



ADAPTIVE FAULT-TOLERANT SCHEDULING FOR THROUGHPUT AND ENERGY EFFICIENCY IN TRAINING DISTRIBUTED MIXTURE-OF-EXPERTS LARGE AI MODELS

KIRORI ZACHARY¹, JAMES MWIKYA²

¹Department of Computer Science and Informatics, Kirinyaga University, Kerugoya, Kenya

²School of Pure and Applied Sciences, Kirinyaga University, Kerugoya, Kenya

*Corresponding author: zkirori@kyu.ac.ke | Co-authors: jmwikya@kyu.ac.ke

ABSTRACT :

The rapid growth of large artificial intelligence models has led to an increased need for distributed GPU systems for model training. At the same time, failures, stragglers, network congestion, and imbalanced workloads significantly reduce the efficiency of training and increase energy consumption. These issues are particularly acute in the case of Mixture-of-Experts (MoE) due to dynamic routing which causes uneven workloads and large amounts of communication between GPUs. This desktop review aims to assess how adaptive fault tolerant scheduling can optimize throughput and energy in the context of distributed MoE training. The literature search was focused on articles published in 2021-2026, including peer-reviewed manuscripts, conferences, and technical reports. Overall, the reviewed literature indicates that adaptive scheduling, dynamic recovery, and energy-aware resource allocation can make large-scale AI training more reliable and efficient. However, most of the aforementioned literature discusses specific aspects of the problem. Therefore, this desktop review concludes that more research is needed in order to develop scheduling algorithms which simultaneously address workload variations, failures, throughput, and energy efficiency.

Keywords: Distributed training, Mixture-of-Experts, adaptive scheduling, fault tolerance, training throughput, energy efficiency.

1. Introduction

1.1 Background

Artificial intelligence (AI) models have increased significantly in terms of size and complexity in the last couple of years. Consequently, training such massive models can no longer be supported by a single computer or even a handful of graphics processing units (GPUs). Instead, training such AI models requires utilizing large-scale distributed systems that comprise hundreds or even thousands of GPUs. These systems allow distributing the training process across multiple machines; however, distributed training involves formidable communication, synchronization, resource allocation, and fault tolerance challenges (Narayanan et al., 2021).

Particularly in the context of distributed training, mixture-of-experts (MoEs) have emerged as a critical technique allowing to reduce the computational burden while retaining the model's capacity. MoEs utilize a routing mechanism that allows to activate only a subset of parameters as opposed to updating all parameters during the training process (Fedus et al., 2022). Such an approach allows to construct models with extremely large numbers of parameters while keeping computational costs at an acceptable level. Hence, it is not surprising that MoE-based approaches have become widespread in the field of large language models (LLMs).

However, while MoEs offer significant benefits in terms of computational costs, they also pose additional challenges in terms of distribution. First, because only a subset of experts is activated for a given input, the computational load per GPU can vary drastically. Second, the tokens processed by different GPUs may be different, which requires extra communication steps to redistribute the tokens among GPUs in order to achieve better utilization. This can lead to underutilization of GPUs, especially when the number of tokens processed by different GPUs is significantly different, which undermines the benefits of sparsity in terms of computational costs (Hwang et al., 2022). Additionally, these challenges become especially evident in terms of failure recovery. Due to heterogeneity and complexity, long-running distributed jobs experience delays and failures. However, recovery from such failures is expensive and time-consuming in terms of both computation and communication costs (Gupta et al., 2024).

The above challenges can no longer be addressed within the traditional paradigm of distributed computing that relies on static scheduling of tasks. Therefore, there is an increasing body of work that tries to address these challenges from a paradigm of adaptive scheduling, which aims to allocate resources and adjust execution based on the actual needs of the distributed jobs. Notably, recently proposed distributed training frameworks such as Tutel (Hwang et al., 2022) and Pollux (Qiao et al., 2021) address these challenges by allowing adaptation within and across training steps. Furthermore, techniques such as Oobleck (Jang et al., 2023) and Just-In-Time Checkpointing (Gupta et al., 2024) address the challenges of distributed training in regards to fault tolerance.

Moreover, the increasing scale of distributed training brings additional challenges in terms of energy costs. GPUs require a lot of power to operate, especially in large-scale distributed training jobs. Therefore, simply relying on conventional wisdom about scheduling in distributed systems that focuses on maximizing CPU utilization can lead to significant waste in terms of energy consumed per unit of computation. However, as described by

Zeus (You et al., 2023) and Perseus (Chung et al., 2024), there are ways to optimize energy costs while keeping the training time within reasonable limits.

Thus, it is evident that scheduling is a critical component of any large-scale distributed training system. There is a wide range of optimization techniques that are typically considered at a higher level than individual GPU scheduling, such as optimizing communication costs and placement of tensors. However, a single unified scheduler has to consider all aspects of the training process in order to maximize the overall training throughput. Furthermore, such a scheduler has to be aware of the current state of the system, including various failures and delays, and be able to react accordingly. This motivates the current research into adaptive fault-tolerant scheduling for distributed MoE-based LLM training.

1.2 Problem Statement

Ideally, a distributed MoE training system should keep its GPUs well utilized, distribute expert workloads efficiently, communicate with minimal delay and recover quickly when failures occur. At the same time, it should accomplish the training task without consuming unnecessary energy. Achieving all of these objectives simultaneously is difficult in large and dynamic computing environments.

One important source of difficulty is the changing nature of MoE workloads. Because tokens are routed dynamically to different experts, the workload experienced by individual GPUs can change from one training iteration to another. Some experts may become overloaded while others receive relatively little work. This imbalance can leave some GPUs waiting for others to finish and consequently reduce overall training throughput (Hwang et al., 2022; Shi et al., 2024).

Hardware and network problems introduce another layer of uncertainty. When a GPU fails or a worker becomes significantly slower than the others, the effects can extend beyond that individual machine. Other workers may remain idle while waiting for synchronization, and previously completed computation may need to be repeated. Conventional fault-tolerance mechanisms can reduce data loss, but checkpointing and recovery themselves introduce storage, communication and computation overheads (Gupta et al., 2024).

Static scheduling makes these challenges more difficult because it assumes that workload and resource conditions remain relatively stable. In reality, large distributed training environments are rarely static. GPU performance can vary, network conditions can change, workloads can become imbalanced and failures can occur unexpectedly. Systems such as Pollux, Tutel and ScheMoE demonstrate that adapting scheduling decisions to current system conditions can improve training efficiency (Qiao et al., 2021; Hwang et al., 2022; Shi et al., 2024).

Energy consumption also needs to be considered. A scheduling decision that improves throughput may require additional GPUs or higher power levels, thereby increasing electricity consumption. Conversely, reducing the number of active resources may save energy but extend training time. Zeus and Perseus demonstrate that these trade-offs can be managed through energy-aware optimization (You et al., 2023; Chung et al., 2024).

The main problem, therefore, is not simply that distributed MoE systems experience failures or consume large amounts of energy. Rather, the challenge lies in making *timely and intelligent scheduling decisions when workload, resource, communication, failure and energy conditions are continuously changing*.

Although previous studies have made important contributions to adaptive MoE execution, distributed scheduling, fault recovery and energy-aware training, these areas have largely been investigated independently. There remains limited evidence on an integrated scheduling approach that can respond simultaneously to MoE workload imbalance, failures, stragglers and energy constraints while maintaining high training throughput. This gap provides the basis for the present review.

1.3 Objectives of the Review

General Objective

The general objective of this review is to assess the impact of adaptive fault tolerant scheduling on the training throughput and energy efficiency of distributed Mixture-of-Experts model training.

Specific Objectives

The review will achieve the following specific objectives:

1. Determine the impact of adaptive scheduling mechanisms on the training throughput of distributed MoE models
2. Establish how fault tolerant scheduling can reduce wastage of computation and enhance energy efficiency
3. Ascertain the research gaps in the intersection of adaptive scheduling fault tolerance, and energy efficient training

1.4 Review Questions

The following review questions will guide this study:

1. What is the impact of adaptive scheduling on the training throughput of distributed MoE models?
2. How can fault tolerant scheduling enhance energy efficiency while still maintaining desirable training performance?
3. What are the research gaps in combining adaptive scheduling, fault tolerance and energy efficient techniques to train MoE models?

1.5 Scope and Justification

This review will focus on distributed training of MoE and large AI models in general, and address scheduling, fault tolerance, straggler mitigation, communication optimization, and energy efficiency. The scope of this study is limited to the period 2021 – 2026, with some relevant works published before 2021 included as contextual background information.

The review will consider peer-reviewed journal articles, leading conferences, technical reports, and selected preprints. This review will focus on training rather than inference since the two settings require drastically different approaches to scheduling and resource management.

A desktop review is appropriate for this study since the research question requires information from different specialism within computer science that

have emerged and evolved independently. By integrating the different area, desktop review will help reveal relationships and research gaps that would have been impossible to identify within the confines of one field of study.

2. Methodology

2.1 Review Design

This study uses a desktop narrative review guided by systematic search principles to collate and interpret disparate results across different classes of distributed AI research with relevance to the research questions explored.

A narrative approach is applied because the studies retrieved do not report equivalent models, infrastructure configurations, workloads, or performance metrics, which would allow synthesis of quantitative findings in a purely aggregative analysis.

2.2 Data Sources

The following scholarly and technical databases were interrogated to identify relevant literature: Scopus, Web of Science, IEEE Xplore, ACM Digital Library, Google Scholar, USENIX, Journal of Machine Learning Research, Proceedings of Machine Learning and Systems, arXiv, and institutional repositories.

2.3 Search Strategy

The search strategy comprised combinations of the following terms:

- Distributed MoE: “distributed training” AND “mixture of experts”
- Adaptive scheduling: “MoE” AND “adaptive scheduling”
- Fault tolerance: “distributed training” AND “fault tolerance”
- Stragglers: “large model training” AND stragglers
- Energy: “AI training” AND “energy efficiency”
- Energy scheduling: “GPU scheduling” AND “energy-aware”
- MoE failures: “mixture of experts” AND failure AND scheduling
- Communication: “MoE” AND “all-to-all communication”
- Recovery: “LLM training” AND checkpointing AND recovery

2.4 Inclusion and Exclusion Criteria

All studies that address distributed AI training were included if they make contributions pertinent to scheduling, resource allocation, MoE execution, fault tolerance, recovery, throughput, or energy efficiency. Conversely, publications were excluded if they addressed single-GPU training, unrelated inference workloads, or tangential topics.

2.5 Screening and Selection

For each publication, the following details were extracted: problem addressed, proposed system or technique, experimental infrastructure, scheduling or fault-tolerance mechanism, measured performance, and concluding observations. The evidence was thematically synthesized to facilitate a comprehensive understanding of the research landscape, rather than reporting individual contributions in isolation.

2.6 Data Extraction and Analysis

The following data were extracted and thematically analyzed: author(s) and year of publication, model/workload, infrastructure, scheduling mechanism, fault model, communication mechanism, throughput, energy, key findings, and limitations.

3. Review Findings

3.1 The Nature of Distributed MoE Training

The main attraction of MoE models in general, and Switch Transformer and other sparse experts in particular, is that they allow for a large model with many parameters while keeping the number of parameters that need to be actively used per request low.

The problem is that while the computation used is lower than what such a model would require otherwise, this does not necessarily translate into lower execution time, due to factors specific to distributed execution. Specifically, once tokens are dispatched to their respective experts on different GPUs, data has to be passed between those GPUs, adding overhead, and the number of tokens processed per GPU can vary greatly, with corresponding imbalances in workload.

This results in an interesting paradox, that while MoE makes computation cheaper, the very nature of distributed execution makes it such that the gains are not automatically reflected in faster execution. Hwang et al. (2022) discuss the issue in Tutel, with related work Shi et al. (2024) examining task scheduling and communication in distributed MoE training.

Therefore, it is important to note that any MoE implementation needs to factor in not just the model-specific optimizations, but also the degree to which

the distributed execution can be made to support the specific characteristics of MoE, such as balancing the load across workers.

3.2 Adaptive Scheduling and Training Throughput

In general, scheduling strategies tend to rely on fairly static assumptions about jobs and available resources, which is not always easy in practice. While static allocation of resources can be fairly effective in cases where workloads are predictable, most large-scale AI training jobs are far from it.

Pollux exemplifies this by being a self-tuning system, capable of adapting not just to general workloads but also to specific training setups, avoiding some of the issues of dynamic allocation by making certain aspects of the system behavior reactive rather than fixed (Qiao et al., 2021). It is worth noting that the system demonstrates excellent results in terms of minimizing average job completion time, outperforming many similar systems.

Tutel's adaptation mechanism similarly takes advantage of the fact that MoE workloads are variable, but at the lower level of individual job execution: by using adaptive parallelism and pipelining that react to the changing nature of the work rather than enforce a strict and static schedule.

ScheMoE takes the principle to the lowest level of execution by making the task scheduling itself adaptive.

While each of these systems applies the principle in slightly different ways, the common theme is that the scheduler should not rely on fixed characteristics of jobs that do not change once execution has started.

It is also important to note that while higher throughput is often associated with good scheduler performance, it is not the only factor that matters. In particular, when training models, it is often useful to distinguish between raw tokens per second and goodput, i.e. the number of tokens that contribute meaningfully to training progress. While the two metrics are obviously related, system factors can cause them to diverge (Qiao et al., 2021). As such, a scheduler that helps maximize goodput should be evaluated based on that metric.

3.3 Faults, Failures and Recovery

With very large systems, particularly those utilizing tens of thousands of GPUs, the likelihood of a fault occurring during training becomes close to 100%, and the impact of such a fault can be enormous, both in terms of wasted computation and the resulting disruption to the schedule. First and foremost, any distributed execution has to contend with the possibility that one of the worker nodes will fail at any point during the execution. This issue becomes particularly acute for synchronization-heavy workloads, as a single failed worker can cause the entire distributed job to restart, resulting in large amounts of recomputation being necessary.

Gupta et al. (2024) examine the issue in the context of Just-In-Time Checkpointing, where the goal is to reduce the overhead of traditional periodic checkpointing while reducing the amount of computation wasted on recomputation after a failure.

A similar problem is addressed by Oobleck, which, however, addresses the issue from the other end. Rather than trying to recover state after a failure, it attempts to avoid such failures by rebuilding the distributed execution based on pipeline templates at run-time (Jang et al., 2023).

The solutions by Gupta et al. and Jang et al. demonstrate the importance of ensuring that faults in execution do not lead to significantly worse training outcomes than would be the case without them. In particular, it is important to note that restarting an AI training job after a worker failure is much more wasteful than simply checkpointing once in a while. Therefore, any distributed training framework is better off using approaches that minimize the amount of computation lost to such failures.

3.4 Stragglers and Workload Imbalance

Not all issues can be seen as the entire GPU being faulty. There are occasions where the GPU processes much slower than other similar hardware units, which makes it a straggler.

Stragglers are especially difficult to handle during synchronous distributed training due to the head-of-line blocking nature of the synchronization mechanism, where faster workers need to potentially wait for the slowest worker to finish before proceeding to the next iteration.

Meanwhile, MoE training can have inherent imbalances causing stragglers even when the underlying hardware is not faulty, since uneven token distribution can cause certain experts, and thus their corresponding GPUs, to finish processing their tokens much later than others (Hwang et al., 2022; Shi et al., 2024).

Furthermore, this indicates that fault tolerant scheduling can also benefit from early indications of imminent failures such as increased latencies and GPU utilization, or network congestion and repeated slowdowns.

A scheduler that is able to detect imminent failures can also be utilized to recover before the error becomes critical or before the imminent slowdown takes full effect.

3.5 Communication as a Scheduling Problem

One major characteristic of distributed MoE training is the communication cost incurred by expert routing.

Since tokens are distributed among the various experts, if the expert is on a different GPU, the tokens need to be communicated across the network for processing. This can cause network congestion if many tokens are routed across nodes.

As such, it is clear that the question "which GPU should be scheduled to do which computation?" is inextricably linked with "how much data will need to be sent to which GPU?"

ScheMoE recognizes this scheduling challenge by explicitly considering the interaction and proposing a distributed MoE training system that takes task

scheduling into account (Shi et al., 2024). HetuMoE similarly studies communication-efficient distributed MoE training (Nie et al., 2022).

For an adaptive scheduler, the scheduling decision needs to take into account all three factors: how much computation a GPU can do, how much work is currently being sent to a GPU, and how much data needs to be sent to a GPU. This is especially true since, at scale, network bottlenecks can become a more critical constraint than raw computational power.

3.6 Energy Efficiency in Distributed AI Training

With the scale of operations of distributed AI training, energy has become a major concern. While it is often a second order consideration at smaller scales, the sheer size of large model training means that even small improvements in energy efficiency can have a major impact, due to the huge scale of operations.

One major finding across the various papers is that more powerful accelerators do not necessarily imply more energy efficient training.

A larger GPU, or adding more GPUs, can reduce training time, but at the cost of increased power consumption. Using higher power GPUs can increase power performance per watt but at the cost of instantaneous compute power. Thus, it becomes a scheduling optimization question between performance and energy efficiency.

Zeus attempts to explicitly incorporate energy considerations into training optimization, with particular emphasis on finding a Pareto frontier between the two. It recognizes the assumption that faster configurations are inherently better is flawed, and attempts to find configurations that offer good performance while also reducing the overall energy consumption for training (You et al., 2023).

The result shows that substantial gains in energy efficiency can be had without large compromises in performance. Perseus similarly looks at energy efficiency by considering the ratio of computation energy to useful computation, finding that energy can be saved while maintaining throughput for the evaluated large model workloads (Chung et al., 2024).

By recognizing energy efficiency as a scheduling consideration, these works allow for optimizations that would not be possible otherwise, such as comparing two scheduling options which offer similar instantaneous performance but with one offering significantly lower energy consumption.

3.7 Bringing Fault Tolerance and Energy Efficiency Together

At first glance, the problem of fault tolerance and energy efficiency may seem unrelated. However, in reality, they are closely linked. Say, for instance, a training job had to restart a GPU computation due to a GPU failure after an hour of uninterrupted computation. If the training job had to recompute the dropped hour, this would lead to additional energy expenditures with no tangible outcome. With improved fault tolerance mechanisms, the system would have the ability to save energy by not recomputing the known results. In effect, a better ability to recover from failures translates into energy efficiency gains.

The inverse of the above is also true: overly conservative fault tolerance mechanisms may lead to higher energy expenditures. For instance, if the system started to keep redundant GPUs in case of a worker failure, the extra GPUs would incur additional energy costs. The energy efficiency challenge, therefore, is linked to the fault tolerance one via the ability to make resource allocation decisions aware of the likelihood of various types of failures.

An adaptive scheduler would be able to improve system-level energy efficiency by doing the following:

- Prioritize redundancy in case of high chances of failure and lower it otherwise.
- Create more checkpoints when their loss would be costly to recover.
- Offload less critical workloads to the failing hardware.
- Allocate energy-efficient GPUs to less critical workloads and place communication-heavy experts physically close to each other.

In other words, the fault tolerance mechanisms should be adjusted based on the value of computations already incurred to avoid wasting resources on transient failures. Finally, the ability to recover from failures should be seen as part of a broader optimization problem, with maximizing the amount of useful training done per unit of computational energy spent being its objective function.

3.8 Heterogeneous Resources

A major consideration when trying to optimize large-scale GPU cluster performance is heterogeneity. First and foremost, any particular cluster is likely to have access to GPUs of different levels of performance. Some GPUs will offer higher peak TFLOPs, some – have larger memory bandwidth and VRAM capacity. Additionally, different types of networking hardware have different communication characteristics. As a result, a scheduler that does not account for heterogeneity will not fully utilize the available resources.

Pollux shows the benefits of heterogeneous resource allocation based on the current workload characteristics (Qiao et al., 2021), while Varuna demonstrates the resource allocation benefits for large model training with heterogeneous infrastructure (Athlur et al., 2022). When it comes to distributed MoE training, heterogeneity presents additional opportunities to allocate workloads based on their characteristics. In particular, more powerful GPUs could be used to process more computationally expensive experts, while energy-efficient GPUs could be utilized for less taxing experts. In effect, there is potential to raise the overall resource utilization of the cluster without allocating more GPUs to the training jobs.

3.9 Geo-Distributed Training

The problem becomes more interesting when distributed workloads span multiple geographical locations. At a high level, a geo-distributed training job has to account for multiple factors, ranging from available communication infrastructure to varying levels of energy consumption, availability, and stability of electricity. The communication costs of MoE tokens flowing over geographical distances create an additional challenge for geo-distributed MoE training jobs. In effect, a scheduler operating in this environment would have to optimize for a number of factors, including but not limited to:

- Computational capabilities of different nodes.

- Communication characteristics of inter-node bandwidth.
- Expert placement strategy and its impact on data center traffic.
- Failure likelihood at different nodes.
- Electricity consumption costs of different nodes.
- Carbon footprint of electricity at various nodes.

In effect, there is potential for future work on MoE scheduling that would operate at the intersection of heterogeneous resource allocation and geographical distribution of training jobs. By accounting for variations in both computational and carbon footprints, such a scheduler would be able to make more optimal resource allocation and expert placement decisions.

3.10. Synthesis of the Reviewed Literature

The literature reviewed in this study does not point to a single solution. Instead, different studies address different parts of the same broader problem.

Table 1: Synthesis of the Reviewed Literature

Study	Main contribution	Relevance to the proposed research
Fedus et al. (2022)	Sparse MoE architecture	Establishes the computational basis for MoE
Narayanan et al. (2021)	Large-scale distributed training	Provides the distributed-training foundation
Qiao et al. (2021)	Adaptive cluster scheduling	Supports adaptive resource allocation
Hwang et al. (2022)	Adaptive MoE execution	Supports dynamic MoE scheduling
Nie et al. (2022)	Distributed MoE communication	Highlights communication challenges
Athlur et al. (2022)	Adaptive resource configuration	Supports heterogeneous resource management
You et al. (2023)	Energy-aware GPU scheduling	Connects scheduling with energy efficiency
Jang et al. (2023)	Dynamic failure recovery	Supports adaptive fault tolerance
Thorpe et al. (2023)	Resilience to preemption	Demonstrates resilient distributed training
Chung et al. (2024)	Reducing energy waste	Supports energy-aware training
Gupta et al. (2024)	Efficient checkpointing	Reduces recovery-related computation
Shi et al. (2024)	MoE task scheduling	Connects workload and communication scheduling
Zeng et al. (2025)	Adaptive MoE load balancing	Strengthens the case for dynamic workload management

The overall picture is therefore becoming clearer. Adaptive scheduling, fault tolerance and energy-aware optimization are each well-researched areas, but their intersection in the context of distributed MoE training is under-explored.

3.11 Theoretical Perspective

The proposed study can be viewed from three theoretical perspectives.

Resilience Engineering

resilience engineering is about system behavior under unforeseen circumstances; about maintaining operations and recovering performance after incidents. Ooblock and Just-In-Time Checkpointing are examples of scheduling techniques that make distributed training more resilient to failures by reducing the amount of computation lost when a worker fails, as they checkpoint less frequently (Jang et al., 2023; Gupta et al., 2024).

Another way to think about the contributions of this work is through the lens of resilience engineering. A truly robust distributed training system is one that recognizes that failures will happen, and is designed in such a way that failures have lesser impact on overall progress.

Adaptive Control

Adaptive scheduling can also be viewed as an implementation of a general concept of adaptive control. Control theory is often thought of as a domain of robotic or mechatronic systems, but the underlying concepts apply to a wide variety of problems. One simple way to think about adaptive control is to imagine a control loop: based on current observations, the controller performs an action, then updates its model of the world based on new observations. Tutel, Pollux and Zeus each perform adaptive scheduling in this manner, varying the degree to which they utilize the flexibility of parallelism (Hwang et al., 2022; Qiao et al., 2021; You et al., 2023).

Sustainable Computing

A third theoretical perspective is to examine scheduling through the lens of sustainable computing . Broadly speaking, this is the idea that computing performance cannot be evaluated in isolation from the resources spent on achieving it. It is therefore a question not only of how quickly a model can be trained, but also of how much electricity and therefore carbon emission must be spent on that training. Sustainable computing is an increasingly important consideration for large-scale AI systems; for example, a small decrease in efficiency can have large downstream effects when considering thousands of GPUs multiplied by weeks or months of training time.

3.12 Proposed Conceptual Framework

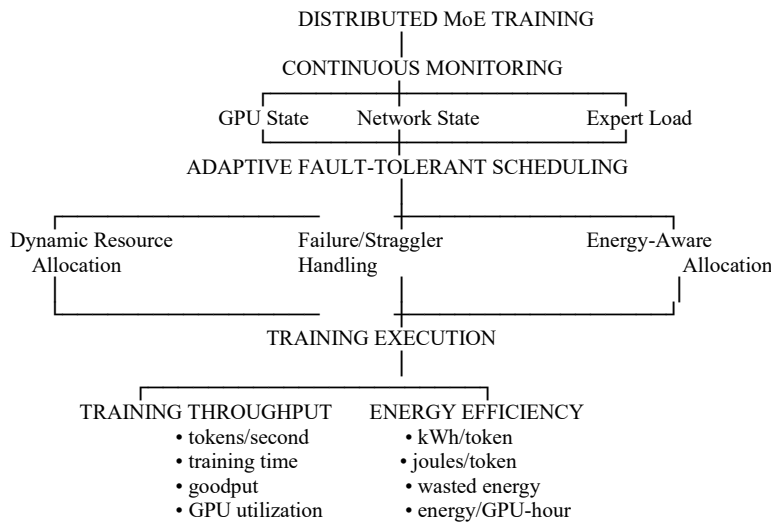


Figure 1: Proposed Conceptual Framework

The framework proposed that adaptive fault-tolerant scheduling is the key to binding changing situations in distributed-systems to both of the target objective, namely training throughput and energy efficiency. By taking in the GPU, networking, expert workloads and recovery as inputs, adaptive fault-tolerant scheduling makes decisions on redistribution, recovery, placement which leads to the final energy-aware decision-making.

The target objective therefore becomes a system where there are minimal waiting time, reduced redundant computation and better utilization of the available energy.

4. Discussion

Based on the reviewed literature, it is possible to conclude that the push towards adaptive scheduling stems from the need to address the growing complexity of distributed AI systems. Traditional distributed systems do not always require adaptive scheduling since the scheduling decisions are based on rather static assumptions regarding workloads and the characteristics of the underlying systems. However, modern AI training presents unique challenges associated with varying workloads at different stages, heterogeneous infrastructures, changing networking conditions, and the likelihood of failures when dealing with extremely large models.

MoE adds to those challenges by introducing additional variations in workloads due to routing. Tutel and ScheMoE show the need for adaptive scheduling in the context of MoE by demonstrating the value of considering the MoE-specific behaviors (Hwang et al., 2022; Shi et al., 2024). Pollux shows the importance of adaptive resource allocation decisions in a similar context (Qiao et al., 2021). Oobleck and Just-In-Time Checkpointing demonstrate the need for adaptive recovery scheduling (Jang et al., 2023; Gupta et al., 2024).

Zeus and Perseus demonstrate that similar adaptations can be used to improve the energy-efficiency of distributed AI training (You et al., 2023; Chung et al., 2024).

It is important to note that while these solutions highlight the importance of adapting to changing conditions, there is still an opportunity to better bind the adaptation to the specific objective.

Adaptive fault-tolerant scheduling is therefore the next step in the evolution of distributed AI training systems. It entails making scheduling decisions that consider the trade-offs between throughput, recovery, and energy-efficiency, thus a multi-objective optimization problem.

A practical example of such a consideration would be to avoid redistributing the workload when a GPU starts to perform below average as the cost of migration could outweigh the benefit.

Similarly, the system would not save a checkpoint after every iteration due to the overhead. The scheduler would have to assess whether a particular action would actually be beneficial in a given situation. That is why adaptive fault-tolerant scheduling is a multi-objective optimization problem.

A simplified objective can be expressed as:

$$\begin{aligned} & \left[\begin{aligned} & \max U = \\ & \alpha T + \beta R + \gamma EE - \delta O \end{aligned} \right] \end{aligned}$$

where:

- (T) = training throughput;
- (R) = resilience;
- (EE) = energy efficiency; and
- (O) = scheduling and recovery overhead.

The weights (α), (β), (γ) and (δ) can be tuned according to the criteria of the target application domain.

4.1 Research Dilemma

Conceptual Dilemma

In this regard, the body of existing work provides fertile ground for future research into holistic scheduling approaches that account for performance, reliability, and energy consumption in a coordinated manner. The individual contributions address adaptive MoE execution, cluster scheduling, fault tolerance, and energy efficiency, but each focuses on isolated aspects.

Methodological Dilemma

Moreover, most of the evaluated methods were tested under particular hardware configurations and workloads, which limits the understanding of their behavior in diverse environments. While the selection of scenarios is natural to assess the method's strengths and weaknesses, the analysis should also consider how a particular scheduler behaves when the level of heterogeneity, the degree of network congestion, or the failure rate varies.

Thus, future studies should test the adaptability and robustness of the considered approaches by varying

- the GPU heterogeneity,
 - the cluster size,
 - the level of network congestion,
 - expert-load imbalance,
 - failure frequency,
 - the cost of recovery,
 - checkpoint intervals, and
- energy constraints.

4.3 Measurement Dilemma

Finally, there is also a dilemma related to the choice of evaluation metrics.

Most works prioritize either performance or energy efficiency; however, an equally important task is to evaluate how a given approach performs when both are considered.

Table 2: Potential metrics include:

Dimension	Metric
Throughput	Tokens/second
Training efficiency	Goodput
Utilization	GPU utilization (%)
Resilience	Recovery time
Wasted computation	Recomputed iterations/tokens
Energy	kWh
Energy efficiency	Joules/token
Sustainability	CO ₂ e
Scheduling cost	Scheduling overhead

This would enable researchers to assess whether a particular scheduler actually improves the utilization of the system as opposed to shifting the overhead elsewhere.

4.4 Contextual Gap

Much research into large-scale AI infrastructure comes from organizations which possess large-scale infrastructure. Less research has been done on smaller shared or heterogeneous clusters.

By addressing this issue, the study can show that adaptive scheduling can make limited clusters more fruitful in terms of research.

4.5 Sustainability Gap

Energy aware scheduling and fault tolerance are both desirable properties, but their intersection is understudied. Faults in the system lead to increased energy consumption due to redundant computations, and redundancy without faults also increases energy consumption.

A holistic understanding of the connection between the two concepts is necessary to design sustainable fault tolerant AI infrastructure

4.6 Implications of the Review

Theoretical Implications

The article demonstrated how three different fields – namely adaptive control theory, resilience engineering and sustainable computing – are woven together to form a cohesive view of the large-scale distributed AI infrastructure as a system which requires continuous adaptation to maintain utility while balancing consumption.

Practical Implications

For the practitioners, this research demonstrates why context aware scheduling is necessary to maximize the utility of the AI infrastructure. Instead of merely utilizing the available GPU resources, the scheduler must attempt to balance the utilization between nodes, taking into account communication overhead, fault tolerance and energy consumption.

The works of Tutel, Pollux, Oobleck and Zeus demonstrate that many of the required features are available in the aforementioned works (Hwang et al., 2022; Qiao et al., 2021; Jang et al., 2023; You et al., 2023).

The practical implication of the research is the combination of these technologies.

Implications for Resource Constrained Environments

This research has the potential to improve the utilization of the resources in the environments where such constraints are present. Those limitations are commonly found in practice, which makes the implications of the research valuable. If the organization cannot scale its infrastructure linearly with the increase of performance requirements, it may utilize the techniques listed in the article to obtain higher utilization of their existing GPUs rather than buying new accelerators. The removal of the overhead of idle time, communication and recomputation would have a significant effect on any business which employs distributed training.

5. Conclusion and Recommendations

5.1 Proposed Research Direction

Based on the literature, the future research should design an adaptive scheduler with six interrelated capabilities.

These include continuous monitoring, resource manager, predictive failure detection, self-healing, dynamic optimization, and machine learning.

The scheduler would continuously monitor GPU utilizations, memory, temperatures, network latencies, bandwidths, expert distributions, power draws, and hardware health.

Conceptually:

$$[P(F_i|X_i)]$$

where (F_i) represents failure of resource (i) , while (X_i) represents its observed characteristics.

Dynamic Resource Allocation

Resources could then be reassigned according to current workload, GPU capabilities, communication conditions and estimated risk for failure.

Adaptive Checkpointing

Could be made based on probability and impact of failure rather than a global uniform schedule.

Multi-Objective Decision Making

Is ultimately needed in order to balance between Throughput, Resilience and Energy Efficiency.

5.2 Conclusion

The evolution of large AI models has fundamentally changed the requirements for distributed computing. Modern training environments must scale to encompass thousands of GPUs, while also accounting for variable workloads, communication overhead, hardware failures and energy constraints.

MoE architectures offer a compelling path towards capacity scaling, but their router-based nature creates both additional overhead and unique risks (Fedus et al., 2022; Hwang et al., 2022). Moreover, modern distributed systems pose serious risks in terms of both hardware failures and straggler processes.

The works reviewed in this survey demonstrate that adaptive approaches are crucial in addressing these challenges.

Tutel and ScheMoE highlight the importance of dynamic adaptation of MoE-based workloads (Hwang et al., 2022; Shi et al., 2024), while Pollux demonstrates the value of adaptive resource scheduling (Qiao et al., 2021). The concepts of Oobleck and Just-In-Time Checkpointing show how training can be optimized in terms of fault tolerance (Jang et al., 2023; Gupta et al., 2024). Finally, Zeus and Perseus highlight the possibility to include energy efficiency as an optimization target (You et al., 2023; Chung et al., 2024).

The key point of this survey is to highlight that these are all facets of the same problem: A distributed MoE training process should not be scheduled as if it was a rigid batch job. Failures cause drops in throughput and extra energy expenditure. Imbalanced router workloads create both straggling processes and wasted energy. Extra redundancy adds to the resilience but also increases energy costs. The same applies to aggressive optimizations that sacrifice quality in favor of speed and capacity.

Thus, the key research challenge is to ensure that the training job always follows an optimal schedule in terms of balancing these factors. This survey therefore identifies adaptive fault tolerant scheduling as a critical research challenge for distributed MoE training. The envisioned scheduler must be adaptive to changing conditions and workloads. It must balance between the various optimization targets. Most importantly, it must account for energy efficiency in both normal operations and recovery from faults.

The envisioned solution should provide significant benefits in terms of training speed, reduced wasted energy and enhanced resilience. However, the

main benefit would come from the fact that a well-designed adaptive scheduler could maximize the utilization of the existing resources, rather than having to provision extra capacity just to account for potential failures.

The key limitation of this survey is that it highlights the lack of research that actually evaluates this hypothesis. In other words, the main recommendation is to perform empirical analysis of how adaptive fault tolerant scheduling would benefit distributed MoE training in terms of throughput, resilience and energy efficiency.

REFERENCES :

- Athlur, S., Saran, N., Sivathanu, M., Ramjee, R., & Kwatra, N. (2022). Varuna: Scalable, low-cost training of massive deep learning models. *Proceedings of the 17th European Conference on Computer Systems (EuroSys 2022)*.
- Chung, J.-W., Gu, Y., Jang, I., Meng, L., Bansal, N., & Chowdhury, M. (2024). Reducing energy bloat in large model training. *Proceedings of the 30th ACM Symposium on Operating Systems Principles (SOSP 2024)*.
- Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120), 1–39.
- Gupta, T., Krishnan, S., Kumar, R., Vijeev, A., Gulavani, B., Kwatra, N., Ramjee, R., & Sivathanu, M. (2024). Just-In-Time checkpointing: Low cost error recovery from deep learning training failures. *Proceedings of the 19th European Conference on Computer Systems (EuroSys 2024)*.
- Hwang, C., Cui, W., Xiong, Y., Yang, Z., Liu, Z., Hu, H., Wang, Z., Salas, R., Jose, J., Ram, P., Chau, J., Cheng, P., Yang, F., & Xiong, Y. (2022). Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 4, 269–287.
- Jang, I., Yang, Z., Zhang, Z., Jin, X., & Chowdhury, M. (2023). Oobleck: Resilient distributed training of large models using pipeline templates. *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP 2023)*.
- Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V. A., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., Phanishayee, A., & Zaharia, M. (2021). Efficient large-scale language model training on GPU clusters using Megatron-LM. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2021)*.
- Nie, X., Zhao, P., Miao, X., Zhao, T., & Cui, B. (2022). HetuMoE: An efficient trillion-scale mixture-of-expert distributed training system. *arXiv preprint arXiv:2203.14685*.
- Qiao, A., Choe, S. K., Subramanya, S. K., Neiswanger, W., Ho, Q., Zhang, H., Ganger, G. R., & Xing, E. P. (2021). Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2021)*.
- Shi, S., Pan, X., Wang, Q., Liu, C., Ren, X., Hu, Z., Yang, Y., Li, B., & Chu, X. (2024). ScheMoE: An extensible mixture-of-experts distributed training system with tasks scheduling. *Proceedings of the 19th European Conference on Computer Systems (EuroSys 2024)*, 236–249.
- Thorpe, J., Zhao, P., Eyolfson, J., Qiao, Y., Jia, Z., Zhang, M., Netravali, R., & Xu, G. H. (2023). Bamboo: Making preemptible instances resilient for affordable training of large DNNs. *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2023)*.
- You, J., Chung, J.-W., & Chowdhury, M. (2023). Zeus: Understanding and optimizing GPU energy consumption of DNN training. *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2023)*, 119–139.
- Zeng, Y., Huang, C., Mei, Y., Zhang, L., Su, T., Ye, W., Shi, W., & Wang, S. (2025). EfficientMoE: Optimizing mixture-of-experts model training with adaptive load balance. *IEEE Transactions on Parallel and Distributed Systems*, 36(4), 677–688.