VOL. 12, NO. 15, AUGUST 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2017 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

DATA SCIENCE-PARTITION AND AGGREGATION OF DATA USING MAPREDUCE

R. Shiva Shankar, V. Mnssvkr Gupta, K. V. S. Murthy and Chinta Someswara Rao
Department of Computer Science Engineering, S.R.K.R Engineering College, Bhimavaram, W.G. District, Pin, A.P. India
E-Mail: shiva.srkr@gmail.com

ABSTRACT

Now a day's information increases rapidly in different directions, that will lead to create a trouble various application fields like data science, data lake, data mining etc., one of the solution for this is the MapReduce programming model simplifies that reduces the large-scale data to small tasks. For this purpose, in this paper, we proposed a mechanism that takes the large data and converting it into small sub tasks with MapReduce and reduce network traffic cost for sub task by aggregation.

Keywords: data science, Data Lake, data mining, MapReduce, aggregation

1. INTRODUCTION

MapReduce has emerged as the most popular computing framework for big data processing due to its simple programming model and automatic management of parallel execution. It is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner [1].

The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform [2]. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job [3].

2. LITERATURE SURVEY

So many researchers work is performed on MapReduce, in this section some of the literature is discussed.

J. Dean *et al* [4] MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a 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. Many real world tasks are expressible in this model, as shown in the paper. 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. Our 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. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

The MapReduce programming model has been successfully used at Google for many different purposes. Authors attribute this success to several reasons. First, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing. Second, a large variety of problems are easily expressible as MapReduce computations. For example, MapReduce is used for the generation of data for Google's production web search service, for sorting, for data mining, for machine learning, and many other systems. Third, authors have developed an implementation of MapReduce that scales to large clusters of machines comprising thousands of machines. The implementation makes efficient use of these machine resources and therefore is suitable for use on many of the large computational problems encountered at Google. Authors have learned several things from this work. First, restricting the programming model makes it easy to parallelize and distribute computations and to make such computations fault-tolerant. Second, network bandwidth is a scarce resource. A number of optimizations in our system are therefore targeted at reducing the amount of data sent across the network: the locality optimization allows us to read data from local disks, and writing a single copy of the intermediate data to local disk saves network bandwidth. Third, redundant execution can be used to reduce the impact of slow machines, and to handle machine failures and data loss.

W. Wang *et al* [5] Scheduling map tasks to improve data locality is crucial to the performance of MapReduce. Many works have been devoted to increasing data locality for better efficiency. However, to the best of our knowledge, fundamental limits of MapReduce computing clusters with data locality, including the capacity region and theoretical bounds on the delay performance, have not been studied. In this paper, authors address these problems from a stochastic network perspective. Our focus is to strike the right balance between data-locality and load-balancing to simultaneously maximize throughput and minimize delay. Authors present a new queuing architecture and propose a map task scheduling algorithm constituted by the Join the Shortest Queue policy together with the MaxWeight policy. Authors identify an outer bound on the capacity region, and then prove that the proposed algorithm stabilizes any arrival rate vector strictly within this outer bound. It shows that the algorithm is throughput optimal and the outer bound coincides with the actual capacity Region. Further, authors study the number of backlogged tasks under the proposed algorithm, which is directly related to the delay performance based on Little's law. Authors prove that the proposed algorithm is heavy-traffic optimal, i.e., it asymptotically minimizes the number of backlogged tasks as the arrival rate vector approaches the boundary of the capacity region. Therefore, the proposed algorithm is also delay optimal in the heavy-traffic regime. Authors considered map scheduling algorithms in MapReduce with data locality. Authors first presented the capacity region of a MapReduce computing cluster with data locality and then authors proved the throughput optimality. Beyond throughput, authors showed that the proposed algorithm asymptotically minimizes the number of backlogged tasks as the arrival rate vector approaches the boundary of the capacity region, i.e., it is heavy-traffic optimal.

F. Chen *et al* [6] MapReduce has achieved tremendous success for large-scale data processing in data centers. A key feature distinguishing MapReduce from previous parallel models is that it interleaves parallel and sequential computation. Past schemes, and especially their theoretical bounds, on general parallel models are therefore, unlikely to be applied to MapReduce directly. There are many recent studies on MapReduce job and task scheduling. These studies assume that the servers are assigned in advance. in current data centers, multiple MapReduce jobs of different Importance levels run together. In this paper, authors investigate a schedule problem for MapReduce taking server assignment in to consideration as well. Authors formulate a MapReduce server-job organizer problem (MSJO) and show that it is NP-complete. Authors develop a 3-approximation algorithm and a fast heuristic. Authors evaluate our algorithms through both simulations and experiments on Amazon EC2 with an implementation in Hadoop. The results confirm the advantage of our algorithms

In this paper, authors studied MapReduce job scheduling with consideration of server assignment. Authors showed that with- out such joint consideration, there can be great performance loss. Authors formulated a MapReduce server-job organizer problem. This problem is NP-complete and authors developed a 3- approximation algorithm MarS. Authors evaluated our algorithm through extensive simulation. The results show that MarS can outperform state-of-the-art strategies by as much as 40 % in terms of total weighted job completion time. Authors also implement a prototype of MarS in Hadoop and test it with experiment on Amazon EC2. The experiment results confirm the advantage of our algorithm

Y. Wang *et al* [7] The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. Authors describe the architecture of HDFS and report on experience using HDFS to manage 25 petabytes of enterprise data at Yahoo.

This section presents some of the future work that the Hadoop team at Yahoo is considering; Hadoop being an open source project implies that new features and changes are de-ided by the Hadoop development community at large. The Hadoop cluster is effectively unavailable when its Name Node is down. Given that Hadoop is used primarily as a batch system, restarting the Name Node has been a satisfactory recovery means. However, authors have taken steps towards auto-mated failover. Currently a Backup Node receives all transactions from the primary Name Node. This will allow a failover to a warm or even a hot BackupNode if authors send block reports to both the primary NameNode and BackupNode. A few Hadoop users outside Yahoo! have experimented with manual failover. Our plan is to use Zookeeper, Yahoo's distributed consensus technology to build an automated failover solution. Scalability of the NameNode has been akey struggle. Because the NameNode keeps all the namespace and block locations in memory, the size of the NameNode heap has limited the number of files and also the number of blocks address-able.

S. Chen *et al*., [8] Recent studies and industry practices build data-center-scale computer systems to meet the high storage and processing demands of data-intensive and compute-intensive applications, such as web searches. The Map-Reduce programming model is one of the most popular programming paradigms on these systems. In this paper, authors report our experiences and insights gained from implementing three data-intensive and compute-intensive tasks that have different Characteristics from previous studies: a large-scale machine learning computation, a physical simulation task, and a digital media processing task. Authors identify desirable features and places to improve in the Map-Reduce model. Our goal is to better understand such large-scale computation and data processing in order to design better supports for them. In this paper, authors studied three data-intensive and compute-intensive applications that have very different characteristics from previous reported Map-Reduce applications. Authors find that although authors can easily implement a semantically correct Map-Reduce program, achieving good performance is tricky. For example, a computation that looks similar to word counting at the first sight may turn out to have very different characteristics, such as the number and variance of intermediate results, thus resulting in unexpected performance. Learning from the application studies, authors explore the design space for supporting data-intensive and compute-intensive applications on data-center-scale computer systems. Authors find two directions are promising: (i) enhancing a job control system with a set of desirable features; (ii) supporting flexible compos able components and including more optimization supports in Map-Reduce System. Authors plan to investigate these directions in future work.

J. Rosen *et al* [9] in this article, authors make the case for a declarative foundation for data-intensive machine learning systems. Instead of creating a new system for each specific flavor of machine learning task, or hard-coding new optimizations, authors argue for the use of recursive queries to program a variety of machine learning algorithms. By taking this approach, database query optimization techniques can be utilized to identify effective execution plans, and the resulting runtime plans can be executed on a single unified data-parallel query processing engine.

The growing demand for machine learning is pushing both industry and academia to design new types of highly scalable iterative computing systems. Examples include Mahout, Pregel, Spark, Twister, Hadoop, and PrItr. However, today's specialized machine learning platforms all tend to mix logical representations and physical implementations. As a result, today's platforms 1) require their developers to rebuild critical components and to hardcode optimization strategies and 2) limit themselves to specific runtime implementations that usually only (naturally) fit a limited subset of the potential machine learning workloads. This leads to the current state of

Practice**:** implementing new scalable machine learning algorithms is very labor-intensive and the overall data processing pipeline involves multiple disparate tools hooked together with file- and workflow-based glue. In contrast, authors have advocated a declarative foundation on which specialized machine learning workflows can be easily constructed and readily tuned. Authors have verified our approach with Datalog implementations of two popular programming models from the machine learning domain: Iterative Map-Reduce-Update, for deriving linear models, and Pregel, for graphical algorithms). The resulting Datalog programs are compact, tunable to a specific task (e.g., Batch Gradient Descent and PageRank), and translated to optimized physical plans. Our experimental results show that on a large real-world dataset and machine cluster, our optimized plans are very competitive with other systems that target the given class of ML tasks). Furthermore, authors demonstrated that our approach can offer a plan tailored to a given target task and data for a given machine resource allocation. In contrast, in our large experiments, Spark failed due to main-memory limitations and Hadoop succeeded but ran an order-of-magnitude less efficiently. The work reported here is just a first step. Authors are currently developing the ScalOps query processing components required to automate the remaining translation steps; these include the Planner/Optimizer as well as a more general algebraic foundation based on extending the Algebricks query algebra and rewrite rule framework of ASTERIX [10]. Authors also plan to investigate support for a wider range of machine learning tasks and for a more asynchronous, GraphLab-inspired programming model for encoding graphical algorithms.

S. Venkataraman *et al* [11] it is cumbersome to write machine learning and graph algorithms in data-parallel models such as MapReduce and Dryad. Authors observe that these algorithms are based on matrix computations and, hence, are inefficient to implement with the restrictive programming and communication interface of such frameworks. In this paper authors show that array-based languages such as R are suitable for implementing complex algorithms and can outperform current data parallel solutions. Since R is single threaded and does not scale to large datasets, authors have built Pronto, a distributed system that extends R and addresses many of its limitations. Pronto efficiently shares sparse structured data can leverage multi-cores, and dynamically partitions data to mitigate load imbalance. Our results show the promise of this approach: many important machine learning and graph algorithms can be expressed in a single framework and are substantially faster than those in Hadoop and Spark. Pronto advocates the use of sparse matrix operations to simplify the implementation of machine learning and graph algorithms in a cluster. Pronto uses distributed arrays for structured processing, efficiently uses multi-cores, and dynamically partitions data to reduce load imbalance. Our experience shows that pronto is a flexible computation model that can be used to implement a variety of complex algorithms

A. Matsunaga *et al* [12] Dealing with large genomic data on a limited computing resource has been an inevitable challenge in life science. Bioinformatics applications have required high performance computation capabilities for next-generation sequencing (NGS) data and the human genome sequencing data with single nucleotide polymorphisms (SNPs). From 2008, Cloud computing platforms have been widely adopted to deal with the large data sets with parallel processing tools. MapReduce parallel programming framework is dominantly used due to its fast and ancient performance for data processing on cloud clusters. This study introduces various research projects regarding to reducing a data analysis time and improving usability with their approaches. Hadoop implementations and work ow toolkits are focused on address parallel data processing tools and easy-to-use environments

These days, individual research laboratory is able to generate terabytes of data (or even larger), which is no suprises to new sequencing technologies in genomic research. High performance computation environments keep improving on processing large-scale data at low cost. The combination of MapReduce and cloud computing facilitates fast and efficient parallel processing on the virtual environment for terabyte-scale data analysis in bioinformatics, if the analysis consists of embarrassingly parallel problems. MapReduce framework is suitable for the simple and dividable tasks such as read alignment, sequence search and image recognition. Easy-to-use methods and user-friendly cloud platforms have been provided to researchers so that they can easily have ac-

VOL. 12, NO. 15, AUGUST 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2017 Asian Research Publishing Network (ARPN). All rights reserved.

cess to the cloud with their large data sets uploaded on the cloud in a secure manner. Scientic work ow may focus on improving data transfer and handling tasks regarding these usability problems. More challenges are expected to deal with data storage and analysis since it grows at unprecendented scales.

J. Wang *et al* [13] as a distributed data-parallelization (DDP) pattern, MapReduce has been adopted by many new big data analysis tools to achieve good scalability and performance in Cluster or Cloud environments. This paper explores how two binary DDP patterns, i.e., Co Group and Match, could also be used in these tools. Authors re-implemented an existing bioinformatics tool, called Cloudburst, with three different DDP pattern combinations. Authors identify two factors, namely, input data balancing and value sparseness, which could greatly affect the performances using different DDP patterns. Our experiments show: (i) a simple DDP pattern switch could speed up performance by almost two times; (ii) the identified factors can explain the differences between the “big data” era, it is very popular and effective to use DDP patterns in order to achieve scalability and parallelization. These DDP patterns also bring challenges on which pattern or pattern combination is the best for a certain tool. This paper demonstrates different DDP patterns could have a great impact on the performances of the same tool. Authors find that although MapReduce can be used for wider range of applications with either one or two input datasets, it is not always the best choice in terms of application complexity and performance. To understand the differences, authors identified two affecting factors, namely input data balancing and value sparseness, on their performance differences. The feasibility of these two factors is verified through experiments. Authors believe many tools in bioinformatics and other domains have a similar logic with CloudBurst as they need to match two input datasets, and therefore could also benefit from our findings. For future work, authors plan to investigate more tools that are suitable for multiple DDP patterns and their performances on other DDP engines like Hadoop, which will generalize our findings. Authors will also study how to utilize the identified factors to automatically select the best DDP pattern combination from multiple available ones.

R. Liao *et al* [14] Nonnegative matrix factorization (NMF) has an established reputation as a useful data analysis technique in numerous applications. However, its usage in practical situations is undergoing challenges in recent years. The fundamental factor to this is the increasingly growing size of the datasets available and needed in the information sciences. To address this, in this work authors propose to use structured random compression, that is, random projections that exploit the data structure, for two NMF variants: classical and separable. In separable NMF (SNMF) the Left factors are a subset of the columns of the input matrix. Authors present suitable formulations for each problem, dealing with different representative algorithms within each one. Authors show that the resulting compressed techniques are faster than their uncompressed variants, vastly reduce memory demands, and do not encompass any significant deterioration in performance. The proposed structured random projections for SNMF allow dealing with arbitrarily shaped large matrices, beyond the standard limit of tall-and-skinny matrices, granting access to very efficient computations in this general setting. Authors accompany the algorithmic presentation with theoretical foundations and numerous and diverse examples, showing the suitability of the proposed approaches.

In this work authors proposed to use structured random projections for NMF and SNMF. For NMF, authors presented formulations for three popular techniques, namely, multiplicative updates, active set method for nonnegative least squares and ADMM. For SNMF, authors presented a general technique that can be used with any algorithm. In all cases, authors showed that the resulting compressed techniques are faster than their uncompressed variants and, at the same time; do not introduce significant errors in the final result. There are in the literature very efficient SNMF algorithms for tall-and-skinny matrices. Interestingly, the use of structured random projections allows computing SNMF for arbitrarily large matrices, granting access to very efficient computations in the general setting. As a byproduct, authors also propose an algorithmic solution for computing structured random projections of extremely large matrices (i.e., matrices so large that even after compression they do not fit in main memory). This is useful as a general tool for computing many different matrix decompositions, such as the singular value decomposition, for example. Authors are currently investigating the problem of replacing the Frobenius norm with and Norm in our compressed variants of NMF and SNMF. In this setting, the fast Cauchy transform is a suitable alternative to structured random projections. Compression consists of sampling and rescaling rows of A, thus identifying the so-called corset of the problem. This formulation is of particular interest for network analysis, where authors need to deal with sparse structures.

3. METHODOLOGY

Intermediate data are shuffled according to a hash function in Hadoop, which would lead to large network traffic because it ignores network topology and data size associated with each key. To tackle this problem incurred by the traffic-oblivious partition scheme, we take into account of both task locations and data size associated with each key in this paper. By assigning keys with larger data size to reduce tasks closer to map tasks, network traffic can be significantly reduced. To further reduce network traffic within a MapReduce job, we consider to aggregate data with the same keys before sending them to remote reduce tasks. Although a similar function, called combiner, has been already adopted by Hadoop, it operates immediately after a map task solely for its generated data, failing to exploit the data aggregation opportunities among multiple tasks on different machines.

VOL. 12, NO. 15, AUGUST 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2017 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

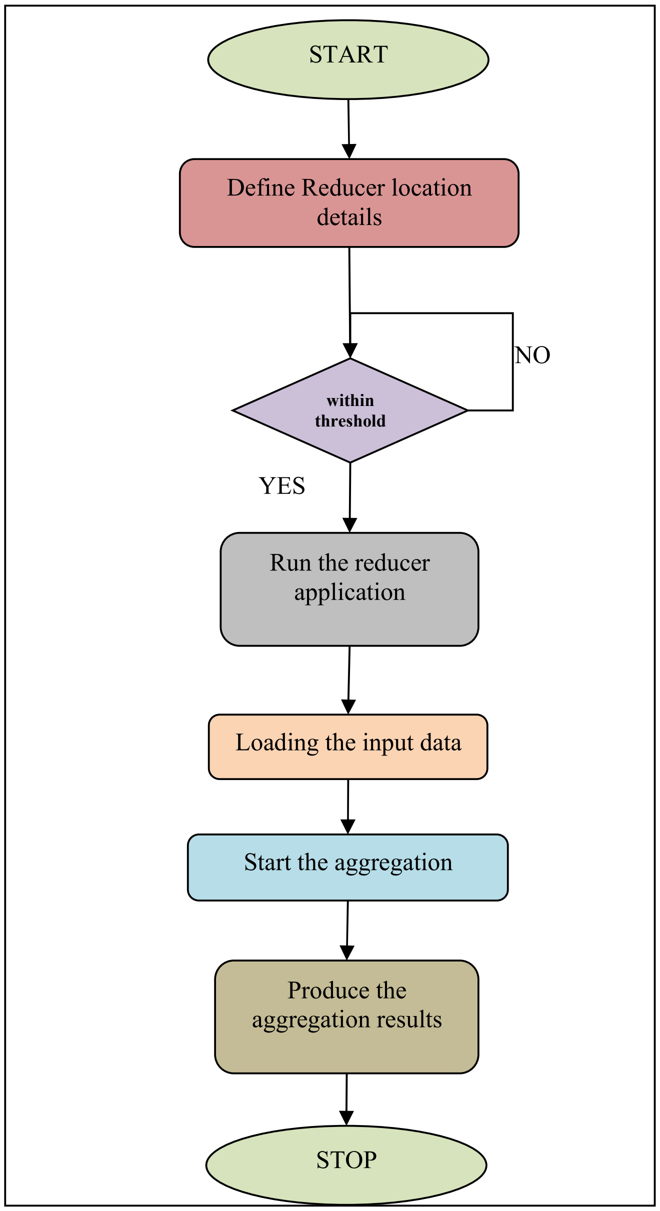
START
Define Reducer location details
NO
within threshold
YES
Run the reducer application
Loading the input data
Start the aggregation
Produce the aggregation results
STOP

Figure-1. System structure.

In this paper, we jointly consider data partition and aggregation for a Map Reduce job with an objective that is to minimize the total network traffic. In particular, we propose a distributed algorithm for big data applications by decomposing the original large-scale problem into several sub problems that can be solved in parallel. Moreover, an online algorithm is designed to deal with the data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost in both offline and online cases. The structure is shown in Figure-1 and actual process is shown in Figure-2. In the proposed system reducer locations are defined, if they are within the threshold, the reducer application is executed, the input data is loaded and finally aggregation is formed.

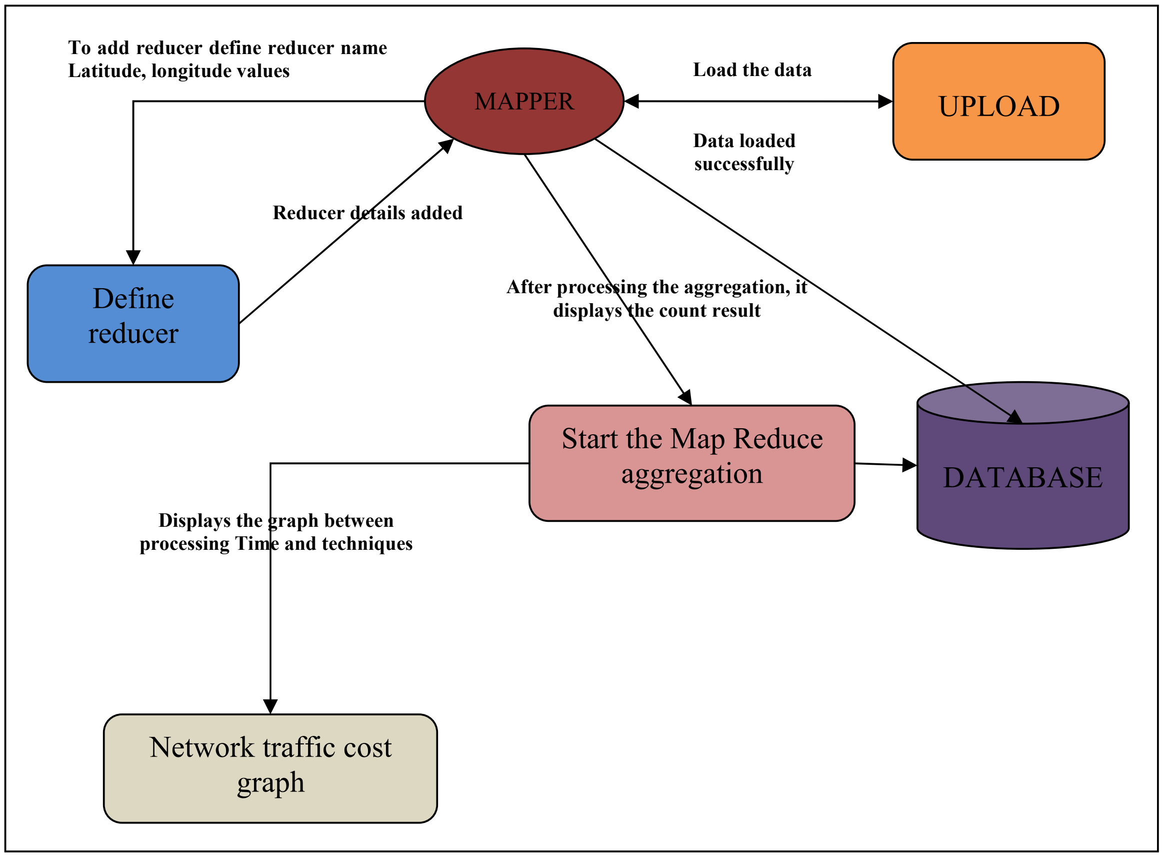
To add reducer define reducer name
Latitude, longitude values
MAPPER
Load the data
UPLOAD
Data loaded successfully
Reducer details added
Define reducer
After processing the aggregation, it displays the count result
Start the Map Reduce aggregation
DATABASE
Displays the graph between processing Time and techniques
Network traffic cost graph

Figure-2. Actual process.

3.1. Module description

In this paper we have three modules, namely, *Upload Module, MapReduce Aggregation Module and Graph Module.*

Upload module: In this module, we can upload the documents and also select the reducer to which the data is to reduce. Firstly select the reducer and then after upload the documents to transfer through the network.

MapReduce aggregation: In this Module, input data has been processed by the reducer which is nearer to the mapper location. After completion of Processing it will display the aggregated data to the network user.

Graph: This graph represents network traffic cost for no aggregation processing time and aggregation processing time.

3.2. Result description

Mapreduce programming model is used to retrieve the analyzing the data. The implementation is the working of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirement specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

To get the Big Data issue solved, we need to get a Hadoop cluster that is able to store it and perform parallel computation across a large computer cluster. Hadoop is the most popular solution. Hadoop is an open source Java framework. The illustrations of experiment result shows that our proposal can successfully decrease network traffic cost under different complex settings. Input data set is shown in Table-1. Aggregated data after successfully processing is shown in Table-2.

Table-1. Input data.

a1.txt
1. HR logs into the online hiring center to generate a requisition once a position is required. The requisition is routed electronically for approval using the applicant tracking system. Hiring managers log into a limited-view version and are able to approve or deny requisitions. Once a hiring official approves a requisition, it moves to the next level of approval. HR is notified automatically at each stage of approval or denial. 2. Within the application, HR staff uses the job description or class specification to create a job posting, write supplement questions and create an exam plan. This includes setting up auto-scoring for minimum qualification, testing hurdles and notes in the recruitment folder. All of this information resides in the tracking system. 3. The job is posted on a specific date, HR then notifies all subscribers on a “rider-alert system” that a new recruitment has opened. Subscribers set their alert preferences to email and /or cell phone. This system is designed for riders of the transit system, but has an opt-in feature people can use to ensure they receive an alert when new jobs post. 4. Applications are screened for minimum qualification by HR generally but this will be done via an auto-scoring system and based on responses to application questions. Those applications passed the minimum qualifications screening are reviewed to determine whether they meet entry requirements through the full exam process by HR and then the hiring officials. 5. Selected applicants will be invited for written test and interviews, depending on the nature of the position and how the exam plan is set up. Invitations to test, schedules and notices are sent out by the system via email and SMS (Cell phones) using template created.
aa.txt
1. HR logs into the online hiring center to generate a requisition once a position is required. The requisition is routed electronically for approval using the applicant tracking system. Hiring managers log into a limited-view version and are able to approve or deny requisitions. Once a hiring official approves a requisition, it moves to the next level of approval. HR is notified automatically at each stage of approval or denial. 2. Within the application, HR staff uses the job description or class specification to create a job posting, write supplement questions and create an exam plan. This includes setting up auto-scoring for minimum qualification, testing hurdles and notes in the recruitment folder. All of this information resides in the tracking system. 3. The job is posted on a specific date, HR then notifies all subscribers on a “rider-alert system” that a new recruitment has opened. Subscribers set their alert preferences to email and /or cell phone. This system is designed for riders of the transit system, but has an opt-in feature people can use to ensure they receive an alert when new jobs post. 4. Applications are screened for minimum qualification by HR generally but this will be done via an auto-scoring system and based on responses to application questions. Those applications passed the minimum qualifications screening are reviewed to determine whether they meet entry requirements through the full exam process by HR and then the hiring officials. 5. Selected applicants will be invited for written test and interviews, depending on the nature of the position and how the exam plan is set up. Invitations to test, schedules and notices are sent out by the system via email and SMS (Cell phones) using template created. 1. HR logs into the online hiring center to generate a requisition once a position is required. The requisition is routed electronically for approval using the applicant tracking system. Hiring managers log into a limited-view version and are able to approve or deny requisitions. Once a hiring official approves a requisition, it moves to the next level of approval. HR is notified automatically at each stage of approval or denial. 2. Within the application, HR staff uses the job description or class specification to create a job posting, write supplement questions and create an exam plan. This includes setting up auto-scoring for minimum qualification, testing hurdles and notes in the recruitment folder. All of this information resides in the tracking system. 3. The job is posted on a specific date, HR then notifies all subscribers on a “rider-alert system” that a new recruitment has opened. Subscribers set their alert preferences to email and /or cell phone. This system is designed for riders of the transit system, but has an opt-in feature people can use to ensure they receive an alert when new jobs post. 4. Applications are screened for minimum qualification by HR generally but this will be done via an auto-scoring system and based on responses to application questions. Those applications passed the minimum qualifications screening are reviewed to determine whether they meet entry requirements through the full exam process by HR and then the hiring officials.

5. Selected applicants will be invited for written test and interviews, depending on the nature of the position and how the exam plan is set up. Invitations to test, schedules and notices are sent out by the system via email and SMS (Cell phones) using template created.

1. HR logs into the online hiring center to generate a requisition once a position is required. The requisition is routed electronically for approval using the applicant tracking system. Hiring managers log into a limited-view version and are able to approve or deny requisitions. Once a hiring official approves a requisition, it moves to the next level of approval. HR is notified automatically at each stage of approval or denial.
2. Within the application, HR staff uses the job description or class specification to create a job posting, write supplement questions and create an exam plan. This includes setting up auto-scoring for minimum qualification, testing hurdles and notes in the recruitment folder. All of this information resides in the tracking system.
3. The job is posted on a specific date, HR then notifies all subscribers on a "rider-alert system" that a new recruitment has opened. Subscribers set their alert preferences to email and /or cell phone. This system is designed for riders of the transit system, but has an opt-in feature people can use to ensure they receive an alert when new jobs post.
4. Applications are screened for minimum qualification by HR generally but this will be done via an auto-scoring system and based on responses to application questions. Those applications passed the minimum qualifications screening are reviewed to determine whether they meet entry requirements through the full exam process by HR and then the hiring officials.

5. Selected applicants will be invited for written test and interviews, depending on the nature of the position and how the exam plan is set up. Invitations to test, schedules and notices are sent out by the system via email and SMS (Cell phones) using template created.

Table-2. Aggregated data.

1@a1.txt,1	**to@a1.txt,1**	**notifies@aa.txt,1**	**next@aa.txt,1**	**up@aa.txt,1**	**are@aa.txt,1**
hr@a1.txt,1	application@a1.txt,1	all@aa.txt,1	level@aa.txt,1	invitations@aa.txt,1	screened@aa.txt,1
logs@a1.txt,1	questions@a1.txt,1	subscribers@aa.txt,1	of@aa.txt,1	to@aa.txt,1	for@aa.txt,1
into@a1.txt,1	those@a1.txt,1	on@aa.txt,1	approval@aa.txt,1	test@aa.txt,1	minimum@aa.txt,1
the@a1.txt,1	applications@a1.txt,1	a@aa.txt,1	hr@aa.txt,1	schedules@aa.txt,1	qualification@aa.txt,1
online@a1.txt,1	passed@a1.txt,1	rider@aa.txt,1	is@aa.txt,1	and@aa.txt,1	by@aa.txt,1
hiring@a1.txt,1	the@a1.txt,1	alert@aa.txt,1	notified@aa.txt,1	notices@aa.txt,1	hr@aa.txt,1
center@a1.txt,1	minimum@a1.txt,1	system@aa.txt,1	automatically@aa.txt,1	are@aa.txt,1	generally@aa.txt,1
to@a1.txt,1	qualifications@a1.txt,1	that@aa.txt,1	at@aa.txt,1	sent@aa.txt,1	but@aa.txt,1
generate@a1.txt,1	screening@a1.txt,1	a@aa.txt,1	each@aa.txt,1	out@aa.txt,1	this@aa.txt,1
a@a1.txt,1	are@a1.txt,1	new@aa.txt,1	stage@aa.txt,1	by@aa.txt,1	will@aa.txt,1
requisition@a1.txt,1	reviewed@a1.txt,1	recruitment@aa.txt,1	of@aa.txt,1	the@aa.txt,1	be@aa.txt,1
once@a1.txt,1	to@a1.txt,1	has@aa.txt,1	approval@aa.txt,1	system@aa.txt,1	done@aa.txt,1
a@a1.txt,1	determine@a1.txt,1	opened@aa.txt,1	or@aa.txt,1	via@aa.txt,1	via@aa.txt,1
position@a1.txt,1	whether@a1.txt,1	subscribers@aa.txt,1	denial@aa.txt,1	email@aa.txt,1	an@aa.txt,1
is@a1.txt,1	they@a1.txt,1	set@aa.txt,1	2@aa.txt,1	and@aa.txt,1	auto@aa.txt,1
required@a1.txt,1	meet@a1.txt,1	their@aa.txt,1	within@aa.txt,1	sms@aa.txt,1	scoring@aa.txt,1
the@a1.txt,1	entry@a1.txt,1	alert@aa.txt,1	the@aa.txt,1	cell@aa.txt,1	system@aa.txt,1
requisition@a1.txt,1	requirements@a1.txt,1	preferences@aa.txt,1	application@aa.txt,1	phones@aa.txt,1	and@aa.txt,1

VOL. 12, NO. 15, AUGUST 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2017 Asian Research Publishing Network (ARPN). All rights reserved.
www.arpnjournals.com

is@a1.txt,1	through@a1.txt,1	to@aa.txt,1	hr@aa.txt,1	using@aa.txt,1	based@aa.txt,1
routed@a1.txt,1	the@a1.txt,1	email@aa.txt,1	staff@aa.txt,1	template@aa.txt,1	on@aa.txt,1
electronically@a1 .txt,1	full@a1.txt,1	and@aa.txt,1	uses@aa.txt,1	created@aa.txt,1	responses@aa.txt, 1
for@a1.txt,1	exam@a1.txt,1	or@aa.txt,1	the@aa.txt,1	1@aa.txt,1	to@aa.txt,1
approval@a1.txt, 1	process@a1.txt,1	cell@aa.txt,1	job@aa.txt,1	hr@aa.txt,1	application@aa.tx t,1
using@a1.txt,1	by@a1.txt,1	phone@aa.txt,1	description@aa.tx t,1	logs@aa.txt,1	questions@aa.txt, 1
the@a1.txt,1	hr@a1.txt,1	this@aa.txt,1	or@aa.txt,1	into@aa.txt,1	those@aa.txt,1
applicant@a1.txt, 1	and@a1.txt,1	system@aa.txt,1	class@aa.txt,1	the@aa.txt,1	applications@aa.t xt,1
tracking@a1.txt,1	then@a1.txt,1	is@aa.txt,1	specification@aa. txt,1	online@aa.txt,1	passed@aa.txt,1
system@a1.txt,1	the@a1.txt,1	designed@aa.txt, 1	to@aa.txt,1	hiring@aa.txt,1	the@aa.txt,1
hiring@a1.txt,1	hiring@a1.txt,1	for@aa.txt,1	create@aa.txt,1	center@aa.txt,1	minimum@aa.txt, 1
managers@a1.txt, 1	officials@a1.txt,1	riders@aa.txt,1	a@aa.txt,1	to@aa.txt,1	qualifications@aa .txt,1
log@a1.txt,1	5@a1.txt,1	of@aa.txt,1	job@aa.txt,1	generate@aa.txt,1	screening@aa.txt, 1
into@a1.txt,1	selected@a1.txt,1	the@aa.txt,1	posting@aa.txt,1	a@aa.txt,1	are@aa.txt,1
a@a1.txt,1	applicants@a1.txt ,1	transit@aa.txt,1	write@aa.txt,1	requisition@aa.tx t,1	reviewed@aa.txt, 1
limited@a1.txt,1	will@a1.txt,1	system@aa.txt,1	supplement@aa.t xt,1	once@aa.txt,1	to@aa.txt,1
view@a1.txt,1	be@a1.txt,1	but@aa.txt,1	questions@aa.txt, 1	a@aa.txt,1	determine@aa.txt ,1
version@a1.txt,1	invited@a1.txt,1	has@aa.txt,1	and@aa.txt,1	position@aa.txt,1	whether@aa.txt,1
and@a1.txt,1	for@a1.txt,1	an@aa.txt,1	create@aa.txt,1	is@aa.txt,1	they@aa.txt,1
are@a1.txt,1	written@a1.txt,1	opt@aa.txt,1	an@aa.txt,1	required@aa.txt,1	meet@aa.txt,1
able@a1.txt,1	test@a1.txt,1	in@aa.txt,1	exam@aa.txt,1	the@aa.txt,1	entry@aa.txt,1
to@a1.txt,1	and@a1.txt,1	feature@aa.txt,1	plan@aa.txt,1	requisition@aa.tx t,1	requirements@aa. txt,1
approve@a1.txt,1	interviews@a1.txt ,1	people@aa.txt,1	this@aa.txt,1	is@aa.txt,1	through@aa.txt,1
or@a1.txt,1	depending@a1.txt ,1	can@aa.txt,1	includes@aa.txt,1	routed@aa.txt,1	the@aa.txt,1
deny@a1.txt,1	on@a1.txt,1	use@aa.txt,1	setting@aa.txt,1	electronically@aa .txt,1	full@aa.txt,1
requisitions@a1.t xt,1	the@a1.txt,1	to@aa.txt,1	up@aa.txt,1	for@aa.txt,1	exam@aa.txt,1
once@a1.txt,1	nature@a1.txt,1	ensure@aa.txt,1	auto@aa.txt,1	approval@aa.txt, 1	process@aa.txt,1
a@a1.txt,1	of@a1.txt,1	they@aa.txt,1	scoring@aa.txt,1	using@aa.txt,1	by@aa.txt,1
hiring@a1.txt,1	the@a1.txt,1	receive@aa.txt,1	for@aa.txt,1	the@aa.txt,1	hr@aa.txt,1
official@a1.txt,1	position@a1.txt,1	an@aa.txt,1	minimum@aa.txt, 1	applicant@aa.txt, 1	and@aa.txt,1
approves@a1.txt, 1	and@a1.txt,1	alert@aa.txt,1	qualification@aa. txt,1	tracking@aa.txt,1	then@aa.txt,1
a@a1.txt,1	how@a1.txt,1	when@aa.txt,1	testing@aa.txt,1	system@aa.txt,1	the@aa.txt,1
requisition@a1.tx t,1	the@a1.txt,1	new@aa.txt,1	hurdles@aa.txt,1	hiring@aa.txt,1	hiring@aa.txt,1

it@a1.txt,1	exam@a1.txt,1	jobs@aa.txt,1	and@aa.txt,1	managers@aa.txt,1	officials@aa.txt,1
moves@a1.txt,1	plan@a1.txt,1	post@aa.txt,1	notes@aa.txt,1	log@aa.txt,1	5@aa.txt,1
to@a1.txt,1	is@a1.txt,1	4@aa.txt,1	in@aa.txt,1	into@aa.txt,1	selected@aa.txt,1
the@a1.txt,1	set@a1.txt,1	applications@aa.txt,1	the@aa.txt,1	a@aa.txt,1	applicants@aa.txt,1
next@a1.txt,1	up@a1.txt,1	are@aa.txt,1	recruitment@aa.txt,1	limited@aa.txt,1	will@aa.txt,1
level@a1.txt,1	invitations@a1.txt,1	screened@aa.txt,1	folder@aa.txt,1	view@aa.txt,1	be@aa.txt,1
of@a1.txt,1	to@a1.txt,1	for@aa.txt,1	all@aa.txt,1	version@aa.txt,1	invited@aa.txt,1
approval@a1.txt,1	test@a1.txt,1	minimum@aa.txt,1	of@aa.txt,1	and@aa.txt,1	for@aa.txt,1
hr@a1.txt,1	schedules@a1.txt,1	qualification@aa.txt,1	this@aa.txt,1	are@aa.txt,1	written@aa.txt,1
is@a1.txt,1	and@a1.txt,1	by@aa.txt,1	information@aa.txt,1	able@aa.txt,1	test@aa.txt,1
notified@a1.txt,1	notices@a1.txt,1	hr@aa.txt,1	resides@aa.txt,1	to@aa.txt,1	and@aa.txt,1
automatically@a1.txt,1	are@a1.txt,1	generally@aa.txt,1	in@aa.txt,1	approve@aa.txt,1	interviews@aa.txt,1
at@a1.txt,1	sent@a1.txt,1	but@aa.txt,1	the@aa.txt,1	or@aa.txt,1	depending@aa.txt,1
each@a1.txt,1	out@a1.txt,1	this@aa.txt,1	tracking@aa.txt,1	deny@aa.txt,1	on@aa.txt,1
stage@a1.txt,1	by@a1.txt,1	will@aa.txt,1	system@aa.txt,1	requisitions@aa.txt,1	the@aa.txt,1
of@a1.txt,1	the@a1.txt,1	be@aa.txt,1	3@aa.txt,1	once@aa.txt,1	nature@aa.txt,1
approval@a1.txt,1	system@a1.txt,1	done@aa.txt,1	the@aa.txt,1	a@aa.txt,1	of@aa.txt,1
or@a1.txt,1	via@a1.txt,1	via@aa.txt,1	job@aa.txt,1	hiring@aa.txt,1	the@aa.txt,1
denial@a1.txt,1	email@a1.txt,1	an@aa.txt,1	is@aa.txt,1	official@aa.txt,1	position@aa.txt,1
2@a1.txt,1	and@a1.txt,1	auto@aa.txt,1	posted@aa.txt,1	approves@aa.txt,1	and@aa.txt,1
within@a1.txt,1	sms@a1.txt,1	scoring@aa.txt,1	on@aa.txt,1	a@aa.txt,1	how@aa.txt,1
the@a1.txt,1	cell@a1.txt,1	system@aa.txt,1	a@aa.txt,1	requisition@aa.txt,1	the@aa.txt,1
application@a1.txt,1	phones@a1.txt,1	and@aa.txt,1	specific@aa.txt,1	it@aa.txt,1	exam@aa.txt,1
hr@a1.txt,1	using@a1.txt,1	based@aa.txt,1	date@aa.txt,1	moves@aa.txt,1	plan@aa.txt,1
staff@a1.txt,1	template@a1.txt,1	on@aa.txt,1	hr@aa.txt,1	to@aa.txt,1	is@aa.txt,1
uses@a1.txt,1	created@a1.txt,1	responses@aa.txt,1	then@aa.txt,1	the@aa.txt,1	set@aa.txt,1
the@a1.txt,1	1@aa.txt,1	to@aa.txt,1	notifies@aa.txt,1	next@aa.txt,1	up@aa.txt,1
job@a1.txt,1	hr@aa.txt,1	application@aa.txt,1	all@aa.txt,1	level@aa.txt,1	invitations@aa.txt,1
description@a1.txt,1	logs@aa.txt,1	questions@aa.txt,1	subscribers@aa.txt,1	of@aa.txt,1	to@aa.txt,1
or@a1.txt,1	into@aa.txt,1	those@aa.txt,1	on@aa.txt,1	approval@aa.txt,1	test@aa.txt,1
class@a1.txt,1	the@aa.txt,1	applications@aa.txt,1	a@aa.txt,1	hr@aa.txt,1	schedules@aa.txt,1
specification@a1.txt,1	online@aa.txt,1	passed@aa.txt,1	rider@aa.txt,1	is@aa.txt,1	and@aa.txt,1
to@a1.txt,1	hiring@aa.txt,1	the@aa.txt,1	alert@aa.txt,1	notified@aa.txt,1	notices@aa.txt,1

create@a1.txt,1	center@aa.txt,1	minimum@aa.txt,1	system@aa.txt,1	automatically@aa.txt,1	are@aa.txt,1
a@a1.txt,1	to@aa.txt,1	qualifications@aa.txt,1	that@aa.txt,1	at@aa.txt,1	sent@aa.txt,1
job@a1.txt,1	generate@aa.txt,1	screening@aa.txt,1	a@aa.txt,1	each@aa.txt,1	out@aa.txt,1
posting@a1.txt,1	a@aa.txt,1	are@aa.txt,1	new@aa.txt,1	stage@aa.txt,1	by@aa.txt,1
write@a1.txt,1	requisition@aa.txt,1	reviewed@aa.txt,1	recruitment@aa.txt,1	of@aa.txt,1	the@aa.txt,1
supplement@a1.txt,1	once@aa.txt,1	to@aa.txt,1	has@aa.txt,1	approval@aa.txt,1	system@aa.txt,1
questions@a1.txt,1	a@aa.txt,1	determine@aa.txt,1	opened@aa.txt,1	or@aa.txt,1	via@aa.txt,1
and@a1.txt,1	position@aa.txt,1	whether@aa.txt,1	subscribers@aa.txt,1	denial@aa.txt,1	email@aa.txt,1
create@a1.txt,1	is@aa.txt,1	they@aa.txt,1	set@aa.txt,1	2@aa.txt,1	and@aa.txt,1
an@a1.txt,1	required@aa.txt,1	meet@aa.txt,1	their@aa.txt,1	within@aa.txt,1	sms@aa.txt,1
exam@a1.txt,1	the@aa.txt,1	entry@aa.txt,1	alert@aa.txt,1	the@aa.txt,1	cell@aa.txt,1
plan@a1.txt,1	requisition@aa.txt,1	requirements@aa.txt,1	preferences@aa.txt,1	application@aa.txt,1	phones@aa.txt,1
this@a1.txt,1	is@aa.txt,1	through@aa.txt,1	to@aa.txt,1	hr@aa.txt,1	using@aa.txt,1
includes@a1.txt,1	routed@aa.txt,1	the@aa.txt,1	email@aa.txt,1	staff@aa.txt,1	template@aa.txt,1
setting@a1.txt,1	electronically@aa.txt,1	full@aa.txt,1	and@aa.txt,1	uses@aa.txt,1	created@aa.txt,1
up@a1.txt,1	for@aa.txt,1	exam@aa.txt,1	or@aa.txt,1	the@aa.txt,1	
auto@a1.txt,1	approval@aa.txt,1	process@aa.txt,1	cell@aa.txt,1	job@aa.txt,1	
scoring@a1.txt,1	using@aa.txt,1	by@aa.txt,1	phone@aa.txt,1	description@aa.txt,1	
for@a1.txt,1	the@aa.txt,1	hr@aa.txt,1	this@aa.txt,1	or@aa.txt,1	
minimum@a1.txt,1	applicant@aa.txt,1	and@aa.txt,1	system@aa.txt,1	class@aa.txt,1	
qualification@a1.txt,1	tracking@aa.txt,1	then@aa.txt,1	is@aa.txt,1	specification@aa.txt,1	
testing@a1.txt,1	system@aa.txt,1	the@aa.txt,1	designed@aa.txt,1	to@aa.txt,1	
hurdles@a1.txt,1	hiring@aa.txt,1	hiring@aa.txt,1	for@aa.txt,1	create@aa.txt,1	
and@a1.txt,1	managers@aa.txt,1	officials@aa.txt,1	riders@aa.txt,1	a@aa.txt,1	
notes@a1.txt,1	log@aa.txt,1	5@aa.txt,1	of@aa.txt,1	job@aa.txt,1	
in@a1.txt,1	into@aa.txt,1	selected@aa.txt,1	the@aa.txt,1	posting@aa.txt,1	
the@a1.txt,1	a@aa.txt,1	applicants@aa.txt,1	transit@aa.txt,1	write@aa.txt,1	
recruitment@a1.txt,1	limited@aa.txt,1	will@aa.txt,1	system@aa.txt,1	supplement@aa.txt,1	
folder@a1.txt,1	view@aa.txt,1	be@aa.txt,1	but@aa.txt,1	questions@aa.txt,1	
all@a1.txt,1	version@aa.txt,1	invited@aa.txt,1	has@aa.txt,1	and@aa.txt,1	
of@a1.txt,1	and@aa.txt,1	for@aa.txt,1	an@aa.txt,1	create@aa.txt,1	
this@a1.txt,1	are@aa.txt,1	written@aa.txt,1	opt@aa.txt,1	an@aa.txt,1	
information@a1.txt,1	able@aa.txt,1	test@aa.txt,1	in@aa.txt,1	exam@aa.txt,1	
resides@a1.txt,1	to@aa.txt,1	and@aa.txt,1	feature@aa.txt,1	plan@aa.txt,1	

VOL. 12, NO. 15, AUGUST 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2017 Asian Research Publishing Network (ARPN). All rights reserved.

in@a1.txt,1	approve@aa.txt,1	interviews@aa.txt,1	people@aa.txt,1	this@aa.txt,1	
the@a1.txt,1	or@aa.txt,1	depending@aa.txt,1	can@aa.txt,1	includes@aa.txt,1	
tracking@a1.txt,1	deny@aa.txt,1	on@aa.txt,1	use@aa.txt,1	setting@aa.txt,1	
system@a1.txt,1	requisitions@aa.txt,1	the@aa.txt,1	to@aa.txt,1	up@aa.txt,1	
3@a1.txt,1	once@aa.txt,1	nature@aa.txt,1	ensure@aa.txt,1	auto@aa.txt,1	
the@a1.txt,1	a@aa.txt,1	of@aa.txt,1	they@aa.txt,1	scoring@aa.txt,1	
job@a1.txt,1	hiring@aa.txt,1	the@aa.txt,1	receive@aa.txt,1	for@aa.txt,1	
is@a1.txt,1	official@aa.txt,1	position@aa.txt,1	an@aa.txt,1	minimum@aa.txt,1	
posted@a1.txt,1	approves@aa.txt,1	and@aa.txt,1	alert@aa.txt,1	qualification@aa.txt,1	
on@a1.txt,1	a@aa.txt,1	how@aa.txt,1	when@aa.txt,1	testing@aa.txt,1	
a@a1.txt,1	requisition@aa.txt,1	the@aa.txt,1	new@aa.txt,1	hurdles@aa.txt,1	
specific@a1.txt,1	it@aa.txt,1	exam@aa.txt,1	jobs@aa.txt,1	and@aa.txt,1	
date@a1.txt,1	moves@aa.txt,1	plan@aa.txt,1	post@aa.txt,1	notes@aa.txt,1	
hr@a1.txt,1	to@aa.txt,1	is@aa.txt,1	4@aa.txt,1	in@aa.txt,1	
then@a1.txt,1	the@aa.txt,1	set@aa.txt,1	applications@aa.txt,1	the@aa.txt,1	
notifies@a1.txt,1	next@aa.txt,1	up@aa.txt,1	are@aa.txt,1	recruitment@aa.txt,1	
all@a1.txt,1	level@aa.txt,1	invitations@aa.txt,1	screened@aa.txt,1	folder@aa.txt,1	
subscribers@a1.txt,1	of@aa.txt,1	to@aa.txt,1	for@aa.txt,1	all@aa.txt,1	
on@a1.txt,1	approval@aa.txt,1	test@aa.txt,1	minimum@aa.txt,1	of@aa.txt,1	
a@a1.txt,1	hr@aa.txt,1	schedules@aa.txt,1	qualification@aa.txt,1	this@aa.txt,1	
rider@a1.txt,1	is@aa.txt,1	and@aa.txt,1	by@aa.txt,1	information@aa.txt,1	
alert@a1.txt,1	notified@aa.txt,1	notices@aa.txt,1	hr@aa.txt,1	resides@aa.txt,1	
system@a1.txt,1	automatically@aa.txt,1	are@aa.txt,1	generally@aa.txt,1	in@aa.txt,1	
that@a1.txt,1	at@aa.txt,1	sent@aa.txt,1	but@aa.txt,1	the@aa.txt,1	
a@a1.txt,1	each@aa.txt,1	out@aa.txt,1	this@aa.txt,1	tracking@aa.txt,1	
new@a1.txt,1	stage@aa.txt,1	by@aa.txt,1	will@aa.txt,1	system@aa.txt,1	
recruitment@a1.txt,1	of@aa.txt,1	the@aa.txt,1	be@aa.txt,1	3@aa.txt,1	
has@a1.txt,1	approval@aa.txt,1	system@aa.txt,1	done@aa.txt,1	the@aa.txt,1	
opened@a1.txt,1	or@aa.txt,1	via@aa.txt,1	via@aa.txt,1	job@aa.txt,1	
subscribers@a1.txt,1	denial@aa.txt,1	email@aa.txt,1	an@aa.txt,1	is@aa.txt,1	
set@a1.txt,1	2@aa.txt,1	and@aa.txt,1	auto@aa.txt,1	posted@aa.txt,1	
their@a1.txt,1	within@aa.txt,1	sms@aa.txt,1	scoring@aa.txt,1	on@aa.txt,1	
alert@a1.txt,1	the@aa.txt,1	cell@aa.txt,1	system@aa.txt,1	a@aa.txt,1	
preferences@a1.txt,1	application@aa.txt,1	phones@aa.txt,1	and@aa.txt,1	specific@aa.txt,1	

to@a1.txt,1	hr@aa.txt,1	using@aa.txt,1	based@aa.txt,1	date@aa.txt,1	
email@a1.txt,1	staff@aa.txt,1	template@aa.txt,1	on@aa.txt,1	hr@aa.txt,1	
and@a1.txt,1	uses@aa.txt,1	created@aa.txt,1	responses@aa.txt,1	then@aa.txt,1	
or@a1.txt,1	the@aa.txt,1	1@aa.txt,1	to@aa.txt,1	notifies@aa.txt,1	
cell@a1.txt,1	job@aa.txt,1	hr@aa.txt,1	application@aa.txt,1	all@aa.txt,1	
phone@a1.txt,1	description@aa.txt,1	logs@aa.txt,1	questions@aa.txt,1	subscribers@aa.txt,1	
this@a1.txt,1	or@aa.txt,1	into@aa.txt,1	those@aa.txt,1	on@aa.txt,1	
system@a1.txt,1	class@aa.txt,1	the@aa.txt,1	applications@aa.txt,1	a@aa.txt,1	
is@a1.txt,1	specification@aa.txt,1	online@aa.txt,1	passed@aa.txt,1	rider@aa.txt,1	
designed@a1.txt,1	to@aa.txt,1	hiring@aa.txt,1	the@aa.txt,1	alert@aa.txt,1	
for@a1.txt,1	create@aa.txt,1	center@aa.txt,1	minimum@aa.txt,1	system@aa.txt,1	
riders@a1.txt,1	a@aa.txt,1	to@aa.txt,1	qualifications@aa.txt,1	that@aa.txt,1	
of@a1.txt,1	job@aa.txt,1	generate@aa.txt,1	screening@aa.txt,1	a@aa.txt,1	
the@a1.txt,1	posting@aa.txt,1	a@aa.txt,1	are@aa.txt,1	new@aa.txt,1	
transit@a1.txt,1	write@aa.txt,1	requisition@aa.txt,1	reviewed@aa.txt,1	recruitment@aa.txt,1	
system@a1.txt,1	supplement@aa.txt,1	once@aa.txt,1	to@aa.txt,1	has@aa.txt,1	
but@a1.txt,1	questions@aa.txt,1	a@aa.txt,1	determine@aa.txt,1	opened@aa.txt,1	
has@a1.txt,1	and@aa.txt,1	position@aa.txt,1	whether@aa.txt,1	subscribers@aa.txt,1	
an@a1.txt,1	create@aa.txt,1	is@aa.txt,1	they@aa.txt,1	set@aa.txt,1	
opt@a1.txt,1	an@aa.txt,1	required@aa.txt,1	meet@aa.txt,1	their@aa.txt,1	
in@a1.txt,1	exam@aa.txt,1	the@aa.txt,1	entry@aa.txt,1	alert@aa.txt,1	
feature@a1.txt,1	plan@aa.txt,1	requisition@aa.txt,1	requirements@aa.txt,1	preferences@aa.txt,1	
people@a1.txt,1	this@aa.txt,1	is@aa.txt,1	through@aa.txt,1	to@aa.txt,1	
can@a1.txt,1	includes@aa.txt,1	routed@aa.txt,1	the@aa.txt,1	email@aa.txt,1	
use@a1.txt,1	setting@aa.txt,1	electronically@aa.txt,1	full@aa.txt,1	and@aa.txt,1	
to@a1.txt,1	up@aa.txt,1	for@aa.txt,1	exam@aa.txt,1	or@aa.txt,1	
ensure@a1.txt,1	auto@aa.txt,1	approval@aa.txt,1	process@aa.txt,1	cell@aa.txt,1	
they@a1.txt,1	scoring@aa.txt,1	using@aa.txt,1	by@aa.txt,1	phone@aa.txt,1	
receive@a1.txt,1	for@aa.txt,1	the@aa.txt,1	hr@aa.txt,1	this@aa.txt,1	
an@a1.txt,1	minimum@aa.txt,1	applicant@aa.txt,1	and@aa.txt,1	system@aa.txt,1	
alert@a1.txt,1	qualification@aa.txt,1	tracking@aa.txt,1	then@aa.txt,1	is@aa.txt,1	
when@a1.txt,1	testing@aa.txt,1	system@aa.txt,1	the@aa.txt,1	designed@aa.txt,1	
new@a1.txt,1	hurdles@aa.txt,1	hiring@aa.txt,1	hiring@aa.txt,1	for@aa.txt,1	

jobs@a1.txt,1	and@aa.txt,1	managers@aa.txt,1	officials@aa.txt,1	riders@aa.txt,1	
post@a1.txt,1	notes@aa.txt,1	log@aa.txt,1	5@aa.txt,1	of@aa.txt,1	
4@a1.txt,1	in@aa.txt,1	into@aa.txt,1	selected@aa.txt,1	the@aa.txt,1	
applications@a1.txt,1	the@aa.txt,1	a@aa.txt,1	applicants@aa.txt,1	transit@aa.txt,1	
are@a1.txt,1	recruitment@aa.txt,1	limited@aa.txt,1	will@aa.txt,1	system@aa.txt,1	
screened@a1.txt,1	folder@aa.txt,1	view@aa.txt,1	be@aa.txt,1	but@aa.txt,1	
for@a1.txt,1	all@aa.txt,1	version@aa.txt,1	invited@aa.txt,1	has@aa.txt,1	
minimum@a1.txt,1	of@aa.txt,1	and@aa.txt,1	for@aa.txt,1	an@aa.txt,1	
qualification@a1.txt,1	this@aa.txt,1	are@aa.txt,1	written@aa.txt,1	opt@aa.txt,1	
by@a1.txt,1	information@aa.txt,1	able@aa.txt,1	test@aa.txt,1	in@aa.txt,1	
hr@a1.txt,1	resides@aa.txt,1	to@aa.txt,1	and@aa.txt,1	feature@aa.txt,1	
generally@a1.txt,1	in@aa.txt,1	approve@aa.txt,1	interviews@aa.txt,1	people@aa.txt,1	
but@a1.txt,1	the@aa.txt,1	or@aa.txt,1	depending@aa.txt,1	can@aa.txt,1	
this@a1.txt,1	tracking@aa.txt,1	deny@aa.txt,1	on@aa.txt,1	use@aa.txt,1	
will@a1.txt,1	system@aa.txt,1	requisitions@aa.txt,1	the@aa.txt,1	to@aa.txt,1	
be@a1.txt,1	3@aa.txt,1	once@aa.txt,1	nature@aa.txt,1	ensure@aa.txt,1	
done@a1.txt,1	the@aa.txt,1	a@aa.txt,1	of@aa.txt,1	they@aa.txt,1	
via@a1.txt,1	job@aa.txt,1	hiring@aa.txt,1	the@aa.txt,1	receive@aa.txt,1	
an@a1.txt,1	is@aa.txt,1	official@aa.txt,1	position@aa.txt,1	an@aa.txt,1	
auto@a1.txt,1	posted@aa.txt,1	approves@aa.txt,1	and@aa.txt,1	alert@aa.txt,1	
scoring@a1.txt,1	on@aa.txt,1	a@aa.txt,1	how@aa.txt,1	when@aa.txt,1	
system@a1.txt,1	a@aa.txt,1	requisition@aa.txt,1	the@aa.txt,1	new@aa.txt,1	
and@a1.txt,1	specific@aa.txt,1	it@aa.txt,1	exam@aa.txt,1	jobs@aa.txt,1	
based@a1.txt,1	date@aa.txt,1	moves@aa.txt,1	plan@aa.txt,1	post@aa.txt,1	
on@a1.txt,1	hr@aa.txt,1	to@aa.txt,1	is@aa.txt,1	4@aa.txt,1	
responses@a1.txt,1	then@aa.txt,1	the@aa.txt,1	set@aa.txt,1	applications@aa.txt,1	

The aggregated vs non-aggregated information graphically shown in Figure-3, from the Figure-3, it is observed that non agreegation takes less time aggregation process, but aggregation gives most relevant results.

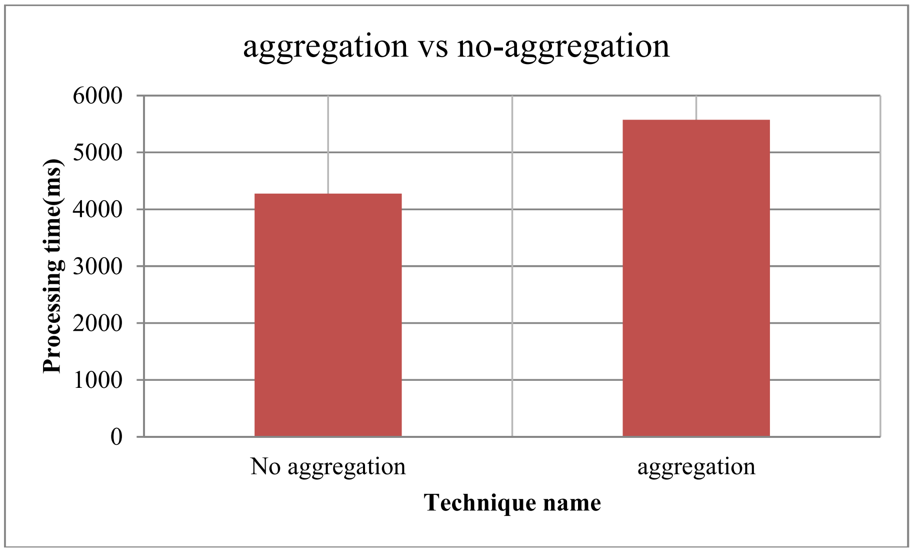

Figure-3. Performance of aggregation and no aggregation.

4. CONCLUSIONS

Information retrieval played a vital role in various applications like data science, data mining, DNA sequencing, etc.,, But deal with large volume of data, it face a trouble that is job is very large, in that situation, system will take much time some time returns irrelevant information. For a partial fulfillment, in this paper, we consider the large volume of data; convert it into small job using MapReduce programming. In this paper not only splitting the large volume of data into sub tasks but also aggregate the tasks and also reduce network traffic cost.

REFERENCES

[1] Prasad PS, Subrahmanyam HB, Singh PK. 2017. Scalable IQRA_IG Algorithm: An Iterative MapReduce Approach for Reduct Computation. Distributed Computing and Internet Technology. pp. 58-69.

[2] Ghazisaeedi E, Huang C. 2017. GreenMap: Green mapping of MapReduce-based virtual networks onto a data center network and managing in cast queueing delay. Computer Networks. pp. 345-359.

[3] Koh JL, Chen CC, Chan CY, Chen AL. 2017. MapReduce skyline query processing with partitioning and distributed dominance tests. Information Sciences. pp. 114-237.

[4] J. Dean and S. Ghemawat. 2008. Mapreduce: simplified data processing on large clusters. Communications of the ACM. 51(1): 107-113.

[5] W. Wang, K. Zhu, L. Ying, J. Tan and L. Zhang. 2013. Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality. IEEE Proceedings on INFOCOM. pp. 1609-1617.

[6] F. Chen, M. Kodialam and T. Lakshman. 2012. Joint scheduling of processing and shuffle phases in mapreduce systems. IEEE Proceedings on INFOCOM. pp. 1143-1151.

[7] Y. Wang, W. Wang, C. Ma, and D. Meng. 2013. Zput: A speedy data uploading approach for the hadoop distributed file system. IEEE International Conference onCluster Computing (CLUSTER). pp. 1-5.

[8] S. Chen and S. W. Schlosser. 2008. Mapreduce meets wider varieties of applications. Intel Research Pittsburgh. pp. 1-8.

[9] J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer, T. Condie and R. Ramakrishnan. 2013. Iterative mapreduce for large scale machine learning. pp. 1-9.

[10] Behm A, Borkar VR, Carey MJ, Grover R, Li C, Onose N, Vernica R, Deutsch A, Papakonstantinou Y, Tsotras VJ. 2011. Asterix: towards a scalable, semistructured data platform for evolving-world models. Distributed and Parallel Databases. pp. 185-216.

[11] S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung and R. S. Schreiber. 2013. Presto: distributed machine learning and graph processing with sparse matrices. Proceedings of the 8th ACM European Conference on Computer Systems. pp. 197-210.

[12] A. Matsunaga, M. Tsugawa and J. Fortes. 2008. Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. IEEE Fourth International Conference on eScience. pp. 222-229.

[13] J. Wang, D. Crawl, I. Altintas, K. Tzoumas and V. Markl. 2013. Comparison of distributed data-parallelization patterns for big data analysis: A bioinformatics case study. In Proceedings of the Fourth International Workshop on Data Intensive Computing in the Clouds (DataCloud). pp. 1-5.

[14] R. Liao, Y. Zhang, J. Guan and S. Zhou. 2014. Cloudnmf: A mapreduceimplementation of nonnegative matrix factorization for largescale biological datasets. Genomics, proteomics and bioinformatics. 12(1): 48-51.