Minimize Execution latency in Hadoop using SVM

Jalpa Shah[1], Saket swandeep[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 one of such type of database which stores the data in distributed manner in the cloud system. Data processing is done in terms of Jobs where client send data file to Master node which is consider as job and Master node assigns that jobs to slave node, but there may be currently no slave node is free or available to process the job at that time Master node checks its resources queue for next available node which can be used to assign job as the slave node gets frees. In Conventional approach, there is gap in case of effective caching of jobs data; also the best slave node for effective caching is not predicted in conventional approach which can be incorporated with the help of SVM algorithm in this work.

Keyword: - *Hadoop, Map Reduce, SVM, Effective data caching*

1. INTRODUCTION

Hadoop:
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.

Support Vector machine (SVM):
SVM is used for classification and regression analysis on the data set which is part of machine learning algorithm. SVM can also be applied in the case of regression. SVM regression performs linear regression in the high-dimension feature space. In Linear SVM, the data in input as (x, y) pair is used for mapping in hyper plan with value on X-axis as x and Y-axis as on y. Linear SVM works on the kernel which is get the best possible solution near to hyper plan for the best result from the input set passed.
Using SVM we can classify the slave nodes in better condition and node which gets fail in multiple times within bounded time line.

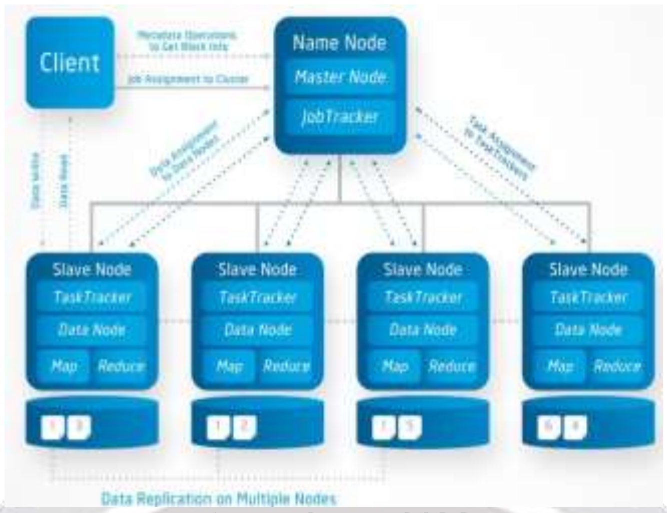
Client
Metadata Operations to Get Block Info
Job Assignment to Cluster
Name Node
Master Node
JobTracker
Data Write
Data Read
Data Assignment to Data Nodes
Task Assignment to TaskTrackers
Slave Node
TaskTracker
Data Node
Map
Reduce
Data Replication on Multiple Nodes

Fig-1: Hadoop Model [7]

2. RELATED WORK

In [1], distributed layered cache system built on HDFS and name it HDFS based Distributed Cache system (HDCache) is designed. This system consists of a client library and multiple cache services. The cache services are designed with 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, Data Node and any other application systems. Aa client dynamic library (*libcache*) and a service daemon are 2 parts of cache system.
Communication & Control and *Shared Memory Access are 2 components of libcache*. 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.

In [2], 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].
Improved completion time of Map-Reduce jobs and saves a significant chunk of CPU execution time are the experimental result of the Dache system. .

In [3], 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 is used which is very efficient while handling multiple jobs on same type of Big data. 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.

In [4], when a client requests to cache a specific file, the cache request message is transmitted to the central Name Node 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. PROBLEM STATEMENT AND PROPOSED WORK

Problem statement

In hadoop conventional approach, there is gap in case of effective caching of jobs data. Also the best slave node for effective caching is not predicted in conventional approach.

Proposed Work

In this as research element a data node prediction model is introduced. This data node prediction model work with the linear SVM regression algorithm which helps in identifying the best fit data nodes for execution of client jobs. This model also helps in caching the data for job at best fit data node such the job execution can be goes smooth and faster.

Proposed work model

Fig.2 shows the logical representation of the proposed work.

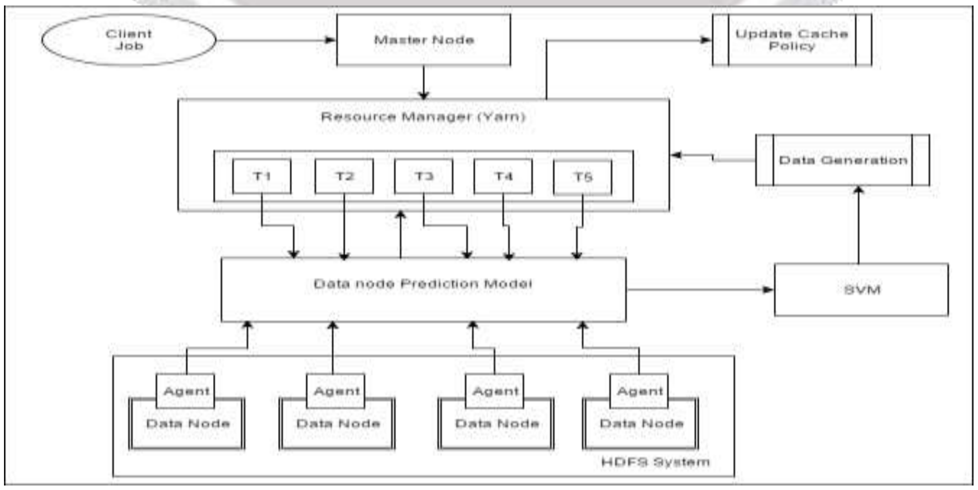

Fig- 2: Proposed model

Work Flow steps

1. Firstly in hadoop master node (M) allocates jobs (j1 to jn) to resources manager (Y).
2. Resources manager (Y) splits these jobs (j1 to jn) into small chunks called tasks.
3. Continuous monitoring of data node (D1 to Dn) through agent.
4. Agent at regular interval of time sends data to resource manager.
5. Data is stored for further processing and prediction mechanism.
6. Data is further processed to find optimum data node to allocate job.
7. Further profiling is done over data and classifier is applied with the help of svm.
8. Accordingly resources manager allocated job to optimum node.
9. This process is repeated at regular intervals to update the cache policy at the Resource Manager (Y).

4. EXPERIMENTAL RESULT

Fig.3 show the experimental results for the proposed work over the convention hadoop system. As shown in the figure as the data size increase execution time for processing data is decrease as compare to the convention hadoop system.

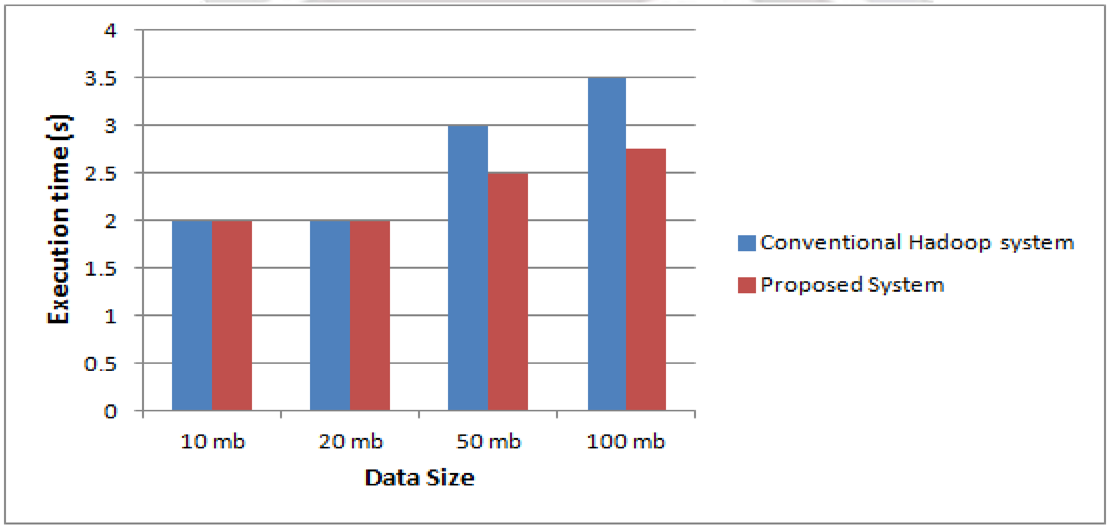

Fig- 3: Result

5. CONCLUSIONS

Hadoop is most widely used distributed data processing framework for the big data application. As Big Data means the huge amount of data to be processed in limited time period to get faster result, by implementing proposed work we have improved the hadoop execution latency or minimize the latency in job execution. Proposed system uses the SVM machine learning algorithm. Also proposed system can be implemented with the other machine learning algorithm.

6. 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]. 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
[4]. 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
[5]. Pradipsinh K. Chavda, Prof. Jitendra S. Dhobi “**Web users Browsing Behavior Prediction by Implementing Support Vector Machines in MapReduce using Cloud Based Hadoop**”, 2015 5th Nirma University International Conference on Engineering (NUiCONE), 978-1-4799-9991-0/15 ©2015 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]. 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
[8]. http://www.rosebt.com/blog/category/hadoop%20technology%20stack/2 (25-11-2016)