Taming the Cloud Object Storage with MOS

Ali Anwar*, Yue Cheng*, Aayush Gupta†, Ali R. Butt*
*Virginia Tech, †IBM Research – Almaden

Abstract

Cloud object stores have become the most widely used form of cloud storage in recent years. They combine key advantages such as high availability, elasticity and a pay-as-you-go pricing model, which allows applications to scale as the usage increases or decreases, with HTTP-based RESTful APIs for data management. Cloud object stores today are deployed using a single set of configuration parameters for all different types of applications. This homogeneous setup results in all applications experiencing the same service level (e.g., data transfer throughput, etc.). However, the vast variety of applications expose extremely different latency and throughput requirements. To this end, we propose MOS, a Micro Object Storage architecture with independently configured microstores each tuned dynamically for a particular type of workload.

1. Motivation

Cloud object stores, such as S3 [1], Swift [3] and Ceph [2], have become the most widely used form of cloud storage in recent years. Cloud object stores today are deployed using a single set of configuration parameters for all different types of applications. This homogeneous setup results in all applications experiencing the same service level (e.g., average latency per request, data transfer throughput, and queries per second (QPS)). However, the vast variety of applications expose extremely different latency and throughput requirements. For example, a social networking or photo sharing application requires low latency to keep a responsive user experience, whereas backup services can tolerate higher latency but require sustained high throughput. Extant object storage services compromise application performance to gain the flexibility advantages. The situation is further complicated by the fact that due to regular system upgrades and introduction of new storage architectures, data centers hosting these object stores are becoming increasingly heterogeneous. However, with the "one-size-fits-all" style of object store deployment, it is impossible to match each set of specific types of hardwares with the right type of application workload. For example, latency-sensitive small-object workloads would require low-latency storage devices and powerful CPU processing capacity whereas large object write-only workloads can be supported with a combination of high network bandwidth and weaker CPU power. Under this scenario, meeting SLA requirement for one of the workload may require, i) adding hardware resources which may not improve the performance for other workloads, ii) software tuning which may decrease the performance for the other workloads.

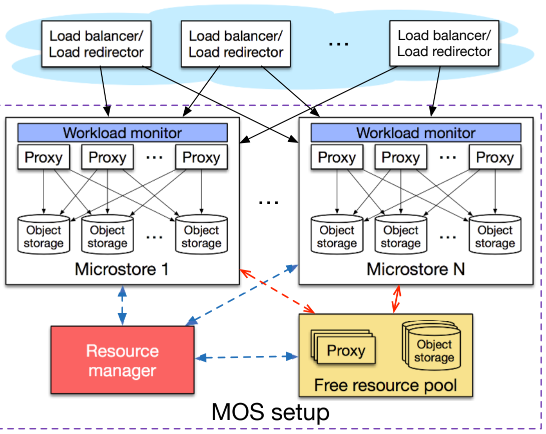

Figure 1: Overview of MOS design.

In this paper, we argue that *it is more beneficial to separately entertain these workloads in finer-grained object stores launched on sub-clusters formed using the available hardware resources*.

2. MOS Design Overview

MOS performs dynamic resource partitioning and provisioning, allowing each microstore within an object storage setup to run as a fully-functional object store unit. As depicted in Figure 1, MOS consists of two layers: (1) **Microstores:** consists of multiple instances of object stores, each called a *microstore* that is allocated a subset of proxy and storage nodes that matches the requirements of application it is meant to support. (2) **MOS substrate:** consists of a resource manager that monitors load on each microstore using a workload monitor and automatically reconfigures resources assigned to microstore to cope with workload shifts.

References

[1] Amazon s3. http://aws.amazon.com/s3/.

[2] Ceph. http://ceph.com/.

[3] Openstack swift. http://docs.openstack.org/developer/swift/.