Check-N-Run: a Checkpointing System for Training Deep Learning Recommendation Models

(Accepted to NSDI 2022)

Assaf Eisenman[1], Kiran Kumar Matam[1], Steven Ingram[1], Dheevatsa Mudigere[1], Raghuraman Krishnamoorthi[1],Krishnakumar Nair[1], Misha Smelyanskiy[1], Murali Annavaram[1,2]

[1]Meta Platforms, Inc. [2]USC

1. Introduction

Deep learning has become extensively adopted in many production scale data center services. In particular, deep learning enabled recommendation systems power a wide variety of products and services. At Meta's datacenter fleet, for example, deep recommendation models consume more than 80% of the machine learning inference cycles and more than 50% of the training cycles. Similar demands can be found at other companies [3].

Typically, the accuracy of deep learning algorithms increases as a function of the model size and number of features. Because of their large size, these models also must be trained with massive datasets and run in a distributed fashion. Therefore, training recommendation models at production scale may take several days, even when training on highly optimized GPU clusters.

Given that the training runs span multiple GPU clusters over multiple days and weeks, there is an abundance of failures that a training run may encounter. These include network issues, hardware failures, system failures (e.g. out of memory), power outages, and code issues. Checkpointing is an important functionality to quickly recover from such failures for reducing the overall training time and ensure progress. Checkpoints are essentially snapshots of the running job state taken at regular intervals and stored in non-volatile memory. To recover from failure and resume training, the most recent checkpoint is loaded.

Checkpoints must meet several key criteria:

(1) **Accuracy:** They must be accurate to avoid training accuracy degradation. In other words, when a training run is restarted from a checkpoint, there should be no perceivable difference in the training accuracy or any other related metric. As has been stated in prior works on production scale recommendation models [4], even a tiny decrease of prediction accuracy would result in an unacceptable loss in user engagement and revenues. Hence, preserving accuracy is a constraint for checkpoint management in recommendation models.

(2) **Frequency:** Checkpoints need to be frequent for minimizing the re-training time (the gap between failure time and the most recent checkpoint timestamp) after resuming from a checkpoint. For instance, taking a checkpoint every 1000 batches of training data may lead to wasting time re-training those 1000 batches. Taking a checkpoint after 5000 batches leads to 5$\times$ more wasted work in the worst case. In the case of online training, the checkpoint frequency directly impacts how quickly the inference adapts in real time and its prediction accuracy.

(3) **Write Bandwidth:** Checkpoints at Meta, as well as in other industrial settings, are written to remote non-volatile memory to provide high availability (including replications) and scalable infrastructure. Writing multiple large checkpoints concurrently from different models that are being trained in parallel (e.g., thousands of checkpoints, each in the order of terabytes) to remote storage requires substantial network and storage bandwidths, which constitute a bottleneck and limit the checkpoint frequency. Hence, it is necessary to minimize the required bandwidth to enable frequent checkpoints.

(4) **Storage capacity:** Storing checkpoints at-scale requires hundreds of petabytes of storage capacity, with high-availability and short access times. Checkpoints at Meta are typically stored for many days, thus the number of stored checkpoints at a given time is reflected by the number of training jobs in that time period. While the last checkpoint per run is saved by default, it is often useful to keep several recent checkpoints (e.g. for debugging and transfer learning). As models keep getting larger and more complex, resulting in an ever increasing storage capacity demand, it is necessary to reduce the corresponding checkpoint size to minimize the required storage capacity for accommodating all checkpoints.

Unfortunately, standard compression algorithms such as Zstandard [2] are not useful enough for deep recommendation workloads. In our experience, we were able to reduce the checkpoint size and the associated write-bandwidth and storage capacity by at most 7% using Zstandard compression.

Based on the above challenges, we present Check-N-Run, a high-performance scalable checkpointing system, particularly tailored for recommendation systems. Check-N-Run's main goal is to significantly reduce the required write-bandwidth and storage capacity, without degrading accuracy. Our goal is to work within the accuracy degradation constraint set by business needs ($< 0.01\%$).

Check-N-Run builds on several techniques:

(1) **Differential checkpointing:** Check-N-Run utilizes differential checkpointing for reducing the checkpoint write bandwidth. This is a technique that is particularly well suited for recommendation models where only a small fraction of

the model parameters are updated after each iteration. This is a unique property of recommendation models. In traditional DNNs the entire model is updated after each iteration since gradients are computed for all the model parameters. Recommendation models, on the other hand, access and update only a small fraction of the model during each iteration. Differential checkpoints leverage this observation by tracking and storing the modified parts of the models.

(2) **Quantization:** Check-N-Run leverages quantization techniques to significantly reduce the size of checkpoints. This optimization reduces the required write bandwidth to storage, and the storage capacity. While quantization of model parameters during training may have a negative impact on accuracy, checkpointing has the advantage that quantization is only used to store the checkpoint, while full precision is used for training. The only time checkpoint quantization may impact training accuracy is when the quantized checkpoint is restored and de-quantized to resume training. Check-N-Run leverages this insight to maintain training accuracy within our strict bounds.

(3) **Decoupling:** To minimize the run time overhead and training stalls, Check-N-Run creates distributed snapshots of the model in multiple CPU host memories. That way, training on the GPUs can continue while Check-N-Run is optimizing and storing the checkpoints in separate processes running on the CPU in the background. Check-N-Run enables the frequent checkpointing of hundreds of complex production training jobs running in parallel over thousands of GPUs, each job training a very large model (in the order of terabytes).

The contributions in this paper include:
(1) To our knowledge, Check-N-Run is the first published checkpointing system that uses quantization and differential views for efficiently storing checkpoints of recommendation systems in non-volatile memory at-scale, demonstrated on real-world workloads.
(2) We design and evaluate a wide range of checkpoint quantization approaches to significantly reduce the consumed storage by 4-13×, without degrading the training accuracy.
(3) We introduce differential checkpoints, which store the modified part of the model, rather than storing the entire model. Differential checkpoints reduce the average write bandwidth by more than 50%, with no impact on accuracy.
(4) Finally, we demonstrate a heterogeneous checkpointing mechanism that combines differential checkpointing with quantization. Check-N-Run provides $6-17\times$ improvement in the required checkpointing write bandwidth, and $2.5\times -8\times$ less capacity, without sacrificing accuracy and run time.

The full details, including the design and evaluation, are elaborated in our full paper [1]. Link

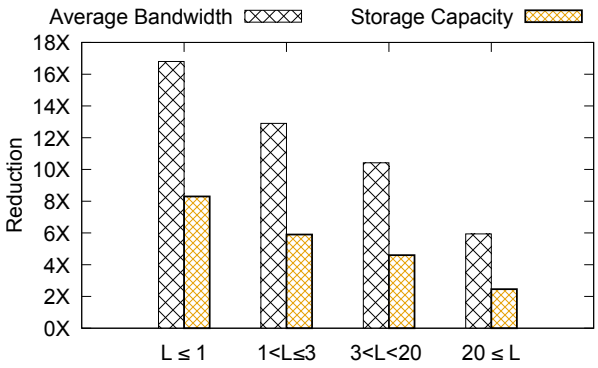

Figure 1. Overall reduction of the checkpoint average write bandwidth and storage capacity. L represents the number of times the training job had to resume from a checkpoint.

2. Results

Figure 1 presents the overall reduction in write bandwidth and storage capacity, when combining both quantization and differential checkpointing. When a training job is expected to resume from checkpoint no more than one time, Check-N-Run reduces the average consumed write bandwidth and maximum storage capacity by $17\times$ and $8\times$, respectively. Even in the not so common case of more than 20 failures, Check-N-Run reduces the average bandwidth by $6\times$ and the maximum storage capacity by $2.5\times$. Note that these savings are not linearly proportional to the chosen quantization bit-width due to the metadata structure. That structure includes the differential checkpoint index and quantization parameters. Metadata structure can be further optimized in future work.

References

[1] Check-N-Run: a checkpointing system for training deep learning recommendation models. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, Renton, WA, 2022. USENIX Association.

[2] Y. Collet and C. Turner. Smaller and faster data compression with zstandard. *http://www. rgoarchitects. com/Files/fallacies. pdf*, 2016.

[3] S. Hsia, U. Gupta, M. Wilkening, C.-J. Wu, G.-Y. Wei, and D. Brooks. Cross-stack workload characterization of deep recommendation systems. In *2020 IEEE International Symposium on Workload Characterization (IISWC)*, pages 157–168. IEEE, 2020.

[4] W. Zhao, J. Zhang, D. Xie, Y. Qian, R. Jia, and P. Li. Aibox: Ctr prediction model training on a single node. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 319–328, 2019.