Cloud Computing and Big Data Analytics

Nitin Choudhary[1] and Prateek Singh[2]

1.Student, Northern India Engineering College, Shastri Park
2 Student, Northern India Engineering College, Shastri Park

Abstract

A large volume of data is generated by many applications which cannot be managed by traditional relational database management system. As organizations use larger and larger data warehouses for ever increasing data processing need, the performance requirements continue to outpace the capabilities of the traditional approaches. The cloud based approach offers a means for meeting the performance and scalability points of the enterprise data management providing agility to the database management infrastructure. As with other cloud environments, data management in the cloud benefits end users by offering a pay-as-you-go (or utility based) model and adaptable resource requirements that free up enterprises from the need to purchase traditional hardware and to go through extensive procurement process frequently. The data management, integration and analytics can be offloaded to public and/or private clouds. By using public cloud, enterprises can get processing power and infrastructure as needed, whereas with public cloud enterprises can improve the utilization of the existing infrastructure. By using cloud computing, enterprises can effectively handle the wide ranging database requirements with minimum effort, thus allowing them to focus on the core work rather than getting bogged down with infrastructure. Despite all these benefits, decision to move from dedicated infrastructure to the cloud based Data processing depends on several logistics and operational factors such as security, privacy, availability etc

1. Introduction

Many industries such as telecom, retail, health care, etc. generate large amount of data. Querying and analysing such massive data for business is becoming the need of the hour. Traditionally, data warehouses have been used to manage the large amount of data. The warehouses and solutions can perform analytics on big volume once in days or one can perform transactions on small amounts of data being produced by industries is very large, for example, the Indian telecom generates more than 1 terabyte of call detail records(CDR's) daily. This large data is known as BIG DATA and it exceeds the processing capacity of conventional database systems. For such large data, warehouses are not practical and their infrastructure is costly and the analytics of data is slow.To overcome the shortcomings of traditional warehouses and systems, clouds are used. They are cheap and allow businesses to off-load computing tasks while saving IT costs and resources. The cloud providers have no intervention and provide self- service. The clients/businesses have to pay just for their usage. As the cloud resources are shared by many users/clients, cost of computing is very low comparatively.This report outlines the challenges in analysing big data for both, data at rest and data in motion. For big data at rest there are two systems:

1. No SQL for interactive data serving environment
2. System for large scale analytics based on Map Reduce Paradigms such as Hadoop.

For big data in motion, the report presents use-cases and illustrative algorithms of data stream management system (DSMS).

This report talks about the following topics:

1. **Cloud data management:** cloud computing can be used for performing massive scale data analytics in a cost effective and scalable manner. In this section we discuss the interplay of cloud computing and data management; and the factors one should consider while moving from a dedicated data management infrastructure to a cloud based infrastructure**.**
2. **No SQL:** It encompasses a number of techniques for processing massive data in distributed manner. Currently it is used for all the alternative data management technologies which are used for solving the problems for which relational databases are a bad fit. It enables efficient capture, storage, search, sharing, analytics, and visualization of the massive scale data.
3. **Hadoop MapReduce:** It is used for writing applications processing vast amount of data in parallel on large clusters of machines in a fault-tolerant manner. This framework involves writing two user defined generic functions: map and reduce.
4. **Data Stream Management System**: Data Stream Management Systems (DSMSs) allow user to analyze the data-in-motion. This analysis is done in real-time thereby allowing users to trigger important events to enable enterprises to perform actions just-in-time yielding better results for the businesses.

5. **Querying data over cloud:** Various high level query languages have been developed so that one can avoid writing low level MapReduce programs. Queries written in these languages are in turn translated into equivalent MapReduce jobs by the compiler and these jobs are then consequently executed on Hadoop. Three of these languages Hive, Jaql and Pig.
6. **Data Management Applications Over Cloud:** In this section, we consider three example applications where large scale data management over cloud is used. These are specific use-case examples in telecom, finance, and sensors domains.
   6.1 Dashboard for CDR Processing
   6.2 Credit Card Fraud Detection
   6.3 Spatio-temporal Data Processing

This report also talks about big data processing that involves interactive processing and decision support processing of data at rest and real time processing of data in motion.

2. CLOUD DATA MANAGEMENT

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation.Cloud computing performs large scale analytics in a cost effective manner. Big data cannot be managed by traditional warehouses or database systems, but it can be managed effectively by cloud. The users don't need to purchase additional hardware as they can pay the cloud providers according to their usage. The data management, integration and analytics can be offloaded to public or/and private clouds. With private cloud usage , businesses can improve the utilization of existing infrastructure with public cloud usage , businesses can get processing power and infrastructure as needed. By using cloud computing , businesses can concentrate on their core work and enhancement rather than face data analytics problems.

There are few factors to be taken care of:

1. **AVAILABILITY GUARANTEES** : Out of real time transactional data & analytics infrastructure , one may be willing to put its analytics infrastructure over the cloud . hence, each cloud computing provider can ensure a certain amount of availability guarantees.
2. **RELIABILTY**: One must ensure the cloud's reliability for providing services before loading it with data.
3. **SECURITY**: For organisations with confidential information , the cloud must be secure and password enabled.
4. **MAINTAINABILITY**: One must make sure that maintenance operations and data organisation facilities are provided by the cloud.

3.Techniques Used

3.1 No SQL

NoSQL consists of a no. of techniques used for processing massive data in distributed manner, which includes efficient capture, storage, search, sharing, analytics and visualization of the massive scale data. Relational databases are not designed for distributed horizontal scaling, therefore the main reason for using NoSQL is the scalability issues. Thus, we use two technologies for meeting scalability requirements:-

1. **Replication:** In replication, we use a master slave architecture wherein reads can be performed at any of the replicated slave, whereas writes are performed at the master.
2. **Sharding:** Sharding also known as partitioning requires the application to be partitioned first, defeating the very purpose of relational databases.

3.1.1 Characteristics of No SQL databases

A very important difference between the traditional databases and NoSQL databases is that the latter does not support updates and deletes. The reason behind this is that most of the applications do not need these operations rather they maintain different versions of the same data. Some of the examples are CDR's (call detail records), human resource databases etc. These operations are handled using insertion with version control. Relational databases provide ACID(atomicity, consistency, integrity and durability) properties.

- Since atomicity of over more than one record is not required in most of the cases, therefore single key atomicity is provided by No SQL databases.
- Consistency can be ensured using eventual consistency where reconciliation happens asynchronously to have eventually consistent database.
- Durability is ensured in both the traditional and No SQL databases but the traditional databases uses expansive hardware whereas No SQL databases provide that with cluster of disks with replication and other fault tolerance mechanism.

Alternatives of ACID

BASE (basic availability soft state and eventual consistent)If one nees to scale up for processing massive amount of data the he/she has to go for BASE.

No SQL data models

No SQL databases are flexible, they can support various types of data models. No SQL databases support key value pairs, hierarchical data, geo-spatial data, graph data, etc., using a simple model. We describe the three data models used in No SQL:

1. **Key value stores**: In a key-value read and write operations to a data item are uniquely identified by its key. Amazon dynamo is an example of key-value store in which values are opaque to the system. It provides incremental scalability; hence keys are partitioned dynamically using a hash function to distribute data over a set of machines or nodes. Each node knows about its peers which in turn allow any node to forward a key's read or write operation to the correct set of nodes. These operations are performed on a number of nodes to handle data durability and availability.

2. **Document stores :** In document stores, value associated with the key is a document which is not opaque to the database; hence, it can be queried. Amazon's simple DB[30], Mongo DB[32] and apache's couch DB[31] are some examples of No SQL databases using this model.
3. **Column family stores:** In this model, data is organized into tables. Each record is identified by a row-key. Each row has a number of columns which are organized into column-families. It supports transactions only under a single row key. Data is partitioned by sorting row-keys lexicographically. Big Table can serve data from disk as well as memory. Data is organized into table which are characterised by their start-key and end-key. Meta-data are maintained to locate tablet-server for a particular key.

3.2 Hadoop

Hadoop is a generic framework to write massive scale data application which involves writing two user defined generic functions:

Map

In the map step, a master node takes the input data
and the processing problem, divides it further into smaller data chunks and sub-problems; and then distributes them to worker nodes, which then processes one or more chunks using the sub-problem assigned to it. Specifically, each map process, takes a set of {*key*, value} pairs and generates one or more intermediate{key, value} pairs for each input key.

Reduce

In the reduce step, intermediate key-value pairs are processed to produce the output of the input problem. Each reduce instance takes a key and an array of values as input and produces output after processing the array of *values*:

Map(k1,v1) • list(k2,v2)
Reduce(k2, list(v2)) • list(v3)

Example of a MapReduce implementation for a scenario where one wants to find the list of customers having total transaction value more than $1000:

```
void map(String rowId, String row):
 // rowId: row name
 // row: a transaction recode
customerId= extract customer-id from row
transactionValue= extract transaction value from row
EmitIntermediate(customerId, transactionValue);
void reduce(String customerId, Iterator partialValues):
 // customerId: Id to identify a customer
 // partialValues: a list of transaction values
 int sum = 0;
 for each pv in partialValues:
 sum += pv;
 if(pv > 1000)
 Emit(cutsomerId, sum);
```

3.2.1 Hadoop

Hadoop is the most popular open source implementation of MapReduce framework Which is used for writing applications processing vast amount of data in parallel on large clusters of machines in a fault-tolerant manner. Machines can be added and removed from the clusters as and when required. In Hadoop, data is stored on Hadoop Distributed File System (HDFS) which is a massively distributed file system designed to run on cheap commodity hardware.

Steps:

1.Each file is broken into a number of blocks and these blocks are then parsed by user-defined code into {*key*, *value*} pairs to be read by *map* functions.

2. The *map* functions are executed on distributed machines to generate output {*key*, *value*} pairs which are written on their respective local disks.

3. Each *reduce* function uses HTTP GET method to pull {*key*, *value*} pairs corresponding to its allocated key space.

4. A reduce instance processes the key and array to get the desired output.

Architecture

HDFS follows master-slave architecture. An HDFS cluster has a single master called name node and a number of slave nodes. The name node manages the file system name space.It divides the file into blocks, and replicates them to different machines. Slaves, also called data nodes, manage the storage corresponding to that node. Fault tolerance is achieved by replicating data blocks over a number of nodes. The master node monitors progress of data processing slave nodes and in case it fails or it is slow, reassigns the corresponding data-block processing to another slave node. In Hadoop, applications can be written as a series of MapReduce tasks also.

4. DATA STREAM MANAGEMENT SYSTEM (DSMS)

DSMS is for analysis of 'data in motion'. In contrast to conventional databases, the analysis is done in real-time and actions are performed 'just in time'. These systems can perform stream operations and these systems can be thought of as a series of connected operators. Source operators are source tuples. Intermediate operators perform operations such as join etc. sink operators have the output.

Various stream processing systems

<u>VARIOUS STREAM PROCESSING SYSTEMS INCLUDE:</u>

IBM's infosphere, twitter's storm and yahoo's s4.

1. IBM's infosphere is a component based distributed stream processing platform. It supports higher data rates and various input types and provides load balancing and scheduling.
2. Twitter's storm provides a general framework for batch operations just like hadoop. Spouts are entities that handle the insertion of data types into the topology and bolts are entities that perform operations. Spouts and bolts are programmed using programming languages. Spouts and bolts are connected by stream to form a directed graph. Storm also provides fault tolerance.
3. In S4 terminology, the basic computation unit is processing element (PE) and processing nodes

(PN's) are logical hosts for PE's. Each stream is described as a sequence of events having pairs of keys and attributes. Each PE consumes exactly those events which correspond to the value on which it is keyed.

4.1 Quering Data Over The Cloud

As discussed earlier, processing data in hadoop requires programming MapReduce using programming languages like python, JAVA, etc. This has many problems as it is time consuming, highly skilled developers are required, proper scheduling required and that all reducers should have equal distribution of data to process.To overcome these problems 'high- level query languages' have been developed. For example: SQL.

Three high-level query languages are:

1. **HIVE**: developed by Facebook.All the features of Hive are very SQL-like so the effort to learn and use Hive is minimal. Like SQL, table schema has to be provided and data has to be filled in. Table can be partitioned on a set of attributes and these partitions make data fetching easier.Hive has the same operations as SQL such as join, group by etc.
2. **PIG**: it is a high level scripting language developed by Yahoo. It follows the declarative style of SQL and low level procedural style of MapReduce. There is step-wise transformation and the transformation carried out in each step is fairly high-level e.g., filtering, aggregation etc., similar to as in SQL. The program written in Pig is parsed and a directed acyclic graph is made with all the necessary optimizations to be performed at this stage. This plan is compiled and then optimized by the MapReduce optimizer and then it is sent to the to Hadoop job manager for execution. Unlike Hive, schemas here are optional. Pig supports complex and non-atomic data types such as map and tuple as fields of a table. Pig provides debugging environment.
3. **JAQL**: Jaql is a functional data query language, designed by IBM and is built upon JavaScript Object Notation (JSON) [8] data model. Jaql is a general purpose data-flow language that manipulates semi-structured information in the form of abstract JSON values. Jaql provides easy migration between different languages like javasrcipt and python. Supports atomic values such as numbers and strings. Jaql also provides a user with the capability of developing *modules*, a concept similar to Java *packages*. A set of related functions can be bunched together to form a module. A Jaql script can import a module and can use the functions provided by the module.

Querying streaming data :

There are two languages for this:
Continuous Query language: CQL is an SQL based declarative language for continuously querying streaming and dynamic data.CQL semantics is based on three classes of operator:

1. Stream-to-Relation, relation-to-relation, and relation-to-stream. In stream-to-relation operators, CQL has three classes of sliding window operators: time-based, tuple-based, and partitioned. In the first two window operators, window size is specified using a time-interval *T* and the number of tuples N, respectively. The partitioned window operator is similar to SQL group-by which groups N tuples using specified attributes as keys.
2. Relation-to-Relation operators are derived from traditional relational queries. CQL has three relation-to-stream operators: Istream, Dstream and Rstream. Applying an Istream/ Dstream (insert/delete stream) operator to a relation R results in a stream of tuples inserted/deleted into/from the relation *R*. The Rstream (relation steam) generates a stream element <*s*,τ> whenever tuple *s* is in relation *R* at time τ. It should be noted that there are no stream-to-stream operators in CQL.

The continuous query is compiled using query plan and added to other queries. Each query plan runs continuously with three types of components: *operators*, *queues*, and *synopses*. Each operator reads from input queues, processes the input based on its semantics, and writes output to output queues.
Synopses store the intermediate stage needed by continuous query plans.

Stream Processing Language:

structured application development language to build applications over Info Sphere streams. It supports structured as well as unstructured data stream processing. It provides a toolkit of operators using which one can implement any relational query with window extensions. Among operators:

- *functor*

is used for performing tuple level operations such as filtering, projection, attribute creation, etc.;

- *aggregate*

is used for grouping and summarization;

- *join*

is used for correlating two streams;

- *barrier*

is used for consuming tuples from multiple streams and outputting a tuple in a particular order;

- *punctor*

is also for tuple level manipulations where conditions on current and past tuples are evaluated for generating punctuations in the output stream;

- *split*

is used for routing tuples to multiple output streams; and

- *delay*

operator is used for delaying a stream based on a user-supplied time interval. Besides these System-S also has edge adaptors and user defined operators.

- *source*

adaptor is used for creating stream from an external source. This adaptor is capable of parsing, tuple creation, and interacting with diverse external devices.

- *sink*
- adaptor can be used to write tuples into a file or a network. It supports three types of windowing: tumbling window, sliding window, and punctuation-based window.

5. Conclusion

1. The increasing amount of data led to the different technologies such as NoSQL, Hadoop, Streaming data processing; Pig, Jaql, Hive, CQL, SPL for querying And distributed processing.
2. There are various advantages in moving to cloud resources from dedicated resources for data management.
3. But some of the enterprises and governments are still skeptical about moving to cloud. More work is required for cloud security, privacy and isolation areas to alleviate these fears.
4. For given cloud resources one needs to associate required resources for both the modules (bulk and stream data processing) so that the whole system can provide the required response time with sufficient accuracy. More research is required for facilitating such systems.

6. Refrences

[1] A. Abouzeid, K. B. Pawlikowski, D. J. Abadi, A. Rasin, and
A. Silberschatz. HadoopDB: An Architectural Hybrid of MapReduce
and DBMS Technologies for Analytical Workloads. PVLDB, 2(1):922–933, 2009.
[2] D. Agrawal, S. Das, and A. E. Abbadi. Big data and cloud computing: New wine or just new bottles? PVLDB, 3(2):1647–1648,
2010.
[3] D. Agrawal, A. El Abbadi, S. Antony, and S. Das. Data Management
Challenges in Cloud Computing Infrastructures. In DNIS, pages
1–10, 2010.
[4] P. Agrawal, A. Silberstein, B. F. Cooper, U. Srivastava, and
R. Ramakrishnan. Asynchronous view maintenance for vlsd
databases. In SIGMOD Conference, pages 179–192, 2009.
[5] S. Aulbach, D. Jacobs, A. Kemper, and M. Seibold. A comparison of
flexible schemas for software as a service. In SIGMOD, pages
881–888, 2009.
[6] P. Bernstein, C. Rein, and S. Das. Hyder – A Transactional Record
Manager for Shared Flash. In CIDR, 2011.
[7] M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska.
Building a database on S3. In SIGMOD, pages 251–264, 2008.
[8] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach,
M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A
Distributed Storage System for Structured Data. In OSDI, pages
205–218, 2006.
[9] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton.
Mad skills: New analysis practices for big data. PVLDB,
2(2):1481–1492, 2009.
[10] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein,
P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni.
PNUTS: Yahoo!'s hosted data serving platform. Proc. VLDB Endow.,
1(2):1277–1288, 2008.
[11] C. Curino, E. Jones, Y. Zhang, E. Wu, and S. Madden. Relational
Cloud: The Case for a Database Service. Technical Report 2010-14,
CSAIL, MIT, 2010. http://hdl.handle.net/1721.1/52606.
[12] S. Das, S. Agarwal, D. Agrawal, and A. El Abbadi. ElasTraS: An
Elastic, Scalable, and Self Managing Transactional Database for the
Cloud. Technical Report 2010-04, CS, UCSB, 2010.
[13] S. Das, D. Agrawal, and A. El Abbadi. ElasTraS: An Elastic
Transactional Data Store in the Cloud. In USENIX HotCloud, 2009.
[14] S. Das, D. Agrawal, and A. El Abbadi. G-Store: A Scalable Data
Store for Transactional Multi key Access in the Cloud. In ACM
SOCC, 2010.
[15] S. Das, S. Nishimura, D. Agrawal, and A. El Abbadi. Live Database
Migration for Elasticity in a Multitenant Database for Cloud
Platforms. Technical Report 2010-09, CS, UCSB, 2010.
[16] S. Das, Y. Sismanis, K. Beyer, R. Gemulla, P. Haas, and
J. McPherson. Ricardo: Integrating R and Hadoop. In SIGMOD,