Survey of Caching Mechanism in Hadoop

Jalpa Shah[1], Hinal Somani[2]

[1] *P.G Student, Department of Information Technology, L.J.I.E.T, Ahmedabad, Gujarat, India*
[2] *Assistant Professor, PG Department, L.J.I.E.T, Ahmedabad, Gujarat, India*

ABSTRACT

Hadoop is open source project by apache which can be used as stand alone, single node and multi node cluster. Hadoop is develop in version 1x to 2x with map-reduce v2 and YARN by the Apache devlopers.Hadoop is mainly consist of HDFS and Map-Reduce, where HDFS is as storage for Hadoop and Map-Reduce is programming model for data file processing. Hadoop has a centralize cache management which is useful for repetitively accessed files. The distributed cache copies files to every node, then map or reduce reads the files from the local file system and make the data any time accessible any time for use as it is distributed to multiple location in the DataNode blocks of fixed sized.

Keyword: - *Hadoop, Cache, Big Data, Map-Reduce, HDFS*

1. INTRODUCTION

Big Data is collection of large and complex data sets, which are difficult to store, process, capture and analysis using the traditional database management system. Big data supports the 3V's: Volume, Velocity and Verity. Big Data is difficult to work with using most relational database management systems, desktop statistics and visualization packages since it requires massive parallel software running on tens, hundreds or even thousands of servers [6]. Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.

Hadoop has 2 main components: HDFS (Hadoop Distributed File System) and Map-Reduce. HDFS is a block structured distributed file system for storing large volumes of data. All files are divided in a fixed sized blocks. Map-Reduce are programming model meant for large clusters. It has a parallel computing framework helps in parallelization, fault tolerance, data distribution and load balancing. The computation of Map-Reduce takes a set of input key/value pairs and generates a set of output key/value pairs. The computation of generating the set of output key/value pairs is divided into two functions: Map function and Reduce function. Fig-1; show the basic architecture of Hadoop.

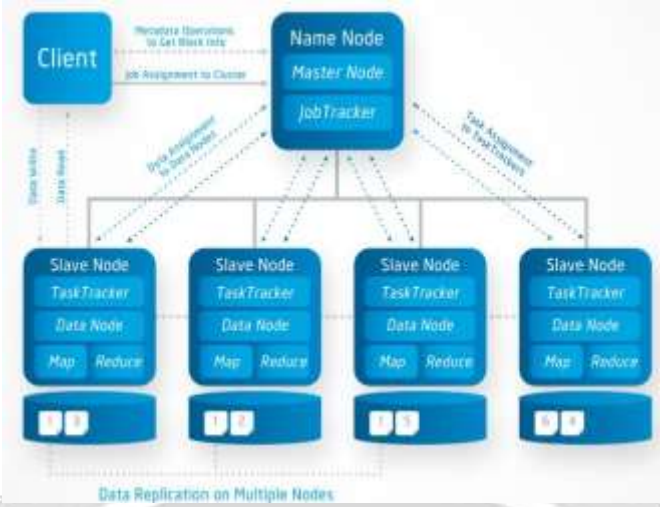

Fig-1 Hadoop Model [7]

2. RELATED WORK

2.1 Jing Zhang, Gongqing Wu, Xuegang Hu, Xindong Wu, "A Distributed Cache for Hadoop Distributed File System in Real-time Cloud Services"

In this paper, author design and implemented distributed layered cache system built on HDFS and name it HDFS based Distributed Cache system (HDCache). This system consists of a client library and multiple cache services. The cache services are designed with three access layers an in-memory cache, a snapshot of the local disk, and the actual disk view as provided by HDFS [1]. The HDCache system can be deployed on HDFS NameNode, DataNode and any other application systems. The cache system contains two parts: a client dynamic library (*libcache*) and a service daemon.

The *libcache* consists of two major components: *Communication & Control* and *Shared Memory Access*. The *Communication & Control* module undertakes the tasks of (1) interoperating with the HDCache service on the same host, (2) communicating with ZooKeeper servers remotely, and (3) calculating hash values of desired files and locating a specific cached file. HDCache is designed for write-once-read-many access model for files, which is approximately held in a cloud computing environment.

2.2 Yaxiong Zhao and Jie Wu, "Dache: A Data Aware Caching for Big-Data Applications Using The MapReduce Framework"

In this paper, data-aware cache framework for big-data applications (Dache) is proposed. Dache aims at extending the MapReduce framework and provisioning a cache layer for efficiently identifying and accessing cache items in a MapReduce job. In Dache, tasks submit their intermediate results to the cache manager. A task, before initiating its execution, queries the cache manager for potential matched processing results, which could accelerate its execution or even completely saves the execution [2].

Experiment results of Dache shows that it improves the completion time of Map-Reduce jobs and saves a significant chunk of CPU execution time.

2.3 Debajyoti Mukhopadhyay , Chetan Agrawal , Devesh Maru , Pooja Yedale , Pranav Gadekar, "Addressing NameNode Scalability Issue in Hadoop Distributed File System using Cache Approach"

The goal of this paper is to provide a solution to reduce the use of NameNode memory space so that the problem of NameNode memory getting full which ultimately results in hindering the scalability of cluster will be delayed [3]. Cache is basically use for storing the frequently used data closer to the processor than the whole data and using it for faster operation can be applied in the NameNode memory management. In this separation algorithm is used which will be used for moving the least recently used metadata from NameNode to secondary storage device. In this

algorithm one extra field count is added which indicated the frequency of use about each file into the cluster. Each time the access time is updated the count is increased by one indicating that the data/file represented by it is been used. In the proposed architecture, a threshold value will be defined on the NameNode memory space. As and when this threshold value will be reached the separation algorithm will run so as to remove the metadata records that are not being used. Initially the NameNode will keep storing the file system metadata as it comes until the predefined threshold is reached. Once the threshold will be reached the separation algorithm will come into play and remove some of the records based on the separation function.

2.4 Mr, Sanjeev G Kanbargi, Mr. Sunil Kumar S, "Cache Utilization for Enhancing Analyzation of Big-Data &Increasing the Performance of Hadoop"

The proposed system creates a novel cache, which stores the intermediate data or mapper's output into a novel cache. Whenever the system needs to analyze same Big-data set, it fetches already processed data from novel cache rather than running mapper function on whole Big-data set again [4]. In hadoop framework job is given to mapper, which generates <key,value> pair as intermediate data and this data is give to reducer function to narrow down to required data. The <key,value> pair or intermediate data can be same for different types of jobs.

Optimal page replacement (OPR) algorithm, which will be very efficient while handling multiple jobs on same type of Big data is used. The performance of the cache management mechanism is measured as ratio of hits and misses. In this cache management approach the upcoming jobs are scanned. If present job's required data is in cache memory then no operation will be performed on cache. If job's required data is not present in cache, then compare each element from cache to the upcoming jobs. The job's data which is not in upcoming jobs can be replaced with the new job.

2.5 Jaewon Kwak, Eunji Hwang, Tae-kyung Yoo, Beomseok Nam, and Young-ri Choi, "In-memory Caching Orchestration for Hadoop"

When a client requests to cache a specific file, the cache request message is transmitted to the central NameNode which knows where the partitioned data blocks for the file are located. The NameNode piggybacks the cache request message on a heartbeat response and delivers it to DataNodes where the data blocks are stored. The DataNodes store the data blocks in their OS page caches and periodically send the NameNode *cache reports* that describe what data blocks are cached in their *in-memory caches*.

Limitations in the current HDFS in-memory caching implementation is that users should manually specify what files should be cached and uncached. For an input file of each application that users want to cache, they must use a command to add the file to the in-memory cache.

Cache affinity (CA) metric, which indicates how much performance improvement it can get using in-memory caching, as follows:

$$CA = MAX(1 - \frac{R_{CL-hot}}{R_{no-cache}}, 0)$$

Where, Rno−cache and RCL−hot are the runtimes of the application with no cache, & CL for hot cache, respectively. Adaptive Cache Local Scheduling Algorithm computes the number of times to skip for achieving the cache locality for a MapReduce job dynamically based on its percentage of cached input data. Cache Affinity Aware Cache Replacement Algorithm basically replaces data locks of applications with low cache affinity with those with high cache affinity.

3. COMPARATIVE ANALYSIS

Table 1: Literature Comparison

Sr No	Paper Title	Method	Advantage	Limitation
1	A Distributed Cache for Hadoop Distributed File System in Real-time Cloud Services	HDCache	Uses Shared memory and come with internet cloud computing	Used for intranet environment outside firewall
2	Dache: A Data Aware Caching for Big-Data Applications Using The MapReduce Framework	Map Reduce Frame work	Eliminate duplicate task from MapReduce Job	Cache life time management by fixed storage or optimal Utility
3	Addressing NameNode Scalability Issue in Hadoop Distributed File System using Cache Approach	Separation Algorithm	NameNode scalability	Stores only more frequently used data for faster operation
4	Cache Utilization for Enhancing Analyzation of Big-Data &Increasing the Performance of Hadoop	OPR Algorithm	Best with Historical data	Need to update cache for real time dataset
5	In-memory Caching Orchestration for Hadoop	Adaptive cache local scheduling algorithm	Reduce scheduling overhead, performance improvement	Users should have to manually specify which file to cache and un cache.

4. CONCLUSIONS

Hadoop is distributed database architecture used for large data set which are frequently access for the analysis purpose so the data processing need to be faster. Hadoop has in memory cache manager which helps to manage centralized cache for fast processing of data from various nodes. Map-reduce help in managing the data with its mapper and reducer functions which store the recent processed data in cache for future use.

5. REFERENCES

[1]. Jing Zhang, Gongqing Wu, Xuegang Hu, Xindong Wu, "**A Distributed Cache for Hadoop Distributed File System in Real-time Cloud Services**", 2012 ACM/IEEE 13th International Conference on Grid Computing, DOI 10.1109/Grid.2012.17, 1550-5510 © 2012 IEEE, page 12-21

[2]. Yaxiong Zhao and Jie Wu , "**Dache: A Data Aware Caching for Big-Data Applications Using The MapReduce Framework**", 2013 Proceedings IEEE INFOCOM , 978-1-4673-5946-7 ©2013 IEEE

[3]. Debajyoti Mukhopadhyay , Chetan Agrawal , Devesh Maru , Pooja Yedale , Pranav Gadekar, "**Addressing NameNode Scalability Issue in Hadoop Distributed File System using Cache Approach**", 2014 International Conference on Information Technology, DOI 10.1109/ICIT.2014.18, 978-1-4799-8084-0 © 2014 IEEE

[4]. Mr, Sanjeev G Kanbargi, Mr. Sunil Kumar S , "Cache Utilization for Enhancing Analyzation of Big-Data &Increasing the Performance of Hadoop", 978-1-4673-6667-0/15©2015 IEEE

[5]. Jaewon Kwak, Eunji Hwang, Tae-kyung Yoo, Beomseok Nam, and Young-ri Choi, "**In-memory Caching Orchestration for Hadoop**", 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, DOI 10.1109/CCGrid.2016.73, 978-1-5090-2453-7 © 2016 IEEE

[6]. Pradeep Adluru, Srikari Sindhoori Datla, Xiaowen Zhang, "**Hadoop Eco System for Big Data Security and Privacy** ", 978-1-4577-1343-9 ©2015 IEEE

[7]. http://www.rosebt.com/blog/category/hadoop%20technology%20stack/2 (25-11-2016)