Dynamic Data Partitioning & Virtual Chunking for Effective Data Retrieval using Big Data & Cloud

Ms. S Aminta Sabatini[1] Dr. N DuraiPandian[2]
[1]M. E. Student [2]Principal
[1,2]Department of Computer Science & Engineering
[1,2]Velammal Engineering College, Chennai, India

Abstract— Today's science is generating significantly larger volume of data than before, data compression can potentially improve the application performance. In some high performance computing system data compression could ameloirate the I/O pressure. A new concept called as virtual chunking is introduced that aims to better employ the compression algorithms in parallel. ABE based keys are distributed among the corresponding admin so that access policy is applied. Data is splitted and stored using dynamic virtualization concept. A small number of references is appended to the end of file. The data's are partitioned into three layers of connectivity. The details of the user will be stored in the database. Once the User login to an account, and request the Job from the Service Provider the results would be displayed based on the user's query. Based on the request, the Service Provider will process the job and respond to them. Using map reduce technique the data's are classified according to the query. Time frame is being generated at each level of connectivity. This process will reduce the time for the fetching any results from the table.

Key words: ABE, Virtual Chunking, Dynamic Virtualization, Map Reduce

I. INTRODUCTION

Big data is an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process using traditional data processing applications. The challenges include analysis, capture, duration, search, sharing, storage, transfer, visualization, and privacy violations. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data. So we can implement big data in our project because every employ has instructed information so that we can make analysis on this data. Cloud is the delivery of on-demand computing resources everything from applications to data centers over the Internet on a pay-for-use basis. Cloud computing is a type of Internet -based computing that provides shared computer processing resources and data to computers and other devices on demand. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers. Hadoop is an open-source Apache Software Foundation project written in Java that enables the distributed processing of large datasets across clusters of commodity. Hadoop has two primary components, namely, HDFS and Map Reduce programming framework. HDFS is a distributed file system designed to run on top of the local file systems of the cluster nodes and store extremely large files suitable for streaming data access .HDFS consists of two types of nodes, namely, a name node called master and several data nodes called slaves. HDFS can also include secondary name nodes. The name node manages the hierarchy of file systems and director namespace (i.e., metadata). File systems are presented in a form of name node that registers attributes, such as access time, modification, permission, and disk space quotas. The file content is split into large blocks, and each block of the file is independently replicated across data nodes for redundancy and to periodically send a report of all existing blocks to the name node. The MapReduce system is managed by two daemons they are Job Tracker and the Task Tracker Job Tracker coordinates the entire job execution. The job tracker does the bookkeeping of all the task which is running on a cluster. Task tracker runs the individual tasks of map and reduce .One map task is created for each input split and also the number of reduce tasks is configurable. The Hadoop cluster divides the layer into two parts they are the map reduce layer and the HDFS layer. The name node and the data node will come under the multi node cluster. Once they are divided it is easy to fetch the results within the time frame.

II. LITERATURE SURVEY

Enhancing Throughput of HDFS Zhao[1] proposed the performance of the HDFS decreases when handling interaction intensive file. The cause of throughput degradation issue is analysed when accessing interaction-intensive files and presents an enhanced HDFS architecture along with an associated storage allocation algorithm which overcomes the performance degradation problem. The paper analyzes the cause of throughput degradation issue when accessing interaction-intensive files and presents an enhanced HDFS architecture along with an associated storage allocation algorithm that overcomes the performance degradation problem. In order to overcome the issue for interaction-intensive tasks, three task are to be performed they are improve metadata structure to provide faster I/O with less overhead extending the name node with a hierarchical structure and to design a storage allocation algorithm to improve data accessibility. PSO-based algorithm is used to find a near optimal storage allocation plan for the incoming file.

A. Supporting the Data Intensive Application

Zhao D[2] proposed the FusionFS technique, every I/O needs to be transferred via the network between the compute and storage resources. A distributed storage layer algorithm has been proposed which is local to the compute nodes. This layer is responsible for the I/O operations and saves extreme amounts of data movement between compute and storage resource. The compute nodes and storage nodes are interconnected by a shared network infrastructure. The design and implementation of the distributed metadata management in FusionFS can be improved with the

namespace, datastructures, network protocol, consistency and persistence. FusionFS is crafted to support extremely intensive metadata operations and is optimized for file writes.

B. Integrating with Large Scale Data

Bicer. T[3]proposed the compression methodology, i.e. computing cycles in high performance systems are increasing faster than both storage and wide area bandwidths. To improve the performance of large-scale data analytics applications, compression has become promising approach. In order to improve the performance of large scale data analytics a new compression methodology, is been developed which exploits the similarities between spatial and temporal neighbors in a popular climate simulation dataset and enables high compression ratios and low decompression costs. Secondly a framework is developed which can be used to incorporate a variety of compression and decompression algorithms. The framework supports a simple API to allow the integration with the existing application. Once a compression algorithm is implemented, the framework automatically mechanizes multithread retrieval and multithread data compression, this will also automatically parallelizes I/O and compression operations, and overlaps these with computation.

C. Data Provenance for Intensive Computing

Shou C proposed the FusionFS and SPADE which is the key issue in evaluating the feasibility of data provenance is its performance, overheads, and scalability. The design optimality is been experimentally done whether provenance metadata should be loosely-coupled or tightly integrated with a file metadata storage systems. FusionFS, implements a distributed file metadata management based on distributed hash tables, SPADE will use a graph database to store audited provenance data and provides distributed module for querying provenance. FusionFS + SPADE are a promising prototype with negligible provenance overhead and has promise to scale to petascale and beyond.

D. Map Reduce under Resource Contention and Task Failure

Zhang R proposed a map reduce technique which is a widely used programming model for large scale data processing. When there are many concurrent IO operations the degradation of the throughput is measured using the curve fitting method inorder to obtain the parameters easily. M/M/1 queuing model is used to describe task failures in a cluster at the angle of probability. In the MapReduce shuffle phase, each reduce task will pull data segments from all map tasks so that the number of network connections increases sharply. MapReduce framework, a job is composed of multiple tasks. A job will be successfully finished until all the tasks are completed. A Task will attempt to execute for multiple rounds and each round is called a Task Attempt.

E. Map Reduce in Heterogeneous Cloud Environment

Cherkasova L proposed a heterogeneity model for Hadoop clusters which is used for clustering and job management. The author has explored the efficiency and performance accuracy of the bounds-based performance model for predicting the MapReduce job completion in heterogeneous Hadoop clusters. Their performance is implicitly reflected in the job profile and it is effectively incorporated in predicting the job completion times. MapReduce jobs are executed across multiple machines, the map stage is partitioned into map tasks and the reduce stage is partitioned into reduce tasks. In the map stage, each map task reads a split of the input data, applies the user-defined map function, and generates the intermediate set of key/value pairs. In the reduce stage, each reduce task fetches its partition of intermediate key/value pairs from all the map tasks and merges the data with the same key. In the reduce stage, each reduce task fetches its partition of intermediate keys pairs from all the map tasks and merges the data with the same key.

F. Optimal Resource Provisioning in Public Cloud

Tian F proposed a technique for optimizing the resource provisioning, with the deployment of web applications, scientific computing, and sensor networks, a large amount of data can be collected from the users MapReduce. Hadoop cluster has different requirements when compared with the public clouds and private cloud. First, for each job normally a Hadoop cluster will be started on a number of virtual nodes to take advantage of the ‘‘pay-as-you-use’’ economical cloud model. Typically, such a on-demand cluster is created for a specific long-running job, where no multi-user or multi-job resource competition happens within the cluster. Next it is the user’s responsibility to set the appropriate number of virtual nodes for the Hadoop cluster. For optimizing resource provisioning, MapReduce programs involves two intertwined factors: the monetary cost of provisioning the virtual machine nodes and the time cost to finish the job. Using these factors the map reduce jobs can be completed.

G. SLO Driven Right Sizing

Li T proposed a novel framework for SLO-driven resource provisioning and sizing of a map reduce jobs. There is an increasing number of MapReduce applications, such as the personalized advertising, spam detection, real-time event log analysis, which is to be completed within a given time window. First, an automated profiling tool is proposed that extracts a compact job profile from the past application or by executing it on a smaller data set. Then, by applying a linear regression technique, the scaling factor is been derived to project the application performance when processing a larger dataset. By completing the above factors, fast and efficient capacity planning model is produced for a MapReduce job with timing requirements and it generates a set of resource provisioning options.

H. I/O For Petascale Computing Systems

Debains C proposed a new petascale computing system, the Big data the gap between the storage performance and an application’s I/O requirement is increasing. The author has presented a novel I/O-aware batch scheduling framework to coordinate ongoing I/O requests on petascale computing systems. The batch scheduler can handle both system state and jobs activities and can also control the jobs status during their execution. The job is treated as periodical subjobs we also have designed two types of I/O-aware scheduling policies have been designed they are conservative policies and the adaptive policies. FCFS policy will work in similar way as traditional job scheduler does. It chooses jobs in

chronological order based on the start time of concurrent I/O requests. The MaxUtil policy aims at maximizing the system utilzation under the storage bandwidth constraint.

I. Map Reduce Framework on Graphics Processors

Govindaraju k proposed a framework on graphics processors. MapReduce is a distributed programming framework originally proposed by Google for the ease of development of web search applications on a large number of CPU. MapReduceis a distributed programming framework originally proposed by Google for the ease of development of web search applications on a large number of CPUs. With the GPU-based framework, the developer writes their code using the simple and familiar MapReduce interfaces. The runtime on the GPU is completely hidden from the developer by our framework.

III. METHODOLOGY

A. Map Reduce Technique

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples.

During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster. The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes. Most of the computing takes place on nodes with data on local disks that reduces the network traffic. After completion of the given tasks, the cluster collects and reduces the data to form an appropriate and sends it back to the Hadoop server

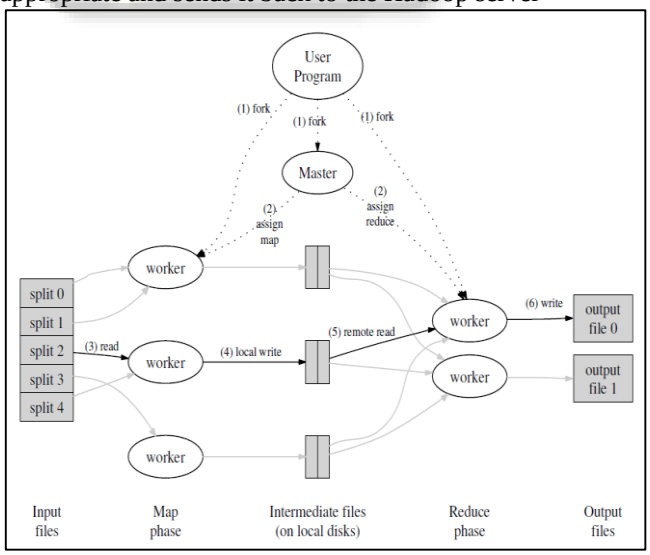

Fig. 1: Map Reduce Processing

IV. CONCLUSION

The main aim of this survey is to split the wholesome datas into virtual chunks. Virtual chunks are logical blocks pointed by the references without breaking the physical continuity of the file content. VC avoids the potentially huge space overhead of conventional physical chunking in data compression and supports an efficient random access to the compressed data. In essence, VC achieves the small special overhead of file-level compression as well as the small computational overhead of the chunk-level decompression. The future work of this project is to allot the time frame for each layers.

REFERENCES

[1] Zhao D., Burlingame K., Debains C, Alvarez-Tabio.P, and Raicu.I,(2013) 'enhancing the performance of hdfs' Proc. IEEE Int. Conf. Cluster Comput.,
[2] Zhao D, K. Qiao K, and I. Raicu I, 'Supporting data intensive computing,' Int. J. Big Data Intell., Feb. 2015
[3] Bicer T., Yin J., Chiu D., Agrawal G., and Schuchardt K. (2013), 'Integrating online compression to accelerate large-scale data analytics applications' in Proc. IEEE 27th Int. Symp. Parallel Distrib. Process. pp. 125–1216
[4] Zhao D, K. Qiao K, and I. Raicu I, 'Supporting data intensive computing,' Int. J. Big Data Intell., Feb. 2015
[5] Cui X. Lin C. Hu R,. Zhang R, and Wang C, (2013)[5]'Modelling the performance of MapReduce under resource contentions and task failures,' in Proc. IEEE 5th Int. Conf. Cloud Computi. Technol. Sci., vol. 1, pp. 158–163
[6] Zhang Z, Cherkasova L, and Loo B.T, 'Performance modelling of MapReduce jobs in heterogeneous cloud environments,' in Proc. IEEE 6th Int. Conf. Cloud Comput., 2013, pp. 839–846
[7] Tian F and Chen K,(2011) 'Towards optimal resource provisioning for running MapReduce programs in public clouds,' in Proc. IEEE 4th Int. Conf. Cloud Comput., pp. 155–162.
[8] Li. T, Zhou X., Brandstatter K., Zhao D., Wang K., Rajendran A., Zhang Z., and Raicu.I,(2013) 'SLO driven right sizing,' in Proc. IEEE Int. Symp. Parallel Distrib. Process. pp. 775–787.
[9] Debains., Powers J., Guo S., and Tian F,(2014) 'I/O for petascale ,' IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 6, pp. 1403– 1412
[10] Fang W,Luo Q, and Wang 'A map reduce framework on graphics processors,' in Proc.17th Int. Cloud Comp.pp 672-679

