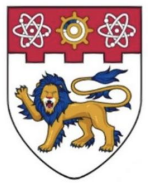
東南大學
SOUTHEAST UNIVERSITY

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 Wang3, Yingwei Luo3, Yonggang Wen2, Tianwei Zhang2



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

NSDI 2024

汇报人：王兆年

sensetime

香港中文大學

東南大學
SOUTHEAST UNIVERSITY

01

研究背景

研究背景

Qwen

Gemini

Claude

头脑风暴，提供创意

协助编程

更快的搜集资料

实时语音对话

研究背景

然而，从零开始开发 LLM 需要大量的计算资源，并且需要很长时间

Meta

使用16,000 个英伟达的 A100 GPU
训练了数月

LLaMA2 模型族
(7B,13B,34B,70B)

Google AI

使用6,144个TPU-v4
训练了两个月

PaLM
540B

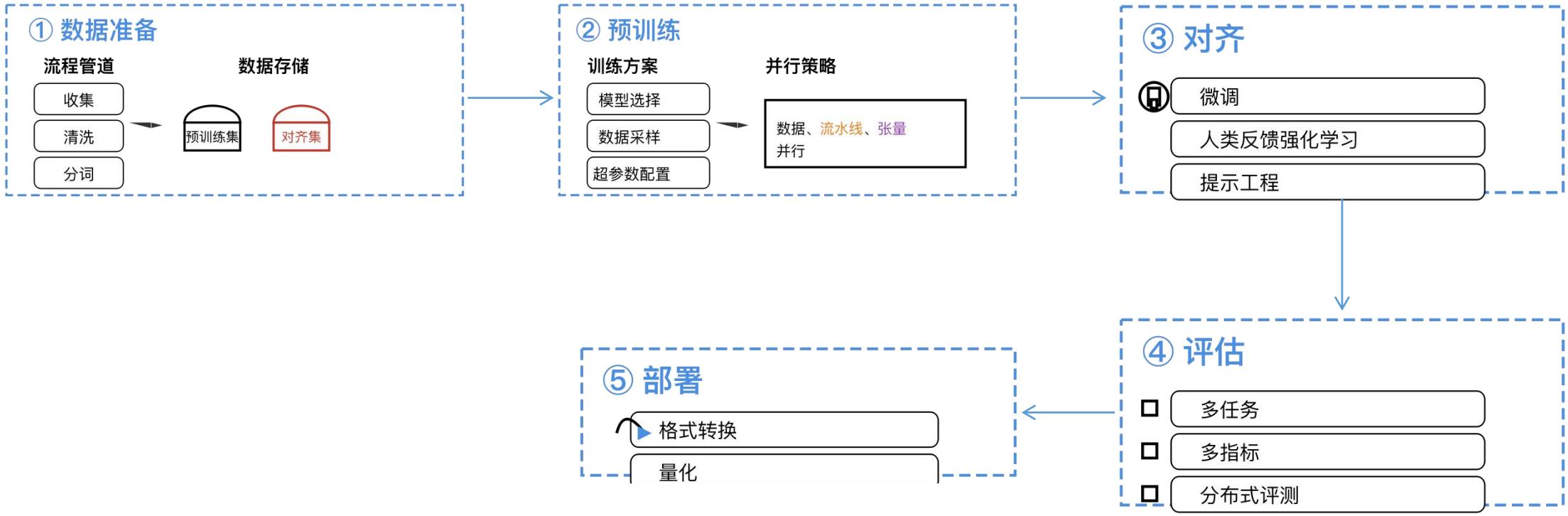
研究背景

1 **新特点** LLM 开发过程中的工作负载有何特点？

2 **新需求** 与以前的DL工作负载相比，运行LLM对数据中心有哪些新要求？

3 **新调整** 如何为 LLM 量身定制系统软件？

LLM的工作流程概况


① 数据准备
流程管道
数据存储
收集
清洗
分词
预训练集
对齐集
② 预训练
训练方案
并行策略
模型选择
数据采样
超参数配置
数据、流水线、张量并行
③ 对齐
微调
人类反馈强化学习
提示工程
④ 评估
多任务
多指标
分布式评测
⑤ 部署
格式转换
量化

02

数据中心特点

LLM的工作流程概况

Acme：上海AI实验室的GPU数据中心

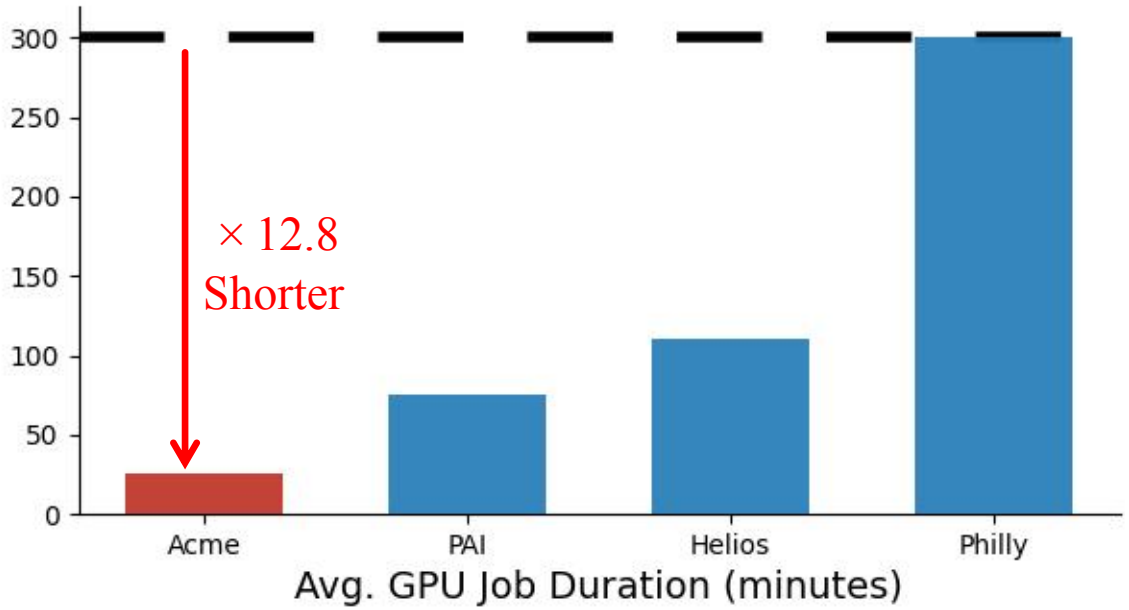
Cluster	#CPUs/node	#GPUs/node	Mem(GB)	Network	#Nodes
Seren	128	8×A100-80GB	1024	1×200Gb/s	286
Kalos			2048	5×200Gb/s	302

不同数据中心的GPU情况

	用于特定任务的DL数据中心			用于LLM的
数据中心	Philly	Helios	PAI	Acme
年份	2017	2020	2020	2023
任务持续时间	3个月	6个月	2个月	6个月
#工作数量	113K	3.36M	1.26M	1.09M
使用GPU平均数	1.9	3.7	0.7	6.3
GPU型号	12GB/24GB	1080Ti/V100	T4/P100/V100	A100
使用GPU总数	2,490	6,416	6,742	4,704

LLM vs DL

- 更短的GPU持续时间



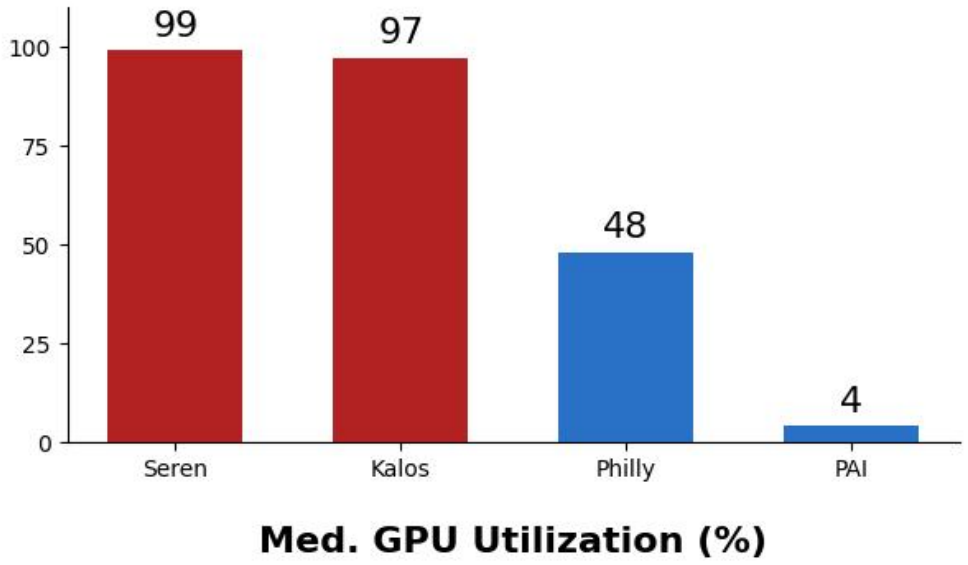
解释：

- 更先进的硬件
- 每一个工作都有更充足的资源（平均5~20个GPU）
- 大量的小规模工作
- 很多失败的工作（大约40%）

启发：需要一个容错系统

LLM vs DL

- 两极分化的GPU利用率
  - LLM工作负载要么是完全空闲，要么是完全活跃

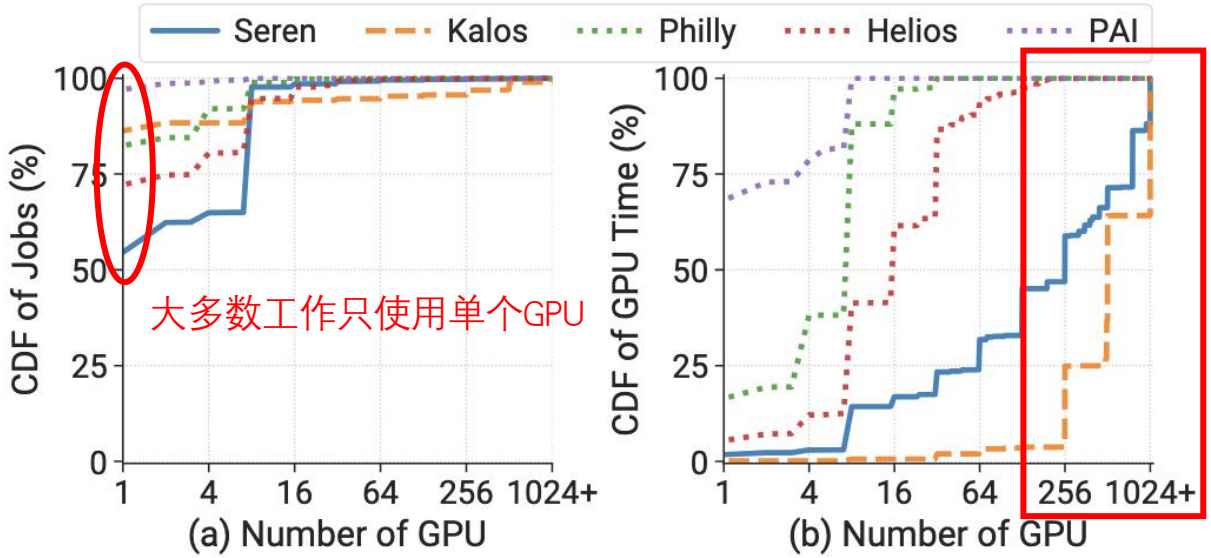


解释：

- 这是由LLM是计算密集型工作的本质导致的
- 很多工作在初始化时就失败了（并没有使用到任何GPU）

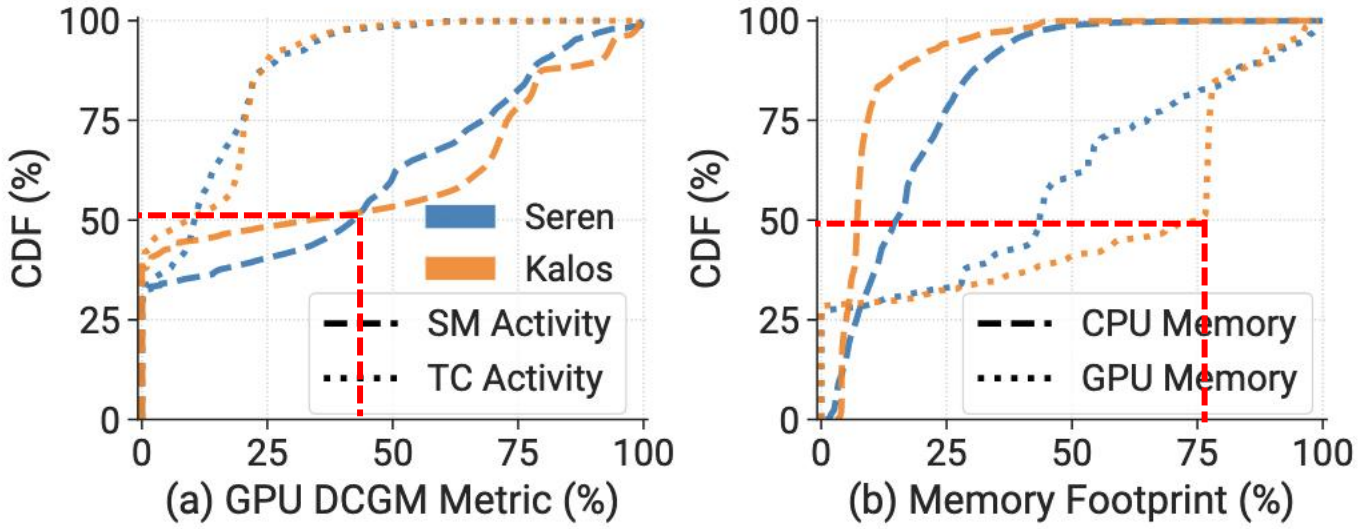
LLM vs DL

- 高度偏斜工作负载分布
  - Kalos(b, orange)：只有5%左右的工作使用了超过256个GPU，占据了约96%的GPU时间
  - Seren(b, blue)：只有2%左右的工作使用了超过64个GPU，占据了约75%的GPU时间



LLM vs DL

- LLM工作具有更高的GPU利用率
  - 两个集群的SM活动中位数均约40%（PAI是20%左右）
  - Kalos 的 GPU 内存消耗中值超过 75%



東南大學
SOUTHEAST UNIVERSITY

03

工作负载分析

3.1 预训练工作分析

3.2 评估工作分析

3.1 预训练分析

工作量：

- 123B internLM，使用 2048 个 A100 GPU

框架和策略：

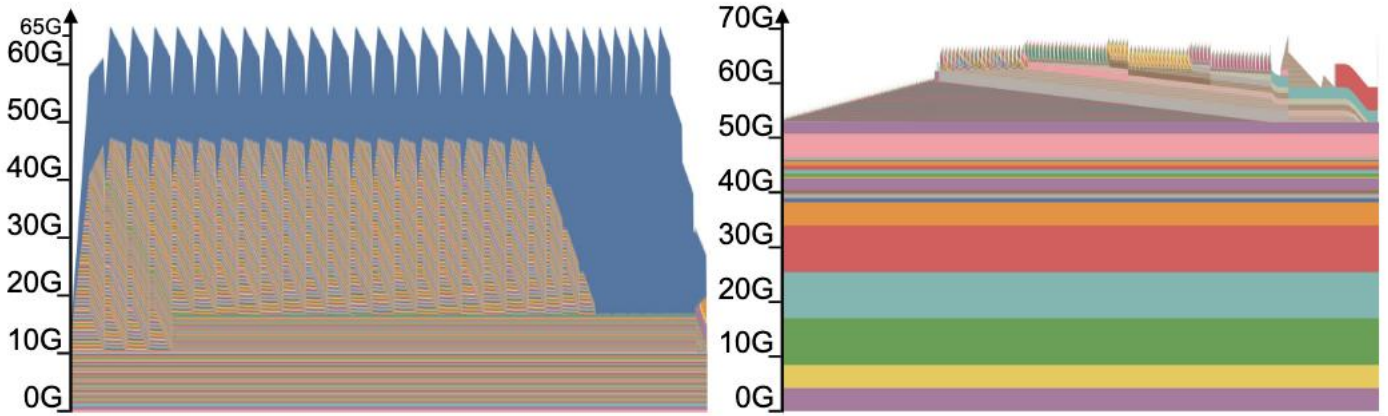
- InternEvo-v1：3D parallelism
  - 流水线并行数=4，张量并行数=8

- InternEvo-v2：Hierarchical ZeRO
  - 在 64 个 GPU 子组内进行参数分片和重新计算

3.1 预训练分析

InterEvo V2 的关键提升：

- GPU内存使用更稳定
- GPU内存使用更均匀
- 峰值更高

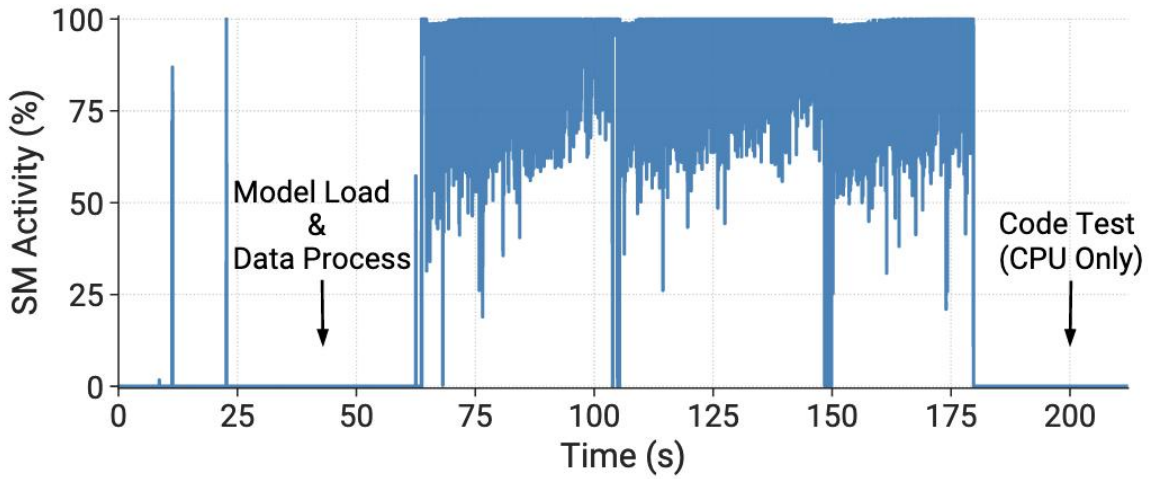


(a) 3D Parallelism　　(b) Hierarchical ZeRO

3.2 评估分析

- 高GPU空闲率

在HumanEval上进行7B参数模型的评估



解释：

- 原因1：模型加载与数据处理开销
- 原因2：度量计算开销

启示：需要在系统层面对LLM评估工作进行优化

04

故障分析

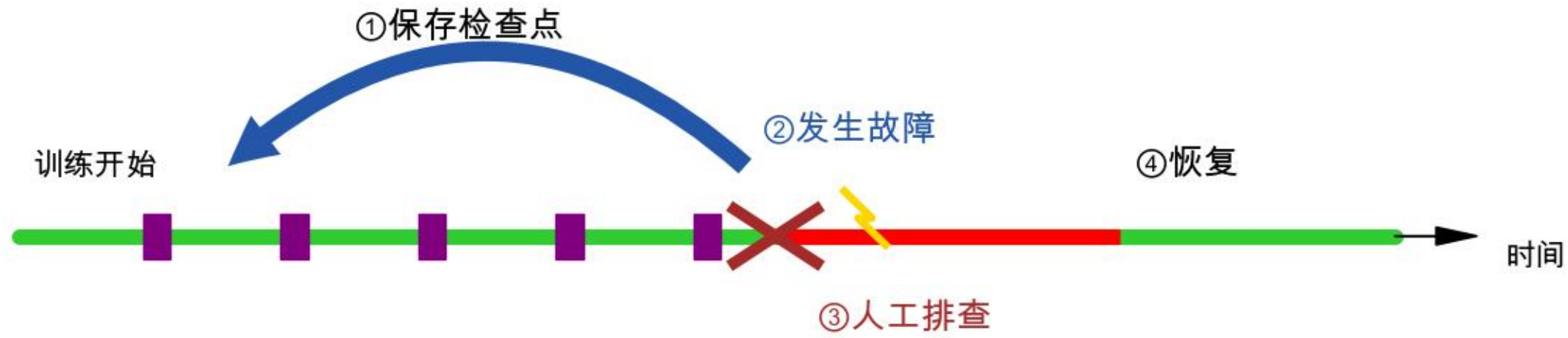
故障分析

来自检查点保存（①）

- 训练在保存检查点时卡住（紫色块）

来自故障恢复（②-④）

- GPU 时间浪费（红色区块）
- 训练进度丢失（红色 ×）



故障分析

类别	原因	数量	平均 GPU 需求	平均任务时长 (分钟)	占比
基础设施	**NVLink 错误**	54	800	868.1	**30.25%**
	CUDA 错误	21	847	923.2	**15.77%**
	节点故障	16	712	1288.8	**14.30%**
	ECC 错误	12	680	1303.4	11.00%
	网络错误	12	758	549.6	4.53%
	连接错误	147	29	51.9	3.44%
	S3 存储错误	10	422	2317.8	2.12%
	NCCL 超时错误	6	596	159.7	0.50%
	NCCL 远程错误	3	1152	50.5	0.15%
框架	数据加载中断	6	445	1580.6	4.38%
	属性错误	67	228	67.8	3.90%
	内存溢出错误	14	572	323.8	3.28%
	运行时错误	65	441	66.4	1.72%
	断言错误	105	413	41.7	1.24%
	数值错误	33	387	9.9	0.16%
	除零错误	5	499	14.5	0.03%
	模型加载错误	104	8	2.6	0.00%
	数据集加载错误	5	1	1.6	0.00%
脚本	文件未找到错误	568	21	14.2	2.83%
	操作系统错误	266	8	9.6	0.28%
	类型错误	620	18	0.9	0.06%
	其他	-	-	-	0.08%

工作量构成:

- 1.3千次预训练任务
- 3.1万次评估任务
- 550次调试任务

数据来源:

- 运行时日志：标准错误与标准输出
- 硬件监控数据

方法论:

- 识别并分类失败任务中的故障

05

方法改进

5.1 Fault-tolerant LLM Pretraining（容错式LLM预训练机制）

5.2 Timely Feedback for Evaluation（评估的及时反馈机制）

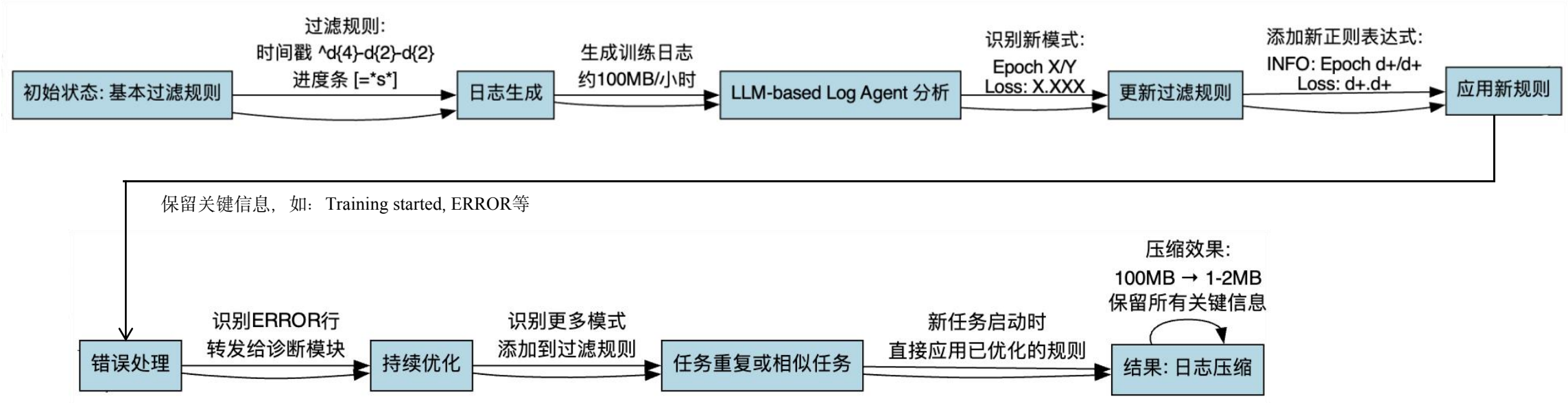
5.1.1 异步检查点

1. 将模型状态存储在主机内存中
2. 后台进程异步将其保存至存储设备

24小时训练周期：　传统方法：有效训练时间 21小时（87.5%）　异步方法：有效训练时间 23小时54分（99.6%）

5.1.2 实时日志压缩+LLM协助诊断

- 结合启发式规则与LLM技术，精准定位各类故障的根本原因



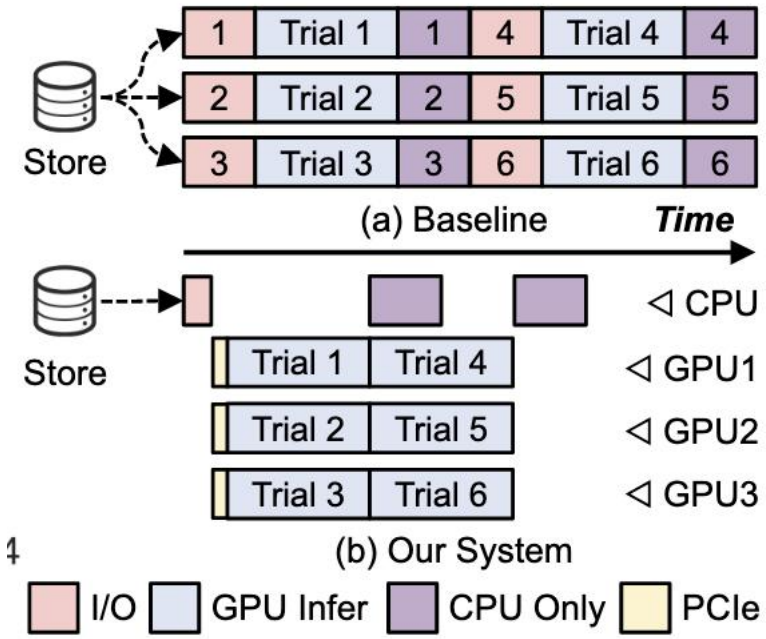

```
2025-05-01 10:00:00 INFO: Training started
2025-05-01 10:00:01 DEBUG: Initializing dataset
2025-05-01 10:00:05 INFO: Epoch 1/100
2025-05-01 10:00:10 INFO: [========      ] 40% Loss: 2.345
2025-05-01 10:00:15 INFO: [==========    ] 60% Loss: 2.123
2025-05-01 10:00:20 ERROR: CUDA out of memory
```

→

```
2025-05-01 10:00:00 INFO: Training started
2025-05-01 10:00:20 ERROR: CUDA out of memory
```

5.2.1 解耦远程模型加载

- 将模型加载过程与评估过程分离



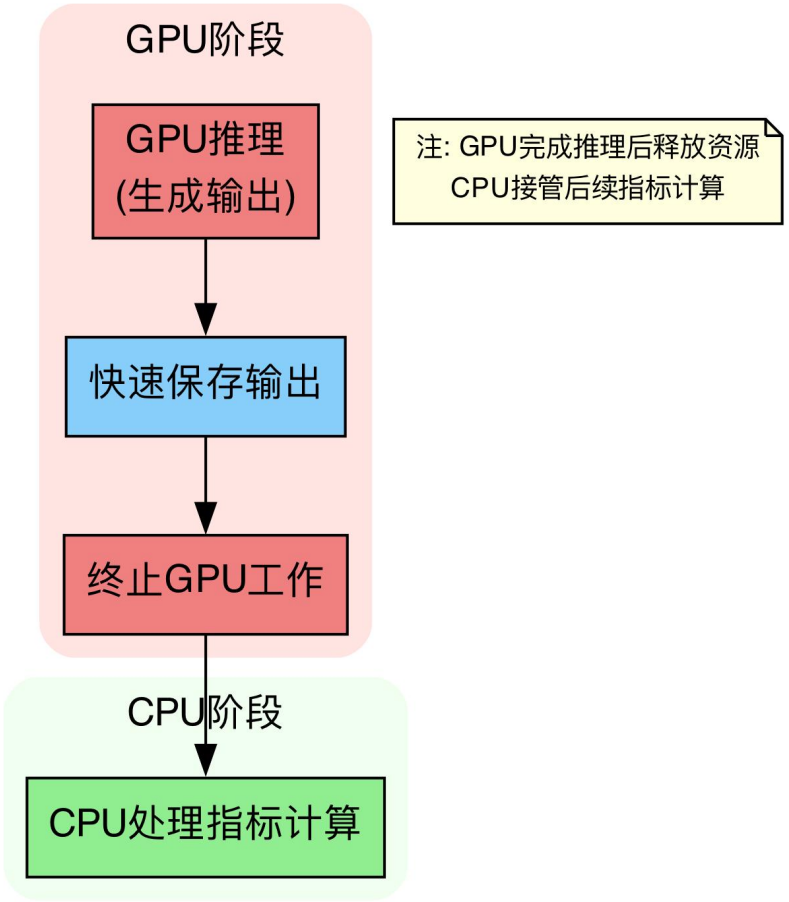
步骤:

1. 首先进行一次集中的 I/O 操作，将模型加载到 CPU 内存。
2. 然后，不同的 GPU 可以并行执行试验，通过高速 PCIe 从 CPU 内存加载模型。

改进后：完成时间缩短1.3-1.8倍

5.2.2 解耦度量计算

- 将指标计算过程从评估试验中分离出来。



步骤:

1. 模型在 GPU 上完成推理（即生成输出）。
2. 推理结果（通常是文本形式）被迅速保存到文件中
3. 保存完成后，GPU 的工作就结束了。
4. 创建 CPU 任务来执行指标计算。

06

思考

思考

1. 优化故障恢复系统：引入自动化的故障预测模型，利用机器学习（例如时间序列分析或深度学习模型）预测硬件故障的发生概率。

2. 分布式评估任务调度优化的进一步提升：可以结合深度学习调度算法，使用强化学习（RL）来优化评估任务的调度策略。

3. 跨集群负载均衡与任务调度：在多个集群之间引入资源协同调度系统。

東南大學
SOUTHEAST UNIVERSITY

感谢各位的聆听

汇报人：王兆年

2025/5/09