

Data Management in Cloud: Study of Existing Systems And Future Opportunities

By

Pooja Sharma, Subhash Chand Gupta

Department of Computer Science and Engineering, ASET, Amity University Noida, India

Department of Computer Science and Engineering, ASET, Amity University Noida, India

fabpooja91@gmail.com, scgupta@amity.edu

ABSTRACT:

The cloud is becoming more popular day by day. More companies are shifting to the cloud computing paradigm. The amount of data generated by e-commerce has grown exponentially. The business can generate huge scale up requirement in a short span of time. The data storage is meant to accommodate the changes required and scale up accordingly. However, the traditional databases do not provide the desired scalability, having their own limitations. Therefore, NoSQL systems come into picture. The paper explains the term 'scalability' and discusses NoSQL systems and their classes. This paper reviews and compares the existing scalable database management systems in cloud.

Keywords: Cloud computing, NoSQL, scalability, database management

1. INTRODUCTION

A cloud refers to a distinct IT environment that is designed for the purpose of remotely provisioning scalable and measured IT resources. The traditional database is designed for the structured (organized) data and complex queries. But in cloud, the amount of data is huge, the data is non-structured, the requirements are dynamic. All these features lie outside the boundary of traditional databases and therefore NoSQL databases come into picture. This paper discusses about some of the non-structured databases. This paper provides a survey of existing large scale data management systems in the cloud environments and which are organized by the leading cloud service providers namely Google, Microsoft, Amazon etc.

Cloud Data Management: Goals Successful cloud data management systems are expected to meet the maximum goals from the wish list:

- **Availability:** They must be always accessible even in the case of network failure or even if the whole datacenter goes offline.
- **Scalability:** The system must be able to automatically reallocate and restructure data to take advantage of the added hardware.
- **Elasticity:** The system must be able to graciously respond to the changing requirements and quickly recover to its balanced state.

- **Performance:** Efficient system performance is essential. System performance also has a direct effect on the cost.
- **Multitenancy:** They must be able to store multiple applications on the data store but the performance of these applications must be in isolation to each other.
- **Load balancing:** They must be able to automatically move load between the various servers for effective utilization of hardware resources and avoid resource overloading.
- **Fault Tolerance :** A fault tolerant cloud data management needs to recover from a failure without losing any data or update(transaction processing) or restart a query if any of the nodes involved in query processing fails.(analytical workloads).

2. SCALABILITY

Scalability is a desirable property of a system, which indicates the ability to either handle growing amounts of work in a gracious manner or its ability to improve throughput when additional resources are added to the system. A system whose performance improves after adding hardware, proportionately to the capacity added is said to be a scalable system.

There are two methods to achieve scalability: vertical and horizontal.

Vertical scaling is simpler and consists of adding resources to the existing system (such as memory, processors). If the resulting system is able to take successful advantage of the added resources, the system is said to scale up.

Horizontal scaling is complex, where new separate systems are added in parallel to the existing system. If the resulting system provides higher performance, the system is said to scale out. The usual parallel systems are divided into three classes depending on their architecture: namely shared memory shared disk and shared nothing.

Companies like Google, Amazon and Facebook realized that performance and real time nature were the key features that the traditional database lacked.

A NoSQL database provides a simple and trivial mechanism for storage and retrieval of data that provides higher scalability and availability than the traditional relational databases. Map-Reduce (MR) techniques allow processing of

partitioned sub problems in a distributed manner, with a higher degree of efficiency.

3. No SQL DATABASES

NoSQL databases were developed from the scratch with the objective of being distributed, scale out databases. They utilize a cluster of standard, physical or virtual servers to store data and support various database operations. To scale, additional servers are combined with the cluster and the data and database operations get spread across a larger cluster. Since commodity servers are probable to fail from time-to-time, NoSQL databases have been built so as to tolerate and recover from any such failure making them highly dependable.

NoSQL databases provide a direct and much easier approach to database scaling. If 10,000 new users start using your application, simply add another database server to your cluster. Add ten thousand more users and add another server. There remains no need to modify the application as you scale since the application is encountering a single (distributed) database. Often, NoSQL databases are classified according to the way they store data and fall under the categories such as key-value stores like Dynamo, BigTable implementations and document store databases like MongoDB.

No SQL databases generally scale horizontally and may not require fixed table schemas. NoSQL architecture often provides weak constancy guarantees and restricted transactional support.

The origin of the No SQL databases can be sketched back to CAP theorem:

It is impossible for a distributed computer system to simultaneously provide all three of the following assurances:

Consistency: All nodes view the same data at the same time. At any given point in time, all clients are able to read the same data.

2. Availability: Node botches do not prevent survivors from continuing to operate. It means that all the data operations (read/write) must result in a response within a specified time.

3. Partition Tolerance: The system continues to operate regardless of random message loss. Operations on the database must continue even if any component fails.

According to the theorem, a distributed system can only satisfy any two of the above guarantees at the same time, but not all three. Therefore, consistency was sacrificed in order to achieve high availability, scalability and partition tolerance. These databases moved from traditional ACID semantics to BASE semantics. BASE stands for "Basically Available, Soft State, Eventually Consistent". Under the BASE semantics, it is enough for the database to eventually be in a consistent state. ACID is pessimistic and forces consistency at the end of every transaction. BASE is optimistic and accepts that while a database may be inconsistent at certain points of time, it will eventually become consistent. This property of relaxed consistency allows the BASE systems to provide high scalability.

3.1 Google: BigTable

A BigTable cluster consists of a set of servers that serve the data; each such server is called a tablet server and is responsible for parts of the table which are known as tablets. A tablet is logically represented as a key range and physically represented as a set of SSTables. A tablet can be defined as the unit of distribution and load balancing. Data from the tables is stored in Google File System (GFS) which provides the benefit of scalable, consistent and fault-tolerant storage.

The user data is not replicated inside BigTable. All replication is handled by the GFS layer. Coordination and synchronization between the tablet server and metadata management is handled by a master and a chubby cluster. It uses centralized solutions to maintain the mapping of key intervals to the servers.

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.

BigTable supports only single-row transactions which can be used to perform atomic read-write-modify sequences on data stored under a single row key.

BigTable uses a distributed lock service called Chubby. It comprises of five active replicas, one of them is selected to be the master and actively serves requests. Chubby is used for a variety of tasks such as:

It ensures that there is at the most one active master at any time.

It stores the bootstrap location of BigTable data.

It stores BigTable schema information and the details of access control.

3.2 MongoDB

MongoDB uses auto-sharding as a way of scalability. It uses traditional dynamic queries. MongoDB is an open-source, document oriented database and the prominent NoSQL Database. It has the following features:

It stores data in the form of JSON style documents and dynamic schemas that provide simplicity and power.

It provides full index support on attributes.

It provides high availability and makes use of replication.

It uses Auto sharding and scales horizontally, does not compromise any functionality.

Queries are rich and document based.

It can store files of any size without task complications.

It provides professional support and consultancy which many of the open source systems lack.

It contains Atomic modifiers for conflict free performance.

It ensures flexibility in data processing and aggregation.

3.3 DYNAMO

It was designed by Amazon to support the shopping carts for Amazon's e-commerce business. It allows applications to relax their consistency assurance under cases when the components' fail.

It uses techniques like consistent hashing, vector clocks, Merkle tree data structure and Gossip. The query models of

Table 1. Comparison of design decisions of cloud storage systems

SYSTEM	DATABASE MODEL	QUERY INTERFACE	CAP OPTIONS	CONSISTENCY	LICENSE
BigTable	Column Families	Low-level API	CP	Eventually Consistent	Internal at Google
PNUTS	Key-Value store	Low-level API	AP	Timeline Consistent	Internal at Yahoo
Dynamo	Key-value store	Low-level API	AP	Eventually Consistent	Internal at Amazon
Cassandra	Column Families	Low-level API	AP	Eventually Consistent	Open source - Apache
MongoDB	Document oriented	Low-level API	AP	Eventually Consistent	Open source

Table 2. Comparison of RDBMS and NoSQL databases

RDBMS	NoSQL Databases
Data within the database is treated entirely and not in form of units.	Each entity is considered as a free unit of data and can be easily moved from one machine to another.
Supports centrally managed architecture.	Follows distributed architecture.
They are statically provisioned	They are dynamically provisioned.
They have limited scalability.	They are highly scalable.
They provide SQL to query data.	They make use of an API to query data.

ACID compliant	Follows BASE
Supports OLTP applications	Support Web 2.0 applications
Oracle, MySQL is the examples.	Amazon SimpleDB, Google BigTable is the examples.

Dynamo depends on simple read and writes operations on a data item that is uniquely recognized by a key.

Dynamo is used for managing those set of services that have very high reliability requirements and need a constricted control over the adjustments between availability, consistency, performance and cost-effectiveness. Dynamo explicitly replicates data and a write request can be processed by any of the replicas.

Dynamo uses consistent hashing along with replication as a partitioning scheme. It uses a distributed peer-to-peer (P2P) approach using consistent hashing. The interface Dynamo provides to client applications consists of only two operations:

get(key), returning a list of objects and a context

put(key, context, object), with no return value

To provide incremental scalability, Dynamo makes use of consistent hashing to vigorously partition the data across the storage hosts that are present in the system at a given time. Dynamo uses replication of data among nodes.

From the above study we can group the advantages and limitations of NoSQL systems.

4. ADVANTAGES OF No SQL SYSTEMS

Elastic Scaling: These systems are designed with the ability to expand transparently in order to take advantage of new nodes being added to suit the increasing demands.

Less Administration: No SQL databases have a simpler data model and are designed to support features like automatic repair, thus requiring less administration.

Better economics: No SQL databases typically use clusters of inexpensive commodity servers.

Flexible data models: NoSQL systems have more relaxed restrictions on the data model and changes can be easily incorporated.

Since no ideal systems exist, everything comes with a group of pros and cons. And there is always scope of further improvement. So we can list down a few limitations of these systems.

5. LIMITATIONS OF No SQL SYSTEMS

Programming models: NoSQL databases offer few facilities for expedient queries. Even a simple query requires significant programming expertise.

The important "join" function cannot be declaratively expressed.

Limited transaction support

Most NoSQL alternatives are still in preproduction versions with many key features which are still unstable or yet to be implemented.

There is yet lack of experts providing NoSQL support in comparison to RDBMS programmers and administrator

6. CONCLUSION

Although the NoSQL databases provide an edge above the relational databases, yet they have their own set of limitations. And there exist many avenues for future research in improving scalability of the database systems deployed in the cloud.

FUTURE RESEARCH IMPLICATIONS

With the evolving Big Data, advent of new applications, more enterprises shifting to the cloud, there exists immense research potential in the field of data management systems in cloud. The key value stores can be further extended to ensure that the performance guarantees are met while ensuring effective resource utilization.

Extending the designs of elastic, self-managing, and scalable systems to this dynamic cloud infrastructure spanning larger scale operations, higher network latency, and lower network bandwidth is a worthwhile direction of future work.

The aim is to integrate declarative queries from the database community into MapReduce like software. Incremental algorithms can be used to build a hybrid solution that allows data to be initially read off the file system. The hybrid solution should balance the performance and fault tolerance.

7. REFERENCES

- [1] Handbook of Cloud Computing, 2010 at <http://www.springerlink.com>
- [2] Abadi, D. J.: Data Management in the Cloud: Limitations and Opportunities. IEEE Bulletin of the Technical Committee on Data Engineering, Vol. 32 (1): 3-12 (2009).
- [3] Ankita Bhatewara, Kalyani Waghmare : Improving network Scalability using Nosql Database. International Journal of advanced computer research, Vol. 2, Issue 6, Dec 2012
- [4] Divyakant Agrawal, Sudipto Das, Amr El Abaddi : Data management in the Cloud: Challenges and opportunities, Google Books
- [5] Divyakant Agrawal, Sudipto Das, Amr El Abaddi : Big Data in cloud computing: Current state and Future opportunities, EDBT 2011
- [6] Daniel J. Abadi : Data Management in the cloud : Limitations and Opportunities. IEEE 2009