APACHE HBASE

Projected by:
LUKA CECXLADZE
BEQA CHELIDZE
Superviser : Nodar Momtsemlidze

About HBase™

- HBase is a column-oriented database management system that runs on top of HDFS. It is well suited for sparse data sets, which are common in many big data use cases. Unlike relational database systems, HBase does not support a structured query language like SQL; in fact, HBase isn't a relational data store at all. HBase applications are written in Java much like a typical MapReduce application.
- An HBase system comprises a set of tables. Each table contains rows and columns, much like a traditional database. Each table must have an element defined as a Primary Key, and all access attempts to HBase tables must use this Primary Key. An HBase column represents an attribute of an object; for example, In fact, HBase allows for many attributes to be grouped together into what are known as column families, such that the elements of a column family are all stored together. This is different from a row-oriented relational database, where all the columns of a given row are stored together.
- Just as HDFS has a NameNode and slave nodes, and MapReduce has JobTracker and TaskTracker slaves, HBase is built on similar concepts. In HBase a master node manages the cluster and region servers store portions of the tables and perform the work on the data. In the same way HDFS has some enterprise concerns due to the availability of the NameNode .

About MapReduce

- MapReduce is the heart of Hadoop®. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce concept is fairly simple to understand for those who are familiar with clustered scale-out data processing solutions.
- For people new to this topic, it can be somewhat difficult to grasp, because it's not typically something people have been exposed to previously. If you're new to Hadoop's MapReduce jobs, don't worry: we're going to describe it in a way that gets you up to speed quickly.
- The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

About HDFS

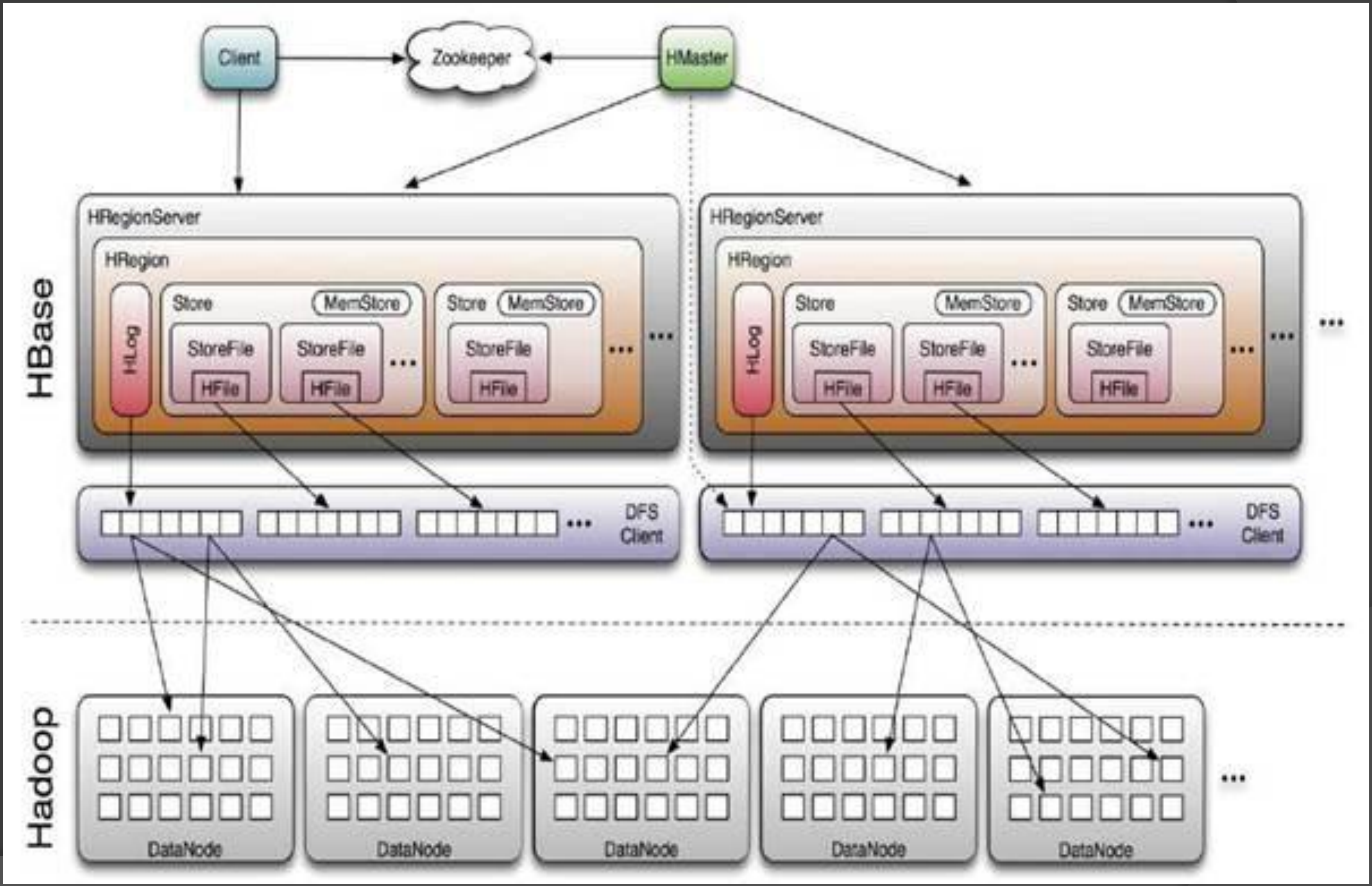
- To understand how it’s possible to scale a Hadoop® cluster to hundreds (and even thousands) of nodes, you have to start with the Hadoop Distributed File System (HDFS). Data in a Hadoop cluster is broken down into smaller pieces (called blocks) and distributed throughout the cluster. In this way, the map and reduce functions can be executed on smaller subsets of your larger data sets, and this provides the scalability that is needed for big data processing.

HBASE VS RDBMS

- RDBMS is relational database management system. Hadoop is node based flat structure.
- RDMS is generally used for OLTP processing whereas Hadoop is currently used for analytical and especially for BIG DATA processing.
- Any maintenance on storage, or data files, a downtime is needed for any available RDBMS. In standalone database systems, to add processing power such as more CPU, physical memory in non-virtualized environment, a downtime is needed for RDBMS such as DB2, Oracle, and SQL Server. However, Hadoop systems are individual independent nodes that can be added in an as needed basis.
- The database cluster uses the same data files stored in shared storage in RDBMS systems, whereas the storage data can be stored independently in each processing node.
- The performance tuning of an RDBMS can go nightmare. Even in proven environment. However, Hadoop enables hot tuning by adding extra nodes which will be self-managed.

HBASE VS RDBMS

- Hbase is a column oriented database whereas RDBMS is row oriented database, the main difference between these two is Hbase can easily scaled up with respect to storage and processing but whereas RDBMS has scalability but limited in storage capacity. when we are working with billions of rows and millions of columns then we need to choose Hbase as our database.
- RDBMS provides e.g. typed columns, secondary indexes, transactions, advanced query languages, etc.. but HBase doesn’t.
- HBase: on the other hand, is built on top of HDFS and provides fast record look-ups and updates for large tables.


Client
Zookeeper
HMaster
HBase
HRegionServer
HRegion
HLog
Store
MemStore
StoreFile
HFile
...
DFS Client
Hadoop
DataNode

About ZOOKEPEER

- ZooKeeper aims at distilling the essence of these different services into a very simple interface to a centralized coordination service. The service itself is distributed and highly reliable. Consensus, group management, and presence protocols will be implemented by the service so that the applications do not need to implement them on their own. Application specific uses of these will consist of a mixture of specific components of Zoo Keeper and application specific conventions.

HBase shell commands

- HBase shell commands are mainly categorized into 6 parts
  1) General HBase shell commands
  2) Tables Management commands
  3) Data Manipulation commands
  4) HBase surgery tools
  5) Cluster replication tools
  6) Security tools

General HBase shell commands

- hbase> status
  hbase> status ‘simple’
  hbase> status ‘summary’
  hbase> status ‘detailed’

- hbase> whoami
  hbase> version

Tables Management commands

- hbase> create ‘t1′, {NAME => ‘f1′, VERSIONS => 5}
- hbase> describe ‘t1′
- hbase> disable_all ‘t.*’
- hbase> drop_all ‘t.*’
- hbase> enable_all ‘t.*’
- hbase> list ‘abc.*’
- hbase> show_filters
- hbase> alter_async ‘t1′, NAME => ‘f1′, METHOD => ‘delete' or a shorter version:hbase> alter_async ‘t1′, ‘delete’ => ‘f1′

Data Manipulation commands

- hbase> count ‘t1′, INTERVAL => 100000
  hbase> count ‘t1′, CACHE => 1000
  hbase> count ‘t1′, INTERVAL => 10, CACHE => 1000
- hbase> delete ‘t1′, ‘r1′, ‘c1′, ts1
- hbase> incr ‘t1′, ‘r1′, ‘c1′
- hbase> put ‘t1′, ‘r1′, ‘c1′, ‘value’, ts1
- hbase>truncate ‘t1′

HBase surgery tools

- hbase> assign ‘REGION_NAME’
- hbase> balancer
- hbase> balance_switch true
- hbase> close_region ‘REGIONNAME’, ‘SERVER_NAME’
- hbase> flush ‘REGIONNAME’
- hbase> move ‘ENCODED_REGIONNAME’, ‘SERVER_NAME’
- hbase> unassign ‘REGIONNAME’, true

Cluster replication tools

- hbase> add_peer ‘2’, “zk1,zk2,zk3:2182:/hbase-prod”
- hbase> remove_peer ‘1’
- hbase> list_peers
- hbase> enable_peer ‘1’
- hbase> start_replication
- hbase> stop_replication

Security tools

- hbase> grant ‘bobsmith’, ‘RW’, ‘t1’, ‘f1’, ‘col1’
- hbase> revoke ‘bobsmith’, ‘t1’, ‘f1’, ‘col1’
- hbase> user_permission ‘table1’

PRESENTED BY :

- **BEQA CHELIDZE**
- **LUKA CECXALDZE**