Assignment 1: Median Redux
Performance comparison of different implementations of MapReduce programs

Pramod Khare

1st February 2015

1 Introduction

There are four implementations of Median Redux:
(v1) Java sequential without threading/concurrent programming,
(v2) Map Reduce (sort in Reduce),
(v3) Map Reduce (using composite keys to let MR do the sort),
(v4) Map Reduce (either form) where each call to map also computes Fibbonacci of N.

2 Note

Performance comparison of multiple MedianRedux implementations is done on following machine hardware Configurations, (Virtual Machine) OS Ubuntu 14.04 LTS, RAM-2.8GB, Processor Core 2 each of Intel Core i3 2.20GHz, No Graphics Processor.

3 Answer Q1

Here are total run-time results from best runs for each version in standalone and pseudo distributed mode:

(time in sec)	V1	V2	V3	V4 (for N <= 10)
Standalone	131.764	272.473	398.346	376.483
Pseudo-distributed	-	319.232	387.790	338.019

From multiple runs of programs v1, v2, v3, it seems clear that all three implementations take approximately similar amount of time. Machine configurations badly affected the total-run-time because of total available physical memory and CPU.

4 Answer Q2

From multiple runs of V4 program, I found out that for larger values of N i.e. more than 10 the run-time of v4 increases significantly.
If we compare v2—v3 with v4, it's pretty clear that for smaller values of N (<= 10), the time taken by mapper function to compute Fibonacci series is small but negligible and total-run-times are almost the similar.

5 Answer Q3

I found out that the number of reducers is same for all (v2, v3, v4) versions on pseudo-distributed mode i.e. 13.
But for standalone mode, its 51 for all v2, v3, v4 versions.

6 Conclusion

From above comparisons its evident that machine configurations does play a major role in affecting MapReduce performance in pseudo and standalone mode.