sai ram

---------

Accumulo Project Report:

**Accumulo providing access control to BigData**

**Abstract:**

Over the past few decades, several trends have driven the progress of data storage and processing systems. A few applications have emerged, in order to effectively use the increasing amounts of available data. perhaps the most popular of these applications is Apache Hadoop, which allows to distribute the data over many commodity class machines and to run distributed processing jobs over the data in parallel.

Apache Accumulo is a NoSQL Database, which is a highly scalable, distributed, open source based on Google's Big Table Designs and built on the top of the Apache Hadoop, Zookeeper and the Thrift server.

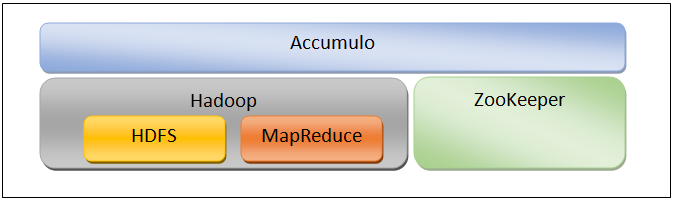
Accumulo includes the features that can be used to support a wide variety of scalable applications including storing structured or semi-structured sparse and dynamic data. Instances of Accumulo have been known to run on over a thousand servers, storing over a petabyte of data, and trillions Key-Values pairs.

In this project, a special feature of Accumulo's ability to combine the data sets of different sensitivity levels is used to create a Java GUI application, that performs fine-grained filtering over the data and allows the subscribers of varying levels to query the same data sets and see only what they are authorized to view.

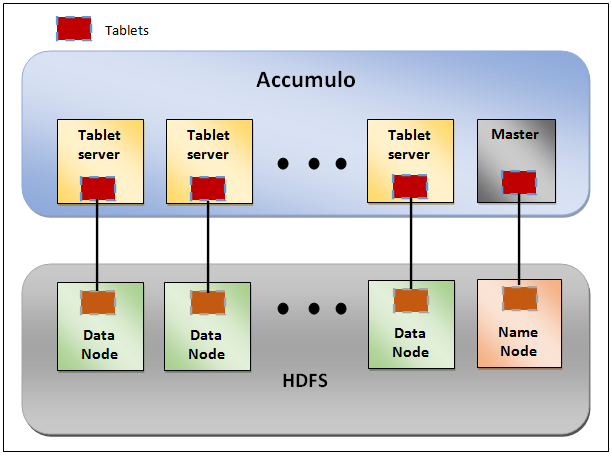
**1.Introduction:**

Apache Accumulo is a highly scalable structured store based on Google's BigTable design. Accumulo operates over Hadoop Distributed File system and can store up to trillions of data elements and keep them organized so that users can perform fast lookups. Accumulo supports efficient storage and retrieval of structured data, including queries for ranges, and support for using Accumulo tables as input and output for Map Reduce jobs.

Accumulo depends on Apache Hadoop for storage and Apache Zookeeper for configuration.



**1.1 Architecture of Accumulo**:

An instance of Accumulo includes many Tablet Servers, one Garbage Collector Process, one Master Server and many Clients. 

1. Tablet Server:

An Accumulo instance typically runs one tablet server per HDFS Data Node. Accumulo tables are split into contiguous ranges called tablets. Each tablet is assigned to a tablet server that is responsible for all reads and writes for the tablet.

Tablet servers also perform recovery of a tablet that was previously on a server that failed, reapplying any writes found in the Write-ahead log to the tablet.

1. Garbage Collector:

The Garbage Collector is a process that deletes files from HDFS when they are no longer used by Accumulo. This is a complex operation because a file may be used by more than one tablet. In order to overcome this issue, there is a section of the metadata table that records for deletion. If a table doesn't need a file any more, it writes a deletion entry for that file to the appropriate section of the metadata table.

the garbage collector reads the deletion section of the metadata table to identify required deletion.

1. Master:

The Accumulo Master's main function is to monitor the status of the tablet servers and tablets, and is responsible for detecting and responding to tablet server failure. Master also tries to balance the load across Tablet Server by assigning tablets carefully and instructing Tablet servers to unload tablets when necessary. The Master also ensures all tablets are assigned to one Tablet Server each, and handles table creation, alteration, and deletion requests from clients. The Master also coordinates startup, graceful shutdown and recovery of changes in write-ahead logs when Tablet servers fail.

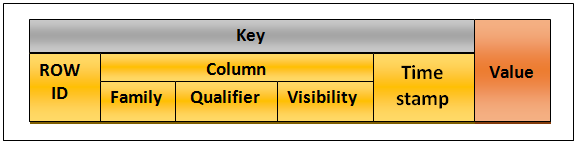
Multiple masters may be run. The masters will chosen among themselves a single master, and the others will become the backups if the master fails.

1. Client:

Accumulo includes a client library that is linked to every application. The client library contains logic for finding servers managing a particular tablet, and communicating with Tablet Severs to write and retrieve Key-Value pairs.

**1.2 Data Model of Accumulo**:

Accumulo provides a richer data model than storing simple key-value pairs on disk, keeping the keys sorted by element and lexicographically in ascending order at all times. Data is represented as key-value pairs, where the key and value are comprised of the following elements:



Accumulo key is a five tuple.

1. Row ID:

The Row ID is used to group several Key value pairs into logical row. All the key-value pairs that have the same Row ID are considered to be a part of the same row. Row IDs are simply byte arrays.

1. Column Family:

The column Family is used to group a set of Key-Value pairs together within a row. Column families can be put into groups called Locality Groups that are stored together on disk.

Unlike BigTable and HBase, Accumulo Column Families need not be declared before being used.

1. Column Qualifier:

Column Qualifiers are used to distinguish individual columns within a Column Family.

1. Column Visibility:

Column Visibility is an element that was added to the original BigTable data model by the Accumulo developers to allow Key-Value pairs to be filtered based on a set of user access tokens.

1. Timestamp:

The Timestamp functions as a kind of version number for Keys. Two Keys that are identical except for the timestamps can be considered to be two versions of the same Key.

If the timestamp is not specified by Accumulo client, the Tablet server that receives the Key-Value pair will use the time at which the data arrived as the timestamp.

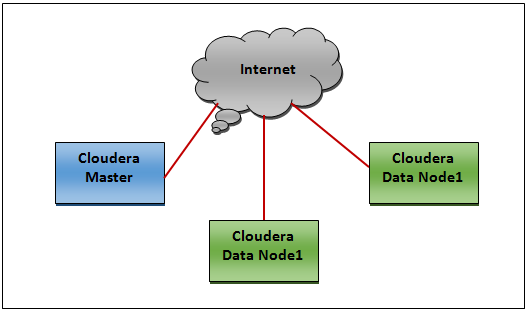
Value:

Values comprise the value part of the Key-value pair. As such, they are not sorted. Values are byte arrays that can hold a wide variety of data. In general, Values should be no bigger than a few 10s of megabytes in size.

**2.Project Infrastructure:**

The Infrastructure for the project involved a three-node Hadoop Cluster, created using the Cloudera distribution while taking the advantage of the capabilities provided by Cloudera Manager. Among the three-node Hadoop cluster, one node as Master node and other two nodes as Data nodes. All three nodes are connected through internet.

All the nodes were implemented using the commodity machines all running CentOs 6.4.The Cloudera software deployed was based on the Cloudera Pracels 5.1.2, which contained the Hadoop Version 2.3. It supports a file browser, Hive, Pig, Oozie and more.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hosts | Host Name | CPU | RAM | Disk |
| Master Node | DN1 | Intel Xeon  (4C) 3GHz | 15.5 GB | 100 GB |
| DataNode1 | Master | Intel Xeon  (4C) 3GHz | 15.5 GB | 100 GB |
| Data Node2 | DN2 | Intel Xeon  (4C) 3GHz | 15.5 GB | 100 GB |

**2.1 Installing and Adding Accumulo Service to the Cluster:**

Step 1: Installing Accumulo:

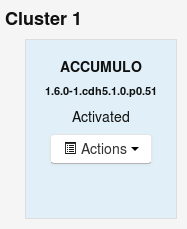
1. From the CLoudera Manager tab, select the parcels.



2. Under the parcel entry for Accumulo 1.6.0 , click Download.

3. Find the Accumulo pracel, under the cluster(in my case cluster name is Cluster 1) and click distribute.

4. Find the Accumulo pracel, under the cluster and click Activate.



After the activation of the parcel, cluster restart message will be prompted, just skip the message and click **close**.

Step 2: Configure HDFS:

To gaurd against the data loss, HDFS must be configured to durably write data upon file close. If the following configuration changes are made, Accumulo will complain loudly via warning messages until issue is corrected.

1. Navigate to cluster's **HDFS Service** page.

2. Click on the "**Configuration**" tab.

3. Search for "hdfs-site.xml"

4. Search for the Service-Wide /Advanced section's property for "**HDFS Service Advanced Configuration Snippet(Safety Value) for hdfs-site.xm**l"

5. CLick on the field and add this snippet:

<Property>

<name>dfs.datanode.synconclose</name>

<value>true</value>

</property>

6. Search for any Gateway group properties lableded " **HDFS CLient Advanced Configuration Snippet(Safety Value) for hdfs-site.xml** "

7. CLick on the field and add this snippet:

<Property>

<name>dfs.datanode.synconclose</name>

<value>true</value>

</property>

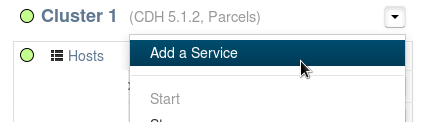
8. Save your changes with a descriptive message, such as "HDFS changes foe Accumulo".

9. Restart the HDFS Service.

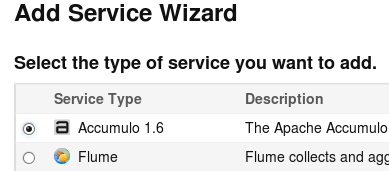
Step 3: Adding the Accumulo service:

1. Navigate to the Cloudera Manager **Home** Page.

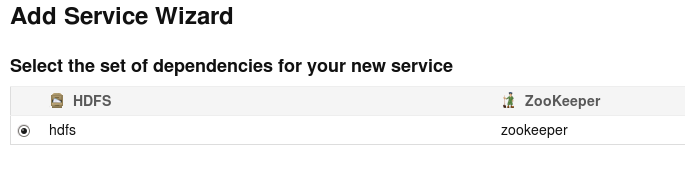
2. Click the actions menu for the cluster to add the Accumulo Service to, select **Add a service**.



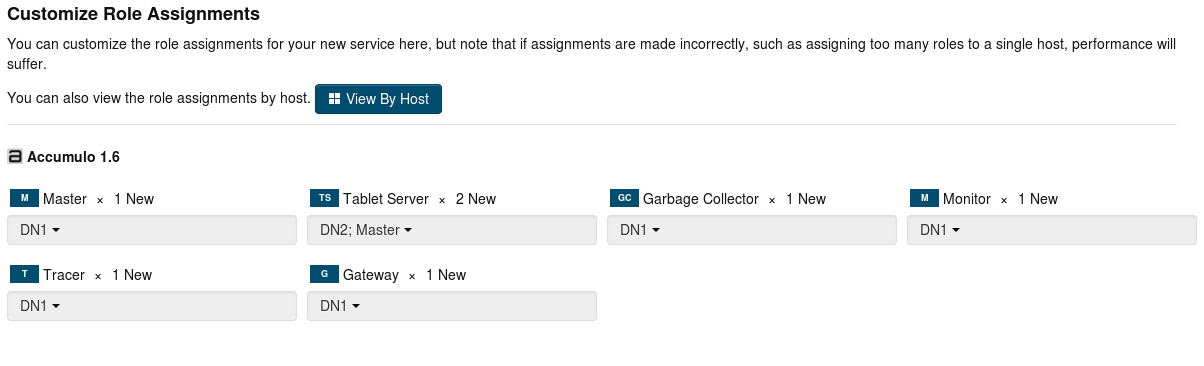
3. Select **Accumulo 1.6** and click **continue**.



4. Select the dependent services(i.e., HDFS and ZooKeeper) and click Continue.



5. Assign the Accumulo roles to the hosts in the cluster. Assign a Tablet Server role on each host that is assigned the DataNode role. The Monitor, garbage Collector, Tracer, and Master roles should be assigned to non-DataNodes. The Gateway role should be assigned to any hosts where there are no Accumulo roles assigned. Following picture depicts the assignment of Accumulo roles in the Project.



6. Configure the **Accumulo Instance Secret** .It is strongly recommended to protect this secret,because the security of the Accumulo service relies on it.

7. Configuration of the **Accumulo Instance Name**.

8. Configure the **Trace user** and **Trace Password** settings. It is strongly recommended that not to leave the Tracer User set to root as this is not a secure configuration.

9. Click **Continue.**

10. Wait while Cloudera Manager does initial service set up.

11. Click **Continue.**

12. Click **Finish**.

**2.2 Roles assigned on the Cluster Nodes:**

Following Table shows the roles assigned to the Cluster hosts to after adding the Accumulo service to the cluster.

|  |  |
| --- | --- |
| Hostnames | Roles |
| DN1 | **?????????????????????????????????** |
| Master | **?????????????????????????????????** |
| DN2 | **?????????????????????????????????** |

**3. Read and Write operations of Accumulo:**

Connecting to Accumulo:

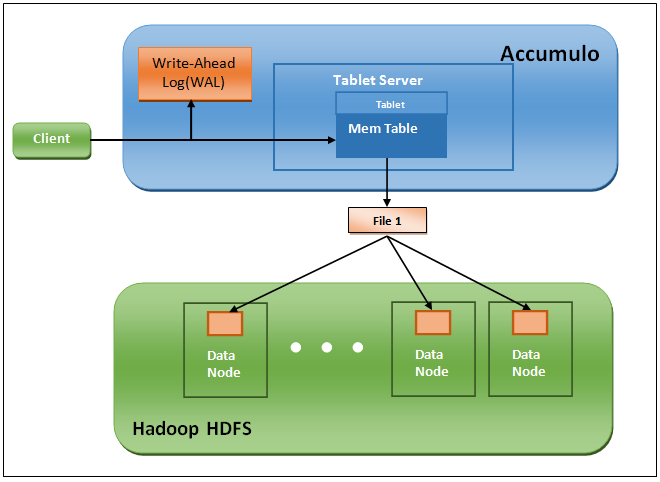
Both data and administrative actions are handled through an Accumulo Connector. The Connector is obtained from an Accumulo Instance. An Accumulo instance is uniquely identified by a set of Zookeeper servers and an instance name.

Writing Data:

Writing data into Accumulo is accomplished by creating a Mutation object and adding it to Batch Writer.

A Mutation encapsulates a set of changes to a single row. The changes can be either puts or deletes. All the changes within a single mutation can be applied atomically, they either succeed or fail as a group, since a row is always assigned to exactly one Tablet Server, which makes Accumulo easy for applications to make concurrent updates to a row without worrying about Mutations being partially applied.

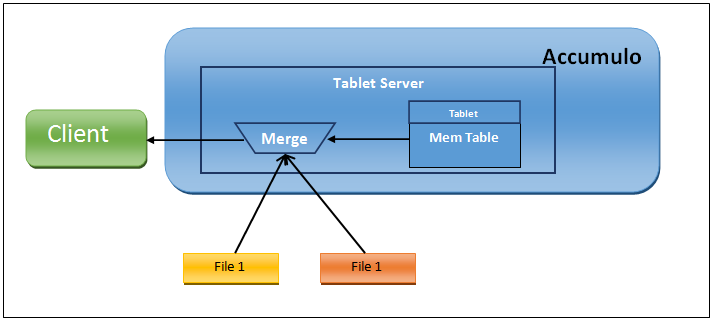
When a write arrives at a Tablet server it is written to a Write-A head Log and then inserted into a sorted data structure in memory called a Mem Table. When the Mem table reaches a certain size the Tablet Server writes out the sorted Key-value pairs to a file in HDFS called Indexed Sequential Access Method (ISAM) file. This process is called a Minor compaction. A new Mem Table is then created and the fact of the compaction is recorded in the Write-Ahead Log.



Reading Data:

Reading data from Accumulo is accomplished with a Scanner. A Scanner returns data via an iterator over Key-Value pairs. By Default a Scanner will return Key-value pairs starting at the beginning of a table and eventually will return all Key-value pairs.

When a request to read arrives at a Tablet Server, the Tablet Server does a binary search across the Mem Table as well as the in-memory indexes associated with each ISAM (Indexed Sequential Access Method) file to find the relevant values. If scan operation is performed, several Key-value pairs are returned in order from the Mem Table and the set of ISAM files by performing a merge-sort as they are read.



**4. Data Model Used for "Books Online" Application:**

The Data Model used for the "Books Online" application is as following:

Following are attributes used for the Key:

1. Book Name serves as RowId , which is used by the Application for the Scanning.

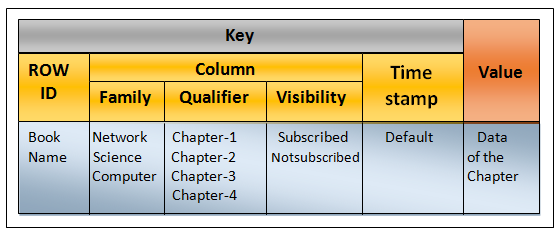
2. Network, Science and Computer are three categories for Column Family.

3. Chapter-1, Chapter-2, Chapter-3 and Chapter-4 are used as Column Qualifier.

4. "Subscribed" and "Notsubscribed" are two Column Visibilities Labels, which are used by the Application to differentiate between the subscribed and not, while retrieving the data.

5. Default time stamp is used for column Timestamp.

Value of the application will be the information from the respective chapter.



**Functionality:**

There are number of ways Accumulo controls access to data in its tables:

1. Authentication

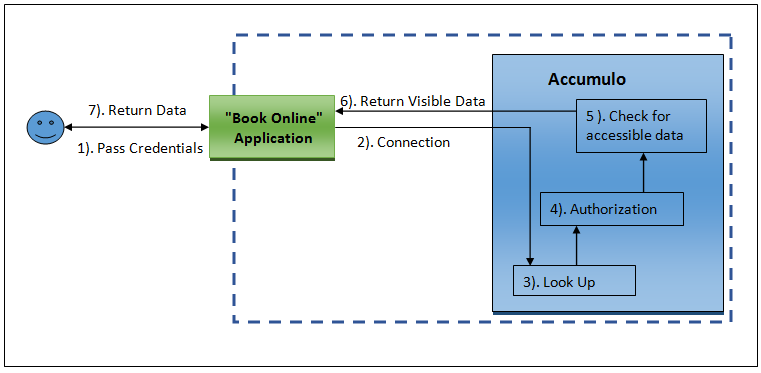
2. Permissions

3. authorization and

4. column visibilities

The below Logical architecture explains the design behind the access control at the various levels especially at the cell level.

When a user sends a read/write request to Accumulo through the GUI Application by passing the credentials, in the form of connection.



**References:**

Oreilly text book

Apache Accumulo for developers

Apache Accumulo Manual 1.6

Accumulo Installation using cloudera Manual

http://www.slideshare.net/jaredwinick/introduction-to-apache-accumulo