Emergence of NoSQL Platforms for Big Data Needs

E

Jyotsna Talreja Wassan
Maitreyi College, University of Delhi, India

INTRODUCTION

Big data is revolutionizing world in the age of Internet. The wide variety of areas like online businesses, electronic health management, social networking, demographics, geographic information systems, online education etc. are gaining insight from *big data principles*. Big data is comprised of heterogeneous datasets which are too large to be handled by traditional relational database systems. An important reason for explosion of interest in big data is that it has become cheap to store volumes of data and there is a major rise in computation capacity. To extract valuable patterns from big data, one needs to choose a right platform for capturing, organizing, searching and analyzing the context of voluminous data in combination with traditional enterprise database management systems.

Different platforms supporting big data management by many software organizations enable easy use of services. These platforms mainly focus on data storage, management, processing, and distribution and on data analytics. Various NoSQL data stores like Cassandra, MongoDB and Hadoop HBASE etc. are in use today to acquire, manage, store and query big data. NoSQL databases are inherently schema-less and permit records to have variable number of fields, making them distinct from other non-relational databases like hierarchical databases and object-oriented databases. These are highly scalable and well suited for dynamic data structures. NoSQL data is characterized by being basically available and eventually consistent. The frameworks like MapReduce, Dryad etc. support processing of large amounts of data in parallel and hence the management and analysis of big data. The technologies like GNU R and Apache MAHOUT are also useful in exploring big data for finding relevant valuable patterns. This article aims at giving an overview of the rationales behind NoSQL movement as well as various big data platforms useful in today's competitive world.

DOI: 10.4018/978-1-4666-5202-6.ch074

BACKGROUND

In 1970's big meant megabytes, subsequently with the increasing data needs, it grew to gigabytes and terabytes and further to zettabytes with the increase in digital information. The traditional world of relational database systems like Oracle RDBMS etc. faced challenges in storing large quantities of data and needed to scale beyond the storage and/or processing capabilities of a single large computer system. Many efforts have been made to store and manage data being generated from everywhere on the Web. Several database management systems were proposed on the basis of master/slave, cluster computing or partitioning architecture like IBM DB2 partitioning, VoltTB etc.

However, the problems in reliance on shared facilities and resources (CPU, Disk, and Processors), scalability and complex administration limitations, augmented by lack of support for critical requirements, led to development of SHARED NOTHING architectures (Strauch,C.,2011; Lee, S.,2011) in 1980's. These systems focused on parallel and distributed data computation and solved big data problems using parallel computations. By 90's, even these solutions faced challenges in running OLTP and queries due to data overload. To provide solutions to these problems, Google responded with its GFS(Ghemawat, S., Gobioff,

H. & Leung, S.T.,2003) followed by a powerful programming paradigm of MapReduce (Dean, J., & Ghemawat, S.,2008).Thereafter a spectrum of new technologies emerged as the NoSQL movement stating a broad class of database management system to support increasing data storage and analytical requirements.

MAIN FOCUS

Major real world applications like business analytics operational on big data, cannot store or process all of the data on just one machine. The data must be stored, distributed or processed in parallel manner for computations to be completed efficiently. Various platforms are making *big data* management and processing more effective, forming the basis of current research theme in the era of *Big Data*. The main focus of this article is to discuss *NoSQL* big data storage platforms which could support processing of futuristic massive volumes of data in parallel.

NoSQL MOVEMENT

The NoSQL stands for "Not Only SQL" or "Not Relational". The growing needs to process continuously increasing volumes of data in lesser time, led to the movement of developing schema less data stores, parallel programming models and various analytical platforms. This movement is commonly covered under the term NoSQL. NoSQL platforms are non-relational, distributed and horizontally scalable. NoSQL platforms have been motivated by large scale Web based applications, where data has to be compulsively partitioned over multiple nodes that share the load, and parallelism is essential. This section aims at providing an overview of several characteristics and categories of NoSQL data stores. NoSQL has some essential features as described in Table 1.

There are various kinds of NoSQL databases namely key-value store, document oriented store, column oriented stores and graph data stores (Russom, P., 2011).A *key-value store* provides one of the simplest possible data extraction base in which a user is allowed to fetch data by using the key and the store doesn't know *anything* about values. This key based approach makes read and writes operations very fast. A *document-oriented store* extends the key-value model and values are stored in a *structured* format (known as document), that the database can understand. For example, a blog post or comments stored as a document in a de-normalized way. Content applications (like Facebook, Twitter etc.) can fetch an entire blog post data with just a single query with the help of *document-oriented data stores*. A *column-oriented*

Table 1. Features of NoSQL databases

Feature of NoSQL	Description
Schema-less	"Tables" don't have a pre-defined schema. Records can have variable number of fields and record contents as well as semantics are enforced by concerned applications.
Shared nothing architecture	Each server uses its own local storage instead of available common storage pool. Thus storage is accessed at local disk speeds instead of network speeds. Capacity is enhanced by adding more number of nodes.
Sharding	Instead of viewing the storage as a monolithic space, records are partitioned into small chunks known as shards which can be managed by a single server. An existing shard splits when it gets too much loaded with data. Applications can assist in data sharding by assigning each record a partition key ID.
Asynchronous replication	NoSQL databases employ asynchronous replication to scale up horizontally, to complete processing more quickly as they don't depend on extra network traffic. This corresponds traditionally to RAID storage or synchronous replication.
Follows CAP or BASE instead of ACID	NoSQL databases emphasize performance and availability. An application works basically all the time (basically available), does not have to be consistent all the time (soft-state) but will be in some known state eventually (eventual consistency).