redis vs PostgreSQL

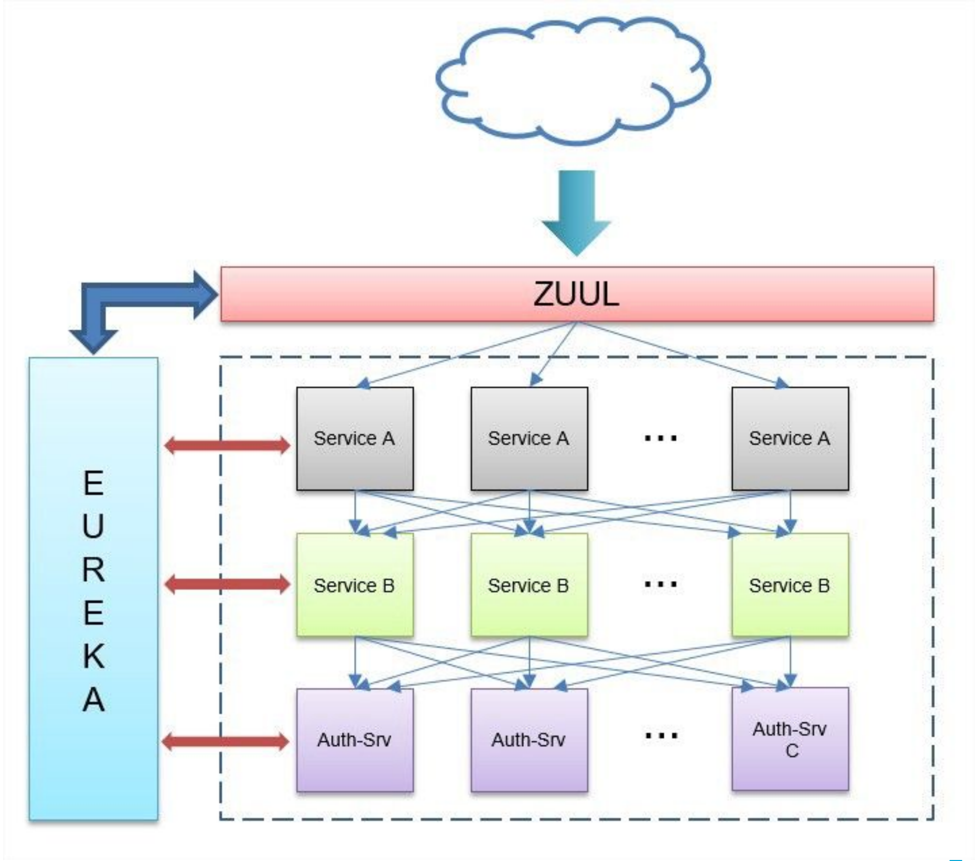
using NETFLIX aws

Group 6

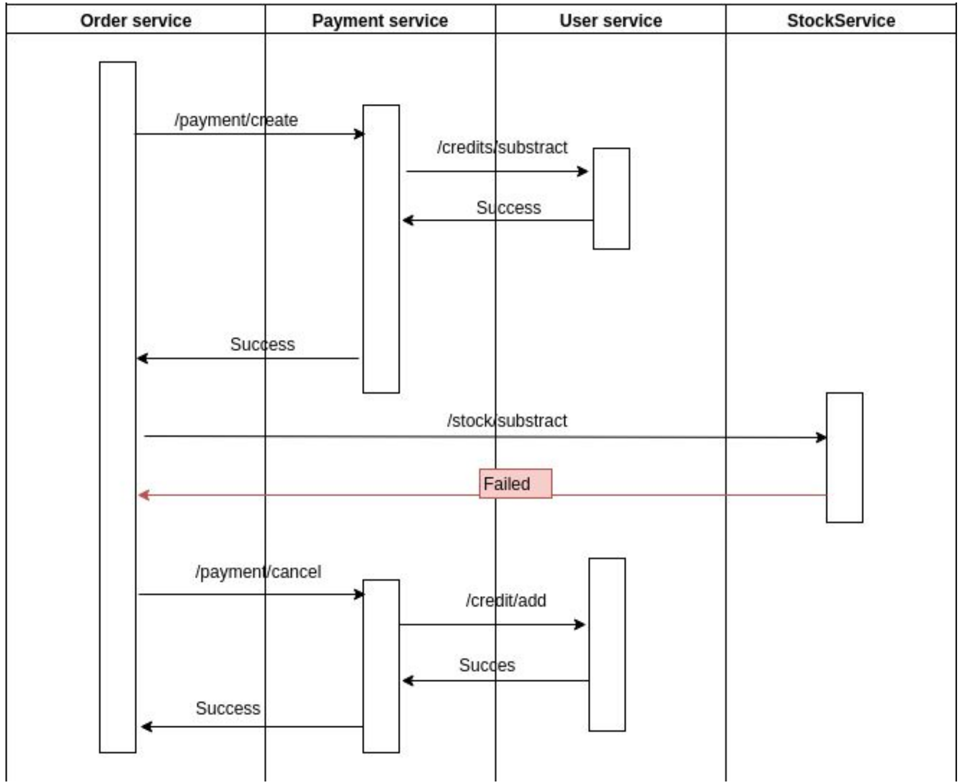
TUDelft

Technologies & Architecture

- Services
  - Spring
  - Sagas
- Postgres
- Redis



Bad weather scenario



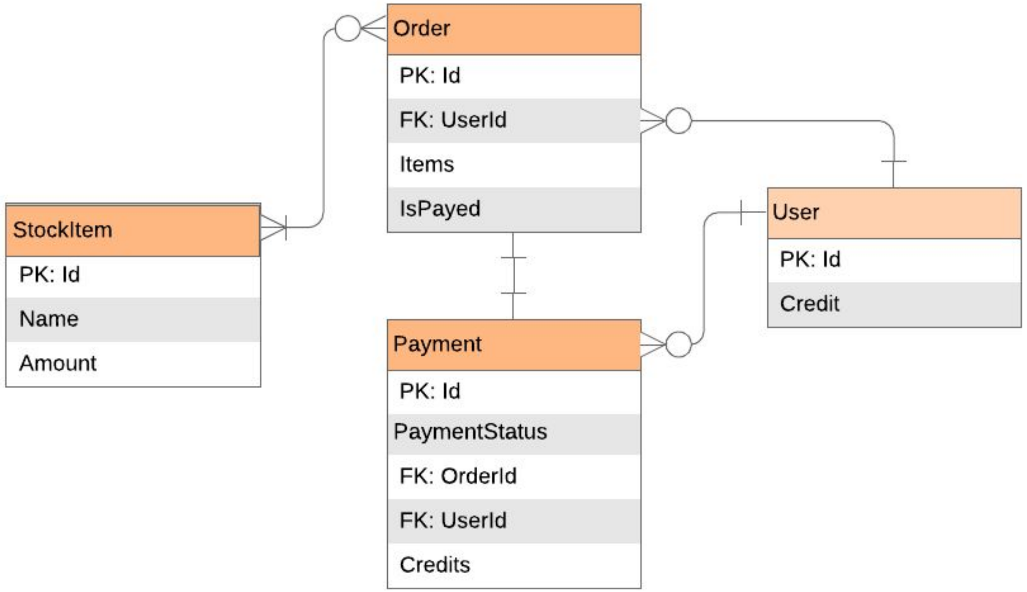
Technologies & Architecture

- Services
- Postgres
    - RDS
- Redis

We used a RESTful controller class to seamlessly switch between redis and postgres

PostgreSQL design choices

- We all know what Postgres is by now
- Simple models, represented in this ERD:



PostgreSQL design choices

- 2 AWS RDS instances
  - db.t2.micro: 1vCPU 1GiB memory
  - db.t3.2xlarge: 8vCPU 32GiB memory

- Use UUIDs instead of numeric IDs
  - Allows us to more easily scale up
  - More costly than a numeric ID

TUDelft

PostgreSQL issues

- Security
  - No security precautions taken anywhere
  - Java string concatenation open to sql injection
- ‘Easy’ get method implementation did not scale well
- Postgres database not in 1st normal form due to multivalued attributes

What is Redis

- In-memory key-value datastore
- NoSQL
- Fast

Hypothesis

- Which DB do we think is faster?
  - Redis
- Which do we think scales more?
  - PostgreSQL
- Which uses more memory?
  - Redis

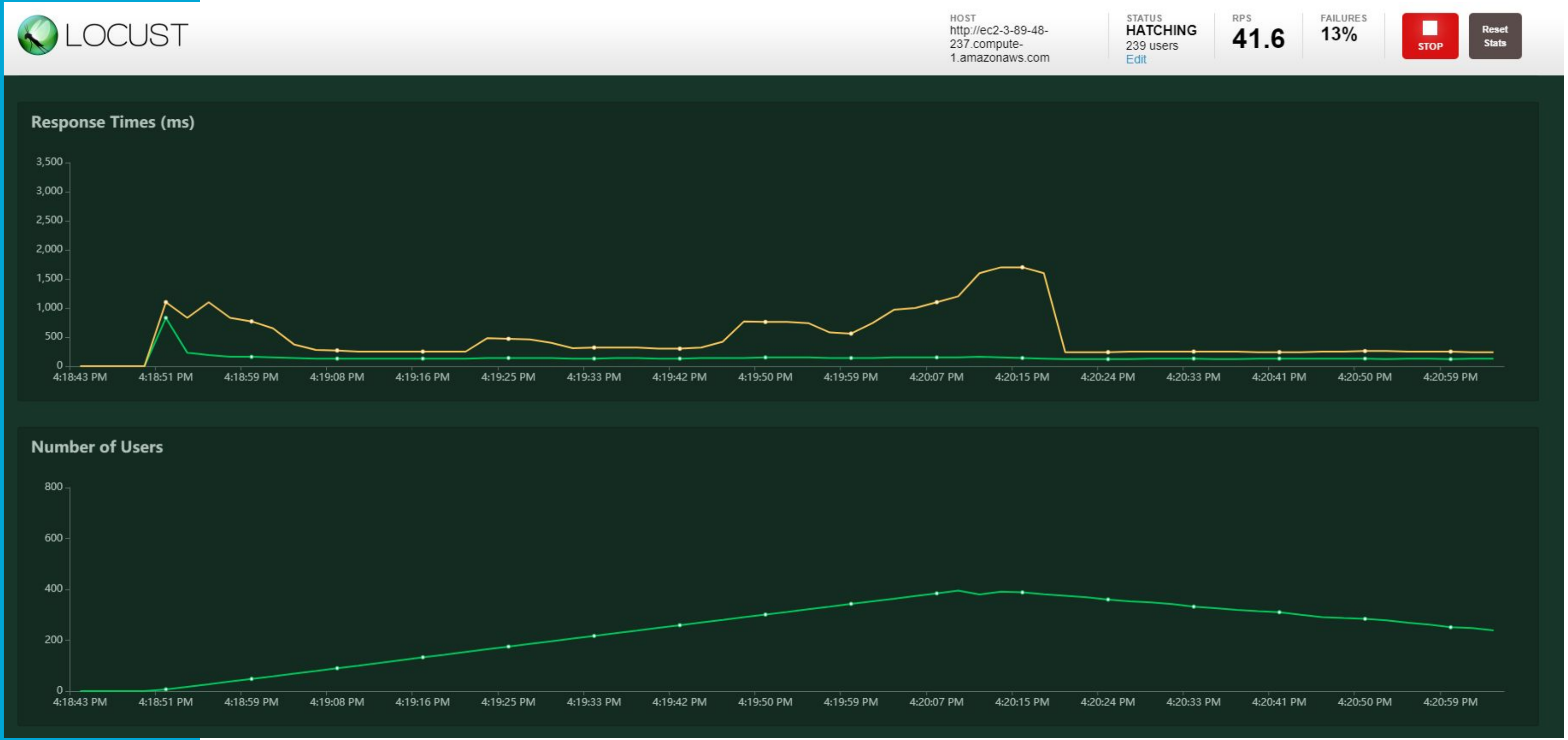
Scalability tests

- Various instance sizes
  - 2 core 512MB
  - 8 core 32GB
- Locust
- Bottlenecks

Locust

- User behaviour was defined to be 'natural' for a webshop
- Main goals: see how many concurrent users the architecture can handle
- Additionally: is the DB the bottleneck?

Medium Redis machine



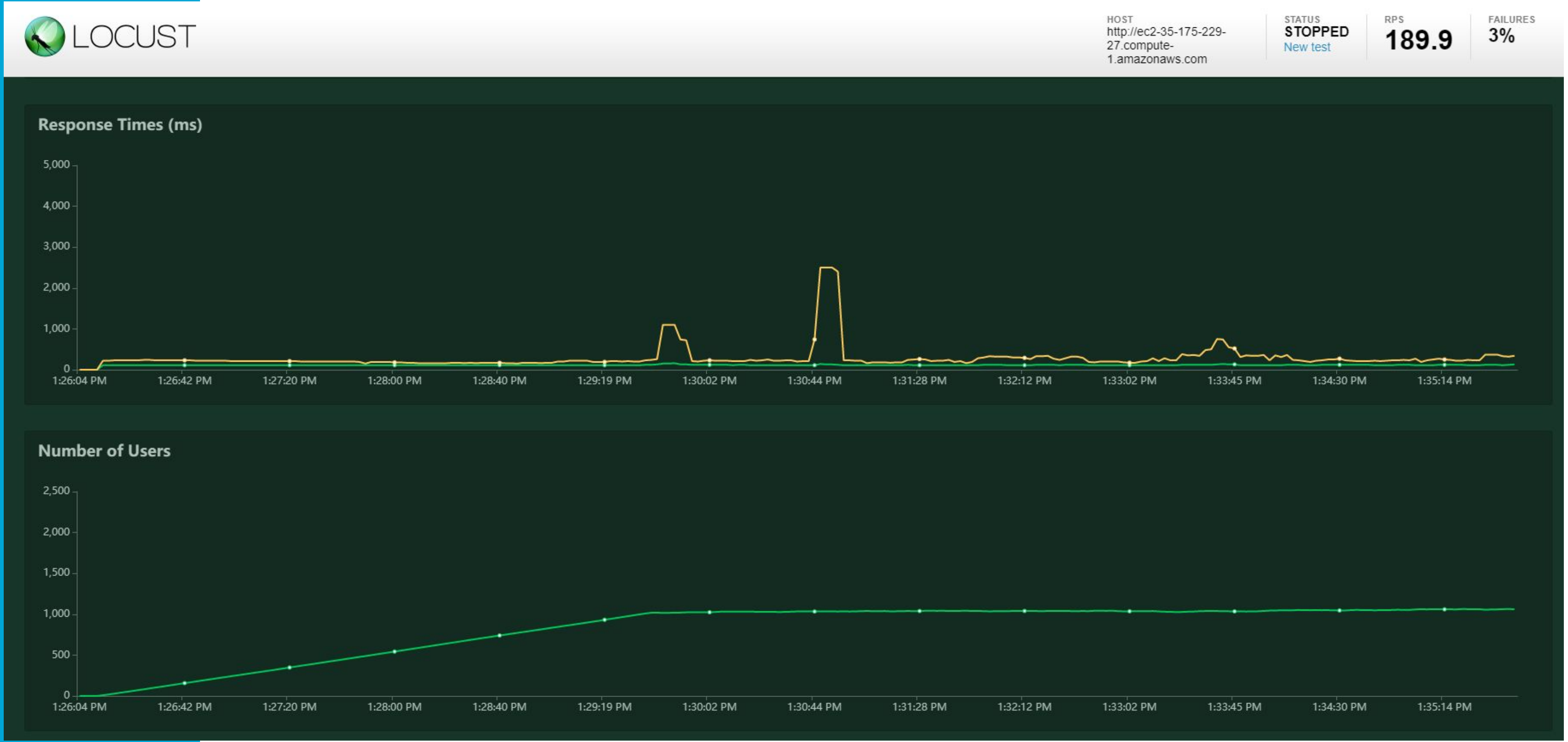
HOST
http://ec2-3-89-48-237.compute-1.amazonaws.com

STATUS
HATCHING
239 users
Edit

RPS
41.6

FAILURES
13%

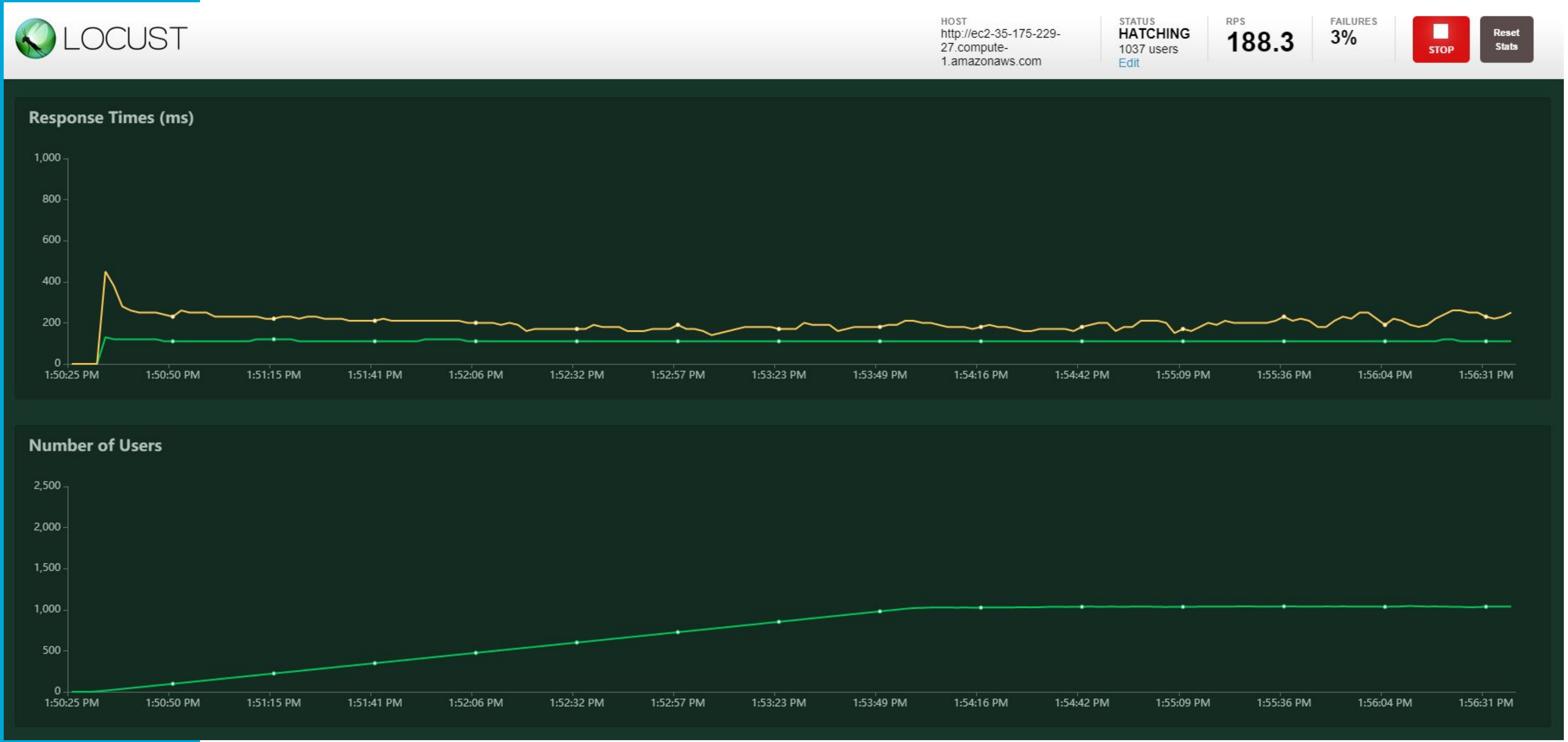
Large redis machine



Cap of 1000 users?

- Perhaps scaling the services to 3 of each service increases the amount of users?

Large redis machine with scaled services



HOST
http://ec2-35-175-229-27.compute-1.amazonaws.com

STATUS
HATCHING
1037 users
Edit

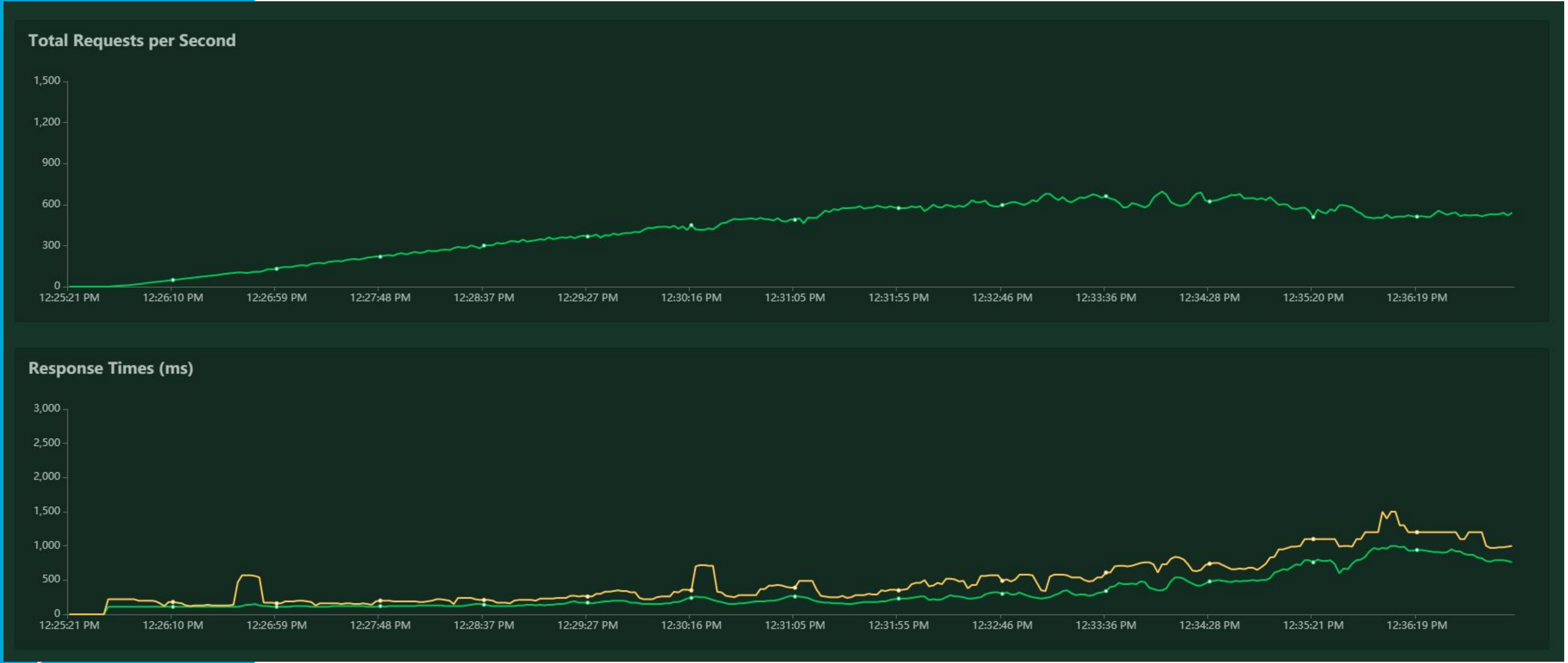
RPS
188.3

FAILURES
3%

Again the same cap...

- Perhaps the Redis database is running out of memory?
- Run just user creation, speed up exponentially

Exponential user creation

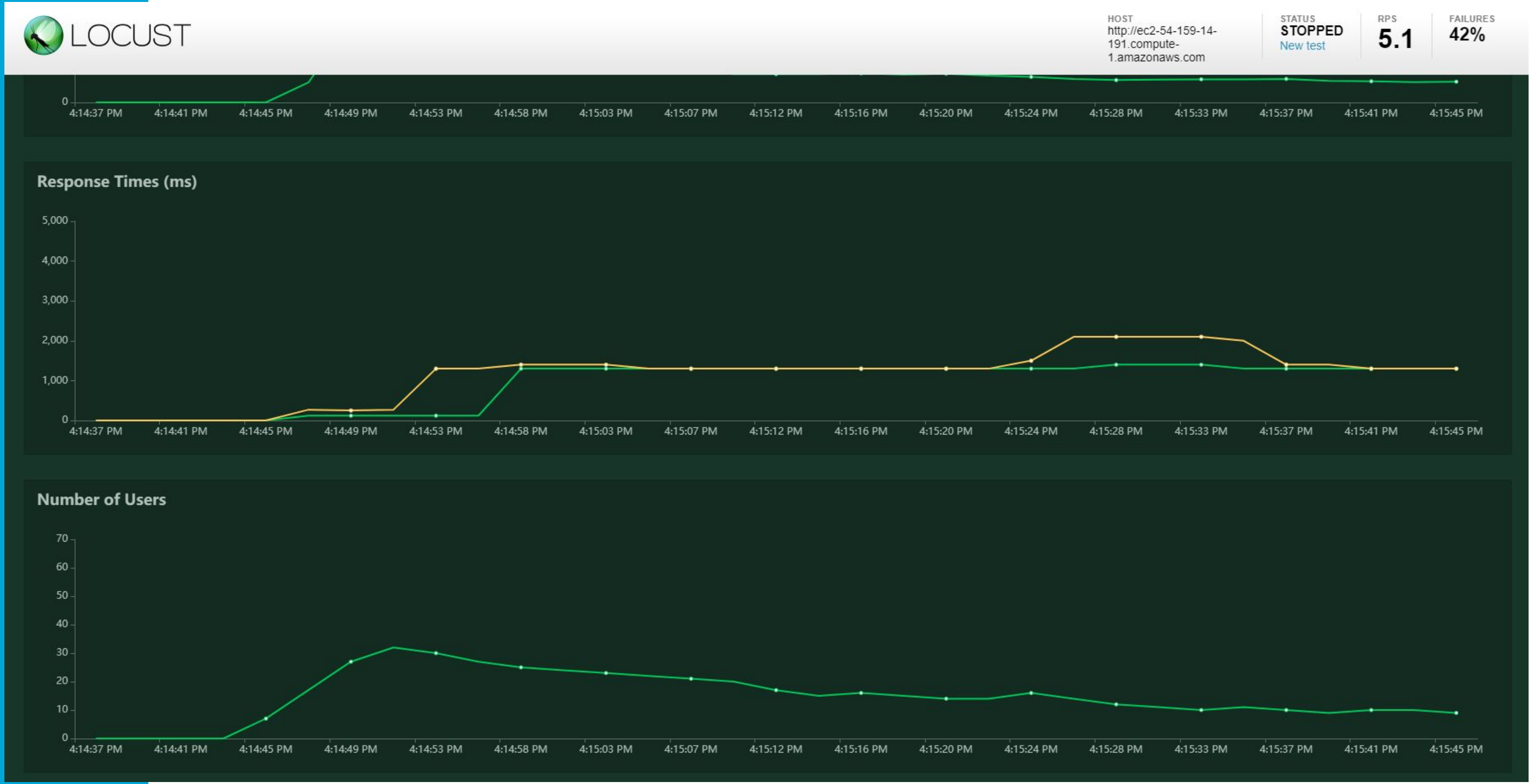


TUDelft

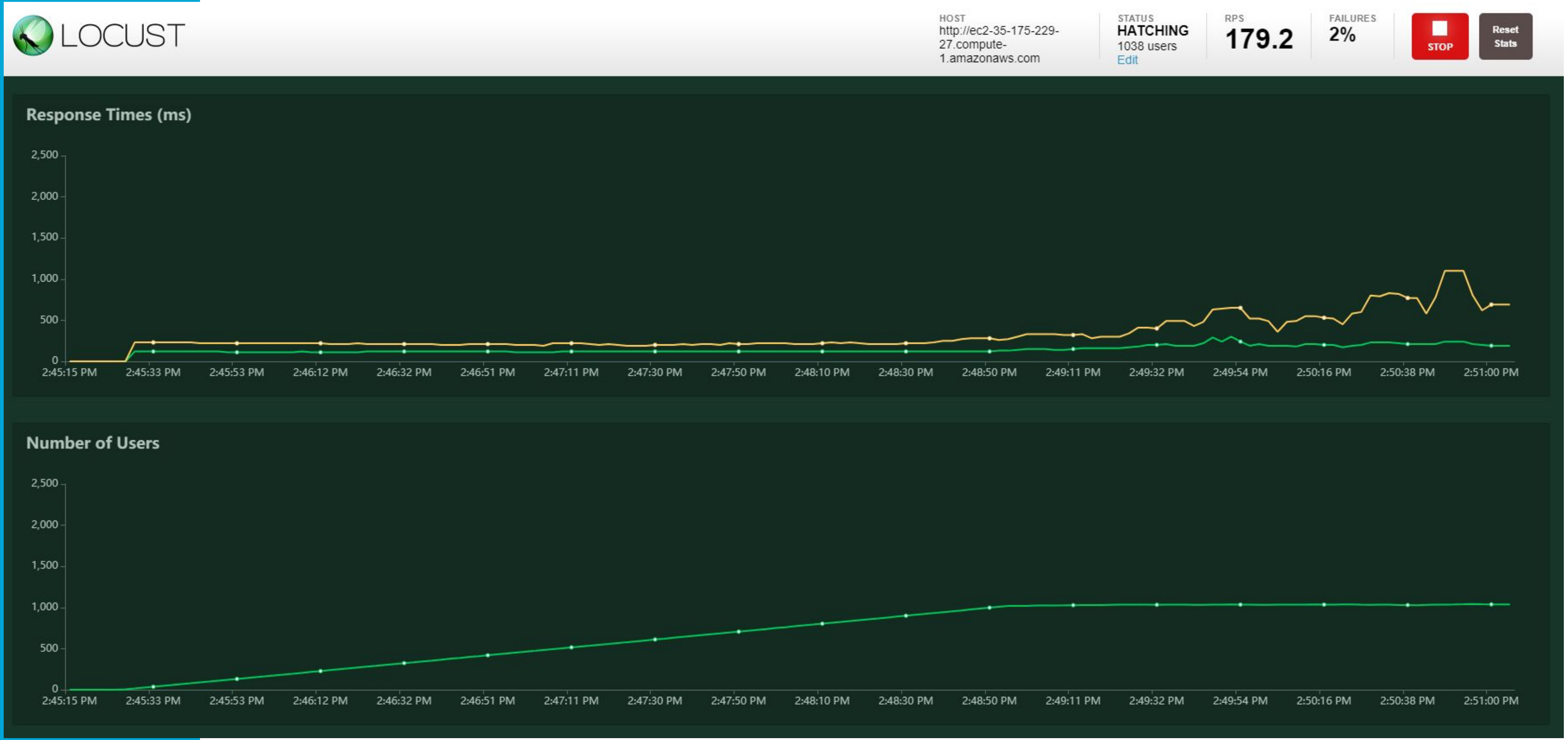
RPS now does increase

- Suspicion that the amount of concurrent connections is the problem
- Redis was able to handle storing thousands and thousands of users just fine
- What about postgres?

Medium Postgres machine



Large Postgres machine



TUDelft

Suspiciously the same user cap...

- (We learned on monday) perhaps we should have tried more locust slaves
- However, main point of failure ‘too many open files’
- Server configuration, ‘ulimit’

Main issues encountered

- Problem setting up a cluster, could have probably fixed the ulimit problem
- Deploying and configuring on AWS gave us more problems than expected
- First focused on Redis, Postgres configuration with RDS slowed us down

Weak spots

- Postgres: all services share same instance
- Not secure at all
- No cluster

Lessons learned

- Deployment and scaling takes time
- Redis is easy to get started with