# PRIVACY PRESERVING DYNAMIC WORK LOAD BALANCING BETWEEN DATA NOTES FOR HADOOP MAP REDUCE

**[1]M. Sai Surya Prasanna [2]Dr. I Hema Latha**
**[1] PG Scholar (M.tech), Department of information technology, Sagi Ramakrishnam Raju Engineering College, Bhimavaram A.P**
**[2]Associative Professor, Department of information technology, Sagi Ramakrishnam Raju Engineering College, Bhimavaram, A.P**

**ABSTRACT**
In Hadoop condition fundamentally we are performing has two segments in particular HDFS and MapReduce. Hadoop stores client information in view of room use of datanodes on the group as opposed to the handling capacity of the datanodes. Here Hadoop keeps running in half and half condition as all datanodes may not be same framework. Hence, workload irregular characteristics will happen when occupations keep running in a Hadoop group bringing about poor execution. So that here I propose a dynamic calculation to adjust the workload between various racks on a Hadoop group in view of the log documents of Hadoop. In any case, if the assignments are executing on basic or delicate information in a secured rack, the information exchange to an unsecured bunch will bring about protection being bargained. We propose a way to deal with exchange information between racks without revealing private data. Moving undertakings from the most over-burden rack to another rack enhances the execution of MapReduce employments. Our reenactments show that the proposed calculation diminishes running time of an occupation by over 50% running on the most over-burden rack.

**Keywords**: Hadoop, Dynamic Workload Balancing, Privacy, rack based data arrangement

## INTRODUCTION

Google MapReduce is a programming model and a product structure for Big - scale dispersed Computing on a lot of information. Figure 1 represents the abnormal state work stream of a MapReduce Task. Application designers determine the calculation as far as a guide and a decrease work, and the fundamental MapReduce Task planning framework consequently parallelizes the calculation over a group of machines. MapReduce acquire notoriety for its basic programming interface and amazing Performance while actualizing a huge range of utilizations. Since most such applications take a gigantic measure of info information, they are named as “Bigdata applications”.

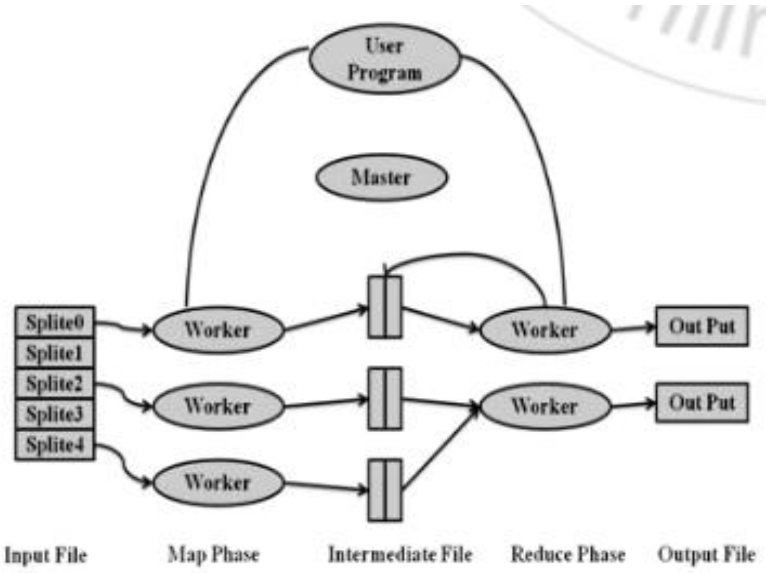


**Figure 1:** The MapReduce programming model architecture.

As shown in Figure 1, input information is first isolated and afterward given to laborers in the guide organize. Singular information things are called records. The MapReduce framework parses the information parts to every laborer and produces records. After the guide stage, middle of the road comes about produced in the guide stage are rearranged and arranged by the MapReduce framework and are then given into the laborers in the decrease stage. Last outcomes are processed by different reducers and composed back to the plate. Hadoop is an open-source execution of the Google MapReduce programming model. Hadoop comprises of the Hadoop Common, which gives access to the record frameworks bolstered by Hadoop. Hadoop Distributed File System (HDFS) gives circulated record stockpiling and is streamlined for expansive unchangeable blobs of information. A little Hadoop group will incorporate a solitary ace and various laborer hubs called as slave. The ace hub runs different procedures, including a TaskTracker and a Name Node. The TaskTracker is having control for overseeing running occupations in the Hadoop group. While the NameNode deals with the HDFS. The TaskTracker and the Name Node are typically gathered on the same physical machine. Different servers in the bunch run a Task Tracker and a Data Node forms.

A MapReduce work is isolated into errands. Assignments are overseen by the TaskTracker. The Task Trackers and the DataNode are examined on similar servers to give information area in calculation. MapReduce gives an institutionalized structure to actualizing huge scale disseminated calculation, called as, the enormous information applications. In any case, there is a limitation of the framework, i.e., the wastefulness in incremental preparing. Incremental preparing alludes to the applications that incrementally develop the information and constantly apply calculations on the contribution to request to create yield. There are potential copy calculations and operations being performed in this procedure. Be that as it may, MapReduce does not have the any method to recognize such copy calculations and quicken work execution. Inspired by this perception, In this paper we propose, an information mindful reserve framework for bigdata applications utilizing the MapReduce system, which goes for expanding the MapReduce structure and provisioning a store layer for effectively distinguishing and getting to store things in a MapReduce work.

Today a surge of information is being created from different sources, for example, Facebook or the New York Stock Exchange at the rate of a few TerraBytes (TB) or even PetaBytes (PB) consistently [9]. Conventional databases are not appropriate for huge information due to the volume and multifaceted nature of the information. Hadoop manages a lot of information, and it gives clients solid stockpiling. Hadoop Distributed File System (HDFS) and MapReduce display are key parts of Hadoop. Hadoop introduced on a substantial

bunch can traverse numerous datacenters including many racks. One issue with Hadoop is adjusting the workload. Each Hadoop rack will have distinctive execution since practically speaking the hubs running Hadoop are heterogeneous and the information put away on datanodes don't compare to the handling capacity of the hubs. Regardless of whether the Hadoop datanodes are homogeneous as imagined in the first form of Hadoop, the uneven dispersion of information to the datanodes brings about load lopsidedness between various racks bringing about lessened execution.

The Hadoop security display [11] gives information encryption and client and hub confirmation. A rack is said to be secure in our work if all hubs in a rack has been validated and information is encoded. Employments on basic or touchy information will occur on secure racks. Notwithstanding, stack adjusting may bring about information exchange to unsecured racks with loss of protection. Guaranteeing security of touchy information due to information exchange has not been contemplated with regards to stack adjusting in Hadoop. In this paper we propose a dynamic rack workload adjusting calculation to enhance the execution of Hadoop. Our proposed calculation will assess the execution of each rack and the current workload on each rack. In light of these parameters our calculation alters the undertaking distribution to each rack keeping in mind the end goal to abbreviate the running time of the employments which are running on the most over-burden rack and have quite a while to wrap up. To guarantee security we propose a record linkage conspire over different racks with the end goal that protection isn't traded off when information is exchanged.

## PROBLEM SPECIFICATION

At the point when clients compose a document to Hadoop it is separated into information squares and repeated over the Hadoop bunch. To keep the entire HDFS bunch adjusted, an instrument called balancer is utilized. Adjusting is done in light of plate usage and does not consider the preparing capacity of information hubs. At the point when a client presents work to Hadoop the jobtracker will part the information intelligently and find appropriate tasktrackers to execute the guide/lessen undertakings. The tasktrackers are on the datanode and send heartbeats to the jobtracker informing the status of errands. There are a sure number of spaces claimed by the tasktracker for the guide and diminish assignments. So when a tasktracker has a vacant space, it will educate the jobtracker about it. For the guide undertaking, the jobtracker will allocate the assignment to the tasktracker that is near the comparing information. For the diminish errand, the jobtracker just allots the undertaking from the rundown of yet-to-be-run decrease assignments to the following tasktracker that is prepared.

## LITERATURE REVIEW

1. Large-scale Incremental Processing Using Distributed Transactions and Notifications [3] Daniel Peng et al. proposed, a framework for incrementally handling updates to a substantial informational index, and conveyed it to make the Google web seek file. By supplanting a batchbased ordering framework with an ordering framework in view of incremental handling utilizing Percolator, Auther process a similar number of records every day.
2. Design and Evaluation of Network-Leviated Merge for Hadoop Acceleration [7] Weikuan Yu et al. proposed, Hadoop-An, a quickening system that advances Hadoop with module parts for quick information development, defeating the current confinements. A novel system suspended union calculation is acquainted with consolidate information without redundancy and circle get to. Moreover, a full pipeline is intended to cover the rearrange, consolidate and lessen stages. Our test comes about demonstrate that Hadoop-An altogether accelerates information development in MapReduce and pairs the throughput of Hadoop.
3. Improving Mapreduce Performance through Data Placement in Heterogeneous Hadoop Cluster [5] Jiong Xie et al. suggested that overlooking the information region issue in heterogeneous situations can discernibly decrease the MapReduce execution. In this paper, creator tends to the issue of how to put information crosswise over hubs in a way that every hub has an adjusted information handling load. Given an information escalated application running on a Hadoop MapReduce group, our information position conspire adaptively balances the measure of information put away in every hub to accomplish enhanced

information handling execution. Test comes about on two genuine information serious applications demonstrate that our information situation system can simply enhance the MapReduce execution by rebalancing information crosswise over hubs previously playing out an information escalated application in a heterogeneous Hadoop bunch.

4. Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization [6] Zhenhua Guo et al. proposed, Benefit Aware Speculative Execution which predicts the advantage of propelling new theoretical errands and enormously wipes out pointless keeps running of theoretical assignments. At long last, MapReduce is for the most part enhanced for homogeneous conditions and its wastefulness in heterogeneous system situations has been seen in their tests. Creators research organize heterogeneity mindful planning of both guide and diminish errands. By and large, the objective is to upgrade Hadoop to adapt to huge framework heterogeneity and enhance asset use.

## SYSTEM ARCHITECTURE

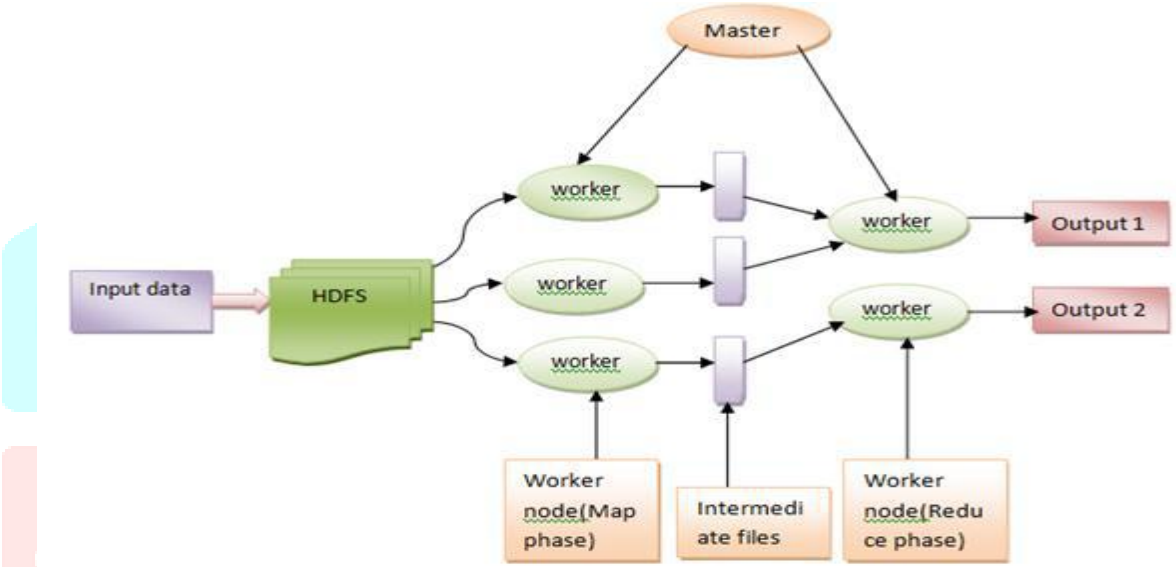


Fig: Execution Overview

## IMPLEMENTATION

To look at the execution of calculations, we need to consider three fundamental factors to be specific region, reasonableness and synchronization. Region is the separation between input information hub and assignment assigned hub. Reasonableness is the exchange offs between the area and reliance between the maps and lessens stages. Synchronization is the way toward transmitting the halfway yield of the guide procedures to the lessen forms as info. The assignment planning exercises is specifically influencing the framework enhancement of Hadoop and framework assets use. A few methodologies have been endeavored to give arrangements. Every calculation will have its own particular upsides and downsides.

## PREDICTING THE NUMBER OF TASKS THAT WILL RUN ON A RACK

On the off chance that a vocation isn't ended or finish, the errands of that activity will have three states: completed the process of, running and holding up. From the log of the jobtracker and tasktracker, we can get the quantity of running and completed undertakings having a place with a vocation on a specific rack. The holding up undertakings have not been relegated to the tasktracker yet and we have to anticipate what number of them will be allocated to a rack.

The jobtracker log contains errand data, for example, undertaking id and area of info split. At the point when the errand is to be doled out, the jobtracker first picks a tasktracker to run this assignment and the tasktracker id will be recorded to the log. The jobtracker records the kind of the undertaking (data-local or a rack-local). In an information nearby assignment the information split of the errand is keep running on the datanode handling this undertaking, while in a rack neighborhood assignment the info split of an errand isn't on the datanode that procedures this errand however on a similar rack. On the off chance that this errand is done, the jobtracker log records that the undertaking has finished effectively. By investigating the log of the jobtracker, we can get the likelihood of datalocal or rack-nearby assignments. That is, t_(d-l)/T or t_(r-l)/T where td-1 is the quantity of information neighborhood undertakings, tr-1 is the quantity of rack-nearby assignments and T is the aggregate number of errands.

In our calculation, we expect that every one of the undertakings are either information neighborhood or rack-nearby. In HDFS, each datum square has three reproductions as a matter of course. The namenode knows where these copies are set. Subsequently when the jobtracker instates an occupation, it will get the information split areas from the namenode and record it to the log. We can get the circulation of all the information parts having a place with an occupation from the log of the jobtracker. We can likewise get from the log the quantity of assignments that have a place with a specific occupation that have the information split on a rack.

By examining the circulation of all the info information split having a place with a vocation on a rack, we compute the quantity of undertakings that will keep running on a rack in light of information area.

$$N_{wating} = \frac{P_d}{N_{replica}} * ( N_A + N_B) + \frac{p_r}{2} * (N_A + \frac{N_B}{2} ) \qquad (1)$$

Here $N_{wating}$ refers to the number of tasks of a job waiting to run on the rack, $P_d$ refers to the probability of a data-local task and _ refers to the probability of a rack-local task. $N_{replica}$ refers to the number of replicas of a data block
in Hadoop, which by default is three; $N_A$ is the number of input split of the tasks that is in the first replica,$N_B$ is the
number of input split of the tasks that is in the second or third replica. If a data node has the input split of a task, the probability of that data node running this task as a data-local type is$\frac{P_d}{N_{replica}}$ and the number of this kind of datanodes in a rack is ( $N_A + N_B$ The input split of a task has replicas on two racks. The probability of running this task as rack-local type is $\frac{p_r}{2}$ . The number of tasks whose input split on this rack is $N_A + \frac{N_B}{2}$

**Predicting remaining time of tasks belonging to a job**

Here decide the time utilized by completed errands on a rack, particularly the most over-burden rack. The logs of the jobtracker and tasktracker record the begin time of the main assignment and the fulfillment time of the last errand. The contrast between these two gives a gauge of the execution of a rack. The rest of the season of a vocation is anticipated as takes after:

$$T_{remaining} = N_{waiting} * \frac{t}{f} (2)$$

where $T_{remaining}$ refers to the predicted remaining time to finish all the tasks of a job on that rack, ( refers to the number of tasks completed belonging to a job on that rack, and ) refers to the time used by these f tasks.

**Rack selection**

The ideal circumstance for an occupation is that all the guide assignments finish in the meantime. In this circumstance the quantity of errands allocated to a rack coordinates the execution of that rack. Undertaking task

depends exclusively on input split area and henceforth in our technique we change the info split area to adjust the workload and decline execution time.

The ratio $\frac{t}{f}$ the denotes the capability of a rack. The number of tasks to be assigned to that rack based on its capability can be estimated by formula 3.

$$N_{assign} = N_{total\_waiting} * \frac{\frac{f_k}{t_k}}{\sum_{i=1}^{n} \frac{f_i}{t_i}} \qquad (3)$$

Where n refers to the number of racks , i refers to the $i^{th}$ rack, $N_{assign}$ refers to the number of waiting tasks that should
be assigned to the $K^{th}$ rack, $N_{total\_waiting}$ refers to the total number of waiting tasks of a job, $f_i$ refers to the amount of finished tasks belonging to a job on the $i^{th}$ rack, $t_i$ refers to the time used by these $f_i$ tasks.

We can predict the number of tasks to be assigned to a rack by eq. (1). For each rack, the difference between the
numbers of tasks that can run based on the capability of a rack and actual number of tasks that will run on the rack is:

$$\Delta N_i = N^i_{assign} - N^i_{waiting} \qquad (4)$$

Where i refers to $i^{th}$ rack, $N^i_{assign}$ refers to the number of tasks that can be assigned to the $i^{th}$ rack based on its capability, $N^i_{waiting}$ refers to the number of tasks that are waiting and will actually run on the $i^{th}$ rack.

A task should be moved only when moving the task to a new rack saves time. This saved time should be larger than transfer time. This is expressed as an inequality in formula 5.

$$\frac{t_k}{f_k} - \frac{t_i}{f_i} > \frac{size\ of\ input\ split}{bandwidt\ h_{ki}} \qquad (5)$$

Where the $K^{th}$ rack is the overloaded rack, $f_i(f_k)$ refers to the number of finished tasks belonging to a job on the ith ($k^{th}$) rack, $t_i$ ($t_k$) refers to the time used by these $f_i$ ($f_k$) tasks, $bandwidth_{ki}$ refers to the bandwidth of the link between the $k^{th}$ and $i^{th}$ racks.

Select the racks whose $\Delta N_i$ are positive. For these racks check if inequality (5) holds and select the racks for which inequality (5) holds. These are the racks to which tasks should be transferred. Dynamic Racks Workload Balancing algorithm Using the above model, the rack-based load balancing algorithm is shown below:

**DYNAMIC RACKS WORKLOAD BALANCING ALGORITHM**

**Input:** all racks and all jobs on the Hadoop cluster
**Begin:**
For each job:
Estimate the number of waiting tasks to run on the $K^{th}$ rack and their remaining time by eq. (1), (2)
Based on the capability and workload of each rack,

Compute the number of tasks to be assigned to each rack
Select the racks, which should get more tasks based on their capability and workload
Select the racks that hold true for inequality (5)
Determine the number of tasks to be moved to each rack
And transfer the data by eq. (3), (4), (6).
End.

## PRIVACY AND SECURITY CONCERNS IN BIG DATA

### Privacy and security concerns

Privacy and security as far as large information is a critical issue. Enormous information security display isn't proposed in case of complex applications because of which it gets handicapped as a matter of course. Be that as it may, in its nonappearance, information can simply be traded off effectively. Accordingly, this area concentrates on the protection and security issues.

**Privacy:** Information privacy is the benefit to have some control over how the individual data is gathered and utilized. Data protection is the limit of an individual or gathering to prevent data about them from getting to be noticeably known to individuals other than those they give the data to. One genuine client protection issue is the recognizable proof of individual data amid transmission over the Internet.

**Security:** Security is the act of protecting data and data resources using innovation, procedures and preparing from:- Unauthorized access, Disclosure, Disruption, Modification, Inspection, Recording, and Destruction.

**Privacy *vs.* security*:*** Data privacy is centered around the utilization and administration of individual information things like setting up approaches set up to guarantee that shoppers' close to home data is being gathered, shared and used in fitting ways. Security focuses more on shielding information from pernicious assaults and the abuse of stolen information for benefit. While security is basic for ensuring information, it's not adequate for tending to protection. Beneath table concentrates on extra distinction amongst protection and security.

| S.NO | Privacy | Security |
|---|---|---|
| 1 | Privacy is the appropriate use of user's information | Security is the "confidentiality, integrity and availability" of data |
| 2 | Privacy is the ability to decide what information of an individual goes where | Security offer the ability to be confident that decisions are respected |
| 3 | The issue of privacy is one that often applies to a consumer's right to safeguard their information from any other parties | Security may provide for confidentiality. The overall goal of most Security system is to protect an enterprise or agency [72] |
| 4 | It is possible to have poor privacy and good security practices | However, it is difficult to have good privacy practices without a good data security program |

| | | |
|---|---|---|
| 5 | For example, if user make a purchase from XYZ company and provide them payment [13] and address information in order for them to setup the product, they cannot then sell user's information to a third party without prior consent to user | The company XYZ uses various techniques (Encryption, Firewall) in order to prevent data compromise from technology or vulnerabilities in the network |

**BIG DATA PRIVACY IN DATA GENERATION PHASE**

Information age can be ordered into dynamic information age and detached information age. By dynamic information age, we imply that the information proprietor will give the information to an outsider [17], while latent information age alludes to the conditions that the information are delivered by information proprietor's online activities (e.g., perusing) and the information proprietor may not think about that the information are being assembled by an outsider. Minimization of the danger of protection infringement in the midst of information age by either confining the entrance or by adulterating information.

***Access restriction*** On the off chance that the information proprietor conceives that the information may reveal touchy data which should be shared, it declines to give such information. In the event that the information proprietor is giving the information latently, a couple of measures could be taken to guarantee protection, for example, hostile to following expansions, ad or content blockers and encryption devices.

***Falsifying data*** In a few conditions, it is improbable to neutralize access of delicate information. All things considered, information can be contorted utilizing certain instruments before the information gotten by some outsider. In the event that the information are mutilated, the genuine data can't be effectively uncovered. The accompanying systems are used by the information proprietor to adulterate the information.

An apparatus Socket puppet is used to cover up online character of individual by double dealing. By using numerous Socket puppets, the information having a place with one particular individual will be viewed as having a place with different individuals. In that way the information authority won't have enough learning to relate distinctive socket puppets to one person

Certain security instruments can be utilized to cover person's character, for example, Mask Me. This is particularly helpful when the information proprietor needs to give the Visa points of interest in the midst of web based shopping.

**BIG DATA PRIVACY IN DATA STORAGE PHASE**

Putting away high volume information isn't a noteworthy test because of the progression in information stockpiling innovations, for instance, the blast in distributed computing. In the event that the huge information stockpiling framework is traded off, it can be outstandingly dangerous as people close to home data can be unveiled. In appropriated condition, an application may require a few datasets from different server farms and along these lines stand up to the test of security insurance.

The ordinary security systems to ensure information can be isolated into four classifications. They are record level information security plans, database level information security plans, media level security plans and application level encryption plans. Reacting to the 3V's idea of the huge information investigation, the capacity foundation should be versatile. It ought to be able to be arranged progressively to suit different applications. One promising innovation to address these prerequisites is capacity virtualization, engaged by the developing distributed computing worldview. Capacity virtualization is process in which various system stockpiling gadgets are joined into what emits an impression of being a solitary stockpiling gadget. SecCloud is one of the models for information security in the cloud that mutually considers both of information stockpiling security and calculation evaluating security in the cloud. Along these lines, there is a constrained exchange if there should be an occurrence of security of information when put away on cloud.

## INTEGRITY VERIFICATION OF BIG DATA STORAGE

Right when distributed computing is utilized for enormous information stockpiling, information proprietor loses control over information. The outsourced information is in danger as cloud server may not be totally trusted. The information proprietor ought to be solidly persuaded that the cloud is putting away information legitimately as per the administration level contract. To guarantee protection to the cloud client is to give the framework the component to permit information proprietor confirm that his information put away on the cloud is in place. The uprightness of information stockpiling in conventional frameworks can be confirmed through number of ways i.e., Reed-Solomon code, checksums, trapdoor hash capacities, message authentication code (MAC), and advanced marks and so forth. Along these lines information trustworthiness check is of basic significance. It thinks about various respectability confirmation plans examined. To check the respectability of the information put away on cloud, straight forward approach is to recover every one of the information from the cloud. To confirm the uprightness of information without retrieving the information from cloud. In respectability check conspire, the cloud server can just give the generous proof of trustworthiness of information when every one of the information are in place. It is exceedingly endorsed that the uprightness check ought to be led routinely to give most elevated amount of information security.

**Input:** Party RA holds dataset $D_A$ of m records A[1],A[2],…A[i],…,A[m], $1 \leq i \leq m$. Each A[i] contains $a_i[1],a_i[2],\ldots,a_i[k],\ldots a_i[r]$ data. $D_A$ have r number of attributes and $1 \leq k \leq r$. Rack RB holds database $D_B$ of n records, B[1],B[2],…,B[i],…B[n], $1 \leq i \leq n$. Each B[i] contain $b_i[1],b_i[2],\ldots,b_i[j],\ldots b_i[s]$ and $D_B$ have s number of attributes and $1 \leq j \leq s$. Here $m \neq n$ and $r \neq s$.

**Objective:** Rack RB is locking for a data w of record A[i] where w€ $D_B$ and w $\epsilon D_A$ and w may or may not be sensitive. We assume party RA agrees to share w with party RB.

**Output:** Party RA sends data w to party RB without disclosing any other information. Party RB does not disclose any information to rack RA.

## PRIVACY PRESERVING ALGORITHM

1. Rack RB generates public key-private key pair(x,y). party RB creates a significant list of attributes L of length l from s number of available attributes, $l \leq s$.
2. Party RB encrypts all the data $b_i[attr_k]$ corresponding to attributes of L with x.Here attr $1 \leq k \leq l$.
3. Rack RB send
   $E_x$ ($b_i[attr_1]$, $Ex(b_i[attr_2])$,……,$Ex(b_i[attr_i]$,x,L,attr[w]To RA. Attr[w] is the attribute of data w that RB is looking for and we assume attr[w] $\epsilon$ r.
4. For I = 1 to m
5. For j = 1 to q where q = l and all the elements of q $\epsilon$ l
6. Rack RA encrypts $a_i[j]$. $Ex(a_i[j])$ denotes encrypt $a_i[j]$.
7. End f or
8. Party RA calclutes jaccard Similarity Coefficient J between $E_x(b_i[attr_1]$, $E_x(b_i[attr_2],\ldots,E_x(b_i[attr_i]$ and $E_x(a_i[1]),Ex(a_i[2],\ldots,E_x(a_i[q])$
9. If J = 1 or J < T
10. If ai[attr[w]] is sensitive
11. Encrypt the data ai[attr[w]]) to rack RB.
12. Else
13. Send the data ai[attr[w]] to race RB
14. End if
15. End if

## RESULT ANALYSYS

Expected Results Figure 5: Execution time of the framework Fig. 5 demonstrates that DRAWs regrouped information and Hadoops arbitrarily put information. The quantity of reducers are utilized with the goal that the diminish stage won't deliver bottleneck. DRAW completed guide stage almost around 30% sooner than the default set information, and the assignments general execution time is additionally 25% better by utilizing DRAW. In proposed framework information is added at the information record. The span of the attached information differs and is spoken to as a rate number to the first information record measure, which is in GBs. Thus Dache can keep away from calculation undertakings that take additional time, which accomplishes more speedups. Dache can finish occupations 20% quicker than Hadoop in all circumstances. It demonstrates that proposed framework sets aside less time for preparing as contrast with existing framework. In proposed framework CPU usage proportion of program is computed by averaging the CPU use proportion of MapReduce work preparing time. Hadoop 30% takes more CPU cycles than Dache, which is normal by the CPUbound idea of the execution method. Unmistakably Dache spares a noteworthy measure of CPU cycles, which is demonstrated by the much lower CPU use proportion.

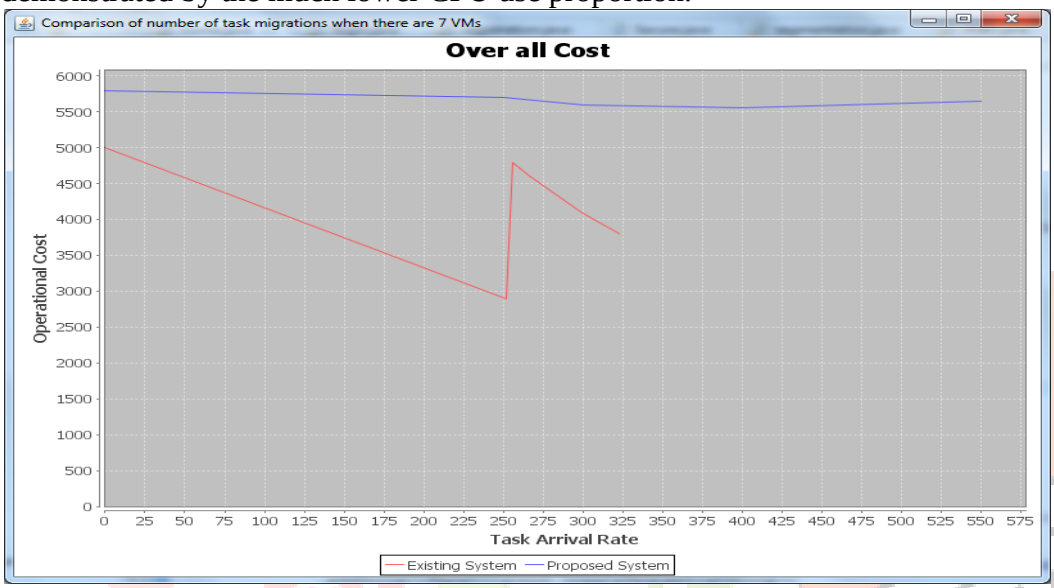


**Figure:** Execution time of the system

## CONCLUSION

The above work, addresses the intricacy of load adjusting by apportioning the cloud. The divided cloud is overseen by the Partition Controller and Job Balancer. The work visualizes the open source Hadoop structure to deal with and control information. It utilizes MapReduce programming model to deal with parallel handling of information. Significance of Hadoop's HDFS is additionally featured keeping in mind the end goal to store information. A near investigation of MapReduce calculations is made. The decision of the calculation relies upon the territory, decency and the measure of the group. The future work will be to move in making a bunch and timetable the errands utilizing the fitting MapReduce algorithm.

## REFERENCES

[1] K. Shvachko, H. Kuang, S. Radia, R. Chansler, "The Hadoop Distributed File System", in Proceedings of IEEE Conference on Mass Storage Systems and Technologies (MSST), 2010.

[2] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", in Proceedings of Sixth Symposium on Operating System Design and Implementation (OSDI'04), San Francisco, CA, December 2004.

[3] M.Zaharia, A.Konwinski, A.Joseph, Y.zatz, and I.Stoica. Improving mapreduce performance in heterogeneous environments. In OSDI'08: 8th USENIX Symposium on Operating Systems Design and Implementation, October 2008.
[4] M.Rafique, B.Rose, A.Butt, and D.Nikolopoulos. Supporting mapreduce on large-scale asymmetric multi-core clusters. SIGOPS Oper. Syst. Rev., 43(2):25-34, 2009.
[5] Tian, C., Zhou, H., He, Y., & Zha, L. (2009, August). A dynamic mapreduce scheduler for heterogeneous workloads. In Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on (pp. 218-224). IEEE..
[6] J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, A. Manzanares, and X. Qin. Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters, Proc. 19th Int'l Heterogeneity in Computing Workshop, Atlanta, Georgia, April 2010.
[7] T. White. Hadoop: The Definitive Guide. O'Reilly Media, Inc, 2010
[8] Hammoud, Suhel, Maozhen Li, Yang Liu, Nasullah Khalid Alham, and Zelong Liu. "MRSim: A discrete event based MapReduce simulator." In *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, vol. 6, pp. 2993-2997. IEEE, 2010.
[9] Kumar, T. K., Kim, J., George, K. M., & Park, N. (2014, June). Dynamic data rebalancing in Hadoop. In Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on (pp. 315-320).IEEE.
[10] Hou, Xiaofei, T. K. Ashwin Kumar, Johnson P. Thomas, and Vijay Varadharajan. "Dynamic Workload Balancing for Hadoop MapReduce." In Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on, pp. 56-62. IEEE, 2014.
[11] Hadoop in Secure Mode, https://hadoop.apache.org /docs/current/hadoop-project-dist/hadoop-common/SecureMode.html #Authentication [last accessed 11-15-2015]
[12] Deza, Michel Marie, and Elena Deza. Encyclopedia of distances. Springer Berlin Heidelberg, 2009.
[13] Rivest, Ronald L., Adi Shamir, and Len Adleman. "A method for obtaining digital signatures and public-key cryptosystems." Communications of the ACM 21, no. 2 (1978): 120-126.
[14] Paillier, Pascal., "Public-key cryptosystems based on composite degree residuosity classes", Advances in cryptology, EUROCRYPT99, Springer Berlin Heidelberg, 1999.
[15] Yao, Andrew Chi-Chih, "How to generate and exchange secrets", Foundations of Computer Science, IEEE 27th Annual Symposium, 1986.
[16] Want, Roy, "Near field communication.", IEEE Pervasive Computing 3, pp 4-7, 2011.
[17] Agrawal, Rakesh., Kiernan, Jerry., Ramakrishnan, Srikant., and Xu, Yirong., "Order preserving encryption for numeric data", in Proceedings 2004 ACM SIGMOD international conference on Management of data, pp. 563-574. ACM, 2004.
[18] Goldreich, Oded., and Yair Oren, Yair., "Definitions and properties of zero-knowledge proof systems." Journal of Cryptology, Vol 7, No. 1, pp 1-32, 1994.
[19] Ogataa, Wakaha and Kurosawab, Kaoru, "Oblivious keyword search" Journal of Complexity, Vol. 20, Issues 2–3, pp. 356–371, April–June 2004,
[20] R. Curtmola, R.., Garay, J., Kamara, S., and Ostrovsky, R., "Searchable symmetric encryption: Improved definitions and efficient constructions", In ACM Conference on Computer and Communications Security (CCS '06), pages 79–88. ACM, 2006
[21] Chor, B., Kushilevitz, E., Goldreich, O., and Sudan, M., "Private Information Retrieval", Journal of the ACM, 45, 1998.