CSPAN: Cost-Effective Geo-Replicated Storage Spanning Multiple Cloud Services

Zhe Wu
UC Riverside
zwu005@cs.ucr.edu

Michael Butkiewicz
UC Riverside
butkiewm@cs.ucr.edu

Dorian Perkins
UC Riverside
dperkins@cs.ucr.edu

Ethan Katz-Bassett
Univ. of Southern California
ethan.kb@usc.edu

Harsha V. Madhyastha
UC Riverside
harsha@cs.ucr.edu

ABSTRACT

Existing cloud computing platforms leave it up to applications to deal with the complexities associated with data replication and propagation across data centers. In our work, we propose the *CSPAN* key-value store to instead export a unified view of storage services in several geographically distributed data centers. To minimize the cost incurred by application providers, we combine two principles. First, *CSPAN* spans the data centers of multiple cloud providers. Second, *CSPAN* judiciously trades off the lower latencies and the higher storage and data propagation costs based on an application's anticipated workload, latency goals, and consistency requirements.

Categories and Subject Descriptors

C.2.4 [**Communication Networks**]: Distributed Systems—*Distributed applications*

Keywords

Cloud services, Storage system, Optimization

1. INTRODUCTION

Today, several cloud providers offer storage as a service. Amazon S3 [1], Google Cloud Storage [2] (GCS), and Microsoft Azure [3] are notable examples. All of these storage services provide storage options in several data centers distributed around the world.

Ideally, web applications should be able to provide low-latency service to their clients by leveraging the distributed locations offered by these services. A number of realities however complicate this goal. First, almost every storage service exports an isolated pool of storage in each of its data centers, leaving data replication to individual applications. Second, while replicating all objects to all data centers can ensure low latency access [4], that approach is costly and may be inefficient. Some applications may value lower costs over the most stringent latency bounds, and different applications may demand different degrees of data consistency. Currently, every application needs to reason on its own about where and how to replicate data to satisfy its latency goals and consistency requirements at low cost.

SIGCOMM'13, August 12–16, 2013, Hong Kong, China.
ACM 978-1-4503-2056-6/13/08.

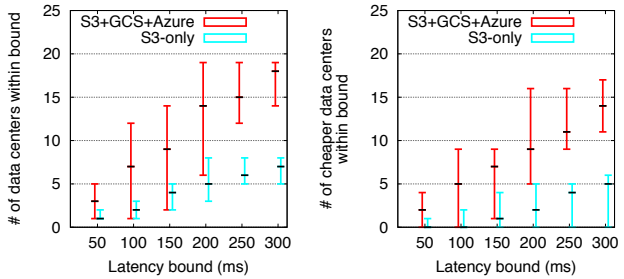


Figure 1: For each latency bound, the minimum, median and maximum (across data centers) of the number of data centers that are within latency bounds (a) and are cheaper than the local data center along some dimension (storage, request, and network bandwidth) (b). Within any latency bound, a storage service that spans multiple cloud services offers a larger number of data centers, many of which are cheaper.

To address this problem, we propose *CSPAN*, a key-value store that presents a unified view of storage services present in several geographically distributed data centers. An application can be deployed on any subset of the data centers that CSPAN spans, and at each of these data centers, the application issues local GETs and PUTs to CSPAN, leaving the complexities of replicating objects across data centers to be handled by CSPAN. Unlike existing geo-replicated storage systems [4], our primary focus is to minimize the cost incurred by latency-sensitive application providers.

Two key principles guide our design of *CSPAN* to minimize cost. First, *CSPAN* spans data centers across multiple cloud services. A storage service that is spread across multiple cloud providers offers both performance and cost benefits. On one hand, the union of data centers across multiple cloud providers offers a geographically denser set of data centers than any single provider has. Due to this, *CSPAN* can offer lower latency guarantees than that possible by utilizing the data centers in a single cloud service. On the other hand, the cost of storage and networking resources can significantly differ across cloud providers. *CSPAN* exploits these discrepancies in pricing to drive down the costs incurred in satisfying application providers' latency goals. Second, to minimize cost, *CSPAN* judiciously determines where to replicate every object and how to perform this replication. When making the decision, *CSPAN* takes into consideration several factors: the anticipated workload for the object (i.e., how often different clients access it), the latency guarantees specified by the application, the level of data consistency desired by the application (e.g., strong versus eventual), and the pricing models of storage services that *CSPAN* builds upon.

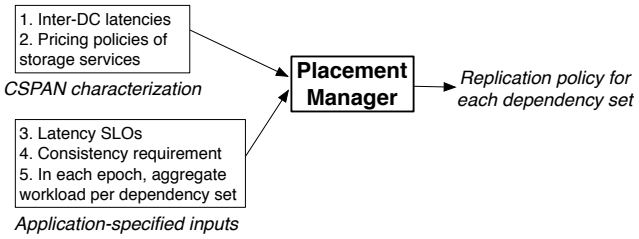


Figure 2: Overview of Placement Manager's inputs and output.

2. OVERVIEW

2.1 Why multi-cloud?

Figure 1 shows that using multiple cloud providers can enable *CSPAN* to offer lower GET/PUT latencies and allow applications to meet a fixed latency service level objects (SLOs) at potentially lower cost than S3-only deployment. For nearly all latency bounds and data centers, we find that deploying across multiple cloud providers increases the number of nearby options, and opens the opportunity for savings without compromising performance.

2.2 Goals

In synthesizing geographically distributed storage services into a single key-value store, we have three objectives.

- **Minimize cost.** This is the primary goal of *CSPAN*.
- **Respect latency SLOs.** *CSPAN* must ensure that the application's latency service level objectives (SLOs) are met.
- **Flexible consistency.** *CSPAN* should respect requirements for strong consistency and exploit cases where eventual consistency suffices to offer lower latencies.

2.3 Challenges

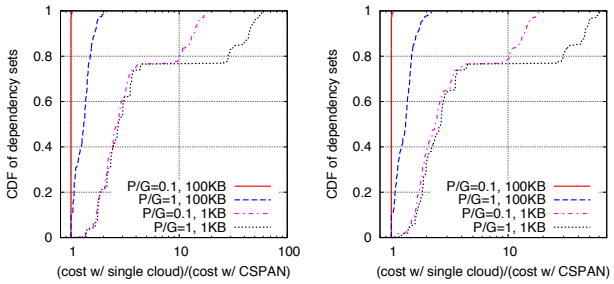
Designing *CSPAN* to meet these goals is challenging for several reasons.

- **Inter-dependencies between goals.** *CSPAN*'s ability to minimize cost for an application is critically dependent on the application's latency and consistency requirements.
- **Dependence on workload.** Even if two applications have the same latency and consistency requirements, the most cost-effective solution for storing their data may differ due to different workloads.
- **Multi-dimensional pricing.** Cost minimization is further complicated by the fact that each of the storage services prices its use based on several metrics.

2.4 Design

At the heart of *CSPAN*'s design is a central Placement Manager (*PM*). The *PM* periodically determines the policy that *CSPAN* should use to replicate objects, and it communicates this replication policy to each of the data centers in which *CSPAN* is deployed.

As shown in Figure 2, to determine the replication policy, the *PM* requires three types of inputs: a characterization of *CSPAN*'s deployment, the application's latency and consistency requirements, and a specification of the application's workload. We let applications specify aggregate workloads across all objects with the same dependency set; an object's dependency set is the set of data centers from which the application will issue PUTs and GETs for the object. CSPAN employs the same replication strategy for all objects with the same dependency set. For any particular dependency set Dep, the replication policy output by *PM* specifies 1) the set of data centers that maintain copies of all objects with that dependency set, and 2) for each data center in Dep, which of these copies *CSPAN* should read from and write to when the application instance at that data center issues a GET or PUT on an object with that dependency set.



(a) Eventual consistency (b) Strong consistency

Figure 3: Comparison of cost with *CSPAN* as compared to that possible if it spanned the data centers of a single cloud service.

The *PM* addresses the trade-off between storage, networking, and PUT/GET requests costs by formulating the problem of determining the replication policy for a given dependency set Dep as a mixed integer program. We build different formulations for eventual consistency and strong consistency. In the case of strong consistency, we provide linearizability, i.e., all PUTs issued by the application for a particular object are ordered and any GET returns the latest PUT for the object.

3. PRELIMINARY RESULTS

We deploy *CSPAN* prototype across the storage services offered by S3, Azure and GCS. We compare the cost with *CSPAN* with the minimum cost required if we used only S3's data centers for storage. Our results are qualitatively similar when we consider application deployments on Azure or GCS. We perform this comparison for four workloads: all four combinations of PUT/GET ratios of 0.1 and 1 and average object sizes of 1 KB and 100 KB. When analyzing the eventual consistency setting, we consider the GET and PUT SLOs to be 100 ms, which is half of the minimum SLO possible when using a single replica. In the strong consistency case, we consider GET SLO=100 ms, and PUT SLO=720 ms; 720 ms is the minimum PUT SLO necessary if an object was replicated at all data centers in its dependency set.

Figure 3 shows that *CSPAN*'s use of multiple cloud services consistently offers significant cost savings when the workload includes 1 KB objects. The small object size makes networking cost negligible in comparison to PUT/GET requests costs, and hence, *CSPAN*'s multi-cloud deployment helps because PUT and GET requests are priced 50x and 4x cheaper on Azure as compared to on S3. When the average object size is 100KB, *CSPAN* still offers cost benefits for a sizeable fraction of dependency sets when the PUT/GET ratio is 1. In this case, since half of the workload (i.e., all PUT operations) require propagation of updates to all replicas, *CSPAN* enables cost savings by exploiting discrepancies in network bandwidth pricing across cloud services.

4. REFERENCES

[1] Amazon S3. `http://aws.amazon.com/s3`.
[2] Google cloud storage. `http://cloud.google.com`.
[3] Windows Azure. `http://www.windowsazure.com/`.
[4] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen. Don't settle for eventual: Scalable causal consistency for wide-area storage with COPS. In *SOSP*, 2011.