Privacy Preservation in Analyzing E-Health Records in Big Data Environment

E. Srimathi
Computer Science and Engineering
K. Ramakrishnan College of Technology
Trichy, TamilNadu, India.
srimathielango@gmail.com

K. A. Apoorva
Computer Science and Engineering
K. Ramakrishnan College of Technology
Trichy, TamilNadu, India.
apoorva.krct@gmail.com

Abstract – Increased use of the Internet and progress in Cloud computing creates a large new datasets with increasing value to business. Data need to be processed by cloud applications are emerging much faster than the computing power. Hadoop-MapReduce has become powerful computation model to address these problems. Nowadays many cloud services require users to share their confidential data like electronic health records for research analysis or data mining, which brings privacy concerns. K-anonymity is one of the widely used privacy model. The scale of data in cloud applications rises extremely in agreement with the Big Data tendency, thereby creating it a dispute for conventional software tools to process such large scale data within an endurable lapsed time. As a consequence, it is a dispute for current anonymization techniques to preserve privacy on confidential extensible data sets due to their inadequacy of scalability. In this project, we propose an extensible two-phase approach to anonymize scalable data sets using dynamic MapReduce framework, Top Down Specialization (TDS) Algorithm and k-Anonymity privacy model. The resources are optimized via three key aspects. First, the under-utilization of map and reduce tasks is improved based on Dynamic Hadoop Slot Allocation (DHSA). Second, the performance tradeoff between the single job and a batch of jobs is balanced using the Speculative Execution Performance Balancing (SEPB). Third, data locality can be improved without any impact on fairness using Slot Pre Scheduling. Experimental evaluation results demonstrate that with this project, the scalability, efficiency and privacy of data sets can be significantly improved over existing approaches.

Keywords – *BigData; Data Anonymization; k – Anonymity; Top Down Specialization; MapReduce.*

*****

I. INTRODUCTION

Cloud computing is becoming more prominent as time goes by, and technology is changing our routine lifestyle. The raise of cloud and data stores has been a facilitator and precursor to the gush of big data [3], [6], [7], [11], [15]. Many healthcare organizations are planning to utilize the latest technologies to enhance healthcare services. Gaining access to high-quality health data is a vital requirement to informed decision making for medical practitioners and pharmaceutical researchers. Cloud computing opened a window of opportunity for healthcare organizations to share their data with other stakeholders such as government agencies, authorized private companies such as insurance companies and other hospitals.

Sharing patients' data serves different purposes that contribute to improve the quality of healthcare services [20]. Yet this sharing needs to have strict regulations on who is sharing the data and how well the privacy of the patients is maintained [12]. When dealing with privacy and sharing information several threats are involved [14]. The top threats include social functions where, although users can choose to be anonymous, they could easily and involuntarily expose their identity or personal information.

Many agencies and institutes consider that the released data is privacy-preserved if explicit identifying information, such as name, social security number, address, and telephone number, are removed. However, substantial research has shown that simply removing explicit identifying information is insufficient for privacy protection. An individual can be re identified by simply matching other attributes, called quasi-identifiers (QID), such as gender, date of birth, and postal code.

Many privacy models, such as *K*-anonymity and its extensions, have been proposed to thwart privacy threats caused by identity and attribute linkages in the context of relational databases. However this privacy models couldn't effectively preserve unstructured data in a big data trend

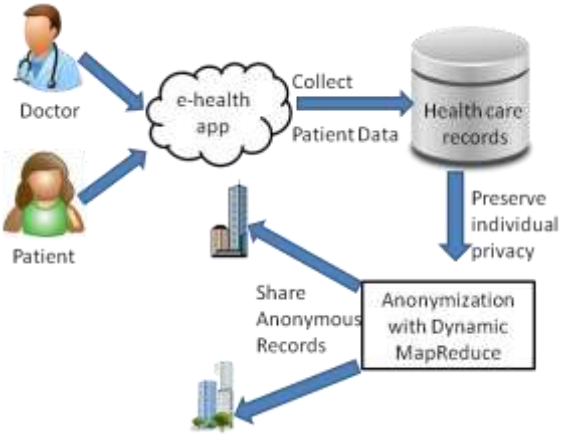

Figure 1: Overview of preserving individual privacy

Figure 1 illustrates the overview of the securing individual identity. K-anonymity model is implemented in

Dynamic MapReduce framework for preserving the individual privacy. The performance of a MapReduce cluster can be improved by optimizing the slot utilization – the DynamicMR approach.

II. RELATED WORK

Kristen LeFevre et al. addressed [10] the scalability of anonymization algorithm using Rothko-Tree (Rothko-T) algorithm based on Rainforest Scalable Decision Tree technique and Rothko-Sampling (Rothko-S) algorithm based on Sampling technique. They implemented k-anonymity and l-diversity with multidimensional generalization technique for privacy preservation, Thereby failed to work in Top down Specialization approach and hence it will not handle data exploration problem. The multidimensional generalization technique does not support scalability also.

Indrajit Roy et al. investigated [13] the privacy issues in MapReduce and presented a paper named Airavat which integrates Mandatory Access Control (MAC) and Differential Privacy in SELinux environment. Airavat cannot guarantee privacy for computations which output keys produced by untrusted mappers and the privacy can be achieved by requiring the computation provider to declare the key in advance. Our research exploits MapReduce itself to anonymize large scale data sets before data are further processed by other MapReduce jobs, privacy is achieved. Zhang et al. [19] automatically partition a computing job in terms of data security levels through MapReduce.

Benjamin C.M. Fung et al. proposed [12] centralized and distributed algorithms by adopting Top down specialization to achieve LKC-privacy model. Taxonomy Indexed Partions (TIPS) data structure is exploited to improve the efficiency of Top Down Specialization. But the approach is centralized, leading to its inadequacy in handling large-scale data sets. In addition, LKC privacy does not support transactional and textual data.

Guo et al. proposed [9] a resource stealing method to enable running tasks to steal resources reserved for idle slots and give them back proportionally whenever new tasks are assigned, by adopting multithreading technique for running tasks on multiple CPU cores. However, it cannot work for the utilization improvement of those purely idle slave nodes without any running tasks. They further proposed Benefit Aware Speculative Execution (BASE) algorithm that can evaluate the potential benefit of speculative tasks and eliminate unnecessary runs.

Zaharia.M et al. introduced [17] Longest Approximate Time to End (LATE), a speculative execution algorithm that focuses on heterogeneous environments by prioritizing tasks to speculate, selecting fast nodes to run on, and capping speculative tasks.When LATE detects a straggled task and an idle slot, it first cheks the number of running speculative tasks. When it is smaller than SpeculativeCap, it will immediately create a speculative task for straggled task and run it immediately. The resource allocated to the slot are being used inefficiently, hence the efficiency of the cluster is reduced.

Zaharia.M et al. achieved locality and fairness [18] in cluster scheduling using delay scheduling. Delay scheduler can improve the data locality by delaying the scheduling of map tasks whose data locality cannot be satisfies for a short period of time, at the expense of fairness.

III. PRELIMINARY

A. Anonymization

The Privacy Preserving Data Publishing (PPDP) scenario involves two phases. In the data collection phase, the data publisher collects data from record owners. (eg. cloud application) In the data publishing phase, the data publisher releases the collected data to a data miner or to the public, called the data recipient, who will then conduct data mining on the published data. In the most basic form, the data publisher has a table of the form.

D(Explicit Identifier, Quasi Identifier, Sensitive Attributes, Non-Sensitive Attributes),

where Explicit Identifier is a set of attributes, such as name and social security number (SSN), containing information that explicitly identifies record owners; Quasi Identifier (QID) is a set of attributes that could potentially identify record owners; Sensitive Attributes consists of sensitive person-specific information such as disease, salary, and disability status; and Non-Sensitive Attributes contains all attributes that do not fall into the previous three categories. The four sets of attributes are disjoint. For more explanation refer [8].

Sweeney referred "*Anonymization* to the PPDP approach that seeks to hide the identity and/or the sensitive data of record owners, assuming that sensitive data must be retained for data analysis".

B. Top Down Specialization (TDS)

Top-Down Specialization method generalizes a table by specializing it from the most general state in which all values are generalized to the most general values of their taxonomy trees. Each round of iteration consists of three main steps, namely, finding best specialization, performing specialization and updating values of the search metric for the next round [8]. At each step, TDS selects the

specialization according to the search metric Information Gain per Privacy Loss (*IGPL)* which is calculated in (1).

A specialization with the highest IGPL value is regarded as the best one and selected in each round. Suppose that the anonymous table is searched by iteratively specializing a general value into child values. Each specialization operation splits each group containing the general value into a number of groups, one for each child value. Each specialization operation *s* gains some information, denoted *IG*(*s*), and loses some privacy, *PL*(*s*). This search metric prefers the specialization *s* that maximizes the information gained per each loss of privacy:

$$IGPL(s) = \frac{IG(s)}{PL(s)+1} \quad (1)$$

The choice of *IG*(*s*) and *PL*(*s*) depends on the information metric and privacy model. Interested readers can refer to [8] for further clarification. The specialization process terminates if no specialization can be performed without violating *k*-anonymity. The data on termination is a minimal *k*-anonymization according to the generalization property.

C. MapReduce

Hadoop MapReduce is a software framework for writing applications easily that process ample amounts of data in-parallel on huge clusters of commodity hardware in a fault-tolerant, reliable manner [4].

"A MapReduce job partitions the input data set into independent chunks which are executed by the map tasks concurrently. The framework sorts the outputs of the map tasks and fed into the reduce tasks. Typically the file system stores both the input and the output of the job. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks". Cloud computing combined with the MapReduce becomes more powerful and elastic as cloud can offer infrastructure resources on demand [21].

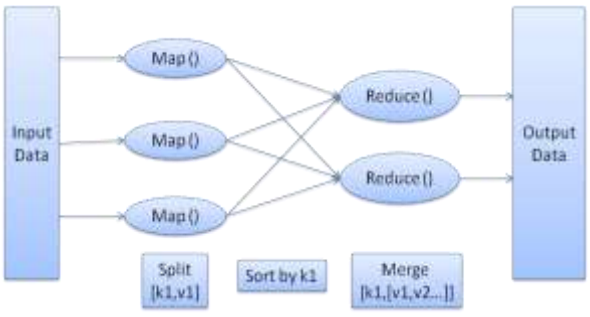

Figure 2: MapReduce Image

The storage nodes and the compute nodes are typically same i.e., the HDFS and MapReduce framework are running in the same set of nodes. The MapReduce framework includes a single master node named JobTracker and one slave per cluster node named TaskTracker. The master node is responsible for scheduling the job's tasks on the slaves, monitor and re-execute the failed tasks. The master node will direct the slave nodes to execute the tasks. Figure 2 describes the MapReduce computation in brief.

D. Two-phase Top Down Specialization

The two phases of Top down Specialization are based on the two levels of parallelization provisioned by MapReduce on cloud. Basically, MapReduce on cloud has two levels of parallelization, i.e., job level and task level. Job level parallelization means that multiple MapReduce jobs can be executed simultaneously to make full use of cloud infrastructure resources.

Combined with cloud, MapReduce becomes more powerful and elastic as cloud can offer infrastructure resources on demand. Task level parallelization refers to that multiple mapper/reducer tasks in a MapReduce job are executed simultaneously over data splits. To achieve high scalability, we parallelizing multiple jobs on data partitions in the first phase, but the resultant anonymization levels are not identical. To obtain finally consistent anonymous data sets, the second phase is necessary to integrate the intermediate results and further anonymize entire data sets [1].

Algorithm 1 shows two phases of the algorithm. In the first phase, an original data set D is partitioned into smaller ones. Let Di, $1 \leq i \leq p$, denote the data sets partitioned from D the, where p is the number of partitions, and $D = \sum_{i=1}^{p} D_i$, $D_i \cap D_j = \Phi$ $1 \leq i < j \leq p$. Then, we run a subroutine over each of the partitioned data sets in parallel to make full use of the job level parallelization of MapReduce.

Input: Data set D, anonymity parameter k, Intermediate anonymity parameter k^I and the number of partitions *p*.
Output: Anonymous data set D^*.
1. Partition D into D_i, $1 \leq i \leq p$ *using random sampling technique*.
2. 1st phase – Call MRTDS $(D_i, k^i, AL^0) \rightarrow AL'_i$, $1 \leq i \leq p$ in parallel as multiple MapReduce jobs.
3. Merge all intermediate anonymization levels into one, merge $(AL_1' AL_2' \ldots AL_p') \rightarrow AL^I$.
4. 2nd phase – Call MRTDS $(D, k, AL^I) \rightarrow AL^*$ to achieve k-anonymity.
5. Specialize D according to AL^*, Output D^*.

Algorithm 1: Sample Algorithm of TPTDS

Random Sampling Technique is adopted to partition the data set D. An original data set D is specialized for anonymization in a one-pass Mapreduce job by calculating the IGPL values for each specialization. MRTDS is the MapReduce version of Top Down Specialization algorithm. The algorithmic design of MRTDS function mentioned in algorithm 1 is briefly described below.

Input: Data set D, anonymization level AL and anonymity parameter k.
Output: Anonymization level AL′.
1. Initialize IGPL values, i.e., for each specialization *spec* $\in U^{m}_{j=1} Cut_j$. The IGPL value of *spec* is computed by job IGPL Initialization.
2. **while** ɔ spec $\in U^{m}_{j=1} Cut_j$ is valid
2.1. Find the best specialization from AL_i, $spec_{Best}$.
2.2. Update AL_i to AL_{i+1}.
2.3. Update information gain of the new specializations in AL_{i+1}, and privacy loss for each specialization via job IGPL Update.
3. **end while**
4. AL′← AL.

Algorithm 2: Sample algorithm of MRTDS

Algorithm 2 leverages anonymization level to manage the process of anonymization. Step 1 initializes the values of information gain and privacy loss for all specializations, which can be done by the job IGPL Initialization.

MRTDS produces the same anonymous data as the centralized TDS in [12], because they follow the same steps. MTRDS mainly differs from centralized TDS on calculating IGPL values. However, calculating IGPL values dominates the scalability of TDS approaches, as it requires TDS algorithms to count the statistical information of data sets iteratively. MRTDS exploits MapReduce on cloud to make the computation of IGPL parallel and scalable [1]. A variation in Hadoop framework like Haloop [5] have been recently used to support iterative map reduce computation

IV. . DYNAMIC MAPREDUCE FRAMEWORK

MapReduce performance can be improved by optimizing the utilization of slots from two basic perspectives. First, the slots can be classified as idle slots (no running tasks) and busy slots (with running tasks). The utilization of slots can be improved by increasing the busy slots and reducing the idle slots. Second, it is worth noting that not all busy slots are utilized effectively. The two main factors that affect the slot utilization are Speculative tasks and Data Locality. Based on these, we propose DynamicMR [2], a dynamic utilization optimization framework for MapReduce. It consists of three slot allocation technique namely, *Dynamic Hadoop Slot Allocation (DHSA), Speculative Execution Performance Balancing (SEPB), Slot PreScheduling*. The performance and slot utilization of a Hadoop cluster can be optimized with the following step by step processes.

1. If a slot is idle, then DynamicMR will first attempt to improve the slot utilization with DHSA technique. It will evaluate based on numerous constraints like fairness, load balance and decide whether to allocate the idle slot to the task or not.

2. If the allocation is true, DynamicMR will further optimize the performance by improving the efficiency of slot utilization with *SEPB*. It works on top of Hadoop speculative scheduler to check whether to allocate the available idle slots to the pending tasks or to the speculative tasks.

3. When to allocate the idle slots for pending/speculative map tasks, DynamicMR will be able to further improve the slot utilization efficiency from the data locality optimization aspect with *Slot PreScheduling*.

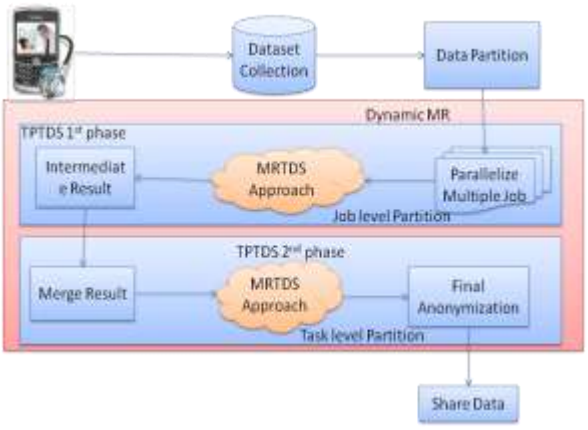

Figure 3. System Architecture

The overall system architecture is described in Figure 3. The data sets are collected from e-health application and it is partitioned into independent chunks. The independent chunks are computed in parallel with the Two-Phase Top down specialization of MRTDS approach. The intermediate results from the first phase are merged and given as a input in the second phase. The final anonymized results are shared with the research institutes. The dynamicMR functionality is briefly described in Figure 4. DHSA will allocate the idle map slots to overloaded reduce slots and vice versa. SEPB will run a backup task to solve the straggler problem and Slot PreScheduling will allocate the idle slots with no impact on fairness [2].

Figure 4. Detailed Design of DynamicMR

A. Dynamic Hadoop Slot Allocation (DHSA)

The dynamic slot allocation policy is based on the observation that as the job proceeds from the map to the reduce phase, there may be idle map (or reduce) slots at different instances. The unused map slots are used for the overloaded reduce tasks (and vice versa) to enhance the performance of the MapReduce. However, two main challenges to be considered are:

(C1) Fairness is an important metric in Hadoop Fair Scheduler (HFS). It is challenging to define and ensure fairness under dynamic slot allocation policy.

(C2) The requirement of resources between the map and the reduce slots are generally different. Hence, dynamic slot allocation policy is designed carefully and need to be aware of such difference [2].

i) Pool-Independent DHSA (PI-DHSA):

HFS adopts max-min fairness [22] to allocate slots across pools with minimum guarantees at the map-phase and reduce-phase respectively. Whenever a job is received, the total demand for map and reduce slots is computed for the current workload.

(C1) If $T_m \leq S_m$ and $T_r \leq S_r$, then the tasks run on their own slots and hence no borrowing.

(C2) If $T_m > S_m$ and $T_r < S_r$, then satisfy reduce tasks for reduce slots first and then use those idle reduce slots for map tasks

(C3) If $T_m < S_m$ and $T_r > S_r$, then schedule unused map slots for running reduce slots.

(C4) If $T_m > S_m$ and $T_r > S_r$, then the system is in completely busy state and hence no borrowing.

Algorithm 3: Scenarios to allocate idle slots to tasks

Then DynamicMR will determine the need to borrow map (or reduce) slots for reduce (or map) tasks based on the scenario described in algorithm 3. Let T_m and T_r be the total number of map tasks and reduce tasks respectively, while S_m and S_r be the total number of map slots and reduce slots respectively [2].

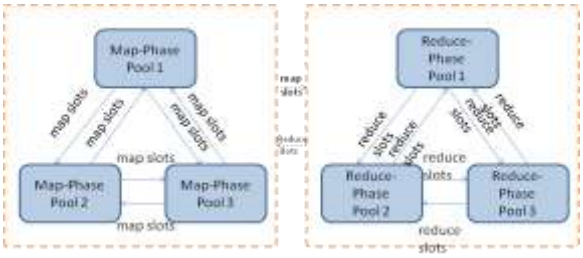

Figure 4. Pool Independent - DHSA

The map slots are borrowed within the map phase pool and the reduce slots are borrowed within the reduce phase pool as shown in Figure 4.

ii) Pool-Dependent DHSA(PD-DHSA):

In dynamic slot allocation, PD-DHSA considers fairness across each pool, as illustrated in Figure 5. The solid line and the dashed line represents the borrow flow for map/reduce slots across the pools. Each pool has two main parts namely map-phase pool and reduce-phase pool. PD-DHSA assumes that each pool is selfish [2]. That is, the primary goal is to utilize its own shared map and reduce slots for its own needs as much as possible before lending them to other pools.

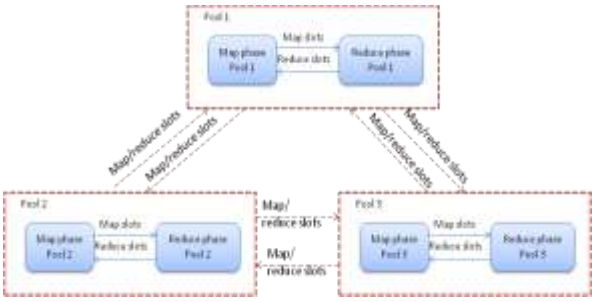

Figure 5. Pool-Dependent DHSA.

B. Speculative Execution Performance Balancing (SEPB)

The execution time in MapReduce jobs are prone to slow running tasks (namely *straggler*) [16]. There are various reasons that cause stragglers, including faulty hardware and software mis-configuration. Stragglers are classified into two types. A task that goes into deadlock status due to endless waiting for certain resources is Hard Straggler. A task that can complete successfully after a longer time is Soft Straggler. However it is not so easy to classify whether it is a hard or a soft straggler initially. To deal it, speculative execution is used in Hadoop. Instead of diagnosing and fixing straggling tasks, it detects the straggling task dynamically using heuristic algorithms such as LATE [17].

Once straggling tasks are detected, it spawns a backup task and allows it to run concurrently along with the

straggler, i.e., there is a computation overlap between the straggler and the backup task. When either of the tasks gets completed the other one will be killed dynamically [2].

C. Slot PreScheduling

Slot PreScheduling technique can improve the data locality while having no negative impact on the fairness of MapReduce jobs. The basic idea is that, there are often some idle slots which cannot be allocated due to the load balancing constrain during runtime, we can pre-allocate those slots of the node to jobs to maximize the data locality.

V. CONCLUSION

The potential of big data is to transform the way healthcare providers use sophisticated technologies to gain knowledge from their clinical and other data sources and make good decisions. In the near future we will see rapid and widespread implementation of big data analytics in health care industry. Big data analytics guarantees privacy and security. The applications of big data analytics are still at budding stage of development and its implementation in the health care industry will surely help its organizations.

The scalability problem of large-scale data is investigated and presented a highly scalable two-phase TDS approach using incremental map reduce clusters. Datasets are partitioned in parallel in the first phase, producing intermediate results. Data Anonymization is a means of partially thwarting this privacy threat in e-health Records. Even if e-health records are in place at many hospitals they are not standardized and this data is usually unstructured, just like a log file, for with Map Reduce is well-suited. Then, the intermediate results are merged and further anonymized to produce consistent k-anonymous data sets in the second phase.

This paper proposes a framework which is aiming that it will improve the performance of MapReduce workloads and at the same time will maintain the fairness. DHSA the technology about which we have mentioned above focuses on the maximum utilization of slots by allocating map (or reduce) slots to map and reduce tasks dynamically. In future we can extend the concepts to analyze big data based optimal balanced scheduling approach in accurate manner.

REFERENCES

[1] Zhang.X, Yang.L.T, Liu.C and Chen.J (2014), "*A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud*" IEEE trans.parallel and distributed sys., vol 25, no.2, pp. 363-373.

[2] Tang.S, Lee.B, He.B, "*DynamicMR: A Dynamic Slot Allocation Optimization Framework for MapReduce Clusters*", IEEE Transactions on Cloud Computing, vol.25, no.5, pp. 520-534.

[3] Armbrust.M, Fox.A, Griffith.R, Joseph.A.D,Katz.R, Konwinski.A, Lee.G, Patterson.D, Rabkin.A, Stoica.I, and Zaharia.M(2010), "*A View of Cloud Computing*," Comm. ACM, vol. 53, no. 4, pp. 50-58.

[4] Borkar.V, Carey.M.J, and Li.C(2012), "*Inside 'Big Data Management': Ogres, Onions, or Parfaits?*," Proc. 15th Int'l Conf. Extending Database Technology (EDBT '12), pp. 3-14.

[5] Bu.Y, Howe.B, Balazinska.M, and Ernst.M.D(2012), "*The Haloop Approach to Large-Scale Iterative Data Analysis*," VLDB J., vol. 21, no. 2, pp. 169-190.

[6] Cao.N, Wang.C, Li.M, Ren.K, and Lou.W(2011), "*Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data*," Proc. IEEE INFOCOM, pp. 829-837.

[7] Chaudhuri.S(2012), "*What Next?: A Half-Dozen Data Management Research Goals for Big Data and the Cloud*," Proc. 31st Symp. Principles of Database Systems (PODS '12), pp. 1-4.

[8] Fung.B.C.M, Wang.K, Chen.R, and Yu.P.S(2010), "*Privacy-Preserving Data Publishing: A Survey of Recent Developments*," ACM Computing Surveys, vol. 42, no. 4, pp. 1-53.

[9] Guo.Z.H, Fo.G, Zhou.M, Ruan.Y, "*Improving Resource utilization in MapReduce*". In IEEE cluster'12. pp,402-410,2012.

[10] LeFevre.K, DeWitt.D.J, and Ramakrishnan.R(2008), "*Workload-Aware Anonymization Techniques for Large-Scale Data Sets*," ACM Trans. Database Systems, vol. 33, no. 3, pp. 1-47, 2008.

[11] Mohan.P, Thakurta.A, Shi.E, Song.D, and Culler.D(2012), "*Gupt: Privacy Preserving Data Analysis Made Easy*," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '12), pp. 349-360.

[12] Mohammed.N, Fung.B, Hung.P.C.K, and Lee.C.K(2010), "*Centralized and Distributed Anonymization for High-Dimensional Healthcare Data*," ACM Trans. Knowledge Discovery from Data, vol. 4, no. 4, Article 18.

[13] Roy.I,Setty.S.T.V, Kilzer.A, Shmatikov.V, and Witchel.E, "*Airavat: Security and Privacy for Mapreduce*," Proc. Seventh USENIX Conf. Networked Systems Design and Implementation (NSDI '10), pp. 297-312, 2010.

[14] Takabi.H, Joshi.J.B.D, and Ahn.G(2010), "*Security and Privacy Challenges in Cloud Computing Environments*," IEEE Security and Privacy, vol. 8, no. 6, pp. 24-31.

[15] Wang.L, Zhan.J, Shi.W, and Liang.Y(2012), "*In Cloud, Can Scientific Communities Benefit from the Economies of Scale?*," IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 2, pp.296-303.

[16] White.T, "*Hadoop: The Definitive Guide*", 3rd version, O'Reilly Media,2012.

[17] Zaharia.M, Konwinski.A, Joseph.A.D, Katz.R, Stoica.I, "*Improving MapReduce Performance in heterogeneous environments*". In OSDI'08, pp.29-42, 2008.

[18] Zaharia.M, Borthakur.D, Sarma.J, Elmeleegy.K, Schenker.S, Stoica.I, *"Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling"*. In EuroSys'10, pp.265-278,2010.

[19] Zhang.K, Zhou.X, Chen.Y, Wang.X, and Ruan.Y(2011), "*Sedic: Privacy-Aware Data Intensive Computing on Hybrid Clouds*," Proc. 18th ACM Conf. Computer and Comm. Security (CCS '11) pp. 515-526.

[20] Microsoft Health Vault, *http://www.microsoft.com/health/ww/products/Pages/healthvault.aspx*, 2013.

[21] Amazon Web Services, "Amazon Elastic Mapreduce," *http://aws.amazon.com/elasticmapreduce/*, 2013.

[22] Max-Min Fairness (Wikipedia). http://en.wikipedia.org/wiki/Max-min_fairness