Lecture 7: Map/Reduce

Mack Joyner
mjoyner@rice.edu

http://comp322.rice.edu
Worksheet #6: Associativity and Commutativity

Recap:
A binary function $f$ is **associative** if $f(f(x,y),z) = f(x,f(y,z))$.
A binary function $f$ is **commutative** if $f(x,y) = f(y,x)$.

Worksheet problems:
1) Claim: A Finish Accumulator (FA) can only be used with operators that are **associative and commutative**. Why? What can go wrong with accumulators if the operator is non-associative or non-commutative? You may get different answers in different executions if the operator is non-associative or non-commutative e.g., an accumulator can be implemented using one “partial accumulator” per processor core.

2) For each of the following functions, indicate if it is associative and/or commutative.
   a) $f(x,y) = x+y$, for integers $x, y$, is associative and commutative
   b) $g(x,y) = (x+y)/2$, for integers $x, y$, is commutative but not associative
   c) $h(s1,s2) = \text{concat}(s1, s2)$ for strings $s1, s2$, e.g., $h(“ab”, “cd”) = “abcd”$, is associative but not commutative
Map/Reduce: Streaming data requirements have skyrocketed

- AT&T processes roughly 30 petabytes per day through its telecommunications network
- Google processed roughly 24 petabytes per day in 2009
- Facebook, Amazon, Twitter, etc, have comparable throughputs
- In comparison, the IBM Watson knowledge base stored roughly 4 terabytes of data when winning at Jeopardy
Parallelism enables processing of big data

• Continuously streaming data needs to be processed at least as fast as it is accumulated, or we will never catch up

• The bottleneck in processing very large data sets is dominated by the speed of disk access

• More processors accessing more disks enables faster processing
MapReduce Pattern

• Apply Map function \( f \) to user supplied record of key-value pairs

• Compute set of intermediate key/value pairs

• Apply Reduce operation \( g \) to all values that share same key to combine derived data properly
  — Often produces smaller set of values

• User supplies Map and Reduce operations in functional model so that the system can parallelize them, and also re-execute them for fault tolerance
MapReduce: Map Step

Input set of key-value pairs

Flattened intermediate set of key-value pairs

MapReduce: Reduce Step

Intermediate key-value pairs

Key-value groups

Output key-value pairs

Map Reduce: Summary

• Input set is of the form \{\{(k_1, v_1), \ldots, (k_n, v_n)\}\}, where \((k_i, v_i)\) consists of a key, \(k_i\), and a value, \(v_i\).
  • Assume key and value objects are immutable

• Map function \(f\) generates sets of intermediate key-value pairs, \(f(k_i, v_i) = \{(k_1', v_1'), \ldots, (k_m', v_m')\}\). The \(k_m'\) keys can be different from \(k_i\) key in the map function.
  • Assume that a flatten operation is performed as a post-pass after the map operations, so as to avoid dealing with a set of sets.

• Reduce operation groups together intermediate key-value pairs, \(\{(k', v_j')\}\) with the same \(k'\), and generates a reduced key-value pair, \((k', v'')\), for each such \(k'\), using reduce function \(g\)
Google Uses MapReduce

- **Web crawl**: Find outgoing links from HTML documents, aggregate by target document

- **Google Earth**: Stitching overlapping satellite images to remove seams and to select high-quality imagery

- **Google Maps**: Processing all road segments on Earth and render map tile images that display segments
MapReduce Execution

Fine granularity tasks: many more map tasks than machines

Bucket sort to get same keys together

2000 servers => ≈ 200,000 Map Tasks, ≈ 5,000 Reduce tasks
Word Count Example

In: set of words
Out: set of (word, count) pairs

Algorithm:
1. For each in word W, emit (W, 1) as a key-value pair (map step).
2. Group together all key-value pairs with the same key (reduce step).
3. Perform a sum reduction on all values with the same key (reduce step).
   - All map operations in step 1 can execute in parallel with only local data accesses
   - Step 2 may involve a major reshuffle of data as all key-value pairs with the same key are grouped together.
   - Step 3 performs a standard reduction algorithm for all values with the same key, and in parallel for different keys.
Pseudocode for Word Count

1. `<String, Integer>` map(String inKey, String inValue):
   2. // inKey: document name
   3. // inValue: document contents
   4. for each word w in inValue:
   5.     emitIntermediate(w, 1) // Produce count of words
   6.
7. `<Integer>` reduce(String outKey, Iterator<Integer> values):
   8. // outKey: a word
   9. // values: a list of counts
10. Integer result = 0
11. for each v in values:
12.     result += v // the value from map was an integer
13. emit(result)
Example Execution of Word Count Program

Distribute

Map 1
that that is
is 1, that 2

Map 2
is that that
is 1, that 2

Map 3
is not is not
is 2, not 2

Map 4
is that it it is
is 2, it 2, that 1

Shuffle

is 1,1,2,2
it 2
Reduce 1
is 6; it 2

that 2,2,1
not 2
Reduce 2
not 2; that 5

Collect

is 6; it 2; not 2; that 5
Announcements & Reminders

• IMPORTANT:
  — Watch video & read handout for topic 2.5 and 2.6 for Friday’s lecture

• HW1 is due TODAY by 11:59pm

• HW2 will be available today and is due by Wednesday, Feb 12th

• Lab2 is tomorrow at 1pm and 4pm

• Quiz for Unit 1 (topics 1.1 - 1.5) is due Friday by 11:59pm

• See Office Hours link on course web site for latest office hours schedule.