Scan

Mark Greenstreet

CpSc 418 – Jan. 13, 2017

Outline:

- Reduce Redux
  - The basic algorithm.
  - Performance model.
  - Implementation considerations.
- Scan
  - Understand how reduce generalizes to a method that produces all N values for a "cumulative" operation in $O(\log N)$ time.
- A few implementation notes

Reduce Redux

Problem statement:

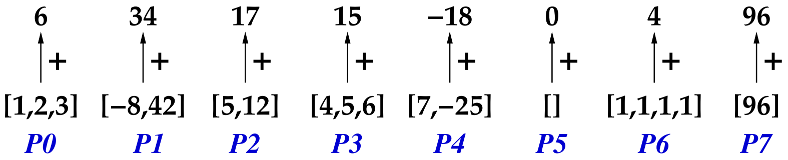
Given P processes that each hold part of an array of numbers, compute the sum of all the numbers in the combined array.

[1,2,3]	[−8,42]	[5,12]	[4,5,6]	[7,−25]	[]	[1,1,1,1]	[96]
P0	***P1***	***P2***	***P3***	***P4***	***P5***	***P6***	***P7***

Reduce Redux

Accumulate step:

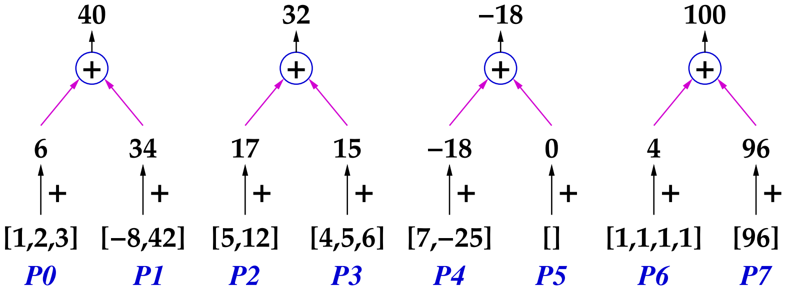
Each process computes the total of the elments in its local part of the array.



Reduce Redux

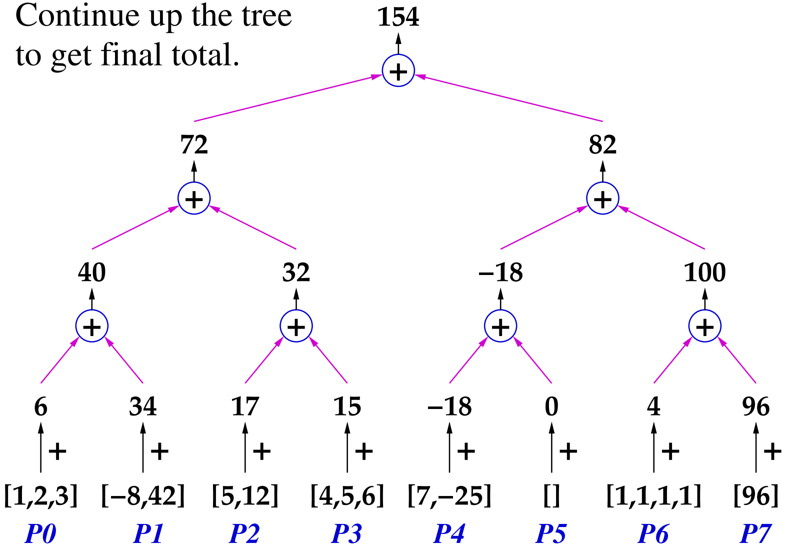
Combine step:

Each process sends its result to a coombiner process.
The combiners compute the sums of the values from adjacent pairs of processes.



Reduce Redux

Continue up the tree to get final total.



Reduce Notes

- For simplicity, I drew the tree as if we used separate processes for accumulating the local arrays and doing the combining.
  - In practice, we use the same processes for both accumulating and combining.
  - Note that ½ of the processes are active in the first level of combine; ¼ of the processes are active in the second level; and so on.
- Simple time model:

$$T \in O\left(\frac{N}{P} + \lambda \log P\right)$$

where λ is **big** – i.e. the communication time.

Scan Problem Statement

- Given an array, A, with N elements.
  - Let $B = \text{scan}_+(A)$:

$$B_i \;=\; \sum_{k=0}^{i} A_k$$

  - Example:

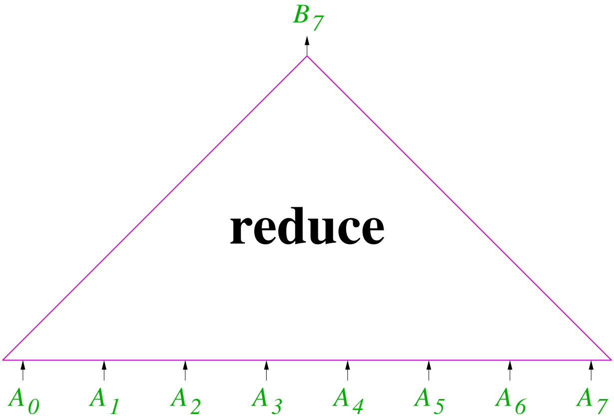
$$\begin{array}{rcl} A & = & [1,2,3,-8,42,5,12,4,5,6,7,-25,1,1,1,1,96] \\ B & = & [1,3,6,-2,40,45,57,61,66,72,79,54,55,56,57,58,154] \end{array}$$

- Is there an efficient parallel algorithm for computing $\text{scan}_+(A)$?
  - I wrote scan_+ because our solution works for any associative operator.

Scan Example: Monthly Bank Statement

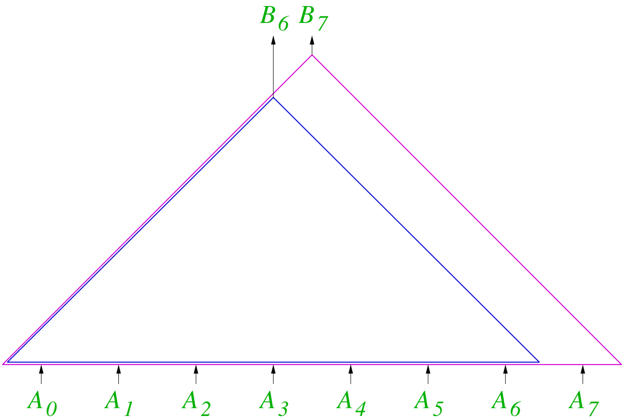
- Assumptions:
  - You make **lots** of transactions; so, the bank needs to use a parallel algorithm just for your account.
  - Months have 32 days – the power-of-two version of the algorithm is simpler. It generalizes to any number of processors.
  - Each processes has the transaction data for one day.
- Using parallel scan:
  - Each process computes the total of the transactions for its day.
  - Using parallel scan, we determine the balance at the beginning of each day for each process.
  - The process can use its start-of-day balance, and compute the balance after each transaction for that day.

Brute force Scan



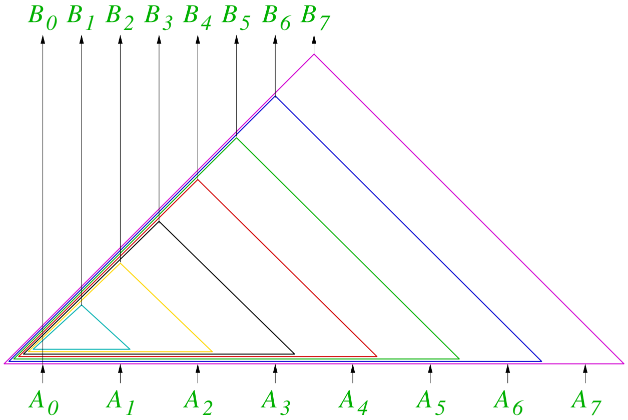
- Use a reduce tree to compute B_7.

Brute force Scan



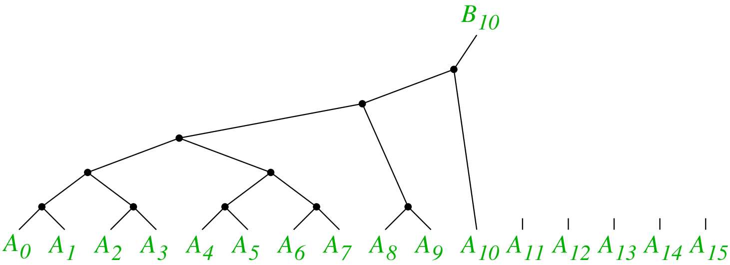
- Use a reduce tree to compute B_7.
- Use another reduce tree to compute B_6.

Brute force Scan



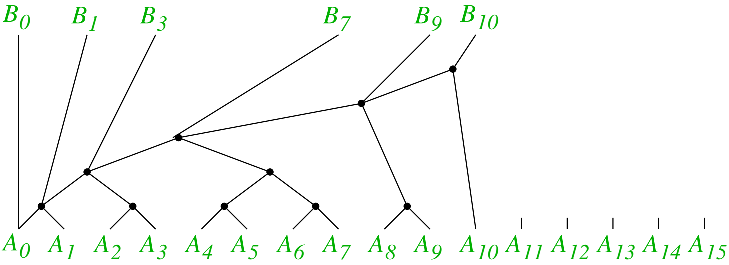
- Use a reduce tree to compute B_7.
- Use another reduce tree to compute B_6.
- Use 6 more reduce trees to compute $B_{5...0}$
- It works. It's $O(\log P)$ time! But it's not very efficient.

Reuse trees



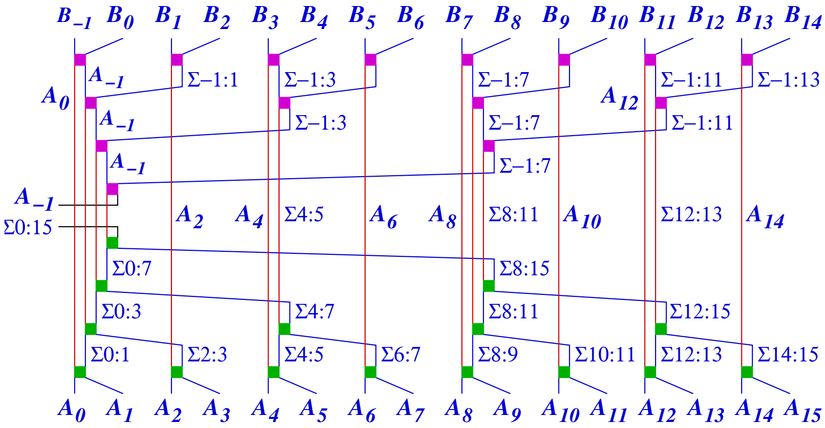
- Key idea: we don't need the trees to be balanced.
- We just want them to be $O(\log P)$ in height.
- If we need a tree for 2^k nodes, we'll make a balanced tree.
- Otherwise:
  - Make the largest balanced tree we can on the left.
  - Repeat this process for what's left on the right.

Reuse trees



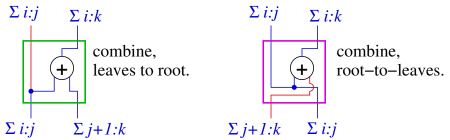
- Key idea: we don't need the trees to be balanced.
- We just want them to be $O(\log P)$ in height.
- If we need a tree for 2^k nodes, we'll make a balanced tree.
- Otherwise:
  - Make the largest balanced tree we can on the left.
  - Repeat this process for what's left on the right.
- Notice that while computing B_{10}, we produced many other of the Bs as intermediate results.

Scan



- See the next slide for an explanation of the notation, etc.

Scan: explained



The green and magenta boxes are both "combine" units.The only difference is the terminal placement, to make the big diagram less cluttered.

- Notation
  - A_{-1} is initializer for the sum.
  - $A_0, A_1, \ldots A_{15}$ is the initial array.
  - $B_{-1}, B_1, \ldots B_{15}$ is the result of the scan.

$$B_i = \sum_{k=-1}^{i} A_k, \text{Include the initializer } A_{-1}$$

  - $\Sigma i : j$ is shorthand for $\sum_{k=i}^{j} A_k$
- Each process needs to compute its local part of the scan at the end, starting from the value it receives from the tree.

A few implementation notes

- On slide 3 I pointed out that for efficiency, it is better to use the same processes for the leaves and the combine.

```
% reduce:
   treeLevels = ceil(log2(NProcs0);
   tally = localAccumulate(...);
   for(k = 0; k < treeLevels; k++) {
      if((myPid & (1 << k)) != 0) {
         send(myPid - (1 << k), myPid, tally);
         break;
      } else
         tally += receive(myPid + (1<< k));
   } // Process 0 now has the grand total.
   // We can use another loop to broadcast the result.
```

- I'll provide an Erlang version on Monday.

Reduce & Scan

Scan is very similar to reduce. We just change the downward tree.

- For reduce, each process just forwards the grand total to its descendants.
- For scan:
  - Each process records the tallies from its left subtree(s) during the upward sweep.
  - During the downward sweep, each process receives the tally for **everything to the left of the subtree for this process**.
    - The process adds the tally from its own left subtree to the value from its parent, and sends this to its own right subtree.
    - The process continues the downward sweep for its own left subtree.
    - When we reach a leaf, the process does the final accumulate.

Preview

January 16: Generalized Reduce and Scan		
	Homework:	Homework 1 deadline for early-bird bonus (11:59pm)
		Homework 2 goes out (due Feb. 1) – Reduce and Scan
January 18: Reduce and Scan Examples		
	Homework:	Homework 1 due **11:59pm**
January 20: Architecture Review		
	Reading:	Pacheco, Chapter 2, through section 2.2
January 23: Shared Memory Architectures		
	Reading:	Pacheco, Chapter 2, through section 2.3
	Homework:	Homework 2 deadline for early-bird bonus (11:59pm)
		Homework 3 goes out (due Feb. 17)
January 25: Message Passing Architectures		
	Homework:	Homework 2 due **11:59pm**

January 27–February 6: Parallel Performance

February 8–17: Parallel Sorting

Review Questions

- What is the cumulative sum of `[1,7,-5,12,73,19,0,12]`?
  - For the same list as above, what is the cumulative product?
  - For the same list as above, what is the cumulative maximum?
- Draw a tree showing how the sum (simple, not cumulative) of the values in the list above can be computed using reduce. Assume that there are eight processes, and each starts with one element of the list.
- Draw a graph like the one on slide 8 for a scan of eight values.
- Label each edge of your graph with the value that will be sent along that edge when computing the cumulative sum of the values in the list above. Assume that there are eight processes, and each starts with one element of the list.
- Add a second label to each edge indicating whether the value is local to that process or if the edge requires inter-process communication. Write 'L' for local, and 'G' for global (i.e. inter-process communication).