CS7960 L14 : Parallel | Sorting

PRAM

1 disk
P processors
n input items

Each time step a processor can:
read, write, operate (+,-,*,<<,...)

shared memory: CRCW (although CREW more realistic)

-------------

Sort (n):
INPUT A = [a_1, a_2, ..., a_n]
Output B = [b_1, b_2, ..., b_n]
so for each a_i = b_j where i->j 1to1, and b_i < b_{i+1}

Sequential?  O(n log n)

PRAM:  O(log^2 n) time,  O(n log n) work

Surplus log n

(possible O(log n) time, O(n log n) work)

---------------------

Merging:
Input A = [a_1, a_2, ..., a_n]
    B = [b_1, b_2, ..., b_n]
    (both sorted, increasing)
Output C = [c_1, c_2, ..., c_{2n}]
    sorted, each c_i = some a_j, or b_j (i.e. sorted merge)

Sequential 0(n)

PRAM:  O(n) work,  O(log n) time

**** Interlude ****
How to get from Merging to Sorting?
    --> Merge Sort!
Arbitrary binary splits into subpieces of size 1 (free)
O(log n) rounds of "merging" sorted lists (each O(log n) time + O(n) work)
How to solve merging problem?
  --> break to arbitrarily small subproblems (i.e. p of size O(n/p)
solve subproblems sequentially on each CPU

---------------------------------

Ranking Problem:

Input A = \{a_1, a_2, \ldots, a_n\}
B = \{b_1, b_2, \ldots, b_n\}
(both sorted, increasing)

Output: A' = \{a'_1, \ldots, a'_n\}
B' = \{b'_1, \ldots, b'_n\}
where a'_i is rank of a_i in B
  b'_i is rank of b_i in A
  i.e. j = rank(i,B) is largest index j of B s.t. a_i > b_j

Sequential : O(n) time  -- scan both lists in parallel, keeping counters in each

Goal:  O(n) work, O(log n) time

---------------------

First Naive O(n \log n) work, O(log n) time
- for each i in A, using binary search in B, to find rank(i,B)
  same for each i in B.
- O(n) elements, each in O(log n) time.
  (surplus-log !)

---------------------

Split A (and B) into n/\log n equal size chunks (size \log n each)
  A_1 = \{a_1, \ldots, a_{\log n}\}
  A_2 = \{a_{1+\log n}, \ldots, a_{2 \log n}\}
  \ldots
  A_{n/\log n} = \{a_{n-\log n}, \ldots, a_n\}
same for B.

For each A_i find which chunk of B it is in.
  O(n/\log n) * O(\log n) work in O(\log n) Ptime.
Same for each B_i in mapped to A

For each chunk of A_i, mapped to chunk B_j, perform sequential Rank (offset by index of B_j).
Same with chunks B_j to chunk A_i.
0(n log n) in O(log n) time/work each = O(n) work, O(log n) Ptime.

Are we done?  Where is the problem?

-------------------

After we get to the end of chunk Bj, we can no longer be confident in our answer for rank (i,B), since it likely spills into B_{j+1} and beyond.

However, solving rank(i,B) (for all i) can be used to solve rank(i,A) (for all i).

A = 1 3 6 7
B = 2 4 5 8
rank(A,B) = 0 1 3 3
rank(B,A) = 1 2 2 4

rank(i,B) = j + rank(i+1,B) = j+k
means that for any l in [j+1,j+k] has rank(l,A) = i

-------------------

So either each Ai can be ranked in matched chunk Bj, or it can be inversely ranked using chunk Bj or B_{j+1}, or larger.

-------------------

Compute Merge(A,B) given rank(A,B) + rank(B,A)

#########################################################
for i=1 to n PARDO
  C(i + rank(i,B)) := A(i)
for i=1 to n PARDO
  C(i + rank(i,A)) := B(i)
#########################################################

O(n) work, O(1) time.

So Rank O(n) Work in O(log n) time --> merge O(n) Work + O(log n) time and
after O(log n) rounds of merges (merge sort)
Sorting O(n log n) Work + O(log^2 n) time.