Exploration and Application of Distributed Object Storage in Converged Media Platforms for News Gathering and Editing: Postprint

Authors: Gao Wen, Ren Baiqing, Wei Haitao, Zhou Chenggeng

Date: 2023-10-08T00:00:00+00:00

Abstract

【Purpose】The rapid development of converged media has significantly propelled the creation and sharing of multimedia content. Meanwhile, with the maturation of 4K, 5G, and AI technologies, files such as photos, music, and videos in manuscripts uploaded and archived by Xinhua News Agency users are increasing daily, exacerbating the explosion of data volume and continuously consuming storage resources.

【Methods】For the storage and management of such unstructured data, traditional block storage (SAN) and file storage (NAS) prove somewhat inadequate due to their inherent technological and architectural limitations, and object storage has emerged as a solution. Based on this, this paper primarily presents the evolution of data storage, as well as the characteristics and advantages of distributed object storage, and explores the feasibility of applying it within the business context of Xinhua News Agency' s editorial converged media platform.

【Results】【Conclusion】By constructing a MinIO object storage environment and transforming the file service interface of the editorial converged media platform, a smooth transition to object storage can be achieved, validating the feasibility of applying object storage in the editorial converged media platform.

Full Text

Preamble

ChinaXiv Collaborative Journal: Exploration and Application of Distributed Object Storage in Editorial Converged Media Platforms

Gao Wen, Ren Baiqing, Wei Haitao, Zhou Chenggeng
(Communication Technology Bureau, Xinhua News Agency, Beijing 100803)

Abstract

Purpose: The rapid development of converged media has significantly accelerated the creation and sharing of multimedia content. Concurrently, with the maturation of 4K, 5G, and AI technologies, Xinhua News Agency has experienced exponential growth in user-uploaded and archived materials—including photos, audio, and video files—intensifying data volume explosions and continuously consuming storage resources.

Method: Traditional block storage (SAN) and file storage (NAS) have proven inadequate for storing and managing such unstructured data due to inherent technological and architectural limitations, giving rise to object storage as a viable solution. This paper introduces the evolution of data storage technologies, examines the characteristics and advantages of distributed object storage, and explores its feasibility for integration with Xinhua' s editorial converged media platform operations.

Results and Conclusion: By deploying a MinIO object storage environment and refactoring the file service interfaces of the editorial converged media platform, a seamless transition to object storage can be achieved, thereby validating its feasibility for editorial converged media applications.

Keywords: distributed; object storage; converged media; MinIO; editorial

CLC Number: TP311
Document Code: A
Article ID: 1671-0134 (2023) 06-104-05
DOI: 10.19483/j.cnki.11-4653/n.2023.06.022
Citation Format: Gao Wen, Ren Baiqing, Wei Haitao, Zhou Chenggeng. Exploration and Application of Distributed Object Storage in Editorial Converged Media Platforms [J]. China Media Technology, 2023(06): 104-107, 158.

Introduction

In recent years, as we have entered the digital economy era, the proliferation of massive applications has triggered an explosion of data, with data types becoming increasingly diverse and complex. According to International Data Corporation (IDC) forecasts, global data volume is expected to reach 175 zettabytes by 2025. Moreover, the nature of data itself is changing; with deeper technological integration, unstructured data—including video, audio, and images—is growing even more rapidly.

Faced with these dual trends of explosive data growth and the increasing proportion of unstructured data, traditional storage systems such as DAS, SAN, and NAS have proven unable to effectively cope due to limitations in their underlying technology and architecture. This has created demand for a fundamentally new storage architecture with extreme scalability to meet requirements ranging from terabyte to exabyte-scale storage capacity. Consequently, object storage has emerged as the solution.

Against this backdrop of deep media integration, traditional media outlets face numerous challenges but also opportunities for transformation driven by new technologies. As a leading force in public opinion guidance, Xinhua News Agency has achieved significant progress and notable results in media convergence, launching a series of converged media reporting products. According to Xinhua News Agency' s Daily Telegraph, during the 2021 Two Sessions, the domestic news department achieved a "complete shift of the main force to the main battlefield"through full-chain planning and multimedia collection, while innovating thinking and leveraging strengths to compete creatively. This resulted in a "reversal" where the ratio of converged media products to traditional wire copy shifted from 1:1 in 2020 to over 2:1.

During the transition from traditional to new media, the storage of large-volume, numerous unstructured media files poses a severe challenge. For media organizations, data and content remain core assets. Throughout the entire workflow—from news material collection, editing, processing, and management to distribution and final review—consideration must be given to enabling multi-user, multi-system sharing, supporting multi-business processes, and ultimately meeting diverse requirements for converged media product creation, circulation, review, backup and archiving, data analysis, and AI applications.

1. Evolution of Data Storage

With the development of information technology and the internet, storage systems have continuously evolved, offering increasingly rich functionality, greater capacity, and enhanced performance. Broadly speaking, based on differences in storage functionality and usage patterns, storage can be categorized into three types.

1.1 Block Storage

Block storage combines multiple disks through a controller into one or more logical disks, which are then presented to hosts as block storage devices. On the host side, these appear as one or more hard drives that typically require partitioning and file system formatting by the host operating system before use. Block storage directly connected to hosts via data cables is called DAS (Direct-Attached Storage), while block storage accessed through Fibre Channel or IP networks is called SAN (Storage Area Network). Block storage suffers from poor scalability and flexibility.

1.2 File Storage

File storage adds a file system layer on top of block storage, allowing remote mounting of the file system over a network. On the host side, this appears as a file system partition that can be used directly. Applications need not concern themselves with differences between various file storage systems; like local file systems, they operate on directories and files through the operating

system' s POSIX standard file system interface. File systems organize files in a hierarchical tree structure, facilitating file lookup. Users can open and modify files through operating system applications—for example, opening and editing Word documents. File storage offers rich functionality and is well-suited for storing various unstructured files such as documents, images, and videos.

1.3 Object Storage

Object storage originated as a cloud storage product offered by public cloud providers, delivering distributed object storage services through Web interfaces in a Key-Value format. Any data can be converted into a byte stream assigned as a Value, along with a string-type Key, which are then submitted together to the object storage system for preservation. During retrieval, only the Key is needed to access the corresponding Value. Object storage functions more like a web application than traditional storage and is commonly used for storing images, videos, and other files to address massive internet file storage and user access requirements.

Typically, usage patterns simulate local file system conventions by using file paths (including directory paths) as Keys and converting file data into byte streams as Values, indirectly achieving file storage. More notably, following object storage rules, the Key can be embedded into a specific URL, enabling direct browser access—an extremely convenient feature for images or videos. Consequently, the primary use case for object storage is storing static content (videos, images, files, software installation packages, etc.) for websites and mobile applications.

Compared to traditional storage, object storage offers numerous innovative features, with the following being particularly noteworthy:

(1) **Versioning**: Every modification to an object generates a new version, allowing users to retrieve any historical version. It also provides object locking and WORM (Write Once Read Many) capabilities.

(2) **Finer-grained security control and auditing**: Building upon identity authentication, it offers object-level access control, allowing individual permission settings for each object. Additionally, since access interfaces are REST APIs, user access logs are easier to obtain and analyze.

(3) **Rich bucket policies**: Buckets are containers for objects. Various policies can be configured for buckets to apply to all objects, including enabling versioning, retention mechanisms, and remote replication.

2. Implementation in Xinhua' s Editorial Platform

2.1 Characteristics of Editorial File Storage

The primary file types stored in Xinhua' s editorial operations include images, audio, video, and documents, totaling approximately 25 million files ranging

from several kilobytes to several gigabytes in size, with real-time write and read operations.

2.2 Software-Defined Storage with MinIO

Given these characteristics of editorial file storage, Xinhua' s editorial converged media platform began exploring the use of object storage for multimedia files. The platform' s object storage system adopts a software-defined storage approach based on the open-source distributed object storage software MinIO. By running object storage programs on ordinary PC servers, it aggregates hard drives from multiple servers into a storage resource pool that provides object storage services externally, thereby replacing expensive traditional centralized storage equipment.

MinIO is open-source software under the GNU AGPLv3 license, providing Amazon S3-compatible interfaces that support all S3 features. It supports both local and containerized deployment, accommodates individual files up to 50TB in size, and theoretically offers unlimited cluster storage capacity. MinIO features simple deployment and is easy to use; its core program is a single Go-compiled binary file with no dependencies on specific software or libraries, configured through environment variables. It supports multi-site replication for automatic data replication between different clusters to achieve disaster recovery. According to MinIO' s official claims, it is the world' s fastest object storage, achieving over 325 GB/s for GET operations and 165 GB/s for PUT operations on a 32-node NVMe drive cluster with 100GbE networking. MinIO includes a web management interface and command-line tool mc. The web interface enables object management, bucket management, user management, data repair, diagnostics, and log management, as shown in [Figure 1: see original paper].

2.3 System Architecture and Deployment

MinIO offers three deployment modes: Single-Node Single-Drive (SNSD), Single-Node Multi-Drive (SNMD), and Multi-Node Multi-Drive (MNMD). SNSD is a standalone mode where files are stored unchanged on MinIO backend disks, and MinIO can only function as an S3 API gateway without many object storage features. SNMD and MNMD are distributed cluster modes that employ erasure coding for redundancy and high availability, providing object-level repair with less overhead than RAID or replication schemes. This paper adopts the MNMD mode.

The hierarchical elements of an MNMD MinIO cluster include Cluster, Server Pool, Erasure Code Set (EC Set), and Drive, with the structure shown in [Figure 2: see original paper]. A MinIO cluster consists of one or more server pools that operate independently without affecting each other, enabling horizontal scaling through the addition of server pools. Each server pool comprises multiple servers and disk drives. Within a server pool, MinIO automatically groups all disk drives into different erasure code sets (striping) based on server and drive

counts. The number of disk drives per erasure code set (stripe size) ranges from 2 to 16, remains fixed after initialization, and all erasure code sets within a server pool share the same stripe size. When MinIO receives an object upload request, it algorithmically selects which erasure code set will store the object. Assuming the stripe size is M (containing M disk drives), the object is real-time encoded into M-N data blocks and N parity blocks, which are then written to respective drives in the erasure code set, with each drive storing one data or parity block. This is generally denoted as EC:N, tolerating N block losses. When data blocks are lost or corrupted, they can be reconstructed using parity blocks. In maximum parity configurations where N equals half of M, the erasure code set can tolerate failure of half its disk drives. Each object is stored in only one server pool and does not span pools. MinIO clusters provide a unified view of all objects across server pools; failure of a single pool does not affect the entire cluster, though an offline pool impacts cluster services.

Beyond networking, MinIO deployment design primarily considers server and disk quantities. Since MinIO' s distributed cluster mode uses erasure coding, the number of nodes and disks in a MinIO cluster server pool must conform to erasure coding algorithm characteristics, though these can vary across server pools. According to MinIO' s recommended deployment topology, a server pool should have an even number of nodes and disk drives, with a greatest common divisor of 16. When exceeding 8 disk drives, parity blocks default to 4. This deployment uses 10GbE networking with 8 nodes totaling 128 2TB SAS disk drives, employing an EC:4 parity scheme for 512TB raw capacity and 384TB usable capacity, as shown in [Figure 3: see original paper].

3. Testing and Application

3.1 Functional and Performance Testing

Using the COSBench testing tool, files of 100KB, 5MB, and 100MB sizes were tested. Functionality was verified, with performance results shown in [Figure 4: see original paper]. The tests demonstrate that distributed object storage delivers good read performance, saturating 1Gb bandwidth for all but 100KB small files, while write performance ranges from 1/5 to 1/4 of read performance.

3.2 Application in Editorial Converged Media Platform

The editorial business environment generates massive quantities of multimedia files including images, audio, video, and documents. The storage, access, and management of these files constitute core supporting functions of the editorial business system. The file service within the core editorial system serves as the critical service for managing and storing all file resources, primarily used for unified management and storage of files produced by various subsystems to facilitate circulation between them while resolving data redundancy issues caused by independent storage.

The file service adopts a Spring Cloud architecture divided into four independent sub-services: static file proxy service, file management service, image processing service, and video transcoding service, each involving file and metadata read/write operations. Accordingly, based on the core system' s file service architecture, interfaces related to file operations were refactored from the original NAS storage path approach to object storage HTTP interface access. New MinIO file upload and registration interfaces were developed to implement file registration functionality via S3 protocol object storage domains, simulating processes for obtaining file metadata including EXIF information for images, bit rates and duration for audio/video files, and more.

The implementation involves: (1) Integrating MinIO by adding dependencies (2) Configuring MinIO with endpoint, port, accessKey, secretKey, and bucketName parameters (3) Defining a MinIO configuration component with appropriate annotations

4. Optimization Practices

Based on the file service architecture of the editorial business core system and the characteristics and applications of object storage, this paper validates the feasibility of applying object storage to editorial operations and identifies architectural optimizations.

At the storage level, using object storage to replace existing file storage (NAS) is feasible but imposes new requirements on other applications and services. For instance, system file services must shift from mounted NAS storage to URL-based requests for accessing object storage resources, with each I/O operation becoming a request—particularly for operations requiring full file access such as MD5 calculation. System file services must carefully consider the impact of I/O operations on logical workflows and optimize existing processes to minimize I/O pressure from object storage usage.

Additionally, implementing a lightweight file metadata management service at the storage layer could manage inherent file attributes such as MD5, extensions, types, image EXIF information, audio/video bit rates, and file reference counts. This would significantly reduce I/O requests between file services and other systems. Such an approach also necessitates enhanced inter-system collaboration and optimization to fully leverage object storage' s potential in editorial converged media platforms.

Note: Figure translations are in progress. See original paper for figures.