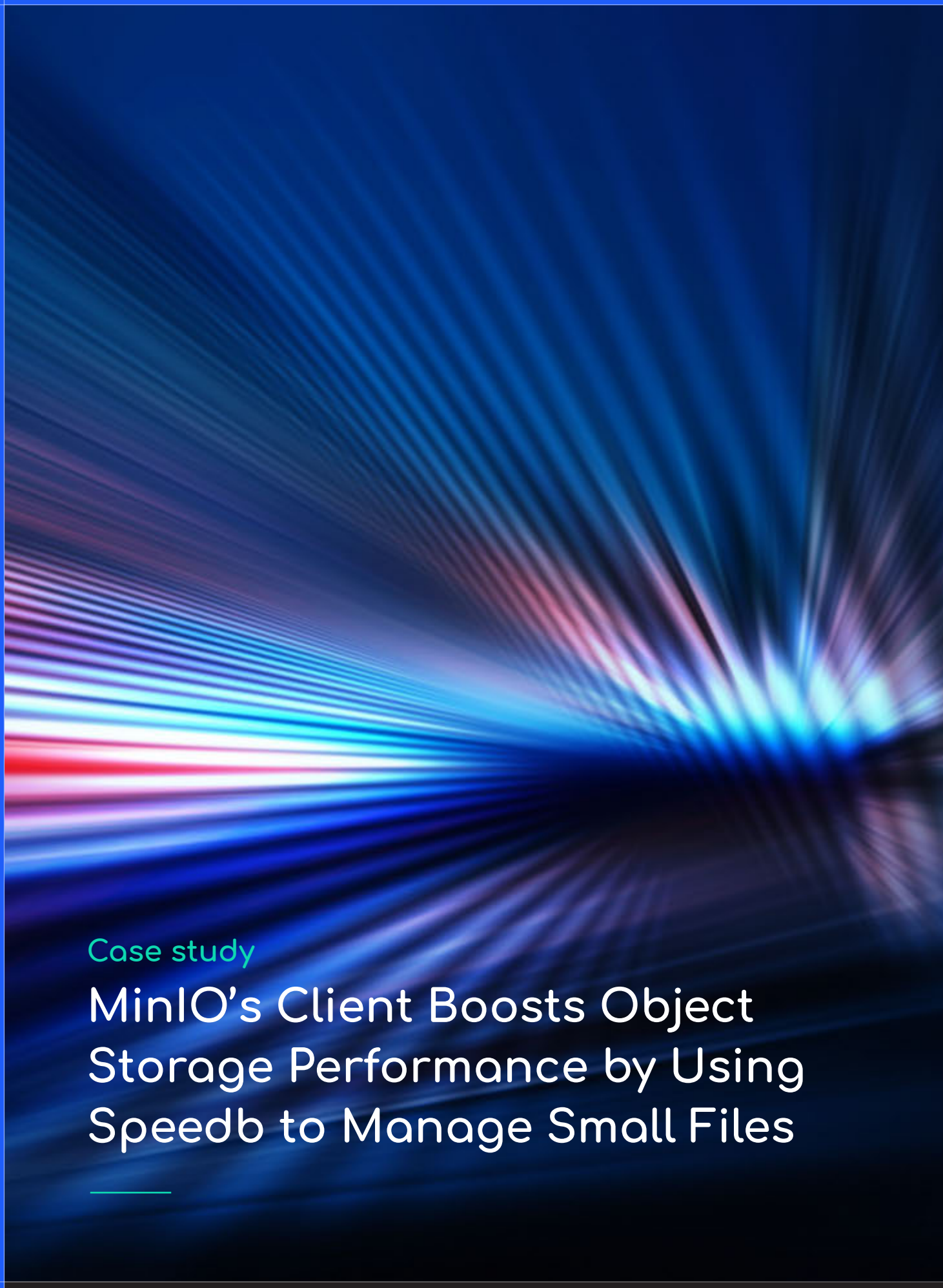


Case study

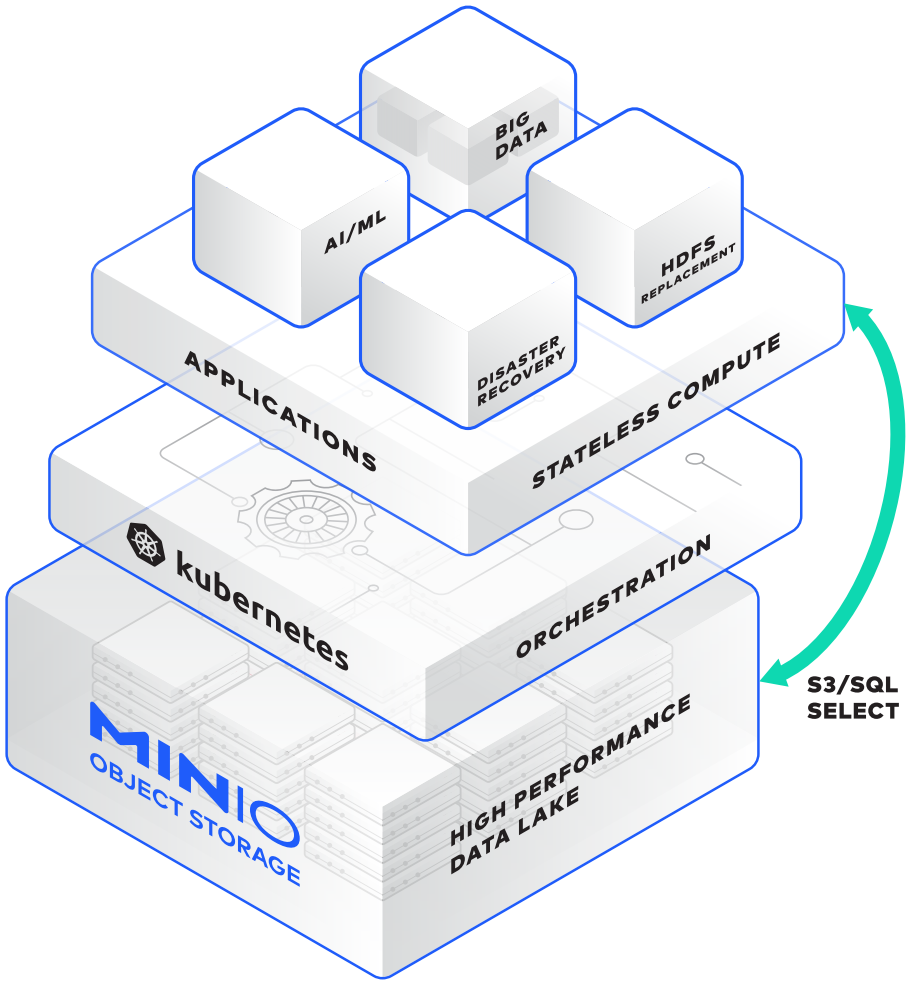
# MinIO's Client Boosts Object Storage Performance by Using Speedb to Manage Small Files

# Background

MinIO is a provider of high-performance object storage that's compatible with Amazon's S3 cloud storage service. Designed to serve only objects, MinIO offers a range of functionalities that support various object storage use cases, including secondary storage, disaster recovery, and archiving.

In addition, MinIO is increasingly being used as primary storage for cloud-native applications that require higher throughput and lower latency, e.g., real-time stream processing, IoT, and AI testing and training. These use cases are driving unprecedented growth in the number of data objects, resulting in significant performance and scalability challenges.

One of MinIO's clients, a large US-based data storage company, was looking for a way to improve the performance of its cloud object storage platform. Powered by MinIO, the platform is aimed at modern use cases that generate large volumes of unstructured data, enabling users to store different types of data including log files and metadata.



# Challenge

Traditionally, object storage systems were designed to handle large objects that are infrequentlyaccessed, e.g., when restoring data from backup in the case of a system failure or a disaster. Driven by the rapid adoption of modern cloud-native applications and rich media content, emerging object storage use cases have led to new patterns of data access.

Most notably, enterprises are increasingly dealing with huge amounts of unstructured data in the form of objects such as images, video, and audio. This increase in the volume of objects has been accompanied by an explosion of metadata, which is commonly used to quickly find data files by assigning and then identifying them with certain properties.

Modern workloads such as stream processing, AI/ML, and IoT tend to generate large volumes of small objects that may only be a few bytes large. However, these objects often hold metadata that's the same size or larger than the object itself. These small objects - and their associated metadata - are accessed much more frequently compared to traditional object storage use cases.

The expansion of metadata creates a significant challenge for legacy object storage systems due to their underlying data architecture, which was designed for "write once read many" (WORM) scenarios. As the number of small objects continues to increase, object storage systems (such as MinIO) are struggling to effectively store and manage them.

MinIO stores files in an underlying POSIX filesystem. Filesystems read and write data randomly, i.e., in no particular order. This means that when a new object file comes in, it's immediately written to a random free place on the disk.

Each file on the filesystem is represented by an inode that contains the file metadata, including a unique number used by the OS to identify the file. Hence, for each operation on the file, e.g., write, read and update, the filesystem must first access the metadata about the file that describes where it resides. And since the metadata is randomly scattered across the disk, the filesystem may not be able to locate it quickly, resulting in significant I/O overhead and latency.

This process must be carried out for every operation on the data, regardless of the file size. Consequently, when dealing with large numbers of small objects, the system may experience significant performance degradation due to the need to repeat the process multiple times. This challenge is exacerbated in modern use cases such as IoT and real-time analytics, which generate large amounts of frequently accessed objects such as event logs. This further increases the number of data operations, stressing the filesystem even more.

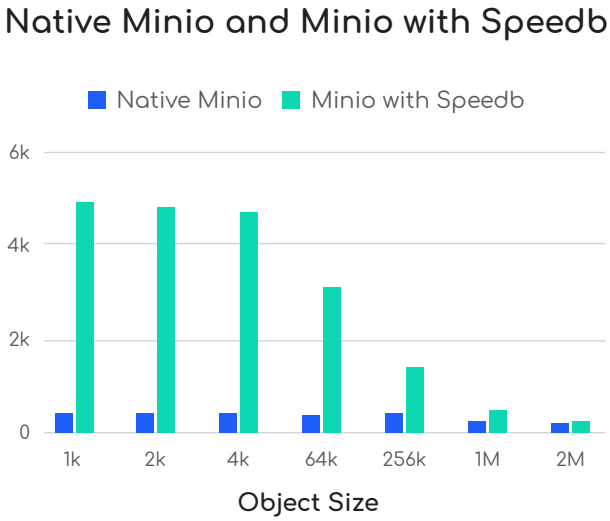
# Solution

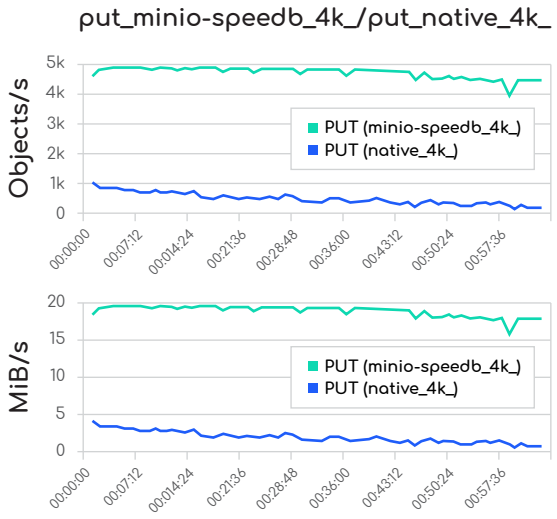
MinIO's client started experiencing performance issues with the increase in the number of small objects handled by MinIO's underlying filesystem. To address this challenge, Speedb implemented a plug-in that automatically captures and sends small objects to the Speedb data engine instead of writing them directly to the filesystem disk.

Speedb, an embedded key-value store (i.e. storage/data engine), is designed to support data-intensive workloads with high scalability and performance without compromising storage capacity. Unlike filesystems, Speedb doesn't execute reads and writes one by one.

Instead, it employs innovative compaction, flow control, and indexing methods to sort the metadata in bulk and write all the data in one go, thus significantly reducing the performance overhead. Without making any changes to the existing code and underlying infrastructure stack, Speedb replaced MinIO's underlying filesystem for small objects that are less than 2MB in size, while larger objects are still stored as files.

The results were immediate. For example, when executing a put command (used for updating an object), Speedb was able to handle more than 5000 objects per second for objects with a size of 1kb compared to less than 500 objects per second with MinIO's native filesystem. This ratio of 1:10 was relatively constant for files that are 4kb in size or smaller. Significant performance benefits were achieved for larger files as well. For example, Speedb was able to handle nearly 3000 objects per second when the object size was 64Kb, compared to 540 objects per second with MinIO's native filesystem.



The implementation of Speedb's data engine enabled MinIO's client to overcome the performance challenges that hindered its ability to effectively support data-intensive cloud-native workloads. By offering high-performance object storage at scale, the client is now well positioned to pursue strategic growth opportunities around big data analytics, multi cloud, AI/ML and more.