ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 4, pp: (159-161), Month: October - December 2015, Available at: www.researchpublish.com

A Data Aware Caching (Dache) for Big-Data Applications Using the MapReduce Framework

Utkarsh Honey[1], Yogesh More[2], Prasad Wandhekar[3], Santosh Wayal[4], Prof. Jayashree Chaudhari[5]

[1, 2, 3, 4, 5] Dr. D Y Patil School of Engineering, Pune, India 411047

Abstract: **The big-data refers to the large-scale distributed data processing applications which works on exceptionally large amounts of data. Google's MapReduce and Apache's Hadoop, its open-source implementation, are the software systems for big-data applications. An observation of the MapReduce framework is that the framework generates a large amount of intermediate data. MapReduce is unable to utilize such data so they are thrown after used. We propose Dache, a data-aware cache framework used for big-data applications. In Dache, tasks submit their intermediate results to the cache manager and queries the cache manager before executing the actual computing work. A novel cache description system and a cache request and reply protocol are designed**

Keyword: **Big-data, MapReduce, Hadoop, caching.**

I. INTRODUCTION

Google MapReduce is a programming model and also a software framework for Big -scale distributed Computing on large amounts of data. Figure illustrates the high level work flow of a MapReduce Task. Application developers specify the computation in terms of reduce function and a map and the underlying MapReduce Task scheduling system automatically parallelizes the computation across the cluster of machines. While MapReduce obtain popularity for its simple programming interface and excellent Performance when implement a large spectrum of applications. Since most such applications take a huge amount of input data, they are called as "Big-data applications".

Input data is first divided and then given to workers in the map stage. Every Individual data items are called records. The MapReduce system parses the input splits to each worker and produces records. After the phase of map, intermediate results generated in the map phase are shuffled and sorted by the MapReduce system which are then given into the workers in the reduce phase. Final results are computed by multiple reducers and after it written back to the disk. Hadoop is an open-source implementation of the Google MapReduce programming model. Hadoop includes of the Hadoop Common, which provides access to the file systems supported by Hadoop. HDFS (Hadoop Distributed File System) provides distributed file storage and is optimized for large unchangeable blocks of data. A small Hadoop cluster will include a single master and multiple worker nodes known as slave. Whereas master node runs multiple processes, including a Task tracker and a Name Node. The Task tracker is having control for the management of running jobs in the Hadoop cluster. While the name node manages the HDFS, The Task tracker and the Name Node are primarily collocated on the same physical machine. The other servers in the cluster run a Task tracker for Data Node processes. A MapReduce job is divided into number of tasks. Tasks are managed by the Task tracker. The Task trackers and the Data Node are collated on the same servers to provide data locality. MapReduce provides a standardized framework for implementing the large-scale distributed computation, known as the big-data applications. Still, there is restriction of the system. i.e. The inefficiency in incremental processing which refers to the applications that incrementally grow up the input data and continuously apply computations on this input and produce output. There are potential duplicate computations and operations are performed in this process. As the MapReduce does not have the any technique which is to identify such duplicate computations and accelerate the job execution. While Motivating by this observation, in same paper we propose, a data-aware cache scheme for big data applications using MapReduce framework, which goals at extending the MapReduce framework and provisioning a cache layer which is used for efficiently identifying and accessing cache items in a MapReduce job.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 4, pp: (159-161), Month: October - December 2015, Available at: www.researchpublish.com

II. OBJECTIVE(S) AND SCOPE

2.1 The Scope Of The Work Can Be Extended To Following:

- Requires minimum change to the original MapReduce programming model.
- Application code only requires slight changes in order to utilize Dache.
- Implement Dache in Hadoop by extending the relevant components.
- Tested experiments show that it can easily eliminate all the duplicate tasks in incremental MapReduce jobs.
- Minimum execution time and CPU utilization

2.2 Problem Statement:

In current Hadoop MapReduce framework is that the framework generates a large flow of intermediate data. MapReduce is unable to save that such data so they are deleted after used. But in our system we introducing the cache memory that holds the intermediate results in it, because of that the data processing, means job executing processing is faster than old system, So that system is a time consuming, repetition of data processing are reduced.

2.3 Map Reduce Architecture:

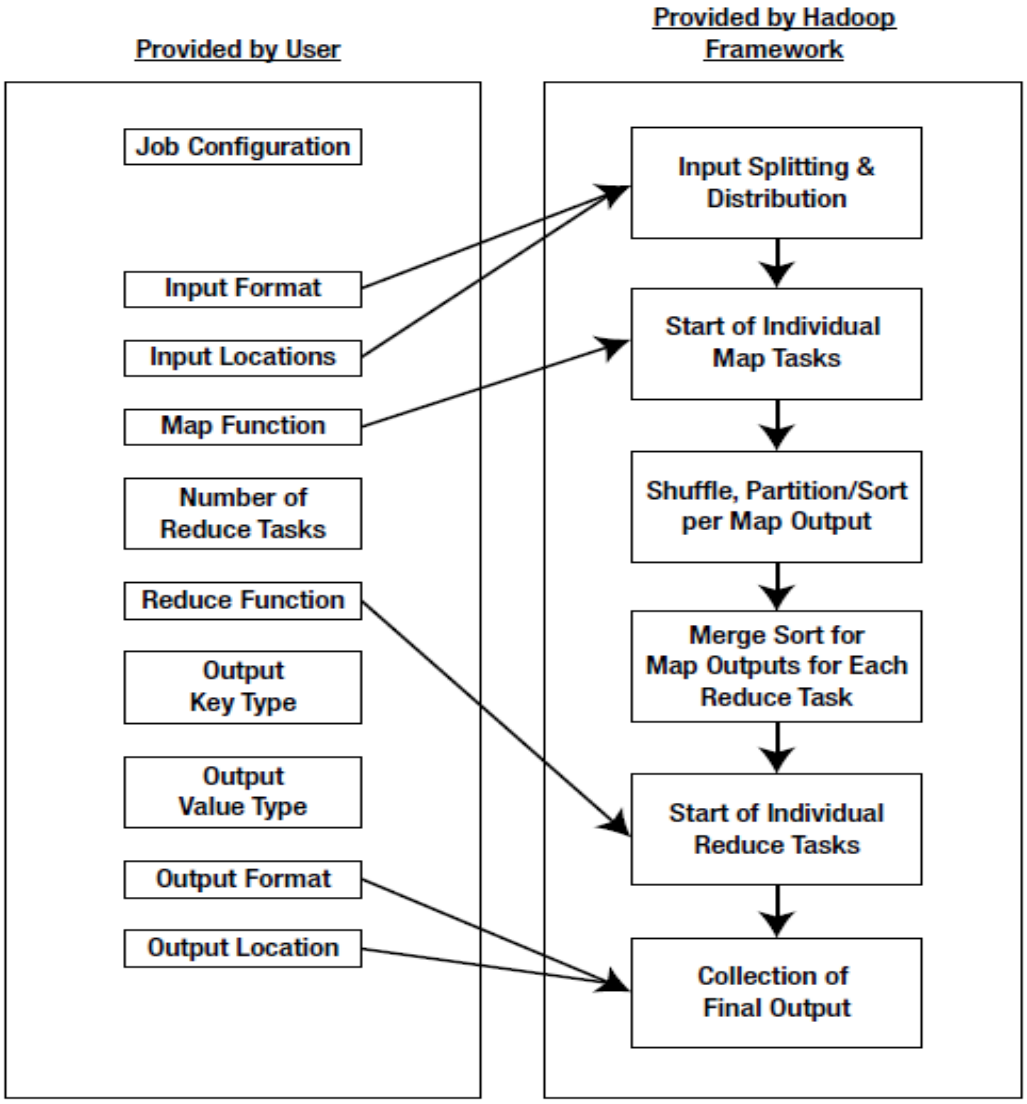

Fig 1: Map Reduce Architecture

2.3.1 Operations:

- Clients submit jobs to the Job Tracker.
- Job Tracker talks to the name node.
- Job Tracker creates execution plan.
- Job Tracker submits works to Task tracker.
- Task Trackers report progress via heart beats.
- Job Tracker manages the phases.
- Job Tracker updates the status.

2.4 Software & Hardware Requirement:

2.4.1 Software Requirements:

Operating System : WINDOWS XP & ABOVE

Database : My SQL,Hadoop

Language : java

2.4.2 Hardware Requirements:

System : Pentium Iv 2.4 GHz

Hard Disk : 40 GB

Monitor : 15 VGA Color

RAM : 512 Mb

III. CONCLUSIONS

This paper present the design and evaluation of a data aware cache framework that requires minimum changes to the original MapReduce programming model for provisioning the incremental processing for Big data applications using MapReduce model. In this paper we propose, a data-aware cache description scheme, architecture and protocol. In this Paper Presented method only requires a slight modification in the input format processing and task management of MapReduce framework. As a result, application code requires only slight changes in order to use Data in data aware caching. This paper implements Hadoop by extending relevant components. In the future, we propose to adapt our framework is to more general application scenarios and also implement the scheme in the Hadoop project.

REFERENCES

[1] J. Dean and S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communication of ACM, vol. 51, no. 1, pp. 107-113.

[2] Hadoop, http://www.Hadoop.apache.org.

[3] Cache algorithms, http://en.wikipedia.org/wiki/Cache algorithms.

[4] Java programming language, http://www.java.com/.

[5] Google compute engine, http://cloud.google.com/products/computeengine.html

[6] Jeffrey Dean and Sanjay Ghemawat, **MapReduce: Simplified Data Processing on Large Clusters** http://labs.google.com/papers/mapreduce.html

[7] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, **The Google File System** http://labs.google.com/papers/gfs.html