



DynamoLLM: Designing LLM Inference Clusters for Performance and Energy Efficiency

HPCA 2025

Jovan Stojkovic, Chaojie Zhang*, Íñigo Goiri*, Josep Torrellas, Esha Choukse*

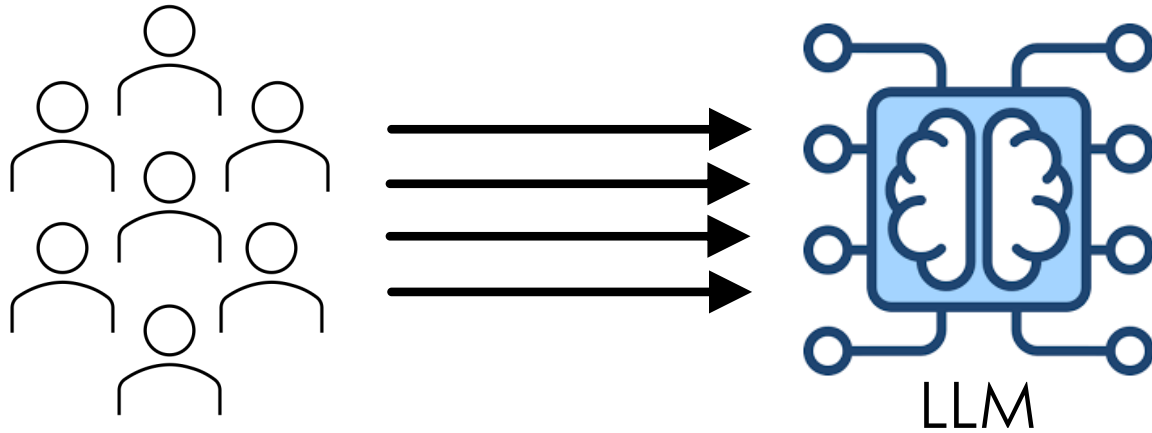
University of Illinois at Urbana-Champaign, *Azure Research Systems

LLM inference is emerging in the Cloud

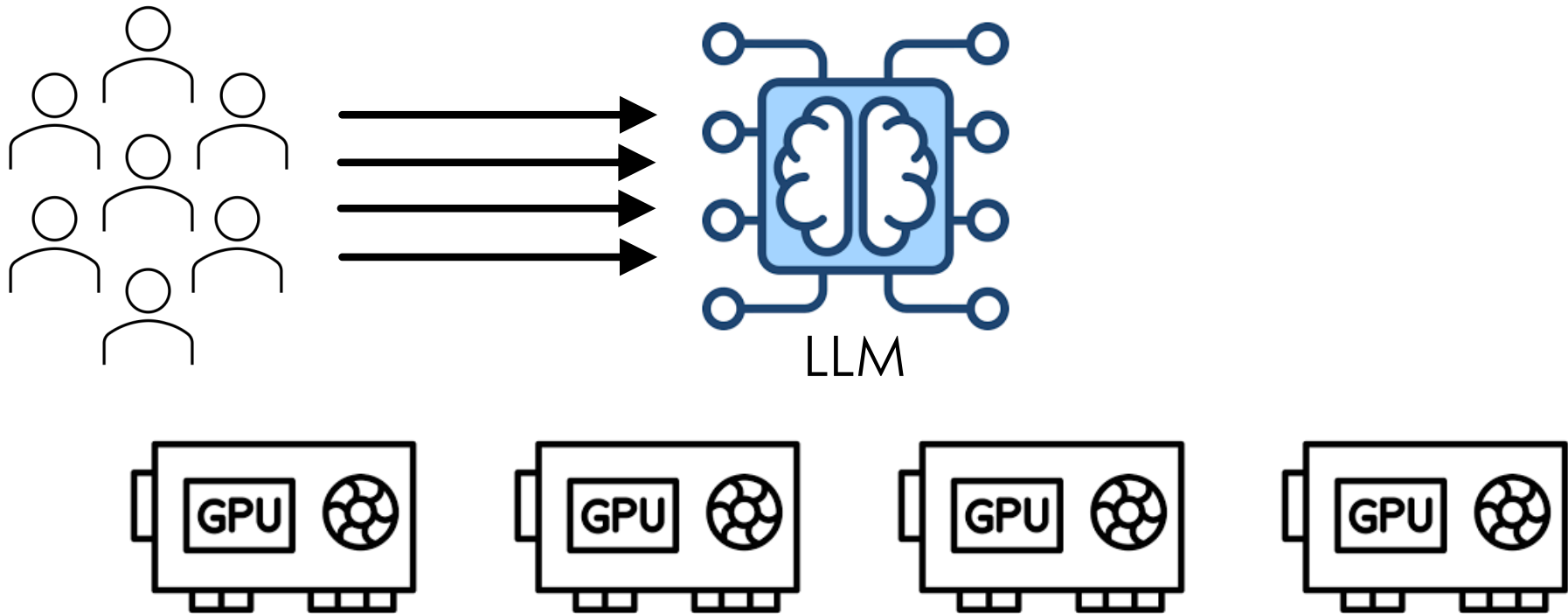
- Modern generative LLMs are turning ubiquitous
 - Use cases: programming, chat-bots, education, healthcare



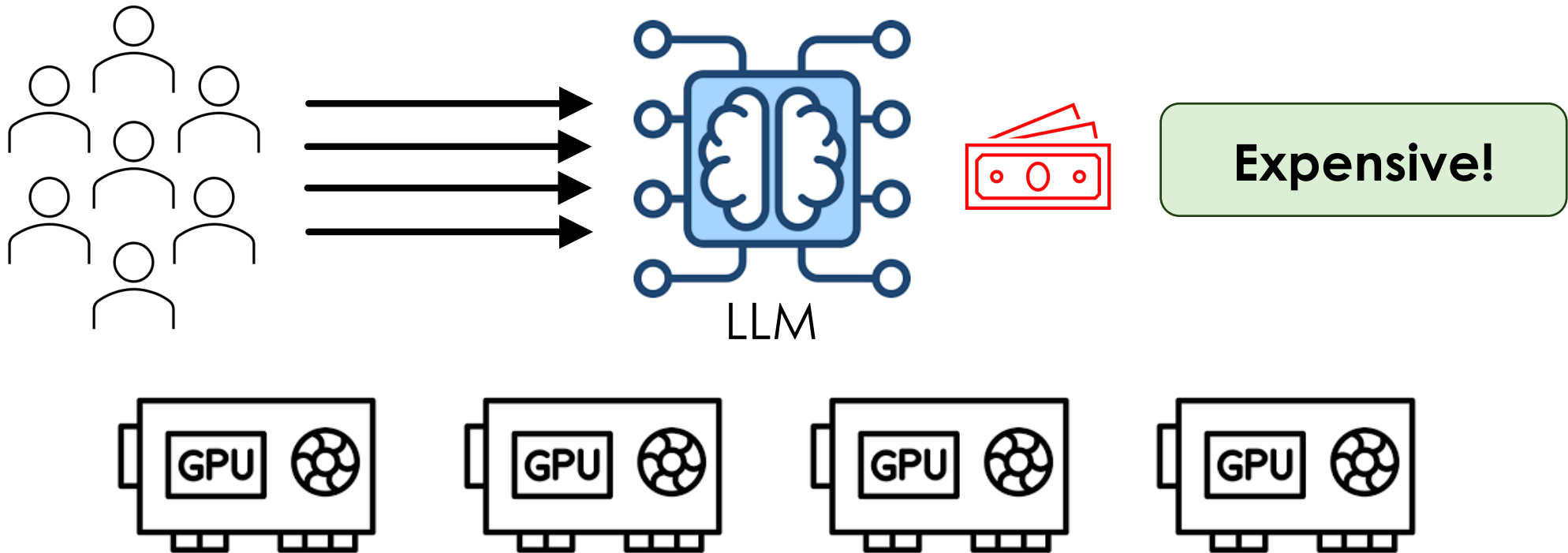
LLM inference stresses cloud infrastructure



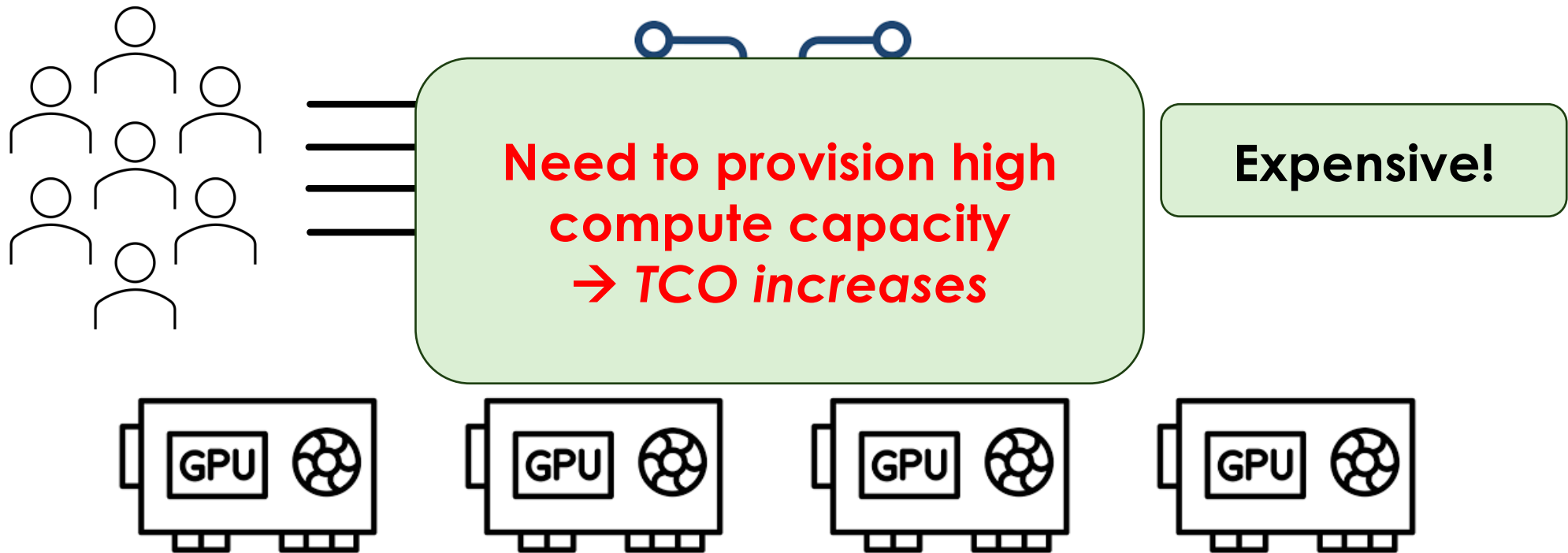
LLM inference stresses cloud infrastructure



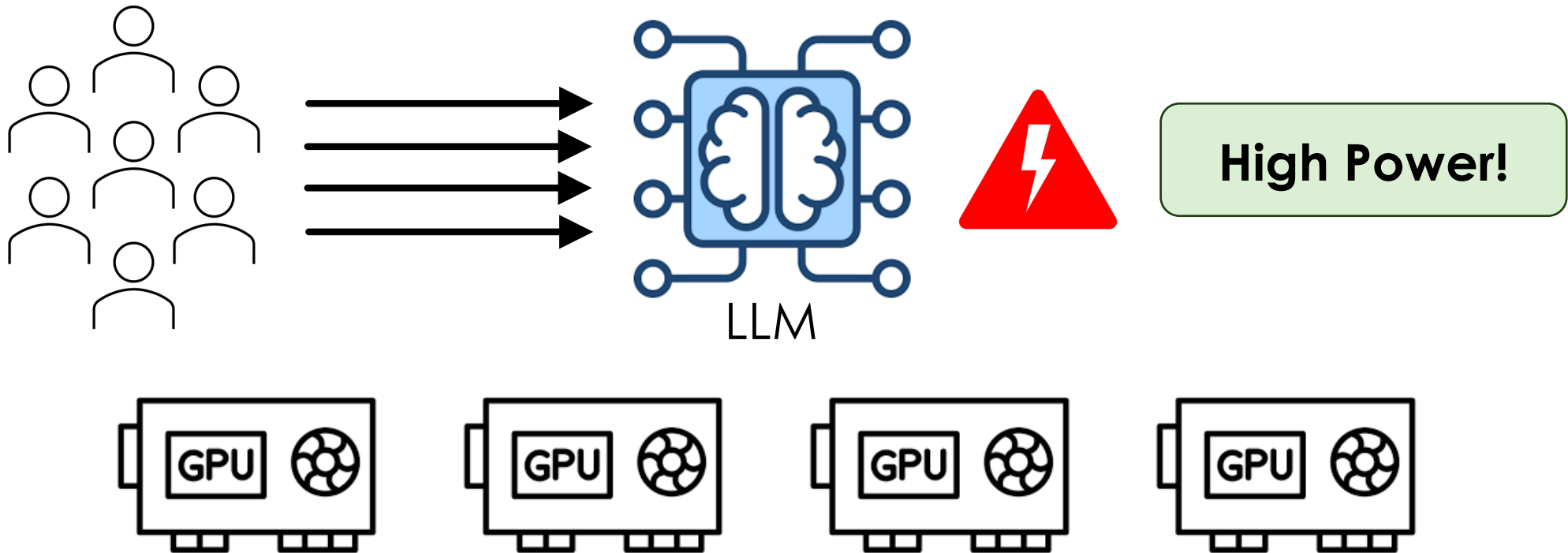
LLM inference stresses cloud infrastructure



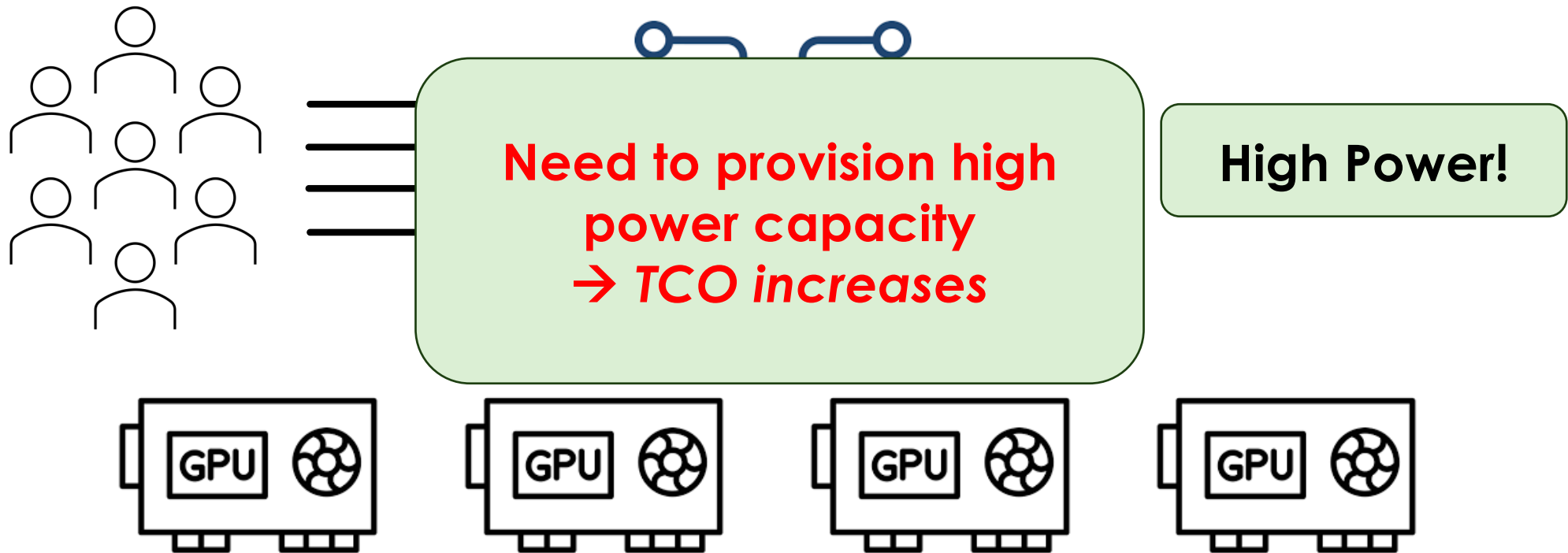
LLM inference stresses cloud infrastructure



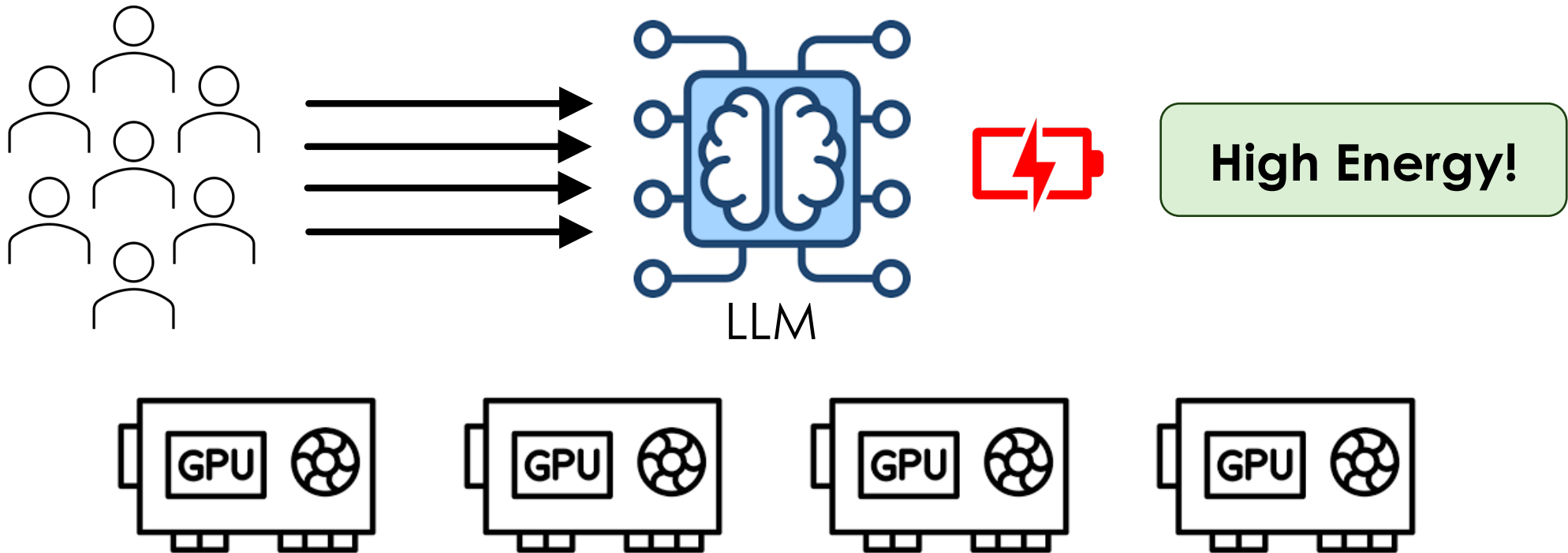
LLM inference stresses cloud infrastructure



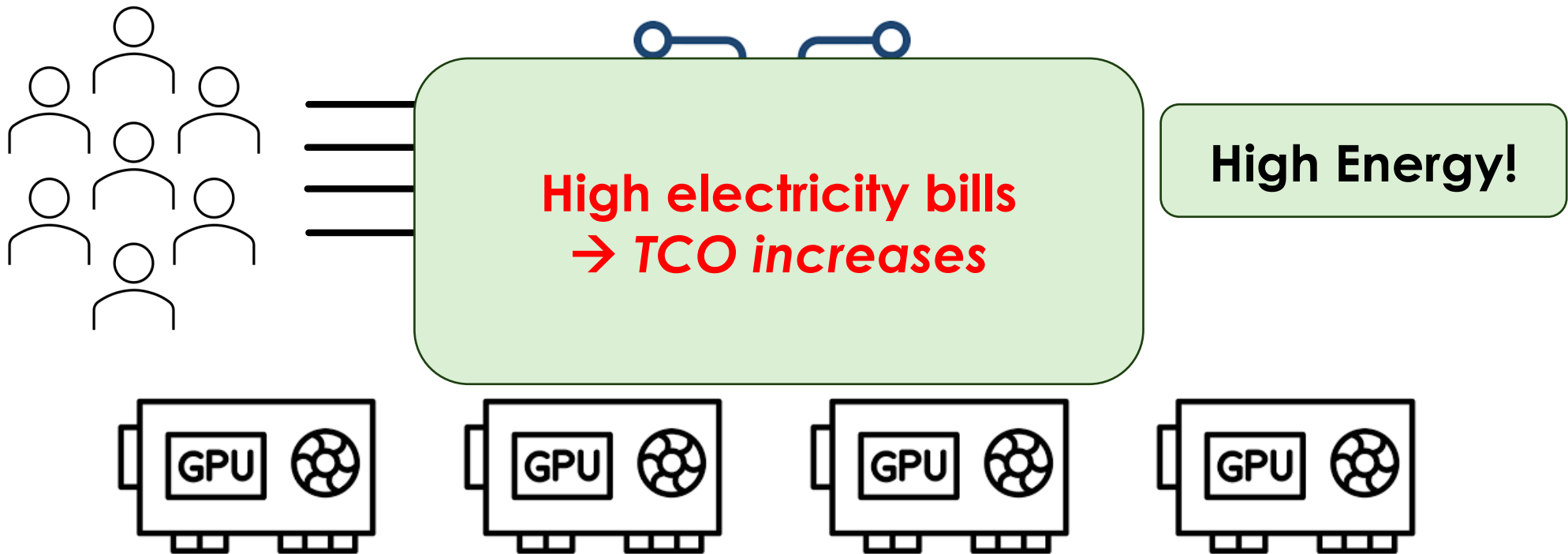
LLM inference stresses cloud infrastructure



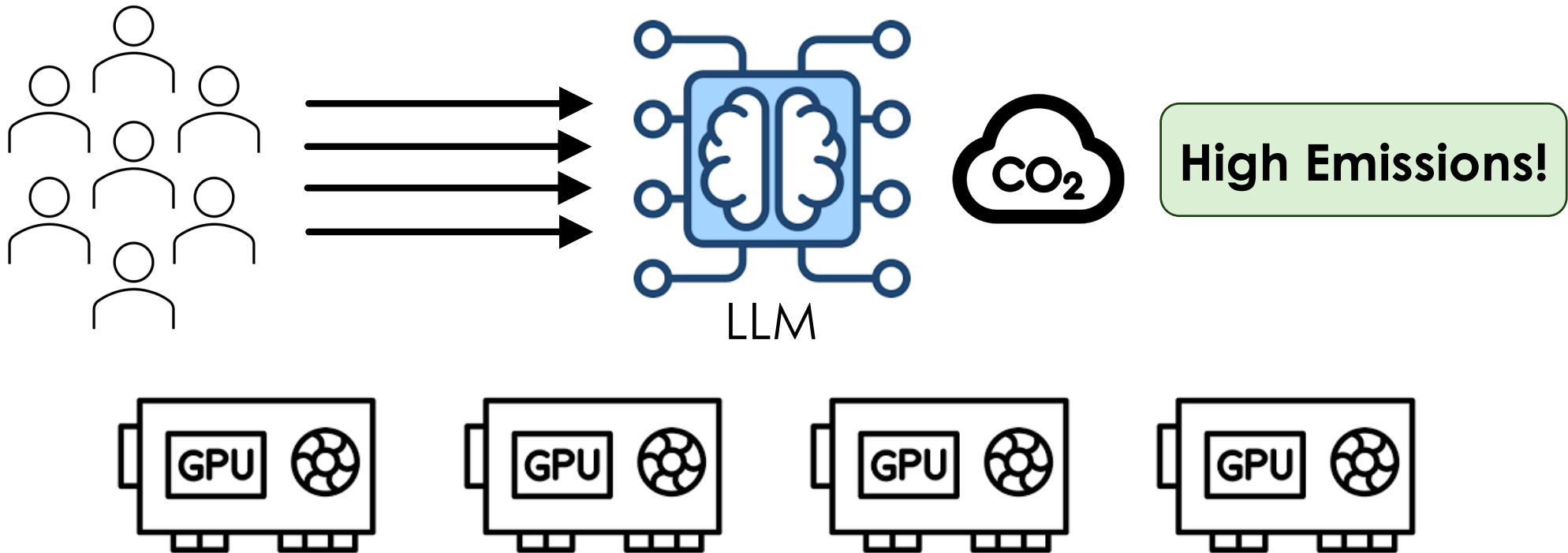
LLM inference stresses cloud infrastructure



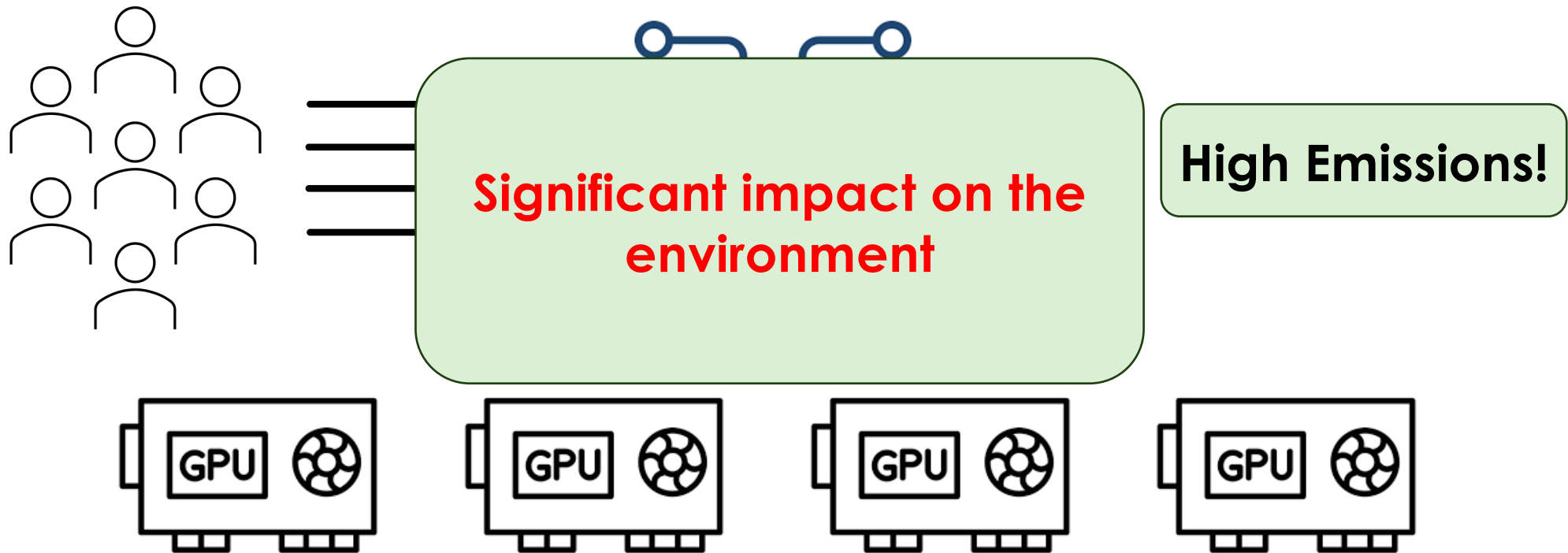
LLM inference stresses cloud infrastructure



LLM inference stresses cloud infrastructure

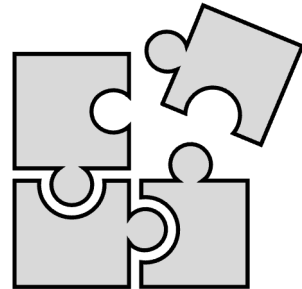


LLM inference stresses cloud infrastructure



How to tame the LLMs?

- Lots of work on performance, accuracy, scalability
- LLMs cause the datacenter cost, energy, and carbon emissions to skyrocket



**Energy efficiency of
LLMs is a missing piece
of a puzzle!**

Contributions

- Characterize energy properties of LLMs
- **DynamoLLM**: the first energy-management framework for LLM inference clusters
- Evaluate DynamoLLM at a large scale
 - 53% less energy and 38% less carbon emissions

How to tune LLMs?

Knob	Energy	Power	Perf	Quality

How to tune LLMs?

Knob	Energy	Power	Perf	Quality
Model Size ↓	↑	↑	↑	↓ ↓

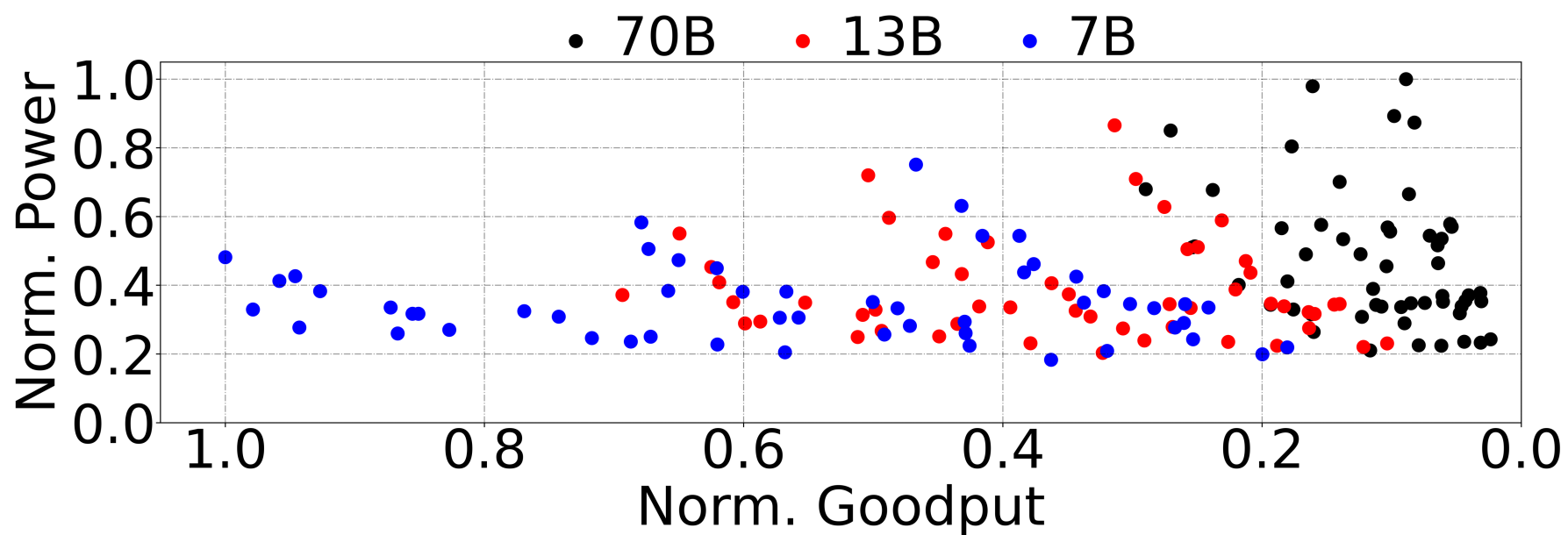
How to tune LLMs?

Knob	Energy	Power	Perf	Quality
Model Size ↓	↑	↑	↑	↓ ↓
Quantize ↓	↑	↑	↑	↓

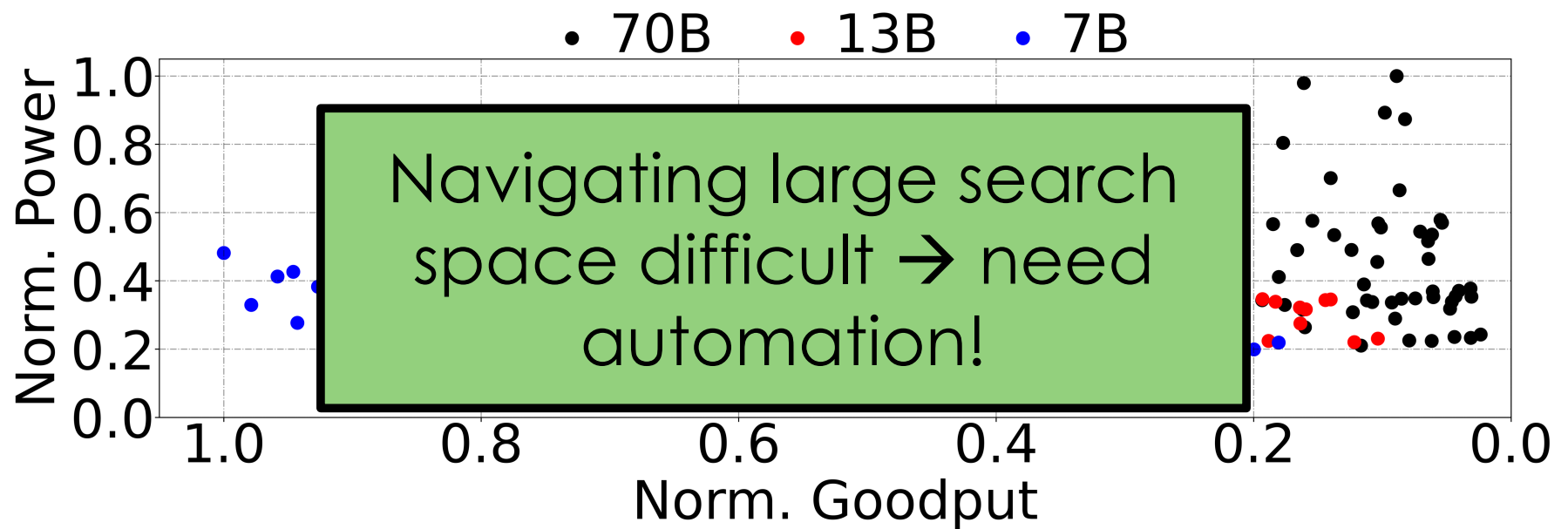
How to tune LLMs?

Knob	Energy	Power	Perf	Quality
Model Size ↓	↑	↑	↑	↓ ↓
Quantize ↓	↑	↑	↑	↓
Parallelism ↓	↓ ↑	↑	↓	
Frequency ↓	↓ ↑	↑	↓	
Batch size ↓	↓ ↑	↑	↓	

Large configuration search-space



Large configuration search-space

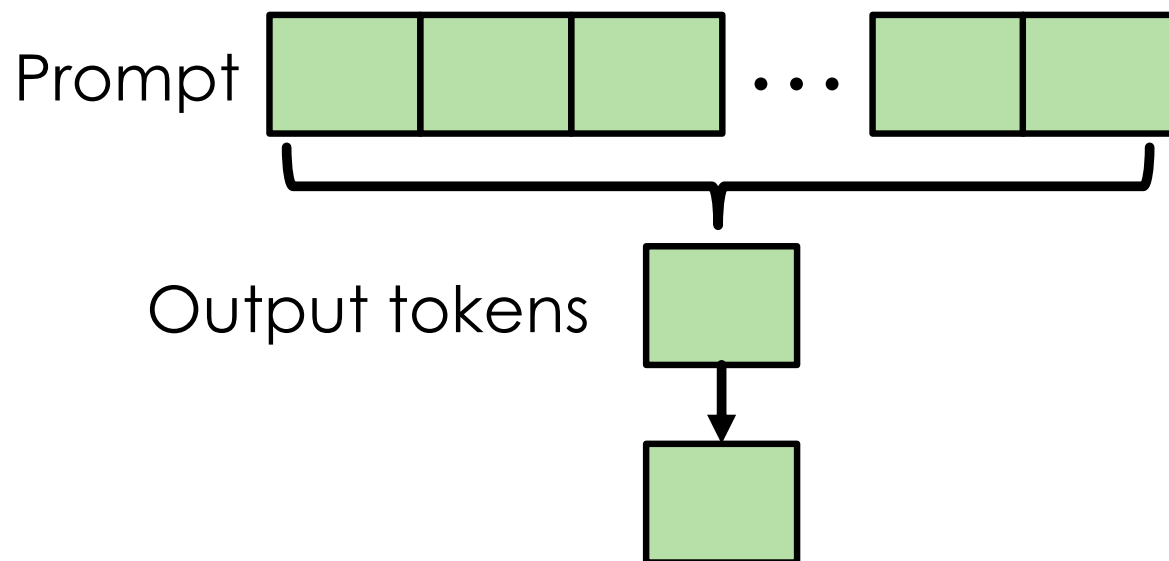


Goal: Make LLMs energy-efficient

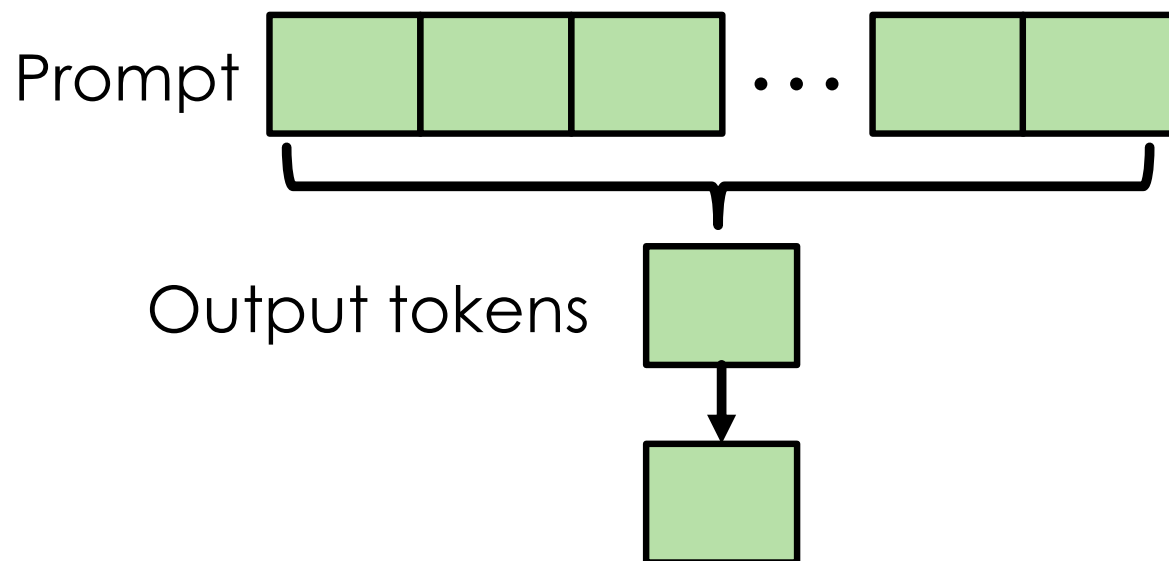
○ Challenges

1. Request heterogeneity
2. Workload dynamics
3. Reconfiguration overheads

Challenge #1: Request Heterogeneity

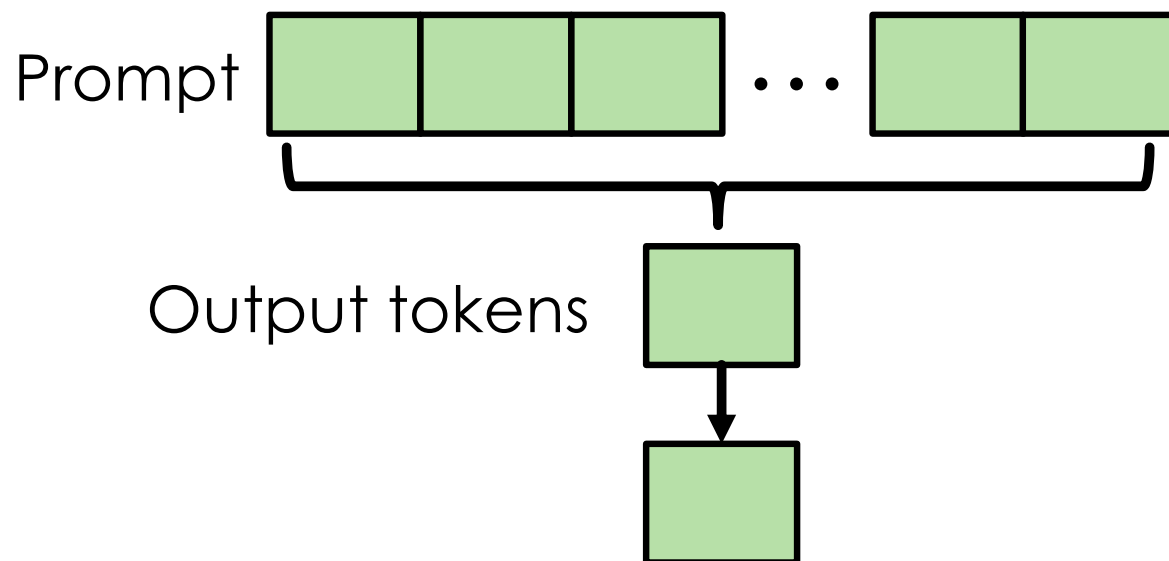


Challenge #1: Request Heterogeneity



**Long input,
short output**

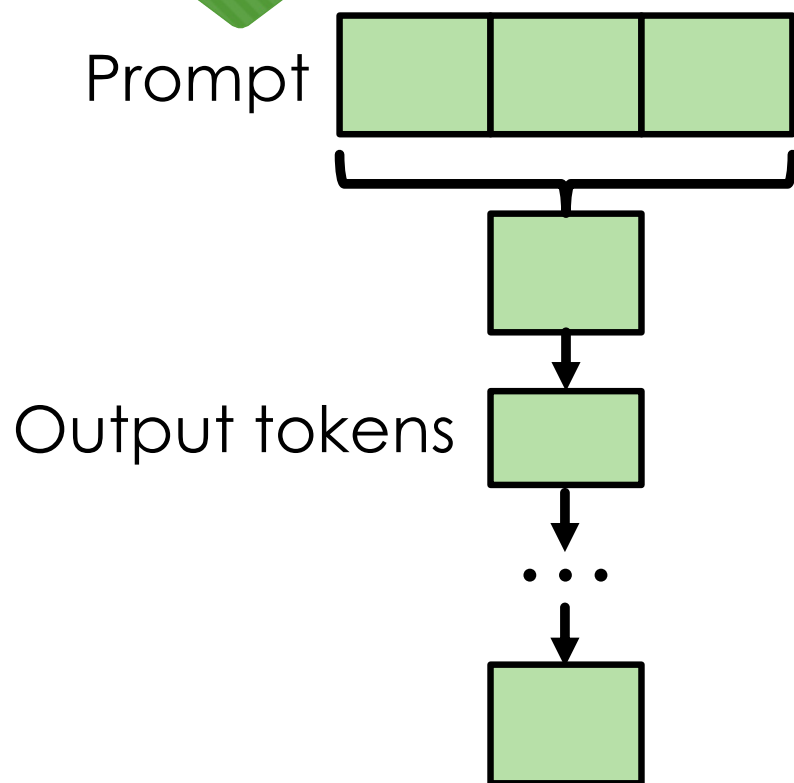
Challenge #1: Request Heterogeneity



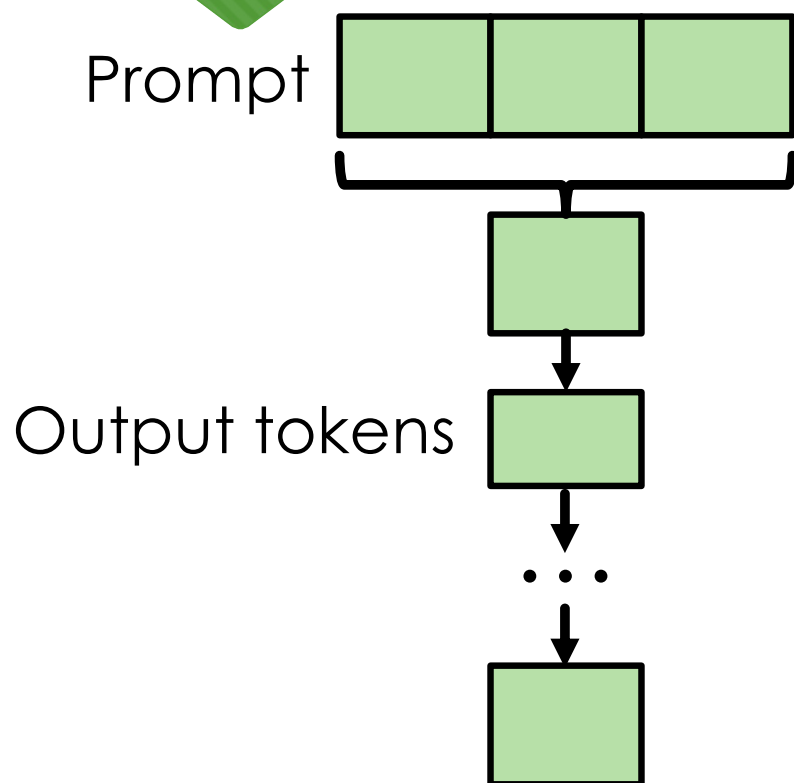
**Long input,
short output**

Compute-bound

Challenge #1: Request Heterogeneity

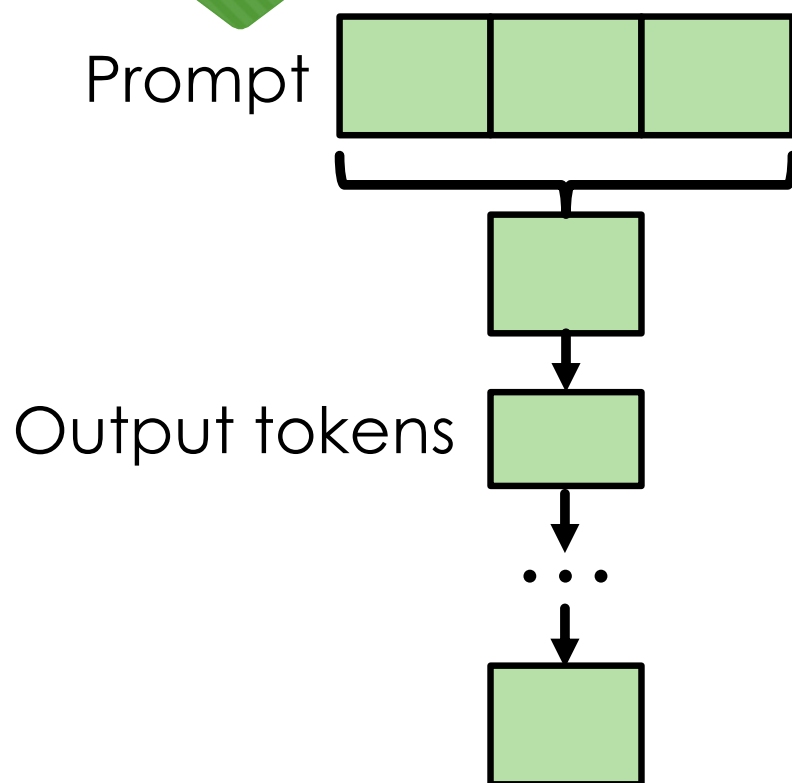


Challenge #1: Request Heterogeneity



**Short input,
long output**

Challenge #1: Request Heterogeneity



Short input,
long output

Memory-bound

Challenge #1: Request Heterogeneity

Shard	TP2				TP4				TP8			
Freq [GHz]	0.8	1.2	1.6	2	0.8	1.2	1.6	2.0	0.8	1.2	1.6	2.0
SS ←	Short input, short output											
SL												
LS												
LL ←	Long input, long output											

Challenge #1: Request Heterogeneity

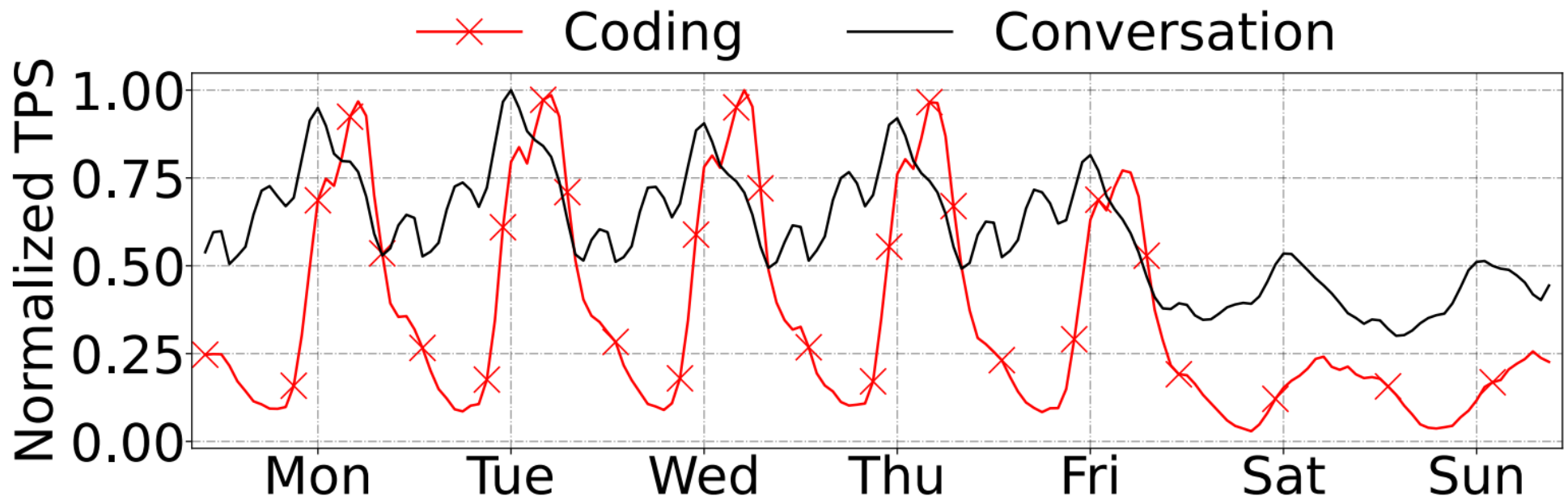
[illegible]

Challenge #1: Request Heterogeneity

Shard	TP2				TP4				TP8			
Freq [GHz]	0.8	1.2	1.6	2	0.8	1.2	1.6	2.0	0.8	1.2	1.6	2.0
SS		✓										
SL												
LS						✓						
LL											✓	

Challenge #1: Different request types require different configurations

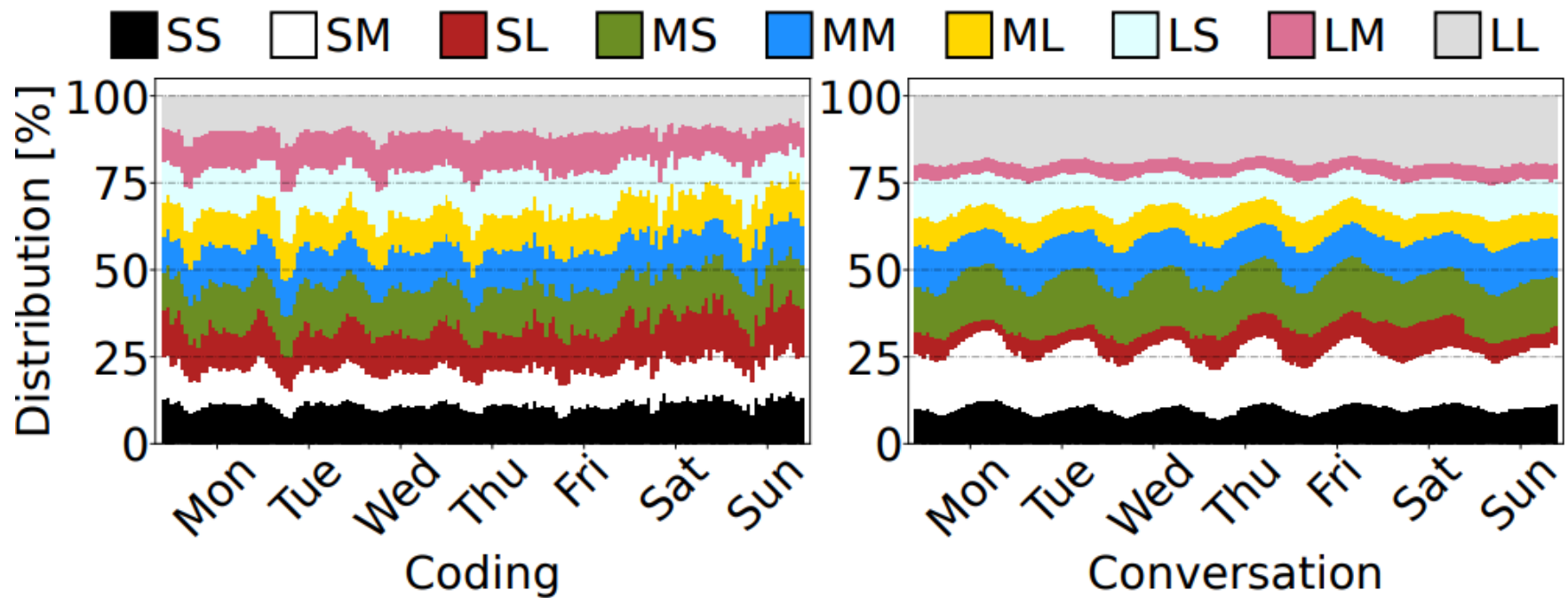
Challenge #2: Workload Dynamics



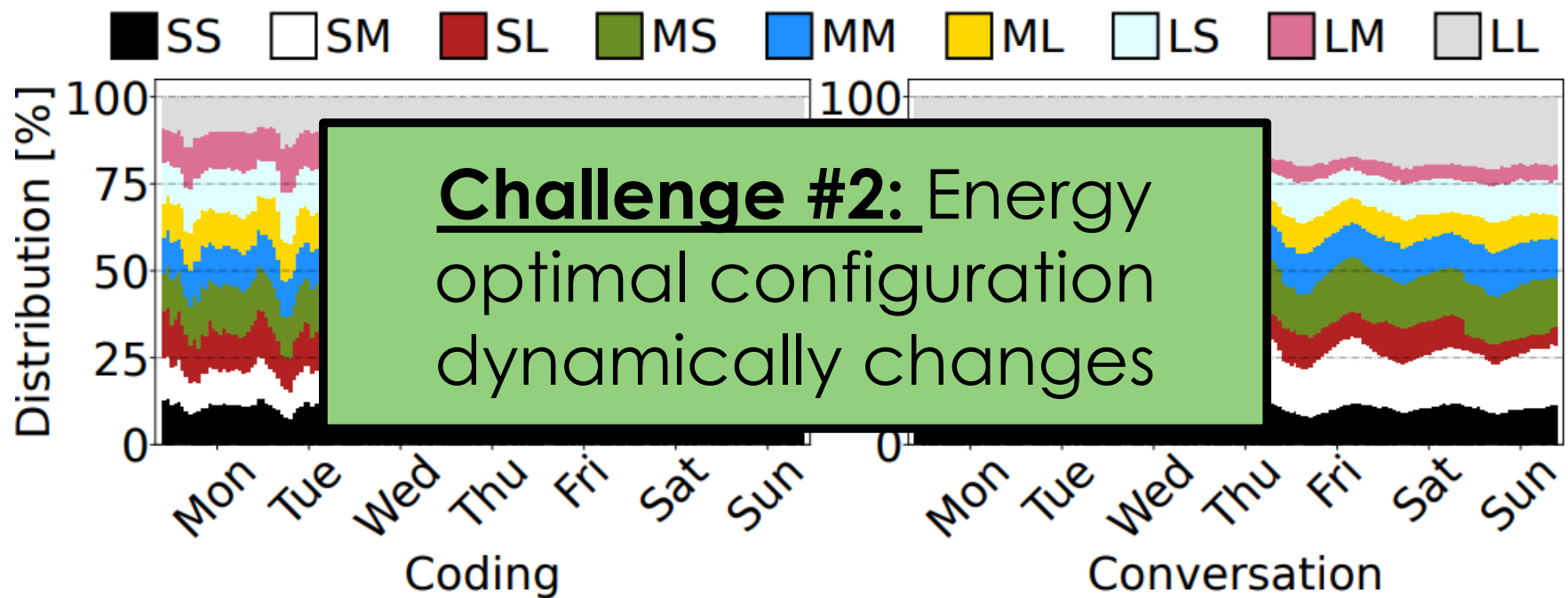
Challenge #2: Workload Dynamics

Sharding	TP4			
Frequency	0.8 GHz	1.2 GHz	1.6 GHz	2.0 GHz
Low Load		✓		
Med Load	✗		✓	
High Load	✗	✗		✓

Challenge #2: Workload Dynamics



Challenge #2: Workload Dynamics



Challenge #3: Re-configuration expensive

- Reconfiguration
 - i. Scale in/out
 - ii. Shard in/out
 - iii. Scale up/down

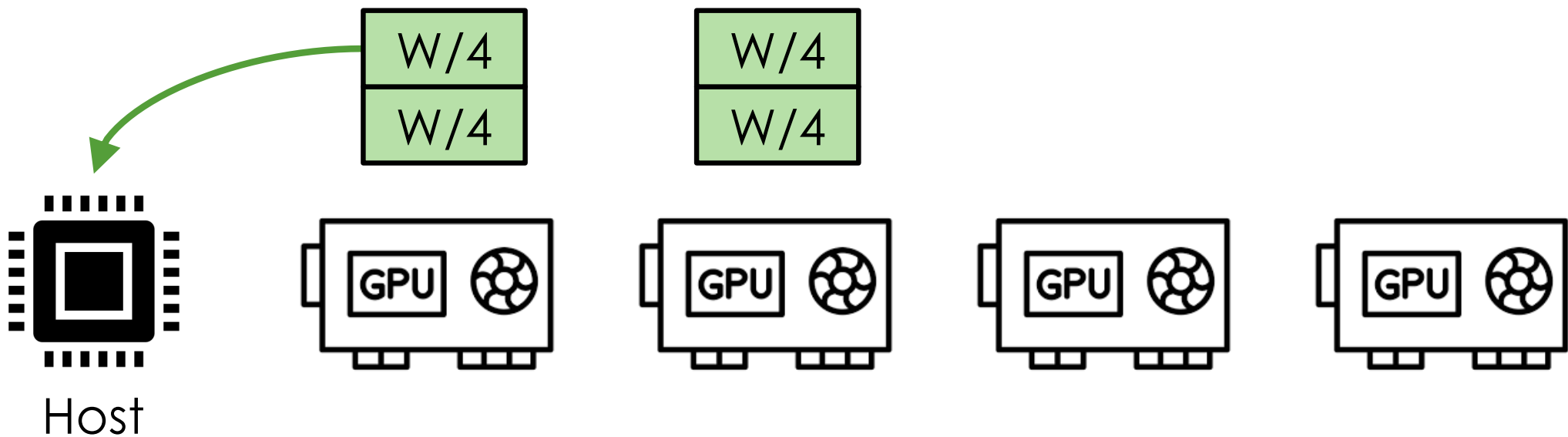
Challenge #3: Re-configuration expensive

i) Scale in/out costs

Overhead source	Time
Create an H100 VM	1-2 min
Init distributed environment	2 min
Download model weight	3 min
Setup inference engine	20 seconds
Install weights and KV cache on GPU	15 seconds
Total	6-8 min

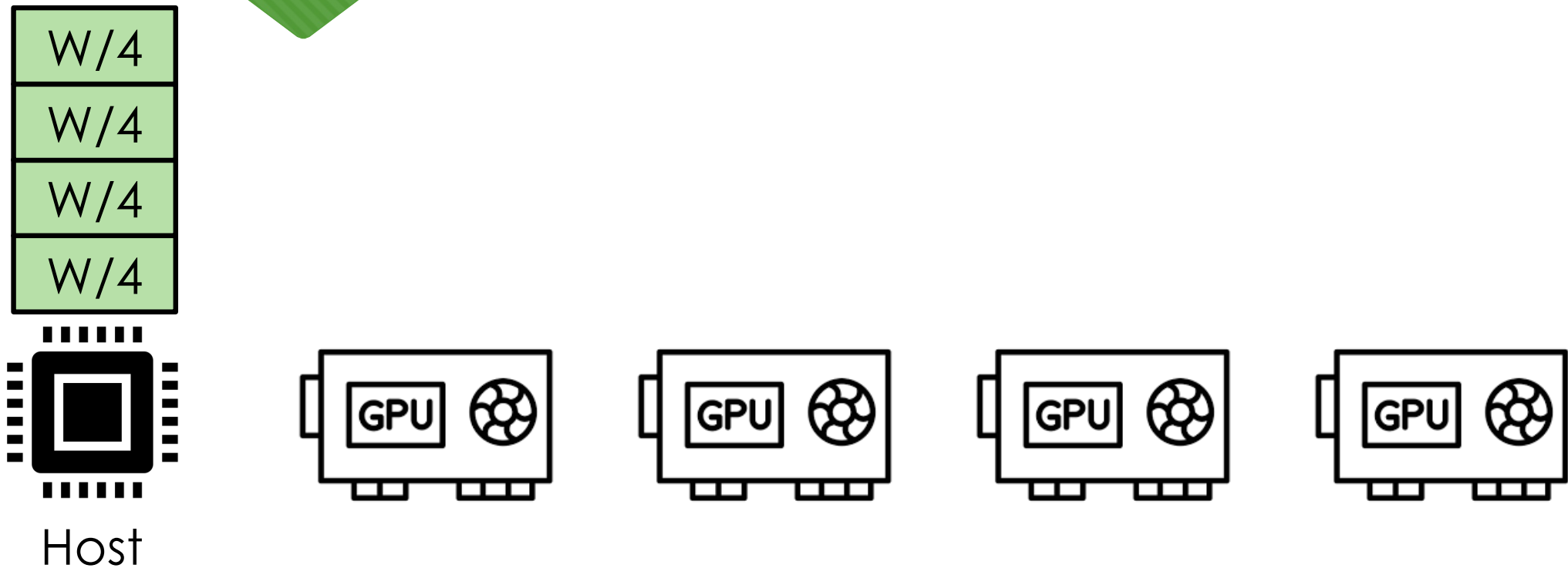
Challenge #3: Re-configuration expensive

ii) Shard in/out costs



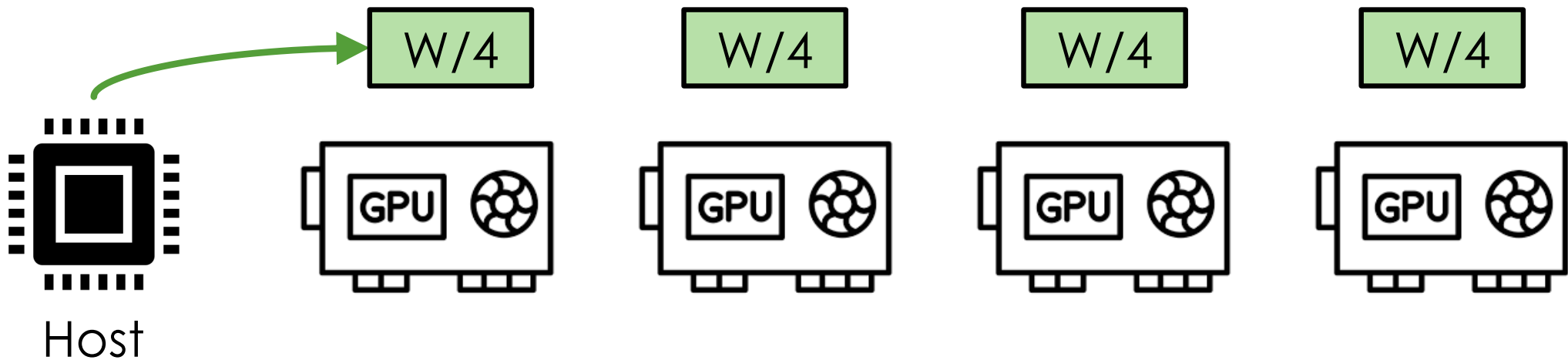
Challenge #3: Re-configuration expensive

ii) Shard in/out costs



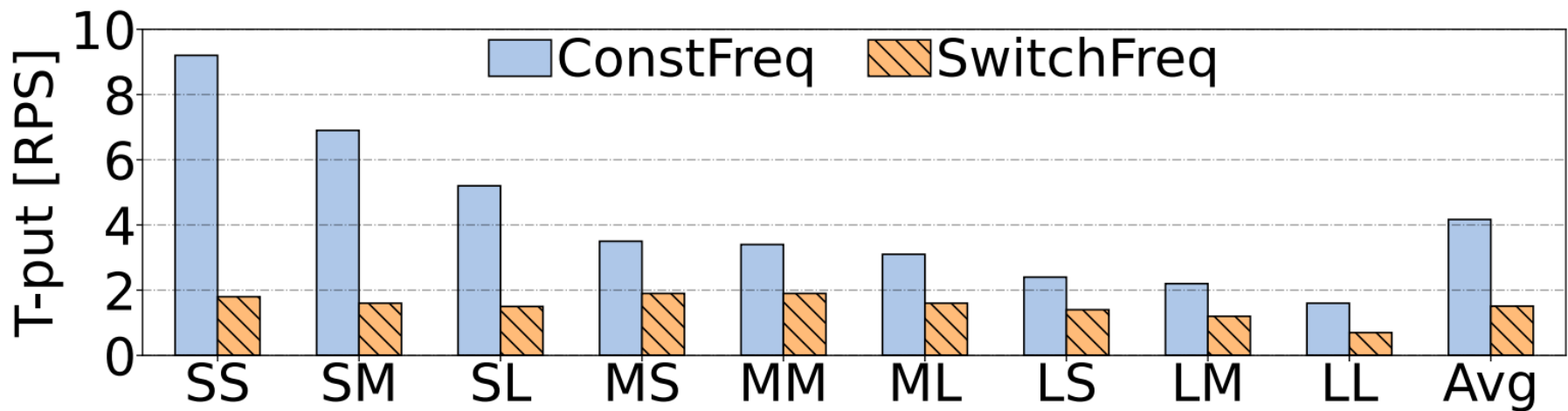
Challenge #3: Re-configuration expensive

ii) Shard in/out costs



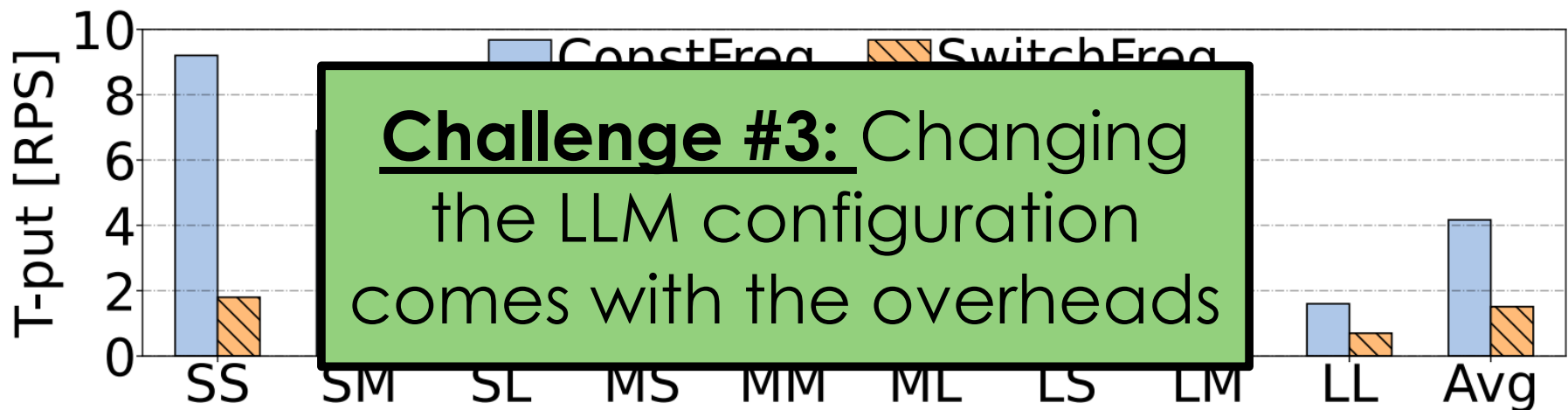
Challenge #3: Re-configuration expensive

iii) Scale up/down costs



Challenge #3: Re-configuration expensive

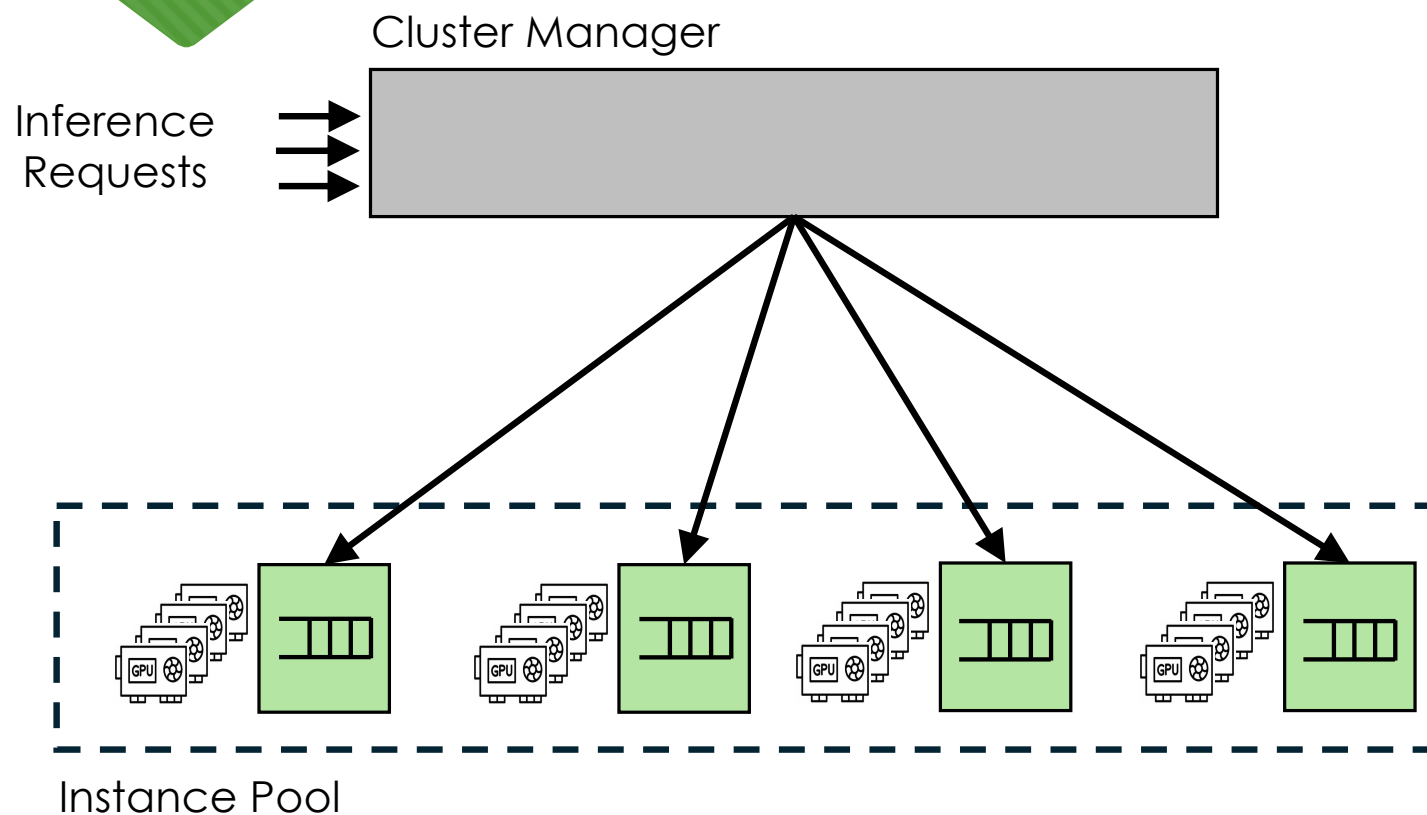
iii) Scale up/down costs



DynamoLLM: Energy-Management Framework for LLM Inference Clusters

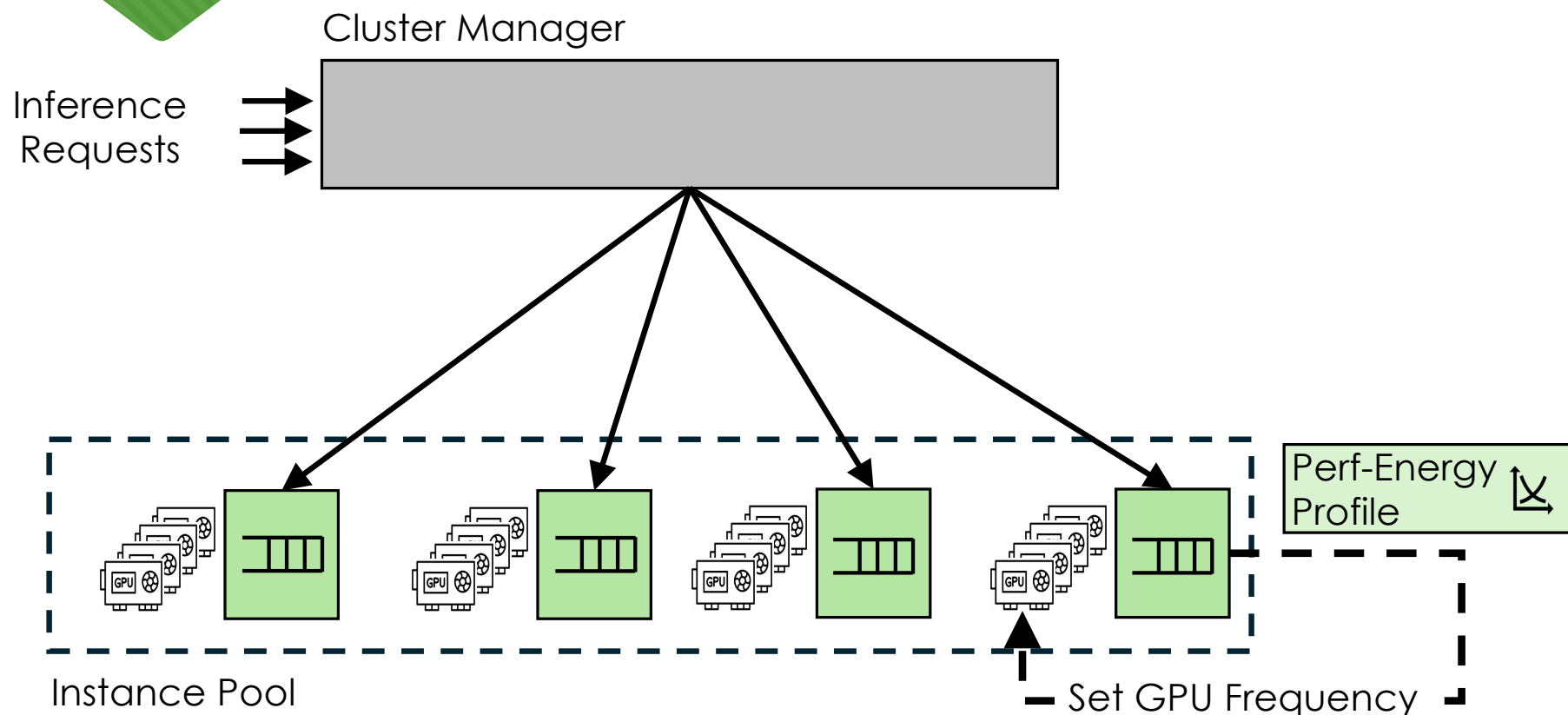
- DynamoLLM
 - i. Profile-driven energy configuration setting
 - ii. Instance pools for diverse workloads
 - iii. Hierarchical control for dynamic load

DynamoLLM: Energy-Management Framework for LLM Inference Clusters



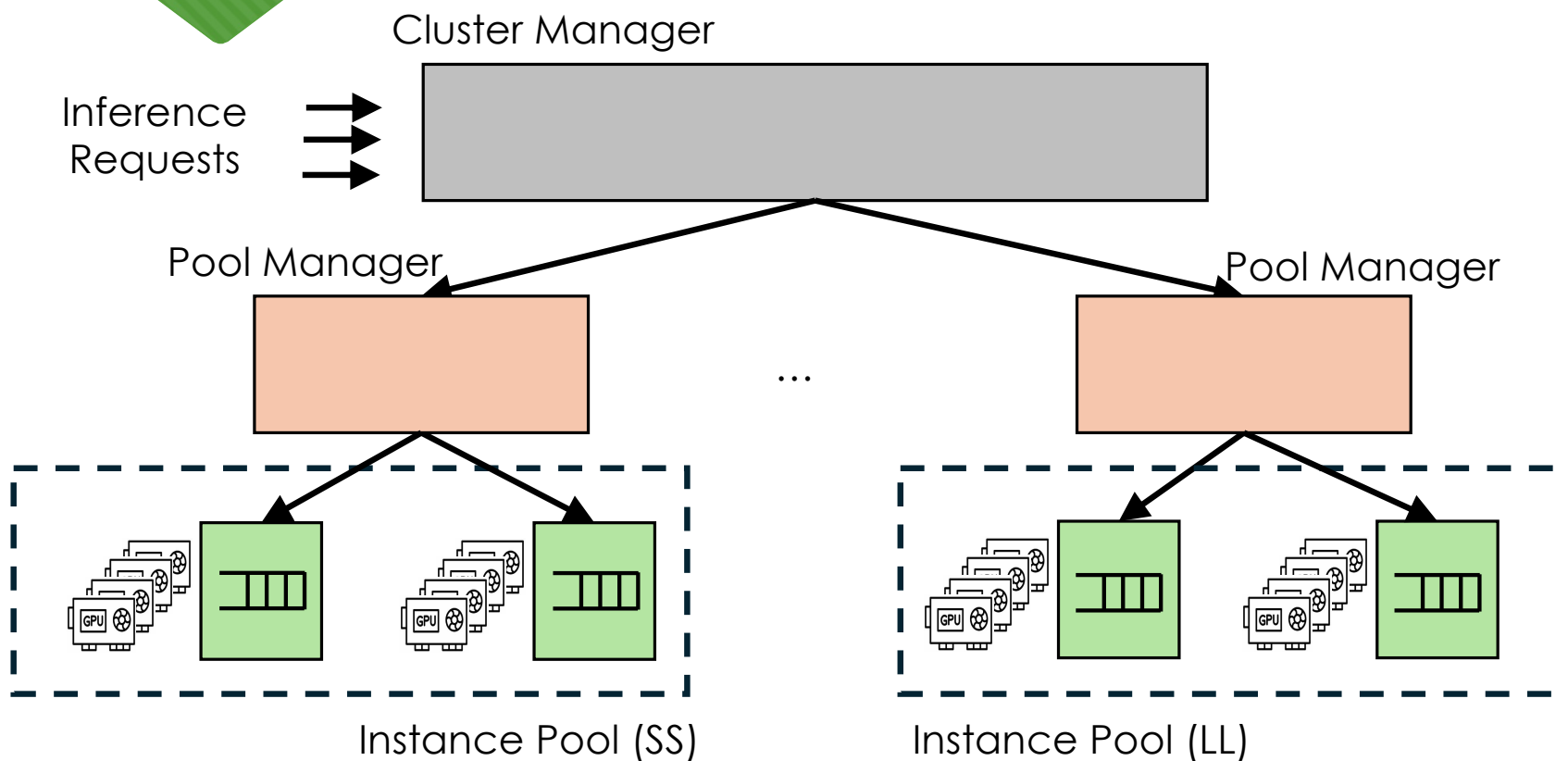
DynamoLLM:

1. Profile-Driven Energy Configuration Setting



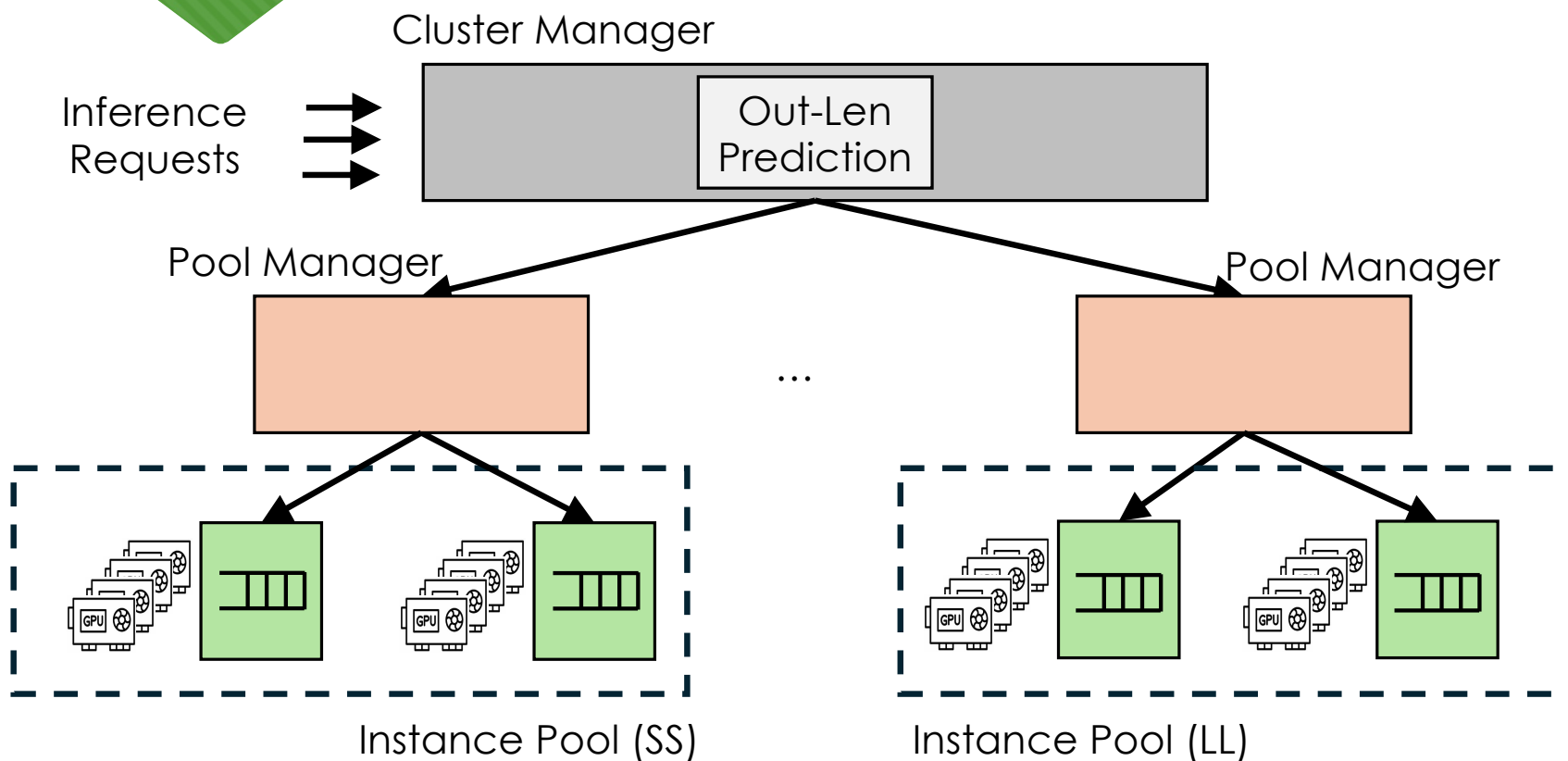
DynamoLLM:

2. Instance Pools for Diverse Workload



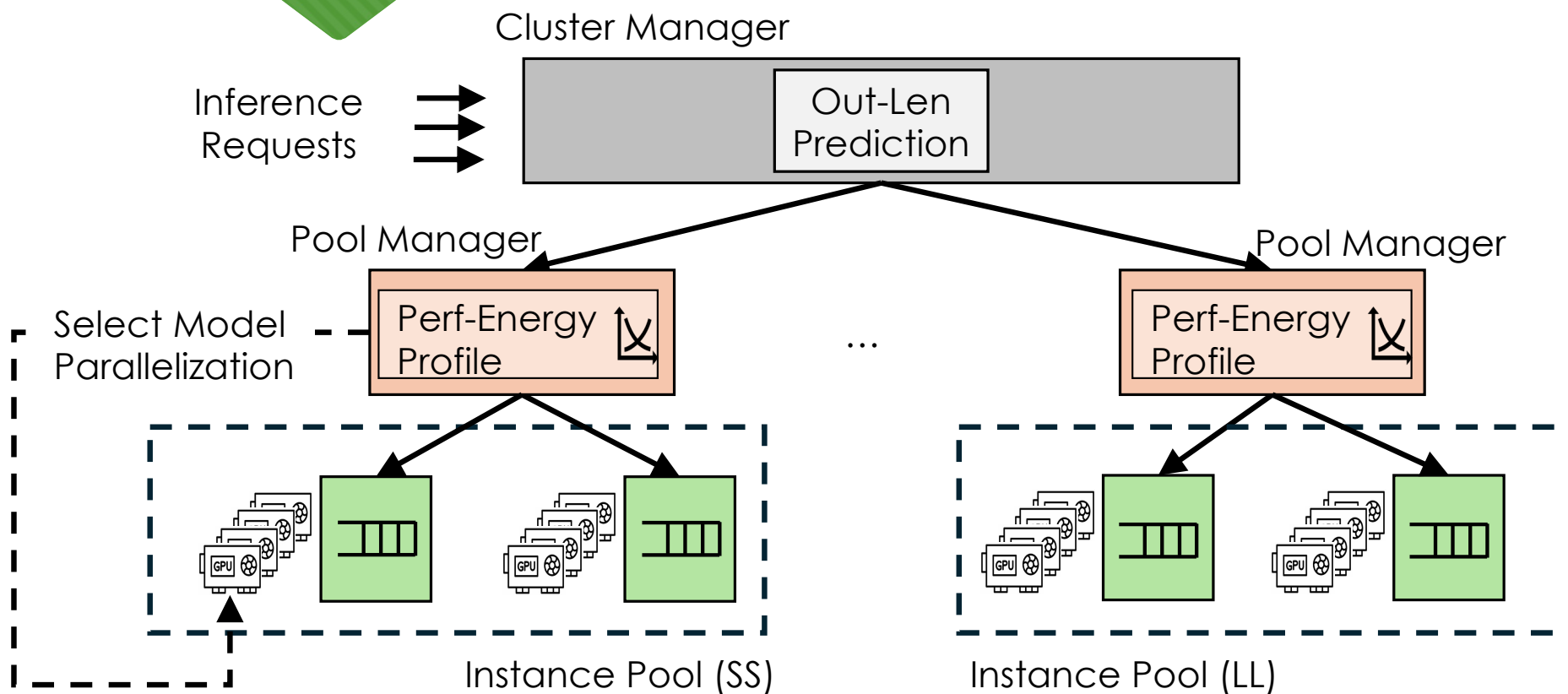
DynamoLLM:

2. Instance Pools for Diverse Workload



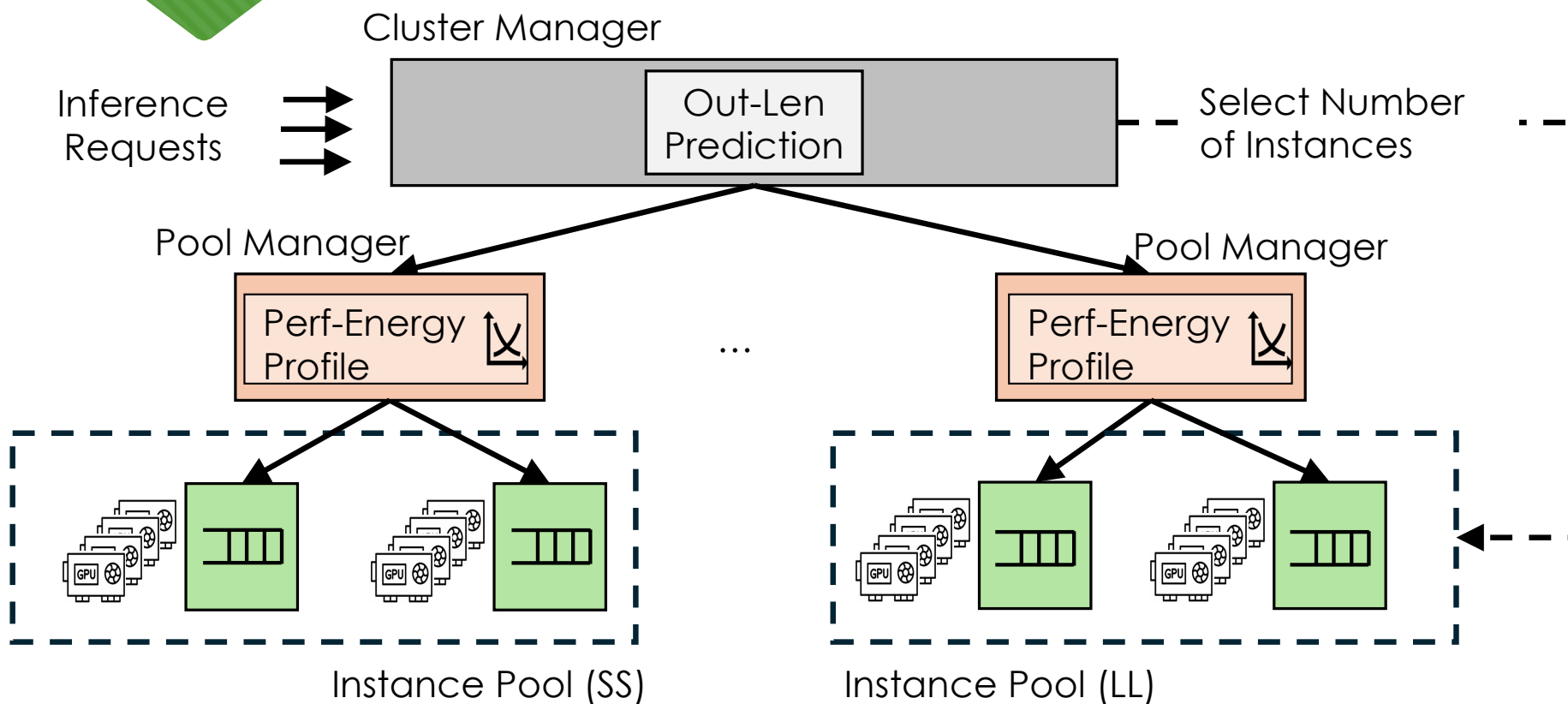
DynamoLLM:

2. Instance Pools for Diverse Workload



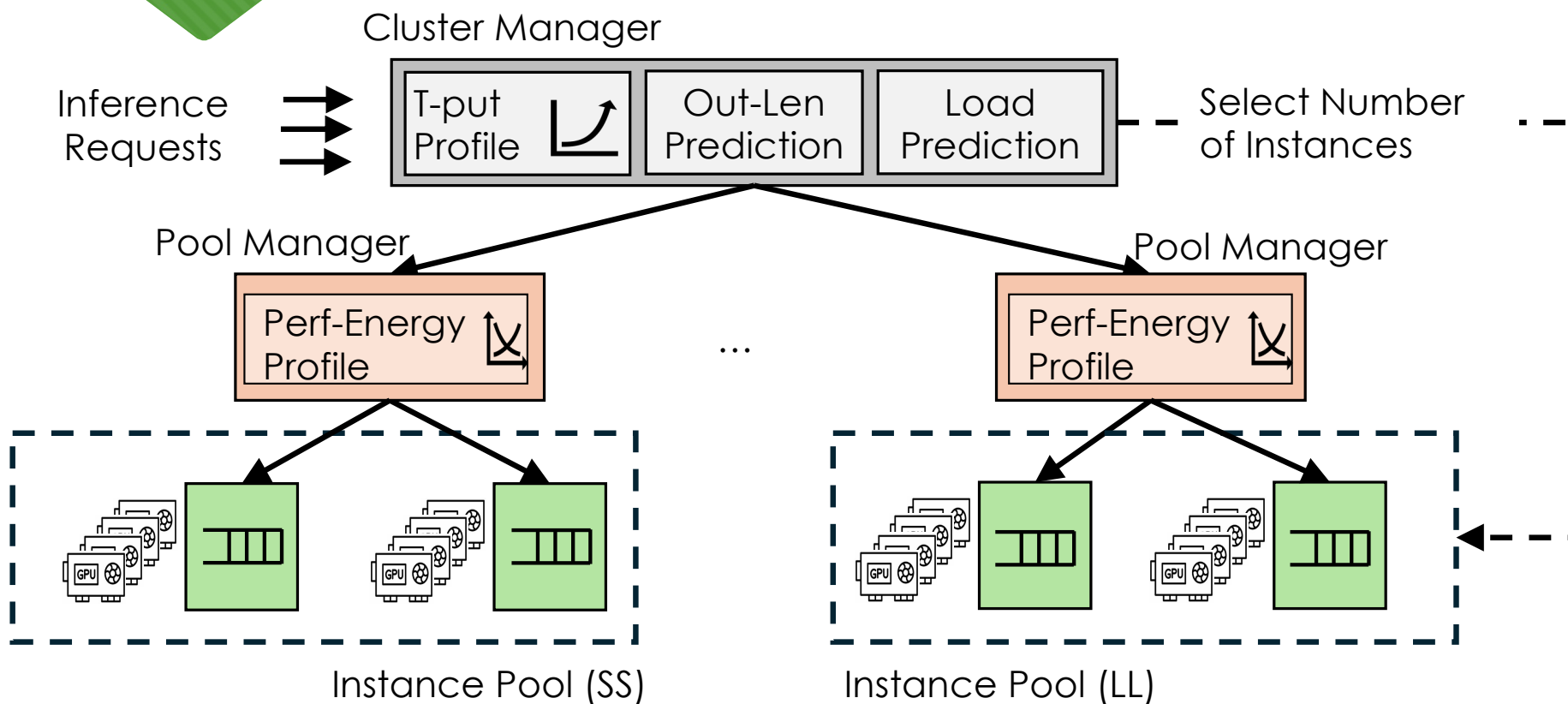
DynamoLLM:

3. Hierarchical Control for Dynamic Load



DynamoLLM:

3. Hierarchical Control for Dynamic Load



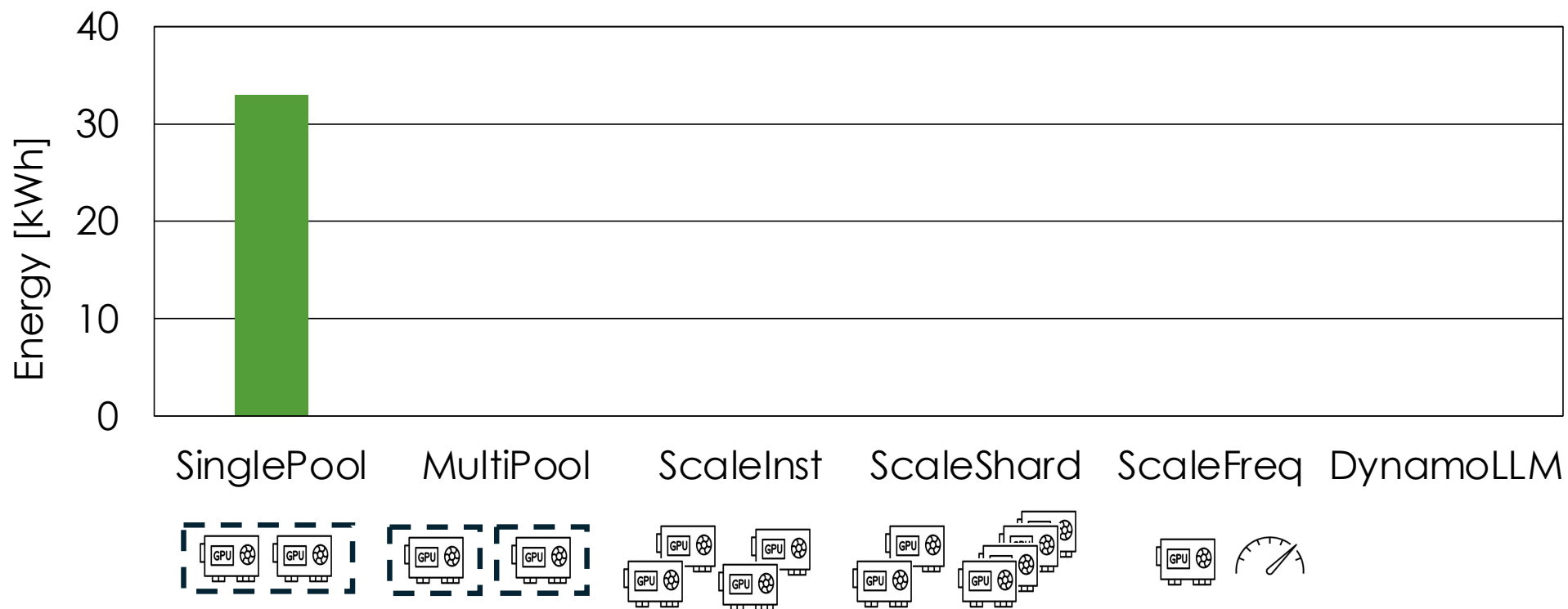
More in the paper

- MILP formulation for optimal energy-efficiency
- Techniques to reduce re-configuration overheads
 - Proactive VM creation
 - Graph-matching algorithm for re-sharding

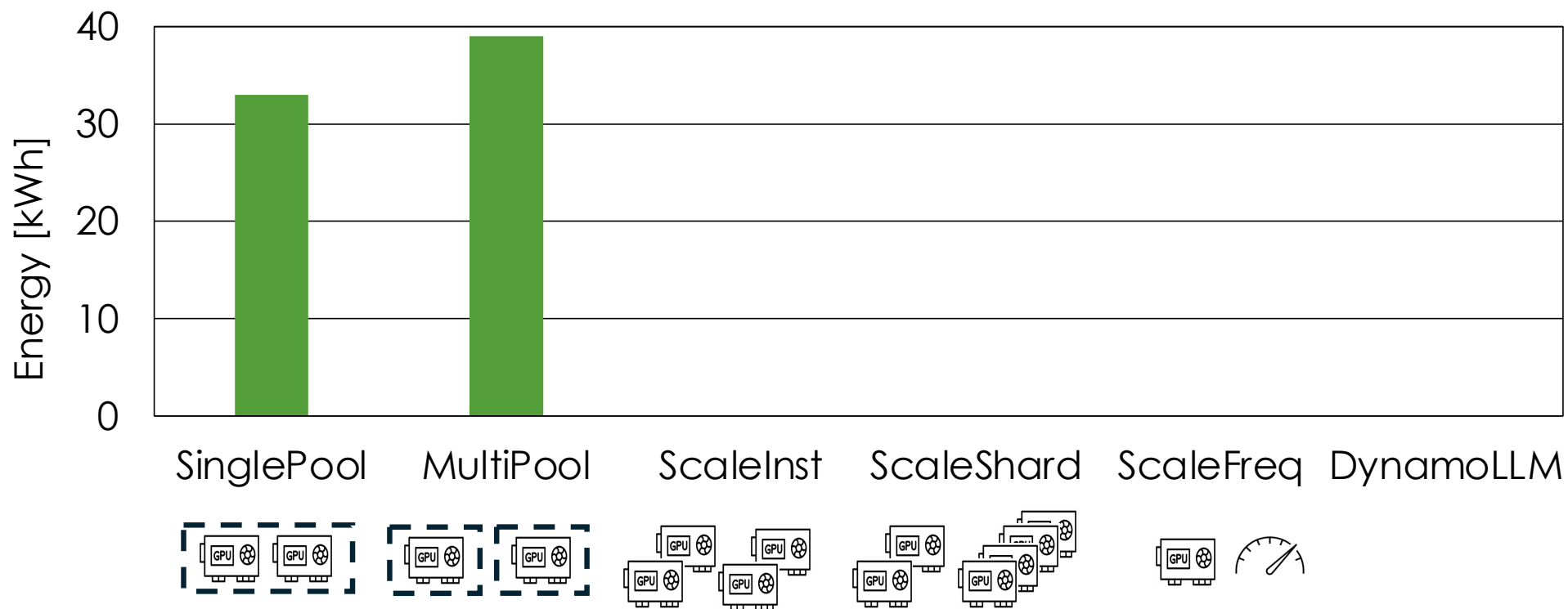
DynamoLLM Evaluation

- A peak hour (open-source) production traces from Azure
 - 12 x 8 H100 VMs for Baseline
- 1-week production traces from Azure
 - Simulate 40 x 8 H100 VMs for Baseline
- Llama2-70B model

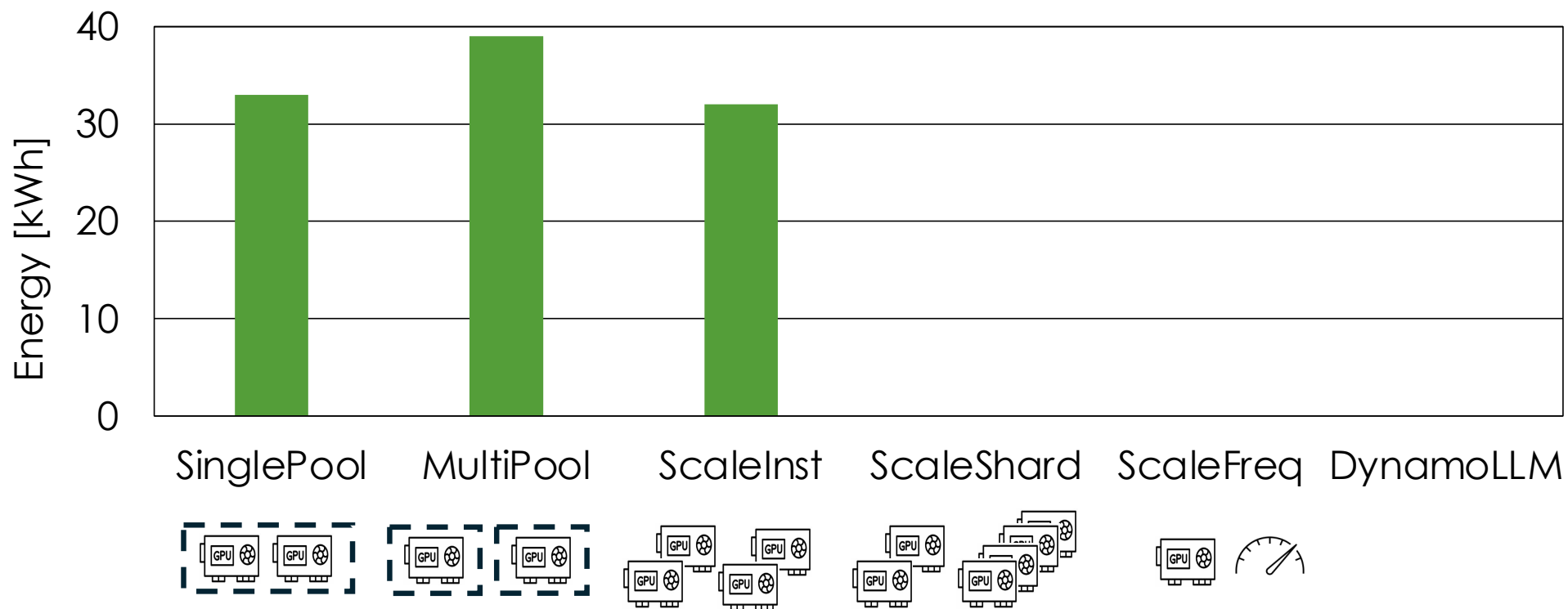
DynamoLLM significantly reduces energy!



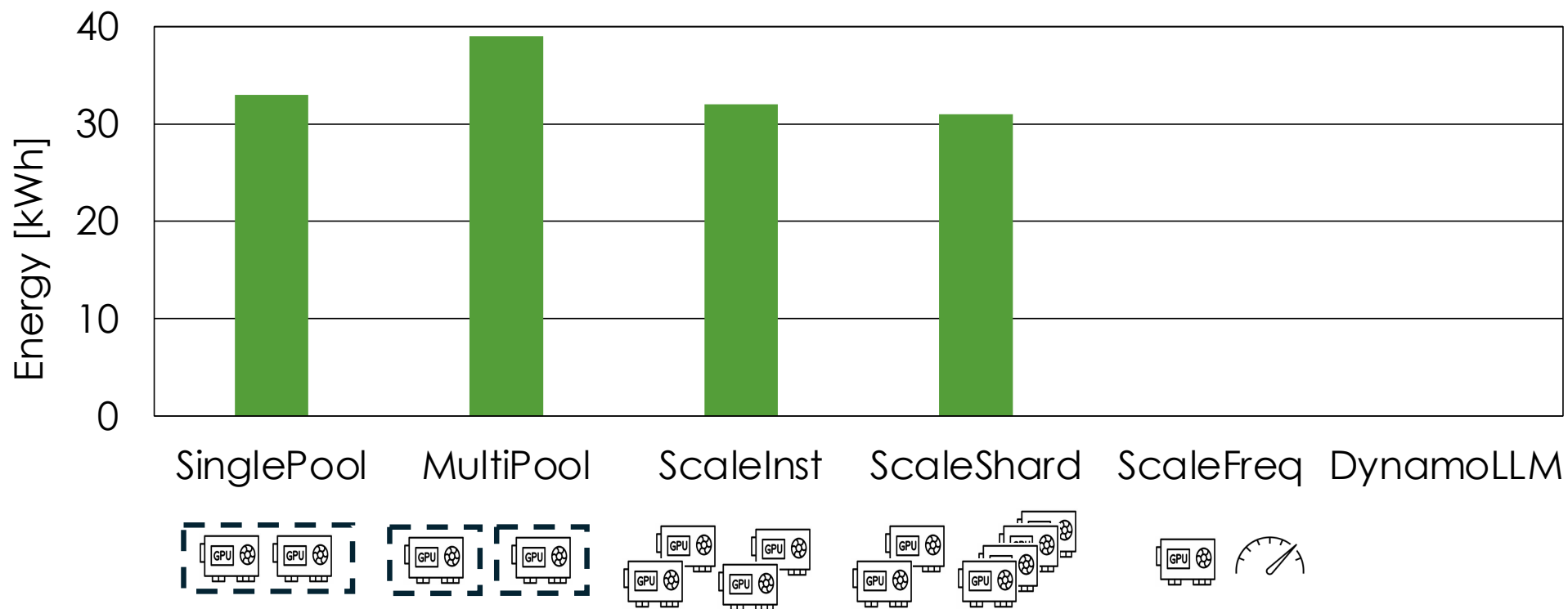
DynamoLLM significantly reduces energy!



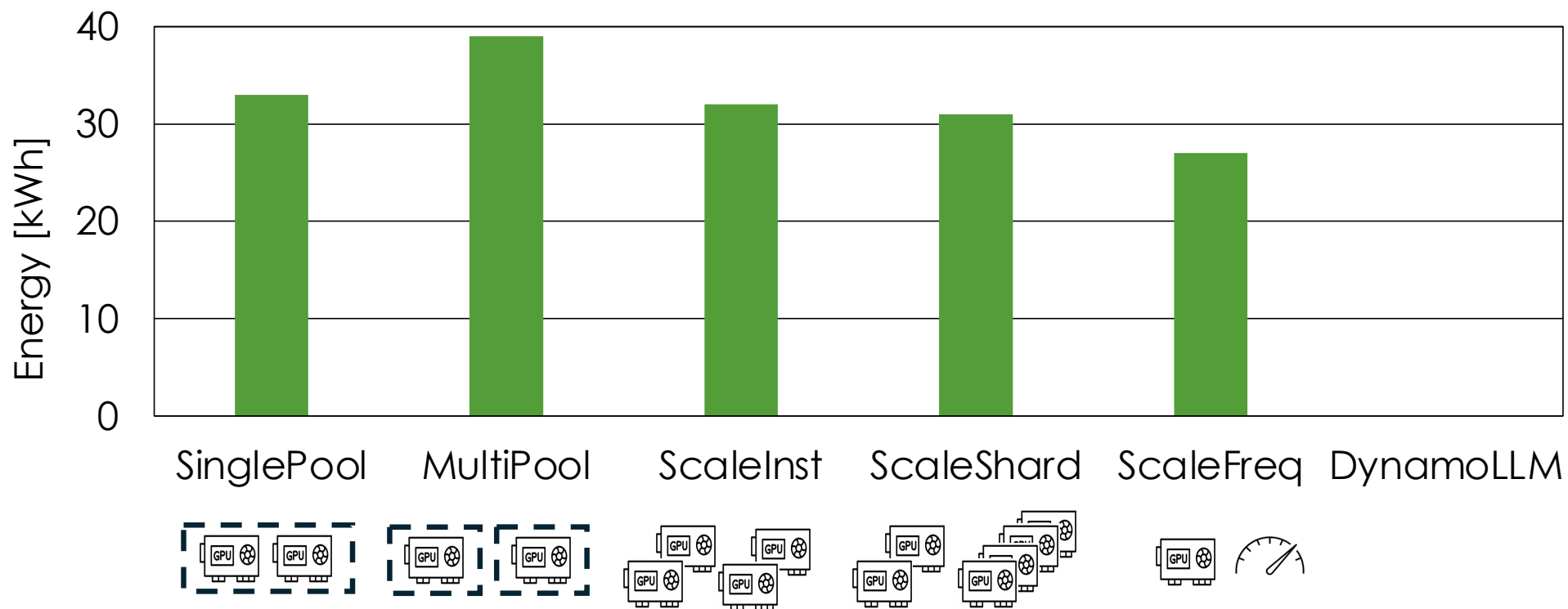
DynamoLLM significantly reduces energy!



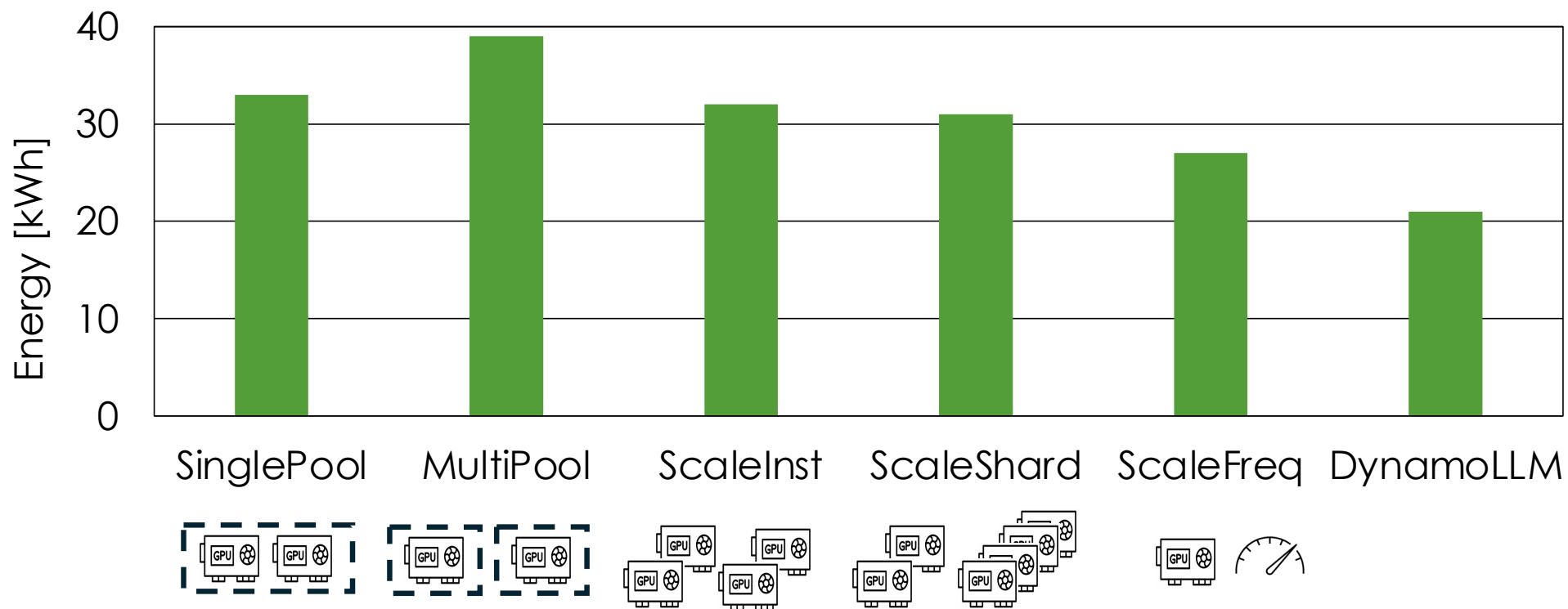
DynamoLLM significantly reduces energy!



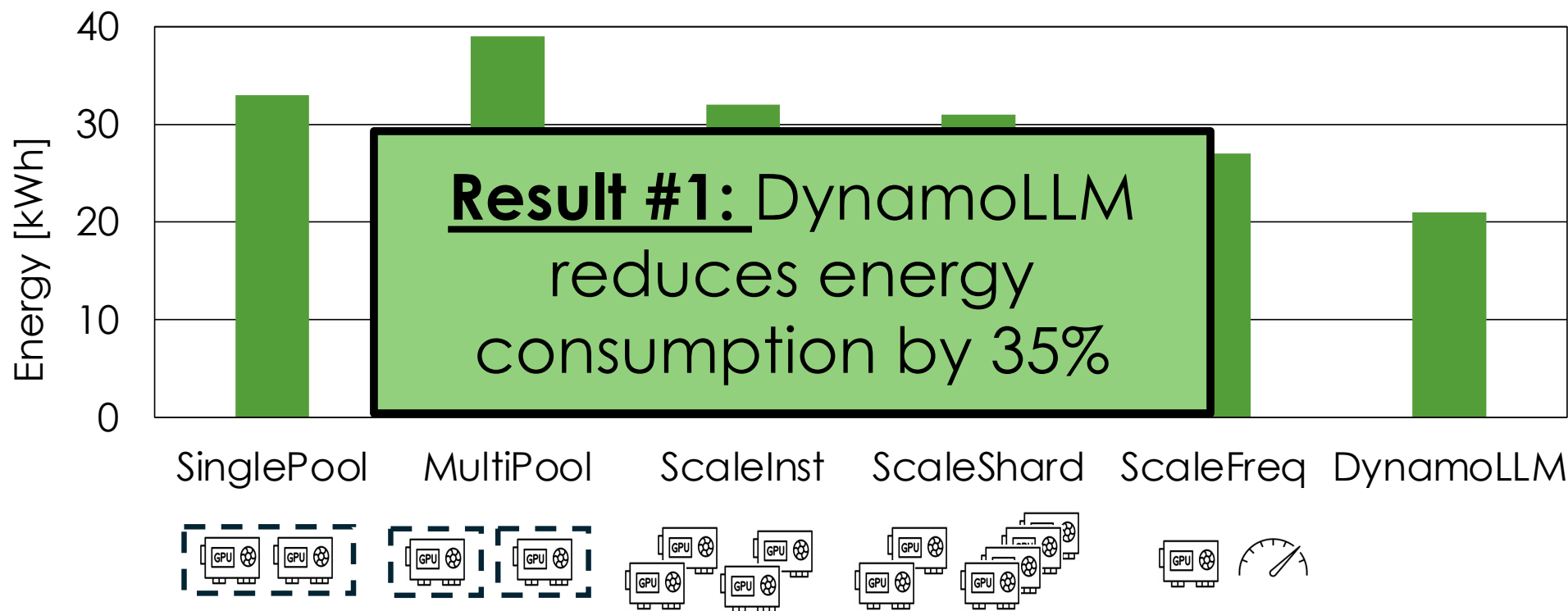
DynamoLLM significantly reduces energy!



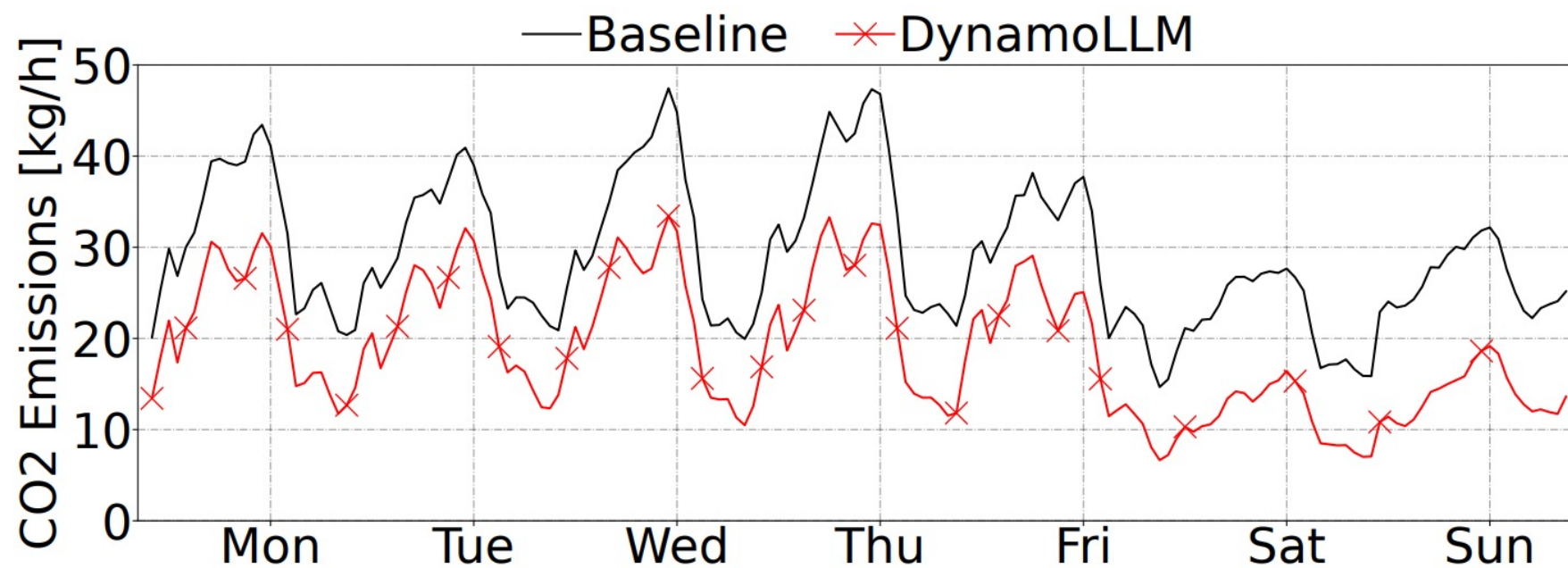
DynamoLLM significantly reduces energy!



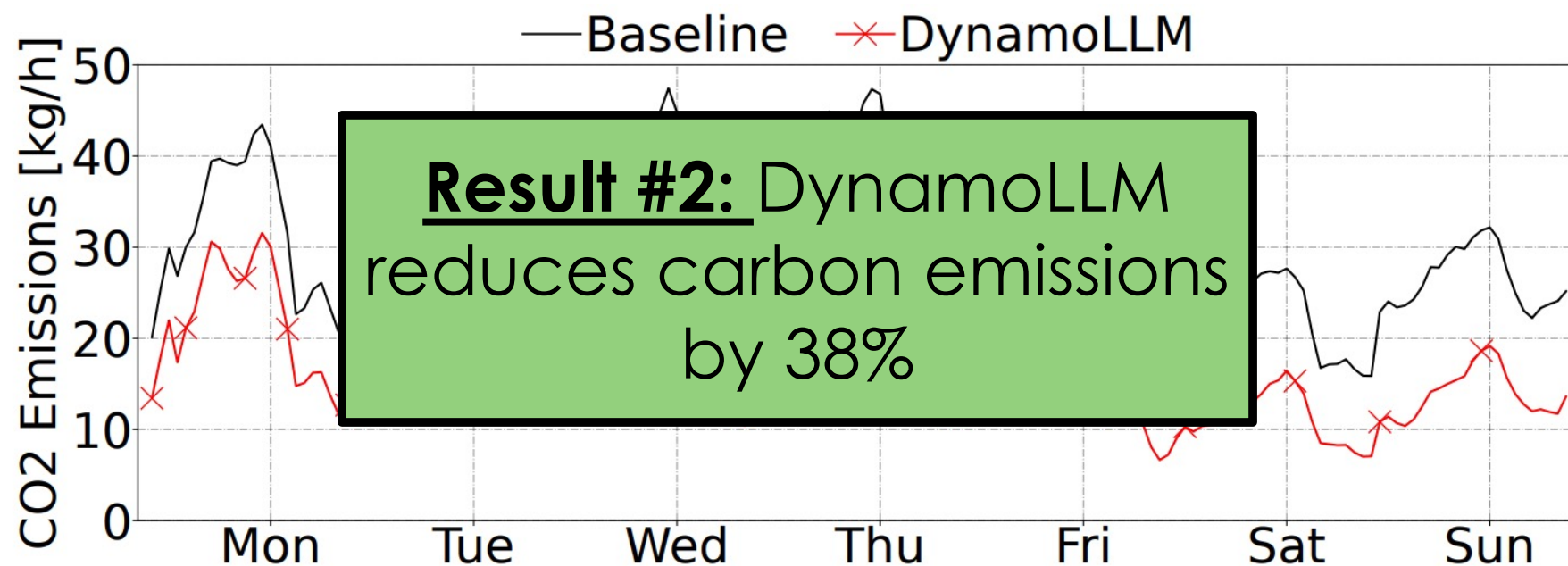
DynamoLLM significantly reduces energy!



DynamoLLM significantly reduces carbon!



DynamoLLM significantly reduces carbon!



Conclusion

- LLM inference emerging workload in the cloud
 - Its execution energy inefficient
- Need to address these challenges for cost-effective and environmentally-conscious datacenters
- **DynamoLLM** as the first step towards our goal!



DynamoLLM: Designing LLM Inference Clusters for Performance and Energy Efficiency

HPCA 2025

Open-source
production traces



Jovan Stojkovic, Chaojie Zhang*, Íñigo Goiri*, Josep Torrellas, Esha Choukse*

University of Illinois at Urbana-Champaign, *Azure Research Systems

Backup Slides

DynamoLLM – Energy Evaluation

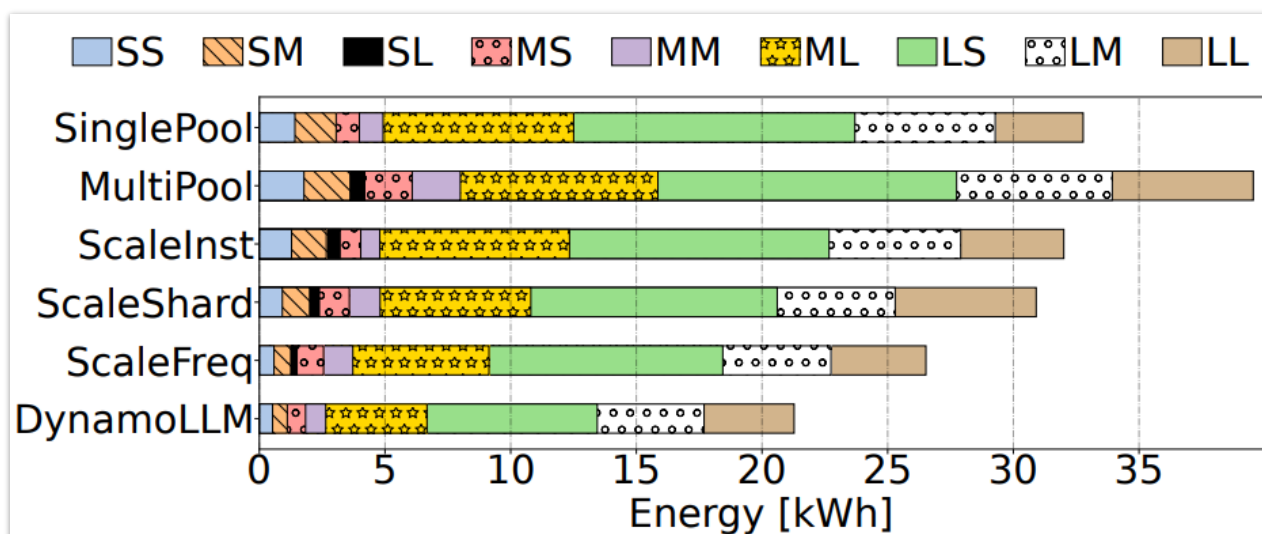
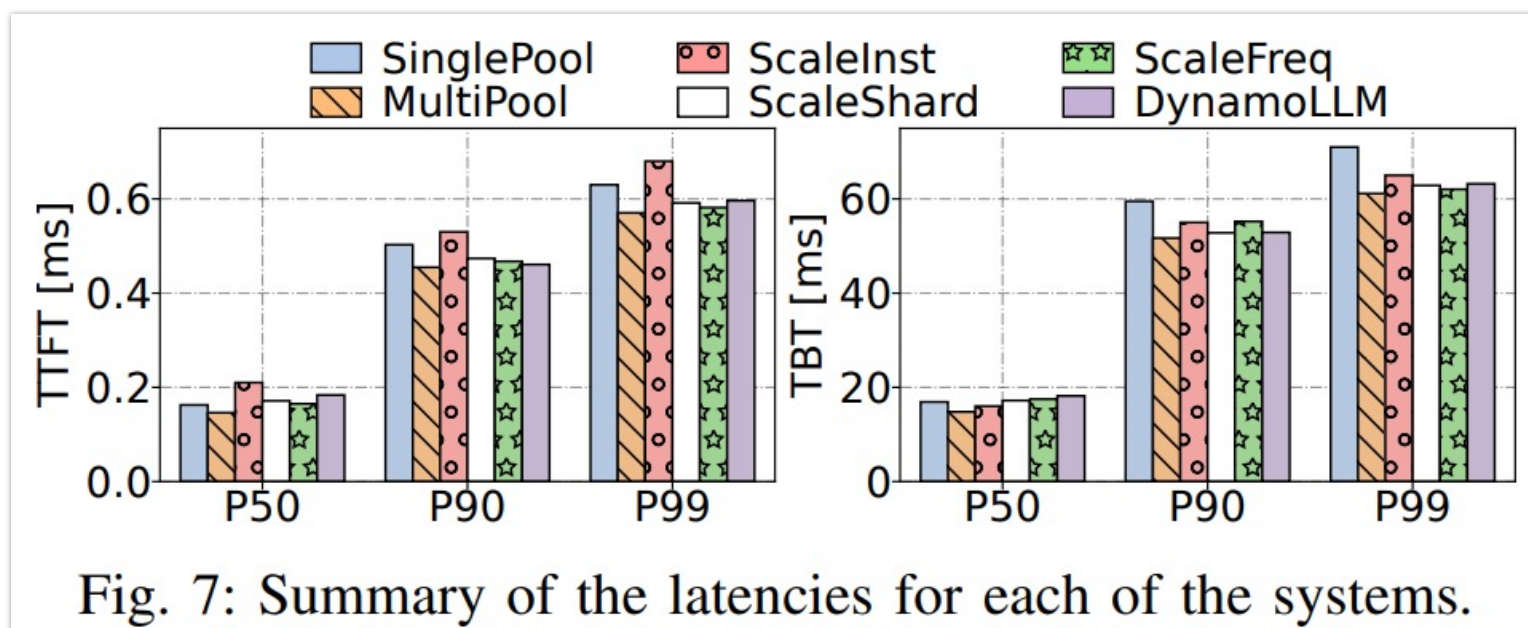


Fig. 6: Energy consumption with the six evaluated systems.

DynamoLLM – Latency Evaluation



Dynamo LLM – Power Evaluation

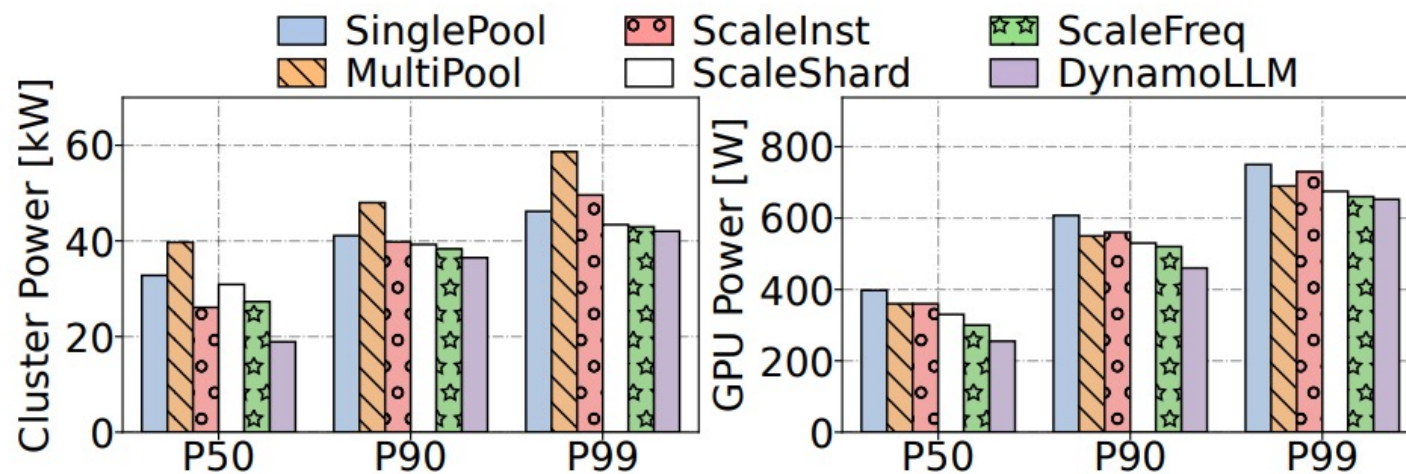


Fig. 8: Summary of the power for each of the systems.

DynamoLLM – Frequency Evaluation

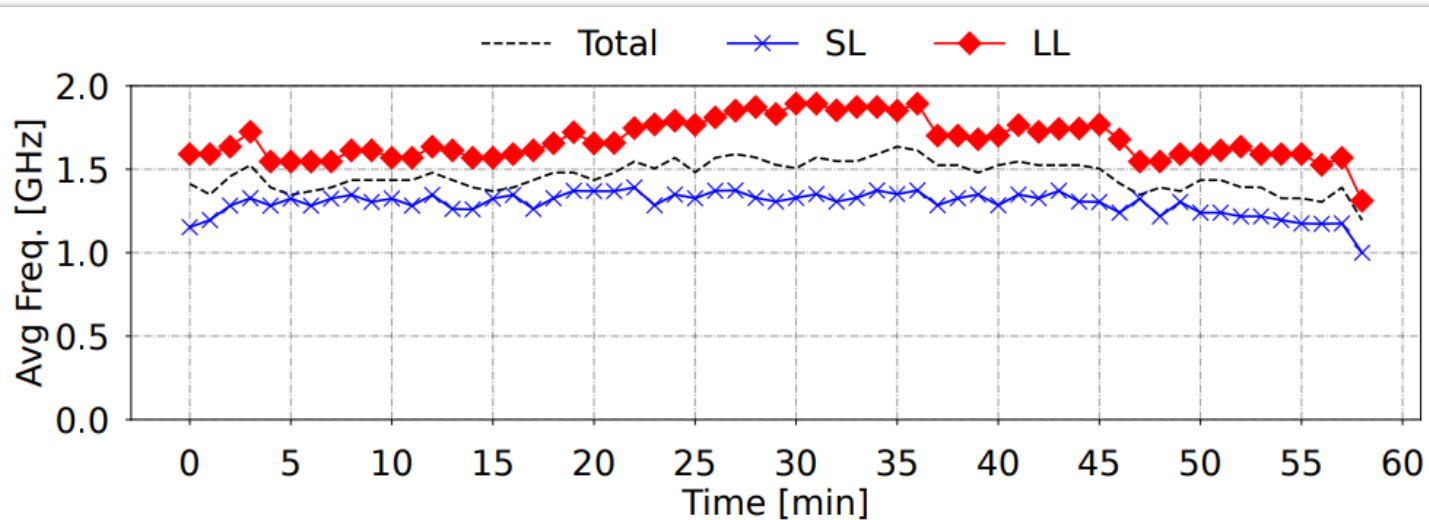


Fig. 9: GPU Frequency over one hour for DynamoLLM.

DynamoLLM – Sharding Evaluation

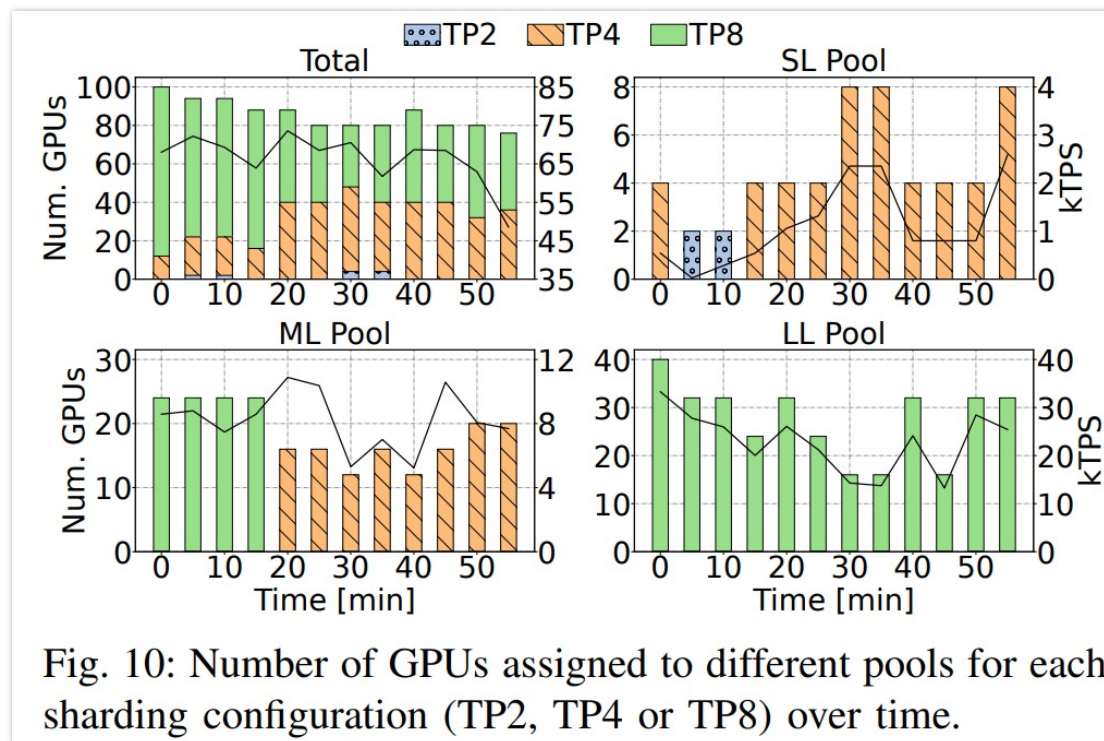


Fig. 10: Number of GPUs assigned to different pools for each sharding configuration (TP2, TP4 or TP8) over time.

DynamoLLM – Sensitivity to Pool Count

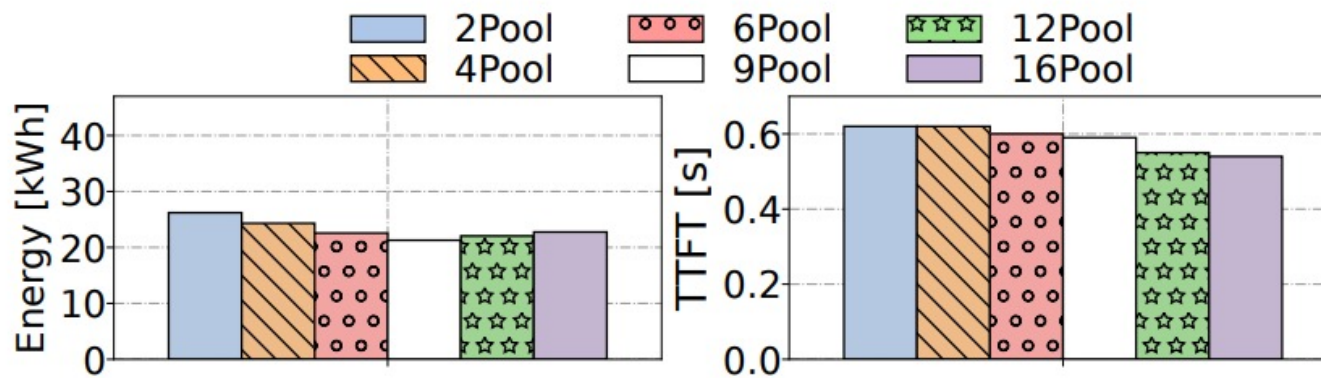


Fig. 13: Energy and performance with different number of pools (or request types).

DynamoLLM – Sensitivity to Accuracy

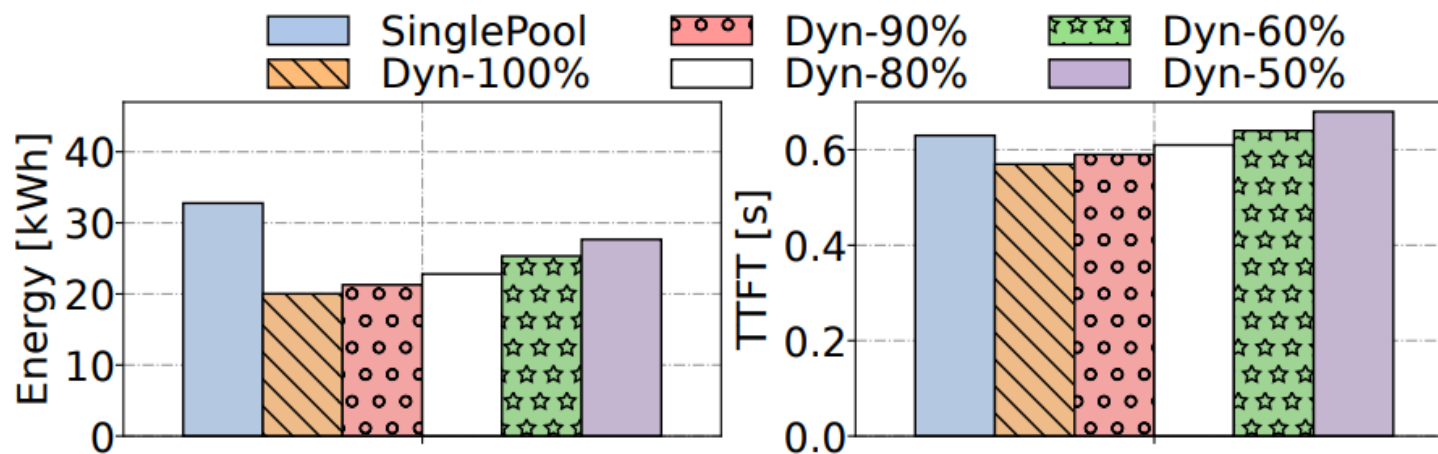


Fig. 11: Energy and performance with different accuracy.

DynamoLLM – Sensitivity to Load

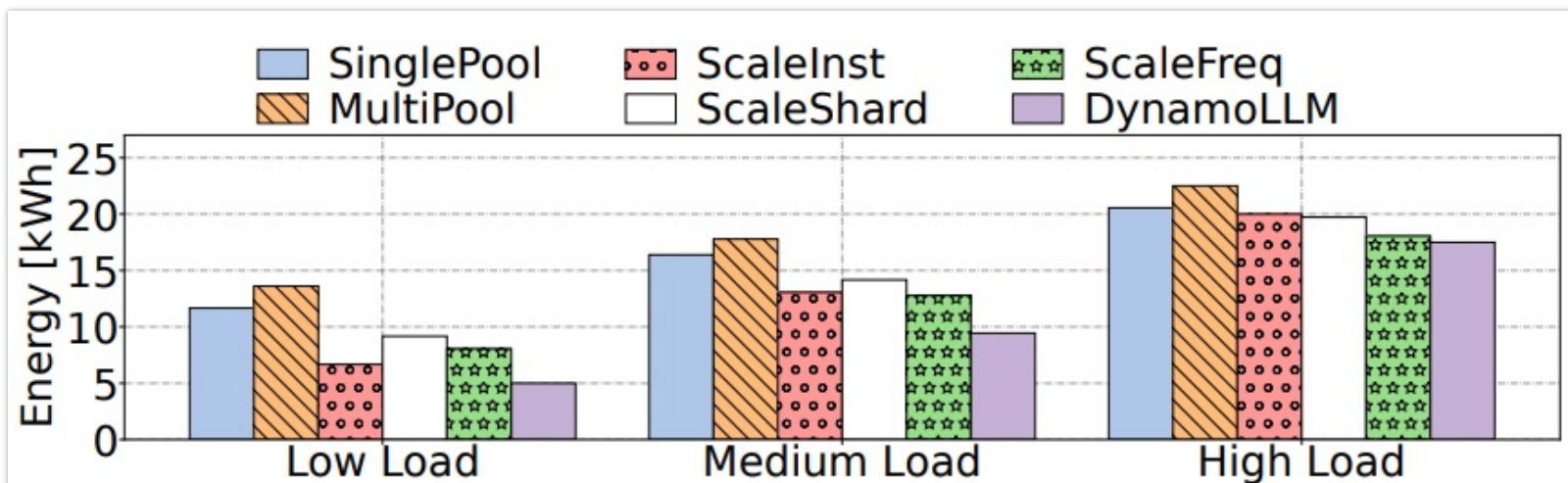


Fig. 12: Energy comparison with different levels of load.