PEMA: Practical Efficient Microservice Autoscaling

Md Rajib Hossen and Mohammad A. Islam
The University of Texas at Arlington, Texas, USA

THE UNIVERSITY OF TEXAS AT ARLINGTON

RiDL Rigorous Design Lab

Microservice Architectures

- Adoption of microservices is growing steadily
- Microservices consist of independent and loosely-coupled units which are easy to deploy and update

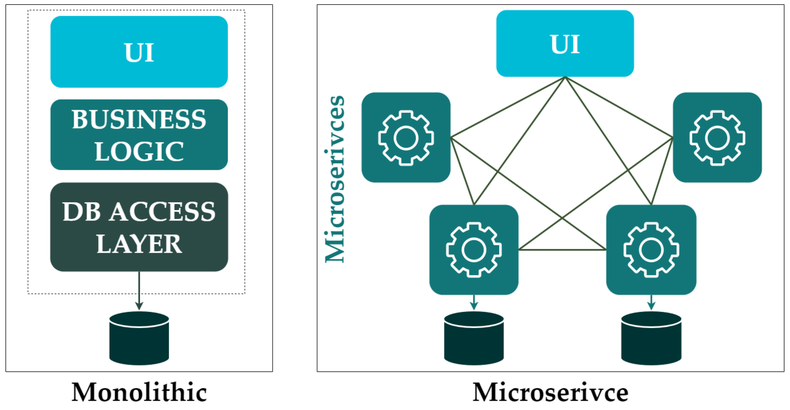

Figure 1: Monolithic vs Microservice architecture

Challenges

- Microservices introduce new challenges in resource management due to their large configuration spaces and complex interactions between services
- Recent approaches for resource management include rule based or Machine Learning (ML) based models
- Rule based fails to handle microservices complexities
- ML requires offline data collection for training and intentional SLO violations for boundary data

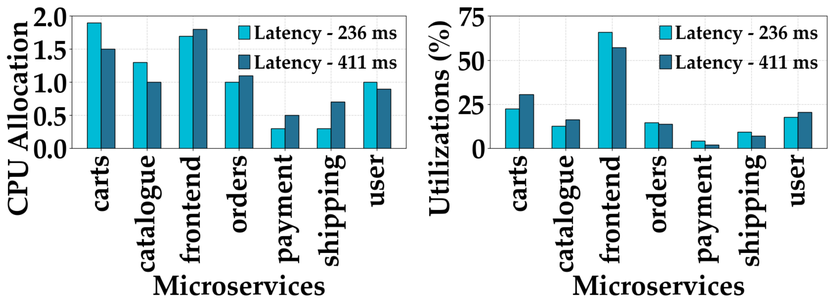

Figure 2: CPU allocation dictates latency.

- Sub-optimal CPU allocations lead to SLO violations
- Threshold of metrics such as utilizations does not help in finding good CPU allocation
- Optimum CPU allocation is critical for performances, yet not easy to find

Our Contributions

- The main goal is to find efficient resource allocation quickly and without violating SLO
- PEMA does not require offline data to train
- PEMA navigates the configuration space from ample resources to optimum resource allocation
- It utilizes the observation that "monotonic resource reduction" leads to monotonic increase in latency

Design of PEMA

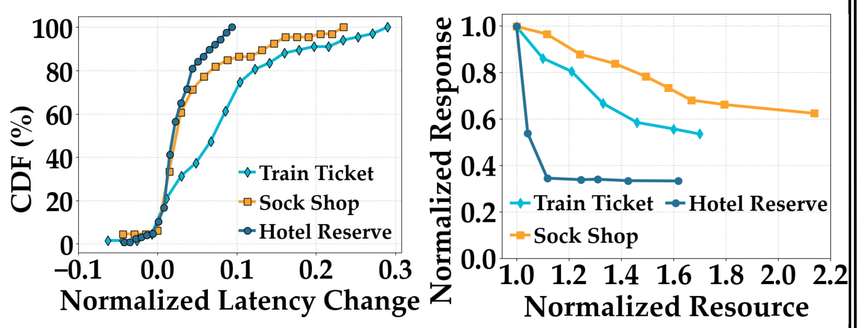

Figure 3: Monotonic resource reductions increases latency in 95% of the time. Response also reach to SLO.

- Optimization objective is to minimize total resources while maintaining SLO in each time step
- Uses the response time as a guide to reaching the optimal resource allocations (Fig 3)
- Uses microservices-wise metrics to avoid bottlenecks and identify candidates for further reductions
- Applies resource reduction iteratively and uses feedback to re-evaluate reduction parameters.
- Tunable parameters to control resource reduction
- Uses exploration to avoid local minima (Fig 4)

Performance Evaluation

Settings

- Local Kubernetes cluster consisting of 5 nodes
- Three representative microservices applications – Sock Shop, Train Ticket, and Hotel Reservation
- Compared with OPTM – offline exhaustive search to find optimum, and rule based (RULE) algorithm.

Results

- PEMA converges to optimal resource allocation for all microservices (Figure 4)
- PEMA performance is close to OPTM (within 5%) and outperformed RULE by 33% (Figure 5)

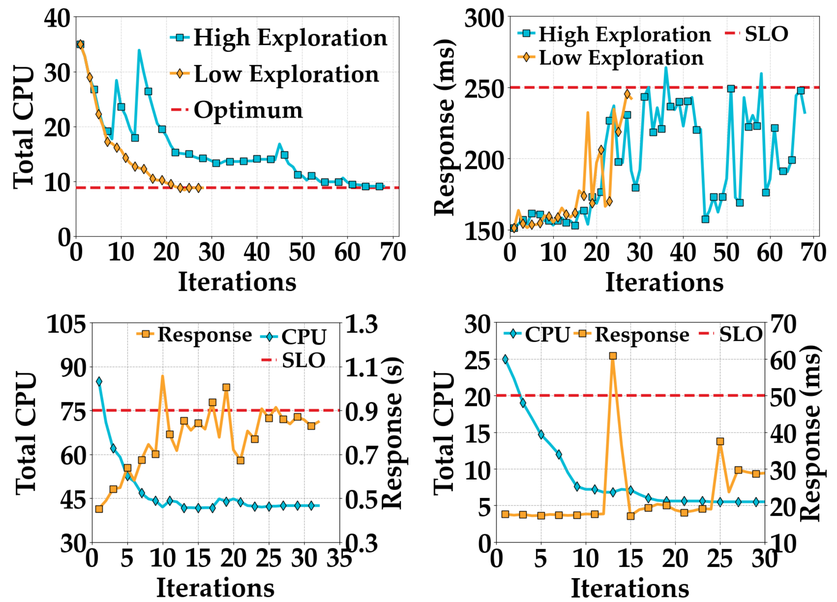

Figure 4: Convergence for Sock Shop (top two), Train Ticket (lower left), and Hotel Reservation (lower right).

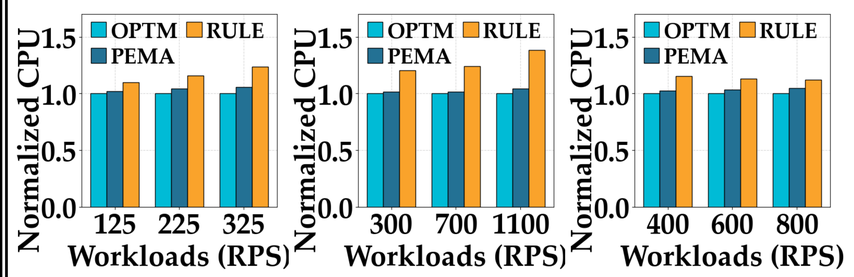

Figure 5: Performance comparison on Train Ticket (left), Sock Shop (middle), and Hotel Reservation (right)

Future Works

- Early detection of SLO violation
- Use knowledge from execution history in future resource allocation
- Enable agile vertical & horizontal scaling

Full version has been accepted in HPDC'22.

Md Rajib Hossen, Mohammad A. Islam, and Kishwar Ahmed. 2022. Practical Efficient Microservice Autoscaling with QoS Assurance. HPDC '22, June 27-July 1, 2022, Minneapolis, MN, USA. https://doi.org/10.1145/3502181.3531460