Alibaba Cloud | Worldwide Cloud Services Partner

OSDI 24

Llumnix: Dynamic Scheduling for Large Language Model Serving

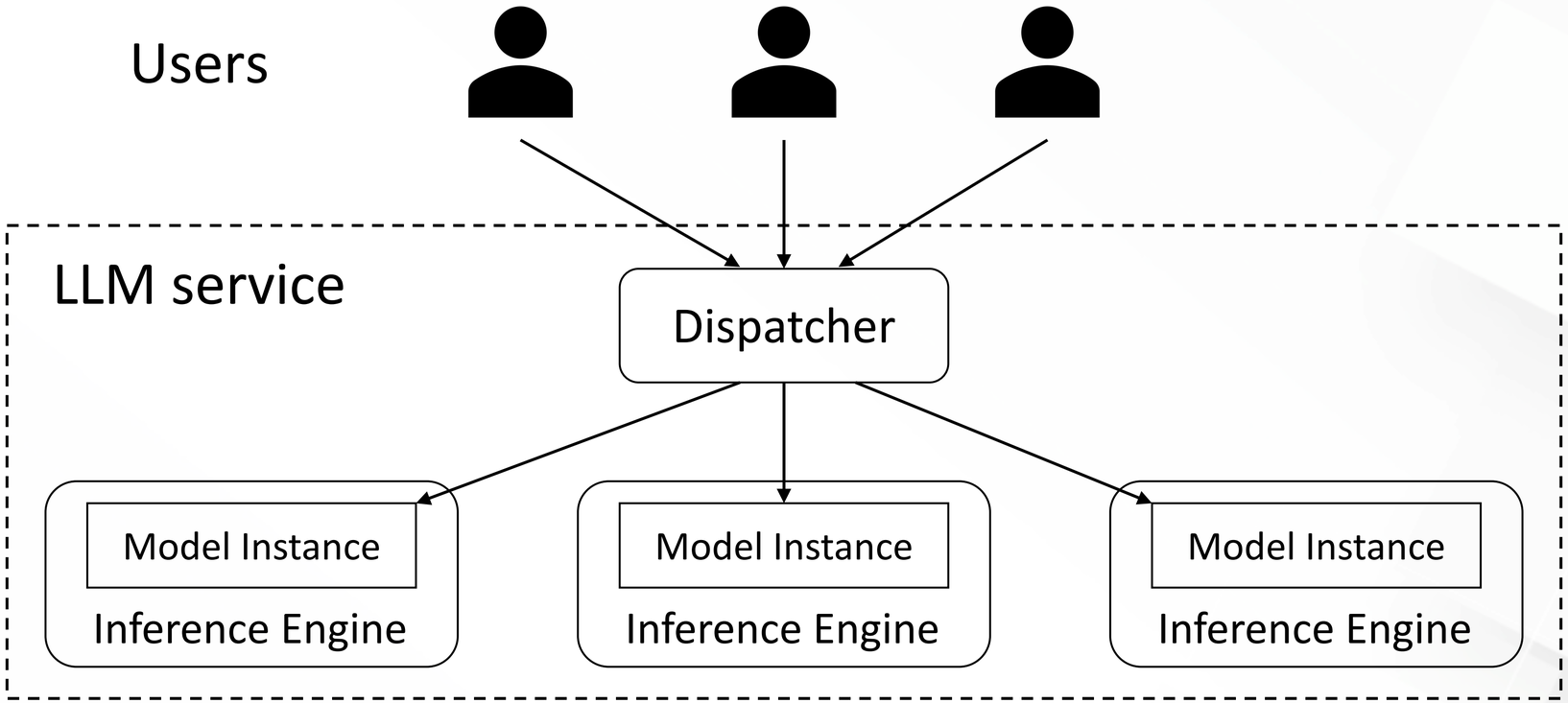
Biao Sun*, Ziming Huang*, **Hanyu Zhao***, Wencong Xiao, Xinyi Zhang, Yong Li, Wei Lin

(* Equal contribution)

Alibaba Group

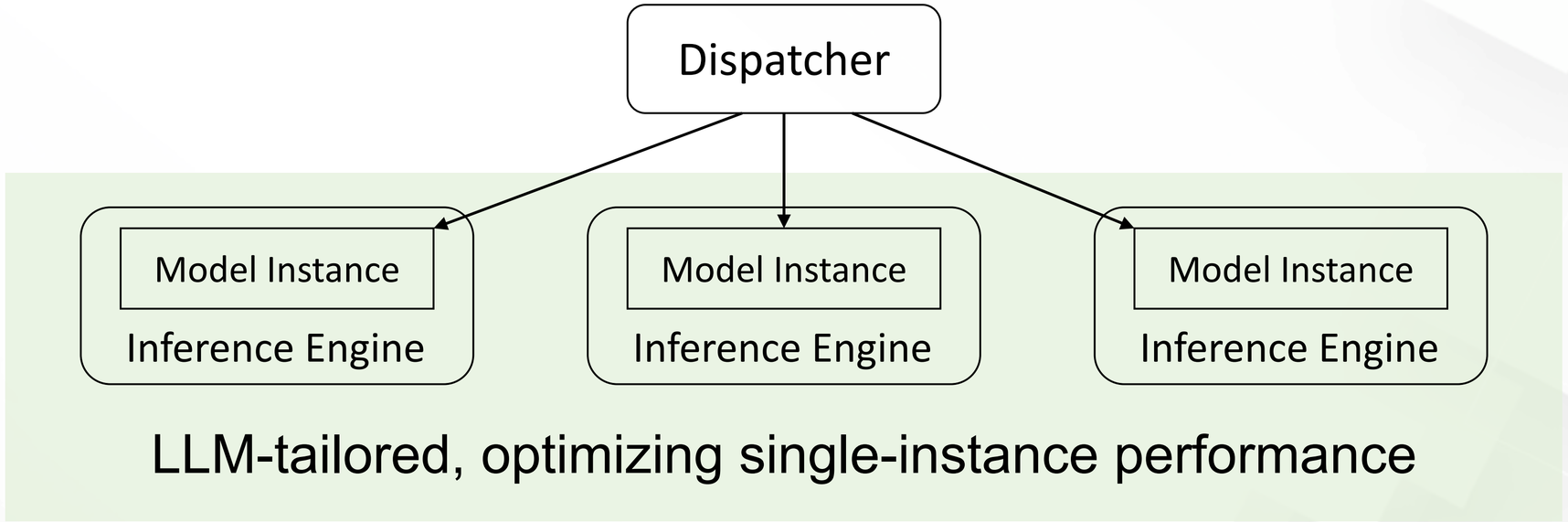
LLM Serving Today: A Cluster Perspective

- A **request dispatcher** + *multiple instances* of an **inference engine**



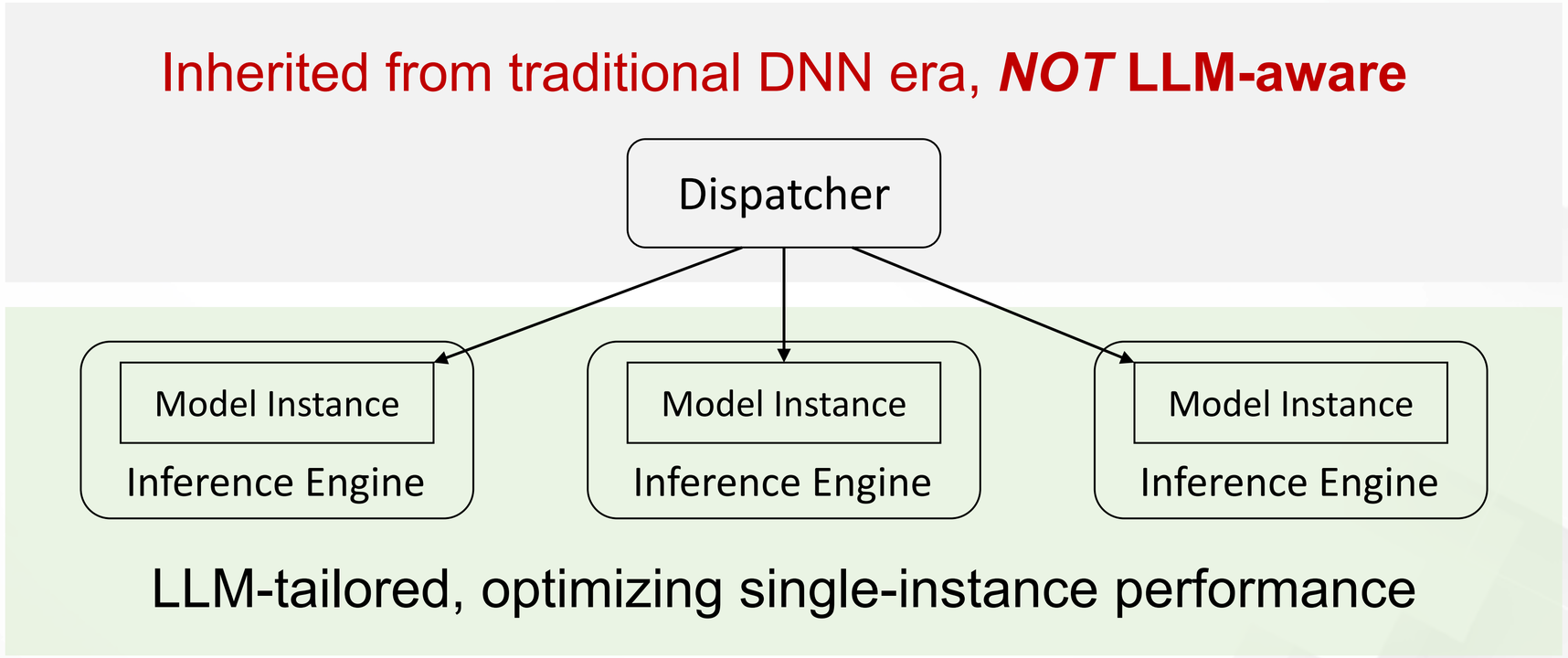
LLM Serving Today: A Cluster Perspective

- A **request dispatcher** + *multiple instances* of an **inference engine**



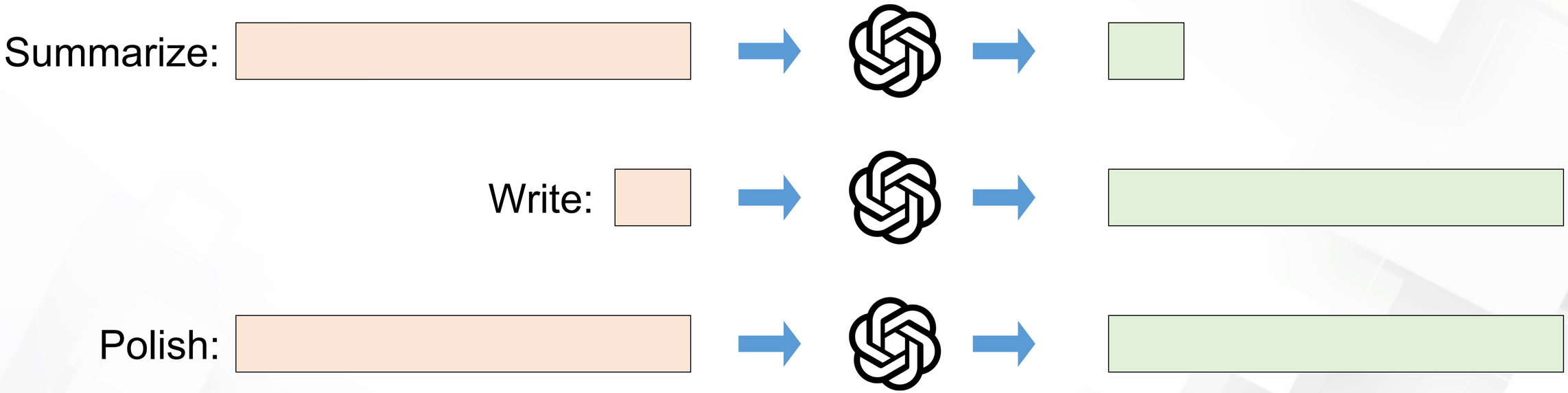
LLM Serving Today: A Cluster Perspective

- A **request dispatcher** + *multiple instances* of an **inference engine**



LLM Characteristic (1): *Workload Heterogeneity*

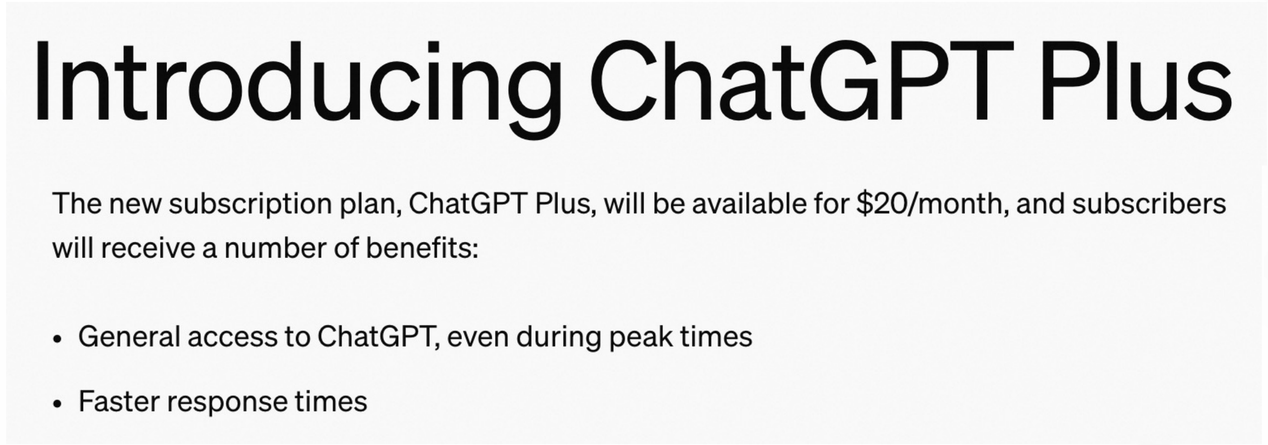
- *Universal* models, *diverse* applications
- Requests are **heterogeneous**
  - *Sequence (input/output) lengths*



LLM Characteristic (1): *Workload Heterogeneity*

- *Universal* models, *diverse* applications

- Requests are **heterogeneous**
  - *Sequence (input/output) lengths*
  - *Latency SLOs*: interactive vs. offline, ChatGPT plus vs. normal



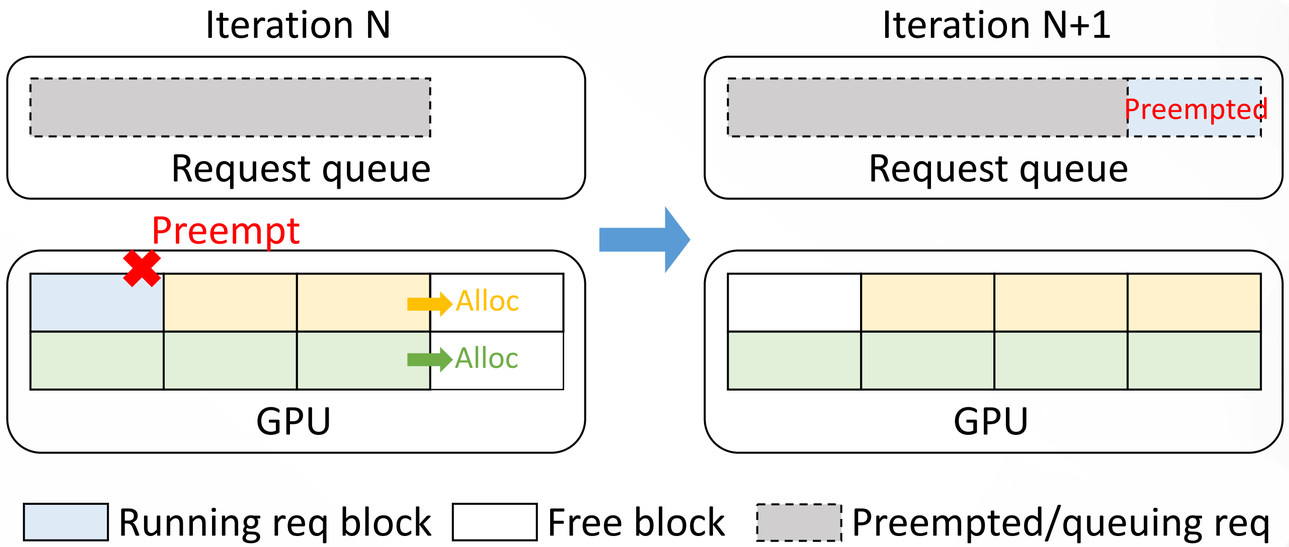
Introducing ChatGPT Plus

The new subscription plan, ChatGPT Plus, will be available for $20/month, and subscribers will receive a number of benefits:

- General access to ChatGPT, even during peak times
- Faster response times

LLM Characteristic (2): *Execution Unpredictability*

- **Autoregressive** execution
  - Output lengths *not known a priori*
  - *Dynamic* GPU memory demands of *KV caches*
- State of the art: paged memory allocation + preemptive scheduling [1]



[1] Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention (SOSP '23)

Challenge (1): Performance Isolation

- Preemptions -> poor tail latencies

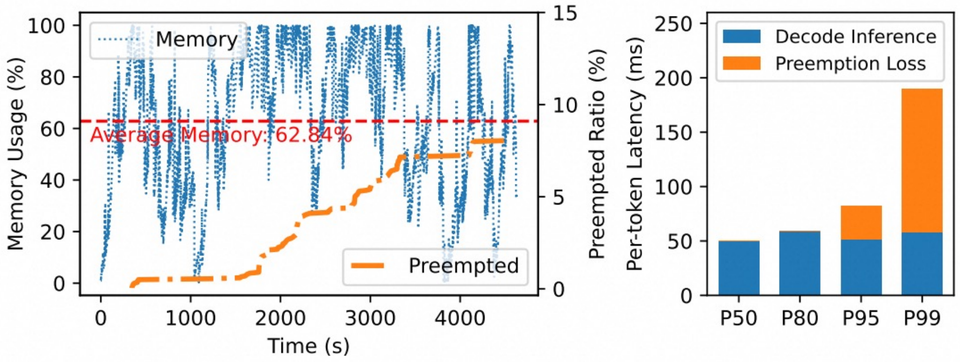


Figure 3: Request preemptions in LLaMA-7B serving.

- Performance interference in a batch

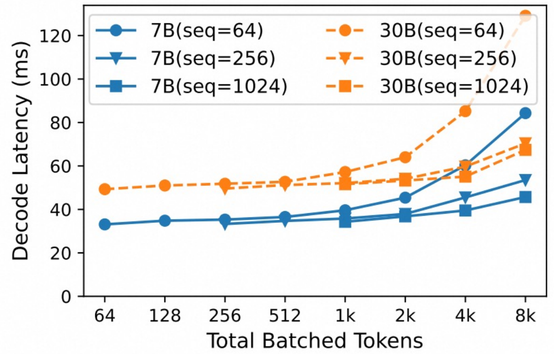


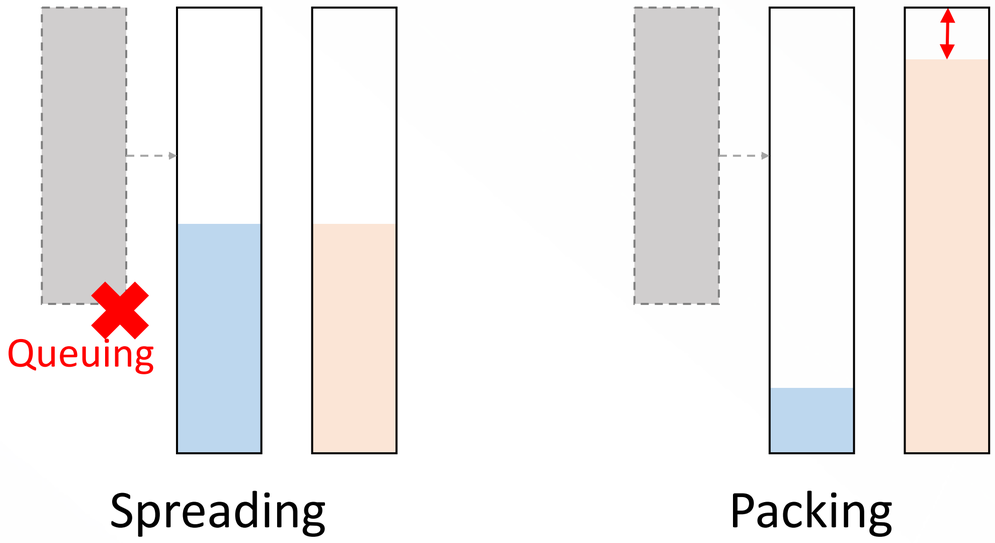
Figure 4: Latencies of one decode step of LLaMA-7B and LLaMA-30B with different sequence lengths and batch sizes.

- Load balancing via *one-shot* dispatching could be suboptimal due to unpredictable execution

Requirement (1): **Continuous** load balancing

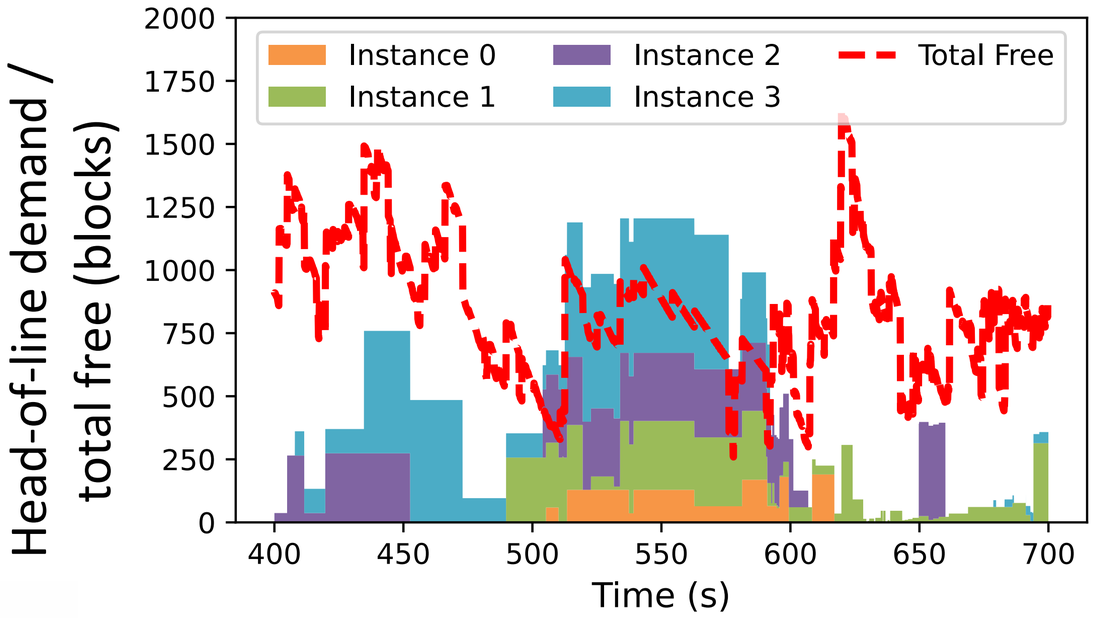
Challenge (2): Memory Fragmentation

- Load balancing -> fragmentation across instances
  - A classic spreading vs. packing tradeoff



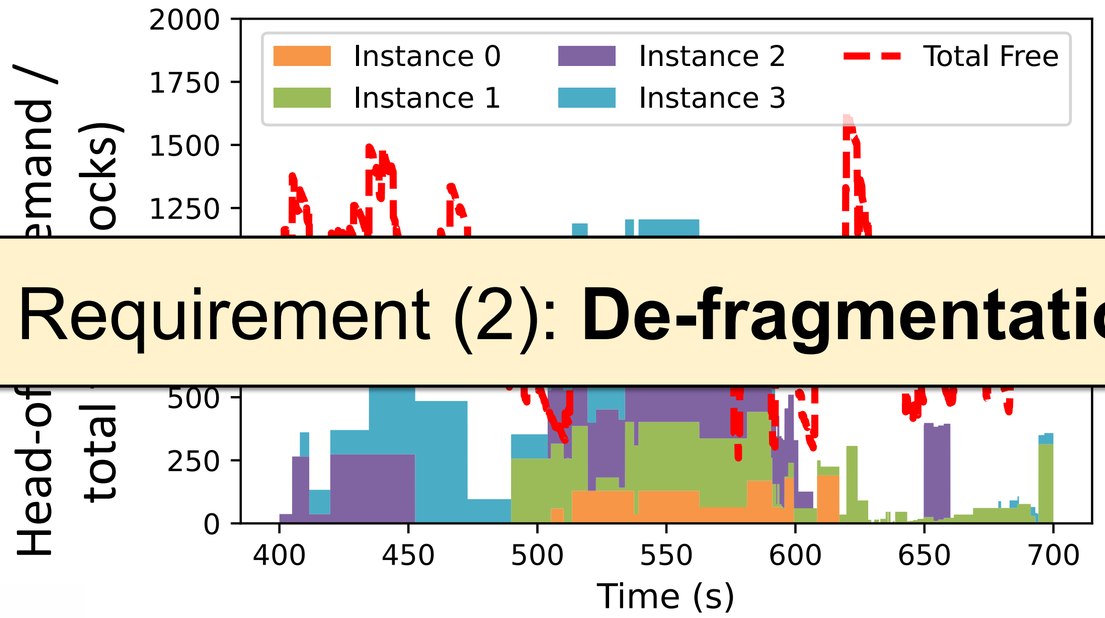
Challenge (2): Memory Fragmentation

- Load balancing -> fragmentation across instances
  - A classic spreading vs. packing tradeoff
- Fragmentation -> **worse queuing delays** (first-token latencies)
  - A large space on one instance needed for the prompt



Challenge (2): Memory Fragmentation

- Load balancing -> fragmentation across instances
  - A classic spreading vs. packing tradeoff
- Fragmentation -> **worse queuing delays** (first-token latencies)
  - A large space on one instance needed for the prompt



Requirement (2): **De-fragmentation**

Challenge (3): Differentiated SLOs

- Existing systems treat all requests **equally**

- Urgent requests could be easily interfered by normal ones
  - Queuing delays
  - Performance interference

Requirement (3): Request **priorities**

LLMs are *Multi-Tenant* and *Dynamic*

A behavior that is:

Different from traditional DNNs

- Homogeneous requests
- Deterministic, stateless execution

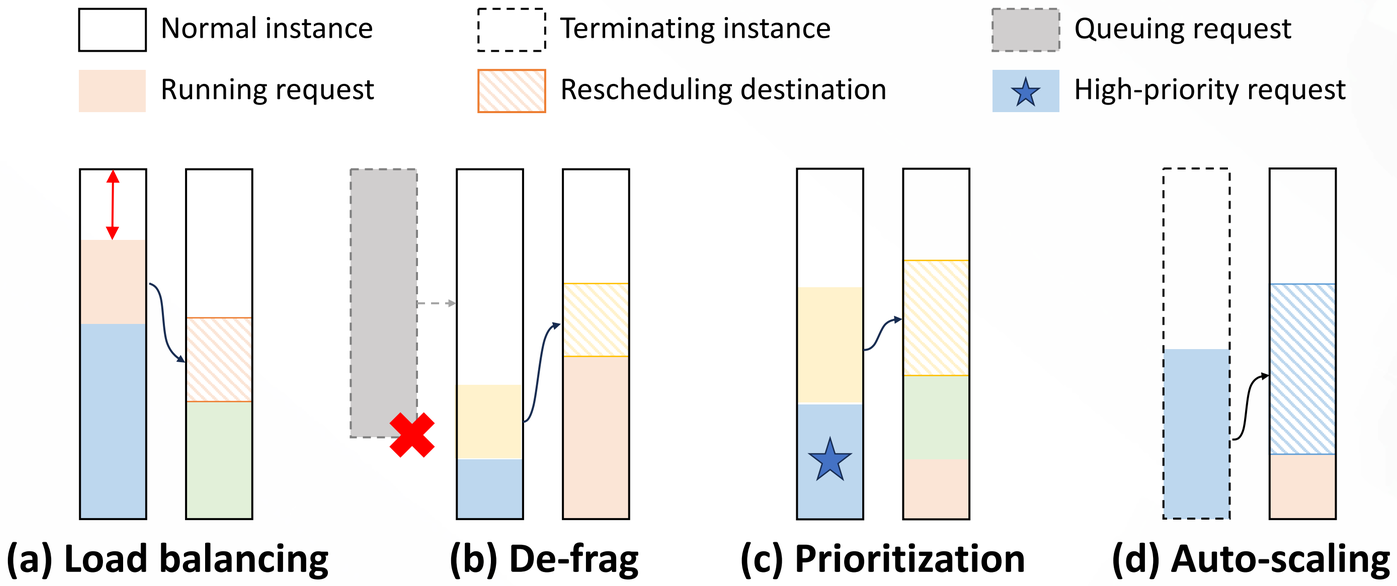
but…

Not new in modern operating / distributed systems

- Processes with dynamic working sets, unknown durations, different priorities, …
- Context switching, process migration, …

Llumnix: Serving LLMs, the “OS” Way

- **Continuous rescheduling** across instances
  - Combined with dispatching and auto-scaling
- Powerful in various scheduling scenarios



Design Goals

Our aim: make rescheduling the *norm* in LLM serving

Efficiency → *Live migration mechanism*

Scalability → *Distributed scheduling architecture*

Scheduling Benefits → *Unified, multi-objective scheduling policy*

Design Goals

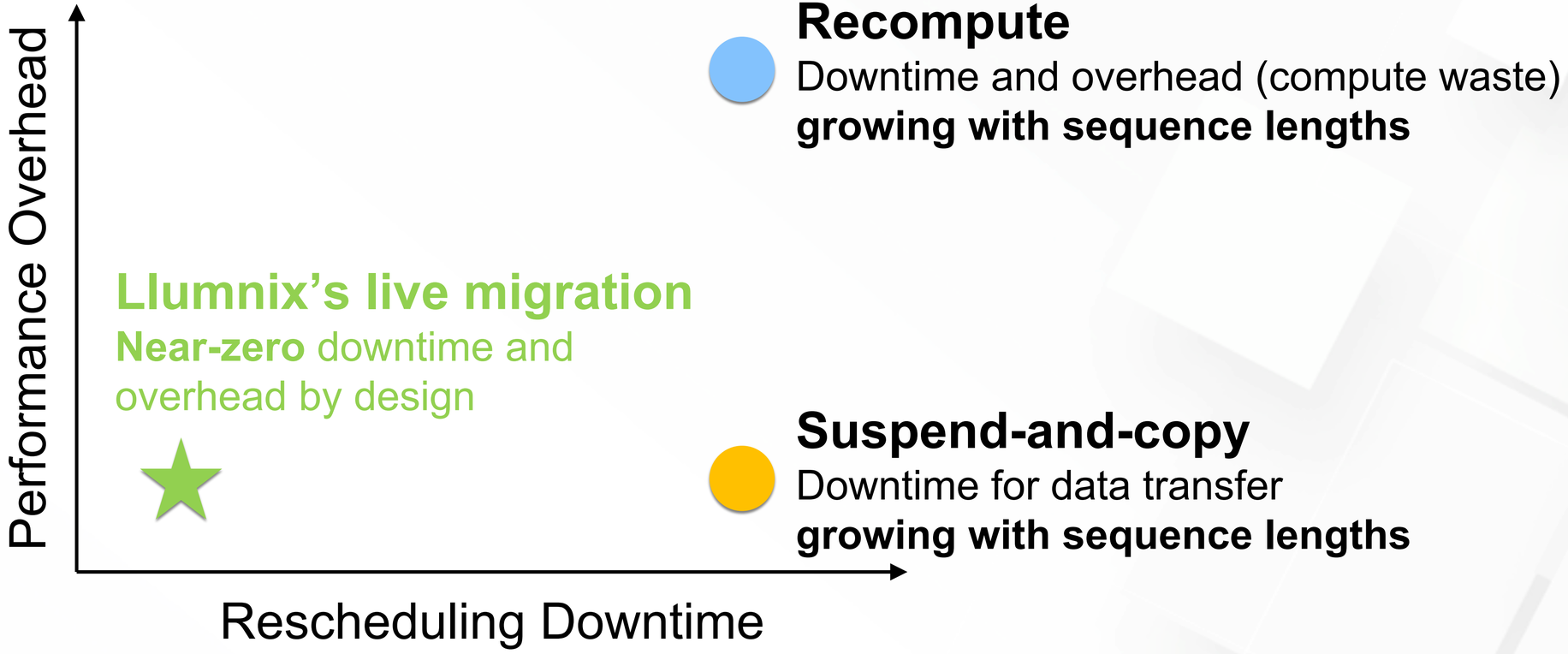
Our aim: make rescheduling the *norm* in LLM serving

Efficiency → *Live migration mechanism*

Scalability → *Distributed scheduling architecture*

Scheduling Benefits → *Unified, multi-objective scheduling policy*

How to Reschedule KV Caches?

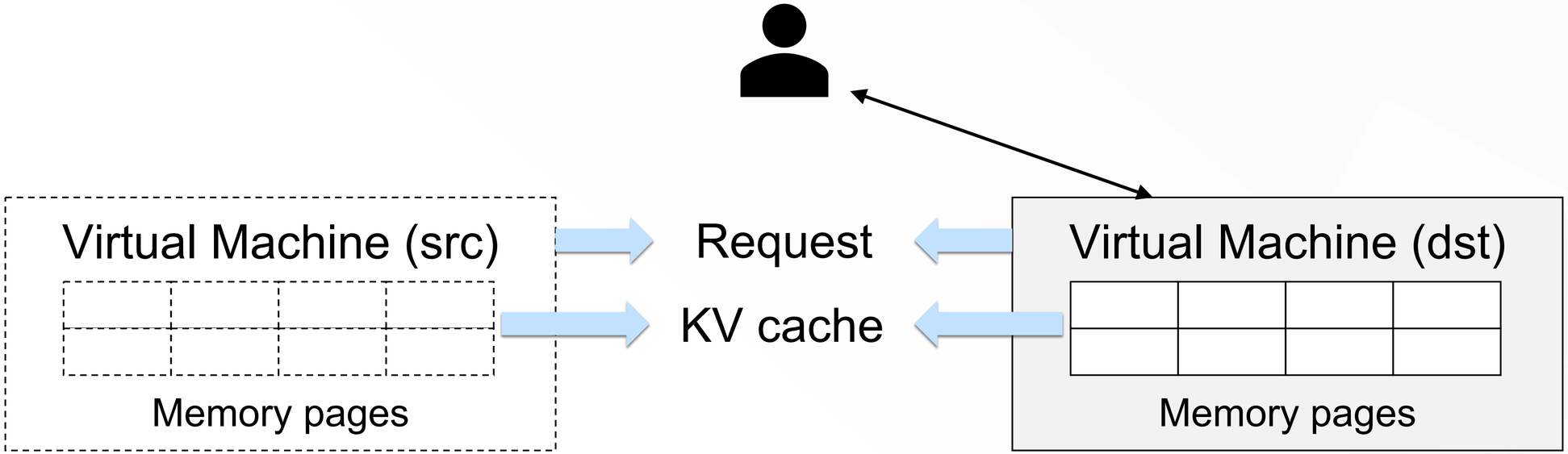


Inspiration: VM Live Migration

Inspiration: VM Live Migration

Inspiration: VM Live Migration

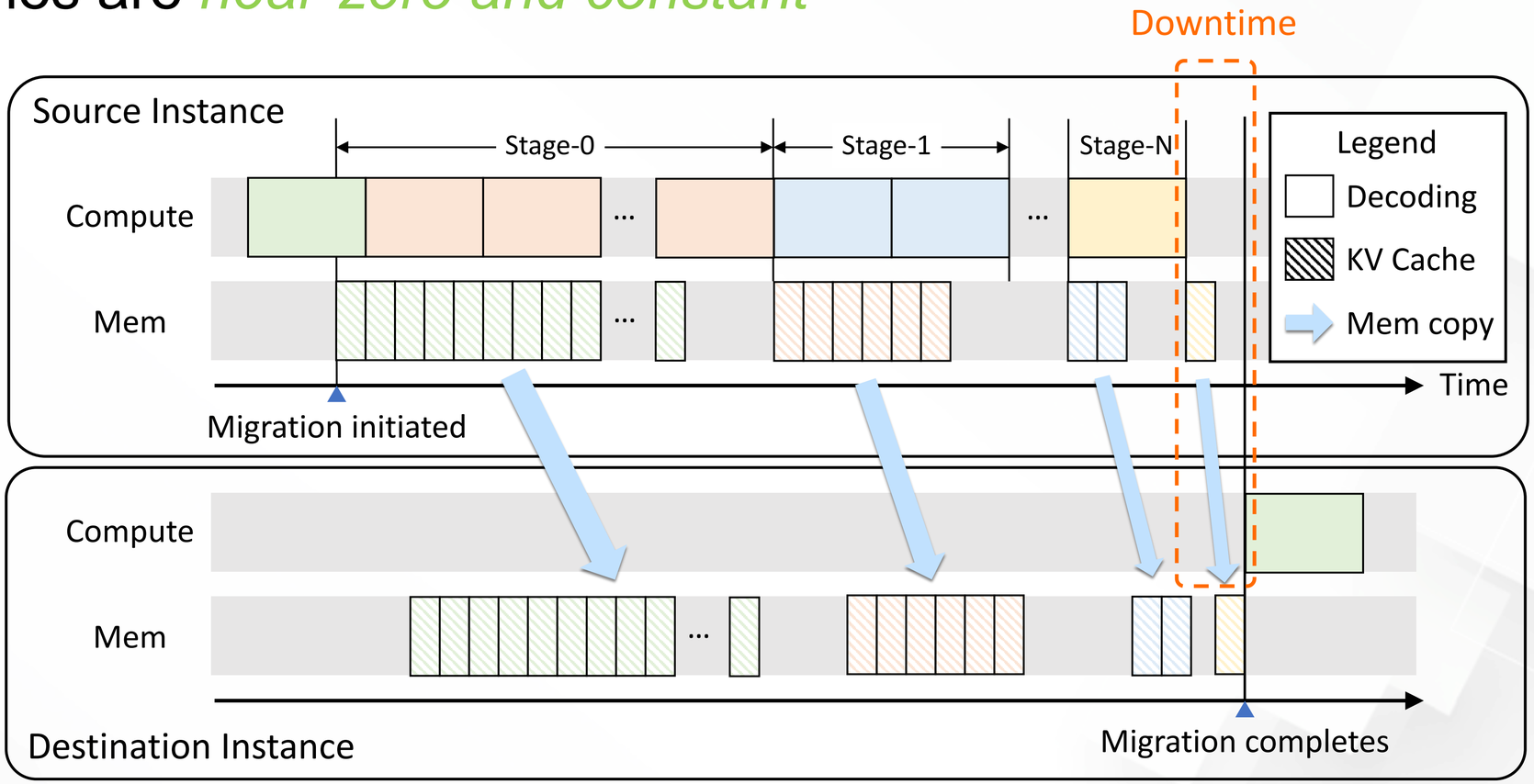
Inspiration: VM Live Migration



What are dirty pages?

Live Migration of LLM Requests

- KV caches are **append-only**
  - Copy ~~dirty~~ *incremental* blocks iteratively
  - Downtimes are *near-zero and constant*



Design Goals

Our aim: make rescheduling the *norm* in LLM serving

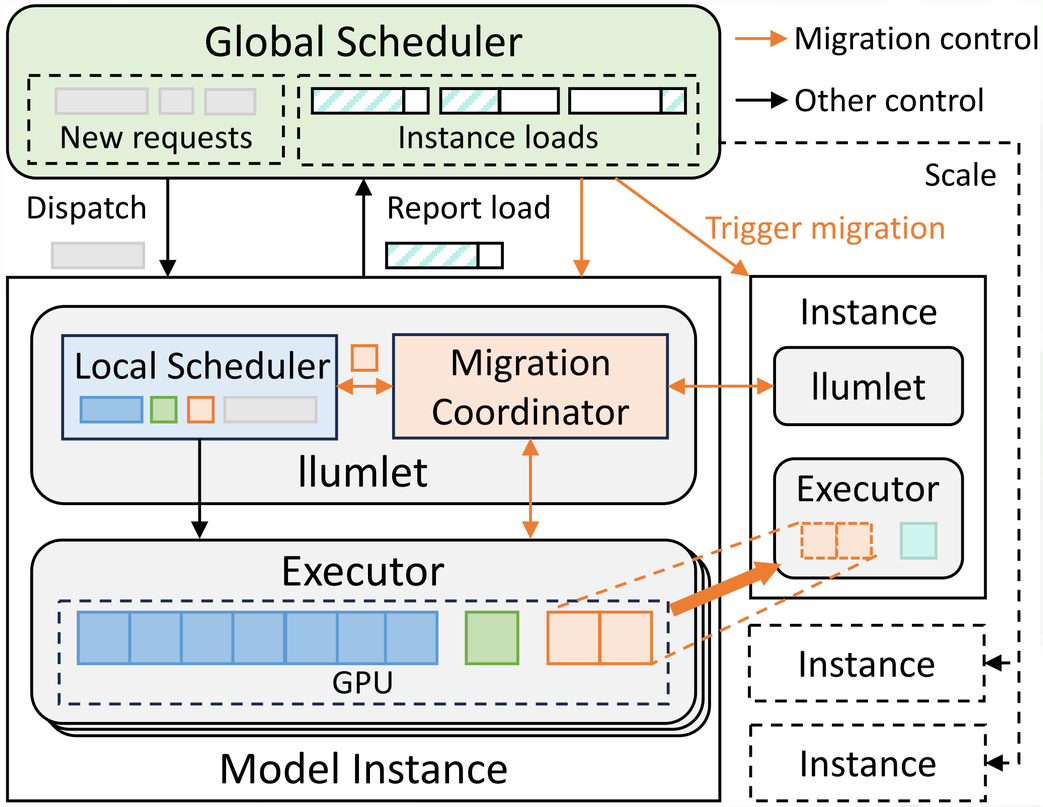
Efficiency → *Live migration mechanism*

Scalability → *Distributed scheduling architecture*

Scheduling Benefits → *Unified, multi-objective scheduling policy*

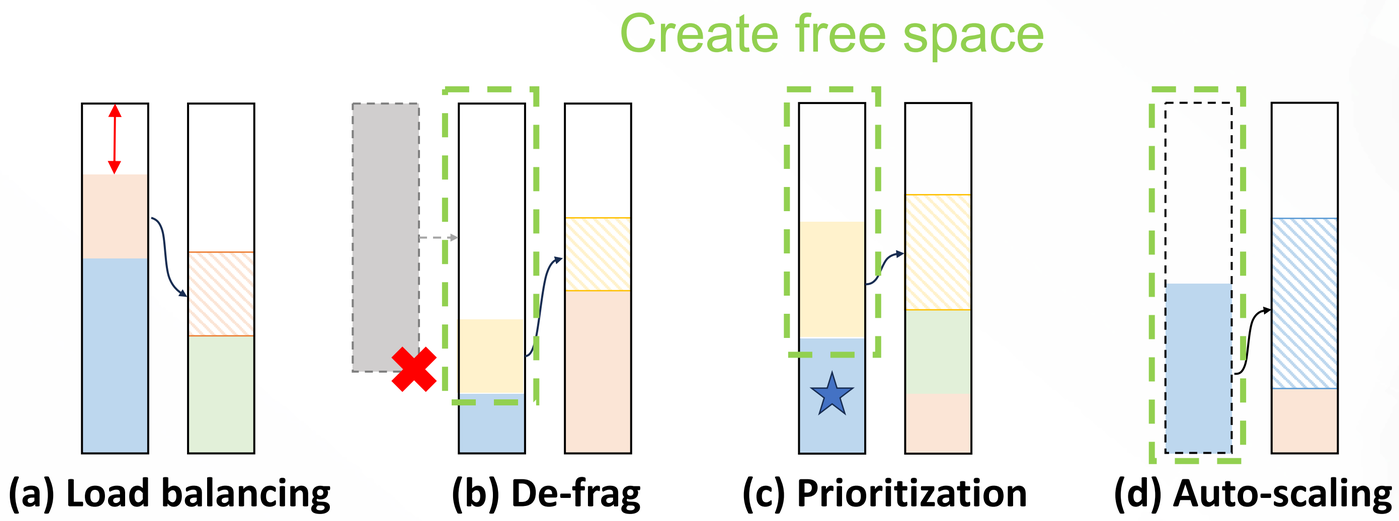
Distributed Scheduling Architecture

- **Global scheduler** for cross-instance scheduling
- Distributed **llumlet**s for local scheduling
- A narrow interface: **instance load**



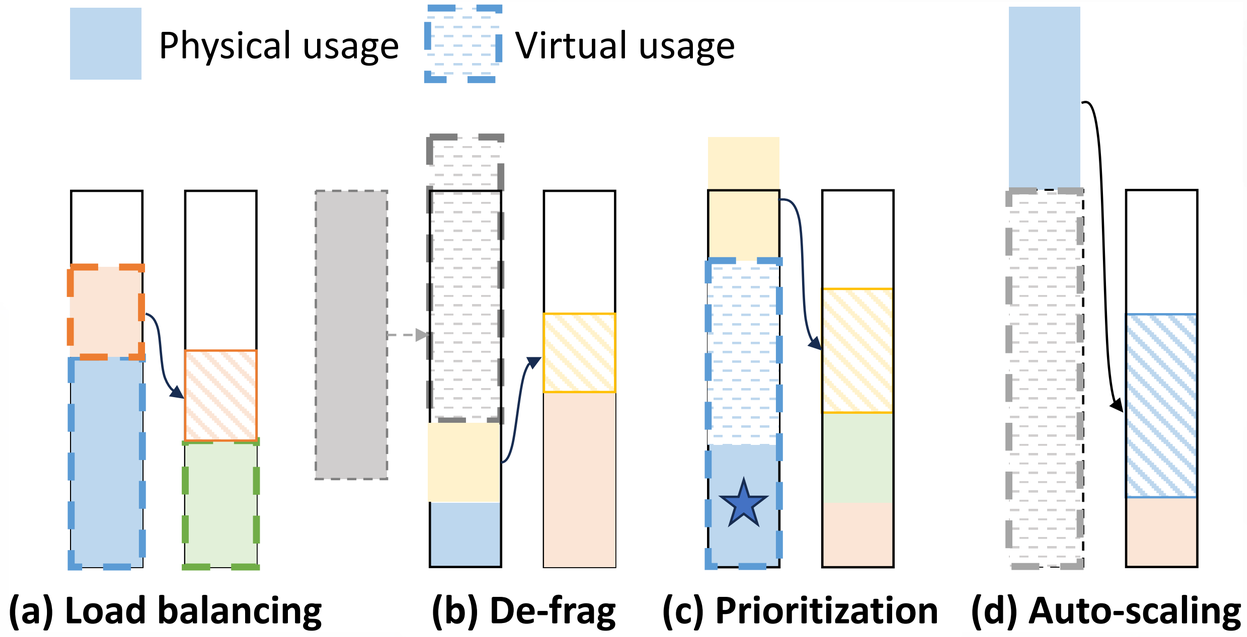
Scheduling Policy (sketch)

- **Virtual usage**: unifying multiple objectives
- Policy: load balancing based on virtual usages



Scheduling Policy (sketch)

- **Virtual usage**: unifying multiple objectives
- Policy: load balancing based on virtual usages

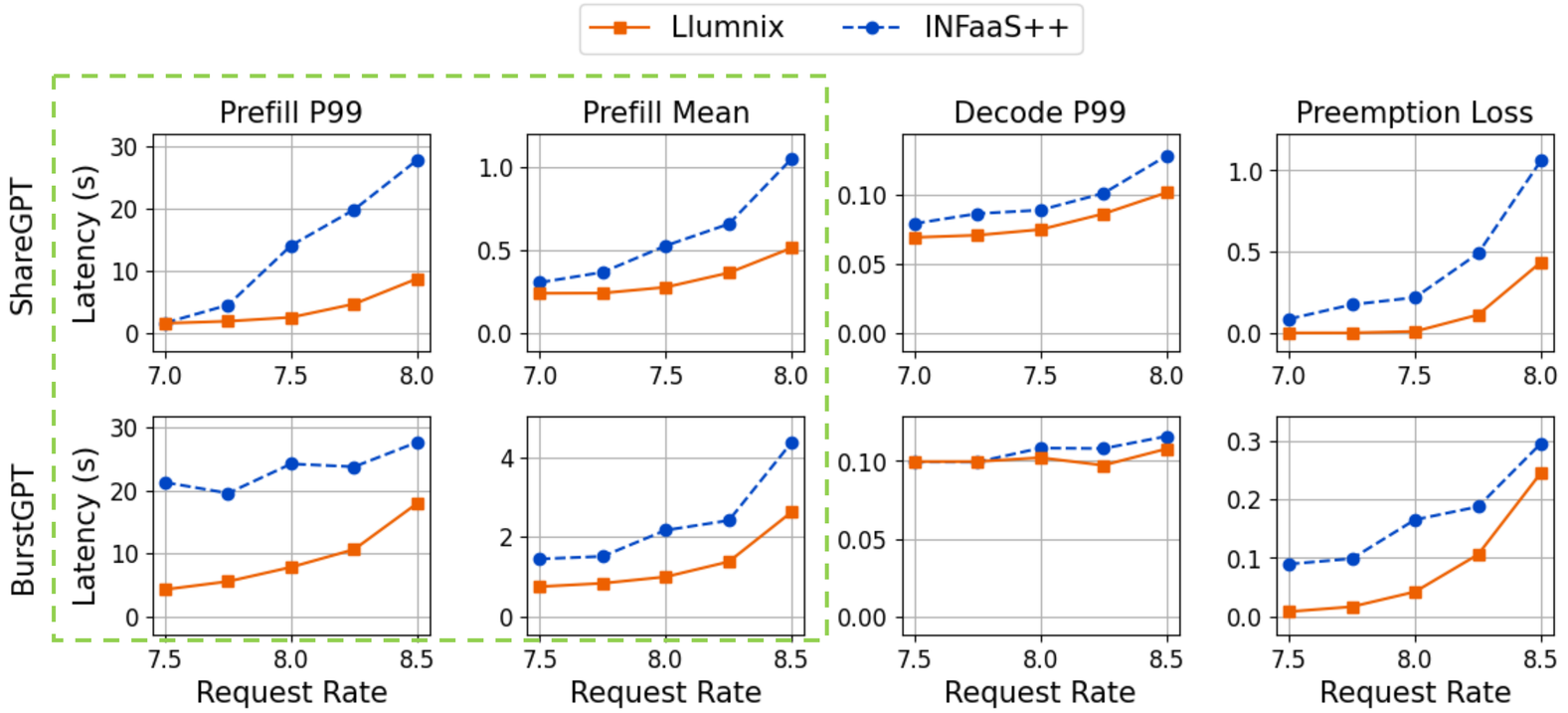


Evaluation

- Implemented as a scheduling layer atop vLLM
- Testbed: 16 A10 GPUs (24GB)
  - 4 4-GPU VMs, PCIe 4.0 in each node, 64Gb/s Ethernet across nodes
- Models: LLaMA-7B and LLaMA-30B
- Traces: ShareGPT, BurstGPT, generated power-law distributions

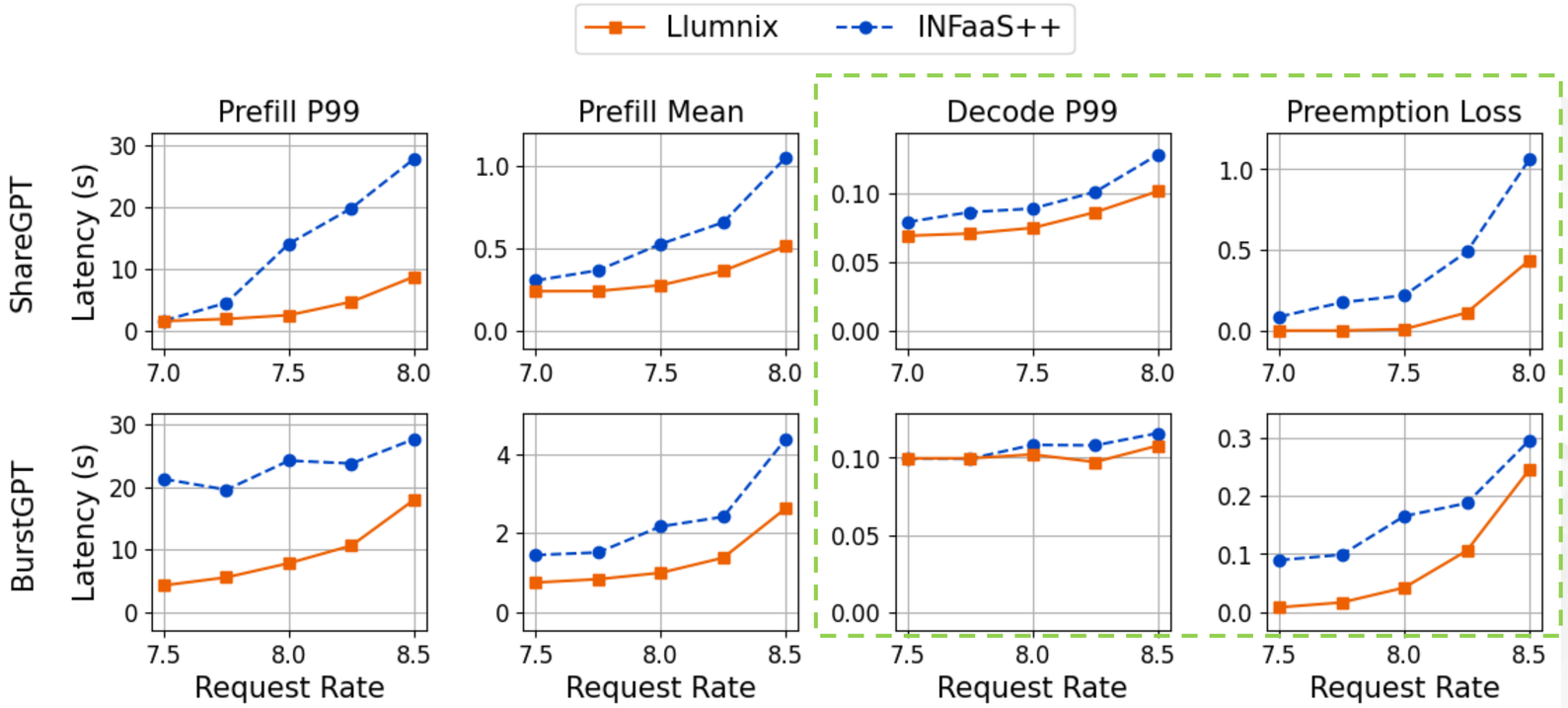
End-to-end Serving Performance (16 LLaMA-7B instances)

- Benefits of migration: compared to dispatch-time load balancing (INFaaS)
  - Up to 2.2x/5.5x for first-token (mean/P99) via de-fragmentation
  - Up to 1.3x for per-token generation P99 via reducing preemptions
- More gains with more diverse sequence lengths (details in paper)



End-to-end Serving Performance (16 LLaMA-7B instances)

- Benefits of migration: compared to dispatch-time load balancing (INFaaS)
  - Up to 2.2x/5.5x for first-token (mean/P99) via de-fragmentation
  - Up to 1.3x for per-token generation P99 via reducing preemptions
- More gains with more diverse sequence lengths (details in paper)



Conclusion

- Dynamic workloads need dynamic scheduling
  - LLMs are no exception

- Llumnix draws lessons from conventional systems wisdom
  - Classic scheduling goals in the new context of LLM serving
  - Implementation of rescheduling with request live migration
  - Continuous, dynamic rescheduling exploiting the migration

https://github.com/AlibabaPAI/llumnix

Alibaba Cloud | Worldwide Cloud Services Partner

Thanks