Column Based NoSQL Database, Scope and Future

Dr. Raghav Mehra
Associate Professor and Dean,
Bhagwant Institute of Technology, Bhagwant Puram,
Muzaffarnagar U.P.

Nirmal Lodhi[2] & Ram Babu[3]
Ph.D. Research Scholar
Bhagwant University, Ajmer, India.

Received Nov. 18, 2015 Accepted Dec. 04, 2015

ABSTRACT

Column-oriented or column based database systems, also known as column-stores, have an important demand in the past few years. Basically, it is about storing each database column separately so that the attributes belonging to the same column would be stored contiguously, compressed and densely-packed in the disk. This method has advantages in reading the records faster as compared to classical row stores in which every row are stored one after another in the disk. These databases are more suitable for data warehousing system to get analysis done faster as data is stored in columnar form. Indexes are much faster in column oriented databases which results in faster data retrieval and hence data analysis. This is an alternate database technology over row oriented database systems. To further improve the read performance of system, there are different technologies based on which new DBMS has been designed and build up, discussed in details in the paper. I would detail only few of the NoSQL DBMS which are available in market.

Key words: Column based database, performance, data warehouse, DBMS, NoSQL, Cassandra, MongoDB, HBase, Couchbase

Column-oriented Database

A *column-oriented DBMS* is a DBMS that stores data tables as sections of columns of data rather than as rows of data. In comparison, most relational DBMSs store data in rows. This column-oriented DBMS has advantages for data warehouses, clinical data analysis, CRM systems, and library card catalogs, and other ad hoc inquiry systemswhere aggregates are computed over large numbers of similar data items.

It is possible to achieve some of the benefits of column-oriented and row-oriented organization with any DBMSs. Denoting one as column-oriented refers to both the ease of expression of a column-oriented structure and the focus on optimizations for column-oriented workloads. This approach is in contrast to *row-oriented or row store* databases and with correlation databases, which use a value-based storage structure.

Column-oriented storage is closely related to database normalization due to the way it restricts the database schema design. However, it was often found to be too restrictive in practice, and thus many column-oriented databases such as Google's BigTable do allow column groupsto avoid frequently needed joins.

A column-oriented database serializes all of the values of a column together, then the values of the next column, and so on. For our example table, the data would be stored in this fashion:
10:001,12:002,11:003,22:004;
Smith:001,Jones:002,Johnson:003,Jones:004;
Joe:001,Mary:002,Cathy:003,Bob:004;
40000:001,50000:002,44000:003,55000:004;

Here any one of the columns more closely matches the structure of an index in a row-based system. This may cause confusion that can lead to the mistaken belief a column-oriented store is really just a row-store with an index on every column. However, it is the mapping of the data that differs dramatically. In a row-oriented indexed system, the primary key is the rowid that is mapped to indexed data. In the column-oriented system, the primary key is the data, mapping back to rowids. This may seem subtle, but the difference can be seen in this common modification to the same store:
…; Smith:001;Jones:002,004;Johnson:003;…

As two of the records store the same value, "Jones", it is possible to store this only once in the column store, along with pointers to all of the rows that match it. For many common searches, like "find all the people with the last name Jones", the answer is retrieved in a single operation. Other operations, like counting the number of matching records or performing math over a set of data, can be greatly improved through this organization.

Whether or not a column-oriented system will be more efficient in operation depends heavily on the workload being automated. It would appear that operations that retrieve data for objects would be slower, requiring numerous disk operations to collect data from multiple columns to build up the record. However, these whole-row operations are generally rare. In the majority of cases, only a limited subset of data is retrieved. In a rolodex application, for instance, operations collecting the first and last names from many rows in order to build a list of contacts is far more common than operations reading the data for any single address. This is even truer for writing data into the database, especially if the data tends to be "sparse" with many optional columns. For this reason, column stores have demonstrated excellent real-world performance in spite of many theoretical disadvantages.

This is a simplification. Moreover, partitioning, indexing, caching, views, OLAP cubes, and transactional systems such as write ahead logging or multiversion concurrency control all dramatically affect the physical organization of either system. That said, online transaction processing (OLTP)-focused RDBMS systems are more row-oriented, while online analytical processing (OLAP)-focused systems are a balance of row-oriented and column-oriented.

Historyof Column-oriented Database

A relational database management system provides data that represents a two-dimensional table, of columns and rows. For example, a database might have this table:

RowId	EmpId	Lastname	Firstname	Salary
1	10.00	Smith	Joe	40000
2	12.00	Jones	Mary	50000
3	11.00	Johnson	Cathy	44000
4	22.00	Jones	Bob	55000

Table: 1

This simple table includes an employee identifier (EmpId), name fields (Lastname and Firstname) and a salary (Salary). This two-dimensional format exists only in theory. In practice, storage hardware requires the data to be serialized into one form or another.

The most expensive operations involving hard disks are seeks. In order to improve overall performance, related data should be stored in a fashion to minimize the number of seeks. This is known as locality of reference, and the basic concept appears in a number of different contexts. Hard disks are organized into a series of blocks of a fixed size, typically enough to store several rows of the table. By organizing the data so rows fit within the blocks, and related rows are grouped together, the number of blocks that need to be read or sought is minimized.

Advantages of Column-oriented DBMS

Comparisons between row-oriented and column-oriented data layouts are typically concerned with the efficiency of hard-disk access for a given workload, as seek time is incredibly long compared to the other delays in computers. Sometimes, reading a megabyte of sequentially stored data takes no more time than one random access.[6] Further, because seek time is improving much more slowly than CPU power this focus will likely continue on systems that rely on hard disks for storage. Following is a set of oversimplified observations which attempt to paint a picture of the trade-offs between column- and row-oriented organizations. Unless, of course, the application can be reasonably assured to fit most/all data into memory, in which case huge optimizations are available from in-memory database systems.

1. Column-oriented organizations are more efficient when an aggregate needs to be computed over many rows but only for a notably smaller subset of all columns of data, because reading that smaller subset of data can be faster than reading all data.
2. Column-oriented organizations are more efficient when new values of a column are supplied for all rows at once, because that column data can be written efficiently and replace old column data without touching any other columns for the rows.
3. Row-oriented organizations are more efficient when many columns of a single row are required at the same time, and when row-size is relatively small, as the entire row can be retrieved with a single disk seek.

4. Row-oriented organizations are more efficient when writing a new row if all of the row data is supplied at the same time, as the entire row can be written with a single disk seek.

In practice, row-oriented storage layouts are well-suited for OLTP-like workloads which are more heavily loaded with interactive transactions. Column-oriented storage layouts are well-suited for OLAP-like workloads such as data warehouses which typically involve a smaller number of highly complex queries over all data.

What is No SQL Database

NoSQL encompasses a wide variety of different database technologies that were developed in response to a rise in the volume of data stored about users, objects and products, the frequency in which this data is accessed, and performance and processing needs. Relational databases, on the other hand, were not designed to cope with the scale and agility challenges that face modern applications, nor were they built to take advantage of the cheap storage and processing power available today.

Background of NoSQL Database

The term *NoSQL* was used by Carlo Strozzi in 1998 to name his lightweight, Strozzi NoSQL open-source relational database that did not expose the standard SQL interface, but was still relational.His NoSQL RDBMS is distinct from the circa-2009 general concept of NoSQL databases. Strozzi suggests that, as the current NoSQL movement departs from the relational model altogether; it should therefore have been called more appropriately NoREL(No Relational').

Johan Oskarsson of Last.fm reintroduced the term *NoSQL* in early 2009 when he organized an event to discuss "open source distributed, non-relational databases. The name attempted to label the emergence of an increasing number of non-relational, distributed data stores, including open source clones of Google's BigTable/MapReduce and Amazon's Dynamo. Most of the early NoSQL systems did not attempt to provide atomicity, consistency, isolation and durability guarantees, contrary to the prevailing practice among relational database systems.

Most Popular NoSQL Database in Market.

The most popular NoSQL databases in market are

1. MongoDB
2. Apache Cassandra
3. HBase
4. Couchbase
5. Solr
6. ElasticSearch
7. Splunk
8. Memcached

Types of NoSQL Database

There have been various approaches to classify NoSQL databases, each with different categories and subcategories, some of which overlap. A basic classification based on data model, with examples:

- **Document databases** pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.
- **Graph stores** are used to store information about networks, such as social connections. Graph stores include Neo4J and HyperGraphDB.

- **Key-value stores** are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or "key"), together with its value. Examples of key-value stores are Riak and Voldemort. Some key-value stores, such as Redis, allow each value to have a type, such as "integer", which adds functionality.
- **Wide-column stores** such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

A more detailed classification is the following, based on one from Stephen Yen

Type	Examples of this type
Key-Value Cache	Coherence, eXtreme Scale, GigaSpaces, GemFire, Hazelcast, Infinispan, JBoss Cache, Memcached, Repcached,Terracotta, Velocity
Key-Value Store	Flare, Keyspace, RAMCloud, SchemaFree, Hyperdex, Aerospike
Key-Value Store (Ordered)	Actord, FoundationDB, Lightcloud, LMDB, Luxio, MemcacheDB, NMDB, Scalaris, TokyoTyrant
Tuple Store	Apache River, Coord, GigaSpaces
Object Database	DB4O, Objectivity/DB, Perst, Shoal, ZopeDB,
Document Store	Clusterpoint, Couchbase, CouchDB, DocumentDB, Lotus Notes, MarkLogic, **MongoDB**, Qizx, XML-databases
Wide Columnar Store	BigTable, **Cassandra**, Druid, **HBase**, Hypertable, KAI, KDI, OpenNeptune, Qbase

Table: 2

Advantages of NoSQLDatabase

When compared to relational databases, NoSQL databases are more scalable and provide superior performance and their data model addresses several issues that the relational model is not designed to address:

- Large volumes of structured, semi-structured, and unstructured data
- Agile sprints, quick iteration, and frequent code pushes
- Object-oriented programming that is easy to use and flexible
- Efficient, scale-out architecture instead of expensive, monolithic architecture

Trending &Benchmarking NoSQL Databases: Cassandra vs. MongoDB vs. HBase vs. Couchbase

Understanding the performance behavior of a NoSQL database like Apache Cassandra™ under various conditions is critical. Conducting a formal proof of concept (POC) in the environment in which the database will run is the best way to evaluate platforms. POC processes that include the right benchmarks such as production configurations, parameters and anticipated data and concurrent user workloads give both IT and business stakeholder's powerful insight about platforms under consideration and a view for how business applications will perform in production.

Independent benchmark analyses and testing of various NoSQL platforms under big data, production-level workloads have been performed over the years and have consistently identified Apache Cassandra as the platform of choice for businesses interested in adopting NoSQL as the database for modern Web, mobile and IOT applications.

One benchmark analysis (Solving Big Data Challenges for Enterprise Application Performance Management) by engineers at the University of Toronto, which in evaluating six different data stores, found Apache Cassandra the "clear winner throughout our experiments". Also, End Point Corporation, a database and open source consulting company, benchmarked the top NoSQL databases including: Apache Cassandra, Apache HBase, Couchbase, and MongoDB using a variety of different workloads on AWS EC2.

The databases involved were:

Apache Cassandra: Highly scalable, high performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.

Apache HBase: Open source, non-relational, distributed database modeled after Google's BigTable and is written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS (Hadoop Distributed File System), providing BigTable-like capabilities for Hadoop.

MongoDB: Cross-platform document-oriented database system that eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas making the integration of data in certain types of applications easier and faster.

Couchbase: Distributed NoSQL document-oriented database that is optimized for interactive applications.

End Point conducted the benchmark of these NoSQL database options on Amazon Web Services EC2 instances, which is an industry-standard platform for hosting horizontally scalable services. In order to minimize the effect of AWS CPU and I/O variability, End Point performed each test 3 times on 3 different days. New EC2 instances were used for each test run to further reduce the impact of any "lame instance" or "noisy neighbor" effects sometimes experienced in cloud environments, on any one test.

NoSQL Database Performance Testing Results

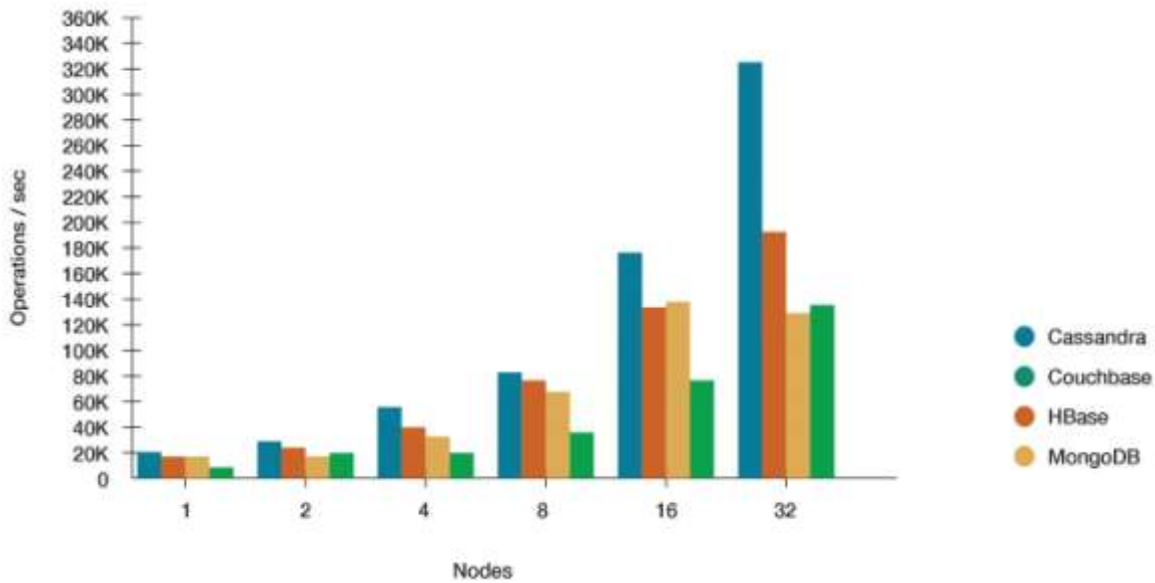
When it comes to performance, it should be noted that there is (to date) no single "winner takes all" among the top NoSQL databases or any other NoSQL engine for that matter. Depending on the use case and deployment conditions, it is almost always possible for one NoSQL database to outperform another and yet lag its competitor when the rules of engagement change. Here are a couple snapshots of the performance benchmark to give you a sense of how each NoSQL database stacks up.

Throughput by Workload

Each workload appears below with the throughput/operations-per-second (more is better) graphed vertically, the number of nodes used for the workload displayed horizontally, and a table with the result numbers following each graph.

Load process

For load, Couchbase, HBase, and MongoDB all had to be configured for non-durable writes to complete in a reasonable amount of time, with Cassandra being the only database performing durable write operations. Therefore, the numbers below for Couchbase, HBase, and MongoDB represent non-durable write metrics.



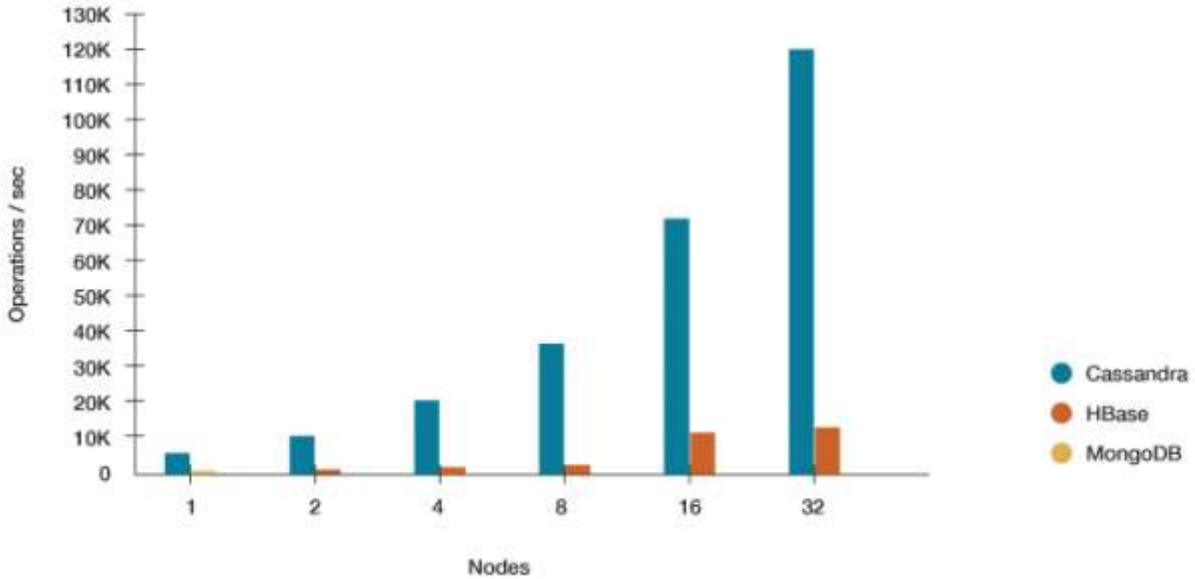
Graph: 1

Nodes	Cassandra	Couchbase	HBase	MongoDB
1	18,683.43	13,761.12	15,617.98	8,368.44
2	31,144.24	26,140.82	23,373.93	13,462.51
4	53,067.62	40,063.34	38,991.82	18,038.49
8	86,924.94	76,504.40	74,405.64	34,305.30
16	173,001.20	131,887.99	143,553.41	73,335.62
32	326,427.07	192,204.94	296,857.36	134,968.87

Table: 3

Mixed Operational and Analytical Workload

Note that Couchbase was eliminated from this test because it does not support scan operations (producing the error: “Range scan is not supported”).



Graph: 2

Nodes	Cassandra	HBase	MongoDB
1	4,690.41	269.30	939.01
2	10,386.08	333.12	30.96
4	18,720.50	1,228.61	10.55
8	36,773.58	2,151.74	39.28
16	78,894.24	5,986.65	377.04
32	128,994.91	8,936.18	227.80

Table: 4

NoSQL Database Performance

These performance metrics are just a few of the many that have solidified Apache Cassandra as the NoSQL database of choice for businesses needing a modern, distributed database for their Web, mobile and IOT applications. Each database option (Cassandra, HBase, Couchbase and MongoDB) will certainly shine in particular use cases, so it's important to test your specific use cases to ensure your selected database meets your performance SLA. Whether you are primarily concerned with throughput or latency, or more interested in the architectural benefits such as having no single point of failure or being able to have elastic scalability across multiple data centers and the cloud, much of an application's success comes down to its ability to deliver the response times Web, mobile and IOT customers expect.

As the benchmarks referenced here showcase, Cassandra's reputation for fast write and read performance, and delivering true linear scale performance in a masterless,

scale-out design, bests its top NoSQL database rivals in many use cases.

Scope and Future of NoSQL Database

The global NoSQL market is forecast to reach $3.4 Billion in 2020, representing a compound annual growth rate (CAGR) of 21% for the period 2015 – 2020.

The fledgling NoSQL marketplace is going through a rapid transition – from the predominantly community-driven platform development to a more mature application-driven market. Scaling up web infrastructure on NoSQL basis have proven successful for Facebook, Digg and Twitter. Successful attempts have been made to develop NOSQL applications in the biotechnology, defense and image/signal processing. Interest in using key-value pair (KVP) technology has reemerged to the point where the traditional RDMS vendors evaluate strategy of developing in-house NoSQL solutions and integrating them in current product offers. It will not take long before we'll see acquisitions driven by emerging NoSQL technology. The future deals will likely be made to better compete both in platform offering and in vertical market segments.

The worldwide NoSQL market is posed for two digit growth in the period 2015-2020. NoSQL is moving in to become a major player in database marketplace.

Conclusion

NoSQL is still relatively new, and while some have adopted it wholeheartedly, others range from reluctant to completely oppose to the new technology. There are a number of challenges that NoSQL still faces:

Lack of trained administrators and developers. The most senior developers and admins have made their livelihood writing code for and managing SQL databases. NoSQL may be uncharted territory for many

Many CIOs are cautious and avoid jumping into new technology that they feel might not be ready for primetime.

While some of the NoSQL databases might have even reached maturity, many CIOs want to see more analytic and transactional applications built around NoSQL and Big Data technology. They also want application development tools that allow them to build their own custom apps.

The future mostly looks bright for NoSQL, and it is clear that it has found a place for itself in the Big Data market alongside Hadoop. Over the next few years, expect to see more competing open source implementations, more startups, and more corporations partnering with startups and/or acquiring them. This will lead to more developers and administrators trained in NoSQL databases and ultimately more customer adoption.

References:-

1. O.S. Tezer: A Comparison Of NoSQL Database Management Systems And Models
2. VeronikaAbramova , Jorge Bernardino and Pedro Furtado, Polytechnic Institute of Coimbra - ISEC / CISUC, Coimbra, Portugal University of Coimbra – DEI / CISUC, Coimbra, Portugal : EXPERIMENTAL EVALUATION OF NOSQL DATABASES
3. Priyanka Sharma, CDAC : NOSQL DATABASES
4. http://www.datastax.com
5. https://www.mongodb.com
6. Market Research Media: NoSQL Market Forecast
7. Data Stax: NoSQL Performance Benchmarks
8. Sergey Sverchkov: Evaluating NoSQL Performance