S.Balachandar *et al*, International Journal of Computer Science and Mobile Applications,
Vol.2 Issue. 12, December- 2014, pg. 47-52 **ISSN: 2321-8363**

A SURVEY ON MAPREDUCE IN CLOUD COMPUTING

Dr.M.Newlin Rajkumar[1], S.Balachandar[2], Dr.V.Venkatesakumar[3], T.Mahadevan[4]

[1] *Asst. Prof, Dept. of CSE ,Anna University Regional Centre, Coimbatore, newlin_rajkumar@yahoo.co.in*
[2] *PG Scholar, Dept. of CSE, Anna University Regional Centre, Coimbatore, balachandar.sri@gmail.com*
[3] *Asst. Prof, Dept. of CSE, Anna University Regional Centre, Coimbatore, mail2venkatesa@gmail.com*
[4] *PG Scholar, Dept. of CSE, Anna University Regional Centre, Coimbatore, tmahadevan1991@gmail.com*

Abstract

Cloud Computing delivers computing resources so that it was appealing great thoughtfulness. Map Reduce is a programming model that be linked with the implementation for treating and creating large data sets. Since there is many drawbacks such as low scalability, does not support edible pricing and stream data processing etc. So to overcome the all drawback of Map Reduce framework, Cloud Map Reduces (CMR) is proposed. The experimental results of CMR show that it is more efficient and development faster than other implementations of the MR method. Also CMR 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.

MapReduce is a programming model and an linked implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a Map () procedure that performs filtering and sorting and a Reduce () procedure that performs a summary operation. The "MapReduce System" orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

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 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.

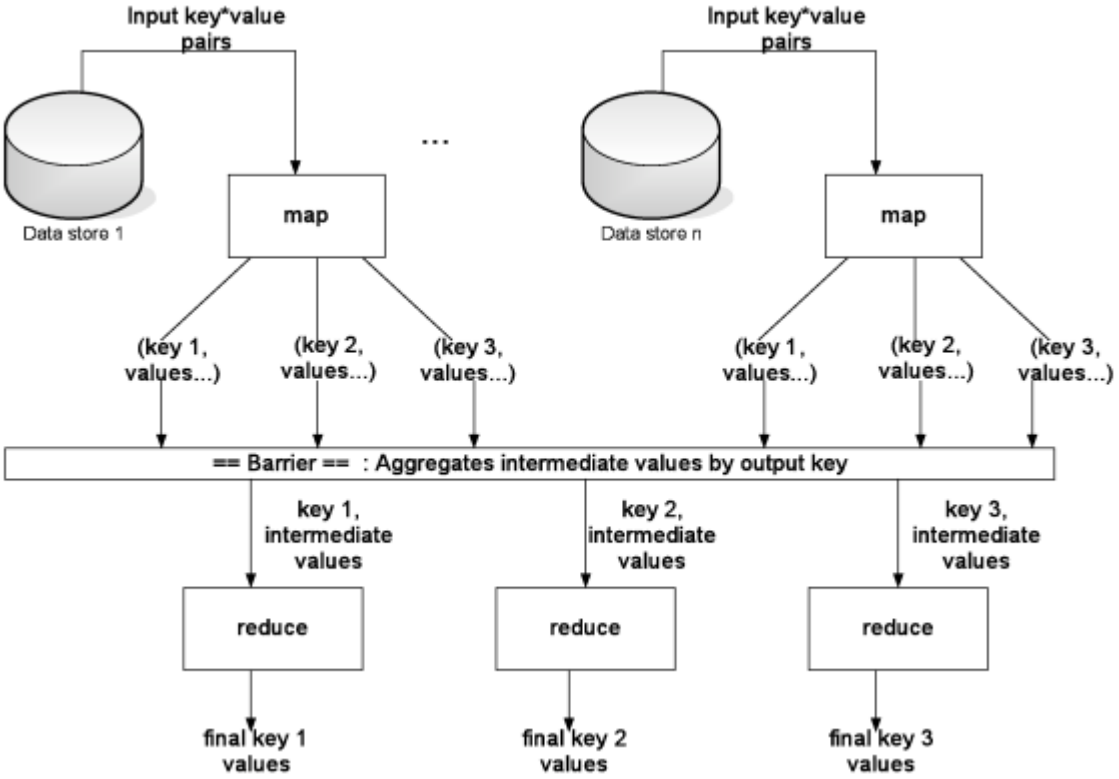

Figure 1: Overview of MapReduce

3. 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.

B. 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.

C. Bringing Elastic MapReduce to Scientific Clouds

The MapReduce was a programming model that was proposed by Google who offers 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.

D. 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.

E. 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.

4. 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. Cloud MapReduce has some highly necessary properties that can be shared by other highly-scalable systems. MapReduce proven to be useful thought and also simplifies the large scale computation easily. Restricting the programming model makes it easy to parallelize and distribute calculations and to make such calculations fault-tolerant. 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, 3r*d 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, IE*EE 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

Author Biography

M.Newlin Rajkumar is presently working as Assistant Professor in The Department of Computer Science and Engineering, Anna University Regional Centre, Coimbatore. He received his Bachelor of Engineering Degree from Bharathiyar University, Master of Science (M.S by Research) from National Chiao Tuns University, Taiwan and Master of Business Administration (IBM) from Anna University, Coimbatore. Presently he is pursuing his Ph.D. in Anna University Chennai. He has more than ten years of Teaching Experience. He has published several papers in reputed International Journals. He is a Professional Member of ACM, Member of IEEE India Council, Life Member of International Association of Computer Science and Information Technology, International Association of Engineers and in many International Associations. His research interest includes cloud Computing, Internet of Things, Big Data Analytics, Network Security, Security Protocols and Network Management.

S.Balachandar is pursing M.E Computer Science and Engineering in the Department of Computer Science and Engineering, Anna University Regional Centre, Coimbatore. He received his MCA from Anna University Chennai. His research interests includes Cloud Computing, BigData and Network Security

V.Venkatesakumar is presently working as Assistant Professor in The Department of Computer Science and Engineering, Anna University Regional Centre, Coimbatore. He received his Bachelor of Engineering Degree from Bharathiyar University, Master of Engineering Degree and Ph.D from Anna University Chennai. He has more than ten years of Teaching Experience. He has published many papers in reputed International Journals and has Chaired many Conferences. He is a Life Member of International Association of Computer Science and Information Technology, International Association of Engineers and in many International Associations. His research interest includes Cloud Computing, Internet of Things, Big Data Analytics, Operating System, Software Engineering and Web Technologies

T. Mahadevan is pursing M.E Computer Science and Engineering (Specialization with Networks) in the Department of Computer Science and Engineering, Anna University Regional Centre, Coimbatore. He received his B.E Computer Science and Engineering from Anna University Chennai. His research interests includes Cloud Computing, Wireless Sensor Network and Network Security

