Energy Efficient Assignment and Deployment of Tasks in Structurally Variable Infrastructures

Angel Cañete
Universidad de Málaga
Andalucía Tech, Málaga, Spain
angelcv@lcc.uma.es

ABSTRACT

The importance of cyber-physical systems is growing very fast, being part of the Internet of Things vision. These devices generate data that could collapse the network and can not be assumed by the cloud. New technologies like Mobile Cloud Computing and Mobile Edge Computing are taking importance as solution for this issue. The idea is offloading some tasks to devices situated closer to the user device, reducing network congestion and improving applications performance (e.g., in terms of latency and energy). However, the variability of the target devices' features and processing tasks' requirements is very diverse, being difficult to decide which device is more adequate to deploy and run such processing tasks. Once decided, task offloading used to be done manually. Then, it is necessary a method to automatize the task assignation and deployment process. In this thesis we propose to model the structural variability of the deployment infrastructure and applications using feature models, on the basis of a SPL engineering process. Combining SPL methodology with Edge Computing, the deployment of applications is addressed as the derivation of a product. The data of the valid configurations is used by a task assignment framework, which determines the optimal tasks offloading solution in different network devices, and the resources of them that should be assigned to each task/user. Our solution provides the most energy and latency efficient deployment solution, accomplishing the QoS requirements of the application in the process.

CCS CONCEPTS

• **Software and its engineering → Software product lines**; • **Computer systems organization → Distributed architectures**; *Embedded and cyber-physical systems*; • **Hardware → Power estimation and optimization**.

KEYWORDS

Software Product Line, Mobile Edge Computing, Mobile Cloud Computing, Energy Efficiency, Latency, Optimisation

ACM Reference Format:
Angel Cañete. 2019. Energy Efficient Assignment and Deployment of Tasks in Structurally Variable Infrastructures. In *23rd International Systems and Software Product Line Conference - Volume B (SPLC '19), September 9–13, 2019, Paris, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3307630.3342704

SPLC '19, September 9–13, 2019, Paris, France

ACM ISBN 978-1-4503-6668-7/19/09…$15.00
https://doi.org/10.1145/3307630.3342704

1 INTRODUCTION AND MOTIVATION

The importance of cyber-physical systems (CPS) [24] continues growing, due to the massive popularization of mobile devices for personal use (smartphones, personal devices, tablets, etc.), the improvement in the communication speed of wireless networks, the development of the Internet of Things (IoT) [1] and the benefits of Cloud Computing (CC) [29], which has played an important role in the development and advancement of mobile applications for the IoT. More recently, technologies such as Edge Computing [38] and Fog computing [5] bring the advantages and power of the cloud closer to where the data is created and consulted.

It is expected that by 2025 there will be around 80 billion of IoT devices -from personal mobile devices, to sensors, household appliances and wearables- connected to Internet, which would generate 175 trillion Gb of data per year [35]. In return, these forecasts imply an increase in traffic in the network that cannot be assumed by the cloud.

Edge Computing (EC) is a decentralized computing infrastructure in which data, computation, storage and applications are located somewhere between the source of the data and the cloud. This technology aims to improve the efficiency of the infrastructure, transferring the most hard-computational tasks of applications to any nearby device with network connectivity and computing power [27]. By this way, the data can be processed or stored nearby it is produced and/or consumed. Among the advantages of EC we can include the reduction of the communication latency respecting to send the data to the cloud (Mobile Cloud Computing, MCC) [41], the reduction in the energy consumption that can be achieved, and others benefits, like the data privacy obtained. EC takes advantage of the large amount of inactive computational power and distributed storage space in different Edge devices (such as Wi-Fi routers and access points, or low-cost computer boards - SBC) located nearby final users and devices (called edge of the network), and connected to the same WLAN. These devices, that forms the deployment infrastructure (DI), perform computationally intensive and/or critical latency tasks. This is known as Multi-Access Edge Computing [33] (formerly Mobile Edge Computing, MEC). These advances in TIC technologies influence the way in which applications are implemented and deployed in the IoT devices, the edge or the cloud [40, 46].

Features of the devices that conform the DI (called *nodes*) are very diverse: computing power, memory capacity, software characteristics, storage, communication latency, etc. Due to this variability,

it is not easy to know which node is the best option to deploy applications' tasks to obtain the benefits that MEC and MCC can provide, and accomplishing the QoS requirements of the applications in the process. It is necessary the creation of new methods and software tools that help developers to manage the variability of the hardware, the complexity of network access infrastructures, the distribution of data, and the decision of where the different computational tasks must be located. These methods and tools would help them in the applications' configuration and deployment, obtaining a good QoS level in the process (in terms of latency, security, etc).

A widely number of works propose code offloading as a solution to improve the applications' energy consumption and latency [14, 16, 19, 20, 23, 25, 28, 30, 34, 40, 43, 44, 49, 50]. Nevertheless, none of them give a solution to model and configure both applications and deployment infrastructures. In addition, only a few apply MCC and MEC at the same time [44, 49], while others do not take into account some factors of the process. For instance, some approaches do not considerate the execution time of tasks offloaded to the cloud [12, 28, 47, 48], while others ignore the communication time to obtain the response from the cloud [11], or just take into account the energy consumption of the CPU [4, 50]. Several MEC based approaches do not consider the workload of the edge devices [28, 37]. In this thesis, we will take into consideration all these issues, providing an engine to weight them according to the company/organization interests.

The main contributions of our proposal are the following:

- **We will create a model to configure DIs and applications for code offloading:** In order to obtain the optimal application deployment solution, it is primordial to considerate both the heterogeneity of applications and nodes of the DIs. Applications are composed by a set of configurable tasks. In the same application domain, every application can be configured to meet different requirements. At the same time, the energy consumption and latency of the applications' tasks will depend on hardware (e.g., node's architecture, CPU power, transmission power, etc) and software features (e.g., data to be transmitted, tasks' computational cost, etc) among other issues. This variability present in tasks and nodes can be modeled using SPL (Software Product Lines) [32] feature models, which allow the configuration of the DIs and the applications to meet specific deployment scenarios. In this thesis we will propose a general model for deployment infrastructures in edge and cloud, and will focus on the family of Augmented Reality (AR) [2] applications. We will decompose the functionality of this applications in intensive computational tasks that could be offloaded to the edge or cloud, which are characterized by the required resources and time requirements.
- **Our approach will define a process to generate the optimal deployment solution according to the application and the DI at runtime:** The configuration of the DI and the application variability models is the input of the Optimal Task Assignment Framework (OTAF), which resolves what is the optimal deployment solution in terms of energy consumption and latency. At the same time, the OTAF will assign hardware resources (storage, RAM, and CPU) to each user. The functional and non functional requirements of the application are taken into account during the process (e.g., hardware requisites, QoS needed, security requirements, etc). To achieve this, the problem must be formalized and solved. Data structures based on weighted graphs, and solver techniques like graph cutting or linear and nonlinear programming can help us in this task [27]. Since the OTAF is a daemon, and it is running in a node (manager node) continuously, each time a new user starts the application, it returns the optimal deployment solution and resources assignment according to the current state of the DI. Finally, to put into the practice this approach, we plan to make use containers [39], virtualization [26] or unikernel [6] solutions.
- **We will provide a solution to adapt user's applications at runtime:** The functionality of the application carried out by the device of the user, typically a smartphone, will depend on the deployment solution. Although the application's behaviour would change from one execution to another, due to the deployment assignment at the nodes of the DI, the application itself will be the same. It makes necessary the introduction of a mechanism to reconfigure applications according to this. For this reason, we will provide an adaptation engine for smartphones. As this can be done by different ways, we will implement a few solutions to adapt applications at loadtime and runtime. Then, we will compare them in order to find the most energy-efficient adaptation engine for smartphones.

The rest of the paper is organized as follow: Section 2 clearly states the research questions that we try to answer in this thesis. Section 3 shows the methodology and our approach. In Section 4 we present the preliminary results. Finally, in Section 5 we outline the structure of this thesis, the actual state of it and a year work plan.

2 RESEARCH QUESTIONS

As part of my thesis, our approach has to answer several research questions in order to fulfill the aim of this project.

Firstly, since we need information about the application's tasks as well as the DI's nodes they will be deployed on, the first question we would like to answer is: **RQ1: How can we model the variability of DI and the application's tasks in order to optimize the deployment assignment solution?** In this thesis we will bring a model to configure DIs that could be applied to any scenario. Nodes of the DI are composed by a set of features that describe them (CPU power, RAM, operative system, kind of device, etc). To model all these features, we will use SPL feature models. Specifically, we will use features with numerical values [7]. Our deployment infrastructure will be composed by a set of nodes. For this reason, we use cardinality-based feature models [17]. As testbed application, we will focus on AR applications' family. Many mobile devices are often unable to cope with the runtime real time requirements of these multimedia apps. Offloading to the cloud is not always a feasible option due to the high and inconstant latency of wide-area networks [42]. In any case, we need to determine which information must be contained in the models in order to determine

the energy consumption of the deployment and the maximum latency permitted. The information resulting by the configuration of the application and the DI feature models is used as input of our assignment problem. So, **RQ2: How can be the problem formalized and solved?** The assignment of tasks and resources to the nodes of the DI is not an easy process. Functional requirements (e.g., the application need a camera to run) as well as non functional requirements (e.g., QoS, security, etc) must be taken into account. In order to accomplish the QoS, dependencies between tasks must be considered. These dependencies can be of two types: sequential, where one task ends before the other task begins, or parallel, where it is not necessary that the previous task ends before starts the next one. Some tasks (or a set of them) could have time limitations in order to achieve the QoS requirements. We need a model that allows us the management of tasks dependencies, along with time restrictions. Graph representation allows both kind of dependencies [22], and linear and/or nonlinear programming are positioned as good ways to solve our problem.

Tasks' assignment process is not only difficult, but it is also a hard-computational process. Therefore, we wonder if it it possible to do it at runtime, and **RQ3: How can we afford a deployment assignment at runtime?** The deployment solution obtained from RQ2 will be provided by a microservice architecture, *structuring the application as a collection of services in order to distribute the tasks that compose the application among the nodes of the DI.* This service must be able to bring a solution at loadtime in a time short enough to be invisible to the final user. Due to the problem's complexity, it could be difficult to achieve this goal. For this reason, collecting information about previous executions allow us to predict the optimal solution for the current problem. As smartphones may not be very resource-rich devices, our service must be running on a DI's node, called manager node.

RQ4: How can be the tasks deployed on DI's nodes? Once an optimal deployment solution is found, we need to automatically deploy each task to the assigned node. These nodes should distribute and reserve part of their resources to each task, so it is necessary a mechanism to make it possible. During the execution, it might happen that some nodes stop working. In addition, the location of the user may change along the execution, being some nodes inaccessible from the new location. Nevertheless, the service cannot be interrupted. For this reason, the solution must include data replication (fault tolerance) and data and functionality migration at runtime from one node to another without stopping the application, making as result a self-adaptive system able to change the tasks' allocation if necessary. Some existing technologies, such as virtualization, containers (e.g., Docker) or unikernel solutions, facilitate redeployment. We will explore all of them in terms of energy consumption, strengthness and weakness , and their influence in QoS. When all offloaded tasks from the user's node to the DI are running, the application of the user must be adapted in order to execute only the tasks that have been assigned to it. We focus in scenarios where user' nodes are smartphones. It poses the next question: **RQ5: How can applications being adapted to the edge-based deployment assignment?** We need a solution to adapt existing applications to the deployment assignment. As the majority users have Android smartphones, we will focus our approach in Android applications, although we do not discard to present a solution to adapt iOS applications too. Our adaptation approach must be as lightweight as possible, consuming a little amount of energy. To achieve this, we will propose different adaptation engines (e.g., based on proxy pattern [21], virtual-method hooking [15]) and then we will measure the energy overhead introduced by them, as well as their scalability. Energy consumption in Android smartphones can be measured using different existing software tools, such as Trepn Profiler[1] or GreenScaler [13].

Once the application is deployed and it is running, we have to determine how good is our deployment assignment. **RQ6: What is the benefit obtained by our optimal assignment process?** Minimize the applications' energy consumption and latency are the goals of this thesis, satisfying the functional and non functional requirement on the way. On one side, we need a validation module that verifies that these requirements are accomplished (functional testing, execution time tests, security testing, etc). On the other side, the benefit obtained by our solution should be evaluated. It can be done by measuring the execution time and energy consumption of the code offloading solutions and then comparing it with the result of execute all the application's tasks at the node of the user. Measuring the energy consumption is not an easy task, as it is highly dependent on the architecture of the device (hardware and software features), its state (e.g., workload), and environmental conditions. Some of them, like the background processes running on the devices, could be difficult to control.

3 RESEARCH METHODOLOGY AND APPROACH

For the accomplishment of the objectives presented in this thesis, we will use a SPL based approach to model the variability of applications and DIs (RQ1). The optimal assignment deployment problem must be formalized, in order to use the information obtained by the models to bring the optimal deployment solution (RQ2).

Our research in this thesis is both applied and empirical. We plan to develop a service-oriented solution to, given a configured deployment infrastructures and application, to generate the optimal deployment assignment (RQ2, RQ3). The tasks are offloaded from the user's node (RQ5) to be deployed to different nodes of the DI (RQ4). In order to obtain the optimal deployment assignment, we plan to use both linear solvers (like Z3) [18] and nonlinear solvers (like Gekko) [3]. The framework will run as a microservice in a manager node and will be attending requests from users. The manager node contains information about the state of each node of the DI (hardware and software information, network and workload). The manager node is responsible for sending the instructions to the rest of nodes in order to deploy application's tasks, as well as assigning resources to each task. We will define the mechanisms to maintain the manager node updated with the status of the rest of nodes, and a protocol to control and perform tasks deployment and data migration. Nodes could use virtualization, containers or unikernel based solutions. At the end, the application works like a microservice-based architecture, where the application's tasks are distributed into different nodes. All the process must be approved by a validation and verification module, which verifies that both

[1] https://developer.qualcomm.com/software/trepn-power-profiler

functional and non functional requirements are accomplished, as well as evaluates how good is the solution (RQ6).

Figure 1 shows a full view of our approach. One part of our proposal is based on domain engineering methodology (top of Figure 1, Sections 3.1, 3.2 and 3.3) and the other part is based on application engineering methodology (bottom of Figure 1, Sections 3.4, 3.5 and 3.6).

3.1 Model of the Deployment Infrastructures

The set of nodes that compose the DI are modeled using cardinality feature models with numerical values. On top of Figure 1, the SPL model of a DI is shown. In this model, nodes have a type associated: they can be computational nodes, user nodes or sensor motes. Computational nodes are devices in which tasks of the user can be offloaded in order to save energy or time. For example, computers, smartphones, IoT Gateways or limited resources computers, like Raspberrys. User nodes are devices in which the main application is executed. Once a user node is connected to the service, it can form part of the DI like computational node or not, according to the policy. By last, sensor motes are nodes in a sensor network capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. The type of information collected by sensor motes depends on the sensor units associated to them. In the model shown on top of Figure 1, there is a constraint that makes mandatory for sensor motes and smartphones to have at least one sensor unit associated. Some application's tasks require one or more sensor units to be executed, so this allows us to determine which nodes are able to run each task (e.g., the task *CaptureVideoFrame* must be ran in a node with an image sensor unit associated). The nodes are composed by a set of numerical features that describe them (CPU power, HD capacity, RAM). Finally, the nodes are connected to internet using a network with a numerical feature associated, its bandwidth, that describes the data transmission rate of the node.

3.2 Model of the application's family

An application's family is composed by a set of software applications that share the major of their features. This allows the configuration of a collection of applications and the reuse of software components. The feature model of an application must contain the information necessary to describe it. The configuration of the feature model of an application determines its functionality, which is used to select the set of tasks that form the application. These tasks have software and hardware requirements that are modeled in the feature model and their granularity depends on the dependency between their functions.

3.3 Optimal Tasks Assignment Framework

The information obtained by the configuration of the DI and the application is used by the OTAF to assign tasks to DI's nodes. The assignment must accomplish both, functional and non functional requirements of the application. It is necessary to define a structure of data that allows to manage the information of tasks (e.g., dependencies, time restrictions, etc) and a solution to convert the data from the feature models to the OTAF's format. Graph representation is posed as a good technique as it allows to manage tasks features and connections between tasks. The OTAF needs information about tasks that are not provided by the feature model of the application, like tasks dependencies or the sets of tasks with time restrictions. This kind of information is pre-planned in the OTAF for the possible input values of the applicatons' configuration.

We will formalize the assignment process like an optimization problem with restrictions, with the objectives of minimizing the energy consumption and latency. Time and energy consumption due to tasks' computation and communication must be defined in order to predict the execution latency and energy consumption of the deployment assignment. We will define polices to center the energy profit on different parts of the DI (e.g., user node, battery powered nodes of the DI, DI's overall powered consumption, etc). Like solvers, we will start using Z3 (for linear optimization problems) and Gekko (for nonlinear optimization problems). Nevertheless, we do not discard the use of other kind of solutions based on genetic algorithms, for example.

The Optimal Tasks Assignment Framework must return the optimal solution in a short time, making possible the assignment at runtime (RQ6). The potential application areas where IoT and edge computing solutions are advantageous, help to show the contributions and benefits of our approach (e.g., eHealth applications [31], smart campus, smart buildings, etc). For instance, in an eHealth application that provide its service in an hospital, the number of users and nodes makes the problem size considerable. One of the main motivation for the decentralization promoted by EC is to support IoT infrastructure scalability rather than mobile applications' interactive performance. The scalability of our proposal must be evaluated in order to predict its applicability for different sizes of the problem and scenarios. The number of assignments for tasks and nodes is $nodes^{tasks}$, so for a large number of nodes and tasks the OTAF may need too time to provide the solution at loadtime. Nevertheless, it can be done by collecting information about previous executions and processing it in order to predict the most appropriate solution for new users (e.g., by using machine-learning techniques). As we focus on mobile users (battery powered devices), the battery level of the user's device is a crucial information to select the policy of the deployment assignment (e.g., maximum power saving on user node). By last, the service of the OTAF must be running on a DI's node (manager node). It can be implement like a REST API, running on a Nginx server[2].

3.4 Tasks' deployment

The solution obtained by the OTAF must be deployed on the DI. In order to do that, we explore solutions based on unikernel, virtualization, and containers. At first look, VM virtualization is more heavyweight than the others [45], as its emulates the hardware of the computer. Containers, instead of virtualizing the underlying computer like a virtual machine (VM), just the OS is virtualized. By last, unikernel solutions pop the OS from the execution stack, compiling all the necessary (including operating system support functions) to run the application in a single executable. Unikernel solutions make the system faster, smaller, and safety; as the OS is deleted from the execution stack, its vulnerabilities are removed

[2] http://nginx.org/en/

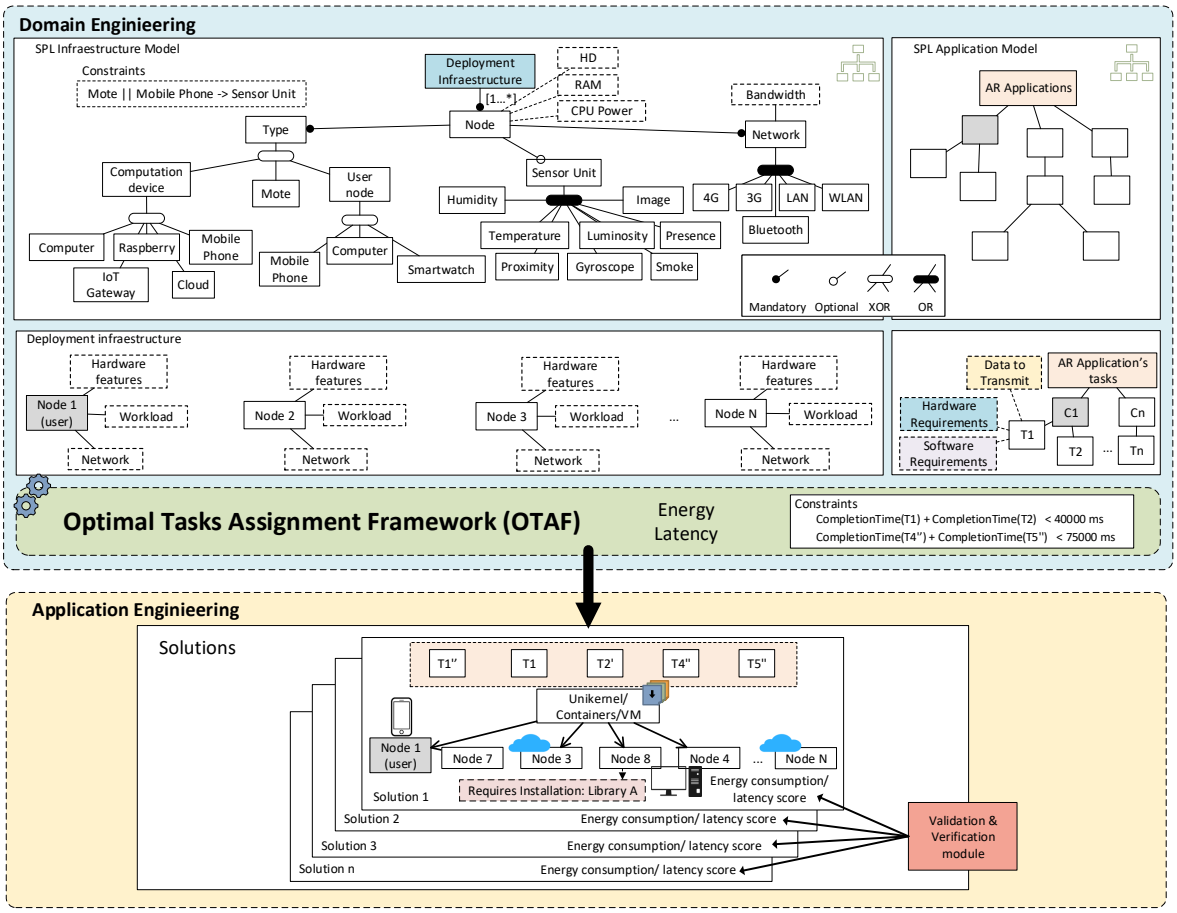

Figure 1: Our approach

too. For a part of the application to start, all the parts of the application that it depends on must be ready. These dependencies between parts of the application, whatever the deployment solution (unikernel, virtualization or containers based), will be evaluated based on ready conditions.

During the execution of the application, it is possible that some DI nodes stop working. Our system must be able to detect faults in the DI's nodes and fix them without stopping the service provided to the user. In order to do that, the manager node maintains information about the state of the DI's nodes and make a backup of the tasks' executions periodically. This information may be sensible and can contain private data; taking into account the decentralized nature of Edge Computing, it will be encrypted in order to secure it. If the manager node detects that a node has stopped, the tasks that were running on this node are assigned to other operational nodes. These nodes are notified in order to deploy the assigned tasks and receives the information of the users executions, obtained by the manager node by restoring it from its backup. Changes in the DI's nodes affect to the prediction system, which has to take into account only the available nodes. In the same way, the service provided by the manger node must be replicated in order to provide fault tolerance in this node.

3.5 Adaptation engines for the applications

The behavior of the application running on the user's device will depend on the deployment assignment at the DI's nodes. In this thesis, we focus on AR applications, where the node of the user is typically a smartphone. Adaptation engines based on dynamic proxies and method hooking (e.g., using Xposed framework [36]) will be explored in order to adapt the applications' functionality to the deployment assignment. Several adaptation engines will be implemented and compared between them. Our aim is to obtain the most energy-efficient adaptation engine at runtime/loadtime. We will focus on Android OS because it is the most extended mobile platform, although we do not discard its research for other mobile OS.

3.6 Verification and validation module

The first step to verify the deployment assignments is to check if software and hardware requirements are accomplished. It can be done by a feed-backing between the configuration of the DI and the application. Then, functional testing (using unit test cases) is necessary in order to prove that the application works as expected. In terms of safety, we will focus on data protection and application access. Tools like BeEF[3] could help us for this task.

[3]BeEF: https://beefproject.com/

For the purpose of knowing how good is our solution, energy consumption and latency must be measured. It can be done by using software or hardware based tools. Software tools, like pTopW [10] (Windows), PowerTop[4] (Linux) or Trepn Profiler and GreenScaler (Android) allow the estimation of the energy consumption via software. Alternatively, hardware based tools (like Kill A Watt[5]) provide more accuracy and are OS independent. Nevertheless, it has no sense its usage in battery powered devices.

3.7 Threats to Validity

As we move forward in our research: (1) the number of tasks and nodes can make the problem too hard-computational to be solved at runtime; (2) although we can estimate the energy consumption of the application's tasks in each node of the DI, it is not possible to know the exact energy consumption of them in the user's node (hardware and software variability, background processes, etc); (3) the amount of energy saved will depend on the DI's nodes, as well as the number of tasks that the user will be able to offload to them; (4) in unikernel based solutions, CPU resources are assigned to users in terms of CPU cores available, hindering the prediction of workload of the nodes in each moment to accomplish the QoS requirements.

4 PRELIMINARY RESULTS

So far, I have presented a full paper [9] at "Jornadas de Ingeniería del Software y Bases de Datos" (JISBD 2018). In this paper, four different adaptation engines for Android applications' code are discussed, in order to find the most energy-efficient solution: (1) based on dynamic proxies, where the adaptation alternatives (functions that replaces the default behavior of the application) are internal to the application; (2) based on dynamic proxies, where the adaptation alternatives are external to the application (e.g., placed in device external storage, placed on a server, etc); (3) method-hooking and proxies based; and (4) based on method-hooking. As scenarios in which these engines would be used are very diverse (e.g., number of adaptation alternatives, number of rules that determines if adaptations should be triggered, etc), the scalability of the adaptation engines is evaluated. It is demonstrated that the number of adaptable functionalities (classes), the number of adaptation rules and the number of adaptation alternatives do not increase the energy consumption of the application. The impact in the energy consumption by the engines themselves are evaluated, concluding that the energy consumption of the isolated adaptation engines is tiny compared with the application one, that is of the 2.68% in the worst case. It makes our solution a good candidate to be applied to the schema presented in this thesis, in order to adapt the application running in the user's device to the application deployment solution at the DI's nodes. Some solutions presented in [9], like (1) and (2) could be used in others OS like iOS, as they are based on a design pattern.

Currently, I am working on a project that forms the basis of this thesis, presenting a first version of the DI feature model, as well as a feature model for AR applications' family. In this work, we formalize the Optimal Assignment Problem like a Linear Optimization Problem. Then, the problem is solved using the Z3 solver. Finally, the latency performance of the solver for different sizes of the problem is evaluated. A first version of this work [8] will be presented at JISBD 2019. In this paper, the first version of the OTAF is presented and evaluated. The Optimal Assignment Problem is posed like mono-objective, where the aim is to optimize the energy consumption taking into account the functional and non functional requirements of the applications. The service of the OTAF is provided in a node with a AMD Ryzen 7 1700X processor, using one core for compute the assignment problem. In this case, the system is able to bring a solution in 0.76 seconds for a application composed by 11 tasks and a DI formed of 5 nodes, obtaining a benefit of the 58% in the energy consumption compared with the execution of the application in the user node.

5 WORK PLAN

This thesis has two main edges; a component based on SPL and a part based on application engineering. First, we define the SPL component (Task 1, 2) and then we apply application engineering to use the information provided by the SPL component to deploy the application's tasks among the DI's nodes (Tasks 3,4,5, and 6). The results of Task 6 give us a feedback information which will be used to improve the model (Task 1, 2). Figure 2 shows a twelve-month work plan. Concretely, the tasks are the following:

Task 1: Refine and improve the feature models. Until now, we have developed a first version of the variability models of AR applications' family and DI. Nevertheless, we need to continue improving them in order to achieve complete and accurate variability models. This task will continue till the middle of July.

Task 2: Assign CPU resources in the nodes of the DI to users. Initially in parallel with Task 1, and further till end-August, we will work in the assignment of the CPU resources of the DI's nodes to users. For this task, we need to formalize and implement the Optimal Assignment Optimization Problem as a nonlinear programming problem. The first choice is to use the Gekko solver, although we do not discard to use another nonlinear solver or even the usage of other kind of solutions for optimization, such as, for instance, solutions based on genetic algorithms. As result, we will obtain a first version of the OTAF with CPU resources assignment.

Task 3: Implementation of the AR applications' tasks as microservices. Tasks of AR applications that can be offloaded to DI's nodes for their execution will be provided as microservices. So, the first step is to define which tasks of AR applications' family can be offloaded to nearby nodes. This step may involve having to change the variability model of the application from Task 1. Once tasks are clearly defined for offloading to the DI, their functionality must be implemented. Finally, these functionalities will be provided by a microservice architecture. In order to do that, we will explore different techniques (e.g., RESTful API, SOAP). We will work in this task from start September to the middle of November.

Task 4: Provide the OTAF like a microservice. From the beginning of November to the middle of December, we will work to supply the functionality of the OTAF (Task 2) like a microservice in

[4]PowerTop: https://01.org/powertop/
[5]Kill A Watt: http://www.p3international.com/products/p4400.html

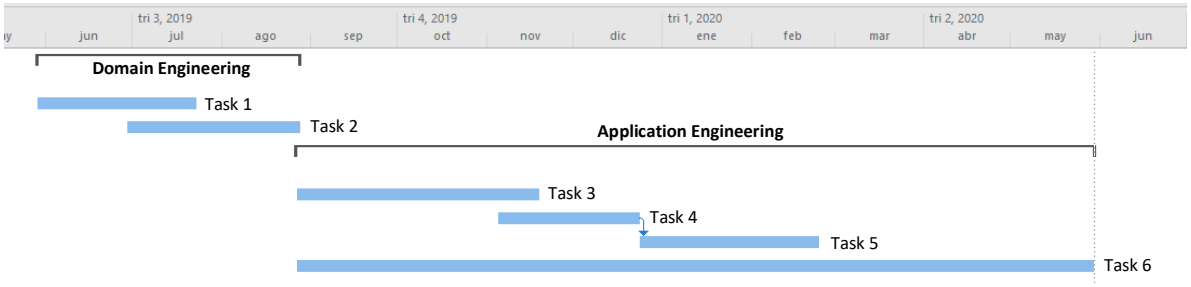

Figure 2: One year work plan

one node of the DI (manager node). This node contains information about the state of all nodes of the DI and provides the service to assign one node to run each task of the users' applications. The creation of the OTAF microservice is a crucial part of this thesis. This microservice can use all the resources of the manager node or only part of them, running on a VM or a container (e.g., Docker).

Task 5: Implementation of the deployment process for AR applications. From the middle of December to the middle of February 2020, we will put into practice the optimal assignment solution obtained by the OTAF (Tasks 2 and 4). Application's tasks have to be allocated at the nodes of the DI according the task assignment, using the minimum amount of resources in the process. For this purpose, the use of virtualization, containers and unikernel solutions will be explored.

Task 6: Tasks' deployment evaluation. This task, which evaluates the energy consumption and execution time, will start at the beginning of Task 3 and will continues after the end of Task 5. At its beginning, during Task 3, Task 6 evaluates the deployment of the each task of the AR applications and other case studies considered. During Task 4, the energy consumption and execution time of the OTAF's microservice will be evaluated, and during Task 5, the evaluation will involve the procedure of assignment of tasks as a whole. From the analysis result, some corrections can be made in the feature model to improve the correctness of the model (Task 1). Quality of deployments will be measured, evaluated and compared in terms of energy consumption and latency. This task will continue to the end of May.

5.1 Publication plan

The form in which the problem is structured poses a set of challenges that should be accomplished separately. All the solutions for these challenges allow us to obtain intermediate solutions susceptible to be published in international congresses and workshops. Due to the features of the thesis, the edges of the conferences and workshops can be diverse, like based on Software Product Lines, microservices or IoT services, according to the topic of the specific approach. The most extended works will be sent to journals indexed in JCR.

Some journals, conferences and workshops appropriated to distribute our solutions are:

- **Journals:**
  - Information and Software Technology
  - Empirical Software Engineering
  - Journal of Systems and Software
  - Software and Systems Modeling
  - Journal of Universal Computer Science
  - Information Systems Research
  - ...
- **Conferences and Workshops:**
  - International Conference on Mobile and Ubiquitous Systems: Networks and Services (MobiQuitous)
  - Workshop on Mobile Computing Systems and Applications (HotMobile)
  - Distributed Applications and Interoperable Systems (ICWS)
  - Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)
  - Systems and Software Product Line Conference (SPLC)
  - European Conference on Software Architecture (ECSA)
  - International Conference on Software Reuse (ICSR)
  - IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)
  - ...

ACKNOWLEDGEMENTS

This work is supported by the projects Magic P12-TIC-1814, HADAS TIN2015-64841-R (co-funded by FEDER funds), TASOVA TIN2017-90644-REDT, and MEDEA RTI2018-099213-B-I00 (co-funded by FEDER funds).

REFERENCES

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15 (2010), 2787 – 2805. https://doi.org/10.1016/j.comnet.2010.05.010

[2] Ronald T Azuma. 1997. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments* 6, 4 (1997), 355–385.

[3] Logan D. R. Beal, Daniel C. Hill, R. Abraham Martin, and John D. Hedengren. 2018. GEKKO Optimization Suite. *Processes* 6, 8 (2018). https://doi.org/10.3390/pr6080106

[4] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems* 28, 5 (2012), 755 – 768. https://doi.org/10.1016/j.future.2011.04.017 Special Section: Energy efficiency in large-scale distributed systems.

[5] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12)*. ACM, New York, NY, USA, 13–16. https://doi.org/10.1145/2342509.2342513

[6] A. Bratterud, A. Walla, H. Haugerud, P. E. Engelstad, and K. Begnum. 2015. IncludeOS: A Minimal, Resource Efficient Unikernel for Cloud Services. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. 250–257. https://doi.org/10.1109/CloudCom.2015.89

[7] Rafael Capilla and Juan C. Dueñas. 2002. Modelling Variability with Features in Distributed Architectures. In *Software Product-Family Engineering*, Frank van der Linden (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 319–329.

[8] Angel Cañete, Mercedes Amor, and Lidia Fuentes. 2019. Sistema de Asignación de Tareas Energéticamente Eficiente en Infraestructuras de Despliegue Variables. (2019).

[9] Angel Cañete, Jose-Miguel Horcas, and Lidia Fuentes. 2018. Mecanismos de Reconfiguración Eco-eficiente de Código en Aplicaciones Móviles Android. (2018).

[10] Hui Chen, Youhuizi Li, and Weisong Shi. 2019. pTopW : A Power Profiling Tool for Windows Platforms. (05 2019).

[11] Huangke Chen, Guipeng Liu, Shu Yin, Xiaocheng Liu, and Dishan Qiu. 2018. ERECT: Energy-efficient reactive scheduling for real-time tasks in heterogeneous virtualized clouds. *Journal of Computational Science* 28 (2018), 416 – 425. https://doi.org/10.1016/j.jocs.2017.03.017

[12] Z. Cheng, P. Li, J. Wang, and S. Guo. 2015. Just-in-Time Code Offloading for Wearable Computing. *IEEE Transactions on Emerging Topics in Computing* 3, 1 (March 2015), 74–83. https://doi.org/10.1109/TETC.2014.2387688

[13] Shaiful Chowdhury, Stephanie Borle, Stephen Romansky, and Abram Hindle. 2018. GreenScaler: training software energy models with automatic test generation. *Empirical Software Engineering* (20 Jul 2018). https://doi.org/10.1007/s10664-018-9640-7

[14] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. 2011. CloneCloud: Elastic execution between mobile device and cloud. *EuroSys'11 - Proceedings of the EuroSys 2011 Conference*, 301–314. https://doi.org/10.1145/1966445.1966473

[15] Valerio Costamagna and Cong Zheng. 2016. ARTDroid: A Virtual-Method Hooking Framework on Android ART Runtime.. In *IMPS@ ESSoS*. 20–28.

[16] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. 2010. MAUI: Making Smartphones Last Longer with Code Offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*. ACM, New York, NY, USA, 49–62. https://doi.org/10.1145/1814433.1814441

[17] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. 2005. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice* 10, 1 (2005), 7–29. https://doi.org/10.1002/spip.213 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spip.213

[18] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08/ETAPS'08)*. Springer-Verlag, Berlin, Heidelberg, 337–340. http://dl.acm.org/citation.cfm?id=1792734.1792766

[19] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek. 2017. Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling. *IEEE Transactions on Communications* 65, 8 (Aug 2017), 3571–3584. https://doi.org/10.1109/TCOMM.2017.2699660

[20] Ioana Giurgiu, Oriana Riva, Dejan Juric, Ivan Krivulev, and Gustavo Alonso. 2009. Calling the Cloud: Enabling Mobile Phones as Interfaces to Cloud Applications. In *Middleware 2009*, Jean M. Bacon and Brian F. Cooper (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 83–102.

[21] Youssef Hassoun, Roger Johnson, and Steve Counsell. 2004. Applications of dynamic proxies in distributed environments. *Software: Practice and Experience* 35, 1 (2004), 75–99. https://doi.org/10.1002/spe.629

[22] M. Jia, J. Cao, and L. Yang. 2014. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 352–357. https://doi.org/10.1109/INFCOMW.2014.6849257

[23] Mads Darø Kristensen and Niels Olof Bouvin. 2010. Scheduling and development support in the Scavenger cyber foraging system. *Pervasive and Mobile Computing* 6, 6 (2010), 677 – 692. https://doi.org/10.1016/j.pmcj.2010.07.004 Special Issue PerCom 2010.

[24] E. A. Lee. 2008. Cyber Physical Systems: Design Challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. 363–369. https://doi.org/10.1109/ISORC.2008.25

[25] Y. Lin, E. T. . Chu, Y. Lai, and T. Huang. 2015. Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds. *IEEE Systems Journal* 9, 2 (June 2015), 393–405. https://doi.org/10.1109/JSYST.2013.2289556

[26] Flavio Lombardi and Roberto Di Pietro. 2011. Secure virtualization for cloud computing. *Journal of Network and Computer Applications* 34, 4 (2011), 1113 – 1122. https://doi.org/10.1016/j.jnca.2010.06.008 Advanced Topics in Cloud Computing.

[27] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. 2017. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys Tutorials* 19, 4 (2017), 2322–2358. https://doi.org/10.1109/COMST.2017.2745201

[28] Y. Mao, J. Zhang, and K. B. Letaief. 2016. Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications* 34, 12 (Dec 2016), 3590–3605. https://doi.org/10.1109/JSAC.2016.2611964

[29] Peter Mell, Tim Grance, et al. 2011. The NIST definition of cloud computing. (2011).

[30] Antti P. Miettinen and Jukka K. Nurminen. 2010. Energy Efficiency of Mobile Clients in Cloud Computing. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud'10)*. USENIX Association, Berkeley, CA, USA, 4–4. http://dl.acm.org/citation.cfm?id=1863103.1863107

[31] Claudia Pagliari, David Sloan, Peter Gregor, Frank Sullivan, Don Detmer, James P Kahan, Wija Oortwijn, and Steve MacGillivray. 2005. What Is eHealth (4): A Scoping Exercise to Map the Field. *J Med Internet Res* 7, 1 (31 Mar 2005), e9. https://doi.org/10.2196/jmir.7.1.e9

[32] Klaus Pohl, Günter Böckle, and Frank J van Der Linden. 2005. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media.

[33] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb. 2018. Survey on Multi-Access Edge Computing for Internet of Things Realization. *IEEE Communications Surveys Tutorials* 20, 4 (2018), 2961–2991. https://doi.org/10.1109/COMST.2018.2849509

[34] Jan M Rabaey, Anantha P Chandrakasan, and Borivoje Nikolic. 2002. *Digital integrated circuits*. Vol. 2. Prentice hall Englewood Cliffs.

[35] Rydning Reinsel, Gantz. 2018. The Digitalization of the World: From Edge to Core. (2018).

[36] rovo89. [n.d.]. Xposed Framework. http://repo.xposed.info/. Online; accessed 13 May 2019.

[37] S. Sardellitti, G. Scutari, and S. Barbarossa. 2015. Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing. *IEEE Transactions on Signal and Information Processing over Networks* 1, 2 (June 2015), 89–103. https://doi.org/10.1109/TSIPN.2015.2448520

[38] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3, 5 (Oct 2016), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[39] Madiha H. Syed and Eduardo B. Fernandez. 2015. The Software Container Pattern. In *Proceedings of the 22Nd Conference on Pattern Languages of Programs (PLoP '15)*. The Hillside Group, USA, Article 15, 7 pages. http://dl.acm.org/citation.cfm?id=3124497.3124515

[40] T. X. Tran and D. Pompili. 2019. Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks. *IEEE Transactions on Vehicular Technology* 68, 1 (Jan 2019), 856–868. https://doi.org/10.1109/TVT.2018.2881191

[41] A. u. R. Khan, M. Othman, S. A. Madani, and S. U. Khan. 2014. A Survey of Mobile Cloud Computing Application Models. *IEEE Communications Surveys Tutorials* 16, 1 (2014), 393–413. https://doi.org/10.1109/SURV.2013.062613.00160

[42] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. 2014. Adaptive deployment and configuration for mobile augmented reality in the cloudlet. *J. Network and Computer Applications* 41 (2014), 206–216. https://doi.org/10.1016/j.jnca.2013.12.002

[43] Tim Verbelen, Tim Stevens, Pieter Simoens, Filip De Turck, and Bart Dhoedt. 2011. Dynamic deployment and quality adaptation for mobile augmented reality applications. *Journal of Systems and Software* 84, 11 (2011), 1871 – 1882. https://doi.org/10.1016/j.jss.2011.06.063 Mobile Applications: Status and Trends.

[44] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung. 2015. Dynamic service migration in mobile edge-clouds. In *2015 IFIP Networking Conference (IFIP Networking)*. 1–9. https://doi.org/10.1109/IFIPNetworking.2015.7145316

[45] Song Wu, Chao Mei, Hai Jin, and Duoqiang Wang. 2018. Android Unikernel: Gearing mobile code offloading towards edge computing. *Future Generation Computer Systems* 86 (2018), 694 – 703. https://doi.org/10.1016/j.future.2018.04.069

[46] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang. 2016. Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks. *IEEE Access* 4 (2016), 5896–5907. https://doi.org/10.1109/ACCESS.2016.2597169

[47] W. Zhang, Y. Wen, and D. O. Wu. 2013. Energy-efficient scheduling policy for collaborative execution in mobile cloud computing. In *2013 Proceedings IEEE INFOCOM*. 190–194. https://doi.org/10.1109/INFCOM.2013.6566761

[48] W. Zhang, Y. Wen, and D. O. Wu. 2015. Collaborative Task Execution in Mobile Cloud Computing Under a Stochastic Wireless Channel. *IEEE Transactions on Wireless Communications* 14, 1 (Jan 2015), 81–93. https://doi.org/10.1109/TWC.2014.2331051

[49] Tianchu Zhao, Sheng Zhou, Xueying Guo, Yun Zhao, and Zhisheng Niu. 2015. A Cooperative Scheduling Scheme of Local Cloud and Internet Cloud for Delay-Aware Mobile Cloud Computing. (11 2015). https://doi.org/10.1109/GLOCOMW.2015.7414063

[50] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu. 2014. Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds. *IEEE Transactions on Cloud Computing* 2, 2 (April 2014), 168–180. https://doi.org/10.1109/TCC.2014.2310452