Open access Journal International journal of Emerging Trends in Science and Technology

Retrieving the Data from Cloud Using Differential Query Services

Authors
Jnana Jyothi.K[1], Jayanthi.M.G[2]
[1]M.tech, Dept. of M.tech, Cambridge Institute of Technology, B'lore, INDIA
E-mail: *Vinus19@gmail.com*
[2]Asst. Prof, Cambridge Institute of Technology, B'lore, INDIA
E-mail: *Jaykumar_singh@rediffmail.com*

Abstract
Cloud computing refers to the delivery of computing resources over the Internet. In a cost efficient cloud environment, a user will experience a degree of delay while retrieving information from the cloud. In such an environment, two main issues facing by the users are efficiency and privacy. This work first focuses on the private keyword based file retrieval scheme, permits users to get the files without losing any information from an unsecured server on demand. The disadvantage of this scheme is that, it leads to heavy querying cost. This work presents a new scheme called EIRQ (Efficient Information Retrieval for Ranked Query), based on an ADL (Aggregation and Distribution Layer), to lessen the querying cost. In EIRQ, user can select a rank to his query, where a highest ranked query will retrieve a higher percent of matched files and vice versa. This is beneficial whenever cloud retrieves large numbers of matched files, but the user is in need of only few files. Valuations are performed to check the efficiency of the scheme in the cloud.
Keywords: *Cloud Computing, Ranking, Aggregation and Distribution layer*

INTRODUCTION

Compute clouds are commonly used by many different users those rely on the existing computing infrastructure to deploy their workloads[2]. Because of the advantages of cloud like flexibility, cost-effectiveness and scalability, many of the enterprises try to share the data with the cloud. For example, if an organisation starts to use the cloud. Thus, the staff of the organisation are authorised to share data with the cloud. While sharing, each file is associated with a group of keywords and Staff can retrieve files by requesting the cloud with the use of keywords on demand. In this cloud environment, protection of privacy of the users becomes a major issue. There are 2 types of user privacy: access privacy and search privacy. Access Privacy is nothing but the cloud should not know anything about files returned to the user. Search Privacy is nothing but the cloud should not know anything about which files are being searched by the user. A simple solution for the protection of privacy of the user is to request for every file in the collection. By this cloud will be able to get in which file/files the user is interested.

Private searching [4] permits a user to get files which are of interest from an unsecured server without losing any useful information on demand. The cloud has to execute the

query on all the files shared. It leads to degradation of performance because the cloud has to execute thousands of queries on a collection of files.

To make private searching suitable for cloud computing, middleware layer, aggregation and distribution layer is deployed between user and cloud inside an organization [1]. It will perform two main functionalities: Aggregation of user requests and Distribution of results. By this, computation cost will be reduced, because the cloud has to execute a single query regardless of number of users requesting. This work introduces a new and better scheme called differential query services, in order to get the required percentage of matched files on demand by giving a rank to his request. This is beneficial, if there are large numbers of files matching a user's request but the user is in need of only a few of them. To better understand, consider if cloud holds 2,000 files, where {F1, ... , F1000} and {F1001, . . . ,F2000} are associated with keywords "A, B" and "A, C" respectively. When Bob needs to get 20% files consisting of keywords "A, B", and Alice needs to get 5% of the files consisting of keywords "A, C". The COPS scheme will retrieve 2, 000 files. In EIRQ, the cloud will retrieve 400 files.

Efficient Information retrieval for Ranked Query (EIRQ), the proposed system, lets users to input a rank to his request to get the required percent of matched files. The EIRQ is based on the construction of a privacy preserving mask matrix.

Objectives of this work:

1) EIRQ schemes provide a cost-efficient way to make private searching suitable in cloud environment.
2) The EIRQ schemes ensures privacy of the users, while providing a differential query service that allows each user to retrieve matched files only on demand.
3) Valuations are performed to check the efficiency of the scheme in the cloud.

RELATED WORK

This work aims to protect privacy and to provide efficiency using differential query services from the cloud. Similar research was found in the stream of private searching [3]. In searchable encryption [5], user searches on encrypted data. While in private searching it searches on unencrypted data based on keyword. Private searching [3],[4], helps to filter out data without considering privacy of users. Private searching returns a buffer with size O (f log (f)) when f' files matches with request. Each file has survival rate associated with it. Survival rate is nothing but the probability of a file being recovered by the user successfully. As per Paillier cryptosystem, the files that will not match a query, will be having less survival rate, thus reduces the communication cost to O (f). The demerit of private searching scheme is that the computation, querying cost and communication costs grow linearly with the number of users executing queries. This is not suitable in large scale cloud environment. The previous work was to make private searching work in cloud environment. Private searching will retrieve all the matched files, leads to waste of bandwidth when user in need of few of the files. A differential query service is proposed to solve this problem.

EXISTING SYSTEM

Private keyword-based file retrieval scheme is the existing system introduced by Ostrovsky. This scheme permits users to get the files without losing any information from an unsecured server on demand. Disadvantages: 1. Computational cost is

more, since the cloud has to execute the query on every file in a collection. 2. It incurs heavy querying overhead.

SYSTEM ARCHITECTURE

As shown in Fig. 1. There are 3 entities in the system model: users, ADL, and the cloud.

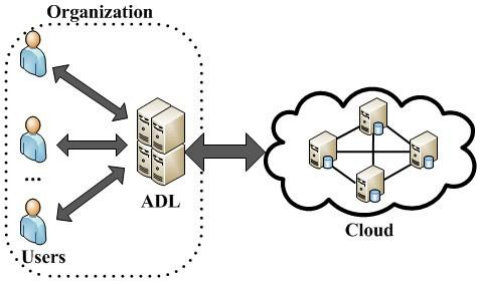

Fig. 1 System model

In an organization, ADL will be deployed to authorize its staff to share files with the cloud. User's send their requests to the middleware server i.e. ADL, that will combine the requests from multiple user's and sends a single request to the cloud. The cloud executes the single query on collection of files and returns all the matched files to the aggregation and distribution layer. ADL will send the results to all the requested users. To aggregate user requests, the ADL has to wait for some time before running EIRQ schemes, which will incur some querying delay.

Differential query service is introduced, again to reduce the communication cost. To get the required percentage of matched files, user can input a rank to his request. This is beneficial, if there are large numbers of files matching a user's request but the user is in need of only a few of them. User privacy is divided into search and access privacy. In this work, queries are divided into different ranks and thus privacy of rank selected by the users also needs to be ensured. Privacy of the rank means, the cloud has to provide differential query services regardless of the rank selected by the users. Design goals:

Cost efficiency: The users can get the matched files on demand to reduce the communication cost.

User privacy: The cloud knows nothing regarding what the user searching for, which file has been returned and rank chosen by the user.

EFFICIENT INFORMATION RETRIEVAL FOR RANKED QUERY (EIRQ)

Two issues should be solved are: Firstly, It requires determining the dependency between rank of a query and the percentage of matched files to be returned. The queries are divided into 0 ~ r ranks. Rank-0 is the top most rank and Rank-r is the lower most rank. This work, determines the dependency between the rank of query and the matched files returned by granting Rank-i queries to recover (1 − i/r) percent of matched files. So, Rank-0 will get 100% of matched files, and Rank-r will not return any of the files. Secondly, it requires determining chance of a file being returned as matched file. This work, determines the chance of a file being returned based on the many queries matching that file. Specifically, ranking of keyword will be done based on highest rank of queries selected it and then ranks all the files by the highest rank of file keywords. If the file rank is i, then the chance of filtering a file is i/r. Therefore, Rank-0 files will be returned with probability 1 and Rank-r files will not be returned. Since unneeded files have been filtered before returning files to ADL, the files present in the buffer will be having probability 1 with high survival rate. EIRQ Efficient mainly consists of four algorithms, since Query Gen and Result Divide algorithms are easily understood; provide the details of algorithms Matrix Construct and File Filter.

The working Process is as follows:

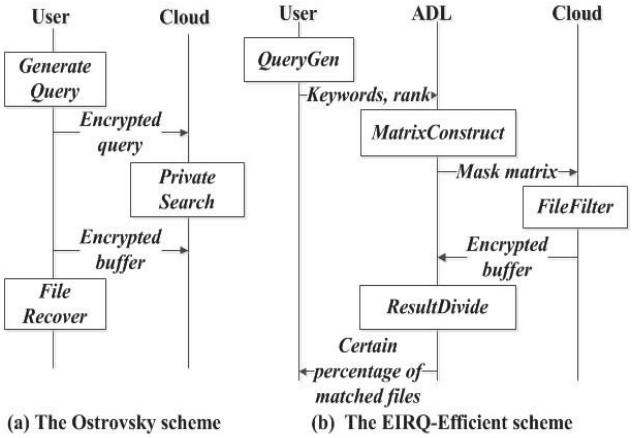

Fig. 2 Working Process

Step 1:
The user executes the Query Gen algorithm to generate a query that consists of keywords and the rank. That will be sent without encrypting to the middleware server called ADL.

Step 2:
After combining queries from all the users, the ADL executes the Matrix Construct algorithm to construct mask matrix and that will be sent to the cloud. Consider d is the no. of keywords in the dictionary, and r is the query rank, the mask matrix M is a d-row and r-column matrix. M[i, j] denotes the element in the i-th row and the j-th column, and if l is the highest rank of queries that choose the i-th keyword Dic[i] from the dictionary. M is constructed as follows: for the i-th row of M that corresponds to Dic[i], M[i, 1], . . . , M[i, r − l] are adjusted to 1, and M[i, r − l + 1], . . . M[i, r] are fixed to 0, then each bit is encrypted using the ADL's public key pk. For the rows that correspond to Rank-l keywords, the ADL sets the first r − l elements to 1. Given Fj with Rank-l, when selected any number k, the probability of all the k-th elements of all the rows that correspond Fj's keywords being 0 is l/r, this can be determined by the highest rank of Fj keywords.

Step 3:
The cloud executes the File Filter algorithm to filter the files which are unneeded. A buffer will be returned to Aggregation and distribution layer that includes files those matching with the request. Specifically, the cloud multiplies the k-th elements of the rows that correspond to Fj keywords to get cj, where k = j mod r. To get ej, it powers| Fj | to cj and maps the c-e pair into multiple entries of a buffer.

Step 4:
The ADL executes the Result Divide algorithm to send search results to all the users requested. File contents are recovered by executing File Recover algorithm. To let the ADL correctly deliver files to all of the users, the cloud required to send keywords along with the file content. By this, ADL will get to know files matching the user requests and then sends the files to particular user.

MODULE DESCRIPTION

There are 4 modules: The user, ADL, Storage and Ranked Queries.

User: In this user module, the unauthorised user can register with the cloud. If user is a registered user, he can request the data from the cloud on demand
by using keywords associating with the file. User can select the rank to his own request in order to retrieve the required percentage of files.

Storage: In this module, only the cloud admin will be having permissions to share the files with cloud along with the file details.

ADL: In this module, the ADL will aggregate the queries from multiple users and sends a combined query to cloud. After the cloud process, the cloud returns a buffer containing matched files. Then the ADL will distribute the files to the respective users.

Ranked Queries: In this module, the user can select the rank of the query in order to get required percent of matched files. The lower ranked query will retrieve less percent matched files and vice versa.

SCHEMES OF EIRQ

In this section, the EIRQ scheme and one of the two extensions have been explained. The EIRQ schemes are EIRQ-Efficient, EIRQ-Simple and EIRQ-Privacy. This work differentiates EIRQ-Efficient, EIRQ-Simple. The EIRQ-Efficient is based on the construction of a privacy-preserving mask matrix. Before mapping the matched files to a buffer, the cloud can filter unneeded files. The survival rate of a file is determined by the size of the buffer and no. of times mapped. Therefore, the basic idea of other two schemes is that, for each rank i ranging from (0, . . . , r), the ADL sets the buffer size and the no. Of times mapped to make the file survival rate qi approach 1-i/r.

The EIRQ-Efficient Scheme

This scheme works as shown in the following algorithm. If file rank is i, then the chance of that file being filtered is i/r. So, Rank-0 will return the files with the probability 1 and Rank-r will not return anything. Since unneeded files have been filtered before returning files to the Aggregation and Distribution Layer.

Algorithm 1 The EIRQ-Efficient scheme

MatrixConstruct (run by the ADL with public key pk)
for $i = 1$ to d do
  set l to be the highest rank of queries choosing $Dic[i]$
  for $j = 1$ to r do
    if $j \leq r - l$ then
      $M[i, j] = E_{pk}(1)$
    else
      $M[i, j] = E_{pk}(0)$
adjust γ and β so that file survival rate is 1
FileFilter (run by the cloud)
for each file F_j stored in the cloud do
  for $i = 1$ to d do
    $k = j \mod r$; $c_j = \prod_{Dic[i] \in F_j} M[i, k]$; $e_j = c_j^{|F_j|}$
    map (c_j, e_j) γ times to a buffer of size β

The files present in the buffer will be having probability 1 with high survival rate. Duplicate files will not be returned.

The EIRQ-Simple Scheme

The working process of EIRQ-Simple is same as in Fig. 2- (b). This scheme works as shown in the following algorithm.

Algorithm 2 The EIRQ-Simple scheme

MatrixConstruct (run by the ADL with public key pk)
for $i = 0$ to $r - 1$ do
  for $j = 1$ to d do
    if $Dic[j]$ is in Rank-i queries then
      $Q_i[j] = E_{pk}(1)$
    else
      $Q_i[j] = E_{pk}(0)$
  adjust γ_i and β_i so that survival rate of Rank-i files is $q_i = 1 - i/r$
FileFilter (run by the cloud)
for $i = 0$ to $r - 1$ do
  for each file F in the cloud do
    for $j = 1$ to d do
      $c = \prod_{Dic[j] \in F} Q_i[j]$; $e = c^{|F|}$
      map (c, e) γ_i times to $\mathbb{B}_i$ of size β_i

The only difference between EIRQ-efficient and simple are in Matrix Construct and FileFilter algorithms. Queries are classified into 0-r ranks; ADL sends r combined queries to the cloud, each with a different rank. For Qi, the ADL sets the j-th bit to an encryption of 1 if the j-th keyword Dic[j] from the dictionary is chosen by at least one Rank-i query. The cloud generates r buffers, each with a different file survival rate. For Bi, the ADL adjusts the mapping time i and the buffer size i so that the survival rate of files in Bi is qi = 1-i/r, where 0<= i<= r-1. The main drawback of EIRQ-simple is that it returns redundant files when there are files satisfying more than one ranked query. For example, if Fi is of interest by Rank-0 and Rank-1 queries, it will be returned twice (in Rank-0 buffer and Rank-1 buffer, respectively), which wastes the bandwidth.

EIRQ MODEL PERFORMANCE ANALYSIS

This section compares 2 EIRQ schemes based on file survival rate and transfer time.

A. File Survival Rate

The queries are classified into 0 ~ 4 ranks, Rank-0, Rank-1, Rank-2, Rank-3, and Rank-4 should retrieve 100%, 75%, 50%, 25%, 0% of matched files, respectively.

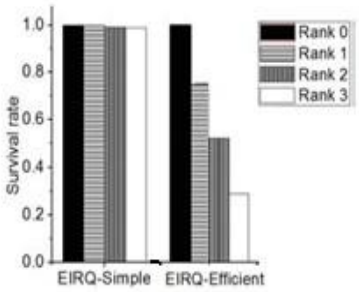

Fig.3 File Survival rate under Ostrovsky Settings

As shown in Fig.3, the failure rate in EIRQ-Simple is lower than i/r, and thus, EIRQ-Efficient has file survival rate higher than the desired value of 1−i/r (about 25% and 50% of files are redundantly returned). Only EIRQ-Efficient filters a certain percentage of matched files before mapping them to a buffer, provides differential query services.

B. Transfer Time in a Real Cloud

To verify the feasibility of our schemes, we deploy our program in Amazon EC2, to test the transfer-in (receiving query) and transfer-out (sending buffer) time at the cloud. The local machine has an Intel Core 2 Duo E8400 3.0 GHz CPU and 8 GB Linux RAM. We subscribe EC2 amzn- ami-2011.02.1.i386-ebs (ami-8c1fece5) AMI and a small type instance with the following specifications: 32-bit platform, a single virtual core equivalent to 1 compute unit CPU, and 1.7 GB RAM. The average bandwidth from EC2 to the local machine is 33.43 MB/s, and from the local machine to EC2 is 42.98 MB/s.

CONCLUSION AND FUTURE ENHANCEMENT

This work proposed two EIRQ schemes based on an ADL to provide efficiency and privacy using differential query services. With these schemes, a user can able to decide the percentage of files to be returned from the cloud by choosing queries of different ranks. The EIRQ schemes make the private searching technique to a cost-efficient cloud environment by further reducing the communication cost. For future work, framework can be designed to provide flexible ranking mechanism for EIRQ schemes.

ACKNOWLEDGEMENT

This work reported here has been guided by Mrs. JAYANTHI.M.G, Assistant professor, Department of Computer Science & Engineering, CAMBRIDGE INSTITUTE OF TECHNOLOGY, BANGALORE. Her support is gratefully acknowledged.

REFERENCES

[1] Qin Liu, Chiu C.Tan, Jie Wu and Fellow (2013) “Towards Differential Query Services in Cost-Efficient Clouds” IEEE Transactions On Parallel and Distributed Systems, vol. 20, no.10, pp-1- 11.

[2] P.Mell and T.Grance, The nist definition of cloud computing (draft), NIST Special Publication, 2011.

[3] X. Yi and E. Bertino, Private searching for single and conjunctive keywords on streaming data, in Proc. of ACM Workshop on Privacy in the Electronic Society, 2011.

[4] Private searching on streaming data, Journal of Cryptology, 2007.

[5] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, Searchable symmetric encryption: improved definitions and e cient

constructions, in Proc. of ACM CCS, 2006.

[6] R.Ostrovsky and W. Skeith, Private searching on streaming data, in Proc. of CRYPTO, 2005.

[7] Shucheng Yu, Cong Wang, Kui Ren† and Wenjing Lou "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing" IEEE Transactions On Parallel and Distributed Systems, vol. 20, no.8, pp-1-9.