Hive for Retail Analysis

Team YS | April 24, 2012 at 9:00 am

People always look for convenience! In the early 20th century, retail industry was still in its infancy taking baby steps across Europe and North America. But the latter half of the 20th century saw the emergence of the hypermarket and the supermarket as they truly simplified the *all-in-one-stop* shopping experience.

Retail industry today is big business and will continue to remain so for the foreseeable future. Recent estimates put world-wide retails sales at USD 7.5 trillion. Wal-Mart has been the leader at the global stage since its inception. The world's top 5 retailers are Wal-Mart (USA), Carrefour (France), Royal Ahold (The Netherlands), Home Depot (USA) & Kroger (USA).

In India, retail industry is growing at a rapid pace. Major Indian retailers in this league include Future Group, Reliance Industries, Tata Group and Aditya Birla Group.

One of the retail groups, let's call it BigX in this article, wanted their last 5 years semi- structured dataset to be analyzed for trends and patterns. Let us see how they can solve their problem using Hadoop.

About BigX

BigX is a chain of hypermarket in India. Currently there are 220+ stores across 85 cities and towns in India and employs 35,000+ people. Its annual revenue for the year 2011 was USD 1 Billion. It offers a wide range of products including fashion and apparels, food products, books, furniture, electronics, health care, general merchandise and entertainment sections.

One-third of their stores have daily sales of USD 25K+. The remaining two-thirds have daily sales of USD 14K+ and USD 10K+. On an average, 1200+ customers walk in and purchase products from each of these stores daily.

Problem Scenario

- One of BigX log datasets that needs to be analyzed was approximately 12TB in overall size and holds 5 years of vital information in semi structured form.
- Traditional business intelligence (BI) tools are good up to a certain degree, usually several hundreds of gigabytes. But when the scale is of the order of terabytes and petabytes, these frameworks become inefficient. Also, BI tools work best when data is present in a known pre-defined schema. The particular dataset from BigX was mostly logs which didn't conform to any specific schema.
- It took around 12+ hours to move the data into their Business Intelligence systems bi-weekly. BigX wanted to reduce this time drastically.
- Querying such large data set was taking too long

Solution

This is where Hadoop shines in all its glory as a solution! Let us see how Hadoop was used to solve this problem.

Since the size of the logs dataset is 12TB, at such a large scale, the problem is 2-fold:

Problem 1: Moving the logs dataset to HDFS periodically

Problem 2: Performing the analysis on this HDFS dataset

We had options like Sqoop, Flume , Chukwa etc when we need to move the dataset into HDFS. Since logs are unstructured in this case, Sqoop was of little or no use. So Flume was used to move the log data periodically into HDFS. Once the dataset is inside HDFS, Hive was used to perform various analyses.

Let us see the overall approach in detail below

Problem 1: How Flume solved the data transfer problem?

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. The primary use case for Flume is

as a logging system that gathers a set of log files on every machine in a cluster and aggregates them to a centralized persistent HDFS store.

Sample Architecture of Flume Implementation is illustrated below:

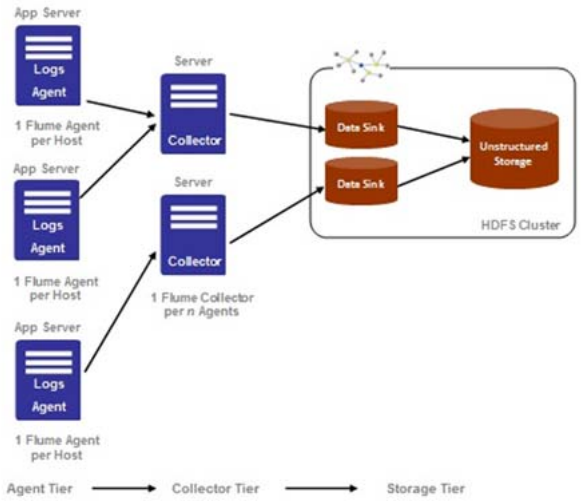

Figure 1: Flume Architecture

Flume's typical *dataflow* is as follows: A Flume Agent is installed on each node of the cluster that produces log messages. These streams of log messages from every node are then sent to the Flume Collector. The collectors then aggregate the streams into larger streams which can then be efficiently written to a storage tier such as HDFS.

Logs from all the nodes can be sent into HDFS on a real-time / daily / weekly / monthly basis. We chose to send certain logs bi-weekly mainly because of the analytical aspect of the requirement and hence daily basis was not warranted in this regard.

Problem 2: Analysis using Hive

Hive is a *data warehouse infrastructure* built on top of Hadoop for providing data summarization, query and analysis. It provides an SQL-like language called HiveQL and converts the query into MapReduce tasks.

Sample Architecture of Hive Implementation is illustrated below:

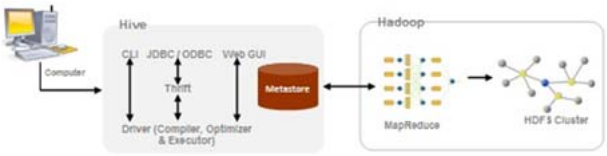

Figure 2: Hive Architecture

Hive uses "*Schema on Read*" unlike a traditional database which uses "*Schema on Write*". Schema on Write implies that a table's schema is enforced at data load time. If the data being loaded doesn't conform to the schema, then it is rejected. This mechanism might slow the loading process of the dataset usually, Whereas Schema on Read doesn't verify the data when it's loaded, but rather when a query is issued. For this precise reason, once the dataset is in HDFS moving it into Hive controlled namespace is usually instantaneous. Hive can also perform analysis on dataset in HDFS or local storage. But the *preferred approach is to move the entire dataset into Hive controlled namespace* (default location – hdfs://user/hive/warehouse) to enable additional query optimizations.

While reading log files, the simplest recommended approach during Hive table creation is to use a RegexSerDe. It uses regular expression (regex) to serialize/deserialize. It deserializes the data using regex and extracts groups as columns. It can also serialize the row object using a format string.

Caveats:

- With RegexSerDe all columns have to be strings. Use "CAST (a AS INT)" to convert columns to other types.
- While moving data from HDFS to Hive, *do not* use the keyword OVERWRITE

Overall Solution Architecture using Flume + Hive

Below figure shows the overall solution architecture implemented for this problem

- Flume was used to collect and transfer log files to HDFS
- Hive was used for analysis

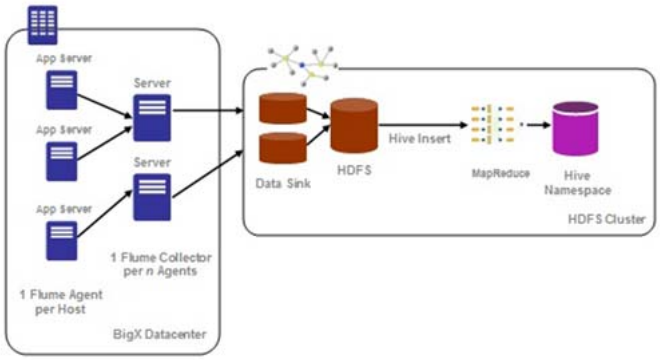

Figure 3: Flume + Hive Architecture

The merchandize details, user information, time of transaction, area / city / state information, coupon codes (if any) , customer data and other related details were collected and aggregated from various backend servers. Flume was installed on these backend servers to transfer the various log files into HDFS. Flume was configured to transfer data on a bi-weekly basis in this case.

As mentioned earlier, the dataset to be analyzed was 12TB. Using the Hadoop default replication factor of 3, it would require 12TB * 3 = 36TB of storage capacity. After a couple of iterations on a smaller sample dataset and subsequent performance tuning, it was decided to go with the following cluster configuration and capacities –

- 45 virtual instances, each with
  - 64-bit OS platform
  - 12 GB RAM
  - 4-6 CPU cores
  - 1 TB Storage

Flume configuration: Following Flume parameters were configured (sample)

- flume.event.max.size.bytes uses the default value of 32KB.
- flume.agent.logdir was changed to point to an appropriate HDFS directory
- flume.master.servers: 3 Flume Masters – flumeMaster1, flumeMaster2, flumeMaster3
- flume.master.store uses the default value – zookeeper

Hive configuration: Following Hive parameters were configured (sample)

- javax.jdo.option.ConnectionURL
- javax.jdo.option.ConnectionDriverName: set the value to “com.mysql.jdbc.Driver”
- javax.jdo.option.ConnectionUserName
- javax.jdo.option.ConnectionPassword

By default, Hive metadata is usually stored in an embeddedDerbydatabase which allows only one user to issue queries. This is not ideal for production purposes. Hence, Hive was configured to use MySQL in this case.

Using the Hadoop system, log transfer time was reduced to ~3 hours bi-weekly and querying time also was significantly improved.

Since this case demands complex querying, Snowflake schema approach was adopted while designing the Hive tables. In a Snowflake schema, *dimension* tables are normalized usually up to 3NF and *fact* tables are not affected.

Some of the schema tables that were present in the final design were – facts, products, customers, categories, locations and payments. Some sample Hive queries that were executed as part of the analysis are as follows –

- **Count the number of transactions**

Select count (*) from facts;

- **Count the number of distinct users by gender**

Select gender, count (DISTINCT customer_id) from customers group by gender;

Only equality joins, inner & outer joins, semi joins and map joins are supported in Hive. Hive does not support join

conditions that are not equality conditions as it is very difficult to express such conditions as a MapReduce job. Also, more than two tables can be joined in Hive.

- **List the category to which the product belongs**

Select products .product_name, products .product_id, categories.category_name from products JOIN categories ON (products.product_category_id = categories.category_id);

- **Count of the number of transactions from each location**

Select locations.location_name, count (DISTINCT facts.payment_id) from facts JOIN locations ON (facts.location_id = locations.location_id) group by locations .location_name;

Interesting trends / analysis using Hive

Some of the interesting trends that were observed from this dataset using Hive were:

- There was a healthy increase in YoY growth across all retail product categories
- *Health & Beauty Products* saw the highest growth rate at 65%, closely followed by *Food Products* (55 %) and *Entertainment (54.6%).*
- Northern India spends more on Health & Beauty Products andSouth India spends more on Books and Food Products
- Delhiand Mumbai take the top spot for the purchase of Fashion & Apparels
- Karnataka tops the list for the purchase of Electronics and Andhra Pradesh & Tamil Nadu for the purchase of Food Products
- A very interesting and out-of-the-ordinary observation was that men shop more than women! Though the difference isn't much, it's quite shocking J (Note: when a couple comes together, that transaction is treated as the man doing the business)

About the authors:
Harish Ganesan is the Chief Technology Officer (CTO) and Co-Founder of 8KMiles and is responsible for the overall technology direction of its products and services. Harish Ganesan holds a management degree from Indian Institute of Management, Bangalore and Master of Computer Applications from Bharathidasan University , India.

Vijay is the Big Data Lead at 8KMiles and has 5+ years of experience in architecting Large Scale Distributed Web Systems and engineering Information Systems – Retrieval, Extraction & Management. He holds M. Tech in Information Retrieval from IIIT-B.

Team YS

Latest Posts

TOPICS

Add a comment...

Also post on Facebook

Posting as Ashok Agarwal (Not you?) Comment

Facebook social plugin