IS 622 Homework Week 2

Ben Arancibia

September 2, 2015

Exercise 2.3.1: Design MapReduce algorithms to take a very large file of integers and produce as output:

(a) The largest integer. input <- file while(length(line <- readLines(input, n=1, warn=FALSE)) > 0) { for(i=0; ilargest) { console.log(largest + " " + array[i]); largest = array[i]; } }

What occurs above is that the map task process a chunk of the input file. Then the map task produces an integer of the largest value in that chunk as a key, value pair. Grouping by key identified duplicates. After this process the results are fed into a single reduce task that produces (integer, 1) of the largest value.

(b) The average of all the integers.

map: (key, value) num = value[2] emit("1", integer)

reduce: (key, value) result = sum(value) / n emit("1", integer)

Map task to produce(key, value) Single reduce task to sum all integers, sum num of integers from Reduce tasks, calculate average; emit

(c) The same set of integers, but with each integer appearing only once.

Map task emit (integer,1) once only for each unique inteer. Maintain an array to track whether integer has been emitted. Shuffle step groups all values for the same integer: (integer,1,1,1,1,1). Reduce task eliminates duplicates for each integer key and emit integer.

(d) The count of the number of distinct integers in the input. Two stages needed, since very large input file First phase: eliminate duplicates in large input file Map task 1: just emit (integer, 1) for unique integers Sort by key, so disjoint set or range of integers goes to each reducer Reduce task 1: Eliminates duplicates, counts unique integers, emits count of unique integers for its range of inputs (count, count)

Second phase: count unique numbers overall Map task 2: Each map task will get some number of counts as input from the previous Reduce phase Adds them together and emits (count, count) Sort step: only one Reducer task, so all (count, count) pairs sent there Reduce task 2: single Reducer, sums all counts from map tasks and produces overall count (could be multiple values for a key)

Exercise 2.3.2 : Our formulation of matrix-vector multiplication assumed that the matrix M was square. Generalize the algorithm to the case where M is an r-by-c matrix for some number of rows r and columns c.

```
Matrix-Multiply(r, c)
  if columns[r] != columns[c]
    then error "incompatible dimensions"
    else for i <- 1 to rows[r]
      do for j <- 1 to columns[c]
        do C[i,j] <- 0
          for k <- 1 to columns[r]
            do C[i,j] <- C[i,j] + r[i,k] * c[k,j]
      return C
```