Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Study on Basically Available, Scalable and Eventually Consistent NOSQL Databases

Dr.K.Chitra *
Dept of Computer Science
Govt Arts College
Melur, Madurai Dt, India

B.JeevaRani
Dept of Computer Science
Research Scholar, Bharathiar University
Coimbatore, India

Abstract— A relational database is a table-based data system where there is no scalability, minimal data duplication, computationally expensive table joins and difficulty in dealing with complex data. The problem with relations in relational database is that complex operations with large data sets quickly become prohibitively resource intense. Relational databases do not lend themselves well to the kind of horizontal scalability that's required for large-scale social networking or cloud applications. NOSQL has emerged as a result of the demand for relational database alternatives. The biggest motivation behind NOSQL is scalability. NOSQL is meant for the current growing breed of web applications that need to scale effectively. This paper analyzes the need of the next generation data storage which is the need of the current large-scale social networking or cloud applications. We also analyze the capabilities of various NOSQL models like BigTable, Cassandra, CouchDB, Dynamo and MongoDB.

Keywords— NOSQL, Scalability, Next Generation Data Storage

I. INTRODUCTION

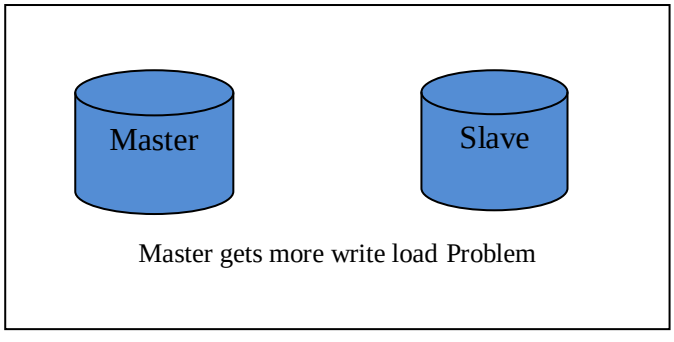
A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd. It supports a tabular structure for the data, with enforced relationships between the tables. Most popular commercial and open source databases currently in use are based on the relational model. The problem with RDBMS is not that they do not scale, it's that they are incredibly hard to scale. The most popular RDBMS are Microsoft SQL Server, DB2, Oracle, MYSQL etc.

Many Web applications simply do not need to represent data as a set of related tables that means all applications need not be a traditional relational database management system (RDBMS) that uses SQL to perform operations on data. Rather, data can be stored in the form of objects, graphs, documents and retrieved using a key. For example, a user profile can be represented as an object graph (such as pojo) with a single key being the user id. Another example: documents or media files can be stored with a single key with indexing of metadata handling by a separate search engine.

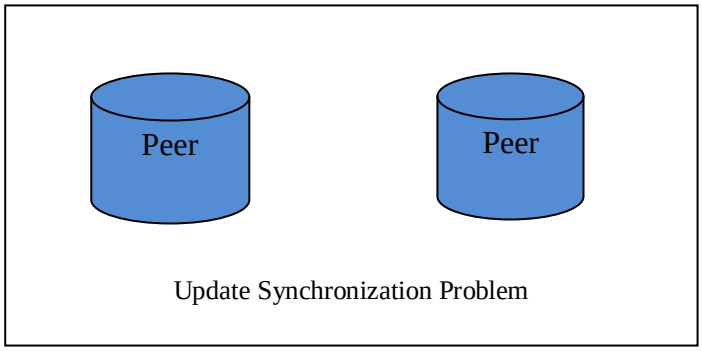
These forms of data storage are not relational and lack SQL, but they may be faster than RDBMS because they do not have to maintain indexes, relationships, constraints and parse SQL. Technology like that has existed since the 1960s (consider, for example, IBM's VSAM file system).

Relational databases are able to handle millions of products and service very large sites. However, it is difficult to create redundancy and parallelism with relational databases, so they become a single point of failure. In particular, replication is not trivial. To understand why, consider the problem of having two database servers that need to have identical data. Having both servers for reading and writing data makes it difficult to synchronize changes. Having one master server and another slave is bad too, because the master has to take all the heat when users are writing information. So as a relational database grows, it becomes a bottleneck and the point of failure for the entire system. As mega e-commerce sites grew over the past decade they became aware of this issue - adding more web servers does not help because it is the database that ends up being a problem.

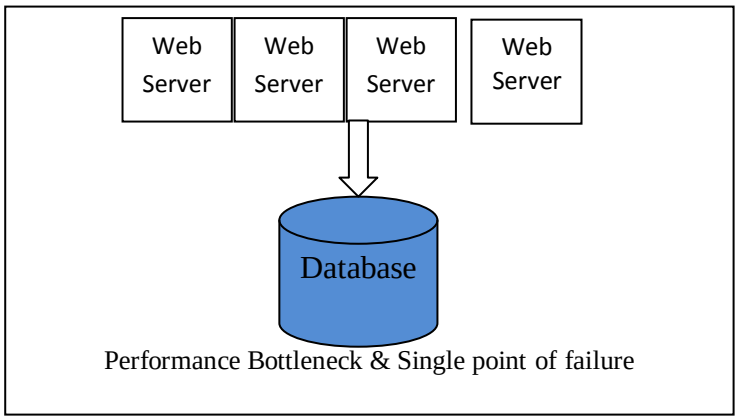
A. Master Slave Setup



B. Multiple Peers setup



C. Single Database Server setup



II. NEXT GENERATION DATA STORAGE MODEL

Trends in computer architectures and modern web-scale databases are pressing databases in a direction that requires:

- storage of non-relational data, which are distributed to be available even when the nodes are crashed
- open-source
- effective scalability (Horizontal & Vertical)
- schema-free
- replication support
- easy API
- eventual consistency

NOSQL-style data stores attempt to address these requirements. It is the future of database technology that is suitable particularly for web applications that need to scale effectively. Many NOSQL products store data as BLOBs. Actually, NOSQL is an offhand term for a collection of technologies, some of which are DBMS (Database Management Systems) and some of which are not. All they have in common is:

- They have something to do with managing data
- They are not relational DBMS, if by "Relational DBMS" you mean "DBMS that want you to talk to them in SQL."
- select fun, profit from real_world where relational=false;
-

D. NOSQL (Not Only SQL)

The NoSQL movement is a combination of an architectural approach for storing data and software products that can store data without using SQL. Hence, the term NoSQL. The NOSQL products that store data using keys are called Key-Value (KV) stores. Because these KV stores are not relational and lack SQL they may be faster than RDBMS. The downside of NOSQL is that you cannot easily perform queries against related data. Prominent closed-source examples are Google's BigTable and Amazon's Dynamo. Several open-source variants exist including Facebook's Cassandra, Apache HBase, CouchDB LinkedIn's Project Voldemort and many others. Apache's open source CouchDB offers a new method of storing data, in what is referred to as a schema-free document-oriented database model.

1) *The technology:* There are three popular types of NoSQL databases.

1.1) *Key-value stores*: As the name implies, a key-value store is a system that stores values indexed for retrieval by keys. These systems can hold structured or unstructured data. Amazon's SimpleDB is a Web service that

provides core database functions of information indexing and querying in the cloud. It provides a simple API for storage and access. Users pay only for the services they use.

1.2) *Column-oriented databases*: Rather than store sets of information in a heavily structured table of columns and rows with uniform sized fields for each record, as is the case with relational databases, column-oriented databases contain one extendable column of closely related data. Facebook created the high-performance Cassandra to help power its website. The Apache Software Foundation developed Hbase, a distributed, open source database that emulates Google's Big Table.

1.3) *Document-based stores*: These databases store and organize data as collections of documents, rather than as structured tables with uniform sized fields for each record. With these databases, users can add any number of fields of any length to a document.

E. NOSQL models for Data Storage

Organizations that collect large amounts of unstructured data are increasingly turning to non-relational databases, now frequently called NoSQL databases.

1) *Bigtable: A Distributed Storage System for Structured Data*

Bigtable is Google's internal database system. Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size i.e. petabytes (1 petabyte = 1.12589991 × 1015 bytes) of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance. These applications place very different demands on Bigtable, both in terms of data size (from URLs to web pages to satellite images) and latency requirements (from backend bulk processing to real-time data serving). Despite these varied demands, Bigtable has successfully provided a flexible, high-performance solution for all of these Google products.

A Bigtable is a sparse, distributed, persistent multidimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes. Each cell in a Bigtable (like field in DBMS) can contain multiple versions of the same data; these versions are indexed by timestamp (Microseconds). Bigtable timestamps are 64-bit integers. Different versions of a cell are stored in decreasing timestamp order, so that the most recent versions can be read first. Bigtable depends on a cluster management system for scheduling jobs, managing resources on shared machines, dealing with machine failures, and monitoring machine status.

2) *Cassandra: A Decentralized, highly scalable, eventually consistent DB*

Cassandra is highly scalable second-generation distributed database. Cassandra was open sourced by Facebook in 2008 and is currently being developed as an Apache Incubator project. The system offers a fault tolerant, high availability, decentralized store for data which can be scaled up by adding hardware nodes to the system. Cassandra implements an "eventually consistent" model which trades-off consistency of data stores in the system for availability. Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime. Cassandra is in use at Rackspace, Digg, Facebook, Twitter, Cisco, Mahalo, Ooyala, and more companies that have large, active data sets. The largest production cluster has over 100 TB of data in over 150 machines.

3) *CouchDB – Extremely scalable, highly available and reliable Storage System*

CouchDB, is a free and open source document-oriented database written in the Erlang programming language for its emphasis on fault tolerance, accessible using a RESTful JavaScript Object Notation (JSON) API. The term "Couch" is an acronym for "Cluster Of Unreliable Commodity Hardware", reflecting the goal of CouchDB being extremely scalable, offering high availability and reliability, even while running on hardware that is typically prone to failure.

Hence CouchDB is:

- A document database server
- Ad-hoc and schema-free with a flat address space
- Distributed, featuring robust, incremental replication with bi-directional conflict detection and management.
- Highly available even if hardware fails
- Query-able and index-able, featuring a table oriented reporting engine that uses Javascript as a query language

CouchDB is Not

- A relational database
- A replacement for relational databases
- An object-oriented database. Or more specifically, meant to function as a seamless persistence layer for an OO programming language

4) *Dynamo - A Distributed Storage System for Amazon website*

Dynamo is an internal technology developed at Amazon to address the need for an incrementally scalable, highly-available key-value storage system. The technology is designed to give its users the ability to trade-off cost, consistency, durability and performance, while maintaining high-availability.

Amazon runs a world-wide e-commerce platform that serves tens of millions customers at peak times using tens of thousands of servers located in many data centers around the world. Reliability is one of the most important

requirements because even the slightest outage has significant financial consequences and impacts customer trust. In addition, to support continuous growth, the platform needs to be highly scalable.

Dynamo has a simple key/value interface, is highly available with a clearly defined consistency window, is efficient in its resource usage, and has a simple scale out scheme to address growth in data set size or request rates. Each service that uses Dynamo runs its own Dynamo instances.

5) MongoDB

MongoDB ("humongous") is a document database designed to be easy to work with, fast, and very scalable. It was also designed to be ideal for website infrastructure. It is perfect for user profiles, sessions, product information, and all forms of Web content (blogs, wikis, comments, messages, and more). It's not great for transactions or perfect durability, as required by something like a banking system. Good fits for MongoDB include applications with complex objects or real-time reporting requirements, and agile projects where the underlying database schema changes often. MongoDB does not suit software with complex (multiple objects) transactions.

5.1) *Inside MongoDB*

MongoDB is an interesting combination of modern Web usage semantics and proven database techniques. In some ways MongoDB is closer to MySQL than to other so-called "NoSQL" databases: It has a query optimizer, ad-hoc queries, and a custom network layer. It also lets you organize document into collections, similar to sql tables, for speed, efficiency, and organization.

To get great performance and horizontal scalability however, MongoDB gives something up: transactions. MongoDB does not support transactions that span multiple collections. You can do atomic operations on a single object, but you can't modify objects from two collections atomically.

MongoDB stores BSON, essentially a JSON document in an efficient binary representation with more data types. BSON documents readily persist many data structures, including maps, structs, associative arrays, and objects in any dynamic language. Using MongoDB and BSON, you can just store your class data directly in the database. MongoDB is also schema-free. You can add fields whenever you need to without performing an expensive change on your database. Adding fields is also quick, which is ideal for agile environments. You need not revise schemas from staging to production or have scripts ready roll changes back.

III. ANALYSIS OF NOSQL

In this section we compare and analyze the nonfunctional requirements of NOSQL databases named BigTable, Cassandra, CouchDB, Dynamo, MongoDB, HBase.

TABLE I
NONFUNCTIONAL REQUIREMENTS ANALYSIS OF NOSQL DATABASES

	DB Type	**Scalability**	**Availability**	**Performance**	**Reliability**
Bigtable	Key-Value store DB	Highly Scalable	Highly Available	High performance	Provides reliability at a massive scale
Cassandra	Column Oriented DB	Highly Scalable	High availability is achieved using replication	High Performance at massive scale	At massive scale is a very big challenge
CouchDB	JSON Document Oriented DB	Easily scalable and readily extensible	Highly Available	Loading speeds are better than retrieval speeds	Reliable and efficient system
Dynamo	Key-Value store DB	Incremental	Highly Available	Performance at massive scale is one of the biggest challenges	Reliability at massive scale is one of the biggest challenges
MongoDB	BSON Document Oriented DB	Scalable	High write availability	Excellent solution for short read	Avoid growing documents Unsafe Writes by default

F. Challenges of NOSQL

NOSQL databases face several challenges.

Overhead and complexity: NOSQL databases don't work with SQL; they require manual query programming, which can be fast for simple tasks but time-consuming for others. In addition, complex query programming for the databases can be difficult.

Reliability: NOSQL databases do not support ACID properties which are natively supported by relational databases. NOSQL databases thus do not natively offer the degree of reliability that ACID provides. If users want NOSQL databases to apply ACID restraints to a data set, they must perform additional programming.

Consistency: Because NOSQL databases do not natively support ACID transactions, they also could compromise consistency, unless manual support is provided. Not providing consistency enables better performance and scalability but is a problem for certain types of applications and transactions, such as those involved in banking.

Unfamiliarity with the technology: Most organizations are unfamiliar with NOSQL databases and thus may not feel knowledgeable enough to choose one or even to determine that the approach might be better for their purposes.

Limited Eco structure: Unlike commercial relational databases, many open source NOSQL applications do not yet come with customer support or management tools.

IV. CONCLUSION

Next Generation Databases mostly address some of the points: being nonrelational, distributed, open-source and horizontal scalable. The original intention has been modern web-scale databases. Often more characteristics apply as: schema-free, replication support, easy API, eventually consistency, and more. Hence the misleading term "NoSQL" is now translated to "Not Only SQL (NOSQL)". NOSQL databases generally process data faster than relational databases. Developers usually do not have their NOSQL databases supporting ACID properties, in order to increase performance, but this can cause problems when used for applications that require great precision. NOSQL databases are also often faster because their data models are simpler. "There's a bit of a trade-off between speed and model complexity, but it is frequently a tradeoff worth making," Because they do not have all the technical requirements that relational databases have. Major NOSQL systems are flexible enough to better enable developers to use the applications in ways that meet their needs.

REFERENCES

[1] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber, 'Bigtable: A Distributed Storage System for Structured Data', OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006.

[2] Hamilton, James, "Perspectives: One Size Does Not Fit All", 13 November 2009. 3. Lakshman, Avinash; Malik, Prashant, Cassandra - A Decentralized Structured Storage System. Cornell University, 13 November 2009.

[3] Chang, Fay; Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. Google. 13 November 2009.

[4] Kellerman, Jim. "HBase: structured storage of sparse data for Hadoop", 13 November 2009.

[5] Ian Eure, Looking to the future with Cassandra. Digg Technology Blog, September 2009

[6] Zhou Wei, Guillaume Pierre and Chi-Hung Chi. CloudTPS: Scalable Transactions for Web Applications in the Cloud. Technical report IR-CS-53, VU University Amsterdam, February 2010.

[7] http://www.allthingsdistributed.com/2007/12/eventually_consistent.

[8] http://s3.amzonaws.com/AllThings/Distributed/sosp/amazon-dynamo

[9] http://labs.google.com/papers/bigtable.html

[10] http://www.cs.brown.edu/~ugur/osfa.pdf

[11] http://www.readwriteweb.com/archives/amazon_dynamo.php

[12] http://nosql-database.org/

[13] http://couchdb.apache.org/

