Merge Sort Example

4, 3, 2, 1

P0

4, 3

P0

4

3

P0

3, 4

2, 1

P1

2

1

P1

1, 2

P0

1, 2, 3, 4

P0
Parallel Merge Sort Complexity

- Merge sort, $O(n \log n)$
- Easy way to remember complexity, $n$ (elements) $\times$ logn (tree depth)
- If we have $n$ processors, $O(\log n)$
Parallel Bubble Sort KLA

• Sort a list of #s from smallest to largest
• Each person represent a processor
• Only has to do 1 comparison per phase
• Compare and swap if necessary
Parallel Merge Sort KLA

- Sort a list of #s from smallest to largest
- Each person represent a processor
- Merge sort in parallel
Post-test

• Try your best!