Motivation

Large tables with 1 million or more regions soon will become a reality at Yahoo. To this end, we have begun working on scaling HBase to meet these requirements. We refer to these 1 million+ region tables as 'humongous' tables and are working to add formal support for them. Our first step is to make it fast to create these humongous tables.

New Feature: Humongous Tables and Hierarchical/Bucketed Regions

One bottleneck we found with creating humongous tables is that as the number of region directories in a table directory increases, the amount of time the namenode takes to create more region directories increases dramatically. Simply put, HDFS was not designed to handle the existence of millions of files/directories in a single directory. To work around this issue, we have augmented the HBase filesystem hierarchy to include the notion of a "bucket" directory. This bucket directory will be situated between the `/table` directory and the `/region` directory. In other words, the new structure is `/table/bucket/region/columnfamily`. Bucket directory names are 4 hex characters in length. There will be up to 16^4 = 65,536 buckets (and thus directories below `/table`), and they are lazily created with the 1st region that can be assigned to that bucket. To create tables with this new structure, users must set a 'humongous' flag in the table descriptor.

Experiments

We now provide some experimental data on the performance of "humongous" tables. The following experiments were performed on a cluster in Yahoo with 175 nodes. Each region server has 16 logical processors (2 physical cpus and 8 physical cores total), about 24 GB memory, and 6x 2TB hard drives. The master server has 32 logical processors (2 physical cpus and 16 physical cores total), about 400 GB memory, and 2x 500 GB hard drives. The namenode server is about the same as a region server, but with 50 GB total memory instead.

Experiment 1: Create 1 million regions test:

Setup: 15GB heap, zk-assignment, bucket_size=4

bucket size	2 (256 buckets)	3 (4k buckets)	4 (64k buckets)
normal table	*20 mins 49 secs*	*20 mins*	*20 mins 35 secs*
humongous table	*18 mins 25 secs*	*15 mins 48 secs*	*15 mins 24 secs*

As you can see from the above table, there are diminishing gains with increasing the number of buckets for a table. With 1 million regions, 4-character buckets do not yield better performance than 3-character buckets. However, Yahoo's scale continues to grow and 4-character buckets may be one day necessary for 10 million+ region tables. Hence we have chosen 4-character bucket names, as ~64k bucket directories is still manageable by HDFS.

Experiment 2: Create 5 million regions test:

The current zk-assignment scheme requires a huge amount of memory on HMaster during region assignment. With a 40GB heap, the HMaster will die due to OOM. The zk-less assignment scheme performs better than zk-assignment on a 5M region table so we use it in this experiment, however, we still saw frequent GC pauses which might crash the master.

Setup: 40GB heap, zkless-assignment, 100 init threads

normal table	*4 hours 23 minutes*
humongous table	*1 hour 27 minutes*

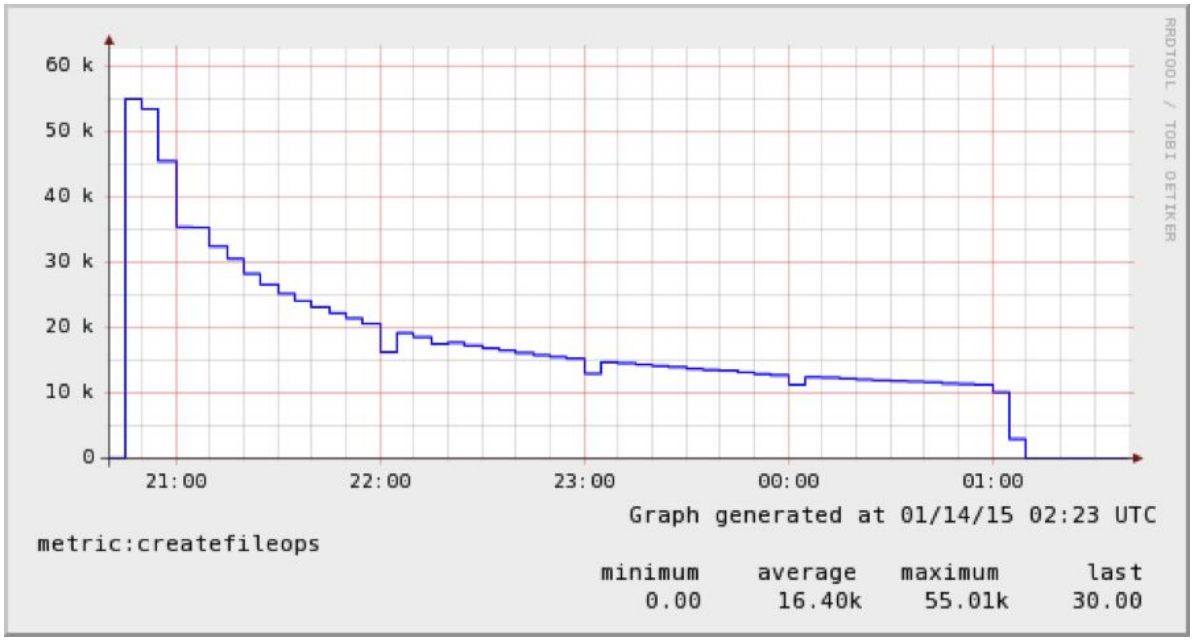

Figure 1: namenode create file ops during region init for 5M normal table

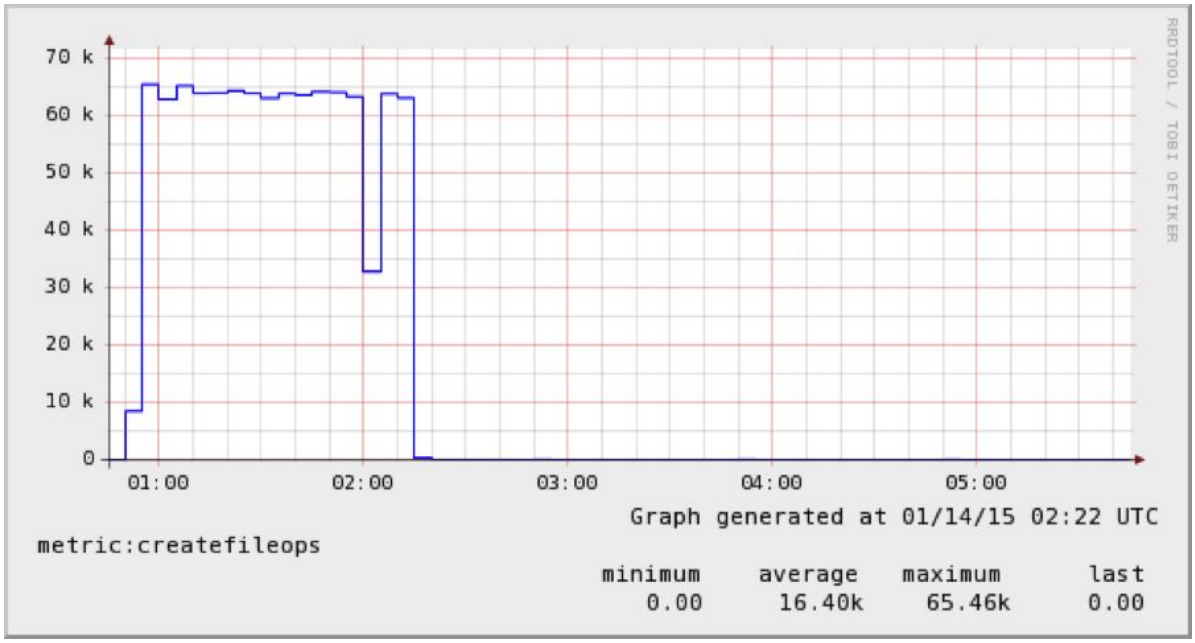

Figure 2: namenode create file ops during region init for 5M humongous table

As you can see from the above graphs, namenode operation throughput is relatively stable for a humongous table from the 1st region created to the last, whereas with a normal table, performance degrades gradually until it is almost 5x worse than when it started.

Experiment 3: Create 10 million regions test:

Setup: 40GB heap, zkless-assignment, 100 init threads

The normal table doesn't work at 10 million region scale due to the maximum number of items in a directory being exceeded (a hard limitation having to do with HDFS protobufs). The humongous table takes 2 hours and 53 minutes to finish initializing region files. Below is the throughput graph.

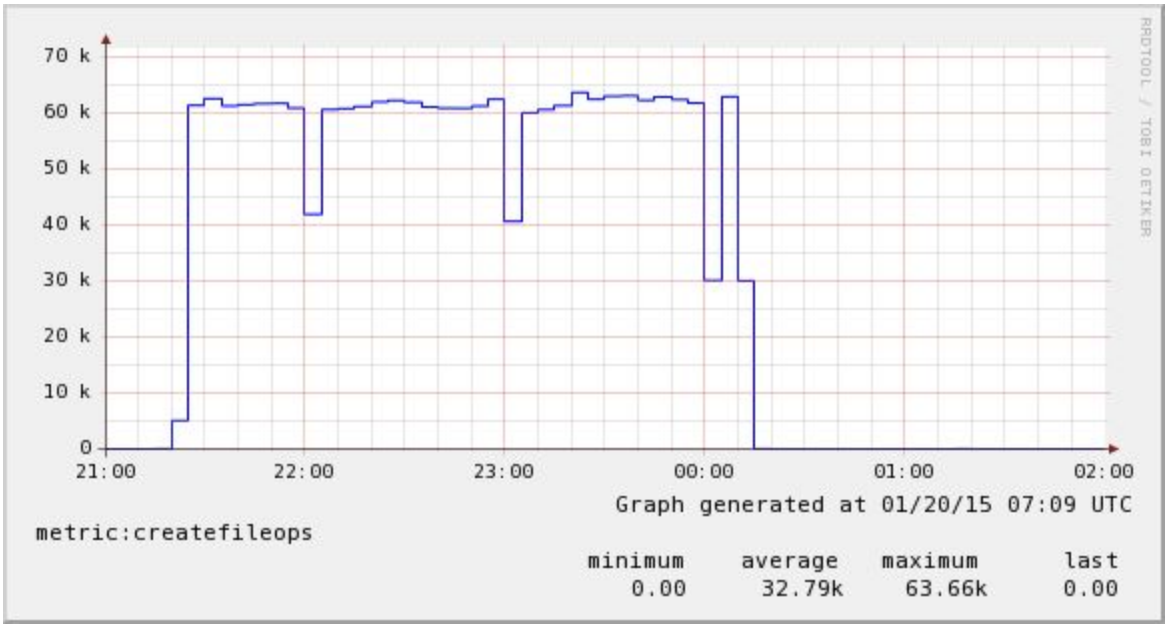

Figure 3: namenode create file ops during region init for 10M humongous table