Towards Affordable External Consistency for Geo-Replicated Systems

Manuel Bravo$^{1,2}$, Luís Rodrigues$^{2}$, Peter Van Roy$^{1}$

$^1$Université Catholic de Louvain, Belgium  $^2$INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Abstract—We propose a novel isolation definition called external causality, which aims at making external consistency affordable for geo-replicated systems. In this short paper, we (i) outline this idea—informally defining external causality, (ii) motivate it with a simple example of an auction service that illustrates how developers may benefit from such a model, and (iii) discuss current state of the project and technical details that we consider relevant for anyone that aims at implementing external causality.

I. INTRODUCTION

Cloud services leverage geo-replication to bring the application state closer to clients, with the intention of lowering latency responses. Unfortunately, due to the long network delays among geo-locations, synchronous replication is prohibitively expensive, making the maintenance of cross-site consistency very challenging and costly. Therefore, system designers are faced with a dilemma [13], [23]: either favor low latency adopting weaker consistency models such as eventual consistency [17], [3], [1] and causal consistency [28], [18], [20], [4], [34]; or favor strong consistency imposing higher latency responses [16].

A promising approach to alleviate the tension between strong consistency and performance in geo-replicated settings consists in allowing multiple consistency levels to coexist [33], [32], [8], [27], [26]. This alternative allows operations to run with different consistency levels: from weaker models that allow for asynchronous replication to stronger levels that unavoidable require cross-site coordination. Therefore, under the hypothesis that most of the application operations only require weak guarantees, these solutions are able to provide meaningful consistency without hampering performance.

Interestingly, previous solutions disregard externally or globally consistent guarantees [22]. An externally consistent system, e.g., Spanner [16], guarantees that clients are always served with a view of the system consistent to how an external observer would witness the succession of events. As a simple example, consider an auction application: for the auctioneer to close an auction, it first has to gather all bids made, and then select the highest. In a geo-replicated system, this implies reading from all replicas, in order to ensure that all bids are considered.

Our work aims at making the following contributions in order to make external consistency affordable for cloud services:

1) The introduction of a novel isolation definition called external causality which adds externally consistent guarantees to causal consistency. The idea behind external causality is that most operations, namely internal operations, are executed locally and asynchronously replicated. Internal operations are causally ordered among them. Nevertheless, stronger operations called external operations, which provide externally consistent guarantees, coexist with internal operations. An external operation is by definition ordered after any other operation—not only other external operations—already applied in the system at the time the external operation was issued. Thus, external operations allow developers to make stronger assumptions, potentially overcoming in part some of the traditional criticisms of causal consistency [15] (e.g., back-channel dependencies), and simplifying application development.

2) To identify the conditions under which operations must be classified as external. Intuitively, operations that require awareness of the whole database must be classified as external.

3) The design of a novel transactional protocol that provides causally consistent snapshot reads and ensures read atomicity attaining latency optimality [30] under both full and partial geo-replication. The efficiency of such a protocol is crucial for the implementation of external causality, as internal operations—the vast majority—will run on it. Unfortunately, the state-of-the-art has either focused on full geo-replication [29], [4], [34], implements weaker semantics [9], [12], or a combination of both [28], [5], [20], [18].

We plan to design, build and evaluate a system that implements external causality.

II. EXTERNAL CAUSALITY

External causality is an isolation definition that considers the coexistence of two type of operations: internal and external. The difference between these operations is given by the snapshot from which each operation is allowed to read. On the one hand, internal operations—as they are assumed to be predominant—read from a snapshot whose only condition is to be consistent with causality. This fact, allows internal operations to be implemented in a scalable manner without compromising availability, and providing low latency. In fact, internal operations provide the strongest semantics implementable by an always-available geo-replicated system [6], [31]. On the other hand, external operations read from a snapshot that includes all previously installed operations. This includes operations being originated at any replica of the
system, adding external consistency guarantees. Unfortunately, external operations will exhibit significantly higher latencies than internal operations, as they require all replicas to coordinate. Nevertheless, this type of operations should be infrequently used, such that most operations are internal and exhibit low latency.

Following, we define the four properties that characterize external causality.

**Property 1 (Causal Snapshot Reads).** All operations read from a causally consistent snapshot.

A snapshot \( S \) is causally consistent iff for any two object versions \( x_i, y_j \in S \) where \( x, y \) are the objects and \( i, j \) are the versions, \( \exists x_k \) such that causally precedes \( y_j \) and causally follows \( x_i \) (denoted as \( x_i \leadsto x_k \leadsto y_j \)).

We now define the conditions that the causal snapshot from which internal and external operations read must satisfy.

**Property 2 (Internal Causal Snapshot Reads).** An internal operation reads from a causal snapshot that includes at least all operations that causally precede it.

**Property 3 (External Causal Snapshot Reads).** An external operation reads from a causal snapshot that include the most recent installed versions (in the whole database) as of the time when the operation began.

Finally, both internal and external operations must provide read atomicity in an effort to prevent fracture reads [7].

**Property 4 (Read Atomicity).** Either all updates for a given transaction are included in a snapshot, or none.

Read atomicity is instrumental for ensuring state transitions consistently with respect to certain invariants such as foreign key constraints to represent relationships between data records (e.g., the symmetry of the friendship or the like relationship in a social network application [14]), secondary indexing to optimize location of partitioned data by attributes, and the maintenance of materialized views [7].

**III. USE CASE: AN AUCTION SERVICE**

In this section, we motivate why external causality has the potential of simplifying the development of cloud services. Let’s assume an auctioning service, similar to eBay [2], in which, among possibly others, two roles are involved: auctioneers and buyers. Auctioneers initiate the sale of items by describing the item and starting the bidding. Buyers bid on items begin sold by auctioneers. We say that an auction is open during the time period defined by the auctioneer. Once the period is over, we say that the auction is closed. At that point, the buyer with the highest bid wins the auction.

Among the many operations available in such a service, such as initiate an new auction, modify the description of an item, extend the time limit of an auction, place a bid (most probably the most common operation), among many others; only closing the auction and therefore identifying the winner is the one requiring global guarantees. This fits our vision, as only the latter operation would be classified as external, letting all other operations to be executed very efficiently, as internal operations.

Without external causality, designers would be forced to opt for some of the following options:

- Favor performance and give up consistency by execute all operations under transactional causal consistency [29], [4], [34]. Therefore, unless a probably complex and error-prone mechanism is implemented at the application level, it would be impossible to ensure that the winner of an auction is the buyer with the highest bid.
- Embrace stronger consistency guarantees that unnecessarily will damage the overall performance of the system [32], [10], [16], [11].

**IV. CURRENT STATE AND PLAN**

We are building a prototype called Exosphere that implements external consistency. Exosphere is being built using the Erlang/OTP programming language. We discuss two important implementation choices: the causal multicast implementation and the use of hybrid clocks.

Our protocols rely on a causal multicast abstraction to implement both the base transactional causally consistent protocol used by every operation and the protocol that computes the snapshot of external operations. Our implementation builds upon our previous work Saturn [12], a distributed metadata service that enforces causal consistency. The reasons (among others) behind this choice are that Saturn: (i) implements genuine partial replication [24], (ii) requires negligible metadata management, (iii) mitigates the impact of false dependencies introduced when compressing metadata, and (iv) its design favors an easy integration with data services as Exosphere.

Another interesting detail of the implementation is the embrace of hybrid clocks [25], which combine the use of loosely synchronized physical clocks and logical clocks. Physical clocks make distributed protocols resilient to workload skew, as they, unlike logical clocks, grow at a similar pace [20]. Nevertheless, distributed protocols that solely use loosely synchronized physical clocks [20], [19], [4], [21] may need to block operations due to clock skew in order to ensure correctness. The logical part of the hybrid clock make distributed protocols resilient to clock skew. This is fundamental for Exosphere, as otherwise, it could not attain latency optimality [30].

Our plan is to evaluate Exosphere with both synthetic workloads—to explore the workload space and better understand Exosphere bottlenecks, and with real applications—to better reason about the feasibility of external causality. We will compare Exosphere with two other systems: a purely transactional causally consistent system (all internal operations) and a externally consistent system (all external operations). This will help us to assess whether external causality can indeed exhibit significantly lower latencies than an externally consistent system without adding a very significant overhead when compared to a system providing weaker guarantees.
REFERENCES