

Knowledge-Driven Hybrid SSD Management Enhanced by Fine-Tuned LLMs

Qian Wei
Shandong University
Qingdao, Shandong, China

Yi Li
University of Texas at Dallas
Richardson, Texas, United States

Zehao Chen
Shandong University
Qingdao, Shandong, China

Tianren Zhou
Shandong University
Qingdao, Shandong, China

Zhaoyan Shen*
Shandong University
Qingdao, Shandong, China

Dongxiao Yu
Shandong University
Qingdao, Shandong, China

Bingzhe Li
University of Texas at Dallas
Richardson, Texas, United States

Abstract

A hybrid Solid-State Drives (SSDs) integrates different modes of flash cells (e.g., single-level cell (SLC) and Quad-Level Cell (QLC)) and enables them to convert between each other, achieving both high performance and storage capacity. However, this hybrid design introduces a significantly larger design space than traditional SSDs with additional design factors such as flash conversion and data migration across different flash modes, leading to higher optimization complexity. Efficient management of such complexity requires deep hybrid SSD knowledge and dynamic adjustment mechanisms. Large language models (LLMs) offer a promising solution through their contextual reasoning and adaptive coordination capabilities.

In this work, we explore the potential of using LLMs in understanding and efficiently managing hybrid SSD design space. We find that leveraging LLMs for knowledge-guided optimization of management parameters enables substantial performance gains. Building on these insights, we propose LLM-hybridSSD, an integrated optimization framework that formulates hybrid SSD management as a parameter-tuning problem, employs an LLM-based tuner for adaptive configuration, and applies reinforcement learning-based fine-tuning to align local lightweight models with domain-specific knowledge. Experimental results show an average 58.92% increase in throughput and a 28.56% reduction in write amplification (WA) compared with state-of-the-art schemes under different real-world workloads.

Keywords

Hybrid SSD Management, LLM, Fine-tuning

ACM Reference Format:

Qian Wei, Yi Li, Zehao Chen, Tianren Zhou, Zhaoyan Shen, Dongxiao Yu, and Bingzhe Li. 2026. Knowledge-Driven Hybrid SSD Management Enhanced by Fine-Tuned LLMs. In *63rd ACM/IEEE Design Automation Conference (DAC '26)*, July 26–29, 2026, Long Beach, CA, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3770743.3803999>

1 Introduction

Hybrid SSDs serve as a versatile storage solution, leveraging the integration of multiple modes of flash memory cells, including SLC and QLC, to simultaneously optimize storage capacity and

performance. To achieve dynamic flexibility, hybrid SSDs enable flash cells to switch between different operational modes [10, 11], allowing the system to adapt to diverse application requirements. This mechanism increases the complexity of flash control logic and elevates both computational and memory demands. In response to these challenges, host-managed SSDs expose the physical data layout to the resource-rich host system, providing a more efficient approach to coordinating flash management.

Despite the abundant system resources of host-managed SSDs, the integration of multiple flash modes still presents significant management challenges across both the hardware and application layers. At the hardware level, dynamic flash cell mode conversions must be carefully coordinated with space allocation and garbage collection. Improper timing or granularity in these conversions can lead to performance degradation and underutilized capacity. On the application side, rapidly changing workload patterns alter data access locality, making it difficult for the system to adaptively place data in the most suitable flash modes [23].

To overcome these challenges, Artificial Intelligence (AI) techniques have been increasingly employed to optimize hybrid SSD management. At the hardware level, Yoo et al. [30] and RL-cSSD [27] utilize machine learning to optimize mode conversion and garbage collection based on internal device states, aiming to strike a balance between performance and endurance. On the application side, HAML [14] leverages kmeans-based hotness classification to guide data placement across different flash modes, effectively enhancing performance stability of hybrid SSD.

However, current AI-driven optimization methods face several inherent limitations. First, many existing AI-based schemes heavily depend on manually designed optimization logic. Domain experts must possess deep knowledge of hybrid SSDs, including aspects such as data placement and mode conversion, and often engage in extensive trial and error to design algorithm parameters. This reliance on expert input leads to high deployment costs and increased complexity. Second, the inherently complex and dynamic nature of hybrid SSDs limits the generalizability of existing AI models. As upper-layer application workloads and internal device states continuously evolve due to shifting access patterns, the lack of dynamic adjustment mechanisms results in inefficient or even incorrect management decisions, ultimately compromising performance and robustness in real-world deployments.

The emergence of LLMs provides new opportunities for hybrid SSD management optimization. Models such as GPT [19] and LLaMA [25] possess strong contextual understanding and knowledge transfer ability, enabling rapid adaptation to diverse scenarios with minimal interactions for more efficient optimization. In addition, the reasoning capabilities of LLMs support cross-dimensional

*Zhaoyan Shen is the corresponding author. Email: shenzhaoyan@sdu.edu.cn.



This work is licensed under a Creative Commons Attribution 4.0 International License. DAC '26, Long Beach, CA, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2254-7/2026/07

<https://doi.org/10.1145/3770743.3803999>

decision-making [2, 12, 24], allowing the generation of globally informed strategies even under interdependent system complexities.

In this paper, we aim to leverage the capabilities of LLMs to enhance the management efficiency of hybrid SSDs. We begin with a preliminary exploration of applying LLMs by directly replacing existing AI-based optimization schemes. The results show that although LLMs demonstrate a basic understanding of system states and management objectives, such direct substitution yields no performance gain while incurring high cost. These findings reveal the need for a more structured integration of LLMs into hybrid SSD management, raising two key challenges: **(1) How can LLMs be effectively integrated into the appropriate layer of hybrid SSD management?** Using LLMs directly is ineffective without clear guidance on what to adjust and at what granularity, making the definition of LLM-driven optimization scope a key challenge. **(2) How to integrate certain domain Hybrid SSD knowledge as-sociate with LLM management to preserve both performance and cost efficiency?** Large-scale LLMs deliver strong reasoning capability but are costly to deploy, while lightweight models offer lower overhead but limited understanding of SSD-specific behaviors, making the integration of domain knowledge for efficient and accurate optimization a key challenge.

To address the first challenge, we leverage LLMs to enhance existing hybrid SSD management by formulating the optimization task as a structured parameter-tuning problem with a clearly defined configuration space. Parameters are categorized into SSD-specific, workload-related, and AI strategy-related groups, and performance-sensitive parameters are selected to reduce tuning complexity while maximizing optimization effect. Based on the formulation, we design an LLM-based tuner that transforms normalized workload and device-state information into structured prompts and generates configuration recommendations. The generated outputs are validated through constraint checking to ensure correctness.

To address the second challenge, we integrate local lightweight LLM deployment with fine-tuning techniques to achieve competitive management performance at low cost. We employ a Guided Reinforcement Policy Optimization (GRPO) mechanism that enables the lightweight model to continuously learn from runtime feedback and progressively internalize hybrid SSD-specific knowledge. This process produces domain-specialized LLMs that not only understand hybrid SSD behaviors more accurately but also retain low computational and storage footprint.

Building on the above designs, we develop an integrated optimization framework, named LLM-hybridSSD. The framework integrates task formulation, configuration tuning, and lightweight model adaptation into a unified LLM-based feedback loop. Experimental results show that LLM-hybridSSD significantly enhances performance, increasing throughput by 58.92% while reducing WA by 28.56% under real-world traces compared with SOTA schemes.

The contributions of this work are summarized as follows:

- We formalize hybrid SSD management as a structured parameter tuning problem, introducing a well-defined configuration space that categorizes SSD-specific, workload-related and AI strategy-related parameters while identifying performance critical factors to reduce tuning complexity.
- We design and fine-tune lightweight LLMs tailored for cost-constrained hybrid SSD environments, enabling efficient deployment and continuous learning through a Guided Reinforcement Policy Optimization mechanism.

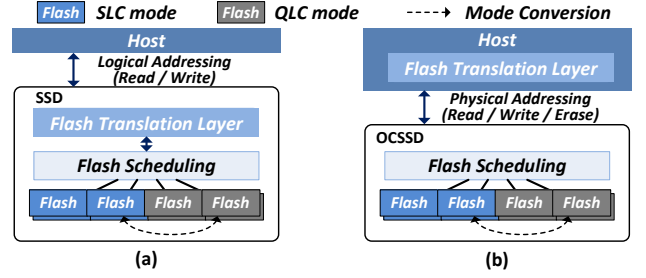


Figure 1: The core module of (a) traditional SSD (b) host-managed SSD.

- We develop LLM-hybridSSD, an integrated optimization framework that unifies task formulation, configuration tuning, and local lightweight model adaptation into a closed-loop LLM-enhanced hybrid SSD management system.
- We conduct comprehensive evaluations under real-world workloads, demonstrating that LLM-hybridSSD substantially outperforms state-of-the-art schemes.

The remainder of this paper is organized as follows: § 2 provides the background. § 3 explores the potential of LLMs in hybrid SSD management. § 4 presents the LLM-hybridSSD framework, and § 5 discusses experimental results. Finally, § 6 concludes this paper.

2 Background

2.1 Hybrid SSDs

A flash-based SSD consists of multiple flash memory chips and a controller that handles address mapping, wear leveling, and garbage collection through the Flash Translation Layer (FTL) [9, 27, 32], as illustrated in Figure 1. To improve flexibility and reduce device-side complexity, host-managed SSDs shift FTL management responsibilities to the host operating system [13].

Flash memory stores multiple bits per cell through electron injection, enabling configurations such as SLC and QLC. While higher-density technologies like QLC provide substantially increased storage capacity, they suffer from lower endurance and degraded performance. Hybrid SSDs integrate multiple flash modes within a single device (e.g., SLC and QLC), and their cells can be reconfigured between modes [17, 23], allowing the system to flexibly trade off between performance and capacity. However, managing such hybrid architectures is challenging because data must be dynamically placed and migrated across regions to sustain both efficiency and device longevity.

AI-based optimization schemes have been proposed for hybrid SSD management, aiming to model workload access behaviors and improve data placement and migration. HAML [14] exemplifies this direction by using k-means clustering on update frequency and access intervals to group data with similar hotness, reducing interference from varying workloads. From another perspective, RL-based schemes target adaptive control of flash-mode conversion. RL-cSSD [27] leverages environment feedback to refine conversion timing and alleviate performance degradation caused by inefficient mode conversions.

2.2 Large Language Models

Large language Models (LLMs), such as GPT [19], Meta LLaMA [25] and Deepseek [3], demonstrate strong reasoning and generalization capabilities, enabling their application beyond language tasks.

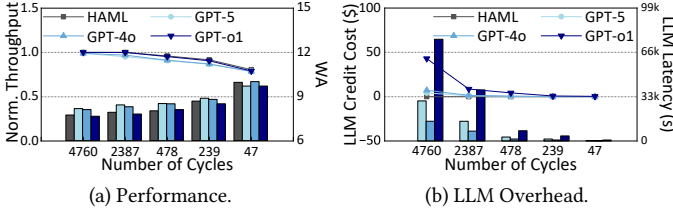


Figure 2: Performance (left axis, lines) and cost (right axis, bars) with different reclassification cycles.

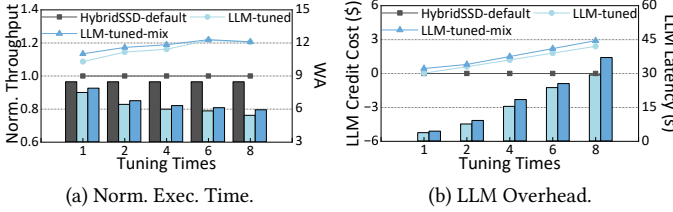


Figure 3: Performance (left axis, lines) and cost (right axis, bars) of different tuning schemes.

By modeling complex relationships and high-dimensional objectives [5], LLMs offer new opportunities for intelligent configuration and adaptive management of large-scale computing systems.

LLMs are typically applied to system and application optimization through two deployment paradigms: cloud-based and local deployment [26, 31]. Cloud-based solutions leverage centralized resources and offer convenient access, but incur recurring costs, raise potential privacy concerns, and often rely on closed-source models. Local deployment enhances data security, avoids network latency, and can utilize open-source models for greater control, but requires substantial hardware resources and user-managed maintenance.

Fine-tuning has become a central technique for adapting LLMs to downstream tasks. Full-parameter fine-tuning [4] offers strong accuracy but incurs high computational cost, while parameter-efficient methods such as adapters [6], prefix-tuning [16], and LoRA [7] adjust only small subsets of weights for efficiency. Further, reinforcement learning-based tuning [20, 22] further aligns model behavior with desired objectives through feedback-driven optimization.

3 Optimization Potential of LLMs

In this section, we conduct preliminary experiments to examine the potential of LLMs in hybrid SSD management, focusing on whether they can replace existing AI-based approaches or further enhance them through parameter tuning. The evaluation setup follows the configuration described in § 5.

3.1 Handling Primary Management by LLMs

To evaluate whether LLMs can handle fundamental hybrid SSD management tasks, we replace the ML-based data hotness classification scheme (HAML [14]) with advanced large-scale LLMs (i.e., GPT-5, GPT-4o, and GPT-o1) via API calls on the host-managed SSD. We serve the trace data (i.e., prn_0 [18]) to each model and request it to generate a data hotness classification file, which is then used to guide subsequent data placement decisions.

Figure 2 shows the results with different reclassification cycles. As shown in Figure 2 (a), under frequent cycles, LLM-based schemes provide similar performance to ML-based schemes, with normalized throughput fluctuation within 2.3%. This reveals that LLM

can effectively handle the hotness classification task. However, reducing the reclassification frequency significantly degrades performance for both ML-based and LLM-based schemes, since infrequent hotness-classification updates fail to capture rapid workload changes. Meanwhile, as shown in Figure 2 (b), although the performance is comparable, the LLM-based approach incurs notable credit cost (up to \$42.97) and latency (up to ~20 hrs) overhead, making direct deployment impractical.

In summary, large-scale LLMs are trained with hybrid SSD knowledge and achieve as good performance as traditional schemes. However, directly applying LLMs to hybrid SSD management (e.g., simply replacing existing schemes) does not yield significant performance gains while introducing unacceptable cost.

3.2 Transferring SSD Tuning into LLMs

Since directly replacing existing schemes with LLMs does not yield substantial performance improvement, we instead investigate how LLMs can be used to assist existing algorithms. Specifically, we explore whether LLMs can enhance current ML-based strategies by tuning their key parameters. Hybrid SSD management involves a vast and complex parameter space covering numerous configurable aspects. These parameters, such as garbage collection thresholds and data migration frequency, work together to guide system behavior and ensure stable and efficient operation.

To ground our study, we select HAML, a state-of-the-art ML-based design, as the baseline method. we then compare three configurations: *HAML*, which uses static parameters; *LLM-tuned*, where best-performing GPT-o1 dynamically adjusts HAML-related parameters such as retraining interval and K-means thresholds; *LLM-tuned-mix*, which expands the tuning space to include additional GC/MC granularity parameters.

Figure 3 shows the performance and cost of these three scheme with different tuning times under the prn_0 workload [18]. As shown, the LLM-tuned configurations achieve noticeable improvements over HAML. Specifically, LLM-tuned configuration improves throughput by 16.28% and decreases WA by 15.2%, indicating that LLMs effectively generate parameter adjustments. When the parameter space expands in LLM-tuned-mix, performance further improves, with throughput increasing by 20.47% and WA dropping by 18.7% compared with HAML. This result demonstrates that LLMs can adapt flexibly to an expanding parameter pool, refining configurations and enhancing system performance accordingly. However, the tuning process also incurs additional overhead due to repeated LLM invocations. In our experiments, the LLM-based approach incurs credit cost (up to \$2.92) and latency (up to ~36.9 s) overhead, partially offsetting the performance gains.

In summary, LLMs show strong potential to optimize hybrid SSD management through dynamic parameter-tuning strategies. However, realizing their full potential requires a well-structured tuning framework that balances optimization efficiency and cost.

4 System Design

4.1 System Overview

In this work, we propose LLM-hybridSSD, a knowledge-driven hybrid SSD management framework that integrates an LLM-based tuner into the SSD management stack to enable adaptive parameter tuning under dynamic workloads, as shown in Figure 4.

The framework consists of three main components. The *hybrid SSD management stack* provides the core management control for hybrid SSDs. Within this stack, we design a sliding window-based

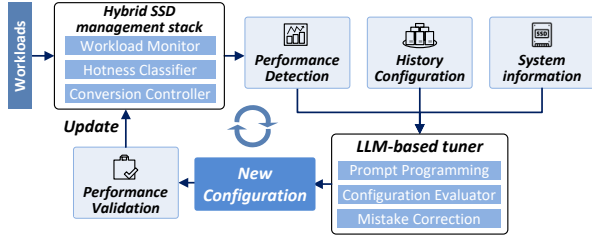


Figure 4: The overall framework of LLM-hybridSSD.

mechanism that tracks access patterns and detects variations to support management decisions. The *LLM-based tuner* employs a fine-tuned local deployed LLM to adjust the carefully designed hybrid SSD configuration space and produce parameter updates. The *performance validation module* supplies feedback to the tuning loop based on short-term system behavior.

The framework operates as a closed feedback loop. At each tuning interval, runtime statistics are collected and encoded as inputs to the LLM-based tuner. The tuner produces a candidate configuration that is applied for a short evaluation window. The performance validation module then measures the corresponding execution metrics and compares them with those from the previous interval. If the candidate configuration satisfies the acceptance criterion, it is adopted as the new management policy. Otherwise, the system restores the previous configuration and may request additional candidates.

4.2 Problem Formulate

In this work, we do not modify the SSD hardware design itself, in contrast to AutoBox [15] which targets hardware configuration during device development. Instead, given a fixed hardware design, we focus on the runtime control policy and formulate it as a structured parameter-tuning problem. This formulation identifies the dynamically adjustable software parameters that meaningfully influence system behavior, enabling the LLM to operate within a well-defined configuration space rather than over raw device states. We group the tunable parameters into three main categories:

- **SSD-specific parameters:** These parameters are inherent to SSD firmware, controlling flash memory allocation and conversion. They are crucial for optimizing performance and enhancing device stability. Typical examples include GC threshold and granularity.
- **Workload-related parameters:** These parameters are used to collect data access patterns and characteristics, which help hybrid SSD management policies better understand current workloads. A typical example is the workload sliding window size.
- **AI Strategy-related parameters:** These parameters are key to AI-driven algorithms. Specifically, they tune the behavior of these strategies, such as ML-based hotness classification and RL-based space management, ensuring better performance and reliability. Examples include K-means max iterations and RL training interval.

Initially, we consider a broad set of parameters spanning these categories. However, including all parameters can be counterproductive, as an large parameter space may exceed the context window of LLM and tuning irrelevant parameters may introduce noise, reducing convergence efficiency and stability [24]. Therefore, we focus on identifying the most performance-sensitive parameters.

Through offline workload analysis, we evaluate the impact of each parameter across different workload scenarios. Parameters with minimal influence, such as the K-means convergence tolerance (tol) whose variations seldom affect clustering outcomes or

Table 1: Sensitive Parameters with Default Values

Category	Parameter Name	Default Value
SSD-specific	Conversion granularity	1
	Conversion trigger threshold	6
	GC granularity	1
	GC trigger threshold	6
	Data placement strategy	SLC first
Workload-related	Window size	2000
	Standard deviation threshold	10000
	Slice size	200MB
	K-means max iterations	10
AI Strategy-related	K-means trigger threshold	10000
	RL training interval	1000
	RL learning rate	0.1
	RL reward	1.6ms
	RL discount factor	0.9
	RL exploration rate	0.1

overall performance, are excluded from consideration. Based on this sensitivity analysis, we select 15 key parameters from an initial set of 32 across the three categories. The selected parameters and their default values, derived from existing strategy settings, are summarized in Table 1.

4.3 LLM-based Tuner of SSD Configurations

This subsection mainly introduces the working flow of LLM-based tuner. There are three major phases: 1) Prompt Engineering, 2) Configuration Generation, and 3) Mistake Correction.

Prompt Engineering. The prompt engineering phase is used to provide detailed and accurate background information for LLM to generate effective responses. The prompt is organized into five key stages. First, **①Role Assignment** is carried out by designating the LLM as an SSD expert. Then, during **②Hybrid SSD Overview** and **③SSD Management**, the LLM is provided with system information, which is categorized into the inherent configuration and the current optimization strategies. In **④Historical Configs & Performance**, a detailed list of past configuration changes, their reasons, and performance impacts is included. Finally, **⑤Specify Requirements** ensures the LLM generates a new configuration list with improved accuracy by defining a clear output format. We give an example of the prompt and its corresponding output:

Input: ① You are an SSD Expert. You are being consulted to improve the SSD configuration by optimizing options file based on system information and benchmark output. ② The SSD consists of SLC and QLC modes... <SSD Settings>. ③ The current SSD scheme is as follows: (1) The hotness classification for is using the K-means algorithm... ; (2) A space management scheme based on Q-learning is employed... ; ...; ④ First, the historical performance and configuration changes are shown. The point in time configuration changes are as follows: ..., The performance is: ... The current option file is: ... ⑤ Based on these information generate a new file in the same format as the options_file to improve the SSD throughput and WA. Enclose the new options file in ‘ ‘.

Output: New configuration:
‘1. K-means trigger threshold: 1000; 2.Windows size: 1500; ...’
Reason: ...

The input prompt size typically remains within the context limits of mainstream LLMs. To further ensure safety under iterative adjustments, we adopt a sliding window approach, splitting long

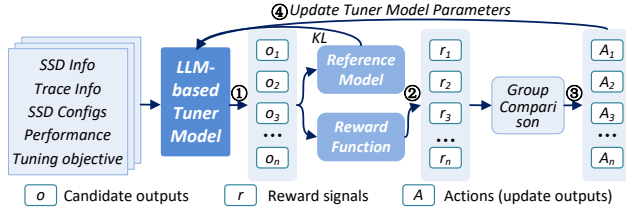


Figure 5: Fine-Tuning workflow under the GRPO strategy.

texts into overlapping segments while retaining the ending of the previous segment to maintain context and avoid information loss.

Configuration Generation. The configuration generation phase is used to transform the natural language output of LLM into a machine-readable format. This module extracts two key components: the reason for the adjustment and the updated configuration list. The historical updates are recorded for future tuning reference.

Mistake Correction. The mistake correction phase validates the updated configuration through format checking and numerical range checking. For format checking, we verify that each configuration matches the correct format and remove any irrelevant descriptions. For numerical range checking, parameter thresholds are set to maintain the correctness of the ranges, preventing performance degradation due to excessive tuning.

4.4 RL-driven Fine-Tuning of Lightweight LLMs

As discussed in § 3.2, deploying large-scale LLMs for hybrid SSD management introduces excessive cost, making them unsuitable for real-world deployment. To address this, we adopt local lightweight models that can operate entirely on a single machine. However, their limited capacity results in insufficient hybrid SSD domain knowledge, causing inaccurate decisions and performance deviation. To address this issue, we fine-tune the lightweight model to help it better understand hybrid SSD domain knowledge and effectively handle management tasks. Specifically, we adopt the Guided Reinforcement Policy Optimization (GRPO) strategy [20] to perform this fine-tuning process.

The fine-tuning workflow is illustrated in Figure 5. ① The LLM-based tuner proposed in § 4.3 receives system information, including SSD info, trace info, configurations and tuning objectives, and generates multiple candidate configuration files ($o_1, \dots, o_n$) representing potential configuration adjustments. ② Each candidate file is executed within a short evaluation to obtain real SSD performance feedback. Based on the results, a frozen reference model and a reward function compute corresponding reward signals ($r_1, \dots, r_n$). ③ GRPO performs group-wise comparison to identify the candidates with the highest rewards and generates new performance-friendly configuration updates ($A_1, A_2, \dots, A_n$). ④ The reward signals and the differences among high-reward candidates are then used to update the tuner model parameters, completing the optimization loop. Through this iterative process, the tuner model internalizes hybrid SSD domain knowledge and applies it to balance performance goals and resource constraints, achieving adaptive and stable management under diverse workload and device conditions.

5 Experimental Results

5.1 Environment Setup

Hardware Platform. We evaluate LLM-hybridSSD using a trace-driven hybrid SSD simulator based on SSDsim [8]. The simulator

Table 2: SSD Configurations

Parameter	Value	Parameter	Value
Capacity	256GB	Write latency (SLC mode)	200us
#Channels	32	Write latency (QLC mode)	2ms
Page size	16KB	Read latency (SLC mode)	20us
Over-provisioning ratio	12.5%	Read latency (QLC mode)	140us
Pages/block (SLC mode)	256	Erase latency (SLC mode)	3ms
Pages/block (QLC mode)	1024	Erase latency (QLC mode)	3.5ms

Table 3: Workload Characteristics

Category	Workloads	Read request (GB)	Write request (GB)
MSR [18]	prn_0	13.12	45.96
	usr_0	35.3	13.08
FIU [1]	homes	5.89	39.6
OLTP [21]	Financial	2.65	14.57
KV store [29]	ssdtrace	199.36	46.6
Mix	mixed_trace	58.38	79.52

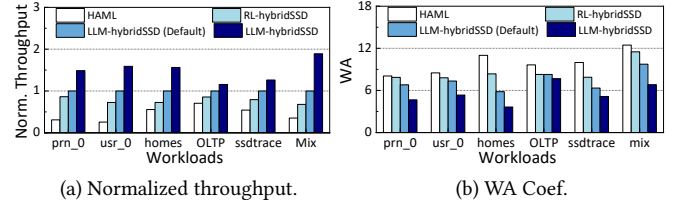


Figure 6: Overall performance with different workloads.

models hybrid architectures, incorporating SLC-QLC mode conversion, data allocation, and garbage collection. The SSD is configured with a 256 GB capacity and a 16 KB page size, with detailed setups provided in Table 2. To ensure comprehensive coverage of all possible scenarios and to expedite the testing process, the available SSD space is limited to 32 GB during the experiments. LLM-hybridSSD is executed on a server equipped with 48 Intel Xeon CPUs operating at 2.1 GHz, where both SSD management and the LLM-based tuner run on the host CPU, adhering to the host-managed SSD paradigm.

Traces and Baselines. We use 5 different workloads (shown in Table 3), which cover various scenarios including KV stores, research, financial, etc. Furthermore, to simulate more complex scenarios involving multiple applications and frequent load variations, we construct a mixed trace, denoted as "Mix", by combining several individual traces (i.e., prn_0, usr_0, and hm_0). We compare the proposed LLM-hybridSSD with three other hybrid SSD designs: HAML [14], RL-cSSD (RL-hybridSSD) [27] and LLM-hybridSSD (default). HAML employs ML-assisted methods to cluster data with similar hotness. RL-hybridSSD designs an RL-assisted space management scheme to coordinate GC and MC processes. LLM-hybridSSD (default) employs the same optimization strategy as LLM-hybridSSD but uses fixed default configurations during the entire workload execution.

LLM Models. For LLM model setup, we conduct GRPO-based fine-tuning on local deployed LLaMA 3.2, which is a widely adopted lightweight model. During the fine-tuning process, we choose the traces (i.e. mds_0, prxy_0, web_0 from MSR [18], mail from FIU [1], storage from Baleen [28]) as the training dataset. Additionally, we implement other LLM-driven prototypes using models such as GPT-o1, DeepSeek-R1, and Qwen3, and compare their performance and cost with that of our fine-tuned model (in § 5.4 and § 5.5).

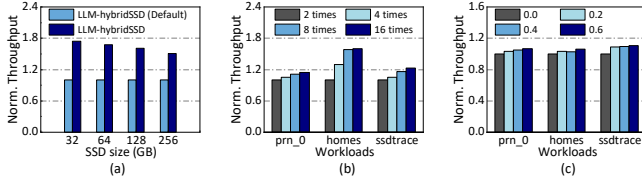


Figure 7: Sensitivity analysis by varying (a) SSD size, (b) tuning iterations, and (c) model temperature.

5.2 Overall Performance

We first evaluate the overall performance of the four implementations in terms of total throughput (normalized to LLM-hybridSSD (default)) and WA. The results are shown in Figure 6.

- **Throughput.** As shown, LLM-hybridSSD outperforms other strategies across the board. On average, the throughput improvements of LLM-hybridSSD over HAML, RL-hybridSSD, and LLM-hybridSSD (default) are 284.85%, 96.17%, and 58.92%, respectively. Notably, for the mixed trace with rapidly changing access patterns, the throughput improvement reaches up to 88.89%. This improvement are attributed to the adaptive parameters tuning capabilities of the LLM-based framework, which dynamically aligns configurations with hybrid SSD and trace characteristics.
- **WA.** LLM-hybridSSD achieves significant reductions in WA, on average 43.39%, 35.25% and 28.56% compared with HAML, RL-hybridSSD and LLM-hybridSSD (default), respectively. This improvement mainly stems from the LLM-guided adjustment of data placement and migration policies, which effectively reduces unnecessary cross-mode migration and mitigates frequent GC operations.

5.3 Sensitivity Analysis

- **Vary the SSD size:** We conduct experiments with SSD settings of different sizes using prn_0 as a sample trace and the results are shown in Figure 7 (a). As shown, the performance improvement slightly declines as SSD capacity increases. This is because larger SSDs accommodate more data, leading to fewer GC and mode conversion operations and thus less optimization potential.
- **Vary the tuning iterations:** We conduct experiments with different tuning iteration numbers and the results are shown in Figure 7 (b). As shown, increasing the number of iterations generally improves system performance, with the average throughput increasing by 32.67%. Most gains occur in the early iterations as key parameters are rapidly aligned with workload characteristics, while improvements beyond 8 iterations gradually level off.
- **Vary the temperature of model:** We evaluate the effect of temperature settings from 0.0 to 0.6, as shown in Figure 7(c). Lower temperatures produce more deterministic outputs, stabilizing configuration selection and increasing throughput. In contrast, higher temperatures improve exploration of alternative configurations, yielding an average performance gain of 7.74%.

5.4 Performance Impact of different LLMs

We further evaluate the impact of different LLM models on hybrid SSD optimization, including GPT-o1, DeepSeek-R1, Qwen3, LLaMA 3.2, and the fine-tuned LLaMA 3.2. Among them, GPT-o1 and DeepSeek-R1 are accessed through cloud services to explore scalability under remote inference, while Qwen3, LLaMA 3.2, and the fine-tuned LLaMA 3.2 are deployed locally to ensure lower cost and better data privacy. Figure 8 presents the performance comparison across these models.

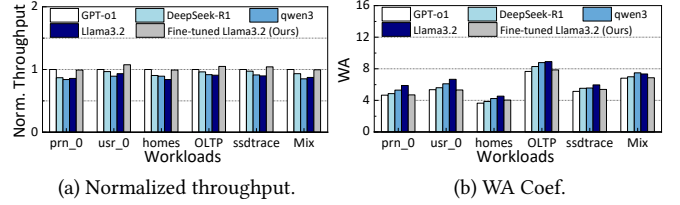


Figure 8: Performance impact of using different LLM models in LLM-hybridSSD.

Table 4: Overhead analysis of different LLMs.

Parameters	Cloud-based		Local deployed	
	GPT-o1 200B	Deepseek-R1 671B	qwen3 4B	Tuned LLaMA 3.2 3B
LLM inference Latency (s)	24.37	69.07	20.97 (CPU) 3.06 (GPU)	18.92 (CPU) 2.48 (GPU)
Cloud model credit Cost (\$)	3.09	0.51	/	/

As shown, the fine-tuned LLaMA 3.2 delivers the best or comparable performance among all tested models, showing both higher throughput and reduced WA across various workloads. Compared with large-scale cloud-based models (GPT-o1 and DeepSeek-R1), the fine-tuned LLaMA 3.2 used in our architecture achieves comparable or even superior optimization performance. Meanwhile, compared with LLaMA 3.2 and Qwen3, the fine-tuned model shows better adaptation to workload characteristics and more stable configuration tuning, resulting in an average throughput improvement of 16.85% and a WA decrease of up to 13.25%. This improvement primarily comes from the fine-tuned model ability to internalize hybrid SSD domain knowledge, enabling it to make more accurate configuration decisions under diverse workloads.

5.5 Overhead Analysis

Table 4 compares the inference latency and cost of different LLMs under prn_0 workload. Cloud-based models such as GPT-o1 and DeepSeek-R1 incur the highest latency (24.37s and 69.07s, respectively) due to remote inference and network communication. In contrast, local deployed lightweight models achieve much lower latency, with Qwen3 and the fine-tuned LLaMA 3.2 recording only 20.97s and 18.92s on CPU, and 3.06s and 2.48s on GPU, respectively. Moreover, cloud-based models introduce additional monetary costs per query, whereas local deployment eliminates credit-based charges and offers greater control. The fine-tuning and sensitive-parameter identification overhead together require about 8.6 hours, paid once offline and fully amortized over long-term deployment.

6 Conclusion

In this paper, we investigate the potential of LLMs to address the challenges associated with managing hybrid SSDs. By leveraging the context-aware reasoning capabilities of LLMs, we develop the LLM-hybridSSD framework, which incorporates a fine-tuned LLM-based auto-tuner to tune the parameters of hybrid SSD management. Our experimental results demonstrate that LLM-hybridSSD achieves significant performance improvements.

7 Acknowledgments

The work described in this paper is partially supported by the National Natural Science Foundation of China under Grant 624B2091, 62272271, U24B20149, the Shandong Provincial Natural Science Foundation under Grant ZR2024LZH004 and the Taishan Scholars Program under Grant tsqn202408009.

References

- [1] 2007. MSR Cambridge Traces (SNIA IOTTA Trace Set 388). In *SNIA IOTTA Trace Repository*. Storage Networking Industry Association. <http://iota.snia.org/traces/block-io?only=388>.
- [2] Mayur Akewar, Gang Quan, Sandeep Madireddy, and Janki Bhimani. 2025. Can LLMs Model the Environmental Impact on SSD?. In *Proceedings of the 17th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage)*. 100–106.
- [3] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, and Honghui Ding et al. 2024. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism. *CoRR* abs/2401.02954 (2024).
- [4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*.
- [5] Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks* 4, 2 (1991), 251–257.
- [6] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 97)*. 2790–2799.
- [7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations (ICLR)*.
- [8] Yang Hu, Hong Jiang, Dan Feng, Lei Tian, Hao Luo, and Chao Ren. 2013. Exploring and exploiting the multilevel parallelism inside SSDs for improved performance and endurance. *IEEE Transactions on Computers (TC)* 62, 6 (2013), 1141–1155.
- [9] Jian Huang, Anirudh Badam, Moinuddin K. Qureshi, and Karsten Schwan. 2015. Unified address translation for memory-mapped SSDs with FlashMap. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)*.
- [10] I. Inc. 2019. Intel ssd 665p series. [Online]. <https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/consumer-ssds/6-series/ssd-665p-series.html>.
- [11] M. Inc. 2018. Micron crucial p1 product. [Online]. <https://www.crucial.com/products/ssd/p1-ssd-series>.
- [12] Eui-Dong Jeong, Heegon Kim, Sukhyun Nam, Jae-Hyoung Yoo, and James Won-Ki Hong. 2024. S-Witch: Switch Configuration Assistant with LLM and Prompt Engineering. In *IEEE Network Operations and Management Symposium (GAIN)*.
- [13] Jaewook Kwak, Sangjin Lee, Kibin Park, Jinwoo Jeong, and Yong Ho Song. 2020. Cosmos+ OpenSSD: Rapid Prototype for Flash Storage Systems. *ACM Transactions on Storage (TOS)* 16, 3 (2020), 15:1–15:35.
- [14] Bingzhe Li, Chunhua Deng, Jinfeng Yang, David J. Lilja, Bo Yuan, and David H. C. Du. 2019. HAML-SSD: A Hardware Accelerated Hotness-Aware Machine Learning based SSD Management. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*.
- [15] Daixuan Li, Jinghan Sun, and Jian Huang. 2023. Learning to Drive Software-Defined Solid-State Drives. In *International Symposium on Microarchitecture (MICRO)*.
- [16] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL)*. 4582–4597.
- [17] Giulio Marotta, Luca De Santis, and Tommaso Vali. 2013. Dynamic SLC/MLC blocks allocations for non-volatile memory.
- [18] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. 2008. Write off-loading: Practical power management for enterprise storage. *ACM Transactions on Storage (TOS)* 4, 3 (2008), 10.
- [19] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023).
- [20] OpenAI. 2024. Learning to Reason with Reinforcement Learning. Available at: <https://openai.com/index/learning-to-reason-with-reinforcement-learning>.
- [21] Storage Performance Council SPC. 2010. SPC TRACE FILE FORMAT SPECIFICATION, Revision 1.0.1. [Online]. www.storageperformance.org.
- [22] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. Learning to summarize from human feedback. *CoRR* abs/2009.01325 (2020).
- [23] Radu Stoica, Roman Pletka, Nikolas Ioannou, Nikolaos Papandreou, Sasa Tomic, and Haris Pozidis. 2019. Understanding the design trade-offs of hybrid flash controllers. In *IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*.
- [24] Viraj Thakkar, Madhumitha Sukumar, Jiaxin Dai, Kaushiki Singh, and Zhichao Cao. 2024. Can Modern LLMs Tune and Configure LSM-based Key-Value Stores?. In *ACM Workshop on Hot Topics in Storage and File System (HotStorage)*.
- [25] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).
- [26] Qian Wei, Zehao Chen, Tianren Zhou, Wenbin Zhu, Zhengge Jia, Mengying Zhao, and Zhaoyan Shen. 2025. Exploiting Large Language Models for Software-Defined Solid-State Drives Design. In *Advanced Parallel Processing Technologies - 16th International Symposium (APPT)*.
- [27] Qian Wei, Yi Li, Zhiping Jia, Mengying Zhao, Zhaoyan Shen, and Bingzhe Li. 2023. Reinforcement Learning-Assisted Management for Convertible SSDs. In *ACM/IEEE Design Automation Conference (DAC)*.
- [28] Daniel Lin-Kit Wong, Hao Wu, Carson Molder, Sathya Gunasekar, Jimmy Lu, Snehal Khandkar, Abhinav Sharma, Daniel S. Berger, Nathan Beckmann, and Gregory R. Ganger. 2024. Baleen: ML Admission & Prefetching for Flash Caches. In *22nd USENIX Conference on File and Storage Technologies (FAST)*. 347–371.
- [29] Gala Yadgar, Moshe Gabel, Shehbaz Jaffer, and Bianca Schroeder. 2021. SSD-based Workload Characteristics and Their Performance Implications. *ACM Transactions on Storage (TOS)* 17, 1 (2021), 8:1–8:26.
- [30] Sangjin Yoo and Dongkun Shin. 2020. Reinforcement Learning-Based SLC Cache Technique for Enhancing SSD Write Performance. In *12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage)*.
- [31] Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. 2025. A Review on Edge Large Language Models: Design, Execution, and Applications. *Comput. Surveys* 57, 8 (2025), 209:1–209:35.
- [32] Wenbin Zhu, Zhaoyan Shen, Qian Wei, Renhai Chen, Xin Yao, Dongxiao Yu, and Zili Shao. 2025. HiDPU: A DPU-Oriented Hybrid Indexing Scheme for Disaggregated Storage Systems. In *USENIX Conference on File and Storage Technologies (FAST)*. 271–285.