Performance testing

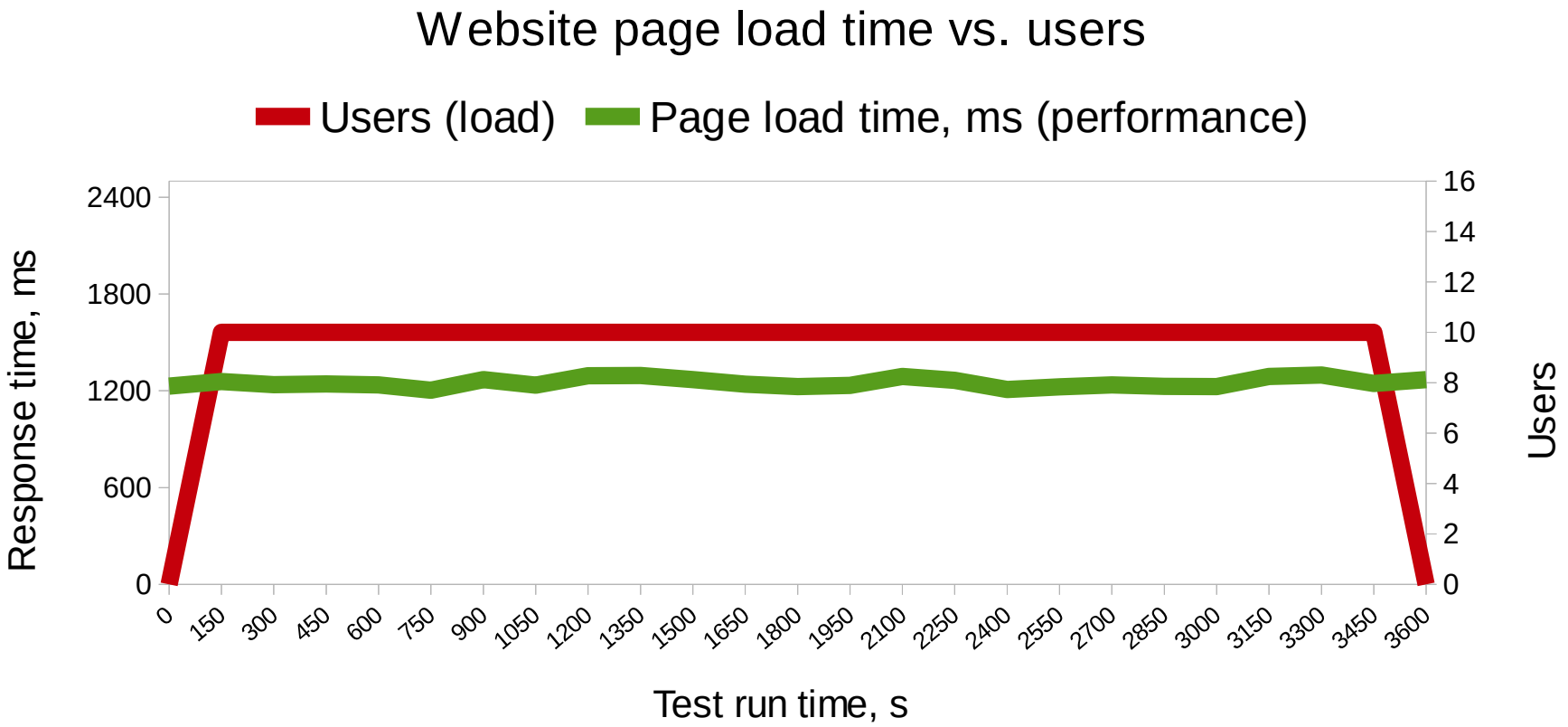
Greta Rudžionienė

Outline

- Introduction
- Case study
- Tips

Performance testing

- Used to determine how system **performs** under particular **workload**

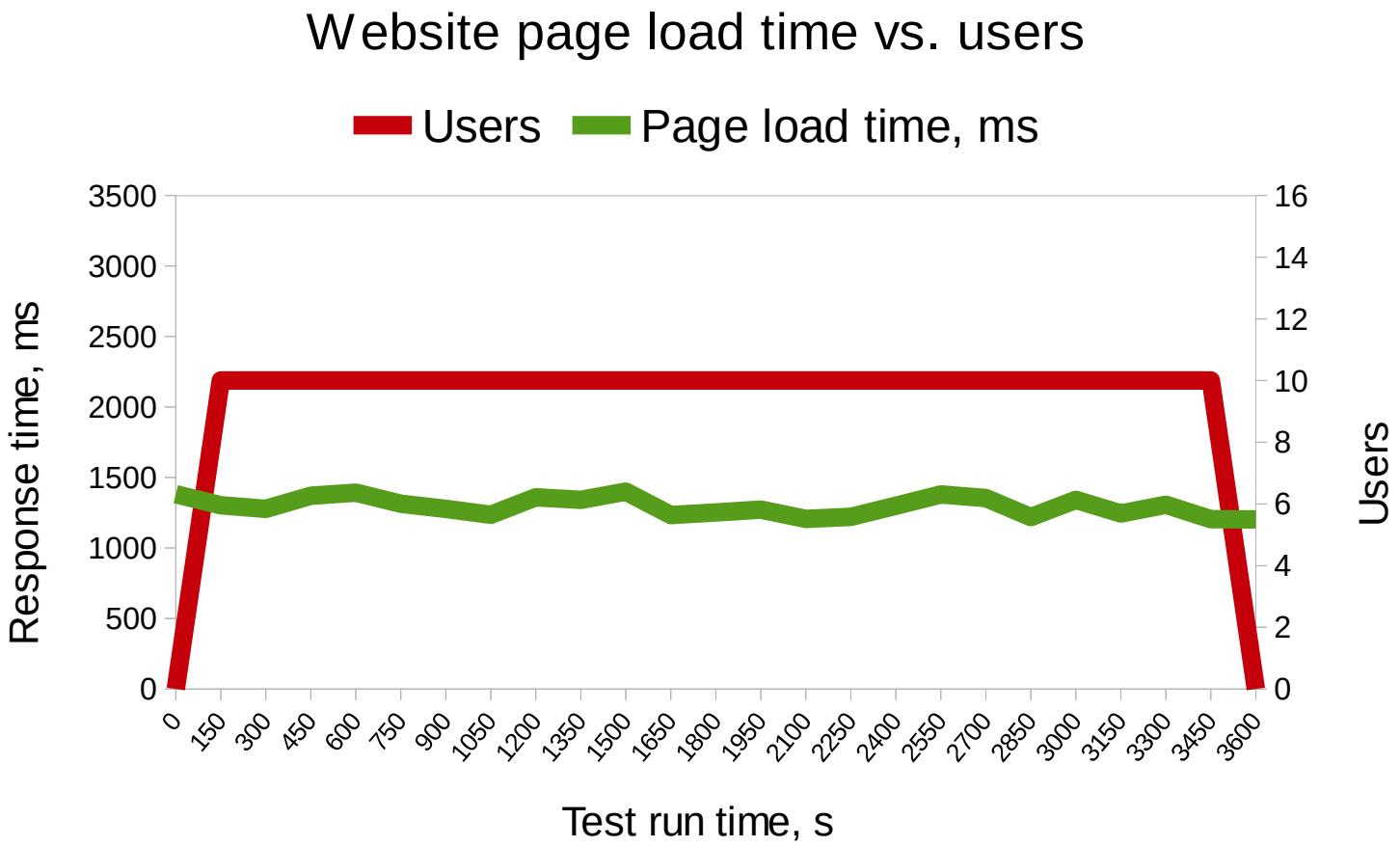


Types

Types: LESS

- **Load**
- Endurance
- Stress
- Spike

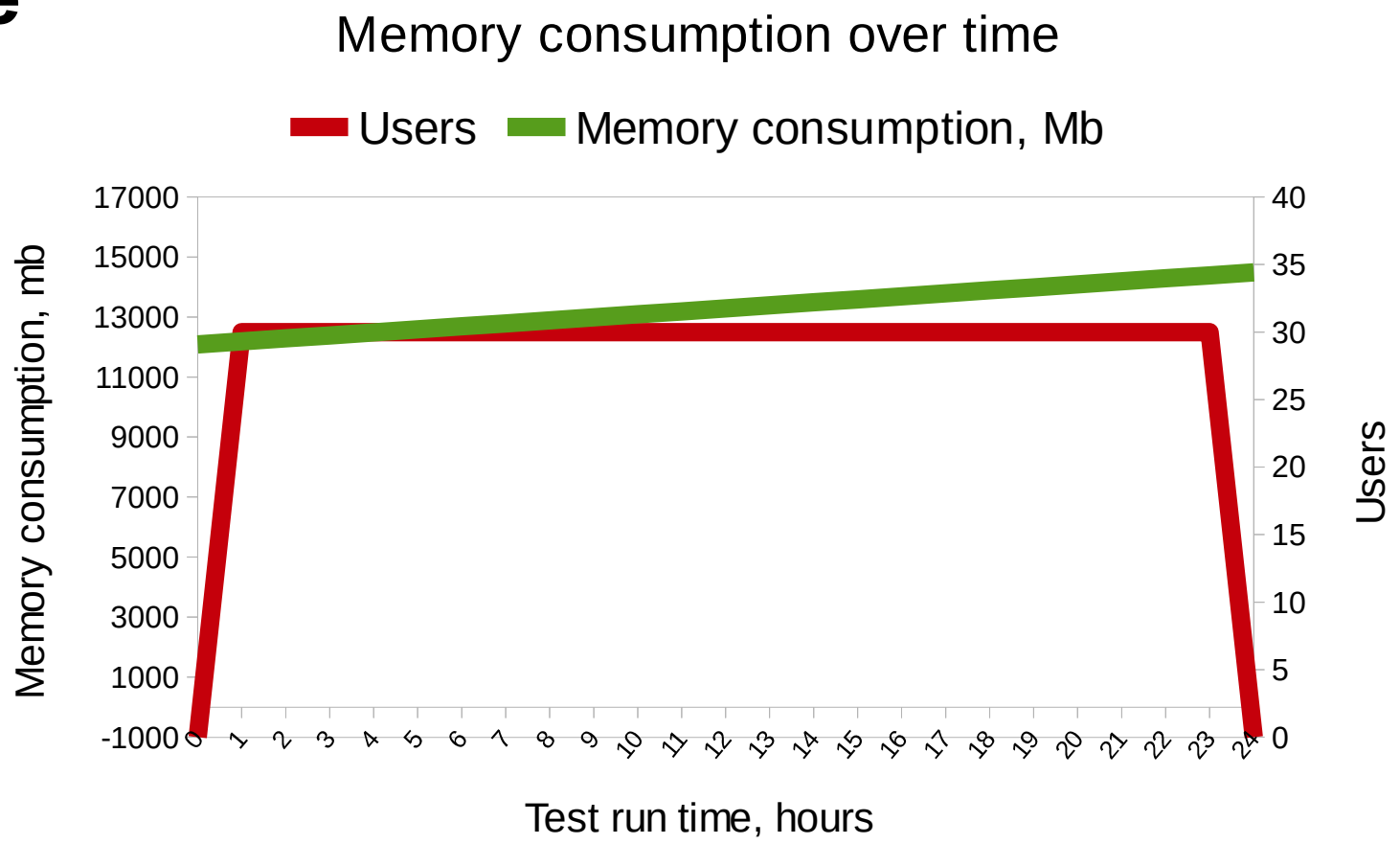
Given the current system load scenario, how will the application behave?



Types: LESS

- Load
- **Endurance**
- Stress
- Spike

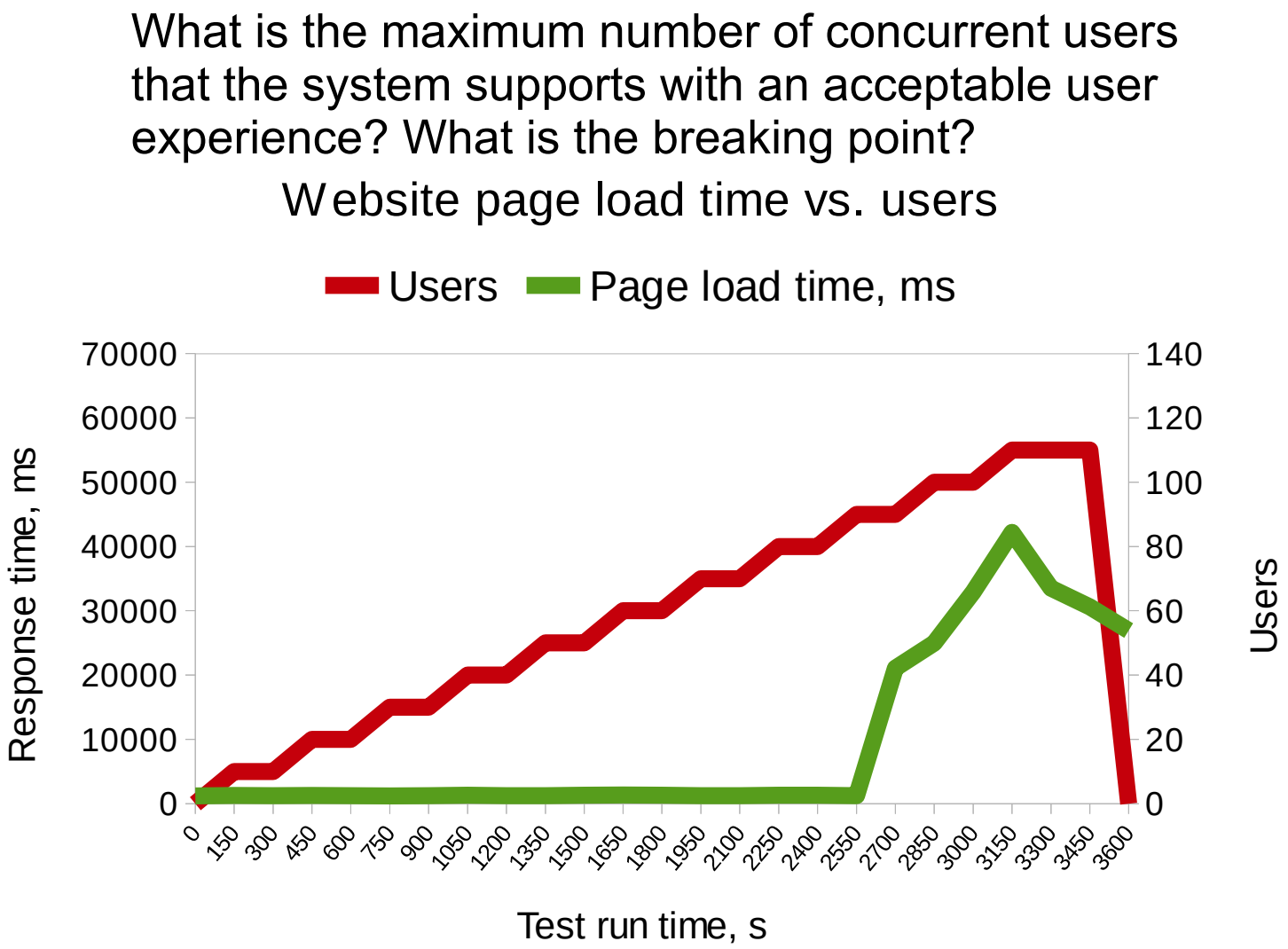
How will the system work after running for longer periods of time (say, after a full day)?



Types: LESS

- Load
- Endurance
- **Stress**
- Spike

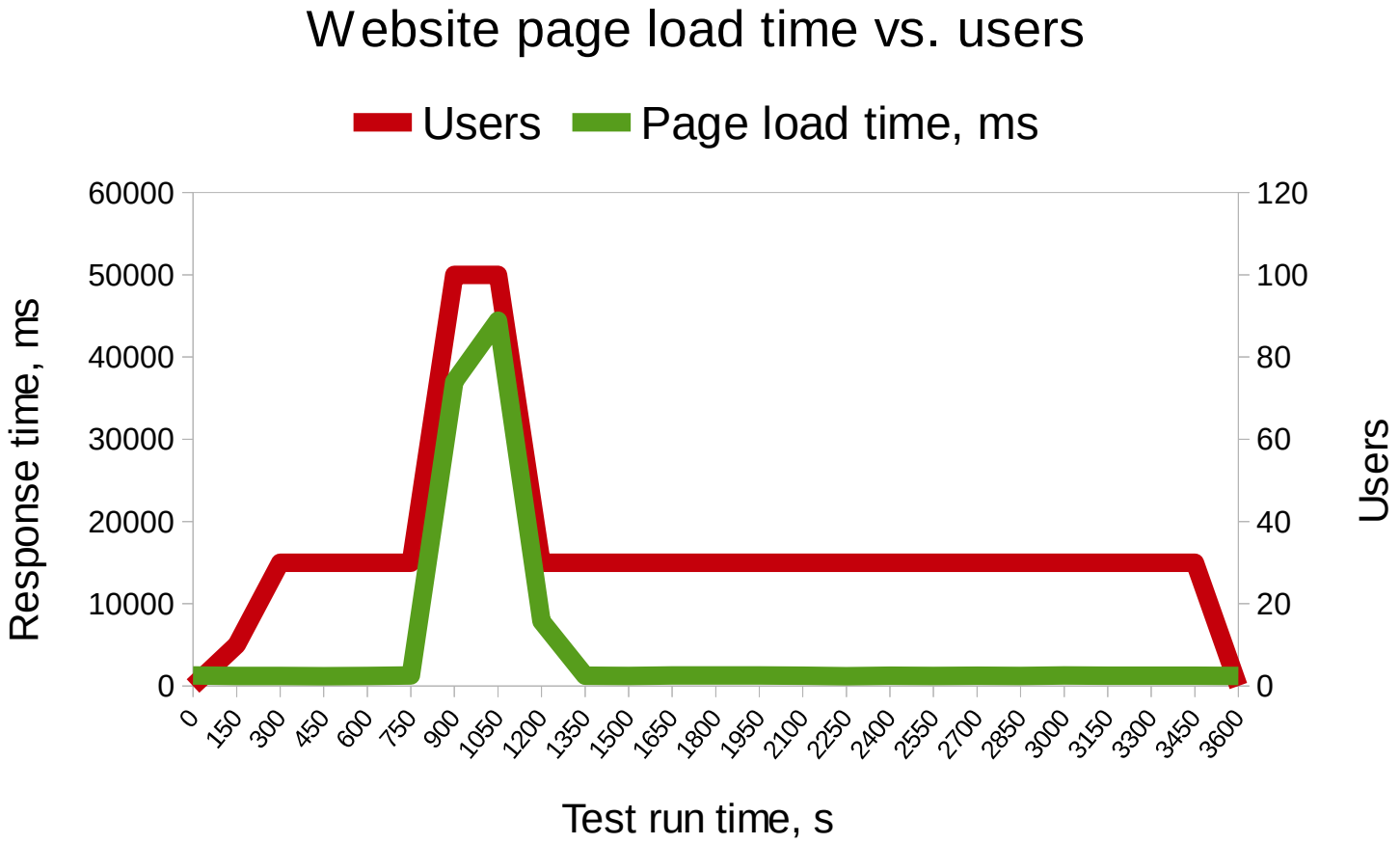
What is the maximum number of concurrent users that the system supports with an acceptable user experience? What is the breaking point?



Types: LESS

- Load
- Endurance
- Stress
- **Spike**

If my normal system works properly and there is a peak in stress, then how fast does the system recover?



Challenges

Challenges

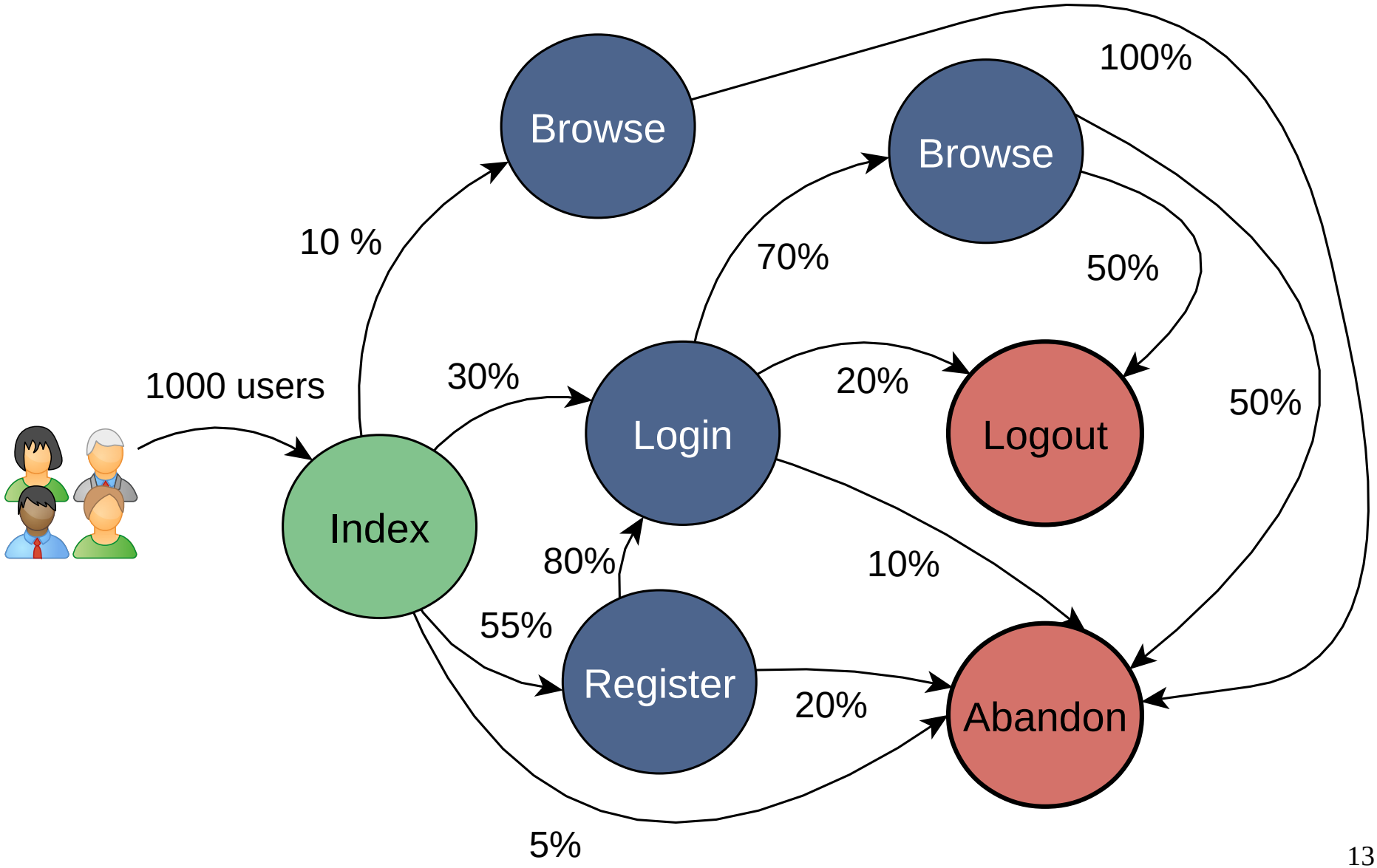
- Environments
- Huge amount of data
- Background noise
- Subjective
- Costs
- Other unknowns

Case study

Case study

- We had to check if web application will be able to handle X user signs up per hour before the advertisement campaign

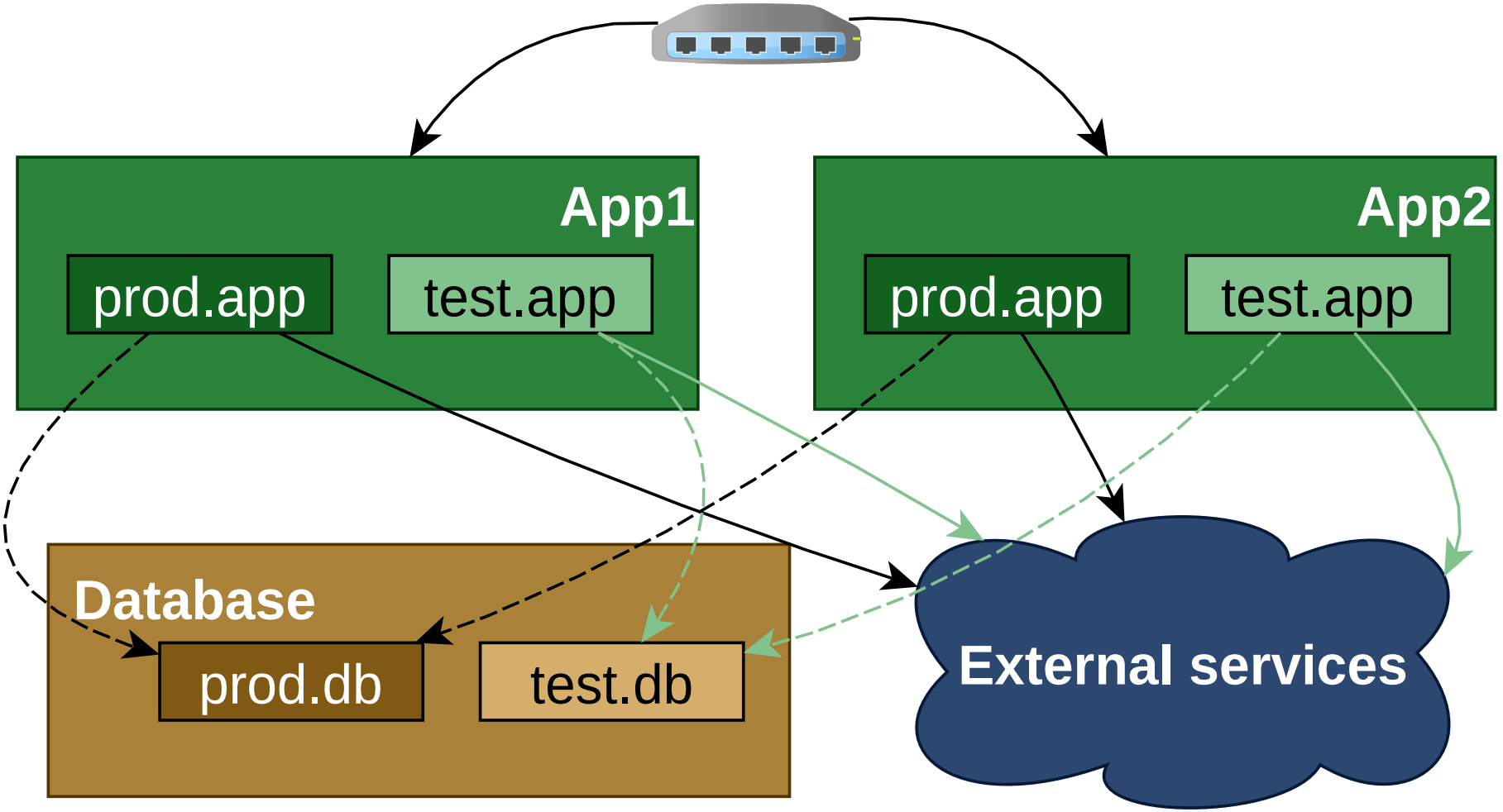
Planning (TVM)



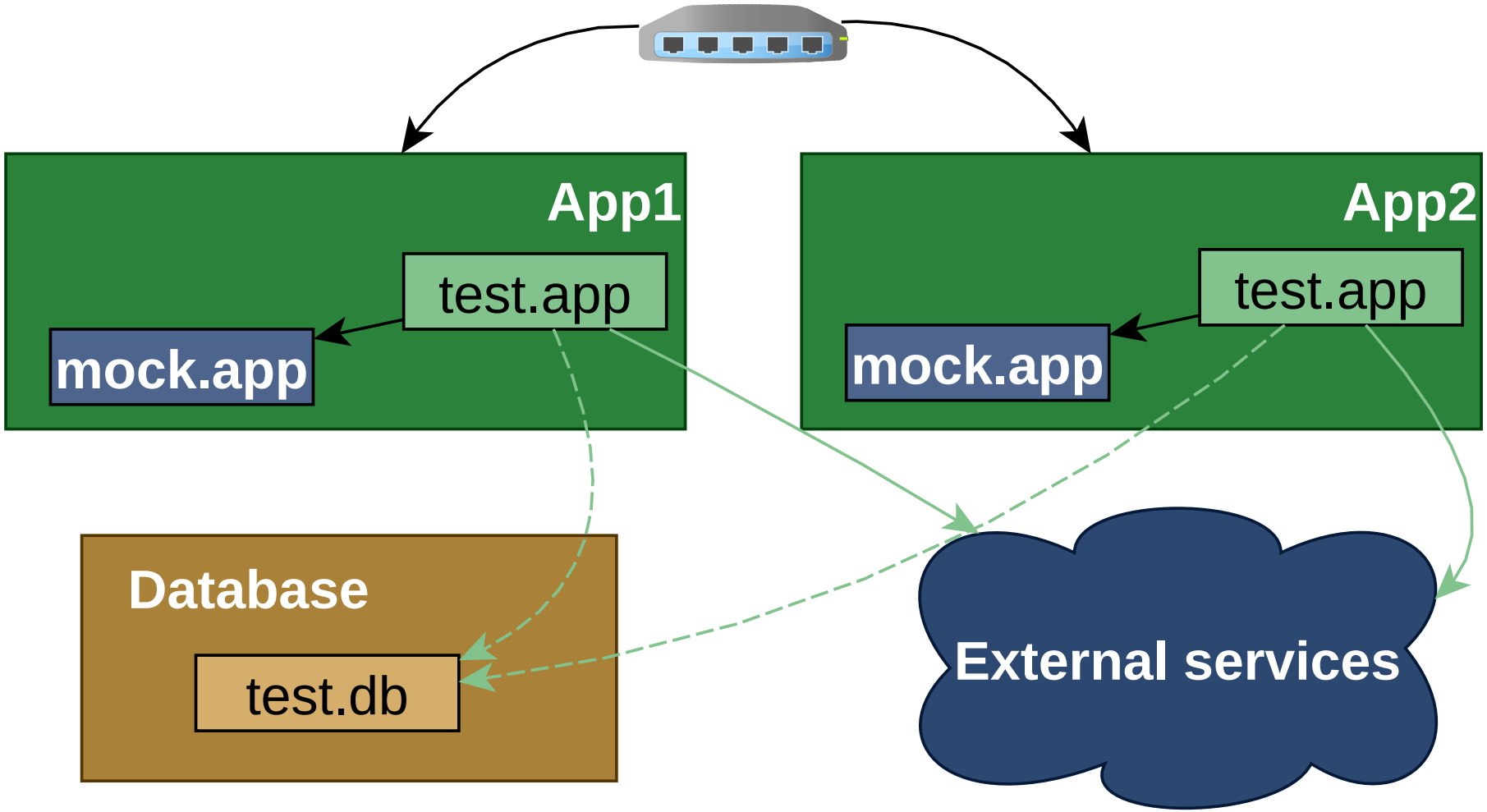
Tools

- HP Loadrunner
- Jmeter
- Gatling
- Cloud based tools
- Others...

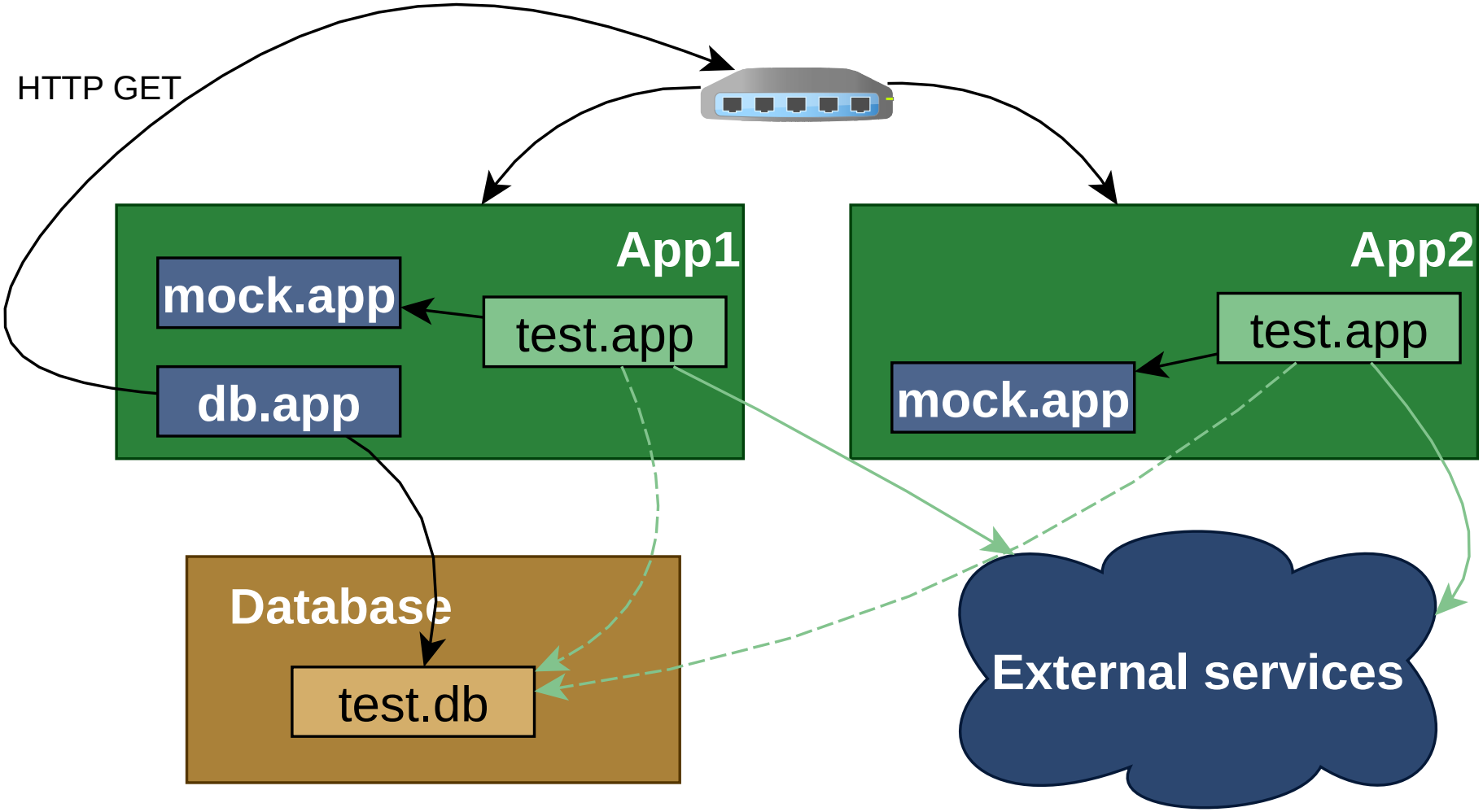
Environment



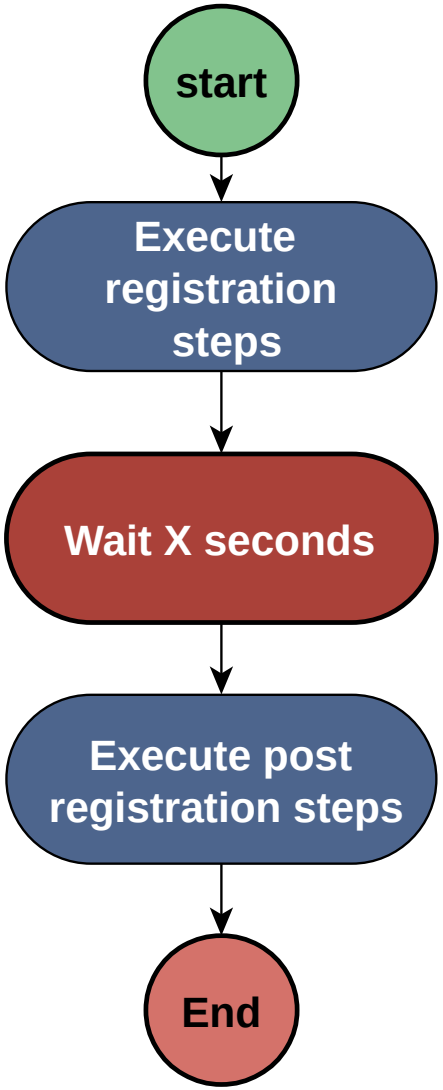
Environment

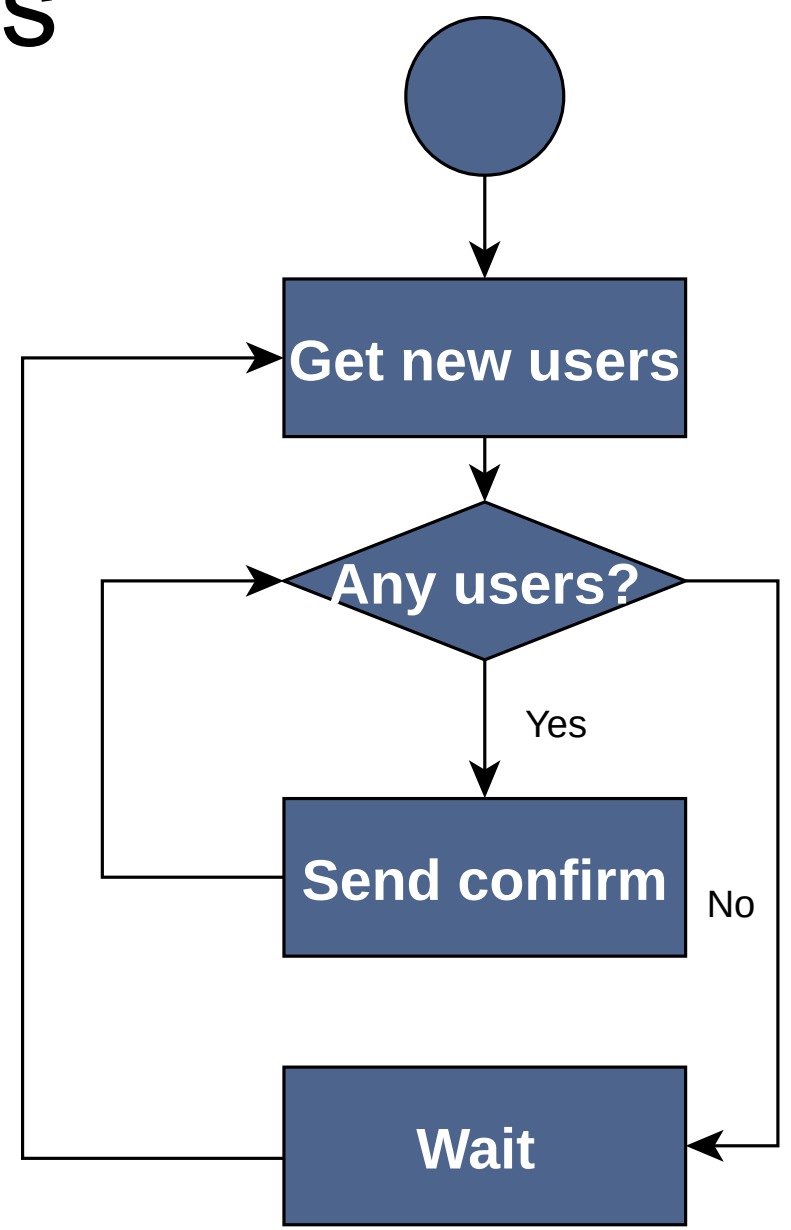


Environment



Tests


start
Execute registration steps
Wait X seconds
Execute post registration steps
End


Get new users
Any users?
Yes
Send confirm
No
Wait

Outcomes

- We got calls from the data center about DDOS attacks from out network
- Scenarios with users not logging out showed session problems
- Missing indexes in database were found

- *In the end system got much more load than expected, but was able to handle it*

Tips

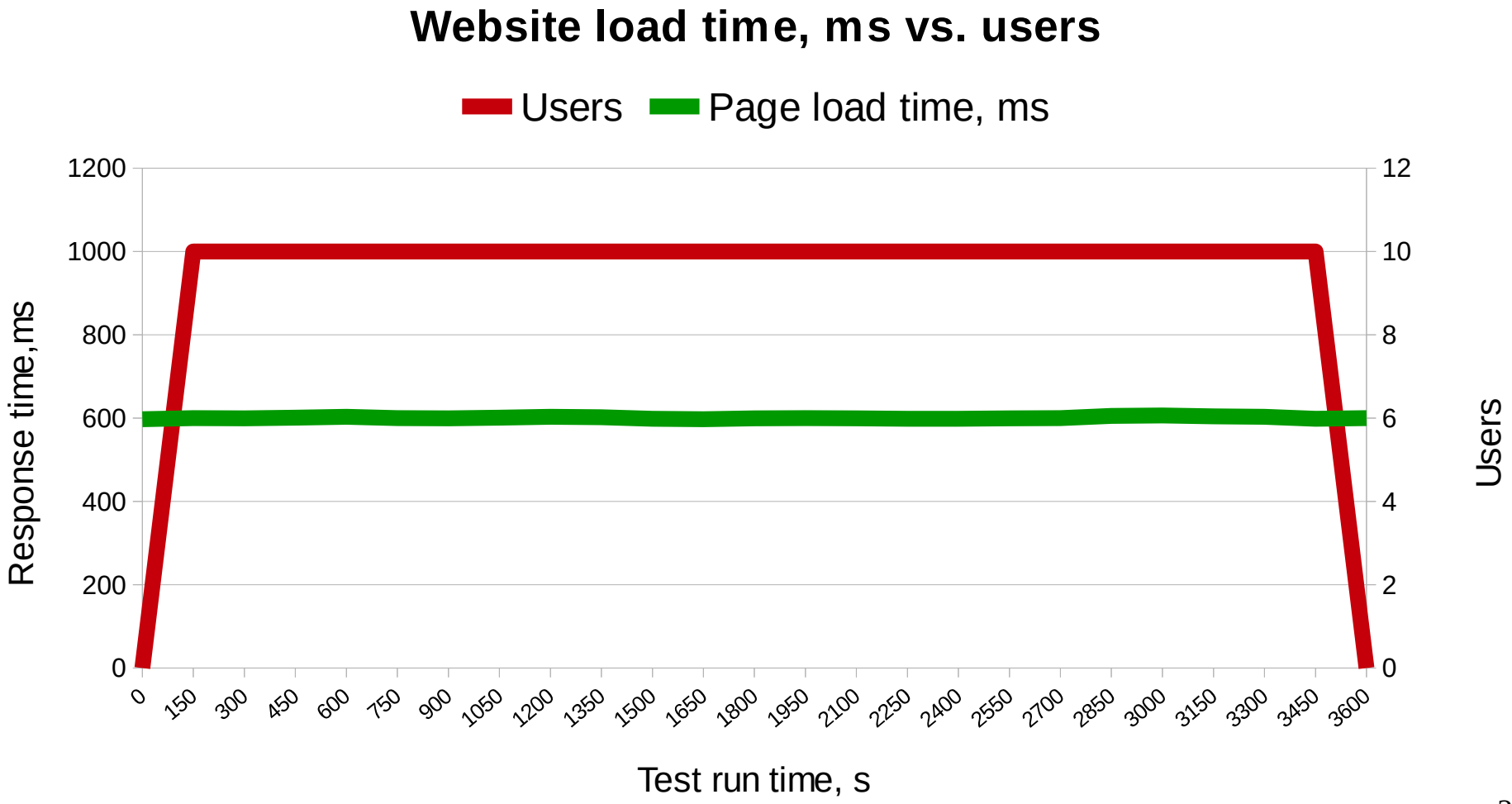
Results

Run time, min	Avg. response time, ms	Requests/s
2	600	2

- Run your tests at least for an hour
- Repeat tests at least 3 times
- Use realistic number of users
- Trusting on average is not good

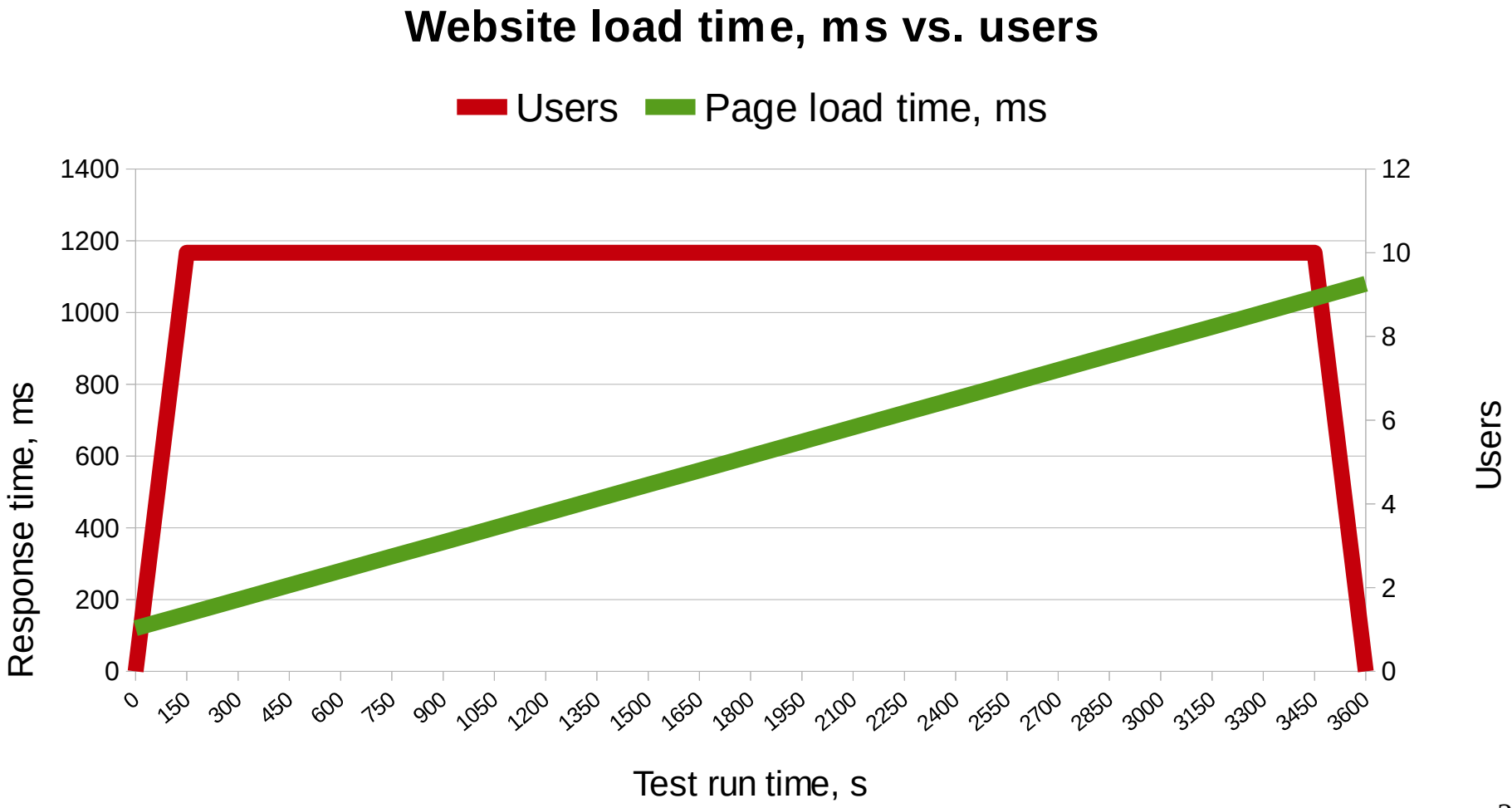
Results: averages

- Supposing you system responds in 600 ms



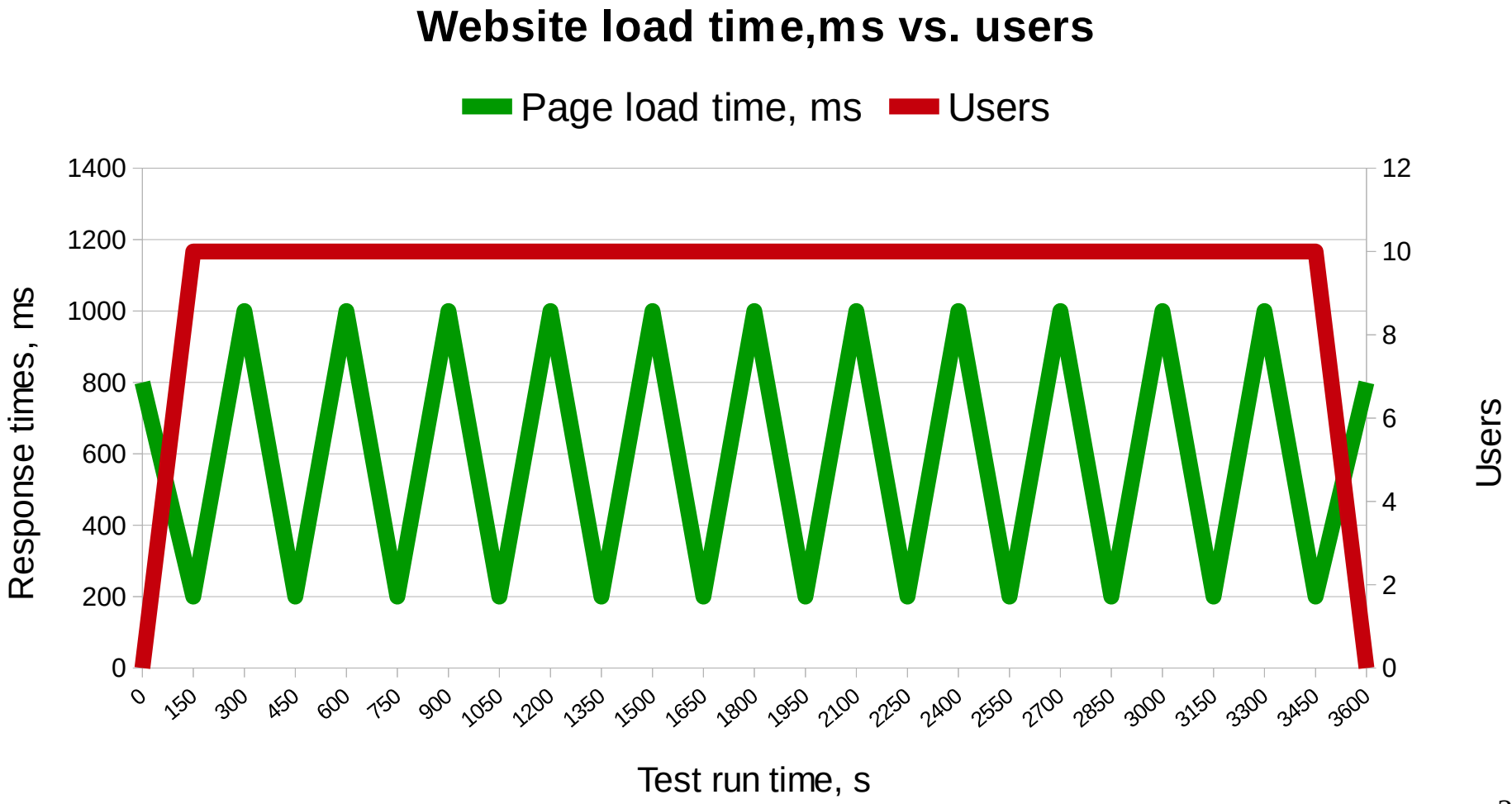
Results: averages

- 600 ms average again



Results: averages

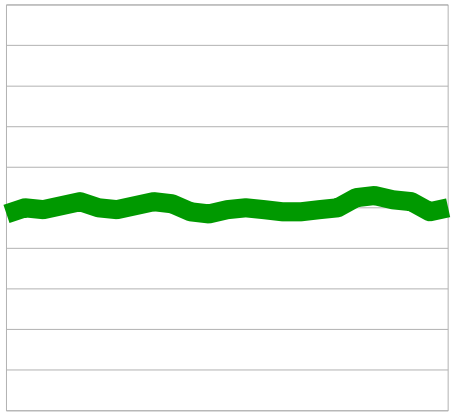
- Still 600 ms average

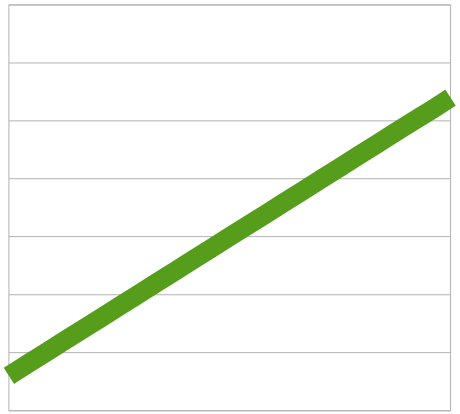


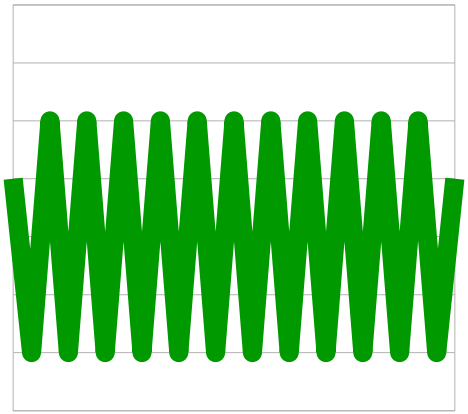
Results

- Use standard deviation and percentiles

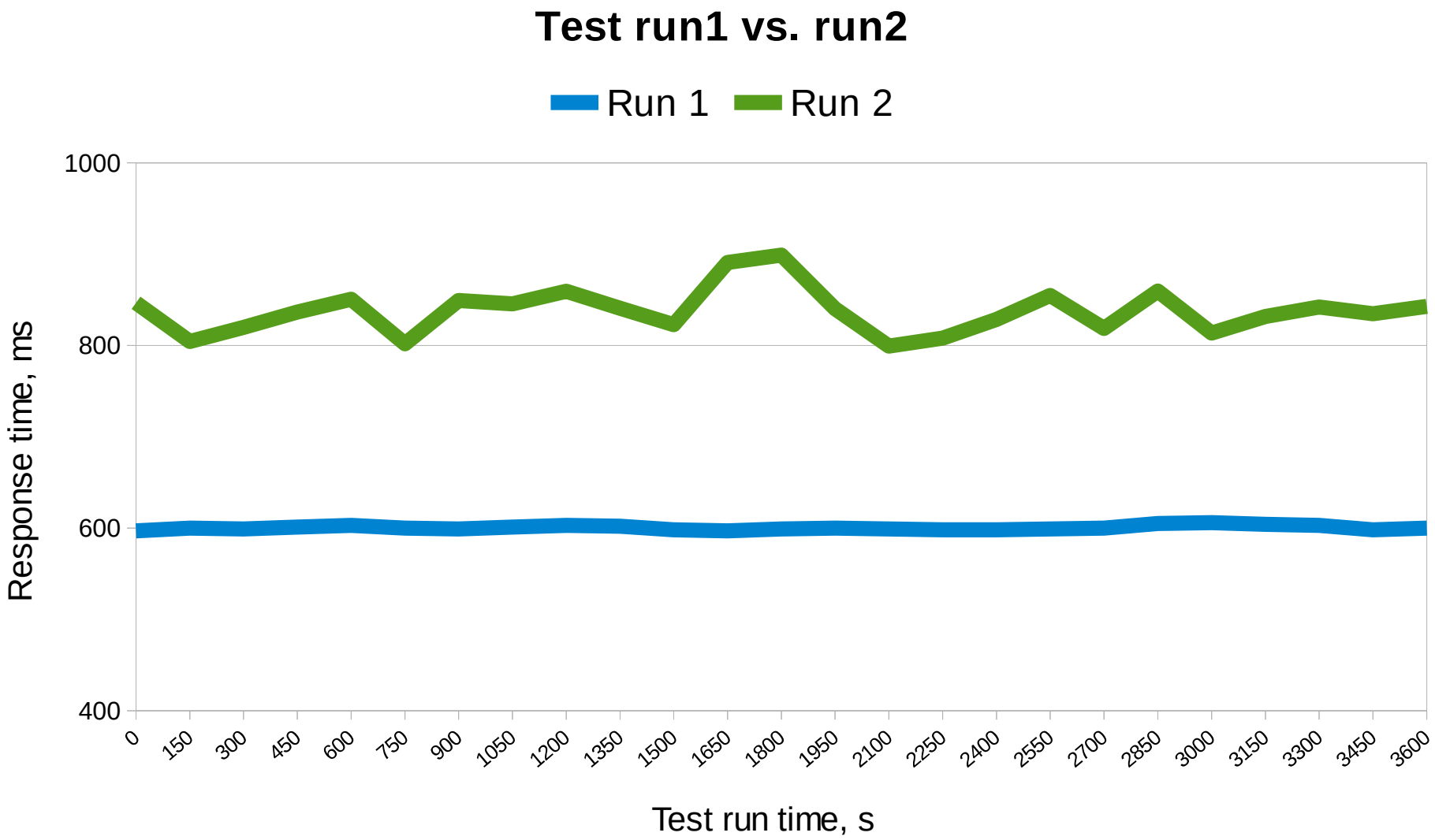
Measure	Sample1	Sample2	Sample3
Avg time, ms	600	600	600
St. deviation, ms	2	395	294
95th, ms	604	1000	1032







Configuration problems



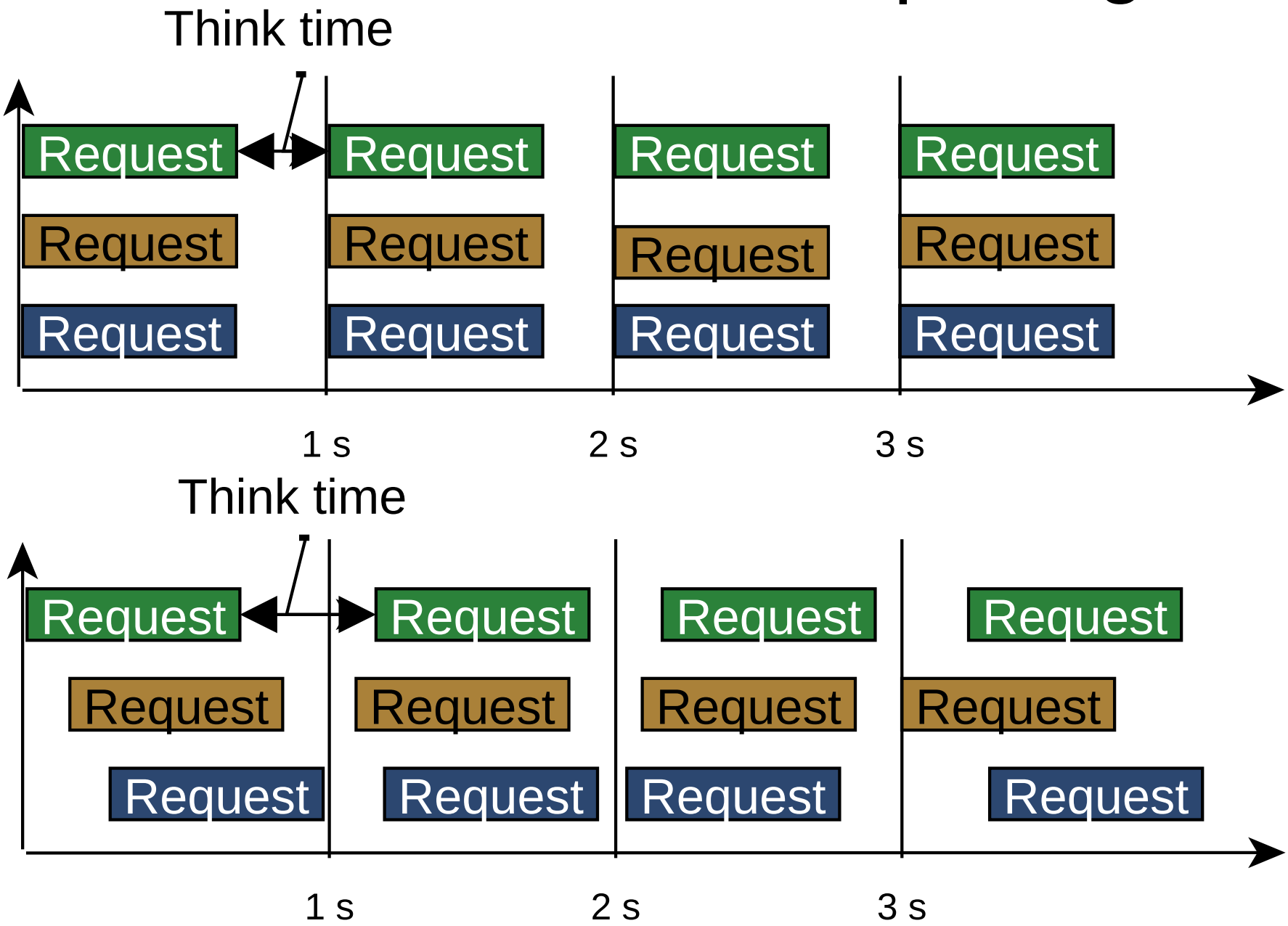
Calculations: real world users vs. threads/users/virtual users

1) 7200 users per hour registering to your app

2) 10000 requests per hour

Measure	Case1	Case2
Avg. request/transaction time, s	5	0.6
Requests/s	7200/3600=2	14400/3600=4
Threads/ VUsers	2*5=10	4*0.6=3

Case2: think time/pacing



Summary

- There are 4 main performance test types: Load, Endurance, Stress, Spike (LESS in short)
- Use realistic test data and loads
- Know your test environment
- Never trust only average values
- And finally – be careful when performance testing

Questions?