Implication CAP Theorem on Big Data Storage Solutions

Tejas Phopale

Student Id: x17162301, Email: x17162301@gmail.com

DSM – Project A

ABSTRACT

The data storage technologies have observed many changes from past decades right from basic hard disk storage and installation to the virtualized cloud environment technology. There is a big thrust to develop more feasible tools to store fast-growing volume of data. In this Era of Big Data, it is necessity to store high volume of data in timely and efficient manner. The paper provides a brief idea about the implication of the existing approaches using CAP theorem on different Big Data storage solutions at both infrastructure and application level. Their characteristics and measures transaction chosen between consistency and availability to investigate the magnitude gains with the aim to accomplish reliable and scalable storage system.

Keywords: Big data, Big data storage, CAP theorem, Scalability, Consistency-partition resilience, Availability-partition resilience, NoSQL databases, Distributed databases, SQL RDBMS

INTRODUCTION

The CAP theorem also known as Brewer's CAP theorem states that it is impossible to provide more than two out of three characteristics in distributed data storage system. Those three characteristics are Consistency, Availability and Partition tolerance. By keeping presence of network partition tolerance, one has to select from other two. At Today's Date, NoSQL also known as ''Not Only SQL" databases have been widely used to solve the latency issue and improve read-write issue. In this paper, we have analysed NoSQL distributed storage system according to CAP theorem, discussed their importance and also list the methods of Brewer's theorem in analysing distributed systems. The well-known database technologies are compared, categorised and analysed under some approach. For analysis the models like key-value, column-oriented, and graph data models are used and categorized them with Brewer's CAP theorem (Siddiqa et al., 2016). Also, we will be discussing the storage implementations of three major giants: Google's Bigtable, Amazon's Dynamo and Yahoo's PNUTS and some of NewSQL databases Clustrix, Google Spanner, NuoDB, VoltDB (Kaur et al., 2015) and index update performance with proposed Different Indexes, a classification of index update schemes like sync full, insert and async simple, session using CAP theorem (Tan et al., 2014).

WHT is CAP Theorem?

CAP theorem (Tan *et al.*, 2014) indicates that any networked and replicated data store can use two of the three properties that are consistency, availability and network partition tolerance. In a distributed data store system, partition-tolerance is always included. This influences that selection to be made between consistency and availability. The Consistency and Availability are aligned opposite to each other so decision need to be made among these two. The reasons for choosing one out of the two is as follows:

1) Work progress ON even after data is at unsynchronized state

2) Increase in latency in distributed environment,

3) Index update is getting delayed due to many workloads are write intensive but read from time to time

4) sometimes in replications data does not takes base data.

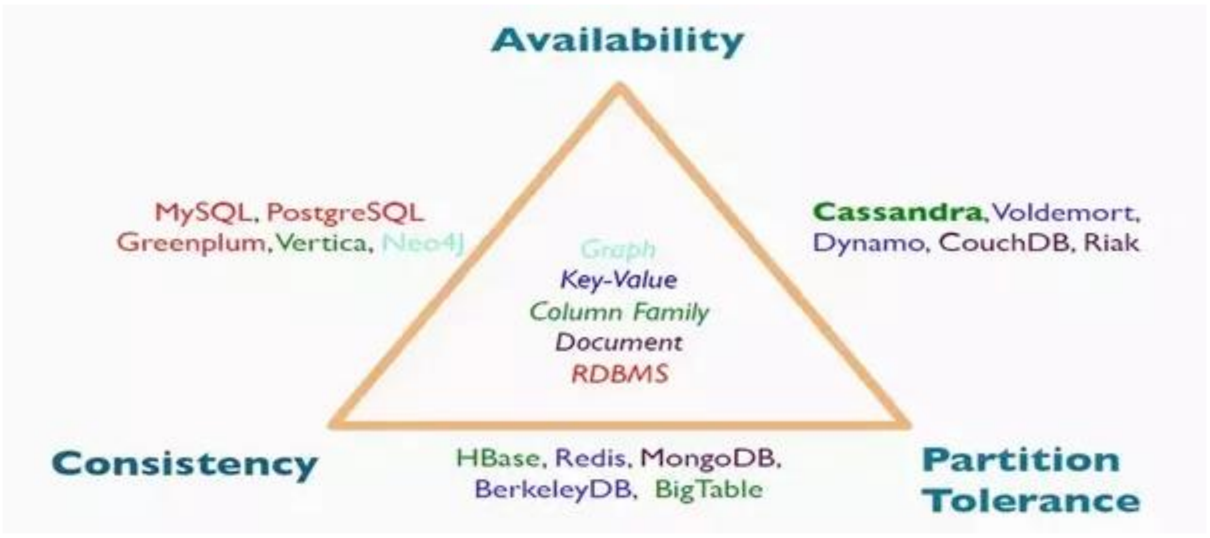


Figure 1: CAP Theorem (Source : https://www.quora.com/HBase-follows-which-features-of-CAP-theorem)

Big Data Storage Solutions using CAP Theorem

Big data is frontier topic of this new era as it refers to rapidly increase in data gathered from different heterogeneous devices. The main motive of the data storage solutions that it should be high in scalable, reliable and efficient. MySQL is still used to store these structured data because of its ACID transactions which compared with referential integrity as MongoDB wherein no records without reference can exist which means that there is no foreign key checking. NoSQL database management systems can store data, which change dynamically in size much better than relational ones. The nodes can be added dynamically to lengthen horizontal scalability. Considering the three guarantees of the CAP theorem, consistency and availability are seems to be most important for the relational database (Ebel *et al.*, 2012). In the comparison of BASE databases, it definite to have Atomicity, Consistency and Partition tolerance. BASE theorem does not support the CAP theorem but works around it. The BASE is designed in such a way that user's database failure affects only 20% of that particular host (Chandra *et al.*, 2015). So finally, we are focussing on global indexes to get nice support on selective queries of big data and distributed index using CAP theorem in distributed environment.

To understand the CAP theorem, we need to focus on it important characteristics. According to CAP theorem (Brewer et al., 2012), the storage system can be of CP type or AP type and it is impossible for a database to have both ACID property together so it composes the combination like consistency–partition resilience and availability–partition resilience. Below is the list of data storage technologies (Siddiqa et al., 2016) differentiated by types of data models such as key-value based, column-oriented, document oriented and graph base.

Key-value-based:

1) **Scalaris**: It is open source datastore with custom query language, eventual consistency, symmetric replication, key-based partitioning and primary key indexing. With CP Brewer’s category, the model is strong, consistent, Scalable, having high availability with load balancing and fault tolerance with very little maintenance overhead.
2) **Aerospike**: It is open source datastore with AQL (Aerospike Query Language) query language, strong consistency synchronized-asynchronized replication, sharding partitioning and secondary indexing. With AP Brewer’s category, it is highly scalable, consistent and reliable.
3) **Redis:** It is open source datastore with eventual consistency, master-slave replication, consistent hashing partitioning and customized indexing. With CP Brewer’s category, it comprises automatic partitioning plus efficient data read or write access fault-tolerant. It is responsive if replica is down.
4) **Voldemort:** It is open source datastore with their internal language eventual consistency tuneable-eventual consistency, selectable replication factor replication, consistent hashing partitioning and customized indexing. With AP Brewer’s category, is comprises of automatic data partitioning. The replication Transparent fault recovery with high read or write availability and its available in horizontal scalability.
5) **KAI:** It is open source datastore with eventual consistency, asynchronized replication, consistent hashing partition. With AP Brewer’s category, it is highly fault-tolerant with low latency and scalable with configurable nodes.
6) **MemcacheDB:** It is open source datastore with API query language, ordered consistency, master-slave replication, no partitioning and customized indexing. With CP Brewer’s category, it has efficient data storage, shows good performance for read or write and high storage availability.
7) **Riak:** It is open source database with MapReduce query language for integration, eventual consistency, multi-master replication, consistent hashing partitioning and secondary indexing. With AP Brewer’s category, it derives DynamoDB, fault-tolerant and highly available. The data conflict resolutions help to make configuration on different commodity hardware.
8) **BerkeleyDB:** It is open/commercial source database with XQuery as query language, eventual consistency, master-slave replication, no partitioning and secondary indexing. With CP Brewer’s category, it comprises Scalability and high performance. The configurable products support to complex data management procedures.
9) **DynamoDB:** It is commercial source database with API query language, eventual consistency, cross-region replication, consistent hashing partitioning and primary-secondary key indexing. With AP Brewer’s category, is has high write availability and

automatic replication with fault tolerance. It uses programming model as JAVA which makes it stronger in object oriented.

10) **PNUTS:** The data model implementation is Key-value pairs and objects are getting stored along with key and context. Transactions can be single or multi record transactions. The replication is asynchronous replications using pub/sub Yahoo! Message Broker (YMB) and the consistency is as per record level mastering. The partitioning System is divided into regions and data tables are partitioned horizontally into tables and these tables store multiple records. The symmetry is centrally managed and maintained with consistency guarantee. It supports low level API calls with different levels of consistency guarantees and comprise Hadoop and Pig as programming model.

Column-oriented:

1) **HBase:** It is open source datastore with Range and MapReduce as query language for integration, transactional consistency, selectable replication factor, sharding partitioning and primary key indexing. With CP Brewer's category, it has concurrent mode, failure exception hindering read write performance, auto split and redistribution of data.
2) **Hypertable:** It is open source database with HQL as query language, transactional consistency, selectable replication factor, sharding partitioning and secondary key indexing. With CP Brewer's category, it is highly available and fast random read/write Compatible with many distributed file systems.
3) **Cassandra:** It is open source database with Range, CQL (same as SQL) is built language, MapReduce as query language for integration, tunable-eventual consistency, selectable replication factor, sharding partitioning and secondary indexing. With AP Brewer's category, it is deriving DynamoDB, easy and cost saving, high write throughput and it has no read locks.
4) **BigTable:** It is proprietary datastore with GQL as query language, transactional consistency, master-slave replication, horizontal partitioning and secondary indexing. With CP Brewer's category, it is easy to data compression process, fast query response. It is allowing an infinite number of columns in a table. It has automatic and less reconfiguration to scale the system. It uses MapReduce as programming model for integration.

Document-oriented:

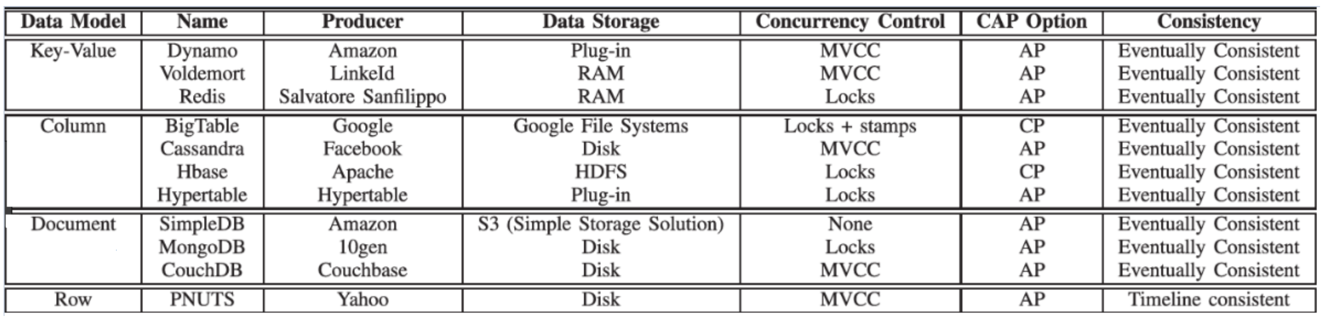
1) **MongoDB:** It is open source database with SQL, MapReduce as query language for integration but uses JSON fragments, eventual consistency, replica set and master-slave replication, sharding partitioning and secondary key indexing. With CP Brewer's category, it is easily scalable, fault-tolerant, and highly available. It has built-in data encryption file system and supports to complex schema with flexibility in data model.
2) **Terrastore:** It is open source datastore with API query language, per page consistency, master-slave replication, horizontal partitioning and secondary key indexing. With CP Brewer's category, it is persistent in-memory document storage mechanism, automatic data redistribution and load balancing with dynamic cluster configuration.
3) **RethinkDB:** It is open source database with ReQl query language, strong consistency, synchronized-asynchronized replication, sharding partition and secondary key indexing. With CP Brewer's category, it is easy in scalability, robust and more consistent and fast real-time query response.

4) **SimpleDB:** It is open source datastore with Erlang as query language, eventual consistency, master-slave replication, sharding partition and customized & metadata indexing. With AP Brewer's category, it is highly available with fast query retrieval.
5) **CouchDB:** It is open source database with MapReduce as query language for integration, eventual consistency, multi-master replication, sharding partition and B-tree indexing. With AP Brewer's category, it is Easy to use, fault-tolerant, observes concurrency for request workload.
6) **OrientDB:** It is open source database with SQL query language, multi-version consistency, multi-master replication, sharding partition and secondary indexing. With AP Brewer's category, it has high-speed storage, Low-cost scalability and heterogeneity of replicating servers with schema-less model.
7) **Rocket U2:** It is proprietary database with RetrieVe and UniQuery as query language, runtime consistency, master-slave & asynchronized replication, no partitioning and secondary indexing. With AP Brewer's category, it dynamically supports to applications and highly efficient, scalable, and reliable for growing data.
8) **Qizx:** It is commercial database with XQuery, REST as query language, transactional consistency, master-slave replication, runtime partitioning and full-text, customize indexing. With AP Brewer's category, it is highly scalable, available, and consistent. It is fast and efficient query execution which supports customized indexing.

Graph-based:

1) **HyperGraphDB:** It is open source database with API query language, eventual consistency, per page consistency and replication and key-value indexing. With CP Brewer's category, it is flexible and dynamic schema, non-blocking concurrency, efficient data modelling and knowledge representation.
2) **Neo4j:** It is open source database with SPARQL and internal inbuilt query languages, strong/eventual consistency, master-slave replication, sharding partition and secondary key indexing. With CP Brewer's category, it is highly scalable and robust, efficient concurrent write transactions without locking. It is fast for write scaling transaction loads.
3) **AllegroGraph:** It is open source database with SPARQL and RDFS++ as query languages, transactional consistency, master-slave replication, federation partitioning with secondary and/or secondary-primary indexing. With CP Brewer's category, it has high storage throughput, highly scalable, available and persistent. It has Fast query execution and high data load speed performance.
4) **InfiniteGraph:** It is commercial database with inbuilt-API as query languages, flexible-eventual consistency, synchronized replication, sharding partition and secondary indexing. With CP Brewer's category, it has rapid development of data-intensive, graph-based applications. It is easy traversal of complex graph elements with processing load distribution. Its availability of data filtering techniques to improve query performance and it is very easy to use.

If we compare the above solution, we could find that **Bigtable, Dynamo and PNUTS** comes up as key major player and best solution among the all. All the three shows great performance with their own strategy in terms of data model implementation, transaction, replication, consistency, partitions, symmetry etc. Dynamo and PNUTS follows AP as CAP option wherein Bigtable uses CP CAP option. The programming model of all these three helps to these solutions as controlled model.



Data Model	Name	Producer	Data Storage	Concurrency Control	CAP Option	Consistency
Key-Value	Dynamo	Amazon	Plug-in	MVCC	AP	Eventually Consistent
	Voldemort	LinkeId	RAM	MVCC	AP	Eventually Consistent
	Redis	Salvatore Sanfilippo	RAM	Locks	AP	Eventually Consistent
Column	BigTable	Google	Google File Systems	Locks + stamps	CP	Eventually Consistent
	Cassandra	Facebook	Disk	MVCC	AP	Eventually Consistent
	Hbase	Apache	HDFS	Locks	CP	Eventually Consistent
	Hypertable	Hypertable	Plug-in	Locks	AP	Eventually Consistent
Document	SimpleDB	Amazon	S3 (Simple Storage Solution)	None	AP	Eventually Consistent
	MongoDB	10gen	Disk	Locks	AP	Eventually Consistent
	CouchDB	Couchbase	Disk	MVCC	AP	Eventually Consistent
Row	PNUTS	Yahoo	Disk	MVCC	AP	Timeline consistent

Figure 2: Comparison of selective major datasets
(Source:https://ieeexplore.ieee.org/document/6842585/?section=abstract)

Also discussed in (Kaur *et al*., 2015), we will add with some of NewSQL databases Clustrix, Google Spanner, NuoDB, VoltDB and their features. Those are the new SQL databases which are upcoming scalable RDBMS that overcome the challenges which are involved in using NoSQL databases such as maturity, support, analytics & business intelligence, administration and expertise etc.

1) **Clustrix:** Clustrix has strong consistency and concurrency controlled by MVCC. It follows Consistent Hashing partitioning using user defined primary key wherein table indices are also partitioned. The replication updates are forwarded to all replicas by transaction manager and those are being executed in parallel.
2) **Google Spanner:** Spanner has strong consistency with read-write locks as concurrency control. It has add-on functionality like lock-free reads to store multiple versions. Data is partitioned ranging from span servers to tablets to directories. The set of rows sharing common key prefixes. For replication it uses "Paxos" State Machine Algorithm.
3) **NuoDB:** NuoDB has eventual consistency with MVCC as concurrency control. The solution does not have partitioning. Storage Manager has complete copy of data. The underlying key-value stores in Storage Manager can partition data but it is not visible. The replication is getting Carried via asynchronous communication among table rows.
4) **VoltDB:** VoltDB has strong consistency with no concurrency control. The partitioning done by consistent hashing and replication updates forwarded to all replicas by transaction manager and there are parallelly getting executed.

CONCLUSION

It is very essential that to have efficient data storage management technologies which needs to be implemented by taking consideration of huge data generation every day. Big data storage solutions seem future of storage technology at both the infrastructure and database application levels. The google Bigtable, Dynamo, PNUTS etc. has grown up as high performance and efficient solutions and we can predict their updated version can be available with new features. Although big data solutions have several advantages over traditional database, the future storage system should be complemented by combining upcoming scalable RDBMS which will help to overcome the mutual challenges.

REFERENCES

Siddiqa, A., Karim, A. and Gani, A., 2017. Big data storage technologies: a survey. Frontiers of Information Technology & Electronic Engineering, 18(8), pp.1040-1070.

Kaur, P.D. and Sharma, G., 2015. Scalable database management in cloud computing. Procedia Computer Science, 70, pp.658-667.

Brewer, E., 2012. Pushing the cap: Strategies for consistency and availability. Computer, 45(2), pp.23-29.

Tan, W., Tata, S., Tang, Y. and Fong, L.L., 2014. Diff-Index: Differentiated Index in Distributed Log-Structured Data Stores. In EDBT (pp. 700-711).

Ebel, M. and Hulin, M., 2012, August. Combining relational and semi-structured databases for an inquiry application. In International Conference on Availability, Reliability, and Security (pp. 73-84). Springer, Berlin, Heidelberg.

Chandra, D.G., 2015. BASE analysis of NoSQL database. Future Generation Computer Systems, 52, pp.13-21.

Major datasets, https://ieeexplore.ieee.org/document/6842585/?section=abstract [Date accessed: 02-05-2018]

CAP-theorem, https://www.quora.com/HBase-follows-which-features-of-CAP-theorem [Date accessed: 02-05-2018]