Research paper based on Bigtable

A Distributed Storage System For Structured Data by Fay Chang, Jeffrey Dean & co.,

Bigtable is a distributed storage system used by Google for storing vast amount of structured data. This research paper is a study of the Bigtable technology, the research orientation given by Richard Schantz and Douglas Schmidt in their paper Middleware for Distributed Systems and the characteristics of ubiquitous computing identified by Tim Kindberg and Armando Fox in their paper on System Software for Ubiquitous Computing.

Grace Ramamoorthy

Student ID: 10278389

11/26/2010

Research paper based on Bigtable

A Distributed Storage System For Structured Data by Fay Chang, Jeffrey Dean & co.,

Technology Used By Bigtable:

Bigtable is a distributed storage system for structured data. Bigtable can handle data that scales to a very large size, even to petabytes, distributed across thousands of servers. Many Google projects such as Google Earth, Google Finance, and Orkut with varied latency requirements and real-time processing use Bigtable to store their data. These applications have asynchronous processes updating the data simultaneously at a very high speed. A read/write of about a million operations per second is what is expected.

Bigtable stores data as a distributed multidimensional sorted map with row, column and timestamp. It places frequently accessed columns together as column families. Storing the timestamp allows multiple versions of the contents to be stored in the same cell and users can access the most recent version or base query on timestamp range. Rows are ordered lexicographically and groups of contiguous rows are stored on same machines as a single tablet for easy access.

Bigtable stores data as a distributed multidimensional sorted map with row, column and timestamp.

Following Google's philosophy, Bigtable is an in-house development designed to run on commodity hardware. Bigtable allows Google to have a very small incremental cost for new services and expanded computing power. Bigtable is built atop Google File System to store data and log files, cluster management system for scheduling jobs, MapReduce for simplified large-scale data processing, and a distributed lock service called Chubby to liaise between the tablets servers that handle the data and the clients.

Chubby is a highly available file server responsible to ensure that there is only one active master, to store the bootstrap location of the Bigtable data, to service tablet servers and to store the schema information.

Bigtable implementation involves one Master server and many tablet servers. The master assigns data tablets which are contiguous rows of data in a table to tablet servers, balances the load and collects garbage. The tablet servers service the clients. The data is stored as tables and each table is split into many tablets based on a range of rows. Table/ tablets are split automatically when the size of the tablet increases or the

load becomes heavy on a tablet. There is no replication of tablets. Each tablet is serviced by only one tablet server. Access to data is in the form of a three level hierarchy. The first level is an address in the chubby to the root directory; the second level is the address of the tablet in the metadata table. The third level is the actual address of the user tablet that contains the data. The traffic on the root directory is regulated by caching the information of the metadata tablet on the client machines and also dedicating one tablet server to service just that metadata tablet. In case of this tablet server going down, the cached data on the clients are used until the metadata tablet is reassigned. If the need arises, it is also possible to replicate this metadata information. The chubby directory keeps track of the tablets assigned to various tablet servers. When the user needs any information, the three-level hierarchical access takes the user straight to the tablet without worrying about the actual physical location. This completely abstracts the path access from the user.

Bigtable And The Schantz And Schmidt Research Orientation:

Looking at the performance of Bigtable over the past few years in search intensive applications with huge data to crunch, we can realize that Bigtable has to some degree overcome the challenges posed by *Schantz and Schmidt in their paper on Middleware for Distributed Systems*. In the section on *challenges and opportunities*, they point out that "*the desirable properties of the system of systems should include predictability, controllability and adaptability with respect to features such as time, quantity of information, accuracy, confidence, and synchronization*".

Bigtable with the help of the secure Chubby locking system has assured users of reliability of service with respect to quantity of information. By restricting the servicing of clients to just the tablet server, Bigtable is able to service clients even in the event of the sudden death of the Master. When the Master comes up again, the Master is able to restore the situation by looking at the directory in the Chubby for tablet server assignments. This ensures that the clients are continuously serviced without any loss of time or resource. Bigtable also provides the users an option to choose their data access remotely or in-memory – This ensures timely response. Caching tablet location information in the client machine is another facility that Bigtable offers to improve performance with respect to time.

Bigtable with the help of the secure Chubby locking system has assured users of reliability of service with respect to quantity of information.

In their paper on Middleware, in the *Research Orientation* section, Schantz and Schmidt propose a concept: "*Multiple system behaviour must be made available based on what*

is best under various conditions". They propose this to ensure that the **middleware abstraction** is well defined. To achieve this, they suggest that information must be gathered regarding user and resource requirements and system conditions. In the Bigtable setup, by allowing users to decide their access preference a balance is reached in performance. Data is then accessed either remotely or in-memory. This will meet the user need optimally. Secondly, the distributed data resides in tablet servers. The master keeps track of the lock position of each tablet server and the Chubby. As and when a tablet server loses connection or removed from the cluster to optimize machine use, the master reallocates the data tablets of that tablet server on other available tablet servers. When users need any data, they will not have to worry about the change in the namespace or the address of the location of the servers, the chubby works through the nitty gritty and the three-level hierarchy and the user is able to get the data from the distributed system. The cluster management system, chubby and the master handle the availability of machines, reallocation of the data, maintaining the current allocation of the data tablets and the list of the active serving tablet servers between them. In case of the tablet server or the master's sudden death, the user is not affected. I feel the middleware abstraction in Bigtable has reached the "*control interoperability*" that Schmidt and Schantz speak about.

The cluster management system, chubby and the master handle the availability of machines, reallocation of the data, maintaining the current allocation of the data tablets and the list of the active serving tablet servers between them.

The I/O operations in Bigtable are **performance tuned**. All writes are queued and written as group jobs on memtables in the memory. These are flushed from time to time when the memtable size reaches a threshold limit and memtables are converted into SSTables which are real files in the GFS. All reads will merge SSTable data with memtable data to get the required data. But these are not visible to the user. All mutation information is stored in log tables.

By addressing network issues, I/O operations, data replication and performance tuning Bigtable optimizes each area to give an overall optimized performance, Bigtable addresses ***QoS requirements*** that Schmidt and Schantz emphasis. According to them, *"Decisions for managing QoS are made at design time, at configuration/deployment time, and /or at runtime."* They talk about "*end-to-end QoS requirement and aggregate requirements"*. Since Bigtable development was not piece-meal but was proprietarily done, an end-to-end optimization is achieved.

Bigtable And The Key Characteristics Defined by Kindberg and Fox:

Kindberg and Fox in their paper on System Software and Ubiquitous Computing, raise two issues: *physical integration and spontaneous interoperation.* According to them *ubicomp must deliver functionality in our everyday world.* One of the challenges they point out is for "*the software to adapt to changing environments, tolerance to routine failures or failure like conditions and security"*. With Bigtable applications such as Google Maps, Google Earth, and Orkut adapting to changing environment and bandwidth is the key as users could potentially use the same app on a laptop or a mobile phone. Bigtable scales well. As regard to ***security in a changing environment*** – all reads and writes are verified in the Chubby for access rights. As regards to *failures* – Chubby failure is a major cause of concern as that's the main arm of the distributed app, but testing have proved that failure due to chubby crash is less than 0.005%.

Bigtable has the potential for spontaneous interoperation as it allows storage and access of information. It works well with other products such as MapReduce and employs the available features in those components without repeating them again. For example when Bigtable works with GFS, it uses the replication facility of GFS and in Google Earth the compression is disabled as the images are already compressed.

Bigtable as a web service will provide an indexed data; APIs for storage and access with a potential to becoming a network service for the planet.

Kindberg and Fox while explaining spontaneous interoperation talk about *bootstrapping, service discovery and interaction.* For Bigtable to achieve this level of interoperation is not far-fetched. The only address a client will require is of the root directory. From then on Bigtable has a good three-level hierarchy for searching data. While explaining *interaction* Kindberg and Fox talk about "*priori knowledge of the methods in an arriving service"*. The interactions according to them can be "*data-oriented*". They say "*Data-oriented interaction is a promising model that has shown its value for spontaneous interaction inside the boundaries of individual environments."* With its existing APIs and excellent infrastructure, Bigtable is an ideal example for such data-oriented interaction. As Dan Farber in his news article puts it, "Bigtable as a web service will provide an indexed data; APIs for storage and access with a potential to becoming a network service for the planet."

As of now each service running Bigtable has its own cluster running Bigtable. There is a move to run a Google-wide Bigtable system service. Bigtable also has **adapted** the content to heterogeneous device types as users can access the data from a Bigtable app using laptop, mobile or any other device.

Another concern Kindberg and Fox express is about **robustness** in a ubiquitous system. *"In spontaneous interoperation, they mention about associations that are gained and lost unpredictably when devices enter or leave an environment".* In Bigtable we provide **data-oriented interaction**. As far as losing service suddenly – due to unavailability of tablet server or a tablet server taken out of the environment- this scenario is handled well in the Bigtable infrastructure. The Master server will track down any unavailable servers and reallocate the tablets to other tablet servers. The scenario of the sudden death of a tablet server without deregistering and causing inconsistencies is again handled by the Master server tracking each tablet server and balancing the load. There is a very small chance of the service being unavailable or lost or causing inconsistencies. Kindberg and Fox talk about "*group communication to rediscover lost resources*". With Bigtable, the Master server detects the status of each tablet server by checking the directory in the Chubby file server as well as by asking the status from the tablet servers. Chubby provides an efficient mechanism to check the status without incurring network traffic.

Next, Kindberg and Fox talk about "***security*** *and* ***trust*****"** in a ubiquitous environment. Chubby is responsible for verifying the authorization of a user performing a read or a write. As we deal with mutation of data, Chubby maintains a list of authorized writers and every write request is validated against this list. Most of the time the list is available in the Chubby client cache due to its frequent access.

With respect to Kindberg and Fox's comment on **physical integration**, the applications listed in the paper do not imply Bigtable's ability to integrate in every day physical environment. However, given Bigtable's potential and a more widespread usage, this could be a possibility soon. For example a smart presentation room where the projector automatically projects all of the previous works of a presenter based on the information collected from a Bigtable cell on the presenter or a doctor's consultation smart pad showing the available treatments for a specific medical condition based on the information collected from a Bigtable cell on the medical condition of a patient are not far-fetched.

In conclusion, most research orientation directions given by Schantz and Schmidt are already incorporated in some way in Bigtable infrastructure. The key characteristics identified by Kindberg and Fox for ubiquitous computing are partly attained by Bigtable design and can be extended to achieve complete physical integration.

References:

- *Bigtable: A Distributed Storage System for Structured Data by Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber*
- *Middleware for Distributed Storage Systems by Richard E. Schantz and Douglas C. Schmidt*
- *System Software for ubiquitous Computing by Tim Kindberg and Armando Fox*
- *http://video.google.com/videoplay?docid=7278544055668715642&q=bigtable#*

- *http://bigtable.appspot.com/*
- *http://andrewhitchcock.org/?post=214*
- *http://glinden.blogspot.com/2006/08/google-bigtable-paper.html*
- *http://news.cnet.com/8301-13953_3-9912172-80.html*