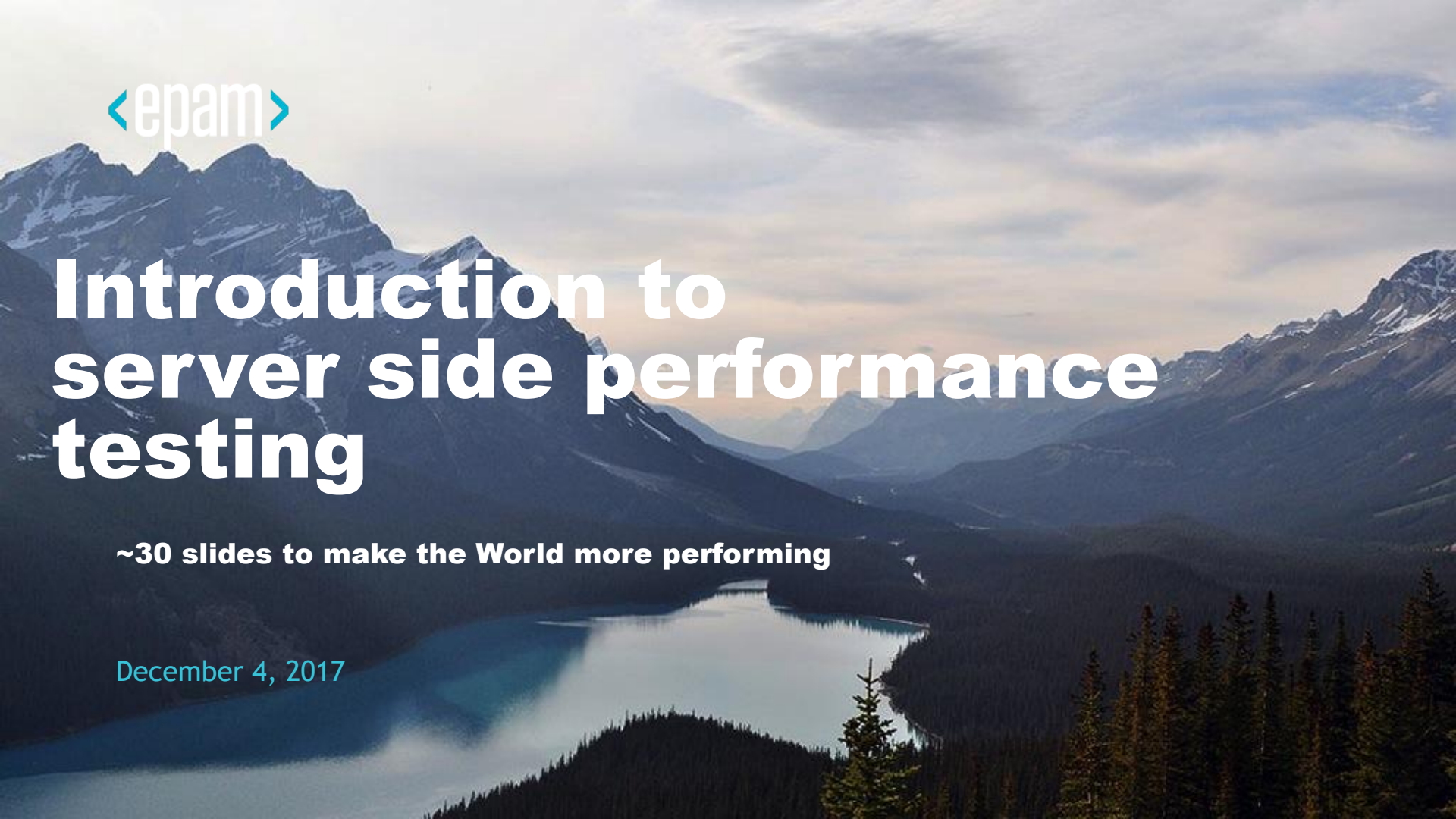


Introduction to server side performance testing

~30 slides to make the World more performing

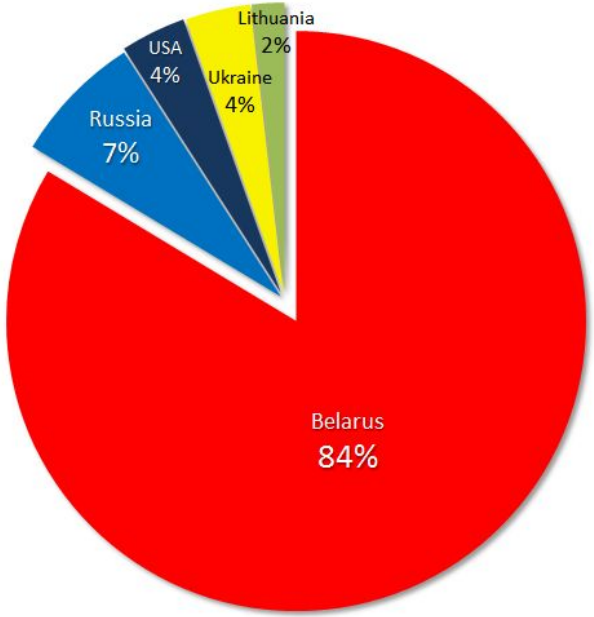
December 4, 2017

AGENDA

1. Quality
2. Performance: Why? Who? How?
3. EPAM POG some facts
4. System under the load – general approach
5. Performance testing definitions
6. Types of tests
7. Monitoring
8. Reporting
9. Literature

EPAM POG facts

- Group started in 1999
- 180+ completed projects (10 industries) at least, plus 40 ongoing projects
- Longest project: 12 years
- Largest project: 22 performance analysts
- Geographical distribution of performance analysts
- across EPAM offices*. Majority is still in Belarus
- Most of performance analysts
- have experience with 2+ performance test tools
- Performance analyst combines skills of
- developer, tester, network engineer, DBA, etc
- Men/Girls is 67%/33%**



SOFTWARE QUALITY

Quality is a property which shows if product meets its functional requirements (expectations) or not.
Anything wrong?

One small correction makes difference:

Quality is a property which shows if product meets its requirements (expectations) or not.

Performance - The degree to which a system or component accomplishes its designated functions within given constraints* (IEEE).

THE PURPOSE OF PERFORMANCE OPTIMIZATION

As per ISO 9126, software product quality consists of:

- ✔Functionality
- ✔Usability
- ✔Reliability
- ✔Efficiency
- ✔Maintainability
- ✔Portability

Functional testing process

Performance optimization process

Our descendants will do that… hopefully

WHY PERFORMANCE?

No Performance → No Quality

Poor Performance → Poor Quality

Good Performance → Good Performance

What are usual consequences of poor performance?

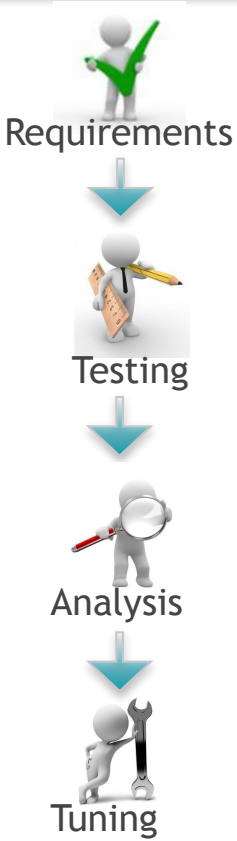
- Unbearable slow software product's reaction to user requests
- Unexpected application crashes
- Expected (still unwanted) application crashes at the moments of extreme increase in load
- Software product vulnerability to attacks
- Problems with product scalability in case it gets more popular
- And more

STEPS TO ASSURE GOOD PERFORMANCE

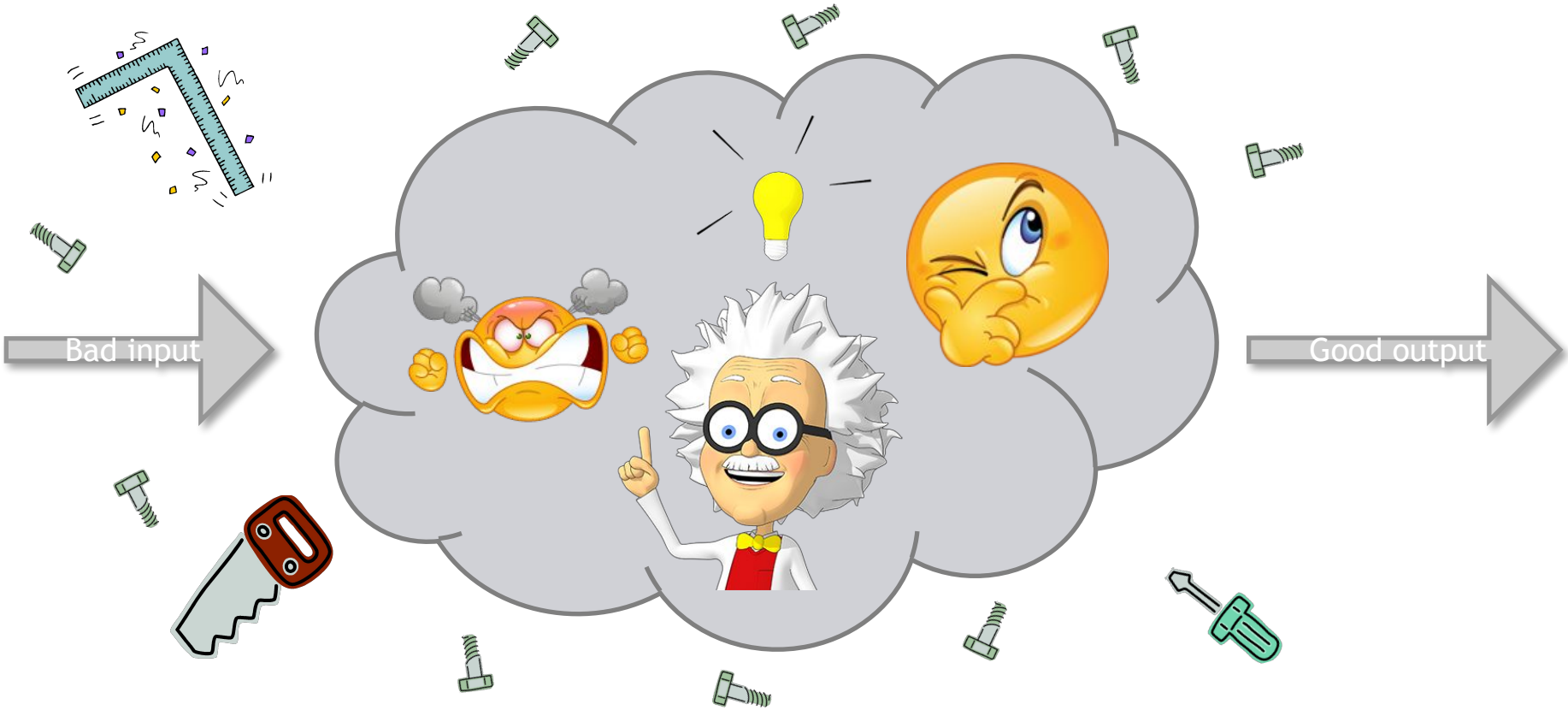
The ultimate goal of performance optimization process is to assure good product performance.

To do that the following steps are to be performed (often in an iterative way):

1. ~~Check~~ Define requirements (= *requirement* analysis)
2. Measure performance (= *performance testing*)
3. Find performance bottlenecks (= *performance analysis*)
4. Fix found problems, so performance increases (= *tuning*)



REALISTIC PICTURE OF THE PROCESS



WHO IS PERFORMANCE ENGINEER?

✔ **Tester**

To conduct a meaningful test

To see system as a target, not as a sacred cow

✔ **System analyst**

To extract information from tons of data

To match cause and effect

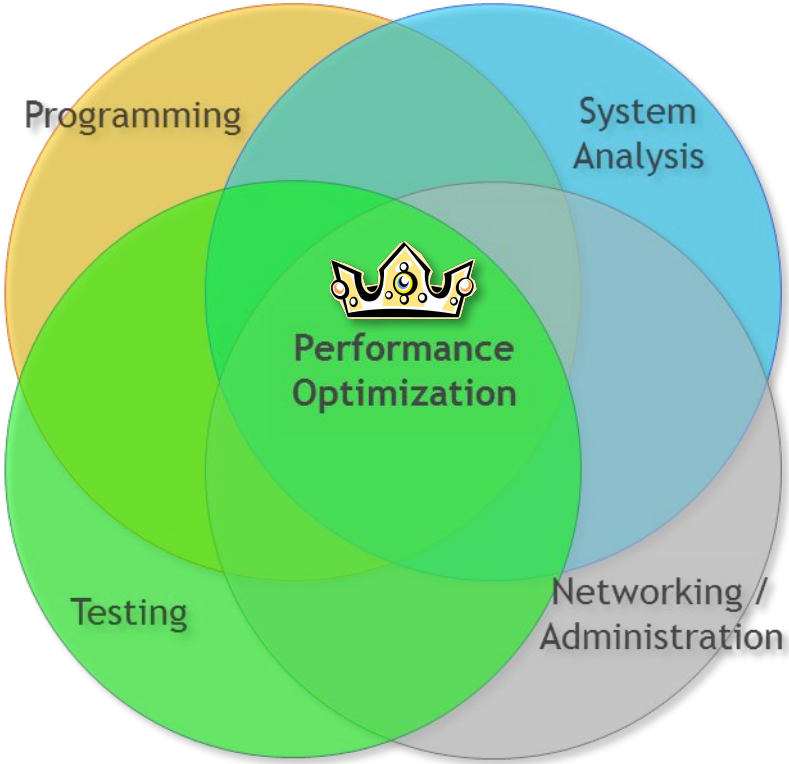
✔ **Programmer**

To develop test script and special tools

To automate testing where it is possible

✔ **Network engineer**

Because no one else cares ☹



WHEN TO START

Classic Approach:

When code is ready and functionally working.
But it's too late because you need time for performance fix.

Better options:

1. Start test at same time as functional team (at least) – meaning you need to develop scripts and prepare everything even earlier
2. Start after functional test, but allocate time for performance fixes and functional re-test

Remember!

You're likely to spend most of time on test results analysis. It is not scripting and test running what makes it all difficult and long lasting, it is rather searching for the performance bottlenecks.

WHERE TO FIND ROOTS OF PROBLEMS?

What? Damn web-site is so slow!

Typical end user

Typical first assumption:

Problem relies somewhere on the end user side.
This is not always true.

User's computer

Bad client side code? JS, Flex, etc.
Weak modem?

Too many hops?
Too slow connection by design?
Broken hardware?
Too far away?

Internet connection

Bad (ineffective, dangerous, stupid, etc.) code?
Improper configuration?

Web-server

http://mynews.com

Weak hardware?
Some hardware problems?
Improper configuration?

Wrong database system?
Bad indices?
Bad code?
Weak design -> locks?

+ wrong topology?

Internet / LAN connection

...

Application Server

News web-service

Internet / LAN connection

Database Server

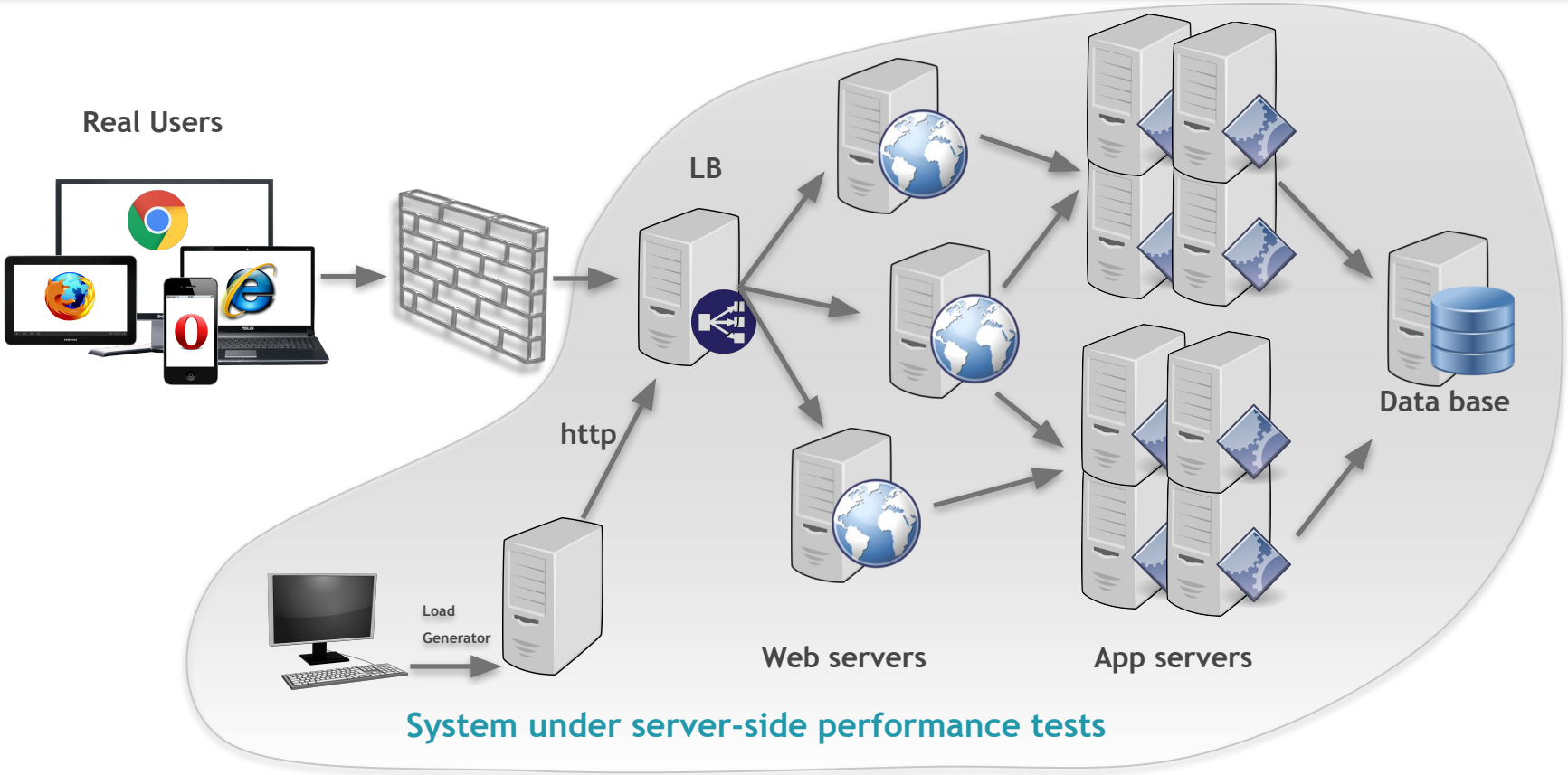
News database

+ Versions incompatibility?

Network again!

...

WHAT IS WEB-SYSTEM FOR PERFORMANCE ANALYST?




COFFEE
BREAK

SOME DEFINITIONS

Performance = response times + capacity + stability + scalability

Performance of a software system: property of a system which indicates its ability to be as fast, powerful, stable, and scalable as required.

SOME DEFINITIONS

Performance bottleneck

Performance testing

Load testing:

TYPES OF PERFORMANCE TESTS

As said before,
Performance =

Capacity

+Response times

+Stability

+ Scalability

So there are types of tests to measure performance:

Ramp-up test

Fixed load test

Longevity test

Also TESTS to mention

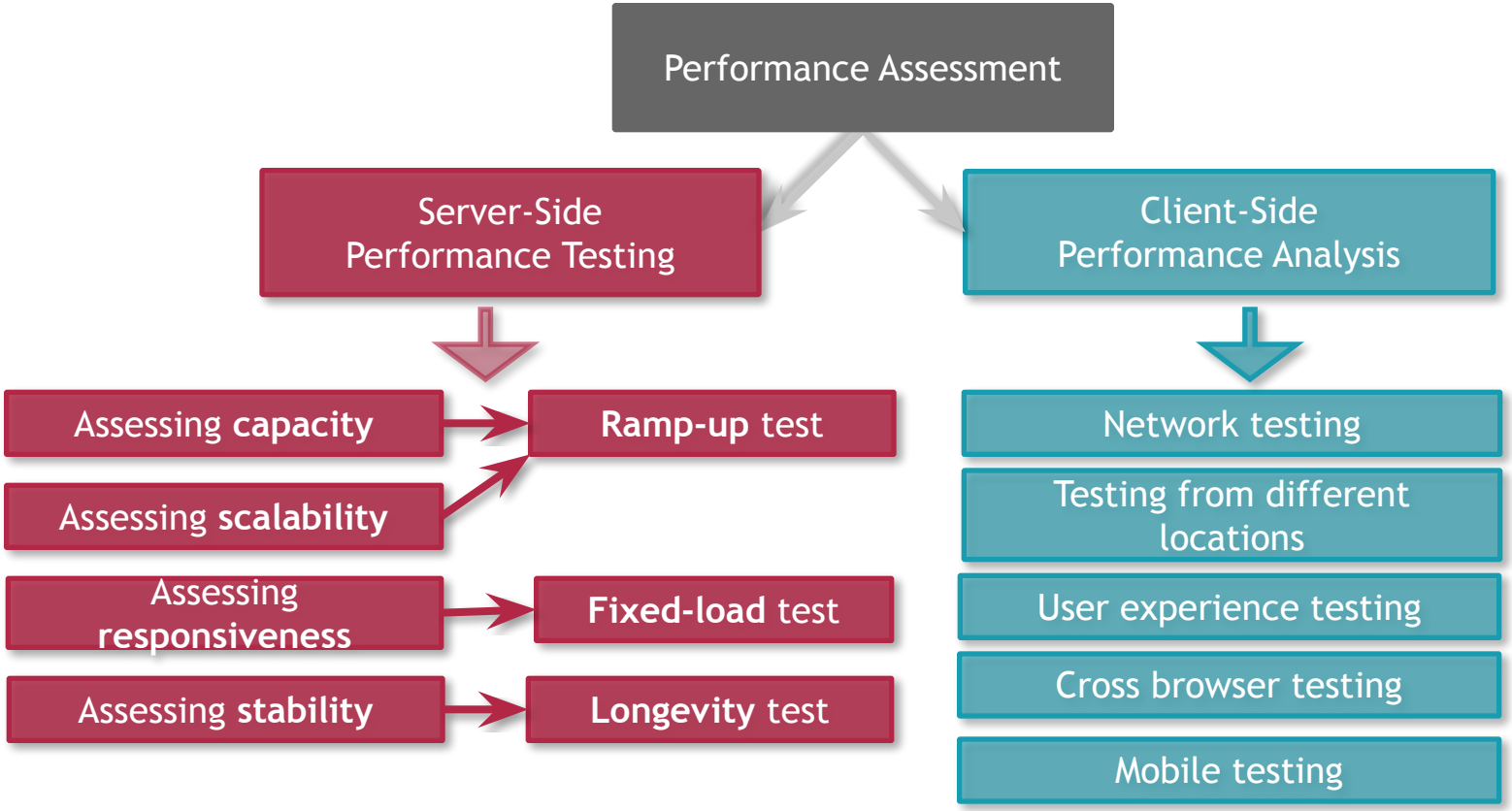
In fact, there are some more tests to mention:

- Rush-hour
- Rendezvous
- Render time
- Production drip

… And even more to invent!

Ask me how to join performance optimization group to know the details ☺️

SUMMARY: TYPES OF PERFORMANCE TESTS



MECHANICS OF CAPACITY

Assume there is some web application that is able to process some requests.

It takes 1 second to process 1 request with about 10% of resources utilization.

What happens if 1 user sends 1 request?

What happens if 2 users simultaneously send a request each?

What happens if 10 users simultaneously send a request each?

What happens if 11 users simultaneously send a request each?

Some options to consider:

Option 1: Application will crash.

Option 2: Application will process 10 requests within 1 second and throw 11th request away (not a good thing, because we make 11th user completely unhappy).

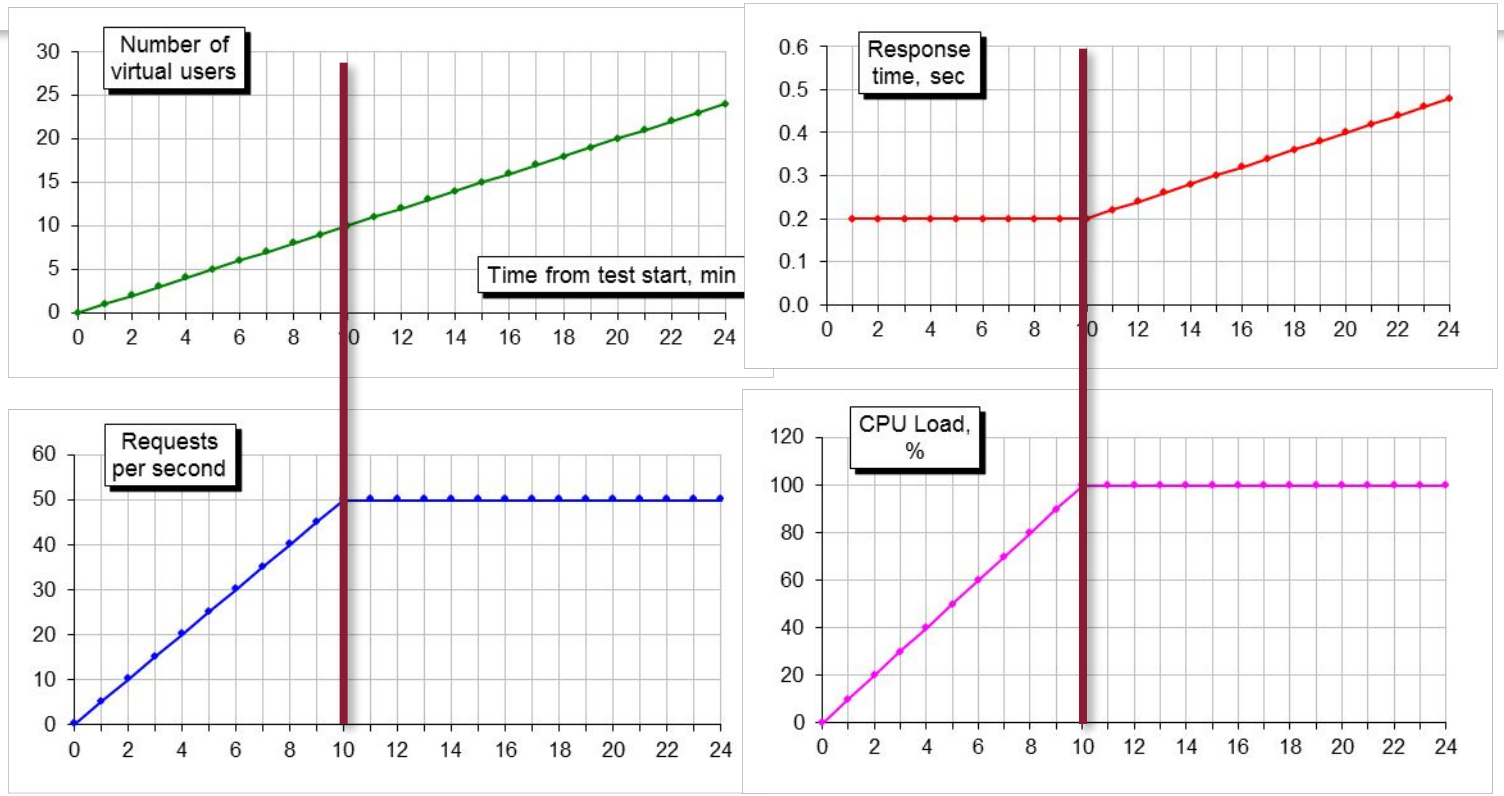
Option 3: Application will process 10 requests within 1 second and put 11th request in the queue (much better, but still not a good thing).

Option 4: Application will serve all 11 requests, but it will take more than 1 second to process each of the requests (sill not a good thing, but much better than options 1-3).

What happens if 20 users simultaneously send a request each?

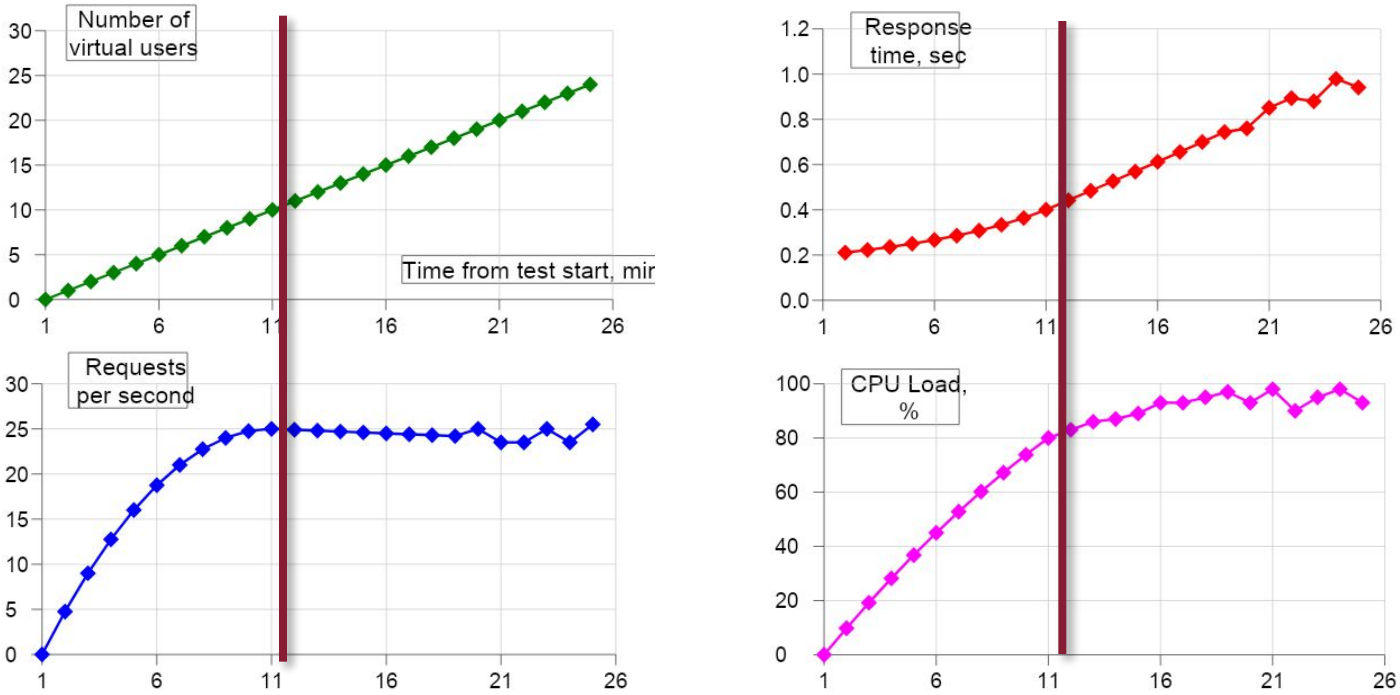
We cannot load this application with more than 10 requests/sec because it is its natural capacity

RAMP-UP TEST AGAINST IDEAL SYSTEM



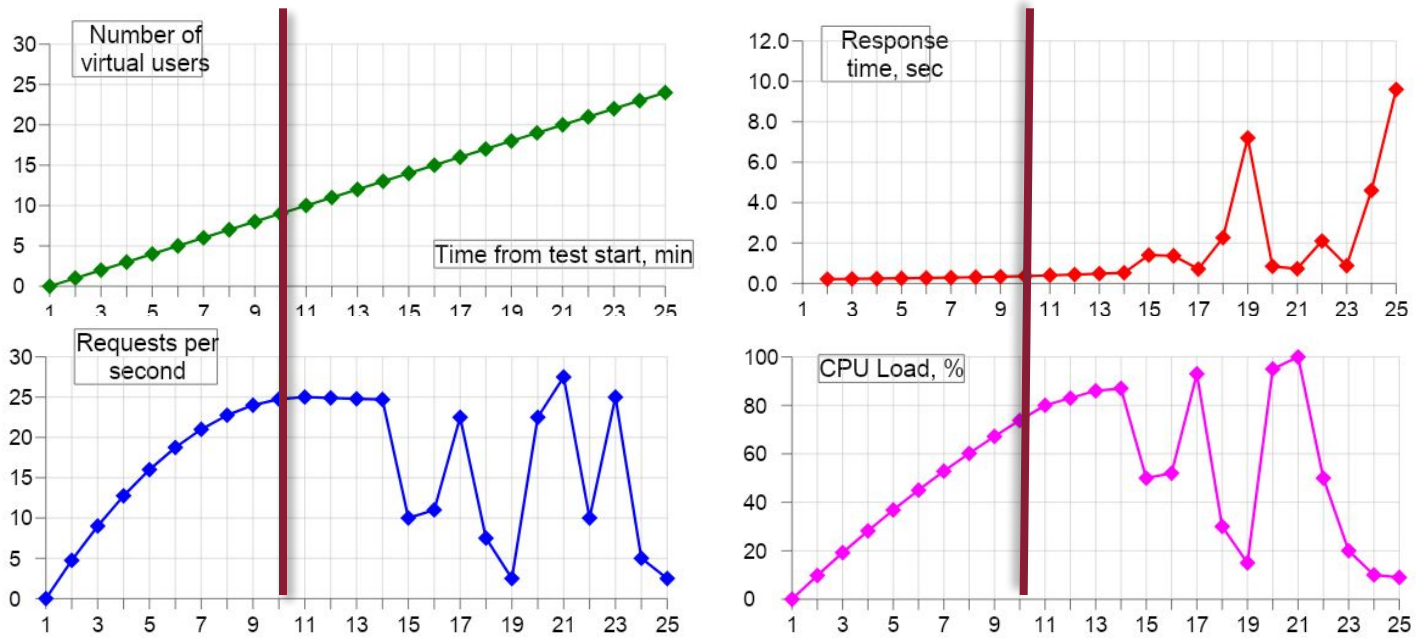
The moment when ramp-up test reaches system capacity is named as system saturation point.

RAMP-UP TEST AGAINST REAL GOOD SYSTEM



- System is stable enough to have capacity (thus saturation point)
- Saturation point is evident

RAMP-UP TEST AGAINST REAL BAD SYSTEM

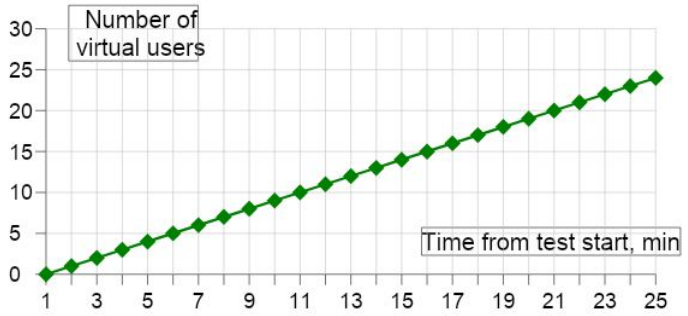


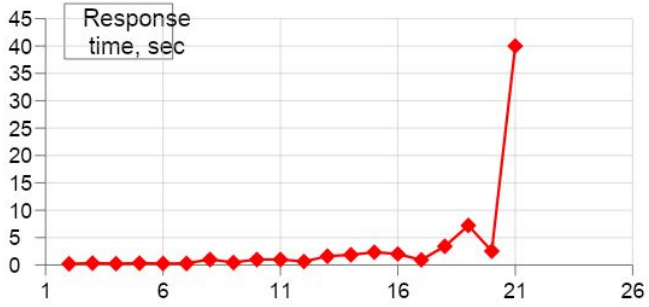
Bad news: system is not stable after saturation

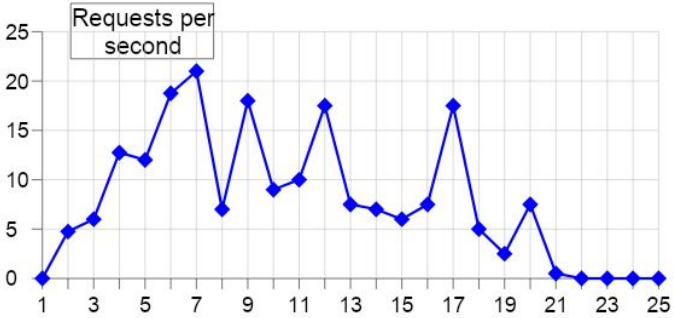
Any good news?

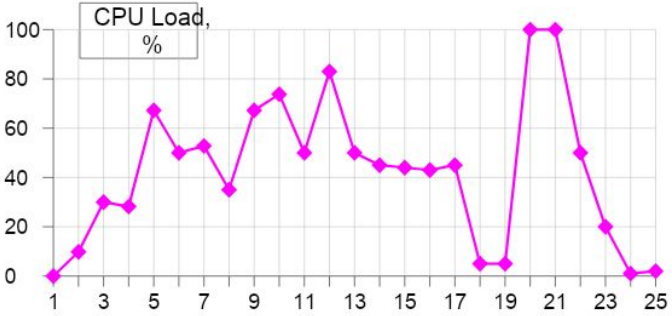
System is stable enough to have capacity (thus saturation point)

RAMP-UP TEST AGAINST REALLY BAD SYSTEM









Bad news: a lot of ones ☺️

Any good news?

It was we who observed that, not real users

Perfect ramp-up test!

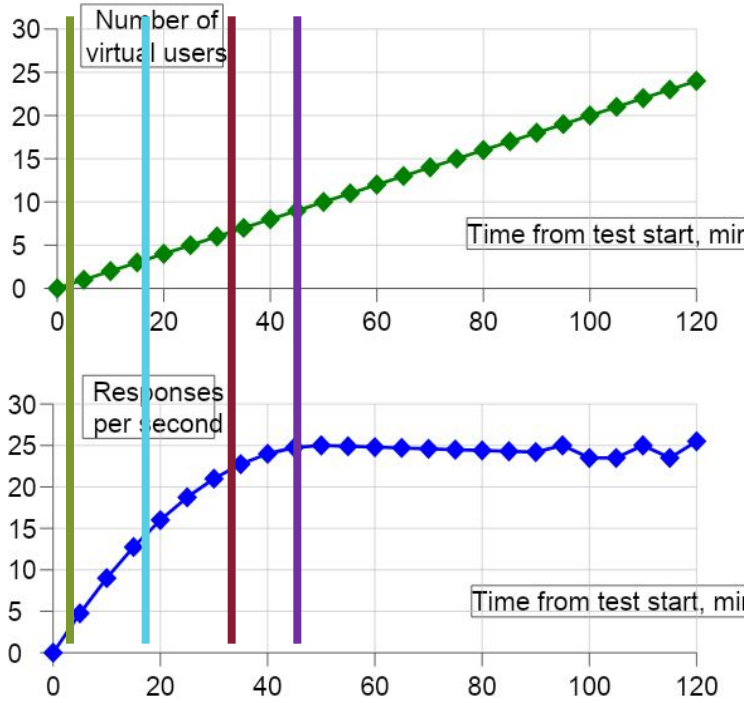
TESTS WITH FIXED LEVEL OF LOAD

To setup the test:

1. Define level of load
2. Define duration

Define the load level vs capacity:

- Low-load as 1 virtual user (~10% of saturation load);
- Mid-load as 4 virtual users (~45% of saturation load);
- High-load as 7 virtual users (~80% of saturation load);
- Or some magical level of load (e.g. production-like, expected, etc.)

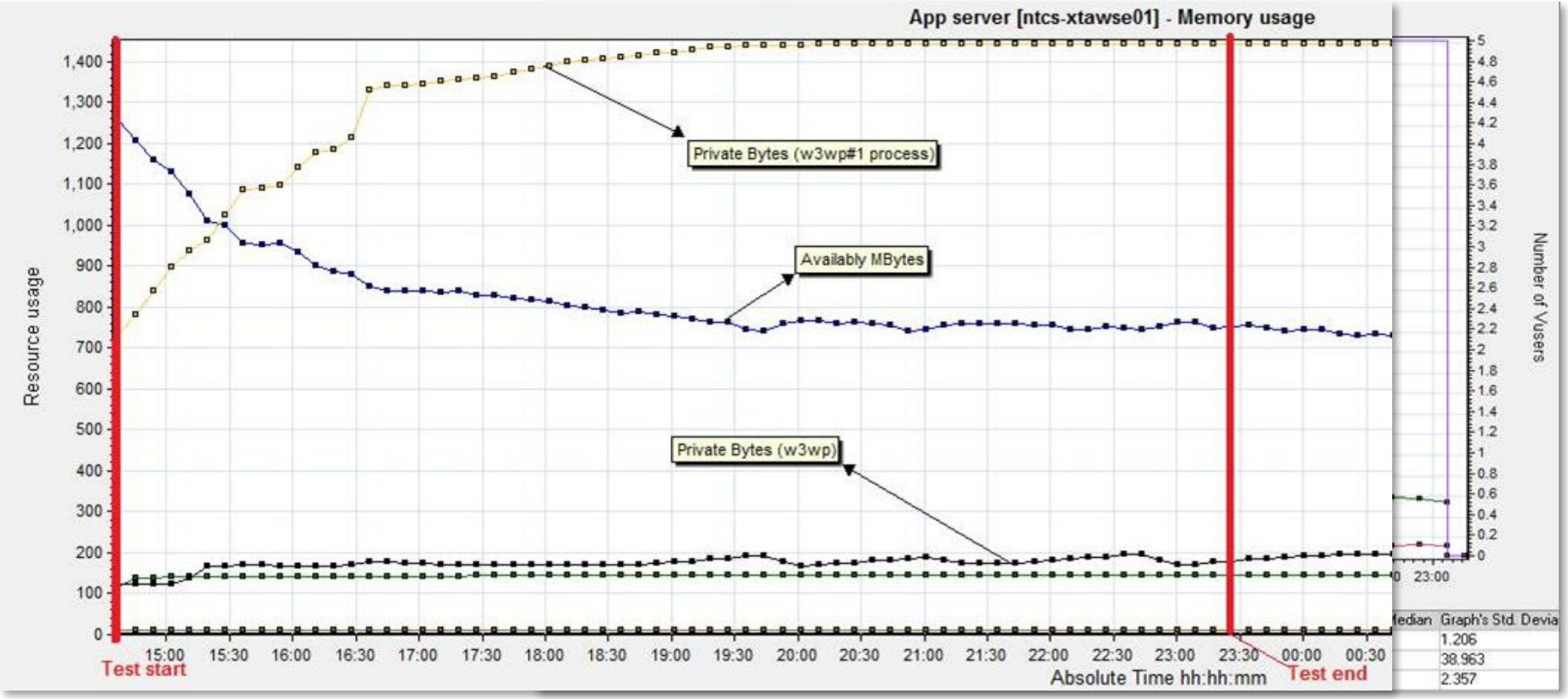


Duration should be:

Long enough;

But also short enough.

LONGEVITY TEST



WHY TO MONITOR

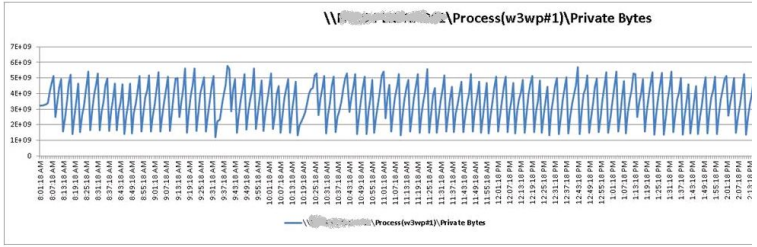
Issues and unexpected situations occur during performance testing quite often. In order to understand the root cause, find the bottlenecks and analyze results thoroughly, hardware resources monitoring of all system components involved is required.

Important: load generator is also a component of the system under test and bottleneck may reside there as well!

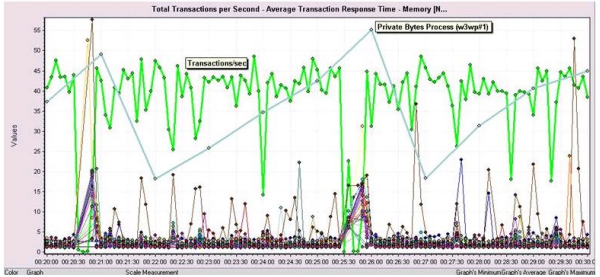
It`s a good practice to monitor resource online during a running test (for ramp-up it`s a must): that way you`ll have at least some high-level idea of what is happening inside the system.

Also, you can try using your application manually from time to time during a running test to check if it`s doing fine: sometimes it`s the only way to catch sophisticated errors that automated tools may not be able to capture.

Below is graph of Private Bytes w3wp#1 process:



Below is graph that shows degradation for total transactions per second and average response times at the moments of recycles:



BASIC PERFORMANCE COUNTERS TO MONITOR

The basic three resources to monitor:

CPU utilization

Memory consumption

Disk Usage

+ possibly

Network traffic via specific NIC

It's not all! It's just what to start with.

For example, in Windows, it's Performance Monitor (search for perfmon in Start)

In perfmon: system monitor is for online monitoring, and log for recording data for analysis.

Perfmon works with Windows Performance Counters.

Hierarchical concept here:

Performance object

Performance counter

Instance

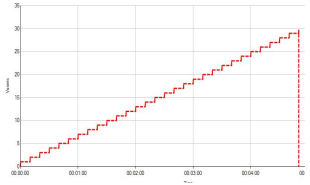
Resource	Object	Counter	Instance
CPU	Processor	% Processor Time	Total
Memory	Memory	% Committed Bytes in Use	n/a
Disk	Physical Disk	% Idle Time	Total
Traffic	Network Interface	Bytes ... /sec	<specific NIC>

WHAT TO REPORT

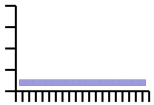
The rule is very simple:
Report results vs. purpose of the test

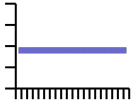
So, just remember what the goals of the tests are, and it’s clear what to report:

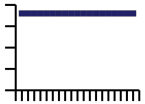
Ramp-up test →



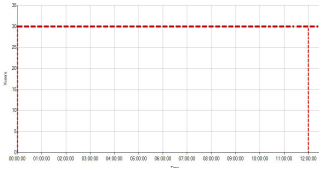
Fixed load test →

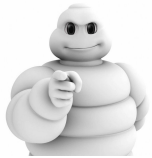






Longevity test →





HOW TO REPORT

Choose the right way of reporting, so your information will reach the addressee

The general rules are simple again:

- Remember you're writing a report for someone who will [make an attempt to] read it
- Although you might be collected tons of data, report should contain useful information, not data
- Keep information clear, accurate, and consistent

Typical sections of a performance test results report are:

- Title of the report, author, and date
- Test setup configuration (what, when, where, and how was tested; objectives of the test)
- Table with related artifacts (links to any raw data, related test results reports, etc)
- Test results section describing findings with all necessary details, supporting charts and tables
- Conclusions and recommendations.* The most difficult one!

TOOLS

LOAD/PERFORMANCE

- Apache JMeter
- Load Runner
- Microsoft Visual Studio
- Silk Performer
- NEOLOAD
- BlazeMeter
- TeamCity
- The Grinder
- ORACLE
- Jenkins

CLIENT-SIDE

- YSlow
- Firebug
- Fiddler Web Debugging Proxy
- HttpWatch
- dynaTrace software
- Selenium

PROFILING

- ANTS PERFORMANCE PROFILER
- YourKit Java Profiler
- ORACLE JROCKIT
- VisualVM
- WinDbg
- dynatrace
- DebugDiag
- ej-TECHNOLOGIES JProfiler
- SciTech Software .NET Memory Profiler
- QUEST SOFTWARE Simplicity At Work
- Windows Sysinternals
- MS SQL Server Profiler

LOG ANALYSIS

- Gather
- Log Parser Studio
- WebLog Expert
- In-house parsers & scripts

NETWORK/ MONITORING

- PuTTY
- WIRESHARK
- tcpdump
- dstat
- sar
- vwstat
- AppDynamics
- Foglight
- OMNITURE
- New Relic.
- ZABBIX

MATERIALS TO EXPLORE

Useful book that provides great introduction to both server and client side performance testing:
“Web load testing for dummies”, by Scott Barber and Colin Mason
www.itexpocenter.nl/iec/compuware/WebLoadTestingForDummies.pdf

Nice book about performance testing .NET web-systems (in fact, a lot of general concepts there):
“Performance testing Microsoft .NET web applications”, by Microsoft ACE team
http://www.microsoft.com/mspress/books/5788.aspx

Statistics for dummies ☺:
“The Cartoon Guide to Statistics”, by Larry Gonick and Woolcott Smith
http://www.amazon.com/Cartoon-Guide-Statistics-Larry-Gonick/dp/0062731025

And, introduction to JMeter: http://habrahabr.ru/post/140310/

THANK YOU

Marharyta Halamuzdava Marharyta_Halamuzdava@epam.com
Victoria Blinova Victoria_Blinova@epam.com
Aliaksandr Kavaliou Aliaksandr_Kavaliou@epam.com