Efficient renaming in Conflict-free Replicated Data Types (CRDTs)

Matthieu Nicolas

November 1, 2018

LogootSplit[1]

- State of the art of *Sequence Conflict-free Replicated Data Types (CRDTs)*[2]
- Allows nodes from a distributed system to replicate and edit sequences without any coordination

- Relies on *identifiers* to ensure convergence

Identifier-based approach

Main idea

- Generate and attach an identifier to each inserted element

Allow to achieve commutative updates

- By identifying uniquely elements
- By ordering them relatively to each other

Identifier constraints

- To fulfill their role, identifiers have to comply to several constraints:

Globally unique

- Identifiers should never be generated twice, neither by different users nor by the same one at different times

Totally ordered

- We should always be able to compare and order two elements using their identifiers

Dense set

- We should always be able to add a new element, and thus a new identifier, between two others

LogootSplit identifiers

- To comply with these constraints, LogootSplit proposes identifiers composed of quadruplets of integers of the following form:

$$< priority, siteId, seq, offset >$$

- *priority* allows to determine the position of this identifier compared to others
- *siteId* refers to the node's identifier, assumed to be unique
- *seq* refers to the node's logical clock, which increases monotonically with local operations
- *offset* refers to the element position in its original block

State of a LogootSplit sequence

- The state of such sequence corresponds to the elements contained in the sequence and their respective identifier
- For simplicity purpose, we will use letters instead of quadruplets of integer to represent identifiers
- We represent the state of a given sequence as the following:

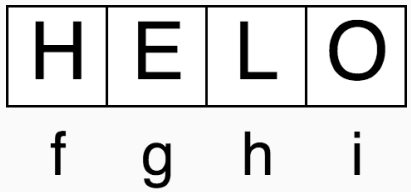

Figure: The state of a sequence which contains the elements "helo" and their corresponding identifiers

Blocks

- To reduce the identifiers' footprint, LogootSplit aggregates elements with *contiguous* identifiers into blocks
- Two identifiers are *contiguous* iff
  - They have the same size
  - All their components but their last *offsets* are equal
  - Given their respective last offsets o and o', o is the successor of o' or conversely
- It allows to reduce the metadata stored to only the identifier of the first element of the block and the last *offset* of the last element
- We represent blocks as the following:

Figure: The state of a sequence which contains the block "helo"

Operations

- Two local operations are defined on the data structure
  - *insert* allows a node to add elements at a given position. It generates and attaches identifiers to inserted elements.
  - *delete* allows a node to remove an interval of elements and their identifiers.

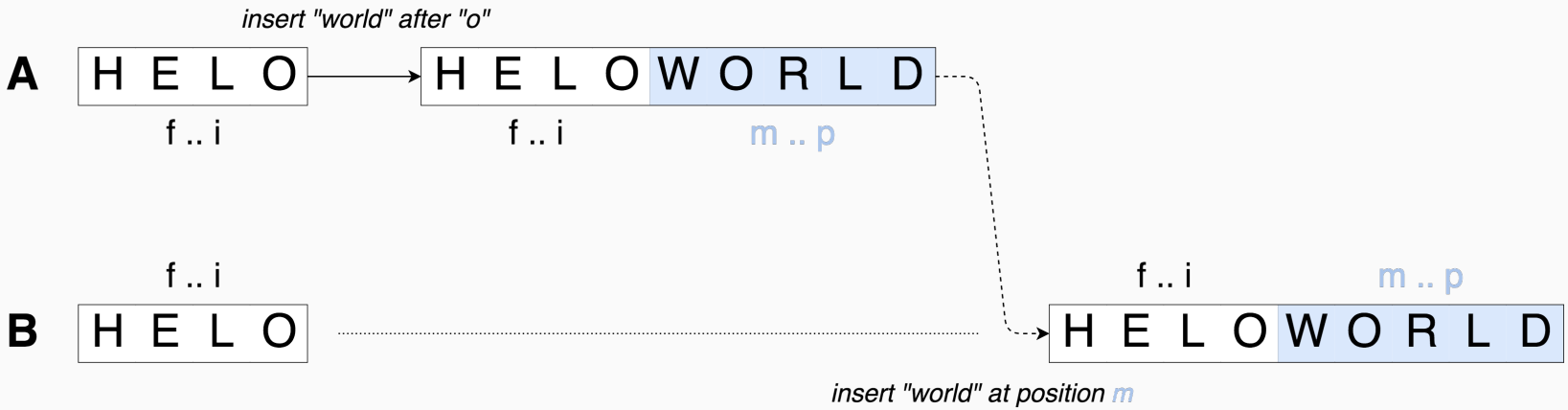

Figure: Example of *insert* operation

- Each local operation is then propagated to other nodes, using identifiers instead of simple integers as positions to ensure convergence

Growing identifiers

- However, a node may try to insert a new element between two others with contiguous identifiers
  - For example, between the identifiers g and h
- There is no mean to generate a new identifier id of same size such as $g < id < h$
- To generate a fitting identifier, LogootSplit concatenates the predecessor's identifier to a newly generated quadruplet of integers
  - For example, g is concatenated to m to generate a valid identifier

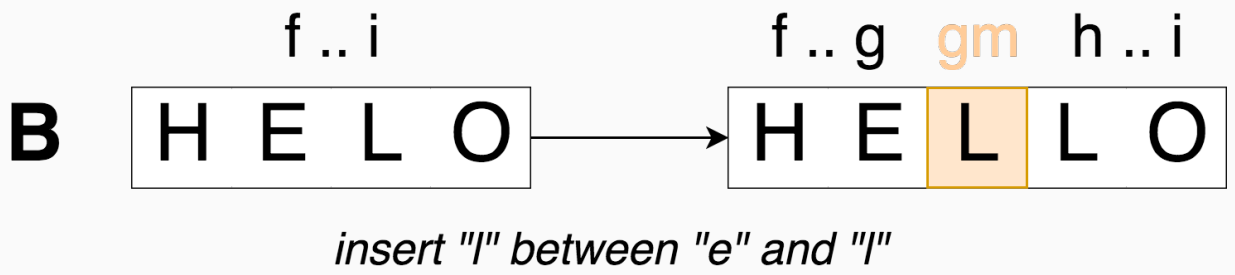

Figure: Insertion leading to the generation of longer identifiers

Declining performances

- Operations performed may lead to an inefficient internal representation
- With blocks containing few elements...
- ... and using long identifiers

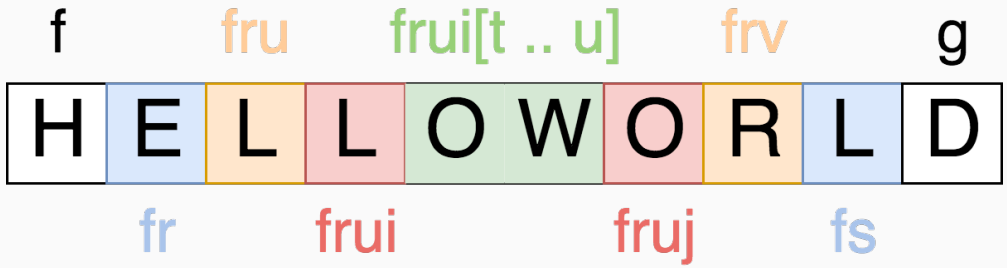

Figure: Example of inefficient internal representation

- The more blocks we have:
  - The more metadata we store
  - The longer it takes to browse the sequence to *insert* or *delete* an element

Renaming

- To address this issue, propose to introduce a *rename* operation
- This operation reassigns an identifier composed of one quadruplet of integers to the first element, based on his previous identifier
- It then generates contiguous identifiers for all following elements
- This allows to aggregate all elements in one block
  - Reducing the metadata footprint to the identifier of the first element and the *offset* of the last one

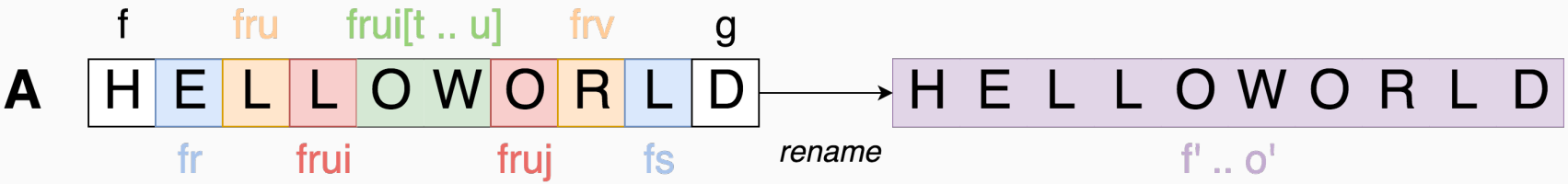

Figure: Example of renaming

Handling concurrent operations

- However, additional metadata is temporary required to handle concurrent operations
- This metadata is used to compute rewriting rules for identifiers inserted or deleted by other nodes concurrently
- It can be garbage collected once every node has observed the *rename* operation

Thanks for your attention!

References i

[1] L. André, S. Martin, G. Oster, and C.-L. Ignat.
Supporting adaptable granularity of changes for massive-scale collaborative editing.
In *International Conference on Collaborative Computing: Networking, Applications and Worksharing - CollaborateCom 2013*, pages 50–59, Austin, TX, USA, Oct. 2013. IEEE Computer Society.

[2] M. Shapiro, N. M. Preguiça, C. Baquero, and M. Zawirski.
Conflict-free replicated data types.
In *Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, SSS 2011, pages 386–400, 2011.