Cognition-driven Maintenance supported by Computing Continuum: COGNIWARE

Nenad Stojanovic, Milan Vuckovic

Nissatech

OnLine

24.5.2022

Outline

- Experiment description
- Project results
- Business impact
- Feedback

Experiment description

CONCEPT AND OBJECTIVES

Product-service systems (PSS) are business models that provide for cohesive delivery of products and services. PSS models are emerging as a means to enable collaborative consumption of both products and services

In product-service systems, the main challenge is that the **services provided by the product** can be updated frequently, so that new services, as well as trainings have to be offered in a very efficient way.

Experiment description

CONCEPT AND OBJECTIVES

Therefore, such an offering requires a **powerful computing infrastructure** which would ensure a comfortable usage of services, e.g. **minimizing the delay** in the case of requesting the service by a lot of users. Consequently, it implies a **trade off between the efficiency and the price of the service**.

Main objective is the performance measurement and scalability test of new cognition-driven maintenance of product-service systems that is able to detect issues/anomalies in the operation as soon as they appear (and before a problem arises)

Experiment description

BACKGROUND AND MOTIVATION

We are developing a new service for the **management of product-service systems** which is based on offering various types of services for particular users and users groups.

There are several types of customers:

- Machine vendors
- Maintenance service companies

Experiment description

EXPERIMENT SET UP

Grid5000 (Inria)	One node	120GB RAM, Xeon CPU, 64 GB Hard disk

Experiment description

EXPERIMENT SET UP

Automatization and preparation

In order to start testing we Dockerized our PRODUCTION system for easier deployment.

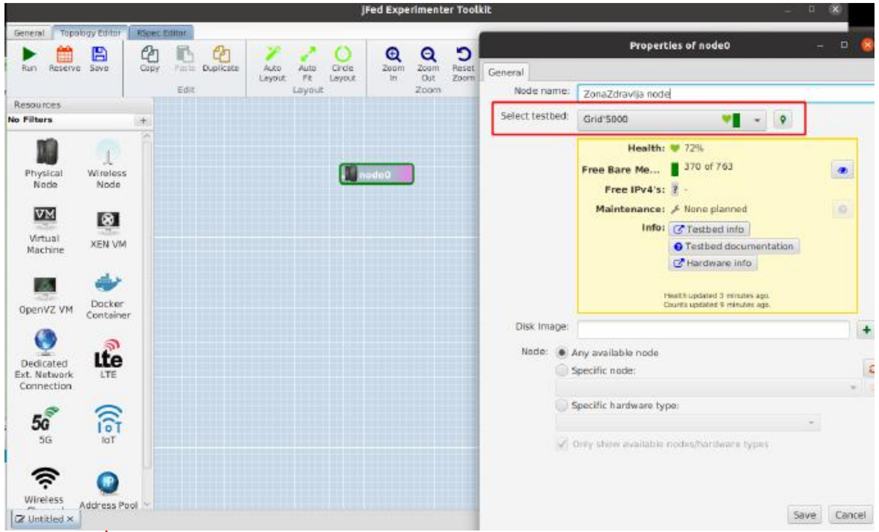
The whole system is around 15Gb of data, which we copied to the server using SCP Linux command (on Ubuntu 18 regular SSH transfer which jTest offers did not work!) connecting from Grid'5000 to our server. After installing needed packages (Docker, Docker-compose, cURL, etc.), and some minor issues which we fixed pretty easily, we started all containers and tested connection and functionality (image below).

Experiment description

EXPERIMENT SET UP

JFed is a Java-based framework for testbed federation, it provides **GUI and CLI** to allow end-users to provision and manage experiments.

We've reserved Grid5000 resource, installed docker, postgres and so on and after that we deployed our app.



General

Node name: ZonaZdravlja node

Select testbed: Grid'5000

Health: 72%

Free Bare Me...: 370 of 763

Free IPv4's: ? -

Maintenance: None planned

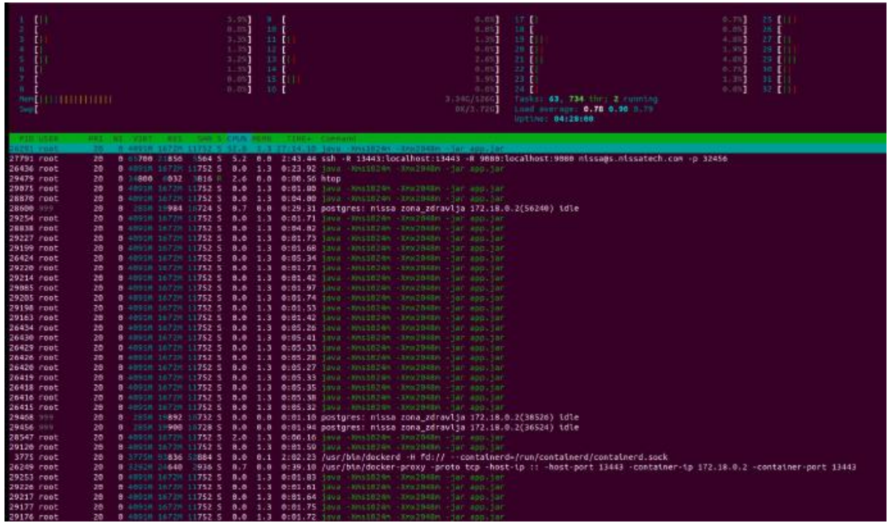
Info:

Health updated 3 minutes ago.
Counts updated 9 minutes ago.

Disk Image:

Node: [✓ Any available node] Specific node: Specific hardware type:

Only show available nodes/hardware types



```
                                          Tasks: 63, 734 thr; 2 running
                                          Load average: 0.78 0.90 0.79
                                          Uptime: 04:28:00

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
27791 root       20   0 65700 21856  5564 S  5.2  0.0  2:43.44 ssh -R 13443:localhost:13443 -R 9000:localhost:9000 nissa@s.nissatech.com -p 32456
26436 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:23.92 java -Xms1024m -Xmx2048m -jar app.jar
29479 root       20   0 34800  4032  3816 R  2.6  0.0  0:00.56 htop
29075 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.80 java -Xms1024m -Xmx2048m -jar app.jar
28870 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:04.00 java -Xms1024m -Xmx2048m -jar app.jar
28600 999        20   0  285M 19984 16724 S  0.7  0.0  0:29.31 postgres: nissa zona_zdravlja 172.18.0.2(56240) idle
29254 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.71 java -Xms1024m -Xmx2048m -jar app.jar
28838 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:04.82 java -Xms1024m -Xmx2048m -jar app.jar
29227 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.73 java -Xms1024m -Xmx2048m -jar app.jar
29199 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.68 java -Xms1024m -Xmx2048m -jar app.jar
26424 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.34 java -Xms1024m -Xmx2048m -jar app.jar
29220 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.73 java -Xms1024m -Xmx2048m -jar app.jar
29214 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.42 java -Xms1024m -Xmx2048m -jar app.jar
29085 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.97 java -Xms1024m -Xmx2048m -jar app.jar
29205 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.74 java -Xms1024m -Xmx2048m -jar app.jar
29198 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.53 java -Xms1024m -Xmx2048m -jar app.jar
29163 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.42 java -Xms1024m -Xmx2048m -jar app.jar
26434 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.26 java -Xms1024m -Xmx2048m -jar app.jar
26430 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.41 java -Xms1024m -Xmx2048m -jar app.jar
26429 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.33 java -Xms1024m -Xmx2048m -jar app.jar
26426 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.28 java -Xms1024m -Xmx2048m -jar app.jar
26420 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.27 java -Xms1024m -Xmx2048m -jar app.jar
26419 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.33 java -Xms1024m -Xmx2048m -jar app.jar
26418 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.35 java -Xms1024m -Xmx2048m -jar app.jar
26416 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.38 java -Xms1024m -Xmx2048m -jar app.jar
26415 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:05.32 java -Xms1024m -Xmx2048m -jar app.jar
29468 999        20   0  285M 19892 16732 S  0.0  0.0  0:01.10 postgres: nissa zona_zdravlja 172.18.0.2(38526) idle
29456 999        20   0  285M 19900 16728 S  0.0  0.0  0:01.94 postgres: nissa zona_zdravlja 172.18.0.2(36524) idle
28547 root       20   0 4091M 1672M 11752 S  2.0  1.3  0:06.16 java -Xms1024m -Xmx2048m -jar app.jar
29120 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.59 java -Xms1024m -Xmx2048m -jar app.jar
 3775 root       20   0 3775M 53836 52884 S  0.0  0.1  2:02.23 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
26249 root       20   0 3292M 24640  2936 S  0.7  0.0  0:39.10 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 13443 -container-ip 172.18.0.2 -container-port 13443
29253 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.83 java -Xms1024m -Xmx2048m -jar app.jar
29226 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.61 java -Xms1024m -Xmx2048m -jar app.jar
29217 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.64 java -Xms1024m -Xmx2048m -jar app.jar
29177 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.75 java -Xms1024m -Xmx2048m -jar app.jar
29176 root       20   0 4091M 1672M 11752 S  0.0  1.3  0:01.72 java -Xms1024m -Xmx2048m -jar app.jar
```

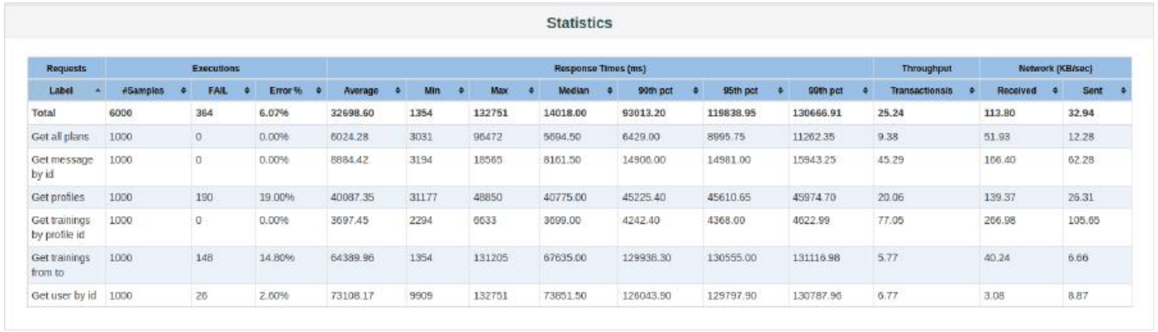
Project result

MEASUREMENTS

Tests concluded with sending the requests at the same time. When we say 6000 users, we mean that 6000 users are active at the same time (which means total users using app actively is much larger).

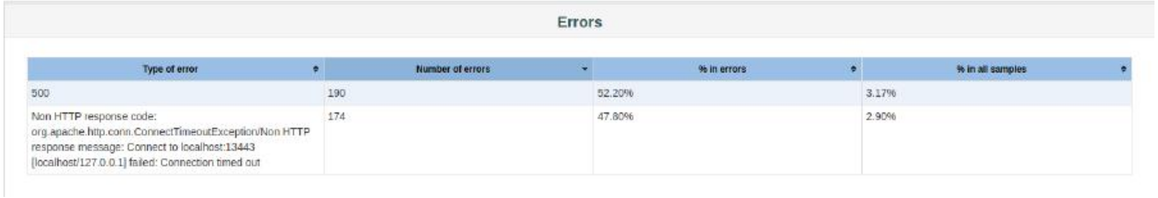
Project result

6000 users (requests at the same time)



Statistics

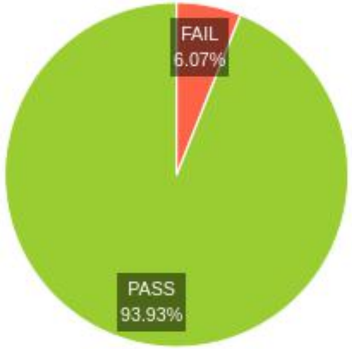
Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**6000**	**364**	**6.07%**	**32698.60**	**1354**	**132751**	**14018.00**	**93013.20**	**119838.95**	**130666.91**	**25.24**	**113.80**	**32.94**
Get all plans	1000	0	0.00%	6024.28	3031	96472	5694.50	6429.00	8995.75	11262.35	9.38	51.93	12.28
Get message by id	1000	0	0.00%	8884.42	3194	18565	8161.50	14906.00	14981.00	15943.25	45.29	166.40	62.28
Get profiles	1000	190	19.00%	40087.35	31177	48850	40775.00	45225.40	45610.65	45974.70	20.06	139.37	26.31
Get trainings by profile id	1000	0	0.00%	3697.45	2294	6633	3699.00	4242.40	4368.00	4622.99	77.05	266.98	105.65
Get trainings from to	1000	148	14.80%	64389.96	1354	131205	67635.00	129938.30	130555.00	131116.98	5.77	40.24	6.66
Get user by id	1000	26	2.60%	73108.17	9909	132751	73851.50	126043.90	129797.90	130787.96	6.77	3.08	8.87



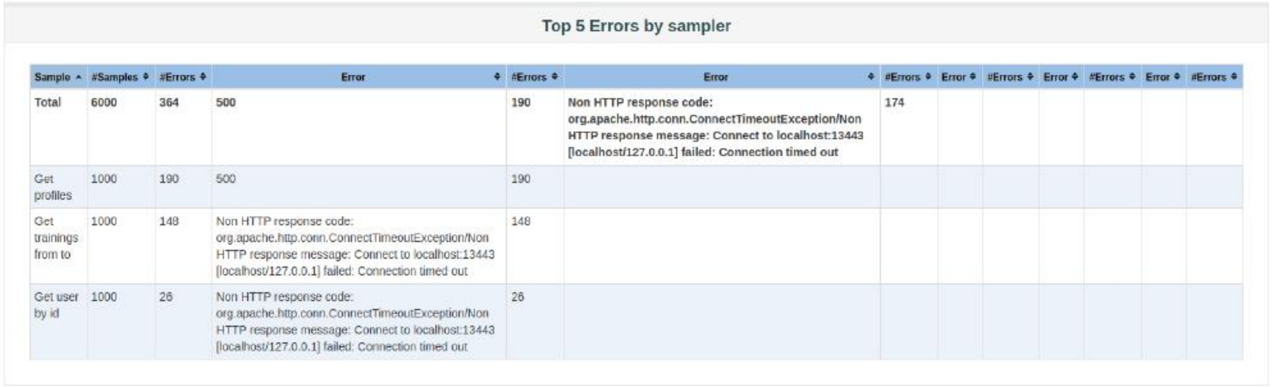
Errors

Type of error	Number of errors	% in errors	% in all samples
500	190	52.20%	3.17%
Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out	174	47.80%	2.90%

Requests Summary

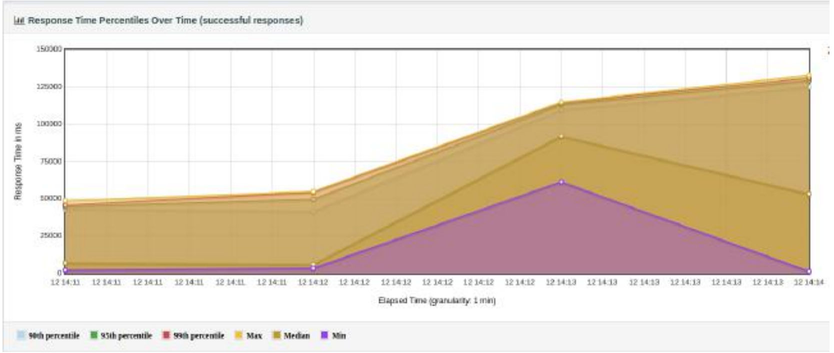


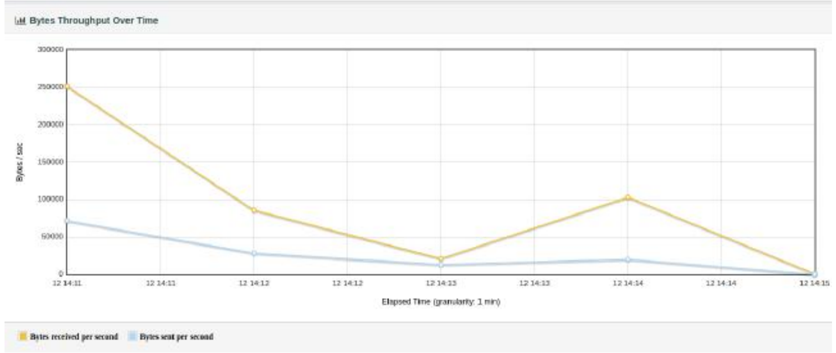
Project result



Top 5 Errors by sampler

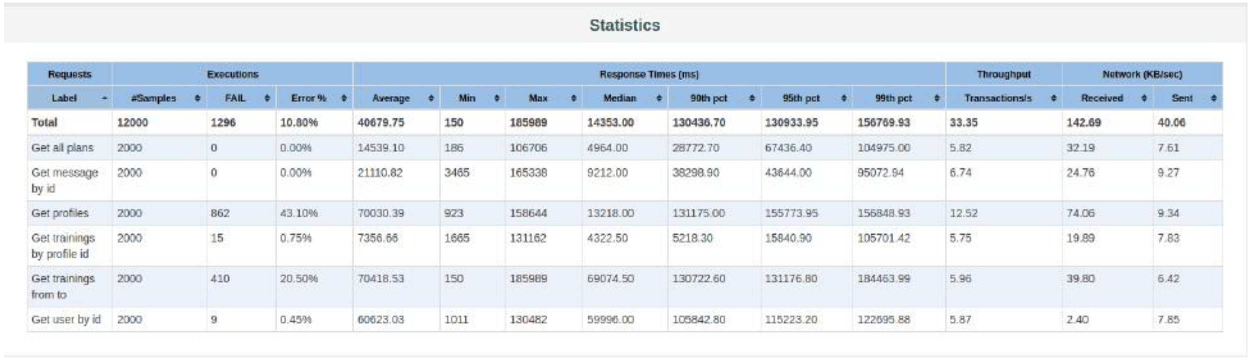
Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	**6000**	**364**	**500**	**190**	**Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out**	**174**						
Get profiles	1000	190	500	190								
Get trainings from to	1000	148	Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out	148								
Get user by id	1000	26	Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out	26								



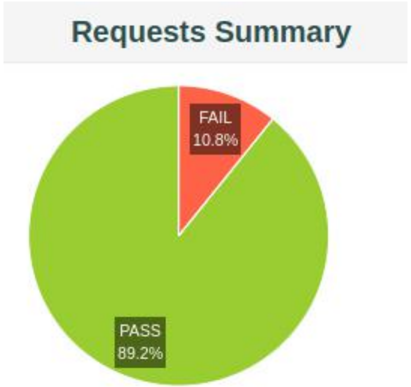


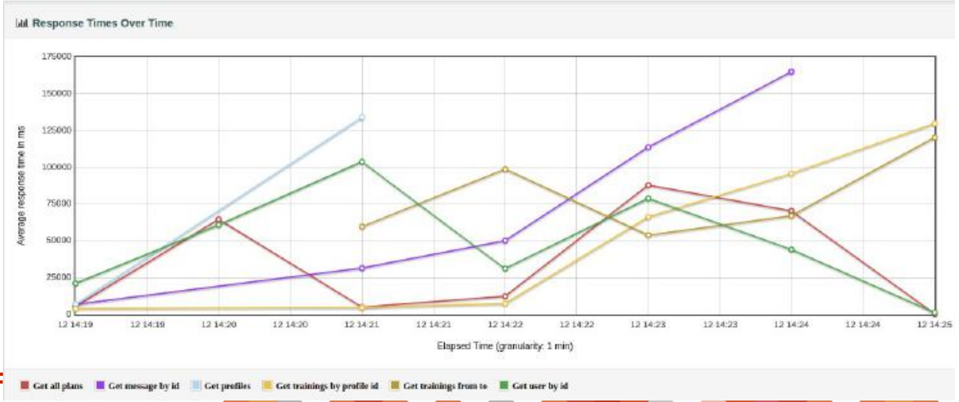
Project result

12000 users (requests at the same time)

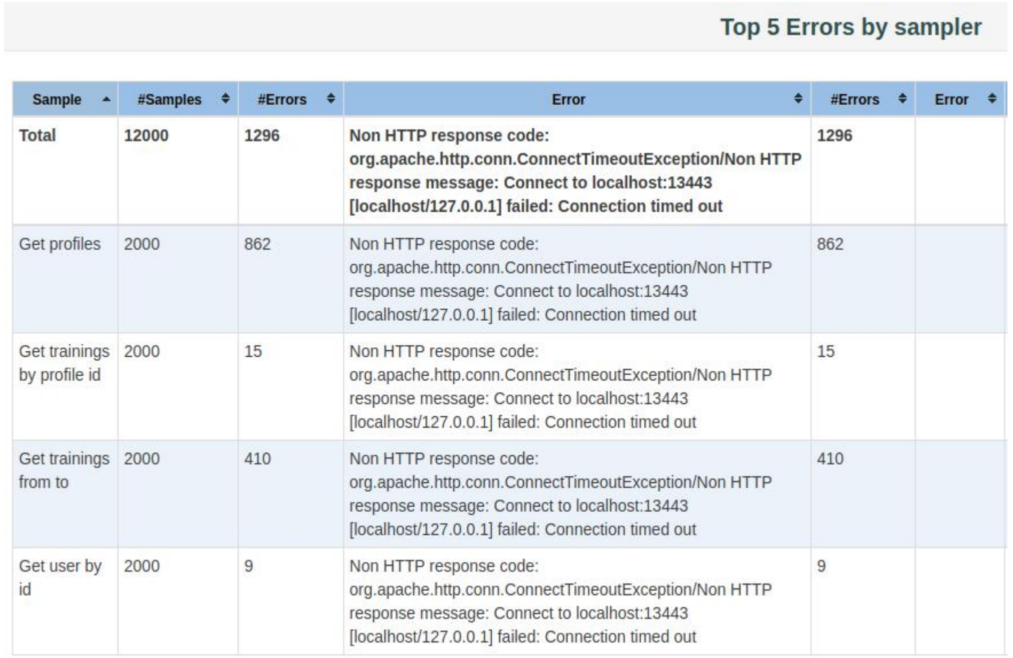

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**12000**	**1296**	**10.80%**	**40679.75**	**150**	**185989**	**14353.00**	**130436.70**	**130933.95**	**156769.93**	**33.35**	**142.69**	**40.06**
Get all plans	2000	0	0.00%	14539.10	186	106706	4964.00	28772.70	67436.40	104975.00	5.82	32.19	7.61
Get message by id	2000	0	0.00%	21110.82	3465	165338	9212.00	38298.90	43644.00	95072.94	6.74	24.76	9.27
Get profiles	2000	862	43.10%	70030.39	923	158644	13218.00	131175.00	155773.95	156848.93	12.52	74.06	9.34
Get trainings by profile id	2000	15	0.75%	7356.66	1665	131162	4322.50	5218.30	15840.90	105701.42	5.75	19.89	7.83
Get trainings from to	2000	410	20.50%	70418.53	150	185989	69074.50	130722.60	131176.80	184463.99	5.96	39.80	6.42
Get user by id	2000	9	0.45%	60623.03	1011	130482	59996.00	105842.80	115223.20	122695.88	5.87	2.40	7.85



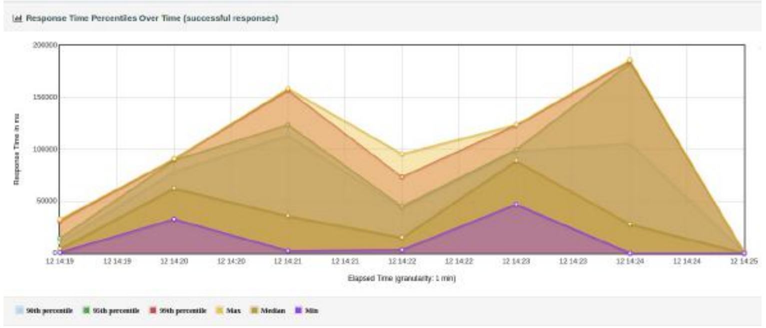


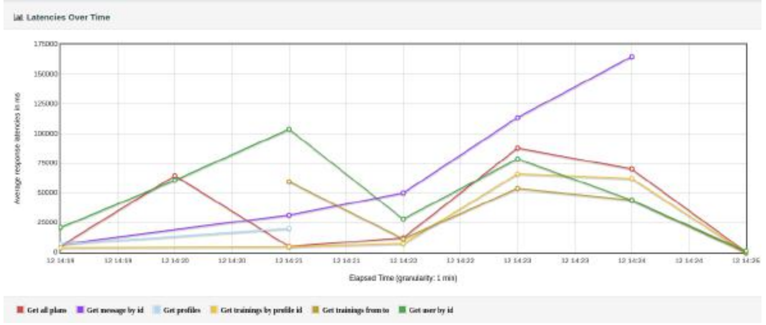
Project result



Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error
Total	**12000**	**1296**	**Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out**	**1296**	
Get profiles	2000	862	Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out	862	
Get trainings by profile id	2000	15	Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out	15	
Get trainings from to	2000	410	Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out	410	
Get user by id	2000	9	Non HTTP response code: org.apache.http.conn.ConnectTimeoutException/Non HTTP response message: Connect to localhost:13443 [localhost/127.0.0.1] failed: Connection timed out	9	





Project result

LESSONS LEARNED

As a result of these tests, we can freely state that our system can support at least 5000 users who are active every day at the same time.

And for further expansion we will plan ahead with this information to multiply instances of our service with Kubernetes probably.

Business impact

ON BUSINESS

Main finding is that the number of users who can use the system in a convenient way depends on **the capacity of the infrastructure in a non-linear manner** - we have obtained some important conclusion for the current **projection of the number of users** (max 20 000 users).

Moreover, there is **another important conclusion** from the business point of view: most critical feature for the user is "**care", i.e. their feeling that the service is tailored to their needs**. This has inspired us for the design of a new customer service which will provide a fast response on customer need (based on fast thinking), even before having a precise answer on the issue at hand. Indeed, we see the we see as future of the customer care and we call it CARE++

Business impact

ON BUSINESS

- Added value regarding checking the technical feasibility of the system

Main value is in the possibility to perform various tests for exploring the limitation/constraints in the current offering of the cognition-driven maintenance for product-service systems. Besides very important findings regarding the performance, like the limitation of the number of users for a selected infrastructure (size), we have found also an interesting conclusion about the nature of the customer support which should be provided in order to keep the user satisfied.

Business impact

ON BUSINESS

- Added value for the further development of the system

We obtained a deep understanding of the real performances of some of the parts of the cognition-driven maintenance process.

- Added value for the further business modelling

Influencing the business decisions about the further commercialisation, especially from the point of view of some non-functional requirement

Feedback

JFed	yes	Installation and usage were standard, we had some issues with some random given disk images, after which we focused on Ubuntu 18 only – and that worked perfect for us (as we have our system on the same image). Issues we had was with transferring files which worked on Debian system, but not on Ubuntu, for unexpected reasons (we sent auto generated reports with this error). But on Debian we had issues installing Docker-Compose, so we focused on CLI and Ubuntu 18. Another issue was having only 24 hours for experiment, in real time that was 8 hours (working hours), we tried creating a disk image from withing Grid’5000 server, but transferring files lasted more than the actual copying (SCP) and deploying.
JFed command Line (CLI)	yes	Using CLI was a positive thing in our experience. We needed a few packages to install, SSH, SCP to transfer files from our servers to Grid’5000 (database, Keycloak data etc.) and reverse port forwarding to enable sending data from our servers (local machine) to Grid’5000.
JMeter	yes	Pretty straight forward, small manageable issues which were expected. Problems we experienced were about methods to fetch a video, which needed to be buffered and we always got memory exceptions (Heap), so we could not test that part very well.
Docker	yes	We used Docker and Docker compose to ease the deployment and setup of our services on Grid’5000. Small issues with root user for Postgres certificates (.pem file).

Feedback

We expected to test our application in a few days, but there is no option to reserve a node for more than 24h, except that everything was fine.

Since our machine lasted only 24 hours, we knew everything was deleted from it (or we hope so).

Besides that, we had Vault for all our passwords withing our deployment, so we did not bat an eye on this issue (also we used beta version of Postgres DB and Keycloak as Auth2.0).

Our system is Dockerized, so we only used docker and docker-compose which got all images and created containers for db, keycloak, filebeat, java etc.

Copying files took some time (15Gb).

Reverse port forwarding was hard to figure out at first, but when we found the command, it was straight forward.

Feedback - missing

A variety of testing software would be a delight. In CLI usage of course! To be added when starting the machine (or selecting testbed).

Getting a machine for more than 24 hours would definitely make testing and deployment way faster and easier.

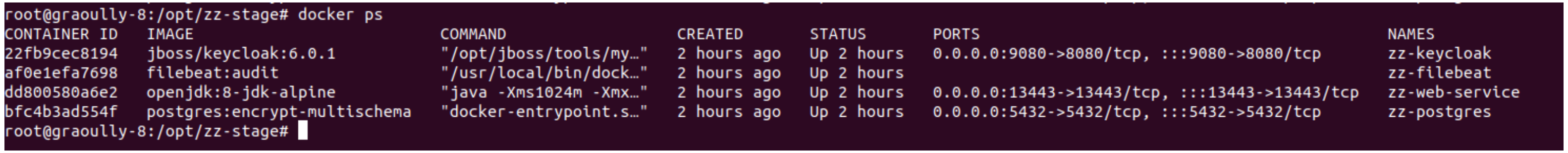
Getting our own disk image would be a perfect option for deployment, setting up whole server into .img file and just transferring and unpacking it onto testbed.

Ability to pick what do we need with the machine, e.g., when we are getting a machine, we can check that we want Docker or Kubernetes within that server, so creating one will start a script which will install all those needed packages and tools.

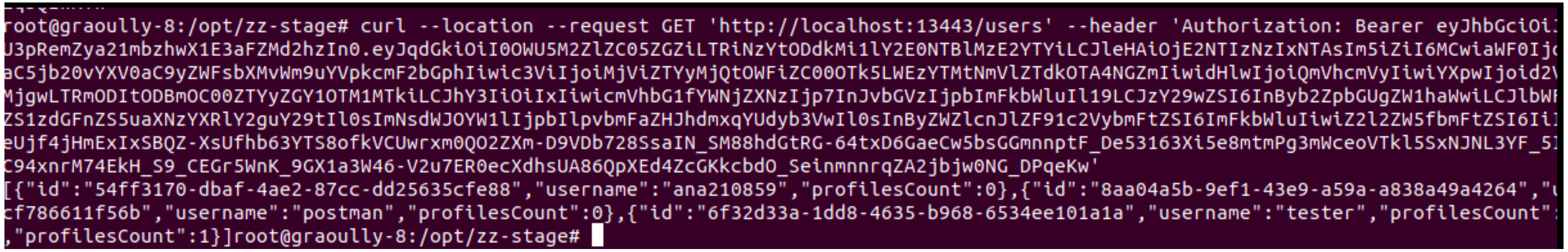
Feedback

In order to run JMeter tests the easier way (with GUI from our computer) we did reverse port forwarding from Grid'5000 machine to our own:

ssh -R 13443:localhost:13443 –R 9080:localhost:9080 {username@ourip.com} -p {internal_port}



```
root@graoully-8:/opt/zz-stage# docker ps
CONTAINER ID   IMAGE                        COMMAND                  CREATED       STATUS       PORTS                                           NAMES
22fb9cec8194   jboss/keycloak:6.0.1         "/opt/jboss/tools/my…"   2 hours ago   Up 2 hours   0.0.0.0:9080->8080/tcp, :::9080->8080/tcp       zz-keycloak
af0e1efa7698   filebeat:audit               "/usr/local/bin/dock…"   2 hours ago   Up 2 hours                                                   zz-filebeat
dd800580a6e2   openjdk:8-jdk-alpine         "java -Xms1024m -Xmx…"   2 hours ago   Up 2 hours   0.0.0.0:13443->13443/tcp, :::13443->13443/tcp   zz-web-service
bfc4b3ad554f   postgres:encrypt-multischema "docker-entrypoint.s…"   2 hours ago   Up 2 hours   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp       zz-postgres
root@graoully-8:/opt/zz-stage#
```



```
root@graoully-8:/opt/zz-stage# curl --location --request GET 'http://localhost:13443/users' --header 'Authorization: Bearer eyJhbGciOi
U3pRemZya21mbzhwX1E3aFZMd2hzIn0.eyJqdGkiOiI0OWU5M2ZlZC05ZGZiLTRiNzYtODdkMi1lY2E0NTBlMzE2YTYiLCJleHAiOjE2NTIzNzIxNTAsIm5iZiI6MCwiaWF0Ijo
aC5jb20vYXV0aC9yZWFsbXMvWm9uYVpkcmF2bGphIiwic3ViIjoiMjViZTYyMjQtOWFiZC00OTk5LWEzYTMtNmVlZTdkOTA4NGZmIiwidHlwIjoiQmVhcmVyIiwiYXpwIjoid2
MjgwLTRmODItODBmOC00ZTYyZGY1OTM1MTkiLCJhY3IiOiIxIiwicmVhbG1fYWNjZXNzIjp7InJvbGVzIjpbImFkbWluIl19LCJzY29wZSI6InByb2ZpbGUgZW1haWwiLCJlbW
ZS1zdGFnZS5uaXNzYXRlY2guY29tIl0sImNsdWJOYW1lIjpbIlpvbmFaZHJhdmxqYUdyb3VwIl0sInByZWZlcnJlZF91c2VybmFtZSI6ImFkbWluIiwiZ2l2ZW5fbmFtZSI6Ii
eUjf4jHmExIxSBQZ-XsUfhb63YTS8ofkVCUwrxm0QO2ZXm-D9VDb728SsaIN_SM88hdGtRG-64txD6GaeCw5bsGGmnnptF_De53163Xi5e8mtmPg3mWceoVTkl5SxNJNL3YF_5
C94xnrM74EkH_S9_CEGr5WnK_9GX1a3W46-V2u7ER0ecXdhsUA86QpXEd4ZcGKkcbdO_SeinmnnrqZA2jbjw0NG_DPqeKw'
[{"id":"54ff3170-dbaf-4ae2-87cc-dd25635cfe88","username":"ana210859","profilesCount":0},{"id":"8aa04a5b-9ef1-43e9-a59a-a838a49a4264","
cf786611f56b","username":"postman","profilesCount":0},{"id":"6f32d33a-1dd8-4635-b968-6534ee101a1a","username":"tester","profilesCount"
,"profilesCount":1}]root@graoully-8:/opt/zz-stage#
```

Testing services in containers

Conclusion

- Experimentation done as planned
- Effective support from Grid5000
- Feedback is very positive with a short wishing list
- Important value for the further business design

FED4FIRE

Co-funded by the European Union

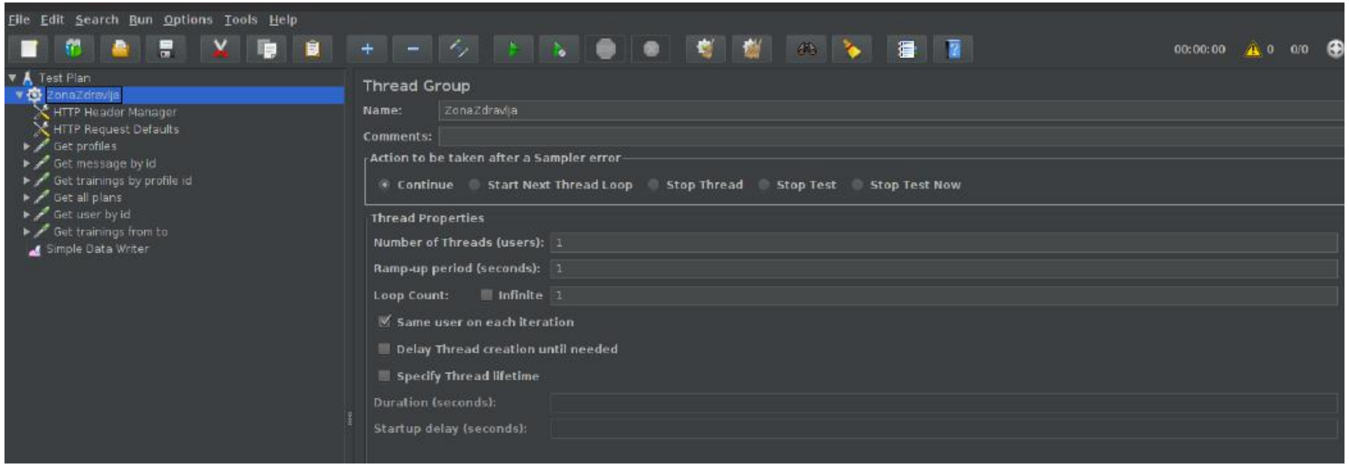
Co-funded by the Swiss Confederation

This project has received funding from the European Union's Horizon 2020 research and innovation programme, which is co-funded by the European Commission and the Swiss State Secretariat for Education, Research and Innovation, under grant agreement No 732638.

WWW.FED4FIRE.EU

Feedback

- JMeter, a testing tool from Apache used **to analyze and measure the performance of applications, different software services and products**. Open-source software entirely written in Java, used to test both web and FTP applications as long as the system supports a Java Virtual Machine (JVM).
- The thread group is part of the Test Plan where the user is able to determine how many requests will be sent.



- Test Plan
  - ZonaZdravlja
    - HTTP Header Manager
    - HTTP Request Defaults
    - Get profiles
    - Get message by id
    - Get trainings by profile id
    - Get all plans
    - Get user by id
    - Get trainings from to
    - Simple Data Writer

Thread Group

Name: ZonaZdravlja

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 1

Ramp-up period (seconds): 1

Loop Count: ☐ Infinite 1

☑ Same user on each iteration

☐ Delay Thread creation until needed

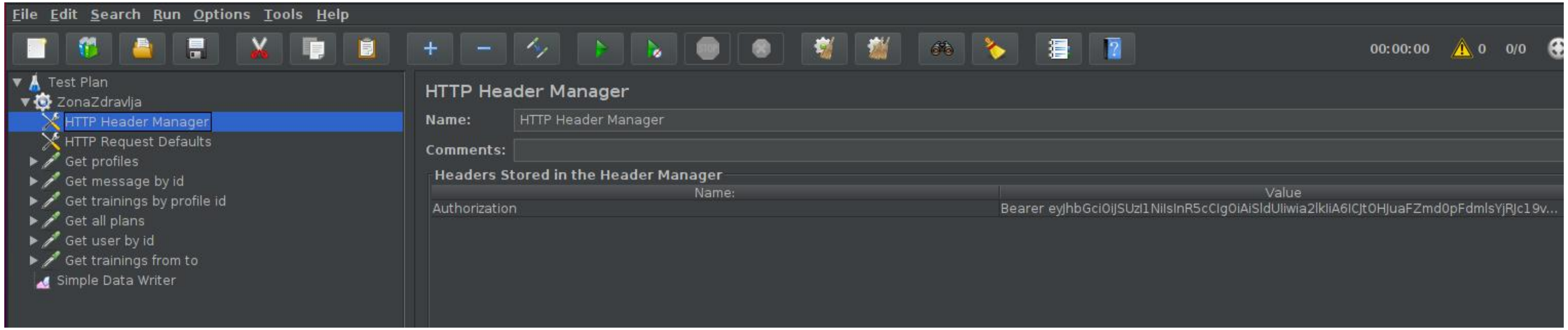
☐ Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

Feedback

HTTP Header Manager is part of the Thread Group and it is used in order to set Authorization. In our case we have a token which must be provided when the request is sent (acquired from our Keycloak on Grid’5000 server).



- Test Plan
  - ZonaZdravlja
    - HTTP Header Manager
    - HTTP Request Defaults
    - Get profiles
    - Get message by id
    - Get trainings by profile id
    - Get all plans
    - Get user by id
    - Get trainings from to
  - Simple Data Writer

HTTP Header Manager

Name: HTTP Header Manager

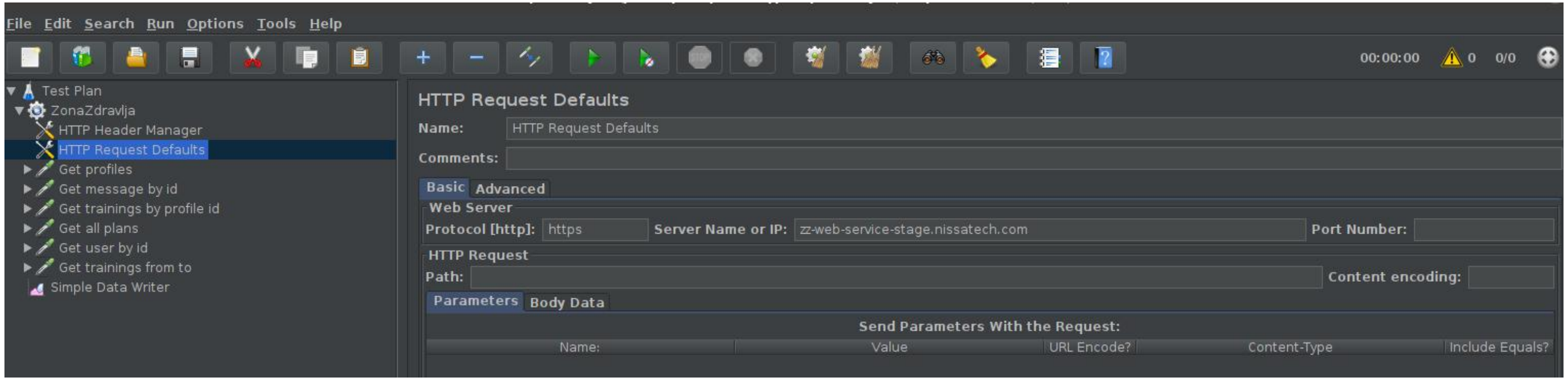
Comments:

Headers Stored in the Header Manager

Name:	Value
Authorization	Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJtOHJuaFZmd0pFdmlsYjRJc19v...

Feedback

HTTP Request Defaults is also a part of the Thread Group ZonaZdravlja and it is used to set parameters that are common to each request. Instead of setting parameters for each request, we set parameters and use them for each request



- Test Plan
  - ZonaZdravlja
    - HTTP Header Manager
    - HTTP Request Defaults
    - Get profiles
    - Get message by id
    - Get trainings by profile id
    - Get all plans
    - Get user by id
    - Get trainings from to
  - Simple Data Writer

HTTP Request Defaults

Name: HTTP Request Defaults

Comments:

Basic | Advanced

Web Server

Protocol [http]: https | Server Name or IP: zz-web-service-stage.nissatech.com | Port Number:

HTTP Request

Path: | Content encoding:

Parameters | Body Data

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?