CSE160
Introduction to parallel computing
Discussion Section Week 2

Yashodhan Karandikar
Jing Zheng
10-09-2013
Outline

• Quiz
• Announcement
• Parallel Merge Sort
Announcement

• Signature change in MergeSort.cpp:

Serial implementation in Provided code

```cpp
bool DC_MergeSort(std::vector<int> *keys0, int l0, int l1, 
std::vector<int> *keys1, int level)
```

Your parallel implementation

```cpp
void DC_MergeSort(std::vector<int> *keys0, int l0, int l1, 
std::vector<int> *keys1, int level, bool& new_dir);
```
Merge Sort

• A divide and conquer algorithm
• Stop the recursion: each thread locally sorts its data using a fast serial algorithm like quicksort
• Merge data in odd-even pairs
Merge Sort

N values to be sorted

Each thread sort N/NT values

Merge the odd-even pairs

Sorted values
Parallel Opportunities

• Pseudo code

(1) MergeSort( keysIn, start, end, keysOut, level)
(2) n = (end-start)+1
(3) mid = start + n/2

(4) if(n<=minN) //do local sort
(5) LSort(keysIn, start, end, level);
(6) else{
(7) MergeSort(keysIn, start, mid-1, keysOut, level+1)
(8) MergeSort(keysIn, mid, end, keysOut, level+1)
(9) }

(10) Merge(keysIn, keysOut, start, end);
Parallel Opportunities

• Pseudo code

```plaintext
(1) MergeSort( keysIn, start, end, keysOut, level)
(2) n = (end - start) + 1
(3) mid = start + n/2

(4) if(n<=minN) // do local sort
(5) LSort(keysIn, start, end, level);
(6) else{
(7) MergeSort(keysIn, start, mid-1, keysOut, level+1)
(8) MergeSort(keysIn, mid, end, keysOut, level+1)
(9) }

(10) Merge(keysIn, keysOut, start, end);
```
Serial Merge

First merge level
Thread 0

Second merge level
Thread 0

No Parallelism!
Serial Merge For Each Odd-Even Pair

First merge level

Thread 0

Second merge level

Thread 0

Thread 1
Serial Merge vs. Parallel Merge

• Serial merge
  ‣ Each odd-even pair uses only one thread
  ‣ $O(\log n)$ parallelism at most

• Parallel merge
  ‣ Use multiple threads
    • Two threads each pair
    • Recursive parallel merge
Two-thread Parallel Merge

A[]

0

m/2

m

B[]

Thread 0 Merge

Binary search

Thread 1 Merge
Two-thread Merge

• Two-thread parallel merge
  ➢ Each odd-even pair uses two threads
  ➢ Parallelism?
    • Serial merge X2
Recursive Parallel Merge

A[]

0

m/2

m

B[]

Thread 0 Merge

Binary search

Thread 1 Merge

Thread 0 Merge

Thread 2 Merge

Thread 1 Merge

Thread 3 Merge
**Multiple-Thread Merge**

NT = 4

- **Multiple-thread merge**
  - Spawn as many threads as possible to do the merge
  - The higher the recursive tree level is, the more threads available for each odd-even pair
Assignment #1 Tasks

• Parallel merge sort with serial merge per pair
• Two-thread parallel merge (full credit)
• Recursive parallel merge (extra credit)
Thanks!
Q&A