Consistency Model for Collaborative Software Development on Cloud

Yin Nyein Aye, Thinn Thu Naing
University of Computer Studies, Yangon, Myanmar.
yinnyeinaye.ptn@gmail,thinnthu@gmail.com

Abstract

Consistency preservation is an important problem in collaborative system that is activated in both traditional distributed systems and cloud based systems for availability and performance. Especially, cloud storage services still need to improve the consistency guarantees in client centric approach. This paper describes a consistency model for constructing collaborative software development on cloud that intends to evaluate the effectiveness of the system by using Monotonic Write Consistency and Vector clock timestamp algorithm are applied to control the collaborative work on cloud in order to provide the time management. We discuss the effectiveness of our approaches compared to normal process and process with consistency model.

Keywords: consistency; collaborative work; software development.

1. Introduction

The advent of the cloud computing, replication has become the problem to deal with scalability issues, load variance and larger number of parallel requests. When a system is a consistent state, all replicas are identical and ordering guarantees of the specific consistency models are not violated. Many cloud data storage platforms use techniques to achieve high availability and low latency that avoid two-phase commit and synchronous access to a quorum of sites. [9] In the cloud environment, replicated design is widely adopted in collaborative systems which are to meet the requirement of high responsiveness. The need for different consistency level is depended on in a variety of application. In a distributed storage system there are two kinds of consistency models. These are data-centric and client-centric consistency models. Both types have advantages and disadvantages for the analysis of consistency guarantees that are depending on the issues of interest. [7]

Client-centric consistency models do not know about the internal state of a storage system. Instead they focus on the consistency guarantees which can observe by one or more clients whether stale data is returned or not. If no state data is observed, the client is satisfied. A user or client can check whether a storage system meets its consistency or not. D.Bermbach et al. [11] presents

efficient algorithms by analyzing the suggestion of interactions between the client and a key-value store and evaluated atomic, regular and safe consistency. However, this paper is checking consistency is an offline technique which has certain limitations.
Software development is a collaborative process where teams of developers work together in order to complete tasks. Software development requires enforcing consistency for providing each individual with the modifications performed by all others developers. For example, a web-based collaborative document editing application likes Google Docs. Two users edit a document at the same time and send their changes to the application. This paper will deal with the problem of consistency by proposing a model which considers on software development. Section 2 presents our proposed system architecture, section 3 explains a consistency model for target scenario and section 4 describes the related work. Finally, we conclude this paper and directions for future work.

2. System Architecture

In this section, the system architecture is described in Figure 1 which shows collaborative work on cloud consists of four parts. These are three groups are involved in this system. Developing Workgroup is a group of developers in different areas prepare writing code together. Management Workgroup is a group of managers in different roles collaborate with each other to work. Outsourcing Workgroup is a group of developers in outsourcing complete the task with connection of managers which manage the task to finish successful. Cloud storage is used developers, managers and developers in outsourcing are encouraged to finish the task with consistent state. Consistency maintenance is required to keep the views that consistent under change when the developers update their data.
Developers create and modify new changes of a component based on their current timestamp. The current timestamp and allows it to be exported for other developers to use. Another developer may subsequently import and merge. Sometimes, the developers do not aware of the conflict after modify or update the data performed by other developers. Therefore, if we would like to get update data, we implement consistency model for this system. The objectives of the system is to keep consistency for developers working together in software development on cloud, to support for developers interaction at a distance that maintain awareness of the state and activities of others and to accomplish the desired task without failure in synchronous communication.

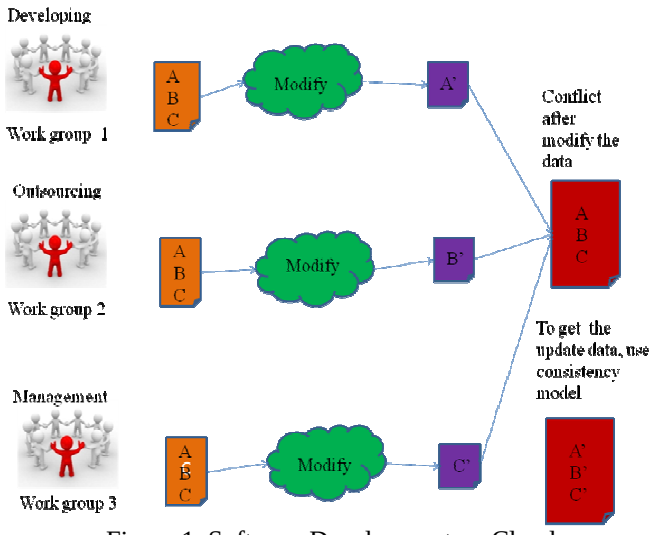

Figure 1. Software Development on Cloud

3. Consistency Model

A consistency model is an agreement between processes and the data store. If the processes agree to follow certain rules, the store keeps promises to work correctly. Maintaining consistency within a single database node is relatively easy for most databases. Most parts of the document which are frequently edited by several persons would be handled by strong consistency guarantees to avoid conflicts all together. Distributed shared systems are designed as different consistency models to achieve high performance of operations on shared data.

3.1 Monotonic Write Consistency and Vector Clock Algorithm

The consistency of the data not only depends on the local operations but also on the operations taken on the other copies. Shared data are replicated at the local storage of each site, so that operations can be performed at local sites instantaneously and propagated to remote sites. Software development is a collaborative process where teams of developers work together in order to complete tasks.

Software development requires enforcing consistency for providing each individual with the modifications performed by all others developers. Multiple views of developers are required to keep consistent under change. The requests of the operations are handled by developers synchronously and these operations are performed by the shared objects consecutively. In this system proposed a consistency model by combining Monotonic Write Consistency and Vector Clock algorithm are applied to control the software development on cloud.

An operation by a developer on a data item (A) is completed before any consecutive write operation on A by the same process. Two updates by the same

client will be serialized in the order that they arrive at the storage system. To avoid lost updates when an application /process first writes and updates data but the update is executed before the initial write and overwritten. Monotonic write consistency is used for a collection of concurrent processes. By using monotonic write consistency, it is shown that a write operation by a client process on a local data item local copy (c) is completed before any successive write operation on c by the same process. Two or Three updates by the client will be serialized in the order. If there are two consecutive writes by the clients and a replica has already written the value of the second write.

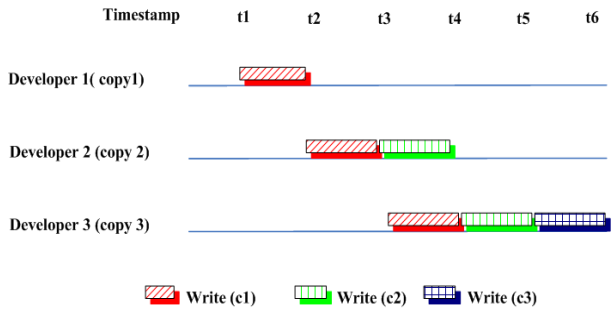

Figure 2. Monotonic Write Consistency

In Table1, the definition of notations is shown. A collaborative work system consists of a subset of local information of the clients; we denote that c_1, c_2, c_i which each sequence of steps called events. The sequence of events executed by process c_i in an execution of the system the local history for that process. The order in which the events occur in the local history of a process p is called the order of p. The entire execution of the system is called the total history denoted H_{CS} and is a partial ordered set of events formed by the union of the local histories of the process. WC_i contains writes issued by the clients C_i. The vector clock can be used to ensure that updates are processed in order. The vector clock timestamp can be used to ensure the changes are processed in order. Each node including vector clock maintains a change number or event. The vector clock timestamp is a list including the current node's change number as well as the change numbers that have been got from other nodes. When an update is transmitted, the vector timestamp is included with the update timestamp and the receiving node compares that other vectors because it can be determined that updates are being received out of sequence or not.

Table 1. Definition and Notations

C	A subset of local information of the clients $C=\{\{c_1\},\{c_2\},,\ldots,\{c_i\},\}$, For example: C={{Developer},{Manager}}
W	Write Vector
op	operation
CS	Cloud Storage
V	Number of Vectors Clock Timestamp $V=\{v_1,v_2,\ldots,v_i\}$
i	Number of Clients
H_{CS}	History of Cloud Storage

Client class associate with one or more cloud storage server class. Client class has two specialized class: Developer class and Manager Class. The developers or managers are allowed writing code. When they are writing code, the vector clock can be used to ensure that updates are processed in order to get monotonic write consistency.

Fig.3. Data Structure of the system

initialize $W \leftarrow 0, V_{CS} \leftarrow 0, H_{CS} \leftarrow \{\}$
Occurrence of local event:
If iswrite(op) then
  $W[i] = W_{Ci}[i]$
end if
send(op,W) to CS

perform op and store results in CS
if iswrite(op) then
  $V_{CS}[i]=V_{CS}[i]+1$
  $H_{CS}=H_{CS} \cup \{op\}$
  send(op,CS,V_{CS})to C_i
end if

if iswrite(op) then
  $W[i]=W_{Ci}[i]+1$
end if

In figure 4 is shown to complete the task with monotonic write consistency and vector clock timestamp. Developer 2 starts working on a class PASSAGE. He/she gets data as a local copy from the Cloud. The system combined monotonic write consistency and vector clock algorithm, displays the class current version at developer 2 site. And then developers start modifying the class. In the meanwhile developer 1 starts working on the same PASSAGE class. The

system makes integration the two write operations by two developers. There is no need for complex communication because of using Monotonic Write Consistency and Vector Clock algorithm both developers are aware of which parts have been changed. Clients can not just create a new program but also run old programs. If client wants to get a program, first developers get data from cloud storage server. And then developers save the program on each client site. When client wants to edit data, every action is counted by vector clock.

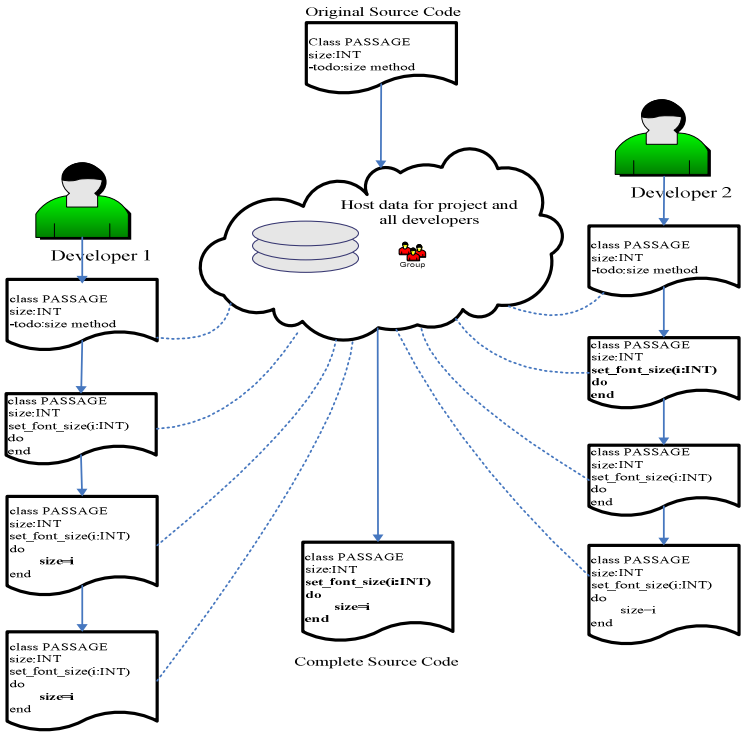

Figure 4. Co2Cloud

The analysis of the algorithm is O (n). The difference between the normal process and process with consistency model is shown in Table2. We would like to compare the evaluation results from the following two processes.

Table2. Normal Process Vs Process with Consistency Model

Metric	Ordering	Staleness	Discrete	Continuous
Normal Process	×	√	√	×
Process with Consistency Model	√	×	×	√

4. Related Works

D.Bermbach et al. [1] present an approach to increase consistency guarantees by using a middleware component running on the same server-side machines as the application code. These middleware service uses vector clocks and client side caching to guarantee monotonic read consistency as well as read your write consistency. In this paper [8] develops and analyzes a transaction management and replication protocol based on implementation of the Paxos with Combination and Promotion (Paxos-CP) that provides true consistency. In [2] describes four session guarantees are proposed to aid users and applications of inadequately consistent replicated data. These session guarantees present individual applications with a view of the database that is consistent with their own actions, even if they read and write from various, potentially inconsistent servers. Google's BigTable [4] provides eventual consistency guarantees. Existing Cloud datastores usually deal consistency for performance and availability. However, it is not clear how an application is affected when it runs under a low level of consistency. In fact, current application designers have no tools that would help which and how many inconsistencies occur for their particular application. Previous results show that consistency violations occur at least from time to time and applications must be able to cope with it. It allows to quantify staleness or to count violations of, e.g., MRC.
S.Patterson et al. [8] describe what types of inconsistency are seen in the results returned from operations, and how often these situations arise. As previously mentioned, most collaborative applications are used in only distributed computing. Cloud Computing allows people to use applications without installing them on their computers and allows access to save files from any computer. Therefore we propose a consistency model for constructing collaborative software development on cloud to complete the task with fast access and without conflict.

5. Conclusions

The developers are tried to state their tasks by using only monotonic operations to solve each updating the set of possible situations and what the code must be written in this target scenario to keep consistency. Vector clocks resolve all conflicts to identify the ordering of different versions. This research intends to address the importance of efficiency when designing a consistency model for software development on cloud. For future work, we are planning to further investigate issues concerning consistency and system performance by calculating the complexity of the system in order to get the availability and maintainability. Moreover, we reduce the overhead of the system.

References

[1] D.Bermbach, J.Kuhlenkamp, B.Derre, M.Klems, S.Tai, "A Middleware Guaranteeing Client-Centric Consistency on Top of Eventually Consistent

Datastores", Karlsruhe Institute of Technology, Karlsruhe, Germany,April,2013.
[2] D.B. Terry, A. J. Demers, K. Petersen, M. J. Spreitzer, M. M. Theimer,and B. B. Welch, "Session Guarantees for Weakly Consistent Replicated Data" , Computer Science Laboratory, Xerox Palo Alto Research Center Palo Alto, California.
[3] G.Convertino, U.Farooq, M.B Rosson, and J.M. Carroll, "Old is Gold: Integrating Older Workers in CSCW", Proceedings of the 38th Hawaii International Conference on System Sciences, 2005, pp. 1-10.
[4] F. Chang et al, "Bigtable: A Distributed Storage System for Structured Data". In Proc. of OSDI, pages 205–218, 2006.
[5] G.DeCandia, D.Hastorun, M.Jampani, G.Kakulapati, A.Lakshman, A.Pilchin,S. Sivasubramanian, Peter Vosshall and Werner Vogels, "Dynamo: Amazon's Highly Available Key-value Store", SOSP'07, October 14–17, 2007, Stevenson, Washington, USA.
[6] A.Karsenty and M.Beaudouin-Lafon. "An algorithm for distributed groupware applications". In Proceedings of the 13th ICDCS, pages 195–202, 1993.
[7] A.Tanenbaum and M. van Steen, "Distributed Systems: Principles and Paradigms". Prentice Hall, 2002
[8] S.Patterson, A.J.Elmore ,F.Nawab, D.Agrawal, A.El Abbadi, "Serializability, not Serial: Concurrency Control and Availability in Multi-Datacenter Datastores", August 27th - 31st 2012, Istanbul, Turkey.
[9] H.Wada, A.Feketez, L.Zhaoy, K.Lee and A.Liu, "Data Consistency Properties and the Trade-offs in Commercial Cloud Storages: the Consumers' Perspective", CIDR'11 Asilomar, California, January,2011.
[10] D.Bermbach, J.Kuhlenkamp, "Consistency in Distributed Storage Systems", Karlsruhe Institute of Technology, Karlsruhe, Germany.
[11] D. Bermbach and S. Tai, "Eventual consistency: How soon is eventual? an evaluation of amazon s3's consistency behavior," in Proceedings of the 6th Workshop on Middleware for Service Oriented Computing. ACM, 2011.