

Software Engineering Conference Russia
October 2017, St. Petersburg



Нагрузочное тестирование на основе Selenium тестов И НЕ ТОЛЬКО

Владимир Трубников

DELL EMC, Principal SW Engineer

Задача и начальные условия

Web-application



Функциональные тесты
(зачастую Selenium)



Большое количество
пользователей



Задача:

Проверить выдержит ли приложение нагрузку в реальном мире?*

- местами есть специфика + чем точнее мы сэмулируем действия пользователей, тем лучше



Существующие решения



- + Бесплатный
 - + Графический интерфейс
 - + Возможность записи нагрузки, производимой пользователем
 - + Cross-platform
 - + Многопоточковый фреймворк
 - + Интеграция с Maven, Jenkins, Gradle
- Только запросы
 - Не совсем честная симуляция
 - Узкая направленность тестов



ApacheBench



– Только запросы ☹️

– Узкая направленность ☹️

Цена вопроса

- **Еще один фреймворк в копилку**
- **Время на изучение или стоимость эксперта**
- **Время на написание**
- **Время на поддержку**
- **Человеческие ресурсы**
- **Стоимость софта**



Что такое нагрузочные тесты?

Нагрузочные тесты = базовый сценарий • N раз,
Базовый сценарий = набор основных действий

&

Функциональные тесты

Тесты на основные действия



Функциональные тесты • N раз = Нагрузочные тесты

Нет доп. поддержке!

Нет доп. софту!

**+ реальный браузер = абсолютно честная
симуляция вместе со всем «фоном»**



Как это устроить? Ч.1

Selenium tests + браузер (headless, phantomjs, честный браузер) + N потоков

Где запускать?



Как это устроить? Ч.1

Selenium tests + браузер (headless, phantomjs, честный браузер) + N потоков

Где запускать?

Option 1: с одной машины

Почему нет?

- На 1 поток phantomjs ~ 1Gb RAM
- все действия с одного адреса
 - неточная симуляция (IP story)

Option 2: набор VMs.

Попробуем.



Как это устроить? Ч.2



Jenkins

На Linux VMs

- 100 linux VMs in Mesos Cloud
- Jenkins Mesos plugin
- Headless browser
- Dockerfile
(install soft, checkout tests)
- Jenkins groovy pipeline job
для параллельного
выполнения тестов

На Windows VMs

- Swarm Jenkins plugin
- 200 VMs from prepared image
- скрипт для подключения
- Jenkins job для 1 юзера
- Jenkins groovy pipeline job с
параллельным выполнением
200 builds из предыдущего
пункта



Linux case: подключение слейвов

Label String	<input type="text" value="my_load_tests"/>
Использование	<input type="text" value="Use this node as much as possible"/>
Node Properties	<input type="button" value="Add Node Property"/>
Jenkins Slave CPUs	<input type="text" value="0.5"/>
Jenkins Slave Memory in MB	<input type="text" value="1536"/>
Minimum number of Executors per Slave	<input type="text" value="5"/>
Maximum number of Executors per Slave	<input type="text" value="5"/>
Jenkins Executor CPUs	<input type="text" value="0.05"/>
Jenkins Executor Memory in MB	<input type="text" value="1024"/>

Use Docker Containerizer

Container Type

Docker

Docker Image

If using Docker, specify the docker image.

Docker Privileged Mode

This will run the image using Docker's privileged mode.

Docker Force Pull Image

This will force a pull of the Docker Image regardless of whether it exists locally.

- 100 VMs with CentOS to Mesos Cloud
 - 16Gb RAM, 2 CPU, 30Gb Disk
 - Connect to mesos cloud (install docker, ntp, firewalls, etc)

<https://docs.mesosphere.com/1.9/administering-clusters/add-a-node/>

- Конфигурация Jenkins Mesos plugin

Linux case: подготовка слейвов

- Dockerfile для подготовки выполнения тестов (<80 строк)

```
22 # install ssh daemon
23
24 #RUN yum -y install openssh-server openssh-clients passwd
25 RUN mkdir -p /var/run/ssh \
26     && ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ''
27
28 # install java
29 ENV JAVA_HOME /usr/java/jdk1.8.0_60/jre
30 RUN yum -y install wget \
31     && wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=http://www.25.com; gclb=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u60-b27/jdk-8u60-linux-
32     && yum -y localinstall jdk-8u60-linux-x64.rpm \
33     && rm -f jdk-8u60-linux-x64.rpm \
34     && keytool -noprompt -import -alias emc_ssl -file $CERTIFICATES_HOME/emc_ssl.crt -keystore $JAVA_HOME/lib/security/cacerts -storepass changeit \
35     && keytool -noprompt -import -alias emc_ca -file $CERTIFICATES_HOME/emc_ca.crt -keystore $JAVA_HOME/lib/security/cacerts -storepass changeit
36
37 ENV MAVEN_VERSION="3.2.5" \
38     M2_HOME=/usr/lib/mvn
39
40 RUN mkdir -p /tmp \
41     && cd /tmp \
42     && wget "http://ftp.unicamp.br/pub/apache/maven/maven-3/$MAVEN_VERSION/apache-maven-$MAVEN_VERSION-bin.tar.gz" \
43     && tar -zxvf "apache-maven-$MAVEN_VERSION-bin.tar.gz" \
44     && mv "apache-maven-$MAVEN_VERSION" "$M2_HOME" \
45     && ln -s "$M2_HOME/bin/mvn" /usr/bin/mvn \
46     && rm -rf /tmp/*
47
48 ▶ RUN pip install fabric==1.12.0
49 RUN pip install pika \
50     jsonschema \
51     pytest \
52     pytest-html \
53     bs4 \
54     requests \
55     pandas \
56     numpy \
57     matplotlib
58
59 ▶ RUN wget --no-check-certificate https://bitbucket.org/ariya/phantomjs/downloads/phantomjs-2.1.1-linux-x86_64.tar.bz2 \
60     && tar xvf phantomjs-2.1.1-linux-x86_64.tar.bz2 \
61     && cp phantomjs-2.1.1-linux-x86_64/bin/phantomjs /usr/local/bin \
62     && rm -f phantomjs-2.1.1-linux-x86_64.tar.bz2
63
64 # add info to profile
65 RUN echo 'export M2_HOME=/usr/lib/mvn' >> /etc/profile \
66     && echo 'export JAVA_HOME=/usr/java/jdk1.8.0_60/jre' >> /etc/profile \
67     && echo 'export PATH=$JAVA_HOME:$PATH' >> /etc/profile
68
69 #add jenkins dir
70 ENV JENKINS_PATH /root/jenkins
71 RUN mkdir -p $JENKINS_PATH
72
73 WORKDIR /root
74
75 RUN git clone -b master --single-branch https://login:password@repo.emc.com/Tests.git
76 RUN cd Test \
77     && mvn -DskipTests=true clean install -B
```

Ставим софт:
Java
Maven
Certificates
Browser (phantomjs)



Выкачиваем и собираем тесты

Linux case: сборка образа

- Jenkins job для сборки docker image

Build

Docker Build and Publish X ?

Repository Name ?

Tag

Docker Host URI ?

Server credentials Add


Docker registry URL ?

Registry credentials Add

Skip Push

Do not push image to registry/index on successful completion

No Cache

Force build - do not use docker cache (may be slower) 

Force Pull

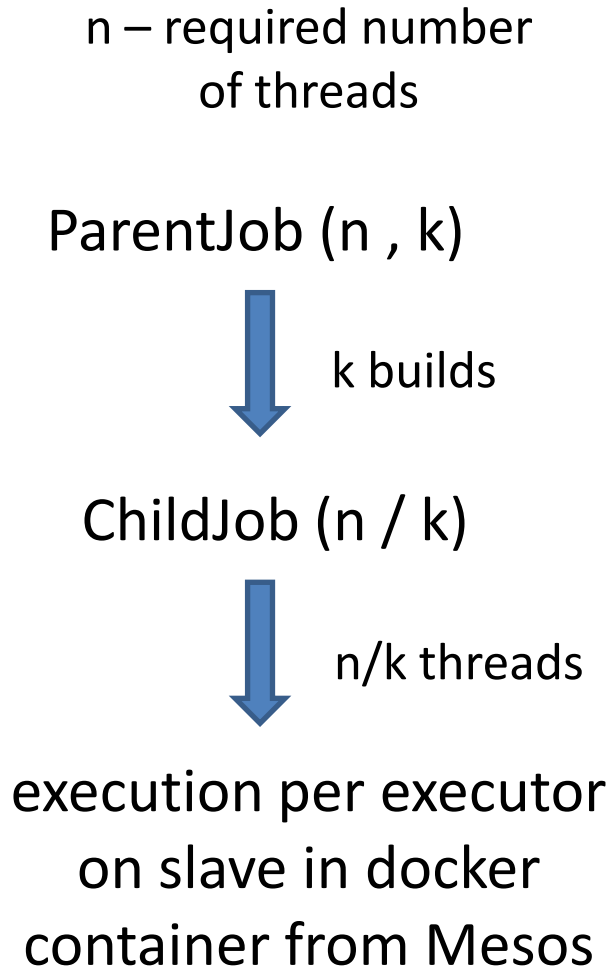
Update the source image before building even when it exists locally

Linux case: ВЫПОЛНЕНИЕ ТЕСТОВ

```
def branches = [:]

for (int i=0; i<executorsNum; i++) {
  def index = i
  branches["branch${index}"] = {
    node('eos2-thehub-performance') {
      def branch_wspace = "wspace_branch" + index;
      sh 'cd /root && rm -rf ' + branch_wspace +
        ' && mkdir ' + branch_wspace +
        ' && cp -r TheHUBPerformance/* ' + branch_wspace +
        ' && cd /root/' + branch_wspace +
        ' && mvn -DsomeProp=someVal -fn -B -q install ' +
        ' && cd /root && rm -rf ' + branch_wspace + ' &'}}
  parallel branches
}
```

Linux case: масштабирование



```
for (int i=0; i<numOfBuilds; i++)
{
    def index = i
    branches["branch${index}"] = {
        build job: 'ChildJob',
        parameters: [
            string(name: 'executorsNum',
                value: num),
            string(name: 'splitNum',
                value: index)]}
}
parallel branches
```

Linux case: полученный опыт

- Dynamic slaves from Mesos
 - Конфиг Mesos
 - Warmup
 - Время жизни слейвов
- Запуск одинаковых билдов
- Время выполнения дольше оценочного
- Остановка тестов
- Memory usage у Jenkins



Windows: подготовка и подключение

1. ~~Mesos, Docker~~. **200 VMs as slaves** in Jenkins + active console (зачем?)

2. **Подготовка образа:**

- Swarm Plugin: положить swarm-client.jar на диск
- Batch скрипт для запуска агента:
`java -jar C:\swarm-client.jar -master http://jenkinsUrl -executors 1 -fsroot C:/home -username myuser -password mypass`
- Batch скрипт для отключения RDP со стороны машины:
`for /f "tokens=4 delims= " %%G in ('tasklist /FI "IMAGENAME eq explorer.exe" /NH') do SET RDP_SESSION=%%G
tscon %RDP_SESSION% /dest:console`

3. **Сделать образ и запустить машины**

4. **PsExec и Remote Desktop Plus для подключения машин**

```
rdp.exe /v:%1 /batch /u:Administrator /p>Password123! /w:1280 /h:800  
PsExec.exe \\%1 -h -u Administrator -p Password123! cmd /c "C:\RDP.bat"  
taskkill /IM mstsc.exe /F
```

Windows: выполнение тестов

- **Jenkins job** для выполнения тестов с разрешенными concurrent builds настроенная на выполнение на нодах с меткой swarm
- **Pipeline** для вызова требуемого количества билдов:

```
def branches = [:]
for (i=0; i<numOfThreads; i++) {
  def index = i
  branches[i] = {
    build job: 'OneThreadJob',
    parameters:
      $class: 'StringParameterValue',
      name: 'dummy',
      value: index
  }}
parallel branches
```

n – требуемое
количество потоков

ParentJob (n)



n builds

OneThreadJob



Выполняется на
предварительно
подключенных при помощи
Swarm-client машинах

Windows: итоги и опыт

- Запуск одинаковых билдов
- Остановка тестов
- Memory usage у Jenkins



Вывод: минусы и плюсы

- Большое количество ресурсов:

- ~160GB RAM + 20 CPUs на 100 потоков (10 VMs*2 slaves* 5 executors)
- ~32Gb + 8 CPUs for Jenkins Master
(Но, можно начать с малого: 10 VMs = 100 потоков)

- Затраты на создание инфраструктуры

(Mesos Cloud, docker registry или получение Windows VMs)

- 1 день на подготовку образа (Win)
- ? дней на получение машин
- 7-14 дней на Mesos Cloud (Linux)
- 1 день на подключение к Mesos (Linux)
- 1 день на подготовку docker registry (Linux)

Вывод: минусы и плюсы

+ Простая имплементация:

- 1 Dockerfile (<100 строк) (Linux)
- 2-3 Jenkins jobs
- 2-3 batch скрипта (<50 строк) (Windows)
- 2-3 groovy скрипта (<100 строк)
- Несколько нехитрых настроек Jenkins
- Пара дней на первую имплементацию
- Пара часов для адаптации под другие тесты

= LTaaS

+ Реалистичная нагрузка (IP story)

+ **Масштабируемая и переиспользуемая** инфраструктура для запуска любых тестов (меняется 1 команда для вызова тестов)

+ **Унификация:** нет необходимости поддерживать специальные load тесты

Q & A ?

PS:

