International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 1 | Issue – 6

Big Data Analytics Architecture and Challenges, Issues of Big Data Analytics

S. Senthil Kumar
Assistant Professor
Dr.SNS Rajalakshmi College of Arts And Science (Autonomous), Coimbatore, Tamil Nadu, India

Ms.V.Kirthika
PG Student
Dr.SNS Rajalakshmi College of Arts And Science (Autonomous), Coimbatore, Tamil Nadu, India

ABSTRACT

Big Data technologies uses a new generation of technologies and architectures, designed for organizations can extract value from very large volumes of a wide variety of data by enabling high-velocity capture, discovery, and/or analysis. Big data is a massive amount of digital data being collected from various sources that are too large. Big data deals with challenges like complexity, security, risks to privacy. Big data is redefining as data management from extraction, transformation, cleaning and reducing. The size and variety of data lead us to think ahead and develop new and faster methods of mining data which uses the parallel computing capability of processors. The above is known as big data.

Keywords: *Big data*

I. INTRODUCTION

Big Data is continuously growing during the recent years, and each data scientist will have to manage much more amount of data every year. Big data is going to be greater extent diverse, larger and faster. Big Data is becoming the latest trend for precise data research and for business applications.

Big data is the latest term for a collection of data sets which are large and complex, it contain structured and unstructured both type of data. Data may come from anywhere, sensors used to gather climate information, posts in social media sites, digital pictures and videos etc.

Big data varies depending upon the capabilities of the organization managing the set, and on the capabilities of the applications that are used to processing and analyzing the data.

ARCHITECTURES FOR BIG DATA ANALYTICS

Big data analytics have been proposed with several architectures. Many of them share common computational models. Depending on our study, we are used to classify big data solutions into three major architectures. Each of these architecture given below has their own advantages as well as limitations, and their suitability depends on the nature and requirements of the algorithm to be implemented. These are discussed below.

(i) MapReduce Architecture:

MapReduce is a data-parallel architecture, originally developed by Google [1]. Parallelism is successfully bring out by multiple machines or nodes performing the same task on different data. Apache Hadoop is a highly used open-source implementation of MapReduce. A MapReduce daemon runs on the nodes all the time continuously. There is one master node that performs the configuration and controls the

execution of the problem. The other alternate nodes are called the worker nodes and perform actual computation on data. The master node splits the data, assigns them to worker nodes, and puts them into the global memory as (key, value) pairs. Figure (a) depicts the basic architecture of MapReduce, where W_i's are the worker nodes.

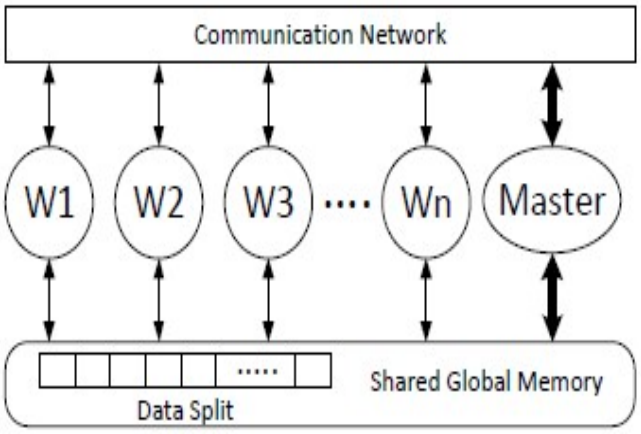

Figure (a) MapReduce architecture

MapReduce works in rounds, each consist of two phases, namely map and reduce phases. A single node can be used in both map and reduce phases. Each phase consists of three states namely: input, computation and output. There exists one synchronization barrier between any two consecutive phases. During synchronization, local memory of the node is cleared and written onto the global memory. The master node can read/write onto the global memory and communicate with the other nodes during all the time. However, the worker nodes can read/write onto the global memory exceptionally during synchronization. In Figure (a), this has been distinguished using thick and thin arrows.

In the map phase, the problem is data-distributed among the worker nodes and the partial results generated by the worker nodes are stored only in the global memory. During the reduce phase, the partial results may be combined to obtain the overall result, to be stored in the global memory. If the intermediate results are to be further processed, the phases will be repeated again.

MapReduce architecture performs satisfactorily when the data size is huge and the problem in hand is embarrassingly parallel. The architecture of MapReduce provides fault-tolerance by re-doing the computation (done by the failing node) for the phase on another node. The MapReduce architecture has limitations for problems involving high computational dependencies among data. The MapReduce architecture cannot be used to express iterative computations and becomes inefficient with high I/O overhead.

Some efforts have been made to mitigate the limitations of the MapReduce architecture and improve its performance. Twister [2] optimizes iterative computations on the MapReduce architecture by using in-memory computations, rather than writing onto the distributed memory after each phase. However, Twister has fault-tolerant issues due to in memory processing. Apache Spark19 extends Hadoop by using Resilient Distributed Database (RDD) [3] to allow in-memory processing as well as fault-tolerance by reconstructing a faulty partition in case of node failure.

(ii) FAULT TOLERANT GRAPH ARCHITECTURE:

MapReduce architecture and its different implementations process data in batch mode, they are not very expressive when complex computational dependencies exist among data. Mostly the machine learning and statistical methods inhibit high data dependencies. Hence, MapReduce is not the best architecture for them. Different and distinct architectures are needed to process the complex and iterative problems efficiently, while supporting fault tolerance.

Fault tolerance is most important for scalability also, since it allows using unreliable networks, such as the Internet. In order to achieve, a fault tolerant graph-based architecture, called GraphLab, was first proposed by Low et al. [4] and many other big data solutions adopted similar architectures. In fault tolerant graph architecture, the computation is divided among nodes in a heterogeneous way, with each of them performing some particular tasks. The data model is divided into two parts namely, a graph with computing nodes and ii) a shared global memory (distributed). The generic fault tolerant graph architecture is depicted in Figure(b). The N_i's represent the computing nodes and the dotted arrows show the dependencies among the nodes, whereas actual communication is performed via the communication network.

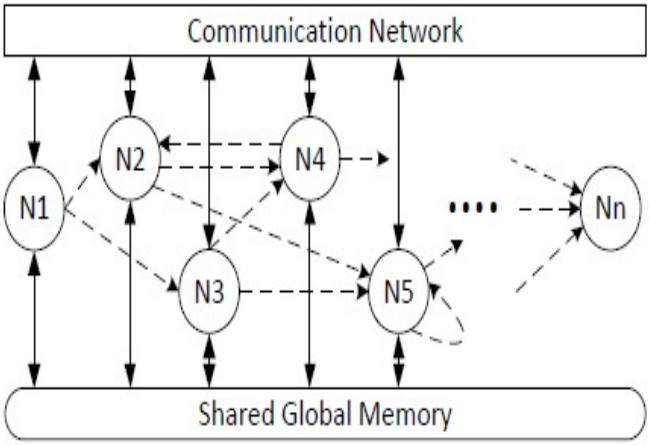

Figure (b) Architecture for Graph with global shared memory

Similar to MapReduce architecture, the computation is carried out in execution cycles in a synchronous manner. The shared database is initialized with the above input data. The beginning of each cycle, a node first reads the shared database and then performs computation using its own and its neighbor's data. Therefore the results are merged and then written back to the global shared database, for use in the next execution cycle. If a node fails in one cycle, it is recomputed and the dependent nodes lose one cycle. Even though it reduces the efficiency by a cycle, the fault tolerance is guaranteed. If a node fails permanently, then it is replaced by another. The above fault tolerant architecture provides high expressiveness for complex problems with data dependency and iterations. However, the fault tolerant architecture demands high disk I/O and therefore, it is not optimized for performance. To the best, we use an improvement using RDD to facilitate in memory processing and fault tolerance is not yet proposed. GraphLab is another other major graph-based big data solutions are Pregel and Giraph. Graph packages is used to develop the MapReduce architecture, such as GraphX and the Hama graph package called Angrapa.

(iii) STREAMING GRAPH ARCHITECTURE:

The graph-based architecture allows scalable distributed computation, complex data dependency among operations, efficient iterative processing, and fault tolerance. Even though, due to its high disk read/write overhead, it is not efficient for stream data. There are packages to perform analytics on stream data on the MapReduce architecture, such as Spark Streaming, they internally convert stream data to batches for processing.

Streaming applications require in-memory processing for high bandwidth. The well known Message Passing Interface (MPI) is a good fit for the above problem. In the application level, MPI has similar API as MapReduce and almost all MapReduce programs can also be implemented using MPI. Figure (c) depicts the graph-based architecture for large scale distributed processing, for high bandwidth and iterative applications with high data dependency among operations. Hence, the architecture is in line with the ever increasing computing speed and improved network bandwidth and reliability.

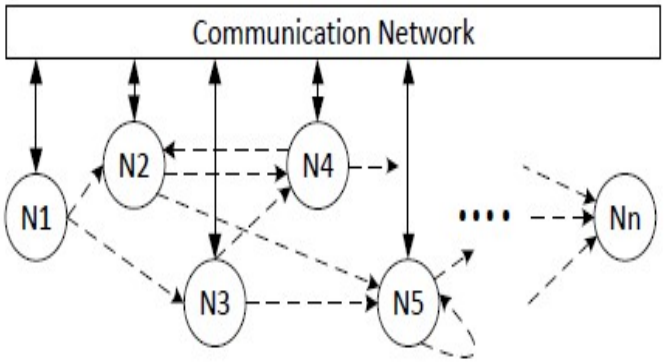

Figure (c) Architecture for graph-based asynchronous processing

There are three major differences between this architecture and the previous one. In the above architecture, a global shared memory is not used, rather the nodes exchange data using peer-to-peer communications directly. Second, the operations are performed in an asynchronous manner. Some different data flows become synchronous only during their merge operations. Finally, in the above architecture, data need not be stored into disks. As memories are becoming cheaper every day, in-memory processing of large volume data is possible, which significantly increases the overall throughput.

The main disadvantage of this architecture is the absence of fault tolerance. If any one of the nodes fails, the process has to start from the beginning all over again. Consequently, the above architecture is unsuitable in unreliable networks, such as the Internet. This in turn causes scalability issues. Even though, if a reliable network is available and the algorithm has high data dependency, then this architecture can provide higher throughput than the other architectures. The above architecture can be

implemented on standalone clusters using MPI to perform analytics on big data.

CHALLENGES IN BIG DATA ANALYTICS:

The techniques used for analysis and visualization of traditional databases are inadequate on big data. The volume, velocity, variety, distributed, and incremental nature of such data impose challenges on the traditional methods for data analytics. The volume of data generated and the speed of data transmission is growing rapidly. The increase in data volume, the speed of data generations and transmissions are also increasing. Real time analytics on big data is more difficult with high data velocity. Although batch mode analytics may be scalable to high data velocity using distributed and parallel computing techniques, the slow I/O operations severely affect the analytics performance. In recent years, I/O speed is lacking far behind computing speed, acting as the limiting factor of computational throughput.

Moreover, continuously generated data are highly heterogeneous in nature. Traditional databases are arranged in a set of defined schemas. Data warehouses store and update data following the extraction-transformation loading operations. Since big data systems continuously fetch new data in high velocity and high variety from heterogeneous sources, a structured database, such as data warehouse, is not suitable for dynamic storage and real time retrieval.

Given the above challenges, the traditional data analytics techniques, such as machine learning and statistical analysis, are inefficient with big data in their original form. As a result, the problem of machine learning enabled analytics has to be studied from the perspective of big data.

Data privacy is another challenge of big data analytics, particularly in the bioinformatics and healthcare domain. Hence to protect sensitive information, data sources might use data anonymity or publish only partial data. Analytics on partial and anonymous data might be more complex and inefficient.

ISSUES IN BIG DATA ANALYTICS:

Big data analytics require processing of huge massive amount of structured, semi-structured, poly-structured, and unstructured data that grows over time and years. Real time analytics needs an additional requirement of time bound computation. Techniques from Artificial Intelligence may be applied to find patterns and relations in unstructured data. Big data analytics techniques can be scaled using parallel and distributed computing technologies, without compromising on accuracy of results. However, traditional data analytics techniques on big data have certain issues regarding scalability and performance, which are discussed below.

1) An integrated big data analytics architecture that uses fault tolerant and able to handle large voluminous and varied data in batches as well as in a continuous stream in real time is still missing.

2) Distributed computing is a good solution to handle the massive volume of big data. However, most of the Artificial Intelligence, data mining, and statistical analysis approaches are not originally designed for distributed computation. Although distributed algorithms have been proposed in the literature [6], [7], [8], they are mostly academic research and lack robust implementation, considering various MapReduce frameworks.

3) A big data store always does not have a uniform data format. In order to process big data analytics, heterogeneous data captured through sensors of various types. Hence, intelligent algorithms are required to find a coherent meaning from disparate data. This increases the complexity of analytics.

4) Unstructured, semi-structured and poly-structured data introduces more and new problems, such as data inconsistency and redundancy. Data pre-processing is costly due to heterogeneous data and massive volume of data. Traditional data analytics techniques handle inconsistent and noisy data are found costly in terms of time and space complexities.

5) Big data analytics required to mine datasets at different levels of abstraction. This significantly increases the complexity of the analytics methods, however, enabling biologists to analyze the data at various level of abstractions help understanding the interest of semantics biological data.

CONCLUSION:

Meeting the challenges by big data will be difficult. The volume of data is already large voluminous and increasing every day. The velocity of its generation and growth is increasing day by day, driven in part by the proliferation of internet connected devices. Furthermore, the variety of data is being generated

and expanding, and organization's capability to capture and process this data is limited. Big data is an emerging trend and the need for rising in all science and engineering domains.

REFERENCES:

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, PP. 107–113, 2008.

[2] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: a runtime for iterative mapreduce," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM, 2010, pp. 810–818.

[3] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley,M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012,pp. 2–2.

[4] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein, "Graphlab: A new framework for parallel machine learning," arXiv preprint arXiv:1408.2041, 2014.

[5] Cisco, "Cisco visual networking index: global mobile data traffic forecast update, 2014–2019," Cisco Public Information, 2015.

[6] A. Choudhury, P. B. Nair, A. J. Keane et al., "A data parallel approach for large-scale gaussian process modeling." in SDM. SIAM, 2002, pp. 95–111.

[7] A. E. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski, "Using bayesian model averaging to calibrate forecast ensembles," Monthly Weather Review, vol. 133, no. 5, pp. 1155–1174, 2005.

[8] R. Wright and Z. Yang, "Privacy-preserving bayesian network structure computation on distributed heterogeneous data," in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004, pp. 713–718.