Improving Replicated Sequences Performances

Matthieu Nicolas, Gérald Oster, and Olivier Perrin

Université de Lorraine, CNRS, Inria, LORIA, F-54500, France

1 Introduction

2 Replicated Sequences

2.1 Sequence

2.2 CRDTs

2.3 LogootSplit

2.4 Limits

Besoin d'une section (ou autre) entre les limites de LogootSplit et le renommage pour justement introduire l'approche choisie (renommage) et les raisons (diminution de la taille des identifiants, aggrégation des blocs existants) – Matthieu

3 Renaming in a (de)centralized setting

3.1 System Model

3.2 Intuition

3.3 Renaming locally

3.4 Dealing with concurrent operations

3.5 Applying a remotely generated renaming operation

3.6 Garbage collection

3.7 Limits

4 Renaming in a fully distributed setting

4.1 System Model

4.2 Intuition

4.3 Strategy to determine leading epoch in case of concurrency

4.4 Transitioning from a losing epoch to the leading one

4.5 Garbage collection

5 Evaluation

6 Discussion

6.1 Offloading on disk unused renaming rules

- As stated previously, nodes have to keep renaming rules as long as another nodes may issue operations which would require to be transformed to be applied
- Thus nodes need to keep track of the progress of others to determine if such operations can still be issued or if it is safe to garbage collect the renaming rules
- In a fully distributed setting, this requirement is difficult to reach as a node may join the collaboration, perform few operations and then disconnect
- From the point of view of other nodes, they are not able to determine if this node disconnected temporarily or if it left definitely the collaboration
- However, as the disconnected node stopped progressing, it holds back the whole system and keeps the current active nodes from garbage collecting old renaming rules
- To limit the impact of stale nodes on active ones, we propose that nodes offload unused renaming rules by storing them on disk

Présenter une méthode pour déterminer les règles de renommage non-utilisées (conserver uniquement les règles utilisées pour traiter les x dernières opérations ?) – Matthieu

6.2 Alternative strategy to determine leading epoch

6.3 Postponing transition between epochs in case of instability

- May reach a situation in which several nodes keep generating concurrent renaming operations on different epoch branches
- In such case, switching repeatedly between these concurrent branches may prove wasteful *"costly" plutôt? – Matthieu*
- However, as long as nodes possess the required renaming rules, they are able to rewrite operations from the other side and to integrate them into their copy, even if they are not on the latest epoch of their branch
  - At the cost of an overhead per operation
- Thus not moving to the new current epoch does not impede the liveness of the system
- Nodes can wait until one branch arise as the leading one then move to this epoch
- To speed up the emergence of such a branch, communications can be increased between nodes in such case to ease synchronisation

6.4 Compressing the renaming operation

- Propagating the renaming operation consists in broadcasting the list of blocks on which the renaming was performed, so that other nodes are able to compute the same rewriting rules
- This could prove costly, as the state before renaming can be composed of many blocks, each using long positions
- We propose an approach to compress this operation to reduce its bandwidth consumption at the cost of additional computations to process it
- Despite the variable length of positions, the parts required to identify an position uniquely are fixed
  - We only need the $siteId$ and the seq of the last tuple of the position to do so
- Instead of broadcasting the list of whole positions, the node which performs the renaming can just broadcast the list of tuples $< siteId, seq >$
- On reception of a compressed renaming operation, a node needs first to regenerate the list of renamed blocks to be able to apply it
- To achieve so, it can browse its current state looking for positions with corresponding tuples $< siteId, seq >$
- If some positions are missing from the state, it means that they were deleted concurrently
- The node can thus browse the concurrent remove operations to the renaming one to find the missing blocks
- Once all positions has been retrieved and the list of blocks computed, the renaming operation can be processed normally

6.5 Operational Transformation

Ajouter une section sur OT pour expliquer que gérer les opérations concurrentes aux renommages consiste en finalité à transformer ces opérations, mais qu'on a décidé de ne pas présenter l'approche comme étant de l'OT dans ce papier pour des raisons de simplicité ? – Matthieu

7 Conclusion

References