A Novel Energy-Driven Computing Paradigm for e-Health Scenarios

Marina Zapater , Patricia Arroba , José L. Ayala , José M. Moya , Katzalin Olcoz

Abstract

A first-rate e-Health system saves lives, provides better patient care, allows complex but useful epidemiologic analysis and saves money. However, there may also be concerns about the costs and complexities associated with e-health implementation, and the need to solve issues about the energy footprint of the high-demanding computing facilities. This paper proposes a novel and evolved computing paradigm that: i) provides the required computing and sensing resources; ii) allows the population-wide diffusion; iii) exploits the storage, communication and computing services provided by the Cloud; iv) tackles the energy-optimization issue as a first-class requirement, taking it into account during the whole development cycle. The novel computing concept, and the multi-layer top-down energy-optimization methodology, obtain promising results in a realistic scenario for cardiovascular tracking and analysis, making the *Home Assisted Living* a reality.

Keywords:
Energy efficiency, Resource management, Data centers, Population analysis, Green IT, Cloud computing

1. Introduction

e-Health is a new concept of health management that produces several benefits. First, it reduces sanitary costs by prevention of potential diseases. Besides, it empowers the patients with a new generation of non-invasive, wearable personalized devices to make them more independent, and to provide early signals of health decline and advice for appropriate actions in daily life. Finally, analysis of the obtained data greatly improves prevention by detecting early patterns of potential diseases; it allows to evaluate the efficacy of treatments, to understand (through complex processing) the evolution of diseases and the factors that influence them. Biomedical engineers envision "a new system of distributed computing tools that will collect authorized medical data about people and store it securely within a network designed to help deliver quick and efficient care" [1].

In order to obtain such benefits, the target population has to be monitorized 24 hours a day, and a Wireless Body Sensor Network (WBSN) is deployed. Thus, the system is composed by a large set of nodes, distributed among the population. Such nodes are non intrusive and portable, which imposes constraints on their energy consumption. Data obtained by the sensors are communicated to the embedded processing elements (PDAs, smartphones, etc.) by means of wireless connections.

Then, the huge set of data must be analyzed with the aim of performing the epidemiologic assessment. Also, diagnosis algorithms have to be implemented to allow early detection of pathologies and to learn the evolution of patients.

Since the target population is large, so it is the number of sensing nodes, and the amount of data to be managed is huge. In order to deal efficiently with such computationally intensive tasks, the use of cloud services is devised. Cloud computing is emerging as the dominant computer platform for scalable online services. Thus, the WBSNs will be connected not only at the node level, but also through the PDA, Smartphone, etc to the Cloud. Part of the data processing and storage will be local to the node, while another part will be communicated and processed in the cloud, depending on the application, on the state of the batteries and on security or privacy requirements of the information. The availability of the aforementioned technologies and the need of a continuous, portable, and non-invasive monitoring of the health information has led us to envision and design a Cloud computing-based real-time health monitoring and analysis framework capable of aiding health-care professionals. This computing environment where the mobile client utilizes mobile network services to communicate with cloud through the Internet is called *Mobile Cloud Computing* (MCC) [2]. Mobile cloud computing can address the problem of scalability by executing mobile applications on resource providers external to the mobile device.

According to [3], one of the main questions to be answered in MCC is how computation can be offloaded and distributed to the cloud efficiently. The reasons for sharing/offloading work from a mobile device would be: limited computational capability, limited battery power, limited connectivity and to make use of idling processing power.

The focus of this work is proposing a novel multi-layered

approach for the energy optimization of MCC technologies, and the validation with a case of study devoted to health monitoring and analysis applications.

1.1. Related Work

The use of MCC environments for the automation of personal health-care systems has been recently related in literature [4, 5, 6]; however, none of these works have approached the energy efficiency in mobile cloud architectures.

Energy consumption is one of the major concerns for the adoption of population-wide health monitoring systems, but energy efficiency cannot be added as an afterthought. Truly energy-efficient monitoring can only be achieved by considering energy as a first-class requirement, taking it into account during the whole development cycle, from design to implementation. Thus, we propose an architecture driven by energy concerns and aimed at optimizing energy consumption globally.

In literature, we can find several energy optimization techniques that target the different abstraction levels of the MCC architecture.

At the distributed computing level, the design of WBSN nodes is mainly focused on maximizing the lifetime of the node by reducing the energy consumption, although other performance requirements such as the delay and quality of the delivered data must be kept into account [7]. Energy efficiency in WBSNs has been tackled by proposing efficient MAC layer alternatives [8, 9], providing stochastic approaches for traffic handling [10] or enabling compressed sensing signal acquisition/compression algorithms [11].

At the server level, one of the main problems to be solved in order to achieve the performance goal is the so called *power-wall*. Semiconductor manufacturers are reaching the limits of voltage scaling, no longer reducing power consumption in new chips. Thus, power consumption limits the advances in computer technology and is becoming a relevant part on the budget of present data centers. According to [12] power is the second-highest operating cost in 70% of all data centers and data centers are responsible for the emission of tens of millions of metric tons of carbon dioxide annually, more than 2% of the total global emissions. As a result there has been as well a recent research interest in the development of energy efficient data centers.

The researchers have done a massive amount of work [13, 14, 15, 16, 17, 18, 19, 20] to provide an energy-aware high performance computing environment. In these works, different scheduling, resource allocation and work assignment mechanisms are studied to improve the energy profile. Multi-layered approaches like ours, that targets both the node and server levels, are still missing.

Some energy optimization policies have been detected but not successfully proposed mainly due to the fact that they do not consider the global power consumption. In particular, they do not take into account the following:

1. that the agents involved in the problem (wireless nodes, embedded processors, network interfaces, high-performance servers, etc.) are very heterogeneous from the energy point of view. Therefore, the energy cost of performing part of the processing in any of the different abstraction layers, from the node to the data center, should be evaluated;
2. a local optimization in one of the abstraction layers can have a bigger negative impact on the others, so that the global energy of the system is increased. In this way, the relationships between all the computational agents have to be taken into account.

Our proposal develops global energy optimization policies that start from the design of the architecture of the system and take into account the energy relationship between the different abstraction layers.

In our work, we manage the whole set of abstraction levels in MCC to obtain the maximum benefit of the energy-aware policies. Among others, we consider computation offloading from the Cloud to the wireless nodes (and viceversa) as an effective mechanism for energy optimization. Computation offloading in MCC scenarios has been proposed by Kumar and Lu [21] and probed to provide high benefits. However, the authors have not considered realistic scenarios like e-Health and have not proposed a multi-layered optimization approach that combines this technique with other optimization mechanisms.

Some authors have recently followed a similar approach to our multi-layered proposal. For example, [22] described a research work on how to reduce GPS power consumption by offloading certain calculations onto the cloud. However, to the best of our knowledge, this is the first time that a work targets for energy optimization purposes the several constituent layers that enable MCC in e-Health scenarios. Our work provides horizontal and vertical approaches to extend the energy savings that these environments require.

1.2. Contributions

This paper makes the following contributions:

- we define a realistic and current application scenario where the computing and energy saving challenges are exposed;
- we propose a new highly-efficient computational paradigm;
- we demonstrate that the application characteristics must be considered in the computational paradigm since the very beginning of the design process;
- we propose a global strategy for energy efficiency in the computational architecture.

The remainder of this paper is organized as follows: Section 2 describes our envisioned energy efficient computational paradigm, whereas Section 3 details the particular case study. Section 4 describes the models used and Sections 5 and 6 show the different optimizations applied. Section 7 integrates the results obtained so far into a multi-layer approach and describes the new challenges. Finally, the most important conclusions are drawn in Section 8.

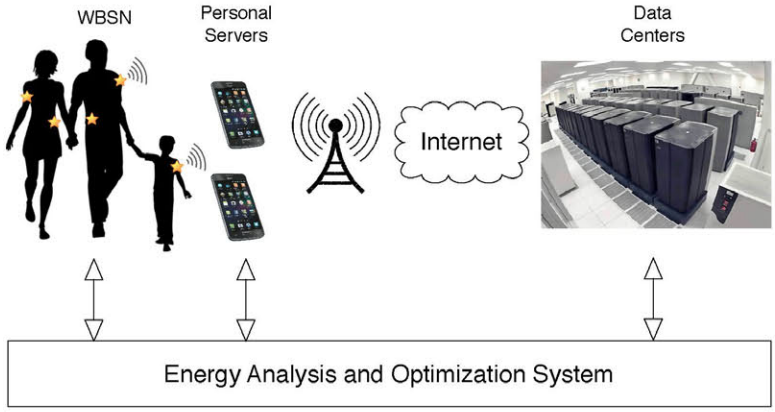

Figure 1: Overview of the proposed architecture for energy optimization of MCC technologies

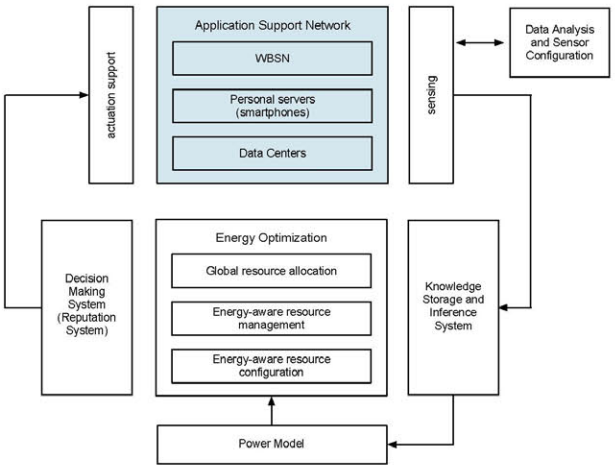

Figure 2: Overview of the proposed energy analysis and optimization system for population analysis applications

2. Devised computer paradigm

As previously described, our envisioned MCC e-Health system is composed of a number of body sensors, wirelessly connected to the cloud through a mobile processing device (as illustrated in Figure 1). The distributed system spans a network comprised of individual health monitoring systems that connect through the Internet to data center facilities.

To provide adequate energy management, this heterogeneous distributed computing system for health monitoring is tightly coupled with an energy analysis and optimization system, which continuously adapts the amount of processing that is performed in the different layers of the distributed system, and the resources assigned to each task.

It is important to stress the need for a top-down approach, driven by the application context and the energy constraints, in order to reach an optimum solution globally.

2.1. Energy optimization system

Figure 2 shows the proposed system architecture for the energy optimization of MCC technologies. Detailed functions of constituents in the system are summarized as follows:

- Application support network: Population analysis applications require an heterogeneous network comprised of sensor nodes, data centers, and some kind of interconnection network. Each node has different computation capacity, functional requirements, communication capacity, battery capacity, power consumption characteristics, etc.
- Sensing infrastructure: Global energy optimization requires a clear understanding of the current state of the network, the characteristics of the different resources and of the analysis to be performed. Therefore, additional HW and/or SW sensors should be added to the system to get an insight.
- Data analysis and sensor configuration: Not every sensor has the same importance to understand the power consumption characteristics of the different components. After a careful analysis, the sensing infrastructure is configured to provide only the relevant data at the required rate for the power model to be useful, and also to minimize the energy overhead of the energy optimization system.
- Storage and inference system: Data provided by the sensing infrastructure has to be stored and statistically analyzed in search of recurrent behaviors that could lead to simple but accurate power models to be used for proactive optimizations. Although the data provided by the sensors is very low-level, simple inference techniques can be used to raise the level of abstraction, for example, to understand the characteristics of energy demand by the different applications.
- Power model: Complex power models are not adequate for online optimization, as different alternatives should be quickly evaluated against the power model to proactively configure the whole system for minimum energy consumption. These power models can be trained with actual data from sensors in order to improve the quality and also to adapt to variations in the heterogeneous application support network.

	Node	Coordinator	Cloud infrastructure
Resource allocation	*Global resource allocation*		
Resource Management (RM) & Configuration	WBSN adaptation [23]	Virtual architecture [24]	*Data center RM* *Cooling control*
Application/Compiler	ECG algorithms [11] [25]	Compiler and application-level tools [26] [27]	*Virtualization*
Architecture	Loop buffering [28] [29] IMO [33]	RAM optimizations and PCM [30]	DVFS [31] [32]

Table 1: Summary of optimizations for different elements of the architecture and abstraction levels

- Optimization: Based on the current state of the system, the historic data, and the energy characteristics of application and resources, many optimization algorithms can be executed to enhance one or more aspects of the population monitoring system. Heterogeneity can be analysed to always assign tasks to the most adequate resources; resources not being used can be turned off and cooling energy can be taken into account when assigning tasks to resources.
- Decision making system: All the different partial optimizations obtained in the system should be integrated in order to obtain the overall energy savings of the architecture.
- Actuation support: Finally, decisions should be executed. Software agents in the body sensors, personal servers (smartphones or PDAs), and data centers are in charge of re-configuring their behavior whenever an optimization decision is made.

The energy optimizations that could be applied at the different elements of the architecture and at different levels of abstraction for the particular case of applications for ECG population analysis are summarized in Table 1. Research has paid attention to some of the above mentioned aspects, however, there are still some areas where contributions are needed. Moreover, these different contributions need to be integrated in a top-down fashion that ensures the global energy minimization.

Our paper makes contributions in the areas not covered by other authors, emphasized in the table, as well as on the integration of all these optimizations. The goal is to perform a multi-objective energy-optimization in all the abstraction levels of design (vertical integration) as well as in all the components of the architecture (horizontal integration). The results of these optimizations are globally evaluated to obtain the energy savings for the whole architecture and observe the impact of each optimization.

3. Case study

In this section we describe a particular case study for ECG monitoring applications that we use throughout the paper to apply all energy models and optimizations (Sections 4, 5 and 6), and that is evaluated from a multi-layer perspective in Section 7. The architecture previously mentioned for MCC technologies includes 4 types of elements: (i) sensors, (ii) nodes, (iii) coordinators and (iv) a cloud computing facility. In our case study:

- Sensor: ECG sensors placed on the chest of the subject to record the signals of interest.
- Node: Shimmer platform. Composed of a Texas Instruments MSP430, a low power IEEE 802.15.4-compliant radio (CC2420) and Bluetooth radio (not used due to high power consumption). HW characteristics: CPU 8MHz, 10KB RAM, 48KB Flash. The Shimmer platform is placed near the sensors and connected to them with wires. This node can perform some conditioning on the signals received and sends them to the coordinator via radio. If more than one ECG sensor is used, they are connected to the coordinator describing a star topology.
- Coordinator: Android-based smartphone. HW characteristics: CPU 1GHz, 1GB of RAM, 16GB of Flash Memory and a battery of 2000mAh. The coordinator is usually placed at the waist of the subject and receives information from all the nodes that the person is wearing. It can perform some computations on the received signals (see Section 6.1) and it forwards data to the cloud computing facility via a 3G or WiFi network.
- Cloud computing facility: Modern data centers are equipped with a large number of enterprise servers. Because of budget limits when renewing the equipment, in these facilities we usually find different generations of machines, often from even different manufacturers. For this case study we assume a heterogeneous cloud computing facility with two different servers (Intel and AMD) from different generations. The Intel server was first shipped in 2010, while the AMD is much older, from year 2003.

Table 2 summarizes the properties of the different components of the architecture. Note that maximum power for rack mounted servers indicates the maximum power measured when fully utilizing the system, not the maximum power that can be drawn by the power supply. Thus, this power is dependant on the particular hardware configuration of our servers

This case study considers a deployment consisting on 300 Shimmer nodes, 300 coordinators (smartphones) and a data center where a total amount of 160 cores, belonging to 40 Intel or AMD machines, are dedicated to our computational needs and placed in an air-cooled data room.

We define three different workload profiles: (i) heavy, (ii) reference and (iii) light workload depending on the amount of tasks and their arrival rate. These profiles emulate the typical distributions for workloads that arrive to a computing facility [34], in which the workload might show different temporal

Component	Model	Processor	#Cores	Memory	Idle Power	Max Power
Node	Shimmer @8MHz	MSP430	1	10KB	6.6mW	85mW
Coordinator	Samsung Galaxy SII	ARM Cortex-A9 @1.2GHz	2	1GB	0.5W	2.5W
Rack Server	SunFire v20z	2x AMD Opteron @2GHz	2	4GB	122W	220W
Rack Server	RX-300 S6	Intel Xeon @2.4GHz	8	16GB	140W	200W

Table 2: Summary of properties for all architecture components

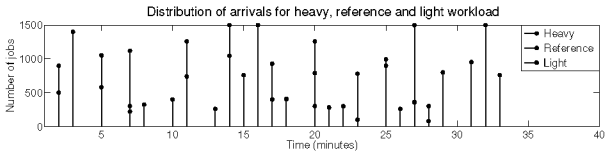

Figure 3: Distribution of arrivals for high, medium and low loads

patterns of utilization at concrete periods or at certain hours of the day [35]. All three workloads are organized in 10 different job sets that arrive following a Poisson distribution with an average of 30 minutes. Each job set consists of a burst of tasks to be executed. The amount of tasks per job set follows a uniform distribution that depends on the workload profile: 1,000 to 1,500 tasks for the heavy workload profile, 500 to 1200 tasks for the reference workload and 300 to 700 tasks for the light workload profile. The arrival rate for the three workloads is the one presented in Figure 3.

The different tasks of the workload are representative of the computation that has to be performed for e-Health monitoring applications and information extraction in data centers. Particularly, we use the following benchmarks and algorithms:

- Customized state-of-the-art ECG acquisition, compression and delineation algorithms, such as Compressed Sensing or Digital Wavelet Transform, as well as encryption and decryption algorithms (e.g. AES) for secure data transmission.
- Statistical analysis algorithms, obtained from the IBM SPSS Statistics software. We choose six applications commonly used to extract information from the data obtained by bio-medical sensor nodes, such as correlation analysis, data regressions, estimation of data parameters and statistical classification.
- CPU-intensive tasks, obtained from the SPEC CPU 2006 benchmark suite [36], representing algorithms of a higher abstraction level for the complex analysis and representation performed over data that has already gone through a data analysis and conditioning step. We choose the 12 tasks of the integer benchmark, among which we find data interpreters, decompression algorithms, combinatorial optimizations, database searching algorithms or event simulations.

	Tasks	Execution Time	Instructions per Cycle
Low demanding	6 (regression)	<360sec	<1.3
Medium demanding	6 (gcc, mcf)	360-600sec	<1.3
High demanding	6 (bzip2, hmmer)	460-800sec	>1.3

Table 3: Classification and main parameters for the tasks of the workload

This workload must be known and profiled in advance for our experimental study. Data centers usually execute the same set of applications, what facilitates this knowledge extraction. Moreover, modern supercomputing infrastructures like CeSViMa [37] provide a mechanism for fast application profiling before the actual execution.

For our purpose, we suppose that each job set is split in two different levels: (i) a Data-Dependant layer, in which most of the algorithms and computation are dependant on the data generated by the Shimmer nodes; and (ii) an Application-dependant layer, in which algorithms operate over data generated in the previous layer, and computation depends on the particular goal that has to be achieved. Because of the number of nodes deployed, we assume that in our workload, a 60% of tasks belong to the Data-Dependant layer and the other 40% to the Application-dependant layer.

Moreover, because of the different nature of the algorithms to be executed, we assume that the tasks of the workload can be split into different classes according to the computing resources needed for execution. We define three different classes: (i) high-demanding applications, (ii) medium-demanding applications and (iii) low-demanding applications. Tasks are classified into a particular category attending to their computational demand by means of the *k-means* clustering technique presented in Section 5. Table 3 shows the amount of tasks per category, example tasks for each category as well as two of the most important parameters for classification criteria (execution time and instructions per cycle).

Finally, each of the two layers contains a different percentage of each of these tasks. The Data Dependant layer contains a 70% of low-demanding tasks, a 25% of medium-demanding and a 5% of high-demanding tasks. The Application-dependant

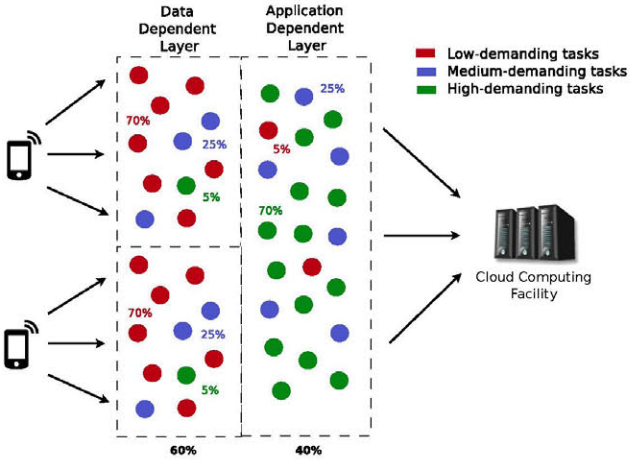

Figure 4: Workload structure for the Data Dependant layer and the Application Dependant layer

layer contains a 70% of high-demanding tasks, a 25% of medium-demanding tasks and a 5% of low-demanding tasks. Figure 4 summarizes the main parameters of the workload and its structure.

4. Power models

In order to optimize energy consumption of the overal MCC environment, we first need to understand which are the different main contributors to the overall power consumption. After that, we will be able to develop power models that explain the behavior of the different elements of our architecture. In this section we present the power modeling for the nodes, coordinators and the cloud infrastructure that are managed by the optimization algorithms presented in Section 6.

4.1. Node model

The main contributors to the energy consumption at the node level are the sensors themselves (signal transducing and analog-to-digital conversion), the microcontroller (because of the calculations performed), the memory and the radio interface. The energy of the node (E_{node}) can thus be described as the sum of those terms:

$$E_{node} \approx E_{sensor} + E_{\mu C} + E_{mem} + E_{radio} \quad (1)$$

Figure 5 shows the power consumption trace of the Shimmer node running a simple ECG streaming application. The Shimmer platform implements a reduced version of the beacon-enabled mode of the IEEE 802.15.4 protocol that uses guaranteed time slots (GTS). The transmission consists of three phases: (i) the beacon reception, in which the radio is receiving and the microcontroller is idle, (ii) Low-Power mode until the start of the assigned GTS, and (iii) transmission of the ECG signal to the coordinator during the GTS. During these phases, microcontroller and radio go through different power states [11].

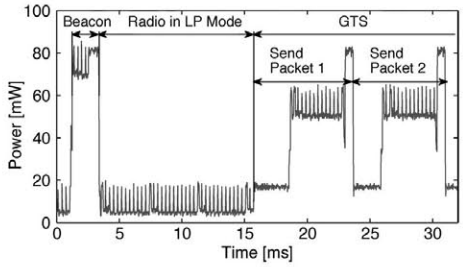

Figure 5: Power dissipated in Shimmer during sampling, processing and transmission [11]

From the energy perspective, there is a trade-off between the amount of information sent through the radio link and the signal processing performed at the microcontroller. In ECG applications, there are several frequently used algorithms for signal compression and reconstruction that are performed at the node level. The most common are: (i) Compressed Sensing (CS), (ii) Digital Wavelet Transform (DWT) and (iii) Multi-lead DWT.

Works by Mamaghanian [11] and Rincon [25] show the energy differences when implementing these algorithms in the Shimmer platform. Table 4 summarizes the different node battery lifetime encountered depending on the algorithms and transmission strategies performed.

Results show that total energy consumption increases with the computational burden of the algorithms and, thus, battery life is reduced. However, the radio interface is not always the responsible for most of the energy consumption.

The previous energy results for different ECG algorithms are used in the optimizations in Sections 5 and 7 to optimize the overall consumption by properly balancing the power consumption of the node elements and the coordinators.

4.2. Coordinator energy modeling

Our efforts in the energy modeling of the coordinator nodes (i.e. the smartphones) focus on being able to describe the impact of running the MCC algorithms on the smartphone battery life.

The main contributors to the power consumption in todays smartphones are the communications (GSM, Wifi, etc.), graphics and the CPU when the system is suspended (i.e. most of the time) and also the display when the system is idle [38]:

Because the Shimmer nodes are responsible for the wireless transmissions, the Shimmer attached to the smartphone is responsible for the radio reception instead of the smartphone itself, meaning that the ECG algorithms are a computational burden that has an impact mainly on the microcontroller power consumption.

Because of that, we consider that the energy consumed by the coordinator (E_{coord}) can be described as in Equation 2:

$$E_{coord} \approx E_{comm,idle} + E_{graphics,idle} + E_{\mu C} \quad (2)$$

where $E_{comm,idle}$ and $E_{graphics,idle}$ are the idle power for communications and graphics and $E_{\mu C}$ is the power consumption for the microcontroller, which varies depending on the algorithm executed. In order to characterize the power consumption

	ECG Streaming	CS	Single lead	2-lead Morph.	2-lead spline
Packed ready every... (*ms*)	304	605.9	2250	2250	2250
Energy Consumption (*mJ*)	7.70	7.29	7.42	8.68	10.12
Lifetime (*h*)	134.6	142.1	139.6	119.3	102.4

Table 4: Node lifetime for the different algorithms

of the microcontroller we use the *Lookbusy* synthetic workload to stress the system during monitored periods of time. *Lookbusy* can stress all the hardware threads to a fixed CPU utilization percentage without memory or disk usage. The usage of a synthetic workload to derive the CPU model has many advantages, the most important of which is that CPU power can be described as linearly dependent with CPU utilization (u_c) and Instructions Per Cycle (IPC), as seen in Equation 3:

$$E_{\mu C,coord} = A_c \cdot u_c + K_c \tag{3}$$

where A_c is a constant. Our coordinator nodes (Samsung Galaxy SII smartphones) are equipped with an ARM Cortex-A9 processor at 1GHz. This processor is commonly found in many embedded system devices, such as the Panda board [1]. To ease the profiling process, we characterize the ARM Cortex-A9 processor in the Panda board by measuring the energy consumption with a Fluke 80i-110s AC/DC current clamp when running *Lookbusy* at different utilization values. We then fit this data by means of a MATLAB regression to obtain constants A_c and K_c.

4.3. Data Center power modeling

The main contributors to the energy consumption in a data center are the computing power (also known as IT power), which is the power drawn by servers in order to run a certain workload, and the cooling power needed to keep the servers within a certain temperature range that ensures safe operation. Together, they account for more than 85% of the total power consumption of the Data Center, being the other 15% the power consumption due to lightning, generators, UPS systems and PDU (power distribution units) [39].

$$P_{DC} = P_{IT} + P_{cooling} + P_{others} \tag{4}$$

The IT power is dominated by the power consumption of the enterprise servers in the data center. The power consumption of an enterprise server can be divided into three different contributors: (i) the dynamic or active power, (ii) the static or leakage power and (iii) the cooling power, due to the server fans:

$$P_{server} = P_{static} + P_{dynamic} + P_{fan} \tag{5}$$

Dynamic power is the power due to the switching of the transistors in electronic devices, i.e. it is the power used to perform calculations. Leakage power is the unwanted result of subthreshold current in the transistors and does not contribute to the microcontroller function. Fan power is becoming a more important contributor by the day to overall server power, and fan control policies can yield up to 10% energy savings at the server level [40].

Cooling power is one of the major contributors to the overall data center power budget, consuming over 30% of the overall electricity bill in typical data centers [41].

In the next subsections we propose a methodology to derive models for the power consumption of the servers and we evaluate the impact of cooling into the server power consumption.

4.3.1. Leakage power

Dynamic consumption has historically dominated the power budget. But when technology scales below the 100*nm* boundary, static consumption becomes much more significant, being around 30-50% [42] of the total power under nominal conditions. This issue is intensified by the influence of temperature on the leakage current behavior. Therefore, it is important to consider the strong impact of static power as well as its temperature dependence and the additional effects influencing their performance. In this section, we derive a leakage model for the static energy consumption of servers and we validate it with real measurements taken in the Intel Xeon machine of our case study.

The current that is generated in a MOS device due to leakage is the one shown in Equation 6.

$$I_{leak} = I_s \cdot e^{\frac{V_{GS}-V_{TH}}{nkT/q}} \cdot (1 - e^{\frac{Vds}{kT/q}}) \tag{6}$$

$$I_s = 2 \cdot n \cdot \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot \frac{kT^2}{q} \tag{7}$$

Research by Rabaey [43] shows that if $V_{DS} > 100mV$ the contribution of the second exponential is negligible, so the previous formula can be rewritten as in Equation 8:

$$I_{leak} = I_s \cdot e^{\frac{V_{GS}-V_{TH}}{nkT/q}} \tag{8}$$

where technology-dependent parameters can be grouped together in a constant (B) to obtain the formula in Equation 9:

$$I_{leak} = B \cdot T^2 \cdot e^{\frac{V_{GS}-V_{TH}}{nkT/q}} \tag{9}$$

Based on the leakage current equation, we can derive the leakage power for a particular machine $m \in \{1, \ldots, M\}$ (Equation 11). Because our goal is to fit a model for the leakage power, we expand the polynomial function into its Taylor second order series (Equation 12) to easily regress the function, where B_m and C_m define constants due to the manufacturing parameters of a server.

[1] http://pandaboard.org/content/resources/references

$$P_{leak,m} = I_{leak,m} \cdot V_{DD,m} \quad (10)$$

$$P_{leak,m} = B_m \cdot T_m^2 \cdot e^{\frac{V_{GS}-V_{TH}}{nkT/q}} \quad (11)$$

$$P_{leak,m} = B_m \cdot T_m + C_m \cdot T_m^2 \quad (12)$$

As temperature-dependent leakage cannot be measured separately from the dynamic power in a server, we again make use of the *Lookbusy* synthetic workload to be able to express dynamic power as linearly dependant with CPU utilization, and isolate the contributions of leakage. Equation 13 provides the formula for the total power consumption:

$$P_{serverm} = A_m \cdot u_{imk} + B_m \cdot T_m + C_m \cdot T_m^2 \quad (13)$$

where A_m is a constant that defines the parameters of a specific machine, u_{mk} is the utilization of a certain task and T_m is the CPU temperature.

Tests have been conducted gathering real data from the Intel Xeon server of the case study. During these tests, fan speed was constant, not interfering with leakage modeling. Total power consumption and CPU temperature have been collected via Intelligent Platform Management Interface (IPMI) tool[2] during the execution of lookbusy at different utilization levels ranging from 10% to 100%. We use MATLAB to fit our data to the energy model obtaining a maximum error of 8.18% and an average error of 2.29% in the fit.

Deriving a leakage power model is of utter importance, as it allows us to exploit the leakage-temperature trade-offs at the server level and exploit all the capabilities of resource selection and configuration.

4.3.2. Dynamic power

In order to derive a dynamic power model we use the information collected via the servers performance counters and IPMI during the execution of a workload. Performance counters are a set of special-purpose registers built into modern CPUs to store the counts of hardware-related events. Because they are integrated into the architecture, polling these counters has a negligible overhead in the performance of the workload being profiled. Modern servers come with a high number of performance counters that can be polled. State-of-the-art workload profiling techniques show that performance counters can be used as a good predictor of energy consumption [44]. Servers are also shipped with a large amount of sensors to collect temperature, fan speed or power consumption data. These data can be gathered via IPMI monitoring tools with negligible overhead.

Data has been gathered from the enterprise servers of the case study under two conditions: i) when executing one instance of SPSS and SPEC CPU 2006 benchmark suite and ii) when executing one instance of SPSS and SPEC CPU 2006 per core (i.e. fully utilizing the CPU) to evaluate possible contention of shared resources. All experiments have taken place in a data room at a constant server inlet temperature of 16°C to minimize the effects of temperature-dependent leakage on the modeling.

[2] http://ipmitool.sourceforge.net/

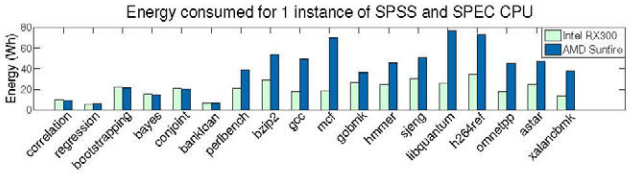

Figure 6: Energy for SPEC CPU INT 2006 benchmark in various servers

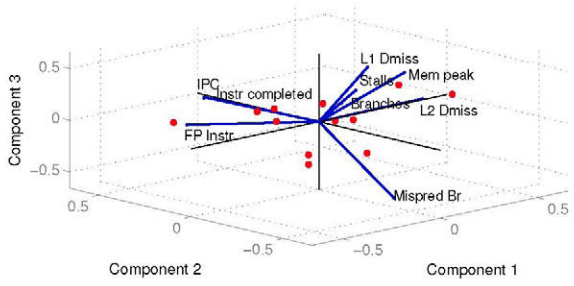

Figure 7: First 3 principal components coefficients for various performance counters

Figure 6 shows the energy consumption of each server when executing one instance of SPSS and SPEC CPU integer benchmarks. As can be seen, the most modern server (Intel Xeon server) outperforms the AMD in all SPEC CPU 2006 benchmarks. That does not happen for the SPSS benchmarks, where *boostrapping*, *bayes* and *bankloan* have a lower energy consumption in the AMD server. Moreover, depending on the SPEC benchmark, the difference is smaller or larger. These differences can be used to perform a better workload scheduling in terms of energy efficiency as shown in Section 6.1.

Intuitively, we explain these differences because of the different idle power of both servers (120W for the AMD server and 140W for the Intel server), as well as because of the different CPU and memory utilization of the benchmarks, i.e. the differences in the server architecture. However, in order to provide a deeper explanation of the energy behaviour of the workload, we must resort to the particular architecture of each server and understand the contention of shared resources that might be taking place. To do so from a high-level of abstraction perspective, we can derive a model for the dynamic power consumption for each one of the servers. We present the methodology and test our results for the Intel Xeon server.

In order to understand the behavior of the workload we collect information of the performance counters by means of PAPI during the execution of the workload in the Intel server. We collect 10 different parameters that are generally the most significant to explain the dynamic power [45]: IPC, Instructions completed, Cycles, Branches, Branch miss-predictions, Floating Point instructions, L1 misses, L2 misses and memory accesses. We apply Principal Component Analysis (PCA) to the data to determine considerable variations across the benchmark. The first 3 components together explain an 87% of the variance. Figure 7 shows the different counters in the principal components space.

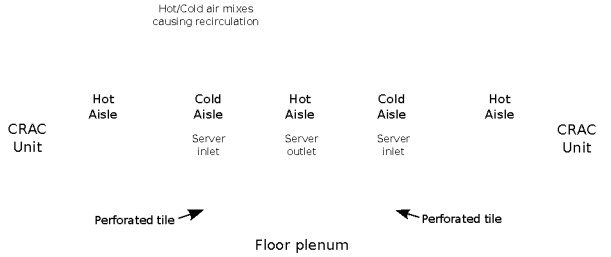

Figure 8: Data Center cooling scheme

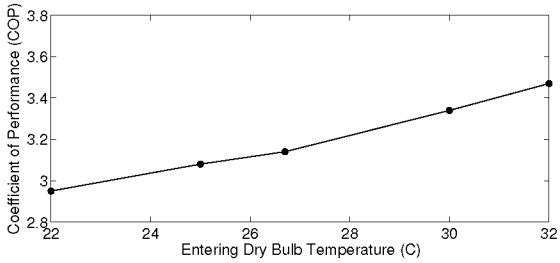

Figure 9: Evolution of the air-conditioning COP with room temperature

As can be seen, IPC and Instructions are closely related, as well as L1 data misses, Memory accesses, Stalls and L2 data misses. With these results, we are able to propose a model that takes into account parameters highly unrelated between themselves: IPC, Floating-Point instructions, Branch Miss-predictions and Resource Stalls. We use techniques to fit a model to these parameters in the way shown in Equation 14, where $\alpha, \beta, \gamma, \theta$ are the constants to be obtained with the regression.

$$P_{dynamic} = \alpha \cdot Stalls + \beta \cdot FP_{instr} + \gamma \cdot Br_{mispred} + \theta \cdot IPC \quad (14)$$

Once we have a model for the dynamic power of the different servers of our system, we can predict the energy consumption of each of them and validate it with our experimental results. For the Intel server and the tasks of the SPEC benchmark (which represent more than 70% of the total tasks executed in the data center according to our previous workload distribution and classification) we obtain a prediction in dynamic power with a maximum error of 13% and an average error of 7%. Given the dynamic power model and the execution time of our tasks, we can easily obtain the parameters for energy consumption per server and task. We apply these data in our fine-grain run-time allocation.

4.3.3. Cooling power

In a typical air-cooled data center room, servers are mounted in racks, arranged in alternate cold/hot aisles, with the server inlets facing cold air and the outlets creating hot aisles. The Computer Room Air Conditioning (CRAC) units pump cold air into the data room and extract the generated heat (see Figure 8). The efficiency of this cycle is generally measured by the *Coefficient of Performance* (COP). The COP is a dimensionless value defined as the ratio between the cooling energy produced by the air-conditioning units (i.e. the amount of heat removed) and the energy consumed by the cooling units (i.e. the amount of work to remove that heat), as shown in Equation 15.

$$COP_{MAX} = \frac{\text{output cooling energy}}{\text{input electrical energy}} \quad (15)$$

Higher values of the COP indicate a higher efficiency. The maximum theoretical COP for an air conditioning system is described by Carnot's theorem as in Equation 16:

$$COP_{MAX} = \frac{T_C}{T_H - T_C} \quad (16)$$

where T_C is the cold temperature, i.e. the temperature of the indoor space to be cooled and T_H is the hot temperature, i.e. the outdoor temperature (both temperatures in Celsius). As the difference between hot and cold air increases, the COP decreases, meaning that the air-conditioning is more efficient (consumes less power) when the temperature difference between the room and the outside is smaller.

The data room considered in our case study is equipped with a Daikin FTXS30 unit, with a nominal cooling capacity of 8.8kW and a nominal power consumption of 2.8KW. For an outdoor temperature of 35°C the theoretical COP curve obtained by using the manufacturer's technical datasheet [46] is shown in Figure 9.

This figure shows how as the room temperature and the heat exhaust temperature raise, approaching the outdoor temperature, the COP increases and, thus the cooling efficiency improves. According to this, one of the techniques to reduce the cooling power is to increase the COP by increasing the data room temperature. However, as we increase room temperature, CPU temperature increases and so does leakage power. Therefore, there is a tradeoff between the reduction in cooling power and the increase in server leakage power. Our hypothesis is that we can find and define two different working regions depending on the impact of ambient temperature in leakage power and thus in the total power consumption of enterprise servers. In this sense, we aim to prove that for the lower range of ambient temperatures, the impact of the temperature-dependant leakage is negligible, whereas for a higher temperature range leakage needs to be considered.

To prove our hypothesis, real power measurements are collected in our data room. We gather the cooling power of the air conditioning unit when the servers are fully utilized running tasks of the SPEC CPU 2006 benchmark. Figure 10 shows the power consumption for different air supply temperatures. Data is gathered for a whole day, as the power consumption exhibits a periodic behavior dependant on the outdoor temperature. The power consumption decreases from 23.17kWh per day at 18°C to 19.65kWh when set to 24°C, yielding a 15% in energy savings.

However, these data has to be shown together with the CPU temperature of the servers and their power consumption. Figure 11a shows the IT power consumption for the Sunfire V20z

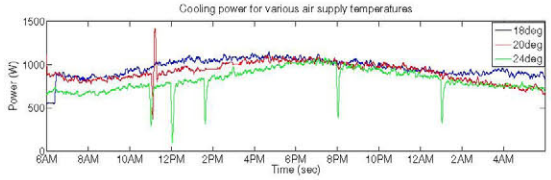

Figure 10: Cooling power for different air supply temperatures

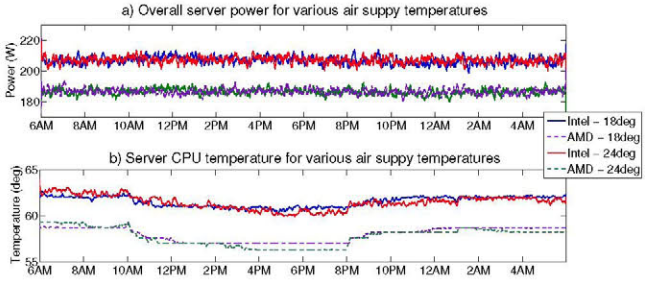

Figure 11: IT power and CPU temperature for fully utilized server at various air supply temperatures

server and the Intel Xeon that exhibit the higher CPU temperatures in our data room. These servers are the ones limiting the air-supply temperature, as we must ensure their safety operation. Figure 11b shows how the server power consumption is stable, meaning that for these CPU temperatures we are working in the region where temperature-dependant leakage power is negligible.

5. Global Resource Allocation techniques

In several traditional distributed Mobile Cloud Computing solutions, the sensor and coordinator nodes of the architecture either perform as much computation as possible (with the inherent penalty in battery lifetime) or forward all computation to the computer facility, even though coordinator nodes have enough computational capabilities to perform other tasks. These strategies do not consider the benefits in terms of energy savings that an efficient allocation of workload can provide.

Our global resource allocation proposes a coarse-grain workload assignment technique that aims to reduce the overall energy consumption of the architecture by optimizing the tradeoff between offloading computation to the data center facility and executing the calculation in the nodes. To do so, low-demanding tasks of the Data Dependant layer are executed in the coordinator nodes, instead of forwarding all the tasks to the data center, while medium and high demanding tasks are offloaded to the data center infrastructure. We leverage the concepts of previous work in this area [47] by improving the state-of-the-art allocation algorithms and refining the modelling and assignment for nodes with lower resources. The goal of the allocation policy is to provide energy savings in three different ways: (i) reducing the power consumption of the overall network by executing tasks in low-power nodes instead of in a cloud computing facility (ii), increasing throughput and overall performance by parallelly executing tasks in both data center

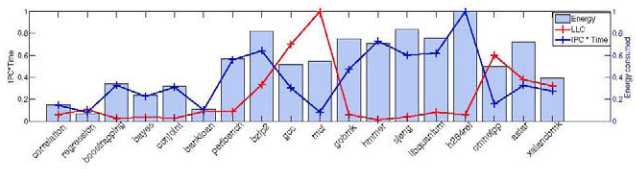

Figure 12: Correlation between $IPC * Time$ and LLC metric (left axis, lines) and Energy (right axis, bars)

and coordinator nodes, and (iii) reducing the energy consumption due to communication by decreasing the amount of data transmitted over the network.

The coarse-grain resource assignment allocates tasks between coordinators (smartphones in our case) and data center. Shimmer nodes are not considered in this assignment because they have very low resources and the executed applications are particularly optimized for the Shimmer architecture.

We propose a two-step methodology for our global resource management policy: (i) classifying tasks of the workload according to their computational demand in the three different classes previously presented in Section 3, and (ii) running a run-time distributed coarse-grain allocation algorithm to decide whether each task should be executed at the coordinator or forwarded to the data center to maximize energy efficiency across the network.

5.1. Task classification

In order to classify the tasks of the workload to be executed, the first time that a task appears in a job set, it is profiled during its execution in the data center. Task profiling is done without performance degradation by gathering information of performance counters and execution time of the application.

Our coarse-grain assignment policy does not need a really accurate metric for the absolute value of the dynamic power consumption of a task. Instead, we aim to obtain a good metric of the energy-performance tradeoff of executing a particular task in a particular type of node (i.e., server or coordinator) that allows us to classify that task . Previous work on this topic [48] shows that IPC and CPU utilization (or the combination of both) is usually a good predictor for power efficiency. However, this approach disregards the memory consumption, which is important in enterprise servers, and can be predicted via the Last Level Cache misses (LLC) [44].

We execute all the different tasks of the workload (i.e. the SPSS and SPEC tasks presented on Section 3) in one of the servers of the data center, the Intel Xeon RX300, and use PAPI [49] to collect performance counters. We also measure the overall energy consumption of the server when executing each task by polling the power sensors integrated in the server via IPMI. We only perform the profiling in the most modern server of the data center, as our goal is just to get a first rough idea of the computational demand of the tasks.

Figure 12 shows the correlation between energy and the $IPC * Time$ and LLC metrics. As can be seen, $IPC * Time$ metric follows the trend of energy consumption, except for some benchmarks such as *gcc*, *omnetpp* or *astar* where the

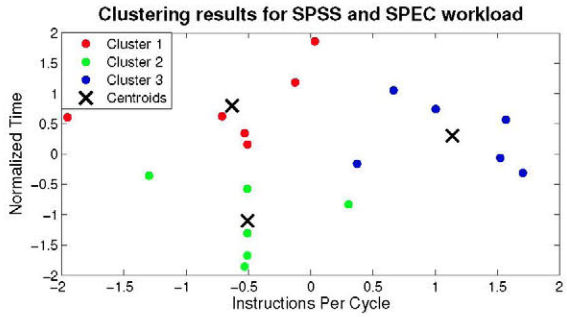

Figure 13: Clusters obtained in k-means classification for SPSS and SPEC

Cluster	Tasks
Low demanding	correlation, regression, bayes bankloan,omnetpp,xalancbmk
Medium demanding	bootstrapping, conjoint, gcc mcf, astar, gobmk
High demanding	perlbench, bzip2, hmmer sjeng, libquantum, h264ref

Table 5: Task classification for SPSS and SPEC tasks

$IPC{*}Time$ metric underestimates energy consumption. In these benchmarks, LLC is particularly high, meaning that they have a higher amount of memory accesses that have an impact in energy consumption.

In this case, as we search for an overall server energy predictor, we propose the usage of the metric $IPC * Time$.

Based on these results, we use IPC and execution time values to classify the different tasks of the workload using a k-means clustering. K-means algorithms need to know a-priori the number of clusters for classification. We use $k = 3$ number of clusters, and compare results with $k = 2$ and $k = 4$, getting the best results for $k = 3$. Figure 13 shows the clusters obtained whereas Table 5 details the task classification. We validate the clustering by checking whether low, medium and high energy consumption tasks are properly assigned to low, medium and high demand clusters. Our validation shows good results for the k-means clustering.

With these results for the task characterization, we can move on to the proposal of the run-time allocation algorithm.

5.2. *Run-time allocation algorithm*

The run-time allocation algorithm proposes to execute some of the low and medium-demanding tasks of the workload presented in Section 3 in the coordinator nodes instead of forwarding all computation to the data center. Each coordinator acts as a concentrator of the information sent by Shimmer nodes and thus, has all the data needed for the computation. If coordinators perform part of the tasks of the Data Dependent layer, the amount of computation in the data center is reduced and, thus, the facility consumes less energy. Moreover, if computations are performed at the coordinators, the amount of data to transmit to the data center is reduced, saving energy in the communication process. In this paper, however, we do not aim to estimate the energy consumption of the coordinator-datacenter communication. Instead, we focus on the benefits on the overall network that come from the reduction in energy due to computation.

When a new job set arrives each coordinator needs to compute whether that task should be executed or forwarded with its data dependencies to the data center. This means that the run-time allocation algorithm must run in a distributed way, i.e. each coordinator launches the algorithm for its particular task, but using the information provided by the data center. As these nodes are battery-operated, it would not be wise to waste their energy calculating the optimum assignment. Instead, we propose the usage of a fast and lightweight distributed allocation algorithm based on Satisfiability Modulo Theory (SMT) formulas.

SMT solvers are fast solvers that determine whether a certain formula can be satisfied. In our case, when a certain amount of low and medium-demanding tasks arrive to a coordinator, it needs to compute whether the workload satisfies certain conditions. We use an SMT solver to calculate which tasks of the workload satisfy those conditions and the amount of tasks that can be executed. Let us denote by $T_{node_i,j}$ the low and medium demanding Data Dependent tasks of a particular job set j that can be executed in a certain node; by $T_{data,j}$ all the Data Dependent tasks in a job set j and by N_{cores} the overall amount of computational cores available at the data center.

Each task t has a duration and consumes a certain amount of energy depending on whether it is executed at the data center or coordinator, noted by σ_{tp} and e_{tp} respectively. As this optimization does not manage the idle power consumption of the elements of the architecture with turn off policies, we assume that both coordinators and servers are always turned on. Because of this, we aim to optimize the energy variation for executing a certain task. For this reason, e_{tp} does not consider the idle power for neither coordinators or servers, i.e., it considers only the amount of energy spent over idle state. The conditions that the workload have to satisfy in order to be executed are proposed next:

$$\left(\sum_{t\in T_{node_i,j}} e_{tp}\cdot\sigma_{tp}\right)_{coord} \leq 0.1\cdot\left(\sum_{t\in T_{data,j}} e_{tp}\cdot\sigma_{tp}\right)_{datacenter} \quad (17)$$

$$\left(\sum_{t\in T_{node_i,j}} \sigma_{tp}\right)_{coord} \leq \alpha\cdot\left(\frac{\sum_{t\in T_{data,j}}\sigma_{tp}}{N_{cores}}\right)_{datacenter} \quad (18)$$

$$\left(\sum_{t\in T_{node_i,j}} e_{tp}\right)_{coord} \leq \beta_{max} \quad (19)$$

$$(20)$$

Equation 17 states that the Energy Delay Product (EDP) of the tasks executed in the coordinators must be at least an order of magnitude less than the EDP product for those same tasks if executed in the data center. EDP weights power against the square of execution time, and is a common metric to compare energy efficiency optimizations from both the data center level and the architectural point of view [50]. Equation 18 constraints

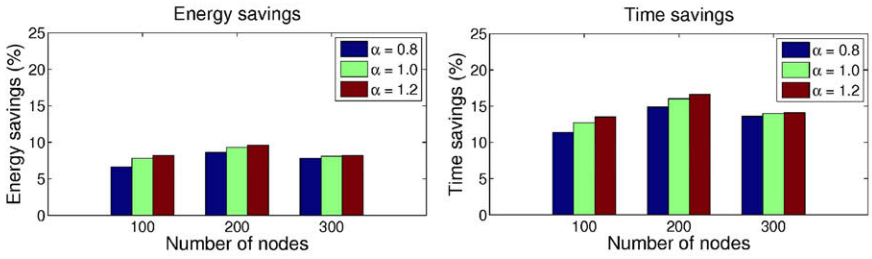

Figure 14: Percentage of energy and time savings for each number of nodes and $\alpha = \{0.8, 1.0, 1.2\}$ under the reference workload

the maximum time taken for the tasks to be executed to the overall time that it would take to execute data dependent tasks at the data center, i.e. it ensures a certain Quality of Service (QoS). This constraint can be adjusted through the parameter $\alpha = [0 \dots 1]$. Finally, Equation 19 constraints the maximum amount of battery used per job set in each coordinator to a maximum energy β_{max}.

In order to perform the run-time allocation, our SMT solver algorithm needs to know an estimation of the energy e_{tp} and duration σ_{tp} of each task t to be executed for each processor p. For this purpose we use the energy profiling results for the Intel Xeon machine and the Samsung Galaxy S2 coordinator of Section 4.

Each job set in our workload contains a 60% of tasks belonging to the Data Dependent layer which, in its turn, has a 70% of low-demanding tasks and a 30% of medium demanding tasks. We assume that each coordinators in the architecture generates the same amount of data in average, so that tasks are uniformly split between nodes. We also assume that not all coordinators might be available at all times for computation purposes, so we might have a different amount of coordinators (from 100 to 300). Depending on the workload profile (heavy, reference or light workload) and the amount of coordinators available, each coordinator executes a different amount of tasks.

Our SMT algorithm is implemented using the Yices SMT solver [3] that runs with negligible performance and energy overhead in the coordinator node, obtaining a solution in less than 1 second for each node. If the conditions to execute a task are satisfied, then the task is executed in the coordinator node. If not, it is off-loaded to the cloud infrastructure. For our case study we use a fixed value of $\beta_{max} = 300mWh$, which represents a 30% of the energy resources of the coordinator node Samsung Galaxy SII. We execute the workload for different parameters of $\alpha = \{0.8, 1, 1.2\}$ and calculate the average amount of tasks executed by the coordinator, the energy consumed by each coordinator in the system, and the energy saved at the data center. We use as a baseline for comparison the execution of all the workload in a data center with 160 cores (40 servers) of the Intel Xeon machine of the case study, without using any coordinator. We run the algorithm for the three different workload profiles with three different number of coordinators (100, 200 and 300) to compare performance. Figure 14 shows the percentage of dynamic energy and time savings compared to the execution of the reference workload in data center facility in 160 Intel cores.

The allocation of the whole workload at the data center facility (no coordinator nodes) consumes around 24kWh plus the idle energy of the servers, and the execution takes around 13h to complete. These 24kWh are the energy variation due to the workload execution. By using coordinators to execute part of the workload, we can obtain up to a 10% savings in energy variation and a 16% savings in execution time for the reference workload; while up to 24% energy savings for the heavy workload by using 300 coordinators can be achieved. The energy savings do not consider the savings obtained in idle power, which come from the reduction in execution time or occupancy that could lead to server turn-off policies specific to the data centers. This and other aspects are studied in Section 6.2. The absolute energy values for each workload profile and coordinator are summarized in Table 9 in Section 7.

6. Data Center horizontal optimizations

In this section we present the main optimizations proposed by the scheme depicted in Section 2.1. We order these optimizations by abstraction level in a top-down fashion (see Energy Optimizations in Figure 2), focusing our analysis on the next optimization layers:

- Data Center resource management policies
- Energy-aware virtualization techniques

Even though optimizations are presented as different horizontal approaches, it must be kept in mind that all these layers are interconnected. Some of the optimizations presented are static, however, most of them are designed to work on runtime during the complete lifetime of the system, adapting and reconfiguring their behavior according to the specific needs of the system. In this way, we can assure that the energy footprint of the computational paradigm is optimized to a maximum degree.

6.1. Data Center resource management

Resource management is a well known concept in the Data Center world and refers to the efficient and effective deployment of computational resources of the facility where they are needed. Resource management techniques are used to allocate

[3] http://yices.csl.sri.com

in a spatio-temporal way the workload to be executed in the data center, optimizing a particular goal. Traditionally, these techniques have focused on maximizing performance by assigning tasks to computational resources in the most efficient way. However, the increasing energy demand of Data Center facilities has shifted the optimization goals towards maximizing energy efficiency. Our work leverages this concept by proposing energy-aware resource management techniques at different levels of abstraction of the proposed computational paradigm. In the previous section we proposed a workload assignment that splits the computation between the coordinator nodes and the data center. In this section we aim to propose a finer-grain workload assignment policy that distributes the computation at run-time inside the data center, taking advantage of (i) the knowledge about the energy behavior of the applications to execute, and (ii) the resource heterogeneity of the data center.

To do so, we first propose a static optimization that aims to select the most appropriate resources of the data center, i.e. the best machines, to execute the workload. Secondly, we perform a run-time allocation that minimizes the energy consumption of the cloud facility.

6.1.1. Data Center server selection

Until now, we had supposed that all the computations of the case study were performed in a homogeneous cluster with 160 cores belonging to Intel Xeon machines. However, even though the Intel servers are the most modern ones, in Section 4 we have shown that for some tasks, the AMD server outperforms the Intel in terms of energy efficiency. Our claim is that we can find a set of heterogeneous servers that outperform the homogeneous scenario, i.e. executes the workload consuming less energy without a penalty in execution time.

The optimization algorithm is based on previous work [51] and can be defined as follows. Let us denote by M a set of machines, by P a set of processors and by T a set of tasks that must be executed. Each machine m consumes a certain power in idle state π_m. Each processor p belongs to one machine m, denoted as p_m. Every task t has a duration and consumes a certain amount of energy depending on the target processor, σ_{tp} and e_{tp} respectively. The minimization function is the one shown in Equation 21:

$$\text{Minimize} \left\{ \sum_{t \in T, p \in P} k_{tp} \cdot e_{tp} + \sum_{m \in M} \pi_m \cdot \tau^{max} \right\} \tag{21}$$

where k_{tp} is a binary variable that is set to 1 if the task t is executed in processor p. τ^{max} is the time instant at which all the tasks have been executed. In this case we aim to obtain the best server set for a limited number of CPU cores which, in our case, is 160 cores. The optimizer takes as input a bunch of machines of each type, and it will provide as output the selection of machines m that should be used to execute the workload. Thus, the constraints that the proposed model must fulfill are the following:

Workload profile	Coordinators	Server selection
Heavy	100	35 Intel + 5 AMD
	200	36 Intel + 4 AMD
	300	37 Intel + 3 AMD
Reference	100	36 Intel + 4 AMD
	200	35 Intel + 5 AMD
	300	36 Intel + 4 AMD
Light	100	31 Intel + 9 AMD
	200	31 Intel + 9 AMD
	300	35 Intel + 5 AMD

Table 6: Selected heterogeneous cluster configuration for each workload

$$\sum_{m \in M} c_m \cdot \psi_m \leq \Psi \tag{22}$$

$$\sum_{t \in T} k_{tp_m} \cdot \sigma_{tp_m} \leq \tau_m^{max} \quad \begin{cases} m = 1, \dots M \\ p_m = 1, \dots P_m \end{cases} \tag{23}$$

$$\tau_m^{max} \leq \tau^{max}, \qquad m = 1, \dots M \tag{24}$$

$$\sum_{p \in P} k_{tp} = 1, \qquad t = 1, \dots T \tag{25}$$

where c_m is the amount of cores of a certain machine m, ψ_m is a binary variable that is set to 1 if a certain machine is used and Ψ is the maximum amount of cores available for computation. Constraint 22 ensures that the total number of cores used is not higher than the maximum. Constraints 23 and 24 ensure that all the tasks are executed within a maximum time, whereas constraint 25 ensures that all the tasks will be executed once and just once in one processor.

The algorithm has been coded using ILOG CPLEX optimization suite. The reason for choosing CPLEX tool is that it provides optimization libraries to solve Mixed Integer Linear Programming (MILP) problems together with a very complete API for Java, C++, Python and .NET that eases the integration with other tools. We feed the algorithms with one job set per workloads to be executed, and the algorithm solves which is the optimum number of servers to be used, and provides an optimum assignment for that job set. Table 6 summarizes the results for every workload arriving to the data center.

As can be seen, the optimizer always chooses an heterogeneous data center containing a small amount of AMD servers that are used to perform the tasks in which they outperform the Intel servers in terms of energy efficiency. Those tasks are the ones previously shown in Section 4 on Figure 6. All the combinations use 160 cores and the amount of required AMD servers is highly dependant on the workload to be executed, as well on the duration of the execution, as the AMD servers present a lower static power than the Intel servers.

In the next section we use these results to assign the whole workload profile and provide results on the energy and time savings when compared to the homogeneous data center scenario.

6.1.2. Run-time workload assignment

The dynamic run-time allocation of the tasks, performed by the resource manager, aims at minimizing the energy consump-

Workload profile	Number of tasks	Energy consumption (kWh)			Execution time(h)		
		AMD	Intel	Intel + AMD	AMD	Intel	Intel + AMD
100 nodes, Heavy	8559	127.1	67.46	63.21	16.3	9.23	8.7
200 nodes, Reference	3765	61.7	34.12	31.89	7.8	4.7	4.5
300 nodes, Light	1961	37.31	28.02	27.6	4.8	4.3	4.3

Table 7: Energy consumption and execution time comparison between SLURM allocation and optimized allocation for various workload

tion of the assignment by placing each task where it wastes the minimum energy in a spatio-temporal way. The minimization objective is the same than the one in Equation 21, but the optimization constraints and the data fed to the optimizer are different. In this case, instead of running just one job set, we optimize the whole workload.

Constraint 22, which was use to fix the amount of cores to select in the data center, is no longer needed as we use the optimum data center found in the static optimization. Instead of constraint 23, we use 26 to take into account tasks that were scheduled in a previous job set but have not yet finished their execution. γ_{pm} is a time offset that represents the amount of time that a processor $p = 1, \ldots, P$ is occupied executing previous tasks when the new job set arrives and has to be taken into account when computing maximum execution time per server.

$$\sum_{t \in T} k_{tp_m} \cdot \sigma_{tp_m} + \gamma_{pm} \leq \tau_m^{max} \quad \begin{cases} m = 1, \ldots M \\ p_m = 1, \ldots P_m \end{cases} \tag{26}$$

Again, we use CPLEX with its C++ API to launch the optimization. Because the workload is divided in job sets, the optimization will be executed each time that a new job set arrives to the resource manager, for a limited period of time, in order to assign tasks to processors. This optimization procedure mainly improves the total energy variation (the aforementioned e_{tp}). The goal of this optimization is not really to reduce drastically the total execution time (as this time is inherently reduced by the static optimization), but just to ensure that it does not exceed a maximum.

In order to compute the energy obtained by our optimization, we compare our solution to the one provided by the SLURM resource manager [52]. SLURM is an open source production-ready software tools used in many data centers to allocate the workload to servers. SLURM uses a round-robin policy to assign tasks to nodes. We use an open source SLURM simulator [53] developed by the Barcelona Supercomputing Centre to simulate the workload assignment with SLURM default allocation policy and our algorithm. Figure 15 provides an insight on the workload coming into the data center for three different scenarios and how that workload is scheduled by SLURM in an Intel 160-core homogeneous cluster. As can be seen, under the heavy workload profile with only 100 coordinator nodes offloading computation, the number of jobs running is always close to the maximum amount of cores, i.e. the system occupancy is very high. Also, there is a huge number of jobs waiting to be scheduled. However, for the light workload profile with 300 coordinators, the system has a low occupancy for long periods of time.

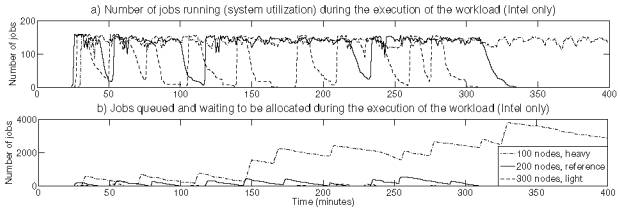

Figure 15: Running and waiting jobs in Intel only scenario for various loads with different number of coordinator nodes

Table 7 shows the comparison in energy and execution time for the execution of the three workloads shown in Figure 15, between the AMD homogeneous data center, the Intel homogeneous data center, and the Intel + AMD heterogeneous solution. As can be seen, the heterogeneous solution clearly outperforms the homogeneous AMD data center. However, our purpose is not to outperform the AMD homogeneous scenario, as those servers are clearly older. Our goal is to prove how by means of application-awareness, i.e. knowing the energy profile of our workload, we can save energy. Moreover, our solution always saves energy without performance degradation when compared to the Intel cluster. This is accomplished not only because of the usage of a heterogeneous data center, but also because we schedule the workload in an optimum way thanks to the a priori knowledge of its behaviour in terms of energy and execution time. The energy savings obtained range from 1.4% for the light workload with 300 coordinators case to a 7.5% for a reference workload with 100 coordinators. All energy saving percentages are shown in Table 9 in Section 7.

6.2. Virtualization techniques

As shown previously, smart resource allocation in physical hosts is crucial to leverage computing resources more efficiently in order to reduce energy consumption. However, in our previous study, we were considering the execution of the workload on physical servers that did not provide virtualization. This has a small impact for the heavy workload profile in which all processors are highly utilized. However, physical servers are under utilized for lighter workload profiles in which utilization of the cluster drops.

The novel paradigm of cloud computing uses the concepts of virtualization and consolidation to offer new services in a platform that achieves a more efficient infrastructure. Virtualization leverages the management of the data center as a pool of resources, allowing a single node to accommodate simultaneously various Virtual Machines (VM) that can be dynamically started and stopped according to the system workload and that share physical resources.

Consolidation uses virtualization to share resources and reduces energy consumption by increasing resource utilization. During periods of lower utilization, consolidation techniques can be used to attenuate power consumption by reducing the active server set, increasing resource utilization, reducing static power consumption and increasing energy efficiency. Handling the operating server set in order to turn off certain servers is a specially useful technique and provides very good results when considering the leakage power as well as the dynamic power consumed by servers. When implementing this type of policies it is important to consider the characterization of data center usage. The demand for resources reaching the data center is variable and usually follows seasonal patterns depending on the time of the day or certain periods of the year. In addition, the data center must be prepared to support peak demands. Also, a certain Quality of Service (QoS) must be satisfied in terms of availability, execution time and response time constraints.

In order to apply virtualization and consolidation techniques to our scenario, we next present the experiments and system definition and an analysis of the results obtained.

6.2.1. System definition

In our system, the data center facility executes the low and medium demanding workload not computed at the coordinator nodes as well as high-demanding applications. Depending on the workload profile (heavy, reference or light) the amount of tasks to be executed varies. Moreover, to procure an efficient placement, it is necessary to consider both the energy consumption and the resource needs of every task of the workload as well as the availability of resources provided by each scenario. The physical needs of computing resources consumed by the applications are required in order to execute them under a required QoS and provide the minimum active server set. For our experiments we define a time QoS that constraints the maximum execution task to a 33% more than its duration under optimal conditions.

To show the benefits of virtualization, our experimental setup involves the comparison of two different scenarios: (i) the previous non-virtualized Intel data center and (ii) the same Intel cluster confirming a virtualized infrastructure. The virtual machines in this scenario also run CentOS under a KVM hypervisor.

Three experimental setups have been proposed to test energy consumption in both scenarios.

- Low-demanding tasks: consists on the parallel execution of 8 *correlation* tasks, which are low-demanding tasks in terms of both CPU and memory utilization and belong to the SPSS benchmark. The test is performed sequentially in order to model different incoming requests during a fixed time determined by the execution of a medium or high demanding task belonging to the SPEC benchmark.

- Mixed low-demanding and CPU intensive tasks: this scenario combines the parallel execution of 6 *correlation* tasks together with a CPU intensive benchmark *perlbench* from the SPEC benchmark suite.

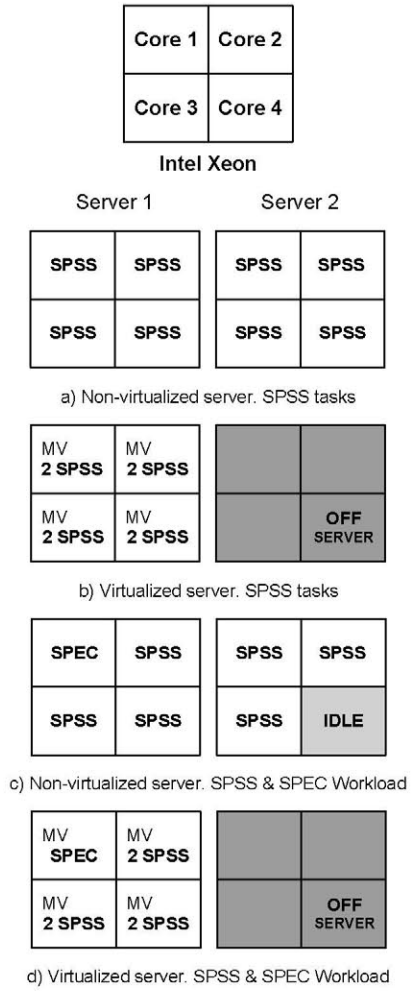

Figure 16: Workload allocation for the different tests.

- Mixed low-demanding and memory intensive tasks: consists on the parallel execution of 6 *correlation* tasks combined with a memory intensive tasks *mcf* from the SPEC benchmark suite.

As the execution of the *correlation* task consumes less than 50% of a CPU core we can use virtualization to consolidate two of these processes in the same VM, in order to use physical resources more efficiently. In the case of the non-virtualized data center we can only run four instances of SPSS simultaneously, one per core, in order to meet the time QoS constraint. However, by allocating one VM per core in a virtualized scenario we could execute 8 SPSS processes simultaneously. On the other hand, intensive applications such as *mcf* and *perlbench* fully utilize the CPU. Both virtualized and non-virtualized scenarios can only perform one of these tests per core without performance degradation.

Figure 16 shows how the different experimental sets can be allocated to fit the minimum operating server set on the data center in each scenario. Reducing the active set of physical machines by turning off idle servers decreases the static con-

Test Workload	Power Server 1 (W) non-virt.	virt.	Power Server 2 (W) non-virt.	virt.	Runtime (s) non-virt.	virt.	Total Energy (Wh) non-virt.	virt.	Energy Savings (%)
Low-demanding (8 correlation)	160.1	171.3	160.1	0.0	483.1	512.6	42.96	24.39	43.21
Low-demanding & CPU intensive (6 correlation + 1 perlbench)	168.0	174.3	155.1	0.0	479.5	502.4	43.03	24.32	43.46
Low-demanding & Memory intensive (6 correlation + 1 mcf)	169.7	173.7	155.1	0.0	355.2	467.2	32.04	22.54	29.66

Table 8: Workload average results for the different experimental scenarios

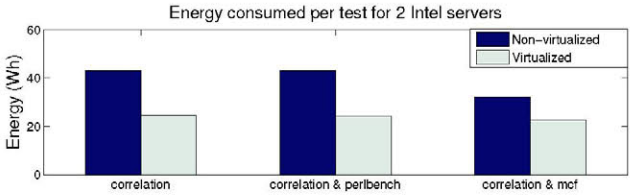

Figure 17: Workload allocation for the different tests.

sumption, which is one of the main contributions to power consumption and particularly relevant for light workload profiles. To evaluate the differences between virtualized and non-virtualized infrastructures we have defined a set of experiments. The purpose of these tests is to compare the energy efficiency obtained by virtualization and consolidation, taking into account the modification of the operating server set.

Tests have been performed in two Intel servers gathering the overall server power consumption during the execution of the workload as well as execution time. The energy savings reached 43.46% due to the reduction of the operating server set offered by the virtualized data center, turning off idle servers, and are presented in Table 8, where they are compared with the case of the non-virtualized scenario.

6.2.2. Results analysis

As can be seen in Table 8, the virtualized scenario presents a power consumption overhead due to the execution of VMs and the additional workload run in parallel for Server 1. The workload consolidation in this scenario also adds a time overhead due to resource sharing. The execution of Low-demanding & Memory intensive tasks experiences this effect to a greater extent due to its high memory demand that results in higher memory contention. Increasing the execution time of the tests rises energy consumption. However, these overheads are compensated by the reduction of the active server set when switching off Server 2 without degrading QoS. As seen in Figure 17, the data collected from the three workload tests show better results for the virtualized scenario achieving considerable energy savings of up to 43.46%.

Virtualization can be applied to reduce the overall power consumption of the workload, specially for light workload profiles of tasks exhibiting low computational demands. However, in order to be able to exploit all the potential of virtualization for our application, we need to reduce the operating server set when the cluster utilization drops during long periods.

7. Multi-layer integration

7.1. Overall energy savings

The goal of this section is to provide an insight on how the models and optimizations developed in this paper can be vertically integrated and applied together from a multi-layer perspective.

To accomplish this objective we first present a summary of the energy savings obtained for the global resource allocation technique developed in Section 5 and the Data Center resource management policies in Section 6, for each of the workload profiles and for a different amount of coordinators. In Table 9 Global resource allocation savings are referred to savings on the dynamic energy consumption only of the data center, whereas the DC policies offer the amount of savings for the Data Center only. In order to obtain an estimation of the impact of each optimization on the overall energy savings, we must first obtain the baseline energy consumption for each kind of workload without any optimization, and then apply the optimizations one after the other.

Table 10 shows the total energy consumption in kWh for the baseline case of not applying any optimization (first row of the table, i.e. the "No optimization" row) and when applying each optimization on top of the previous one. These values show the energy consumed for the whole architecture, i.e. coordinator nodes plus data center IT power plus data center cooling power. Percentages show the amount of energy savings when compared to the baseline case. These data has been obtained by simulating all the workload profiles by means of the SLURM simulator.

The second row of the table calculates the impact of offloading computation to the coordinator nodes. In Section 5 we were presenting the dynamic power savings at the data center level. Here we use SLURM simulator to re-run the workload arriving to the data center for a different amount of coordinators, so that we can see the impact in execution time and static energy consumption. As can be seen, the impact of offloading techniques is huge, and is highly dependant on the workload profile. The third row adds the impact of increasing the air-supply temperature of the cooling system from 18°C to 24°C in the four air conditioning units needed to cool the 40 machines of the experimental set-up. Finally, the fourth row adds the impact of the data center resource selection and optimum workload assignment policies. As we are showing the results for around 13 hours of computation in the worst-case scenario, we do not integrate the results for virtualization techniques in Table 10, as the

	High workload			*Medium workload*			*Low workload*		
	100	200	300	100	200	300	100	200	300
Global	7.8%	9.3%	8.1%	4.9%	3.4%	6.4%	3.6%	11.6%	24.0%
DC policies	6.3%	6.0%	5.7%	7.5%	6.5%	5.2%	3.1%	2.3%	1.4%

Table 9: Summary of savings for each optimization

	Heavy workload			*Reference workload*			*Light workload*		
	100	200	300	100	200	300	100	200	300
No optimization	153.6	153.6	153.6	101.0	101.0	101.0	49.0	49.0	49.0
Global allocation	103.1	95.3	93.0	53.4	52.5	51.3	45.2	47.7	44.6
	(32.8%)	(37.9%)	(39.4%)	(47.1%)	(48.0%)	(49.2%)	(7.7%)	(2.6%)	(9.3%)
Cooling	97.7	90.2	88.1	50.6	49.7	48.5	42.7	42.2	42.1
	(36.4%)	(41.2%)	(42.6%)	(49.9%)	(50.7%)	(51.9%)	(12.8%)	(13.9%)	(14.1%)
DC Allocation	91.8	84.9	83.0	46.9	46.6	46.1	41.9	41.7	41.7
	(40.2%)	(44.7%)	(45.9%)	(53.6%)	(53.9%)	(54.3%)	(14.5%)	(14.9%)	(14.9%)

Table 10: Overall energy savings (in kWh and percentage) for the whole architecture when integrating all optimizations

utilization does not drop long enough periods to propose the reduction of the operating server set by means of turn-off policies. However, the virtualization techniques proposed in Section 6.2 would be extremely useful for the drops in demand that cloud infrastructures experience in this kind of applications during in some periods.

7.2. New challenges

The research work presented in this paper has open new challenges, profusely explored by the authors, that represent a novel and evolved conception of the distributed and high-performance computing paradigm. Along this paper, we have tackled the following topics:

- the concept of *heterogeneity* has been considered at different abstraction levels (horizontal heterogeneity among the server architectures of the data center, and vertical heterogeneity between the node-level and the data center-level architectures). This concept has been proved to provide further opportunities for energy-optimization (thanks to the workload distribution mechanisms), but it also encourages the seeking of global-optimization techniques that consider the heterogeneity of the system since the application conception.
- the conceived optimization techniques take into account the *dynamism of the scenario*, where variable workloads and tasks arrive to the computing platform and a varying number of processing nodes can be available for processing or ready to feed new data.
- the constraints imposed by the *Ubiquitous Computing* model have been exposed to be determinant on the architecture of the computing paradigm. Not only a set-up of wearable processing and sensing nodes is required, but also an all-over access to the computing services provided by the Cloud.
- the need of efficient energy-saving techniques in e-Health application scenarios has driven the conception of a new computing paradigm where the design of the architecture is pushed by the energy consumption of such application. Only with such *application-driven* design style, the energy footprint of the whole computing scheme can be reduced, while the reliability and performance requirements are still satisfied.

8. Conclusions

Home assisted living reduces sanitary costs by prevention of potential diseases, provides early signals of health decline and advices for appropriate actions in daily life, and allows complex epidemiologic analysis that improve prevention and efficacy of treatments. However, energy consumption is one of the major concerns for the adoption of population-wide health monitoring systems, but energy efficiency cannot be added as an afterthought.

In this paper, we have presented a novel concept of the computing paradigm that combines the deployment of population-wide Wireless Body Sensor Networks, wearable computing devices, high-performance computing data-centers, and services delivered by Cloud computing. Moreover, we propose an architecture driven by energy concerns and aimed at optimizing energy consumption globally.

This paper considers, for the first time, energy as a first-class requirement, taking it into account during the whole development cycle, from design to implementation. The novel strategies presented in the experimental work focus on every abstraction layer, and obtain promising results for a realistic scenario that depicts the cardiovascular tracking and analysis of a broad population.

We believe that the computing paradigm presented in this work, as well as the evolved methodology for energy reduction, deals with many and important challenges, often forgotten in the current related literature.

Acknowledgement

Research by Marina Zapater has been partly supported by a PICATA predoctoral fellowship of the Moncloa Campus of International Excellence (UCM-UPM). This work has been partially supported by the Spanish Ministry of Economy and Competitiveness, under contracts TEC2012-33892 and IPT-2012-1041-430000, and INCOTEC. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa).

References

[1] Grand challenges for engineering.
URL http://www.engineeringchallenges.org

[2] A. N. Khan, M. M. Kiah, S. U. Khan, S. A. Madani, Towards secure mobile cloud computing: A survey, Future Generation Computer Systems (29) (2013) 1278–1299.

[3] N. Fernando, S. W. Loke, W. Rahayu, Mobile cloud computing: A survey, Future Generation Computer Systems 29 (1) (2013) 84–106. doi:10.1016/j.future.2012.05.023.

[4] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, M. A. Hossain, A survey on sensor-cloud: Architecture, applications, and approaches, International Journal of Distributed Sensor Networks 2013 (2013) 18.

[5] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, R. Buyya, An autonomic cloud environment for hosting ecg data analysis services, Future Generation Computer Systems 28 (1) (2012) 147 – 154.

[6] V. Jones, A. Halteren, I. Widya, N. Dokovsky, G. Koprinkov, R. Bults, D. Konstantas, R. Herzog, Mobihealth: Mobile health services based on body area networks, in: R. Istepanian, S. Laxminarayan, C. Pattichis (Eds.), M-Health, Topics in Biomedical Engineering, Springer US, 2006, pp. 219–236. doi:10.1007/0-387-26559-7-16.

[7] I. Beretta, F. Rincon, N. Khaled, P. R. Grassi, V. Rana, D. Atienza, D. Sciuto, Model-based design for wireless body sensor network nodes, in: Test Workshop (LATW), 2012 13th Latin American, 2012, pp. 1 –6.

[8] B. Otal, L. Alonso, C. Verikoukis, Towards energy saving wireless body sensor networks in health care systems, in: Communications Workshops (ICC), 2010 IEEE International Conference on, 2010, pp. 1–5.

[9] B. Otal, L. Alonso, C. Verikoukis, Highly reliable energy-saving mac for wireless body sensor networks in healthcare systems, Selected Areas in Communications, IEEE Journal on 27 (4) (2009) 553–565.

[10] A. Fapojuwo, C. Tse, F. Lau, Energy consumption in wireless sensor networks under varying sensor node traffic, in: Wireless Communications and Networking Conference (WCNC), 2010 IEEE, 2010, pp. 1–6.

[11] H. Mamaghanian, N. Khaled, D. Atienza, P. Vandergheynst, Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes, IEEE Transactions on Biomedical Engineering (TBME) 58 (2011) 2456–2466.

[12] D. Barbagallo, E. Di Nitto, D. J. Dubois, R. Mirandola, A bio-inspired algorithm for energy optimization in a self-organizing data center, in: Proceedings of the First international conference on Self-organizing architectures, SOAR'09, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 127–151.

[13] C. O. Diaz, M. Guzek, J. E. Pecero, P. Bouvry, S. U. Khan, Scalable and energy-efficient scheduling techniques for large-scale systems, in: Proceedings of the 11th IEEE International Conference on Computer and Information Technology, 2011, pp. 641–647.

[14] L. Wang, S. U. Khan, Review of performance metrics for green data centers: a taxonomy study, The Journal of Supercomputing 63 (3) (2013) 639–656.

[15] I. Goiri, J. L. Berral, J. O. Fitó, F. Julià, R. Nou, J. Guitart, R. Gavaldà, J. Torres, Energy-efficient and multifaceted resource management for profit-driven virtualized data centers, Future Generation Computer Systems 28 (5) (2012) 718–731.

[16] D. Kliazovich, P. Bouvry, S. U. Khan, Dens: data center energy-efficient network-aware scheduling, Cluster Computing 16 (1) (2013) 65–75.

[17] P. Lindberg, J. Leingang, D. Lysaker, S. U. Khan, J. Li, Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems, The Journal of Supercomputing 59 (1) (2012) 323–360.

[18] D. M. Quan, F. Mezza, D. Sannelli, R. Giafreda, T-alloc: A practical energy efficient resource allocation algorithm for traditional data centers, Future Generation Computer Systems 28 (5) (2012) 791–800.

[19] C. O. Diaz, M. Guzek, J. E. Pecero, G. Danoy, P. Bouvry, S. U. Khan, Energy-aware fast scheduling heuristics in heterogeneous computing systems., in: W. W. Smari, J. P. McIntire (Eds.), HPCS, IEEE, 2011, pp. 478–484.

[20] S. U. Khan, A self-adaptive weighted sum technique for the joint optimization of performance and power consumption in data centers., in: J. H. Graham, A. Skjellum (Eds.), ISCA PDCCS, ISCA, 2009, pp. 13–18.

[21] K. Kumar, Y.-H. Lu, Cloud computing for mobile users: Can offloading computation save energy?, IEEE Computer 43 (4) (2010) 51–56.

[22] K. Greene, Cloud-powered GPS chip slashes smartphone power consumption, www.technologyreview.com/news/509176/cloud-poweredgps-chip-slashes-smartphone-powerconsumption (2012).

[23] M. Vallejo, J. Recas, J. L. Ayala, Channel analysis and dynamic adaptation for energy-efficient wbsns, in: J. Bravo, D. López-de Ipiña, F. Moya (Eds.), Ubiquitous Computing and Ambient Intelligence, Vol. 7656 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 42–49.

[24] J. Andrus, C. Dall, A. V. Hof, O. Laadan, J. Nieh, Cells: a virtual mobile smartphone architecture, in: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, SOSP '11, ACM, New York, NY, USA, 2011, pp. 173–187. doi:10.1145/2043556.2043574.

[25] F. Rincón, J. Recas, N. Khaled, D. Atienza, Development and evaluation of multilead wavelet-based ecg delineation algorithms for embedded wireless sensor nodes, Information Technology in Biomedicine, IEEE Transactions on 15 (6) (2011) 854 –863.

[26] C. Thompson, D. C. Schmidt, H. A. Turner, J. White, Analyzing mobile application software power consumption via model-driven engineering., in: C. Benavente-Peces, J. Filipe (Eds.), PECCS, SciTePress, 2011, pp. 101–113.

[27] F. X. Lin, Z. Wang, R. LiKamWa, L. Zhong, Reflex: using low-power processors in smartphones without knowing them, in: Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS XVII, ACM, New York, NY, USA, 2012, pp. 13–24. doi:10.1145/2150976.2150979.

[28] A. Artes, J. L. Ayala, F. Catthoor, Power impact of loop buffer schemes for biomedical wireless sensor nodes, Sensors 12 (11) (2012) 15088–15118.

[29] J. R. Villarreal, R. Lysecky, S. Cotterell, F. Vahid, A Study on the Loop Behavior of Embedded Programs, Tech. Rep. UCR–CSE–01–03, University of California, Riverside, Riverside, CA, USA (2001).

[30] R. Duan, M. Bi, C. Gniady, Exploring memory energy optimizations in smartphones, 2012 International Green Computing Conference (IGCC) 0 (2011) 1–8.

[31] S. Lee, J. Kim, Using dynamic voltage scaling for energy-efficient flash-based storage devices, in: SoC Design Conference (ISOCC), 2010 International, 2010, pp. 63 –66. doi:10.1109/SOCDC.2010.5682971.

[32] E. Le Sueur, G. Heiser, Dynamic voltage and frequency scaling: the laws of diminishing returns, in: Proceedings of the 2010 international conference on Power aware computing and systems, HotPower'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 1–8.

[33] A. Artés, J. L. Ayala, J. Huisken, F. Catthoor, Survey of low-energy techniques for instruction memory organisations in embedded systems, Signal Processing Systems 70 (1) (2012) 1–19.

[34] D. Meisner, T. F. Wenisch, Stochastic queuing simulation for data center workloads, in: EXERT, 2010.

[35] A. Berl, E. Gelenbe, M. D. Girolamo, G. Giuliani, H. D. Meer, M. Q. Dang, K. Pentikousis, Energy-efficient cloud computing, The Computer Journal Incorporating Special Issue: Architecture/OS Support for Embedded Multi-Core Systems.

[36] SPEC CPU Subcommittee and John L. Henning, SPEC CPU 2006 benchmark descriptions, http://www.spec.org/cpu2006/.

[37] Centro de Supercomputación y Visualización de Madrid (CeSViMa), http://www.cesvima.upm.es.

[38] A. Carroll, G. Heiser, An analysis of power consumption in a smartphone,

in: Proceedings of the 2010 USENIX conference on USENIX annual technical conference, USENIX'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 21–21.

[39] J. Koomey, Growth in data center electricity use 2005 to 2010, Tech. rep., Analytics Press, Oakland, CA (2011).

[40] M. Zapater, J. L. Ayala, J. M. Moya, K. Vaidyanathan, K. Gross, A. K. Coskun, Leakage and temperature aware server control for improving energy efficiency in data centers, in: Design Automation Test Conference in Europe, DATE'13, DATE '13, 2013.

[41] T. Breen, E. Walsh, J. Punch, A. Shah, C. Bash, From chip to cooling tower data center modeling: Part i influence of server inlet temperature and temperature rise across cabinet, in: Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2010 12th IEEE Intersociety Conference on, 2010, pp. 1–10.

[42] S. Narendra, A. Chandrakasan, Leakage in Nanometer CMOS Technologies, Integrated Circuits and Systems, Springer, 2010.

[43] J. Rabaey, Low Power Design Essentials, Engineering (Springer-11647), Springer, 2009.

[44] W. Bircher, L. John, Complete system power estimation using processor performance events, Computers, IEEE Transactions on 61 (4) (2012) 563–577. doi:10.1109/TC.2011.47.

[45] C. Hankendi, A. Coskun, Reducing the energy cost of computing through efficient co-scheduling of parallel workloads, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2012, 2012, pp. 994 –999.

[46] Daikin AC (Americas), Inc., Engineering data split, ftxs-l series, `http://www.daikinac.com/content/resources/manuals/engineering-manuals/` (2010).

[47] M. Zapater, C. Sanchez, J. L. Ayala, J. M. Moya, J. L. Risco-Martin, Ubiquitous green computing techniques for high demand applications in smart environments, Sensors 12 (8) (2012) 10659–10677.

[48] C. Hankendi, A. Coskun, Adaptive energy-efficient resource sharing for multi-threaded workloads in virtualized systems, in: International Workshop on Computing in Heterogeneous, Autonomous 'N' Goal-oriented Environments, CHANGE, 2012, 2012.

[49] P. J. Mucci, S. Browne, C. Deane, G. Ho, Papi: A portable interface to hardware performance counters, in: In Proceedings of the Department of Defense HPCMP Users Group Conference, 1999, pp. 7–10.

[50] S. Rivoire, M. Shah, P. Ranganatban, C. Kozyrakis, J. Meza, Models and metrics to enable energy-efficiency optimizations, Computer 40 (12) (2007) 39 –48.

[51] M. Zapater, J. L. Ayala, J. M. Moya, Leveraging heterogeneity for energy minimization in data centers, in: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 752–757.

[52] M. Yoo, A. B. Jette, M. A. Grondona, SLURM: Simple Linux Utility for Resource Management, Lecture Notes in Computer Science.

[53] A. Lucero, Simulation of batch scheduling using real production-ready software tools, `http://www.bsc.es/media/4856.pdf`.