Llumnix: Dynamic Scheduling for Large Language Model Serving

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

Alibaba Group

OSDI 2024

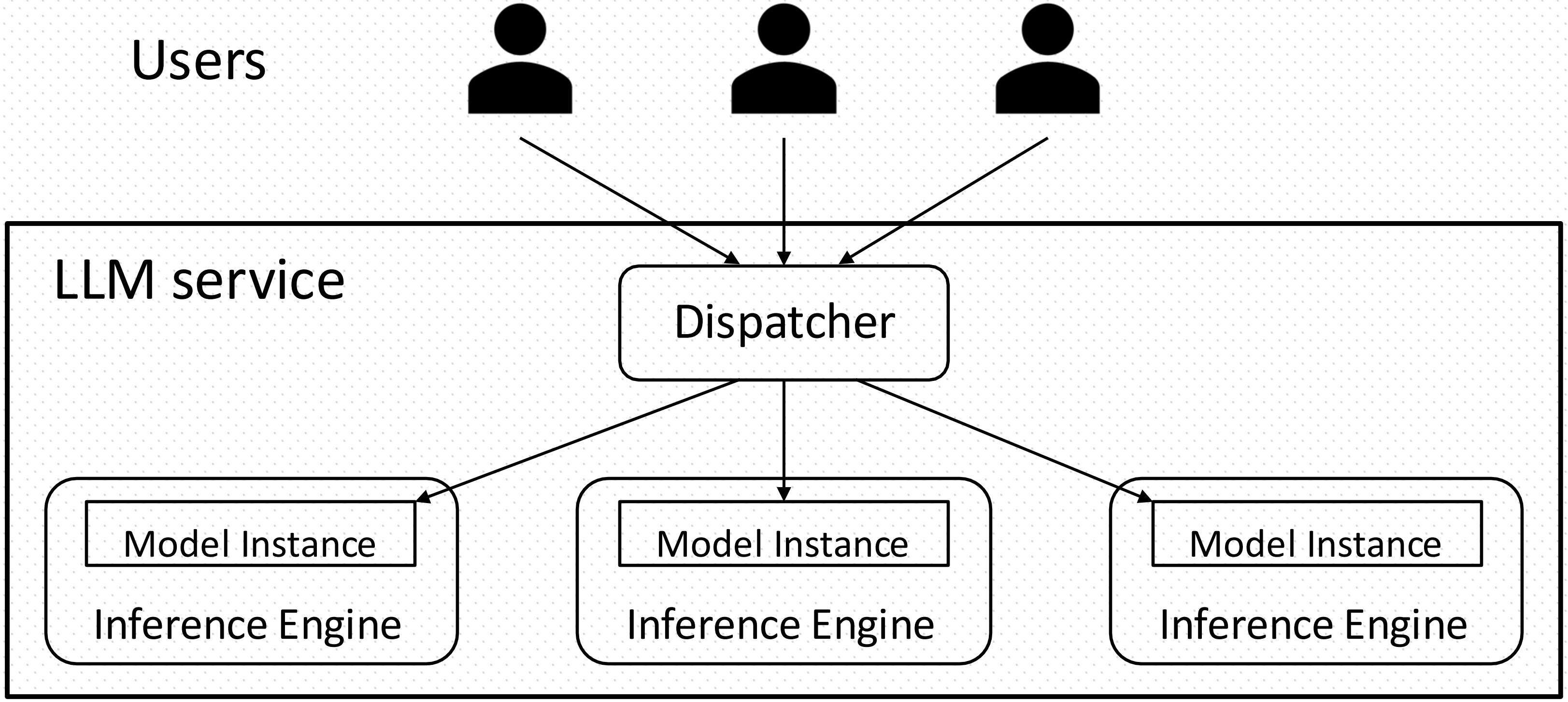
Presented by Kunzhao Xu

USTC, CHINA
ADSLAB
先进数据系统实验室

LLM Serving Today: A Cluster Perspective

USTC, CHINA ADSLAB

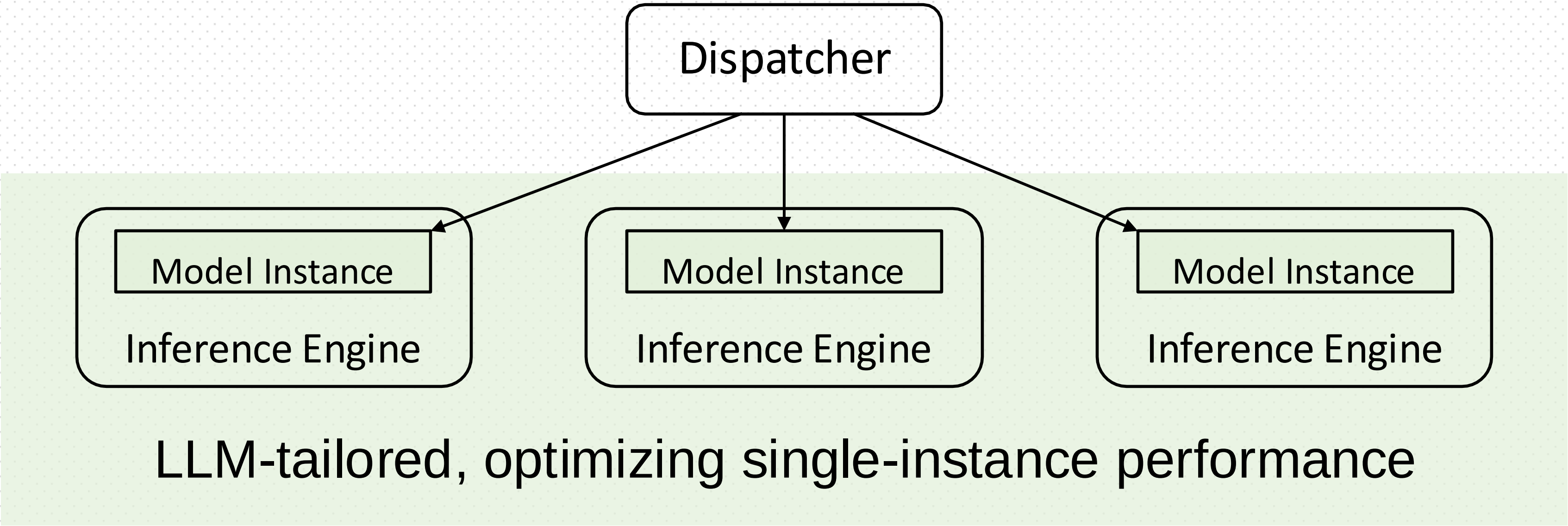
- A **request dispatcher** + *multiple instances* of an **inference engine**



LLM Serving Today: A Cluster Perspective

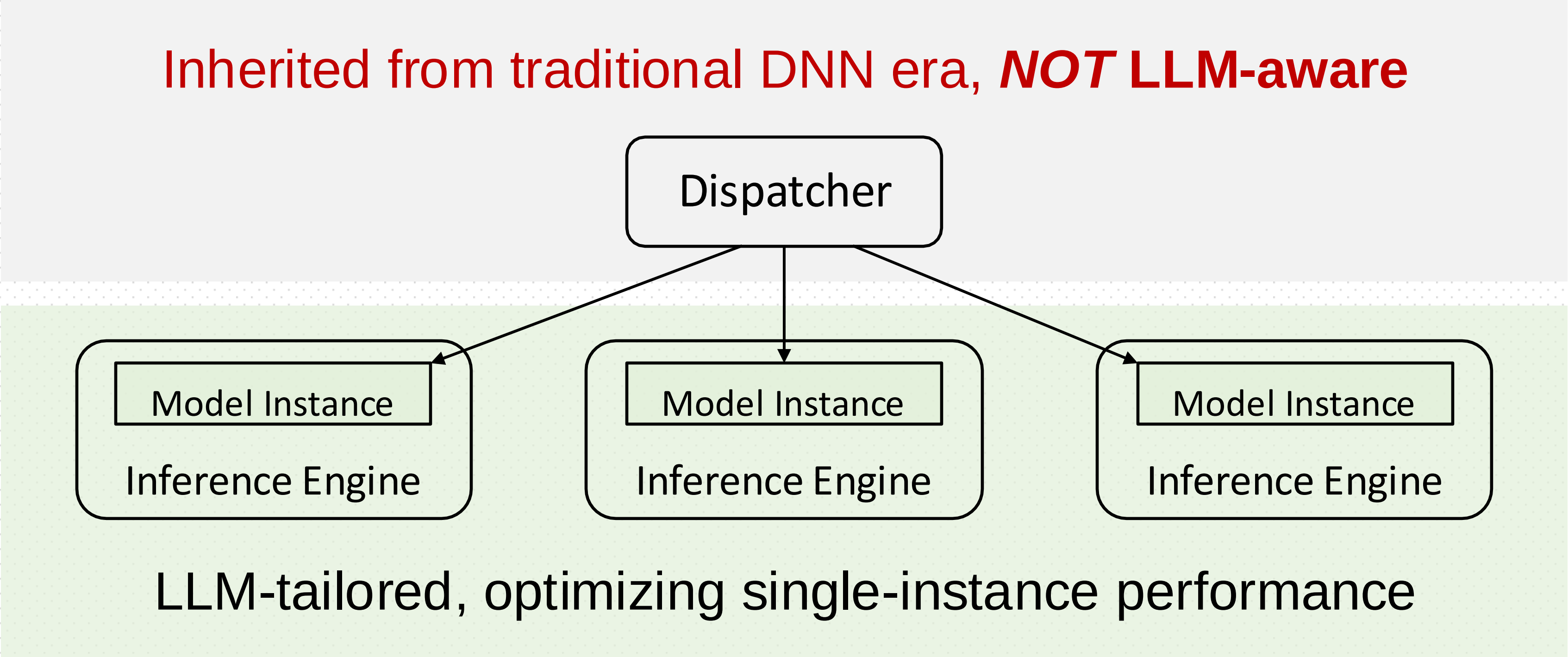
USTC, CHINA ADSLAB

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



LLM Serving Today: A Cluster Perspective

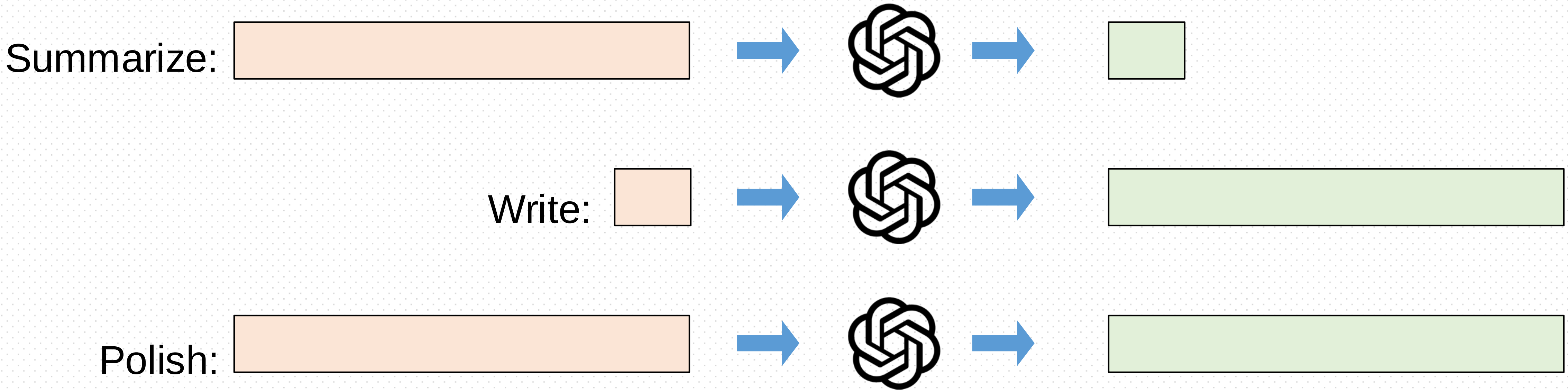
- A **request dispatcher** + *multiple instances* of an **inference engine**



LLM Characteristic (1): Workload Heterogeneity

- *Universal* models, *diverse* applications

- Requests are **heterogeneous**
  - *Sequence (input/output) lengths*



LLM Characteristic (1): Workload Heterogeneity

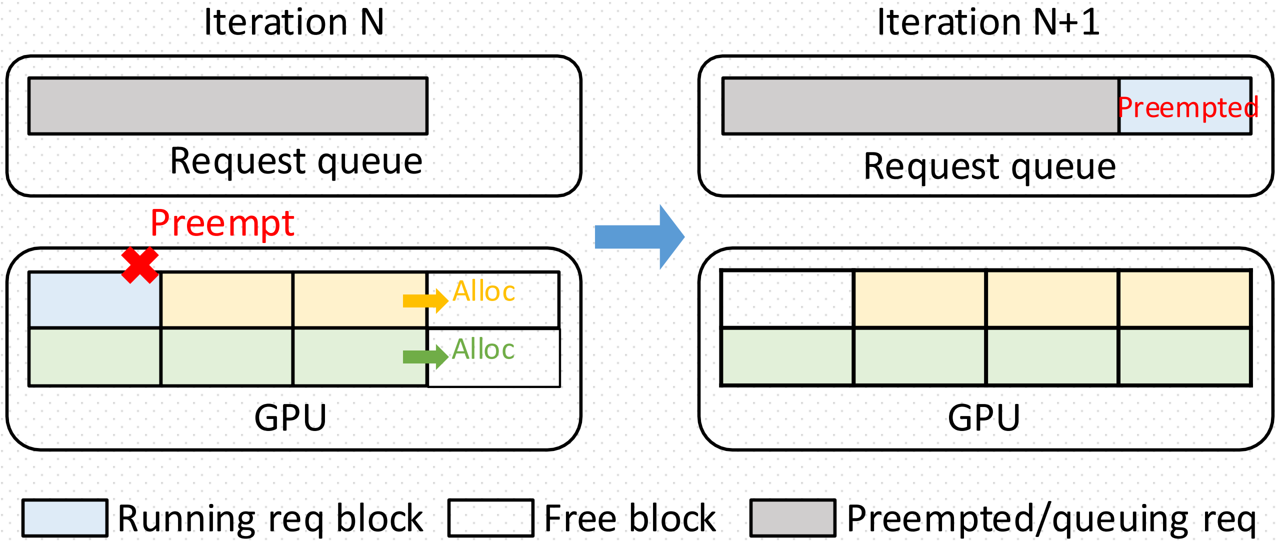
- *Universal* models, *diverse* applications

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

LLM Characteristic (2): Execution Unpredictability

USTC, CHINA
ADSLAB

- **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

USTC, CHINA ADSLAB

- Preemptions -> poor tail latencies
- Performance interference in a batch

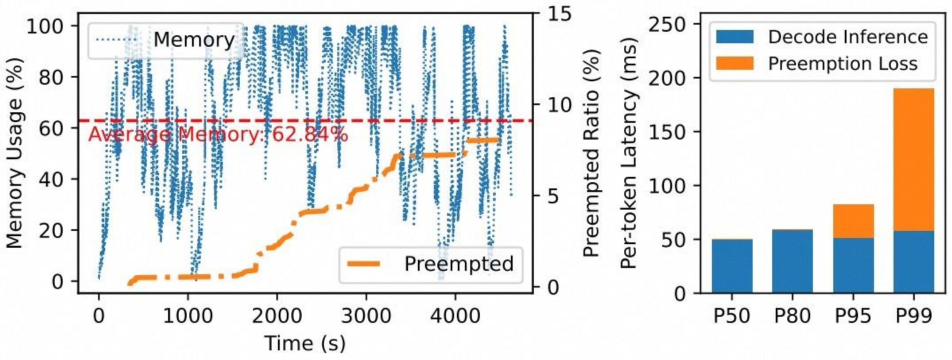


Figure 3: Request preemptions in LLaMA-7B serving.

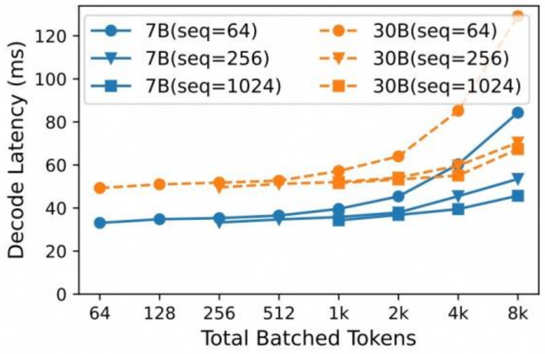


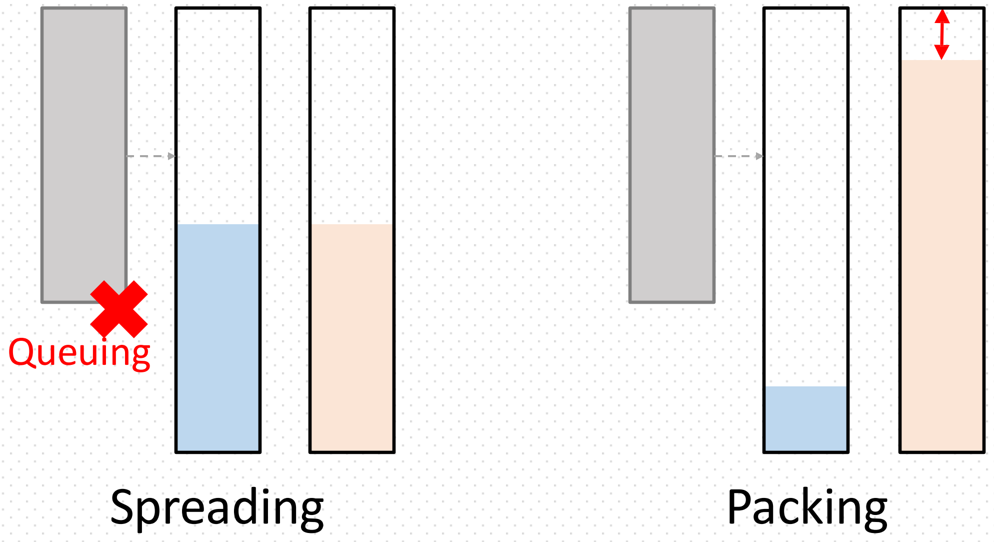
Figure 4: Latencies of one decode step of LLaMA-7B and LLaMA-30B with different sequence lengths and batch sizes.

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

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

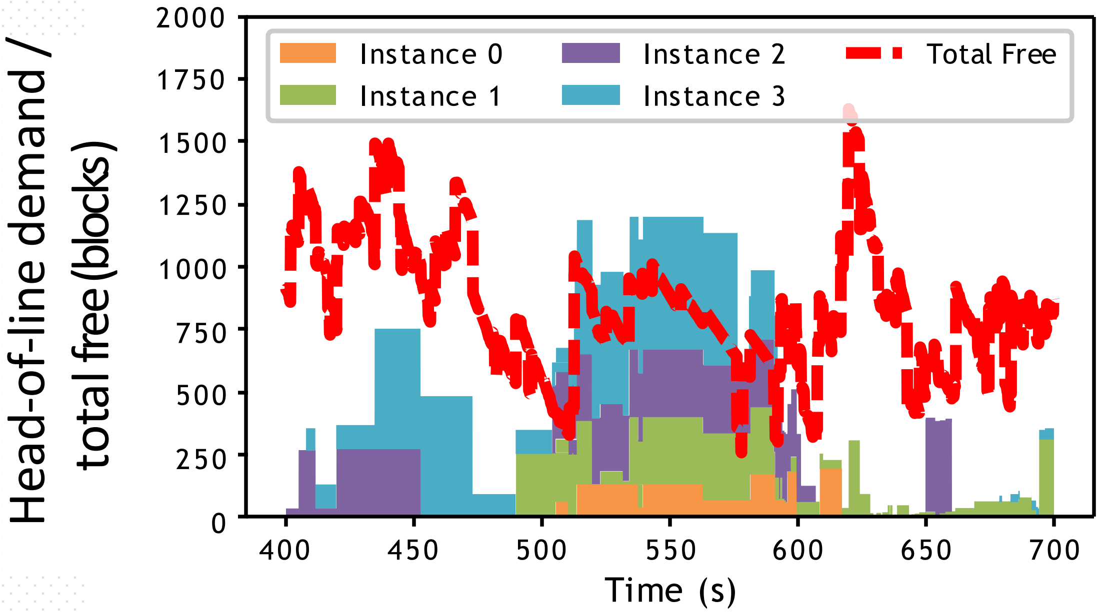
Challenge (2): Memory Fragmentation

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



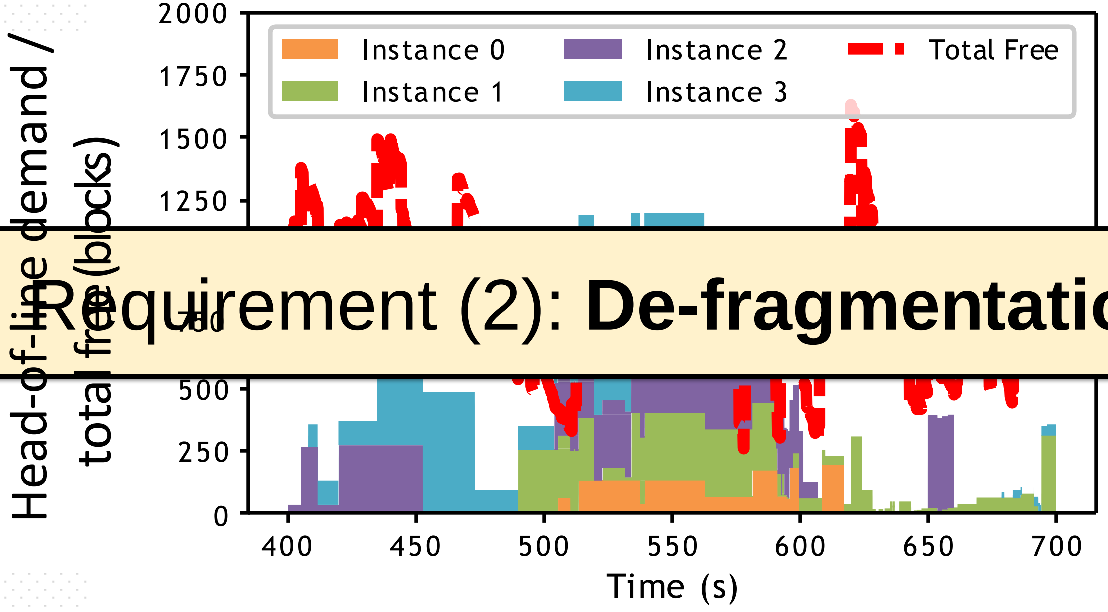
Challenge (2): Memory Fragmentation

- 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

USTC, CHINA ADSLAB

- 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…

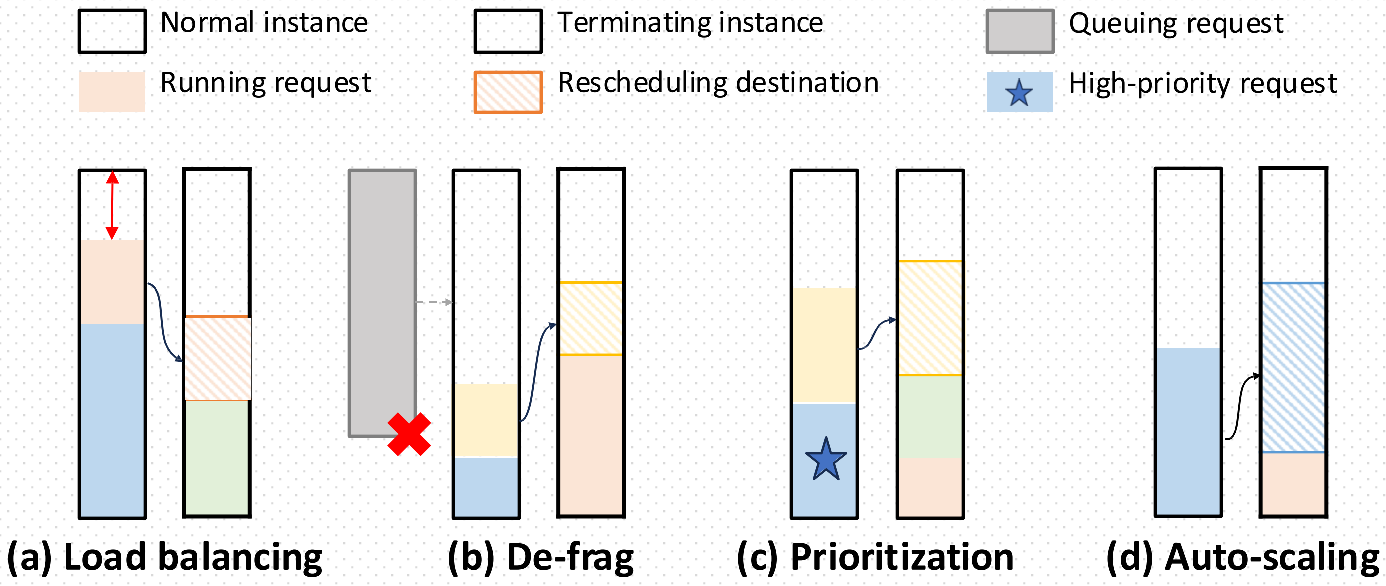
Not new in modern operating / distributed systems

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

Llumnix: Serving LLMs, the "OS" Way

USTC, CHINA ADSLAB

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



Design Goals

USTC, CHINA ADSLAB

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

USTC, CHINA ADSLAB

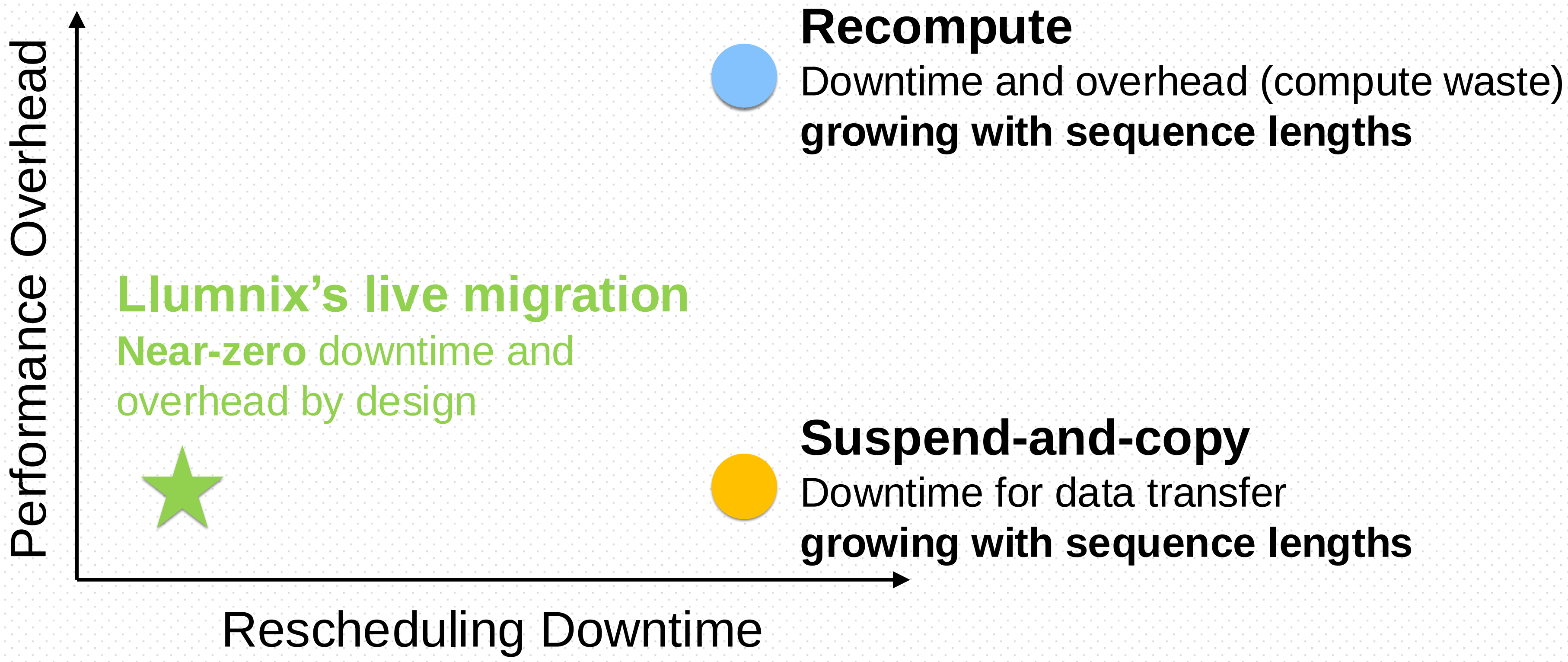
Aim: make rescheduling the *norm* in LLM serving

Efficiency → *Live migration mechanism*

Scalability → *Distributed scheduling architecture*

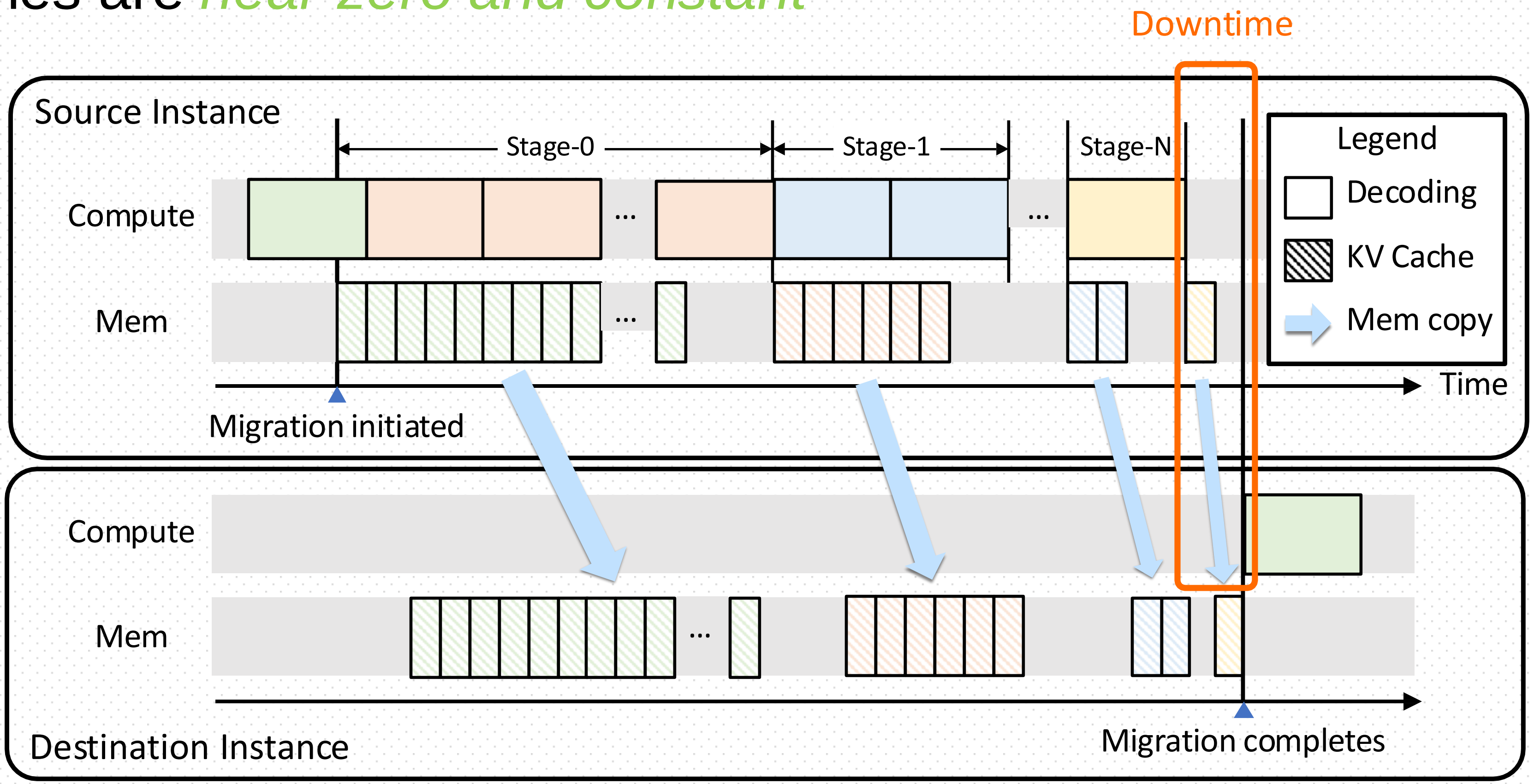
Scheduling Benefits → *Unified, multi-objective scheduling policy*

How to Reschedule KV Caches?



Live Migration of LLM Requests

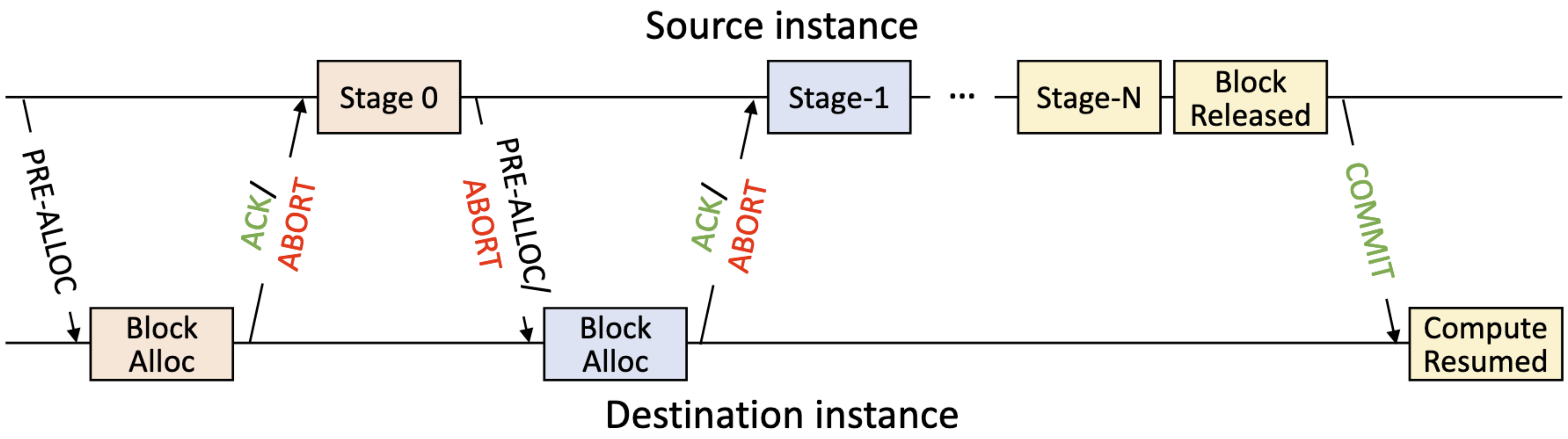
- KV caches are **append-only**
  - Copy *incremental* blocks iteratively
  - Downtimes are *near-zero and constant*



Live Migration of LLM Requests

USTC, CHINA ADSLAB

- LLM generation is unpredictable
  - Source and destination may run out of memory
  - Request can complete in the middle of migration
- Handshake during migration



Design Goals

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

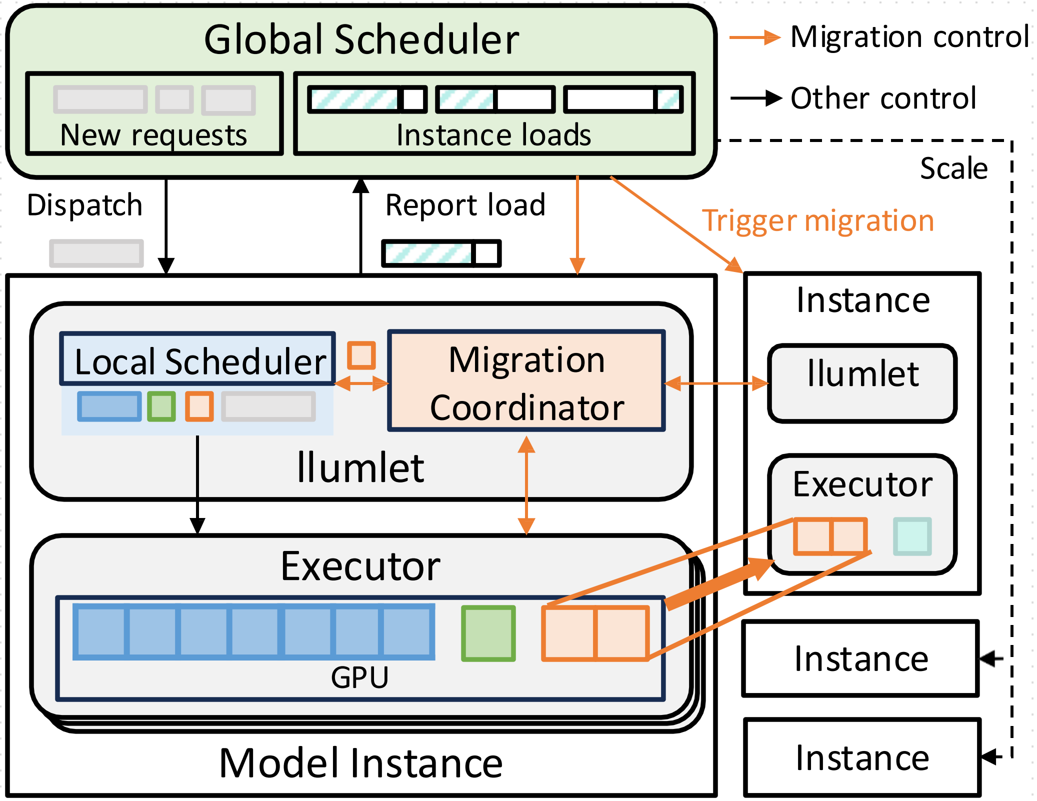
Efficiency → *Live migration mechanism*

Scalability → *Distributed scheduling architecture*

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

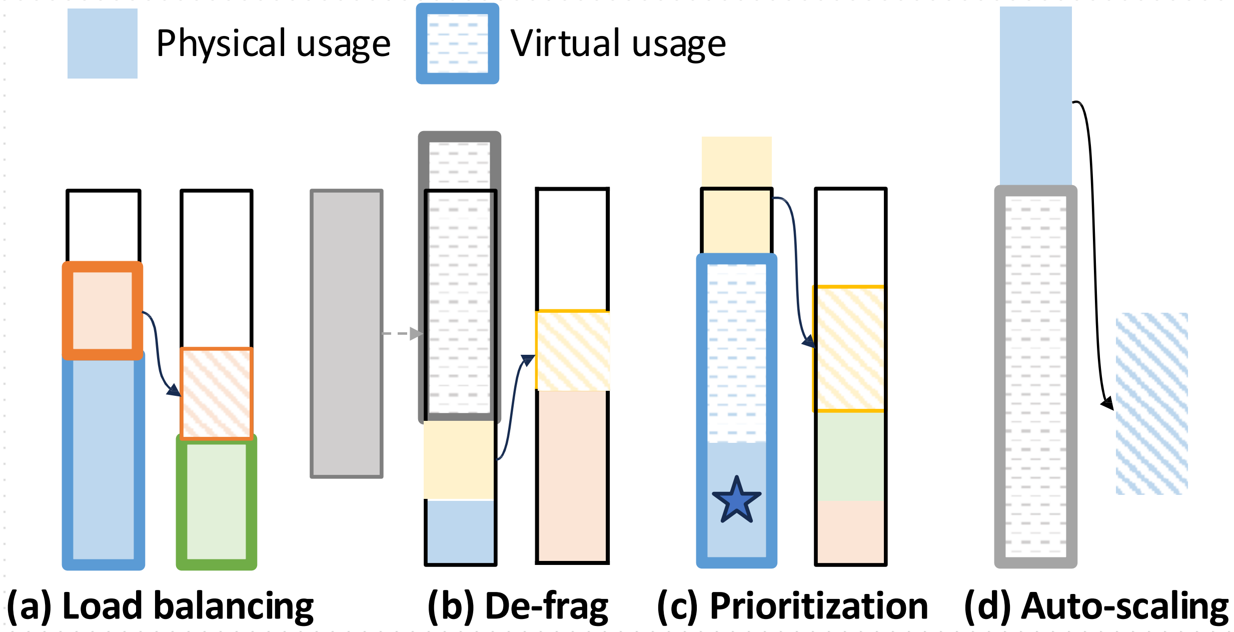
Distributed Scheduling Architecture

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



Scheduling Policy

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

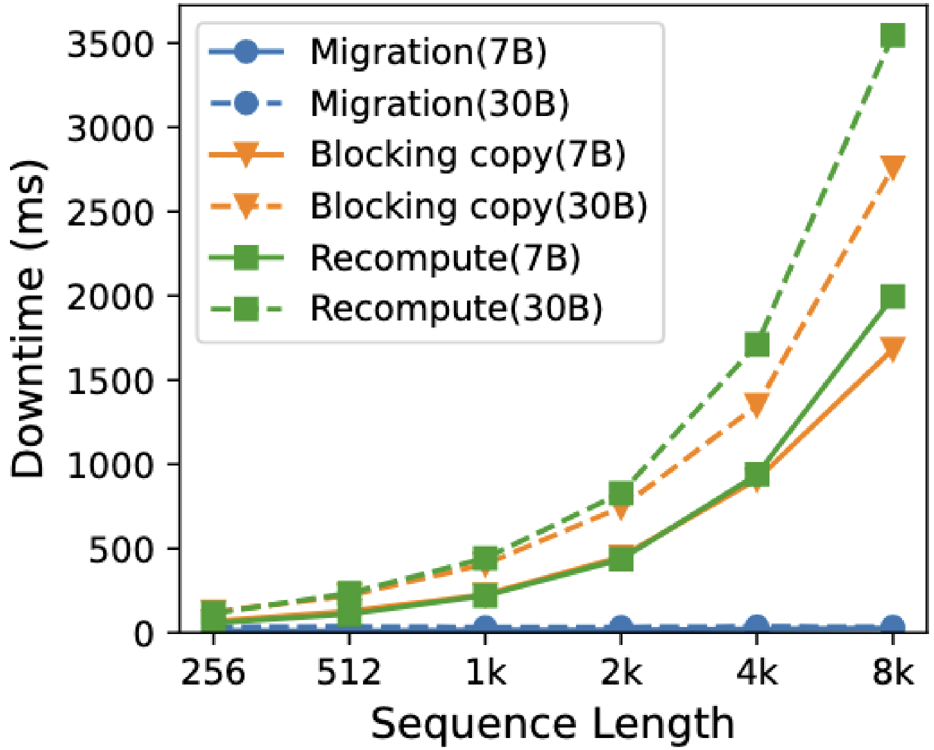


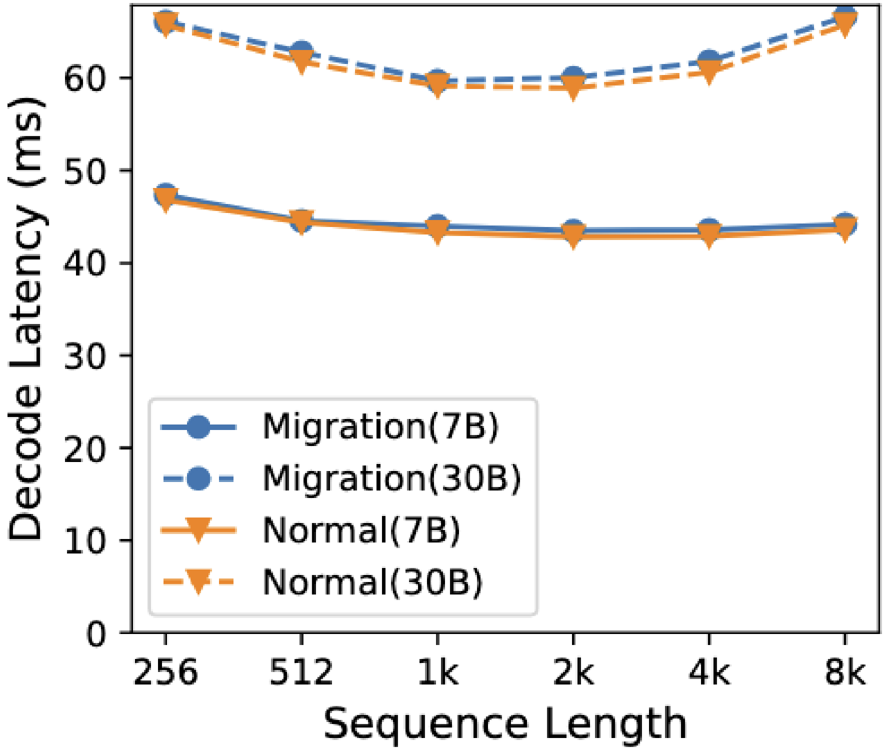
Scheduling Policy

- **Virtual usage**: unifying multiple objectives
  - Normal Case: virtual usage = physical memory usage
  - Queuing requests: virtual usage = real demand
  - Priority request: virtual usage = real demand + headroom
  - Terminate instance: send a fake request with a virtual usage of ∞

Evaluation: Migration Efficiency

- Up to 111x less downtime
- Up to 1% performance difference





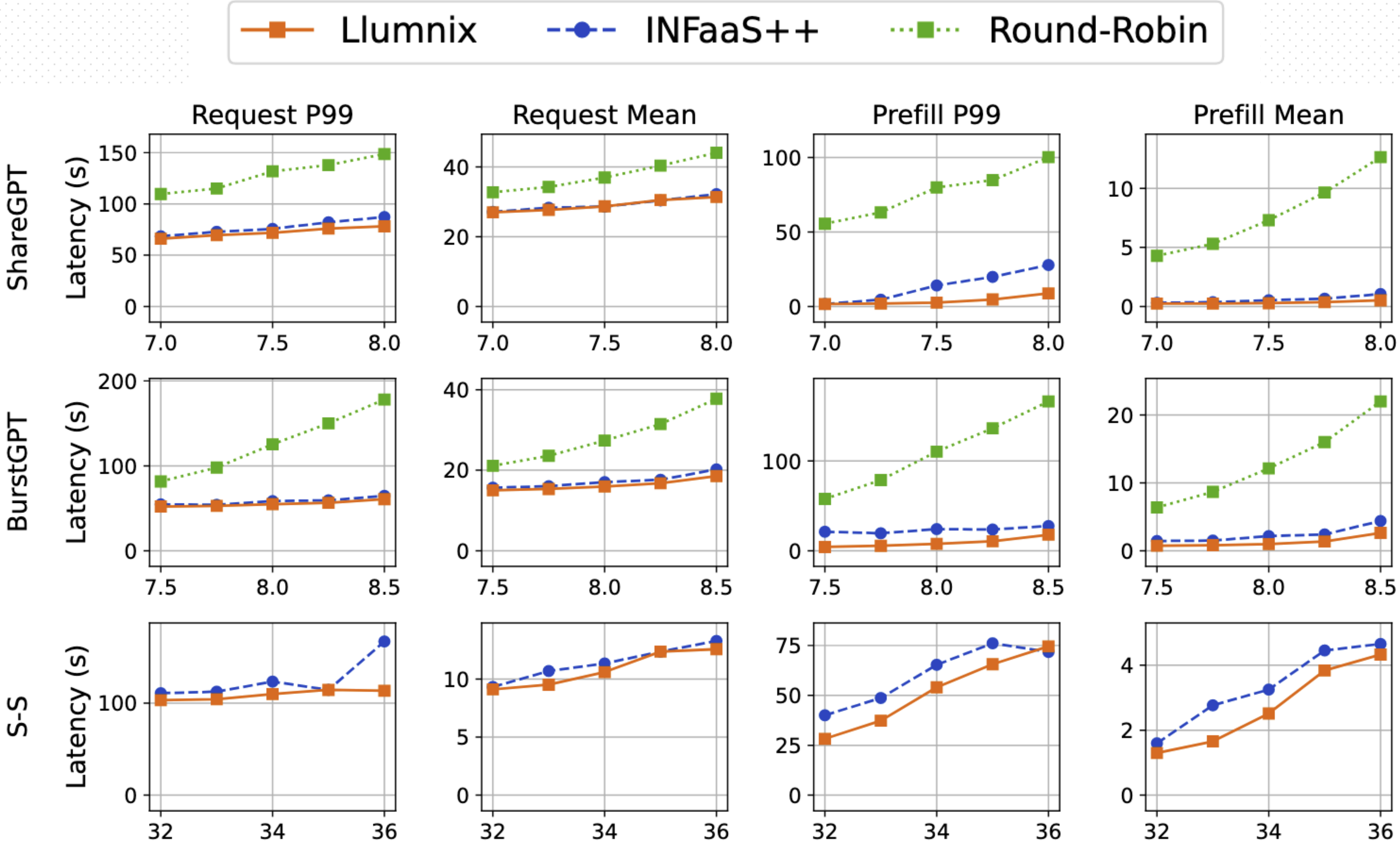
Evaluation: End-to-end Serving Performance

USTC, CHINA ADSLAB

- 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

Distribution			Mean	P50	P80	P95	P99
Real	ShareGPT	In	306	74	348	1484	3388
		Out	500	487	781	988	1234
	BurstGPT	In	830	582	1427	2345	3549
		Out	271	243	434	669	964
Gen	Short (S)		128	38	113	413	1464
	Medium (M)		256	32	173	1288	4208
	Long (L)		512	55	582	3113	5166

Evaluation: End-to-end Serving Performance

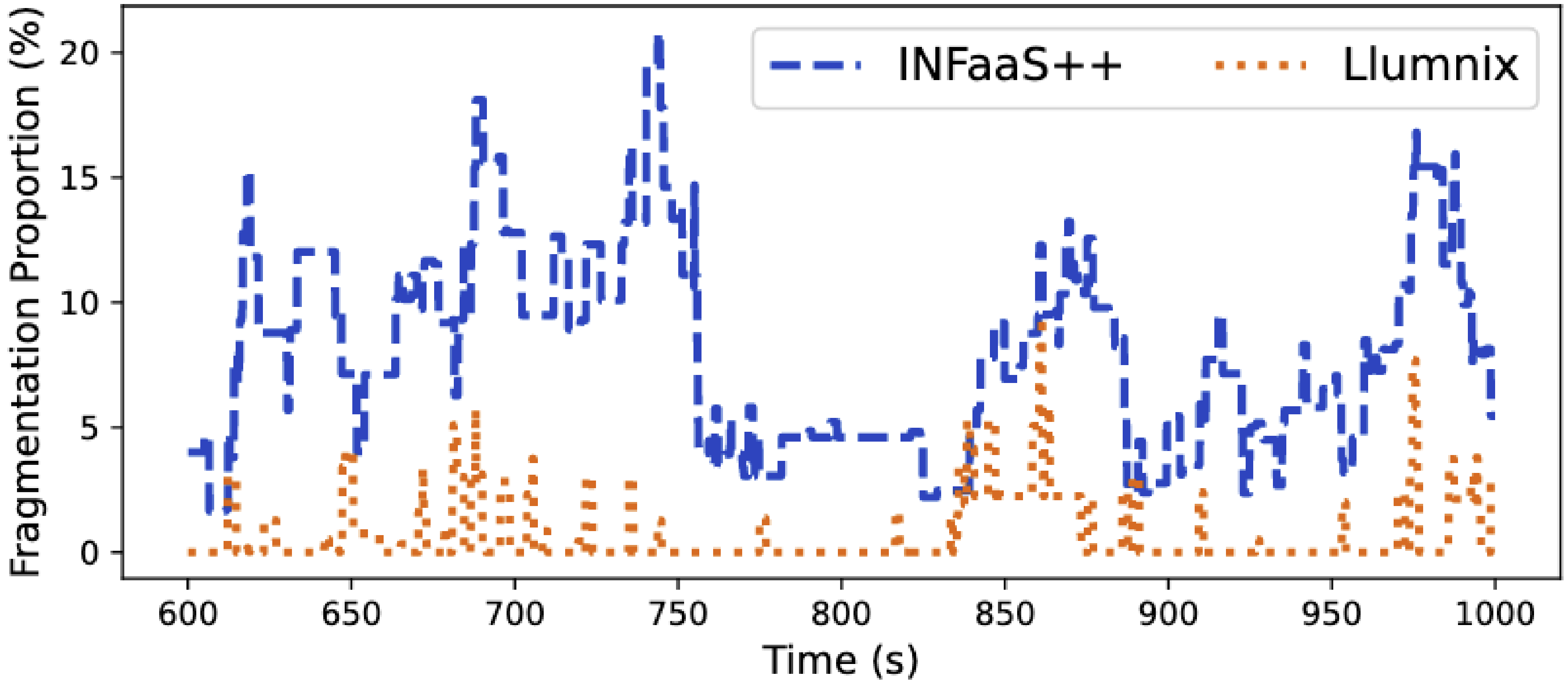


Evaluation: End-to-end Serving Performance

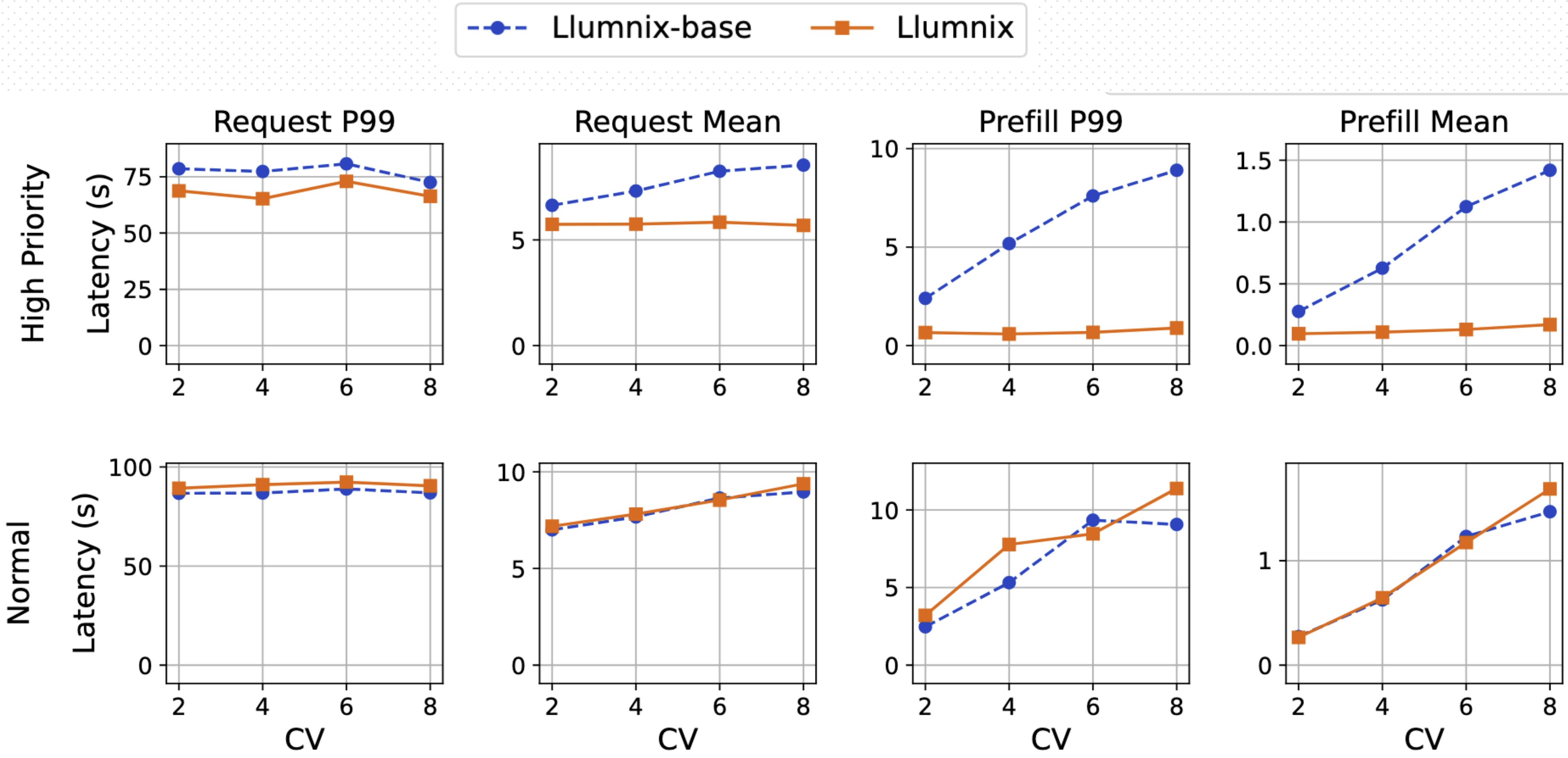
USTC, CHINA ADSLAB

- 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

Evaluation: Memory Fragmentation



Evaluation: Prioritization



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

Q&A