Multi-Resource Interleaving for Deep Learning Training

Yihao Zhao, Yuanqiang Liu, Yanghua Peng, Yibo Zhu, Xuanzhe Liu, Xin Jin
Deep Learning Training

Deep Learning (DL) is popular
• DL training becomes an important workload in enterprises’ clusters

DL training uses **multiple resource types**
• DL training is **iterative**
• DL training is **staged** and each stage mainly uses a specific resource type
DL Training in Clusters

Users and DL training jobs

Goal:
- Minimize finish time
- Maximize resource utilization
- ...

Current DL Scheduler:
- Most allocate GPUs to a job exclusively
- Some explore only GPU sharing

Cluster

Node 1

Node N

Scheduler

Goal: Miss the opportunity of multi-resource sharing!
DL Training in Clusters

Users and DL training jobs

Cluster

Scheduler

Challenges of multi-resource sharing

- Reduce interference among shared DL jobs
- Improve both job and cluster efficiency
Our approach (Muri)

A DL cluster scheduler that utilizes **MUlti-Resource Interleaving** to improve job and cluster efficiency

**Multi-resource interleaving**
- Pack jobs on the same set of resources by *interleaving stages in time*
- Reduce interference among shared jobs

**Blossom-based scheduler**
- Assign *sharing groups and patterns* to maximize interleaving efficiency
- Improve both job and cluster efficiency
Muri Architecture

Submit Job

Muri Scheduler

Resource Profiler  Job Scheduler  Worker Monitor

Muri Executor

Muri Executor

Muri Executor
Muri Architecture

Submit Job

Muri Scheduler

Job Queue

Resource Profiler

Job Scheduler

Worker Monitor

Muri Executor

Muri Executor

...
Muri Architecture

Submit Job

Job Queue

Resource Profiler

Job Scheduler

Worker Monitor

Muri Scheduler

Muri Executor

Muri Executor
Muri Architecture

Submit Job

Job Queue

Muri Scheduler

Resource Profiler
Job Scheduler
Worker Monitor

Muri Executor

...
Muri Architecture

Submit Job

Job Queue

Resource Profiler
Job Scheduler
Worker Monitor

Muri Scheduler

Muri Executor
CPU GPU Storage Network

Muri Executor
CPU GPU Storage Network
Multi-Resource Sharing

**Space sharing**

- **High interference** among shared jobs leads to longer iteration duration
  - At every moment, each resource type on one machine can be used by multiple jobs

- **Low interference** among shared jobs brings shorter iteration duration
  - At every moment, each resource type on one machine can be used by only one job

**Time sharing**

Iteration duration = 9

Lengthened due to interference

Iteration duration = 4
Muri: Multi-Resource Interleaving

Muri exploits fine-grained multi-resource interleaving in time

• Staged pattern of DL training brings inherent stages to interleave
• Iterative pattern of DL training enables low-overhead scheduling decision for interleaving

Low interference among shared jobs brings shorter iteration duration
• At every moment, each resource type on one machine can be used by only one job
Multi-Resource Interleaving vs. Pipelining

**Pipelining**

Overlap multiple resources **intra-job**

Throughput when job A and B are run separately: \( \frac{1}{5.5+5.5} = \frac{1}{11} \) iterations/s

**Multi-resource interleaving**

Overlap multiple resources **inter-job**

Throughput when job A and B are interleaved: \( \frac{1}{\max(5.5,6.5)} = \frac{1}{6.5} \) iterations/s

1.7× higher throughput!
**Interleaving efficiency** represents how perfect a grouping plan can overlap the resource usage of the jobs.

\[
\gamma = \frac{1}{k} \sum_{j=0}^{k-1} \frac{\sum_{i=0}^{p-1} t_{ij}}{T}
\]

Iteration duration can be estimated by

\[
T = \sum_{j=0}^{k-1} \max_{i=0}^{p-1} t_{i(i+j)\mod k}
\]

- \( k \): the number of resource types
- \( p \): the number of jobs in one group
- \( t_{ij} \): the duration that job \( i \) uses resource \( j \)
Muri Scheduler: Select Jobs to Interleave

Formulate as a **maximum weighted matching problem** for two resource types

- **Node**: a group of jobs that are interleaved
- **Edge**: interleave the jobs in the two nodes
- **Edge weight**: the interleaving efficiency
- **Matching**: a grouping plan

---

**Optimal** for two resource types
Muri Scheduler: Select Jobs to Interleave

For more than two resource types…

- Maximum weighted k-uniform hypergraph matching
- NP-Hard!
Muri Scheduler: Select Jobs to Interleave

Multi-round heuristic algorithm for multiple resource types

Multi-round for multiple resource types
Muri: Other Design Details

Handle multi-GPU jobs
- Only group jobs with the same GPU requirement as intra-job synchronization brings slowdown

Optimize interleaving efficiency
- (a) has interleaving efficiency $\gamma \approx 0.5$
- (b) has interleaving efficiency $\gamma = 0.4$
- Enumerate all orderings of a group as the ordering of jobs affects the interleaving efficiency

Optimize average JCT
- Assign a priority to each job
- SRSF when job durations are known
- 2D-LAS when job durations are unknown
Evaluation

- Implementation: ~7,000 LOC
  - PyTorch 1.8.1
  - CUDA 11.1
- Testbed
  - 64-GPU cluster, NVIDIA Tesla V100 GPU
- Traces
  - Philly Trace from Microsoft [Jeon et al. 2019]
- Models
  - CV: ResNet18, ShuffleNet, VGG16, VGG19
  - NLP: Bert, GPT-2
  - RL: A2C, DQN
Testbed Experiments: Overall Performance

8 nodes w/ 8 GPUs each (V100)
400 DL jobs submitted over 10s days

<table>
<thead>
<tr>
<th></th>
<th>SRTF</th>
<th>SRSF</th>
<th>Muri-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized JCT</td>
<td>2.12</td>
<td>2.03</td>
<td>1</td>
</tr>
<tr>
<td>Normalized Makespan</td>
<td>1.56</td>
<td>1.59</td>
<td>1</td>
</tr>
<tr>
<td>Normalized 99th%-ile JCT</td>
<td>3.31</td>
<td>3.82</td>
<td>1</td>
</tr>
</tbody>
</table>

Job durations are known

<table>
<thead>
<tr>
<th></th>
<th>Tiresias</th>
<th>Themis</th>
<th>Muri-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized JCT</td>
<td>2.59</td>
<td>3.56</td>
<td>1</td>
</tr>
<tr>
<td>Normalized Makespan</td>
<td>1.48</td>
<td>1.47</td>
<td>1</td>
</tr>
<tr>
<td>Normalized 99th%-ile JCT</td>
<td>2.54</td>
<td>2.60</td>
<td>1</td>
</tr>
</tbody>
</table>
## Testbed Experiments: Overall Performance

8 nodes w/ 8 GPUs each (V100)
400 DL jobs submitted over 10s days

<table>
<thead>
<tr>
<th></th>
<th>SRTF</th>
<th>SRSF</th>
<th>Muri-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized JCT</td>
<td>2.12</td>
<td>2.03</td>
<td>1</td>
</tr>
<tr>
<td>Normalized Makespan</td>
<td>1.56</td>
<td>1.59</td>
<td>1</td>
</tr>
<tr>
<td>Normalized 99(^{th})%-ile JCT</td>
<td>3.31</td>
<td>3.82</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Tiresias</th>
<th>Themis</th>
<th>Muri-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized JCT</td>
<td>2.59</td>
<td>3.56</td>
<td>1</td>
</tr>
<tr>
<td>Normalized Makespan</td>
<td>1.48</td>
<td>1.47</td>
<td>1</td>
</tr>
<tr>
<td>Normalized 99(^{th})%-ile JCT</td>
<td>2.54</td>
<td>2.60</td>
<td>1</td>
</tr>
</tbody>
</table>

Job durations are unknown
Testbed Experiments: Overall Performance

8 nodes w/ 8 GPUs each (V100)
400 DL jobs submitted over 10s days

Job efficiency
• > 2× faster average job completion time
• > 2.5× faster tail job completion time

Cluster efficiency
• > 1.4× faster makespan

<table>
<thead>
<tr>
<th></th>
<th>SRTF</th>
<th>SRSF</th>
<th>Muri-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized JCT</td>
<td>2.12</td>
<td>2.03</td>
<td>1</td>
</tr>
<tr>
<td>Normalized Makespan</td>
<td>1.56</td>
<td>1.59</td>
<td>1</td>
</tr>
<tr>
<td>Normalized 99th%-ile JCT</td>
<td>3.31</td>
<td>3.82</td>
<td>1</td>
</tr>
</tbody>
</table>

Job durations are known

<table>
<thead>
<tr>
<th></th>
<th>Tiresias</th>
<th>Themis</th>
<th>Muri-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized JCT</td>
<td>2.59</td>
<td>3.56</td>
<td>1</td>
</tr>
<tr>
<td>Normalized Makespan</td>
<td>1.48</td>
<td>1.47</td>
<td>1</td>
</tr>
<tr>
<td>Normalized 99th%-ile JCT</td>
<td>2.54</td>
<td>2.60</td>
<td>1</td>
</tr>
</tbody>
</table>

Job durations are unknown
Job durations are unknown

Higher utilization

- 36% higher average GPU utilization
- 30% higher average CPU utilization
- Other resources in our paper!
Trace-Driven Simulations

Job durations are unknown

(a) Average JCT

(b) Makespan

(c) 99th%-ile JCT
Trace-Driven Simulations

Job durations are unknown

(a) Average JCT

(b) Makespan

(c) 99th%-ile JCT
Trace-Driven Simulations

Job durations are unknown

Results: improve up to $6.1 \times$ avg. JCT, $1.5 \times$ makespan, and $5.4 \times$ tail JCT
More Experiments in our Paper

- Performance when job durations are known
- More detailed metrics
- Analysis of Muri
  - Impact of designs
  - Impact of workload distributions
  - Impact of inaccurate profiling
- …
Conclusion

• Muri: a multi-resource cluster scheduler for DL workloads
  • Introduce multi-resource interleaving to share jobs in time
  • Utilize a Blossom-based scheduling algorithm to maximize the interleaving efficiency
• Muri improves average JCT by up to $6.1\times$ and makespan by up to $1.6\times$

Open-sourced at [https://github.com/Rivendile/Muri](https://github.com/Rivendile/Muri)

Thanks!

zhaoyh98@pku.edu.cn