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

Assaf Eisenman, Kiran Kumar Matam, Steven Ingram, Dheevatsa Mudigere, Raghuraman Krishnamoorthi, Krishnakumar Nair, Misha Smelyanskiy, Murali Annavaram
Recommendation Models are Important

• Use cases include:
  • E-commerce marketplaces
  • Social media platforms
  • Entertainment services

• Consumes most of AI compute cycle at Meta
  • > 50% of training compute cycle
  • > 80% of inference compute cycles
Recommendation Model Architecture

```
Top MLP

Feature Interaction

<table>
<thead>
<tr>
<th>Bottom MLP</th>
<th>Emb Table</th>
<th>Emb Table</th>
<th>Emb Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense Features</td>
<td>Sparse Feature</td>
<td>Sparse Feature</td>
<td>Sparse Feature</td>
</tr>
</tbody>
</table>
```

High Performance Training at Meta

- Reader Cluster: Train Dataset & Reader Threads
- Training Cluster: Training GPUs
  - Host CPUs: Get model state, Quantize, Diff increments, Send to checkpoint store
- Checkpoint Cluster: Checkpoint Store
  - Checkpoint Data

Flow: Training Data → Checkpoint Data
The Criticality of Checkpointing

• Failure recovery (ensure progress)
• Migrating training jobs
• Publishing snapshots
• Transfer learning
Checkpoint Challenges

• Accuracy
• Frequency
• Write bandwidth
• Storage capacity
Check-n-Run

• Goal: a checkpointing system that significantly reduces the required write-bandwidth and storage capacity, without degrading accuracy
• What to Checkpoint?
• Decoupled Checkpointing
• Reducing write-bandwidth (WB) and storage capacity
Checkpointing Workflow

- **Training Dataset**
- **Master Reader**
- **Worker Reader**
- **Checkpoint Storage**
- **Trainer Node**

Checkpoints are stored in the **Checkpoint Storage** for future use.
Reducing WB with Differential Checkpointing

• Motivation: model accesses are sparse
Approaches for Differential Checkpointing

- One-Shot Differential Checkpoint
- Consecutive Incremental Checkpoint
- Intermittent Differential Checkpoint

**Write Bandwidth:**
- One-shot baseline
- Intermittent baseline
- Consecutive increment

**Storage:**
- One-shot baseline
- Intermittent baseline
- Consecutive increment
Checkpoint Quantization

• Compress checkpoint without degrading training accuracy

• Approaches:

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.21</td>
<td>-0.31</td>
<td>0.03</td>
<td>0.01</td>
<td>-0.05</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Uniform:

(-0.31) (-0.05, 0.21) (0.03, 0.01)

Non-Uniform:

(-0.31) (-0.05) (0.03, 0.21) 0.01
Comparing Quantization Strategies

- Uniform quantization
- Non-uniform quantization using k-means
- Adaptive uniform quantization
Quantization Bit-width Selection

- Quantization error may accumulate
- Select bit-width based on the probability of a click
Overall Reduction

The image shows a bar graph illustrating the overall reduction in bandwidth and storage capacity across different categories: $L \leq 1$, $1 < L \leq 3$, $3 < L \leq 20$, and $20 \leq L$. The y-axis represents the reduction, while the x-axis categorizes the bandwidth and storage capacity.
Summary

• The checkpointing of large recommendation systems at scale is challenging

• Check-n-run:
  • High performance checkpointing
  • Significantly reduces the required write-bandwidth and storage capacity

• Questions? aeisenman@fb.com