UNIVERSITY OF AMSTERDAM

Scaling of containerized network functions

Mohanad Elamin
University of Amsterdam
melamin@os3.nl

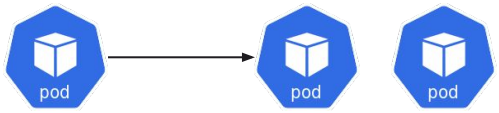
Pim Paardekooper
University of Amsterdam
ppaardekooper@os3.nl

Supervisor:

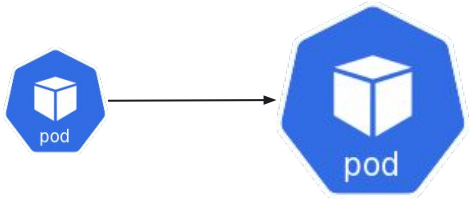
Jamila Alsayed Kassem
UvA

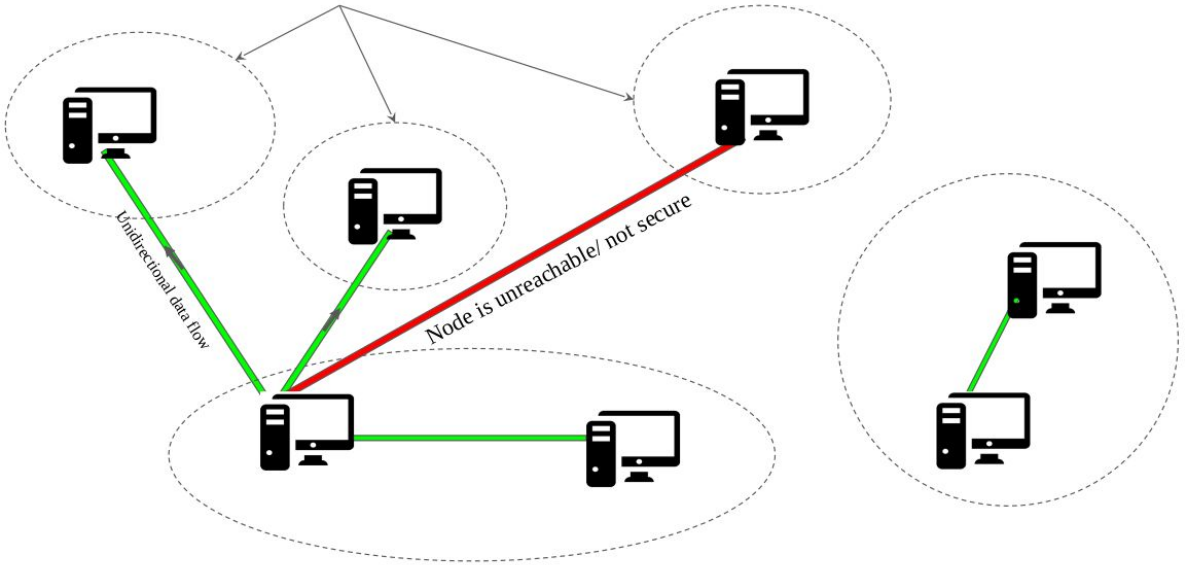
Introduction

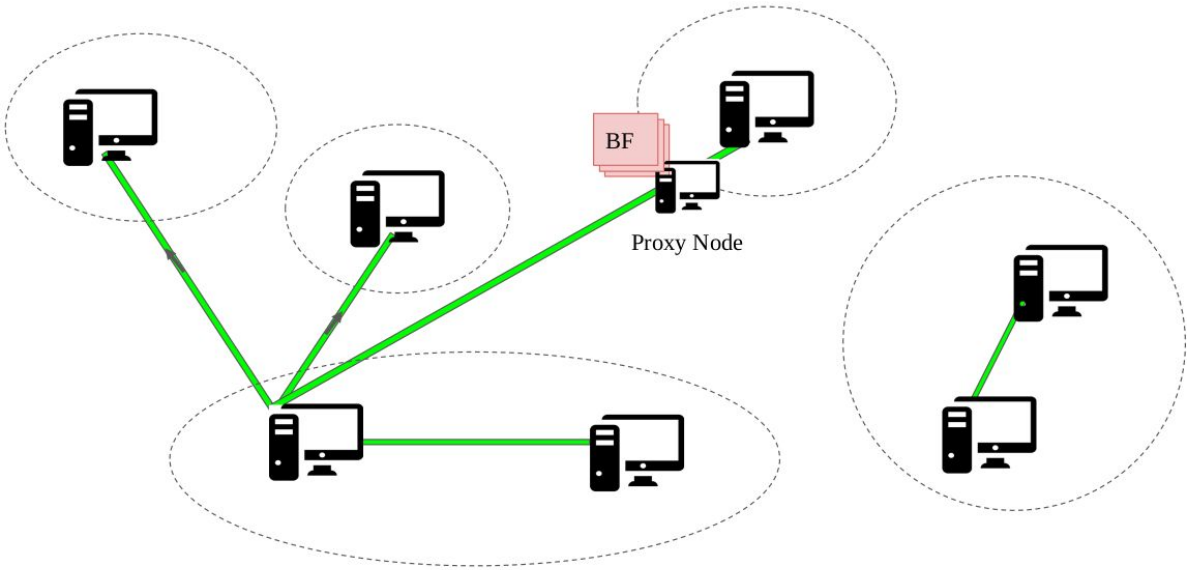
- EPI framework
  - Secure connection
  - Setup bridging function
- Scaling bridging functions
  - Horizontal



  - Vertical







Research Question

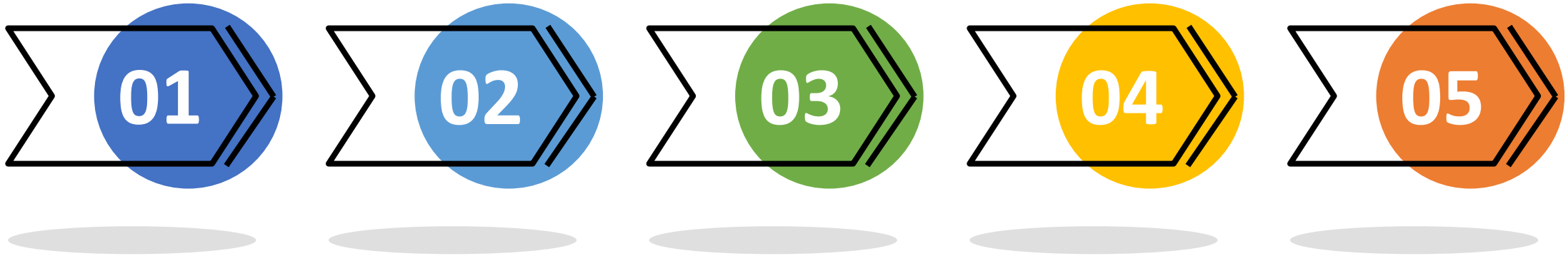
What is the impact of varying k8s autoscaling thresholds for bridging functions on end-user application traffic?

- What metrics should trigger the scaling?
- What is the impact of the bridging functions horizontal scaling on application traffic?
- What is the impact of reconfiguration on in-transit traffic?

Related Work

- Duc-Hung et al. explained the autoscaling mechanisms available in existing container orchestrators in the IT domain, with special focus on the mobile core elements.
- Salman Taherizadeh and Marko Grobelnik, proposes three key factors which should be considered for auto-scaling methods in Kubernetes
- Steven Van Rossem et al points out that predicting the performance of a VNF chain based on the performance of the discrete network functions is not accurate.
- Adel Nadjaran Toosi et al research produces a tool called ElasticSFC, which shows that their auto-scaling techniques based on the VNF chain can reduce cost.

Methodology



PoC

Build Proof of Concept on top of the EPI framework

Identification

Identify scaling possibilities

Experiment

Define and do experiments based on the scaling possibilities

Plotting

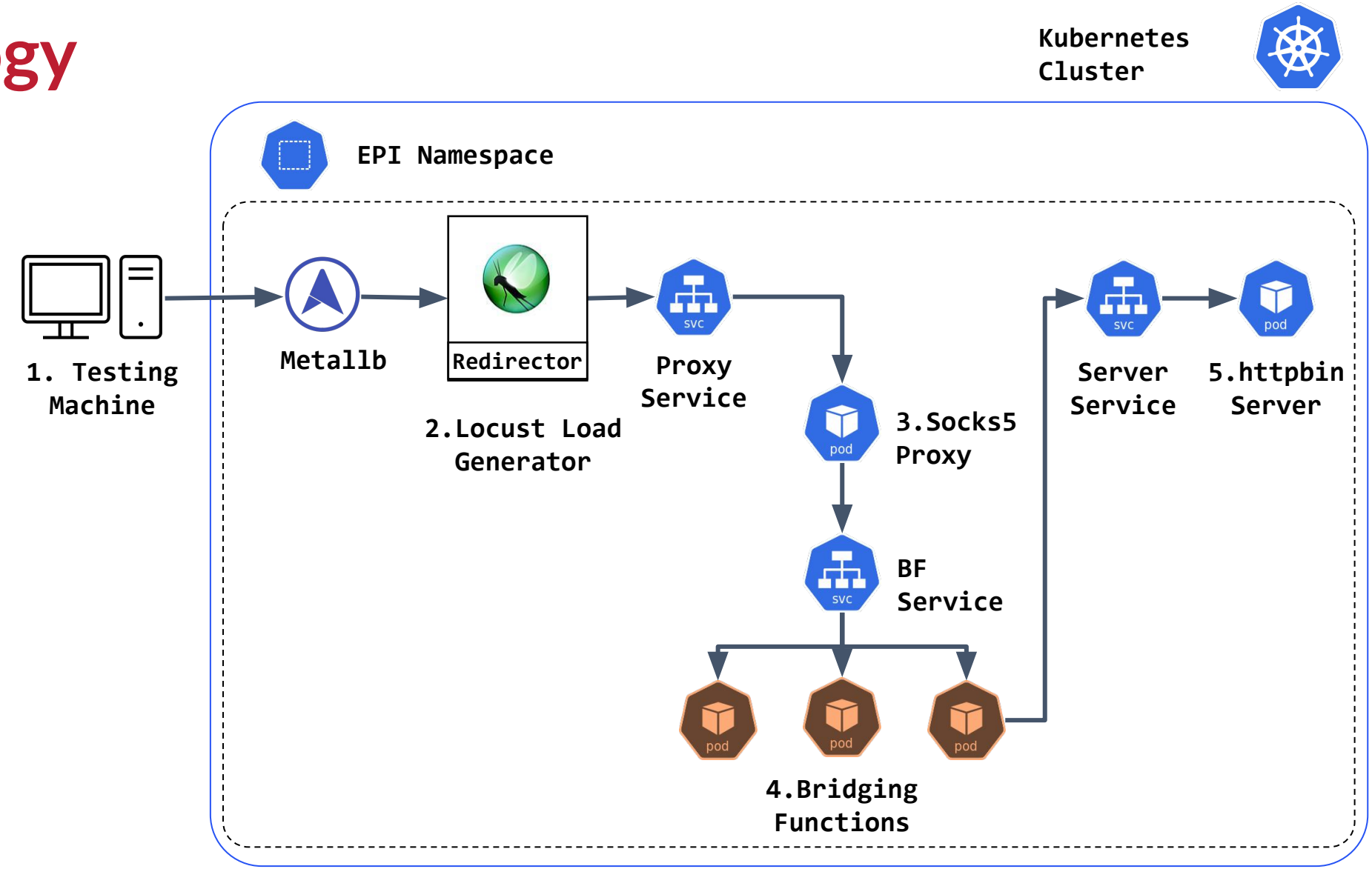
Create graphs that will answer our research question

Conclusion

Answer the research question based on the plots

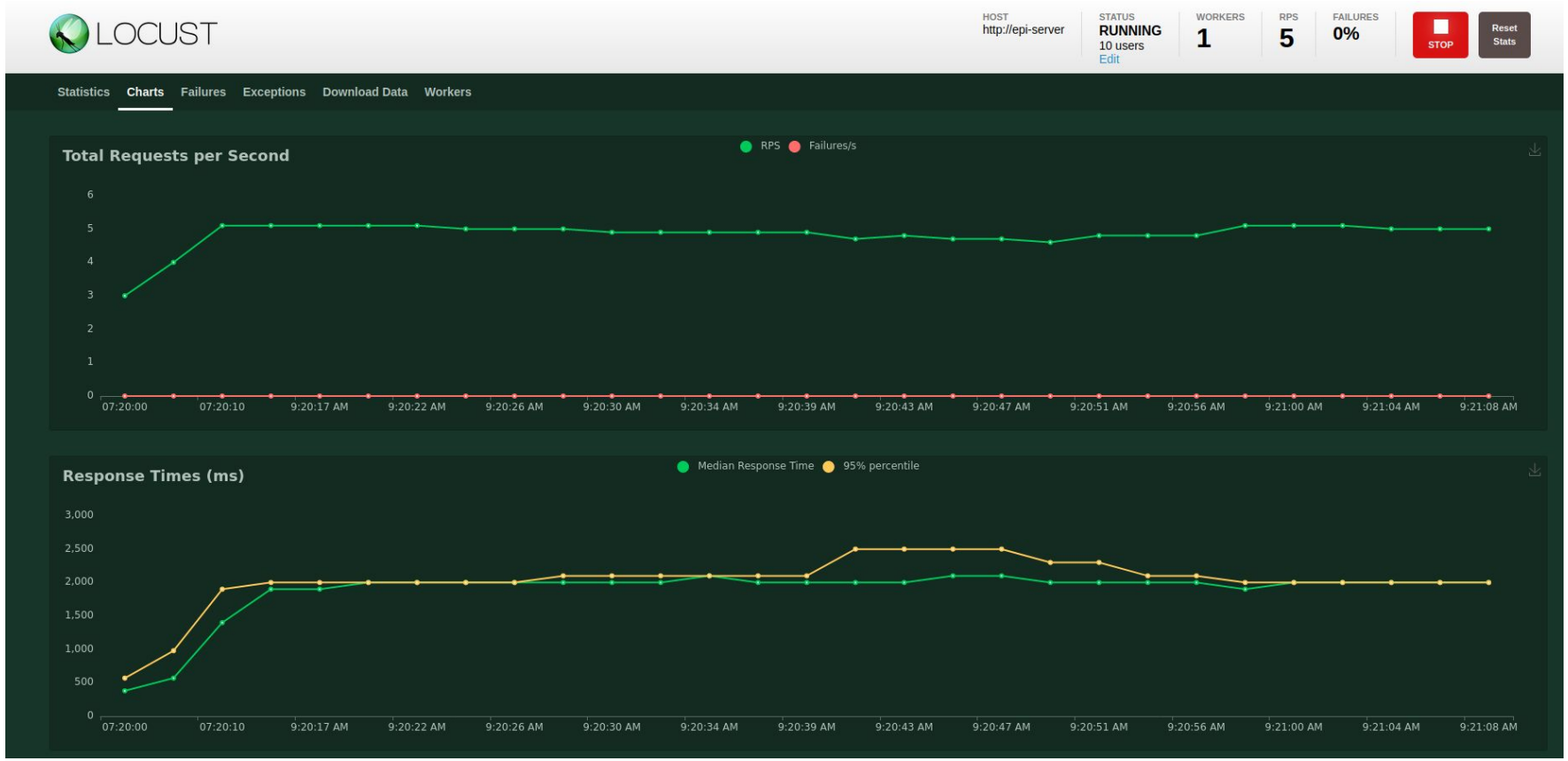
Setup - Topology

1. Testing machine
2. Locust load balancer
3. Socks5 proxy
4. Bridging function
5. Httpbin server



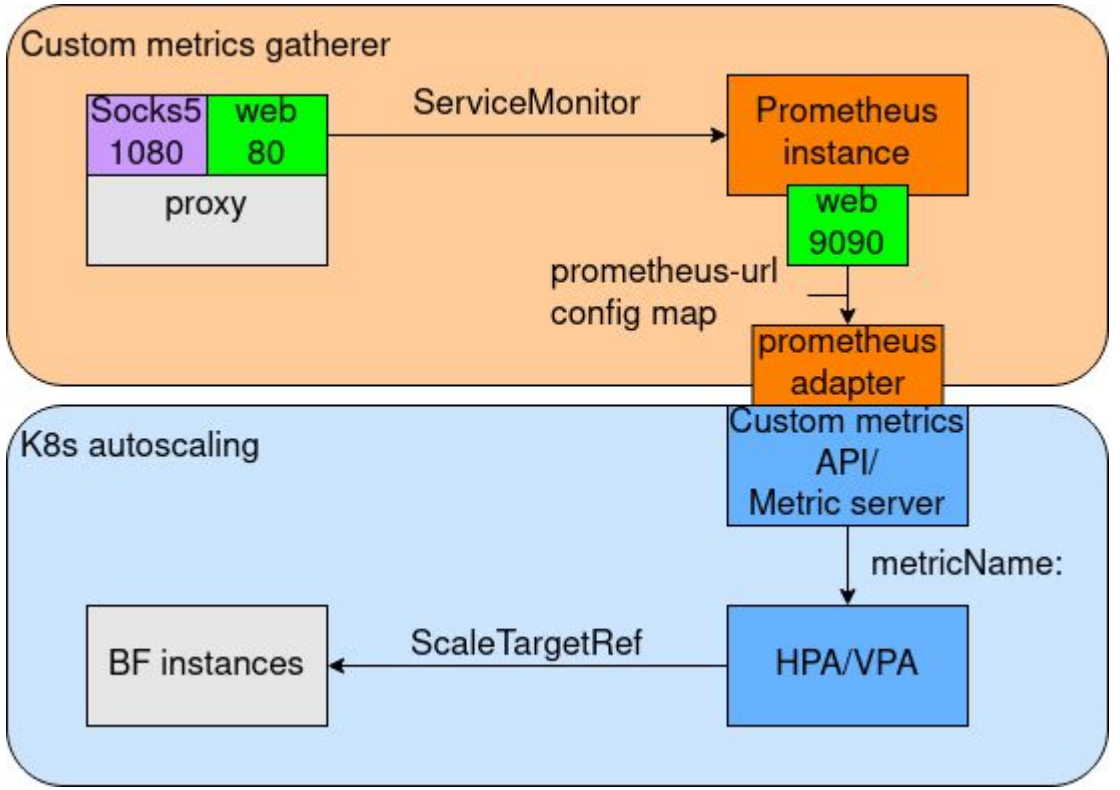
Setup - Locust Load Generator

- Load tests:
  - Number of users
  - Spawn rate
  - Tasks
- Locust metrics:
  - RPS
  - Users
  - Response time



Kubernetes autoscalers implementations

- HPA
  - desiredReplicas = ceil[currentReplicas * (currentMetricValue/desiredMetricValue)]
- VPA
  - Desired value calculation
    - Recommender
    - Updater
    - Admission plugin
- Metric server
- Custom metrics
  - Prometheus



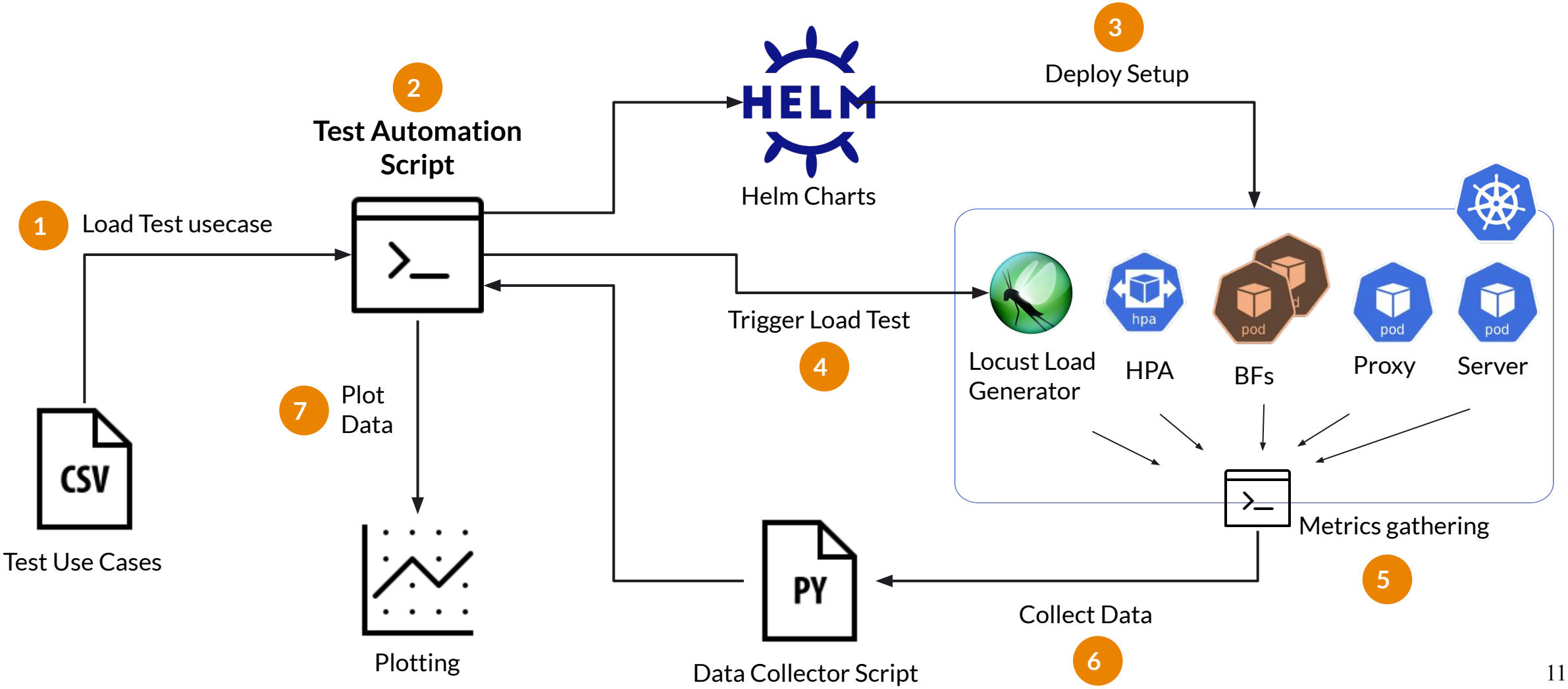
Experiments

- No VPA
- Increasing HPA threshold for CPU utilization
- Increasing number of users in locust load generator
- Data collected:
  - Response time
  - CPU usage
    - 1 CPU = 1s = 1000 milicore = 1 000 000 000 n
    - CPU in nanoseconds/1 000 000 = CPU in milicore

Test Scenarios

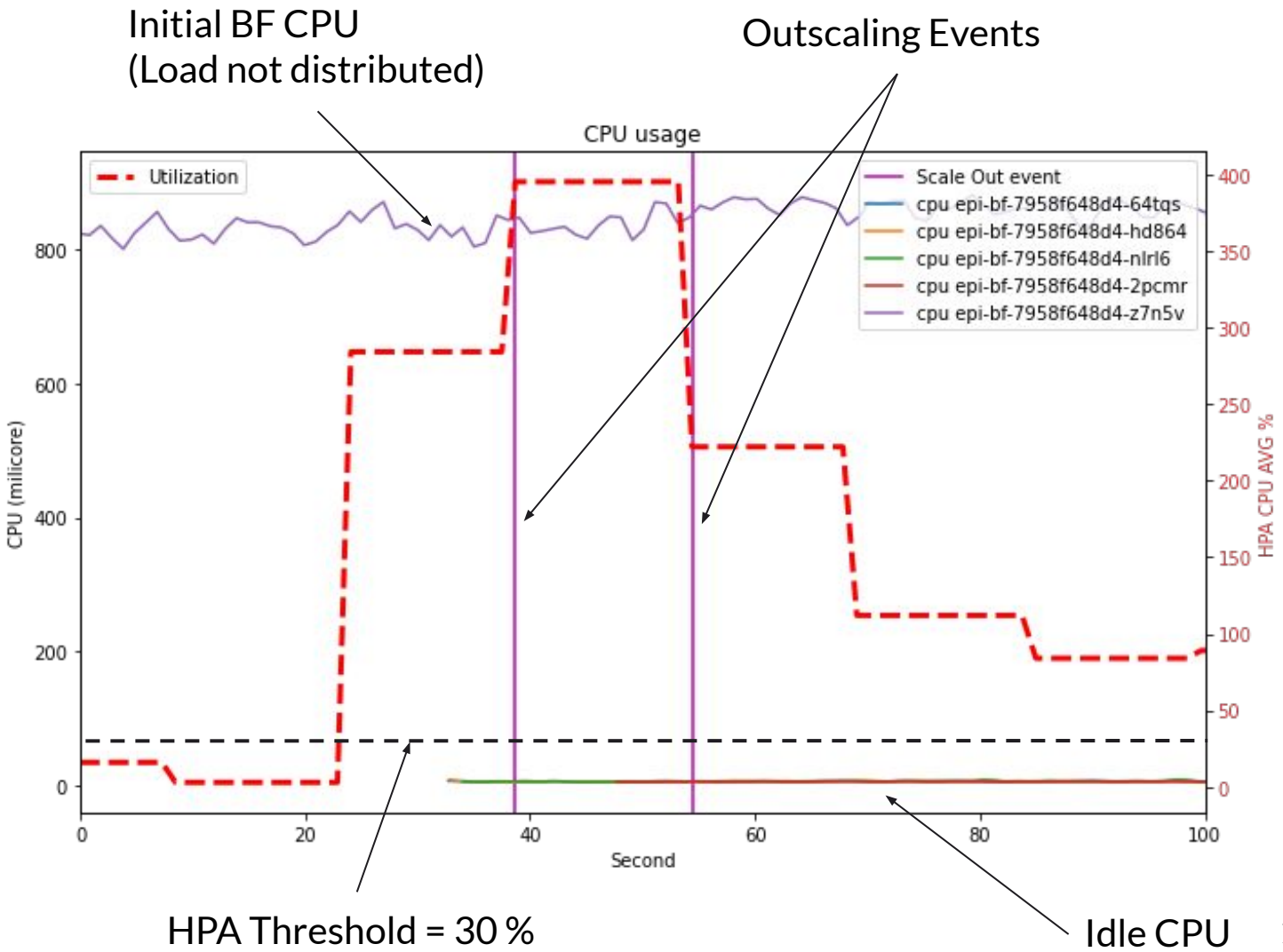
Test #	# of Users	Spawn rate	Run time (Sec)	HPA MAX REPLICAS	HPA Utilization	CPU Limit	Memory Limit
1	**10**	1	100	5	**30**	1000m	500Mi
2	**10**	1	100	5	**50**	1000m	500Mi
3	**10**	1	100	5	**80**	1000m	500Mi
4	**10**	1	100	5	**90**	1000m	500Mi
5	**100**	10	100	5	**30**	1000m	500Mi
6	**100**	10	100	5	**50**	1000m	500Mi
7	**100**	10	100	5	**80**	1000m	500Mi
8	**100**	10	100	5	**90**	1000m	500Mi
9	**1000**	100	100	5	**30**	1000m	500Mi
10	**1000**	100	100	5	**50**	1000m	500Mi
11	**1000**	100	100	5	**80**	1000m	500Mi
12	**1000**	100	100	5	**90**	1000m	500Mi
13	**10000**	1000	100	5	**30**	1000m	500Mi
14	**10000**	1000	100	5	**50**	1000m	500Mi
15	**10000**	1000	100	5	**80**	1000m	500Mi
16	**10000**	1000	100	5	**90**	1000m	500Mi

Setup - Automated Deployment and Testing



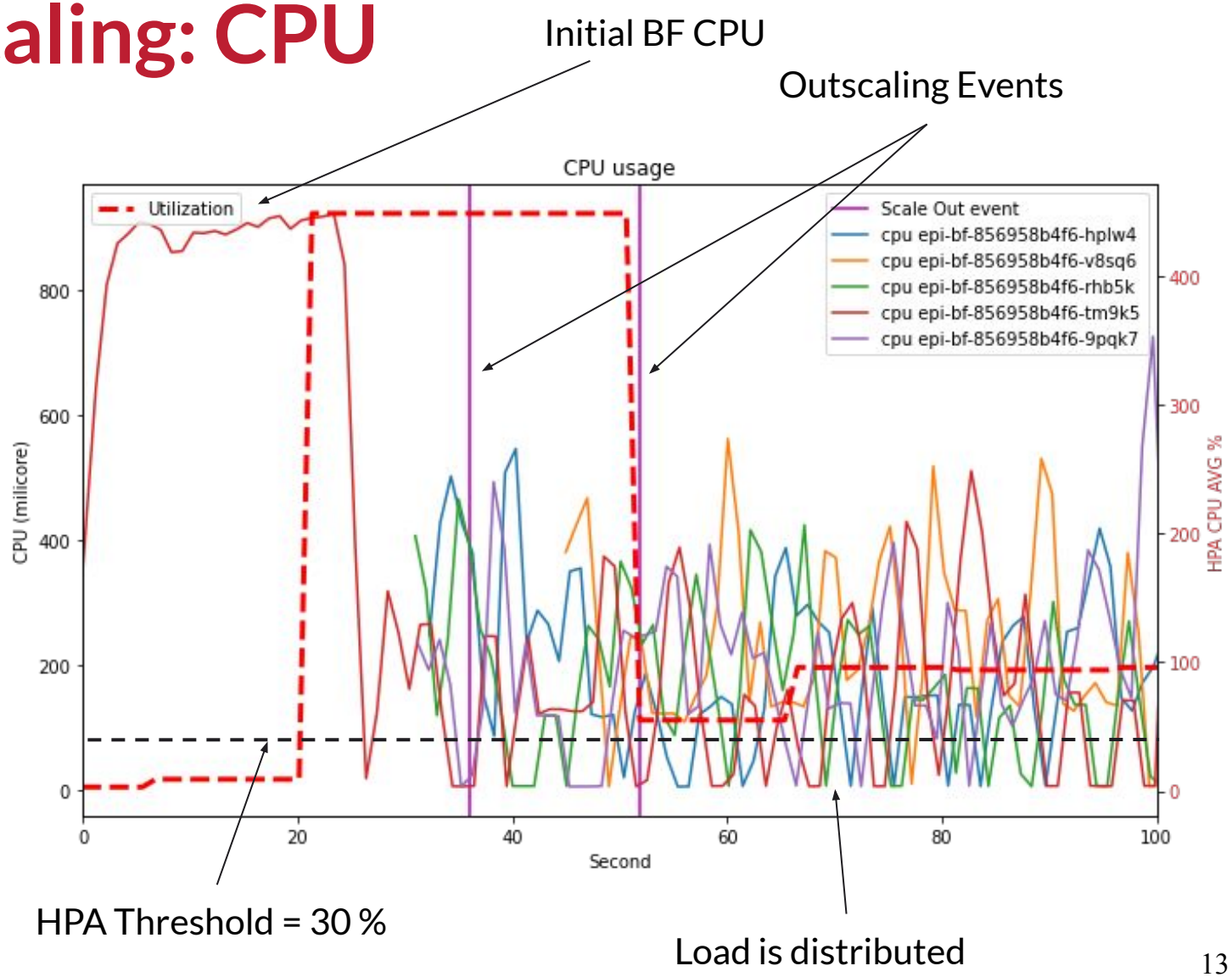
Results - Initial Test

- By default, Locust uses an HTTP persistent connection (**keep-alive**).
- Kubernetes service pin the session to single Bridging Function.
- The **load is not distributed** after scale-out events.
- **'Connection': 'close'** can be used to change the behavior.
- Vertical scaling is a better fit for such traffic.



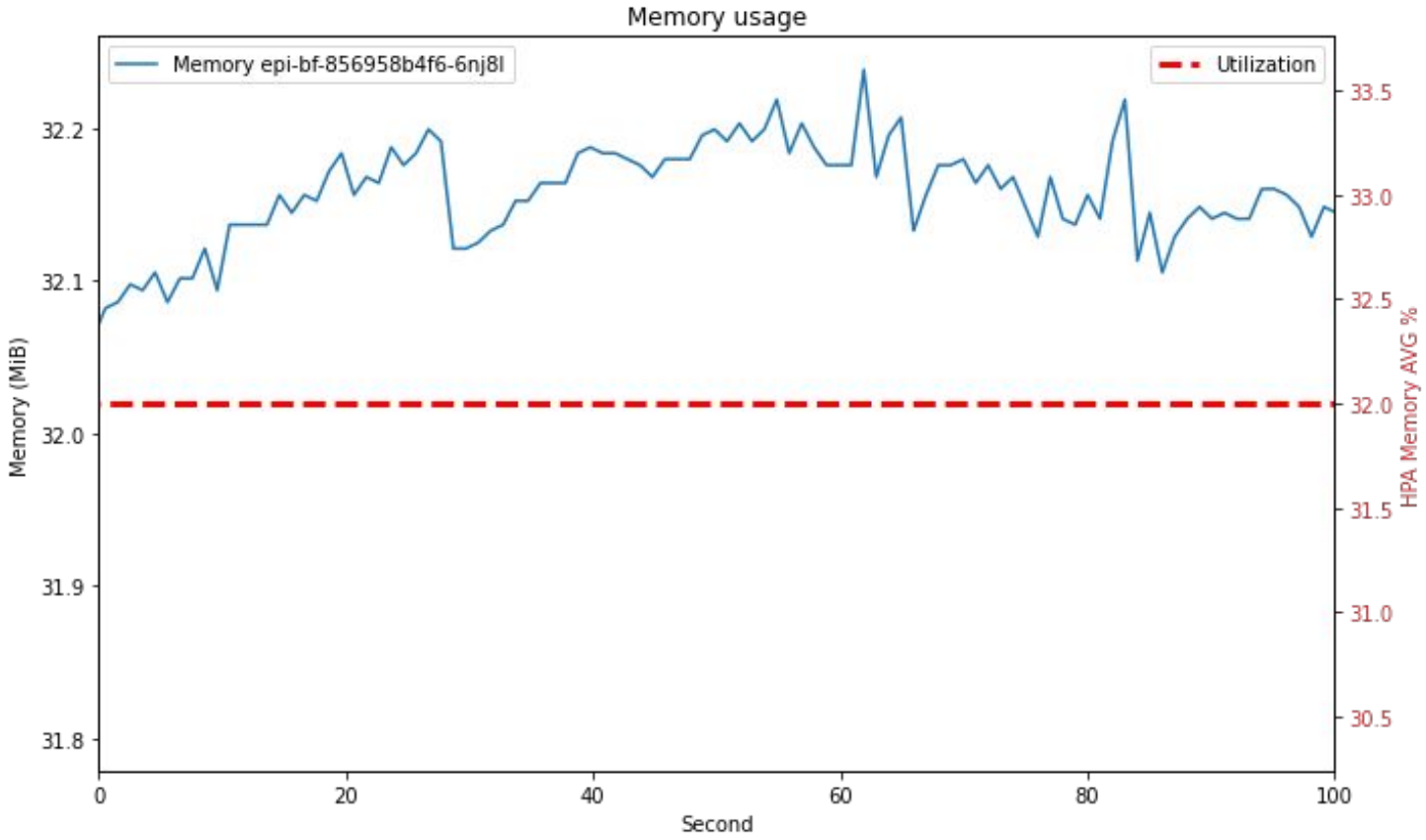
Results - Horizontal Scaling: CPU

- Bridging function CPU is more balanced with short lived sessions.
- The HPA operates on a **ratio between desired and current metrics**.
- The scale out event is more aggressive if the threshold is low.
  - Bursty traffic



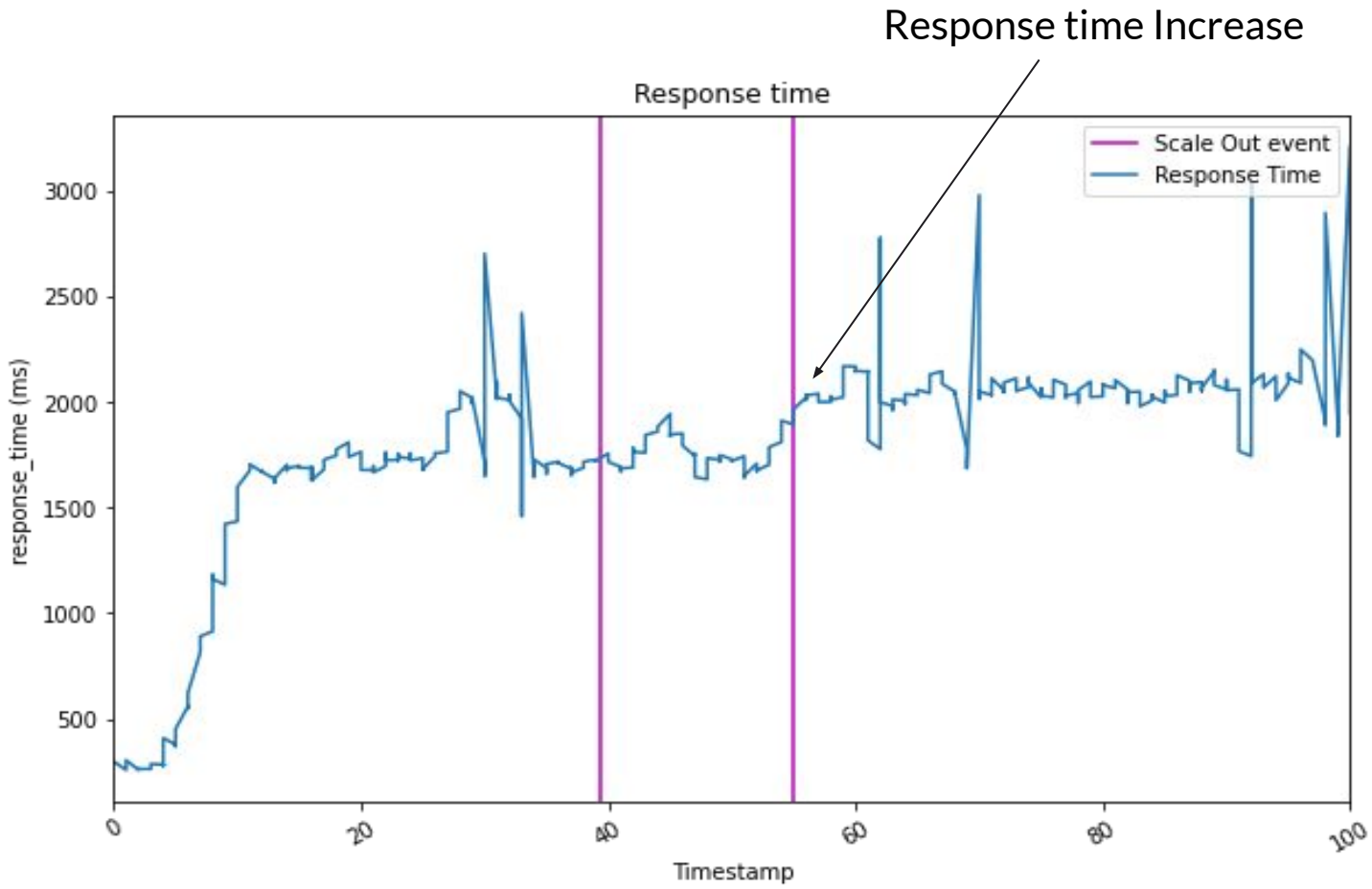
Results - Horizontal Scaling: Memory

- CPU is the primary metric used during testing.
- The tested bridging function consumed **steady memory.**
- Memory usage not decreasing after scaling out.



Results - Horizontal Scaling: Response Time

- The Locust load generator records the response time of the HTTP requests.
- Response time increases after the autoscaling events due to the **load-balancing overhead**.
- Bridging function high CPU is not causing an increase in response time.
- The processing logic of the bridging function used is very insignificant.



Conclusion

- The metric for triggering Autoscaling is highly dependant on the bridging function logic.
  - CPU.
  - Memory.
  - Custom Metrics.
- Long lived sessions doesn’t benefits from Horizontal Scaling.
- There is no single optimum threshold to trigger autoscaling.
- Overhead added by autoscaling can impact application traffic.

Future Work

- Use data analysis and machine learning to find the best threshold for different metrics and traffic use cases.
- Test with different application traffic.
- Use production grade bridging function.
- Research the impact of bridging function chaining.
- Vertical Pod Autoscaling.

Thank You

https://github.com/mohanadelamin/rp2-epif

Mohanad Elamin
University of Amsterdam
melamin@os3.nl

Pim Paardekooper
University of Amsterdam
ppaardekooper@os3.nl

Supervisor:

Jamila Alsayed Kassem
UvA