Synergy: Looking Beyond GPUs for DNN Scheduling on Multi-Tenant Clusters

Jayashree Mohan, Amar Phanishayee, and Janardhan Kulkarni, MS Research; Vijay Chidambaram, The University of Texas and VMware Research

USENIX OSDI'22

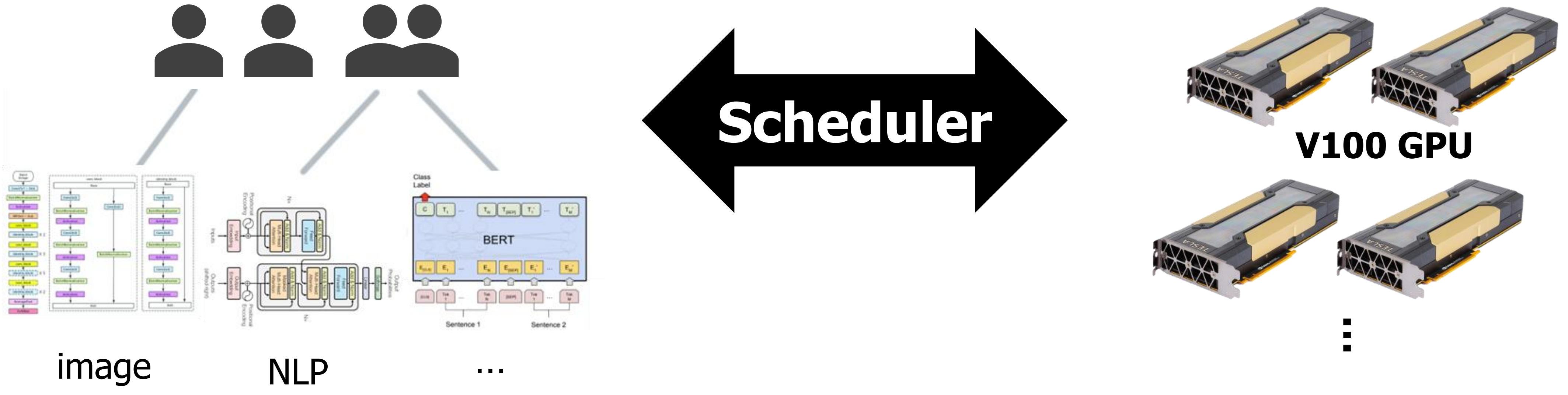
슬라이드 노트에 각 장별 설명이 나와있습니다!

Presented by Yejin Han

yj0225@dankook.ac.kr

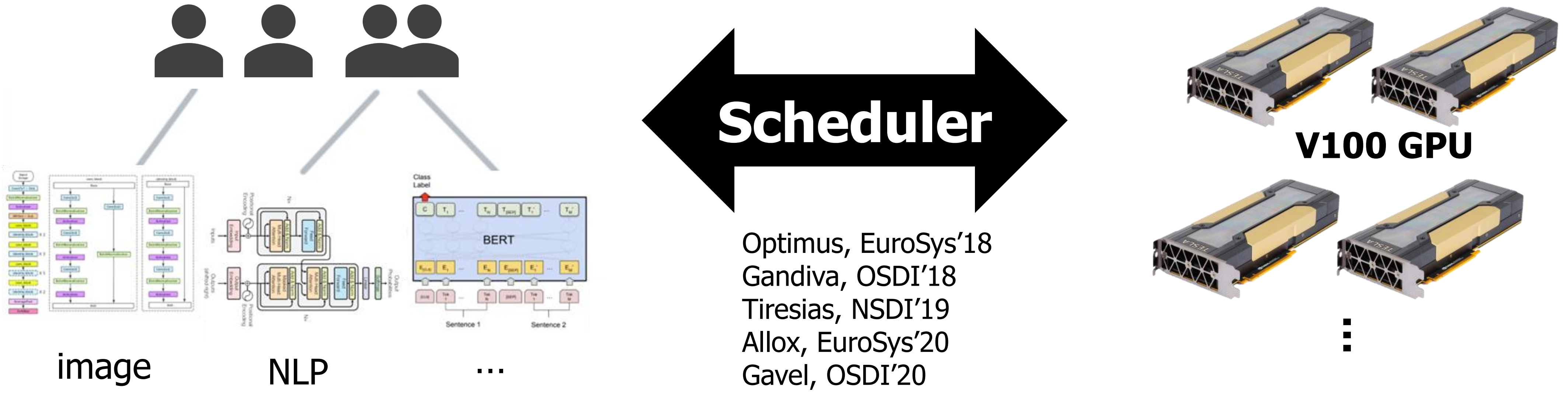
Introduction

- DNN training on multi-tenant clusters
  - Co-locating training workloads in a shared, multi-tenant cluster is a common setup



Introduction

- DNN training on multi-tenant clusters
  - Co-locating training workloads in a shared, multi-tenant cluster is a common setup

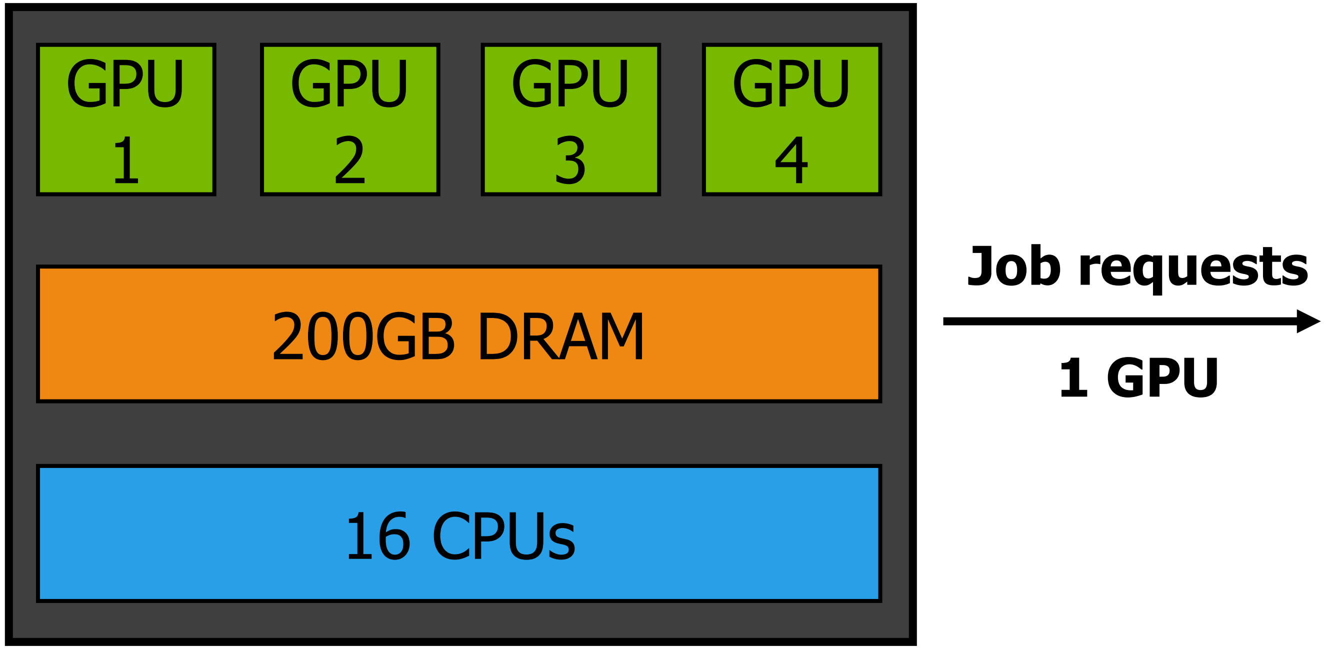


Optimus, EuroSys’18
Gandiva, OSDI’18
Tiresias, NSDI’19
Allox, EuroSys’20
Gavel, OSDI’20
…

Existing DNN cluster schedulers allocate resources GPU-proportional

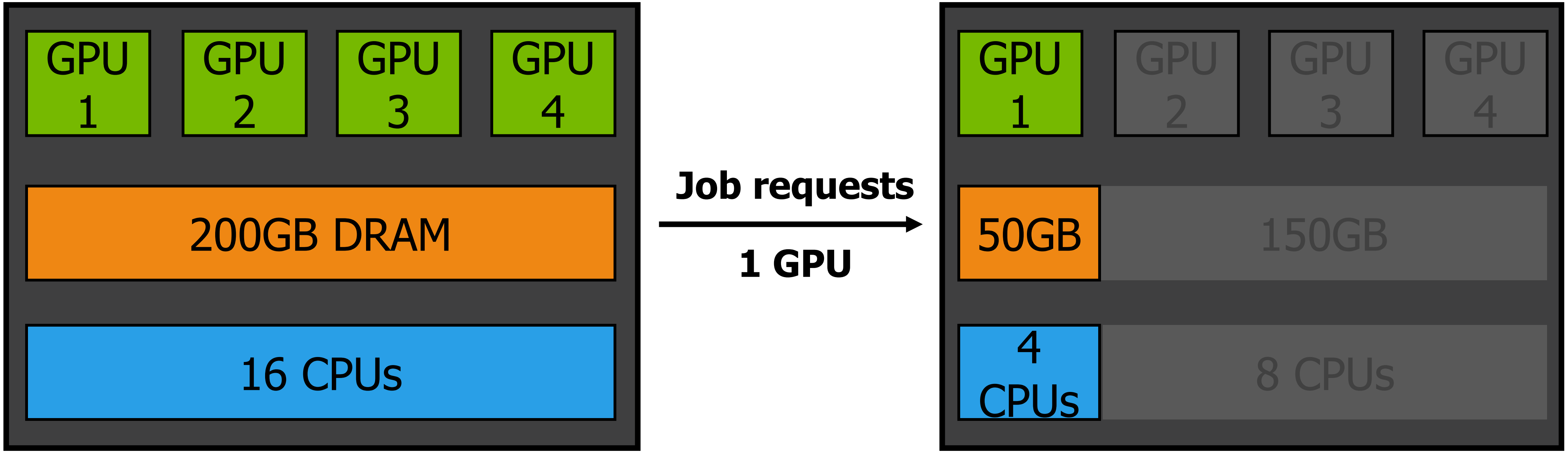
What is GPU-proportional allocation?

- User requests a fixed number of GPUs for their DNN job
- Other resources are allocated proportional to the number of GPUs



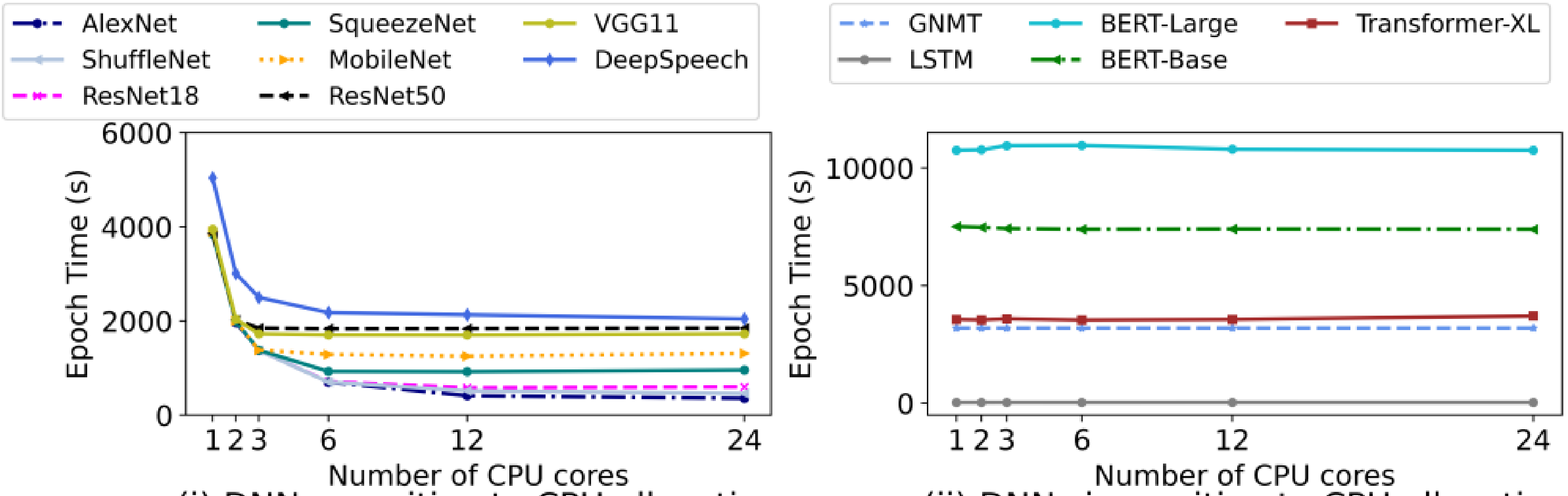
What is GPU-proportional allocation?

- User requests a fixed number of GPUs for their DNN job
- Other resources are allocated proportional to the number of GPUs



Motivation: Resource-sensitivity

- DNNs exhibit varied sensitivity to the amount of CPU and DRAM



(i) DNNs sensitive to CPU allocation

(ii) DNNs insensitive to CPU allocation

(a) CPU sensitivity

CPU: GPU	SKU
3:1	NVIDIA DGX-2 Internal servers at X
4:1	AWS p3.16xlarge
5:1	NVIDIA DGX-1 Azure NDv2
6:1	Azure NC24s_v3

(b) GPU VM SKUs

We need to consider resource sensitivity of DNN training jobs

Motivation: Resource-sensitivity

- Co-locate a CPU/memory sensitive job with an insensitive one

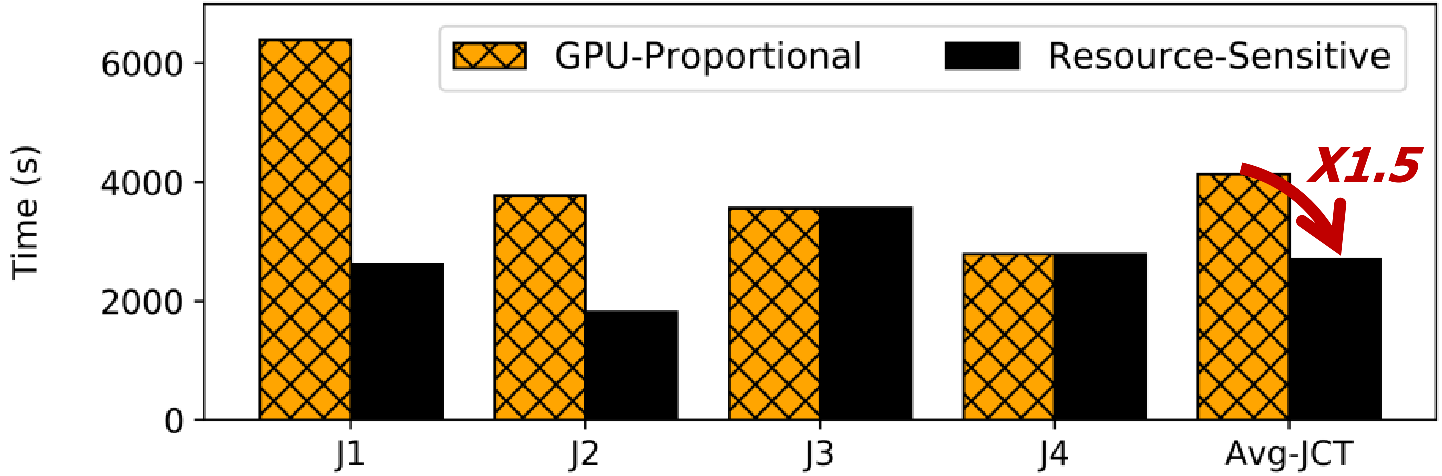


Figure 3: **Resource sensitive scheduling**. We compare the runtime of the jobs with two different schedules; GPU-proportional and resource-sensitive. By allocating resources disproportionately, CPU and memory sensitive jobs see increased throughputs which reduces the average JCT by 1.5×.

Job	Model
J_1	ResNet18
J_2	Audio-M5
J_3	Transformer
J_4	GNMT

Table 1: Example jobs

Server	Job	GPU	CPU	Mem
S_1	$J1$	4	12	250
	$J2$	4	12	250
S_2	$J3$	4	12	250
	$J4$	4	12	250

Table 2: GPU-proportional allocation

Server	Job	GPU	CPU	Mem
S_1	$J1$	4	23	400
	$J3$	4	1	100
S_2	$J2$	4	12	450
	$J4$	4	12	50

Table 3: Resource-sensitive allocation

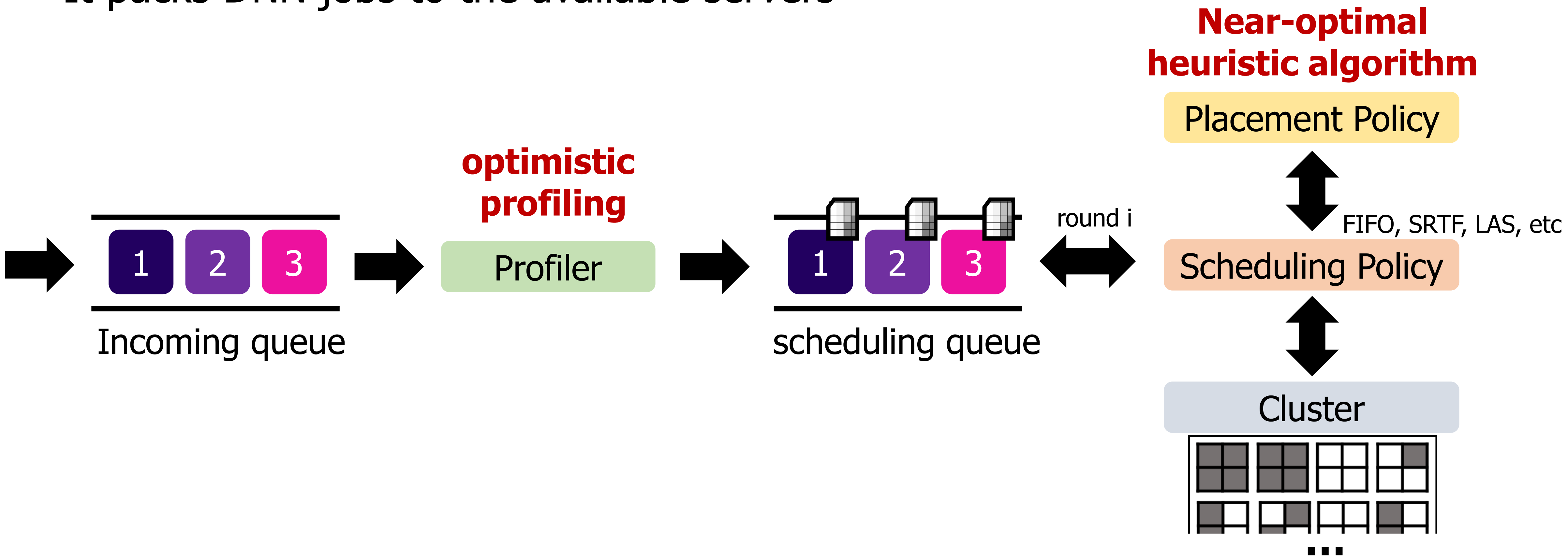
What are the challenges?

- What is the ideal resource requirement for each job?
- How can this be determined with low overhead?
- How should we pack these jobs onto servers efficiently?

Synergy

* FIFO: First In First Out
* SRTF: Shortest Remaining Time First
* LAS: Least Attained Service

- Synergy is a round-based scheduler that arbitrates GPU, CPU, and memory
- It identifies the job's best-case CPU/memory requirements
- It packs DNN jobs to the available servers



Optimistic Profiling

- Resource-sensitivity matrix for combinations of CPU and memory
- Predictable job performance to memory allocation
  - Using DNN-aware caching MinIO (VLDB'21)

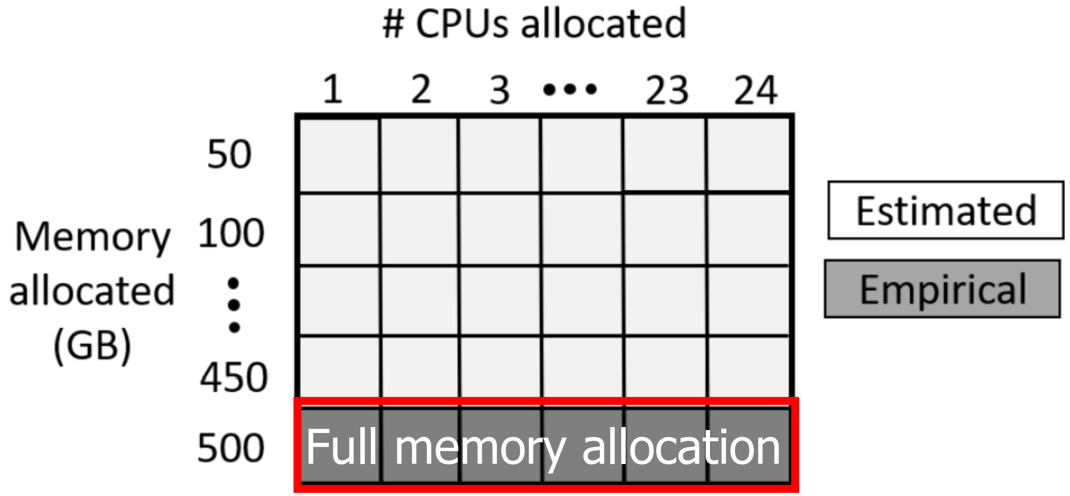
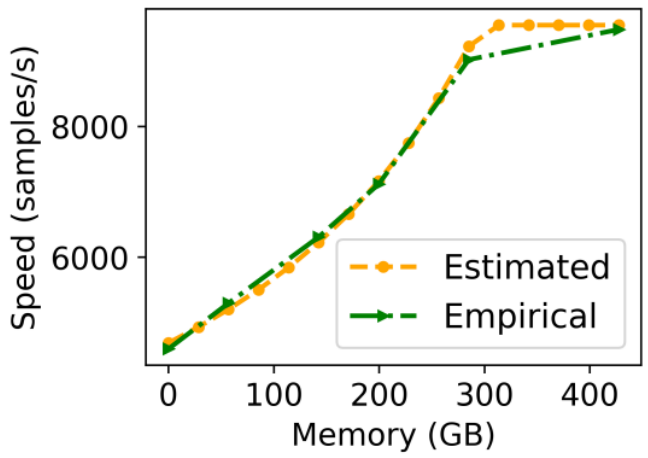


Figure 4: **Optimistic profiling** empirically evaluates the sensitivity of a model to varying # CPUs assuming a fully cached dataset; the rest of the matrix is completed using estimation

*8-GPU ResNet18 job

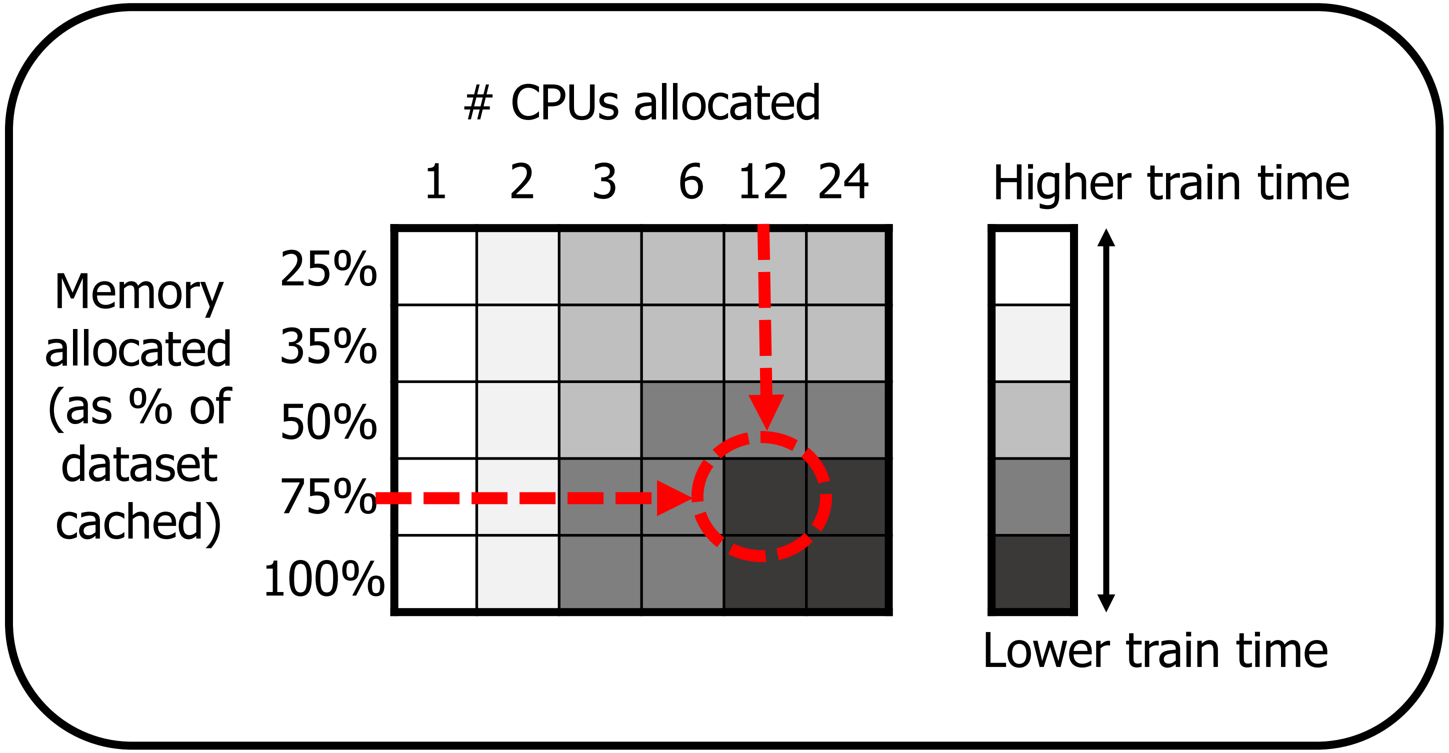


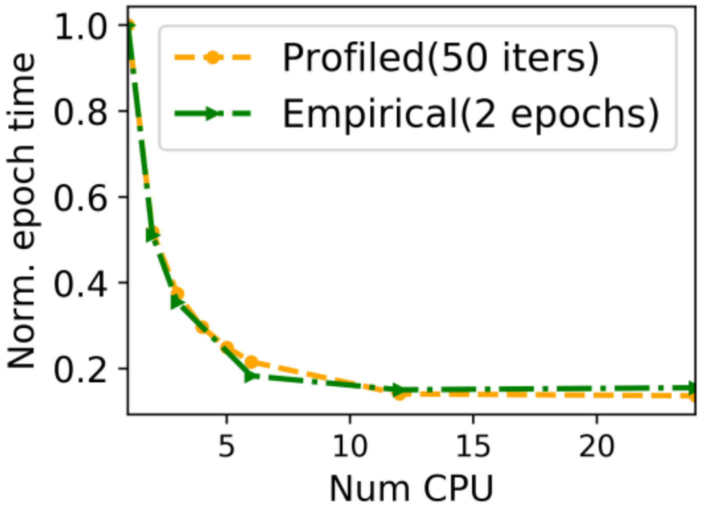
(a) Memory Validation

Figure 5: **Optimistic profiling**.

Optimistic Profiling

- Pick the minimum value of (CPU + memory) that saturates max throughput



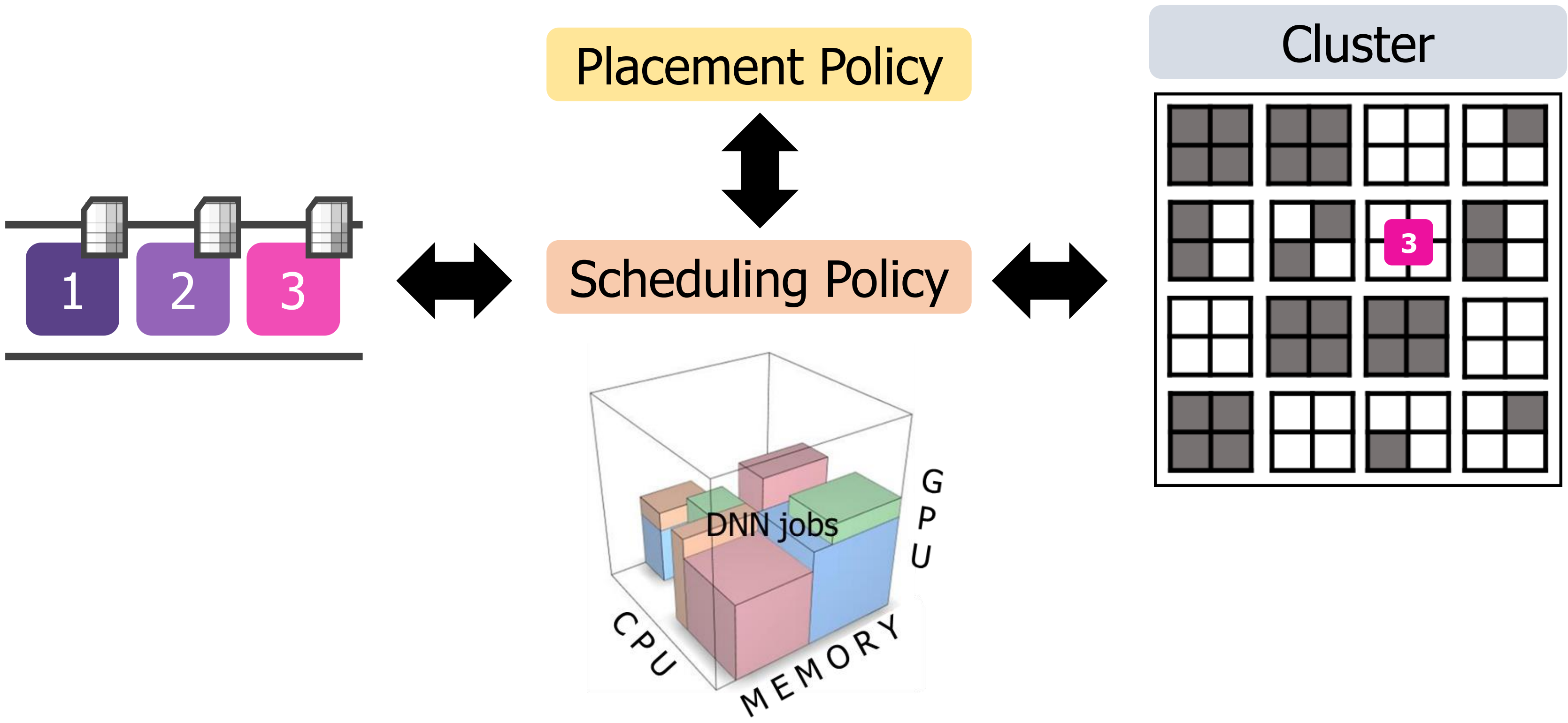


(b) CPU validation

Figure 5: **Optimistic profiling**.

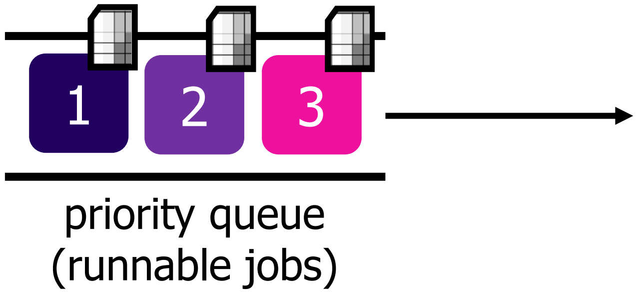
Job placement

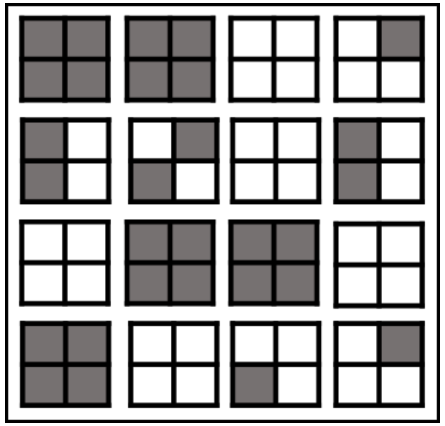
- What is the best placement for a set of jobs in the given round?
- Multi-dimensional bin packing problem



Synergy-TUNE scheduling

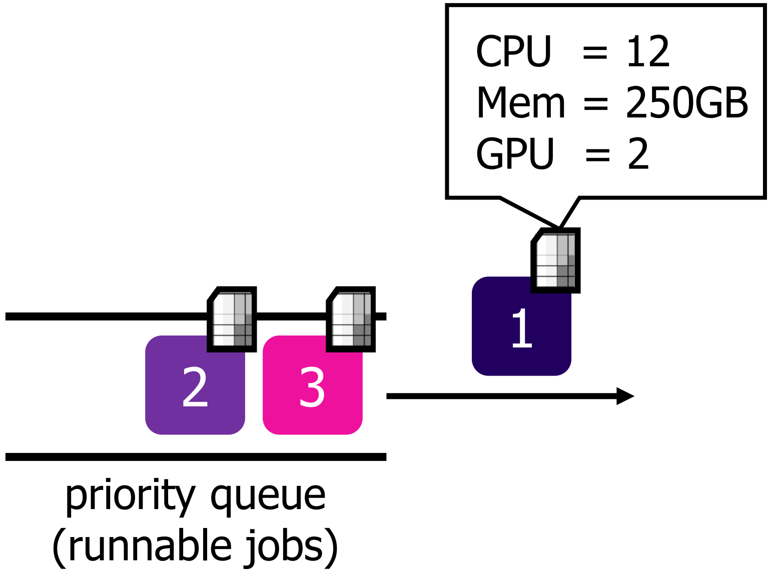
- Identifies runnable jobs for the current round from the scheduling queue

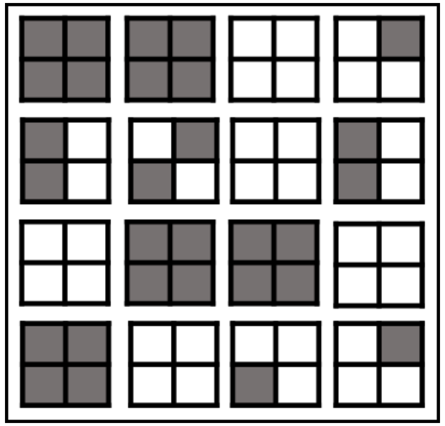




Synergy-TUNE scheduling

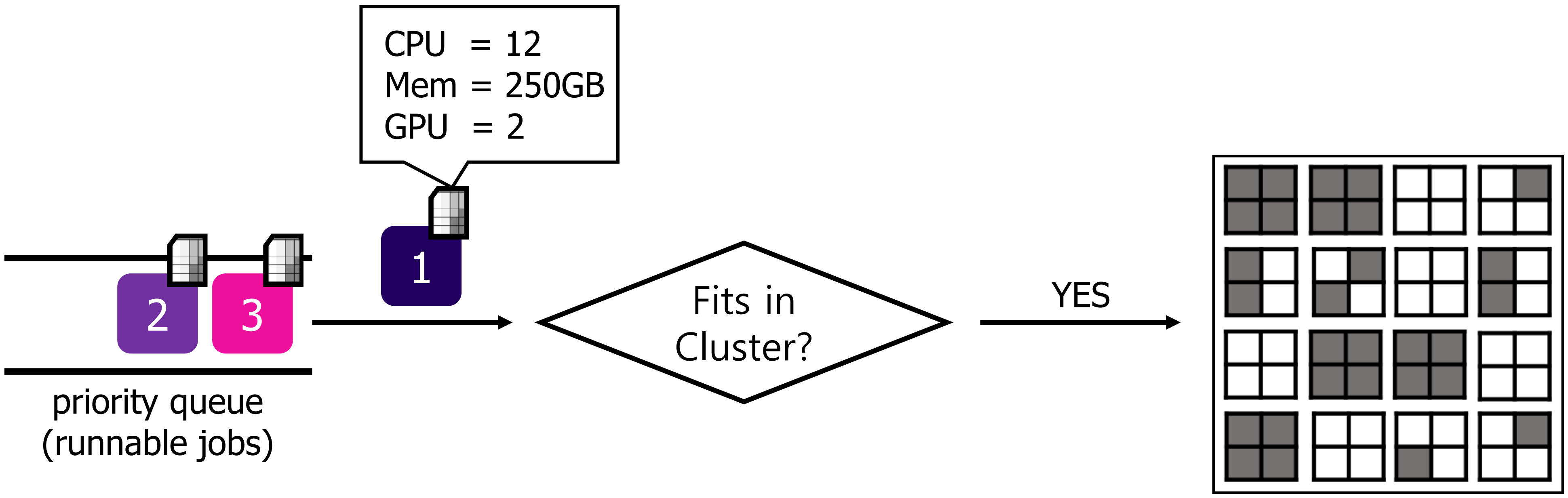
- The next job to be scheduled with the priority is picked from the queue





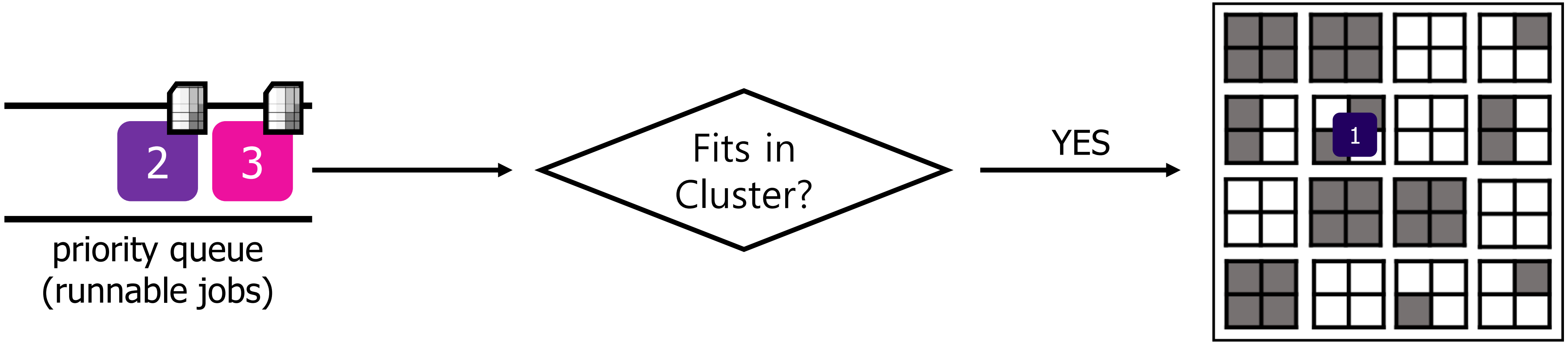
Synergy-TUNE scheduling

- Synergy checks if the jobs demand can be fitted in the cluster



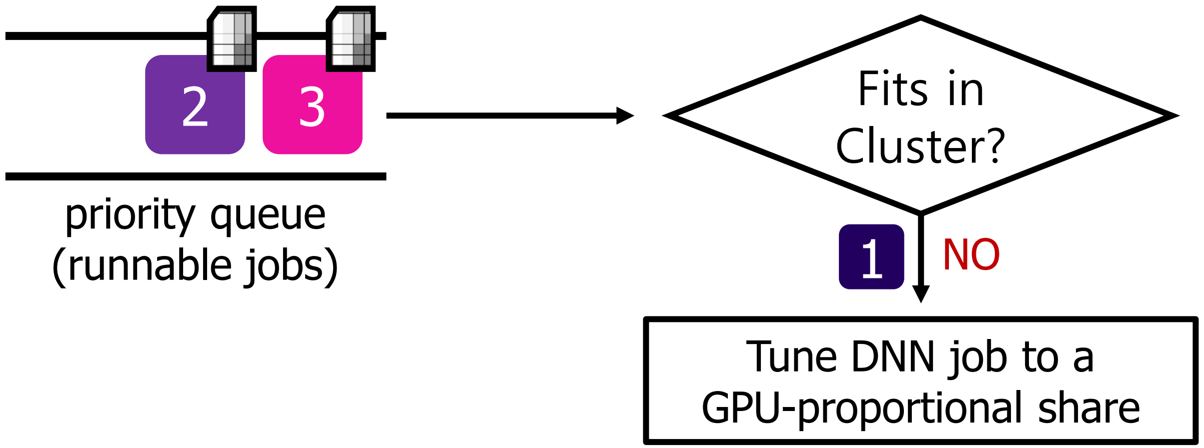
Synergy-TUNE scheduling

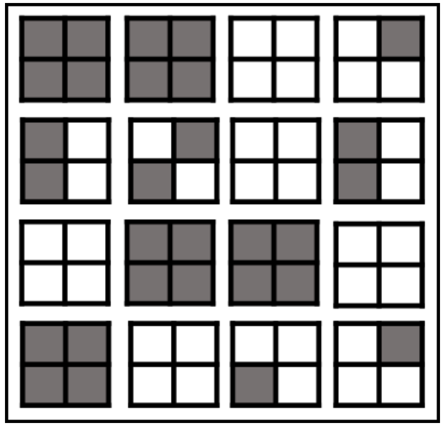
- Synergy checks if the jobs demand can be fitted in the cluster
- If it does fit the cluster, then the job is assigned to the cluster



Synergy-TUNE scheduling

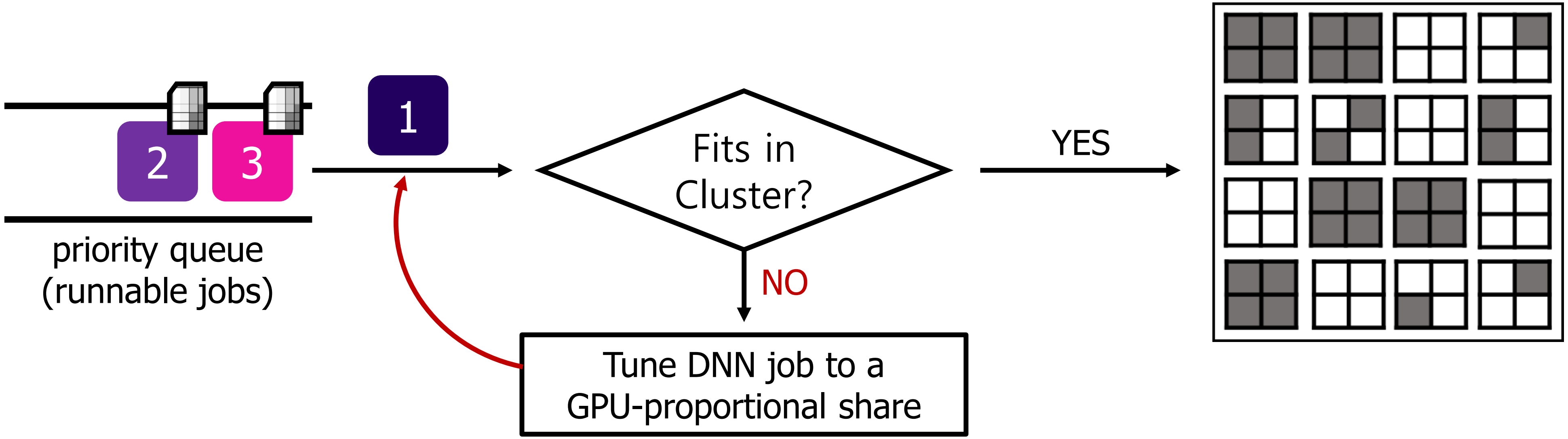
- Synergy check if the job's demand is greater than proportional share
- If so, switch to GPU-proportional share and retry





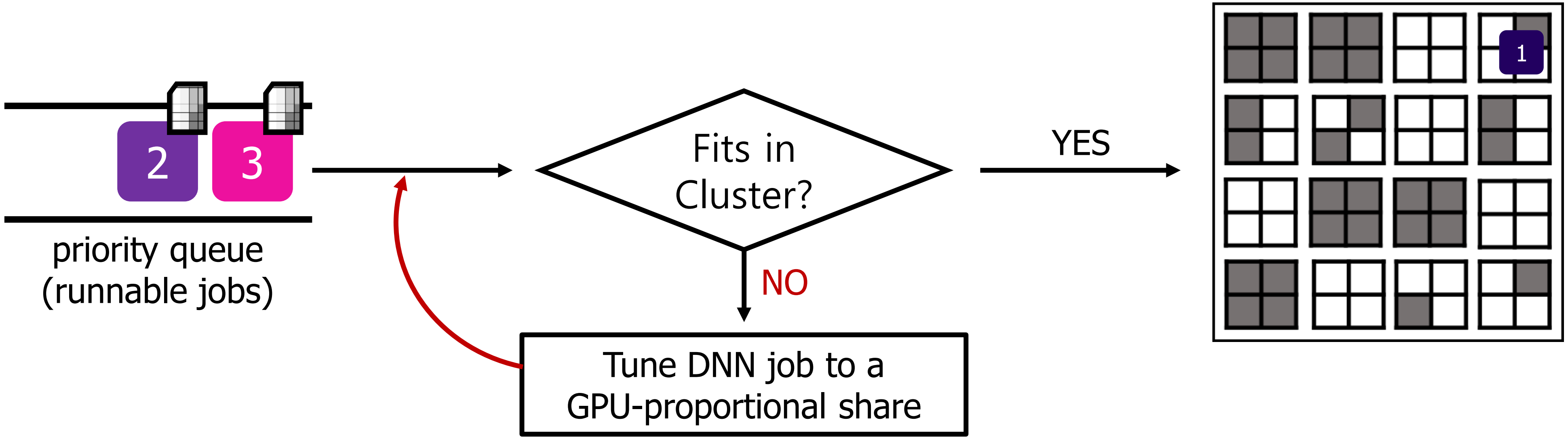
Synergy-TUNE scheduling

- Synergy check if the job's demand is greater than proportional share
- If so, switch to GPU-proportional share and retry



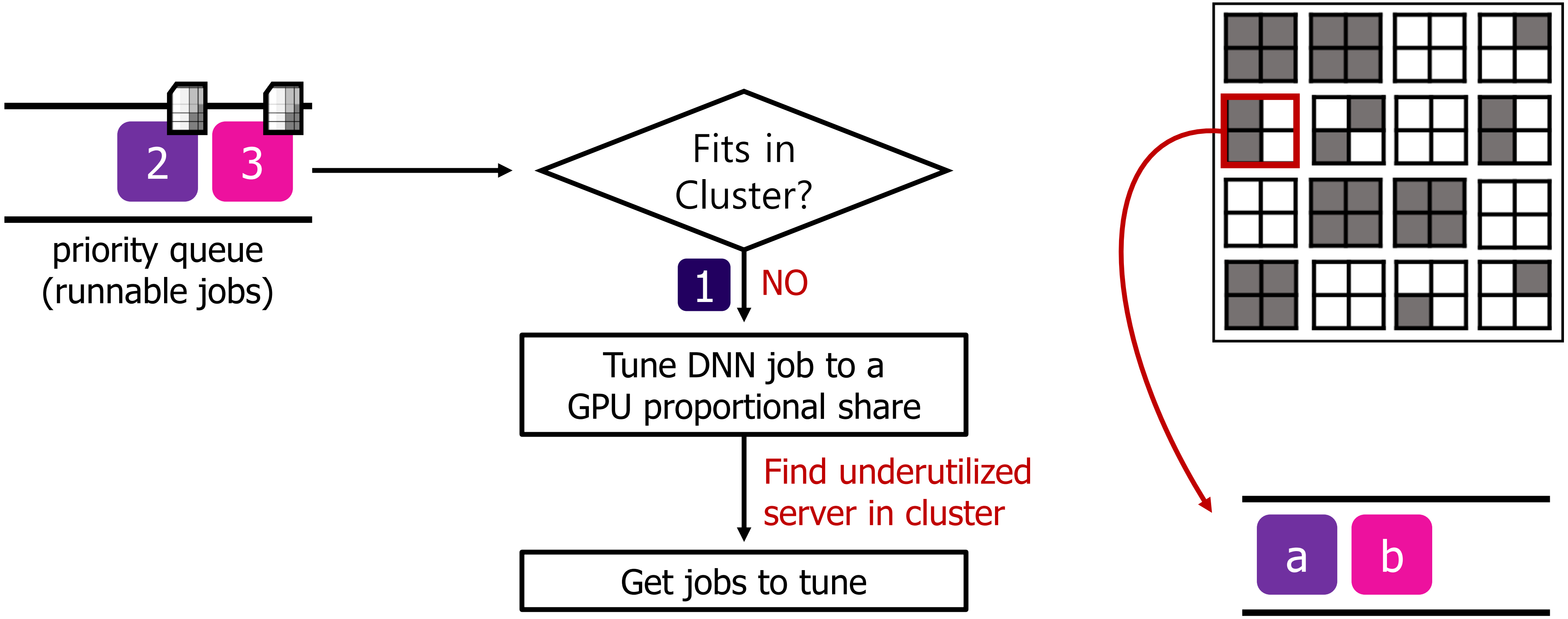
Synergy-TUNE scheduling

- Synergy check if the job's demand is greater than proportional share
- If so, switch to GPU-proportional share and retry



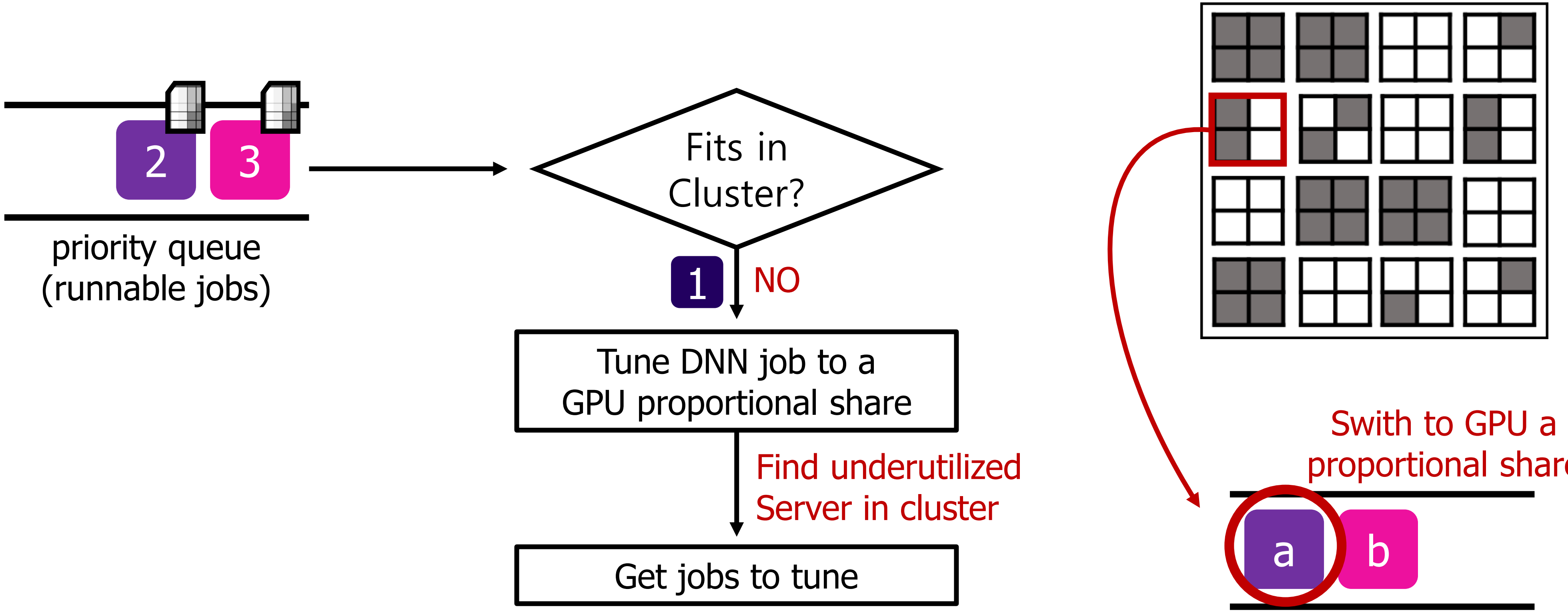
Synergy-TUNE scheduling

- If the job still doesn't fit the cluster, then find the underutilized server



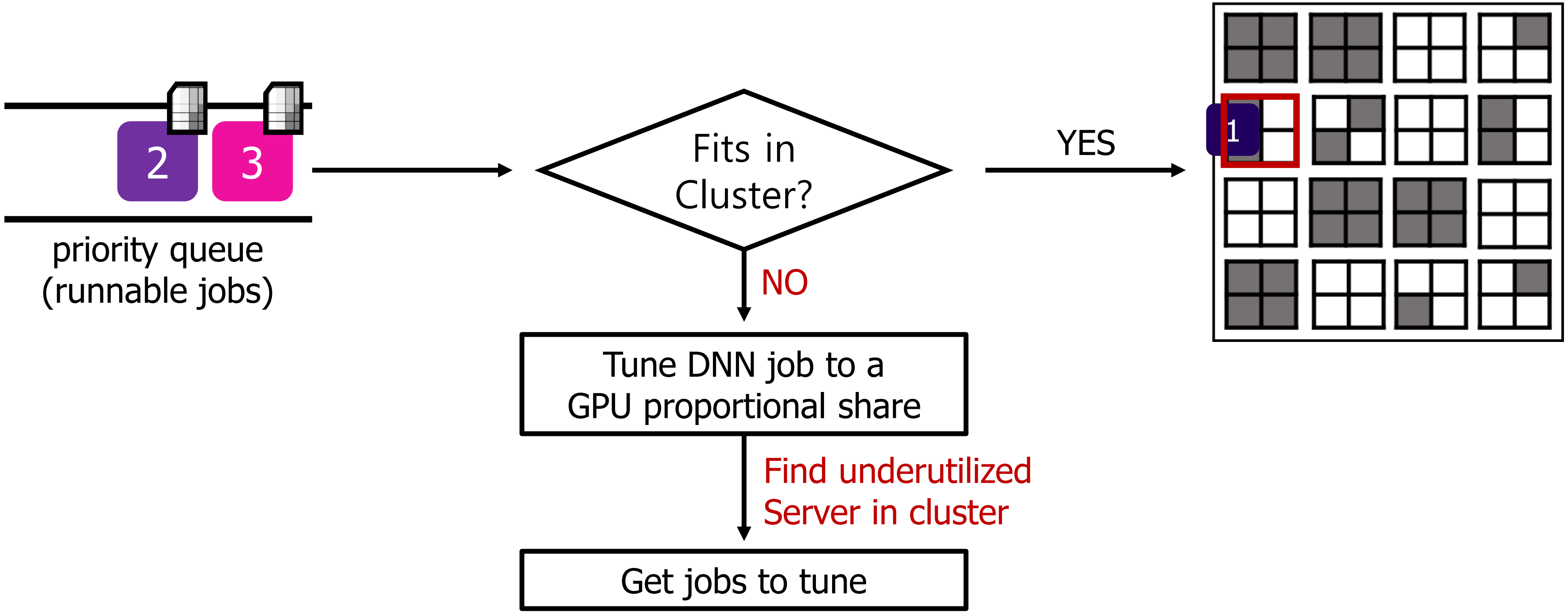
Synergy-TUNE scheduling

- Synergy identify the job that is allocated more than GPU-proportional share
- Then it switch the job to GPU-proportional share



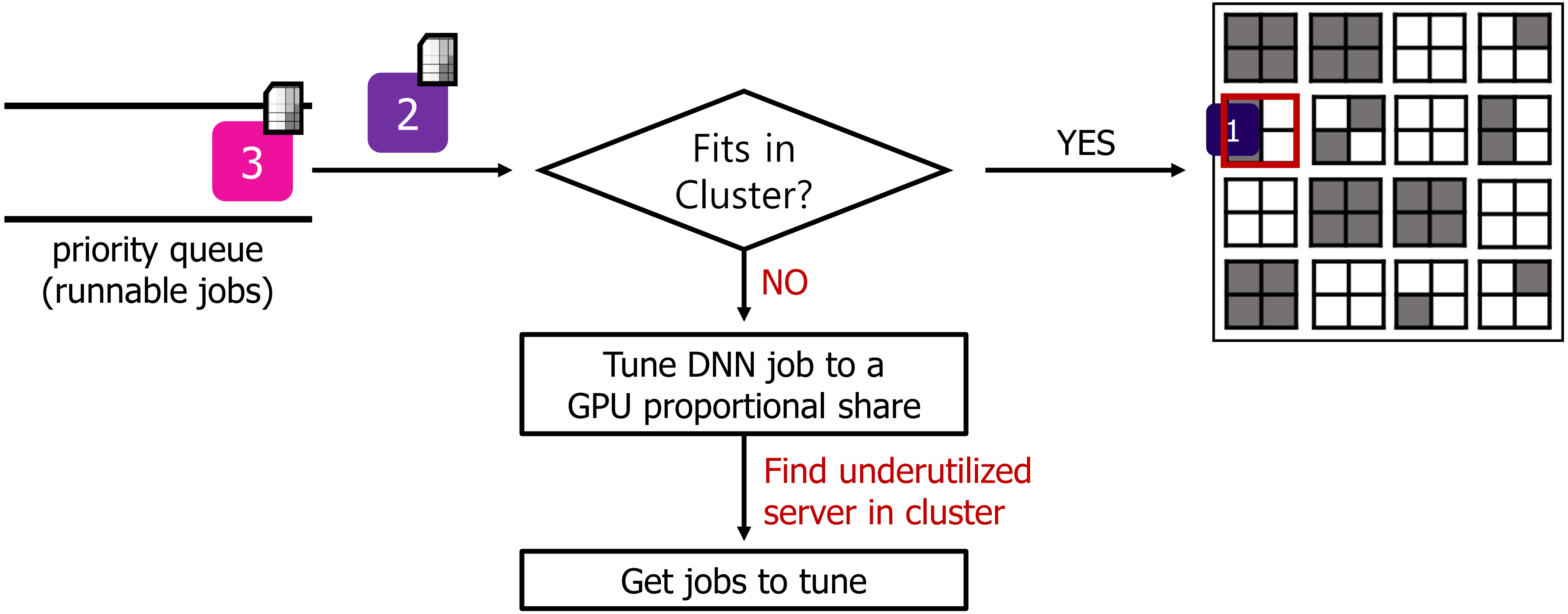
Synergy-TUNE scheduling

- Synergy identify the job that is allocated more than GPU-proportional share
- Then it switch it to GPU-proportional share



Synergy-TUNE scheduling

- Synergy identify the job that is allocated more than GPU-proportional share
- Then it switch it to GPU-proportional share



Continue until no more free GPUs/jobs in the cluster

Evaluation

- Experimental setup

Task	Model	Dataset
Image	Shufflenetv2 [58] AlexNet [32] Resnet18 [28] MobileNetv2 [50] ResNet50 [28]	ImageNet [49]
Language	GNMT [54] LSTM [47] Transformer-XL [16]	WMT16 [9] Wikitext-2 [36] Wikitext-103 [36]
Speech	M5 [15] DeepSpeech [27]	Free Music [17] LibriSpeech [45]

Table 4: **Models used in this work.**

- **Performed on 4 servers on a MS cluster**
- **10 DNN models (CNNs, RNNs, LSTMs)**
- **PyTorch 1.1.0**

- **24 core CPU**
- **500GB DRAM**
- **V100 GPU X 8**

Evaluation

- Physical cluster experiment

Policy (Metric)	Workload Split	Mechanism	Time (hrs) Deploy	Simulate
FIFO (Makespan)	60-30-10	Proportional	16	15.67
		Tune	11.6	11.33
		Opt	-	11.01
SRTF (Avg JCT)	30-60-10	Proportional	4.81	4.52
		Tune	3.21	3.19
		Opt	-	3.06
SRTF (99 Percentile JCT)	30-60-10	Proportional	17.32	16.85
		Tune	8.59	8.54
		Opt	-	8.21

Table 5: **Physical cluster experiments**. This table compares the makespan, average JCT, and 99th percentile JCT for two different traces; (1) a static trace using FIFO (2) a dynamic trace using SRTF. Synergy-TUNE improves makespan by 1.4×, average JCT by 1.5 × and 99th percentile JCT by 2×.

Synergy improves cluster objectives (makespan, Avg JCT, 99p JCT)

Evaluation

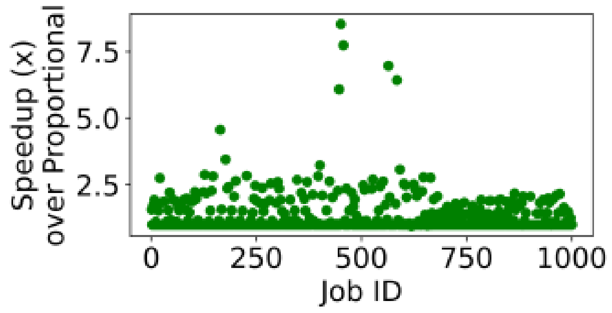
- Simulation with production traces

Policy	Avg JCT(hrs) SRTF	LAS	FIFO
GPU-prop.	30	32	71
Synergy	26	28	62

(a) Average JCT with Synergy

JCT (hrs)		Short	Long
Avg	Prop.	2	80
	Synergy	1.7	68
99p	Prop.	9	660
	Synergy	4	641

(b) Cluster metrics (SRTF)

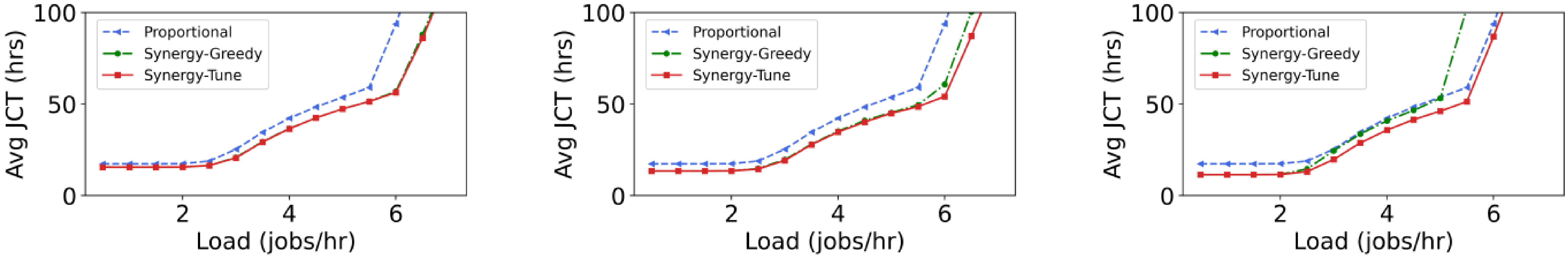


(c) JCT speedup across jobs

Figure 6: **Evaluation on Philly Trace**. On a real production trace, Synergy improves avg JCT across a range of scheduling policies over GPU-proportional scheduling. The JCT of individual jobs improves by upto 9× with Synergy.

Evaluation

- Impact of workload split (% of image, language, and speech models)



(a) Split=(20,70,10) (b) Split=(33,33,33) (c) Split=(50,0,50)

Figure 11: Evaluation of Synergy with varying workload split

Conclusion

- DNN training jobs in the cluster have different CPU and memory sensitivity
- Synergy profiles auxiliary resource requirement and performs workload-aware allocation
- It gives unutilized resources from one job to another, improving job performance

https://github.com/msr-fiddle/synergy

Synergy: Looking Beyond GPUs for DNN Scheduling on Multi-Tenant Clusters

Jayashree Mohan, Amar Phanishayee, and Janardhan Kulkarni, MS Research; Vijay Chidambaram, The University of Texas and VMware Research

USENIX OSDI'22

Thank You !

Presented by Yejin Han

yj0225@dankook.ac.kr