

CSE 332: Data Structures and Parallelism

Spring 2022

Richard Anderson

Lecture 22: Parallel Algorithms

Announcements

- This week:
 - Monday: Parallel Sorting
 - Wednesday: Concurrency
 - Friday: Holiday
 - Monday: Concurrency

Data Parallel Programming

- Programming primitives for operating on Arrays
 - Reduce: Combine array elements with an operator, e.g., +.
 - Map: Apply an operation to every element, e.g., multiply by two
 - Prefix sum: Compute all partial sums
 - Pack: Shift all values satisfying a predicate to start of the array

Parallel Prefix

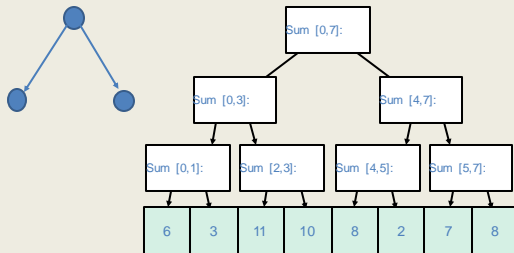
- Prefix-sum:

input	6	3	11	10	8	2	7	8	
output	0	6	9	20	30	38	40	47	55

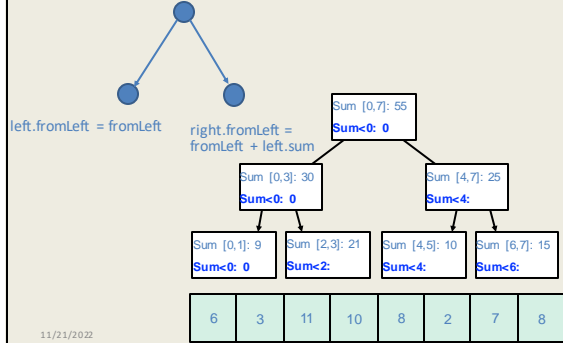
- $output[j + 1] = \sum_{i=0}^j input[i]$
- ForkJoin Implementation
- $O(n)$ work, $O(\log n)$ span

First Pass: Sum

$sum = left.sum + right.sum$



2nd Pass: Use Sum for Prefix-Sum



Parallel Prefix, Generalized

- Parallel Prefix can be generalized to many operators
- Example: Sum bygroup

6	3	*	10	8	2	*	8	6	3	*	*	8	2	*	8
6	9	*	10	18	20	*	8	14	17	*	*	8	10	*	8

$$A \bullet B = A + B \quad \underline{A} \bullet B = \underline{A + B} \quad * \bullet \underline{B} = \underline{B}$$

$$A \bullet * = *$$

$$A \bullet \underline{B} = \underline{B}$$

$$* \bullet B = \underline{B}$$

$$\underline{A} \bullet \underline{B} = \underline{B}$$

$$* \bullet * = *$$

$$\underline{A} \bullet * = *$$

11/21/2022

CSE332

7

Parallel Pack

1. map test input, output [0,1] bit vector

IsPrime(x) ?

input	6	3	11	10	8	2	7	8	13	4	23	5
test	0	1	1	0	0	1	1	0	1	0	1	1

2. prefix-sum on bit vector

pos	0	1	2	2	2	3	4	4	5	5	6	7
-----	---	---	---	---	---	---	---	---	---	---	---	---

3. map input to corresponding positions in output

output	3	11	2	7	13	23	5					
--------	---	----	---	---	----	----	---	--	--	--	--	--

- if (test[i] == 1) output[pos[i]] = input[i]

11/21/2022

CSE332

8

Parallel Algorithms

- T_P is the running time on P processors
- Work**: How long it would take one processor: T_1
- Span**: How long it would take with infinite processors: T_∞
- Goal: parallel algorithm with $T_P \approx T_1/P$
- Assume $P \ll n$

11/21/2022

CSE332

9

Parallel Sorting

- Goal: $O(n \log n)$ work, $O(\log n)$ span
 - We will achieve $O(n \log n)$ work, $O(\log^2 n)$ span
- Look at parallel versions of Quicksort and Mergesort

```

QS(S)                                MS
if |S| < SeqCutoff                    if |S| < SeqCutoff
    return Sort(S)                       return Sort(S)
x = Pivot(S)                            S1, S2 = Split(S)
S1, S2 = Partition(S, x)                 S1 = MS(S1); S2 = MS(S2)
return QS(S1), QS(S2)                   return Merge(S1, S2)
    
```

11/21/2022

CSE332

10

Sequential Quicksort

- Quicksort (review):
 - Pick a pivot $O(1)$
 - Partition into two sub-arrays $O(n)$
 - values less than pivot
 - values greater than pivot
 - Recursively sort A and B $2T(n/2)$ Sort of
- Complexity
 - $T(n) = n + 2T(n/2)$ $T(0) = T(1) = 1$
 - $O(n \log n)$

- How to parallelize?

11/21/2022

CSE332

11

Avoiding bad cases for quicksort

- Quicksort can be $\Omega(n^2)$ with bad pivot choices
- If input is random then Quicksort is $O(n \log n)$ with high probability
- If pivots are random then Quicksort is $O(n \log n)$ with high probability
- Pick 5 elements at random, choose the middle as a pivot

11/21/2022

CSE332

12

Parallel Mergesort

- MergeSort(Arr, lo, hi)
 - Threads to compute MS(Arr, lo, mid), MS(Arr, mid, hi)
 - Merge Arr[lo,mid] and Arr[mid,hi] into Arr[lo,hi]
- Can stop at a sequential cut off

11/21/2022

CSE 332

19

Parallel Merge



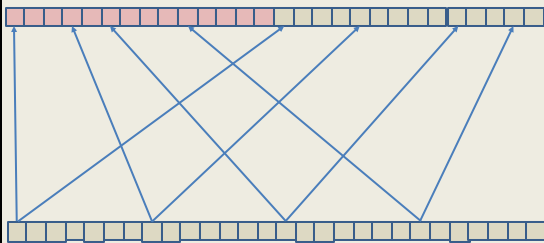
- How to merge two sorted lists in parallel?

11/21/2022

CSE 332

20

Parallel Merge



11/21/2022

CSE 332

21

Parallel Merge: n items with p threads

- Each thread needs to know where to start in the two arrays being merged
- If starting points are given, select the next n/p items
- Finding the starting points can be done in $O(\log n)$ time using a modified binary search

11/21/2022

CSE 332

22

Finding the starting point

- Given two sorted arrays A, B, find the item of rank k in the combined arrays
- Compare $A[k/2]$ and $B[k/2]$
 - If $A[k/2] < B[k/2]$ discard first k/2 items of A, otherwise discard first k/2 items of B
- Look for item of rank k/2 in remaining items
- Logarithmic process

11/21/2022

CSE 332

23

Parallel Quicksort and Mergesort

- Both algorithms can be implemented as efficient parallel algorithms
- With p processors, a speedup of p is achievable provided $p \ll n$
- Speedup comes from:
 - Doing much of the work on sorting items below the sequential cutoff
 - Taking advantage of parallelism in the combine steps to avoid a sequential bottleneck,

11/21/2022

CSE 332

24