Characterization of Large Language Model Development in the Datacenter

Qinghao Hu* [1,2], Zhisheng Ye* [1,3], Zerui Wang* [1,4], Guoteng Wang[1], Meng Zhang[1,2], Qiaoling Chen[1,2], Peng Sun[1,5], Dahua Lin[1], Xiaolin Wang[3], Yingwei luo[3], Yonggang Wen[2], Tianwei Zhang[2]

1 上海人工智能实验室 Shanghai Artificial Intelligence Laboratory
2 NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE
3 北京大學 PEKING UNIVERSITY
4 上海交通大學 SHANGHAI JIAO TONG UNIVERSITY
5 商汤 sensetime

Background

In the Era of Large Language Models

InternLM

Qwen

OpenAI
ChatGPT-4 Turbo

Gemini

Brainstorming | Coding | Planning | Writing

Write an email to apply for a Phd with a professor.

浦

Subject: Request for PhD Position under Your Supervision

Dear Professor [Last Name],

I hope this message finds you well. My name is [Your Full Name], and I am writing to express my keen interest in pursuing a PhD under your esteemed supervision at [University Name]. I have recently completed my Master of Science degree in [Field of Study] from [Name of University], where my research focus was on [Briefly mention thesis topic].

Throughout my academic journey, I have nurtured a deep passion for [related field/discipline] which has led me to explore various aspects of it through research projects, internships, and publications. During this time, I came across several of your notable works, including "[Title of Publication]" and "[Title of Research Project]," both of which left a profound impact on my intellectual curiosity.

The reason behind my eagerness to pursue doctoral studies under your guidance stems not only from your groundbreaking contributions to the field but also because our mutual interests seem to align significantly. The ongoing work at your lab, particularly on [specific project/area of interest], resonates strongly with my long-term career goals and aspirations.

Enclosed herewith, please find:

In the Era of Large Language Models

Developing LLMs demands extensive computational resources and can span several months.

Meta

16,000 NVIDIA A100 GPUs

Several months development

LLaMA2 Model Family
(7B, 13B, 34B, 70B)

Google AI

6,144 TPU-v4

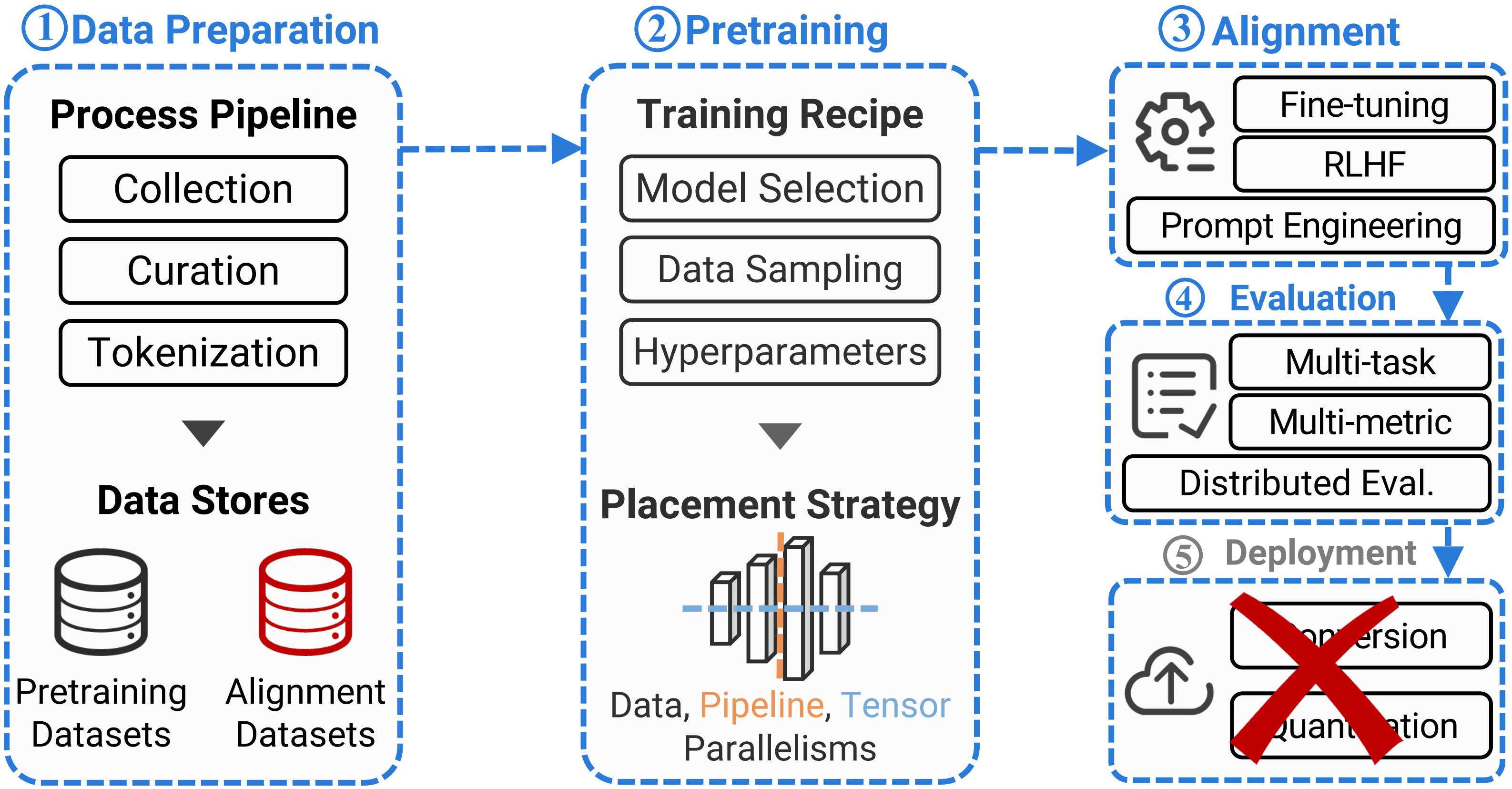
2 months pretraining

PaLM 540B

Motivation

- *What are the **characteristics of workloads** during the development of LLMs?*
- *What are the new **datacenter requirements** for running LLMs compared to prior DL workloads?*
- *How to **tailor system software** for LLMs?*

LLM Development Pipeline: An Overview



Deployment is not discussed

Acme: GPU Datacenter of Shanghai AI Laboratory

- **Two GPU Clusters Dedicated to LLM: Seren & Kalos**

Cluster	#CPUs/node	#GPUs/node	Mem(GB)	Network	#Nodes
Seren	**128**	8 × A100-80GB	1,024	1×200Gb/s	286
Kalos			2,048	5×200Gb/s	302

Totally ***4704 A100 GPUs*** interconnected by NVLink and InfiniBand

- **Model Scale:** InternLM (LLaMA like architecture)from 7B~123B

internlm/internlm2-7b
Text Generation • Updated 22 days ago • 59.2k • 30

internlm/internlm2-20b
Text Generation • Updated 21 days ago • 11.4k • 42

More models see: https://huggingface.co/internlm

Trace Overview

- **Collection Period**: traces are collected from March to August 2023
- **Number of Jobs**: a total of **684K GPU jobs** and **410K CPU jobs** in Seren and Kalos
- **Trace Sources:** (1) *Job Metadata, (2) Hardware Monitor Data*, (2)*Runtime Log*, (4) *Profiling Data*

Datacenter	**Acme** ***Shanghai AI Lab***	**Helios** *SenseTime*	**PAI** *Alibaba*	**Philly** *Microsoft*
Year	**2023**	2020	2020	2017
Duration	**6 months**	6 months	2 months	3 months
#Jobs	**1.09M**	3.36 M	1.25 M	113K
Workload	**LLM Development**	General DL workloads		
GPU Model	**A100**	1080Ti/V100	T4/P100/V100	12GB/24GB
Total #GPUs	**4,704**	6,416	6,742	2,490

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

商汤 sensetime

Alibaba

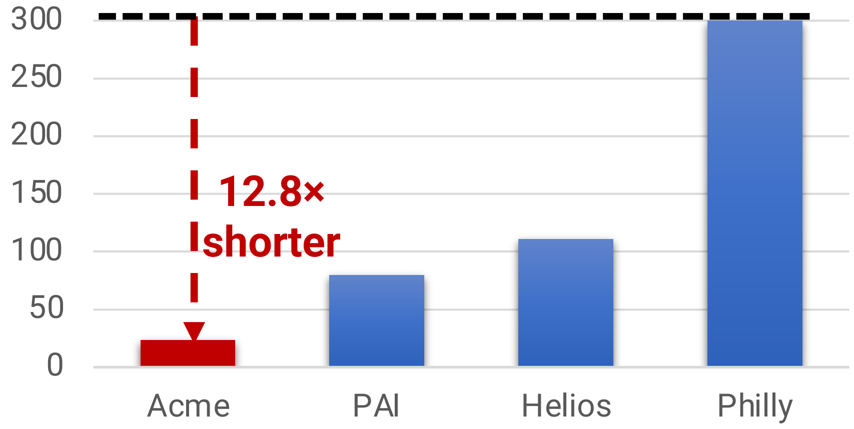
Microsoft

Trace available at https://github.com/InternLM/AcmeTrace

Datacenter Characterization

LLMs versus Prior DL Workloads

- **Shorter GPU Job Duration**



Avg. GPU Job Duration (minutes)

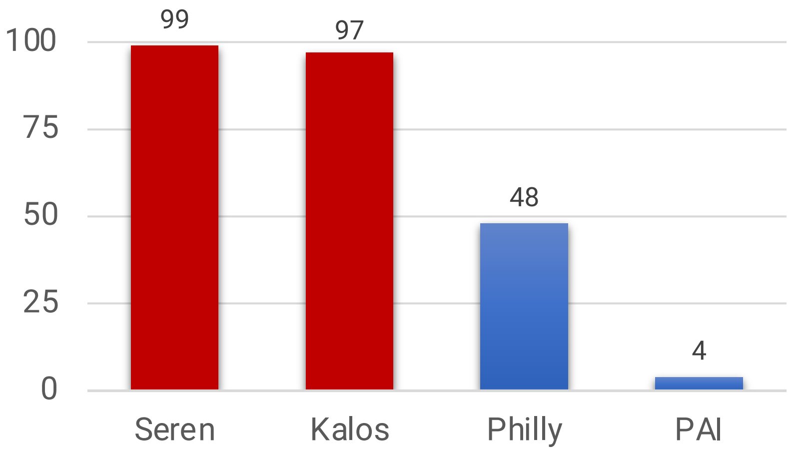
Explanation

- More advanced hardware
- Abundant resources of each job (Avg. 5~20 GPU)
- Extensive small-scale jobs
- <u>Many failed jobs (~40%)</u>

Key Insight: Need for a fault-tolerant system

LLMs versus Prior DL Workloads

- **Polarized GPU Utilization**
  - LLM workloads are either **almost entirely idle** or **fully active**



Med. GPU Utilization (%)

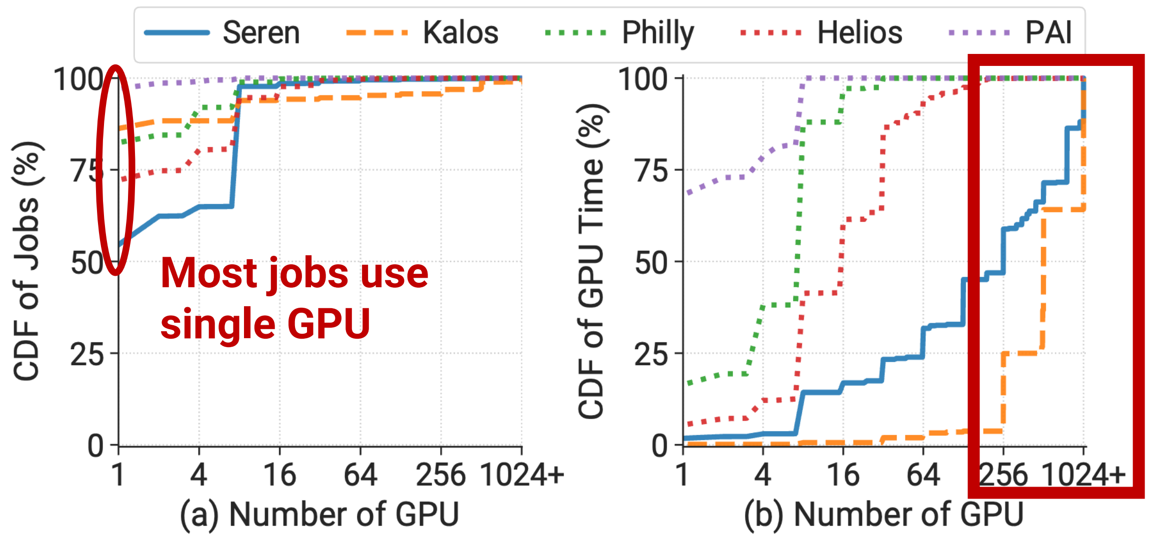
Explanation

- The computationally intensive nature of LLMs
- Many jobs fails at initialization without using any GPUs

Key Insight: GPU-sharing techniques may not be optimal for LLM development.

High-skewed Workload Distribution of LLMs

- **Kalos (b, orange)**: Top 5% of jobs, using >256 GPUs, account for ~96% of GPU time.
- **Seren (b, blue)**: Top 2% of jobs, using >64 GPUs, account for ~75% of GPU time.



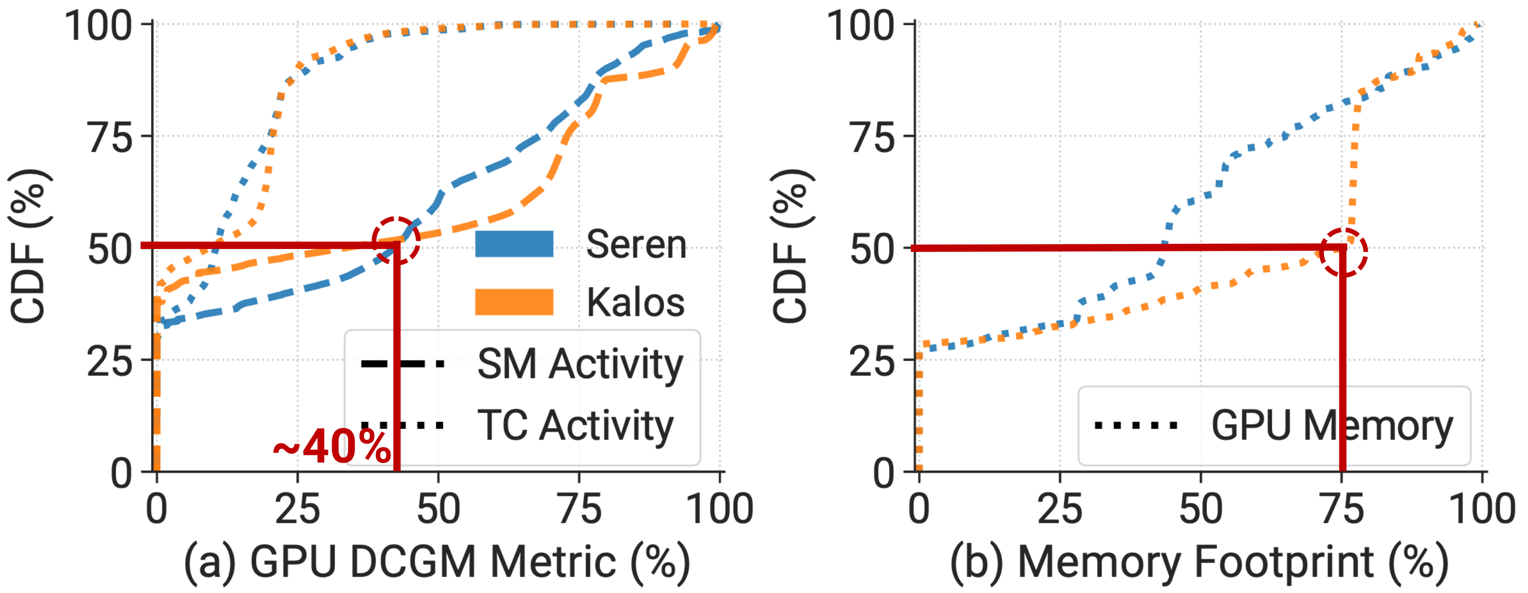
Key Insight: Design scheduler optimizations for LLM clusters considering the skew in GPU usage.

Infrastructure Utilization Patterns

Higher GPU utilization in LLM development

(a) Median **SM activity ~40%** in both clusters. (20% in PAI)

(b) The majority of GPUs consume **>75% GPU memory** in Kalos (<25% in PAI)



Workload Profiling

Pretraining

Evaluation

Profiling Pretraining Workloads: Methodology

Workloads

- 123B InternLM using 2,048 A100 GPUs

Frameworks and Strategies

- **InternEvo-v1:** 3D Parallelism (Megatron-LM)
  - with <u>pipeline parallelism=4</u>, <u>tensor parallelism=8</u>
- **InternEvo-v2:** Hierarchical ZeRO [1]
  - Parameter sharding limited in subgroups of 64 GPUs
  - Recomputing

[1] Qiaoling Chen, Diandian Gu, et al. Internevo: Efficient long-sequence large language model training via hybrid parallelism and redundant sharding. CoRR, abs/2401.09149, 2024.

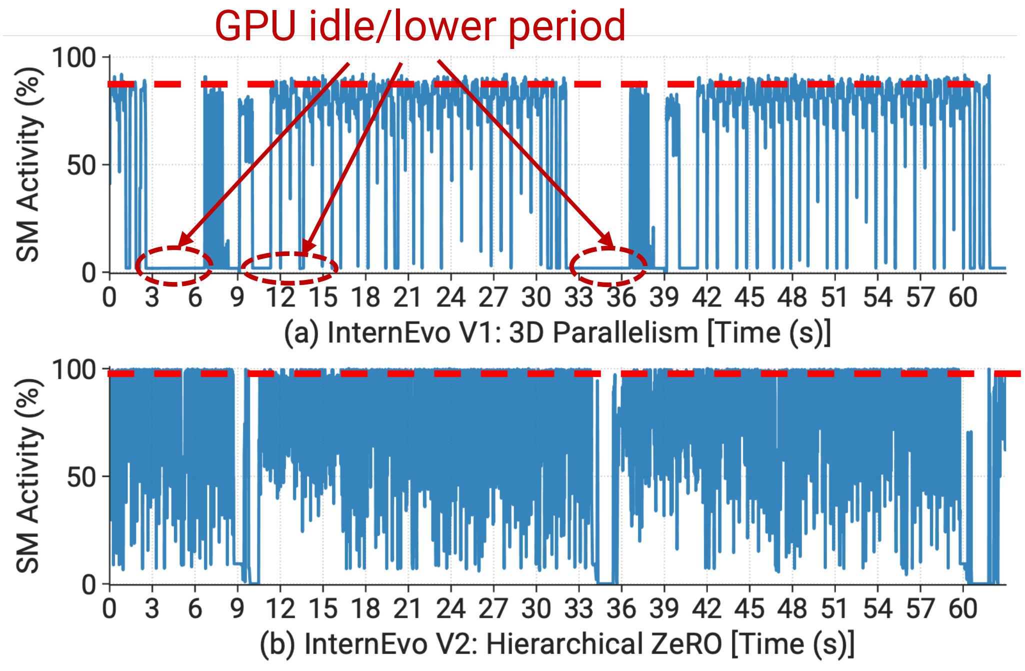
Profiling Pretraining Workloads: GPU SM Utilization

Key Improvements of *InternEvo V2*

- Reduced GPU idle/lower period
- Higher peak SM utilization

Optimizations Under the Hood

- Selective Communication Overlap
- Effective hybrid parallelism strategy

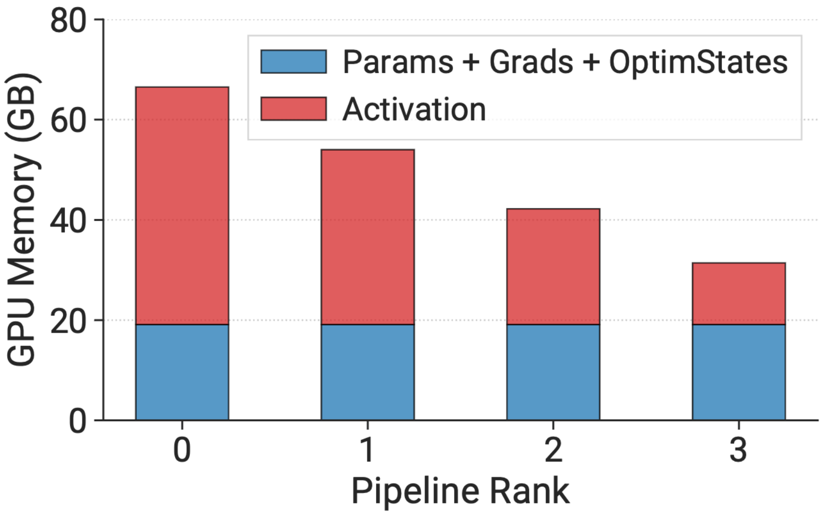


See the ***InternEvo*** paper[1] for more details

[1] Qiaoling Chen, Diandian Gu, et al. Internevo: Efficient long-sequence large language model training via hybrid parallelism and redundant sharding. CoRR, abs/2401.09149, 2024.

Profiling Pretraining Workloads: GPU Memory Footprint

Imbalance in activation size



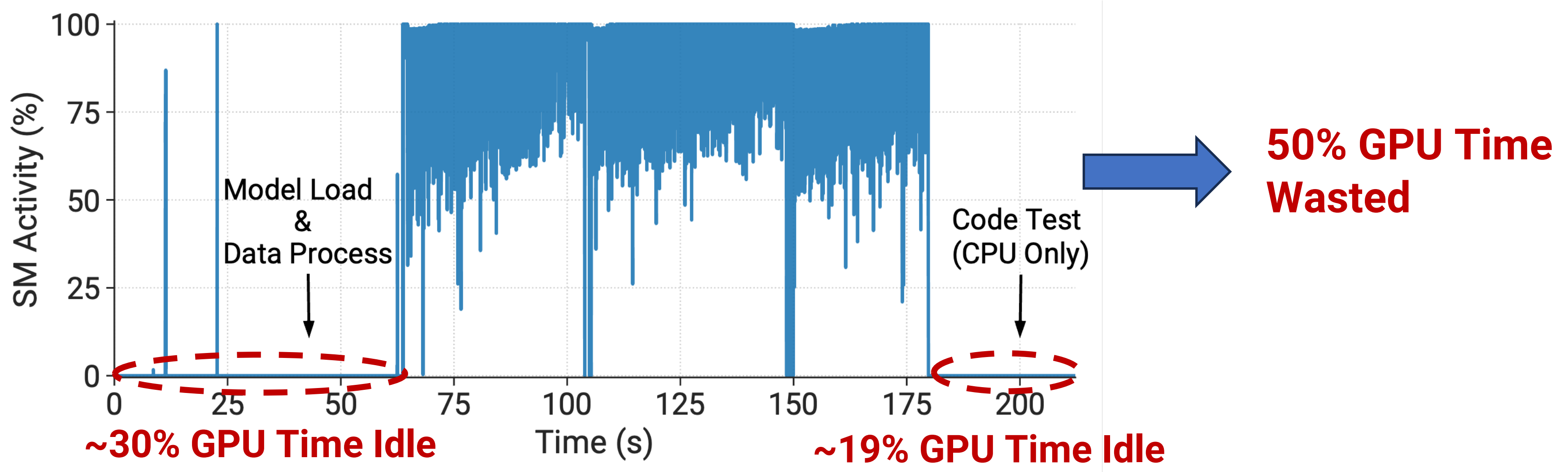
Memory consumption for each pipeline rank in InternEvo V1 employing 1F1B [1].

Key Insight: A specialized partitioning mechanism is needed

[1] Deepak Narayanan et al. Pipedream: generalized pipeline parallelism for DNN training. In Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP '19, 2019.

Profiling Evaluation Workloads: High GPU Idle Rate

- Source1 👉 Model loading and data processing overhead
- Source2 👉 Metric computing overhead. Up to 30 mins
- An example: 7B model evaluation on HumanEval

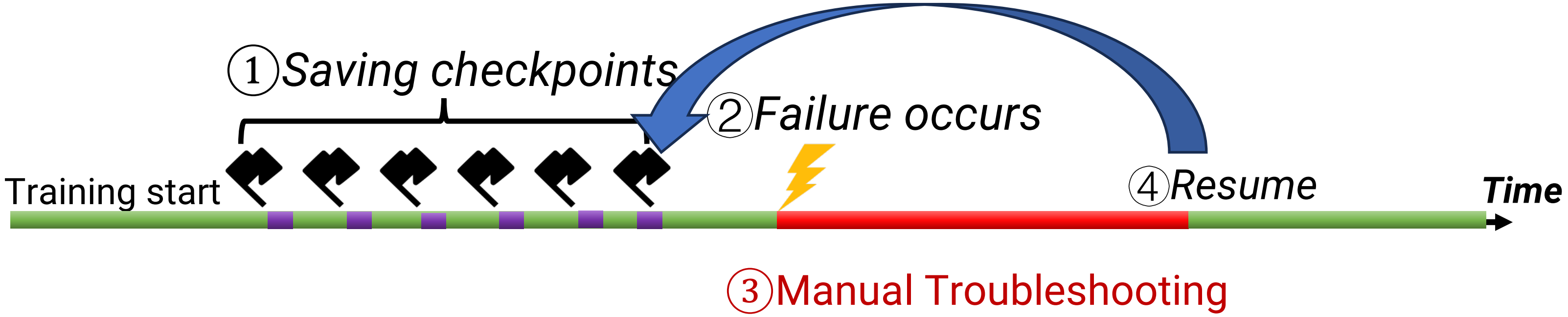


Key Insight: Improvement in the evaluation process of LLMs in **system-level**.

Failure Analysis

Impact of Job Failures: Typical Failure Recovery Process

- LLM Jobs suffer from early job termination due to various failures



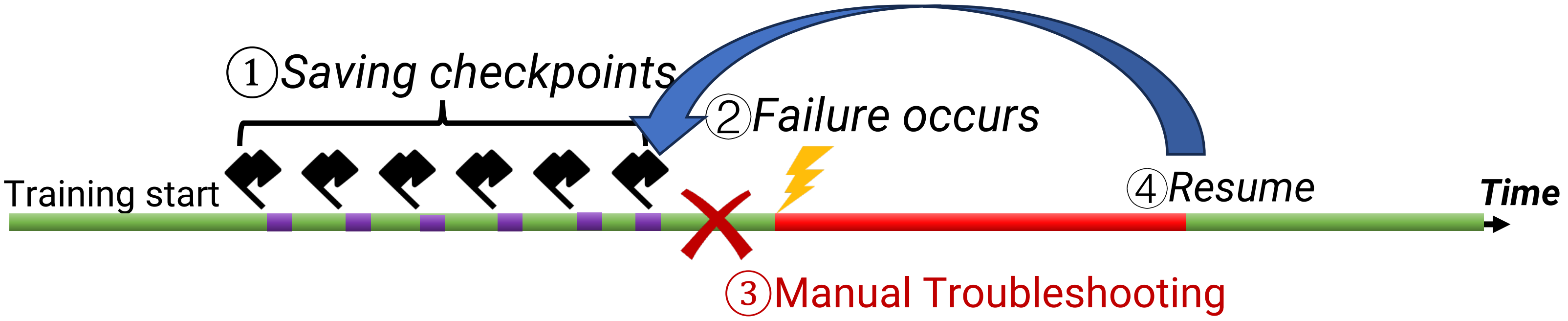
Impact of Job Failures: GPU Wastage and Progress Loss

From Checkpointing (①)

- Training get stuck during checkpoint saving (**purple chunk**)

From Failure Recovery (②-④)

- GPU time wastage (**red chunk**)
- Training progress loss (**red x**)



Key Insight: Build a system that minimizes the failure recovery overhead

Sources of Job Failures

- Failures Happen across the Stacks

User Script

```
import torch
from models import
InternLM

def train(args):
  model = InternLM()
  for epoch in range():
    train_one_epoch(model)

if __name__ == "__main__":
  train(args)
```

Framework

InternEvo

PyTorch

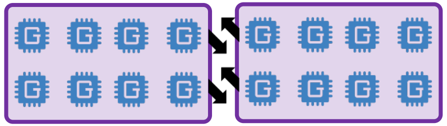
Eval Framework

Infrastructure

Scheduler

Storage

Network



CPUs, Memory, NVLink

Job Failure Analysis

Workload Composition
- ***1.3k*** pretraining jobs
- ***31K*** evaluation jobs
- *550* debug jobs

Data Sources
- Runtime log: stderr & stdout
- Hardware monitoring data

Methodology
- Identify and categories failures of the failed jobs

Category	Reason	Num	Avg. GPU Demands	Avg. TF (mins)	Total %
Infrastructure	**NVLinkError**	54	800	868.1	**30.25%**
	CUDAError	21	847	923.2	**15.77%**
	NodeFailure	16	712	1288.8	**14.30%**
	ECCError	12	680	1303.4	**11.00%**
	NetworkError	12	758	549.6	4.53%
	ConnectionError	147	29	51.9	3.44%
	S3StorageError	10	422	2317.8	2.12%
	NCCLTimeoutError	6	596	159.7	0.50%
	NCCLRemoteError	3	1152	50.5	0.15%
Framework	DataloaderKilled	6	445	1580.6	4.38%
	AttributeError	67	228	67.8	3.90%
	OutOfMemoryError	14	572	323.8	3.28%
	RuntimeError	65	441	66.4	1.72%
	AssertionError	105	413	41.7	1.24%
	ValueError	33	387	9.9	0.16%
	ZeroDivisionError	5	499	14.5	0.03%
	ModelLoadingError	104	8	2.6	0.00%
	DatasetLoadingError	5	1	1.6	0.00%
Script	FileNotFoundError	568	21	14.2	2.83%
	OSError	266	8	9.6	0.28%
	TypeError	620	18	0.9	0.06%
	Others	-	-	-	0.08%

System for LLM

Fault-tolerant LLM Pretraining

Timely Feedback for Evaluation

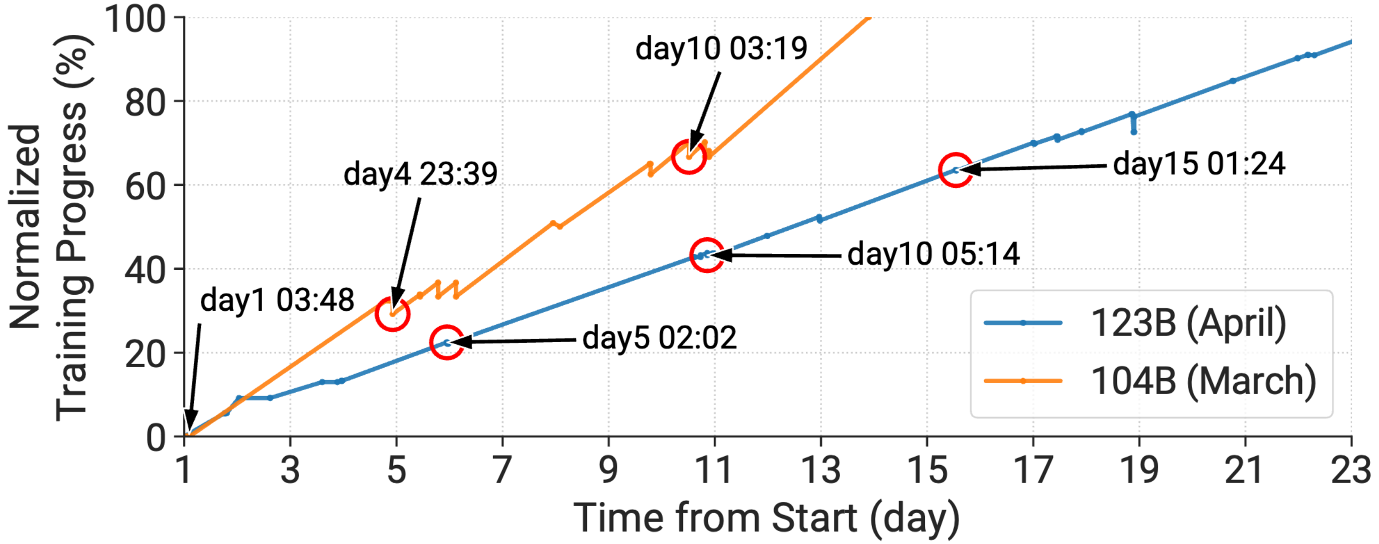
Fault-tolerant Pretraining: Technique 1

Asynchronous Checkpointing

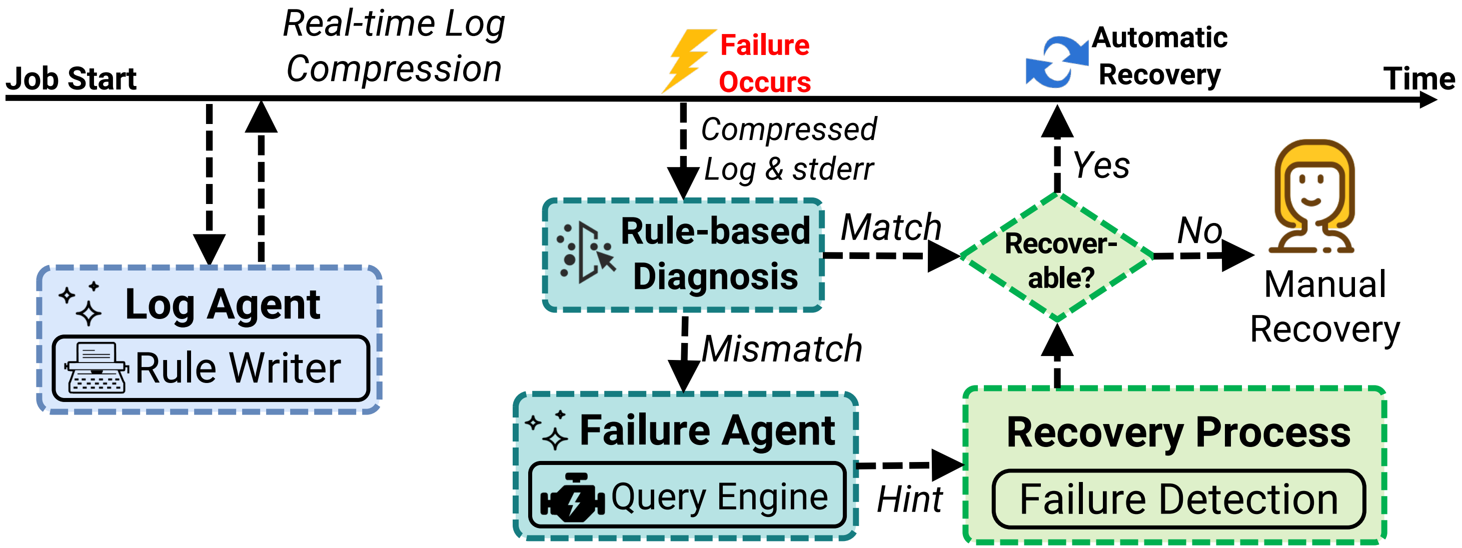
1. Store model states in host memory
2. A background process asynchronously save them to the storage

Improvements (blue line vs orange line)

- 58 × checkpointing speedup 👉 less progress loss



Technique 2: LLM-assisted Diagnosis and Recovery



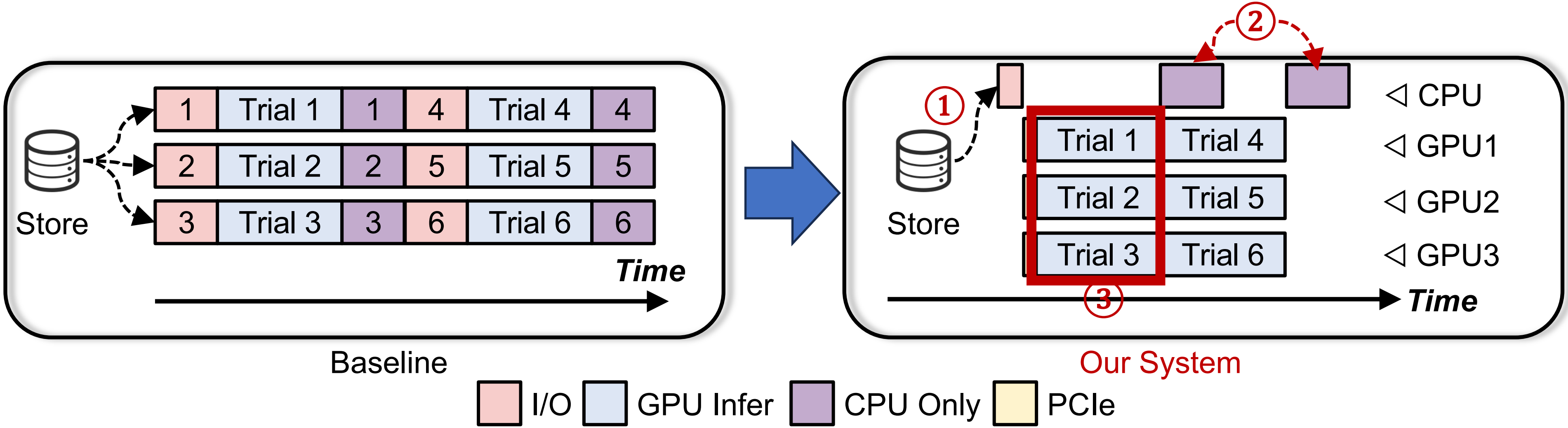
Timely Feedback for Evaluation: System Design

① Reducing I/O Overhead

- Download models once and load via PCIe

② Async Metric Computation (CPU Only)

③ Batch Trails to Improve Throughput



Improvement: Reducing the makespan by 1.3 ~ 1.8 times

More in the Paper

- More **profiling** results:
  - 123B model with 1k GPUs
  - Profiling MoE models
- Statistics on the **workload categories**
- Detailed **failure analysis**
- **Environment impact** of LLM development in Acme

Characterization of Large Language Model Development in the Datacenter

Qinghao Hu[*☆1], Zhisheng Ye[*☆3], Zerui Wang[*☆4], Guoteng Wang[☆], Meng Zhang[☆1], Qiaoling Chen[☆1]
Peng Sun[☆5], Dahua Lin[☆6], Xiaolin Wang[3], Yingwei Luo[3], Yonggang Wen[2], Tianwei Zhang[2]

[☆]*Shanghai AI Laboratory* [1]*S-Lab, Nanyang Technological University* [2]*NTU*
[3]*Peking University* [4]*Shanghai Jiao Tong University* [5]*SenseTime Research* [6]*CUHK*

Abstract

Large Language Models (LLMs) have presented impressive performance across several transformative tasks. However, it is non-trivial to efficiently utilize large-scale cluster resources to develop LLMs, often riddled with numerous challenges such as frequent hardware failures, intricate parallelization strategies, and imbalanced resource utilization. In this paper, we present an in-depth characterization study of a six-month LLM development workload trace collected from our GPU datacenter `Acme`. Specifically, we investigate discrepancies between LLMs and prior task-specific Deep Learning (DL) workloads, explore resource utilization patterns, and identify the impact of various job failures. Our analysis summarizes hurdles we encountered and uncovers potential opportunities to optimize systems tailored for LLMs. Furthermore, we introduce our system efforts: (1) *fault-tolerant pretraining*, which enhances fault tolerance through LLM-involved failure diagnosis and automatic recovery. (2) *decoupled scheduling for evaluation*, which achieves timely performance feedback via trial decomposition and scheduling optimization.

1 Introduction

Over the years, advances in LLMs have attracted significant attention from both academia and industry owing to their impressive performance and capabilities, such as ChatGPT [2] and GitHub Copilot [3]. However, due to their immense model sizes and extensive data demands, training such models necessitates a substantial computational infrastructure with thousands of accelerators [27, 68]. Hence, it is a common practice for tech companies and cloud providers to build large-scale GPU clusters to facilitate LLM development, especially after the popularity of ChatGPT. Nevertheless, it is non-trivial to perform efficient LLM development on such high-cost infrastructure. Developers often confront numerous issues and challenges, including frequent hardware failures [64, 96], intricate parallelization strategies [68, 113], unstable training progress [1, 110], long queuing delay [104], etc.

Developing LLMs is closely intertwined with the support of GPU clusters in various aspects. A thorough analysis of cluster workloads is essential for comprehending challenges and uncovering opportunities in designing systems tailored for LLMs. However, many conclusions and implications from existing DL workloads analysis works [38, 45, 97], conducted before the rise of LLMs, are not applicable to LLM development. This is primarily due to the divergent characteristics and requirements of LLMs:

(1) *Paradigm Transition*. DL workloads generally follow a *task-specific* paradigm that trains the model on domain-specific data to tackle a particular task (e.g., translation [18]). In contrast, LLMs follow an emerging paradigm that performs self-supervised training on broad data to generate a *foundation model* [19] and further adapts to a wide range of downstream tasks. This shift signifies a substantial divergence in the model development pipeline (e.g., pretraining [85], alignment [37]) and workload characteristics from prior DL workloads (§2.1).
(2) *Tailored Software Stack*. To accommodate the enormous model size of LLMs, a series of systems implement advanced techniques to optimize the execution of LLMs. For instance, Deepspeed [79], Megatron [68] and Alpa [113] accelerate the training via hybrid parallelism or state-sharding optimizer. As for model serving, Orca [104] and vLLM [51] improve throughput via iteration scheduling or memory management.
(3) *Unified Architecture*. Prior DL workloads usually employ various model architectures (e.g., CNN [54], RNN [18]) to address diverse tasks. In contrast, LLMs commonly embrace the Transformer [93] architecture, like BERT [31], GPT-3 [20], LLaMA [91] and PaLM [27]. The architectural homogeneity suggests a high level of uniformity in the LLM development pipeline and similarity across different datacenters.

To bridge this gap, we present an in-depth study of our operational experiences in the datacenter `Acme` of Shanghai AI Laboratory. It houses two distinct clusters, `Seren` and `Kalos`, dedicated to LLM development and equipped with 4,704 A100 GPUs in total. Our analysis draws upon traces collected over a six-month period from March to August 2023, encompassing scheduler logs, infrastructure monitoring data, failure logs, and fine-grained profiling data. Our key findings and identified challenges can be summarized as follows:

- **Shorter Job Duration and Unfair Queuing Delay.** In contrast to the common stereotype that LLM workloads are usually long-term, the workloads in our datacenter exhibit 2.7∼12.8× shorter average job duration compared to the DL workloads in previous traces [38, 45, 97]. This can be

[*]Equal Contribution.

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

Acme

Seren & Kalos

Datacenter Characterization

LLM workload & resource utilization

Resource Inefficiencies

GPU time wastage of evaluation & pretraining workloads

Failure Impacts

Failures severely affect LLM development

Systems Efforts for LLM

Fault-Tolerant Pretraining System & **Timely Feedback Evaluation**

InternLM: https://github.com/InternLM

Email: wangzerui@pjlab.org.cn