MINIO

Hadoop Migration Guide

WHITE PAPER

Hadoop Migration Guide

This paper is written for technical leaders who are interested in using object storage to replace Hadoop HDFS and want to understand its commutability, benefits and challenges. The typical reader will have a general understanding of object storage, perhaps through implementations such as Amazon S3 or MinIO, and HDFS experience with Cloudera or MapR, but will be looking for further details to ensure a smooth migration.

The following topics cover critical points for planning and deploying a migration from HDFS to Object Storage:

- Architectural comparison of HDFS/MapRFS and object storage, and how they can be commutable in solution architectures
- Key technical considerations in migrating applications and data analytics workloads from HDFS to object storage

For the convenience of the reader this document starts with a distilled list of the benefits of object storage over HDFS / MapRFS. The main section of the document provides additional detail on the similarities and differences between HDFS, MapRFS and object storage, how they might be taken advantage of in a migration plan, and what operational considerations these differences might invite.

Benefit Overview

Even for workloads traditionally associated with Hadoop HDFS, object storage provides improvements in performance-per-dollar, durability, consistency, and infrastructure overhead. In addition, object storage offers opportunities for performance improvements with the migration to a disaggregated architecture. The following summarizes some of the advantages that object storage has over Hadoop HDFS:

- **Flexible Accommodation:** Users and business groups are freed from cluster multi-tenancy when task specific clusters can live only when needed, and scale out according to compute demands.
- **Reduced Compute Overhead:** Compute resources are no longer devoted to a specific cluster, and can be shared among an entire enterprise. Predicate pushdown for SELECT clauses save network and compute requirements on the SQL engines and client applications.
- **Reduced Storage Overhead:** More efficient redundancy techniques require less overhead.
- **Durability:** Strict consistency and elimination of master node point of failure.

Migration Considerations

To inform enterprises preparing to move from HDFS / MapRFS to object storage, this paper will address disaggregation at an architectural level and then proceed into the changes one can expect with administration, pipeline development and data consumption in the object storage architecture.

Disaggregation

The resource boundaries that define and enclose a Hadoop cluster continue to be an operational legacy for YARN and HDFS today. Replacing HDFS with object storage is a natural fit when considering a disaggregated compute infrastructure managed with an orchestration platform like Kubernetes. A discussion of new tool options and migration paths is given in the technical section.

Flexible accommodation and reduced overhead through disaggregation can be achieved as follows:

- **Disaggregation:** Separation of the compute infrastructure from the storage infrastructure of a cluster into discreetly managed identities.
- **Independant Scalability / Resource Reduction:** Compute infrastructure can be sized to support the needs of high performance analytics tools, and the storage can be sized independently to support the storage needs of the data lake.
- **Container Support:** Hadoop HDFS's aggregated architecture does not support containerization, blended storage or dense compute in a way that makes sense in a modern, microservices architecture.
- **Independent Tiering:** With an architecture where compute and storage are separated, the architect gains the ability to create tiers of data with different performance and cost profiles.
- **Kubernetes vs YARN:** Modern architectures can manage the resources of the infrastructure as a whole rather than using YARN to share the resources of a fixed Hadoop cluster.

While disaggregation is strongly recommended given the above benefits, the following sections also describes benefits when using an object store as a part of a Hadoop infrastructure as well.

Impact on Hadoop Users and Administrators

As one migrates Hadoop infrastructure either onto object storage, or replaces Hadoop with Kubernetes orchestration, the following considerations need to be made for data consumers, administrators and ETL developers.

File migration will be given the most discussion given it represents the majority of volume. This section will also briefly discuss the chief considerations for those running databases (e.g. Hive and Hadoop) on HDFS and MapRFS.

Data Consumption

Many applications and users will consume data through REST queries. Object Storage and WebHDFS expose very similar REST APIs, which address each object with a unique URL. Object storage implementations such as Amazon S3 and MinIO provide the ability to run SQL-like Select statements reducing the overall network and client-side resource demand.

ETL Development

With the large adoption of Amazon Cloud and its object store 'S3', many data processing tools - even Hadoop tools - have developed full featured compatibility with object storage.

ETL tools provide the option to designate HDFS or S3 in their connection URL - leveling the playing field for data transformation and ingestion.

Even so, there are opportunities for tuning that are unique to each platform. A discussion of these opportunities, including choice of connectors and committers will be discussed in the "Technical: Object Storage Tuning for Hadoop and Hive" portion of this document.

Command Line Administration

MapR implements most of the POSIX standard, meaning that native file system commands and applications will work when mounted via FUSE or NFS. In addition, MapR also requires a special command line utility to administer features unique to MapR.

HDFS and object stores do not use native file system commands, and instead are completely dependent on special command line utilities (e.g. 'hadoop fs' and 's3cmd'). However, these utilities provide a syntax that will be familiar to those used to native commands, reducing the friction for a migration.

Database Administration

Perhaps the most salient display of commutability between HDFS and object stores is the ability to run dynamic databases like HBase or the merge functionality in Hive directly on an object store.

A larger treatment of Hive tuning and database alternatives are contained in the "Technical: Object Storage Tuning for Hadoop and Hive" section.

Cluster Management and Administration

In order to make their platforms less error prone and easier to maintain, neither Hadoop HDFS nor object storage support the ability to edit files or objects. This stems from the common belief that it is easier to work around file editing at the application level than to implement the infrastructure needed for editing at the platform level. For MapRFS users, where editing is allowed, the same application work arounds developed for HDFS and object storage will apply here just as well.

Replication and Resiliency

Since its inception, the standard best practice for top tier speed and reliability with HDFS means that data should be replicated three times. Following that advice means growing storage by a factor of three. While Amazon S3's is opaque when it comes to what technology they use for durability guarantees, modern on-premise object storage systems like MinIO employ sophisticated erasure coding. Erasure coding can reduce the storage growth to a factor of 1.5, while providing higher levels of durability than HDFS.[1]

[1] At the time of this writing, HDFS erasure coding is still in the early stages of operational adoption. However, it is a popular feature request that will likely be quickly adopted, and is yet another way that Hadoop is evolving to be more like object storage.

Along with erasure coding, MinIO provides strict write consistency to ensure that changes to the data are visible to all clients at the same time. Other S3 object storage providers such as Amazon S3, and configurations of HDFS only provide eventual consistency. This means that changes made by one application are not immediately visible to other clients.

Technical Considerations

At this point the reader should be familiar with the overall landscape of choices. In this section a more indepth treatment is provided to evaluate a short term and long term migration strategy. The following technical considerations are offered to help evaluate a spectrum of migration options from continuing to support certain legacy Hadoop applications to architecting a modern, fully disaggregated microservices powered software stack.

Tools for a Disaggregated Architecture

Containerized microservices are managed by service orchestration infrastructures like Kubernetes. They are the key to realizing the compute-on-demand functionality of server disaggregation. For workloads typical to Hadoop, this orchestration allows enterprises to quickly ramp up the instances of analytics applications to match spikes in demand.

Cloud service providers, such as AWS, GCP or Azure, have native object storage that is very well suited to serve orchestrated applications. To achieve similar objectives in the private cloud, however, enterprises will need to adopt components that employ the same cloud native technologies. Modern object storage solutions (MinIO and others) offer cloud-native approaches that are purpose built for containerization and Kubernetes orchestration.

Object Storage Tuning for Hadoop and Hive

There are subtle differences in how object storage treats file operations. These differences can impact performance. In object stores, creating directories, even deeply nested directories, is a single rather than a recursive operation. On the other hand, in HDFS moving a file or directory can be achieved as a single rename operation where it requires a copy / delete operation in object storage.

Choosing a committer that understands these implementation details can have a material impact on performance. Amazon has helped produce the S3A committers for the purpose of tuning the staging and output of Hadoop jobs to object storage.

Object stores like Amazon S3 and MinIO support S3 Select, a predicate pushdown of the SELECT statement directly in the HTTP GET request. While S3 Select will benefit a range of use cases, it should provide performance improvements for legacy Hadoop query engines like Hive.

Object Storage Tuning for Spark

Databricks recommends the DBIO committer for Spark processes both for speed and the ability to complete output in discrete transactions.

As noted above, Spark can also utilize S3 Select on compatible object stores such as Amazon S3 and MinIO.

Other Cloud Native Data Processing Alternatives to Hadoop

Applications that run in a disaggregated cluster will be stateless, they will be containerized and they will not depend on HDFS for storage. The shift to elastic computing means enterprises can adopt modern data analytics applications that are purpose built to work in an elastic computing infrastructure.

The following popular data analytics and machine learning applications are not native to the official Hadoop ecosystem but are compatible. All of the applications on the list also support object storage but with additional benefit of running in containers which in turn are orchestrated with Kubernetes:

- H2O.ai for Machine Learning and Data Science
- Presto, or Dremio for SQL queries on large datasets
- Spark or Flink for data processing
- Splunk for Log Analytics
- Tensorflow for Machine Learning and Model Deployment
- Teradata for Business Analytics

For those enterprises migrating away from Hadoop entirely, the following recommendations are offered.

Workload	Legacy Hadoop	Alternative
SQL Query Engine	Hive	Presto
Data Processing	Spark	Spark
Business Analytics	Impala	Teradata
Log Processing	Hunk	Splunk
Distributed Keystore	HBase/MapR-DB Binary	Redis
Time Series	TSDB	TICK Stack
JSON Database	MapR-DB JSON	MongoDB

Enterprises will doubtless have other applications running on Hadoop HDFS with varying importance to the organization. This document has not touched on data governance, data vaulting, or access management. Each will involve detailed planning outside of the scope of this document. Insofar as those application interface with Hive or other Hadoop utilities, however, they should have the same utility on either an object store or on HDFS.

Identity Management and Encryption

There are three aspects to securing an analytics cluster - controlling access, securing data and auditing who did what to the system.

Controlling access is accomplished by identifying, authenticating, and authorizing users and applications. Data encryption is used to secure data in transit and at rest. Audit logs are used to track the actions of users and applications that access the resources of the cluster.

The following sections discuss these security topics and how they apply to both Hadoop/HDFS and object storage.

Authentication - Validating clients (both users and applications) is the purpose of authentication. The most widely used authentication service is Kerberos. Kerberos works equally well on both Hadoop and MinIO or other object storage solutions and there should be minimal friction in the migration.

Authorization - Once a client is authenticated it must also be authorized to access individual files or objects. Hadoop/HDFS uses a POSIX-style read, write, and execute permissions model. Modern object storage follows the Amazon model of using policies to determine read and write access to buckets and objects. The object storage approach is generally considered to be superior from a security perspective, but there will be work required to adopt the new model. The POSIX approach will not work on object storage.

Encryption - To properly secure data it must be encrypted in transit and at rest. TLS or SSL is used to encrypt data in transit. While commercial vendors of Hadoop like Cloudera have built specific modules to address encryption, the object storage model is superior in a number of ways that are beyond the scope of this paper. There will be no incremental work associated with encryption if an enterprise is migrating, these features (client side and server-side encryption) will be part of the object storage package. More importantly, in well designed systems like MinIO, the performance overhead associated with encryption is negligible, enabling it to run as a default setting.

Auditing - audit logs are used to provide a historical view of the users and applications who attempt to access resources on the cluster. Both HDFS and object storage have robust logging functionality and there would be minimal to no work associated with a migration when it comes to auditing.

Tools for Migrating Data from HDFS to S3

Several tools can be used to copy or mirror data from Hadoop/HDFS to object storage.

- DistCp (distributed copy) - a tool used to copy data between Hadoop clusters, and between Hadoop and S3 compatible object storage. DistCp uses MapReduce to implement its distribution, error handling, and reporting. It expands a list of files and directories into map tasks, each of which copies a partition of the files specified in the source list. The DistCp command is more efficient than the Hadoop 'fs -cp' operation.

- MinIO Client (mc) - a modern alternative to UNIX commands like ls, cat, cp, mirror, diff, find etc. The 'mc mirror' command is used to mirror data between S3 compatible object stores and from Hadoop/HDFS to any S3 compatible object store.

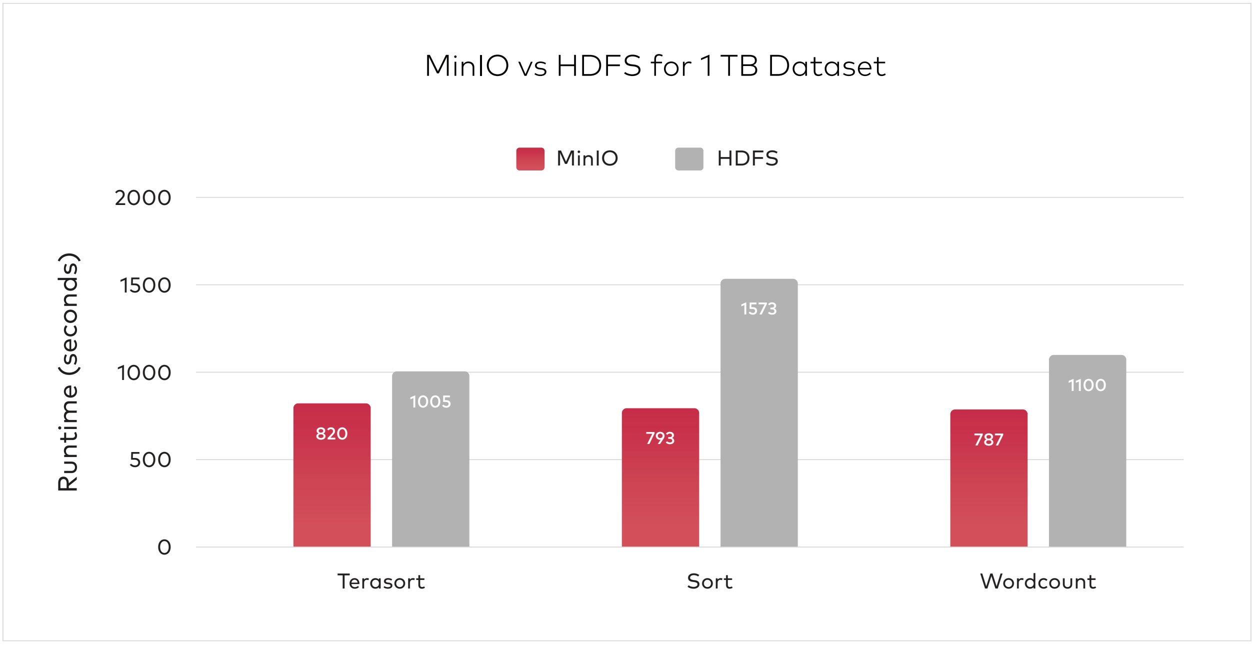
Modern, High Performance Object Storage from MinIO

MinIO is a pioneer in the development of modern, high-performance object storage. It is different in that it was designed from its inception to be the standard in private cloud object storage. Because MinIO is purpose-built to serve only objects, a single-layer architecture achieves all of the necessary functionality without compromise. The result is a cloud-native object server that is simultaneously performant, scalable and lightweight.

Philosophically MinIO is guided by a few core principles:

Disaggregation. At the heart of the hyper-scale revolution is the concept that compute and storage should be separate and best of breed. This architecture leads to better performance, capacity utilization and I/O while simultaneously lowering space, cooling and power requirements. MinIO is committed to building the best, private-cloud, software-defined object storage system on the planet.

Performance. With its focus on high performance, MinIO enables enterprises to support multiple use cases with the same platform. For example, MinIO's performance characteristics mean that you can run multiple Spark, Presto, and Hive queries to quickly test, train and deploy AI algorithms, without suffering a storage bottleneck. More importantly, MinIO is the only private cloud object storage solution capable of delivering Hadoop-like performance.



smaller values indicate higher performance

Scalability. MinIO leverages the hard won knowledge of the web scalers to bring a simple scaling model to object storage. This accompanies a strongly held belief that "simple things scale." At MinIO scaling starts with a single cluster which is then federated with other MinIO clusters to create a global namespace, spanning multiple data centers if needed. Expansion of the namespace is possible by adding more clusters, more racks until the goal is achieved.

Simplicity. Minimalism is a guiding design principal at MinIO. Simplicity reduces opportunities for errors, improves uptime, delivers reliability while serving as the foundation for performance. MinIO can be installed and configured within minutes simply by downloading a single binary and then executing. The amount of configuration options and variations is kept to a minimum which results in near-zero system administration tasks and few paths to failure. Upgrading MinIO is done with a single command which is non-disruptive and incurs zero downtime - lowering total cost of ownership.

Amazon S3 Compatibility. Amazon's S3 API is the defacto standard in the object storage world and represents the most modern storage API in the market. MinIO adopted S3 compatibility early on and was the first to extend it to support S3 Select. MinIO supports the S3A connector as well.

Freedom. MinIO is 100% open source under the Apache V2 license. This means that MinIO's customers are free from lock in, free to inspect, free to innovate, free to modify and free to redistribute. Indeed, MinIO powers product offerings from multiple Fortune 500 organizations. The purity of MinIO's licesencing approach and the quality of its subscription support offering have made the company the fastest growing, private cloud object storage system in the world.

As a result of these choices, the software scales infinitely and performs at the top end of the hardware stack - making MinIO an ideal S3-compatible replacement for Hadoop HDFS for machine learning, artificial intelligence and other big data workloads.

To learn more about MinIO feel free to reach out to us at hello@min.io.