Geo-placement: Geo-replicated Database Placement

Victor Zakhary Faisal Nawab Divyakant Agrawal Amr El Abbadi
Department of Computer Science, University of California, Santa Barbara, CA 93106
Email:{victorzakhary,nawab,agrawal,amr}@cs.ucsb.com

***Abstract*—Geo-replication is the process of maintaining copies of data at geographically dispersed datacenters for better availability and fault-tolerance. The distinguishing characteristic of geo-replication is the large wide-area latency between datacenters that varies widely depending on the location of the datacenters. Thus, choosing which datacenters to deploy a cloud application has a direct impact on the observable response time. We propose an optimization framework that automatically derives a geo-replication placement plan with the objective of minimizing latency. By running the optimization framework on real placement scenarios, we learn a set of placement principles for geo-replication. In this paper, we highlight the geo-replication placement principles.**

I. Introduction

Geo-replication is used to bring data closer to the user and to increase the level of read availability. When geo-replicating a cloud application, the system administrator is faced with an important design decision: at which datacenters should the data be placed? Cloud providers offer more than a few datacenters for deployment (*e.g.*, Amazon AWS hosts applications in 10 datacenters around the world each with multiple availability zones). With multiple cloud providers, the possibilities for geo-replicated deployments are the subsets of tens of datacenters. The placement decision affects many performance characteristics.

In this work, we focus on the effect of geo-replication placement on the response time for transactional workloads with serializability as the correctness guarantee. The topology affects response time differently for different replication protocols. Our study focuses on majority protocols.

We propose an optimization framework to make geo-replication placement decisions. The optimization framework, *framework* for short, constructs a model of the system. The model includes the topology, workload, and user distribution. Then, the model is used in an optimization problem formulation with the objective of minimizing response time in addition to adhering to fault-tolerance and quality-of-service guarantees. The model incorporates a set of best practices, or principles, for geo-replication placement.

II. Background

System model. Our model of geo-replication consists of a topology of datacenters and clients executing transactional workload. Data is fully replicated to a subset of datacenters. Users issue transactions that consist of read and write operations. Each client execute transactions back-to-back. The execution of transactions depends on the replication protocol. We focus in this work on majority protocols. Majority protocols vary widely and next, we explain the variation that we use of the majority protocol.

Majority. We adopt a variant of the majority protocol. A client executes a transaction by performing the reads and buffering the writes. Read requests are sent to a majority of datacenters. The highest version read is used. After executing reads and writes a vote request is sent to datacenters. The vote request consists of the read versions and the buffered write operations. Each datacenter, upon receiving a vote request, attempts to lock objects that are being written. Additionally, it verifies that the read versions were not overwritten. If both are successful, the datacenter sends back a positive vote. Otherwise, a negative vote is sent. The client commits the transaction if a majority of positive votes is received and aborts the transaction otherwise. The client sends the decision *to all replicas*. Once a majority acknowledges the receipt of the decision, the transaction terminates. The transaction latency is the time from the beginning of executing operations until terminating the transaction. We define the commit latency as the time spent committing the transaction, which is equivalent to the transaction latency without the time spent executing operations.

Related work. Geo-replication and data placement are areas that have undergone extensive research. Recent works study the problem of data placement for geo-replication [4], [2], [1]. Spanstore [4] is the closest work to our optimization framework. It tackles the data placement problem in geo-replication using an optimization formulation. Ping et. al. [2] propose the use of a utility function to derive a placement that balances between speed and availability. Volley [1] analyzes usage logs and leverages an optimization formulation to place data partitions. Unlike our optimization framework, these works do not support a multi-access transactional workload with strong consistency. A transactional workload requires more complex coordination between replicas to detect conflicts. In addition, our framework is designed to engage in the design decisions of which techniques to be used in the replication protocol, and illuminates unintuitive optimizations to achieve a better performance.

III. Geo-replication Placement

Optimization Framework

Where copies of data are placed is a critical design decision that affects, among others, the performance of the application. To address the large space of placements and replication protocol variations, we developed an optimization framework. The framework takes as input the replicas network topology,

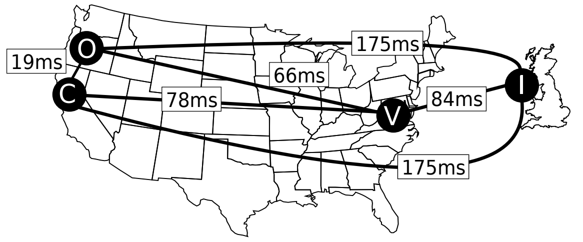

Fig. 1. An example of replicating to four datacenters in Oregon (O), Virginia (V), California (C), and Ireland (I)

workload parameters, and availability constraints. It searches through the space of placements and protocol variations and chooses the ones that minimizes response latency. In our study, we have repeatedly seen a set of optimizations reoccur. Some of them are straight-forward and some of them are surprising. Next, we summarize the derived placement principles for geo-replicated systems.

Commit hand-off. In a fully-replicated system, a client accessing data typically sends the transaction request to the local, or closest, replica. Our optimization framework shows that this is not always the best practice in a geo-replicated environment; it is sometimes better to send the transaction request to a datacenter other than the local one. For example, consider Figure 1 that shows an example of geo-replication: four datacenters in Oregon (O), Virginia (V), California (C), and Ireland (I). The communication Round-Trip Times (RTTs) between the datacenters are shown in the figure. The RTTs between Ireland and the other datacenters are significantly higher than the remaining RTTs. Assume that a majority protocol is used to commit transactions in this scenario. The commit latency of a majority protocol is two RTTs to the closest majority. Thus, the commit latency of clients in Oregon is $132ms$, in California and Virginia is $156ms$, and in Ireland is $350ms$.

The large latency of clients in Ireland is due to the common convention that driving the transaction commitment from the local datacenter is the best practice. However, running this scenario in our optimization framework shows that this convention is not always true. In fact, clients in Ireland are in a better position sending their transaction requests to Virginia. Committing a transaction in Virginia takes $156ms$. And sending the request from Ireland to Virginia and waiting for the decision to be sent back takes $84ms$. This means that the commit latency becomes the sum of the two latencies, which is equal to $240ms$ achieving 31% improvement for clients in Ireland.

Passive replicas. Our second principle is a byproduct of the commit hand-off principle. In the scenario in Figure 1, when applying the hand-off principle, clients at Ireland send their transaction requests to Virginia. Clients in other datacenters send transaction requests to their local replicas. This means that the replica at Ireland does not receive transaction requests. With this knowledge, it is easy to observe that the demand placed on Ireland is lower than the other datacenters. For most of the time, Ireland will only serve read requests and receive the outcome of transactions and not drive the commitment of transactions. Thus, less resources need to be provisioned in Ireland and more resources need to be provisioned elsewhere.

When the optimization framework recommends the hand-off principle, it is sometimes accompanied by an interesting side effect. The side effect is that the replica that is not receiving transaction requests becomes a *passive replica*. A passive replica is a replica that serves read requests but does not engage in the commit protocol. This means that it does not become part of the majority protocol, but more like a cache of data used for reading. In the example in Figure 1, this means that the majority protocol will involve getting votes only from Oregon, Virginia, and California. The number of replicas to constitute a majority has thus been lowered from three replicas when four datacenters were involved to two replicas now that only three replicas are involved. This makes the transaction latency of Oregon and California $38ms$, of Virginia $132ms$, and of Ireland $216ms$. Making Ireland a passive replica reduced the average latency by 39% which is useful for systems that require to tolerate only a single datacenter failure.

Optimistic reading. A conventional majority protocol reads from a majority of replicas to ensure that the most recent version is read. However, it is possible to optimistically read from the local replica only, and then validate the read in a majority of replicas in the commit phase [3]. The choice of whether to read optimistically or from a majority is controlled by a trade-off between the latency of read operations and contention. An optimistic read lowers the read latency but it increases contention because the version at the local replica might be stale. Reading from a majority requires a larger latency but ensures getting the most recent version. Running our optimization framework on various geo-replication scenarios reveals that optimistic reads are more favorable.

IV. Conclusion

In geo-replication, the location of replicas plays a significant role in performance. We have developed an optimization framework that derives the optimal placement of replicas. Unlike prior work, our optimization framework models a transactional workload. Geo-replication placement principles are derived using the optimization framework. Most surprising is the hand-off principle, which shows that sometimes it is rewarding to send transaction requests to a farther datacenter rather than the local one.

References

[1] S. Agarwal et al. Volley: Automated data placement for geo-distributed cloud services. In *NSDI*, 2010.

[2] F. Ping, J.-H. Hwang, X. Li, C. McConnell, and R. Vabbalareddy. Wide area placement of data replicas for fast and highly available data access. In *the International Workshop on Data-intensive Distributed Computing (DIDC)*, 2011.

[3] R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM TODS*, 1979.

[4] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha. Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. In *SOSP*, 2013.