Understanding Filesystem Imbalance in Hadoop

Andrew D. Ferguson
Brown University

Rodrigo Fonseca
Brown University

The Hadoop platform for MapReduce [1] is an increasingly popular method for executing distributed computations, driven by free availability, an adaptable model, and support for very large data sets. In order to support such data sets efficiently, Hadoop executes most computations near the data, rather than transferring the data over the network. As a result, Hadoop's performance is directly affected by the distribution of data in the Hadoop Distributed Filesystem (HDFS).

In this work, we investigate the placement of blocks in HDFS and show that it exhibits surprising non-uniformity. When blocks are placed non-uniformly in the distributed filesystem, network transfers must occur during job execution in order to bring input data to available computational cores. Because cross-rack network bandwidth is one of the most limited resources in the cluster, these unnecessary transfers can degrade performance.

The locations of file blocks read by a MapReduce job are collectively called the *input split*. In order to achieve best performance, the input split should intuitively consist of an equal number of file blocks on each node in the cluster. We show that under Hadoop's default block placement strategy, the number of blocks on each node in the cluster is instead binomially distributed.

In order to visualize the existing file placement strategy and its effect on task performance, we have developed a real-time "heatmap" which illustrates how "hot" or "cold" each host in the cluster is. A node is considered "hot" if it is carrying at least one standard deviation above the expected number of input splits. A node is "cold" if it supports less than one standard deviation below expectation. This heatmap is prepared both on a job-by-job level, and across the complete workload. By watching the heatmap during job execution, its possible to observe the block imbalance directly.

In order to further characterize the placement of blocks in HDFS, we have analyzed 93 MapReduce jobs run at a large internet company. These tasks were each run on a cluster with around 1400 nodes. For large jobs (those with more than 1000 tasks), the well-known normal approximation to the binomial distribution applies (see Figure 1). We are currently investigating the hypothesis that tasks on nodes at both ends of the featured histogram become stragglers because they are competing for network or disk bandwidth. Reducing the performance effect of this data placement imbalance may also mitigate the need for speculative task execution.

Further analysis of these jobs indicate that non-local tasks are quite prevalent. Of the 41,377 tasks in the sample, 13,299 (32.14%) read data from the local rack, and 2,938 (7.1%) read data from another rack. For small jobs, the percentage of non-local tasks is higher than average, because it is unlikely for a task slot to be available on a node which also contains data for that small job.

Ultimately, we are interested in dynamically rebalancing the input data for submitted jobs in order to minimize contention for the disks and network. As a first step, we consider alternate algorithms for static block placement such as round-robin. We evaluate the performance change by comparing the effect on the runtime of a representative benchmark suite.

References

[1] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters, *Commun. ACM*, 51(1):107-113, 2008.

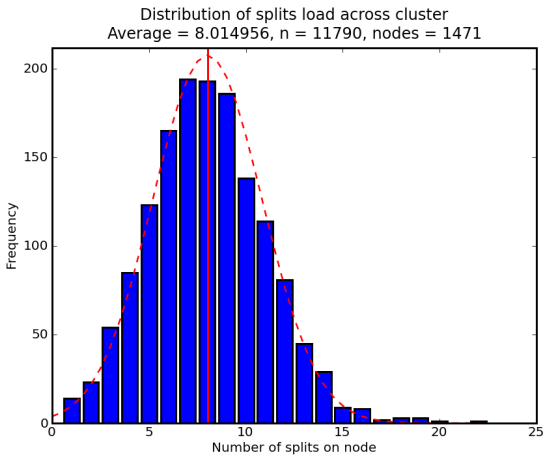

Figure 1: Distribution of blocks for a job with 11,790 tasks run on 1471 hosts. The curve plots the normal approximation to a binomial distribution with n = 11,790 and p = 1/1471.