
**Scalable Computing Software Laboratory**

**Technical Report**

**Department of Computer Science**

**Illinois Institute of Technology**

# Towards a Unified Data Access System: Mapping Files to Objects

Anthony Kougkas, Hariharan Devarajan, Xian-He Sun

Illinois Institute of Technology, Department of Computer Science

{akougkas, hdevarajan}@hawk.iit.edu, sun@iit.edu

February 2017

Technical Report No. IIT/CS-SCS2017-1

http://cs.iit.edu

10 West 31st Street, Chicago, IL 60616

# Towards a Unified Data Access System: Mapping Files to Objects

*Anthony Kougkas, Hariharan Devarajan, Xian-He Sun*

*Illinois Institute of Technology, Department of Computer Science*

*{akougkas, hdevarajan}@hawk.iit.edu, sun@iit.edu*

**Abstract**: File and block storage are well-defined concepts in computing and have been used as common components ofcomputer systems for decades. Big data has led to new types of storage. A highly successful newly emerged type of storage is object storage. However, object storage and traditional file storage are designed for different purpose and for different applications. Many applications need to access data from both types of storage. In this paper, we first explore the key differences between object storage and the more traditional storage systems. We also provide several efficient techniques to map user's file structures to an underlying object store. Our evaluation shows that by achieving an efficient such mapping, our library can grant almost 2x higher performance against a simple file-to-object mapping and with mapping overheads as low as around 5%.

**Keywords**: File Systems, Object Storage, Integrated access, Unified storage, Mapping files to objects

## 1. Introduction

Historically, data are stored and accessed as files, blocks or objects in equivalent storage systems. File and block storage have been around for considerably longer than object storage, and are something most people are familiar with. These systems have been developed and highly optimized through the years. Popular interfaces and standards such as POSIX I/O, MPI-IO, and HDF5 expose data to the applications and allow users to interact with the underlying file system through extensive APIs. In a large scale environment the underlying file system is usually a parallel file system (PFS) with Lustre, GPFS, PVFS2 being some popular examples or a distributed file system such as GoogleFS or HDFS. However, applications are increasingly dealing with high volume, velocity, and variety of data which leads to an explosion of storage requirements and increased data management complexity. Most of these file systems face significant challenges in performance, scalability, complexity, and limited metadata services.

On the other hand, object storage was born from the need to increase the scalability and programmatic accessibility of storing data. It is widely popular in the cloud community and there are a lot of different implementations freely available. It offers simplistic APIs with basic get(), put(), and delete() operations. Most notable examples include the Amazon S3 and the OpenStack Swift API. So far, object storage has been used widely for stale, archival data, which fits nicely with the fact that changes are accommodated by creation of new versions of data, rather than modifying existing data. However, this seems to be changing and we see more high-performance and low latency solutions. Few examples include Cassandra, MongoDB, and HyperDex. The flat name space organization of the data in object storage, in combination with its expandable metadata functionality, facilitate its ease of use, its scalability, and its resiliency via replicated objects. Lastly, object stores are the best option to store, access, and process unstructured and semi-structured data making them a widely used storage solution for Big Data problems.

In this paper, we explore several ways to map files to objects. Specifically, we designed and implemented three new mappings of a typical POSIX file to one or more objects. These mappings can cover MPI-IO as well since it is using POSIX files at its core. We also implemented a novel mapping of an HDF5 file to one or more objects. Our mappings pave the way towards a unified storage access

system. Using our mappings one can efficiently utilize a file interface to access and process data that reside to a totally different storage subsystem such as an object store. With this extra functionality, we can leverage strengths of each storage solution and complement each other for known limitations. This is a powerful abstraction for the user who, under our solution, still uses the familiar file interface while a scalable and efficient object store supports all I/O operations.