LLM Fine-Tuning with aiDAPTIV+: A Viable Training Strategy on Resource-Constrained Compute Platforms

Abhishek Patel, Krishan Gopal Gupta, Shashank Sharma, Sowmya Shree, Sanjay Wandhekar
Centre for Development of Advanced Computing (C-DAC), Pune, India
Email: {abhishekp, krishang, shashank.sharma, ssowmya, sanjayw}@cdac.in

***Abstract*—The paper addresses the challenge of fine-tuning Large Language Models (LLMs) on resource-constrained hardware. The need for effective fine-tuning solutions has grown dramatically due to the quick development of open-source LLMs like the Large Language Model Meta AI (LLaMA). For researchers and practitioners with limited hardware, training these models is costly and difficult due to the significant computational resources needed. In order to enable and improve LLM fine-tuning performance, this study investigates an alternative method that makes use of MiPhi aiDAPTIV+ AI Solid-State Drives (SSDs). The impact of these SSDs is compared to traditional dense Graphics Processing Unit (GPU)-accelerated training setups that exclusively rely on the memory and compute power of high-end GPUs. The primary objective is to investigate whether AI SSDs can help older High Performance Computing (HPC) clusters and workstations overcome memory and processing constraints. The methodology is equally applicable to public cloud environments (e.g., AWS, Azure, GCP), where GPU instances are often constrained by cost, memory and processing constraints, making it easier to run contemporary LLM workloads even with constrained GPU resources. Llama 2 and Llama 3 models were fine-tuned with a precision of Brain floating point 16 (BF16) using extensive experiments on the Center for Development of Advanced Computing's (CDAC) PARAM Rudra HPC platform and PARAM Siddhi AI platform, both of which had NVIDIA A100 40GB GPUs. Using the DeepSpeed framework, the effects of MiPhi aiDAPTIV+ AI SSDs and their middleware are evaluated against a dense GPU system with an emphasis on increases in data transfer speeds, model checkpointing and overall system utilization. The study shows that the feasible configuration space for LLM fine-tuning on modest hardware is greatly expanded by AI SSD-accelerated training, allowing models that would otherwise fail due to memory exhaustion to run successfully on minimal GPU configurations.**

Keywords-LLM; fine-tuning; MiPhi aiDAPTIV+; HPC; Token Throughput Optimization.

I. INTRODUCTION

Natural Language Processing (NLP) has undergone a significant transformation with the emergence of LLMs [**2**], enabling advanced applications across domains, such as software engineering, healthcare, finance, customer service, and scientific research [**1**]. Recent open-source models, including Meta's LLaMA series and the DeepSeek family, have demonstrated strong capabilities in few-shot learning, reasoning, and coherent text generation. These capabilities are largely driven by large-scale pre-training on diverse corpora followed by fine-tuning on domain-specific datasets. As organizations increasingly integrate LLMs into critical workflows, the demand for cost-effective and resource-efficient training methodologies has grown substantially.

Despite these advancements, modern LLM training and fine-tuning introduce several system-level challenges [**3**]:

- **High GPU Memory Requirements:** Even relatively smaller LLM variants with billions of parameters require GPUs with large High Bandwidth Memory (HBM), making them incompatible with older-generation hardware.
- **Infrastructure Bottlenecks:** The cost and complexity of deploying next-generation GPU infrastructure pose significant barriers for research institutions, government laboratories, and startups.
- **Limitations of Legacy HPC Systems:** Traditional High Performance Computing (HPC) clusters, primarily optimized for simulation workloads, often struggle with the memory bandwidth and I/O patterns required for LLM training.
- **Scaling Challenges:** Increasing batch sizes and model complexity impose additional pressure on system architecture, requiring efficient data access, parameter offloading, and high-bandwidth memory management for optimizer states.
- **Framework Constraints:** While frameworks such as DeepSpeed, Hugging Face Accelerate, and parameter-efficient techniques like Low-Rank Adaptation (LoRA) improve computational efficiency, their performance remains constrained by storage transfer rates and I/O bottlenecks.
- **Need for Alternative Architectures:** These limitations highlight the necessity of exploring alternative storage and memory hierarchies, such as SSD-accelerated workflows and burst buffer integration, to mitigate system bottlenecks.

To address these challenges, this paper investigates MiPhi AI-accelerated SSDs as an alternative to conventional GPU-centric training architectures. Unlike standard NVMe SSDs, MiPhi AI SSDs, integrated with the aiDAPTIV+ middleware, employ a co-designed hardware-software approach optimized for transformer-based workloads. The system dynamically partitions model states and offloads less frequently accessed data from GPU memory to high-speed SSD storage, effectively expanding usable memory capacity. This enables efficient data prefetching, real-time checkpointing, and high-throughput training without requiring modifications to existing training pipelines.

The performance of MiPhi aiDAPTIV+ SSDs [**4**] is evaluated during fine-tuning of LLaMA 2 and LLaMA 3 models

using BF16 precision. The study focuses on analyzing improvements in I/O throughput and training efficiency across both single-GPU and multi-GPU environments. The key contributions of this work are summarized as follows:

- **Empirical Evaluation:** A comprehensive benchmark assessing the impact of AI-accelerated SSDs on data loading, checkpointing, and overall system throughput during LLM fine-tuning.
- **Single-GPU Optimization:** An analysis of performance gains achievable for researchers operating under limited hardware constraints.
- **Cost-Performance Analysis:** A comparative study between GPU-only and AI SSD-accelerated setups to guide cost-efficient infrastructure decisions for LLM training.
- **Configuration Space Characterization:** A systematic mapping of feasible LLM fine-tuning configurations (model size, batch size, sequence length, GPU count) enabled by AI SSD offloading on production HPC nodes, providing practitioners with a reproducible methodology applicable to both HPC and public cloud environments.

The remainder of this paper is organized as follows. Section 2 reviews recent advancements in LLM training methodologies, including parameter-efficient fine-tuning, offloading strategies, distributed training, and quantization techniques. Section 3 presents the MiPhi aiDAPTIV+ architecture, detailing its hardware design and middleware components. Section 4 describes the experimental setup, including system configurations, datasets, and workload parameters. Section 5 outlines the execution methodology. Sections 6 and 7 present performance results and throughput benchmarks on PARAM Siddhi [**5**] and PARAM Rudra systems [**6**]. Section 8 discusses key observations and technical insights, and Section 9 concludes the paper with directions for future work.

II. Training Requirements of LLMs: Current State of the Art

Training and fine-tuning of LLMs have become essential for adapting general-purpose foundation models to domain-specific applications. While models such as LLaMA, GPT, and DeepSeek are pre-trained on large and diverse datasets, further fine-tuning is required to incorporate specialized knowledge, enforce organizational constraints, and align with user-specific requirements. However, this process introduces significant computational and memory challenges.

To address these limitations, several optimization techniques have been developed:

- **Parameter-Efficient Fine-Tuning (PEFT):** PEFT techniques, including Low-Rank Adaptation (LoRA), Quantized LoRA (QLoRA), and Adapter-based methods, enable fine-tuning by introducing a small number of trainable parameters into pre-trained transformer layers [**7**][**8**][**9**]. By keeping the majority of model weights frozen, these methods significantly reduce memory and computational requirements, allowing LLM fine-tuning on commodity GPUs with limited memory capacity.
- **Offloading and CPU-GPU Hybridization:** To further reduce GPU memory pressure, techniques, such as ZeRO-Offload in DeepSpeed and hybrid execution pipelines in Hugging Face Accelerate move optimizer states and intermediate activations to CPU memory or NVMe storage [**10**]. Frameworks like Colossal-AI [**14**] extend this concept by enabling hierarchical memory management across CPU and GPU tiers.
- **Sharded and Distributed Training:** Large-scale training is facilitated through distributed strategies such as Fully Sharded Data Parallel (FSDP) and Megatron-DeepSpeed, which partition model parameters, gradients, and optimizer states across multiple devices [**11**]. These approaches reduce per-device memory requirements and enable multi-node scalability, but introduce communication overhead and increased system complexity.
- **Mixed Precision and Quantization:** The use of reduced precision formats such as BF16 and INT8 significantly lowers memory consumption and improves computational throughput. Techniques such as Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ) enable efficient deployment of large models with minimal accuracy degradation [**12**][**13**]. Notably, QLoRA employs 4-bit quantization to achieve substantial memory savings, enabling fine-tuning of models with billions of parameters on limited hardware resources.

Despite these advancements, training and fine-tuning of LLMs continue to face substantial computational challenges. These include managing parameter updates, maintaining optimizer states, and performing checkpointing within constrained GPU memory environments.

For instance, full-precision fine-tuning of a 7B parameter model such as LLaMA 2 using the Adam optimizer can require over 100 GB of GPU memory due to the storage of model parameters, gradients, and optimizer states, which together can consume approximately 16 bytes per parameter [**16**]. Additionally, large-scale training pipelines demand high-throughput storage systems, often requiring terabytes of capacity for data preprocessing, tokenization, checkpointing, and evaluation. Consequently, memory bandwidth and I/O throughput remain critical bottlenecks in efficient LLM training.

III. MiPhi aiDAPTIV+ Architecture

MiPhi's aiDAPTIV+ is a combined hardware and software system created to address memory issues during the LLM's training. The sections that follow explain both the hardware architecture and the middleware stack.

A. aiDAPTIV+ Hardware Architecture and Integration

The main hardware element features an AI100E SSD, offered in both U.2 and M.2 formats, that functions as an aiDAPTIVCache to expand GPU memory. It supports up to 320 GB for personal computers and 8 TB for workstations or servers and provides up to 100 Drive Writes Per Day (DWPD) using Single Level Cell (SLC) NAND along with advanced NAND correction algorithms, allowing for the operation of

transformer-based models such as LLaMA 3 70B and Falcon 180B on a single server.

B. aiDAPTIV+ Middleware and Software Stack

The aiDAPTIV+ software architecture is intended to allow seamless integration into current AI workflows, especially those developed using the PyTorch framework. The middleware layer aiDAPTIVLink functions as a virtual memory controller, enabling smooth interaction between the AI application and the underlying aiDAPTIVCache hardware. The aiDAPTIV+ middleware library extends available GPU memory through dynamic memory virtualization using SSDs, enabling the offloading of tensors and model parameters. This enables the training and inference of large transformer-based models like LLaMA and Mistral within current PyTorch-based workflows, without the need to alter the training process.

The architecture is structured to support high-speed, low-latency I/O operations and precise management of data placement. The platform can be accessed through both a Command-Line Interface (CLI) and a graphical user interface (aiDAPTIVPro Suite) and it supports workflows such as data ingestion and fine-tuning.

IV. EXPERIMENTAL SETUP

This section describes the system configuration, datasets, and experimental parameters used to evaluate MiPhi aiDAPTIV+ performance.

A. Compute Infrastructure Overview

To evaluate the impact of MiPhi AI SSD acceleration on LLM fine-tuning, experiments were conducted on single node of two high-performance computing (HPC) platforms: **PARAM Siddhi AI** and **PARAM Rudra**. Their summarized node-level specifications are shown in Table I.

These platforms played complementary roles: **PARAM Siddhi AI (TestBed 1)** offered a dense multi-GPU environment, while **PARAM Rudra (TestBed 2)** represented a typical 2-GPU server setup.

1) System Software Stack Overview: Table II summarizes the software environment used on the PARAM Rudra and PARAM Siddhi AI clusters. These software configurations are crucial for ensuring compatibility and performance during large-scale training using MiPhi AI SSD middleware.

TABLE I. SYSTEM SPECIFICATIONS OF PARAM SIDDHI AI (TESTBED 1) AND PARAM RUDRA (TESTBED 2).

Specification	**PARAM Siddhi AI (TestBed 1)**	**PARAM Rudra (TestBed 2)**
CPU	Dual AMD EPYC 7742	Dual Intel Xeon Gold 6240R
Cores/Threads per Node	128 cores, 256 threads	48 physical cores
Clock Speed	2.25 GHz	2.40 GHz
L3 Cache	256 MB	36 MB
System Memory (RAM)	1 TB per node	192 GB per node
GPU	8× NVIDIA A100 (SXM4, 40 GB each)	2× NVIDIA A100 (PCIe Gen3, 40 GB each)
GPU Memory per Node	320 GB	80 GB
MiPhi AI SSD	4 TB	2 TB
Networking	Mellanox ConnectX-6 VPI (InfiniBand HDR), 1.6 Tbps	InfiniBand HDR fabric
Shared Storage	10.5 PiB Lustre parallel file system	1 PiB Lustre parallel file system

TABLE II. SOFTWARE STACK ON PARAM RUDRA AND PARAM SIDDHI AI.

Component	**PARAM Rudra**	**PARAM Siddhi AI**
OS	Ubuntu 22.04.5 LTS	Ubuntu 20.04.2 LTS
Kernel Version	5.15.0-142-generic	5.4.0-80-generic
Architecture	x86_64	x86_64
NVIDIA Driver	550.163.01	450.142.00

B. Workload Configurations

The study focused on fine-tuning various Llama 2 model sizes 7B, 13B, and 70B across different system configurations. The dataset Cohere/miracl-zh-queries-22-12 [**17**] was used for this purpose. The dataset is designed for multilingual retrieval tasks and features natural language queries that closely mimic real-world usage, especially in information retrieval and question-answering domains.

The dataset is loaded using a 100% training split (no validation or test subsets) and models are trained with varying batch sizes and sequence lengths to evaluate system throughput and fine-tuning efficiency. The training parameters used include:

- Batch Sizes: {8, 16, 32, 64, 128}
- Sequence Lengths: {128, 256, 512, 1024, 2048}
- Model Sizes: {7B, 13B, 70B}
- Nodes: {1, 2, 4, 8} across PARAM Rudra and PARAM Siddhi
- Precision: BF16

The choice of batch sizes and sequence lengths in powers of two minimizes memory fragmentation and improves GPU utilization, which directly impacts training throughput and system stability. Similarly, exploring all permutations of model size, batch size and sequence length allows a thorough evaluation of training scalability and system behavior under diverse conditions. This comprehensive matrix helps identify hardware bottlenecks, such as memory exhaustion or

data transfer latencies, particularly relevant when comparing traditional storage versus AI SSD acceleration.

V. Execution Workflow

The experimental process is intended to assess how the performance of LLM fine-tuning changes when using single-node, multi-GPU configurations with MiPhi aiDAPTIV+ technology compared to a standard DeepSpeed setup. The process takes place in the following manner:

- **Data Preparation and Staging:** Pre-tokenized datasets are loaded into the MiPhi AI SSD at the start, minimizing host-to-device I/O latency during subsequent epochs.
- **Model Initialization:** Llama variants (7B, 13B, 70B) are instantiated in GPU memory using BF16 precision, ensuring sufficient headroom for batch-based workloads.
- **Activation Offloading:** The aiDAPTIV+ middleware offloads and prefetches activations to and from the SSD concurrently with ongoing GPU computation, allowing I/O transfers to overlap with arithmetic operations.
- **Checkpointing and Streaming:** Checkpoints are periodically written to the SSD with minimal I/O stalling, enabled by high-throughput, low-latency access in conjunction with intelligent storage scheduling.
- **Performance Monitoring:** Throughout training, key performance metrics such as tokens per second (throughput), micro-batch latency, and loss convergence are logged to quantitatively evaluate the efficacy of the aiDAPTIV+ stack.

By offloading memory pressure from GPU DRAM to SSD and overlapping data movement with computation, this workflow maintains high utilization even when training large-context LLMs (7B-70B) a challenge often highlighted in recent SSD offloading research [**15**]. Key training parameters such as batch size and sequence length are varied, keeping precision as BF16, to test the boundaries of model size that can be trained given the number of GPUs.

VI. Results and Observations

This section presents the experimental results from both PARAM Siddhi and PARAM Rudra platforms, evaluating the impact of MiPhi aiDAPTIV+ SSD on GPU memory efficiency, model scalability, and throughput.

A. PARAM Siddhi: With & Without MiPhi aiDAPTIV+ SSD

The primary objective was to evaluate the impact of MiPhi AI SSD on GPU memory efficiency, model scalability, and throughput improvement across varying model sizes and configurations.

1) PARAM Siddhi: Without MiPhi aiDAPTIV+ SSD: A set of experiments was carried out on the PARAM Siddhi HPC system a high-density GPU system to assess the training potential of LLaMA 2 models of different sizes (7B, 13B, and 70B), both with and without MiPhi aiDAPTIV+ SSD optimization. In baseline configuration, without MiPhi, we employed BF16 precision and DeepSpeed ZeRO-3 optimization to enable memory-efficient fine-tuning. These experiments allowed us to identify the minimum GPU requirements and limitations when MiPhi acceleration was not used, which served as a comparative baseline MiPhi's impact.The configurations were:

TABLE III. Minimum GPU Requirements Without MiPhi aiDAPTIV+ SSD for LLaMA 2 Models.

Model	**Batch Size**	**Seq Length**	**Minimum GPUs Required W/o MiPhi AI SSd**
LLaMA 2 – 7B	4	2048	4 GPUs
LLaMA 2 – 13B	4	512	7 GPUs
LLaMA 2 – 70B	1	–	> Unable to fit even with 8 GPUs

Table III summarizes the minimum GPU requirements, and Figure 1 visualizes the corresponding training efficiency across GPU counts.

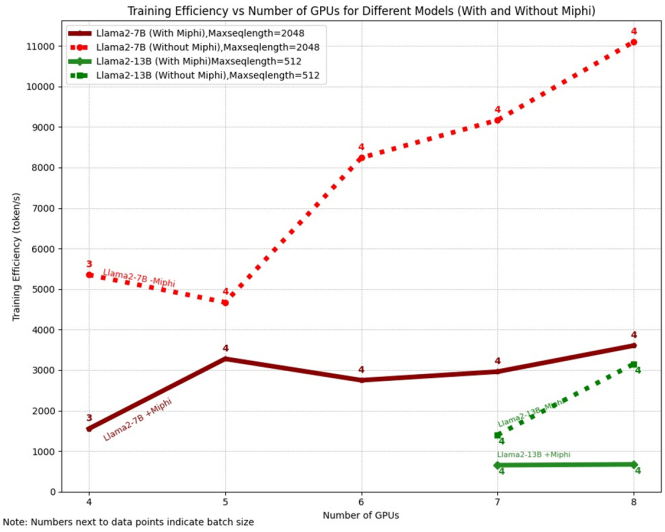

Figure 1. Training efficiency vs. number of GPUs for different Llama-2 models with and without MiPhi aiDAPTIV+ SSD. Numbers indicate batch sizes.

Figure 1 illustrates the training efficiency across varying GPU counts for LLaMA 2 models on PARAM Siddhi without MiPhi aiDAPTIV+ SSD. Key findings:

- **LLaMA 2 – 7B:** Scaled well from 4 to 8 GPUs, reaching over 11, 000 tokens/sec.
- **LLaMA 2 – 13B:** Required at least 7 GPUs and showed diminishing returns.
- **LLaMA 2 – 70B:** Failed to fit on 8 GPUs, confirming its infeasibility without memory offloading.
- **LLaMA 2 – 7B:** Achieved stable scaling from 4 to 8 GPUs at sequence length 2048, increasing throughput

from ∼5000 to over 11, 000 tokens/sec. Minimum 4 GPUs were required to fit the model.

- **LLaMA 2 – 13B:** Required at least 7–8 GPUs even at sequence length 512. Throughput remained under 3000 tokens/sec, showing diminishing returns with increasing GPU count.
- **LLaMA 2 – 70B:** Failed to fit on 8 GPUs even with minimum batch size and sequence length, indicating infeasibility without memory offloading.

These observations confirm that while smaller models scale efficiently without MiPhi aiDAPTIV+ SSD, larger models (13B and above) quickly encounter memory bottlenecks emphasizing the necessity of AI SSD-assisted training infrastructure for modern LLM workloads.

2) PARAM Siddhi: Training Efficiency and Scaling Performance With MiPhi aiDAPTIV+ SSD: To assess the scalability and efficiency of MiPhi-accelerated LLM training, a controlled set of experiments was conducted by fixing the sequence length to 2048 tokens and varying the batch size and number of GPUs. These experiments were conducted using LLaMA 2 models (7B and 13B) on PARAM Siddhi with MiPhi aiDAPTIV+ integration. Training efficiency is measured in tokens per second and does not indicate model accuracy or loss. The results highlight:

- Higher batch size improves throughput but not necessarily convergence or optimal training.
- Model performance remains reliant on proper hyperparameter tuning.

Figure 2 presents the training throughput for LLaMA 2 models using MiPhi aiDAPTIV+ acceleration across different GPU counts.

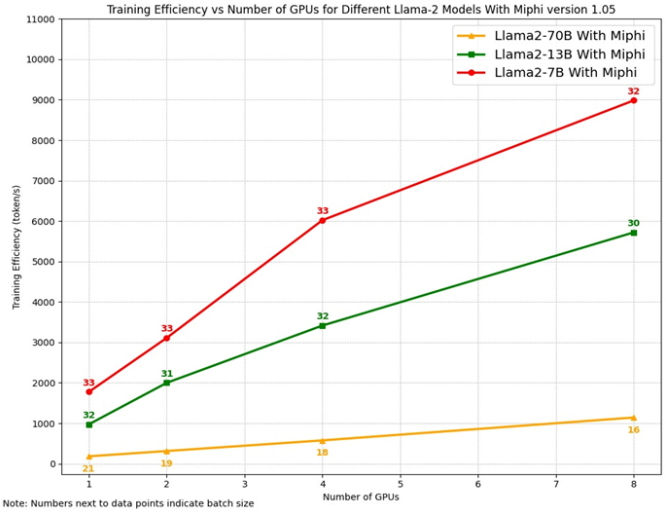

Figure 2. Training efficiency vs. number of GPUs using MiPhi aiDAPTIV+ v1.05 with maximum batch size. Linear scaling observed up to 8 GPUs.

Key insights:

- **LLaMA 2 – 7B:** Achieved linear scaling up to 8 GPUs with training efficiency exceeding 9000 tokens/sec. Batch size remained consistent (∼32–33), indicating effective memory offloading via MiPhi aiDAPTIV+.
- **LLaMA 2 – 13B:** Demonstrated steady scaling from 1 to 8 GPUs, reaching 5000+ tokens/sec. MiPhi aiDAPTIV+ enabled consistent batch sizes (∼30–32), which are infeasible on traditional setups without memory augmentation.
- **LLaMA 2 – 70B:** Successfully trained on as few as 1–2 GPUs with MiPhi aiDAPTIV+ (batch sizes 19-21), scaling up to 1000+ tokens/sec on 8 GPUs. Without MiPhi aiDAPTIV+ SSD, this model could not be trained on a single node due to memory limitations.

Overall, MiPhi aiDAPTIV+ SSD improved scalability and training feasibility across model sizes by enabling larger batch sizes, stable throughput, and efficient memory utilization, even for massive models like LLaMA 2-70B.

B. PARAM Rudra: With MiPhi aiDAPTIV+ SSD

The evaluation was further extended to the PARAM Rudra platform a smaller GPU system which features a single node equipped with 2×NVIDIA A100 GPUs. In this setup, a single 2 TB MiPhi aiDAPTIV+ AI100E SSD (U.2 form factor) was integrated and experiments were conducted to assess the feasibility and efficiency of LLM fine-tuning using both 1-GPU and 2-GPU configurations.

Training was performed using the MiPhi aiDAPTIV+ stack with middleware versions 1.5 and 2.0. Version 1.5 was tested for scaling behavior across varying batch sizes for LLaMA 2 models, while version 2.0, which provides support for newer foundation models, enabled successful fine-tuning of the **LLaMA 3.1–8B** model.

1) Training Requirements and Observations: **Hardware Configuration:** All experiments were conducted on the PARAM Rudra Server Board using the aiDAPTIV+ v2.0 Kit. Training was performed using Llama 2 and Llama 3.1 variants with the following configuration:

- **Dataset:** `Cohere/miracl-zh-queries-22-12`
- **Batch Sizes Tested:** 8, 16, 24, 32
- **Maximum Sequence Lengths:** 128, 256, 512, 1024, 2048

Figure 3 and Figure 4 show token throughput for LLaMA 2-7B and LLaMA 3.1-8B respectively on a single GPU on PARAM Rudra, across varying batch sizes and sequence lengths.

The performance graphs demonstrate token throughput for LLaMA 2-7B and LLaMA 3.1-8B models using a single NVIDIA A100 GPU on the PARAM Rudra platform. For LLaMA 2-7B, throughput scales consistently with both increasing sequence length and batch size, achieving over 1100 tokens/sec at the maximum configuration (batch size 32, sequence length 2048). This indicates effective utilization of GPU resources for smaller models.

In contrast, LLaMA 3.1-8B shows more constrained scaling behavior. The model faces memory limitations beyond sequence lengths of 512 for batch sizes larger than 16, preventing data collection for higher configurations. Nonetheless, for supported configurations, performance trends upward, reaching close to 500 tokens/sec for batch size 8 and sequence length 2048.

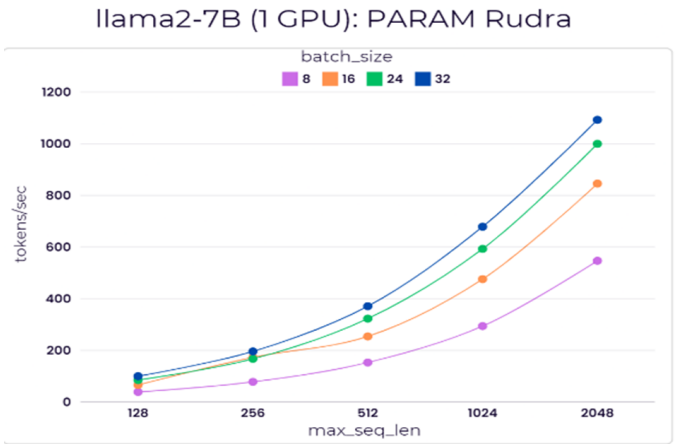

Figure 3. Llama 2 (7B), 1 GPU on PARAM Rudra: Scaling efficiency increased with increase in batch size and sequence length

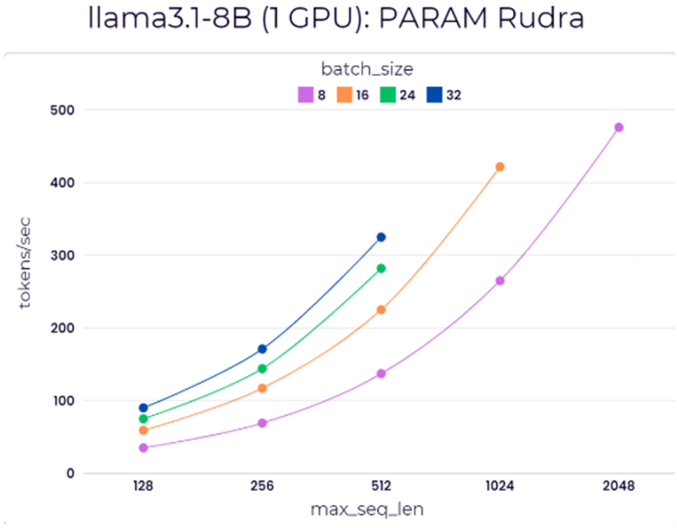

Figure 4. Llama 3.1 (8B), 1 GPUs with MiPhi aiDAPTIV+ on PARAM Rudra. Experiments failed on large batch size and sequence length combinations.

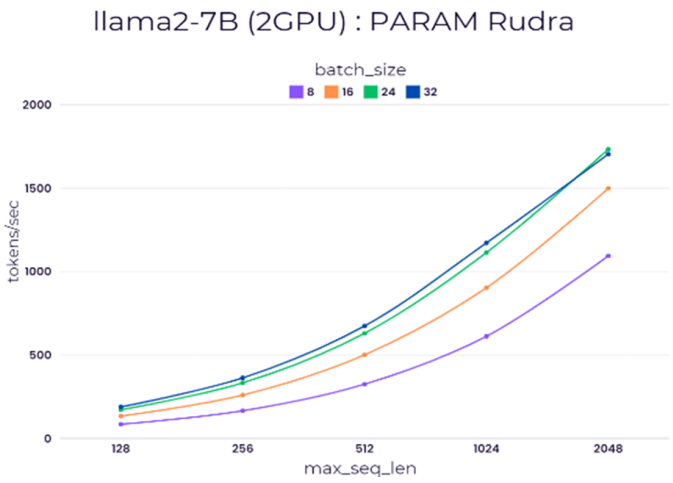

Figure 5. Llama 2 (7B), 2 GPUs with MiPhi aiDAPTIV+ on PARAM Rudra.

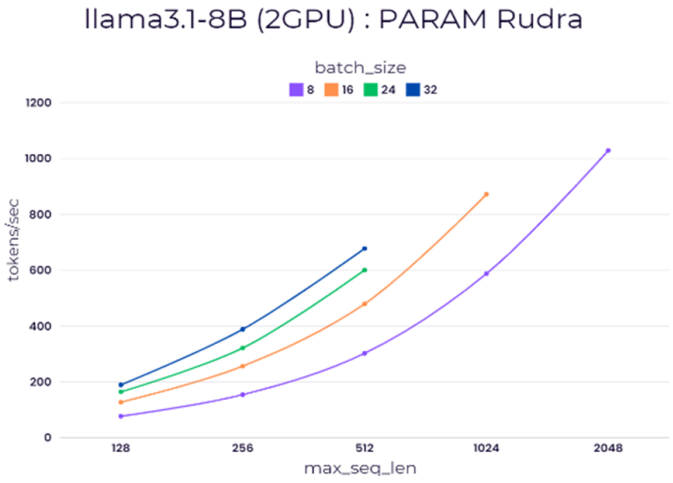

Figure 6. Llama 3.1 (8B), 2 GPUs with MiPhi aiDAPTIV+ on PARAM Rudra. Experiments failed on large batch size and sequence length combinations.

To determine the minimum GPU requirements for training the LLaMA 3.1-8B model with a larger maximum sequence length, experiments were conducted using a 2-GPU configuration for further evaluation. As shown in Figure 5 and Figure 6, experiments conducted using two GPUs revealed similar behavior to the single-GPU configuration, with no significant changes observed in performance when increasing the number of GPUs.

While scaling in terms of **training efficiency** (tokens/sec) was observed as the number of GPUs increased, several configurations failed due to **CUDA Out of Memory (OOM)** errors. **Failed Experiment Configurations for LLaMA 3.1-8B(2GPUs):**

- **Max Sequence Length = 1024:** Batch sizes of 24 and 32
- **Max Sequence Length = 2048:** Batch sizes of 16, 24, and 32

These observations suggest that, even with multiple GPUs, memory constraints remain a key bottleneck. This highlights the need for further investigation into the extent to which large models can be accommodated using **NVMe-based memory extension** solutions such as MiPhi's AI SSD.

C. Comparative Analysis of MiPhi aiDAPTIV+ v2.0 on PARAM Siddhi and PARAM Rudra

To evaluate the performance of **MiPhi aiDAPTIV+ v2.0**, benchmarking was carried out on two supercomputing infrastructures: **PARAM Siddhi** and **PARAM Rudra**. The experiments used a maximum sequence length of 2048 tokens across various GPU and batch size configurations.

The key hardware specifications of the two systems are summarized in Table IV. This highlights differences in GPU interconnect, NVMe configuration, aiDAPTIV kit capacity, and system RAM, which directly influence the observed throughput variations.

Following the hardware overview, the throughput in tokens per second (`tokens/sec`) was recorded for varying GPU

TABLE IV. HARDWARE DIFFERENCES: PARAM SIDDHI VS. RUDRA.

Component	Siddhi	Rudra
GPU Interface	SXM4 (600 GB/s)	PCIe Gen3 (8 GB/s)
NVMe Config	RAID 0	Non-RAID
aiDAPTIV kits	2 × 2 TB	1 × 2 TB
System RAM	1 TB	192 GB

and batch size configurations, as shown in Table V and visualized in Figure 7.

D. Results: Training Throughput on PARAM Rudra

Benchmarking experiments were performed on the LLaMA-2 7B and LLaMA-3.1 8B models using PARAM Rudra with both single-GPU and dual-GPU configurations. The experiments involved testing different combinations of batch sizes and sequence lengths to evaluate the resulting throughput and to analyze the impact of increasing maximum sequence length at various batch sizes. Additionally, these tests allowed identification of the practical limitations of fine-tuning larger models, specifically determining the batch size and sequence length at which training fails due to CUDA OOM errors. A summary of these benchmarking results is presented in Table VI.

TABLE V. TOKENS/SEC FOR DIFFERENT GPU AND BATCH SIZE CONFIGURATIONS ON SIDDHI AND RUDRA (MAX_SEQ_LEN = 2048)

GPUs (Batch Size)	Siddhi (tokens/sec)	Rudra (tokens/sec)
1 (18)	1803	914
1 (33)	1776	1180
2 (18)	3806	1451
2 (33)	3235	1739

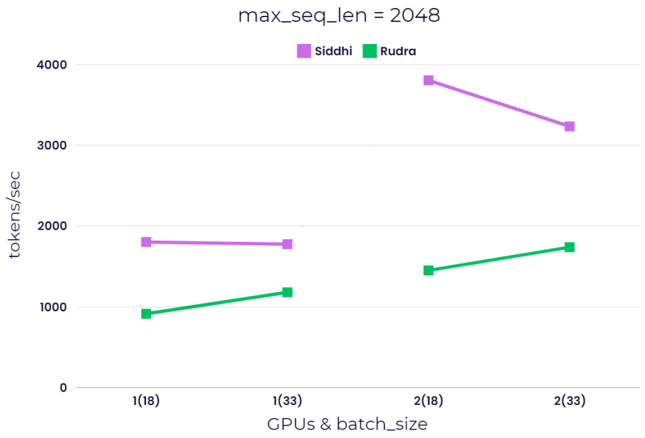

Figure 7. Performance comparison of MiPhi aiDAPTIV+ v2.0 on PARAM Siddhi and Rudra with varying GPU and batch size configurations (max_seq_len = 2048).

Using **MiPhi aiDAPTIV+ v2.0**, the fine-tuning performance of the LLaMA-3.1 70B model on **PARAM Rudra**, was evaluated, aiming to determine the maximum batch size achievable and how throughput scales with increasing batch size. In Figure 8, the token efficiency (tokens/sec) was empirically evaluated for fine-tuning the LLaMA-3.1 70B model on a single PARAM Rudra node using one and two NVIDIA A100 (40GB) GPUs, paired with a 2 TB MiPhi AI DAPTIV SSD. The evaluation spanned batch sizes from 1 to 80, with a fixed sequence length of 256 tokens.

- With a single GPU, the peak token efficiency achieved was **52.3 tokens/sec** at batch size 80.
- With both GPUs utilized, efficiency rose to a maximum of **86.9 tokens/sec**, marking a **65.9% improvement** over the single-GPU case.
- At 52.3 tokens/sec, training on a dataset of 10 billion tokens would require approximately **2.22 days** on a single GPU (assuming uninterrupted operation).
- With 2 GPUs and peak efficiency of 86.9 tokens/sec, the same workload would complete in **1.33 days**.
- This aligns with token-based training time estimates reported by Meta for Llama training runs, which cite token throughput as a core determinant of training duration and hardware planning.
- In academic labs or small HPC centers, limited compute resources slow down model training and hinder rapid experimentation.
- Multi-GPU scaling boosts throughput, but the costs of hardware, energy, and maintenance often outweigh the gains unless paired with efficiency methods such as mixed-precision training, parameter-efficient tuning (e.g., LoRA), or quantization.
- Overall, these experiments contribute to the discourse on **democratizing LLM training and inference**, underscoring the relevance of I/O-aware system design and cost-effective hardware strategies for practical deployment across constrained computing environments.

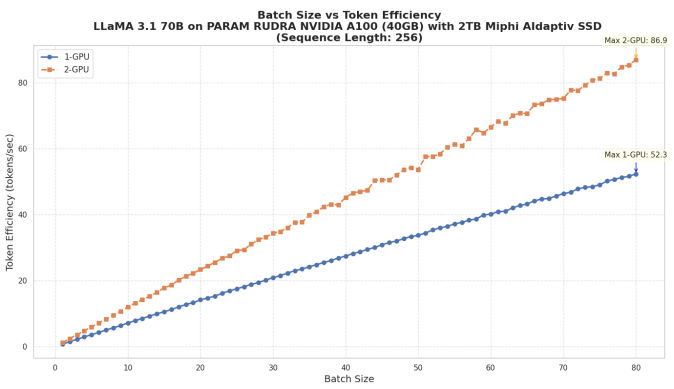

Figure 8. Batch Size vs Token Efficiency for Llama 3.1 70B on PARAM Rudra (1-GPU vs 2-GPU) with 2TB MiPhi AI SSD, Sequence Length: 256

VII. OBSERVATIONS

These observations summarize how the MiPhi aiDAPTIV AI SSD influences fine-tuning behavior across heterogeneous compute nodes, focusing on practical configuration choices, stability boundaries and operational checklists. The intent is not to rank systems, but to highlight how interconnects,

TABLE VI. TOKENS/SEC FOR LLAMA 2-7B AND LLAMA 3.1-8B ON RUDRA ACROSS GPU AND BATCH SIZE CONFIGURATIONS.

Max Seq.	Batch Size	Llama 2-7B		Llama 3.1-8B	
		1 GPU	2 GPU	1 GPU	2 GPU
128	8	39	84	35	77
	16	66	133	59	127
	24	85	171	75	164
	32	100	188	85	189
256	8	78	165	69	154
	16	173	259	117	256
	24	167	332	144	321
	32	196	362	171	388
512	8	153	324	137	302
	16	254	500	225	479
	24	323	629	282	600
	32	371	673	325	677
1024	8	294	611	265	587
	16	476	902	422	871
	24	593	1113	OOM	OOM
	32	679	1171	OOM	OOM
2048	8	547	1093	488	1028
	16	846	1498	OOM	OOM
	24	1000	1732	OOM	OOM
	32	1093	1703	OOM	OOM

memory capacity and storage paths shape the effective use of MiPhi, so practitioners can select batch and sequence settings, scaling strategies and checkpoint policies that deliver predictable throughput and robust runs across both dense multi-GPU and modest GPU environments.

- **Role of system topology:** Inter-GPU connectivity, host memory capacity and storage throughput primarily determine how far MiPhi aiDAPTIV can raise feasible batch size and sequence length before encountering OOM or stalls. Dense interconnects help scaling efficiency, while smaller memory footprints benefit from careful batch and sequence tuning with MiPhi's offload and prefetch mechanisms.
- **MiPhi's core value - feasibility envelope expansion:** MiPhi aiDAPTIV consistently extends the workable configuration space by offloading activations and parameters and overlapping I/O, enabling larger batches and longer contexts to fit on both dense and modest GPU nodes. Effectiveness is seen in enabling configurations that otherwise fail under baseline GPU-only fine-tuning.
- **Throughput vs. stability trade-off:** Increasing the batch size generally improves tokens/sec until memory or backend limits are reached. With MiPhi, the knee point shifts higher, but operators should validate convergence behavior since throughput gains do not substitute for hyperparameter tuning or task-specific objectives.
- **Sequence length is the dominant memory driver:** OOM errors correlate strongly with long contexts (e.g., 1024 to 2048), especially on larger models. Prioritizing sequence length control and using MiPhi to support moderate-to-high batches at shorter-to-moderate contexts yields stable, high-utilization runs on both system types.
- **Multi-GPU scaling considerations:** Scaling efficiency depends on interconnect and synchronization overheads. MiPhi helps by smoothing I/O and memory pressure, but communication topology still governs how close scaling approaches linearly. All-reduce timings and overlap should be monitored to tune GPU counts effectively in each environment.
- **Storage staging and checkpoint strategy:** Pre-staging tokenized data on the AI SSD and streaming checkpoints at appropriate cadence reduces stalls. Checkpoint frequency should be chosen to balance fault tolerance with I/O pressure, leveraging MiPhi's higher-throughput paths to keep GPU pipelines busy on both small and large nodes.
- **Safe operating regions per model size:** 7B models reliably sustain higher batch sizes across broader sequence ranges, whereas 13B and above benefit more from MiPhi to avoid OOM, but require conservative sequence lengths or graduated batch ramps — identifying per-model green zones and using MiPhi to widen them iteratively is recommended.
- **Practical scaling is sub-linear by design:** Doubling GPUs or batch size rarely doubles tokens/sec due to attention complexity, optimizer and dataloader overheads, interconnect characteristics and I/O contention. MiPhi should be used to mitigate memory and I/O bottlenecks while accepting intrinsic scaling limits, then optimizing overlap and micro-batching.
- **Validation on two contrasting nodes:** On a dense node, MiPhi's overlap and offload help sustain long-context training and higher aggregate throughput under multi-GPU scaling. On a lower-GPU node, MiPhi enables fitting larger configurations and improves single and dual-GPU usability, with attention to OOM thresholds at long contexts and larger batches.
- **Monitoring and diagnostics:** Tokens/sec, micro-batch latency, CUDA OOM events, offload bandwidth, checkpoint time and all-reduce and communications breakdowns should be tracked. Sequence length should be adjusted first for stability, then batch size for throughput and MiPhi profiling should be used to ensure offload and prefetch overlap is effective under the given interconnect and storage stack.
- **Model-choice guidance:** When time-to-train and stability are priorities under constrained GPUs, smaller models or shorter contexts with MiPhi-enabled larger batches are preferable. For larger models, MiPhi aiDAPTIV should be combined with parameter-efficient methods and conservative sequence lengths, then GPUs scaled as interconnect and memory allow.
- **Deployment guidance across heterogeneous clusters:** Topology-aware scheduling that matches workload char-

acteristics to node traits should be adopted. MiPhi should be used broadly to standardize training feasibility and batch and sequence profiles should be tailored per node type to achieve consistent utilization and predictable run stability across the fleet.

These results underscore the importance of infrastructure-aware training, particularly in large model scaling scenarios. A balance between batch size and sequence length must be strategically designed to avoid OOM errors and to fully utilize available GPU memory bandwidth and compute power.

VIII. Conclusion and Future Work

This study presents an infrastructure-aware evaluation of Llama 2 fine-tuning on CDAC's PARAM Rudra system, emphasizing both GPU memory constraints and storage I/O efficiency. Our experiments demonstrate that shorter sequence lengths enable higher batch sizes up to 256 for a sequence length of 128 without encountering OOM errors, aligning with the quadratic memory scaling of transformer models.

Results indicate that MiPhi-based aiDAPTIV acceleration facilitates efficient fine-tuning of Llama models (7B, 13B, 70B) on minimal GPU configurations (1 to 2x A100). This is achieved through hardware-aware scheduling and reduced memory bottlenecks, allowing sustained throughput (e.g., >3200 tokens/sec) in constrained environments. These insights support the feasibility of cost-effective LLM fine-tuning on existing HPC infrastructure, reducing reliance on large-scale GPU clusters.

Our findings highlight the importance of co-optimizing batch size, sequence length and I/O bandwidth to achieve scalable and stable performance, paving the way for inclusive and efficient LLM customization.

Future work will focus on large-scale multi-node training with DeepSpeed ZeRO-3, exploring varying batch sizes, sequence lengths, and LLM architectures (e.g., LLaMA-2 and LLaMA-3). It will also include direct comparisons with advanced SSD offloading frameworks such as ZeRO-Infinity [**10**] and MemAscend [**15**] to better evaluate aiDAPTIV+ against existing approaches. Additionally, experiments will extend to longer sequence lengths beyond 2048 tokens and analyze I/O bottlenecks on lower-bandwidth interconnects (e.g., PCIe Gen3), improving overall performance understanding and generalizability.

Additionally, integration with emerging optimization techniques such as quantized fine-tuning (e.g., QLoRA [**8**]) and unified memory strategies may further reduce resource footprints. As open-source LLMs scale in size and complexity, such system-aware evaluations will be critical for enabling cost-effective and accessible AI development across heterogeneous HPC infrastructures.

Acknowledgments

We sincerely thank C-DAC for providing access to the PARAM Rudra supercomputing platform under the National Supercomputing Mission (NSM). We also acknowledge the support of NSM, a joint initiative of the Department of Science and Technology (DST) and the Ministry of Electronics and Information Technology (MeitY), for advancing India's supercomputing capabilities and indigenous technologies. We are grateful to the C-DAC technical team for their support in enabling smooth execution of our experiments.

References

[1] Z. Chen et al., "A Survey on Large Language Models for Critical Societal Domains: Finance, Healthcare, and Law, " arXiv preprint arXiv: 2405.01769, 2024.

[2] A. Vaswani et al., "Attention Is All You Need," arXiv preprint arXiv:1706.03762, 2023.

[3] X.-K. Wu et al., "LLM Fine-Tuning: Concepts, Opportunities, and Challenges," *Big Data Cogn. Comput.*, vol. 9, p. 87, 2025. https://doi.org/10.3390/bdcc9040087

[4] MiPhi, "Cost-Effective Onsite LLM Training and Better Inferencing, " MiPhi Technologies, Product Datasheet, 2024. [Online]. Available: https://www.miphi.in/uploads/aiDAPTIV-Plus-PageFlyer.pdf

[5] National Supercomputing Mission (NSM), "PARAM Siddhi-AI, " C-DAC, Pune, India. [Online]. Available: https://nsmindia.in/infrastructure/nsm-systems/param-siddhi-ai/ [retrieved: Mar. 2026].

[6] National Supercomputing Mission (NSM), "Rudra-I: C-DAC's Indigenous Server Platform, " C-DAC, R&D Products. [Online]. Available: https: //nsmindia.in/rnd/rudra-i/ [retrieved: Mar. 2026].

[7] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey, " arXiv preprint arXiv: 2403.14608, 2024.

[8] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models, " arXiv preprint arXiv: 2106.09685, 2021.

[9] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs, " arXiv preprint arXiv: 2305.14314, 2023.

[10] S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith, and Y. He, "ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning, " arXiv preprint arXiv: 2104.07857, 2021.

[11] Y. Zhao et al., "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel," arXiv preprint arXiv:2304.11277, 2023.

[12] M. Rakka, M. Fournarakis, O. Krestinskaya, J. Bazzi, K. N. Salama, F. Kurdahi, A. M. Eltawil, and M. E. Fouda, "Mixed-Precision Quantization for Language Models: Techniques and Prospects, " arXiv preprint arXiv: 2510.16805, 2025.

[13] Z. Guan, H. Huang, Y. Su, H. Huang, N. Wong, and H. Yu, "APTQ: Attention-aware Post-Training Mixed-Precision Quantization for Large Language Models, " in *Proc. 61st ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2024.

[14] S. Li, H. Liu, Z. Bian, J. Fang, H. Huang, Y. Liu, B. Wang, and Y. You, "Colossal-AI: A Unified Deep Learning System For Large-Scale Parallel Training, " in *Proc. 52nd Int. Conf. Parallel Processing (ICPP)*, New York, NY, USA: ACM, pp. 766–775, 2023.

[15] Y.-C. Liaw and S.-H. Chen, "MemAscend: System Memory Optimization for SSD-Offloaded LLM Fine-Tuning, " arXiv preprint arXiv: 2505.23254, 2026.

[16] K. Lv et al., "Full Parameter Fine-tuning for Large Language Models with Limited Resources, " arXiv preprint arXiv: 2306.09782, 2023.

[17] Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. MIRACL: A Multilingual Retrieval Dataset Covering 18 Diverse Languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131, 2023. https://aclanthology.org/2023.tacl-1.63/.