Optimizing Distributed Tiered Data Storage Systems with DITIS

Sotiris Vasileiadis
Cyprus University of Technology
Limassol, Cyprus
sr.vasileiadis@edu.cut.ac.cy

Matthew Paraskeva
Cyprus University of Technology
Limassol, Cyprus
mp.paraskeva@edu.cut.ac.cy

George Savva
Cyprus University of Technology
Limassol, Cyprus
gec.savva@edu.cut.ac.cy

Andreas Efstathiou
Cyprus University of Technology
Limassol, Cyprus
andreasefstathiouudt@gmail.com

Edson Ramiro Lucas Filho
Cyprus University of Technology
Limassol, Cyprus
edson.lucas@cut.ac.cy

Jianqiang Shen
Huawei Technologies Co., Ltd.
Shenzhen, China
shenjianqiang@huawei.com

Lun Yang
Huawei Technologies Co., Ltd.
Shenzhen, China
yanglun12@huawei.com

Kebo Fu
Huawei Technologies Co., Ltd.
Shenzhen, China
fukebo@huawei.com

Herodotos Herodotou
Cyprus University of Technology
Limassol, Cyprus
herodotos.herodotou@cut.ac.cy

ABSTRACT

Modern data storage systems are characterized by a distributed architecture as well as the presence of multiple storage tiers and caches. Both system developers and operators are challenged with the complexity of such systems as it is hard to evaluate how a configuration change will impact the workload or system performance and identify the best configuration to satisfy some performance objective. DITIS is a new simulator that models the end-to-end execution of file requests on distributed tiered storage systems that addresses the aforementioned challenges efficiently without any costly system redeployments. The demonstration will showcase the key functionalities and benefits offered by DITIS, including (i) analyzing workload traces to understand their characteristics and the behavior of the underlying storage system; (ii) running simulations with different configurations to evaluate their impact on performance; and (iii) running optimizations over custom search spaces to find the best configuration that satisfies a given objective.

PVLDB Reference Format:
Sotiris Vasileiadis, Matthew Paraskeva, George Savva, Andreas Efstathiou, Edson Ramiro Lucas Filho, Jianqiang Shen, Lun Yang, Kebo Fu, and Herodotos Herodotou. Optimizing Distributed Tiered Data Storage Systems with DITIS. PVLDB, 17(12): 4393 - 4396, 2024.
doi:10.14778/3685800.3685883

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/cut-dicl/ditis-ui.

1 INTRODUCTION

Modern data storage systems exhibit considerable complexity due to their distributed nature and the need to balance data and load across the storage nodes [7]. In addition, these systems incorporate multiple storage tiers, encompassing numerous HDDs and SSDs, along with multiple cache levels of DRAM and NVRAM. Consequently, new data management policies are required to optimize performance and resource utilization. Furthermore, these systems integrate diverse redundancy mechanisms, including replication and erasure coding, to ensure data durability and fault tolerance.

The multifaceted architecture of modern data storage systems necessitates sophisticated management strategies to harness their full potential, for both system developers and operators. For developers, evaluating the impact of new policies for caching, tiering, and other mechanisms is cumbersome and time-consuming as it requires system redeployments. Hence, it is very difficult to explore the design space for promoting changes in the system. For operators, it is challenging to evaluate how their workloads will behave after a system reconfiguration or upgrade as well as determine the best system configuration that will satisfy their objectives.

Simulation presents a logical approach to address the aforementioned challenges. Numerous simulators concentrate on modeling particular aspects of storage systems, including caching and tiering policies [6], scheduling [5], network communication [1], and file system behavior [2]. Some simulators are also available for simulating either single-node multi-tier storage systems [4, 10] or distributed single-tier storage systems [8, 9]. However, none of the current simulators can fully encompass the complexity and nuances of contemporary storage systems that feature multiple storage nodes, diverse storage tiers, and various cache levels.

DITIS [3] is a new comprehensive simulator that models the end-to-end execution of file requests on distributed multi-tier storage systems. The key novelties of DITIS include (i) an architecture based on an adaptation of the actor model instead of the typical event-oriented or process-oriented models; (ii) a machine learning-based initialization process for placing data to the appropriate tier/cache before the simulation begins; and (iii) fine-grained but efficient performance cost models for HDD, SSD, NVRAM, DRAM, and network communications. Moreover, DITIS is extremely configurable with 131 configuration parameters touching all aspects of a storage system (e.g., number of nodes/tiers/caches, device and network

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685883

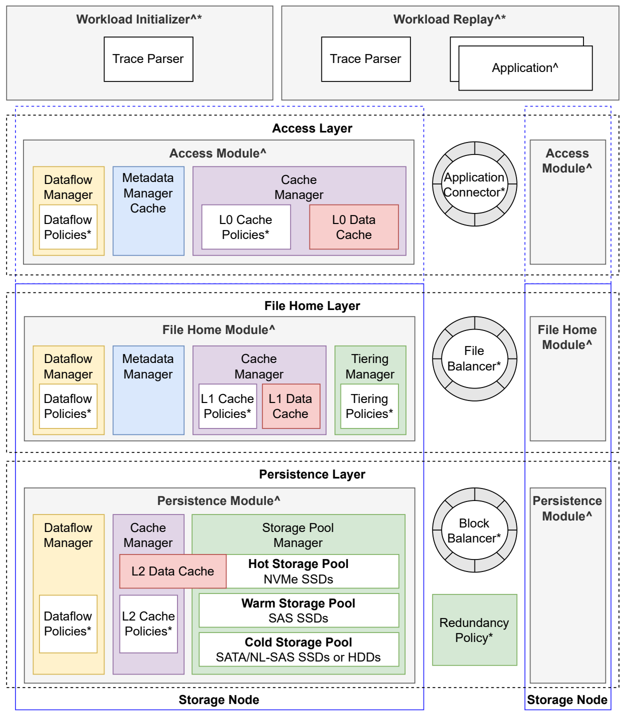


Figure 1: DITIS architecture. Components marked with ^ are actors and with * are pluggable policies.

characteristics, etc.) as well as extensible with 44 pluggable policies controlling all aspects of data flow, caching, and tiering decisions.

This demonstration aims at showcasing the benefits to both storage system developers and operators from using DITIS to:

- Understand workload characteristics and the behavior of the underlying storage system;
- Evaluate the impact of different storage configuration setups and policies to the workload performance; and
- Optimize the storage configuration to satisfy a workload or system objective such as maximizing throughput or cache hit ratio.

2 DITIS SIMULATOR

DITIS is a simulator designed for distributed and tiered file-based storage systems, capable of handling up to three storage tiers plus three levels of caches. It allows for the configuration of various storage media devices such as HDD, SSD, NVRAM, and DRAM, each with specific performance characteristics. The simulator calculates the duration of I/O requests for each tier, cache, and network using detailed performance cost models for each device type.

2.1 System Architecture

DITIS requires two inputs to simulate a workload execution on a storage system: (1) a text file containing a workload trace with file requests containing details like process ID, timestamp, file operation, offset, length, file size, and original duration; and (2) a storage configuration that defines the storage system's behavior and structure, allowing for the customization of components of the storage system, specifying their performance characteristics, and determining which policies to use during the simulation. The main system components of DITIS, shown in Figure 1, are:

- **Workload Initializer**: Sets up the initial state of the system (e.g., creates pre-existing files) before the trace is executed.
- **Workload Replay**: Controls the order and timing of submitting and simulating the file requests from the workload trace.
- **Application**: Represents external applications (processes) that submit file requests to the storage system in parallel.
- **Application Connector**: Balances incoming application connections to the available Access Modules.
- **Access Module**: Represents a storage system's access component running on a storage node or a system's client.
- **File Balancer**: Distributes files and file requests to File Home Modules based on full file paths.
- **File Home Module**: Maintains a partition of the system's namespace on a storage node.
- **Block Balancer**: Distributes data blocks and block requests to Persistence Modules based on a block IDs (or addresses).
- **Persistence Module**: Stores and processes data blocks in tiered storage pools on a storage node.
- **Redundancy Policy**: Controls the redundancy policy of blocks (e.g., using erasure coding, replication, or RAID).

DITIS generates an output trace, accurately capturing the sequence of file requests from the input trace but with a simulated duration for each operation. Alongside, DITIS produces a detailed report containing information and statistics about the workload execution and the storage system (described in Section 3).

2.2 Simulation Process

Following the Actor Model, the key components of the system are treated as actors (marked with ^ in Figure 1). These actors maintain their own private state, process messages received from other actors, and send messages to other actors enabling asynchronous message exchange, while respecting a simulation clock. This approach ensures the sequence and timing of events accurately reflect those of a real distributed multi-threaded system. In DITIS, the process for handling I/O requests begins when an Application connects to an Access Module and sends a file request (e.g., a read request) to the storage system. If the request can be served by the Access Module (e.g., perform the read from the local cache), a respond is sent back to the Application. Otherwise, the request is forwarded to the appropriate File Home Module for processing. The request can be completed there or forwarded as one or more block requests to the appropriate Persistence Module(s). There, the request is served from the appropriate storage pool and returned upstream. While the requests are getting processed, all relevant decisions are made by pluggable policies, such as cache admission, eviction, and prefetching, tier migrations, block redundancy, etc. At each step, the duration is calculated based on several pluggable performance cost models, while taking into account the potential concurrent execution of requests as well as resource contention.

2.3 Performance Cost Modeling

The performance cost models for all devices and network are designed to be pluggable, thus making it easy to replace to accommodate different simulation needs or advancements in technology.

Hard Disk Drive (HDD) modeling takes into account the seek time, rotation time, and transfer time of the device as well as the presence of a queue containing concurrent pending requests. In addition, it differentiates between sequential and random I/O requests for more accurate cost modeling.

The Solid State Drive (SSD) model focuses on the steps involved in processing data requests, including the decoding of I/O requests, reading data from the flash array into the data register, and transferring this data to the I/O bus. It accounts for both single-page reads and the handling of multiple pages. This model also considers the queuing of requests and calculates wait times based on the current load and pending operations.

Accessing data in DRAM involves multiple steps counted in clock cycles. DRAM manufacturers specify the clock cycles needed for these actions, including the number of clock cycles between getting a request and having the data ready, the cycles to open a new row, and the duration a row stays open for writing.

The network model accounts for network bandwidth for TCP connections, considering packet losses due to network congestion. It utilizes maximum segment size, round trip time, and packet loss probability to determine transfer latency. In the presence of concurrent transfers, the bandwidth is shared and the model adjusts for the number of active connections to estimate data transfer rates.

3 DEMONSTRATION PLAN

The demonstration will showcase the key functionalities and benefits offered by DITIS. The key functionalities include (i) uploading and analyzing workload traces, (ii) creating storage and optimization search space configurations, (iii) running simulations using different storage configurations and traces, and getting extensive reports about the simulation run, and (iv) running optimizations using different storage configurations and custom search spaces to find the best storage configuration that meets a given objective. For the purposes of the demonstration, we will run DITIS on a laptop, use production traces provided by Huawei Technologies, and execute realistic scenarios as the ones described below.

3.1 Workload Analysis

The first demonstration part will show how a user can use DITIS to gain insights for a given workload. DITIS supports uploading workload traces of different formats, including SNIA IOTTA file traces [11] as well as a custom format used by Huawei Technologies. Upon uploading a new trace, users are able to visualize a detailed analysis of their workload (shown in Figure 2), comprising: (i) a distribution of the request types (e.g., read, write) appearing in the workload; (ii) important performance metrics such as IOPS, throughput, and latency per request type; (iii) the interleaving and duration of concurrent applications that appear in the workload; and (iv) a histogram of file sizes for the files accessed in the trace.

3.2 Simulation Analysis

Motivated by the real needs of both developers and operations teams, DITIS can be used to evaluate the impact of different storage setups and policies to a workload. For example, developers often need to study how different caching or tiering policies will affect

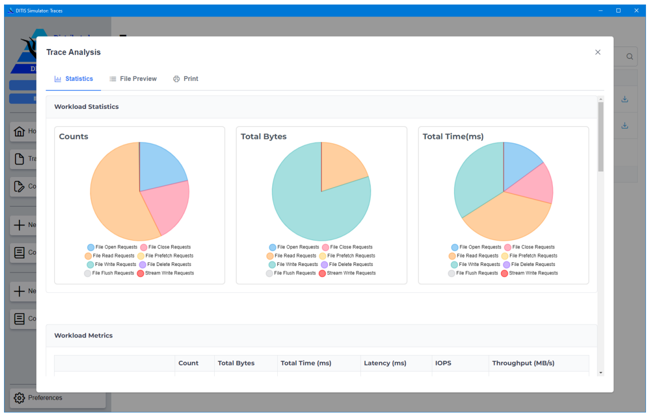
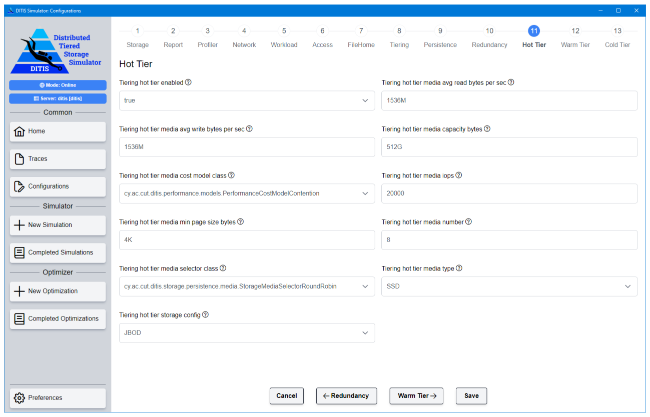


Figure 2: Workload trace analysis view in DITIS



Hot Tier

Tiering hot tier enabled: true	Tiering hot tier media avg read bytes per sec: 1536M
Tiering hot tier media avg write bytes per sec: 1536M	Tiering hot tier media capacity bytes: 512G
Tiering hot tier media cost model class: cy.ac.cut.ditis.performance.models.PerformanceCostModelContention	Tiering hot tier media iops: 20000
Tiering hot tier media min page size bytes: 4K	Tiering hot tier media number: 8
Tiering hot tier media selector class: cy.ac.cut.ditis.storage.persistence.media.StorageMediaSelectorRoundRobin	Tiering hot tier media type: SSD
Tiering hot tier storage config: JBOD	

Figure 3: Storage configuration form in DITIS

the performance of the system under different workloads in order to enhance user experience. Conversely, operations teams are urged to evaluate various system adjustments, such as incorporating additional SSDs or expanding the memory cache, to improve the workload performance and scalability. By adjusting the storage configuration, users can attain their desired objectives while also exploring diverse scenarios, all without the requirement of physically possessing the hardware. This approach offers a versatile and streamlined testing environment, enabling users to experiment with different setups and fine-tune their systems effortlessly.

A user can configure the simulator to match an existing or hypothetical storage system by using the configuration form of the interface shown in Figure 3. The form includes 131 parameters organized in 13 categories configuring all aspects of a storage system such as the network (e.g., bandwidth of frontend and backend network), caches (e.g., size, performance characteristics, cache policies), storage tiers (e.g., number of devices per tier, tiering policies), storage devices (e.g., type, size, performance characteristics), and redundancy (e.g., erasure coding, replication) among others. Configurations can then be saved and used for running simulations in order to understand how the workload and the system would behave under those circumstances.

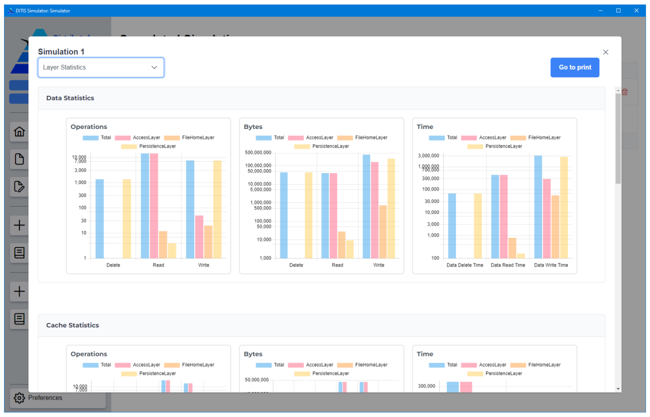


Figure 4: Simulation analysis report in DITIS

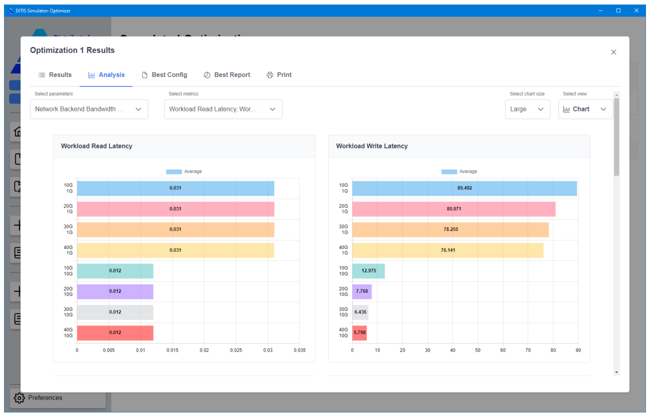


Figure 5: Optimization analysis results in DITIS

Running a simulation yields a comprehensive analysis report, presenting various types of analyses, views, and graphs (e.g., pie, bar, sankey) to provide insights at different levels of granularity as shown in Figure 4. An indicative (non-comprehensive) list of insights presented in the simulation analysis includes:

- Operation statistics (e.g., count, bytes, and time per operation) for the overall workload and for each application, storage layer, storage node, and module.
- Performance metrics (e.g., latency, throughput, IOPS) for the overall workload and for each storage layer, node, and module.
- Cache statistics (e.g., hit ratio, byte hit ratio, precision, recall) for cache admission, eviction, and prefetching for each cache.
- Data migration statistics revealing the distribution of counts and bytes among the different storage tiers.
- Runtimes for each application, the overall workload, and the various simulation stages.

3.3 Optimization Analysis

The primary goal of optimization analysis is to find the best storage configuration that satisfies a workload or system objective. This objective may vary depending on the real needs of developers and operators. From a developer's perspective, for example, one aim is to identify the best default migration up and down policies that maximize system throughput. On the other hand, an operator might be interested in finding the best combination of fault tolerance configuration and number of drives to minimize read latency. Currently, DITIS supports 15 optimization objectives including maximizing throughput and IOPS or minimizing latency.

With DITIS, users have the ability to define their own custom search space for the optimization process. In particular, users can select which configuration parameters they would like to include in the search space, and for each parameter, they can define either a list of values (e.g., LRU, LFU, and EXD for cache eviction in the Access Layer), an arithmetic sequence (e.g., {3, 5, 7} storage nodes), or a geometric sequence of values (e.g., {4, 8, 16} SSDs in the warm storage tier). DITIS will then enumerate the full search space as the Cartesian product of all parameter values and will use one of three available search algorithms to explore the space: grid search, recursive random search, and genetic algorithm. Hence, users have full flexibility on both how to define and how to search the parameter space of their interest.

Running an optimization generates a detailed comparative report as well as identifies the storage configuration that optimizes the desired objective. The report contains a table with all the configurations that were executed along with a long list of performance metrics per run (e.g., latency, throughput, cache hit ratio). The user can also perform a slice and dice analysis to examine the results from different viewpoints as shown in Figure 5. Several tables, bar charts, and box plots are generated according to user-selected parameters that summarize the performance metrics showing distribution of values, min/max values, as well as median and mean values. This process enables users to understand the impact of the various configuration parameter values on the performance of the workload and storage system.

REFERENCES

[1] Kenenbek Arzymatov, Andrey Sapronov, Vladislav Belavin, Leonid Gremyachikh, Maksim Karpov, et al. 2020. SANgo: A Storage Infrastructure Simulator with Reinforcement Learning Support. *PeerJ Computer Science* 2020, 5 (2020), 1–16.

[2] Bo Feng, Ning Liu, Shuibing He, and Xian He Sun. 2014. HPIS3: Towards a High-performance Simulator for Hybrid Parallel I/O and Storage Systems. In *Proc. of the 9th Parallel Data Storage Workshop*. IEEE, 37–42.

[3] Edson Ramiro Lucas Filho, Lambros Odysseos, Lun Yang, Kebo Fu, and Herodotos Herodotou. 2022. DITIS: A Distributed Tiered Storage Simulator. *Infocommunications Journal* 14, 4 (2022), 18–25.

[4] Sebastien Gougeaud, Soraya Zertal, Jacques Charles Lafoucriere, and Philippe Deniel. 2016. A Generic and Open Simulation Tool for Large Multi-Tiered Hierarchical Storage Systems. *Simulation Series* 48, 8 (2016), 91–98.

[5] Julien Monniot, François Tessier, Matthieu Robert, and Gabriel Antoniu. 2022. StorAlloc: A Simulator for Job Scheduling on Heterogeneous Storage Resources. In *European Conference on Parallel Processing*. Springer, 211–222.

[6] Louis Marie Nicolas, Luis Thomas, Yassine Hadjadj-Aoul, and Jalil Boukhobza. 2022. SLRL: A Simple Least Remaining Lifetime File Evicition Policy for HPC Multi-tier Storage Systems. In *Proc. of the Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (CHEOPS)*. ACM, 33–39.

[7] Junpeng Niu, Jun Xu, and Lihua Xie. 2018. Hybrid Storage Systems: A Survey of Architectures and Algorithms. *IEEE Access* 6 (2018), 13385–13406.

[8] Alberto Núñez, Javier Fernández, Jose D. Garcia, Laura Prada, and Jesús Carretero. 2008. SIMCAN: A Simulator Framework for Computer Architectures and Storage Networks. In *Proc. of the 1st Intl. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, SimuTools*. ICST/ACM, 73.

[9] Ramya Prabhakar, Erik Kruus, Guanlin Lu, and Cristian Ungureanu. 2011. EEffSim: A Discrete Event Simulator for Energy Efficiency in Large-scale Storage Systems. In *IEEE Intl. Conf. on Energy Aware Computing (ICEAC)*. IEEE, 1–6.

[10] Cesar San-Lucas and Cristina L. Abad. 2016. Towards a Fast Multi-tier Storage System Simulator. In *Ecuador Technical Chapters Meeting (ETCM)*. IEEE, 1–5.

[11] SNIA 2024. SNIA IOTTA Repository. http://iotta.snia.org/. Accessed: 2024-03-24.