Summary « Evaluating CRDTs for Real-time Document Editing »

Ronan-Alexandre Cherrueau
Adrien Bougouin
Alban Ménager

Year 2011-2012
(University of Nantes)

Nowadays, collaboring work is increasing and real-time editing systems are catching on. Google Docs is a good example, since it enables multiple authors to edit the same document at the same time, whenever it is and wherever they are.

By now, real-time editing systems use a replication mechanism to ensure consistency when merging concurrent changes performed on the same document. Optimistic replication gives to users a low time of latency. However, they use a centralized approach which doesn't only store the documents information but the personal one as well and and it may as well be a privacy threat if they are used by corporations.

So, to prevent this, a good solution for not using centralized mechanisms is to replace them by an decentralized ones. So, Peer-to-peer seems to be a solution. Nevertheless, the main key factor is performance. Obviously, if the editing application can't respond to users' actions in a reasonable time (about 50ms), they may get frustrated and stop the current application.

The goal of this experiments is to select some algorithms based on optimistic replication and evaluate them on a decentralized real-time collaborative editing system. The evaluations will be based on real context on the sames conditions and using the same data flow.

The first approach is *Operation Transformation*. This approche takes into account the effect of concurrents operations. An *Operation Transformation* is used to keep document consistency after severals concurrents operations. Google Docs use an algorithm named *Jupiter* and use a vector clock to detect concurrents operations, but this solution doesn't get on well in the peer-to-peer system.

A new approach called *Commutative Replicated Data Types* (CRDT) for peer-to-peer environment was introduced as a new class of replication mechanisms to preserve consistency. CRDT doesn't required concurrent operations detections because it's designed for concurrent operations to be natively commutative.

CRDT has some features :

- The concurrent operations are natively commutative.
- The document is a linear sequence of elements.
- A single position identifier.

Regarding to experiments, they select differents algorithms to generate the single position identifier :

- Logoot
- RGA
- WOOT
- WOOTO
- WOOTH

The figure 1 p.2 (with R the number of replicas and by H the number of operations on the document) shows the theoretical evaluation of these differents algorithms. Thus we see that RGA and Logoot have the bests results.

Algorithm	LOCAL		REMOTE	
	INS	DEL	INS	DEL
WOOT	$O(H^3)$	$O(H)$	$O(H^3)$	$O(H)$
WOOTO	$O(H^2)$	$O(H)$	$O(H^2)$	$O(H)$
WOOTH	$O(H^2)$	$O(H)$	$O(H^2)$	$O(log(H))$
Logoot	$O(H)$	$O(1)$	$O(H.log(H))$	$O(H.log(H))$
RGA	$O(H)$	$O(H)$	$O(H)$	$O(log(H))$
SOCT2/TTF	$O(H+R)$	$O(H+R)$	$O(H^2)$	$O(H^2)$

Figure 1 – Worst-case time-complexity analysis

But this is just theoretical. The team who wrote the paper designed real-time peer-to-peer collaborations application in order to obtain there logs and apply them on the algorithms. About the real applications, two experiments and severals groups have been mades :

- 3 groups have to do their semester report by only using the collaborating editor for one hour and a half :
  - 2 groups of 4 students.
  - 1 group of 5 students.
- 9 groups of 2 students have to translate an episode of *The Big Bang Theory*

Figure 2 p.3 shows the amount of users/character operations :

- A user operation is adding/deleting letters or groups (copy/past).
- A character operation is the transformation of each user operation into a character operation (for example, copy/past "alma" are the character operations :
  - adding 'a' at space 0
  - adding 'l' at space 1
  - ...

	Report			Series	
	Group 1	Group 2	Group 3	Doc 1	Doc 2
No. user operations	11 211	11 066	13 702	9 042	9 828
No. char. operations	26 956	47 992	42 443	29 882	10 268
% of del	12	12	12	9	5

Figure 2 – Total number of user/character operations

All the algorithms have been runned 10 times with same data and the same conditions. Figures 3 and 4 p.3 shows the total number of users operations.

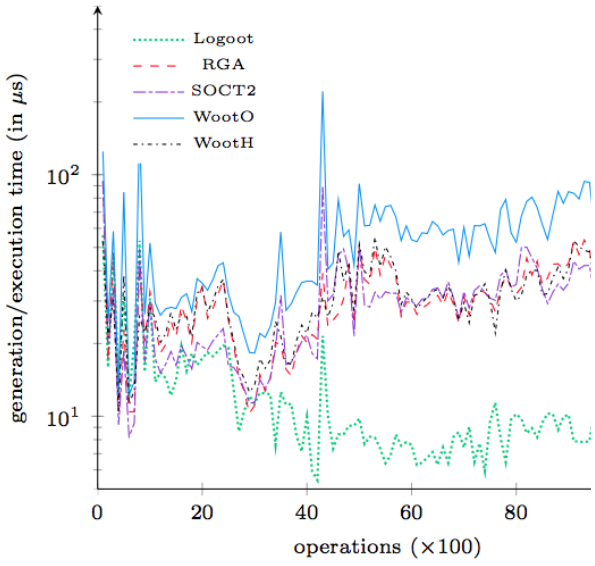

Figure 3 – User operation execution times - 2nd group report

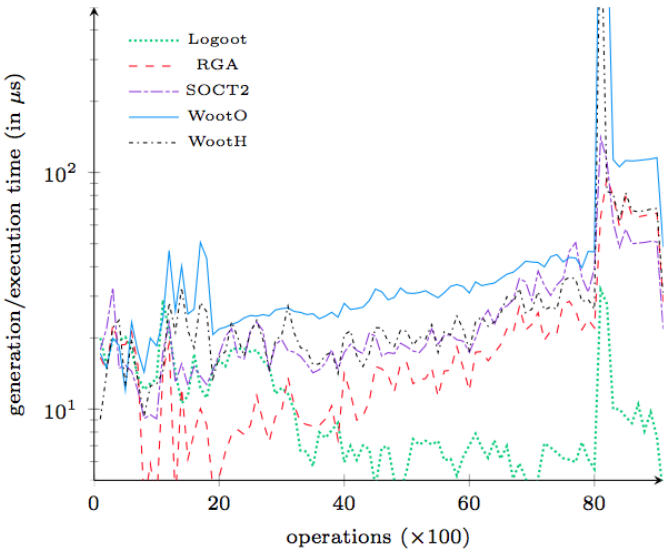

Figure 4 – User operation execution times - 1st series

The increases are due to insert or delete a large group of characters on the document. All the algorithms decrease over the time because of the size of the document (the unique identifier gets bigger when there are a lot of operations) but logoot decreases slowly compared to the others.

Figures 5 and 6 p.4 shows the total number of characters operations.

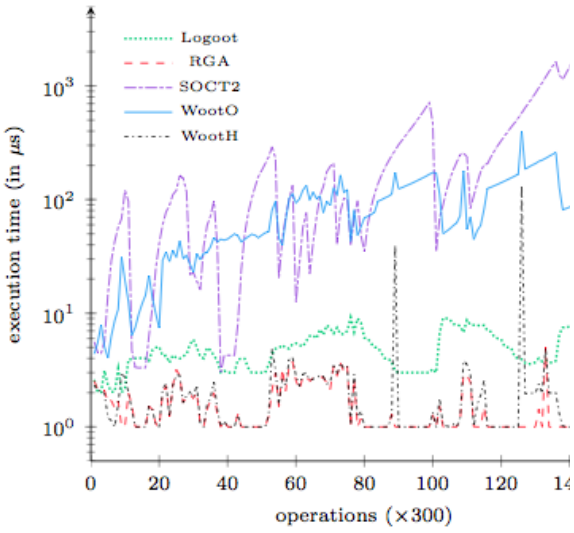

FIGURE 5 – Character operation execution times - 2nd group report

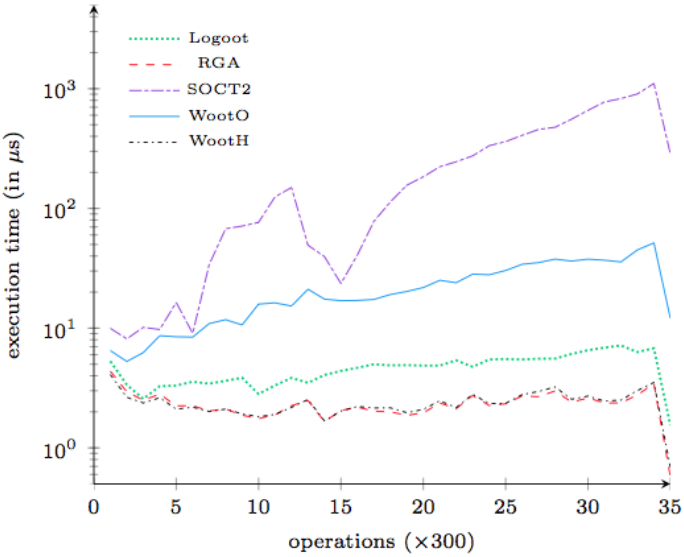

FIGURE 6 – Character operation execution times - 1 time series

Wooto, Logoot and RGA remains stable during al the time. The behavior of SOCT2 performance is mainly due to its garbage collection mechanism. Users have a period of inactivity and the garbage mechanism cannot purge the history log. SOCT2 can't be used for real-time editing because of his response time increasing 50ms quickly.

Figure 5 is a bite different than 6 because of the number of characters deletes (students didn't delete as much as in the serie experiments as in the report experiment). RGA and WootH are better than the others because their algorithms are based on hash tables.

In conclusion, this is the first performance evaluation of algorithms with real collaboration traces including concurrency. This experimentation proves the suitability of CRDT algorithms in real-time collaboration and they outperform some representative operational transformation approaches that were well established for real-time collaboration in terms of local generation time and remote integration time.