Robust performance modeling of containerized microservices with Probabilistic Machine learning

Peng Kang† , Department of Compute Science, Palden Lama‡, PhD
‡The University of Texas at San Antonio, San Antonio, Texas 78249

Background & Motivation

Large-scale web services are increasingly being built with many small modular components (microservices), which can be deployed, updated and scaled seamlessly. These microservices are packaged to run in a lightweight isolated execution environment (containers) and deployed on compute resources rented from cloud providers. However, the complex interactions and the contention of shared hardware resources in cloud data centers pose significant challenges in managing web service performance.

In this paper, we develop probabilistic machine learning-based performance models, which can quickly adapt to changing system dynamics and directly provide confidence bounds in the predictions even when the data is noisy and sparse, to enable cloud platforms to provide robust performance guarantee for large-scale web serivces. We also leverage multi-layered data collected from container-level resource usage metrics and virtual machine-level hardware performance counter metrics for enhancing performance modeling accurate.

Framework & Platform

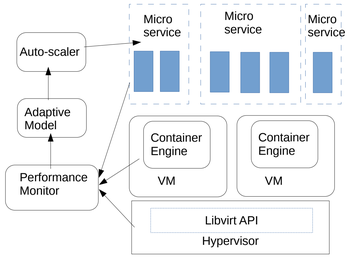

Fig. 1: framework for automatically adaptive performance modeling with multi-layer data.

Experimental Testbed. We implemented and evaluated our modeling approach on NSF Cloud's Chameleon[1] testbed using KVM for virtualization, Docker Engine for containerization and Kubernetes[2] for container orchestration.

Workloads. We used an open-source microservices benchmark, Robot Shop[7], for performance characterization and the Locust tool[3] to generate user traffic for the Robot Shop benchmark.

Microservices

Microservice architecture aims to overcome various limitations of traditional monolithic architecture for software development[5, 4]. Microservice architecture splits the application into many smaller self-contained components, called microservices, that serve specific functions and communicate with each other via lightweight language-agnostic APIs as shown in Figure 2.

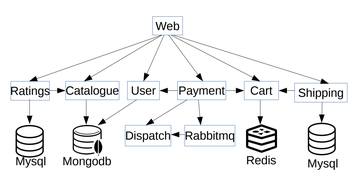

Fig. 2: Microservices.

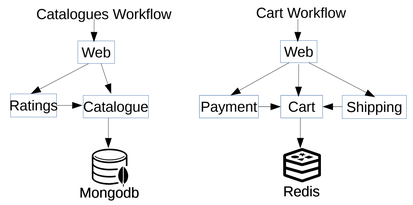

Fig. 3: Workflow DAGs.

Adaptive Modeling

Gaussian Process(GP) Regression, a non-parametric kernel-based probabilistic modeling technique, utilizes the concept of Guassian Process and Bayesian inference to find a distribution over all the possible functions that are consistent with the observed data[6]. Hence, this probabilistic modeling approach can provide confidence bounds on its predictions, which is critical for making robust resource management decision in an uncertain cloud environment.

Problem Formulation. We aim to find the highest resource utilization values of the relevant microservices, at which the given SLO target will not be violated. Therefore, we design a constrained optimization problem as follows:

$$max \sum_{i \epsilon S} \sum_{r \epsilon R} x_{ir}$$
$$\text{s.t. } \forall : \bar{m}_j(x) + \kappa \bar{\sigma}_j(x) \leq SLO_j^{target}$$
$$\forall i, r : x_{ir} \leq 0$$
$$\mathrm{x} = ((\mathrm{x}_{ir})_{r \epsilon R})_{i \epsilon S_j}$$

Symbol	Description
S	Set of microservices relevant to the target workflows
R	Set of container resource types(CPU, I/O, network)
SLO_j^{target}	tail latency target of workflow j
x_{ir}	Average utilization of resource r in microsevices i
$\bar{m}_j(x)$	Posterior mean function of workflow j
$\bar{\sigma}_j(x)$	Posterior standrad deviation of workflow j
S_j	Set of microservices relevant to workflow j

Evaluation & Result

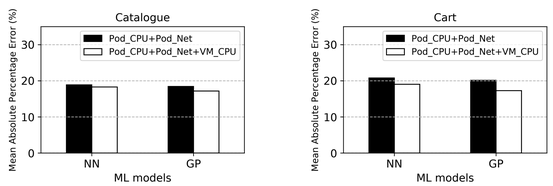

Fig. 4: Comparison of Prediction accuracy between NN and GP.

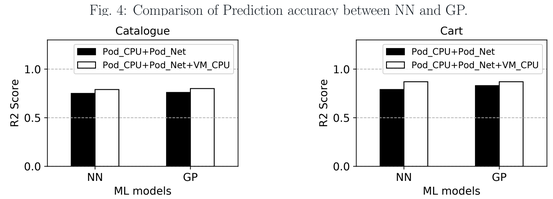

Fig. 5: Comparison of R^2 Score Between NN and GP.

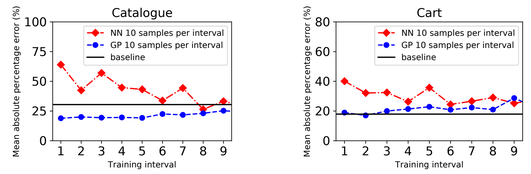

Fig. 6: Evaluating adaptiveness of performance models.

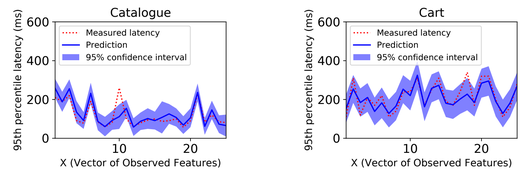

Fig. 7: Confidence Interval for latency prediction of two different workflows.(Computed by GP)

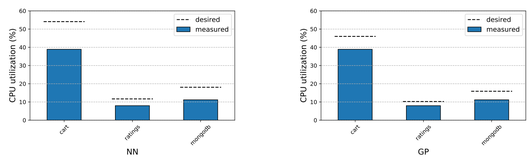

Fig. 8: Current vs desired average CPU utilization of various microsevices.

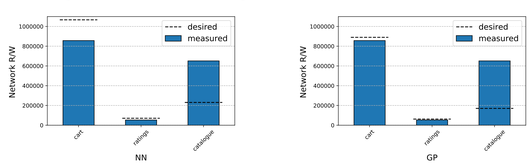

Fig. 9: Current vs desired average Network Read/Write of various microsevices.

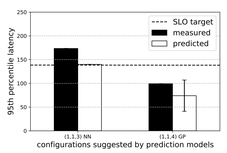

Fig. 10: Observed vs Predicted 95th percentile latency, for resource configurations suggested by NN and GP.

Results. The scalar configuration implements the decision as suggested by the models. However, actual observed latency suggest the scaling decision for NN failed to meet the SLO target, where as GP meets the SLO target.

Acknowledgement

This research was supported by NSF research grant CNS-1911012. Results presented in this paper were obtained using the Chameleon testbed supported by the NSF.

Reference

[1] *Chameleon: A configurable experiemental environment for large-scale cloud research.* `https://www.chameleoncloud.org/`.