LargeDataSetConsiderations

Using Cassandra for large data sets (lots of data per node)

This page aims to to give some advice as to the issues one may need to consider when using Cassandra for large data sets (meaning hundreds of gigabytes or terabytes per node). The intent is not to make original claims, but to collect in one place some issues that are operationally relevant. Other parts of the wiki are highly recommended in order to fully understand the issues involved.

This is a work in progress. If you find information out of date (e.g., a JIRA ticket referenced has been resolved but this document has not been updated), please help by editing or e-mailing cassandra-user.

Note that not all of these issues are specific to Cassandra. For example, any storage system is subject to the trade-offs of cache sizes relative to active set size, and IOPS will always be strongly correlated with the percentage of requests that penetrate caching layers. Also of note, the more data stored per node, the more data will have to be streamed in bootstrapping new or replacement nodes.

Assumes Cassandra 1.2+

Significant work has been done to allow for more data to be stored on each node:

- Row cache can be serialized off-heap. Keep in mind that this still stores entire rows off-heap and when they are used the full row is temporarily deserialized in the heap. Alternate row cache implementations are being worked on to make the row cache more generally useful, see CASSANDRA-2864.
- Bloom filters:
  - Moved off-heap CASSANDRA-4865
  - Tunable via bloom_filter_fp_chance. Starting in Cassandra 1.2 there are better defaults: 0.01 for column families using the SizeTieredCompactionStrategy and 0.1 for column families using the LeveledCompactionStrategy. Note that a change to this property will take effect as new sstables are built.
- Compression metadata has been moved off-heap in 1.2 (reference)
- Partition summary has been reduced in 1.2.5 and moved off-heap in 2.0
- Key cache has been serialized off-heap in 2.0 (reference)
- Parallel compactions as in CASSANDRA-2191 and CASSANDRA-4310
- Multi-threaded compaction for high IO hardware (reference)
- Virtual nodes to increase the parallelism and reduce the time of bootstrapping new nodes
- sstable index files are no longer loaded on startup (reference)

On moving data structures off-heap

- Moving data structures off-heap means that the structure gets serialized off-heap until it is needed. Then it is deserialized temporarily in the heap and is garbage collected when it is no longer needed.

Other points to consider:

- Disk space usage in Cassandra can vary over time:
  - Compaction: with the SizeTieredCompactionStrategy, compaction can up to double the disk space used. The LeveledCompactionStrategy usually only requires about 10% overhead (see http://www.datastax.com/dev/blog/leveled-compaction-in-apache-cassandra).
  - Repair: repair operations can increase disk space demands, see http://www.datastax.com/dev/blog/advanced-repair-techniques for details and how it can be improved.
- Consider the choice of file system. Removal of large files is notoriously slow and seek bound on e.g. ext2/ext3. Consider xfs or ext4fs. This affects background unlink():ing of sstables that happens every now and then, and also affects start-up time (if there are sstables pending removal when a node is starting up, they are removed as part of the start-up procees; it may thus be detrimental if removing a terabyte of sstables takes an hour (numbers are ballparks, not accurately measured and depends on circumstances)).
- Adding nodes is a slow process if each node is responsible for a large amount of data. Plan for this; do not try to throw additional hardware at a cluster at the last minute.
- The operating system's page cache is affected by compaction and repair operations. If you are relying on the page cache to keep the active set in memory, you may see significant degradation on performance as a result of compaction and repair operations. See the cassandra.yaml for settings to reduce this impact.
- The partition (or sampled) index entries for each sstable can start to add up. You can reduce the memory usage by tuning the interval that it samples at. The setting is index_interval in cassandra.yaml. See the comments there for more information.

Other references to improvements:

- Performance improvements in Cassandra 1.2
- Six mid-series changes in Cassandra 1.2