

State of Big Data Analysis in the Cloud

Sanjay P. Ahuja¹ & Bryan Moore¹

¹ School of Computing, University of North Florida, Jacksonville, USA

Correspondence: Sanjay P. Ahuja, School of Computing, University of North Florida, Jacksonville, FL 32224, USA. E-mail: sahuja@unf.edu

Received: April 15, 2013 Accepted: May 14, 2013 Online Published: May 22, 2013

doi:10.5539/nct.v2n1p62

URL: <http://dx.doi.org/10.5539/nct.v2n1p62>

Abstract

Big Data is data that either is too large, grows too fast, or does not fit into traditional architectures. Within such data can be valuable information that can be discovered through data analysis. With the emergence of cloud computing services, big data processing has become a less costly task. In this paper, we examine the current trends and characteristics of Big Data, its analysis and how these are presenting challenges in data collection, storage and management in cloud computing.

Keywords: cloud, big data, big data analysis

1. Introduction

In recent years, there has been a continually growing amount of data being produced. Some of this data is generated online through sites such as social media giants Facebook, Twitter and YouTube. Other data is being generated everywhere from online purchase transactions, to scientific DNA analysis. Big data may have big potential, but it can also present interesting challenges. Big data means large datasets. How to store these datasets is different from the traditional relational database. This is not only because of the sheer volume, but also because of both types of structured and unstructured data. For rapidly growing datasets, there can be a need to analyze data as it arrives to get the maximum value due to its time sensitivity. Lastly, the validity of the data needs to be considered whether the derived information from the analysis can be trusted (Singh 2012).

Another aspect of big data to consider is the cost to both store and process these massive amounts of data. Cloud computing can be a possible solution as it provides a solution that is cost efficient while meeting the need of rapid scalability - an important feature when dealing with big data. But even in cloud computing, big data analysis is not without its problems. Careful consideration must be given to the cloud architecture and the techniques for distributing these data intensive tasks across the cloud (Ji, 2012).

In this paper, we examine the trends and issues in big data analysis in cloud computing. The paper is organized as followed: Section 2 reviews related work, Section 3 briefly reviews big data analysis, Section 4 describes the current challenges of big data, Section 5 examines how big data can benefit from the cloud, Section 6 reviews current solutions of big data analysis in the cloud, Section 7 describes the open problems of big data analysis, and Section 8 concludes the paper.

2. Related Work

When considering big data, techniques and approaches to store the data need to be reevaluated. For storing large volumes of data, distributed file systems are a possible solution. In the context of data analysis, a distributed model is advantageous both due to the ability to push computation to various nodes in a cluster and the scalability it provides. One example of this type of system would be the distributed file system coupled with the MapReduce engine in Apache's Hadoop project (Ji, 2012). Other challenges in data storage are due to the variety of structured and unstructured data. One approach to this problem is addressed by NoSQL databases. NoSQL databases are characteristically non-relational and typically do not provide SQL for data manipulation. NoSQL describes a class of databases that include: graph, document and key-value stores. For key-value stores, everything is stored as a key and its value. Data manipulation typically includes just three functions: get, put and remove. Graph databases are typically suited for representation of network topologies or social networks. In document-oriented databases, data is stored in documents which typically take a format such as JSON or XML. These databases do not have a rigid schema, such that each document can store varied information. In comparison to the traditional relational DBMS, NoSQL databases eschew the management to the application as

it aims to provide a highly scalable database. Furthermore, these databases are schema less, allowing changes to the data structure rapidly rather than having to do table rewrites. This gives an advantage over relational DBMS's in regards to horizontal scalability through the cloud. As a response to the scalability that NoSQL introduced, a new class of NewSQL databases has developed that follow the relational model but either distributes the data or transaction processing across nodes in a cluster to achieve comparable scalability (Pokorny, 2011).

There are several factors that need to be mitigated in order for data management in the cloud to be an adequate solution. These factors are spread across the different types of existing systems. For Key-Value stores these would include enhancing the available feature set with ad-hoc querying as well as providing some consistency guarantees. With regards to relational systems, there exists the challenge of making the systems scalable such that the cloud resources can be used efficiently with attention given to the load balancing across the cloud. Finally, a major challenge has to be considered is the engineering of systems that provide both security and privacy for the data that resides in the cloud (Agrawal, 2011).

With ever increasing datasets, there comes the need to move data at greater speeds. Current protocols such as TCP have shown to have performance issues at speeds over 40 Gbps. These issues relate primarily to the increasing consumption of CPU resources. Tierney et al suggested a Remote Direct Memory Access (RDMA) protocol over Converged Ethernet (RoCE). Their results suggested that RoCE would provide scalability beyond 40 Gbps with little CPU overhead (Tierney, 2012). Other methods have been investigated to manage data as it arrives into the data processing system such as using a Data Stream Management Systems (DSMS) that processes data as it arrives into the system. The DSMS aims to reduce the amount of bad data from being collected, thus reducing future costs for storage and processing as well as finding potential valuable information and patterns sooner in the data analysis pipeline (Ari, 2012).

Systems need to be kept modular such that analytic tools and approaches can be applied when needed. Other factors have been outlined as to what should be considered when constructing a system for the consumption of big data. These would include the format and size of the data being ingested, the types of analytics being conducted, and the objectives of the system. The data size and format is a major factor for deciding the architectural components for storing data in the system. Doing so can avoid making decisions on a non-relational model for its scalability when the size of data does not warrant a need (Begoli, 2012).

3. Big Data Analysis

Data analysis can also be described as knowledge discovery from data. Knowledge discovery is a method where new knowledge is derived from a data set. More accurately, knowledge discovery is a process where different practices of managing and analyzing data are used to extract this new knowledge (Begoli, 2012).

3.1 Data Collection

The first step in the data processing pipeline is data collection. In this step all data that is to be processed is consolidated for analysis. Difficulties with data collection lie in the different forms that data may have as they arrive from different sources. Data integration is later performed to keep data as cohesive as possible.

3.2 Data Cleansing

After collection, data cleansing or cleaning is performed. There may be data that is either noisy, erroneous or missing values. Data cleaning uses different methods to eliminate this bad data from the dataset. After cleaning, data may need to be transformed as final preparation for analytics (Agrawal, 2012).

3.3 Data Analysis

After data processing the analysis can begin. In this stage, many different analytic methods and techniques may be performed. These methods and techniques can be broken down into three categories: statistical analysis, data mining and machine learning. Statistical analysis creates models for predication and summarizes datasets. Data mining uses a variety of techniques (clustering, classification, etc.) to discover patterns and models present in the data. Machine learning is used to discover relationships that are present within the data.

4. Challenges of Big Data

4.1 Collection

One of the major problems with big data is its sheer size. The world's data is growing at an exponential rate. Cloud computing provides a solution that meets some scalability needs. The major problem with this system would be getting the data into the cloud to begin processing. Using standard Internet connections to upload the

data to the cloud would be a significant bottleneck in the process. New techniques need to be investigated and developed to increase the efficiency of data movement into the cloud as well as across clouds (Bryant, 2008).

4.2 Storage

A significant problem with handling big data is the type of storage. Using a cloud approach, the traditional database is not currently suited to take advantage of the cloud's horizontal scalability. Current systems that exist handle scalability but do so at the expense of many of the advantages the relational model provides. New systems need to carefully take into account the need for these features while also providing a scalable model.

4.3 Analysis

The major reason behind the need for handling big data is to be able to gain value from data analysis. Analytic techniques and methods need to be further researched to develop techniques that can be able to process large and growing data sets. Simplification of the analysis process of big data towards an automated approach is a major goal behind big data (Bryant, 2008).

4.4 Security

With the advent of knowledge discovery on big data there can be new information derived. There is much focus on two main problems when securing these large data systems. The first is to secure these systems such that there is a limited amount of overhead introduced so that performance will be greatly unaffected. More research and development needs to be conducted on securing data in the new types of big data systems and throughout the data analysis pipeline. For instance, a potential attack on the MapReduce paradigm could be a malicious mapper that accesses sensitive data and modifies the result. Unlike most RDBMS, NoSQL security is largely relied on outside of the database system. Research into the types of attacks that are possible on these new systems would be beneficial (Rajan, 2012).

5. Big Data in the Cloud

5.1 Data Transfer

To take advantage of the cloud for big data analysis, data must first be in the cloud. To address the issue of uploading big data to the cloud, a number of approaches to WAN optimization have been established. The techniques include: compression, data deduplication, caching, and protocol optimization.

Compression can provide some benefits for WAN optimizations. These benefits would be dependent on the type of data that is being compressed. For example, plain text data would be more compressible than encrypted data. In this case, using compression would be most beneficial when the data that is being sent is known to be both homogenous and susceptible to compression (Zhang, 2012).

Another technique that aims to reduce the size of data being transported is data deduplication. Data deduplication looks at data both at the file and block level. When there are duplicates that exist, they are replaced with a pointer to the other copy. This technique is also called redundancy elimination (Choi, 2009).

Other techniques are targeted towards protocol optimizations. The digital transfer company Aspera uses such a method in their *fasp* transport technology. In this protocol, one TCP and UDP port are utilized for session control and data transfer (Bloomberg, 2013).

5.2 Data Storage & Management

Big data has changed the architecture of systems for data storage. The two major factors for this shift are in the need to be highly scalable and flexible enough to effectively handle big data. For storage, distributed systems like the Google File System were designed to use commodity clusters for storage that is both reliable and efficient. In this system, data is stored as file blocks of 64MB across the nodes of the cluster. Two additional replicas are stored to provide redundancy. On top of GFS, MapReduce is used for processing data across the nodes. It is more efficient to push computations to where the data resides rather than the opposite. MapReduce takes advantage of the distributed architecture of the file system by sending jobs to the nodes on the cluster where the data resides.

There has been a significant amount of research on the MapReduce paradigm developed by Google for processing large data sets. This can be largely attributed to two reasons. The simplicity of the functions for processing and the challenges it handles (replication, storage, etc.). The most popular implementation, Hadoop consists of two components, the MapReduce engine and Hadoop Distributed File System. However the fact that the MapReduce paradigm is essentially both index and schemaless has been a point of contention against the

framework. This has led to the development of several systems built on top of the Hadoop core components to address these problems.

5.3 Hadoop

Hadoop is a framework that supports reliable and scalable distributed computing. Based on Google's white papers for GFS and MapReduce, there are three major components: the Hadoop Distributed File System (HDFS), the MapReduce engine, and the utilities for the other Hadoop Modules. There are several Hadoop modules of varying system types such as database (HBase and Cassandra), querying (Hive and Pig), and coordination services (ZooKeeper) (Ganesan, 2013).

5.3.1 Hive

Apache Hive is a data warehousing system used with Hadoop for querying, summarization, and analysis of data stored in Hadoop. Queries are expressed in the "SQL-like" Hive Querying Language. A compiler translates the HQL into a set of MapReduce jobs that are executed on the Hadoop system. As such, Hive provides a means to perform data manipulations with high level HiveQL, without having to write the more complex map reduce functions that are harder to maintain and reuse.

Hive organizes data into tables, partitions and buckets. A table logically consists of rows and columns. Physically, a table is HDFS directory that containing all of its data. Within a table can be multiple partitions that are organized by column value. Further, buckets within partitions are divided by the hash of the column.

The Hive Driver manages the lifecycle of a HiveQL statement from compilation to execution on Hadoop. The compilation process outputs a directed acyclic graph of map reduce jobs. This step includes optimizations to reduce the number of outputted jobs. Furthermore, users can influence the optimization by providing hints through HiveQL (Ashish, 2009).

5.3.2 Pig

Apache Pig is described as a platform for the analysis of large datasets. Like Hive, it uses a high level language that is compiled into MapReduce programs that are executed on Hadoop. Pig's high-level language is called Pig Latin. In addition, Pig allows extensions to Pig Latin with User Defined Functions that can be written in Java, JavaScript, or Python. Pig is primarily used to cut down on the complexity of writing map reduce functions, allowing for the ease of ad hoc querying and analysis (Olston, 2008).

5.3.3 Hbase

Hbase is a distributed, column-oriented NoSQL database that operates on top of the Hadoop Distributed File System. Modeled Google's BigTable, Hbase provides a distributed data store that is highly scalable with consistent reads and writes. Data is stored as indexed Storefiles on HDFS. Since it is built on top of HDFS, it is also fault tolerant (Apache Hbase, 2013).

5.3.4 ZooKeeper

ZooKeeper provides coordination services such that synchronization can be enabled throughout a Hadoop cluster. ZooKeeper achieves this by maintaining objects containing information such as information and namespaces in-memory. This information is kept across distributed ZooKeeper servers that would be retrieved the client applications that require them. The advantage of ZooKeeper would be the ability for synchronization across various Hadoop-based systems rather than having to implement those system-specific functionalities (Reed, 2012).

5.4 Cassandra

Initially developed by Facebook, Cassandra is a distributed column-oriented database. In Cassandra, the smallest piece of data is the column: consisting of a name, value and timestamp. A row contains of multiple columns, column families contain rows, and keyspaces contain column families. Column families are stored in separate files. Data is separated into partitions across the nodes in the distributed database. Each node has a random position on a hash ring. Based upon the hash value of the data item's key, it would be placed on the node that is positioned adjacent to that hash value. As data may not be distributed uniformly, a node's position is adjusted to alleviate nodes that are under pressure (Lakshman, 2009).

5.5 Voldemort

Developed by LinkedIn, Voldemort is a highly scalable, distributed key-value data store. Data is automatically replicated and partitioned among the nodes in the distributed system. Each node is independent of the others such that there is no single point of failure. Read and write access is limited to key-value access. As such, there

are only three types of queries available: get, put and delete. This simplicity provides predictability in the performance of queries (Sumbaly, 2012).

6. Current Solutions

6.1 Facebook

Facebook hosts the largest Hadoop cluster by volume, consisting of a total of over 4400 nodes and 100+ petabytes of data. In their system there are five major components: the Hadoop Core, a log data collector called Scribe, Hive, a UI for querying with Hive called HiPal, and an automation framework called NoCron. For their needs they have made some adjustments to their Hadoop configuration. Their HDFS uses a Federated HDFS configuration. HDFS redundancy is accomplished through RAID rather than level three replication to lower the footprint of their system. Facebook also uses Hive to simplify the interaction with Hadoop by their analysts. Roughly 90% of their MapReduce jobs are built on Hive (Menon, 2012).

6.2 Twitter

Twitter has significant data storage and processing needs, which they meet with the use of Hadoop. All of their data is stored on the Hadoop Distributed File System using LZO (Lempel-Ziv-Oberhumer) compression. Additionally, Google's Protocol Buffers are used to efficiently read and write data into their cluster through data serialization with the generated code it provides (Weil, 2010). Specifically their suggestion and recommendation features are powered by Hadoop. They also employ their Scalding solution as a means for developing their Hadoop based applications using scala. Scalding is a framework aimed to provide a simpler way of creating MapReduce jobs much like Hive and Pig (Ryaboy, 2012).

6.3 LinkedIn

At LinkedIn, Hadoop is used to support features such as People You May Know and Endorsements using predictive analytics and querying. Billions of LinkedIn relationships are processed each day to compute People You May Know. For creating their engagement emails, detailing a user's profile views versus those in their industry, LinkedIn adopted Apache Pig to avoid writing complex MapReduce programs (Luu, 2011). To better understand their Hadoop cluster usage across these different features, LinkedIn developed a Hadoop log aggregator and dashboard called White Elephant that visualizes the utilization across the users in the cluster (Hayes, 2013).

7. Open Problems

There still exist several open issues and problems when dealing with big data. The amount of data that is being produced is continually growing in quantity. This presents problems as transporting data across current networks may become the bottleneck in the process that takes more time than the actual processing. Solutions to this problem could include either processing data where it is stored rather than retrieving, or methods of identifying quality data such that only a subset is used. The second solution presents another set of problems in relation to how quality data can be gleaned from a quantity of data. Being able to evaluate every data item on its validity is not possible given the volume; new methods need to be investigated to validate data items. This includes being able to make decisions on how much data is enough to draw accurate conclusions, how to evaluate the quality of data, and how to decide whether data is accurate and reliable (Kaisler, 2013).

8. Conclusions

Hadoop has become a common solution for processing large amounts of data. However, Hadoop uses a batch-processing approach and does not provide adequate solutions for real-time ad hoc querying needs. Solutions such as Pig and Hive provide a means of simplifying querying, yet underneath they are still using MapReduce jobs to query Hadoop. Future development is expected to focus on systems that provide real-time, ad hoc querying capabilities over large scale data. Other interest is expected to include the development of querying systems that make use of SQL, in order to leverage existing SQL knowledge amongst users to query against Hadoop systems. For managing big data, there are several solutions that make use of NoSQL, essentially eschewing some of the advantages of relational databases for the scalability of a simpler system. There is expected to be attention given to the development of distributed relational systems that introduce the scalability that big data requires while providing the advantages of a relational model. Big Data is becoming a new way for exploring and discovering interesting, valuable information. The volume of data that exists is constantly magnifying such that the majority of data that exists has been created in just the past few years. With that in mind, the "big data" of tomorrow can be expected to be magnitudes larger than it is by today's standards. As a result, the problems of big data will only become more prominent in the future as solutions are being developed to meet the emerging needs.

Acknowledgement

This research has been supported by the Fidelity National Financial Distinguished Professorship in Computer and Information Sciences.

References

- Agrawal, D., Bernstein, P., Bertino, E., Davidson, S., Dayal, U., Franklin, M., ... Widom, J. (2012). *Challenges and Opportunities with Big Data*. Retrieved from <http://cra.org/ccc/docs/init/bigdatawhitepaper.pdf>
- Agrawal, D., Das, S., & El Abbadi, A. (2011). Big data and cloud computing: current state and future opportunities. In *Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT '11)* (pp. 530-533). A. Anastasia, A. Y. Sihem, P. Jignesh, R. Tore, S. Pierre, & S. Julia (Eds.). New York, NY, USA: ACM. Retrieved from <http://doi.acm.org/10.1145/1951365.1951432>
- Apache Hbase. (2013). Retrieved from <http://hbase.apache.org>
- Ari, I., Olmezogullari, E., & Celebi, O. F. (2012). Data stream analytics and mining in the cloud. *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on* (pp. 857-862). Retrieved from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6427563&isnumber=6427477>
- Ashish, T., Joydeep, S. S., Namit, J., Zheng, S., Prasad, C., Suresh, A., ... Raghobham, M. (2009). Hive: a warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, 2(2), 1626-1629. Retrieved from <http://www.vldb.org/pvldb/2/vldb09-938.pdf>
- Begoli, E., & Horey, J. (2012). Design Principles for Effective Knowledge Discovery from Big Data. *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on* (pp. 215-218). <http://dx.doi.org/10.1109/WICSA-ECSA.212.32>
- Bloomberg, J. (2013). *Big Data Processing in Cloud Computing Environments*. Retrieved from <http://www.zapthink.com/2013/01/29/the-big-data-bottleneck-uploading-to-the-cloud/>
- Bryant, R. E., Katz, R. H., & Lazowska, E. D. (2008). Big-data computing: Creating revolutionary breakthroughs in commerce, science, and society. In *Computing Research Initiatives for the 21st Century. Computing Research Association, 2008*. Retrieved from http://www.cra.org/ccc/docs/init/Big_Data.pdf
- Chou, W. (2009). Optimizing the WAN between Branch Offices and the Data Center. *IT Professional*, 11(4), 24-27. <http://dx.doi.org/10.1109/MITP.2009.78>
- Ganesan, H., & Olety, V. (2012). *Introduction to Big Data Hadoop Ecosystem*. Retrieved from <http://cloudstory.in/2012/04/introduction-to-big-data-hadoop-ecosystem-part-1/>
- Hayes, M. (2013). *White: Elephant: The Hadoop Tool You Never Know You Needed*. Retrieved from <http://engineering.linkedin.com/hadoop/white-elephant-hadoop-tool-you-never-knew-you-needed>
- Ji, C., Li, Y., Qiu, W., Awada, U., & Li, K. (2012). Big Data Processing in Cloud Computing Environments. *Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on* (17-23). <http://dx.doi.org/10.1109/I-SPAN.2012.9>
- Kaisler, S., Armour, F., Espinosa, J. A., & Money, W. (2013). Big Data: Issues and Challenges Moving Forward. *System Sciences (HICSS), 2013 46th Hawaii International Conference on* (pp. 995-1004). <http://dx.doi.org/10.1109/HICSS.2013.645>
- Lakshman, A., & Malik, P. (2010). Cassandra: a decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 44(2), 35-40. <http://dx.doi.org/10.1145/1773912.1773922>
- Luu, H. (2011). *User Engagement Powered by Apache Pig and Hadoop*. Retrieved from <http://engineering.linkedin.com/hadoop/user-engagement-powered-apache-pig-and-hadoop>
- Menon, A. (2012). Big data @ facebook. In *Proceedings of the 2012 workshop on Management of big data systems (MBDS '12)* (pp. 31-32). New York, NY, USA: ACM. <http://dx.doi.org/10.1145/2378356.2378364>
- Olston, C., Reed, B., Srivastava, U., Kumar, R., & Tomkins, A. (2008). Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08)* (pp. 1099-1110). New York, NY, USA: ACM. <http://dx.doi.org/10.1145/1376616.1376726>
- Pokorny, J. (2011). NoSQL databases: a step to database scalability in web environment. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services (iiWAS*

- '11) (pp. 278-283). New York, NY, USA: ACM. Retrieved from <http://doi.acm.org/10.1145/2095536.2095583>
- Rajan, S. et al. (2012). *Top Ten Big Data Security and Privacy Challenges*. Retrieved from https://downloads.cloudsecurityalliance.org/initiatives/bdwdg/Big_Data_Top_Ten_v1.pdf
- Reed, B. (2012). *ZooKeeper Overview*. Retrieved from <https://cwiki.apache.org/confluence/display/ZOOKEEPER/ProjectDescription>
- Ryaboy, D. (2012). *Twitter at the Hadoop Summit*. Retrieved from <http://engineering.twitter.com/2012/06/twitter-at-hadoop-summit.html>
- Singh, S., & Singh, N. (2012). Big Data analytics. *Communication, Information & Computing Technology (ICCICT), 2012 International Conference on* (pp. 1-4). <http://dx.doi.org/10.1109/ICCICT.2012.6398180>
- Sumbaly, R., Kreps, J., Gao, L., Feinberg, A., Soman, C., & Shah, S. (2012). Serving large-scale batch computed data with project Voldemort. In *Proceedings of the 10th USENIX conference on File and Storage Technologies (FAST'12)* (p. 18). Berkeley, CA, USA: USENIX Association.
- Tierney, B., Kissel, E., Swany, M., & Pouyoul, E. (2012). Efficient data transfer protocols for big data. *E-Science (e-Science), 2012 IEEE 8th International Conference on* (pp. 1-9). <http://dx.doi.org/10.1109/eScience.2012.6404462>
- Weil, K. (2010). *Hadoop at Twitter*. Retrieved from <http://engineering.twitter.com/2010/04/hadoop-at-twitter.html>
- Zhang, Y., Ansari, N., Wu, M., & Yu, H. (2012). On Wide Area Network Optimization. *Communications Surveys & Tutorials, IEEE, 14*(4), 1090-1113. <http://dx.doi.org/10.1109/SURV.2011.092311.00071>