# Investigative Study of Data Tiering Systems and Supports

**Gargi Mhaskar [1], Soham Kulkarni [2], Mokshad Vaidya [3], Vaibhav Mahajan [4] & Dr. Shweta Dharmadhikari [5]**

[1]Gargi Mhaskar; PICT (Information Technology), Pune, Maharashtra, India, mhaskargargi268@gmail.com
[2]Soham Kulkarni; PICT (Information Technology), Pune, Maharashtra, India, sohamkulkarni0706@gmail.com
[3]Mokshad Vaidya; PICT (Information Technology), Pune, Maharashtra, India, mokshad.nv0709@gmail.com
[4]Vaibhav Mahajan; PICT (Information Technology), Pune, Maharashtra, India, nvaibhavmahajan00@gmail.com
[5]Dr. Shweta Dharmadhikari; PICT (Information Technology), Pune, Maharashtra, India, scdharmadhikari@pict.edu

**Abstract:**

Data storage and management in modern computing environments pose significant challenges, with a growing need for efficient and cost-effective solutions. Data tiering is a crucial aspect of modern storage systems, enabling efficient management and optimization of data placement across different storage tiers based on various criteria such as access frequency, performance requirements, and cost considerations. This paper provides a comprehensive study of data tiering approaches, methodologies, and implementations across a variety of systems.

**Keywords:** Data Tiering, Storage Management, Cold Data, File System

## 1. Introduction

In today's age of data, it has become crucial to optimize the storage of that data to reduce cost and latency while improving throughput, reliability, and availability. One of the strategies for storage optimization is data tiering. It is a data storage strategy that defines different storage ties and shifts data between tiers to fulfil the requirements of an organization depending on the usage. Cloud tiering involves storing frequently accessed data in fast cloud storage while less used data is assigned to slower and cheaper storage. Advantages of cloud tiering also include more granular control over cloud resources and rapid data recovery in case of emergencies such as natural disasters or sudden system failure. Extending this need for storage optimization to the file system present on compute clusters and on-premise servers, we can utilize the primary, more expensive storage more judiciously by shifting a bulk of the data to secondary storage services such as AWS S3 and Google Cloud Storage, which implement a Pay-As-You-Go model with high durability. Certain factors need to be considered while implementing data tiering. These factors are as follows:

- *Block-Level vs File-Level Tiering*: File-level tiering operates at a higher granularity, allowing for more precise control over which files are moved to lower-cost storage tiers based on their access patterns and importance. This approach maximizes savings without locking organizations into specific storage vendors or technologies. In contrast, block-level tiering works at a lower level, moving data in fixed-size blocks between tiers based on usage patterns. While block-level tiering offers benefits like faster data retrieval, file-level tiering provides more flexibility and cost-efficiency, especially in heterogeneous environments or those leveraging both on-premises and cloud storage. [11]
- *Identification of Cold Data*: Identifying cold blocks accurately is a crucial task when we are implementing cloud tiering. Cold blocks can be identified using algorithms such as Least Recently Used (LRU) and AI-based Ensemble Models that consider factors like access frequency, file size and user information. The analysis of metadata can be leveraged for cold data identification by considering both access frequency and recency.
- *Primary Storage Utilization*: It is important to note that data tiering has been developed to allow for more judicious use of the primary storage without losing data. While data tiering prevents the primary storage from being over utilized, we have to prevent it from being under- utilized. Some methods of ensuring this are designating a maximum limit of utilization of the primary storage beyond which cold data is tiered, or defining a hot-cold storage splitting ratio. [2]
- *Secondary Tier Read Write Costs*: The costs associated with reading data from and writing data to the

secondary tier need to be considered while implementing data tiering. A policy needs to be defined for data currently stored in the secondary tier that is accessed frequently, and, therefore, can no longer be considered as cold data, to be migrated back to the primary tier. File access patterns of the cloud tier need to be monitored to judge which data needs to be written back to the primary tier. The present utilization of the primary tier should also be considered, along with migration cost and secondary tier read cost trade off.

Careful consideration of the above factors allows for the construction of data tiering supports that are best suited to the requirements of any particular organization. A major challenge in this field is, therefore, selecting the right parameters as making the wrong choices can lead to increased read-write costs because of selecting the wrong data to be tiered or underutilization of the primary tier. The objective of this paper is to study various data tiering supports and their methodology.

## 2. Literature Survey

This section presents an understanding of the different methodologies used by various data tiering supports.

### 2.1 Automated Cloud Storage Tiering through Hot-Cold Data Classification [1]

The proposed automated storage tiering (AST) system in [1] aims to predict data temperature (frequency of access) and automate data migration between hot and cold storage tiers in cloud environments. The system leverages machine learning techniques, including stacked ensemble learning, to categorize data and allocate it to the appropriate storage tier without manual intervention.

1) Data Temperature Prediction: The system learns from past data access patterns and user behaviors to predict the future access frequency of files. This prediction engine utilizes various input features such as file access frequency, size, type, user information, etc. Deep neural networks (DNN), support vector machines (SVM), and ensemble learning methods are all used as prediction models.
2) Automated Data Migration: Based on the predicted data temperature, files are automatically allocated to either hot or cold storage. The system operates on scheduled routines (e.g., weekly, biweekly, monthly) to perform data migration tasks. New files are initially stored in a reserved hot space until the next scheduled operation.
3) Components of the System:
   - Data Content Manager: Collects access logs, monitors storage usage, and generates metadata for input datasets.
   - Hot-Cold Data Prediction Engine: Utilizes super- vised learning methods and ensemble learning to predict future file access patterns and determine data temperature.
   - Flexible Output: Generates a file priority list based on predicted data temperature, allowing customizable hot-cold storage splitting ratios.
4) Evaluation and Testing: Evaluation of the system is done by considering Feasibility, Reliability, and QoE. The cost savings, quality of experience (QoE), correctness of hot data predictions, and flexibility with different cloud storage providers' business strategies are measured and considered during the testing and evaluation.

### 2.2 FabricPool: A NetApp® ONTAP® software component [2]

FabricPool is a feature in ONTAP that enables tiering of data between a local tier (storage aggregate) and a cloud tier (external object store), optimizing storage efficiency and performance. ONTAP is a proprietary operating system used in storage disk arrays.

1) Block Temperature: Blocks are assigned temperature values indicating their activity level. Hot blocks are stored in the local tier, while cold blocks are moved to the cloud tier based on a cooling scan.
2) Object Creation: Blocks are grouped into 4MB objects before being written to the cloud tier. Each object contains 1,024 4KB blocks.
3) Data Movement:

   a. Tiering to Object Store: Cold blocks are identified, grouped into objects, and then moved to the cloud tier in accordance with volume tiering policies. Tiering to the cloud tier is only enabled by default if the local tier is more than 50% full, to prevent underutilization of the local tier.
   b. Reading from Object Store: When data is read from the cloud tier, concurrent byte-ranged GET operations are initiated, optimizing network efficiency. Only the necessary Write-Anywhere-Fall- Layout (WAFL) are read.
4) Random Reads: Cold blocks read randomly from the cloud tier are made hot and written back to the local tier for improved performance.
5) Sequential Reads: Sequential read-ahead requests are made to the cloud tier to optimize performance, but the data remains in the cloud tier.
6) Write-Back Prevention: When the local tier is over 90% full, or in other words, it is heavily utilized, cold data is read directly from the cloud tier rather than being written back to maintain local tier performance.
7) Object Deletion and Defragmentation: FabricPool deletes entire objects based on the percentage of un-referenced blocks within an object.

**2.3 Auto-Tiering System using Deep Reinforcement Learning [3]**

This method leverages DRL to optimize storage costs while maintaining performance requirements. The challenge faced by cloud storage providers is in efficiently managing data across two tiers of storage: high- performance, high-cost storage and low-performance, low-cost storage. The goal is to minimize storage costs while ensuring that performance requirements are met for different data access patterns.

1) Auto-Tiering System: An auto- tiering system is pro- posed that dynamically classifies data into different storage tiers based on its access patterns and cost considerations. By automatically placing data in the appropriate tier, RLTiering aims to optimize storage costs while maintaining performance.
2) Deep Reinforcement Learning (DRL): DRL techniques are used to train an agent to make tiering decisions. DRL allows the agent to learn optimal tiering policies through trial and error, maximizing long-term rewards (cost reduction) over time.
3) State Representation: The state of the storage environment is represented by various factors, including data access frequency, data size, storage costs, and performance metrics.
4) Action Selection: The DRL agent selects actions corresponding to tiering decisions, such as promoting data to a higher-performance tier or demoting data to a lower-cost tier. The actions are chosen to maximize the expected cumulative rewards over time, balancing cost savings with performance requirements.
5) Reward Mechanism: A reward function is defined to incentivize cost reduction while maintaining performance. The agent receives rewards based on how well its tiering decisions align with cost-saving objectives. For example, the agent may receive higher rewards for demoting infrequently accessed data to a lower-cost tier.

**2.4 Cloud Tiering for Layered Tiers [4]**

[4] focuses on optimizing storage tiers for cost- effective data storage by proposing a rule-based classification approach. This approach considers four storage tiers instead of two, aiming to improve performance, efficiency, and cost savings.

1) Rule-Based Classification: The classification scheme utilizes IF-THEN rules for predicting the storage tier for each object based on specific criteria such as access frequency, data size, and age of the stored object. Rules are defined to assign objects to different storage tiers based on their characteristics.
2) Classification Criteria: Four general tiers are defined: Premium, Hot, Cold, and Archive.
   a. Premium tier: For continuously or near-continuously accessed data requiring high performance and durability.
   b. Hot tier: For data with frequent access patterns, accessed daily or weekly, requiring fast access times.

c. Cold tier: For data with infrequent or irregular access patterns, accessed monthly or quarterly.

d. Archive tier: For rarely accessed data with minimal retrieval requirements, used for long-term storage and compliance.

3) Solution Approach: Weights (W) are defined for factors such as size (Z), access frequency (F), and age (A) of the data. The data is represented as a list of dictionaries. Each dictionary contains size, access frequency, and age of an object. The priority score for each object is calculated using defined weightings for size, access frequency, and age. Objects are, then, divided into groups based on priority score thresholds for each storage tier. Priority score thresholds are set for each tier to balance trade-offs between data size, access frequency, and age. These P The thresholds are as follows:

a. Premium: Priority score threshold of 1.0 for critical and frequently accessed data.

b. Hot: Priority score threshold of 0.7 for data with slightly lower priority but significant access requirements.

c. Cold: Priority score threshold of 0.4 for less frequently accessed data, offering cost-effective storage.

d. Archive: Priority score threshold of 0.1 for rarely accessed data, serving as a long-term storage solution with cost optimization.

4) Window Definitions for Access Frequency: Nineteen possible windows are defined ranging from hourly to yearly to capture different access frequency patterns.

### 2.5 Middleware enabling Multi-Tiered Cloud Storage Instances [5]

The middleware proposed in [5], named Tiera, aims to address the complexity introduced by multi- tiered cloud storage services by providing a middleware solution encompassing numerous cloud storage providers and enables the simple setting of data storage policies to accomplish required trade-offs. Tiera instances encapsulate multiple cloud storage services and allow for easy specification of storage policies. Clients of Tiera instances are shielded from the under- lying complexity of multi-tiered cloud storage services. Tiera supports dynamic replacement/addition of storage policies and tiers. The architecture is as follows:

1) Data Model: Tiera uses an object storage model so that data is managed as objects, enabling unified access to data distributed among different storage services. Objects are treated as uninterpreted sequences of bytes with common attributes such as size, access frequency, dirty flag, location, and time of last access.

2) Architecture Overview: Tiera instances are composed of three layers: the application interface layer, the storage interface layer, and the control layer. The application interface layer provides a simple PUT/GET API for storing and retrieving data. All of the varied storage services encompassed by the Tiera instance interact with the storage interface layer. The control layer manages data placement and movement across tiers based on event-response mechanisms.

3) Event-Response Mechanism: Tiera supports three types of events: timer events, threshold events, and action events. Responses to events include storing, retrieving, moving, copying, compressing, encrypting, decrypting, and deleting data. Responses can be combined and customized to define data management policies easily.

### 2.6 Hybrid Cloud Storage Architecture [6]

The method proposed in [6], entitled CloudMW, utilizes a combination of S3, EBS, and EC2 local storage to support efficient data management in cloud environments. The architecture consists of two main layers: cloud management and storage management middleware.

1) Cloud Management Layer: By means of service-provider-defined APIs, this layer is in charge of data storage and data services delivery. It manages the consistency of data transfers between S3 and EBS as well as operations like backup, restoration, deduplication, and encryption and decryption. Working data sets are primarily stored on EBS volumes but backed up to S3 for dependability and archival purposes.

2) Storage Management Middleware: This layer is responsible for data services for client virtual machines (VMs) and facilitates data sharing among multiple VM clients accessing shared data on EBS volumes. It incorporates various mechanisms to enhance performance and efficiency, including Data Chunking (files are partitioned into chunks and distributed across storage nodes for parallel access, improving data transfer bandwidth), Data Replication, Storage Virtualization (cloud storage space is virtualized and can be accessed as a single storage device). Some crucial mechanisms also include:
   - Multi-level Caching: Different tiers of storage are utilized, with S3 serving as the back-end long-term storage, EBS volumes providing online storage services, and EC2 instances offering direct-attached storage. Caching technology is employed to optimize data access performance.
   - I/O Scheduler: It handles issues such as changeability in I/O sizes, locality of accesses across VMs, and different request priorities based on application requirements.
   - Metadata Management: Manages metadata information using SOAP/REST protocols. It generates small- sized indirection maps to convert file system block addresses to the present locations of blocks on cloud storage and EC2's local storage.

### 2.7 Automated Data Tiering in a Centralized Storage System for Private Cloud Environments [7]

The storage system implements automated data tiering to efficiently manage user files based on access patterns and performance requirements. The automated data tiering methodology efficiently manages file storage by dynamically migrating data between high-speed SSD drives and high capacity, lower cost SAS drives based on access frequency, optimizing performance and cost-effectiveness.

1) System Overview: The storage system consists of three pairs of duplicated storage array controllers connected to disk chassis with a total capacity exceeding 3 PB. Near- line SAS disk drives are used for bulk storage, while 24-TB solid-state disk (SSD) drives provide high- speed access. Five pairs of duplicated NAS heads are included for load balancing, connected to all disk array controllers via duplicated Fiber channel switches.
2) Automated Data Tiering: The array controller features automated data tiering, allowing users to write files to the fastest disk drives (Tier 1). The frequency of access to each block is calculated, and blocks not accessed for over two weeks are migrated to slower disk drives (Tier 2). Conversely, regularly accessed blocks are migrated from Tier 2 to Tier 1. The migration process occurs nightly to optimize file access speed and reduce costs associated with high-speed disk drives.
3) Types of Services: Two types of storage services are provided: high-speed and archive. High-speed access is required for high-speed services, utilizing SSD drives (Tier 1), while archive volumes are configured only on Tier 2 (hard disk drives). A migration mechanism between RAID1 and RAID5/6 is implemented, with RAID1 providing faster access but requiring more space, and RAID5/6 effectively utilizing allocated space but with slower access. Users can initially write archiving files on RAID1 blocks, which then migrate to RAID5/6 blocks after a few weeks.

### 2.8 Adaptive Fine-grained Cache Management for SSD-based File Systems [8]

The adaptive fine-grained cache management scheme aims to improve system performance and reduce write-back traffic to Solid State Drives (SSD) in a cloud storage environment.

1) Adaptive Fine-grained Cache Management (AFCM) dynamically selects cache modes based on a page's "dirty ratio." The dirty ratio represents the proportion of modified data within a page compared to its total size.
2) AFCM employs a Cost-Benefit Model to determine the most suitable cache mode for each synchronized page.
3) There are two cache modes: Coarse-Grained Cache Mode and Fine-Grained Cache Mode.
   a. Coarse-Grained Cache Mode: It involves caching entire data pages at once. It requires less overhead for indexing and management, and provides lower granularity, meaning that a larger portion of data is cached together. However, it results in lower cache utilization.

b. Fine-Grained Cache Mode: It involves caching smaller units of data, such as individual cache lines or smaller segments. It requires higher overhead for indexing and management due to the finer granularity. The higher granularity allows for more precise caching of modified data. However, it has higher overhead.

4) The benefits of AFCM include reduced SSD write operations, while the costs encompass metadata overhead. If benefits outweigh costs (indicating a high dirty ratio), AFCM selects fine-grained cache mode to optimize synchronization efficiency. If costs outweigh benefits (indicating a low dirty ratio), AFCM opts for coarse- grained cache mode to minimize metadata overhead.

5) AFCM dynamically adjusts cache modes as the dirty ratio fluctuates, ensuring efficient cache management in response to workload changes.

### 2.9 Tiered File System for Persistent Memories and Disks [9]

In [9], the authors have proposed TPFS. It is a tiered file system spanning disks and persistent memory (PM), that leverages and PM and disks to offer high file performance for various access patterns. TPFS is implemented based on NOVA. NOVA is a PM-based kernel-space file system which maximizes performance on hybrid memory systems and provides robust consistency guarantees. TPFS inherits the design of log-structured file layout and scalable memory allocators from NOVA.

1) Sending I/O to the Most Suitable Tier: TPFS dynamically determines whether to send file writes and reads to PM or disks based on access patterns and workload characteristics. Small, synchronous writes are directed to PM for higher bandwidth and lower latency, while larger asynchronous writes are targeted to disks to leverage faster DRAM buffering. For reads, if DRAM bandwidth is underutilized, TPFS migrates read-hot data from PM to disk and caches it in DRAM for accelerated reads.
2) Accurate File Temperature Profiling: TPFS identifies the temperature of file blocks based on their access patterns and historical modification/read times. Write-cold files have older-than-average modification times and Read- hot files have newer-than-average read times.
3) High PM Space Utilization: TPFS optimally utilizes PM space by absorbing synchronous writes and using disks to expand capacity. It dynamically adjusts PM utilization thresholds based on application read-write patterns to efficiently handle file writes.
4) Exploiting DRAM Read Bandwidth: TPFS monitors read data flow from PM and DRAM, migrating read-hot data to disk and caching it in DRAM if DRAM bandwidth is underutilized. This approach maximizes read throughput by leveraging both volatile DRAM and non-volatile PM.
5) Migrating File Data in Groups: TPFS merges adjacent file data into big chunks for migration to disks, using sequential disk bandwidth to improve migration efficiency.
6) High Scalability: TPFS builds upon NOVA's per-CPU storage space allocators to comprise all storage tiers, improving overall scalability. It utilizes per-CPU migration and page cache writeback threads to improve the scalability of the tiered file system.

### 2.10 Utilization of Primary Storage of Compute Clusters as a Caching Tier [10]

The primary objective of this system is to enable VMware customers to leverage their vSphere clusters for application execution while utilizing secondary storage solutions for cost- effective data retention. This involves developing a caching hybrid storage system that integrates primary and secondary storage seamlessly. The methodology involves building upon VDFS (VMware Distributed File System), a hyper-converged distributed file system running on vSAN, to create caching- tier volumes. These volumes serve as the caching tier of the hybrid system, caching the working set of application data on primary storage while utilizing secondary storage for bulk data retention. The architecture is as follows:

1) VDFS Caching-Tier Volumes: These volumes cache the working set of application data on primary storage, hiding the latency of secondary storage. They utilize write- back cache on primary storage to optimize performance.
2) Caching Model: VDFS exposes cache-related properties to applications through extended attributes, enabling application-specific caching policies. For example, ap- plications can control caching behavior

based on the cached state of files, improving cache hit rates.

3) Caching at the FS Layer vs. App Layer: Caching functionality is exposed to applications at the file system layer, allowing different compute frameworks to manipulate caching state uniformly. This enables shared cache- related state across various applications, enhancing performance for the entire pipeline.
4) Write-Back Mechanism: Write-back of the cache follows a directed approach, giving applications fine-grained control over when updates are written back. This ensures continued operation even in the face of disconnection from the secondary storage.
5) Backends: The system supports various secondary storage backends such as Amazon S3 and remote file systems. Applications can seamlessly access these backends through VDFS caching-tier volumes, without needing to modify their operations.

### 3. Comparative Analysis

| Ref No. | Title | Authors | Publication and Year | Type of Tiering | Methodology Used | Strengths | Weaknesses |
|---|---|---|---|---|---|---|---|
| 1 | A Novel Automated Cloud Storage Tiering System through Hot-Cold Data Classification (AST) | Y.-F. Hsu, R. Irie, S. Murata, and M. Matsuoka | IEEE 11th International Conference on Cloud Computing 2018 | Cloud Tiering | Stacked ensemble learning for data temperature prediction. Data tiering happens automatically on scheduled routine basis such as weekly, monthly, etc. | Successful prediction of cold data leading to cost savings. System also ensures reliability, QOE, and flexibility | System has only been tested on non-commercial data. Training the ML model takes a long time. |
| 2 | FabricPool best practices, TR-4598 NetApp ONTAP 9.11.1 | J. Lantz | NetApp Media 2024 | Cloud Tiering | Enables automated data tiering at the block level using block temperature. | Has provisions to differentiate between over and under-utilized local storage, and random reads and sequential reads. | Specific only to NetApp's ONTAP system. Supports only block tiering. |
| 3 | RLTiering: Auto-Tiering System using Deep Reinforcement Learning | M. Liu, L. Pan and S. Liu | IEEE Transactions on Parallel and Distributed Systems Volume: 34, 01 | Cloud Tiering | Dynamically classifies data into different storage tiers based on its access patterns and cost | Optimization because of deep reinforcement Has a Cost-driven approach without | May require significant computational resources for training |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | February 2023 | | considerations using Deep Reinforcement Learning. | reduction in performance | |
| 4 | Towards Cloud Storage Tier Optimization with Rule-based Classification | A. Q. Khan, N. Nikolov, M. Matskin, R. Prodan, C. Bussler, D. Roman, A. Soylu | Lecture Notes in Computer Science, vol 14183. Springer, Cham 2023 | Cloud Tiering | Cloud storage tier optimization using rule-based classification. Defines 4 tiers of Premium, Hot, Cold and Archive, and calculates weighted priority score for effective allocation of data. | Better differentiation between data due to consideration of 4 tiers. Assigns appropriate weights to all factors concerning data while calculating priority score. | May not be suitable for all types of data. Requires clear rule definitions. |
| 5 | Tiera: Towards Flexible Multi-Tiered Cloud Storage Instances | A. Raghavan, A. Chandra, and J. B. Weissman | Proceedings of the 15th International Middleware Conference December 2014 | Multi-tiered Cloud Tiering | Middleware for easy specification of multi-tiered cloud storage instances and storage policies. | Provides easy specification of multi-tiered cloud storage. Enables rich array of storage policies. | May introduce additional complexities. Not yet compatible with horizontal scaling. |
| 6 | Optimizing storage performance in public cloud platforms | J. Wang, P. Varman, C. Xie | Journal of Zhejiang University SCIENCE December 2011 | Multi-tiered Cloud Tiering | Hybrid cluster storage. Cloud management layer manages storing of data and provides data services. Middleware layer provides data services for client VMs and facilitates data sharing in shared data. | Decreases the possibility of performance variations and leads the way for realistic I/O QOS guarantees. | May not be applicable to private cloud environments. |
| 7 | A Centralized Storage System with Automated Data Tiering for Private | M. Shikida, H. Nakano, S. Kozaka, M. Mato, S. Uda | Proceedings of the 41st annual ACM SIGUCCS conference | SSD to SAS Drives Tiering | Automated data tiering that manages file storage by dynamically migrating | Provides high-speed and low-cost storage services at a low power | Specific to the authors' private cloud environment. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Cloud Environment | | on User services November 2013 | | data between high-speed SSD drives and high capacity, lower cost SAS drives based on access frequency. | consumption . Friendly interface for user. | |
| 8 | Efficient and Consistent NVMM Cache for SSD-based File System | Y. Chen, Y. Lu, P. Chen and J. Shu | IEEE Transactions on Computers, vol. 68, 1 Aug. 2019 | Hybrid Cache based on DRAM and NVMM | AFCM that dynamically selects cache modes based on a page's "dirty ratio." Employs a Cost-Benefit Model to determine the most suitable cache mode | Significantly improves the system performance and reduces the write traffic to SSD compared to the conventional coarse grained cache management scheme. | Specific to SSD-based file systems |
| 9 | TPFS: A High Performance Tiered File System for Persistent Memories and Disks | S. Zheng, M. Hoseinzadeh, S. Swanson, and L. Huang | ACM Transactions on Storage, Volume 19 Issue 2, 6 March 2023 | PM to SSD Drives Tiering | Dynamically determines whether to send file writes and reads to PM or disks based on access patterns and workload characteristics . | Bridges the gap between disk-based storage and PM-based storage. Provides high performance and large capacity to applications. | Specific to NOVA filesystem |
| 10 | Hybrid Cloud Storage: Bridging the Gap between Compute Clusters and Cloud Storage | A. Gupta, R. Spillane, W. Wang, M. Austruy, V. Fereydouny, and C. Karamanolis | ACM SIGOPS Operating Systems Review Volume 51 Issue 1, 2017 | Cloud to Cache Tiering | Caching hybrid storage system built upon VDFS to create caching tier volumes, caching the working set of application data on primary storage while utilizing secondary | Data mostly resides on inexpensive cloud while compute runs faster primary vSAN storage enabling. | Specific to VMware's architecture. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | storage for bulk data retention. | | |

## 4. Proposed Methodology

We have developed a Proof-of-Concept specifically to implement block-level cloud tiering in XFS. The concepts involved in the POC are detailed below.

### 4.1 Introduction to XFS and Cold Block Identification

XFS is a high-performance 64-bit journaling file system renowned for its ability to efficiently handle large data volumes with minimal performance degradation, even under heavy loads. It supports variable block sizes, adjustable to system requirements, and excels in parallel I/O operations due to its allocation group design. Each allocation group, a subdivision of physical volumes, manages its own nodes and free space, facilitating concurrent processes and threads. As the primary local storage tier, XFS organizes files into blocks, which are fixed-sized data units. Block sizes in XFS can range from 512 bytes to 64 KB, depending on user needs and system commands. [12]

Considering that XFS files are organized as blocks, the blocks that are accessed infrequently i.e. cold blocks were identified using system commands that trace block-level operations. The system commands generate an output with multiple data points that have to be preprocessed to generate data points that can be utilized for cold-block identified. [13] In the methodology, the use of K-means clustering, a machine learning model, to generate a list of Hot and Cold blocks is proposed. The use of K-means is prompted by the dataset being unlabeled. Some of the features that are proposed to be used for training the model are Frequency, Recency and Mean Time Between Access.

### 4.2 Block Migration to Cloud Tier

Once the identification of the cold blocks is complete, the devised system has a four-step process to migrate the cold blocks to the cloud tier. The steps are detailed below

#### 4.2.1 Mapping Block Numbers to Files

The system command used to deallocate blocks requires the input of file path and offset value. Hence, the cold block numbers identified need to be mapped to the respective file path and calculating the offset.

#### 4.2.2 Storing Information of Cold Blocks

The identified cold block will soon be deallocated so it is essential that the primary tier has access to information about the deallocated block so as to perform read-write operations. The relevant data is stored in a table that is maintained on the local tier.

#### 4.2.3 Uploading Cold Blocks to the Cloud and Deallocating space

Once the table is updated, the cold blocks are iteratively uploaded to the cloud tier. Simultaneously, every block that has been successfully uploaded to the cloud tier is immediately deallocated from the primary tier.

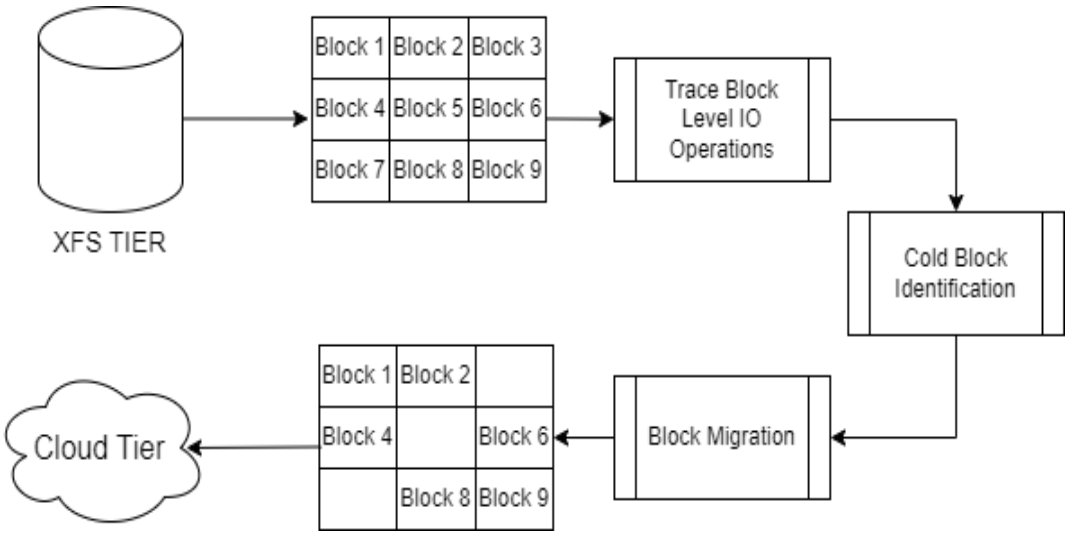


Fig. 1. Proposed System Architecture

### 4.3 Results and Discussion

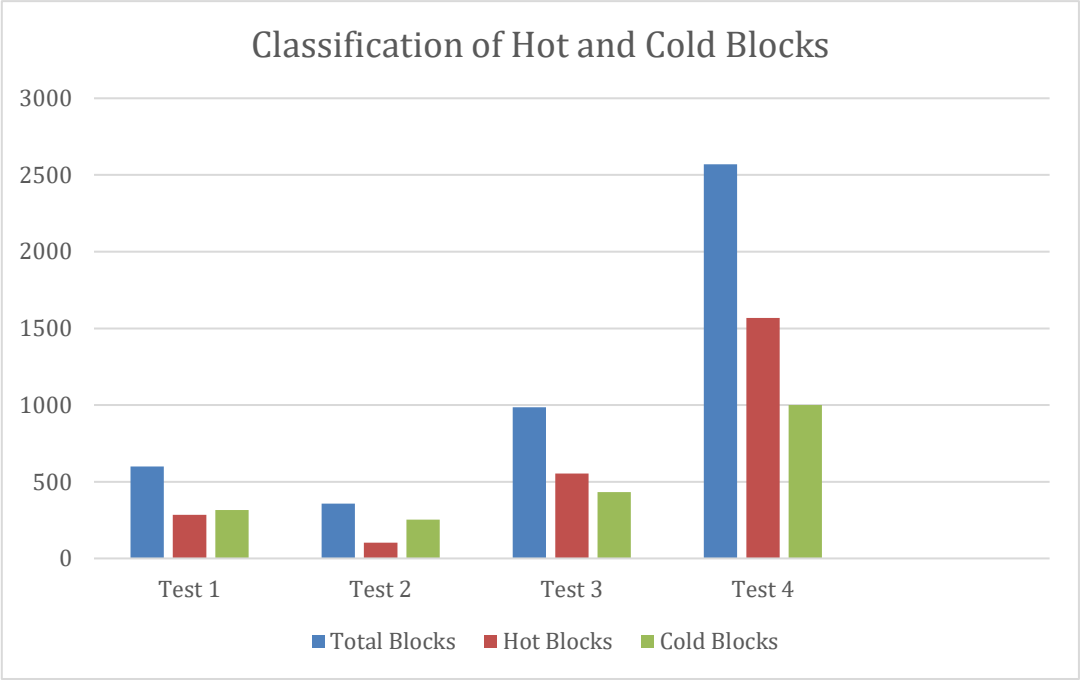


Fig. 2. Testing of Hot and Cold Block Classification

For evaluating the system, a dataset where blocks were randomly accessed to simulate the usage of an organization was generated. The data was preprocessed as described above and then K-means clustering was applied to generate two lists of Hot Blocks and Cold Blocks. As can be seen in the above graph, the percentage of cold blocks identified is dynamic. In the above graph, test 1 has 52.6% cold blocks, test 2 has 71.1% cold blocks, test 3 has 43.8% cold blocks, and test 4 has 38.9% of cold blocks. The system will then upload these blocks to the cloud tier, freeing up that space in the primary tier. By changing the number of clusters that the machine learning model is generating, the cloud tiering can be refined.

The system features a modular architecture allowing users to integrate various components, such as different log tracing methods, cold-block identification algorithms, or block migration commands, without affecting overall functionality. The solution presented is OS-agnostic and operates on any operating systems with XFS mounted,

requiring no additional infrastructure from Cloud Service Providers. The system proves that block-level tiering can be efficiently implemented on XFS, providing a cost-effective method for handling big data.

While current data tiering strategies address various scenarios, the proposed system specifically applies cloud tiering to XFS, making it highly effective for organizations managing large volumes of data. The proposed system's current limitations include the lack of testing on a dataset that accurately reflects an organization's usage patterns, which hinders the ability to assess the model's accuracy and develop a more precise model. Future improvements include generating a genuine dataset for better precision, enhancing user control, adding more cloud tiers for nuanced data usage patterns [14], and refining algorithms to optimize local tier utilization.

## 5. Conclusion

This paper extensively explored various data tiering systems, encompassing tiering between local storage and the cloud, multi-tiered cloud storage, and SSD-based file systems. Each system employs diverse approaches to identify data for migration across tiers. Additionally, these systems carefully weigh the costs and benefits to determine their effectiveness in optimizing storage. Looking ahead, there's ample room for advancement in data tiering methodologies.

The paper also elaborates on the proposed methodology for cloud tiering in XFS includes tracing system block operations, identifying cold blocks, and managing block migration to the cloud. The modular design allows customization without disrupting the overall system. Using K-means clustering (k=2) for cold block identification, the system classifies 38-70% of blocks as cold. However, the value of k can be changed according the number of tiers available for more nuanced cloud tiering. Further, the system can be modified to prevent underutilization of the primary tier. Future improvements in the system could include ensemble classification models, tiering based on primary storage utilization and, extensive testing on actual organizational data to improve the model.

Data Tiering and Cloud Tiering, in particular represent an important aspect of optimized storage management that could lead to significant cost reductions for every data-heavy organization. Further research in this field should focus on file system-specific tiering that incorporates both access frequency and recency for enhanced optimization.

## References

[1] Y.-F. Hsu, R. Irie, S. Murata, and M. Matsuoka, "A novel automated cloud storage tiering system through hot- cold data classification," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018.

[2] J. Lantz, "FabricPool best practices TR-4598 NetApp." https://www.netapp.com/pdf.html?item=/media/17239-tr-4598.pdf, 2024

[3] M. Liu, L. Pan and S. Liu, "RLTiering: A Cost-Driven Auto-Tiering System for Two-Tier Cloud Storage Using Deep Reinforcement Learning," in IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 2, pp. 501-518, 1 Feb. 2023

[4] A. Q. Khan, N. Nikolov, M. Matskin, R. Prodan, C. Bussler, D. Roman, A. Soylu, "Towards Cloud Storage Tier Optimization with Rule-Based Classification" in 10th IFIP WG 6.12 European Conference, ESOCC 2023, October 24–25, 2023, Proceedings.

[5] A. Raghavan, A. Chandra, and J. B. Weissman, "Tiera," Proceedings of the 15th International Middleware Conference on- Middleware '14, 2014.

[6] J. Wang, P. Varman, C. Xie, "Optimizing storage performance in public cloud platforms", Journal of Zhejiang University - Science C. 12. 951- 964, 2011.

[7] M. Shikida, H. Nakano, S. Kozaka, M. Mato, S. Uda, "A Centralized Storage System with Automated Data Tiering for Private Cloud Environment" in Proceedings of the 41st annual ACM SIGUCCS conference on User services, 2013.

[8] Y. Chen, Y. Lu, P. Chen and J. Shu, "Efficient and Consistent NVMM Cache for SSD-Based File System," in IEEE Transactions on Computers, vol. 68, no. 8, pp. 1147-1158, 1 Aug. 2019.

[9] S. Zheng, M. Hoseinzadeh, S. Swanson, and L. Huang, "TPFS: A High- Performance Tiered File System for Persistent Memories and Disks," in ACM Trans. Storage 19, 2, Article 20 (May 2023), 28 pages.

[10] A. Gupta, R. Spillane, W. Wang, M. Austruy, V. Fereydouny, and C. Karamanolis, "Hybrid cloud storage: Bridging the gap between Compute Clusters and cloud storage", ACM SIGOPS Operating Systems Review: Vol 51, no 1.

[11] Komprise, "Block-level Tiering vs File-level Tiering - Komprise Data Management," https://www.komprise.com/resources/block-level-tiering-vs-file-level-tiering-read-online-ppc-2/#:~:text=File%2Dlevel%20tiering%20provides%20up .

[12] Z. Wang, "Research of data storage mode and recovery method based on XFS file system," in 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, pp. 369-372, 2016

[13] B. Zoetekouw, "btrace(8) - Linux man page.", https://linux.die.net/man/8/btrace.

[14] A. Q. Khan, M. Matskin, R. Prodan, C. Bussler, D. Roman, and A. Soylu, "Cloud storage tier optimization through storage object classification," Computing, Apr. 2024.