Google File Systems

Prudhvi Krishna Narra

Date: 5/8/2014

Google*

Ref: **http://www.labouseur.com/courses/db/papers/ghemawat.sosp03.gfs.pdf**

Main idea of this paper

To maintain the structured data Google had implemented and deployed a distributed storage system called as Bigtable after examining a variety of uses of a Bigtable like system. The basic idea behind it is to store and manage the myriad amount of information or data. Through Bigtable it was presumed to leverage data and achieve scalability, applicability, availability and high performance. It provides clients a simple data model through which the data gets controlled and served dynamically from out of memory or from the disk and allows them to reason about location properties of the data at the underlying storage.

A Bigtable is a multidimensional sorted map and is indexed by a row key, column key and a timestamp where each value in the map is an uninterpreted array of bytes. The row keys are the strings which can be up to 64 KB though users use only around 10 to 100 bytes for it. Every read and write operation under a row key is atomic so that clients can differentiate the concurrent updates to the same row. The data in Bigtable is maintained in alphabetical order by row key where the row range is dynamically partitioned and this each row range is called *tablet,* it is the unit of distribution and load balancing. Read operation on short row ranges are efficient that they can be fetched by communicating through a small number of machines. The row key is generally reverse of the website, pages from the same domain are placed near to each other so that manipulation of data will be efficient.

The basic unit of access control is done by the column keys, these are grouped into sets called column families. Each column family store homogenous type of data. If there is any data to be stored under a column ,firstly a column family to that column key need to be created. A column key is named using the syntax "family.qualifier" like while storing the data of a particular website, which has got many web pages, information about the language of the web page can be stored in a column family called *language,* in the form of a language ID. Access control is completely performed at the column family level. With this we can add the base data, new data and also we can derive the new base data with column families.

Data keeps on changing and we need to revise it, to be up to date. For this reason multiple versions of the same data are created with updates, these versions are indexed by timestamp. Bigtable time stamps are 64 bit integers, these are assigned by Bigtable in the form of date and time including microseconds. Different versions of a cell are stored in descending order so that the latest one can be picked first. These versions presence in a cell are manageable that we can ask the Bigtable to garbage collect the data which is no more required. We can put the last two versions alone or last ten days revisions etc.

In this way we can make the huge amounts of data get stored in a row range. With the help of this row keys, column keys which are wrapped into column families, updated data or revised data with a timestamp makes the Bigtable to maintain the scalability, applicability, availability and high performance.

Implementation of the Idea

The components in implementing the Bigtable are a library linked to every client, a master server and a number of tablet servers. Master servers does the following, It assigns tablets to tablet servers, addition and expiration of tablet servers, balancing load of tablet-server and garbage collection of files in GFS, handles column and family creations.

Tablet server manages a set of tablets. If a tablet is too large, it splits them in order to keep up the efficiency. It handles all the read and write requests to the tablets with out any interference with the master server, Because of this master server is lightly loaded.

A Bigtable cluster stores data in a number of tablets. Tablet is the one which contains all the data associated with in its row range. The tablets creation starts with just one tablet and as it grows, automatically each one is split into multiple tablets, approximately each is of size 100 to 200 MB.

The location of the root tablet is stored in a file that in turn is stored in a chubby, which is treated as first level in locating the tablet. This root tablet contains the location of the Metadata table which contains the location of the tablet. Root tablet is the first tablet in Metadata table, this will not be split even if the tablet grows a bit longer, so as to maintain three levels in the hierarchy to find the tablet location.

Each Metadata row stores around 1 KB of data in memory, with the 120 MB of Metadata tablets, this three level location scheme can address 2^34 tablets. The tablet locations are cached in the client location. If the tablets location information is found incorrect or it is not present in the cache then it recursively moves up the hierarchy to get the location information. When the clients cache is empty, to find the tablets location this algorithm requires three network round trips.

If the clients cache is stale, the algorithm takes up to 6 round trips as it misses to find the tablet location at a single shot, as the Metadata tablets do not move very frequently. The secondary information like the log information of all events pertaining to each tablet will also be stored in the Metadata tablets, which helps us in debugging and analysing the performance of the database Bigtable.

Implementation of this kind of algorithm to read, write operations will go through the above hierarchy to reach the tablet.

My analysis of the idea and implementation

As Business grows big it has been challenging to maintain information or data for any organization. To manipulate this huge data like creating, updating, deleting in a just low storage capable RDBMS becomes tough. In order to maintain these kinds of data at the same time achieving the high performance, scalability, applicability we need to go for a RDBMS which can brag all these features.

Bigtable is one of the simple database which has shown all the above features, it uses row keys which are arranged in lexicographical manner. As shorter is the length of the row key, it is as quicker it is identified. The row ranges are dynamically partitioned and known as *tablets*, which are the basic part of a table. These tablets contain the information pertaining to a row range. The row key is nothing but the reverse of the website name. This row can have any number of columns but we intent to use a limited so that it can be maintained with high performance.

The first column can contain the web pages of the website like the contents of a web page and you can name the column key with "contents". Next column can be used to store the related data to the web pages in the contents column like language and this is called as the column family, we have to first create a column family and then the data can be stored with column key. The timestamp is a concept to keep the revised editions to be read first, as here the web pages are arranged in descending order of their timestamp.

Implementation of this algorithm which is used in the Bigtable is making it no where complex in manipulating the data. Here the chubby which store the Root tablet address which is present on top of the Metadata table will contain the address of the Metadata tablets. Here the Root tablet can not be split, where as the Metadata tablet will be split as it grows longer. This Metadata tablet will contain the actual tablet address in the user tables. This way we can move over the hierarchical path to fetch the address of the tablet.

Each Metadata row store 1 KB of data in memory. With the 128 MB metadata tablets this three level location scheme can address 2^34 tablets. Clients can refer to their cache for the data, if it was recently fetched, so that we can avoid the processing time to search for it over the huge database. If it is not present in the cache or the information present was found to be wrong then this algorithm requires three round trips in the network to fetch the data. Metadata tablets will store the secondary information like log information of all events pertaining to each tablet.This helps in debugging and analysing the performance of the Bigtable.

Comparing emerging Big Table (GFS) and Traditional RDBMS

1. Horizontal Scalability

Traditional RDBMS approach allows us to scale the system vertically i.e. if we want to excel the storage capacity of the system then we should include Hard disks or DASD's into the same server. As a response it will lead less performance eventually.

Big Table approach allows us to scale the system horizontally i.e. rather than including so many hard drives into same server, we directly include another server and forms a cluster. It will increase the performance of the system.

2. Schema Flexibility

According to IBM(One of the biggest vendors of big data solution) analyzes that 80% data evolving from different operations like social media, experiments …etc. is unstructured in nature and it doesn't always follows specific format i.e. schema.

Traditional RDBMS should adhere to fixed schema i.e. type, size ..etc.it won't be sufficient all the times to have fixed schema.

It has been nullified by the introduction of NoSQL databases like Big Table. Here it doesn't restrict us to follow fixed schema, It will allow different kinds of formats even with in the structure i.e. Table (Column oriented NoSQL databases allows table structures)

3. Complex Queries

Usually RDBMS run's through different tables to get the desired output. It is called JOIN, because of join operations the performance of the system degrades

One of the major outsets of the NoSQL based Big Table database is it doesn't support JOIN operations at all. So, the performance of the system never degrades. If we want to get multiple tables data then we should roam across different table desperately.

4. Data Replication and Distributed Computing

Traditional giant RDBMS usually stores data over a specified server. So, during server failure or disaster time it is difficult to imagine immediate data availability.

It has been one of the Major issues for a long time and now it is overcome by introducing NoSQL Databases. These databases usually stores same copy of data at different nodes in a cluster, if one node failure happens still there is a chance of data retrieval from other nodes and these nodes share data among different nodes at different locations and provides the concept called Distributed Computing.

Advantages and Disadvantages

Advantages:

1. Highly scalable Database and It is capable of handling huge volumes of Unstructured data.

2. It allows us to not to define a particular schema

3. It adhere BASE properties (Basic availability, soft state, Eventual Consistency)

4. It is capable of handling Big Data (in terms of Petabytes)

5. It doesn't allow JOINS. So, it gives very good performance

6. It doesn't require very high maintenance as like RDBMS.

7. It is eventually consistent, i.e. data is consistent in database at some times only

but not always.

Disadvantages:

Apart from having scintillating advantages it has some serious issues to discuss

1. It doesn't adhere to ACID properties

As it's main theme is to give Database availability at any time without locking concept and all. In the case data driven applications high consistency is the proprietary. So, NoSQL databases not and never be suggested.

So, In case of financial, manufacturing and retails applications where data needs to be adhere to ACID properties it has no footprints.

Eg: Banking, Retail … etc.

2. Risk and Support issues.

The repository giant RDBMS has been in the field of data storage for decades and it has so many vendors. So , if any problem occurs we will get instant support for recovery.

Big Table is a new NoSQL database and it hasn't been adopted by many organization completely. So, any issue occurs with it .Instant support hardly happens.

3. Though on paper we can write as, it doesn't require huge administration. But practically it does require skilled persons to install and maintain it. Now industry is far behind in numbers for skilled professionals in this area.