Wonderful : A Terrific Application and Fascinating Paper

Your N. Here
Your Institution

Second Name
Second Institution

Abstract

Your Abstract Text Goes Here. Just a few facts. Whet our appetites.

1 Introduction

As large-scale unified storage systems evolve to meet the requirements of next-generation hardware and an increasingly diverse set of applications, *de jure* approaches of the past—based on standardized interfaces—are giving way domain-specific interfaces and optimizations. While promising, current approaches to co-design are based on ad-hoc strategies that are untenable.

fixed storage apis force applications to use external data management, or duplicate complex error prone processes when the system doesnt meet their needs. fear of vendor lock-in is beginning to subside with open-source. by exposing internal services applications can compose existing services to support application requirements. also we may see entirely new sotrage system based on apsecific cases, often based on consitency requirements.

data is the most critical component, but interfaces are just as critical because they define access. how do we manage this trend?

The design space is large, and also quite different than traditional database optimization techniques handle.

be very large, and it can be difficult to choose a design that future proofs an implementation against hardware and software upgrades

its hard because the design space is quite large. afterall, how long have we been dealing with blocks and files. the reason we wanted to move away from standard storage apis in the first place is because the storage system wants to evolve. so we cant advocate a fixed api. domain specific knowledge of both applications and storage systems is needed then to optimize any particular instance of co-design.

Malacology [1] is a recently proposed *programmable* storage system that exposes common sub-systems found in distributed storage systems for reuse by applications, avoiding duplication of complex error-prone services.

Malacology demonstrates a set of prinicples for exposing services, and demonstrates with real world examples. While the exact form of these services is not well-defined, we argue in this paper that in order for storage systems to continue to evolve, managing interface change must be a fundamental component of programmable storage.

Declarative specification is the way to go here.

2 The Programmable Storage Challenge

When application requirements are not met by an underlying storage system the most common approach is to design a workaround that will fall roughly into one of three categories:

Extra services. 'Bolt-on" services are intended to improve performance or enable a feature, but come at the expensive of additional sub-systems and dependencies that the application must manage, as well as trust.

Application changes. The second approach to adapting to a storage system deficiency is to change the application itself by adding more data management intelligence into the application or as domain-specific middleware. When application changes depend on non-standard semantics exposed by the storage system the coupling that results can be fragile and result in lock-in.

Storage modifications. When these two approaches fail to meet an application's needs, developers may turn their attention to any number of heavy-weight solutions ranging from changing the storage system itself, up to and including designing entirely new systems. This approach requires significant domain knowledge, and extreme care when altering code-hardened systems.

2.1 Programmable Storage

Malacology is a recently proposed approach that advocates a design strategy called programmable storage in which existing storage system services are safely exposed such that they can be composed to form application specific services. Figure 1 shows the architecture of Malacology as implemented in Ceph, which exposes a variety of low-level internal services such object interfaces, and cluster metadata management that are in-turn composed to form a set of new system services that support the requirements of multiple real-world applications.

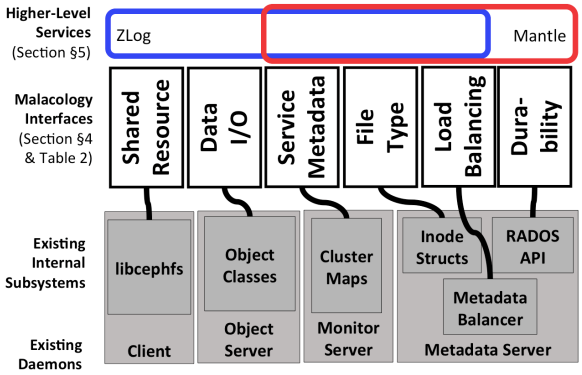

Figure 1: Malacology implementation in Ceph.

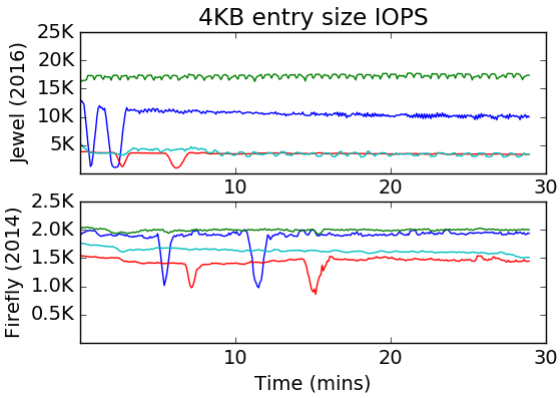

Figure 2: asdf

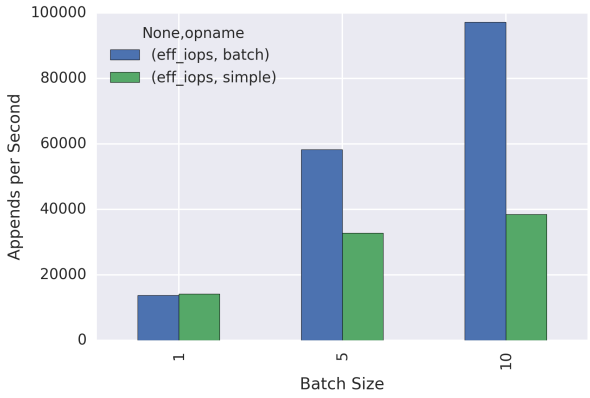

Figure 3: asdf

2.2 Everyone Loves A Standard

The narrow interface exposed by storage systems has been a boon in allow systems and applications to evolve independently, in affect limiting the size of the design space where applications couple with storage. Programmable storage lifts the veil the system, and with it, forces applications developers to confront a large set of possible designs.

To illustrate this challenge we implemented as an object interface the CORFU storage device specification, that is a write-once interface over a 64-bit address space. The interface is used as a building block of the CORFU protocol to read and write log entries that are striped over an entire cluster. The implementations differ in their optimization strategy of utilizing internal system interfaces. For instance one implementation uses a key-value interface to manage the address space index and entry data, while another implementation stores the entry data using a byte-addressable interface.

Figure 2 shows the append throughput of four such implementations run on two versions of Ceph from 2014 and 2016. The first observation to be made is that performance in general is significantly better in the newer version of Ceph. However, what is interesting is the relationship between the implementations. Run on a version of Ceph from 2014, the top two implementations perform with nearly identical throughput, but have strikingly different implementation complexities. The performance of the same implementations on a newer version of Ceph illustrate a challenge: given a reasonable choice of a simpler implementation in 2014, a storage interface will perform worse in 2016, requiring significant rework of low-level interface implementations.

3 Design Space

4 Programming Model

5 Additional Optimizations

References

[1] Sevilla, M., Watkins, N., Jimenez, I., Alvaro, P., Finkelstein, S., LeFevre, J., and Maltzahn, C. Malacology: A programmable storage system. In *Eurosys 2017*. To Appear.

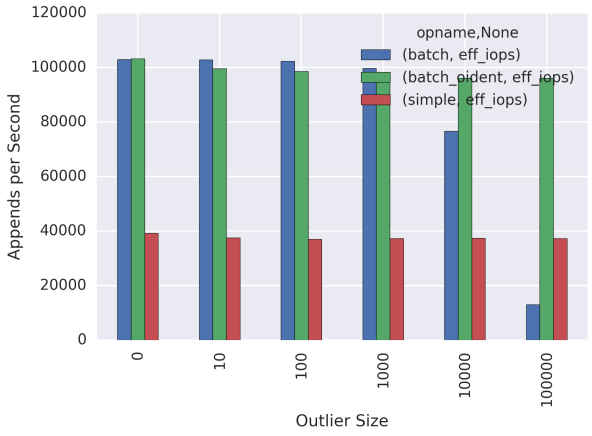

Figure 4: asdf