ISSN (Online): 2349-7084
GLOBAL IMPACT FACTOR 0.238
DIIF 0.876

INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 8, AUGUST 2015, PP 463-469

A SURVEY ON IMPLEMANTATION OF COLUMN ORIENTED NOSQL DATA STORES

(BIGTABLE & CASSANDRA)

M.Purna Chary [1], Srinivasa S P Kumar.B [2], T.RamDas Naik [3]

1. Assistant professor, Department of Informatics, Nizam college, Hyderabad, Telangana
2. Assistant professor, Department of CSE, CBIT, Hyderabad, Telangana
3. Assistant professor, Department of Informatics, Nizam college, Hyderabad, Telangana

Abstract - NOSQL is a database provides a mechanism for storage and retrieval of data that is modeled for huge amount of data which is used in big data and Cloud Computing. NOSQL systems are also called "Not only SQL" to emphasize that they may support SQL-like query languages. A basic classification of NOSQL is based on data model; they are like column, Document, Key-Value etc. The objective of this paper is to study and compare the implantation of various column oriented data stores like Bigtable, Cassandra.

Keywords-NoSQL,Bigtable, Cassandra, Transaction, Atomocity, Consistency, CAP

---

1. INTRODUCTION

NOSQL technology is an open source concept and used for applications handling huge volumes of data. NOSQL systems are also called "Not only SQL" to emphasize that they may support SQL-like query languages. These databases do not have fixed schema and are not based on relational Concept like the relational database management systems (RDBMS). NOSQL database management system provides high speed access to semi-structured and un-structured data and is very flexible to use. There are several types of NOSQL database management system like Key-value stores, Document-oriented and Column oriented database etc. The Computerworld article summarizes reasons commonly given to develop and use NoSQL data stores:

1.1 AVOIDANCE OF UNNEEDED COMPLEXITY Relational databases provide a variety of features and strict data consistency. But this rich feature set and the ACID properties implemented by RDBMSs might be more than necessary for particular applications and use cases.

1.2 HIGH THROUGHPUT Some NoSQL databases provide a significantly higher data throughput than traditional RDBMSs. For instance, the column-store Hypertable which pursues Google's Bigtable approach allows the local search engine Zvent to store one billion data cells per day. To give another example, Google is able to process 20 petabyte a day stored in Bigtable via its Map Reduce approach

1.3 AVOIDANCE OF EXPENSIVE OBJECT-RELATIONAL MAPPING Most of the NoSQL databases are designed to store data structures that are either simple or more similar to the ones of object-oriented programming languages compared to relational data structures. They do not make expensive object-relational mapping necessary (such as Key/Value-Stores or Document-Stores).

2. CHARACTRISTICS OF NOSQL

2.1 CONSISTENCY, AVAILABILITY, PARTITION TOLERANCE (CAP)

ISSN (Online): 2349-7084
INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 8, AUGUST 2015, PP 463-469

When evaluating NoSQL or other distributed systems, you'll inevitably hear about the "CAP theorem." In 2000 Eric Brewer proposed the idea that in a distributed system you can't continually maintain perfect consistency, availability, and partition tolerance simultaneously.CAP is defined by Wikipedia [7] as:

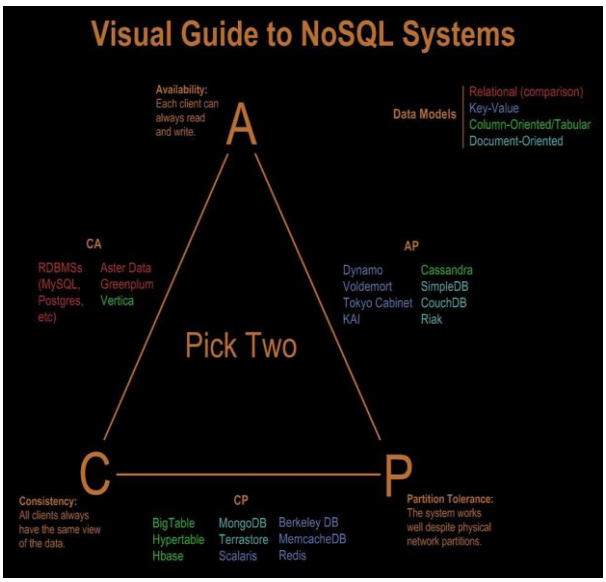
Consistency: all nodes see the same data at the same time

Availability: a guarantee that every request receives a response about whether it was successful or failed

Partition tolerance: the system continues to operate despite arbitrary message loss

The theorem states that you cannot simultaneously have all three; you must make tradeoffs among them. The CAP theorem is sometimes incorrectly described as a simple design-time decision—"pick any two [when designing a distributed system]"—when in fact the theorem allows for systems to make tradeoffs at run-time to accommodate different requirements. Too often you will hear something like, "We trade consistency (C) for AP," which can be true but is often too broad and exposes a misunderstanding of the constraints imposed by the CAP theorem. Look for systems that talk about CAP tradeoffs relative to operations the product provides rather than relative to the product as a whole.

Figure 2.1 : CAP Visual Guide



2.2 RELAXING ACID

ACID properties are the fundamental elements of transactions in RDBMS: atomicity, consistency, isolation, and durability. Together, these qualities define the basics of any transaction. As NoSQL solutions developed it became clear that in order to deliver scalability it might be necessary to relax or redefine some of these qualities, in particular consistency and durability. Complete consistency in a distributed environment requires a great deal of communication involving locks, which force systems to wait on each other before proceeding to mutate shared data. Even in cases where multiple systems are generally not operating on the same piece of data, there is a great deal of overhead that prevents systems from scaling.

To address this, A "BASE" acronym is usually used in the context of NoSQL data stores in contrast to the "ACID" acronym. "BASE" means Basically Available, Soft state and eventually consistent. Soft state means that strict constraints are not followed. Eventual consistency indicates a loosened consistency model where the updates are propagated eventually to all the copies i.e. strict one copy consistency is not followed.

2.3 DATA AND ACCESS MODEL

The relational data model with its tables, views, rows, and columns has been very successful and can be used to model most data problems. By using constraints, triggers, fine-grained access control, and other features, developers can create systems that enforce structure and referential integrity and that secure data. These are all good things, but they come at a price. First, there is no overlap in the data representation in SQL databases and in programming languages; each access requires translation to and from the database.

NoSQL solutions have taken a different approach. In fact, NoSQL solutions diverge quite a bit from one another as well as from the RDBMS norm. There are three main data representation camps within NoSQL: document, key/value, and graph. There is still a fairly diverse set of solutions within each of these categories. For instance, Riak, Redis, and Cassandra are all key/value databases, but with Cassandra you'll find a slightly more complex concept, based on Google's Bigtable, called "column families," which is very different from the more SimpleDB-like "buckets containing key/value pairs" approach of the other two.

2.4 DISTRIBUTED DATA, DISTRIBUTED PROCESSING

ISSN (Online): 2349-7084
INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 8, AUGUST 2015, PP 463-469

NoSQL solutions are generally designed to manage large amounts of data, more than you would store on any single system, and so all generally have some notion of partitioning (or sharding) data across the storage found on multiple servers rather than expecting a centrally connected SAN or networked file system. The benefits of doing this transparently are scalability and reliability. The additional reliability comes when partitions overlap, keeping redundant copies of the same data at multiple nodes in the system. Not all NoSQL systems do this.

3. COLUMN ORIENTED DATA STORES

Column Family Stores are also known as column oriented stores, extensible record stores and wide columnar stores. All stores are inspired by Google's Bigtable which is a distributed storage system for managing structured data that is designed to scale to a very large size. Column stores in nosql are actually hybrid row/column store unlike pure relational column databases. Although it shares the concept of column-by-column storage of columnar databases and columnar extensions to row-based databases, column stores do not store data in tables but store the data in massively distributed architectures. In column stores, each key is associated with one or more attributes (columns). A Column store stores its data in such a manner that it can be aggregated
Rapidly with less I/O activity. It offers high scalability in data storage. The data which is stored in the database is based on the sort order of the column family.[3] Columns can be grouped to column families, which is especially important for data organization and partitioning Columns and rows can be added very flexibly at runtime but column families have to be predefined oftentimes, which leads to less flexibility than key value stores and document stores offer. Examples of column family data stores include Hbase, Hypertable, cassandra.

3.1. GOOGLE `S BIGTABLE

The approach of column-oriented storage originated in analytics and business intelligence where column stores on shared nothing - parallel architecture is used for developing high performance applications. Bigtable is a distributed storage system that can handle structured data. It has immense scaling capacity and can handle peta bytes of data across thousands of commodity servers. It is used in more than 60 projects at Google which differ in their data size, infrastructure and latency requirements. Bigtable was successful in achieving various goals like applicability, scalability, performance and high availability.

1. DATA MODEL

The key-value pair model used by different data stores has many limitations and it cannot be the only building block provided to the developers. In Bigtable, the data model is richer than key value and support sparse and semi structured data. It is a sparse, sorted, distributed and multi dimensional map. The values are stored as byte arrays and are addressed by the triple (row key, column key, time stamp). Figure 1 shows an example of how Bigtable stores information emitted by a simple web crawler. It contains arbitrary number of rows representing the domains and non fixed number of columns. The first column contains the page contents and the other columns store the link texts from the referring domains. Every value will be associated with a time stamp and the value is addressed here by the triple (domain name, column name, time stamp). Row keys are strings and are maintained in lexicographic order. The unit of distribution and load balancing in Bigtable are tablets which is a collection of rows. The number of columns is not fixed and sets of columns can be grouped by the common prefix to form column families. Time stamps are 64 bit integers and are used to distinguish different versions of a cell value. The value of the time stamp is assigned by either the data store or by the client application.

2. IMPLEMENTATION

Every Bigtable instance consists of three major components:

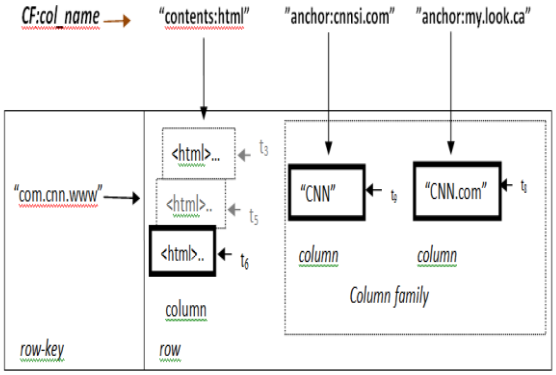
Multiple Tablet Servers – for handling read and write requests for tablets and splitting of tablets.

A Client Library – for the applications to interact with Bigtable instances.

One Master Server – for managing tablets and tablet servers, distributing workload, processing changes to schema etc.

Figure 3.1: Google's Bigtable – Example of Web Crawler Results (taken from [CDG+06, p. 2])

ISSN (Online): 2349-7084
INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 8, AUGUST 2015, PP 463-469



3.2 APACHE CASSANDRA

Apache Cassandra adopts ideas and concepts of both Amazon's Dynamo as well as Google's Bigtable. It was originally developed by Facebook and open-sourced in 2008. Cassandra is a "distributed storage system for managing structured data that is designed to scale to a very large size". It does not completely follow relational data model but provides clients a simple model which enables dynamic control over the data layout and format. Other than Facebook, Twitter, Digg, Rack space etc use Cassandra.

1. DATA MODEL

A Cassandra instance normally has a single table which can be considered as a distributed multi dimensional map indexed by a key. A table in Cassandra has the following dimensions:

Rows – identified by keys, which are strings of arbitrary length.

Column Families – arbitrary number of column families can be present per row

Columns – arbitrary number of columns can be present for every row and they store a number of values per row. The values can be distinguished using timestamps as in Bigtable.

Super Columns - have a name and an arbitrary number of columns can be associated with them. The number of columns per super-column may differ per row.

2. IMPLEMENTATION

A server participating in a Cassandra cluster executes modules providing the following functionality:

- Partitioning.
- Storage engine.
- Cluster membership and failure detection.

All system control messages follow UDP messaging while the application related messages relies on TCP. Request routing is implemented by means of a state machine on the storage nodes. When a request arrives, it has the following states:

1. Identify the nodes that have the data corresponding to the requested key.
2. Route the request to the identified nodes in step 1 and wait for the response.
3. The request fails and returns to the client if the nodes contacted in step 2 do not reply within the configured amount of time.
4. Determine the latest response based on timestamp.
5. If any replica is not having the latest data, update it.

4. COMPARISON OF BIGTABLE AND CASSANSRA

Bigtable and Cassandra are compared with respect to various issues in detailed in table 4.1, 4.2, 4.3 respectively as follows. In Table 4.1, issues like Integrity and in 4.2, Design and Features and in 4.3, Indexing and Distribution are addressed.

ISSUE	Title	BIGTABLE	CASSANDRA
Integrity	Integrity model	ACID	BASE
	Atomicity	?	Yes
	Consistency	Yes	Yes
	Isolation	No	No
	Durability (data storage)	?	Yes
	Transactions	Yes	No
	Referential integrity	?	No
	Revision control	Yes	Yes
	Locking model	?	Lock Free Model

ISSN (Online): 2349-7084
INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 8, AUGUST 2015, PP 463-469

Table 4.1 :
In Table 4.1, Integrity Issues are Compared. Here we observe that Atomocity, Durability, Integrity constraints and locking model are not addressed in Bigtable.

In the Table 4.2 we try to compare the issues like Design and Features. Here also some of the Issues are not addressed by Bigtable.

ISSUE	Title	BIGTABLE	CASSANDRA
Design	Database model	Column-oriented	Column-oriented Key-value
	Data storage	GFS2	File System
	Embeddable	?	No
Features	Query language	API calls	API calls CQL Thrift
	Data types	String	All MySQL JSON BLOB userdefined STATIC COLUMN
	Conditional entry updates	?	Yes
	Map and reduce	Yes	Yes
	Unicode	Yes	Yes
	TTL for entries	?	Yes
	Compression	Yes	Yes

Table 4.2 :Comparison of Bigtable and Cassandra
In Table 4.3 we try to compare some other issues like Indexing and Distribution. Here composite Key Issues are not addressed by Bigtable.

5.OBSERVATION

In the above comparison it is observed that some of the Integrity issues like atomocity, Durability, Referencial Integrity and Locking model should be addressed. Apart from this some of the design and indexing, distribution issues should also be addressed in Bigtable. In Cassandra, Many security issues should be Addressed.

ISSUE	Title	BIGTABLE	CASSANDRA
INDEXIN	Secondary Indexes	Yes	Yes
	Composite keys	?	Yes
	Full te search	Yes	No
	Geospatial Indexes	Yes	No
	Graph support	Yes	No
DISTRIBUION	Secondary Indexes	Yes	Yes
	Composite keys	?	Yes
	Full te search	Yes	No
	Geospatial Indexes	Yes	No
	Graph support	Yes	No

Table 4.3: Comparison of Bigtable and Cassandra.

6. CONCLUSION

The approach to store and process data by column

instead of row has its origin in analytics and business intelligence where column-stores operating in a shared-nothing massively parallel processing architecture can be used to build high-performance applications. In this paper it is examined the implementation process of column oriented data stores like Bigtable and Cassandra in NOSQL Data stores and also compared both the Bigtable and Cassandra each other with respect to various issues like Features, Integrity, Indexing ,Distributions, Design etc. Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size. Cassandra is a distributed database designed to be highly scalable both in terms of storage volume and request throughput while not being subject to any single point of failure.

REFERENCES

[1]. S. Gilbert and N. Lynch, "Brewer's conjecture an the feasibility of consistent, available, and partitior tolerant web services", ACM SIGACT News 33, 2, pp. 51-59, March 2002.

[2]. Anna Bjorklund, —NoSQL databases for Softwar Project datall . January 18, 2011.

[3] D. J. Abadi. Query execution in column-oriente database systems. MIT PhD Dissertation, 2008. Phl Thesis.

[4] D. J. Abadi, S. R. Madden, and M. Ferreir Integrating compression and execution in columr oriented database systems. In SIGMOD, pages 671 682, 2006.

[5] D. J. Abadi, D. S. Myers, D. J. DeWitt, and S. I Madden. Materialization strategies in a columr oriented DBMS. In ICDE, pages 466–475, 2007.

[6] ABADI, D. J., MADDEN, S. R., AND FERREIR M. C. Integrating compression and execution i columnoriented database systems. Proc. of SIGMOI (2006).

[7] AILAMAKI, A., DEWITT, D. J., HILL, M. D., ANI SKOUNAKIS,M. Weaving relations for cach performance.In The VLDB Journal (2001), pp. 169.180

[8] BANGA, G., DRUSCHEL, P., AND MOGUL, J. C Resource containers: A new facility for resourc managementin server systems. In Proc.

[9] BARU, C. K., FECTEAU, G., GOYAL, A., HSIAC H., JHINGRAN, A., PADMANABHAN, S COPELAND,

[10] Misc. Authors Apache Cassandra 0.6.3 Java Source Code Available from http://cassandra.apache.org

[11] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra,A. Fikes, and R. E. Gruber, Bigtable: A Distributed Storage System for Structured Data OSDI'06: Seventh Symposium on Operating System Design and Implementation, 2006, Seattle,WA, 2006.

[12] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogel Dynamo: Amazons Highly Available Keyvalue Stor In Proceedings of twenty-first ACM SIGOP symposium on Operating systems principles (2007 ACM Press New York, NY, USA, pp. 205220

[13] A. Lakshman, P. Malik, Cassandra - Decentralized Structured Storage System, Cornel 2009.

[14] M. Slee, A. Agarwal, M. Kwiatkowski, Thrif Scalable Cross-Language Services Implementatio Facebook, Palo Alto, CA, 2007.

[15] R. Tavory, Hector a Java Cassandr clienthttp://prettyprint.me/2010/02/23/hector-ajava-cassandra-client February, 2010

[16] R. Tavory, Hector Java Source Code Availabl from http://github.com/rantav/hector

[17] Thrift Wiki http://wiki.apache.org/thrift

[18] F. Cristian, Understanding Fault-Toleran Distributed Systems University of California.

[19] www.ijecse.org ISSN- 2277-1956 ISSN 2277-1956/V2N1-133-141Bigtable, Dynamo Cassandra – A Review Kala Karun A.

ISSN (Online): 2349-7084
INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 8, AUGUST 2015, PP 463-469