The Power of Prediction: Microservice Auto Scaling via Workload Learning

Shutian Luo[* 1,2,3], Huanle Xu[* 3], Kejiang Ye[1],
Guoyao Xu[4], Liping Zhang[4], Guodong Yang[4] and Chengzhong Xu[3]

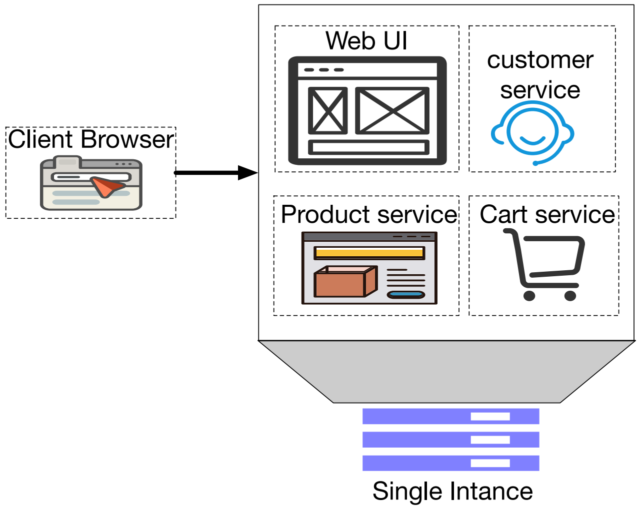
[1]SIAT, CAS
[2]University of CAS
[3]University of Macau
[4]Alibaba Group

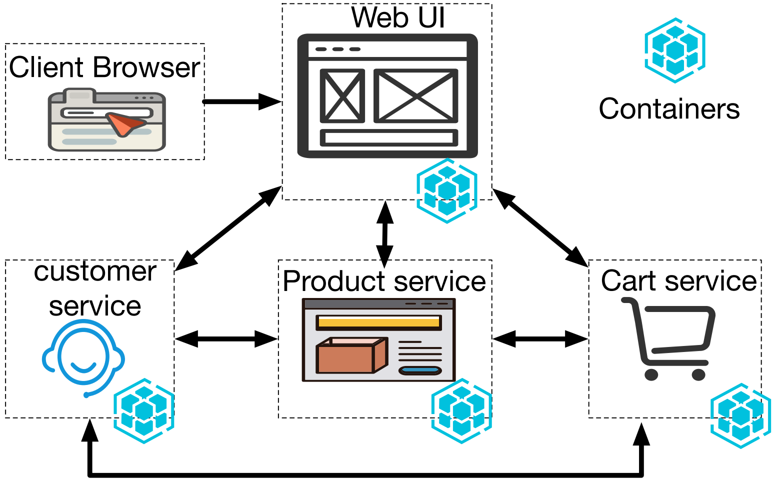
Outline

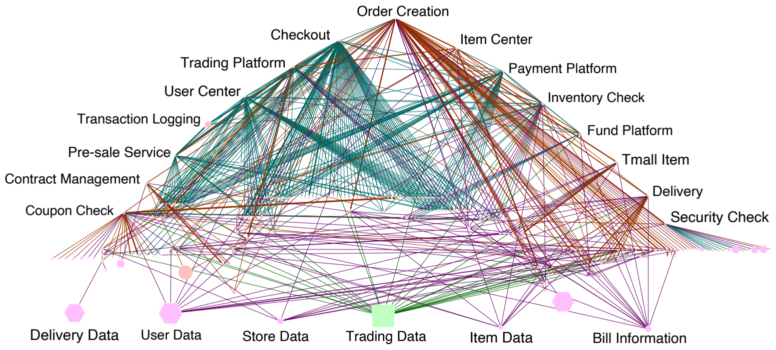
- Background
- Problem and Challenges
- Design of Madu
- Evaluation
- Summary

from Monolith to Microservices (MS)

- A monolithic application can be divided to a set of light-weight and loosely-coupled MSs







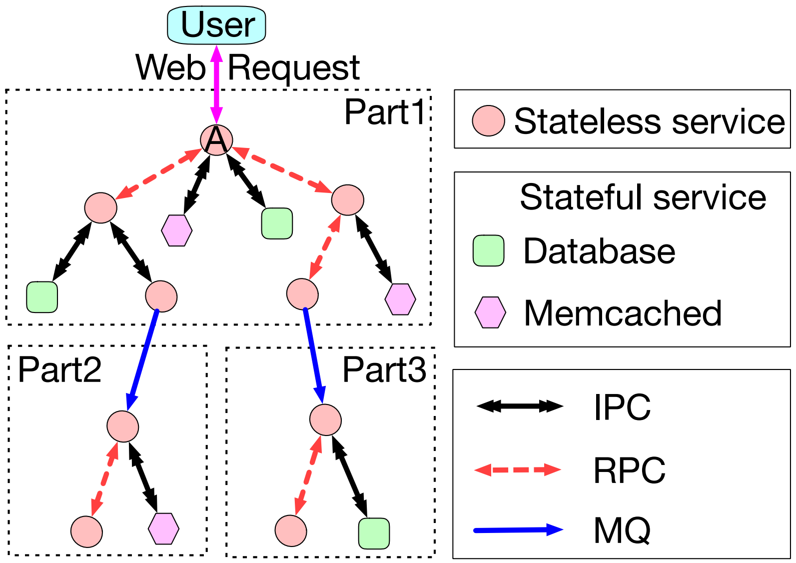
Complicated MSs dependencies graph

- It is easy to manage MS architecture.
  - Scale MSs independently instead of scaling the whole application.

MS Dependency Graph (DG)

- MS DG of an online service
  - Calls between MS triggered a request form a graph.

- End-to-end latency of an online service
  - From user sending a request to it receiving the reply.



Problem

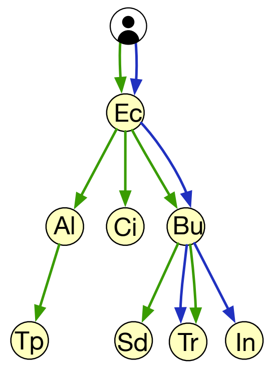
- MS is over-provisioned
  - Meet peak resource demand to satisfy service level agreements (SLA)
    - The average of resource utilization is less than 10%. MS Trace Analysis [SoCC'21]

Reactive Auto-scaler

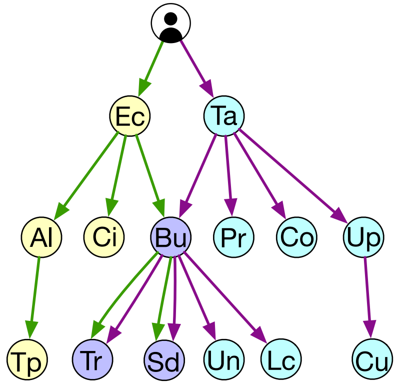
- Use feedback control to tune resources. SHOWAR[SoCC'21], Pema[HPDC'22]
- Perform unsatisfactorily under MS frameworks
  - Delayed queueing effect.
    - MS at the bottom of long MS chain cannot experience the change of workload immediately.
  - Scaling each MS requires fetching container images from the repository.
    - Take seconds to complete.

Proactive Auto-scaler

- Predict end-to-end latency based on DG. Sinan[ASPLOS'21], DeepRest[EuroSys'22]



- Do not consider two distinct characteristics of MS
  - Dynamic DG
    - Requests from the same online service can go through different sets of MS.
  - MS multiplexing
    - 5% of MS are shared by 90% of online services. MS Trace Analysis [SoCC'21]
    - Online services have different workload pattern and SLA requirement.

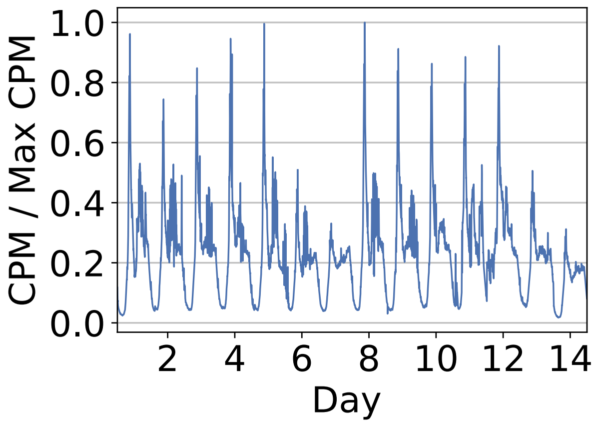


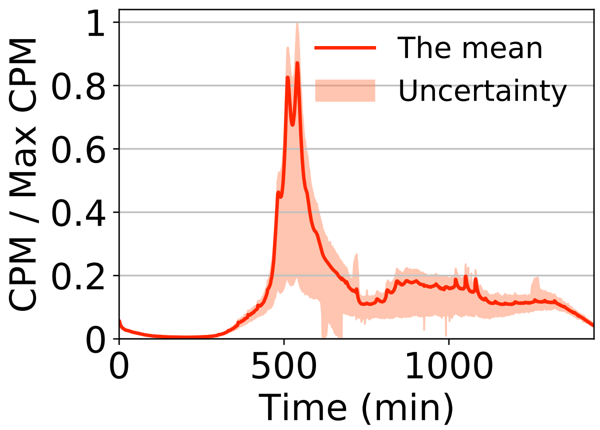
- Predict the performance of each individual MS [Our system Madu]
  - Avoid modelling dynamic DG and shared MS
  - Achieving accurate prediction is highly dependent on the knowledge of MS workload.

Challenge

➢ MS workload is periodic but has varying degrees of uncertainty.

- Uncertainty is the variance of calls per minute (CPM) at the same moment across different periods.
- Peak workload has higher uncertainty.

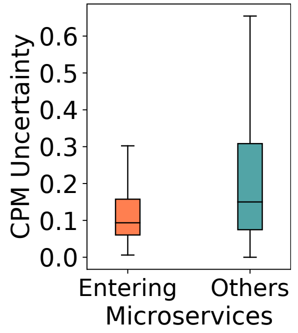




Observations in Workload Uncertainty

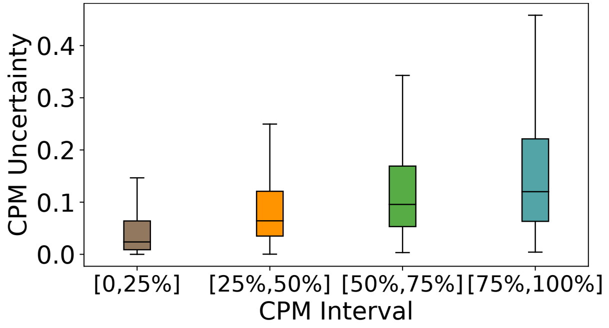
➢ Uncertainty is mainly caused by the dynamic DG

- Uncertainty of non-entering MS at peak workloads is much higher (2×) than that of entering MS



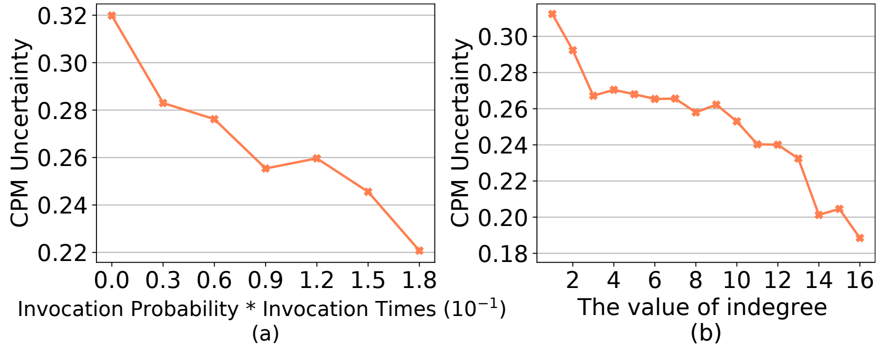
➢ Strong data-dependent uncertainty

- Variance of workloads across periods is related to the mean workload



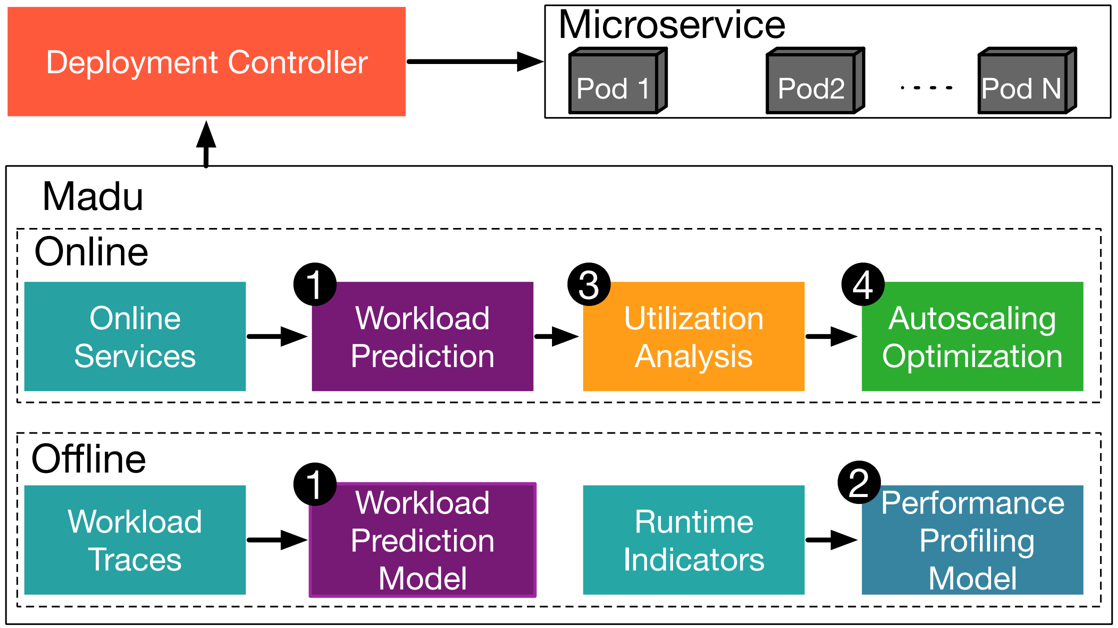
➢ Non-uniform workload uncertainty

- Depends on specific dynamic dependencies
- Fine-grained workload prediction for each MS



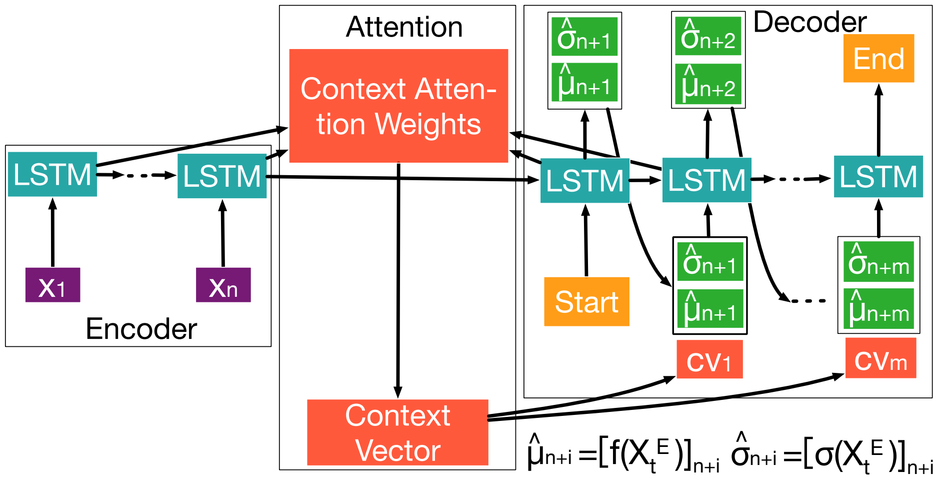
Design of Madu

➢ System overview



Workload Prediction

- ➢ Data-dependent uncertainty learning
  - Incorporate data uncertainty into the loss function
- ➢ Stochastic attention mechanism
  - Input data has similar uncertainty patterns

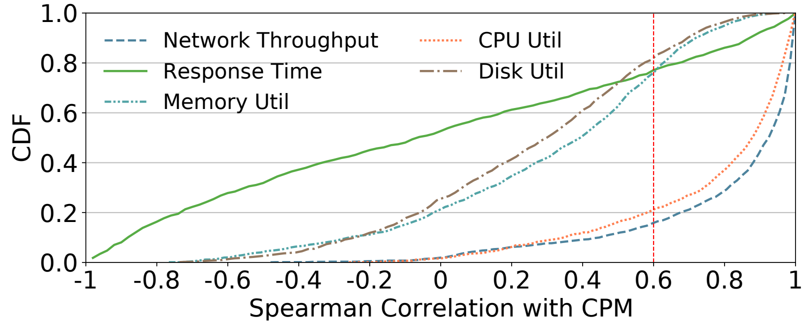


- ➢ Incorporate uncertainty into final prediction result

Performance Profiling

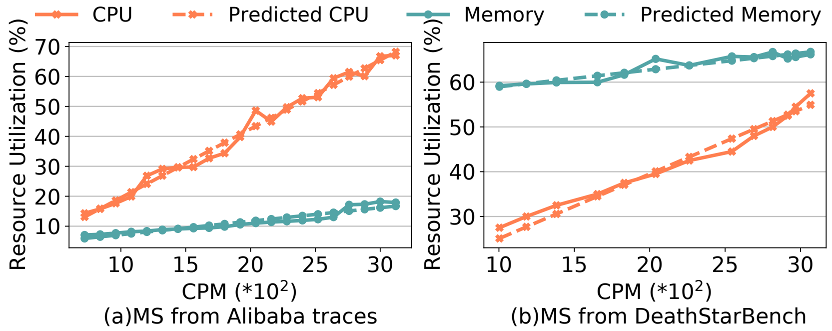
➢ Performance metrics

- CPU and memory utilization are much more strongly correlated with workloads than MS response time.



➢ Estimate resource usage based on predicted workload

- CPU and memory utilization of MS containers grows almost linearly in CPM.



Utilization Analysis

- Optimal resource allocation
  - Minimize the allocated resource based on predefined performance threshold.

$$
\begin{aligned}
\min_{c_i(t)\in\mathcal{N}} \quad & c_i(t) \\
\text{s.t.} \quad & g_i^{CPU}\big(L_i(t)/c_i(t)\big) \le T_i^{CPU}, \\
& g_i^{Mem}\big(L_i(t)/c_i(t)\big) \le T_i^{Mem},
\end{aligned}
$$

$c_i(t)$: allocated resource for MS i, $g(*)$: resource utilization estimation, T : predefined threshold

Autoscaling Optimization

- Avoid frequent scaling

- Minimize Scaling overhead
  - Target: minimize the scaling containers in the following m interval
  - Constraint: guarantee MS performance and ensure high utilization
    - ρ is a parameter that balances the performance and utilization trade-off.

$$\min_{\boldsymbol{x}_i} \sum_{k=1}^{m} \big(x_i(t+k-1) - x_i(t+k)\big)^2$$
$$\text{s.t., } c_i(t+k) \le x_i(t+k) \le (1+\rho) \cdot c_i(t+k).$$

$c_i(t)$: the minimum number of container for MS i in time t

Experiment Setup

- Benchmark: DeathStarBench

- Cluster: A local K8s cluster with 20 two-socket physical node

- Workload Generated from Alibaba traces
  - Traces will be released soon.

- Baseline Schemes
  - Reactive auto-scaler: K8S HPA, Google Autopilot[EuroSys'20]
  - Proactive auto-scaler: *Seq2Seq*, *DUBNN*[NeurIPS'17] , *BNN*[NeurIPS'19], ARIMA

Workload Predictor

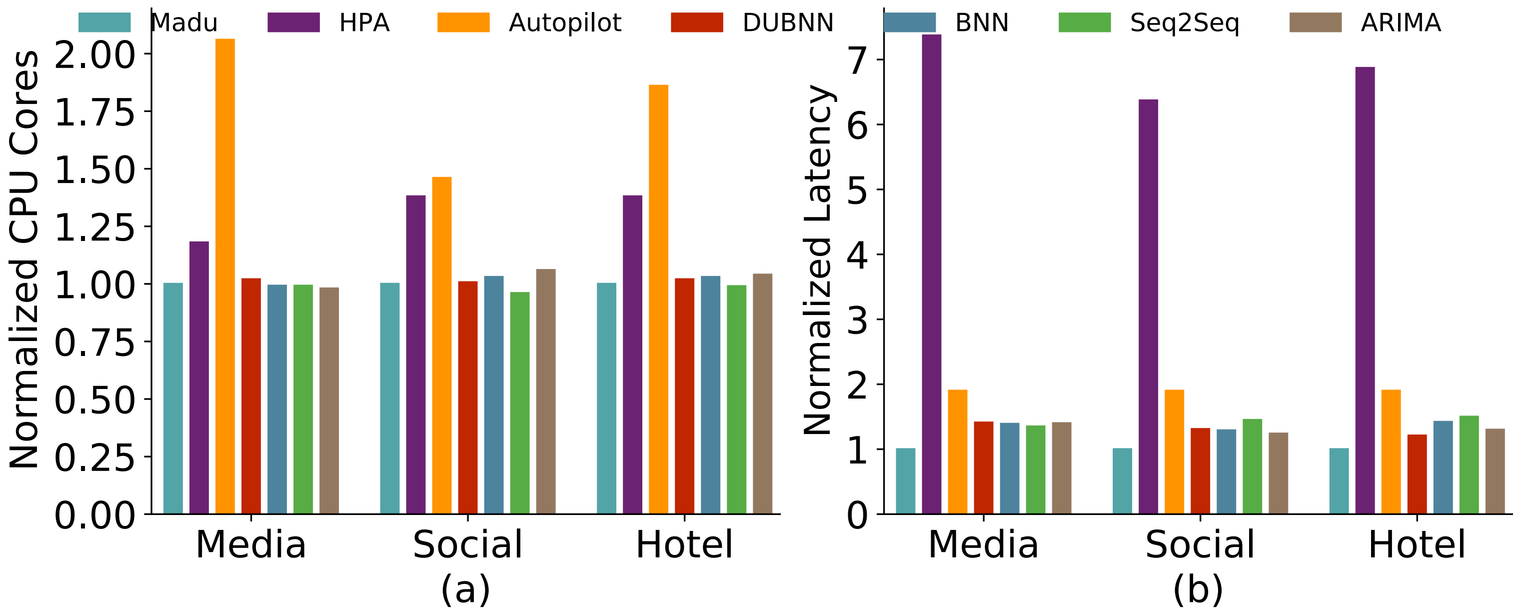
- Prediction accuracy:

Percentile	ARIMA	Seq2Seq	BNN	DUBNN	Madu
[0,50%]	72.1	83.6	73.3	74.4	91.1
[50%,95%]	86.1	87.1	89.4	88.7	93.8
[95%,100%]	88.8	89.7	89.2	90.8	91.5
Avg	79.3	87.1	81.3	81.6	92.3

Madu can outperform other baseline schemes by 13.1%.

Evaluation on All Auto-scaler

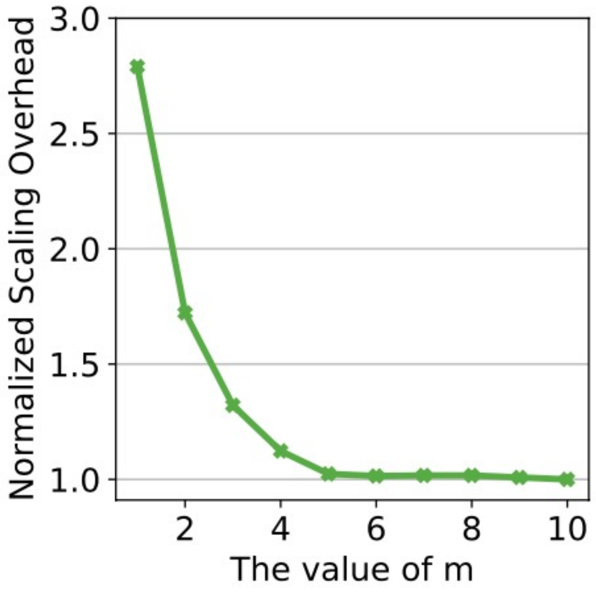
- Comparison between different scalers using different applications

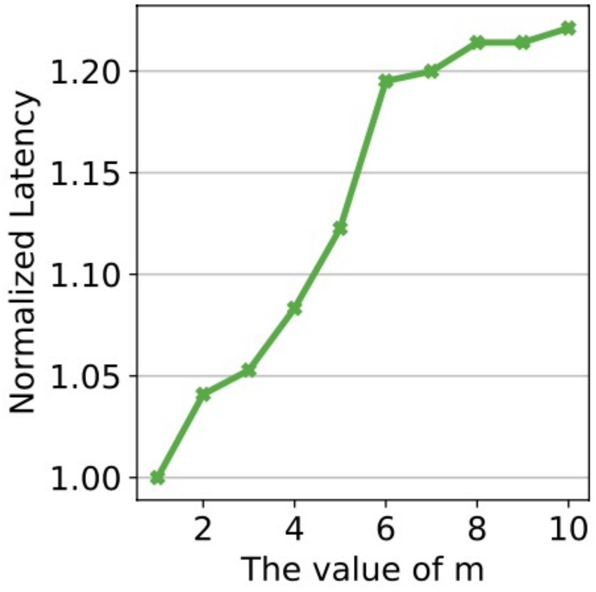


Madu saves up to 40% allocated resource and reduces the end-to-end latency by 36%.

The Length *m* of the Lookahead Period

➢ Trade-off between scalability and performance





When m = 5, the worst end-to-end latency is 10% higher than that under m = 1.

Summary

- The first to predict data-dependent uncertainty for MS workload
- Proactive auto-scaler leverages workload uncertainty prediction.
- Optimize the scaling overhead and MS performance

Q&A

THANKS

Email: st.luo@siat.ac.cn