Experimental Evaluation of the Cloud-Native Application Design

Sandro Brunner, Martin Blöchlinger, Giovanni Toffetti, Josef Spillner, Thomas Michael Bohnert
<josef.spillner@zhaw.ch>

Service Prototyping Lab (blog.zhaw.ch/icclab)
Zurich University of Applied Sciences, Switzerland

December 7, 2015 | 4th CloudAM, Limassol, Cyprus

Cloud-Native Apps: Significant Trend!

LINUX FOUNDATION COLLABORATIVE PROJECTS

CLOUD NATIVE
COMPUTING FOUNDATION

James Governor's Monkchips

An industry analyst blog looking at software ecosystems and convergence

Cloud Native is Nice and All, but How Do We Get There?

And you may ask yourself
house?
self

Cloud-Native Apps at VMware

O'REILLY®

Compliments of Pivotal

Migrating to Cloud-Native Application Architectures

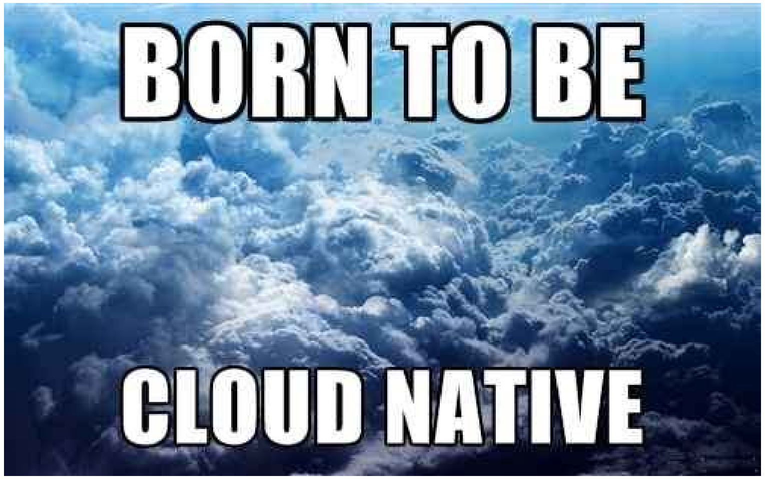
Ryan J Baxter

Using Microservices To Build Cloud Native Applications – Part 1

Cloud-Native Apps: Definition (sort of)

Software applications which

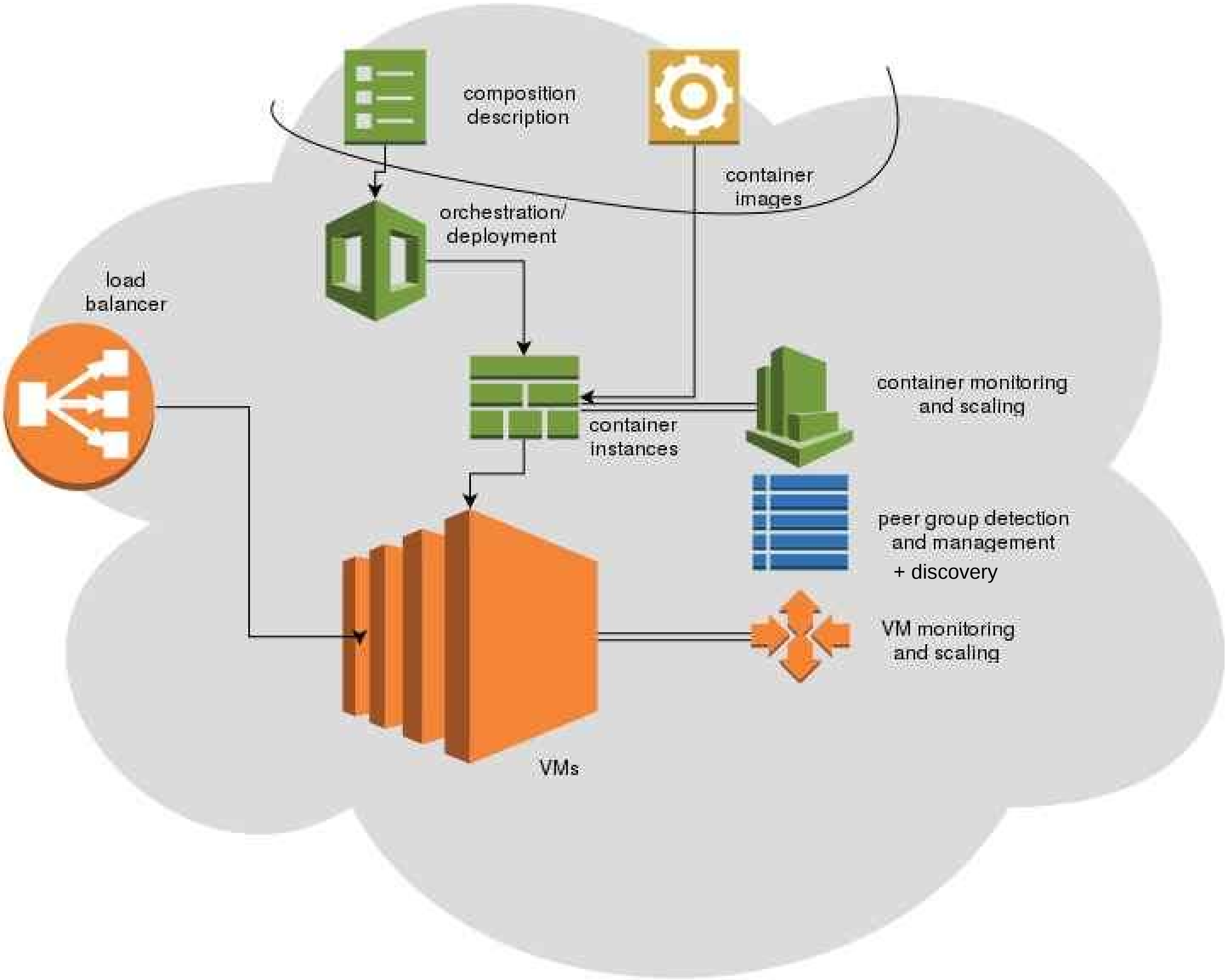
- fully exploit cloud features (APIs, infrastructure, platform, processes)
- are **resilient** against failures
- are elastically **scalable**
- run as services or end-user applications



Implications

- design: fully redundant microservices, fully/partially redundant data
- technology: rapidly manageable units → containers

Cloud-Native Apps: Generic Design



Research Questions & Method

CNA are scalable → Does it scale?

CNA are resilient → Does it self-heal?

How to find out:

- Using a typical business application: Zurmo CRM
  - customer relationship management
  - 3-tier architecture: web frontend, PHP backend, MySQL datastore

Prepare telephone directory for the company v2

☑ Gather the list of employees with there contact details

□ Enter the data into excel

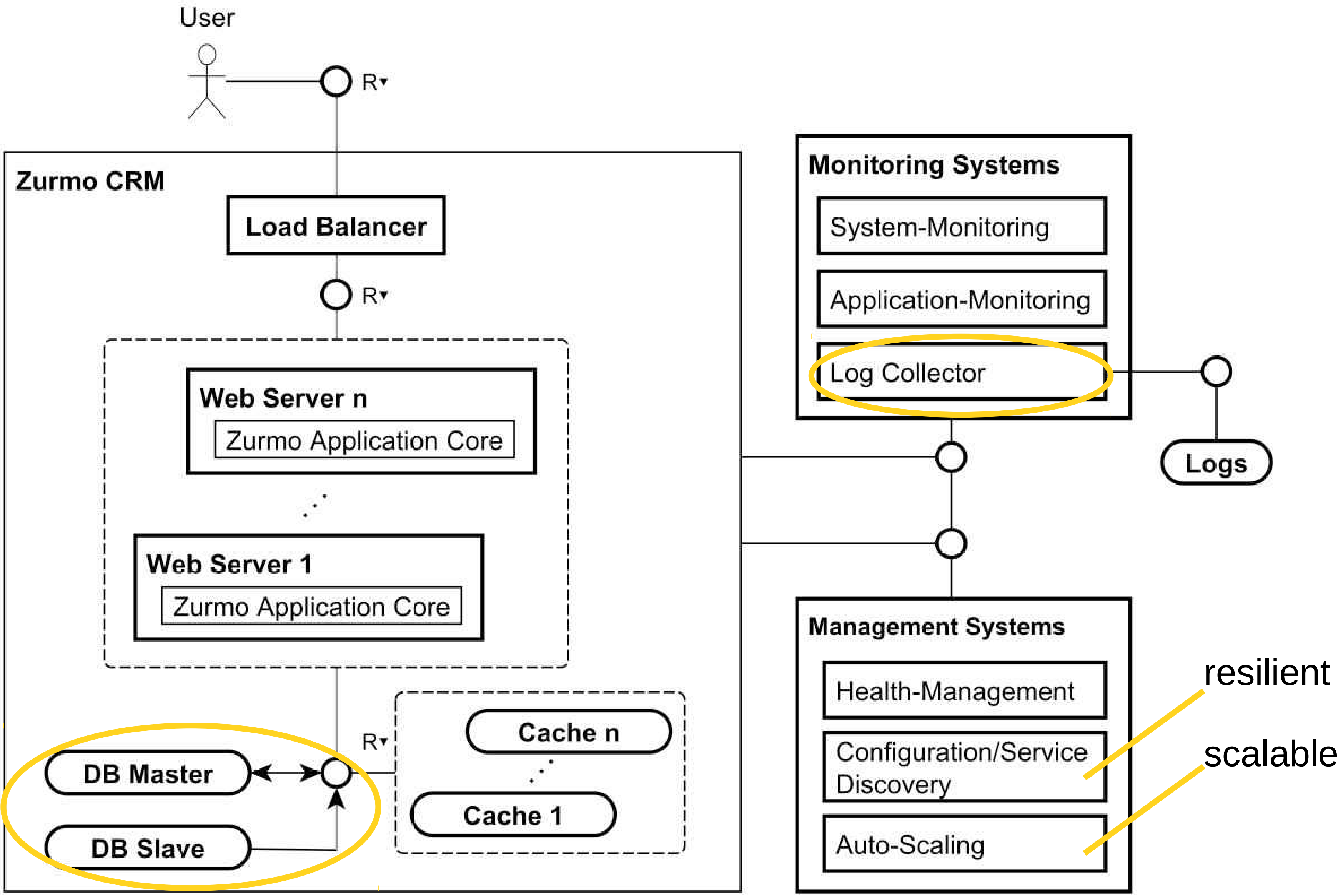
Status: Rejected

Owner: Jim Smith

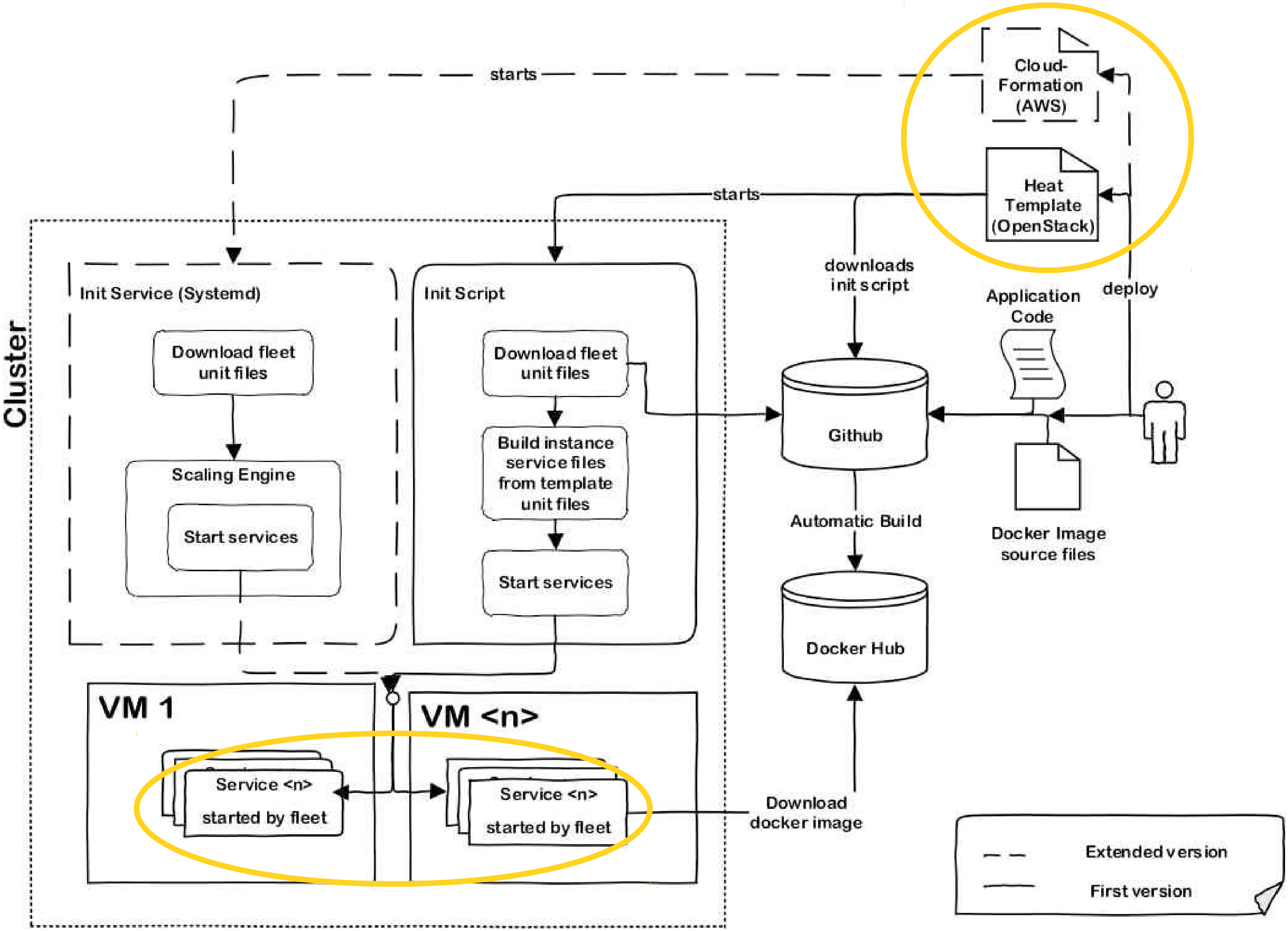
Requested By User: Mary Smith

Get an accounting software v2

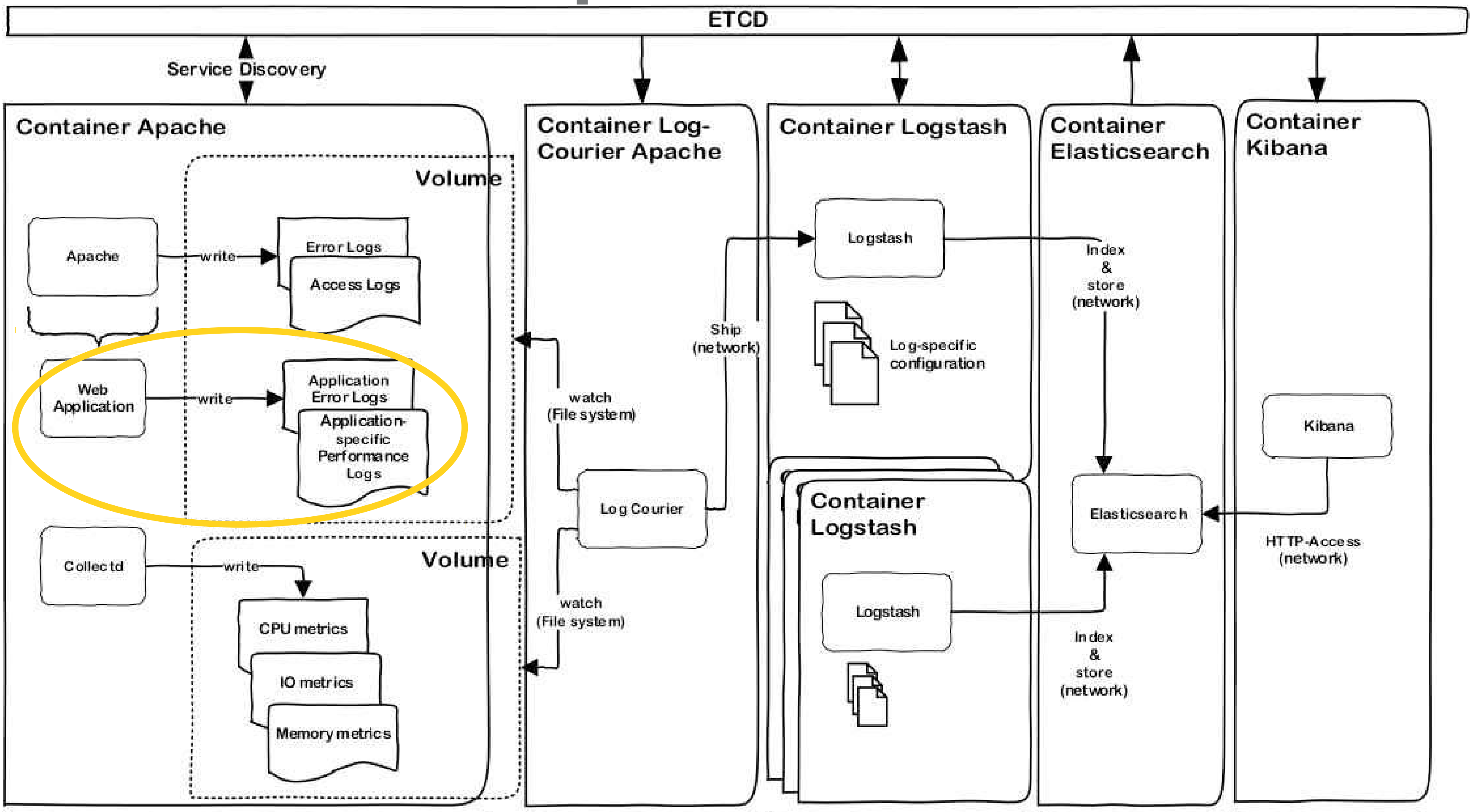
Experiment Architecture



Orchestrated Containers Setup



Containers in Operation



- Just log to files
- Provide Log-Files as Volumes

- Annotate Log-Files with tags
- Ship to Logstash via lumberjack protocol

- Format Log-Entries nicely
- Aggregate statistics
- Write Log-Events to
  - Elasticsearch
  - etcd

- Store Log-Data
- Index Log-Data

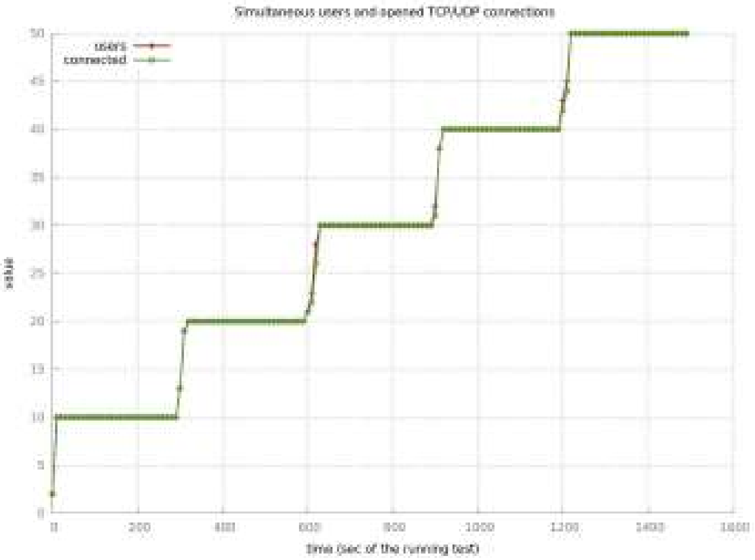
- Visualize
- System metrics
- Log-Data

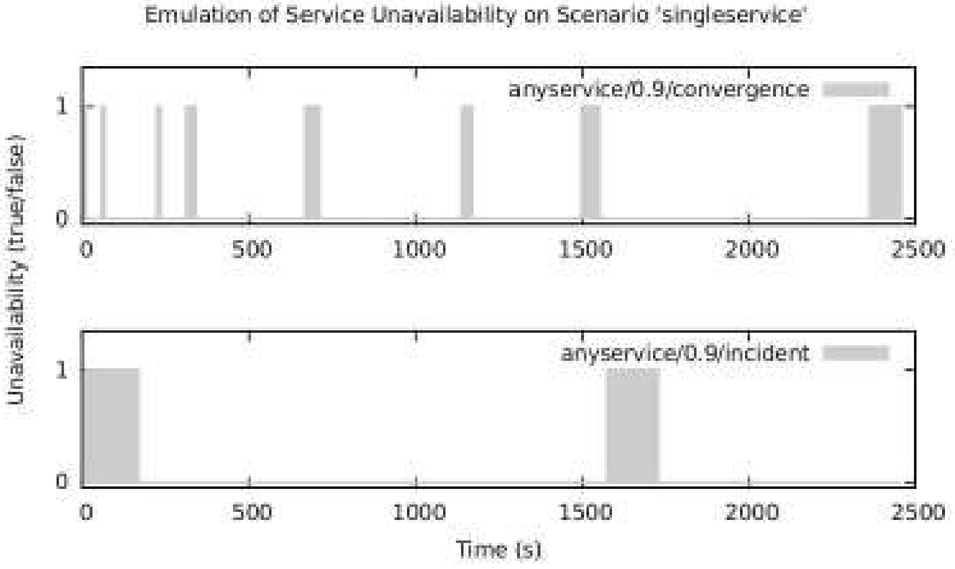
Conducting the Experiment

Tools

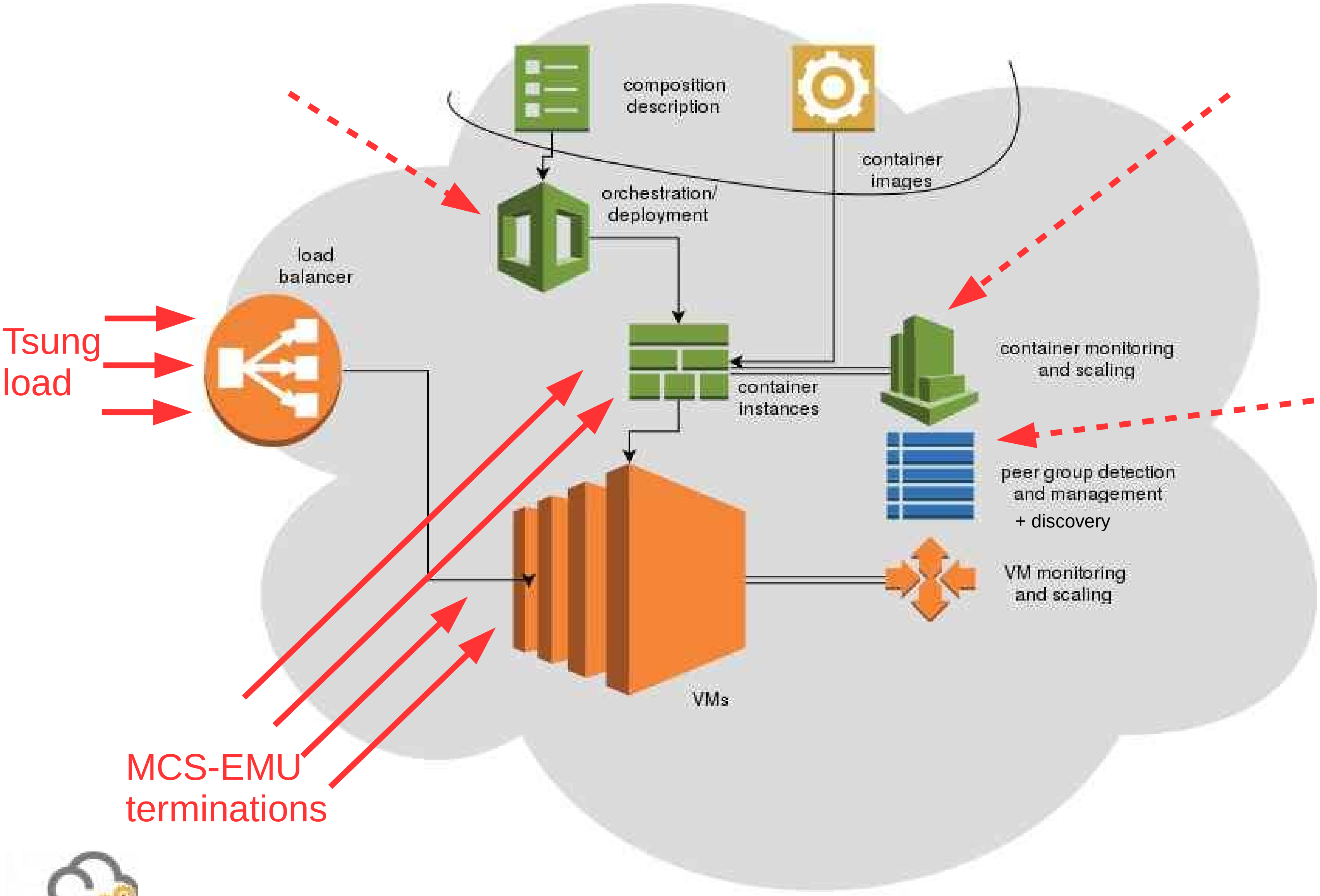
- Tsung user load generator (to provoke scalability)
  - performs web navigation randomly
- MCS-EMU: multi-cloud unavailability emulator (to provoke resilience)
  - terminates Docker containers and VMs randomly, cf. ChaosMonkey, but with multiple (un)availability models

Input functions: load, unavailability + configuration (3-10 VMs)





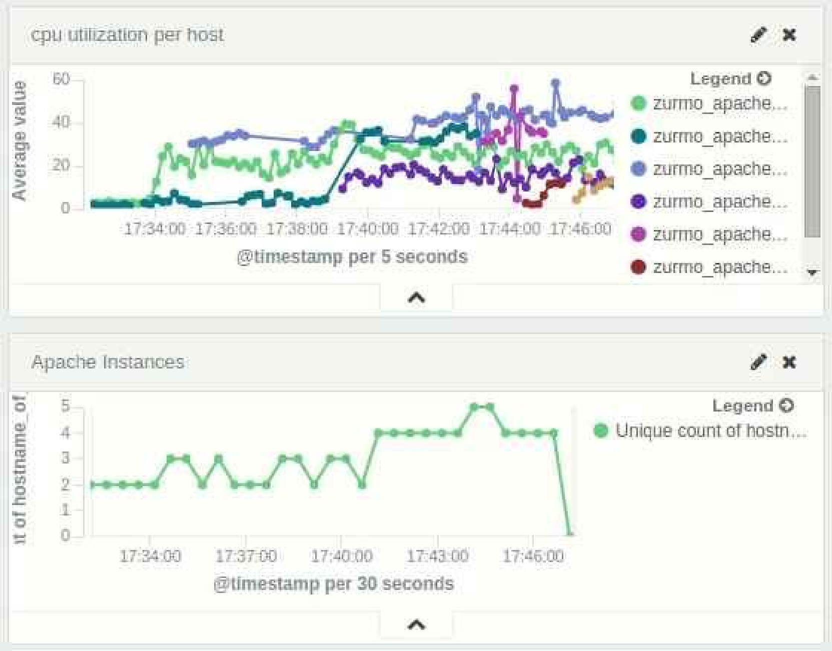
Conducting the Experiment



Tsung
load

MCS-EMU
terminations

Observations

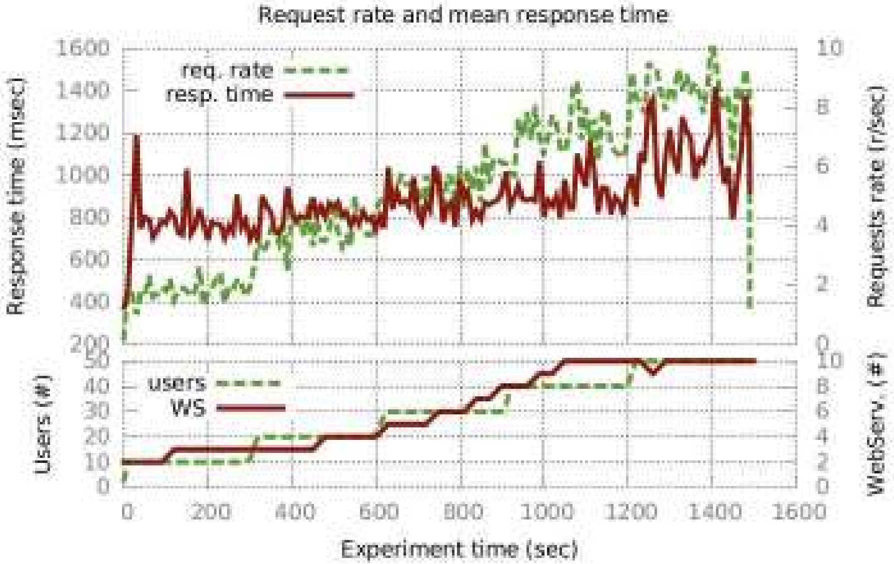


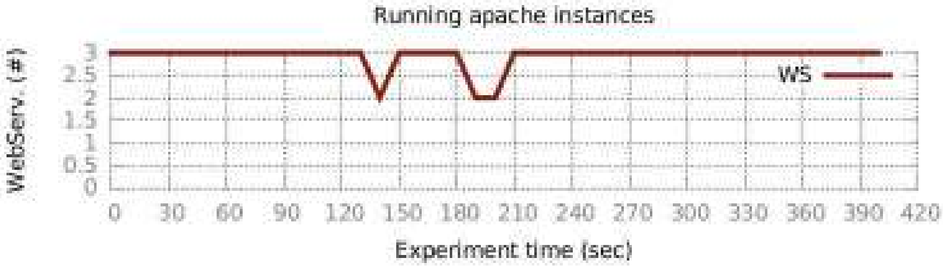
Output function assessment

- Tsung trace file
- Kibana dashboard views
- Zurmo application behaviour
- internal states: etcd, AWS dashboard, logs etc.

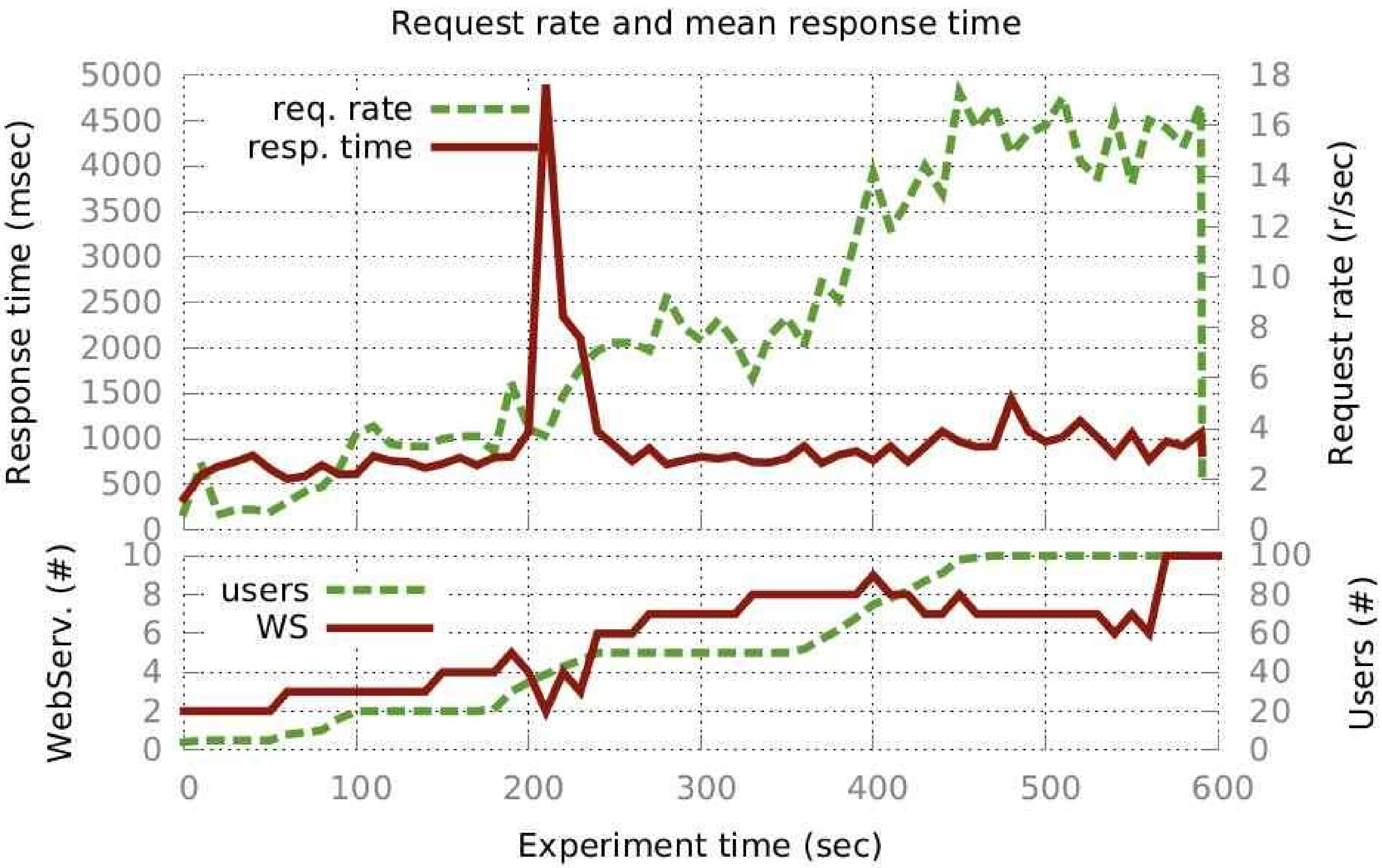
Comparison with desired behaviour

- response times should remain +/- stable no matter what (for 3 VMs)





Observations with more (10) VMs



Findings (incl. delta to paper)

Answers to Research Questions

1. Does it scale?

   → Yes, but:

   - question of trigger metrics: external vs. application-internal
   - still some startup overhead with containers

2. Does it self-heal?

   → Yes, but:

   - tooling itself not resilient, random termination affects experiments
   - deficiencies in standard software, e.g. MySQL clustering init
   - container managers -- fleet in our case -- may misbehave, assumption is correct behaviour

Conclusions

Evaluation: CNA design

- is effective & re-usable, if done right
- but: very tricky especially with used tooling
- alternative approaches: Kubernetes looks promising

Re-usable contributions

- Dynamite scaling engine
- Testing tools
- Dockerised scenario application

Code available!

```
https://github.com/icclab/cna-seed-project
```

Video available soon! (3 minutes demo cut)

'Methodology' + Lessons Learnt

Step 1: Use case identification

Step 2: Platform

- CoreOS bug: concurrent pull of containers from public hub
- Fleet bug: sometimes, containers are not scheduled for launch
- Docker bug #471: only partial download → failure cascade
- etcd restriction: cannot kill 3 member nodes → «Monsanto solution»
- etcd bug: no more requests accepted when disk full

Step 3: Architectural changes

- outsourced session handling to cache + database in parallel

Step 4: Monitoring

- new Logstash output adapter which forwards to etcd

Step 5: Autoscaling

- Dynamite instructs Fleet for horizontal scale-out; is itself CNA