A Dynamic and Cost-Efficient Multi-Region Object Storage System Using Erasure Coding

Redacted for double blind review

***Abstract*—Traditional object storage services generally constrain data storage to a single geographic region, limiting flexibility, scalability, and resilience. Our proposed object storage system introduces a novel, adaptive approach that uses random linear network coding to dynamically distribute data fragments across multiple regions. Key features include a gateway architecture that manages encryption, erasure coding, and regional transfers, along with machine learning models that optimize fragment placement and redundancy levels in response to demand and user-defined performance metrics such as latency and bandwidth. This approach aims to deliver high availability and performance while minimizing storage costs.**

***Index Terms*—object storage, machine learning, erasure coding, distributed systems**

I. Introduction

Object storage is a foundational element in cloud computing, enabling scalable storage and retrieval of unstructured data. However, conventional object storage systems are regionally restricted, with data often confined to a single user-designated region. This geographic rigidity not only affects performance and availability but also prevents dynamic adaptation to user access patterns. Furthermore, most systems use a fixed redundancy level for data, which can be inefficient and costly as demand fluctuates.

Our object storage system addresses these limitations through an adaptive multi-region design that utilizes erasure coding and machine learning. The gateway-based architecture provides a seamless user experience, handling encryption, coding, and data routing while allowing access via a single service URL. By dynamically managing fragment distribution and redundancy across regions, our system achieves high availability and responsiveness with cost-efficient storage, tailored to each bucket's performance needs.

II. Related Work

Erasure coding has emerged as a key technique in modern object storage, providing durability and fault tolerance with reduced storage overhead compared to traditional replication strategies. By splitting data into coded fragments that allow reconstruction even if some fragments are lost, erasure coding enables more efficient use of storage in distributed systems [1]. Recent studies demonstrate its scalability and cost-effectiveness, particularly in multi-region setups, where data availability is paramount [2].

Dynamic data placement using machine learning has also proven effective for improving performance in distributed storage. Zhang et al. used deep reinforcement learning to optimize data placement, reducing latency by adapting to network conditions and access demand [3]. Similarly, Zhou et al. reviewed various machine learning models for service placement, highlighting their ability to improve resource allocation and operational cost in distributed systems [4]. Machine learning techniques have also been applied to optimize resource allocation in distributed machine learning clusters, with Li and Hu focusing on minimizing latency and interference in job placements [5]. Inspired by these approaches, our system integrates machine learning to dynamically adjust fragment placement and redundancy levels, balancing cost with performance in a flexible, multi-region object storage framework.

Fig. 1. Illustration of dynamically changing redundancy and fragment placement based on real traffic

III. Proposed System Design

The core of our proposed system design lies in its dynamic and intelligent fragment management. Each object uploaded to a bucket is encoded into multiple fragments using random linear network coding. Instead of a fixed redundancy and placement strategy, our system adapts both parameters based on current access demands and optionally, user-given performance requirements for each bucket. In the following, we briefly describe key components of this system.

A. Gateway Architecture

Our system uses a gateway architecture where clients connect to perform API requests (e.g., bucket creation, object upload/download). Each gateway instance handles encryption and erasure coding of objects and routes the coded fragments to the storage regions assigned to the bucket. During downloads, the gateway retrieves the necessary fragments, reconstructs and decrypts the object, and returns it to the client. Gateways are deployed at several locations in Europe and North America, close to major user clusters, and clients are automatically routed to the nearest gateway instance through anycast DNS. The Gateways will also include a local read-through cache to avoid pulling data from the external storage locations. The common challenge of cache invalidation is prevented by the immutable nature of object storage. This design ensures reduced latency and a unified interface, providing seamless access to the service from any location.

B. Dynamic Fragment Placement

Unlike traditional object storage solutions where data resides in a single region, our system allows fragments to be dynamically relocated across regions. This is particularly useful in scenarios where traffic originates from varying locations over time. By analyzing real-time traffic patterns, the system adjusts the placement of fragments to regions closer to the sources of demand, optimizing download latency for end-users. Machine learning models are employed to predict shifts in access patterns and relocate fragments to regions with higher demand, either preemptively or soon after an unforeseen traffic spike.

C. Adaptive Redundancy Management

The number of redundant fragments stored across regions is also dynamically adjusted. When a bucket experiences low access demand, the system reduces redundancy to save on storage costs, retaining only the minimum number of fragments required for data durability. Conversely, when access demand is high or coming from multiple distant origins, additional redundant fragments are created to maintain availability even if some regions experience temporary outages, as illustrated on Figure 1. This adaptive redundancy approach is key to balancing cost efficiency with availability.

D. Performance-Driven Optimization

Users can specify latency and download speed requirements for each bucket. Our system continually assesses these metrics, using machine learning models to determine optimal configurations of fragment numbers and placement regions to meet these requirements. For example, if a bucket needs to achieve low latency for users in a specific geographic area, fragments are moved closer to that region to enhance responsiveness. Similarly, if high bandwidth is required, additional fragments may be allocated to regions with higher network throughput.

IV. Machine Learning Models for Optimization

Our system leverages several machine learning models to optimize fragment placement, redundancy levels, and storage configurations based on access patterns, latency, and bandwidth considerations

- **Demand Prediction Model**: This model analyzes historical access data and user behavior to forecast future traffic patterns. By predicting likely demand sources, it allows the system to move fragments closer to expected high-demand regions, reducing latency and meeting performance expectations.
- **Redundancy Optimization Model**: This model decides the appropriate level of redundancy based on access frequency and predicted demand. For buckets with low activity, it reduces the redundancy factor to lower storage costs, while for high-demand buckets, it increases redundancy to enhance availability.
- **Performance Prediction and Fragment Placement Model**: This model predicts the optimal storage regions for fragments based on latency and bandwidth estimates. We regularly measure download speeds and latency between every gateway and each storage region using synthetic traffic of varying sizes (from 1 byte to 50 MB). These periodic measurements, taken at randomized times, capture daily fluctuations in network performance, providing insights into regional variability in latency and bandwidth. For example, regions served by smaller providers may exhibit reduced bandwidth during business hours, while regions supported by larger providers may experience peak usage in the evenings. With this data, the model can predict access speeds for any fragment size across regions and adjust placement to ensure that user-specified latency and bandwidth requirements are met.
- **Cost Minimization Model**: This model identifies storage configurations that minimize costs without compromising performance. It accounts for storage, data transfer costs, and the number of fragments necessary to meet reliability standards, suggesting configurations that balance operational costs with service quality.

These models work together to dynamically adapt fragment distribution and redundancy based on evolving access demands, providing a cost-effective, high-performance storage solution that meets diverse user requirements.

V. Conclusions and Future Work

Our proposed system design introduces a flexible, multi-region object storage solution that adjusts both fragment placement and redundancy in response to user demand and specified performance metrics. By leveraging erasure coding and machine learning, our design provides improved availability and performance while minimizing storage costs. Initial simulations indicate promising results in achieving low latency and high availability with optimized redundancy configurations.

Future work will focus on refining the predictive capabilities of the machine learning models, potentially incorporating real-time network conditions to further optimize fragment placement. Additionally, we plan to explore other coding techniques that could enhance the system's fault tolerance and performance in edge scenarios, where bandwidth and latency requirements are more stringent. Through these advancements, we aim to make our dynamic object storage system adaptable for a wide range of applications, from enterprise cloud storage to IoT data management.

References

[1] MinIO, "Erasure coding vs. raid explained," https://blog.min.io/erasure-coding-vs-raid/, 2023, accessed: 2024-11-04.

[2] Qumulo, "Erasure coding vs. raid explained," https://qumulo.com/blog/erasure-coding-vs-raid-explained/, 2022, accessed: 2024-11-04.

[3] Y. Zhang *et al.*, "Deep reinforcement learning for optimizing data placement in data center networks," *IEEE*, 2023, accessed: 2024-11-04. [Online]. Available: https://ieeexplore.ieee.org/document/10386549

[4] L. Zhou *et al.*, "Machine learning methods for service placement: A review," *Springer*, 2024, accessed: 2024-11-04. [Online]. Available: https://link.springer.com/article/10.1007/s10462-023-10684-0

[5] X. Li and W. Hu, "Deep learning-based job placement in distributed machine learning clusters," *IEEE*, 2023, accessed: 2024-11-04. [Online]. Available: https://ieeexplore.ieee.org/document/8737460