Information Processing Letters 138 (2018) 57–60

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Analyzing linearizability violations in the presence of read-modify-write operations

Hua Fan *, Wojciech Golab **,[1]

Department of Electrical and Computer Engineering, University of Waterloo, 200 University Ave. West, Waterloo, Ontario, N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 30 August 2014
Received in revised form 23 November 2017
Accepted 5 June 2018
Available online 18 June 2018
Communicated by Gregory Chockler

Keywords:
Distributed systems
Consistency
Linearizability
Verification

ABSTRACT

We consider an algorithmic problem related to analyzing consistency anomalies in distributed storage systems. Specifically, given a history of read, write, and read-modify-write operations applied by clients, we quantify how far the history deviates from the "gold standard" of *linearizability* (Herlihy and Wing, 1990). Our solution generalizes a known algorithm that considers reads and writes only.

1. Introduction

Distributed storage systems support essential online services including web search, social networking, and cloud file sharing. To meet stringent demands for high availability and low latency, such systems maintain multiple replicas of data, often in geographically distributed data centers. Storage operations are therefore executed using distributed protocols that ensure crucial correctness properties in a highly concurrent, failure-prone environment. Reasoning about the correctness of such protocols is notoriously difficult, especially for systems that support lightweight transactions (e.g., conditional write operations, increments) and latency-reducing optimizations (e.g., eventual consistency).

In this paper we focus on "black-box" testing of the consistency of a distributed storage system. The input is a history of operations applied to the system, and the output is a number that quantifies how far the history deviates from Herlihy and Wing's *linearizability* property [1]. Violations of linearizability, such as stale reads that fail to return the last updated value, can be quantified in units of time using the recently proposed Γ (Gamma) metric [2]. Such tests are more broadly applicable than rigorous proofs of correctness and model checking techniques, both of which assume knowledge of the system's internals and apply to an abstract model that may differ from the practical implementation.

Our contribution is an efficient algorithm for computing Γ given a history of read, write, and read-modify-write (e.g., conditional write) operations. The algorithm has time complexity $O(n^2)$ for a history of n operations, and generalizes a solution of Golab et al. for histories of reads and writes [2]. The simpler problem of deciding linearizability, which is equivalent to deciding whether Γ equals zero, was solved earlier by Misra [3] and by Gibbons and Korach [4]. Known algorithms for computing Γ and deciding linearizability efficiently assume a *read mapping*: for every operation that reads a value, the operation that wrote this

* Principal corresponding author.
** Corresponding author.
E-mail addresses: h27fan@uwaterloo.ca (H. Fan), wgolab@uwaterloo.ca (W. Golab).

[1] Author supported by the Google Faculty Research Awards Program and the Natural Sciences and Engineering Research Council (NSERC) of Canada, Discovery Grants Program.

https://doi.org/10.1016/j.ipl.2018.06.004
0020-0190/

value can be identified uniquely. Deciding linearizability is NP-complete otherwise [4], and solvable using state space exploration [5].

2. Preliminaries

Similarly to [1,4], we formalize the observed behavior of a storage system as a *history*—a sequence of *events* representing invocations and responses of *operations*. *Linearizability* is a widely-adopted correctness condition for histories [1]. Informally, it states that one can assign for each operation a distinct *linearization point (LP)* between its invocation and response where it "appears to take effect." More precisely, if operations on a given object are ordered according to their LPs, their responses must be consistent with some sequence of state transitions permitted by the object's sequential specification. For example, updates appear to take effect serially, and reads always return the last updated value.

To quantify linearizability violations in units of time, we generalize the definition of a history from [1] by assuming that events are ordered by explicit timestamps. The invocation and response timestamp of an operation op are denoted as $inv(op)$ and $rsp(op)$, respectively. The *time interval* for such an op is the closed interval $[inv(op), rsp(op)]$, $inv(op) < rsp(op)$. The Γ-*relaxation* of a history is obtained by shifting the invocation and response times by $-\Gamma/2$ and $+\Gamma/2$ time units, respectively. The Γ-*value* of a history is the minimum Γ for which the Γ-relaxation of the history is linearizable [2].

Because linearizability is a *local* property [1], the Γ-relaxation of a history is linearizable if and only if, for each object x, the subhistory of the Γ-relaxation comprising operations applied to x is linearizable. Therefore, to compute the Γ-value of a history H it suffices to compute the Γ-value of each subhistory where all operations are applied to a common object, and then obtain the maximum. The subhistory that requires the greatest Γ-relaxation determines the Γ-value of H.

In this paper, we consider three types of operations on objects: a *read* of value v is denoted (R, v); a write of value v is denoted (W, v); and a *read-modify-write* (RMW) that atomically reads v_r and then writes v_w, $v_w \neq v_r$, is denoted (RW, v_r, v_w).[2] An operation (W, v_w) or (RW, v_r, v_w) is called *dictating* with respect to an operation (R, v'_r) or (RW, v'_r, v'_w) if $v_w = v'_r$.

We make several assumptions regarding histories: (A1) following [2,4], each operation has both an invocation and a response event; (A2) the history begins with a write operation that assigns the initial value of the shared object, and that does not overlap in time with any other operation; (A3) each write or RMW operation assigns a unique value; (A4) each read or RMW operation has a dictating operation; and (A5) two RMW operations never read the same value. Assumptions A3–A4 establish the read mapping and circumvent NP-completeness. Assumptions A1–A5 collectively ensure that the history can be made linearizable by way of Γ-relaxation alone, as opposed to by adding or removing operations, and this implies that the Γ value of the history is well-defined.

To express our result, we borrow a number of definitions from [4]. For any value v, the *cluster for* v, denoted C_v, is the set of all operations that read value v, as well as their unique dictating operation.[3] The *zone* for a cluster C_v, denoted Z_v, is the closed interval of time between $Z_v.minrsp = \min_{op \in C_v} rsp(op)$ and $Z_v.maxinv = \max_{op \in C_v} inv(op)$. Intuitively, Z_v is a minimal subset of points in time where the LPs of operations in C_v may be chosen. A zone Z_v is called *forward* if $Z_v.minrsp < Z_v.maxinv$, meaning that the object had value v *continuously* from time $Z_v.minrsp$ to time $Z_v.maxinv$. Z_v is called *backward* if $Z_v.minrsp \geq Z_v.maxinv$, meaning that all operations in C_v overlap over the zone, and so the object had value v at least *at some point* inside the zone.

Gibbons and Korach characterize linearizability for histories of reads and writes as the absence of *conflicts* among pairs of zones [4]. Conflicts occur when two forward zones overlap, or when a backward zone is contained entirely within a forward zone. For histories that contain RMW operations, the clusters are first arranged into *cluster sequences* of the form $S = C_{v_1} C_{v_2} \ldots C_{v_k}$ where $(W, v_1) \in C_{v_1}$ and $(RW, v_{i-1}, v_i) \in C_{v_i}$ for $1 < i \leq k$.[4] A zone is defined for each cluster sequence S as the interval between $S_{minrsp} = \min_{op \in C_i, C_i \in S} rsp(op)$ and $S_{maxinv} = \max_{op \in C_i, C_i \in S} inv(op)$. The history is linearizable if and only if: (i) there are no conflicts among pairs of zones representing cluster sequences; and (ii) each cluster sequence is linearizable. The interval structure makes it possible to decide linearizability in $O(n \log n)$ time for a history of n operations [4].

3. Results

In this section we present our novel algorithm for computing Γ in histories of read, write and RMW operations. The main technical challenge lies in characterizing linearizability solely in terms of conflicts among zones; this reduces the problem of computing Γ to deciding the minimum Γ-relaxation required to remove every conflict [2]. Because Gibbons and Korach's verification algorithm (discussed at the end of Section 2) is only partially zone-based, we first modify it by introducing a new conflict type—*descendant-precedence*—to model conflicts within a single cluster sequence. Following [2], we also consider as a conflict the case where an operation that reads a value v completes before the operation that writes v. The final Γ algorithm based upon these ideas appears in Section 3.2.

3.1. Linearizability verification using conflict detection

We first characterize linearizability for reads, writes and RMW operations in terms of zone conflicts alone. We say that a zone Z_v *precedes* a zone $Z_{v'}, (v \neq v')$, denoted

[2] An unsuccessful Compare-And-Swap operation is represented by a read.

[3] An RMW operation is always part of two clusters under assumptions A1–A4.

[4] Assumption A2 ensures that (W, v_1) exists to form the cluster sequence.