Multi-Scale GPU Resource Management for Deep Learning

Mosharaf Chowdhury

Deep Learning is Ubiquitous Today

Image processing
Natural language processing
Speech synthesis
Intelligent assistants
Autonomous vehicles
Search
Video analytics

"Ok Google"

Deep Learning Lifecycle from 10K Feet

Hyperparameter Tuning → Training → Inference

Minimize makespan of exploring many configurations

Minimize completion time

Maximize throughput and meet deadline

Deep Learning is Repetitive

Each iteration is predictable

- Duration
- Memory usage profile
- Communication characteristics

Number of iterations is unpredictable

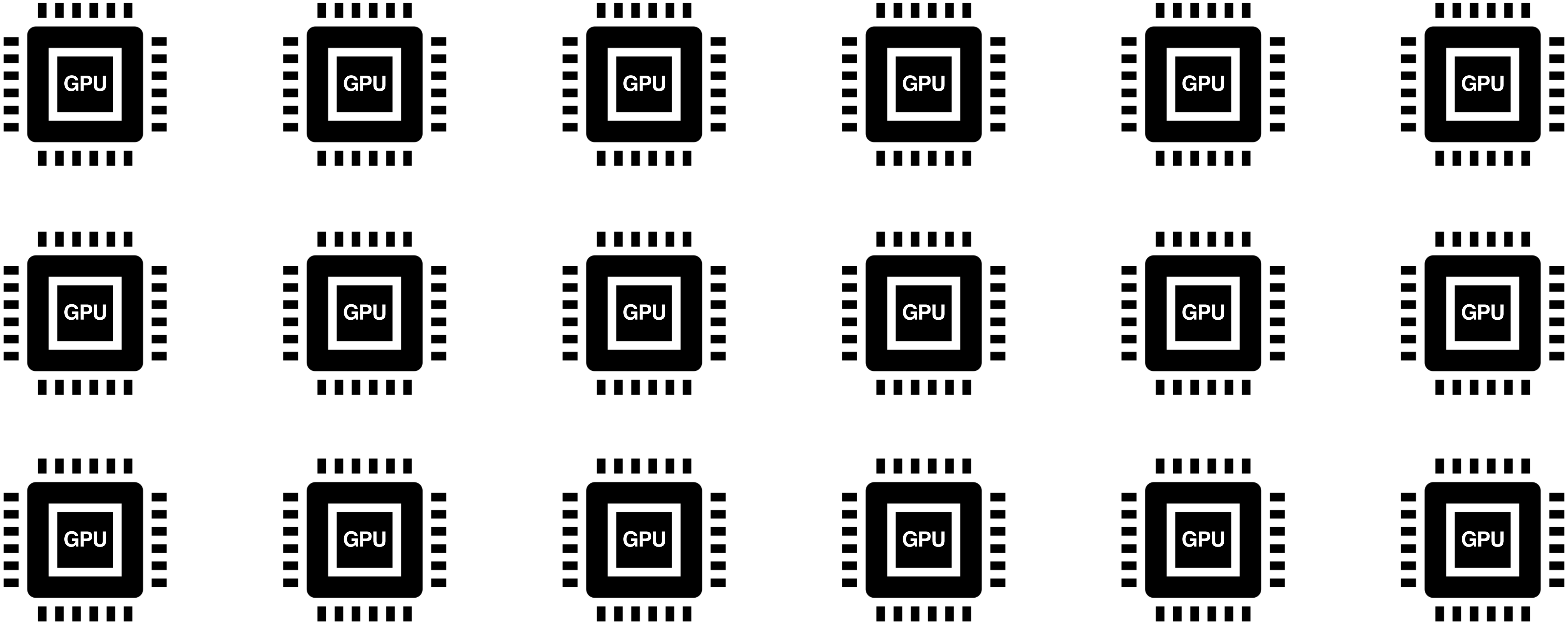
Deep Learning is Computationally Heavy

	Deep Neural Networks
Inherently Parallel	✓
Matrix Operations	✓

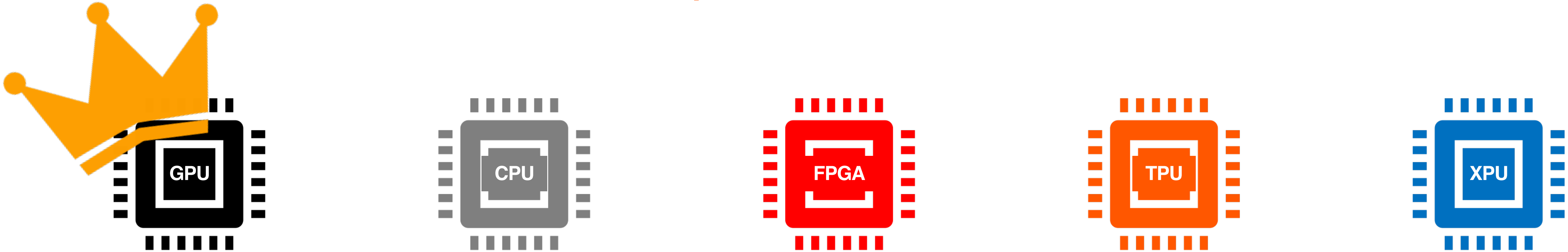
GPU is King!

	Deep Neural Networks	GPU
Inherently Parallel	✓	✓
Matrix Operations	✓	✓

GPU Clusters



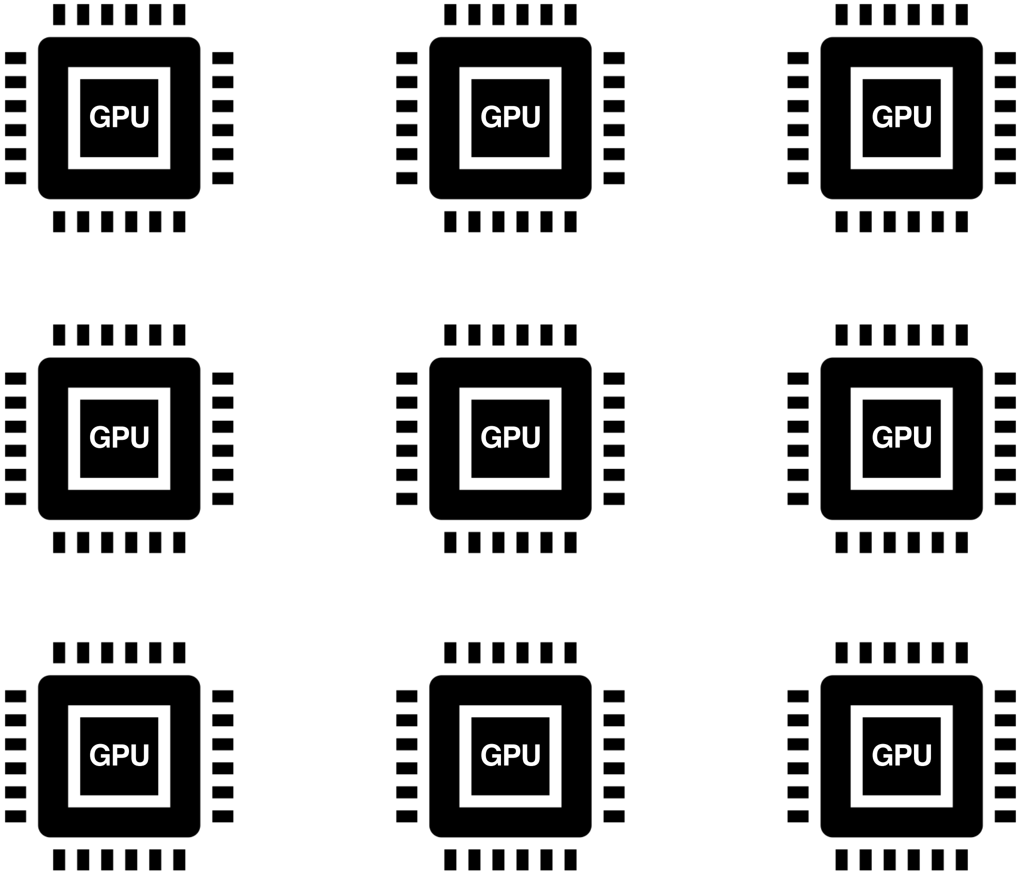
And Other Compute Devices…



Techniques described in this talk are generalizable

- We assume the compute device(s) to be black box

1. Macro-Scale Challenges



Performance

- Finish jobs quickly

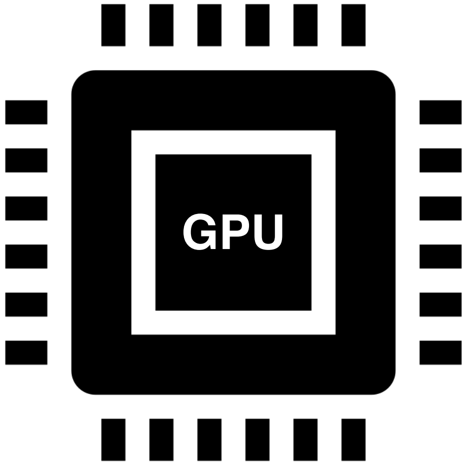
Efficiency

- Use all devices

Fairness

- Share all resources equitably

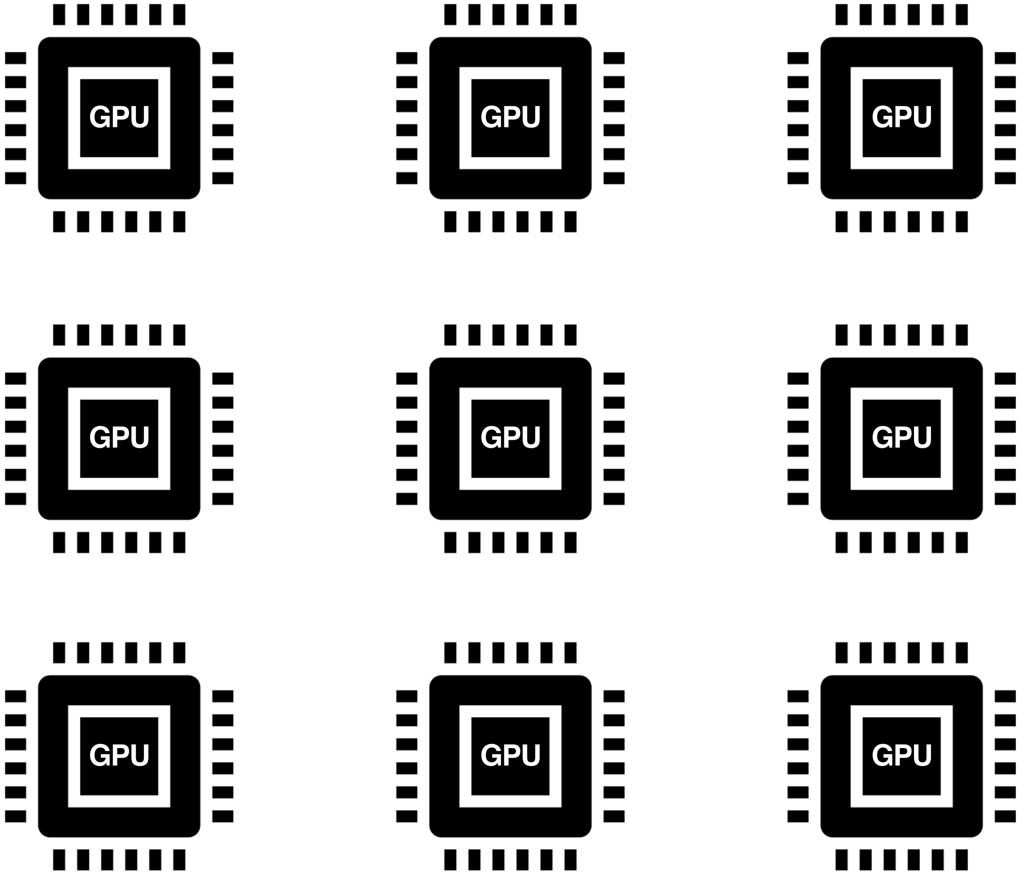
2. Micro-Scale Challenges



High Utilization

- Use all a device's resources

1.1 Macro-Scale Challenges



Performance

- Finish jobs quickly

Efficiency

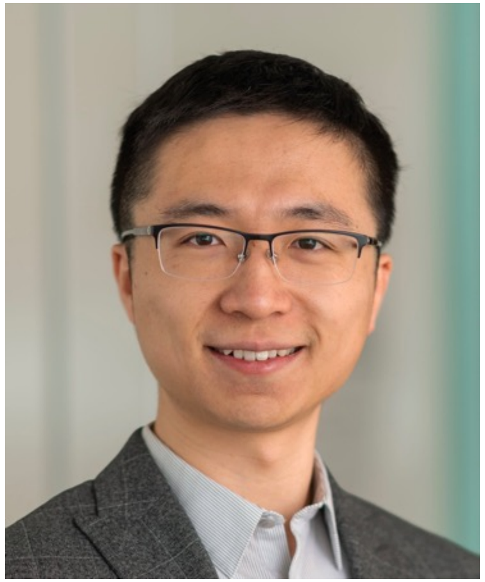
- Use all devices

Fairness

- Share all resources equitably

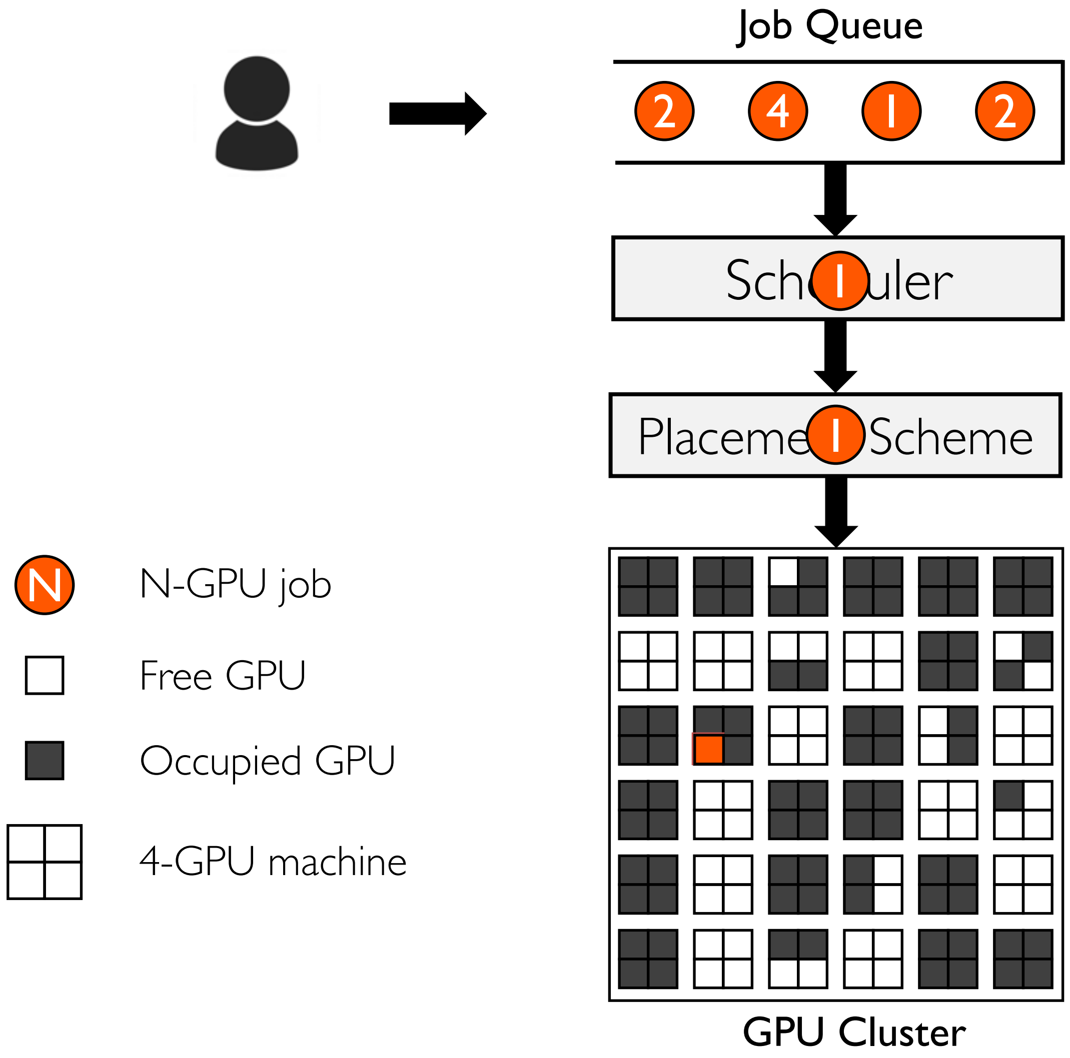
Tiresias

A GPU Cluster Manager for Distributed Deep Learning



w/ Juncheng Gu and many others

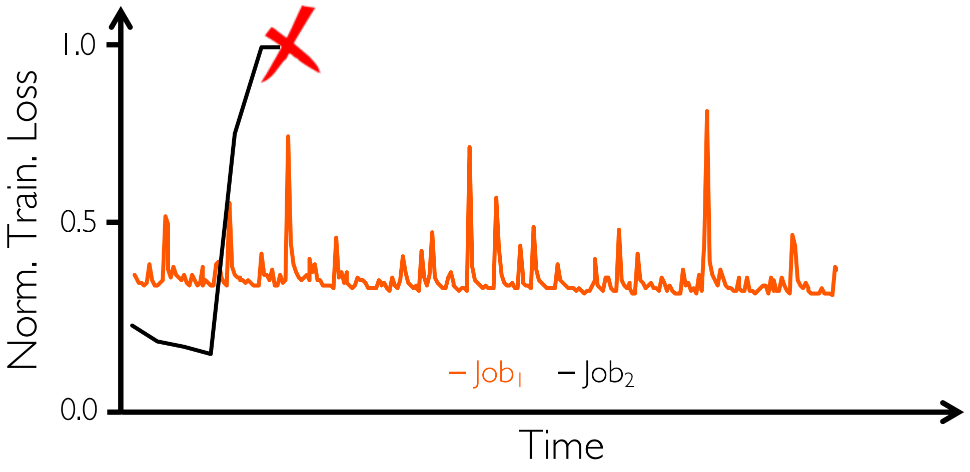
Lifecycle of a Job



Minimize the Average Job Completion Time

Given

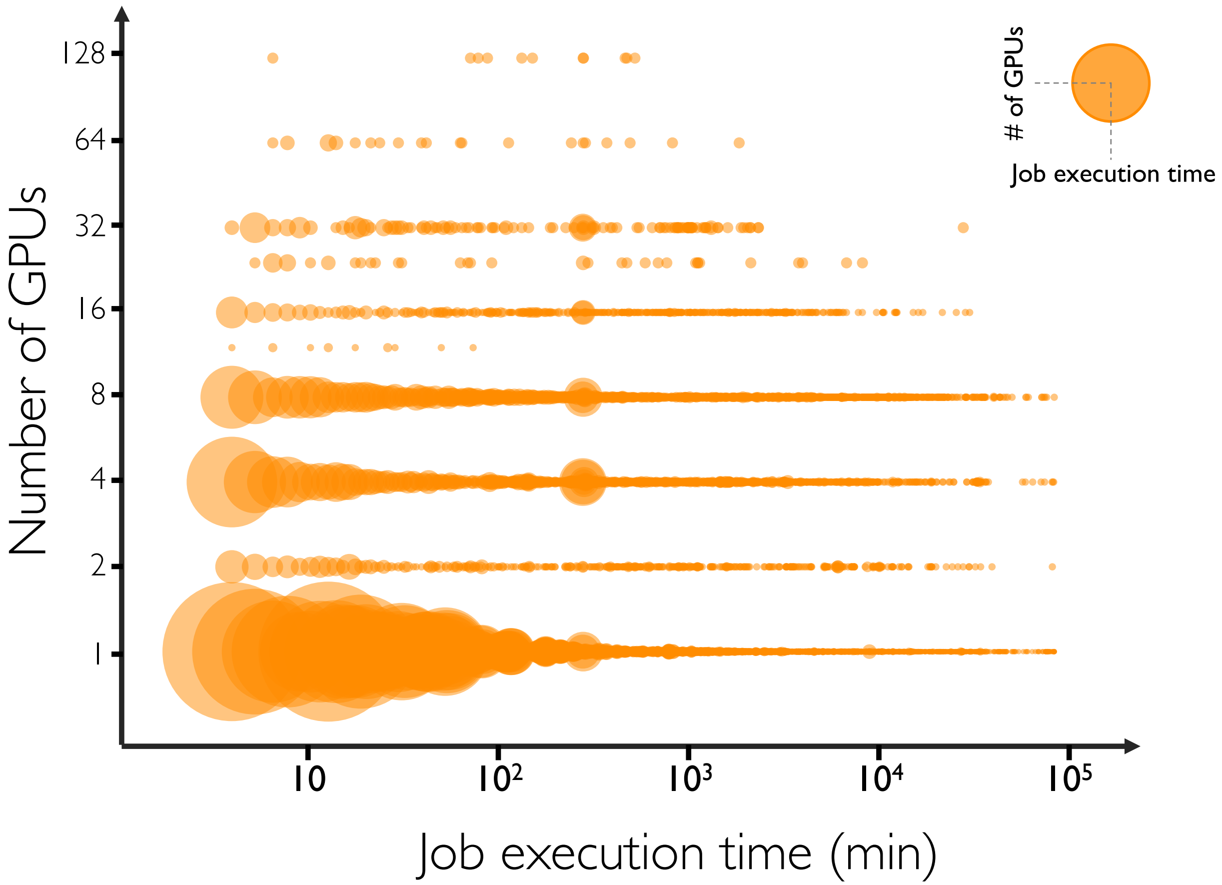
- Online job arrival
- Heterogeneous resource demands
- Unpredictable job duration



Minimize the Average Job Completion Time

Given

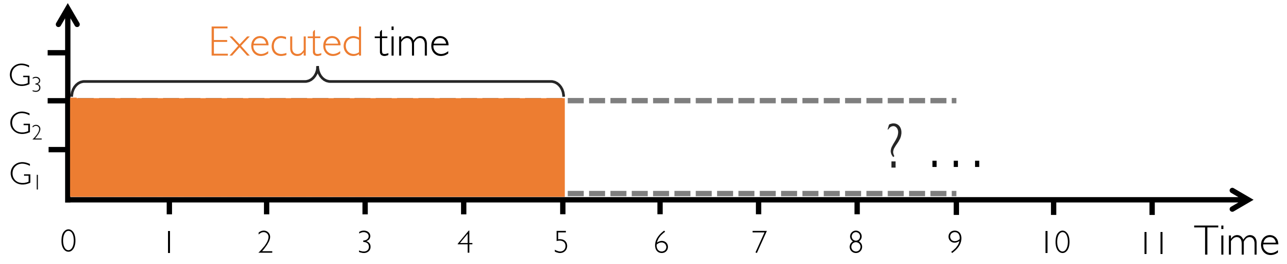
- Online job arrival
- Heterogeneous resource demands
- Unpredictable job duration
- Wide Spatiotemporal Variations



Trace from Microsoft's Philly cluster

Available Job Information

1. Spatial: number of GPUs
2. Temporal: *executed* time



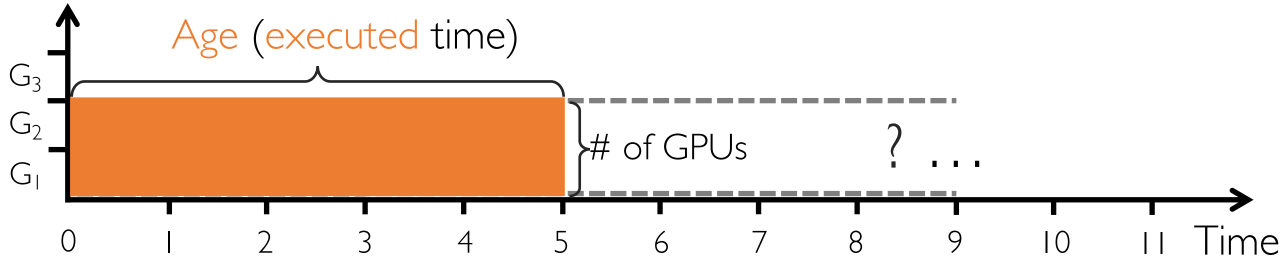
Age-Based Schedulers

Least-Attained Service[1] (LAS)

- Prioritize job that has the shortest executed time

Gittins Index policy[2]

- Need the ***distribution of job execution time***
- Prioritize job that has the highest probability to complete in the near future



1. Feedback queueing models for time-shared systems. JACM, 1968
2. Multi-armed bandit allocation indices. Wiley, 1989

Two-Dimensional Age-Based Scheduler (2DAS)

Age calculated by two-dimensional attained service

- A job's total executed GPU time (# of GPUs × executed time)

No prior information

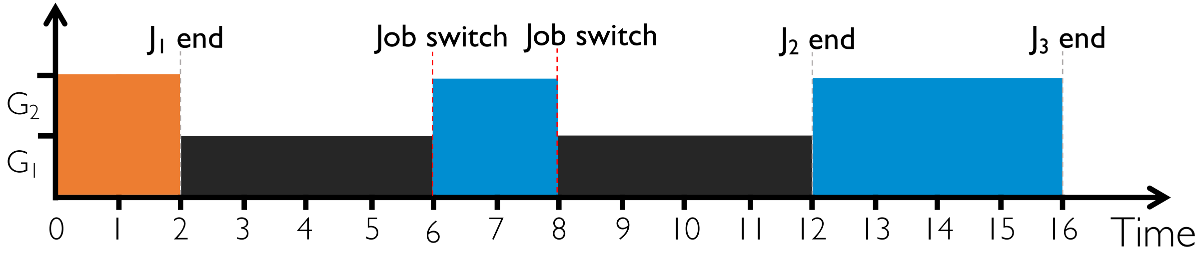
- 2D-LAS

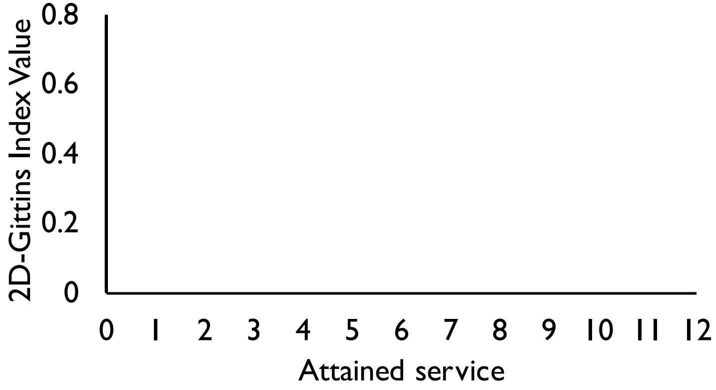
With partial information: distribution of job GPU time

- 2D-Gittins Index

2D Gittins Index: Partial Information

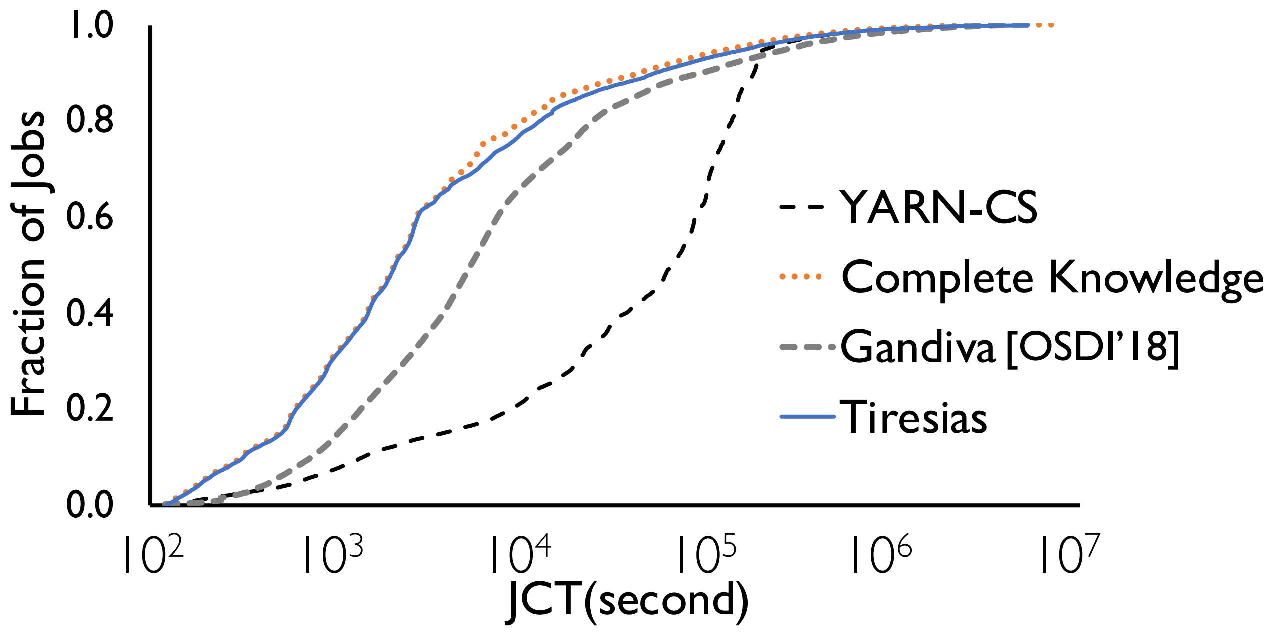
	# of GPUs	Distribution
J_1	2	2
J_2	1	(4, 8, 12)
J_3	2	6





Higher probability to complete (Gittins Index), higher priority

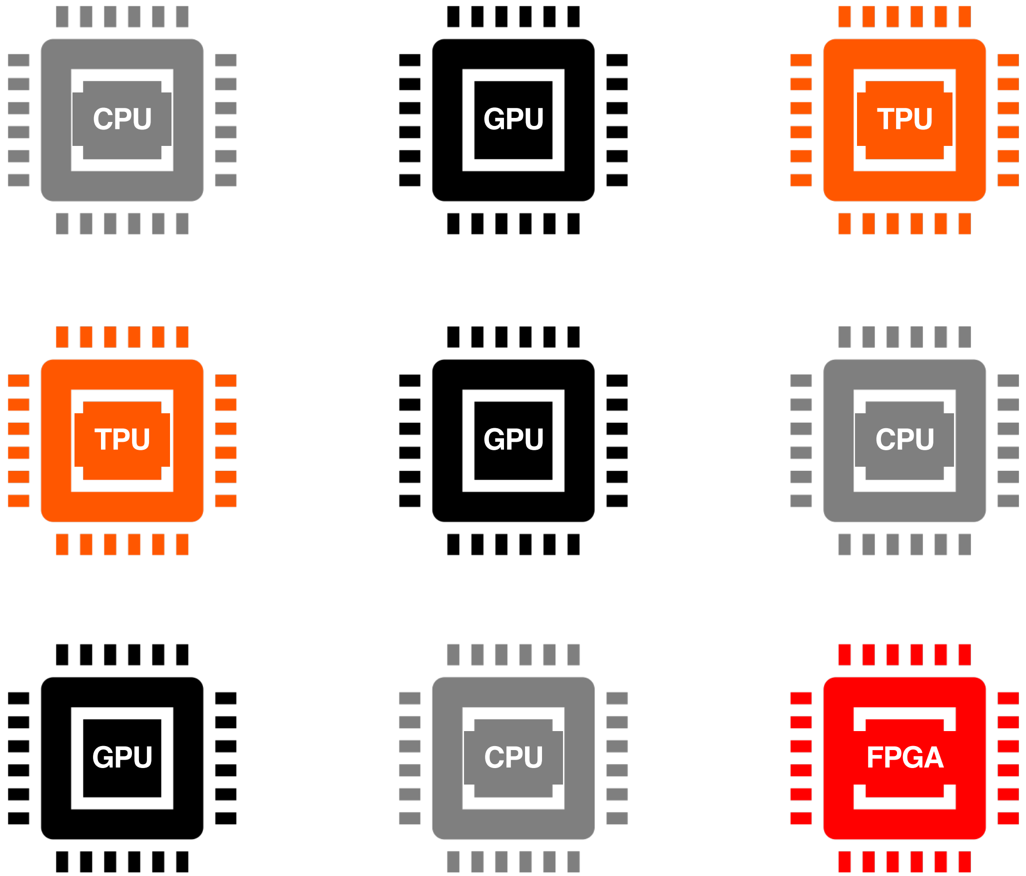
5.5X Improvement in Average JCT



Trace from a 2000-GPU cluster

1.2X Improvement in Makespan

1.2 Macro-Scale Challenges



Performance

- Finish jobs quickly

Efficiency

- Use all devices

Fairness

- Share all resources equitably

AlloX

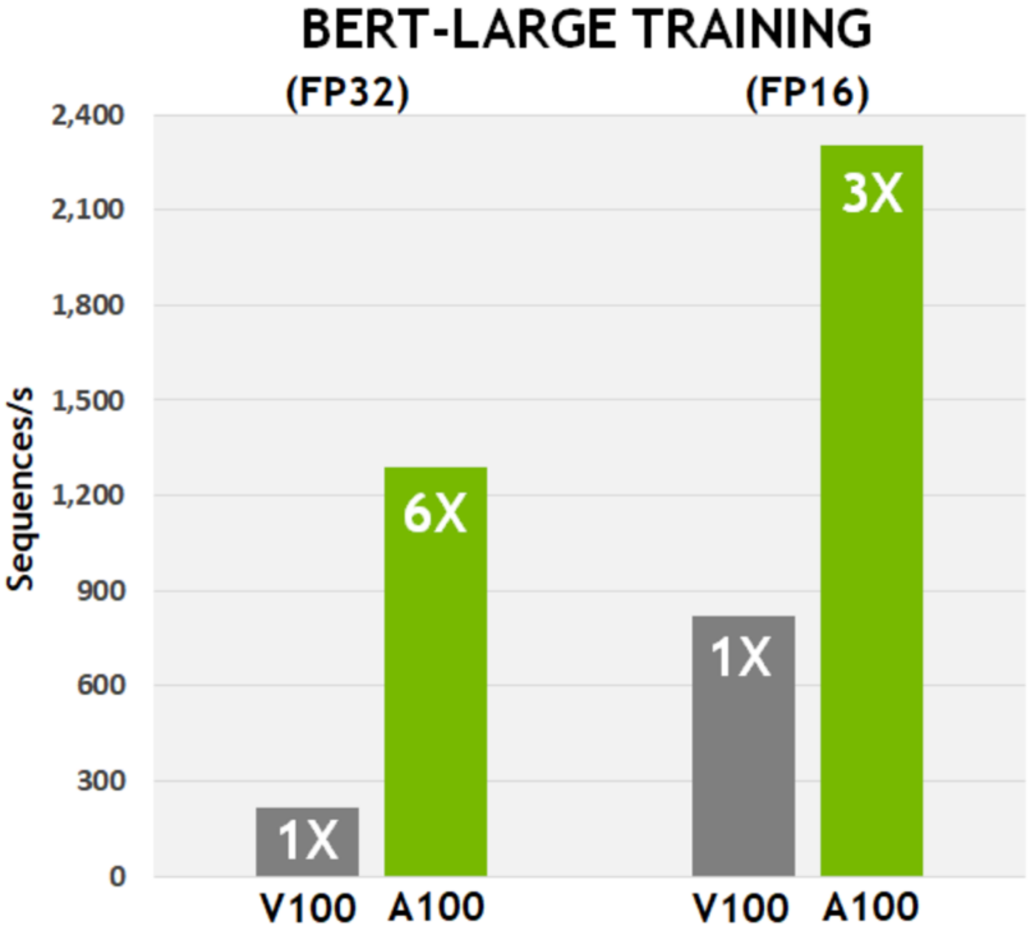
Compute Allocation in Hybrid Clusters

w/ Tan Le, Xiao Sun, and Zhenhua Liu

Interchangeable Resources

	CPU	GPU	TPU	FPGA
PyTorch				
PaddlePaddle				

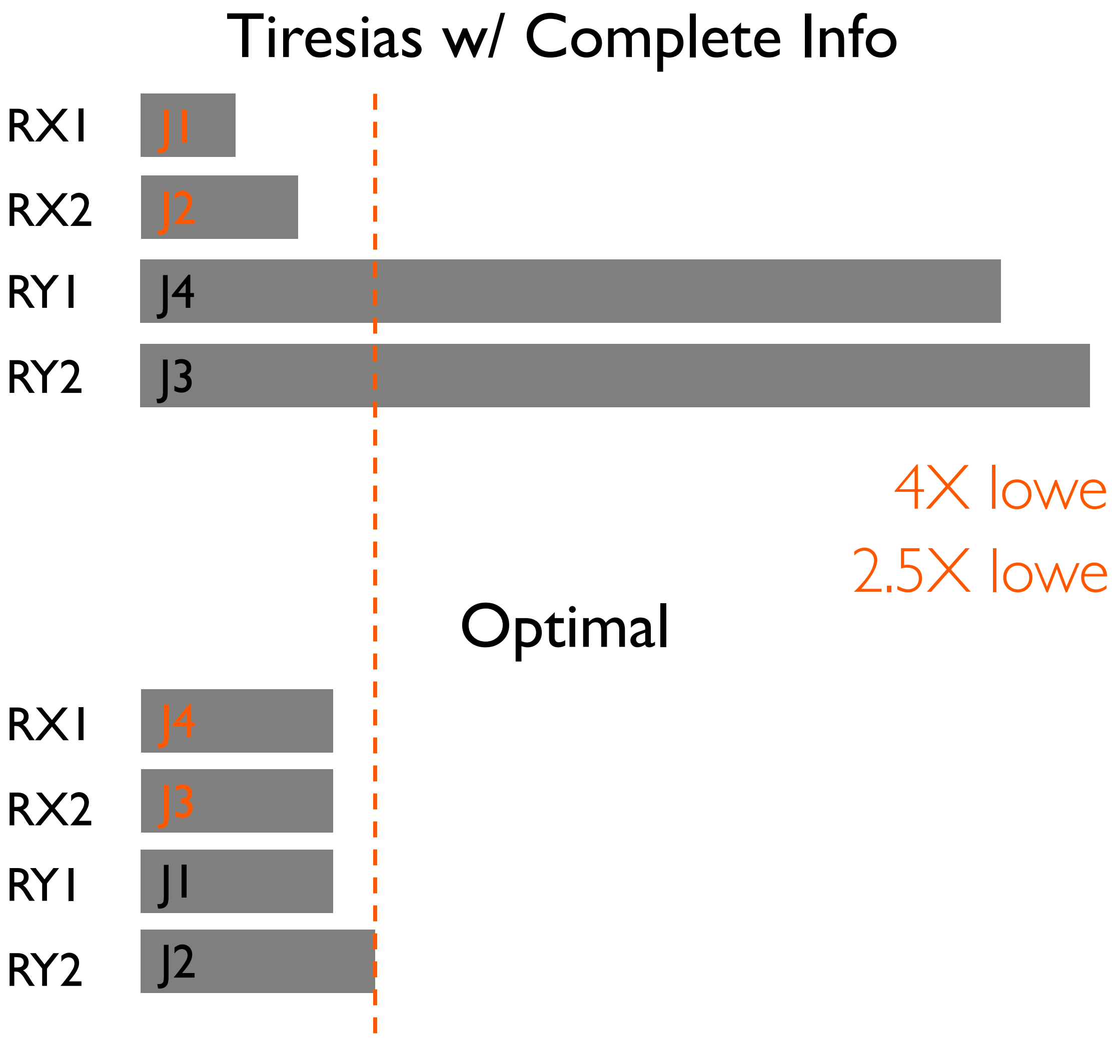
Distinct Speedup Rates



Source: NVIDIA A100 News Release

How to Assign Jobs to Different Compute?

	Resource-X compl. time	Resource-Y compl. time
J1	10	20
J2	15	25
J3	20	100
J4	20	90



4X lower makespan

2.5X lower avg. JCT

Minimize the Average JCT

Given

- Offline job arrival
- Heterogeneous resource demands
- Interchangeable resources

Profile and Match

- Determine speedup ratios
- Solve a min-cost bipartite matching problem

Repeatedly apply the offline solution for online scenario

Maintain Fairness

Users may not be happy if they keep getting slower compute

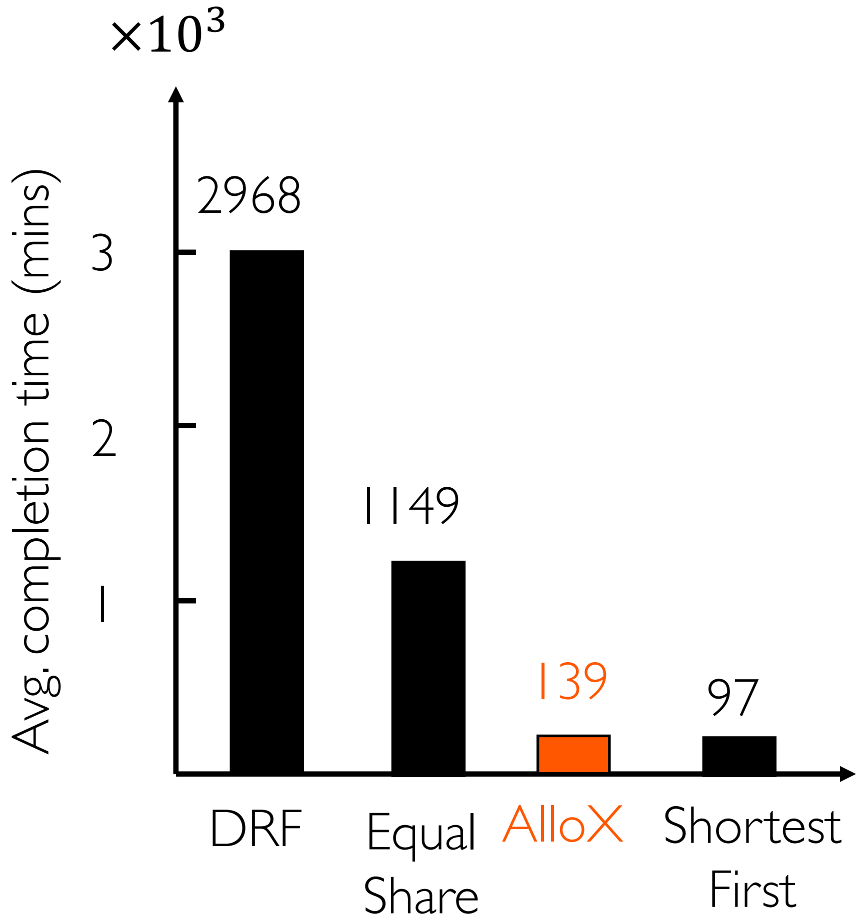
Equalize progress of jobs

- Independent of resource

We also prove the following negative result

- No multi-configuration allocation can satisfy (1) pareto efficiency (PE) and sharing incentive (SI), and (2) strategyproofness (SP) simultaneously unless the relative speedup of the two resources is the same for all jobs.

20X Average JCT Improvement *and* Fair



TensorFlow CNN benchmarks

Takeaways

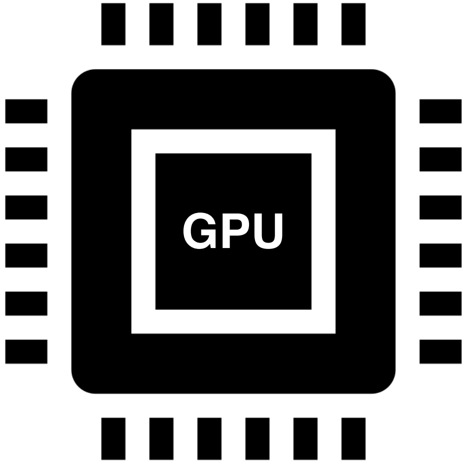
Job duration is often unpredictable

- Scheduling based on the past can work well

Compute resources may have different speed

- Rethink existing schedulers for heterogeneity

2. Micro-Scale Challenges



High Utilization

- Use all a GPU's resources

Salus

Fine-grained GPU Sharing Primitives For Deep Learning

w/ Peifeng Yu

Exclusive GPU Access

A GPU entirely belongs to one job

- Simple to reason about and deal with

Limits flexibility

- Expensive preemption

Leads to underutilization

- High variance model size

Model	Peak Memory Usage
VAE	28M
Super Resolution	529M
Deep Speech	3993M
Inception4	11355M

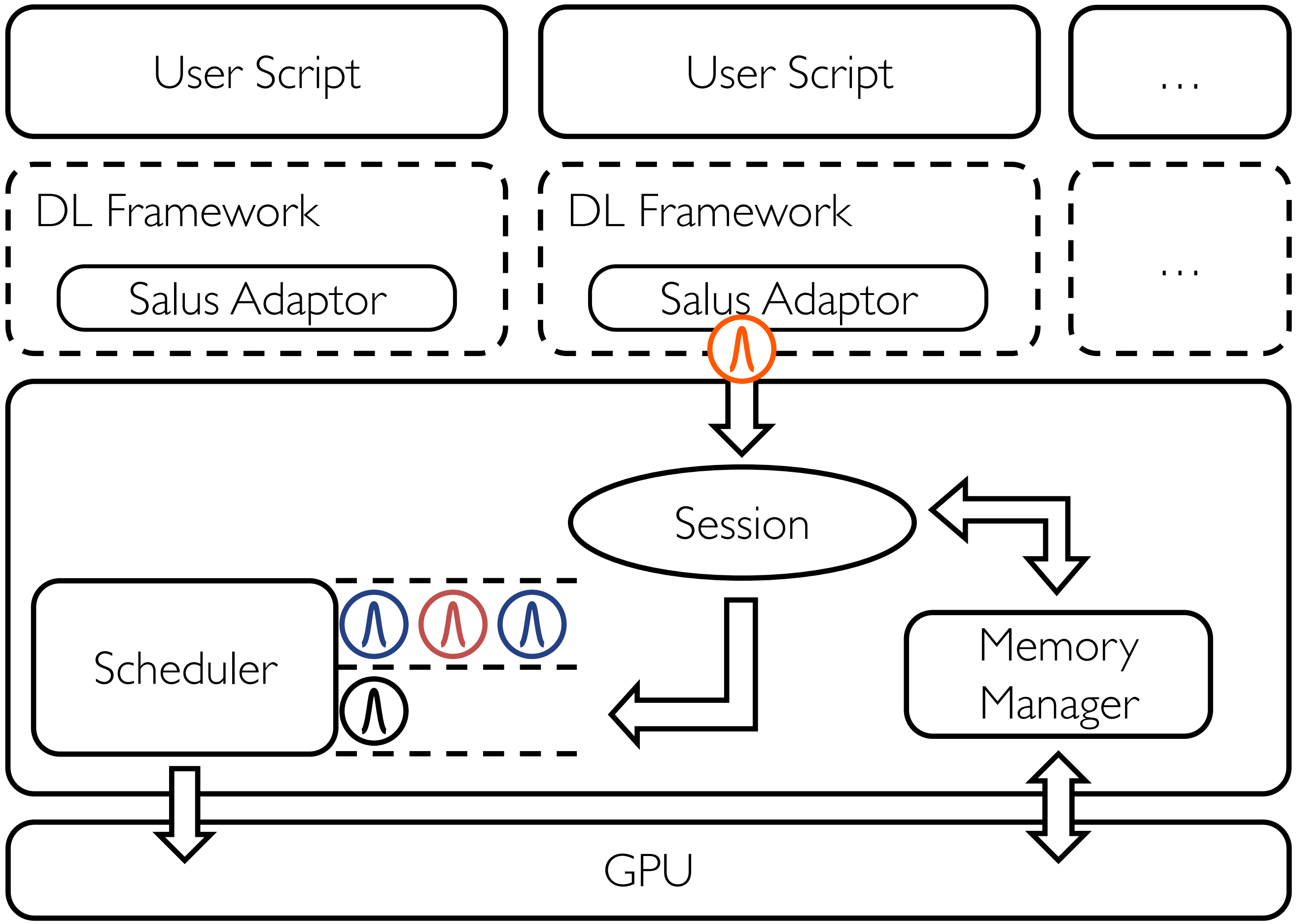
GPU Sharing

Approach	Efficiency	Dynamic Memory	Flexible Scheduling
Static Partitioning (SP)	No	No	Yes
Multi-Process Service (MPS)	Yes	No	No
Salus	Yes	Yes	Yes

Lifecycle of an Iteration

Create session
Send computation graph
For each iteration

- Send input
- Check memory
- Queue in scheduler

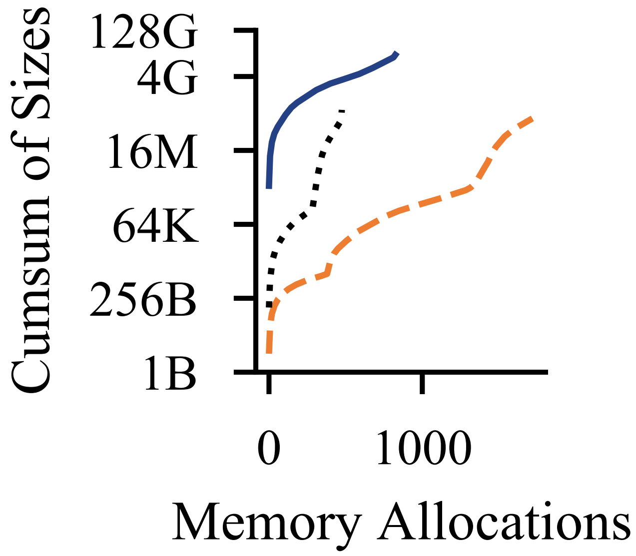


GPU Memory Usage in Deep Learning

Three types of memory

- Model
- Ephemeral
- Framework-internal

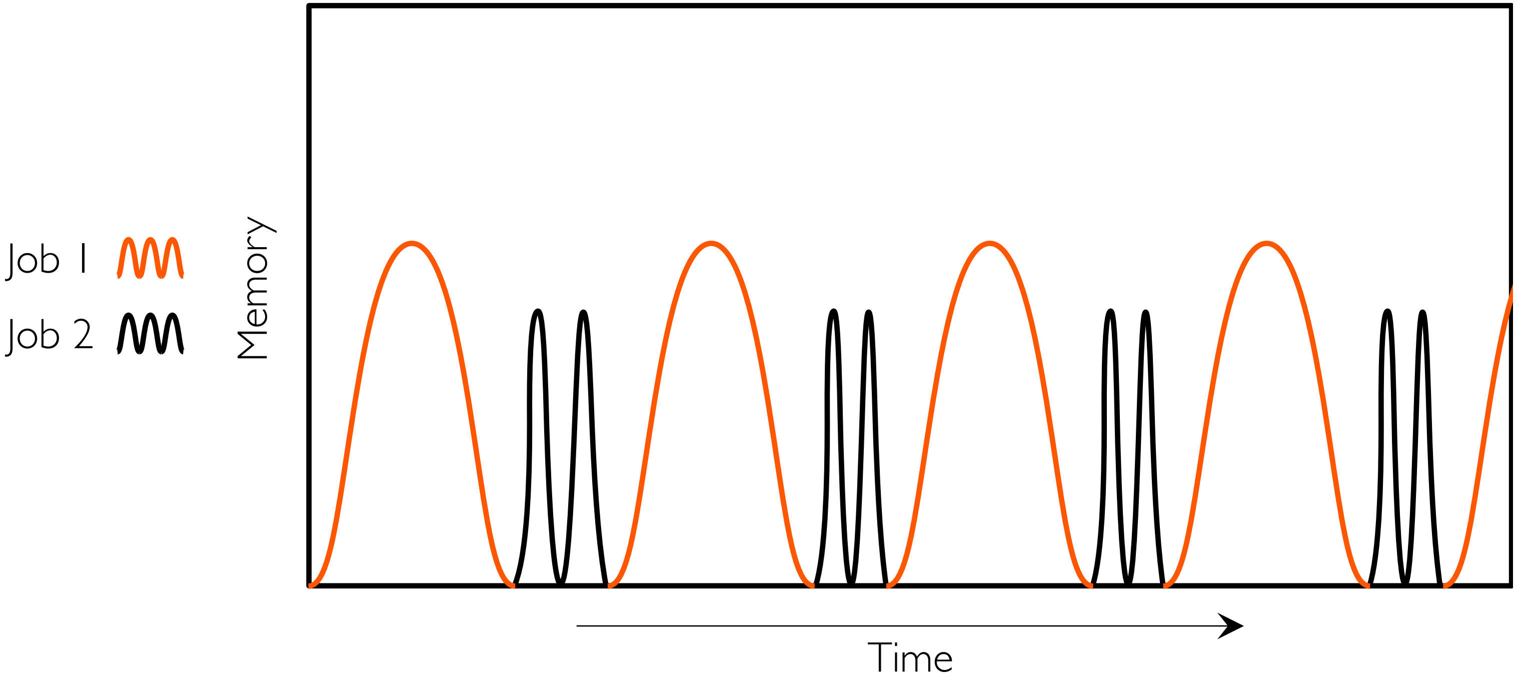
Model memory << GPU memory capacity



Fast Job Switching

Job switching is done by determine which job's iteration to run next

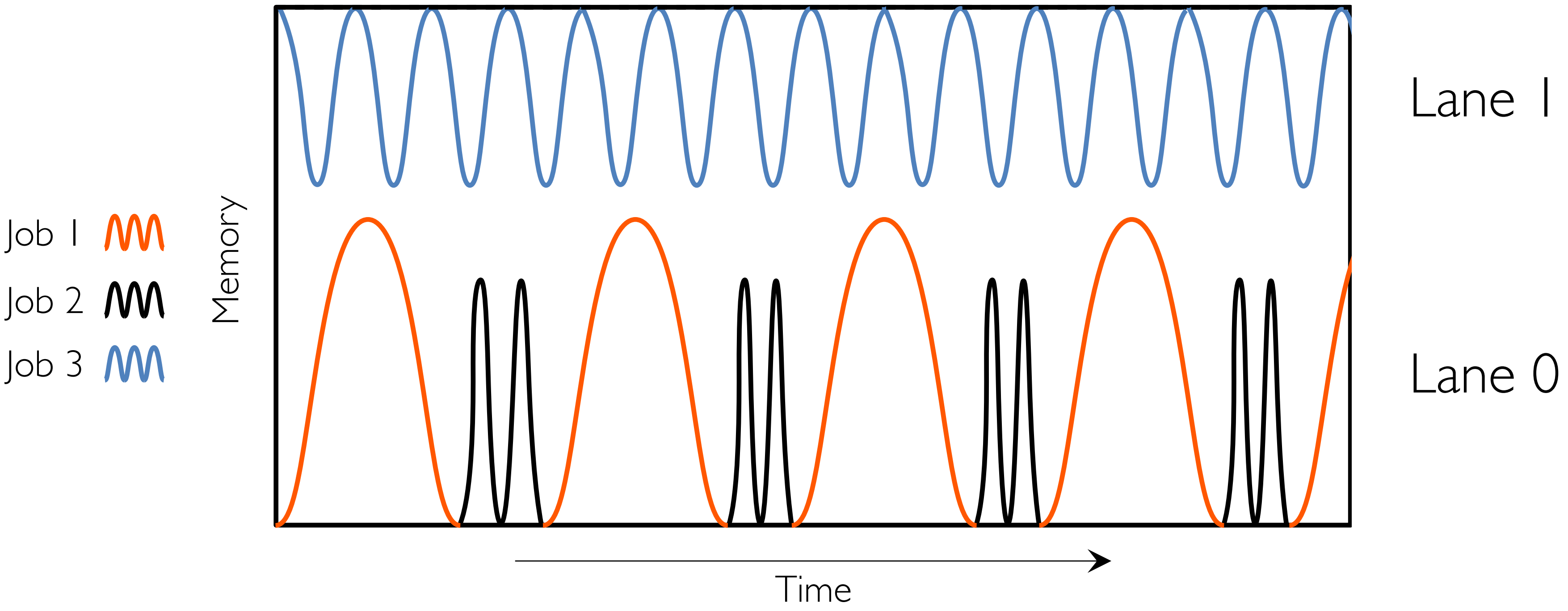
- Trade-off between maximum utilization and execution performance



GPU Lane

Contiguous physical memory + GPU stream

- Time-slicing within lane, parallel across lanes
- Dynamic re-partitioning (lane assignment)
- Avoid in-lane fragmentation



GPU Lane: Safety Conditions

A lane cannot accept arbitrary number of jobs

- The safety condition determines whether a job can go in a lane without deadlock

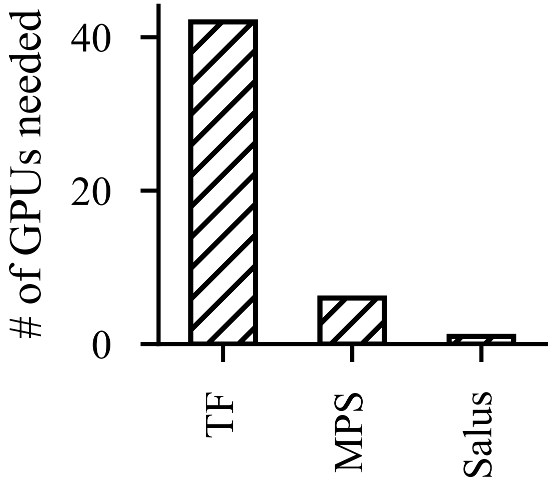
instead of

$$\sum_i P_i + \max_i T_i \le C_l \qquad \sum_i T_i$$

P_i: Model and framework-internal memory for job i
T_i: Ephemeral memory for job i
C_l: Memory capacity of lane l

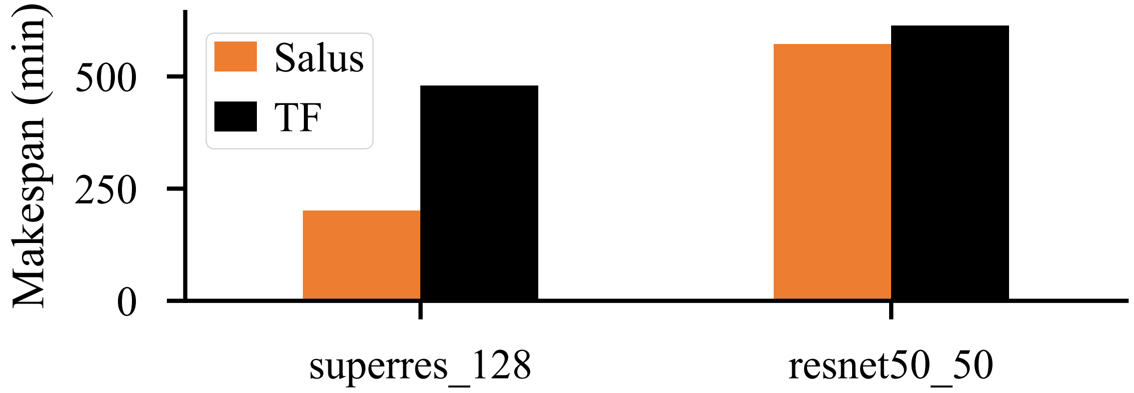
High Utilization

Packing 42 Inference Models



14 CNN and RNN models; 3 copies of each

Hyperparameter Tuning



300 configurations in each job

Low Overhead in Most Cases

*Outliers have significant CPU computation

Takeaways

Device memory is often a limiting factor

- Distinguishing between ephemeral memory and the rest is critical for squeezing more out of it

Blackbox hardware put rigid constraints

- Software can provide flexibility and generality with little overhead

Summary

https://github.com/symbioticlab

Too much is unknown

- Blackbox hardware and unpredictable jobs

Resources are too expensive to waste

- We need resource management both at the cluster level[1,2] and in individual devices[3]
- Both in homogeneous[1,3] and heterogeneous[2] settings with interchangeable resources
- To achieve performance,[1,2] efficiency,[1,3] and fairness[2]

Short-term certainty can be enough for long-term gains

Deep Learning Workload

Multi-Scale Resource Management

Hardware for Deep Learning

1. Tiresias: A GPU Cluster Manager for Distributed Deep Learning, NSDI'19
2. AlloX: Compute Allocation in Hybrid Clusters, EuroSys'20
3. Salus: Fine-Grained GPU Sharing Primitives for Deep Learning Applications, MLSys'20