Zico

Efficient GPU Memory Sharing for Concurrent DNN Training

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

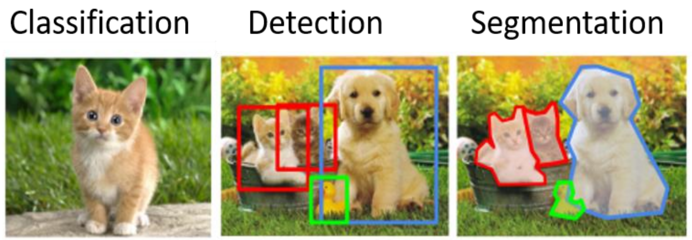
* UNIST, § Ajou University, † Alibaba Group, ‡ KAIST

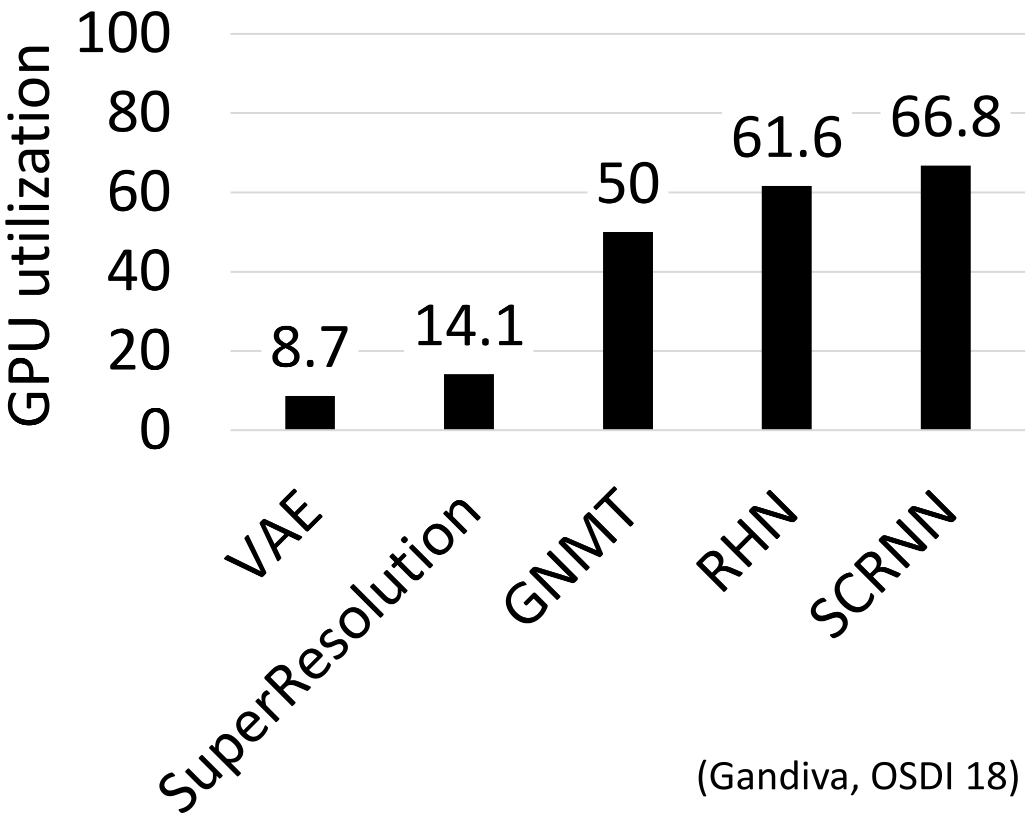
GPU Utilization in DNN Training

DNN training jobs require GPU

GPU core is often under-utilized

[Gandiva OSDI 18, Philly ATC 19, Salus MLSys 20]

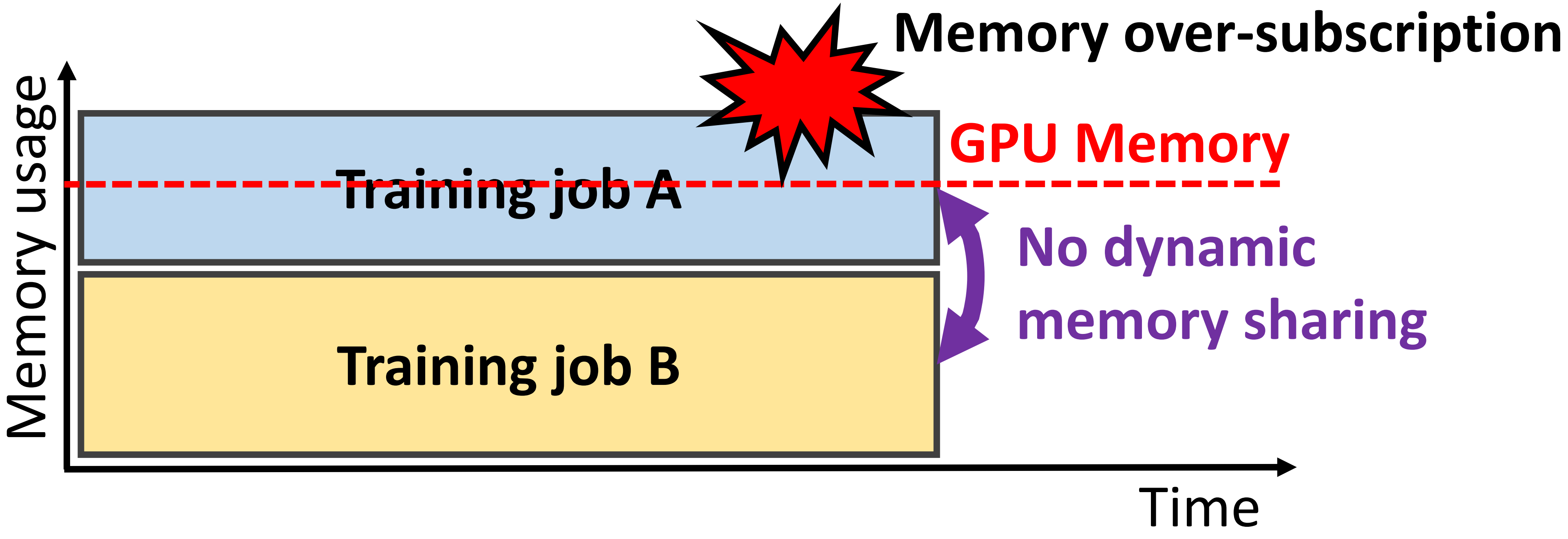




(Gandiva, OSDI 18)

Existing GPU Sharing Solution

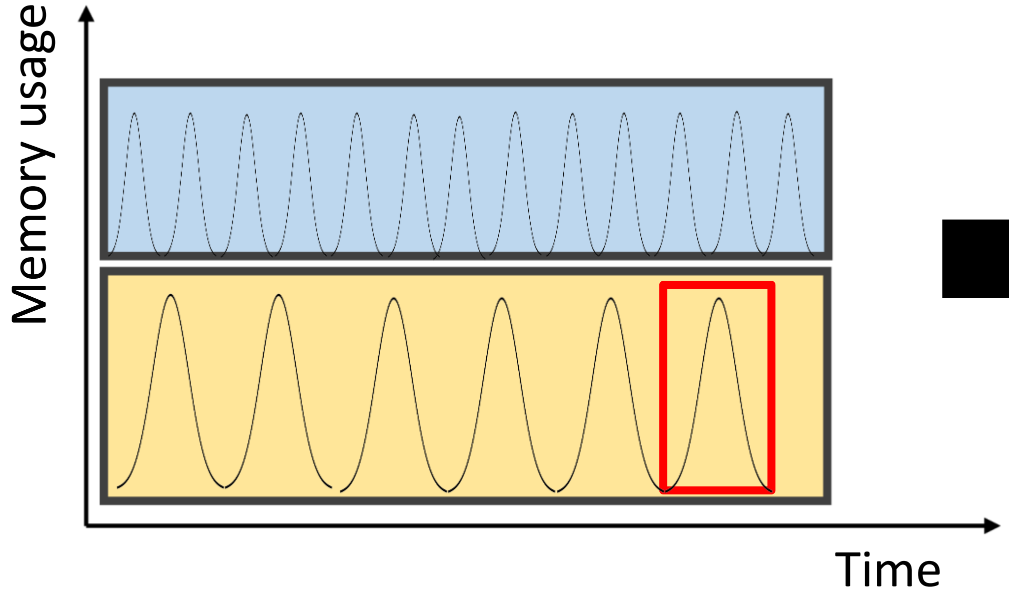
Working set in concurrent training easily exceeds GPU memory

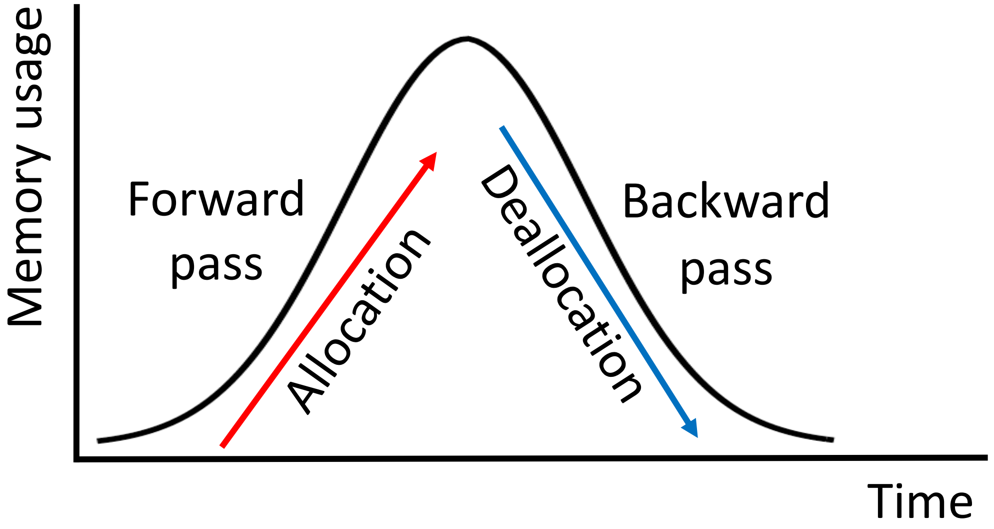


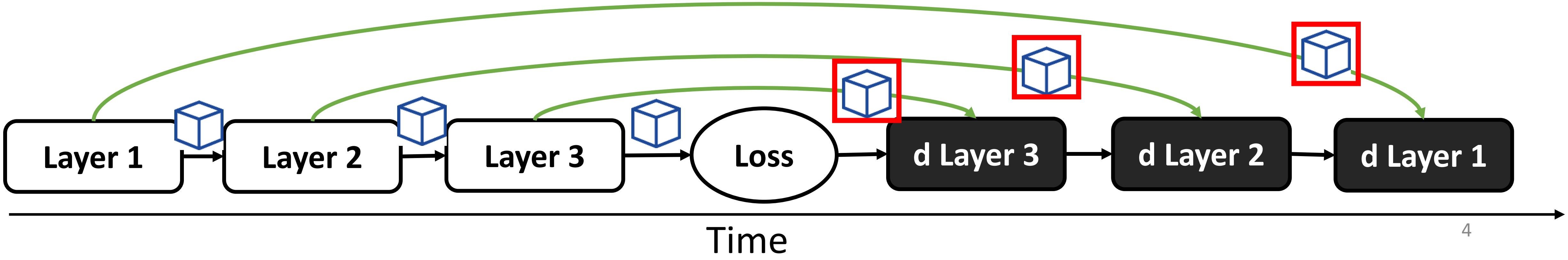
e.g. NVIDIA MPS, NVIDIA MIG, Salus

Cyclic Memory Usage Pattern

DNN training job shows cyclic memory usage pattern

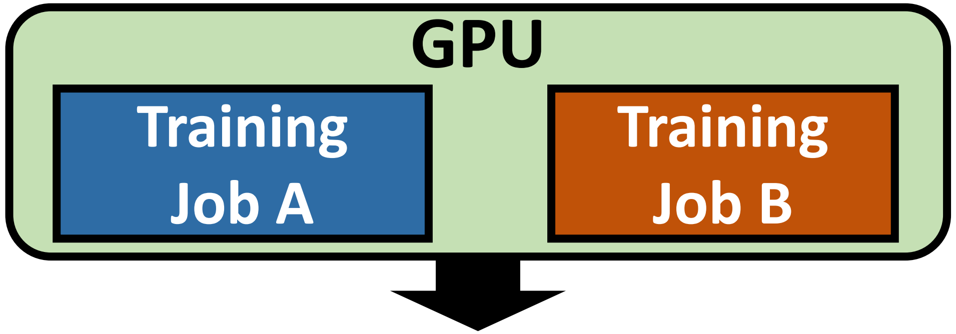






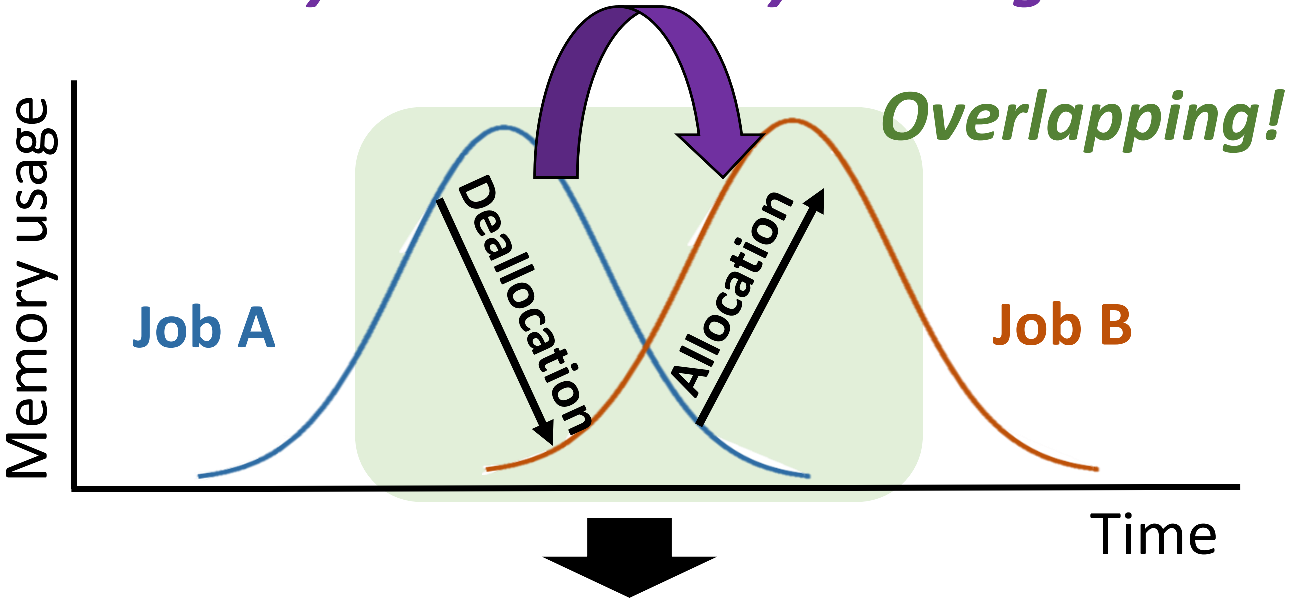
Memory Sharing Opportunity

Concurrent training jobs



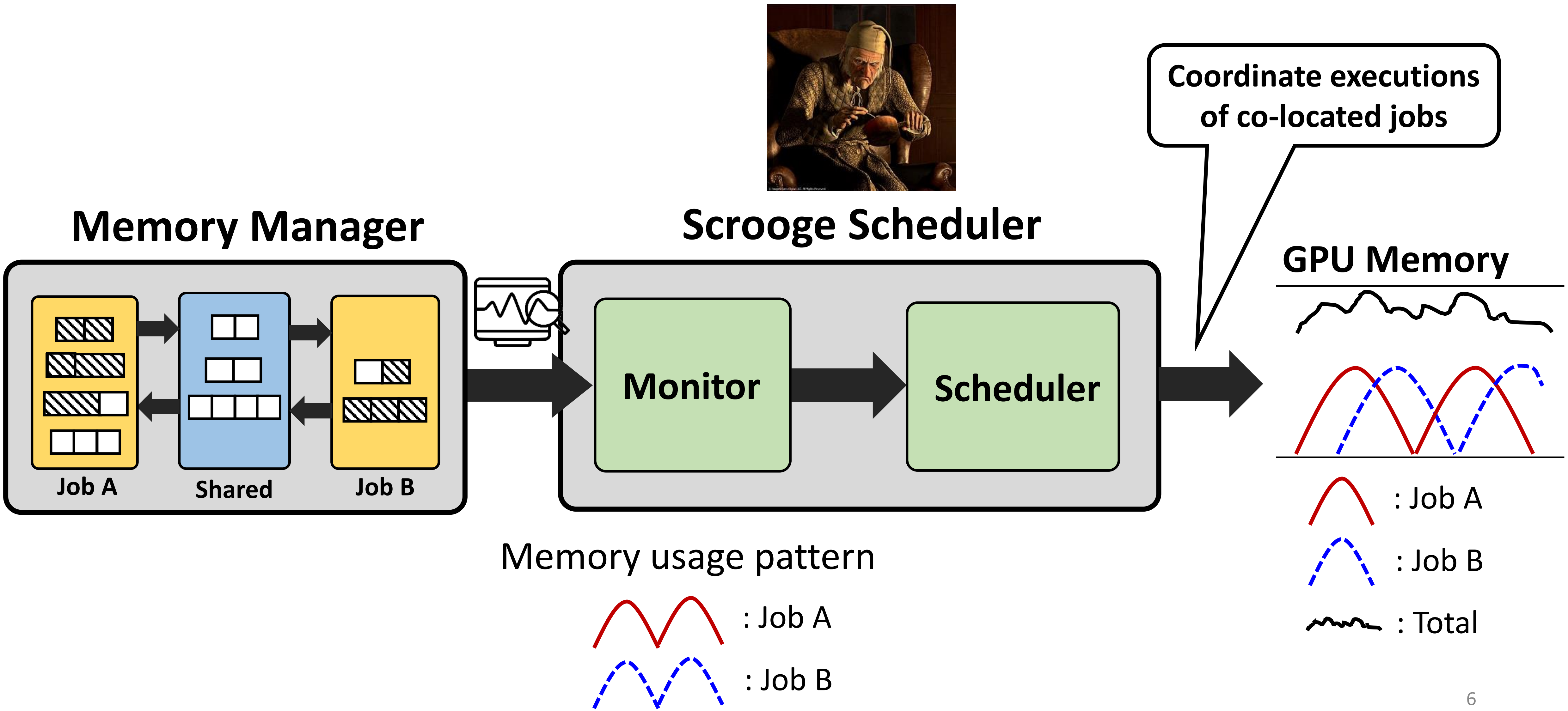
Dynamic memory sharing

Coordination



Efficiently reducing the system-wide memory footprint

Zico Overview



Coordinate executions of co-located jobs

Memory Manager

Job A

Shared

Job B

Scrooge Scheduler

Monitor

Scheduler

GPU Memory

: Job A

: Job B

: Total

Memory usage pattern

: Job A

: Job B

Contributions

Safe and efficient memory management

Handling asynchrony between CPU and GPU

Preventing early allocation

Memory-aware scheduling

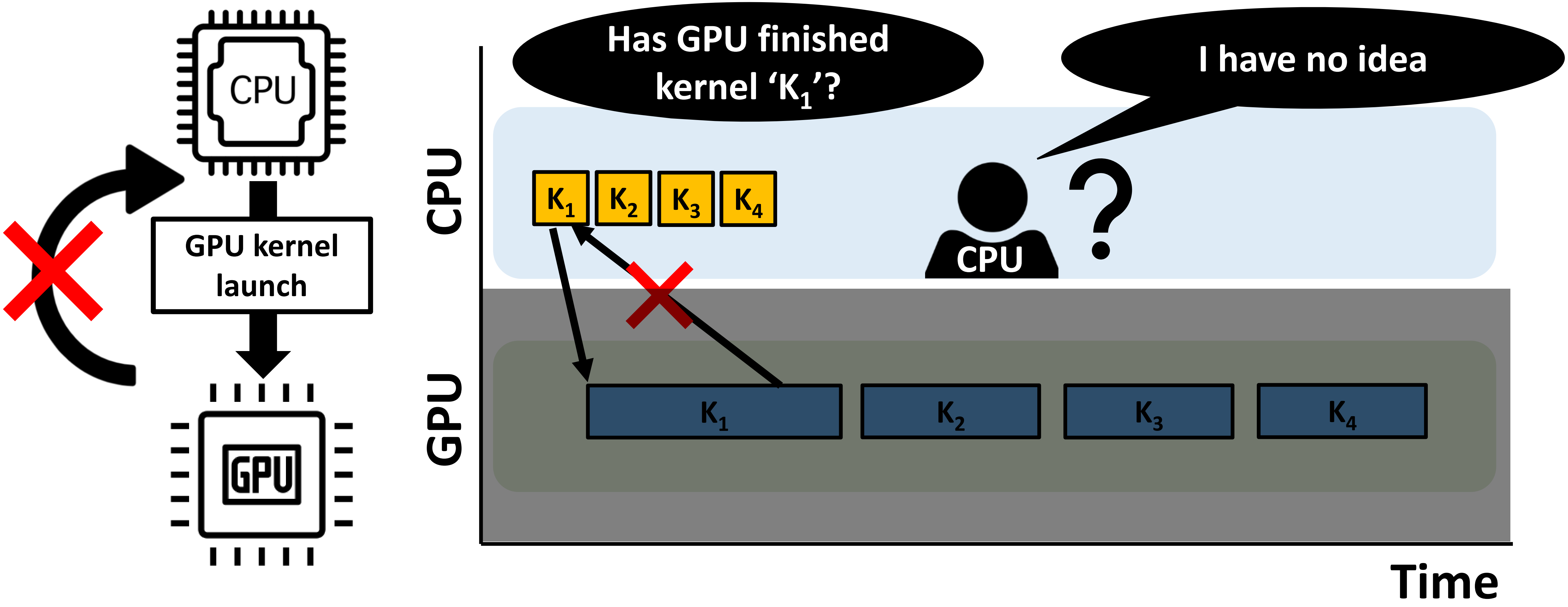
Minimizing time delay while maximizing throughput

Widely applicable (identical jobs, non-identical jobs)

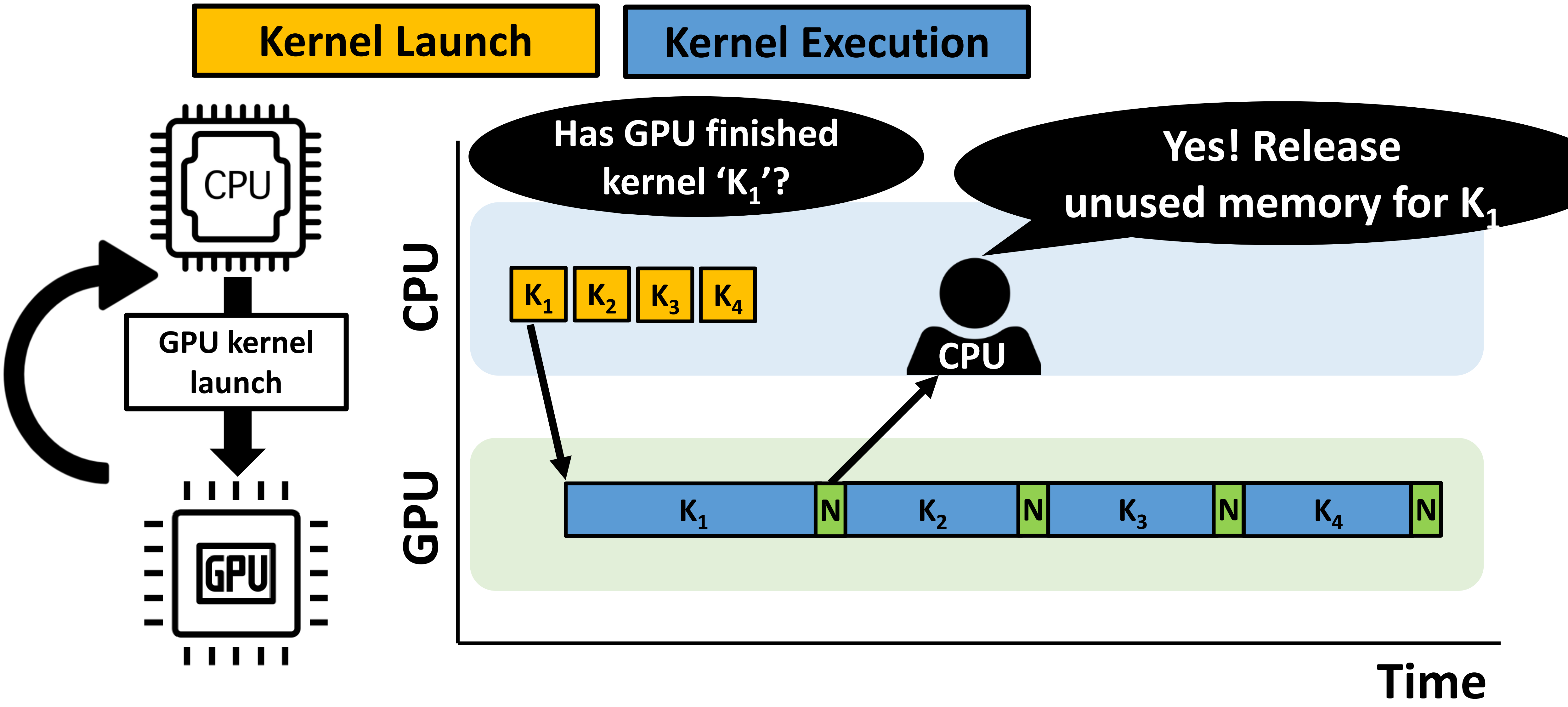
Asynchrony between CPU and GPU

Kernel Launch

Kernel Execution



Synchronization between CPU and GPU

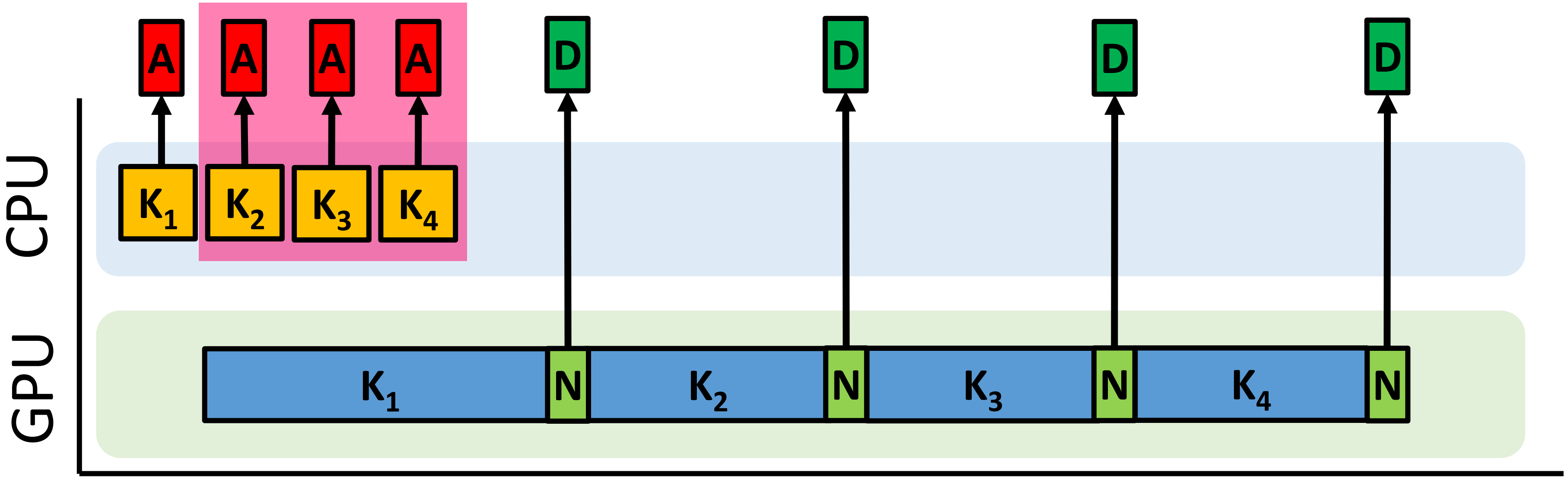

Kernel Launch
Kernel Execution
CPU
GPU kernel launch
GPU
Has GPU finished kernel ‘K_1’?
Yes! Release unused memory for K_1
CPU
K_1 K_2 K_3 K_4
CPU
GPU
K_1 N K_2 N K_3 N K_4 N
Time

Early Memory Allocation

CPU kernel launch speed > GPU execution speed

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

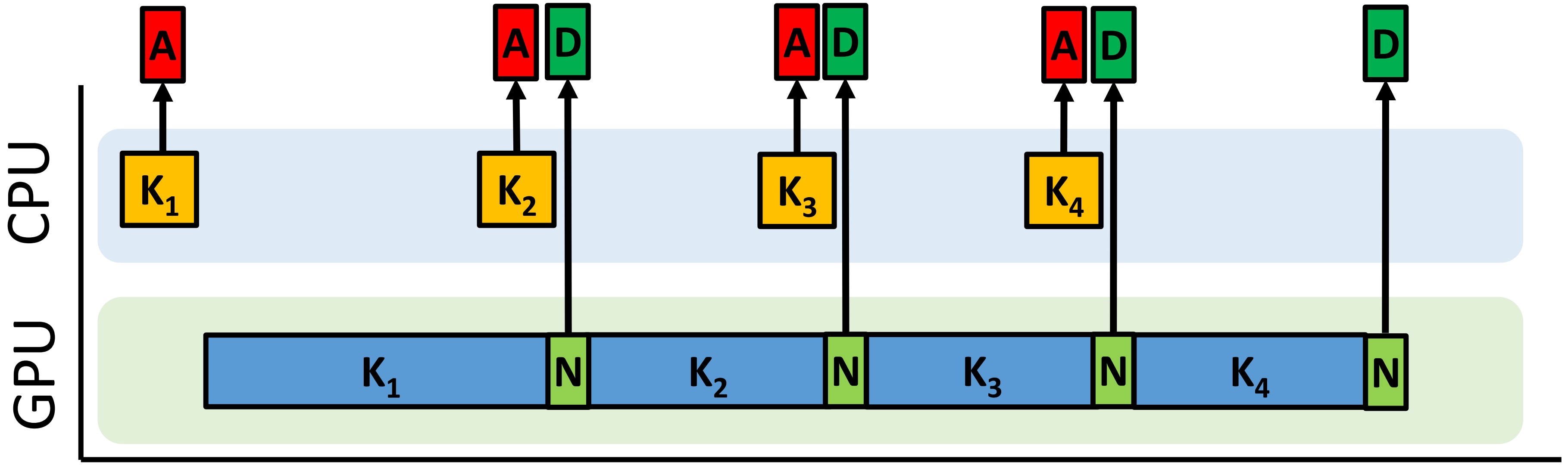
Increasing memory consumption unnecessarily



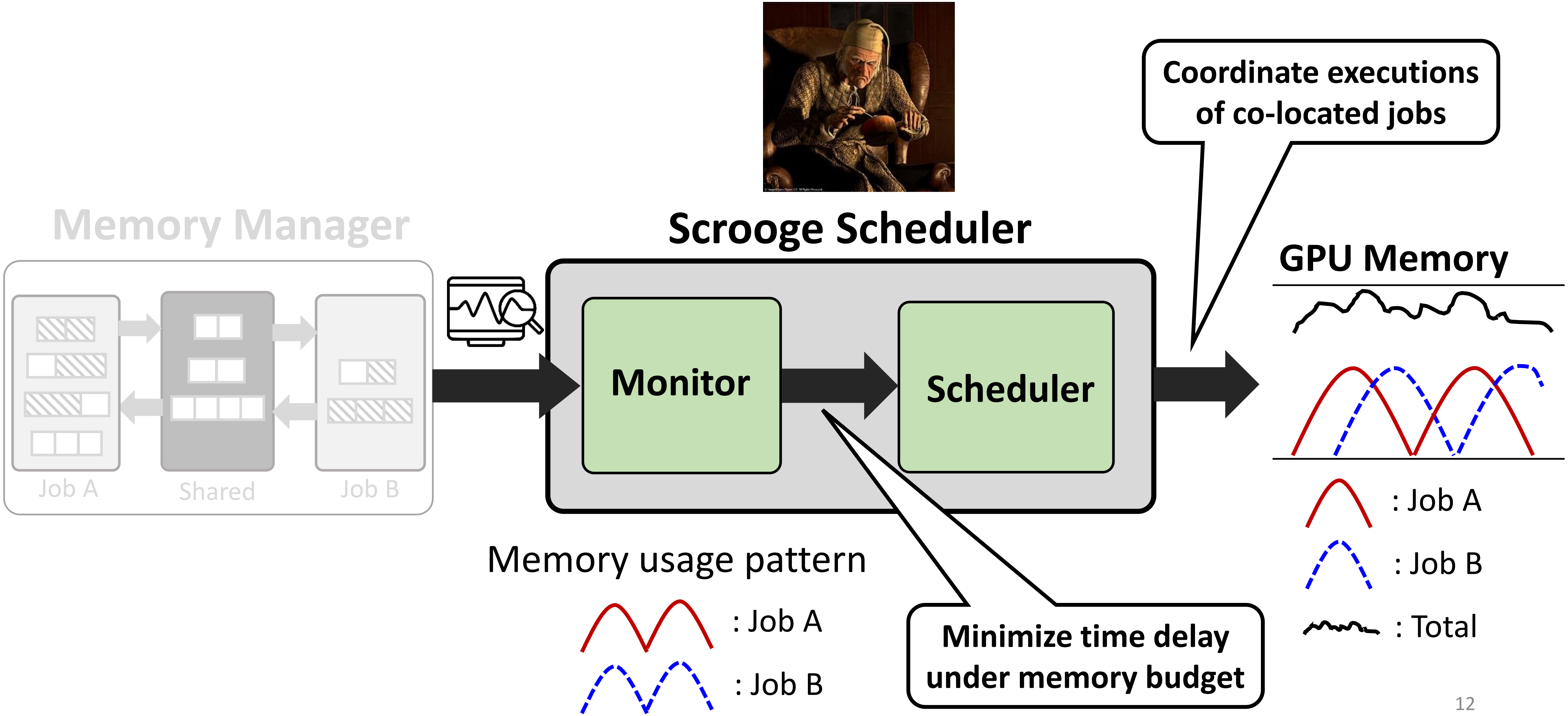
Controlling Inflight Kernel

Controlling the number of inflight kernel

Preventing early allocation



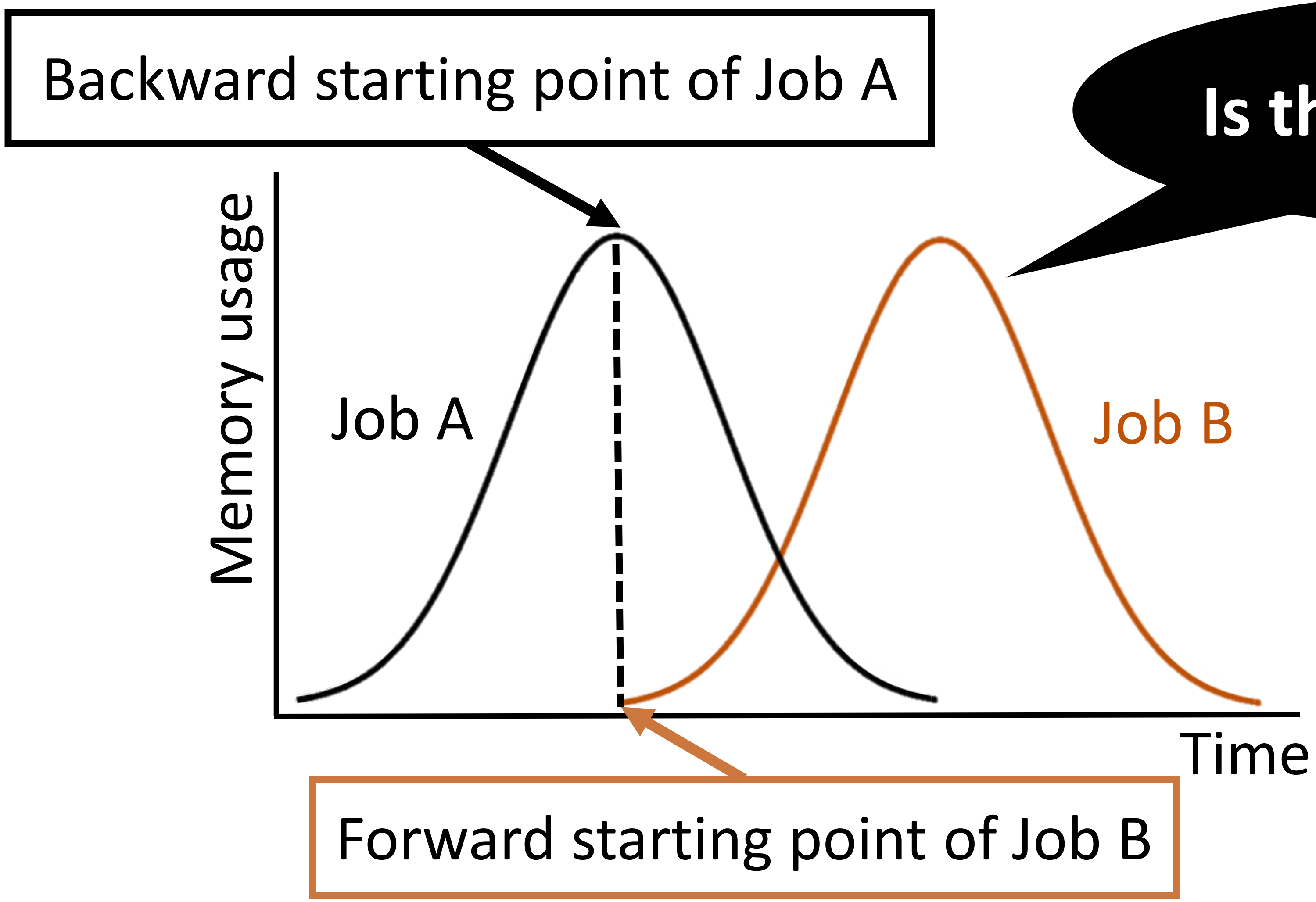
Roadmap



Coordinate executions of co-located jobs

Minimize time delay under memory budget

Naïve Scheduling



Backward starting point of Job A

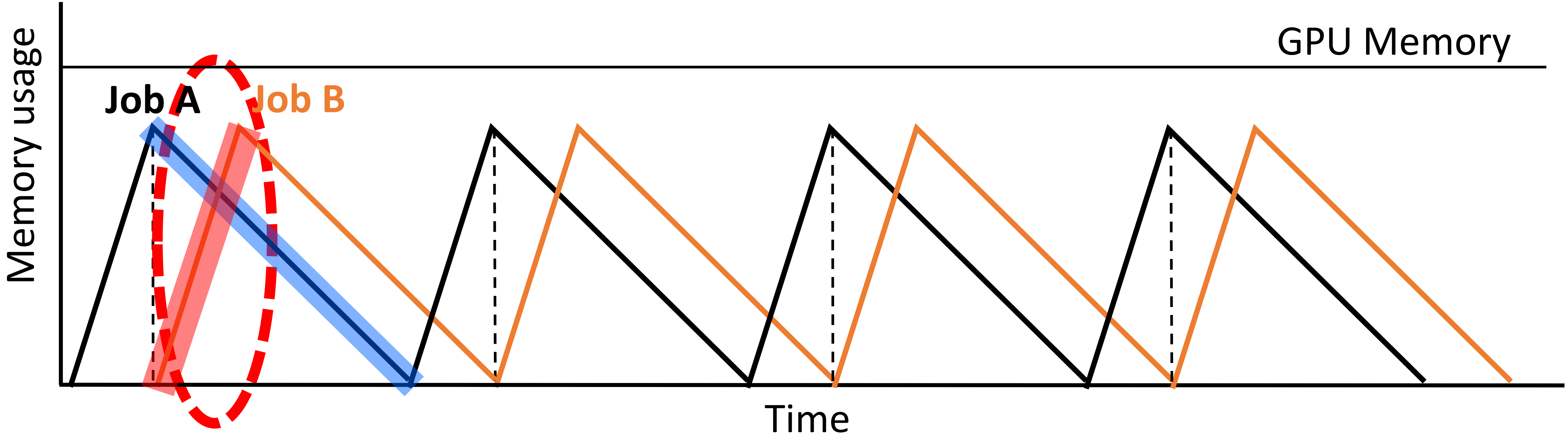
Forward starting point of Job B

Is there any issue?

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

Limitation of Naïve Scheduling

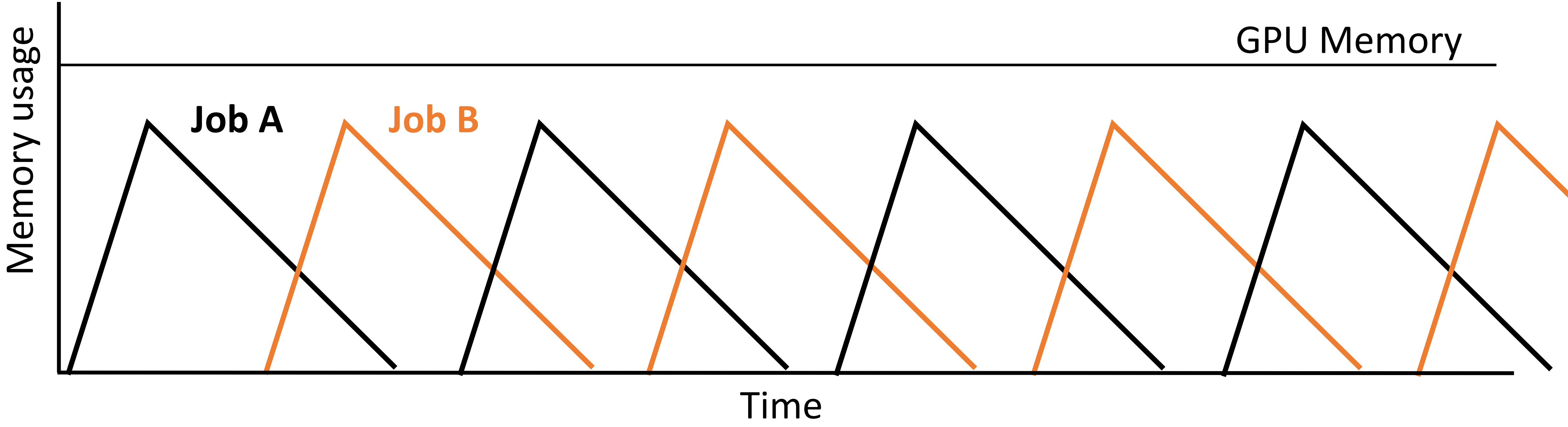
☹ **Memory over-subscription ➡ Hurting the throughput**



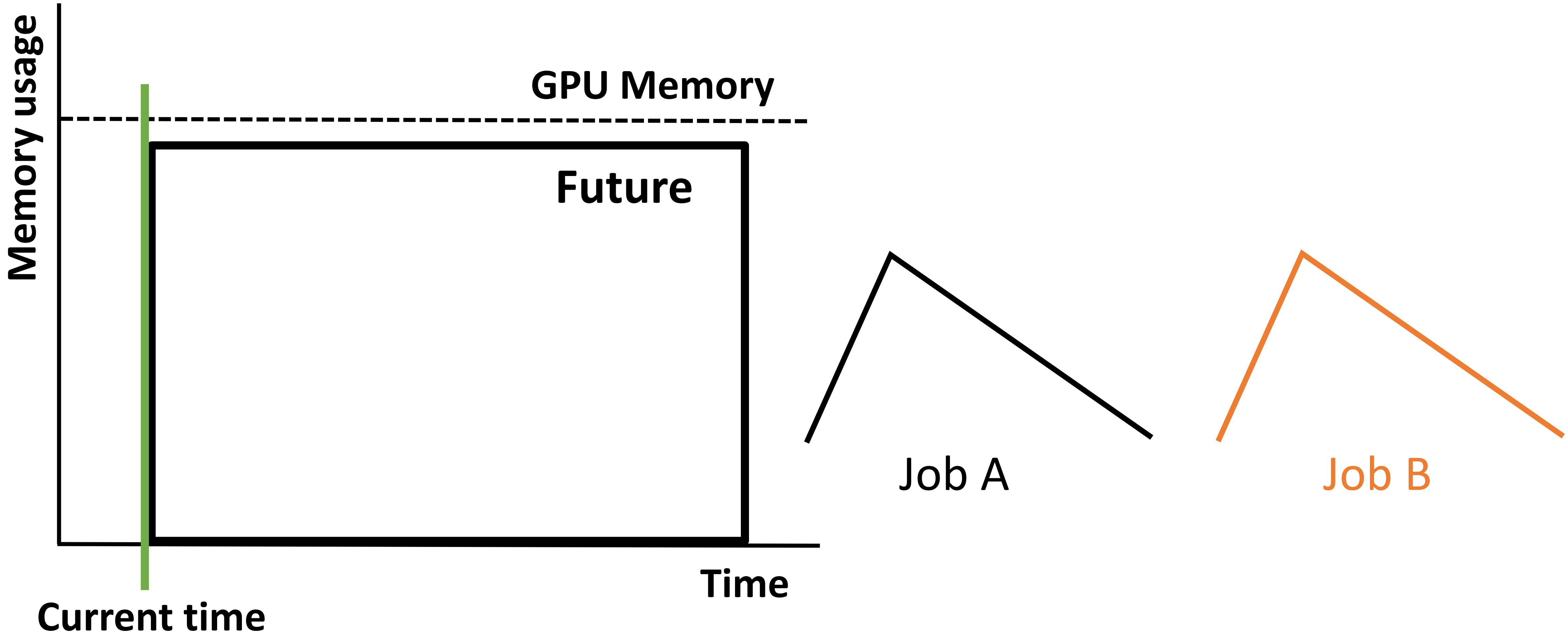
What We Want to Achieve

☹ **Memory over-subscription ➡ Hurting the throughput**

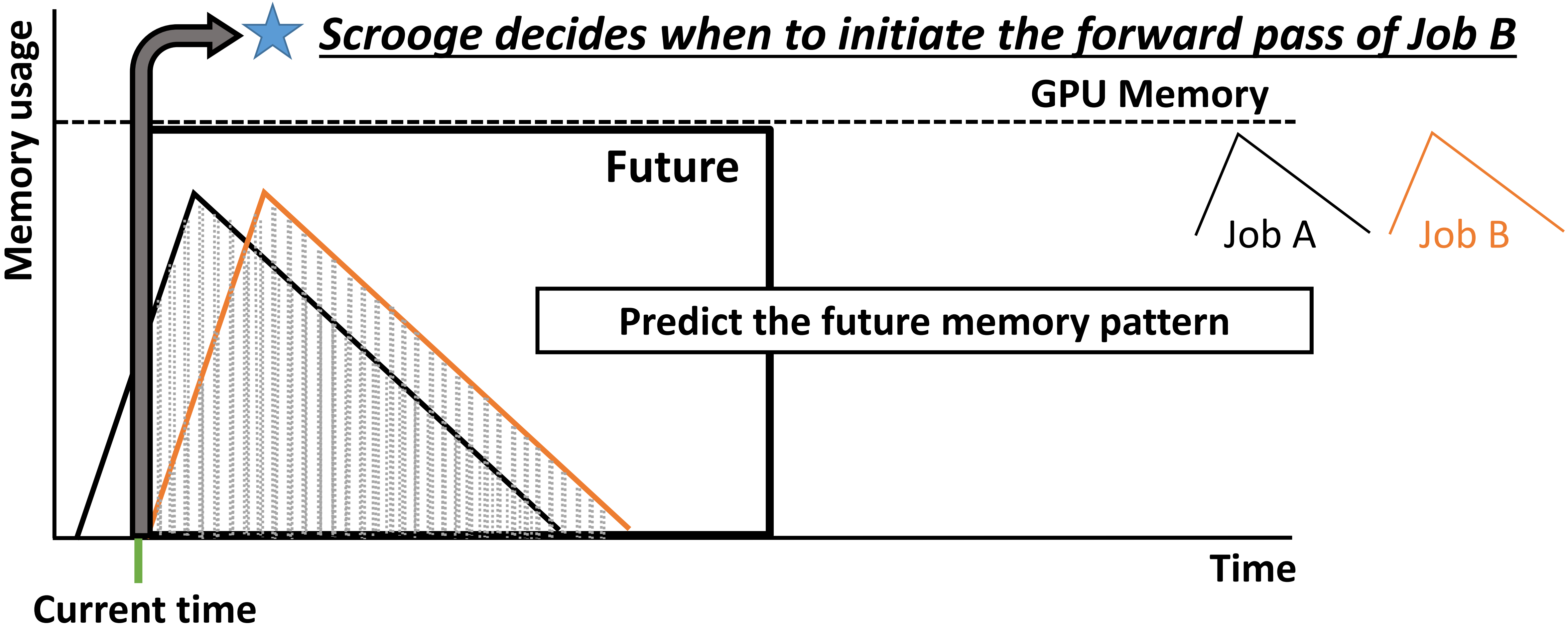
☺ **Proper coordination ➡ Preventing over-subscription**



Scrooge Scheduler



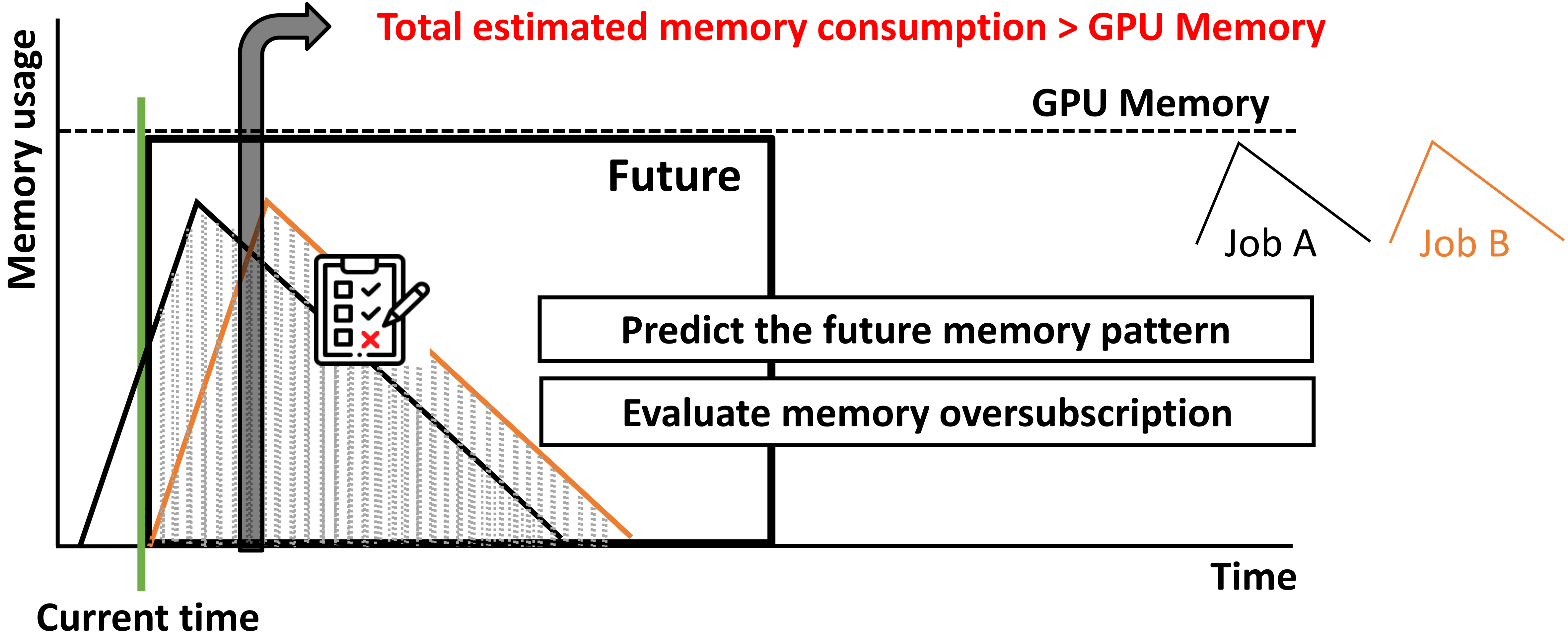
Scrooge Scheduler



Scrooge decides when to initiate the forward pass of Job B
GPU Memory
Memory usage
Future
Job A
Job B
Predict the future memory pattern
Time
Current time

Scrooge Scheduler

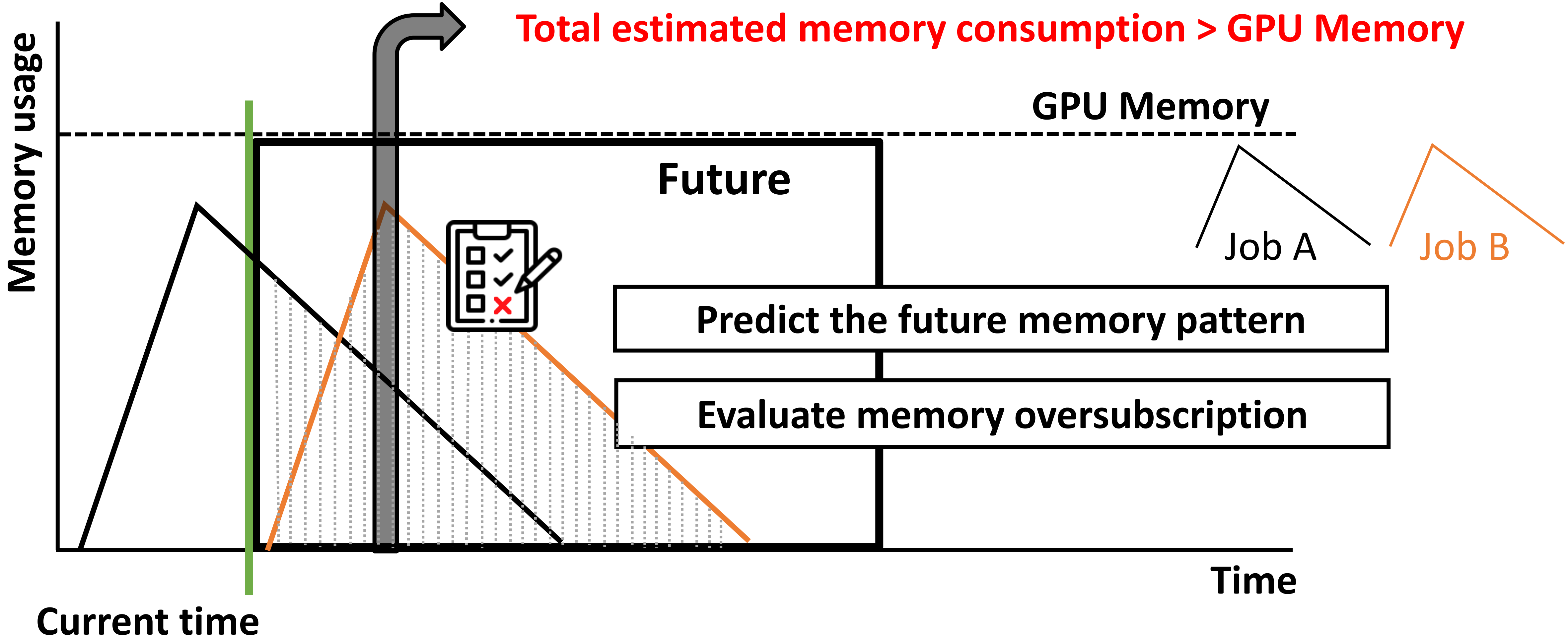
Fail ➡ Delay job B's forward pass!


Total estimated memory consumption > GPU Memory
Memory usage
GPU Memory
Future
Job A
Job B
Predict the future memory pattern
Evaluate memory oversubscription
Time
Current time

Scrooge Scheduler

Fail ➡ Delay job B’s forward pass!

Total estimated memory consumption > GPU Memory

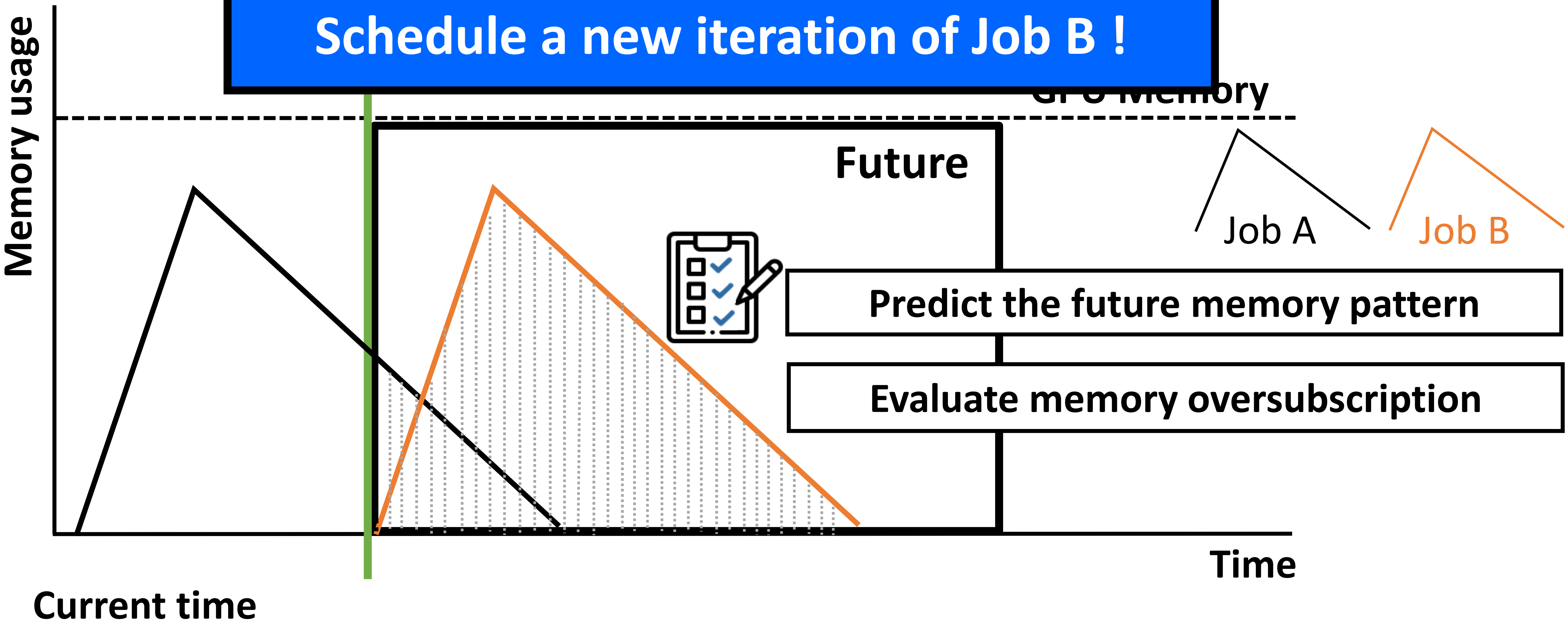


Predict the future memory pattern

Evaluate memory oversubscription

Scrooge Scheduler

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

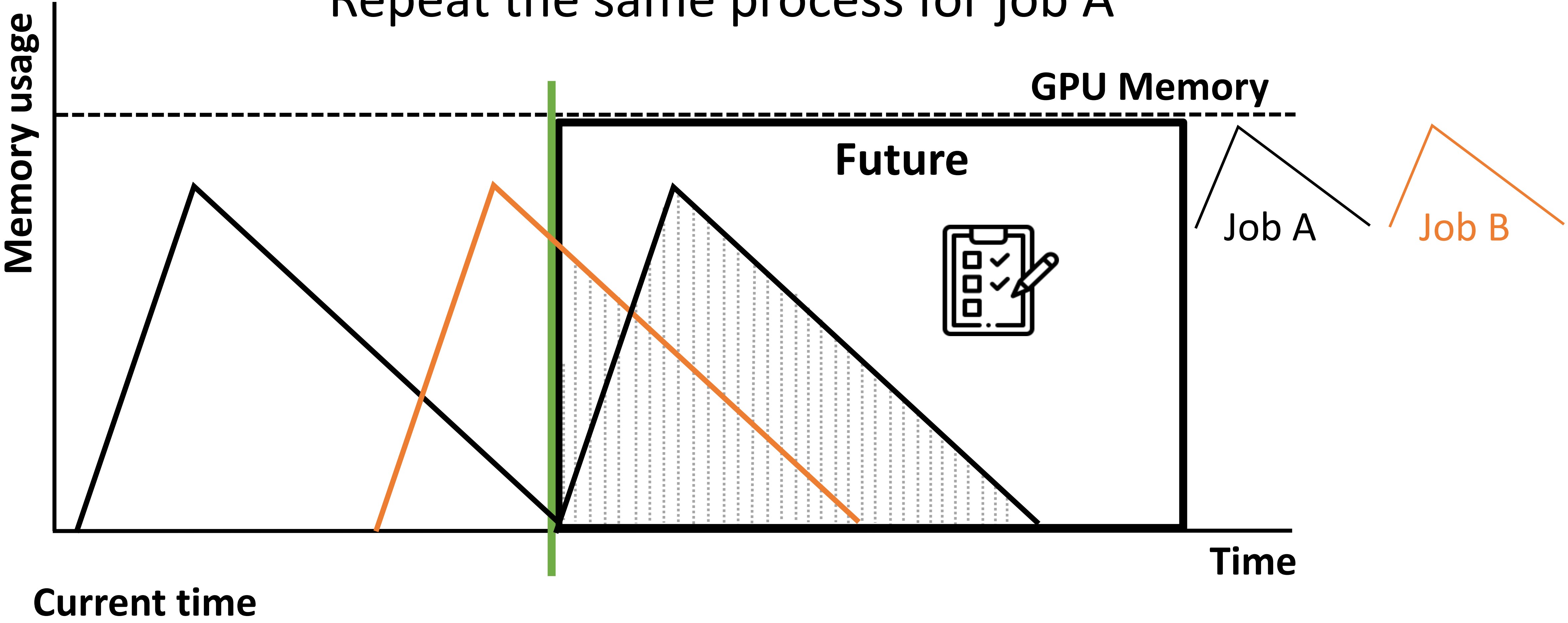


Predict the future memory pattern

Evaluate memory oversubscription

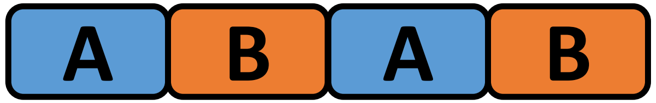
Scrooge Scheduler

Repeat the same process for job A



Evaluation

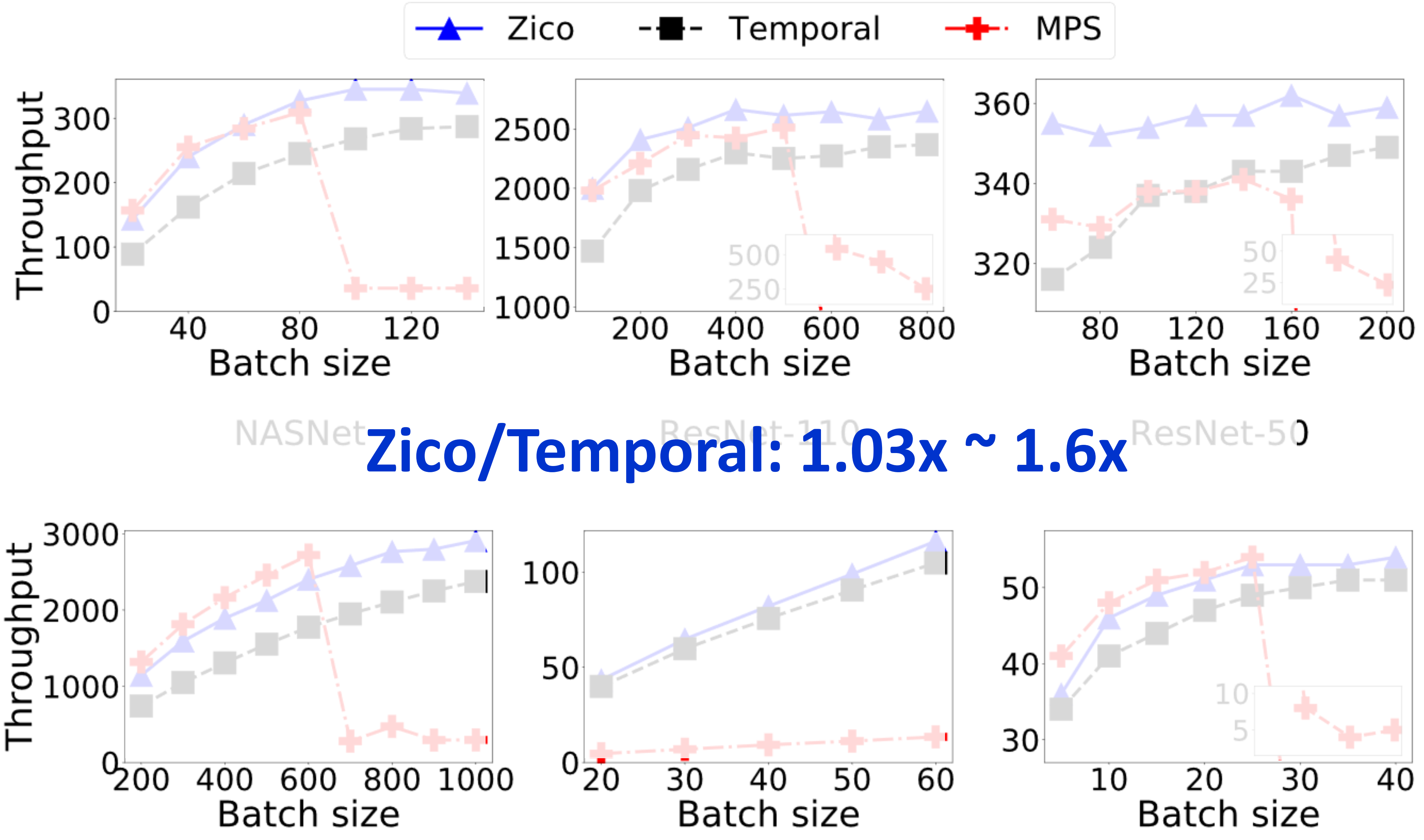
- Machine
  - GPU: Tesla V100 GPU, RTX 2080 Ti GPU
  - CPU: 3.8 GHz Intel Xeon(R)Gold 5222 4 CPU cores
  - RAM: 64 GB
- Benchmark
  - NASNet, ResNet-110, ResNet-50, GNMT, BERT, RHN
- Policies in comparison
  - Temporal: Ideal temporal sharing (no job switching overhead)
  - Spatial: NVIDIA MPS (no dynamic memory sharing)



A B A B

- Base framework: TensorFlow v1.13.1

Throughput: Identical Jobs

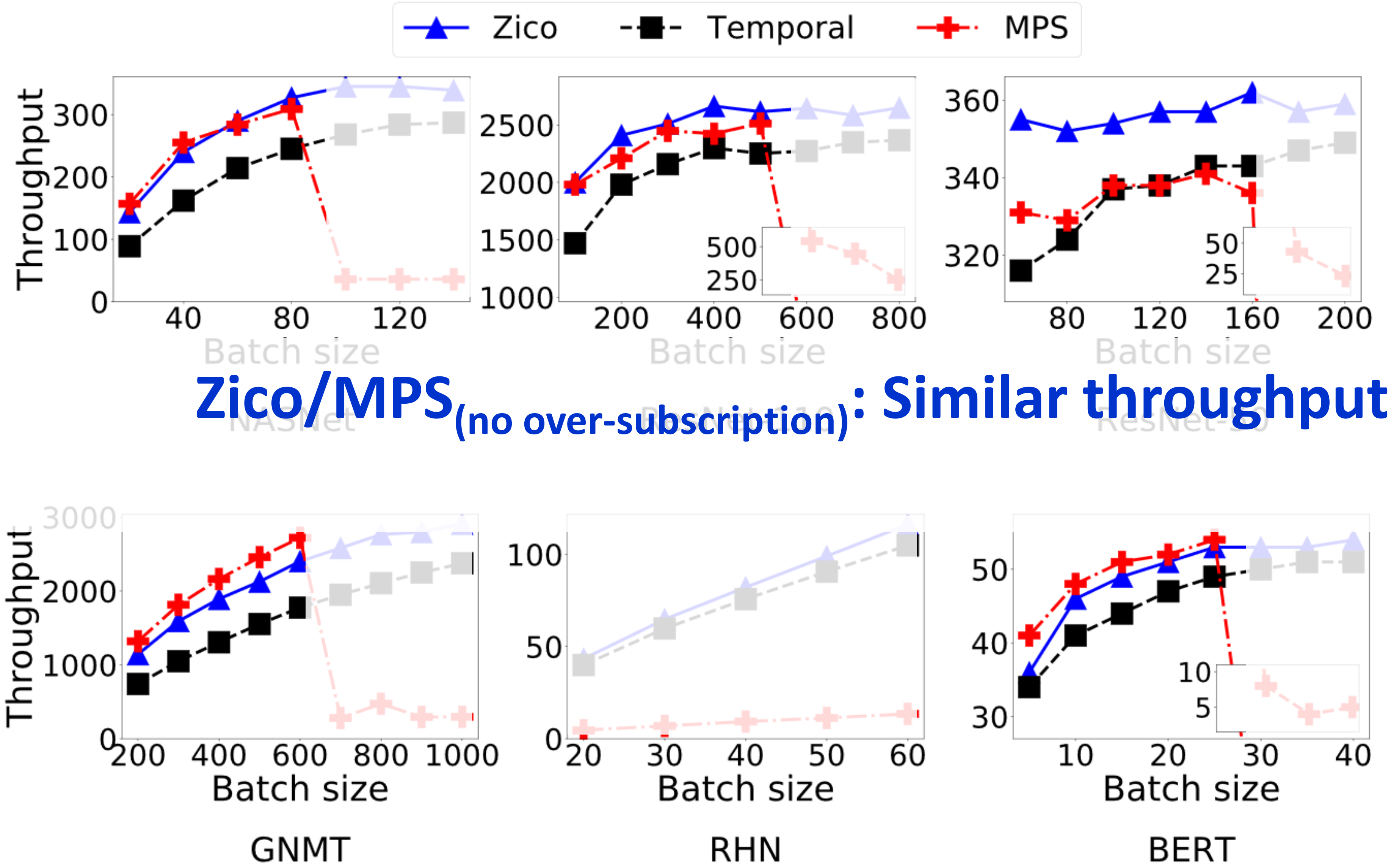


NASNet ResNet-110 ResNet-50

Zico/Temporal: 1.03x ~ 1.6x

GNMT RHN BERT

Throughput: Identical Jobs



NASNet

ResNet-110

ResNet-50

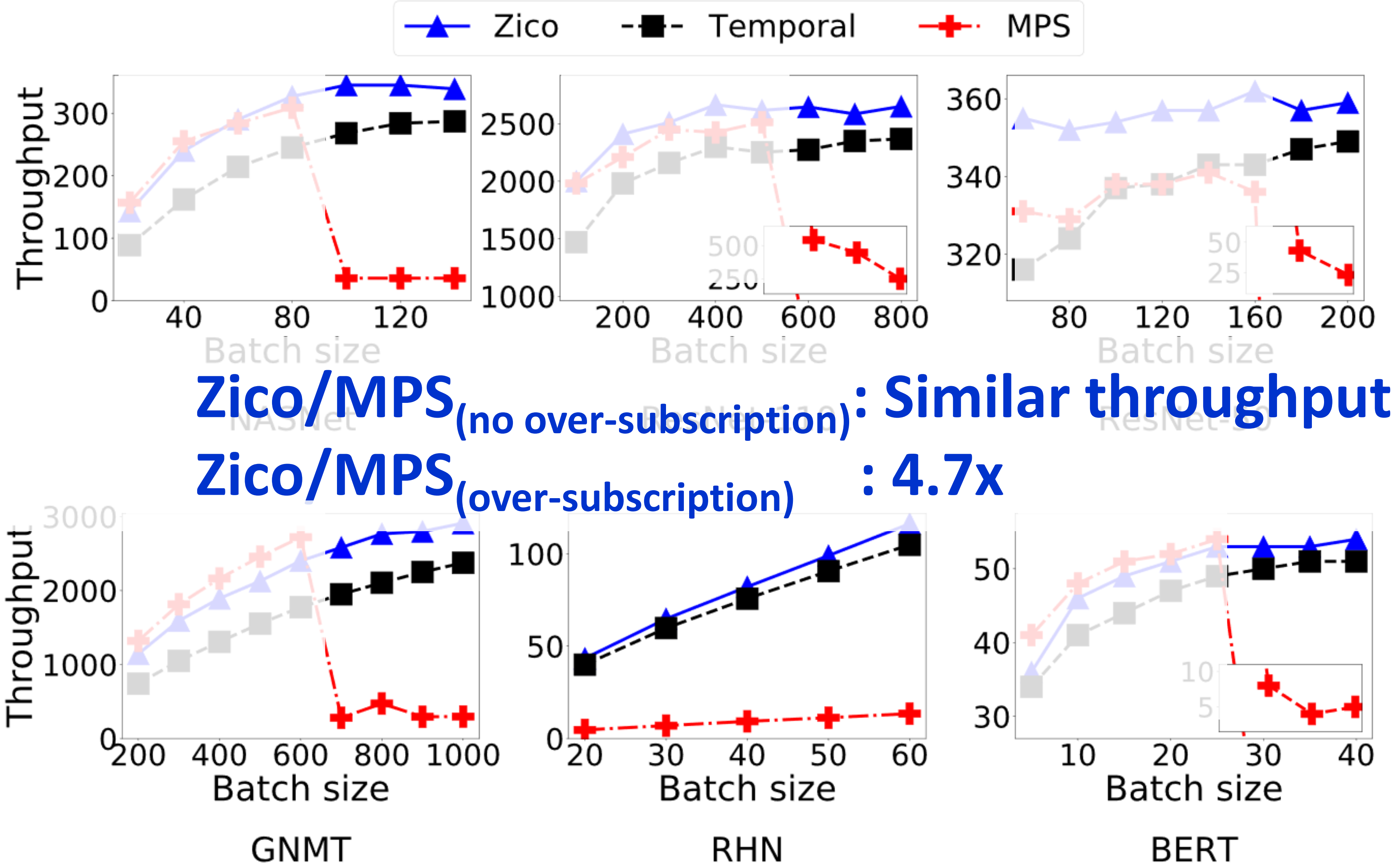
GNMT

RHN

BERT

Zico/MPS$_{\text{(no over-subscription)}}$: Similar throughput

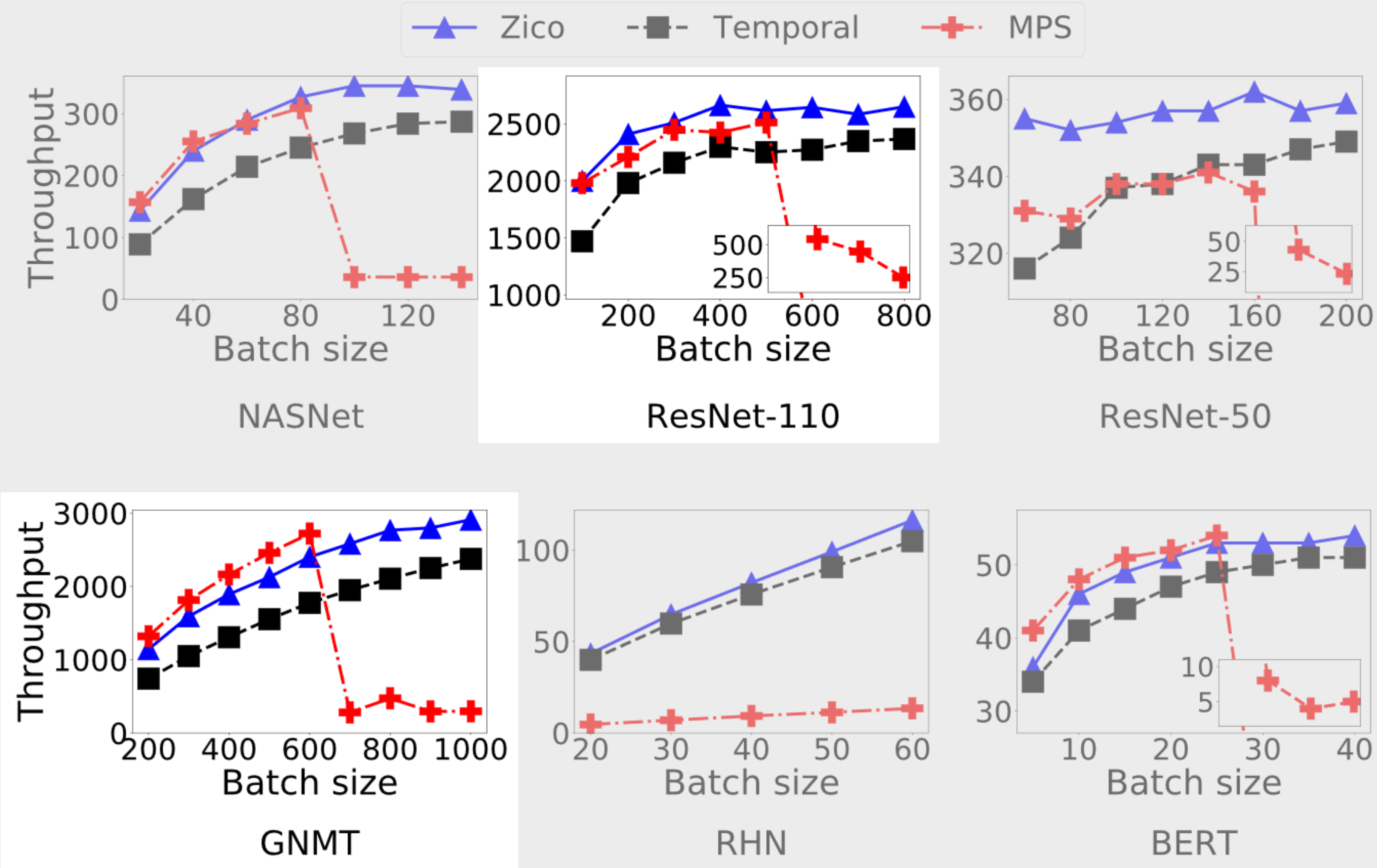
Throughput: Identical Jobs



Zico/MPS$_{\text{(no over-subscription)}}$: Similar throughput

Zico/MPS$_{\text{(over-subscription)}}$: 4.7x

Throughput: Identical Jobs



NASNet

ResNet-110

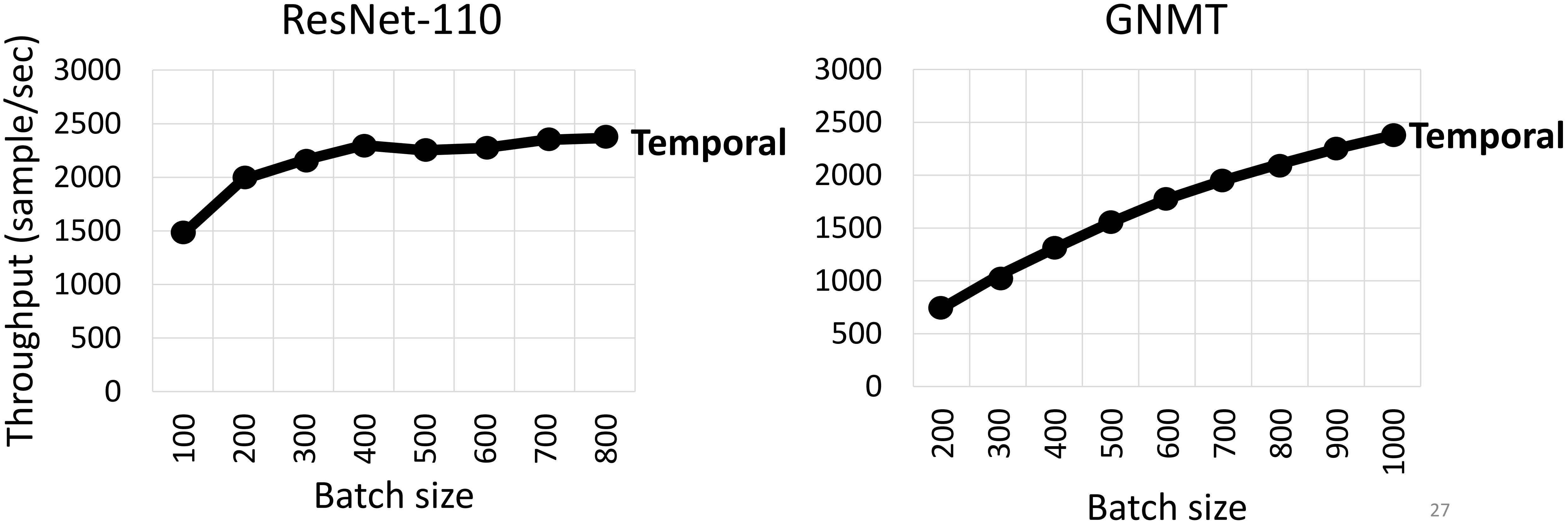
ResNet-50

GNMT

RHN

BERT

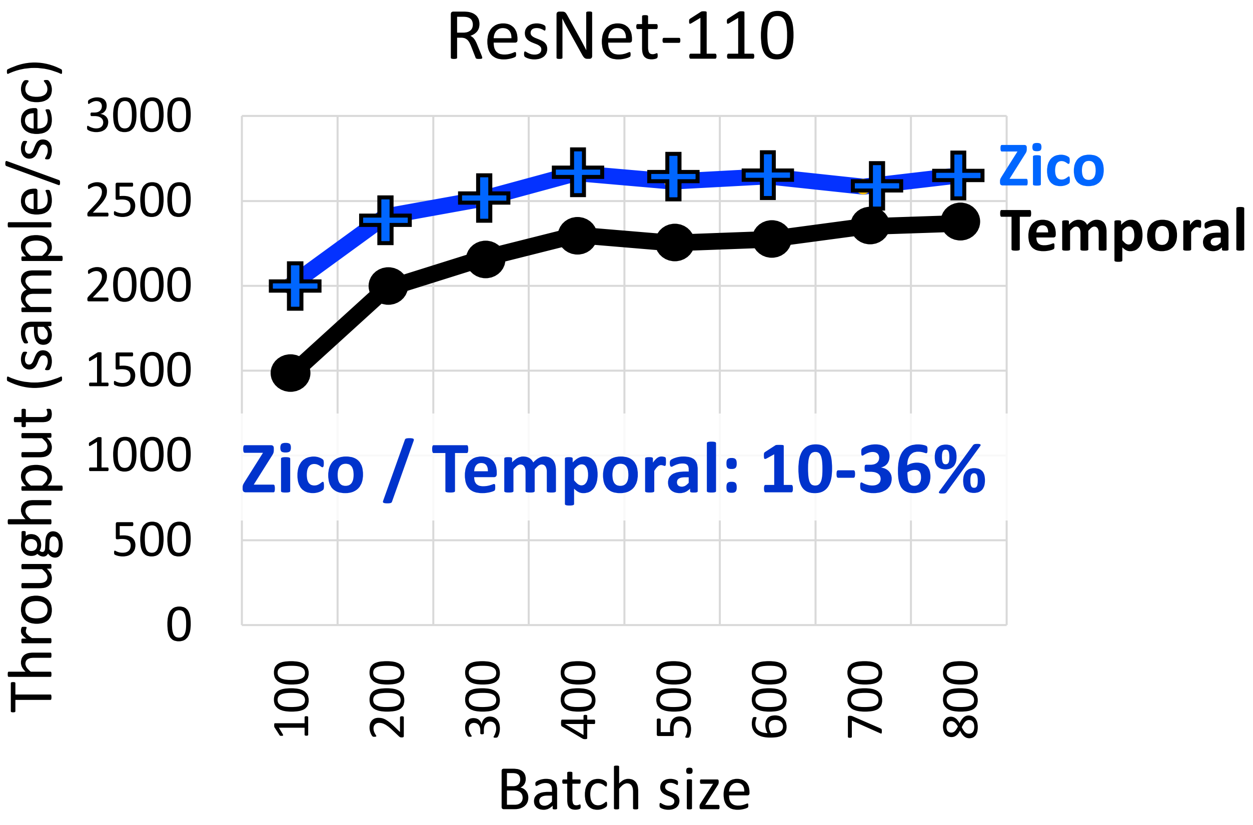
Throughput: Identical Models

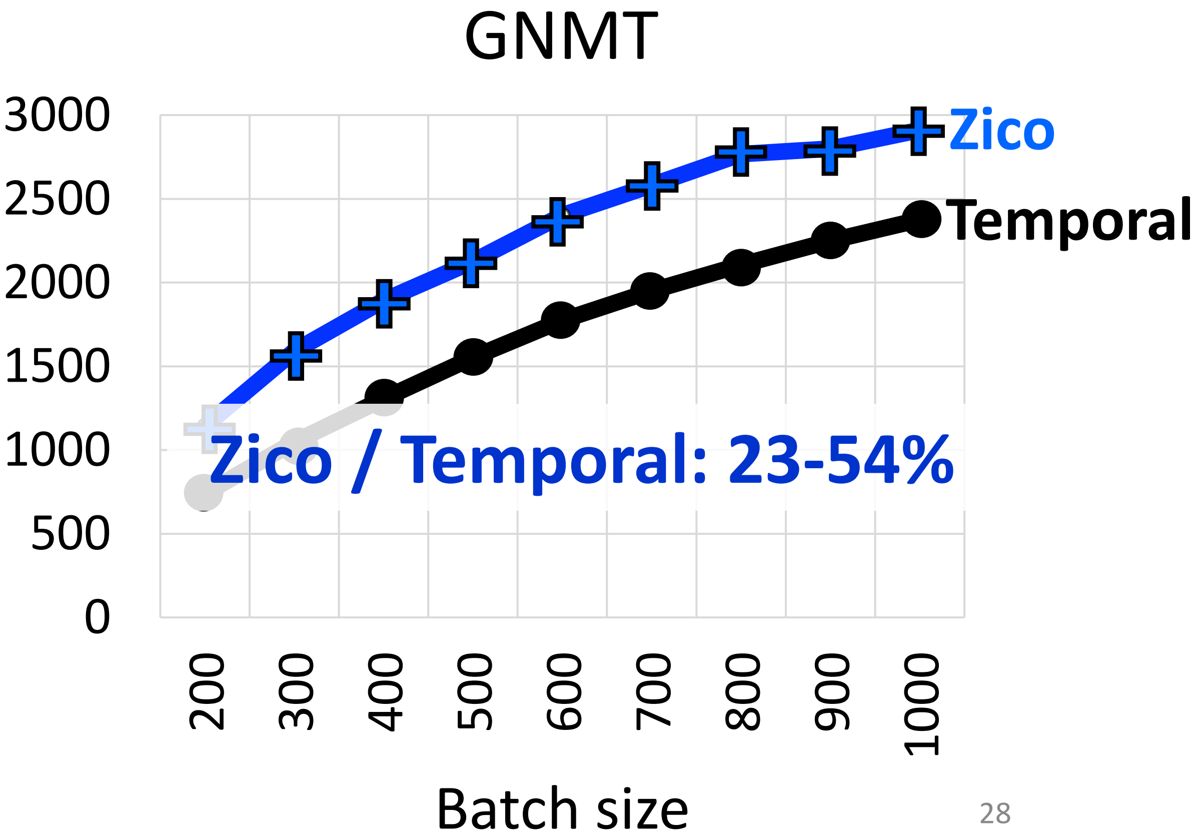


Throughput: Identical Models

Temporal doesn't fully utilize GPU

Zico always outperforms Temporal!

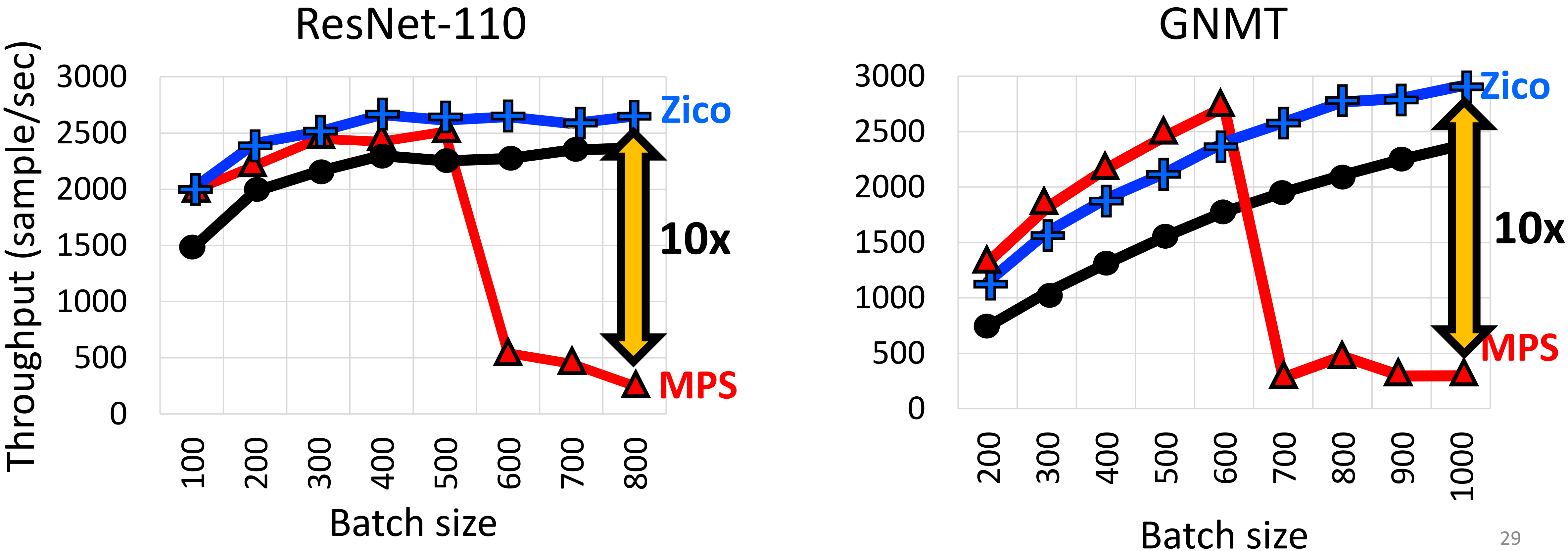




Throughput: Identical Models

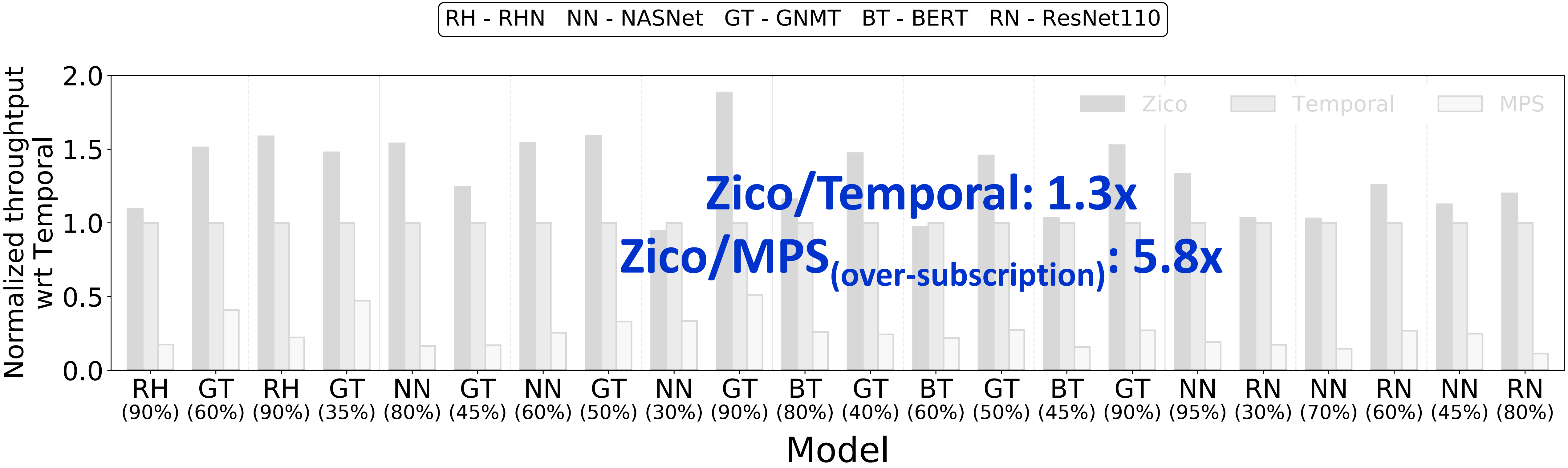
MPS suffers from memory over-subscription

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



Throughput: Non-identical Jobs

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

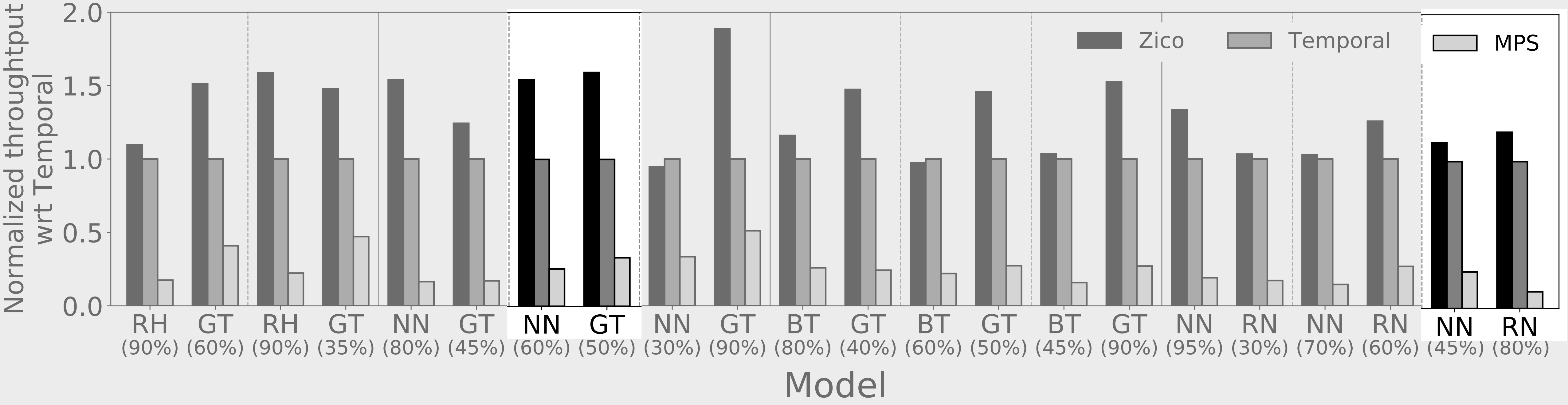


Zico/Temporal: 1.3x

Zico/MPS$_{\text{(over-subscription)}}$: 5.8x

Throughput: Non-identical Jobs

RH - RHN   NN - NASNet   GT - GNMT   BT - BERT   RN - ResNet110



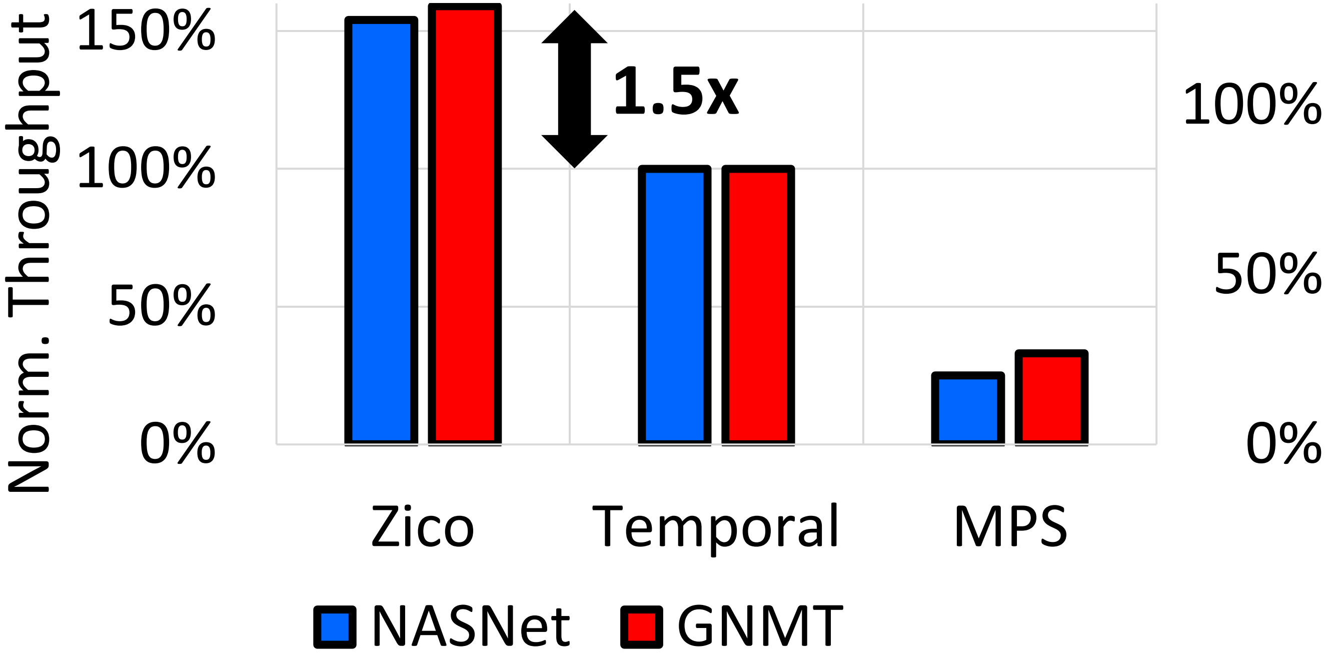
Throughput: Non-identical Models

Zico successfully co-locates non-identical models!

More improvement when low utilization model is co-located

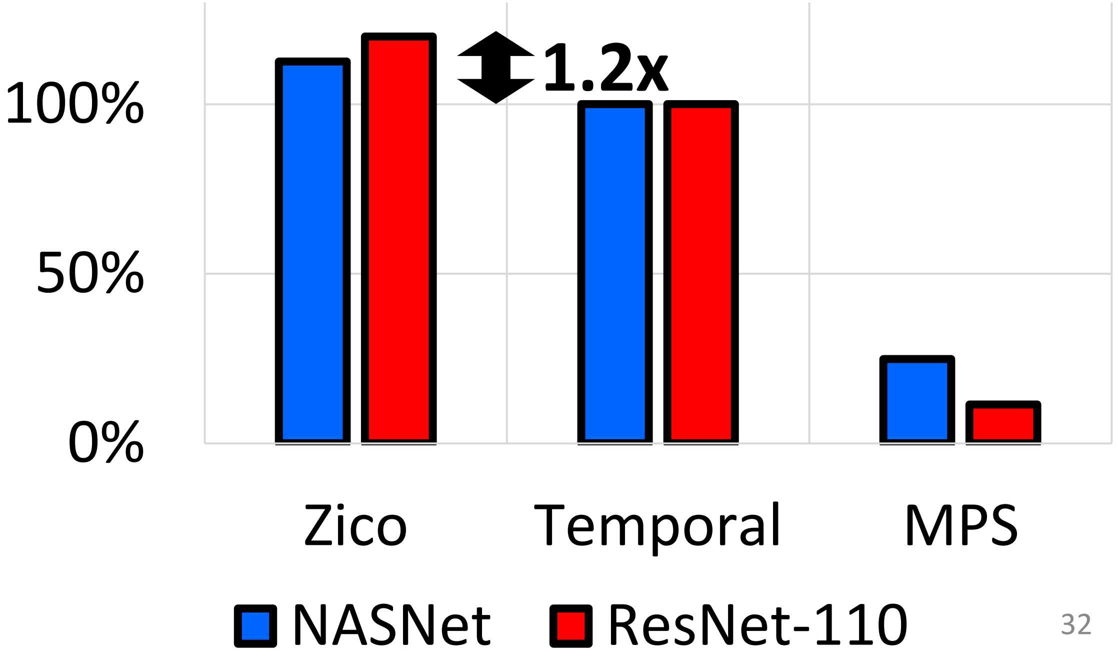
High util. + Low util.

(NASNet) (GNMT)



High util. + High util.

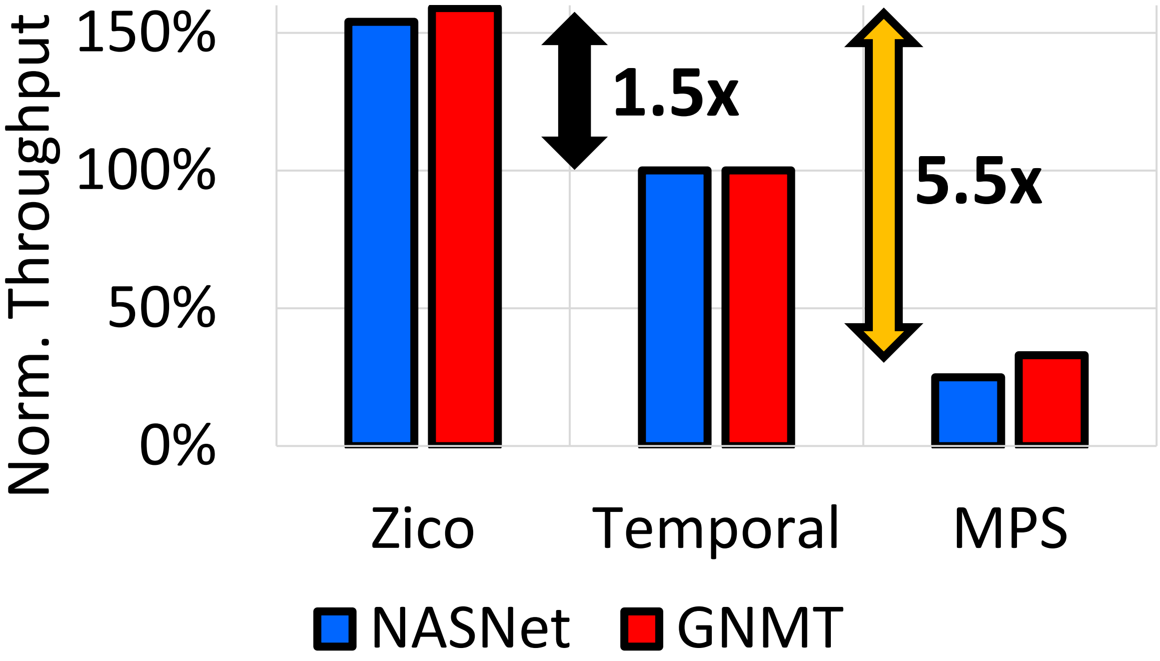
(NASNet) (ResNet-110)



Throughput: Non-identical Jobs

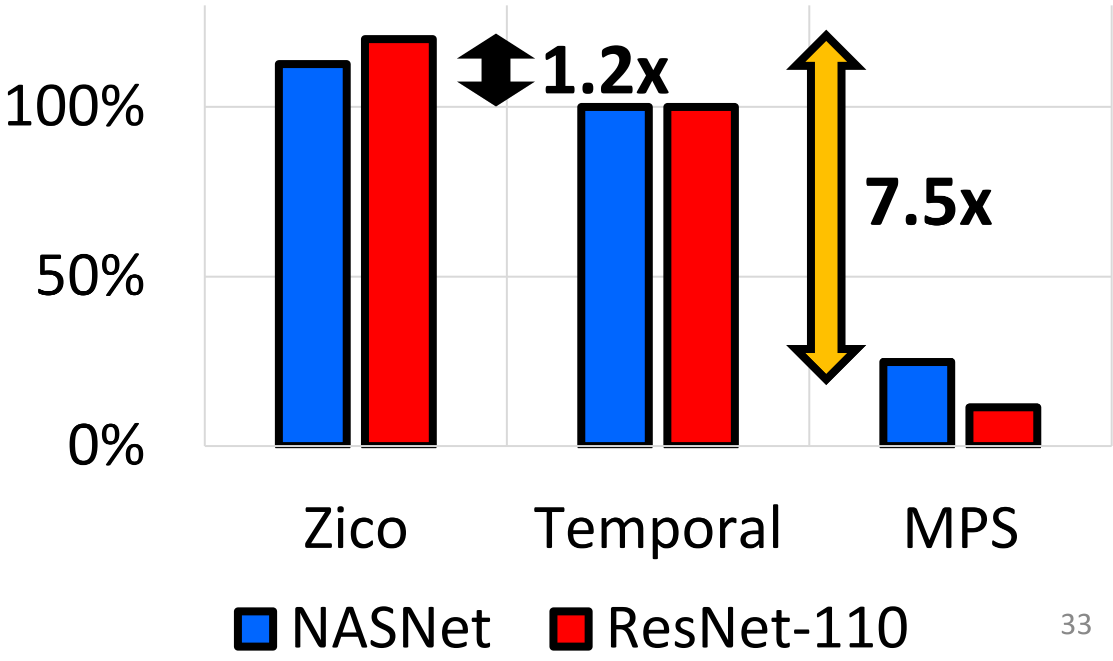
High util. + Low util.

(NASNet) (GNMT)



High util. + Low util.

(NASNet) (ResNet-110)

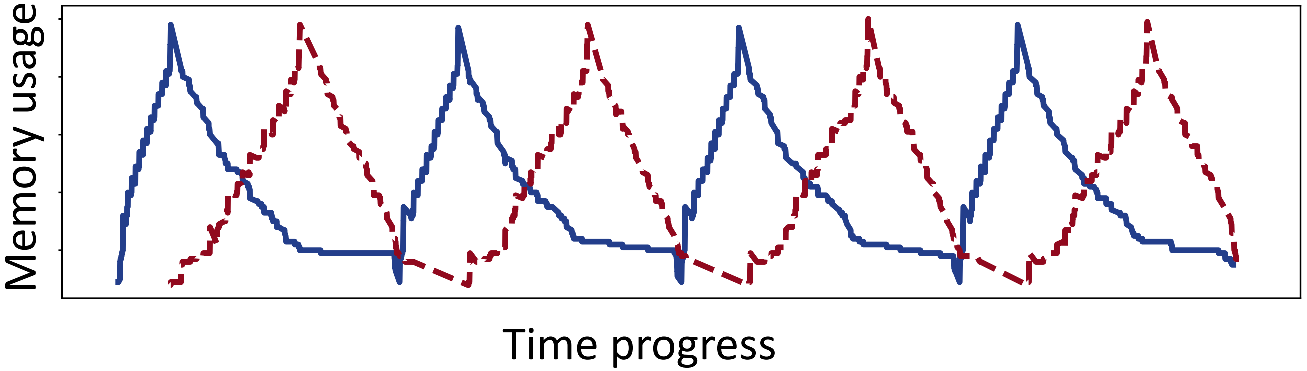


Scheduling Example

BERT + BERT

(budge: 32GB)

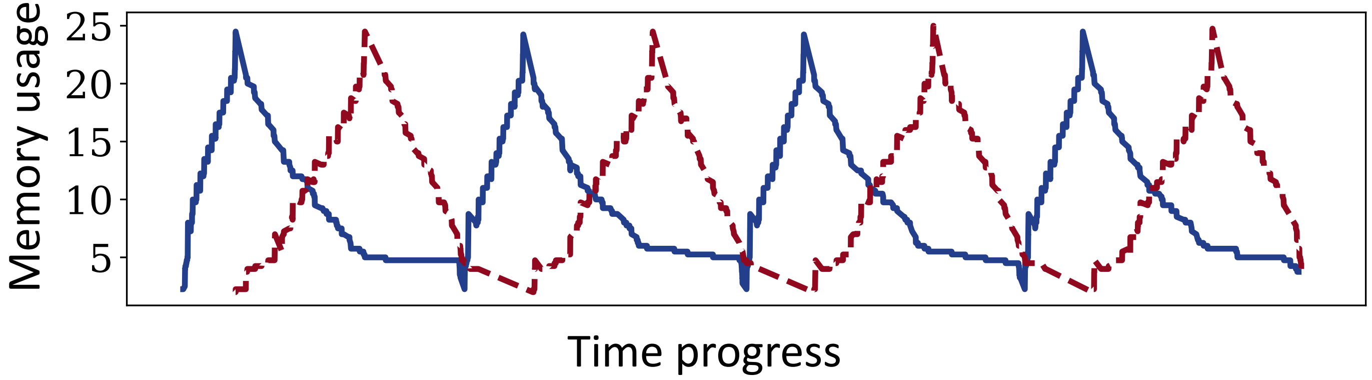
Identical model co-location



Scheduling Example

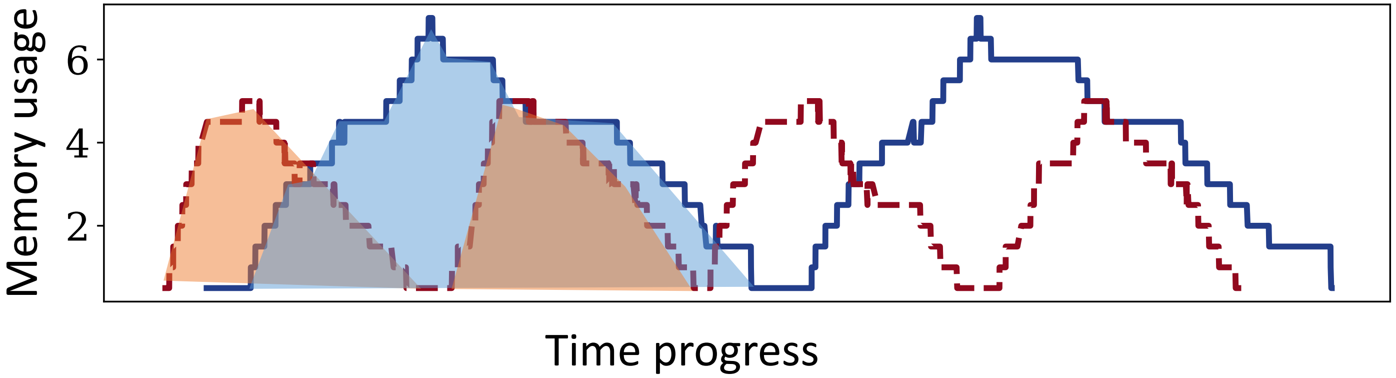
BERT + BERT
(budge: 32GB)

Identical model co-location



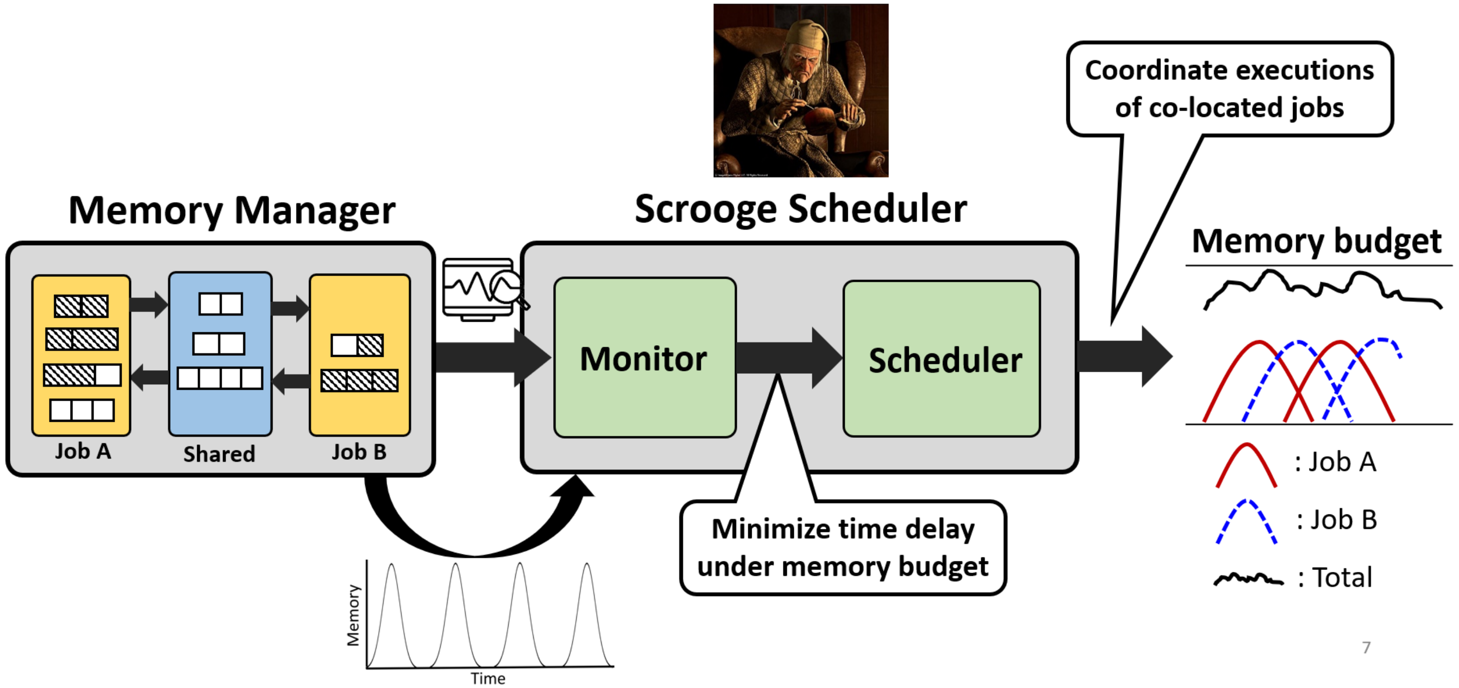
NASNet + ResNet-110
(budge: 11GB)

Non-identical model co-location



Summary

- Zico is the first introducing memory-aware scheduler
- Zico proposes widely applicable GPU sharing techniques for training


Memory Manager
Job A
Shared
Job B
Scrooge Scheduler
Monitor
Scheduler
Coordinate executions of co-located jobs
Minimize time delay under memory budget
Memory budget
: Job A
: Job B
: Total
Memory
Time
7