UDC 004.4

PURE OP-BASED REPLICATED LIST WITH CARDINALITY CONSTRAINTS

Semenov G.V. (ITMO University)
Научный руководитель – к.ф.-м.н. Грищенко В.С. (НИУ ВШЭ)

Introduction. Distributed systems and local-first applications employ an optimistic replication approach to ensure high availability of data. Generally, a conflict resolution strategy shall be presented to merge divergent revisions of data and ensure system-wide eventual consistency. Conflict-free replicated data types avoid this problem by implying a semilattice (state-based CRDT, or CvRDT) or a commutative operational basis in presence of causally-ordered broadcast protocol (op-based CRDT, or CmRDT). Portfolio of known CRDTs features various types of counters, registers, collections, collaborative editing types, graphs etc.

Generally, a replicated list is a CmRDT providing a dynamic array interface with modifying operations of insertion and deletion by index. Though it is clear that insertion operations at the same index cause conflict in terms of user intentions, they are deemed conflict-free in presence of causal relationship (e.g. vector clock) or a partial ordering tag (e.g. unix timestamp). Various implementations of replicated lists are known, such as WOOT, Treedoc, Causal Tree, RGA, Chronofold, Fugue etc.

In this thesis a simple pure op-based replicated list data type implemented using a Cartesian tree set – *replicated dense list* – is proposed. It is constructed on an ordered dense space, supports range move operation and is implementation-agnostic. Furthermore, it facilitates manual conflict resolution made by users a posteriori to restore original intentions. We also show that additional cardinality semantics may be seamlessly added to a dense list.

Main part. Firstly, we define *tag space* $T = \left(T_+, T_-, F\right)$, where T_+ and T_- are disjoint dense sets of *anchors* and *tombstones* with a total order and homomorphism F: $T_+ \leftrightarrow T_-$. We will be considering a rational tag space $R = \left(\Re_+, \Re_-, -\right)$.

Dense list L on a tag space T is a generic set I of items $i = (t, v)$ where $i \in I$, $t \in T$. We define list tag sets as $T(L) \equiv \{t \mid \exists\, (t, v) \in I\}$, $T_+(L) = T(L) \cap T_+$, $T_-(L) = T(L) \cap T_-$. Dense list item index $\#i$ of item $i = (t, v)$ is defined as a rank position of t within anchor tag list $T_+(L)$.

Replicated dense list (RDL) is a dense list where item set I is a grow-only set of last-write-wins tag registers with associated item values. In other words, RDL is a pure [1] op-based CRDT with a low-level operational basis of spawning a list item and modifying its tag value. It is worth noticing that RDL does not require a specific clock type, so LWW registers may even use a real time clock. Let us define list interface operations on a RDL.

Operation with intention of inserting value v after element $i' = (t', v')$ which is locally followed by $i'' = (t'', v'')$ is expressed as inserting element $i = (K(t', t''), v)$ to a grow-only set I, where K is an arbitrary kernel operator on a tag space T.

Operation with intention of deleting item $i = (t, v)$ is expressed as applying homomorphism F on an anchor, which is in practice setting a negative bit in the item tag register. Deleted items correspond to tombstone tags and are not visible in the list. Garbage collection of tombstones may be implemented using an identification of causally stable [1] item deletions. However, it seems difficult to physically remove items without vector clock timestamps associated with operations which inevitably lead to metadata overhead problems and linear size of each operation.

Operation with intention of moving a single item $i = (t, v)$ from its current position to the position between items $i' = (t', v')$ and $i'' = (t'', v'')$ is expressed as changing t to $K(t', t'')$, similar to insertion operation. Naive range extension of this operation can be a composition of single move operations.

Though RDL item set may be implemented in various ways, a self-balancing Cartesian tree set provides implicit key indexing for $O(log\,n)$ and a way to move ranges of RDL items for $O(log\,n)$ complexity. Formally, a *Cartesian tree* is a binary tree of *<x, y>* tuples with a binary search tree property by x and a max heap property by y. *Treap* (tree + heap) is a self-balanced binary search tree which is constructed as a Cartesian tree with randomly distributed y. Insertion and deletion operations are compound of split and merge operations and their amortized complexity is O(log n).

RDL facilitates user-driven intention conflict resolution by modifying item anchors a posteriori. Thus, explicit order of simultaneous insertion operations may be restored by changing anchor value on a desired group of items. Interleaving problems may be resolved by choosing a kernel operator tolerant to range insertions.

Replicated object models may require additional constraints on collection cardinality, i.e. minimum and maximum allowed list size. In general, it is impossible to satisfy these rules within optimistically replicated collaboration, because concurrent insertion or deletion operations on various items lead to cardinality overflow or underflow. We show that RDL is able to preserve size constraints by weakening intention preservation property while preserving eventual consistency.

Let us consider a RDL with inclusive cardinality range constraint $[a, b]$ with initially spawned $s_{initial} \geq s_1$ elements. Then we can redefine dense list item index $\#i$ of item $i = (t, v)$ as a rank position of t within $min_b(T_+(L)) \cup max_{max(0,\, a - |T_+(L)|)}(T_-(L))$. Informally, when maximum cardinality constraint is violated, odd elements are omitted while being persisted in the item set. On the contrary, when there is cardinality underflow, tombstone elements are resurrected on their previous positions until cardinality constraint is met. Tombstone elements which precede a greatest elements can be safely garbage collected.

RDL can be compared with existing replicated list data types. FugueMax [2] operates on a similar tag mechanism to reduce interleavings. Treedoc implements dense order within a tree, but requires rebalancing which is nontrivial. RGA uses a similar linked list idea with logical timestamps as an improvement over WOOT.

Conclusion. Thus, a replicated list data type with total order on a dense space with clear semantics is presented. It is simple and does not necessarily require strong causality preservation mechanisms by introducing pure commutativity of operations. Further research involves experimental study of RDL performance, design of kernel operators which minimize interleavings and object framework design based on RDL.

References:

1. Baquero, C., Almeida, P. S., & Shoker, A. (2017). Pure operation-based replicated data types. *arXiv preprint arXiv:1710.04469*.
2. Weidner, M., & Kleppmann, M. (2023). The Art of the Fugue: Minimizing Interleaving in Collaborative Text Editing. *arXiv preprint arXiv:2305.00583*.