The architecture of Kaligreen V2: A Middleware aware of hardware opportunities to save energy

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

The architecture of Kaligreen V2: A Middleware aware of hardware opportunities to save energy

Hernán H. Álvarez Valera*, Marc Dalmau† and Philippe Roose‡
LIUPPA, Université de Pau et des Pays de l'Adour
Anglet-France
Email: *hhavalera@univ-pau.fr, †Marc.Dalmau@iutbayonne.univ-pau.fr,
‡Philippe.Roose@iutbayonne.univ-pau.fr

Christina Herzog
EFFICIT SAS
Mauzac, France
herzog@efficit.com

***Abstract*—Nowadays, energy saving in the use of information technologies is a very important issue both from the economic and sustainability point of view. Many scientists investigate methods to save energy at different application levels (cloud: i.e. architectures, grid: i.e. middlewares and frameworks and hardware management: i.e. operating systems) and many of them agree on the strategy of executing programs, processes or virtual machines only using the time and resources that are strictly necessary. For this, it is necessary to plan strategies for deployment and relocation of processes; but always taking into account hardware repercussions and the knowledge of the architecture and applications behavior. On the other hand, it has already been demonstrated that the use of microservices brings numerous advantages in availability and efficiency; but we do not find many jobs that exploit this technique on the energy level. In this article, we present the architecture of a middleware for distributed microservices-based applications, which allows any negotiation-based scheduling algorithm to duplicate or move microservices from one device to another in a non-centralized way for energy savings, taking into account the consumption characteristics of the microservices and the capabilities that the hardware components offer.**

***Index Terms*—microservices, middleware, energy, consumption, CPU, network, hard disk**

I. Introduction

Currently, energy saving has become a very important factor. Green computing scientists, normally base the incentive of their jobs in (1) the cost generated by spending energy and (2) the serious problem of sustainability that our planet has. We consider that both represent very serious problems. For example, the Office of Scientific and Technical Information of the USA points out that the data center sector was estimated to have consumed about 61 billion kilowatt/hours (kWh) in 2006 (1.5 percent of total U.S. electricity consumption) for a total electricity cost of about $4.5 billion (2006 dollars). The electricity use of the nation's servers and data centers in 2006 was more than double the electricity that was estimated to have been consumed for this purpose in 2000 [1]. On the other hand, attending to sustainability and using the same idea, carbon emissions have grown and will continue to grow exponentially in the following years, seriously deteriorating our planet.

These reasons have inspired many computer scientists to develop methods that allow cloud and grid designers to be aware of the energy they spend in each of their operations. They argue that cloud administrators should have policies and tools to optimize their services in execution operations [2] [3] [4], by analyzing variables like, the natural environment in which the data centers are located and correctly studying customers' real needs.

From other points of view, energy can also be saved by moving the modules of distributed applications or virtual machines through the nodes of a network, in such a way that some of the hardware's features are exploited [5] [6] [7] [8]. In this last approach, computer scientist found the relationship between the load of node devices and the running processes. With this, a software module could be moved or stoped with the aim of allowing (1) a CPU to turn off its cores or decrease its frequency [8] [9] [10], (2) Decrease disk access [11] [12] and (3) eventually turn off or suspend the entire node [6].

Finally other types of work focus on energy savings through the energetic analysis of programming languages in some operating system [13] and their repercussions at instruction level [14]. Thus, we classify energy saving techniques on three different levels in two main moments:

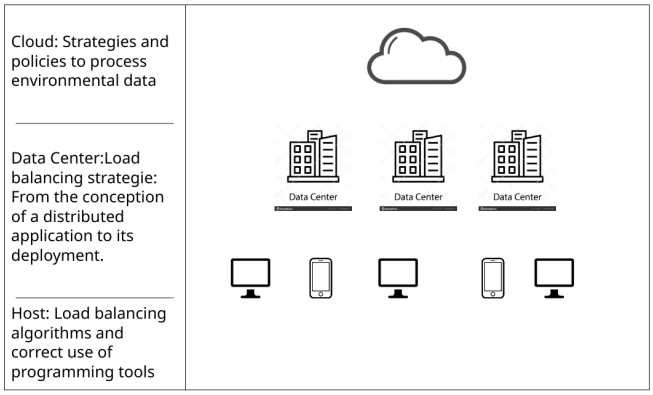

Fig. 1. Strategies to save energy in 3 different levels

In Figure 1, we show three levels of execution of techniques for energy saving: Cloud, data center and node. On each of these three levels, the design moment and the deployment of applications are taken into account to apply control or execution techniques.

As it's possible to see, most work concentrate their efforts on saving energy from the perspective of an administrator. There are no inspiring statistics focusing in users contexts (For example, the relationship between user's applications and the deterioration of smartphones batteries) and there are not many works that talk about the possibility of saving energy by studying the conception and deployment of applications from the user's point of view.

We already proposed a middleware capable of managing user applications based on microservices [15]. This middleware is based on Kalimucho, which uses the "Osagaia" container to start, stop, move or duplicate microservices through user devices [16].

In order to design and develop it, first, we differentiated the microservices that can be moved (in order to release processing load) or duplicated (allowing to distribute the load of an operation in different devices to free transfer rate or use of cpu) without altering the user's experience. Then, we demonstrated that if an application is conceived from the point of view of microservices, it is possible to apply some of the load balancing techniques mentioned above for energy saving.

Kaligreen [15] allows the communication and negotiation among users' common devices (laptops, smartphones, etc.), installing in each of these, a special microservice in charge of monitoring the CPU frequency, amount of used ram, use of the network and of the internal storage . If these values represent an inefficient load condition (i.e. overloading of any of this components) affecting the efficient use of energy, or mainly causing battery problem, the device's monitoring microservice will send a light vector with the load that it wishes to transfer (that is, what the heaviest microservice consumes) in order to find some generous devices. On the other hand, if a device is not being used enough, it can use the vector to propose others to send to it some load if necessary. This negotiation is applied iteratively every certain amount of time, in such a way that the load distribution is sensitive to new changes caused by the user (opening or closing applications) and at the same time, this produces the greatest possible energy savings.

Finally, after having applied the load balancing algorithm, we demonstrate that the final consumption of a device is optimized and that the applications that run on devices with batteries, can survive longer without being connected to electricity.

Now we go further: To save energy (even at cloud and grid level), it is logically necessary to optimally use all the hardware components, in such a way that they are only used at a time and intensity that is really necessary while understanding its capabilities, behaviour and opportunities. In this article we will use this knowledge to design a middleware that is as aware of the hardware features as possible. This will allow filtering those microservices (in order to reduce scheduling complexity and increase efficiency) that we consider candidates to be treated (moved or duplicated) in relation to the opportunities that each hardware component offers, so that then, these candidates are categorized and ordered according to their consumption characteristics. With this, any distributed scheduling algorithm (which we will implement in our next work) will be able to operate based on intelligent and complete negotiations between the user devices for energy saving.

Thereby, in Section II, we found out which are the most important hardware devices to take into account to analyze the energy consumption in a node/server/user device. In Sections III, IV and V, we will explain the opportunities that each of these hardware devices offers for the best control of energy consumption. In Section VI we propose our proposal middleware architecture. Finally, in Section VII we expose our conclusions and future works.

II. Most energy-expensive hardware components

In order to save energy at a computational level, no matter the level of abstraction, it is necessary to study the consumption of each of the installed devices' components. The question is: Is it necessary to study all these components when designing middlewares (i.e. motherboard, CPU, video card, network card, RAM, etc)?. For us, this will depend on the middleware's goal.

In the context of cloud and data centers, for example, the CPU is one of the main agents of energy consumption. There is a consumption relationship between the CPU, RAM and other components such as the video card or the motherboard (This last components are understood as constants directly proportional to the use of the CPU) [8]. This analysis can also be done by taking into account the daily execution time [6]. On the other hand, in the context of a host, consumption relations have been demonstrated between processing (high efficiency and idle state) and the transfer rate [17] (use of the network) and between the consumption of an entire system and the proper use of storage devices [18].

We believe that all the devices mentioned are important, since they also offer interesting features and opportunities for saving energy in the context of a user's daily life. However, we will also take into account the type of application that is being addressed. Thereby, in the following sections we will analyze the opportunities that the CPU, network devices and storage devices offer for energy saving which can also be exploited from the perspective of our middleware.

III. CPU analysis

The CPU is one of the components with the fastest evolution, since there are always improvements in physical size, frequency, cache, energy consumption, etc. To understand how the CPU consumes electrical power, it is necessary to study the relationship between its way of operating and the amount of voltage / amperage expended. Thus, to measure its power consumption, SPEC CPU2017 [19] performs stress operations while studying the use of voltage and amperage according to the type of benchmark. On the other hand, SpeedStep [20] shows the relationship between the clock frequency and the energy consumed. Logically then, to save energy (even from the middlewares' perspective) at the CPU level, it is necessary to work with the proper frequency in the proper

time, intelligently using the opportunities that processors offer and that we explain below:

A. Automatic overclocking techniques (Performance boosting technologies

In an attempt to increase their efficiency, CPU designers try to raise each core's processing frequency. The challenge is to do it using the least amount of energy possible (and thus, save battery for example) and generating as little heat as possible. One of the ways to do this is by making the CPU work at non energy expensive frequencies in normal conditions and only raise them (i.e. overlocking) when necessary without any user/configuration intervention. Examples of this technique is Intel Turbo Boost [21] , or AMD Turbo Core [22]. For us it is important to understand this feature since it allows: (1) our middleware to request the CPU at maximum processing loads at prudent moments in a implicit or indirect way by applying load balancing algorithms and invoquing operating system configurations (2) and for us to conceive applications taking into account some energetical repercussions in the applications programming using techniques like CPU-multithreading [23] or other source code features [14] [24].

B. Automatic shutdown of CPU cores (PCPG)

On the other hand, as it is possible that the CPU automatically increases its frequency when a high processing capacity is needed, it is possible that it also turns off its cores automatically (users or developers are able to configure this) when they are not being used (e.g. Intel Atom CPU [25]). Jacob Leverith et al. [9] demonstrated that using PCPG is up to 30% more efficient than using DVFS and we believe that this capability can also be used when designing middlewares by: (a) invoking specific operating system configurations, like the same author did by modifying task priorities through the "/proc/stat" file and (b) studying energy consumption models [8] [10] to design energy awareness load balancing algorithms or to design more energetically configurable applications.

C. Frequency reduction techniques invoked by software (DVFS)

Dynamically adjusting the CPU voltage and/or frequency is an opportunity to save energy (e.g.: SpeedStep [26] or AMD PowerNow [27]), and there are several authors who have taken advantage of this feature in software development/research, by for example, relating I / O operations with ideal CPU frequencies [18], modifying kernel governors [12], studying energy impact and cpu frequencies equations to intelligently allocate virtual machines [5] and finding the relationship between the type of task and the clock cycles [8].

We will explain in the next section, how we can take advantage of these characteristics in designing our middleware and in the conception of the applications that it will handle.

D. Middlewares and improvement opportunities

Many authors have developed middlewares with the aim of saving energy by focusing on the use of the CPU, exposing different points of view, such as analyzing/scheduling virtual machines [5] [28] [29]; applying process load balancing algorithms between servers and establishing minimum allowed loads [6]; finding the process/application and CPU consumption relationship to apply load balancing algorithms [30] [8] [13] [18] [29] [24]. These perspectives have shown good results. However, we also believe that in order to achieve a maximum level of energy savings at the CPU level in the context of a middleware, it is necessary not to omit any of the features mentioned above, since perfect energy savings will only occur when in complete awareness of energy expenditure of the hardware. In the particular case of our middleware, we argue that there are two types of information necessary to make good use (i.e. perform load balancing operations) of energy in cpu terms: (1) the energetical characteristics offered by the CPU and (2) how the application and its microservices were conceived. For this second point, we consider two important facts: (1) if a microservice needs to be executed persistently (like a daemon), or if it is a process that will disappear once its task has been completed (for example: analyzing an image); and (2) the impact that the microservice causes to the CPU. These characteristics allow Kaligreen V2 to opportunistically and naturally take advantage of the CPU characteristics previously shown, since CPU energetic characteristics implication is given according to the intensity and time with which they are invoked.

M.S. Features		CPU Features		
Persistent MS	High CPU Consumption	Boosting	PCPG	DVFS
NO	YES	Candidate	Candidate	Candidate
NO	NO	–	–	–
YES	YES	Candidate	Candidate	Candidate
YES	NO	–	Candidate	Candidate

TABLE I
CORRELATION BETWEEN MICROSERVICES FEATURES AND CPU CAPABILITIES

In Table I, we show our criteria for selecting microservices candidates to be moved/duplicated by Kaligreen V2, according to the CPU capabilities and to the microservices characteristics. The table shows, for example, that in the presence of a microservice that needs to be executed permanently (i.e. "persistent microservice") and that has a high computational cost, the middleware will evaluate moving it to a device that does not contain the boosting characteristic. On the other hand, Kaligreen V2 could try to move a microservice that does not have high consumption, when the CPU has the possibility to lower its frequency or turn off cores. This will be decided by the scheduling algorithm by studying the current load of the CPU and other important variables. At the end of the article we will show a heuristic in which we will use these relationships, after detailing the network device and hard disk features in the following sections.

It is important to note that the "high" or "low" criteria expressed in the table will be established according to the device in our next work, since it will depend on the criteria analyzed by a scheduling algorithm.

IV. The Network device analysis

Network devices are one of the most important agents of energy consumption, since regardless of the level of analysis (Section I: cloud, data-center or a single device), the sending of massive information influences the overall efficiency of the system and the level of charge of the network cards. While it is true that it is possible to be aware of the amount of energy expended by each network device in its different states thanks to descriptives jobs like the one done by Salvatore Chiaravalloti [31], and it is possible to perform some basic operations of power management of network devices, such as putting them in sleep mode (e.g. automatic D'link green ethernet [32]) or rate adaptation [33]; we consider, that for the development of a middleware (even more for ours), it is better to leave the management of these capabilities to the operating system and the automatic configuration of drivers, since these directly affect physical phenomena such as the length of a network cable [34] or the distance between two Wi-Fi points (i.e. the capacity of the i.w command , to manage the state and the energy used during the data transmission [35]). Thereby, the best way to manage energy at network level is by: (a) scheduling the amount of data transmitted [36] [37] [38] [39] (b) making improvements on the network topology [40] [39] [41] and (c) improving communication paths [42] [43], even taking into account battery states of each node [17].

As we described in our previous middleware [15], we worked on a topology in which load balancing does not work in a centralized way. This characteristic makes it difficult for us to make improvements in terms of communication paths or topological hierarchies; but it does allow us to focus on the amount of information sent (for us, microservice size and amount of data it sends/receives) and on the transmission protocols when the middleware moves/replies to a microservice.

In Table II, we show the microservices that we consider candidates (Cand.) to be moved, duplicated or eventually also move their data; taking into account its characteristics that we consider most influential at the moment of performing these load balancing operations. These characteristics are: (a) if the microservice is persistent or not (b) if it is heavy at ram level (c) if it causes high network transfer (d) if it communicates with other microservices on the same device: communications between them they will be done through the network if the microservice is moved and (e) if the microservice is related to a large database: Since this could allow moving the microservice with its database if possible.

All this information should be understood as a set of abstract "rules" that our middleware must follow in order to apply a planning algorithm, in which the terms: large, short, small, high and low should be understood according to the properties of the device and thresholds that in future work we will study. For example, there is a microservice that occupies 500kb, fulfills permanent functions (like a daemon) and saturates more than 90% of the bandwidth of the device, our middleware will try to move it, as the first operation, to another device that does not depend on the battery or has a better type of network card.

M.S. Features					Network operations		
Persist. M.S.	Heavy M.S.	Hard NET Usage	Many M.S. relat.	large M.S. data	Move MS	Dup. M.S.	Move M.S. Data
yes	yes	yes	yes	yes	Cand.	Cand.	–
yes	yes	yes	yes	no	Cand.	Cand.	Cand.
yes	yes	yes	no	yes	Cand.	Cand.	–
yes	yes	yes	no	no	Cand.	Cand.	Cand.
yes	yes	no	yes	yes	–	Cand.	–
yes	yes	no	yes	no	–	Cand.	Cand.
yes	yes	no	no	yes	Cand.	Cand.	–
yes	yes	no	no	no	Cand.	Cand.	Cand.
yes	no	yes	yes	yes	Cand.	Cand.	–
yes	no	yes	yes	no	Cand.	Cand.	Cand.
yes	no	yes	no	yes	Cand.	Cand.	–
yes	no	yes	no	no	Cand.	Cand.	Cand.
yes	no	no	yes	yes	Cand.	Cand.	–
yes	no	no	yes	no	Cand.	Cand.	Cand.
yes	no	no	no	yes	Cand.	Cand.	–
yes	no	no	no	no	Cand.	Cand.	Cand.
no	yes	yes	yes	yes	–	–	–
no	yes	yes	yes	no	–	–	Cand.
no	yes	yes	no	yes	Cand.	Cand.	–
no	yes	yes	no	no	Cand.	Cand.	Cand.
no	yes	no	yes	yes	–	Cand.	–
no	yes	no	yes	no	–	Cand.	Cand.
no	yes	no	no	yes	–	–	–
no	yes	no	no	no	–	–	Cand.
no	no	yes	yes	yes	Cand.	Cand.	–
no	no	yes	yes	no	Cand.	Cand.	Cand.
no	no	yes	no	yes	Cand.	Cand.	–
no	no	yes	no	no	Cand.	Cand.	Cand.
no	no	no	yes	yes	–	–	–
no	no	no	yes	no	–	–	Cand.
no	no	no	no	yes	Cand.	Cand.	–
no	no	no	no	no	Cand.	Cand.	Cand.

TABLE II
CORRELATION BETWEEN THE MICROSERVICES FEATURES AND MIDDLEWARE OPERATIONS AT NETWORK LEVEL

On the other hand, if a microservice occupies 1Gb, it does not saturate the network, and the process will end when it has fulfilled its mission (like the calculation of movement in a game), then our middleware will avoid operating with this microservice. On the other hand, if a microservice occupies 1Gb, it does not saturate the network, and the process will end when it has fulfilled its mission (like the calculation of movement in a game), then our middleware will avoid operating with this microservice.

It is important to note that the high or low criteria expressed in the table will be established according to the device in our next work, since it will depend on the criteria analyzed by a scheduling algorithm.

In the next section, we will analyze the opportunities given by storage devices. Finally we will propose an analysis of all the hardware in general so that our middleware can operate.

V. The Hard Disk Analysis

Some hard drives also offer the ability to save energy by changing their working status [44]. For this reason there are some researchers that study the relationship between the I / O requirements and the opportunity to keep the hard drive in a

suspended state [11], even in the context of virtual machines [45].

For us, this feature represents an opportunity in cases where a disk node can be seen as underutilized. In this particular case, for example, our middleware could help free the device from I / O processes.

Condition	Action
if Hdd requirements of Application Microservice == total load of hard disk now	Candidate to move

TABLE III
CORRELATION BETWEEN HARD DISK CAPABILITIES AND MIDDLEWARE ACTIONS

Table III shows the only relation useful for us. That is, to move microservices when they are the only reason why a hard disk is still in an active mode.

Finally, in the next section, we will explain the architecture of Kaligreen V2. Our middleware that will apply planning algorithms based on the hardware's energy opportunities.

VI. KALIGREEN V.2 ARCHITECTURE

Kaligreen [15] is a middleware that applies an iterative and negotiation-based scheduling algorithm in a non-centralized way. That is, each user device belonging to a network, can exchange microservices with its neighbors to balance the processing and transmission load in a collaborative way.

	CPU (1.5ghz)	NET	HDD
MS1	1,3ghz	1Mb/s	2Mb/s
MS2	0,3ghz	0,1Mb/s	0,2Mb/s
MS3	0,1ghz	0,3Mb/s	0,2Mb/s

Vector	1,3ghz	1Mb/s	2Mb/s

Fig. 2. A device in Kaligreen V1

Figure 2 represents an example of the context in which Kaligreen works. Each device Dev, in an undesirable energy situation (i.e. underloading or overloading of a certain hardware component or low remaining battery like the smartphone of the figure 2), builds a metadata vector called $vector$, representing the CPU, network (use of the card + microservice dependencies) and disk consumption of the microservice that is energetically problematic (In case of the figure 2, $MS1$ is the most expensive in terms of energy, because it uses more of the CPU, NET and HDD than the other microservices), which we call MS_P. Then, Dev uses a special microservice called $monitor$ to send $vector$ to all visible devices in the network. Each of these devices will analyze $vector$ and decide if they can process MS_P in a better energetic condition. After that, they return to Dev a processing offer, using the same kind of vector as $vector$, exposing its own components load situation. Finally Dev decides to move MS_P to the candidate that it considers has the best energetic conditions to process it.

In this article, we improve Kaligreen to create its new version 2.0. It has a better level of analysis to determine if a microservice really provokes an energetically problematic condition. We argue that in any environment and at any level where scheduling algorithms are applied, all hardware components should be analyzed in terms of their capabilities and energetic repercussions in order to move (releasing processing load) or duplicate (allowing to distribute the load of an operation in different devices to free transfer rate or use of cpu) a microservice.

For example, if a microservice is launched and causes the CPU to activate its overlocking capability, Kaligreen V.2. will not move or duplicate it if it has a short lifetime (as seen in Table I). On the contrary, if the microservice will be executed permanently (like a demon), Kaligreen V.2. will move it to another device as long as the new energetic conditions are really better (i.e. after analyzing the new CPU, RAM and network repercussions).

The architecture of Kaligreen V2, then, consists of a set of user devices connected together. Each device Dev will have a special microservice called $monitor$, which evaluates the energy status of each hardware components (i.e. CPU, disk and network) every certain amount of time T. $monitor$ detects undesirable energy situations such as: (1) Low battery (2) component overload or (3) component underload. Then, it executes algorithm 1 to filter the microservices (taking into account it's features and the features/capabilities of each device component) into three *non-disjoint* (A microservice can be expensive for the CPU, NETWORK and RAM at the same time) lists of candidates to be moved or replicated in other devices depending on its energetic repercussions.

Algorithm 1 Algorithm 1: Selecting candidate microservices

1: $L\_M \leftarrow List\_of\_all\_Microservices$
2: **while** $true$ **do**
3: $\quad L\_CPU \leftarrow filter\_by\_Table1(L\_M)$
4: $\quad L\_DISK \leftarrow filter\_by\_Table3(L\_M)$
5: $\quad L\_NETWORK \leftarrow filter\_by\_Table2(L\_M)$
6: $\quad SLEEP(T)$
7: **end while**

With Algorithm 1, Kaligreen V.2. filters the microservices that are energy costly at the CPU level (considering their internal properties, such as Boosting) and that have ideal characteristics (i.e. CPU consumption and persistence condition) to be moved or duplicated, obtaining the L_CPU list. Then, our middleware does the same analysis but at the hard disk level, considering the load generated on disk and its ability to save energy through the suspended state, obtaining the L_DISK list. Finally, Kaligreen V.2. analyze the microservices, based on network availability and repercussions (i.e. size of the microservice, dependency links, use of the network card and persistence condition), obtaining the $L\_NETWORK$ list. As a result, we have three *non-disjoint* sets of microservices ideal to be moved or duplicated to reduce energy consumption. We consider that the filtering is useful to avoid that a future

planning algorithm increases its analysis complexity and lose its efficiency without obtaining significantly better results.

After the operations described, Kaligreen V2, through the $monitor$, is able to sort the lists of microservice candidates of each device according to all possible comparison criteria. That is:

1) CPU usage: Amount in percentage of CPU used by each microservice.
2) Network usage. Traffic in Mb/s generated by each microservice.
3) Disk Usage. I/O disk operations in Mb/s generated by each microservice.
4) Overall consumption: Number of joules per unit of time consumed per microservice. For this point, we made our consumption modeling like Abhishek Jaiantilal et al. [8] (i.e. in an additive model way); but instead of taking the cycles as a unit of measure, we will use the measurement units visible from the point of view of a middleware that analyze a particular software entity consumption:

$$MS\_Cons = T*(F(N+extern)+F(D)+F(C)) \quad (1)$$

Where:

- T = Time in seconds
- N=Network transfer rate generated by internal microservice operations (in Kb/s).
  - $externals$ = transfer rate generated by the communication with the linked microservices installed in devices differents to Dev.
  - $F(N)$ = Function to measure the network consumption in $joules/bit$ unit of measurement [46] [47] [48].
- D=Disk I/O transfer rate in Mb/s.
  - $F(D)$ = Function to measure the disk consumption in joules [48].
- C=Microservice CPU requirements in Ghz.
  - $F(C)$ = Function to measure the CPU consumption in joules [48].

Formula 1 calculates the energy consumption generated by a microservice M in a device Dev at a certain moment, taking into account the load that it generates in terms of each of Dev hardware components. It is important to note that in a distributed application, its microservices can be installed in different devices in a certain time . This causes that when these microservices communicate between them, the involved devices network load is increased. This is the reason why we consider the value of $externals$ in the formula.

This diversity of points of view (i.e. orders criteria), will allow to execute any type of scheduling algorithm as appropriate. Kaligreen V2 will be able to select in a dynamic way, through the $monitor$, one or a set of microservices installed in a Device Dev, for example, (a) The microservice(s) that consumes more energy in terms of CPU (b) The microservice(s) that consumes less energy in terms of CPU (c) The third microservice that consumes more energy globally, etc. Then the $monitor$ in Dev, will build a metadata Matrix (before: a vector for only one microservice metadata, in Kaligreen V.1.0), which will be ready to be sent intelligently. Finally, the devices that receive this matrix will be able to analyze several microservices in the best way taking into account the hardware components involved.

Thus, for each device, a future planning algorithm will have the ability to select the ideal microservice (among several represented in the matrix) and move or replicate it. For this reason, in our current work, we are developing a planning algorithm using this architecture, which is based on graph theory and some heuristics similar to those of the SRTF or those of the SJF. This algorithm will be the reason for our next article.

VII. Conclusions

In the present article, we present an evolution of Kaligreen's architecture called Kaligreen V2. It consists of a set of user devices connected together using a special monitoring microservice called $monitor$, to generate awareness of the capacities of the hardware components (i.e. CPU, network and hard disk) of each device. The $monitor$ studies the correlation of the characteristics of each microservice running on a device and the energy opportunities of its hardware components, in order to filter only those microservices that are candidates to be moved (release processing load directly) or duplicates (delegate part of the load to another device) by our middleware. Kaligreen V.2. does this filtering by generating 3 sorted groups: The candidate microservices from the point of view CPU, network and hard disk.

This architecture will allow the implementation of any scheduling algorithm based on decentralized negotiations, with awareness of the particular energy opportunities of each hardware component. Our next work will focus on the development of this point.

References

[1] A. Shehabi, S. J. Smith, D. A. Sartor, R. E. Brown, M. Herrlin, J. G. Koomey, E. R. Masanet, N. Horner, I. L. Azevedo, and W. Lintner, "United states data center energy usage report," Tech. Rep., 06/2016 2016.

[2] G. Procaccianti, P. Lago, and G. A. Lewis, "Green architectural tactics for the cloud," in *Proceedings of the 2014 IEEE/IFIP Conference on Software Architecture*, ser. WICSA '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 41–44. [Online]. Available: https://doi.org/10.1109/WICSA.2014.30

[3] M. I. Hassan and R. Bahsoon, "Green-as-a-service (gaas) for cloud service provision operation," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 1219–1220. [Online]. Available: http://doi.acm.org/10.1145/2554850.2555182

[4] G. R. Howard and S. Lubbe, "Synthesis of green is frameworks for achieving strong environmental sustainability in organisations," in *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, ser. SAICSIT '12. New York, NY, USA: ACM, 2012, pp. 306–315. [Online]. Available: http://doi.acm.org/10.1145/2389836.2389873

[5] N. M. Azmy, I. A. El-Maddah, and H. K. Mohamed, "Adaptive power panel of cloud computing controlling cloud power consumption," in *Proceedings of the 2Nd Africa and Middle East Conference on Software Engineering*, ser. AMECSE '16. New York, NY, USA: ACM, 2016, pp. 9–14. [Online]. Available: http://doi.acm.org/10.1145/2944165.2944167

[6] I. Siddavatam, E. Johri, and D. Patole, "Optimization of load balancing algorithm for green it," in *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, ser. ICWET '11. New York, NY, USA: ACM, 2011, pp. 1344–1346. [Online]. Available: http://doi.acm.org/10.1145/1980022.1980321

[7] M. Pawlish, A. S. Varde, S. A. Robila, and A. Ranganathan, "A call for energy efficiency in data centers," *SIGMOD Rec.*, vol. 43, no. 1, pp. 45–51, May 2014. [Online]. Available: http://doi.acm.org/10.1145/2627692.2627703

[8] A. Jaiantilal, Y. Jiang, and S. Mishra, "Modeling cpu energy consumption for energy efficient scheduling," in *Proceedings of the 1st Workshop on Green Computing*, ser. GCM '10. New York, NY, USA: ACM, 2010, pp. 10–15. [Online]. Available: http://doi.acm.org/10.1145/1925013.1925015

[9] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, "Power management of datacenter workloads using per-core power gating," *Computer Architecture Letters*, vol. 8, pp. 48–51, 02 2009.

[10] N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram, "Guarded power gating in a multi-core setting," in *Proceedings of the 2010 International Conference on Computer Architecture*, ser. ISCA'10. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 198–210. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24322-6_17

[11] F. Chen and X. Zhang, "Caching for bursts (c-burst): Let hard disks sleep well and work energetically," in *Proceedings of the 2008 International Symposium on Low Power Electronics & Design*, ser. ISLPED '08. New York, NY, USA: ACM, 2008, pp. 141–146. [Online]. Available: http://doi.acm.org/10.1145/1393921.1393961

[12] L. Corral, A. B. Georgiev, A. Janes, and S. Kofler, "Energy-aware performance evaluation of android custom kernels," in *Proceedings of the Fourth International Workshop on Green and Sustainable Software*, ser. GREENS '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 1–7. [Online]. Available: http://dl.acm.org/citation.cfm?id=2820158.2820160

[13] L. Corral, A. B. Georgiev, A. Sillitti, and G. Succi, "Method reallocation to reduce energy consumption: An implementation in android os," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 1213–1218. [Online]. Available: http://doi.acm.org/10.1145/2554850.2555064

[14] N. Nikzad, O. Chipara, and W. G. Griswold, "Ape: An annotation language and middleware for energy-efficient mobile application development," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 515–526. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568288

[15] H. H. lvarez Valera, P. Roose, M. Dalmau, C. Herzog, and K. Respicio, "Kaligreen: A distributed scheduler for energy saving," *Procedia Computer Science*, vol. 141, pp. 223 – 230, 2018, the 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050918318222

[16] K. Da, M. Dalmau, and P. Roose, "Kalimucho: Middleware for mobile applications," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 413–419. [Online]. Available: http://doi.acm.org/10.1145/2554850.2554883

[17] P. Zhang and M. Martonosi, "Energy adaptation techniques to optimize data delivery in store-and-forward sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 405–406. [Online]. Available: http://doi.acm.org/10.1145/1182807.1182878

[18] R. Ge, X. Feng, and X.-H. Sun, "Sera-io: Integrating energy consciousness into parallel i/o middleware," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, ser. CCGRID '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 204–211. [Online]. Available: https://doi.org/10.1109/CCGrid.2012.39

[19] J. Bucek, K.-D. Lange, and J. v. Kistowski, "Spec cpu2017: Next-generation compute benchmark," in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '18. New York, NY, USA: ACM, 2018, pp. 41–42. [Online]. Available: http://doi.acm.org/10.1145/3185768.3185771

[20] Intel, "Intel - speedstep." [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000005723/processors.html

[21] I. TurboBoost, "Intel - turboboost." [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000007359/processors/intel-core-processors.html

[22] AMD, "Amd - turbocore." [Online]. Available: https://www.amd.com/en/technologies/turbo-core

[23] S. Kondguli and M. Huang, "A case for a more effective, power-efficient turbo boosting," *ACM Trans. Archit. Code Optim.*, vol. 15, no. 1, pp. 5:1–5:22, Mar. 2018. [Online]. Available: http://doi.acm.org/10.1145/3170433

[24] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "Estimating android applications' cpu energy usage via bytecode profiling," in *Proceedings of the First International Workshop on Green and Sustainable Software*, ser. GREENS '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 1–7. [Online]. Available: http://dl.acm.org/citation.cfm?id=2663779.2663780

[25] Intel-Atom, "Intel atom - power gating." [Online]. Available: https://www.intel.com/content/www/us/en/embedded/software/emgd/emgd-dynamic-power-gating-paper.html

[26] Intel-SpeedTest, "Intel-speedtest." [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000005723/processors.html

[27] AMD-PowerNow, "Amd-powernow." [Online]. Available: http://www.amd-k6.com/wp-content/uploads/2012/07/24404a.pdf

[28] D. Seo, "A study of workload consolidation and power consumption on a multi-core processor," in *Proceedings of the 2012 ACM Research in Applied Computation Symposium*, ser. RACS '12. New York, NY, USA: ACM, 2012, pp. 457–458. [Online]. Available: http://doi.acm.org/10.1145/2401603.2401702

[29] Q. Chen and J. Li, "The balance mechanism of power and performance in the virtualization," in *Proceedings of the Second International Conference on Innovative Computing and Cloud Computing*, ser. ICCC '13. New York, NY, USA: ACM, 2013, pp. 189:189–189:192. [Online]. Available: http://doi.acm.org/10.1145/2556871.2556912

[30] N. Yigitbasi, K. Datta, N. Jain, and T. Willke, "Energy efficient scheduling of mapreduce workloads on heterogeneous clusters," in *Green Computing Middleware on Proceedings of the 2Nd International Workshop*, ser. GCM '11. New York, NY, USA: ACM, 2011, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/2088996.2088997

[31] S. Chiaravalloti, F. Idzikowski, and L. Budzisz, "Power consumption of wlan network elements," 08 2011.

[32] dlink Green, "Technologie d-link green." [Online]. Available: https://eu.dlink.com/fr/fr/support/faq/knowledge/technologie-dlink-green

[33] V. D. Maio, V. Nae, and R. Prodan, "Evaluating energy efficiency of gigabit ethernet and infiniband software stacks in data centres," in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, ser. UCC '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 21–28. [Online]. Available: http://dx.doi.org/10.1109/UCC.2014.10

[34] dlink, "Dlinkgreen." [Online]. Available: http://www.dlinkgreen.com/energyefficiency.asp

[35] L. man page, "iw." [Online]. Available: https://linux.die.net/man/8/iw

[36] S. Kiertscher and B. Schnor, "Scalability evaluation of an energy-aware resource management system for clusters of web servers," in *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, ser. Spects '15. San Diego, CA, USA: Society for Computer Simulation International, 2015, pp. 1–8. [Online]. Available: http://dl.acm.org/citation.cfm?id=2874988.2875004

[37] N. P. Esfahani and A. E. Cerpa, "Poster: Energy optimization framework in wireless sensor network," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '15. New York, NY, USA: ACM, 2015, pp. 441–442. [Online]. Available: http://doi.acm.org/10.1145/2809695.2817904

[38] K. Zhan, C.-H. Lung, and P. Srivastava, "A green analysis of mobile cloud computing applications," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 357–362. [Online]. Available: http://doi.acm.org/10.1145/2554850.2555069

[39] J. Jo, Y. Kim, K. H. Lee, S. H. Cho, and J. H. Kim, "The energy saving strategy using the network coding in the wireless mesh network," in *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '12. New York, NY, USA: ACM, 2012, pp. 127:1–127:4. [Online]. Available: http://doi.acm.org/10.1145/2184751.2184895

[40] T. Malik, L. Nistor, and A. Gehani, "Middleware for managing provenance metadata," in *Middleware '10 Posters and Demos Track*, ser. Middleware Posters '10. New York, NY, USA: ACM, 2010, pp. 5:1–5:2. [Online]. Available: http://doi.acm.org/10.1145/1930028.1930033

[41] K. Bao, I. Mauser, S. Kochanneck, H. Xu, and H. Schmeck, "A microservice architecture for the intranet of things and energy in smart buildings: Research paper," in *Proceedings of the 1st International Workshop on Mashups of Things and APIs*, ser. MOTA '16. New York, NY, USA: ACM, 2016, pp. 3:1–3:6. [Online]. Available: http://doi.acm.org/10.1145/3007203.3007215

[42] E. Yaacoub, A. Kadri, and A. Abu-Dayya, "Cooperative wireless sensor networks for green internet of things," in *Proceedings of the 8H ACM Symposium on QoS and Security for Wireless and Mobile Networks*, ser. Q2SWinet '12. New York, NY, USA: ACM, 2012, pp. 79–80. [Online]. Available: http://doi.acm.org/10.1145/2387218.2387235

[43] Y.-F. Lu, J. Wu, and C.-F. Kuo, "A path generation scheme for real-time green internet of things," *SIGAPP Appl. Comput. Rev.*, vol. 14, no. 2, pp. 45–58, Jun. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656864.2656868

[44] Seagate, "Seagate-baracuda." [Online]. Available: https://www.seagate.com/www-content/product-content/barracuda-fam/barracuda-new/en-us/docs/100804187g.pdf

[45] L. Ye, G. Lu, S. Kumar, C. Gniady, and J. H. Hartman, "Energy-efficient storage in virtual machine environments," in *Proceedings of the 6th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '10. New York, NY, USA: ACM, 2010, pp. 75–84. [Online]. Available: http://doi.acm.org/10.1145/1735997.1736009

[46] M. Yan, C. A. Chan, A. F. Gygax, J. Yan, L. Campbell, A. Nirmalathas, and C. Leckie, "Modeling the total energy consumption of mobile network services and applications," *Energies*, vol. 12, no. 1, 2019. [Online]. Available: http://www.mdpi.com/1996-1073/12/1/184

[47] E. Björnson and E. G. Larsson, "How energy-efficient can a wireless communication system become?" *CoRR*, vol. abs/1812.01688, 2018. [Online]. Available: http://arxiv.org/abs/1812.01688

[48] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, Firstquarter 2016.