International Journal of Research in Advent Technology, Special Issue, March 2019
E-ISSN: 2321-9637
International Conference on Technological Emerging Challenges (ICTEC-2019)
Available online at www.ijrat.org

Secure Provable And Survey On Mapreduce In Cloud Computing

B.Rajani[1], E.V.N.Jyothi[2], Dr.V.Purna Chandra Rao[3].
Ph.D.Scholar[1], Department of Computer Science and Engineering, Shri Jagdishprasad Jhabarmal Tibrewala University,India.
(Assoc. Professor in CSE Dept. in KMIT)
Ph.D.Scholar[2], Department of Computer Science and Engineering, Shri Jagdishprasad Jhabarmal Tibrewala University,India.
(Assoc. Professor in CSE Dept. in PACEITS)
Professor[3], Department of Computer Science and Engineering, SVIT, Hyderabad, India.

Abstract: The computer industry is being challenged to develop methods and techniques for affordable data processing on large datasets at optimum response times. The technical challenges in dealing with the increasing demand to handle vast quantities of data is daunting and on the rise. One of the recent processing models with a more efficient and intuitive solution to rapidly process large amount of data in parallel is called MapReduce. Cloud Computing delivers computing resources so that it was appealing great thoughtfulness. The experimental results of CMR show that it is more efficient and development faster than other implementations of the MR method. Also Cloud Map Reduce can be improved to Support stream data processing, edible pricing using Amazon Cloud's spot instances. Improve speed-up to process over traditional MR that processes more than 30% for large data sets and provides flexibility and scalability. CMR is aimed for handling batch data with major modifications are made in the basic structure of CMR. The pipelining between Map and Reduce phases for supporting flow data processing is introduced here also previous feature of CMR is now called as Continuous Cloud MapReduce (C-CMR). The architecture of CMR consists of various components such as Simple Storage Service, Input/Map queue, Simple database, Map Workers, Combiners; etc. CMR is suitable to start on a Map Reduce job since the nodes are symmetric.

Keywords*:* Cloud Computing, MapReduce, spot market, Elastic cloud computing.

1. INTRODUCTION

Cloud computing is the released of computing as a service rather than a product, shared resource, software and information are provided to computers and other devices as a utility over a network. It gives computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that deliver the services.

Today a growing number of companies have to process huge amounts of data in a cost-efficient manner. Typical representatives for those companies are operator of Internet search engines, like Google, Yahoo or Microsoft. Cloud computing providers deliver applications via the internet, which are accessed from web browsers and desktop and mobile applications. Cloud storage is a model of networked online storage where data is stored on virtualized pools of storage which are generally hosted by third parties. Hosting companies operate large data canter's and people who require their data to be hosted buy or lease storage capacity from them and use it for their storage needs. The data center operators in the background, virtualizes the resources according to the requirements of the customer and expose them as storage pools, which the customers can themselves use to store files or data objects.

The question is how to process large amounts of distributed data quickly with good response times and replication at minimum cost? One of the best ways for huge data processing is to perform parallel and distributed computing in a cloud computing environment. Cloud computing as a distributed computing paradigm aims at large datasets to be processed on available computer nodes by using a MapReduce framework. MapReduce is a software framework introduced to the world by Google in 2004; it runs on a large cluster of machines and is highly scalable [3]. It is a high-performance processing technique to solve large-scale dataset problems. MapReduce computation processes petabyte to terabyte of unit data on thousands of processors. Google uses MapReduce for indexing web pages. Its main aim is to process large amount of data in parallel stored on a distributed cluster of computers. This study presents a way to solve large-scale dataset processing problems in parallel and distributed mode operating on a large cluster of machines by using MapReduce framework. It is a basis to take advantage of cloud computing paradigm as a new realistic computation industry standard.

The model is inspired by the map and reduces functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as in their original forms. The key contributions of the MapReduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once. As such, a single-threaded implementation of MapReduce (such as Mongo DB) will usually not be faster than a traditional (non-MapReduce) implementation, any gains are usually only seen with multi-threaded implementations. Only when the

optimized distributed shuffle operation (which reduces network communication cost) and fault tolerance features of the MapReduce framework come into play, is the use of this model beneficial.

2. OVERVIEW

MapReduce has been facilitated by Google as a programming framework to analyse massive amounts of data. It uses for distributed data processing on large datasets across a cluster of machines. Since the input data is too large, the computation needs to be distributed across thousands of machines within a cluster in order to finish each part of computation in a reasonable amount of time. This distributed concept implies to parallelize computations easily and using re-execution as the main technique for fault tolerance. J.Dean and S.Ghemawat [3] from Google Inc. published a paper in 2004 describing MapReduce. Google never released their implementation of MapReduce. Finally, the Apache Company made available a concrete implementation of MapReduce named Hadoop [4]. MapReduce allows developers to perform complex computations in a simple way while hiding the details of data distributation, parallelization, and fault tolerance.

MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Processing can occur on data stored either in a file system (unstructured) or in a database (structured). MapReduce can take advantage of locality of data, processing it on or near the storage assets in order to reduce the distance over which it must be transmitted.

Map step: Each worker node applies the "map()" function to the local data, and writes the output to a temporary storage. A master node orchestrates that for redundant copies of input data, only one is processed.

Shuffle step: Worker nodes redistribute data based on the output keys such that all data belonging to one key is located on the same worker node.

Reduce step: Worker nodes now process each group of output data, per key, in parallel. MapReduce allows for distributed processing of the map and reduction operations. Provided that each mapping operation is independent of the others, all maps can be performed in parallel – though in practice this is limited by the number of independent data sources and/or the number of CPUs near each source. Similarly, a set of 'reducers' can perform the reduction phase, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative. While this process can often appear inefficient compared to algorithms that are more sequential, MapReduce can be applied to significantly larger datasets than "commodity" servers can handle – a large server farm can use MapReduce to sort a petabyte of data in only a few hours.

2.1 A Mapreduce Programming Model Drives From Three Fundamental Phases:

1. Map phase: partition into M Map function (Mapper); each Mapper runs in parallel. The outputs of Map phase are intermediate key and value pairs.
2. Shuffle and Sort phase: the output of each Mapper is partitioned by hashing the output key. In this phase, the number of partitions is equal to the number of reducers; all key and value pairs in shuffle phase share the same key that belongs to the same partition. After partitioning the Map output, each partition is stored by a key to merge all values for that key.
3. Reduce phase: partition into R Reduce function (Reducer); each Reducer also runs in parallel and processes different intermediate keys.

3. MAPREDUCE IN CLOUD COMPUTING

Cloud Computing refers to both the applications delivered as services over the Internet, the hardware and the system software in the datacenters that provide those services [10]. Cloud platform, or platform as a service, refers to provide a computer platform or software stack as a service[11]. Developers by using the cloud computing paradigm are enabled to perform parallel data processing in a distributed environment with affordable cost and reasonable time. Thus, the advantage of data processing using cloud computing is the capability to easily do parallel and distributed computing on large clusters.

It is a fact that many cloud computing based computational processes will handle large datasets in a fully distributed environment in both a wired and wireless computer networking and communication environment. By using cloud computing, we enable to store, collect, share and transfer large amounts of data at very high speeds in a seamless and transparent manner that would out of sheer necessity classify all data to be “totally virtual”. Hence, all data in cloud computing captures the concept of data virtualization through a new programming paradigm or model which treats all data as a single entity through a process called MapReduce. MapReduce is a popular computing framework that is widely used for big data processing in cloud platforms. Cloud computing as a distributed computing paradigm, provides an environment to perform large-scale data processing. It enables massive data analytics on available computer nodes by using MapReduce platform. MapReduce and its open-source implementation Hadoop, allow developers to process terabytes of data that take hours to finish while hiding the complexity of parallel execution across hundreds of servers in a cloud environment. The main reason of using MapReduce with cloud computing is a key point of MapReduce that hides how parallel programming work away from the developer.

A Major web company, Amazon web services platform

offers a service called Amazon Elastic MapReduce to store and process massive datasets by running MapReduce system on Amazon cloud. It utilizes a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). EC2 is a web service platform that provides resizable compute capacity in a cloud [12]. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web [12].

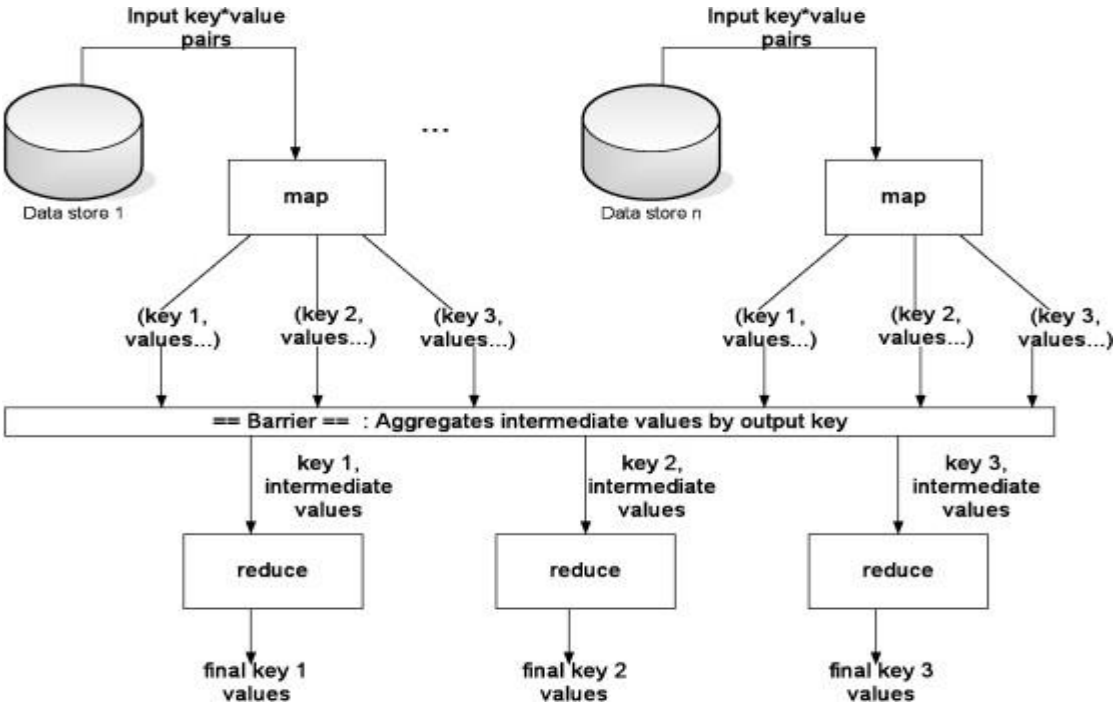

Figure 1: Overview of MapReduce

4. MAPREDUCE REQUIREMENTS

4.1 Fundamental And Specific Requirements

MapReduce platform is a distributed runtime engine for managing, scheduling and running MapReduce systems in a cluster of servers across an open distributed file system. To develop a MapReduce system based on the proposed framework, fundamental and specific requirements have been captured. Table 1 lists the summary of fundamental requirements for MapReduce system. Table 2 lists the summary of specific requirements. Both requirements essentially must be met to make a MapReduce system for large-scale processing more efficient.

Table 1.Fundamental Requirements.

	Fundamental Requirements	**Description**
	Scalability	To scale petabytes of data on thousands of machines.
	Parallelism	All tasks must run in parallel.
	Distributed Data	MapReduce distributes a data file to all nodes across a cluster to execute the application.
4	Cost Efficiency	ford to buy cheaper hardware and pay less for eration, especially if the size of dataset is too big.

Table 2.Specific Requirements.

No	Specific Requirements	Description
1	Availability	Many machine nodes and servers should be available in a computing cluster in failure mode.
2	Reliability	Multiple copies of data should be automatically stored in case of failure.
3	Flexibility	The system should enable to analyse and process various kinds of structured and unstructured data.
4	Security	Before running the system, user authentication is required.
5	Usability	The system should be developed as a service for running arbitrary code.
6	Locality	The system should divide tasks based on location of input file; each part is 64 MB same size of Google File System.
7	Data Consistency	The system should support coordination of data changes and it helps to provide consistency of data to ensure correctness of the execution and result.
8	Trust	When all nodes faithfully execute their tasks, the result is accurate and can trust the result.

5. SURVEY OF EXISTING METHODS

A. Reducing Costs of Spot Instances via Check pointing in the Amazon Elastic Compute Cloud

The Spot instances in the Amazon Elastic Compute Cloud (EC2) through check pointing were introduced that offers low cost. Mechanisms and tools that are used under this scheme are of great value for users looking for to reduce their costs while preserving high reliability. The check pointing mechanism was used to diminish the cost and instability of resource provisioning. Using real price hints of Amazon's spot instances, various dynamic check pointing strategies are analysed that can adapt to the current instance price and show their benefit compared to static, cost-ignorant strategies. The several adaptive check pointing schemes were compared with the proposed method in terms of economic costs and development of job completion times. Some of the Check pointing Schemes considered here are an Hour-boundary Check pointing, rising edge-driven Check pointing, Check pointing with Adaptive Decision and Check pointing Combinations. Simulation results show that our proposed approach decrease significantly both price and the task completion times.

Profiling Network Performance for Multi-Tier Data Centre Applications

There are several problems in the performance of Network and they are difficult to analyse. This is magnified when applications are often split into multiple tiers of components spread over large number of servers in a data centre. Problems often arise in the communication between the tiers, where either the application or the network or both of it could be to fault. In this paper, a scalable network-application profiler (SNAP) was presented that monitor the developers to identify and fix the performance problems. SNAP collects TCP statistics and socket-call logs with less calculation and storage overhead, and correlates across shared resources like host, link, switch and connections to locate the place of the problem. The profiler quickly identifies the right location, the right layer (application, network stack, or network), at the right time. Our one- week placement of SNAP in a production data centre had already helped developers uncover 15 major performance problems in application software, the network stack on the server, and the underlying network.

B. Bringing Elastic MapReduce to Scientific Clouds

The MapReduce was a programming model that was proposed by Google who off ers a simple and efficient way to perform distributed computation over large data sets. The Apache Hadoop framework was a free

and open-source implementation of MapReduce. To simplify the usage of Hadoop, Amazon Web Services provides Elastic MapReduce, a web service that enables users to submit MapReduce jobs. Elastic MapReduce takes care of resource provisioning, Hadoop configuration and performance tuning, data staging, fault tolerance, etc. This service drastically reduces the entry barrier to perform MapReduce computations in the cloud. However, Elastic MapReduce was limited to using Amazon EC2 resources, and requires an extra fee. In this paper, our work towards creating an implementation of Elastic MapReduce was presented which is able to use resources from other clouds than Amazon EC2, such as scientific clouds. This work will also serve as a foundation for more advanced experiments, such as performing MapReduce computations over multiple distributed clouds.

C. Evaluating MapReduce for Multi-core and Multiprocessor Systems

The MapReduce as a programming environment for shared-memory systems. They described Phoenix, an implementation of MapReduce that uses shared memory in order to minimize the overheads of task spawning and data communication. With Phoenix, the programmer provides a simple, functional expression of the algorithm and leaves parallelization and scheduling to the runtime system. They showed that Phoenix leads to scalable performance for both multi-core chips and conventional symmetric multiprocessors. Phoenix automatically handles key scheduling decisions during parallel execution. It can also recover from transient and permanent errors in Map and Reduce tasks. They compared the performance of Phoenix to that of parallel code written directly in P-threads.

D. Optimizing MapReduce for Multicore Architectures

The behaviour of MapReduce on commodity multicore processors, and proposes the Metis library. The paper's main insight was that the organization of the intermediate values produced by Map invocations and consumed by Reduce invocations is central to achieving good performance on multicore processors. Metis stores these intermediate values using an efficient data structure consisting of a hash table per Map thread with a b+ tree in each hash entry. As a result, Metis can achieve better performance than Phoenix on MapReduce applications that interact with the library frequently (e.g., applications with many keys). They have found Metis useful in practice, using it on our 16-core computers for counting and sorting gigabytes of data generated as part of another research project.

6. CONCLUSION

CMR is most important method to develop a processing frameworks using cloud services and also used to implement the MapReduce programming model. Cloud MapReduce had high scalability in the number of computing nodes. Our results show that the CMR is a practical system and its performance is higher than Hadoop. MapReduce is an easy, effective and flexible tool for large-scale fault tolerant data analysis. It has proven to be a useful abstraction that allows programmers to develop easily high performance system for running on cloud platforms and to distribute the processing over as many processors as possible. We also determined the fundamental and specific requirements to develop a framework. Cloud MapReduce has some highly necessary properties that can be shared by other highly-scalable systems. The network bandwidth is a rare resource and the numbers of optimizations in our systems are targeted at dropping the amount of data passed across the network. The redundant execution can be used to decrease the effect of slow machines, and to handle machine fault and loss of data.

REFERENCES

[1] Sangho Yi and Derrick Kondo, Artur Andrzejak, 2010, Reducing Costs of Spot Instances via Check pointing in the Amazon Elastic Compute Cloud, IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 236 – 243

[2] Minlan Yu, Albert Greenberg, Dave Maltz, Jennifer Rexford,Lihua Yuan,Srikanth Kandula,Changhoon Kim., 2011, Profiling Network Performance for Multi-Tier Data Centre Applications, Proceedings of the 8th USENIX conference on networked system design and implementation, pp. 57-70

[3] Pierre Riteau, Kate Keahey and Christine Morin., 2011, Bringing Elastic MapReduce to Scientific Clouds, 3rd Annual Workshop on Cloud Computing and Its Applications: Poster Session.

[4] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, Christos Kozyrakis, 2007, Evaluating MapReduce for Multi-core and Multiprocessor Systems, IEEE 13th International Symposium on High performace computer Architecture.

[5] Yandong Mao Robert Morris M. Frans Kaashoek, 2013, Optimizing MapReduce for Multicore Architectures,ELSEVIER International Conference on Computational Science, pp.2587–2590