ebay classifieds group

LnP Practice in Nexus

Forest Xie
Nexus International Team

Agenda

- QE Team Brief Introduction
- LnP Phase
- LnP Structure
- Challenge
- Q & A

Automation

What we did?

Maui | PikeMan | SoapUI

What are we doing?

QA Lab | RUI | BDD

What we want to achieve?

High Coverage → Stability → Efficiency

Dev OPS

CI – Availability and High Performance

- Build Pipeline
- Add deployment scripts for fp or staging

Tool – Smoothly & automatically

- Create tools to detect potential issue
- Migrate data between production and QA

Support – Round the Clock

- Fix the build and deployment issue
- Effective communication cross team and geography.

LnP

Are you Ready?

What is LnP

- Words come into my mind
  - Fast or Slow
  - Healthy or Crash under load
  - Memory leak/ Deadlock?
  - JMeter or Grinder
  - Response time

- Using Pareto Principle (80-20 rule) all the time
  - **LnP testing can't cover everything but you can try to achieve 80%**
    - Test cases/ test scenarios
    - Test metrics
- LnP testing should be a collaboration work
  - PM
  - PD/OPS
  - QE/Platform QA

Difference

From QA perspective(Find what):

- Is it meaningful for giving such a load on the system?
- Where is slowness of the application?
- What kind of tool should I look for?
- Id there a performance issue in backend service?
- …

From Dev perspective(Find how):

- Why our application or that action is so slow?
- How can I make it faster?
- How can I avoid memory leak next time?
- What kind of data should being cached, what is isolation level right now?
- …

When Start LnP

1. Does the feature affect any major business flows, services or daemons? E.g., send money, sign up，login, checkout, pimp or crypto
2. Does the feature add or change any queries?
3. Does the feature require a new database on live？
4. Does the feature add any new tables?

What we will do in LnP

- Performance testing:
  - Run scripts using small number of workload and little sleep time
  - Start doing performance testing from API level even there is no UI
- Load testing:
  - Get the main idea of application performance grade
  - With anticipated real-world load/pre-defined requirement
  - Run scripts using certain number of workload and normal sleep time

What we will do in LnP

- Stress testing:
  - Find out how the application handles overload and servers' break point
  - Adding workload until the load exceed the anticipate workload
  - Should simulate the peak load situation, like holidays etc.
- Stability testing:
  - 24 hours even longer continuously

LnP life cycle

- Planning
- Develop scripts
- Execute

LnP Planning Phase

Understand application in depth

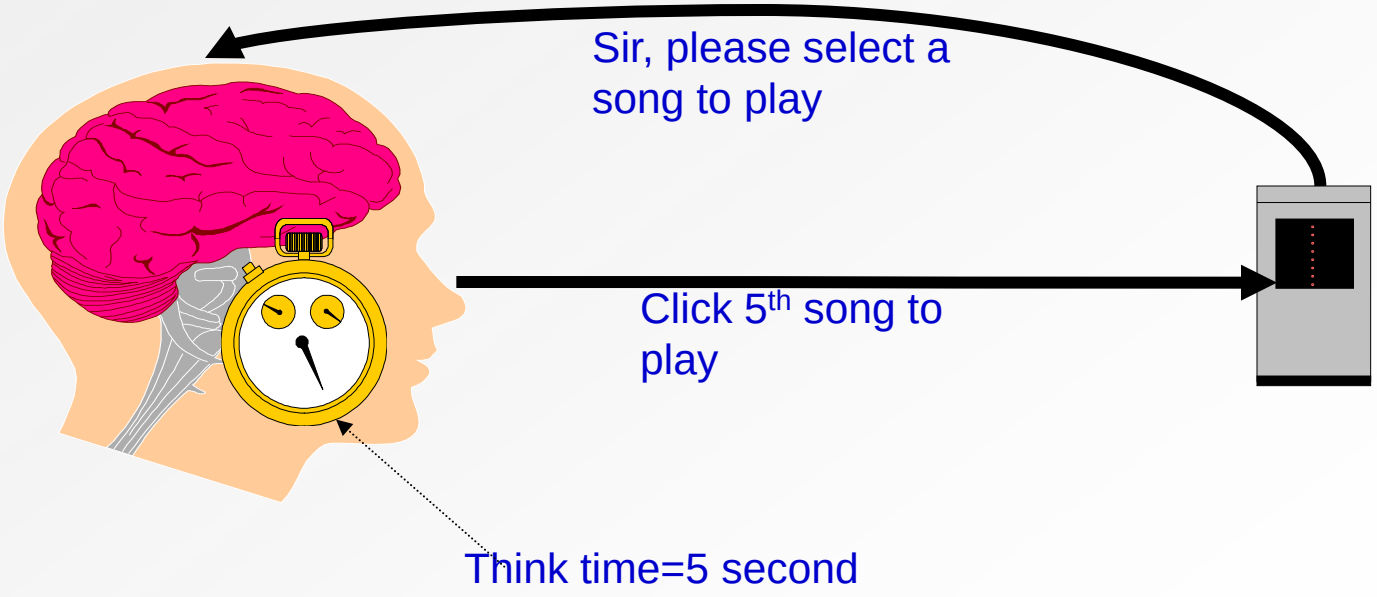
- Which actions are mission critical (post ad, registration)
- Which actions use most system resources (DB call)
- Which action may meet heavy throughput in a short time (login)
- Some concerns from dev or architecture, need QE engineer to verify
- Learn user distribution from log analysis or common sense

Test cases and test scenario design with PD and PM

- Test cases and scenario design based on the principles above
- Use simple diagram to show the whole user modeling

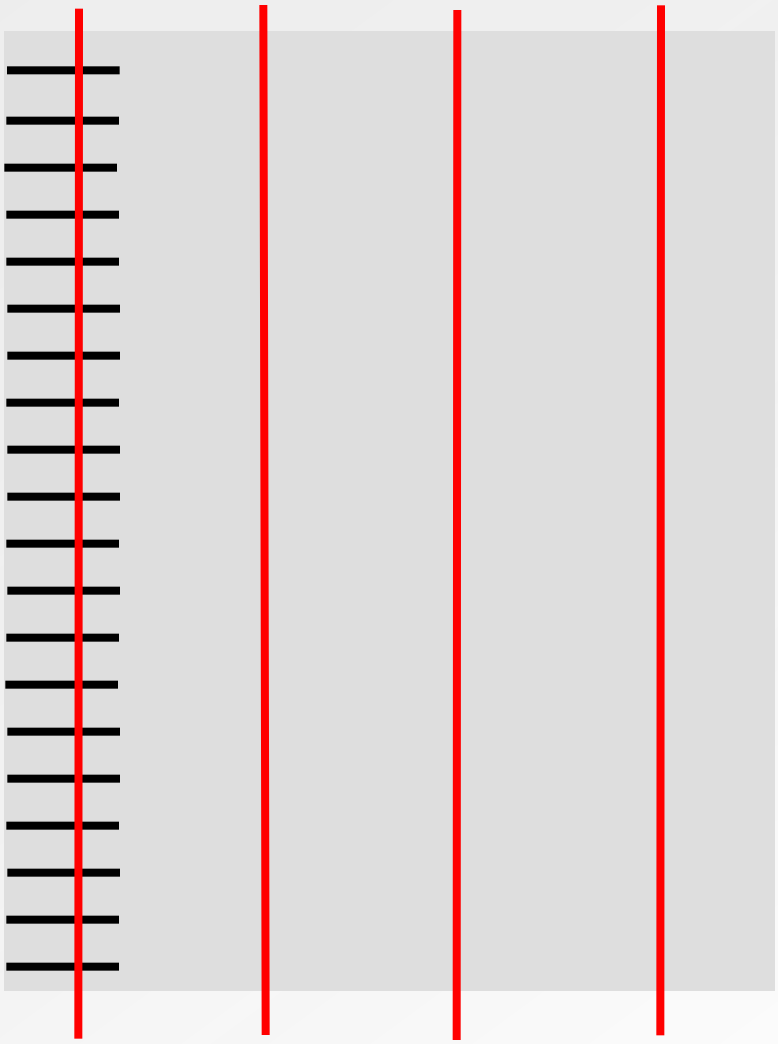
Real user VS. Virtual User

- The time it takes a client (or virtual client) to respond to an application may have a significant impact on the number of clients an application can support
  - This is a key aspect of concurrency

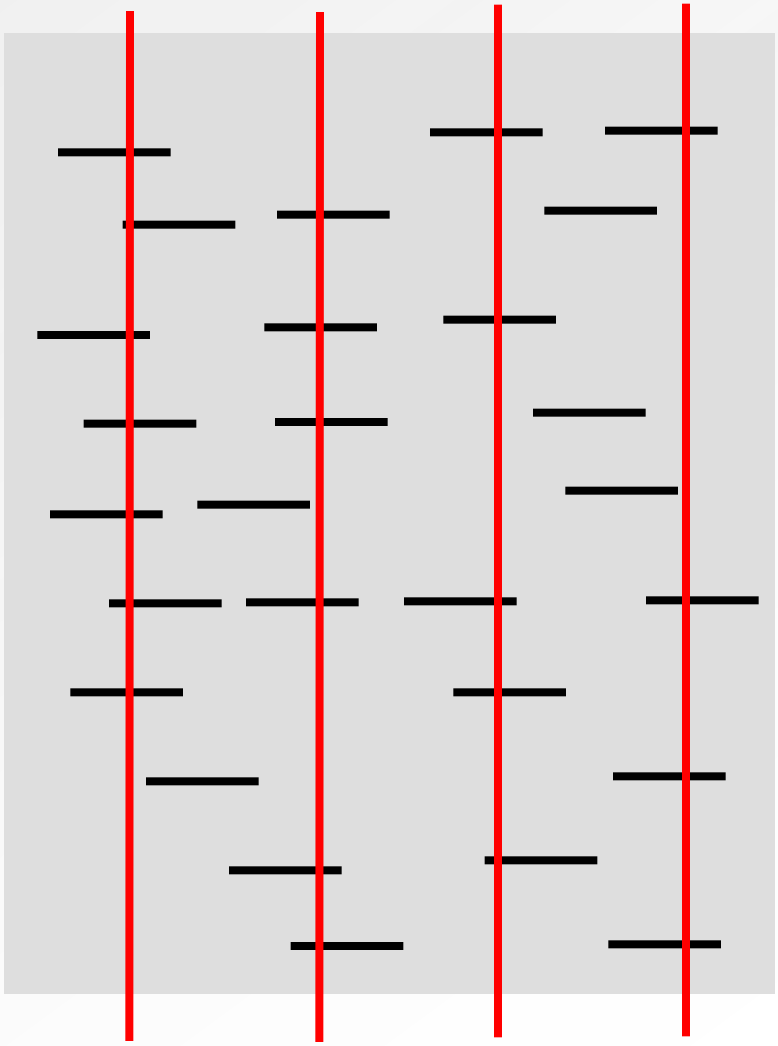


Concurrency

action1



action1



LnP Planning Phase

- Identify Performance metrics:
  - Response time
    - Average, Min, Max, 90% percentile, std deviation
  - Throughput
    - Requests per second in JMeter
  - Resource utilization monitoring
    - CPU%, Memory usage, Disk I/O, JVM memory Heap, Top threads
- Identify Performance criteria:
  - User load (50 concurrent users with proper sleep time)
  - Response time (3s from end to end within local network)
  - Throughput (15 hits per sec in average or create 500 RFPs per hour)
  - Error rate (Error rate should under 5% during 10 hours testing)
  - Resource utilization rate (CPU% <80% under peak load)

LnP Scripting phase

- Using Jmeter/Grinder
  - Record settings
  - Quickly record and parameterize the scripts
  - Create a few user groups to simulate user distribution or work flows
  - Run-time settings
  - Schedule the testing
  - Result data collection/result analysis

- BTW, Jmeter/nGrinder is not browser
  - It can not calculate js execute and page render time
  - Need another tool to measure our front-end performance Grade (Harstorage)
  - Using Selenium + BrowserMob

LnP Executing Phase

Executing does not mean only to click “Run” button:

- Application deployment is ready (functional test pass under right version)
- Check Server is still alive during testing
- DB size should be realistic by Data preparation or using production data
- Eliminate the network factor first by using local network environment
- Manually go through the test scenario just after server start-up
- Set proper rump-up time, not all users at once
- Schedule each round of testing a certain duration/loops
- Each test should be executed several times, till we find some sort of statistical consistency
- Make a matrix table to record all performance data “in daily”, so that you can make it comparable and easy to track
- Iterative way of testing
  - ✓ split into several **modules** first and then integrate, so more focus each time
  - ✓ **adjust load** dynamically based on your report

monitoring and profiling

- "Response time" is not whole story
  - Monitoring (usually in testing phase):
    - OS level resource utilization
    - Web/App/DB servers, load generators
    - Administrative tool->Performance or Process explorer
  - Profiling (usually in tuning phase):
    - JVM Heap memory usage, top threads, classes loaded, deadlock, GC
    - Methods execution time
    - Object reference
    - SQL statements
    - Open source solution: JConsole + profiler log + SQL profiler
    - Commercial integration profiling tool :Jprofiler or Yourkit java profiler

LnP tool we used

Performance testing

- Jmeter http://jmeter.apache.org/
- Jmeter plugin http://code.google.com/p/jmeter-plugins/

Stress/Load Testing

- Jmeter
- Siege 3.0 http://www.joedog.org/siege-home/

Stability Testing

- nGrinder http://www.nhnopensource.org/ngrinder/

FrontEnd Testing

- HarStorage http://code.google.com/p/harstorage/
- YSlow + PhantomJS http://phantomjs.org/
- Pcapperf http://pcapperf.appspot.com/

Calculation model

If our site need support 5M PV/ per day.

The request number per second= ((80% * Total PV Volume) /(24 *60 * 60seconds *40%)).

One of the key parameter are 80% and 40%. Represents 80% of the requests in a day occur within 40% of the time in a day.

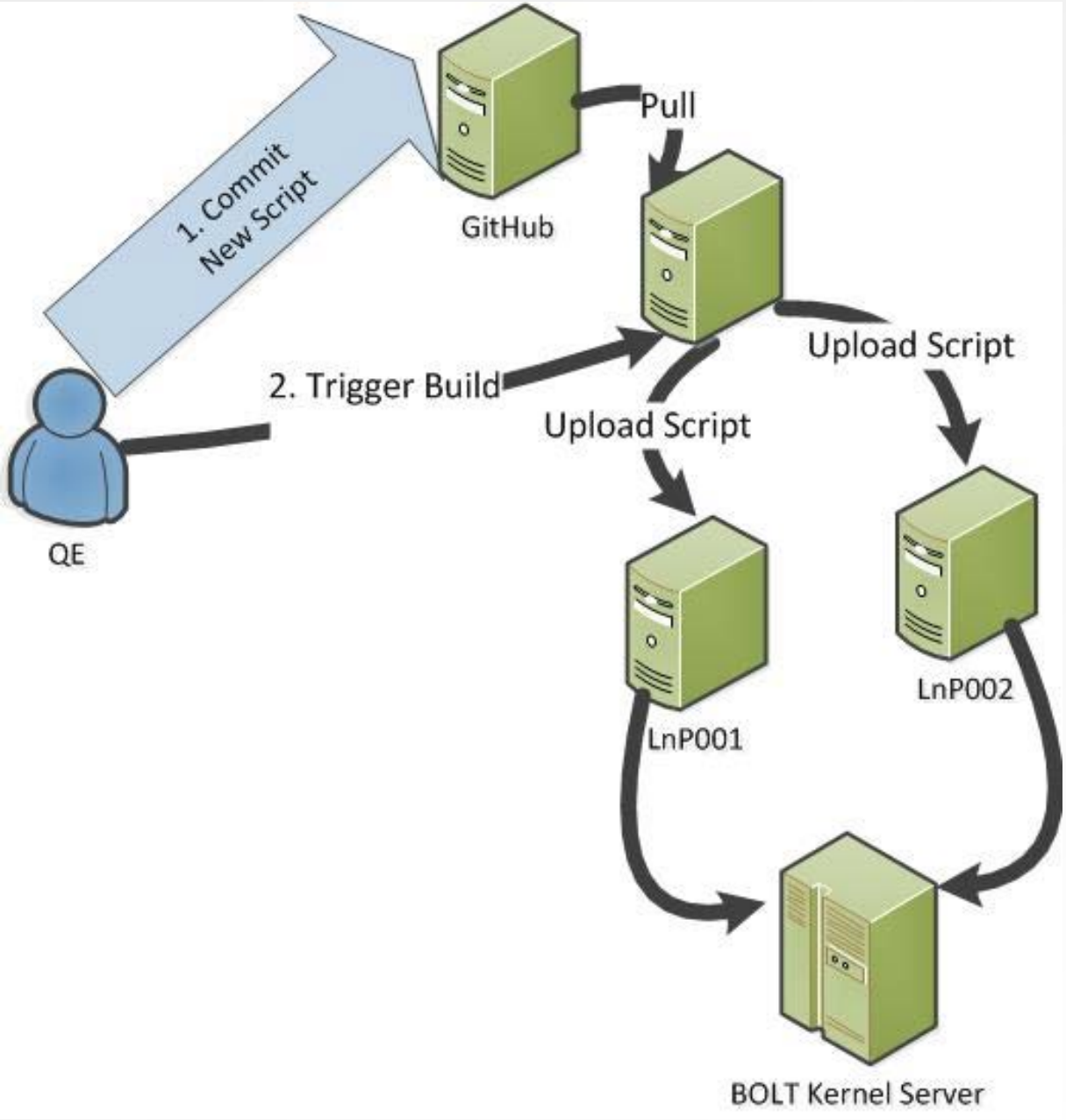
40% of the 24-hour 9.6 hours, **80%** of the requests are occurred in 9.6 hours a day

The result of a simple calculation:

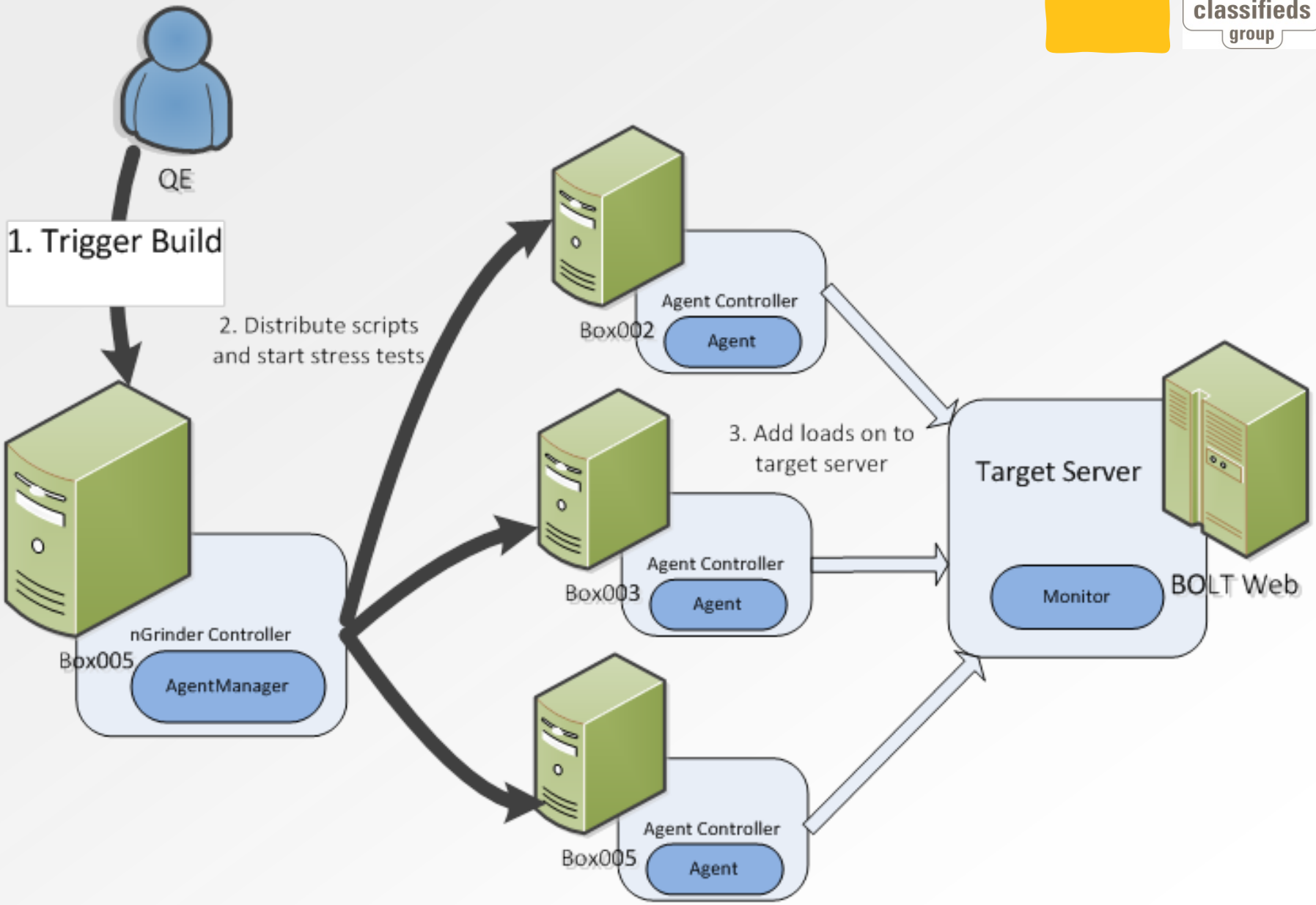
((80% * 500000) / (24 hours* 60 * 60 seconds* 40%)) / 1 =**115.7** requests / sec

((80% * 100 000) / (24 hours * 60* 60 seconds *40%)) /1 = **23.1**requests / sec

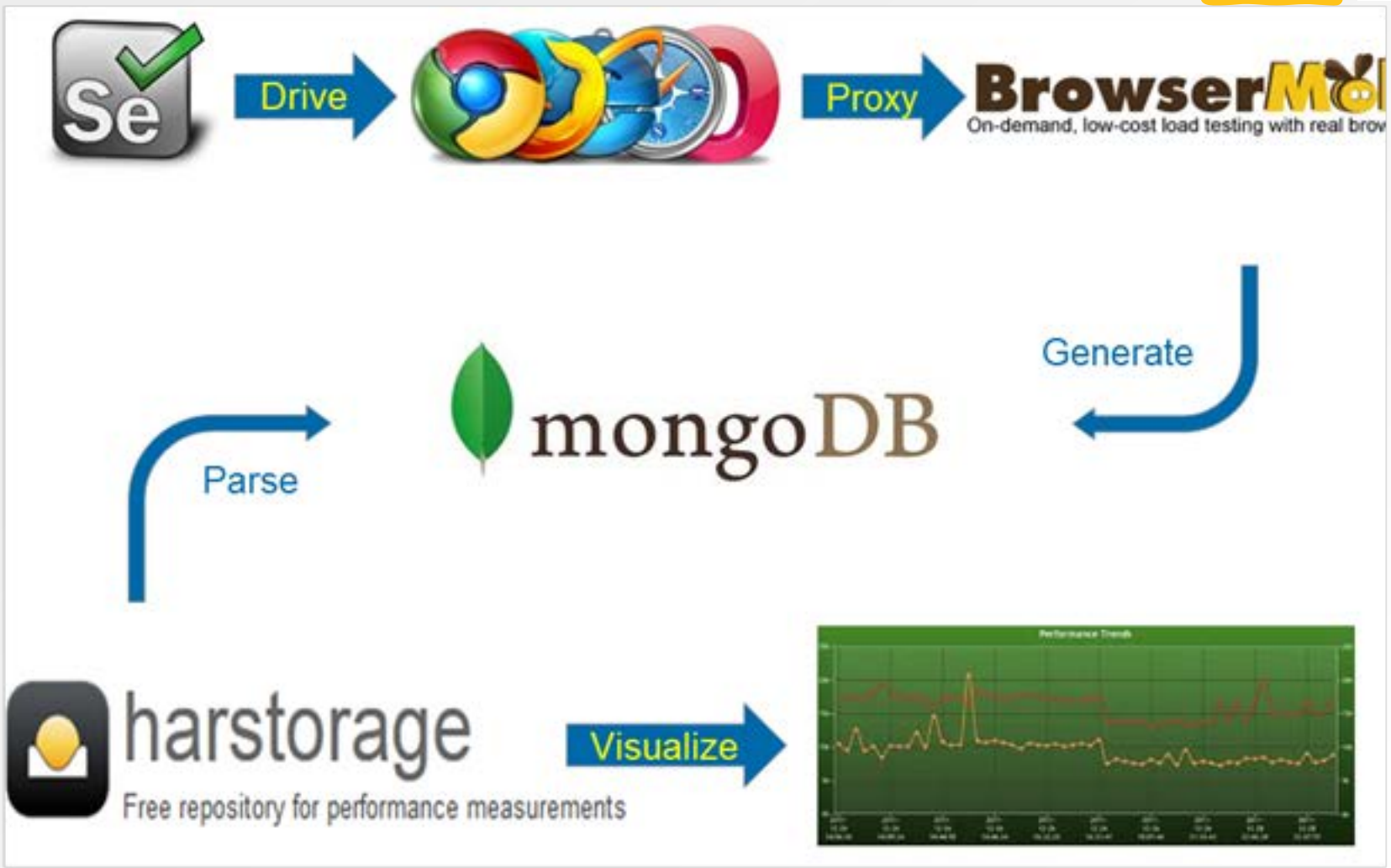
Performance/Load Test Structure



Stability test Structure



Frontend Performance Test Structure



Challenges

- QE need to know more about development skills
  - Do a better planning or user modeling need collaboration work
  - “Try to know everything”
- Agile LnP testing
  - How to balance between flexible and control.
  - What kind of process can be best fit in scrum.
- RUI design
  - Need check the performance result on both desktop and different kinds of Mobile devices.

Q&A