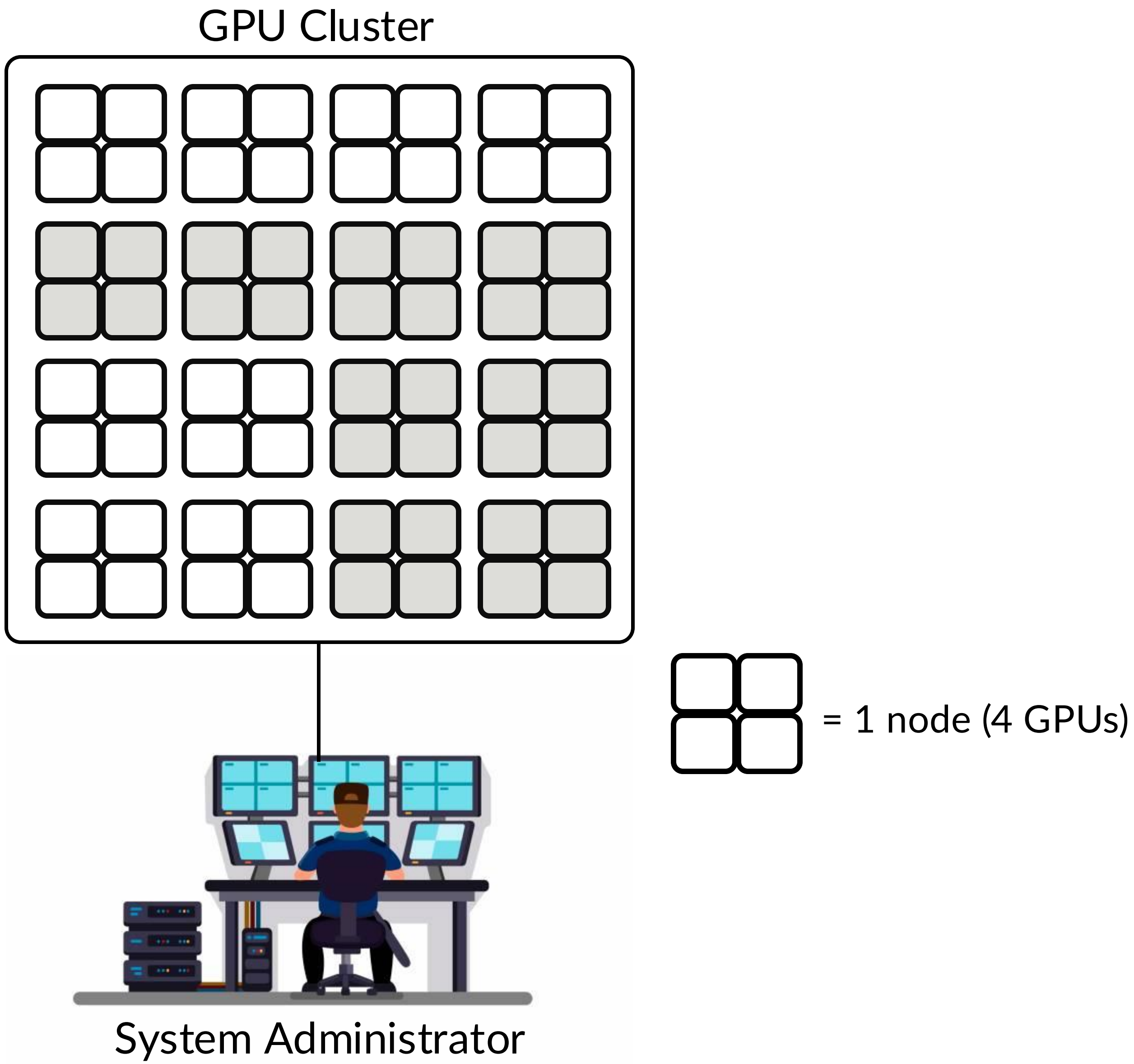
PAL: A VARIABILITY-AWARE POLICY FOR SCHEDULING ML WORKLOADS IN GPU CLUSTERS

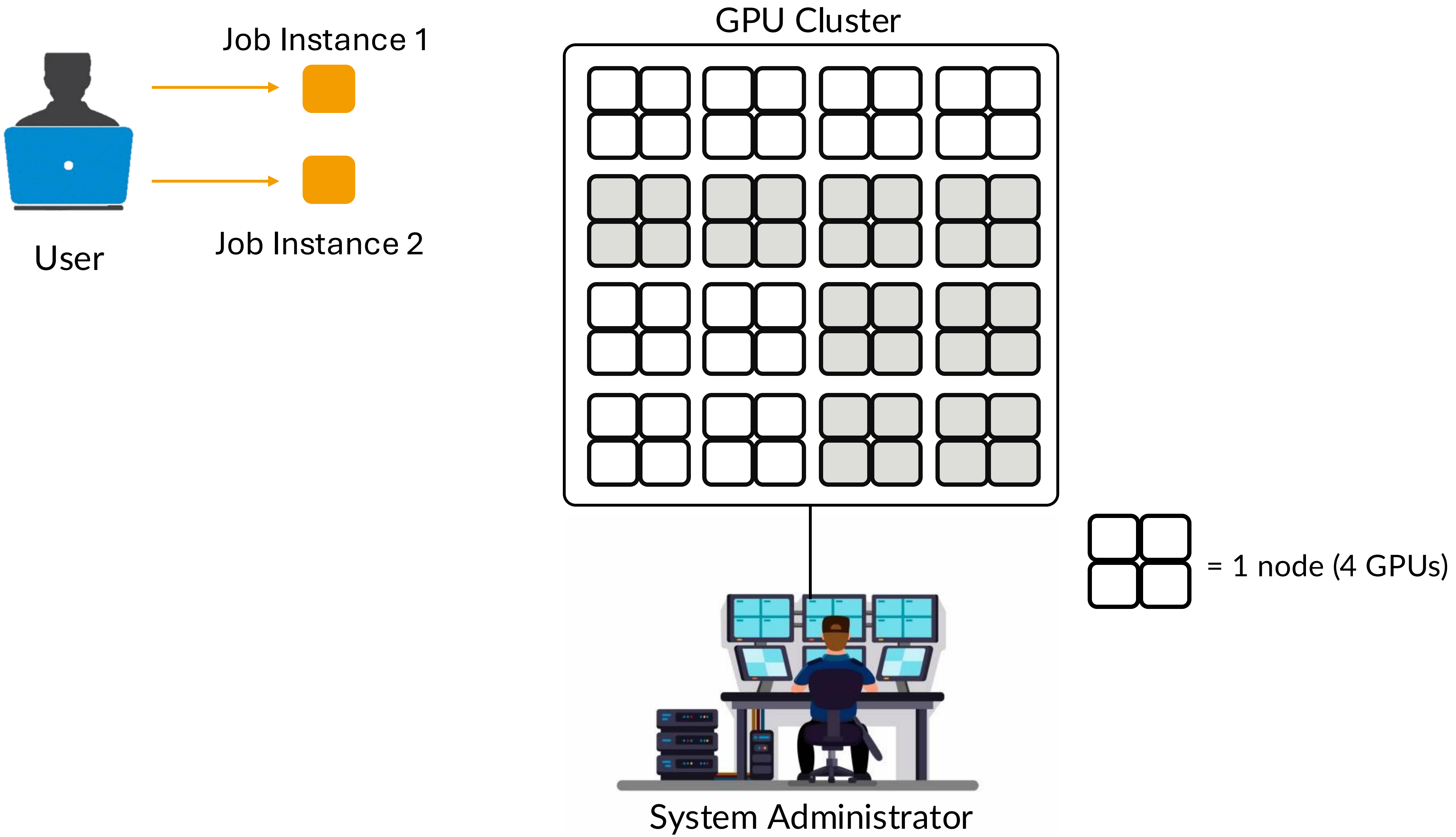
Rutwik Jain, Brandon Tran, Keting Chen, Matthew D. Sinclair, Shivaram Venkataraman

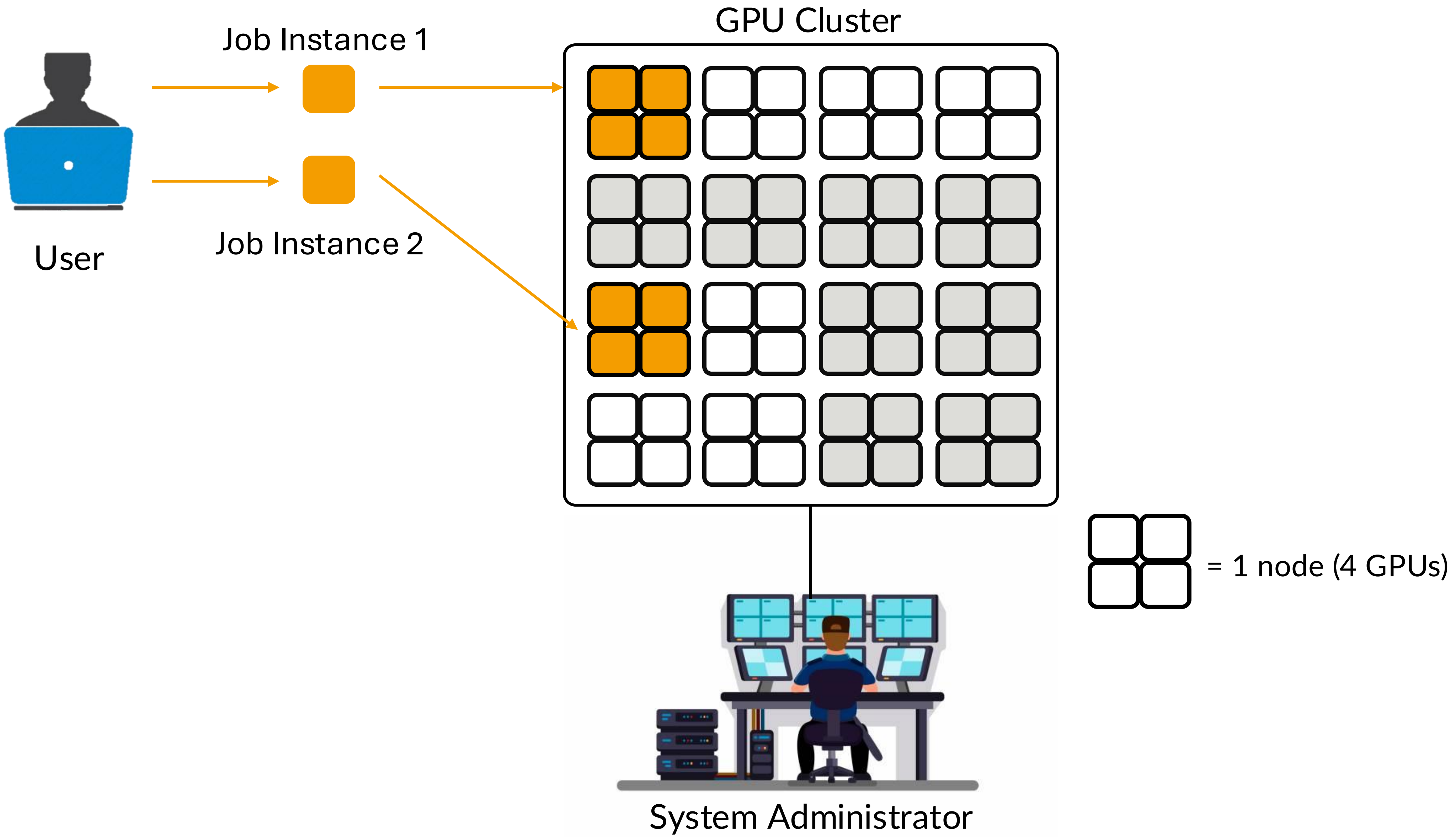
Computer Sciences Department, University of Wisconsin-Madison

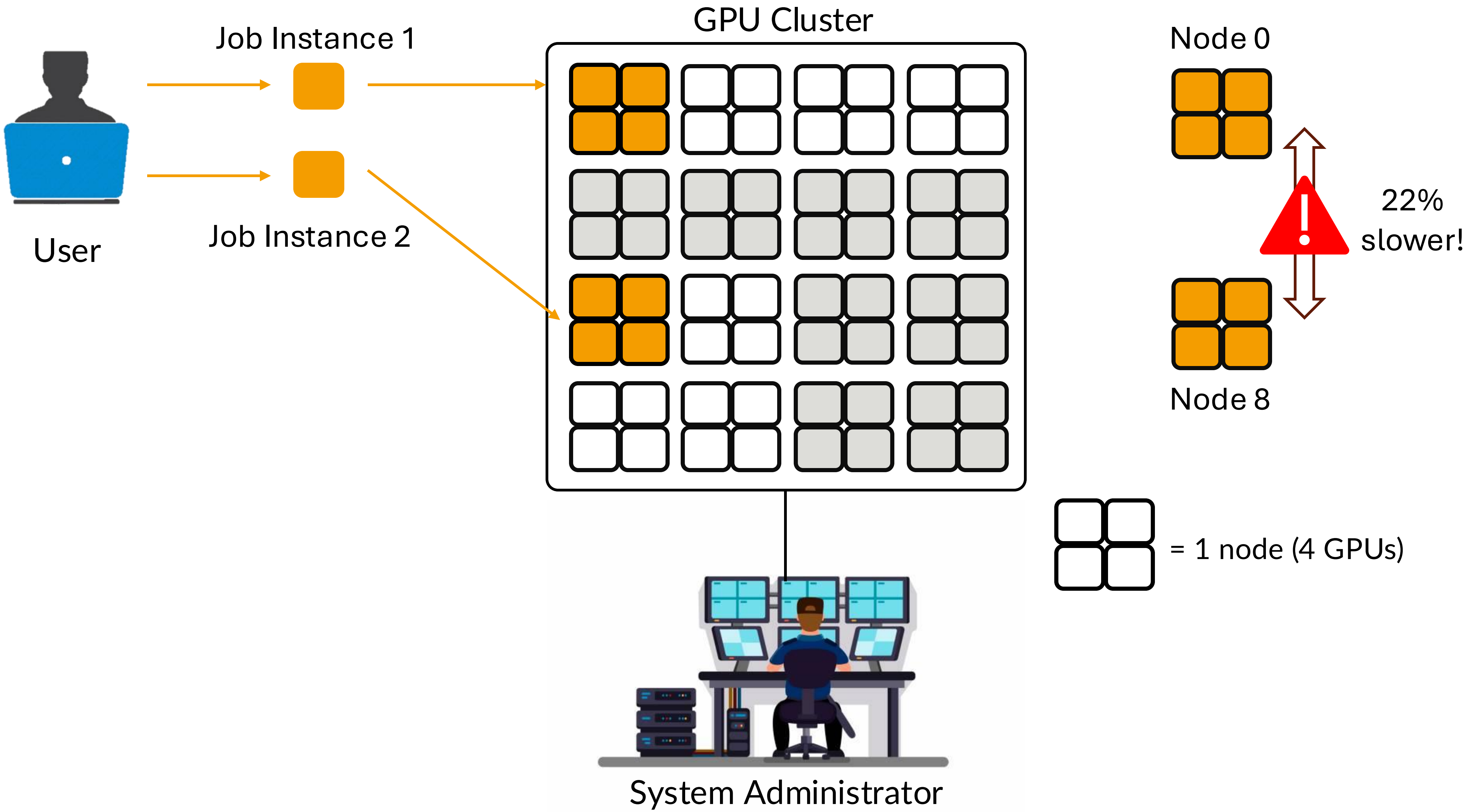
Computer Sciences
SCHOOL OF COMPUTER, DATA & INFORMATION SCIENCES
UNIVERSITY OF WISCONSIN–MADISON

SC24 Atlanta, GA | hpc creates.


GPU Cluster
= 1 node (4 GPUs)
System Administrator


Job Instance 1
User
Job Instance 2
GPU Cluster
= 1 node (4 GPUs)
System Administrator

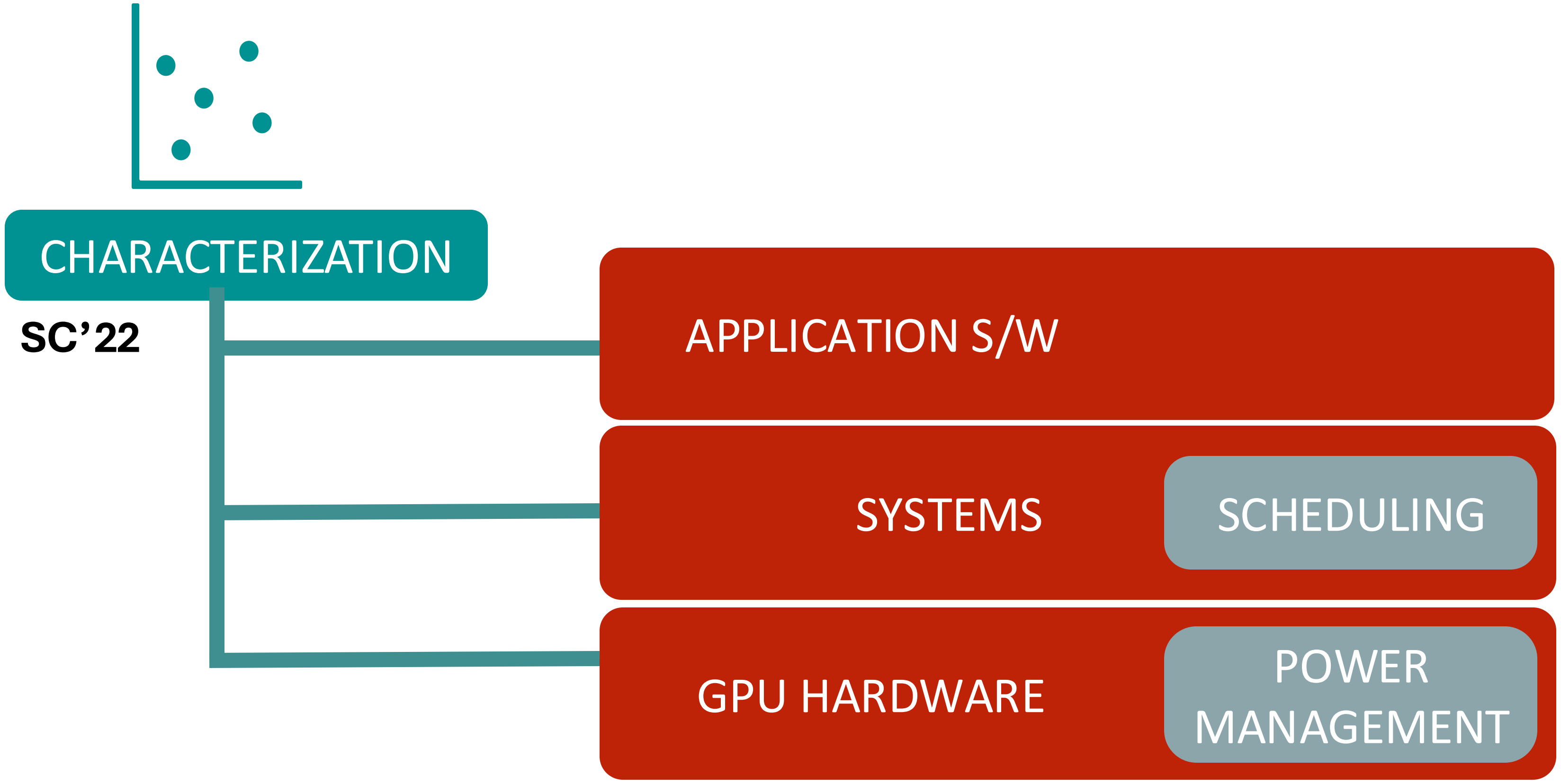

GPU Cluster
Job Instance 1
Job Instance 2
User
= 1 node (4 GPUs)
System Administrator


Job Instance 1
GPU Cluster
Node 0
User
Job Instance 2
22%
slower!
Node 8
= 1 node (4 GPUs)
System Administrator

GPU PERFORMANCE VARIABILITY

- Culprit: **Variability**!
- GPUs exhibit variability because of:
  - Local Power Management (PM)
  - Thermal/non-uniform cooling effects
  - Process variation
- **Impact**
  - Users: Hard to get **repeatable, consistent** performance for applications.
  - Sysadmins: **Resource underutilization** for multi-GPU jobs – faster GPUs wait for stragglers

CHARACTERIZING VARIABILITY



Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems (SC22)
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

CHARACTERIZING VARIABILITY

CHARACTERIZATION

SC'22

APPLICATION S/W

SYSTEMS

SCHEDULING

GPU HARDWARE

POWER MANAGEMENT

GPU vendors
GPU types

vs

Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems (SC22)
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

CHARACTERIZING VARIABILITY

CHARACTERIZATION

SC'22

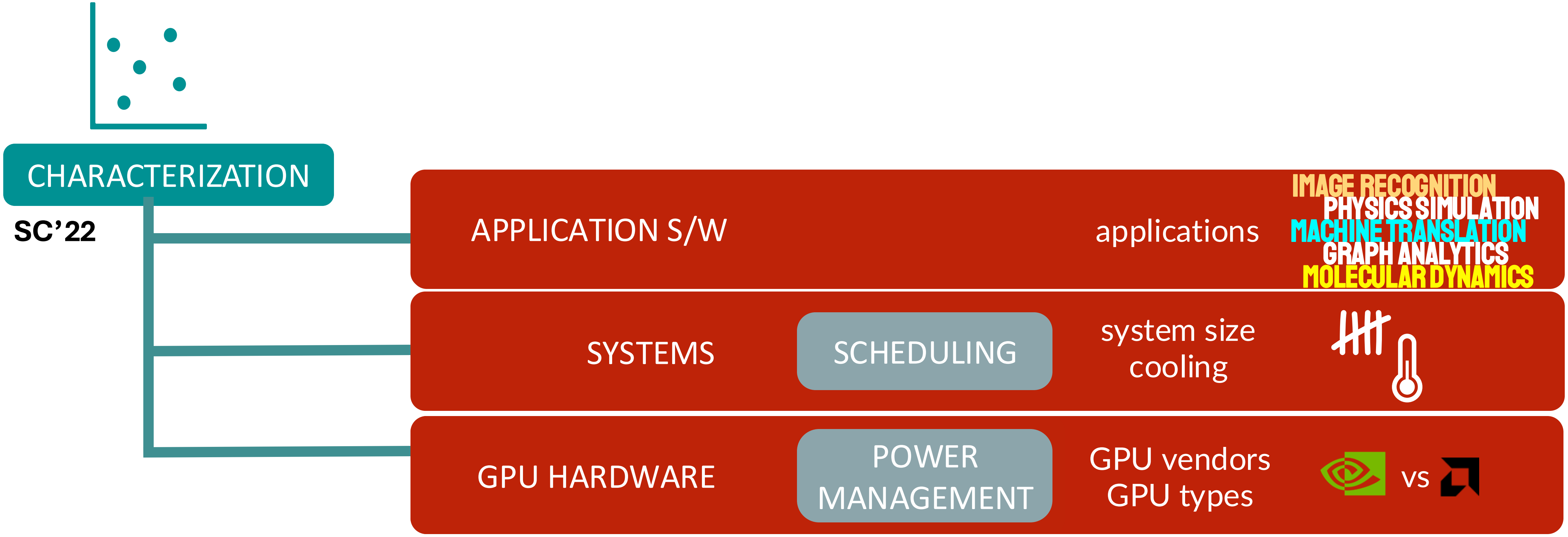
APPLICATION S/W

SYSTEMS | SCHEDULING | system size cooling

GPU HARDWARE | POWER MANAGEMENT | GPU vendors GPU types | vs

Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems (SC22)
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

CHARACTERIZING VARIABILITY



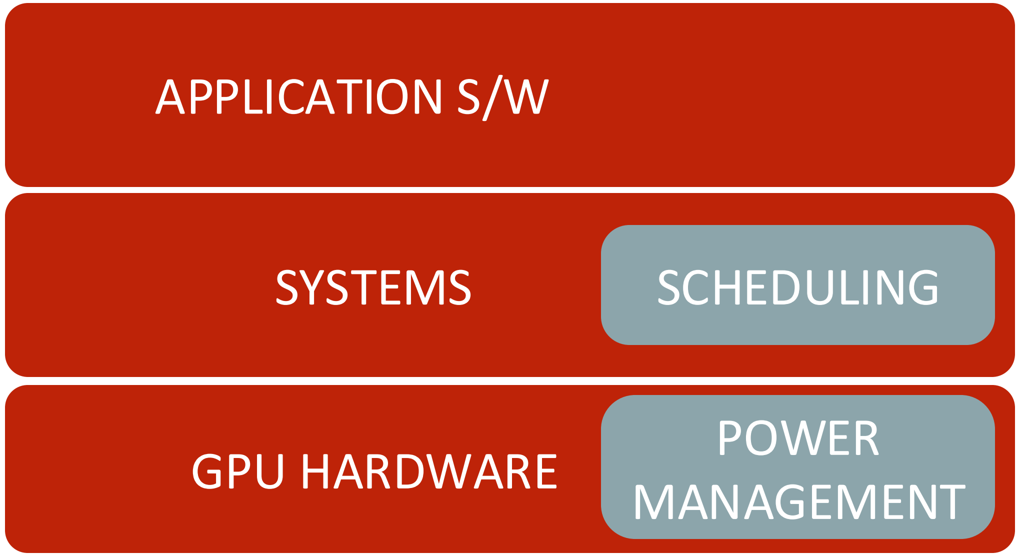
Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems (SC22)
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

CHARACTERIZATION STUDY: TAKEAWAYs

1. Consistent performance variability across clusters, GPU vendors and cooling methods (8% for SGEMM, outliers up to 1.5x slower)
2. Variability is not a transient effect, ill-performing GPUs are consistently ill-performing
3. **Variability is application-specific** – compute-intensive workloads are more variability sensitive (ResNet-50 shows 22% variability while PageRank has <1%)

Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

KEY INSIGHT: EMBRACE VARIABILITY



Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems (SC22)
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

KEY INSIGHT: EMBRACE VARIABILITY

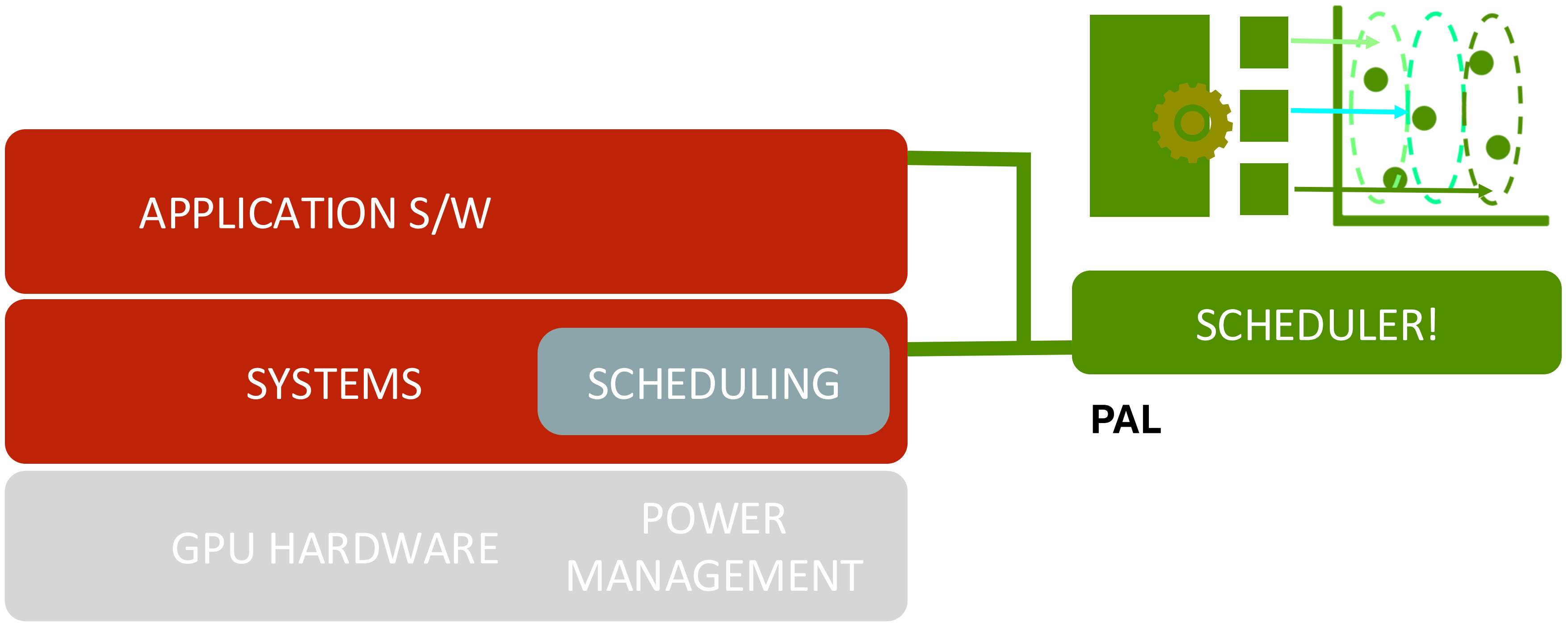
APPLICATION S/W

SYSTEMS | SCHEDULING

GPU HARDWARE | POWER MANAGEMENT

Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems (SC22)
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

KEY INSIGHT: EMBRACE VARIABILITY



Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems (SC22)
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

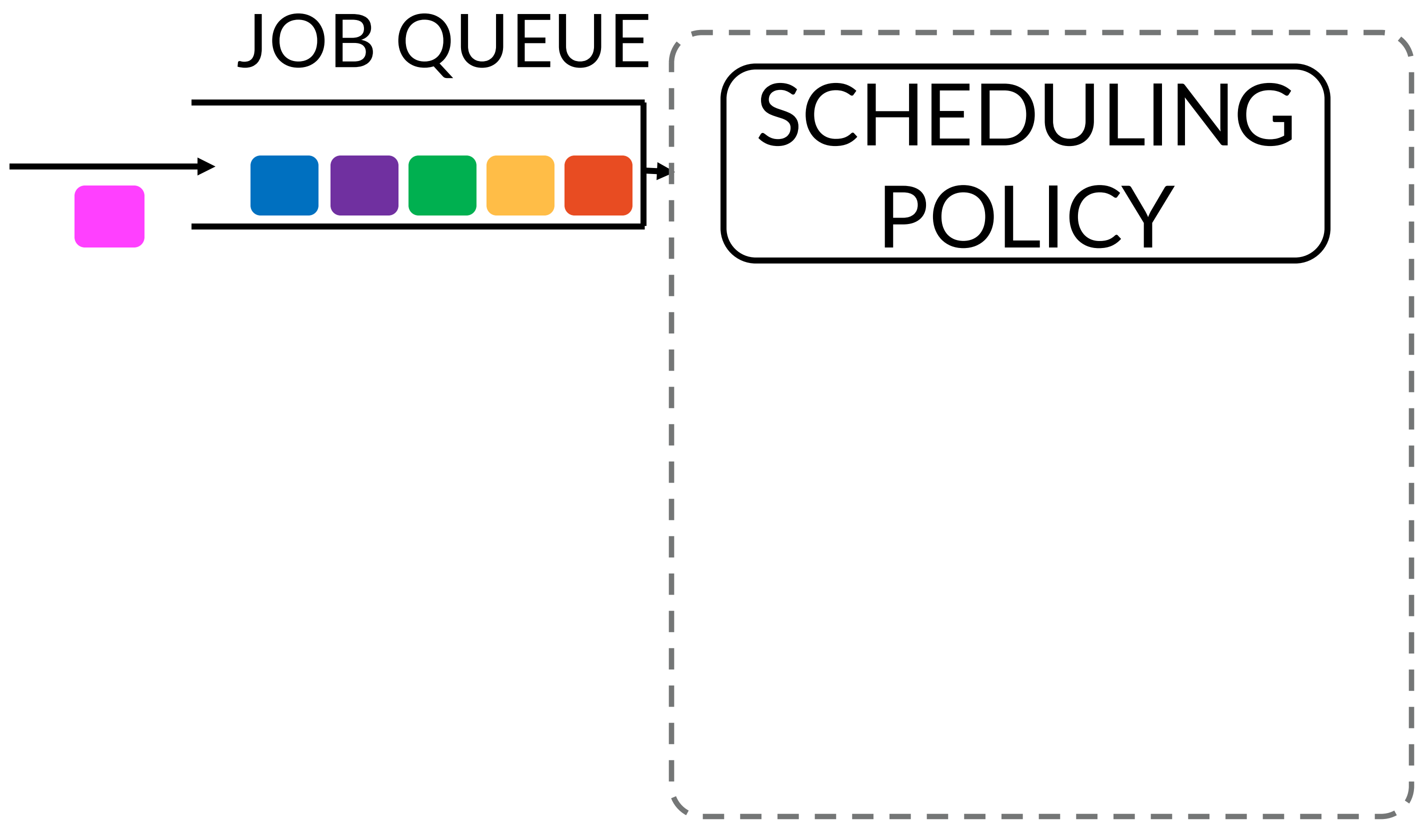
OUR APPROACH

- We characterize which applications are more likely to suffer from performance **variability** & take that into account **while placing jobs on the cluster.**
- Novel placement policies: **PM-First** and **PAL**
  - **PM-First** uses **application-specific variability profiles** to improve performance and utilization.
  - **PAL** further improves scheduling by balancing **variability** with **locality**.
- Overall, PAL improves geomean Job Completion Time (JCT) by **42%** and cluster utilization by **28%** over state-of-the-art schedulers like Tiresias

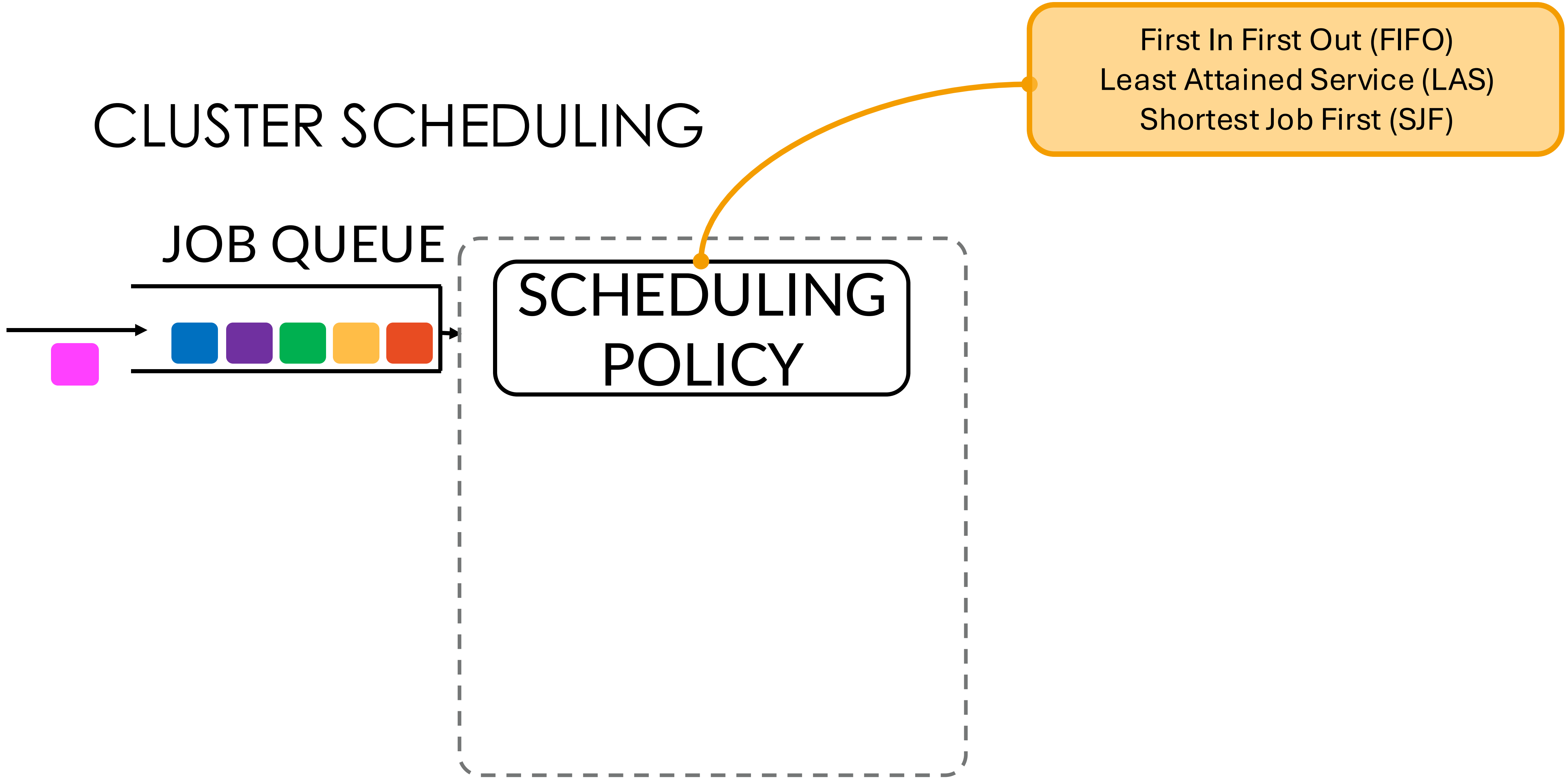
OUTLINE

- Introduction
- **Cluster Scheduling**
- **Design**
- **Methodology**
- **Evaluation**
- **Conclusion**

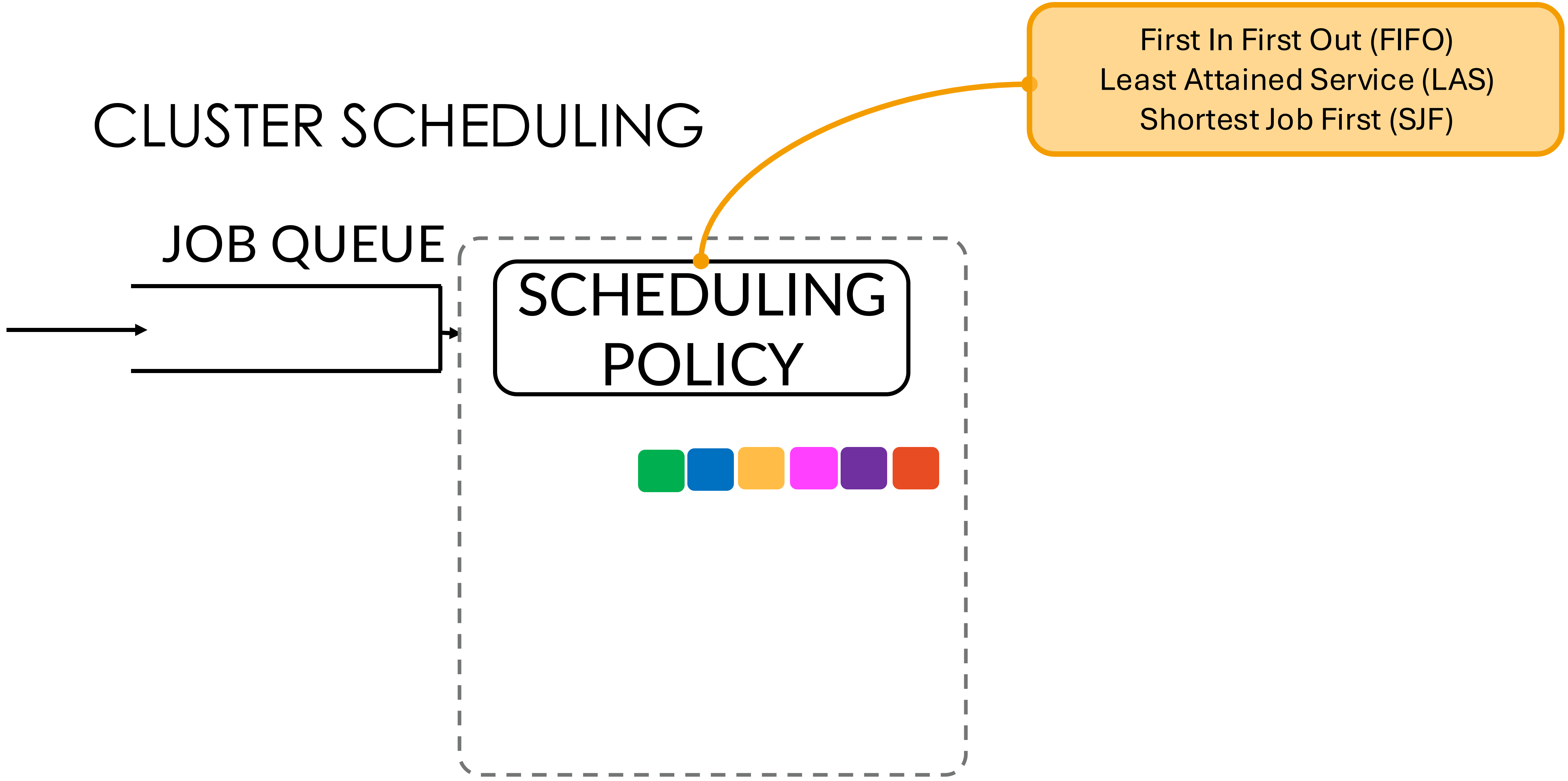
CLUSTER SCHEDULING



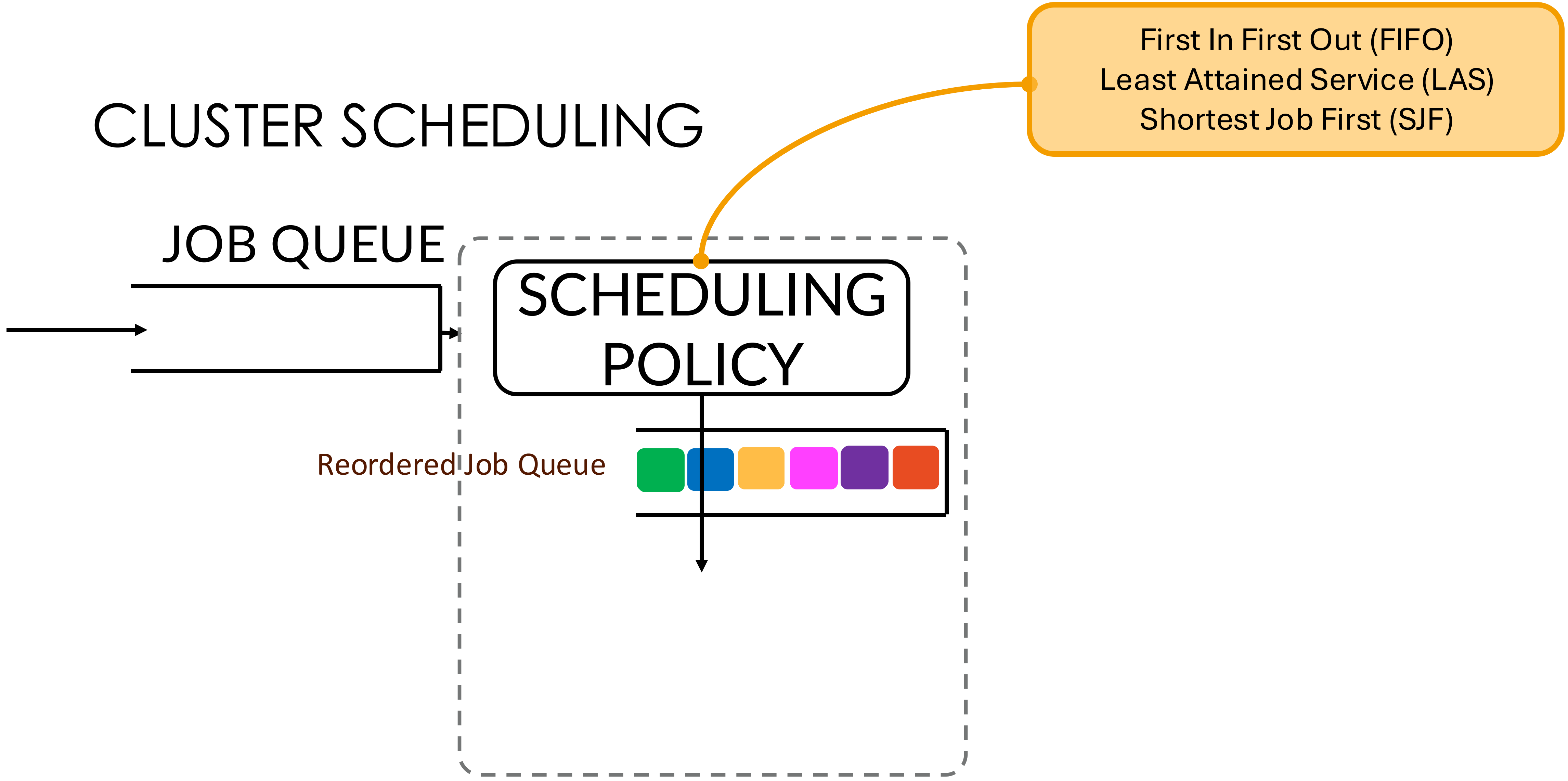
CLUSTER SCHEDULING



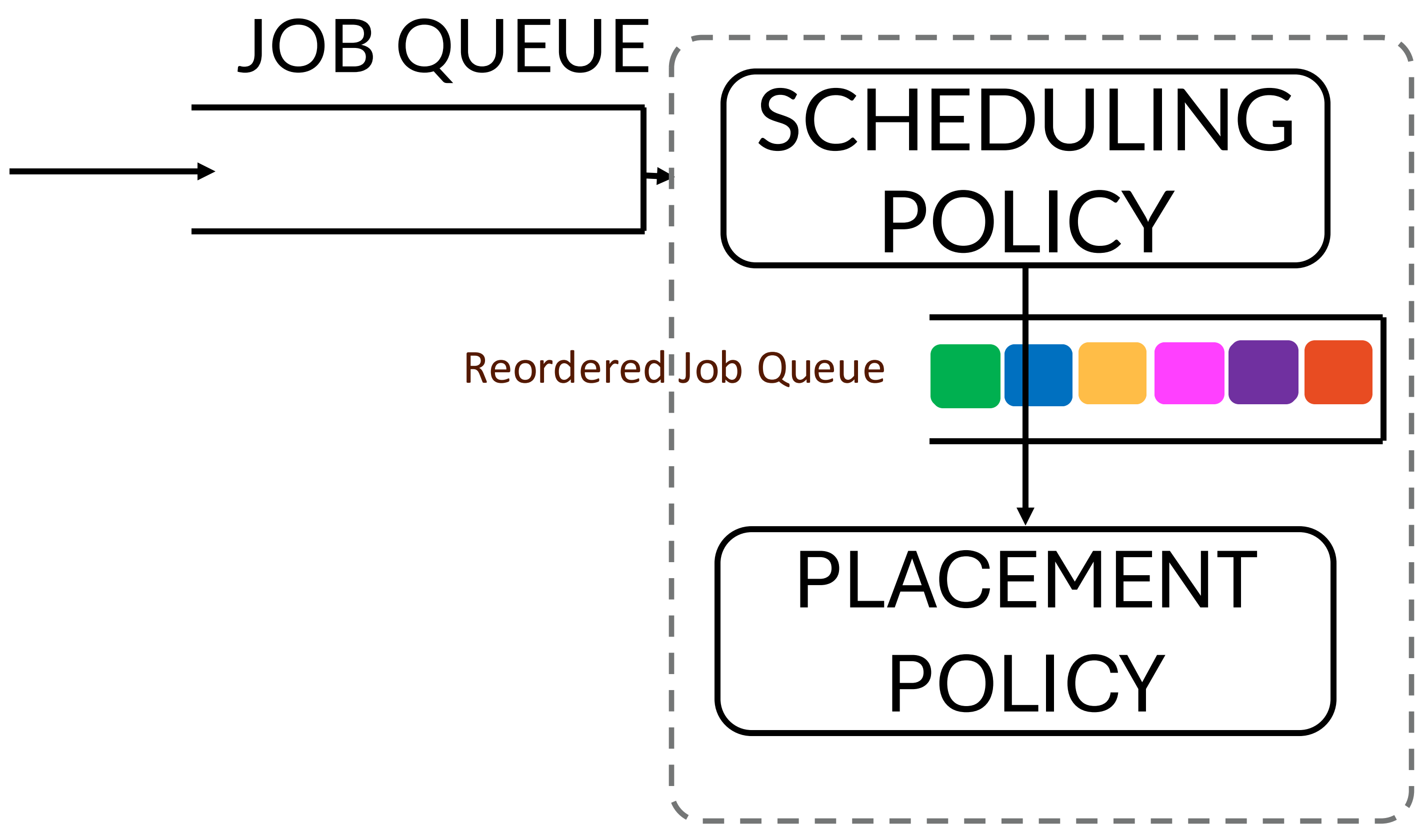
CLUSTER SCHEDULING



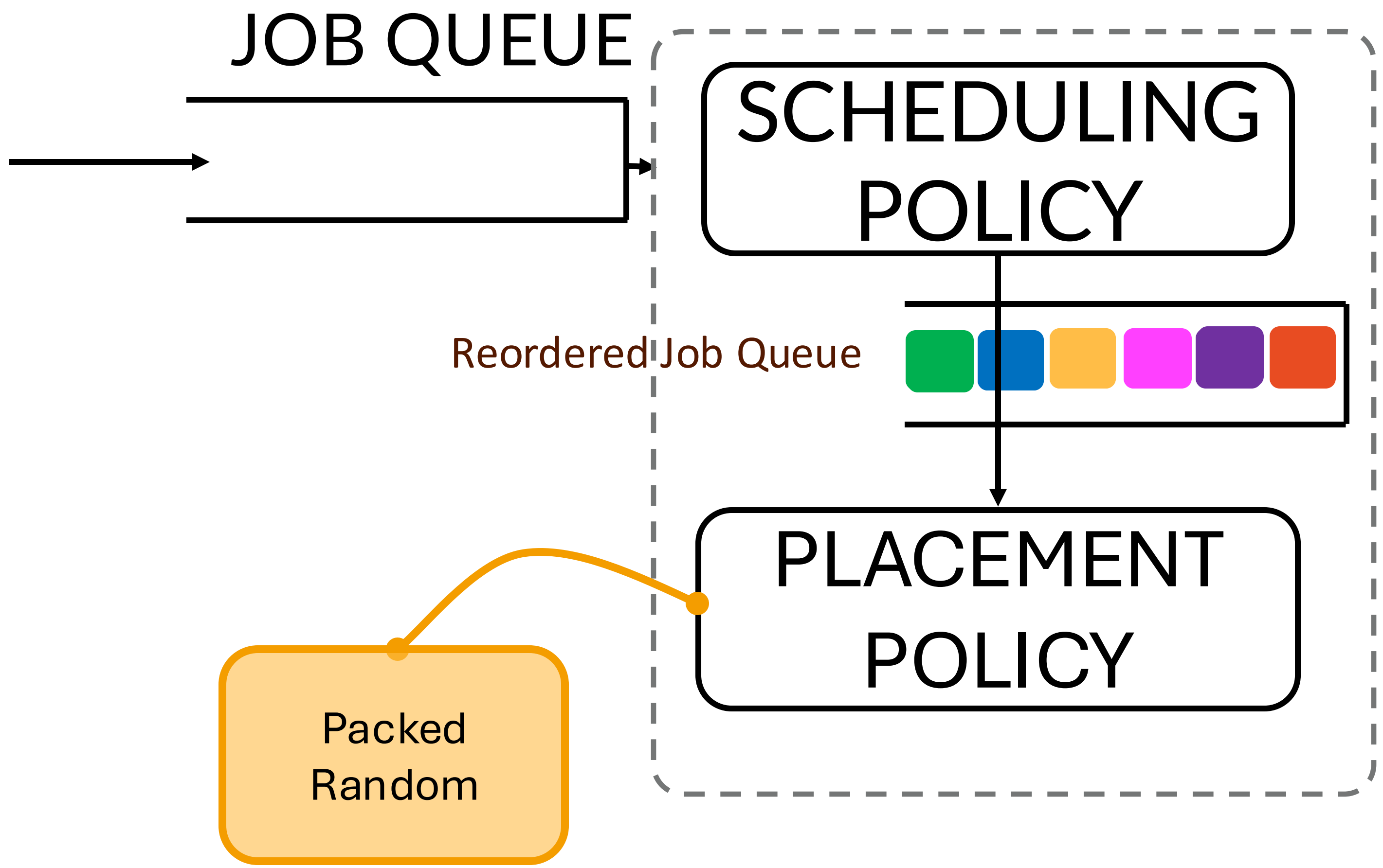
CLUSTER SCHEDULING



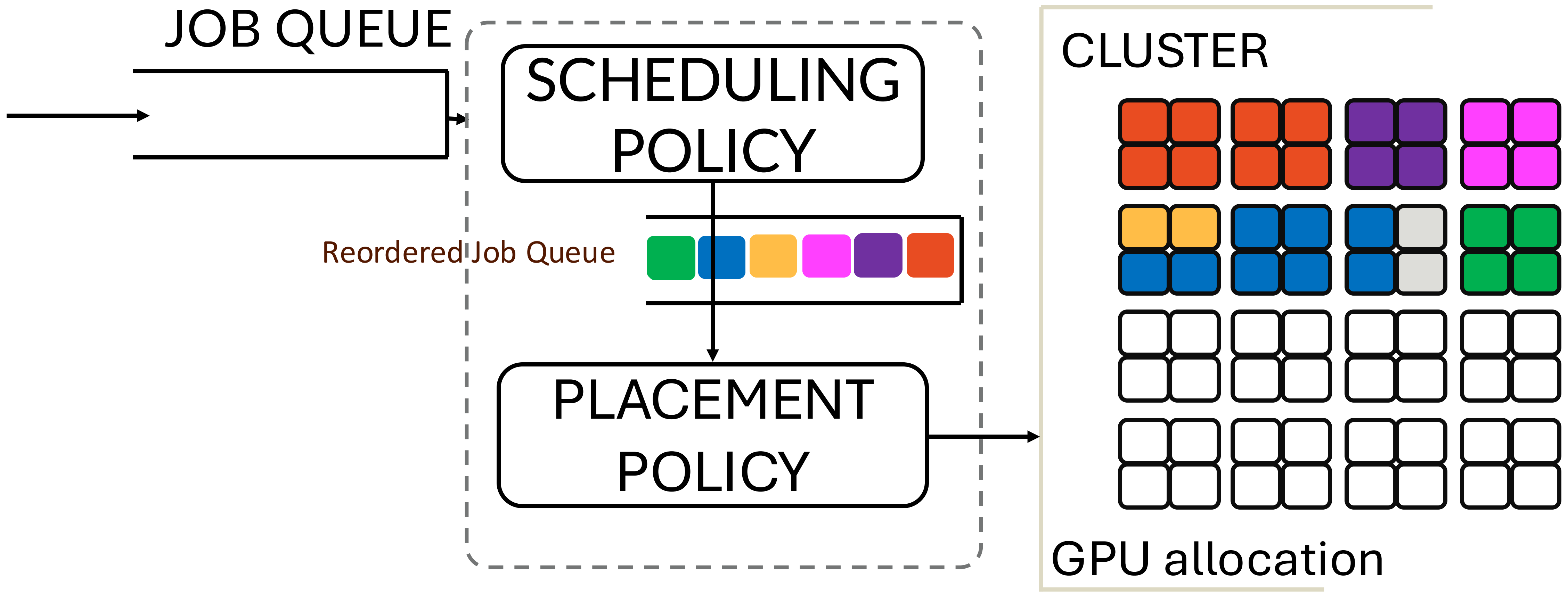
CLUSTER SCHEDULING



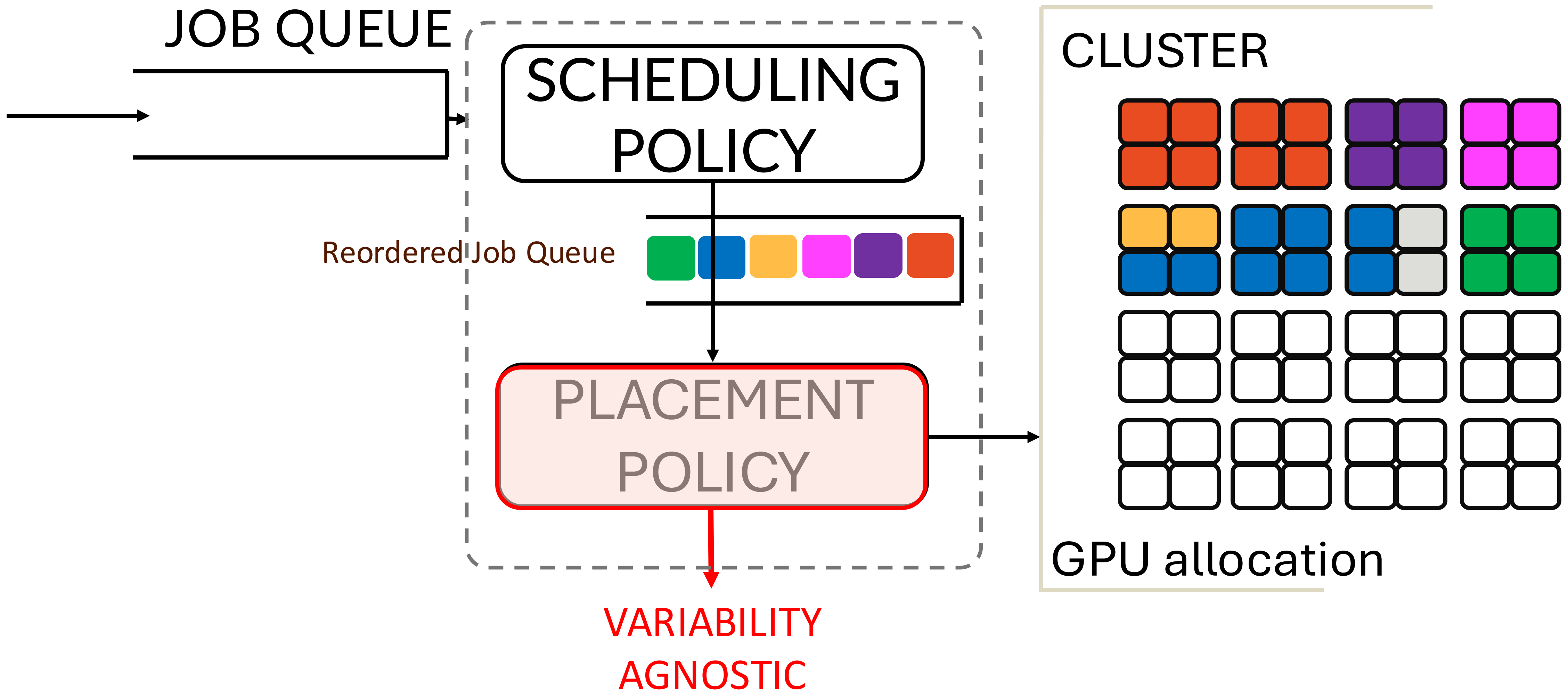
CLUSTER SCHEDULING



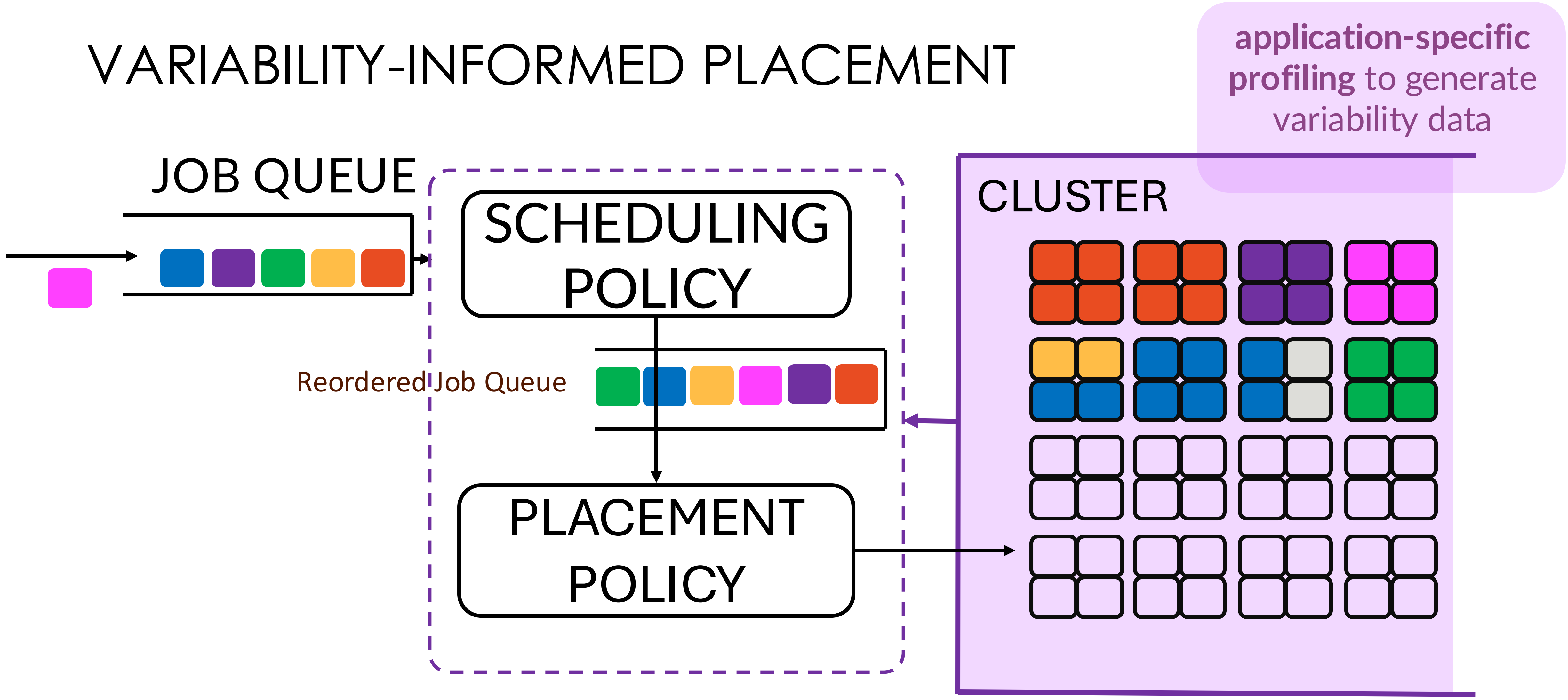
CLUSTER SCHEDULING



CLUSTER SCHEDULING

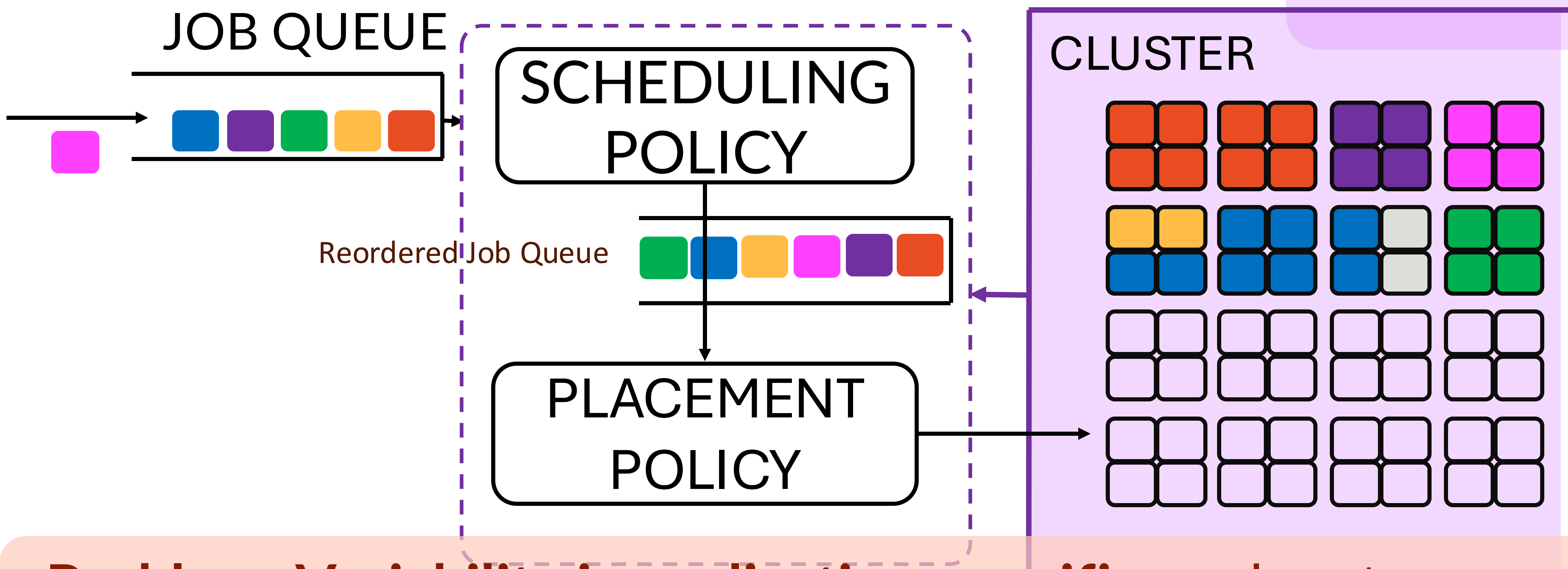


VARIABILITY-INFORMED PLACEMENT



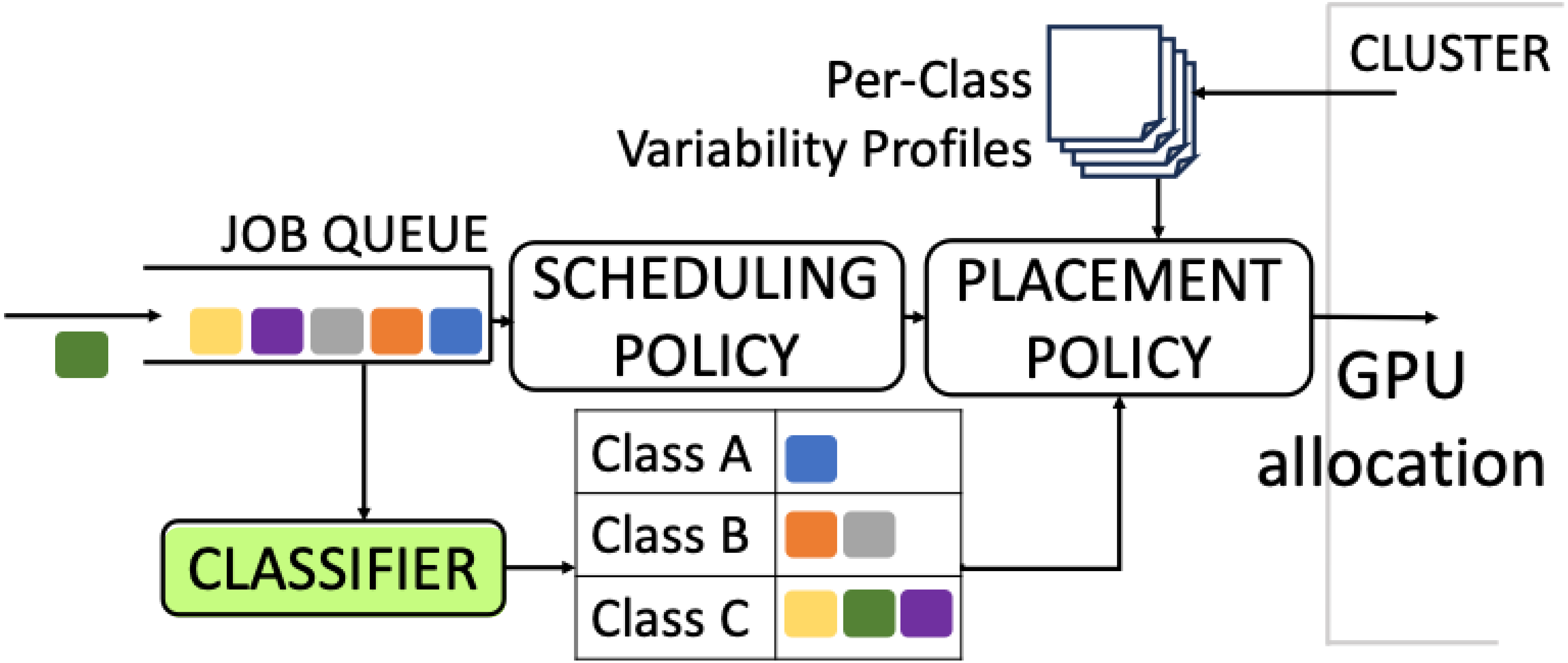
VARIABILITY-INFORMED PLACEMENT

application-specific **profiling** to generate variability data



Problem: Variability is application-specific, and systems run a wide variety of workloads

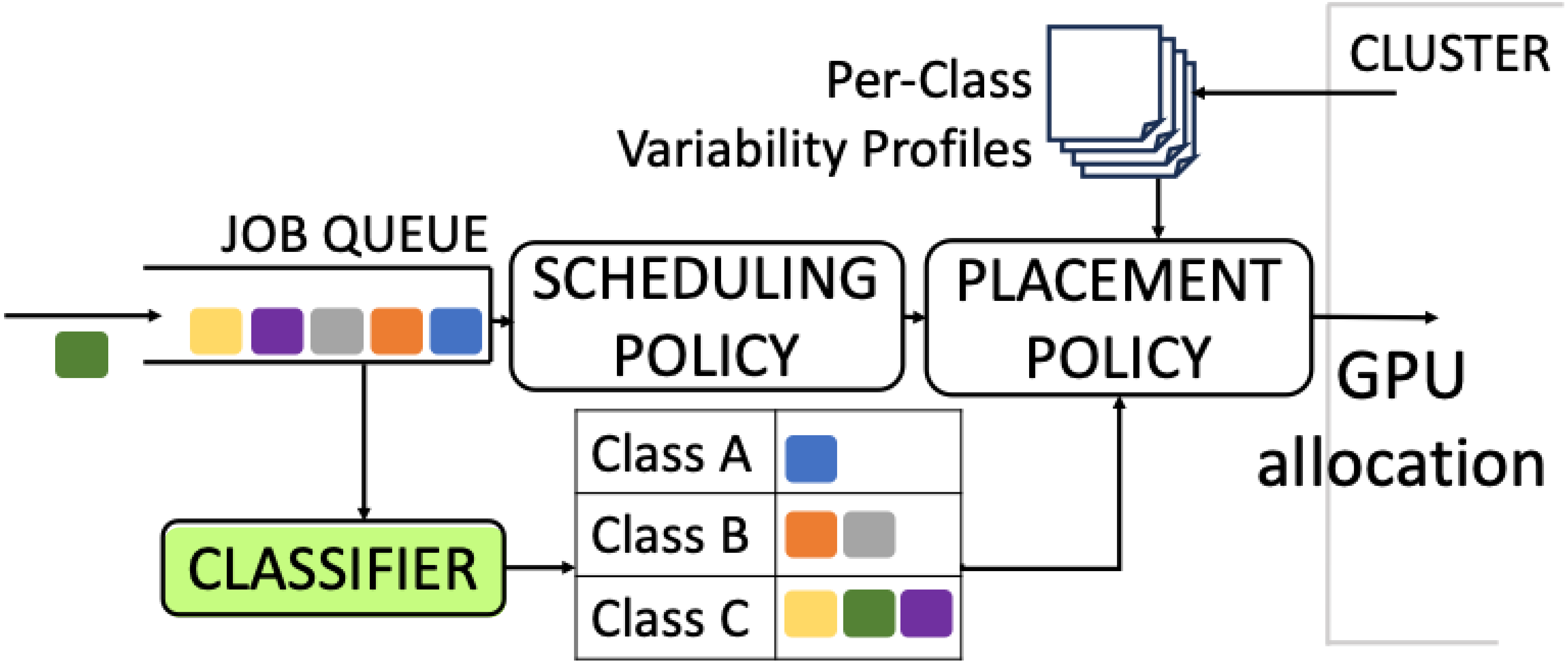
VARIABILITY-INFORMED PLACEMENT



Solution: have a finite number of classes
(compute bound ⟷memory bound)

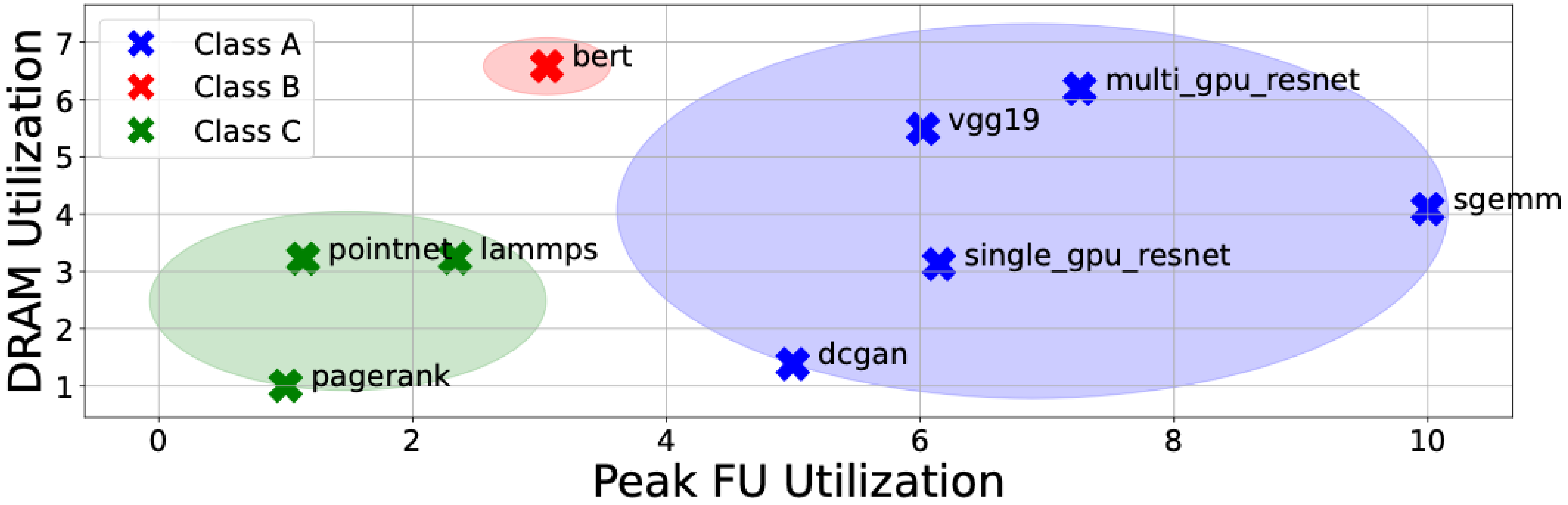
VARIABILITY-INFORMED PLACEMENT

use one variability profile per “class” as proxy for variability behavior of all applications in that class



Solution: have a finite number of classes (compute bound ⟷memory bound)

CLASSIFICATION LAYER



- Similar to Guerreiro et al. [Parallel Computing 2019], we used nsight compute to measure workloads' DRAM Throughput and Peak SM Throughput.
- **2D K-Means clustering** to produce ordered classes.
- Any new application can fall into one of these classes based on its DRAM and SM throughput.

SOLUTION 1: PM-FIRST PLACEMENT

SOLUTION 1: PM-FIRST PLACEMENT

- Give variability first-order precedence when assigning GPUs to jobs.

SOLUTION 1: PM-FIRST PLACEMENT

- Give variability first-order precedence when assigning GPUs to jobs.
- We associate a **PM-score** with each GPU
  - How fast/slow the GPU is relative to median GPU in the cluster (normalized performance)

$$P_i = \frac{t_i}{t_{median}}$$

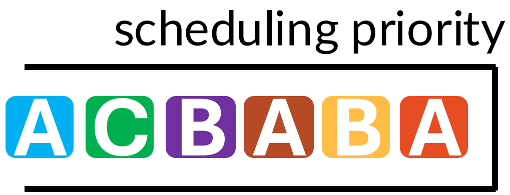
  - PM-score is class-specific.

SOLUTION 1: PM-FIRST PLACEMENT

- Give variability first-order precedence when assigning GPUs to jobs.
- We associate a **PM-score** with each GPU
  - How fast/slow the GPU is relative to median GPU in the cluster (normalized performance)

$$P_i = \frac{t_i}{t_{median}}$$

  - PM-score is class-specific.
- We also associate a **placement priority** for each job

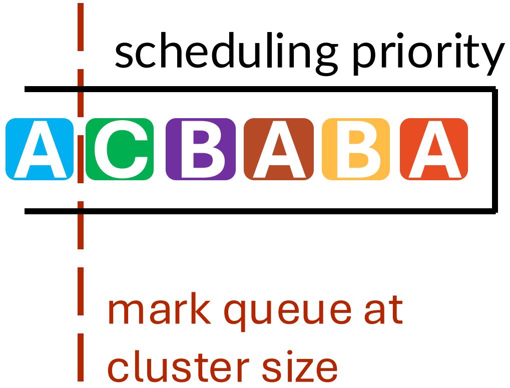


SOLUTION 1: PM-FIRST PLACEMENT

- Give variability first-order precedence when assigning GPUs to jobs.
- We associate a **PM-score** with each GPU
  - How fast/slow the GPU is relative to median GPU in the cluster (normalized performance)

$$P_i = \frac{t_i}{t_{median}}$$

  - PM-score is class-specific.
- We also associate a **placement priority** for each job

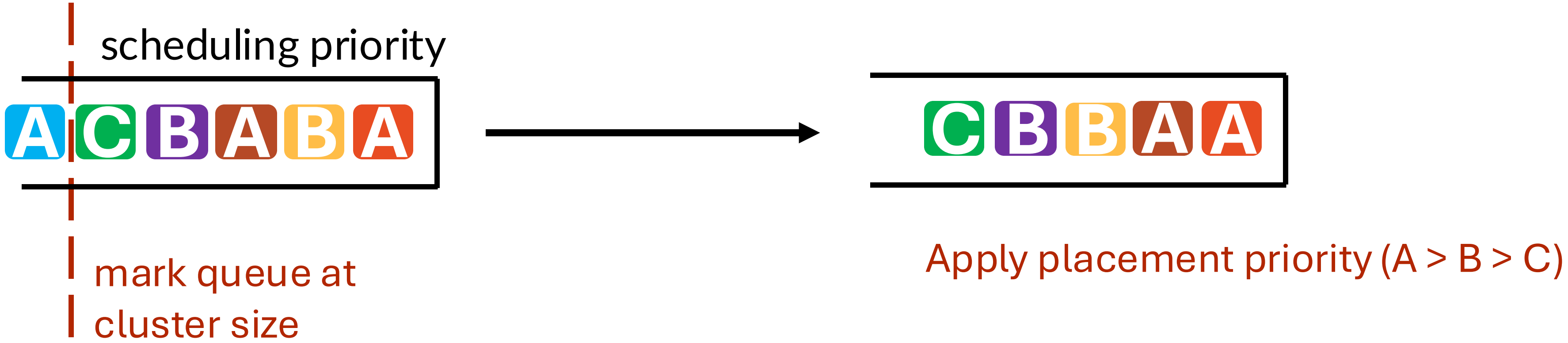


SOLUTION 1: PM-FIRST PLACEMENT

- Give variability first-order precedence when assigning GPUs to jobs.
- We associate a **PM-score** with each GPU
  - How fast/slow the GPU is relative to median GPU in the cluster (normalized performance)

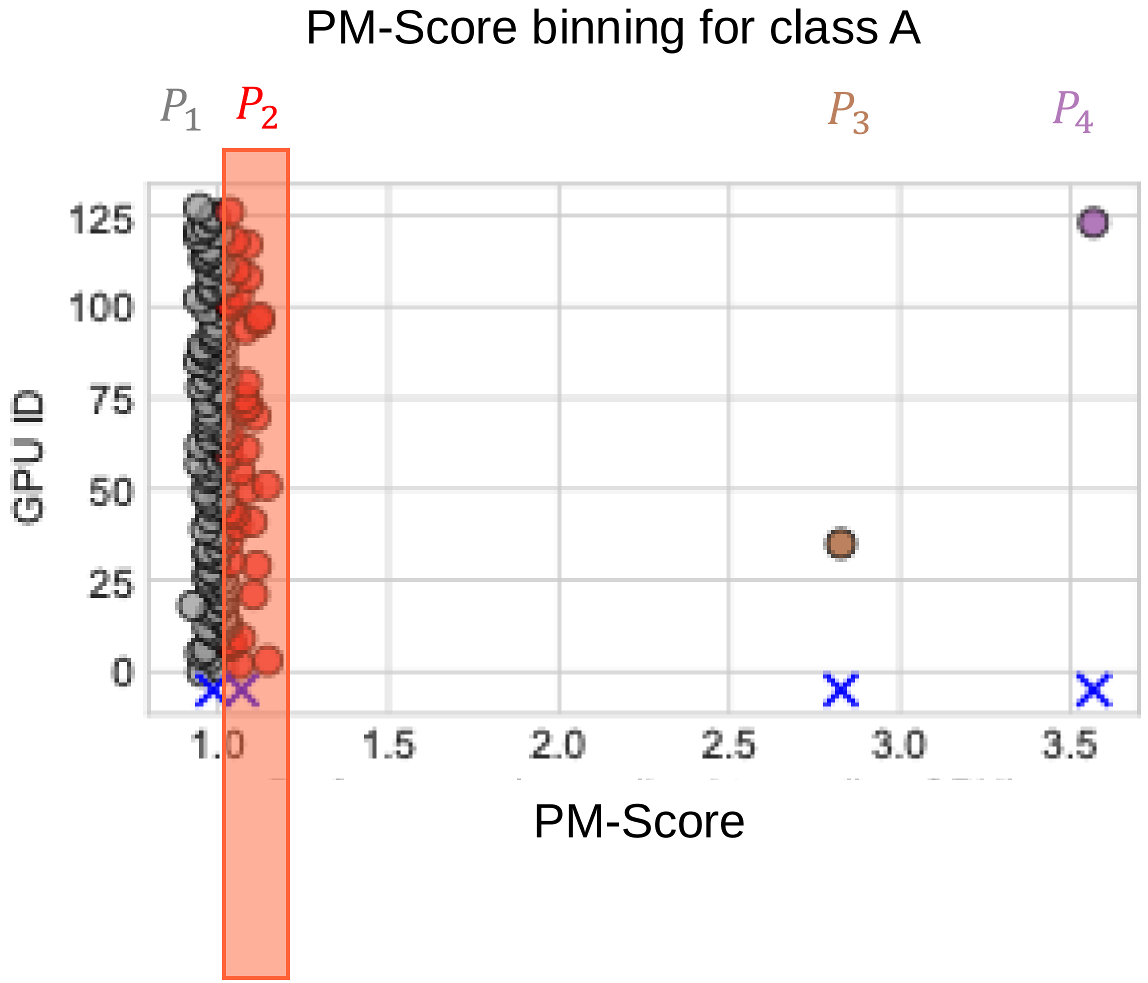
$$P_i = \frac{t_i}{t_{median}}$$

  - PM-score is class-specific.
- We also associate a **placement priority** for each job



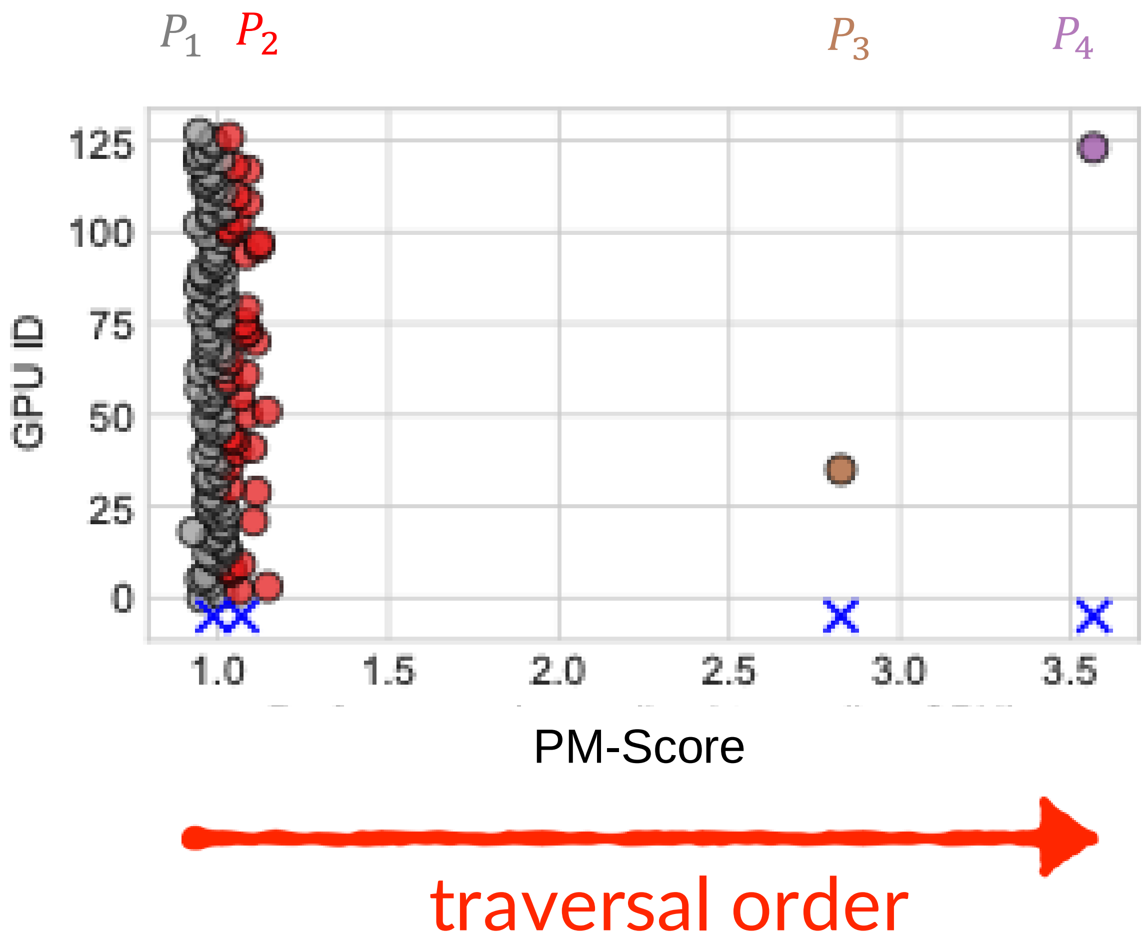
SOLUTION 1: PM-FIRST PLACEMENT

- For each class, we make **bins** of GPUs with similar **PM-Scores**.
- Why?
    - Fine-grained variability for large-scale systems is expensive
    - Memory-bound applications have little variability – one large pool of GPUs for them.



all GPUs of the red bin have similar variability

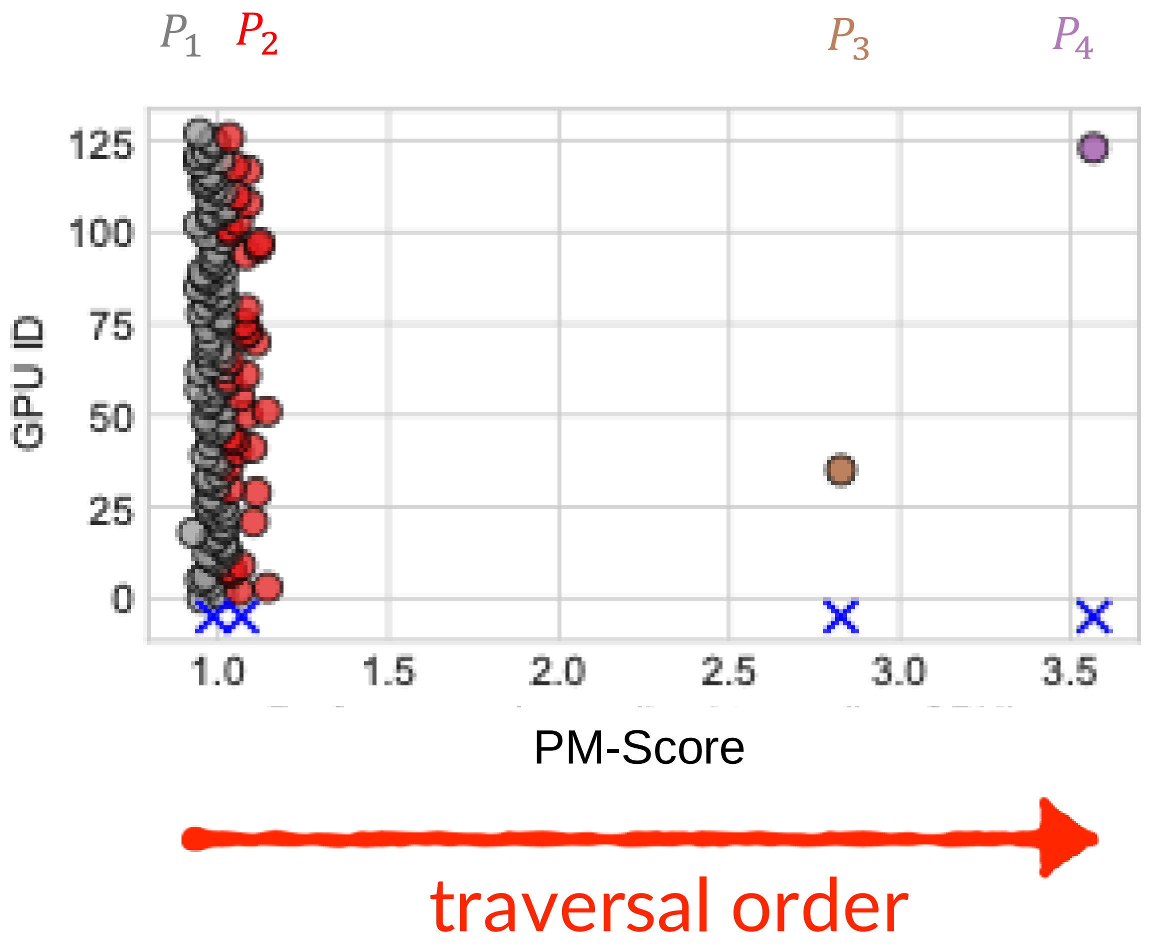
PM-FIRST PLACEMENT POLICY



PM-FIRST PLACEMENT POLICY

In each scheduling round,

- Get job queue from scheduling policy
- Mark queue at cluster size, and assign placement priority



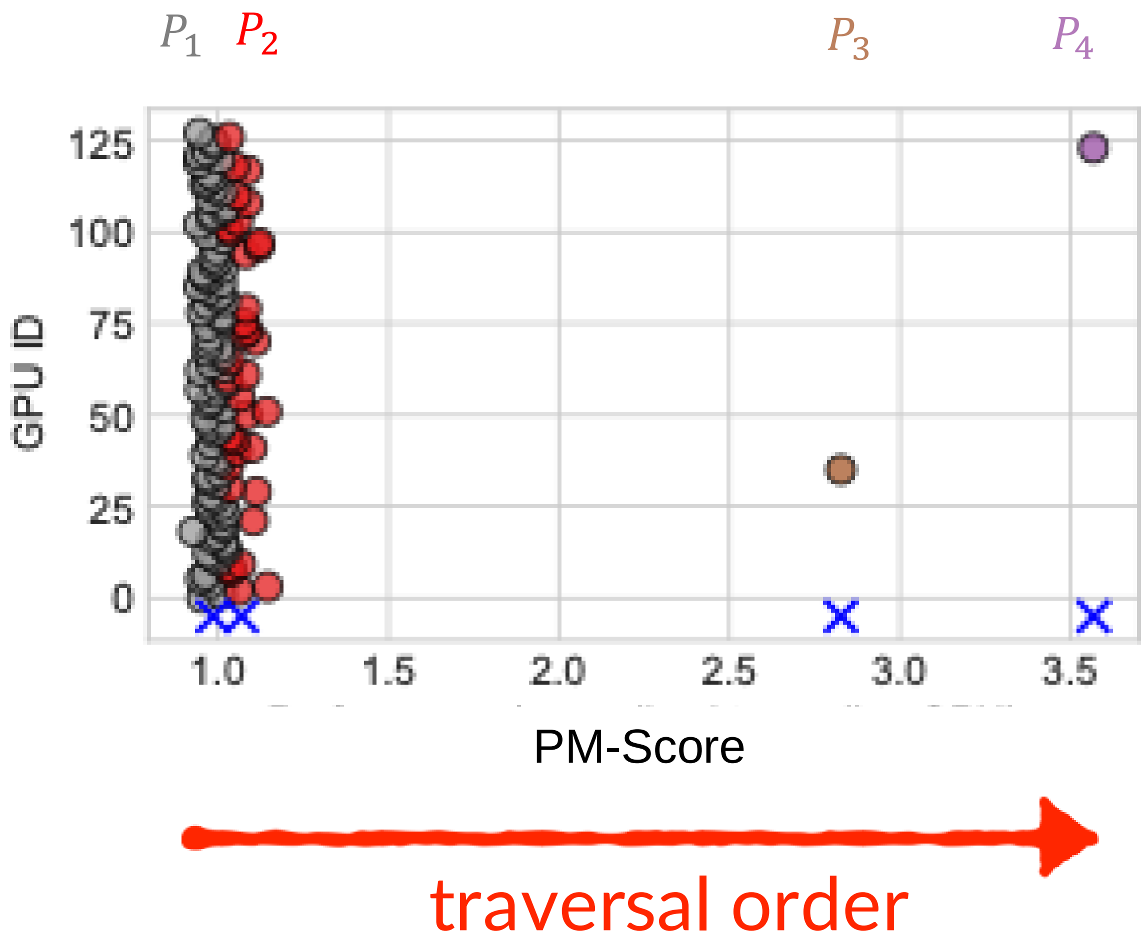
PM-FIRST PLACEMENT POLICY

In each scheduling round,

- Get job queue from scheduling policy
- Mark queue at cluster size, and assign placement priority

for each job j in order of placement priority:

- get job class and GPU demand N
- get freelist of GPUs available and **sort** by their **PM-Score** value (for job class)
- allocate first N GPUs from this sorted list.
- Mark GPUs in use (remove from freelist)



WHAT ABOUT LOCALITY?

- PM-First ignores communication overheads due to ineffective packing
- Some jobs are more sensitive to locality of GPUs than variability

WHAT ABOUT LOCALITY?

- PM-First ignores communication overheads due to ineffective packing
- Some jobs are more sensitive to locality of GPUs than variability

- Assume a simplified locality model:
  - Multi-GPU job incurs penalty $\boldsymbol{L_{across}}$ if it's allocation spills across nodes
  - No performance degradation if allocation is within a node
- Modified iteration time for a job running with set of GPUs G spread across > 1 node:

$$t_{iter} = L_{across} \times max_{g \in G}(P_g) \times t_{iter}^{orig}$$

where P_g is the PM-Score of GPU g

WHAT ABOUT LOCALITY?

- PM-First ignores communication overheads due to ineffective packing
- Some jobs are more sensitive to locality of GPUs than variability

- Assume a simplified locality model:
  - Multi-GPU job incurs penalty $\boldsymbol{L_{across}}$ if it's allocation spills across nodes
  - No performance degradation if allocation is within a node
- Modified iteration time for a job running with set of GPUs G spread across > 1 node:

$$t_{iter} = L_{across} \times max_{g \in G}(P_g) \times t_{iter}^{orig}$$

where P_g is the PM-Score of GPU g

$$\downarrow t_{iter} = \downarrow (L_{across} \times V) \times t_{iter}^{orig}$$

LV-PRODUCT

$$\min LV\text{−product} = \min \left(L_{across} \times V\right)$$

Slowdown due to locality

Slowdown due to variability

Lower V values are better

Example ClassA	$V_1 = 0.89$	$V_2 = 0.94$	$V_3 = 1.06$	$V_4 = 2.55$
Within Node $L_{within} = 1$	**0.89**	**0.94**	**1.06**	**2.55**
Across node $L_{across} = 1.5$	**1.35**	**1.41**	**1.59**	**3.82**

LV-PRODUCT

$$\min LV\text{-product} = \min\left(L_{across} \times V\right)$$

Slowdown due to locality

Slowdown due to variability

Lower V values are better

Example ClassA	$V_1 = 0.89$	$V_2 = 0.94$	$V_3 = 1.06$	$V_4 = 2.55$
Within Node $L_{within} = 1$	0.89	0.94	1.06	2.55
Across node $L_{across} = 1.5$	1.35	1.41	1.59	3.82

PAL traversal order

LV-PRODUCT

$$\min LV\text{−product} = \min\left(L_{across} \times V\right)$$

Slowdown due to locality

Slowdown due to variability

Example ClassA	$V_1 = 0.89$	$V_2 = 0.94$	$V_3 = 1.06$	$V_4 = 2.55$
Within Node $L_{within} = 1$	**0.89**	**0.94**	**1.06**	**2.55**
Across node $L_{across} = 1.5$	**1.35**	**1.41**	**1.59**	**3.82**

PM-First traversal order

OUTLINE

- Introduction
- Cluster Scheduling
- Design
- **Methodology**
- Evaluation
- Conclusion

SYSTEM

- **Blox (open-source modular toolkit)**
- **Physical Cluster**

TEXAS ADVANCED COMPUTING CENTER

TACC Frontera
360 GPUs
mineral oil cooled
NVIDIA Quadro **RTX5000** GPUs
4x GPUs per node
16GB memory per GPU

- **Testbed**
  - 16 node (64 GPUs)

SYSTEM

- **Blox (open-source modular toolkit)**
- **Physical Cluster**

TACC Frontera
360 GPUs
mineral oil cooled
NVIDIA Quadro **RTX5000** GPUs
4x GPUs per node
16GB memory per GPU

- **Testbed**
  - 16 node (64 GPUs)

Experiment(s)	Cluster Size	Workload Trace	Paper Section
Cluster Testbed Evaluation	64	Sia-Cluster*	V-A
Baseline Simulation Varying Locality Penalty	64	Sia-Philly*	V-B
Varying Job Load Varying Scheduling Policy	256	Synergy†	V-C

* [Subramanya et al. SOSP'23]
† [Mohan et al. OSDI'22]

BASELINE PLACEMENT POLICIES

BASELINE PLACEMENT POLICIES

Locality

Packed

Random

BASELINE PLACEMENT POLICIES

Sticky

Non-Sticky

Reallocation

METHODOLOGY: PLACEMENT POLICIES

Locality

Packed-Sticky

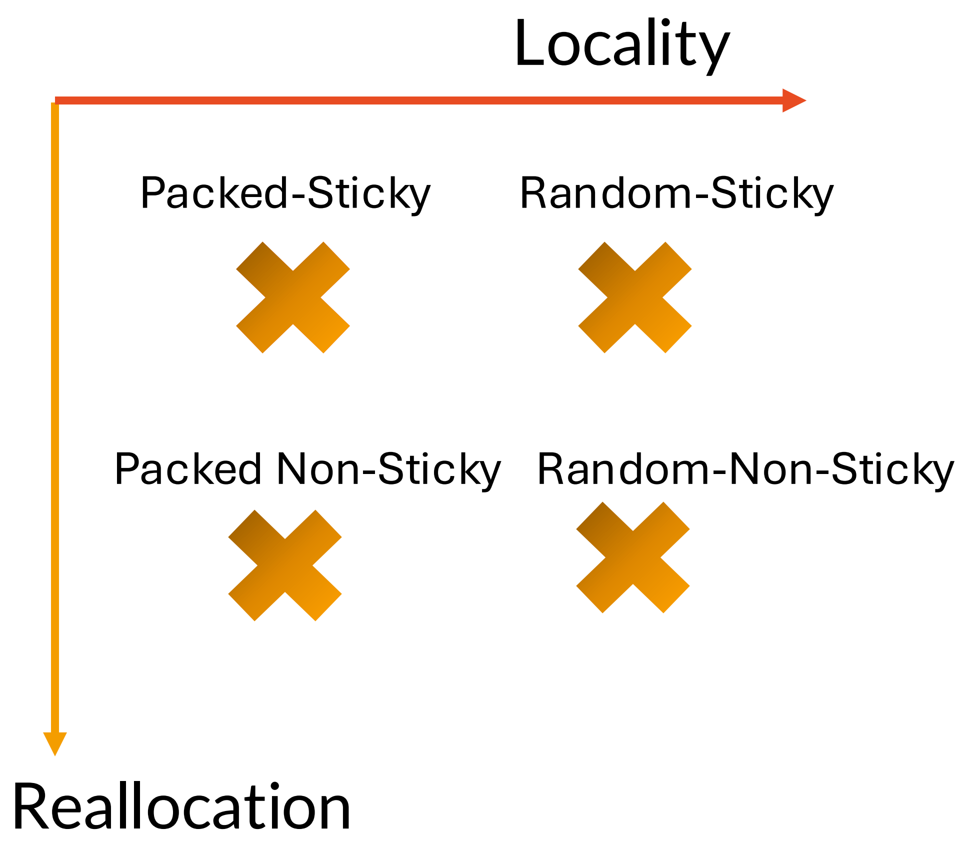
Random-Sticky

Packed Non-Sticky

Random-Non-Sticky

Reallocation

METHODOLOGY: PLACEMENT POLICIES

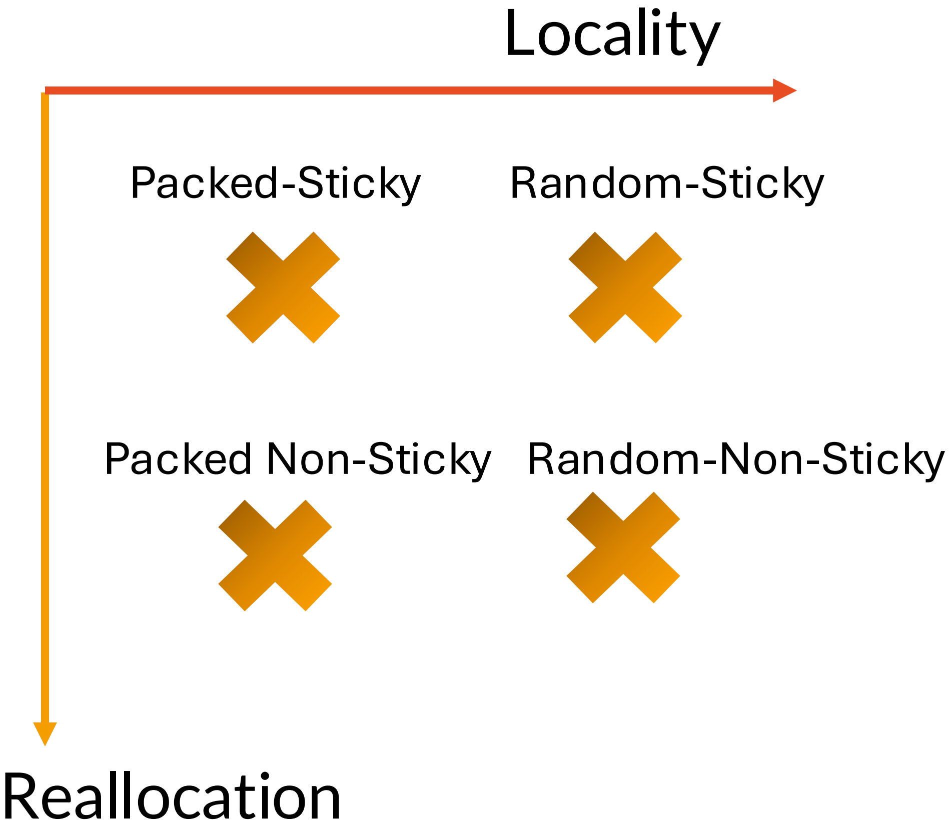


Baseline placement policies

- **Tiresias** [Gu et al. NSDI'19]
  - Packed Sticky
- **Gandiva** [Xia et al. OSDI'18]
  - Packed Non-sticky
- **Random-Sticky**
- **Random-Non-Sticky**

HotGauge [Hankin et al. IISWC'21]

METHODOLOGY: PLACEMENT POLICIES



Baseline placement policies

- **Tiresias** [Gu et al. NSDI'19]
  - Packed Sticky
- **Gandiva** [Xia et al. OSDI'18]
  - Packed Non-sticky
- **Random-Sticky**
- **Random-Non-Sticky**

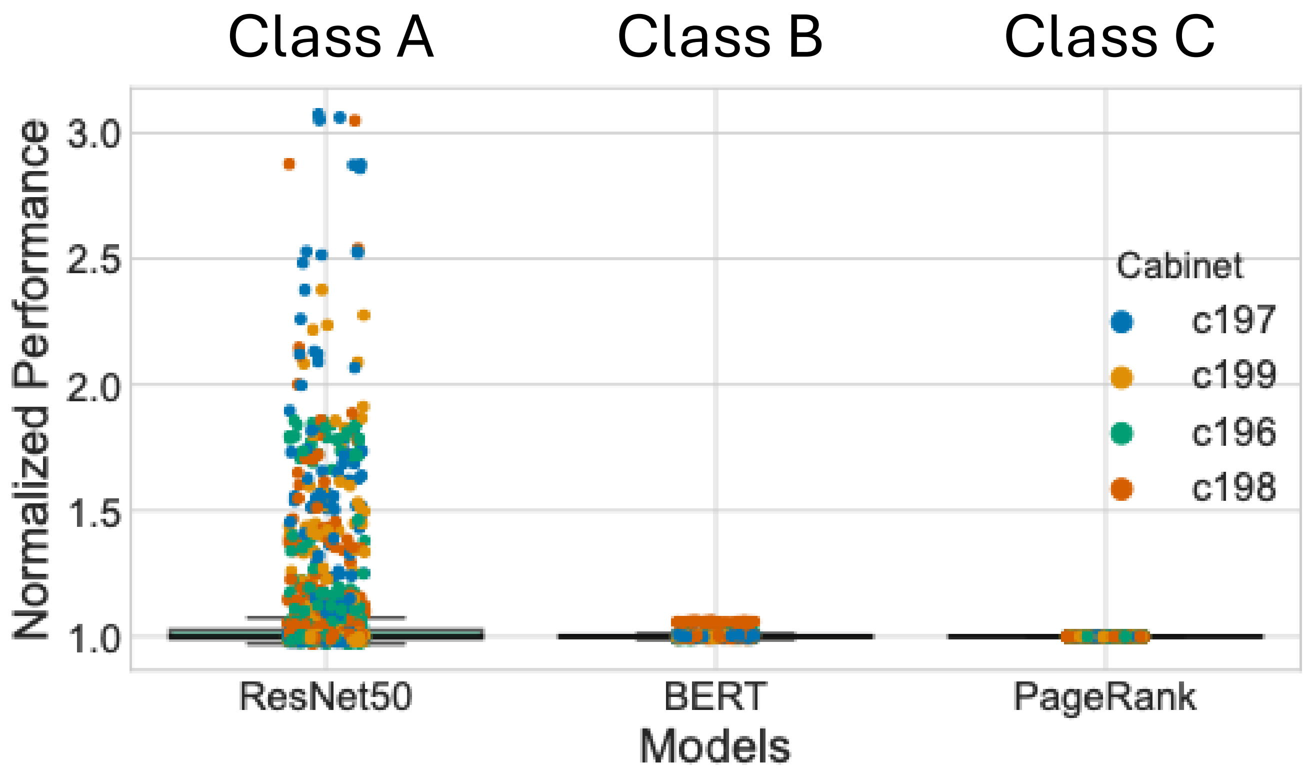
HotGauge [Hankin et al. IISWC'21]

- **PM-First and PAL**
  - Both non-sticky

APPLICATIONS: PHYSICAL CLUSTER EXPERIMENTS

Task	Model	Dataset	Batch Size	Class
Image	PointNet	ShapeNet	32	C
Image	vgg19	ImageNet2012	32	A
Vision	DCGAN	LSUN	128	A
Language	BERT	WikiText	64	B
Image	ResNet-50	Imagenet2012	32	A
Language	GPT2	Wikitext	128	B

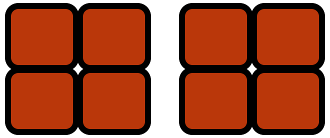
PM-SCORES: VARIABILITY PROFILES



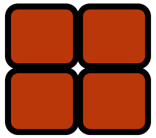
TACC Frontera

$$P_i = \frac{t_i}{t_{median}}$$

LOCALITY PENALTY

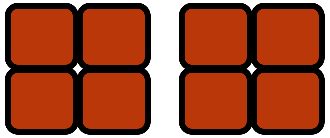


8 GPU ResNet-50 job
Batch Size 128

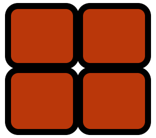


4 GPU ResNet-50 job
Batch Size 64

LOCALITY PENALTY



8 GPU ResNet-50 job
Batch Size 128



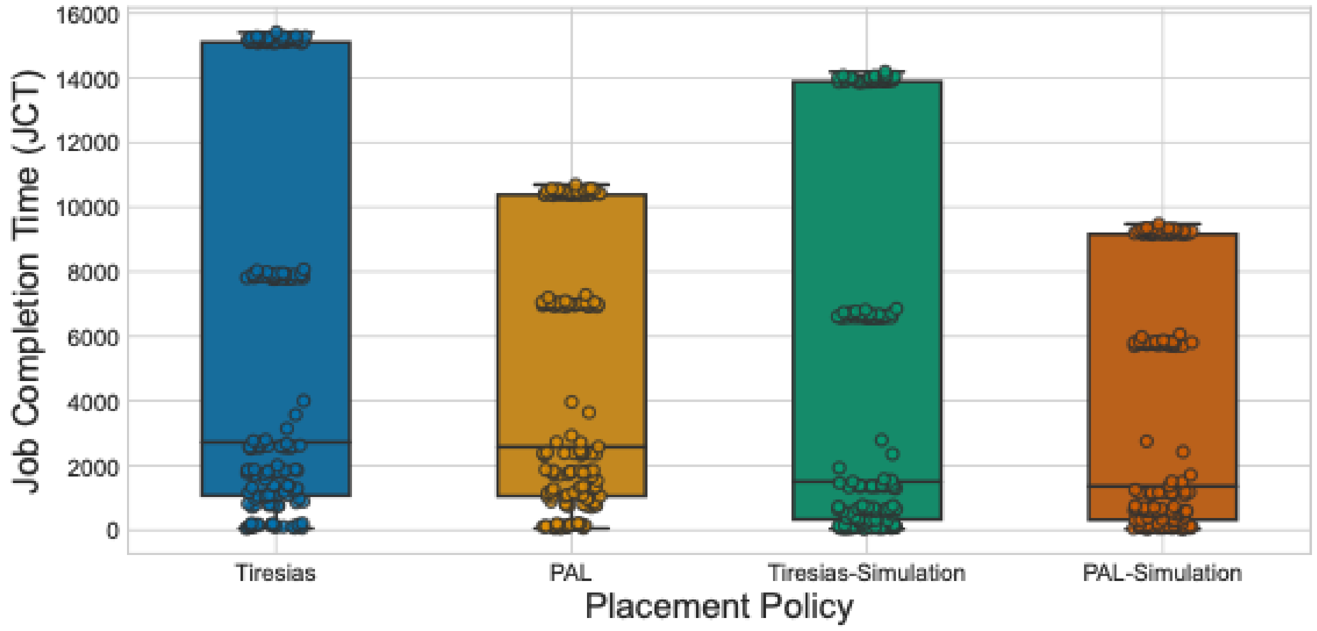
4 GPU ResNet-50 job
Batch Size 64

$$\frac{t_{iter,avg}}{t_{iter,avg}} = L_{across}$$

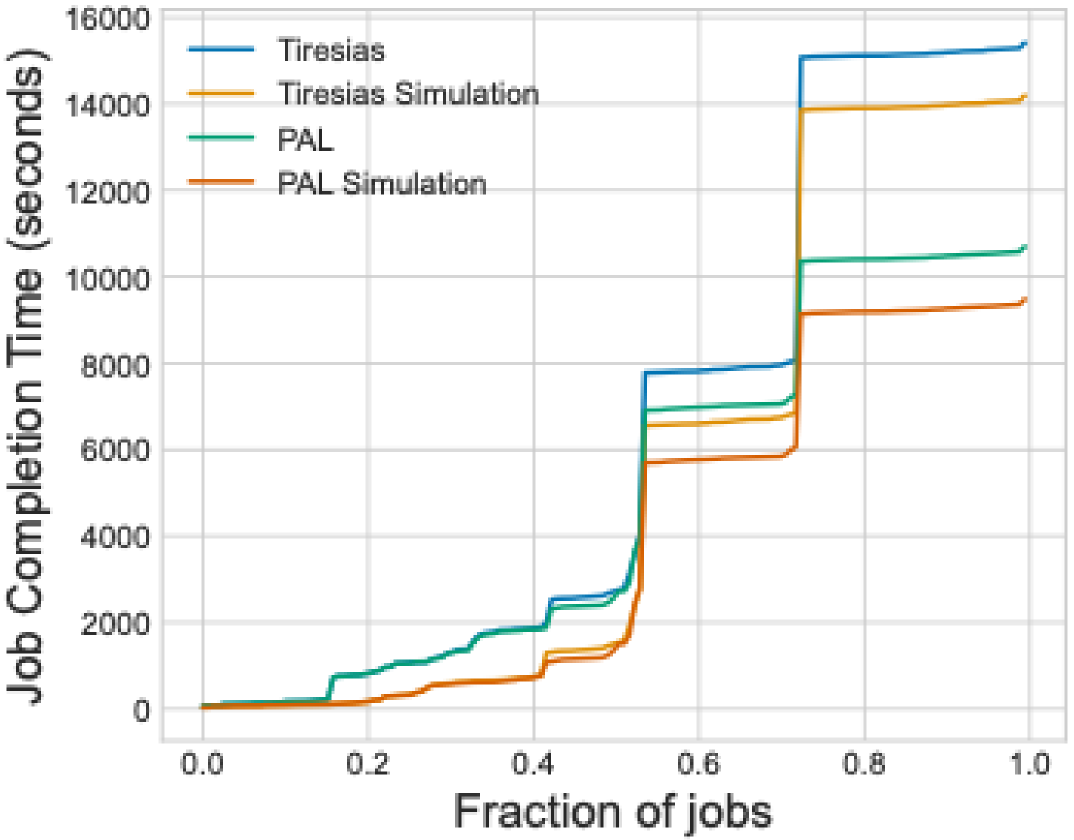
OUTLINE

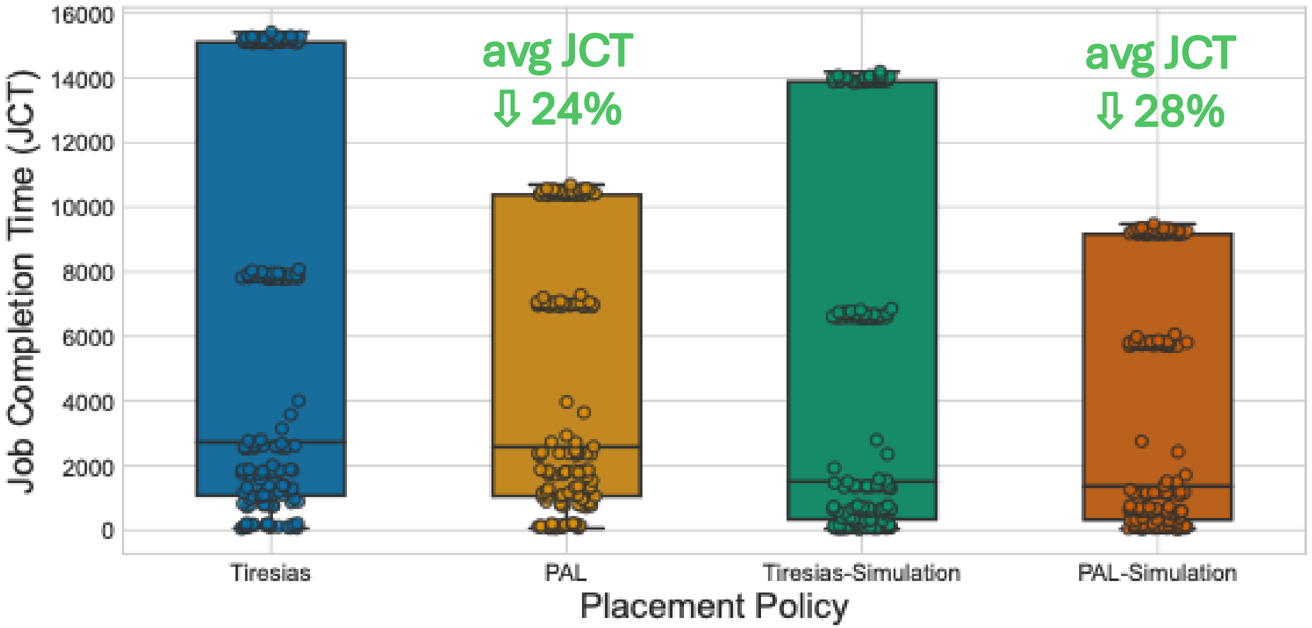
- Introduction
- Cluster Scheduling
- Design
- Methodology
- **Evaluation**
- Conclusion

PHYSICAL CLUSTER EXPERIMENT: 64 GPUs

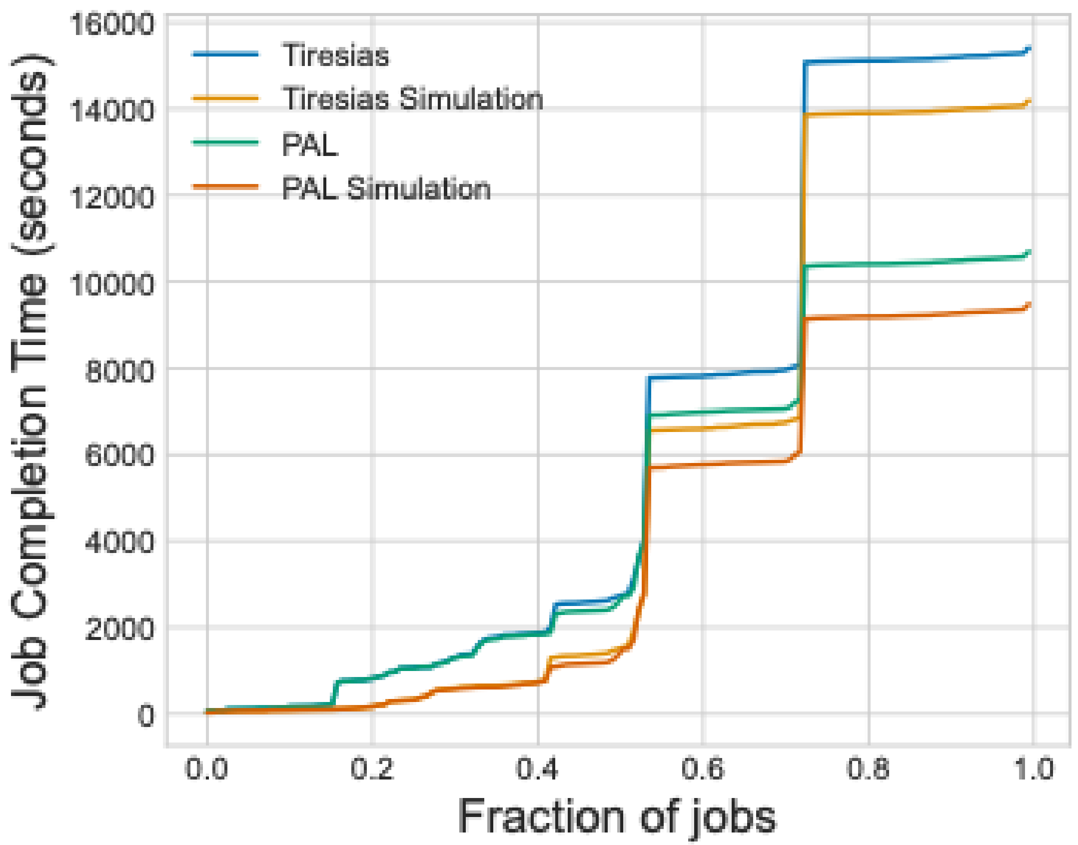


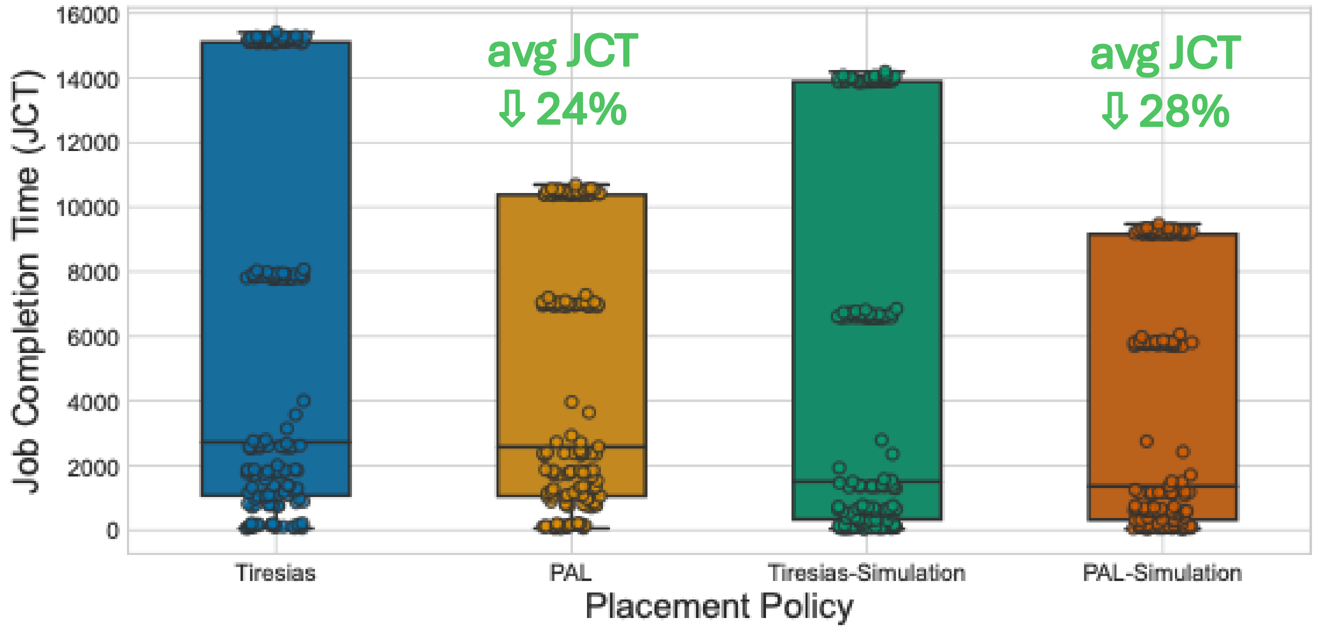
PHYSICAL CLUSTER EXPERIMENT: 64 GPUs





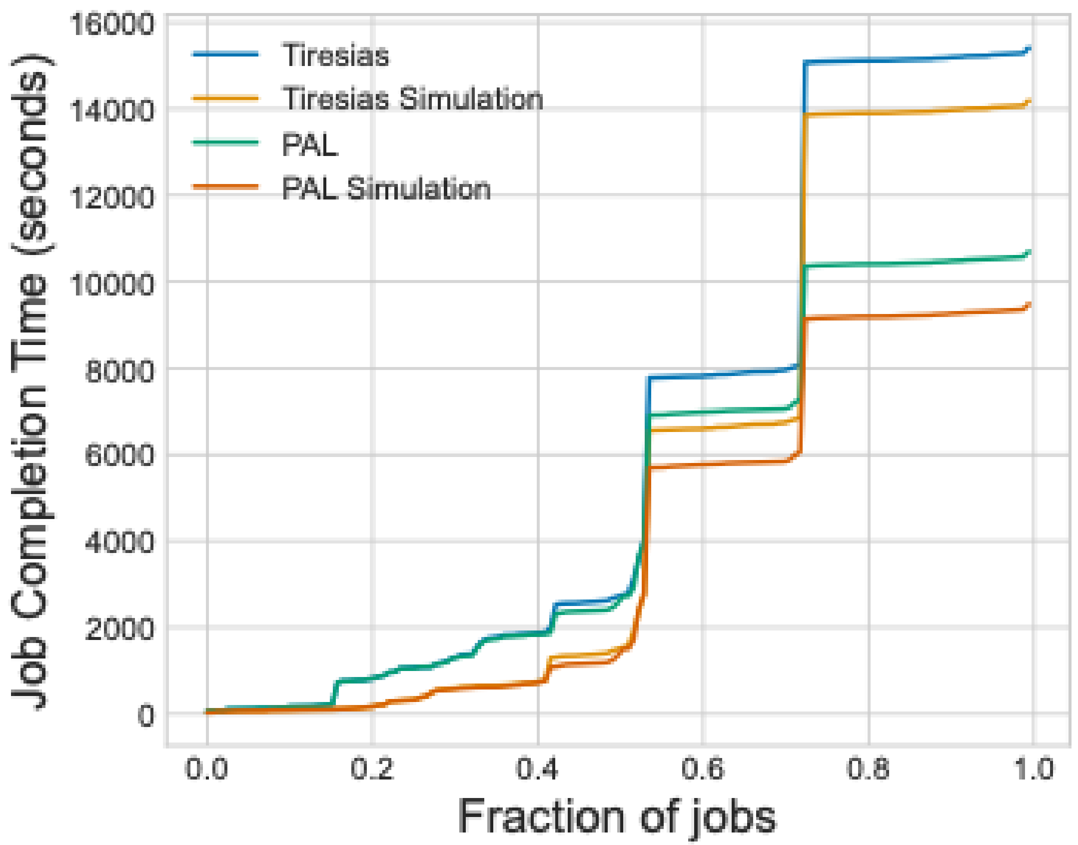
PHYSICAL CLUSTER EXPERIMENT: 64 GPUs

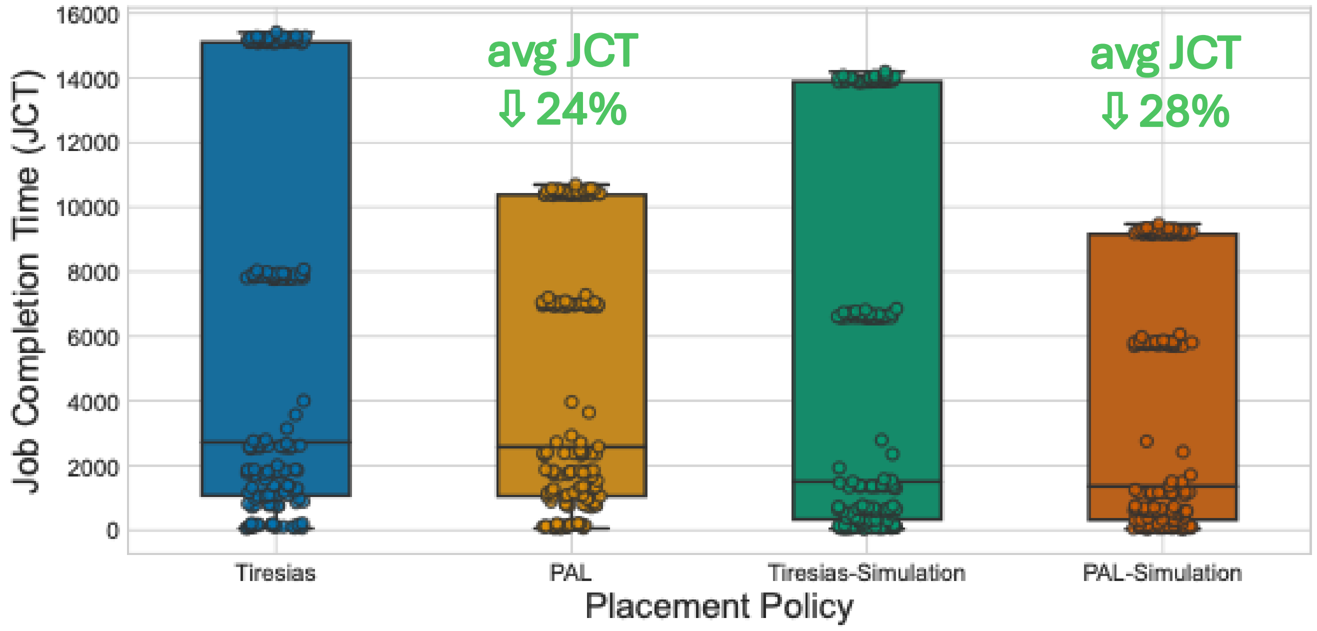




cluster-to-sim diff
11-14%

PHYSICAL CLUSTER EXPERIMENT: 64 GPUs

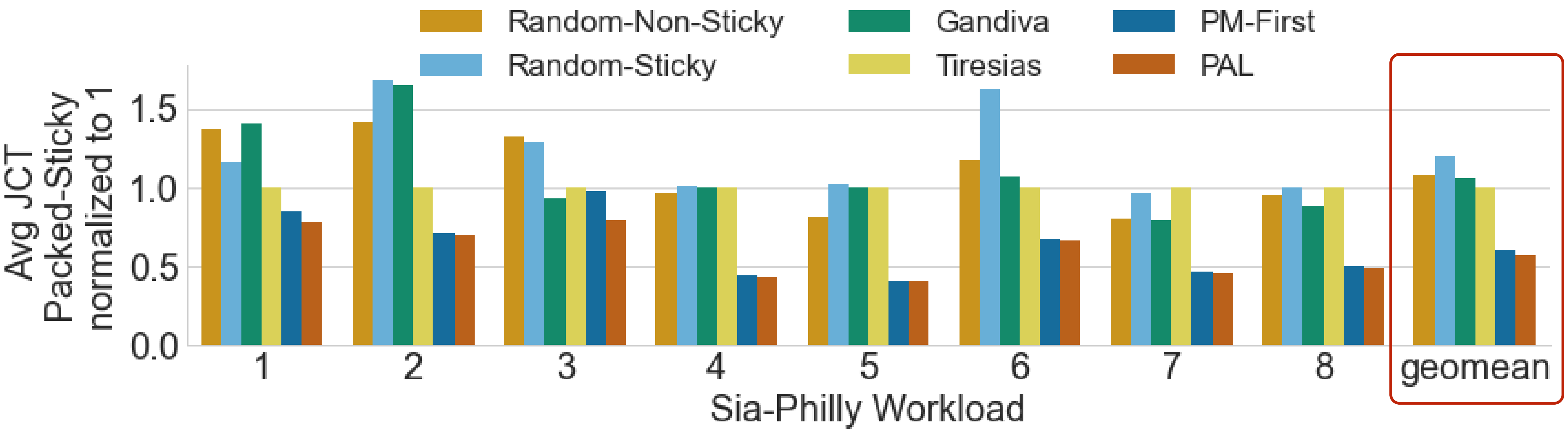




First-order trends are accurate and correlate well in simulation

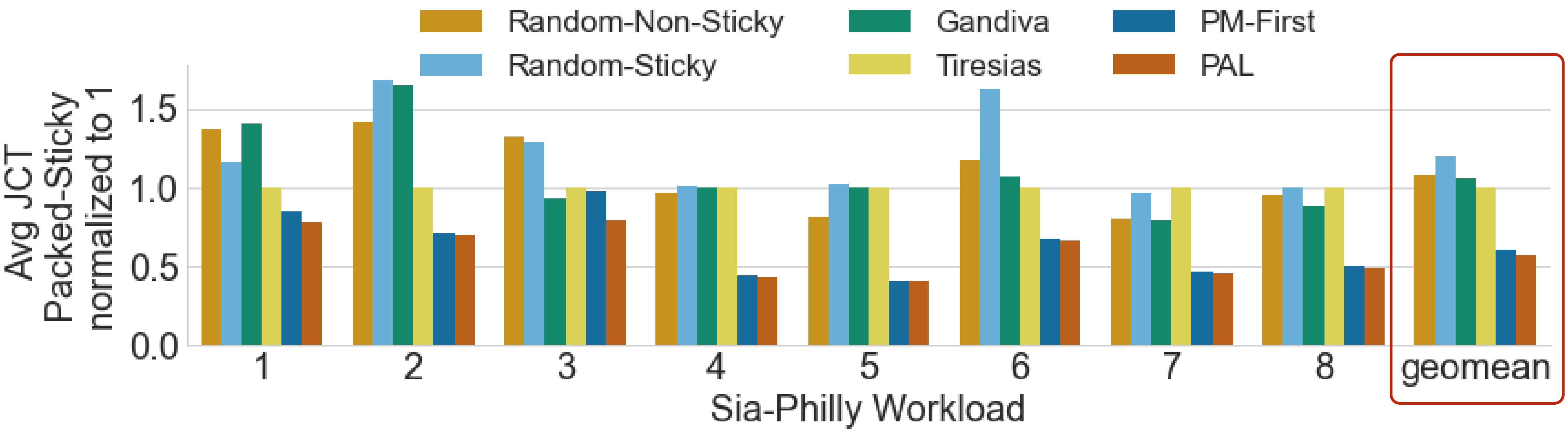
SIA-PHILLY SIMULATIONS

Scheduler: **FIFO**
# of GPUs: **64**
Job trace: **Sia-Philly trace (160 jobs, requesting up to 48 GPUs)**



SIA-PHILLY SIMULATIONS

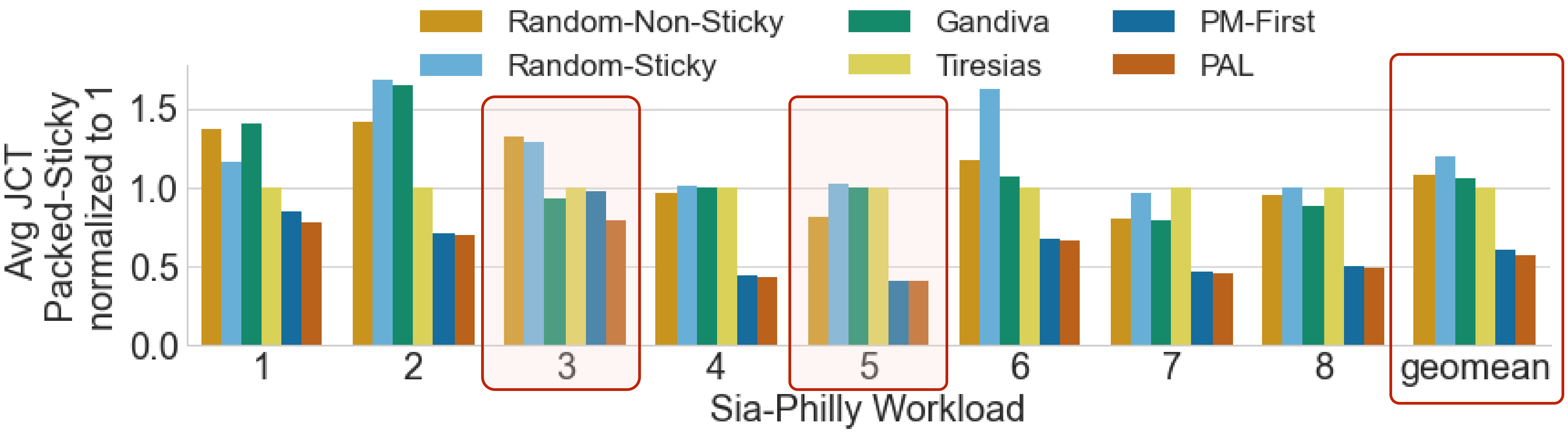
Scheduler: **FIFO**
# of GPUs: **64**
Job trace: **Sia-Philly trace (160 jobs, requesting up to 48 GPUs)**



PM-First improves geomean JCT by 40%, PAL further improves JCTs by considering locality in addition to variability

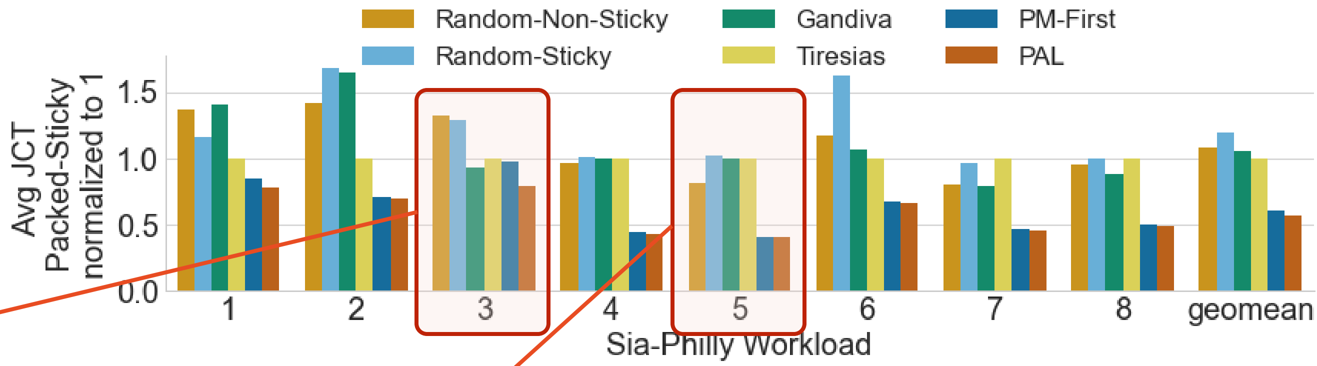
SIA-PHILLY SIMULATIONS

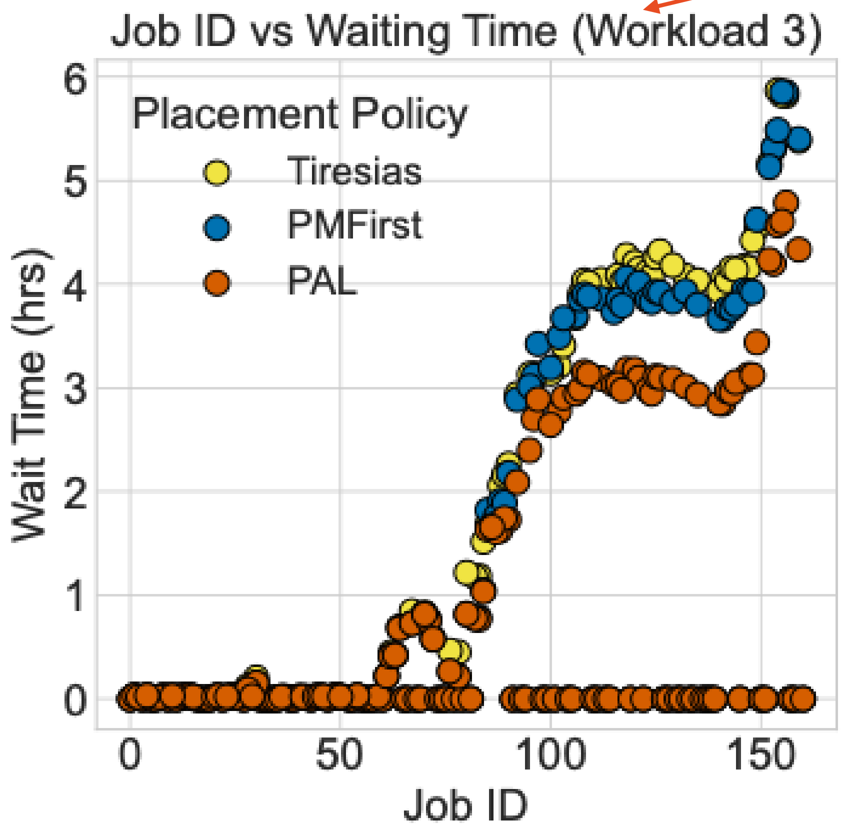
Scheduler: **FIFO**
# of GPUs: **64**
Job trace: **Sia-Philly trace (160 jobs, requesting up to 48 GPUs)**

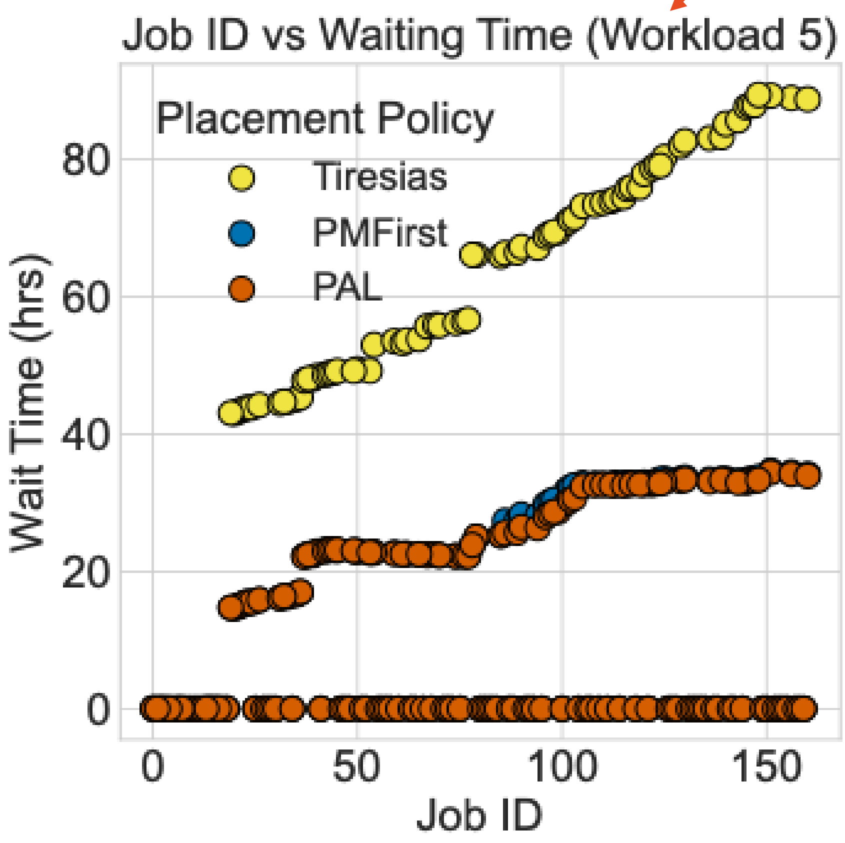


PM-First improves geomean JCT by 40%, PAL further improves JCTs by considering locality in addition to variability

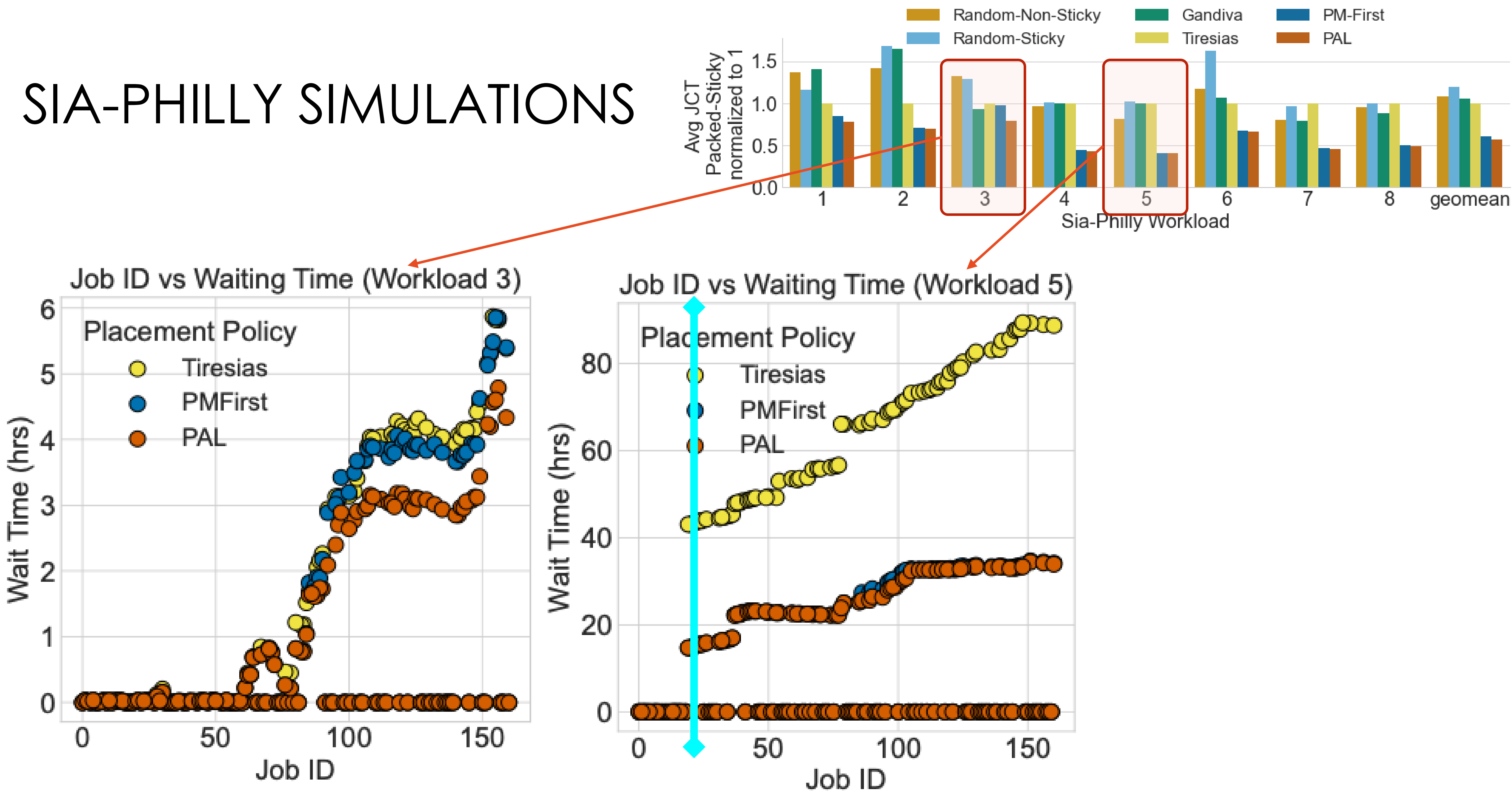
SIA-PHILLY SIMULATIONS



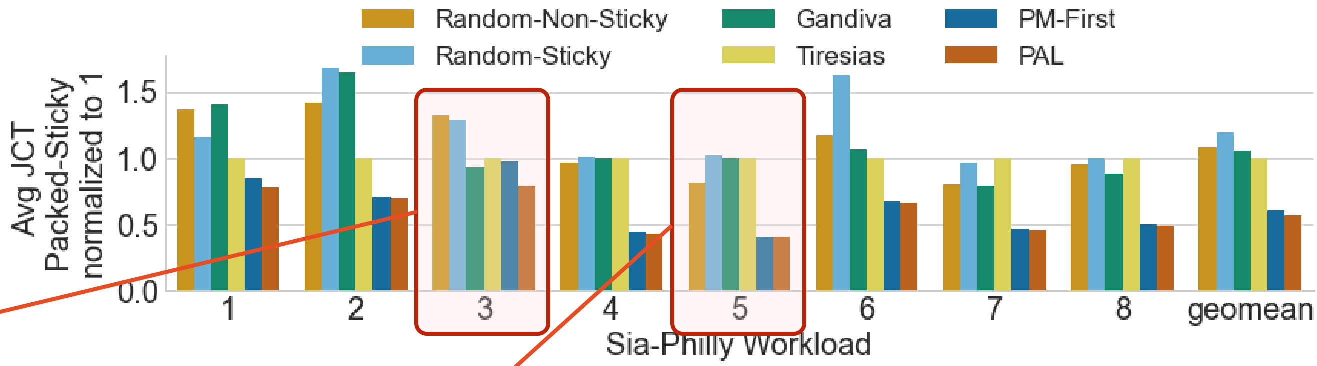


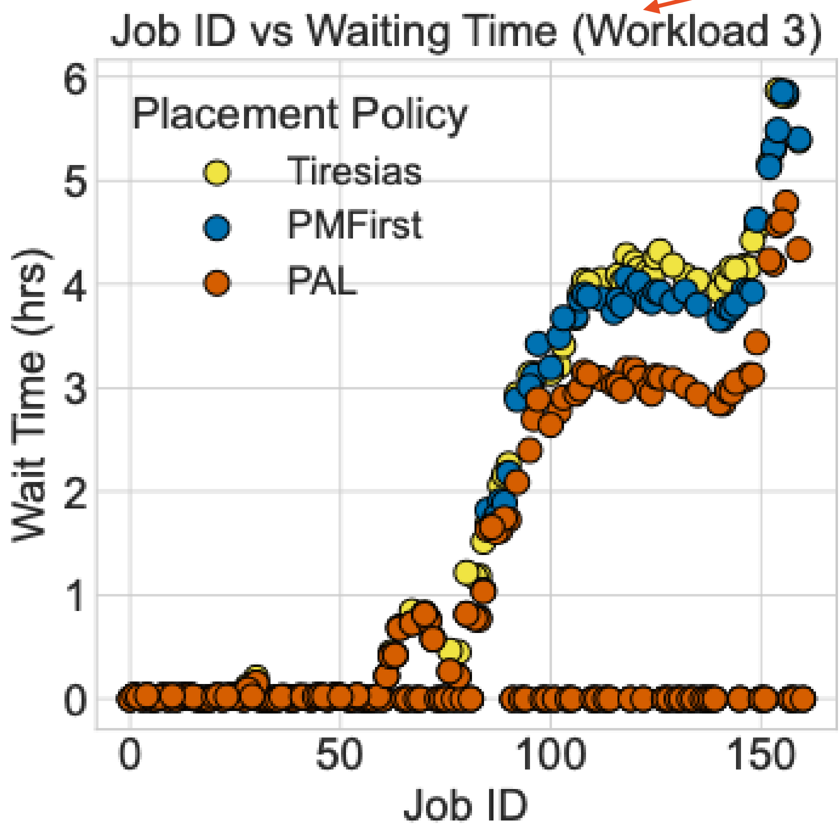


SIA-PHILLY SIMULATIONS



SIA-PHILLY SIMULATIONS

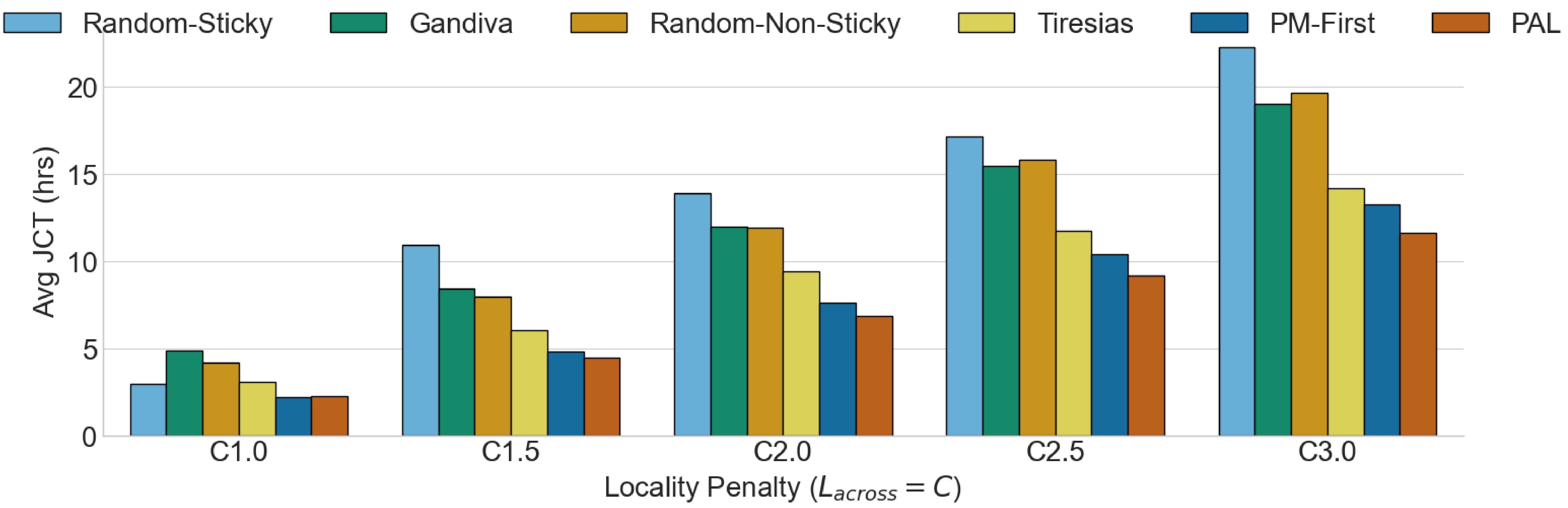




Large jobs hog cluster resources, causing cascading wait time impact!

Jobs finish faster with PAL and subsequent wait times consequently reduce

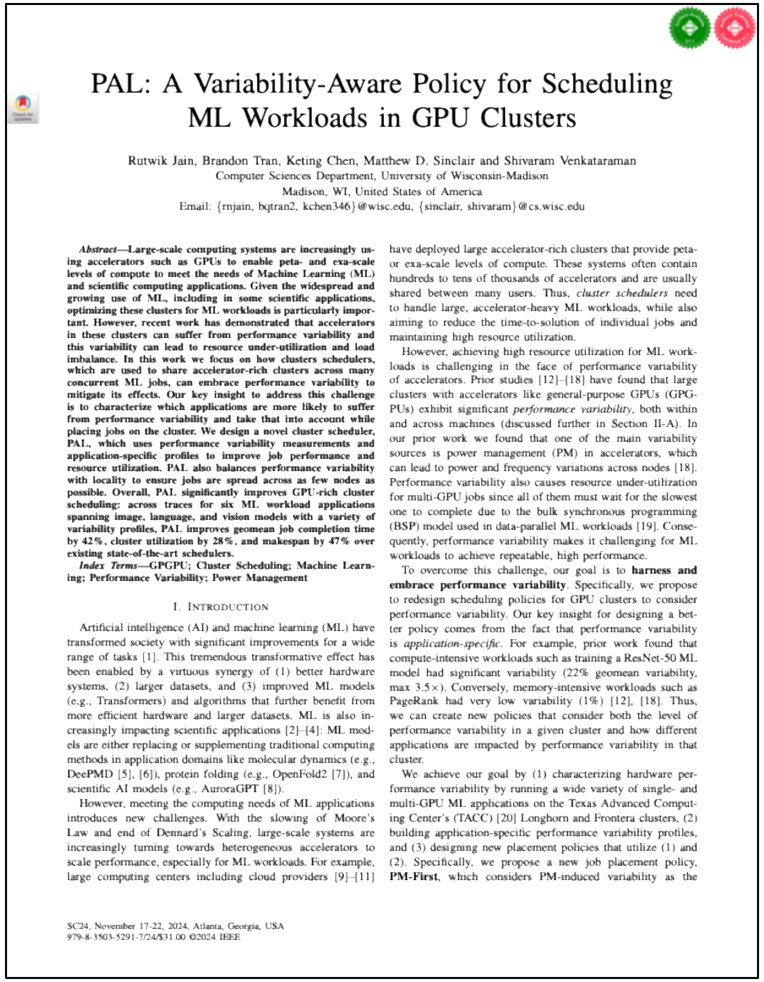
Varying Locality Penalty



packing is more important

job arrival rate, cluster utilization, placement overheads…

- Varying cluster size (64 vs 256) – **Section V-B vs V-C**
- Varying locality penalty – **Section V-B (1) and V-C (2)**
- Varying job load (jobs/hr) – **Section V-C (1)**
  - Average JCT
  - Utilization
- Different proportions of single GPU jobs - **Section V-B vs V-C**
- Varying scheduling policy - **Section V-C (1)**
  - FIFO
  - SRTF
  - LAS
- Placement policy overheads - **Section V-C (2)**



PAL: A Variability-Aware Policy for Scheduling ML Workloads in GPU Clusters

Rutwik Jain, Brandon Tran, Keting Chen, Matthew D. Sinclair and Shivaram Venkataraman
Computer Sciences Department, University of Wisconsin-Madison
Madison, WI, United States of America
Email: {rnjain, bqtran2, kchen346}@wisc.edu, {sinclair, shivaram}@cs.wisc.edu

***Abstract*—Large-scale computing systems are increasingly using accelerators such as GPUs to enable peta- and exa-scale levels of compute to meet the needs of Machine Learning (ML) and scientific computing applications. Given the widespread and growing use of ML, including in some scientific applications, optimizing these clusters for ML workloads is particularly important. However, recent work has demonstrated that accelerators in these clusters can suffer from performance variability and this variability can lead to resource under-utilization and load imbalance. In this work we focus on how clusters schedulers, which are used to share accelerator-rich clusters across many concurrent ML jobs, can embrace performance variability to mitigate its effects. Our key insight to address this challenge is to characterize which applications are more likely to suffer from performance variability and take that into account while placing jobs on the cluster. We design a novel cluster scheduler, PAL, which uses performance variability measurements and application-specific profiles to improve job performance and resource utilization. PAL also balances performance variability with locality to ensure jobs are spread across as few nodes as possible. Overall, PAL significantly improves GPU-rich cluster scheduling: across traces for six ML workload applications spanning image, language, and vision models with a variety of variability profiles, PAL improves geomean job completion time by 42%, cluster utilization by 28%, and makespan by 47% over existing state-of-the-art schedulers.**

***Index Terms*—GPGPU; Cluster Scheduling; Machine Learning; Performance Variability; Power Management**

I. Introduction

Artificial intelligence (AI) and machine learning (ML) have transformed society with significant improvements for a wide range of tasks [1]. This tremendous transformative effect has been enabled by a virtuous synergy of (1) better hardware systems, (2) larger datasets, and (3) improved ML models (e.g., Transformers) and algorithms that further benefit from more efficient hardware and larger datasets. ML is also increasingly impacting scientific applications [2]–[4]: ML models are either replacing or supplementing traditional computing methods in application domains like molecular dynamics (e.g., DeePMD [5], [6]), protein folding (e.g., OpenFold2 [7]), and scientific AI models (e.g., AuroraGPT [8]).

However, meeting the computing needs of ML applications introduces new challenges. With the slowing of Moore's Law and end of Dennard's Scaling, large-scale systems are increasingly turning towards heterogeneous accelerators to scale performance, especially for ML workloads. For example, large computing centers including cloud providers [9]–[11] have deployed large accelerator-rich clusters that provide peta- or exa-scale levels of compute. These systems often contain hundreds to tens of thousands of accelerators and are usually shared between many users. Thus, *cluster schedulers* need to handle large, accelerator-heavy ML workloads, while also aiming to reduce the time-to-solution of individual jobs and maintaining high resource utilization.

However, achieving high resource utilization for ML workloads is challenging in the face of performance variability of accelerators. Prior studies [12]–[18] have found that large clusters with accelerators like general-purpose GPUs (GPGPUs) exhibit significant *performance variability*, both within and across machines (discussed further in Section II-A). In our prior work we found that one of the main variability sources is power management (PM) in accelerators, which can lead to power and frequency variations across nodes [18]. Performance variability also causes resource under-utilization for multi-GPU jobs since all of them must wait for the slowest one to complete due to the bulk synchronous programming (BSP) model used in data-parallel ML workloads [19]. Consequently, performance variability makes it challenging for ML workloads to achieve repeatable, high performance.

To overcome this challenge, our goal is to **harness and embrace performance variability**. Specifically, we propose to redesign scheduling policies for GPU clusters to consider performance variability. Our key insight for designing a better policy comes from the fact that performance variability is *application-specific*. For example, prior work found that compute-intensive workloads such as training a ResNet-50 ML model had significant variability (22% geomean variability, max 3.5×). Conversely, memory-intensive workloads such as PageRank had very low variability (1%) [12], [18]. Thus, we can create new policies that consider both the level of performance variability in a given cluster and how different applications are impacted by performance variability in that cluster.

We achieve our goal by (1) characterizing hardware performance variability by running a wide variety of single- and multi-GPU ML applications on the Texas Advanced Computing Center's (TACC) [20] Longhorn and Frontera clusters, (2) building application-specific performance variability profiles, and (3) designing new placement policies that utilize (1) and (2). Specifically, we propose a new job placement policy, **PM-First**, which considers PM-induced variability as the

SC24, November 17-22, 2024, Atlanta, Georgia, USA
979-8-3503-5291-7/24/$31.00 ©2024 IEEE

CONCLUSION

CONCLUSION

Variability can affect cluster performance, utilization and load balancing on GPU clusters. Likely to get worse as **ML algorithms** and **cluster size** scales up, while **transistors** shrink.

CONCLUSION

Variability can affect cluster performance, utilization and load balancing on GPU clusters. Likely to get worse as **ML algorithms** and cl**uster size** scales up, while **transistors** shrink.

Key Insight: Schedulers must **embrace** and harness this variability.

CONCLUSION

Variability can affect cluster performance, utilization and load balancing on GPU clusters. Likely to get worse as **ML algorithms** and cl**uster size** scales up, while **transistors** shrink.

Key Insight: Schedulers must **embrace** and harness this variability.

We proposed **PAL** which uses application-specific variability to optimize job placement. Across a mix of ML workloads, PAL improves on SOTA ML cluster schedulers across metrics.

CONCLUSION

Variability can affect cluster performance, utilization and load balancing on GPU clusters.
Likely to get worse as **ML algorithms** and clu**ster size** scales up, while **transistors** shrink.

Key Insight: Schedulers must **embrace** and harness this variability.

We proposed **PAL** which uses application-specific variability to optimize job placement.
Across a mix of ML workloads, PAL improves on SOTA ML cluster schedulers across metrics.

We are working to extend this to HPC and HPC+ML workloads.
Looking at HW-SW codesign with variability as a first-order constraint.

SUMMARY

Variability can affect cluster performance, utilization and load balancing on GPU clusters
Likely to get worse as ML algorithms and cluster size scales up, while transistors shrink

Key Insight: Schedulers must embrace and harness this variability

We proposed **PAL** which uses application-specific variability to optimize job placement
Across a mix of ML workloads, PAL improves on SOTA ML cluster schedulers across metrics.

We are working to extend this to HPC and HPC+ML workloads.
Looking at HW-SW codesign with variability as a first-order constraint.

Contact

rnjain@wisc.edu

https://pages.cs.wisc.edu/~rnjain/

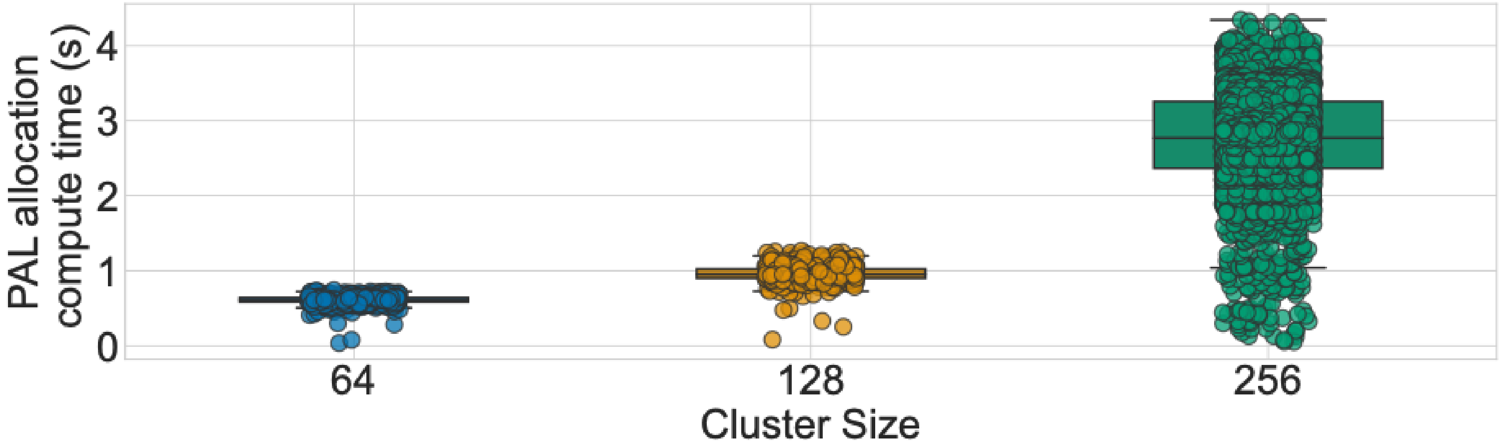
Artifact

THANK YOU

Questions?

Placement Policy Overheads

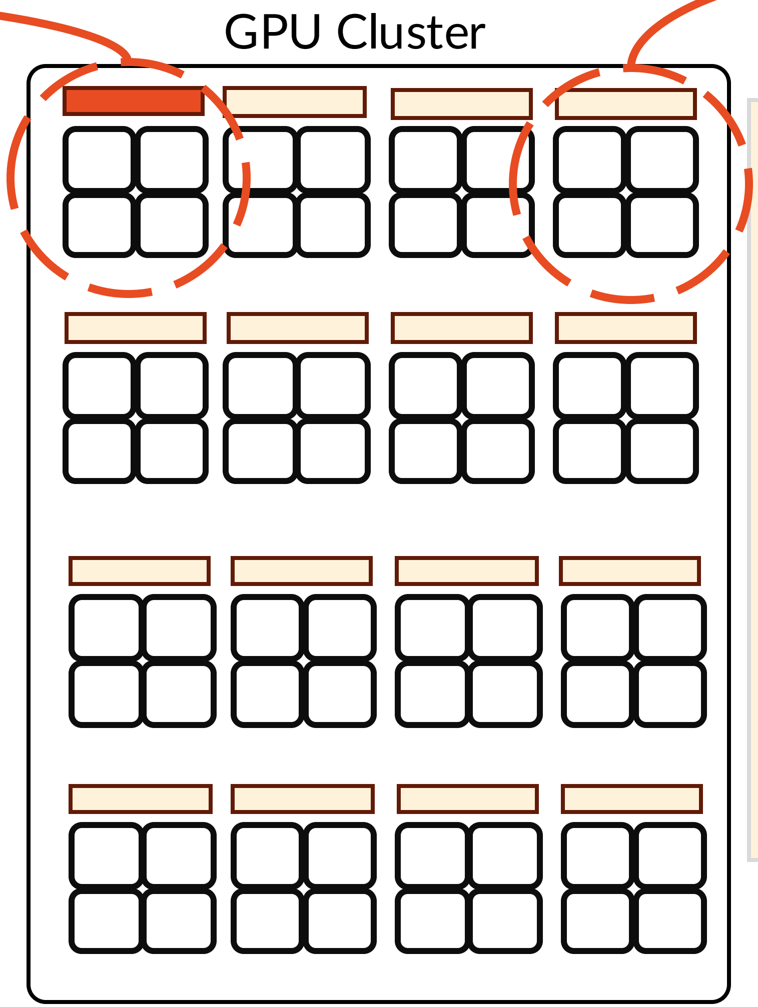
- Scheduling round duration is 300s (5 mins)
- PAL
  - worst-case 4 seconds
  - median of 2.8 seconds
- PM-First
  - worst-case 2 seconds



Blox Cluster Setup

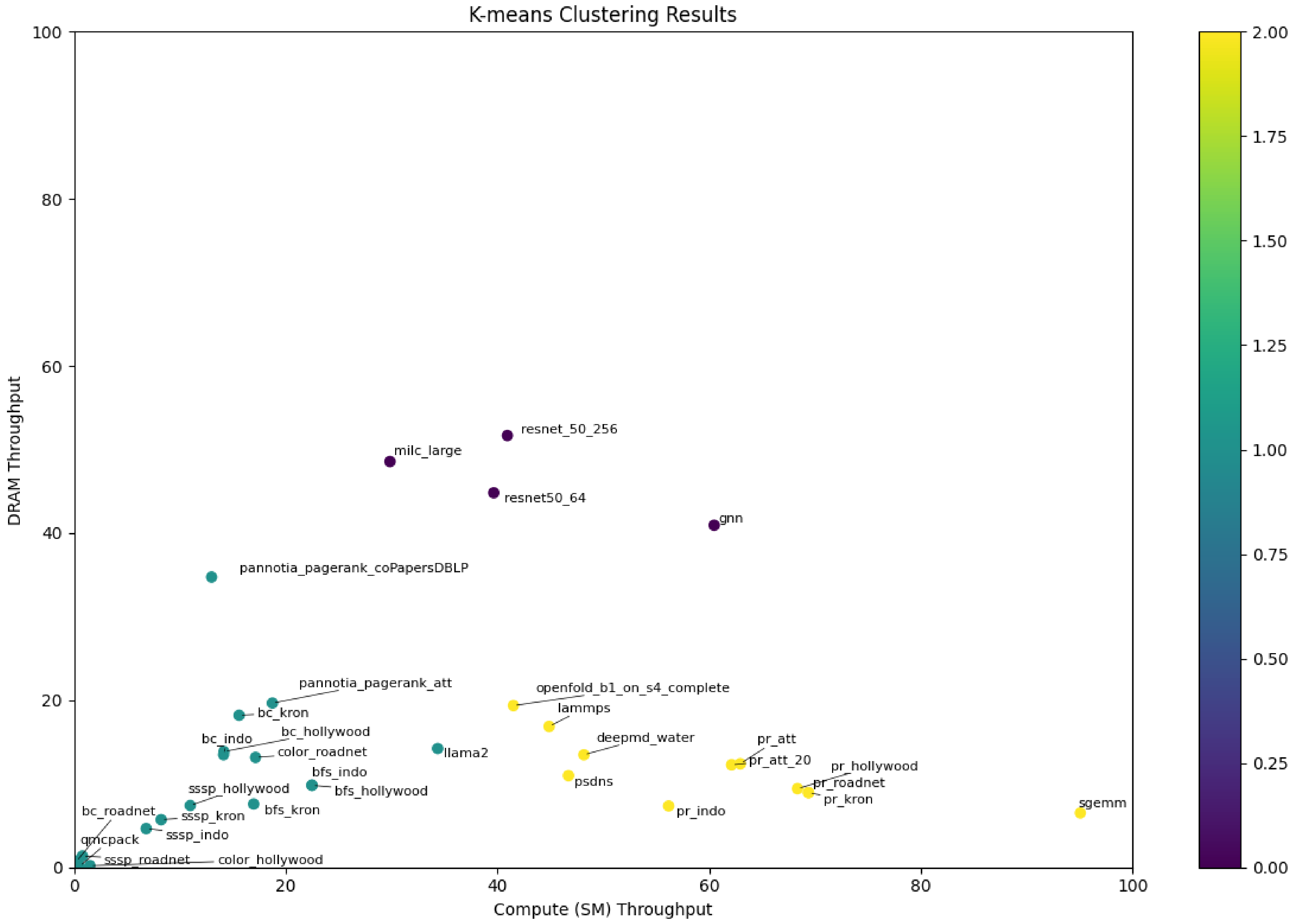
Scheduler Node: CPU Host

- gRPC Server that registers all GPU UUIDs.
- Another CPU threads sends job requests at specified arrival times.
- Scheduler thread runs placement policy algorithm to determine set of GPUs to run on.
- Server dispatches job by sending a gRPC packet to the clients running on relevant node(s).



Execution Node

- Host on each execution node acts as gRPC client
- At startup, client registers it’s GPUs with the scheduler node and receives acknowledgement.
- It then waits for job request packets from server
- Whenever it receives a job request, it runs corresponding job command on specified list of GPUs.


K-means Clustering Results
DRAM Throughput
Compute (SM) Throughput
0
20
40
60
80
100
resnet_50_256
milc_large
resnet50_64
gnn
pannotia_pagerank_coPapersDBLP
pannotia_pagerank_att
bc_kron
openfold_b1_on_s4_complete
lammps
bc_indo
bc_hollywood
color_roadnet
llama2
deepmd_water
pr_att
pr_att_20
pr_hollywood
pr_roadnet
pr_kron
psdns
bfs_indo
bfs_hollywood
sssp_hollywood
bfs_kron
sssp_kron
sssp_indo
bc_roadnet
qmcpack
sssp_roadnet
color_hollywood
pr_indo
sgemm
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

How do we define variability?

Interquartile Range

$$IQR = Q_3 - Q_2$$

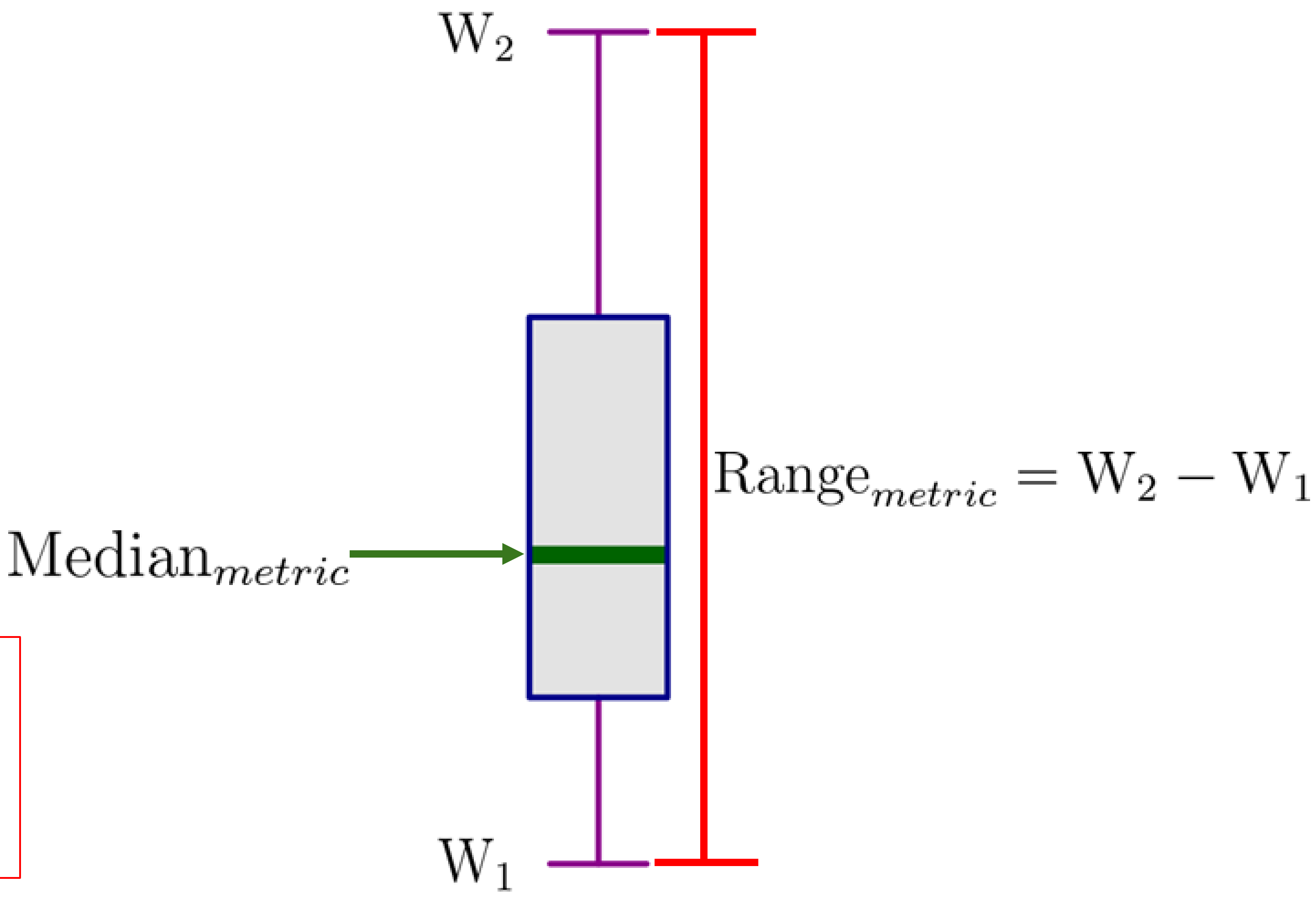
Whiskers and Range

$$W_2 = Q_3 + 1.5 \times IQR$$

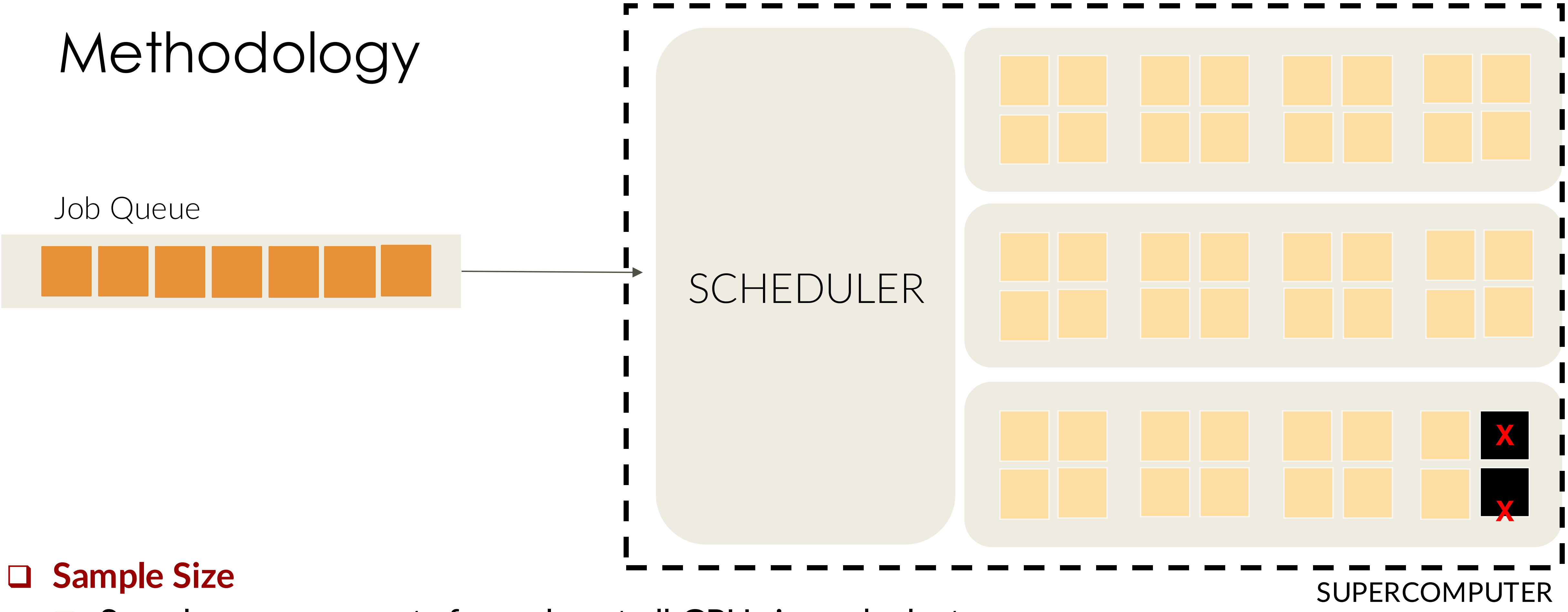
$$W_1 = Q_1 - 1.5 \times IQR$$

$$Range_{metric} = W_2 - W_1$$

$$\boxed{Variability_{metric} = \frac{Range_{metric}}{Median_{metric}}}$$

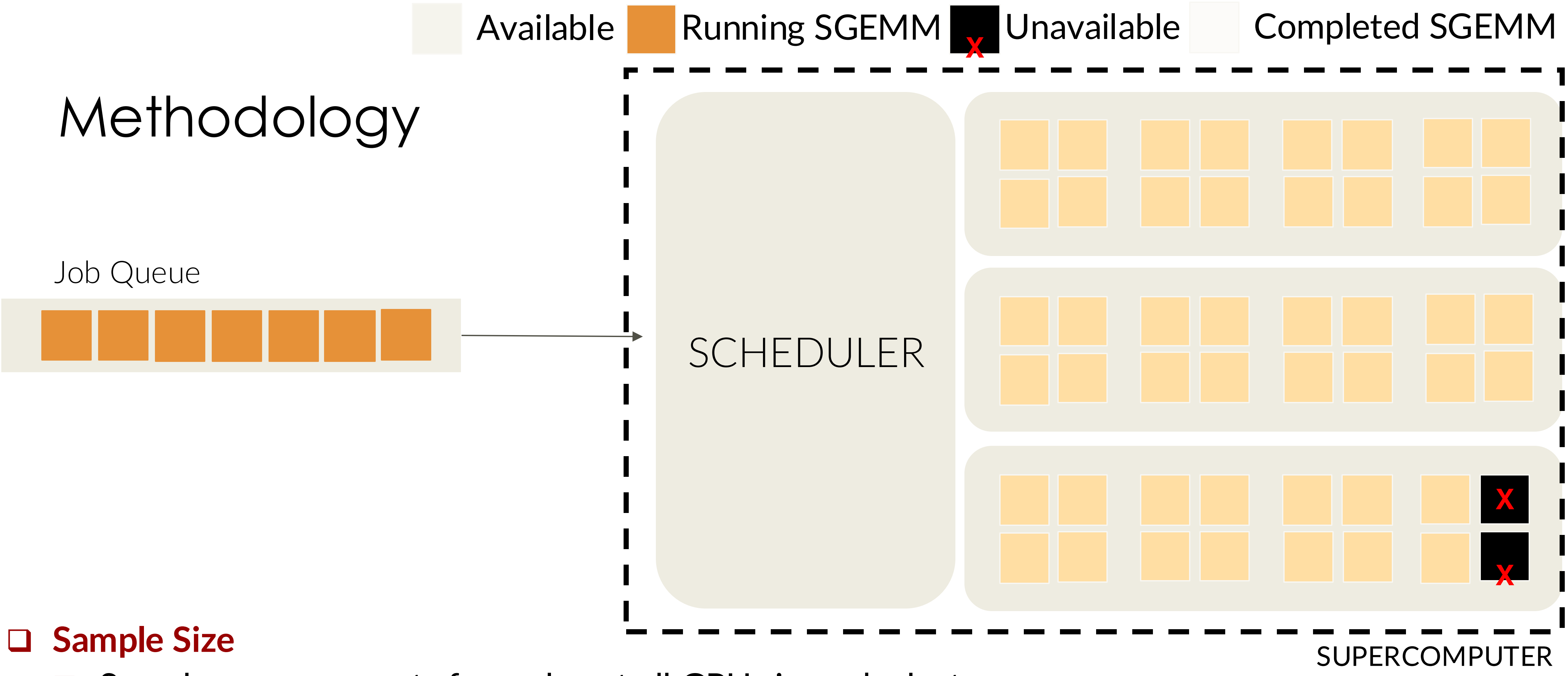


Methodology

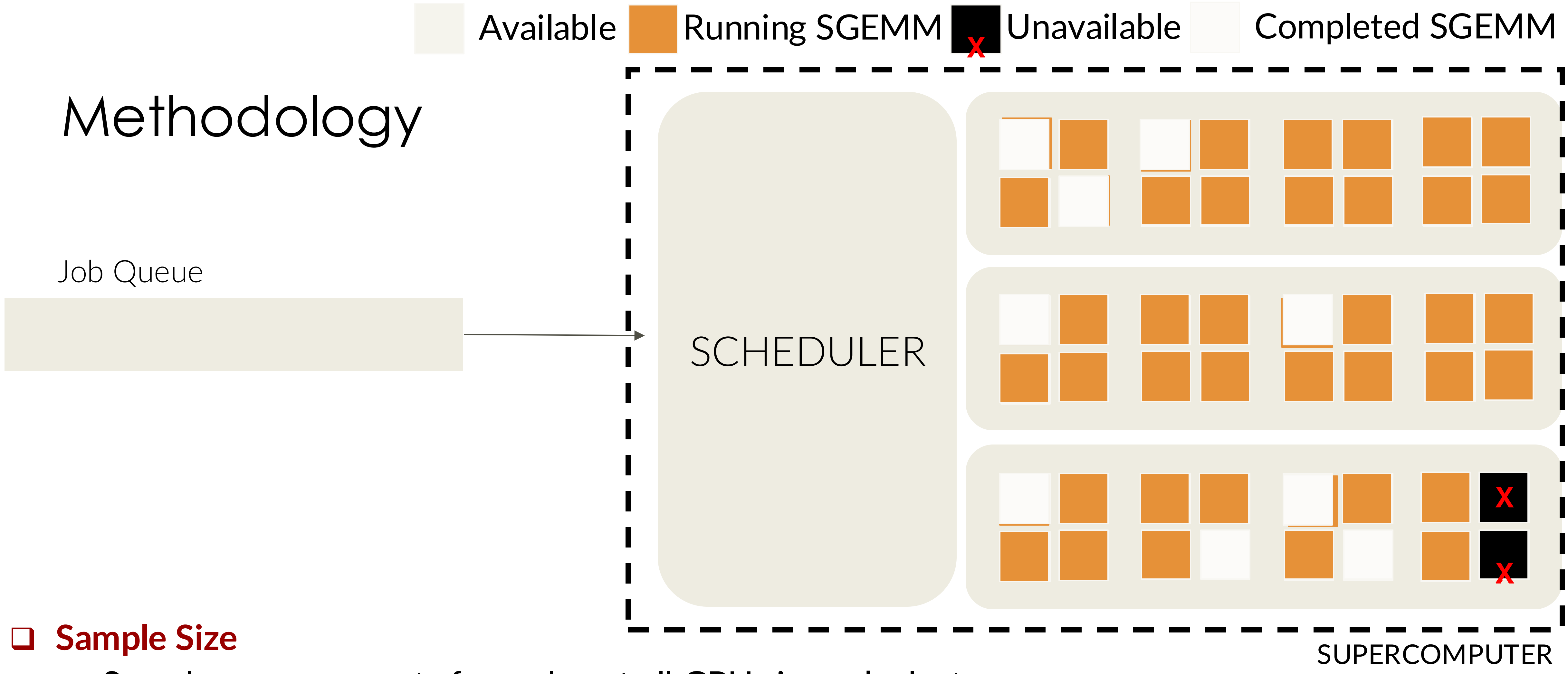


- ❑ **Sample Size**
  - ❑ Sample measurements from almost all GPUs in each cluster
  - ❑ Profiled 2.9x more GPUs than worst-case recommendations for statistical significance [Scogland, et al. SC'15]

Methodology



- **Sample Size**
  - Sample measurements from almost all GPUs in each cluster
  - Profiled 2.9x more GPUs than worst-case recommendations for statistical significance [Scogland, et al. SC'15]



Methodology

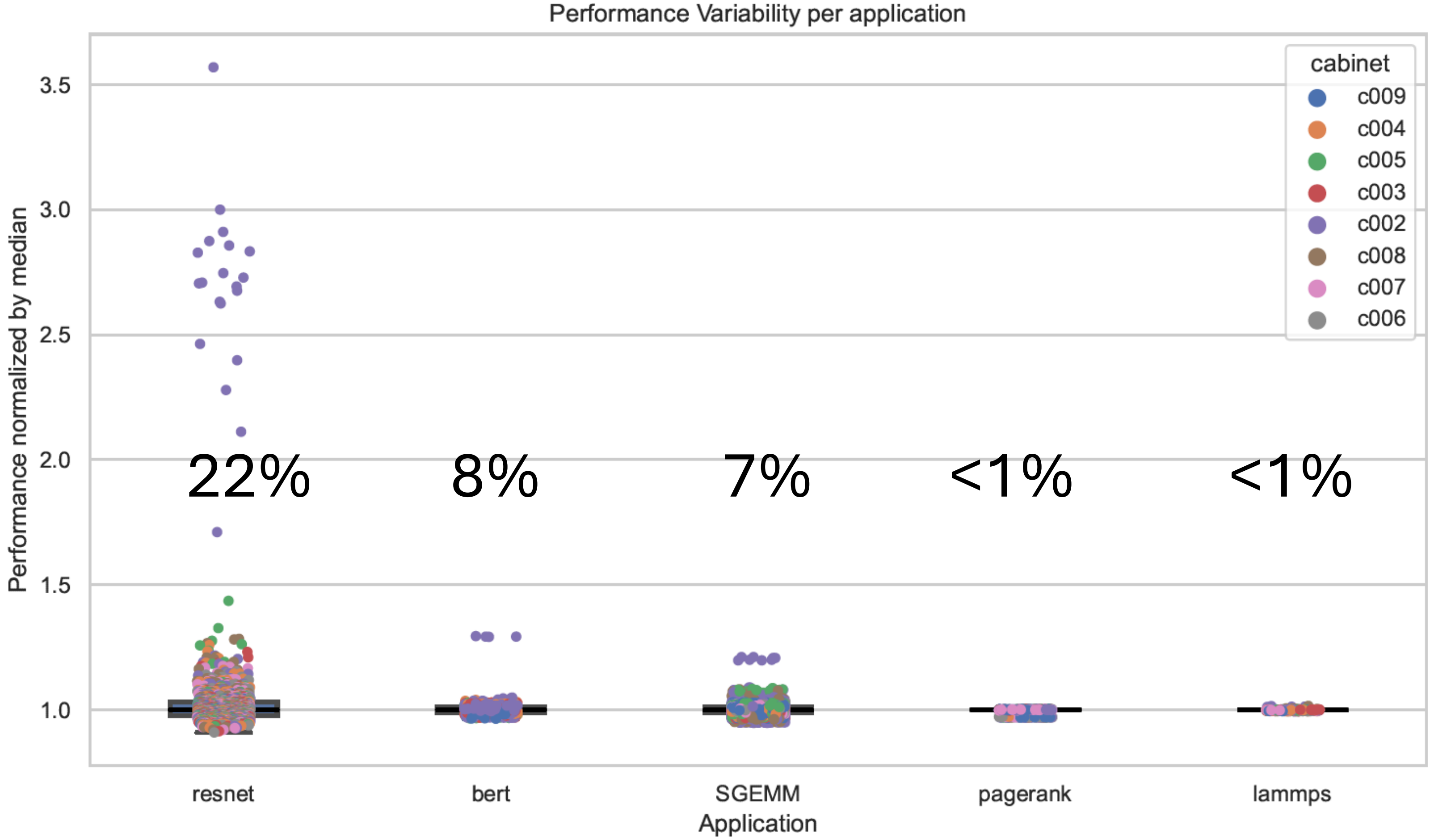
- **Sample Size**
  - Sample measurements from almost all GPUs in each cluster
  - Profiled 2.9x more GPUs than worst-case recommendations for statistical significance [Scogland, et al. SC'15]

CLUSTER PARAMETERS

Cluster	GPU Architecture	Number of GPUs	Cooling Method
LLNL Corona	AMD MI60	328	**air** cooled
TACC Longhorn	NVIDIA V100	416	**air** cooled
TACC Frontera	NVIDIA Quadro RTX 5000	360	**mineral oil** cooled
SNL Vortex	NVIDIA V100	216	**water** cooled
ORNL Summit	NVIDIA V100	27648	**air** cooled

Not All GPUs Are Created Equal: Characterizing Variability in Large-Scale, Accelerator-Rich Systems
https://dl.acm.org/doi/abs/10.5555/3571885.3571971

VARIABILITY ACROSS APPLICATIONS



Compute bound for the input size and configurations they were run in

Memory bound