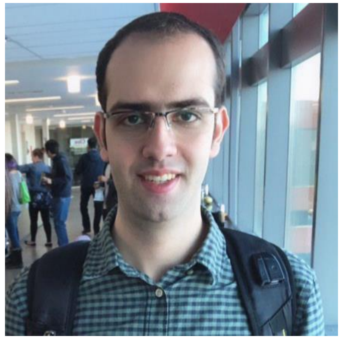
A Framework for Satisfying the Performance Requirements of Containerized Software Systems Through Multi-Versioning



Sara Gholami

Alireza Goli

Cor-Paul Bezemer

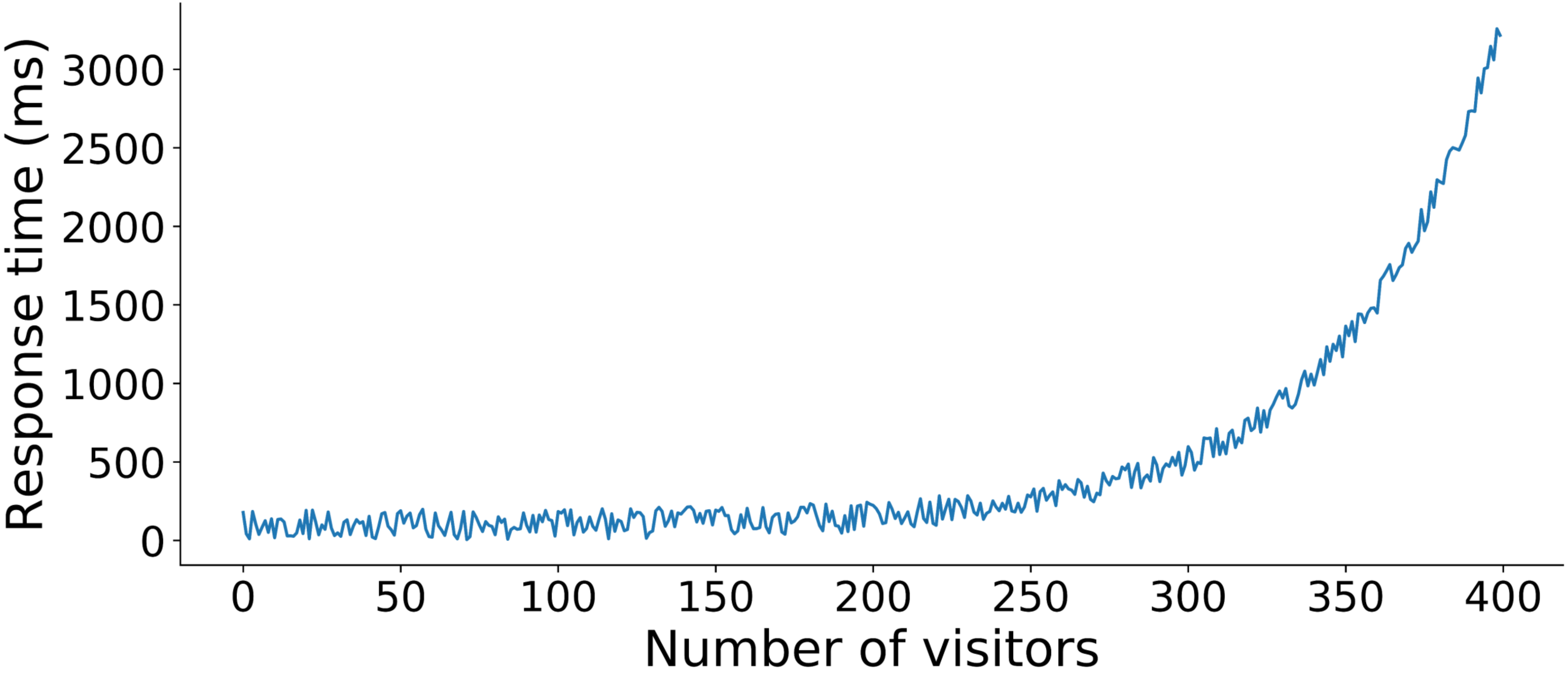
Hamzeh Khazaei

UNIVERSITY OF ALBERTA
EDMONTON·ALBERTA·CANADA

ASGAARD
Analytics of Software, Games and Repository Data

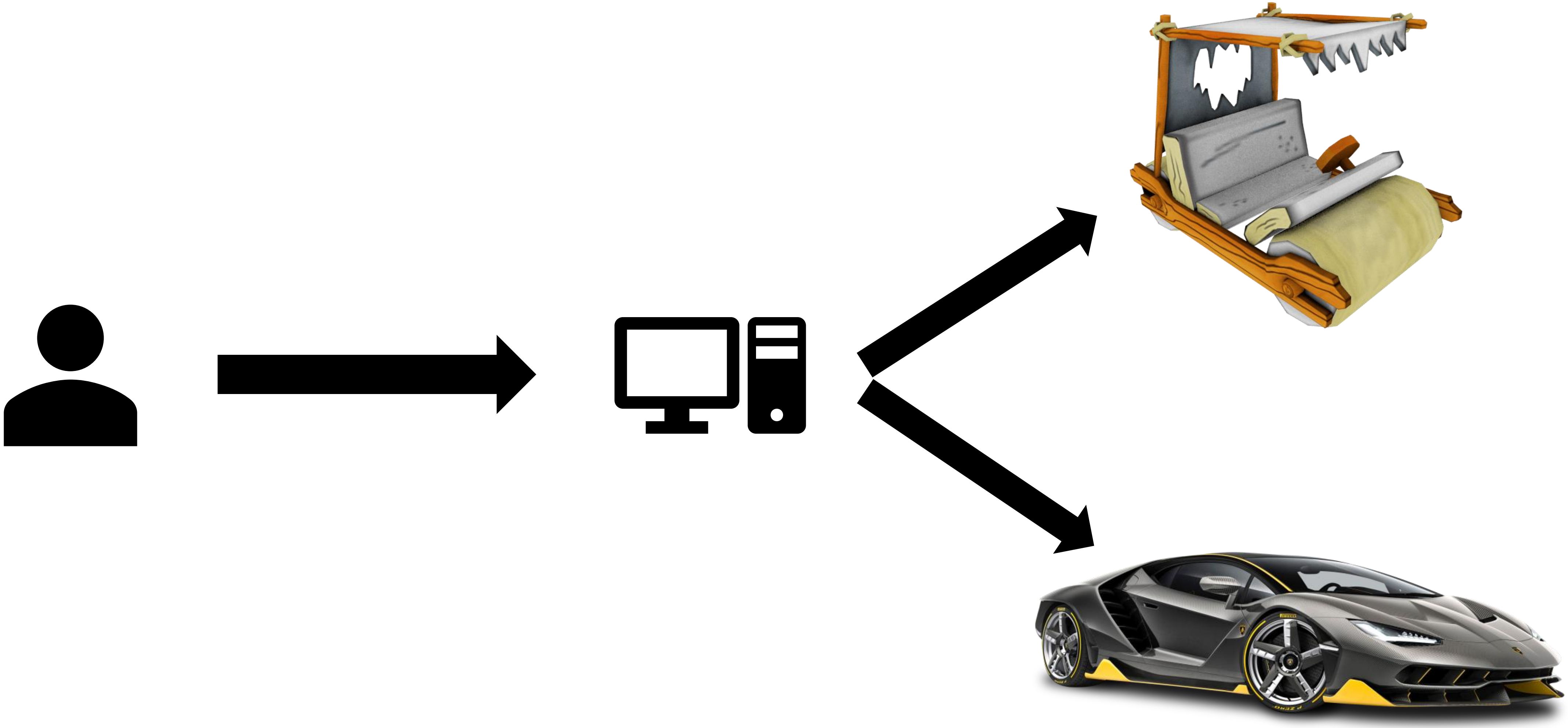
YORK
UNIVERSITÉ
UNIVERSITY

An example problem: The Slashdot Effect

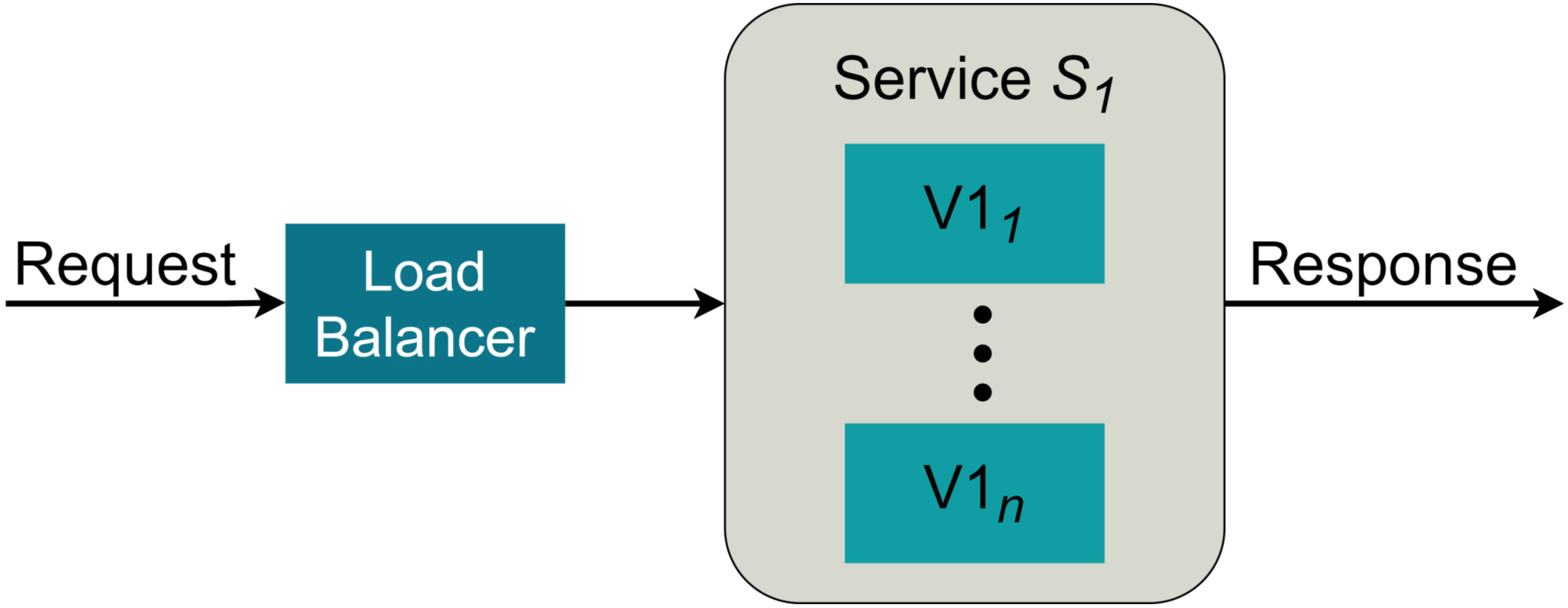


One solution to the Slashdot Effect is to increase the resources

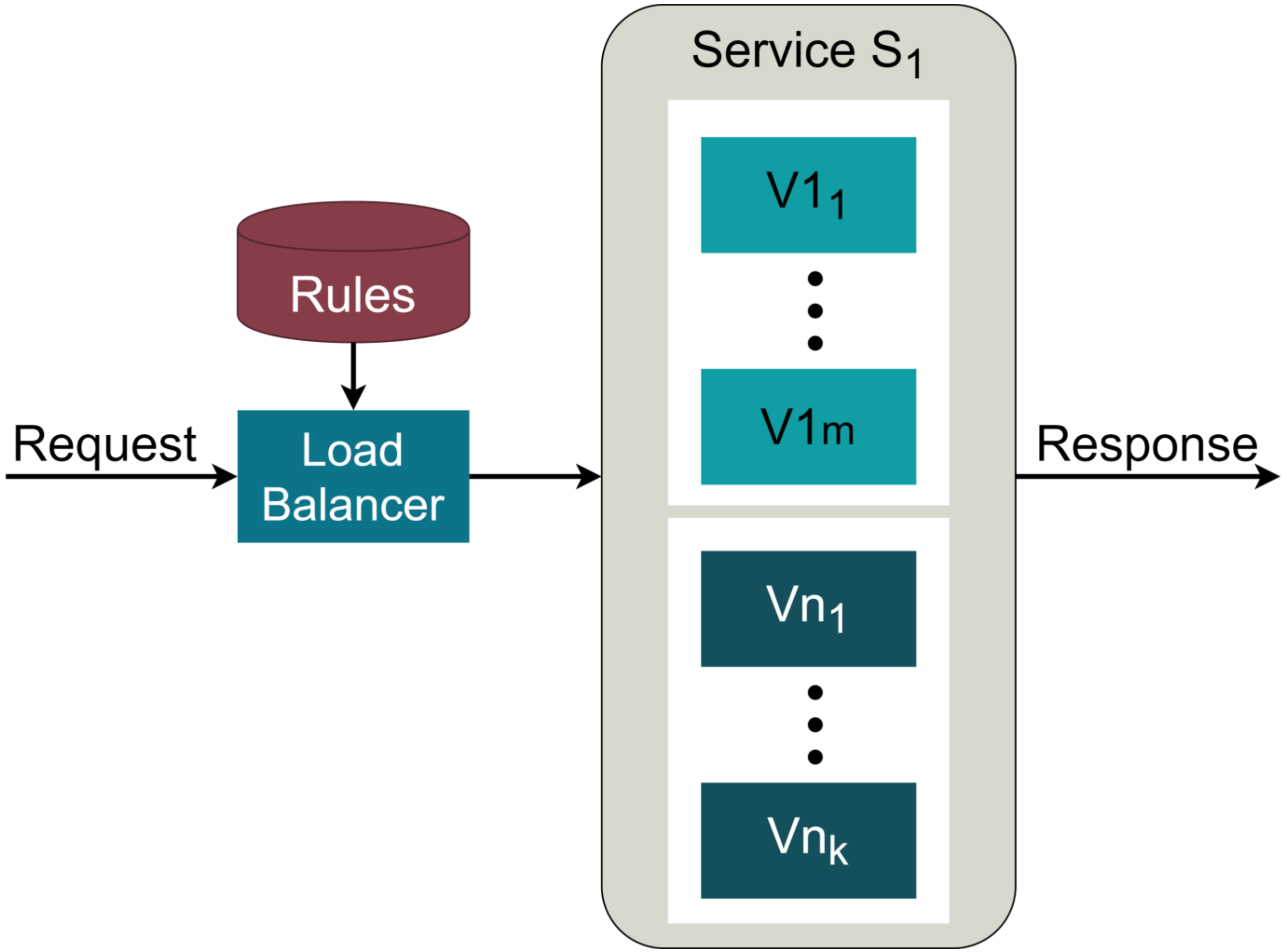
Another solution is to manage the available resources better



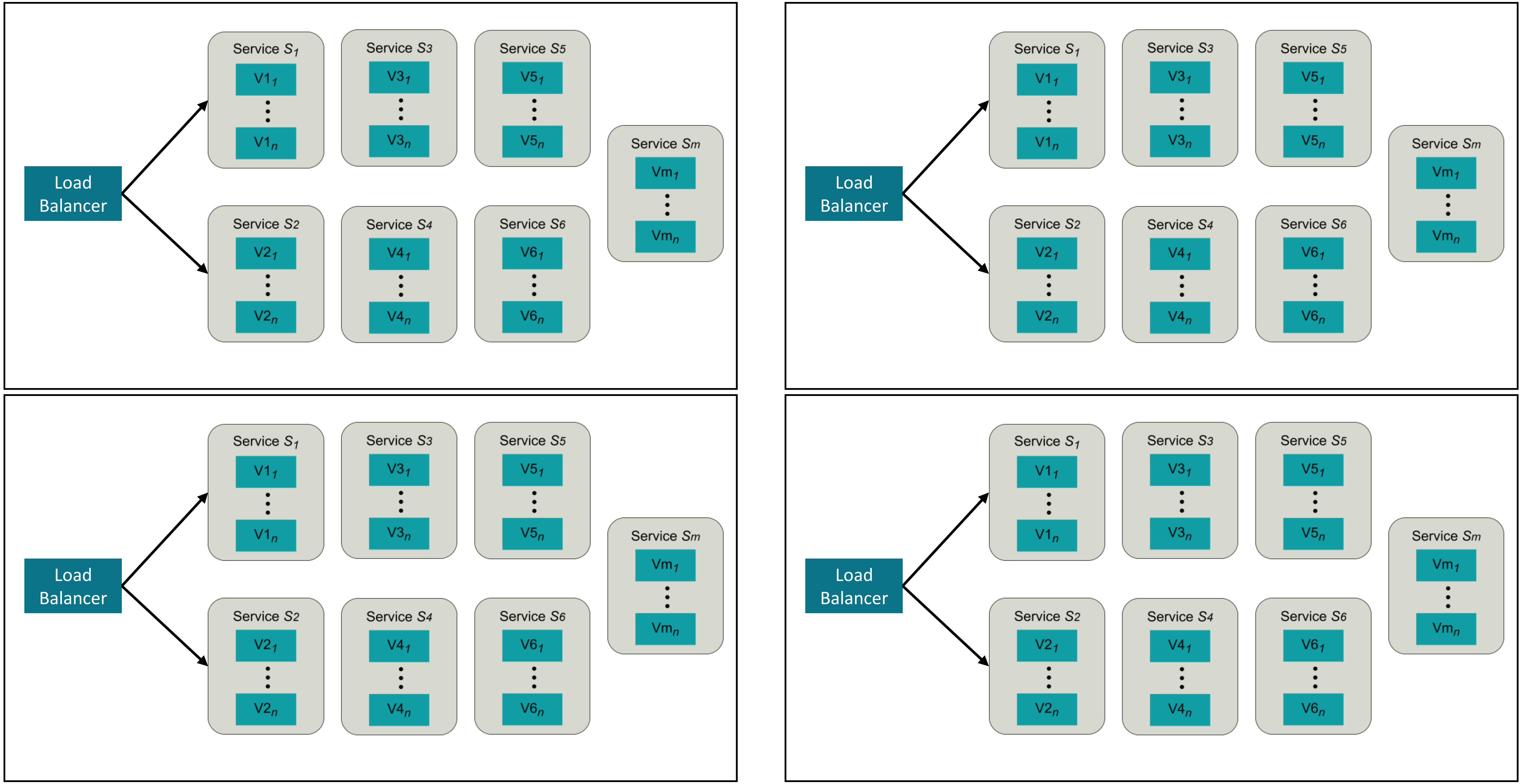
High-level architecture of a service in Docker, where requests are load balanced in a Round Robin manner



We present Docker with multi-versioning: DockerMV



DockerMV takes away the service management nightmare



Load Balancer

Service S_1 $V1_1$ ⋮ $V1_n$

Service S_2 $V2_1$ ⋮ $V2_n$

Service S_3 $V3_1$ ⋮ $V3_n$

Service S_4 $V4_1$ ⋮ $V4_n$

Service S_5 $V5_1$ ⋮ $V5_n$

Service S_6 $V6_1$ ⋮ $V6_n$

Service S_m Vm_1 ⋮ Vm_n

Load Balancer

Service S_1 $V1_1$ ⋮ $V1_n$

Service S_2 $V2_1$ ⋮ $V2_n$

Service S_3 $V3_1$ ⋮ $V3_n$

Service S_4 $V4_1$ ⋮ $V4_n$

Service S_5 $V5_1$ ⋮ $V5_n$

Service S_6 $V6_1$ ⋮ $V6_n$

Service S_m Vm_1 ⋮ Vm_n

Load Balancer

Service S_1 $V1_1$ ⋮ $V1_n$

Service S_2 $V2_1$ ⋮ $V2_n$

Service S_3 $V3_1$ ⋮ $V3_n$

Service S_4 $V4_1$ ⋮ $V4_n$

Service S_5 $V5_1$ ⋮ $V5_n$

Service S_6 $V6_1$ ⋮ $V6_n$

Service S_m Vm_1 ⋮ Vm_n

Load Balancer

Service S_1 $V1_1$ ⋮ $V1_n$

Service S_2 $V2_1$ ⋮ $V2_n$

Service S_3 $V3_1$ ⋮ $V3_n$

Service S_4 $V4_1$ ⋮ $V4_n$

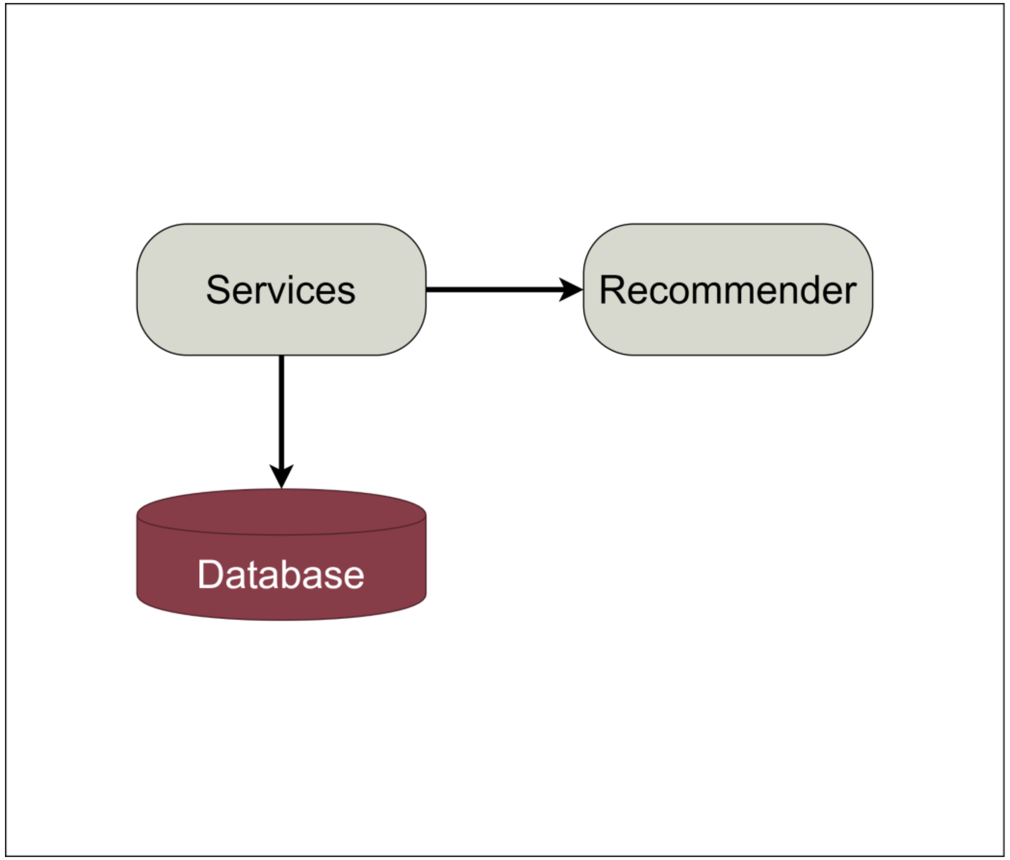
Service S_5 $V5_1$ ⋮ $V5_n$

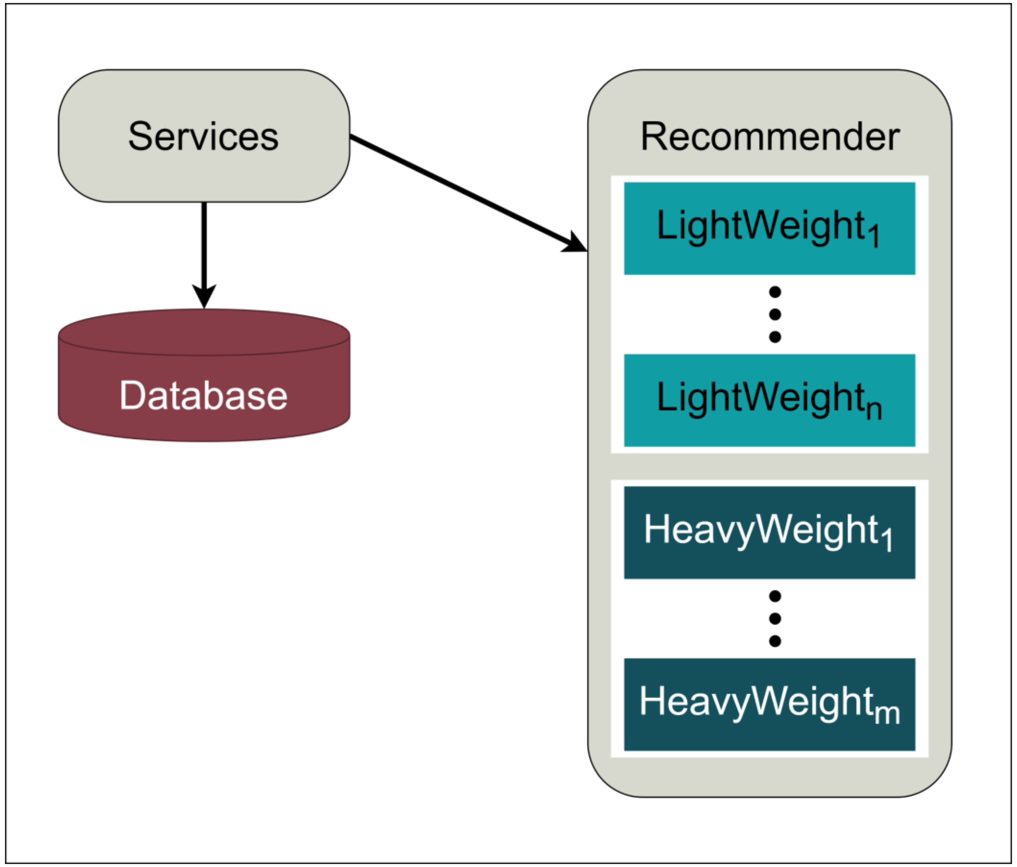
Service S_6 $V6_1$ ⋮ $V6_n$

Service S_m Vm_1 ⋮ Vm_n

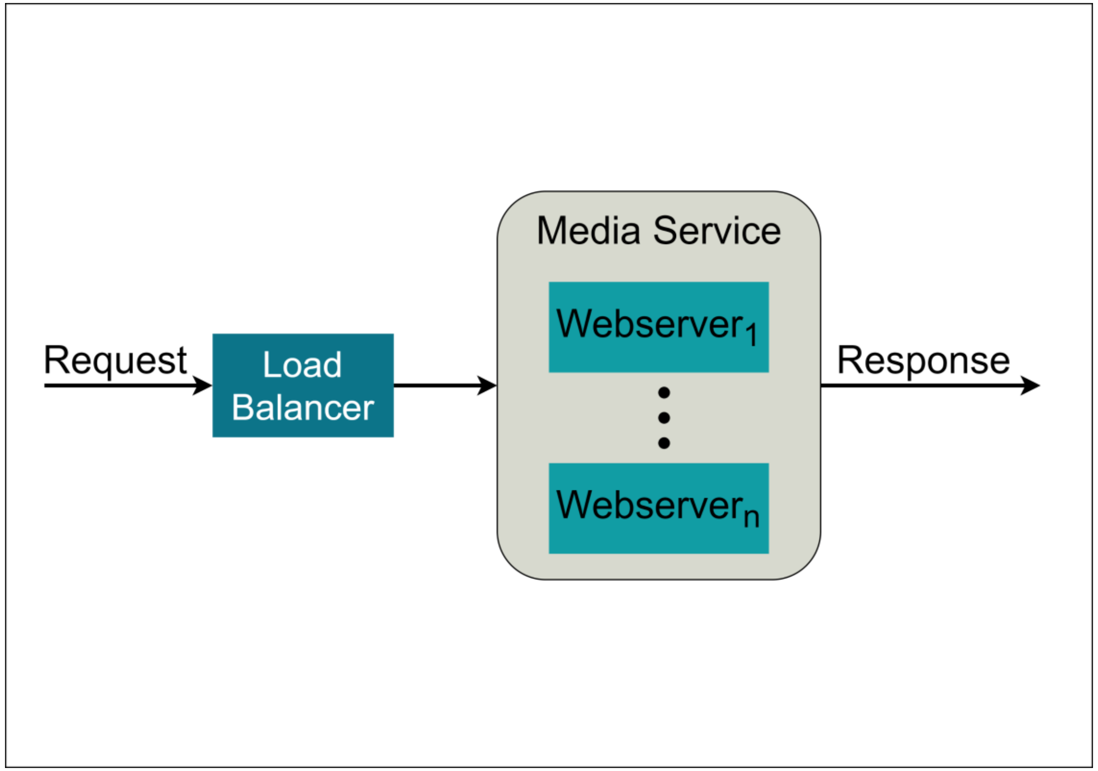
How do we evaluate DockerMV?

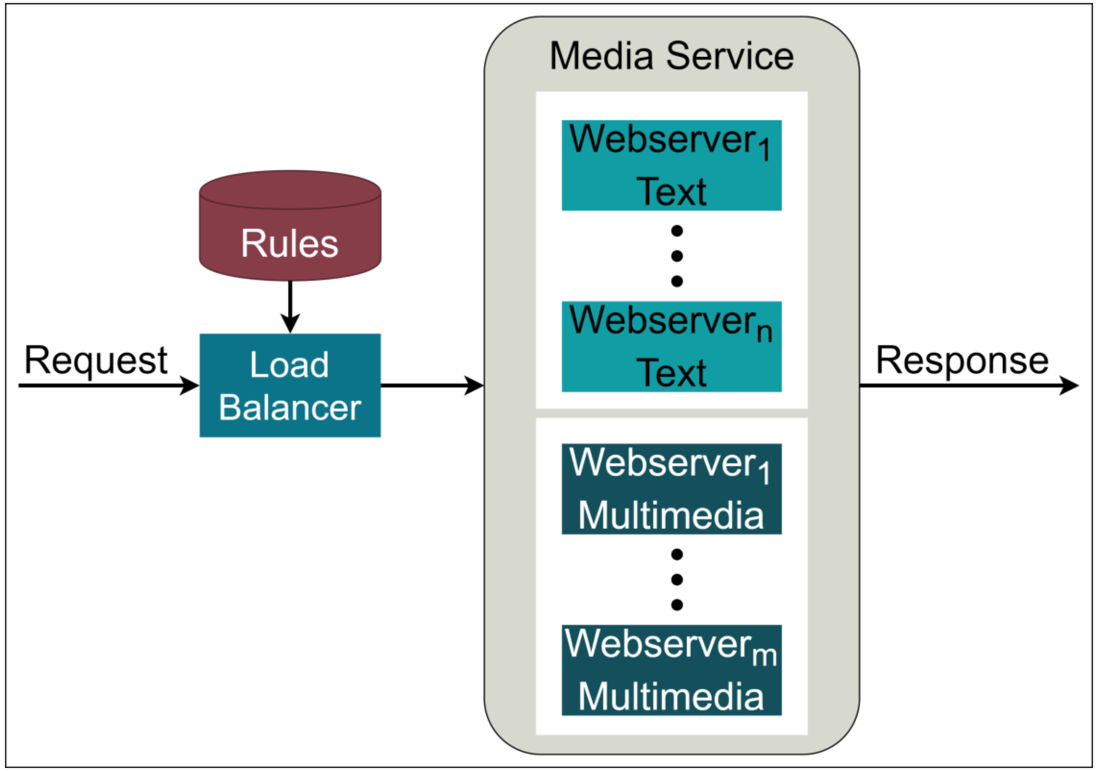
The TeaStore application, an online webstore application





The Znn application, a three-tier online news application





We conducted three experiments for the TeaStore application

Ideal case experiment
Recommender with multiple training
(Only heavy weight)

Adaptive experiment
Adaptive load distribution
(Mix of heavy and light weight)

Worst case experiment
Recommender with single training
(Only light weight)

We conducted three experiments for the Znn application

Ideal case experiment
Multimedia responses only
(Only heavy weight)

Worst case experiment
Text responses only
(Only light weight)

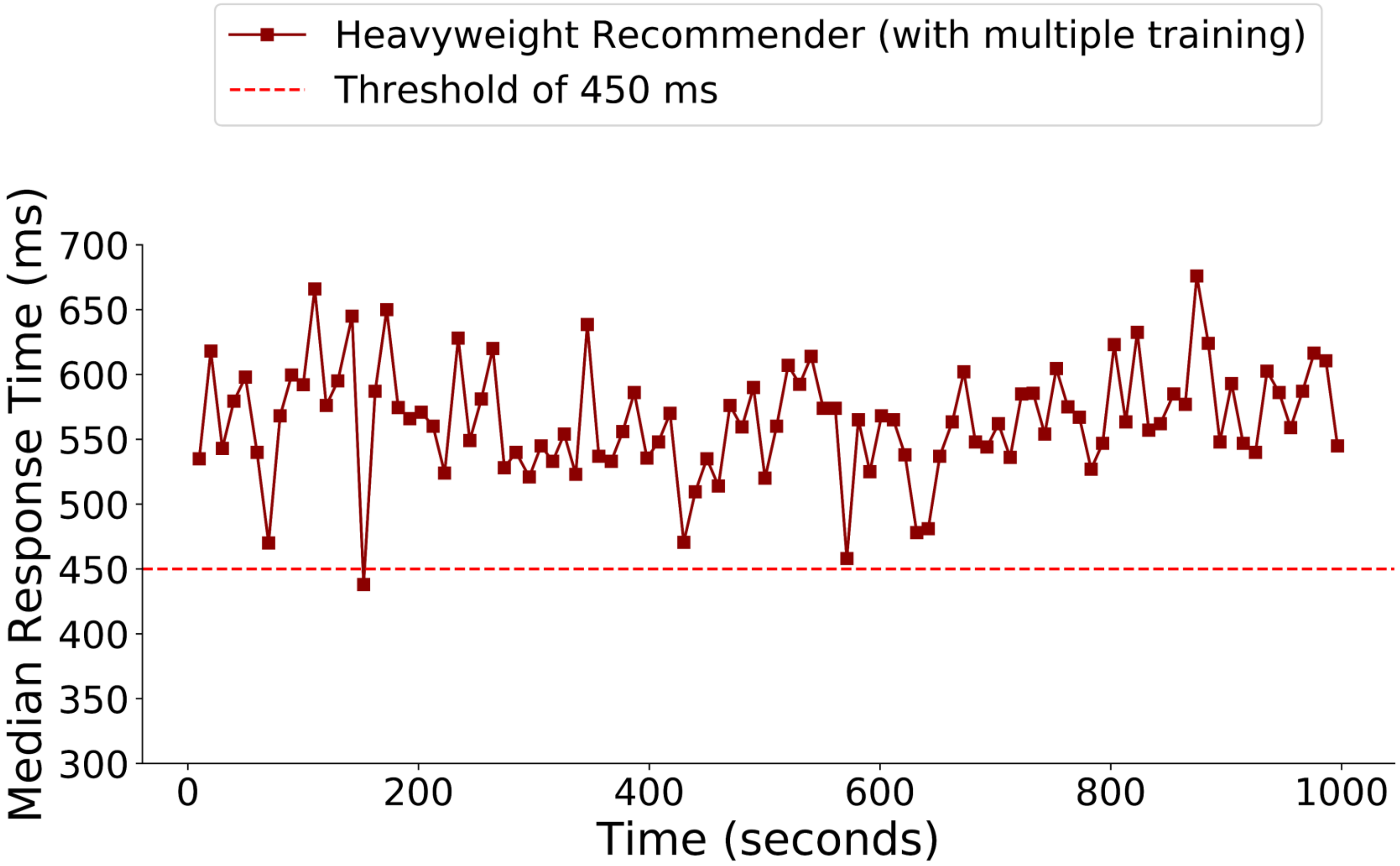
Adaptive experiment
Adaptive load distribution
(Mix of heavy and light weight)

Workload applied to the TeaStore application

100 users sending HTTP requests for 1,000 seconds
(Almost 97 requests per second)

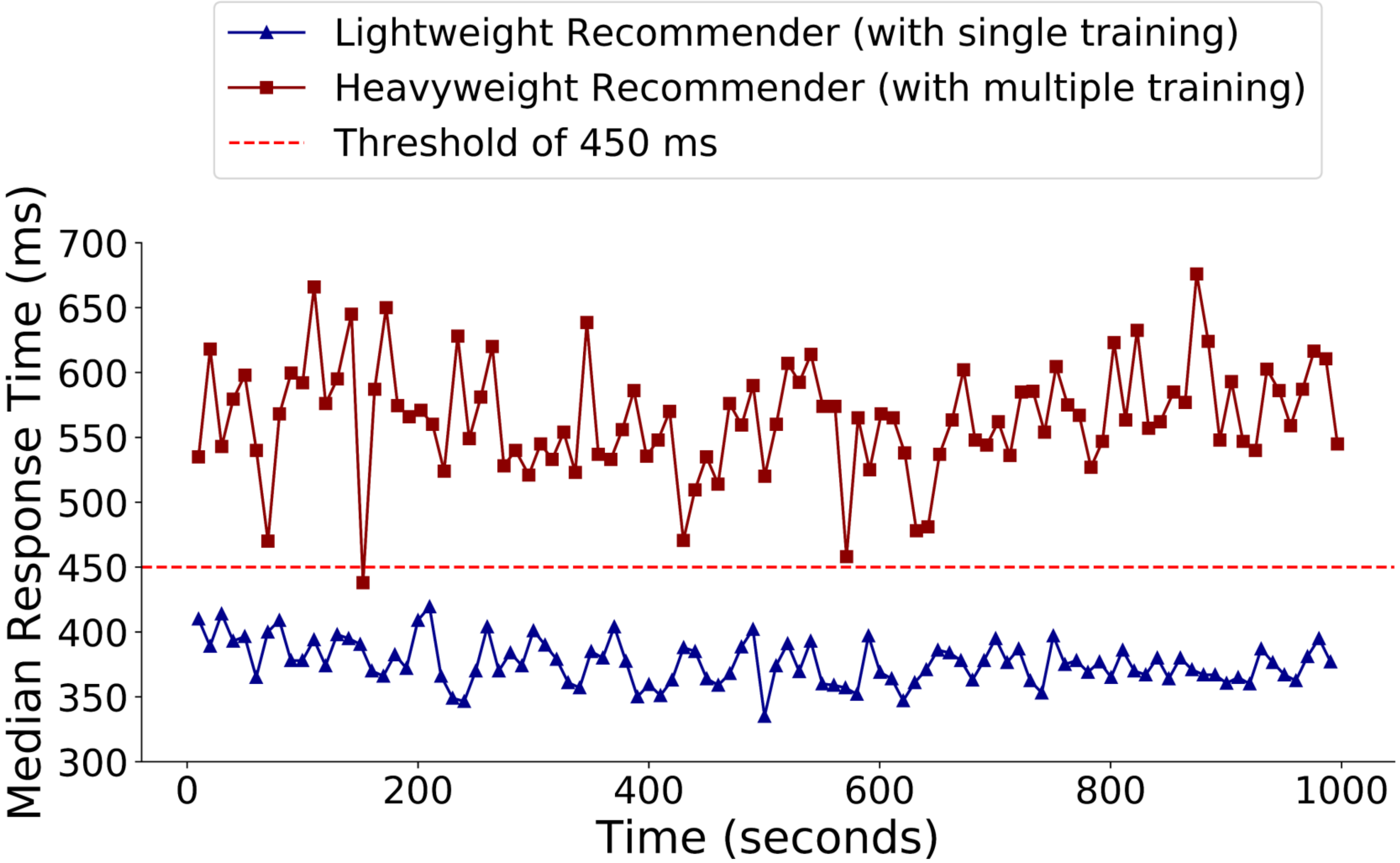
We defined the SLA response time threshold to be 450 ms

In the TeaStore application's ideal case experiment, the response time exceeds the threshold under the load



Ideal case exp.

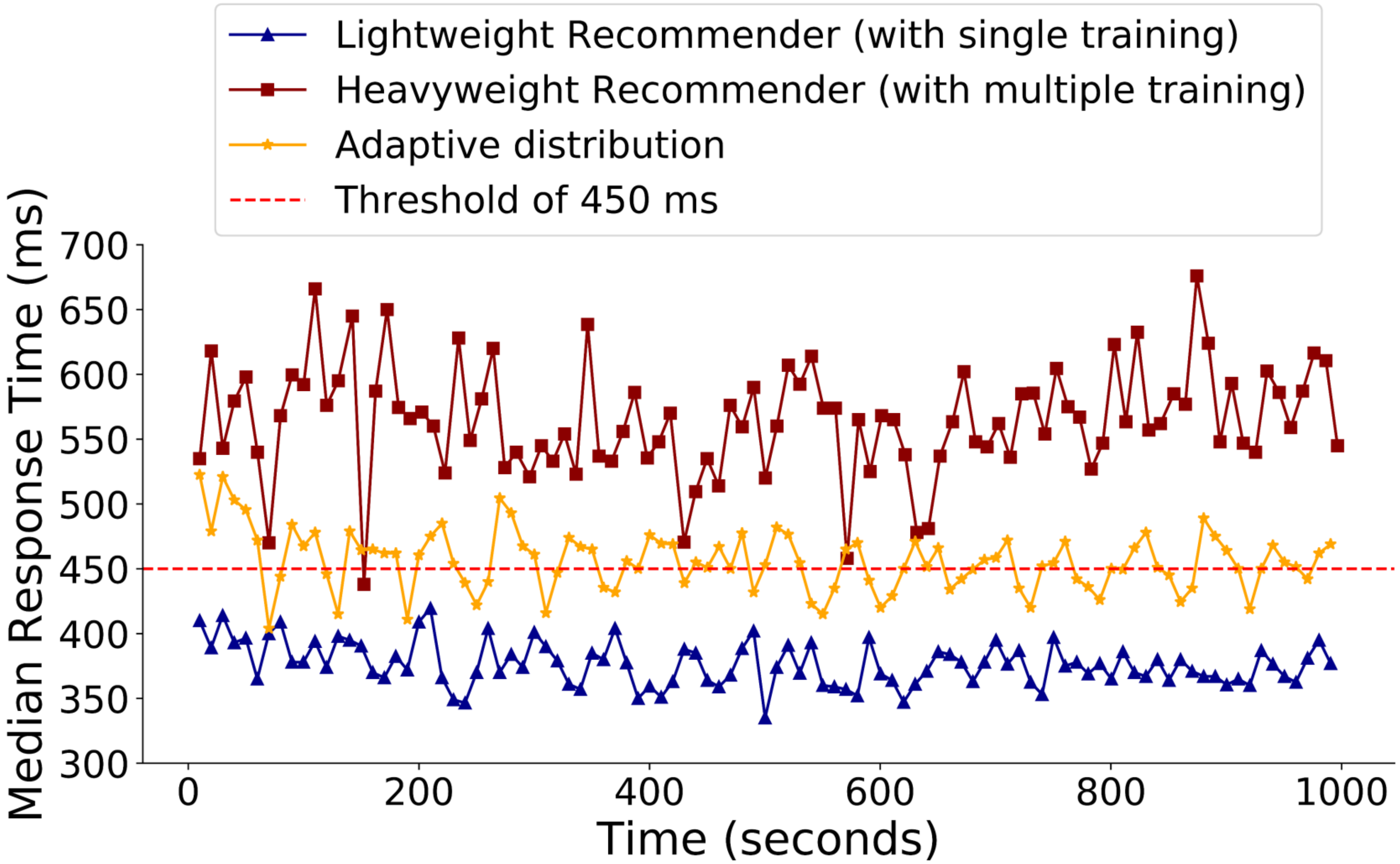
In the TeaStore application's worst case experiment, the response time falls below the threshold under the load



😞

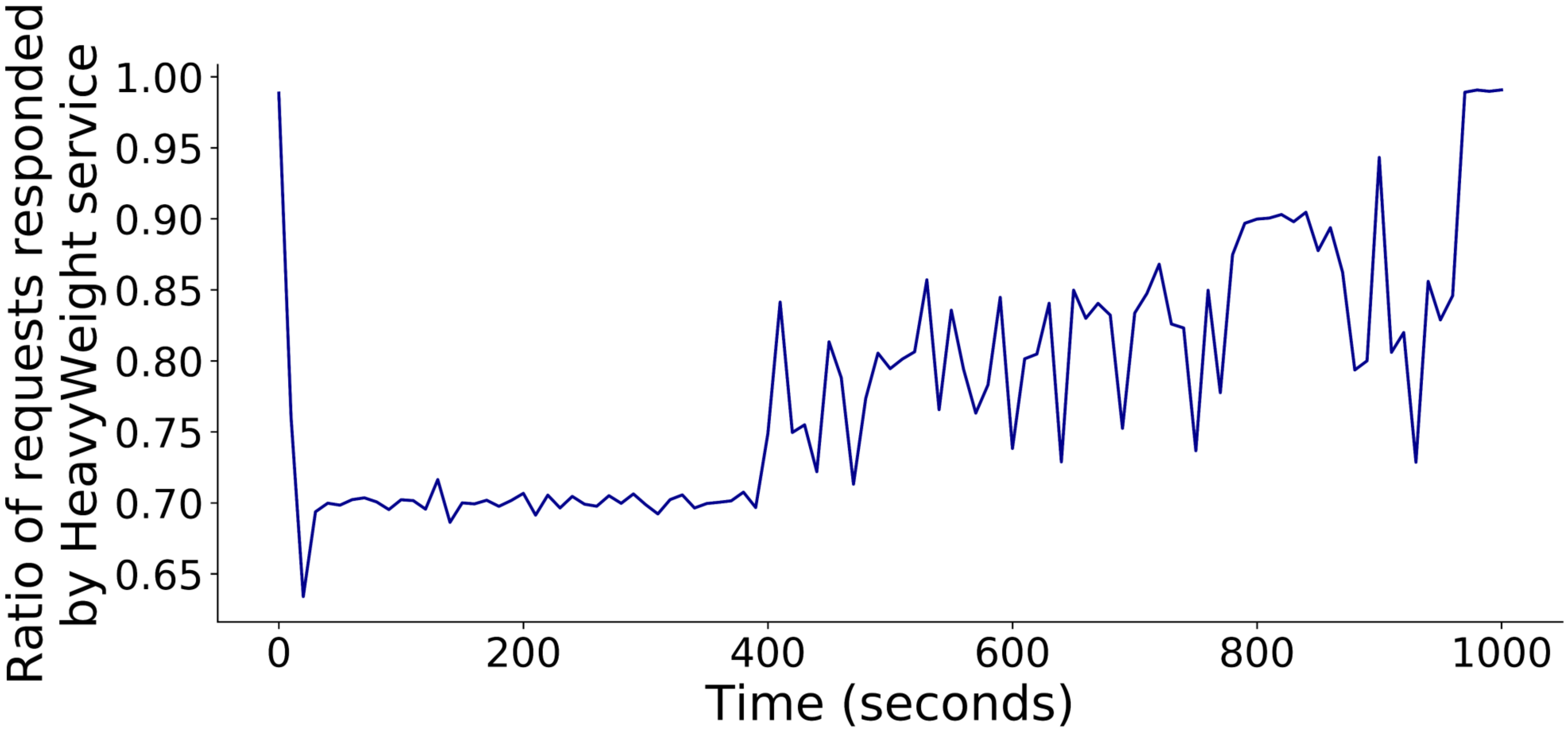
Worst case exp.

In the TeaStore application's adaptive experiment, the response time is maintained close to the threshold

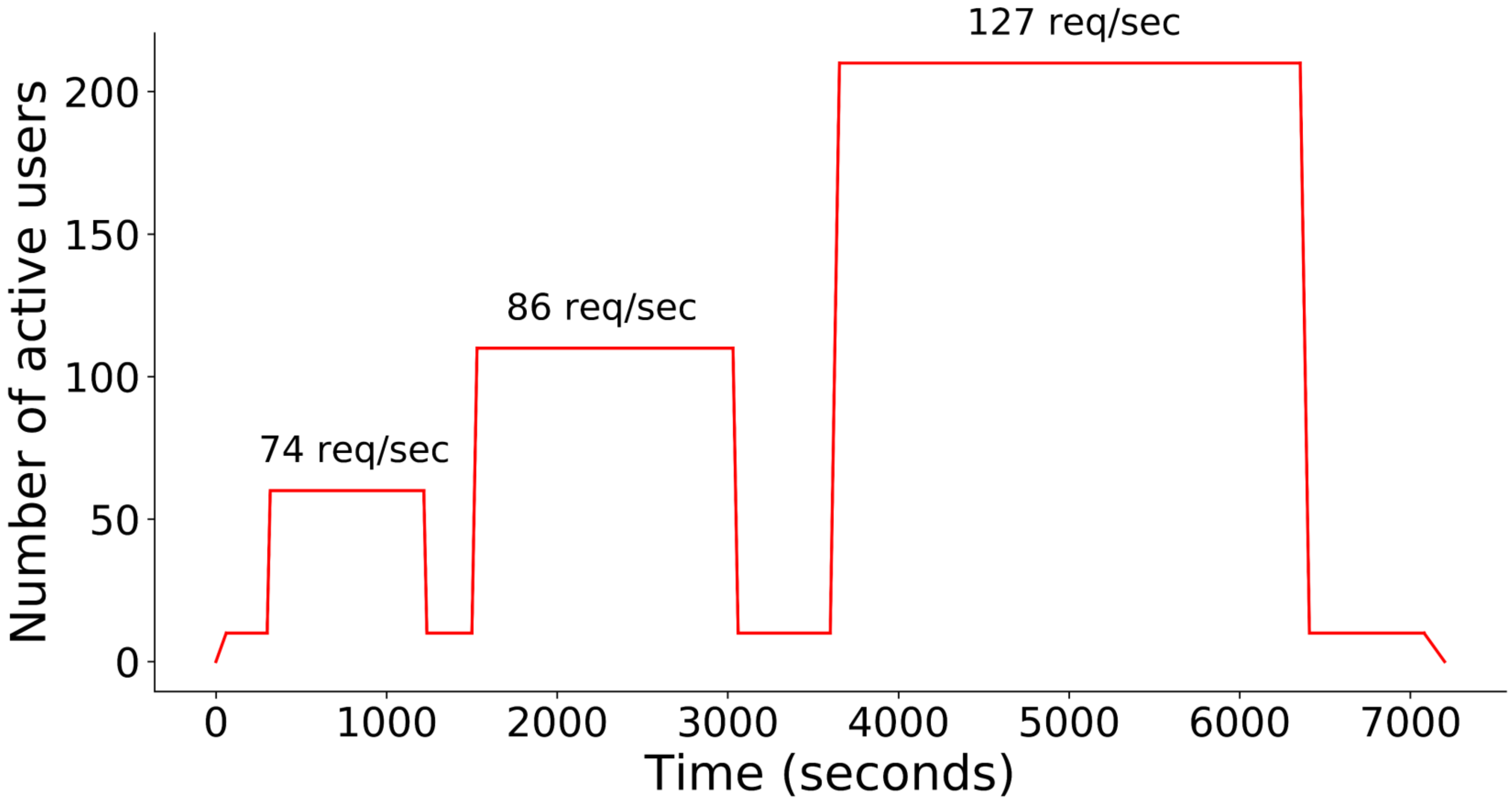


Adaptive exp.

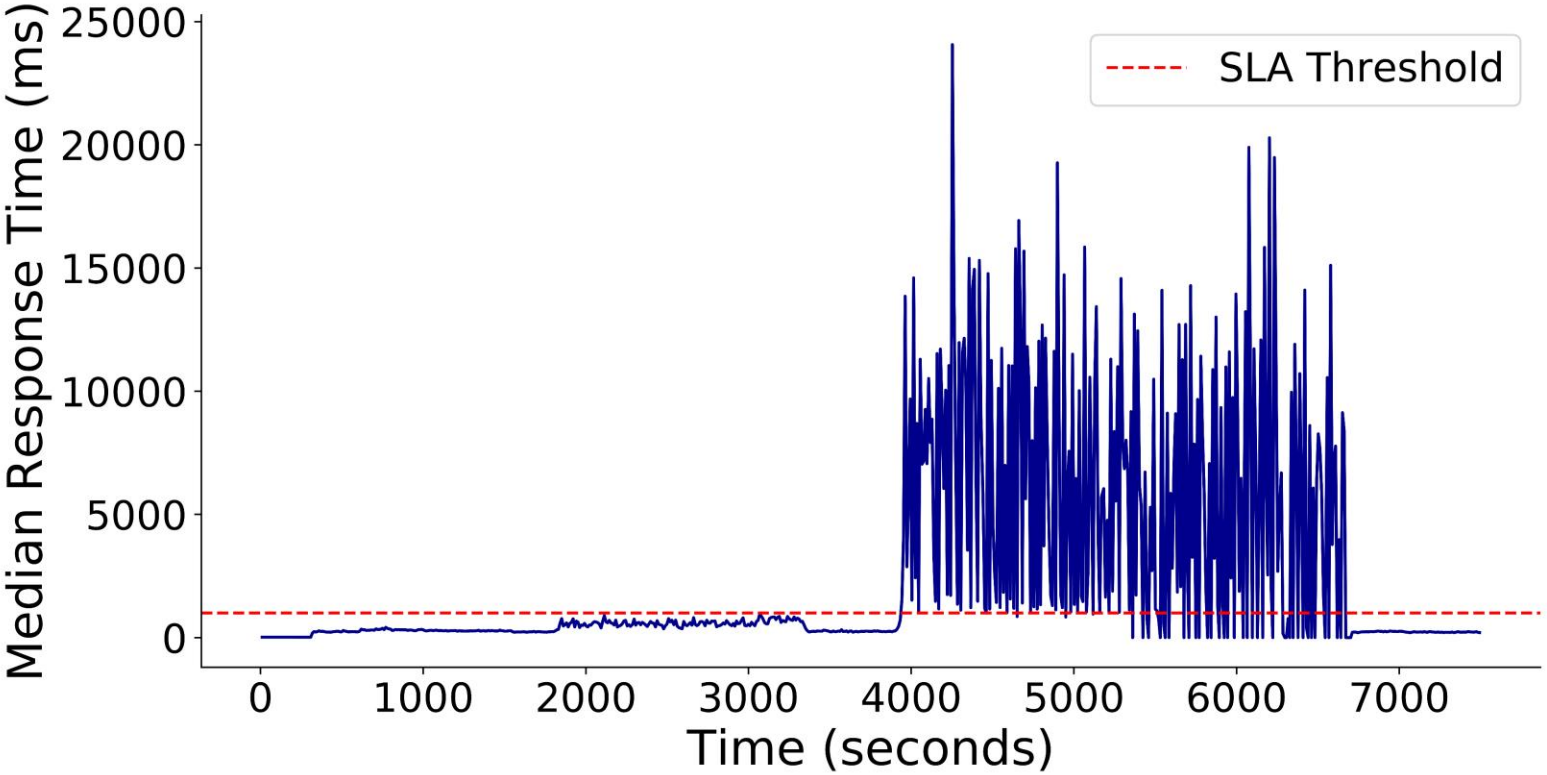
The ratio of requests responded by the HeavyWeight version of the Recommender service in the TeaStore's adaptive experiment



Workload applied to the Znn application

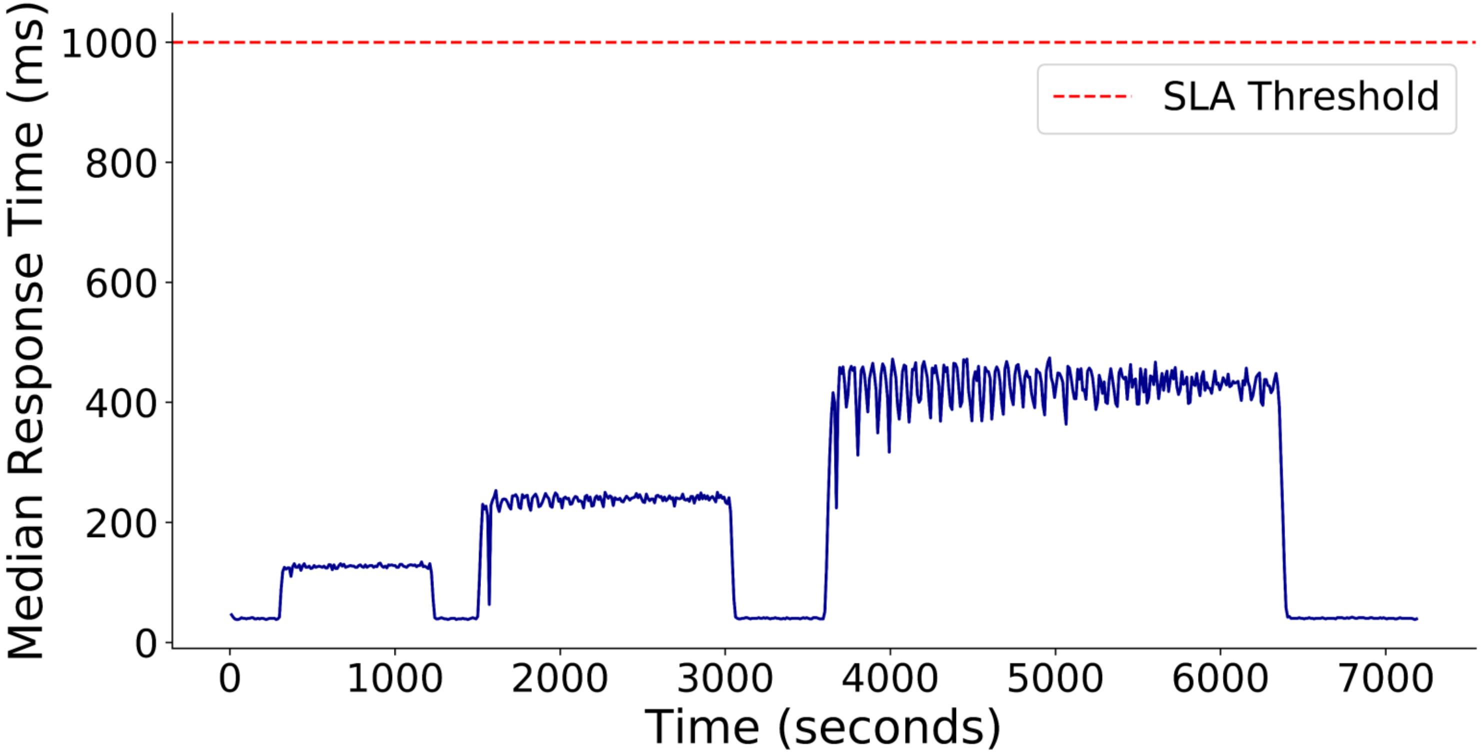


In the Znn application's ideal case experiment, the response time exceeds the threshold under the load



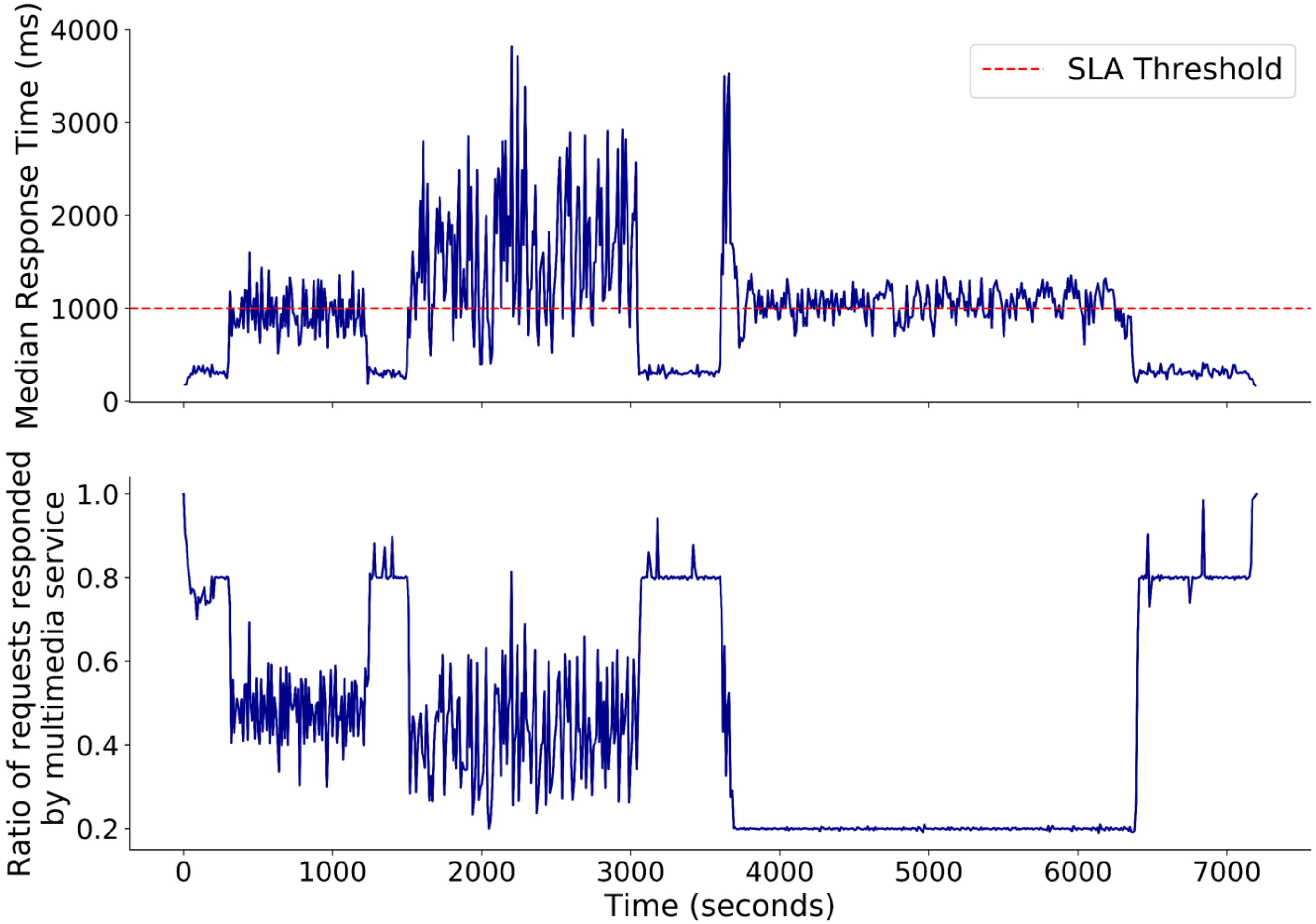
Ideal case exp.

In the Znn application's worst case experiment, the response time falls below the threshold under the load



Worst case exp.

In the Znn application's adaptive experiment, the response time is maintained around the threshold

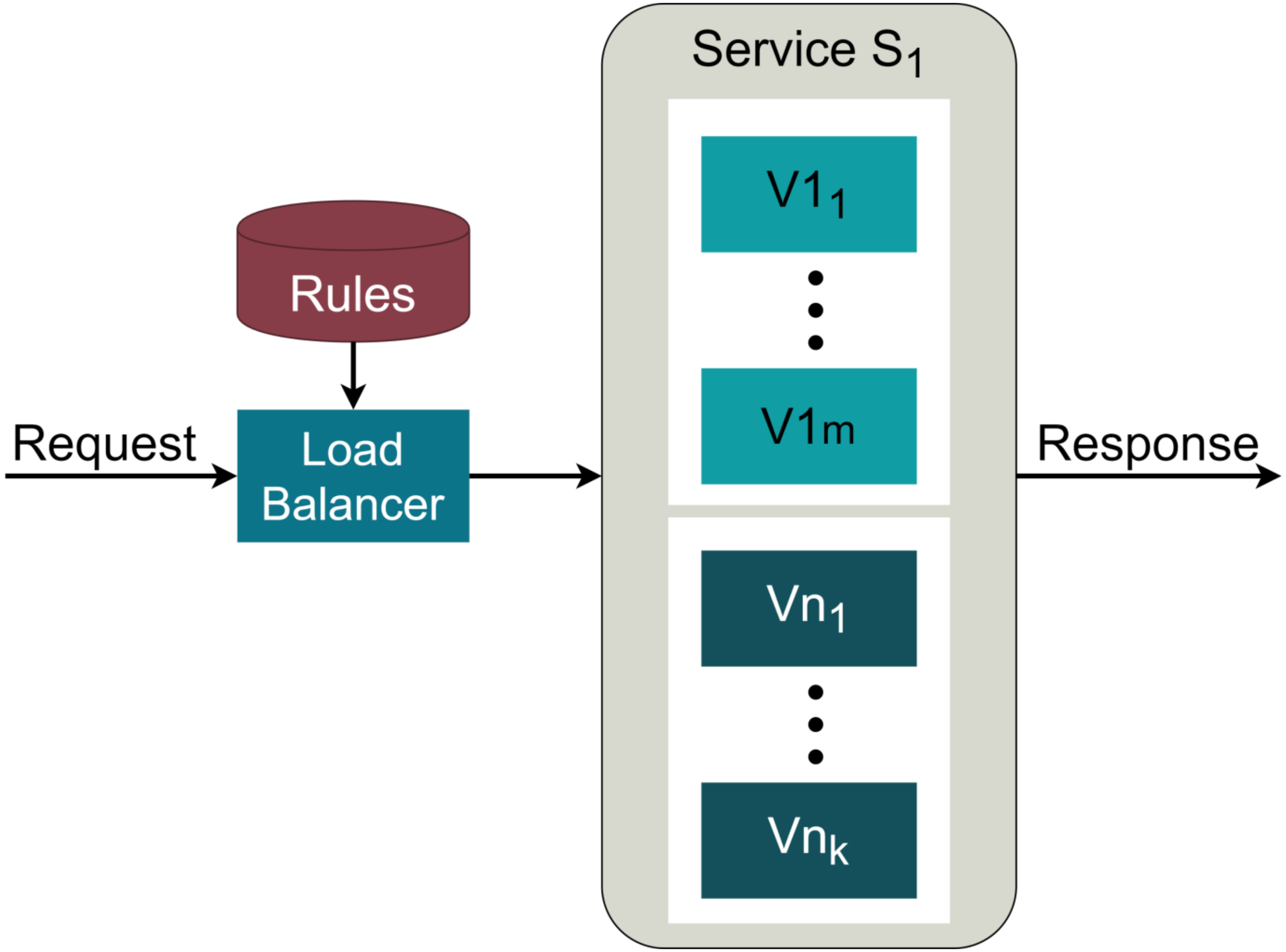


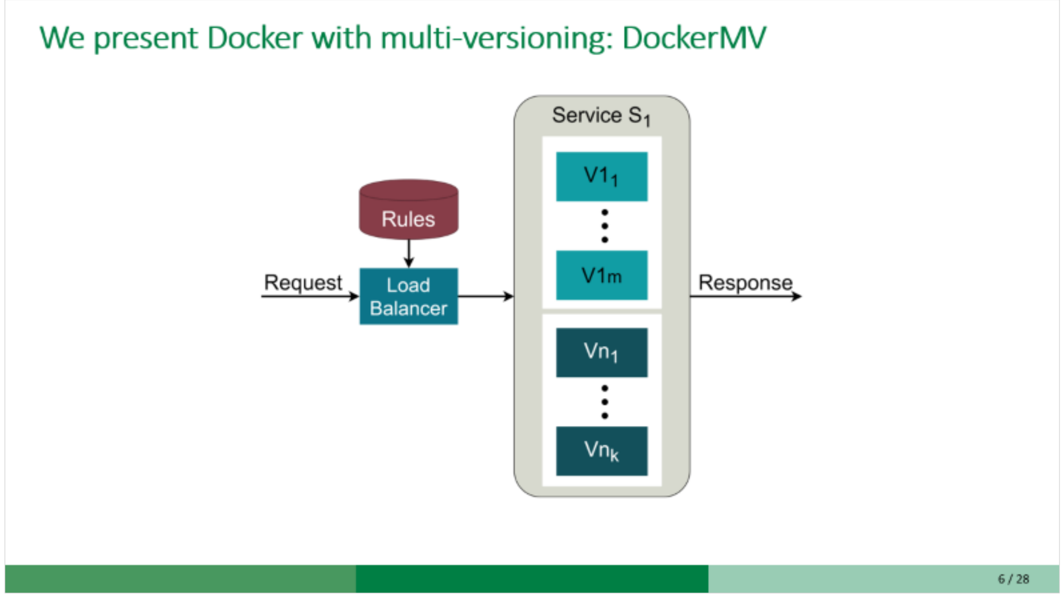
Adaptive exp.

DockerMV source code is publicly available on GitHub

https://github.com/pacslab/DockerMV

We present Docker with multi-versioning: DockerMV

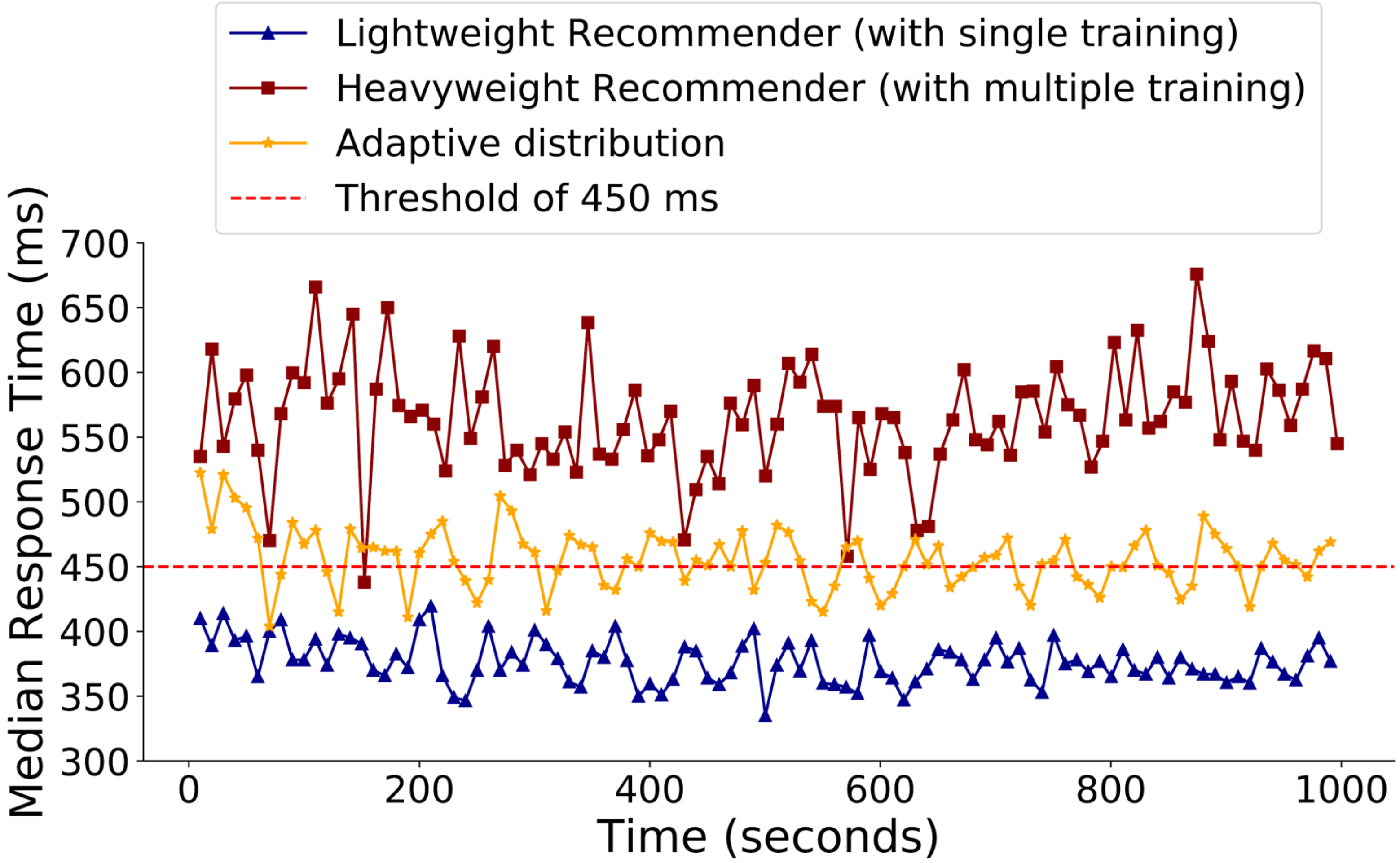




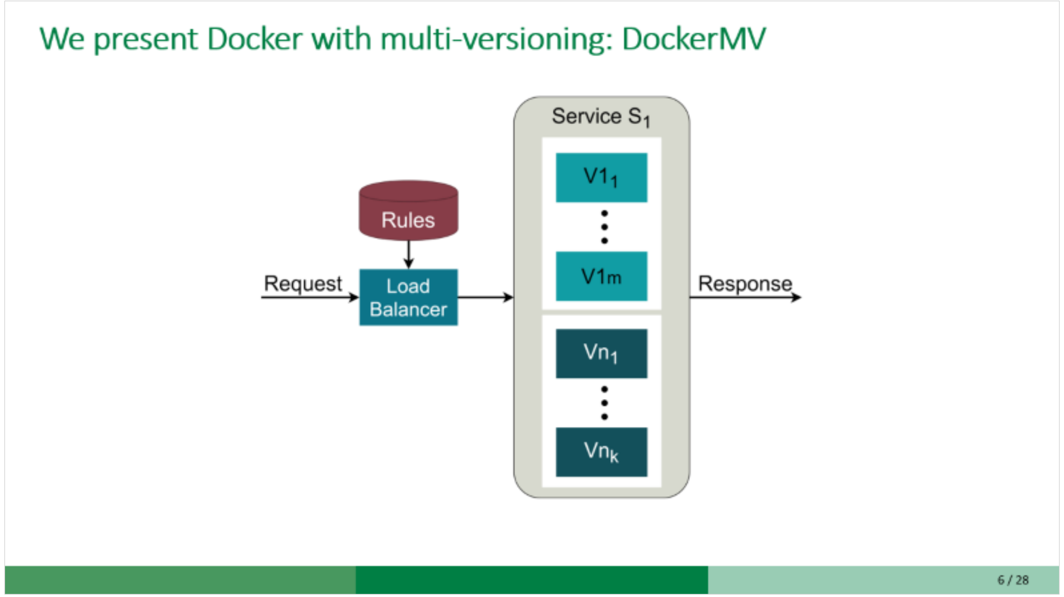
We present Docker with multi-versioning: DockerMV

6 / 28

In the TeaStore application’s adaptive experiment, the response time is close to the threshold

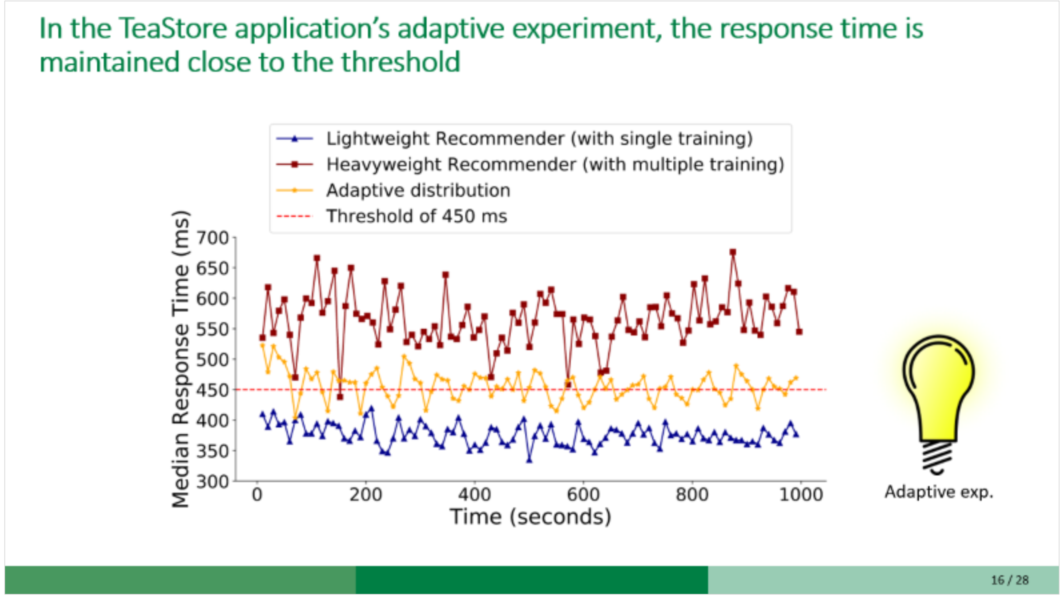


Adaptive exp.



We present Docker with multi-versioning: DockerMV

6 / 28

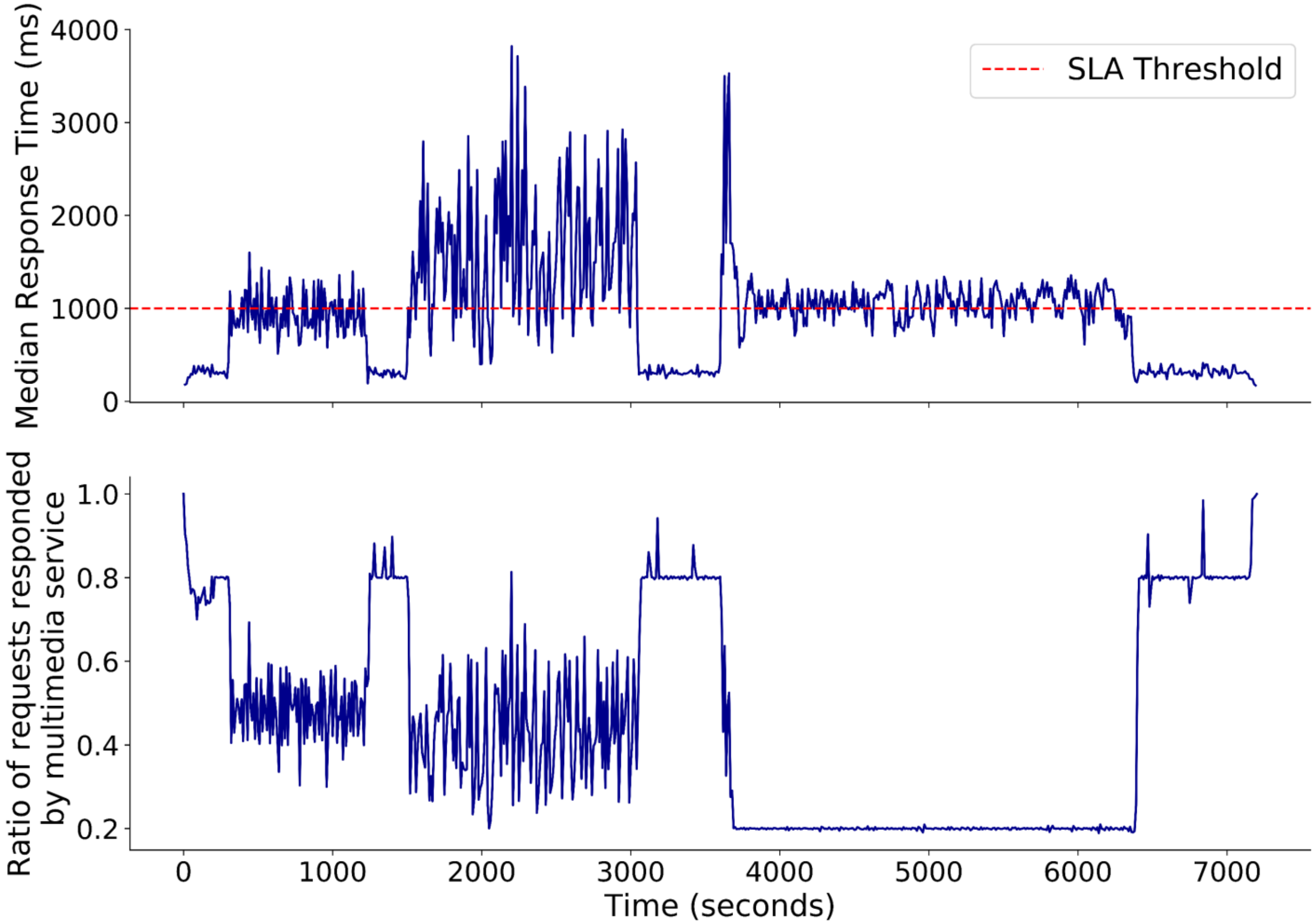


In the TeaStore application's adaptive experiment, the response time is maintained close to the threshold

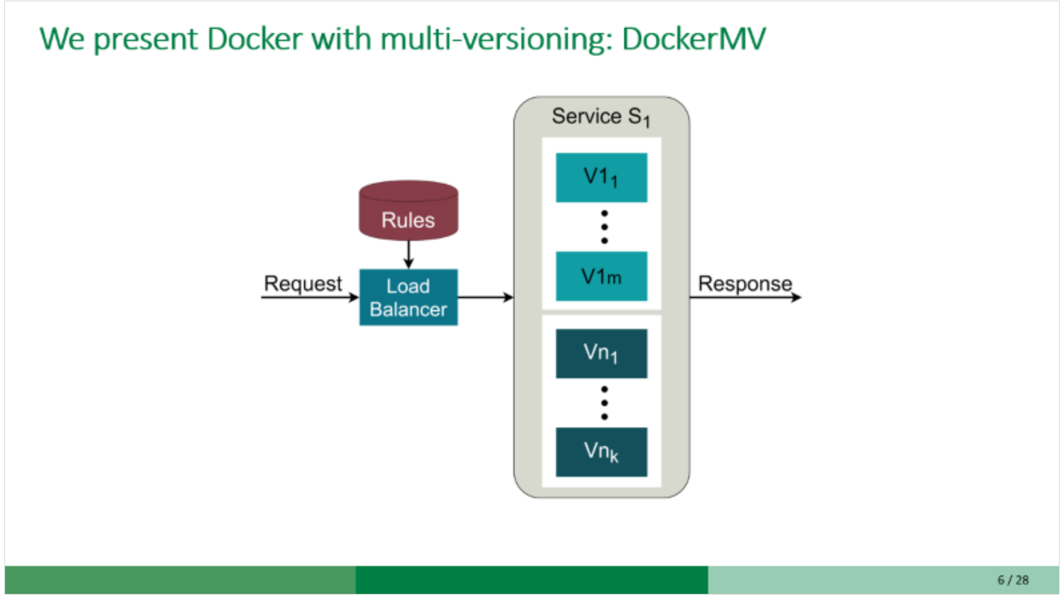
Lightweight Recommender (with single training)
Heavyweight Recommender (with multiple training)
Adaptive distribution
Threshold of 450 ms
Median Response Time (ms)
700
650
600
550
500
450
400
350
300
0
200
400
600
800
1000
Time (seconds)
Adaptive exp.

16 / 28

In the Znn application’s adaptive experiment, the response time is maintained around the threshold

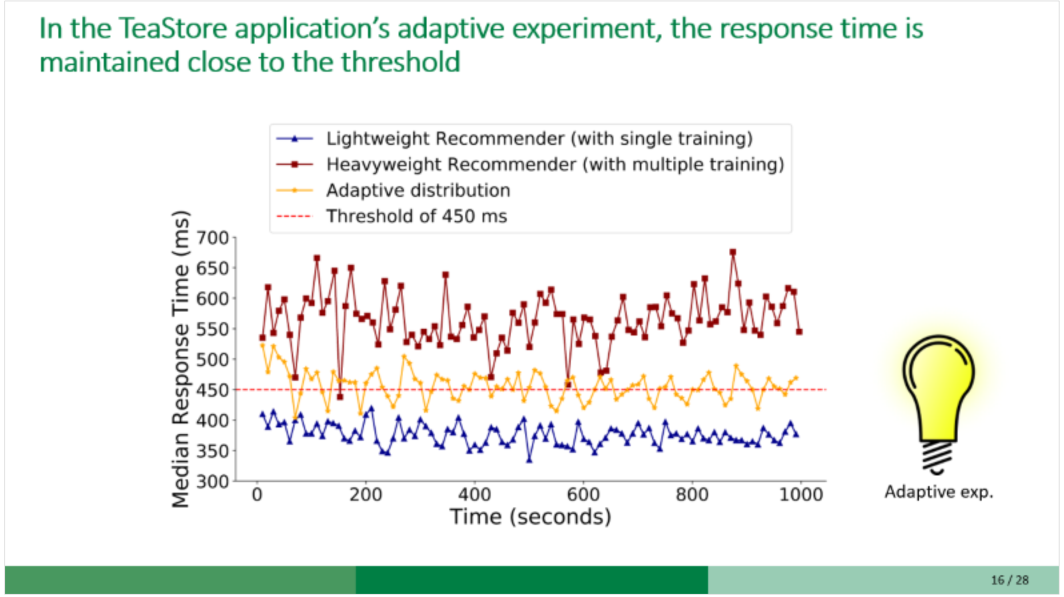


Adaptive exp.



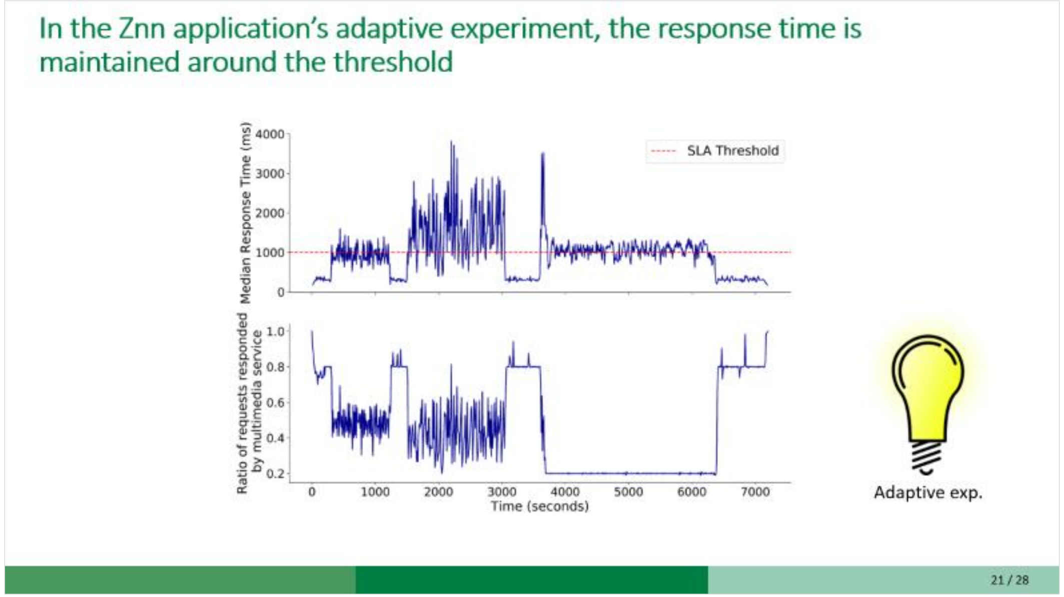
We present Docker with multi-versioning: DockerMV

6 / 28



In the TeaStore application's adaptive experiment, the response time is maintained close to the threshold

16 / 28



In the Znn application's adaptive experiment, the response time is maintained around the threshold

21 / 28

Sara Gholami
sgholami @ualberta.ca

Format of rules for the load balancer

```
$METRIC $OPERATOR $THRESHOLD,
        (version $VERSION_NAME perc =$PERCENTAGE;)+
```

For example:

```
RT > 0.4,
        version recommender:HeavyWeight perc=40;
        version recommender:LightWeight perc=60;
```

How to run DockerMV

docker service create [$OPTIONS]
$\$IMAGE_1$ $\$REPLICATION_1$
…
$\$IMAGE_n$ $\$REPLICATION_n$

For example,

docker service create
e REGISTRY_HOST=host_ip **e** REGISTRY_PORT=1000
10.2.5.26 Network recommender 8080 rules.txt
sgholami/teastore-recommender:HeavyWeight 1
sgholami/teastore-recommender:LightWeight 1