**Unit-5**

**Archive Data using Hadoop**

**Why do we archive data?** Data has been growing at a fast pace, and it has been impractical to archive all data.The practice of archiving “low” valued data is often performed by moving from its primary storage to secondary storage. This is a common practice for most organizations, with varying standards and criteria for which data is considered for archiving.

The concept of “Tiered Data Archiving” is assigning tiers to an organization’s data based on its perceived value. Tier 1 data is then made to always be readily available in transactional systems, allowing the application to be less bloated and more responsive. Tier 2 data is often still viewed as valuable to the business, and may be potentially archived while still allowing for analytics to be run, but is not essential for daily transactional systems. Tier 3 data may have very little business value, but may be required to retain for legal or auditing purposes.
Sounds good, right? Unfortunately, an average of 80% or more of a company’s data is unstructured.

**Typical motivations for data archiving are:**

* Improve performance of primary storage or RDBMS system
* Reduce storage cost
* Maintain historical data typically for audit purposes, or government mandated compliance
* Value to Cost Ratio: As time progresses, value of past data decreases whereas cost does not

Archiving data is an unfortunate necessity, and policies differ from organization to organization. Traditional data archiving is expensive, difficult to analyze, and is often unstructured; when needs arise to access this data, companies are faced with two options:

* Tape/SAN/NAS Storage: Queries or analytics of unstructured data will require custom programming which is difficult to manage for urgent, “last-minute” requests.
* RDBMS Storage: Expensive (including costs from hardware, software, and resourcing), requires maintenance, and has volume & capacity constraints for large amounts of data.

**Traditional Data Archiving Solutions:**

* Historical data is archived based on either a pre-defined time period or certain audit or legal requirements
* Once data is archived, access to that data is extremely difficult and expensive
* Lack of tiered data storage:

o Expensive, easy to access RDBMS
o Cheap, almost impossible to access Tape and similar archival media

Additionally, different application platforms require unique strategies for handling data. An Oracle ADF platform will differ greatly from a Microsoft .Net/SQL based solution, with differing dependencies based on the value/volume of the data, and other requirements like legal mandates (such as PCI or HIPAA). This archived data is almost impossible to revive back into the enterprise.

**Hadoop Based Archiving with HDFS**

Using Hadoops Distributed File System (HDFS) to archive data, many of the challenges with data archiving are eliminated. All existing data archiving can be channeled through Hadoop, removing the need for backups that have little to no analytical value. Once in the HDFS, this data can be easily accessed through a variety of 3rd party analytics tools such as Impala, OBIEE, or Tableu. With a valuable active archive that is easily accessible for reporting, transaction systems can be unimpeded by these off-loaded analytics processes, which results in overall system performance gains. With the ability to archive all data to Hadoop, even data that at a granular level may not seem that important, but at an aggregated level this data may be extremely important. Because the cost of archiving data in Hadoop is so low, the value of the data in relationship to the cost may increase. With Hadoop, data can also be queried in real-time, and can be brought back into the enterprise. This is especially useful for organizations that have data scientist or researchers, who would now have the ability to access this data as needed for analytical purposes.



**What’s Next : Data Analytics**

 

Depicted above, Hadoop can be configured to be the keeper of all enterprise data, starting with the ingestion of data. If Hadoop and HDFS is the data hub from the beginning, the options are endless. Since Hadoop is distributed computing platform, it makes for an efficient compute engine; an example would be configuring Hadoop for replacing long-running ETL jobs in need of performance improvements.

When considering use of Hadoop for archiving, it’s important to understand the Hadoop framework’s architecture:

* Designed for running on large clusters of commodity hardware
* Allows for distributed processing of large data sets across clusters of servers using simple programming models

 

Shown above, at the bottom of the stack is the Hadoop Distributed File System (HDFS), which is schema-less. On top of the HDFS is the Map-Reduce computational logic, which allows for parallel distributed processing. Since Map-Reduce programming models are fairly complex, another layer of top-level abstractions are needed to access the database. Some of these include PIG, Hive, and Sqoop. These platforms are related to more traditional programming models and languages. For example, Hive’s language is very similar to SQL; using Hive, developers can write SQL queries which will essentially be converted into Map-Reduce. These intermediaries allow for commonly used database and business intelligence/analytical tools (such as OBIEE or Tableau) via an ODBC connection to access HDFS data.

Another primary method of accessing data in Hadoop is utilizing Sqoop, which is similar to Oracle’s SQL Loader, which is a batch export process.

 

The first step is to transfer data into HDFS, of which there are a few different methods. The first in the above diagram, Flume, is a tool used that takes streaming data such as Twitter feeds or log files and transfers it into HDFS. The second method is utilizing Sqoop, which interfaces well with traditional RDBMS’, such as Oracle or SQL Server. Another means of transferring static/file-based data is the use of HTTPFS, which uses provides an HTTP gateway with a RESTful API to serve data to Hadoop.

Once the data is within HDFS, we can compute against it, such as running an ETL process or connecting via Sqoop to push the data back into the enterprise to run analytics.

**Applying Hadoop for Archiving in your Organization**

Now that we have a basic understanding of Hadoop and some of the tools surrounding it, we can consider a ready-to-use Hadoop Active Archiving Framework by Aptude, which is designed to be used over and over again. Our design uses a declarative approach, where the declarations used by our framework are used by XML. These declarations will contain unique variables needed for the archival architecture to run, with no additional coding needed from your organization. For example, the XML will contain the source database, the parent-child relationships among tables, and the related table keys. The declarations will also include the periodicity for data archival. Our Hadoop solution framework will utilize Hive to orchestrate the declarations created by your organization.



In the diagram above, we are utilizing a java controller program which is scheduled via Oozie. This is a program that coordinates the previously mentioned declarations. Sqoop will then extract the data from the appropriate RDBMS and load data into HDFS. On the initial load, we will simply import the data. Subsequent loads will go through a merge process, which will occur in a staged approach; after completion of the merge, the data will be available within HDFS via Hive for queries or additional analytics.

The above flow diagram shows our archival method; our Oozie coordinator will oversee the workflow and will periodically the archival job will be initiated. The initial steps include using Scoop to load our desired data from the RDBMS into the HDFS staging area. We will then run a merge process, in which we are utilizing two warehouses configured for redundancy and fault tolerance.

**Conclusion**

As we have discussed throughout this article, one of the advantages of using Hadoop for archiving is that it is an active archive. Although the physical RDBMS structure doesn’t exist via HDFS, we can create an abstraction via Hive, which is similar to the tables used within an EDW. Using Hive, we can query the data using a SQL-like language. In addition, the archive data which is within HDFS can connect to an analytical tool such as OBIEE or Tableau.