Estimating Energy Impact of Software Releases and Deployment Strategies

Verdecchia, Roberto; Procaccianti, Giuseppe; Malavolta, Ivano; Lago, Patricia; Koedijk, Joost

published in
2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) 2017

DOI (link to publisher)
10.1109/ESEM.2017.39

document version
Publisher's PDF, also known as Version of record

Estimating Energy Impact of Software Releases and Deployment Strategies: the KPMG Case Study

Roberto Verdecchia*†, Giuseppe Procaccianti †, Ivano Malavolta†, Patricia Lago†, Joost Koedijk‡
*Gran Sasso Science Institute, L'Aquila, Italy - roberto.verdecchia@gssi.it
†Vrije Universiteit Amsterdam, The Netherlands - { g.procaccianti | i.malavolta | p.lago }@vu.nl
‡KPMG, The Netherlands - koedijk.joost@kpmg.nl

***Abstract*—Background. Often motivated by optimization objectives, software products are characterized by different subsequent releases and deployed through different strategies. The impact of these two aspects of software on energy consumption has still to be completely understood and can be improved by carrying out *ad-hoc* analyses for specific software products.**
Aims. In this research we report on an industrial collaboration aiming at assessing the different impact that releases and deployment strategies of a software product can have on the energy consumption of its underlying hardware infrastructure.
Method. We designed and performed an empirical experiment in a controlled environment. Deployment strategies, releases and use case scenarios of an industrial third-party software product were adopted as experimental factors. The use case scenarios were used as a blocking factor and adopted to dynamically load-test the software product. Power consumption and execution time were selected as response variables to measure the energy consumption.
Results. We observed that both deployment strategies and software releases significantly influence the energy consumption of the hardware infrastructure. A strong interaction between the two factors was identified. The impact of such interaction highly varied depending on which use case scenario was considered, making the identification of the most frequently adopted use case scenario critical for energy optimisation. The collaboration between industry and academia has been productive for both parties, even if some practitioners manifested low interest/awareness on software energy efficiency.
Conclusions. For the software product considered there is no absolute preferable release or deployment strategy with respect to energy efficiency, as the interaction of these factors has to be considered. The number of machines involved in a software deployment strategy does not simply constitute an additive effect of the energy consumption of the underlying hardware infrastructure.

***Index Terms*—Energy, Software Releases, Deployment.**

I. Introduction

Software-intensive systems are taking up an ever-growing part of our overall energy consumption and a considerable amount of resources is nowadays allocated to power IT infrastructures [23]. According to the constant growth of digital content, big data and cloud computing, software systems are reported to be among the fastest-growing consumers of electricity [17]. The question to be answered is therefore: *how can the impact of IT on energy consumption be reduced?*

A possible solution lies in the continuous advancement of hardware technologies. According to *Koomey's Law*, the number of computations per Joule of energy dissipated is expected to double approximately every two and a half years. This trend has been remarkably stable through the years [11]. Energy consumption of software systems, in divergence with the energy efficiency improvements reflected by *Koomey's Law*, still increases linearly year after year. The cause of this trend can be attributed to the concept expressed by Wirth with his prominent computing adage *"Software is getting slower more rapidly than hardware is getting faster"*. In other words, while the performance of computations increases year after year, due to the high confidence in constant hardware improvements, the number of computations needed to fulfil a task increases even more. This trend, first noticed about 30 years ago [25], is usually referred to as *software bloat* [16].

In this paper we aim to empirically assess the extent to which different releases of an industrial software product implementing the same functional requirements in different/optimized ways, impact the energy consumption at the infrastructure level. The choice of choosing similar releases, i.e. releases implementing the same functional requirements, was led by the possibility to compare these releases by considering the same use case scenarios to be executed. The interest of the industrial partner on particular releases also contributed to the selection process.

This research stems from a collaboration between the Vrije Universiteit Amsterdam and KPMG, a professional service company based in Amstelveen, The Netherlands, with more than 189,000 employees. The underlying goal of both parties is to understand to what extent releases of a software product (that do not drastically differ in terms of functional requirements) can influence in an appreciable way the energy consumption of the underlying hardware components, leading to the selection of the most convenient to adopt w.r.t. energy consumption. Also, we assess the impact of the different deployment strategies available for the software product on the hardware energy consumption. In fact, while the effects of deployment strategies on energy consumption (and possible optimisation techniques such as server consolidation heuristics [21]) have been researched, *ad-hoc* analyses can be utilised to assess the specific behaviour of industrial software products.

The **goal** of this paper is hence to assess the different impact that software releases and deployment strategies of an industrial software product can have on the energy consumption of the underlying hardware. This enables the selection of the combination of version and deployment strategies available

978-1-5090-4039-1/17 $31.00 © 2017 IEEE
DOI 10.1109/ESEM.2017.39

for the industrial software product that enable its best energy efficiency. It is important to notice that, in order to load test the application, a set of use case scenarios encompassing the core functional requirements of the industrial software product considered has been adopted.

In order to achieve the aforementioned goal, we designed, executed, and reported on an empirical study that considers different use case scenarios of a professionally developed software product called Qubus [12]. Qubus is a platform for supporting governance, risk, and compliance processes (GRC) developed by KPMG. The main **contributions** of this paper are: (i) the description of a well-defined process to measure the energy consumption of an industrial software product; (ii) the results of an experiment on the impact of different software releases and deployment strategies on the energy consumption of a professionally developed software product; (iii) the discussion of the obtained results and their implications; (v) a replication package containing and the raw data of the experiment and the scripts for data analysis.

The **target audience** of this paper is composed of both academics and practitioners interested in measuring the energy efficiency of a software product. Our results also provide an overview on the impact that different releases and deployment strategies of software products can have on energy consumption, and hence help in reasoning about which strategy should be preferred from an energy optimization point of view.

The paper is organized as follows. In Section III the object of the study, namely the Qubus application, is presented. Section IV reports the design of the experiment we performed. Section V provides the details of the experiment execution, while Section VI reports on the results of the analysis of the data gathered from the experiment. An in-depth discussion of the obtained results is provided in Section VII. Section VIII presents and evaluates the threats to validity of the experiment. Section II discusses related work, whereas Section IX closes the paper and discusses future work.

II. Related Work

Through an empirical study, Sahin et al. [19] provided evidence that refactoring operations significantly influence energy efficiency. In order to ease refactoring of identified energy hotspots, Manotas et al. [15] developed a decision framework to support code-level energy optimisation refactoring. By taking into account mobile applications, several approaches aimed to identify energy-related issues, bugs, and hotspots at different levels of abstraction have been proposed [18], [17], [14].

Regarding the impact of different software releases on energy consumption, to the best of our knowledge only few researches addressing this study can be found in the literature. An empirical research conducted by Jagroep et al. [10] demonstrated how the energy consumption of different software releases could be quantified down to the level of individual processes. In a study by Hindle [9] a methodology aimed to relate software change to energy consumption is proposed. The method was then validated through two case studies. As for the experiment proposed in this research, the energy consumption highly depended on the adopted use case scenario.

III. Industrial case study: The Qubus solution

As shown in Figure 1, the problem to be addressed was jointly formulated in a similar fashion of the *technology transfer model* of Gorschek et al. [8]. The identification of what characteristics of *Qubus* to focus on was then carried out by the industrial party and validated with respect to the predefined goal by the academic one. The selection of the concrete factors was then carried out by the industrial partner, according to their specific interest. The academic researchers then conducted an in-depth analysis of the software product according to the selected factors and underlying goal. During this phase, the interaction with the industrial partner was not paused, but enforced whenever clarifications about the software product or refinements of the research goal and factors were required. Finally, after the empirical experiment was carried out, the results were presented by the researchers and jointly discussed with the industrial partner.

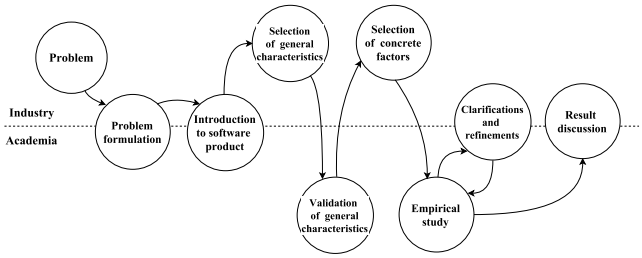

Fig. 1. Interaction between academia and industry for this study

The object of this study is a platform named *Qubus*, which supports Governance, Risk, and Compliance (GRC) processes and is developed and maintained by KPMG. The selected software product, based on dynamic and customizable content, supports a wide range of enterprise management processes including financial control management, strategic and operational risk management, compliance management and external audit management. The application implements these processes through a set of personalized questionnaires to be filled in by multiple (and possibly concurrent) users. Users are also provided with the ability to implement and configure ad-hoc structured sets of dynamically linked questionnaires, aimed to capture evidence for customized GRC processes. The data gathered through questionnaires is then automatically processed by the platform and presented by graphical and textual means with dashboards and custom-built reports.

Figure 2 provides the high-level architectural view of the software components of the Qubus application and its functionalities. The functionalities of the Qubus platform are grouped in three distinct clusters according to their characteristics. In a first cluster the functionalities that formed the end-user experience are grouped. The services of this cluster are all accessible by the end-user through dynamically generated web pages based on ASP.NET. This category of functionalities

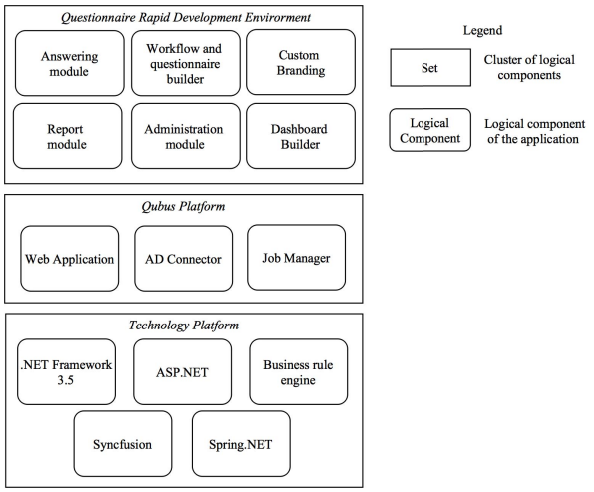

Fig. 2. Qubus high-level architecture view

are part of the logical component named *Questionnaire Rapid Development Environment*. Some of the most prominent modules that fall under this category are: the *Answering module*, the *Workflow and Questionnaire builder*, and the *Dashboard builder*.

The second layer (*Qubus Platform*) is represented by the three applications composing the Qubus platform. The *web application* is the main, web-based, component the Qubus platform, implementing all the logic required to build and use the tools of the platform. The *AD Connector* is a Windows service managing the integration of the *Web application* with the AD directory service of the Windows Server on which the database of the application is deployed. Finally, the *Job manager* module represents an independent Windows service application taking care of sending notifications to users, automatic transition of a questionnaire to other states and automatic assignment of the end-users to questionnaires.

The third layer (*Technology platform*) is composed of the technologies on which the Qubus application relies. The components of this last group are frameworks and engines common to many .NET projects, such as the ASP.NET framework, utilized to create dynamic event-driven web pages.

IV. Experiment Plan

In order to provide objective and replicable results, we planned our experiment by following well-known guidelines on empirical software engineering [26], [20]. In the remainder of this section we report on how we planned the experiment.

A. Goal and Research Questions

The defined **goal** of the experiment was agreed upon by following the Goal-Question-Metric approach [5] and is as follows: *"Evaluate **releases and deployment strategies** for the purpose of **determining their impact** on **energy consumption** as seen from the viewpoint of the **software engineer**, in the context of the **Qubus application**".*

From the goal of the experiment we derived the two **research questions** of our study, which are the following:

RQ1 - *What is the impact of different software releases on hardware energy consumption?*

RQ2 - *What is the impact of software deployment strategies on hardware energy consumption?*

B. Variables Selection

In this section we discuss the identified dependent and independent variables of the experiment, together with a description about the experiment design.

1) Dependent variables: In order to answer each of the above stated research questions the energy consumption of the hardware infrastructure on which the industrial software product is running had to be measured. This was possible by keeping track of two distinct dependent variables, namely:

- **Power consumption**: the power required by the underlying hardware infrastructure in order to run the application, measured in Watts;
- **Execution time**: the time required by the application to execute a predefined use case scenario, measured in seconds.

These two dependent variables are required in order to measure the **energy consumption**, that constitutes a derived dependent variable. This latter variable is calculated as $E = P \cdot \Delta\ t$, where P is the average power consumption during the execution time Δt.

2) Independent variables: Three distinct factors were selected, namely the *Qubus release*, *deployment strategy* (DS) and *use case scenario* (UCS). The rationale behind the selection of the first two factors was guided by the interest of the industrial partners to understand the impact that these easy to deploy changes might have on energy efficiency. In addition, UCS was adopted as independent variable in order to simulate realistically the common usage of the software product. The factors and their corresponding treatments are detailed as follows:

• **Qubus release**: the *Qubus 6.2.3 release* and *Qubus 6.5.7 release* were selected for the experiment, we will refer to them as R_1 and R_2, respectively. These two releases can be considered as equivalent from both the architectural and implemented features perspective, making them very good candidates for our experiment due to their manageable comparability w.r.t. UCS;

• **Deployment strategy**: the Qubus application can be deployed through two main strategies. In the first one, both the database and the web application are hosted on a single machine, we refer to it as *centralized deployment strategy* (D_c). In the second deployment strategy the database and the web application are deployed on two dedicated machines, we refer to it as *distributed deployment strategy* (D_d);

• **Use Case Scenario**[1]: we identified two different use case scenarios (UCS) containing the core functionalities of the

[1]For an in-depth description of the UCS we refer the reader to the online replication package provided for this study.

Qubus application. The first UCS consists of the most common UCS of the Qubus application, namely the completion of a questionnaire by filling in its fields; we refer to it as *questionnaire completion* (U_1). The second UCS represents the steps required in order to generate reports from the information gathered through the questionnaires; we refer to it as *report generation* (U_2).

C. Hypotheses Formulation

By formalizing the research questions presented in Section IV-A, and by taking into account the previously presented factors, we can derive the hypotheses underling our experiment as follows:

1) Hypotheses for the releases:

- **Null hypothesis** H_0^1: There is no significant difference in energy consumption between the two Qubus releases considered.

$$\mu_{R_1} = \mu_{R_2} \tag{1}$$

- **Alternative hypothesis** H_a^1: There is a significant difference in energy consumption between the two Qubus releases considered.

$$\mu_{R_1} \neq \mu_{R_2} \tag{2}$$

2) Hypotheses for the deployment strategies:

- **Null hypothesis** H_0^2: There is no significant difference in power consumption between the two deployment strategies utilized.

$$\mu_{D_c} = \mu_{D_d} \tag{3}$$

- **Alternative hypothesis** H_a^2: There is a significant difference in power consumption between the two deployment strategies utilized.

$$\mu_{D_c} \neq \mu_{D_d} \tag{4}$$

It is important to notice that the UCSs U_1 and U_2 are adopted as a *blocking factor*, hence the hypothesis are tested independently for the two datasets associated to the UCSs.

D. Experiment Design

We purposefully kept the number of factors and treatments low in this experiment so that the exhaustive set of all possible combinations of treatments can be tested. This enables us to gather comprehensive results by taking into account the totality of the 3-way interactions between treatments.

The UCS factor is used as a blocking factor for the experiment, i.e. the trials are divided into two different sets (or blocks) according to the UCS that characterizes them. The values gathered through these two sets are then analyzed independently in order to isolate the effect of the blocking factor. In order to avoid a possible threat to conclusion validity two representative UCS of the application considered are selected for our experiment. Tables I and II report the complete test suite of our experiment.

TABLE I
EXPERIMENTAL TEST SUITE U_1

		Factors & Treatments	
		Qubus release	*Deployment Strategy*
Trials	Trial 1	R_1	D_c
	Trial 2	R_1	D_d
	Trial 3	R_2	D_c
	Trial 4	R_2	D_d

TABLE II
EXPERIMENTAL TEST SUITE U_2

		Factors & Treatments	
		Qubus release	*Deployment Strategy*
Trials	Trial 5	R_1	D_c
	Trial 6	R_1	D_d
	Trial 7	R_2	D_c
	Trial 8	R_2	D_d

E. Analysis Methodology

All of the hypotheses presented in Section IV-C will be tested via two-tailed tests, and a significance level (α) of 0.05 is utilized to evaluate the distribution of the empirically gathered values. Due to the variability of the values gathered with the dynamic analysis, the execution of the UCS (also referred further to as *"run"*), are carried out 30 times for each combination of factors in order to gather enough statistically sound data. This leads to the identification of an accurate mean value and to the calculation of the standard deviation for each of the experimental measurements carried out for each combination of factors.

T-tests are carried out in order to compare the means of the independent samples. A significance level (α) of 0.05 is used to analyze the differences between the dynamically gathered values. Furthermore, the variation within and between the treatments and the independent variables is tested via the well-known Analysis of Variance (ANOVA) statistical model [7]. In order to evaluate the heteroscedasticity of the linear regression models of the independent variables Breusch-Pagan tests [4] are utilized. The significance level (α) of these tests is in line the previously set value of 0.05. A post hoc analysis is carried out in order to detect patterns, useful to answer the research questions that were not specified *a priori*. This process is carried out by adopting the Tukey's honest significant difference (HSD) test [22]. In case that the assumptions of the Tukey's HDS do not hold for a specific data set, a t-test in conjunction with an α adjustment calculated through the Holm-Bonferroni correction [3] is utilized.

F. Replicability of the Experiment

To allow easy replication and verification of our experiment, a complete replication package[2] is publicly available to interested researchers. Our replication package includes: an in-depth step by step definition of the UCS, the source code of the measurement scripts, the complete raw data of each trial

[2]http://cs.gssi.it/ESEM2017ReplicationPackage

run, the R scripts developed for analysing the data and the analysis output.

V. Experiment Execution

A. Instrumentation

The experiments were conducted by deploying the Qubus web application and the SQL database required by the application in a virtualization environment based on VMware Vsphere[3]. Both the web application and the SQL database were installed on machines running the *Windows Server 2012 Enterprise edition* operating system. All the runs of the experiment were executed on a HP Proliant DL380 Generation 7 (G7) server equipped with two Quad-core Xeon processors and a 18GB RAM memory with a 72GB HP SAS 2.0 Drive with RAID 0 configuration and a P400i Smart Array RAID controller. The dynamic measurements were carried out by utilizing a *Wattsup PRO meter*[4]. This tool enabled us to measure the overall power consumption of the machine on which the VMs adopted for the experiment were hosted on a per-second basis. The decision to adopt the *Wattsup PRO meter* was led by the high precision of the measurements carried out by this tool, the absence of induced power overhead and the ease with which this tool could be interfaced with the orchestration scripts.

B. Application deployment

In order to carry out the measurements the applications were installed on host servers by following the deployment strategies characterizing the single trials. Multiple instances of the application could be installed on the utilized servers, as the impact on the measurements of other instances installed on the same machine could be completely isolated by simply turning off the instances that were not of interest. This led to the set up of three distinct virtualized host servers, namely the application server and the database server of the distributed setting (D_d) and the single server necessary for the centralized deployment strategy (D_c).

The VMs were connected to each other through abstracted network devices; those devices internally bridged the traffic between the two virtual machines of D_d. This constituted an ideal environment in which the communication between the application server and the database server were expected to experience a latency close to zero.

In addition to the physical host server on which the VMs were deployed, a supplementary monitoring server was utilized in order to execute the scripts necessary to run the experiment. An overview of the relation between the physical and virtualized servers, as well as distribution of the experimental artifacts, is provided in Figure 3.

C. Load test execution

The interactions of users with the system, based on the UCS reported in Section IV-B, were simulated by implementing a

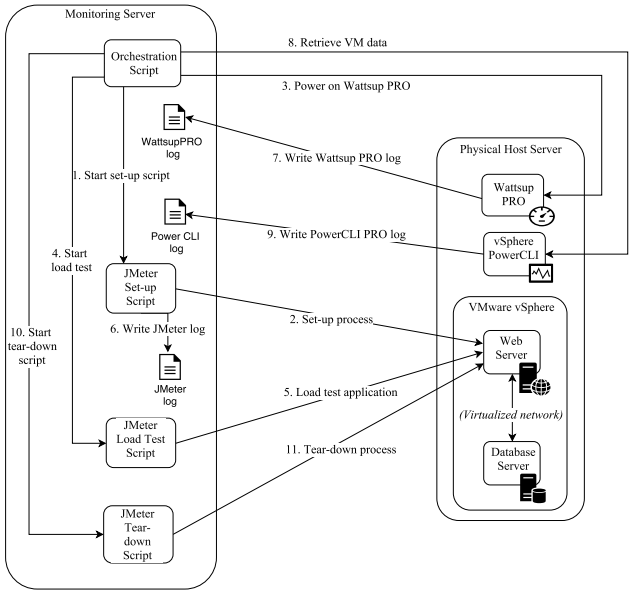

Fig. 3. Step-by-step overview of the experiment execution process (considering the distributed deployment strategy)

set of Apache JMeter[5] scripts; those scripts have been used for automatically load test the functional behavior of the application. The number of concurrent users simulated by the scripts was set to six, in accordance to the typical load of the platform (as the developers of the application reported through preliminary structured interviews). A *ramp-up* time corresponding to a delay of 100 milliseconds between the start of each user thread was adopted. This allowed to reduce the probability of concurrent threads locking shared resources simultaneously causing a potential deadlock.

To simulate a realistic input speed of the simulated users, a "think time" for the JMeter threads was randomly defined by utilizing a Poisson distribution with a mean value λ set to three seconds. The λ value was chosen by taking into account the average user input speed with the standard QWERTY keyboard method that, according to Arif et al. [2], is approximated to 64.80 words per minute.

In addition to the load tests further JMeter scripts were utilized in order to automatically satisfy preconditions of the UCS (*Step 1-2* of Figure 3) and carry out post-execution tear-down processes (*Step 10-11*). The recursive execution of the set-up, load test and tear-down processes was automatically coordinated by an ad-hoc Powershell orchestration script, as can be seen in Figure 3. This latter type of script additionally included the set up of the environment in which the experiments were run, e.g. powering the energy meters (*Step 3*), and supervised the retrieval of log files (*Step 8-9*). In order to completely isolate the impact of the simulated UCS on the experimental measurements waiting times were utilized between the set-up, load test, and tear-down processes. All the

[3]http://www.vmware.com/products/vsphere, last retrieved March 13, 2017
[4]https://www.wattsupmeters.com/secure, last retrieved March 13, 2017
[5]http://jmeter.apache.org/, last retrieved March 13, 2017

scripts utilised for the experiment execution were executed in a monitoring server, as depicted in Figure 3, in order to avoid a possible interaction of the script execution on the experimental measures.

For each of the trials reported in Table I and Table II sets of 30 subsequent runs were utilized to gather the statistical data. The execution of a set of runs ranged approximately from four to five hours.

D. Collected data

The output of each experimental run consisted of three artifacts, that were stored on the monitoring server (see Figure 3): • JMeter log: log file reporting an entry for each HTML request of the load test. Some attributes present in the file worth mentioning are *timestamp*, *Thread name*, *Total number of threads running*, *elapsed time*, *latency* and *response code*. These logs were automatically checked for consistency in order to ensure the correct execution of the load tests;
• Wattsup PRO meter log: a file containing the Watts, Voltage, and Current consumption (with relative timestamp) of the physical host machine where the VMware vSphere server virtualization environment was deployed. This data was utilized for the hypothesis testing process of our experiment;
• PowerCLI logs: a set of files containing information of the VMs on which the Qubus application was deployed. These data was collected through the VMware vSphere interface PowerCLI. Some of the stored information contained in such logs are: *Total CPU usage*, *Total Memory usage* and *Total disk usage*. This data was utilized in order to have further insights about the load test execution.

VI. Results

This section describes the results of the hypothesis testing by means of statistical analysis of the collected data. Additional findings deemed relevant for the study are also reported in this section. For a detailed report about the performed statistical analyses and the hypotheses testing we refer the reader to our replication package.

A. Impact of Software Releases on Energy Consumption (RQ1)

From a two tailed t-test carried out by controlling the familywise error rate through the Holm-Bonferroni correction the two means of the energy consumption result to significantly differ ($p\text{-}value$ = 0.0024) for U_1. The performed two-way ANOVA test showed that, while both treatments impact the measurand significantly ($p\text{-}value < 2 \cdot 10^{-16}$), the interaction of the two treatments also has a statistically significant effect on the dependent variable ($p\text{-}value < 2 \cdot 10^{-16}$). In order to better understand this interaction we calculated the effect size of the two treatments by adopting a partial eta-squared [6] on the previously processed ANOVA model. The results of the analysis are reported in Table III.

As we can notice from the values reported in Table III, ***both*** **the treatments and their interaction result to impact to a large extent the energy consumption of $\mathbf{U_1}$.** This trend can also intuitively be spotted in Figure 4, reporting the average energy consumption of the single trials.

TABLE III
ETA-SQUARED RESULTS ON ANOVA OF THE INTERACTION OF TREATMENTS (U_1)

Treatment and interactions	Partial eta-squared
Release	0.6979525
Deployment Strategy	0.9558263
Release : Deployment Strategy	0.7996772

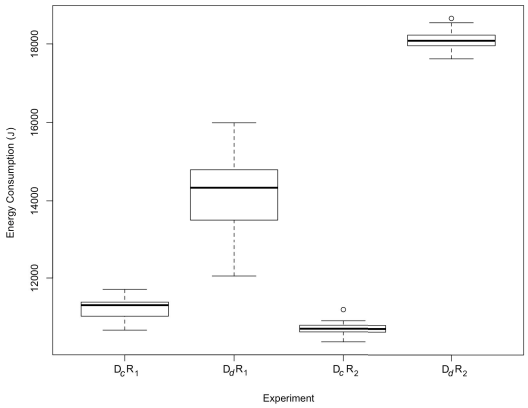

Fig. 4. Total energy consumption per trial (U_1)

Regarding U_2, an ANOVA analysis showed that software releases have a statistically relevant impact on the energy consumption ($p\text{-}value < 2.2 \cdot 10^{-16}$).

From a pairwise t-test with the Holm-Bonferroni correction the two means result to significantly differ ($p\text{-}value = 2 \cdot 10^{-16}$). As for U_1, a two way ANOVA evidenced a strong interaction between the two factors on the energy consumption that could not be neglected ($p\text{-}value < 2^{-16}$). By calculating the effect size through a partial eta-squared **the single treatments and their interaction result to have large effect size on the energy consumption of $\mathbf{U_2}$.** In Table IV the partial eta-squared values of the calculated ANOVA are reported.

TABLE IV
ETA-SQUARED RESULTS ON ANOVA OF THE INTERACTION OF TREATMENTS (U_2)

Treatment and interactions	Partial eta-squared
Release	0.9651552
Deployment Strategy	0.9370992
Release : Deployment Strategy	0.8932505

In light of the strong interaction between the treatments, the single trials have to be analyzed independently. In Figure 5 the distribution of the energy consumption grouped by trial of U_2 is depicted.

By considering the trials of both UCS, a further analysis revealed a **significant difference in energy consumption between the two Qubus releases**. In fact a Tukey's HSD test revealed a significant difference for U_1 ($p\text{-}value \leq 0.0035435$).

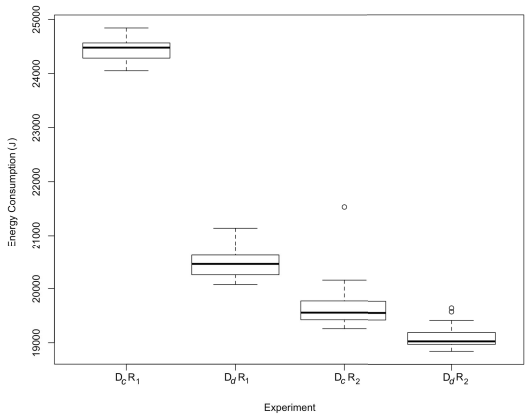

Fig. 5. Total energy consumption per trial (U_2)

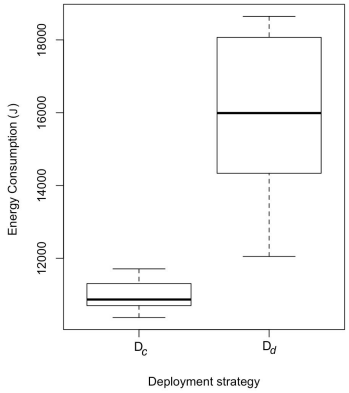

Fig. 6. Total energy consumption per deployment strategy (U_1)

Similar results were gathered by applying a pairwise t-test[6] adjusted with the Holm-Bonferroni method to the data of U_2 ($p\text{-}value \leq 2 \cdot 10^{-}11$). In light of the findings of these we can confidently **reject the *null* hypothesis** H_0^1.

Main findings - Impact of **software releases** on energy consumption (RQ1):

- ▶ both the treatments and their interaction result to largely impact the energy consumption of use case scenario U_1;
- ▶ both the treatments and their interaction result to largely impact the energy consumption of use case scenario U_2;
- ▶ significant difference in energy consumption between the two Qubus releases has been observed.

B. Impact of Deployment Strategies on Energy Consumption (RQ2)

In order to assess the impact of the deployment strategy on the energy consumption of U_1 an omnibus ANOVA test was carried out. From this statistical analysis the null hypothesis H_0^2 could be rejected ($p\text{-}value = 2.2 \cdot 10^{-16}$). In Figure 6 the distribution of the energy consumption for the two deployment strategies is depicted.

As reported in Section VI-B an **independent analysis of the treatments is insufficient to evaluate the impact of the deployment strategies** on the energy consumption. In fact, as can be intuitively seen in Figure 4, a strong interaction between the two treatments is influencing the measurand. For the sake of readability the previously carried out analysis on the interaction of the two treatments is not repeated in this section. Nevertheless the previously achieved results hold also for the testing of the second hypothesis, i.e. ***both* the deployment strategy and its interaction with the software release result to largely effect the energy consumption measurements of the load tests relative to $\mathbf{U_1}$.**

Regarding U_2, from an initial omnibus ANOVA test the deployment strategy treatment resulted to significantly impact on the energy consumption of the test cases ($p\text{-}value = 2.9 \cdot 10^{-10}$). From a pairwise t-test with the Holm-Bonferroni correction we observed that the mean of the two experimental groups significantly differ ($p\text{-}value = 3 \cdot 10^{-10}$) and hence we could conclude that the second treatment significantly affects the energy consumption. In Figure 7 the distribution of the energy consumption per deployment strategy of the Qubus application load tested through U_2 is reported.

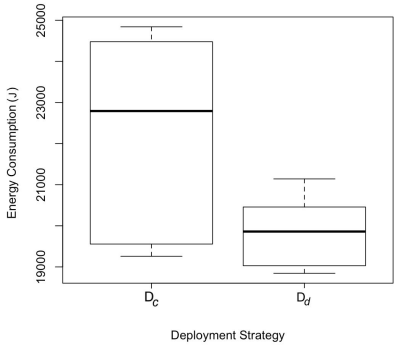

Fig. 7. Total energy consumptionper deployment strategy (U_2)

From the results of the first hypothesis testing, we know that **the interaction between treatments has to be considered before conclusions can be drawn for $\mathbf{U_2}$.** In fact, while the effect size of the DS treatment resulted to be very large (*partial eta-squared* ≈ 0.94) the same could be found for the other treatment (*partial eta-squared* ≈ 0.96) and the interaction of the two (*partial eta-squared* ≈ 0.89). This means that both the DS and its interaction with the other treatment significantly effect the energy consumption. Consequently we can conclude that the null hypothesis H_0^2 has to be rejected also for U_2, i.e. **there is a significant difference in energy consumption**

[6]The choice of adopting this latter type of analysis for U_2 was due to the heteroscedasticity of the variable calculated through a Breusch-Pagan test.

between the two deployment strategies with regards of U_2.

In conclusion, we can confidently state that **there is a significant difference in energy consumption between the two deployment strategies for both UCSs**. Hence we can confidently reject the *null* hypothesis H_0^2.

Main findings - Impact of **deployment strategies** on energy consumption (RQ2):

- ▶ both the deployment strategy and its interaction with the software release result to largely affect the energy consumption measurements of the load tests relative to use case scenario U_1;
- ▶ we observed a significant difference in energy consumption between the two deployment strategies with regard to use case scenario U_2;
- ▶ we observed a significant difference in energy consumption between the two deployment strategies for both use case scenarios.

VII. Lessons learned

In order to provide the industrial partner with as much information as possible regarding the data gathered from the experiment, in addition to the hypothesis testing, we carried out further statistical analyses. This section presents the key findings of the empirical experiment reported back to the KPMG *Qubus* team and a retrospective on the collaboration between academia and industry.

A. Relations between execution time, energy consumption, and power consumption

A first consideration has to be made on the relation between the power and energy consumption of the trials. We noticed that power consumption did not vary as drastically as the energy consumption of the load tests; also, it was not directly proportional to the total energy consumption. While from ANOVA omnibus tests the power consumption resulted to significantly influence the energy consumption ($p\text{-}value = 9.359 \cdot 10^{-9}$ for U_1 and $p\text{-}value = 1.921 \cdot 10^{-4}$ for U_2), it seemed as if another factor is influencing more these values. After a pairwise comparison, carried out by graphical means through matrices of scatter plots, the execution time was identified as the only dependent variable that grow linearly with respect to the energy consumption.

By calculating the effect sizes of the treatments on the power consumption and the execution times of the two UCSs, the execution times are affected to a large extent by the adopted treatments, while the power consumption exhibits only a small variation. We can therefore conclude that the performance variation outweighs the power consumption w.r.t. energy efficiency, i.e. **the variation of execution times results to have a deeper impact than the observed power variation on the energy consumption** due to the low variability of power consumption.

Hence, on one hand, developers of the *Qubus* application should not spend too much effort in lowering the overall power consumption of the underlying hardware through software changes, as their impact is with high probability negligible. On the other hand, improving the performance of the software product leads to an appreciable energy optimisation while improving also other quality attributes of the product. In addition, this heuristic enables to carry out refactoring processes aimed for energy efficiency improvements without the requirement of a sophisticated hardware and software infrastructure such as the one adopted for this empirical experiment.

B. Impact of treatments on energy consumption

Regarding the impact of the application release on the energy consumption, it is not possible to precisely isolate the effect of the release for U_1 without taking into account the deployment strategy. While the most recent release (R_2) resulted to consume slightly less energy in the centralized setting, it exhibited a higher energy consumption in the distributed one if compared to Qubus release (R_1). In contrast, for U_2, the effect of this treatment resulted to be more stable, with release R_2 consuming significantly less energy than R_1 in both deployment strategy. We can therefore conclude that a high dependency is present between software and hardware components when considering energy efficiency, and the **deployment strategies do not simply constitute an additive effect to the energy consumption w.r.t. the number of machines utilised.** In fact, two machines resulted jointly to consume less power than a single one for U_2. **Thus deployment strategies affect differently the energy efficiency according to which UCS is exercised and the selection on which to prefer w.r.t. energy efficiency has to be choosed according to which UCS is used more frequently**.

By considering the significant difference in energy consumption of the two UCS considered, we can conjecture that this divergence is caused by the different database operations entailed in the two UCS. In fact, by considering U_1, we can observe that the low presence in this UCS of database operations makes D_c more energy efficient. In contrast, the high frequency of READ operations required for U_2 make D_d the most energy efficient option, as it is more fitted to handle the high load of requests involved in U_2.

As an additional research activity we also investigated if changes in the source code of the two versions, measured through software metrics, could be correlated to energy consumption variation. Nevertheless, due to the low variability of the values of the software metrics adopted, no evident correlation was observable.[7]

Regarding the deployment strategy, we can notice that for U_1 this treatment had a remarkable impact on the energy consumption. A higher energy consumption was associated to the distributed deployment strategy. This is in accordance with the intuitive results that would be expected if two machines (virtual or physical) are used for the execution of a software

[7]We considered the following metrics: Mantainability Index, Cyclomatic Complexity, Depth of Inheritance, Class Coupling, Weighted Methods per Class, Number of Children, Lack of cohesion in methods and Effective Lines of Code. Due to NDA the measured values could not be made public.

application instead of a single one. Nevertheless the vast majority of the gathered measurements did not result to double in the distributed setting with respect to the centralized one, potentially due to the different load on the servers and the baseline values of the measurements that were not considered. Surprisingly, for U_2 the distributed deployment strategy resulted to consume less energy than the centralized one for both releases. From the inspection of the VMware vSphere logs we can conjecture that the increased energy consumption is caused by the higher load that the single server of D_c was exposed to.

In conclusion **there is no absolute best option of release and deployment strategy, as the interaction of these two factors must be considered.** Furthermore, UCS also influences the energy consumption, hence **it is crucial to identify the most recurrent use case scenarios** during the evaluation of which combination of release and deployment strategy to adopt. In our case the combination of D_c and R_2 results to be the best option, as it results to be the most energy efficient with respect to the most recurrent UCS (see Fig. 4) and exhibits a similar energy behaviour for the second UCS (see Fig. 6).

C. Retrospective on the academia-industry collaboration

The collaboration process between industry and academia was smooth and productive for both parties. We conjecture that the joint formulation of the problem to be solved played a key role in the successful collaboration.

A minor difficulty was experienced due to the availability of some of the key developer team members, as they were not always formally allocated to tasks aimed to support this research, and hence had sometimes to self-organise in order to find the time to assist us and provide feedback. Nevertheless, the interest of the industrial partner was sustained by formal meetings during which the important aspects of the research were discussed.

Another minor hindrance was constituted by the low interest of a small portion of developers in the topic of software energy efficiency. This is a problem known in academia [24], usually tackled by means of *people awareness strategies* [13].

VIII. Threats to Validity

In this section we discuss the threats to validity of our study based on the categorization given in [26].

Conclusion validity. For the vast majority of the results reported in Section VI the statistical tests produced sound $p\text{-}values$, far below the chosen significance level of 0.05. Furthermore, by carrying out additional post hoc tests, such as the Tukey's HSD, the high statistical power of the analysis was reconfirmed. In order to avoid violations of assumptions of the statistical tests, additional tests, such as the Breusch-Pagan Trialwere carried out and, in order to minimize the error rate of the results, the Holm-Bonferroni was adopted to adjust the significance level when required. A minor threat to conclusion validity is the *reliability of the implementation of the distributed deployment strategy*. As reported in Section V, the web and database servers were connected through abstracted network devices bridging the traffic internally between the two VMs. This constituted an ideal deployment environment in which the latency between the two machines is close to zero, and hence differs from the custom deployment of the application on physical machines, as in the industrial setting. Nevertheless, a higher latency between the two machines would likely even amplify the impact of the first treatment, leading to results in line with the ones reported in this experiment. Our choice of adopting a virtualised solution was guided by the need of having a controlled and isolated environment for the experiment. This decision is supported by the increasing adoption of virtualisation techniques in industrial settings [1].

Internal validity. After each experimental run the tested applications were reset to their initial states. Therefore the history of the experimental runs did not influence the results. Particular attention was paid to avoid energy measurements that could have been influenced by prior runs or preconditions' setup processes. As presented in Section VI, the interaction effects between the two treatments were analyzed in depth by carrying out ANOVA and post-hoc analyses on the gathered data. By using these approaches, the possible internal validity threat of unknown third factors that could potentially influence the measurements was reduced as much as possible. A final possible threat regards the *implementation of the load tests*. While the threads for simulating the user interaction with the system where created to emulate as closely as possible the real interaction with the system, the hard-coded values used for the ramp-up time of the threads are, in all real life setting, of random nature. Also, all the incoming requests of the Qubus server(s) were launched by the same client, i.e. the machine on which the JMeter scripts were executed, residing in the same subnet of the machines where the application was deployed.

Construct validity. While the experiment was not characterized by a strict *mono-operation* or *mono-method biases* [26], by analyzing a *higher number of releases* the results could have been more generalizable. In fact, only two software releases of a single software product were considered for our experiment. A higher number of samples would be therefore needed in order to accurately generalize the findings to a wider range of software applications. Furthermore, the *selection of the releases*, guided by the developers of the software product, has to be considered as an additional threat to construct validity. By considering the interactions between the different treatments, while their presence were remarkably present in the data set, the statistical analysis that followed did not violate the construct validity, as these interactions were carefully analyzed with post-hoc analyses, such as the partial eta-squared method. The selection of the UCS as *blocking factor* might be considered as a possible threat, as it leads to a reduction of the power of the ANOVA tests.

External validity. A possible threat to external validity resides in the *low number of samples utilized* as measurands in the experiment. This choice was primarily led by the time constraints under which this research was carried out. As presented in Section VI, a *strong interaction of selection and treatment* was present in the data set, making it difficult to generalize the

results to a wider range of applications. Furthermore, from the obtained results we observed that the functionality to be tested deeply influenced the energy consumption evaluation, leading to an additional interaction between selection and treatment. Additionally, a *minor interaction of setting and treatments* could be present in the experiment due to the low latency times caused by our virtualization environment. This could lead to a reduced representativeness of the industrial environment that was emulated for the experiments. Nevertheless, the impact of this threat has to be considered as very low, and hence it does not significantly influence the results. Finally, the *results are bound to the hardware and software instrumentation used to carry out the load tests*. This fact has to be taken into account when considering a wider range of applications and possible deployment configurations. Particular attention should be paid to investigate thoroughly the possible impact of the processor base frequency, that in our hardware configuration resulted to be slightly lower than the one specified in the minimum requirements of the application.

IX. Conclusions and Future Work

This work is motivated by the fast-growing energy consumption of IT. In collaboration with an industrial partner, KPMG, we investigated how deployment strategies can influence the energy consumption of a software product, and if software releases play a role in this evaluation. We measured and collected data for over 200 simulations of a platform for governance, risk, and compliance process management via automated load tests.In addition to the in-depth analysis reported to the industrial partner, the findings of this research deliver the following messages, and recommendations, to researchers and practitioners: (i) the power variation of underlying hardware might be negligible when optimising energy efficiency at software level; (ii) utilising a higher number of machines does not always imply a higher energy consumption of a software application; (iii) there is no absolute optimal option for releases and deployment strategies w.r.t. energy efficiency, as the interaction of these two factors has to be considered; (iv) the identification of the most recurrent UCS is crucial to efficiently carry out energy optimisations of a software product; (v) the collaboration between industry and academia resulted to be productive for both parties, even if a minor portion of practitioners manifested low interest/awareness on software energy efficiency.

As future work we are planning to extend this research by: (i) analysing a higher number of heterogeneous software products in order to collect enough statistical data to improve the generalizability of the experiment, (ii) considering a wider set of use case scenarios for better investigating its relation to the deployment strategies, and (iii) identifying and using open-source products with a publicly available source code repository (e.g., GitHub) in order to have better control on the various releases to be considered in the experiment.

References

[1] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *Journal of Network and Computer Applications*, 52:11–25, 2015.

[2] A. S. Arif and W. Stuerzlinger. Analysis of text entry performance metrics. In *Science and Technology for Humanity (TIC-STH), IEEE Toronto International Conference*, pages 100–105, Sept 2009.

[3] C. E. Bonferroni. *Teoria statistica delle classi e calcolo delle probabilita*. Libreria internazionale Seeber, 1936.

[4] T. S. Breusch and A. R. Pagan. A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the Econometric Society*, pages 1287–1294, 1979.

[5] G. Caldiera, V. Basili, D. Rombach, and J. Marciniak. Goal question metric paradigm. *Encyclopedia of Software Engineering*, 1:528–532, 1994.

[6] J. Cohen. Eta-squared and partial eta-squared in fixed factor anova designs. *Educational and psychological measurement*, 1973.

[7] E. R. Girden. *ANOVA: Repeated measures*. Number 84 in Quantitive Applications in the Social Sciences. Sage, 1992.

[8] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE software*, 23(6):88–95, 2006.

[9] A. Hindle. Green mining: A methodology of relating software change to power consumption. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, pages 78–87, 2012.

[10] E. A. Jagroep, J. M. van der Werf, S. Brinkkemper, G. Procaccianti, P. Lago, L. Blom, and R. van Vliet. Software energy profiling: Comparing releases of a software product. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 523–532. ACM, 2016.

[11] J. G. Koomey. Outperforming Moore's law. *IEEE Spectrum*, 47(3):68–68, 2010.

[12] KPMG. Qubus: Home, 2017. Online, available at: http://www.qubussoftware.com; accessed 1 August 2017.

[13] P. Lago and T. Jansen. Creating environmental awareness in service oriented software engineering. In *International conference on service-oriented computing*, pages 181–186. Springer, 2010.

[14] I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirovic. Assessing the impact of service workers on the energy efficiency of progressive web apps. In *Proceedings of the International Conference on Mobile Software Engineering and Systems, MOBILESoft '17, Buenos Aires, Argentina, May, 2017*, pages 35–45, 2017.

[15] I. Manotas, L. Pollock, and J. Clause. SEEDS: A software engineer's energy-optimization decision support framework. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, pages 503–514, 2014.

[16] J. McGrenere. "Bloat": The objective and subject dimensions. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, pages 337–338. ACM, 2000.

[17] S. Murugesan and G. R. Gangadharan. *Harnessing Green IT: Principles and Practices*. John Wiley & Sons, 2012.

[18] A. Pathak, A. Jindal, Y. C. Hu, and S. P. Midkiff. What is keeping my phone awake? characterizing and detecting no-sleep energy bugs in smartphone apps. In *10th International Conference on Mobile systems, applications, and services*, pages 267–280. ACM, 2012.

[19] C. Sahin, L. Pollock, and J. Clause. How do code refactorings affect energy usage? In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 36:1–36:10, 2014.

[20] F. Shull, J. Singer, and D. I. Sjøberg. *Guide to advanced empirical software engineering*, volume 93. Springer, 2008.

[21] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the conference on Power aware computing and systems*, volume 10, pages 1–5, 2008.

[22] J. W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114, 1949.

[23] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester. Trends in worldwide ict electricity consumption from 2007 to 2012. *Computer Communications*, 50:64–76, 2014.

[24] R. Verdecchia, F. Ricchiuti, A. Hankel, P. Lago, and G. Procaccianti. Green ict research and challenges. In *Advances and New Trends in Environmental Informatics*, pages 37–48. Springer, 2017.

[25] G. Welsh. *Yes, There IS a Difference Between Micros and 'Big' Computers*. TPUG News, 1987.

[26] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Computer Science. Springer, 2012.