

Partial Experts Checkpoint: Efficient Fault Tolerance for Sparse Mixture-of-Experts Model Training

Weilin Cai Le Qin Jiayi Huang *

The Hong Kong University of Science and Technology (Guangzhou)
 {wcai738, lqin674}@connect.hkust-gz.edu.cn
 {hjiy}@hkust-gz.edu.cn

Abstract

As large language models continue to scale up, the imperative for fault tolerance in distributed deep learning systems intensifies, becoming a focal area of AI infrastructure research. Checkpoint has emerged as the predominant fault tolerance strategy, with extensive studies dedicated to optimizing its efficiency. However, the advent of the sparse Mixture-of-Experts (MoE) model presents new challenges for traditional checkpoint techniques due to the substantial increase in model size, despite comparable computational demands to dense models. Breaking new ground in the realm of efficient fault tolerance for MoE model training, we introduce a novel Partial Experts Checkpoint (PEC) mechanism alongside a corresponding PEC fault-tolerant system. Our approach strategically checkpoints a selected subset of experts, thereby significantly reducing the checkpoint size for MoE models to a level comparable with that of dense models. The empirical analysis on our 8-expert GPT-MoE model demonstrates that the proposed PEC approach facilitates a substantial 54.2% decrease in the size of non-redundant checkpoint (no data-parallel duplication), without compromising the final model quality. Moreover, our PEC fault-tolerant system achieves a 76.9% reduction in checkpoint workload per data-parallel distributed rank, thereby correspondingly diminishing the checkpointing time and facilitating complete overlap with the training process.

1 Introduction

Transformer-based large language models (LLMs), which scale to billions or even trillions of parameters, have emerged as the most trending topic in AI research due to their impressive capabilities [1, 5, 55, 33, 54, 3, 50]. Recently, the sparsely-gated mixture-of-experts (MoE) has become the preferred methodology to increase parameter counts and enhance the model quality of LLMs without a proportional increase in computational requirements [43, 41, 42, 15]. To facilitate the training of MoE models and their deployment across expansive computing clusters, distributed deep learning systems have been refined to incorporate expert parallelism [20, 10, 14, 46, 12] alongside established frameworks of data parallelism [38, 40] and model parallelism [45, 30, 47, 13, 29, 22, 18]. With the escalation in the number of deployed computing devices and the incidence of faults [26, 56, 16, 11, 7], ensuring fault tolerance has become a critical component of AI system infrastructure.

Although numerous studies have effectively addressed fault tolerance in dense (non-MoE) models [52, 16, 31, 56, 51, 49], the distinctive characteristics of MoE models necessitate specialized strategies to assure their reliable and efficient fault-tolerant training. As MoE models scale to unprecedented sizes, they introduce significant complications to the currently mainstream fault tolerance method—checkpointing [28, 26, 9, 52], which involves periodically saving training states, usually including model parameters, optimizer state, and epoch/iteration counts. The dramatic escalation

*Corresponding author.

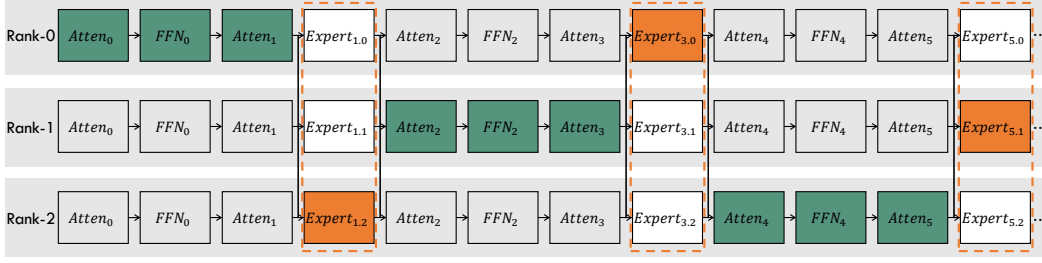


Figure 1: The saved components of training states in a PEC checkpoint, including the expert part with partial selection (orange) and the non-expert part with data-parallel sharding (green). In the model training with data parallelism and expert parallelism, the non-expert part is replicated across distributed ranks while each unique expert in a MoE layer is assigned to a specific rank. Unsaved experts are indicated in white, and grey denotes non-expert segments not saved due to redundancy.

in model size intrinsic to MoE models leads to a significant increase in checkpoint data volume, imposing a storage burden that distributed filesystems struggle to manage efficiently [37, 44, 8]. Moreover, the enlarged checkpointing duration of MoE models cannot be completely overlapped with the training process via existing asynchronous strategies [26, 28, 31, 52], yielding additional overhead that prolongs the total time required for fault-tolerant training. In summary, current research has not yet provided an effective solution to the efficiency challenges arising from the substantial increase in checkpoint size of MoE models.

Pioneering efficient fault tolerance for MoE model training, we introduce a novel Partial Experts Checkpoint (PEC) mechanism designed to substantially reduce checkpoint size. The PEC selectively saves only a subset of experts (K_{pec} experts per MoE layer) during each checkpointing, while fully saving the non-expert components of the model. When K_{pec} is set to 1, the resulting checkpoint size is on par with that of a dense model, thereby markedly diminishing the checkpointing overhead.

However, it is crucial to acknowledge that the recovery process of PEC may potentially compromise the model’s training accuracy, as it introduces a variant of dropout on the updates of the expert parameters, contingent on the input tokens. To quantitatively evaluate the impact of PEC on accuracy, we propose the Portion of Lost Tokens (PLT) metric. Our empirical results reveal an inverse relationship between model quality and PLT, yet we find that the model quality maintains akin to the non-fault case when PLT does not exceed 7.5%. We posit that a tolerable degree of PLT is unlikely to harm final model accuracy, considering the intrinsic presence of dropout in model layers [48] and the expert-capacity-related dropout that facilitates token load balancing within MoE layers [20, 10].

Building upon the efficacy of PEC, we introduce the PEC fault-tolerant system, enhanced by a series of optimization strategies. As depicted in Figure 1, we achieve further reductions in checkpoint size through the integration of our proposed adaptive data-parallel sharding strategy. Additionally, we implement an asynchronous checkpointing scheme utilizing a triple-buffering approach, which allows for full overlap between the checkpointing process and the training process. Considering the trade-off between fault tolerance efficiency and model quality, we further introduce a Dynamic-K strategy and a two-level PEC structure, comprising snapshot-PEC and persist-PEC.

Experimental results from training our 8-expert GPT-MoE model indicate that our proposed PEC approach yields a significant 54.2% reduction in the size of non-redundant checkpoint (no data-parallel duplication) while preserving the final model quality. Moreover, our PEC fault-tolerant system achieves a 76.9% reduction in checkpoint workload per data-parallel distributed rank, thereby correspondingly decreasing checkpointing duration and permitting full overlap with the training workflow. Additionally, extensive experiments substantiate the effectiveness of our other proposed methods in optimizing the trade-off between efficiency and model quality. In summary, our contributions are:

- We propose the Partial Experts Checkpoint (PEC) mechanism, a novel approach designed to significantly reduce the checkpoint size of MoE model to a level akin to that of dense model.
- We introduce the PEC fault-tolerant system, integrating PEC with a series of optimization strategies to further enhance the efficiency of fault-tolerant checkpointing.

- We propose the Portion of Lost Tokens (PLT) metric to quantitatively evaluate the accuracy impact of PEC, and conduct extensive experiments to substantiate the superior performance of our approach in enhancing fault tolerance efficiency without sacrificing model quality.

2 Background & Related Work

2.1 Sparse Mixture-of-Experts Models

The sparse Mixture-of-Experts (MoE) layer, as introduced by Shazeer et al. [43], consists of multiple feed-forward networks (FFNs), termed “experts”, and a trainable gating network for selectively activating a subset of these experts during each iteration. Formally, with N expert networks $\{E_i\}_1^N$, gating network G , and input x , the MoE layer’s output is given by:

$$MoE(x) = \sum_{i=1}^N G(x)_i E_i(x) \quad (1)$$

In alignment with common practices in existing MoE research, we use the noisy top-k softmax gating network to select the top-ranked experts for the computation, formulated as

$$G(x) = TopK(Softmax(f(x) + \epsilon)) \quad (2)$$

where $f(\cdot)$ denotes the gating linear transformation and ϵ is the Gaussian noise. Leveraging the sparse activations yielded by $G(x)$, this approach facilitates a substantial augmentation of model parameters without causing a proportional increase in computational cost. Furthermore, the multiplicity of FFN experts within MoE models engenders a significant rise in checkpoint data volume, thereby presenting the challenges of checkpointing efficiency.

2.2 Fault-tolerant Checkpointing for Distributed Training System

Checkpointing serves as a critical mechanism for augmenting fault tolerance in distributed training systems by facilitating the periodic preservation and recovery of training states [28, 16, 56, 19, 17]. This safeguard ensures that training progression is not totally lost in unexpected faults and can resume post-restart. However, the checkpointing process incurs significant data transfer and storage overhead, alongside additional overhead in the event of a fault.

The main overhead introduced by fault-tolerant checkpointing, denoted as O_{ckpt} , can be quantified by aggregating the overhead of training states saving O_{save} under normal training, the overhead of system/task restart $O_{restart}$ and lost computations O_{lost} during the fault recovery. It is formulated as:

$$O_{ckpt} = O_{save} \frac{I_{total}}{I_{ckpt}} + \sum_{i=1}^{N_{fault}} (O_{restart}^i + O_{lost}^i) \quad (3)$$

where I_{total} represents the total number of iterations in training and I_{ckpt} denotes the iteration interval of checkpointing. Each fault occurrence, totaling N_{fault} , contributes to the overhead, with O_{lost} being contingent on I_{ckpt} and averaging $\frac{I_{ckpt}}{2}$, and $O_{restart}$ remaining relatively constant.

In pursuit of optimizing fault tolerance efficiency, existing research has explored various methods to diminish each component of the above overhead. Some studies minimize O_{save} by leveraging hierarchical storage [28, 51] and heterogeneous network [52] to accelerate checkpointing, as well as adopting asynchronous execution to concurrently run checkpointing alongside the training workflow [28, 26, 16]. Such enhancements to the speed of data movement and storage also favorably impact $O_{restart}$. Moreover, some studies reduce O_{lost} by increasing checkpoint frequency (reducing I_{ckpt}) [28, 52, 9], though this adjustment may enlarge $O_{save} \frac{I_{total}}{I_{ckpt}}$ and necessitates a considered trade-off between the two.

Given that data volume substantially determines communication and storage durations, some studies reduce checkpoint size via quantization [9] and data-parallel sharding [31, 51]. Additionally, strategies for checkpointing and recovering partial training states have been proposed [35], exhibiting efficacy in deep learning recommendation systems [26, 9], which is kind of similar to our proposed PEC.

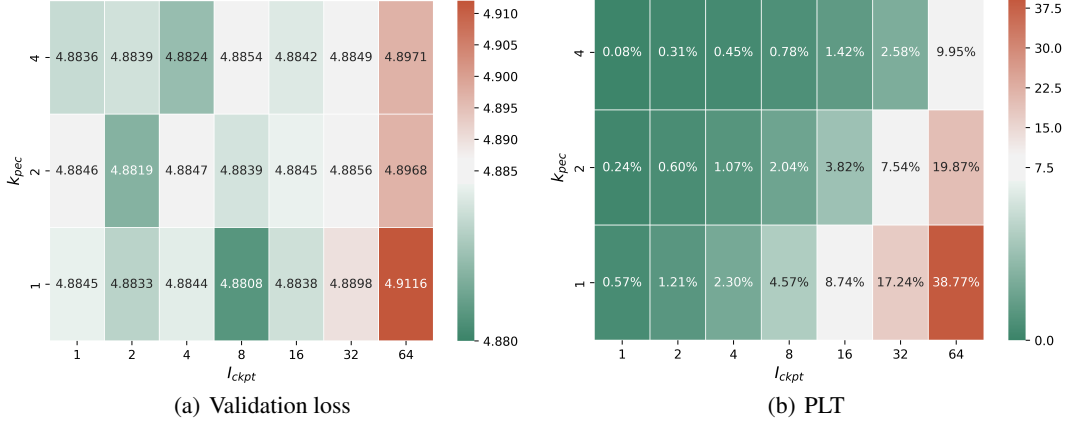


Figure 2: Correlation analysis between (a) the final validation loss and (b) the Portion of Lost Tokens (PLT) in our 8-expert GPT-MoE model training with one fault occurring. In (a), the loss of 4.8851 from the non-fault case is taken as the center value to highlight the accuracy deviations under various PEC configurations. In (b), the PLT centers on 7.5% observed in a PEC configuration of $K_{pec} = 2$ and $I_{ckpt} = 32$, which slightly degrades the model quality compared to the non-fault case.

3 Method

3.1 Partial Experts Checkpoint (PEC)

In light of the substantial increase in checkpoint size predominantly attributed to the multiplicity of FFN experts within the MoE model, we introduce the concept of partial experts checkpoint (PEC) to significantly downsize the checkpoint. In PEC approach, a subset of experts—specifically, K_{pec} per MoE layer—is saved, while the non-expert parts are preserved in their entirety. This strategy results in a checkpoint size comparable to that of a dense model when K_{pec} is set to 1.

As a device-agnostic checkpoint mechanism, PEC is universally applicable across various MoE model training scenarios. In this paper, our primary focus is on the prevalent approach to distributed MoE training that integrates data parallelism with expert parallelism, wherein distinct experts within each MoE layer are allocated across disparate data-parallel ranks. While we concentrate on this configuration, we simplify our discussion by excluding other parallel and deployment strategies that arise at each data-parallel rank, such as tensor-parallelism and pipeline-parallelism, though we note that these methods are compatible with our PEC framework.

Sequential Expert Selection. Considering the expert selection for PEC, the straightforward strategy is to sequentially checkpoint experts within each MoE layer. For example, data-parallel $rank_0$ would save its owned experts at the first checkpoint, followed by $rank_1$ saving its owned experts at the second checkpoint, and this pattern would continue iteratively. To ensure the balance of the storage workload across ranks, we implement an interleaved checkpointing schedule for experts across MoE layers, as exemplified by Figure 1.

It is important to consider that the recovery process of PEC may affect model accuracy in training, as it introduces a form of dropout for experts’ updates posed by the input tokens. To quantitatively assess the potential accuracy impact of PEC, we propose a novel metric, the Portion of Lost Tokens (PLT), defined as follows:

$$PLT = \frac{L_{total}}{T_{total}} = \frac{\sum_{i=1}^{N_{fault}} L_i(F_{select}, K_{pec}, I_{ckpt})}{\sum_{j=1}^{N_{moe}} T_j \cdot TopK_j} \quad (4)$$

where L_{total} refers to the total number of lost tokens, which is influenced by the function F_{select} (e.g. sequential or load-aware methods) and the number K_{pec} of partial expert selection, and the checkpointing interval I_{ckpt} , during N_{fault} checkpointed faults. T_{total} denotes the total number of tokens processed by all experts in N_{moe} MoE layers during one epoch, calculated by multiplying the number of input tokens T and $TopK$ of MoE gating. It is worth noting that the actual count of

Algorithm 1 Load-aware Expert Selection

```

1:  $capacity = \lceil [N_{moe} \times K_{pec} / N_{rank}], \dots \rceil$ 
2:  $unsaved\_load = [0, \dots]$ 
3: for  $last\_layer$  to  $first\_layer$  do
4:    $unsaved\_load += AllReduce(update\_load)$ 
5:    $sorted\_id = Sort(unsaved\_load)$ 
6:    $saved\_count = 0$ 
7:   for  $i = 0$  to  $Length(sorted\_id)$  do
8:     if  $saved\_count = K_{pec}$  then
9:       Break
10:    if  $capacity[sorted\_id[i]] > 0$  then
11:      if  $rank\_id = sorted\_id[i]$  then
12:        CheckpointExpert()
13:         $capacity[sorted\_id[i]] -= 1$ 
14:         $unsaved\_load[sorted\_id[i]] = 0$ 
15:         $saved\_count += 1$ 

```

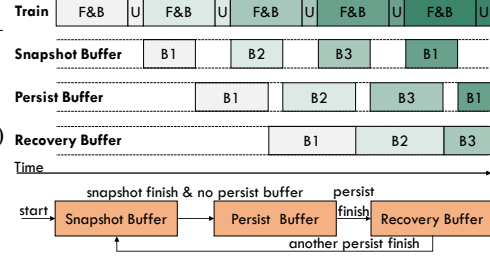


Figure 3: Timeline of the asynchronous checkpointing process with triple-buffering during training. “F&B” denotes the forward and backward passes for each iteration, while “U” indicates the weight update phase. The orange part illustrates the state transition among the three buffers (B1, B2, B3).

tokens processed by all experts is typically less than T_{total} , primarily due to the constraints imposed by the expert capacity [20].

Figure 2 illustrates the impact of PLT on model quality, evidencing that the final validation loss of models recovering from PEC experiences fluctuations (4.8808-4.8854) yet remains comparable to the non-fault case (4.8851) when PLT is below 7.5%. It substantiates that PEC effectively minimizes checkpoint size without compromising model quality. Additionally, it highlights a correlation between smaller K_{pec} and larger I_{ckpt} with increased PLT, which may impact the model quality.

Load-aware Expert Selection. We extend our investigation into the expert selection mechanism employed in PEC by incorporating a load-aware approach that prioritizes the checkpointing of K_{pec} experts, characterized by the highest load of unsaved tokens, as delineated in Algorithm 1. To achieve it, each rank maintains the same list of unsaved loads, which synchronizes the updated token load via the AllReduce operation in each iteration. Additionally, we constrain the equal save capacity of each rank to achieve workload balance among distributed ranks.

Dynamic-K for Fault Accumulation During Prolonged Training Sessions. In practical scenarios, extensive training sessions are susceptible to multiple faults, potentially leading to an augmented PLT. To mitigate this issue, we propose the Dynamic-K strategy, which modulates the K_{pec} parameter in response to fault accumulation, with the objective of maintaining PLT below a 5% threshold. This method recalibrates the K_{pec} value subsequent to each fault recovery incident, predicated on the aggregated PLT incurred by the system. If the aggregated PLT attributable to a specific K_{pec} surpasses its limit, K_{pec} will be doubled, and this process is reiterated until checkpointing all experts.

3.2 PEC Fault-tolerant System

Leveraging our proposed PEC mechanism, we further introduce the PEC fault-tolerant system, meticulously engineered for practical deployment. This system integrates seamlessly with a suite of optimization techniques to ensure efficient orchestration of model checkpoint components. In order to achieve efficient management of all modules of the model checkpoint, we design to store them as key-value pairs within the in-memory database and distributed filesystem, each entry uniquely identified by a distinctive identifier. Additionally, we implement an optimized DistributedSampler with recoverability, efficiently bypassing previously processed data to eliminate extra loading overhead.

Adaptive Data-Parallel Sharding. We explore enhanced checkpointing strategies for the non-expert components that are duplicated across distributed ranks due to data parallelism. By partitioning the shared non-expert components, distributed ranks can concurrently checkpoint their respective shards, thus accelerating the process. Since the checkpointing workload for each rank is contingent upon both expert and non-expert parts, balanced optimization of either part in isolation does not suffice for overall load balance. To evenly distribute the checkpointing workload among ranks, we propose an Adaptive Data-Parallel Sharding (ADPS) strategy, which adaptively allocates non-expert

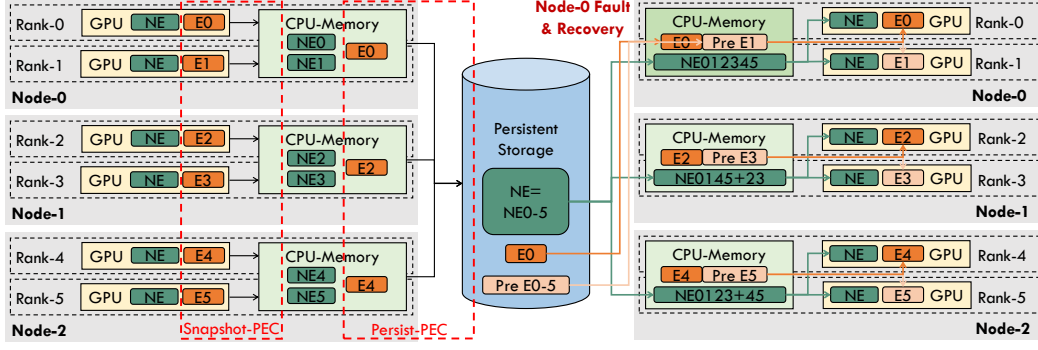


Figure 4: Schematic of two-level PEC saving and recovery within PEC fault-tolerant system. “E(0-5)” denotes different experts across distributed ranks, while “NE” denotes the entire non-expert part and “NE(0-5)” denotes data-parallel shards. “Pre” indicates the data saved by the previous checkpointing. “NE0123+45” and “NE0145+23” indicate that non-fault nodes retrieve only the parts that are not present in the local in-memory snapshot.

components according to the selected partial experts within PEC. ADPS employs a greedy algorithm for shard assignment, prioritizing the allocation of larger model modules to ranks with the least accumulated storage, thereby optimizing workload balance and minimizing total checkpointing duration. Furthermore, no additional synchronization is required, as distributed ranks adhere to the same consistent pattern in PEC expert selection. However, ADPS is incompatible with frameworks like DeepSpeed-Zero [38, 39, 40] and FSDP [57], where model sharding is predetermined and ranks must checkpoint according to this established pattern.

Asynchronous Checkpointing with Triple-Buffering. As depicted in Figure 3, the checkpointing workflow comprises two phases: snapshot (GPU memory to CPU memory) and persist (CPU memory to persistent storage). To minimize the impact of checkpointing on training throughput, we implement an asynchronous checkpointing mechanism that utilizes a triple-buffering approach. This triple-buffering scheme in CPU memory, consisting of snapshot, persist, and recovery buffers, is carefully architected to ensure data integrity during the saving process and data consistency during recovery. Initially, all three buffers are in a snapshot state and each snapshot operation transmits data from GPU into one snapshot buffer. When a snapshot operation completes—and in the absence of an ongoing persist buffer—the corresponding buffer transitions to a persist buffer to begin data transfer to persistent storage. Upon completion of persist phase, the buffer then becomes a recovery buffer, reflecting the most recent checkpoint in persistent storage and available for in-memory snapshot recovery, until another persist buffer transitions. In Figure 3, the time span of each buffer in snapshot or persist state can reflect the time cost of snapshot or persist operation. Additionally, our preliminary analysis suggests that recovering from unsynchronized checkpoints, whether preceding or succeeding the target iteration, is unlikely to harm the final model quality. This insight presents an opportunity to simplify the triple-buffering approach, a prospect we intend to explore and confirm in the future.

It is essential to ensure that snapshot operations are completed within the duration of the forward and backward passes to prevent blocking the training process. Moreover, the duration of the persist phase determines the upper bound for the checkpoint interval, which ideally should be decreased. To optimize the trade-off between efficiency and PLT, we propose a two-level PEC approach: snapshot-PEC and persist-PEC, tailored to the snapshot and persist phases, respectively, thereby adjusting both snapshot and persist durations.

Two-level PEC Saving and Recovery. As illustrated in Figure 4, snapshot-PEC and persist-PEC respectively alleviate data transmission burdens during their respective phases, resulting in reduced time requirements for both snapshot and persist. The composite outcome of these two PEC stages determines the final subset of experts stored in persistent storage. Specifically, the selected experts and data-parallel shards are snapshotted to each node’s CPU memory, while the shared distributed filesystem receives the full non-expert components along with a subset of experts at each interval. In the event of a fault, while the fault node may lose its data, other normal nodes can recover from the in-memory snapshots, thereby mitigating the PLT attributable to persist-PEC. Take Figure 4 as an example, the restarted node-0 needs to load experts from persistent storage, while other nodes

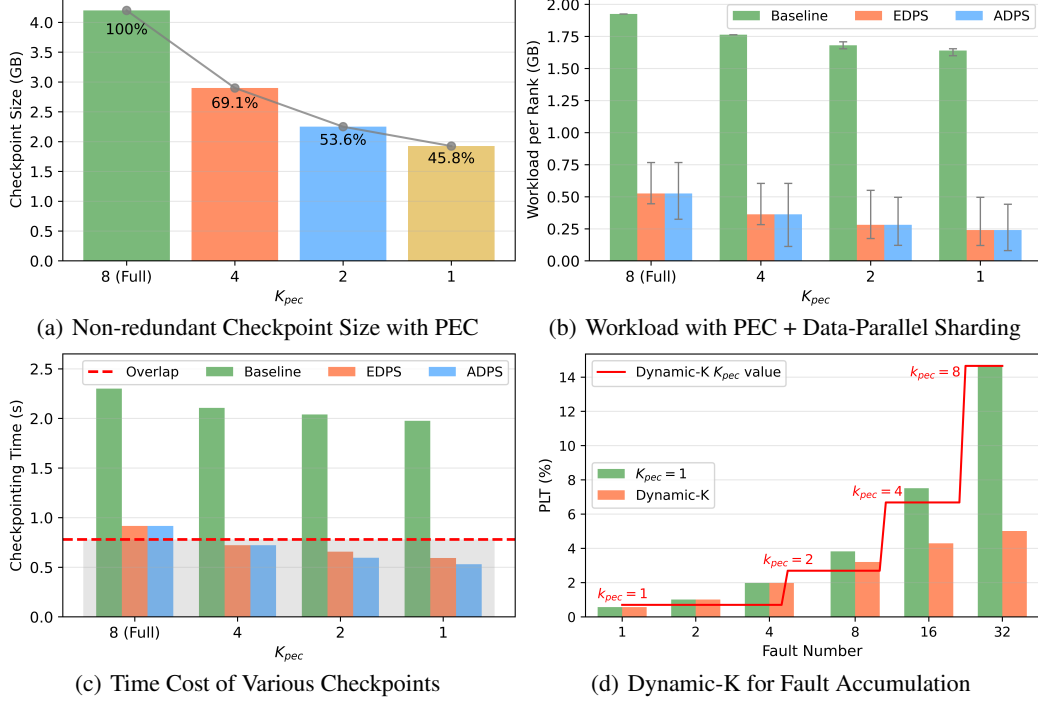


Figure 5: Experimental results of training GPT-MoE with PEC using sequential expert selection. In (b) and (c), the baseline reflects the conventional checkpointing practice wherein each data-parallel rank saves its entire training state. "EDPS" represents the fixed equal data-parallel sharding, and "ADPS" illustrates our novel Adaptive Data-Parallel Sharding. The red "overlap" line in (c) marks the duration of the forward and backward passes per iteration, with the grey area indicating the overlap part of checkpointing with the training process. The red line in (d) tracks the dynamic adjustments of K_{pec} under our Dynamic-K strategy, with setting I_{ckpt} to 1.

can load experts from local in-memory snapshots. Consequently, we can maximize the value of K_{pec} for snapshot-PEC to the extent that it can be fully overlapped by training progress without causing delays. In addition, If employing equal data-parallel sharding of non-expert components like DeepSpeed-Zero [38, 39, 40] and FSDP [57], each node snapshots an equivalent number of experts will not increase overhead, thus increasing K_{pec} of snapshot-PEC while maintaining efficiency.

4 Evaluation

4.1 Experimental Setup

To evaluate the efficacy of our proposed PEC mechanism and PEC fault-tolerant system, we conduct extensive experiments of fault-tolerant MoE model training on a computing node equipped with 8 A30 GPUs in a real data center. Our implemented distributed training system for the experiments is based on PyTorch DistributedDataParallel (DDP) [21, 34] and Tutel [14]. We validate the PEC's influence on model accuracy by testing it on the language-related GPT-MoE [36] model and the vision-related SwinV2-MoE [14, 24] model, respectively. Specifically, the GPT-MoE model is trained on the Wikitext dataset [27], while the SwinV2-MoE model is trained on the ImageNet-1K image classification dataset [6]. Given the constraints of our hardware setup, we configure each MoE module with eight experts, with one expert per GPU, to facilitate both data and expert parallelism. More experimental details are illustrated in Appendix A.

4.2 Results and Analysis

In comparison to the dense GPT2-Small model, the MoE version, which replaces every other FFN layer with an 8-expert MoE layer, increases the number of model parameters by 2.18 times. As

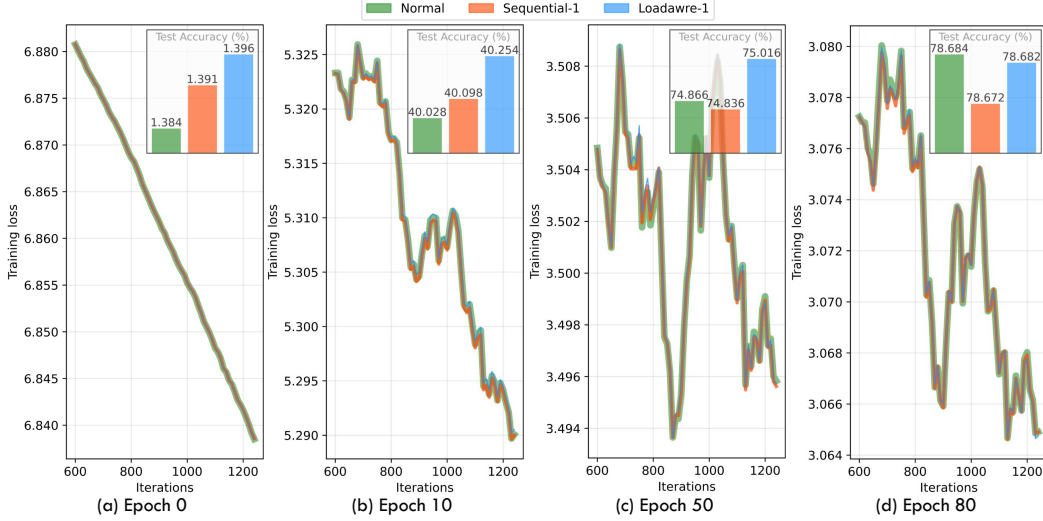


Figure 6: The training curve and final test accuracy of SwinV2-MoE model training across various epochs, with a fault occurring at the 600th iteration of these epochs. “Normal” represents the non-fault case, “Sequential-1” indicates using sequential expert selection with $K_{pec} = 1$, and “Loadaware-1” indicates using load-aware expert selection with $K_{pec} = 1$. The I_{ckpt} is set to 1.

shown in Figure 5 (a), our proposed PEC achieves a 54.2% reduction in non-redundant checkpoint size (no data-parallel duplication), attaining a checkpoint size comparable to the dense version, when $K_{pec} = 1$.

Nonetheless, merely decreasing non-redundant checkpoint size does not suffice for optimizing efficiency in distributed training. Using the prevalent checkpointing method [34, 32, 14] as the baseline (in Figure 5 (b) and (c)), where each data-parallel rank saves its entire training states, incurs a significant storage workload for every rank, even with our PEC (only reducing 15% checkpointing workload per distributed data-parallel rank when $K_{pec} = 1$). By employing data-parallel sharding strategies, we can effectively decrease the storage workload per rank due to the elimination of training state redundancy across ranks.

Adaptive Data-Parallel Sharding (ADPS). The PEC dynamic selection and the varying size of model modules lead to checkpointing workload fluctuations (indicated by the error bars) with both our proposed ADPS and fixed equal data-parallel sharding (EDPS). Notably, ADPS demonstrates a lower upper bound of fluctuation, while maintaining the same average workload as EDPS because of their same overall workload. Checkpointing is a synchronized operation across distributed ranks, hence the slowest rank determines the total duration. Therefore, ADPS outperforms EDPS due to its lower peak workload, with an 11.7% reduction in checkpointing time.

Asynchronous Checkpointing. Moreover, to attain complete overlap between the checkpointing and training process, the checkpointing duration must be less than the time of the forward and backward passes in an iteration, which is measured at 0.78 seconds as indicated by the red “overlap” line in Figure 5(c). When all experts are saved ($K_{pec} = 8$), none of the checkpointing strategies meet the threshold required for complete overlap. Only by applying PEC with partial expert saving ($K_{pec} < 8$) and data-parallel sharding can the checkpointing time be fully overlapped by the training workflow. This approach effectively reduces the O_{save} to zero and allows for the lowest I_{ckpt} and the minimal O_{lost} , thereby minimizing the overall checkpoint overhead O_{ckpt} , as discussed in Section 2.2.

Dynamic-K. In our investigation of the correlation between model quality and PLT detailed in Section 3.1 and Figure 2, we note that an increase in faults correlates with elevated PLT, which can detrimentally affect final model quality. Figure 5(d) demonstrates that our Dynamic-K strategy effectively maintains PLT below the pre-set threshold of 5% by dynamically modifying the K_{pec} parameter in response to escalating fault occurrences.

As we validate the efficacy of our PEC methods on GPT-MoE model training, we further extend our empirical investigation to the SwinV2-MoE model. As depicted in Figure 6, our PEC methods

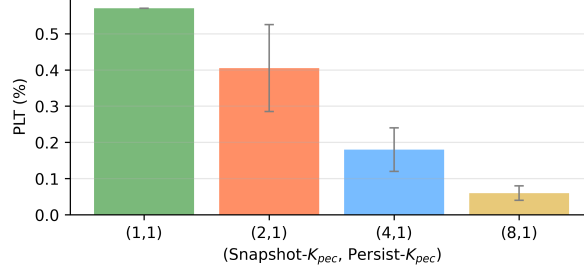


Figure 7: Correlation between PLT and various combinations of Snapshot-PEC’s K_{pec} and Persist-PEC’s K_{pec} on the training of the GPT-MoE model with a checkpoint interval (I_{ckpt}) of 1.

demonstrate robustness by maintaining training loss and test accuracy on par with the non-fault case across various stages of training (totally training 90 epochs, with the initial 0-9 epochs designated as the warm-up phase), irrespective of the fault occurrence timing.

Sequential versus Load-aware Expert Selection. While the test accuracy seemingly benefits from the load-aware expert selection (0.005%-0.18% higher than sequential expert selection), this approach incurs extra control and synchronization costs. Conversely, sequential expert selection, which exhibits only marginal accuracy differences when compared to the load-aware approach, is characterized by its minimal demand for additional control mechanisms. This renders sequential expert selection as the more viable option for practical applications and systems deployment.

Two-level PEC Saving and Recovery. We evaluate the effectiveness of a two-level PEC saving and recovery scheme in minimizing PLT. Our computing node with 8 GPUs is utilized to emulate a distributed environment comprising two nodes, each equipped with 4 GPUs. Given the higher speed of snapshot operations compared to persistence, we configure Persist- $K_{pec} = 1$ and experiment with varying Snapshot- K_{pec} values, as depicted in Figure 7. Compared with the baseline (Snapshot-1, Persist-1) setup, which corresponds to the PLT observed at $K_{pec} = 1$ in prior experiments, incrementing Snapshot- K_{pec} markedly decreases PLT, owing to the retrieval of partial experts from the in-memory snapshots on the non-fault node. Our results indicate that the implementation of two-level PEC can mitigate PLT escalations attributable to system faults, thereby ensuring the final quality of the trained model. For real-world deployments, we advise maximizing Snapshot- K_{pec} while ensuring that the snapshot’s size remains sufficiently small to allow for complete overlap with the training process, thereby avoiding an extension of normal training time. Moreover, setting Persist- $K_{pec} = 1$ to a minimal value is recommended to optimize efficiency.

5 Conclusion & Future Work

The advent of MoE models poses efficiency challenges for conventional fault-tolerant checkpointing methods, owing to the substantial escalation in model parameters. To enhance the fault-tolerance efficiency of MoE model training, we propose an innovative Partial Experts Checkpoint (PEC) mechanism, in conjunction with a PEC fault-tolerant system incorporating multiple optimization strategies. The empirical evaluations substantiate that our PEC methods markedly reduce the non-redundant checkpoint size without compromising model quality. Furthermore, our PEC fault-tolerant system substantially decreases the checkpointing workload for each data-parallel rank, consequently reducing the checkpointing duration and enabling full overlap with the training workflow.

In future work, we intend to delve into the influence of selective parameter update omissions on the model quality during the training phase. Many studies currently improve communication efficiency in distributed training by minimizing gradient updates through strategies like gradient compression [23, 58, 25] and gradient sparsification [53, 4, 2], resonating with the substance of our proposed PEC. Our exploration will focus on the importance of diverse updates, examining their impact across various modules such as Attention, FFN, and MoE, as well as updates stemming from disparate input tokens. These investigations are anticipated to inform the development of a more sophisticated and efficient fault-tolerant distributed training system.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.
- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [4] Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [7] Catello Di Martino, Zbigniew Kalbarczyk, Ravishankar K Iyer, Fabio Baccanico, Joseph Fullop, and William Kramer. Lessons learned from the analysis of system failures at petascale: The case of blue waters. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 610–621. IEEE, 2014.
- [8] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.
- [9] Assaf Eisenman, Kiran Kumar Matam, Steven Ingram, Dheevatsa Mudigere, Raghuraman Krishnamoorthi, Krishnakumar Nair, Misha Smelyanskiy, and Murali Annavaram. {Check-N-Run}: A checkpointing system for training deep learning recommendation models. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 929–943, 2022.
- [10] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.
- [11] Saurabh Gupta, Tirthak Patel, Christian Engelmann, and Devesh Tiwari. Failures in large scale systems: long-term measurement, analysis, and implications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2017.
- [12] Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*, 2021.
- [13] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.
- [14] Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5, 2023.
- [15] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.
- [16] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, et al. {MegaScale}: Scaling large language model training

- to more than 10,000 {GPUs}. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 745–760, 2024.
- [17] Richard Koo and Sam Toueg. Checkpointing and rollback-recovery for distributed systems. *IEEE Transactions on software Engineering*, (1):23–31, 1987.
 - [18] Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 5, 2023.
 - [19] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.
 - [20] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.
 - [21] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020.
 - [22] Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. *arXiv preprint arXiv:2105.13120*, 2021.
 - [23] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.
 - [24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9992–10002. IEEE, 2021.
 - [25] Ahmed M Abdelmoniem, Ahmed Elzanaty, Mohamed-Slim Alouini, and Marco Canini. An efficient statistical-based gradient compression technique for distributed training systems. *Proceedings of Machine Learning and Systems*, 3:297–322, 2021.
 - [26] Kiwan Maeng, Shivam Bharuka, Isabel Gao, Mark Jeffrey, Vikram Saraph, Bor-Yiing Su, Caroline Trippel, Jiyan Yang, Mike Rabbat, Brandon Lucia, et al. Understanding and improving failure tolerant training for deep learning recommendation with partial recovery. *Proceedings of Machine Learning and Systems*, 3:637–651, 2021.
 - [27] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.
 - [28] Jayashree Mohan, Amar Phanishayee, and Vijay Chidambaram. {CheckFreq}: Frequent,{Fine-Grained}{DNN} checkpointing. In *19th USENIX Conference on File and Storage Technologies (FAST 21)*, pages 203–216, 2021.
 - [29] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. Pipedream: generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM symposium on operating systems principles*, pages 1–15, 2019.
 - [30] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.
 - [31] Bogdan Nicolae, Jiali Li, Justin M Wozniak, George Bosilca, Matthieu Dorier, and Franck Cappello. Deepfreeze: Towards scalable asynchronous checkpointing of deep learning models. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 172–181. IEEE, 2020.
 - [32] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.

- [33] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- [34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [35] Aurick Qiao, Bryon Aragam, Bingjing Zhang, and Eric Xing. Fault tolerance in iterative-convergent machine learning. In *International Conference on Machine Learning*, pages 5220–5230. PMLR, 2019.
- [36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [37] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pages 18332–18346. PMLR, 2022.
- [38] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.
- [39] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–14, 2021.
- [40] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. {Zero-offload}: Democratizing {billion-scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564, 2021.
- [41] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In Marc’Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8583–8595, 2021.
- [42] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.
- [43] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [44] Liang Shen, Zhihua Wu, WeiBao Gong, Hongxiang Hao, Yangfan Bai, HuaChao Wu, Xinxuan Wu, Jiang Bian, Haoyi Xiong, Dianhai Yu, et al. Se-moe: A scalable and efficient mixture-of-experts distributed training and inference system. *arXiv preprint arXiv:2205.10034*, 2022.
- [45] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.
- [46] Siddharth Singh, Olatunji Ruwase, Ammar Ahmad Awan, Samyam Rajbhandari, Yuxiong He, and Abhinav Bhatele. A hybrid tensor-expert-data parallelism approach to optimize mixture-of-experts training. In *Proceedings of the 37th International Conference on Supercomputing*, pages 203–214, 2023.
- [47] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

- [48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [49] John Thorpe, Pengzhan Zhao, Jonathan Eyolfson, Yifan Qiao, Zhihao Jia, Minjia Zhang, Ravi Netravali, and Guoqing Harry Xu. Bamboo: Making preemptible instances resilient for affordable training of large {DNNs}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 497–513, 2023.
- [50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.
- [51] Yuxin Wang, Shaohuai Shi, Xin He, Zhenheng Tang, Xinglin Pan, Yang Zheng, Xiaoyu Wu, Amelie Chi Zhou, Bingsheng He, and Xiaowen Chu. Reliable and efficient in-memory fault tolerance of large language model pretraining. *arXiv preprint arXiv:2310.12670*, 2023.
- [52] Zhuang Wang, Zhen Jia, Shuai Zheng, Zhen Zhang, Xinwei Fu, TS Eugene Ng, and Yida Wang. Gemini: Fast failure recovery in distributed training with in-memory checkpoints. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 364–381, 2023.
- [53] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.
- [54] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.
- [55] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- [56] Baodong Wu, Lei Xia, Qingping Li, Kangyu Li, Xu Chen, Yongqiang Guo, Tieyao Xiang, Yuheng Chen, and Shigang Li. Transom: An efficient fault-tolerant system for training llms. *arXiv preprint arXiv:2310.10046*, 2023.
- [57] Yuanzhong Xu, HyoukJoong Lee, Dehao Chen, Hongjun Choi, Blake Hechtman, and Shibo Wang. Automatic cross-replica sharding of weight update in data-parallel training. *arXiv preprint arXiv:2004.13336*, 2020.
- [58] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

A Appendix

Table 1 summarizes the hyperparameters for training the GPT-MoE model. Table 2 summarizes the hyperparameters for training the Swin-MoE model

Table 1: Hyperparameters for GPT2-MoE.

Parameter	GPT2-MoE-Small
Num. layers	12
Embedding dim	768
Num. attention heads	12
Num. experts per layer	8
Num. parameters	323M
Context/sequence length	1K
Training tokens	180B
Warmup tokens	100M
Batch size	256
Learning rate	$3.0e-4$
LR-Scheduler	Inverse_sqrt
Capacity factor	1.00
MoE loss coefficient	0.01

Table 2: Hyperparameters for SwinV2-MoE.

Model	SwinV2-MoE
Image size	192×192
Window size	12×12
Embedding dim	96
Num. layers	[2, 2, 18, 2]
Num. attention heads	[3, 6, 12, 24]
Num. experts per layer	8
Num. parameters	157M
Batch size	1024
Epochs	90
Warmup epochs	10
Base LR	$1.25e-4$
Warmup LR	$1.25e-7$
Min LR	$1.25e-6$
Capacity factor	1.25
MoE loss coefficient	0.01