Understanding Filesystem Imbalance in Hadoop

Andrew Ferguson
adf@cs.brown.edu

Rodrigo Fonseca
rfonseca@cs.brown.edu

Hadoop Architecture

- Users submit jobs to Hadoop
- Jobs consist of map and reduce tasks executed by TaskTrackers
- Each map task processes one chunk from Hadoop distributed filesystem
- Chunk locations are known as the input split, which is computed in advance
- Tasks try to read the closest chunk, stored locally, rack-locally, or remotely

Hadoop 0.20.1 was augmented with round-robin placement in addition to uniform-at-random (standard placement)

Users submit jobs to Hadoop

Jobs consist of map and reduce tasks executed by TaskTrackers

Each map task processes one chunk from Hadoop distributed filesystem

Chunk locations are known as the input split, which is computed in advance

Tasks try to read the closest chunk, stored locally, rack-locally, or remotely

Typical Deployment

- Drives are placed 2-4 per node
- Nodes are organized into racks with full bandwidth
- Racks are connected at 1:5 or 1:8 bandwidth

HDFS places chunks uniformly at random in the cluster

The number of chunks on each node is the sum of i.i.d. Bernoulli random variables, which is binomially distributed

When a block is not available locally, it must be read over a (relatively) slow network link, and compete for resources

Filesystem Imbalance in HDFS

- HDFS places chunks uniformly at random in the cluster
- The number of chunks on each node is the sum of i.i.d. Bernoulli random variables, which is binomially distributed
- When a block is not available locally, it must be read over a (relatively) slow network link, and compete for resources

Imbalance in the Real World

- Analyzed 93 jobs from a large company of varying sizes (34 tasks to 11,340 tasks) with a total of 41,377 tasks in total
- 13,299 tasks (32.14%) had input data local to the rack; 2,938 (7.1%) fetched data from another rack; the rest had local data

This problem is worse for small jobs:

<table>
<thead>
<tr>
<th>Job Size</th>
<th>Small</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Tasks</td>
<td>181</td>
<td>2006</td>
</tr>
<tr>
<td>Local Tasks</td>
<td>22 (12.19%)</td>
<td>2099 (71.49%)</td>
</tr>
<tr>
<td>Rack-Local Tasks</td>
<td>111 (61.33%)</td>
<td>700 (23.84%)</td>
</tr>
<tr>
<td>Remote Tasks</td>
<td>48 (26.52%)</td>
<td>137 (4.67%)</td>
</tr>
</tbody>
</table>

- Observed input split distributions match predictions:

\[ E(n) = \frac{N \times S}{N + S} \]

- Hypothesis:

Round-robin placement will decrease the variance of the splits distribution and yield improved performance.

Evaluating Round-Robin Placement

- Hadoop ver. 0.20.1 was augmented with round-robin placement in addition to uniform-at-random (standard placement)
- Experiments were run on a cluster of 63 nodes (21 per rack) with two additional nodes as masters. Nodes had 4 x 2.4 GHz CPUs, 3 GB of RAM, and gigabit Ethernet connections

10 GB Grep Benchmark (I/O Intensive)

10 GB Grep Benchmark (I/O Intensive)

Conclusions and Next Steps

- The performance improvements from round-robin placement illustrate the benefits of a more-balanced filesystem.
- In future work, we will examine whether round-robin-like block placement can improve the performance of the new "delay-scheduling" technique, and construct a characterization of the theoretically best read pattern for a given input split.