International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 2 | ISSN : 2456-3307

# Efficient Multicast Delivery for Data Redundancy Minimization over Wireless Data Centres

**A. Sudalaimani[1], D. Stalin David[2]**

[1]PG Scholar, Department of M. Sc(Software Engineering), PSN College of Engineering & Technology, Tirunelveli, Tamilnadu, India
[2]Research Supervisor, Department of M. Sc(Software Engineering), PSN College of Engineering & Technology, Tirunelveli, Tamilnadu, India

## ABSTRACT

With the explosive growth of cloud-based services, large-scale data centers are widely built for housing critical computing resources to gain significant economic benefits. In data centers, the cloud services are generally accomplished by multicast based group communications. Recently, many well-known industries, such as Microsoft, Google and IBM, adopt high speed wireless technologies to augment network capacity in data centers. However, those well-known multicast delivery schemes for traditional wired data centers. We prove the problem are NP-hard and propose efficient heuristic algorithms for the two problems. Based on real traces and practical settings obtained from commercial data centers, a series of experiment conducted and the experimental results show that our proposed algorithm are effective for reducing multicast data traffic. The results also provide useful insights into the design of multicast tree construction and maintenance for wireless data canter networks. Cloud data owners prefer to outsource documents in an encrypted form for the purpose of privacy preserving. Therefore it is essential to develop efficient and reliable cipher text search techniques. In this paper, a hierarchical clustering method is proposed to support more semantistics and also meet the command for fast cipher text search with in a big data environment. The proposed hierarchical approach clusters the documents based on the minimum relevance threshold. The results show that with a sharp increase of documents in the data set. The search time of the proposed method increases exponentially. Furthermore, the proposed method has advantage over the traditional method in the rank privacy and relevance of retrieved documents. However, Bandwidth constraints may restrict the number of reference views sent to clients, limiting the quality of the synthesized viewpoints. In this work, we study the problem of in-network reference view synthesis aimed at improving the navigation quality at the clients. We consider a distributed cloud network architecture, where data stored in a main cloud is delivered to end users with the help of cloudlets, i. e ., resource-rich proxies close to the users. We argue that, in case of limited bandwidth from the cloudlet to the users, re-sampling at the could let the viewpoints of the 3D scene (i. e ., synthesizing novel virtual views in the cloudlets to be used as new references to the decoder) is beneficial compared to mere sub sampling of the original set of camera views. We therefore cast a new reference view selection problem that seeks the subset of views minimizing the distortion over a view navigation window defined by the user under bandwidth constraints.

**Keywords :** PrefDB, data, SQL, Query parser, tuples.

## I. INTRODUCTION

PrefDB, a preference-aware relational system that transparently and efficiently handles queries with preferences. In its core, PrefDB employs a preference-aware data model and algebra, where preferences are treated as first-class citizens. We define a reference using a condition on the tuples affected, a scoring function that scores these tuples, and a confidence that shows how confident these scores are. In our data model, tuples carry scores with confidences. Our algebra comprises the standard relational operators extended to handle scores and confidences. For

example, the join operator will join two tuples and compute a new score-confidence pair by combining the scores and confidences that come with the two tuples. In addition, our algebra contains a new operator, prefer, that evaluates a preference on a relation, i. e ., given as inputs a relation and a preference on this relation, prefer outputs the relation with new scores and confidences. During preference evaluation, both the conditional and the scoring part of a preference are used. The conditional part acts as 'soft' constraint that determines which tuples are scored without disqualifying any tuples from the query result. In this way, PrefDB separates preference evaluation from tuple filtering. This separation is a distinguishing feature of our work with respect to previous works. It allows us to define the algebraic properties of the prefer operator and build generic query optimization and processing strategies that are applicable regardless of the type of reference specified in a query or the expected type of answer. Several approaches to integrating preferences into database queries have been proposed and can be roughly divided into two categories. Plug-in approaches operate on top of the database engine and they typically translate preferences into conventional query constructs. On the other hand, native approaches focus on supporting more efficiently specific queries, such as top-k or skyline queries, by injecting new operators inside the database engine. Unfortunately, both approaches have several limitations. In plug-in methods, the way preferences will be used, for example as additional query constraints or as ranking constructs, the query execution flow as well as the expected type of answer (e. g ., top-k or skyline) are all hard-wired in the method, which hinders application development and maintenance. On the other hand, native methods consider preference evaluation and filtering as one operation. Due to this tight coupling, these methods are also tailored to one type of query. Furthermore, they require modifications of the database core, which may not be feasible or practical in real life. Overall, both native and plug-in approaches do not offer a holistic solution to flexible processing of queries with preferences.

## II. THE PROPOSED SYSTEM

PrefDB is a prototype system that is based on the preference and extended relational data and query models that we presented earlier. Section 2 provides an overview of its functionality and architecture and also describes the implementation of p-relations and the operators. Query processing in PrefDB Figure 2 depicts the system's architecture. Modules depicted in yellow are provided by the native DBMS, whereas the blue-colored ones are those developed for PrefDB. As shown, PrefDB offers two alternative query options: preferences can be provided along with the input query or the system can enrich a non-preferential query with related preferences. In the first query option, preferences are specified in a declarative way, additionally to the standard SQL query part. In the second case, relevant preferences are provided by the profile manager module, which accesses user preferences stored in the database. Stored preferences can be collected from user ratings or by analyzing past queries or clickthrough data [7]. Since preference collection is orthogonal to query processing, which is the primary goal of PrefDB, in our implementation, we simply store preferences specified by users through a visual tool we have developed [7] as well as preferences specified in past Query Parser Query + Preferences Query Optimizer Extended Query Plan SQL Execution Engine Database Engine Scoring, aggregate functions Data Operators σ, π, λ, Optimized Query Plan Profile manager Query + Preferences user queries. For both query options, the query and the preferences are given as input to the query parser. Apart from the core PrefDB query processing strategies that blend preference evaluation into query processing, we have also implemented a set of plug-in methods, which are described in the Appendix. Below is an overview of the core PrefDB modules

**Figure 1.** System Architecture

- ✓ The profile manager selects from the database preferences that can be combined with the conditions of the issued query. For this purpose, we use the preference selection algorithm proposed in [20]
- ✓ The query parser takes as input the query and preferences and generates an extended query plan that is passed to the PrefDB query optimizer.

- ✓ The query optimizer improves the input plan by applying a set of algebraic rules. This improved plan and a cost model for preference evaluation are used for generating alternative plans that interleave preference evaluation and query processing in different ways and for picking the plan with the cheapest estimated cost.
- ✓ The execution engine realizes the execution of the query plan selected by the query optimizer using one of our execution methods. We discuss

## III. RELATED WORK

The concept of preference-aware query processing appears in many applications, where there is a matter of choice among alternatives, including query personalization [10], [18], [20], recommendations [4] and multi-criteria decision making [9], [13]. We discuss prior work with respect to how preferences are represented in the context of relational data and how they are integrated and processed in queries. In representing preferences, there are two approaches. In the qualitative approach, preferences are specified using binary predicates called preference relations [5], [10], [18]. In quantitative approaches, preferences are expressed as scores assigned to tuples [6], [23] be specified based on any combination of scores, confidences and context. Our framework allows us to process in a uniform way all these different query and preference types. In terms of preference integration and processing, one approach is to translate preferences into conventional queries and execute them over the DBMS [14], [19], [20], [21], [24]. Several efficient algorithms have been proposed for processing different types of queries, including top-k queries [13] and skylines [9]. These algorithms as well as query translation methods are typically implemented outside the DBMS. Thus, they can only apply coarse grained query optimizations, such as reducing the number of queries sent to the DBMS. Further, as we will also demonstrate experimentally plug-in methods do not scale well when faced with multi-join queries or queries involving many preferences. Native implementations modify the database engine by adding specific physical operators and algorithms. RankSQL [23] extends the relational algebra with a new operator called rank that enables pipelining and hence optimizing top-k queries. Another example of operator is the winnow operator [10], which selects all tuples corresponding to the Pareto optimal set. Our approach is different from existing works in several ways. First, existing techniques are limited to a particular type of query. In contrast to these approaches, we consider preference evaluation (how preferences are evaluated on data) and selection of the preferred tuples that will comprise the query answer as two operations. We focus on preference evaluation as a single operator that can be combined with other operators and we use its algebraic properties in order to develop generic query optimization and processing techniques. Finally, we follow a hybrid implementation that is closer to the database than plug-in approaches yet not purely native, thus combining the pros of both worlds. A different approach to flexible processing of queries with preferences is enabled in FlexPref [22]. FlexPref allows integrating different preference algorithms into the database with minimal changes in the database engine by simply defining rules that determine the most preferred tuples. Once these rules are specified a new operator can be used inside queries. It is worth noting that both FlexPref and our work are motivated by the limitations of plug-in and native approaches. FlexPref approaches the problem from an extensibility viewpoint. Our focus is on the problem of preference evaluation as an operator that is separate from the selection of preferred answers, and we study how this operator can be integrated into query processing in an effective yet not obtrusive to the database engine way.

## IV. PROPOSED METHODOLOGY

In this paper, we first construct an extended query plan that contains all operators that comprise a query and we optimize it. Then, for processing the optimized query plan, our general strategy is to blend query execution with preference evaluation and leverage the native query engine to process parts of the query that do not involve a prefer operator. Given a query with preferences, the goal of query optimization is to minimize the cost related with preference evaluation. Based on the algebraic properties of prefer, we apply a set of heuristic rules aiming to minimize the number of tuples that are given as input to the prefer operators. We further provide a cost-based query optimization approach. Using the output plan of the first step as a skeleton and a cost model for preference evaluation, the query optimizer calculates the costs of alternative plans that interleave preference evaluation and query processing in different ways. Two plan enumeration methods, i. e ., a dynamic programming and a greedy algorithm are proposed. For executing an optimized query plan with preferences, we describe an improved

version of our processing algorithm (GBU) (an earlier version is described in. The improved algorithm uses the native query engine in a more efficient way by better grouping operators together and by reducing the out-of-the-engine query processing.

Modules:

Registration & Interest Sum up
Query Formation
Query Optimization & Execution

A preferential query combines p-relations, extended relational and prefer operators and returns a set of tuples that satisfy the boolean query conditions along with their score and confidence values that have been calculated after evaluating all prefer operators on the corresponding relations. Intuitively, the better a tuple matches preferences and the more (or more confident) preferences it satisfies, the higher its final score and confidence will be, respectively. The query parser adds a prefer operator for each preference. Finally, the query parser checks for each preference, whether it involves an attribute (either in the conditional or the scoring part) that does not appear in the query and modifies project operators, such that these attributes will be projected as well. proportional to the number of tuples flowing through the operators in the query plan. Assuming a fixed position for the other operators, the goal of our query optimizer is essentially to place the prefer operators inside the plan, such that the number of tuples flowing through the score tables is minimized. The execution engine of PrefDB is responsible for processing a preferential query and supports various algorithms. Another example of operator is the winnow operator [10], which selects all tuples corresponding to the Pareto optimal set. Our approach is different from existing works in several ways. First, existing techniques are limited to a particular type of query. In contrast to these approaches, we consider preference evaluation (how preferences are evaluated on data) and selection of the preferred tuples that will comprise the query answer as two operations. We focus on preference evaluation as a single operator that can be combined with other operators and we use its algebraic properties in order to develop generic query optimization and processing techniques. Finally, we follow a hybrid implementation that is closer to the database than plug-in approaches yet not purely native, thus combining the pros of both worlds. A different approach to flexible processing of queries with preferences is enabled in FlexPref [22]. FlexPref allows integrating different preference algorithms into the database with minimal changes in the database engine by simply defining rules that determine the most preferred tuples. essentially to place the prefer operators inside the plan, such that the number of tuples flowing through the score tables is minimized. The execution engine of PrefDB is responsible for processing a preferential query and supports various algorithms. Another example of operator is the winnow operator [10], which selects all tuples corresponding to the Pareto optimal set. Our approach is different from existing works in several ways. First, existing techniques are limited to a particular type of query. In contrast to these approaches, we consider preference evaluation (how preferences are evaluated on data) and selection of the preferred tuples that will comprise the query answer as two operations. We focus on preference evaluation as a single operator that can be combined with other operators and we use Once these rules are specified a new operator can be used inside queries. It is worth noting that both FlexPref and our work are motivated by the limitations of plug-in and native approaches. FlexPref approaches the problem from an extensibility viewpoint.

## V. EXPERIMENTAL RESULTS

The proposed system is implemented on an Intel core i5 processor system running at 2. 20 GHz, 3GB RAM using Java and Ulteo OVD virtual desktop for building cloud environment. The implemented system consists of 5 modules: User registration, encrypt and upload, file sharing, decrypt and download and verification auditing.

**1. UserRegistration**:

The registration function allows users to create secure account. Here the user enters his/her information necessary for signing up like user's name, password, mobile no and email-address. The validations and required fields are effectively handled. Each user will be provided his/her own space on cloud.

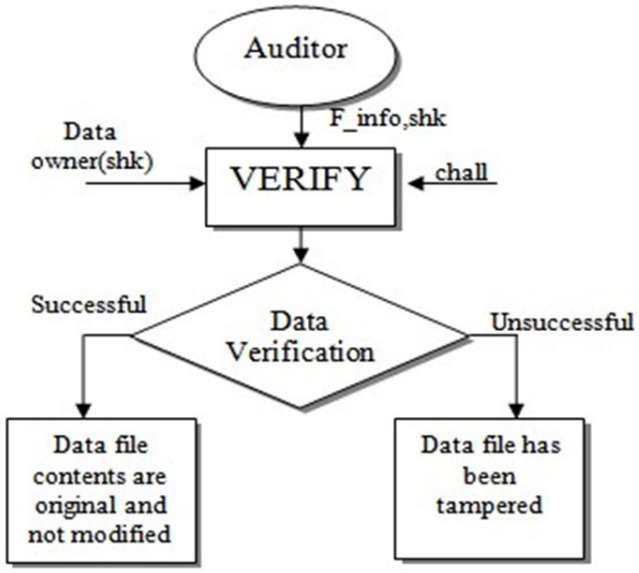


**Figure 2.** Verification Inspection

**2. Encrypt and Upload**

After registering, the user may login into the system. Every user is provided space on cloud where they may upload their files. The encrypt function will encrypt the data files of users before storing them on cloud storage using ElGamal cryptosystem. The owner will generate secrete hash key using SHA-256 and secrete key to download the uploaded file. The secret hash key is further mailed to TPA for data integrity verification. The time required to encrypt the files using ElGamal is also recorded.

**3. File Sharing**

The data owner may share the outsourced files with other users in cloud using the share module. The secret key generated during encryption is also mailed to the shared user in order to grant them access to the shared file. The shared user may download the file, make changes and again upload the file. In such a case, TPA informs the original data owner of that file about the latest modifications done by a shared user. Every user is provided space on cloud where they may upload their files. The encrypt function will encrypt the data files of users before storing them on cloud storage using ElGamal cryptosystem. The owner will generate secrete hash key using SHA-256 and secrete key to download the uploaded file. The secret hash key is further mailed to TPA for data integrity verification. The time required to encrypt the files using ElGamal is also recorded.

**4. Decrypt and Download:**

The data owner or a shared user may need to download the file. Since the data files stored on cloud server are in encrypted form, decryption must be performed before downloading the file. Initially, the system validates whether the user requesting to download the file is a Legitimate user by demanding the secret key from that user. The decryption module then performs data decryption using both RSA and ElGamal decryption scheme and downloads the data using secret key sent by the data owner. The time required to decrypt the file is also recorded. The data owner may share the outsourced files with other users in cloud using the share module. The secret key generated during encryption is also mailed to the shared user in order to grant them access to the shared file. The shared user may download the file, make changes and again upload the file. In such a case, TPA informs the original data owner of that file about the latest modifications done by a shared user. Every user is provided space on cloud where they may upload their files. The encrypt function will encrypt the data files of users before storing them on cloud storage using ElGamal cryptosystem. The owner will generate secrete hash key using SHA-256 and secrete key to download the uploaded file.

**5. Verification Auditing**

In order to authenticate the integrity of the user's uploaded data, the TPA is granted access to the system. The TPA validates the integrity of the cloud data files on remote server on behalf of cloud user itself. TPA verifies the legitimacy of data using secret hash key sent by the cloud user. If the secret hash key matches with hash key in the cloud server, the verification proves to be successful, thus implying that the data files has not been modified. However, if the verification is unsuccessful, an email is dispatched to the data owner of the file informing about the last modifications done to his file.

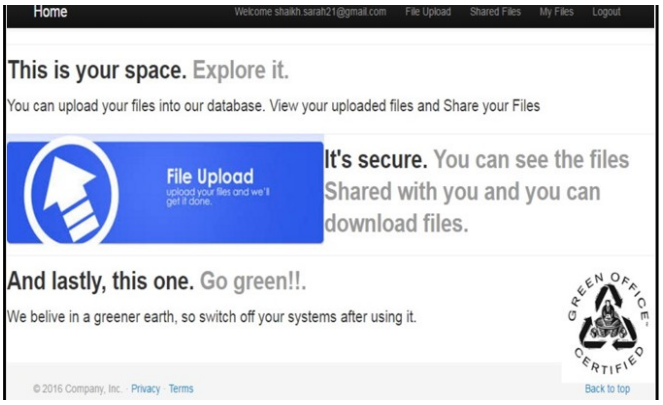


**Figure 3.** Cloud Users' Home Page

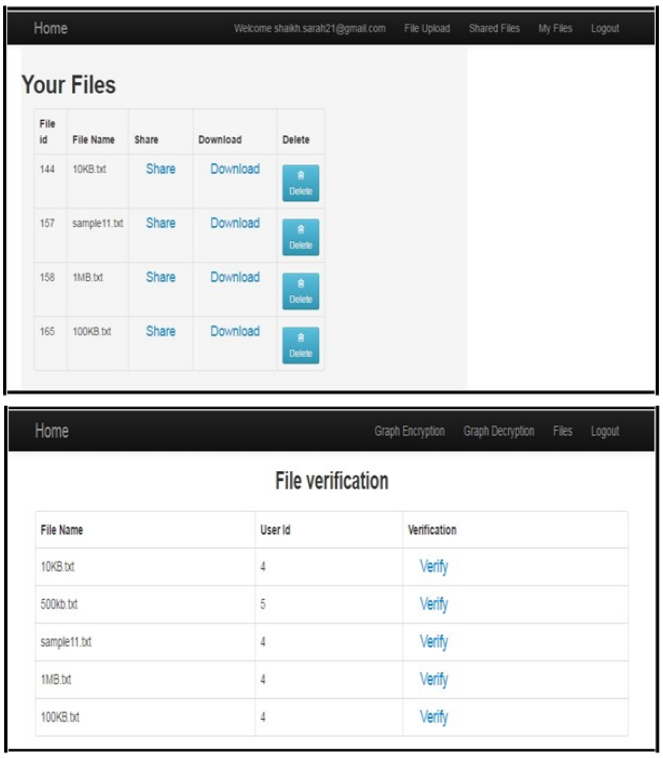


**Figure 4.** File Sharing and Downloading

## VI.CONCLUSION

To authenticate the integrity of data uploaded on cloud server, it is significant to permit a third party auditor to assess the quality of data content outsourced on cloud server. Public auditing system permits the clients to allocate the data integrity authentication tasks to a third party as they themselves can be unstable or may not possess essential computational resources to perform periodic integrity verifications. However, delegating data integrity verification task to a third party (TPA) raises privacy issues since the TPA may derive the actual data content from server during verification process.

Thus the proposed system uses public auditing scheme for data storage security on cloud while protecting the confidentiality of the user's data. ElGamal encryption along with SHA-256 hash algorithm are used to make sure that the TPA should not get access to the outsourced data on the cloud server while performing integrity check thereby increasing the effectiveness of the auditing process. This eliminates the overhead of performing auditing task from the client and also lessens the cloud users' concern that their uploaded data may be accessed by an untrusted organization or individual.

**TABLE I:** Comparative analysis of encryption time.

| File Size(KB) | RSA (ms) | ElGamal (ms) |
|---|---|---|
| 1 | 1076 | 261 |
| 2 | 277 | 16 |
| 10 | 707 | 89 |
| 100 | 1353 | 1319 |
| 500 | 4243 | 33212 |
| 1000 | 8277 | 326752 |
| Average | 2655 | 60235 |

**TABLE 2:** Comparative analysis of decryption time

| File Size(KB) | RSA (ms) | ElGamal (ms) |
|---|---|---|
| 1 | 733 | 310 |
| 2 | 706 | 250 |
| 10 | 3019 | 1010 |
| 100 | 26586 | 13463 |
| 500 | 186825 | 81228 |
| 1000 | 218162 | 130312 |
| Average | 72671 | 37724 |

## VII. REFERENCES

[1]. Giuseppe Ateniese, "Provable Data Possession at Untrusted Stores", Proc. of ACM Conference on Computer and Comm. Security (CCS), 2007.

[2]. Ari Juels and Burton S. Kaliski Jr, "Pors: Proofs of Retrievability for Large Files", Proc. of ACM Conference on Computer and Comm. Security (CCS), pp. 584-597, 2007.

[3]. Hovav Shacham and Brent Waters, "Compact Proofs of Retrievability, " International Conf. on Theory and Application of Cryptology and Information Security: Advances in Cryptology, pp. 90-107, 2008.

[4]. Giuseppe Ateniese, "Scalable and Efficient Provable Data Possession", International Conf. on Security and Privacy in Comm. Networks (SecureComm), 2008.
[5]. C. Chris Erway, Alptekin Kupcu, Charalampos Papamanthou, Roberto Tamassia, "Dynamic Provable Data Possession", ACM International Conf. on Computer and Comm. Security (CCS), 2009.
[6]. Qian Wang, Cong Wang, Kui Ren, Wenjing Lou and Jin Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", IEEE Transaction on Parallel and Distributed System, vol. 22, no. 5, pp. 847 859, 2011.
[7]. Cong Wang, S. M. Chow, Qian Wang, Kui Ren and Wenjing Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage", IEEE Transaction on Computers, Vol. 62, no. 2, 2013.
[8]. Sarah Shaikh, Deepali Vora, "Review of Privacy Preserving Auditing Techniques", International Journal of Computer Applications (0975- 887), Volume 145 No. 13, July 2016.
[9]. T S Khatri, G B Jethava "Improving Dynamic Data Integrity Verification in Cloud Computing", 4th IEEE ICCCNT 2013.
[10]. S. V. Baghel, D. P. Theng " A Survey for Secure Communication of Cloud Third Party Authenticator ", 2nd International Conference on Electronics and Communication Systems, IEEE ICECS '2015.
[11]. ElGamal Cryptosystem, http://lxmayr1. informatik. tu- muenchen. de/konferenzen/Jass05/courses/1/papers/meier_paper. pdf