A Framework for Satisfying the Performance Requirements of Containerized Software Systems Through Multi-Versioning
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
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)
100 users sending HTTP requests for 1,000 seconds
(Aprahost 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.

Diagram:
- **Heavyweight Recommender (with multiple training)**
- **Threshold of 450 ms**

Median Response Time (ms) vs. Time (seconds)
In the TeaStore application's worst case experiment, the response time falls below the threshold under the load.
In the TeaStore application’s adaptive experiment, the response time is maintained close to the threshold.
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

The diagram shows the number of active users over time. Initially, there are 74 requests per second. This is followed by a period of 86 requests per second. Finally, the workload increases to 127 requests per second.
In the Znn application’s ideal case experiment, the response time exceeds the threshold under the load.
In the Znn application's worst case experiment, the response time falls below the threshold under the load.
In the Znn application’s adaptive experiment, the response time is maintained around the threshold.
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
In the TeaStore application’s adaptive experiment, the response time is close to the threshold.
We present Docker with multi-versioning: DockerMV

In the TeaStore application’s adaptive experiment, the response time is maintained close to the threshold
In the Znn application’s adaptive experiment, the response time is maintained around the threshold.
We present Docker with multi-versioning: DockerMV

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

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