Enhancement of Continuous Cloud Map Reduce via Splitting, shuffling and Spot with Snap shot Process

V. Devi
PG Student, University College of Engineering, konam

M. Muthu Selvi
Assistant Professor, University College of Engineering, Konam

Abstract – In the existing process of continuous cloud map reduce in that there used for enhancement using splitting process and shuffling process for the enhancement. In Cloud's spot instances is deal with massive machine terminations caused by process termination. In this process to avoid termination the Snapshot technique will used. If there used the proposed snapshot technique means the total process time of each file must calculated in that the process the Snapshot functionality has been included in C-CMR in order to provide the user with a means to view the results of a Map-Reduce job run before the completion of the job. For example, for a 20 min. long job, the user could request for a snapshot of the job results at time T=15min. It can visualize the output corresponding to the result of the Map and Reduce operations performed on the data that has arrived at the input from the start of the job till T=15min. This allows the user to get an idea of the results beforehand, instead of waiting for the entire job to be completed. Also, for batch processing, if no more input data is detected, a "snapshot" is automatically invoked, writing the results to S3. The implementation involves pushing a "flag" file in S3 when a user requests a snapshot. The snapshot thread monitors the S3 location to detect the availability of this file. When the file is detected, the thread deletes the file, reads the output queue to gather results generated, and writes the results to the S3 bucket specified as the output location. This file can be downloaded by the user to read the snapshot of the result whenever required. As the functionality is implemented as a separate thread, it does not interfere with the normal operation of the MapReduce job, and occurs concurrently with it. This also leads to network and processor parallelism as the data fetch and store stages occur concurrently with the processing stages, thus reducing bottlenecks.

Index Terms – Cloud, Splitting, Shuffling, C-CMR.

1. INTRODUCTION

JEffrey Dean and Sanjay Ghemawat [1] defined MapReduce(MR) as a programming model and an associated implementation for processing and generating large data sets. Cloud MapReduce (CMR) is an implementation of MapReduceframework on Amazon Web Services [2][3]. By using queues, CMR easily parallelizes the Map and the Shuffling stages. By using Amazon's visibility timeout mechanism, it implements fault-tolerance. CMR is a fully distributed architecture with no single point of failure and scalability bottleneck as it exploits Amazon Web Service's fully distributed features. Beyond using Amazon Web Services to simplify the implementation, this architecture is novel in many aspects as compared with the master/slave Hadoop [4]. It is faster than Hadoop and is very simple as well (3000 LoC, compared toHadoop's nearly 300K LoC).

2. RELATED WORK

CMR, by itself, is not well suited for spot market environments to cope with massive machine terminations caused by spot price fluctuations when using spot instances on Amazon EC2.

- Spot Cloud Map Reduce
- C-CMR via Splitting and shuffling

2.1 Spot Cloud Map Reduce

CMR, by itself, is not well suited for spot market environments to cope with massive machine terminations caused by spot price fluctuations when using spot instances on Amazon EC2. If Map Reduce jobs are running on spot instances, and these instances are turned off and on due to fluctuations in prices of spot instances, then it leads to increase in the job completion time to a very great extent. The existing CMR architecture can be enhanced to tackle this issue of massive termination of spot instances.

2.2 C-CMR via Splitting and shuffling Splitting

In which the input data is split into chunks and distributed across multiple nodes to be processed upon by a user defined function. Multiple chunks so that it can be processed by multiple Map workers simultaneously. A reference to each chunk is kept in the input queues, which will be picked by one of the splitter to process. Input queue is an instance of SQS provided by Amazon.

Shuffling

By using queues, CMR easily parallelizes the Map and the Shuffling stages. CMR uses the network to transfer intermediate key- value pairs as soon as they are available, thus it overlaps data shuffling with Map processing.

Overlapping shuffling is used when pipelining MapReduce. Compared to the implementation to pairwise socket connections and buffering/copying mechanism, the implementation using queues is much simpler.

3. PORPOSED MODELLING

This work describes and execution of Cloud Map Reduce by Amazon Web Services. We start with the high level architecture, and then discuss implementation issues. CMR is a scalable, flexible, fast implementation of the Map Reduce structure that allows programmers to use the repayment of organization big information dispensation jobs on an obscure plat-form. It provides soaring facts throughput as facts comes beginning several servers and connections with the servers potentially all cross dissimilar system path. It is storage for the internet. Amazon S3 provides a simple web interface that can be used to store and retrieve any amount of data. This is used in CMR to provide input data to process before starting a CMR job.

This is a particularly presented and flexible non-relational in order store, which offloads the service of verification administrator. Customer has an ability to provide combiner occupation, which is parallel to decrease function. The clothes which affect combiner occupation on the Mappers' production are called Combiners. Combiners are frequently used to implement record side pre-aggregation which reduces the size of system shift required among Map and Reduce phases. A recently planned and implemented planning of the system, modeled after the CMR framework, incorporating changes necessary for dispensation streaming data and incremental online aggregation. Work describes and completion of an original advance designed for optimizing and attractive CMR by cylinder among plan and decrease phases.

Spot Cloud Map Reduce (Spot CMR), a Map Reduce execution modified for a spot promote situation. To the best of our knowledge, it is the primary Map Reduce execution that might accept huge node terminations induced by the price variation in a spot market. Toward conquer the mark throughput control of an only easy DB area; every employee arbitrarily picks one of some domains to mark the position. During classify construct a major scheme on top of AWS, a original totally spread planning, specifically CMR, to execute the Map Reduce training form. CMR is a significant advance to upward meting out frameworks by cloud services.

4. RESULTS AND DISCUSSIONS

In this section all the results and the discussions should be made.

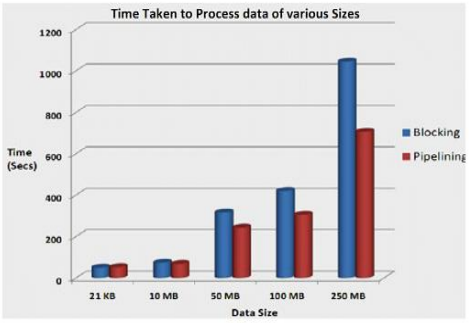

Figure 1 Resultant Graph of the Proposed System

5 CONCLUSION

Work succeeded in extra improving on the labor of CMR to offer further functionality and improved production by employing pipelining among the Map and Reduce phases. In C-CMR also included maintain for stream information processing, snapshots and cascaded MR jobs in CMR. This is a significant progress in the situation of present facts processing desires, as evidenced by new flow processing applications. The progressing Window functionality to have been moderately implemented in C-CMR needs to be during totally valuable with a parallel, concurrent module for aggregating the result formed by the Mappers according to the time-window provided by the user. This will suggest true suppleness to the client to observation the Mapper production of some specified time-window of the put in flow.

REFERENCES

[1] Dean J. and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in OSDI, 2004, pp. 137–150.
[2] Guo C., G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," in Proc. 2009 SIGCOMM, pp. 63–74.
[3] Karve R., D. Dahiphale, and A. Chhajer, "Optimizing cloud mapreduce for processing stream data using pipelining," in EMS, 2011, pp. 344–349.
[4] Mysore R. N., A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A.Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in Proc. 2009 SIGCOMM, pp. 39–50.
[5] Vasudevan V., A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained tcp retransmissions for datacenter communication," in Proc. 2009 SIGCOMM, pp. 303–314.
[6] Vasudevan V., A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained tcp retransmissions for datacenter communication," in Proc. 2009 SIGCOMM, pp. 303–314.
[7] H. Liu and D. Orban, "Cloud mapreduce: a mapreduce implementation on top of a cloud operating system," in Proc. 2011 IEEE International Symposium on Cluster Computing and the Grid, vol. 0, pp. 464–474.
[8] Yang, S. Kamata, and A. Ahrary, "NIR: content based image retrieval on cloud computing," in Proc. 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, vol. 3, pp. 556–559.