Zico

Efficient GPU Memory Sharing for Concurrent DNN Training

Gangmuk Lim*, Jeongseob Ahn§, Wencong Xiao†, Youngjin Kwon‡, Myeongjae Jeon*

* UNIST, § Ajou University, † Alibaba Group, ‡ KAIST
GPU Utilization in DNN Training

DNN training jobs require GPU

GPU core is often under-utilized

[Gandiva OSDI 18, Philly ATC 19, Salus MLSys 20]
Working set in concurrent training easily exceeds GPU memory

Existing GPU Sharing Solution

e.g. NVIDIA MPS, NVIDIA MIG, Salus
Cyclic Memory Usage Pattern

DNN training job shows cyclic memory usage pattern
Memory Sharing Opportunity

Concurrent training jobs

Dynamic memory sharing

Coordination

Efficiently reducing the system-wide memory footprint
Zico Overview

Coordinate executions of co-located jobs

Memory Manager

Scrooge Scheduler

Monitor

Scheduler

GPU Memory

Job A

Job B

Shared

Job A

Job B

Memory usage pattern

: Job A

: Job B

: Total
Contributions

Safe and efficient memory management
Handling asynchrony between CPU and GPU
Preventing early allocation

Memory-aware scheduling
Minimizing time delay while maximizing throughput
Widely applicable (identical jobs, non-identical jobs)
Asynchrony between CPU and GPU

Kernel Launch

Kernel Execution

Has GPU finished kernel ‘K_1’?

I have no idea

CPU

GPU

Time

K_1

K_2

K_3

K_4

K_1

K_2

K_3

K_4

CPU kernel launch

GPU kernel launch

I have no idea

CPU

GPU
Synchronization between CPU and GPU

Kernel Launch

Kernel Execution

Has GPU finished kernel ‘K₁’?

Yes! Release unused memory for K₁
Early Memory Allocation

CPU kernel launch speed > GPU execution speed

Early memory allocations for kernels which have not started its execution yet

*Increasing memory consumption unnecessarily*
Controlling Inflight Kernel

Controlling the number of inflight kernel

*Preventing early allocation*
Roadmap

- Coordinate executions of co-located jobs
- Minimize time delay under memory budget

Memory usage pattern:
- Job A
- Job B

GPU Memory:
- Job A
- Job B
- Total
Naïve Scheduling

Backward starting point of Job A

Forward starting point of Job B

Is there any issue?

Wavelet: Efficient DNN Training with Tick-Tock Scheduling, MLSys 21
Limitation of Naïve Scheduling

Memory over-subscription → Hurting the throughput
What We Want to Achieve

😊 Proper coordination ➔ Preventing over-subscription
😊 Memory over-subscription ➔ Hurting the throughput

![Memory usage over time with Job A and Job B](image)

Time

Memory usage

GPU Memory
Scrooge Scheduler

Current time

Future

Time

Job A

Job B

Memory usage

GPU Memory
Scrooge Scheduler

Scrooge decides when to initiate the forward pass of Job B

Current time

Time

GPU Memory

Future

Job A

Job B

Predict the future memory pattern
Scrooge Scheduler

Total estimated memory consumption > GPU Memory

Fail → Delay job B’s forward pass!

Predict the future memory pattern

Evaluate memory oversubscription
Scrooge Scheduler

- **Memory usage**
  - Current time
  - Future
  - Total estimated memory consumption > GPU Memory

- **GPU Memory**
- **Future**
  - Predict the future memory pattern
  - Evaluate memory oversubscription

**Fail ➔ Delay job B’s forward pass!**
Scrooge Scheduler

No over-subscription will occur. Schedule a new iteration of Job B!

Predict the future memory pattern
Evaluate memory oversubscription

Current time

Time

Job A
Job B

Scrooge Scheduler
Memory usage

GPU Memory

Future

No over-subscription will occur
Schedule a new iteration of Job B!
Scrooge Scheduler

Repeat the same process for job A
Evaluation

• Machine
  • GPU: Tesla V100 GPU, RTX 2080 Ti GPU
  • CPU: 3.8 GHz Intel Xeon(R)Gold 5222 4 CPU cores
  • RAM: 64 GB

• Benchmark
  • NASNet, ResNet-110, ResNet-50, GNMT, BERT, RHN

• Policies in comparison
  • Temporal: Ideal temporal sharing (no job switching overhead)
  • Spatial: NVIDIA MPS (no dynamic memory sharing)

• Base framework: TensorFlow v1.13.1
Throughput: Identical Jobs

Zico/Temporal: 1.03x ~ 1.6x
Throughput: Identical Jobs

Zico/MPS\textsubscript{(no over-subscription)}: Similar throughput
Throughput: Identical Jobs

Zico/MPS\textsubscript{(no over-subscription)}: Similar throughput

Zico/MPS\textsubscript{(over-subscription)}: 4.7x
Throughput: Identical Jobs

- **NASNet**
  - Zico
  - Temporal
  - MPS

- **ResNet-110**
  - Zico
  - Temporal
  - MPS

- **ResNet-50**
  - Zico
  - Temporal
  - MPS

- **GNMT**
  - Zico
  - Temporal
  - MPS

- **RHN**
  - Zico
  - Temporal
  - MPS

- **BERT**
  - Zico
  - Temporal
  - MPS
Throughput: Identical Models

![ResNet-110 Throughput Graph](image1)

![GNMT Throughput Graph](image2)

Throughput (sample/sec) vs. Batch size for ResNet-110 and GNMT models.
Throughput: Identical Models

Temporal doesn’t fully utilize GPU
Zico always outperforms Temporal!

ResNet-110

Throughput (sample/sec)

Temporal doesn’t fully utilize GPU
Zico always outperforms Temporal!

GNMT

Temporal doesn’t fully utilize GPU
Zico always outperforms Temporal!

Zico / Temporal: 10-36%

Zico / Temporal: 23-54%
Throughput: Identical Models

MPS suffers from memory over-subscription

Zico successfully co-locates two jobs w/o over-subscription

Throughput (sample/sec) vs. Batch size for ResNet-110 and GNMT models.
Throughput: Non-identical Jobs

Note: In non-identical jobs experiment, MPS is set to always over-subscribe the memory.

Zico/Temporal: 1.3x
Zico/MPS\textsubscript{(over-subscription)}: 5.8x
Throughput: Non-identical Jobs
Zico successfully co-locates non-identical models!

More improvement when low utilization model is co-located
Throughput: Non-identical Jobs

High util. + Low util.
(NASNet) (GNMT)

150%
100%
50%
0%

Zico Temporal MPS

1.5x 5.5x

High util. + Low util.
(NASNet) (ResNet-110)

100%
50%
0%

Zico Temporal MPS

1.2x 7.5x

NASNet GNMT

NASNet ResNet-110
BERT + BERT
(budge: 32GB)

Identical model co-location

Memory usage

Time progress
Scheduling Example

BERT + BERT
(budge: 32GB)

NASNet + ResNet-110
(budge: 11GB)
Summary

• Zico is the first introducing memory-aware scheduler

• Zico proposes widely applicable GPU sharing techniques for training