CUERA: Performance evaluation of a generic data- and undo/redo-consistency framework

Daniel Stolzenberg, Erika Müller
Institute of Communications Engineering, University of Rostock, Germany
{daniel.stolzenberg, erika.mueller}@uni-rostock.de

***Abstract*—In this paper, a generic data- and undo/redo-consistency framework for realtime interactive collaboration applications is evaluated in terms of performance and scalability. Since data is represented based on Entity-Relationship-Models, the framework is applicable in a wide range of domains. It provides a set of state-focusing operations, that is able to abstract arbitrary data interfaces. In addition, non-reflexive meta-operations allow to undo/redo any user action. Consistency is maintained by forcing concurrent operations to commute by transitive precedence rules. Both convergence and undo/redo control are mirrored at runtime to provide maximal performance and scalability regarding session duration, data and number of participants. These claims are successfully verified in this paper by analyzing average execution times under a realtime collaboration workload simulated in a turn-based system.**

***Index Terms*—computer supported collaborative work (CSCW), optimistic replication, eventual consistency, commutative replicated data type (CRDT), any-undo/redo, performance evaluation**

I. Introduction

Realtime interactive collaboration applications (RTICAs) provide a shared context, where a group of users can interact simultaneously albeit being spatially separated. To achieve high responsiveness and fluid interactivity, data representing the context has to be *optimistically replicated*, requiring to actively ensure *eventual consistency* [1]. In order to be useful, consistency solutions do not only have to be correct, but ought to perform as well. In particular, performance of state retrieval and user action execution must not degrade when the collaboration setting is scaled.

But many existing consistency solutions actually neither perform nor scale well. In addition, most systems are limited to linear data structures or do not provide necessary undo/redo features. Hence, the authors recently proposed the generic *collaborative undoable entity-relationship-actions* or CUERA framework [2]. But up to now, performance and scalability have only been examined regarding computational complexity.

Consequently, this paper aims at proving CUERA's actual performance by measuring execution times. But instead of observing real-world collaboration sessions, workloads are generated in a turn-based simulation. This approach allows to specifically and separately address parameters that affect performance and underlie scaling. Particularly, the effects of scaling session duration, context data volume and the number of users are evaluated. In addition, performance and scalability of several distributed clock techniques is evaluated, to identify a suitable choice for CUERA implementations.

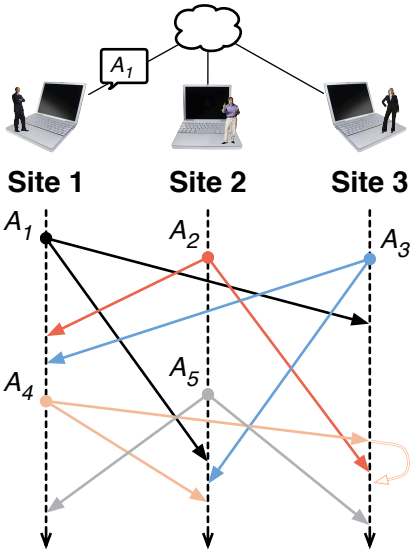

Fig. 1. Collaboration Network (Time-Space-Diagram)

In sections II and III, the consistency problem and the fundamental design of the framework is presented. Then performance and scalability of CUERA is evaluated in section IV. Finally, section V points at some related research before section VI summarizes the paper.

II. RTICA Consistency

A. Optimistic Replication & Eventual Consistency

RTICAs operate in a *collaboration network* composed of a set of *sites* S_i. The number of sites dynamically changes, when users join or leave the collaboration session. Every participating site maintains a local *replica* of the shared context data for fast access. Any *action* A_j of a user is immediately executed at the local replica, before it is *asynchronously* send over the network. (Figure 1 gives an example of action propagation and execution timings in a simple collaboration network of three sites.) This *optimistic replication* enables high responsiveness and fluid interactivity, since users are never blocked from initiating actions and effects are immediately visible at the local site. However, replicas temporarily diverge to inconsistent states, while actions are propagated to remote sites. But convergence to *eventual consistency* is assumed, when every action performed in the network has been received and executed at all sites – including late joining sites.

COLLABORATECOM 2012, October 14-17, Pittsburgh, United States

DOI 10.4108/icst.collaboratecom.2012.250374

B. Causality, Convergence & Intention Preservation

Yet optimistic replication poses significant challenges in terms of consistency and user expectations, which can be described using the CCI model by Sun [3].

- Users expect that semantic cause-effect-relations between actions are preserved. But in general, networks feature varying propagation times, that can distort order relations. So to avoid user confusion, RTICAs themselves have to force *causality preservation* in place of the network.
- Moreover, eventual consistency is far from self-evident: Because background propagation does not block users from contributing actions concurrently, execution orders can vary among sites (Figure 1). And since actions do not inherently commute in general [3], different execution sequences lead to inconsistent replicas. Thus, RTICAs must take active measures for replica *convergence*.
- In addition, users expect that the effects, they intended when performing an action, are reflected in the consistent replica state. But in face of conflicting concurrent actions, perfect *intention preservation* is impossible. So instead, RTICAs have to provide *automatic conflict resolution*.

In summary, the consistency problem is largely evoked by properties of the shared data and the actions performed upon it, non-commutativity and conflicting intentions of concurrent actions in particular.

C. Replication Transfer Mode & Semantic Resolution

The previous section suggested, that the most important design decision of optimistically replicated systems is the definition of actions propagated in the network. According to Saito et al. [1], two principle replication transfer modes can be distinguished for the same type of data. *State transfer* captures any local action by simply propagating the entire resulting state. In contrast, *operation transfer* reflects only the state differences by explicitly propagating the types of the performed operations along with their parameters. Thus, operation transfer can reduce network traffic. But it is more complicated, because sites have to maintain a history of operations and agree on the result of its execution.

While both state and operation transfer are equally suited to achieve convergence, they significantly differ in respect to intention preservation. With state transfer, all concurrent actions are in conflict, whereas operation transfer can make use of semantical information to detect and even resolve conflicts to preserve user intentions.

Consequently, the intended framework is based on operation transfer. But still, the level of semantic resolution of operation transfer can be varied in the design. In fact, semantics of user interaction is an emergent phenomenon that is difficult to grasp: The cognitive intention of a user interaction is not necessarily reflected by the expression via a graphical user interface and the underlying software abstractions. And though it is desirable to achieve the highest level of intention preservation possible, practical reasons argue against full scale semantic resolution for non-trivial RTICAs:

- Typically many different data types are needed.
- These data types are often very application specific.
- Most data types feature a comprehensive operation set.
- Semantics of operations are rather sophisticated.
- Concurrency further complicates operation semantics.
- Data types are subject to changes during development.

Complexity of convergence and intention preservation grows quadratically with the number of operation types. So assuring consistency can be a downright daunting task when semantic resolution is taken to the limit. Thus, is seems practical to strike a balance between simplicity and intention preservation. This applies all the more to the intended generic consistency solution, as it is impossible to directly generalize the operation set of arbitrary data types.

III. CUERA Framework

A. Data Representation

In order to develop a generic consistency solution, a capable data representation has to be identified, that is able to serve common data needs of RTICAs. Therefore, one of the most universal and promising abstractions of data is considered: *Entity-Relationship-Models* (ERMs). The vast majority of software projects use ERMs to determine their static data structure. Hence, ERMs are applicable in a wide range of domains.

Several *entity-types* (T_{ent}) classify the constituting elements of the domain, while *attribute-types* (T_{att}) reflect relevant element properties. Similarly, connections and dependencies are formalized by *relationship-types* (T_{rel}) between their corresponding entity-types. *Cardinalities* for both directions specify the number of relations, an element can be part of for the same type. At least, singular cardinalities ("To-One") are distinguished from multiple cardinalities ("To-Many"). Some advanced features of ERMs are beyond the scope of this paper, but will be examined in future investigations: Inherently *ordered relationships*, *specialization/generalization* of entity-types and *aggregation/composition* of entities.

In object-oriented systems, these abstract descriptions are translated into class hierarchies, which are instantiated into a *model object graph* at runtime. In particular, bidirectional relationship-types in the ERM are represented by inverse object reference pairs in the model graph. Based on the private state represented by objects, member variables and references, the methods in the *model interface* expose functionality and behavior to the application.

In order to provide this common data scheme for RTICAs, a *replicated entity relationship graph* (RERG) data type is introduced, that literally realizes an ERM: The graph's vertices are incarnations of entity-types defined therein and contain their respective attributes, while its bidirectional edges are incarnations of relationship-types. According to the previous section, operation-types are intentionally constrained to simple semantics, primarily focusing the private model state. Since user intentions commonly refer to non-atomic tasks, a closed action is composed of an ordered set of operations of the following *operation-types*:

- $EntityOperation\ (T_{ent}, ID_{ent})$
  - $Create$
  - $Destroy$
- $AttributeOperation\ (T_{att}, ID_{ent}, [Parameters])$
  - $Set\ (\ldots, Value)$
- $RelationshipOperation\ (T_{rel}, ID_{src}, ID_{dst})$
  - Add
  - $Remove$

As the operation signatures suggest, entities are identified by unique and unambiguous ID_{ent} generated at the initiating site. Moreover, entities are *non-reviving*: Once destroyed, they cannot be created again. Integrity of this graph is assured by *invalidating* relationships, that feature destroyed entities. In a very similar way, conformance with the ERM is enforced: Any existing relationship at a singular cardinality is *replaced*, when a new relationship of the same type is added to the entity.

When facing concurrency, it is substantially easier to obey these criteria for valid relationships than controlling integrity of the implicit reference representation commonly used in model object graphs. Still, the reference representation has to be translated to and from the relationship representation to provide a familiar programming model. Consequently, using the RERG data type does not involve a conceptual change in comparison to using native constructs: In both cases, the model interface and the semantics of its operations are build upon a syntax of private state.

B. Consistency Maintenance

As stated above, convergence is threatened by any non-commutativity of concurrent operations. So all combinations of operation-types are analyzed for inherent commutativity. In particular, a naive RERG implementation is assumed, that respects the invalidation and replacement criteria for relationships mentioned above. In general, operations, that do not overlap in the RERG, always commute. Thus analysis takes place within the same region of the graph. Due to unique and non-reviving entities, strictly no operation can involve an entity created concurrently. From the remaining combinations, only three do not inherently commute:

- Concurrently setting the same attribute of the same entity to different values does not commute, since the attribute always reflects the last executed operation.
- Concurrently adding and removing the same relationship does not commute, because the relationship reflects the last executed operation.
- Concurrently adding different relationships to the same singular cardinality does not commute, because the last executed addition replaces all previous ones.

For replicas to converge, these critical combinations have to be forced to commute. The *Operation Commutativity by Precedence Transitivity* (OCbyPT) concept introduced by Roh et al. [4] provides a theoretically proved way to do this. Basically, all actions are brought into a linear *precedence order*, mimicking a single stream of events in time. This allows to conceptually treat concurrent actions in the same way, as if they were truly subsequent in time. So the consistent state is defined to be the result of executing actions in this linear order. Any other valid execution order has to exactly reveal this state by commuting actions *as if* they were executed in precedence order. The correct effect of any action can be inferred based on the precedence relations (PR) to actions already executed.

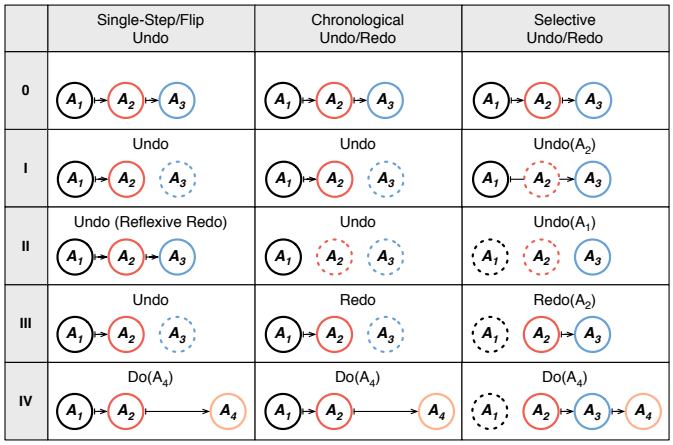

Fig. 2. Important Undo/Redo Modes

This approach *does not restrict execution to precedence order*. In practice, the precedence order is derived from timestamps of the *logical clock* commonly used for causality preservation. Still, *convergence is not dependent on causality preservation*, although precedence relations should be defined to generalize causal relations.

Because of state-focusing operation semantics, Thomas' write rule (LWW - Last Writer Wins) can be leveraged to force commutativity and resolve conflicts of RERG operations. Only one operation is *effective* for each element in the graph:

- The set operation with the highest precedence is effective for an attribute and defines its value.
- The addition or removal with the highest precedence is effective for a relationship and defines its existence.
- The relationship with the highest addition precedence is effective for a singular cardinality – regardless of being removed, invalidated or replaced (in "One-To-One").

C. Collaborative Undo/Redo

In addition to directly manipulating data, users must be able to recover from erroneous manipulations by invoking undo/redo features. Figure 2 visualizes several undo/redo modes, that have been proposed to help users selecting the action(s), they need to undo/redo to achieve the intended error recovery effect. To support these modes, collaborative undo/redo features have to be able to toggle the *meta-state* of *any action* initiated by *any user*. In the background, an algorithm has to produce the desired data state, that reflects all actions' meta-states. The convergence control technique introduced in the previous section already implies the approach to a consistent collaborative undo/redo: The result is produced by executing only actions in precedence order, that currently are *not undone*.

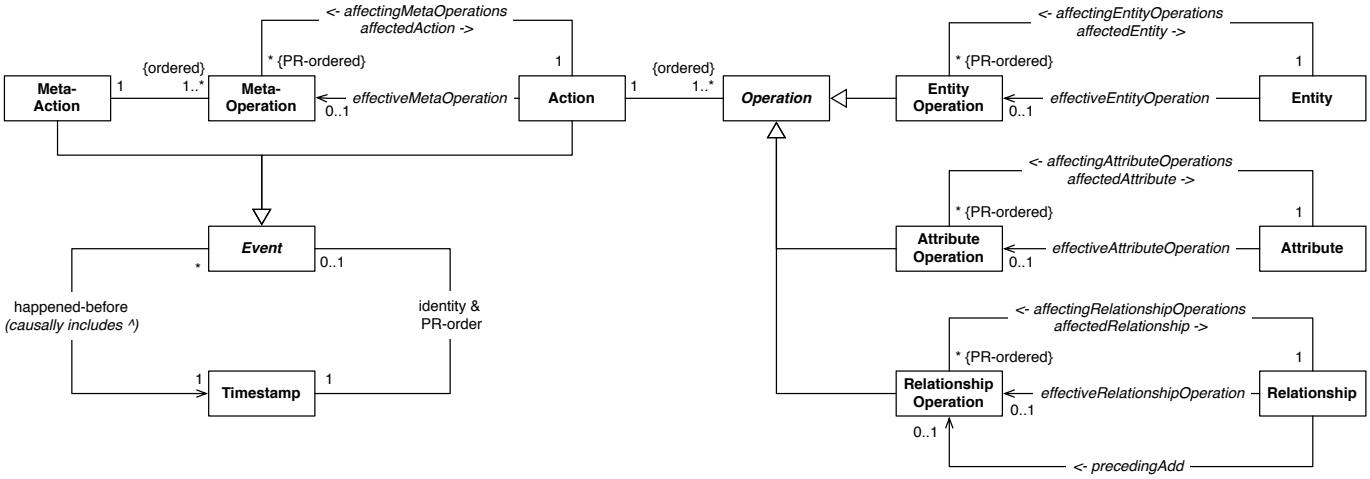

Fig. 3. The CUERA Framework

In order to align with the granularity of user intentions, undo/redo features have to target operations at the action level. Equivalently, expressing an error recovery user intention in general requires to undo/redo more than one original action as well. Hence, *meta-actions* are introduced, which are composed of an ordered set of *meta-operations*:

$MetaOperation(ID_{act})$

- $UndoAction$
- $RedoAction$

Again, the signatures state that actions are identified by ID_{act} generated at the initiating sites. Moreover, the non-reflexive definition of undo/redo is revealed: To undo an undo meta-operation, the original action has to be redone. This both simplifies the structure that determines the meta-state of actions, as well as the user-facing undo/redo feature.

To force commutativity of concurrently undoing and redoing the same action, meta-actions and meta-operations take part in the same LWW scheme used by the RERG elements: An action‘s meta-operation with the highest precedence is effective and defines its meta-state.

D. Model State Retrieval

The previous sections introduced all CUERA concepts used to represent data and user actions. At runtime, both the LWW convergence and meta-state control are exactly mirrored by an efficient *scope organization* as depicted in Figure 3:

Each element references its affecting operations and sorts them by precedence. In addition, the operation with the highest precedence and active meta-state is explicitly tracked as the effective operation. The same organization stretches out to actions and their affecting meta-operations. Moreover, relationships track their topmost preceding Add operation. This allows to track the effective relationship at singular cardinalities. Then, both invalidation by destroyed entities and replacement at singular cardinalities can be taken into account, when retrieving relationship states. So this scope organization mostly reduces state retrieval to single operation inspection:

- An entity exists, if its $Create$ operation is effective.
- If the surrounding entity is existent, all the contained attributes are defined by their effective Set operations.
- To provide a familiar model interface, relationships have to be translated into a reference representation: If the entity at the origin of the retrieval exists, connected relationships of the type in question are inspected (for a singular cardinality at the origin of retrieval only the effective relationship is considered):
  - The existence of the relationship itself is checked: Its effective operation must be of Add type.
  - In case of singular target cardinality, it is examined if the relationship is the effective one.
  - The target entity is tested for existence.

  If all conditions are met, a reference to the target entity is included in the result.
- An action is only active regarding meta-state, if no $UndoAction$ operation is in effect.

E. Local & Remote Action Execution

Local interaction with the public model interface has to be recorded into a (meta-)action object. To assure that a local (meta-)action always precedes all existing (meta-)actions, first the timestamps are attached to the action with the clock being incremented in between. Then the action components can be recorded:

- $Create/Destroy$ operations for entities. Moreover, a fresh ID_{ent} is generated for created entities.
- Set operations for attributes.
- Again, the reference representation has to be translated into relationships: From the ID_{ent} of each reference the $Add/Remove$ operation can be build. In addition, the relationship element has to be created if necessary.
- $Undo/RedoAction$ meta-operation for actions.

To be reflected in the private model state, each recorded (meta-)operation must be inserted into its affected elements‘ scope. Because of its most recent timestamp, it immediately

becomes the effective operation, which in turn triggers a change notification: For example, when an $UndoAction$ meta-operation becomes effective on its affected action, the action changes to inactive meta-state. So it notifies all its contained operations, which notify their affected elements of becoming inactive. This process goes down the chain of dependent elements, updating the scope of each affected element.

Finally, the local (meta-)action is encoded and propagated in the collaboration network. After reception at a remote site, the (meta-)action is decoded before it can be executed. But causality constrains the execution of remote (meta-)actions: If happened-before relations were violated, the execution must be delayed until every happened-before is received and executed.

When it is causally ready for execution, the remote (meta-) action is integrated in the private model state representation. At first, affected elements must be retrieved or created, if they do not exist yet. Then, the same as for local execution, each (meta-) operation is inserted into the scope of its affected element: Precedence of the received (meta-)operation is compared to all existing ones in this scope in top-down progression. If it becomes effective and the resulting state changes, a change notification is triggered.

IV. Evaluation

A. Evaluation Design

While the time complexity analysis, that is included in the CUERA introduction [2], provided a valuable starting point to estimate the scalability characteristics, it is not sufficient to prove its actual performance. This can only be fulfilled by evaluating in-depth measurements of the framework in action, which is the main contribution of this paper. At first glance, this seems to inevitably require inspection of collaborations of human users. Though it is unquestionable, that such evaluation can be worthwhile, indeed it is more appropriate to analyze the characteristics of collaboration sessions than those of the software system. Instead, necessary collaboration workloads can be simulated computationally, which has some important advantages: Besides easily producing results, the relevant parameters potentially influencing performance and scalability can be adjusted precisely and studied independently:

- Session Duration (Number of Actions)
- Data Volume (Amount of Entities & Relationships)
- Network Size (Number of Participants)
- Undo/Redo (Percentage of Meta-Actions)
- Area-Of-Interest (Scope Concurrency)

According to Roh et al. [4], collaboration workloads can be modelled using a turn-based simulation system: In each turn, every participating site randomly either generates a local action or executes a remote action. At the end of each turn, a simple propagation system inserts remote actions into a queue maintained for each site. Thus, the duration of remote action propagation is simulated by the residence time in the queue. Since all sites have to generate an action in the first turn, the queues get filled immediately. In the course of the simulation, the length of the queues fluctuates due to random generation of actions. So, a realistic variation of the propagation interval can be assumed. Still, causality is preserved, if the queue is operated in strict FIFO order.

This simulation approach is leveraged for performance evaluations by recording execution times in different setups. Within the following test series, a basic driving model for the simulation network is employed: In each run, a constant number of sites (s) produce the same number of actions (a), uniformly distributed over turns. For each action, a constant amount of operations ($o = 4$) is generated. A fairly simple data model is used, that connects two entity-types in a relationship-type with "One-To-Many" cardinalities. The entity-types both feature one attribute-type only. Despite being simple, this data model serves the purpose of this evaluation well.

In general, every setup is executed in 10 simulation runs, to provide better statistics. Furthermore, only computations related to the framework are observed, while any support tasks required for the simulation itself are filtered out. Particularly, times are logged separately for all operation-types as well as actions, both split into local and remote execution. Precisely, action execution time measurement does not include execution of the contained operations, in order to clearly separate the clock related aspects of causality preservation, timestamp generation and inclusion. In addition to performance analysis, convergence of replicas is verified at the end of the run.

Simulations were conducted on a dual-core 2.66 GHz Intel Xeon processor with 6 GB of DDR-2 RAM running Mac OS 10.6.8. The following results are based on a very basic in-memory implementation of the CUERA framework in Objective-C. While it is functionally complete and serves the requirements of this evaluation, it is not a production system and there is a lot of potential for optimization. For example, destroyed entities, invalid relationships as well as obsolete actions and operations might be purged from memory and lazily fetched from storage if needed to support unconstrained undo/redo of any action.

Besides evaluating the CUERA framework itself, this paper also addresses the performance of distributed clock techniques. To determine the raw performance of CUERA, its acausal-convergence-property (Section III) is exploited: A simple loosely synchronized wall clock [5] can be used, that is not sufficient for causality preservation, but still allows to derive a precedence order. In relation to this and the standard vector clocks [6][7], the performance of *Interval Tree Clocks* (ITC) is compared, that were introduced by Almeida et al. [8]. ITC was chosen, because its high dynamism and decentralized operation are interesting features for RTICAs. Moreover, ITC's performance has not yet been evaluated.

As no precedence relation has been specified for ITC, it is defined similar to the one for vector clocks [4]: First, timestamps are ordered by comparing the integral values of the event function component. For timestamps with identical event integrals, its first moment in relation to zero is used as secondary order relation. This definition guarantees the total order of actions required by the CUERA frameworks consistency maintenance technique (OCbyPT and LWW).

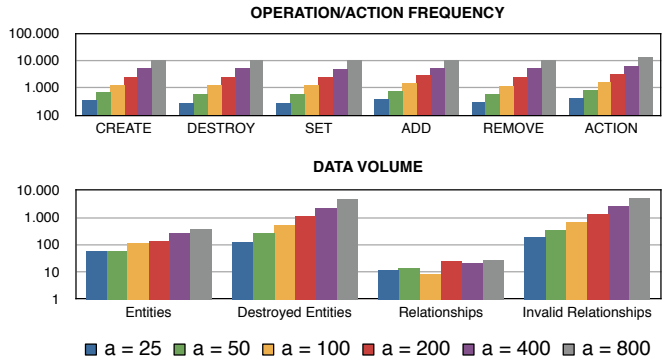

Fig. 4. Duration & Data Effect

B. Duration & Data Effect

With the first series of tests, the baseline performance of the CUERA implementation in a collaboration network of constant size ($s = 8$) is targeted: Both operation-types and parameters (affected entities, attributes and relationships) are chosen randomly, while undo/redo features remain untouched for this fundamental performance evaluation. In particular, it is tested if performance remains stable after prolonged periods of collaboration. Besides its increasing number of actions and operations, extended collaboration duration with randomly chosen operations is characterized by constantly growing data volumes. Therefore this test series will also highlight the capabilities and limits of the simulation routine.

With each test, the number of actions per site is increased in an exponential sequence ($a = 25, 50, 100, 200, 400, 800$). To verify the random choice of operations, the number of actually performed operations and actions is depicted in the upper half of Figure 4. The lower half reflects the effect on data volume: Its exponential growth is clearly visible, especially tombstones (destroyed entities and invalid relationships) are making up the major part. So in fact, two of the scaling factors to be analyzed are comprised in this series of tests: Duration & Data. Therefore the averaged execution times of 10 runs depicted in Figure 5 indeed represent the baseline performance of the CUERA implementation.

The overall level of performance is in the range of tens of microseconds, with a slight logarithmic increase across the board. But in detail, local operation execution faces a major rise between $a = 200$ and $a = 800$, particularly the creation of entities. This does not fit well into the expectations, since in theory, local operations are not susceptible to scaling. It can be assumed, that this in fact highlights the limitations of the CUERA, simulation and data acquisition implementations in terms of memory management.

Therefore, additional detail about the statistics is provided by the frequency distributions in the histograms of Figure 6. For local execution of $Create$, the initial quasi mono-modal characteristics split into bi-modal distributions over the course of the duration. The histogram supports the conclusion, that the second mode emerges in the later part of the simulated session.

Fig. 5. Duration & Data Effect On Operation Execution

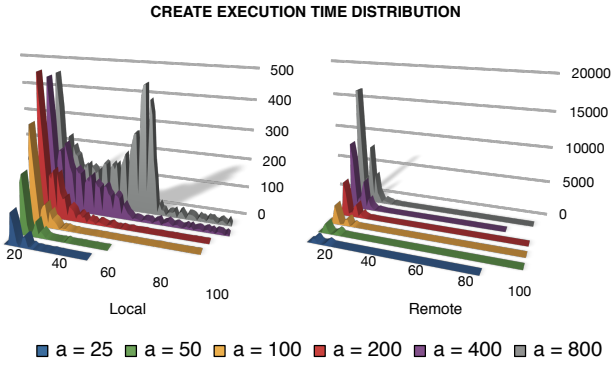

Fig. 6. Memory Paging Effect On Local Execution

A distinct point of separation can be estimated at a duration parameter of $a = 200$. Contrastingly, remote execution shows a similar distribution from the beginning, that remains stable. Additional data beyond the scope of this discussion suggests, that this effect is part of the intense memory paging needed within the operation generation part of the simulation system. This aligns with the fact, that $Create$ operations involve significantly more memory allocation, because the internal infrastructure for attributes and relationships is instantiated.

In summary, operation execution performance is provably stable even in extended collaboration sessions. The result for remote operations is remarkable, since they are particularly decisive for the overall performance of RTICAs [4].

Action execution performance has been measured as well, to determine performance of the distributed clock mechanisms. In Figure 7, the paging effect on local execution is visible again. All clocks feature a very moderate logarithmic local and almost constant remote execution. Most importantly the simple wall clock outperforms ITC by a factor of 10. Thus, performance reflects the increasing complexity of the tested techniques. Still, all of them perform acceptably well when compared to the execution of the contained operations.

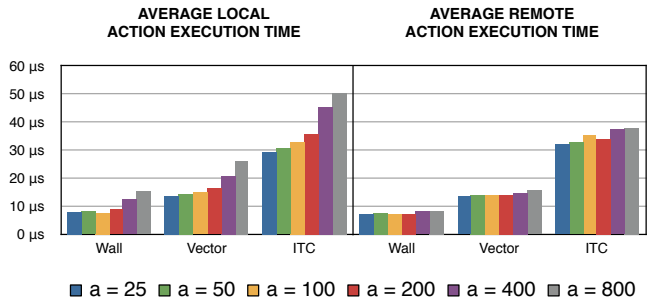

Fig. 7. Duration & Data Effect On Action Execution

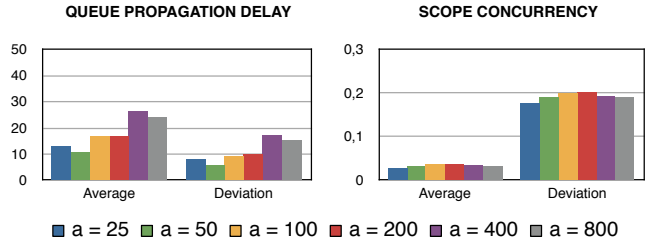

Fig. 8. Duration & Data Effect: Propagation Delay and Scope Concurrency

Finally, the propagation simulation using action queues is verified: Figure 8 presents average and standard deviation of action propagation delay measured in turns. It shows, that the queue propagation scheme induces a reasonable amount of concurrency in terms of delay. Still, the random operation generation does not result in a significant concurrency in terms of operation scope: The area-of-interest (AOI) of the simulated "users" barely overlap. The metric used in Figure 8 is gripped from the scope insertion process and reflects the number of concurrent operations already present in the same scope, that possess higher precedence. Therefore, this baseline performance evaluation can only provide a starting point.

C. Network Effect

But before this matter is explored in depth, the effect of scaling the network in size is examined. To avoid the memory management concerns and still provide reasonable session duration, the $a = 200, s = 8$ setup is chosen to base the variation upon. All other simulation settings are equal to the first series of tests to maximize comparability. Once again, network sizes are modulated in a exponential fashion, so scalability effects become visible ($s = 2, 4, 8, 16, 32, 64$). In order to keep the overall number of actions ($A = s \times a = 1600$) constant, the amount of actions per site is varied accordingly.

That this strategy worked out, can be read from Figure 9: Both the number of performed operations and actions, as well as the amount of data is approximately constant over the series of tests. Thus, it can be assumed, that any duration and data effect is eliminated and solely network scaling is in effect.

Again execution times of operations underlie very modest logarithmic progression as depicted in Figure 10. These results confirm the assumption, that the memory paging effect visible above could be avoided. Hence, operation execution is proved

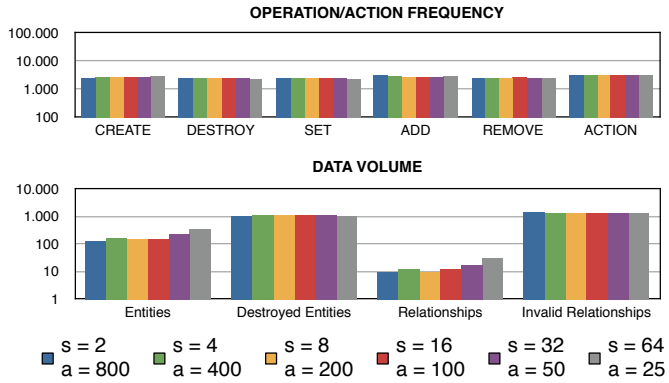

Fig. 9. Network Effect

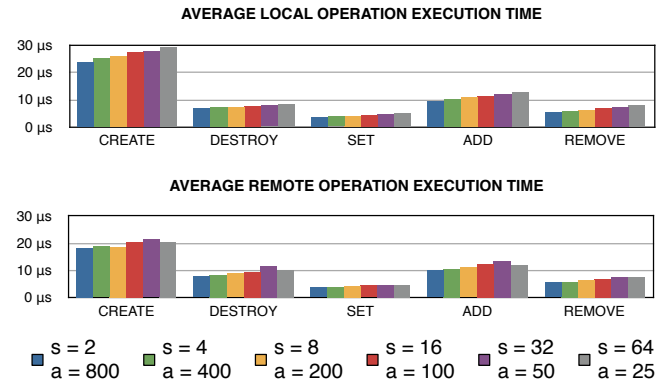

Fig. 10. Network Effect On Operation Execution

to scale well even in terms of network size, with average times in the range of tens of microseconds. Equal to the duration and data effect results in Figure 5, execution of $Create$ operations is significantly slower than the other operations, because it involves creating the internal infrastructure of attributes and relationships. In a similar way, Add operations are executed slightly slower, because of the overhead of controlling relationship replacement at singular cardinalities. In agreement with this, Set operations feature the fastest execution, since they do not involve any side effects.

The most severe insight of this investigation of the network effect is related to action execution performance. According to Figure 11, both causality-preserving clocks linearly degrade with network size, while the precedence-only wall technique features constant performance. This highlights the importance of CUERA's ability to converge even in face of acausally executed actions: It allows perfectly scaling massive user collaboration, either by relying on causal propagation within the network itself or by sacrificing causality preservation.

Still, Vector clocks seriously outperform ITC concerning the slope of this degradation. Thus, the advantages of ITC's dynamism and decentralization come at a significant cost in scalability. But still it is remarkable, that the accumulated execution time of an action and its contained operations in a network of 64 users are well beyond the limit of 50 ms commonly denoted as acceptable from a user perspective.

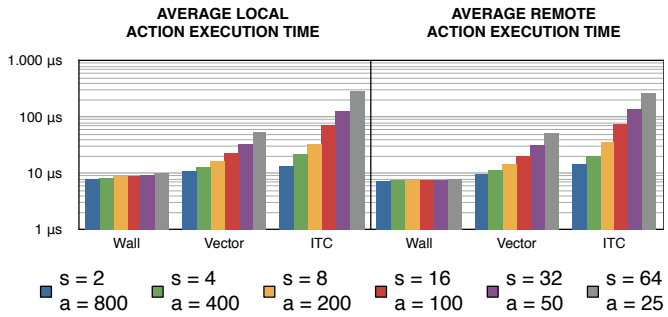

Fig. 11. Network Effect On Action Execution

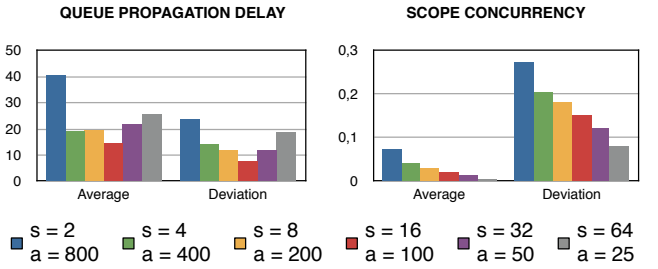

Fig. 12. Network Effect: Propagation Delay and Scope Concurrency

Finally, Figure 12 increases confidence in the metric used for AOI measurement based on scope insertion: The already low level of concurrent operations, that actually require replica convergence control, steadily declines with network size under the regime of random operation and AOI generation.

D. Undo/Redo Effect

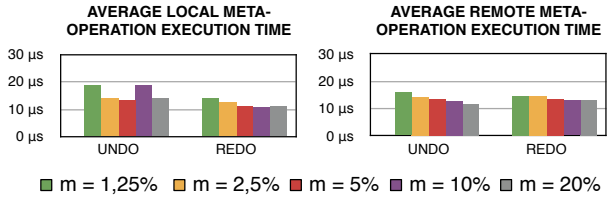

Fig. 13. Undo Effect On Meta-Operation Execution

To include undo/redo features, another series of tests has been conducted based on the $a = 200, s = 8$ setup, that varies the percentage of meta-actions m of all generated actions in a wide range. According to the results in Figure 13 meta-operations perform in similar ways as data-targeting ones. The supposed effect of increased execution time due to change notification down the chain of dependent elements is comparable to the aforementioned $Create$ overhead. In summary, RTICAs are able to feature sophisticated and still performant undo/redo facilities.

E. Area-Of-Interest Effect

Finally, the last series of tests addresses the remaining issue of operation scope concurrency. A different operation generation scheme is employed to simulate a reliably focused

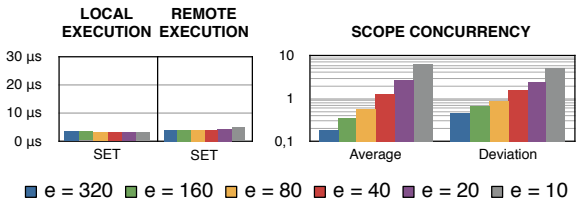

Fig. 14. AOI Effect

area-of-interest shared by all virtual ”users”: Within the first turns of each run, a definite number of entities ($E = 320$) is created. After this initial setup, the number of entities remains constant and no relationships are added. Instead, all sites exclusively produce Set operations targeting the single attribute-type of the existing entities. Then, the area-of-interest is varied by restricting access to a definite subset of entities (e). The rationale behind this generation scheme arises from the conceptual and behavioral similarities of all operation-types. Hence, the Set operations are considered prototypical and the observed performance characterstics can be generalized.

Figure 14 presents the results of this evaluation. First and foremost, operation execution time remains almost constant. And correctness of this result is confirmed by the exponential trend of the scope concurrency metric: It clearly indicates the massive amount of concurrent operations targeting the same attribute of the same entity. In fact, this density of scope concurrency will barely be present in any real-world collaboration session of human users. Therefore it is safe to assume, that performance of the CUERA framework is suitable for most collaborative settings.

In summary, the evaluation results support the scalability and performance claims: The CUERA framework performs in the range of tens of microseconds and scales well in terms of collaboration duration, data and scope concurrency. Only the linearly degrading action execution constraints its scalability regarding the network size, but is related to the clock mechanism not the framework itself. Since convergence is assured even with violated causality, future investigations will try to improve action execution scalability by relaxing causality preservation.

V. Related Work

The previous sections are related to the following research relevant within the scope of this paper. Most of it is rooted in the field of collaborative text editing systems, featuring $insert$ and $delete$ operations on a linear data type. *Operational Transformation* (OT) [9] dominates research in this area and has been extended to a wide range of domains. Conceptually, concurrent operations are transformed against each other to let different execution orders converge. In fact, elaborating this basic idea has led to a plethora of algorithms [10], most of them failing in special situations called ”puzzles” [11]. Algorithms known to be correct are very complex to design in practice and computationally expensive when implemented [12], [13], so performance and scalability are a serious issue.

In reaction, some recent research targets convergence by commutativity of operations without transformation. In particular, Shapiro et al. [14] and Roh et al. [4] developed commutatively replicated variants of important abstract data types (Counters, Registers, Sets, Arrays and even Graphs). But none of them offers the spectrum of capabilities targeted in this paper and especially undo/redo has been neglected.

Though many OT based systems support undo/redo of any action [15][16], non-OT research on this matter is rare. Most of it is related to text editing in P2P-Wikis [17] [18]: Besides the limitation to sequential data, their counter-based undo/redo results in a user experience not suitable for realtime collaboration. The latter also holds for the XML tree editing system in [5], but XML nodes and attributes at least partly resemble the model object graph concepts identified in Section III. Hence, some of its approaches did inspire the development of the intended CUERA framework: The history organization of operations in their respective scope and the LWW technique for attribute convergence. But most aspects of a generic data and undo/redo consistency solution are still unresolved and motivate this paper.

Actual performance evaluations beyond complexity analysis are rare in research literature. Most notably, Roh et al. [4] presented the idea of simulating collaboration workloads using a turn-based system. Their work on evaluating the RGA (replicated growable array) data type has greatly influenced this paper. In contrast to CUERA, the RGA type does not provide undo/redo features and therefore allows to purge obsolesced data elements. Most recently, Ahmed-Nacer et al. [19] conducted a comparison evaluation of several consistency solutions for collaborative text editing. The study actually monitored collaborating human users and acquired logs of the sessions. Later on, the logs were replayed against a variety of algorithms to determine their performance. In particular, it was confirmed, that post-OT systems can outperform the top OT algorithms. Neither of those investigations had to deal with the AOI variation, because text editing as well as array operations take place in a single scope.

VI. Conclusion

This paper evaluated performance of the CUERA framework, that was recently proposed by the authors. Its foundation is a capable replicated data representation based on ERMs. In order to allow abstraction of arbitrary data interfaces, a state-focusing approach is taken for the definition of the operation set. In face of concurrent actions, replica convergence is assured using a LWW rule backed by the proven OCbyPT technique, even in face of acausal action execution. Powerful undo/redo features allow recovery of any mistaken actions.

The core part of this paper aimed at proving scalability and performance of the framework. In particular, a turn-based simulation was used to generate intense collaboration workloads. By measuring execution times in different settings, scalability regarding collaboration duration, number of sites, volume of shared data and degree of concurrency has been studied. The results attested, that the framework performs tasks in the range of tens of microseconds across the bank. Moreover, performance remains stable even when important parameters are scaled in a wide range. Only if the framework has to preserve causality in place of the propagation network, performance of the distributed clocks linearly degrades with the number of users. Still performance has been found to be acceptable for at least up to 64 participating users.

Future research will enhance the data representation features by adding support for ordered "To-Many" relationships. In addition, removal of obsolete action and data elements will be examined to reduce memory consumption and still support full unlimited undo/redo.

References

[1] Y. Saito and M. Shapiro, "Optimistic replication," *ACM Computing Surveys (CSUR)*, 2005.

[2] D. Stolzenberg and E. Müller, "CUERA: A Generic Data- and Undo/Redo-Consistency Framework for Realtime Interactive Collaboration Applications," *Proc. of the 18th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'12)*, 2012.

[3] C. Sun, Y. Zhang, X. Jia, and Y. Yang, "A generic operation transformation scheme for consistency maintenance in real-time cooperative editing systems," *GROUP '97: Proceedings of the international ACM SIGGROUP conference on supporting group work*, 1997.

[4] H.-G. Roh, M. Jeon, J.-S. Kim, and J. Lee, "Replicated abstract data types: Building blocks for collaborative applications," *Journal of Parallel and Distributed Computing*, 2011.

[5] S. Martin, P. Urso, and S. Weiss, "Scalable xml collaborative editing with undo," *OTM 2010*, 2010.

[6] C. J. Fidge, "Timestamps in message-passing systems that preserve the partial ordering," *Proc. of the 11th Australian Computer Science Conference*, 1988.

[7] F. Mattern, "Virtual time and global states of distributed systems," *Proc. of the Workshop on Parallel and Distributed Algorithms*, 1988.

[8] P. S. Almeida, C. Baquero, and V. Fonte, "Interval tree clocks: A logical clock for dynamic systems," *OPODIS 2008*, 2008.

[9] C. Ellis and S. Gibbs, "Concurrency control in groupware systems," *SIGMOD '89: Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, 1989.

[10] S. Kumwat and A. Kkunteta, "A survey on operational transformation algorithms: Challenges, issues and achievements," *International Journal of Computer Applications*, 2010.

[11] A. Imine, P. Molli, G. Oster, and M. Rusinowitch, "Proving correctness of transformation functions in real-time groupware," *ECSCW'03: Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*, 2003.

[12] G. Oster, P. Urso, P. Molli, and A. Imine, "Real time group editors without operational transformation," INRIA Rocquencourt, France, Tech. Rep. 5580, 2005.

[13] N. Preguiça, J. Marques, M. Shapiro, and M. Letia, "A commutative replicated data type for cooperative editing," *ICDCS '09: Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*, 2009.

[14] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "A comprehensive study of convergent and commutative replicated data types," INRIA Rocquencourt, France, Tech. Rep. 7506, 2011.

[15] B. Shao, D. Li, and N. Gu, "An algorithm for selective undo of any operation in collaborative applications," *GROUP 2010*, 2010.

[16] S. Weiss, P. Urso, and S. Molli, "An undo framework for p2p collaborative editing," *CollaborateCom 2008*, 2009.

[17] C. Rahhal, S. Weiss, H. Skaf-Molli, P. Urso, and P. Molli, "Undo in peer-to-peer semantic wikis," *ESWC 2009*, 2009.

[18] S. Weiss, P. Urso, and S. Molli, "Logoot-undo: Distributed collaborative editing system on p2p networks," *IEEE transactions on parallel and distributed systems*, 2010.

[19] M. Ahmed-Nacer, C.-L. Ignat, G. Oster, H.-G. Roh, and P. Urso, "Evaluating crdts for real-time document editing," *DocEng '11: Proceedings of the 11th ACM symposium on Document engineering*, 2011.