ELASTICITY CONTROL FOR LATENCY-INTOLERANT MOBILE EDGE APPLICATIONS

Chanh Nguyen, Cristian Klein, Erik Elmroth

Dept. Computing Science

Umeå University, Sweden

UMEÅ UNIVERSITET

ELASTICITY IN CLOUD

- What is Elasticity?
- How does Cloud Computing Control Elasticity?
  - Re-active.
  - Pro-active.
  - Hybrid.

ELASTICITY CONTROL IN MOBILE EDGE CLOUD THE NECESSITY

- Most MECs applications are latency-sensitive applications.
- Limited resources with higher resource costs at the edge data centers (EDCs).
- The stochastic nature of user mobility causes resource demand fluctuated.
- Auctuation delays – allocated resources is not ready to use immediately.

ELASTICITY CONTROL IN MOBILE EDGE CLOUD GOAL

- MECs operator’s perspective:
  - Average resource utilization at EDCs.
  - System stability.
- End-user’s perspective:
  - Average rejected rate.

PRO-ACTIVE ELASTIC CONTROL FRAMEWORK

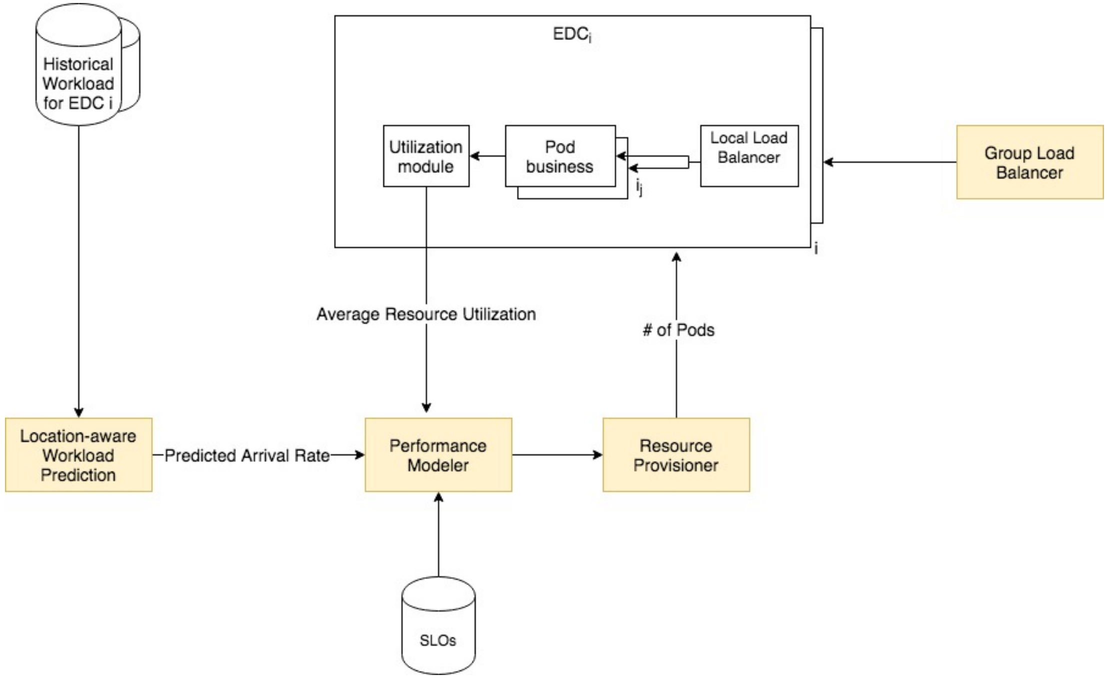


Figure 1: Components of the proposed controller.

PRO-ACTIVE ELASTIC CONTROL FRAMEWORK

- **Location-aware Workload Predictor**
  - Multi-variate LSTM networks.
- **Performance Modeler**
  - Resources are abstracted at Pod modelled as a M/M/1/k FIFO queue.
- **Resource Provisioner**
  - cross-evaluating the resource requirements of EDCs in a group and determine a final number of desired resources for each EDC.
- **Group Load-balancer**
  - Weight round-robin load balancing approach.

EXPERIMENT SETTING

- Emulated MEC:
  - MEC with EDCs distributed over a metropolitan area.
- Application:
  - Extremely latency-intolerant AR application.
- Workload:
  - Real taxi mobility traces.

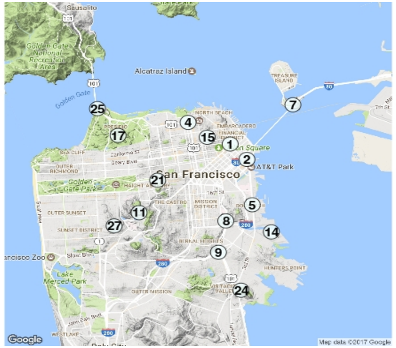


Figure 2: Distribution of EDCs in San Francisco.

EXPERIMENT SETTING

- Predefined Service Level Objectives:
  - Average Utilization =80%.
  - Rejection rate = 1%.
- Controller settings:
  - Pro-active Auto Scaler.
  - Pro-active Auto Scaler + Group Load Balancer.
  - Re-active Auto Scaler: Kubernetes HPA*.

TABLE I: Group settings.

GroupID	EDCs
#1	#1, #2, #3, #5, #10
#2	#8, #12, #15
#3	#11, #14
#4	#4, #6, #7, #9, #13

*https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

EXPERIMENT SETTING

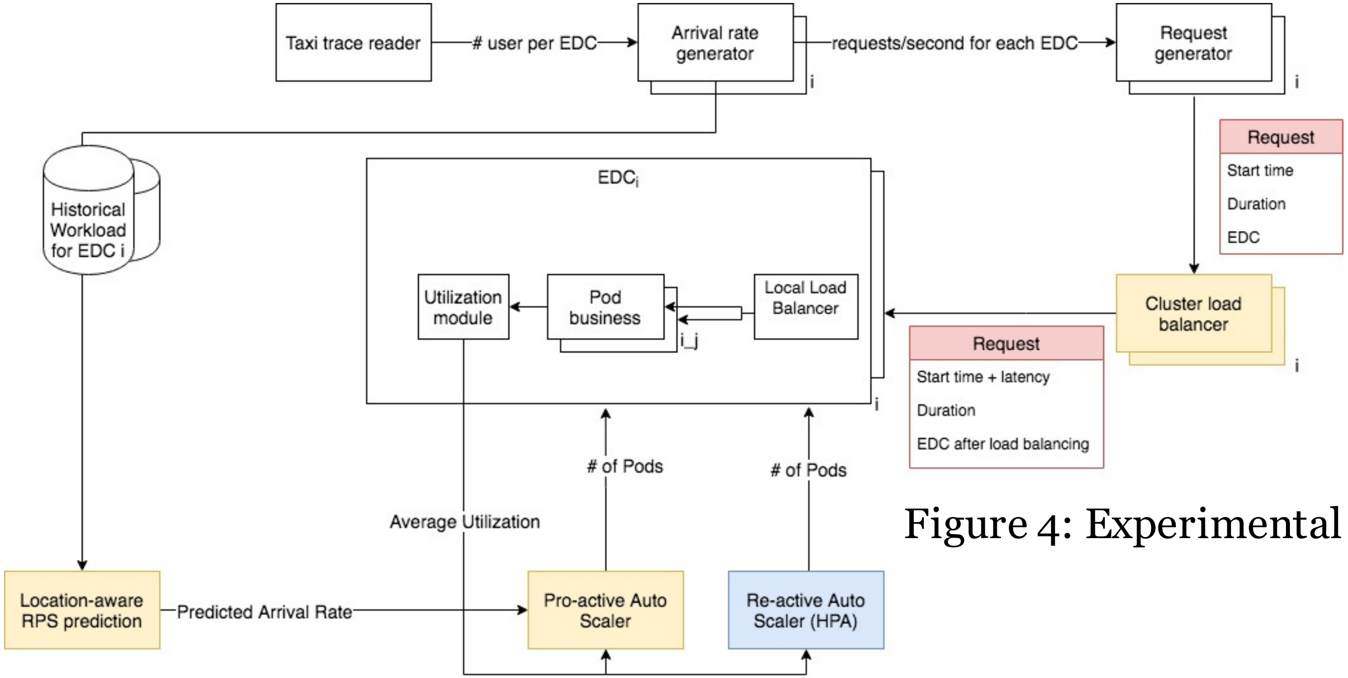


Figure 4: Experimental simulation.

EVALUATION - PERFORMANCE METRIC

- System and user-oriented metrics: recommend by SPEC*
  - Under-provisioning accuracy,
  - Over-provisioning accuracy,
  - Under-provisioning timeshare,
  - Over-provisioning timeshare,
  - Instability.

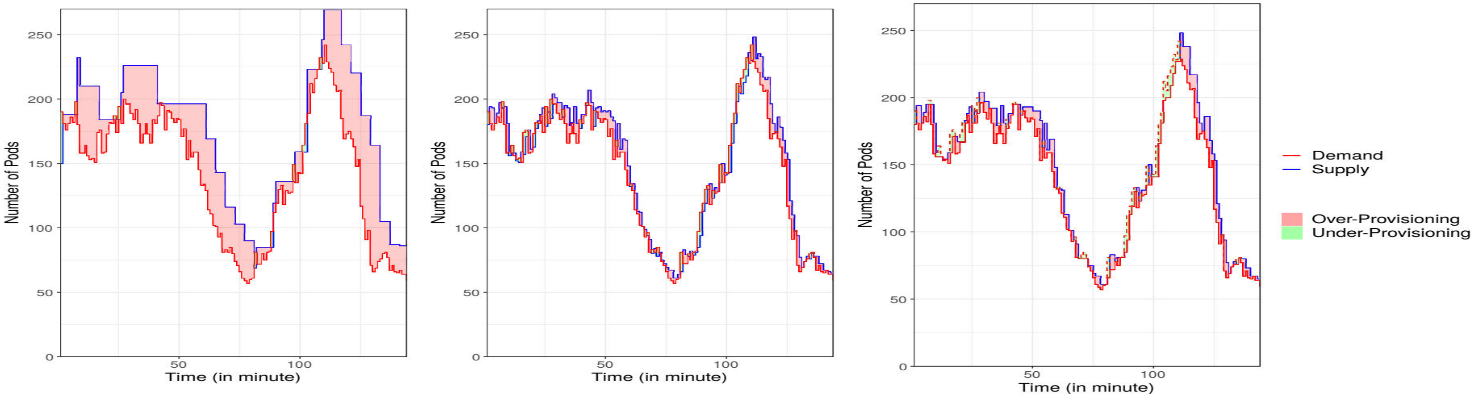
*Nikolas Herbst et al., Ready for rain? A view from SPEC research on the future of cloud metrics

How does the proposed pro-acitve controller perform when compared to the re-active controller?

Metric	Pro-active AS + LB	Pro-active AS	Re-active AS
θ_U	13.6	41.2	**5.4**
θ_O	**14.2**	39.5	305.6
τ_U	**4%**	43%	5.3%
τ_O	**2.5%**	46.7%	94.1%
υ	**2.44%**	2.8%	3.9%
Avg. resource utilization	**85.9%**	80.5%	68.4%
Rejection rate	**0.02%**	0.26%	0.04%
total Pods	**3154**	4405	5337
Avg. Pod lifetime (minute)	**73.3**	35.2	29.6

Table II: The performance of the three controllers based on the elasticity metrics.

How does the proposed pro-acitve controller perform when compared to the re-active controller?



a) Re-active b) Pro-active AS c) Pro-active AS + LB

Figure 5: The scaling behavior of three controllers on EDC#1.

How does the proposed pro-acitve controller perform when compared to the re-active controller?

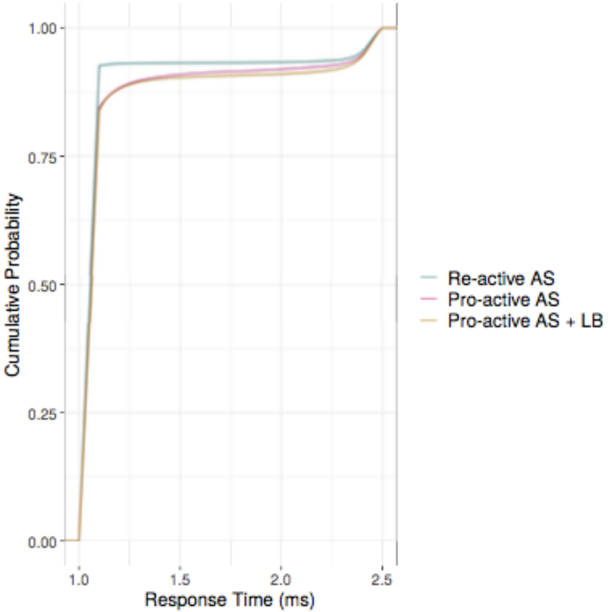
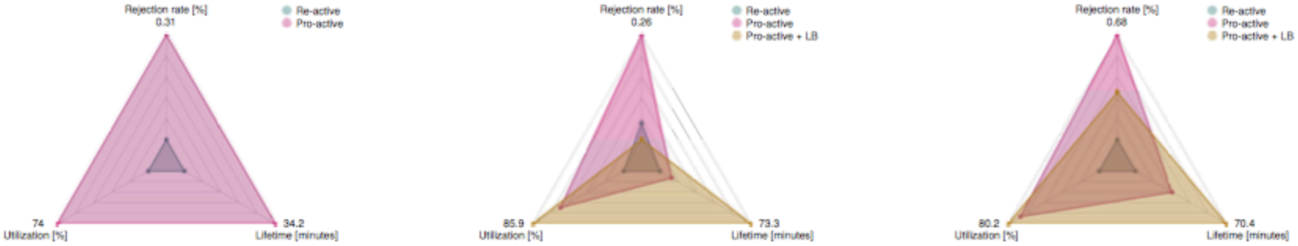


Figure 6: Cumulative density of response times of the application in three elastic controller settings.

To what degree does location-awareness improve scaling behavior?

- Conduct another experiment which a group is set with different size k
  - k = 1
  - k = 15



(a) Groups consisting of 1 EDC only ($k = 1$).

(b) Groups with neighboring EDCs as specified in table I.

(c) Single group consisting of all 15 EDCs ($k = 15$).

Figure 7: Performance of the three studied controller configurations based on the three major elasticity metrics when the number of neighboring EDCs is varied.

What is the decision time of the elastic controller?

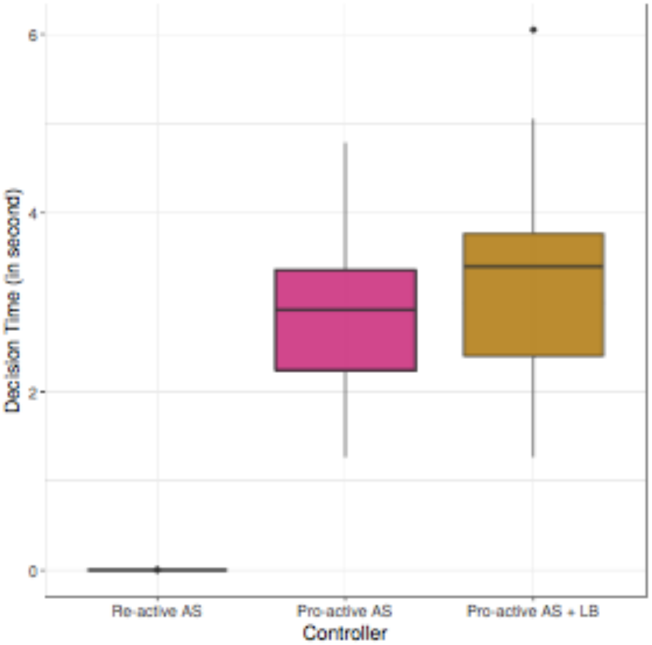


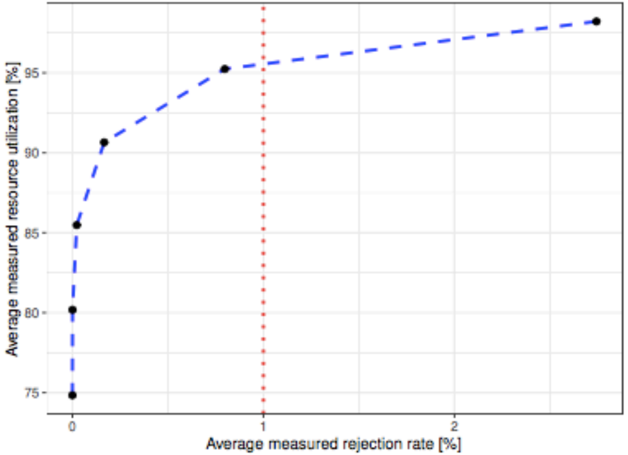
Figure 8: Average Decision Time of the three controllers.

What is the impact of the two predefined threshold on the controller's scaling behavior?

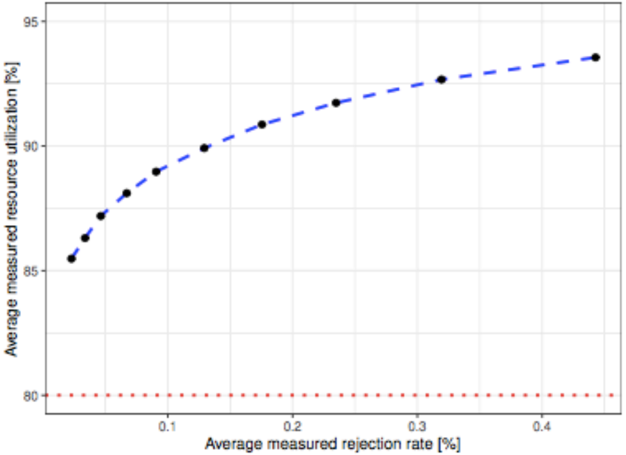
Targeted rejection rate[%]	Targeted resource utilization[%]	Measured resource utilization[%]	Measured rejection rate[%]	Total Pods
1	70	74.8	0	3812
	75	80.2	7e-4	3484
	80	85.9	0.02	3154
	85	90.6	0.16	2890
	90	95.2	0.8	2653
	95	**98.2**	**2.7**	**1995**
10	80	93.5	0.44	2753
...		...	...	...
3		87.2	0.05	3065
2		86.3	0.03	3113
1		85.9	0.02	3154

Table III: The scaling behavior of the proposed controller with different predefined threshold settings.

What is the impact of the two predefined threshold on the controller’s scaling behavior?



(a) The targeted resource utilization is changed, while the targeted rejection rate is held constant at 1%.



(b) The targeted rejection rate is changed, while the targeted resource utilization is held constant at 80%.

Figure 9: The controller’s scaling behavior when varying the threshold settings.

CONCLUSION

- The correlation of workload variation in physically neighboring EDCs help improve the resource estimation.
- The Group Load-balancer further helps minimize the request rejection rate.
- The proposed controller achieves a significant better scaling behavior as compared against the state-of-the-art re-active controller.

THANK YOU

Contact for further discussion:

Name: Chanh Nguyen

Email: chanh@cs.umu.se

UMEÅ UNIVERSITET