An Investigation of NoSQL Technologies Beneath Cloud Computing

Ruisheng Fu
21421190
alex4814@zju.edu.cn

Guanyu Guo
21421189
guanyguo@gmail.com

Chen Sui
21421183
1456587116@qq.com

January 5, 2015

Abstract

A lot of changes in database management system has been made since the inception of cloud computing. Such critical and increasing needs within cloud computing as scalability, elasticity and processing a huge amount of data can be fulfilled by the NoSQL databases as opposed to RDBMS[1]. In this report we have dived into several primary NoSQL databases used in leading cloud vendors, summarizing and discussing detailed techniques as well as analysis with comparisons.

1 Introduction

Cloud computing has been an evolving computing terminology that is very much in the public eye. It is its responsibility to manage and group remote servers that allow data storage and online access to various services.

In the field of computing, the various advancements and aspects are key evidences that explain the reason why higher priorities are given to scalability, resource utilization and power savings rather than consistency, with respect to data storage. The traditional RDBMSs offer functionalities like clustering, synchronization (always consistent), and structured querying. However, what classical RDBMS could not do so well is to scale[2] to heavy workloads compared to NoSQL databases. As non-relational databases have cropped up both inside and outside the cloud, there comes heated debate around SQL and NoSQL databases.[1]

The two solutions, manual sharding and caching, applied to classical SQL databases are not adequate enough to cope up with the modern web applications, thus agility can't be achieved. On the contrary, NoSQL databases are designed to handle such sort of problems. In those applications where high availability, speed, fault tolerance or consistency are needed, NoSQL is the choice, in that it is designed to scale out to provide elasticity and to be highly available. The misleading term *NoSQL* should be seen as the definition[12] that is "Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable", and is mostly translated with "Not only SQL".

In this report, we firstly examine several major NoSQL databases implemented and used in cloud vendors like Google, Amazon, and Yahoo, along with description about the main ideas of each design. Afterwards, analysis towards different NoSQL databases and we present benchmarking of top NoSQL as a visualizing comparison. Finally, a brief summarization is included in conclusion and further studies as well as challenges are discussed.

2 Major Streams

Currently there are approximately 150 NoSQL databases categorized by data models into a bunch

[1] Relational Database Management System

[2] Scaling, in a google sense, means that an application runs on small commodity PC hardware, but supports essentially unbounded load as more PC's are added.

of classes.[12] We review certain amount of prevailing NoSQL databases through several aspects, including data model, partitions, availability, and consistency.

Bigtable[4] designed by Google describes the well-known data model, which gives clients dynamic control over data layout and format and has successfully provided a flexible, high-performance solution for many Google products. The Amazon Dynamo[5] presents a highly available key-value storage system that some of Amazon's core services use to provide an "always-on" experience. Cassandra[9] concludes design model and implementation part and claims that it is designed to achieve high write throughput and without sacrificing read efficiency, which is a combination of Bigtable and Dynamo. [13] describes the motivation of PNUTS and table storage and replication layers, and at the end presents performance analysis with experimental results. Paper [10] pictures the architecture of Dremel, and explains how it complements with Map/Reduce-based computing. It also present a novel columnar storage representation for nested records and discuss experiments on few-thousand node instances of the system. [15] makes a brief introduction to MongoDB with an installation guide, and mainly focus on the performance benchmark.

2.1 Data Model

The major data models adopted we are going to investigate are *Wide Column Store* (Column Families), *Document Store*, and *Key Value Store*, whereas the minor ones (Graph Databases, Object Databases, etc.) are beyond our scope of discussion in this report.

Column Family Store Columnar databases are logically similar to tabular databases. The difference is that the data are column-wise stored and retrieved.

Bigtable[4] is a kind of sparse, distributed, persistent sorted map introduced by Google. It is indexed by three elements: row key, column key, and timestamp like this:

$$(row : string, column : string, time : int64) \to string$$

Each value in this map is an uninterpreted array of bytes. The row keys which is used to maintains data in lexicographic order are arbitrary strings. Column key consist of family and qualifier. Column keys are grouped into sets called column families, which form the basic unit of access control. Besides, Bigtable can contain multiple versions of each cell by indexed timestamp.

Similarly, Casandra comes under column family. Columns in Cassandra are grouped together very much similar to what happens in the Bigtable system. Cassandra[9] exposes two kinds of columns families, *Simple* and *Super* column families. Super column families can be visualized as a column family within a column family. Casandra also allows columns to be sorted either by time or by name, which is often exploited by different applications.

Document Store A document-oriented database eschews the table-based relational database structure. MongoDB is the well-known member of the family. In general, it stores business subjects in the minimal number of documents instead of breaking it up into relational structures[6] in favor of JSON-like formats with dynamic schemas.[15] This flexibility facilitates the mapping of documents to an entity or an object in MongoDB, in which there are two tools to allow applications to represent relationships between data: *references* and *embedded documents*.[11]

Key Value Store In key-value store, object consists of (K, V) pairs. We can store and retrieve objects based on their key. Dynamo is Amazon's key-value storage system that provides an "always-on" experience.

2.2 Availability

As server downtime implies lost revenue, high availability is the key factor to sustain services. To achieve this goal at a high level, various methods has emerged and implemented.

Replication Replication is one way to ensure consistency between redundant resources, to improve reliability, fault-tolerance, or accessibility.

There are two types of replication supported in MongoDB: *master-slave* and *replica sets.*[15] The latter works the same as the former, except that it is possible to elect a new master if the original master went down.

PNUTS does not rely on log or archive data. Instead, it use a guaranteed delivery pub/sub mechanism to act as a redo log, replaying updates which are lost before being applied to disk due to failure. It replicate the data to multiple regions providing additional reliability, obviating the need for archiving or backups.

Unlike HBase, Cassandra uses a coordinator node in charge of the replication of the data items and locally stores each key within its range. It also provides three replication policies: *Rack Unaware*, *Rack Aware*, and *Datacenter Aware*. For "Rack Unaware" replication strategy, the con-coordinator replicas are chosen by picking certain amount of successors[3] of the coordinator on the ring. The rest two strategies involve a leader node elected by Zookeeper.[7]

Replications of Dynamo[5] resembles Cassandra. It replicates its data on multiple hosts, Each data item is replicated at N hosts using the same consistent hashing.

Failure Detection Failure detection is a mechanism by which a node can locally determine if any other node in the system is up or down. It is vital to support cluster membership.

Dynamo uses hinted hand-off to ensure that read and write operations are not failed due to temporary node or network failures. Applications that need the highest level of availability can set W to 1, which ensures that a write is accepted as long as a single node in the system has durably written the key it to its local store.

In Cassandra, a modified version of the Φ Accrual Failure Detector is applied. The main idea of the detector is to emit a value representing a suspicion level, rather than a boolean value, for each of monitored nodes. All that the value Φ conveys is the likelihood that we will make a mistake. [9] also claims that AFD is good in both accuracy and speed and adjust well to network or load conditions.

Bigtable relies on a highly-available and persistent distributed lock service called Chubby[3]. 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.

2.3 Partition

Due to huge amount of data across applications, it needs to partition the data to distribute it over a cluster. There are two main approaches for partitioning: *range partition* and *hash partition*. The ability to dynamically partition the data over nodes ensures scaling incrementally.

Range What range partition does is to order records lexicographically based on keys and divide it according to the ordering result.

Besides the column family described in section 2.1, Bigtable maintains data in lexicographic order by row key. The row range for a table is dynamically partitioned. Each row range is called a tablet, which is the unit of distribution and load balancing. As a result, reads of short row ranges are efficient and typically require communication with only a small number of machines. In this manner, clients can exploit this property by selecting their row keys so that they get good locality for their data accesses.

Hashing By hashing, we mean that hash records based on key to a linear space and then divide space among different servers.

Dynamo's partitioning scheme relies on *consistent hashing*[4] to distribute the load across multiple storage hosts. Figure 1 shows the picture of the ring.

[3] Rigorously, $N-1$ replicas are picked where N is the replication factor.

[4] In consistent hashing, the output range of a hash function is treated as a fixed circular space or "ring". Each node in the system is assigned a random value within this space which represents its "position" on the ring.

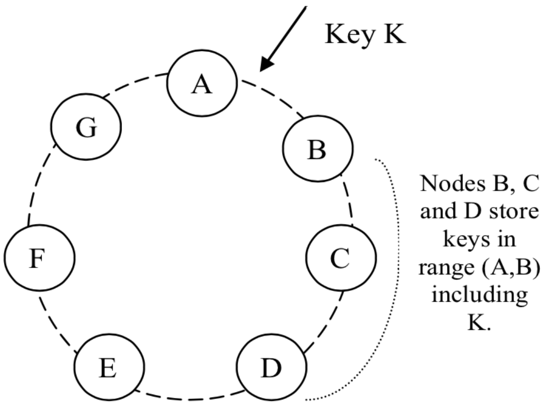


Figure 1: Partitioning and replication of keys

Cassandra also partitions data across the cluster using *consistent hashing* but uses an order preserving hash function to do so. In [9], it points out the challenges (non-uniform data and load distribution) the basic consistent hashing[8] algorithm is facing, and adopt the latter of two suggested ways in [14] to address these issues, because it makes the design and implementation tractable and helps to make choices about load balancing.

Combining these two, PNUTS allows applications to declare tables to be hashed or ordered[5], supporting both workloads efficiently.

Sharding In addition to the two ways above, it is necessary to mention *sharding*. Sharding is a method for storing data across multiple machines. Automatic sharding is one feature supported by MongoDB. The administrator only has to define a sharding key for each collection. In such an environment, the clients connect to a special master node which analyses the query and redirects it to the appropriate node(s). To avoid data losses, every logical node can consist of multiple physical servers which act as a replica set. Thus it is also possible to use Map/Reduce to work on the available data set having a very good performance.

2.4 Consistency

In Dynamo, the consistency is maintained by a quorum-like technique and a decentralized replica synchronization protocol.[5] Dynamo provides *eventual consistency*, which allows for updates to be propagated to all replicas asynchronously.

The Cassandra system relies on the local file system for data persistence. Since Cassandra borrows from Bigtable, the process which runs in the background to collate such files[6] into one is very similar to the compaction process that happens in the Bigtable system. Meanwhile it dedicates a disk on each machine for the commit log to maximize disk throughput.[9]

PNUTS provides a consistency model that is between the two extremes of general serializability and eventual consistency. It provide per-record timeline consistency: all replicas of a given record apply all updates to the record in the same order. An example sequence of updates to a record is shown in Figure 2. A read of any replica will return a consistent version from this timeline, and replicas always move forward in the timeline.

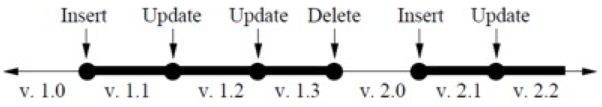


Figure 2: PNUTS Consistency Model

3 Performance

Bigtable[4] has set up a Bigtable cluster with N tablet servers to measure the performance and scalability of Bigtable as N is varied. N client machines generated the Bigtable load used for these tests. The result is show as Figure 3, from which implies slowest random read but fast scan since the tablet server

[5] Hashing is fully supported while order range is claimed as their future work.

[6] Typical write operation involves a write into a commit log for durability and recoverability and an update into an in-memory data structure. The write into the in-memory data structure is performed only after a successful write into the commit log. Over time many such files could exist on disk.

can return a large number of values in response to a single client RPC.

Cassandra system tests two kinds of search features: (a) term search (b) interactions.[9] In order to make the searches fast Cassandra provides certain hooks for intelligent caching of data, thus searches (read performance) are dealt within average of 15.69ms and 18.27ms for interactions and term search respectively.[7] However, it doesn't provide any other results.

[13] runs a series of experiments to evaluate the performance of PNUTS. Its performance metric mainly focus on the average request latency. They compared the performance of the hash and ordered tables and get useful lessons about how to improve system. From these results we can see that the performance of ordered tables is better than by hashing.

MongoDB also have done lots of testings. It compares with Postgres and runs much faster when doing simple inserts, since MongoDB does not use transactions or ensures durable writes. A second experiment turns out that MongoDB is really not good at tags search over a lot of data objects.[8] Finally, MongoDB also loses in geospatial queries. Thus, MongoDB is helpful when you need schema-free, or just simple CRUD operations databases.

4 Limitations

Though Dynamo, Cassandra, MongoDB claims to provide scalability, high performance, and wide applicability, there are serveral drawbacks or challenges they currently face.

These systems is each optimized for a different point in the design space. To our knowledge no publications describing support for geographic replication, secondary indexes, materialized views, and hash-organized tables in Google's Bigtable. Dynamo does indeed provides geographic replication through a gossip mechanism, but its eventual consistency model is not suffice for certain applications, and it does not support ordered tables.

Both PNUTS and Dynamo are distributed data storage systems and are used as scalable back-ends for the various online applications and services of Yahoo and Amazon respectively. Both of these systems have some common features and design considerations and are yet distinct with regards to their architectures and implementations. In order to manage the state of services that require high reliability and need tight control over the trade offs between availability, consistency, cost-effectiveness and performance, Dynamo sacrifices consistency under certain failure scenarios.

As for MongoDB, transactions are not directly supported. Other limitations includes not supporting full single server durability which means you need multiple replications to avoid data losses if one server suffers a power loss or crash. Another drawback is the fact that it uses much more storage space for the same data. Because, as opposed to relational databases, every document can have different keys the whole document has to be stored, not only the values. That's why it is recommended to use short key names.

Other large scale distributed storage systems include SimpleDB services, and Microsoft's CloudDB initiative, but there is little information publicly available about the architecture of these systems.

5 Chanllenges

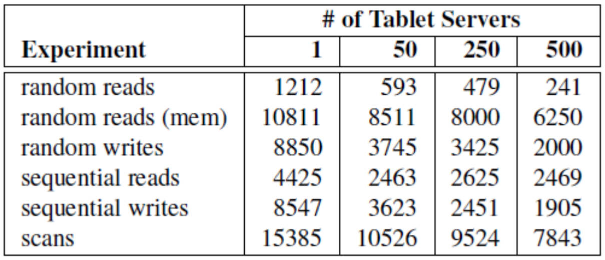
Future works can be driven by these drawbacks or missed components in NoSQL designs. Following are some of them.

Indexes and Materialized Views One important ability most NoSQL databases need is to process query in great efficiency. Unfortunately most of them do not implement secondary indexes and materialized views. The maintainer has the magic to capture stream of updates and generate the counterpart. It is necessary to examine the semantic of answering queries using indexes and views.

Bundled Updates Consistency is diminished in most NoSQL databases. However, needs for an extension of consistency guarantees is on demand. Bun-

[7] The system currently stores about 50+TB of data on a 150 node cluster.

[8] Using sharding may improve its performance.



Experiment	# of Tablet Servers			
	1	50	250	500
random reads	1212	593	479	241
random reads (mem)	10811	8511	8000	6250
random writes	8850	3745	3425	2000
sequential reads	4425	2463	2625	2469
sequential writes	8547	3623	2451	1905
scans	15385	10526	9524	7843

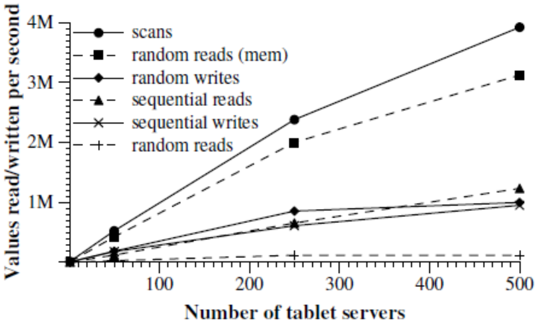


Figure 3: Number of 1000-byte values read/written per second. The table shows the rate per tablet server; the graph shows the aggregate rate.

dled updates provides atomic, no isolated updates to multiple records. In other words, all updates in the bundle are guaranteed to eventually complete, but other transactions may see intermediate states resulting from a subset of the updates.

The challenges lie in how to ensure the timeline consistency guarantees when the updates in the bundle are asynchronously and independently applied, and to provide a convenient mechanism to notify clients when all updates in the bundle have completed.

Batch-Query Processing It is helpful for NoSQL database to serve as a data store for batch and bulk processing.[9] Further investigation should be made on how a scan-oriented bulk workload interacts with a seek-oriented serving workload before doing such improvements.

6 Conclusion

The existence of NoSQL has a great deal with Brewer's (CAP) Theorem.[2] What he said was there are three core systemic requirements that exist in a special relationship when it comes to designing and deploying applications in a distributed environment, and they are *Consistency*, *Availability* and *Partition Tolerance*. The theorem tells us that you can only guarantee two out of three, and that is real and evidenced by the most successful websites as we summarized in sections 2.

Of the systems we covered most favor AP sacrificing C. As cloud computing inevitably involve distributed framework, P is almost a must. Availability is also a necessity when comes to web applications, while consistency is not needed in many cases. However, it does not mean consistency is not important. NoSQL databases still provide "eventually consistent" instead as Dynamo and Cassandra does. Transactions are usually not supported, since NoSQL is related to BASE rather than ACID.

The major goal of NoSQL databases is almost the same: to provide scalable and robust solutions for load balancing, membership and failure detection, failure recovery, replica synchronization, overload handling, state transfer, concurrency and job scheduling, request routing, system monitoring and alarming, and configuration management. On the other hand, different applications vary in requirements of services. Hence, when choosing between SQL and NoSQL, tt really depends on the needs of your application. For web applications that have light querying, key/value stores are very useful. For enterprise databases where reporting is typically very heavy, relational databases fit better.

[9] Except Map/Reduce

References

[1] Tony Bain. Is the Relational Database Doomed? http://readwrite.com/2009/02/12/is-the-relational-database-doomed.

[2] Julian Browne. Brewer's CAP Theorem, 2009. http://www.julianbrowne.com/article/viewer/brewers-cap-theorem.

[3] Mike Burrows. The Chubby lock service for loosely-coupled distributed systems. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation SE - OSDI '06*, pages 335–350, 2006.

[4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. In *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*, pages 205–218, 2006.

[5] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's Highly Available Key-value Store. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, volume 41, pages 205–220, 2007.

[6] Steve Hoberman. *Data Modeling for MongoDB.* Technics Publ., 2014.

[7] Patrick Hunt, Mahadev Konar, FP Junqueira, and Benjamin Reed. ZooKeeper: Wait-free Coordination for Internet-scale Systems. *USENIX Annual Technical . . .*, 8:11–11, 2010.

[8] David Karger, Tom Leightonl, Daniel Lewinl, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 654–663, 1997.

[9] Laksham Avinash and Prashant Malik. Cassandra: a decentralized structured storage system, 2010.

[10] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. Dremel: Interactive Analysis of Web-Scale Datasets. In *36th International Conference on Very Large Data Bases*, pages 330–339, 2010.

[11] MongoDB Inc. Data Modeling Introduction.

[12] NoSQL. NOSQL Databases, 2012. http://nosql-database.org/.

[13] Adam Silberstein, Brian F. Cooper, Utkarsh Srivastava, Erik Vee, Ramana Yerneni, and Raghu Ramakrishnan. PNUTS: Yahoo!'s Hosted Data Serving PLatform. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, page 765, 2008.

[14] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 11:17–32, 2003.

[15] Rico Suter. MongoDB: An Introduction and performance Analysis. In *Seminar Thesis, Rapperswil*, 2012.