NoSQL-
NO PROBLEM WITH 'BIG DATA'

Tabular data remains tabular and the spreadsheet is still a business's favourite data modeling tool. SQL is not going away any time soon. However until now we've been creative in working with and around the constraints of a typical relational datastore. NoSQL offers the chance to think differently about data and that is a tremendously exciting prospect.

A.K. HOTA
Technical Director
NIC Odisha
ak.hota@nic.in

D. MADAN PRABHU
S.O/Engineer-SB
NIC Odisha
madan.prabhu@nic.in

Edited by
R. GAYATRI

WHAT DO WE MEAN BY 'BIG DATA'

'Big data' is a term applied to data sets whose size is beyond the ability of commonly used software tools to capture, manage, and process the data within a tolerable elapsed time. The explosion of data volumes, sometimes needing real-time processing are often not lending itself to the structuring rules and processes implemented by RDBMS. To address this problem the solutions such as Google's Map-Reduce and the Hadoop open source solution etc have been developed which are widely known as 'NoSQL'.

WHAT IS NOSQL?

NoSQL is not the name of any particular database. It refers to a broad class of non-relational databases that differ from classical relational database management systems (RDBMS) in some significant aspects. They do not use SQL as their primary query language, instead providing access by means of Application Programming Interfaces (API).

NoSQL can be considered "Internet age" databases that are being used by Amazon, Facebook, Google and the like to address performance and scalability requirements that cannot be met by traditional relational databases. NoSQL databases and data-processing frameworks are primarily utilized because of their speed, scalability and flexibility. Because of these reasons, sometimes NoSQL even refers as 'Cloud Databases'.

Feature	Description
Schema-less	"Tables" don't have a pre-defined schema. Records have a variable number of fields that can vary from record to record. Record contents and semantics are enforced by applications.
Shared nothing architecture	Instead of using a common storage pool (e.g., SAN), each server uses its own local storage. This allows storage to be accessed at local disk speeds instead of network speeds, and it allows capacity to be increased by adding more nodes. Cost is also reduced since commodity hardware can be used.
Elasticity	Both storage and server capacity can be added on-the-fly by merely adding more servers. No downtime is required. When a new node is added, the database begins giving it something to do and requests to fulfill.

Sharding	Instead of viewing the storage as a monolithic space, records are partitioned into shards. Usually, a shard is small enough to be managed by a single server, though shards are usually replicated. Sharding can be automatic (e.g., an existing shard splits when it gets too big), or applications can assist in data sharding by assigning each record a partition ID.
Asynchronous replication	Compared to RAID storage (mirroring and/or striping) or synchronous replication, NoSQL databases employ asynchronous replication. This allows writes to complete more quickly since they don't depend on extra network traffic. One side effect of this strategy is that data is not immediately replicated and could be lost in certain windows. Also, locking is usually not available to protect all copies of a specific unit of data.
BASE instead of ACID	NoSQL databases emphasize performance and availability. This requires prioritizing the components of the CAP theorem that tends to make true ACID transactions implausible. BASE (Basically Available, Soft state, Eventual consistency) means that given a sufficiently long period of time over which no changes are sent, all updates can be expected to propagate eventually through the system and all the replicas will be consistent

FLAVORS OF NOSQL

Today's open source NoSQL data stores fall into several different categories as follows:

KEY VALUE STORES

Enables populating a database using keys, such as "Joe" and associated values such as his address

Examples: Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB

Typical applications: Content caching

Strengths: Fast lookups

Weaknesses: Stored data has no schema

GRAPH DATABASES

This kind of database is designed for data whose relations are well represented as a graph (elements interconnected with an undetermined number of relations between them). The kind of data could be social relations, public transport links, road maps or network topologies

Examples: Neo4J, InfoGrid, Infinite Graph

Typical applications: Social networking, Recommendations

Strengths: Graph algorithms e.g. shortest path, connectedness, n degree relationships, etc.

Weaknesses: Has to traverse the entire graph to achieve a definitive answer. Not easy to cluster.

TABULAR STORES

Similar to a relational system, designed to hold data in a spreadsheet-like format where entries can be searched and retrieved

Examples: Cassandra, HBase, Riak

Typical applications: Distributed file systems

Strengths: Fast lookups, good distributed storage of data

Weaknesses: Very low-level API

DOCUMENT DATABASES

Similar to a content management system, designed to store documents and index the documents for quick access

Examples: CouchDB, MongoDb

Typical applications: Web applications

Strengths: Tolerant of incomplete data

Weaknesses: Query performance, no standard query syntax

XML DATABASES

Specifically for XML content, using a query language for XML documents, such as Xquery

Examples: Exist, Oracle, MarkLogic

Typical applications: Publishing

Strengths: Mature search technologies, Schema validation

Weaknesses: No real binary solution, easier to re-write documents than update them

OBJECT STORES

Designed specifically to store and retrieve objects based on their associated metadata

Examples: Oracle Coherence, db4o, ObjectStore, GemStone, Polar

Typical applications: Finance systems

Strengths: Matches OO development paradigm, low-latency ACID, mature technology

Weaknesses: Limited querying or batch-update options

ADVANTAGES OF NOSQL DATABASES

- Supports BigData
- Elastic scaling
- Fast key-value access
- Schema migration without downtime
- Easier maintainability, administration and operations of Database
- No single point of failure

- Supports Agility
- Programmer ease of use
- Distributed systems support and much more.

POPULAR INCARNATIONS OF NOSQL DATABASES

Apache Cassandra

Apache Cassandra is an open-source, distributed database-management system designed to handle very large amounts of data spread out across many commodity servers while providing a high degree of service availability with no single point of failure. It is particularly fast at write operations as opposed to reads and might therefore lend itself best to applications that require analysis of large sets of data with write-backs.

Hadoop / HBase

HBase is an open-source, distributed database modeled after Google's BigTable. HBase technologies are not strictly a data-store, but generally work closely with a NoSQL database to accomplish highly scalable analyses. HBase scales linearly with the number of nodes and can quickly return queries on tables consisting of billions of rows and millions of columns. India UID, Aadhar Project uses Haddop for handling 1.22 billion citizen's BigData.

BigTable

BigTable can be defined as a sparse, distributed, multi-dimensional sorted map. BigTable is designed to scale into the petabyte range (a petabyte is equivalent to 1 million gigabytes) across hundreds or thousands of machines and to make it easy to add more machines to the system and start taking advantage of those resources automatically without any reconfiguration.

MongoDB

MongoDB is open source document-oriented NoSQL database system. Instead of storing data in tables as is done in a "classical" relational database, MongoDB stores structured data as JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.

Coherence and Ehcache

Coherence and Ehcache are equipped with In-Memory caches. Coherence is in heavy use at financial industries where network latency (the time it takes to cross a network connection from sender to receiver) is a factor.

Possible applications of NoSql Databases

NoSQL databases should generally be considered as potential options when any high-intensity computation or analysis of large data sets is required, especially when performing real-time analysis. This can easily make their use in many sectors e.g.

- Nationwide Medical Prescription, a comprehensive drug treatment history of every citizen, available to patients, doctors, and medical systems
- Open Government Initiative platform, where all datasets are available for public. like data.gov.in
- Managing GIS data with NoSQL
- RTI portal
- e Voting
- Fraud detection by comparing transactions to known patterns in real-time.

CONCLUSION

Relational databases, especially the columnar variety, do not generally perform well on updates. As a result, a NoSQL database might present itself as a viable alternative in cases where massive updates are required. In situations involving variable-record templates or sparse data, NoSQL document databases can offer a welcome alternative.

REFERENCES

http://en.wikipedia.org/wiki/NoSQL
http://nosql-database.org/
http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html
http://www.thoughtworks.com/articles/nosql-comparison