

# Data aware Caching using Map reduce Framework

K. Jaya Jones<sup>1</sup>, A. Madhavi<sup>2</sup>

<sup>1</sup>PG Student, CSE Department, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Telangana, 500072, India. E-mail: jayajones162@gmail.com

<sup>2</sup>Assistant Professor, CSE Department, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Telangana, 500072, India . E-mail: madhavi\_a@vnrvjiet.in

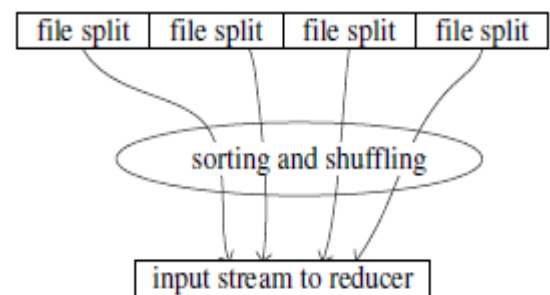
**Abstract** – The big data is the large spectrum of data also can be referred as discrete data. It usually operates on large amounts of data. The two core concepts of the Hadoop are Hadoop Distributed File System(HDFS), the storage mechanism and Map Reduce which is the programming language. The study of the Map Reduce framework is that it generates a large amount of intermediate data. Such information is discarded after the task is done. Big data open-source implementation deals with extensively large amount of data where the data can either be structured or unstructured for big-data applications. Data-aware cache framework is meant for large data applications. In Dache, tasks submit their intermediate results to the cache manager and queries the cache manager before executing the actual computing work. We implement Dache by extending the relevant components of the Hadoop project.

**Keywords:** *Hadoop Distribute File System, Dache, Mapreduce Framework*

## 1. INTRODUCTION

The main reason for big data to come into existence is when the traditional relational databases management systems were unable to process the unstructured data . Usually big data is measured in Zeta bytes and Terabytes. There are several challenges in Big data which includes sharing, storage, visualization, analysis, privacy and security. Big data analytics has become all the rage. The Hadoop Map Reduce is an open source software framework developed by Apache which assists in distributed processing of larger data sets across clusters of commodity servers. The three main components of Apache Hadoop are Hadoop Distributed File System, another Resource Negotiator and a Map Reduce framework. The Map Reduce is a used for

processing large amount of data sets, Hadoop Map Reduce is a software framework for processing applications which possesses very large amounts of data.



A Map Reduce job usually splits the input data into independent parts of data which are

processed by the map reduce in a parallel manner. The framework looks after scheduling tasks, re-executes the unsuccessful tasks and also monitors them.

Google Map Reduce is a programming model and also a software framework for Big -scale distributed Computing on large amounts of data. Figure (i) shows the high level work flow of any task. Application developers depict the computation in terms of reduce function and the underlying Map Reduce Task scheduling system automatically parallelizes the computation across the cluster of machines.

In Big and Distributed data application, large amount of input data is presented and is split in the Task Tracker. All data files are termed as “Records”. In the Map Reduce phase all input data is distributed to the Task Tracker. After Mapping, sorting and rearranging is done by using intermediate file which is created by the Task Tracker. Again it is submitted to the Task Tracker for the Reducing phase. Finally by using Map Reduce, the Reduced output is generated in the disk.

However, there is a restriction in the system, i.e., the inefficiency of incremental processing. Incremental processing means the applications which incrementally grow the input data and continuously apply computations on the input to produce output. There are duplicate computations and operations that are being performed in this process. However, Map Reduce does not have the any specified technique to identify such duplicate computations and speed up the execution. Inspired by this observation, in this paper we propose, a data-aware cache system using the Map Reduce framework, which aims at extending the Map Reduce framework and using

a cache layer for efficiently identifying and accessing cache items in a Map Reduce job.

## 2. RELATED WORK / LITERATURE REVIEW

Daniel Peng et al. proposed, a system for incrementally processing updates to a large data set, and deployed it to create the Google web search index. By replacing a batch based indexing system with an indexing system based on incremental processing using Percolator, Author process the same number of documents per day

Weikuan Yu et al. proposed, Hadoop-A, an acceleration framework that optimizes Hadoop with plugin components for fast data movement, overcoming the existing limitations. A novel network-levitated merge algorithm is introduced to merge data without repetition and disk access. In addition, a full pipeline is designed to overlap the shuffle, merge and reduce phases. Our experimental results show us that Hadoop-A significantly speeds up data movement in MapReduce and doubles the throughput of Hadoop.

Jiong Xie et al. proposed that ignoring the data locality issue in heterogeneous environments can noticeably reduce the MapReduce performance. In this paper, author addresses the problem of how to place data across nodes in a way that each node has a balanced data processing load. Given a data intensive application running on a Hadoop MapReduce cluster, our data placement scheme adaptively balances the amount of data stored in each node to achieve improved data-processing performance. Experimental results on two real data-intensive applications show that our data placement strategy can always improve the MapReduce performance by rebalancing data across nodes before performing a data-intensive application in a heterogeneous Hadoop cluster.

### 3. PROBLEM STATEMENT

In current Hadoop Map Reduce framework is that the framework generates a large flow of intermediate data. Map Reduce 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.

### 4. CACHE DESCRIPTION ON BIG DATA ENVIRONMENT

Cache refers to the intermediate data which is produced by the worker nodes/processes during the execution of any Map Reduce task. A part of cached data is stored in a Distributed File System (DFS). The content of the cache item is described by the original data and the operations applied. Formally, a cache item is described by a 2-tuple: Origin, Operation. Origin is the name of the file in the DFS. Operation is a linear list of available operations performed on the Origin file. For instance, in the word count application, each mapper node/process emits a list of word, count tuples that record the count of each word in the file that the mapper processes. Dache stores this list to a file which becomes a cache item. Given an original input data file, word list abc.txt, the cache item is described by word list abc.txt, item count. Here, the item refers to the white-space-separated character strings. Note that the new line character is also considered as one of the white spaces, so item precisely captures the word in a text file and item count directly corresponds to the word count operation performed on the data file. The exact format of the cache description of different applications varies according to their specific semantic contexts. This can be designed and implemented by application developers who are responsible

for implementing their MapReduce tasks. In our prototype, we present several supported operations such as:

**Item Count:** The count of all the occurrences of each item in a text file. The items are separated by a user defined separator.

**Sort:** This operation sorts the records of the file. The comparison operator is defined on two items and returns the order of precedence.

**Selection:** This operation selects an item that meets a given criterion. It could be an order in the list of items. A special selection operation involves selecting the median of a linear list of items.

**Transform:** This operation transforms each item in the input file into a different item. The transformation is described further by the other information in the operation descriptions. This can only be specified by the application developers.

**Classification:** This operation classifies the items in the input file into multiple groups. This could be an exact classification, where a deterministic classification criterion is applied sequentially on each item, or an approximate classification, where an iterative classification process is applied and the iteration count should be recorded.

### 5. PROTOCOL

Dache has classified the cache items into two types, map cache and reduce cache. These two types interact with each other in different cases and there are some complexities when it comes to sharing. In map cache, sharing becomes a effortless task since the operations enforced are transparent but in reduce phase sharing becomes a complex one.

**Cache request and reply:** The cache request and reply takes place in both map cache and reduce cache.

**Map Cache:** Before commencing the file splitting phase, the job tracker is responsible for sending out cache requests to the cache manager. In return, the cache manager responds with a list of cache descriptions.

**Reduce Cache:** The requested cache item is compared with the cached items in the cache manager's database. Cache manager identifies the overlaps of the original input files of the requested cache and stored cache and for this purpose linear scan method is used.

## 6. CONCLUSION

Hadoop is the tool which is used to manage and process the big data contents, which is the biggest challenge in the recent years. By using Hadoop distributed file system and map reduce concepts in hadoop we can process any big data contents within short period of time. HDFS acts as the storage mechanism in the hadoop and map reduce is used as the programming language in order to process the contents. Map reduce is operated with the help of two functions, mapper function and the reducer function.

## 7. FUTURE WORKS

Hadoop is used mainly used for strategic oriented decision making in the big data applications. It has many number of applications of which fraud detection, content optimizing, pattern recognition marketing analysis, large data transformations etc. are some of them. Hadoop framework can be used to make informed decisions in order to perform the freight audit. By doing freight audit one can prevent organizations from overpaying for the

services of freight forwarders, which are not used.

## 8. REFERENCES

- [1] J. Dean and S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Commun. of ACM, vol. 51, no. 1, pp. 107-113, 2008.
- [2] Hadoop, <http://hadoop.apache.org/>, 2013.
- [3] Java programming language, <http://www.java.com/>, 2013.
- [4] P. Th. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv., vol. 35, no. 2, pp. 114-131, 2003.
- [5] Cache algorithms, [http://en.wikipedia.org/wiki/Cache\\_algorithms](http://en.wikipedia.org/wiki/Cache_algorithms), 2013.
- [6] Amazon web services, <http://aws.amazon.com/>, 2013.
- [7] Google compute engine, <http://cloud.google.com/products/computeengine.html>, 2013.
- [8] G. Ramalingam and T. Reps. A categorized bibliography on incremental computation, in Proc. of POPL '93, New York, NY, USA, 1993.
- [9] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: A distributed storage system for structured data, in Proc. Of OSDI'2006, Berkeley, CA, USA, 2006.
- [10] S. Ghemawat, H. Gobioff, and S.-T. Leung, The google file system, SIGOPS Oper. Syst. Rev., vol. 37, no. 5, pp. 29-43, 2003.
- [11] D. Peng and F. Dabek, Largescale incremental processing using distributed transactions and notifications, in Proc. Of OSDI' 2010, Berkeley, CA, USA, 2010.
- [12] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazières, S. Mitra, A. Narayanan, D. Ongaro, G. Parulkar, M. Rosenblum, S. M. Rumble, Stratmann, and R. Stutsman, The case for ramcloud, Commun. of ACM, vol. 54, no. 7, pp. 121-130, 2011.

- [13] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, Dryad: Distributed data-parallel programs from sequential building blocks, *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 3, pp. 59-72, 2007.
- [14] L. Popa, M. Budiu, Y. Yu, and M. Isard, Dryadinc: Reusing work in large-scale computations, in *Proc. Of HotCloud'09*, Berkeley, CA, USA, 2009.
- [15] C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V. B. N. Rao, V. Sankarasubramanian, S. Seth, C. Tian, T. ZiCornell, and X. Wang, Nova: Continuous pig/hadoop workflows, in *Proc. of SIGMOD'2011*, New York, NY, USA, 2011.
- [16] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, Pig latin: A not-so-foreign language for data processing, in *Proc. of SIGMOD'2008*, New York, NY, USA, 2008.