Engineering Excellence - J1

(Hadoop Cluster Setup - Movielens Data)

Solution Architecture

TABLE OF CONTENT

Solution Architecture – Hadoop Data Lake

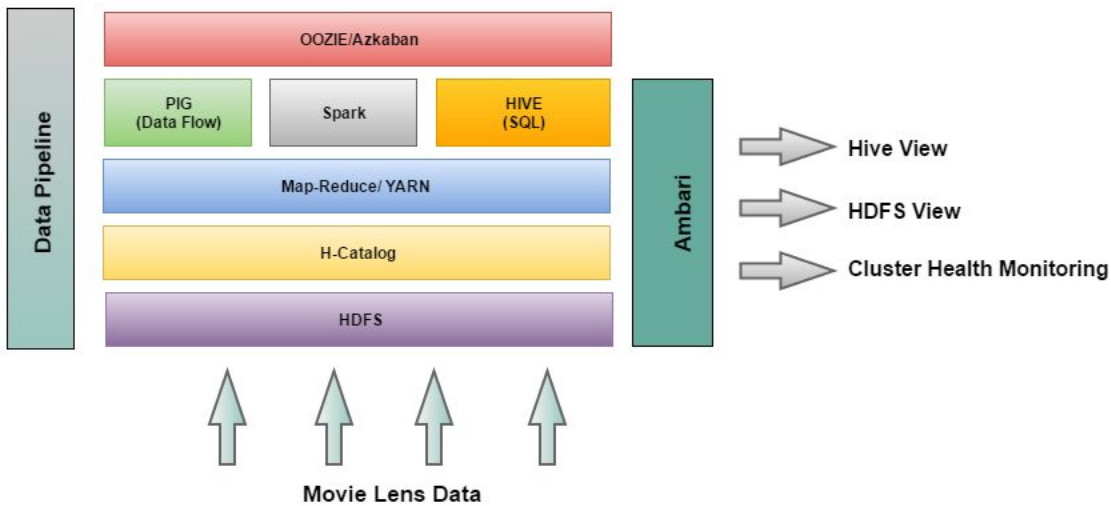
Hadoop Data Lake

A Hadoop data lake is a **data management platform** comprising one or more Hadoop clusters used principally to process and store non-relational data such as log files, Internet clickstream records, sensor data, JSON objects, images and social media posts. In our case we are using Movie Lens Data.

Such systems can also hold transactional data pulled from relational databases, but they're designed to support analytics applications, not to handle transaction processing themselves.

Hadoop Data lake is responsible for spanning everything from Hadoop tuning to setting up the top chain & processing the data. The following diagram shows the complexity of the stack.

The main detail here is that data pipelines take raw data and convert it into insight (or value).



Solution Architecture - Hadoop Data Lake

The Initial Capabilities of Hadoop Data Lake (Movielens data):

This data lake supports the following capabilities:

- To capture and store raw data.
- To store many types of data in the same repository.
- To perform transformations on the data.
- To define the structure of the data at the time it is used, referred to as schema on read.
- To perform new types of data processing.
- To perform single subject analytics based on very specific use cases.

Cluster Details

Below 4 nodes are used as part of this case study-

impetus-i0161.impetus.co.in

impetus-i0203.impetus.co.in

impetus-i0163.impetus.co.in

impetus-i0095.impetus.co.in

These nodes contain following configurations:

Operating System	64-Bit Linux 3.13.0-24-generic #46-Ubuntu SMP
Processor	Quad Core Processor: Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz
RAM	8 GB
Disc Capacity of Cluster	1.1 TB

Load/Ingest Data

Movielens data

This dataset describes 5-star rating and free-text tagging activity from [MovieLens] (http://movielens.org), a movie recommendation service. It contains 47644977 ratings for 34308 movies. These data were created by 247853 users between

January 09, 1995 and January 29, 2016. This dataset was generated on January 29, 2016.

Staging

Once the data has arrived in Hadoop as a whole, there remains the task of staging it for processing. This is not just about storing it somewhere, but rather storing it in the right format, with the right size, and the right access mask.

Ratings Data File Structure (ratings.csv)

All ratings are contained in the file `ratings.csv`. Each line of this file after the header row represents one rating of one movie by one user, and has the following format:

userId, movieId, rating, timestamp

The lines within this file are ordered first by userId, then, within user, by movieId.

Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

Below is the sample records from ratings.csv

hdfs@impetus-i0161:~$ hdfs dfs -tail /user/hdfs/movie_lens_data/ratings/ratings.csv

247752,4993,0.5,1287412650

247752,5952,0.5,1287412663

247752,7153,0.5,1287412661

247752,8874,4.0,1287412729

247752,27773,2.5,1287413266

247752,30749,4.0,1287412625

Movies Data File Structure (movies.csv)

Movie information is contained in the file `movies.csv`. Each line of this file after the header row represents one movie, and has the following format:

movieId, title, genres

Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

Genres are a pipe-separated list, and are selected from the following:

* Action
* Adventure
* Animation
* Children's
* Comedy
* Crime
* Documentary
* Drama
* Fantasy
* Film-Noir
* Horror
* Musical
* Mystery
* Romance
* Sci-Fi
* Thriller
* War
* Western
* (no genres listed)

Below is the sample records from movies.csv

hdfs@impetus-i0161:~$ hdfs dfs -tail /user/hdfs/movie_lens_data_bkp/movies/movies.csv

151657,iMurders (2008),Drama|Horror|Mystery|Thriller

151661,Autoerotic (2011),Drama|Romance

151663,"Semen, a Love Sample (2005)",Comedy|Romance

151667,Romance on the Run (1938),(no genres listed)

151669,Genetic Me (2014),(no genres listed)

151671,The Chosen (2015),Thriller

151673,Hustle & Heat (2003),Action|Comedy|Crime|Romance|Thriller

Users Data File Structure (users.csv)

This file contains demographic information about the users; this is a comma separated list with following format:

userId, age, gender, occupation, zip-code

Below is the sample records from movies.csv

hdfs@impetus-i0161:~$ hdfs dfs -tail /user/hdfs/movie_lens_data_bkp/users/users.csv

247732,Test User 247732,27,F,Engineer,900673

247733,Test User 247733,35,F,None,425922

247734,Test User 247734,24,M,HomeMaker,885037

247735,Test User 247735,28,F,HomeMaker,849347

247736,Test User 247736,23,M,Writer,732639

247737,Test User 247737,16,M,Librarian,847508

247738,Test User 247738,16,M,Executive,585023

Data Partitioning

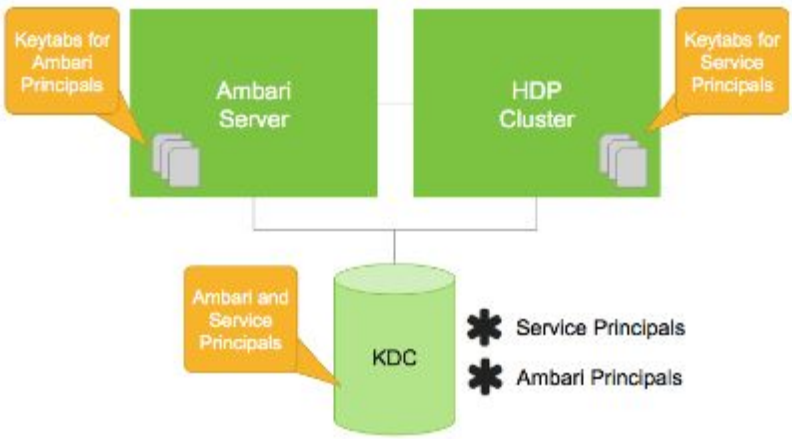
As you land data, there is another important aspect to consider: how you partition or, more generally, size data. For starters, Hadoop is good at managing fewer very large files. You do not want to design an architecture that lands many small files in HDFS and then be surprised when the NameNode starts to perform badly.

Access Control - Kerberos

The last part you have to consider is what we call information architecture (IA), which addresses the need to lay out the data in such a way that multiple teams can work safely on a shared cluster—also referred to as multi-tenancy.

Hadoop and Kerberos Principals

Each service and sub-service in Hadoop must have its own principal. A **principal** name in a given realm consists of a primary name and an instance name, in this case the instance name is the FQDN of the host that runs that service. As services do not log in with a password to acquire their tickets, their principal's authentication credentials are stored in a **keytab** file, which is extracted from the Kerberos database and stored locally in a secured directory with the service principal on the service component host.



Data Transformation & Processing

As part of data processing:

- PIG will extract data from HDFS (source), transform it according to a rule set and then load it into a datastore i.e HIVE. It will ingest data from files, streams (.csv) into Hive tables using HCatalog.

  Below are the PIG scripts used for loading use case data into HIVE tables using PIG:

Load Movie Data

grunt> movies = LOAD 'hdfs://EETeamJ1/user/hdfs/movie_lens_data/movies/movies.csv' USING PigStorage(',') as (movie_id:long,title:chararray,genres:chararray);

grunt> STORE movies INTO 'movie_lens_data.movies' USING org.apache.hive.hcatalog.pig.HCatStorer();

Load Users table

users = LOAD 'hdfs://EETeamJ1//user/hdfs/movie_lens_data/users/users.csv' USING PigStorage(',') as (user_id:long,name:chararray,age:int,gender:chararray,occupation:chararray,zip_code:chararray);

STORE users INTO 'movie_lens_data.users' USING org.apache.hive.hcatalog.pig.HCatStorer();

Load Ratings table

ratings = LOAD 'hdfs://EETeamJ1//user/hdfs/movie_lens_data/ratings/ratings.csv' USING PigStorage(',') as (user_id:long,movie_id:long,rating:float,time_stamp:chararray);

STORE ratings INTO 'movie_lens_data.ratings' USING org.apache.hive.hcatalog.pig.HCatStorer();

- Once data is populated into Hive tables then various data processing & analytics such as select, iteration, and other transforms can be performed using HIVE queries.

Sample HIVE query for Use Case: List all the movies and the number of ratings

```
SELECT movie_id, title, count(*)
 FROM movies
 RIGHT OUTER JOIN ratings
 ON movies.movie_id=ratings.movie_id
 GROUP BY movies.movie_id, title;
```

- Finally results will be stored into the HIVE tables or Hadoop Data File System.

```
INSERT OVERWRITE TABLE mov_rating_count
 SELECT movie_id, title, count(*)
 FROM movies
 RIGHT OUTER JOIN ratings
 ON movies.movie_id=ratings.movie_id
 GROUP BY movies.movie_id, title;
```

Result:

```
hive> select * from mov_rating_count LIMIT 10;
OK
2       Jumanji (1995)  23950
3       Grumpier Old Men (1995) 15267
5       Father of the Bride Part II (1995)      14769
6       Heat (1995)     26593
10      GoldenEye (1995)        31357
13      Balto (1995)    1648
```

14 Nixon (1995) 6750

18 Four Rooms (1995) 5781

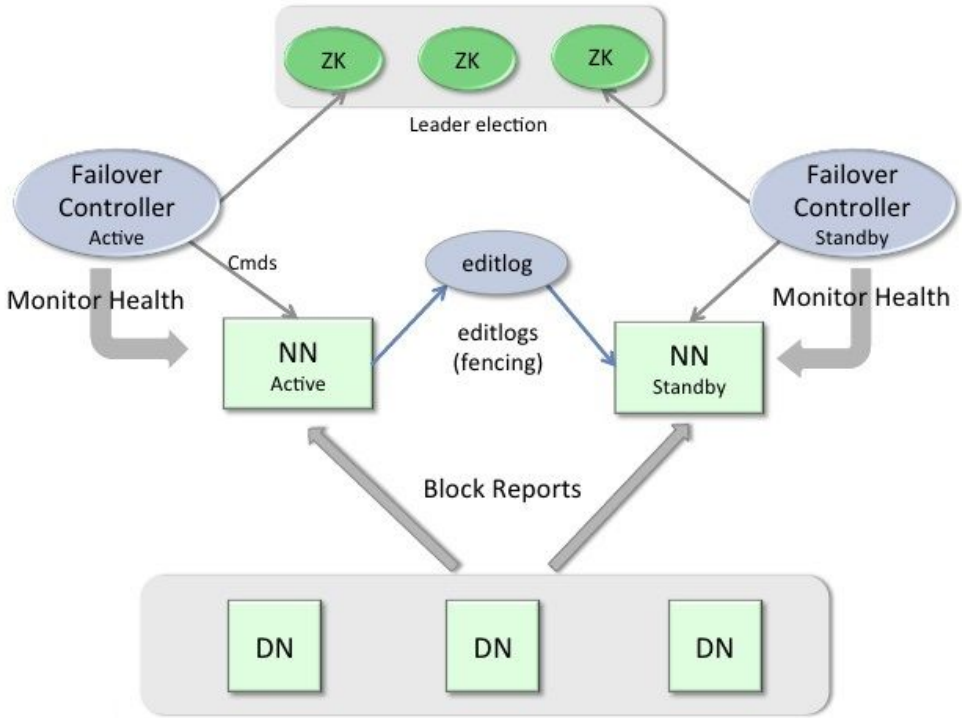
19 Ace Ventura: When Nature Calls (1995) 22877

23 Assassins (1995) 4636

Time taken: 0.867 seconds, Fetched: 10 row(s)

hive>

HDFS High Availability Architecture



HDFS High Availability Architecture

High Availability Set up and configuration

Virtual machine	IP address	Host name
Active name node	172.26.60.19	impetus-i0095
Standby name node	172.26.60.17	impetus-i0203
Active Resource Manager	172.26.60.19	impetus-i0095
Standby Resource Manager	172.26.60.16	impetus-i0161
Data node -1	172.26.60.16	impetus-i0161
Data node -2	172.26.60.17	impetus-i0163
Data node -3	172.26.60.19	impetus-i0095

In a typical HA cluster, two separate machines are configured as NameNodes. In a working cluster, one of the NameNode machine is in the **Active** state, and the other is in the **Standby** state.

The Active NameNode is responsible for all client operations in the cluster, while the Standby acts as a slave. The Standby machine maintains enough state to provide a fast failover (if required).

In order for the Standby node to keep its state synchronized with the Active node, both nodes communicate with a group of separate daemons called **JournalNodes** (JNs). When the Active node performs any namespace modification, the Active node durably logs a modification record to a majority of these JNs. The Standby node reads the edits from the JNs and continuously watches the JNs for changes to the edit log. Once the Standby Node observes the edits, it applies these edits to its own namespace. When using QJM, JournalNodes acts the shared editlog storage. In a failover event, the Standby ensures that it has read all of the edits from the JounalNodes before promoting itself to the Active state. (This mechanism ensures that the namespace state is fully synchronized before a failover completes.)

Note: Secondary NameNode is not required in HA configuration because the Standby node also performs the tasks of the Secondary NameNode.

To provide a fast failover, it is also necessary that the Standby node have up-to-date information on the location of blocks in your cluster. To get accurate information about the block locations, DataNodes are configured with the location of both of the NameNodes, and send block location information and heartbeats to both NameNode machines.

It is vital for the correct operation of an HA cluster that only one of the NameNodes should be Active at a time. Failure to do so, would cause the namespace state to quickly diverge between the two NameNode machines thus causing potential data

loss. (This situation is called a **split-brain scenario**.). To prevent the split-brain scenario, the JournalNodes allow only one NameNode to be a writer at a time. During failover, the NameNode, that is to chosen to become active, takes over the role of writing to the JournalNodes. This process prevents the other NameNode from continuing in the Active state and thus lets the new Active node proceed with the failover safely.

Ambari Monitoring Architecture

The Ambari Server serves as the collection point for data from across your cluster. Each host has a copy of the Ambari Agent - either installed automatically by the Install wizard or manually - which allows the Ambari Server to control each host.

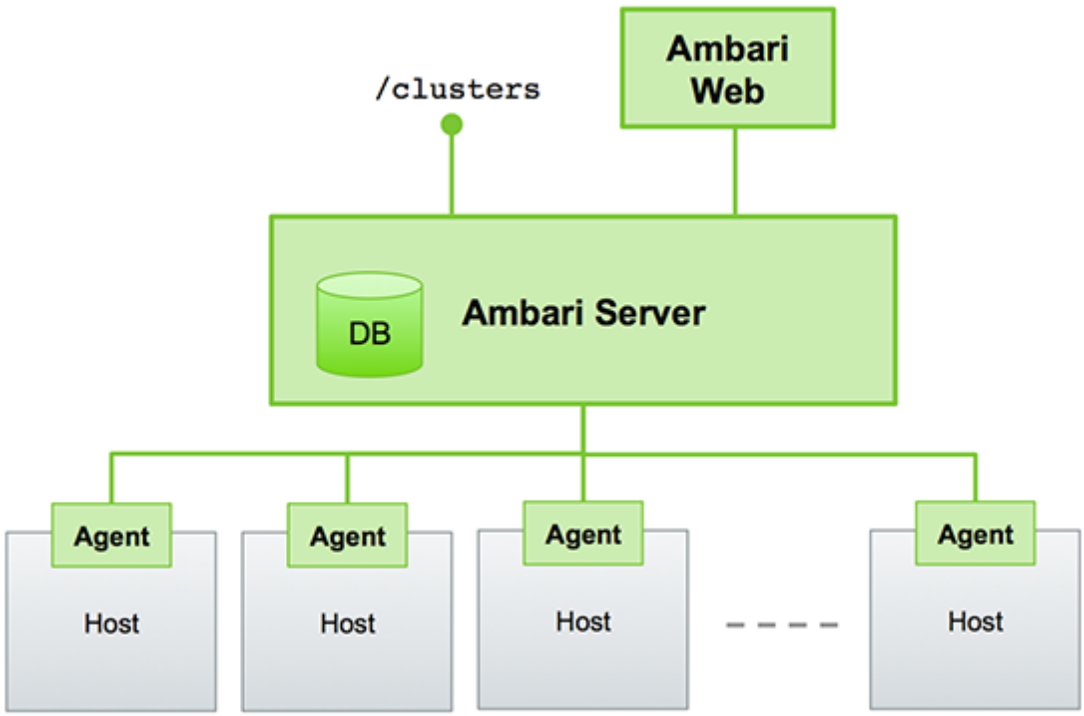


Figure - Ambari Server Architecture

Sessions

Ambari Web is a client-side JavaScript application, which calls the Ambari REST API (accessible from the Ambari Server) to access cluster information and perform cluster operations. After authenticating to Ambari Web, the application authenticates

to the Ambari Server. Communication between the browser and server occurs asynchronously via the REST API.

Ambari Web sessions do not time out. The Ambari Server application constantly accesses the Ambari REST API, which resets the session timeout. During any period of Ambari Web inactivity, the Ambari Web user interface (UI) refreshes automatically. You must explicitly sign out of the Ambari Web UI to destroy the Ambari session with the server.

Ranger

Comprehensive security for Enterprise Hadoop

Apache Ranger delivers a comprehensive approach to security for a Hadoop cluster. It provides central security policy administration across the core enterprise security requirements of authorization, authentication, audit and data protection.

Apache Ranger offers a centralized security framework to manage fine-grained access control over Hadoop data access components like Apache Hive and Apache HBase. Using the Apache Ranger console, security administrators can easily manage policies for access to files, folders, databases, tables, or column. These policies can be set for individual users or groups and then enforced within Hadoop.

Security administrators can also use Apache Ranger to manage audit tracking and policy analytics for deeper control of the environment. The solution also provides an option to delegate administration of certain data to other group owners, with the aim of securely decentralizing data ownership.

Apache Ranger currently supports authorization, authentication, auditing, data encryption and security administration for the following HDP components:

- Apache Hadoop HDFS
- Apache Hive
- Apache HBase
- Apache Storm
- Apache Knox
- Apache Solr
- Apache Kafka
- YARN

Monitoring Tools Used

Ganglia

Ganglia is a complete and real-time monitoring and execution system intended to provide system statistics and important performance metrics for cluster computing systems. It is capable of providing excellent support for clusters as small as a few computers to those as large as 512 systems. One noteworthy feature is that it uses widely popular tools such as RRDtools, XRD, XML and PHP enabled web interfaces for its operation.

Nagios

Nagios is the most popular, open source, powerful monitoring system for any kind of infrastructure. It enables organizations to identify and resolve IT infrastructure problems before they affect critical business processes. Nagios is useful for keeping an inventory of your servers, and making sure your critical services are up and running. Using a monitoring system, like Nagios, is an essential tool for any production server environment.

References

https://www.draw.io/

https://hortonworks.com/products/data-center/hdp/