DRACeo : A smart simulator to deploy energy saving methods in a services/microservices based network

Hernán H Álvarez Valera, Marc Dalmau, Philippe Roose, Jorge Larracoechea, Christina Herzog

Compas'2020 : Parallélisme / Architecture/ Système/ Temps Réel
MILC - Lyon, France, du 30 juin au 3 juillet 2020

DRACeo : A smart simulator to deploy energy saving methods in a services/microservices based network

Hernan Humberto Alvarez Valera, Marc Dalmau, Philippe Roose, Christina Herzog, Jorge Andres Larracoechea Gonzalez

Université de Pau et des Pays de l'Adour, E2S UPPA
64600 Anglet / FRANCE
hhavalera@univ-pau.fr, Marc.Dalmau@iutbayonne.univ-pau.fr,
Philippe.Roose@iutbayonne.univ-pau.fr,
jorge-andres.larracoechea@etud.univ-pau.fr,herzog@efficit.com

---

Abstract
Nowadays, there are many architectures for microservices deployment through the network (e.g. Docker-Kubernetes, Kalimucho, etc.) which aim at the creation of modular applications in the most efficient way. On the other hand, energy saving is an issue that concerns both sustainability and business global economy. For this reason, many researchers work to identify deployment solutions in order to save energy without decreasing functional QoS. In this work, we present DRACeo : a simulator to deploy and schedule microservices and their dependencies on various devices with software and hardware heterogeneity. DRACeo is able to apply various scheduling heuristics algorithms, including various philosophic management : centralized vs non-centralized in order to save energy without losing applications QoS. The simulator acts on moving, duplicating, starting/stopping microservices and devices. In addition, it performs important operations such as resource status monitoring, calculation of energy saved and current consumption (at device and network level). We have experimented some ideas based on our previous work (called Kaligreen) to demonstrate the effectiveness of DRACeo.

Mots-clés : microservices, middleware, energy, hardware, prototype, simulator

---

1. Introduction

Currently, many companies and scientists base their applications on the use of microservices because they allow architectural advantages such as acceleration of deployment cycles, modularity, improvement of maintainability, scalability, between others [18] [8] [19] [17]. For example, Docker and Kubernetes [6] manage microservices on the cloud or even on grid environments, while Kalimucho [10] does the same on user device level.
Kalimucho middleware [5] deploys ,starts, stops, moves and duplicates microservices in devices, regardless of whether a device belongs to cloud entities or user terminals. As Kalimucho is able to measure devices hardware capacities, it is a good middleware candidate for energy savings purposes. For this reason, in some of our previous works [12] [11] we proposed a middleware called "Kaligreen" strongly inspired by Kalimucho. In Kaligreen, we denote the necessity to find a balance between the hardware and operating system agnosticism and the energy and

QoS repercussions that should be evaluated in the applications. Kaligreen proposes a supervisory entity for each device, which constantly monitors the energy consumption, and the load of its hardware components. This supervisor is able to differentiate between different types of executing microservices and their restrictions. In addition, the supervisor can execute its own scheduling algorithm based on smart negotiations. With Kaligreen, we have shown that a decentralized planning algorithm can be very effective in saving energy purposes [12]. However, we did not study enough the heterogeneity and capabilities of devices and its components,nor did we consistently consider the QoS of applications when microservices are scheduled. For these reasons, we propose a tool that allows evaluating different heuristics of microservice deployment and scheduling, considering the capabilities and peculiarities of each hardware component, as well as the evaluation of QoS when a planning algorithm is executed. Thereby, DRACeo is not dedicated to a particular middleware. DRACeo is able to virtualize an environment controlled by any middleware that deploys and schedules microservices, allowing researchers to create any type of scheduling algorithm and measure its consequences in terms of energy and application efficiency.
To better understand this approach, we will now explain the related works in section II, then, the policies and architecture of the simulator will be explained in Section III and finally in section 4, we will show some examples and results.

2. Related work

In order to understand and formally express microservices based application efficiency and its energy consumption, a multi-scenario modeling tool that allows implementing various deployment and scheduling algorithms is necessary. Packet oriented simulators can evaluate network traffic as well as the result of using certain transmission protocols. For instance Packet Storm [16], IP WAN Emulator [9] and Cisco Packet Tracer [3], allow modeling scenarios to evaluate data streaming techniques and features, latency, cases of data loss and data duplication between others. In addition, O-ICN Simulator [1] allows us to evaluate the status of a network and to model a special architecture called Information Centric Networking (ICN), while GreenCloud [7] allows us to understand energetic issues at packet network-protocol level.
These simulators are very useful for modeling protocol-oriented scenarios and understanding connection phenomena. However, we believe these tools cannot replicate the behavior (i.e. use of resources, application QoS and energy consumption) of a service-based architecture, where a higher level of abstraction is needed.
For this reason, [Petr Novotny et al. 2016] [15], propose a framework for the simulation of architectures based on services in MANET networks. It allows efficiently and with an excellent granularity to identify the possible scenarios when the services exchange information ; however, it does not consider the energy repercussions of services deployment nor does it allow to know the best way to schedule them in a given network. Cloudsim [4], on the other hand, allows modeling service-based architectures with different virtualization politics and in different CLOUD/GRID layers. It allows the study of diverse energy and QoS repercussions when managing services ; while DockerSim [14], considers not only strategies at service-based architecture, but also full packet-level network and protocol behaviors.
However, we believe these tools do not offer a transparent way of deploying high-level distributed/centralized planning algorithms among heterogeneous user devices, in which, for example, the relation between battery analysis and application QoS is important.
Table 1 shows the characteristics of some simulators offered by important scientists, developers and enterprises where :(A)Considers behavior aspects at connection level (protocols, data loss,

TABLE 1 – Analysis of existing Simulators

	A	B	C	D	E	F	G	H
PacketStorm,IPWAN, O-ICN and PacketTracer	X							
GreenCloud	X	X						
[Petr Novotny et al. 2016]			X			X	X	
CloudSim			X		X	X		
DockerSim	X		X			X	X	
DRACeo			**X**	**X**	**X**	**X**	**X**	**X**

etc.)(B)Considers energy consumption at connection level (protocols, data loss, etc.)(C)Supports microservice deployment in cloud and local network environments.(D)Considers special type of microservice management.(E)Considers energetic issues when operating microservices.(F) Considers communication phenomena between services.(G)Considers Microservices dependencies when operating them. (H)Support of centralized and non-centralized scheduling algorithms in a network, allowing to modelize fog or edge architectures and QoS criteria.
As seen in this table, DRACeo is strongly focused on the application of centralized and distributed scheduling algorithms, allowing to model in real time : 1) the formal behavior of applications QoS 2) the load of the hardware components of each node and 3) the energy consumption of a particular device as well as the entire modeled network. Furthermore, DRACeo supports virtualization technologies such as containers and various architectures such as fog and edge as well as the management of a granular approach based on microservices.

3. DRACeo Simulator

Considering we study the deployment of distributed applications in a network of heterogeneous devices to save energy without losing QoS, we developed DRACeo. DRACeo is a simulator able to manage any microservice based application in any network architecture. Thus, in our simulator context, we define an application as a directed graph whose nodes are microservices and edges the connections between them. Then, an application is deployed in a graph of connected devices, where nodes are devices with heterogeneous hardware characteristics and the edges are any form of network connections between them. Then, DRACeo is able to move or duplicate a microservice from one device to another, analyzing the impact in the physical network and in the transfer rate between microservices.
On the other hand, DRACeo evaluates deployed applications, by calculating in real time their performance of energy consumption and QoS (i.e. transfer/calcul bottlenecks,desired hardware components performance, efficient microservices dependencies management, etc). Finally, our simulator enables the analysis of centralized and non-centralized (re)deployment algorithms, by executing them and plotting the energy consumption of the entire device network, as well as of each particular device. To do that, it manages various entities and operations, which, in general terms, are : 1) an operation center that manages all devices and microservices. 2) devices, which run microservices. 3) the microservices supervisor, responsible for monitoring the status of every device. 4) device connections, which know the linked microservices connections. 5) the applications, which are graphs composed of microservices and 6) the scheduling algorithm (under test), which can be distributed in the devices or centralized in the operations center. The particular characteristics of each entity are explained in the next section.

3.1. Simulator entities

(1)Device : It is capable of executing a microservice and by default, has a supervisory entity installed [12] . DRACeo can deploy several types of devices and evaluate in real time the status

TABLE 2 – Criteria for analyzing the components of a device

CPU	RAM	NET Card	Hard Disk	Battery
Current/Available load	Current/Available load	Current/Available load	Current/Available load	Used.
Presence or absence of PCPG/DVFS/Overlocking	Energy expenditure by time interval/real time	Energy expenditure by time interval/real time	Energy expenditure by time interval/real time	Remaining
Energy expenditure by time interval/real time				

of each of its components in terms of the criteria shown in table 2 :
It is worth mentioning that even though it is known that the criteria in table 2 are handled directly by the operating system, we have already argued in previous work [12] [11] that evaluating these at a middleware level can benefit for energy savings when distributed scheduling algorithms are executed. Thereby, a global evaluation (centralized) or distributed through the supervisory entity may be carried out in terms of : 1) microservices in execution 2) energy status and 3) current load of components, in order to do the best deployment analysis. (2)Microservice : In the context of our simulator, a microservice is an entity with a defined function. DRACeo supports by default 3 types of microservices with performance and contextual differences : graphical user interface microservices (may not be moved, since the user experience would be affected), calculation microservices and database management microservices (may be duplicated or moved to another nearby device). (3) Connection : DRACeo defines and manages two kinds of connections : Physical connections that exist between devices and logical/functional connections between microservices. A physical connection logically stores many microservices connections that use it and calculates the current transfer rate respecting the maximum possible transfer rate. Thereby, this allows understanding : 1) the general state (i.e. at energy and load level) of a given device network taking into account the existence of abstract entities such as clusters, cloudlets, etc. 2) the set of microservice connections that use a certain physical connection. 3) the optimal (at energy or QoS level) device path to achieve the connection between two microservices. On the other hand, connections between microservices logically store the set of physical connections they use to communicate. That is, it is possible to know(through an implementation in the supervisory microservice) the path a microservice connection uses to work, as well as the current and expected transfer rate. (4)Application : As mentioned at the beginning of the section, an application is a directed graph of microservices. Each microservice can have connections as dependencies with several other microservices, which run on different devices on the network. The energetic performance and QoS of an application, is defined by the efficiency of execution and communication of each microservice that compose it. (5)Operations Center : DRACeo also has a centralized entity, in which all references to connections and existing devices of a deployed scenario are stored. In this way, any type of middleware can be configured and any centralized scheduling algorithm can also be applied. (6)Abstract Entities : DRACeo also supports entities such as 1) cloudlets as a dynamic resource provider and 2) clusters, as a set of devices which are stable and static resources providers. Both can be deployed and connected in the same way as the devices, establishing the amount of resources to offer and manage. With these, our simulator is compatible with both small networks of user devices and networks based on cloud technologies.

3.2. Simulator Operations

DRACeo is able to apply different types of heuristics for microservices (re)deployment and scheduling through a network. For this, the following operations have been implemented :
(1)Microservice migration : DRACeo can move microservices from one device to another. This

operation can be done from 1) the central operations entity, which knows the entire list of microservices and connections; and is capable of executing centralized planning algorithms, or from 2) a particular device, which knows its connected devices and is capable of executing non-centralized scheduling algorithms. This operation allows to find the best execution host for a microservice, both in terms of energy and QoS. (2)Microservice duplication : DRACeo knows each microservice function. Then, it is also capable of duplicating microservices when, for example, the scheduling algorithm decides this function is very requested and its overload produces excessive energy consumption.(3)Microservices and devices Start/Stop : DRACeo allows starting or stopping devices or microservices on a scheduled or manual basis. This allows generating scenarios in which high availability may be evaluated or, special energy saving techniques such as put into sleep mode the used devices [2] may be tested (4)Centralized/non-centralized scheduling algorithm Start/Stop (5)QoS analysis : Our simulator, before deploying a scenario, is able to find an ideal efficiency of an application in an ideal scenario, in which there are no processes that generate competition and network connections are always entirely available. Then, it can evaluate the efficiency of application at configurable intervals. Operations such as graphical PLOT, data storage and linear/polynomial regression are supported. (6)Energy consumption analysis : Measurement of energy consumption in time intervals is available. This analysis can be done on each device or on the entire network. Operations such as graphical PLOT, data storage and linear/polynomial regression are also supported. (7)Save/Load scenario from the initial state or the already modified state.

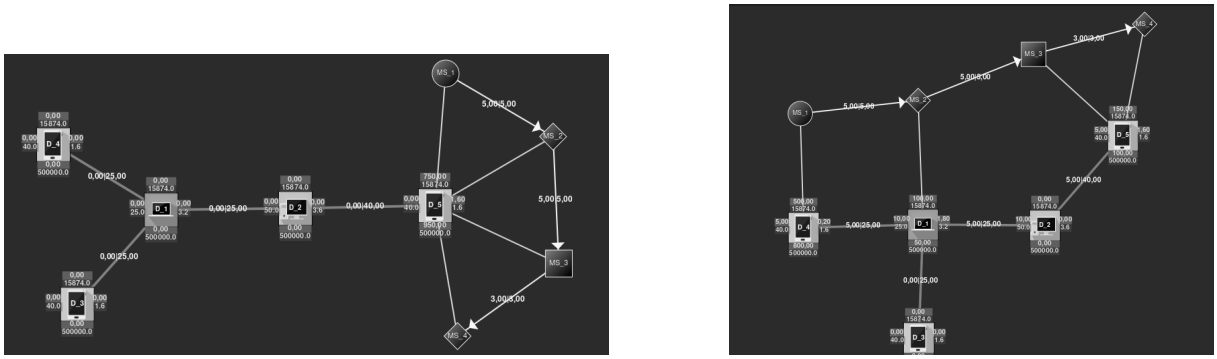
4. Results

To test DRACeo effectiveness, we have implemented a simple and naive algorithm for energy savings. Our goal is to show that we can deploy any type of energy saving scheduling approach and its analysis regarding QoS. In this algorithm, each supervisor microservice analyzes the energy consumption of each device and compares it with a threshold that we set. If threshold is exceeded, then, the device will attempt to move the microservice that consumes more resources to the most free of load connected device, as demonstrated by algorithm 1.

Algorithm 1 Algorithm running in all device (threads) D_i

```
1: while TRUE do
2:     if isExeededThreshold() == true then
3:         msApp_h = selectHeaviestMicroservice()
4:         freeDev = selectFreeDevice()
5:         moveMicroservice(msApp_h, freeDev)
6:         wait(givenTime)
7:     end if
8: end while
```

Let's consider the mixed network of figure 1a. A smartphone (D_5) is running an application composed of 4 microservices that send information linearly (i.e. from MS_1 to MS_2, from MS_2 to MS_3 and from MS_3 to MS_4). This smartphone is connected to a computer(D_2). Then, a laptop D_1 is connected to D_2 too and is linked with two other smartphones D_3 and D_4.
In algorithm 1, each device tries to move the heaviest microservice to the least loaded neighbor. It is important to say that since the algorithm 1 has no end and is quite naive, it makes the microservices MS_1 and MS_2 oscillate between the device D_1 and D_2; however, some interesting results are obtained. DRACeo is able to store the best solution found by an algorithm in some of its iterations. In this particular case, we find that the best deployment is the one represented in the figure 1b, where the microservices that saturated the CPU of the D_5 smartphone are running now on the laptop D_1 and smartphone D_4 too.



(a) Beginning (b) Result

FIGURE 1 – Topology example

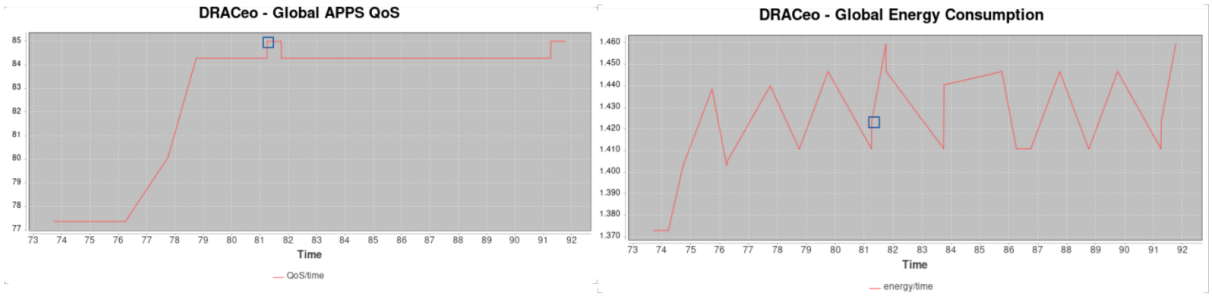


FIGURE 2 – Results of QoS and Energy

In the solution analysis, represented in figure 2, the power consumption was very low since only the resources of device D_5 are used. Then, it increased considerably due to the saturation of network cards in D_1, D_2, D_4 and D_5 .The D_1 and D_4 CPU and RAM have also increased their load as well. However, the prototype interprets it as the best solution since it obtains highest QoS (85 % at second 81.2 of execution) while lowest energy use (1422 joules in the second 81.2 of execution). We can conclude for example that, if we apply this kind of algorithms, we will obtain a direct relation between QoS and energy consumption.

5. Conclusions

In this work, we have presented DRACeo, a simulator capable of deploying and scheduling distributed applications in any type of network architecture. It implements functions for moving or duplicating microservices, enabling the execution of centralized and non-centralized planning algorithms. The objective of DRACeo is to allow the test of dynamic deployment and scheduling algorithms that re-deploy microservices in order to reduce energy consumption while keeping a certain QoS. DRACeo is generic as it allows to replace deployment/scheduling algorithms. Thereby, it will help developers and researchers find the best application deployment and best distribution behavior in any network. To our knowledge, DRACeo is the only simulator focused on the deployment of these two types of algorithms, managing in real time multiple types of devices and analyzing QoS and energy consumption.

Bibliographie

1. Agrawal (S.), Shailendra (S.), Panigrahi (B.), Rath (H. K.) et Simha (A.). – O-icn simulator (oicnsim) : An ns-3 based simulator for overlay information centric networking (o-icn). – In *Proceedings of the 1st Workshop on Complex Networked Systems for Smart Infrastructure*, New York, NY, USA, 2018. Association for Computing Machinery.
2. Azmy (N. M.), El-Maddah (I. A.) et Mohamed (H. K.). – Adaptive power panel of cloud computing controlling cloud power consumption. – In *Proceedings of the 2Nd Africa and Middle East Conference on Software Engineering*, New York, NY, USA, 2016. ACM.
3. CISCO. – Cisco packet tracer. – `https://www.netacad.com/courses/packet-tracer`, 2020.
4. cloudbus. – Containercloudsim : An environment for modeling and simulation of containers in cloud data centers. – `http://www.cloudbus.org/cloudsim/container.html`, 2016.
5. Da (K.), Dalmau (M.) et Roose (P.). – Kalimucho : Middleware for mobile applications. – In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, New York, NY, USA, 2014. Association for Computing Machinery.
6. DOCKER. – Debug your app, not your environment. – `https://www.docker.com/`, 2020.
7. Greencloud. – Greencloud - the green cloud simulator. – `https://greencloud.gforge.uni.lu/`, 2017.
8. IBM. – Take a cloud-native approach to building mobile and web applications with a microservices architecture. – `https://www.ibm.com/cloud/architecture/architectures/microservices`, 2020.
9. Inc. (G. C.). – Ip wan emulator. – `https://www.gl.com/wan-link-emulation-ipnetsim.html`, 2020.
10. Louberry (C.), Roose (P.) et Dalmau (M.). – Kalimucho : Adaptation platform for mobile applications. – In *2011 11th Annual International Conference on New Technologies of Distributed Systems*, pp. 1–8, May 2011.
11. lvarez Valera (H. H.), Dalmau (M.), Roose (P.) et Herzog (C.). – The architecture of kaligreen v2 : A middleware aware of hardware opportunities to save energy. – In *2019 Sixth International Conference on Internet of Things : Systems, Management and Security (IOTSMS)*, pp. 79–86, Oct 2019.
12. lvarez Valera (H. H.), Roose (P.), Dalmau (M.), Herzog (C.) et Respicio (K.). – Kaligreen : A distributed scheduler for energy saving. *Procedia Computer Science*, vol. 141, 2018.
13. MICROSOFT. – Creating composite ui based on microservices. – `https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/microservice-based-composite-ui-shape-layout`, 2018.
14. Nikdel (Z.), Gao (B.) et Neville (S. W.). – Dockersim : Full-stack simulation of container-based software-as-a-service (saas) cloud deployments and environments. – In *2017 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pp. 1–6, Aug 2017.
15. Novotny (P.) et Wolf (A.). – Simulating services-based systems hosted in networks with dynamic topology. 02 2016.
16. packetstorm. – Network simulation. – `https://packetstorm.com/network-simulation/`, 2018.
17. REDHAT. – Understanding microservices. – `https://www.redhat.com/en/topics/`

microservices, 2020.
18. Services (A. W.). – Implementing microservices on aws. vol. 2019, 2019.
19. Shadija (D.), Rezai (M.) et Hill (R.). – Microservices : Granularity vs. performance. – In *Companion Proceedings of The10th International Conference on Utility and Cloud Computing*. Association for Computing Machinery, 2017.