

**Welcome. Experience. Quality. Passion.
Together. Beograd. Namics.**




**Coding Serbia.
Systematic Load Testing
of Web Applications.**

Jürg Stuker. CEO. Partner.


October 9, 2015

Nutrition Facts

Serving Size about 45 Minutes



| | % Daily Value* |
|---------------------------------------|----------------|
| Performance Tuning | 1% |
| Load Test Basics (Client View) | 25% |
| My Experience | 20% |
| Questions & Answers | 15% |
| Tool Demo | 39% |



No significant source of Business Blah Blah.

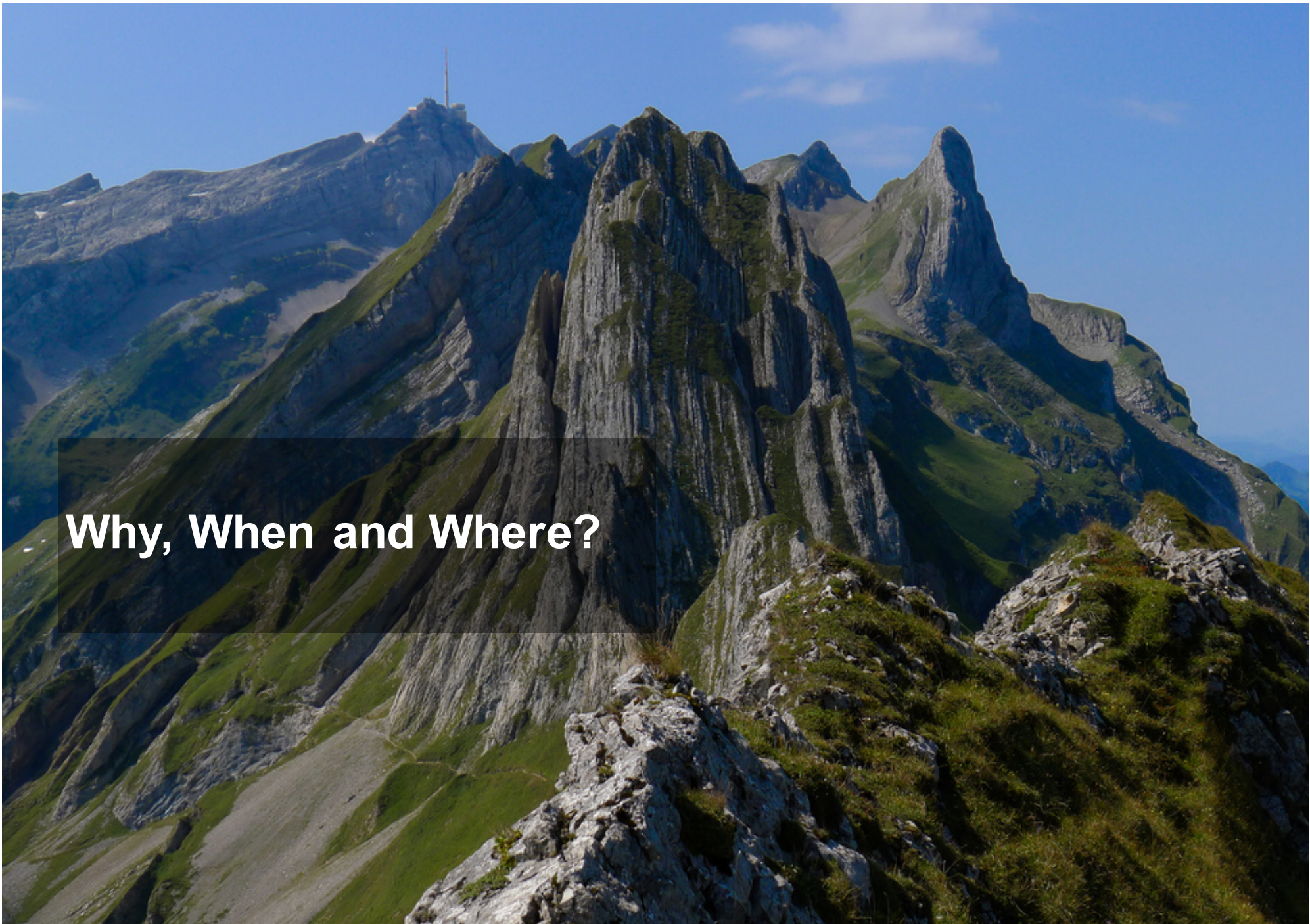
* Percent Daily Values are based on a tech diet.

Please shoot!

→ **The sum of the expertise of the people in the audience is greater than the sum of expertise of the people on stage, so please...**

#WebLoadTest

@jstuker



Why, When and Where?

Why should I test?

→ **Load capacity and performance**

- How many users can the application serve (in parallel)?
- How long does it take to deliver pages?
- How much bandwidth does the system use?

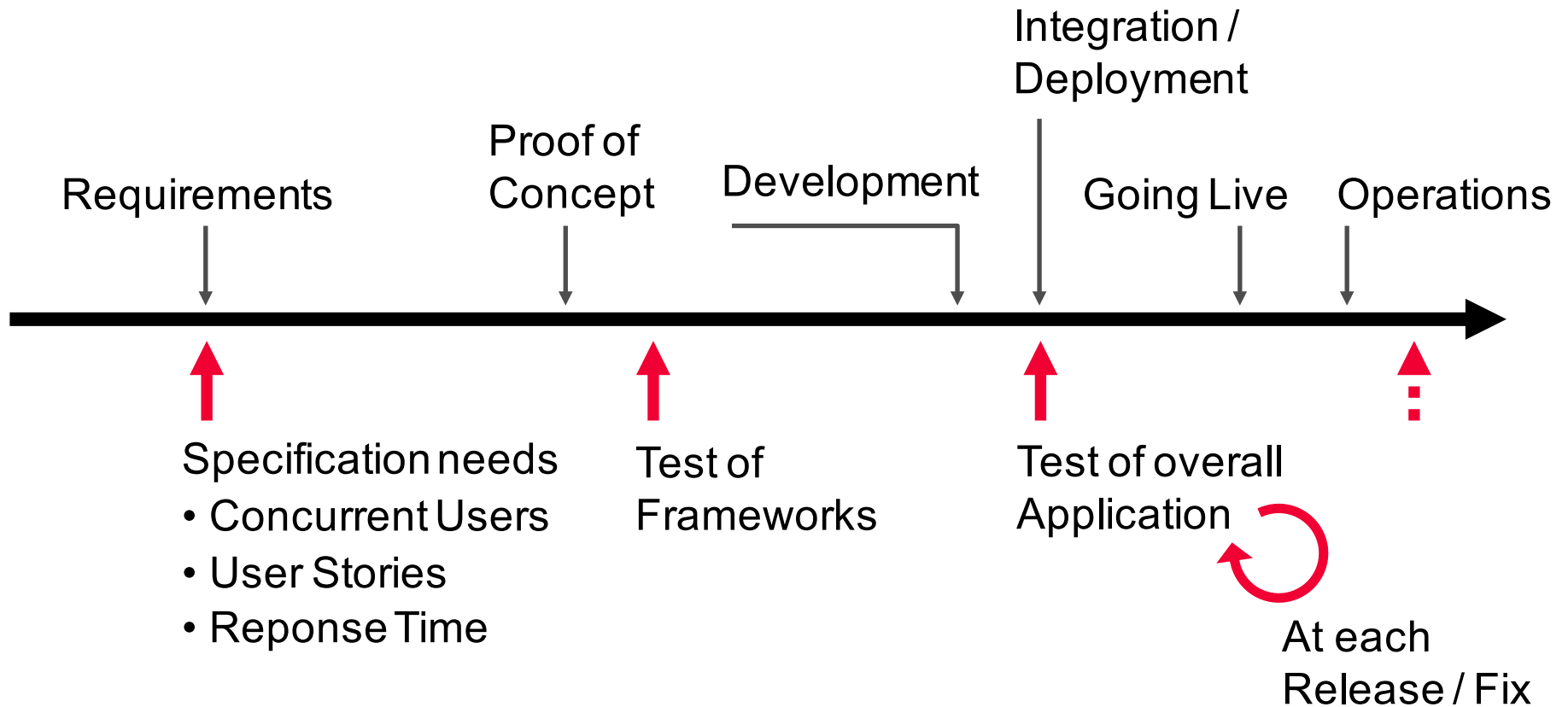
→ **Stability**

- Over time → Memory leaks & overflows
- Under load → Concurrency & deadlocks

→ **Fitness of Infrastructure**

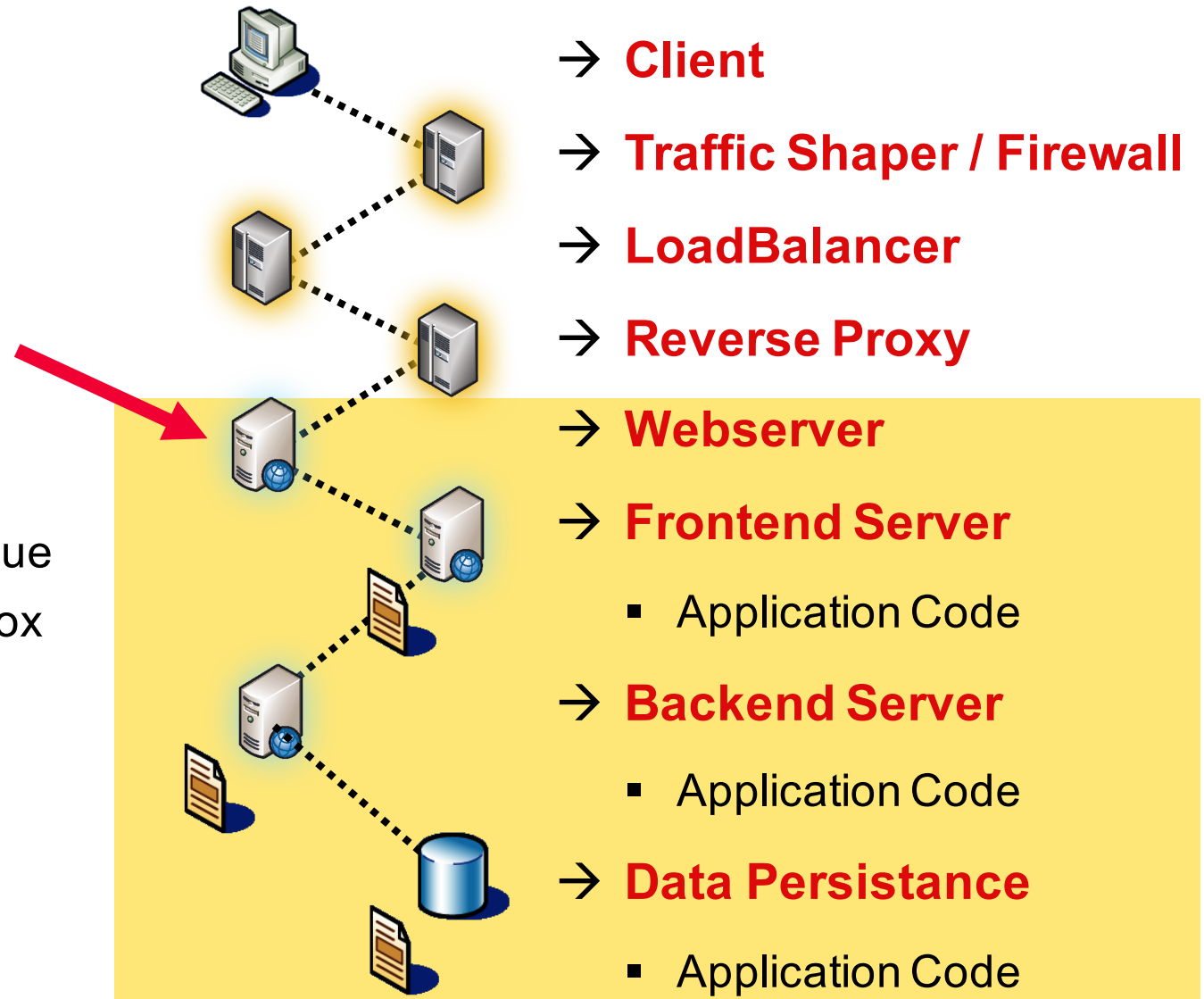
→ **Reproduction of problems in order to fix them**

When to think about testing?



Where to test?

1. Do not test the environment first
2. Treat application as a black box
3. When there is an issue
 - Look into black box
 - Change point of measurement





Very many... – what we use regularly (1/2)

→ **ApacheBench**

- Hammer on application
- <http://httpd.apache.org/docs/2.2/programs/ab.html>

→ **Apache JMeter**

- Loadtest workhorse, weak usability
- <http://jmeter.apache.org/>

→ **Gatling**

- Loadtest modern, easy to integrate
- <http://gatling.io/>

→ **Dynatrace**

- Application monitoring, white box, very powerful
 - <http://www.dynatrace.com/>
-

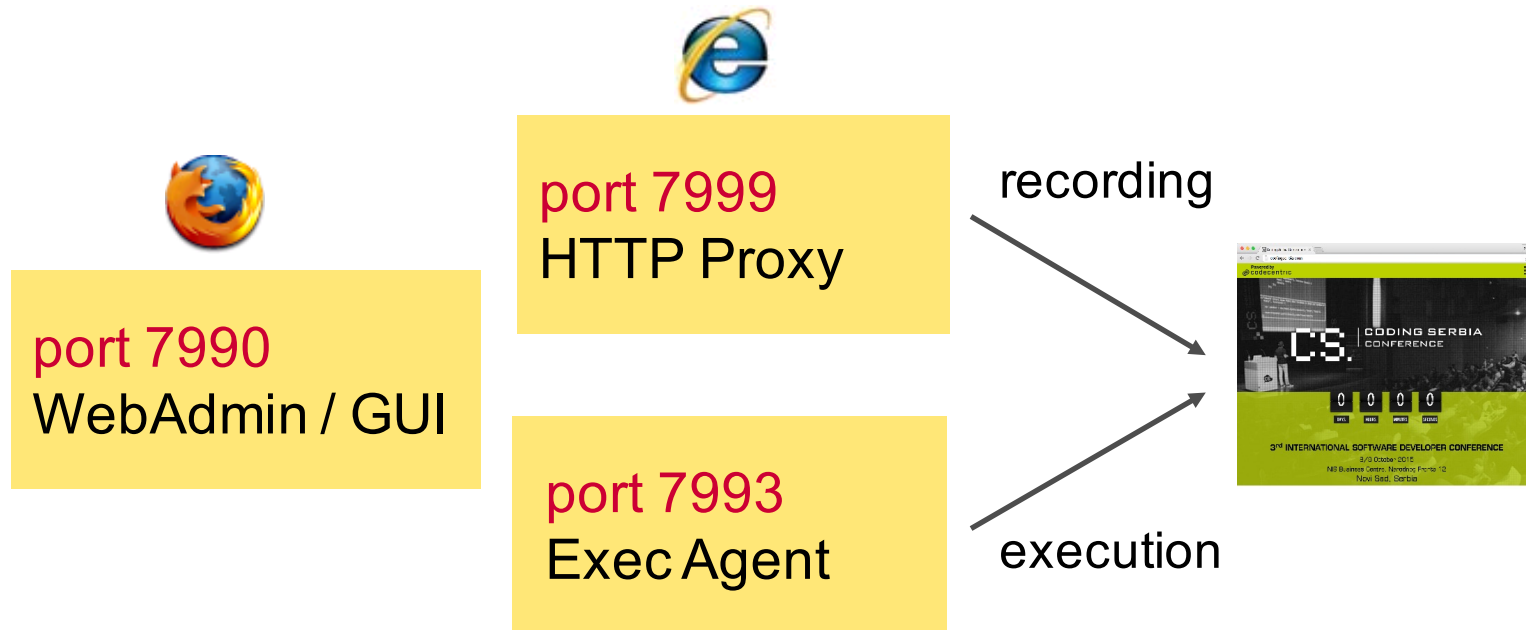
Very many... – what we use regularly (2/2)

→ **Proxy Sniffer**

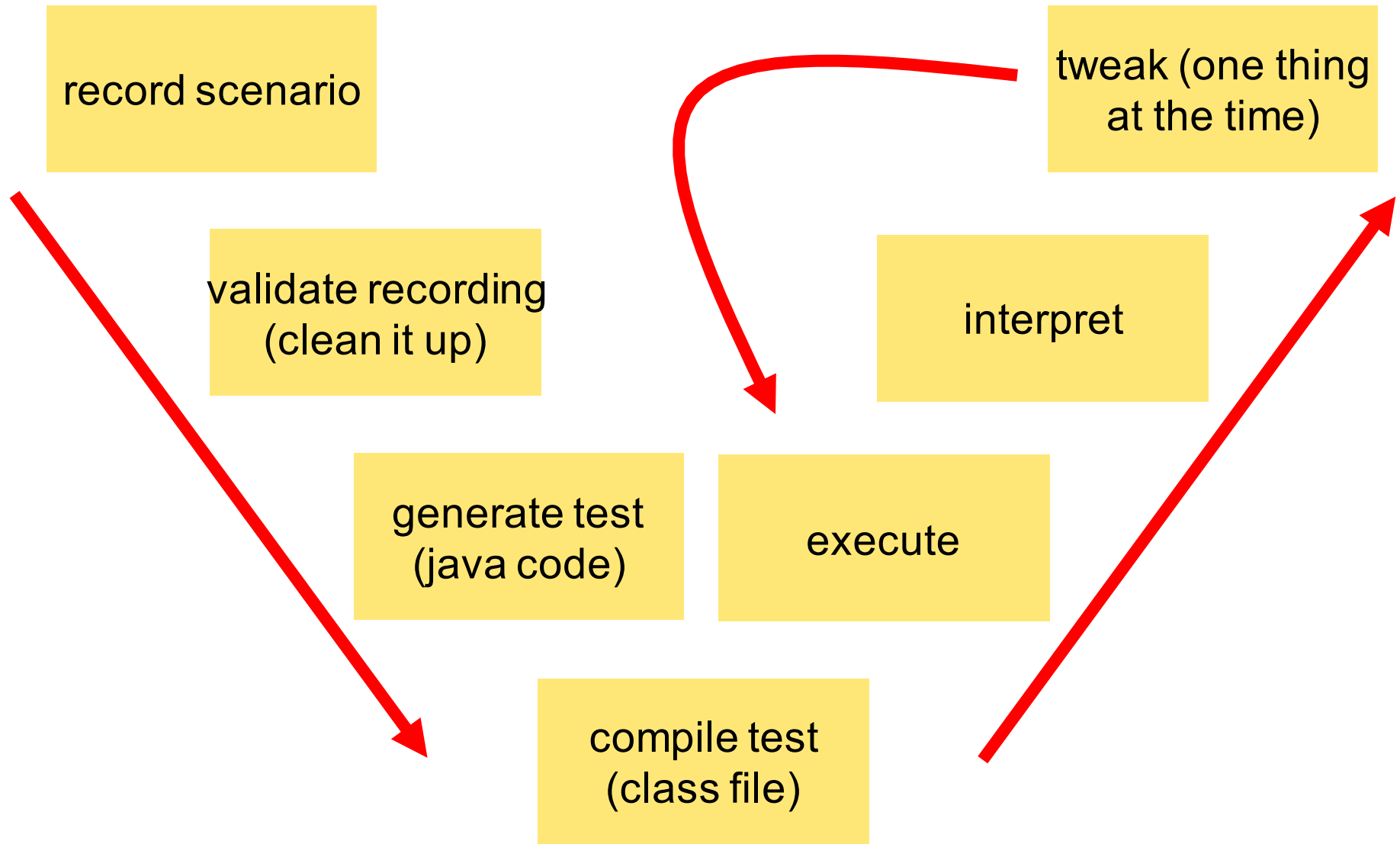
- <http://www.proxy-sniffer.com/> or <https://www.apicasystem.com/load-testing/tools/>
- Originally built in Switzerland by David Fischer, now owned by Apica Systems
- Free Version (20 virtual users, duration 10 minutes)
- Browser based, just needs > Java 1.5 (runs on Unix-like, Mac and Windows)
- Verrrrrry flexible (scriptable, cloud and small footprint)
- Nice documentation and reporting

Components of Proxy Sniffer Application

→ **Just two files: prxsniiff.jar and prxsniiff.key**



Sequence of a Test with Proxy Sniffer





Let's do some work

Let's do some work...

A word of caution

Load testing generates load ;-)

Let's do some work...

Live Demo or Recording ;)

→ https://youtu.be/0Akj5qw8_So

The screenshot shows a YouTube video player displaying a live demo of a load testing tool. The video content is a screenshot of a 'Project Navigator' window titled 'PRJ: Analyse Load Tests - Load Curves'. The window shows several performance graphs and a network log.

Overall Load Curves: Response Time per Page (Average, 90% Percentile), Response Time per URL (Average, 90% Percentile), Errors.

Response Time Behaviour - Average Session Time per User and Loop: Graph showing session time in seconds vs concurrent user. Data points: (1, 35.100), (2, 35.100), (3, 35.100), (4, 35.100), (5, 35.100).

Server Performance - Web Transaction Rate: Graph showing hits per second vs concurrent user. Data points: (1, 1.5), (2, 1.5), (3, 1.5), (4, 1.5), (5, 1.5).

Stability - Session Failure Rate: Graph showing percentage of failed sessions vs concurrent user. Data points: (1, 0.00), (2, 0.00), (3, 0.00), (4, 0.00), (5, 0.00).

Network Influence - Average TCP Socket Connect Time: Graph showing time to establish a TCP network connection in milliseconds vs concurrent user. Data points: (1, 133), (2, 133), (3, 133), (4, 133), (5, 133).

Users Waiting for Response (Outstanding Requests): Graph showing number of users waiting for response from the web server vs concurrent user. Data points: (1, 0), (2, 0), (3, 0), (4, 0), (5, 0).

URL Error Rate: Graph showing percentage of failed up calls vs concurrent user. Data points: (1, 0.00), (2, 0.00), (3, 0.00), (4, 0.00), (5, 0.00).

Network Log: Shows HTTP requests and responses. Example: GET http://codingserbia.com/ → 200 (OK) TEXT/HTML. Total: 2 Requests, 12.84 bytes/sec.

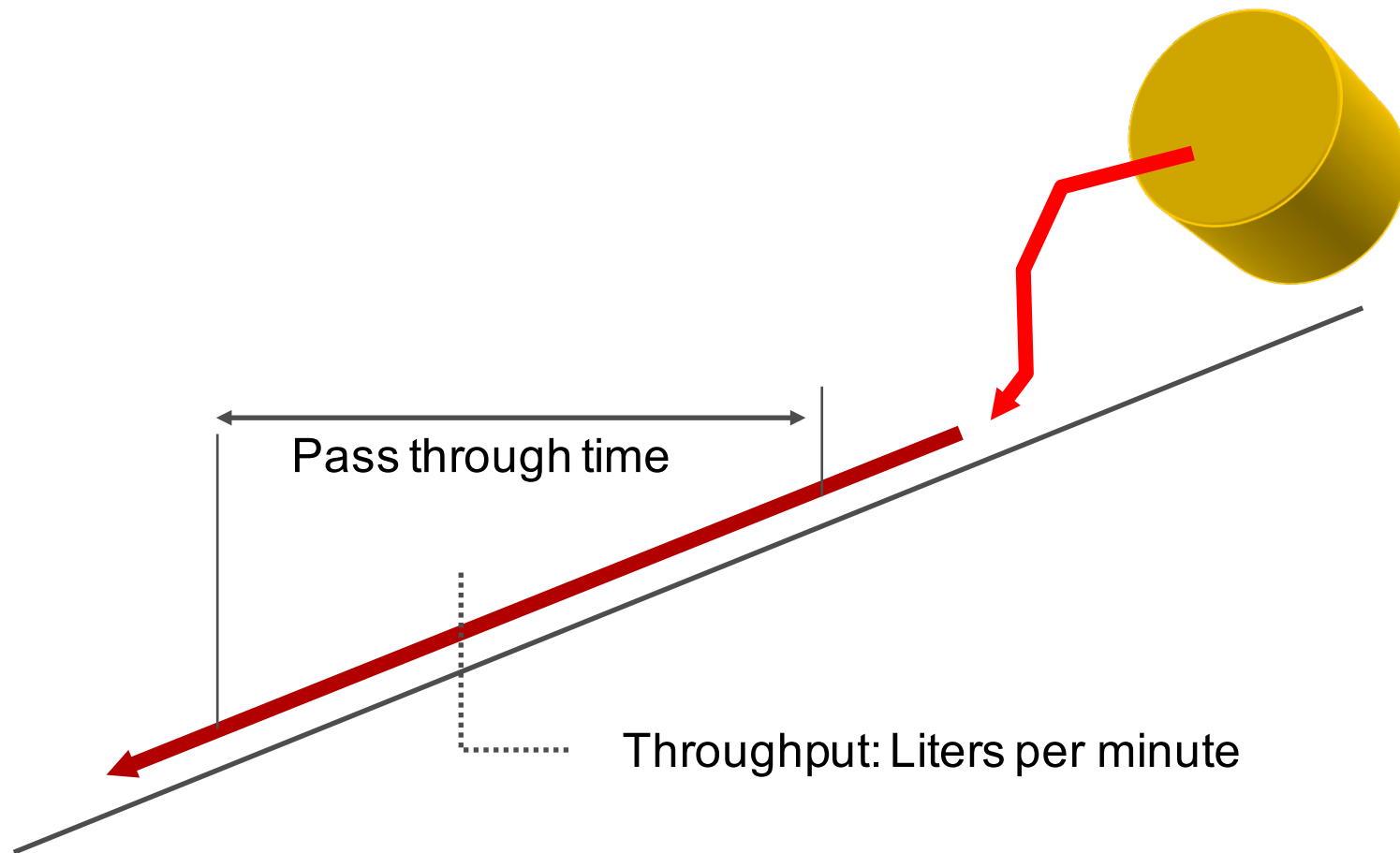
The video player interface includes a 'Watch Later' button, a progress bar at 10:57 / 12:45, and a 'Up next' section showing 'Alfresco Tech Talk Live 85: 2015 Global Virtual Hack-a-thon'.



Interpreting the Outcome

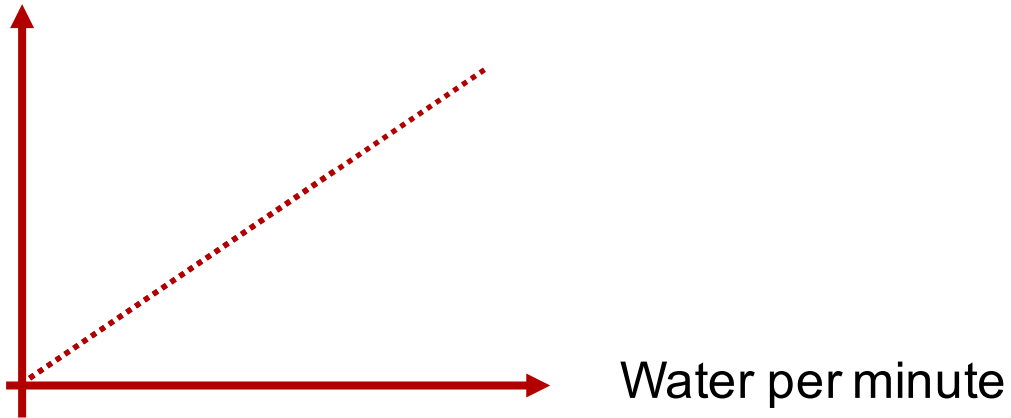
Linear Water Channel

- **More water == more throughput**
- **Pass through time is (close to) constant**

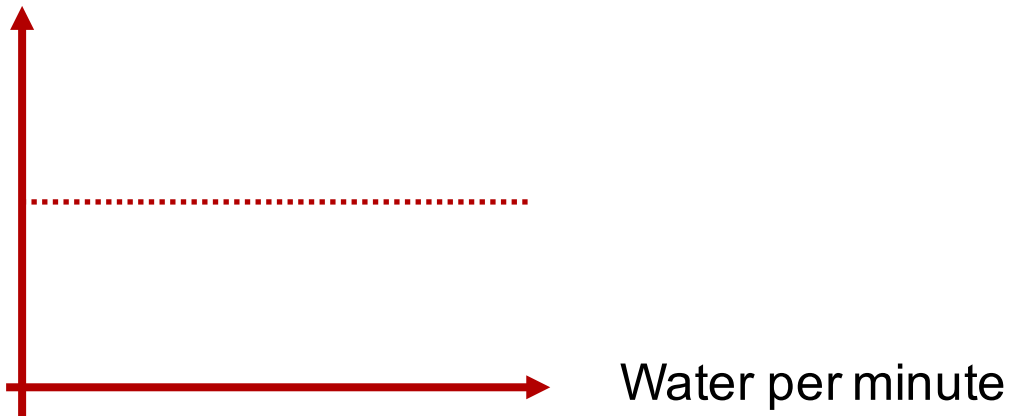


Unlimited Linear Model

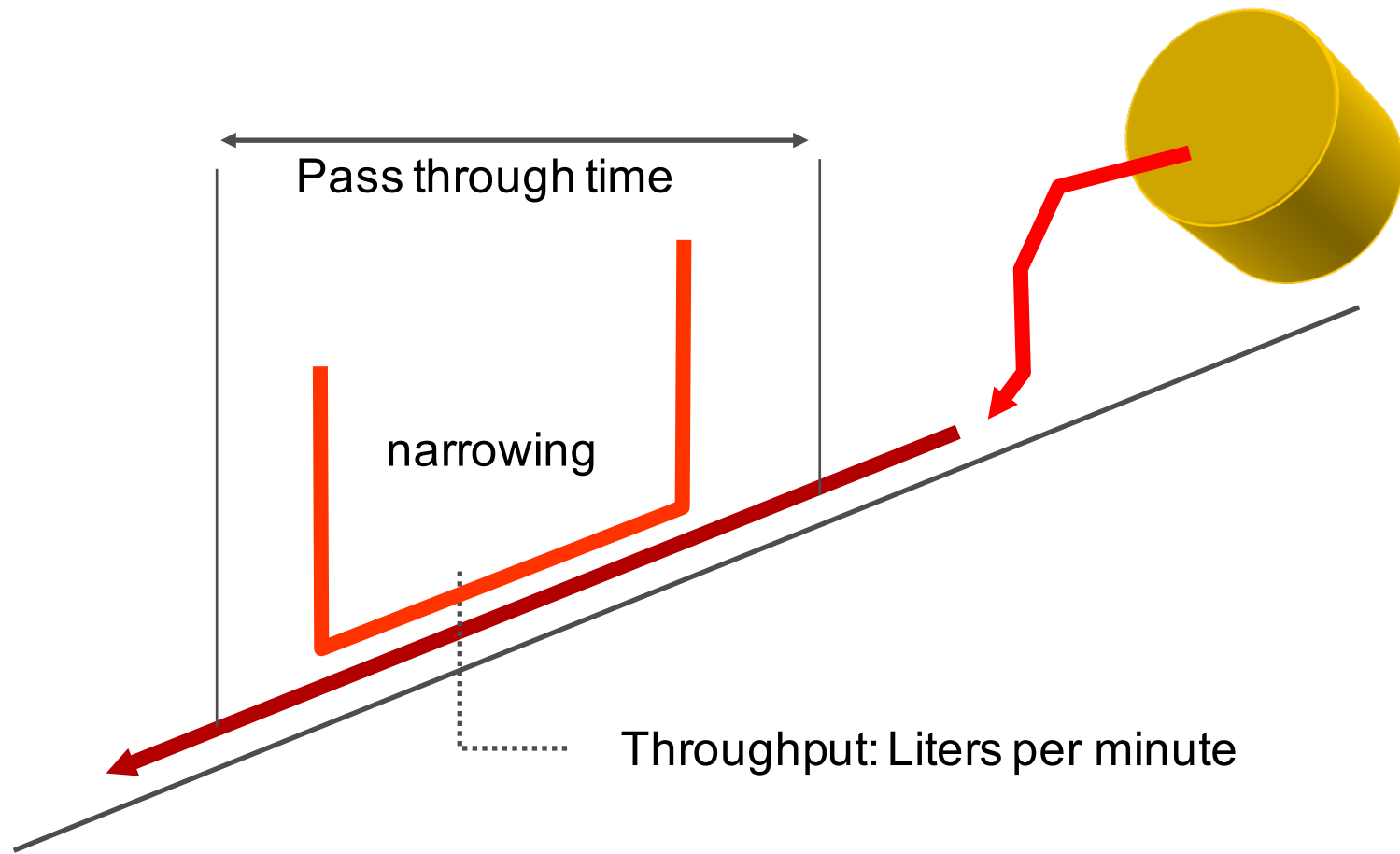
Throughput:
Liters per minute



Pass through time

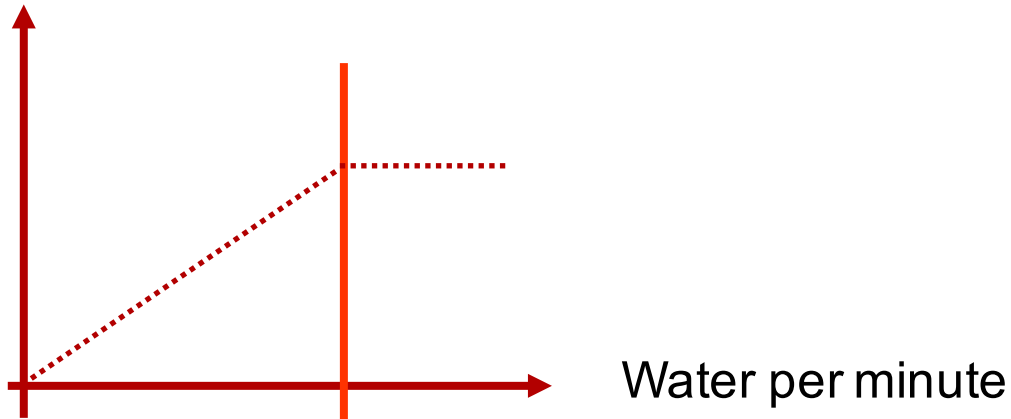


Limited Water Channel

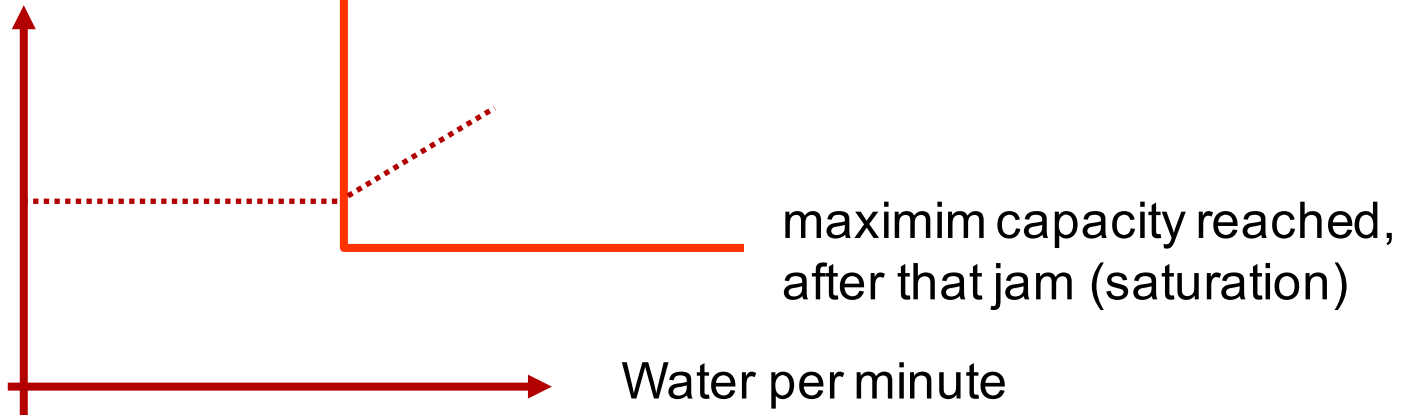


Limited Linear Model

Throughput:
Liters per minute

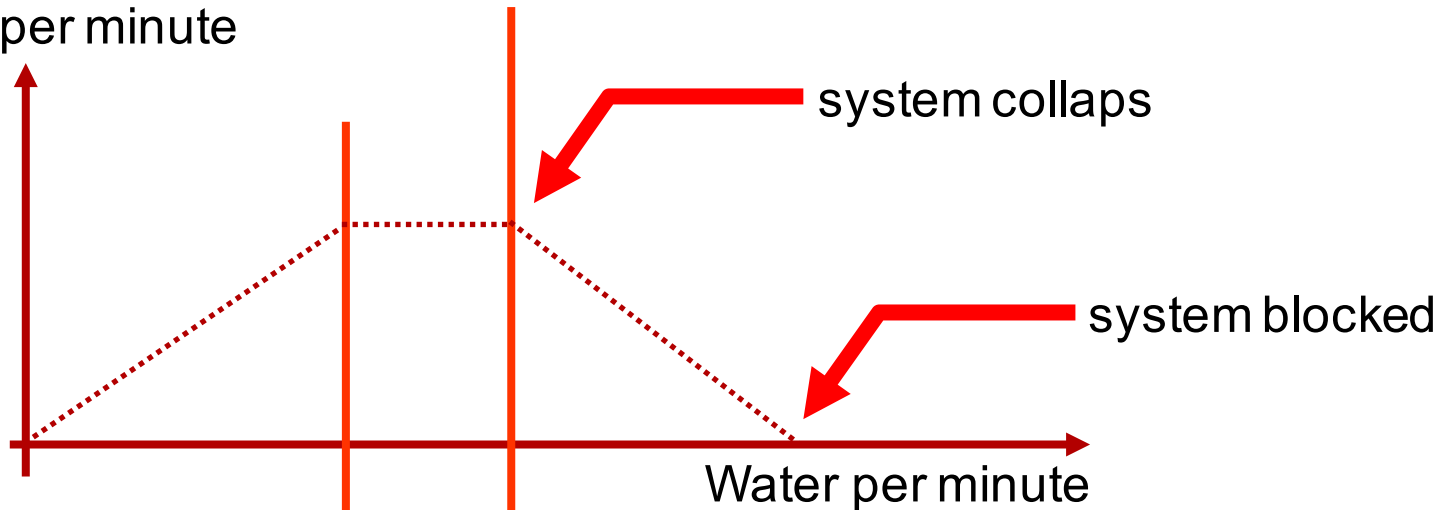


Pass through time

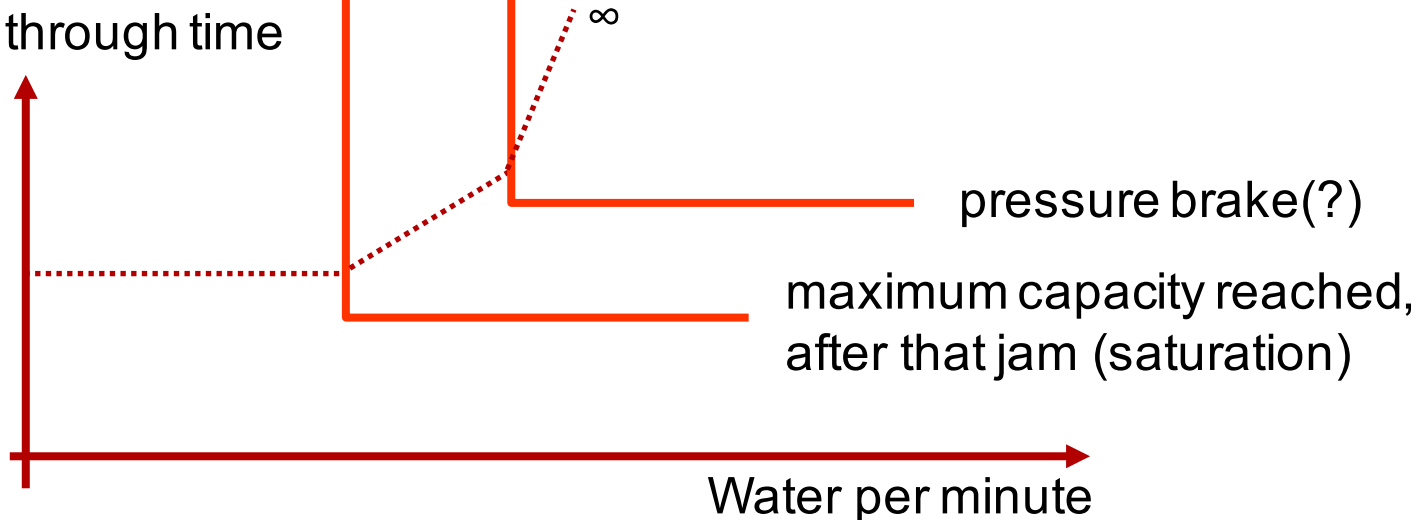


Non Linear System

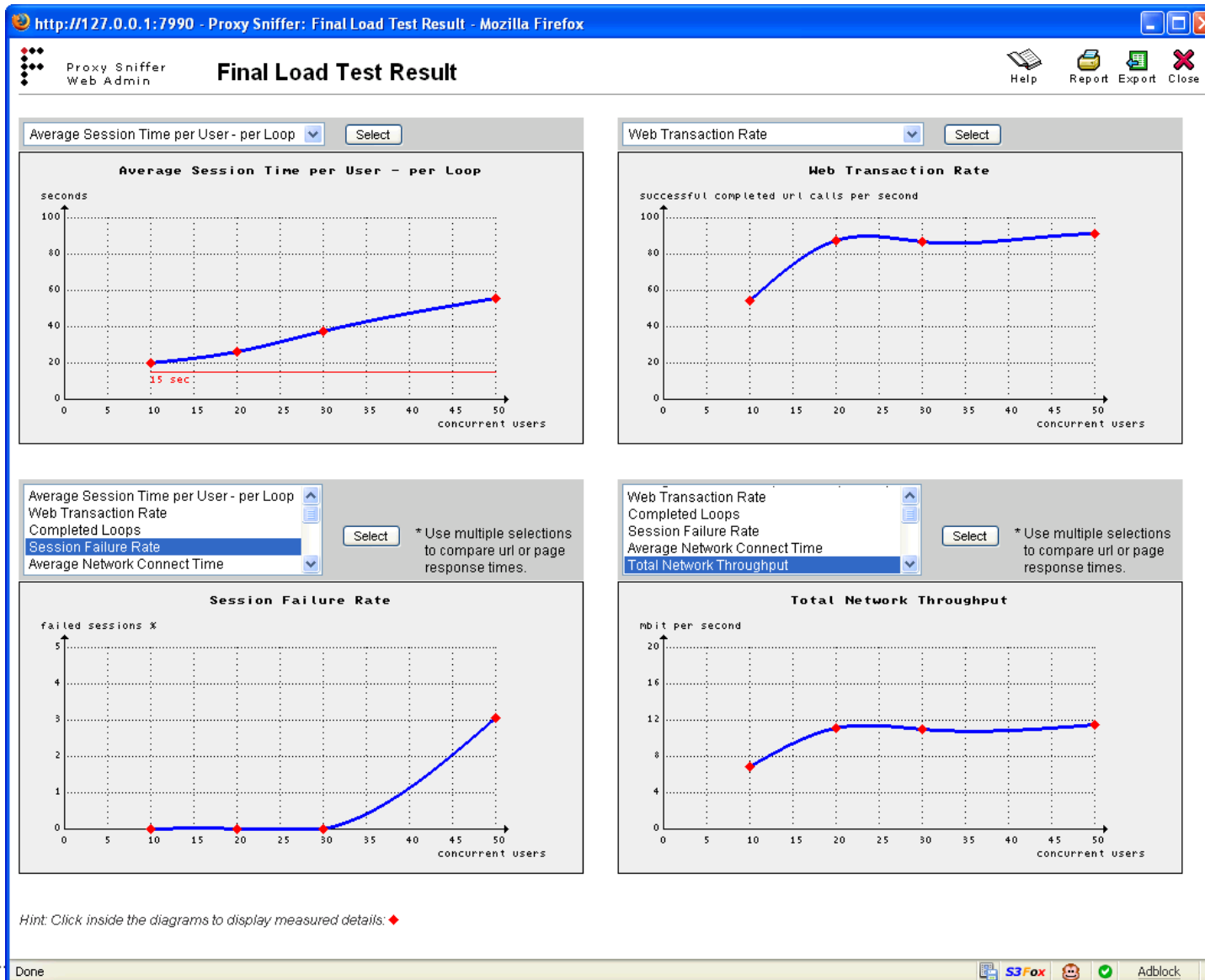
Throughput:
Liters per minute



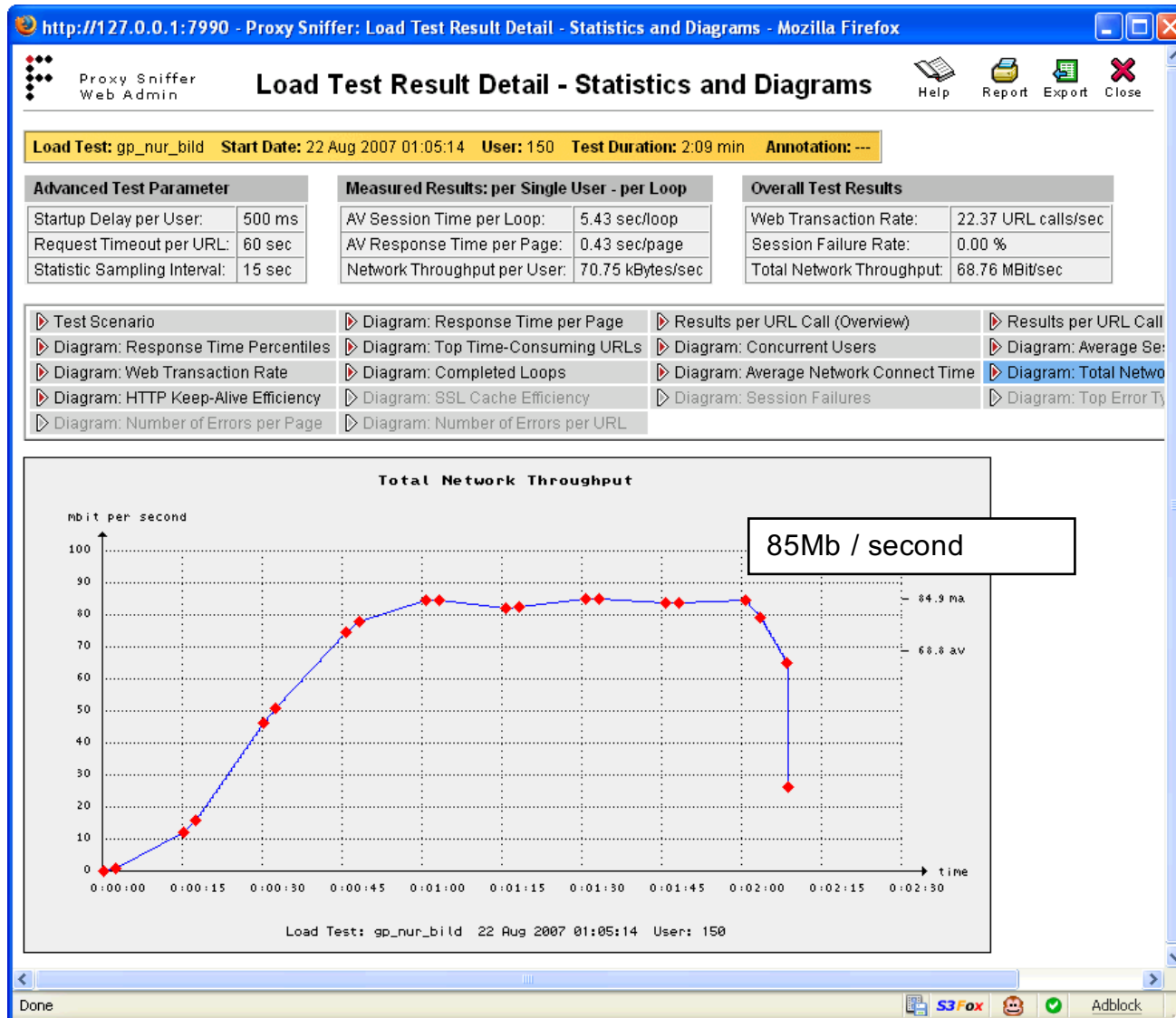
Pass through time



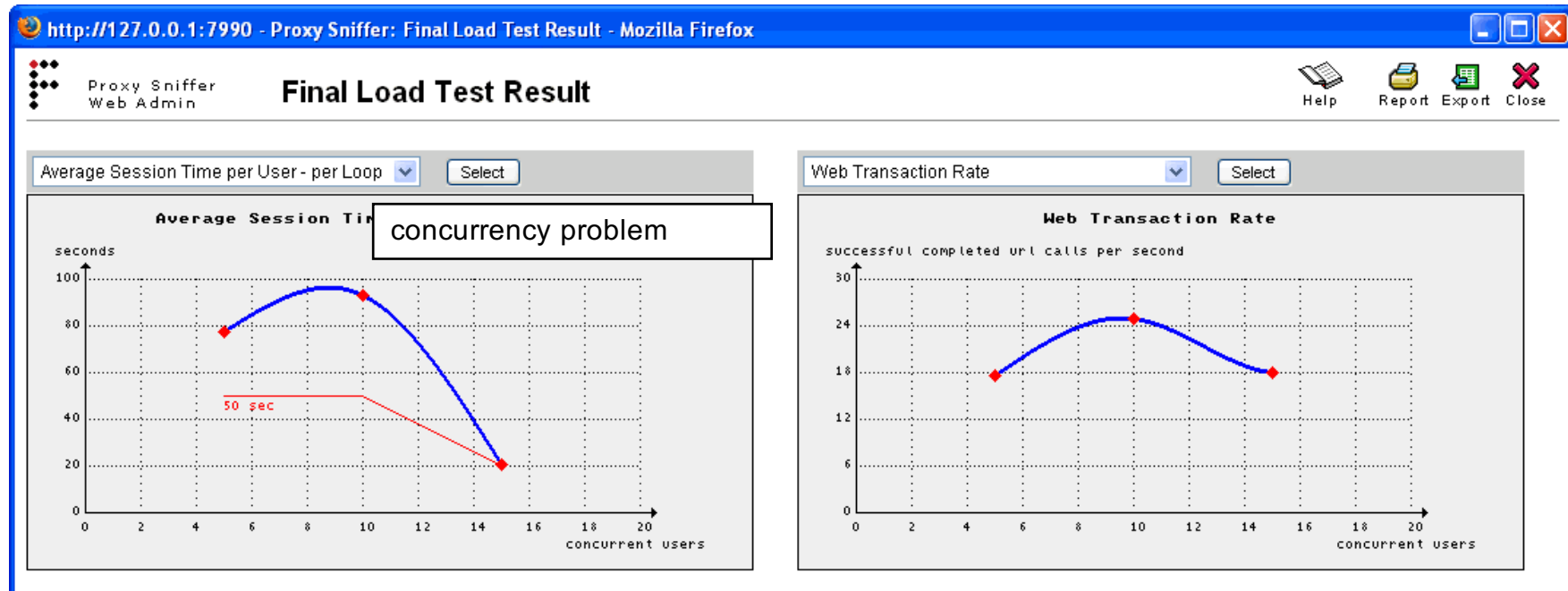
Example of test results (1 of 4)



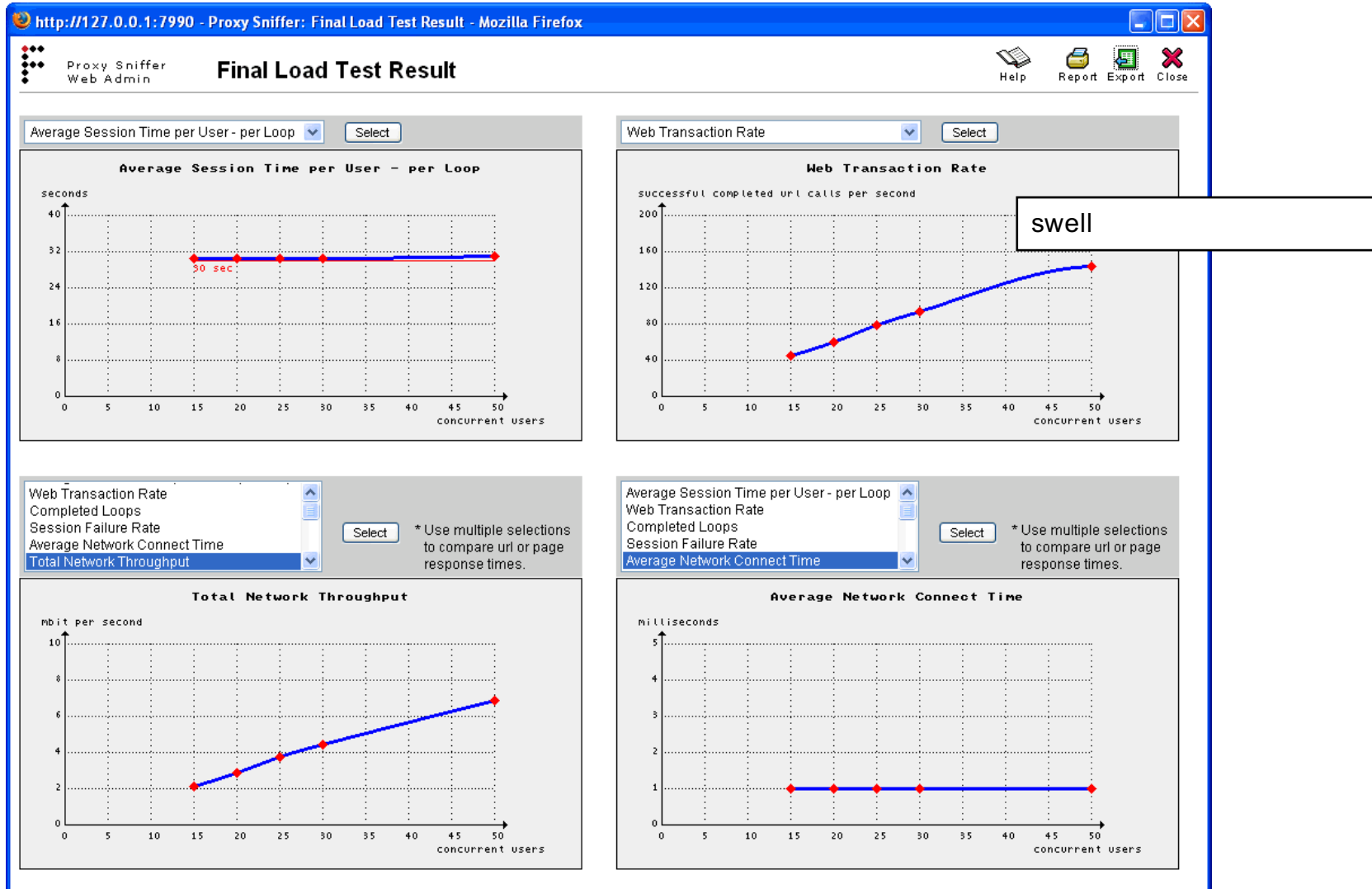
Example of test results (2 of 4)



Example of test results (3 of 4)



Example of test results (4 of 4)



Tipps for executing the test

→ **To find the correct scenario is really hard**

- Look at historic logfiles / stats (plus safety margin)
- Ask domain experts
- Take the user stories (“follow the money”)

→ **What to go for**

- Bandwith → Fetch one large item (again and again)
- # Transactions → Short test cycles with 5, 10, 20, 40... users to find out where the systems levels (or decreases or crashes)
- Long runner → Execute a slow test during hours

→ **What's nice to know too**

- Start system under load
 - Stop system under load
 - Degradation (i.e. cluster node shutdown)
-

Watch out for

→ **Realistic scenario**

- request mix
- think time
- read vs write (POST requests)
- caching

→ **Get 100% CPU of the tested system first**

→ **Look at the database (slow queries)**

→ **Look at the HTTP response content/headers of the test**

→ **Did the system that executes the test deliver the load?**

→ **Error logs**

The usual suspects

| | |
|------------------------------------|------|
| → Network (Pipe, Router, Firewall) | < 1% |
| → Load Balancer | 5 % |
| → Reverse Proxy (Product) | 5 % |
| → Reverse Proxy (Configuration) | 10 % |
| → OS configuration (TCP/IP Stack) | 5 % |
| → Application Framework | 10 % |
| → Configuration of Application | 20 % |
| → Programming/Code of Application | 40 % |
| → Database | 5% |



Summary

- **Load testing MUST be part of every project plan**
- **Product owner must define what's needed**
- **Do not test/optimize performance early in the dev cycle**
- **Test as close to the application as you can get**
 - and only when you find issues → change view
- **Percentile statistics are OK (90% of requests fit)**
- **Scenarios are though to get**
- **Validate test results....**

Homework

→ **Read the HTTP Specification**

- <https://tools.ietf.org/html/rfc7230>



About Namics

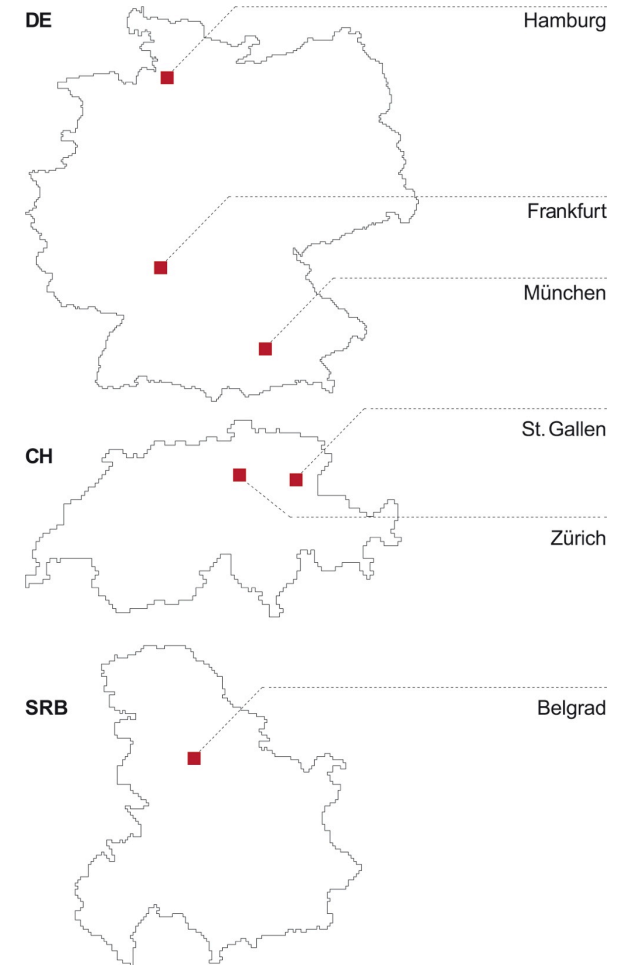
Who is Namics?

Focus

- High-end e-business services for large, global customers
- Leading also in complex large-scale projects

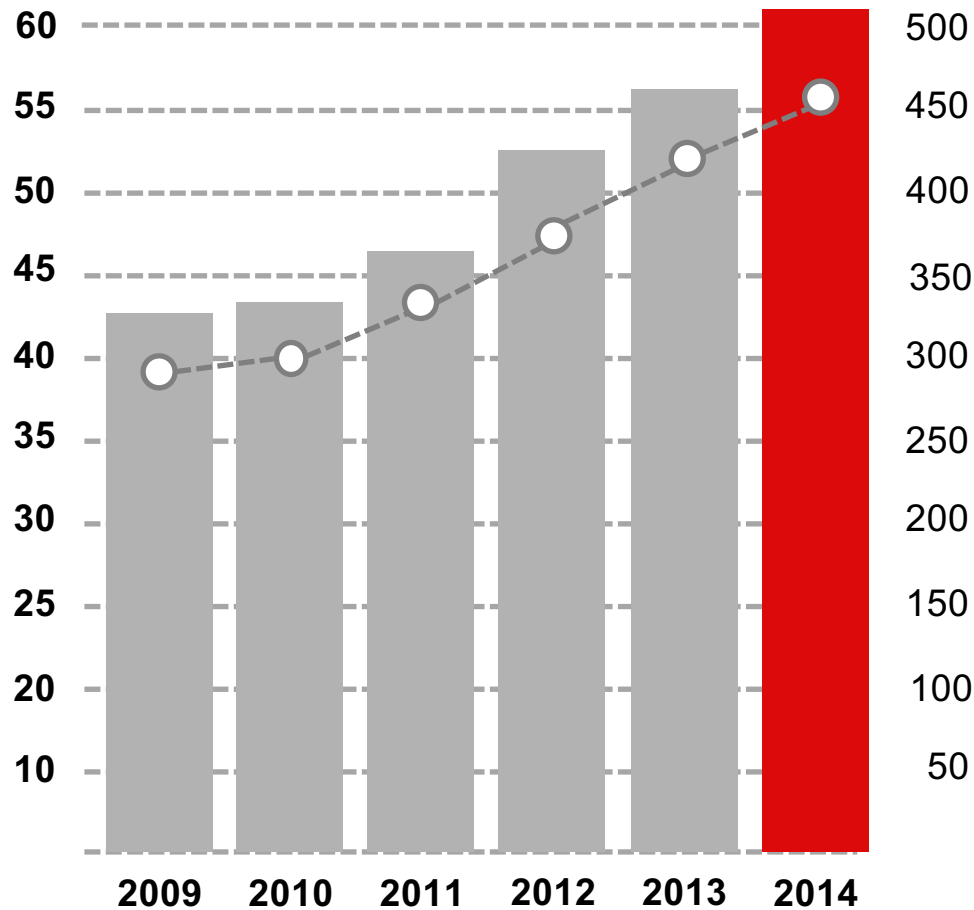
Stability

- Long-term customer relationships and stable organisation
- Owned by a group of 29 employees (Namics Partners)
- Company operating since 1995



Namics figures in a Nutshell

Revenue (in Mio. CHF)
and number of employees



→ **Quality as growth engine**

- Best of Swiss Web agency in 2013 & 2014 (Best of Swiss Web Association)
- Proven track record

→ **Stable relationship with our key clients**

- More than 70% of our clients stay with us for 3+ years

Namics in Serbia

Highlights

- newest location in Europe
- cross country teams, interdisciplinary

Belgrade in a nutshell

- start-up culture
- new 480m2 office
- collaboration
- Awesome since 1995.
Now in Serbia. Namics.



We are looking for new talent in Belgrade!



→ <http://namics.com/jobs>

Welcome. Experience. Quality. Passion.
Together. Beograd. Namics.



Download-Link:
<http://nam.to/webloadtest>

jurg.stuker@namics.com
[@jstuker](https://twitter.com/jstuker)