Lecture 6: DL Cluster Schedulers

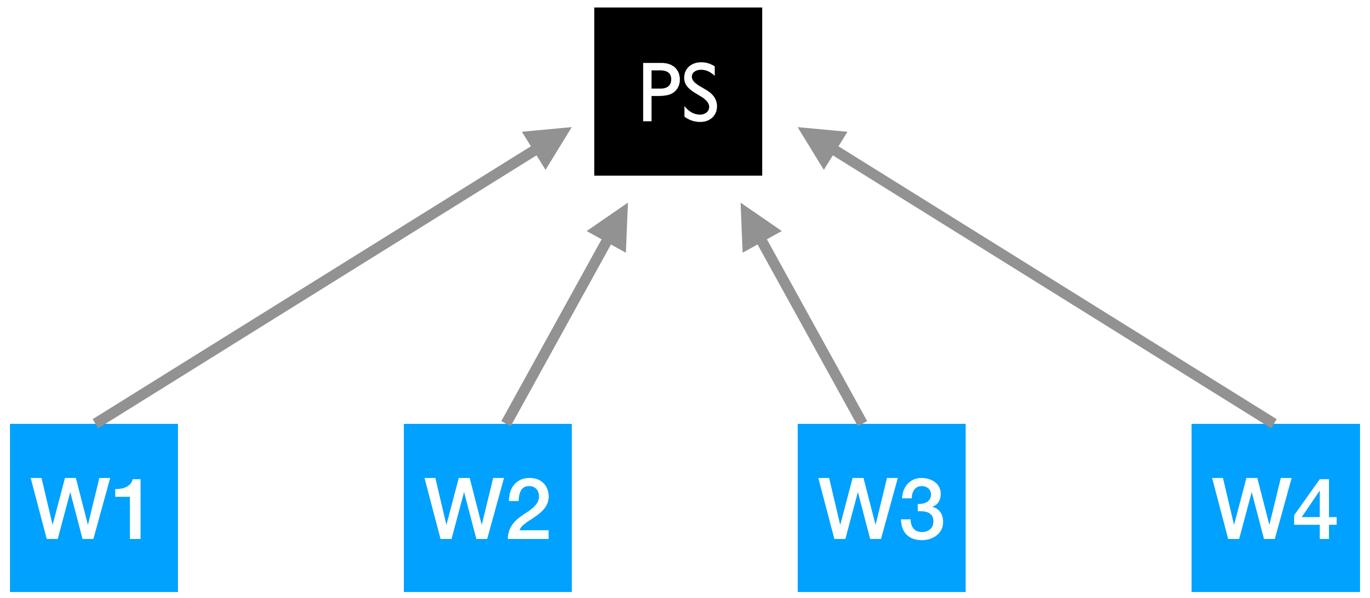
CS 256: Systems and Machine Learning

Sangeetha Abdu Jyothi

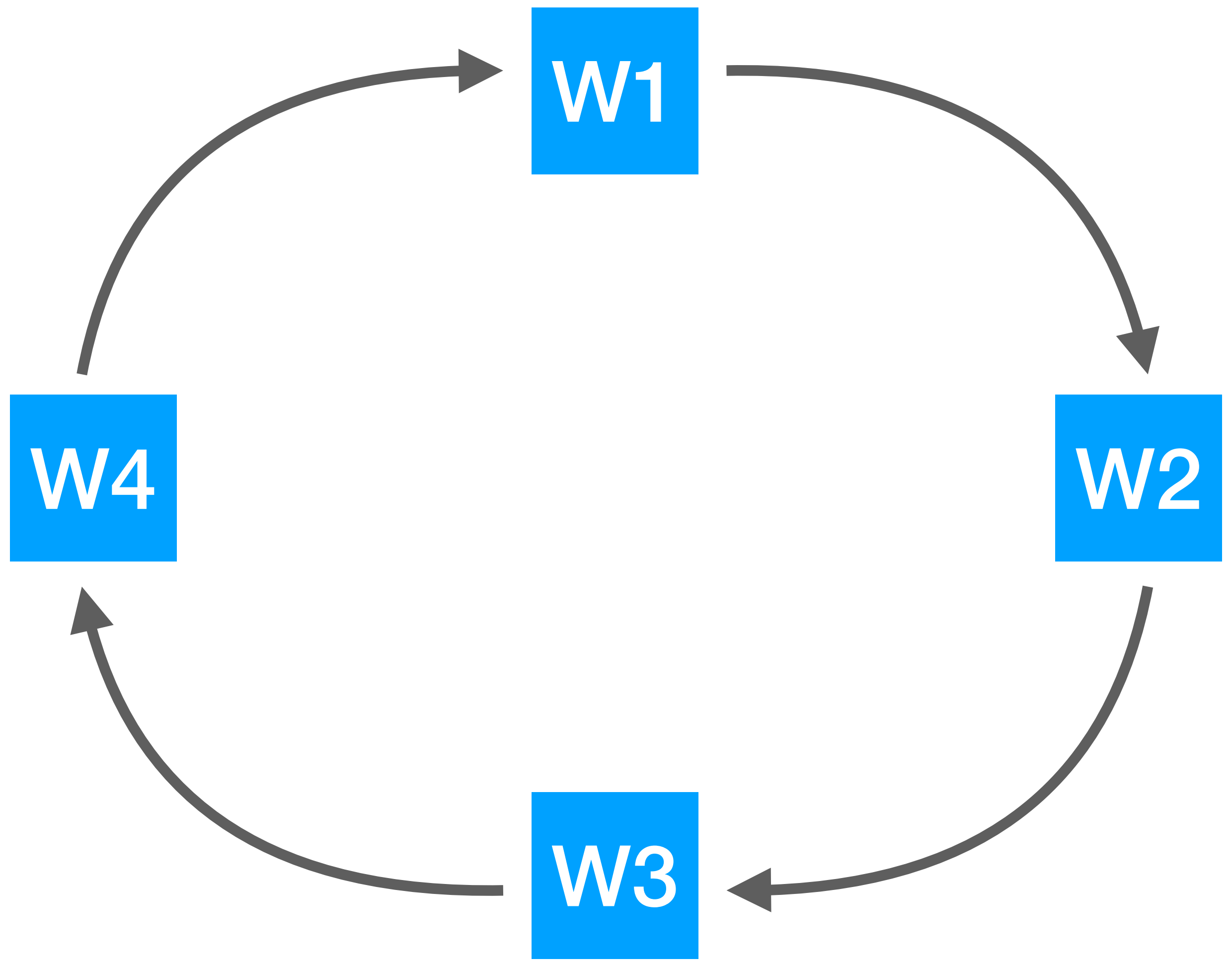
UCIrvine

Parts of this lecture were adapted from talks on Gandiva, Tiresias, Gavel

Last Lecture: Network aggregation

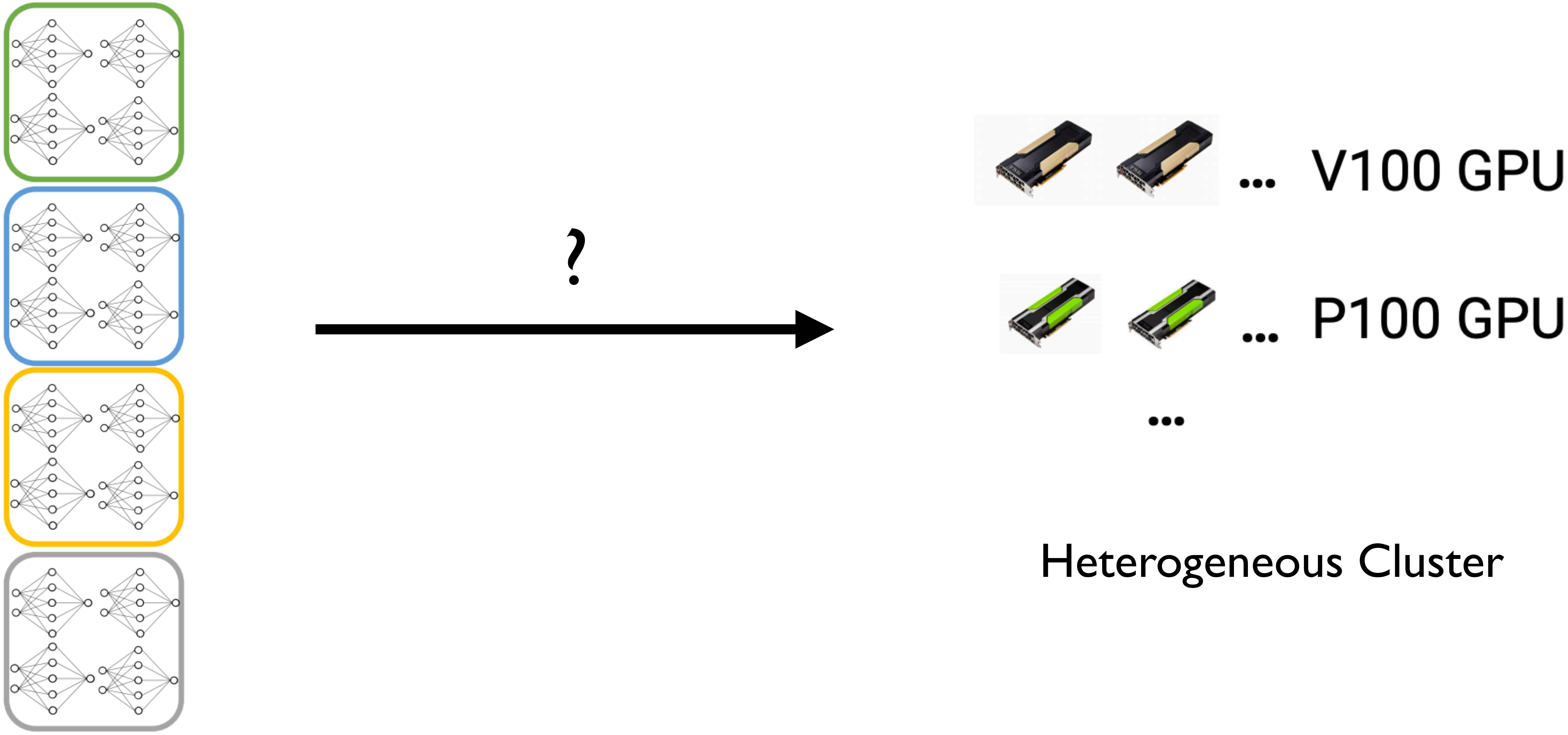


Parameter Server



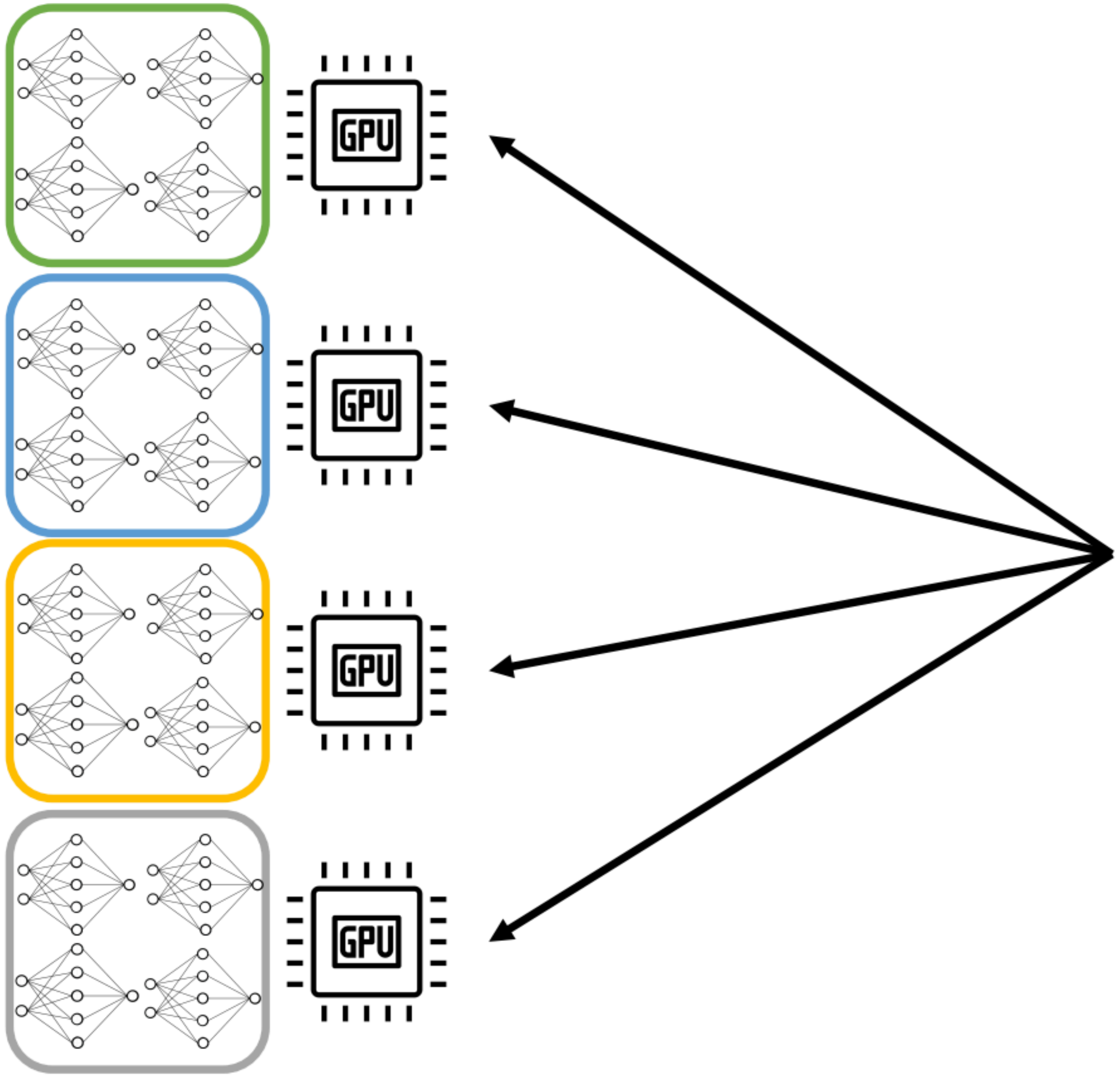
Decentralized Aggregation

Deep Learning Clusters



Diverse set of DL jobs

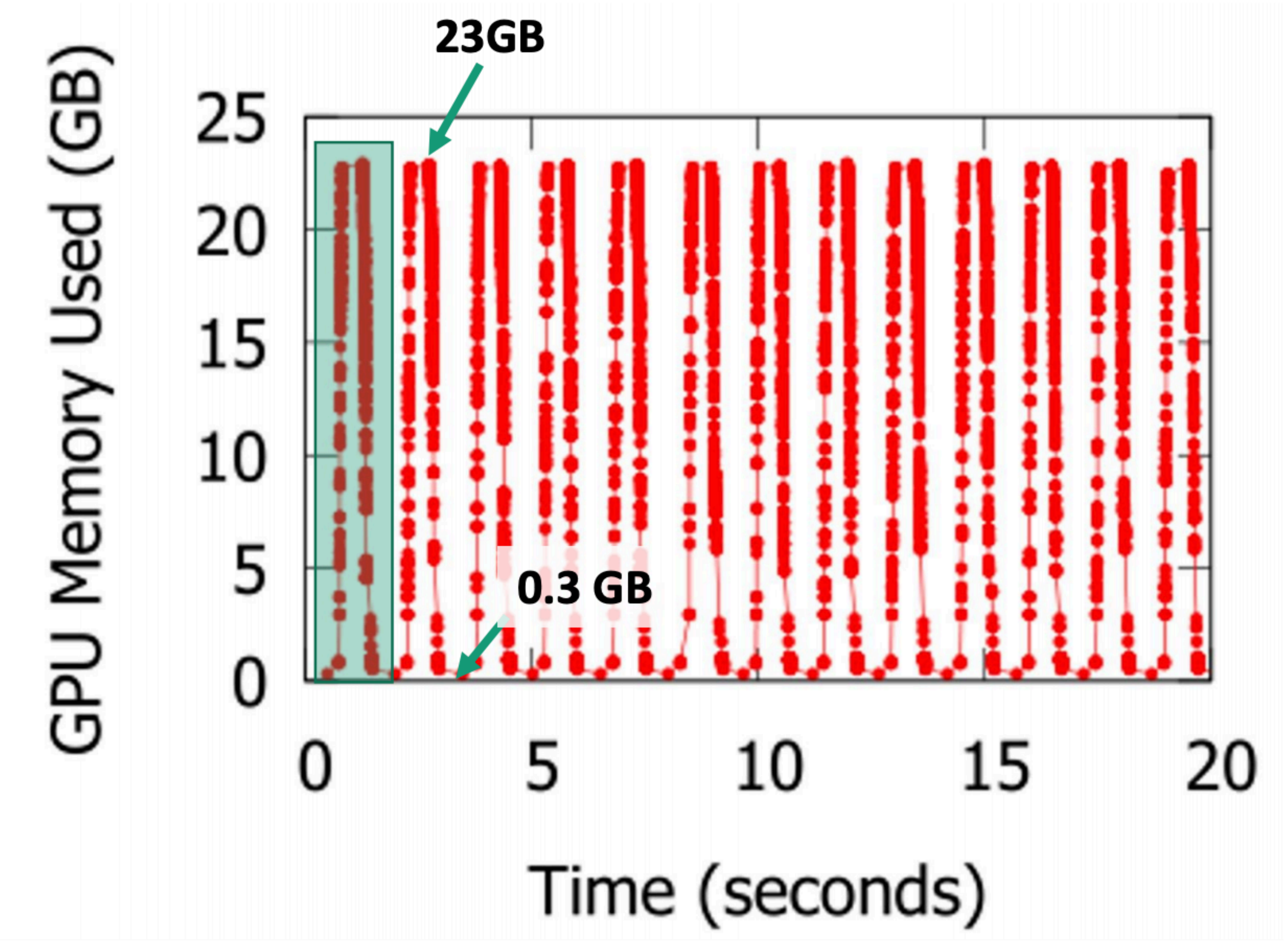
Naive Approach



Independent GPU Instances

Issues with Naive Approach?

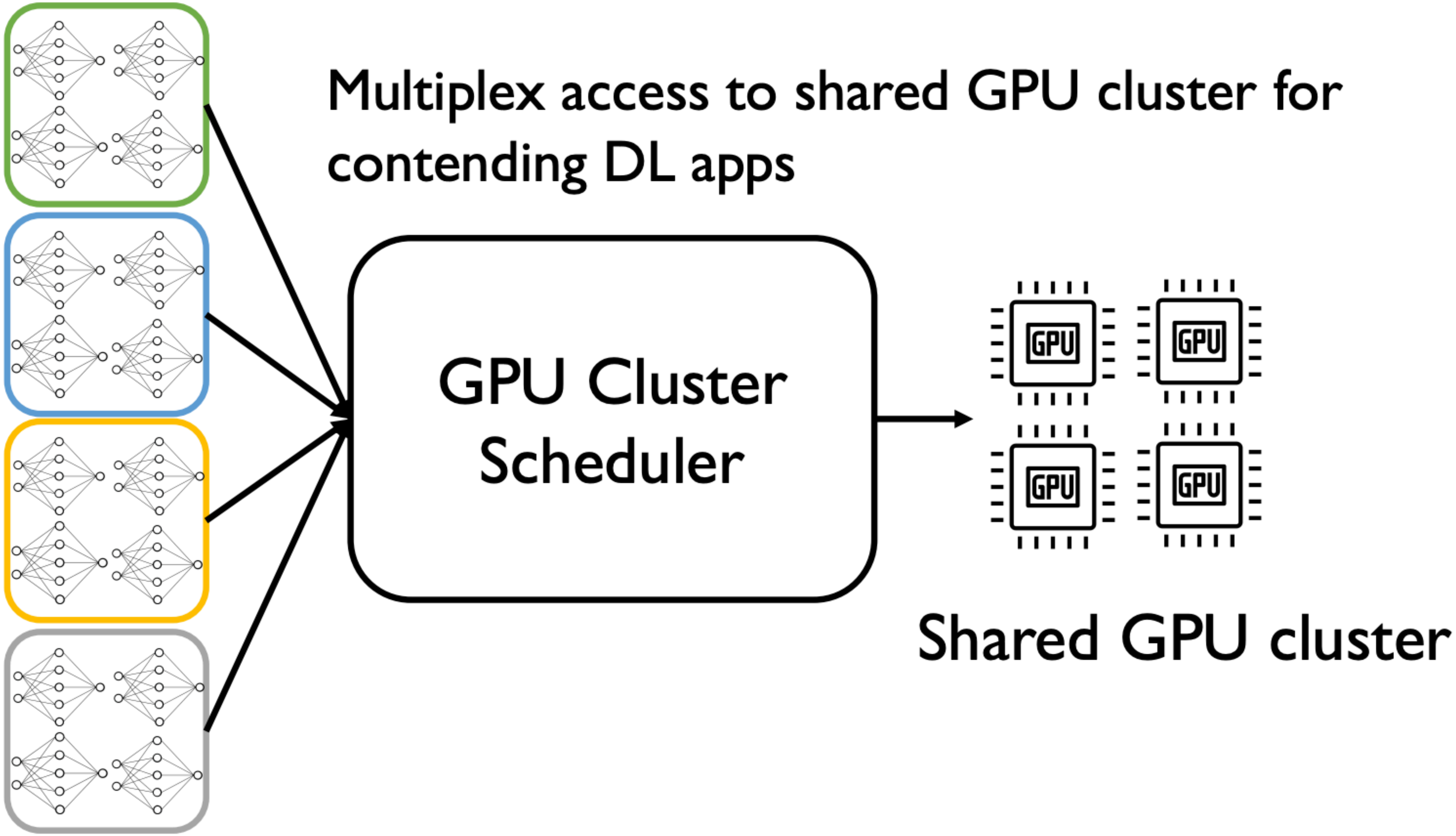
- High latency due to head of line blocking
- Low efficiency due to fixed decisions at job-placement time
- Unknown execution time of DL training jobs
- Low utilization



ResNet50 training on ImageNet data

Cluster Scheduler Goals

Multiplex access to shared GPU cluster for contending DL apps



Gandiva [OSDI'18]

- Key characteristics
  - Time-slicing
  - Migration
  - Application-aware profiling

Gandiva: Time-slicing

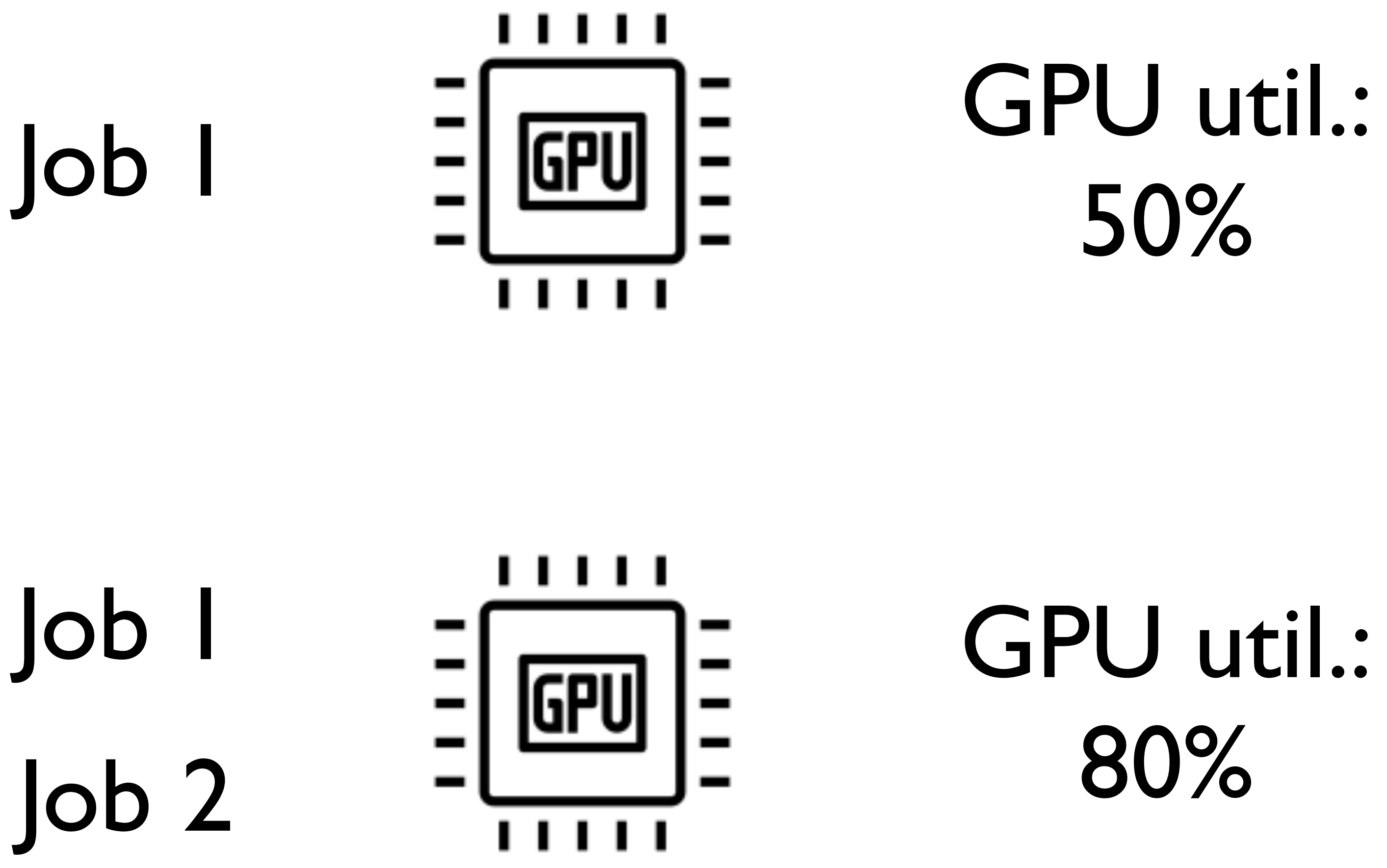
- Over-subscription as a first-class feature (similar to OS)
  - Time quantum of ~1 min (~100 mini-batches)
  - Better than queueing: Faster time-to-early feedback

Gandiva: Migration / Packing

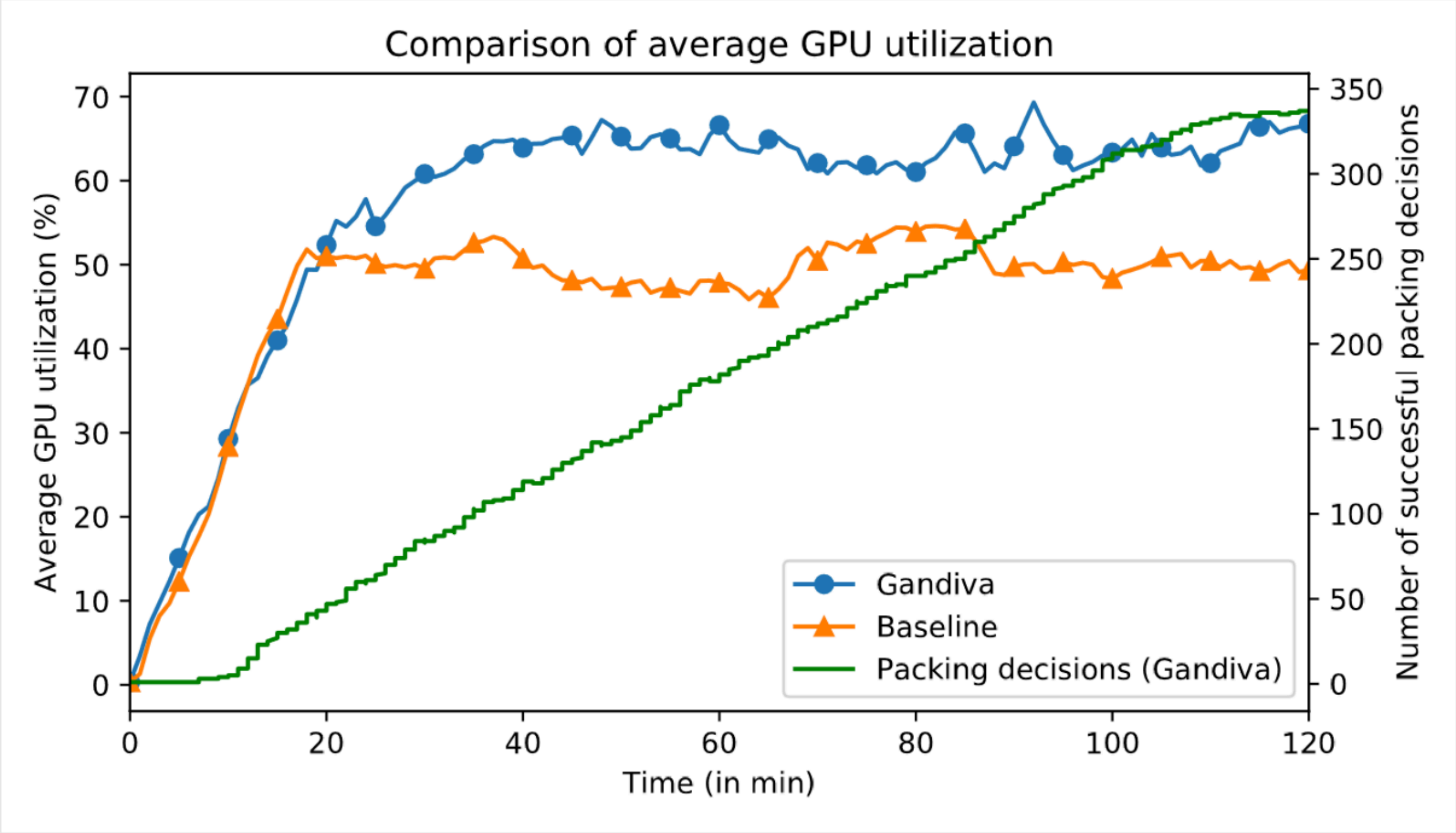
- Move jobs across GPUs to improve efficiency
- Generic distributed process migration is unreliable / slow
    - Solution: Integration with toolkit checkpointing makes it fast/robust
- Scenarios where it helps
    - De-fragment multi-GPU jobs
    - Exploit heterogeneity
    - Pack multiple jobs onto the same GPU

Gandiva: Application-aware profiling

- Two possibilities in utilization change
  - 30% more useful work done
  - Overhead due to interference (could be net loss)
- Solution: Measure useful work directly
  - Job runtime exports “time-per-minibatch”
- Allows simple “introspection” policy
  - Try migration/packing, measure benefit, revert if negative



Gandiva: Performance



Cluster of 180 GPUs

Synthetic DLT jobs modelled from a production trace

Efficiency

Cluster throughput improves by 26%

Latency

4.5x reduction in avg. time to first 100 mini-batches

Gandiva Shortcomings

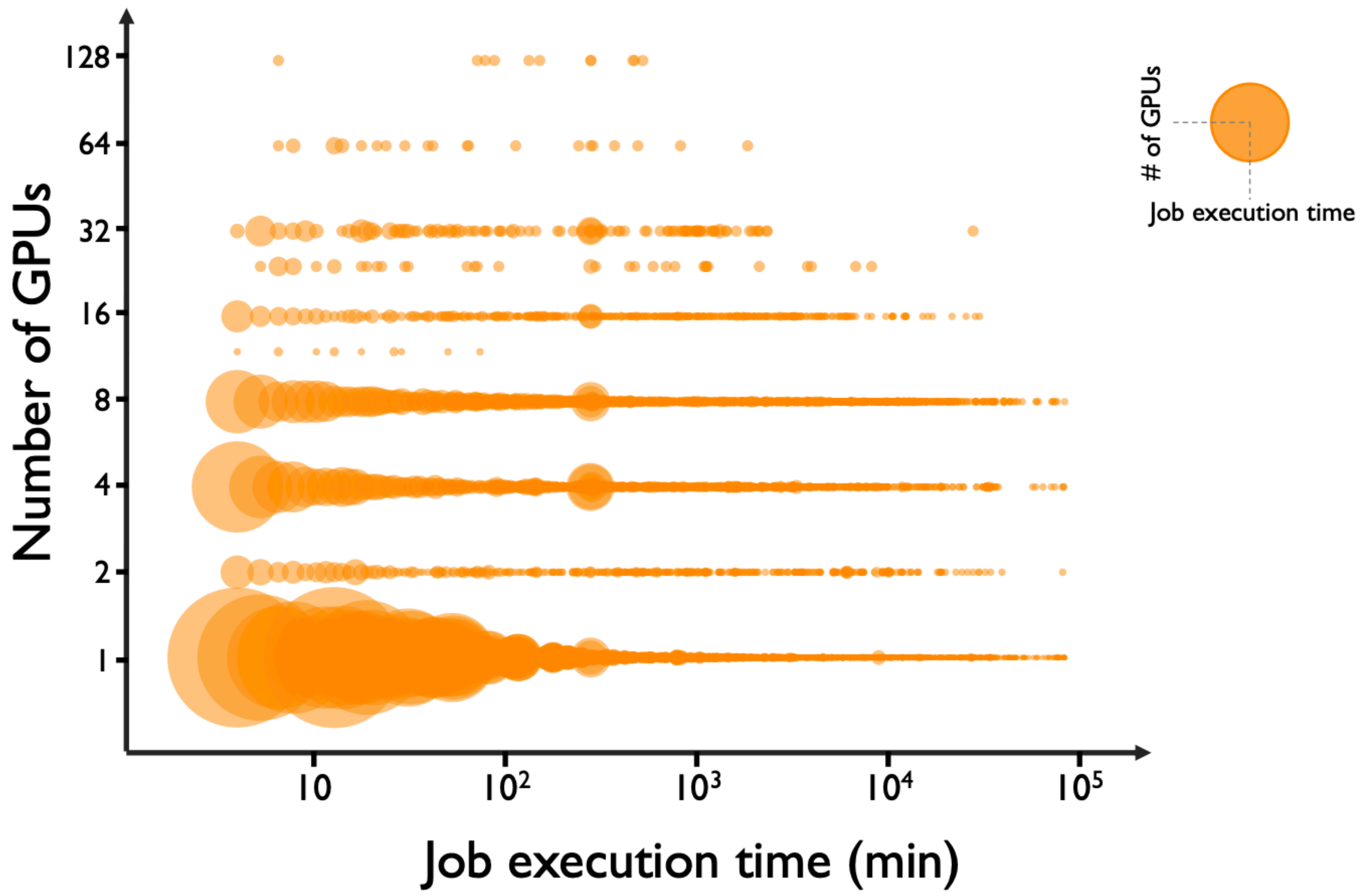
- Time-Sharing based design
    - Works well for fairness, but does not optimize for job completion time
- Job placement
    - Works well when complete information of job is available
    - If no affinities specified, placement is based on trial and error

Tiresias [NSDI'19]

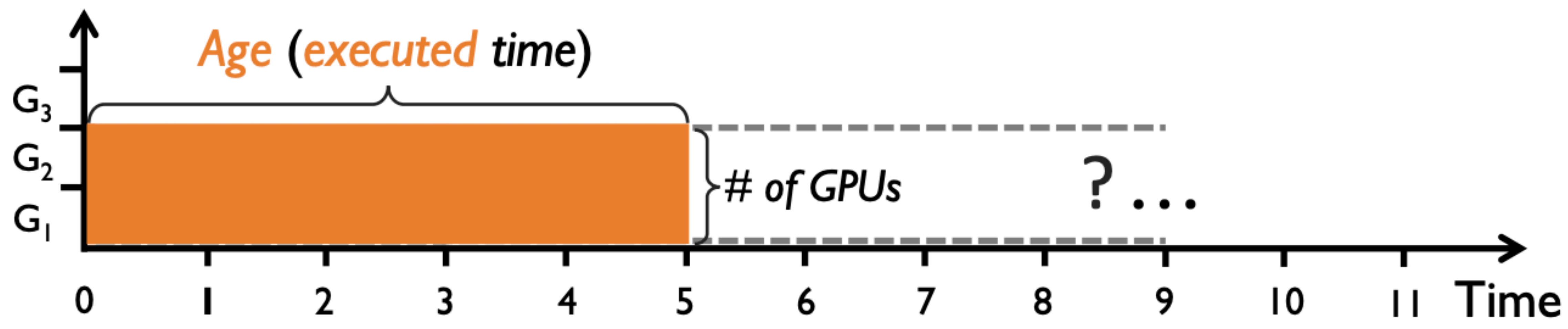
- Key characteristics
  - Age-based scheduler
    - Minimize Job Completion Time (JCT) without complete knowledge about the job
  - Model Profile-based Placement
    - Place jobs without additional information from users
    - Relies on a model profiler

Tiresias Motivation

- Variations in temporal and spatial aspects of DL training jobs



Tiresias: Age-Based Scheduling Background



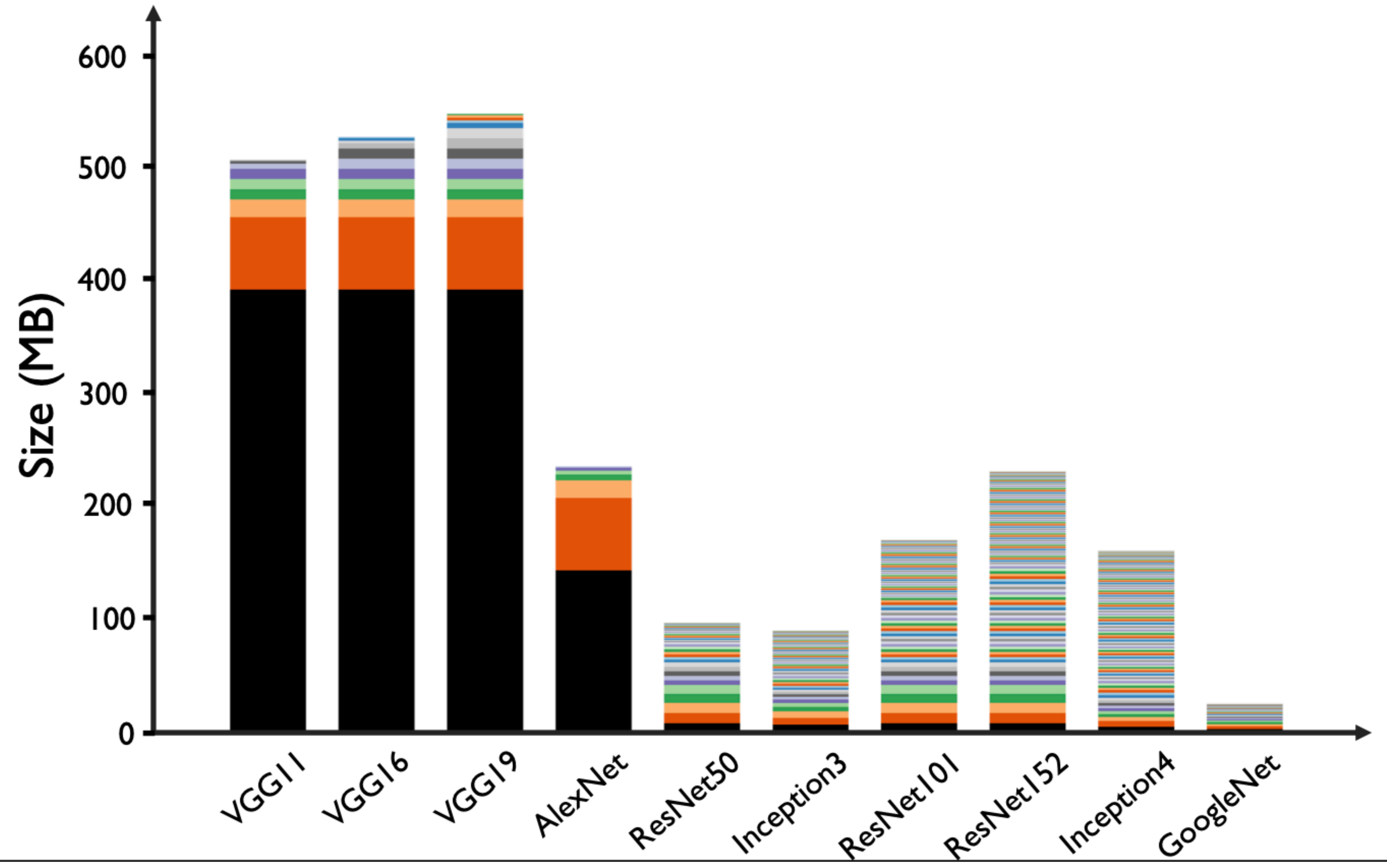
- Least-Attained Service (LAS)
  - Prioritize job that has the shortest executed time
- Gittins Index policy
  - Need the distribution of job execution time
  - Prioritize job that has the highest probability to complete in the near future

Tiresias: Two-Dimensional Age-Based Scheduler (2DAS)

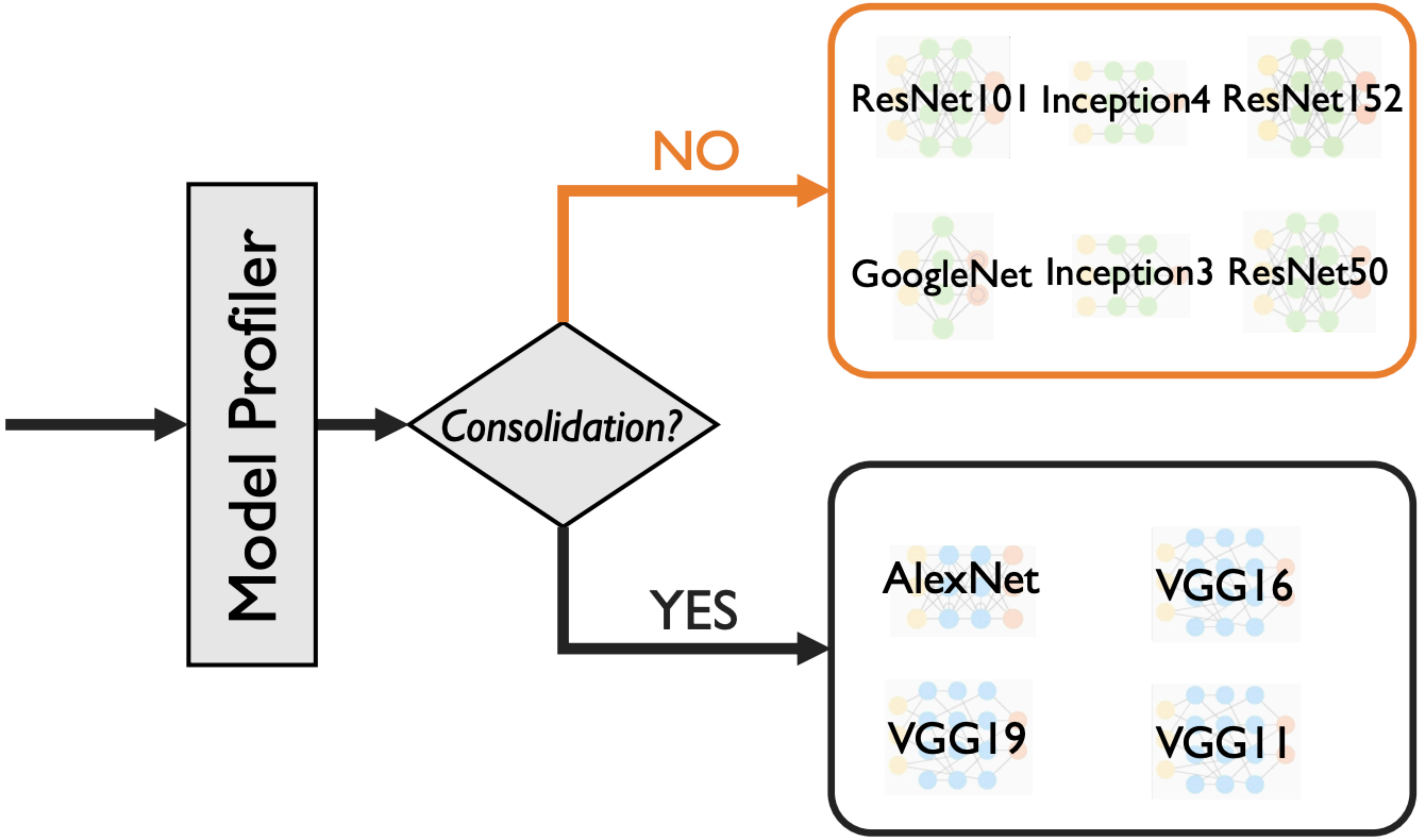
- Age calculated by two-dimensional attained service
  - i.e., a job's total executed GPU time (# of GPUs × executed time)
- No prior information
  - 2D-LAS
- With partial information: distribution of job GPU time
  - 2D-Gittins Index

Tiresias: Model Profile-Based Placement Motivation

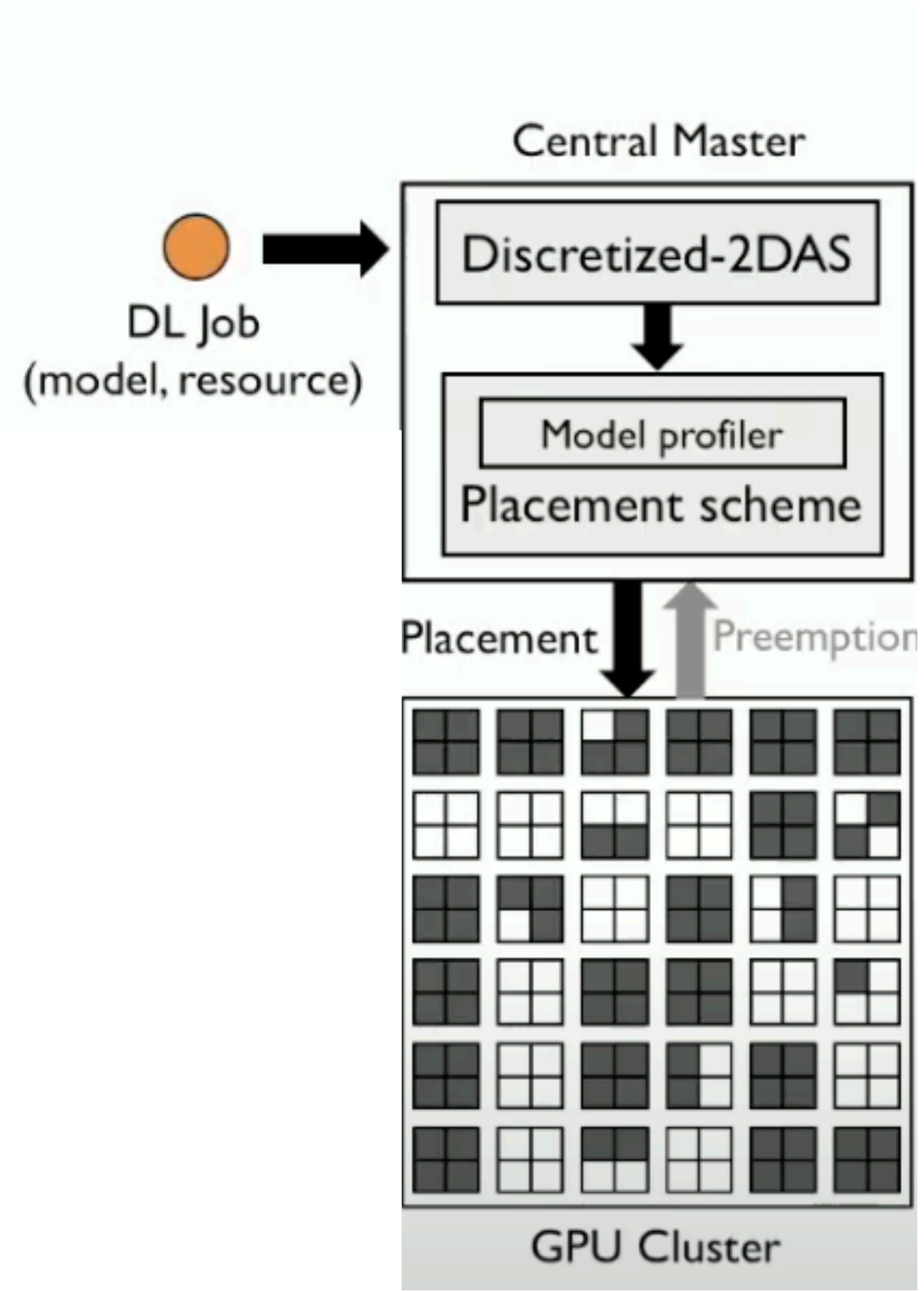
- Skewed distribution of tensors in DL models
- Large tensors cause network imbalance and contention
- Consolidated placement is needed when the model is highly skewed in its tensor size



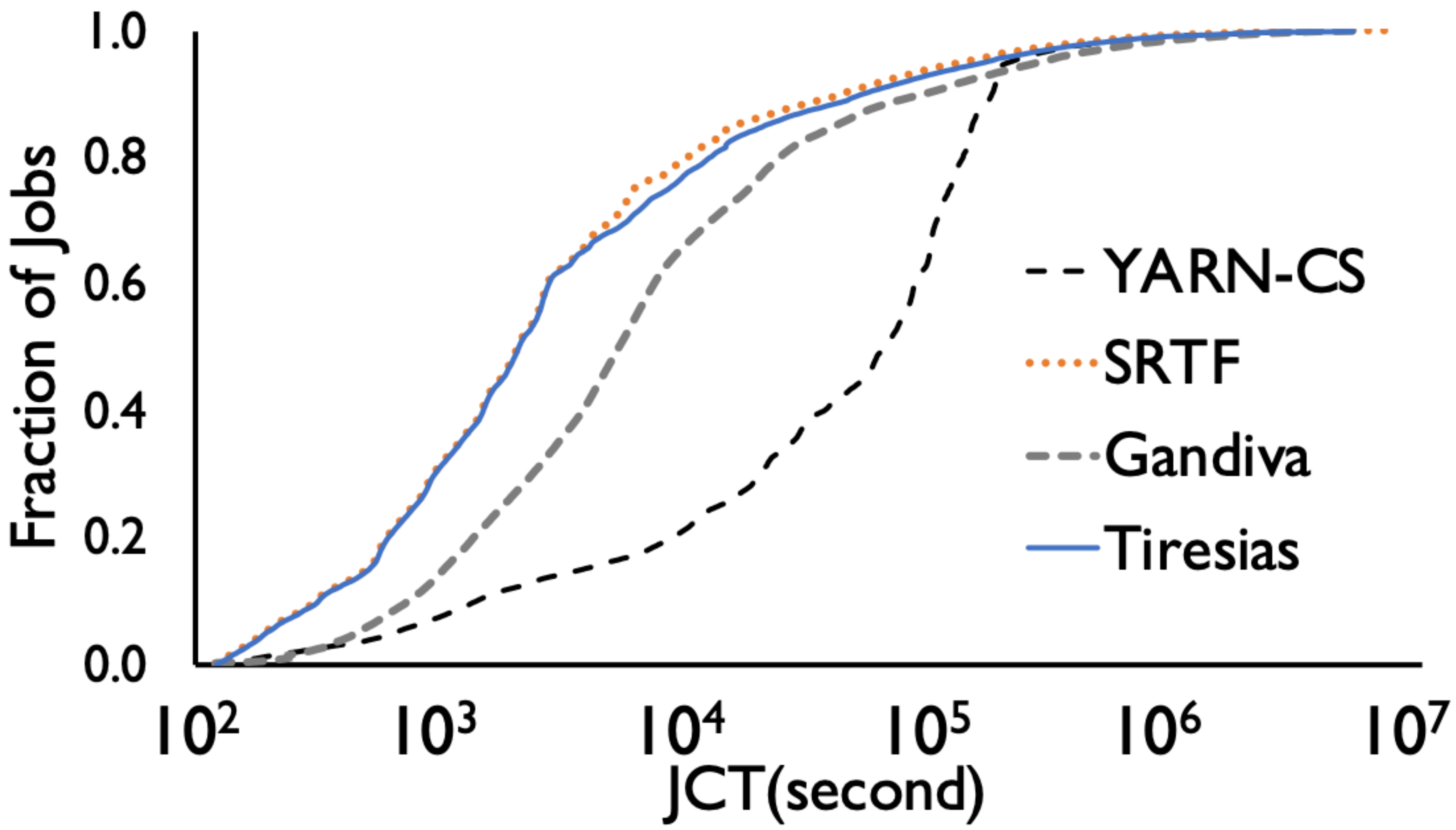
Model Profile-Based Placement



Tiresias System Model



Tiresias: Evaluation



Tiresias Summary

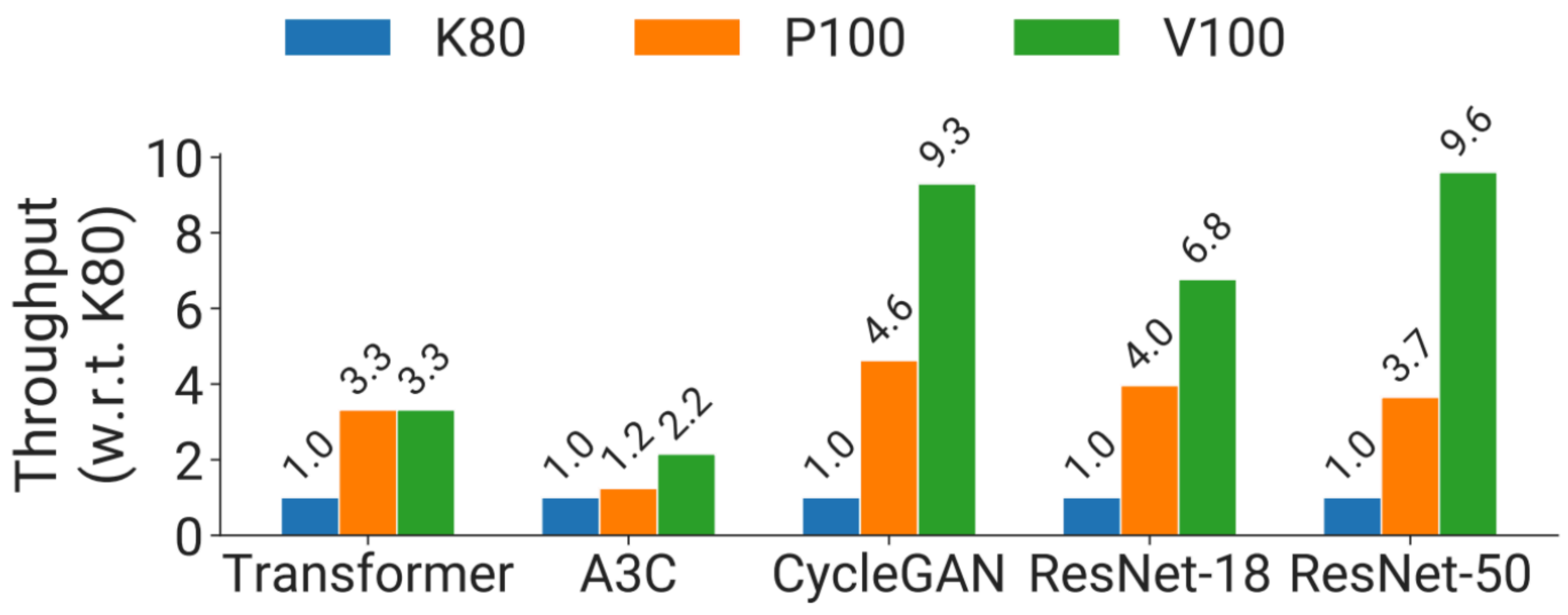
- Takes into account both spatial and temporal aspect
- Can optimize job completion time with no or partial job information
- Cannot handle diverse objectives (e.g., some parts of the cluster need fairness, others care about completion time)

Gavel [OSDI'20]

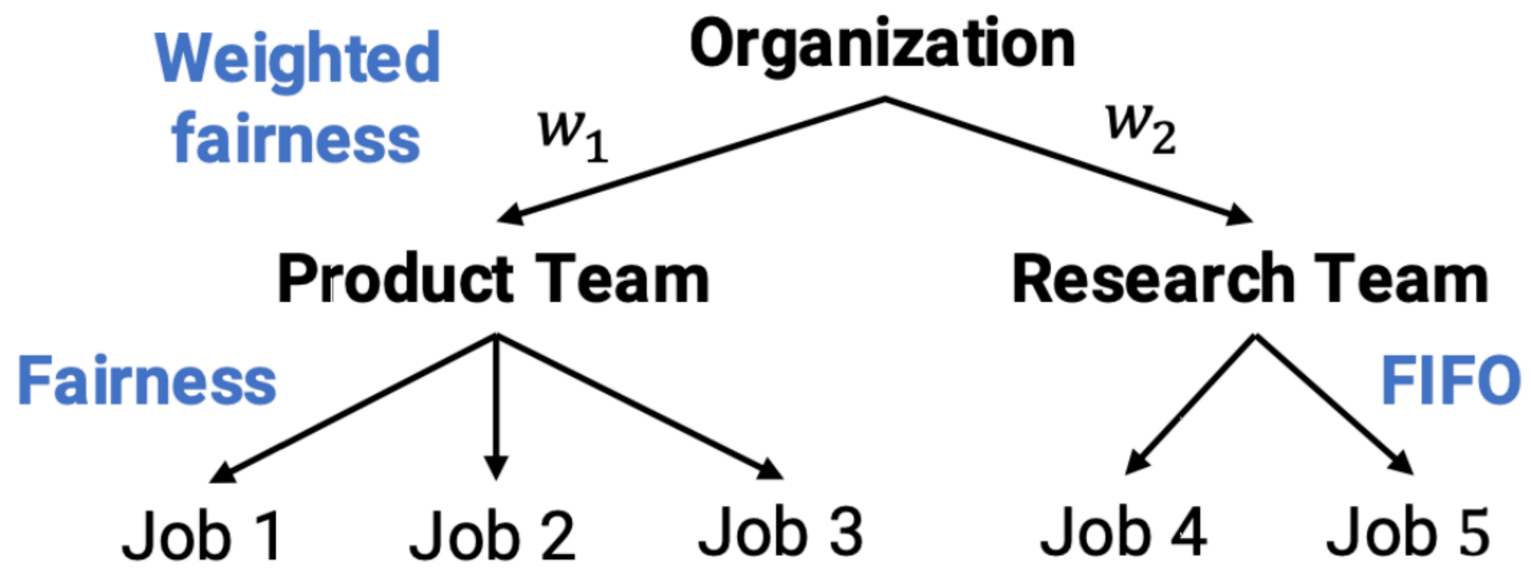
- Key characteristics
    - Generalizes a wide range of existing scheduling policies
    - Heterogeneity-aware Policies
    - Round-based Scheduling Mechanism

Gavel: Motivation

- Heterogeneous Performance
  - Models and operators (e.g., convolution, attention) perform differently across hardware architectures
  - Disregarding heterogeneity can lead to unfair allocations



- Diverse scheduling objectives
  - Single-job objectives: "maximize throughput" or "minimize cost"
  - Multi-job objectives: fairness or more complicated hierarchical policies



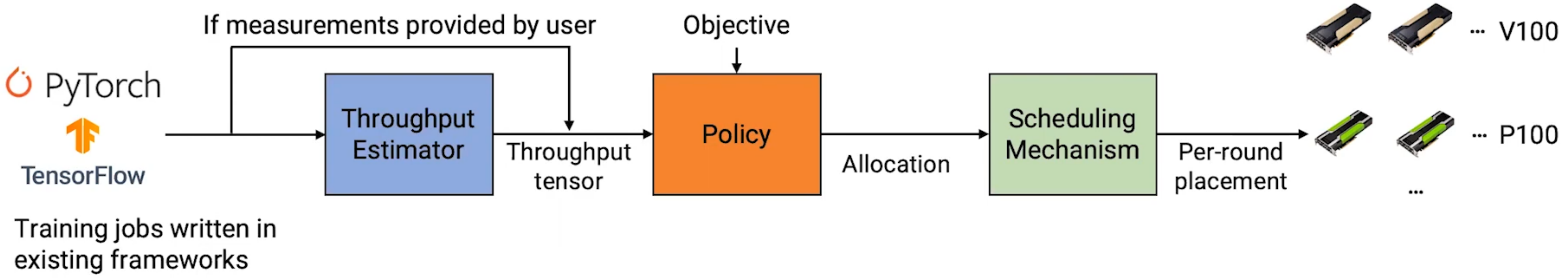
Hierarchical policy: Weighted fairness

Gavel: Heterogeneity-Aware Scheduling Policies

- FIFO: First in, first out
- Shortest Job First: Minimize time taken by shortest job
- Minimize Makespan: Minimize time taken by batch of jobs
- Minimize cost (w/ SLOs): Minimize total cost in public cloud (subject to SLOs)
- LAS: Max-min fairness by total compute time
- LAS w/ weights: Max-min fairness by total compute time with weights
- Finish Time Fairness: Maximize minimum job speedup
- Hierarchical: Multi-level policy with fairness as top-level policy, and FIFO or fairness as lower-level policies. Per-job weights can be specified

Gavel: Heterogeneity Aware Cluster Scheduler

- Generalizes a wide range of existing scheduling policies by expressing policies as optimization problems over the allocation
- Provides abstraction to incorporate performance heterogeneity
- Round-based scheduling mechanism ensures jobs receive optimal allocation
- Improves objectives such as average job completion time by 3.5×



Gavel: Policies as Optimization Problems

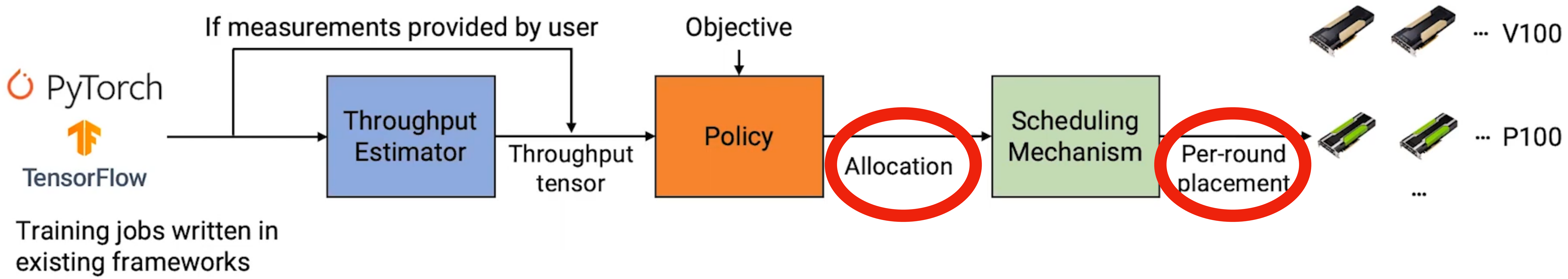
- In a homogeneous cluster, policy objectives are functions of throughput (e.g., duration = training steps / throughput) and allocation
- On a homogeneous cluster, Least Attained Service policy is a max-min fairness policy that equalizes the total compute time each job receives
- Jobs can see unequal throughput reductions on heterogeneous clusters
- To make policies heterogeneity-aware, policy objectives are expressed in terms of effective throughput
- Optimal allocations computed using linear programs

Gavel: Round-Robin Based Scheduling Mechanism

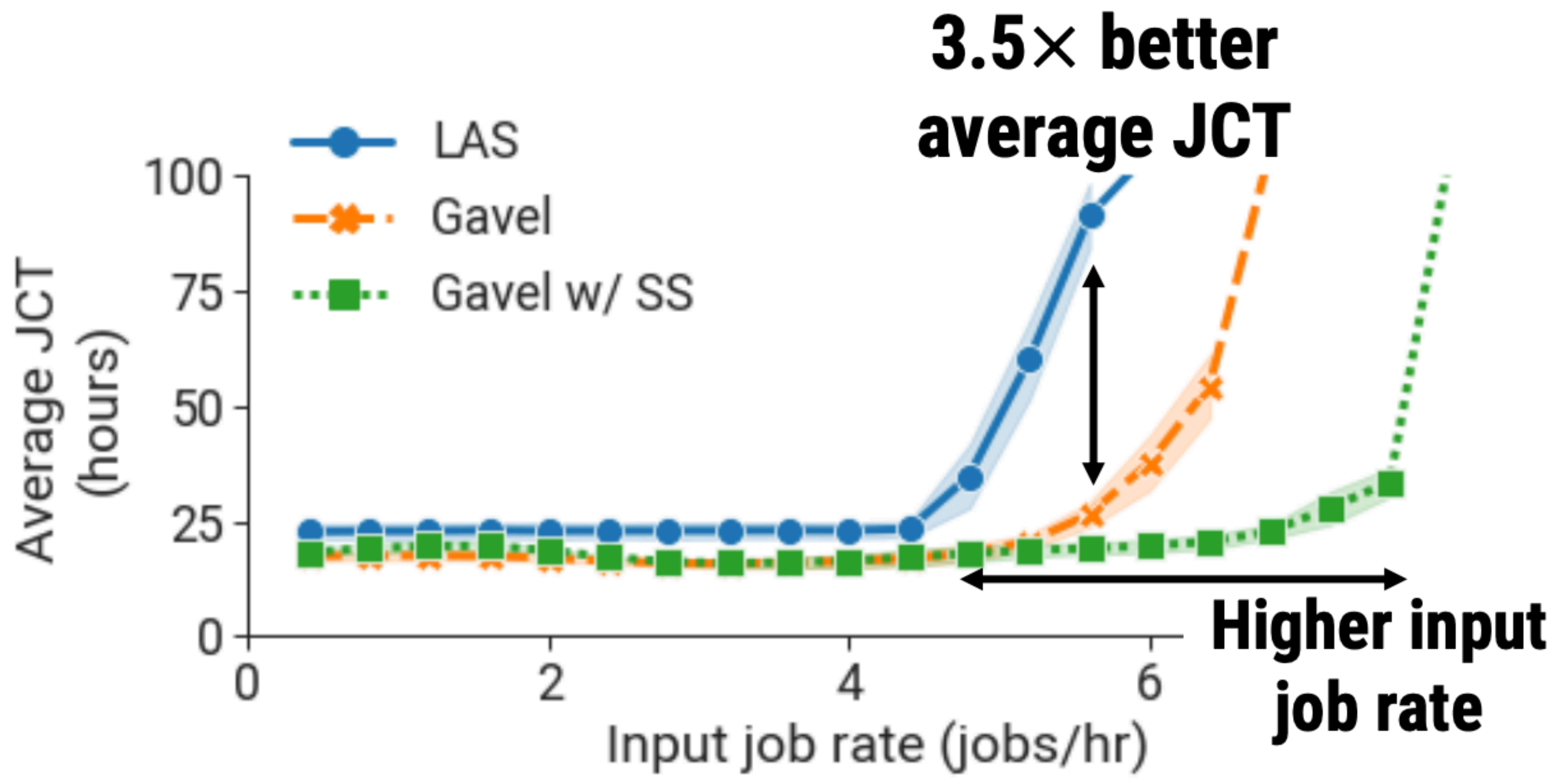
- Ensures jobs receive time on accelerator types according to the computed optimal allocation

- Priority score for every (job, accelerator) combination

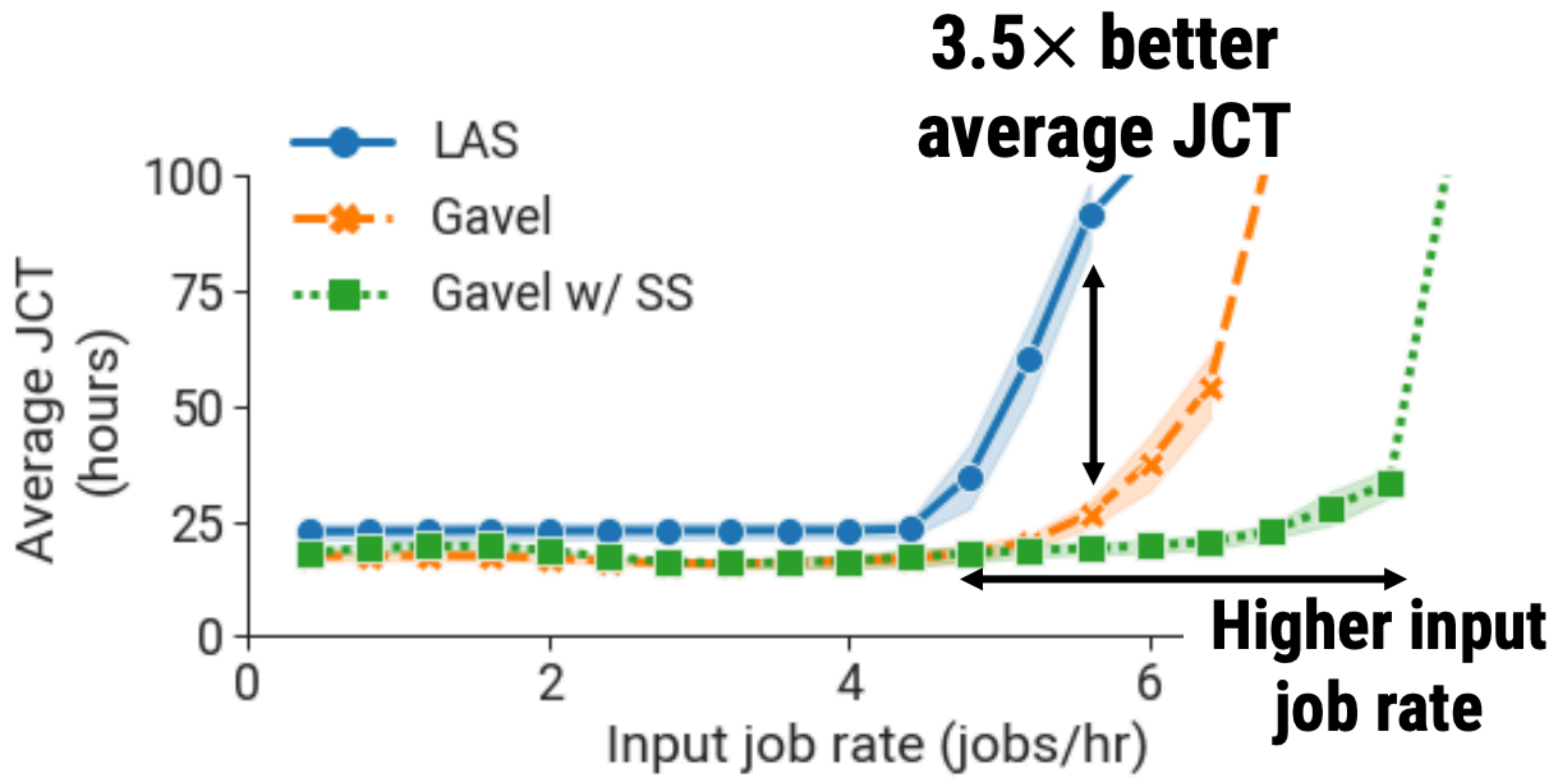
- Jobs placed on resources where they have priority

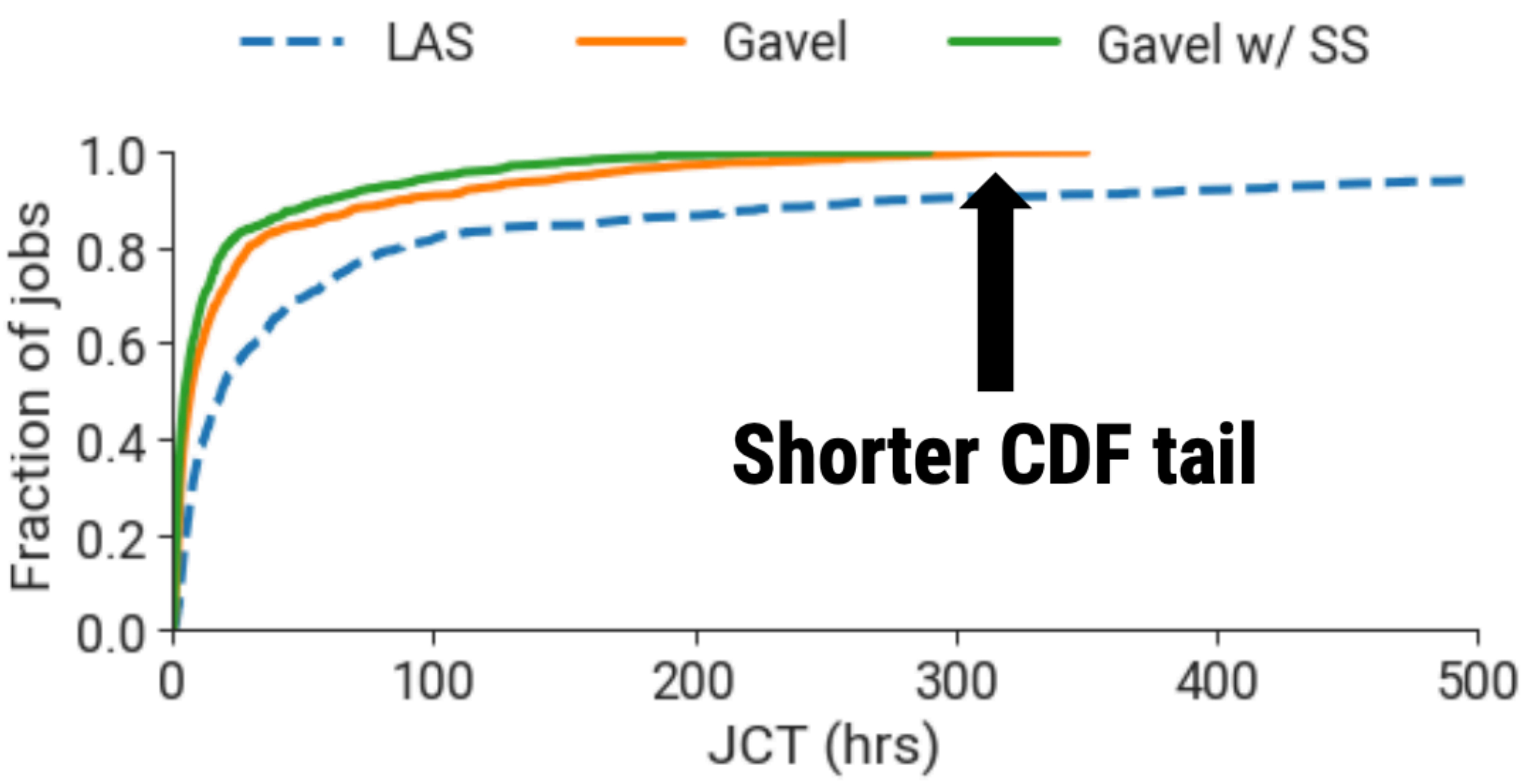


Gavel: Evaluation



Gavel: Evaluation





JCT CDF (input job rate = 5.6 jobs/hr)

Thanks!