Performance Comparison between Column Store NoSQL Databases

Preeti Bhatt*
** D.C.S.A. Panjab University sector-14 Chandigarh*
Preeti.gcg@gmail.com

Abstract- After the emergence of cloud computing and other distributed applications, the need of a non-relational database increases that can handle large volumes of unstructured or semi-structured data. Not only SQL (NoSQL) database is distributed, non-relational database which provide scalability and availability and provide a variations in database which we can choose according to our application's requirement. In this paper we describe the column-store NoSQL databases and the performance comparison between them.

Keywords- Non-Relational, NoSQL, Column Store Databases, Scalability, Distributive Databases.

I.INTRODUCTION

Column-store NoSQL databases are very popular in social networking websites, mobile apps, google projects, medical fields and provide High availability, faster access and consistency and partition tolerance. In a column-store NoSQL database, data is stored in columns that are logically grouped into column families. Column-family consists of rows which contains columns. Rows are referenced by row keys which is unique identifier of that row. Column structure consists of name, value and timestamp. Figure 1 shows the structure of column in column store database.

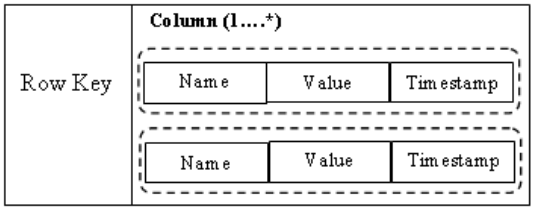

Figure1: Structure of column in column-store databases

1.1 The Structure of a Column Store Database

All columns belong to a column–family and column family belongs to keyspace. Column families can contain any number of columns that can be created at runtime or while defining the schema. Column-family consist of multiple rows. Each row consists of any number of columns. Column structure (size and name) in each row need not be the same. Figure2: shows three customer records.

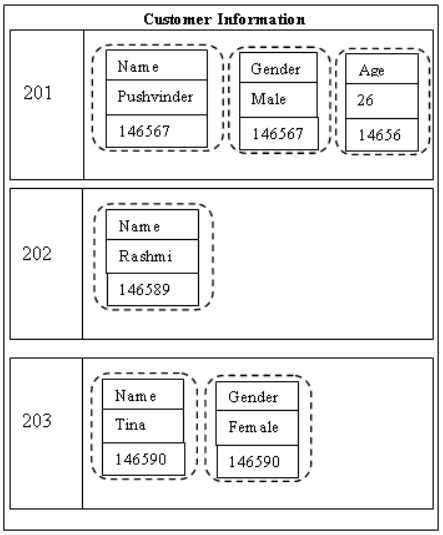

Figure1: A column-family structure shows variable customer information

1.2 Benefits of Column Store Databases

Some key benefits of columnar databases include:

- Compression: Column stores provide optimized data compression.
- Aggregation queries. Faster execution of aggregation queries due to wide column structure.
- Scalability. Columnar databases are very scalable. They are well suited to massively parallel processing (MPP), which involves having data spread across a large cluster of machines – often thousands of machines.
- Fast to load and query: Immediately load, query and analyze any number of tables.
- Preserving Disk Space: Unlike RDBMS, Column Store databases solves sparse data problem by ignoring blank padding null value and don't require fields to always be present. Thus preserves disk space.
- Easy Data Retrieval: Rather than using the complex Structured Query Language (SQL) join, all related information can be retrieved using a single record ID with little data analyses and upfront modeling.
- More Efficient and less Error Prone.: Column store NoSQL database if more efficient in storage and less error prone in development for complex and variable relational data structure.
- Faster access: Storing data in columns gives faster search and data aggregation as compared to storing data in rows (in RDBMS) because in Column store databases stores all the cells corresponding to a column as a continuous disk entry thus makes the search/access faster. But RDBMS stores different rows at different places in disk.

Like relational databases, the concept of rows and columns and structure of data is known before loading data into database. However, data is organized in columns instead of rows for faster access. This column centric data organization make it ideal for searching records against multiple columns and for running aggregate functions.
Column stores are also named as Big Tables, reflecting their common ancestor, Google's Bigtable.

1.3 Use cases: Developers mainly use column databases in:

- Content management systems
- Blogging platforms
- Systems that maintain counters
- Services that have expiring usage
- Systems that require heavy write requests (like log aggregators)

1.4 When should column store database be used?

- For semi-structured data, Column store database is used because it requires scalability and high performance.
- Queries that involve only a few columns
- Aggregation queries against vast amounts of data
- Column-wise compression

1.5 When should column store database be avoided?

- If you have to use complex querying.
- If you're querying patterns frequently change.
- If you don't have an established database requirement.
- Incremental data loading
- Online Transaction Processing (OLTP) usage
- Queries against only a few rows

Examples of column store NoSQL databases are Cassandra [15], Apache Hadoop HBase [17], BigTable [16], HyperTable, DynamoDB [14], and Click House.

II.RELATED WORK

[1] Did some performance test on Hbase column-store database, including column-family test conclude that when we increase number of column families then the reading and writing speed becomes slower because in Hbase each column family is stored separately. Sorting test says there is no difference in performance when row keys are written in lexicographic or reverse lexicographic order because HBase uses B-tree storage with write cache and the speed does not affected by number and order of writes. Query test says that query speed increases with multi-cluster environment because HBase tables have simple key-value pair's relations and have very loose structure.

Amazon S3 [2] as a storage technology provides availability, scaling and reliability by synthesizing technologies. [3] Did performance comparison between MongoDB, Redis, couchbase, Cassandra and HBase. [4,5] describe a lab-based benchmark which uses a measurement tool named YCSB(Yahoo Cloud Serving Benchmark).Write performance of HBase is improved by using a memory and Cassandra by using a log disk. [6,7] Presents NoSQL benchmark and capabilities in dataset generation and workload with advanced YCSB++ to benchmark advanced features of column store databases. [8,9] compares and tests MongoDB and Cassandra based of main characteristics such as data loading, only read, read –modify-write, only updates, reads and updates using YCSB as measurement tool. [10] Proposed the brief description on MongoDB and CouchBase and compare insertion and retrieval time of databases to insert or retrieve various size images in databases using Java as front end tool. [11] analyzed the performance of CouchDB and Elasticsearch based on insertion, deletion, updating and selection operations and conclude that CouchDB works efficiently on insertion, deletion and update operation but Elasticsearch works much better in case of selection operation. [12] Compares NoSQL databases based on few features such as Replication, Storage type, CAP features, and Map Reduce and also did time comparison between insertions, deletion, updating operation. [13] Summarized main features of three databases named BigTable, DynamoDB, and Cassandra and do comparison and contrast between them.

III.COMPARITIVE ANALYSIS OF COLUMN STORE NOSQL DATABASES

Table1: The Comparison and contrast of column store databases

Column Store Databases	Cassandra	DynamoDB	BigTable	Hbase	Hyper Table
Development language	Java	Java, Net, Perl, JavaScript, C#	JACOB	Java	C++
Storage type	Wide column store	Hybrid: Document store Key-value store	Wide column store	Wide column store	Wide Column Store
Developer	Apache Software Foundation (2008)	Amazon (2012)	Google (2015)	Power Set (2007), Apache Software Foundation (2008)	Zvents Inc. (2008)
Influences/Sponsors	Dynamo Facebook/Digg/Rackspace	Amazon	BigTable	BigTable	Google's BigTable
Protocol	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP
Transactions	No (Local-Atomicity and isolation are supported for single operations)	ACID	Atomic single-row operations	Single row ACID (across millions of columns)	No
Replication	Selectable replication factor	Yes	Internal replication in Colossus, and regional replication between two clusters in different zones	Master-master replication, Master-slave replication	Selectable replication factor on file system level
Concurrency	Two-Phase Locking (Deadlock	Optimistic Concurrency Control (OCC)	No	Multi-version Concurrency Control (MVCC)	Multi-Version Concurrency Control (MVCC)

	Prevention), Optimistic Concurrency Control (OCC)				
Triggers	Yes	Yes	No	Yes	No
CAP Theorem	High Availability, Partition tolerance.	High Availability, Partition tolerance	Consistency, Partition tolerance	Consistency, Partition tolerance	Consistency, Partition tolerance
Operating System/Platform	Cross-platform	Cross-platform	Google Cloud Platform	Cross-platform	Cross-platform
Data Storage	Disk	Binary-Value Object	Disk	Disk	Disk
Persistence	Yes	Yes	Yes	Yes	Yes
High Availability	Yes distributed	Yes	No	No	No
Rack locality awareness	Yes (Inherited from Hadoop)	No	Yes	Yes (Inherited from Hadoop)	Yes (Inherited from Hadoop)
License type	Open Source Apache 2.0	Commercial	Commercial	Open Source Apache 2.0	Open Source (GNU General Public License 2.0)
Map reduce	Yes	No	Yes	Yes	Yes
Consistency	Eventually consistent	Eventual Consistency, Immediate Consistency	Immediate consistency (for a single cluster), Eventual consistency (for two or more replicated clusters)	Eventual Consistency or Immediate Consistency	Immediate Consistency
Querying	Cassandra Query Language (CQL)	SQL	(Map Reduce) 1. Look Up (Read a Single Row) 2. Scan (Read a subset of rows) 3. Write 4. Delete 5.Customized Scripts	(Map Reduce) Get, Put, Scan and Delete. DDL operations, e.g., Create.	HyperTable Query Language (HQL)
Partitioning scheme	Consistent Hashing, Sharding	Sharding	Sharding	Sharding	Sharding
Replication mode	Async	Async	Async	Async	Async
Scalability	Liner scalability	Incremental (Massive and seamless scalability)	High massively scalable	High scalability	High scalability
Database applicability	Facebook, Instagram, eBay, Netflix etc.	The web, social, mobile apps.	Applicable for google projects and products	Medical, Sports, Oil and Petroleum, E-Commerce, Social networking, Web	analytics, sorted URL lists, messaging applications
API	Get, Put	RESTful HTTP API (Put, get)	gRPC (using protocol buffers) API HappyBase (Python library) HBase compatible API (Java)	Java Client API and Thrift/REST API	C++ API Thrift
Data sets (Real time processing)	Structured and unstructured	Structured and unstructured	Structured data	Structured and unstructured	Structured and unstructured
System Orientation	Shared-Nothing (Symmetric)	Shared-Nothing (Symmetric)	Symmetric	Shared Disk	Shared Disk
Storage Architecture	Disk-oriented	Disk-oriented	Disk-oriented	Disk-oriented	Disk-oriented
System isolation	Serializable	Read, Uncommitted Read, Committed Repeatable Read	No	Read Uncommitted, Read Committed	Snapshot isolation

Data Scheme	Scheme free	Scheme free	Scheme free	schema-free, schema definition possible	Scheme free
Foreign keys	Not Supported	Not Supported	Not Supported	Not Supported	Not Supported
Joins	Not Supported	Not Supported	Not Supported	Not Supported	Not supported
Stored Procedures	Not Supported	Not Supported	Not Supported	Not Supported	Not Supported
Views	Not Supported	Not Supported	Not Supported	Not Supported	Not Supported
Logging	Logical Logging, Physical Logging, Physiological Logging, Command Logging	Not Supported	Physical logging	Logical Logging	Not Supported
Indexes	Skip List, Hash Table, BitMap	Not Supported	Not Supported	B+ Tree	Not Supported
Checkpoints	Non-Blocking, Consistent, Blocking, Fuzzy	Not Supported	Not Supported	Non-blocking	Consistent
Query Interface	Custom API	Custom API, SQL, Command-line / Shell	Custom API	Custom API	Custom API, Command-line / Shell
Cloud-based only	No	Yes	Yes	No	No
User Concept	Access rights for users can be defined per object	Access rights for users and roles can be defined via the AWS Identity and Access Management (IAM)	Access rights for users, groups and roles based on Google Cloud Identity and Access Management (IAM)	Access Control Lists (ACL) for RBAC, integration with Apache Ranger for RBAC & ABAC	No
Durability	Yes	Yes	Yes	Yes	Yes
Server-side scripts	No	No	No	Yes (Coprocessors in Java)	no
Supported programming languages	C# C++ Clojure Erlang Go Haskell Java JavaScript info Perl PHP Python Ruby Scala	Net ColdFusion Erlang Groovy Java JavaScript Perl PHP Python Ruby	C# C++ Go Java JavaScript (Node.js) Python	C C# C++ Groovy Java PHP Python Scala	C++ Java Perl PHP Python Ruby

IV.CONCLUSION

In this paper we describe structure of column store databases and their benefits and did comparative analysis of various column-store databases. We found that Cassandra and DynamoDB gives high Availability and Partition tolerance but HBase, BigTable and HyperTable gives Consistency and Partition tolerance. BigTable doesn't support concurrency control. Cassandra supports Master-Slave Architecture and HBase support Hadoop Distributive Architecture. Today Facebook and other social networking websites prefer Cassandra over HBase because of its availability, open source, minimal administration, no SPoF (Single Point of Failure) and provide security in every financial transaction. Companies like Bloomberg, Bank of America, Verizon and much more using HBase. HBase is good at intensive reads, whereas Cassandra is good at writes. Cassandra Lacks data consistency while HBase lacks data availability. Each database has its own advantages and disadvantages.

REFERENCES

[1] Naheman W, Wei J. Review of NoSQL databases and performance testing on HBase. InProceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC) 2013 Dec 20 (pp. 2304-2309). IEEE.

[2] DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W. Dynamo: amazon's highly available key-value store. InACM SIGOPS operating systems review 2007 Oct 14 (Vol. 41, No. 6, pp. 205-220). ACM.

[3] Tang E, Fan Y. Performance comparison between five NoSQL databases. In2016 7th International Conference on Cloud Computing and Big Data (CCBD) 2016 Nov 16 (pp. 105-109). IEEE

[4] Tudorica BG, Bucur C. A comparison between several NoSQL databases with comments and notes. In2011 RoEduNet international conference 10th edition: Networking in education and research 2011 Jun 23 (pp. 1-5). IEEE.

[5] Cooper BF, Silberstein A, Tam E, Ramakrishnan R, Sears R. Benchmarking cloud serving systems with YCSB. InProceedings of the 1st ACM symposium on Cloud computing 2010 Jun 10 (pp. 143-154). ACM.

[6] Reniers V, Van Landuyt D, Rafique A, Joosen W. On the state of nosql benchmarks. InProceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion 2017 Apr 18 (pp. 107-112). ACM.

[7] Patil S, Polte M, Ren K, Tantisiriroj W, Xiao L, López J, Gibson G, Fuchs A, Rinaldi B. YCSB++: benchmarking and performance debugging advanced features in scalable table stores. InProceedings of the 2nd ACM Symposium on Cloud Computing 2011 Oct 26 (p. 9). ACM.

[8] Abramova V, Bernardino J. NoSQL databases: MongoDB vs cassandra. InProceedings of the international C* conference on computer science and software engineering 2013 Jul 10 (pp. 14-22). ACM.

[9] Jayathilake D, Sooriaarachchi C, Gunawardena T, Kulasuriya B, Dayaratne T. A study into the capabilities of NoSQL databases in handling a highly heterogeneous tree. In2012 IEEE 6th International Conference on Information and Automation for Sustainability 2012 Sep 27 (pp. 106-111). IEEE.

[10] Chopade MR, Dhavase NS. Mongodb, couchbase: performance comparison for image dataset. In2017 2nd International Conference for Convergence in Technology (I2CT) 2017 Apr 7 (pp. 255-258). IEEE.

[11] Gupta S, Rani R. A comparative study of elasticsearch and CouchDB document oriented databases. In2016 International Conference on Inventive Computation Technologies (ICICT) 2016 Aug 26 (Vol. 1, pp. 1-4). IEEE.

[12] Kumar KS, Mohanavalli S. A performance comparison of document oriented NoSQL databases. In2017 International Conference on Computer, Communication and Signal Processing (ICCCSP) 2017 Jan 10 (pp. 1-6). IEEE.

[13] Kalid S, Syed A, Mohammad A, Halgamuge MN. Big-data NoSQL databases: A comparison and analysis of “Big-Table”,“DynamoDB”, and “Cassandra”. In2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(2017 Mar 10 (pp. 89-93). IEEE.

[14] https://aws.amazon.com/dynamodb/

[15] http://cassandra.apache.org/

[16] https://cloud.google.com/bigtable/

[17] https://hbase.apache.org/