Merging OT and CRDT Algorithms

Mehdi Ahmed-Nacer, Pascal Urso, Valter Balegas, Nuno Preguiça

Merging OT and CRDT Algorithms*

Ahmed-Nacer Mehdi[†] & Pascal Urso[‡]
Université de Lorraine
LORIA, INRIA

Valter Balegas[§] & Nuno Perguiça[¶]
CITI/FCT-Universidade
Nova de Lisboa

ABSTRACT

Nowadays, a large number of collaborative editing applications have been developed. Some of them are deployed on the cloud such as Google Drive and Microsoft Office at SkyDrive. Massively used editing systems make use of operational transformation (OT), a traditional replication mechanism for concurrent document editing. Such algorithms do not scale well in peer-to-peer environments with dynamic groups. Recently, Commutative Replicated Data Types (CRDTs) were introduced as a new class of replication mechanisms whose concurrent operations are designed to be natively commutative. They ensure consistency of highly dynamic contents on peer-to-peer networks.

Through this paper, we propose an architecture to take advantage of both approaches – OT and CRDT – and to improve the performance of collaborative editing applications. We merge both algorithms on the proposed architecture and we study their suitability.

Categories and Subject Descriptors

I.7.1 [**Document and Text Processing**]: Document and Text Editing; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, erformance measures*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Algorithms, Experimentation, Performance

Keywords

Collaborative Editing, Commutative Replicated Data Types, Operational Transformation, Cloud, Algorithms, Experimentation, Performance

*This research was partially supported by FCT/MCT projects PEst-OE/EEI/UI0527/2011 and PTDC/EEI-SCR/1837/2012; by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 609551, SyncFree project.

†mahmedna@loria.fr

‡pascal.urso@loria.fr

§v.sousa@campus.fct.unl.pt

¶nuno.preguica@fct.unl.pt

1. INTRODUCTION

Collaborative editing systems allow multiple users to work over the same shared documents.

To support disconnected work and achieve high responsiveness, data are optimistically replicated [9]. CAP theorem [4, 6] states that it is impossible to achieve simultaneously strong consistency (C), availability (A) and to tolerate network partition (P). In *Eventual Consistency (EC)* model, the replicas are allowed to diverge, but must eventually reach the same value if no more mutations occur. Eventual consistency promises better availability, performance and can be obtained in large-scale systems.

It is essential to maintain the user's perceived latency low, otherwise users may get frustrated and quit the application. In addition, to achieve high availability, operations can be executed locally without coordination, which induce replica divergence.

In real-time collaborative editing applications, the modifications are propagated to other members of the group upon execution. In recent years, a new generation of real-time collaborative editing tools have been developed. These applications are deployed on the cloud such as Google Drive [1] and Microsoft Office at SkyDrive [2]. They aim to provide a good environment for collaboration by supporting large number of users and provide a considerable data storage.

Operational transformation (OT) has been used in real-time collaboration [5, 11] for decades. This approach transforms the index of an operations to take into account the effects of concurrent operations and assure replica convergence. There are different architectures for deploying OT systems: centralized solutions use a central sequencer, which fails to provide low latency for remote client [12]. On the other hand, decentralized solutions use event tracking mechanisms that incur in high meta-data overheads that grow with the number of clients [11]. A few commercial document editing systems such as Google Docs adopt centralized OT algorithms, such as Jupiter [8]. This kind of algorithm are light in the client devices. However, in large system a consensus is needed between data-centers to synchronize the distributed sequencer. Due to the disadvantage of the centralization and the concurrency control, the consistency is not achieved and the fault tolerance is not supported [4, 6]. Regardless the architecture used in the deployments, OT has been recognized to be difficult to reason about [3].

CRDT (Commutative Replicated Data Types) [10] approaches were proposed in order to ensure convergence without blocking client operations and without having to deal with consensus.

In this paper, we investigate the combination of both protocols – OT and CRDT – to provide low-latency and scalable real-time collaboration tools, in this case a text editor. Our preliminary results show that the proposed architecture is able to mitigate to overheads of each solution, which supports our objectives.

2. ARCHITECTURE

The deployment of our system consists of client and server nodes. Clients run the OT algorithm and can execute in the user device, or in a remote machine where a web interface is exposed to the client. Servers execute in the data-center and execute two protocols: The OT algorithm is used to accept clients operations within the same region; a CRDT algorithm is used to handle inter data center replication.

Figure 1 shows the deployment of our systems: The clients execute operations locally and reply to the client before transforming the operations; The OT sequencer is executed in the same data center and orders clients operations; After the server integrates an OT operation, it generating a new CRDT operation that will be broadcasted to the other data centers. A server that receives a CRDT operation generates a new local OT operation that is executed and forwarded to the local clients. The two operations are executed in isolation to ensure that each pair of OT and CRDT operations generated at the server are equivalent.

The benefit of the approach is that operations on the client node have their meta-data and latency reduced because they use OT algorithm. Operations are propagated between data centers using CRDTs, which avoids using a global sequencer and since the CRDT only handle data centers operations, meta-data is kept small (proportional to the number of data centers). The trade-off is that the servers have to execute each operation twice, but we can expect the data centers to have great computational power.

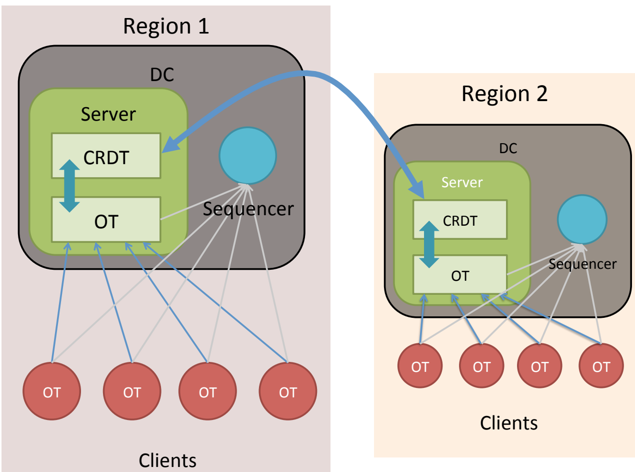


Figure 1: Hybrid deployment of the collaborative text editor.

3. STUDIED ALGORITHMS

SOCT4 algorithm [12] is a representative OT algorithm evaluated in this paper. In this algorithm, the operations are ordered globally using a timestamp given by a sequencer. They are delivered then on each site in this order. When a causally ready operation is integrated at a site, the whole log of operations is traversed and reordered. After reordering, causally preceding operations come before concurrent ones in the history buffer. Finally, the remote operation has to be transformed according to the sequence of concurrent operations. To control the concurrency, OT algorithms keep the operations in the history. Unfortunately, the history buffer grows over time and the integration of the operations takes much time.

Logoot Algorithm [13] is CRDT approach that ensures consistency of linear structures. Logoot associates to the list of elements of the structure, an ordered list of identifiers. Identifiers are composed by a list of positions. Positions are 3-tuples formed with a digit in specific numeric base, a unique site identifier and a clock value. When inserting an element, Logoot generates a new identifier. Identifiers have unbounded lengths and are totally ordered by a lexicographic order. So a new identifier can always be generated between two consecutive elements. The disadvantage of Logoot is the size of the identifier that can grow unbounded.

4. EXPERIMENT

To evaluate algorithms performance, we designed the architecture described previously in Java and integrate a simulator to generate the operations. We simulate a real collaboration between 20 clients following the features obtained from [3]. The simulator lets clients of every algorithm to generate operations in its own formats. We deployed the experiment in local machines and we measured the average execution time that take each client to generate/integrate an operation in figure2a and 2b, and the average size of messages exchanged between the clients in figure 2c. The framework uses java.lang.System.nanoTime() for the measurement of execution time and the default serialization interface f Java to estimate the size of messages.

To obtain the presented results, we ran all algorithms on the same JVM execution. The experiment was made on Amazon EC2 instance, with dual processor machine with Intel(R) Xeon(R) 5160 dual-core processor, that has installed GNU/Linux 12.0.4 LTS.

To observe the benefits of our proposed architecture, we made two experiments:

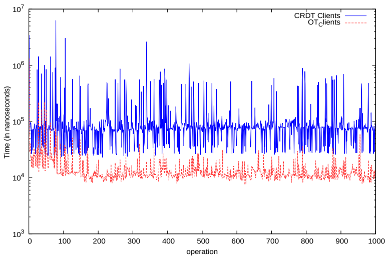
1. OT/CRDT: We deploy SOCT4 algorithm in each client, and keeping the data-centers with OT-CRDT.
2. CRDT/CRDT: We deploy in both clients and data-centers only CRDT algorithm.

The local execution time in SOCT4 algorithm is much better than Logoot algorithm. Indeed, Logoot algorithm needs to generate a unique identifier for each operation during the local execution. While, SOCT4 algorithm executes the local operation directly and do not need any identifier.

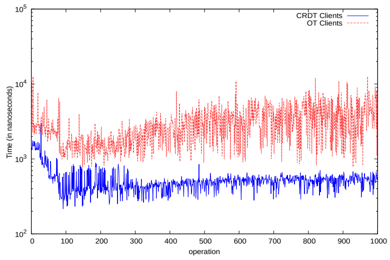
Accordingly to the nature of OT algorithms, the performances of SOCT4 eventually degrade over time, during the integration of the remote operation. Indeed, the algorithm has to parse all history of operations and transform the current operation with the concurrent one. However, using Logoot algorithm, the client has just to find the correct position of the identifier.

The size of messages sent by the clients that use SOCT4 algorithm is much lower than generated by CRDT algorithm. In addition to the content of the operation, Logoot integrates also the identifier of the operation. The size of these identifiers grows quickly when the clients insert between two consecutive characters, for instance a case of copy/paste. While, SOCT4 algorithm has just to add an integer in the operation – timestamp – to help the clients to detect the concurrent operations.

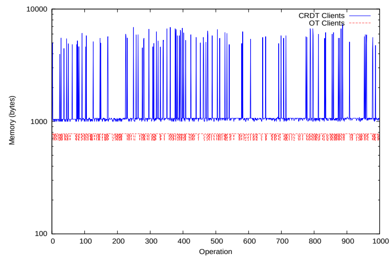
The architecture based on OT/CRDT should be more efficient than CRDT/CRDT. Indeed, OT/CRDT perform well in local execution time and size of messages. So, the latency should be reduced



(a) Local execution time



(b) Time of integration



(c) Messages size

between the clients. The only disadvantage of this combination is during the integration of the remote operation. However, the performance remains acceptable since they do not exceed 50ms [7]. In order to reduce the time of the integration of a remote operation, it is possible to prune the history buffer by using a garbage collection mechanism [11]. Clients can use this mechanism to remove operations they know to be received by all other clients.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed an architecture of the collaborative editing application deployed on the cloud. We evaluated the performance of OT and CRDT algorithms and we studied their suitability in such applications.

We found that, collaborative editing applications can be improved by merging OT and CRDT approaches. Based on this solution, the applications support more developers, reduce the flow in the network and the latency between the clients. In addition, the proposed architecture does not require a consensus across the data-centers to synchronize the distributed sequencer. In contrary to centralized OT editing systems, our solution ensures eventual consistency, availability and the partition tolerance.

In this paper, all the clients have been executed in the same EC2 machine, we computed the average local/remote execution time and the size of messages for each client. Our directions for future work is to deploy this architecture in the cloud, where each client and server is deployed in independent machine. Then, we study the behaviors of algorithms, and measure the delay that take an operation between the clients.

6. REFERENCES

[1] Google drive. https://drive.google.com.

[2] Microsoft skydrive, 2013. https://skydrive.live.com/.

[3] M. Ahmed-Nacer, C.-L. Ignat, G. Oster, H.-G. Roh, and P. Urso. Evaluating crdts for real-time document editing. In ACM, editor, *ACM Symposium on Document Engineering*, page 10 pages, San Francisco, CA, USA, september 2011.

[4] E. A. Brewer. Towards robust distributed systems (abstract). In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, PODC '00, pages 7–, New York, NY, USA, 2000. ACM.

[5] C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. In J. Clifford, B. G. Lindsay, and D. Maier, editors, *SIGMOD Conference*, pages 399–407. ACM Press, 1989.

[6] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33:51–59, June 2002.

[7] C. Jay, M. Glencross, and R. Hubbold. Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment. *ACM Transactions on Computer-Human Interaction*, 14(2), August 2007.

[8] D. A. Nichols, P. Curtis, M. Dixon, and J. Lamping. High-latency, low-bandwidth windowing in the jupiter collaboration system. In *Proceedings of the 8th annual ACM symposium on User interface and software technology*, UIST '95, pages 111–120, New York, NY, USA, 1995. ACM.

[9] Y. Saito and M. Shapiro. Optimistic replication. *ACM Computing Surveys*, 37(1):42–81, 2005.

[10] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. In X. Défago, F. Petit, and V. Villain, editors, *Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 6976, pages 386–400, Grenoble, France, October 2011.

[11] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 5(1):63–108, March 1998.

[12] N. Vidot, M. Cart, J. Ferrié, and M. Suleiman. Copies convergence in a distributed real-time collaborative environment. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 171–180, New York, NY, USA, 2000. ACM.

[13] S. Weiss, P. Urso, and P. Molli. Logoot: A scalable optimistic replication algorithm for collaborative editing on p2p networks. In *29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009)*, pages 404 –412, Montréal, Québec, Canada, jun. 2009. IEEE Computer Society.