HBASE - Sharding

Bhavneet Soni

**HBASE - Sharding**

Sharding is a concept of storing huge database over a distributed file system. Sharding is simply put as partitioning of a database so that it can be stored in part at different nodes in a cluster. To understand sharding and need for sharding we need to understand how a traditional database stores data. In a traditional database data is usually stored in different tables which are joined together via relationships and keys. While it’s a very easy to pull information off of these database using SQL, it has some inherent drawbacks. For one RDBMS were designed to be used on a single machine so In the world of big data where scalability is the driving force, database files needs to be hosted over a number of nodes in a cluster which is very difficult to achieve with RDBMS. Also it requires a lot of effort to setup tables and schemas. HBASE is a database that store data in key value pairs and instead of having relationships and tables, all the data that is accessed together is stored together, with row being the key. Columns are combined in to column families based on data types. Data is grouped together based on the use of the data. Database table is split (sharded) based on the key ranges, called as region. When regions becomes too big, these are dynamically sharded again. Contiguous rows in a region are stored at one location and replicated over a distributed network using commodity hardware. One server can hosts multiple regions, regions provide for fast recovery in case of server failure. Splitting assists in load balancing and regions are moved between servers when one region becomes too big. Splitting is almost instantaneous as the split regions simply read from the original storage files and are rewritten asynchronously in to separate regions during compaction phase. HBase achieves fast random I/O. It accomplishes this magic by a combination of batching writes in memory and persisting data to disk using log-structured merge trees. As a result, all random writes are performed in memory, and when data is flushed to disk, the data is first sorted, then written sequentially with an accompanying index.

Sharding has some ups and downside associated with it. Since the shards are split by key ranges, if we have to query based on keys within the same shard it gives us a great performance boost, and also grouping the data by key atomic unit updates are done based on the keys. Data consistency is also higher with in the shard. Downside of the sharding is that it makes querying for data much harder like doing sorting or range queries for which need to have all complete range of data in memory takes a toll on the performance as you have to query all the shards. Data consistency is also lower than non sharded debase.

Resharding (joining) the data is also very difficult and expensive process, as it takes toll on the storage medium and I/O performance of the machine.

References

George, L. (2011). Chapter 1. Introduction. In L. George, *HBase: The Definitive Guide, 2nd Edition.* O'Rilley .

*HBase Data Distribution Features*. (2015, Feb 3). Retrieved from cmu.edu: https://quabase.sei.cmu.edu/mediawiki/index.php/HBase\_Data\_Distribution\_Features

Kim, J. (2014, Dec 5). *How Sharding Works*. Retrieved from medium.com: https://medium.com/@jeeyoungk/how-sharding-works-b4dec46b3f6#.n0rat4tkz

McDonald, C. (2015, June 26). *HBase and MapR-DB: Designed for Distribution, Scale, and Speed*. Retrieved from www.mapr.com: https://www.mapr.com/blog/hbase-and-mapr-db-designed-distribution-scale-and-speed#.VcKFNflVhBc