Cloud Computing
GPUs and GPU cloud

Minchen Yu
SDS@CUHK-SZ
Fall 2024

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

SCHOOL OF
DATA SCIENCE
數據科學學院

Outline

GPU introduction and architecture

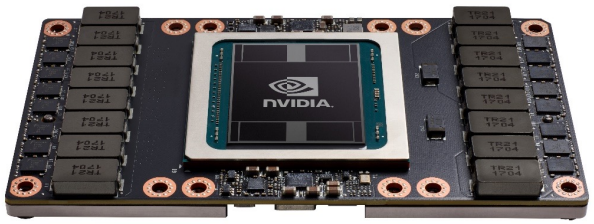
GPU virtualization

GPU cluster in the cloud

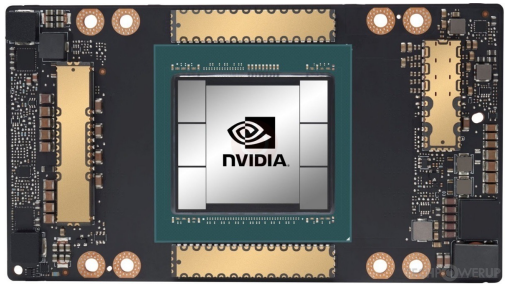
GPU introduction

What is a GPU?

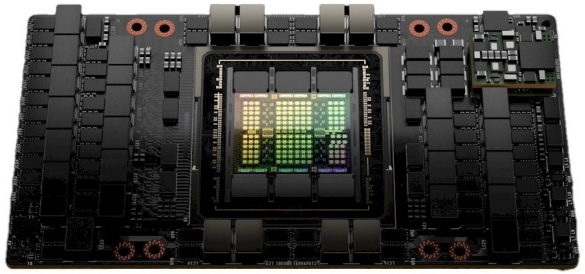
- GPU = Graphics Processing Unit
  - Accelerator for raster based graphics (OpenGL, DirectX)
  - Highly programmable
  - Commodity hardware
  - 100s of ALUs; 10s of 1000s of concurrent threads



NVIDIA Volta: V100



NVIDIA Ampere: A100



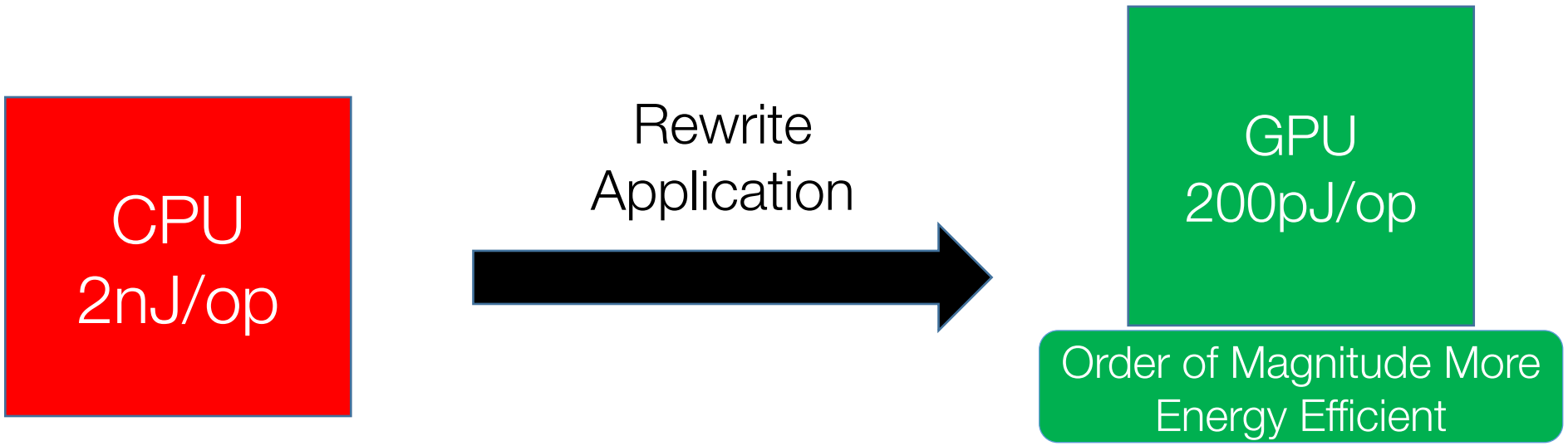
NVIDIA Hopper: H100

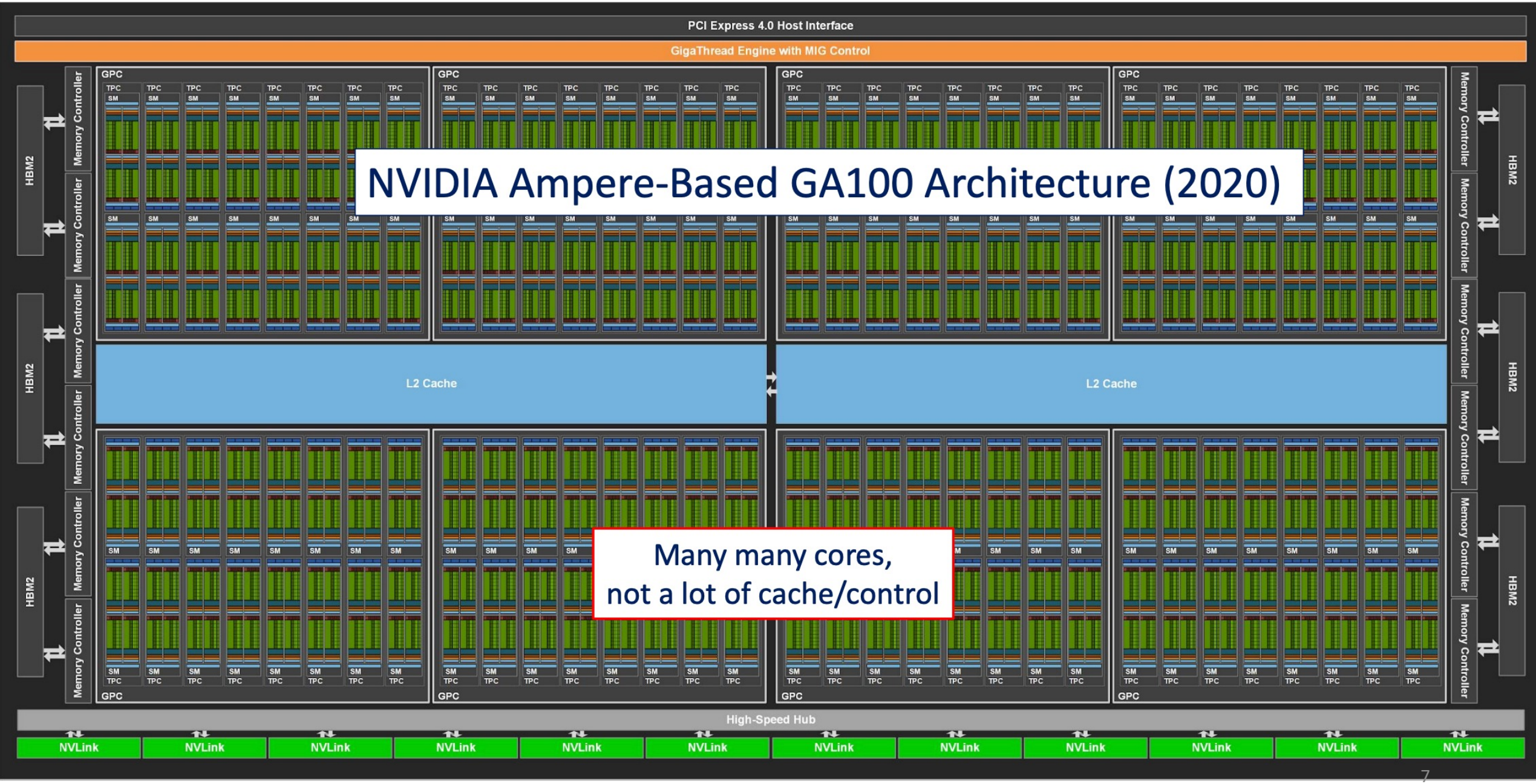
Graphic Processing – Some History

- 1990s: Real-time 3D rendering for video games were becoming common
- 3D graphics processing is immensely computation-intensive
- Before 3D accelerators (GPUs) were common, CPUs had to do all graphics computation, while maintaining framerate!
  - Many tricks were played
- Much of 3D processing is short algorithms repeated on a lot of data
  - pixels, polygons, textures, …
- Dedicated accelerators with simple, massively parallel computation

Why GPU?

- GPU uses larger fraction of silicon for computation than CPU.
- At peak performance GPU uses order of magnitude less energy per operation than CPU.

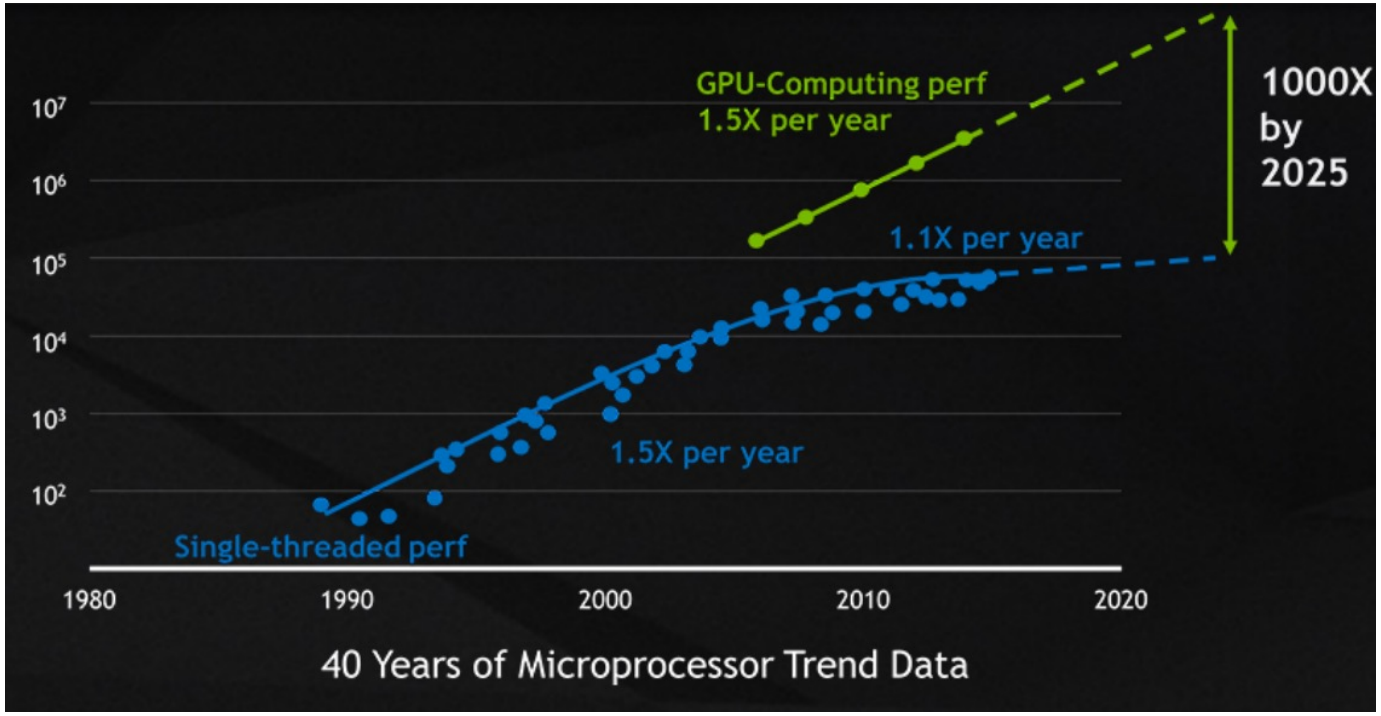



PCI Express 4.0 Host Interface
GigaThread Engine with MIG Control
NVIDIA Ampere-Based GA100 Architecture (2020)
GPC
TPC
SM
L2 Cache
Memory Controller
HBM2
Many many cores,
not a lot of cache/control
High-Speed Hub
NVLink

Peak Performance vs. CPU

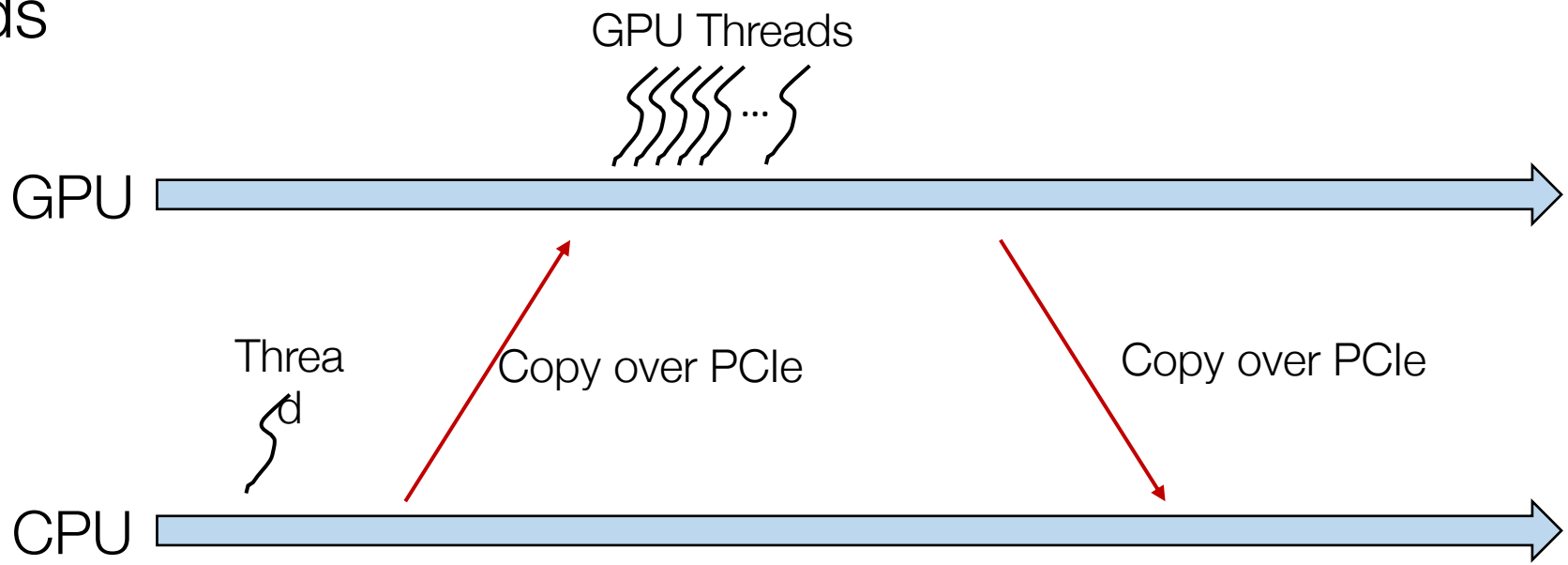
	Throughput	Power	Throughput/Power
Intel Skylake	128 SP GFLOPS/4 Cores	100+ Watts	~1 GFLOPS/Watt
NVIDIA V100	15 TFLOPS	200+ Watts	~75 GFLOPS/Watt

Also,



Massively Parallel Architecture For Massively Parallel Workloads!

- NVIDIA CUDA (Compute Uniform Device Architecture) – 2007
  - A way to run custom programs on the massively parallel architecture!
- OpenCL specification released – 2008
- Both platforms expose synchronous execution of a massive number of threads



GPU and deep learning

- Good match between GPU and deep learning
- DNNs are matrix multiplication-intensive
- Back to CNNs
  - “Building High-Level Features Using Large Scale Unsupervised Learning” 16,000 CPU cores in 2012
  - “Deep learning with COTS HPC systems” Two CPU cores and two GPUs in 2013
- Now widely used in modern deep learning models (e.g., LLMs, etc.)

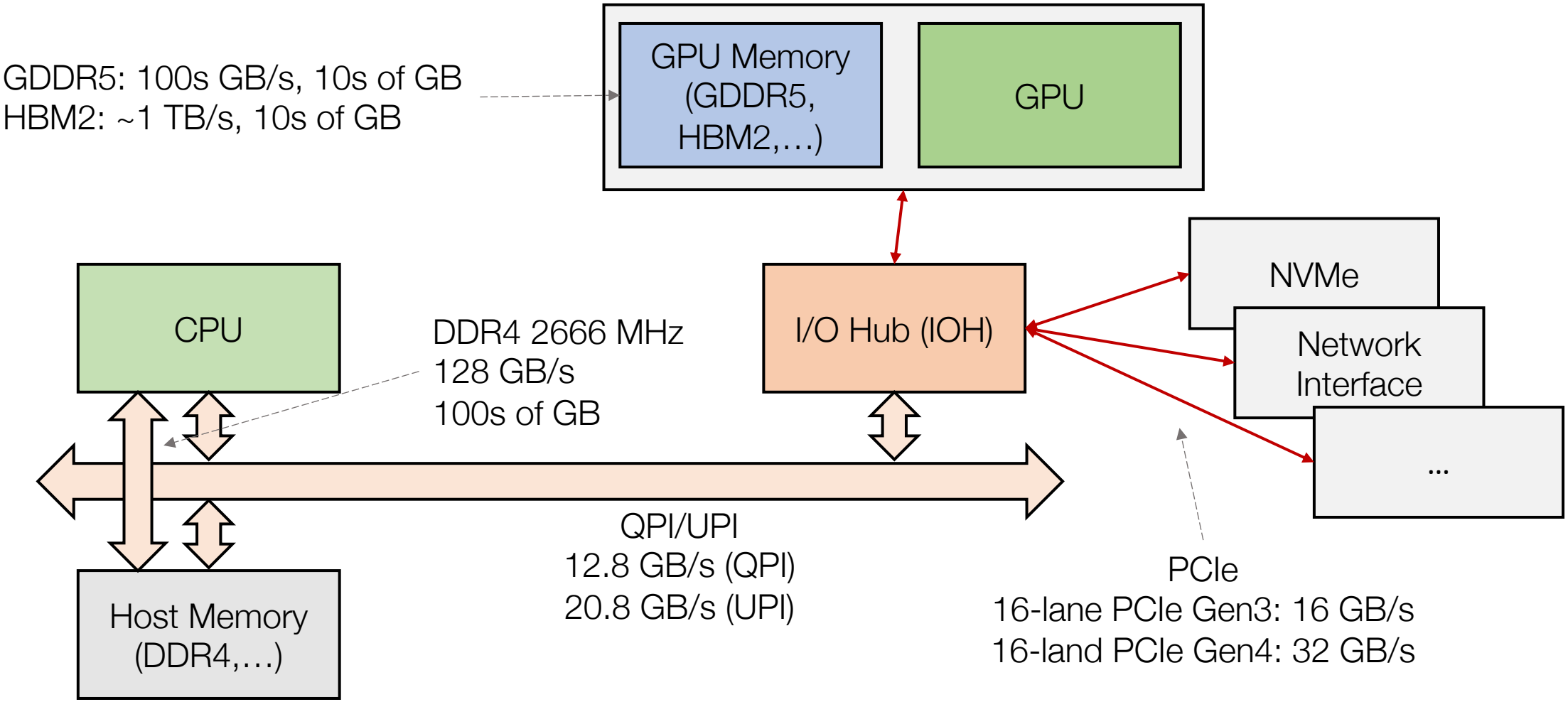
GPU architecture

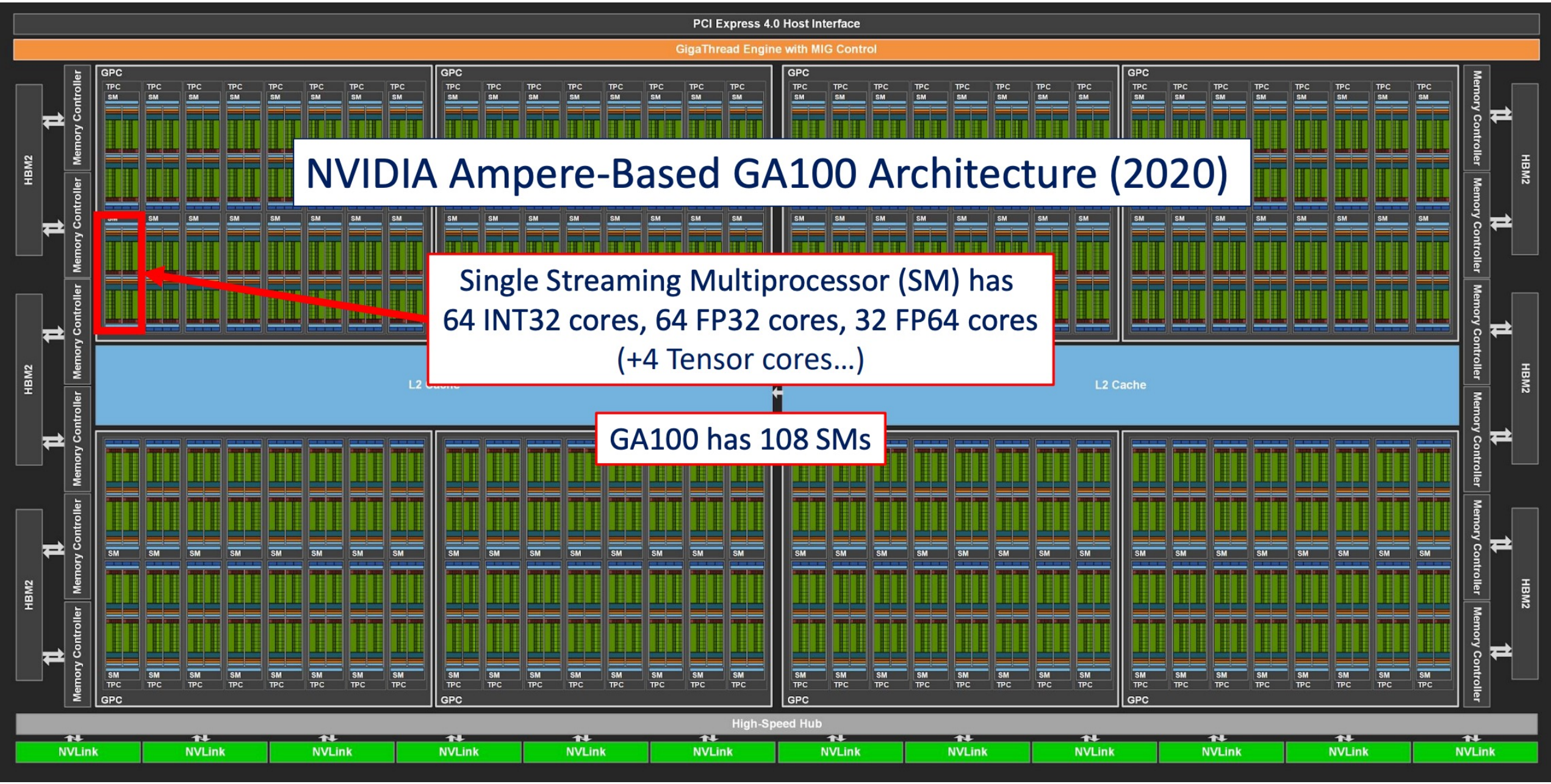
GPU architecture

- GPUs have thousands of threads running concurrently at GHzs
- Much simpler processor architecture
  - Dozens of threads scheduled together in a SIMD fashion
  - Much simpler microarchitecture (doesn't need to boot Linux)
- Much higher power budget
  - CPUs try to maintain 100 W power budget (Pentium 4 till now)
  - Thermal design power (TDP) for modern GPUs around 300 W
    - TDP: Safe level of power consumption for effective cooling

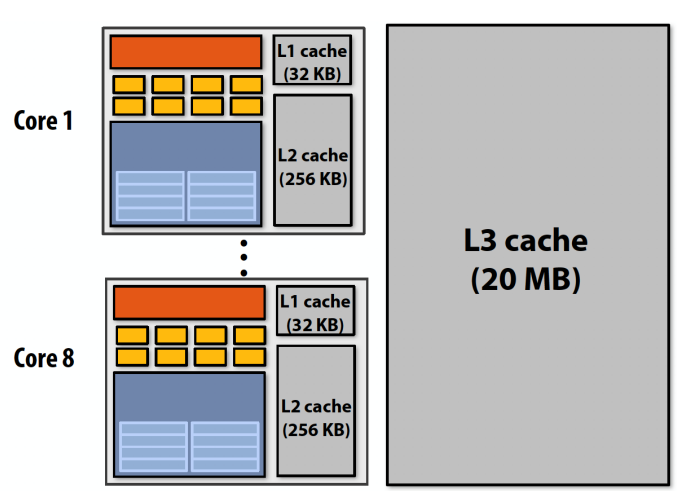
CPU (i7) adding 1 Billion floats: 2.14s, NVIDIA Turing with only one thread: 29.16s

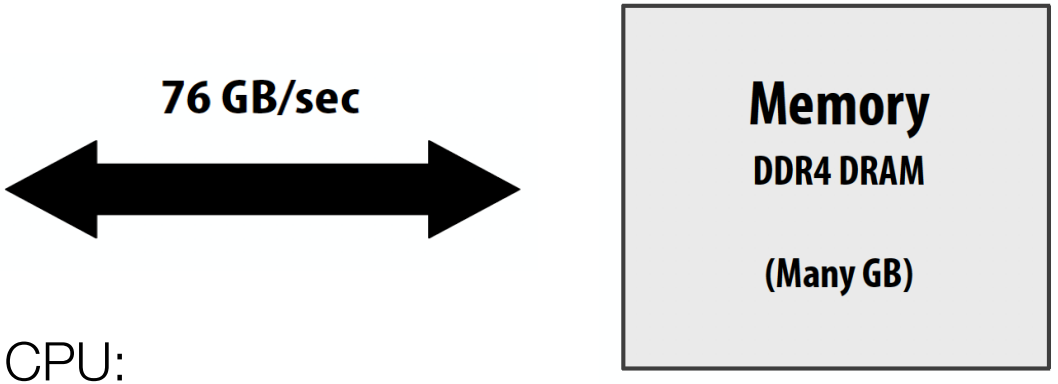
System Architecture Snapshot With a GPU




PCI Express 4.0 Host Interface
GigaThread Engine with MIG Control
NVIDIA Ampere-Based GA100 Architecture (2020)
Single Streaming Multiprocessor (SM) has
64 INT32 cores, 64 FP32 cores, 32 FP64 cores
(+4 Tensor cores…)
GA100 has 108 SMs
L2 Cache
GPC
TPC
SM
HBM2
Memory Controller
High-Speed Hub
NVLink

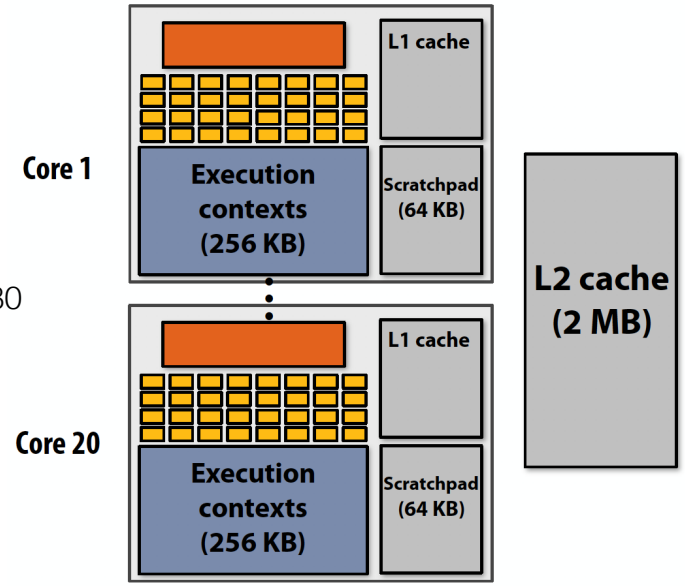
CPU v.s. GPU Memory Hierarchies

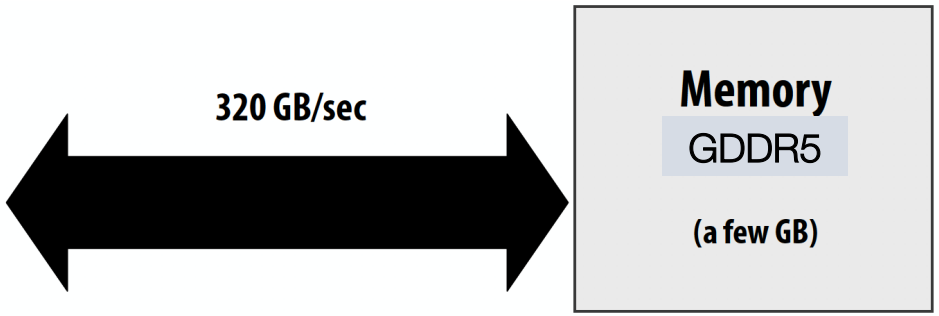




CPU:
Big caches, few threads per core, modest memory BW
Rely mainly on caches and prefetching

NVIDIA GTX 1080 (Pascal)

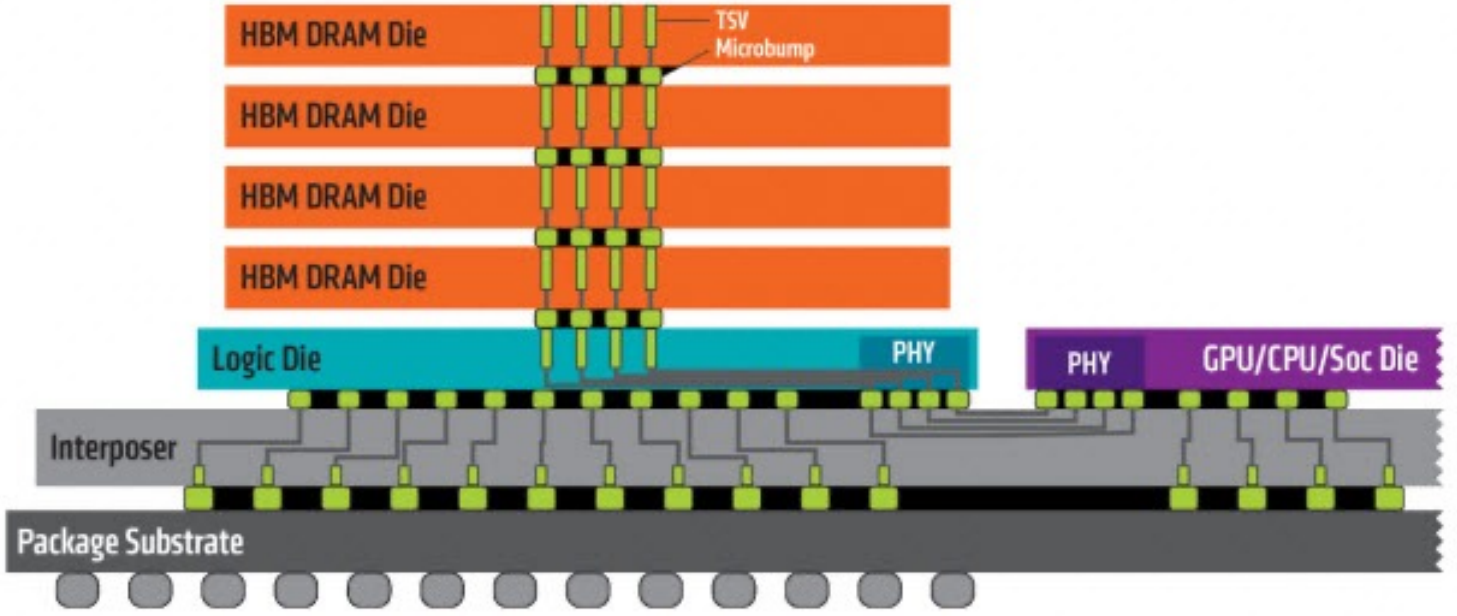




GPU:
Small caches, many threads, huge memory BW
Rely heavily on multi-threading for performance

High-Performance Graphics Memory

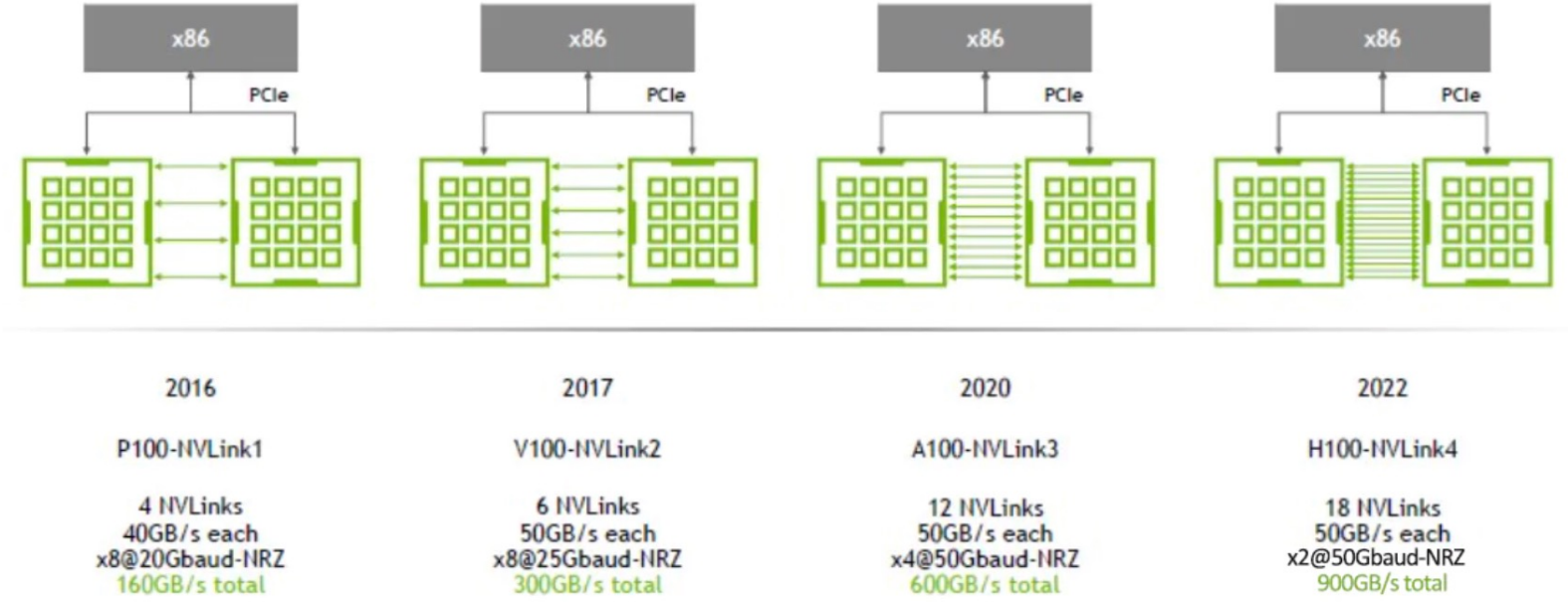
- Modern GPUs even employing 3D-stacked memory via silicon interposer
  - Very wide bus, very high bandwidth
  - e.g., HBM2 in Volta



Graphics Card Hub, “GDDR5 vs GDDR5X vs HBM vs HBM2 vs GDDR6 Memory Comparison,” 2019

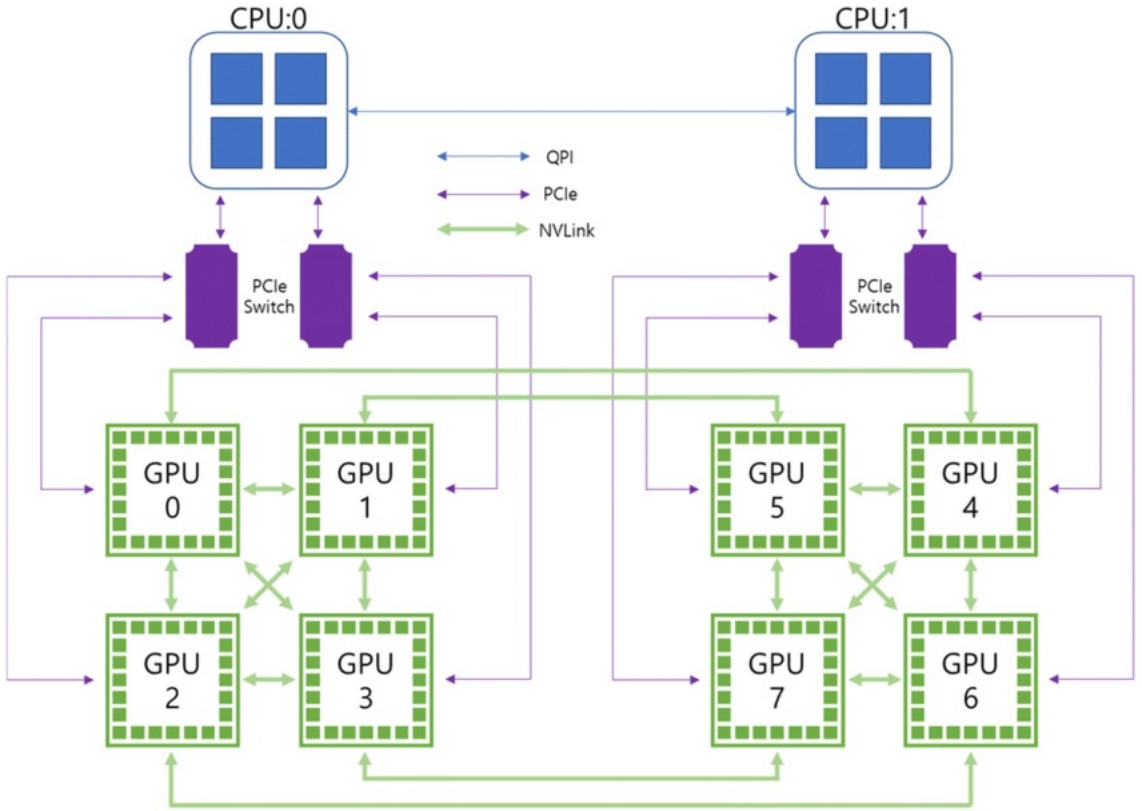
High-speed interconnect between GPUs

- NVLink
  - bidirectional, direct GPU-to-GPU interconnect



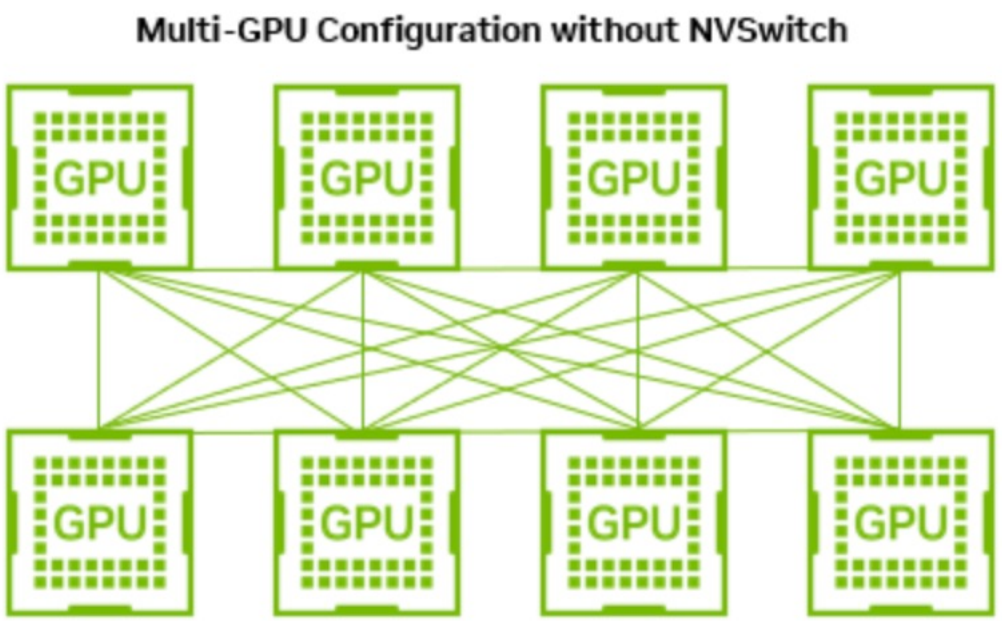
Multi-GPU topology

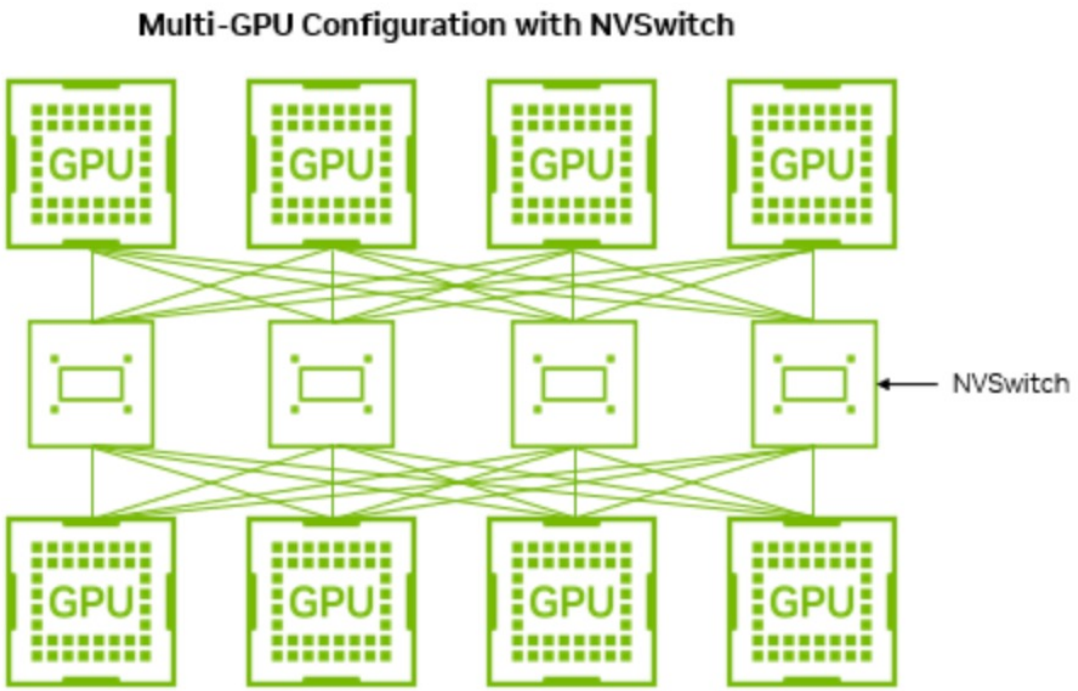
- Multi-GPU topology before DGX-A100



Multi-GPU topology

- NVSwitch (NVLink Switch) connects multiple NVLinks to provide all-to-all GPU communication at full NVLink speed



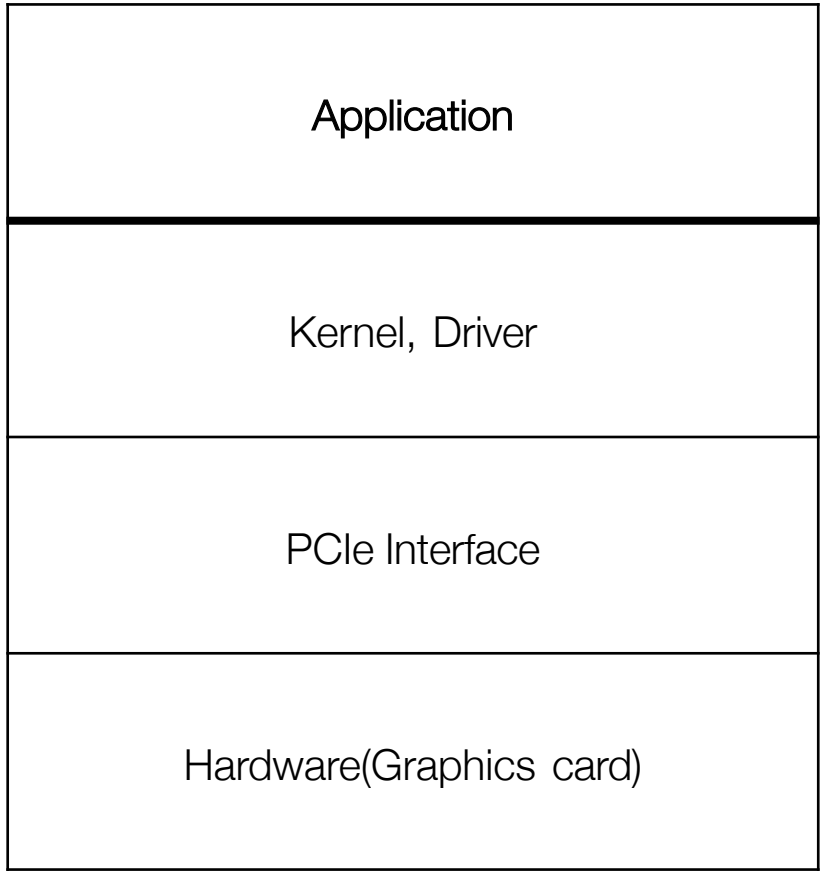


GPU virtualization

GPU virtualization

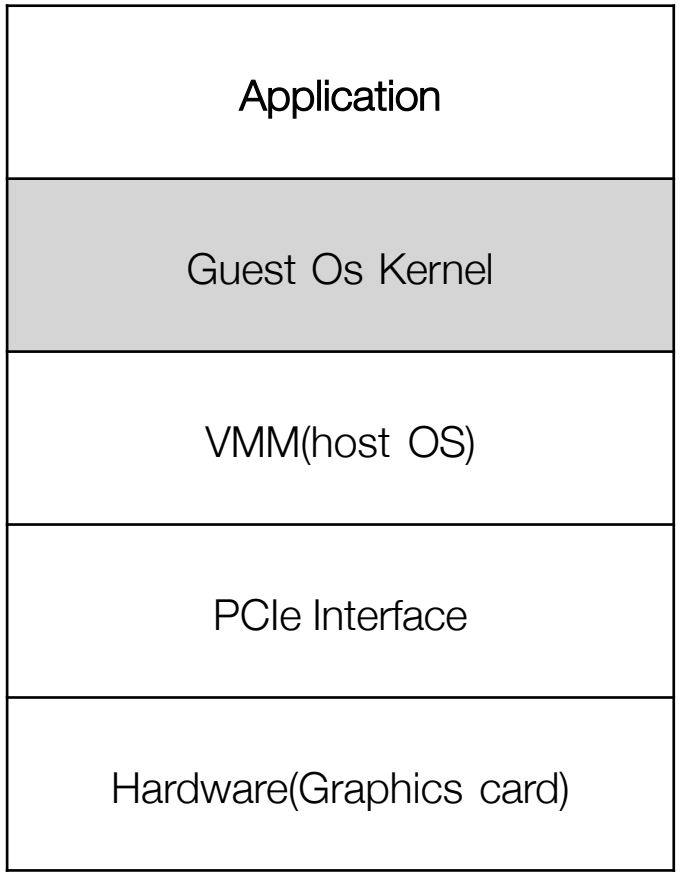
- Hot direction in today's cloud
  - GPU sharing among cloud users
- Some solutions at various levels
  - Provided by hardware vendors, e.g., Nvidia MPS, MIG
  - Enabled at application level, e.g., vCUDA
  - …

Without Virtualization



GPU virtualization is more like I/O virtualization

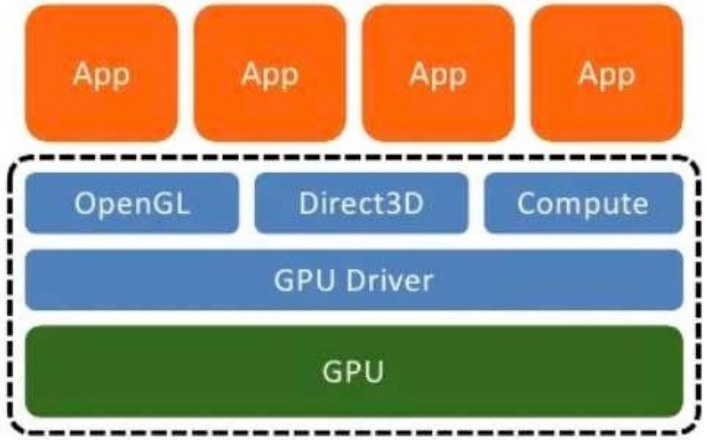
- GPU is connected to CPU via PCIe



CPU	GPU
Fidelity Performance Safety	Fidelity Performance (safety)Multiplexing (safety)Interposition

Unique challenges

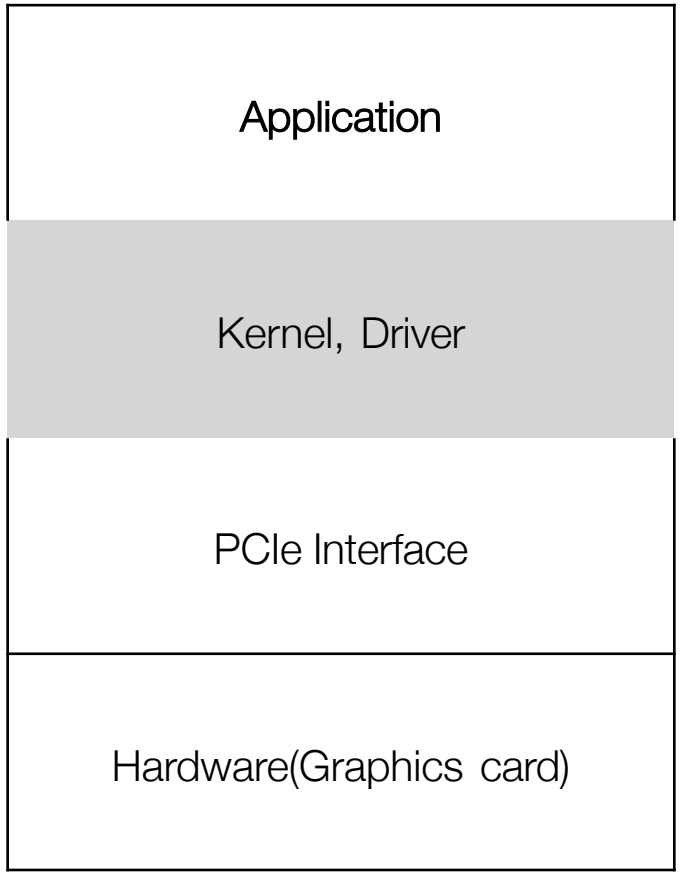
- Hardware specs
  - Diverse, changes frequently
  - Closely guarded secret
  - Speed vs. portability
- Hardware state
  - Up to gigabytes of data
  - Highly device-specific format
  - In-progress DMA and computation

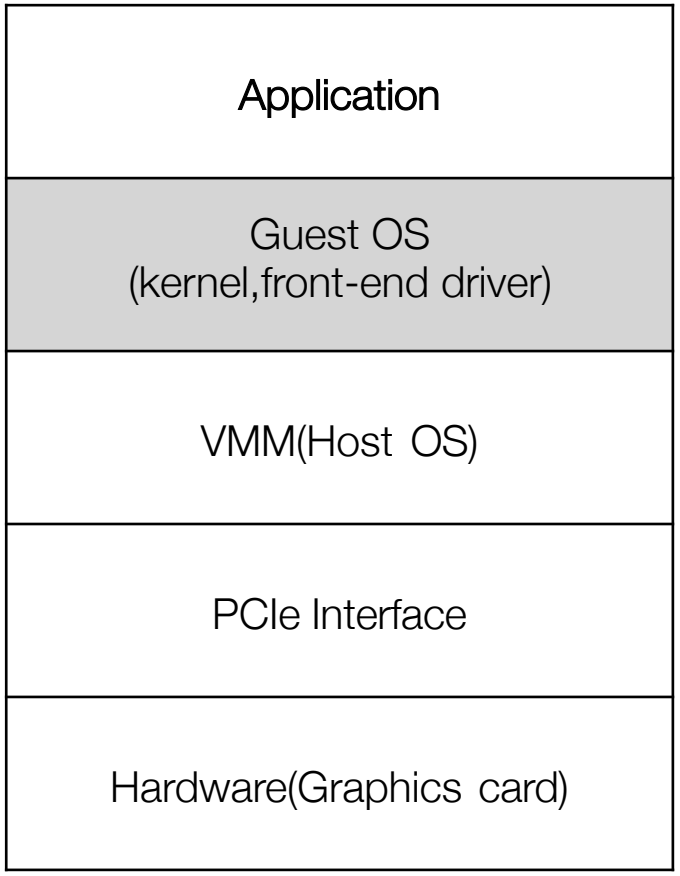


Taxonomy of GPU virtualization

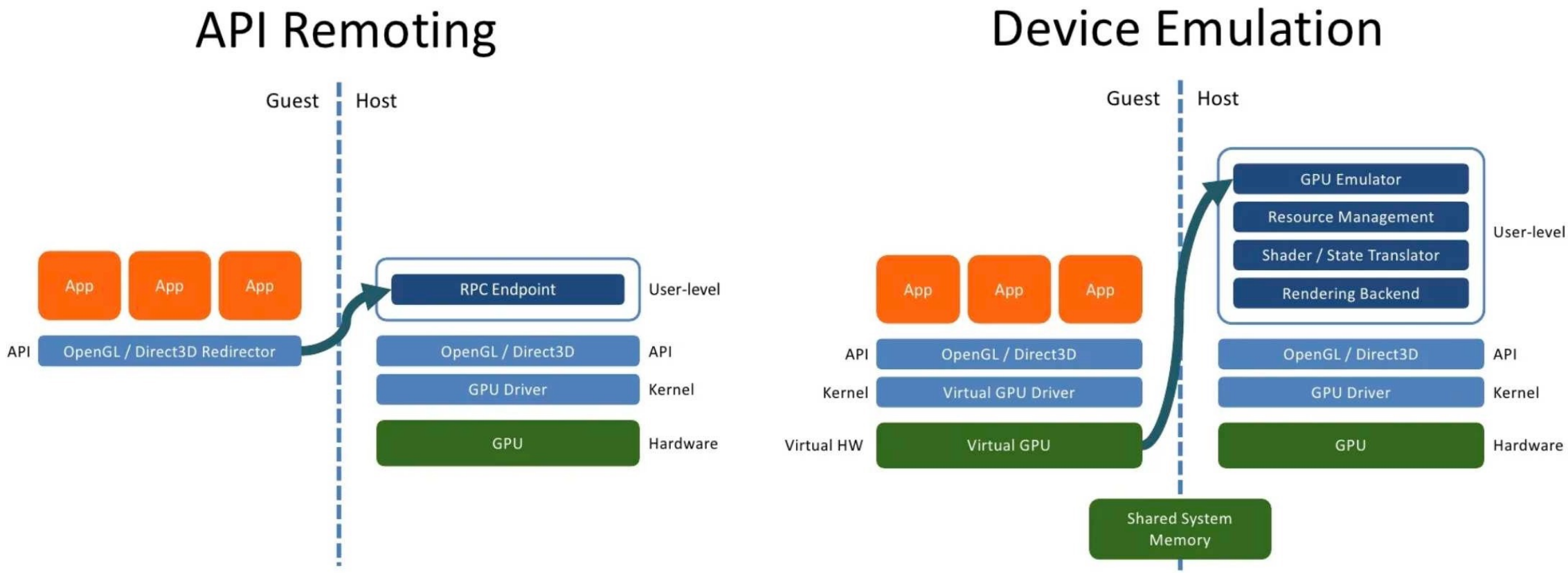
- Front-end
- Back-end

Front-end Virtualization





Front-end Virtualization

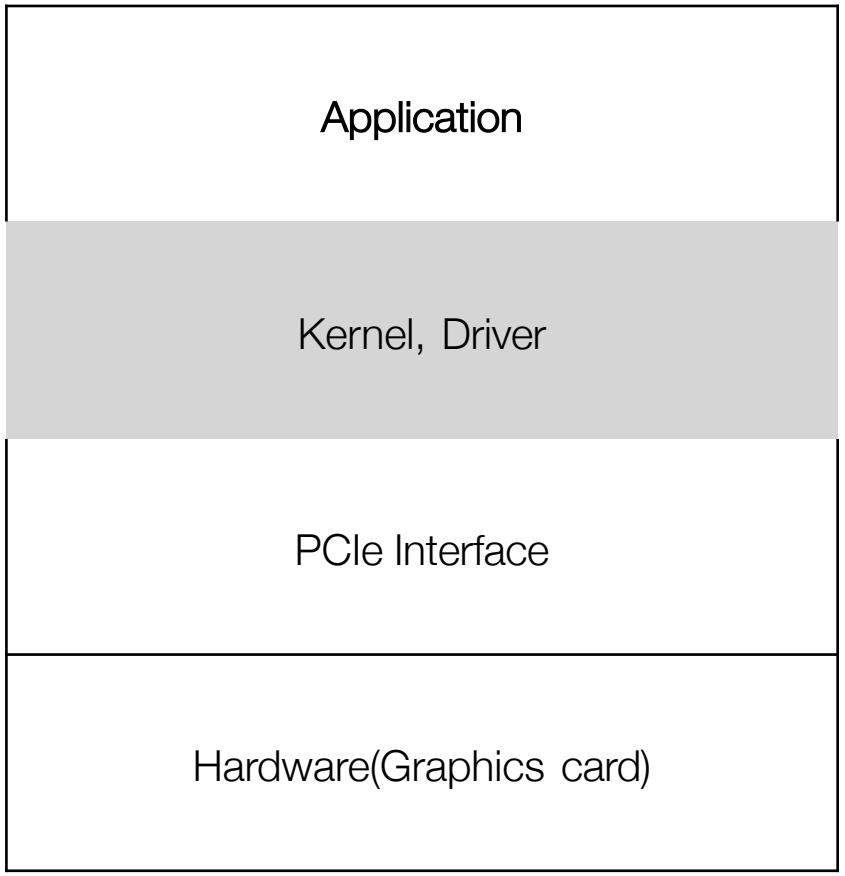


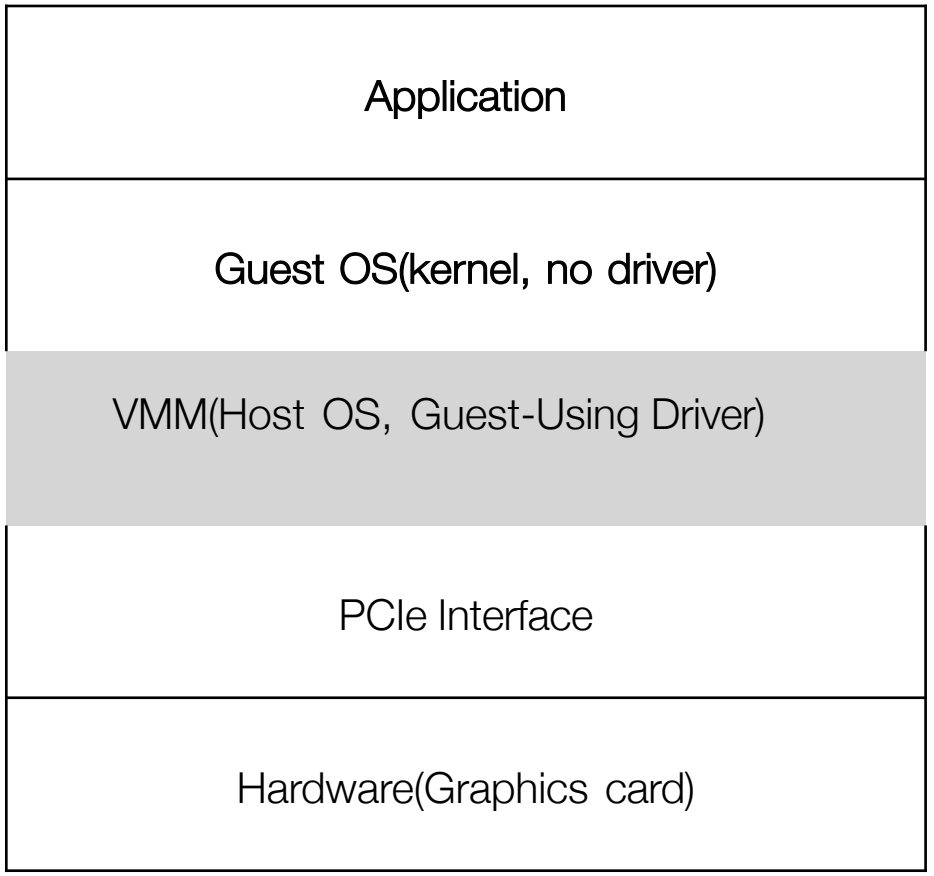
https://www.slideshare.net/guestb3fc97/gpu-virtualization-on-vmwares-hosted-io-architecture-presentation

Front-end Virtualization

- Two ways to enable the designed driver in the guest OS
- Pure API remoting
  - Put the api ability to somewhere else. Could be vmm, could be not.
  - Large data transfer, Low interposition. Easy implementation.
- Pure device emulation
  - Emulate the graphics card behavior by cpu.
  - Highly complicated and undocumented behavior. Good interposition.

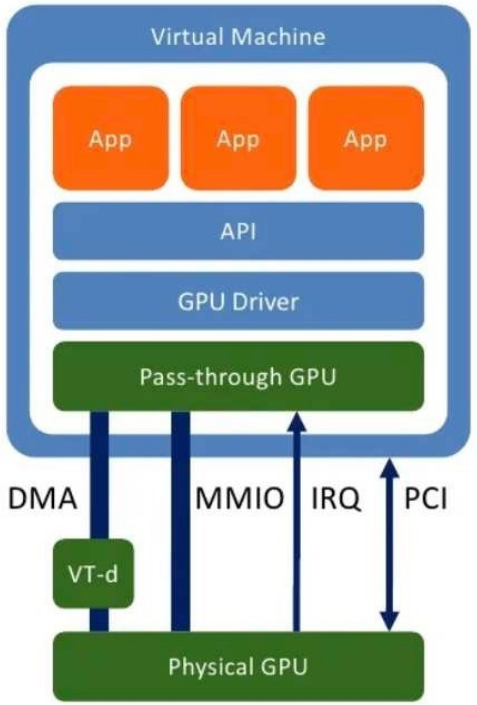
Back-end Virtualization



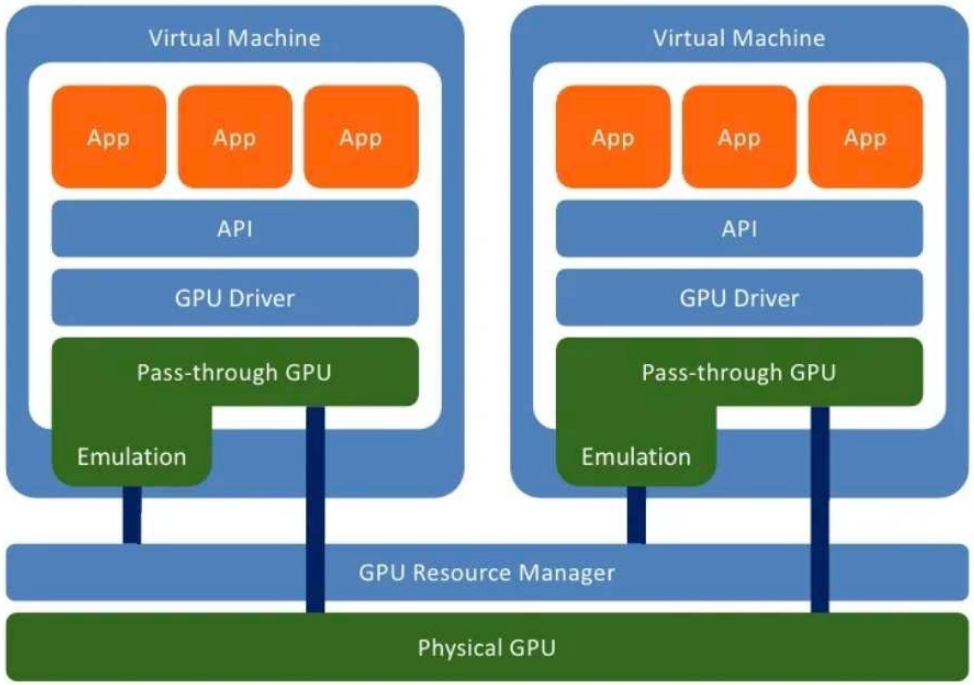


Back-end Virtualization

Fixed pass-through



Mediated pass-through

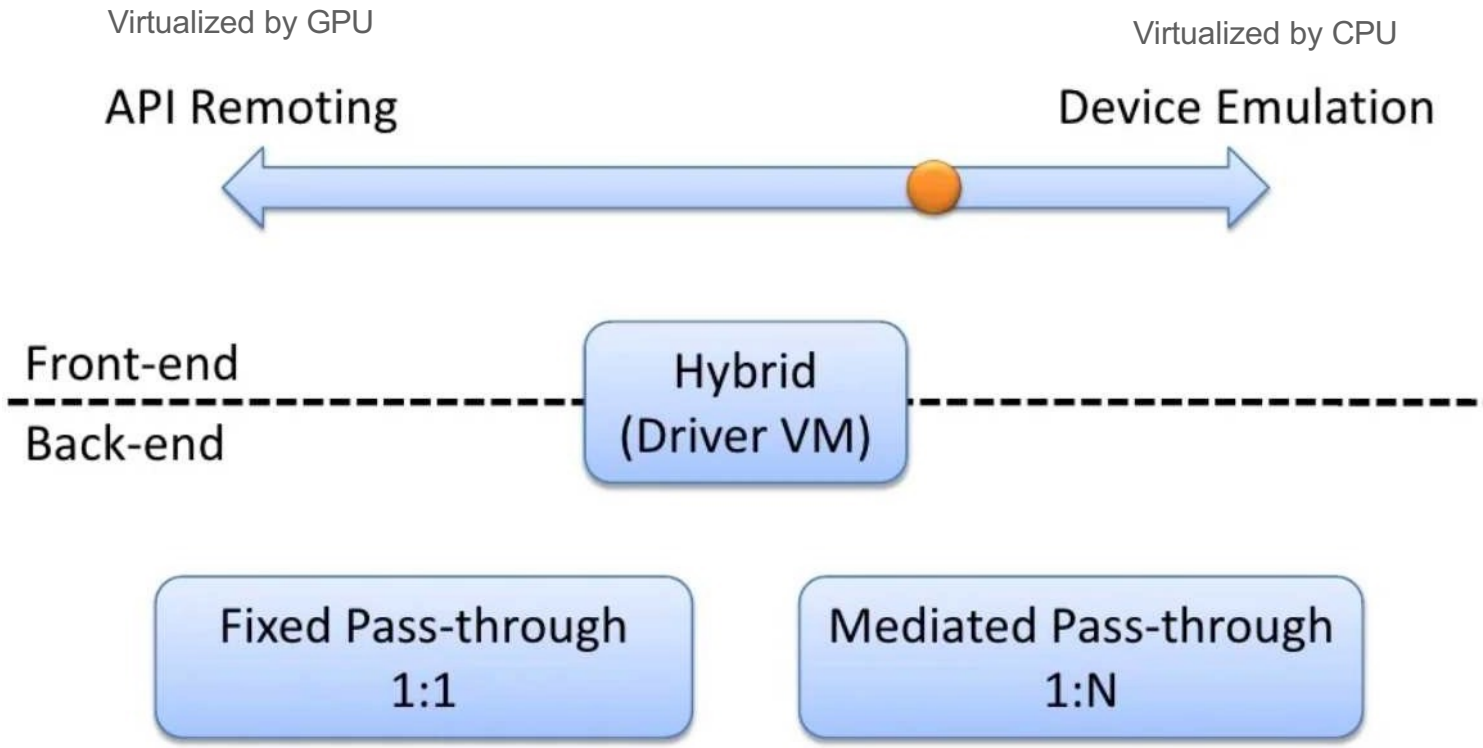


https://www.slideshare.net/guestb3fc97/gpu-virtualization-on-vmwares-hosted-io-architecture-presentation

Back-end Virtualization

- Hard to multiplex and interposition
- Multiplex strategy
  - fixed pass-through: the permanent association of a virtual machine with full exclusive access to a physical GPU.
  - mediated pass-through: just give different vm different context. So they can multiplex.
    - Hard to map to different virtual machines with low overheads and the host/hypervisor must have enough of a hardware driver to allocate and manage GPU contexts.

GPU Virtualization Taxonomy



https://www.slideshare.net/guestb3fc97/gpu-virtualization-on-vmwares-hosted-io-architecture-presentation

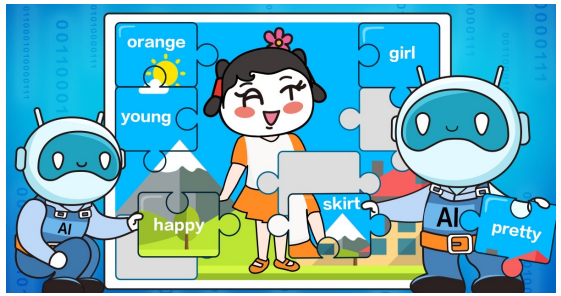
GPU cluster in the cloud

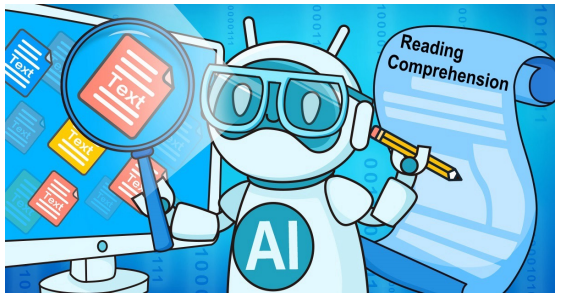
“MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters” in NSDI’22

Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, Yu Ding

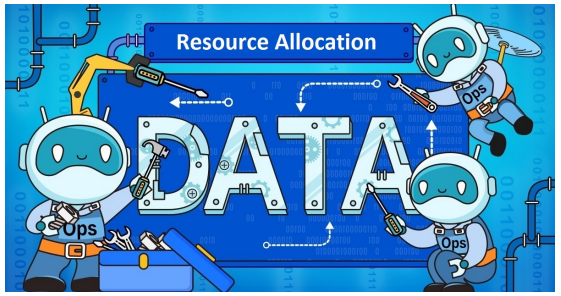
Production ML Workloads

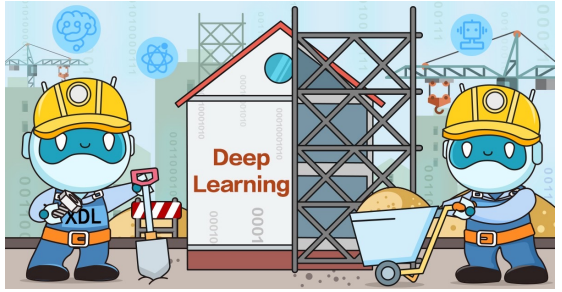
- Machine Learning (ML) is ubiquitous: CV, NLP, Recommend ...





- ML jobs can scale to hundreds of GPUs, requesting TBs of data.

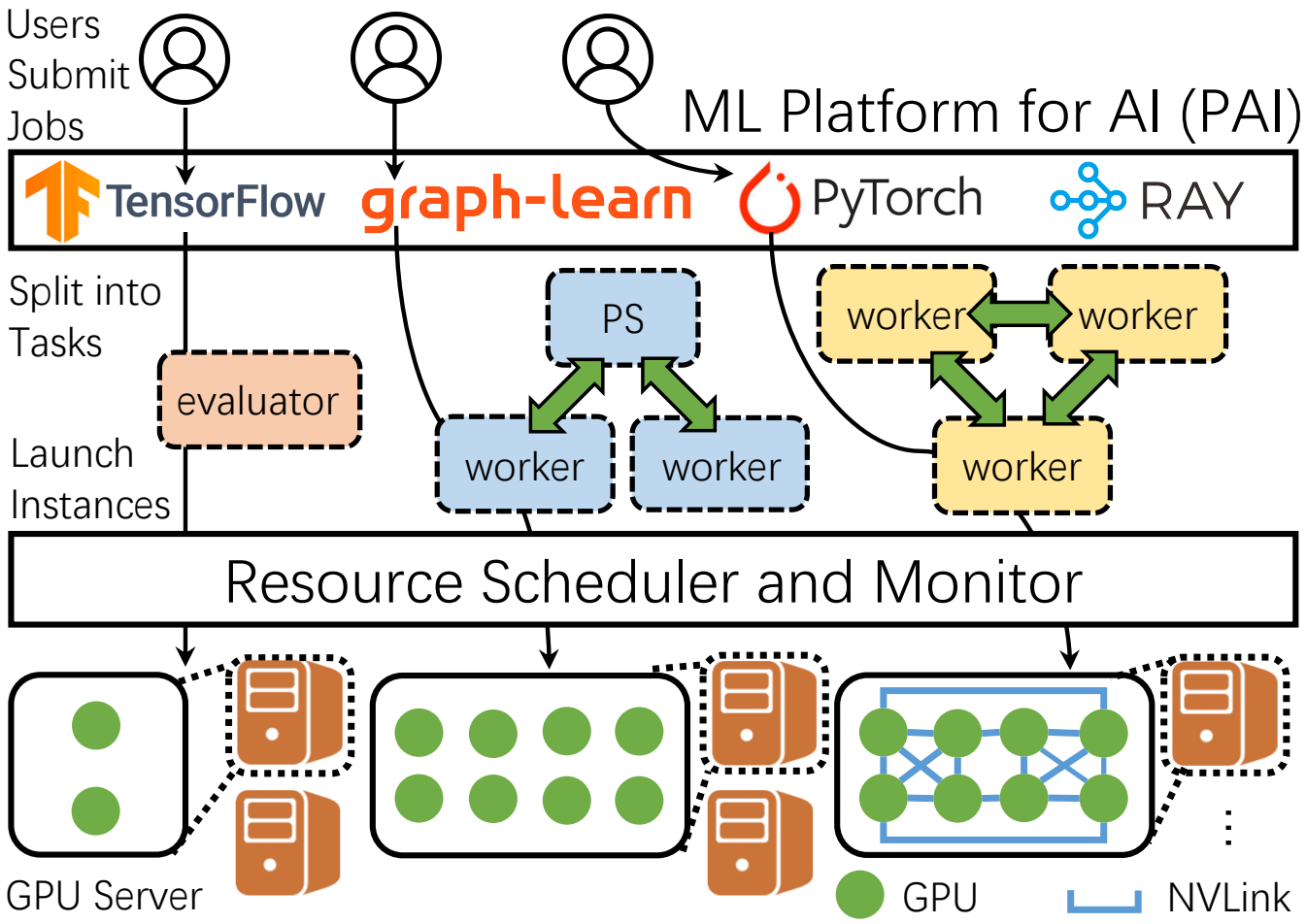




Images credit to: https://alibabatech.medium.com/

PAI: a Cloud-based MLaaS Platform

- All-in-one platform for training & inference
- Support a variety of ML frameworks
- Allocate containerized instances to hetero. nodes



Trace Basics

Traces are released at: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-gpu-v2020

Trace Overview

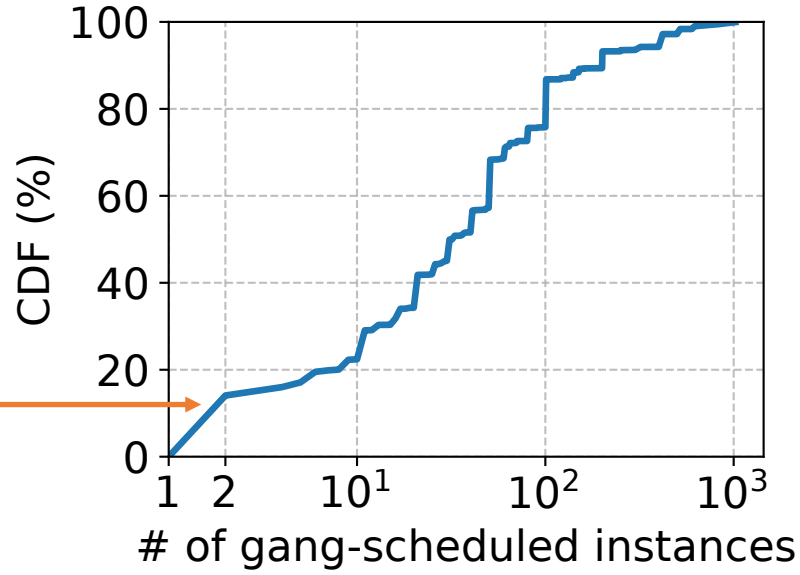
- Collected in Jul. and Aug. 2020 on 1800+ nodes, 6700+ GPUs

† are equipped with NVLink

System	#CPUs	Mem (GiB)	#GPUs	GPU type	#Nodes
PAI	64	512	2	P100	798
	96	512	2	T4	497
	96	512	8	Misc.	280
	96	384	8	V100M32†	135
	96	512/384	8	V100†	104
	96	512	0	N/A	83

Trace Overview

- Over 1300 users submitted 1.2M tasks with 7.5M instances
  - Allow GPU sharing & type specifying
  - 85% Instances are gang-scheduled

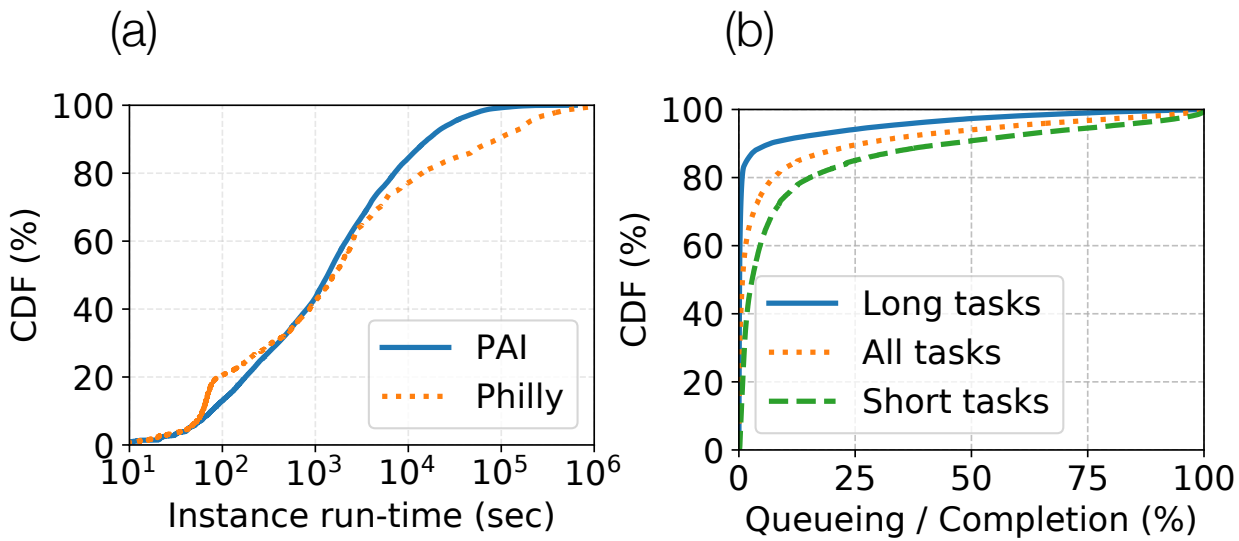


A sample job

task_name	num_inst	plan_cpu	plan_mem	plan_gpu	gpu
worker	20	800	30000	100	T4
ps	25	1500	40000	N/A	N/A

Run-time and Queueing delays

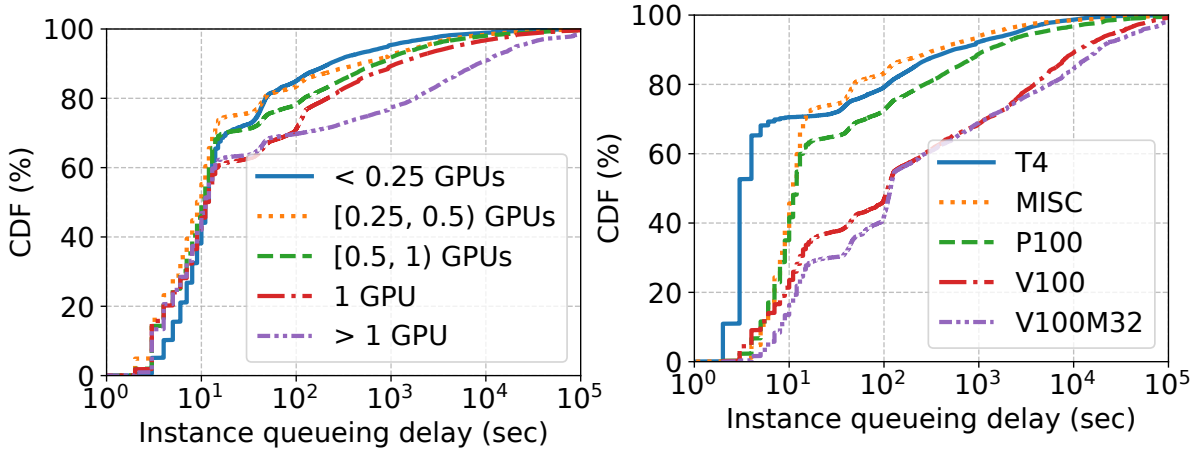
- Instance run-time ranges from seconds to hours (c.f. Philly[1]) (a)
- Short tasks wait for a larger proportion of their runtime (b)



[1] Jeon, Myeongjae et al. “Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads.” ATC ‘19 https://github.com/msr-fiddle/philly-traces

Run-time and Queueing delays

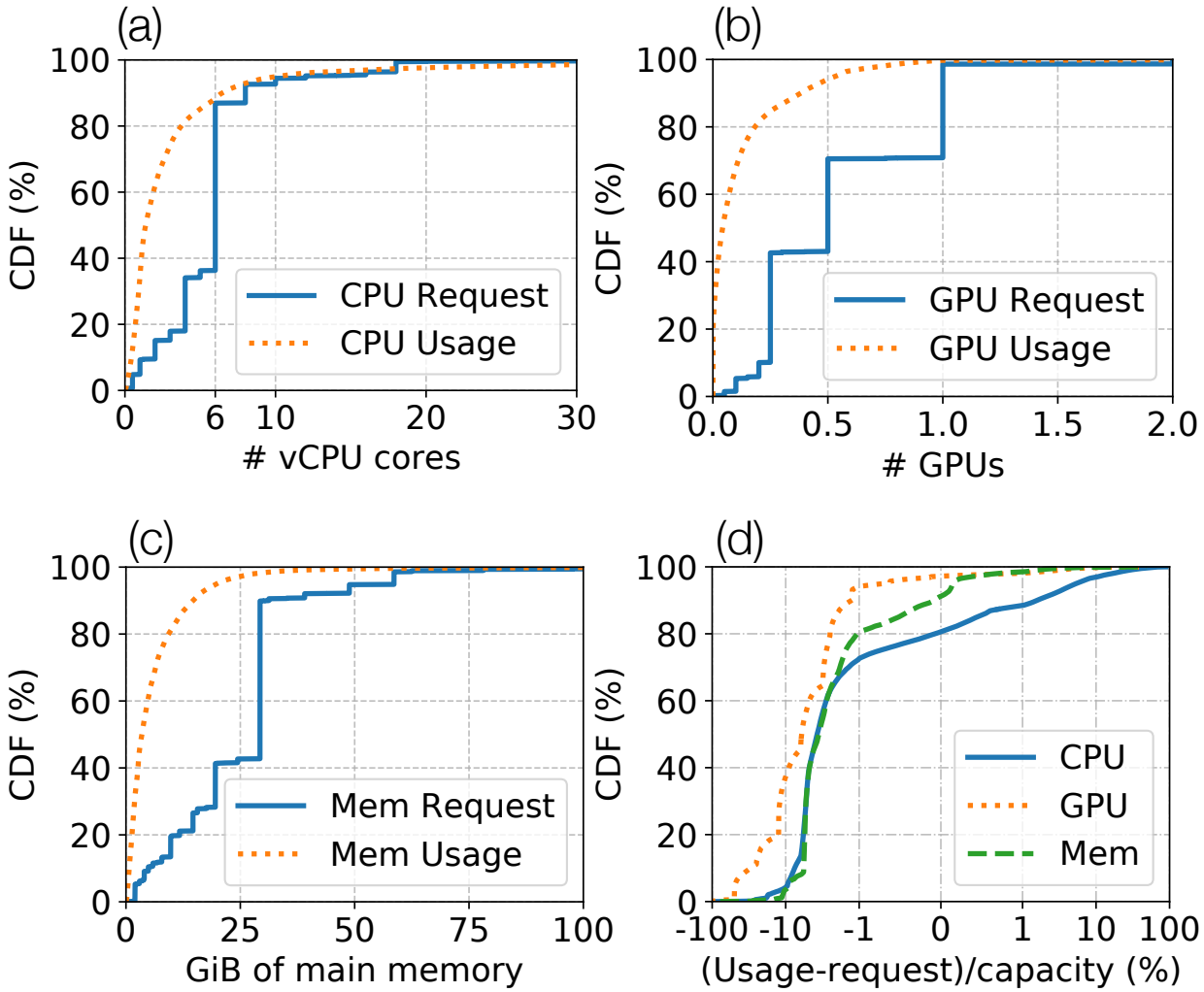
- Tasks requesting full and/or high-end GPUs wait longer



[1] Jeon, Myeongjae et al. “Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads.” ATC ‘19 https://github.com/msr-fiddle/philly-traces

Resource Requests & Usage

- Heavy-tailed distribution of requests: ~20% are “elephants” (solid lines in (a) (b) (c))
- Uneven resource usage: High on CPUs but low on GPUs (dotted lines in (a) (b) (c))
- Usage / Request: 19% instances overuse CPUs (solid lines in (d)); only 3% instances overuse GPUs (dotted lines in (d)).

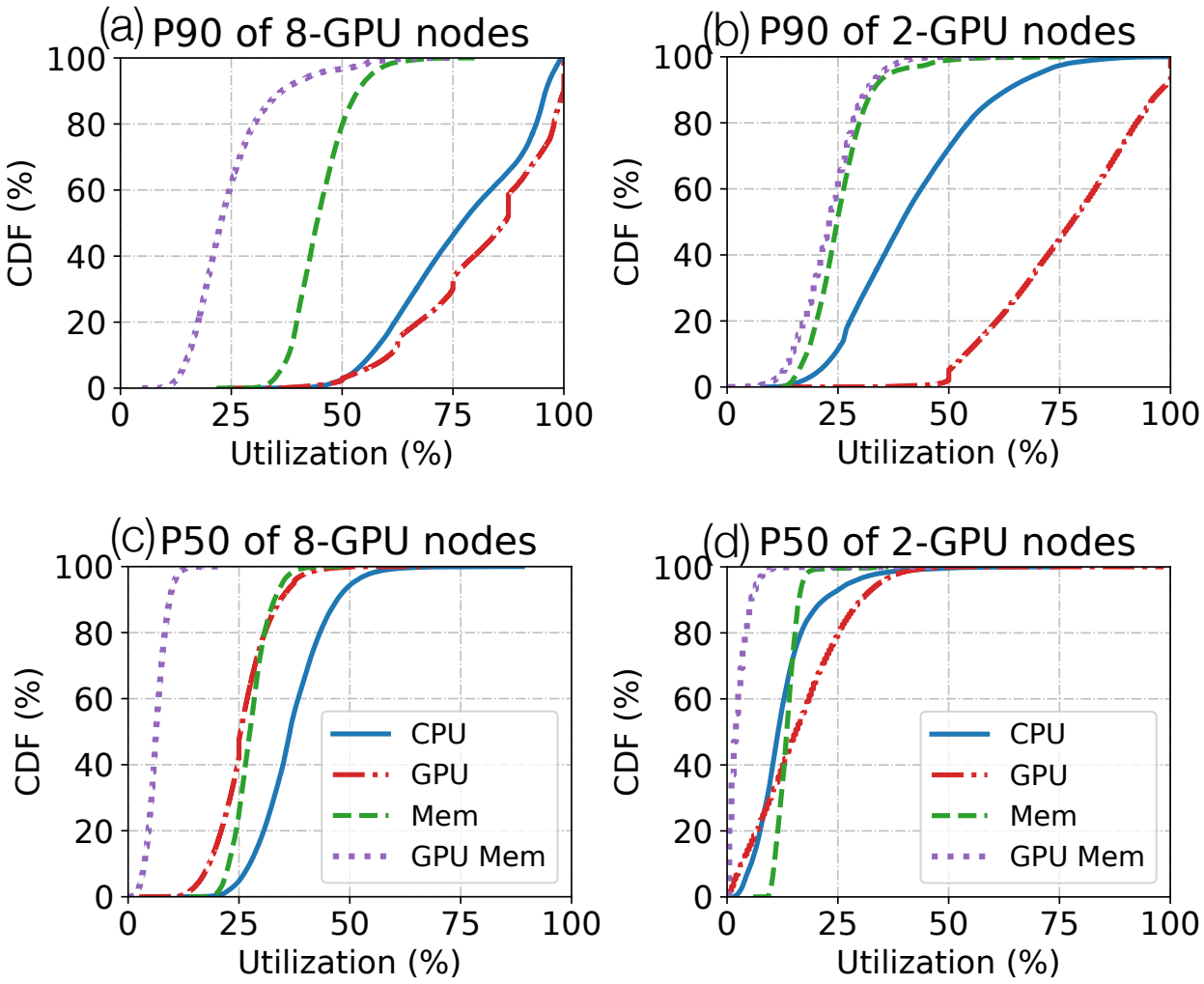


Machine Resource Utilization

- 8-GPU machines (a) (c)
  - High GPU, high CPU utilization
- 2-GPU machines (b) (d)
  - High GPU, medium CPU utilization

- Gap between P90 and P50 (median) util. is large for GPUs

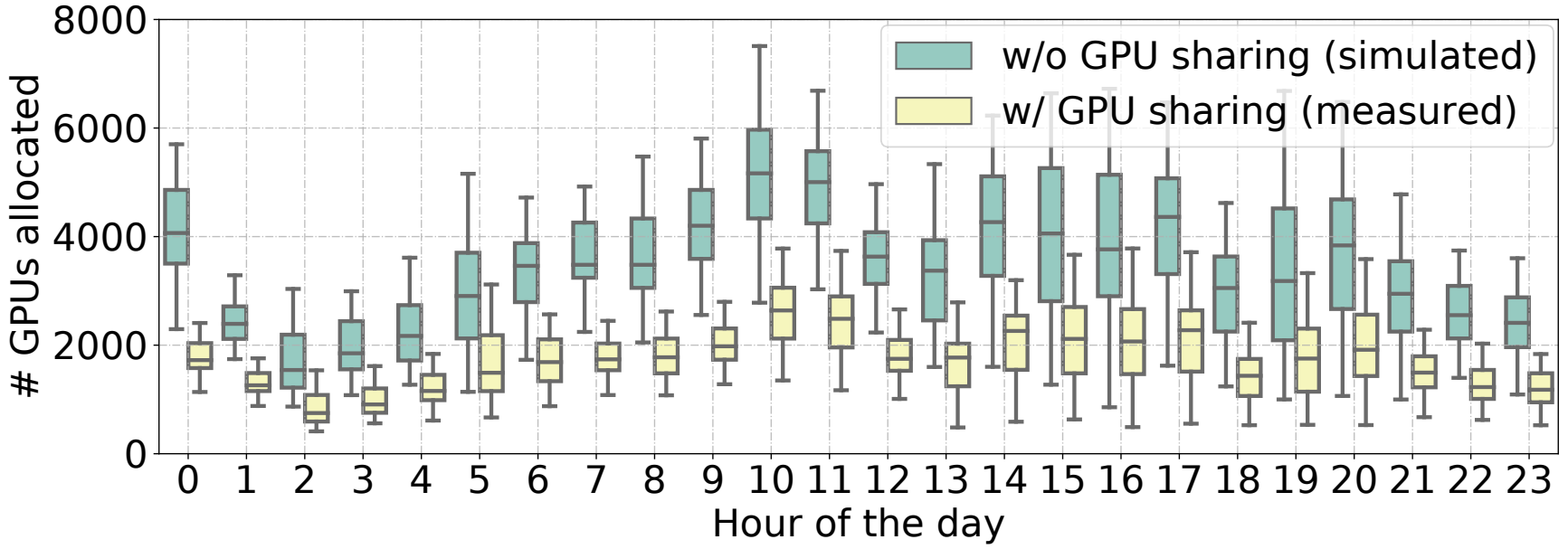
- Main memory and GPU memory are sufficient.



Opportunities

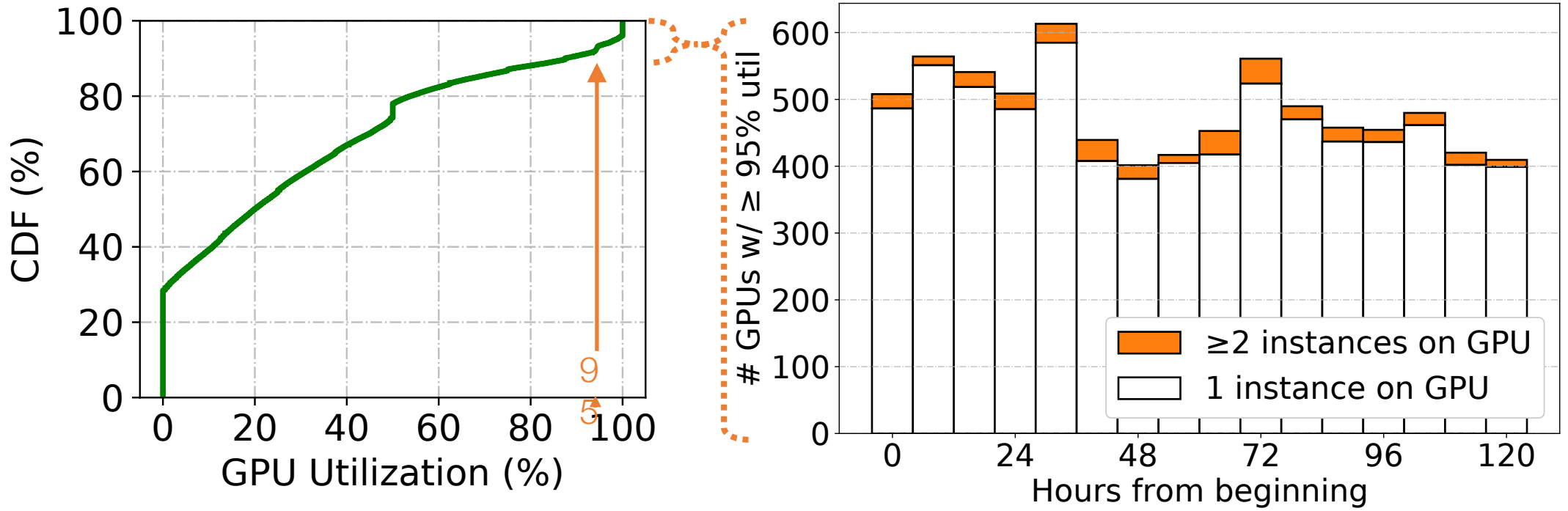
GPU Sharing

- Space-multiplex: GPU memory. Time-multiplex: GPU compute units.
- Sharing saves 50% GPUs on avg. and 2,500 GPUs in peak hrs.



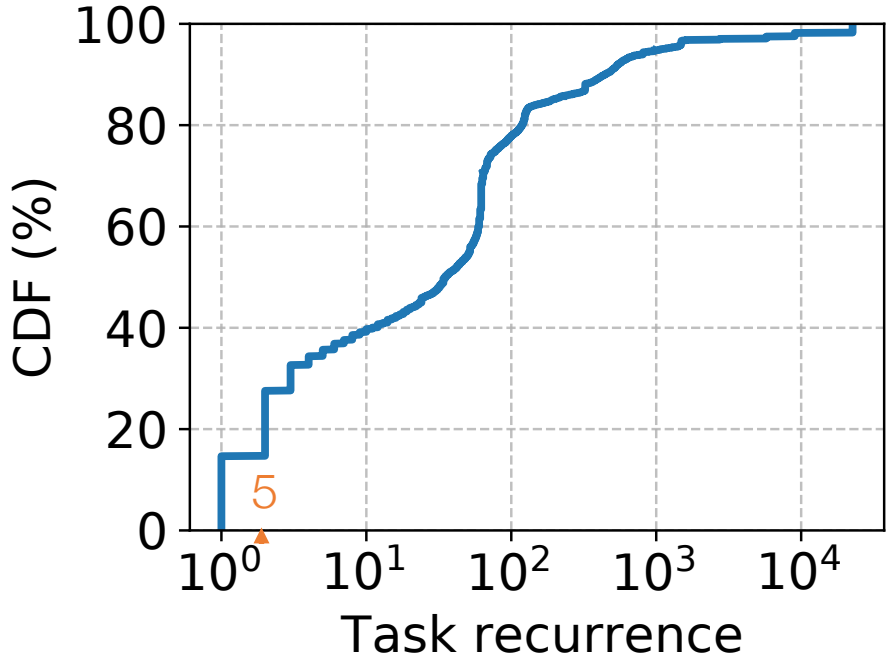
GPU Sharing

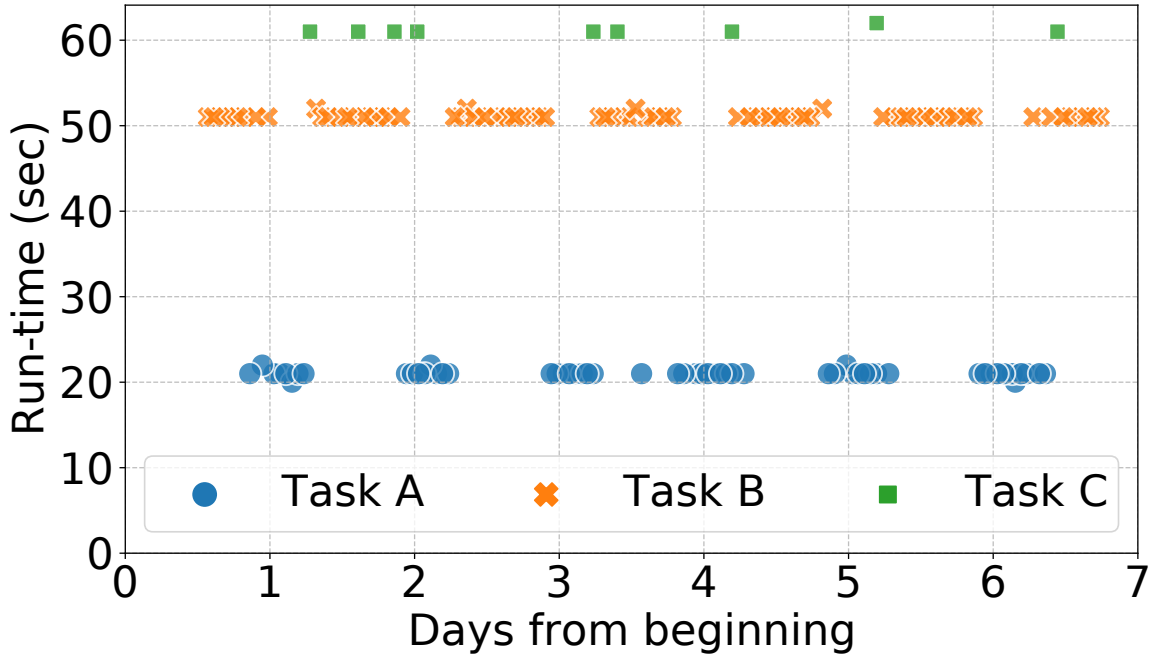
- Current GPU sharing does not cause severe GPU contention
  - On ≥ 95% utilized GPUs, only ~4% of them have ≥ 2 workers run simultaneously



Duration Predict for Recurring Tasks

- Recurring tasks are prevalent (65% tasks repeat ≥ 5 times)
  - Criteria of repetition: Group tag—hash value of jobs' customized input (e.g., entry scripts, command-line params, data sources and sinks)

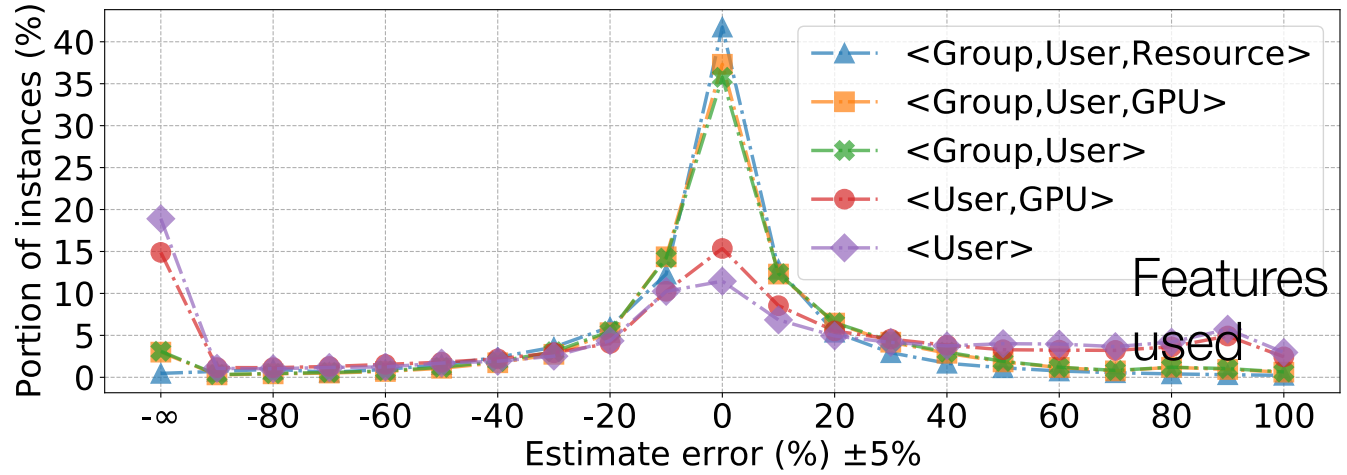


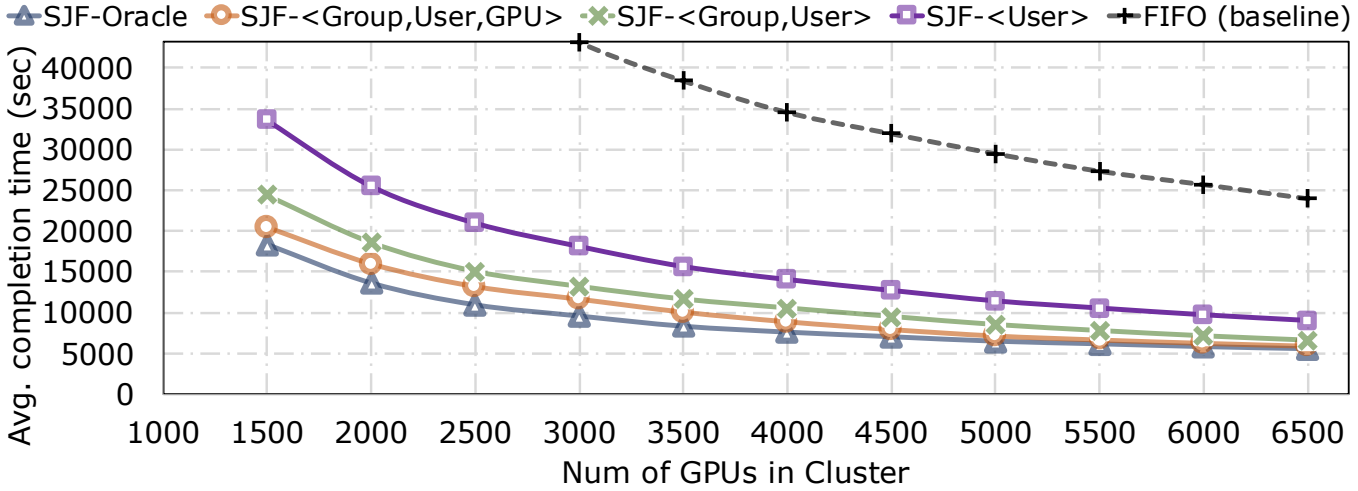


Sample inference tasks

Duration Predict for Recurring Tasks

- Predictor: Regression tree
  - Predict 78% instances duration within ±25% error
- Scheduling: SJF vs. FIFO
  - Speedup by 63%-77% with different predictors



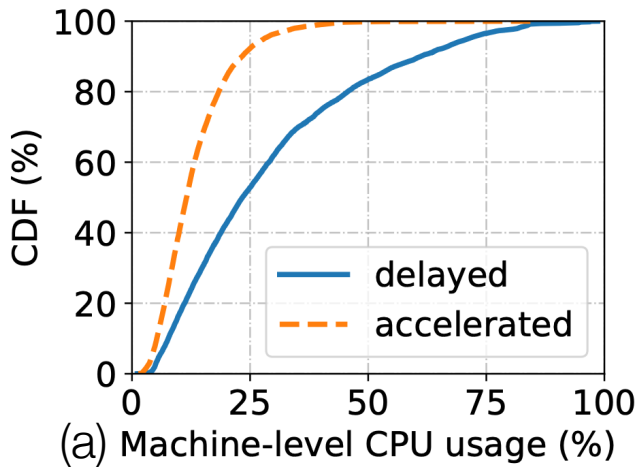


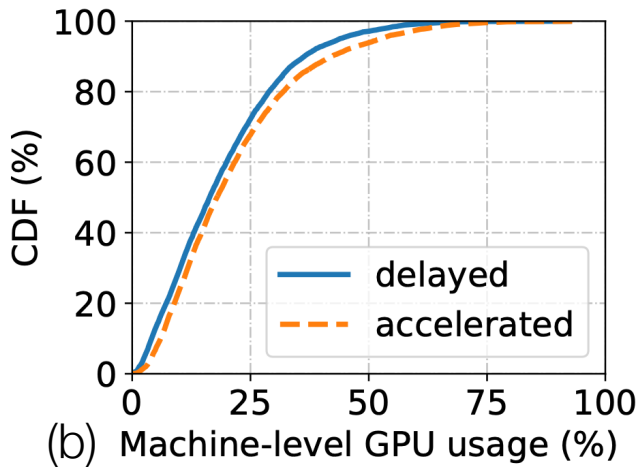
Challenges

CPU can be the bottleneck

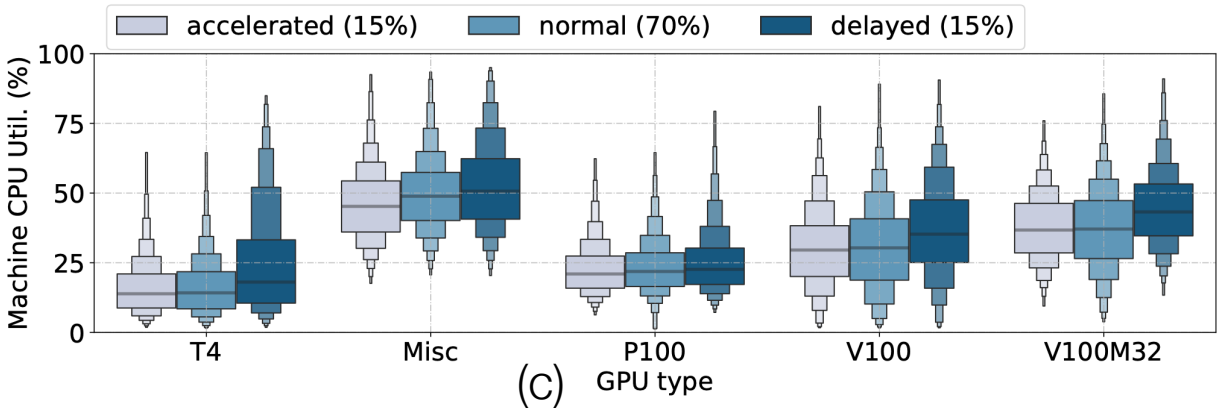
- Task delays are more likely to be observed under high CPU util. (a) (c), rather than high GPU util. (b) (d)

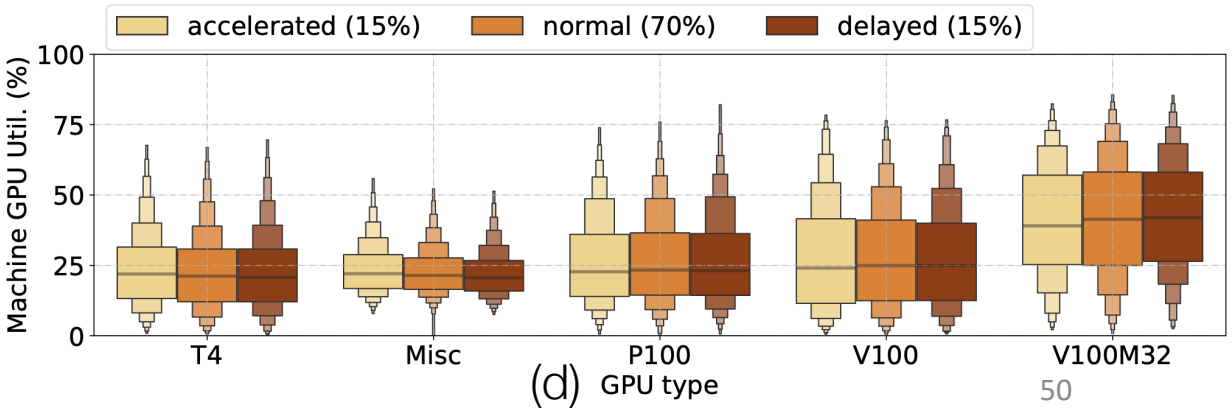
A sample Click-Through Rate (CTR) prediction task





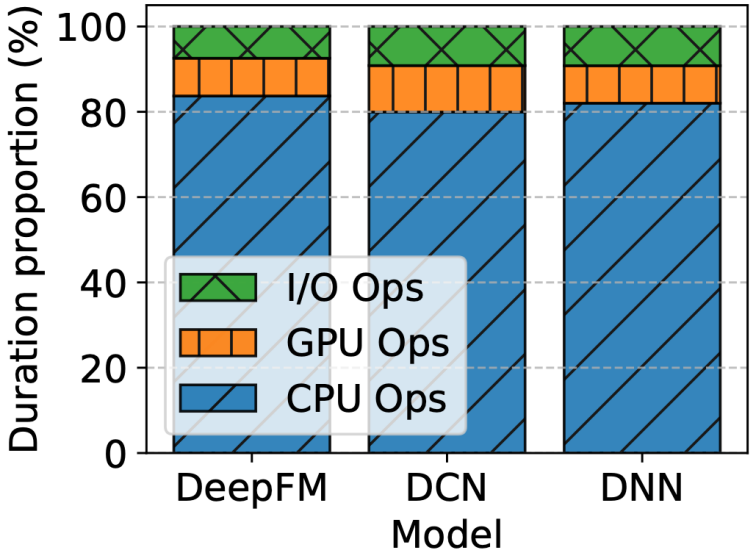
General tasks on various GPU nodes



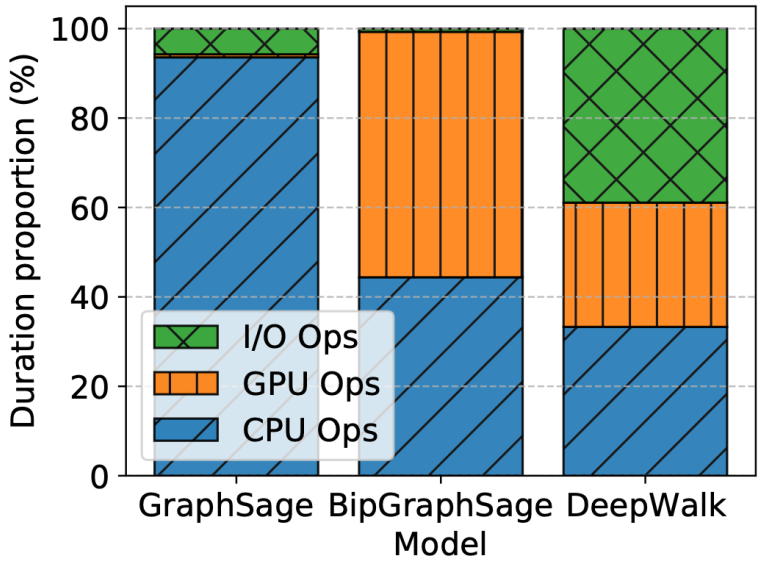


Majority of High-CPU-Low-GPU Tasks

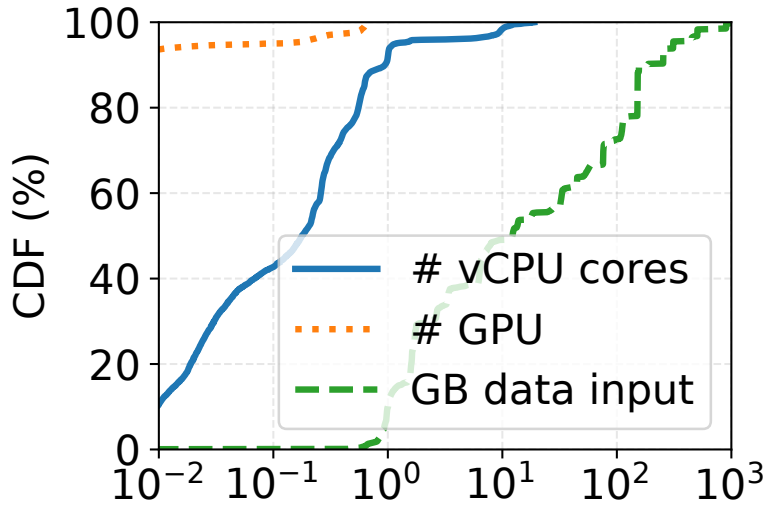
- CTR tasks could spend ~80% time on CPU (fetch and pre-process)
- GNNs spend 30-90% time on Edge Iteration, Neighbor Sampling, …
- RL launches massive CPU-intensive instances to run simulations



(a) Recommending: CTR models



(b) Graph Neural Networks (GNNs)

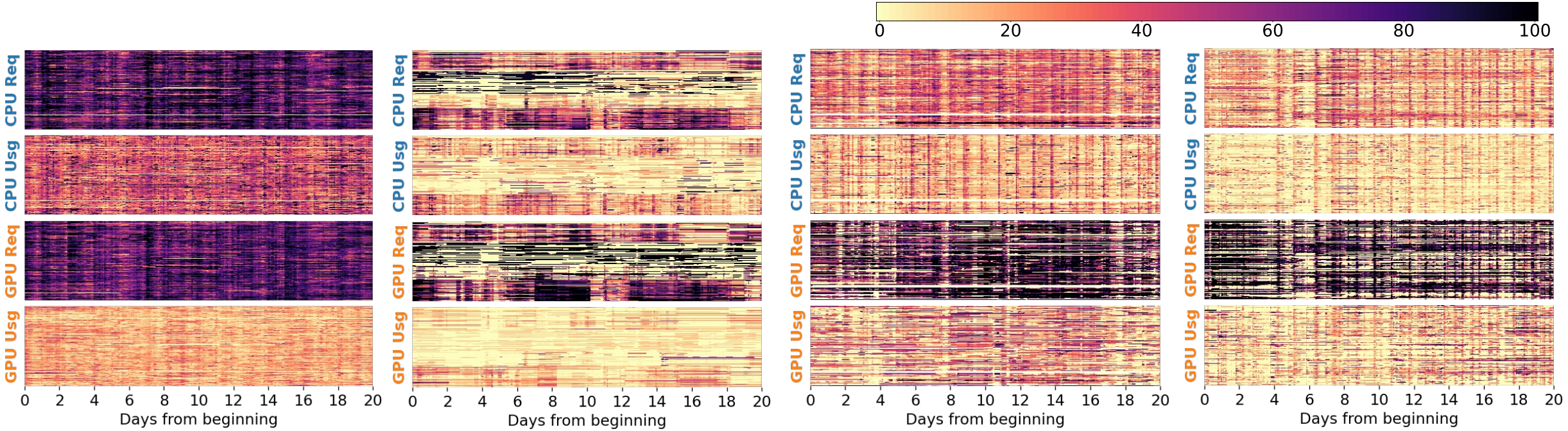


(c) Reinforcement Learning (RL)

Imbalanced Scheduling

- Overcrowding weak 8-GPU nodes (a)
- Packing on high-end V100 nodes (b)
- Underutilizing 2-GPU nodes (c) (d)

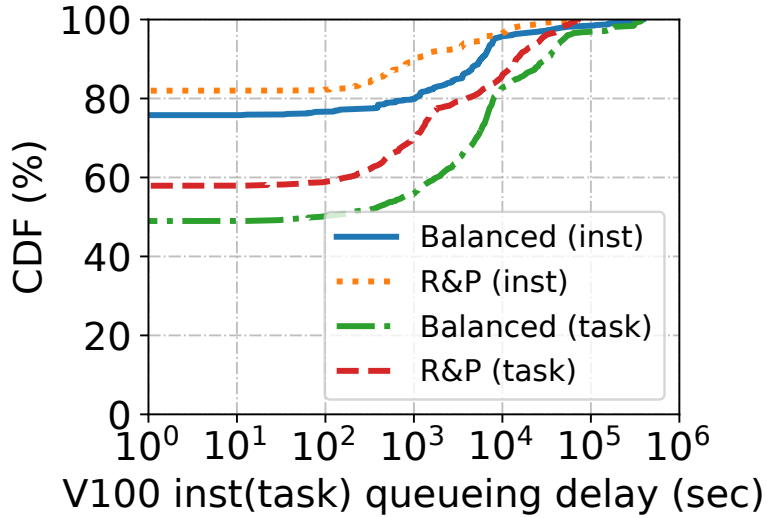
vCPU cores per GPU	All nodes	8-GPU nodes	2-GPU nodes
Machine specs	23.2	12.0	38.1
Instance requests	21.4	22.8	18.1



(a) 8-Misc-GPU Nodes (b) 8-V100-GPU Nodes (c) 2-P100-GPU Nodes (d) 2-T4-GPU Nodes

The Deployed Scheduling Policy

- Reserving-and-packing
  - Reserves high-end GPUs for high-GPU tasks
  - Packs other workloads to less-advanced GPUs
- Load-balancing
  - Assigns instances to nodes with low allocation rate



- Our scheduler prioritizes reserving-and-packing over load-balancing
  - Avoids extremely long latency for multi-GPU training jobs.
  - Simulation: V100 instances wait 68% shorter time in avg. with R&P than Balanced.

Takeaways

- A majority of tasks have gang-scheduled instances, mostly small in GPU
- GPU sharing and duration prediction of recurring tasks can help
- CPU could be the bottleneck, esp. under imbalanced scheduling

Open problems in GPU cloud

- Reduce GPU fragmentation
- GPU virtualization and GPU pooling
- Efficiency for prevalent domain applications (e.g., LLMs)

…

Credits

- Some slides are adapted from course slides of CSE291 in UCSD
- Some slides are adapted from course slides of CS 250B in UCI
- Q. Weng, W. Xiao, Y. Yu, et al., MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters. In USENIX NSDI, 2022.