Improved Aggressive Update Propagation Technique in Cloud Data Storage

Mohammed Radi

Computer science department, Faculty of applied science, Alqsa University Gaza

Abstract: Recently, cloud computing became a new global trend of computing, it’s a modern style of using the power of the Internet and wide area network (WAN) to offer resources remotely. Cloud computing is a new solution and strategy to achieve high availability, flexibility, cost reduced and on demand scalability. The consistency of multiple data replicas in cloud data storage is discussed in this paper. An Improved Aggressive Update (IAUP) technique is used to maintain replica consistency, which introduce a high degree of consistency and improves the reliability without affecting performance of the system.

Key Words: Cloud data storage, replication, replica consistency

1. Introduction
Cloud computing is a well-known computing paradigm in business and is receiving a large amount of attention from the research community. Cloud computing is a general term for anything that involves supplying hosted services over the Internet. Examples of services available in the cloud are Infrastructure-as-a-Service(IaaS), Platform-(PaaS) and Software-as-a-Service (SaaS) [1, 2]. Moreover Cloud Computing provides an opportunity to store data in Cloud Storage instead of storing it locally, so users do not need to maintain large storage infrastructures. They can store data in remote data centers, controlled and managed by big companies like Apple, Microsoft, Google, and Amazon etc. Files saved in the cloud storage can be accessed from anywhere with any device with an Internet connection. Cloud Storage is an important part of cloud computing. Cloud storage is an online storage available on network hosted by third party vendors[3]. Nowadays Cloud storage services are widespread as they promise high scalability and availability at low cost.

Cloud Storage, Data as a service (DaaS) and Database as a service (DbaaS) are the different terms used for data management in the Cloud. They differ on the basis of how data is stored and managed. Cloud storage is a virtual storage that enables users to store documents and objects[4].

Data storage in cloud relies on file replication to achieve better performance and reliability by storing copies of data sets on different nodes[5]. Files are distributed across data nodes to achieve availability fault tolerance, scalability and performance. Unfortunately, the more replication increases the problem of inconsistent replicas. Data storage with file replication needs consistency maintenance to keep the consistency between a file and its replicas and on the other hand the overhead of consistency maintenance is determined by the number of replicas.

The consistency of multiple data replicas in cloud data storage is discussed in this paper. An Improved Aggressive Update Propagation (IAUP) technique is used to maintain replica consistency, which introduce a high degree of consistency and improves the reliability without affecting performance of the system. This paper is organized as follows: section 2 reviews related work in cloud computing and data replication. Section 3 presents the approach for lazy update propagation for data replication in cloud computing. Section 4 gives some experimental results for our solution. Section 5 concludes this paper and describes the future work.

2. Related works
In distributed systems, data replication is used to achieve a higher level of performance, reliability and availability. The main design requirements of cloud storage are scalability, availability, security, multi-tenancy, reliability, speed, control, cost, and simplicity. Cloud Storage design requirements can scale up or down depending on business requirements. So, Cloud storage should be scalable to meet requests from unlimited and concurrent users without affecting performance and speed [4]. Cloud Storage services should be available round the clock. Decentralization techniques such as replication are used for fault-tolerance and better availability of cloud services[6]. Data is replicated on different servers residing at different locations to avoid a single point of failure. Multiple nodes provide same services, if primary node fails, backup nodes take over.

Data integrity and consistency is the most critical requirement of all business applications and is maintained through database constraints. Cloud databases follow BASE (Basically Available, Soft state, eventually consistent) in contrast to the ACID (Atomicity, Consistency, Isolation and Durability) guarantees. So, Cloud databases support eventual consistency due to replication of data at multiple distributed locations. It becomes difficult to maintain the consistency of a transaction in a database which changes too quickly especially in the case of transactional data. There are many consistency models proposed in for distributed systems and database systems includes [7] [8].
In data grid environment many replica consistency models have been proposed[9-11]. Our work extends these established models by allowing levels of consistency to be defined and adapting the consistency guarantees.

Yin Nyein Aye in [5] address the problem of consistency on cloud storage by using probability for data consistency on cloud storage system which intends to prove the maximize resource utilization on the cloud environment. The paper proposed an analytical model using MM1 queuing for data consistency on private cloud storage system to evaluate the discarding probability based on update requests to handle update conflict. Moreover, an approach is also proposed to improve the readability after update and consistency of storage on the cloud environment. [12] Discussed the problem of maintaining the consistence of data replication. Lazy update approaches are used to separate the process of data replica updating and data access in cloud. The approach improves the throughput of the information and data service while reducing response time. Initial experiments show the methods are efficient and effective. Islam proposes a tree-based consistency approach that reduces interdependency among replica servers by introducing partially consistent and fully consistent states of cloud databases. The tree is formed such a way that the maximum reliable path is ensured from the primary server to all replica servers [13]. Ximei Wang propose an application-based adaptive mechanism of replica consistency in cloud data storage they divide the consistency of applications into four categories according to their read frequencies and update frequencies, and then design corresponding consistency strategies. The results show that the mechanism decreases the amount of operations while guaranteeing the application’s consistency requirement[14] Kraska proposes a strategy that system can switch the level of consistency between serializability and session consistency dynamically according to running condition in the cloud. [15] It divides the data into three categories, and treats each category differently depending on the consistency level provided. The consistency level will be changed accordingly while the data’s impact changes continuously at runtime. Dynamo [16] only promises eventual consistency. Cassandra [17] provides quorum-based consistency and eventual consistency.

3. Replica consistency in Cloud Computing

The protocol used to maintain consistency among object replicas is an important aspect of a replication-based system (Jim, et al. 1996) including cloud data storage [12]. This is because replica is not just a simple copy of an original file but has logical connection to the original copy. In general, the data consistency problem deals with keeping two or more data items consistent. In order to keep the consistency among replicas, conservative update replication systems (synchronous replication) prevent all concurrent updates and require locks and votes from other replicas before committing any changes to a data item. This synchronous approach guarantees that all replicas are always fully synchronized and fully consistent. However, it results in a slow performance of write operation and consumes network bandwidth to check the consistency on every update. Cloud data storage applications do not require such conservative update replication as the update conflicts are rare. The relatively slow performance and the high consumption of network bandwidth for the write operation in conservative environment make asynchronous replication a more appropriate protocol in cloud data storage. In an asynchronous replication, each update injected at a single site must be propagated to all the other sites. All sites become fully consistent only when all updating activities have stopped and the system has become quiescent. When client sends update job to the master site, the update transaction commits after updating the master replica. After the transaction commits, the updates are propagated towards the other replicas which are then updated in separate refresh transactions. The updates can be propagated to other replica sites in different form depending on the ability of the system. The time for propagating the update information to other replica sites depends in the propagation approach and it can be in a lazy mode or in an aggressive mode.

In Aggressive Update Propagation(AUP) approach, also referred to as Push protocols, updates are propagated to all other replica sites without those replica sites even asking for the updates. Aggressive protocol is applied when replicas generally need to maintain a relatively high degree of consistency. In other words, replicas need to be kept identical. Push-based is needed to support high degree of consistency. Consequently, the read-to-update ratio at each replica is relatively high. In these cases, push-based protocols are efficient such that every pushed update can be expected to be used for one or more readers. In addition, push-based protocols make consistent data immediately available when asked for. In lazy Update propagation (LUP) approach, also referred to as pull-based approach, a server or client requests another server to send it any updates it has at that moment. Pull-based protocols, can save network bandwidth resources without transferring up-to-date replicas every time when some modifications are made. However the main drawback of a pull-based strategy in comparison to a push-based approach is that the response time increases in the case of up-to-date data miss. Sometimes there will also be a need to transfer the whole file if the update information is not available since the master replica site did not keep the old version or the update information regarding to storage cost.

3.1 Proposed Replica Consistency Framework

The proposed framework consists of three components, the master site, secondary sites and a metadata. The master site contains the master replica which is the only
site receives the updates from users. Secondary sites hold a read only replicas and do not receives updates from users. Meta data contains replication information; mainly information about all sites holds a copy of the replica and the kind of replication mode.

![Figure 1: Replica consistency framework](image)

The interaction between the update propagation framework components is shown in Figure 1. The proposed replica consistency framework is based on an Improved Aggressive Update Propagation (IAUP) technique.

### 3.2 Improved Aggressive Update Propagation Technique

The Improved Aggressive Update Propagation (IAUP) technique is an enhancement version of the traditional Aggressive Update Propagation (AUP). In IAUP modify the internal process by giving the update transaction the high priority to be applied directly as soon as it reaches the secondary replica site. IAUP has two versions, one at the master site which is stated in Figure 2 and the other at the secondary replica sites and is stated in Figure 3.

#### Algorithm: improve aggressive update propagation at master site

**Input:** update transaction commit at master site for Fi

**Output:** the replica sites have receive the update message.

1. **Begin**
2. Increase Fi timestamp by 1
3. Prepare the update message
4. Read the Meta data file for file Fi
5. **For** each record do
6. **Send** the update message
7. **Receive** acknowledgement
8. **End**

![Figure 2: Improver aggressive update propagation at master site](image)

Update messages are sent to all replica sites after the update transaction is committed successfully at the master site. At the receiving replica site, the algorithm is started as soon as receiving an update message. A correct data version will be read by cloud user from a given data storage only after the update transaction has been commit at that data storage, otherwise the transaction will read inconsistent data. Multiple updates may create file inconsistencies due to the different arrival time to the remote replica sites which may create updates conflicts. Resolving these conflicts depends on the semantics and the requirements of the applications.

To ensure the correct semantics of the update propagation technique, a sensible arriving order of non-commutative update operation has to be defined and enforced over the whole replica group. In cloud computing environment each file has its owner; the owner is the only site which has a role to do update operation for that file. This implies that all the updates operation done by different sites are commutative, and no need for any sensible arriving order between them. On the other hand, the update operations initiated by the owner site are expected to be non-commutative and it is necessary to guarantee a sensible arriving order for those non-cumulative updates.

In a single master scenario, there is one sender and a set of receivers; in this case the enforcement of the FIFO order is needed. The FIFO order requires that all requests from the same sender to be delivered First-In-First-Out in all receivers. This means that the updates initiated by the same master replica site are to be executed by other secondary replica sites in the order they are sent. In the IAUP technique the reconciliation method supports using FIFO order.

#### Algorithm: improve aggressive update propagation at Replica site

**Input:** receive an update message

**Output:** the replica site is up-to-date

1. **Begin**
2. Get the incoming message m from incoming queue
3. En-queue m the update queue
4. If m-timestamp - local replica timestamp\(=1\)
5. Apply the update message to the local replica
6. Commit
7. Send acknowledgement to master site
8. Else
9. Send a Pull message to master replica site
10. Wait until receive new update message
11. **End**

![Figure 3: Improved Refreshment algorithm at Replica site](image)
FIFO order: for the applications that the owner updates are non-commutative a FIFO order is supported. FIFO order is implemented by the proposed update propagation technique as follows: Each file is initially assigned with a timestamp, whose initial value is 0. For each update operation over a given file Fi increases the file timestamp by one. This timestamp is attached to the update message. The remote site processes the update message locally only if the timestamp in the received message is equal to the local timestamp + 1; otherwise the site logs this message for later processing, and sends an order-pull message to the sender. After receiving the missing update message(s), the site performs the updates in order. In general by pull message, a secondary replica site requests the master site to send to it any updates the server has at that moment. When the sender receives an order-pull message, first it checks the timestamp of the Order-Pull Message and compares it with its local timestamp, and find out the missing messages and sends it back to the pulling site, which process the updates locally.

4. Performance Analysis
In order to evaluate the methods we illustrated, a discrete event simulator has been developed using a C++ language. The number of inconsistent read and average response time of the servers are the primary metrics. Table 1 summarized the parameters used in the simulation.

We define the number of inconsistent read as counting the total number transactions read inconsistent data and the average response time as the average amount of time (in minutes) required to handle a request.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of machine</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>Number of transaction</td>
<td>5000</td>
</tr>
<tr>
<td>λ</td>
<td>Mean arrival time minutes</td>
<td>0.2</td>
</tr>
<tr>
<td>µ</td>
<td>Mean service time minutes</td>
<td>0.5</td>
</tr>
<tr>
<td>P</td>
<td>Probability of updates</td>
<td>1/10, 1/20, 1/30…1/100</td>
</tr>
</tbody>
</table>

Figure 4 reveals the simulation results of the number of inconsistent reads in LUP, AUP and IAUP. The horizontal axis indicates the probability of updates while the vertical axis indicates the average response time. The response time includes the needed data transmission time plus the transaction service time. So data transmission time is the most significant factor to influence response time, and it is essential to reduce it. When increasing the probability of updates, the average response time is getting higher in the three techniques. In terms of the LUP, more data transmissions need to be performed during job execution. Therefore, the lazy protocol has the longest average response time due to the rising number of data transmissions. The IAUP and AUP techniques has the minimal average response time, as can be seen from the experiments, without suffering from the long data transmission delay because the data are always up-to-date within a grid site.

5. Conclusion
In this study, the consistency of multiple data replicas in cloud data storage is discussed. We have suggests An Improved Aggressive Update (IAUP) technique in order to maintain replica consistency in cloud data storage. The simulation results improved clearly that the propose
technique introduce a high degree of consistency and improves the reliability without affecting performance of the system.

References

[1] Hoefer, C. and G. Karagiannis. Taxonomy of cloud computing services. in GLOBECOM Workshops (GC Wkshps), 2010 IEEE. 2010. IEEE.


Mohammed Radi has completed his PhD degree at 2009 in Database from University Putra Malaysia. His MSc was at 2003 in computer science from University of Jordan and BSc in 2001 in computer science from AlAzhar University Gaza. Currently he is an Assistant professor ant Al-Aqsa University - Gaza. His primary research interest is could computing, distributed system and distributed database.