ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2016

MAPReduce based Cache Management for Big Data Framework

Kiran[1], Surender Singh[2]

M.Tech Scholar, Dept. of Computer Science and Engineering, OITM, Hisar, India[1]

Assistant Professor, Dept. of Computer Science and Engineering, OITM, Hisar India[2]

ABSTRACT: Large scale Big data processing consumes a huge volume of computational power and it is necessary to reserve the critical resources like main memory/Cache for successful process execution. Researchers have developed various solutions for Big data applications but each scheme solves a specific issue. This paper introduces a process management scheme that offers fair utilization of Cache memory. It supports two different states i.e. Idle State and Active State and in Idle State, it does not consume memory and finally it can be utilize by other processes. We also performed comparison with the existing scheme and results show the effectiveness of the proposed scheme.

KEYWORDS: Big Data, cache, memory, resource management, MAPReduce

I. INTRODUCTION

MapReduce offers a fast track process execution method for distributed environment. It performs two different tasks MAP and Reduce. MAP task process a block of data and coverts it into key & value pairs which are further reduce to small elements by Reduce task. Reduce process follows the MAP process. Following are the different stages for MAPReduce method [12]:

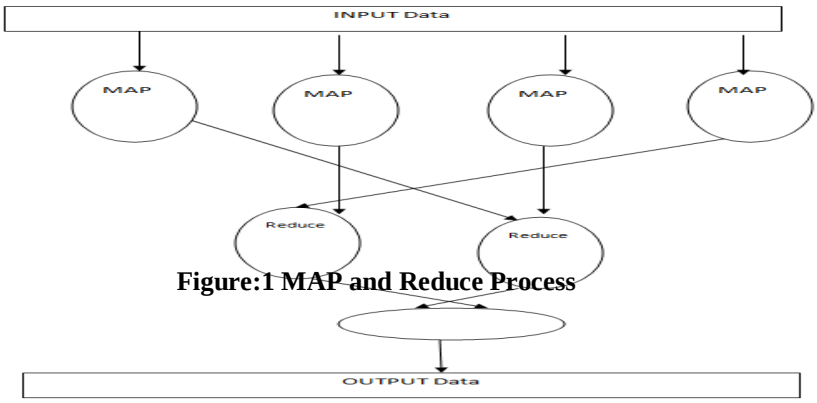

Figure:1 MAP and Reduce Process

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2016

MAP stage: at this stage, MAP processes input data and output is stored at HDFS Reduce Stage: in this stage, reducer takes the output of MAP process and after processing this data is saved on HDFS.

Input data

	Input	**Output**
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Table-1 Input/Output [13]

Example of Input Data

1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

Figure:2 Input Data

1981	34
1984	40
1985	45

Figure:3 Output Data

Following are the steps to communicate with MAPReduce Job:

OPTION(S)	**Description**
-submit <job-file>	Submits a job.
-status <job-id>	Show the map and reduce completion status
-counter <job-id> <group-name> <countername>	Shows the counter value.
-kill <job-id>	Terminates the current job.
-events <job-id> <fromevent-#> <#-of-events>	Displays the jobtracker details
-history [all] <jobOutputDir> - history < jobOutputDir>	Shows the log of all executed processes
-list[all]	Shows all jobs in a processing Queue
-kill-task <task-id>	Terminates a specific task
-fail-task <task-id>	Log of incomplete tasks
-set-priority <job-id> <priority>	Assigns job priorities

Table-3 MApReduce commands

II. LITERATURE SURVEY

H. TAKASAKI et al. [1] explored the various Hadoop applications and developed a solution which can optimize the memory utilization and conserve the energy also. Proposed scheme uses two different modes i.e. in thread mode, it uses multiple thread for MAP process and in normal mode, it does not use the thread mode. Experimental results show that context switching between these modes can enhance the performance of task scheduler. Proposed work can be extended to provide the support for multiple Hadoop clusters.

H. Aksu et al. [2] proposed a cache aware solution using Apache Hbase framework. Graph portioning and use of different cache policies improves the cache hit ratio. For experiment, Flicker and Twitter databases were used and results show that it can perform in distributed and parallel execution environment.

Zhao-Rong Lai et al. [3] considered the issues related to load balancing, job scheduling and response time of Mapreduce. They analyzed the job scheduling problem on the basis of linear programming and used a polynomial-time

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2016

approximation method for online job arrival to schedule the tasks. Simulation results show its performance in terms of satisfaction ratio using real time systems. Current research work can be extended to provide the support for medical image processing.

Rong Gu, Furao Shen et al. [4] proposed a modified parallel computing framework, known as cNeural which can use large scale training samples for fast training. It uses HBase for large scale training dataset storage and parallel loading. It also uses a parallel in-memory computing framework for fast iterative training. Experimental results show its performance in terms of efficiency as compared to Hadoop with dozens of machines using millions of samples. Current work can be extended to resolve large scale neural network issues by introducing some more neural network based algorithms.

Yaxiong Zhao et al. [5] investigated a cache framework that requires minimum change to the original MapReduce programming model for provisioning incremental processing for Bigdata applications using the MapReduce model. Dache, a data-aware cache description scheme, protocol and architecture. This method requires only a slight modification in the input format processing and task management of the MapReduce framework. As a result, application code only requires slight changes in order to utilize Dache. Analysis shows that it can identify the replicated tasks in incremental MapReduce jobs and does not require substantial changes to the application code.

Xiao Yu et al. [6] explored the Hadoop framework and developed a solution for cache management. Proposed solution uses a cache aware binary input scheduler which can be used by multiple Hadoop applications. Experimental results show that it can reduce the number of attempts required to access remote data and also optimizes the process execution time. Proposed scheme can be extended to provide the support of multiple schedulers.

S.D. Yoon et al. [7] developed a local caching solution for parallel applications which can shuffle the data between local file system and HDFS by managing the buffers at multiple locations. It uses priority queues for data management and a dispatcher is used to keep the track of data in specific queues. A controller monitors the number of hit/miss and estimates the required in a queue. Data is written in to queue on priorities basis and it is read out from local file system and search fails, then a miss is counted otherwise it is considered as a hit. Proposed scheme can be extended to develop an equation for workload.

Divya M et al. [8] investigated the workload attributes and the number of resources required for the same workload and designed a solution which can estimate the job attributes before assigning it to a scheduler. Results show that it performs better as compared to FIFO methods.

Tak-Lon Wu et al. [9] provided a fast cache solution for big data processing application by using the combination of Hadoop and PIG frameworks. They used use K-means clustering and Page Rank that has three components: a python control-flow script, a Pig data-transform script for a single iteration, and two K-means user-defined functions written with a Pig-provided Java interface. During each iteration, Loader in each Mapper loads the aggregated centroids into memory as vector objects from the distributed cache on disk before computing the Euclidean distances for data points in the Loader stage. Each loader outputs assigned centroids and data points as fields in a single bag, each field in bag is defined as string data type which further splits into tuples for matching Pig's GROUP operation to collect partial centroid vectors from mappers. It takes the average of all partitions, emits to a final centroids file and saves it to HDFS. Results show that proposed scheme is able to provide the fast cache response for Mapreduce.

D. Wei et al. [10] proposed a multi-granularity content tree model and pay-as-you-go mode to support evolvement data modeling. These features help to split the data model,, position data content precisely and to expand the dimensions of the main features that described according to the data subject, and then gradually discover data contained information and relationships among the information. Considering the large size of the data features, this paper designs data persistence mode based on HBase, so as to achieve the purpose of data processing by using technologies within the Hadoop system. Authors also presented data content extraction and content tree initial state algorithms under MapReduce framework, and dynamic loading and local caching algorithms of content tree, thus forming a basic extract-store-load process. This work can be extended for geosciences information and knowledge discovery using platform which is based on big data technology, and construct the value chain of geological data results; construct

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2016

geosciences information resource sharing and value-added demonstration services to create favorable conditions for information integration, resource sharing and knowledge innovation for the whole society

III. PROPOSED SCHEME

Proposed Scheme: In case of Big data processing, application deals with the large volume of data and it is necessary to manage the memory resources for smooth process execution

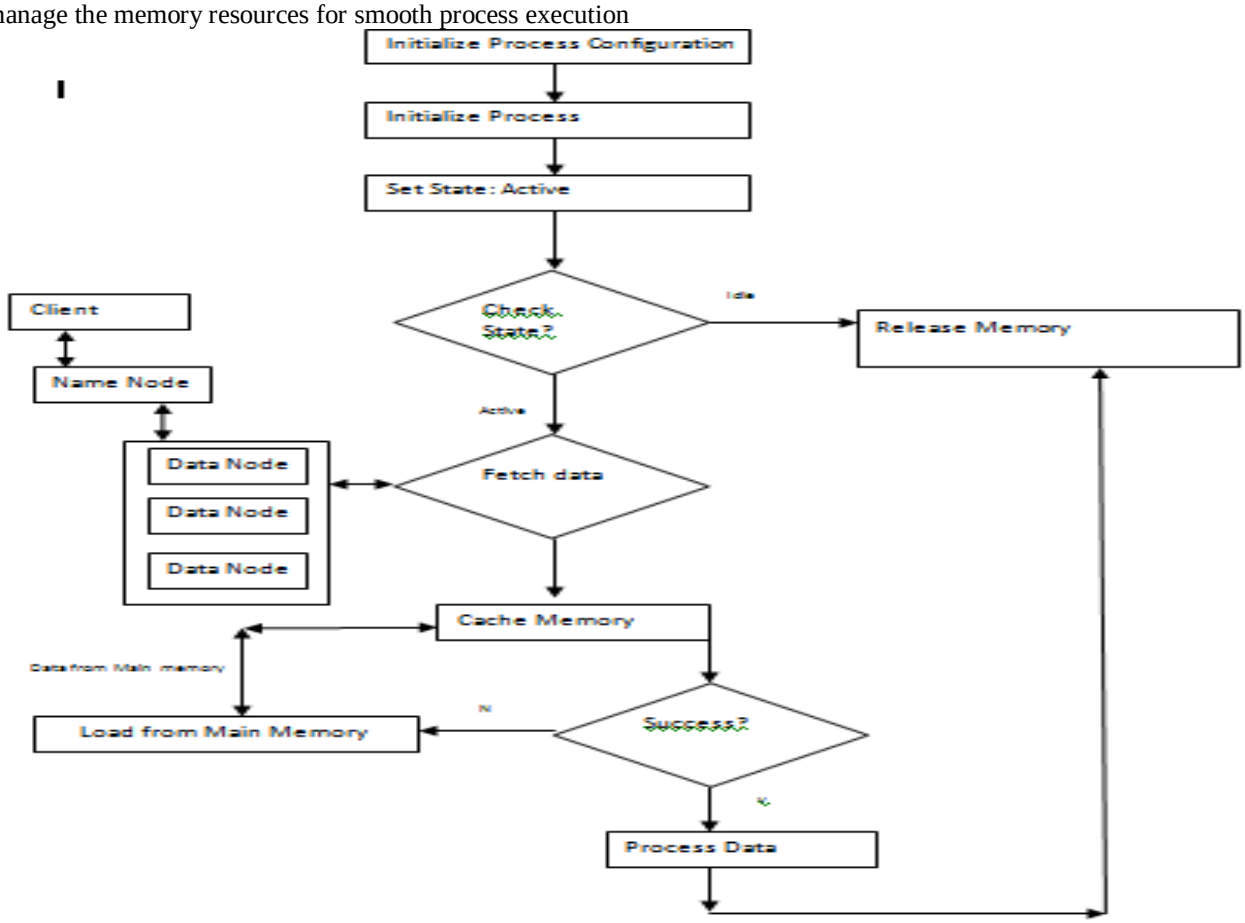

Figure 4: Proposed Scheme

Cache contains the frequent data to be utilize over HDFS. Caching mechanism allows clients to locate the *paths* to be cached by HDFS. The Name Node communicates with Data Node that has the required blocks on disk, and instructs it to cache the blocks in off-heap caches. Figure: above shows that process can switch the states i.e. Idle state to Active State. If it is in idle state, Data node cannot access the cache data and releases the hold over occupied memory blocks but if it is Active state, Data node can fetch the data from cache and if it is not available then it is loaded from main memory and forwarded to Name node ad finally it reaches to the end user.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2016

IV. PERFORMANCE ANALYSIS

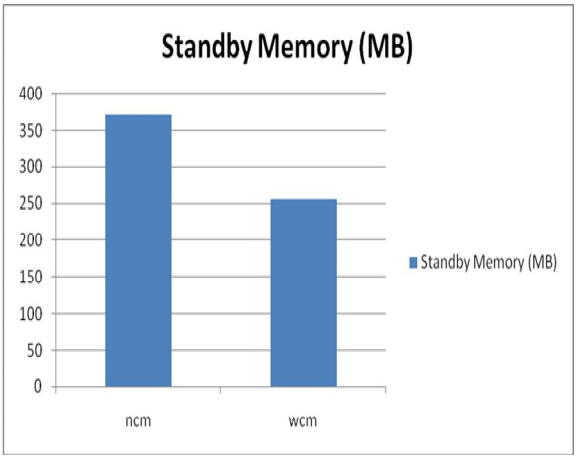

Figure 5: Standby Memory

Figure: above shows the Standby Memory for ncm (no cache management)and wcm ((with cache management)). It can be observed that wcm requires less Standby memory as compared to ncm. For ncm, available memory is 372 MB and for wcm, it is 256 MB.

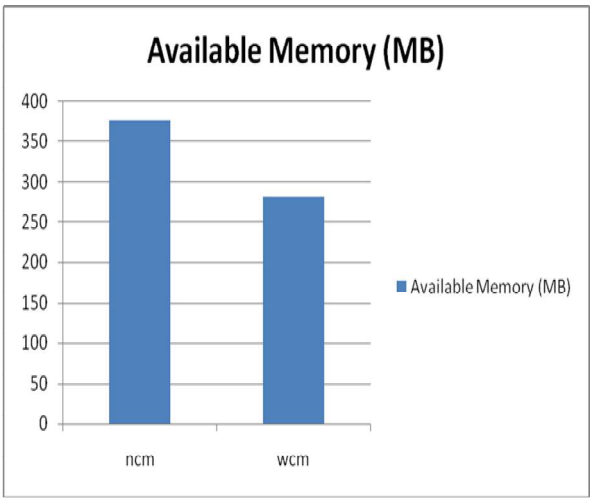

Figure 6: Available Memory

Figure: above shows the available Memory by ncm and wcm. It can be observed that available memory is quite les for wcm as compared to ncm. For ncm, available memory is 377 MB and for wcm, it is 282 MB.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2016

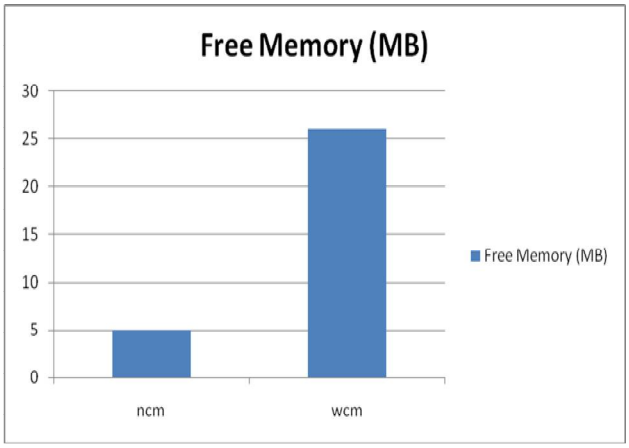

Figure 7: Free Memory

Figure: above shows the free Memory by ncm and wcm. It can be observed that ncm has less free memory is as compared to wcm. For ncm, free memory is 5 MB and for wcm, it is 26 MB.

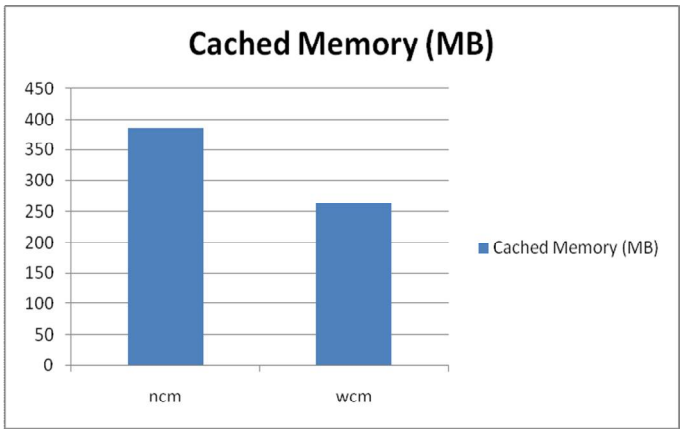

Figure 8: Cached Memory

Figure: above shows the Cached Memory by ncm and wcm. It can be observed that less cache memory is required by wcm as compared to ncm. For ncm, Cached memory is 386 MB and for wcm, it is 264 MB.

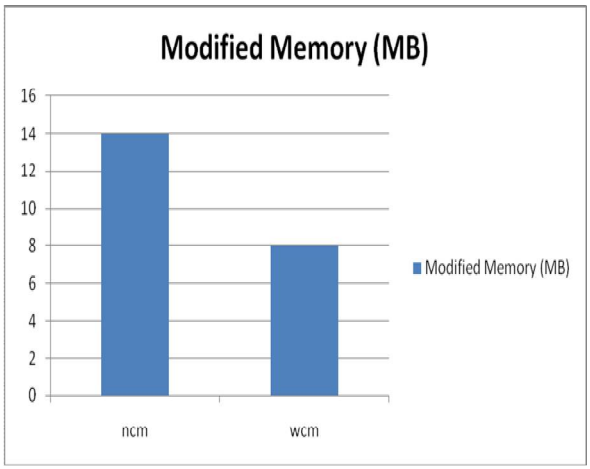

Figure 9: Modified Memory

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2016

Figure: above shows the modified Memory by ncm and wcm. It can be observed that less memory blocks modification is required by wcm as compared to ncm. For ncm, modified memory is 14MB and for wcm, it is 8 MB.

V. CONCLUSION

. In this research work, issues related to cache management using Hadoop framework and proposed a scheme which can optimize the cache utilization. As per our proposed scheme, it releases the memory resources in idle state and uses the cache memory only in active state. Results show that switching between idle and active state, processes consume less memory as compared to the process those still consumes cache, even if they are not performing any tasks. Analysis results show that wcm requires less Standby memory as compared to ncm. For ncm, available memory is 372 MB and for wcm, it is 256 MB. wcm requires less available memory as compared to ncm. For ncm, available memory is 377 MB and for wcm, it is 282 MB. ncm has less free memory is as compared to wcm. For ncm, free memory is 5 MB and for wcm, it is 26 MB. Results show that less cache memory is required by wcm as compared to ncm. For ncm, Cached memory is 386 MB and for wcm, it is 264 MB. It can be also be observed that less memory blocks modification is required by wcm as compared to ncm. For ncm, modified memory is 14MB and for wcm, it is 8 MB. Finally, it can be concluded that WCM performs well as compared to NCM under the constraints of Standby, available, cache memory and quite less memory block modification is required by WCM. Proposed work can be extended to provide the support for large scale distributed databases, Load balancers and Distributed Query Execution over Hadoop Framework.

REFERENCES

[1] Hiroaki Takasaki, Samih M. Mostafa, Shigeru Kusakabe, "Monitoring Hadoop by Using IEEE1888 in Implementing Energy-Aware Thread Scheduling", UTC-ATC-ScalCom, IEEE , pp.655-658, 2014
[2] Hidayet Aksu,Mustafa Canim, Yuan-Chi Chang, Ibrahim Korpeoglu,Özgür Ulusoy, "Graph Aware Caching Policy for Distributed Graph Stores", Cloud Engineering (IC2E), IEEE , pp. 6 – 15, 2015
[3] Zhao-Rong Lai; Che-Wei Chang; Xue Liu; Tei-Wei Kuo; Pi-Cheng Hsiu, "Deadline-aware load balancing for MapReduce", 20th International Conference on Embedded and Real-Time Computing Systems and Applications, IEEE, pp.1-10,2014
[4] Rong Gu; Furao Shen; Yihua Huang, "A parallel computing platform for training large scale neural networks", Big Data, IEEE, pp.376 – 384, 2013
[5] Yaxiong Zhao; Jie Wu; Cong Liu, "Dache: A data aware caching for big-data applications using the MapReduce framework", IEEE, pp.39-50,2014
[6] Xiao Yu and Bo Hong, "Bi-Hadoop: Extending Hadoop To Improve Support For Binary-Input Applications", IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, pp.245 – 252, 2013
[7] Sang-Deok Yoon, In-Yong Jung, Ki-Hyun Kim and Chang-Sung Jeong, "Improving HDFS performance using Local Caching System", FGCT-IEEE, pp.153-156,2013
[8] Divya M., Annappa B., "Workload Characteristics and Resource Aware Hadoop Scheduler", ReTIS, IEEE, pp.163 – 168,2015
[9] Tak-Lon Wu, Abhilash Koppula, Judy Qiu, "Integrating Pig with Harp to Support Iterative Applications with Fast Cache and Customized Communication", International Workshop on Data-Intensive Computing in the Clouds, IEEE, pp.33-39,2014
[10] Dongqi Wei, Chaoling Li, WumutiNaheman, "Organizing and Storing Method for Large-scale Unstructured Data Set with Complex Content", International Conference on Computing for Geospatial Research and Application, IEEE, pp.70-76,2013
[11] https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm
[12] https://hadoop.apache.org/
[13] https://www.mapr.com/blog/how-write-mapreduce-program