ARK: GPU-driven Code Execution for Distributed Deep Learning

Changho Hwang[1, 2], KyoungSoo Park[1], Ran Shu[2], Xinyuan Qu[2], Peng Cheng[2], and Yongqiang Xiong[2]

[1]*KAIST,* [2]*Microsoft Research*

Talk in NSDI'23

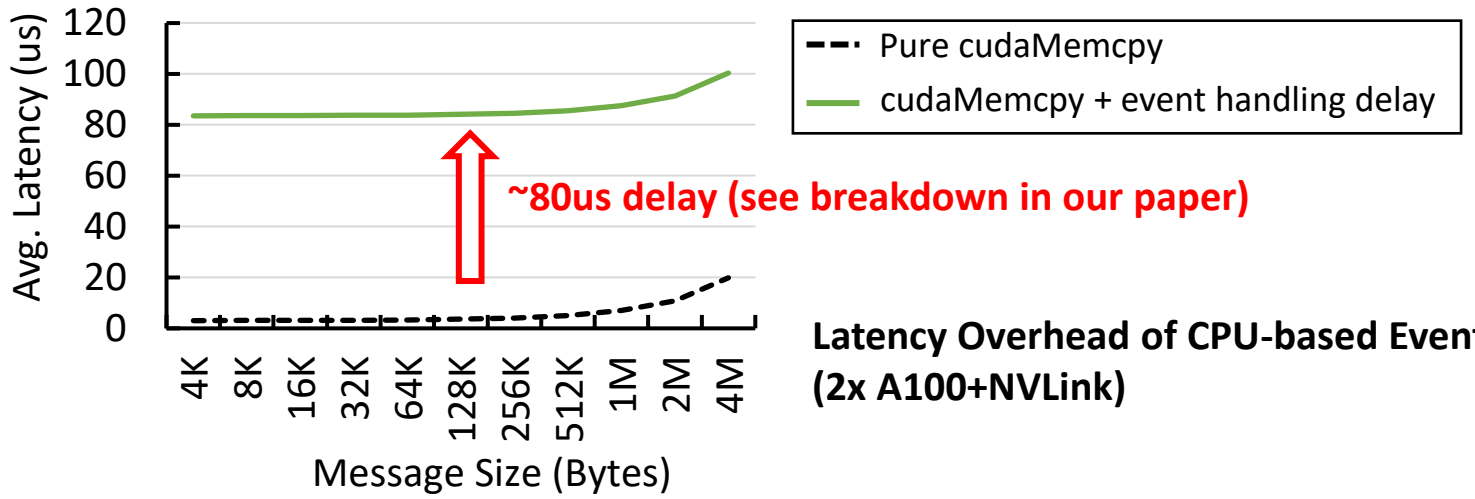
Modern AI Systems at Scale

- Increasingly large model size (> 10s of trillions of parameters)
- More co-operating accelerators (GPUs in this work)
- **Inter-GPU communication** as a key component of AI systems

Communication Overhead in Distributed DL (1 / 2)

1. Small data transfer (down to ~10s KB)
   - Model architecture
   - Multi-stage collective communication (ring, tree, hierarchical, ...)
   - Large-scale: more stages, smaller size per transfer

→ **Control plane overhead: event handling** between small data transfers

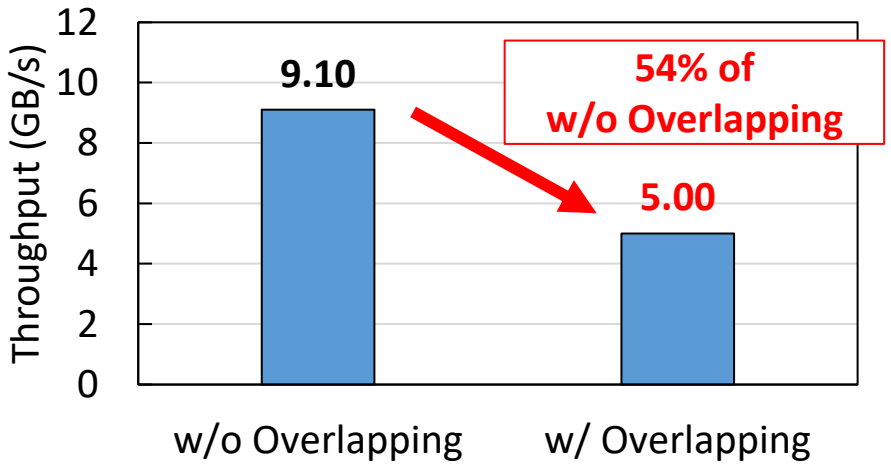


Latency Overhead of CPU-based Event Handling (cudaEvent) (2x A100+NVLink)

Communication Overhead in Distributed DL (2 / 2)

2. Overlapping computation and communication
   - Data-/pipeline-parallelism
   - Substantial slowdown by interference
     - E.g., NCCL use GPU threads for data I/O

→ **Data plane overhead: I/O interference** to parallel computation



Average NCCL v2.11 AllReduce throughput during BERT-Large data-parallel training (8x V100+PCIe)

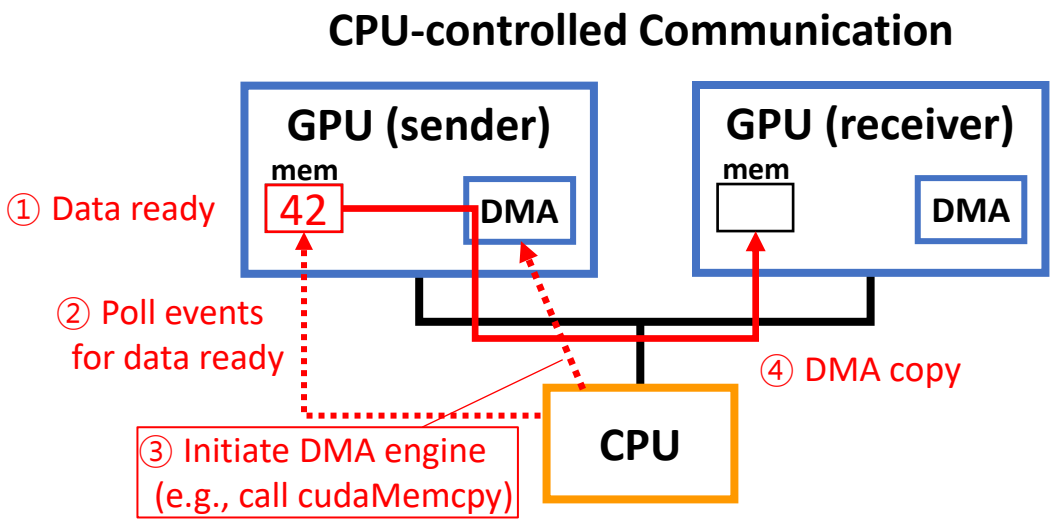
Existing Systems: Overview

- Existing systems tackle only one aspect, but not both
- **CPU-controlled vs. GPU-controlled** communication
  - Handle communication events on CPU vs. GPU

	Control Plane: Event Handling	Data Plane: I/O Interference
CPU-controlled communication	✗	✓
GPU-controlled communication	✓	✗
This work	✓	✓

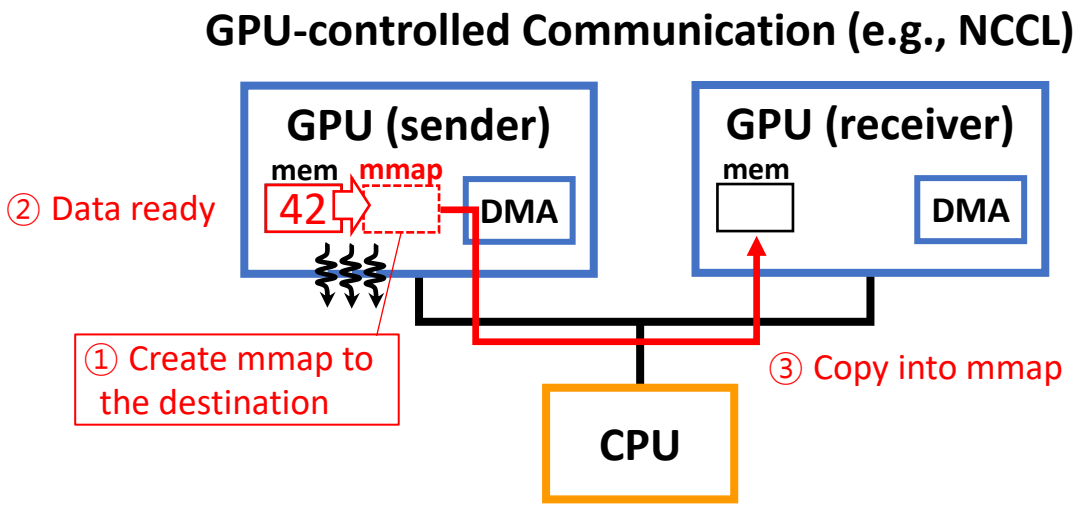
(details in following slides)

CPU-control vs. GPU-control



+ Minimize I/O overhead on GPU

– Long latency from CPU intervention



+ Low latency w/o CPU enrollment

– Substantial I/O overhead on GPU

Neither tackles both latency and I/O overhead at the same time!

ARK: A *GPU-driven System* – Key Design

Design 1.
GPU-controlled DMA

Let GPU threads initiate DMA directly

Fast event handling
Control plane on GPU

Minimal I/O overhead
Offload I/O to HW DMA engines

Design 2.
Autonomous GPU Execution Control

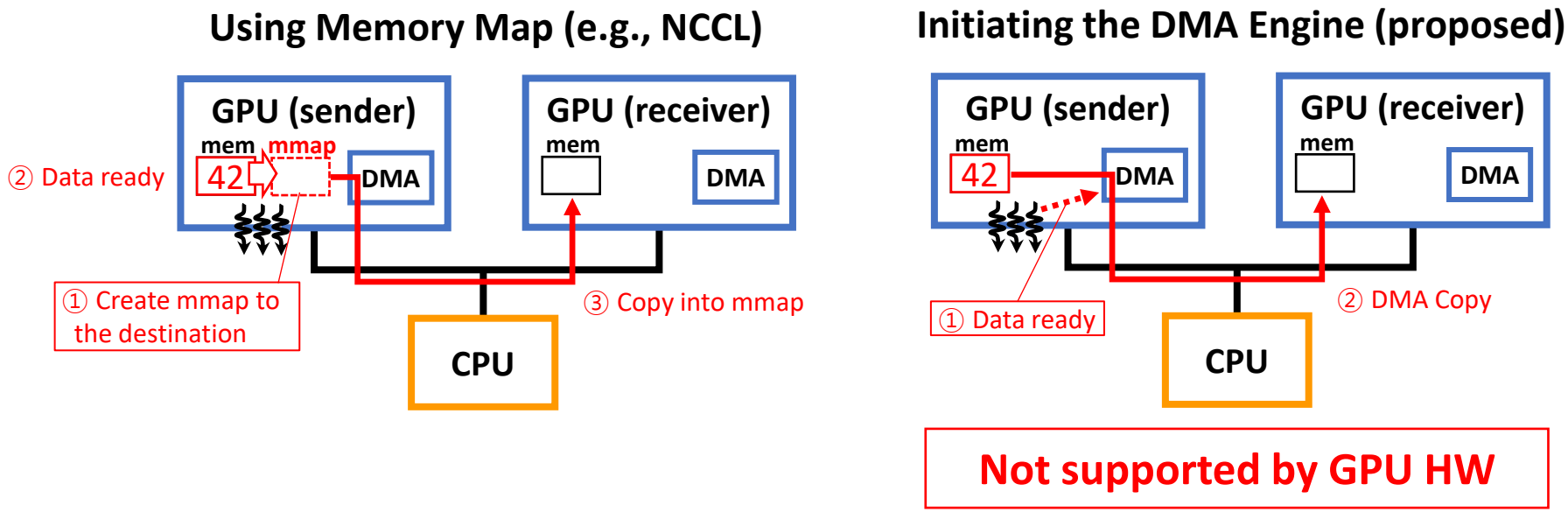
All GPU tasks in a single GPU kernel (*loop kernel)*

Fast event handling
No external control signal

Efficient computation
Fine-grained, software-defined GPU task scheduling

GPU-controlled DMA Design

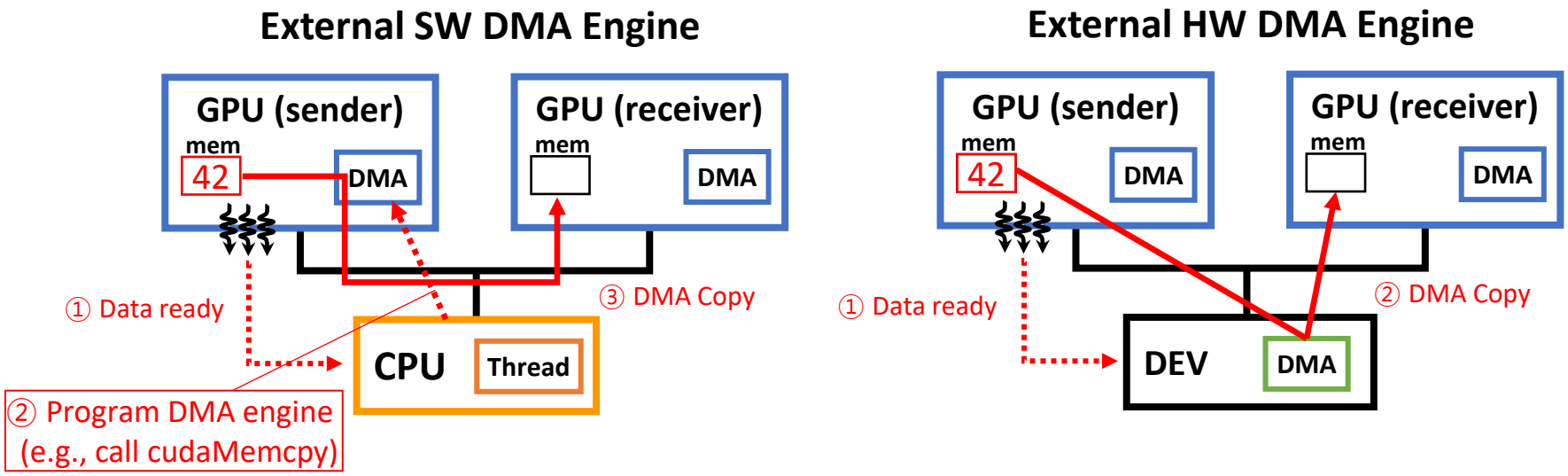
- Initiate the DMA engine by GPU threads



Design 1. GPU-controlled DMA

External DMA Engines

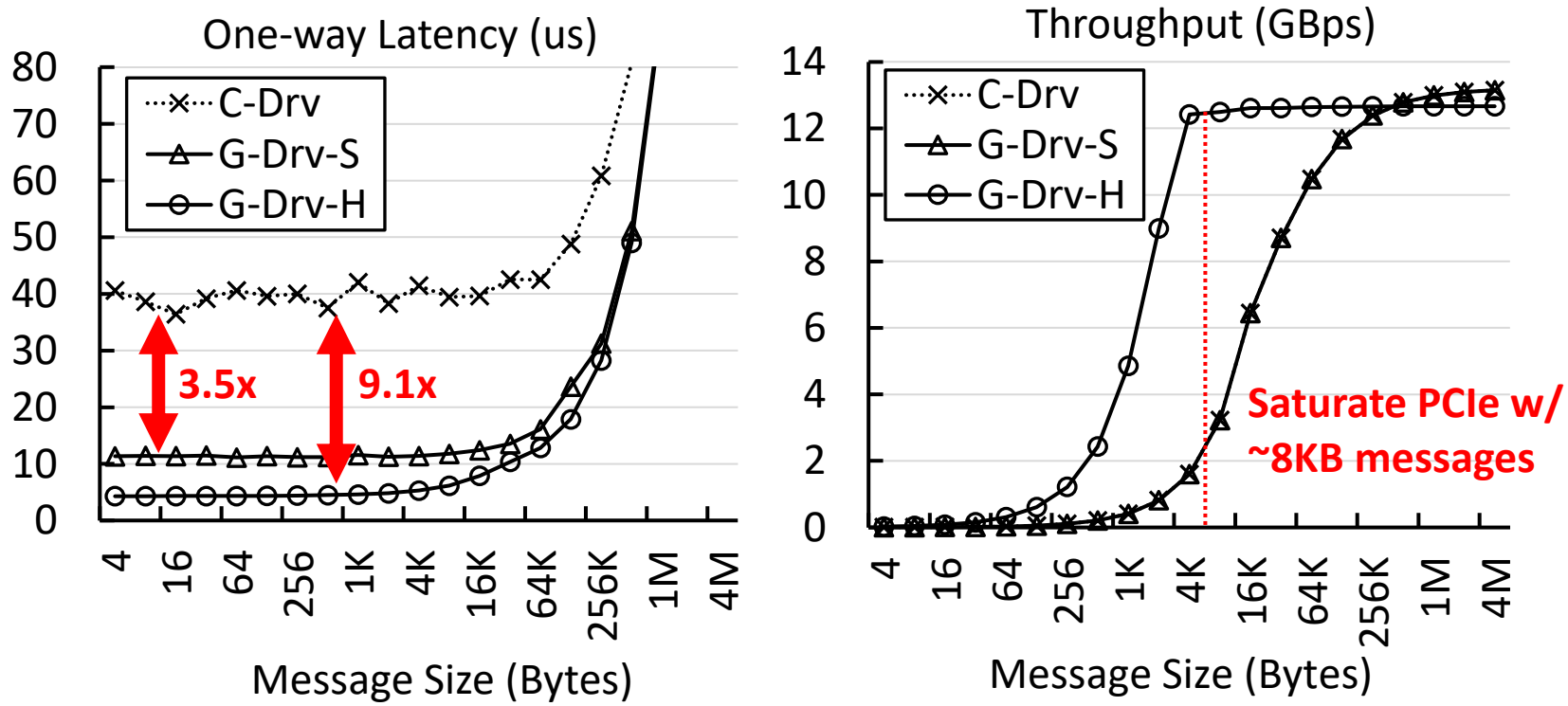
- Two implementations
  - SW engine: easy to deploy
  - HW engine (prototype): improved latency & throughput



Design 1. GPU-controlled DMA

DMA Engine Microbenchmark

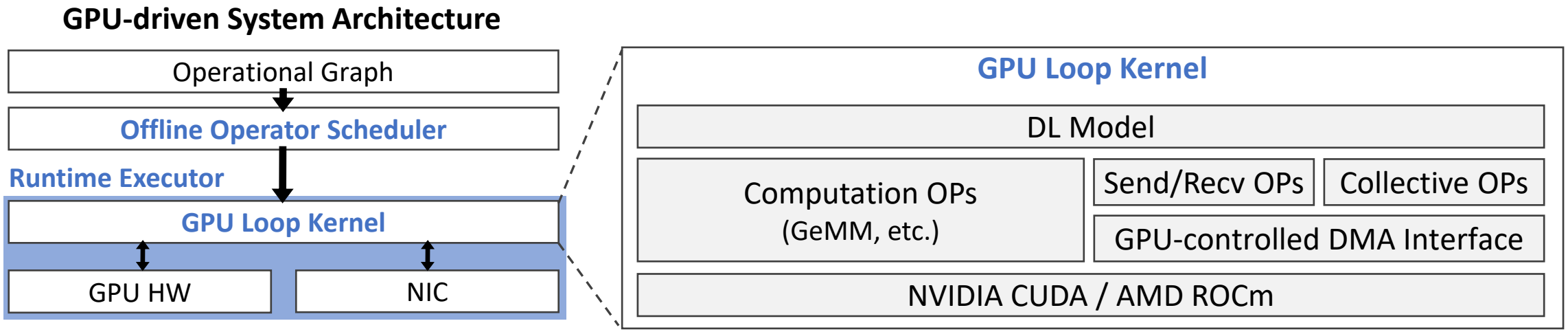
- One-way latency & throughput comparison
  - C-Drv: CPU-controlled (baseline)
  - G-Drv-S/H: GPU-controlled SW/HW engine



Used 2x NVIDIA V100 GPUs (PCIe v3.0, PIX linked) + Intel Xeon Gold 6240R CPU @ 2.40GHz

Design 2. Autonomous GPU Execution Control

GPU-driven System Architecture

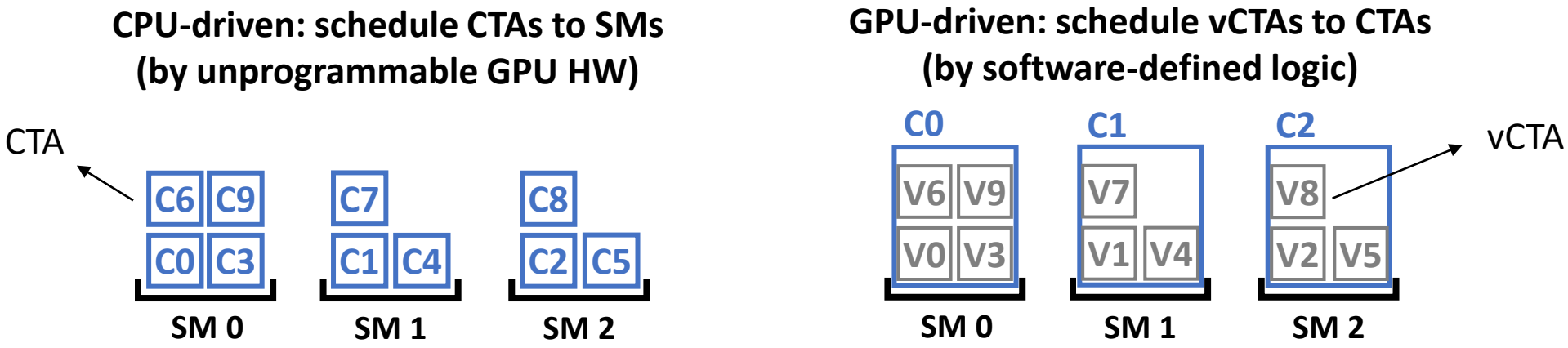


- Benefits
  - GPU-controlled DMA
  - Fine-grained & software-defined GPU task scheduling
    - Holistic optimization with global view

Virtual CTA (Cooperative Thread Array)

- Software-defined GPU task scheduler: *how does it work?*

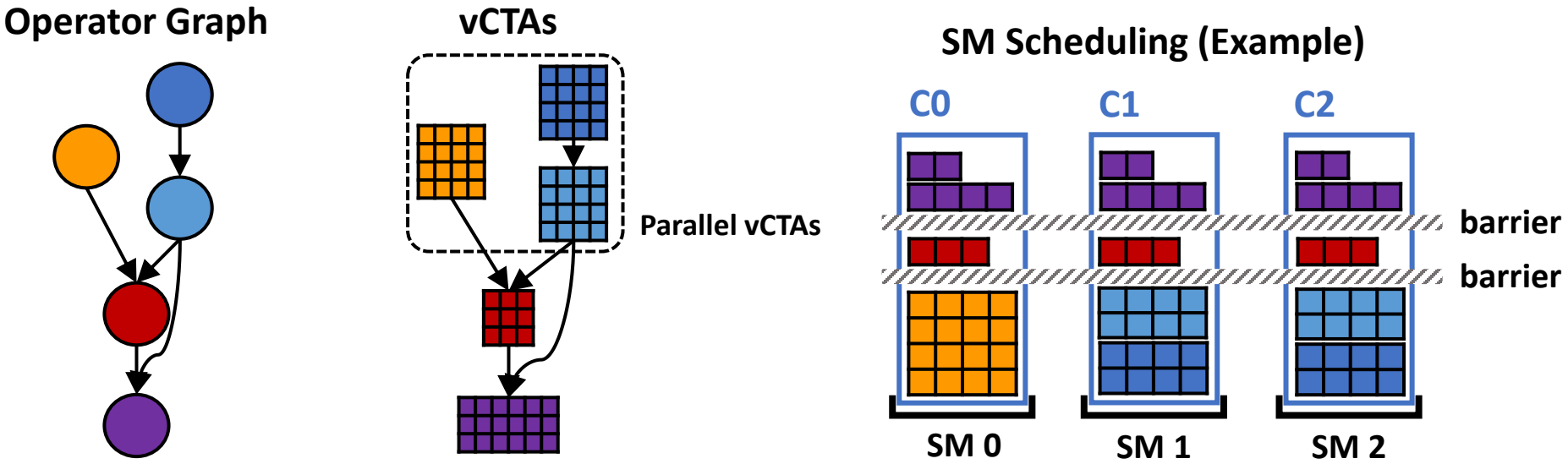
1. Abstract all GPU operators into a set of ***virtual CTAs*** **(vCTAs)**



Scheduling vCTAs to SMs

- Software-defined GPU task scheduler: *how does it work?*

2. Analyze the operator graph to grep dependencies between vCTAs
3. Schedule vCTAs across SMs considering their orders & dependencies

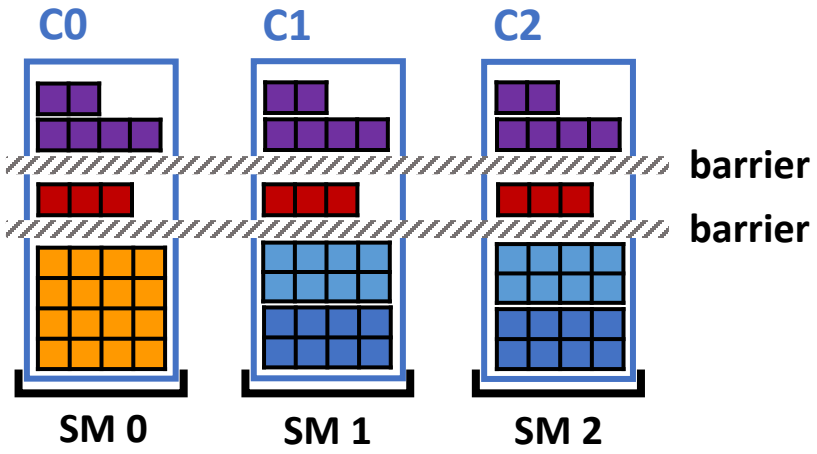


Code Generation & Compilation

- Software-defined GPU task scheduler: *how does it work?*

4. Generate loop kernel code according to the scheduling

SM Scheduling



Loop Kernel Code (in SIMT manner)

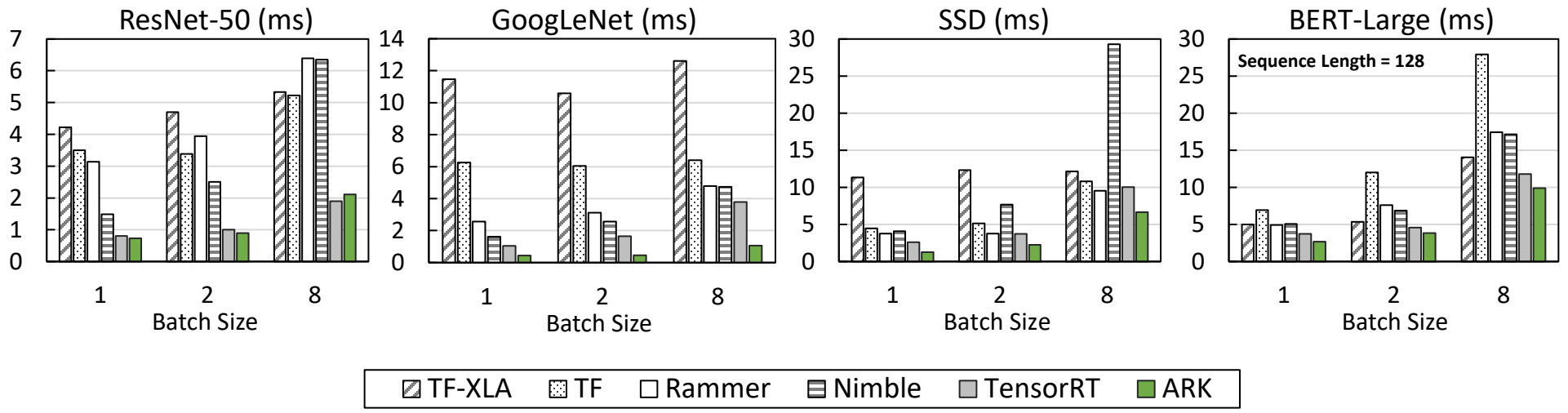
```
if (sm==0) { x16;}
if (sm==1) { x8; x8;}
if (sm==2) { x8; x8;}
barrier();
 x3;
barrier();
 x6;
```

See more details on the scheduler & loop kernel implementation in our paper

Design 2. Autonomous GPU Execution Control

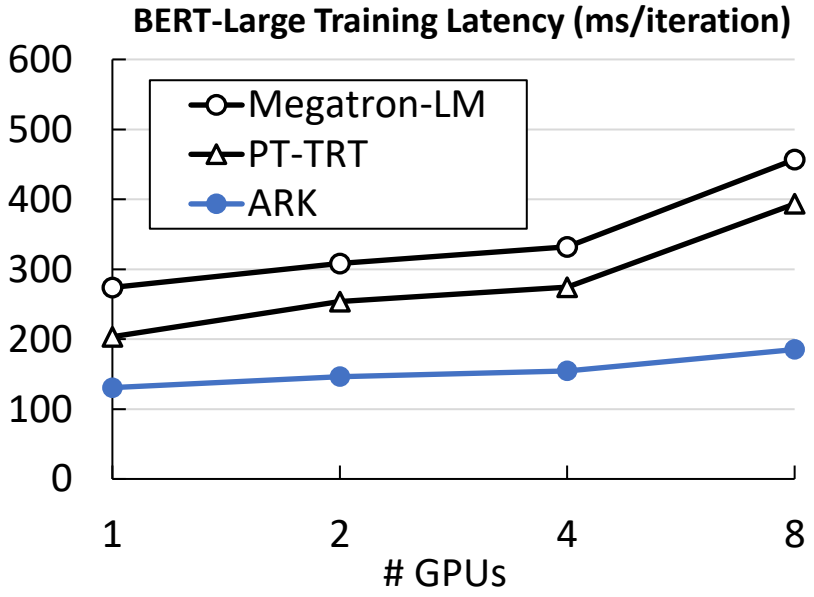
Single-GPU Microbenchmark

- Compare inference latency of 1-GPU models
- GPU-driven system achieves comparable or better perf than baselines
  - **1.1x ~ 3.5x** faster than TensorRT

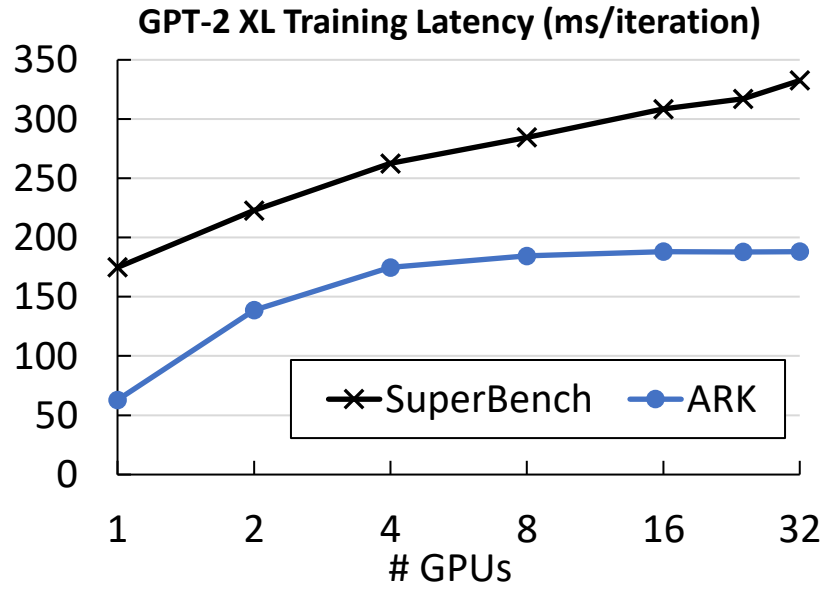


Evaluation: Data-parallel Training

- Data-parallel training
  - BERT-Large: **2.1x** faster than PT-TRT (PyTorch+TensorRT+NCCL) using 8x V100
    - **64% of the gain comes from removing the I/O interference**
  - GPT-2 XL: **1.7x** faster than SuperBench (PyTorch+NCCL) using 32x A100



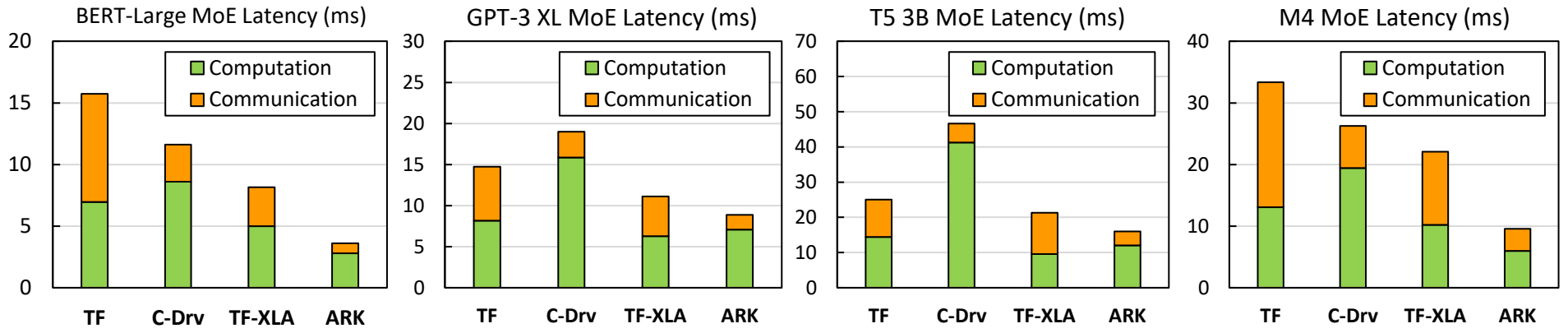
Used 8x NVIDIA V100 GPUs (PCIe Gen3, single NUMA domain), mixed-precision, sequence length 384, per-GPU batch size 10



Used 4x Azure NDv4 SKUs (8x NVIDIA A100-NVLink GPUs per node), mixed-precision, sequence length 384, per-GPU batch size 4

Evaluation: Tensor-parallel Inference

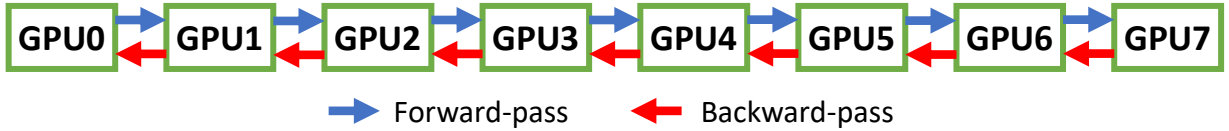
- Mixture-of-Experts model inference (2 GPUs, 1 expert/GPU)
  - BERT-Large, GPT-3 XL, T5 3B, and M4
- ARK vs. TF-XLA: overall **1.2x ~ 2.3x** faster
  - **1.7x ~ 3.3x** faster communication



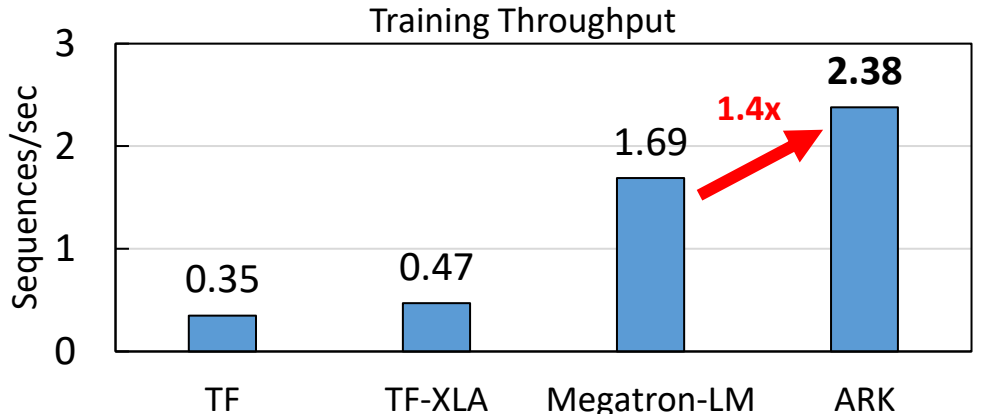
Used 2x NVIDIA V100 GPUs (PCIe Gen3, PIX linked) + Intel Xeon Gold 6240R CPU @ 2.40GHz, mixed-precision, batch size 1

Evaluation: Pipeline-parallel Training

- GPT-3 6.7B model training
  - Each GPU runs 4 Transformer layers
  - Pipeline 5 stages & each stage batch size 1



- Most improvement comes from computational gain due to large message sizes



Used 8x NVIDIA V100 GPUs (PCIe Gen3, single NUMA domain) + Intel Xeon Gold 6240R CPU @ 2.40GHz, mixed-precision, sequence length 2048

Conclusion

- Tackle control overhead on CPU and I/O overhead on GPU
- GPU-controlled DMA and GPU-driven system design
- Outperforms in various distributed DL scenarios

ARK is going to be open source: https://github.com/microsoft/ark

WE ARE HIRING! Please contact me offline or changhohwang@microsoft.com