Sorting (Grama Ch 9, Matloff Ch 12)

Common operation
Internal (in memory) vs external (using supplementary storage)
Lot of different sequential algorithms
Comparison Based
bubble, insertion, shell, merge, quick, ...
non-comparison based
radix (O(n * keysize))
Compare-exchange, O(n log n) best algorithms
Parallel compare-exchange best expected O(log n)
Basis for compare-exchange can be modeled by a circut
circuits are another model of parallel computation!

Batcher's bitonic mergesort

bitonic sequence
at most two "changes of direction"
and remains so for any rotation of the sequence
4 2 1 3 7 8 10 9
originally designed for circuits, can be generalized to processes
gates -- the computation elements
width of circut -- parallelism, time -- parallel time
Bitonic merge
compare corresponding items in each half
results in two bitonic sequences ready to be merged, all numbers in one greater than other

 3
 7
 8
 10
 9
 4
 2
 1
 (4 away comparisons)

 3
 4
 2
 1
 9
 7
 8
 10
 (2 away comparisons)

 2
 1
 3
 4
 8
 7
 9
 10
 (1 away comparisons)

 1
 2
 3
 4
 7
 8
 9
 10

Communication pattern?

Bitonic mergesort (page 2)

Recursive Version
Bitonic merge sort (elements) (assending)
in parallel, sort first 1/2 assending, sort second 1/2 descending
in parallel, merge bitonic sequence

 \Box Iterative bitonic mergesort algorithm, p == n

for i = 1 to log p

perform a bitonic merge on PE groups of size 2^h in alternating directions

 \Box Bitonic merge sort, p < n?

□a) Local sort

□b) "exchange in merge" sends up to n/p elements neighbor and saves lower or upper half

Quicksort

O(n^2) worst case, O(n log n) average

Algorithm □ Partiton to "left, pivot, right" □ sort left and right with quicksort.

First try:
Do both sub sorts in parallel.
pivot takes O(n)
log n rounds
time bounded below by n (more like n log n)
time-processor product bounded below by n^2

Pivot in O(1)?

 \Box if so, time bounded below by log n

 \Box only on a PRAM!

CRCW PRAM algorithm

Arbitrary model CRCW Algorithm builds a binary tree □ pivot is root □ less is left, greater is right □ sorted is inorder traversal □ must move data so tree is inorder in array □ Does this sound parallel?

CRCW algorithm ... build tree!

□Data/arrays

 \Box root - common

 \Box parent[1..n] (p[i])

 \Box leftchild[1..n] (lc[i])

 \Box rightchild[1..n] (rc[i])

Algorithm

```
procedure build_tree (a[1...n]) {
 foreach process i {
  root <- i; /* CW step */
  p[i] <-rot; \quad lc[i] <-rc[i] <-n+1; (NULL?)
 if i != root {
  repeat until all processes terminated {
   if a[i] < a[p[i]] or
      (a[i] = a[p[i]] \text{ and } i < p[i]) 
     lc[p[i]] <- i; /* CW step */
     if lc[p[i]] = i { exit; }
     p[i] <- lc[p[i]];
    } else {
     rc[p[i]] <- i; /* CW step */
     if rc[p[i]] = i { exit; }
     p[i] <- rc[p[i]];
```

Final details

```
□ number the nodes (O(log n) on CRCW)
□ How? (Grama leaves it to the reader, problem 9.15)
□ move into position (O(1))
```

Quicksort A[1..n]: □ build_tree (A[1..n]) yeilding p[], rc[], lc[] □ number_nodes (A, p, rc, lc) □ move to proper location

Times?

□Build Tree?

 $\Box O(\log n)$

 \Box Number nodes O(log n)

 \Box Move (O(1))

 \Box Total O(log n)

Shared Memory version □ n numbers, p processors □ initial n/p per processor

Quicksort (A, start, stop, p processors (pstart, pstop))

Processor pstart selects pivot (usual methods)
Broadcasts pivot to all processors in pstart->pstop
Each processor does local partition
Do 2 prefix sums on number < pivot and >= pivot
keep prefix & total sums. S is set < pivot
Using result of Prefix sums
New <- reorded numbers (barrier)
A <- new (barrier)
Recurse: (Sub Quicksorts in parallel)
np <- ceil(|S|p/n+.5)
Quicksort (A, start, start+|S|-1, np, (pstart, pstart+np-1))
Quicksort (A, start+|S|, stop, p-np, (pstart+np-1, pstop))

Times?

□ Select pivot -- O(1), broadcast?, local partition O(n/p), Prefix sums O(log p) □ Reorder O(n/p), expected recursion O(log p) □ Total: O(n/p log p) Quicksort on a message passing machine?

Do the shared memory algorithm in a distributed fashion!

□ prefix sums ... doable

 \Box movement ... All to All comm

□Similar ...

Quicksort on a hypercube? (Matloff 12.1.3 Hyperquicksort)

each PE sorts the sub array assigned to that PE

for i = d downto 1

for each i-cube:

root of the i-cube broadcasts its median to all in the i-cube, to serve as pivot

consider the two (i-1)-subcubes of this i-cube

each pair of partners in the (i-1)-subcubes exchanges data:

low-numbered PE gives its partner its data larger than pivot

high-numbered PE gives its partner its data smaller than pivot

□ Issues?

□ final placement of elements, e.g. sort is sub step in most algorithms □ parallel prefix needed to calculate final location □ final movement?

