Enfoque UTE
E-ISSN: 1390-6542
enfoque@ute.edu.ec
Universidad Tecnológica Equinoccial
Ecuador

Moscoso Zea, Oswaldo
Megastore: structured storage for Big Data
Enfoque UTE, vol. 3, núm. 2, julio-diciembre, 2012, pp. 1-12
Universidad Tecnológica Equinoccial

Enfoque UTE, V.3-N.2, Dic.2012: pp.1-12

ISSN: 1390-6542

Megastore: structured storage for Big Data

Oswaldo Moscoso Zea[1]

Resumen

Megastore es uno de los componentes principales de la infraestructura de datos de Google, el cual ha permitido el procesamiento y almacenamiento de grandes volúmenes de datos (Big Data) con alta escalabilidad, confiabilidad y seguridad. Las compañías e individuos que usan está tecnología se están beneficiando al mismo tiempo de un servicio estable y de alta disponibilidad. En este artículo se realiza un análisis de la infraestructura de datos de Google, comenzando por una revisión de los componentes principales que se han implementado en los últimos años hasta la creación de Megastore. Se presenta también un análisis de los aspectos técnicos más importantes que se han implementado en este sistema de almacenamiento y que le han permitido cumplir con los objetivos para los que fue creado.

Palabras clave:

Base de Datos NoSql, Megastore, Bigtable, Almacenamiento de Datos.

Abstract

Megastore is one of the building blocks of Google's data infrastructure. It has allowed storing and processing operations of huge volumes of data (Big Data) with high scalability, reliability and security. Companies and individuals using this technology benefit from a highly available and stable service. In this paper an analysis of Google's data infrastructure is made, starting with a review of the core components that have been developed in recent years until the implementation of Megastore. An analysis is also made of the most important technical aspects that this storage system has implemented to achieve its objectives.

Keywords:

NoSql Database, Megastore, BigTable, Data Storage.

1. Introduction

Information plays a leading role for companies when it is managed properly and with the right technologies. It can be a highly differentiating factor for generating a competitive advantage (Porter & Millar, 1985). Advances in science and the proliferation of online services bring an exponential increase in the amount and size of information that companies have to store, process and analyze. This quantity and size of data that companies manage today brought about the trendy concept Big Data (Bryant, Katz, & Lazowska, 2008).

Traditional storage systems experience performance problems when handling disproportionately large data volumes and scaling to millions of users (Baker et al., 2011). That is why information based companies like Google, Yahoo and Facebook among others seek alternative storage options to maintain service levels, scalability and high availability in the handling of information and Big Data that meets the requirements and demands of users.

Google is constantly seeking innovation and excellence in all projects it undertakes (Google, 2012). This quest for continuous improvement has enabled the firm to become one of the pioneers

[1] Universidad Tecnológica Equinoccial, Facultad de Ciencias de la Ingeniería, Quito – Ecuador (omoscoso@ute.edu.ec).

in developing their own distributed data storage infrastructure. This infrastructure has evolved over time. This evolution has led to Megastore a storage system based on a non-traditional architecture and developed to meet the current requirements of interactive online services (Baker et al., 2011).

One of the motivations for writing this paper is Big Data. The term Big Data is not just a technological buzzword anymore and has become a term that requires further attention due to the information explosion. Nowadays the management of this universe of data is not only an issue that should be of concern for Search Engine companies like Google or Yahoo but for all companies that capture information of value to the business (AMD Inc., 2010).

It is clear then, that managing Big Data is no longer just for Google, but Google may provide their data storage infrastructure with platforms as Google App Engine so that companies can save important costs when developing web solutions that require a scalable and reliable data storage (Severance, 2009).

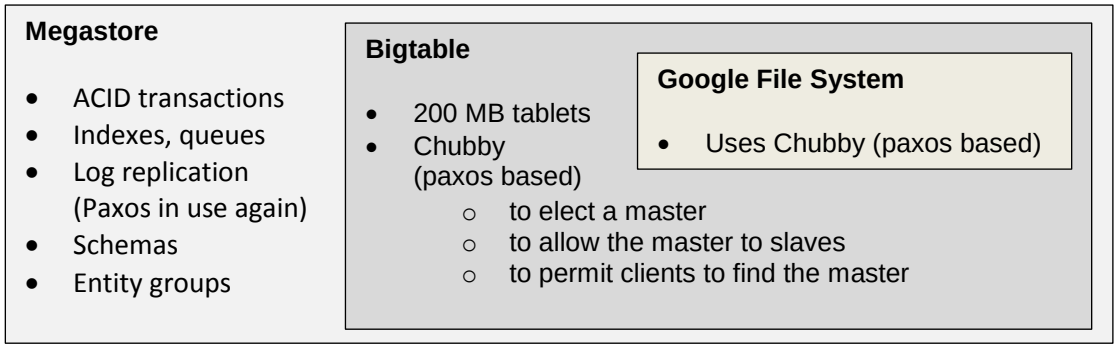
It is also important to mention that besides being a pioneer in managing Big Data from the creation of Google File System in 2003 and including High Replication Megastore in 2011. Megastore provides a highly reliable service with the new ability to withstand datacenter outages. At January 2012 Megastore has more than 100,000 apps running and nearly 0% downtime(Ross, 2012). High Replication Megastore and the most important components of Google's infrastructure are described in this paper.

2. Google Data Store Components

In order to better understand the present work, it is important to describe the technologies that work as the foundation for Google's Data Storage and that allow it to achieve the objectives for which it was created, these are: Google File System, Bigtable and Megastore. Figure 1 shows the 3 components.

2.1. Google File System

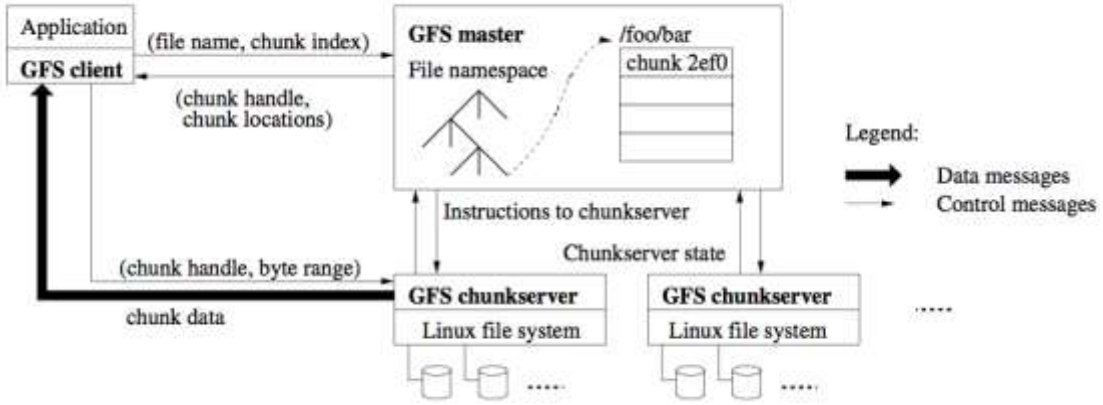
Google File System (GFS) is a distributed file system developed by Google for its own use. It was implemented over Linux for managing distributed data applications in order to achieve scalability and performance(Ghemawat, Gobioff, & Leung, 2003). It was designed largely by observing applications workloads and the technological environment inside the company. One of the key points in the design of the system is that failures occur regularly and become a norm rather than an exception (Ghemawat et al., 2003).



(Source: based on Bahadir, 2011)

Figure 1. Google Data Infrastructure

GFS architecture is depicted in Figure 2. A typical GFS Cluster consists of one master, multiple chunkservers that run in Linux commodity machines. This cluster is accessed by many clients and files are divided in 64MB chunks which are identified by a 64bit chunk handle. These chunks are replicated in at least three different replicas with the purpose of reliability. The Master continuously communicates with chunkservers to obtain state and send instructions about chunk location and replication. Clients do not read and write directly to the master instead they ask the master which replica they should use. Then clients interact with the replica without further intervention of the master for a defined time frame.



(Source: Ghemawat et al., 2003)

Figure 2. Google File System Architecture

2.2. Bigtable

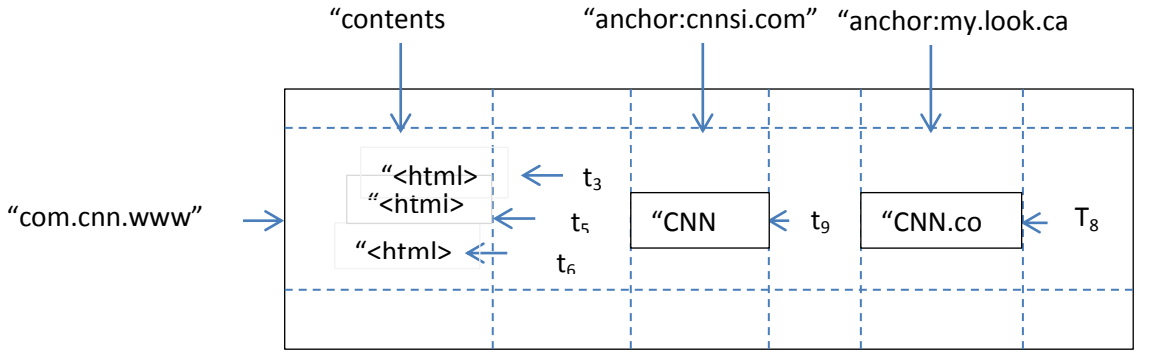
Bigtable is a NoSql distributed storage system which was created in order to scale huge amounts of data across thousands of servers while providing high availability and high performance. It uses GFS to store data files and logs (Chang et al., 2006) and can use MapReduce (Dean & Ghemawat, 2004) for processing intensive parallel computations.

The data model is based on a key-value multidimensional structure that contains rows, column and timestamps. Columns can be grouped in sets called column families. These column families along with timestamps can be added at any time. The latter allow managing multiple versions of the

same data more easily. In Bigtable each table is initially formed of one tablet which is a data structure that can handle the range of a row that reaches a size of up to 200MB, as the table grows, new tablets are created and divided.

Bigtable is in production since April 2005 (Chang et al., 2006), but one of its biggest drawbacks is the little known interface which makes it difficult for developers to use. However the advantage is that implements a simple model that allows it to achieve high scalability, availability and fault tolerance.

In Figure 3 an example is shown of a Bigtable row, with column families and timestamps.



(Source: Chang et al., 2006)

Figure 3. A part of an example table that stores Web pages.

2.3. Megastore

Megastore is the storage system built by Google on top of Bigtable that supports typed schemas, multi-row transactions, secondary indices, and the most important which is consistent synchronous replication across data centers (Baker et al., 2011).

In order to meet the requirements of online services currently Megastore allows a balance between the benefits of a NoSql data storage such as scalability and the advantages of a relational data base management system (RDBMS). A RDBMS is a program that lets users create and manage a relational database with features such as friendly user interface (Baker et al., 2011). Megastore architecture is based on Bigtable with management capabilities similar to those of a traditional RDBMS; this along with the optimal use of synchronous replication algorithm has allowed Google to offer a monthly uptime greater than 99.95% as a part of the service level agreement to Google App Engine customers. Discussion of important technical details of Megastore will be made in Section 3.

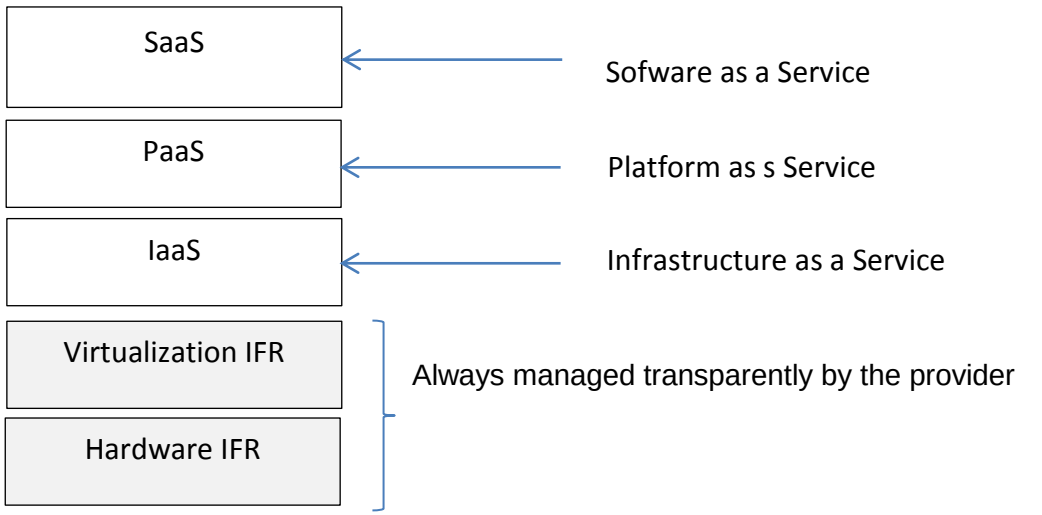
2.4. Google App Engine

Google App Engine (GAE) is a tool for building web applications using Python or JAVA and Google's Infrastructure (Google Inc., 2011a). Once the main elements of Google's data storage have been described, I believe important to give one concrete example of an application that uses

the infrastructure. This may help readers to visualize the environment in which Megastore operates and provide a better point of view to people or companies who want to use Google's data storage. For this reason, an introduction of cloud computing will be made.

2.4.1. Cloud Computing

Is the new Internet paradigm based on a series of other technologies such as cluster computing and grid computing. Figure 4 shows an overview of different models.



(Source: Haselmann & Vossen, 2010)

Figure 4. Visual Overview of different models in cloud computing.

Its most important feature is that users can use a specified infrastructure as services provided in different levels of abstraction, this infrastructure can scale as needed in minutes and users pay only what they use (Vaquero, Rodero-merino, Caceres, & Lindner, 2009).

NIST definition will be used in order to describe Cloud Computing.

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell & Grance, 2011).

To establish the relationship between cloud computing and the data storage that is being analyzed GAE is a good example of a cloud.

2.4.2. GAE as a Platform as a Service

A Platform as a service offers users the opportunity to develop their own applications in the cloud by using the provider's infrastructure. The infrastructure comprehends software tools, storage systems and hardware. All the tasks for managing the infrastructure are performed by the service

provider. The provider puts certain conditions, such as fixed programming languages and interfaces (Haselmann & Vossen, 2010).

GAE comprises all of the features of a platform as a service in the cloud. It allows developers to deploy their own web applications by providing an integrated development environment with Python and JAVA as the programming languages. The interface allows developers the possibility to use and interact with Google's hardware infrastructure and data storage.

The data storage infrastructure as explained in Section 2 includes GFS, Bigtable and Megastore. Infinite storage, unlimited resources and high availability are guaranteed through these data storage systems. Applications developed with GAE can benefit from the most important features of Megastore and use high replication across data centers as a default option for data storage. Moreover Google is offering a migration tool for users that have applications running with the Master Slave data storage option. With the use of GAE companies can benefit from cost savings by implementing scalable Big Data projects without incurring huge infrastructure expenditures (Severance, 2009).

2.4.3 Google's Hardware Infrastructure.

GAE uses Google's Data Centers to perform its operations. These data centers are located in different continents and are used as giant data processing centers with thousands of servers, offering different levels of security to protect data, reducing sourcing of materials, reusing components and recycling (Google Inc., 2011b).

GAE uses Linux machines with GFS as their file system. GFS has superior advantages over a traditional file system and allows handling files that can reach hundreds of terabytes. The primary objective of GFS is to achieve unlimited storage.

3. Megastore

Megastore is a storage infrastructure built by Google on top of Bigtable. It is used by over 100 productive applications. Megastore handles 3 billion writes and 20 billion read transactions daily. Megastore combines the features of a traditional RDBMS that simplifies the development of applications with the scalability of NoSql datastores to satisfy the requirements of today's cloud services. The analysis of Megastore in this chapter builds on Google's Megastore Paper (Baker et al., 2011).

3.1. Replication among distant Data Centers

Having a schema with replicas between servers in the same physical data center improves availability. The reason for this is that the failures and shortcomings of hardware can be overcome by moving the workload of one server to other server. Nevertheless, there are different kinds of failures that can affect the whole data center such as network problems or failures caused by the

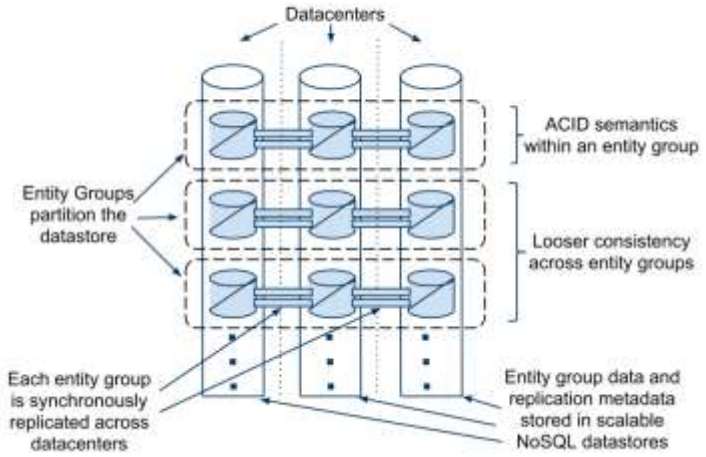
power and cooling infrastructure. This is the main reason why it is important to replicate data across geographically distributed data centers.

Megastore uses a synchronous replication strategy that implements Paxos. Paxos is a fault tolerant algorithm that does not require a specific master as the log replicator. The Paxos algorithm consists in three steps. First a replica is selected as a coordinator, this coordinator then sends a message to the others replicas, and these in turn acknowledge the message or reject it. Finally, when the majority of replicas acknowledge the message a consensus is reached and the coordinator sends a commit message to replicas (Chandra, 2007).

Megastore's engineering team made adjustments to the original algorithm in order to provide ACID transactions and to improve latency. One of these adjustments is the use of multiple replicated logs. This also extends the possibility of local reads and single roundtrip writes.

3.2. Partitioning data and concurrency

In order to improve availability and at the same time maximize throughput it is not enough to have replicas in geographically different locations. Partitioning data within a data store is the Google´s answer to address this issue. Partitions are done into so called entity groups which define boundaries for grouping data in order to achieve fast operations. Each entity group has its own log and it is replicated separately which helps to improve replication performance. Data is stored in a NoSql datastore Bigtable and there is ACID semantics within entity groups as seen in Figure 5.



(Source: Baker et al., 2011)

Figure 5. Scalable Replication

Single ACID transactions are guaranteed within an entity group using Paxos algorithm for replication of the commit record. Transactions spanning more than one entity group are done with a two phase commit or asynchronous messaging communication using queues. This messaging is between logically distant entity groups not between replicas in different data centers.Every change within a transaction is written first into the entity group log then changes are applied to data. One

of the important features of Bigtable which was previously mentioned is Timestamp. Timestamp is the core element for concurrency that allows users to perform read and write operations without blocking each other. Values are written at the timestamp of the transaction and readers use the data of the last committed timestamp, sometimes when latency requirements are important, there is a possibility of inconsistent reads which allows users to read the values directly without taking into account the log state.

3.3. Megastore and Bigtable

Megastore uses Bigtable for data and log storage operations within a unique data center. Applications can control placement of data by selecting Big Table instances and specifying locality. In order to maximize efficiency and minimize latency, data is placed in the same geographic location of the user. On the other hand, replicas are placed in different data centers but in the same geographic location from which data is accessed most. Furthermore entity groups within the same data center are held in continuous ranges of Bigtable. A Bigtable column name is the concatenation of megastore table name and property name as seen in Figure 6.

Row key	User. Name	Photo. time	Photo. Tag	Photo. _url
101	John			
101,500		12:30:01	Dinner, Paris	…
101,502		12:15:22	Betty, Paris	…
102	Mary			

(Source: Baker et al., 2011)

Figure 6. Sample Data layout in Bigtable

3.4. Megastore Data Model Design

One of the most important goals of Megastore is to help developers rapidly build scalable applications. Megastore provides some features that are similar to a traditional RDBMS. For example the Data Model is thought to be a strongly typed Schema. This Schema can have a set of tables each with a set of entities, which in turn have a set of strongly typed properties that can be annotated as required, optional or repeated.

Tables in megastore can be labelled as entity group root tables or child tables. The child tables have a foreign key (ENTITY GROUP KEY) to reference the root table, see Figure 7. Child entities reference a root entity in the respective root table. An entity group consists of the root entity and all of the child entities with the root references.

It is important to point out that Bigtable storage contains one single Bigtable row for each entity group; in Figure 7 for example Photo and User are seen as different tables which share the user_id property. The annotation IN TABLE tells Megastore that the data for these tables should be stored in the same Bigtable. Megastore also supports two types of secondary indexes, local indexes used to find data within an entity group and global indexes used to find entities across entity groups.

```
CREATE SCHEMA PhotoApp;
CREATE TABLE User {
  required int64 user_id;
  required string name;
  } PRIMARY KEY(user_id), ENTITY GROUP ROOT;

CREATE TABLE Photo {
  required int64 user_id;
  required int64 photo_id;
  required int64 time;
  required  string full_url;
  optional string thumbnail_url;
  repeated string tag;
 } PRIMARY KEY(user_id, photo_id),
    IN TABLE USER
    ENTITY GROUP KEY (user_id, time) REFERENCES User;

CREATE LOCAL INDEX PhotosByTime
   On Photo (user_id, time);

CREATE GLOBAL INDEX PhotosByTag
   On Photo (tag) STORING (thumbnail_url);
```

(Source: Baker et al., 2011)

Figure 7. Megastore Photo Schema Example

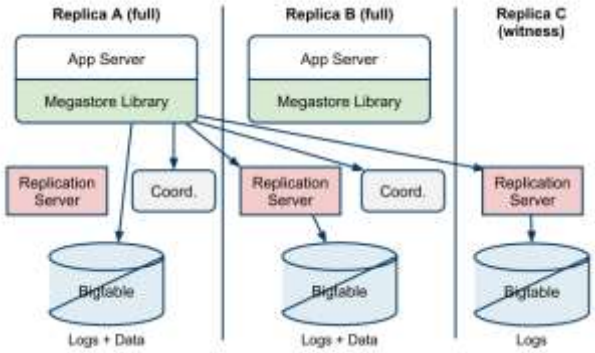
3.5. High Replication Data Store

At the heart of Megastore is the synchronous replication algorithm which allows reads and writes to be performed from any replica while maintaining ACID transactions. Replication is done for each entity group by replicating their transaction log to all replicas. One of the most important features added by Megastore is that its design is not based on a Master Slave approach; this enhances flexibility and fault recovery.

Using distant replicas over a wide geographic area without the need for a master allows faster consistent reads because writes are synchronously replicated to all replicas. This keeps them up to date all the time and allows local reads to minimize latency. A coordinator is introduced for each data center to keep track of all local replicas and allows a replica with a complete commit state to serve local reads. If a write fails in one replica it is not considered committed until the group's key is removed from the coordinator.

Another important feature added to Paxos algorithm in megastore are called leaders, which are replicas that prepare the log position for the next write. Each successful write includes a prepare message granting the leader the right to issue accept messages for the next log position. This improves latency since a writer must communicate with the leader before submitting values to other replicas.

There is another element called witness replica which is introduced with the purpose of improving decision of consensus among entities. It is used when there are not enough replicas to form quorum. A witness replica can acknowledge or reject a value without storing data. This decreases storage costs while improving communication when failing to acknowledge a write. Figure 8 shows Megastore architecture.



(Source: Baker et al., 2011)

Figure 8. Megastore Architecture

4. Conclusions

Google has succeeded with the implementation of Megastore in many aspects. This has allowed the company to offer an efficient service with great performance and throughput. Furthermore Megastore has achieved the goals for which it was created these are scalability, consistency and availability; two major factors have contributed to this. The RDBMS like data model and API offered by Google App Engine and the consistent synchronous replication across distant data centers.

High replication algorithm based in Paxos used in the design of the system has performed efficiently, this is one of the reasons that High Replication Data store Megastore is the default option nowadays for GAE. Google is encouraging every GAE user to migrate from the traditional master slave design into the successful HRD, and is providing tools for this purpose.

One of the shortcomings that critics argue is that the Data Storage is over engineered and that entire infrastructure is obsolete since is based on old systems with more than 10 years of use (Prasanna, 2011).

Since more and more companies nowadays need to have an infrastructure that can support processing and storage of Big Data. GAE and Megastore are great alternatives for constructing applications without having to be concerned of infrastructure's technical problems. Google's goal at the same time is to maintain its competitive advantage in this field and to generate strategies to enhance its data storage infrastructure and to attract potential customers to use GAE.

Bibliography

AMD Inc. (2010). Big Data — It ' s not just for Google Any More. *AMD*. Retrieved from http://sites.amd.com/us/Documents/Big_Data_Whitepaper.pdf

Bahadir. (2011). Megastore: Providing Scalable, Highly Available Storage for Interactive Services. Retrieved from http://cse708.blogspot.de/2011/03/megastore-providing-scalable-highly.html

Baker, J., Bond, C., Corbett, J. C., Furman, J. J., Khorlin, A., Larson, J., Jean-michel, L., et al. (2011). Megastore : Providing Scalable , Highly Available Storage for Interactive Services. *CIDR 2011*, 223–234.

Bryant, R. E., Katz, R. H., & Lazowska, E. D. (2008). Big-Data Computing : Creating revolutionary breakthroughs in commerce , science , and society Motivation : Our Data-Driven World. *Library*.

Chandra, T. (2007). Paxos Made Live - An Engineering Perspective. *PODC '07* (pp. 1–16). New York, NY, USA.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., et al. (2006). Bigtable : A Distributed Storage System for Structured Data. *OSDI 2006*.

Dean, J., & Ghemawat, S. (2004). MapReduce : Simplified Data Processing on Large Clusters, 1–13.

Ghemawat, S., Gobioff, H., & Leung, S. (2003). The Google File System. *Architecture*.

Google. (2012). Corporate Information- Our Philosophy. *Google*. Retrieved from http://www.google.com/corporate/tenthings.html

Google Inc. (2011a). What is Google App Engine. Retrieved from https://developers.google.com/appengine/docs/whatisgoogleappengine

Google Inc. (2011b). Google Data Center. Retrieved from http://www.google.com/about/datacenters/#

Haselmann, T., & Vossen, G. (2010). Database-as-a-Service für kleine und mittlere Unternehmen. Münster. Retrieved from http://dbis-group.uni-muenster.de/

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*.

Porter, M., & Millar, V. (1985). How information gives you competitive advantage. *Harvard business review*, 149–160. Retrieved from http://www.mendeley.com/research/copyright-2001-all-rights-reserved-8/

Prasanna, D. R. (2011). Waving Goodbye0. Retrieved from http://rethrick.com/#waving-goodbye

Ross, M. (2012). Happy Birthday High Replication Datastore. Retrieved from http://googleappengine.blogspot.de/2012/01/happy-birthday-high-replication.html

Severance, C. (2009). *Using Google App Engine*. *O'Reilly Media Inc.* 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media Inc. Retrieved from www.oreilly.com

Vaquero, L. M., Rodero-merino, L., Caceres, J., & Lindner, M. (2009). A Break in the Clouds : Towards a Cloud Definition. *Computer Communication Review*, *39*(1), 50–55.