TeraByte Sort on Apache Hadoop

Owen O'Malley
Yahoo!
owen@yahoo-inc.com

May 2008

Apache Hadoop is a open source software framework that dramatically simplifies writing distributed data intensive applications. It provides a distributed file system, which is modelled after the Google File System[2], and a map/reduce[1] implementation that manages distributed computation. Since the primary primitive of map/reduce is a distributed sort, most of the custom code is glue to get the desired behavior.

I wrote 3 Hadoop applications to run the terabyte sort:

1. **TeraGen** is a map/reduce program to generate the data.
2. **TeraSort** samples the input data and uses map/reduce to sort the data into a total order.
3. **TeraValidate** is a map/reduce program that validates the output is sorted.

The total is around 1000 lines of Java code, which will be checked in to the Hadoop example directory.

TeraGen generates output data that is byte for byte equivalent to the C version including the newlines and specific keys. It divides the desired number of rows by the desired number of tasks and assigns ranges of rows to each map. The map jumps the random number generator to the correct value for the first row and generates the following rows. For the final run, I configured TeraGen to use 1800 tasks to generate a total of 10 billion rows in HDFS, with a block size of 512MB.

TeraSort is a standard map/reduce sort, except for a custom partitioner that uses a sorted list of $N-1$ sampled keys that define the key range for each reduce. In particular, all keys such that $sample[i-1] <= key < sample[i]$ are sent to reduce i. This guarantees that the output of reduce i are all less than the output of reduce $i+1$. To speed up the partitioning, the partitioner builds a two level trie that quickly indexes into the list of sample keys based on the first two bytes of the key. TeraSort generates the sample keys by sampling the input before the job is submitted and writing the list of keys into HDFS. I wrote an input and output format, which are used by all 3 applications, that read and

write the text files in the right format. The output of the reduce has replication set to 1, instead of the default 3, because the contest does not require the output data be replicated on to multiple nodes. I configured the job with 1800 maps and 1800 reduces and *io.sort.mb*, *io.sort.factor*, *fs.inmemory.size.mb*, and task heap size sufficient that transient data was never spilled to disk other at the end of the map. The sampler used 100,000 keys to determine the reduce boundaries, although as can be seen in figure 2, the distribution between reduces was hardly perfect and would benefit from more samples.

TeraValidate ensures that the output is globally sorted. It creates one map per a file in the output directory and each map ensures that each key is less than or equal to the previous one. The map also generates records with the first and last keys of the file and the reduce ensures that the first key of file i is greater that the last key of file $i - 1$. Any problems are reported as output of the reduce with the keys that are out of order.

The cluster I ran on was:

- 910 nodes
- 4 dual core Xeons @ 2.0ghz per a node
- 4 SATA disks per a node
- 8G RAM per a node
- 1 gigabit ethernet on each node
- 40 nodes per a rack
- 8 gigabit ethernet uplinks from each rack to the core
- Red Hat Enterprise Linux Server Release 5.1 (kernel 2.6.18)
- Sun Java JDK 1.6.0_05-b13

The sort completed in **209 seconds (3.48 minutes)**. I ran Hadoop trunk (pre-0.18.0) with patches for HADOOP-3443 and HADOOP-3446 which were required to remove intermediate writes to disk. Although I had the 910 nodes mostly to myself, the network core was shared with another active 2000 node cluster, so the times varied a lot depending on the other activity.

References

[1] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, December 2004.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *19th Symposium on Operating Systems Principles*, Lake George, NY, October 2003. ACM.

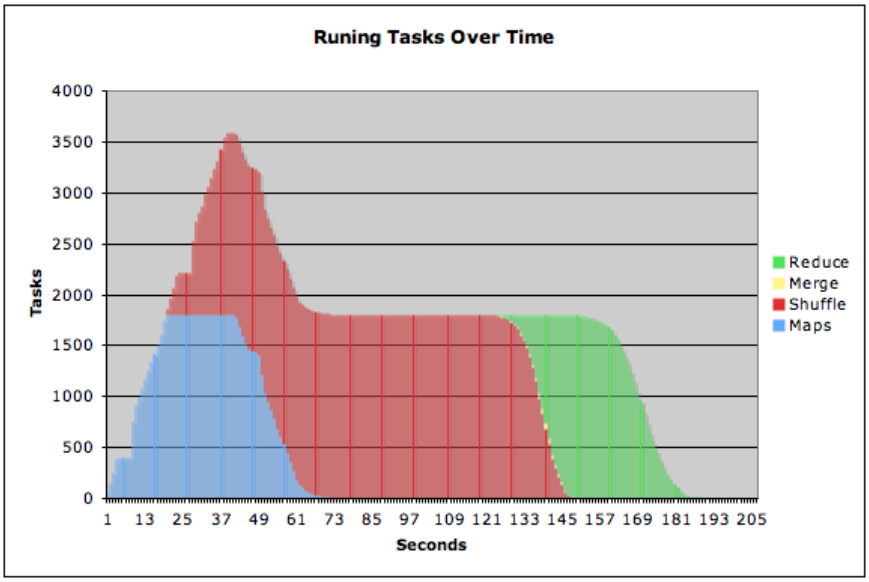

Figure 1: Number of tasks in each phase across time

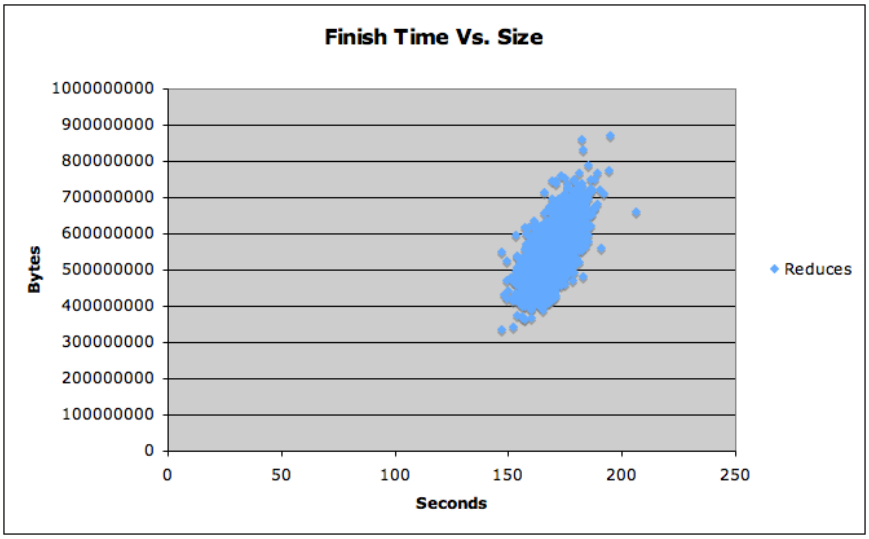

Figure 2: Plot of reduce output size versus finish time