AN EFFICIENT DISTRIBUTED SYSTEM FOR PRIVATE INFORMATION RETRIEVAL IN PCLOUD

[1]Durga.R and [2]Nithya.L.M
[1] III-MTECH, Information And Technology (part-time),
SNS College of Technology, Anna University,
Coimbatore - 35, Tamil Nadu
[2]Professor and Head, Department of Information Technology
SNS College of Technology, Anna University, Coimbatore -35, Tamil Nadu
Email :- sdurgasns@gmail.com, lmnithya@gmail.com

Abstract:- The peer-to-peer computing has attracted much attention as a new distributed computing paradigm. pCloud organizes the peers in overlay network, partitions the database into disjoint data segments, and disseminates the individual segments to the peers. To distribute the database to a number of cooperative peers, and leverage their computational resources to process cPIR queries in parallel by using the stripping technique. pCloud reduces the query response time compared to the traditional client/server model, and has a very low communication overhead Additionally it scales well with an increasing number of peers, achieving a linear speedup

Keywords: Private Information Retrieval (PIR), Update with continuous Timestamp (UCT), pCloud,P2P.

I. INTRODUCTION

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Private cloud **(pCloud)** infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Private information retrieval (PIR) [9] is the task of fetching an item from a database server There are two main types of PIR: information-theoretic and computational. In information-theoretic PIR, the server is unable to determine any information about your query even with unbounded computing power. In computational PIR (CPIR), the privacy of the query be guaranteed

Peer-to-Peer (P2P) network overlays[4] because they provide a good substrate for creating large-scale data sharing, content distribution, and application-level multicast applications. These P2P overlay networks attempt to provide a long list of features, such as: selection of nearby peers, redundant storage, efficient search/location of data items, data permanence or guarantees, hierarchical naming, trust and authentication, and anonymity. P2P networks potentially offer an efficient routing architecture that is self-organizing, massively scalable, and robust in the wide-area, combining fault tolerance, load balancing, and explicit notion of locality.

II.EXISTING SYSTEM

There is an n-bit Database server. A PIR [10] allows a client to retrieve a bit, from the server. PIR can also retrieve blocks of data (e.g., an -bit record) by viewing the database as n elements, each with size bits. The peers hold the disjoint partitions of

the database DB and ready to answer queries upon request. The database server holds the most current view of the entire DB. The inclusion of the server is necessary for the following reasons: (i) the PIR query need to process every bit of the database. Suppose that the client does not receive replies for all the different database partitions. This suggests that at least one partition has not been processed, because either it did not exist in the network, or the node that accommodated it failed. (ii) The server is the only entity that receives the data updates and, therefore, holds the most up-to-date DB instance at all times. Note that, although a centralized entity exists, query processing inside the P2P network is completely independent (e.g., the server does not have a guide for query propagation)

The challenge lies in how to partition and distribute DB to the peers, so that query processing experiences a linear performance speed-up with respect to the number of partitions. Towards this end, pCloud exploits the features of the striping technique that is simply subdivide DB into k partitions, and disseminates them among the peers; the query execution cost will be k times lower, compared to the traditional client/server model [7]. The above statement is true, provided that every partition is accommodated by at least one peer, and is processed during the query execution [3]. The query processing in pCloud involves two distinct phases: query and result propagation.

*Query flooding***:**

The client needs to retrieve every unique partition that resides inside the network, so flooding protocol [1] is used. The server distributes the database among the cooperative peers. The database is distributed randomly so every peer will not get all databases. If any peer requires particular information which it doesn't have then, it may arise a query. That query is passed to all neighboring peers.

Result propagation

The query is passed to all the neighboring peers in the pCloud. A path is followed during query generation [2]. The peer which has the required information replies and sends the information through the path.

If same query is arise more than threshold time, then data replication will take place and the peer which has the information will replicate it to all the peers in that path. In the peer which had the required information sends it through the path then the peer who queried for that particular information will extract it from the nearby peers

- **DISADVANTAGE:**

When the server receives a number of updates, it modifies the corresponding pages, and produces a new set of signatures that incorporate the current timestamp. Subsequently, it broadcasts (using the flooding mechanism of query propagation) a list with the outdated (i.e., affected) partitions to all the peers. If a peer receiving the server's message is the owner of an obsolete partition, it 1) suspends query processing (i.e., it does not return any results), and 2) contacts the server to receive the updates. To save communication resources, the server does not transmit the entire partition, but only sends the blocks that are affected by the updates. If updates are very frequent the client has to request many partitions from the server, which greatly increases the query response time

III.PROPOSED SYSTEM

Continuous Timestamp based Replication Management (CTRM), which deals with the efficient storage, retrieval and updating of replicas. To perform updates on replicas, a new protocol is proposed that stamps the updates with timestamps which are generated in a distributed fashion using groups of peer. The updates' timestamps are not only monotonically increasing but also continuous, i.e. without gap. The property of monotonically increasing allows CTRM to determine a total order on updates and to deal with concurrent updates

After each update on a data, the corresponding patch is sent to the group where a monotonically increasing timestamp is generated by one of the members, i.e. the responsible of the group. Then the patch and its timestamp are published to the members of the group using an update protocol, called UCT protocol [5]. To retrieve an up-to-date replica of a data, the request is sent to the responsible of the data's replica holder group. The responsible peer sends the data and the latest generated timestamp to the group members, one by one, and the first member that has received all patches returns its replica to the requester. To verify whether all patches are received, replica holders check the two following conditions, called *up-to-date conditions*: 1) the timestamps of the received patches are continuous; 2) the latest generated timestamp is equal to the timestamp of the latest patch received by the replica holder. The above up-to-date conditions are also

verified periodically by each member of the group. If the conditions do not hold, the member updates its replica by retrieving the missed patches and their corresponding timestamps from the responsible of the group or other members that hold them. The UCT protocol proceeds as follows:

-*Update request:*

In this phase, the update requester p_0, obtains the address of the responsible of the replica holder group p_1, using the lookup service, and sends to it an update request Then, p_0 waits for a commit message from p_1. It also uses a failure detector and monitors p_1. The wait time is limited by a default value, *e.g.* by using a timer. If p_0 receives the terminate message from p_1, then it commits the operation. If the timer timeouts or the failure detector reports a fault of p_1, then p_0 checks whether the update has been done or not, *i.e.* by checking the data at replica holders. If the answer is positive, then the operation is committed, else it is aborted.

*Timestamp generation and replica publication***:**

After receiving the update request, p_1 generates a timestamp for *k*, Then, it sends *(k)* to the replica holders, *i.e.* the members of its group, and asks them to return an acknowledgement. When a replica holder receives, it returns the acknowledgement to p_1 and maintains the data in a temporary memory on disk

Update confirmation:

In this phase, p_1 sends the commit message to the replica holders. When a replica holder receives the commit message. In the case of concurrent updates, e.g. two or more peers want to update a data *d* at the same time. In this case, the concurrent peers send their request to the responsible of the group, say p_1. The peer p_1 determines an order for the requests, e.g. depending on their arrival time or on the distance of requesters if the requests arrive at the same time. Then it processes the requests one by one according their order, i.e. it commits or aborts one request and starts the next one.

ADVANTAGES:

1. Concurrent updates make no problem of inconsistency for replication management
2. It avoids the node failures, thereby increase the query response time.

IV. RELATED WORK

Most existing P2P systems support data replication, but usually they do not deal with concurrent and missed updates.

OceanStore [14] is a data management system designed to provide a highly available storage utility on top of P2P systems. It allows concurrent updates on replicated data, and relies on reconciliation to assure data consistency. The reconciliation is done by a set of powerful servers using a consensus algorithm. The servers agree on which operations to apply, and in what order. However, in the applications, which we address, the presence of powerful servers is not guaranteed.

The BRICKS project [15] provides high data availability in DHTs through replication. For replicating a data, BRICKS stores the data in the DHT using multiple keys, which are correlated to the data key, e.g. *k*. There is a function that given *k*, determines its correlated keys. To be able to retrieve an up-to-date replica, BRICKS uses versioning. Each replica has a version number which is increased after each update. However, because of concurrent updates, it may happen that two different replicas have the same version number, thus making it impossible to decide which one is the latest replica.

In [13], an update management service, called UMS, was proposed to support data currency in DHTs, *i.e.* the ability to return an up-to-date replica. However, UMS does not guarantee continuous time stamping which is a main requirement for collaborative applications which need to reconcile replica updates. UMS uses a set of *m* hash functions and replicates randomly the data at *m* different peers, and this is more expensive than the groups which we use in CTRM, particularly in terms of communication cost

V. RESULTS AND DISCUSSION

When compared to the traditional client/server architecture, pCloud drops the query response time by orders of magnitude, and its performance improves linearly with the number of peers. Finally, it is resilient to node failures and can handle updates.

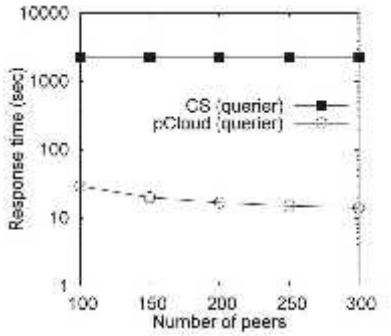

Figure 5.1: Comparison of Pcloud with C/S model for response time

In pCloud, the response time is up to more than two orders of magnitude smaller than in CS. The reason is that in CS the server processes all partitions sequentially, leading to an excessive computational peers perform the reply generation algorithm on a very small portion of the database in parallel.

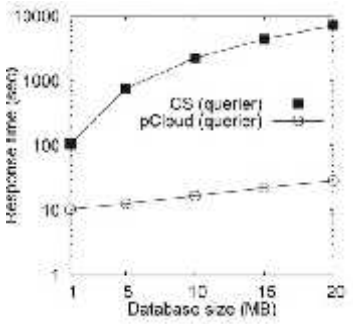

Figure 5.2: Comparison of Pcloud with C/S model on varying the database size

The main database server is kept outside the P2P network. The database is stored at the server. The database is segmented into t blocks. The server distributes the database among the cooperative peers. The database is distributed randomly so every peer will not get all databases.

If any peer requires particular information which it doesn't have then, it may arise a query. That query is passed to all neighboring peers. A path is followed during query generation. The peer which has the required information replies and send the information through the path. When the peer which had the required information sends it through the path then the peer who queried for that particular information will extract it from the nearby peer.

While any updating done on the server it has to be replicated on the peer also. So the request from the peer has to send to the server on the timestamp generation basis and replica publication. If any update done, then it will send to the corresponding peer through the stripping technique.

VI.CONCLUSION AND FUTURE ENHANCEMENT

The data replication among the peers inside the cloud takes place. Due to this replication the searching of information can be achieved in less time. Instead of accessing the same peer for that particular information, the peers nearby to the peer which require information can store the information in it due to replication. In pCloud, the response time is up to more than two orders of magnitude smaller than in CS.

The future enhancement, which deals with the efficient storage, retrieval and updating of replicas. The replica holders cannot able to detect the existence of missed updates they have received. When it was detected then query response time will be more efficient in pcloud.

REFERENCES

1. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: (2001) A scalable content- addressable network. *SIGCOMM Conf.*, 161-172 [1] Vassilios V. Dimakopoulos, Member, IEEE, and Evaggelia Pitoura, Member, IEEE, 2006 "On the Performance of Flooding-Based Resource Discovery", IEEE transactions on parallel and distributed systems, vol. 17, no. 11, November.
2. Dongsheng Li, Jiannong Cao, Senior Member, (2009).IEEE, Xicheng Lu, and Keith C.C. Chan," Efficient Range Query Processing in Peer-to-Peer Systems" IEEE transactions on knowledge and data engineering, vol. 21, no. 1, january
3. Haiying (Helen) Shen, Member, IEEE, (2010),.IRM: Integrated File Replication andConsistency Maintenance in P2P Systems, IEEE transactions on parallel and distributed systems, vol. 21, no. 1, january
4. Yi Hu, *Student Member, IEEE,* Laxmi N. Bhuyan, *Fellow, IEEE*, and Min Feng, 2005 **"**maintaining data consistency in structured

5. p2p system", IEEE transactions on parallel and distributed systems
6. Reza Akbarinia1, Mounir Tlili2, Esther Pacitti3, Patrick Valduriez4, Alexandre 2005 "continuous time stamping for efficient replication management in dhtse", In *ICALP*,PP: 41 - 50.
7. Eng Keong Lua, Jon Crowcroft, And Marcelo Pias, University Of Cambridgeravi Sharma, Nanyang Technological UniversitySteven Lim, Microsoft Asia 2005 "a survey and comparison of peer-to-peer overlay network schemes SECOND QUARTER, VOLUME 7, NO. 2
8. Iliev.A and Smith S.W (2004), "Private information storage with logarithm-space secure hardware", In Proc. International InformationSecurity Workshops, PP: 148 - 157.
9. Papadopoulos.S,Bakiras.S, Papadias.D (2012), 'pCloud A Distributed System for Practical PIR' TDSC,VOLUME:9, ISSUE: 1, PP: 115-127.
10. Goldberg.I (2007), 'Improving the Robustness of Private Information Retrieval',SP, PP: 131-148.
11. Melchor C.A, Crespin.B, Gaborit.P, Jolivet.V, Rousseau.P (2008) , 'High-Speed Private Information Retrieval Computation on GPU' PP: 263-272
12. Melchor C.A, Gaborit.P (2008), 'A fast private information retrieval protocol', ISIT, PP:1848-1852.
13. Papadopoulos.S,Bakiras.S, Papadias.D (2012), 'pCloud A Distributed System for Practical PIR' TDSC,VOLUME:9, ISSUE: 1, PP: 115-127.
14. Akbarinia, R., Pacitti, E., Valduriez, P.: Data Currency in Replicated DHTs.*SIGMOD Conf.*, 211-222 (2007)
15. Rhea, S.C., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B., Kubiatowicz, J.:Pond: the OceanStore Prototype. *USENIX Conf. on File and Storage Technologies*, 1-14 (2003)
16. Knezevic, P., Wombacher, A., Risse, T.: Enabling High Data Availability in a DHT. *Proc. of Int. Workshop on Grid and P2P Computing Impacts on Large Scale Heterogeneous Distributed Database Systems*, 363-367 (2005)