Streaming

See Streaming2 for Apache Cassandra 2.0 and above.

There are two main instances of streaming (post 0.7):

- Transfer - Occurs when a Source pushes SSTables for certain ranges to a Destination. Initiated and controlled by the Source.
- Request - Occurs when a Destination requests a set of ranges from a Source. Initiated and controlled by the Destination.

Transfer

The following steps occur for Stream Transfers.

1. Source has a list of ranges it must transfer to another node.
2. Source copies the data in those ranges to sstable files in preparation for streaming. This is called anti-compaction (because compaction merges multiple sstable files into one, and this does the opposite).
3. Source builds a list of PendingFile's which contains information on each sstable to be transfered.
4. Source starts streaming the first file from the list, followed by the log "Waiting for transfer to $some_node to complete". The header for the stream contains information on the streamed file for the Destination to interpret what to do with the incoming stream.
5. Destination receives the file writes it to disk and sends a FileStatus.
6. On successful transfer the Source streams the next file until its done, on error it re-streams the same file.

Request

1. Destination compiles a list of ranges it needs from another node.
2. Destination sends a Stream{{`Request}}`Message to the Source node with the list of ranges.
3. Source prepares the SSTables for those ranges and creates the PendingFile's.
4. Source starts streaming the first file in the list. The header for the first stream contains info of the current stream and a list of the remaining PendingFile's that fall in the requested ranges.
5. Destination receives the stream and writes it to disk, followed by the log message "Streaming added org.apache.cassandra.io.sstable. SSTableReader(path='/var/lib/cassandra/data/Keyspace1/Standard1-e-1-Data.db')".
6. Destination then takes the lead and requests the remaining files one at a time. If an error occurs it re-requests the same file, if not continues with the next file until done.
7. Source streams each of the requested files. The files are already anti-compacted, so it just streams them to the Destination.

Transfer (0.6 and below)

1. Source starts with STREAM_INITIATE (Prepares the request ranges and sends a list of pending files)
2. Destination acknowledges with STREAM_INITIATE_DONE (Adds to list of pending files per node)
3. Source starts streaming the first file from the list of files it has prepared for that Destination node.
4. Destination receives the file returns a Stream_Status. Here the order of the files is maintained and only one transfer from a node can happen.
5. Source based on status, restreams file or streams the next file until complete.

Request (0.6 and below)

1. Destination invokes STREAM_REQUEST (Compiles a set of ranges that it needs from the source)
2. From this point on the steps in Transfer (0.6) are followed.

Note: Order is very important in 0.6 streaming and the source can only transfer one file at a time, and the destination can only receive one set of transfers from a source at any instant. Multiple streams would break the process.

Streaming Invocations

Streaming in Cassandra between nodes is invoked in the following contexts:

1. **Bootstrapping** - During bootstrap the node requests ranges from other nodes. It invokes *Stream Request*.
2. **Repair** - The Anti-Entropy service performs repairs by comparing Merkle trees and the final step in the process is to transfer conflicting ranges to other nodes and request conflicting changes from other nodes in that order. So both *Request* and *Transfer* are invoked here.
3. **Restore Replica** - StorageService performs a *Stream Request*.
4. **Un-bootstrap** - During node decommission or move, un-bootstrap is invoked to transfer the nodes ranges to other nodes. *Transfer* is invoked in this case.

Monitoring

Anti-compaction, both during request and transfer takes the most amount of time. It can be monitored using the `org.apache.cassandra.db. CompactionManager` mbean on the Source.

Monitoring the status of streaming on both Source and Destination nodes can be found under the `org.apache.cassandra.streaming.StreamingService` MBean. The `Status` attribute gives an easy indication of what a node is doing with respect to streaming. The operations `getOutgoingFiles(host)` and `getIncomingFiles(host)` each return a list of strings describing the status of individual files being streamed to and from a given host. Each string follows this format: `[path to file] [bytes sent/received]/[file size]` If you think that streaming is taking too long on your cluster, the first thing you should do is check `StreamSources` or `StreamDestinations` to figure out which hosts are streaming files. Use those hosts as inputs to `getOutgoingFiles()` or `getIncomingFiles()` to check on the status of individual files from the problematic source and destination nodes. Streaming is conducted in 32MB chunks, so you should refresh the file status after a few seconds to see if the sent/received values change. If they do not change, or change more slowly than you'd like, something is wrong.

The streaming status can also be monitored using `nodetool -h <hostname/IP> -p <jmxport> netstats`

`https://c.statcounter.com/9397521/0/fe557aad/1/|stats`