Dear authors,

The program committee for the 38th ACM Symposium on Principles of Distributed Computing (PODC 2019) is happy to inform you that your paper #69 Partially Replicated Causally Consistent Shared Memory: Lower Bounds and An Algorithm has been accepted. Only 50 regular papers from among 173 submissions were accepted. Your paper will be allocated a 20 minute talk at the conference and up to 10 (double column) pages in the proceedings. The camera ready version of your paper is due by May 27.

Reviews of your paper are appended to this email.

Contact Faith Ellen <faith@cs.toronto.edu> with any questions or concerns.

-PODC 2019 Program Committee

Review #69A

Paper summary

The paper studies the problem of implementing a collection of causally consistent read-write registers in an asynchronous failure-free message-passing system where not every node (called "replica") in the system stores a copy of the simulated register. Previous work has observed that partial replication requires more metadata than full replication. The goal of this paper is to characterize the tradeoff between "flexibility of replication, number of false dependencies on update messages, and overhead of the metadata".

The paper considers the class of algorithms that implement causal consistency using metadata that encodes some information about the execution observed so far. The metadata is stored at the replicas and attached to messages. Updates (i.e., writes) are applied to a local copy at a replica depending only on the metadata stored at the replica and the metadata contained in the update message.

For this class of algorithms, the paper shows that there is a set of edges in the "share graph" that must be tracked as part of the metadata of any algorithm. (The share graph was introduced in [15] and characterizes how copies of registers are distributed among the nodes.) Then the paper presents an algorithm that only tracks that...
set of edges, and thus is considered "optimal". The paper also shows a lower bound on the size of the metadata by making a connection to coloring a conflict graph; the lower bound is achievable when the share graph has certain topologies such as a tree, a cycle, or a clique.

Extensions are discussed for going from the peer-to-peer architecture to the client-server architecture (see Figure 1).

Positive Qualities of the Paper
-------------------------------
+ Finding optimal implementations of causally consistent memory is relevant to theory and practice.

+ The paper describes an algorithm for the problem that tracks the minimal set of edges in the share graph, and thus improves on a previously known algorithm, which was incorrectly thought to be optimal.

+ The conditions on the share graph are novel, to my knowledge.

+ The paper reflects a lot of effort at making the paper clear and understandable; the figures and examples are especially helpful.

Negative Qualities of the Paper
-------------------------------
- Lower bound on metadata only applies to a limited class of algorithms. No discussion is given for how much of a restriction this is.

- Formalism for the lower bound seems too handwavy: for instance, what does it mean rigorously to say that a replica "must not be oblivious to" an update on a certain edge?

- The relationships between this paper and prior work needs to be fleshed out. (See detailed comments.)

Detailed technical comments
-----------------------------
* I found it confusing to call the nodes in the systems "replicas". I would expect a "replica" to be a particular copy of a particular simulated register.

* Lamport's paper should be cited when defining happened before.

* page 4: "our algorithm in the previous section" - there is no algorithm in the previous section.

* bottom of page 4: "the metadata is some proper encoding of the information about the execution history": what exactly is a "proper encoding"? The characterization of this class of algorithms is a bit handwavy.

* Definition 4: Please give a verbal intuition for the three conditions (instead of making the reader have to figure out via the example).

* page 6, first bullet after Definition 6: "replica i's causal dependency graph may equal R_0 at some point of time, and R_1 subsequently". I don't understand what this is saying; perhaps the
use of "may" is contributing to my confusion.

* bottom of page 6: There is a duplication of the sentence beginning "Intuitively..."

* page 8: "Our algorithm is similar to standard causal multicast algorithms" - provide citations. Also, what about other algorithms for causal consistency, e.g., Schiper, Eggli, Sandoz 1989; Raynal, Schiper, Toueg 1991; Schwarz and Mattern 1994?

* Definition 10: Please provide intuitive explanation.

* Theorem 2: What is the relationship to Charron-Bost's lower bound on the size of vector clocks?

* It appears that the extensions to the client-server architecture in Section 5 can add false dependencies since extra edges are added to the share graph, something that was deemed a negative earlier in the paper. Is this true or not?

* In the related work discussion of algorithms for message passing: "but the results do not directly apply to our problem setting" - why not?

* In the related work discussion of algorithms for causal consistency, how exactly do the algorithms in [3,32,4,34,18,16] compare to yours?

* References [23] and [25] have no venue given.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Review #69B
====================================================================

Paper summary
-------------

The paper studies the problem of guaranteeing causal consistency in partially replicated shared memory systems, where each replica stores a subset of the objects of the domain. The core issue to guarantee causal consistency is related to when a replica has to make the result of an update operation available. In particular, an update operation can take effect, i.e., its result can be readable at a replica, only after all the update operations that causally precede that update have taken effect at that replica. This is achieved by storing and exchanging metadata that can keep track of causal dependencies among operations.

The paper proves a necessary condition on the metadata that must be maintained by each replica, and a lower bound on the size of the metadata. In addition, it also presents an algorithm that implements causal consistency in partially replicated shared memory systems, and uses metadata of size equal to the lower bound.

Positive Qualities of the Paper
-------------------------------

- Statements of the problem are clear.
- Objectives are relevant.
- Results that the authors aim to prove seem to be sound.

Negative Qualities of the Paper
-------------------------------

- Not well written.
- Lack of accuracy and formalism.
- Hard to prove its validity.

Detailed technical comments
---------------------------

The paper has some potential. The statement on the problem to solve is clear, the objectives are relevant, and the results that the authors aim to prove seem to be sound. However, the paper is not ready for publication. In my opinion, it lacks of accuracy and formalism. And that makes the validation of the proofs really hard.

I provide some detailed comments, which I hope can help the authors to improve the presentation of the paper.

- "overhead is expected to be larger than that for full replication" -> Can the authors better explain this? Isn't the scheme that can be adopted in full replication the worst case one for partial replication?

- "there is no reason for such delay, since the virtual register y will not be accessed by any client from this replica" -> Indeed, there is no reason for such a delay. But the authors are describing the behavior like it is an expected one. They should cite some work that is actually showing that. Doesn't the behavior depend on the specific algorithm? How is this related to distributed protocols in genuine partially replicated storage systems? E.g., "GMU: Genuine Multiversion Update-Serializable Partial Data Replication", in IEEE TPDS'15.

- In section 2, the authors should formally define the sets of replicas and registers.

- "In practice, set X_r for replica r may change dynamically," -> How relevant is this? Do the results hold anyway with such a weaker assumption?

- "In the following, e_{ij} is a directed edge from i to j." -> This should just be part of definition 1. e_{ij} \in E.

- "Share Graph [15]" -> Share graph is defined as an undirected graph in [15].

- "issue updates and applying an update" -> What's the difference between a write and an update? What does a replica do when it receives a read operation from a client? Also, a formal definition of events for sending and applying updates is needed. Subsequent definitions can rely on the formalism to define relations on updates, e.g., happened-before relation.

- "Intuitively, replica-centric causal consistency is similar to the requirement…” -> The authors should actually provide the intuition here. They just say that is similar to something else.

- Definition 2 is not that clear to me. Are two updates issued by the same replica related? For instance, in [15], program order implies causal order. Furthermore, doesn't the relation depend on the specific object that has been updated? For instance, in [15] read-from order implies causal order. By formally defining the events (read, write, update, application of updates, sending of updates), the authors can also clarify that. It is needed in definition 3 as well.

- "For three important reasons" -> 'important' is not technical. How can important be quantified in this case?

- What's the definition of client-centric?

- "in practice, the replica-centric approach is convenient" -> What's an approach in this case? Isn't a replica-centric an architecture? The usage of a timestamp seems to be a feature of an algorithm. How is this related to an approach? Furthermore, convenient is not technical. Is the algorithm space efficient? Is the space consumption optimal?

- About the paragraph "Relation with Causal Group Multicast". How relevant is presenting that paragraph in section 2. That breaks the flow, and it does not help the reader to better understand the results.

- "necessary and sufficient to “keep track” of for each replica" -> I think that something is missing after 'of'.
- “Basically, when there exists an (i,e_{jk})-loop, replica i needs to keeps information…” -> I would present this after definition 4. At this stage the reader has no clue about the meaning of such a loop.

- In definition 4, k precedes j in the definition of the cycle but the edge is defined as e_{jk}. Since that looks counterintuitive to me, the authors should provide some additional explanation. In addition, in graph theory a loop is an edge that connects a node to itself. Maybe, the authors want to name this as a cycle.

- “Figure 4 illustrates such a loop.” -> This is a clarification. Better not to include it in the definition.

- The authors might want to use the minus set operator, e.g., $\setminus$, in definition 4.

- Right after definition 4, an explanation of the properties of the definition is needed. What’s special with such a loop? Why is that needed? The feeling that I have is that the authors are trying to formalize a scenario where two updates are issued by j: one to k, and another one to r2. In addition, there exists a chain of updates along the path from r2 to k, going through i, which causally depends on the update from j to r2. I am not sure about that, though. Especially because definition 2 does not define any relation between two updates that are issued by the same replica.

- “The timestamp at replica i must reflect information regarding updates on edge e_{jk} if an (i,e_{jk})-loop exists.” -> Why that? And can an update on edge e_{jk} be an update on edge e_{kj} as well?

- “Then there is a sequence of causally dependent updates propagated along the path…” -> I think that is not an implication. Maybe the authors want to write “And a sequence of…” Also, I do not understand the relation between update u and the path. u is an update that has been issued by j to k. Is the first update in the path issued by j to r_{2}? If so, has that been issued after u? If so, is there any relation between those two updates according to definition 2? It does not seem so.

- “the timestamp must contain adequate information” -> What does adequate mean? It is not accurate.

- “vertices in S” -> Aren’t the elements of set S vertices of a graph? If so, that graph has to be in the definition of S. On the other hand, it seems that S contains elements of U, which are updates.

- “Theorem 1 will use the following terminology” -> The authors should present those as actual definitions. And as such, they should be formal. For instance:
  * what’s the definition of execution? How is R_0 derived from an execution?
  * “provided that” -> this seems an IFF (if and only if) to me.
  * “the edges between vertices corresponding to updates in...both the causal dependency graphs.” -> This sentence is not clear. There is lack of formalism. Please, the authors should define the causal dependency graphs, the set of their edges, and the relations between those sets.
  * “We will say that replica i with causal dependency graph R_0 is oblivious…” -> How does this relate to the classic definition of indistinguishable executions that is used in distributed/concurrent computing? E.g., “Universal Constructions that Ensure Disjoint-Access Parallelism and Wait-Freedom”, in PODC’12. Also, I do not understand the following. Definition 5 defines a timestamp graph of a share graph. Here the authors write about a timestamp of a causal dependency graph, which depends on an execution. That is confusing.

- “Intuitively, a replica that is oblivious to updates” -> please, delete duplicates.

- I might see the validity of theorem 1. There exists an information, i.e., timestamp graph, that keeps track of dependencies for i, and i has to be “aware of” those dependencies. However:
  * The definition of timestamp graph relies on definition 4 of (i,e_{jk})-loop, which is not clear to me. The intuition after the definition explains how such a loop can be used but it does not explain what the meaning of the loop is.
  * The definition of an oblivious replica is not formal. There is no definition of execution, and hence the relation on causal dependency graphs lacks of formalism.

- How practical is the presented algorithm? It would be useful if the authors provided a way to pre-compute E_i.
- Rewriting definition in section 3.3 is not needed.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Review #69C

Paper summary

This paper considers causally consistent data stores that are *partially replicated*, i.e., where a replica holds only a subset of the objects in the system. In principle, a partially replicated design enables a replica to apply an update $u$ without waiting to receive all updates that $u$ depends on (i.e., updates that happened before $u$ under the usual notion of happens-before). Instead, a replica only has to wait for updates to its own objects that happened before $u$. The question is how does this relaxation impact the size of message metadata that encodes causal dependencies.

Previous work studied this question using a notion of a *share graph*, wherein two replicas are connected if their subset of objects intersect, and proved that message metadata must somehow track updates across certain edges in this graph.

This paper proves a similar condition but for a tighter set of edges. This condition also translates to a metadata size lower bound (in bits). The bound isn't explicit; it's a function of the chromatic number of some abstract graph. But in some cases, the abstract bound translates to an explicit bound.

The paper also shows that the above condition is tight, by presenting an algorithm whose metadata is a vector clock indexed by those edges in the graph.

Positive Qualities of the Paper


Negative Qualities of the Paper

Compared to the previous condition by Helary and Milani (also based on share graphs), the improvement here seems to reduce timestamp size by a constant factor, not asymptotically.

Detailed technical comments

I found this paper satisfying in the sense that it really nails down a necessary and sufficient condition for which edges to track in the share graph. All the more so since you found that Helary and Milani's claimed tight condition was not, in fact, tight. The share graph edge-indexed timestamps in your algorithm are a novel and a nice idea.

However, mathematical satisfaction aside, I'm not sure how significant this result is.
* Practically speaking, it looks (from Appendix C) like you can reduce timestamp size by a constant factor, for certain share graphs. It's not clear whether such share graph topologies occur in practice. Is there any hope that this a practically relevant result?

* Theoretically speaking, it doesn't seem like there's a new theoretical tool here to take forward. The basic proof technique seems similar to Helary and Milani's.

Nevertheless, for me, "nailing down" this problem trumps these concerns. I think it would be a good paper for PODC.

Specific comments:

* In Section 1, it's not clear why "for partial replication, in general, the timestamp (or metadata) overhead is expected to be larger than that for full replication," because it seems that one can always just use vector clocks. The reason seems to be the desire to avoid false dependencies, but this becomes clear only later. I suggest to clarify this part of the introduction.

* The definition of liveness (Definition 3) is strange. It's what most would consider a safety property, since it can be violated by a finite execution. Perhaps call it "No false dependencies"?

* It would be helpful to formally define what an "update" is. E.g., a tuple consisting of replica $x$$, write value $v$$, and register $r$$. Without such a definition, the concept of comparing dependency graphs isn't formally well defined, because how do we relate updates from different executions?

* In Theorem 1, and generally in the discussion of $(i,e_{jk})$-loops, I kept getting confused by the fact that $i$$, $j$$, and $k$ have additional $r/l$ names. Is there any way to make this friendlier to the reader? At the very least, it would help to explicitly point out the places where it matters in the proofs (e.g., that $w_0 \neq w_1$ in case 3.1 of the proof).

** Paper summary **

Partial replication in shared storage systems under causal consistency (CC) is hard. The paper explores the state of the art.
It presents a necessary condition on the size of metadata required to track causality. It extends a previous work to the common-case client-server architecture.

Positive Qualities of the Paper

- Important problem that has not been well studied before.
- Good presentation of the state of the art and related work.
- Thorough treatment from first principles.
- Results appear interesting in practice.
- Pretty readable despite being quite mathematical. The conditions on metadata are well explained, with correctness proofs.

Negative Qualities of the Paper
---------------------------------
- The paper does not explain the intuition behind the formal constructs, which get more and more complex as the paper progresses.
- Expected better motivation of the algorithm.
- There should be a numeric illustration of the analytical results, to help the casual reader.
- Missing a real use-case.

Detailed technical comments
-----------------------------
- Explain how tracking edges with (small-size) timestamp graph can help building better applications, for example in the web client-server model.
- How does your solution compete with current indexing algorithms in storage systems?

p. 1
- The abstract is verbose; explain the problem tersely (like the first sentence of your introduction).
- "Partial replication:" I believe you actually mean "genuine partial replication" i.e. processes that are not replicas of variable $x$ do not receive messages relative to updates to $x$. For the terminology, see Schiper et al. OPODIS 2006 [DOI 10.1007/11945529_7].
- Define register: is it just a read-write memory location, or are you assuming some particular properties?
- "vector timestamp of length equal to the number of replicas [21]:" a better reference is Charron-Bost [IPL 1991, DOI 10.1016/0020-0190(91)90055-M]
- Figure 1b: for most people, a client-server system is a star architecture, mostly obsolete. What you depict is more like a modern tiered architecture.

p. 2
- "higher read-write latencies." Do you mean response time or propagation delay (staleness)? Under causal consistency, you can always read the local replica, hence response time should be essentially zero.
- "In general, PR yields a trade-off...": is this your intuition, or a formal result?
- Section 2: what is your failure model? What is the client model?
- The rest of the paper mentions only writes; what about reads?

p 3
- Section 2.1: define "issue" and "apply" updates.
- Inconsistent numbering: Condition 1 in Def. 2 is later called "Condition (i)".

p 4
- Your attempted definition of liveness is incorrect, as it allows to delay applying an update indefinitely (it only says "after").
- latex formatting issues: should be "Liveness:\footnote{(...)}", and "i.e., all"
- Define "client-centric".

p 5
- Your formal constructs are more and more complex as the paper progresses. Please help the poor reader by explaining the intuition and purpose of these constructs.
- Figure 4 is oriented in the unusual counter-clockwise direction; this is confusing.
- Not clear why you swap $j$ and $k$ in Definition 4.
- Notation: use \setminus (i.e., backslash) for set difference
- Under "Intuition": "The timestamp at replica $i$ must reflect information regarding...": explain why.
"an update issued by replica j to replica k": not sure what that means. An update issued at replica j needs to propagate to all other replicas, and is not directed specifically to k.
- The "Intuition" paragraph doesn't really help to understand the intuition. In particular, it never explains the rationale behind conditions (i)–(iii).

Section 6
- Related work must cite the seminal result of Charron-Bost [IPL 1991, DOI 10.1016/0020-0190(91)90055-M] establishing the minimal size of a vector clock.

p 14
- The acknowledgment is in mild violation of the blind submission rules.

Committee Decision
--------------
Partial replication under causal consistency is a really hard practical problem. This paper advances the state of the art, but remains at an excessively theoretical level to be useable by practitioners. The PC recommends adding more practical material, for instance some numerical results (e.g., in some reasonable topology, vary the replication ratio from 0 to 1: what is the size of the clocks?).