Introduction to HBase

Apache HBase

The agenda of this discussion would be

- What is HBase
- Why HBase
- When to you use HBase
- When not to you HBase and
- Some of the industry use cases of HBase?

First of all what is HBase?

HBase is basically a column oriented database management system that runs on top of the Hadoop distributed file system that is your HDFS. It is an open source, non-relational, distributed database model after Google's big table and it is completely written in java.

HBase basically allows for low latency quick look's up in Hadoop and also adds transactional capabilities to Hadoop allowing users to conduct updates, inserts and deletes.

It is primarily well suited for parse data sets which are common in many big data use cases and like relational database management systems HBase does not support a structured query language likes SQL, in fact HBase isn't a relational data store at all. HBase applications are primarily written in Java, much like a typical MapReduce applications and also

supports writing applications in Avro, REST APIs and Thrift.

Why HBase?
We already have HDFS as a data store in Hadoop. So why we do need another data store? For that's question let's take a look at some of the properties of HDFS to get an answer.

HDFS is a distributed file system which has the following properties. It is optimized for streaming access of large files. Here you would typically store files that are in hundreds of Mb's and upwards in HDFS and try access them through MapReduce to process them in batch mode and HDFS files are right once files you can append to the file in some of the recent version of Hadoop.

But that is not a feature that is very commonly used. Consider HDFS files as write-once and read many times kind of files. Here there is no concept of random writes. Finally, HDFS does not do random reads as well. That is you cannot read from the middle of the file in HDFS. This is why HBase actually comes into picture.

HBase is able to address these problems using a data structure that is based on LSM trees or law structure merge trees. We are going to

talk about LSM trees briefly in the latest stages. that because of using such a data structure and because of the services it runs as the H master region server, we are going to talk about these concepts as well later, now because of these things HBase provides you the following features

- HBase provide low latency access to small amounts of data within a large data set which HDFS cannot do.
- You can access single rows very quickly from say a billion row table. It also stores the key value pairs in columnar fashion.
- Iit has a flexible data model to work with and data here is indexed by row key.
- Fast scan across huge tables is possible that is one of the most important features that HBase provides and it also provides scalability in terms of writes as well as total volume of data because HBase internally stores the data on HDFS and we know that further fact that HDFS and Hadoop generally scales really well for large amounts of data and this features is also extended to HBase. So basically HDFS is most suited for off line batch processing needs whereas HBase is used when you have a real time look up need for extremely fast look up's.

Now let's look at a few reason or scenario as to when you could potentially use HBase. HBase is not suitable for every problem. Remember that these are just the guide lines.

The first thing is that we need to make sure that we have lots and lots of data that if you have hundreds or millions or billions of rows then probably HBase is a good candidate.

You should use HBase whenever you have a need for random real time read or write access to your data in a HDFS. The goal of HBase is to host very large tables that is billions of rows and millions of columns. However, this large amount of data which is consisting of millions of rows and billions of columns is going to be stored on top of clusters of commodity hardware that is the goal of HBase.

Finally, if your application has a variable schema where each row is slightly or may be even drastically different from the other then you should be looking at HBase.

Let's look at some of the guide lines as when not to use HBase. HBase is not a whole sale replacement for your relational databases.

HBase first of all doesn't speak the language of structured query language or SQL. it does not have a query Optimizer and it does not support cross record transactions or joins so if you are not going to use any of these features in a database application then HBase would be a good fit else HBase would be a bad fit.

Do not use HBase if you have only few thousand or few million records. In such cases using a traditional RDBS might be a better choice. The reason for this is even if you use HBase cluster for smaller data sets all of the data might end up in a single node or probably two or three nodes at the max and the rest of the cluster might be sitting idle, which is not a good situation for your resources.

HBase does not have some of the features of the RDBMS such as

- Typed columns
- Secondary indexes
- Transaction etc.

So if you need them in the database make sure to use such kind of features but you cannot use them in your HBase.

HBase is not optimized for classical transactional applications or even relational analytics. So you cannot use HBase there as well.

The other thing is that we need to make sure that you have enough hardware resources. HBase does not work well with anything less than five nodes, five physical nodes. Primarily because it uses HDFS as a storage layer and when we use HDFS we know that HDFS will not work well with anything less than five data nodes because of the factors such as HDFS block replication which has to be having a default value of three and because it also has to have a name node. so cluster with five data nodes and a name node is a minimum requirement for HBase and typically Hadoop HDFS as well. if you plan to use the cluster of five node or lesser then you really need to think again unless you are going to scale subsequently.

With this knowledge let's look at some of the companies that are actually using HBase.