PreServe: Intelligent Management for LMaaS Systems via Hierarchical Prediction

Zhihan Jiang, Yujie Huang, Guangba Yu, Junjie Huang,

Jiazhen Gu, Michael R. Lyu

The Chinese University of Hong Kong

ARISE
Automated Reliable Intelligent Software Engineering

香港中文大學
The Chinese University of Hong Kong

Accepted by ICSE'26

Background

The Language Model-as-a-Service (LMaaS) paradigm has been widely adopted to deliver diverse LLMs to users in everyday applications.

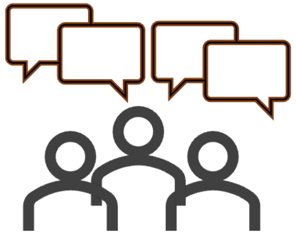
Gemini DeepSeek

ChatGPT Qwen

LLM Services

OpenAPI

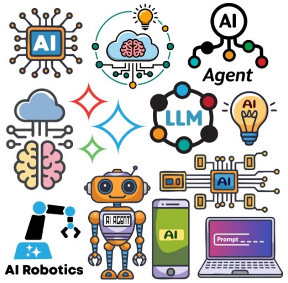
Web Interface



Users

OpenAPI

Web Interface

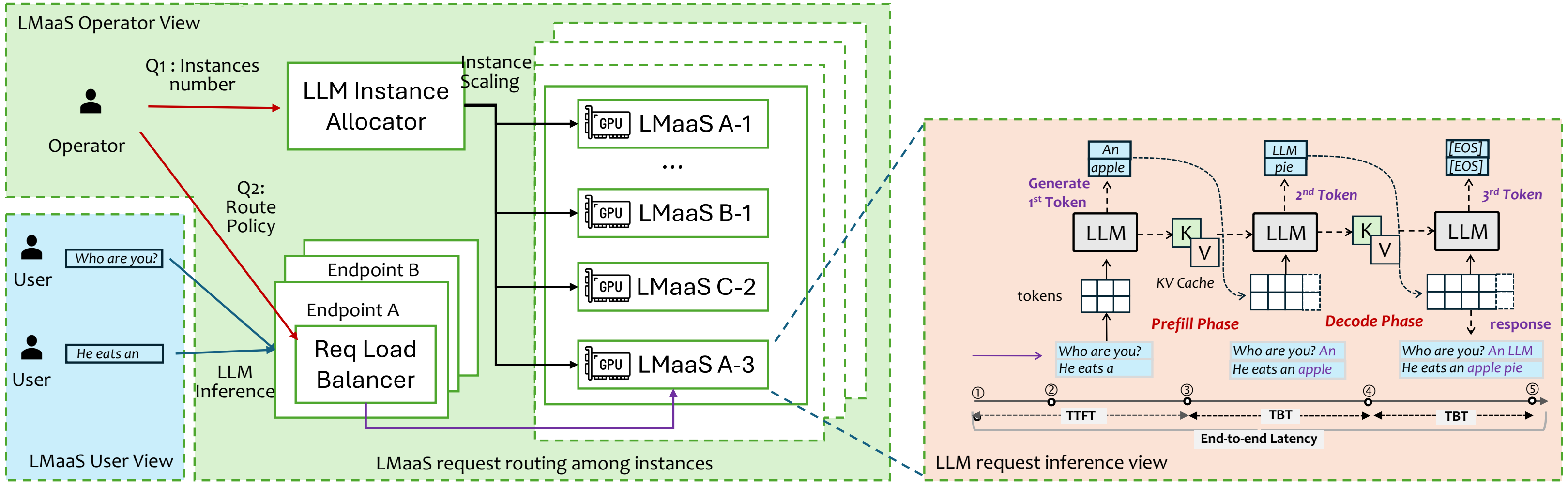


LLM Applications

millions of queries per day

Motivation

Effective management of LMaaS platforms is critical.



Example of LMaaS Management Platform

Goal of LMaaS Management

Auto-scaling LLM instances:

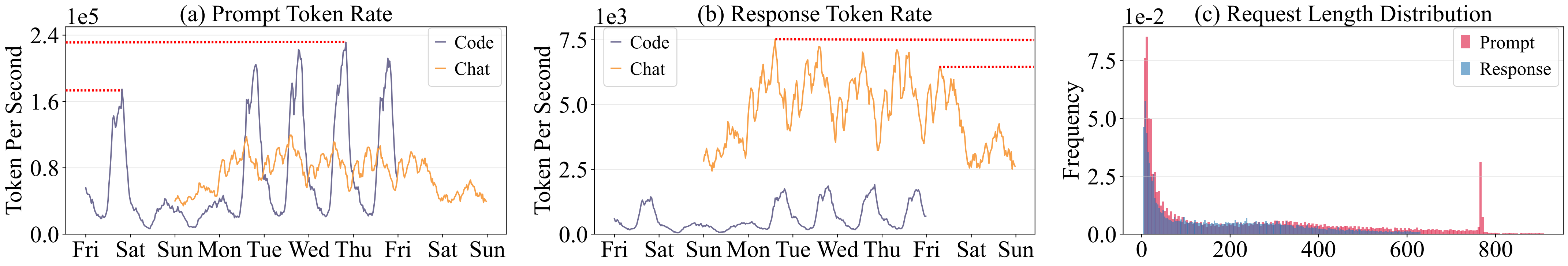
- avoid resource under- or over-provisioning

Routing LLM requests:

- ensure load balance across instances

Challenges

LMaaS exhibits distinct characteristics that introduce new management challenges.



Real-world LLM Service Workload from Azure

High Service workload variability

- Substantial TPS fluctuations
- Unpredictable peak demands
- Long cold-start issues

High request load variance

- Prefill and decode stress different resources
- Request load varies by orders of magnitude
- Resource demand increases during generation

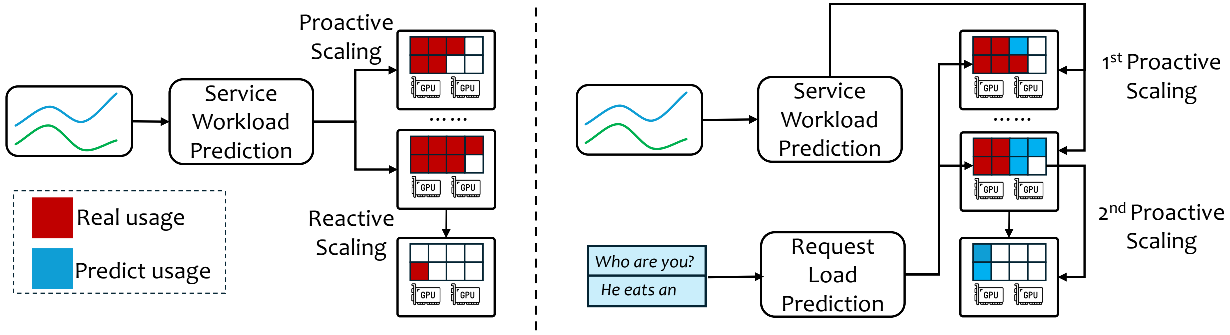
Traditional service management techniques are not suitable for LMaaS.

Opportunities

Hierarchical (global and local) prediction for proactive management.

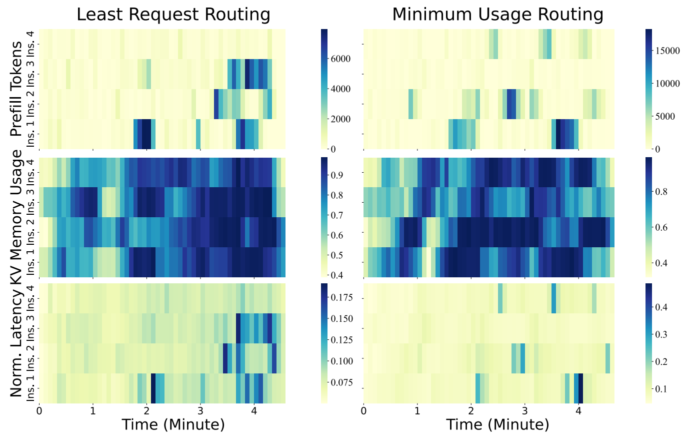
Service-level Workload Prediction
- forecast aggregate demand from historical TPS
- pre-scale resources for upcoming time windows

Request-level Load Prediction
- estimate load for individual LLM requests
- model resource trend of each LLM instance



(a) Proactive + Reactive Scaling Framework
(b) Hierarchical Proactive Scaling Framework

Different auto-scaling paradigms



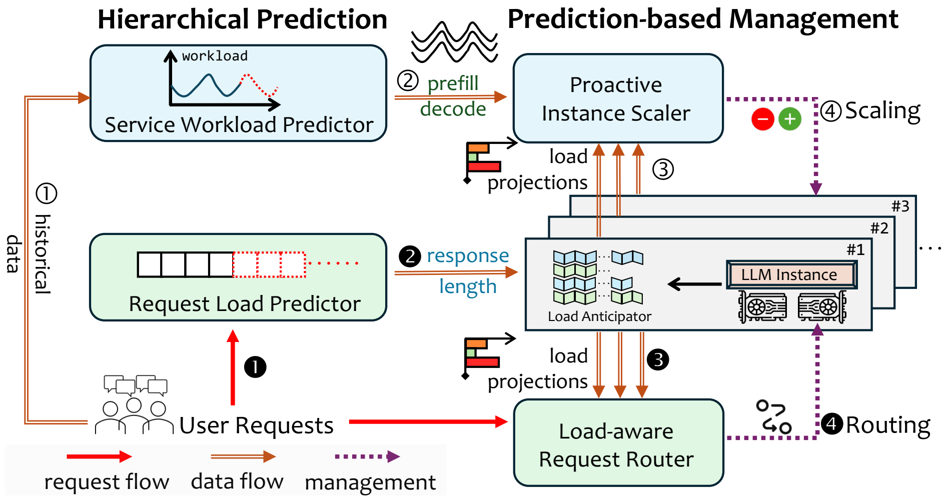
Different load balancing strategies

Methods

PreServe: a hierarchical prediction-based LMaaS management framework

Service-level Workload Prediction
- forecast aggregate demand from historical TPS
- pre-scale resources for upcoming time windows

Request-level Load Prediction
- estimate load for individual LLM requests
- model resource trend of each LLM instance



The overall framework of PreServe

Methods

PreServe: a hierarchical prediction-based LMaaS management framework

Service-level Workload Prediction

- forecast aggregate demand from historical TPS
- pre-scale resources for upcoming time windows

Request-level Load Prediction

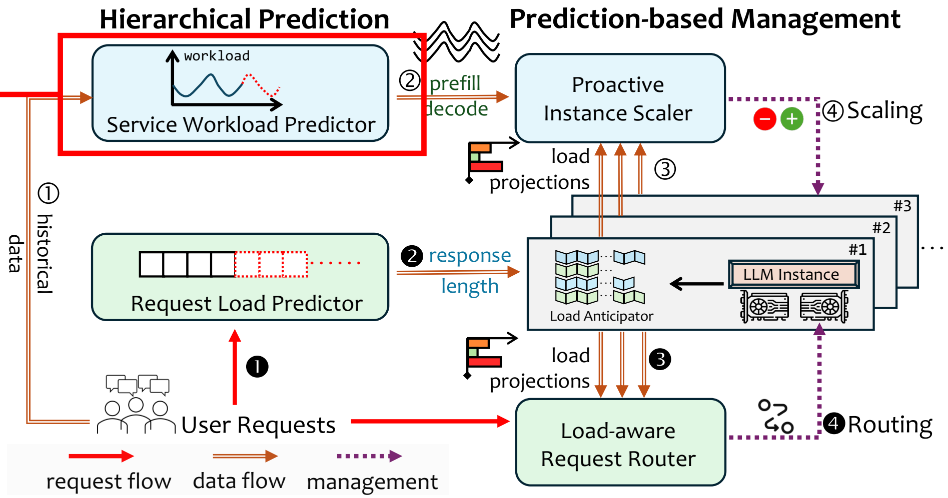
- estimate load for individual LLM requests
- model resource trend of each LLM instance

An mLSTM model captures long-term trends to predict aggregate TPS over 10-minute intervals.

Algorithm 1: Offline Phase of Service Workload Prediction

Input: Historical requests $\mathcal{R}$ and time windows $\mathcal{T} = \{T_1, T_2, \cdots\}$
Output: Prediction model $\mathcal{M}_P$, $\mathcal{M}_D$ and profiled rate μ_p, μ_d, μ_t

for $T_i \in \mathcal{T}$ **do**
  Aggregate the prefill and decode tokens within $T_{i-k}, \cdots, T_i \Rightarrow$ Generate $\mathbf{P} = \{P_{i-k}, \ldots, P_i\}, \mathbf{D} = \{D_{i-k}, \ldots, D_i\}$
  $\mathbf{P} = \frac{\mathbf{P}-\min(\mathbf{P})}{\max(\mathbf{P})-\min(\mathbf{P})}$, $\mathbf{D} = \frac{\mathbf{D}-\min(\mathbf{D})}{\max(\mathbf{D})-\min(\mathbf{D})}$ `// Normalization`
  `/* Construct training datasets */`
  Add training pair $\{P_{i-k}, \ldots, P_{i-1}\} \rightarrow P_i$ to S_P
  Add training pair $\{D_{i-k}, \ldots, D_{i-1}\} \rightarrow D_i$ to S_D
  `/* Update the maximum serving capability */`
  if *all requests in T_i complete without an SLO violation* **then**
    $\mu_p = \max\left(\mu_p, \frac{\sum p(r) \in T_i}{|T_i|}\right), \mu_d = \max\left(\mu_d, \frac{\sum d(r) \in T_i}{|T_i|}\right)$
    $\mu_t = \max\left(\mu_t, \frac{\sum (p(r)+d(r)) \in T_i}{|T_i|}\right)$
use S_P to train $\mathcal{M}_P$ and use S_D to train $\mathcal{M}_D$



The overall framework of PreServe

Algorithm 2: Online Phase of Service Workload Prediction

Input: Historical aggregated token sequences:
$\mathbf{P} = \{P_{i-k}, \cdots, P_{i-1}\}, \mathbf{D} = \{D_{i-k}, \cdots, D_{i-1}\}$
Output: Estimated N_1 LLM instances for the next time window

for *each current time window T_i* **do**
  Predict current window tokens: $\hat{P}_i = \mathcal{M}_P(\mathbf{P}), \hat{D}_i = \mathcal{M}_D(\mathbf{D})$
  `/* Extend historical sequences */`
  $\mathbf{P}' = \mathbf{P} + \{\hat{P}_i\}, \quad \mathbf{D}' = \mathbf{D} + \{\hat{D}_i\}$
  Predict new window tokens: $\hat{P_{i+1}} = \mathcal{M}_P(\mathbf{P}'), \hat{D_{i+1}} = \mathcal{M}_D(\mathbf{D}')$
  `/* Determine the required number of instances */`
  $N_{i+1} = \max\left(\frac{\hat{P_{i+1}}}{\mu_p}, \frac{\hat{D_{i+1}}}{\mu_d}, \frac{\hat{P_{i+1}}+\hat{D_{i+1}}}{\mu_t}\right)$
  if T_i *has concluded* **then**
    Update historical sequences: $\mathbf{P} \leftarrow \mathbf{P} + \{P_i\}, \mathbf{D} \leftarrow \mathbf{D} + \{D_i\}$

Methods

PreServe: a hierarchical prediction-based LMaaS management framework

Service-level Workload Prediction

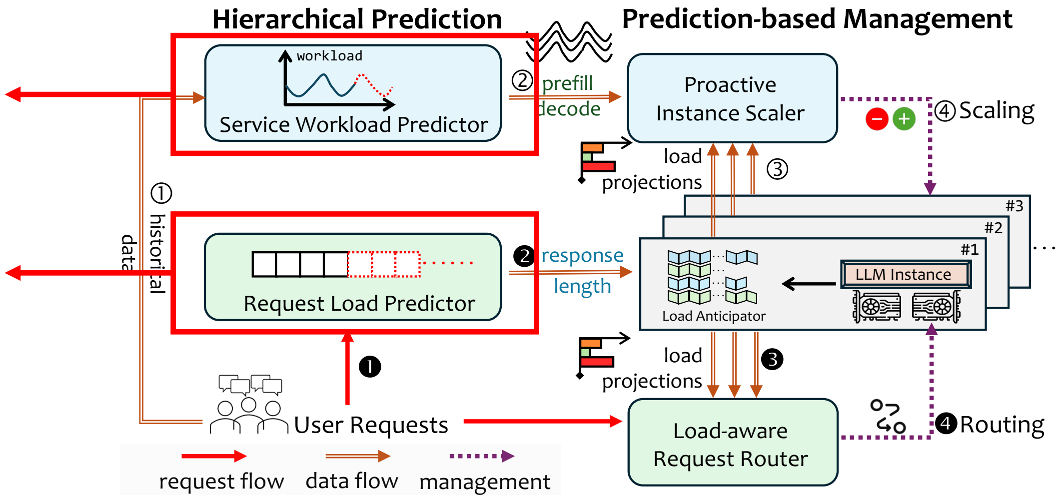
- forecast aggregate demand from historical TPS
- pre-scale resources for upcoming time windows

Request-level Load Prediction

- estimate load for individual LLM requests
- model resource trend of each LLM instance

An mLSTM model captures long-term trends to predict aggregate TPS over 10-minute intervals.

An DistilBert model estimates the individual request load based on semantics in real time



The overall framework of PreServe

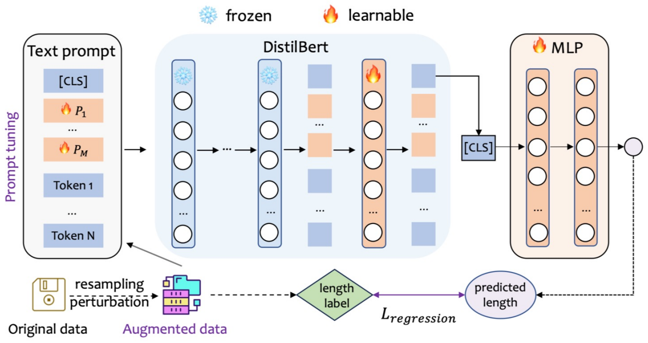


Figure 6: Request Load predictor training in PreServe.

Methods

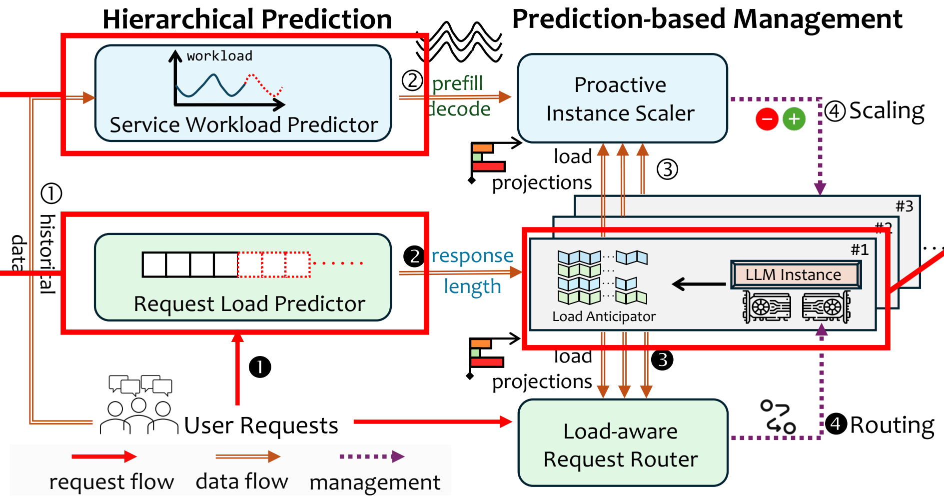
PreServe: a hierarchical prediction-based LMaaS management framework

Service-level Workload Prediction
- forecast aggregate demand from historical TPS
- pre-scale resources for upcoming time windows

Request-level Load Prediction
- estimate load for individual LLM requests
- model resource trend of each LLM instance

An mLSTM model captures long-term trends to predict aggregate TPS over 10-minute intervals.

An DistilBert model estimates the individual request load based on semantics in real time



The overall framework of PreServe

Construction of local load forecasts for each instance

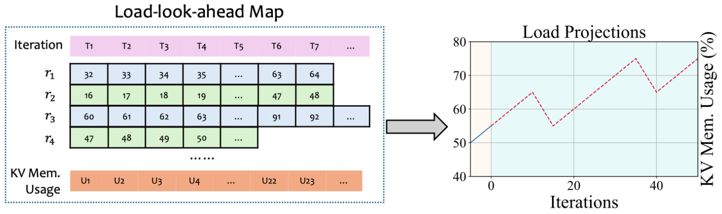
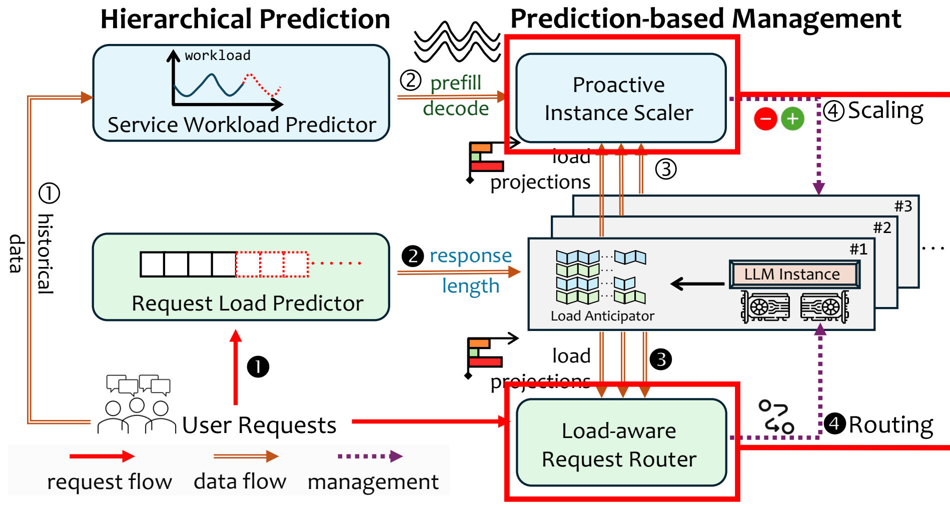


Figure 7: The load anticipator within each LLM instance.

Methods

PreServe: a hierarchical prediction-based LMaaS management framework



The overall framework of PreServe

Proactive Instance Scaler

- combine both long- and short-term resource demands
- estimates optimal instance count for auto-scaling

Load-aware Request Router

- consider both current and anticipated future loads
- determine the optimal request using "what-if" analysis

Improving resource utilization while ensure load-balancing

Experiments

Settings

- Framework: vLLM
- Model: LLaMA-2-7B and 13B
- Workload:
  - Azure LLM inference trace
  - ShareGPT datasets

RQ1: Hierarchical Prediction Accuracy

Table 1: Mean and maximum absolute percentage error (APE) of workload prediction in Azure datasets (10min-window).

Methods	Mean APE				Max APE			
	Azure-code		Azure-chat		Azure-code		Azure-chat	
	prompt	response	prompt	response	prompt	response	prompt	response
ARIMA	59.17%	61.44%	15.94%	16.12%	91.00%	90.18%	74.03%	82.09%
ETS	54.63%	55.55%	15.93%	16.10%	86.71%	83.54%	73.95%	81.96%
Prophet	26.26%	28.49%	8.05%	8.28%	67.88%	62.27%	27.30%	25.12%
PreServe	**7.74%**	**8.45%**	**4.15%**	**4.30%**	**26.25%**	**30.30%**	**21.16%**	**19.88%**
average	**8.10%**		**4.23%**		**28.28%**		**20.52%**	
	6.17%				**24.40%**			

Table 2: The response length prediction accuracy (up to 4096).

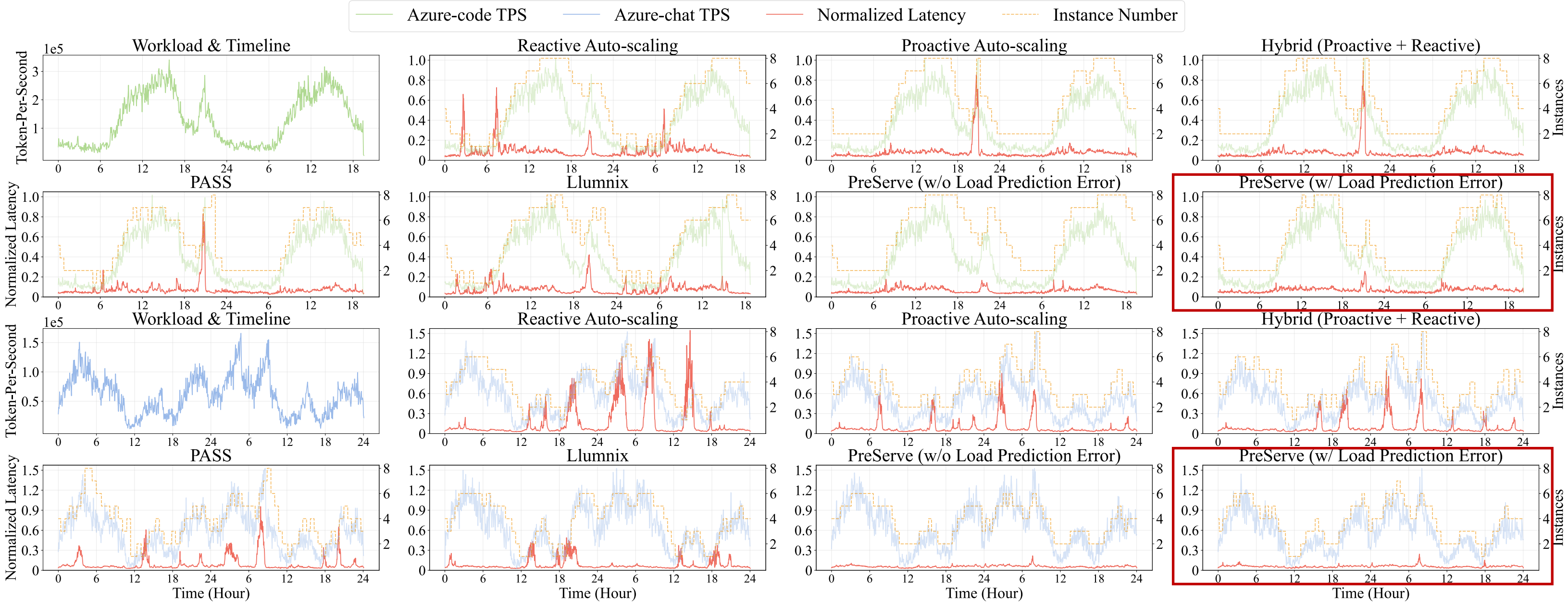
Methods	MAE	Acc-25	Acc-50	Acc-100
μ-Serve	355.59	32.31%	49.35%	65.25%
PiA (Vicuna-13B)	283.86	39.27%	54.18%	68.56%
PiA (ChatGPT)	127.41	50.42%	61.25%	70.34%
PreServe	**78.25**	**56.77%**	**68.79%**	**77.95%**
Improvement	↑ (38.6%)	↑ (12.6%)	↑ (12.3%)	↑ (10.8%)

Achieves SOTA prediction accuracy in both levels

RQ2: Instance Scaling

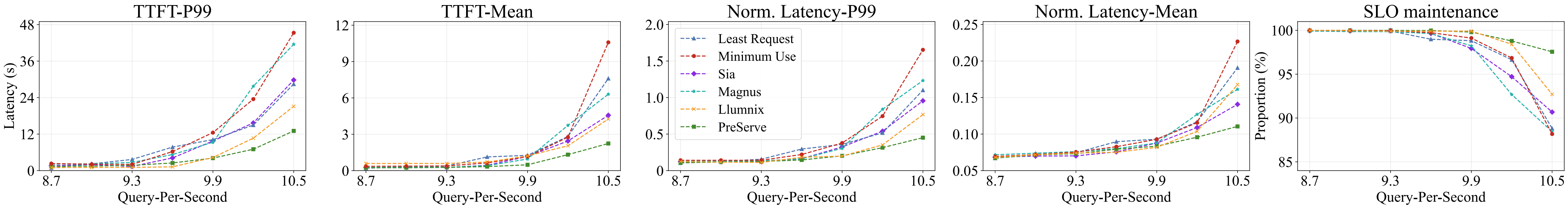
Reduces **peak normalized latency by 45.1%** than Llumnix [OSDI'24].

Cuts **resource usage by 49.4%** with minimal SLO violations (Llumnix: –48.3% but high violations)



Experiments

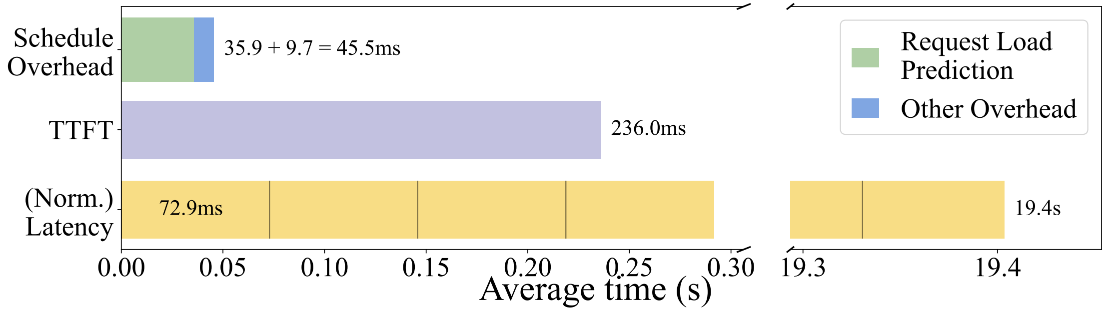
RQ3: Request Routing



Reduces **mean TTFT, P99 normalized latency, SLO violation rate by 47.4%, 41.3% and 66.58%** compared to Llumnix [OSDI'24].

RQ4: Management Efficiency

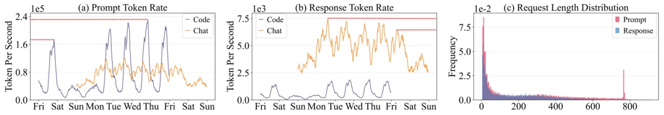
Introduces only 45.5 ms overhead on average, just **0.23% of request e2e latency**.



Conclusion

PreServe: Intelligent Management for LMaaS Systems via Hierarchical Prediction

LMaaS exhibits distinct characteristics that introduce new management challenges.



Real-world LLM Service Workload from Azure

High Service workload variability

- Substantial TPS fluctuations
- Unpredictable peak demands
- Long cold-start issues

High request load variance

- Prefill and decode stress different resources
- Request load varies by orders of magnitude
- Resource demand increases during generation

Traditional service management techniques are not suitable for LMaaS.

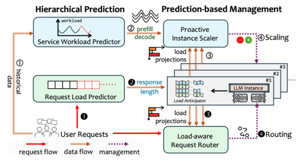
PreServe: a hierarchical prediction-based LMaaS management framework

Service-level Workload Prediction

- forecast aggregate demand from historical TPS
- pre-scale resources for upcoming time windows

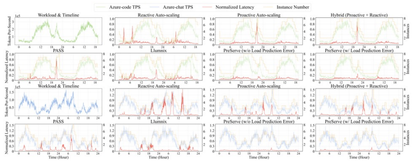
Request-level Load Prediction

- estimate load for individual LLM requests
- model resource trend of each LLM instance

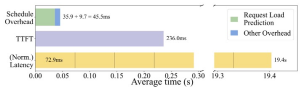


The overall framework of PreServe

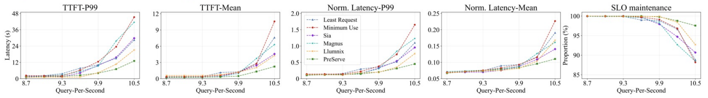
RQ2: Instance Scaling



RQ4: Management Efficiency



RQ3: Request Routing



Motivation **Method** **Results**

Pre-print paper

Artifact

Thank you for listening

Contact: zhjiang@link.cuhk.edu.hk

Accepted by ICSE'26