International Journal of Computer Science
Oddity...Probe...Reviste...

http://www.ijcsjournal.com Volume 2, Issue 1, No 4, 2014. ISSN: 2348-6600.
Reference ID: IJCS-049. PAGE NO: 278-286.

Privacy Preserving Cost Reducing Heuristic Approach For Intermediate Datasets In Cloud

M.Savitha
PG Student
Department of CSE
Bharathiyar Institute of Engineering For Women,
Attur, Tamil Nadu, India.
savicse25@gmail.com

Abstract— **Cloud computing provides massive computation power and storage capacity which enable users to deploy computation and data intensive applications without infrastructure investment. Along the processing of such applications, a large volume of intermediate datasets will be generated and often stored them to save the cost of recomputing them. In this paper, toward achieving the minimum cost benchmark and for cost effectively storing large volume of generated application datasets in the cloud, we propose a novel highly cost effective and practical storage strategy that can automatically decide whether a generated dataset should be stored or not at runtime in the cloud and from that stored dataset, inorder to provide security to the sensitive dataset, we propose a novel upper bound privacy leakage constraint-based approach to identify which intermediate data sets need to be encrypted and which do not, so that privacy-preserving cost can be saved and also the privacy requirements of data holders can be satisfied.**

Key words—**Datasets storage, computation, cloud computing, data storage privacy, privacy preserving, intermediate dataset, privacy upper bound.**

I.INTRODUCTION

Cloud computing is regarded as an ingenious combination of serious of technologies, establishing a novel model by offering several services and using economies of scale [1], [2].Participants in the business chain of cloud computing can benefit from this model and cloud users can save huge capital investment of infrastructure and concentrate on their own core business [3].Many companies and organizations have been migrating or building their business into cloud environment. So in this, during the execution of scientific applications a huge amount datasets will be generated. These generated datasets contain important intermediate or final results of the computation and need to be stored as valuable resources. This is because of two reasons. First the data can be reused i.e., scientists may need to re-analyze the results or apply new analyzes on the existing datasets [5] .Second the data can be shared i.e., for collaboration the computation results are shared and hence the datasets are used by scientists from different institutions [6]. Storing valuable datasets can save their regeneration cost when they are reused. However the big challenge is its huge size .Due to this the large amount of storage space and costs are required, along with this the major problem is the privacy of the dataset holders that is affected due to this storage of intermediate datasets in cloud because that can be accessed and processed by multiple parties. Cloud computing adopts the pay-as-you-go model, where users are charged according to the usage of cloud services such as computing, storage and network services [7].

With the pay-as-you-go model, the total application cost in the cloud highly depends on the strategy of storing the application datasets, e.g., storing all the generated application datasets in the cloud may result in a high storage cost, because some datasets may be rarely used but large in size and also deleting all the generated datasets and regenerating them every time when needed may result in a high computation cost. And another major problem due to this storage is security and privacy of dataset and dataset holders respectively [4], [5]. The privacy concerns caused by retaining intermediate datasets in cloud are important. The storage of intermediate datasets enlarges attack surfaces so that privacy requirements data holders are at risk of being violated. Usually intermediate datasets in cloud are accessed and processed by multiple parties, but rarely controlled by original dataset holders. This enables an adversary to collect intermediate datasets together and menace privacy-sensitive information from them and bringing considerable economic loss or severe social reputation impairment to data owners.

II. EXISTING SYSTEM

A.APPLICATION DATA AND DDG

There are two types of data stored in the cloud:

1. Original data: These are the data uploaded by users, for example, they are usually the raw data. For these data, users need to decide whether they should be stored or deleted because they cannot be regenerated by the system once deleted.
2.
3. General data: These are the data newly produced in the cloud while the applications run. They are the intermediate or final computation results of the applications, which can be reduced in the future. For these data, their storage can be decided by the system because they can be regenerated if their provenance is known. Hence, the data sets storage strategy is only applied to the generated data in the cloud that can be automatically decide the storage status of generated data sets in applications.

Data Dependency Graph (DDG) is a directed acyclic graph (DAG), which is based on data provenance in scientific applications. All the data sets once generated in the cloud, whether stored or deleted, their references are recorded in DDG i.e., it depicts the generation relationship of data sets, with which the deleted data sets can be regenerated from their nearest existing preceding data sets. Fig.1 depicts a simple DDG, where every node in the graph denotes a data set. We denote data set d_i in DDG as $d_i \in$ DDG. And also d_1 pointing to d_2 means that d_1 is used to generate d_2;d_2 pointing to d_3 and d_5 means that d_2 is used to generate d_3 and d_5 based on different operations,d_4 and d_6 pointing to d_7 means that d_4 and d_6 are used together to generate d_7.To better describe the relationships of data sets in DDG, we define a symbol The arrow, $\rightarrow$, which denotes that two data sets have a generation relationship, where $d_i \rightarrow d_j$ means that di is a predecessor data set of dj in DDG. For example, in Fig. 1's DDG, we have $d_1 \rightarrow d_2$, $d_1 \rightarrow d_4$, $d_5 \rightarrow d_7$, and so on. Furthermore, $\rightarrow$is transitive, i.e., $d_i \rightarrow d_j \rightarrow d_k \Leftrightarrow d_i \rightarrow d_j \wedge d_j \rightarrow d_k \Rightarrow d_i \rightarrow d_k$.

B. DATA SETS STORAGE COST MODEL

In a commercial cloud computing environment, service providers have their cost models to charge users. In general, there are two basic types of resources in the cloud: Computation and storage. Popular cloud services providers cost models are based on these types of resources. For example, Amazon cloud services prices are as follows:

- \$0.1 per CPU instance hour for the computation resources;
- \$0.15 per Gigabyte per month for the storage resources;

Therefore the data sets storage cost model in the cloud as follows:
Cost = Computation + Storage,

Where the total cost of the application data sets storage, Cost, is the sum of Computation, which is the total cost of computation resources used to regenerate data sets, and Storage, which is the total cost of storage resources used to store the data sets. To utilize the data sets storage cost model, we define the attributes for the data sets in DDG the same as in [30].For or data set d_i, its attributes are denoted as: $<x_i,y_i,f_i,v_i,provSet_i,CostR_i>$,where:

- x_i denotes the regeneration cost of the data set d_i from its direct predecessors in the cloud.
- y_i denotes the cost of storing data set d_i in the cloud per time unit.
- f_i is flag, which denotes the status of whether data set d_i is stored or deleted in the cloud.
- v_i denotes the usage frequency, which indicates how often d_i is used.
- $provSet_i$ denotes the set of stored provenance that are needed when d_6 are used together to generate d_7.

C.SENSITIVE INTERMEDIATE DATA SET MANAGEMENT

Data provenance is used to manage intermediate data sets. Provenance is commonly defined as the origin, source or history of derivation of some objects and data, which can be used as the information upon how data were generated. Reproducibility of data provenance can help to regenerate data sets. The information recorded in data provenance is leveraged to build up the generation relationships of data sets [6].

Let d_o be a privacy-sensitive original data set. We use D= $\{d_1,d_2,\ldots,d_n\}$ to denote a group of intermediate data sets generated from d_0 where n is the number of intermediate data sets. The notion of intermediate data herein refers to both intermediate and resultant data [6]. Directed Acyclic Graph (DAG) is exploited to capture the topological structure of generation relationships among these data sets.

Definition 1 (Sensitive intermediate data set graph). A DAG representing the generation relationships of intermediate data sets D from d_o is defined as a sensitive Intermediate data set Graph, denoted as SIG. Formally, SIG=(V,E), where $V=\{d_o\}\cup D$, E is a set of directed edges. A directed edge (d_p,d_c) in E means that part or all of d_c is generated from d_p, where $d_p,d_c \in \{d_o\}\cup D$. In particular, an SIG becomes a tree structure if each data set in D is generated from only one parent data set. Then, we have the following definition for this situation.

Definition 2 (Sensitive intermediate data set tree (SIT)).An SIG is defined as a sensitive intermediate data set Tree if it is a tree structure. The root of the tree is d_o. An SIG or SIT not only represents the generation relationships of an original data set and its intermediate data sets, but also captures the propagation of privacy-sensitive information in d_o is scattered into its offspring data sets. Hence, an SIG or SIT can be employed to analyze privacy disclosure of multiple data sets. An intermediate data set is assumed to have been anonymized to satisfy certain privacy requirements. Privacy leakage of a data d is denoted as $PL_s(d)$, meaning the privacy-sensitive information obtained by an adversary after d is observed. The value of $PL_s(d)$ can be deduced directly from d. Similarly, privacy leakages of multiple data sets in D are observed. It is challenging to acquire the exact value of PL_m (D) due to the inference channels among multiple data sets .

D.PRIVACY-PRESERVING COST PROBLEM

Privacy-Preserving cost of intermediate data sets stems from frequent en / decryption with charged cloud services. Cloud service venders have set up various pricing models to support the pay-as-you-go model, e.g., Amazon web services pricing model. Practically, en / decryption need computation power, data storage, and other cloud services. To avoid pricing details and focus on the discussion of our core ideas, we combine the prices of various services required by en / decryption into one. This combined price is denoted as PR. PR indicates the overhead of en / decryption on per GB data per execution.

III PROPOSED SYSTEM.

A.PRACTICAL DATA SETS STORAGE STRATEGY

We first enhance the linear CTT-SP algorithm to incorporate user's preferences then, we introduce our highly practical local optimization based storage strategy for approaching the minimum cost benchmark.

a) Enhanced CTT-SP Algorithm for linear DDG Segment

With the excessive computation and storage resources in the cloud, users can flexibly choose storage strategies for application generated data sets. The CCT-SP algorithm can find the minimum cost storage strategy for saving the storage cost we have to regenerate it whenever it needs to be reused. Regeneration causes not only the computation resources, but also a delay for accessing the data, i.e., waiting for the set to get ready. Depending on the requirements of applications, users may have different tolerable computation delay on accessing different data sets. Some data sets are stored at an immediate data access. Furthermore, knowing the minimum cost on storing more data sets to reduce the average computation delay. We have enhanced the linear CTT-SP algorithm by introducing two new parameters that can represent user's preferences and provide users some flexibility in using the storage strategy. The two parameters are denoted as T and λ.

T is the parameter used to represent user's tolerance on data accessing delay. Users need to inform the cloud service provider about the data sets that they have requirements on their availabilities. For a data set d_i, which needs regeneration, T_i is the delay time that users can tolerant when they want to access it. Furthermore, T is also related to the requirements of applications. For example, some applications may have fixed time constraints [9], such as the weather forecast application. In this situation, for some particular data sets, the value for T_i can set according to the starting time and finishing time (i.e., deadline) of the application. In a word, T is the time constraint of data sets regeneration. In the storage strategy, the regeneration time of any deleted data set d_i cannot exceed its T_i. Especially, if T_i is smaller than the generation time of data set d_i itself (i.e., $T_i<x_i/Price_{cpu}$, where $Price_{cpu}$ is the price of computation

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 1, No 4, 2014. ISSN: 2348-6600.
Reference ID: IJCS-049. PAGE NO: 278-286.

resources used to regenerate d_i in the cloud), then we have to store d_i, no matter how expensive d_i's storage cost is.

λ is the parameter used to adjust the storage strategy when users have extra budget on the minimum cost benchmark to store more data sets for reducing the average data sets accessing time. Based on users extra budget, we can calculate a proper value of λ^7, which is between 0 and 1. We multiply every data set d_i's storage cost rate (i.e., y_i) by λ, and use it to compare with d_i's regeneration cost rate(i.e., $genCost(d_i)*v_i$) for deciding its storage status. Hence, more data sets tend t be stored, and literally speaking, data sets will be deleted only when their storage cost rates are (1/λ) times higher than their regeneration cost rates. For example, λ_i=0.8 means that users are willing to store data sets with the storage cost up to 1.25 times higher than the regeneration cost.We enhance the linear CTT-SP algorithm by incorporating these two new parameters. As defined in the CTT-SP algorithm, for every two data sets in the DDG, there is a cost edge in the Cost Transitive Tournament (CTT), i.e.,

$$\left(\forall d_i, d_j \in DDG \wedge d_i \rightarrow d_j\right) \Rightarrow \exists e <$$

To incorporate the parameter of data accessing delay tolerance (i.e., T), in the enhanced linear CTT-SP algorithm, the edge e<d_i,d_j> has to further satisfy the condition:

$$e < d_i, d_j > \Rightarrow \forall d_k \epsilon DDG \wedge \left(d_i \rightarrow d_k \rightarrow d_j\right) \wedge \left(\frac{genCost\,(d_k)}{Price_{cpu}} < T_k\right).$$

With this condition, long cost edges may be eliminated from the CTT. It guarantees that in all storage strategies found by the algorithm, for any deleted data set d_i, its regeneration time is smaller than T_i, if users have the requirement on its availability.To incorporate the parameter os users cost preference of storage (i.e., λ), in the enhanced linear CTT-SP algorithm, we set the weight of a cost edge in CTT as

By introducing the two new parameters, the enhanced linear CTT-SP algorithm can find the minimum cost storage strategy of a linear DDG satisfying users preferences on storage with a time complexity of $O(n^4)$, where n is the number of data sets in the DDG or DDG segment on which the algorithm applies. These two parameters are generic for data sets storage strategies and their values are dependent on the requirements of specific applications.

b) Practical Cost – Effective Data Sets Storage Strategy

In this section, we introduce our local-optimization-based data sets storage strategy, which is designed based on the enhanced linear CTT-SP algorithm. The philosophy is to derive localized minimum costs instead of a global one, aiming at approaching the minimum cost benchmark with highly practical time complexity. Our strategy contains the following four rules:

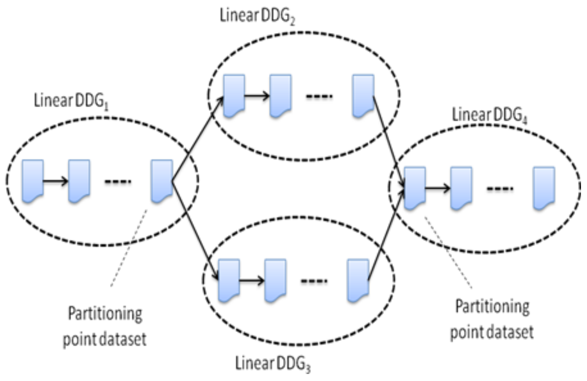


Fig. 2. Dividing a DDG into linear DDG segments

Given a general DDG, we first partition the DDG into linear segments and apply the enhanced CTT-SP algorithm. We search for the data sets that have multiple direct predecessors or successors (i.e., the join and split data sets in the DDG), and use these data sets as the partitioning points ti divide it into linear DDG segments, as shown in Fig.2. Based on the linear DDG segments, we use the enhanced linear DDG segments, we use the enhanced linear CTT-SP algorithm to find their storage strategies. This is the essence of local optimization. When new data sets are generated in the system they are treated as a new DDG segment and added to the old DDG. Correspondingly, its storage status is calculated in the same way as the old DDG.When a data set's usage frequency is changed, the storage status of the linear DDG segment that contains this data set is recalculated.

B. PRIVACY LEAKAGE UPPER BOUND CONSTRAINT-BASED APPROACH FOR PRIVACY PRESERVING

We propose an upper bound constraint-based approach to select the necessary subset of intermediate data sets that needs to be encrypted for minimizing privacy-preserving cost.

a)Basic Notations and Properties on an SIT

Let $d_r \epsilon D$ denote an intermediate data set in an SIT. Its directly generated data sets constitute a set CD (d_r) {$d_{r1}, \ldots, d_{ri}$}, where $d_{r1}, \ldots, d_{ri} \in D$. For any $d \in CD(d_r)$,$PL_s(d) \leq Pl_s(d_r)$ because all information in d is from d_r. Further, let PD(d_r)) {$d_{r1}, \ldots, d_{rj}$} be all posterity data sets generated from d_r, where $d_{r1}, \ldots, d_{rj} \in D$. Similar to CD($d_r$), $PL_s(d) \leq PL_s(d_r)$ for any $d \in PD(d_r)$. A subtree of an SIT with $d_r \in D$ being its root is denoted as SDT(d_r).

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 1, No 4, 2014. ISSN: 2348-6600.
Reference ID: IJCS-049. PAGE NO: 278-286.

b)Recursive Privacy Leakage Constraint Decomposition

To satisfy the PLC, we decompose the PLC recursively into different layers in an SIT. Let the privacy leakage threshold required in the layer L_i be ε_i, $1 \leq i \leq H$. The privacy leakage incurred by UD_i in the solution π_i can never be larger than ε_i, i.e., $PL_m(UD_i) \leq \varepsilon_i$. The threshold ε_i can be regarded as the privacy leakage threshold of the remainder part of an SIT after the layer L_{i-1}. In terms of the basic idea of our approach, the privacy leakage constraint $PL_m(UD_i) \leq \varepsilon_i$ is substituted by one of its sufficient conditions. The PLC can be substituted by a set of privacy leakage constraints, named as PLC_i:

$$\sum_{d \in UD_i} PL_s(d) \leq \varepsilon_i, 1 \leq i \leq H.$$

The above threshold ε_i, , is calculated by

$$\begin{cases} \varepsilon_i = \varepsilon_{i-1} - \sum_{d \in UD_{i-1}} PL_s(d) \\ \varepsilon_1 = \varepsilon. \end{cases}$$

A local encryption solution in the layer L_i is feasible if it satisfies the PLC_1. The set of feasible solution in L_i is denoted as , where j is the number feasible solutions. Similarly, feasible global encryption solution can be denoted as π, where

$$\pi_{ij_i} \in \Lambda_i^f, 1 \leq i \leq H, 1 \leq k \leq$$

Given a feasible global solution for an SIT, we compress the SIT into a "compressed" tree layer by layer from L_1 to L_H, denoted as CT is achieved via three steps. First, the data sets in ED_i are "compressed" into one encrypted node. According to the EDT property, these compressed nodes together with the original data set appear to be a string with the length being H. Second, all offspring data sets of the data sets in UD_i are omitted. This will not affect the privacy preserving in terms of the RPC property. Third, the data sets in UD_i are compressed into one node. Note that the action "compressed" here is imaginary from a logical perspective for a demonstration purpose.

c)Minimum Privacy-Preserving Cost

Usually, more than one feasible global encryption solution exists under the PLC_1 constraints, because there are many alternative local solutions in each layer. Further, each intermediate data set has various size and frequency of usage, leading to different overall cost with different solutions. Therefore, it is desired to find a feasible solution with the minimum privacy-preserving cost under privacy leakage constraints. Note that the minimum solution mentioned herein is somewhat pseudo minimum because an upper bound of joint privacy leakage is just an approximation of its exact value. But a solution can be exactly minimal in the sense of the PLC_1 constraints. The minimum cost for privacy preserving of the data sets after L_{i-1} under the privacy leakage threshold ε_i is represented as $CM_i(\varepsilon_i)$, $1 \leq i \leq H$. Given a feasible local encryption solution $\pi_i = (ED_i, UD_i)$ in L_i, the cost incurred by the encrypted data sets in L_i is denoted as $C_i(\pi_i)$

$$C_i(\pi_i) \triangleq \sum_{d_k \in ED_i} S_k.PR.f_k, 1 \leq i \leq H.$$

Then, $CM_i(\varepsilon_i)$ is calculated by the recursive formula:

$$\begin{cases} CM_i(\varepsilon_i) = \min_{\pi_{ij} \in \Lambda_i^f} \left\{ \sum_{d_k \in ED_i} (S_k.PR.f_k) \right. \\ \left. + CM_{i+1}\left(\varepsilon_i - \sum_{d_k \in UD_{ij}} (PL_s(d_k) \right) \right\}, \\ CM_{H+1}(\varepsilon_{H+1}) = 0. \end{cases}$$

As a result, $CM_1(\varepsilon)$ is the minimum privacy-preserving cost required in the optimization problem .The privacy-preserving solution (D^{enc}, D^{une}) can be determined during the process of acquiring $CM_1(\varepsilon)$. According to the specification of $CM_1(\varepsilon_i)$, an optimal algorithm can be designed to identify the optimal privacy-preserving solution.

d)Privacy-Preserving Cost Reducing Heuristic Algorithm

In this section we design a heuristic algorithm to reduce privacy-preserving cost. In the state-search space for an SIT, a state node SN_i in the layer L_i herein refers to a vector of partial local solutions, i.e., SN_i corresponds to , where π Note that the state-search tree generated according to an SIT is different from the SIT itself, but the height is the same. Appropriate heuristic information is quite vital to guide the search path to the goal sate. The goal state in our algorithm is to find a near-optimal solution in a limited search space.Heuristic values are obtained via heuristic functions. A heuristic function, denoted as $f(SN_i)$, is defined to compute the heuristic value of SN_i.Generally, $f(SN_i)$ consists of two parts of heuristic information, i.e., $f(SN_i)=g(SN_i)+h(SN_i)$, where the information $g(SN_i)$ is gained from the start state to the current state node SN_i and the information $h(SN_i)$ is estimated from the current state node to the goal state, respectively.

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 1, No 4, 2014. ISSN: 2348-6600.
Reference ID: IJCS-049. PAGE NO: 278-286.

Intuitively, the heuristic function is expected to guide the algorithm to select the data sets with small cost but high privacy leakage to encrypt. Based in this, $g(SN_i)$ is defined as $g(SN_i) \triangleq$ where C_{cur} is the privacy-preserving cost that has been incurred so far, ε is the initial privacy leakage threshold, and ε_{i+1} is the privacy leakage threshold for the layers after L_i. Specifically, C_{cur} is calculated by $C_{cur} = \sum_{d_j \epsilon \cup_{k=1}^{i}}$

The value of $h(SN_i)$ is defined as $h(SN_i)=(\varepsilon_{i+1}.C_{des}.BF_{AVG})/PL_{AVG}$. Similar to the meaning of in $g(SN_i)$, smaller ε_{i+1} in $h(SN_i)$ implies more data sets before L_{i+1} are kept unencrypted. If a data set with smaller depth in an SIT is encrypted, more data sets are possibly unencrypted than with larger depth, because the former possibly has more descendant data sets. For a state node SN_i, the data sets in its corresponding ED_k are the roots of a variety of subtrees of the SIT. These trees constitute a forest, denoted as In $h(SN_i)$, C_{des} represents the total cost of the data sets in , and is computed via $C_{des} = \sum_{d_l \epsilon ED_k} \sum_{d_j \epsilon PD(d_l}$ Potentially, the less C_{des} is, the fewer data sets in following layers will be encrypted. BF_{AVG} is the average branch factor of the forest , and can be computed by $BF_{AVG}=N_E/N_1$, where N_E is the number of edges and N_I is the number of internal data sets in .Smaller BF_{AVG} means the search space for sequent layers will be smaller, so that we can find a near-optimal solution faster. The value of PL_{AVG} indicates the average privacy leakage of data sets in , calculated by $PL_{AVG} = \sum_{d_l \epsilon ED_k} \sum_{d_j \epsilon PD(d_l)} Pl$ Heruistically, the algorithm prefers to encrpt the data sets which incur less cost but disclose more privacy-sensitive information. Thus, higher PL_{AVG} means more data sets in should be encrypted to preserve privacy from a global perspective. Based on the above analysis, the heuristic value of the search node SN_i can be computed by the formula:

$$f(SN_i) = \frac{C_{cur}}{\varepsilon - \varepsilon_{i+1}} + \frac{\varepsilon_{i+1}.C_{des}.BF_{AVG}}{PL_{AVG}}.$$

Based on this heuristic, we design a heuristic privacy-preserving cost reduction algorithm, denoted as H_PPCR. The basic idea is that the algorithm iteratively selects a state node with the highest heuristic value and then extends its child state nodes until it reaches a goal state node. The privacy-preserving solution and corresponding cost are derived from the goal state.

Extension to SIG

Although SITs can suit many applications, SIGs are also common, i.e., an intermediate data set can originate from more than one parent data set Thus, it is possible that $PL_s(d_5)$ is larger than $PL_s(d_2)$ or $PL_s(d_3)$, resulting in the failure of Lemma 1 and RPC property. As a result, our approach cannot be directly applied to an SIG. However, we can adapt it to an SIG with minor modifications. Let d_m denote a merging data set that inherits data from more than one predecessor. As only one root data set is assumed to exist in an SIG, all paths from the root data set d_o to d_m must converge at one point beside d_m itself. Let d_s denote this source data set. The inequality $PL_s(d_m) \leq PL_s(d_s)$ holds because all privacy information of d_m comes from d_s. let $PD_i(d_s)$ be the set of the offspring data sets of d_s in the layer L_i, then $PD_i(d_s) \subseteq PD(d_s)$. Data sets in $PD_i(d_s)$ are split into ED_i and UD_i when determining which data sets are encrypted.

We discus three cases where the graph structure can affect the applicability of our approach on an SIG. The first one is $PD_i(d_s) \subseteq UD_i$, i.e., all data sets in $PD_i(d_s)$ will keep unencrypted. All ancestor data sets of d_m after the layer L_i will keep unencrypted according to the RPC property. So, the data set d_m poses little influence on applying our algorithm to an SIC because d_m will not be considered in following steps. The second one is $PD_i(d_s) \subseteq ED_i$, i.e., all data sets in $PD_i(d_s)$ are encrypted. If d_m is a child of a data set in $PD_i(d_s)$, d_m is added to CDE_{i+1} for the next round. Assume the parent data set is d_p. Then, we delete the edges pointing to d_m from parents except d_p, e.g., (d_2,d_5) is retained while (d_3,d_5) and (d_4,d_5) are deleted in Fig.3c. Logically, d_m can be deemed as a "compressed" candidate data set of several imaginary data sets in CDE_{i+1}, which is similar to the construction of a compressed tree. The lastone is that $D_x \subseteq UD_i$ and $D_y \subseteq ED_i$, where $D_x \cap D_y = \emptyset$ and $D_x \cap D_y = PD_i(d_s)$, i.e., part of data sets in $PD_i(d_s)$ are encrypted while the remainder keep unencrypted. According to the RPC property, it is safe to expose part of privacy information in d_m, where the part of privacy information is from D_x. The edges which point to d_m form data sets in D_x are deleted. Further, the value of its direct parents who are data sets in D_y or the offspring of D_y if $PL_s(d_m)$ is larger than the maximum.

To make the approach for an SIT available to an SIG as well, three minor modifications are required. The first one is to identify all merging data sets. The second one is to adjust the SIG according to the third case discussed above if $UD_\pi \neq \emptyset$ after we get a local solution $\pi=(ED_\pi,UD_\pi)$. The third one is to label the data sets that have been processed. In this way, it is unnecessary to explicitly delete edges discussed in the second case.

IV SIMULATION RESULT

The random simulations are conducted on randomly generated DDG with data sets of random sizes, generation times, and usage frequencies. In the experiments, we use a linear DDG segment with 50 data sets, each with a random size from 100 GB to 1 TB. The generation time is also random, from 1 to 10 hours. The usage frequency is again random, from 1 to 10 days (time between every usage). The prices of clod services follow the well-known Amazon's cost model, i.e., $0.1 per CPU instance hour for computation and $0.15 per gigabyte per month for storage.

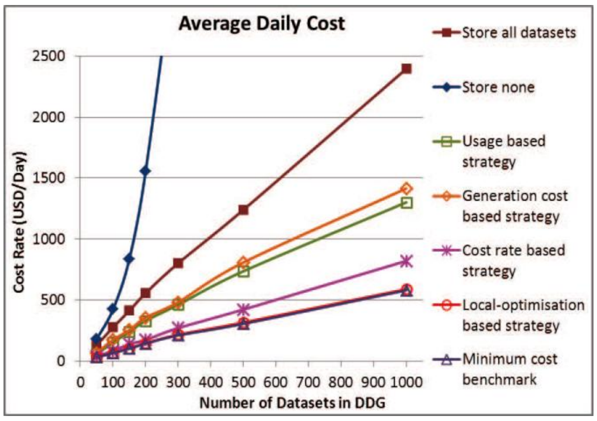


Fig 3.Performance graph for dataset storage

Overall Performance Evaluation

First, we evaluate the cost-effectiveness of our local-optimization-based storage strategy. We randomly connect the linear DDG segments into large DDGs with different numbers of data sets and utilize different storage strategies to calculate their cost rates (i.e., average daily cost) of storing the DDGs. For our local-optimization-based strategy, linear segments are also treated as the smallest units of the DDG partition. Fig3 shows the increase of the daily cost of different strategies as the number of data sets grows in the DDG. From Fig. 3, we can see that the "store none data set" and "store all data sets" strategies are very cost effective, because their daily cost rates grow fast as the data sets number grows.

The cost rate-based strategy has a better performance than both the generation cost-based strategy and usage based strategy, but it is still much higher than the minimum cost bench mark. Our local-optimization-based strategy is the most cost-effective data sets storage strategy, which has the average cost rate only 1.6 percent higher than the minimum cost benchmark in our random simulations. For a specific example, for the DDG with 200 data sets, the cost rate of our local-optimization-based strategy(i.e., USD 145.7 per day) is only 1.5 percent higher than the minimum cost benchmark(i.e., USD 143.5 per day). In contrast, the cost rate-based strategy (the second most cost effective strategy) has the cost rate (i.e., USD 173.3 per day)17.2 percent higher than the minimum cost bench mark. This results indicates that our local-optimization-based strategy is very close to the minimum cost benchmark. Next , encrypting all data sets for privacy preserving is widely adopted in existing research [8], [9], [10]. This category of approach is denoted as ALL_ENC. The privacy-preserving cost of ALL-ENC is denoted as C_{ALL}. To facilitate the comparison, the privacy-preserving cost of H_PPCR is denoted as C_{HEU}.

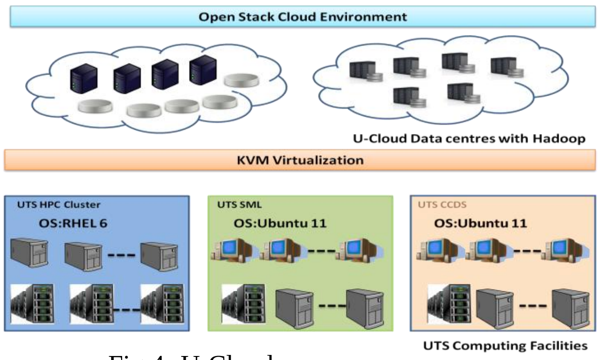
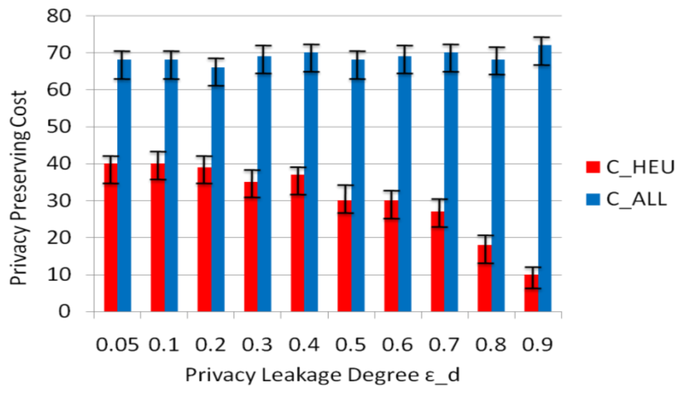


Fig 4. U-Cloud

U-Cloud is a cloud computing environment at the University of Technology Sydney(UTS). The system over-view of U-



http://www.ijcsjournal.com Volume 2, Issue 1, No 4, 2014. ISSN: 2348-6600.
Reference ID: IJCS-049. PAGE NO: 278-286.

Cloud is depicted in Fig. 4. The computing facilities of this system are located among several labs at UTS. On top of hardware and Linux operating system, we install KVM virtualization software which virtualizes the infrastructure and provides unified computing and storage resources. To create virtualized data centers, we install OpenStack open-source cloud environment for global management, resource scheduling and interaction with users. Further, Hadoop is installed based on the cloud built via OpenStack to facilitate massive data processing. Our experiments are conducted in this cloud environment. The experimental result on real-world data sets is depicted in Fig. 5, our approach can reduce the privacy-preserving cost significantly in real-world scenarios.

IV.CONCLUSION

Thus the novel approach for achieving minimum dataset storage cost by using local optimization based strategy and linear CTT-SP algorithm was created. To preserve the privacy of stored intermediate datasets we design an upper bound constraint approach that identifies which part of intermediate datasets needs to be encrypted while the rest does not, inorder to save the privacy preserving cost was used. By implementing both the storage and privacy preserving approach ,the cost can be reduced and security will be increased than the existing system .

REFERENCES:

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

[2] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic,“Cloud Computing and Emerging It Platforms: Vision, Hype, and Reality for Delivering Computing as the Fifth Utility,” Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, 2009.

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

[4] H. Takabi, J.B.D. Joshi, and G. Ahn, “Security and Privacy Challenges in Cloud Computing Environments,” IEEE Security & Privacy, vol. 8, no. 6, pp. 24-31, Nov./Dec. 2010.

[5] D. Zissis and D. Lekkas, “Addressing Cloud Computing Security Issues,” Future Generation Computer Systems, vol. 28, no. 3, pp. 583-592, 2011.

[6] D. Yuan, Y. Yang, X. Liu, and J. Chen, “On-Demand Minimum Cost Benchmarking for Intermediate Data Set Storage in Scientific Cloud Workflow Systems,” J. Parallel Distributed Computing,vol. 71, no. 2, pp. 316-332, 2011.

[7] S.Y. Ko, I. Hoque, B. Cho, and I. Gupta, “Making Cloud Intermediate Data Fault-Tolerant,” Proc. First ACM Symp. Cloud Computing (SoCC ’10), pp. 181-192, 2010.

[8] H. Lin and W. Tzeng, “A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding,” IEEE Trans.Parallel and Distributed Systems, vol. 23, no. 6, pp. 995-1003, June 2012.

[9] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data,” Proc.IEEE INFOCOM ’11, pp. 829-837, 2011.
[10] M. Li, S. Yu, N. Cao, and W. Lou, “Authorized Private Keyword Search over Encrypted Data in Cloud Computing,” Proc. 31st Int’l Conf. Distributed Computing Systems (ICDCS ’11), pp. 383-392, 2011.

[11] C. Gentry, “Fully Homomorphic Encryption Using Ideal Lattices,”Proc. 41st Ann. ACM Symp. Theory of Computing (STOC ’09),pp. 169-178, 2009.

[12] B.C.M. Fung, K. Wang, and P.S. Yu, “Anonymizing Classification Data for Privacy Preservation,” IEEE Trans. Knowledge and Data Eng., vol. 19, no. 5, pp. 711-725, May 2007.

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

[14] X. Zhang, C. Liu, J. Chen, and W. Dou, “An Upper-Bound Control Approach for Cost-Effective Privacy Protection of Intermediate Data Set Storage in Cloud,” Proc.

http://www.ijcsjournal.com Volume 2, Issue 1, No 4, 2014. ISSN: 2348-6600.
Reference ID: IJCS-049. PAGE NO: 278-286.

Ninth IEEE Int'l Conf. Dependable,Autonomic and Secure Computing (DASC '11), pp. 518-525, 2011.

[15] I. Roy, S.T.V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel,"Airavat: Security and Privacy for Mapreduce," Proc. Seventh USENIX Conf. Networked Systems Design and Implementation (NSDI'10), p. 20, 2010.

[16] K.P.N. Puttaswamy, C. Kruegel, and B.Y. Zhao, "Silverline:Toward Data Confidentiality in Storage-Intensive Cloud Applications,"Proc. Second ACM Symp. Cloud Computing (SoCC '11), 2011.

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

[18] V. Ciriani, S.D.C.D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi,and P. Samarati, "Combining Fragmentation and Encryption to Protect Privacy in Data Storage," ACM Trans. Information and System Security, vol. 13, no. 3, pp. 1-33, 2010.

[19] S.B. Davidson, S. Khanna, T. Milo, D. Panigrahi, and S. Roy,"Provenance Views for Module Privacy," Proc. 30th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS '11), pp. 175-186, 2011.

[20] S.B. Davidson, S. Khanna, S. Roy, J. Stoyanovich, V. Tannen, and Y. Chen, "On Provenance and Privacy," Proc. 14th Int'l Conf.Database Theory, pp. 3-10, 2011.