Future Generation Computer Systems 51 (2015) 20–35

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Bumper: Sheltering distributed transactions from conflicts

Nuno Diegues *, Paolo Romano

INESC-ID, Rua Alves Redol 9, Lisbon, Portugal
Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais 1, Portugal

HIGHLIGHTS

- Cloud data stores that expose distributed transactions suffer of transaction aborts.
- We identify that many aborts can be avoided while preserving strong consistency.
- With Bumper, we reduced aborts for transactions to nearly 0% in many workloads.
- The performance and scalability improved up to 3x in conflict-prone applications.
- Our approach uses a novel distributed protocol that scales to hundreds of servers.

ARTICLE INFO

Article history:
Received 6 February 2015
Received in revised form
15 March 2015
Accepted 3 April 2015
Available online 30 April 2015

Keywords:
Distributed transactions
Spurious aborts
1-copy serializability
High scalability

ABSTRACT

Large scale cloud applications are difficult to program due to the need to access data in a consistent manner. To lift this burden from programmers, Deferred Update Replication (DUR) protocols provide serializable transactions with both high availability and performance in read-dominated workloads. However, the inherently optimistic nature of DUR protocols makes them prone to thrashing in conflict-intensive scenarios: existing DUR schemes, in fact, avoid any synchronization during transaction execution; thus, these schemes end up aborting any update transaction whose reads are no longer up to date by the time it attempts to commit.

To tackle this problem, we introduce Bumper, a set of innovative techniques to reduce aborts of transactions in high-contention scenarios. At its core, Bumper relies on two key ideas. First, we spare update transactions from spurious aborts (i.e., unnecessary aborts of serializable transactions), by attempting to serialize the transactions in the past. For this, we use a novel distributed concurrency control scheme that we call Distributed Time-Warping (DTW). And second, we avoid aborts due to contention hot spots (that cannot be tackled by DTW) via a programming abstraction that we call Delayed Actions. These, allow for efficiently serializing, in an abort-free fashion, the execution of conflict-prone data manipulations.

By means of an extensive evaluation, on a distributed system of 160 nodes, we show that Bumper can boost performance up to 3× in conflict-intensive workloads, while imposing negligible (about 2.5%) overheads in uncontended scenarios.

1. Introduction

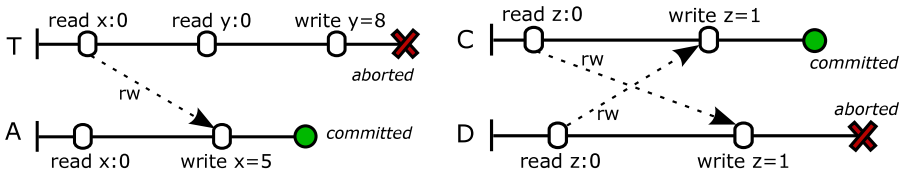
The advent of the cloud computing paradigm has empowered programmers with the ability to scale out their applications easily to hundreds of nodes in a distributed system. However, developing applications capable of effectively exploiting the computational capabilities of large scale distributed cloud platforms is far from being a trivial task.

Data management systems help programmers to deal with this by providing the abstraction of serializable distributed transactions. A well-established approach to implement this abstraction is that of Deferred Update Replication (DUR) [1,2]: servers replicate data, to which clients perform requests of transactions; servers synchronize only at the commit of the transactions to either atomically update data across the servers or to abort.

This type of protocols follows an optimistic approach to concurrency control [3], as accesses in transactions are performed without enforcing synchronization, and the serializability is ensured in the commit operation during validation of the transaction. The literature is rich in enhancements to optimistic concurrency control, for instance by relying on multi-versions [4–6]: these allow efficient read-only transactions, by sparing them from any aborts and remote validations. Another key property to enhance scalability is that of *genuine* partial replication. In this property the execution

* Corresponding author at: INESC-ID, Rua Alves Redol 9, Lisbon, Portugal.
E-mail address: nmld@tecnico.ulisboa.pt (N. Diegues).

http://dx.doi.org/10.1016/j.future.2015.04.002
0167-739X/



(a) Abort that time-warp can avoid. (b) Abort that delayed action can avoid.

Fig. 1. Examples of executions that cause spurious aborts when using typical DUR protocols [12,4,5,13,14]. These can be avoided by using the two techniques that Bumper encompass: namely time-warping and delayed actions.

of a transaction can only involve nodes that replicate data items it accessed [2].

1.1. Identified problems

The aforementioned DUR systems were shown to perform well, even in large scale, while providing strong semantics in the form of transactions. However, as we shall see later in the paper, the scalability of these systems can be critically challenged in conflict-prone scenarios.

The main factors constraining the scalability of these systems are of a twofold nature. They are both related to the algorithms used to regulate concurrency among transactions, as well as to the degree of parallelism admitted by the applications:

- State of the art DUR protocols rely on overly conservative validation schemes. These schemes abort an update transaction, whenever any of its reads is no longer up to date, by the time it requests to commit. This mechanism gained wide adoption because it can be implemented efficiently. However, we note that it does not represent a necessary condition to detect non-serializable histories [7], and, as we will show, it can induce a high number of spurious (i.e., unnecessary) aborts.
- It is well understood that the maximum degree of parallelism (and hence, of scalability) admitted by any transactional system is deeply affected by the data access patterns exhibited by the applications deployed over them [8,9]. For instance, several standard online processing transactional profiles are characterized by contention hot spots; these are frequently updated data items, such as warehouse balance counters in the known Transaction Processing Performance Council Benchmark C (TPC-C) [10]. Transactions accessing such data items are not only inherently non-parallelizable; they are also prone to undergo repeated aborts, which can have detrimental effects on the system's throughput and user-perceived responsiveness.

1.2. Contributions

We address the issues discussed above by introducing Bumper: a set of mechanisms aimed to shelter transactions from conflicts, thus enhancing scalability in conflict-prone scenarios while ensuring strong-consistency (1-copy serializability [11]). At its core, Bumper relies on two novel mechanisms to prevent different types of conflicts: *distributed time-warping* (time-warping for the sake of brevity) and *delayed actions*.

The idea at the basis of time-warping is to prevent (a type of) spurious aborts that do not threaten the serializability of transactions. We illustrate such type of aborts in Fig. 1(a): many typical DUR protocols [12,4,5,13,14] abort the update transaction T in the example; the reason is that it read item x, and transaction A concurrently committed with a write to x, making the read of T stale.

The intuition of this common approach that leads to such aborts is that, in DUR protocols, update transactions are required to commit in the logical present, i.e. the snapshot observed by transaction T must be valid by taking into account every transaction committed before T. Looking back at Fig. 1(a), we see that such an approach leads to a spurious abort of T. In fact, in such a scenario, it is possible to safely serialize T before A, and thus spare its abort.

A key property of time-warping consists in its efficiency. From a theoretical perspective, it is straightforward to design an algorithm capable of accepting every serializable history: it suffices to track the full graph of dependencies between *every* transaction and to ensure its acyclicity [3]. Unfortunately, this is an unbearably onerous approach, especially in a large scale system. Conversely, time-warping uses a novel, lightweight validation mechanism, which tracks only *direct* dependencies developed by a transaction during its execution (such as the ones shown in Fig. 1). Not only does this mechanism prevent spurious aborts that would be caused by the validation schemes employed by traditional systems; it can also be implemented very efficiently and in a *genuine* fashion (i.e., by only collecting information at nodes that are involved in the distributed transaction). One of the key contributions of this article is precisely in designing an efficient implementation of time-warping. We show that this technique can effectively reduce aborts, while introducing minimal additional overhead.

Clearly, there exist limits to the aborts that time-warping can avoid. An example is shown in Fig. 1(b), where transactions C and D read and write the same data item z. Since they mutually miss each others write, neither one can be time-warp committed and serialized before the other.

To cope with these challenging conflict patterns, we introduce a programming abstraction for distributed transactions, complementary to time-warping, which we name *delayed action*: this is a code fragment to be executed transactionally, but whose side-effects/outputs are not observed elsewhere in the encompassing transaction. By allowing programmers to wrap conflict-prone code within a delayed action, Bumper can postpone its execution until the transaction's commit procedure. At that point it can guarantee that the snapshot observed will not be invalidated by a concurrent transaction. This allows ensuring that a delayed action cannot cause the abort of its encompassing transaction, while guaranteeing that it is atomically executed in the scope of the transaction that triggered it.

The key ideas at the basis of Bumper (i.e., time-warping and delayed actions) are applicable to several systems, such as SCORe [5], P-Store [2], Spanner [6] or D^{2}STM [12]. In this paper, we demonstrate their practicality by integrating them with SCORe [5], a highly scalable DUR protocol that employs genuine partial replication and a decentralized multi-versioning algorithm. Our evaluation, employing 4 well-known benchmarks and 160 nodes, shows that Bumper can boost performance up to 3× in conflict-intensive workloads, with negligible (2%) overheads in uncontended scenarios.

The structure for the rest of this paper is as follows. We start by presenting the system model in Section 2. Section 3 presents the mechanisms at the base of Bumper, whose correctness we discuss in Section 4. We further refine the proposed solution in Section 5, and present its experimental evaluation in Section 6. Finally, we overview related work in Section 7 and conclude in Section 8.