TOWARDS DIFFERENTIAL QUERY SERVICES IN COST-EFFICIENT CLOUDS

G. Mahidhar Sai , Dr. M. Sreedevi
Student, Dept. of Computer Science, SVU College of CM & CS, SV University, Tirupati.

Abstract- **Cloud computing as an emerging technology trend is expected to reshape the advances in information technology an Efficient Information Retrieval for Ranked Queries (EIRQ) scheme is recovery of ranked files on user demand. An EIRQ worked based on the Aggregation and Distribution Layer (ADL). An ADL is act as mediator between cloud and end-users. An EIRQ scheme reduces the communication cost and communication overhead. Mask Matrix is used to filter out as what user really wants matched data before recurring to the Aggregation and Distribution Layer (ADL). A user can retrieve files on demand by choosing queries of different ranks. This feature is useful when there are a large number of matched files, but the user only needs a small subset of them. Under different parameter settings, extensive evaluations have been conducted on both analytical models and on a real cloud environment, in order to examine the effectiveness of our schemes To avoid small scale of interruptions in cloud computing, follow two essential issues: - Privacy and Efficiency. Private keyword based file retrieval scheme was anticipated by Ostrovsky.**

Index Terms- **Cloud Computing, Cost Efficiency, Differential Query Services, Privacy.**

I. INTRODUCTION

Cloud computing technology is a most necessary technology for information technology. Many more organizations are used cloud computing for outsource sharing. The organizations needs to submit access the services of cloud and authorizes organizations workers to split files in the cloud. Each and every file is described by place keywords. The authorized workers at an organization can access the data of their benefits by querying from the cloud with particular keywords. In Cloud environment, user privacy can be protected on every transaction. User privacy is categorized by 2 types. They are search privacy and access privacy. Search privacy is a process of searching, but cloud doesn‟t know anything about what user really searching for and Access privacy is searching technique. Here cloud knows about what user really searching on search engine. Private searching was introduced by ostrovsky scheme allows to users to recover data from the un-trusted servers n leakage of data. Ostrovsky scheme is lofty computational outlay, because the cloud need to process keywords in the each and every file in the cloud. The user can send a query to every time to process the query. Because of this process the cloud is over headed queries from the many users from different organization. Through this process the communication and computation beyond the expectation. Private searching was proposed by Ostrovsky et al. Which allows a user to retrieve files of interest from an untrusted server without leaking any information otherwise; the cloud will learn that certain files, without processing, are of no interest to the user.

Commercial clouds follow a pay-as-you-go model, where the customer is billed for different operations such as bandwidth, CPU time, and so on. Solutions that incur excessive computation and communication costs are unacceptable to customers. To make private searching applicable in a cloud environment, our previous work designed a cooperate private searching protocol (COPS), where a proxy server, called the aggregation and distribution layer (ADL), is introduced between the users and the cloud. The ADL deployed inside an organization has two main functionalities: aggregating user queries and distributing search results. Under the ADL, the computation cost incurred on the cloud can be largely reduced, since the cloud only needs to execute a combined query once, no matter how many users are executing queries. Furthermore, the communication cost incurred on the cloud will also be reduced, since files shared by the users need to be returned only once. Most importantly, by using a series of secure functions, COPS can protect user privacy from the ADL, the cloud, and other users. In this paper, we introduce a novel concept, differential query services, to COPS, where the users are allowed to personally decide how many matched files will be returned.

This is motivated by the fact that under certain cases, there are a lot of files matching a user‟s query, but the user is interested in only a certain percentage of matched files In the Ostrovsky scheme, the cloud will have to return 2,000 files. In the COPS scheme, the cloud will have to return 1,000 files.

Aggregation and Distribution Layer: An ADL is deployed in an organization that authorizes its staff to share data in the cloud. The staff members, as the authorized users, send their queries to the ADL, which will aggregate user queries and send a combined query to the cloud. Then, the cloud processes the combined query on the file collection and returns a buffer that contains all of matched files to the ADL, which will distribute the search results to each user. To aggregate sufficient queries, the organization may require the ADL to wait for a period of time before running our schemes, which may incur a certain querying delay. In the supplementary file, we will discuss the computation and communication costs as well as the querying delay incurred on the ADL.

Ranked Queries: To further reduce the communication cost, a differential query service is provided by allowing each user to retrieve matched files on demand. Specifically, a user selects a particular rank for his query to determine the percentage of matched files to be returned. This feature is useful when there are a lot of files that match a user‟s query, but the user only needs a small subset of them.

In our scheme, the cloud only needs to return 200 files. Therefore, by allowing the users to retrieve matched files on demand, the bandwidth consumed in the cloud can be largely reduced. Efficient Information retrieval for Ranked Query (EIRQ), in which each user can choose the rank of his query to determine the percentage of matched files to be returned. The basic idea of EIRQ is to construct a privacy-preserving mask matrix that allows the cloud to filter out a certain percentage of matched files before returning to the ADL. This is not a trivial work, since the cloud needs to correctly filter out files according to the rank of queries without knowing anything about user privacy. Our key contributions are as follows:

- We propose three EIRQ schemes based on the ADL to provide a cost-efficient solution for private searching in cloud computing.
- The EIRQ schemes can protect user privacy while providing a differential query service that allows each user to retrieve matched files on demand.
- We provide two solutions to adjust related parameters; one is based on the Ostrovsky scheme, and the other is based on Bloom filters.
- Extensive experiments were performed using a combination of simulations and real cloud deployments to validate our schemes.

The remainder of this paper is organized as follows. Architecture in SectionII Proposed Model in SectionIII. We conduct evaluations in SectionIV. Finally, we conclude this paper in SectionV.

Because of this process to reduce the communication cost and query overhead.

A. Module Description

Differential Query Services: We introduce a novel concept, differential query services, to COPS, where the users are allowed to personally decide how many matched files will be returned. This is motivated by the fact that under certain cases, there are a lot of files matching a user‟s query, but the user is interested in only a certain percentage of matched files. To illustrate, let us assume that Alice wants to retrieve 2% of the files that contain keywords “A, B”, and Bob wants to retrieve 20% of the files that contain keywords “A, C”. The cloud holds 1,000 files, where $\{F_1, \ldots, F_{500}\}$ and $\{F_{501}, \ldots, F1000\}$ are described by keywords “A, B” and “A, C”, respectively. In the Ostrovsky scheme, the cloud will have to return 2, 000 files. In the COPS scheme, the cloud will have to return 1, 000 files. In our scheme, the cloud only needs to return 200 files. Therefore, by allowing the users to retrieve matched files on demand, the bandwidth consumed in the cloud can be largely reduced.

Efficient Information Retrieval for Ranked Query: We propose a scheme, termed Efficient Information retrieval for Ranked Query (EIRQ), in which each user can choose the rank of his query to determine the percentage of matched files to be returned as shown in Fig.1. The basic idea of EIRQ is to construct a privacy preserving mask matrix that allows the cloud to filter out a certain percentage of matched files before returning to the ADL. This is not a trivial work, since the cloud needs to correctly filter out files according to the rank of queries without knowing anything about user privacy. Focusing on different design goals, we provide two

II. ARCHITECTURE

Co-operate searching protocol (cops) is like a proxy server called as aggregation and distribution layer (ADL) is placed inside an organization. This ADL is act as a mediator between the cloud and an organization. The functioning of ADL is the aggregation and distribution. The ADL only reduces the computation cost.

extensions: the first extension emphasizes simplicity by requiring the least amount of modifications from the Ostrovsky scheme, and the second extension emphasizes privacy by leaking the least amount of information to the cloud.

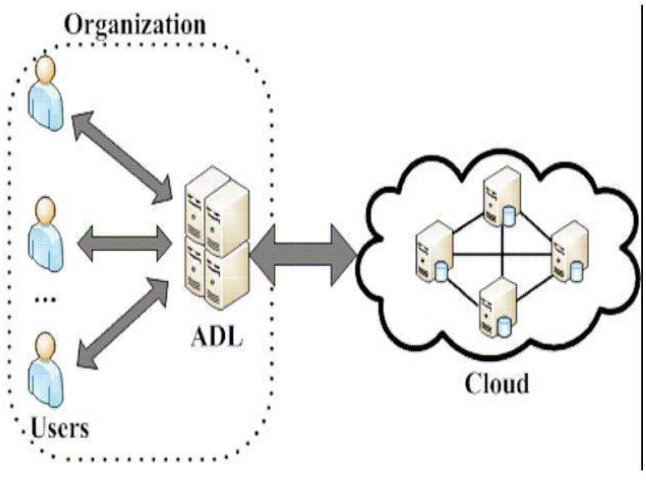

Fig.1. Architecture of EIRQ.

The working of an ADL is the many users can send many queries to ADL. Then adl can aggregate the different user‟s queries makes into a single query and then sends to cloud. The cloud will process the query sends response to ADL. Then the adl will distribute the results to particular users.

III. PROPOSED MODEL

Here introduce a major concept differential query services. Where users are sends the queries to the cloud and process the query sends results to users as shown in Fig.2. Lot of files is matched users query. But the user doesn‟t want that files, only they interested on certain percentage of files.

- This process is going on both wired network and wireless network also. First send request from the user to cloud for establishment of a connection form the cloud. Then authorized user should have their own login name and passwords.
- After login to user Generate a query. This query is encrypted into 0‟s and 1‟s and then sends to cloud. At the cloud side Private Search has been done. So those find out the matched files.
- Cloud sends the matched files to encrypted buffer. Then Files are recovered at the user side. This scheme is very query overhead as well as every time accesses the broadband connection. This process is more costly to accessing files at every query.

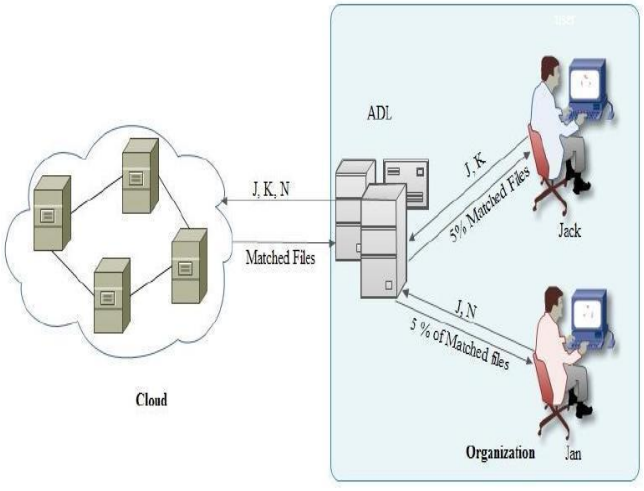

Fig.2. EIRQ Model.

In the proposed model have the cloud, organization and ADL. ADL is placed inside the organization based on requirement of number users. In this model used only single ADL inside an organization. Assume an organization have two users. They are Jack and Jan. They want files from the cloud. The Jack and Jan want files which are starts with the letters J, K and J, N respectively. The design goals of this scheme are Cost Efficiency and User Privacy. We achieve these goals by using Bloom Filters. Ostrovsky Scheme: The Ostrovsky scheme is a process of accessing the files from cloud to clients as shown in Fig.3. This process has the following steps:

EIRQ Scheme: The EIRQ scheme is a process of recover the files from cloud to clients as shown in Fig.4. This process has the following steps:

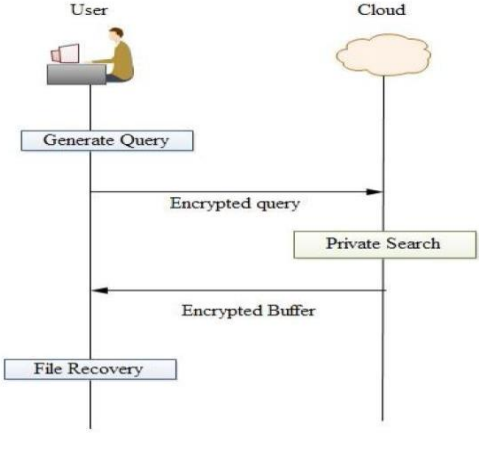

Fig.3. Working process of Ostrovsky Scheme.

- Ostrovsky Scheme having the user and cloud. The users are only authorized from the cloud network, and then only accessing is possible otherwise it is not possible.

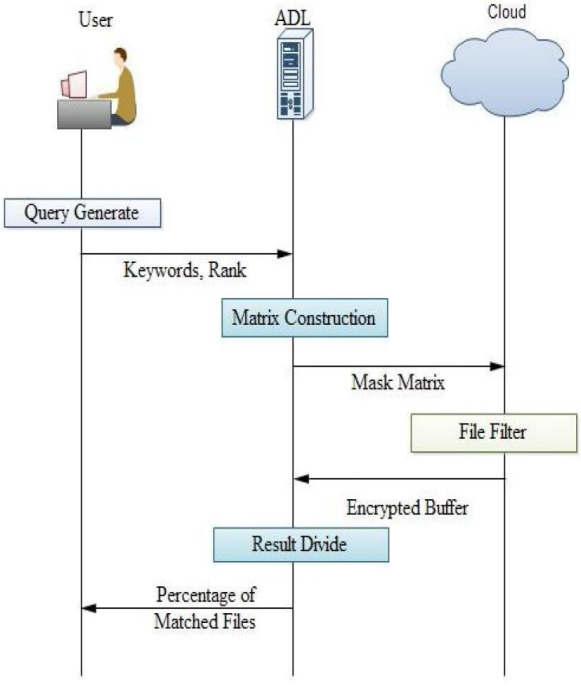

Fig.4. Working process of EIRQ Scheme.

- The EIRQ Scheme having the user and cloud. The users are only authorized from the cloud network, and then only accessing is possible otherwise it is not possible.

- This process is going on both wired network and wireless network also. First send request from the user to ADL for establishment of a connection form the ADL. Then authorized user should have own login name and passwords.
- After login to user generate a query. This query is encrypted into 0‟s and 1‟s and then sends to ADL. At the ADL side Matrix Construct Algorithm has been done based on that Keywords and Ranks. This process we called as Aggregation.
- After the aggregation process, ADL sends the Mask Matrix to Cloud. At cloud side File Filter Algorithm has been done. This algorithm filter out the files based on the Ranks and keywords.

IV. EVALUATION

In this section, we will compare three EIRQ schemes from the following aspects: file survival rate and computation/ communication cost incurred on the cloud. Then, based on the simulation results, we deploy our program in Amazon Elastic Compute Cloud (EC2) to test the transfer-in and transfer-out time incurred on the cloud when executing private searches. Note that the energy performance trade-off is crucial to the success of cloud computing, and existing energy-saving techniques are hard to directly extend to a cloud environment. As part of our future extensions, we will evaluate the consumed energy overhead in the cloud to verify the effectiveness of our schemes. We use No Rank to denote unranked queries under the ADL. The summary of the experiment parameters are shown in Table 1.

TABLE I: Parameters

Notation	Description	Value
$\|F\|$	File content	$1KB$
$\|w\|$	Keyword content	$1KB$
n	The number of users	1-100
d	The number of keywords in Dic	100
k	The number of keywords in each query	1-5
w	The number of keywords in each file	1-5
t	The number of files stored in the cloud	1,000
r	The lowest user rank	4
α	Threshold value	0.1

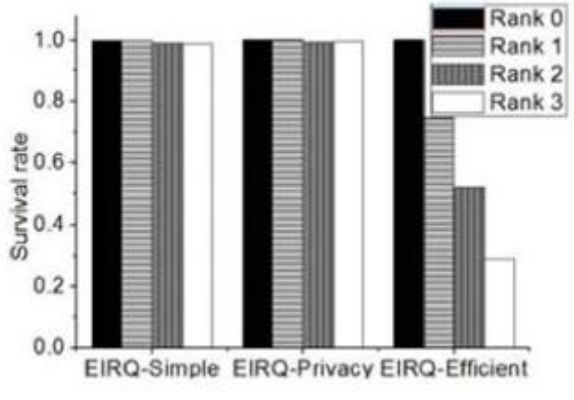

Fig.5. File survival rate under Ostrovsky setting.

A. File Survival Rate

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

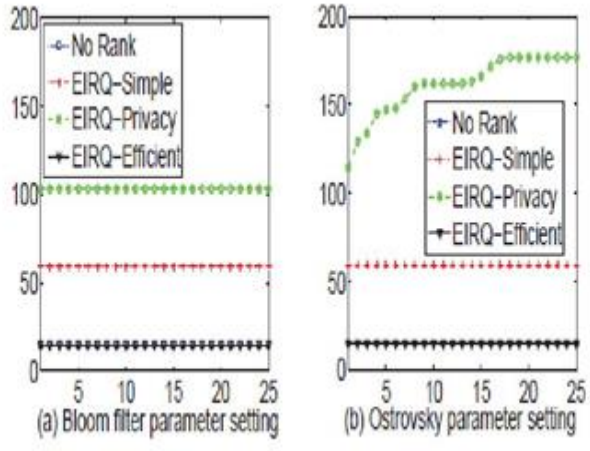

Fig.6. File survival rate under Bloom filter setting.

However, in Fig.5, the real failure rate in EIRQ-Simple and EIRQ-Privacy under the Ostrovsky parameter setting is much lower than i/r, and thus, the real file survival rate is higher than the desired value of $1 - i/r$ (about 25% and 50% of files are redundantly returned to users); Only EIRQ-Efficient, which filters a certain percentage of matched files before mapping them to a buffer, provides differential query services. Under the Bloom filter parameter setting, we first obtain corresponding mapping times. Specifically, for file survival rate 100%, 75%, 50%, 25%, we have the optimal mapping times 7, 2, 1, 0.4, respectively. Based on these values, the buffer size can be calculated different schemes. In practice, γ and β must be integers. Thus, we use $\lfloor\gamma\rfloor$ and

$\lfloor\beta\rfloor$ to replace the corresponding values. Using these parameters, the file survival rates for different ranks are shown in Fig.6, where three EIRQ schemes can provide differential query services, and no bandwidth is wasted in each EIRQ scheme. Therefore, in terms of file survival rate, the Bloom filter parameter setting can achieve better performance than the Ostrovsky parameter setting.

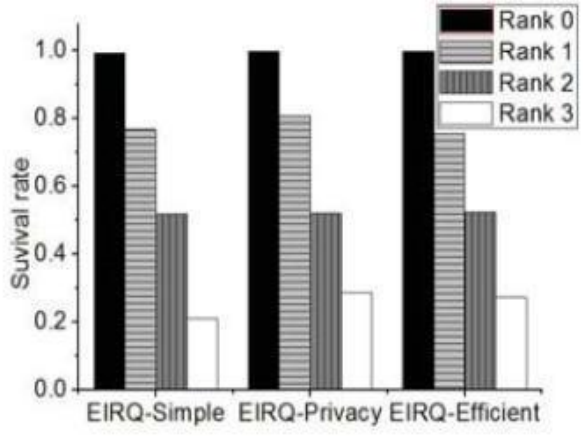

Fig.7. Comparison of computational cost at the cloud the x-axis denotes the number of queries in each rank, and the y-axis denotes the computation time (s). (a) Bloom filters parameter setting. (b) Ostrovsky parameter setting.

B. Computational Cost

As the computational cost is mainly determined by the number of exponentiations performed by the cloud, which is almost the same under the Bloom filter and the Ostrovsky parameter settings. In order to justify the analyses, we will compare the computational cost between No Rank and three EIRQ schemes. The comparisons of computational cost on the cloud are shown in Fig. 5, where the number of queries in each rank ranges from 1 to 25. In Fig. 7-(a), under the Bloom filter parameter setting, the computational cost is approximately 14.807s in No Rank, 59.274s in EIRQ Simple, 101.075s in EIRQ-Privacy, and 14.861s in EIRQ Efficient. In Fig.7-(b), under the Ostrovsky parameter setting, the computational cost approximately ranges from 14.8270s to 14.8788s in No Rank, from 59.1671s to 59.3838s in EIRQ-Simple, from 114.0475s to 176.5107s in EIRQ-Privacy, and from 14.8664s to 14.9269s in EIRQ Efficient. In both settings, EIRQ-Privacy consumes the most computation cost, and EIRQ-Efficient, like No Rank, consumes the least computation cost.

C. Communication Cost

As the communication cost mainly depends on the buffer size generated by the cloud, which is calculated in different ways under different parameter settings. Furthermore, the buffer size depends on the number of files that match the queries, which is different when users have different common interests, i.e., the average number of common keywords
among user queries. Therefore, in different parameter settings, we will analyze the buffer size under different common interests. In the following experiments, 1 common keyword, 2 common keywords, and 4 common keywords denote that the average common keywords among user queries are 1, 2, and 4, respectively; random keywords denote that each user randomly chooses keywords for its query.

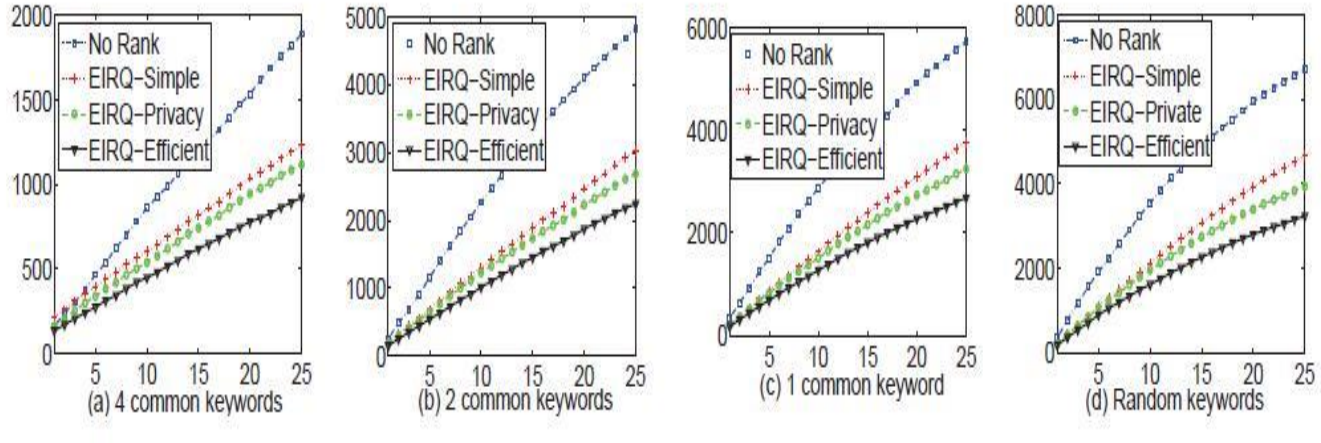

Fig.8. Comparison of communication cost under the Bloom filter setting the x-axis denotes the number of queries in each rank and the y-axis denotes the buffer size (KB). (a) 4 common keywords; (b) 2 common keywords; (c) 1 common

keyword; (d) random keywords.

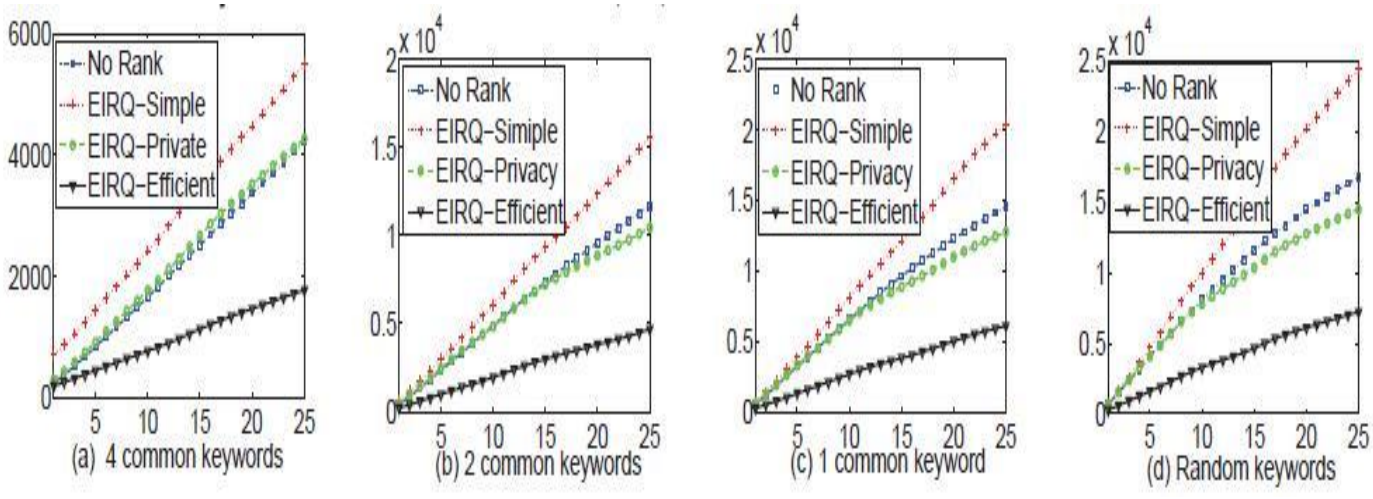

Fig.9. Comparison of communication cost under the Ostrovsky setting the x-axis denotes the number of queries in each rank and the y-axis denotes the buffer size (KB). (a) 4 common keywords; (b) 2 common keywords; (c) 1 common keyword; (d) random keywords.

From Figs. 8 and 9, we know that the EIRQ schemes perform better under the Bloom filter setting compared to under the Ostrovsky setting. Under the Bloom filter setting, all of the EIRQ schemes consume less communication costs than No Rank, e.g., EIRQ-Efficient, EIRQ Privacy, and EIRQ-Simple can further reduce communication costs by about 50%, 35%, and 30% compared to No Rank, respectively, when the queries share 4 common keywords. Under the Ostrovsky setting, EIRQ-Simple always consumes more bandwidth than No Rank, and EIRQ-Privacy only performs better than No Rank under certain conditions. In both settings, the EIRQ schemes consume less bandwidth as the common interests among users increase. For example, when there are 25 users in each rank under the Bloom filter setting, EIRQ-Efficient only generates a 1MB buffer under 4 common keywords, but 3MB under 1 common keyword.

Notice that in both settings, EIRQ-Efficient always has the best performance, the next is EIRQ-Privacy, and the last is EIRQ-Simple. Furthermore, EIRQ-Efficient works better than No Rank when only a few users are conducting searches. For example, when there are 5 queries with 4 common keywords, EIRQ-Efficient generates a buffer of size 274KB, but No Rank generates a buffer of size 467KB, under the Bloom filter setting; EIRQ Efficient generates a buffer of size 439KB, but No Rank generates a buffer of size 834KB under the Ostrovsky setting. When there are 5 queries in each rank with 1 common keyword, EIRQ-Efficient generates a buffer of size 687KB, but No Rank generates a buffer of size 1513KB, under the Bloom filter setting; EIRQ-Efficient generates a buffer of size 1309KB, but No Rank generates a buffer of size 3194KB, under the Ostrovsky setting. However, under the Ostrovsky parameter setting, the mapping times depend on the number of matched files, which in turn depends on the common interests among queries. The comparisons of transfer-in time are shown in Fig. 10.

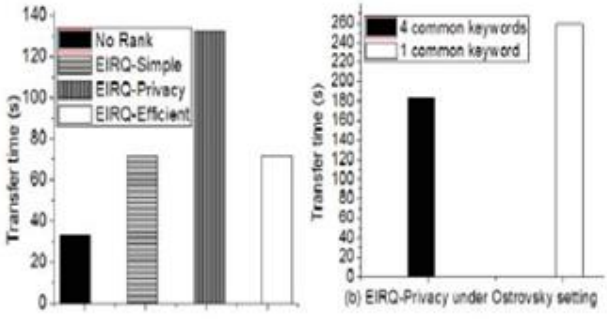

Fig.10. Comparison of transfer-in time. (a) Comparison under Bloom filter setting; (b) EIRQ-privacy under Ostrovsky setting.

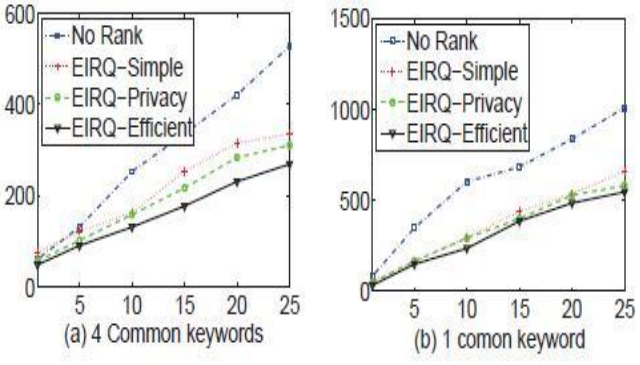

Fig.11. Transfer-out time in real cloud under the Bloom filter setting. The x-axis denotes the number of users, and the y-axis denotes transferring time (s). (a) 4 common keywords; (b) 1 common keyword.

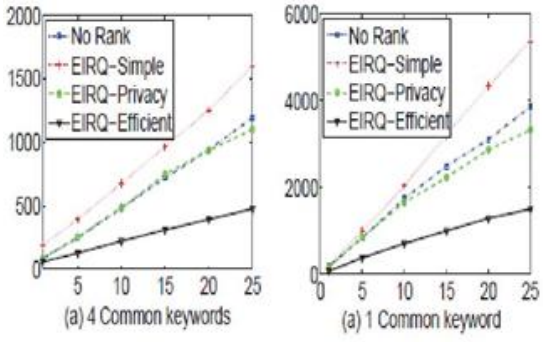

Fig.12. Transfer-out time in real cloud under the Ostrovsky setting. The x-axis denotes the number of users, and the y-axis denotes the transferring time (s). (a) 4 common keywords; (b) 1 common keyword.

D. 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 are 42.98 MB/s. First, we test the transfer-in time in the real cloud, which is mainly incurred by receiving queries from the ADL. Under both parameter settings, the query size for No Rank, EIRQ-Simple, EIRQ-Privacy, and EIRQ Efficient can be calculated with O(d), O(r · d), O(max γ_i · d), and O(r · d), respectively. Given d = 100, r = 4, and |w| = 1KB, the query size for No Rank, EIRQ-Simple, and EIRQ-Efficient is about 100KB, 400KB, and 400KB, respectively. For EIRQ-Privacy, the mapping times are calculated in different ways under different parameter settings. Under the Bloom filter parameter setting, the mapping times are 7, 4, 1, 1, respectively, and thus the query size is about 700KB.

Then, we test the transfer-out time at the cloud, which is mainly incurred by returning files to the ADL. The results are shown in Figs.11 and 12. In all cases, EIRQ Efficient consumes the least amount of transfer time, and EIRQ-Simple works better than No-Rank under the Bloom filter setting. For example, under the Ostrovsky scheme, No-Rank consumes from 83.6s to 1191.8s, EIRQ simple consumes from 189.8s to 1597.6s, EIRQ-Privacy consumes from 83.3s to 1099.9s, and EIRQ-Efficient consumes from 57.4s to 475.1s when there are 4 common keywords; No-Rank consumes from 191.1s to 3857.5s, EIRQ-simple consumes from 181.5s to 5369.7s, EIRQ Privacy consumes from 161.8s to 3323.4s, and EIRQ Efficient consumes from 81.3s to 1502.7s when there is 1 common keyword. Therefore, EIRQ-Efficient is most suitable to be deployed to a cloud environment. For example, the time to transfer a query from the ADL to the cloud consumes less than 100 seconds, and the time to transfer the buffer from the cloud to the ADL consumes less than 500 seconds, under 4 common keywords.

V. CONCLUSION

We propose three EIRQ schemes (EIRQ Simple, EIRQ Privacy, and EIRQ Efficient) are worked through ADL. It offers differential query services, which will also protect the user privacy. These schemes are provide, clients are recovered certain percentage of matched records by particular queries of various ranks. Private searching technique is used to cost efficient cloud environments. In our EIRQ scheme assign ranks for each query, then highest rank files are matched and user recovered certain percentage of matched files. However, in the EIRQ schemes, we simply determine the rank of each file by the

highest rank of queries it matches. For our future work, we will try to design a flexible ranking mechanism for the EIRQ schemes.

REFERENCES:

[1] P. Mell and T. Grance, „„The NIST Definition of Cloud Computing (Draft),"" in NIST Special Publication. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2011.

[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, „„Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions,"" in Proc. ACM CCS, 2006, pp. 79-88.

[3] R. Ostrovsky and W. Skeith, „„Private Searching on Streaming Data,"" in Proc. CRYPTO, 2005, pp. 233-240.

[4] R. Ostrovsky and W. Skeith, „„Private Searching on Streaming Data,"" J. Cryptol., vol. 20, no. 4, pp. 397-430, Oct. 2007.

[5] J. Bethencourt, D. Song, and B. Waters, „„New Constructions and Practical Applications for Private Stream Searching,"" in Proc. IEEE SP, 2006, pp. 1-6.

[6] J. Bethencourt, D. Song, and B. Waters, „„New Techniques for Private Stream Searching,"" ACM Trans. Inf. Syst. Security, vol. 12, no. 3, p. 16, Jan. 2009.

[7] Q. Liu, C. Tan, J. Wu, and G. Wang, „„Cooperative Private Searching in Clouds,"" J. Parallel Distrib. Comput., vol. 72, no. 8, pp. 1019-1031, Aug. 2012.

[8] G. Danezis and C. Diaz, „„Improving the Decoding Efficiency of Private Search,"" Int"l Assoc. Cryptol. Res., IACR Eprint Archive No. 024, Schloss Dagstuhl, Germany, 2006.

[9] G. Danezis and C. Diaz, „„Space-Efficient Private Search with Applications to Rate less Codes,"" in Proc. Financial Cryptogr. Data Security, 2007, pp. 148-162.

[10] M. Finiasz and K. Ramchandran, „„Private Stream Search at the Same Communication Cost as a Regular Search: Role of LDPC Codes,"" in Proc. IEEE ISIT, 2012, pp. 2556-2560.

[11] X. Yi and E. Bertino, „„Private Searching for Single and Conjunctive Keywords on Streaming Data,"" in Proc. ACM Workshop Privacy Electron. Soc., 2011, pp. 153-158.