www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 11 Nov. 2016, Page No. 18948-18956

A survey paper on logical perspective to manage BigData with incremental map reduce

Snehal Dhamelia and Prof.A.P.Kankale
Rajarshi Shahu College of Engineering Buldana

Abstract- Big Data is a collection of a huge and complex data that it becomes extremely drab to seize, store, process, reclaim and inspect it with the help of on hand database management tools or traditional data processing techniques. Now a day, data is constantly evolving and becomes a big data. The data is being generated from different sources – undertaking, social media, sensors, digital images, video, audio and clickstreams for domains together with healthcare, retail, energy and utilities. It is intended to scale up from single server to thousands of machines, each offering local computations and storage Big data with 3 V's: volume, variety and velocity. For processing such big volume of data, variety of data and the data with high velocity and having high storage capacity, we introduced Hadoop which is evolved day by day. We used MapReduce at this point as a programming model. In this paper we worn Incremental MapReduce most extensively used framework for processing big data. To improve the time of processing big data and optimizing data content of big data we applied PageRank and k-means iteratively along with MapReduce. Therefore to process big data incremental MapReduce approach is used. Incremental MapReduce 1) performs key-value pair level incremental processing, 2) supports complicated duplication computation, which is widely used in data mining applications. That means incremental MapReduce processes big data in a less time and stores it in a more optimized form.

Key words: Big data, Hadoop, incremental MapReduce, iterative.

[1] INTRODUCTION

Big data is the term for a collection of data sets so big and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. Big data streams are characterized by having high volume data (data in petabytes and zetabytes), variety of data (data may be structured, semi structured or unstructured) and velocity of data (how fast data processes). To process such big data hadoop is evolved. Hadoop is an open source framework that allows to store and process big data in distributed environment across clusters of computers using simple programming models. Hadoop is a framework for running applications on large clusters. Modeled after Google's MapReduce/GFS framework and implemented in Java. Amazon/A9, Facebook, Google, IBM, Intel Research uses hadoop.

A. Hadoop Sake

Cost effective – Because it works on commodity hardware.

Big cluster (1000 nodes on cluster) – Big storage and more processing power due to number of nodes on cluster.

Parallel processing – In MapReduce framework thousands of nodes processes in parallel and generate results in timely manner.

Big storage – Thousands of nodes with high capacity (100 GB). So storage capacity increases.

Failover – Automatic failover. If node crashes, framework will identify automatically.

Data distribution – Hadoop framework handles itself because it has thousands of nodes.

MapReduce framework – Designed and will work on hadoop framework as map and reduce fu Moving code

DOI: 10.18535/ijecs/v5i11.40

to data – Writes a MapReduce code, sends MapReduce code to cluster and get data.

Heterogeneous hardware – Can use hardware (CPU, memory etc.) of IBM, oracle, HP, Intel etc.

Hadoop uses java framework for processing huge volume of data in cluster. Hadoop includes core technology

1) HDFS - a distributed filesystem for storage purpose and
2) Map/Reduce - offline computing engine.

Hadoop File System was developed by using distributed file system design. It is run on commodity hardware. Contrasting other distributed systems, HDFS is highly fault-tolerant and designed using low-cost hardware. HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS(Hadoop distributed file system) also makes applications available to parallel processing.

Map Reduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. Map Reduce program poised of a Map function and Reduce function. Map ( ) procedure that performs filtering and sorting, reduce ( ) method that performs a summary operation. Map Reduce execution sequence is shown in fig.1.a,1.b

map(K1,V1) [<K2,V2>] reduce(K2,{V2}) [<K3,V3>]

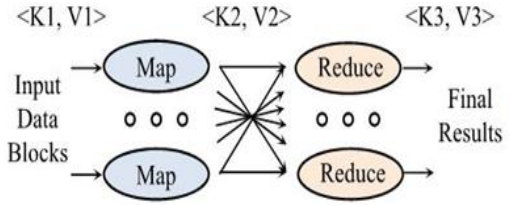

Fig1.a MapReduce computation

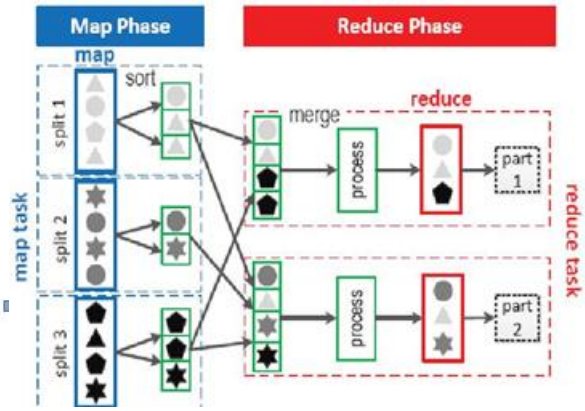

Fig1.b Execution sequence

The Map function takes a kv-pair <K1, V1> as input and computes zero or more intermediate kv-pairs <K2, V2>s. Then all <K2, V2>s are grouped by K2. The Reduce function takes a K2 and a list of {V2} as input and computes the final output kv-pairs <K3, V3>s.

A MapReduce system reads the input data of the MapReduce computation from and writes the final results to a distributed file system, which divides a file into equal-sized (e.g., 64 MB) blocks and stores the blocks across a cluster of machines. For a MapReduce program, the MapReduce system runs a JobTracker process on a maste node supervise job improvement, and a set of Task Tracker processes on worker nodes to execute the actual Map and Reduce tasks.

The Job Tracker starts a Map task per data block, and typically assigns it to the Task Tracker on the machine that holds the corresponding data block in order to minimize communication overhead. Each Map task calls the Map function for every input <K1, V1> and stores the intermediate kv-pairs <K2, V2>s on local disks. Intermediate consequences are shuffled to Reduce tasks according to a partition function on K2. After a Reduce task obtains and merges intermediate results from all Map Tasks, it invokes the Reduce function on each <K2, {V2}> to generate the final output kv-pairs <K3, V3>s.

The MapReduce system orchestrates the processing by marshaling the distributed servers, running the different tasks in parallel, organization all communications and data relocate between the different parts of the system and provided that for redundancy and fault tolerance.

Incremental MapReduce is an addition to MapReduce used for processing big data efficiently and

timely with the optimization in data content of big data and also required optimized time to process this big data.

[2] LITERATURE SURVEY

We begin by reviewing the concept of MapReduce programming model, HaLoop, IterMR, Incoop, incMR , *Iterative processing* ,Incremental processing for iterative application, and Dataflow

A. MapReduce

MapReduce is a programming model and an allied execution for processing and generating large data sets. Map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling ma-chine failures, and managing the required inter-machine communication but it takes a more time for processing [2]. Implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines

hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine Communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system [5].

B. HaLoop

Bu et al. used a new technique called as HaLoop which is modified version of Hadoop MapReduce Framework, as Map Reduce lacks built-in-support for iterative programs HaLoop allows iterative applications to be assembled from existing Hadoop programs without modification, and significantly improves their efficiency by providing inter-iteration caching mechanisms and a loop-aware scheduler to exploit these caches. HaLoop is built on top of Hadoop and extends it with a new programming model and several important optimizations that include (1) a loop-aware task scheduler, (2) loop-invariant data caching, and (3) caching for efficient fix point verification. HaLoop performs worse than plain MapReduce. This is because HaLoop employs an extra MapReduce job in each iteration to join the structure and state data [3].

A number of distributed frameworks have newly emerged for big data processing. HaLoop improves the efficiency of iterative computation by making the task scheduler loop-aware and by employing caching mechanisms. Twister employs a lightweight iterative MapReduce runtime system by sensibly constructing a Reduce-to-Map loop. IMapReduce supports iterative processing by directly passing the Reduce outputs to Map and by distinguishing variant state data from the static data [4]. Bu et al. used a new technique called as HaLoop which is modified version of Hadoop MapReduce Framework, as Map Reduce lacks built-in-support for iterative programs HaLoop allows iterative applications to be assembled from existing Hadoop programs without modification, and significantly improves their efficiency by providing inter- iteration caching mechanisms and a loop-aware scheduler to exploit these caches. HaLoop is built on top of Hadoop

and extends it with a new programming model and several important optimizations that include (1) a loop-aware task scheduler, (2) loop-invariant data caching, and (3) caching for efficient fix point verification [7].

C. Incoop

Incoop, which allows existing MapReduce programs, not designed for incremental processing, to execute transparently in an incremental manner. In Incoop, computations can respond automatically and efficiently to modifications to their input data by reusing intermediate results from previous computations, and incrementally updating the output according to the changes in the input. But Incoop Supports only task-level incremental processing and supports only one-step computation [5].

D. IncMR

IncMR framework is for incrementally processing new data of a large data set, which takes state as implicit input and combines it with new data. Map tasks are created according to new splits instead of entire splits while reduce tasks fetch their inputs including the state and the intermediate results of new map tasks from designate nodes or local nodes. Data
locality is considered as one of the main optimization means for job scheduling. But optimization of data is not done so size and location of state data is not recognized [6].

E. Iterative processing

A number of distributed frameworks have newly emerged for big data processing. HaLoop improves the efficiency of iterative computation by making the task scheduler loop-aware and by employing caching mechanisms. Twister employs a lightweight iterative MapReduce runtime system by sensibly constructing a Reduce-to-Map loop. IMapReduce supports iterative processing by directly passing the Reduce outputs to Map and by distinguishing variant state data from the static data [1].

F. Incremental processing for one-step application.

Besides Incoop, several recent studies aim at supporting incremental processing for one-step applications. Incoop detects changes to the inputs and enables the automatic update of the outputs by employing an efficient, fine-grained result reuse mechanism. This incremental nature of data suggests that performing large-scale computations incrementally can improve efficiency dramatically. But Incoop supports only task-level incremental processing. So, Incoop do not allow for reusing the large existing base of MapReduce programs.Incoop supports only one step computation [2].

G. Incremental processing for iterative application.

Naiad [14] proposes a timely dataflow paradigm that allows stateful computation and arbitrary nested iterations. To support incremental iterative computation, programmers have to completely rewrite their MapReduce programs for Naiad. In comparison, we extend the widely used MapReduce model for incremental iterative computation. Existing Map-Reduce programs can be slightly changed to run on i2MapReduce for incremental processing [3].

1.Continuous MapReduce

Ad-hoc data processing is a critical paradigm for wide-scale data processing especially for unstructured data. Ad-hoc data processing abstraction is a distributed stream processor based on MapReduce programming model to support continuous inputs. MapReduce Online adopts pipelining technique within a job and

between jobs, supports single-job and multi-job online aggregation, and also provides database continuous queries over data streams. MapReduce programming model with the continuous query model characterized by Cut-Rewind to process dynamic stream data chunk. CMR, continuous MapReduce, is architecture for continuous and large-scale data analysis. Continuous processing is a special case of incremental processing [4].

2. Incremental parallel data processing

It is a generalized architecture for continuous incremental bulk processing. It takes the prior state as an explicit input combined with the new input. A set of dataflow primitives is also provided to perform web analytics and mine large scale and evolving graphs. Percolator is a system for incrementally data processing by using distributed transactions and notifications. It is mainly used to create Google Web search index. PIESVM is a special parallel incremental extreme SVM classifier. It is designed based on MapReduce model and can save training time for SVM algorithm [4].

H. Dataflow

Systems such as CIEL, Spark, Spark Streaming and Optimus extend acyclic batch dataflow to allow dynamic modification of the dataflow graph, and thus support iteration and incremental computation without adding cycles to the dataflow. By adopting a batch-computation model, these systems inherit powerful existing techniques including fault tolerance with parallel recovery; in exchange each requires centralized modifications to the dataflow graph, which introduce substantial overhead that Naiad avoids [6].

I. Asynchronous computation

Several systems have abandoned synchronous execution in favor of a model that asynchronously updates a distributed shared data structure, in order to achieve low-latency incremental updates and fine-grained computational dependencies. Percolator structures a web indexing computation as triggers that run when new values are written into a distributed key-value store [6].

[3] IMPLEMENTATION DETAILS

Processing is done on a datasets which are the iterative with respect to its base file. Four iterative algorithms are used for the accomplishment. In section A, we design PageRank algorithm. Section B includes Kmeans algorithm. Section C and D includes the explanation of MapReduce and incremental MapReduce respectively. System architecture is shown in fig. 2.

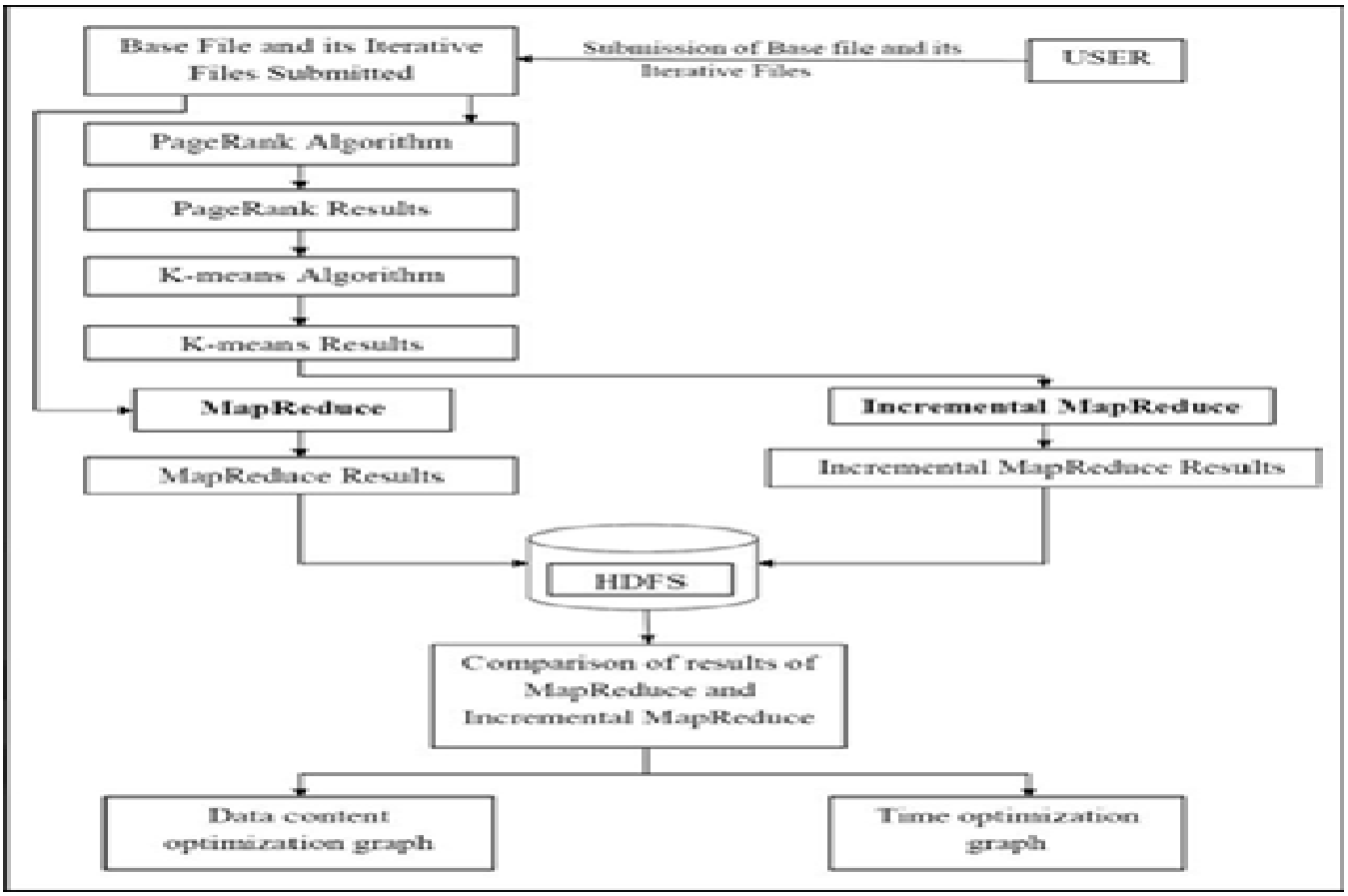

Fig. 2: System Architecture

A. PageRank

PageRank is a link investigation algorithm that assigns a numerical weighting to every element of a hyperlinked set of documents, with the point of measuring its comparative significance within the set. Votes throw by pages that are themselves "important" weigh up more deeply and facilitate to make further pages "important". PageRank is a well-known iterative graph algorithm for ranking web pages. It computes a ranking score for all vertex in a graph. After initializing all ranking scores, the working out performs a MapReduce job per iteration.

The PageRank algorithm is at the heart of the Google search engine. It is this algorithm that in spirit decides how significant a precise page is and thus how elevated it will demonstrate in a search result. In PageRank algorithm a page is important, if other important pages link to it. This scheme can be seen as a method of calculating the importance of pages by voting for them. Each link is viewed as a vote - a de facto suggestion for the importance of a page - whatever reasons the page has for connecting to a precise page. The PageRank-algorithm can, with this explanation, be seen as the contradict of an online survey, where pages vote for the importance of others, and this result is then tallied by PageRank and is reflected in the search results.

B. K-means

Clustering is a classification of objects into dissimilar groups in such a way the partitioning of dataset into subsets so that data in each subset share some common feature according to some defined distance measure

which is Euclidean distance measure.
Euclidean distance,
$d = \sqrt{\sum_{i=1}^{n}(pi - qi)^2}$ …… (3.1)

K-means clustering algorithm is a simple technique for estimating the mean of a set of k-group. K-means is a frequent and well-known clustering algorithm. It partitions a set of 'n' objects into 'k' clusters based on a similarity measurement of the objects in the dataset. The clusters have an prominent intra-cluster and a little inter-cluster similarity. As the number of objects in the cluster vary, the center of gravity of the cluster shifts.

Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group. K is positive integer number. The grouping is done by minimizing the su m of squares of distances between data and the corresponding cluster centroid

C. MapReduce

MapReduce is a programming model and an allied implementation for processing and generating big data sets. Users state a map function that processes a key/value pair to build a set of intermediary key/value pairs, and a reduce function that merges the total intermediary values allied with the same intermediate key. Implementation of MapReduce runs on a large cluster of product machines and is extremely scalable: an individual MapReduce calculation processes numerous terabytes of data on thousands of machines.

The solid part of the MapReduce output framework is a large distributed sort. The hot spots, which the application defines, are:

Input reader – The input reader divides the input into proper size splits and the framework assigns on split to each map function. The input reader reads data from steady storage space and generates key/value pairs.

Map function – The map function takes a series of key/value pairs, processes each, and generates zero or more output key/value pairs.

Partition function – Each map function output is allocated to a particular reducer by the applications partition function for sharing purposes. The partition function is given the key and the number of reducers and returns the key of the desired reducer.

Comparison function – the input for each reducer is pulled from the machine where the map ran and stores using the application's comparison function.

Reduce function – The framework calls the applications reduce function one of each unique key in the stored order. The reduce function can iterate through the values that are associated with that key and produce zero or more outputs.

Output writer – The output writer writes the output of the reduce function to the stable storage

D. Incremental MapReduce

In order to hold up incremental and iterative processing, a few MapReduce APIs are changed or added. Incremental MapReduce is an enhancement in the MapReduce. Incremental MapReduce is a task-level coarse-grain incremental processing system. Incremental MapReduce exploits inspection to stay re-computation by preliminary since the previously converged state, and during the stage incremental updates on the changeable information. Incremental MapReduce improves the run time of re-computation on plain MapReduce by an eight fold speedup. Incremental MapReduce takes input file as the output of the k-means clustering algorithm. Therefore in incremental MapReduce there is optimization of data content. Incremental MapReduce requires computation time less than the computation time of MapReduce because of the optimization in the data content.

Remark

Here we are going to use Big Data for the storage and processing Hadoop technology. HDFS will be used for the storage and MapReduce for the computation. Incremental MapReduce is a MapReduce based framework designed for incremental iterative computations. Incremental MapReduce supports more complex state-to-structure relationships. Also it supports big data sets of terabytes and petabytes.

REFERENCES

[1] Yanfeng Zhang, Shimin Chen, Qiang Wang, and Ge Yu,Member, IEEE, "i^2MapReduce: Incremental MapReduce for Mining Evolving Big Data" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 7, JULY 2015.

[2] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin, "Incoop: Mapreduce for incremental computations," in Proc. 2nd ACM Symp. Cloud Comput., 2011, pp. 7:1–7:14.

[3] Y. Zhang and S. Chen, "i2mapreduce: Incremental Iterative MapReduce," CoRR, vol. abs/1501.04854, 2013.

[4] C. Yan, X. Yang, Z. Yu, M. Li, and X. Li, "IncMR: Incremental data processing based on mapreduce," in Proc. IEEE 5th Int. Conf. Cloud Comput., 2012, pp.pp. 534–541.

[5] Jeffrey Dean et al. MapReduce: Simplified data processing on large clusters. In Proceedings of the 6th USENIX OSDI, pages 137–150, 2004.

[6] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: A timely dataflow system," in Proc. 24th ACM Symp. Oper. Syst. Principles, 2013, pp. 439– 455

[7] Yingyi Bu_ Bill Howe _ Magdalena Balazinska _ Michael D. Ernst "The HaLoop Approach to Large-Scale Iterative Data Analysis" VLDB 2010 paper "HaLoop: Efficient Iterative Data Processing on Large Clusters.

[8] Xindong Wu, Fellow, IEEE, Xingquan Zhu, Gong-Qing Wu, and Wei Ding" Data Mining with Big Data" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 1, JANUARY 2014.

[9] D. Gillick, A. Faria, and J. DeNero, MapReduce: Distributed Computing for Machine Learning, Berkley, Dec. 2006.

[10] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating MapReduce for Multi-Core and Multi-processor Systems," Proc. IEEE 13th Int'l Symp. High Perfor-mance Computer Architecture (HPCA '07),pp. 13-24, 2007.

[11] Shadi Ibrahim_ Hai Jin_ Lu Lu "Handling Partitioning Skew in MapReduce using LEEN"ACM 51 (2008) 107–113.

[12] A. Ghoting and E. Pednault, "Hadoop-ML: An Infrastructure for the Rapid Implementation of Parallel Reusable Analytics," Proc. Large-Scale Machine Learning: Parallelism and Massive Data Sets Workshop (NIPS '09),2009.

[13] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein "Online Aggregation and Continuous Query support in MapReduce" SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA. Copyright 2010 ACM 978-1-4503-0032-2/10/06.

[14] Balaji Palanisamy, Member, IEEE, Aameek Singh, Member, IEEE Ling Liu, Senior Member, IEEE" Cost-effective Resource Provisioning for MapReduce in a Cloud"gartner report 2010, 25.

[15] Niketan Pansare1, Vinayak Borkar2, Chris Jermaine1, Tyson Condie "Online Aggregation for Large MapReduce Jobs" August 29September 3, 2011, Seattle, WA Copyright 2011

VLDB Endowment, ACM.

[16] Kyong-Ha Lee and Hyunsik Choi "Parallel Data Processing with MapReduce: A Survey" SIGMOD Record, December 2011 (Vol. 40, No. 4).

[17] Albert Bifet "Mining Big Data In Real Time" Informatica 37 (2013) 15–20 DEC 2012.

[18] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "imapreduce: A distributed computing framework for iterative computation,"J. Grid Comput., vol. 10, no. 1, pp. 47–68, 2012.

[19] Jonathan Stuart Ward and Adam Barker "Undefined By Data: A Survey of Big Data Definitions" Stamford, CT: Gartner, 2012.

[20] Jimmy Lin "MapReduce Is Good Enough?" The control project. IEEE Computer 32 (2013).

[21] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: A timely dataflow system," inProc. 24th ACM Symp. Oper. Syst. Principles, 2013, pp. 439–455.

[22] Kiran kumara Reddi & Dnvsl Indira "Different Technique to Transfer Big Data : survey" IEEE Transactions on 52(8) (Aug.2013) 2348 { 2355}