International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-5, March 2020

Analytical Design of the DIS Architecture: The Hybrid Model

Mahesh S Nayak, M. Hanumanthappa, B R Prakash, Dattasmita H V

Abstract*: In the last decades, and due to emergence of Internet appliance, there is a strategical increase in the usage of data which had a high impact on the storage and mining technologies. It is also observed that the scientific/research field's produces the zig-zag structure of data viz., structured, semi-structured, and unstructured data. Comparably, processing of such data is relatively increased due to rugged requirements. There are sustainable technologies to address the challenges and to expedite scalable services via effective physical infrastructure (in terms of mining), smart networking solutions, and useful software approaches. Indeed, the Cloud computing aims at data-intensive computing, by facilitating scalable processing of huge data. But still, the problem remains unaddressed with reference to huge data and conversely the data is growing exponentially faster. At this juncture, the recommendable algorithm is, the well-known model i.e., MapReduce, to compress the huge and voluminous data. Conceptualization of any problem with the current model is, less fault-tolerant and reliability, which may be surmounted by Hadoop architecture. On Contrary case, Hadoop is fault tolerant, and has the high throughput which is recommendable for applications having huge volume of data sets, file system requiring the streaming access. The paper examines and unravels, what efficient architectural/design changes are necessary to bring the benefits of the Everest model, HBase algorithm, and the existing MR algorithms.

Keywords: MapReduce, HPC, HBase, Hadoop, HDFS, Cloud Computing, Google MapReduce, HPQL, Cap3, ADT.

I. INTRODUCTION

It is a general observation that, the Data-intensive computing do the task of combining, review and comprehending data with respect to the volume of data that pushes up the borderline of existing technologies. [1]

For instance, it is difficult to manage the multi-structured data just by using the traditional relational database technology. Such problem is very often in the Web-based companies such as Google, and LinkedIn, where the data are large voluminous and supposed to be retrieved in a cost-effective: Cost and Speed. In such context, MapReduce and Google File System possibly solve the deficiency of the reliability and effectiveness. Later which the advent of Bigtable was built into the DBMS System. It is possible to search and return the results of millions of pages in milliseconds or less with the assistance of the algorithms (Google's MapReduce framework) having search services enablers [3]. It is still a challenge even today to analyze the "big data". Inadvertently, there were eras of technological inventions and invasions for analyzing big data and thus a significant impact resulted in an advanced analytic approach viz., MapReduce, Hadoop and MapReduce which are extensions to existing relational databases [4].

Traditionally, the possible four steps in Map Reduce are:

- Data Splitting
- The output of map method is passed as input to reduce procedure.
- The inputs to reduce procedure is thus sorted accordingly basing the intermediate keys.
- Quality of services.

For instance, the Salesforce.com- a first mover of cloud computing (1999). The concept of delivering business applications through a simple website was introduced in the era. Subsequently, the Amazon (2002), and most recently the Google docs had brought the cloud computing to public on 2006. Surprisingly the Elastic Compute Cloud (2006) was induced by the Amazon and no sooner the Microsoft began the era of Microsoft Azure (2010) in cloud computing. [1].

1.1 Apropos of Characteristics of Data in terms of Big Data:

When dealt with huge data, the difficult scenario would be handling the large set of data. The growing of data is always a challenge to be addressed of. Factually, Terabytes to Petabytes, the representants for the huge data in the digital world. The Principal strategies with reference to big data are the 3V's: Velocity, Volume, and Variety. Indeed, the data will be structured and unstructured in terms of retrieval. But, in general, business strategies finds it difficult to analyze and interpret the unstructured data. The tabular representation, Table-1 depicts the analytical usage of the data during the decades.

Table 1: DIS Architecture[1]

Parameters	90's	00's	10's
Capacity	2.1GB	200GB	3000GB
Price	$157	$0.5	$0.05
Speed	16.6MBPS	56.5MBPS	210MBPS
Time Required	126sec	58min	4hours

Revised Manuscript Received on March 30, 2020.
* Correspondence Author
Mahesh S Nayak , Research scholar, Bharathiar University, Coimbatore – 641 046
Dr. M. Hanumanthappa, Professor, Department of Computer Science & Applications, Bangalore University, Bangalore.
***Dr. B R Prakash**, Assistant Professor, Department of Computer Science, Government First Grade College, Tiptur, Karnataka.
Dattasmita H V ICT Manager. Tumakuru Smart City Limited, Tumakuru, Karnataka, India

1.2 Sample events generating data [26]

- On an average, Facebook has not less than 800 million active users. Not less than 250 billion photos are uploaded to Facebook daily. Facebook generates 04 Petabytes of data every day.
- The number of text messages that will be sent today is greater than the population of the entire world.
- 269 billion emails are sent every day.
- Not less than 500 million tweets will be sent every day.
- Minimum of 400 tweets per minute contains the YouTube links.
- More than 7.75 billion camera phones worldwide with GPS.
- Now, the total data may be estimated to be 1 ZB, in 2020 it may be projected at an estimated to be 35ZB.

1.3 MapReduce

A MapReduce algorithm consists of a mapping process (or technique conducting filtering and sorting, i.e. sorting First Name workers into queues, as one queue for each name), and a reduction function (performing a summary task, i.e. counting the number of patterns in each queue, outputting the name-frequency). The Map-Reduce's main contributions are not the empirical map-reduction process, but the scalability and fault-tolerance obtained by execution engine optimization for the number of applications. In addition, a single-threaded Map-Reduce implementation would normally be no faster than the conventional (non-MapReduce) implementations; any improvements are commonly seen with multi-threaded implementations. [13] [14]

1.4 Hadoop

Hadoop is a flattering remark for Map-Reduce data analysis when data is a challenge. Its meticulous use cases includes data search, analysis of data, commentate on details of data, exhaustive file indexing (viz., log files or data from web crawlers), and contrasting data processing chores using a colloquially technique in the developing world i.e., “Big Data.” [13] The Hadoop's general architecture consists of three basic components; a distributed file system, a framework for parallel programming, and a resource / job management system. Hadoop is compatible with Linux / Windows operating systems, but it is also known and assumed to operate with BSD, Mac OS / X and Open Solaris. [13]

1.4.1 Hadoop Distributed File System (HDFS)

Nevertheless, Hadoop, an open-source, Java-based clustered file system, called the HDFS, which is based on distributed computing cost-effective, robust, and scalable. The HDFS architecture is highly fault-tolerant, and the low-cost hardware is ideal for construction. [13]

1.4.2 Hadoop MapReduce

The Hadoop framework's purpose is to store and (disseminated) process huge amounts of data sets through computer clusters using a Map-Reduce paradigm [13]

The input (file-set) is divided into different smaller packets, which are interpreted separately (the "path" portion of each other) using Path-Reduce. Until after the task is completed, the individual results are then compiled and analyzed as clusters (the "reduce" part). If a particular file is disproportionately big, which can significantly decrease the performance of the search time, then these tasks can be further divided into several "Hadoop splits". [13]

ARCHITECTURE OF DATA INTENSIVE SERVICE

An HPC architecture is described by innumerable processors, viz., the memory heaps, robust system administration — cohesively soldered on a common-crosswise over rack- servers. An HPC compute node is made up of generic servers-nodes, perhaps the algorithm plays a vital role in filling a whole of the server with power-hungry racks. [06]

The HPC task would ordinarily include a reproduction of numerical models or investigation of information from logical instrumentation. It would be full of equipment and frameworks, programming wrangled up, that allow scientists to produce new science at the power of high-performance computing. [06]

II. DATA INTESIVE SCALABLE COMPUTING

DISC tools, emerging technology in the field of data analysis, used to grub up on huge Web datasets.The system offers the capability to analyze comparably larger quantity of stored data while streaming software is configured to process multiple updates per second, which can also be implemented through MapReduce (MR) and Streaming.

Nonetheless, Hadoop1, a distributed computing architecture that applies the Map-Reduce model (Dean and Ghemawat, 2004) along with an adjunct distributed file system named Hadoop Distributed File System (HDFS). Hadoop also facilitates the distributed processing by means of a basic functional programming model of bulk data sets across different computer clusters. [1]http://hadoop.apache.org
Where will "big data" be established, however? Of note, the concept is appropriate and progresses with advances in technology. Yes, a terabyte was thought huge 30 years ago, although nowadays such a volume of data is a very common issue.

III. DIFFERENTIATION ANALYSIS

3.1 Google Map Reduce

The MapReduce architecture allows the programmers conventional programming style to create/generate a map method, a key-value pair associated with the input data to generate a set of intermediate key-value pairs, and a reduce method to merge all intermediate values associated with the such intermediate key. [05]

IV. EXISTING SYSTEMS

Everest enforces the Platform as a Service (PaaS) paradigm as a means of providing functionality via remote web and programming interfaces as a key compared with conventional high-performance computing platforms. Everest introduces and incorporates, refractorily, an agent with servers and device clusters, using the tool that acts as a mediator between the interface and services. [08].

Whereas, the purpose of the DIS initiative was largely to support services that include a limited amount of data, but a minute change is required to effectively implement the DIS Everest system. Since such data migration from external storage to the network must be applied and vice versa, bypassing the interface. In addition, the incorporation of the agent into Hadoop application components or similar technology for data storage and processing on the cluster is also needed. [07]

Secondly, the HBase [09], the Bigtable distributed storage system's, an open source version. This operates in parallel to HDFS and provides large-scale data-management capabilities. HBase is essentially written for device freedom in Java. [10].

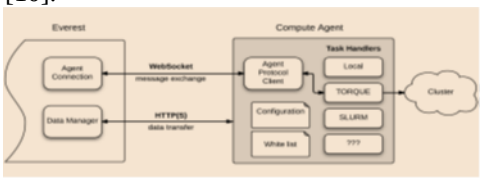

Figure 1: Block Diagram of the proposed application

V. PROPOSED ARCHITECTURE

The proposed architecture for the DIS modularly uses the two algorithms having the intensive services, which acts as the platform for processing and providing the services through PaaS. In section 5, it was discussed about the base model and architectural design of the Everest which justify being a web-based distributed computing platform under High Performance Computing, which will facilitate the Data Intensive Scalable Computing algorithm along with the rapid deployment of robust computational services.

The blocks will hold the metadata information which are logically mapped to other nodes. In case of any request by the client to read/write onto the filesystem, the respective Data node serves the request respectively through parallel processing, the operations viz., CRUD on the directory will be served by the Name Node.

The Name Node will be run by the dedicated machine which is deployment of HDFS. The Software does not allow multiple instances of the Data Node to run on the same machine simultaneously, hence each machine in the cluster typically run one instance of the Data Node software. The Name Node is never concerned of user data but of metadata repository and control. A Special kind of log is maintained by Name Node, named Edit Log, for the persistence check of metadata. [7]

VI. PERFORMANCE

$$Efficiency\ E(p)=\frac{T(1)}{pT(p)}$$

It is necessary to comprehend the performance and abilities of the Hadoop with respect to the map-reduce function and other cloud technologies. The analysis glimpses out a conclusion that performance of Hadoop with respect to the non-homogeneous data [16], and the inbuilt feature for load balancing in Hadoop which is due to its dynamic global level scheduling capability which uses static-task partitioning. [15] The analysis gist up the procedure which was carried out to analyse the performance and scalability using the *Cap3* program.

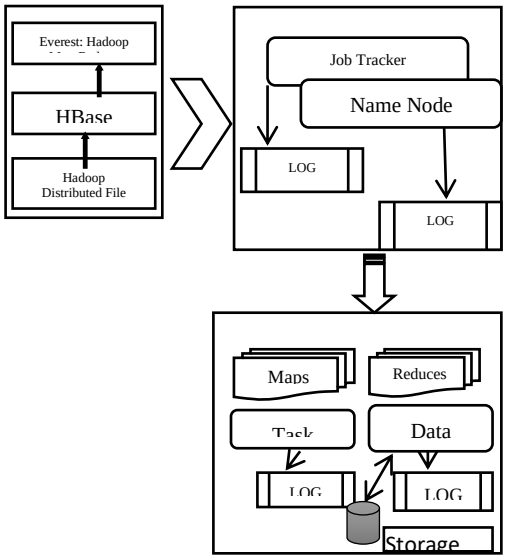

Figure 1: Proposed System Architecture Flow Chart

VII. FUNCTIONAL PROGRAMMING

In lieu of complicated structures, multiple processors are used by the side of each other, viz., in a computer cluster. In the above scenarion, a centralized system, the speed and flexibility become vital, and modern supercomputers has its unique approaches from enhanced InfiniBand systems to three-dimensional torus interconnects.

```
class Mapper
        method Map (id a; doc d)
                for all term t ∈ 2 doc d do
                        Emit (term t; count 1)
class Reducer
        method Reduce (
        term t; counts [c1; c2;:::])
                sumî0
                for all count c ∈ 2 counts
                [c1; c2;:::]
                do
                    sum î sum + c
                Emit (term t; count sum)
```

Snippet 1: Code snippet of Python API

To be exact, three distinct and related definitions are applied to MapReduce.

First, it's a template for programming. Furthermore, it refers to the \runtime execution system that manages program execution. Finally, the programming model is applied by code and there by execution framework: For example, Google's proprietary based implementation vs. the open-source Hadoop implementation in Java. [12]

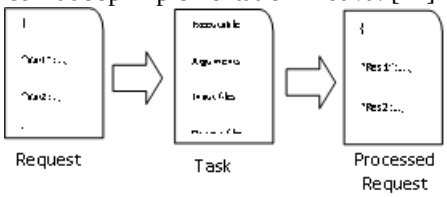

Figure 2: Proposed System Architecture

9.1 Sample dataset

The files in the dataset are usually organized in an n-gramper line containing the year, the number of occurrences in the year in question, and the number of pages in which the word / key occurs in the year in question. A sample line from a n-gram file that shows the number of occurrences for the 2007 and 2008 e-content terms.

```
(0,” HOMEY 2007 15 09 17")
(20, "HOMEY 2008 20 25 29")
(40, "HONOUR 2007 514306 386142 442520")
(72, " HONOUR 2008 657559 776543 498765")
```

Everest takes an API-centric philosophy by building the structure around an interactive programming interface. Usually apps are exposed via the REST API to allow remote access to those applications from other systems. It enables automation mechanisms from third parties to compose or integrate host Everest software with other apps.[25] Thus the incorporation of the HBase is necessary.

The sample set contains an example of a program using the Python API. It implements a simple workflow that consists of running four different applications - A, B, C and D. [25]

```
session = Everest. Session(' https://everest.distcomp.org', user = '...', password = '...' )
service = Everest. Service(' INSERT APP ID', session)
service = Everest. Service('...', session)
service = Everest. Service('...', session)
serviceD = Everest.Service('...', session)
job = serviceA.run({'a': '...'})
jobB = serviceB.run({'b': jobA.output('out1')})
jobC = serviceC.run({'c': jobA.output('out2')})
jobD = serviceD.run({'d1': jobB.output('out'), 'd2': jobC.output('out')})
print (jobD.result())
```

Snippet 2: Code snippet of Phyton API/Algorithm

The program uses an user token to import Everest (Python) API and create a new session. In order to authenticate themselves, each client accessing Everest must present such token with their query. To access the applications, an App object will be generated. The software lifecycle is started by calling the application object's run) (method). As a Python dictionary, outputs are passed with keys and values corresponding to the output names and values. The run) (function output will return a Work object used to check the status of the work. [25]

VIII. IMPLEMENTATIONAL ASPECTS

The creation of computation intensive research methods and programs involves working with stakeholders and evaluating these observational facts and figures. The dataset can differ in proportion between the analytical figures. Addressing successful teamwork is difficult, but often they can view its implementation very differently. Developing infrastructure that includes the most suitable algorithm as mentioned above was a right challenge. The goal is to create a program that is noticeable, the stakeholder appreciates as distinctive and optimally efficient algorithm.

IX. RESULT ANALYSIS

Indeed, estimating the cost of an enhanced DIS algorithm is quite difficult, because of the inherent distributed system, and certainly, the hidden cost of the shuffling phases HBase. The proposed model has three main steps of an Everest: The Map and Reduce functionalities, and the volume of the data to be shuffled and stored in HBase. The cost associated with the function instance with the largest input is considered. The following table depicts the Azure clouds pricing snapshot with reference to the above algorithm and architecture. Here X represents unknown. Comparatively, the Azure and EC2 differs at the extreme levels. If the memory of 1.7 GB is considered for the above formulae, EC2 consumes less memory than Azure. However, the prize is slightly higher through the efficiency.

Type of Cloud	Mem. (GB)	Compute units	Storage (GB)	Charges per hours
Azure	1.75	X+1	224+70	Rs. 6.37/-
Azure +2	3.5	X+2	489+135	Rs. 12.74/-
Azure +3.5	7	X+4	999+285	Rs. 25.47/-
Azure +7.0	14	X+8	2039+605	Rs. 50.95/-
Hadoop	1.7	1	160	Rs. 6.44/-

X. CONCLUSION

Data Intensive relates to the use of mining on the Internet. The huge network success and large delivery of information generation and computation have dramatically increased [4,5]. The huge growth in use and analysis of data has also contributed to the enormous potential of a broader range of consumers and their apps. In turn, there have been extraordinary issues with the results. Data intensive computation and substantial-scale data processing innovations have been applied at various levels of system structures that are usable in virtual networks for over Twenty decades including concurrent and hierarchical interaction database management systems. The design suggested would fill the appropriate distance between the DIS and HDC.

REFERENCES

1. N Suresh Goud. Data Intensive Computing in the Clouds, Thesis, Department of Computer Science and Engineering, National Institute of Technology Rourkela.
2. Seema Maitreya, C.K. Jha. MapReduce: Simplified Data Analysis of Big Data, Procedia Computer Science 57 (2015) 563 – 571.
3. J R Swedlow, G Zanetti, C Best. Channeling the data deluge. Nature Methods, 2011, 8: 463-465
4. 2. G C Fox, S H Bae, et al. Parallel Data Mining from Multicore to Cloudy Grids. High-Performance Computing and Grids workshop, 2008
5. Priya Trivedi, and Sanya Harneja. Data-Intensive Computing: A Parallel And Distributed Approach For Big Data, International Journal Of Innovative Research In Technology, 2014 IJIRT | Volume 1 Issue 5 | ISSN: 2349-6002, Page: 867-870.
6. Ahmed Arefin. Introduction to HPC Clusters, website: www.scientificprogramming.io/ learn-hpc.
7. Oleg Sukhoroslov and Alexander Afanasiev. Development of Data-Intensive Services with Everest, Proceedings of the XIX International Conference “Data Analytics and Management in Data Intensive Domains” (DAM DID/RCDL’2017), Moscow, Russia, October 10-13, 2017

8. Everest. http://everest.distcomp.org/
9. HBase: The Apache HBase Project. http://hbase. apache.org/
10. Huang, J., Ouyang, X., Jose, J., Wasi-ur-Rahman, M., Wang, H., Luo, M., Subramoni, H., Murthy, C., Panda, D.: High-performance design of HBase with RDMA over InfiniBand. In: IEEE 26th International Parallel & Distributed Processing Symposium (IPDPS) (2012)
11. Oleg Sukhoroslov, Sergey Volkov, and Alexander Afanasiev. A Web-based Platform for Publication and Distributed Execution of Computing Applications, Link: http://everest.distcomp.org/research/ispdc2015.pdf.
12. Jimmy Lin and Chris Dyer. Manuscript: *Data-Intensive Text Processing with MapReduce*, University of Maryland, College Park, April 11, 2010.
13. "MongoDB: Terrible MapReduce Performance". Stack Overflow. October 16, 2010. "The MapReduce implementation in MongoDB has little to do with map reduce apparently. Because for all I read, it is single-threaded, while map-reduce is meant to be used highly parallel on a cluster. ... MongoDB MapReduce is single threaded on a single server...".
14. Link: https://en.wikipedia.org/wiki/MapReduce#cite_note-3
15. Thilina Gunarathne, Tak-Lon Wu, Jong Youl Choi, Seung-Hee Bae, Judy Qiu. Cloud Computing Paradigms for Pleasingly Parallel Biomedical Applications, Bloomington, Link: http://grids.ucs.indiana.edu/ptliupages/publications/ecmls_jour_15.pdf.
16. J. Ekanayake, T. Gunarathne, J. Qiu, and G. Fox. Cloud Technologies for Bioinformatics Applications, Accepted for publication in Journal of IEEE Transactions on Parallel and Distributed Systems, 2010.
17. Erin O' Meara. Developing a Recordkeeping Framework for Social Scientists Conducting Data-Intensive Research, Society of American Archivists, April 2008.