

面向大语言模型训练和推理系统的容错机制综述

韩淑捷¹⁾ 李柏晴²⁾ 李晓露³⁾ 黄志杰¹⁾ 赵楠楠¹⁾ 赵晓南¹⁾ 宋凌云^{1,4)} 张晓¹⁾ 尚学群^{1)*}

¹⁾(西北工业大学计算机学院, 西安 710072)

²⁾(香港中文大学计算机科学与工程系, 香港)

³⁾(华中科技大学计算机学院, 武汉 430074)

⁴⁾(西北工业大学深圳研究院, 深圳 518057)

摘 要 大语言模型的迅猛发展推动了人工智能基础设施向万卡级超大规模集群演进。然而, 随着模型参数与集群规模的急剧扩张, 故障已从偶发异常转变为常态化特征, 严重制约了训练任务的收敛效率与在线推理的服务等级协议达成。传统的通用容错机制受限于高昂的恢复开销、粗放的控制粒度以及对模型语义特征的感知缺失, 已难以应对当前系统的韧性挑战。本文基于近五年百余篇顶级学术会议与期刊论文, 针对不同故障类型, 系统综述了面向大语言模型负载特性的容错技术进展。文章首先构建了涵盖“训练/推理”工作负载与“中断型故障/迟滞型故障/静默数据损坏”故障类型的二维正交分类体系; 深入剖析了不同故障模式的传播机理及其对大语言模型系统可靠性的具体影响; 继而详细梳理了训练阶段和推理阶段分别针对三类故障类型的容错机制。最后, 本文展望了大语言模型系统容错机制的未来研究方向。

关键词 大语言模型; 容错机制; 分布式训练; 在线推理; 中断型故障; 迟滞型故障; 静默数据损坏

中图法分类号 **** **DOI 号** *投稿时不提供 DOI 号*

A Survey on Fault Tolerance Mechanisms of Training and Inference Systems for Large Language Models

HAN Shu-Jie¹⁾ LEE Pak-Ching²⁾ LI Xiao-Lu³⁾ HUANG Zhi-Jie¹⁾ ZHAO Nan-Nan¹⁾ ZHAO Xiao-Nan¹⁾
SONG Ling-Yun^{1,4)} ZHANG Xiao¹⁾ SHANG Xue-Qun¹⁾

¹⁾(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072)

²⁾(Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong)

³⁾(School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

⁴⁾(Research & Development Institute of Northwestern Polytechnical University in Shenzhen, Shenzhen 518057)

Abstract The growth of large language models (LLMs) has driven artificial intelligence infrastructure toward clusters comprising tens of thousands of GPUs. As model parameter counts and cluster sizes increase, failures have become routine, impairing training convergence efficiency and compliance with service level agreements

收稿日期: 2026年1月3日; 最终修改稿收到日期: 2026年9月20日. 本课题得到新一代人工智能国家科技重大专项(No. 2026ZD0128201)、青年科学基金项目(C类)(No.62502391和No.62302175)、国家自然科学基金面上项目(No.62272394)、国家自然科学基金联合基金重点项目(No.U25B2022)、广东省基础与应用基础研究基金(No.2024A1515011715和2025A1515012995)资助. 韩淑捷, 女, 1994年生, 博士, 副教授, 计算机学会(CCF)会员, 主要研究领域为AI赋能存储系统、存储系统可靠性. E-mail: shujiehan@nwpu.edu.cn. 李柏晴, 男, 1978年生, 博士, 教授, 主要研究领域为存储系统和计算机网络可靠性, 包括容错、恢复、安全及性能保证. E-mail: pcleee@cse.cuhk.edu.hk. 李晓露, 女, 1993年生, 博士, 讲师, 计算机学会(CCF)会员, 主要研究领域为存储系统的可靠性、大模型的存储优化. E-mail: lixl666@hust.edu.cn. 黄志杰, 男, 1983年生, 博士, 副教授, 计算机学会(CCF)会员, 主要研究领域为存储系统的可靠性与安全性、云存储、编码理论、区块链技术. E-mail: jayzy.huang@nwpu.edu.cn. 赵楠楠, 女, 1987年生, 博士, 副教授, 计算机学会(CCF)会员, 主要研究领域为分布式存储系统、闪存、容器技术、内存架构. E-mail: nannanzhao@nwpu.edu.cn. 赵晓南, 女, 1979年生, 博士, 副教授, 计算机学会(CCF)会员, 主要研究领域为分布式文件系统管理和优化. E-mail: zhaoxn@nwpu.edu.cn. 宋凌云, 男, 1988年生, 博士, 副教授, 计算机学会(CCF)会员, 主要研究领域为可视化、跨媒体认知与推理、计算机视觉. E-mail: lysong@nwpu.edu.cn. 张晓(通信作者), 男, 1978年生, 博士, 教授, 计算机学会(CCF)高级会员, 主要研究领域为存储系统、计算机网络、AI系统. E-mail: zhangxiao@nwpu.edu.cn. 尚学群(通信作者), 女, 1973年生, 博士, 教授, 计算机学会(CCF)高级会员, 主要研究领域为数据挖掘、机器学习、大数据、生物信息学. E-mail: shang@nwpu.edu.cn.

(SLAs) for online inference. These challenges are amplified by extensive model states, tightly coupled execution, and stateful autoregressive generation. Traditional general-purpose fault tolerance mechanisms incur high recovery overheads, operate at coarse control granularities, and lack awareness of model semantics, limiting their effectiveness in maintaining system resilience. Drawing on more than 100 papers published at leading conferences and in journals over the past five years, this survey reviews fault tolerance mechanisms tailored to LLM workloads. We construct a two-dimensional orthogonal taxonomy combining two workload stages, training and inference, with three failure categories: fail-stop failures, fail-slow failures, and silent data corruption (SDC). The resulting six technical domains provide a framework for examining how workload characteristics shape failure impacts and protection mechanisms. We first analyze the causes, propagation mechanisms, and consequences of these failures. Fail-stop failures interrupt computation and propagate through communication dependencies, causing training stalls or inference context loss. Fail-slow failures leave components operational but degrade performance, extending training iterations through synchronization barriers and increasing inference tail latency. SDC introduces erroneous computations or corrupted states without notifications. These errors can spread through gradient aggregation and parameter updates during training, or through autoregressive generation during inference, compromising convergence and output quality. This analysis connects failure behaviors to workload-specific reliability requirements. For training, we examine mechanisms that preserve distributed states and reduce interruption costs, including proactive fault detection and diagnosis, efficient checkpointing, rapid runtime recovery, and model-aware state reduction. We also review straggler detection, performance prediction, and adaptive task rescheduling for mitigating fail-slow failures. For SDC, we discuss protection based on algorithmic properties and lightweight algorithm-based fault tolerance, emphasizing tradeoffs between error coverage and computational overhead. These techniques address training progress, resource efficiency, and numerical correctness. For inference, the survey focuses on service continuity, latency, and generation quality. Mechanisms for fail-stop failures include context migration, streaming backup, and rapid state reconstruction, particularly key-value cache preservation and recovery. Techniques for fail-slow failures include prefill-decoding disaggregation, dynamic routing in mixture-of-experts models, and scheduling that accounts for resource heterogeneity and interference. For SDC, we examine sensitivity-aware selective protection and coordinated verification at the software and algorithm levels. These mechanisms protect stateful generation while meeting service latency requirements under constrained resources. Finally, we identify future research directions across the six domains. Training requires hardware-software co-design for low-overhead state persistence, model-driven autonomous diagnosis and proactive mitigation, and sensitivity-guided protection against silent errors. Inference calls for decoupled persistence and migration of key-value caches, adaptive scheduling and elasticity guided by service objectives, and semantic-aware verifiable inference. These directions emphasize coordinated designs that account for failure behavior, workload characteristics, and protection costs. By connecting failure propagation with workload-specific protection and recovery, this survey provides a basis for understanding approaches and identifying research opportunities in resilient LLM training and inference systems.

Key words large language models; fault tolerance mechanisms; distributed training; online inference; fail-stop failures; fail-slow failures; silent data corruption

1 引言

大语言模型 (Large Language Model, LLM) 是以 Transformer^[1]架构为核心、海量数据为基座、万卡级异构集群为载体的新一代人工智能基础设

施。随着 Llama-3^[2]、GPT-4^[3]等万亿参数模型的涌现, LLM 正从单一的算法模型演进为高度复杂的分布式系统。这种规模化的演进虽显著提升了系统性能, 但模型参数量与集群规模的双重膨胀, 极大地加深了对底层硬件、通信网络及存储系统的依赖, 致使故障场景下的系统可靠性问题愈发严峻。

在支撑 LLM 训练与推理的超大规模集群中，故障已从偶发事件转变为常态。以 Meta 公司^[2]为例，其在使用 1.6 万个图形处理器（Graphics Processing Unit, GPU）[⊖]训练 Llama-3 模型的 54 天内，观测到 466 次任务中断，其中 419 次为不可预测的突发故障（平均故障间隔时间仅约 3 小时），且由 GPU 硬件及网络通信引发的失效占比接近 70%。根据表现形式，这些故障可被归纳为三种典型形态：首先是中断型故障（*fail-stop failures*），表现为软硬件的彻底失效，这不仅迫使耗时的训练任务中断回滚，更会导致推理请求的上下文丢失而直接造成服务不可用；其次是较难检测的迟滞型故障（*fail-slow failures*），即组件虽未宕机但性能显著退化，除常规硬件热节流外，LLM 复杂的计算特征引发的资源争抢或负载不均亦是导致严重尾部延迟的主因；最后是更为隐蔽的静默数据损坏（*Silent Data Corruption, SDC*），此类硬件层面的隐蔽计算错误往往在系统无报错的情况下发生，极易在无感知中破坏模型训练的收敛性，或在推理阶段诱发逻辑谬误与事实性错误。

相较于传统分布式系统，LLM 在训练与推理阶段的负载展现出状态规模巨量化、计算架构复杂化及故障表征隐蔽化的三大显著特征。这使得基于“故障-停止”假设的传统容错机制（如通用进程级检查点、简单节点重启）^[4]难以应对当前系统的中断型故障、迟滞型故障及静默错误损坏。这种供需错配正驱动容错技术体系发生深刻变革：

- **状态规模的巨量化（难以承受粗粒度回滚）**：传统粗粒度的“崩溃-回滚”策略^[5]假定状态规模有限且易于快照。然而，LLM 训练涉及万亿级参数与动态优化器状态，推理阶段更需维护随上下文线性增长的键值缓存（Key Value Cache, KV Cache）^[6]。因此，这种全量回滚模式会导致昂贵的算力浪费或服务中断。这驱动容错手段从全量静态快照向**细粒度动态管理演进**，以实现低开销的状态保护。
- **计算架构的复杂化（难以容忍单点迟滞）**：早期深度学习多采用同构密集计算，而 LLM 引入了训练阶段的混合并行^[7]与推理阶段的混合专家（Mixture of Experts, MoE）^[8]、预填充/解码（Prefill/Decode）分离^{[9][10]}等新范式，致使计算图依赖高度复杂。在此架构下，单点的迟滞型故

障不再仅是局部慢节点，而是会引发流水线气泡或队头阻塞，严重破坏端到端的服务等级协议（Service Level Agreement, SLA）。这驱动容错手段从二元故障处理向**性能感知调度演进**，以消除异构资源间的干扰。

- **故障表征的隐蔽化（难以依赖显式异常）**：传统容错依赖硬件或操作系统抛出的显式异常来触发恢复。然而，静默数据损坏具有极高的隐蔽性，往往在系统无报错的情况下破坏模型训练收敛性^[11]或导致推理生成逻辑谬误^[12]。这驱动容错防线从系统层上移，向**算法质量保障演进**，构建面向生成质量的防护体系。

本文中，我们将可靠性（Reliability）定义为系统追求无故障运行的目标，而将韧性（Resilience）定义为系统在遭遇故障时的综合生存能力：既包括通过弹性调度、状态迁移等手段从中断与迟滞型故障中快速恢复，也涵盖对静默数据损坏等隐蔽性错误的感知与防御，以在不确定环境中同时维持服务的可用性与计算的正确性。当前研究正呈现出从单纯追求可靠性向构建系统韧性演进的趋势。基于此，本文以故障类型为经，以工作负载阶段（训练/推理）为纬，构建了如图 1 所示的分类框架，将现有 LLM 训练和推理系统的容错机制归纳为六个核心技术域：

- **面向训练的中断型故障容错**：针对高昂的停机恢复开销，构建了从主动故障诊断与探测出发，涵盖高效检查点技术、集群运行时资源快速容错以及模型结构状态约减的多跨层协同韧性机制；
面向训练的迟滞型故障缓解：针对慢节点拖慢全局同步，技术重心从被动的落后者检测，演进为基于性能预测的主动任务重调度与验证机制；
- **面向训练的静默数据损坏防护**：针对隐蔽的数值错误，技术重心从硬件级冗余，演进为基于算法特征分析与轻量级基于算法容错的精度保障技术；
- **面向推理的中断型故障恢复**：针对在线服务的高可用需求，聚焦于请求上下文的热迁移、流式备份以及基于状态物化的快速恢复机制；
- **面向推理的静默数据损坏容错**：针对生成质量受损风险，聚焦于敏感度感知的选择性防护及软件

⊖ 除非特别说明，本文用 GPU 泛指各类人工智能加速器。

• 本文中的“训练”特指预训练，不涉及微调。

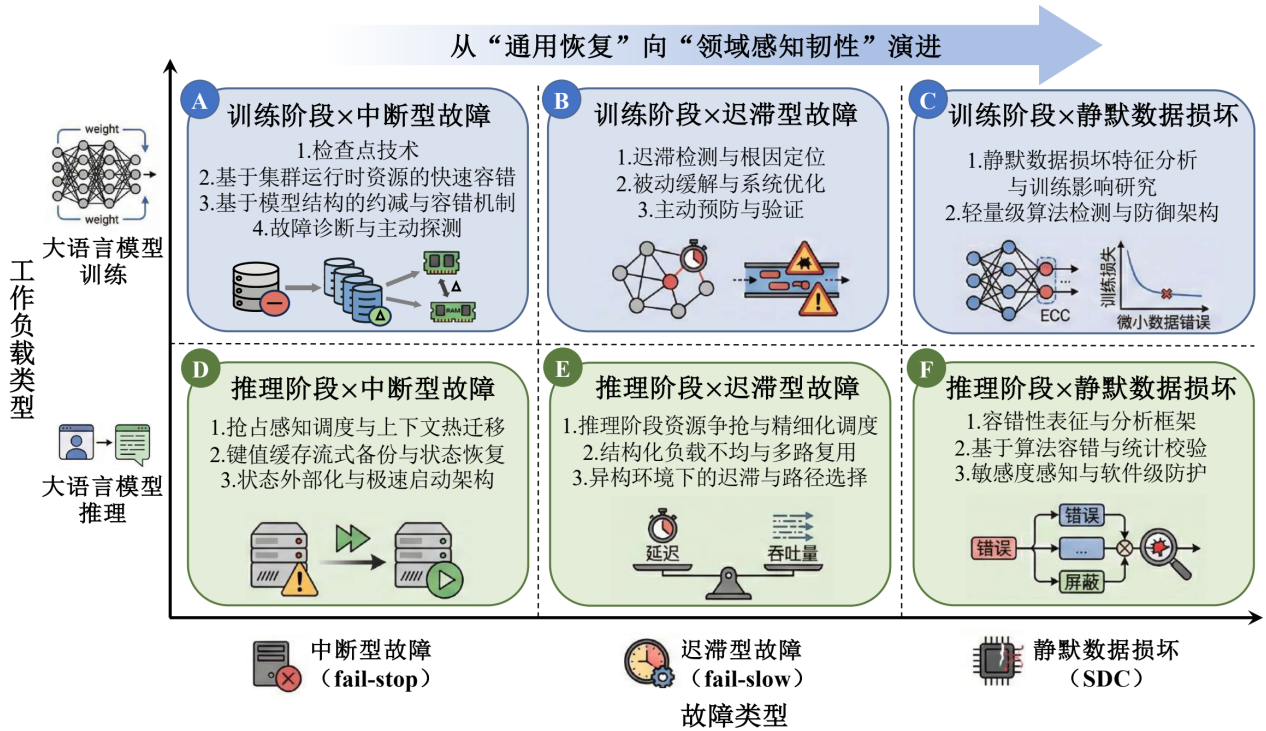


图1 LLM系统“工作负载-故障类型”容错研究概览

级与算法级的协同校验方案。

1.1 现有综述

近年来,随着 LLM 技术的迅猛发展,学术界涌现出大量从不同视角切入的综述工作。在容错领域,文献^[13]面向分布式图计算作业,构建了基于成本、效率和质量维度的评估框架,并详细论述了检查点、日志、复制及算法补偿四类容错机制;文献^[14]则与本文的研究范畴最为接近,该工作聚焦于深度神经网络(Deep Neural Network, DNN)的训练容错,系统梳理了检查点演进、非检查点容错及系统模块级容错技术,并提出了基于“通信-存储-计算”维度的检查点优化分类体系。然而,上述综述分别侧重于迭代式图计算负载或通用的 DNN 训练场景,难以完全适配 LLM 负载的独特性,未能充分涵盖如推理阶段的静默错误传播机制、键值缓存引发的状态空间爆炸等新挑战。

除容错领域外,近期关于 LLM 的综述主要集中在模型演进与性能优化两个维度。在模型演进维度,现有工作回顾了 LLM 的发布历程与演进技术^[15]、梳理了模型精度的评测方法^[16]以及探讨了可解释性难题^[17]。在系统性能优化维度,文献^[18]深入分析了大规模存储系统面临的数据碎片化与异构扩容瓶颈;预印本系列综述则分别聚焦于高效推理^[19]、数据交互^[20]及键值缓存加速管理^[21];此外,文献^[22]与文献^[23]分别从服务化趋势和“算法-系统”

协同视角总结了生成式 LLM 的演进路径。尽管上述综述深入探讨了性能提升技术,但普遍侧重于“效率”指标,而在一定程度上忽视了超大规模集群中故障频发对系统“可靠性”构成的严峻威胁。相较而言,本文构建了如图 1 所示的容错技术演进框架,主要贡献如下:

- **纵向维度——全生命周期覆盖:** 鉴于大规模集群常采用训练与推理混合部署模式,且两阶段的工作负载特征迥异,本文打破了以往仅关注单一阶段的局限,将 LLM 预训练与在线推理同时纳入研究范畴,对全生命周期的故障影响与容错机制进行了一体化分析;
- **横向维度——故障全谱系分析:** 本文覆盖了中断型故障、迟滞型故障及静默数据损坏三大类故障,深入剖析了从底层硬件组件失效到上层软件配置错误的复杂根因,揭示了不同故障对系统吞吐性能与模型生成精度的双重影响;
- **时间维度——前沿时效性聚焦:** 如图 2 所示,本文重点关注 2023-2026 年间的最新研究成果,引用的 102 篇前沿文献(占比约 69%) 涵盖了操作系统、文件系统、计算机网络、高性能计算、机器学习系统领域等顶级会议与期刊的最新进展,确保了综述内容的前沿性。

1.2 论文结构

本文后续章节组织如下:第 2 节阐述大语言模

型系统的研究现状与容错挑战，聚焦训练阶段的混合并行机制与故障级联效应，以及推理阶段的键值缓存瓶颈、服务化架构革新及其引发的状态敏感性与通信脆弱性。第3节剖析 LLM 系统的可靠性威胁，从中断型故障、迟滞型故障及静默数据损坏等三类故障，解析其来源、分布特征及对训练和推理任务的影响。第4节与第5节分别聚焦于训练和推理两类负载，系统梳理了针对上述三类故障的关键容错机制与技术演进。第6节展望未来研究方向；第7节总结全文。

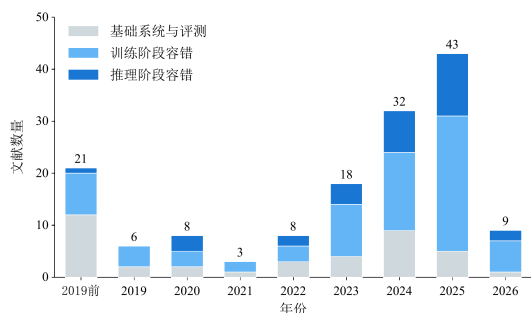


图2 本文所引用的文献年份分布图

2 大模型系统现状及容错挑战

LLM 作为深度学习发展的最新形态，依托 Transformer^[1]架构与“缩放法则”(Scaling Law)^[24]，通过持续拓展网络深度与参数规模，实现了海量数据高维表征与生成能力的质变。然而，这种由百万级向万亿级参数规模^[2]的跨越，深刻重构了底层系统的运行范式：训练阶段演变为极度依赖高带宽互联的长周期、大规模同步作业，而推理阶段则转型为对延迟与吞吐极为敏感的高并发在线服务。这两种截然不同的负载特征，叠加万卡级异构集群固有的硬件故障率，使得单纯追求系统可靠性的传统容错机制难以维系，迫使系统必须在保证训练收敛性和推理可用性的前提下，应对前所未有的容错挑战。

2.1 大语言模型的训练范式与容错挑战

LLM 训练是一个基于海量数据进行“前向传播-反向传播-参数更新”的高强度迭代循环过程。然而，随着模型规模的指数级增长，这一标准流程在状态管理与计算模式上呈现出显著的独特性，深刻重塑了容错机制的设计约束。

庞大的训练状态空间：LLM 训练的容错挑战首先源于其需要持久化的状态空间远超静态的模型权重本身。在故障恢复过程中，系统必须从存储

系统中精准还原随模型规模膨胀的动态状态，这构成了容错 I/O 开销的决定性因素。具体而言，优化器状态占据了数据量的主导地位——在使用 Adam^[25]等主流优化器进行混合精度训练时，系统需为每个参数维护 32 位浮点数精度的主权重备份及一阶、二阶动量。以 Llama-3-405B^[2]为例，仅其优化器状态的容量便高达约 4.9TB，若叠加模型权重，单次快照需持久化的静态状态总数据量将突破 6.5TB。这种 TB 级的海量状态数据，在万卡集群并发写入时会产生巨大的 I/O 突发流量，使得高频的检查点保存成为拖累训练效率的沉重负担（详见 4.1 节）。

基于分治的多维混合并行：现代 LLM 训练普遍采用复杂的混合并行^[7]策略（如图 3 所示），通过将计算图与状态空间在成百上千个设备间进行多维切分，构建了高度依赖的分布式计算拓扑：

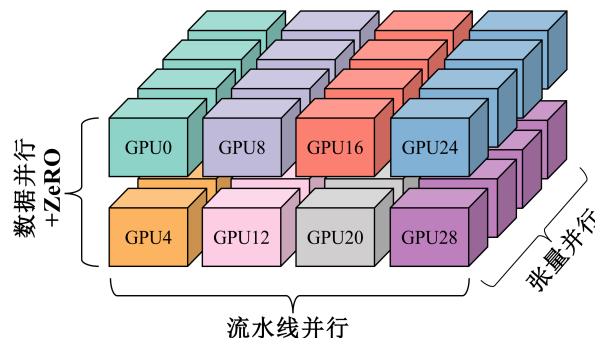


图3 混合并行训练模式，未展示混合专家并行

- **数据并行（Data Parallel, DP）与零冗余优化器（ZeRO）：**利用零冗余优化器技术^{[26][27]}，将模型参数、梯度和优化器状态细粒度地分片至不同设备。这种状态的碎片化分布，使得检查点保存从简单的“本地转储”演变为复杂的“分布式全局聚合”，极大增加了状态一致性维护的难度。
- **流水线并行（Pipeline Parallel, PP）：**将模型层切分为多个阶段跨节点部署，形成深度的计算依赖链。这种链式结构引入了严格的时序约束，除固有的“流水线气泡”^[28]外，节点间的强依赖性使得任何局部的迟滞都会导致整条流水线的拥塞。
- **张量并行（Tensor Parallel, TP）：**将单算子（如矩阵乘法）拆分至多张 GPU 协同计算^[29]。由于张量并行需要在算子内部进行高频通信，其构建了节点内设备的强耦合关系。
- **专家并行（Expert Parallel, EP）：**针对混合专家模型，将专家层动态分配给不同设备^[8]。由于词元（Token）路由的稀疏性与动态性，极易导

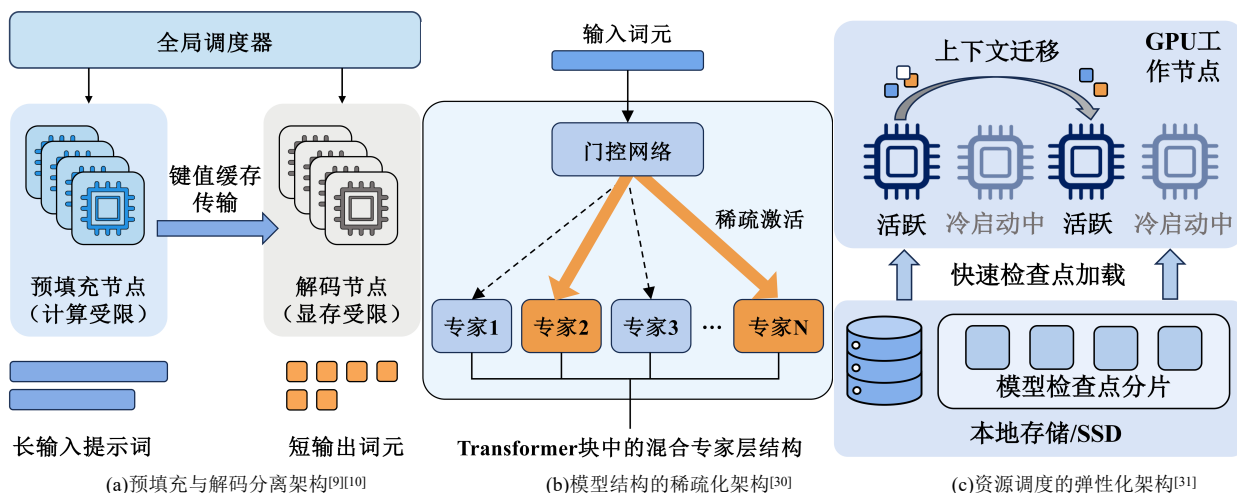


图4 LLM推理服务架构革新

致设备间负载不均,成为引发迟滞型故障的结构性质诱因(详见4.2节)。

故障放大的级联效应与容错挑战:上述多维并行策略虽然解决了状态存储与算力供给问题,但其构建的强耦合拓扑结构通常运行于批量同步并行(Bulk Synchronous Parallel, BSP)模式下,导致了严重的故障放大效应。这种紧密的依赖关系意味着任何单点的局部故障都会通过通信链路迅速扩散为全局停机。以图3为例,若GPU0发生中断或迟滞,其影响范围将呈多维辐射状:

- 在**张量并行域**,导致同节点的GPU1-3因无法协同完成算子计算而立即中断;
- 在**流水线并行域**,导致下游依赖链上的GPU8、16、24因数据断供而停滞;
- 在**数据并行域**,导致同组的GPU4等节点在全规约(All-Reduce)阶段陷入无限等待。

综上所述,LLM训练的容错设计面临双重困境:一方面,复杂的并行拓扑使得故障具有极强的传染性与破坏力;另一方面,需要在故障发生后,从分布式的碎片化存储中高效保存与恢复庞大的训练状态,这构成了容错技术必须解决的核心难题。

2.2 大语言模型的推理范式与容错挑战

LLM推理是面向用户的在线服务过程,其核心特征在于“自回归生成”的计算模式^[1]。该过程由两个阶段构成,首先是**预填充**阶段,系统并行处理输入的提示词序列,计算其查询(Q)、键(K)与值(V)矩阵。由于涉及大规模矩阵运算,该阶段表现为典型的**计算密集型**特征。随后进入**解码**阶段,系统逐个生成后续词元。在此过程中,为避免重复计算历史序列的注意力状态,系统引入了**键值**

缓存(KV Cache)机制^[6],将历史词元的 K 与 V 矩阵驻留在显存中。这一机制虽然消除了冗余计算,却将解码阶段转变为受限于显存带宽与容量的**访存密集型**任务。因此,推理系统必须同时处理“计算密集”与“访存密集”两类负载,且随着序列长度增长,动态膨胀的键值缓存成为了系统资源管理的核心瓶颈。

LLM推理服务化转型和架构革新:随着LLM推理演进为云环境下的多用户核心服务,系统设计面临着高服务质量与低部署成本的双重约束。一方面,用户对交互体验的极致追求,迫使系统必须兼顾预填充阶段的低首字延迟(Time To First Token, TTFT)与解码阶段的高吞吐量(Time Per Output Token, TPOT);另一方面,面对具有显著潮汐特征的流量,系统需利用廉价算力并具备毫秒级弹性,以实现成本效益最大化。为此,现代推理系统的后端架构发生了结构性的范式转移,如图4所示:

- **预填充和解码分离:**为解决预填充和解码负载在共置场景下的资源争抢,DistServe^[9]和Splitwise^[10]等工作提出了分离架构,将预填充和解码任务调度至不同硬件规格的节点,但这要求键值缓存必须在节点间进行实时传输。
- **模型结构的稀疏化:**为降低单次推理的算力成本,MegaScale-Infer^[30]在混合专家模型引入了稀疏激活机制。然而,流量分布的倾斜往往导致特定专家节点过载,引发结构性负载不均。
- **资源调度的弹性化:**为进一步利用廉价算力,推理任务开始向可抢占式实例(Spot Instances)以及训练和推理混合部署环境迁移,ServerlessLLM^[31]等系统实现了基于模型分片与词元级上下文迁移的无服务器架构(Serverless)。

这种架构赋予了系统毫秒级的弹性扩缩容能力，使其能够在异构资源间快速冷启动与热迁移，从而应对潮汐流量。

LLM 推理架构革新带来的容错挑战：LLM 推理架构的革新在提升效能的同时，也急剧放大了系统的脆弱性：预填充/解码分离构建了脆弱的跨节点通信依赖链，任何链路波动都会导致流水线停滞；混合专家的动态路由机制加剧了迟滞型故障的不可预测性；而引入可抢占式实例则使得节点中断从偶发异常变为常态。更关键的是，推理服务对中间状态（KV Cache）高度敏感——状态的丢失意味着需进行阻塞性的重计算，这将直接导致解码吞吐量显著下降或服务降级。如何在架构日益复杂且硬件不可靠的背景下，保障有状态推理服务的高可用性，已成为当前容错研究的重点。

3 大语言模型系统的可靠性威胁

本节聚焦于威胁 LLM 系统的三类主要故障（如表 1 所示）：中断型故障、迟滞型故障以及静默数据损坏，重点梳理了其来源分布，并总结了这些故障对训练与推理过程的差异化影响。

3.1 中断型故障

中断型故障是 LLM 系统中表现最为剧烈且频繁故障类型，通常指软硬件组件的彻底失效（如进程崩溃、节点宕机）。此类故障具有“非黑即白”的二元特征：一旦发生，相关任务即刻终止，服务完全不可用。由于 LLM 集群规模庞大且采用紧密耦合的同步通信模式，单点中断往往会引发全局性的级联停机，是影响系统可用性的首要因素。本节将深入探讨其多样的软硬件根因，并剖析其对训练状态连续性和推理服务稳定性的破坏机制。

3.1.1 中断型故障的来源和分布占比

中断型故障的根因涵盖软件与硬件两个维度，且随着深度学习框架的成熟与集群规模的扩张，其主导诱因呈现出显著的演进趋势。

早期深度学习阶段：聚焦软件故障。微软团队基于训练任务日志分析^[32]显示，由软件栈层面导致的中断型故障可系统性归纳为四类：

- **执行环境故障**（占比 48.0%），主要归因于本地与平台环境配置差异导致的依赖路径缺失；
- **通用代码错误**（占比 36.5%），涵盖了非法参数输入或空引用等常规软件工程缺陷；
- **深度学习特定故障**（占比 13.5%），主要包括显

存溢出、应用程序接口误用及张量形状不匹配；

- **数据相关故障**（占比 2.0%），涉及数据格式异常与字符编码错误等数据预处理问题。

LLM 时代：硬件故障成为首要诱因。得益于 PyTorch 等深度学习开发工具栈的日益成熟及用户开发经验的积累，软件层面错误的占比呈现显著下降趋势。然而，随着训练集群规模向万卡级演进，硬件故障已上升为制约系统稳定性的主要矛盾，在各类故障中占比已高达 55.8%^[33]，确立了其作为导致系统失效首要诱因的地位^{[33][34]}。根据故障发生部件的不同，现有研究将其系统性地划分为以下三类^[35]：

- **GPU 相关故障**（占比 45.6%），主要包括由显存位翻转引发的纠错码（Error Correction Code, ECC）错误^{[36][37]}、统一计算设备架构（Compute Unified Device Architecture, CUDA）内核执行异常导致的挂起，以及因物理链路问题导致的 GPU 或英伟达高速互联技术（NVIDIA Link, NVLink）总线脱机；
- **通信子系统故障**（占比 26.2%），涵盖了外围组件快速互联（Peripheral Component Interface Express, PCIe）链路带宽降级、网卡设备脱机或端口状态抖动，以及由光模块老化或光纤物理损伤所引发的链路传输错误；
- **主机级故障**（占比 28.1%），涉及处理器、内存与存储设备的硬件错误，以及因温度过高触发热节流或保护性宕机的电源与散热子系统异常。

综合来看，GPU 相关故障及通信子系统故障合计占比超过总故障数的一半，构成了硬件故障中的“重灾区”，揭示了大规模 LLM 任务对高性能计算单元与高带宽互连架构的极端依赖性。

3.1.2 中断型故障对训练的影响

中断型故障对 LLM 训练系统的性能构成了灾难性影响，主要表现为紧耦合架构下的全局协同中断、滞后性引发的巨额算力损耗与恢复开销，以及故障发现与诊断的“双重滞后”。具体剖析如下：

全局协同中断：由于深度学习训练普遍采用批量同步并行模式，所有计算节点须在每个同步点（如 All-Reduce 阶段）完成严格的通信同步。这种紧耦合特性致使单点硬件故障极易引发级联崩溃，从而造成巨大的资源损耗与时间浪费。例如，单个 GPU 的故障会通过英伟达集合通信库（NVIDIA Collective Communications Library, NCCL）的超时机制迅速扩散至所有参与节点，导致全局训练任务

表 1 大语言模型系统中的故障分类及其对训练与推理任务的影响

故障类别	典型成因域	具体故障示例	对训练任务的影响 (Training)	对推理任务的影响 (Inference)
中断型故障 (Fail-stop Failures)	硬件	<div>- 不可纠正的 ECC 错误</div> <div>- GPU 挂起 / 驱动崩溃</div> <div>- NVLink/PCIe 链路中断</div> <div>- 节点掉电故障</div> <div>- 光模块故障软件</div>	<div>1. 全局协同中断:</div> <div>批量同步并行模式导致所有节点停顿。</div> <div>2. 有效算力损耗和状态恢复开销大:</div> <div>需从检查点回滚, 浪费计算资源。</div> <div>3. 故障发现与诊断的“双重滞后”:</div> <div>大规模集群中故障显现时间长, 根因定位耗时长。</div>	<div>1. 服务可用性中断:</div> <div>实例宕机导致请求失败。</div> <div>2. 上下文状态丢失:</div> <div>键值缓存丢失, 需重新预填充。</div> <div>3. 级联重算与状态重建开销:</div> <div>下游实例因依赖缺失被迫重算。</div>
	软件	<div>- 显存溢出</div> <div>- 依赖库/路径错误</div> <div>- 张量形状不匹配</div> <div>- NCCL 通信超时</div>		
迟滞型故障 (Fail-slow Failures)	硬件	<div>- GPU 热节流</div> <div>- 网络设备物理异常</div> <div>- PCIe 带宽降级</div> <div>- 硬件冗余机制副作用</div>	<div>1. 木桶效应与同步屏障:</div> <div>拖慢整体迭代时间。</div> <div>2. 延迟放大的双重机制:</div> <div>依赖性气泡和队头阻塞导致正常节点等待慢节点。</div> <div>3. 隐性资源浪费与检测盲区:</div> <div>不触发日志, 难以被传统监控发现。</div>	<div>1. 共置架构中的计算阻塞:</div> <div>导致 TTFT 和 TPOT 指标恶化。</div> <div>2. 分离架构中的传输饥饿:</div> <div>预填充与解码阶段速度失配。</div> <div>3. 违反服务等级目标:</div> <div>响应时间超出服务质量承诺。</div>
	软件	<div>- 网络拥塞</div> <div>- CPU 资源争用</div> <div>- 软件运行时问题</div>		
静默数据损坏 (SDC)	硬件	<div>- 逻辑电路缺陷</div> <div>- 电压变化</div> <div>- 环境射线</div> <div>- 特定单元敏感性</div> <div>- GPU 内存比特翻转</div>	<div>1. 训练精度隐性侵蚀:</div> <div>通过集合通信污染全局模型, 导致模型发散或陷入次优解。</div> <div>2. 全局状态污染与检查点中毒:</div> <div>全局状态污染导致无法从检查点恢复。</div> <div>3. 极高的调试壁垒:</div> <div>错误隐蔽且难以复现。</div>	<div>1. 生成质量阶梯式劣化:</div> <div>产生重复词元、幻觉或事实错误。</div> <div>2. 任务类型的敏感性差异:</div> <div>生成式任务比选择题任务更脆弱。</div> <div>3. 语义漂移与上下文断层:</div> <div>键值缓存错误导致逻辑断层。</div>
	软件	<div>- 静默语义违规</div>		

被迫中断^{[33][35][38]}。Google 关于其第四代张量处理单元 (Tensor Processing Unit, TPU) TPUv4 超算集群^[39]的运行数据进一步佐证了这一点: 日均约 0.08% 的 TPU 机器、0.005% 的 NVLink 级互连链路及 0.04% 的光电路交换机 (Optical Circuit Switching, OCS) 发生可检测故障。尽管 OCS 故障较为罕见, 但因其服务于全集群通信, 单点失效将直接导致与其相连的多个 TPU 计算单元间的通信链路彻底断开, 从而严重破坏全局拓扑的连通性与集体通信性能。

有效算力损耗和状态恢复开销大: 中断型故障对训练效率的冲击主要体现在计算资源的隐性浪费与状态恢复的高昂时间成本两个维度。一方面, 故障触发的滞后性导致了严重的算力损耗。现有研究^[40]表明, 尽管约 30% 的深度学习作业在早期即失败, 但资源代价与作业规模呈显著正相关; 针对 LLM 作业的进一步分析^[41]证实了这种极高的**资源损耗不对称性**——约 40% 的早期失败作业仅消耗 10% 的 GPU 资源, 而仅占 7% 的长期运行后被取消作业却消耗了超 60% 的资源。另一方面, 故障发生后的状态恢复进一步加剧了效率折损。由于故障会

导致关键训练上下文丢失^{[7][42]}, 传统的恢复流程往往耗时数小时, 这种高昂的时间成本严重拉低了有效训练时间比 (Effective Training Time Ratio, ETTR), 成为制约大规模训练效率提升的关键瓶颈^[43]。

故障发现与诊断的“双重滞后”: 在 LLM 大规模训练场景下, 系统运维面临着故障显现周期长与根因定位难度大的双重时效性挑战。一方面, 故障普遍具有极长的潜伏期。虽然大多数语法和配置类错误在作业启动初期即可显现, 但 50% 的消息传递接口运行时故障往往在深层计算阶段才被触发, 平均需耗时 1389 分钟以上才能被系统捕获, 直接导致了故障发现前的大量无效计算^[40]。另一方面, 故障发生后的根因诊断过程已演变为系统韧性的核心瓶颈。基于 428 起生产级 LLM 训练失败案例的实证分析表明^[44], 故障平均诊断耗时高达 34.7 小时, 且 89.9% 的案例高度依赖人工审查训练日志。这主要是因为传统的日志异常检测方法 (如基于日志级别、频率或语义的方法) 在 LLM 场景下基本失效: 数据显示, 54.8% 的故障指示日志仅为信息/调试级而非错误级, 且 16.3% 的故障日志属于高频

事件，直接违背了传统检测中“低频即异常”的假设。这一现状揭示：新一代 LLM 容错系统不仅需要关注运行时的防错，更需构建适应 LLM 时空特征的自动化诊断能力，以显著缩短平均修复时间（Mean Time To Repair, MTTR）。

3.1.3 中断型故障对推理的影响

中断型故障对 LLM 推理系统性能的影响主要表现为服务中断、状态丢失以及由此引发的级联重算，具体剖析如下：

服务可用性中断：软件故障是导致推理服务不可用的直接诱因。尽管由于推理作业生命周期远短于训练，其遭遇的运行时软件故障相对较少^[41]，但配置错误（如路径配置，占比 39.4%）和数据处理异常仍是常见故障源^[32]。更值得注意的是，在追求极致资源利用率的多用户推理集群中，实例抢占（Instance Preemption）^[45]已成为一种常态化的“逻辑中断型故障”。当低优先级推理实例被高优先级任务抢占时，服务会被强制终止，这种非预期的资源回收直接威胁在线服务的服务等级协议达成率。

上下文状态丢失：中断型故障会导致驻留在显存中的键值缓存即刻失效，这从根本上破坏了 LLM 推理“自回归”生成的逻辑连续性。由于推理过程严格依赖历史时刻的注意力状态，且现有的推理系统普遍缺乏细粒度的状态检查点机制，任何单点的显存数据丢失都意味着当前的生成任务陷入“不可恢复”状态^[46]。系统无法在故障点直接恢复上下文，而必须被迫回退至初始阶段，这种这种缺乏中间状态恢复能力的脆弱性，使得推理服务在面对故障时缺乏传统无状态服务的弹性。

级联重算与状态重建开销：在新型推理架构下，故障恢复不再是单点的重启，而演变为高昂的级联式重建过程。首先，在预填充/解码分离架构中，节点间的紧密依赖将故障代价显著放大：上游节点的失效会导致下游陷入“数据饥饿”引发资源空转，而下游节点的崩溃则迫使上游重新执行计算密集的预填充，造成端到端延迟的剧烈抖动与吞吐量损失^{[9][10]}。其次，在面向可抢占实例的弹性调度场景中，频繁的实例回收若缺乏状态迁移支持，会导致长序列推理任务已累积的计算结果彻底丢失，反复的“中断-重算”循环将严重违背长上下文服务的服务等级协议。

3.2 迟滞型故障

与显性的中断型故障不同，迟滞型故障表现为组件功能尚存但性能显著低于预期，即通常所说的

“慢节点”（Straggler）或“灰故障”（Gray Failure）^{[47][48][49][50]}。此类故障具有极高的隐蔽性，不仅难以通过简单的心跳存活检测发现，且在分布式环境中极易被掩盖。然而，在 LLM 严格同步的并行计算范式下，单点迟滞会通过“木桶效应”被放大为全局性的性能瓶颈，导致巨大的算力闲置。需要指出的是，文中总结的部分迟滞型故障在实际应用中（如计算或通信迟滞）常由在线负载调整或流量突发引发的负载不均导致，传统上这常被视作单纯的性能优化或系统负载均衡问题。然而，在大语言模型训练极其严格的批量同步并行约束与在线推理的苛刻服务等级协议下，这种局部的性能退化会迅速被放大，引发级联阻塞并实质上导致全局服务的中断与失效。因此，将其纳入广义的“容错”与“系统韧性”范畴，有助于全面审视并解决大模型系统的可靠性挑战。本节将揭示其复杂的诱因（如热节流、资源争抢），并分析其对端到端训练效率和推理延迟造成的隐性拖累。

3.2.1 迟滞型故障的来源和分布占比

迟滞型故障的诱因错综复杂，在硬件层面主要集中于计算、网络与互联架构等维度。基于大规模人工智能集群的长期观测与故障特征分析，其具体来源可归纳为以下四类：

- **GPU 热节流引发的性能退化：**微软^[51]、阿里巴巴^[52]、字节跳动^[7]的实测数据均指出，GPU 热节流（即因温度过高触发的硬件级降频降压保护）是导致大规模集群性能退化的关键诱因。数据显示，虽然发生热节流的机器仅占集群总数的约 0.5%^{[7][52]}，但在所有故障设备中，此类故障占比高达 58.5%^[53]，构成了主要的性能瓶颈源。
- **网络设备物理异常：**字节跳动^[7]和腾讯^[53]均报告了由网卡及交换机端口抖动引发的训练中断或速率下降。腾讯的追踪数据显示，18.3%的迟滞型故障直接源于网卡异常^[53]。
- **PCIe 带宽降级：**腾讯报告指出，9.5%的迟滞型故障由 PCIe 链路问题导致^[53]，而 Minder^[33]的研究进一步揭示了其级联影响：单个主机的 PCIe 链路降级足以拖慢整个集群数千个 GPU 的同步性能。
- **硬件冗余机制的副作用：**微软 SuperBench^[51]指出迟滞型故障的深层根源可能潜伏于云基础设施广泛采用的硬件冗余机制中（如 GPU 计算单元冗余、显存行重映射、网络链路冗余配置等）。这些旨在提高可靠性的冗余设计，在面对具有长

期性与强同步特征的人工智能工作负载时,反而可能掩盖早期硬件缺陷,诱发难以察觉的性能退化。

在软件层面,迟滞型故障主要源于资源竞争与运行时系统的非确定性行为:

- **网络拥塞:** 在多用户共享集群中,当多个训练任务争抢网络链路时,远程直接内存访问(Remote Direct Memory Access, RDMA)网络中的拥塞通知包激增将触发反压机制。阿里巴巴的实证数据显示,集群中40%的灰故障由此引发,且平均持续时间长达24分钟^[52]。
- **CPU资源争用:** 当训练任务与高CPU负载的后台服务共置时,CPU瓶颈会显著抑制GPU流多处理器(Streaming Multiprocessor, SM)的利用率^[52]。这种争用的外因通常是任务请求与物理机器规格的错配,导致高负载节点发生严重的计算资源竞争^[54];内因则涉及工作负载分配的不均衡,如流水线阶段划分不当或微批次内序列长度差异^[55],从而引发CPU预处理的长尾延迟。
- **软件运行时问题:** 软件栈的非确定性行为(如Python垃圾回收机制在不同进程间的异步触发)这种微小的延迟在数千个进程的同步过程中被累积放大,最终造成显著的训练性能损失^[55]。

3.2.2 迟滞型故障对训练的影响

迟滞型故障对LLM训练系统的冲击主要通过“木桶效应”传导,直接引致全局算力闲置与延迟放大。更为严峻的是,相较于显性的中断型故障,此类故障具有极高的隐蔽性,往往在未被察觉的情况下持续侵蚀系统性能,造成巨大的隐性资源浪费。其具体破坏机制与量化影响剖析如下:

木桶效应与同步屏障: 迟滞型故障的破坏力取决于并行策略下的同步机制。在广泛采用的数据并行或混合并行模式中,全局梯度同步构成了严格的同步屏障。受“木桶效应”制约,单点迟滞将迫使整个集群在同步点等待最慢节点,导致所有正常节点的算力在等待中被空置,显著拉长训练迭代时间^[56]。

延迟放大的双重机制: PipeMorph^[57]量化揭示了不同并行策略在迟滞故障面前的脆弱性差异:高通信负载的数据并行模式最为敏感,在8节点GPT-2训练中,仅30ms的通信延迟即可导致迭代时间激增8.04倍;即便是相对鲁棒的流水线并行,同等条件下仍会增加2.24倍的开销。具体而言,迟

滞型故障通过以下两种机制在系统内部引发延迟放大:

- **依赖性气泡:** 在数据并行模式下,当阶段间的通信延迟超过系统的“松弛度”(Slackness)阈值时,会触发“多米诺骨牌效应”。上游的迟滞导致下游设备因数据饥饿而产生级联的GPU计算气泡。仿真数据显示^[57],通信延迟从10ms增至20ms,可导致整体延迟非线性地从400ms增至440ms,其中额外的40ms完全源于级联气泡的累积。
- **队头阻塞:** GPU内核调度的串行性进一步加剧了故障影响。缓慢的NCCL通信操作会阻塞同一CUDA流中的后续指令队列,迫使原本准备就绪的计算内核空闲等待。例如,在30ms延迟场景下,队头阻塞效应可将单次迭代时间从710ms延长至1350ms,单独造成约1.9倍的性能损耗^[57]。

隐性资源浪费与检测盲区: 与中断型故障相比,迟滞型故障具有“高频、高耗、高隐蔽”特征。腾讯集群数据^[53]显示,迟滞型故障日均发生上千次,频率比中断型故障高出3个数量级;在8192张GPU的大规模训练中,仅在单月内,其累计造成的计算时间浪费即达32.38小时(约占月度总训练时长的4.5%)。这一由性能波动导致的隐性算力损耗,其规模已比肩甚至超过了显性的中断型故障。字节跳动分析指出,迟滞型故障波及了42.5%的训练任务;且其资源损耗具有极强的长尾特征,仅前1%的严重迟滞型故障导致了总GPU浪费时间的45%^[55]。阿里巴巴的调研同样证实,迟滞型故障致使大型任务的平均完成时间延长了1.34倍^[52]。然而,此类故障的检测难度极高。特别是在混合并行场景下,慢节点会通过同步机制“拖累”全局进度,导致所有节点的GPU利用率和网络指标同步下降。这种全局性的低负载表象往往掩盖了单点的异常根因,致使传统的单机硬件指标监控(如GPU流多处理器利用率、拥塞丢包计数)失效,难以精确定位故障源^[52]。

3.2.3 迟滞型故障对推理的影响

迟滞型故障对LLM推理系统的冲击主要体现在端到端延迟的恶化与服务等级目标的违约。由于推理系统普遍采用流水线并行策略,不同架构下的故障传导机制呈现出差异化的特征,如图5所示:

共置架构中的计算阻塞: 在预填充与解码任务共置于同一GPU的架构中,迟滞型故障主要通过资源耦合引发“队头阻塞”。计算密集型的预填充

任务对算力效率波动极其敏感，一旦遭遇 GPU 节流等硬件层面的迟滞，首字延迟会急剧上升^[10]。更严重的是，这种延长的预填充任务如同“巨石”长时间抢占 GPU 流处理器，迫使同批次中轻量级的解码任务进入等待队列。这种计算资源的争抢破坏了连续批处理的调度节律，导致逻辑上的流水线停滞与资源空转，显著拉长每字生成时间^[9]。

分离架构中的传输饥饿：在预填充与解码分离部署的架构中，故障瓶颈从计算侧向网络侧转移。该架构构成严格的“生产者-消费者”模型，键值缓存需跨节点传输。若上游预填充实例因硬件老化或降频导致生产速率低于预期，或者网络侧出现微小的迟滞（如微突发拥塞、链路质量降级），下游解码实例将因缺乏数据而陷入“数据饥饿”状态^[9]。这种供需失衡会在流水线各阶段间引发级联式的气泡，导致端到端延迟随网络抖动呈现线性累积，最终致使集群有效吞吐量（Goodput）显著下降。

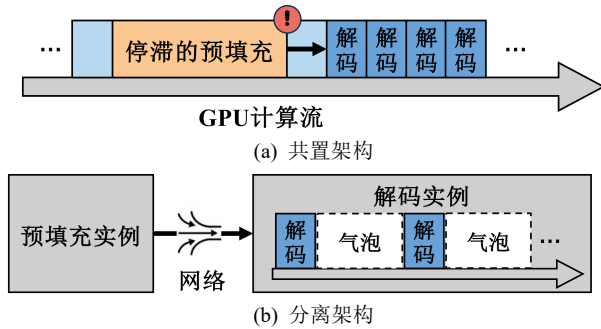


图5 迟滞型故障对推理阶段不同架构的影响

违反服务等级目标：迟滞型故障引入的性能抖动是导致尾延迟失控的主要诱因。特别是在无服务器（Serverless）或高并发场景下，存储带宽限制^{[41][54]}或 GPU 驱动的不稳定性^[31]会导致调度器错误预估节点状态（如低估状态清理时间达 623ms）^[31]。在推理请求爆发时，系统资源余量本就捉襟见肘，任何微小的硬件性能迟滞都会被放大，导致尾延迟显著上升，直接击穿服务等级目标底线^[58]。

3.3 静默数据损坏

静默数据损坏是三类故障中最为险恶且难以防范的一种。与中断型故障的即刻终止或迟滞型故障的可观测性能退化不同，静默数据损坏表现为系统在未触发任何报错信号（如异常中断、错误码）的情况下，直接输出了错误的计算结果^{[11][59][60]}。此类故障并不破坏服务的可用性或性能表象，而是悄然侵蚀计算的正确性实质。对于 LLM 而言，这种“正常的错误”具有灾难性后果：在训练阶段，微

小的数值偏差可能累积导致耗时数月的模型无法收敛；在推理阶段，则可能诱发模型产生严重的逻辑幻觉或非事实性输出。本节将追溯静默数据损坏的根源，并分析其如何在系统状态指标看似完全正常的掩护下，对模型精度和生成质量造成破坏。

3.3.1 静默数据损坏的来源和分布占比

随着纠错码机制在现代存储器中的普及，静默数据损坏的主要来源已从存储单元转移至难以被保护的处理器逻辑单元。实证数据显示，静默数据损坏已成为大规模 LLM 集群中的常态化风险：Meta^[2]报告在 Llama-3 训练期间，有 6 次大规模中断直接源于静默数据损坏；Google 的估算^[61]则显示，在支撑 Gemini Ultra 训练的跨数据中心万卡级 TPU 集群中，静默数据损坏的发生频率高达每两周甚至每周一次。在硬件层面，其核心诱因主要集中于逻辑缺陷、单元敏感性与物理老化三个维度：

- **逻辑电路的逃逸缺陷：**生产环境中的静默数据损坏通常源于潜在的微架构硬件缺陷^[60]。与易于防护的存储错误不同，算术逻辑单元或浮点运算单元中的缺陷难以通过常规纠错码发现。这些逃逸的缺陷往往仅在特定的高负载条件或特定计算模式下才会被激发，导致硬件潜在地输出错误结果。
- **特定单元的微架构敏感性：**硬件内部存在引发静默数据损坏的“高危区域”。基于英伟达深度学习加速器（NVDLA）的研究揭示了故障风险的非均匀分布^[11]：仅占总触发器数量约 9.8% 的全局控制触发器和 5.5% 的高位指数位触发器，分别贡献了高达约 68.5% 和 44.3% 的意外训练错误。这表明，控制逻辑与高位数值单元的比特翻转极易扩散，对模型参数造成破坏性影响。
- **硬件老化效应与磨损：**硬件的物理老化与长期磨损是诱发静默数据损坏的重要时间维度因素^[60]。这种机制导致静默数据损坏在时空分布上呈现显著的非均匀性：随着设备使用寿命的延长，部分节点会表现出极高的计算频率不匹配或偶发性的错误尖峰。这种由老化引起的随机性比特翻转，使得故障定位与复现变得异常困难。

在软件层面，静默数据损坏演变为“静默语义违规”（Silent Semantic Violations），即系统在未触发任何显式错误信号（如异常抛出或日志报警）的状态下，隐蔽地违反了预期的语义规范^[62]。尽管目前鲜有针对 LLM 软件栈静默数据损坏来源的直接研究，但支撑 LLM 训练与推理的底层分布式基础

设施（如 HDFS^[63]、ZooKeeper^[64]、Kafka^[65]）同样面临严峻的静默数据损坏风险^[62]，且此类风险极易向上传导至 LLM 系统。基于大规模生产环境的实证研究发现此类故障有三大关键特征^[62]：

- **高占比与高危害性**：底层系统的静默错误在生产环境故障总数中占比高达 39%，且其中 45% 被定级为“阻断级”或“严重级”。
- **深层逻辑缺陷主导**：与浅层的计算误差或代码拼写错误不同，88% 的软件静默数据损坏源于深层的特定逻辑缺陷（如分布式协议中的状态机转换错误），往往难以通过常规的单元测试覆盖。
- **管理操作诱发与核心语义破坏**：分布式系统中不可避免的管理操作（如节点扩容、版本升级）是 23% 静默数据损坏的直接诱因。同时，68% 的违规行为破坏的是系统上线初期即存在的“既有核心语义”（Old Semantics），而非新特性，这意味着系统最基础的功能基石可能随时处于风险之中。

3.3.2 静默数据损坏对训练的影响

静默数据损坏对 LLM 训练系统的威胁远超传统认知，其核心危害在于对训练精度的隐性侵蚀、对模型状态的不可逆破坏，以及由此构筑的极高调试与诊断壁垒。与显性的中断型故障（导致任务即刻终止）或迟滞型故障（导致吞吐量下降）截然不同，静默数据损坏往往在未触发任何显式错误信号（如非数 NaN 或无穷大 Inf 中断）的情况下，通过高频的迭代更新悄然累积数值误差。这种隐蔽的破坏最终会导致模型无法收敛或完全不可用，致使耗时数月的训练成果付诸东流，造成难以估量的算力与时间沉没成本。具体剖析如下：

训练精度的隐性侵蚀：静默数据损坏对模型训练的威胁在于其不可感知性与误差传播性。尽管神经网络具有一定的容错能力，但关键子集的静默数据损坏会导致灾难性后果^[66]：

- **收敛异常与发散**：故障引发的优化器梯度历史值异常（如梯度爆炸前兆），会破坏参数更新的方向性，导致模型训练轨迹不可逆地偏离最优解，损失函数震荡不收敛或收敛至局部高损失值^[11]。
- **权重漂移**：即便训练损失曲线在表面上未见异常，底层权重的数值分布可能已发生偏移。这种“静默漂移”会导致最终模型在下游微调任务中的准确率急剧下降，甚至完全失效^[60]。
- **关键位翻转的敏感性**：数据类型与层级结构决定了故障敏感度^[59]。例如，高位指数位的比特翻转

极易导致首位（Top-1）预测结果发生改变，这种微小的底层错误足以直接破坏高层语义的正确性。

全局状态污染与检查点中毒：静默数据损坏会导致分布式训练中的全局状态一致性崩溃。在数据并行或混合并行模式下，单个节点上的静默错误（如参数更新计算错误）会通过集合通信算子（All-Reduce）迅速扩散，进而污染全局模型参数^[11]。由于静默数据损坏具有时空非均匀性且无显式报错，受损的模型状态极易在系统无感知的情况下被持久化，从而导致检查点内容的实质性污染（即“检查点中毒”）^[60]。这意味着，当后续任务尝试从检查点恢复时，加载的是已损坏的参数，导致传统的回滚恢复策略失效。这迫使训练任务必须追溯到更早期的健康状态，甚至面临彻底重头开始的风险，造成巨大的算力浪费。

极高的调试壁垒：静默数据损坏因其极高的隐蔽性与潜伏期，被业界形象地称为“地狱级 Bug”^[11]。以缓慢退化模式为例，训练精度可能在故障发生数千次迭代后才开始呈现下降趋势，此时已错过了最佳的故障现场定位时机，难以回溯根因。进一步的研究发现^[60]，传统的基于算法的容错（Algorithm-Based Fault Tolerance, ABFT）机制在低精度（如 BF16）训练下存在检测盲区，且静默数据损坏在不同节点上的表现形式差异巨大（如部分节点表现为损失尖峰，而其他节点仅表现为权重漂移）。这种非确定性的表现使得基于规则的监控和传统调试手段几乎失效，极大地延长了平均修复时间。

3.3.3 静默数据损坏对推理的影响

在推理阶段，静默数据损坏的威胁核心在于其不可见性与语义侵蚀性。与显性的中断型故障或仅导致延迟指标恶化的迟滞型故障截然不同，静默数据损坏并不破坏服务的可用性表象，而是隐蔽地侵蚀生成的正确性实质——即模型在输出看似正常的响应时，其内部逻辑或语义内容已发生严重的偏离。基于现有研究，其对推理任务的具体破坏机制可归纳为以下三个维度：

生成质量的阶梯式劣化：静默数据损坏对生成内容的侵蚀呈现出从显性崩坏到隐性误导的渐进特征。现有分类体系将此类故障划分为表层的句法错误（如乱码、非目标语言、词元重复）与深层的语义错误（如关键词义反转、逻辑谬误）^{[12][67]}。实证研究^[12]表明，尽管 LLM 具备一定的鲁棒性，但

特定的静默错误仍能穿透模型防御，导致语义完全相反的输出（例如将“同情”误译为“感激”），造成严重的语义扭曲。尤为值得警惕的是硬件故障破坏力的非线性特征^[68]：即使是浮点数指数位上的单个比特翻转，也足以触发“蝴蝶效应”，导致模型推理精度从 82%骤降至 0%，造成生成质量的全面崩塌。

任务类型的敏感性差异：不同推理任务对静默数据损坏的敏感度呈现出显著的异质性。大规模测量研究表明^[67]，生成式任务相较于判别式任务（如多项选择）表现出极高的脆弱性。这是由于在自回归生成模式下，单个词元的偏差会随着时间步向后传播并被级联放大；特别是在数学推理等强逻辑链任务中，中间步骤微小的计算谬误（如符号翻转）足以导致最终答案的彻底偏离。相比之下，分类或选择题任务主要依赖最终输出层的未归一化预测值（Logits）排序，Softmax 等非线性算子能够提供一定的数值界限保护，从而在一定程度上掩蔽了底层数值的微小扰动。然而，这种屏蔽效应并非绝对，随着序列长度的增加，持续累积的数值误差仍可能突破容错阈值，导致预测失效^[69]。

语义漂移与上下文断层：静默数据损坏会导致模型内部状态发生偏移，进而引发上下文逻辑的隐性断层。针对注意力机制的错误传播路径分析揭示^[69]， Q 、 K 与 V 矩阵计算中的静默错误会直接改变注意力权重分布，迫使模型错误地聚焦于无关或错误的上下文信息。进一步的对比研究表明^[67]，相较于计算逻辑单元的单点故障，内存系统故障具有更强的破坏性与扩散性：权重矩阵中的单比特翻转会通过矩阵乘法影响整列或整行的输出，严重破坏键值缓存的数据一致性。当推理过程依赖这些受损的缓存状态进行解码时，模型会产生“语义漂移”，即生成的文本在语法上依然通顺，但逻辑上已与上下文完全脱节，这种隐蔽的逻辑断层极难通过常规的监控手段被及时发现。

4 训练阶段容错

基于第 3 章构建的故障分类，本章将系统梳理面向训练阶段的容错技术体系。首先，针对破坏性最强的**中断型故障**，剖析由细粒度检查点、资源冗余恢复、模型结构化容错及主动故障诊断构成的多层次防护体系；（4.1 节）；其次，面向隐蔽的**迟滞型故障**，介绍基于性能画像的主动检测与缓解策

略，旨在消除同步过程中的长尾延迟瓶颈（4.2 节）；最后，聚焦于最为险恶的**静默数据损坏**，论述跨层级的算法防护技术，以严格确保存储与计算过程中的数值正确性（4.3 节）。表 2 系统总结了这些技术的分类、代表性工作、资源开销与适用场景，展示了训练阶段容错机制的多样化与层次化发展态势。

4.1 针对中断型故障的容错机制

针对中断型故障，学术界与工业界发展出了多层次的容错防线：作为数据持久化基石的**检查点技术**（4.1.1 节），正从全量快照向细粒度、异步化演进；利用计算与存储资源的**冗余容错机制**（4.1.1 节），旨在挖掘集群内的资源潜力，以冗余资源换系统 I/O 开销；利用**模型结构**（4.1.3 节），深入剖析参数的时空冗余与架构稀疏性，通过“白盒”视角的状态约减与定制化容错降低资源开销；防御前置的**故障诊断与探测**（4.1.4 节），试图变被动响应为主动防御，通过提前识别异常来规避训练崩溃带来的高昂代价。

4.1.1 检查点容错

检查点技术通过周期性持久化内存状态，确保故障后能通过回滚或重放恢复至最近的一致性状态。检查点技术演进脉络清晰：该技术起源于上世纪 80 年代的理论研究^{[118][119]}，随后在高性能计算领域得到了广泛应用^[4]；2010 年后，研究转向异构与混合存储探索^{[120][121][122][123]}，利用非易失内存及异步机制优化检查点持久化开销。

进入深度学习时代，面对计算与 I/O 速度失配引发的性能瓶颈以及大规模分布式训练的随机性挑战，主要优化范式演进为：

- **通信与 I/O 掩蔽：**旨在弥合计算与存储的速度差异，如 Orpheus^[124]引入增量传输机制，VeloC^[125]与 DeepFreeze^[126]则利用多级存储层次与细粒度异步 I/O 来隐藏写入延迟；
- **自适应频率与确定性：**旨在应对训练的高成本与不确定性。如 CheckFreq^[42]通过成本模型动态调整保存频率，DETrain^[127]则通过随机数生成器的状态管控保障了训练过程的可重现性。

进入 LLM 时代，主流分布式训练框架默认的同步阻塞检查点机制面临严峻挑战。正如 3.1.2 节中所深入剖析的，海量动态状态持久化所带来的高昂 I/O 开销与算力闲置，已成为制约大规模集群训练效率的核心痛点。为此，前沿研究聚焦于“加速持久化过程与计算解耦”这一核心思路，试图通过

表 2 训练阶段容错机制分类对比

故障类别	关键技术分类	代表性工作及其主要技术特征	资源开销和恢复速度	适用场景
中断型故障 (Fail-stop)	检查点技术	异步流水线与计算解耦: Transom ^[70] , DataStates-LLM ^[71] , 基于 Ramdisk 检查点 ^[72] , Swift ^[73] , FlowCheck ^[74] , Checkflow ^[75] , Sailor ^[76]	内存缓冲与旁路硬件掩盖停顿时间; 依赖持久化介质重载, 秒级至分钟级恢复	千卡至万卡集群, 存储 I/O 带宽受限场景
		多级存储与系统级优化: Gemini ^[77] , REFT ^[78] , LightCheck ^[79] , Portus ^[80] , 异步多级检查点 ^[81] , ParaCkpt ^[82] , FastPersist ^[83] , PCcheck ^[84] , StreamCache ^[85] , PhoenixOS ^[86]	异构介质层级聚合与内核重构, 吞吐损耗微小; 本地极速重载, 秒级快速恢复	具备多级存储介质的超大规模分布式系统
		弹性抽象与拓扑重配置: Tenplex ^[87] , Universal Checkpointing ^[88] , ByteCheckpoint ^[89]	节点波动感知和流式参数编排, 零存储开销; 秒级拓扑重配置	弹性公有云或混部集群
		自适应策略与频率控制: Mimose ^[90] , BAFT ^[91] , SCAR ^[92] , Convergence-aware ^[93]	基于算法收敛性自适应调控; 分钟级恢复	长周期探索性训练任务
	基于集群运行时资源的快速容错	基于内存副本的即时恢复: Oobleck ^[94] , JIT-Checkpointing ^[95] , HSDP ^[96] , CheckFree ^[97]	复用存在的健康内存状态副本; 亚秒级至分钟级恢复	有健康副本的万卡级集群
		基于计算冗余的主动备份: Bamboo ^[98]	冗余计算填充流水线气泡; 零停机恢复	高气泡率流水线
		基于运行时状态迁移的无感扩缩容: Varuna ^[99] , Recycle ^[100]	显存网络直传免除磁盘 I/O; 近零停机	实例抢占云平台
	基于模型结构的约减与容错机制	基于时空特征的状态约减: Check-N-Run ^[101] , IncrCP ^[102] , LowDiff ^[103] , MSCH ^[104] , Inshrinkerator ^[105] , AsymCheck ^[106] , AdaCheck ^[107]	显著降低存储写入量, 仅引入局部微量压缩/差量计算; 秒级至分钟级恢复, 平滑 I/O 阻塞	存储带宽受限或对吞吐损耗敏感的环境
		基于混合专家架构的定制化容错: MoEvement ^[108] , MoC-System ^[109] , Lazarus ^[110] , FT-MoE ^[111]	稀疏快照减少保存开销; 副本机制增加内存与调度开销, 支持故障后快速恢复	稀疏激活特征混合专家模型
	故障诊断与主动探测	多源数据挖掘与故障定位: L4 ^[44] , Minder ^[33] , Aegis ^[35]	旁路日志采集与分析开销; 分钟级诊断	万卡级集群
		通信死锁分析与网络异常感知: Mycroft ^[112] , C4 ^[38]	低通信损耗; 微秒级链路隔离	网络拥塞集群
		端到端集成化自愈基础设施: Unicorn ^[113] , ByteRobust ^[43] , TPUv4 光路交换架构 ^[39]	定制控制面或专用硬件开销; 毫秒级拓扑重构	万卡自动化自愈或特定硬件集群
		环境感知与主动防御: Parcae ^[114] , 预测感知弹性调度 ^[115]	预测迁移开销, 受误报制约; 秒级拦截	弹性云混部集群
迟滞型故障 (Fail-slow)	迟滞检测与根因定位	GREYHOUND ^[52] , Holmes ^[53] , 动态自适应框架 ^[50]	低监控开销; 秒级异常节点捕捉	性能波动集群
	被动缓解与系统优化	PipeMorph ^[57] , OptiReduce ^[56]	有调度开销; 亚秒级平滑延迟	多租户集群
	主动预防与验证	SuperBench ^[51]	离线开销, 训练前阻断迟滞隐患	LLM 训练前
静默数据损坏 (SDC)	静默数据损坏表征与影响	CPU 静默数据错误 ^[116] , 加速器去黑盒化 ^[11] , 语义层面错误传播 ^[60] , 高性能环境故障注入 ^[66]	离线计算开销; 不直接参与运行时在线恢复	大规模部署前评估
	轻量级算法检测防御	ATTNChecker ^[117] , TrainCheck ^[34]	算子级软硬协同校验开销; 毫秒级拦截	长周期训练任务

系统层面的资源调度与架构革新, 在不阻断计算的前提下最大化掩盖或消除 I/O 开销。依据实现层面的不同, 现有工作可系统地划分为以下两类:

异步流水线与计算解耦: 此类方法旨在通过挖掘内存带宽、流水线气泡及网络流量等系统闲置资源, 打破 I/O 同步阻塞机制。典型方法包括: **基于内存缓冲的异步拷贝技术**利用模型参数的不可变性, 通过后台线程 (Transom^[70]、DataStates-LLM^[71])

或 Ramdisk 分段并发写入与周期性清理^[72]掩盖持久化延迟, 在 13B 至 175B 参数规模的大模型训练中, 可将检查点保存带来的训练停顿时间显著降低 90%至 95% (例如将 GPT-175B 的同步阻塞时间从约 170 秒缩减至 10 秒以内^[70])。此方法虽大幅缓解了算力闲置痛点, 但由于常态化挤占主机/设备缓存, 仍面临峰值内存溢出的风险; **基于执行间隙的调度优化技术**则挖掘流水线并行中的气泡时间

(Swift^[73]) 或通过整数线性规划精细调度 PCIe 带宽 (Checkflow^[75]), 实现了检查点与计算的完全重叠, 但也引入了高度复杂的调度逻辑与带宽依赖; **基于硬件卸载的旁路技术**进一步利用交换机端口镜像 (FlowCheck^[74]) 或弹性调度框架 (Sailor^[76]) 实现零侵入式的状态提取, 彻底解耦了训练与备份过程, 但其适用性往往受限于特定的网络硬件支持或集群拓扑的动态变化。

多级存储与系统级优化: 此类方法旨在挖掘异构存储层级与操作系统内核的性能潜力, 以突破 I/O 瓶颈。典型方法包括: **基于异构介质的层级加速技术**利用 CPU 内存或持久化内存作为中间缓冲, 结合流量调度与纠删码机制 (Gemini^[77]、REFT^[78]) 或零拷贝与 RDMA 技术 (LightCheck^[79]、Portus^[80]、异步多级检查点^[81]) 加速数据落盘, 虽大幅降低了写入延迟, 但受限于昂贵的硬件成本与容量瓶颈; **基于多路并发的带宽聚合技术**聚焦于固态存储, 通过构建多路径传输机制 (ParaCkpt^[82]) 或利用非易失性内存总线 (Non-Volatile Memory Express, NVMe) 并行队列与多路并发编排 (FastPersist^[83]、PCcheck^[84]) 最大化写入带宽, 实现了极高的 I/O 吞吐量, 但也引入了总线争抢与带宽饱和难题; **基于内核重构的透明优化技术**进一步深入操作系统底层, 通过重构页缓存机制 (StreamCache^[85]) 或引入验证式推测技术 (PhoenixOS^[86]) 实现透明检查点支持, 彻底屏蔽了应用感知, 但其部署往往依赖对内核的深度修改。

除 I/O 性能瓶颈外, LLM 检查点机制在配置与部署层面同样面临严峻挑战: 一方面, 检查点状态与集群物理拓扑的强耦合, 限制了训练任务在异构资源间的弹性伸缩与迁移能力; 另一方面, 静态固定的保存策略难以适应动态波动的系统负载与模型收敛状态, 导致资源利用率低下。为突破上述刚性约束, 提升系统的部署灵活性与全局效能, 现有研究主要聚焦于以下两类关键技术:

弹性抽象与拓扑重配置: 针对动态资源调度中“状态-拓扑”强耦合的痛点, 现有研究呈现出从运行时重排向通用格式解耦演进的趋势。Tenplex^[87]通过可并行化张量集合实现状态外置与动态重部署, 但基于特定中间件的方案伴随较高的状态迁移与通信开销。为进一步降低对特定并行策略的依赖, Universal Checkpointing^[88]引入拓扑无关的原子检查点以支持灵活转换, 虽实现了存储格式层面的彻底解耦, 却在保存与加载阶段引入了额外的张量

重切分计算成本。相比之下, ByteCheckpoint^[89]则聚焦于多框架生态割裂的现实挑战, 通过统一表示与自动对齐机制屏蔽了底层框架差异, 在支撑万卡弹性伸缩的同时打破了跨框架通用的生态壁垒。

自适应策略与频率控制: 此类方法摒弃了静态配置范式, 转向基于系统状态或算法收敛性的动态决策。在资源感知的自适应方面, Mimose^[90]针对输入数据的动态特征, 在检查点保存中, 实时规划最优的激活值重计算与保存策略, 在显存受限场景下动态平衡计算与存储开销; 而 BAFT^[91]则聚焦于流水线并行的传输开销, 挖掘调度中的气泡时隙以实现检查点的异步隐藏, 但其收益在紧凑调度 (如 1F1B) 下受限于气泡率。在算法收敛感知的频率控制方面, Qiao 等人^[92]率先在迭代收敛的机器学习任务中深入探索了部分状态丢失或容错操作带来的收敛性偏移, 并利用算法的自我纠错特性来放宽对精确检查点的依赖; 在此基础上, Li 等人^[93]进一步通过量化故障对收敛的影响动态推导最优检查点间隔, 相比传统吞吐导向策略, 该方法在故障恢复成本 (重新计算时间) 与日常 I/O 开销之间取得了更符合最终训练时效的全局最优平衡。

权衡分析: 大模型训练中的检查点技术正在经历从全局同步阻塞向多级异构异步掩蔽的彻底蜕变。面对 LLM 训练状态带来的 I/O 瓶颈, 流水线解耦、并发带宽聚合以及内核级缓存重构等前沿方案, 其核心逻辑均在于将高昂的持久化开销从常态训练的临界路径中剥离未来的技术突破将更深入地依赖于如何在不挤占系统核心显存与互联带宽的前提下, 实现 I/O 延迟的无感掩盖。

4.1.2 基于集群运行时资源的快速容错

为突破检查点机制的 I/O 瓶颈, 现有研究提出利用集群内部的冗余资源构建轻量级容错方案。此类方法的核心在于挖掘训练系统中的闲置算力、内存副本及拓扑重组潜力, 旨在通过“以计算/内存换 I/O”的策略规避昂贵的持久化开销。图 6 展示了该领域三类代表性工作的技术演进:

基于内存副本的即时恢复: 此类方法利用数据并行中天然存在的冗余内存副本, 旨在消除故障后的重规划或磁盘读取开销。Oobleck^[94]提出了一种基于预计算流水线模板的弹性架构, 故障发生时, 仅需匹配存活节点的资源规格并应用预置模板进行任务路由, 实现了无需全局重规划的亚秒级恢复, 但其有效性受限于集群异构资源组合对模板的覆盖率。JIT-Checkpointing^[95]则提出“即时”

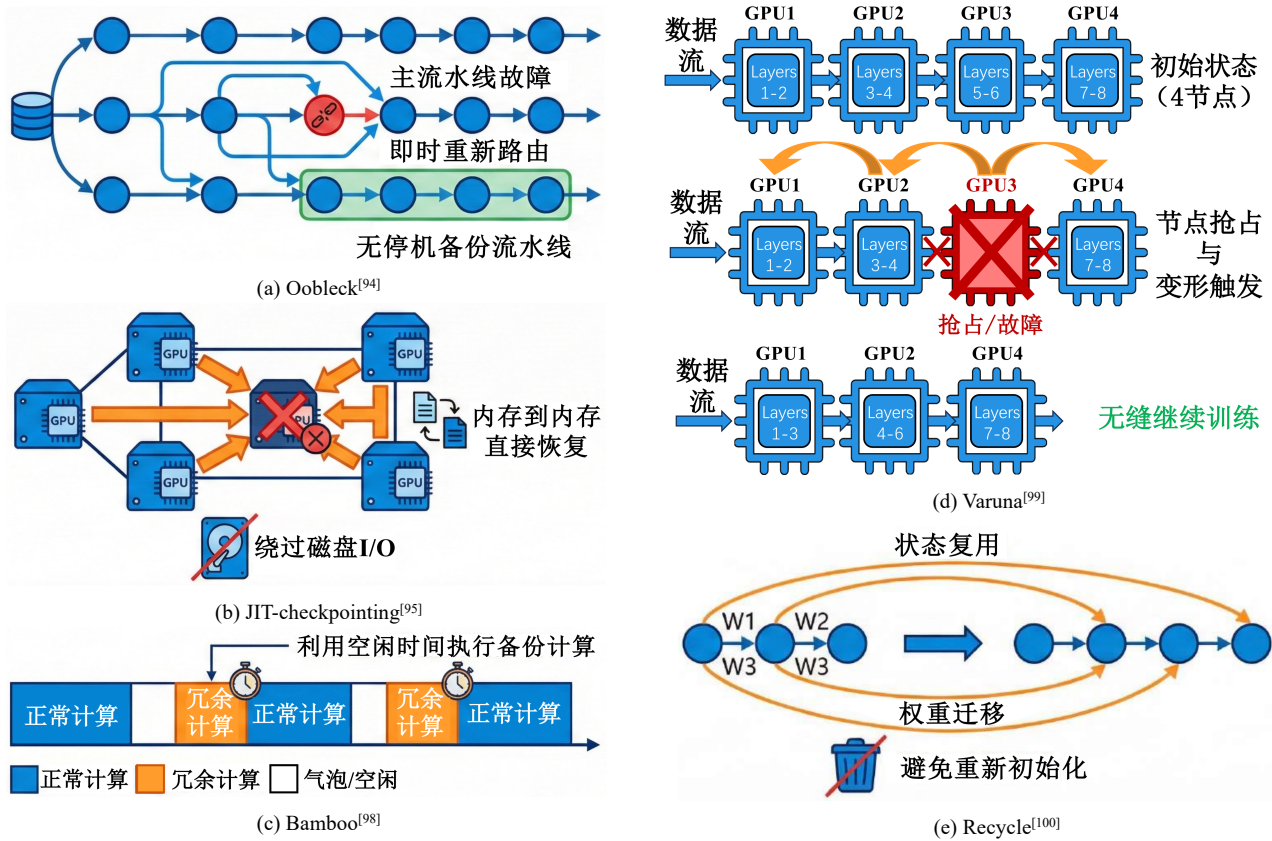
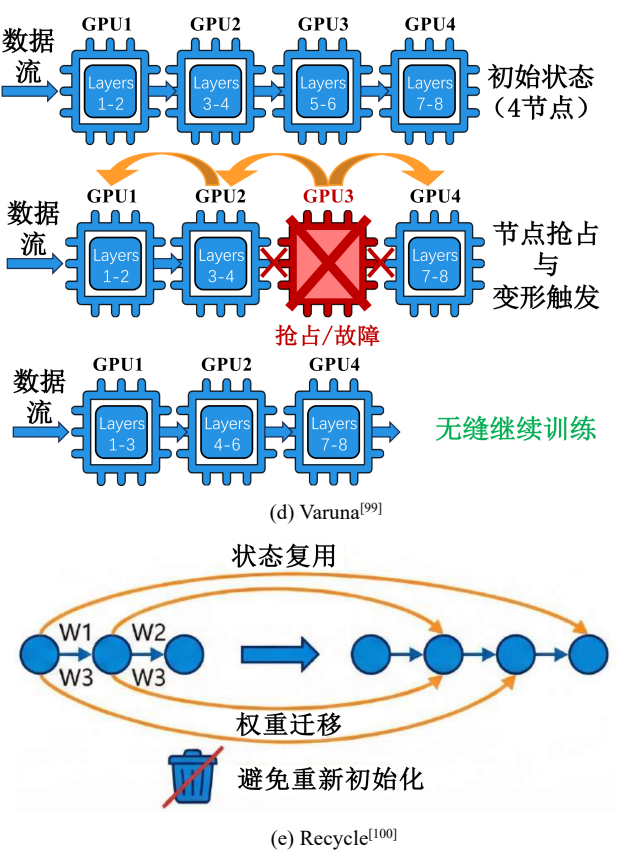


图6 基于集群运行时资源的快速容错

(Just-In-Time) 恢复范式, 其并未引入额外的计算冗余, 而是通过 RDMA 直接从邻近存活节点的内存中拉取状态副本, 通过深度复用数据并行中天然存在的健康状态来规避磁盘 I/O, 属于集群内源状态复用的高效流式容错策略。HSDP^[96]将这一理念应用于十万卡规模的混合分片数据并行架构, 通过后台进程跨分片组直接从健康的对等节点中快速拉取并重建丢失的参数分片, 将由于故障恢复的训练停顿时间从 10 分钟压缩至 3 分钟。更进一步, CheckFree^[97]打破了显式状态备份的传统约束, 在无副本可用的局部宕机场景下, 它直接利用幸存节点的梯度历史与优化器动量, 在代数层面近似重建丢失权重, 从而将常态下的容错存储与通信开销压降至零。尽管这些内存复用与状态重建机制极大提升了常态下的训练吞吐, 但它们仍深度依赖集群中局部幸存状态的完整性与可预测性, 在面对全盘崩溃或严重数值污染的关联故障时依然存在局限。

基于计算冗余的主动备份: 针对紧凑流水线中的依赖丢失问题, 此类方法转向计算冗余策略。Bamboo^[98]巧妙利用流水线并行中的气泡时隙, 主动执行冗余的微批次计算, 确保相邻节点持有必要的依赖数据。这种方法本质上是以少量的额外算力消耗换取训练的连续性 (即无需回滚), 但在低气



泡率的高填充流水线配置中, 其可利用的时隙有限, 导致收益边际递减。

基于运行时状态迁移的无感扩缩容: 与前述依赖磁盘 I/O 的检查点重配置不同, 此类方法旨在利用集群内部的高速互连网络, 通过运行时显存状态直传来规避昂贵的磁盘读写开销。Varuna^[99]提出了“作业变形” (Job Morphing) 技术, 在检测到资源波动 (如实例抢占) 时, 不经由磁盘, 而是直接通过网络在 GPU 间重新分配状态并调整流水线切分点, 实现了低成本资源的弹性利用。然而, Varuna 的“停机-迁移”模式仍存在服务中断。在此基础上, ReCycle^[100]进一步利用流水线并行中天然存在的气泡资源, 引入了细粒度的状态复用机制。它通过启发式算法识别新旧拓扑间的参数映射, 利用执行间隙直接在 GPU 间迁移权重与激活值。这种策略通过以网络带宽换存储 I/O, 有效避免了传统方法的重初始化与磁盘读写过程, 实现了近乎零停机的无感扩缩容。

权衡分析: 利用集群运行时资源进行快速容错, 开辟了绕过物理磁盘 I/O 瓶颈的新路径。这类机制的本质是以内源资源换取时间开销, 通过深度挖掘数据并行中天然存在的内存状态副本、流水线并行中的气泡时隙以及高速网络的互联潜力, 实现

亚秒级甚至无感的故障恢复。然而，其代价是即使在未发生故障的常态下，也会增加额外的任务调度负担，并挤占原本用于训练的通信带宽，系统设计需在恢复时效与常态化资源损耗之间进行精准的性能比权衡。

4.1.3 基于模型结构的约减与容错机制

除了利用冗余系统资源进行“黑盒”加速外，另一条核心路径是“打开模型白盒”，利用 LLM 自身的结构特征优化容错效率。此类方法不再将模型状态视为无差别的二进制数据块，而是深入剖析参数的时序冗余性、数值稀疏性以及模型网络架构特点。现有前沿研究主要从时空维度的状态约减与特定架构的定制化容错两个层面展开。此外，深度学习推荐模型（Deep Learning Recommendation Models, DLRM）在超大规模稀疏状态管理上的成熟经验，也为下一代 LLM 容错设计提供了极具价值的跨领域范式。

基于时空特征的状态约减：此方向旨在利用参数在训练过程中的冗余特征，在大幅降低检查点 I/O 写入量的同时维持模型精度。在时序维度，LLM 训练参数更新呈现出的稀疏性与 DLRM 场景高度契合。受推荐系统领域 Check-N-Run^[101]与 IncrCP^[102]中增量快照技术的启发，LowDiff^[103]通过仅持久化参数更新的差量，实现了低开销的增量检查点，虽显著降低了写入带宽压力，但需注意链式增量带来的故障恢复与状态重组延迟。在数值与空间维度，MSCH^[104]与 Inshrinkerator^[105]进一步结合选择性激活保存与动态量化技术削减存储需求；而 AsymCheck^[106]则深入模型内部，利用参数空间分布的非均匀特性构建非对称分区机制，从结构层面精准化解了底层文件系统的 I/O 瓶颈；同时，AdaCheck^[107]进一步延伸了空间复用的边界，通过自适应挖掘内存中的状态冗余与集群空闲带宽，实现了快照写入策略的动态自适应优化。此类方法虽有效提升了 I/O 吞吐，但复杂的自适应调度或有损压缩策略在极端数值分布下可能构成潜在的精度风险，需在压缩率、系统开销与数据完整性之间建立严格的权衡机制。

基于混合专家架构的定制化容错：混合专家架构独特的稀疏激活与专家并行策略，使得全量检查点的存储与传输开销尤为突出。在**稀疏感知减量**方面，MoEvent^[108]通过跨迭代选择性保存专家状态降低检查点开销，并将稀疏快照转换为一致的稠密检查点，结合流水线阶段边界的激活与梯度日志支

持局部恢复；MoC-System^[109]利用混合专家的稀疏激活特征减少检查点数据量。在**动态副本调度**方面，Lazarus^[110]构建了自适应的专家放置与副本管理机制，依据节点算力与网络拓扑动态编排专家分布及内存副本，利用内存热备份实现了抢占式实例的亚秒级恢复，但也引入了额外的显存占用与动态路由通信开销。

来自推荐系统的跨领域启示：尽管目前的 LLM 容错研究主要聚焦于上述约减与架构适配但 DLRM 领域为应对 TB 级嵌入式表而发展出的低开销冗余与松弛同步机制，为解决 LLM 日益严峻的内存墙与同步瓶颈提供了重要的借鉴思路。

- **基于纠删码的内存冗余：**面对大规模状态带来的内存压力，在 DLRM 场景下引入纠删码（Erasure Coding, EC）替代传统的 N 倍副本^[128]，以较低的计算开销换取内存占用的显著降低，对于未来构建低开销的 LLM 内存检查点系统具有启发意义。
- **松弛同步与部分恢复：**为最大化可用性，Ekko^[129]与 CPR^[130]提出了“部分恢复”（Partial Recovery）范式，即允许非故障节点在故障期间继续训练，通过牺牲暂时的参数一致性换取系统吞吐量。尽管直接应用于强同步的 LLM 训练存在收敛性挑战，但其蕴含的状态解耦思想对于优化超大规模并行下的容错效率仍具探索价值。

权衡分析：模型结构化容错代表了从黑盒系统级防护向白盒算法级感知的范式重构。通过解析大模型内部参数的时序冗余、数值稀疏性以及特有的网络架构，研究者得以实施更精细的状态约减与定制化容错。这种跨领域的范式融合（如借鉴 DLRM 的超大规模稀疏状态管理经验）表明，深度利用大模型自身的数学与结构特性，是打破传统通用容错机制存储与算力开销天花板的另一种方法。

4.1.4 故障诊断与主动探测

相比于侧重事后恢复的检查点与冗余机制，无显式报错的任务挂起（如死锁）常令传统超时检测失效，导致算力空耗，更难以精准归因。因此，构建基于多源指标的主动检测与诊断体系，将容错窗口前移至“故障初发期”，是强化系统韧性的关键。节依据数据模态与集成维度将现有工作归纳为四类：

多源数据挖掘与故障定位：故障定位技术正经历从单一模态依赖向多源信息融合的演进。L4^[44]、Minder^[33]与 Aegis^[35]分别利用日志周期性规律、多

维指标关联及全生命周期监控, 试解决海量日志噪声与底层硬件感知的归因难题。然而, 既有方法在感知边界与工程代价上仍存局限: 单纯依赖日志的被动分析 (L4) 在面对无显式报错的静默故障时由于信息缺失而失效受限指标颗粒度的关联诊断 (Minder) 难以触达深层逻辑错误; 而基于启发式规则库的级联流程 (Aegis) 则显著推高了系统的工程维护复杂度。作为跨领域参考, FT-MoE^[111]面向边缘环境中的故障检测与分类, 将主机资源特征和任务调度关系分别交由自适应混合专家网络与动态图注意力网络建模, 并通过交叉多头注意力融合两类信息; 结合离线训练与在线微调, 模型可随环境变化调整专家配置。该方法可为 LLM 集群的故障预测提供借鉴, 但其评估基于边缘服务场景, 在大规模 LLM 训练中的适用性仍需验证。

通信死锁分析与网络异常感知: 在应对集合通信死锁与网络拥塞时, 现有研究呈现出“端侧白盒化”与“网侧透明化”两种截然不同的技术路径。Mycroft^[112]选择深入通信库内部, 通过构建算子级全局依赖图打破了传统黑盒限制, 实现了死锁环路的精准复原, 但这种侵入式的插桩手段不可避免地引入了运行时开销与版本适配难题; 相比之下, C4^[38]探索了基于交换机的网侧卸载路径, 利用网络遥测数据实现零侵入的拥塞感知与先验干预, 虽彻底规避了主机干扰, 却因依赖特定可编程硬件而牺牲了在通用环境下的普适性。

端到端集成化自愈基础设施: 自愈架构正由工具堆叠向系统级闭环演进, 面临通用性与定制化的博弈。Unicron^[113]代表调度器中心化思路, 通过成本感知规划最优恢复路径, 虽降低了开销, 但因深度耦合底层调度器而增加了移植难度 ByteRobust^[43]针对万卡集群特征提出了“快速隔离优先”的自动化容错框架, 通过数据驱动的故障隔离与温热备份机制保障训练连续性, 但其高度定制化的控制平面与资源调度逻辑, 显著增加了系统部署的复杂度与维护门槛; Zu 等人^[39]进一步将容错下沉至硬件定义层面, 利用 TPUv4 光路交换实现物理拓扑的毫秒级重构, 虽触及可用性天花板, 却因强依赖专用硬件而难以在通用 GPU 集群复制。

环境感知与主动防御: 此类方法利用云基础设施的物理特征与故障规律实现先发防御。针对云抢占式实例, Parcae^[114]提出有效吞吐量导向的主动迁移机制, 通过预测资源可用性预先调整并行策略以最大化期望吞吐但其有效性依赖于预测的准确性,

在突发大规模抢占或预测失效时, 系统仍需退化为高开销的反应式检查点回滚机制; Jin 等人^[115]则探索基于预测性维护的弹性策略, 利用机器学习模型分析集群状态以动态调节检查点频率, 然而其收益高度依赖故障预测的召回率与准确性, “假阳性”误报引发的频繁操作反而可能拖慢整体训练进度。

权衡分析: 故障诊断与主动探测技术正推动大模型训练的容错防线由被动事后回滚向主动先验干预前移。上述机制分别从多维状态观测、底层通信解剖、系统架构自愈与环境概率预测四个维度, 试图在故障引发全局崩溃前完成精准的定位与熔断。然而, 主动防御的工程落地始终面临着严苛的权衡: 深度的异常归因往往以高昂的代码侵入性与运行时开销为代价, 极致的拓扑重构过度依赖专用的底层网络硬件, 预测性调度的净收益则因“假阳性”误报导致的无效迁移而受到严重制约。因此, 如何在干扰常态化高速互联通信的前提下, 利用旁路遥测与轻量级启发式算法构建“高置信、低侵入”的闭环自愈体系, 是下一代智能算力基础设施演进的核心命题。

4.2 针对迟滞型故障的容错机制

继 4.1 节讨论了针对中断型故障的硬核恢复策略后, 本节将视线转向更为隐蔽的迟滞型故障 (见 3.2 节), 围绕精准检测定位、运行时被动缓解及先验性主动预防三个技术维度展开深入剖析。

迟滞检测与根因定位: 迟滞检测的核心在于从动态噪声中剥离慢节点, 不同工作在检测精度与系统开销之间探索平衡。针对混合并行训练中的计算与通信迟滞, GREYHOUND^[52]通过监测迭代耗时并采用贝叶斯在线变化点检测发现异常, 随后利用轻量级剖析缩小可疑节点组范围, 再短暂挂起训练并执行局部基准测试, 以定位慢 GPU 或通信链路。该方法兼顾监控开销与定位精度, 但验证阶段仍会引入训练停顿; 面对无显式错误日志的挑战, Holmes^[53]通过构建通信算子图并结合统计学习模型将迟滞型故障定位粒度精细化至具体链路或 GPU 单元, 但受限于对通信日志的单一分析, 计算侧故障的诊断只能间接推断且依赖底层通信库的实现; Lu 等人^[50]则揭示了单一策略的失效本质 (One-Size-Fits-None), 提出了动态自适应框架和轻量级库以替代复杂的人工阈值调优, 但在处理系统启动等边缘场景时仍需人工介入。

被动缓解与系统优化: 针对无需停机的迟滞, 现有研究在资源利用率与计算精确度之间进行权

衡。计算调度上, PipeMorph^[57]通过动态调整热身步数构建级间松弛度, 利用时间缓冲吸收通信延迟, 但其抗干扰能力受限于流水线所能构建的最大松弛度阈值及显存资源约束。通信同步上, OptiReduce^[56]设计了基于块的近似全规约原语, 允许在达到超时阈值后, 仅利用部分最快节点的贡献完成当前块的聚合, 忽略极少数慢节点梯度来换取确定性的尾部延迟, 但这本质上是有损计算, 在非凸优化曲面或训练后期可能面临模型收敛精度下降的风险。

主动预防与验证:防患于未然的主动治理正成为提升效能的必选项, 其核心博弈在于离线验证成本与运行时覆盖率。针对迟滞型故障, SuperBench^[51]构建了映射真实负载的微基准测试套件, 成功在部署阶段即识别并隔离亚健康节点, 从源头降低了迟滞风险; 但这种仅针对部署时刻硬件健康状态的“准入式”验证机制, 难以捕捉训练过程中因热积累或老化导致的动态性能衰退, 仍需运行时监控互为补充。

权衡分析:迟滞型故障的容错机制已经形成“识别异常、吸收拖尾、提前筛除”的三段式治理链路: 检测定位方法从混合并行噪声中分离真正的慢节点, 被动缓解机制通过调度松弛或近似同步削弱尾部放大, 而主动验证则在部署期尽可能阻断亚健康资源进入训练闭环。三类方法共同揭示, 迟滞型故障并非单点性能下降, 而是计算、通信与调度耦合失衡后被全局同步放大的系统性问题。

4.3 静默数据损坏的感知与容错机制

相较于中断型与迟滞型故障, 隐蔽的静默数据损坏(见 3.3 节)潜在地破坏计算数值的正确性, 直接导致模型收敛失败。应对这一挑战的核心在于构建算法与软件层面的低开销校验体系, 其理论基石可追溯至 1984 年提出的基于算法的容错 (Algorithm-Based Fault Tolerance, ABFT) 范式^[131]。如图 7 所示, 经典 ABFT 通过在原始矩阵中附加校验行与校验列, 利用矩阵运算的线性特征, 在计算过程中同步完成完整校验结果的推导。现代面向大规模 LLM 的静默数据损坏容错技术实质上是 ABFT 思想在高维、分布式计算场景下的现代化重构与演进。本节将从静默数据损坏的传播机理分析与高效检测防御架构两个层面进行阐述。

静默数据损坏表征与影响:有效防御静默数据损坏的关键, 在于厘清其从底层硬件缺陷到上层模型收敛的跨层传播机理。事实上, 针对底层硬件的

隐性故障特征, 工业界与学术界早已在中央处理器 (Central Processing Unit, CPU) 领域开展了深入

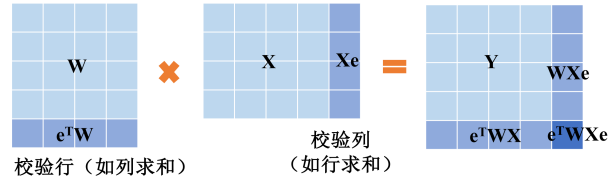


图7 基于算法容错原理图^[131]

的规模化测量。例如, Wang 等人^[116]对某超大规模云服务商生产环境中的 CPU 集群进行了长期的静默数据损坏剖析, 揭示了浮点运算与向量指令由于工艺缺陷或老化, 极易在海量并发中诱发无异常抛出的计算偏差。这种在通用计算平台上被广泛证实的“隐性计算侵蚀”现象, 为理解大语言模型训练中高频、高密度矩阵运算所面临的硬件脆弱性提供了坚实的物理基础。在此研究脉络的延伸下, 针对 GPU 的特定架构, He 等人^[11]率先完成了深度学习加速器中硬件故障的去黑盒化工作, 揭示了存储位翻转在硬件逻辑层的微观表现, 为理解静默数据损坏的物理成因奠定了基石, 但尚未触及 LLM 训练特有的敏感情境。针对这一缺口, Ma 等人^[60]将研究视角上移至应用语义层, 定量剖析了不同张量 (权重/梯度/激活) 的静默数据损坏对损失值震荡与梯度范数的非线性冲击, 深刻指出了 LLM 训练中错误屏蔽与训练发散并存的复杂特性, 确立了基于训练动态的故障判定边界。Yu 等人^[66]则在高性能计算环境下开展了大规模故障注入实证, 绘制了详尽的故障行为图谱, 系统总结了不同故障模式对收敛性的长尾影响。

轻量级算法检测防御:为解决全量校验带来的高昂开销, 现有研究致力于探索嵌入关键路径的轻量级检测机制, 并在算子定制化与框架通用性之间寻求平衡。ATTNChecker^[117]针对计算密集且数值敏感的注意力算子, 利用矩阵乘法固有的线性代数性质设计了 ABFT 方案, 以极微小的算力代价实现了对关键算子的精准防护, 但这种白盒化设计难以直接迁移至非线性算子或黑盒算子。相比之下, TrainCheck^[34]利用自动推断的不变量作为静默错误的运行时检测器, 填补了系统级防护在数据正确性保障上的空白, 尽管其在通用性与自动化上实现了质的飞跃, 但其离线推断的预分析成本及跨任务迁移的误报风险仍是潜在挑战。

权衡分析训练阶段静默数据损坏的研究正在沿着“先刻画传播机理, 再实施选择性防护”的路

线演进：故障注入与收敛分析工作厘清了不同张量 和故障模式对训练精度的影响边界，轻量级检测机

表 3 推理阶段容错机制分类对比

故障类别	关键技术分类	代表性工作及其主要技术特征	资源开销和恢复速度	适用场景
中断型故障 (Fail-stop)	抢占感知调度与上下文热迁移	SpotServe ^[45] : 利用抢占通知窗口进行主动上下文迁移与重并行化 Llumnix ^[132] : 跨节点细粒度页级键值缓存迁移与显存碎片重整	迁移时短暂挤占网络带宽, 秒级恢复 迁移时短暂挤占网络带宽, 秒级恢复	云端廉价易失性实例
	键值缓存流式备份与状态复制	DéjàVu ^[46] : 计算与状态解耦; 基于流水线的键值缓存异步流式备份 GhostServe ^[133] : 影子模式异步流水线, 前向计算与通信交叠掩盖	从持续占用网络到低开销, 毫秒级恢复	从百卡级至万卡级推理平台
	状态外部化与极速启动架构	Medusa ^[134] : 将中间状态“物化”至外部存储, 加速 Serverless 冷启动 BlitzScale ^[135] : 基于主机内存的全局参数缓存, 实现 O(1)启动 FailSafe ^[136] : 循环键值缓存放置与混合注意力机制, 支持降级容错	占用主机内存资源, 降级时引入复杂路由; 毫秒级至秒级恢复	万卡级, Serverless 推理扩容
迟滞型故障 (Fail-slow)	推理阶段资源争抢与精细化调度	Orca ^[137] : 请求级调度导致队头阻塞, 迭代级调度打破批处理原子性 vLLM ^[138] : 解决显存碎片问题, 引入操作系统分页思想管理显存 DistServe ^[9] 和 Splitwise ^[10] : 分离预填充与解码阶段, 阻断资源争抢 Sarathi-Serve ^[139] : 逻辑虚拟化, 分块预填充与背负式调度平滑延迟	跨节点键值传输开销, 毫秒级缓解	高并发、长序列大规模商业推理平台
	结构化负载不均与多路复用均衡	MegaScale-Infer ^[30] : 混合专家层构建动态路由, 消除专家热点迟滞 AlpaServe ^[58] : 多路复用, 利用模型流量的非相关性进行资源超售 Aegaeon ^[140] : 将资源池化粒度从模型级细化至词元级, 自动扩缩容	上下文切换与高并发路由调度开销, 亚秒级调节	多模型共存的云端服务平台
	异构环境下的迟滞与路径择优	ServerlessLLM ^[31] : 多级存储流水线与本地检查点, 提升本地命中率 FaaSca ^[141] : 边加载边执行策略与 RDMA 组播, 重叠计算与传输 Parallax ^[142] : 请求时动态管道选择, 主动规避慢节点	高性能本地存储、网络和动态规划算力; 毫秒至秒级恢复	异构环境下的大规模推理平台
静默数据损坏 (SDC)	容错性表征与分析框架	Ares ^[143] 和 Fidelity ^[144] : 启发式剪枝与重要性采样, 全量故障模拟 LLTFI ^[12] : 针对 Transformer, 揭示 LLM 不同于 CNN 的敏感特性 Sun 等人 ^[67] : 端到端 LLM 系统性剖析, 揭示生成质量与逻辑退化	纯离线开销, 无在线损耗, 不涉及运行时恢复	模型上线前的离线脆弱性评估与防线设计
	基于算法容错与统计校验	Ibrahim 等人 ^[145] 和 FT-CNN ^[146] : 构建校验和矩阵, 保护卷积运算 FT-Transformer ^[69] : 通过跨步式基于算法容错进行模块级容错 ReaLM ^[147] : 算法容错框架, 基于激活值进行统计采样校验	较高乘加计算开销, 阻塞前向传播, 毫秒级拦截静默数据损坏	百卡级场景, 对计算正确性要求极高等专用推理
	敏感度感知与软件级防护	FT2 ^[148] : 非均匀防护机制, 基于首词元锚点的动态关键层防护 SAVE ^[68] : 用权重与键值缓存的位级敏感度构建纯软件虚拟纠错码	微量额外计算开销低, 毫秒级拦截	万卡级大规模在线服务, 长文本推理

制则据此将 ABFT 校验或不变量检测嵌入关键算子与运行时路径。对日后研究的启示在于, 训练阶段静默数据损坏的容错不应追求对所有数值一视同仁的全量保护而应优先保护最可能诱发发散、最难以被训练自恢复掩蔽的关键状态。

5 推理阶段容错

与第 4 章探讨的训练阶段容错(侧重状态持久化)不同, 在线推理面临低延迟与高可用挑战。尽管模型权重在推理期间保持只读状态, 但为了加速自回归生成过程, 系统必须实时维护并管理随请求序列动态增长的上下文状态——键值缓存(见 2.2 节)。因此在高并发与低延迟的双重约束下, 迫使

容错机制实现对这一动态状态的高效管理与故障恢复。表 3 系统梳理了推理阶段应对各类故障的容错机制。

5.1 针对中断型故障的容错机制

推理服务常因硬件失效、网络分区或云端实例抢占而面临中断型故障。此类故障会导致驻留显存中的活跃请求键值缓存丢失。若采用传统的重启策略, 系统必须重新执行计算密集的预填充阶段以重建状态, 这将引发不可接受的延迟抖动和长尾效应。因此, 推理阶段容错的关键在于构建低开销的状态持久化与恢复通道, 力求在毫秒级至秒级内完成上下文重建, 以保障服务级目标。本节依据状态管理与恢复范式的不同, 将现有工作归纳为以下三类:

抢占感知调度与上下文热迁移：针对云原生环境下的易失性资源，核心挑战在于打破廉价算力与高可用服务之间的零和博弈。SpotServe^[45]开创了主动逃逸范式，利用云平台的抢占通知窗口提前将键值缓存迁移至安全节点，结合动态重并行化技术适配异构资源，将不可靠的廉价算力转化为可靠的服务能力，但其有效性严格受限于迁移带宽与云厂商提供的通知时间窗口长度；Llumnix^[132]虽最初旨在解决显存碎片化与负载均衡问题，但其构建的跨节点细粒度页级迁移机制，也为弹性容错提供了最底层的操作系统原语，使得推理状态的即时解耦成为可能，然而这种细粒度的内存管理也显著增加了控制平面的复杂性与元数据同步开销。

键值缓存流式备份与状态复制：为规避节点故障后昂贵的预填充重计算代价，此类方法致力于将推理状态从计算节点中剥离。DéjàVu^[46]提出了一种面向生成式 LLM 的流式容错架构，通过异步流水线将键值缓存实时复制至备用副本或远端存储，本质上是利用网络带宽置换计算延迟，在主节点失效时实现了对用户透明的切换；然而，在高并发长序列场景下，频繁的跨节点状态同步可能竞争推理主链路的通信带宽，进而影响正常服务的吞吐量。为破解这一通信争抢瓶颈，GhostServe^[133]提出了一种以“影子模式”运行的轻量级容错系统，利用底层的计算与通信空闲间隙进行调度，将键值缓存的后台备份流与前向推理的计算过程进行了深度的执行交叠与通信掩盖，在几乎不干扰主链路吞吐的前提下，为高并发在线服务提供了极低开销的节点级容错支持。

状态外部化与极速启动架构：此类工作通过物理层面的加速或逻辑层面的松弛来最小化故障恢复时间。针对无服务器（Serverless）环境的冷启动痛点，Medusa^[134]提出“状态物化”，将中间状态卸载至外部存储，极大地压缩了容器重启的准备时间，但这种静态物化策略对动态批大小和输入长度变化的适应性较弱；BlitzScale^[135]则针对大规模集群的模型加载 I/O 瓶颈，引入了基于主机内存的全局参数缓存池，以 $O(1)$ 复杂度实现了任意节点的毫秒级原地复活，但这本质上是以昂贵的 Host 内存资源为代价换取极致的启动速度；FailSafe^[136]则挑战了张量并行必须全员存活的僵化约束，通过循环键值缓存放置与混合注意力机制，允许系统在部分 GPU 故障的降级模式下继续运行，打破了张量并行

架构的脆弱性，但这种灵活性也带来了更复杂的路由逻辑与潜在的负载不均衡风险。

权衡分析：推理阶段中断型故障容错的核心目标在于消除故障后昂贵的预填充重复计算。无论是借由抢占窗口的上下文热迁移，还是通过流式复制、状态物化与降级执行优化恢复路径，其本质皆为规避冷启动开销。这些方法分别对应了故障前迁移状态、异步复制状态与状态快速重载或单点降级运行三种截然不同的状态治理路径，共同推动大模型推理容错从节点级重启迈向状态级重绑定。

5.2 针对迟滞型故障的容错机制

不同于训练阶段的全局同步等待，推理阶段的迟滞现象直接映射为用户侧的高尾部延迟。除了基础设施的随机抖动，推理架构的革新（如 2.2 节所述的解耦架构）在运行时也扮演了系统韧性防线的角色。在此需明确，虽然部分机制常被归类为“性能优化”，但其对突发瓶颈的规避能力实质上构成了容错的基础。这种架构级的“主动防御”通过减少系统对软硬件波动的敏感性，为应对复杂环境下的负载不均筑起了第一道防线。本节将重点从动态感知的角度，进一步阐述计算特征解耦、多维负载均衡及异构环境下的容错缓解技术。

推理阶段资源争抢与精细化调度：推理优化的核心在于解决预填充与解码的特征冲突。Orca^[137]率先识别出请求级调度是导致队头阻塞的根源，通过引入迭代级调度打破了批处理的原子性限制，确立了现代 LLM 推理连续批处理的基准范式在 vLLM^[138]奠定的显存管理基础上，现有研究针对两阶段干扰问题呈现出物理分离与逻辑虚拟化两条路径：DistServe^[9]与 Splitwise^[10]主张物理分离，将两阶段分别调度至高算力与高带宽节点，彻底阻断了资源争抢，实现了最优的硬件利用率，但引入了跨节点的键值缓存传输开销；相比之下，Sarathi-Serve^[139]选择了逻辑虚拟化路径，通过“分块预填充”与“背负式调度”平滑延迟，成为资源受限场景下缓解迟滞现象的有效方案。

结构化负载不均与多路复用均衡：在多模型与混合专家场景下，静态资源划分往往难以应对流量的瞬时倾斜。MegaScale-Infer^[30]针对混合专家架构固有的专家热点问题，提出了解耦专家并行架构，通过独立伸缩专家层并构建动态路由网络，打破了计算与存储的物理绑定，有效消除了单点过载迟滞。面向多模型共存的碎片化挑战，AlpaServe^[58]借鉴了云计算的统计多路复用思想，利用模型流量

的非相关性进行资源超售,显著提升了吞吐量,但其收益高度依赖对流量突发的预测准确性,难以应对多模型同时爆发的“惊群效应”。Aegaon^[140]则将资源池化粒度从模型级细化至词元级,结合精细的键值缓存同步机制实现亚秒级自动扩缩容,解决了请求碎片化导致的资源空置问题,然而这种高频的上下文切换与同步逻辑也显著推高了控制平面的调度负载。

异构环境下的迟滞与路径择优:在 Serverless 与去中心化环境中, I/O 阻塞会引发冷启动超时,而节点异构则导致执行长尾时延。ServerlessLLM^[31]致力于解决冷启动中的 I/O 阻塞,设计了多层级存储流水线与本地化检查点策略,充分挖掘闲置存储带宽以实现 TB 级参数的秒级加载,将重心从远程拉取转移至本地命中。FaaS^[141]通过流水线化组播推理,将 RDMA 模型传输与跨节点推理执行重叠,在模型装载完成前利用已接收的模型分块启动推理流水线,从而缩短突发负载下的扩容启动延迟。而在去中心化环境中, Parallax^[142]面对节点性能的剧烈波动,从被动适应转向主动规划,提出了请求时的动态管道选择机制,通过智能规避慢节点在不稳定的基础设施之上构建了确定性的低延迟服务。

权衡分析:迟滞型故障的容错机制虽分属计算解耦、负载均衡与异构路径择优三个方向,其核心皆为削弱端到端延迟对慢节点与长尾链路的依赖。计算解耦阻断了阶段间资源争抢,多路复用与动态路由缓解了多模型与混合专家架构的结构热点,动态加载与主动避让则将长尾节点隔离在关键链路之外。可见,推理迟滞绝非单纯的局部算力瓶颈,而是状态排队与硬件异构交织的系统性长尾。因此,本节的核心权衡在于:系统须以牺牲数据局部性并推高调度复杂度为代价,换取严苛服务等级协议约束下尾部延迟的绝对稳定。

5.3 针对静默数据损坏的容错机制

与前文探讨的中断型故障与迟滞型故障(侧重于服务的可用性与时效性)截然不同,静默数据损坏威胁推理阶段的生成质量(见 3.3 节)。面对此类威胁,传统的容错范式难以适用:被动式的检查点恢复机制因无法感知静默错误而失效,而主动式的硬件三模冗余(Triple Modular Redundancy, TMR)^{[145][146]}则因高昂的时空开销无法满足在线推理严格的延迟服务等级目标。因此,推理阶段的容错机制设计需要在低运行时开销与高故障覆盖率之间寻

求新的平衡。本节将从容错性表征、算法级校验及敏感度感知防护三个维度梳理技术演进。

容错性表征与分析框架:设计针对静默数据损坏的高效容错机制,首要前提是量化模型对硬件故障的敏感度并解析其传播机理。为应对深度学习庞大的状态空间,Ares^[143]与 Fidelity^[144]分别提出了针对通用 DNN 和加速器的轻量级故障注入框架,利用启发式剪枝与重要性采样技术,有效解决了全量模拟不可行的问题。对于 Transformer^[1]架构,LLTF^[12]率先利用中间表示(Intermediate Representation, IR)层面的注入实验,揭示了 LLM 具有与传统 CNN 截然不同的故障敏感特性。Sun 等人^[67]则进一步从端到端视角对大规模 LLM 进行了系统性剖析,指出传统聚合指标(如分类准确率)掩盖了生成质量的降级,并发现随机位翻转会导致严重的逻辑退化,为面向生成质量的容错设计提供了理论支撑。

基于算法容错与统计校验:此类方法继承了基于算法容错的思想,利用矩阵运算的线性特征或数据统计规律实现在线检测。在 CNN 领域,Ibrahim 等人^[145]与 FT-CNN^[146]先后设计了系统化的算法级容错方案,通过构建校验和矩阵(Checksum Matrices),在无需修改硬件的前提下有效保护了计算密集型的卷积操作,为后续研究奠定了基础;但在面对 Transformer^[1]复杂的非线性依赖时,传统 ABFT 陷入困境。FT-Transformer^[69]为此提出了模块级防护策略,将注意力机制视为整体内核进行端到端保护,并通过跨步式(Strided)基于算法容错机制消除通信开销,将 ABFT 的边界拓展至非线性领域。ReaLM^[147]则更进一步提出了统计式基于算法容错框架,摒弃全量精确校验,转而利用激活值的统计分布特性进行轻量级采样,在极低算力损耗下敏锐捕捉异常数值,实现了可靠性与效率的优化平衡。

敏感度感知与软件级防护:鉴于对 LLM 进行全量防护会带来不可接受的延迟代价,非均匀防护成为打破性能瓶颈的关键。FT2^[148]敏锐地捕捉到序列生成的“首词元效应”,即首个词元的质量决定了后续生成的逻辑轨迹,据此设计了以首词元为锚点的动态防护机制,以极小的局部代价换来全局生成质量的提升。针对显存位翻转问题,SAVE^[68]跳出了纠错码必须依赖硬件的传统观念,通过深入解构模型权重与算子激活值的位级敏感度,提出了一种利用 GPU 内少量可靠内存区域进行差异化分配

的软件容错机制，但该方案主要面向视觉与决策模型的推理，尚未包含对 LLM 键值缓存的保护。

权衡分析：推理阶段针对静默数据损坏的防护形成从容错表征、轻量级检测到差异化加固的技术主线。容错表征技术精准定位最易受损的模型层与生成阶段，算法与统计级校验借此将异常检测无缝融入关键算子，敏感度感知机制则将有限的保护预算向首词元等核心路径倾斜。推理阶段的静默错误直接引发瞬时的语义漂移与逻辑崩塌，因此，需要以微小的边际算力与延迟代价，精准构筑起保障最终生成质量的关键路径防线。

6 未来研究方向与展望

基于本文构建的 LLM 系统“工作负载-故障类型”容错机制的二维分类体系，未来面向 LLM 系统容错机制的研究热点将超越单一维度的优化，向着更深层次的软硬件协同与语义感知方向演进。

6.1 训练阶段的容错机制展望

中断型故障的容错机制：近零开销的软硬协同透明持久化。尽管当前检查点技术通过异步流水线与多级存储缓解了 I/O 阻塞，但在万卡级工业生产中，海量状态数据的跨节点搬运仍是制约系统扩展性的核心瓶颈。目前学术界研究多基于理想化的开放设施，而工业界实际落地则深陷底层系统“封闭性”（如专有集群调度、私有通信库）与高“硬件依赖”（如绑定特定芯片与特种固件）的现实鸿沟。未来研究必须打破两界壁垒，从纯软件优化转向全面解耦与深度共生并存的软硬协同容错体系：

- **基于新型互连架构的硬件级透明化：**需探索存算分离（Disaggregated Storage and Compute）架构下的非易失性内存应用，利用计算快速链接（Compute Express Link, CXL）等下一代互连技术构建全局共享的持久化内存池。核心在于以内存语义的直接访问代替传统的文件 I/O 堆栈，向应用层屏蔽底层专有硬件的物理拓扑封闭性，实现极低延迟的状态快照与原地恢复，从而彻底消除“停机-快照”模式带来的算力闲置与流水线气泡。
- **基于模型语义感知的算法级近似容错：**未来的检查点策略将不再盲目地保存全量二进制状态，而是结合模型结构感知进行数据约减，探索仅持久化关键算子状态或高频更新参数的“有损”策略，

并利用深度学习算法自身的鲁棒性（如动量修正或自适应学习率）来补偿近似恢复带来的偏差，在微量精度损失与 I/O 效率的显著提升之间，确立符合收敛性约束的帕累托最优解。

- **兼容开源软件生态与厂商独占硬件的跨层自适应适配：**对工业界生产环境的生态封闭瓶颈，容错框架需向通用标准化和自适应中介化演进。通过在运行时拦截或重构主流开源框架（如 PyTorch^[149]、DeepSpeed^[26]）的底层内存与通信原语，构建一套不依赖厂商独占应用程序接口的轻量级跨层中介。这既能在封闭式自研集群内最大化激发特定网络硬件的容错潜力，又能有效解耦对专有软硬件栈的过度绑定，打破工业落地中的软硬件垄断墙。

迟滞型故障的容错机制：大模型驱动的自治运维。面对万卡集群中的迟滞型故障，现有方法通过迭代耗时监测、在线变化点检测和局部基准测试定位慢节点^[52]，其跨负载适配与验证阶段的训练停顿仍有优化空间。未来可探索大模型驱动的自治运维方法，结合多源运行信息与领域知识，提升异常解释和处置决策能力。

- **基于多模态融合的语义感知诊断：**需突破现有统计学习方法的语义盲区，通过大模型整合日志文本、性能指标与调用链等多模态数据，从语义层面精准归因网络抖动或时钟偏移等复杂亚健康状态，解决传统方法区分噪声和故障的难题。
- **基于时序预测的主动迁移防御：**需构建基于时序分析的在线预测模型实时监测硬件的老化趋势与热积累效应；结合虚拟化热迁移技术，在节点性能劣化演变为硬中断或导致模型精度受损（避免 OptiReduce^[56]类方法的有损妥协）之前，通过无感迁移实现“先发制人”的主动防御。

静默数据损坏的容错机制：敏感度驱动的分级防御与内生鲁棒性。鉴于全量 ABFT 校验产生的算力开销在万亿参数训练中日益显著，且通用系统级检查^[34]仍存在误报风险，未来研究将致力于打破全量防护的刚性范式，转向基于参数敏感度的分级防御。

- **基于错误传播机理的细粒度敏感度量化：**应在揭示错误传播机理^[60]的基础上，构建可解释的细粒度敏感度量化模型。目标是将防护粒度从算子级下探至张量乃至比特级，精确评估特定层级或数值位对最终损失值发散的贡献度，从而实现差异化防御：在梯度范数剧烈波动的关键路径（如注

意力机制的核心计算)部署强一致性的 ABFT 逻辑,而在非敏感区采用低开销的近似计算或概率性抽样校验。

- **算法-系统协同的内生免疫纠错范式:** 需超越传统的“被动检测-回滚纠正”逻辑,探索算法-系统协同的主动纠错新范式。利用深度神经网络天然参数冗余与容错特性,设计针对特定硬件噪声模式具备内生免疫力的鲁棒算子。从算法源头解耦对底层硬件完美性的依赖,在最小化吞吐量损耗的同时,为训练收敛确立确定性的边界。

6.2 推理阶段的容错机制展望

中断型故障的容错机制:键值缓存的解耦持久化与无感漂移。随着长上下文推理成为常态,键值缓存已演变为高价值的有状态资产。尽管 SpotServe^[45]等工作探索了状态迁移,但在高并发场景下仍面临通信带宽竞争与抢占窗口期过短的瓶颈。未来的研究将致力于实现键值缓存的解耦持久化与流式迁移。

- **构建独立生命周期的全局键值存储池:** 针对键值缓存强依附于推理实例的局限,未来架构将向存算分离演进。例如利用 CXL 构建全局共享内存池,将键值缓存从依附于计算节点的临时状态升级为可全局寻址的持久化资源。通过解耦其生命周期与推理实例的绑定关系,确保在计算节点失效时,缓存状态仍能独立存活并被其他节点即时访问,从架构底层规避昂贵的预填充重计算。
- **基于内存语义的零拷贝瞬时迁移:** 为解决迁移拷贝带来的 I/O 延迟与带宽争抢,需建立基于内存语义的高速迁移通道。例如融合 RDMA 网络技术,构建零拷贝的卸载与加载机制,使 TB 级推理状态能在微秒级内通过直接内存访问迁移至健康节点,力求消除对云端抢占通知窗口的依赖,实现以 $O(1)$ 复杂度完成长文本推理任务的“断点续生”。

迟滞型故障的容错机制:异构资源的服务等级目标感知极致弹性。随着推理服务向“云-边-端”网络演进,异构资源的性能波动(迟滞)已成为常态。静态切分或简单规避策略^[142]难以在消费级异构资源池中满足严格的服务等级目标。未来研究需致力于服务等级目标感知的亚秒级弹性伸缩。

- **细粒度的异构流水线自适应调度:** 需超越词元级调度^[140]思路,在降低控制面开销的同时,针对抢占式实例及消费级硬件的性能差异,开发自适应的并行切分与流水线气泡填充策略。旨在通过

动态的负载再平衡,将异构资源的性能波动平摊至微秒级的流水线间隙中,消除“短板效应”。

- **基于请求语义的自适应降级与容错:** 针对资源受限场景,需建立推理精度与延迟的动态权衡模型。当检测到严重的性能抖动时,探索基于请求语义的优先级动态调整推理策略(如混合精度降级、投机采样步数截断或模型层级剪枝),在物理资源不可扩展的硬约束下,通过算法层的柔性适应保障端到端服务的延迟协议。

静默数据损坏的容错机制:语义感知的可验证推理。尽管现有工作^[69]已构建了数值校验防线,但在生成式应用中,底层比特翻转引发的威胁已从单纯的精度下降异化为宏观的语义崩塌。未来研究将超越数值收敛视角,聚焦于语义感知的可验证推理。

- **跨层级的实时语义完整性监视:** 需构建从微观比特翻转到宏观语义漂移的因果映射,开发实时语义监视器,利用“小模型监督大模型”的层级化架构,拦截数值校验难以捕捉的逻辑谬误与幻觉输出,建立从底层比特到上层语义的验证体系。
- **基于可信执行环境的内生安全底座:** 鉴于静默数据损坏引发的比特翻转可能破坏推理过程的控制流完整性或权重一致性,致使生成结果出现不可控的逻辑崩塌或隐私边界失效,未来的容错设计需与可信执行环境(Trusted Execution Environment, TEE)深度结合,确保推理过程中的隐私数据与安全护栏不因硬件噪声而失效,构建具备内生安全与语义确定性的生成式服务底座。

7 结论

本文以中断型故障、迟滞型故障及静默数据损坏三类故障为经,以训练和推理两类负载为纬,构建了面向大语言模型容错机制的二维分类体系首先,本文深入剖析了大规模集群面临的可靠性威胁:中断型故障导致训练状态丢失与推理服务中断;迟滞型故障通过同步屏障引发训练算力闲置与推理尾延迟恶化;而静默数据损坏则隐蔽地破坏训练收敛性与推理生成语义的正确性。在此基础上,本文系统梳理了六大核心技术域的关键容错机制:在训练阶段,涵盖了从全量检查点向内存内差分快照演进的状态保护技术,消除长尾延迟的主动检测与调度策略,以及基于算法特征分析的静默错误轻

量级防护体系。在推理阶段，聚焦于保障服务高可用的上下文热迁移与流式备份机制，应对资源干扰的计算解耦与抗干扰调度方案，以及面向生成质量保障的敏感度感知校验技术。最后，本文展望了 LLM 系统容错机制的演进趋势。

参考文献

- [1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need//Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS'17). Long Beach, USA, 2017: 5998-6008.
- [2] Dubey A, Jauhri A, Pandey A, et al. The Llama 3 herd of models. 2024: arXiv preprint arXiv: 2407.21783.
- [3] Achiam J, Adler S, Agarwal S, et al. GPT-4 technical report. 2023: arXiv preprint arXiv:2303.08774.
- [4] Egwuotoha I P, Levy D, Selic B, et al. A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems. *The Journal of Supercomputing*, 2013, 65 (3): 1302-1326.
- [5] Koo R, Toueg S. Checkpointing and recovery-rollback for distributed systems. *IEEE Transactions on Software Engineering*, 1987, 13 (1): 23-31.
- [6] Pope R, Douglas S, Chowdhery A, et al. Efficiently scaling transformer inference//Proceedings of the Sixth Conference on Machine Learning and Systems (MLSys'23). Miami, USA, 2023: 606-624.
- [7] Jiang Z, Lin H, Zhong Y, et al. MegaScale: Scaling large language model training to more than 10,000 GPUs//Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI'24). Santa Clara, USA, 2024: 745-760.
- [8] Rajbhandari S, Li C, Yao Z, et al. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale//Proceedings of the 39th International Conference on Machine Learning (ICML'22). Baltimore, USA, 2022: 18332-18346.
- [9] Zhong Y, Liu S, Chen J, et al. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving//Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI'24). Santa Clara, USA, 2024: 193-210.
- [10] Patel P, Choukse E, Zhang C, et al. Splitwise: Efficient generative LLM inference using phase splitting//Proceedings of the 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA). Buenos Aires, Argentina, 2024: 118-132.
- [11] He Y, Hutton M, Chan S, et al. Understanding and mitigating hardware failures in deep learning training systems//Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA'23). Orlando, USA, 2023: 1-16.
- [12] Agarwal U K, Chan A, Pattabiraman K. Resilience assessment of large language models under transient hardware faults//Proceedings of the 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE'23). Florence, Italy, 2023: 659-670.
- [13] Zhang C B, Li Y, Jia T. Survey of state-of-the-art fault tolerance for distributed graph processing jobs. *Journal of Software*, 2021, 32 (7): 2078-2102 (in Chinese)
(张程博, 李影, 贾统等. 面向分布式图计算作业的容错技术研究综述. *软件学报*, 2021, 32(7): 2078-2102).
- [14] Xu G Y, Zhang Y Q, Shi H Z. Review of fault-tolerant technologies for large-scale DNN training scenarios. *Journal of Frontiers of Computer Science & Technology*, 2025, 19 (7): 1771-1788 (in Chinese)
(许光远, 张亚强, 史宏志等. 面向大规模 DNN 训练场景的容错技术综述. *计算机科学与探索*, 2025, 19(7): 1771-1788).
- [15] Naveed H, Khan A U, Qiu S, et al. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 2025, 16 (5): 1-72.
- [16] Chang Y, Wang X, Wang J, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 2024, 15 (3): 1-45.
- [17] Zhao H, Chen H, Yang F, et al. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 2024, 15 (2): 1-38.
- [18] Feng Y Y, Wang Q, Shu J W. Challenges and technical development of storage systems in the era of large language models. *Big Data Research*, 2025, 11 (1): 79-91 (in Chinese)
(冯杨洋, 汪庆, 舒继武等. 大模型时代下的存储系统挑战与技术发展. *大数据*, 2025, 11(1): 79-91).
- [19] Zhou Z, Ning X, Hong K, et al. A survey on efficient inference for large language models. 2024: arXiv preprint arXiv:2404.14294.
- [20] Zhou X, He J, Zhou W, et al. A survey of LLM × data. 2025: arXiv preprint arXiv:2505.18458.
- [21] Li H, Li Y, Tian A, et al. A survey on large language model acceleration based on KV cache management. *Transactions on Machine Learning Research*, 2025.
- [22] Li B, Jiang Y, Gadepally V, et al. LLM inference serving: Survey of recent advances and opportunities//Proceedings of the 2024 IEEE High Performance Extreme Computing Conference (HPEC'24). Wakefield, USA, 2024: 1-8.
- [23] Miao X, Oliaro G, Zhang Z, et al. Towards efficient generative large language model serving: A survey from algorithms to systems. *ACM Computing Surveys*, 2025, 58 (1): 1-37.
- [24] Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models. 2020: arXiv preprint arXiv:2001.08361.
- [25] Kingma D P, Ba J. Adam: A method for stochastic optimization//Proceedings of the 3rd International Conference on Learning Representations (ICLR'15). San Diego, USA, 2015: 1-15.
- [26] Rasley J, Rajbhandari S, Ruwase O, et al. Deepspeed: System

- optimizations enable training deep learning models with over 100 billion parameters//Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (KDD'20). San Diego, USA, 2020: 3505-3506.
- [27] Rajbhandari S, Rasley J, Ruwase O, et al. ZeRO: Memory optimizations toward training trillion parameter models//Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'20). Atlanta, Georgia, 2020: 1-16.
- [28] Huang Y, Cheng Y, Bapna A, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism//Proceedings of Advances in Neural Information Processing Systems (NeurIPS'19). Vancouver, Canada, 2019: 1-10.
- [29] Shoyebi M, Patwary M, Puri R, et al. Megatron-LM: Training multi-billion parameter language models using model parallelism. 2019: arXiv preprint arXiv:1909.08053.
- [30] Zhu R, Jiang Z, Jin C, et al. MegaScale-Infer: Efficient mixture-of-experts model serving with disaggregated expert parallelism//Proceedings of the 2025 ACM Conference on SIGCOMM (SIGCOMM'25). Coimbra, Portugal, 2025: 592-608.
- [31] Fu Y, Xue L, Huang Y, et al. ServerlessLLM: Low-latency serverless inference for large language models//Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI'24). Santa Clara, USA, 2024: 135-153.
- [32] Zhang R, Xiao W, Zhang H, et al. An empirical study on program failures of deep learning jobs//Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE'20). Seoul, South Korea, 2020: 1159-1170.
- [33] Deng Y, Shi X, Jiang Z, et al. Minder: Faulty machine detection for large-scale distributed model training//Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'25). Philadelphia, USA, 2025: 505-521.
- [34] Jiang Y, Zhou Z, Xu B, et al. Training with confidence: Catching silent errors in deep learning training with automated proactive checks. Proceedings of the 19th USENIX Symposium on Operating Systems Design and Implementation (OSDI'25), 2025: 313-329.
- [35] Dong J, Qian K, Zhang P, et al. Evolution of aegis: Fault diagnosis for AI model training service in production//Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'25). Philadelphia, USA, 2025: 865-881.
- [36] Tiwari D, Gupta S, Gallano G, et al. Reliability lessons learned from GPU experience with the titan supercomputer at oak ridge leadership computing facility//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15). Austin, USA, 2015: 1-12.
- [37] Tiwari D, Gupta S, Rogers J, et al. Understanding GPU errors on large-scale HPC systems and the implications for system design and operation//Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA'15). San Francisco, USA, 2015: 331-342.
- [38] Dong J, Luo B, Zhang J, et al. Enhancing large-scale AI training efficiency: The C4 solution for real-time anomaly detection and communication optimization//Proceedings of the 2025 IEEE International Symposium on High Performance Computer Architecture (HPCA'25). Las Vegas, USA, 2025: 1246-1258.
- [39] Zu Y, Ghaffarkhah A, Dang H V, et al. Resiliency at scale: Managing Google's TPUv4 machine learning supercomputer//Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI'24). Santa Clara, USA, 2024: 761-774.
- [40] Jeon M, Venkataraman S, Phanishayee A, et al. Analysis of large-scale multi-tenant GPU clusters for DNN training workloads//Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC'19). Washington, USA, 2019: 947-960.
- [41] Hu Q, Ye Z, Wang Z, et al. Characterization of large language model development in the datacenter//Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI'24). Santa Clara, USA, 2024: 709-729.
- [42] Mohan J, Phanishayee A, Chidambaram V. CheckFreq: Frequent, fine-grained DNN checkpointing//Proceedings of the 19th USENIX Conference on File and Storage Technologies (FAST'21). Virtual event, 2021: 203-216.
- [43] Wan B, Liu G, Song Z, et al. Robust LLM training infrastructure at ByteDance//Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles (SOSP'25). Seoul, South Korea, 2025: 186-203.
- [44] Jiang Z, Huang J, Yu G, et al. L4: Diagnosing large-scale LLM training failures via automated log analysis//Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE'25). Trondheim, Norway, 2025: 51-63.
- [45] Miao X, Shi C, Duan J, et al. Spotserve: Serving generative large language models on preemptible instances//Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'24). San Diego, USA, 2024: 1112-1127.
- [46] Strati F, McAllister S, Phanishayee A, et al. DéjàVu: Kv-cache streaming for fast, fault-tolerant generative LLM serving//Proceedings of the 41st International Conference on Machine Learning (ICML'24). Beijing, China, 2024: 46745-46771.
- [47] Huang P, Guo C, Zhou L, et al. Gray failure: The Achilles' heel of cloud-scale systems//Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS'17). British Columbia, Canada, 2017: 150-155.
- [48] Gunawi H S, Suminto R O, Sears R, et al. Fail-slow at scale: Evidence of hardware performance faults in large production systems. ACM Transactions on Storage, 2018, 14 (3): 1-26.
- [49] Lu R, Xu E, Zhang Y, et al. Perseus: A Fail-Slow detection framework for cloud storage systems//Proceedings of the 21st USENIX Conference on File and Storage Technologies (FAST'23). Santa

- Clara, USA, 2023: 49-64.
- [50] Lu R, Lu Y, Jiang Y, et al. One-Size-Fits-None: Understanding and enhancing Slow-Fault tolerance in modern distributed systems//Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'25). Philadelphia, USA, 2025: 359-378.
- [51] Xiong Y, Jiang Y, Yang Z, et al. SuperBench: Improving cloud AI infrastructure reliability with proactive validation//Proceedings of the 2024 USENIX Annual Technical Conference (USENIX ATC'24). Santa Clara, USA, 2024: 835-850.
- [52] Wu T, Wang W, Yu Y, et al. GreyHound: Hunting Fail-Slows in Hybrid-Parallel training at scale//Proceedings of the 2025 USENIX Annual Technical Conference (USENIX ATC'25). Boston, USA, 2025: 731-747.
- [53] Yao Z, Hu P, Miao C, et al. Holmes: Localizing irregularities in LLM training with mega-scale GPU clusters//Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'25). Philadelphia, USA, 2025: 523-540.
- [54] Weng Q, Xiao W, Yu Y, et al. MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI'22). Renton, USA, 2022: 945-960.
- [55] Lin J, Jiang Z, Song Z, et al. Understanding stragglers in large model training using what-if analysis. Proceedings of the 19th USENIX Symposium on Operating Systems Design and Implementation (OSDI'25), 2025: 483-498.
- [56] Warraich E, Shabtai O, Manaa K, et al. OptiReduce: Resilient and Tail-Optimal AllReduce for distributed deep learning in the cloud//Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'25). Philadelphia, USA, 2025: 685-703.
- [57] Wu T, Cao L, Lu H, et al. Attack of the bubbles: Straggler-resilient pipeline parallelism for large model training//Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI'26). Renton, USA, 2026.
- [58] Li Z, Zheng L, Zhong Y, et al. AlpaServe: Statistical multiplexing with model parallelism for deep learning serving//Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI'23). Boston, USA, 2023: 663-679.
- [59] Li G, Hari S K S, Sullivan M, et al. Understanding error propagation in deep learning neural network (DNN) accelerators and applications//Proceedings of the international conference for high performance computing, networking, storage and analysis (SC'17). Denver, USA, 2017: 1-12.
- [60] Ma J J, Pei H, Lausen L, et al. Understanding silent data corruption in LLM training//Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL'25). Vienna, Austria, 2025: 20372-20394.
- [61] Team G, Anil R, Borgeaud S, et al. Gemini: a family of highly capable multimodal models. 2023: arXiv preprint arXiv:2312.11805.
- [62] Lou C, Jing Y, Huang P. Demystifying and checking silent semantic violations in large distributed systems//Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI'22). Carlsbad, USA, 2022: 91-107.
- [63] Shvachko K, Kuang H, Radia S, et al. The Hadoop distributed file system//Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST'10). Washington, USA, 2010: 1-10.
- [64] Hunt P, Konar M, Junqueira F P, et al. ZooKeeper: Wait-free coordination for internet-scale systems//Proceedings of the 2010 USENIX Annual Technical Conference (USENIX ATC'10). Boston, USA, 2010: 1-14.
- [65] Kreps J, Narkhede N, Rao J. Kafka: A distributed messaging system for log processing//Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB'11). Athens, Greece, 2011: 1-7.
- [66] Yu P, Gu J, Han H, et al. Exploring and mitigating failure behavior of large language model training workloads in HPC systems//Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'25). St. Louis, USA, 2025.
- [67] Sun Y, Coalson Z, Cheng S, et al. Demystifying the resilience of large language model inference: An end-to-end perspective//Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'25). St. Louis, USA, 2025.
- [68] Zheng W, Xu B, Gu J, et al. SAVE: Software-Implemented fault tolerance for model inference against GPU memory bit flips//Proceedings of the 2025 USENIX Annual Technical Conference (USENIX ATC'25). Boston, USA, 2025: 1585-1604.
- [69] Dai H, Wu S, Huang J, et al. FT-Transformer: Resilient and reliable transformer with end-to-end fault tolerant attention//Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'25). St. Louis, USA, 2025.
- [70] Wu B, Xia L, Li Q, et al. Transom: An efficient fault-tolerant system for training LLMs. 2023: arXiv preprint arXiv:2310.10046.
- [71] Maurya A, Underwood R, Rafique M M, et al. Datastates-LLM: Lazy asynchronous checkpointing for large language models//Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing (HPDC'24). Pisa, Italy, 2024: 227-239.
- [72] Xu W, Huang X, Meng S, et al. An efficient checkpointing system for large machine learning model training// Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'24-W). Atlanta, USA, 2024: 896-900.
- [73] Zhong Y, Sheng G, Liu J, et al. Swift: Expedited failure recovery for large-scale dnn training. IEEE Transactions on Parallel and Distributed Systems, 2024, 35 (9): 1644-1656.
- [74] Huang Z, Nie H, Jia H, et al. FlowCheck: Decoupling checkpointing

- and training of large-scale models//Proceedings of the 20th European Conference on Computer Systems (EuroSys'25). Rotterdam, The Netherlands, 2025: 1334-1349.
- [75] Liu H, Luo S, Li K, et al. Checkflow: Low-overhead checkpointing for deep learning training. *IEEE Computer Architecture Letters*, 2025.
- [76] Strati F, Zhang Z, Manos G, et al. Sailor: Automating distributed training over dynamic, heterogeneous, and geo-distributed clusters//Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles (SOSP'25). Seoul, South Korea, 2025: 204-220.
- [77] Wang Z, Jia Z, Zheng S, et al. Gemini: Fast failure recovery in distributed training with in-memory checkpoints//Proceedings of the 29th Symposium on Operating Systems Principles (SOSP'23). Koblenz, Germany, 2023: 364-381.
- [78] Wang Y, Kang X, Shi S, et al. Fault-tolerant hybrid-parallel training at scale with reliable and efficient in-memory checkpointing. 2024: arXiv preprint arXiv:2310.12670.
- [79] Chen M, Hua Y, Bai R, et al. A cost-efficient failure-tolerant scheme for distributed dnn training//Proceedings of the 2023 IEEE 41st International Conference on Computer Design (ICCD'23). Washington, USA, 2023: 150-157.
- [80] Li Y, Wu T, Li G, et al. Portus: Efficient dnn checkpointing to persistent memory with zero-copy//Proceedings of the 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS'24). New Jersey, USA, 2024: 59-70.
- [81] Maurya A, Rafique M M, Tonellot T, et al. GPU-enabled asynchronous multi-level checkpoint caching and prefetching//Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing (HPDC'23). Orlando, USA, 2023: 73-85.
- [82] Wang S, Cao Q, Zhou K, et al. Parackpt: Heterogeneous multi-path checkpointing mechanism for training deep learning models//Proceedings of the 2024 IEEE 42nd International Conference on Computer Design (ICCD'24). Milan, Italy, 2024: 183-190.
- [83] Wang G, Ruwase O, Xie B, et al. FastPersist: Accelerating model checkpointing in deep learning. 2024: arXiv preprint arXiv:2406.13768.
- [84] Strati F, Friedman M, Klimovic A. PCcheck: Persistent concurrent checkpointing for ML//Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25). Rotterdam, The Netherlands, 2025: 811-827.
- [85] Li Z, Zhang G. StreamCache: Revisiting page cache for file scanning on fast storage devices//Proceedings of the 2024 USENIX Annual Technical Conference (USENIX ATC'24). Santa Clara, USA, 2024: 1119-1134.
- [86] Wei X, Huang Z, Sun T, et al. PhoenixOS: Concurrent os-level GPU checkpoint and restore with validated speculation//Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles (SOSP'25). Seoul, South Korea, 2025: 996-1013.
- [87] Wagenländer M, Li G, Zhao B, et al. Tenplex: Dynamic parallelism for deep learning using parallelizable tensor collections//Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles (SOSP'24). Austin, USA, 2024: 195-210.
- [88] Lian X, Jacobs S A, Kurilenko L, et al. Universal Checkpointing: A flexible and efficient distributed checkpointing system for Large-Scale DNN training with reconfigurable parallelism//Proceedings of the 2025 USENIX Annual Technical Conference (USENIX ATC'25). Boston, USA, 2025: 1519-1534.
- [89] Wan B, Han M, Sheng Y, et al. ByteCheckpoint: A unified checkpointing system for large foundation model development//Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'25). Philadelphia, USA, 2025: 559-578.
- [90] Liao J, Li M, Yang H, et al. Exploiting input tensor dynamics in activation checkpointing for efficient training on GPU//Proceedings of the 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS'23). St. Petersburg, USA, 2023: 156-166.
- [91] Chen R, Lu G, Wang Y, et al. BAFT: bubble-aware fault-tolerant framework for distributed DNN training with hybrid parallelism. *Frontiers of Computer Science*, 2025, 19 (1): 191102.
- [92] Qiao A, Aragam B, Zhang B, et al. Fault tolerance in iterative-convergent machine learning//Proceedings of International Conference on Machine Learning (ICML'19). Long Beach, USA, 2019: 5220-5230.
- [93] Li H, Wang Z, Zhao H, et al. Convergence-aware optimal checkpointing for exploratory deep learning training jobs. *Future Generation Computer Systems*, 2025, 164: 107597.
- [94] Jang I, Yang Z, Zhang Z, et al. Oobleck: Resilient distributed training of large models using pipeline templates//Proceedings of the 29th Symposium on Operating Systems Principles (SOSP'23). Koblenz, Germany, 2023: 382-395.
- [95] Gupta T, Krishnan S, Kumar R, et al. Just-in-time checkpointing: Low cost error recovery from deep learning training failures//Proceedings of the 19th European Conference on Computer Systems (EuroSys'24). Athens, Greece, 2024: 1110-1125.
- [96] Salpekar O, Varma R, Yu K, et al. Training LLMs with fault tolerant HSDP on 100,000 GPUs. 2026.
- [97] Blagoev N, Ersoy O, Chen L Y. All is not lost: LLM recovery without checkpoints//Proceedings of the 6th Workshop on Machine Learning and Systems (EuroMLSys'26). Edinburgh, UK, 2026.
- [98] Thorpe J, Zhao P, Eyolfson J, et al. Bamboo: Making preemptible instances resilient for affordable training of large DNNs//Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI'23). Boston, USA, 2023: 497-513.
- [99] Athlur S, Saran N, Sivathanu M, et al. Varuna: scalable, low-cost training of massive deep learning models//Proceedings of the 17th European Conference on Computer Systems (EuroSys'22). Rennes,

- France, 2022: 472-487.
- [100] Gandhi S, Zhao M, Skiadopoulos A, et al. Recycle: Resilient training of large dnns using pipeline adaptation//Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles (SOSP'24). Austin, USA, 2024: 211-228.
- [101] Eisenman A, Matam K K, Ingram S, et al. Check-N-Run: A checkpointing system for training deep learning recommendation models//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI'22). Renton, USA, 2022: 929-943.
- [102] Lin Q, Du J, Li R, et al. IncrCP: Decomposing and orchestrating incremental checkpoints for effective recommendation model training. Proceedings of the VLDB Endowment, 2024, 18 (4): 1049-1062.
- [103] Yao C, Hu Y, Liu F, et al. LowDiff: Efficient frequent checkpointing via low-cost differential for high-performance distributed training systems//Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'25). St. Louis, USA, 2025.
- [104] Zhang C, Yu M, Yu L, et al. MSCH: Microbatch-based selective activation checkpointing with recomputation hidden for efficient training of LLM models. IEEE Access, 2024, 12: 178460-178475.
- [105] Agrawal A, Reddy S, Bhattamishra S, et al. Inshrinkerator: Compressing deep learning training checkpoints via dynamic quantization//Proceedings of the 2024 ACM Symposium on Cloud Computing (SoCC'24). Redmond, USA, 2024: 1012-1031.
- [106] Ming Z, Hu Y, Luo Z, et al. AsymCheck: Asymmetric partitioned checkpointing for efficient large language model training//Proceedings of the ACM/IEEE 63rd Design Automation Conference (DAC'26). Long Beach, USA, 2026.
- [107] Liu W, Li S, Lai Z, et al. AdaCheck: An adaptive checkpointing system for efficient LLM training with redundancy utilization//Proceedings of the 24th USENIX Conference on File and Storage Technologies (FAST'26). Santa Clara, USA, 2026.
- [108] Gandhi S, Kozyrakis C. Sparse checkpointing for fast and reliable MoE training//Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI'26). Renton, USA, 2026.
- [109] Cai W, Qin L, Huang J. MoC-System: Efficient fault tolerance for sparse mixture-of-experts model training//Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25). Rotterdam, the Netherlands, 2025: 655-671.
- [110] Wu Y, Qu W, Tao T, et al. Lazarus: Resilient and elastic training of mixture-of-experts models with adaptive expert placement. 2024: arXiv preprint arXiv:2407.04656.
- [111] Xiao W, Song W, Chen M, et al. FT-MoE: Sustainable-learning mixture of experts for fault-tolerant computing. Proceedings of the AAAI Conference on Artificial Intelligence, 2026, 40 (19): 16004-16012.
- [112] Deng Y, Zhang L, Wang Q, et al. Mycroft: Tracing dependencies in collective communication towards reliable LLM training//Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles (SOSP'25). Seoul, South Korea, 2025: 254-269.
- [113] He T, Li X, Wang Z, et al. Unicorn: Economizing self-healing llm training at scale. 2023: arXiv preprint arXiv:2401.00134.
- [114] Duan J, Song Z, Miao X, et al. Parcae: Proactive, liveput-optimized DNN training on preemptible instances//Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI'24). Santa Clara, USA, 2024: 1121-1139.
- [115] Jin Y, Yang Z, Xu X, et al. Adaptive fault tolerance mechanisms of large language models in cloud computing environments//Proceedings of the 2025 6th International Conference on Computer Engineering and Application (ICCEA'25). Jakarta, Indonesia, 2025: 754-757.
- [116] Wang S, Zhang G, Wei J, et al. Understanding silent data corruptions in a large production CPU population//Proceedings of the 29th Symposium on Operating Systems Principles (SOSP'23). Koblenz, Germany, 2023: 343-357.
- [117] Liang Y, Li X, Ren J, et al. ATTNChecker: Highly-optimized fault tolerant attention for large language model training//Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP'25). Las Vegas, USA, 2025: 252-266.
- [118] Plank J S. An overview of checkpointing in uniprocessor and distributed systems, focusing on implementation and performance. Technical Report UT-CS-97-372, Department of Computer Science, University of Tennessee, 1997.
- [119] Elnozahy E N, Alvisi L, Wang Y M, et al. A survey of rollback-recovery protocols in message-passing systems. ACM Computing Surveys, 2002, 34 (3): 375-408.
- [120] Dong X, Xie Y, Muralimanohar N, et al. Hybrid checkpointing using emerging nonvolatile memories for future exascale systems. ACM Transactions on Architecture and Code Optimization, 2011, 8 (2): 1-29.
- [121] Kannan S, Farooqui N, Gavrilovska A, et al. Heterocheckpoint: Efficient checkpointing for accelerator-based systems//Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'14). Atlanta, USA, 2014: 738-743.
- [122] Gao S, He B, Xu J. Real-time in-memory checkpointing for future hybrid memory systems//Proceedings of the 29th ACM on International Conference on Supercomputing (ICS'15). Irvine, USA, 2015: 263-272.
- [123] Ren K, Diamond T, Abadi D J, et al. Low-overhead asynchronous checkpointing in main-memory database systems//Proceedings of the 2016 International Conference on Management of Data (SIGMOD'16). San Francisco, USA, 2016: 1539-1551.

- [124] Xie P, Kim J K, Ho Q, et al. Orpheus: Efficient distributed machine learning via system and algorithm co-design//Proceedings of the ACM Symposium on Cloud Computing (SoCC'18). Carlsbad, USA, 2018: 1-13.
- [125] Nicolae B, Moody A, Gonsiorowski E, et al. Veloc: Towards high performance adaptive asynchronous checkpointing at large scale//Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS'19). Rio de Janeiro, Brazil, 2019: 911-920.
- [126] Nicolae B, Li J, Wozniak J M, et al. DeepFreeze: Towards scalable asynchronous checkpointing of deep learning models//Proceedings of the 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID'20). Melbourne, Australia, 2020: 172-181.
- [127] Xu X, Liu H, Tao G, et al. Checkpointing and deterministic training for deep learning//Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI (CAIN'22). Pittsburgh, USA, 2022: 65-76.
- [128] Zhang T, Liu K, Kosaian J, et al. Efficient fault tolerance for recommendation model training via erasure coding. Proceedings of the VLDB Endowment, 2023, 16 (11): 3137-3150.
- [129] Sima C, Fu Y, Sit M K, et al. Ekko: A Large-Scale deep learning recommender system with Low-Latency model update// 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI'22). Carlsbad, USA, 2022: 821-839.
- [130] Maeng K, Bharuka S, Gao I, et al. CPR: understanding and improving failure tolerant training for deep learning recommendation with partial recovery. Proceedings of Machine Learning and Systems, 2021, 3: 637-651.
- [131] Huang K H, Abraham J A. Algorithm-based fault tolerance for matrix operations. IEEE Transactions on Computers, 1984, 100 (6): 518-528.
- [132] Sun B, Huang Z, Zhao H, et al. Llumnix: Dynamic scheduling for large language model serving// 18th USENIX symposium on operating systems design and implementation (OSDI'24). 2024: 173-191.
- [133] Jayakody S, Zhao Y, Nehate C, et al. GhostServe: A lightweight checkpointing system in the shadow for fault-tolerant LLM serving//Proceedings of the 9th Conference on Machine Learning and Systems (MLSys'26). Bellevue, USA, 2026.
- [134] Zeng S, Xie M, Gao S, et al. Medusa: Accelerating serverless LLM inference with materialization//Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25). Rotterdam, the Netherlands, 2025: 653-668.
- [135] Zhang D, Wang H, Liu Y, et al. BlitzScale: Fast and live large model autoscaling with O(1) host caching//Proceedings of the 19th USENIX Symposium on Operating Systems Design and Implementation (OSDI'25). Boston, USA, 2025: 275-293.
- [136] Xu Z, Xie Z, Gandhi S, et al. FailSafe: High-performance Resilient Serving. arXiv preprint arXiv:2511.14116, 2025.
- [137] Yu G I, Jeong J S, Kim G W, et al. Orca: A distributed serving system for Transformer-based generative models//Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI'22). Carlsbad, USA, 2022: 521-538.
- [138] Kwon W, Li Z, Zhuang S, et al. Efficient memory management for large language model serving with PagedAttention//Proceedings of the 29th Symposium on Operating Systems Principles (SOSP'23). Koblenz, Germany, 2023: 611-626.
- [139] Agrawal A, Kedia N, Panwar A, et al. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve//Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI'24). Santa Clara, USA, 2024: 117-134.
- [140] Xiang Y, Li X, Qian K, et al. Aegaeon: Effective GPU pooling for concurrent LLM serving on the market//Proceedings of the 30th ACM Symposium on Operating Systems Principles (SOSP'25). Seoul, South Korea, 2025: 1030-1045.
- [141] Yu M, Yang R, Jia C, et al. FaaSScale: Unlocking fast LLM scaling for serverless inference//Proceedings of the 9th Conference on Machine Learning and Systems (MLSys'26). Bellevue, USA, 2026.
- [142] Tong C, Jiang Y, Chen G, et al. Parallax: Efficient LLM inference service over decentralized environment. 2025: arXiv preprint arXiv:2509.26182.
- [143] Reagen B, Gupta U, Pentecost L, et al. Ares: A framework for quantifying the resilience of deep neural networks//Proceedings of the 55th Annual Design Automation Conference (DAC'18). Shanghai, China, 2018: 1-6.
- [144] He Y, Balaprakash P, Li Y. Fidelity: Efficient resilience analysis framework for deep learning accelerators//Proceedings of the 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'20). Athens, Greece, 2020: 270-281.
- [145] Ibrahim Y, Wang H, Bai M, et al. Soft error resilience of deep residual networks for object recognition. IEEE Access, 2020, 8: 19490-19503.
- [146] Zhao K, Di S, Li S, et al. FT-CNN: Algorithm-based fault tolerance for convolutional neural networks. IEEE Transactions on Parallel and Distributed Systems, 2020, 32 (7): 1677-1689.
- [147] Xie T, Zhao J, Wan Z, et al. ReaLM: Reliable and efficient large language model inference with statistical algorithm-based fault tolerance//Proceedings of the ACM/IEEE Design Automation Conference (DAC'25). San Francisco, USA, 2025.
- [148] Sun Y, Zhu Z, Mulpuru C, et al. FT2: First-token-inspired online fault tolerance on critical layers for generative large language models//Proceedings of the 34th International Symposium on High-Performance Parallel and Distributed Computing (HPDC'25). Notre Dame, USA, 2025: 1-14.
- [149] Paszke A, Gross S, Massa F, et al. PyTorch: An imperative style, high-performance deep learning library// Advances in Neural Information Processing Systems (NeurIPS'19). Vancouver, Canada, 2019: 8024-8035.



HAN Shu-Jie Ph.D., Associate Professor. Her research interests include AI for storage systems and storage reliability.

LEE Pak-Ching Ph.D., Professor. His research interests are on improving the dependability of storage systems and computer networks, including fault tolerance, recovery, security, and performance guarantees.

LI Xiao-Lu Ph.D., Lecturer. Her research interests are on storage reliability and storage optimizations for large models.

HUANG Zhi-Jie Ph.D., Associate Professor. His research interests are on storage reliability and security, cloud storage, coding theory, and blockchains.

Background

Large Language Models (LLMs), such as Llama-3 and GPT-4, have evolved from standalone algorithms into complex distributed systems running on massive heterogeneous clusters. In hyper-scale clusters, failures have shifted from rare anomalies to operational norms. These failures manifest in three forms: fail-stop failures, causing crashes that force costly rollbacks, fail-slow failures, where degradation creates “stragglers” violating Service Level Agreements (SLAs), and Silent Data Corruption (SDC), where undetected errors compromise training convergence or inference logic without triggering alarms.

Traditional fault tolerance mechanisms are ill-suited for LLM workloads. First, the massive state scale renders coarse-grained checkpointing prohibitively expensive, necessitating fine-grained in-memory management. Second, complex computation architectures amplify the impact of local degradation, requiring performance-aware scheduling. Finally, the stealthy nature of SDCs demands shifting from system-level exception handling to algorithm-level quality assurance.

Existing surveys primarily focus on model evolution or system performance optimization, such as evaluation metrics, interpretability, and efficient inference. Within the scope of fault tolerance, prior work [14] targets general Deep Neural Network training but fails to consider LLM-specific workloads

ZHAO Nan-Nan Ph.D., Associate Professor. Her research interests are on distributed storage systems, flash memory, container technologies, and memory architectures.

ZHAO Xiao-Nan Ph.D., Associate Professor. Her research interests are on management and optimization of distributed file systems.

SONG Ling-Yun Ph.D., Associate Professor. His research interests are on visualization, cross-media cognition and reasoning, and computer vision.

ZHANG Xiao Ph.D., Professor. His research interests include storage systems, computer networks, and AI systems.

SHANG Xue-Qun Ph.D., Professor. Her research interests are on data mining, machine learning, big data, and bioinformatics.

and different failure types. Thus, current literature prioritizes efficiency while largely overlooking the system reliability threats inherent to hyperscale clusters.

To bring this gap, this survey proposes a comprehensive taxonomy of fault tolerance for LLM systems, categorizing studies by workload phase (Training and Inference) and failure types (fail-stop failures, fail-slow failures, and SDC). We systematically review fault tolerance mechanisms and outline the necessary paradigm shift for reliability in the LLM era.

This work is supported in part by the National Key Research and Development Program of China under Grant No. 2026ZD0128201, the National Natural Science Foundation of China under Grant Nos. 62502391, 62302175, 62272394, and U25B2022, and the Guangdong Basic and Applied Basic Research Foundation under Grant Nos. 2024A1515011715 and 2025A1515012995. Our research group studies reliable and efficient systems for LLM training and inference. Our recent work includes wdCP (CF’2026), which uses windowed incremental checkpointing to reduce checkpoint overhead and bound recovery cost. SemanticKV (CSCWD’2026) reduces KV cache memory usage for long-prompt inference, while HyperGen (APSys’2025) enables long-prompt inference under limited GPU memory. Within this broader research effort, the present survey organizes failure modes and fault tolerance mechanisms across training and inference, clarifies the

tradeoffs among runtime overhead, recovery cost, and output correctness, and identifies open challenges. It provides a basis

for designing and evaluating mechanisms that improve LLM system reliability with manageable resource costs.