World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 3.419

VIRTUAL MACHINE BASED DATA SAMPLING APPROACH TO IMPROVE QUERY PERFORMANCE FOR VIRTULIZED HADOOP

Anupama S[1]*, Kavya G[2], Kannika J S[3], Arjun T R[4] and Malatesh S H[5]

[1234]B. E, Student of M.S.E.C, Bengaluru, India.

[5]HOD (BE, MTech., PhD), Department of Computer science and Engineering, M.S.E.C, Bengaluru, India.

Article Received on 14/03/2016 **Article Revised on 04/04/2016** **Article Accepted on 24/04/2016**

***Corresponding Author**
Anupama S.
B.E, Student of M.S.E.C, Bengaluru, India.

ABSTRACT

MapReduce emerges as an important distributed programming paradigm for large-scale data analysis applications. As an open-source implementation of MapReduce, Hadoop presents an attractive usage system for many enterprises. There are some drawbacks in a traditional Hadoop cluster deployed with a large scale of physical machines, such as burdensome cluster management and fluctuating resource utilization. Virtualized Hadoop cluster not only simplifies cluster management, but also facilitates cost-effective workload consolidation for resource utilization. In Hadoop system, the data locality and query performance are the critical factors impacting on performance of MapReduce applications.

KEYWORDS: MapReduce, Hadoop system.

I. INTRODUCTION

Big data is a huge collection of data in the form of volume variety and velocity, for processing and storing big data the traditional tools are not required so new tool came called hadoop. Hadoop system allows enterprises to analyze large-scale data like terabytes and pet bytes more conveniently than ever.

Hadoop consists of two components HDFS, Mapreduce.

HDFS: used for data storing.

Mapreduce: used for processing of data which is stored in HDFS.

In companies like yahoo, google, facebook came across huge amount of data. google uses google file system for storing data and processed by mapreduce after that they produce white paper for further study of google file system and mapreduce.

Dowg cutting accepted and invented or modified HDFS and Mapreduce. Later hadoop became an open source.

II. SYSTEM IMPLEMENTATION

Implememtation of the project include:

a. DispersalArchitecture

b. DataSampling

c. Sqoop Transfer

a. Dispersal Architecture

A virtualized Hadoop cluster is built with dispersal architecture which deploys computing node and storage node in same VMs. This deployment model overcomes those disadvantages of traditional manner. On the one hand, it has strong scalability, which allows of respectively fluctuating amount of computing nodes or storage nodes. On the other hand, live migration is very agile when a computing node is migrated without considering any other storage node VMs.

Here, we build virtualized Hadoop cluster with dispersal architecture, and its deployment model can be demonstrated by Fig. 1. In this figure, different kinds of arrows indicate different kinds of data locality. Using dispersal architecture, there is not node locality, but server locality can be acted as node locality which is equivalent in traditional deployment manner. To improve data locality in virtualized Hadoop cluster, we can live migrate computing node VM to physical server running storage node VM stored a data replication needed by that computing node.

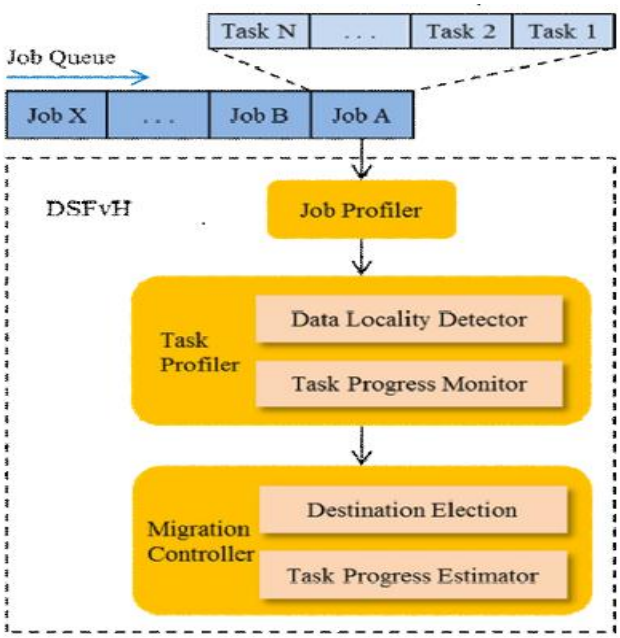

Fig. Architecture of DSFvH

a. JobProfiler: The main function of Job Profiler module is analyzing the job submitted to DSFvH from Job Queue, parsing out the tasks included in this job, as well as input data blocks for each task. In Hadoop system, the Job Tracker is the service that farms out MapReduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack.

b. Task Profiler: The Task Profiler module is responsible for detecting the data locality of all tasks, and mainly monitoring tasks without server locality. It will identify the stragglers which run notably slower than other similar tasks. This module consists of two sub-modules: Data Locality Detector and Task Progress Monitor.

c. Data Locality Detector: It detects the data locality types of all tasks included in this job, according to distribution of input data replicas and deployment location of computing node VM.

b. Data Sampling

Apache Hive is an SQL-like software used with Hadoop to give users the capability of performing SQL-like queries on it is own language, HiveQL, quickly and efficiently. It also gives users additional query and analytical abilities not available on traditional SQL structures.

With Apache Hive, users can use HiveQL or traditional MapReduce systems, depending on individual needs and preferences. Hive is particularly ideal for analyzing large datasets (pet bytes) and also includes a variety of storage options.

Hive is full of unique tools that allow users to quickly and efficiently perform data queries and analysis. In order to make full use of all these tools, it's important for users to use best practices for Hive implementation.

Data sampling has two layers.

a. Data Partitioning
b. Data Bucketing
c. Data bucketing with Partitioning

a. Data Partitioning

Hive partitioning is an effective method to improve the query performance on larger tables. Partitioning allows you to store data in separate sub-directories under table location. It greatly helps the queries which are queried upon the partition key(s). Although the selection of partition key is always a sensitive decision, it should always be a low cardinal attribute, e.g. if your data is associated with time dimension, then date could be a good partition key. Similarly, if data has association with location, like a country or state, then it's a good idea to have hierarchical partitions like country/state.

b. Data Bucketing

Bucketing improves the join performance if the bucket key and join keys are common. Bucketing in Hive distributes the data in different buckets based on the hash results on the bucket key. It also reduces the I/O scans during the join process if the process is happening on the same keys (columns).

Additionally it's important to ensure the bucketing flag is set (**SET hive.enforce. bucketing=true;**) every time before writing data to the bucketed table. To leverage the bucketing in the joint operation we should **SET hive.optimize.bucketmapjoin=true**. This setting hints to Hive to do bucket level join during the map stage join. It also reduces the scan cycles to find a particular key because bucketing ensures that the key is present in a certain bucket.

Hashing: On the basis of hash function value the particular data of a table will be inserted into particular bucket.

Formula: Hash Function (bucketed column) mod No. of buckets.

Data Bucketing with Partition: Embedding both Partitioning and bucketing within a single table also be know as data skewing.

c. Sqoop Transfer

1. Sqoop Import: Sqoop is designed to import tables from a database into Hive.

2. Sqoop Export: Sqoop is designed to import tables from a Hive into Database

III. SYSTEM REQUIREMENTS

Functional requirements

A description of the facility or feature required. Functional requirements deal with what the system should do or provide for users. They include description of the required functions, outlines of associated reports or online queries, and details of data to be held in the system

- automated creation of a Cloudera Hadoop environment
- addition of VM's to an existing Hadoop environment
- deletion of VM's from an existing Hadoop environment
- deletion of an entire Hadoop environment
- variability in the number of machines created in the setting up process.
- automated load balancing for physical machines
- define the environment name
- check redundancy on environment names
- provide an interface for managing the features
- provide an interface to interact with the Hadoop scheduler
- Installing and running Hive Queries
- Installing and running Sqoop Commands

Non-functional requirements

Non-functional requirements define the overall qualities or attributes of the resulting system. Non-functional requirements place restrictions on the product being developed, the development process, and specify external constraints that the product must meet. Examples of NFR include safety, security, usability, reliability and performance requirements

- web-oriented GUI
- develop the GUI with ZK
- develop the system in Java
- virtualization over Oracle virtual box

- run over in browser
- run over in command line
- intuitive navigation
- coherent information
- clear information
- error information
- extensible
- scalable
- service always available
- Run over i386 and x64
- Secure machines interaction

IV. CONCLUSION

In this Project to improve performance of MapReduce applications in virtualized Hadoop cluster, we argue that improved data locality for both Map and Reduce phases of the job is very necessary in such system. In this Project, we build virtualized Hadoop cluster with dispersal architecture which deploys computing node and storage node in respective VMs. We propose a novel task scheduling approach which aims to improve data locality in virtualized Hadoop cluster through migrating the computing node VM to the physical server running VM acted as storage node that holds a data replica needed by that computing node at appropriate moment. Meanwhile, we also included data sampling to achieve data portioning and data bucking increases the performance of a distributed Hadoop file system. This also increases the performance of MapReduce in the virtualized Hadoop Cluster systems.

V. FUTURE ENHANCEMENT

In our future work, we plan to improve the performance of our approach for machine learning MapReduce applications and investigate the feasibility of DSFvH with the next generation of Hadoop's compute platform.

VI. REFERENCE

1. Ron C. Chiang, H. Howie Huang, "TRACON: Interference-Aware Scheduling for Data-Intensive Applications in Virtualized Environments", in Proceedings of ACM International Conference for High Performance Computing, Net working, Storage and Analysis (SC), Seattle, USA, Nov. 2011; 1-12.

2. Ganesh Ananthanarayanan, Sameer Agarwal, Srikanth Kandula, Albert Greenberg, Ion Stoica, Duke Harlan, and Ed Harris, "Scarlett: Coping with Skewed Content Popularity in Map Reduce Clusters", in Proceedings of the 6th European Conference on Computer Systems (EuroSys), Salzburg, Austria, Apr. 2011; 287–300.
3. M. Zaharia, A. Konwinski, A. Joseph, R. Katz, and I. Stoica,"Improving Map Reduce Performance in Heterogeneous Environments", in Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI), San Diego, USA, Dec. 2008; 29-42.