BIG TABLE AND APPLICATION IN BANKING

Mrs. G. Anitha Krishnan
Asst. Professor, Dept. of MCA
SCMS School of Technology and Management,
SSTM, Muttom, Aluva.

Athira T D
P G Scholar, Dept. of MCA
SCMS School of Technology and Management,
SSTM, Muttom, Aluva.

***Abstract* – Bigtable is a distributed storage system and it is used for managing structured data that is designed to a large amounts of data across thousands of commodity servers. In Google many projects of data are stored in Bigtable. The data containing web indexing, Google Finance and Google Earth. In Bigtable these types of applications are place in different demands in terms of both data size and latency size. For all of these kind of Google products Bigtable provided flexible, high- performance solution. In this paper, I describe the single data model provided by Bigtable which gives clients dynamic control over data layout and format and describe the implementation of Bigtable in Banking instead of relational database.**

Keywords: - Column families, data model, timestamps.

I. INTRODUCTION

In Google, A distributed system used for managing structured data is called Bigtable. Bigtable is designed to securely scale to petabytes of data and thousands of machines. Bigtable has several goals such as scalability, high performance, wide applicability and high availability. In Google, more than sixty products and projects Bigtable is used. This including Google Finance, Google Analytics, Personalized search, Writely, and Google Earth. Bigtable is used for these products for a different types of demanding workloads, which starts from throughput-oriented batch processing jobs to latency-sensitive serving of data to users.
The clusters of Bigtable used by these products span a wide range of configurations, from a thousands of servers, and store upto hundred terabytes of data.

In several ways, Bigtable similar to a database: it provides many implementation strategies with databases. The parallel
And main-memory databases have achieved scalability and high performance but Bigtable provides a different interface than such systems. Bigtable does not support a complete relational data model; instead of relational data model, it provides clients with a simple data model that supports dynamic control over data layout and format, and it allows clients to reason about the locality properties of the data represented in the underlying storage. In Bigtable data is indexed using row and column names that can be arbitrary strings. Bigtable also treats data in the form of uninterrupted strings, and clients often serialize various forms of structured and semi-structured data into these strings. Clients can control the locality of their data through careful choices in their schemas. At last schema parameters let clients dynamically control whether to serve data out of memory of from disk.

II. DATA MODEL

A Bigtable is a sparse, distributed persistent multidimensional sorted map. The map is indexed by a row, column key, and a timestamp; each value in the map is an uninterpreted array of bytes.
(row:string, column:string, time:int64) →string

Rows

In a table the row keys are arbitrary strings (up to 64KB in size, The typical size for most of the users is 10-100 bytes).Under a single row the read or write of data is atomic (regardless of the number of different columns being read or written in the row), a design decision helps it easier for clients to reason about the system's behavior in the presence of concurrent updates to the same row.

Bigtable is maintains data in lexicographic order by row key. The range of a row is dynamically partitioned. Each row range is called a *tablet*, which is the load balancing and unit of distribution. As a result, reads of short row ranges are efficient and typically requires communication with only a small number of machines. Clients can exploit this property by selecting their row keys so that they get good locality for their data accesses.

Column Families

Column keys are grouped into sets called *column families*, which form the basic unit of access control. All data stored usually of the same type in a column family. A column family should be created before data can be stored under any column key in that family; after a family has been created, any column key within the family can be used. The main intent that the number of distinct column families in a table be small, and that family rarely change during operation. A table may have an unbounded number of columns.

A column key is named using the following syntax:

Family: qualifier. Column family names must be printable, but qualifiers may be arbitrary strings. The qualifier is the name of the referring site; and the cell content is the link text.

Timestamps

In a Bigtable each cell can contain multiple versions of the same data; these kind of versions are indexed by time stamp. The timestamps in Bigtable are 64-bit integers. They can be assigned by Bigtable, in which case they represent "real time" in microseconds, or explicitly assigned by client applications. Application that need to avoid collisions must generate unique timestamps themselves. Different versions of a cell are stored in decreasing timestamp order, so that the most recent versions can be read first.

To make the management of versioned data less onerous, it support two per-column-family settings that tell Bigtable to garbage-collect cell versions automatically. The client can specify either that only the last n versions be kept.

```
// Open the table
Table *T = OpenOrDie("/bigtable/web/webtable");
 // Write a new anchor and delete an old anchor
RowMutationr1(T,"com.cnn.www"); r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");
Operation op;
Apply(&op, &r1);
```

Figure1: Writing to Bigtable

III. API

The Bigtable API provides functions for creating and deleting tables and column families. It also provides functions for changing cluster, table, and column family metadata, such as access control rights.
The client applications can write or delete values in Bigtable, look up values from individual rows, or iterate over a subset of the data in a table.
Figure1 shows C++ code that uses a RowMutation abstraction to perform a series of updates. (Irrelevant details were elided to keep the example short.) The call to Apply performs an atomic mutation to the Webtable: it adds one anchor to www.cnn.com and deletes a different anchor.
Figure 2 shows C++ code that uses a Scanner abstraction to iterate over all anchors in a particular row. Clients can iterate over multiple column families, and there are several mechanisms for limiting the rows, columns, and timestamps produced by a scan. For example, It restrict the scan above to only produce anchors whose columns match the regular expression anchor:*.cnn.com, or to only produce anchors whose timestamps fall within ten days of the current time.

```
Scanner scanner(T);
ScanStream *scanner.FetchColumnFamily("anchor");
stream->SetReturnAllVersions(); scanner.Lookup("com.cnn.www");
for (; !stream->Done(); stream->Next())
{
   printf("%s %s %lld %s\n",
   scanner.RowName(),
   stream->ColumnName(),
   stream->MicroTimestamp(),
   stream->Value());
   }
```

Figure2: Reading from Bigtable

Bigtable supports several other features that allow the user to manipulate data in more complex ways. First, Bigtable supports single-row transactions, which can be used to perform atomic read-modify-write sequences on data stored under a single row key. Bigtable does not currently support general transactions across row keys, although it provides an interface for batching writes across row keys at the clients. Second, Bigtable allows cells to be used as integer counters. Finally, Bigtable supports the execution of client-supplied scripts in the address spaces of the servers. The scripts are written in a language developed at Google for processing data called Sawzall .
Bigtable can be used with MapReduce , a framework for running large-scale parallel computations developed at Google.

IV. Building Blocks

Bigtable is built on several other pieces of Google infrastructure. Bigtable uses the distributed Google File System (GFS) to store log and data files. A Bigtable cluster typically operates in a shared pool of machines that run a wide variety of other distributed applications, and Bigtable processes often share the same machines with processes from other applications. Bigtable depends on a cluster management system for scheduling jobs, managing resources on shared machines, dealing with machine failures, and monitoring machine status.

The Google SSTable file format is used internally to store Bigtable data. An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings. Operations are provided to look up the value associated with a specified key, and to iterate over all key/value pairs in a specified key range. Internally, each SSTable contains a sequence of blocks (typically each block is 64KB in size, but this is configurable). A block index (stored at the end of the SSTable) is used to locate blocks; the index is loaded into memory when the SSTable is opened. allows us to perform lookups and scans without touching disk.

Bigtable relies on a highly-available and persistent distributed lock service called Chubby. A Chubby service consists of five active replicas, one of which is elected to be the master and actively serve requests. The service is live when a majority of the replicas are running and can communicate with each other. Chubby uses the Paxos algorithm to keep its replicas consistent in the face of failure. Chubby provides a namespace that consists of directories and small files. Each directory or file can be used as a lock, and reads and writes to a file are atomic. The Chubby client library provides consistent caching of chubby files. Each Chubby client maintains a session with a chubby service. A client's session expires if it is unable to renew its session lease within the lease expiration time. When a client's session expires, it loses any locks and open handles. Chubby clients can also register callbacks on chubby files and directories for notification of changes or session expiration.

Bigtable uses Chubby for a variety of tasks: to ensure that there is at most one active master at any time; to store the bootstrap location of Bigtable data ; to discover tablet servers and finalize tablet server deaths ; to store Bigtable schema information (the column family information for each table); and to store access control lists. If Chubby becomes unavailable for an extended period of time, Bigtable becomes unavailable.

V. Implementation

The Bigtable implementation has three major components: a library that is linked into every client, one master server, and many tablet servers. Tablet servers can be dynamically added (or removed) from a cluster to accommodate changes in workloads. The master is responsible for assigning tablets to tablet servers, detecting the addition and expiration of tablet servers, balancing tablet-server load, and garbage collection of files in GFS. In addition, it handles schema changes such as table and column family creations. Each tablet server manages a set of tablets (typically we have somewhere between ten to a thousand tablets per tablet server). The tablet server handles read and write requests to the tablets that it has loaded, and also splits tablets that have grown too large.

As with many single-master distributed storage systems, client data does not move through the master: clients communicate directly with tablet servers for reads and writes. Because Bigtable clients do not rely on the master for tablet location information, most clients never communicate with the master. As a result, the master is lightly loaded in practice.

A Bigtable cluster stores a number of tables. Each table consists of a set of tablets, and each tablet contains all data associated with a row range. Initially, each table consists of just one tablet. As a table grows, it is automatically split into multiple tablets, each approximately 100-200 MB in size by default.

VII. Real Applications

In Real time Bigtable is used in several products such as Google analytics, Google earth and personalized search.

6.1, Google Analytics

Google Analytics (analytics.google.com) is a service that helps webmasters analyze traffic patterns at their web sites. It provides aggregate statistics, such as the number of unique visitors per day and the page views per URL per day, as well as site-tracking reports, such as the percentage of users that made a purchase, given that they earlier viewed a specific page.

To enable the service, webmasters embed a small JavaScript program in their web pages. This program is invoked whenever a page is visited. It records various information about the request in Google Analytics, such as a user identifier and information about the page being fetched.

Google Analytics summarizes this data and makes it available to webmasters.

6.2, Google Earth

Google operates a collection of services that provide users with access to high-resolution satellite imagery of the world's surface, both through the web-based Google Maps interface and through the Google Earth custom client software. These products allow users to navigate across the world's surface: they can pan, view, and annotate satellite imagery at many different levels of resolution. This system uses one table to preprocess data, and a different set of tables for serving client data.

6.3. Personalized Search

PersonalizedSearch is an opt-in service that records user queries and clicks across a variety of Google properties such as web search, images, and news. Users can browse their search histories to revisit their old queries and clicks, and they can ask for personalized search results based on their historical Google usage patterns. Each data element uses as its Bigtable timestamp the time at which the corresponding user action occurred.

The Personalized Search data is replicated across several Bigtable clusters to increase availability and to reduce latency due to distance from clients. The Personalized Search team originally built a client-side replication mechanism on top of Bigtable that ensured eventual consistency of all replicas. The current system now uses a replication subsystem that is built into the servers. The design of the Personalized Search storage system allows other groups to add new per-user information in their own columns, and the system is now used by many other Google properties that need to store per-user configuration options and settings. Sharing a table amongst many groups resulted in an unusually large number of column families. To help support sharing, we added a simple quota mechanism to Bigtable to limit the storage consumption by any particular client in shared tables; this mechanism provides some isolation between the various product groups using this system for per-user information storage.

VII. Bigtable In Banking

In Banking Fields currently data are stored in relational databases. There are certain limitations in using relational databases.

7.1) Limitation of Relational Databases

In current scenario, the disadvantages using banking sectors are:-

1, Abundance of Information

Advances in the complexity of information causes certain problems in the relational databases. Relational databases are made for organizing data by common characteristics. In the case of large Banks, there are lot of information s(complex information's) are to be stored.

2, Structured Limits

Some relational databases have limits on field lengths. At the time of design the database should specify the amount of data you can fit into a field. Some names or search queries are shorter than the actual, and this can lead to data loss.

3, Cost

The relational databases is the expensive of setting up and maintaining database system.

4, Isolated Databases

Complex relational database systems can lead to these database becoming "islands of information" where the information cannot shared easily from one large system to another. In the case of Banking, Finance department used one database and the personnel department used a different database. Getting those databases to "talk" to each other can be a large, and expensive, undertaking, yet in a complex system.

I supposed to suggest, to overcome these disadvantages by introducing Bigtable in Bank's instead of relational databases. Bigtable helps to improve the performance in banking fields and it also provides more security than relational databases. Big table offers several advantages over relational databases.

7.2) Advantages of Bigtable

1) Lower costs:

Google has introduced an interesting pricing strategy for Bigtable by separating the storage and computing needs of organizations. This makes Bigtable a very useful proposition for banking sectors who might need to store large amounts of data over an extended period of time, especially if they only require minimal access and manipulation of that information. This makes it much more cost effective compared to competitors who usually charge for each read or write operation on the data.

2) Performance:

Google is no stranger to high-performance requirements, and as Bigtable has already been used internally, there is not much doubt that it can provide the needful to external business customers as well. Much of the setup and initial storage calculation is done in an instinctive manner, requiring minimal user inputs, that results in saving much timeand effort for new customers.

3) Security:

With large amounts of data, concerns for data security also escalate just as much. Bigtable offers a replicated storage strategy, with algorithms for encryption of data; something that is sure to help allay these concerns. Customers can also bank on Google's expertise in this area, with their long-standing experience of handling the privacy and security of large amounts of data.

4) Maturity:

Due to the simple fact that Bigtable has been used internally for a significant period of time by a data giant like Google, it can promise a high level of stability and maturity to its users. It is not at all comparable to a new and untested product, and might probably score favorably on many fronts when compared to longstanding players in the arena as well. Due to its internal use, customers can also be sure of its continued availability and enhancement.

VIII. Conclusions

Bigtable, a distributed system for storing structured data at Google. Bigtable clusters have been in production use since April 2005 and As of August 2006, more than sixty projects are using Bigtable.I supposed to suggests, Instead of using the relational database s in Banks using Bigtable helps to improve the performance and it helps to store more information's than relational databases.

If new users are sometimes uncertain of how to best use the Bigtable interface, particularly if they are accustomed to using relational databases that support general-purpose transactions. Nevertheless, the fact that many Google products successfully use Bigtable demonstrates that design works well in practice. The process of implementing several additional Bigtable features, such as support for secondary indices and infrastructure for building cross-data-center replicated Bigtables with multiple master replicas.
Bigtables with multiple master replicas.

References

[1] BARU, C. K., FECTEAU, G., GOYAL, A., HSIAO, H., JHINGRAN, A., PADMANABHAN, S., COPELAND, G. P., AND WILSON, W. G. DB2 parallel edition. IBM Systems Journal.

[2] BANGA, G., DRUSCHEL, P., AND MOGUL, J. C. Resource containers: A new facility for resource management in server systems. In Proc. of the 3rd OSDI (Feb. 1999),

[3]AILAMAKI, A., DEWITT, D. J., HILL, M. D., AND SKOUNAKIS, M. Weaving relations for cache performance. In The VLDB Journal.

[4] CHANDRA, T., GRIESEMER, R., AND REDSTONE, J. Paxos made live — an engineering perspective. In Proc. of PODC (2007).

[5]STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proc. of SIGCOMM (Aug. 2001)

AUTHORS

Mrs. G Anitha Krishnan, Master of Computer Applications (MCA) from Noble P.G.College, Osmania University and Bachelor of Science (MPC) from Nizam College (Autonomous), Osmania University.Currently She is working as Asst. Professor at SCMS School of Technology and Management (SSTM), Muttom. Her Research interest lies in the area of Artificial Intelligence

Athira T D is currently pursing Post Graduation in MCA at SCMS School of Technology and Management (SSTM), Muttom, Alwaye, Cochin. Her keen interest lies in the areas of Bigtable.