AOMO: An AI-aided Optimizer for Microservices Orchestration

SIGCOMM 2019 BEIJING

Xue Leng[Ɣ], Tzung-Han Juang[Ψ], Yan Chen[Ψ], Han Liu[Ψ]

[Ɣ] Zhejiang University, [Ψ] Northwestern University

Abstract

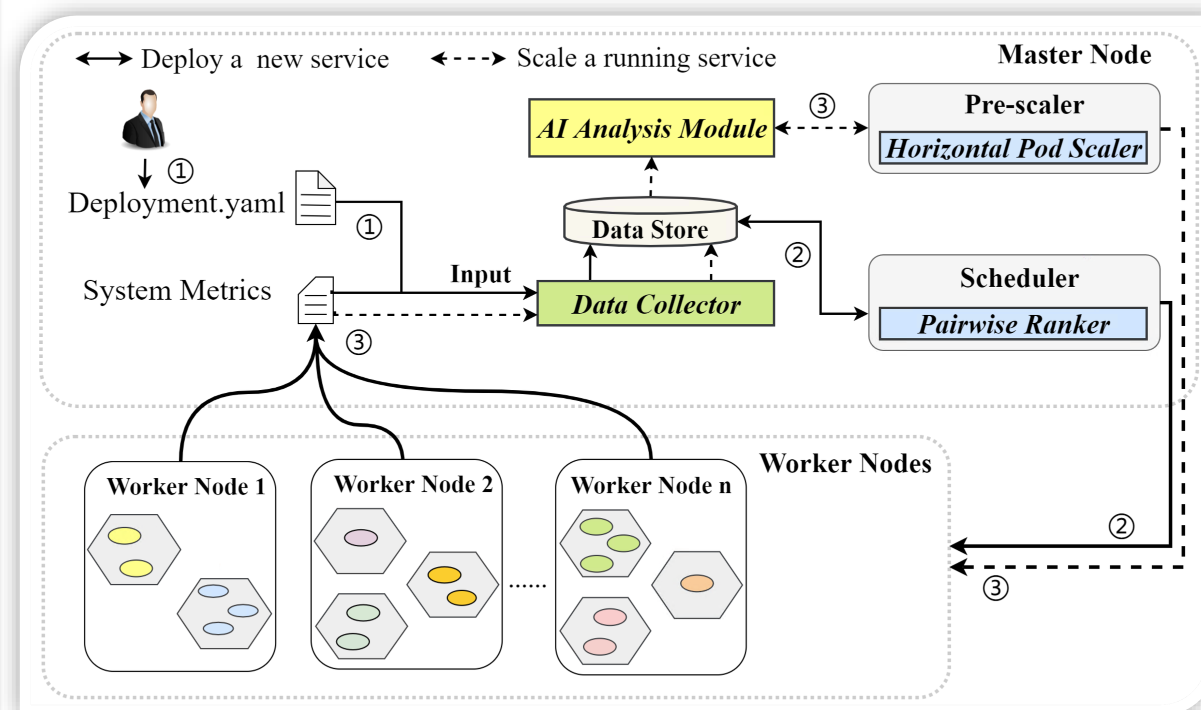
Cost-effective and high-performance microservices orchestration is a challenge for Microservice Management Service Providers (MMSPs). Current microservices scheduling mechanisms cannot obtain the optimal solution for large-scale microservices within a short period of time [1] and scaling mechanisms are either threshold-based or semi-automatic. In this case, the resources of the cluster are not fully utilized which increases unnecessary costs of MMSPs.

To address the downsides mentioned above and reduce the total costs of MMSPs, in this paper, we propose **AOMO**, an AI-aided Optimizer for Microservices Orchestration, which can achieve cost-effective and high-performance microservices orchestration across microservices' lifecycle. To improve the resource utilization of cluster, we propose a ***ranking-based p-batch scheduling mechanism***, which adopts pairwise ranker to obtain the scheduling plan rapidly for large-scale microservices. To improve the scaling agility of microservices, we propose a ***proactive prediction-based scaling mechanism***, which performs scaling operation in advance based on the prediction of resource usage.

Motivation-Challenges

- **Inefficiency of the scheduling mechanism** → propose a mechanism to deploy large-scale microservices in a short time while achieving high resource utilization
- **Lack of scaling agility** → design a microservices scaling mechanism to adjust the number of microservice instances to adapt to workload fluctuations without triggering the critical state

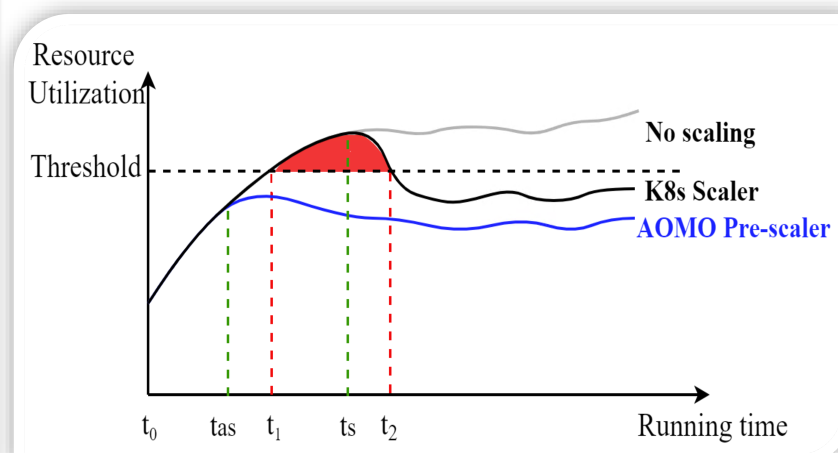
Framework



- **Data Collector** collects CPU and Memory usage of pods and nodes [2].
- **Scheduler** selects a node for pods to run on.
- **Pre-scaler** scales up/down pods in advance based on the resource usage predictions.

Pre-scaler Design

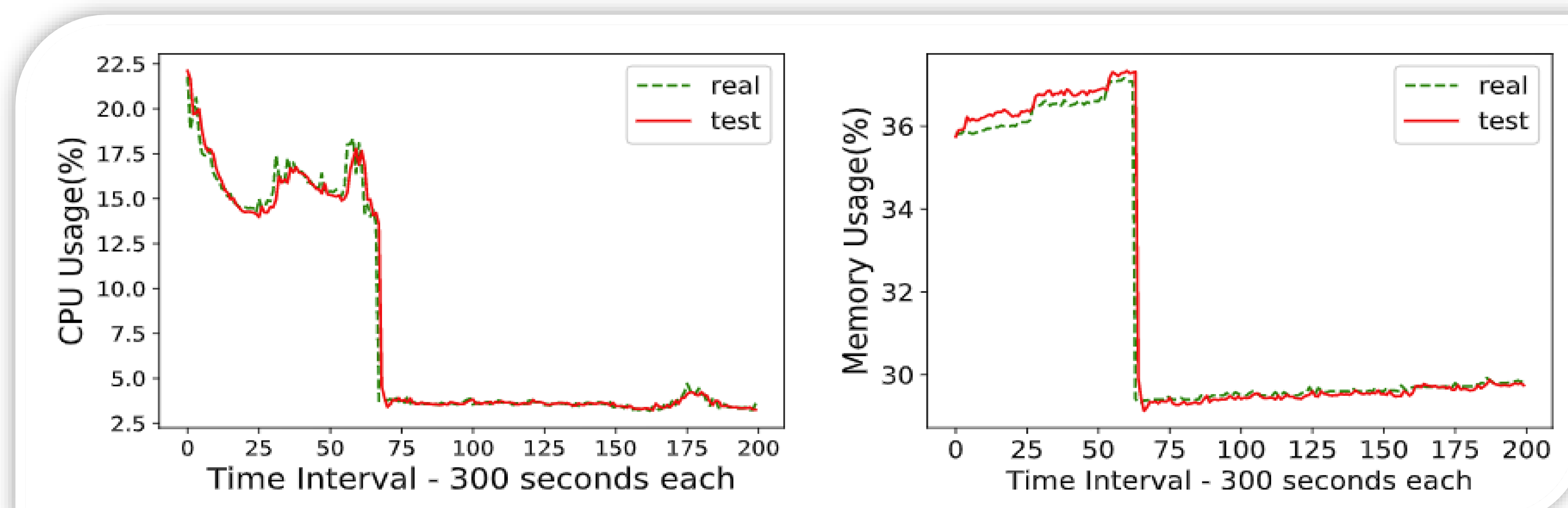
We propose a *proactive prediction-based scaling mechanism*, which can automatically scale microservices in advance based on the resource usage prediction.



The Bidirectional Long Short-Term Memory (BI-LSTM) model is adopted to predict the resource (e.g. CPU and Memory) usage of microservices in the next 15 minutes.

This scaling mechanism can effectively avoid resource usage reaching the threshold and mitigate resource contention and service disruption.

We adopt public Alibaba Cluster Data [3] to assess the accuracy of resource usage prediction. The average RMSE is 2.85.

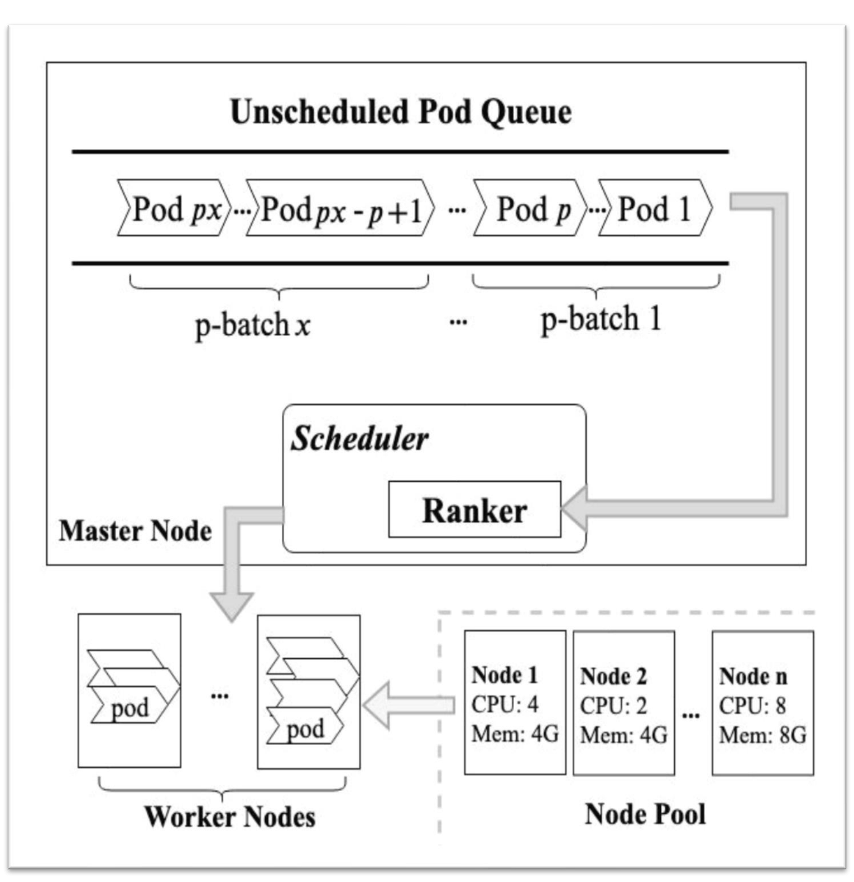


Scheduler Design

We propose a *ranking-based p-batch scheduling mechanism* to determine the deployment plan of microservice instances and achieve rapid and cost-effective deployment.

PERFORMANCE COMPARISON BETWEEN THE *K8s Scheduler* AND THE *AMIO Scheduler*.

# of Pods	Total Resources Requested by Pods		# of Nodes		CPU Utilization		Memory Utilization	
	CPU (core)	Memory (GB)	AOMO	K8s	AOMO	K8s	AOMO	K8s
50	4.75	3.77	2	10	59.38%	11.05%	31.42%	8.77%
	9.50	7.54	4	10	67.86%	22.09%	37.70%	17.53%
100	9.97	8.02	5	10	49.83%	23.17%	40.10%	18.65%
	19.93	16.04	8	10	64.29%	46.35%	45.83%	37.30%



Ranking-based

Pairwise ranker is adopted and trained offline with millions of data generated by the ranker simulator. With the **scheduling principle** of preferring active nodes than inactive nodes and making the ratio of consumed CPU to Memory close to the ratio of allocatable CPU to Memory of the node, the number of nodes running in the cluster is significantly reduced.

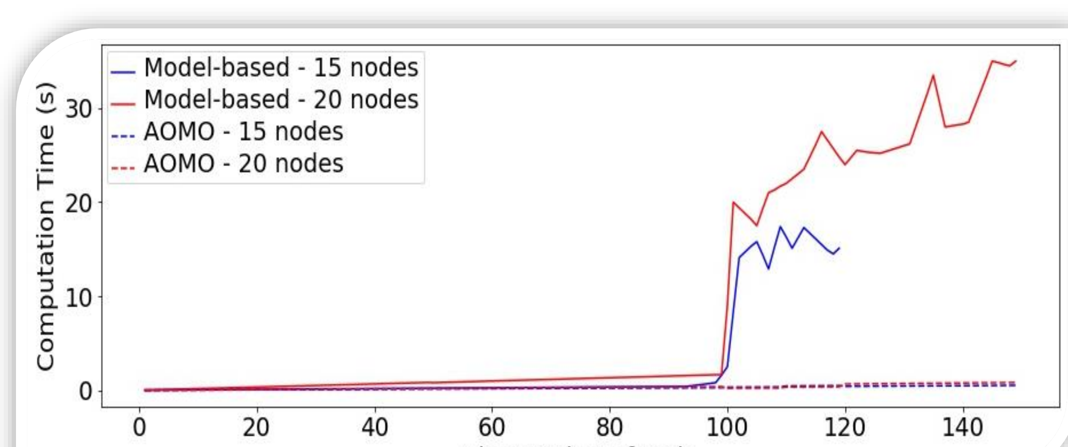
$$P(N_j \geq N_k | R_{pod}, C_{N_j}, C_{N_k}) = \frac{1}{1 + e^{-f_\omega(R_{pod}, C_{N_j}, C_{N_k})}}$$

p-batch

Scheduler considers *p* pods as a batch in each scheduling round to optimize the deployment solution.

Results

The average computation time of the ranker is 3.67ms.



References

[1] Adalberto R Sampaio, et al. 2019. Improving microservice-based applications with runtime placement adaptation. JISA 10, 1 (2019), 4.

[2] Kubernetes, production-grade container orchestration, 2019. http://bit.ly/k8sio

[3] Alibaba Cluster Data v2017. https://github.com/alibaba/clusterdata