Inter-cloud Data Integration System Considering Privacy and Cost

Yuan Tian, Biao Song, JunYoung Park, Eui-Nam Huh

Department of Computer Engineering
Internet Computing and Network Security Lab
KyungHee University Global Campus, South Korea
E-mail: {ytian, bsong, parkhans, johnhuh}@khu.ac.kr

Abstract. In spite of all the advantages delivered by cloud computing, still many challenges are hindering the migration of customer software and data into the cloud. Whenever information is shared in the cloud, privacy and security questions may arise. Although many technologies have been proposed in order to meet users' requirements from privacy concern, however, at the same time, with the increasing number of processed data, the cost for privacy protection also increases dramatically. In this paper, we present a privacy-aware inter-cloud data integration system considering tradeoff between the privacy requirements from users and the charging for those data protection and processing. In contrast to existing data sharing techniques, our method is more practical as the cost for technical supporting privacy must be considered in the commoditized cloud computing environment.

Keywords: Cloud computing, Privacy, Service pricing

1 Introduction

Cloud computing becomes increasingly pervasive and more and more people are starting to take advantage of the power of the cloud these days. Comparing with the traditional systems, which are masked behind firewalls and other gateway boundaries and attackers must do intensive intelligence gathering to know that they exist, cloud computing, is highly visible and are designed to be accessible to anywhere by anyone.

Similar to the real world utilities, nearly all the services provided in cloud computing are on-demand and highly commoditized, based on consumers' usage and quality of service expectations, they just need to pay for those services like water, electricity, gas and telephony [6], rather than investing heavily to maintain their own computing infrastructure.

Cloud computing fundamentally shifts the traditional "desktop as a platform" to "internet as a platform", however, at the same time, arises more problems than ever before, especially when valuable data from thousands of users in a single site are being attacked [2], which also means, whenever information is shared in the cloud, privacy and security questions may arise. When a user stores his data in a third party

like cloud computing provider, those data may have fewer or weaker privacy protections than when the data just remains in the possession of this user.

However, there are few privacy laws apply to restrict the disclosure of customers or employees' personal information from a business to the cloud provider. Even privacy laws apply to particular categories of customer or employee information, disclosure to a cloud provider may not be restricted, as current laws that protect electronic communications may apply differently to different aspects of cloud computing [1]. We can only enjoy the full benefits of Cloud computing if we can address the very real privacy and security concerns that come along with storing sensitive personal information in databases and software scattered around the Internet [1].

In order to meet the growing requirement for privacy management, many privacy technologies have been proposed. Those technologies meet users' requirements from privacy concern, however, at the same time, with the increasing number of processed data, the cost for privacy protection also increases dramatically.

In [4], Yau et al present a method which could safeguard the privacy of data in a very secure way. They provided a privacy preserving repository to accept integration requirements from users, help data sharing services share data, collect and integrate the required data from data sharing services, and return the integration results to users. The main contribution of their work is that the processing of data is kept securely in both data sharing services and repository: data is randomized before sending to repository and encryption/decryption are used from information releasing in the repository.

However in the above method, they did not consider the practicability of uploading all the unprocessed data to the repository, as in the real scenario, the high cost for transmitting data is even more unbearable compare with keeping the privacy of data which is not that important [8, 9]. As a vital building block in many fields, privacy is possible to be a new service in the cloud environment. Consumers do not have to concern about how to protect their privacy by which technology, instead, according to consumers' requirements, the privacy services can be offered by providers to execute their applications only if consumer pay for that [3]. Both service provider and consumer would like to pay different price for those privacy services with different protection assurances according to the importance of their data.

Thus, based on the above analysis, we present a privacy-aware inter-cloud data integration system considering tradeoff between the privacy requirements from users and the charging for those data protection and processing. In general, it is up to customers to decide on a strategy of how to get a service fulfilled on the basis of their personal feeling of the importance of their data, and the cost for processing data.

The structure of the rest of the paper is as follows. Section 2 introduces a scenario that is used as a running example throughout the paper. A component-based view of the proposed system is presented in Section 3 and the design approach is given in Section 4, and the last section of this paper presents our conclusions.

2 A Motivating Example

This section presents a scenario used throughout the paper and we use this motivating example to show how our system works. The scenario is a revised version of the case study proposed in [4, 7].

In a healthcare system there are multiple collaborated clouds which participate in processing, sharing and integrating data. We assume for the purpose of getting the menu which is proposed to ulcer suffering, the clouds may contains the data from medical research institutes, hospitals and pharmacies. For simplicity, only four databases are considered: a disease record database *T1(Disease, Patient)* which storing patient's names and corresponding diseases which they are suffering, an identification information database *T2(ID, Patient)* which stores patients' names and their IDs, a hospital database *T3(ID, Drug, Menu)* storing hospitalization information and a pharmacy database *T4(Disease, Drug)* which stores popular drugs for each disease.

Table 1 The databases which used in the motivating example

Disease	Patient
Tuberculosis	Bree
Aids	Bob
Ulcer	Ada
Diabetes	Alice

(a)

ID	Patient
10001	Alice
10002	Tom
10003	Susan
10004	Bree

(b)

ID	Drug	Menu
10001	D1	M3
10003	D2	M3
10004	D3	M2
10005	D1	M1

(c)

Disease	Drug
Ulcer	D4
Cancer	D2
Flu	D1
Diabetes	D3

(d)

(a) Disease records T1 (b)Identification Information T2 (c)Hospital T3 (d)Pharmacy T4

Now we are going to express our motivating example by the following four SQL queries which shown in Table 2. Query Q1, Q2 and Q3 generates three temporary tables Tmp1, Tmp2 and Tmp3 respectively, and the final results are from the last query Q4.

Table 2 The integrated query

Q1 -> Tmp1 Select *T1.Patient* From *T1* where *T1.Disease*= "Ulcer"	Q2 -> Tmp2 Select T2.ID From Tmp1, T2 where Tmp1.Patient=T2.Patient
Q3 -> Tmp3 Select T4.Drug From T4 where T4.Disease= "Ulcer"	Q4 Select T3.Menu From Tmp2, Tmp3 and T3 where T3.ID=Tmp2.ID and T3.Drug=Tmp3.Drug

As some queries may need other queries' results as inputs, the repository randomizes those results by using Hush function, which avoid the need for the repository to know that results but still keep the mapping relation between data. For example, we replace the Q1's results {Bob, Alice} by {H(Bob), H(Alice)} which protects Q1's results without disturbing Q2, as the hashed name usually remains unique, the repository can easily evaluate Q2 by comparing H(Tmp1. Patient) and H(T2.Patient). Since H(Susan) is not in the Q1's hashed result{H(Bob), H(Alice)},

the repository can find the patient Susan whose ID is 10004 is not an Aids nor diabetes patient.

3 The proposed system

Our proposed system aims to mediate between users' privacy preferences and the cost for privacy protection. The overall architecture of our system is illustrated in Fig.1. The user sends his integration requirements to the repository cloud and only the required data for users' integration request is collected. The query plan wrapper in the repository cloud will correctly construct a query plan for users' integrated query, decompose the query into a set of sub-queries, and discover corresponding service providers. The data which was conducted or collected in the query plan wrapper are then sent to the query allocator.

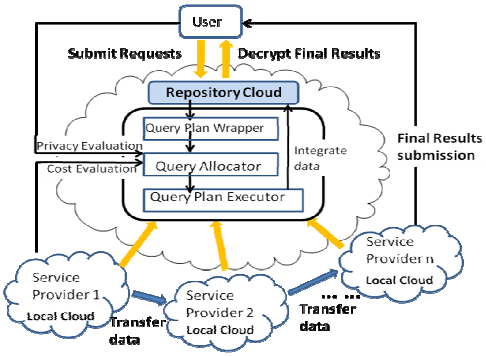

Fig.1 System Architecture

Each sub-query could be executed in local cloud or repository cloud. For the former case, the sub-query will be executed in the service provider which stores the corresponding database. Thus, only the results for the sub-query are returned to the repository cloud or transferred to the successive service provider. The price for processing data in local cloud is relative cheaper, however, it may not be secure as both the data storing and processing are done in the same service provider. While the later one is to fetch data in local cloud, randomize all of those data and then send to the query plan executor for further processing. Executing query in query plan executor guarantees secure protection as the data is stored and processed separately. Therefore, the cost for processing data in repository cloud is high as it requires randomizing all the data in the local service provider and transfers it back to the repository cloud.

In order to help users find a balance between privacy insurance and the cost for privacy protection, the query plan executor decides at where the sub-query should be executed. Before sending those sub-queries to the query plan executor, three separating phases are proposed:

- ✓ Privacy evaluation: based on user's query, allows user to express their privacy preferences by setting risk values to decide the importance of data and the relationship between those data.

- ✓ Cost evaluation: here, the pricing mechanism is that the service provider in repository cloud should estimate a price for processing data, whereas other providers which located outside the repository cloud provides the costs for both processing data and uploading data to the repository cloud.
- ✓ Query Allocation: the query allocator chooses an optimal strategy according to user's preferences in the above two steps, and sends it to the query plan executor to enforce that method. At last, the requested data is sent back to user from the query plan executor.

4 System Design

In this section, we presented the process of our system implementation. Before discussing the functionality of our system, some basic definitions are introduced in order to establish a common ground of concepts.

Firstly, the query plan wrapper converts user's integrated query to a query plan graph G. For a integrated query, the query plan graph $G=\{V, E, C\}$ is a labeled directed acyclic graph. Among which, $V=\{v_1, v_2, \ldots, v_m,\}$ is a set of nodes where each v_i represents an intermediate search result; $E=(e_1, e_2, ..., e_l)$ is a set of edges where each edge $e_{ij}=(v_i, v_j)$ represents a data integration relation between v_i an v_j. $C=(c_1, c_2, ..., c_l)$ is a set of labels attached to each $e_{ij} \in E$ and each label $c_{i,j}=(op, attr1, attr2) \in C$ specifies that the data of v_i and v_j is integrated by the data integration operator *op* between v_i's attribute attr1 and v_j's attribute attr2. Generally, the operator *op* can be any binary comparison operator chosen from $\{=, \neq, >, <\}$ or any aggregate operator chosen from $\{SUM, AVG, MAX, MIN\}$.

The motivating example in Section 2 can be represented by the query plan graph shown in Fig. 2. The query plan graph is decomposed to several sub-graphs $\{G_1, G_2, G_3, ..., G_n\}$ where each sub-graph represents a sub query in Table 2. The overlapping of sub-graphs shows that the immediate result from a sub-query is used as the input for another sub-query.

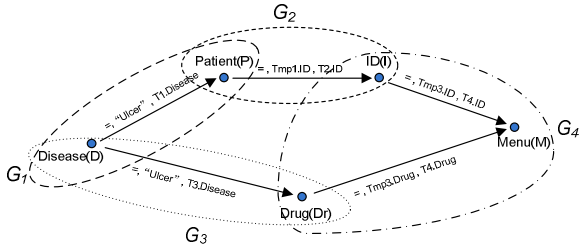

Fig.2 The query plan graph from the motivating example

We use a matrix D_{m*m} to represent the disclosure condition of the searching results and the relationships between them. For each $d_{ii} \in D$, $d_{ii}=0$ denotes the information of v_i is not disclosed, while $d_{ii}=1$ denotes the information of v_i can be

known by service provider. Similarly, $d_{ij} = 0$ denotes the relationship between v_i and v_j is not disclosed while $d_{ij} = 1$ shows the relationship between v_i and v_j that can be known by service provider. An instance of data disclosure condition matrix is shown in Fig.3.

	D	P	I	Dr	M
D	1	1	0	0	0
P	0	1	0	0	0
I	0	0	0	0	0
Dr	0	0	0	0	0
M	0	0	0	0	0

Fig.3 An instance of a data disclosure condition matrix

As the service providers which locate in different clouds may collude each other in order to get more information that they are not supposed to get, i.e., if the relationship between disease and patient is disclosed to one service provider and the relationship between patient and ID is disclosed to another service provider, then is possible for them to find out the relationship between disease and ID. Thus, we define $d_{ij} = 1$ if there exists a path P between v_i and v_j where the starting node of P is v_i, the ending node of P is v_j, and every edge in P has been disclosed.

As we mentioned before, each sub-query can be executed in local cloud or repository cloud, for the sub-graph G_k, we set $A(G_i) = 1$ to denote the former case and $A(G_i) = 0$ to denote the later one where $A(G_i)$ is the sub-allocation decision of sub-graph G_i. As each sub-graph contains nodes and edges, if the value of sub-graph is set as "1", the values of all the nodes and edges within the sub-graph should be also set as "1". For the nodes which may appear in several sub-graphs with different setting values "0" and "1", we should consider this node (represents intermediate results) is executed in local cloud and set it's value as "1" as it may be disclosed in the local cloud.

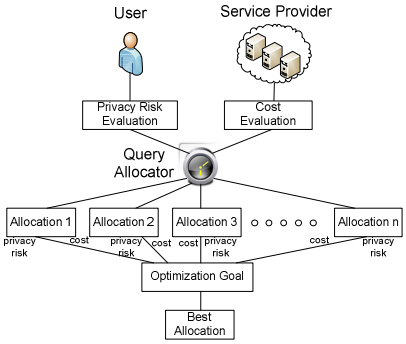

Fig.4. Process of the proposed system

The process of our proposed approach is briefly presented in Fig.4. Based on user's query, user and service providers first submit their evaluation for risk and cost evaluation respectively. After that, query allocator decides an optimal allocation which provides minimum trade-off value between user's risk and service cost.

4.1 Risk Evaluation

Before submitting user's sub-requests to the query plan executor, user could express his privacy preference by giving risk value to those intermediate nodes. Each node represents the results from the previous query and the values on the nodes show how sensitive that user care about those data or the relationships between the data.

	D	P	I	Dr	M
D	4	9	9	1	1
P	–	5	∞	–	5
I	–	–	3	–	1
Dr	–	–	–	3	1
M	–	–	–	–	0

Fig.5. An instance of a penalty matrix

The risk values for the intermediate nodes are presented in a risk matrix R_{m*m} in Fig.5. The $r_{ii} \in R$ denotes the risk value on intermediate search result v_i and $r_{ij} \in R$ denotes the risk value on the relationship between the intermediate search result v_i and v_j. Every value in the matrix is an integer ranging from 0 to 9, or $+\infty$. For example, from Fig.5 we can see that the user gives the "Menu" information away and set the risk value to "0" as he may think this information is not that important. Whereas he cares which patients are suffer from which diseases as this kind of data disclosure may offend patients' privacy, so the risk value on the relationship between "Patient" and "Disease" is set to infinity.

	D	P	I	Dr	M
D	1	1	0	0	0
P	0	1	0	0	0
I	0	0	0	0	0
Dr	0	0	0	0	0
M	0	0	0	0	0

*

	D	P	I	Dr	M
D	4	9	9	1	1
P	–	5	∞	–	5
I	–	–	3	–	1
Dr	–	–	–	3	1
M	–	–	–	–	0

= 18

Fig.6 Risk Penalty Value

Given disclosure condition matrix D and risk matrix R, the risk penalty value RPV can be calculated by the formula RPV= $\sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij} \times r_{ij}$. An instance for showing how to calculate the risk penalty value is presented in Fig. 6.

4.2 Cost Evaluation

At the same time, the service providers should fix the service price considering different situations. Our pricing mechanism includes four kinds of costs:

1) Data randomization cost

As we mentioned before, the sub-query could be executed in the repository cloud in order to assure the data can be stored and processed separately. Thus, all the data in the local service provider needs to be randomized before transmitting to the repository cloud. Service provider evaluates this price according to the amount of data.

2） Cost for uploading all data

The user who wants to process the data in the repository cloud needs to upload all of the randomized data. This cost is for submitting those randomized data.

3） Query execution cost

The cost is for executing sub-query in local cloud or repository cloud. The query execution cost varies based on the amount of data.

4） Cost for uploading intermediate result

This cost is for transmitting the results of the sub-query after executing it locally. According to user's preference, the results can be transferred to next local server provider or to the repository cloud for further processing.

Fig.7 shows an example of our pricing model. We use notation $C_r(G_i)$ to denote the cost for processing a sub-query G_i in the repository cloud while $C_l(G_i)$ represents the cost for processing a sub-query G_i in the local cloud. If the user decides to execute sub-query in repository cloud, the service provider should firstly randomize the whole data and send it to repository cloud to execute the sub-query. So $C_r(G_i)$ includes the cost for randomization data, uploading all data, and executing query.

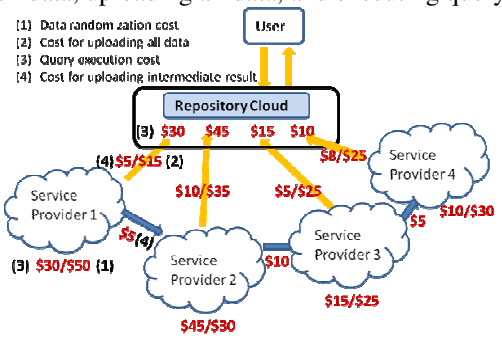

Fig.7 Pricing Mechanism

For example, from Fig.7 we can calculate that $C_r(G_1) = \$50 + \$15 + \$30 = \$95$, which means the total cost for executing G_1 in repository cloud is \$95. On the other hand, if G_i is executed in local cloud, only the results of that sub-query will be submitted to the repository cloud or transferred to the successive service provider. So $C_l(G_i)$ includes the cost for executing query and uploading intermediate results.

We can get $C_l(G_1) = \$30 + \$5 = \$35$ in the example in Fig.7, so the total cost for executing G_1 in service provider 1 is \$35.

4.3 Query Allocation

Based on the above analysis, the query allocator provides several strategies for user considering both price and privacy. User could select a service with lowest cost allocation which subjects to privacy risk constraint, or a service with lowest privacy protection which subjects to cost constraint, or a service considering trade-off between cost and privacy penalty. Based on user's choice, every possible allocation is examined and the best allocation will be selected.

Given a set of sub-allocation decision $\{A(G_1), A(G_2), ..., A(G_n)\}$, the disclosure condition matrix D can be generated by combining all the sub-allocation decisions as follows:

$$\exists(v_i \in G_k \ \& \ A(G_k) = 1) \xrightarrow{\forall i,k} d_{ii} = 1$$

$$\exists(e_{ij} \in G_k \ \& \ A(G_k) = 1) \xrightarrow{\forall i,j,k} d_{ij} = 1$$

$$\exists(d_{ii} = 1 \ \& \ d_{jj} = 1 \ \& \ P = \{d_{ik_1} = 1, d_{k_1k_2} = 1, \ldots, d_{k_t j} = 1\}) \xrightarrow{\forall i,j,k_1,\ldots,k_t} d_{ij} = 1$$

Using RPV= $\sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij} \times r_{ij}$ we can get the risk penalty value of the allocation. Then the total cost is calculated by using $\sum_{i=1}^{n} C(G_i)$ $where \begin{cases} C(G_i) = C_r(G_i) & when\ A(G_i) = 0 \\ C(G_i) = C_l(G_i) & when\ A(G_i) = 1 \end{cases}$

As there are four sub-queries in our motivating example, totally $2^4 = 16$ allocations are provided. In Table 3, we list all the possible allocations associated with the cost and privacy penalty. Suppose that the user prefers to select a service that produces lowest privacy penalty value while costs less than \$250. From Table 3 we can see that the allocation which meets this requirement is {0,0,1,1}, which means, sub-query 1 and 2 should be executed in the repository cloud while sub-query 3 and 4 should be executed in the local cloud.

Table 3. Allocation Results

$\{A(G_1), A(G_2), A(G_3), A(G_4)\}$	Cost	Penalty	$\{A(G_1), A(G_2), A(G_3), A(G_4)\}$	Cost	Penalty
{0, 0,0,0}	335	0	{1,0,0,0}	275	18
{0,0,0,1}	288	8	{1,0,0,1}	228	26
{0,0,1,0}	290	8	{1,0,1,0}	230	22
{0,0,1,1}	205	14	{1,0,1,1}	183	28
{0,1,0,0,}	280	∞	{1,1,0,0}	220	∞
{0,1,0,1}	233	∞	{1,1,0,1}	173	∞
{0,1,1,0}	235	∞	{1,1,1,0}	175	∞
{0,1,1,1}	188	∞	{1,1,1,1}	128	∞

5 Conclusions and future work

In this paper, a privacy-aware inter-cloud data integration system is presented, which strikes a balance between the privacy requirements from users and the cost for data protection and processing. In contrast to existing data sharing techniques, our method is more practical as the cost for technical supporting privacy must be considered in the commoditized cloud computing environment. This work is still evolving and in the future work, we will consider more conflicted situation like how the system works under the trustable service providers.

Acknowledgment

"This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)" (NIPA-2010-(C1090-1021-0003)) "This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. 2010-0016959)".

References

1. Privacy in the clouds: risks to privacy and confidentiality from cloud computing, http://www.worldprivacyforum.org/pdf/WPF_Cloud_Privacy_Report.pdf. (Accessed on April 1, 2010)
2. Steve Mansfield-Devine, Danger in the clouds, Network security, http://www.webvivant.com/dangers-in-the-cloud.html (Accessed on May 15, 2010)
3. Chee Shin Yeo, Srikumar Venugopal, Xingchen Chu, Rajkumar Buyya, Autonomic Metered Pricing for a Utility Computing Service, 2008.
4. Stephen S. Yau et al, a privacy preserving repository for data integration across data sharing services, IEEE transactions on services computing.
5. E. Michael Maximilien, Tyrone Grandison, Tony Sun, Dwayne Richardson, Sherry Guo, Kun LiuPrivacy-as-a-Service: Models, Algorithms, and Results on the Facebook Platform, In Web 2.0 Security and Privacy 2009, held in conjunction with the 2009 IEEE Symposium on Security and Privacy. Oakland, California. May 2009.
6. I. Foster, C. Kesselman (Eds.), The Grid 2: Blueprint for a new computing infrastructure, morgan Kaufmann, San Francisco, CA, USA, 2003.
7. Gerardo Canfora, Elisa Costante, Igino Pennino, and Corrado Aaron Visaggio, A three-layered model to implement data privacy policies, Computer Standards & Interfaces, Elsevier Science Publishers B. V. , Amsterdam, The Netherlands, The Netherlands, 2008, pp. 398-409.
8. Chee Shin Yeoa, Srikumar Venugopalb, Xingchen Chua, and Rajkumar Buyya, Autonomic metered pricing for a utility computing service, Future Generation Computer Systems, 2009.
9. Tarry Singh, Pavan Kumar Vara, Smart Metering the Clouds, 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, IEEE Computer Society, Washington, DC, USA, pp. 66-71.