

# Load Testing with Locust

---

Kubilay Kahveci  
Full Stack Software Engineer

@mkubilayk  
mkahveci@bloomberg.net

W My Security Worksheet: 857 IBEX heatmap

File Analytics Edit Format Formulas Debug Non UX

View Library Send Refresh Export to Excel

Create Share Rename Favorite

Ticker		Last Price (1 Year)	Net	%1D	%1M	%3M	Chg 6M Pct	%1M	Volume	Bid	Ask	BEst P/E
IBEX	d	10229.20	+99.60	+0.98%	-0.52%	-6.82%	+2.21%	-0.52%	17941			14.315
Communications (2)												
TEF SM	d	9.008	+0.051	+0.57%	-2.03%	-10.46%	-12.92%	-2.03%	541996	9.003	9.008	10.598
TL5 SM	d	9.505	+0.067	+0.71%	-9.65%	-15.96%	-16.08%	-9.65%	81735	9.505	9.514	
Consumer Discretionary (3)												
IAG SM	d	6.565	+0.04	+0.61%	-1.29%	-1.52%	+1.39%	-1.29%	205801	6.564	6.566	
ITX SM	d	31.33	+0.435	+1.41%	-6.74%	-13.94%	-0.33%	-6.74%	316690	31.32	31.335	26.939
MEL SM	d	12.335	+0.135	+1.11%	-2.53%	-9.00%	-1.48%	-2.53%	15716	12.33	12.355	
Consumer Staples (1)												
DIA SM	d	5.265	+0.027	+0.52%	-2.97%	-5.17%	+2.70%	-2.97%	70504	5.266	5.272	
Energy (2)												
REP SM	d	14.66	+0.12	+0.83%	+4.38%	-1.15%	+1.70%	+4.38%	164958	14.66	14.67	10.456
SGRE SM	d	12.585	+0.215	+1.74%	-4.15%	-38.07%	-30.40%	-4.15%	207028	12.575	12.59	11.115
Financials (9)												
BBVA SM	d	7.251	+0.077	+1.07%	-4.49%	-2.78%	+5.18%	-4.49%	774105	7.253	7.258	11.327
BKIA SM	d	3.918	+0.039	+1.01%	-5.25%	-7.66%	-4.35%	-5.25%	134946	3.913	3.916	15.672
BKT SM	d	7.777	+0.079	+1.03%	-3.76%	-6.24%	+3.75%	-3.76%	131463	7.774	7.778	14.398
CABK SM	d	4.21	+0.044	+1.06%	-2.55%	+0.41%	+11.83%	-2.55%	672567	4.207	4.21	11.529
COL SM	d	8.37	+0.092	+1.11%	+4.38%	+11.60%	+23.31%	+4.38%	49310	8.366	8.373	
MAP SM	d	2.876	+0.051	+1.81%	-6.01%	-10.63%	-5.49%	-6.01%	253514	2.875	2.877	10.601
MRL SM	d	11.63	+0.06	+0.52%	+1.79%	+0.87%	+12.12%	+1.79%	77863	11.625	11.63	
SAB SM	d	1.733	+0.023	+1.35%	-5.15%	-1.31%	+6.91%	-5.15%	1597249	1.733	1.734	12.379
SAN SM	d	5.427	+0.083	+1.55%	-3.09%	-9.84%	+0.43%	-3.09%	2735641	5.426	5.428	10.429
Health Care (1)												
GRF SM	d	24.485	+0.045	+0.18%	+5.18%	-2.08%	+17.15%	+5.18%	14357	24.475	24.505	22.613
Industrials (7)												
ABE SM	d	17.045	+0.085	+0.50%	+1.76%	+4.35%	+20.16%	+1.76%	226533	17.045	17.05	
ACS SM	d	31.94	+0.29	+0.92%	+3.03%	-11.15%	+5.78%	+3.03%	26542	31.915	31.95	
AENA SM	d	159.60	+1.15	+0.73%	-2.89%	-13.12%	+17.77%	-2.89%	6872	159.50	159.60	
ANA SM	d	72.13	+0.51	+0.71%	+0.92%	-15.41%	-0.79%	+0.92%	1789	72.01	72.13	10.657
CLNX SM	d	19.465	+0.275	+1.43%	+4.68%	+2.64%	+27.67%	+4.68%	72283	19.455	19.47	
FER SM	d	19.39	+0.135	+0.70%	+6.22%	-2.64%	+8.15%	+6.22%	30866	19.365	19.38	
TRE SM	d	28.695	--	--	-1.05%	-14.90%	-22.88%	-1.05%	18480	28.695	28.71	9.905
Materials (3)												
ACX SM	d	12.125	+0.045	+0.37%	+13.21%	-0.41%	-7.56%	+13.21%	46590	12.125	12.14	
MTS SM	d	22.685	+0.415	+1.86%	+3.56%	+19.58%	-4.41%	+3.56%	140997	22.67	22.695	



# What is your capacity?



# Explore system qualities



# CAPACITY

Is the infrastructure  
sized adequately?



## CAPACITY

Is the infrastructure  
sized adequately?



## SPEED

Does the system  
respond quickly  
enough?



## CAPACITY

Is the infrastructure  
sized adequately?



## SPEED

Does the system  
respond quickly  
enough?



## SCALABILITY

Can the system grow  
to handle future  
volumes?





## CAPACITY

Is the infrastructure  
sized adequately?



## SPEED

Does the system  
respond quickly  
enough?



## SCALABILITY

Can the system grow  
to handle future  
volumes?



## STABILITY

Does the system  
behave correctly  
under load?





# How to assess?



# PERFORMANCE TESTING

- Evaluate component performance
  - No heavy load
  - Tuning
- 
- Do not aim to find defects
  - Establish the benchmark behaviour



## PERFORMANCE TESTING

- Evaluate component performance
  - No heavy load
  - Tuning
- 
- Do not aim to find defects
  - Establish the benchmark behaviour



## LOAD TESTING

- Feed the system the largest task it can handle
  - Increase the load
  - Simulate with virtual users
- 
- Expose defects
  - Determine the upper limit for all components



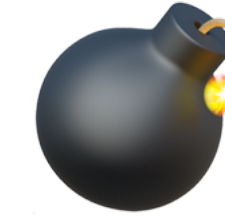
## PERFORMANCE TESTING

- Evaluate component performance
- No heavy load
- Tuning
- Do not aim to find defects
- Establish the benchmark behaviour



## LOAD TESTING

- Feed the system the largest task it can handle
- Increase the load
- Simulate with virtual users
- Expose defects
- Determine the upper limit for all components



## STRESS TESTING

- Attempt to break the system down
- Overwhelm its resources
- Take resources away
- Graceful failure and recovery
- Define application behaviour after failure



# Before you invest

Have monitoring tools

Identify usage patterns

Define success criteria in measurable terms

Isolate the testing environment



Isolate the testing environment





# Locust (locust.io)


Open source

User behaviour in code (Python)

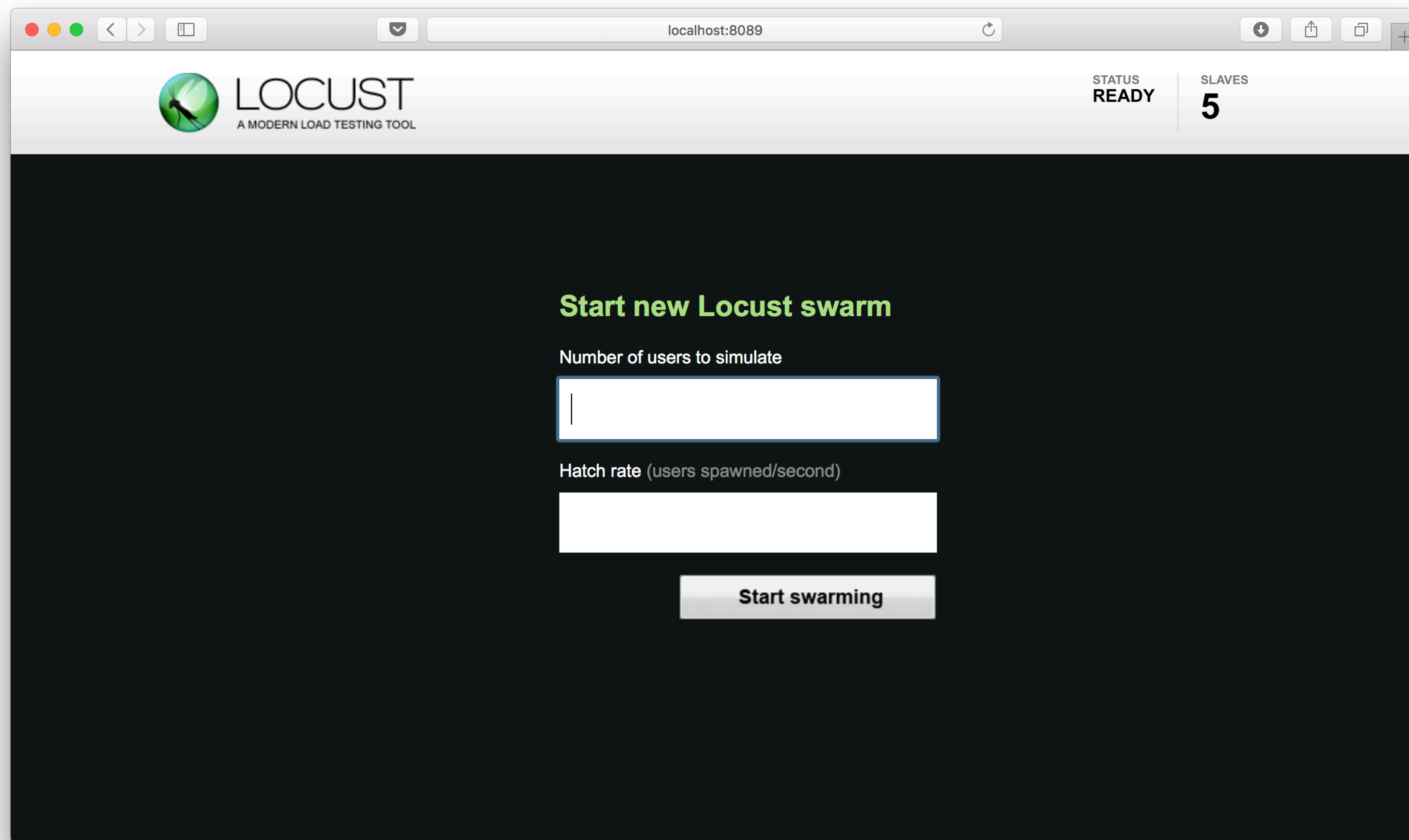
Based on co-routines, async approach

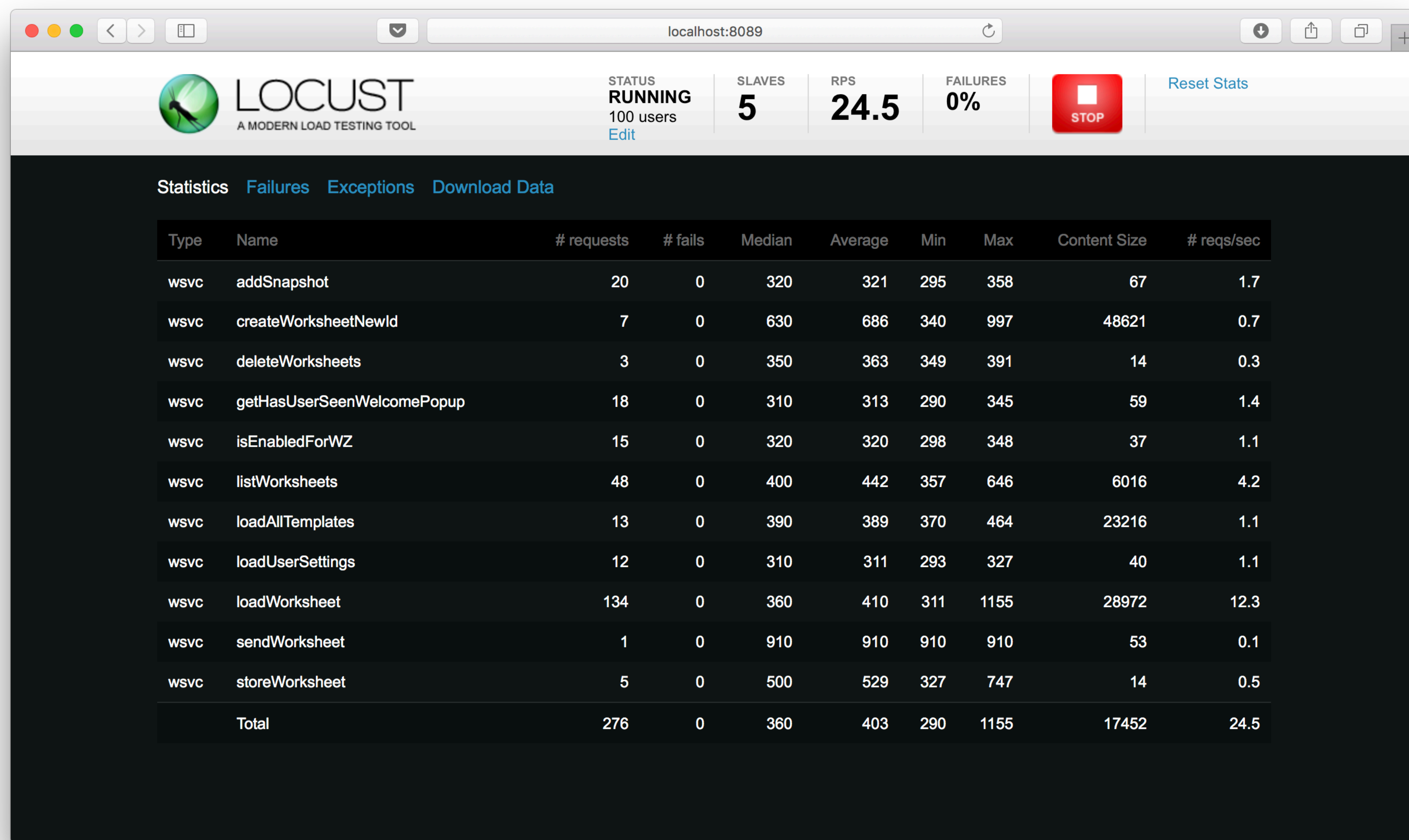
Distributed and scalable

Battle tested



```
1 $ locust --no-web \  
2     -f locustfile.py \  
3     -c 100 \  
4     -r 10 \  
5     -n 1000 \  
6     --print-stats
```





```
1 from locust import HttpLocust, TaskSet, task
2
3 class WebsiteTasks(TaskSet):
4     def on_start(self):
5         self.client.post("/login", {
6             "username": "kubilay",
7             "password": "*****"
8         })
9
10    @task(4)
11    def home(self):
12        self.client.get("/home")
13
14    @task(2)
15    def profile(self):
16        self.client.get("/profile")
17
18 class WebsiteUser(HttpLocust):
19     task_set = WebsiteTasks
20     min_wait = 5000
21     max_wait = 15000
```

```
1 from locust import Locust, events
2 from my_protocol import Client
3 import time
4
5 class CustomClient():
6     def __init__(self):
7         self.client = protocol.Client()
8
9     def sendRequest(self, request):
10         params = {
11             'request_type': request['type'],
12             'name': request['name']
13         }
14         event = None
15         start_time = time.time()
16
17         try:
18             self.client.send(request)
19         except Exception as e:
20             params['exception'] = e
21             event = events.request_failure          # FAILURE
22         else:
23             event = events.request_success          # SUCCESS
24         finally:
25             params['response_time'] = int((time.time() - start_time) * 1000)
26             event.fire(**params)
```





```
1 from locust import Locust
2
3 class CustomLocust(Locust):
4     def __init__(self, *args, **kwargs):
5         super(CustomLocust, self).__init__(*args, **kwargs)
6         self.client = CustomClient()
```



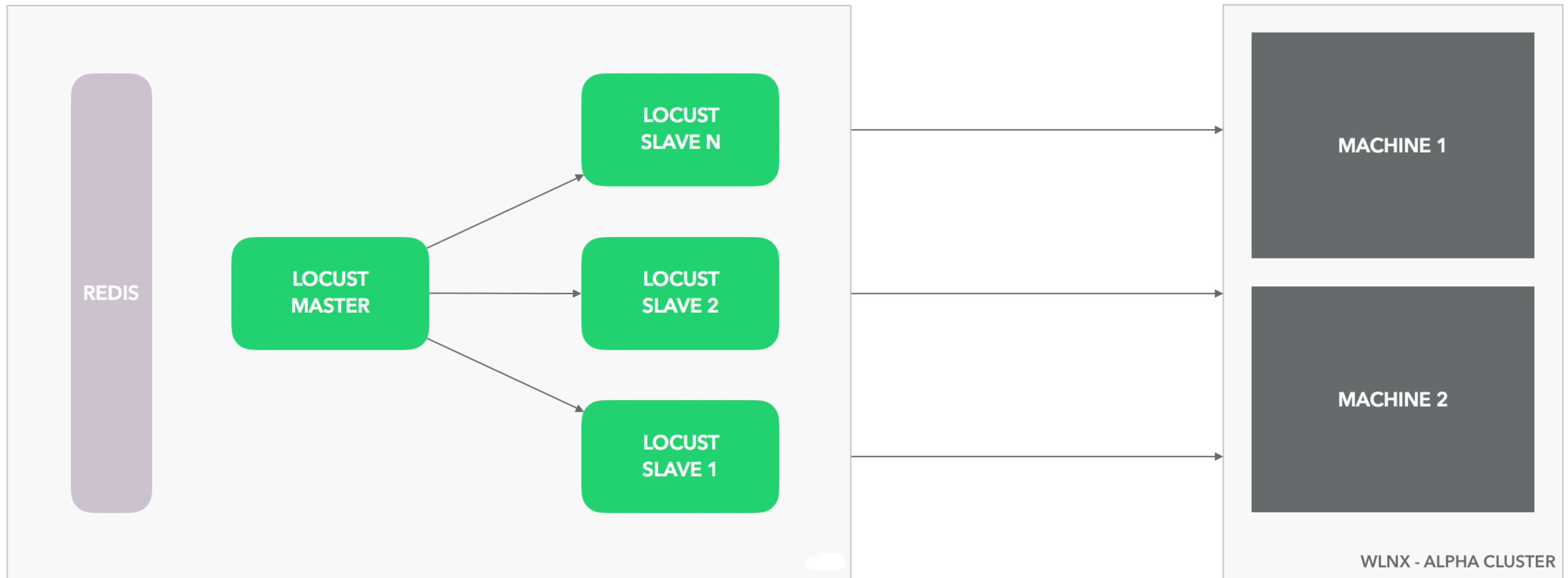
# Containers FTW

Lightweight

Well suited to singular tasks

Simple to deploy

Distributed





# Test runs

Many dropped requests

Single request taking too long, blocking others

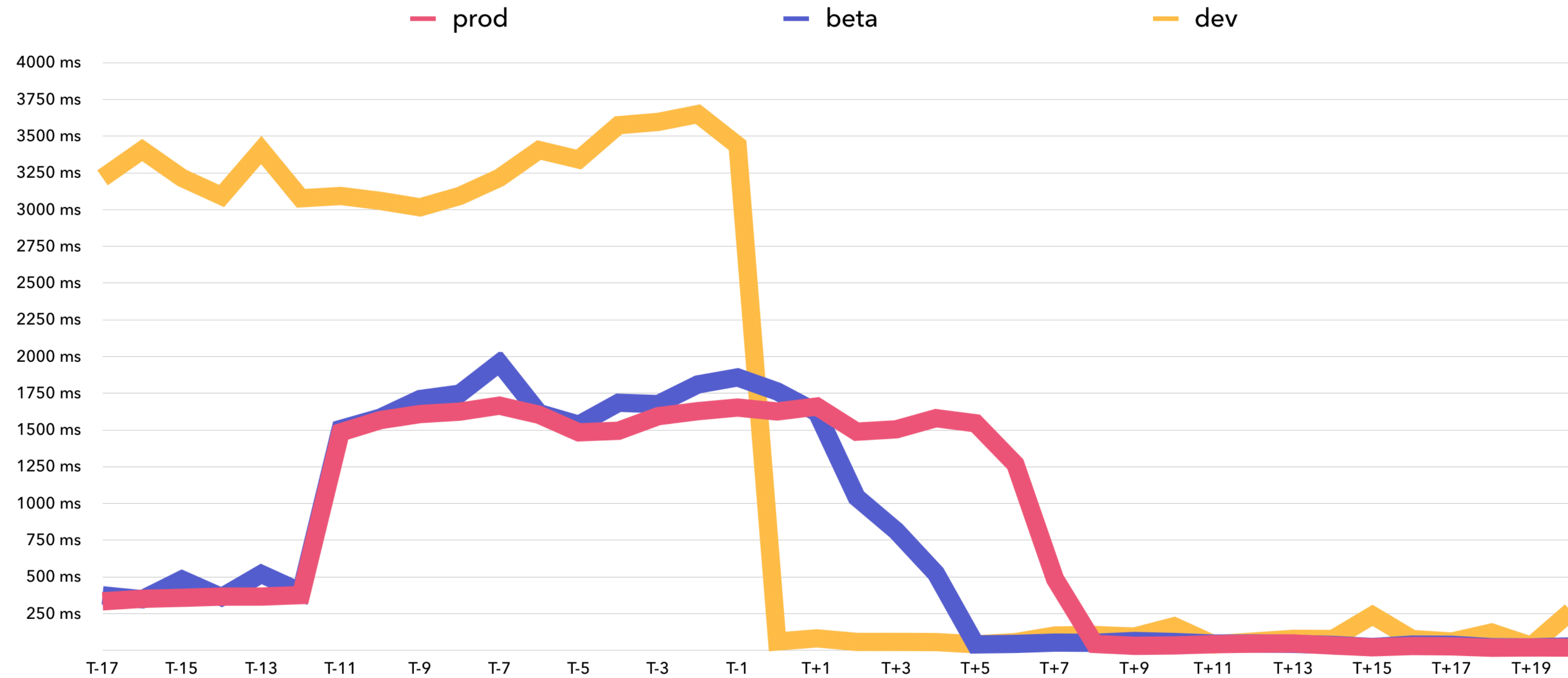
Queues filled up pretty quickly

More instrumentation

Regression in DB access

Fix and ship





Average Response Time



Thank you!

mkahveci@bloomberg.net

@mkubilayk