OPERA

LOw Power Heterogeneous Architecture
for NExt Generation of SmaRt Infrastructure and Platforms
in Industrial and Societal Applications

Virtual Desktop Use Case 2

Co-funded by the Horizon 2020
Framework Programme of the European Union

DELIVERABLE NUMBER	D7.8
DELIVERABLE TITLE	Virtual Desktop Use Case 2
RESPONSIBLE AUTHOR	CSI Piemonte

GRANT AGREEMENT N.	688386
PROJECT REF. NO	H2020- 688386
PROJECT ACRONYM	OPERA
PROJECT FULL NAME	LOw Power Heterogeneous Architecture for Next Generation of SmaRt Infrastructure and Platform in Industrial and Societal Applications
STARTING DATE (DUR.)	01/01/2015
ENDING DATE	30/11/2018
PROJECT WEBSITE	www.operaproject.eu

WORKPACKAGE N. | TITLE	WP7 | Hardware/Software integration and Validation in Real Life workload
WORKPACKAGE LEADER	CSI Piemonte
DELIVERABLE N. | TITLE	D7.8 | Virtual Desktop Use Case 2
RESPONSIBLE AUTHOR	CSI Piemonte
DATE OF DELIVERY (CONTRACTUAL)	31/03/2018 (M28)
DATE OF DELIVERY (SUBMITTED)	31/03/2018 (M28)
VERSION | STATUS	V1.0
NATURE	R(Report)
DISSEMINATION LEVEL	PU(Public)
AUTHORS (PARTNER)	CSI Piemonte

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	Draft version of the Document	29/01/2018	Luca Scanavino (CSI)
0.2	Input CSI	13-3-2018	Davide De Santis (CSI)
0.3	Input IBM	15-03-2018	Joel Nider (IBM)
0.4	Nallatech Review	21-03-2018	Gallig Renaud (Nall)
0.5	ISMB Review	27-3-2018	Alberto Scionti (ISMB)
1.0	Final Review	27-03-2018	Luca Scanavino(CSI)

PARTICIPANTS	CONTACT
STMICROELECTRONICS SRL	Giulio Urlini Email: Giulio.urlini@st.com
IBM ISRAEL SCIENCE AND TECHNOLOGY LTD	Joel Nider Email: joeln@il.ibm.com
HEWLETT PACKARD CENTRE DE COMPETENCES (FRANCE)	Cristian Gruia Email: cristian.gruia@hpe.com
NALLATECH LTD	Richard Chamberlain Email: richard.chamberlain@molex.com
ISTITUTO SUPERIORE MARIO BOELLA	Olivier Terzo Email: terzo@ismb.it
TECHNION ISRAEL INSTITUTE OF TECHNOLOGY	Dan Tsafrir Email: dan@cs.technion.ac.il
CSI PIEMONTE	Vittorio Vallero Email: Vittorio.vallero@csi.it
NEAVIA TECHNOLOGIES	Jean Hubert Wilbrod Email: jean-hubert-wilbrod@neavia.com
CERIOS GREEN BV	Frank Verhagen Email: frank.verhagen@certios.nl
TESEO SPA	Stefano Serra Email: s.serra@teseo.clemessy.com
DEPARTEMENT DE L'ISERE	Olivier Latouille Email: olivier.latouille@isere.fr

ACRONYMS LIST

Acronym	Description
VDI	Virtual Desktop Infrastructure
VU	Virtual User
OSST	Open Source Software Testing
API	Application Programming Interface
SAR	System Activity Reporter
SP	Silk Performer
CEE	Chaos Engineering Experiment
ISA	Instruction Set Architecture
RDMA	Remote Direct Memory Access
NIC	Network Interface Controller

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Position of the Deliverable within OPERA Project

This Deliverable describes the activities on the VDI Use Case during the second cycle of the OPERA project, specifically the period between M20 and M28.

For this reason, the content of the document in question is closely related to the Deliverable D7.7 *VDI Use Case 1* in which we described not only the activities and the results we accomplished during the first phase of Opera Project (M11 – M20) but also other information that are important for the second one. In this case we don't repeat those points, but we refer to them where necessary.

Therefore, we can consider as valid the considerations about the "Position of the deliverable within Opera Project" reported in D7.7, because we have the same target, the same strategy and the same relationship with the other Work Packages and other Deliverables.

Description of the Deliverable

Aiming at guaranteeing coherence and consistency with the other tasks and deliverables strictly related to D7.8, we chose this structure:

- Inputs – Starting Points, what can be used to launch the Deliverable;
- Outcomes – Results of the use case during the second phase;
- Outputs – Input of the use case for Lesson Learned and final remarks.

The main and important inputs for D7.8 are contained in the deliverable D7.7 (chapters 2 *Evolution During OPERA Project*, 3 *First Validation Cycle Details* and paragraphs 4.1 Stress Test, 5.1 *Measurements)* where there are these topics:

- The description of OPERA Project targets for each phase
- The configuration of OPERA infrastructure set up during the first phase
- The methodology to stress the environment
- The measurements in different condition of SaaS application

In these elements we find the experience and the knowledge of the first phase, but there are also other inputs that can be taken into account: the technological improvements and the measurements indications provided by WP4, WP5 and WP6.

Regarding the second Cycle concerned to the current deliverable, we provide updates on the planning, the activities completed, the current technological results, stress test methodology and measurements.
About this last topic, we considered the same type of HPE Moonshot cartridge involved during the first phase with the same SaaS application (OwnCloud) but stressing the environment with a large number of concurrent end users with a large number of concurrent end users compared to the baseline value as reported in D2.1 Initial Requirements Analysis.
During this cycle we focus our attention only on SaaS technology because we demonstrated in D7.7 that it requires less energy than RDS one.
The knowledge and the experience gained during the second phase will be used as input for the Lesson Learned.

List of Actions and Roles

Roles
Activities regarding D7.8 involve only some partners of the OPERA Consortium, whose skills, knowledge and capabilities can support the aim of VDI Use Case.

ACTIVITIES LIST AND PARTNERS ROLES	CERTIOS	CSI	HPE	IBM	ISMB	NALLATECH
Description of the current state of VDI Use Case	I	P	I	P	P	
Description of the evolution of VDI Use Case	I	P	I	P	P	
Setting up of VDI Use Case – II° Cycle	I	P	R	P	R	
Measurements about VDI Use Case – II° Cycle	R	P	P	R	R	
Structure and writing of D7.8	R	P	I	I	I	
Deliverable Review		P			P	P

- P = Participating (includes I & R)
- I = Input delivery (Includes R)
- R = review

ISMB. Technical coordination; they lead the integration of TOSCA descriptor into the adopted cloud stack solution; they contributed in defining the mechanism to trigger (also in relation to power consumption aspects) and manage the allocation of application components in the cloud stack; they contributed to the design of the main architecture for setting up the testbed and OPERA Infrastructure. They are responsible for the internal review of D7.8.

CERTIOS. It leaded the study about the definition of the baseline system according to the aim of the project, and it provided a substantial contribution about all aspects concerning power consumption.

HPE. It developed and provided the hardware to host the cloud stack software, the TOSCA descriptor and the container migration software; it contributed to defining the tool for collecting measurements and targets.

IBM. They lead the development of the software for container migration; they contribute to integrate the TOSCA descriptor; contribute to the design of the architecture to set up testbed and OPERA infrastructure and in setting up the SaaS applications on OPERA Infrastructure.

CSI. Task leader; it coordinated activities to set up the OPERA solution with the support of the other partners. It contributed in setting up the infrastructure, in defining the measurements methodology and in taking measurements.

NALLATECH. They are an additional reviewer in the internal review of D7.8.

TABLE OF CONTENTS

1 STARTING POINT

During the first cycle we took measurements of two different types of applications (RDS and SaaS) installed on OPERA infrastructure, simulating a workload of 240 concurrent end users, which is the number of a real pool of users (baseline). Thanks to this effort, we discovered that SaaS technology requires less energy than an RDS system and for this reason in the last two cycles we will consider only these type of applications.

In this section we want to highlight two aspects. The first one is the description of the new elements that we took into account during this phase. The second one is an update on the evolution of the project for the second and the third cycles.

1.1 MICROSERVICES

In this cycle of the project, we introduce the LXC technology to set up SaaS applications on containers. LXC is a container technology that is based on Linux and which provides lightweight virtualization (isolated execution environment) for applications. Specifically, we have two different environments containing microservices as shown in figure 1:

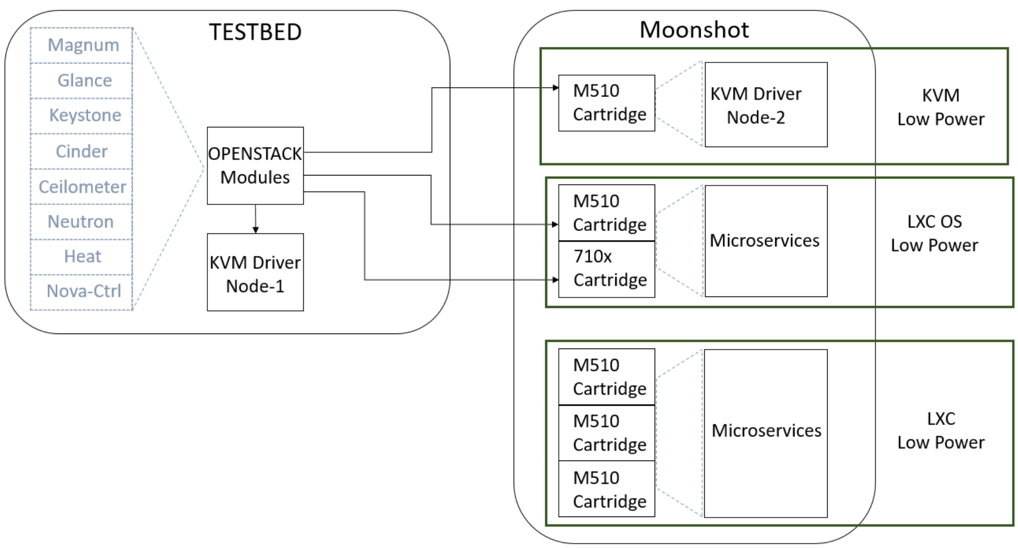

Figure 1 – SaaS Low Power Macro Architecture

The first one is indicated as *LXC OS Low Power* and consists of two cartridges (one M510 and one 710X) that host OwnCloud services on containers as *LXC Low Power* but managed by OpenStack.

The second system is indicated as *LCX Low Power* and consists of three M510 cartridges with four containers to provide the OwnCloud service.

The other components are described in paragraph 3.2 *SaaS Environment* in D7.7 and concerns the KVM technology.

1.2 MASSIVE STRESS TEST

During the first cycle we took a measurement simulating the behaviour of 240 end users, that is the same baseline value (as described in D7.7 paragraph 1.2 *Baseline*). But in this way, we create about 11% CPU load in OPERA Infrastructure as reported in D7.7 paragraph 5.1.3 *SaaS – Moonshot*, that is a value far from the initial baseline: 55% CPU load.

Considering this aspect and the best practice in real-life to maintain a CPU usage about 50%, we simulate a workload of 55% for this parameter in two different configurations:

1. KVM on Low Power server, that is the configuration described in the paragraph 5.1.3 *SaaS – Moonshot* of D7.7, where we have one M510 cartridge with KVM technologies to host OwnCloud service and indicated as KVM Low Power in figure 1;
2. LXC on Low Power server, that is the first configuration described in the previous paragraph and indicated as LXC Low Power in figure 1.

To achieve this workload, we introduced more concurrent end users than I° cycle following the methodology described in chapter 2 – *Methodology for stress test and how measure*.

1.3 UPDATE EVOLUTION DURING THE OPERA PROJECT

1.3.1 Second cycle

The macro architecture for this cycle is showed in the figure 1, we can see that LXC OS Low Power section, is very close to the II° cycle target declared in paragraph 2.2 - *II° CYCLE – TECHNICAL OUTCOMES AND BASELINE* in D7.7. In fact, there are two cartridges with the same hardware architecture with LXC managed by OpenStack. The main difference is the FPGA absence; this is due to problems with the FPGA support earlier in the project. For this reason, we have replaced it with RDMA (Mellanox ConnectX-5 NIC). The reasoning is explained in detail in D4.7 Optical Interconnect Energy Efficiency – Intermediate release.

The code to move containers among nodes using the same ISA is ready and the TOSCA descriptor is ongoing, therefore we will take measurements using this configuration the next cycle. During this cycle we consider the LXC Low Power section described in paragraph 1.1.

About the result, we maintain the same value declared in paragraph 2.2 *II° CYCLE – TECHNICAL OUTCOMES AND BASELINE* in D7.7:

$$EE_{VDI} = 48 \text{ users / kWh}$$

1.3.2 Third cycle

During the last cycle, as showed in the figure below, we want to achieve the cross-ISA configuration using the Mellanox ConnectX-5 card (and no longer the FPGA-based interconnection technology) supporting the RDMA protocol to move containers through the post-copy migration method.

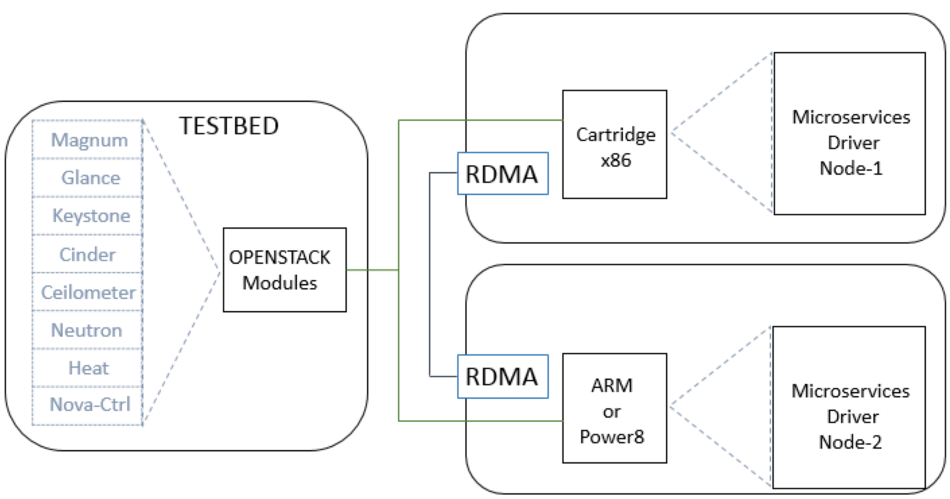

Figure 2 – III° Cycle – Macro Architecture

In particular, we will move container between one X86_64 cartridge and another server with ARM or POWER based processor architecture. About the result, we can maintain the same value declared in paragraph 2.3 - *III° CYCLE – TECHNICAL OUTCOMES AND BASELINE* in D7.7:

$$EE_{VDI} = 57 \text{ users / kWh}$$

2 METHODOLOGY FOR STRESS TEST AND HOW TO MEASURE

The main request for this cycle of testing is obtaining power consumption related to CPU metric with a load heavier than in the last cycle. The target is loading more than 240 Virtual Users (VUs) on ownCloud software deployed within the systems but using different technologies: KVM and LXC. A Virtual User replicates the behaviour of a real end user by simulating it with a dedicated software.

The steps followed to this new goal are typical of Test Management and consist of:

- Risk analysis
- Monitor strategies
- Test strategies
- Test activities: recording scripts, parametrization, workload definition, executions, results analysis, and reporting
- Spreading results with stakeholders

Some parts of the test activities are discussed inside the 'test strategies' paragraph, while 'reporting and spreading results' are omitted because they are out of the scope.

2.1.1 Risk Analysis

The goal: Measurement energy consumption in SaaS environments (VDI) of ownCloud software under certain load conditions.

Risks identified: High numbers of VUs, ownCloud tuning with a lot of time in troubleshooting and configuration, open source software reliability with higher number of VUs

The following table indicates the levels of risk identified (qualitative and quantitative) and if their mitigation is possible in a short time and with limited resources:

N°	Testing Tools	Likelihood (L)	Impact (I)	Risk (L*I)	Mitigation
1	Only 900 VUs	Higher (4)	Higher (4)	16	**NO**
2	Open Source Reliability	High (3)	Higher (4)	9	**Partially**
3	ownCloud Tuning	Medium (2)	High (3)	6	**Partially**

Table 1 Measurements - Risk Analysis

The risk n° 1 is the highest one and without mitigation because:

1. CSI Piemonte is owner only of 900 VUs for its stress test solution, Silk Performer (SP), a commercial software licenced by number of VU.
2. The purchase of licences for more VUs for a limited time is not practicable, because of high cost (tens of thousands of euro for less than a month) that is out of project scope.

The risk n° 2 is high though less than risk n° 1 and mitigation is possible but on condition that:

- scouting Open Source Software Testing (OSST) to identify right tools for this activity;
- verify the reliability of the identified tools with high load (>1000 VUs) whether with concurrent VUs or during ramp up;
- reports must be easy in reading and spreading;
- outcomes must be comparable with real life situations (for instance hit/s and throughput)

A mitigation is possible also for risk number 3 on condition that:

- the setup of each system has to guarantee the same standard parameters and 100 concurrent clients as max number of clients on ownCloud
- the ownCloud needs to be compiled in "mpm_prefork_work"
- data base configuration must be tuned
- apache2 configuration must be tuned

2.1.2 Monitor Strategies

In the previous cycle, we took measurements using a script solution (implemented ad-hoc), that is good up to 300 VUs; however, beyond this value there is hardware overloading and no data collection is possible.

For this reason, we designed, developed and implemented a solution, whose schema is showed in the following figure, that is composed of a bash scripts console, Moonshot API, Apache2 Server-Status and SAR metrics.

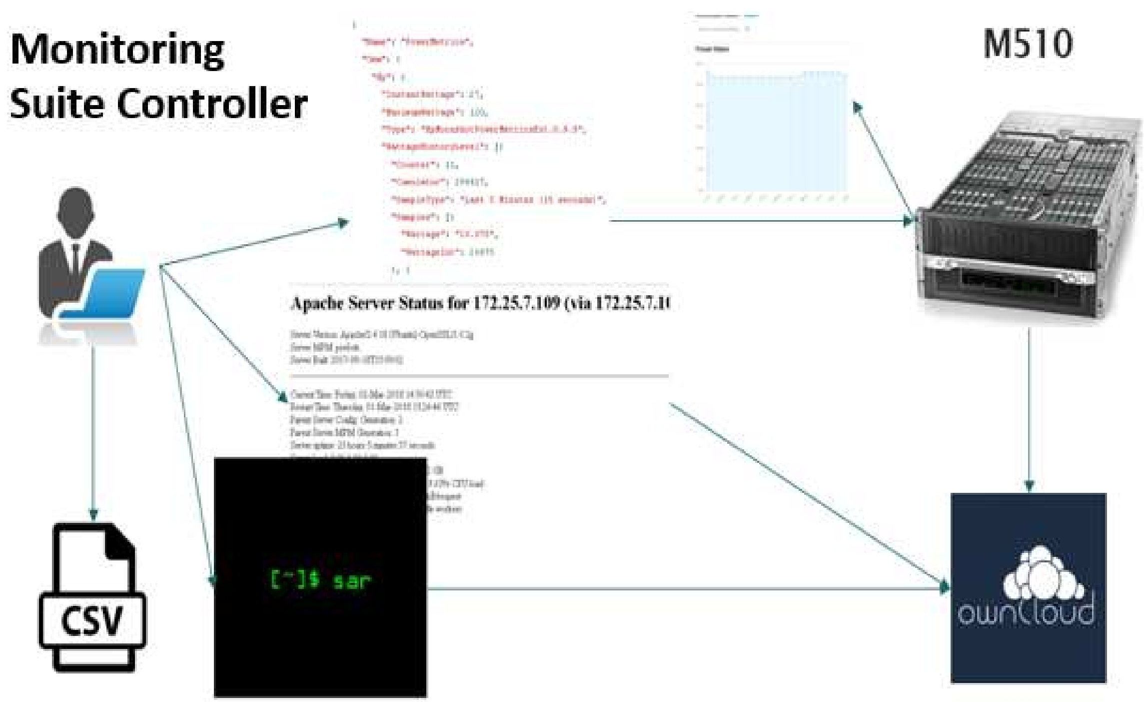

Figure 3 – Metrics Collection Suite - Schema

This new metrics collection suite must be parametrized with elements like frequency and duration of collection and can provide instant measurements and save outcomes (CVS format), in addition it is easy to maintain and reusable for future projects.

2.1.3 Test Strategies

Silk Perform allows testing easily OwnCloud, but we have a limited number of VUs as discussed in the Risk Analysis Paragraph.

An overview on open source performance tools showed that the number of VU limitation could be satisfied, but the test environment resources, ramp up definition and metrics collection are activities more complex than Silk Performer ones.

For this reason, we selected three different strategies to achieve the target.

1. The Strategy 1 uses SP until the maximum number of VUs and continues with prediction method (data forecasting) and follows this schema:

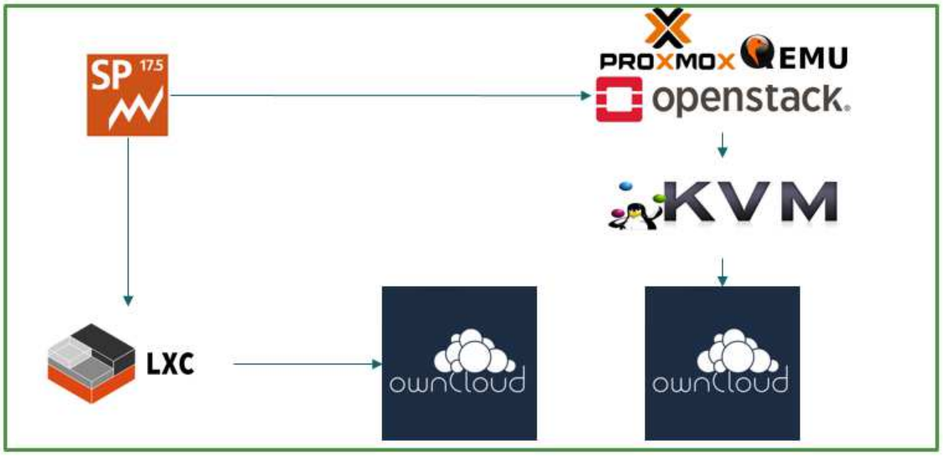

Figure 4 - Stress Test – Strategy 1 - Schema

In the figure, we can see that SP simulates the workload on the two different installations of ownClowd (LXC and KVM).

This strategy has a problem, with more than 500 VUs, because ownCloud is not able to respond properly to the requests and we have too many errors (connections lost, connections timeout), that means much more 3% of total requests are not managed.

This problem can be partially solved intervening on mpm_prefork file configuration of ownCloud, but the complete tuning requires an effort and skills that are out of the project scope. This strategy cannot be the final solution.

2. The Strategy 2 involves open source tools (Gatling and Jmeter) to stress both ownCloud environments as reported in the following schema:

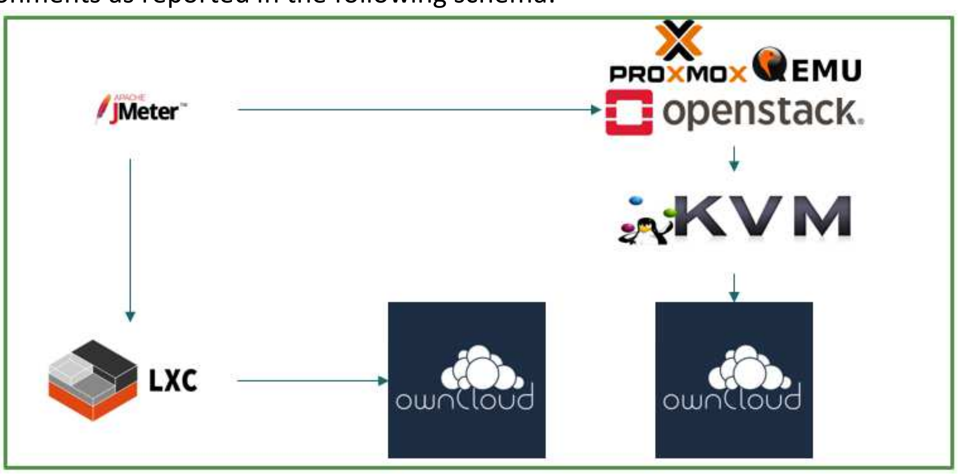

Figure 5 - Stress Test – Strategy 2 - Schema

This strategy has a problem, because both tools cannot introduce VUs gradually and canot create a ramp up like SP despite being declared as one of their features. We analyzed the software documentation and involved also the community, but we did not define a way to collect data uniformly and injecting VUs gradually.

It is out of project scope to invest too many time about studying the code and developing workaround. This strategy cannot be the final solution.

3. The Strategy 3 is an evolution of the first one and aggregates SP and the discipline called Chaos Engineering (CE), as reported in the following figure.

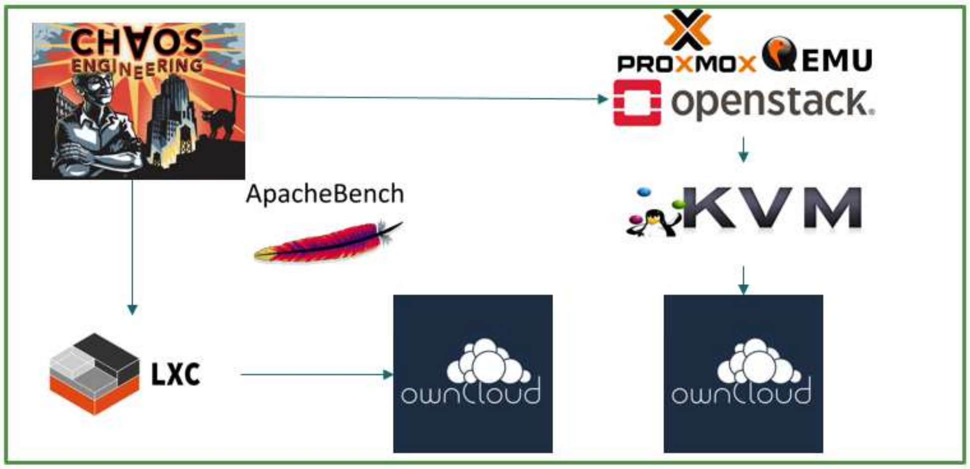

Figure 6 - Stress Test – Strategy 3 - Schema

At the beginning, we use SP with different values of VUs for both technologies, starting from 200 VUs and increasing with steps of 100 VUs, until we have more than 3% of connection errors, that happens with 500 VUs. After 400 VUs, due to connection errors, both workload and power consumption decreased as showed in the following figure. Thanks to that, we know that the start point of prediction is 400 VUs, after this value we apply prediction method that means a linear projection, to find out when we achieve 55% CPU load (the established target in the previous chapter), as described in following figure:

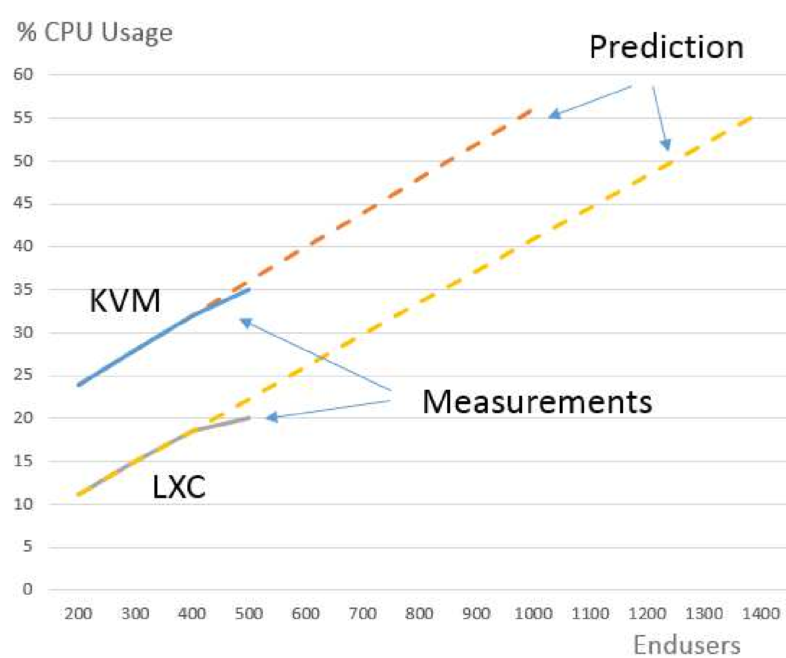

Figure 7 - Stress Test – Strategy 3 – % CPU usage prediction

From the previous figure we can see that for KVM we need 1000 VUs to achieve 55% CPU load, instead for LXC 1400 VUs.

The same hypothesis can be applied also for power consumption, as reported in the following figure:

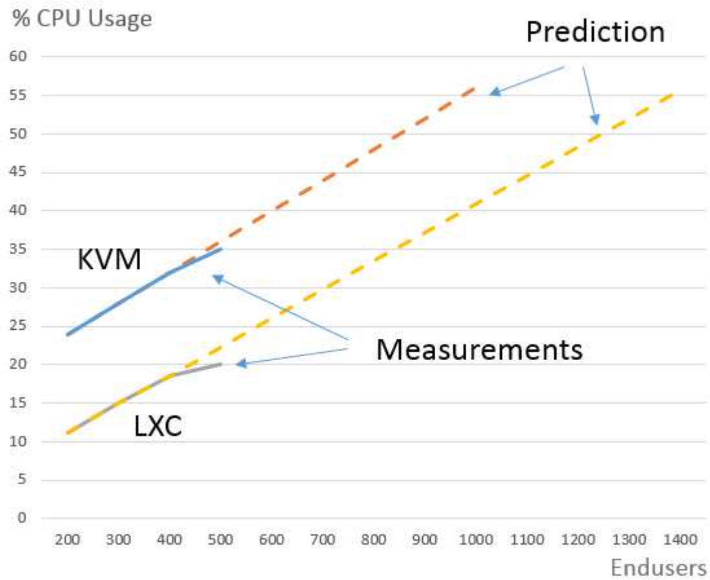

Figure 8 - Stress Test – Strategy 3 – Power consumption prediction

From the figure, we can see that for 1000 VUs with KVM we need 105 W and for 1400 VUs with LXC we need 117 W.

After that, we apply the CE discipline that means defining an hypothesis and doing a CE Experiment (CEE). The experiment consists in loading all CPUs of the systems until 55% (total average ~55%) and measuring the power consumption during experiment to confirm the load. This load covers the amount of load of the predictive method and it is injected with scripts tuned to obtain 55% of CPU load calling the 2 ownCloud servers with an Apache Benchmarking technic. The direct injection on the systems avoids some problems like network latency, number of errors, traffic slowed from firewall. The experiment takes 80 seconds and is the described in the following figure:

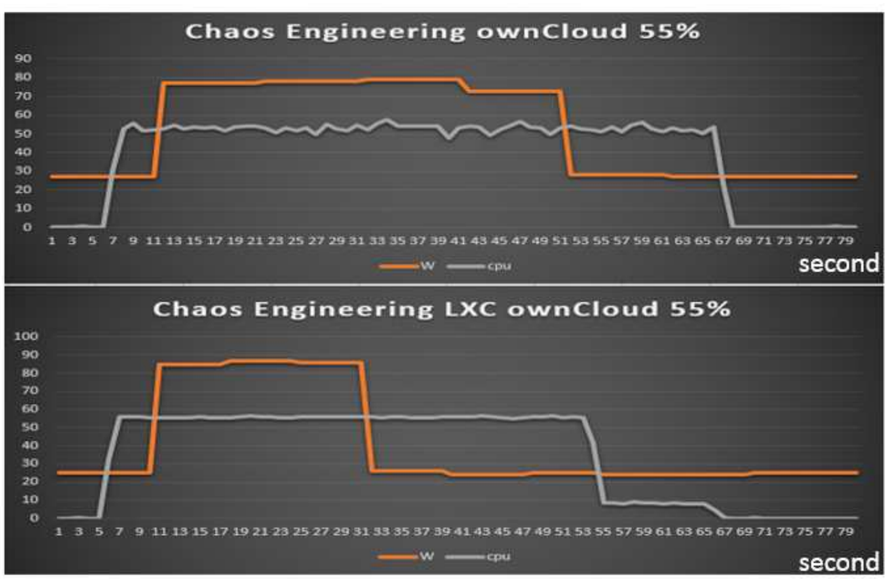

Figure 9 - Stress Test – Strategy 3 – CEE

From the figure, in the first 30 seconds (hits/s starts at 5”) we inject load on each system at the same time, while from 31” the traffic injection stops and the VU ramp down (throughput/s stops at 69”).

The experiment terminates within 80” when all traffic is processed.

The two graphs show the parameters (Watt and CPU) in the two systems at the same time. We can see that LXC is better than KVM about these aspects:

1. CPU usage more steady
2. Lower average power consumption

About the last topic, we can highlight that, at the beginning of the experiment, LXC requires more power (~86W instead of ~78W) but can respond to all the requests faster than KVM, indeed the peak duration is shorter than KVM (20’’ vs 40”).

The tool provides also values for a set of parameters (see chapter 5.1), one of them is “Time per request” and for KVM we obtain 2.704 ms, indeed for LXC 1.416 ms. That means LXC is faster to answer requests than KVM.

As expected, we can consider the prediction method correct and the hypothesis is confirmed, in fact, the measurements are more conservative than prediction and proves the different behaviour of LXC and KVM also with more VUs.

3 II° CYCLE DETAILS

In this chapter we describe the measurements that we did in this cycle and how they can be useful for OPERA project.

As introduced in the previous chapter we need to analyse the behaviour of OPERA Infrastructure also with a large number of concurrent end users, to achieve the same % CPU usage as the Baseline following the methodology reported in chapter 2 *Methodology for stress test and how measure*.

In addition, during this phase we introduce also another technology element, microservices, thanks to LXC technologies. In this way it will be possible to compare the relative energy consumptions of KVM and LXC.

3.1 MASSIVE STRESS TEST – KVM

In this section, we consider the same environment described in paragraph 3.2 – *SaaS Environment* and paragraph 5.1.3 – *SaaS Moonshot* in D7.7, but simulating enough concurrent end users to obtain 55% CPU usage.

As we can see from the following figure, when the % CPU usage achieves the target value there are 1,000 concurrent end users.

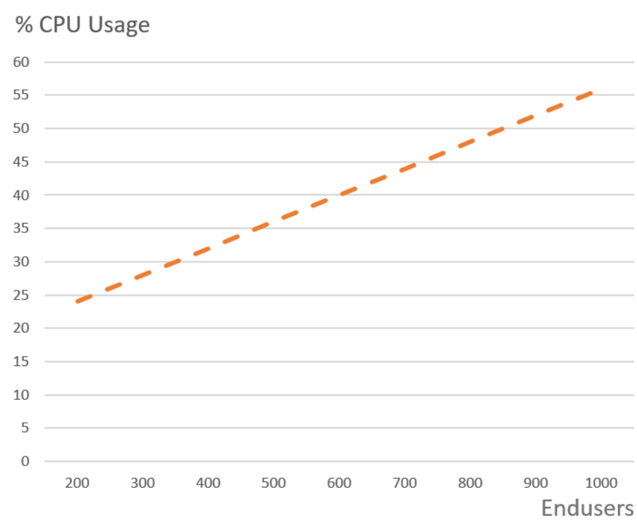

Figure 10 – KVM – CPU Usage

This specific value of concurrent end users requires 105 W as reported in the following figure:

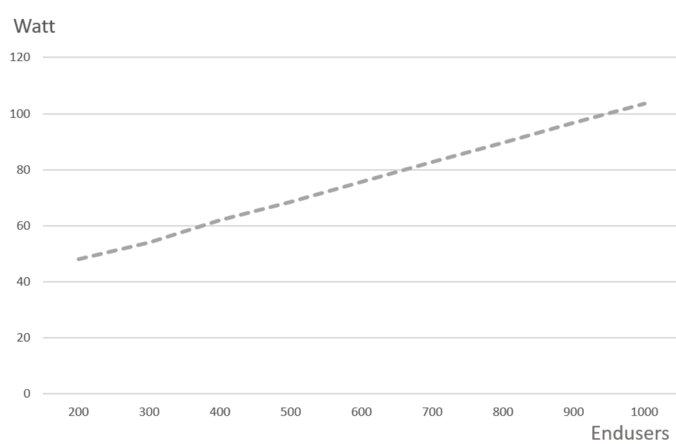

Figure 11 – KVM – Power Consumption

In addition, as described in in paragraph 3.3 *Virtual Desktop Use Case* in D2.1, we can consider that over a week, 64% of the time the system is in idle condition (no active end users) and the rest of the time (36%) in active condition (1,000 concurrent end users).

In idle condition, the power consumption is 26 W as reported in paragraph 5.1.3 *SaaS Moonshot* of D7.7.

Thanks to these values is possible to evaluate the average power consumption and the Energy Efficiency according the formula described in D4.1 *Report on energy efficiency metrics*:

Average Power Consumption – KVM = (26 W * 64%) + (105 W * 36%) = 54.4 W

Energy Efficiency – KVM = 109,33 users / kWh

3.2 MASSIVE STRESS TEST – LXC

In this paragraph, we consider the LXC Low Power configuration, involving only one of the three cartridges. For this measurement we follow the same pattern described in the previous paragraph, we simulate enough concurrent end users to achieve 55% CPU usage.

Thanks to the measurements reported in the figure below, we can see that we obtain 55% CPU usage with 1,400 concurrent end users.

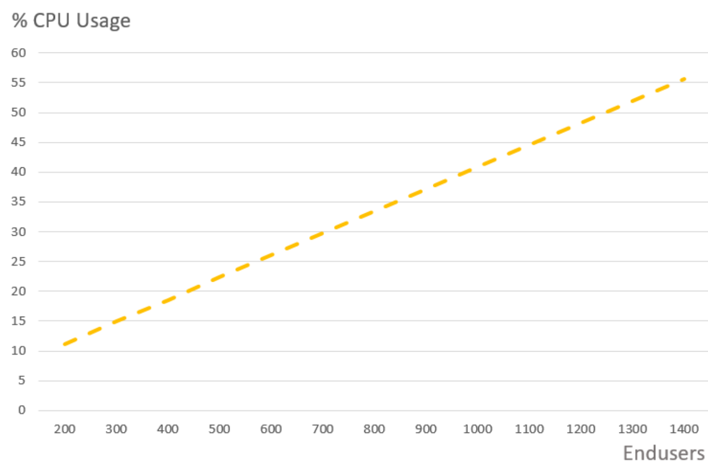

Figure 12 – LXC – CPU Usage

When we simulate this number of concurrent end users, we have 117W as power consumption. It is possible to appreciate that in figure 13.

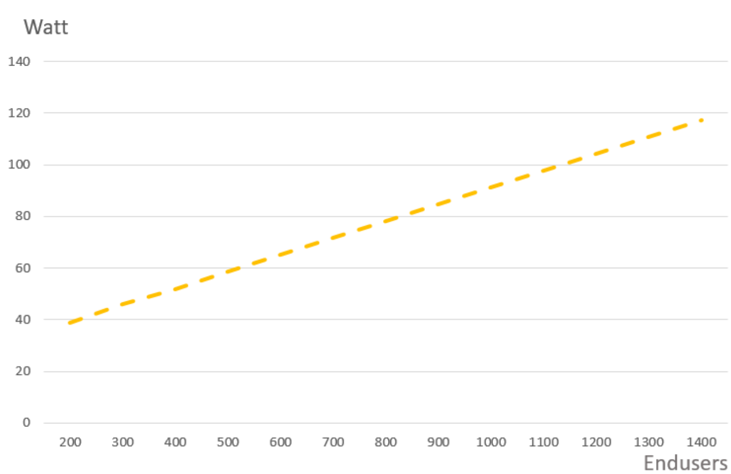

Figure 13 – LXC – Power Consumption

In addition, we can consider the same distribution over a week between idle and active periods and a reference of 26W as power consumption in the idle state.

Thanks to these values is possible to evaluate the average power consumption and the Energy Efficiency according the formula described in D4.1 *Report on energy efficiency metrics*:

Average Power Consumption – KVM = (26 W * 64%) + (117 W * 36%) = 58.7 W

Energy Efficiency – KVM = 141,8 users / kWh

4 OUTPUTS

In this chapter, we want to highlight the results that we gained during this cycle, reporting them from different point of views:

- Real-life: the activities and the results, obtained in this cycle, show that by introducing new technological paradigm it is possible to reduce energy consumption;
- OPERA project: even if we did not achieve all the targets planned in D7.7 in terms of activities, we achieved good results in terms of energy efficiency;
- Lesson learned: during this cycle we gained experience that can be reported and reused.

Thanks to all these aspects, it is possible to find out the Final Remarks for II° Cycle.

4.1 REAL-LIFE RESULTS

The configuration, described in the paragraph 1.1 – *Microservices* about LXC Low Power, is the current (last) configuration of OPERA and even if it is not the final one, thanks to that we can demonstrate LXC is better than KVM for many reasons. First of all, the measurements reported in chapter 3 - *II° Cycle details*, show that with the same CPU usage on the same type of cartridge it is possible to provide the same service (OwnCLoud) for more concurrent VUs (+40%).

In addition, thanks to chapter 2 – *Methodology for stress test and how measure*, we can see that LXC can respond faster than KVM to VUs twice as fast as KVM.

These results demonstrate that it is better to invest in LXC technology than KVM, because we can optimize the workload and provide services more efficient, indeed data (see paragraph 5.1) confirm the trend of prediction method, the LXC server works 1.9 times better than VMs in terms of response time.

4.2 OPERA PROJECT

According to the planning described in chapter 2 – Evolution during OPERA project, we should have completed more activities (cross-ISA migration, decomposition in microservices of Libre Office and Unified Communication, TOSCA descriptor integration). Nevertheless, we can see in the table below that over the project we improved a lot (more than 20 times) the SOTA solution (CSI Citrix Environment) described in chapter 3.3.7 Baseline in D2.1:

Environment	CPU Usage	N° Concurrent Endusers	EE
CSI Citrix	55%	240	7
KVM – Testbed – Traditional server	65%	240	9.9
KVM – Low Power – Low Power Server	55%	1000	109,33
LXC – Low Power Server	55%	1400	141,8

Table 2 - EE comparison

We elaborated the last two values about KVM and LXC Low Power server in the previous chapter. Here, we can evaluate the energy efficiency about CSI Citrix and KVM Testbed thanks to the values reported in paragraph 3.3.7 – *Baseline* in D2.1 and paragraph 5.1.2 *SaaS – Testbed* in D7.7:

$$EE_{Citrix} = 240 \text{ endusers} / [(0{,}18 \text{ kW} * 107{,}52 \text{ h}) + (0{,}24 \text{ kW} * 60{,}48)] = 7$$

$$EE_{Testbed} = 240 \text{ endusers} / [(0{,}12 \text{ kW} * 107{,}52 \text{ h}) + (0{,}185 \text{ kW} * 60{,}48)] = 9.9$$

We achieved this result by intervening in two different ways. The first one is the use of low power servers and the second one is the service containerization.

According to these values and future improvements expected in the last cycle of OPERA project, we need to review the III° cycle target, that means to consider anymore the value reported in 1.3.2 paragraph:

$$EE_{VDI} = 57 \text{ users} / \text{kWh}$$

But, instead, a higher value:

$$EE_{VDI} = 145 \text{ users} / \text{kWh}$$

4.3 INPUTS FOR LESSON LEARNED

During this phase of the project, even if we did not achieve the planned results declared in previous cycle, we gained experience that could be used in the future for the project and for other real-life contexts.

The main aspect is the use of LXC, because with the service containerization instead of KVM, it's possible to obtain two improvements:

1. CPU workload optimization: we can guarantee the same services to more end users;
2. Services are more efficient: without the additional workload due to virtualization it is possible to reduce the response time.

4.4 CONCLUSIONS

During this cycle, we spent a lot of effort to find out with which card to use to move containers, and for this reason we did not complete all the planned activities but we are close to that, and we think to achieve both these tasks and the III° cycle expected ones.

Despite the delay in providing and testing the software to optimize the workload distribution, we improved 20 times the energy efficiency parameter, introducing low power server and LXC technologies. In addition, the good results in terms of energy efficiency led us to review the III° cycle target.

These observations can lead the IT provider technological choices because with a limited effort (that means involving low power server and studying LXC) it is possible to improve services.

5 ANNEX

5.1 APACHE Benchmark results of CEE

In the following we have the values about different parameter evaluate with Apache Benchmark both for KVM and LXC.

KVM:

Concurrency Level: 19

Time taken for tests: 60.002 seconds

Complete requests: 22190

Requests per second: 369.82 [#/sec] (mean)

Time per request: 51.376 [ms] (mean)

Time per request: 2.704 [ms] (mean, across all concurrent requests)

Connection Times (ms)

	min	mean	[+/-sd]	median	max
Connect:	2	4	1.6	4	35
Processing:	25	47	18.5	43	353
Waiting:	23	45	18.0	41	351
Total:	27	51	18.8	47	356

LXC:

Concurrency Level: 19

Time taken for tests: 60.001 seconds

Complete requests: 42387

Requests per second: 706.44 [#/sec] (mean)

Time per request: 26.895 [ms] (mean)

Time per request: 1.416 [ms] (mean, across all concurrent requests)

Connection Times (ms)

	min	mean	[+/-sd]	median	max
Connect:	0	0	0.1	0	2
Processing:	12	27	8.4	26	79
Waiting:	12	26	7.8	25	77
Total:	12	27	8.4	27	81