Cluster based approach for processing Voting Data Using MapReduce Approach

[*1]R. Dhanalakshmi, [*2]S. Kalaiarasi, [*3]K. Sivagangai, [*4]J. RamyaRajalakshmi(Asst. Prof.,)
Department of computer science and engineering
Sri Aravindar Engineering College, Sedarpet, Villupuram Dist,.

***Abstract*—Big Data is the most promising research area in the field of Cloud Computing. In areas like social networking applications, e- commerce, finance, health care, education, etc, huge amount of data is being accumulated. As new data and updates are constantly arriving, the results of data mining applications become stale and obsolete over time. Hence another approach is required for mining big data. Hence in this paper, a cluster based approach is introduced which guarantees an efficient processing of big data. This approach is experimented with the real time voting data of an election. Furthermore, the performance is compared with the existing data mining applications of big data.**

***Key words*— Cloud computing; Big Data; MapReduce**

I. INTRODUCTION

Now-a-days, in Information communication Technology (ICT), cloud computing plays a vital role to perform complex computing and large scale operations. Cloud computing has the advantages like virtualized resources , parallel processing, security and data service integration with scalable data storage. Cloud computing can not only minimize the cost and restriction for automation and computerization by individuals and enterprises but can also provide reduced infrastructure maintenance cost, efficient management, and user access . As a result of the these advantages, a number of applications that influence various cloud platforms have been developed and resulted in a tremendous increase in the scale of data generated and consumed by such applications. Some of the first adopters of big data in cloud computing are users that deployed Hadoop clusters in highly scalable and elastic computing environments provided by vendors, such as IBM, Microsoft Azure, and Amazon AWS. [1]

In recent years, Big data is the new emerging concept in the field of information technology. The term Big data refers increases in the volume of data which are difficult to store, process and analyze through traditional database technologies. The data may be structured, semi-structured or unstructured. Big data can be characterized as four Vs. Volume, Variety, Velocity and Value. Thus big data can also be defined as a set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are diverse, complex and of massive scale.

II. COMPARING MAP-REDUCE TO TRADITIONAL PARALLELISM

In order to appreciate what map-reduce brings to the table, I think it is most meaningful to contrast it to what I call traditional computing problems. I define "traditional" computing problems as those which use libraries like MPI, OpenMP, CUDA, or pthreads to produce results by utilizing multiple CPUs to perform some sort of numerical calculation concurrently. Problems that are well suited to being solved with these traditional methods typically share two common features:

- They are cpu-bound: the part of the problem that takes the most time is doing calculations involving floating point or integer arithmetic
- Input data is gigabyte-scale: the data that is necessary to describe the conditions of the calculation are typically less than a hundred gigabytes, and very often only a few hundred megabytes at most

Item 1 may seem trivial; after all, computers are meant to compute, so wouldn't all of the problems that need to be parallelized be fundamentally limited by how quickly the computer can do numerical calculations? Traditionally, the answer to this question has been yes, but the technological landscape has been rapidly changing over the last decade. Sources of vast, unending data (e.g., social media, inexpensive genenome sequencing) have converged with inexpensive, high-capacity hard drives and the advanced filesystems to support them, and now data-intensive computing problems are emerging. In contrast to the aforementioned traditional computing problems, data-intensive problems demonstrate the following features:

- Input data is far beyond gigabyte-scale: datasets are commonly on the order of tens, hundreds, or thousands of terabytes
- They are I/O-bound: it takes longer for the computer to get data from its permanent location to the CPU than it takes for the CPU to operate on that data

III. RELATED WORKS

To perform complex computations and massive scale operations cloud computing is the powerful technology. Because it eliminates the need to maintain expensive computing hardware, dedicated space and software. Big data generated through cloud computing has been observed. In [2] the rise of big data is reviewed. The definition characteristics and classification of big data along with some discussions on cloud computing are introduced. The relationship between big data and cloud computing, big data storage systems and Hadoop technology are also discussed. Furthermore, research challenges are investigated, with focus on scalability, availability, data integrity, data transformation, data quality, data heterogeneity, privacy, legal and regulatory issues and governance. Lastly, open research issues that require substantial research efforts are also summarized.

Despite the advances in hardware for hand-held mobile devices, resource-intensive applications (e.g., video and image storage and processing or map-reduce type) still remain off bounds since they require large computation and storage capabilities. Recent research has attempted to address these issues by employing remote servers, such as clouds and peer mobile devices. For mobile devices deployed in dynamic networks (i.e., with frequent topology changes because of node failure/unavailability and mobility as in a mobile cloud), however, challenges of reliability and energy efficiency remain largely unaddressed. To the best of our knowledge, we are the first to address these challenges in an integrated manner for both data storage and processing in mobile cloud,an approach we call k-out-of-n computing. In our solution, mobile devices successfully retrieve or process data, in the most energy-efficient way, as long as k out of n remote servers are accessible. Through a real system implementation we prove the feasibility of our approach. Extensive simulations demonstrate the fault tolerance and energy efficiency performance of our framework in larger scale networks [3].

Data analysis is an important functionality in cloud computing which allows a huge amount of data to be processed over very large clusters. MapReduce is recognized as a popular way to handle data in the cloud environment due to its excellent scalability and good fault tolerance. However, compared to parallel databases, the performance of MapReduce is slower when it is adopted to perform complex data analysis tasks that require the joining of multiple data sets in order to compute certain aggregates. A common concern is whether MapReduce can be improved to produce a system with both scalability and efficiency. Map-Join-Reduce, a system that extends and improves MapReduce runtime framework to efficiently process complex data analysis tasks on large clusters is introduced in [4]. They first proposed a filtering-join-aggregation programming model, a natural extension of MapReduce's filtering-aggregation programming model. Then, presented a new data processing strategy which performs filtering-join-aggregation tasks in two successive MapReduce jobs. The first job applies filtering logic to all the data sets in parallel, joins the qualified tuples, and pushes the join results to the reducers for partial aggregation. The second job combines all partial aggregation results and produces the final answer. The advantage of their approach is joining multiple data sets in one go and thus avoid frequent checkpointing and shuffling of intermediate results, a major performance bottleneck in most of the current MapReduce-based systems.

In recent days sensors plays a vital role and they are becoming ubiquitous in collecting information and having a great influence in industrial applications to intelligent vehicles, smart city applications, and healthcare applications, etc. but each and every applications uses different types of sensors depending upon their usage. The rate of increase in the amount of data produced by these sensors is much more dramatic since sensors usually continuously produce data. It becomes crucial for these data to be stored for future reference and to be analyzed for finding valuable information, such as fault diagnosis information. So in [5] a scalable and distributed architecture is described for sensor data collection, storage, and analysis. The system uses several open source technologies and runs on a cluster of virtual servers. GPS sensors are used as data source and run machine-learning algorithms for data analysis

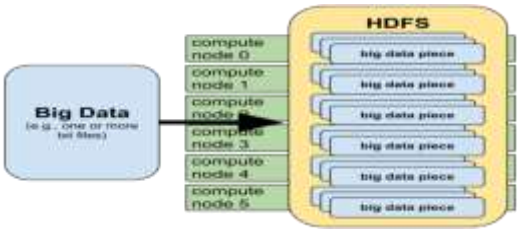
I. MAP REDUCE

Map Reduce is the heart of Hadoop. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The term Map Reduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name Map Reduce implies, the reduce job is always performed after the map job.

II. HADOOP - A MAP-REDUCE IMPLEMENTATION

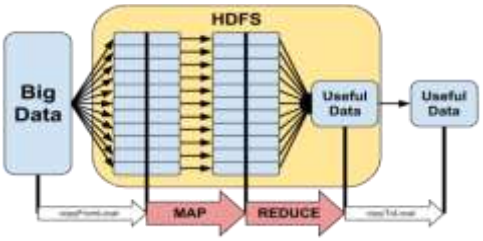
The idea underpinning map-reduce–bringing compute to the data instead of the opposite–should sound like a very simple solution to the I/O bottleneck inherent in traditional parallelism. However, the devil is in the details, and implementing a framework where a single large file is transparently diced up and distributed across multiple physical computing elements (all while appearing to remain a single file to the user) is not trivial. Hadoop, perhaps the most widely used map-reduce framework, accomplishes this feat using HDFS, the Hadoop Distributed File System. HDFS is fundamental to Hadoop because it provides the data chunking and distribution across compute elements necessary for map-reduce applications to be efficient. Since we're now talking about an actual map-reduce implementation and not an abstract concept; let's refer to the abstract compute elements now as compute nodes.

HDFS exists as a file system into which you can copy files to and from in a manner not unlike any other file system. Many of the typical commands for manipulating files (ls, mkdir, rm, mv, cp, cat, tail, and chmod, to name a few) behave as you might expect in any other standard file system (e.g., Linux's ext4). The magical part of HDFS is what is going on just underneath the surface. Although it appears to be a file system that contains files like any other, in reality those files are distributed across multiple physical compute nodes. When you copy a file into HDFS as depicted above, that file is transparently sliced into 64 MB "chunks" and replicated three times for reliability. Each of these chunks are distributed to various compute nodes in the Hadoop cluster so that a given 64 MB chunk exists on three independent nodes. Although physically chunked up and distributed in triplicate, all of your interactions with the file on HDFS still make it appear as the same single file you copied into HDFS initially. Thus, HDFS handles all of the burden of slicing, distributing, and recombining your data for you.



I. SYSTEM DESIGN

All the Indian citizens should vote. It is very difficult to count and process the voting data in order to reveal the results as the population gets increased. Hence this cluster based data processing is introduced to process the big data. This work is performed in the Hadoop open source framework and with the help of eclipse. Initially the real time voting data is moved the HDFS which is installed in four machines. Then the data is splitted and passed to the mapper function and converted into key, value pairs (K, V). The key is unique whereas value may have repeated values. This key, value pair is given as input to the reduce function so that the reduced output is represented as the final result. This can be represented in figure 3.



Once a map-reduce job is initiated, the map step

- Launches a number of parallel mappers across the compute nodes that contain chunks of your input data
- For each chunk, a mapper then "splits" the data into individual lines of text on newline characters (\n)
- Each split (line of text that was terminated by \n) is given to your mapper function
- Your mapper function is expected to turn each line into zero or more key-value pairs and then "emit" these key-value pairs for the subsequent reduce ste

That is, the map step's job is to transform your raw input data into a series of key-value pairs with the expectation that these parsed key-value pairs can be analyzed meaningfully by the reduce step. It's perfectly fine for duplicate keys to be emitted by mappers.

A. The Reduce Step

Once all of the mappers have finished digesting the input data and have emitted all of their key-value pairs, those key-value pairs are sorted according to their keys and then passed on to the reducers. The reducers are given key-value pairs in such a way that all key-value pairs sharing the same key always go to the same reducer. The corollary is then that if one particular reducer has one specific key, it is guaranteed to have all other key-value pairs sharing that same key, and all those common keys will be in a continuous strip of key-value pairs that reducer received.

Your job's reducer function then does some sort of calculation based on all of the values that share a common key. For example, the reducer might

calculate the sum of all values for each key (e.g., the word count example). The reducers then emit key-value pairs back to HDFS where each key is unique, and each of these unique keys' values is the result of the reducer function's calculation.

IV. CONCLUSION

As the population increases the process of counting and processing the number of votes also gets difficult. Thus Big data plays the major role while considering the applications such as social networking applications. Iterative mining is required to obtain the updated data. This is performed in this work. The real time voting data of an election is taken for year of a particular area and the operations are performed. The big data is processed iteratively so that resulting in an accurate mining of the updated and new growing data. The person who got more number of votes is declared as the winner. This proposed cluster based big data processing process is fast and efficient when comparing with the existing approaches.

REFERENCES

[1] Q. Yang, "Introduction to the IEEE Transactions on Big Data", IEEE Transactions on Big data, no. 1, vol. 1, pp. 2-14, January 2015.

[2] S. Chen, Q. Wang, G. Yu and Y. Zhang, "i2MapReduce: Incremental MapReduce for Mining Evolving Big Data", IEEE transactions on Knowledge and Data Engineering, vol. 27, no. 7, pp. 1906-1919, July 2015.

[3] Hashem, I. Yaqoob, S. Mokhtar, A. Gani and S. Khan, "The rise of "big data" on cloudcomputing: Review and open research issues", Elsevier, no. 47, pp. 98-115, 2015.

[4] C. Chen, W. Myounggyu, R. Stoleru and G. G. Xie, "Energy-Efficient Fault-Tolerant Data Storage and Processing in Mobile Cloud", IEEE Transactions on cloud computing, pp. 28-41, March 2015.

[5] D. Dahiphale, R. Karve, A. V. Vasilakos, H. Liu, "An Advanced MapReduce: Cloud MapReduce, Enhancements and Applications", IEEE Transactions on Network and service Management, pp. 101-115, April 2014.

