This document is a capture of content from the "How Region Splits are implemented" section of http://hortonworks.com/blog/apache-hbase-region-splitting-and-merging/ on November 3, 2014, by Andrew Purtell.

How HBase Region Splits are Implemented

As write requests are handled by the region server, they accumulate in an in-memory storage system called the "memstore". Once the memstore fills, its content are written to disk as additional store files. This event is called a "memstore flush". As store files accumulate, the RegionServer will "compact" them into combined, larger files. After each flush or compaction finishes, a region split request is enqueued if the region split policy decides that the region should be split into two. Since all data files in HBase are immutable, when a split happens, the newly created daughter regions will not rewrite all the data into new files. Instead, they will create small symlink like files, named Reference files, which point to either top or bottom part of the parent store file according to the split point. The reference file will be used just like a regular data file, but only half of the records. The region can only be split if there are no more references to the immutable data files of the parent region. Those reference files are cleaned gradually by compactions, so that the region will stop referring to its parents files, and can be split further.

Although splitting the region is a local decision made at the RegionServer, the split process itself must coordinate with many actors. The RegionServer notifies the Master before and after the split, updates the .META. table so that clients can discover the new daughter regions, and rearranges the directory structure and data files in HDFS. Split is a multi task process. To enable rollback in case of an error, the RegionServer keeps an in-memory journal about the execution state. The steps taken by the RegionServer to execute the split are illustrated by Figure 1. Each step is labeled with its step number. Actions from RegionServers or Master are shown in red, while actions from the clients are show in green.

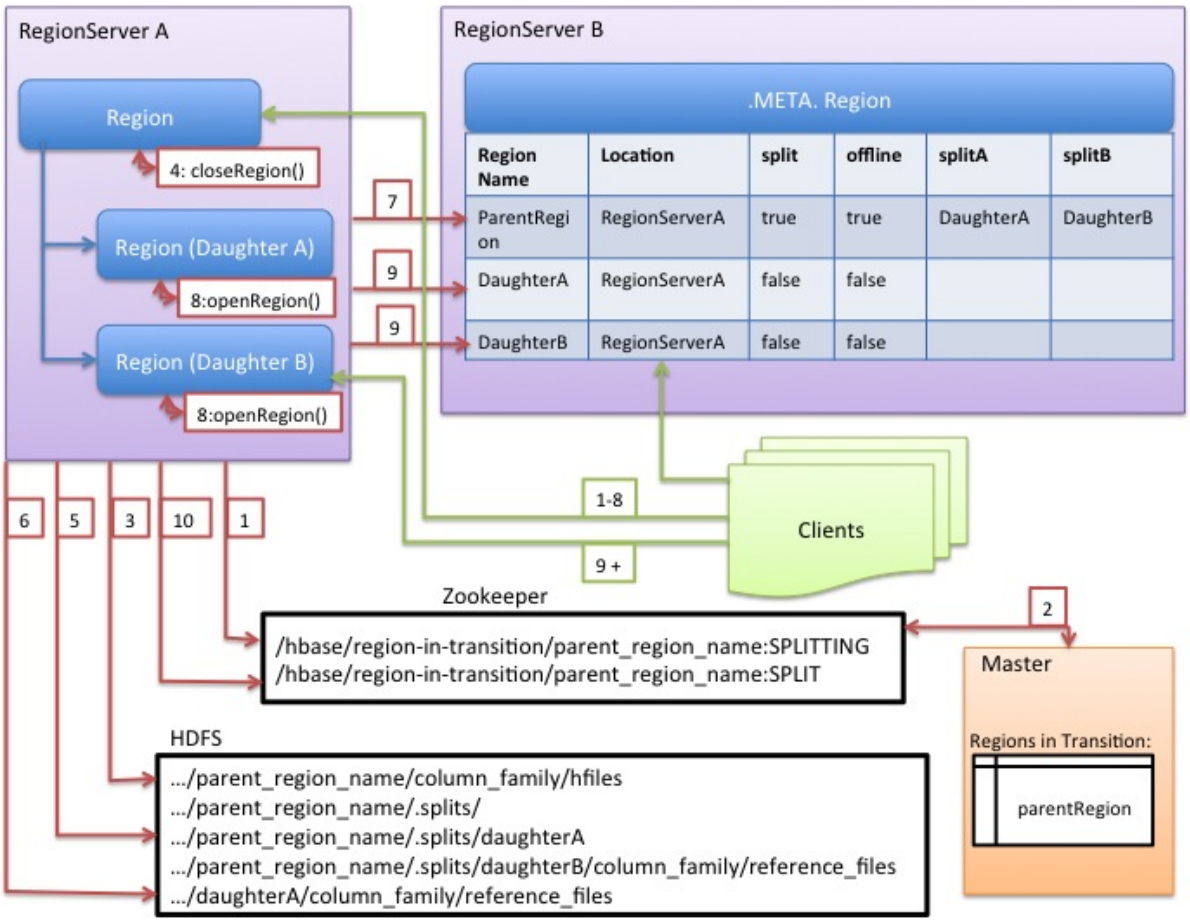

Figure 1

1. RegionServer decides locally to split the region, and prepares the split. **THE SPLIT TRANSACTION IS STARTED.** As a first step, it acquires a shared read lock on the table so schema modifications are prevented during the splitting process. Then it creates a znode in zookeeper under /hbase/region-in-transition/region-name in SPLITTING state.

2. The Master learns about this znode, since it has a watcher for the parent region-in-transition znode.

3. RegionServer creates a sub-directory named ".splits" under the parent's region directory in HDFS.

4. RegionServer closes the parent region. It marks the region as offline in its local data structures. **THE SPLITTING REGION IS NOW OFFLINE.** At this point, client requests coming to the parent region will

throw NotServingRegionException. The client will retry with some backoff. The closing region is flushed.

5. RegionServer create the region directories under .splits directory, for daughter regions A and B, and creates necessary data structures. Then it splits the store files, in the sense that it creates two Reference files per store file in the parent region. Those reference files will point to the parent regions files.

6. RegionServer creates the actual region directory in HDFS, and moves the reference files for each daughter.

7. RegionServer sends a Put request to the .META. table, and sets the parent as offline in the .META. table and adds information about daughter regions. At this point, there won't be individual entries in .META. for the daughters. Clients will see the parent region is split if they scan .META., but won't know about the daughters until they appear in .META.. Also, if this Put to .META. succeeds, the parent will be effectively split. If the RegionServer fails before this RPC succeeds, Master and the next region server opening the region will clean dirty state about the region split. After the .META. update, though, the region split will be rolled-forward by Master.

8. RegionServer opens daughters A and B in parallel.

9. RegionServer adds the daughters A and B to .META. together with information that it hosts the regions. **THE SPLIT REGIONS (DAUGHTERS WITH REFERENCES TO PARENT) ARE NOW ONLINE.** After this point, clients can discover the new regions, and issue requests to the new region. Clients cache the .META. entries locally, but when they make requests to the region server or .META., their caches will be invalidated, and they will learn about the new regions from .META..

10. RegionServer updates znode /hbase/region-in-transition/region-name in zookeeper to state SPLIT, so that the master can learn about it. The balancer can freely re-assign the daughter regions to other region servers if it chooses so. **THE SPLIT TRANSACTION IS NOW FINISHED.**

11. After the split, meta and HDFS will still contain references to the parent region. Those references will be removed when compactions in daughter regions rewrite the data files. Garbage collection

tasks in the master periodically checks whether the daughter regions still refer to parents files. If not, the parent region will be removed.