Singaporean Journal of Scientific Research(SJSR)
Journal of Technology and Engineering System(IJTES)
Vol.8.No.3 2016 Pp.164-175
available at :www.iaaet.org/sjsr
Paper Received : 08-03-2016
Paper Accepted: 19-04-2016
Paper Reviewed by: 1.Prof. Cheng Yu 2. Dr.M. Akshay Kumar
Editor : Dr. Chu Lio

SECURE TRANSACTION OF CROSS-CLOUD AND SERVICE COMPOSITION FOR BIG DATA APPLICATION USING HIRESOME-II

M.Padmavathy[1] and Dr.S.Babu[2]
PG Scholar[1], Associate Professor[2]
Department of Computer Science and Engineering
IFET College of Engineering
Villupuram, India

.

ABSTRACT*:* **Cloud computing promises a scalable infrastructure for processing big data applications such as medical data analysis. Cross-cloud service composition provides a concrete approach capable for large-scale big data processing. However, the complexity of potential compositions of cloud services calls for new composition and aggregation methods, especially when some private clouds refuse to disclose all details of their service transaction records due to business privacy concerns in cross-cloud scenarios. Moreover, the credibility of cross-clouds and on-line service compositions will become suspicional, if a cloud fails to deliver its services according to its ''promised'' quality. In view of these challenges, we propose a privacy-aware cross-cloud service composition method, named HireSome-II (History record-based Service optimization method) based on its previous basic version HireSome-I. In this paper provides the review for comparing various QoS to secure cloud.**

INDEX TERMS*:* Cloud, Service Composition, QoS, Big Data, Transaction History Records**.**

I. INTRODUCTION

Cloud storage is gaining popularity recently. In enterprise settings, we see the rise in demand for data outsourcing, which assists in the strategic management of corporate data. It is also used as a core technology behind many online services for personal applications. Nowadays, it is easy to apply for free accounts for email, photo album, file sharing and/or remote access, with storage size more than 25GB (or a few dollars for more than 1TB). Together with the current wireless technology, users can access almost all of their files and emails by a mobile phone in any corner of the world.

Many cloud service providers (e.g,Amazon, Google, Microsoft, IBM, etc) are now available in the market to provide cloud services such as Governance as a Service (gaas), Business as a Service (baas), Infrastructure as a Service (iaas), Platform as a

Service (paas) and Software as a Service (saas). Cloud technology stack has also become main stream in enterprise data centers, where private and hybrid cloud architectures are increasingly adopted [1]. However, though cloud computing has tremendous advantages; there are challenges in the area of Quality of Service (qos).

Qos denotes the levels of performance, reliability and availability offered by an application and by the platform or infrastructure that hosts it. Qos is fundamental for cloud users, who expect providers to deliver the advertised quality characteristics, and for cloud providers, who need to find the right tradeoffs between qos levels and operational costs. Any violation of service level agreement (SLA) entails a loss for both cloud providers and cloud users [2].

Over provision is often adopted by providers as an approach to satisfy the SLA, but it fails to optimize. Though qos properties have received constant attention even before the evolution of cloud computing, performance and heterogeneity and resource isolation mechanisms of cloud platforms have significantly complicated qos analysis, prediction and assurance. Thus, several researchers are investigating automated qos management methods that can leverage the high programmability of hardware and software resources in the cloud[5].

Cross-cloud service composition provides a concrete approach capable for large-scale big data processing. Existing (global) analysis techniques for service composition, however, often mandate every participant service provider to unveil the details of services for network-aware service composition, especially the qos information of the services. Unfortunately, such an analysis is infeasible when a private cloud or a local host refuses to disclose all its service in detail for privacy or business reasons [8].

II. RELATED WORKS

A. *Skyline Service Method*

The problem of QoS-based web service selection and composition has received a lot of attention by many researchers. The work of Zeng at al. [3] focuses on dynamic and quality-driven selection of services. The authors use global planning to find the best service components for the composition. They use (mixed) linear programming techniques [4] to find the optimal selection of component services. The goal is to select a set of services, one from each service class, that maximize the overall utility, while satisfying all the specified constraints. The main idea in our approach is to perform a skyline query on the services in each class to distinguish between those services that are potential candidates for the composition, and those that cannot possibly be part of the final solution. First, we briefly introduce skyline queries, and then we describe how we apply them in our approach. We also deal with the problem that arises when the number of services in the skyline is still too large.

Once the problem of QoS-based service composition has been formulated as a constraint optimization problem, MIP techniques can be employed [3]. To cope with this limitation, we first prune all non-skyline services from the MIP model in order to keep its size as small as possible. By focusing only on the skyline services of each service class, we speed up the selection process, while still being able to find the optimal selection, we present a method for selecting representative skyline services in order to address the situation where the number of skyline services K of a certain service class S is too large and thus cannot be handled efficiently. The main challenge that arises is how to identify a set of representative skyline services that best represent all trade-offs of the various QoS parameters, so that it is possible to find a solution that satisfies the constraints and has also a high utility score.

To overcome this problem, in the future work, we plan to develop a method for estimating the difficulty of each composition problem.

B. *GA based approach*

One of the most interesting challenges introduced by web services is represented by Quality Of Service (QoS)-aware composition and late-(binding). Genetic-Algorithms, while being slower than integer programming, represent a more scalable choice, and are more suitable to handle generic QoS attributes. GAs is unconstrained procedures by their nature, and so it is necessary to find ways to integrate constraint-handling. To solve our problem, we have used a simple distance-based penalty approach that has turned successful also in scheduling problems. work aims to propose an approach, based on GAs, to quickly determine a set of concrete services to be bound to the abstract services composing the workflow of a composite service GAs do not impose constraints on the linearity of the QoS composition operators. To let the GA search for a solution of our problem, we first need to encode the problem with a suitable genome. In our case, the genome is represented by an integer array with a number of items equals to the number of distinct abstract services composing our service. Each item, in turn, contains an index to the array of the concrete services matching that abstract service.

GA permits to deal with QoS attributes having non-linear aggregation functions. GA is able to scale-up when the number of concretizations increases to deal with constraints, it is possible to adopt a fitness function with both static and dynamic penalty. Another approach for constraint handling derives from Artificial Intelligence, namely Constraint Logic Programming (CLP). CLP has the advantage of a richer modelling capability, but providing a good model for the problem at hand has implications on the performance of the approach. The peculiarity of CLP is constraint propagation, i.e., constraints are used to identify allowed sub domains for the variable values, which is continuously applied during the search. Obviously, this approach can lead to a more efficient search but it is expensive itself.

Future work will aim to apply the proposed approach to some large-scale service-oriented system, and to perform a thorough comparison of GA with other non-linear technique, such as non-linear integer programming. Multi-objective fitness functions will also be considered as an alternative to single-objective fitness functions where factors are aggregated using a weighted sum.

III. EXISTING SYSTEM

The Existing System furnishes the techniques and algorithms to cluster the QoS history records of each web services using a optimization method based on history records and clustering i.e., QHRC.

A OPTIMIZATION METHOD BASED ON HISTORY RECORDS AND CLUSTERING (QHRC):

It aims at ranking the service composition plans to select a most qualified composition plan to select a most qualified composition plan.We apply the H2D-SC algorithm to cluster the QoS history records of each web service. In each hierarchical level, we use the centroids of the sub-clusters to generate record composition plans, then ranking the service composition plans by the corresponding record composition plans.

Steps for fast clustering method

- At the beginning the QoS history records of each Web service are respectively viewed as full members of a unique huge cluster. The decision whether to split the clusters into more specific sub-clusters is taken by assessing the quality (multiple endogenous and exogenous clusters' properties) of the clusters.
- In the case the cluster has been evaluated worth splitting, the number of sub-clusters to generate is determined depending on cluster's properties by formula, if the clusters selected to generate the next level of some Web services could not be split, and then the clusters will not change in the next hierarchical level.

- The H2D-SC algorithm applies a (soft) clustering algorithm (in the experiments the FCMs algorithm was applied) having the number of sub-clusters generated in step 2 as input so as to get the soft sub-clusters.

The H2D-SC algorithm can be regarded as an extension of the FCMs. The FCMs is the most common fuzzy clustering algorithm. It generates a flat fuzzy partition based on the minimization of an objective function. In the FCMs, the membership degree of a record to a cluster can be interpreted as the probability of occurrence of an event. The H2D-SC algorithm is used to execute information retrieval, While in this paper, we employ this algorithm to the Web service composition problem to cluster the QoS history records for each Web service.

- We use the sub-clusters' centroids QoS history records of each Web service to generate record composition plans. And we calculate the utility scores of the record composition plans by formula.
- We accumulate the utility scores of the corresponding record composition plans to the utility scores of each service composition plan respectively.
- Each sub-cluster has a variable to accumulate the utility scores of the record composition plans which the sub cluster participates in. For each Web service, the sub-cluster with the highest score will be used to generate the next hierarchical level.
- The process iterates until no more clusters are evaluated worth splitting.

IV. PROPOSED MODEL

The proposed system aims at enhancing the credibility of service composition. HireSome-I is incapable of dealing with the privacy issue in cross-cloud serv composition. Compared to HireSom HireSome-II greatly speed up the process selecting a optimal service composition plan, and protects the privacy of a cloud service for cross-cloud service composition.

K-MEANS CLUSTERING ALGORITHM (OR)PADDY-FIELD CLUSTERING ALGORITHM

k-means is one of the simplest unsupervised learning algorithms that solve t he well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center.A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function knows as squared error function given by:

$$J(V)= \sum_{i=1}^{c} \sum_{j=1}^{c_i} (\left|\left| x_i - v_j \right|\right|)^2$$

where,

'$||x_i - v_i||$' is the Euclidean distance between x_i and v_j.

'c_i' is the number of data points in i^{th} cluster.

'c' is the number of cluster centers.

Algorithmic steps for k-means clustering

Let $X = \{x_1,x_2,x_3,\ldots\ldots,x_n\}$ be the set of data points and $V = \{v_1,v_2,\ldots\ldots,v_c\}$ be the set of centers.

1) Randomly select *'c'* cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..

4) Recalculate the new cluster center using:

$$v_i = (1/C_i) \sum_{i=1}^{c_i} x_i$$

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop, otherwise repeat from step 3.

```
Let n be the number of clusters you want
Let S be the set of feature vectors (|S| is the
size of the set)
Let A be the set of associated clusters for
each feature
vector
Let sim(x,y) be the similarity function
Let c[n] be the vectors for our clusters
Init:
Let S' = S
//choose n random vectors to start our
clusters
fori=1 to n
j = rand(|S'|)
c[n] = S'[j]
S' = S' - {c[n]} //remove that vector from S'
so we
can't choose it again
End
//assign initial clusters
fori=1 to |S|
A[i] = argmax(j = 1 to n) { sim(S[i], c[j]) }
End
Run:
Let change = true
while change
change = false //assume there is no change
//reassign feature vectors to clusters
fori = 1 to |S|
a = argmax(j = 1 to n) { sim(S[i], c[j]) }
```

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977 [RIVE78]. The basic technique was first discovered in 1973 by Clifford Cocks [COCK73] of CESG (part of the British GCHQ) but this was a secret until 1997. The patent taken out by RSA Labs has expired.

The RSA cryptosystem is the most widely-used public key cryptography algorithm in the world. It can be used to encrypt a message without the need to exchange a secret key separately.The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

Basic idea

An individual A who wishes to receive messages confidentially will use the pair of integers {e,n} as his/her public key. At the same time, this individual can use the pair of integers {d,n} as the private key. The definitions of n, e, and d are as in the previous subsection.

- Another party B wishing to send a message M to A confidentially will encrypt M using A's public key {e,n} to create ciphertext C. Subsequently, only A will be able to decrypt C using his/her private key {d,n}.
- If the plaintext message M is too long, B may choose to use RSA as a block cipher for encrypting the message meant for A.

The RSA works because:

If n = pq, where p and q are large primes (several hundred digits), then

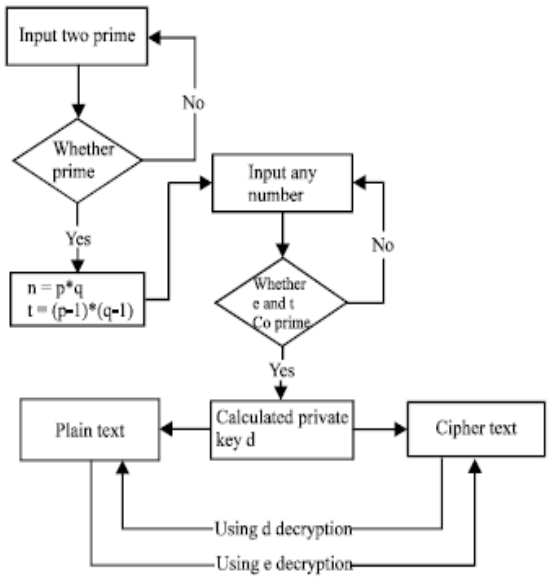
i) Given p and q, we can easily multiply them to obtain n, but

ii) Given n, there is no known way to factor n as pq in any reasonable amount of time.

Key-Generation Algorithm:

1. Generate two large random primes, *p* and *q*, of approximately equal size such that their product n = pq is of the required bit length, e.g. 1024 bits.
2. Compute n = pq and (phi) φ = (p-1)(q-1).

3. Choose an integer *e*, 1 < e < phi, such that gcd(e, phi) = 1. Compute the secret exponent *d*, 1 < d < phi, such that ed ≡ 1 (mod phi).
4. The public key is (n, e) and the private key (d, p, q). Keep all the values d, p, q and phi secret. [We prefer sometimes to write the private key as (n, d) because you need the value of n when using d. Other times we might write the key pair as ((N, e), d).]
   - n is known as the *modulus*.
   - e is known as the *public exponent* or *encryption exponent* or just the *exponent*.
   - d is known as the *secret exponent* or *decryption exponent*.



Operation

The RSA algorithm involves four steps: key generation, key distribution, encryption and decryption.

RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

The basic principle behind RSA is the observation that it is practical to find three very large positive integers e,d and n such that with modular exponentiation for all m:

$$(m^e)^d \mod n = m$$

and that even knowing *e* and *n* or even *m* it can be extremely difficult to find *d*.

Additionally, for some operations it is convenient that the order of the two exponentiations can be changed and that this relation also implies:

$$(m^d)^e \mod n = m$$

Key distribution

To enable Bob to send his encrypted messages, Alice transmits her public key (*n*, *e*) to Bob via a reliable, but not necessarily secret route. The private key is never distributed.

Encryption

Suppose that Bob would like to send message *M* to Alice.

He first turns *M* into an integer *m*, such that $0 \leq m < n$ and $\gcd(m, n) = 1$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext *c*, using Alice's public key *e*, corresponding to

$$c \equiv m^e \mod n$$

This can be done efficiently, even for 500-bit numbers, using modular exponentiation. Bob then transmits *c* to Alice.

Decryption

Alice can recover *m* from *c* by using her private key exponent *d* by computing

$$c^d \equiv (m^e)^d \equiv m \mod n$$

Given *m*, she can recover the original message *M* by reversing the padding scheme.

Key generation

The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers *p* and *q*.

For security purposes, the integers *p* and *q* should be chosen at random, and should be similar in magnitude but 'differ in length by a few digits to make factoring harder. Prime integers can be efficiently found using a primality test.

Compute $n = pq$.

n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

Compute $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1) = n-(p+q-1)$, where φ is Euler's totient function. This value is kept private.

Choose an integer *e* such that **$1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$**; i.e., *e* and φ(*n*) are co prime.

Determine *d* as **$d \equiv e^{-1} \pmod{\varphi(n)}$**; i.e., *d* is the modular multiplicative inverse of *e* (modulo φ(*n*))

- This is more clearly stated as: solve for *d* given $d \cdot e \equiv 1 \pmod{\varphi(n)}$
- *e* having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65{,}537$. However, much smaller values of *e* (such as 3) have been shown to be less secure in some settings.
- *e* is released as the public key exponent.
- *d* is kept as the private key exponent.

The *public key* consists of the modulus *n* and the public (or encryption) exponent *e*. The *private key* consists of the modulus *n* and the private (or decryption) exponent *d*, which must be kept secret. *p*, *q*, and φ(*n*) must also be kept secret because they can be used to calculate *d*.

- An alternative, used by PKCS#1, is to choose *d* matching $de \equiv 1 \pmod{\lambda}$ with $\lambda = \text{lcm}(p-1, q-1)$, where lcm is the least common multiple. Using λ instead of φ(*n*) allows more choices for *d*. λ can also be defined using the Carmichael function, λ(*n*).

Since any common factors of (p − 1) and (q − 1) are present in the factorisation of pq − 1, it is recommended that (p − 1) and (q − 1) have only very small common factors, if any besides the necessary 2.

V. SYSTEM ARCHITECTURE

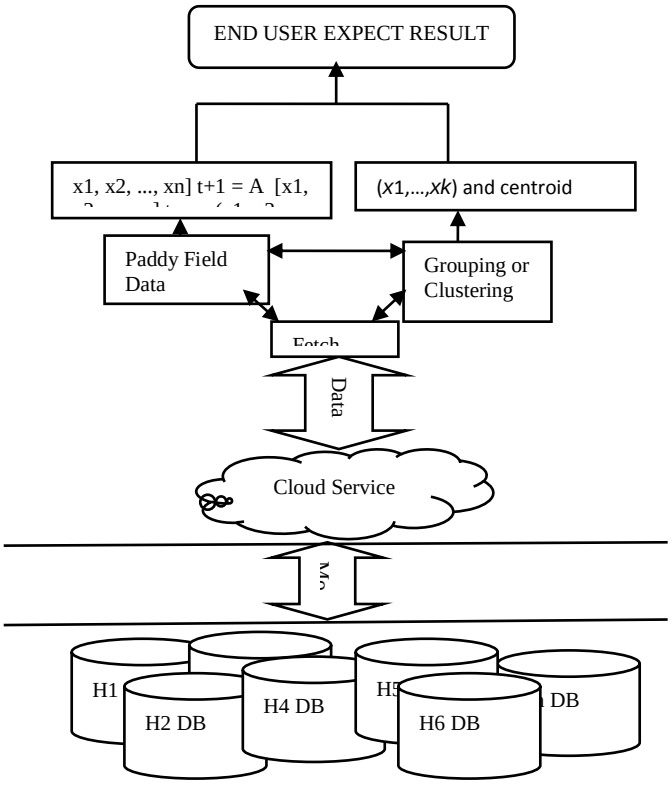


Fig. 1 Proposed System Architecture

The system architecture of our proposed system is been designed as a 3 layer or the 3-tier architecture. The first layer of the architecture contains the details of different hospital databases. The hospital databases may contains the patients details about the problems they have and the amount they have spend for curing their problems. These databases of the various hospitals are been allowed to move towards the cloud where the second layer of the architecture is the cloud service. The clouds service is the place where all the databases are been stored.

The client can fetch the data's from the cloud service. The fetched data can be retrieved and it is allowed to process by using the methodology Paddy-Field clustering / K-means clustering algorithm. Paddy-Field clustering algorithm means that data's are been randomly selected and throwed, with the help of paddy-field clustering algorithm the thrown datas are been grouped and send it to the user. Data's which are been selected before is been selected with the help of user key.

The proposed system includes the following modules:

The proposed system is to design paddy field clustering algorithm for various hospital databases into a cloud system with privacy. The basic modules to be implemented

1. **Cloud First Access :**

This is to access the cloud, an user has to first select the valid client type, the client type here refers to the user, doctor or an any research scholars. If the client who is been trying to access the cloud is not applicable means that the client is not a valid client to access the cloud, once it is not valid the process will get terminated and the user has to give another option.

2. **Authenticate Cloud Access**

This is to check that whether the cloud is been accessed by the valid user or not so that an user has to first login into the system using the valid credentials. The validation is done using the client specific data which is already stored in the database. The access permission is authenticated using the email id and password entered by the client.

3. **Upload sensitive data**

Once the client's authentication is successful by giving a valid username and password or the required details for accessing the cloud, the clients can able to upload the details which has to be stored in the cloud or the client can able to update any new data in the existing one and also the client can make some changes in the existing data as required.

4. **PFC Algorithm Based Data Retrieval**

Once the data has been retrieved as per the client's requirement. The data retrieved and the data which has been feeded from the databases are processing using the paddy-field/k-means clustering algorithm. Paddy-Field clustering algorithm means that datas are been randomly selected and throwed, with the help of paddy-field clustering algorithm the thrown datas are been grouped and send it to the user. datas which are been selected before is been selected with the help of user key.

VI. SYSTEM IMPLEMENTATION

Home page

Provide username and password to get permission for access. This is to access the cloud, an user has to first select the valid client type, the client type here refers to the user, doctor or an any research scholars. Became an authenticated person to request and process the request.

Patient Home page

Design of user process list. User can select from that user menu list. List contains various options as requesting for key, searching for key and to view their details

Patient Details Uploading

Once the user got the security key for viewing the patient details the user can able to upload any new details about the patient by choosing the appropriate public key.

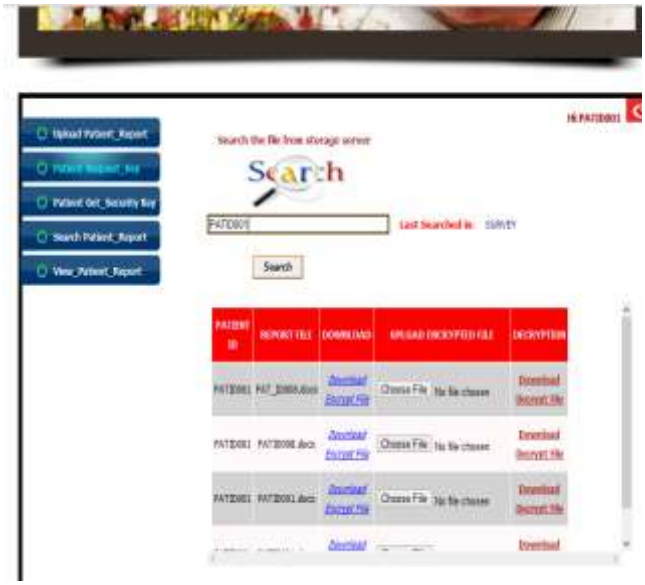
Request Key page

For requesting the key page the patient has to enter his valid e-mail id with his id and the requesting message for searching the user's databases in the cloud

Key Generation

Admin can generate key for secure transferred. It will forward generated key to requested user. The key is been generated by using the RSA algorithm for secure transformation.

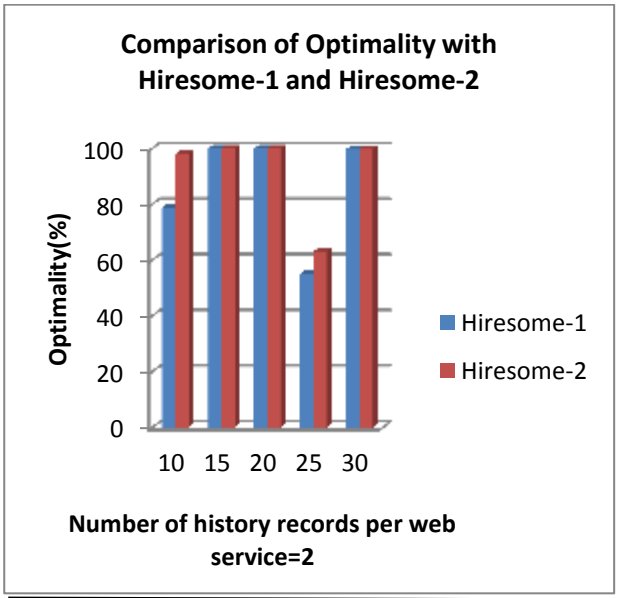
Search Data in Cloud

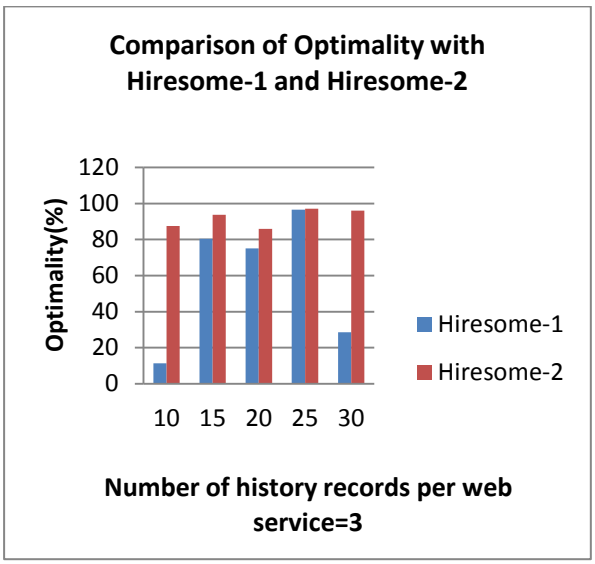


Once the requesting for a page is successfully made the user can able to search their database in the cloud. From his various databases in the cloud the user can able to select the selective data which the user needs.

RESULTS AND DISCUSSION

Comparision of Optimality with Hiresome-1 and Hiresome-2

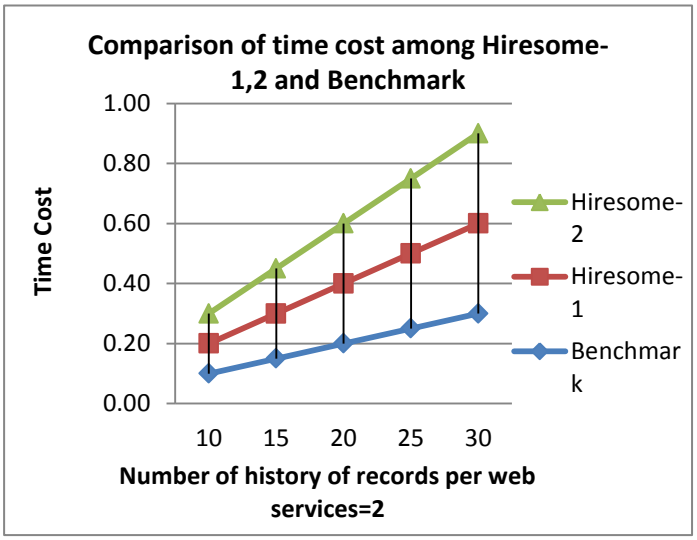


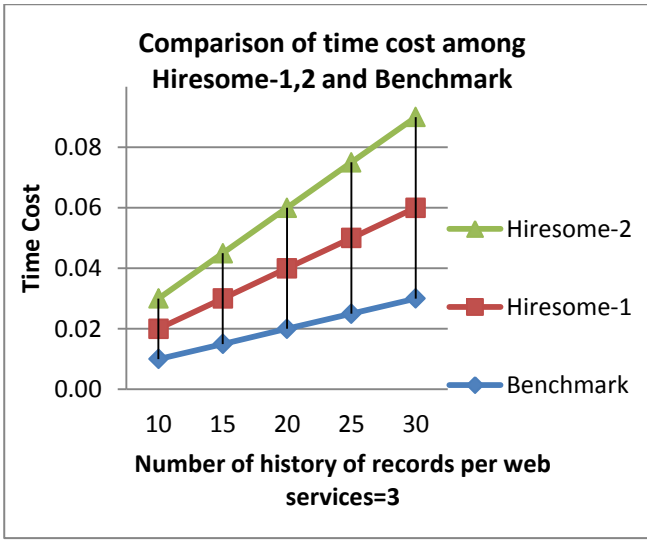


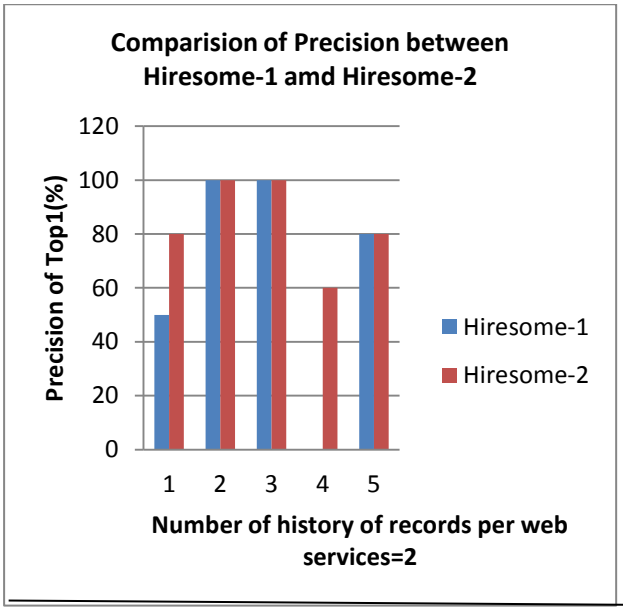
The optimality of HireSome-I and HireSome-II will be computed taking advantage of the following formula: Optimality $U=U/U_{Benchmark}$.

The above chart indicates the average optimality value of HireSome-I and HireSome-II with the variation in their web servicesthe optimality of HireSome-I and HireSome-II performs are close to the Benchmark method, and HireSome-II performs better than HireSome-I.

Comparison of Time Cost with Hiresome-1and Hiresome-2 with Benchmark



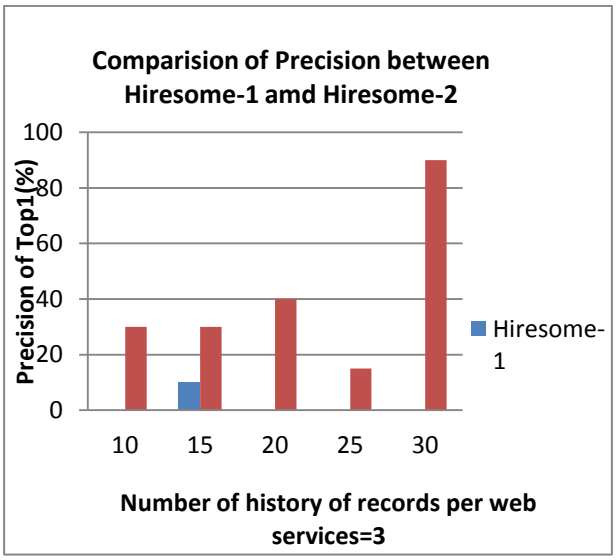




Indicate the evaluation results with these assumptions, the simulation is conducted for 100 times initiated by different set of history records. As per the comparison HireSome-II have the least time to produce the optimal service composition plan.,

While for the Benchmark, the time cost curve increases acutely with the increasing of the number of history records per web service. Therefore, compared to the Benchmark, HireSome-I and HireSome-II have an advantage in large-scale service composition.

Comparison of Precision between Hiresome-1 amd Hiresome-2



Aim at investigating the probability that the final composition plan respectively produced by HireSome-I and HireSome-II is as same as the optimal one computed by the Benchmark. HireSome-I and HireSome-II are nearly equal to each other in their performance. HireSome-II is greatly superior to HireSome-I in performance.

VII. CONCLUSION

Hiresome-II is developed for privacy-aware cross-cloud service composition for processing Big Data applications. It can effectively promote cross-cloud service

composition in the situation where a cloud refuses to disclose all details of its service transaction records for business privacy issues. Our approach achieves two advantages. Firstly, reduces the time complexity as only some representative history records are recruited, which is highly demanded for big data application. Secondly, protects cloud privacy as a cloud is not required to unveil all of its transaction records, which accordingly protects privacy for big data.

FUTURE WORK

Future work will aim to apply the proposed approach to some large-scale service-oriented system, and to perform a thorough comparison of GA with other non-linear technique, such as non-linear integer programming. Multi-objective fitness functions will also be considered as an alternative to single-objective fitness functions where factors are aggregated using a weighted sum.

REFERENCES

[1]. MengShunmei, Liu Zhenxing, Dou Wanchun,"A QoS-Aware Service Optimization Method Based on History Records and Clustering", in *Second International Conference on Cloud and Green Computing*,pp 89-96, 2012

[2]. W. Lin, W. Dou, X. Luo, and J. Chen, ''A History Record-Based Service Optimization Method for QoS-Aware Service Composition,'' in *Proc. 2011 IEEE ICWS*, pp. 666-673, , July 2011

[3]. M. Alrifai, D. Skoutas, and T. Risse, ''Selecting Skyline Services for QoS-Based Web Service Composition,'' in *Proc. 19th Int'l Conf.WWW*, pp. 11-20, Apr. 2010.

[4]. Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, Maria Luisa Villani, "An Approach for QoS aware Service Composition based on Genetic Algorithms",*Proc. Intl Conf. Cooperative Information Systems*, pp. 1069-1075, June 2005

[5]. M. C. Jaeger, G. Muhl, and S. Golze, "QoS-Aware Composition of Web Services: An Evaluation of Selection Algorithms," *Proc. Intl Conf. Cooperative Information Systems*, 2005.

[6]. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, pp. 311-327, 2004.

[7]. W. Lin, W. Dou, X. Luo, and J. Chen, "A History Record-based Service Optimization Method for QoS-Aware Service Composition," *IEEE International Conference on Web Services*, pp. 666- 673, July 2011.

[8]. G. Bordogna, G. Pasi. "A quality driven Hierarchical Data Divisive Soft Clustering for information retrieval," *International Journal for Knowledge- Based Systems*, Vol. 26, No. 1, pp. 9-19, 2012.

[9]. R. Jurca, B. Faltings, and W. Binder, "Reliable QoS Monitoring Based on Client Feedback," *Proceeding of the 17th International World Wide Web Conference (WWW07)*, pp.1003-1011, 2007.

[10]. K. Kritikos, D. Plexousakis, "Mixed-Integer Programming for QoS-based web service matchmaking," *IEEE Transactions on Services Computing*, Vol. 2, No. 6, pp. 122-139, 2009.

[11]. S. Y. Hwang, E. P. Lim, C. H. Lee, and C. H. Chen, "Dynamic Web Service Selection for Reliable Web Service Composition," *IEEE Transactions on Services Computing*, Vol. 1, No. 2, pp. 104-116, 2008.

[12.]. M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, ''Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131-143, Jan. 2013.