A Measurement-based Characterization of the Energy Consumption in Data Center Servers

Jordi Arjona Aroca, Angelos Chatzipapas, Antonio Fernández Anta, and Vincenzo Mancuso

***Abstract*—*In this work we present an exhaustive empirical characterization of the power requirements of multiple components of data center servers. To do so, we devise different experiments to stress these components, taking into account the multiple available frequencies and the fact that we are working with multicore servers. In these experiments, we measure energy consumption of server components and identify their optimal operational points. Our study proves that the curve defining the minimal CPU power utilization, as a function of the load in Active Cycles Per Second, is neither concave nor purely convex. Instead, it definitively shows a super-linear dependence on the load. Similarly, we present results on how to improve the efficiency of network cards and disks. Finally, we validate the accuracy of the model derived from our characterization by comparing the real energy consumed by two Hadoop applications—PageRank and WordCount—with the estimation from our model, obtaining errors below* 4.1% *on average.*

***Index Terms*—Cloud Computing, CPU, data centers, disk, DVFS, energy efficiency, energy measurements, network**

I. Introduction

Cloud computing represents the possibility of outsourcing our computational services without incurring amortization costs, scaling up or down the required resources to match the current demand, and in a pay-as-you-go way. However, the cloud has to be physically deployed somewhere. To this purpose we require data centers. Indeed, cloud computing is the main reason why the number of data centers built or being built has skyrocketed during the last years. However, the advantages of cloud computing come at a cost, the huge amount of energy data centers consume yearly. In particular, according to the most recent estimations [1], data centers' total energy consumption in 2012 was about 270 TWh, which corresponds to roughly 2% of the global electricity consumption, with an approximated annual growth rate of 4.3%.

In this article, we concentrate on the characterization of data center servers' energy consumption. Indeed, in order to obtain full benefit of energy efficient techniques proposed in the literature [2], [3], it is crucial to profile the utilization of the data center servers' components. Moreover, it is necessary to understand the energy consumption of servers and how it is affected by different load configurations. There is a large body of work on characterizing servers' energy consumption. However, the existing literature does not jointly consider phenomena like the irruption of multicore servers and dynamic voltage and frequency scaling (DVFS) [4], which are key to achieve scalability and flexibility in the architecture of a server. Therefore, more complex/complete models which study the energy consumed by a server are needed. To be consistent, these models have to be based on empirical values. However, we found that there is a lack of empirical works studying servers' energy behavior.

Contributions and main results: Our contribution is threefold: (i) we propose a methodology to empirically characterize the energy consumption of a server, (ii) we provide novel, experimental-based, insights on the power consumption of the components that contribute the most to the server's power consumption, and (iii) we propose an accurate technique to estimate the energy consumption of cloud applications.

As concerns the methodology, we observe that *active CPU cycles per second* (ACPS) is a convenient metric of CPU (central processing unit) load in multi-core/multi-frequency architectures. We show how to isolate the contribution of energy consumption due to CPU, disk I/O operations, and network activity by just measuring server's total energy consumption and a few activity indicators reported by the operating system. We also show that the *baseline* energy consumption of a server — i.e., the energy consumed just because the server is turned on — has a strong impact on server's total consumption.

As concerns the components' energy characterization, we show that, besides the *baseline* consumption, the CPU has the largest impact among all components, and its energy consumption is not linear with the load. Disk I/O operations are the second highest cause of consumption, and their efficiency is strongly affected by the I/O block size used by the application. Eventually, network activity plays a minor yet not negligible role in the energy consumption, and the network impact scales almost linearly with the network transmission rate. All other components (e.g., memory, fans, GPU, etc.) can be accounted for the *baseline* energy consumption, which is subject to minor variations under different operational conditions. Specifically, the main results of our measurement campaign are listed below:

- The CPU power utilization depends on the number of working cores, the CPU frequency, and the CPU load (in ACPS units). Our measurements confirm that the energy consumption with a single working core at constant frequency can be closely approximated by a linear function of the CPU load. However, given a CPU frequency, the energy consumption in multicore architectures is a concave function of the CPU load and can be approximated by a

J. Arjona Aroca is with University Carlos III of Madrid, Spain; A. Chatzipapas, is with IMDEA Networks Institute, Madrid, Spain, and University Carlos III of Madrid, Spain; A. Fernándes Anta and V. Mancuso are with IMDEA Networks Institute, Madrid, Spain. e-mail: {jorge.arjona, angelos.chatzipapas, vincenzo.mancuso, antonio.fernandez}@imdea.org

This research was partially supported by the Madrid Regional Government (CM) through the grant Cloud4BigData (S2013/ICE- 2894, cofunded by FSE & FEDER) and by the Spanish Ministry of Economy and Competitiveness with grant HyperAdapt (TEC2014- 55713-R).

low-order polynomial. The energy consumption for a fixed CPU load is, in general, minimized by using the highest number of cores and the lowest frequency at which the load can be served. However, the minimum achievable energy consumption is a piecewise concave function of the CPU load.

- The energy consumed by hard disks for reading and writing depends on the CPU frequency and the I/O block sizes. Both reading and writing energy costs increase slightly with the CPU frequency. While the energy consumption due to reading is not affected by block size, the energy consumption due to writing increases with the block size. The reading efficiency (expressed in MB/J) is barely affected by the CPU frequency, while writing efficiency is a concave function of the block size since it boosts the throughput of writing until a saturation value is reached.
- The energy consumption and the efficiency of the NIC (Network Interface Card), both in transmission and reception, depends on the CPU frequency, the packet size, and the transmission rate. The efficiency of data transmission increases almost linearly with the transmission rate, with steeper slopes corresponding to lower CPU frequencies. Although a linear relation between transmission rate and efficiency holds for data reception as well, small packet sizes yield higher efficiency in reception.

Overall, supported by our measurements, we provide a holistic energy consumption model that only requires a few calibration parameters for every different server architecture which we want to evaluate (a universal energy model will be too simplistic and inaccurate). We validate our model by means of a server computing the *PageRank* metric of a graph and a *WordCount* application in a *Hadoop* platform, first without network activity, next with bulky network activity, and finally in the cloud. We will find that the error of our energy estimates is below 4.1% on average and never worse than a 10%.

Differences from our prior work. The present work is an extended version of [5]. With respect to our conference paper, here the entire body of the paper has been revisited, new and more detailed experimental and analytical results have been included, as well as new findings on the energy-efficiency observed for the tested servers. Changes and new contents can be summarized as follows:

- More measurements have been realized for CPU, disk and network and new material has been added to reflect the new results regarding the energy efficiency of the aforementioned server components. Specifically, the characterization of the CPU energy consumption and efficiency, and of disk energy consumption has been extended; similarly, the characterization of power efficiency of network interface cards includes now richer details and new results, especially for large servers.
- Additional validation for our proposed model is provided, which includes more cloud-based scenarios.

Organization of the rest of the manuscript. Section II describes the methodology we used for our experiments. Section III presents our measurement campaign, for every single component which we tested. In Section IV we model the energy consumption of the servers based on a few calibration parameters which we find during our measurement campaign. In Section V we discuss our findings and their implications. Section VI provides information about related works and, finally, Section VII concludes the paper.

II. Methodology

In this section we introduce the measurement techniques we used to characterize the energy consumption of CPU activity, disk access (read and write operations), and network activity. We start our measurements by profiling the CPU energy consumption, from where we obtain information about the *baseline* energy consumption of the servers and the energy consumption due to CPU load. Afterwards, we profile the other two components, namely, disk and network. Note that CPU and baseline measurements are of capital importance in order to evaluate the other components, because every time that we run a script to profile the behavior of another component, some CPU cycles are needed in order to execute it as well as to use the component that has to perform the task. Therefore, to understand the contribution of any component, we first need to identify the contribution of the CPU and the baseline and calculate the difference.

To explore the possible parameters which determine the energy consumption of a data center server and to obtain statistical consistency, we run our experiments multiple times. Similarly, we run these experiments in different server architectures in order to validate our results and give consistency to our conclusions.

A. Collecting System Data and Fixing Frequency Parameters

One prerequisite for our experiments is to have Linux machines because we can freely modify and check the Linux kernel, for instance to add kernel modules and utilities[1] which allow us to change CPU frequencies at will, or to log CPU activity stats so we can periodically read the core frequency and the number of *active* and *passive* CPU ticks at each core[2]. Once we have the number of ticks and the core frequency, since a tick represents a hundredth of second, cycles can be calculated as 100 *ticks/frequency*.

We use active cycles per second (ACPS) instead of CPU load percentage to characterize CPU load because ACPS depend on the CPU frequency used, as the higher the frequency the more the work that can be processed. In contrast, CPU load percentages cannot be compared when different frequencies are used, while the amount of ACPS that can be processed can be considered as an absolute magnitude. In order to get (set) information about the operative frequency of the system we used the `cpufrequtils` package[3]. With those tools, we can monitor the CPU frequency at which the system works and assign different frequencies to the cores. However, to limit the number of possible combinations to characterize, we assign the same frequency to all cores.

[1]e.g., cpufrequtils, acpi-cpufreq.

[2]File `/proc/stat` reports the number of ticks since the server started, devoted to *user*, *niced* and *system* processes, waiting (*iowait*), processing interrupts (i.e., *irq* and *softirq*), and *idle*. In our experiments we count both waiting and idle ticks as *passive* ticks, while we denote the aggregated value of the rest of ticks as *active*.

[3]https://wiki.archlinux.org/index.php/CPU_Frequency_Scaling

B. CPU

In order to evaluate the CPU power utilization we prepared a script based on a benchmark application, `lookbusy`.[4] Note that `lookbusy` allows us to load one or more CPU cores with the same load. Our `lookbusy`-based experiment follows the next steps: we first fix the CPU frequency to the lowest possible frequency in the system; then we run `lookbusy` with fixed amount of load for one core during timeslots of 30 seconds, starting with the maximum load and then decreasing the load gradually. After the last `lookbusy` run we measure the power used during an additional timeslot with *no* `lookbusy` load offered. We register the active cycles and the power used during each timeslot.

After taking these different samples for one frequency we move to the immediately higher frequency (we can list and change frequencies thanks to `cpufrequtils`) and repeat the previous steps. After going through all the available frequencies, we restart the whole process but increasing by one the number of active cores. We repeat this whole process until all the cores of the server are active. Note that when we change the frequency of the cores we change it in all of them, active or not, for consistency. Similarly, when more than one core is active, the load for all the active cores is the same.

Once explained the scheme of our experiments, we must clarify the meaning of running a timeslot with *no* load. Note that zero-load is clearly not possible as there is always going to be load in the system due to, e.g., the operating system. However, during the timeslot in which we do not run `lookbusy`, we measure the power corresponding to the operational conditions which are as close as possible to the ones of an idle system. Moreover, the decision of using timeslots of 30 seconds is to guarantee enough, yet not excessive, time for the measurements. In fact, as we start and stop `lookbusy` at the beginning and end of the timeslots, we need to ignore the first and the last few seconds of measurements in each timeslot to avoid measurement noise due to power ramps and operational transitions.

The measured values of load (in ACPS) and power in each timeslot are used to obtain a least squares polynomial fittings curve. These fittings characterize the CPU power utilization for each combination of frequency and number of active cores. We will use as *baseline power utilization* of each one of these configurations the zero-order coefficient of the polynomial of these fittings curves.

C. Disks

The energy consumption of the hard drive was evaluated using two different scripts (for reading and writing) based on the `dd` linux command.[5] We chose `dd` as it allows us to read files, write files from scratch, control the size of the blocks we write (read), control the amount of blocks written (read) and force the commit of writing operations after each block in order to reduce the effect of operating system caches and memory. We combine this tool with flushing the RAM and caches after each reading experiment.

[4]http://www.devin.com/lookbusy.

[5]http://linux.die.net/man/1/dd.

In both our scripts we perform write (read) operations for a set of different I/O block sizes and for different data volumes to be written (read). We record the CPU active cycles, the total power and time used in each one of these operations for each combination of block size and available frequency.

Finally, we identify the contribution of the disk to the total power utilization by subtracting the contribution of both the baseline and the CPU from the measured total power.

Disk I/O experiments shed light on the relevance of the block sizes when reading or writing as well as whether there is an influence of the frequency on these operations.

D. Network

In order to evaluate the contribution of the network to the energy consumption of a cloud data center server, we devised a set of experiments based on a client-server C script devised on purpose for this task.

There are a few aspects that we consider relevant in order to characterize the impact of the NIC on the total energy consumption of a server and that led us to choose these two tools. First, the ability of performing tests in which the server under study acts as sender or as receiver during a network connection, and therefore we can observe server's energy consumption while sending data or receiving it. To clarify the terms, *sender* is the server which injects traffic to the network, and *receiver* is the server which accepts traffic from the network. Second, the ability of those tools to change several parameters that we consider relevant for the energy characterization of the servers, namely, the packet size and the offered load, jointly with the frequency of the system.

Our experiments consist, then, on measuring the achieved data rate, the CPU active cycles per second (ACPS) and the total energy consumption of the server under study either as sender or as receiver using different packet sizes and different transfer rates. We run each experiment multiple times for statistical consistency.

Finally, using the CPU active cycles per second which were measured during the experiment, we identify the energy consumption due to CPU. Subtracting both CPU energy consumption and the baseline energy consumption from the total energy consumption of the experiment, we can isolate the energy consumption of the network.

III. Measurements

A. Devices and Setup

In order to monitor and store the instantaneous power used by a server during the different experiments we used a Voltech PM1000+ power analyzer[6], which is able to measure the total instantaneous power used by the server under test on a per-second basis. In Fig. 1 we show a schematic representation of the setup we used and the components under testing. More specificaly, in order to take our measurements we connected the server being measured to the power analyzer and the latter to the power supply. In the case of servers with power redundancy one of the two power sources was unplugged to ensure that the power measurement was correct.

[6]More information about the PM1000 can be found in http://www.farnell.com/datasheets/320316.pdf

TABLE I: Characteristics of the servers under study

Component	Servers		
	`Survivor`	`Nemesis`	`Erdos`
CPU (#cores)	4	4	64
Freqs List ([GHz])	1.2, 1.333, 1.467, 1.6, 1.733, 1.867, 2.0, 2.133	1.596, 1.729, 1.862, 1.995, 2.128, 2.261, 2.394, 2.527, 2.666, 2.793, 2.794	1.4, 1.6, 1.8, 1.8, 2.1,2.3
RAM	4 *GB*	4 *GB*	512 *GB*
Disk	2 *TB*	2+3 *TB*	2×146 *GB*, 4×1 *TB*
Network	1 *Gbps*	3×1 *Gbps*	4×1 *Gbps*, 2×10 *Gbps*

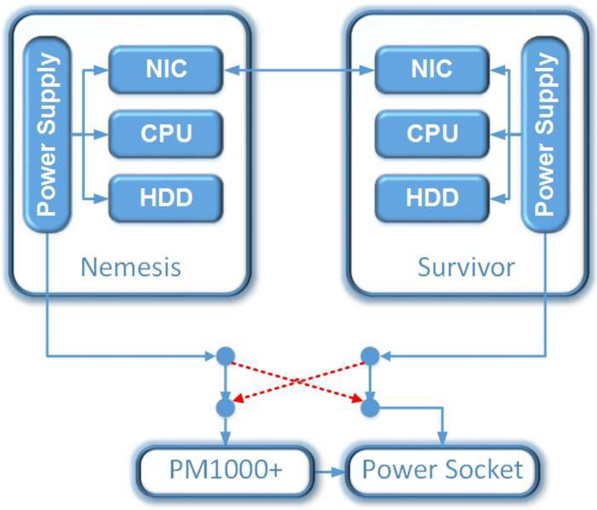


Fig. 1: Schematic representation of our setup when Nemesis is being measured. Red arrows show the alternative scheme to measure Survivor (or Erdos).

In the experiments where the network was not involved (CPU and disk), we unplugged the network cable from the server, which has an impact on the power utilization as the port goes idle. In the network based experiments we established an Ethernet connection between the server under study and a second machine in order to study the server behavior, both as a receiver and as a sender.

We evaluated three different servers: `Survivor`, `Nemesis`, and `Erdos`. We will now present these servers although their main characteristics, including their sets of available CPU frequencies, can be also found in Table I. `Survivor` has an Intel Xeon E5606 4-core processor[7], with 4 *GB* of RAM, a 2 *TB* Seagate Barracuda XT hard drive and a 1 *Gigabit* Ethernet card integrated in the motherboard. `Nemesis` is a Dell Precision T3500 with an Intel Xeon W3530 4-core processor, 4 *GB* of RAM, 2 hard drives (a 2 *TB* Seagate Barracuda XT and a 3 *TB* Seagate Barracuda), a 1 *Gigabit* Ethernet card integrated in the motherboard, and a separate Ethernet card with two 1 *Gigabit* ports. In this study we only evaluate the Seagate Barracuda XT disk and the integrated Ethernet card. Both `Survivor` and `Nemesis` use the Ubuntu Server edition 10.4 LTS Linux distribution. Finally, `Erdos` is a Dell PowerEdge R815 with 4 AMD Opteron 6276 16-core processors (i.e., 64 cores in total), 512 *GB* of RAM, two 146 *GB* SAS hard drives configured as a single RAID1 system (which is the "disk" analyzed here) and four 1 *TB* Near-line SAS hard drives. It also includes four 1 *Gigabit* and two 10 *Gigabit* ports. `Erdos` is a high-end server and uses Linux Debian 7 Wheezy.

[7]FSB frequency was fixed for all CPU frequencies in the experiments performed with Intel machines.

B. Baseline and CPU

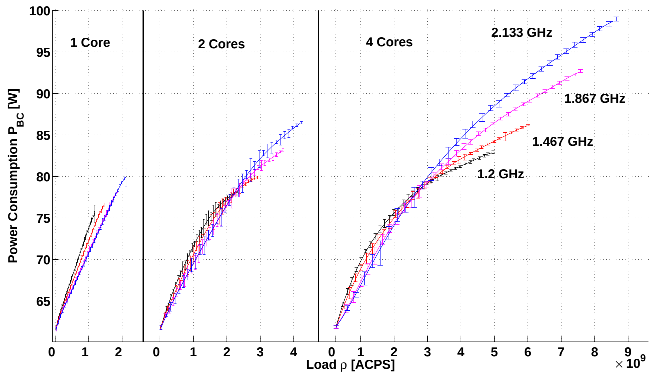
As we mentioned in Section II, for each server we have measured the power it uses with neither disk accesses nor network traffic. We assume that the power utilization observed is the sum of the baseline consumption plus the power used by the CPU. We have obtained samples of the power consumed under different configurations that vary in the number of active cores used, the frequency at which the CPU operates (all cores operate at the same frequency), and the active cores load (all active cores are equally loaded). The list of available and tested CPU frequencies and cores can be found in Table I. We tune the total load ρ by using `lookbusy`, as described in Section II. Each experiment lasts 30 s and it is repeated 10 times. Results are summarized in terms of average and standard deviation. Specifically, in the figures reported in this section, the power utilization for each tested configuration is depicted by means of a vertical segment centered on the average power utilization measured, and with segment size equal to two times the standard deviation of the samples.

The results of these experiments for each of the 3 servers are presented in Fig. 2 (the measurements for some frequencies and some number of cores are omitted for clarity). Here, for each configuration of number of active cores, frequency, and load in ACPS, the mean and standard deviation of all the experiments with that configuration are presented. Also the least squares polynomial fitting curve for the samples is shown for each number of cores and frequency. The curves shown are for polynomials of degree 7, but we observed that using a degree 3 polynomial instead does not reduce drastically the quality of the fit (e.g., the relative average error of the fitting increases from 0.7% with 7-th degree polynomials to 1.5% with degree equal to 3 for `Erdos`, while it remains practically stable and below 0.7% for `Nemesis`). In general, we can use an expression like the following to characterize the CPU power utilization:

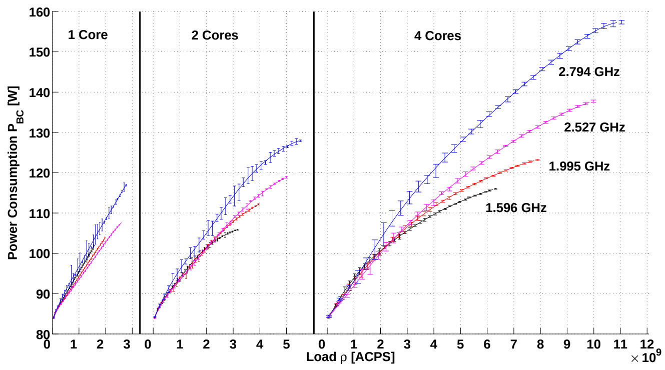
$$P_{BC}(\rho) = \sum_{k=0}^{n} \alpha_k \rho^k, \quad n \le 7, \qquad (1)$$

where P_{BC} includes both the baseline power utilization of the servers and the power used by the CPU, and ρ is the load expressed in active cycles per second. Therefore, coefficient α_0 in Eq. 1 represents the consumption of the system when the CPU activity tends to 0, and we can thereby interpret α_0 as the baseline power utilization of the system. Note that the polynomial fitting, and hence the baseline power utilization α_0, depends on the particular combination of number of cores and frequency adopted. However, for sake of readability, we do not explicitly account for such a dependency in the notation.

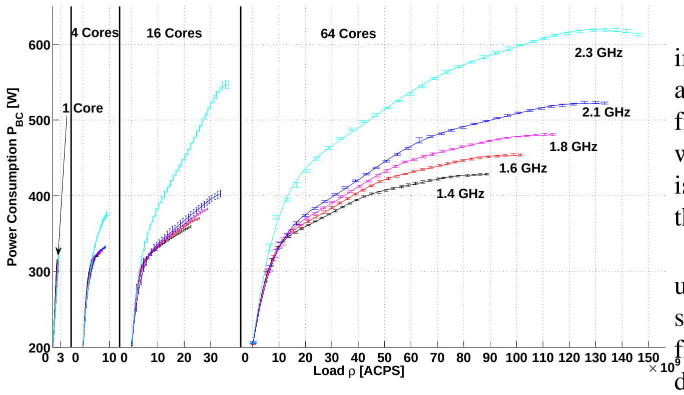
A first observation of the fitting curves for each particular server in Fig. 2 reveals that the power for near-zero load is almost the same in curves (e.g., for `Nemesis` this value is



(a) Survivor

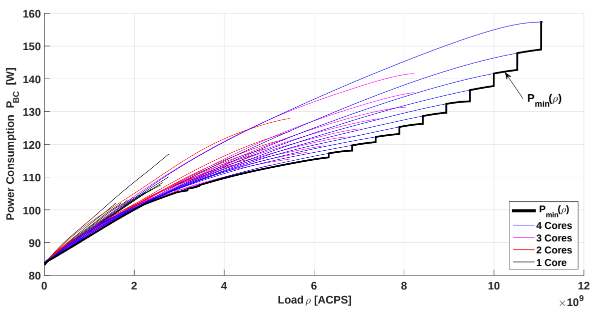


(b) Nemesis

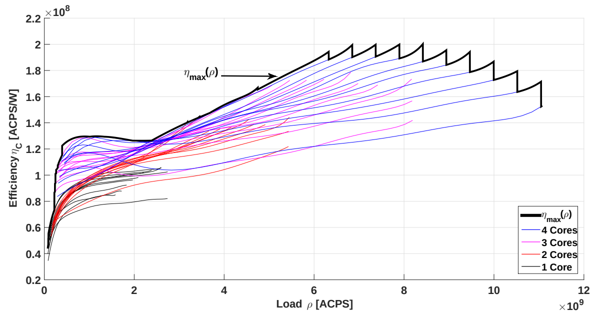


(c) Erdos

Fig. 2: Power utilization of 3 servers (Survivor, Nemesis, and Erdos) for baseline and CPU characterization experiments. In each figure, the color of the curve identifies the frequency used.



(a) Minimal power.



(b) Maximal efficiency.

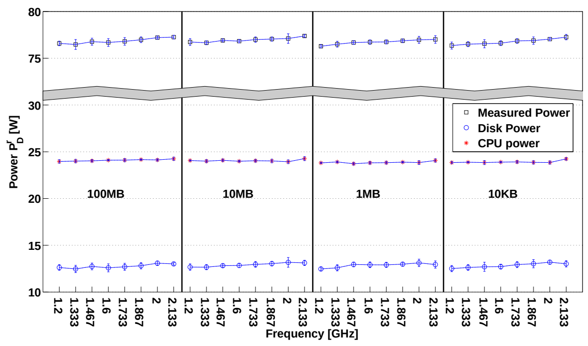
Fig. 3: CPU performance bounds of Nemesis.

between 84 and 85 W). Observe that it is impossible to run an experiment in which the load of the CPU is actually zero to obtain the baseline power utilization of a server. However, all the fitting curves converge to a similar value for $\rho \to 0$, which can be assumed to represent the baseline power utilization.

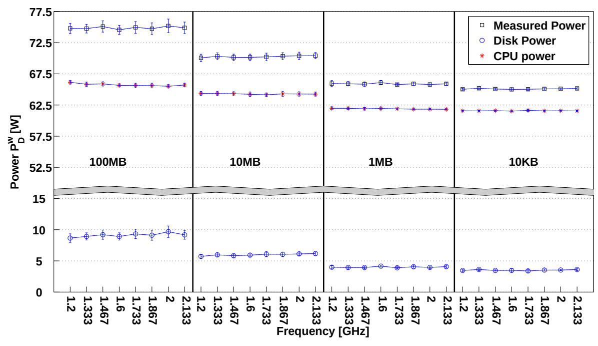
A second observation is that for one core the curves grow linearly with the load. However, as soon as two or more cores are used, the curves are clearly concave, which implies that for a fixed frequency the efficiency grows with the load (we will discuss later the efficiency in terms of number of active cycles per energy unit).

A third observation is that frequency does not significantly impact the power utilization when the load is low. In contrast, at high load, the power clearly increases with the CPU frequency. More precisely, the power grows superlinearly with the frequency, for a fixed load and number of cores. This is particularly evident in the curves characterizing Erdos, the most powerful among our servers.

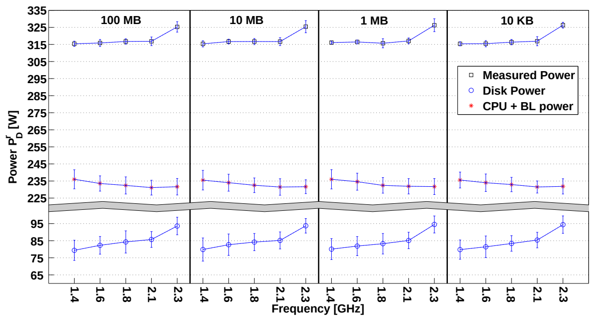
From the previous figures it emerges that the power utilization due to CPU and baseline can be minimized by selecting the right number of active cores and a suitable CPU frequency. Similarly, we can expect that the energy efficiency, defined as number of active cycles per energy unit, can be maximized by tuning the same operational parameters. We graphically represent the impact of operation parameters on power utilization and energy efficiency in Figs. 3, 4 and 5 respectively for Nemesis, Survivor and Erdos. In particular, Figs. 3(a), 4(a) and 5(a) report all possible fitting curves for the power measurements, plus a curve marking the lowest achievable power utilization at a given load. We name such a curve “minimal power curve” $P_{\min}(\rho)$, and we observe that (i) it only depends on the load ρ, and (ii) it is a piecewise concave function, which makes it suitable to formulate power optimization problems. Finally, to evaluate the energy efficiency of the CPU, we report in Figs. 3(b), 4(b) and 5(b) the number of active cycles per energy unit obtained from our measurements respectively for Nemesis, Survivor and Erdos. We compute the power due to active cycles as the power $P_{BC} - \alpha_0$, i.e., by subtracting the baseline consumption from P_{BC}, and we obtain the efficiency η_C by dividing the load (in active cycles per second) by the power due to active cycles, i.e., $\eta_C = \frac{\rho}{P_{BC}(\rho)-\alpha_0}$. Also in this case we show the curve



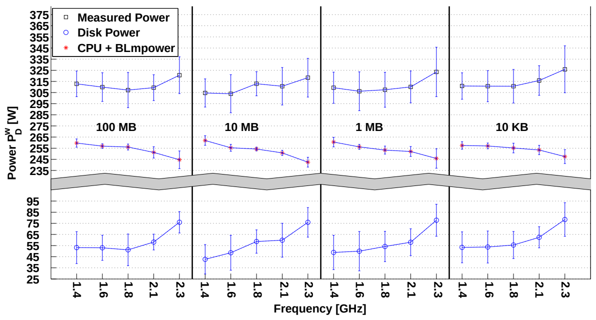
(a) Power utilization during reading (`Survivor`).



(b) Power utilization during writing (`Survivor`).

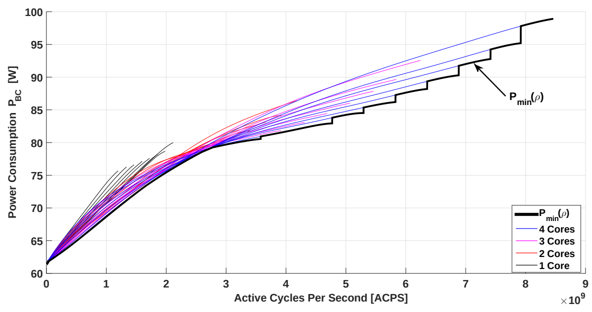


(c) Power utilization during reading (`Erdos`).

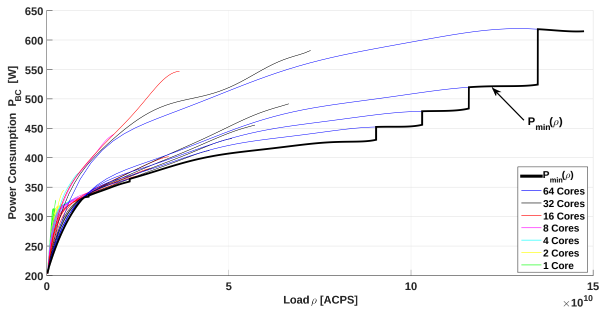


(d) Power utilization during writing (`Erdos`).

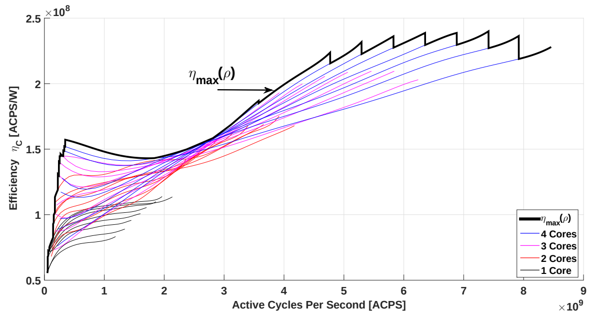
Fig. 6: Instantaneous power utilization for a reading/writing operations. Results are presented for every frequency and for 4 different block sizes for each one of our servers.



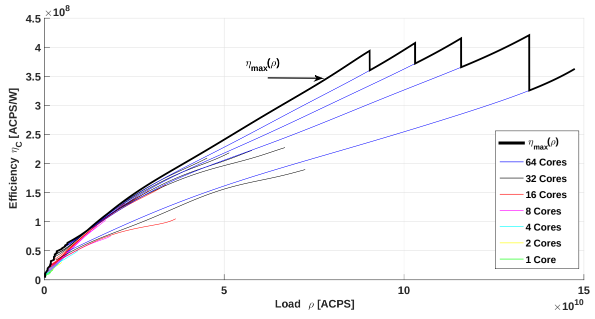
(a) Minimal power.



(a) Minimal power.



(b) Maximal efficiency.



(b) Maximal efficiency.

Fig. 4: CPU performance bounds of `Survivor`.

Fig. 5: CPU performance bounds of `Erdos`.

that maximizes the efficiency at a given load, which we name "Maximal efficiency curve" $\eta_{\max}(\rho)$. Interestingly, we observe that (i) $\eta_{\max}(\rho)$ presents multiple local maxima, (ii) for a given configuration of frequency and number of active cores, the efficiency is maximized at the highest achievable load, (iii) all local maxima corresponds to the use of all available active cores, but (iv) the absolute maximum is *not* achieved neither at the highest CPU frequency nor at the lowest.

C. Disks

We now characterize the power and energy consumption of disk I/O operations. During the experiments, we continuously commit either read or write operations, while keeping the CPU load ρ as low as possible (i.e., we disconnect the network and we do not run other tasks). Still, the power measurements obtained during the disk experiments contain both the power used by the disk and power due to CPU and baseline. Indeed, Fig. 6 shows, for each experiment, the total measured power P_t, the power P_{BC} computed according to Eq. 1 at the load ρ measured during the experiment, and the power due to disk operations, computed as $P_D^x = P_t - P_{BC}(\rho), \quad x \in \{r, w\}$, where r and w refer to reading and writing operations, respectively. We test sequentially all the available frequencies for each server (see Table I), and I/O block sizes ranging from 10 *KB* to 100 *MB*. Fig. 6 shows average and standard deviation of the measures over 10 experiment repetitions. Results for `Nemesis` are omitted since they are like `Survivor`' results. Indeed, `Survivor` and `Nemesis` have similar disks and file systems, while `Erdos` is equipped with SAS disks with RAID. In all cases shown in the figure, the disk power is small but not negligible with respect to the baseline consumption. Furthermore, we can observe that the two servers presented behave differently. Indeed, while the power utilization due to writing is affected by the block size B for both machines, we observe that `Survivor`' disk writing power P_D^w is not affected by the CPU frequency, while `Erdos`' results show an increase with the frequency. A similar behavior is also observed for the reading power of the disk. The main difference is that reading is a more costly operation since the power consumption in reading is approximately 30% higher as can be observed in Figs. 6(a),6(b) and Figs. 6(c),6(d).

Moreover, the results obtained with `Erdos` are affected by a substantial amount of variability in the measurements, which we believe is due to the caching operations enforced by the RAID mechanism in `Erdos`. Furthermore, `Erdos` shows a baseline plus CPU power decrease for both reading and writing. This behavior is because `Erdos` is very powerful and higher CPU frequencies finish the workload faster (keep in mind that disks are the bottleneck for disk-intensive tasks) and therefore in accordance with Fig. 2 the average load ρ will be lower when higher frequencies are used.

Similarly to what was described for the CPU, we can also compute the energy efficiencies η_D^r and η_D^w of disk reading and writing operations, respectively. This efficiency can be computed by subtracting the baseline power from the total power, and by measuring the volume V of data read or written in an interval T as $\eta_D^x = \frac{V}{P_D^x T}, \quad x \in \{r, w\}$. Similarly to what can be seen in Fig. 6, reading efficiency is almost constant at any frequency and for each block size, while writing is more efficient with large block sizes. Also, the efficiency changes very little with the adopted CPU frequency. Efficiency, however, saturates to a disk-dependent asymptotic value, which is due to the mechanical constraints of the disk (e.g., due to the non-negligible *seek* time, the number of read/write operations per second is limited). Although we do not provide a figure due to space limitations, it can be shown that η_D^w is a concave function of the block size B.

D. Network

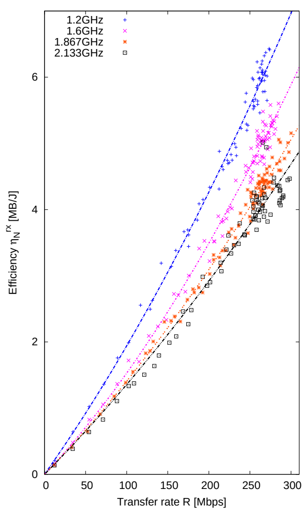
The last server component that we characterize via measurements is the NIC. Similarly to CPU and disk, we run experiments in which only the operating system and our test scripts are active. For the network, we run the scripts described in Sec. II-D to transmit (receive) traffic over a gigabit Ethernet connection and count the system active cycles ρ. We measure the total power utilization P_t during the experiment, so that the power due to network activity can be then estimated as $P_N^x = P_t - P_{BC}(\rho)$, $x \in \{tx, rx\}$, where P_N^{tx} and P_N^{rx} refer to the power consumed when acting as a sender and as a receiver, respectively.

In the experiments, we sequentially test all the available frequencies for each server (see Table I), and fix the packet size and transmission rate within the achievable set of rates (which depends on the packet size, e.g., < 950 *Mbps* for 1470-B packets). We report the results for the network energy in terms of efficiencies η_N^{tx} and η_N^{rx} (volume of data transferred per unit of energy). These efficiencies are computed as $\eta_N^x = \frac{R}{P_N^x}$, $x \in \{tx, rx\}$, where R is the transmission rate during the experiment.

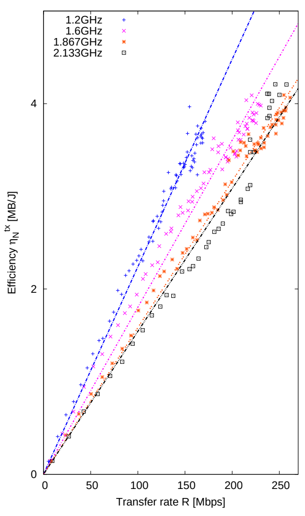
Fig. 7 shows the network efficiencies of `Survivor`, `Nemesis` and `Erdos` averaged over 5 samples per transmission rate R[8]. Due to space limitations, for `Nemesis` and `Erdos`, we include the figures only for η_N^{tx}, however, their behavior when acting as receiver, i.e, η_N^{rx}, is very similar to the one exhibited by `Survivor`. For the sake of readability, the figures only shows results for the biggest and smallest packet sizes, i.e., 64-B and 1470-B packets. For `Nemesis` and `Survivor` we report four CPU frequencies: the lowest, the highest, the most efficient (according to Figs. 3(b) and 4(b)) and an intermediate one, while all five available frequencies for `Erdos` are shown. The figure also reports the polynomial fitting curves for efficiency, which we found to be at most of second order. Since the efficiency is represented in terms of network activity only, in the fitting we force the zero-order coefficient of the polynomials to be 0. Therefore, we can characterize the network efficiencies of our servers as $\eta_N^x = \beta_1 R + \beta_2 R^2$, $x \in \{tx, rx\}$, where the β_i coefficients are computed by minimizing the least square error of the fitting.

It can be observed in Fig. 7 that efficiencies are almost linear or slightly superlinear with the transfer rate, e.g., the receiving efficiency of `Survivor` exhibits an evident quadratic behavior. Indeed, our measurements show that the network power utilization is independent from the throughput, which is a well known result for legacy Ethernet devices. In fact, the NICs of our servers are not equipped with power saving features like, e.g., the recently standardized IEEE 802.3az [6].

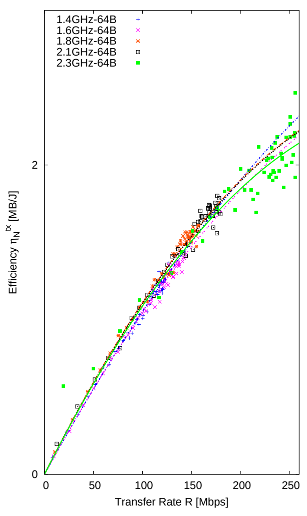
[8]Network results are obtained by using a point-to-point Ethernet connection between two controlled servers.



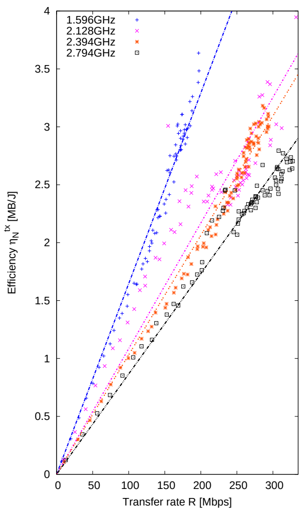
(a) ReceiverEfficiency (**Survivor**).



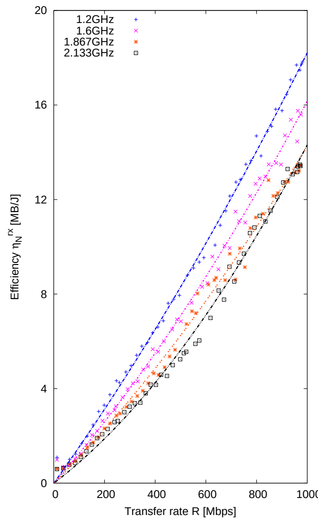
(b) Senderefficiency (**Survivor**).



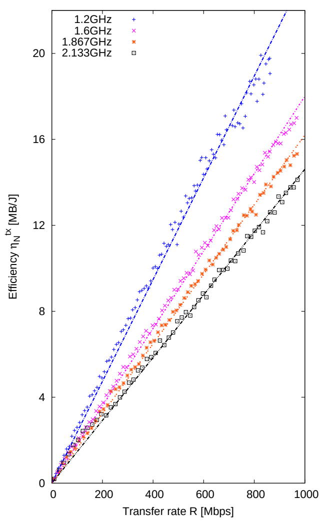
(c) Senderefficiency (**Erdos**).



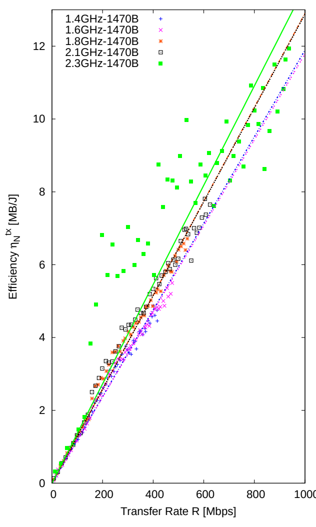
(d) Senderefficiency (**Nemesis**).



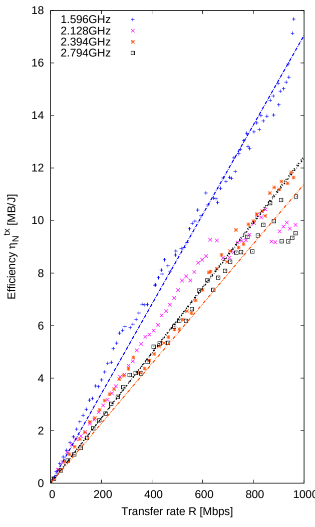
(e) Receiverefficiency (**Survivor**).



(f) Senderefficiency (**Survivor**).



(g) Senderefficiency (**Erdos**).



(h) Senderefficiency (**Nemesis**).

Fig. 7: Network efficiencies for different frequencies and 64-B (upper) and 1470-B (bottom) packets.

In all cases, the efficiency is strongly affected by the selected CPU frequency. Moreover, efficiency is also affected by packet size, although the impact of packet size changes from server to server, e.g., `Survivor` sending efficiency is only slightly affected by it.

Another observation is that, depending on the packet size and frequency used, sending can be more energy efficient than receiving at a given transmission rate, and using the highest CPU frequency is never the most efficient solution. Note also that the efficiency decreases with the packet size, although this effect is particularly evident at the receiver side, while it only slightly impacts the efficiency of the packet sender. However, network activity also causes non-negligible CPU activity, as shown in Fig. 8. Due to space limitations, we show the costs for several configurations for both the sending and receiving cases in `Survivor`, while only for the sending case in `Nemesis` and `Erdos`, however, results were similar to the one from `Survivor`. Overall, the lowest CPU frequency yields the lowest total power utilization during network activity periods.

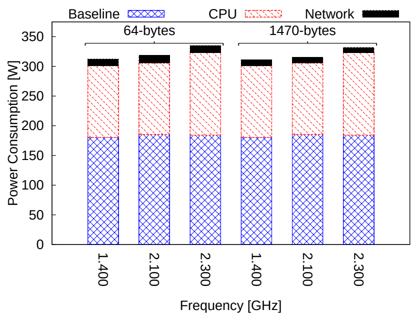
IV. Estimating Energy Consumption

While the results presented in the previous sections are useful to understand the energy consumption pattern of CPU, disk and network, we believe that a much more important use of these results is to estimate the energy consumption of applications. In this section we describe how this can be done from simple data about the application. Moreover, we validate the proposed approach by estimating the energy consumed by several *map-reduce Hadoop* computations.

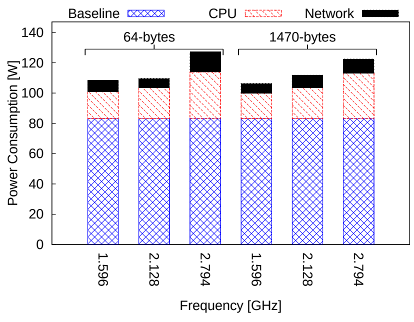
A. Energy Estimation Hypothesis

The approach we propose to estimate the energy E_{app} consumed by an application lays on the basic assumption that the energy is essentially the sum of the baseline energy E_B (baseline power times application running time), the energy consumed by the CPU E_C, the energy consumed by the disk E_D, and the energy consumed by the network interface E_N:

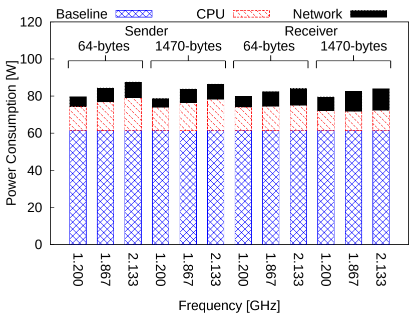
$$E_{app} = E_B + E_C + E_D + E_N. \tag{2}$$



(a) **Erdos (sender).**



(b) **Nemesis (sender).**



(c) **Survivor (sender-receiver).**

Fig. 8: Power utilization with network activity for `Erdos`, `Nemesis` and `Survivor` (64-B experiments were run with a transmission rate $R = 150$ *Mbps*, while $R = 400$ *Mbps* for the experiments with 1470-B packets).

Hence, the process of estimating E_{app} is reduced to estimating these four terms. In order to estimate the first two terms, we need to know the total number of active cycles that the application will execute, C_{app}, and the load ρ_{app} (in ACPS) that the execution will incur in the CPU. From this, the total running time T_{app} can be computed as $T_{app} = C_{app}/\rho_{app}$. Then, once the number of cores and the frequency that will be used have been defined, it is also possible to estimate the baseline plus the CPU energy consumption $E_B + E_C$. For this estimation, we use the fitting curves in Fig. 2 to extract the power utilization, P_{BC}, and multiplying by the execution time of the application, T_{app}, we get the corresponding energy consumption:

$$E_B + E_C = P_{BC}T_{app} = P_{BC}C_{app}/\rho_{app}. \quad (3)$$

The energy consumed by the disk is simply the energy consumed while reading and writing, i.e., $E_D = E_D^r + E_D^w$. To estimate these latter values, the block size to be used has to be decided, from which we can obtain an estimate of the efficiency of reading, η_D^r, and writing, η_D^w. These, combined with the total volume of data read and written by the application, denoted as V_D^r and V_D^w respectively, allow to obtain the estimate energy as

$$E_D = \frac{V_D^r}{\eta_D^r} + \frac{V_D^w}{\eta_D^w}. \quad (4)$$

Finally, to estimate E_N, the transfer rate R and the packet size S have to be chosen, which combined with the frequency used, yield sending and receiving efficiencies η_N^{tx} and η_N^{rx} (see Fig. 7). Then, if the total volumes of data to be sent and received are V_N^{tx} and V_N^{rx}, respectively, the energy spent due to network is as follows:

$$E_N = \frac{V_N^{tx}}{\eta_N^{tx}} + \frac{V_N^{rx}}{\eta_N^{rx}}. \quad (5)$$

Summing up Eqs. 3, 4, and 5 we obtain the estimated E_{app}.

B. Applications and Scenarios for Validation

In this subsection we present the applications and scenarios we experimented with in order to validate the model presented in Section IV-A. Our goal was to be able to estimate the energy consumed by an application deployed on a data center based on the usage of its different components. For that, we executed two different Hadoop applications, PageRank and WordCount, in three different scenarios: first with an Isolated Server (no network), second with a server connected to the network, and finally with a two-server cloud. For the first two scenarios we used `Nemesis`, whereas, for the cloud case, we used both `Nemesis` and `Survivor`. We describe applications and scenarios in detail below.

Our first application is a **Hadoop Map-Reduce PageRank** based application that follows the approach from Castagna [7]. This application, that we denote PageRank for simplicity, computes several iterations of the pagerank algorithm on an Erdos-Renyi random (directed) graph with 1 million nodes and average degree 5[9] The execution of the PageRank application has three phases: preprocessing, map-reduce, and postprocessing. On its side, the map-reduce phase is a sequence of several homogeneous iterations of the PageRank algorithm that runs until a certain threshold is met. For simplicity, we only estimate the energy consumed during the map-reduce phase of the pagerank algorithm, which we force to run 10 times.

Our second application is the **Hadoop Map-Reduce WordCount**. This is a simple program that reads text files and counts how often words occur. For WordCount we use a few hundreds of books as input and estimate the energy consumed for the whole map-reduce process.

As we have mentioned above, these applications are run in 3 different scenarios. In the first scenario, denoted as **Isolated Server**, we run Hadoop in `Nemesis` keeping it disconnected from the network. When we run our applications in this scenario we are basically measuring the impact on the energy consumption of the baseline, CPU and hard disk.

In the second scenario, denoted as **Connected Server**, we run Hadoop in `Nemesis` while it exchanges data on a gigabit LAN. In order to measure the effect of the network on the energy consumption, we evaluate 4 different cases for each application. These cases result from combining 2 different behaviors, depending on whether `Nemesis` acts as a sender or as a receiver of data, with 2 different packet sizes, 64 and 1470 bytes. To do so, we run Iperf, as a server or as a client according to the case, in parallel with Hadoop.

[9]Our PageRank algorithm assigns one input graph to each mapper so, in order to have one map task in each machine, two instances of this graph had to be used in the cluster scenario.

Finally, in the third scenario, denoted as **Cloud**, we set up a two-server Hadoop cluster with `Nemesis` and `Survivor`. In this scenario `Nemesis` is configured as the master node of the cluster and `Survivor` as a slave node. The execution of the applications is shared by both nodes so Hadoop itself exchanges traffic between both servers, and we do not insert additional network traffic in this case. Finally, in order to have a better control of the experiment, we force the reduce tasks to be mandatorily run in `Nemesis`, which also conditions the way the data is exchanged between `Nemesis` and `Survivor`.

Observe that all 3 scenarios are based on Hadoop. This implies that, apart from the map and reduce tasks due to the applications being run, there are some extra processes executed in the servers we are using. The most important processes that we can find in `Nemesis` are *NameNode* (the process that keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept), *Secondary NameNode* (that performs periodic checkpoints of the *NameNode*), *DataNode* (the process that is in charge of storing data in the Hadoop File System (HDFS)), *JobTracker* (that receives the jobs and submits MapReduce tasks to the cluster nodes) and *TaskTracker* (a per node process that can accept a determined number of MapReduce tasks). On its side, `Survivor` runs, in the cloud scenario, DataNode and TaskTracker.

C. Experiments and Observed Results

For the sake of consistency in the results, we ran both applications 10 times per frequency for each one of the considered scenarios and averaged the results.

We start by describing the *Isolated Server* scenario. For each run i we record the total number of active cycles executed C^i_{app}, the time spent T^i_{app} and the volume of data read (written), $V^{r,i}_D$ ($V^{w,i}_D$). Since we cannot measure the instantaneous CPU load, we assume that the CPU load is the same during the run for a given frequency. Hence, it can be estimated as $\rho^i_{app} = C^i_{app}/T^i_{app}$. Then, from ρ^i_{app} we obtain the estimate of the instantaneous power P^i_{BC} using the fitting curves as described in Section III. Finally, using Eq. 3 we compute the estimate $E^i_B + E^i_C$. In order to estimate the energy consumed by the disk operations, we use the fact that Hadoop uses a block size of 64 MB. This allows us to estimate the reading (writing) efficiencies, $\eta^{r,i}_D$ ($\eta^{w,i}_D$) that we compute, in Joules per byte. Combining these values with the measured volume of data read and written ($V^{r,i}_D$ and $V^{w,i}_D$), as described in Eq. 4, we obtain E^i_D.

The total estimated energy of the application, E^i_{app}, is obtained by summing up the energy of the different components used in run i, as stated in Eq. 2 (remember that, in the *Isolated Server* the network is not used). We sum the values of the ten runs of an experiment and we get the estimated $E_{app} = \sum_{i=1}^{10} E^i_{app}$. The (approximated) total *real* energy $\hat{E}^i_{app}$ consumed in run i is computed by the average value of the power samples which we registered with the power analyzer during the run, and we multiply it with the run time T_{app}. Then, the total energy consumed by the experiment is obtained as $\hat{E}_{app} = \sum_{i=1}^{10} \hat{E}^i_{app}$. The estimation error for each experiment is then computed as $\hat{E}_{app} - E_{app}$.

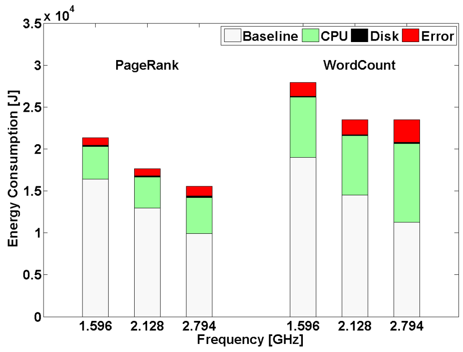


Fig. 9: Energy consumption of `Nemesis` in the Isolated Server scenario.

We show the results obtained for the *Isolated Server* scenario with the minimum, the maximum, and the most efficient[10] frequencies (the results for the remaining frequencies are similar) in Fig. 9. The figure shows the results for both PageRank and WordCount. As can be seen, the error is relatively small, except for the case when we run WordCount at the maximum frequency. Errors are of 4%, 4%, 7%, 5%, 7% and 10% respectively, following the same order as in Fig. 9.

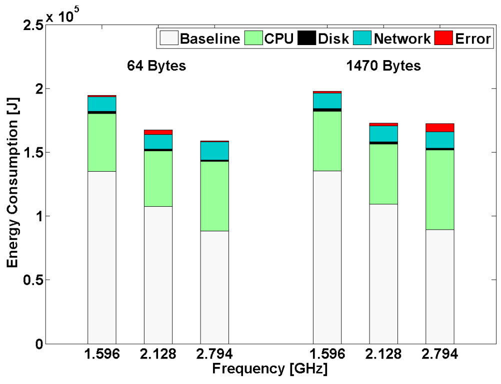
We move now to the *Connected Server* scenario. As we described in the previous section, this scenario is studied in 4 different cases depending on whether `Nemesis` acts as sender or receiver and whether the size of the packets is of 64 or 1470 bytes. Of course, another relevant parameter is the rate at which these packets are sent. The rates used are 150 and 400 Mbps when using packets of 64 or 1470 bytes, respectively.

The total energy consumed in these cases is computed in the same way as we did for the *Isolated Server* scenario but adding the contribution of the network. In order to estimate the network consumption in one run with `Nemesis` sending traffic (resp., receiving traffic), the sending efficiency η^{tx}_N, (resp., receiving efficiency η^{rx}_N) is obtained from the transfer rate R, the frequency and packet size used (see Fig. 7). The amount of data sent (resp., received) can be obtained from the server itself by consulting the OS registers[11]. Therefore, the energy of the network for an run i, E^i_N, is obtained using Eq. 5. Then, including E^i_N for each run in the computation of E^i_{app} we can obtain the total energy consumed by the application. Following the same steps as in the previous scenario, we get the results shown in Fig. 10 and 11. The error measured is again relatively smaller for PageRank than for WordCount. The error measured for each of the cases can be found in Table II.

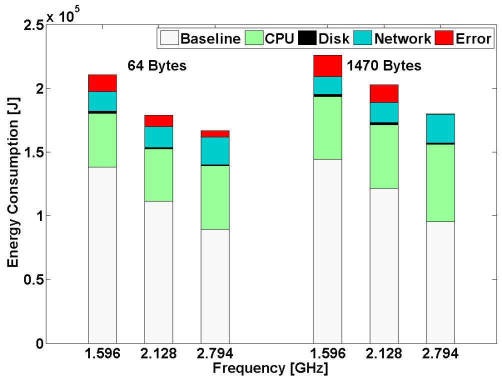
We finally analyze the *Cloud* scenario. In this scenario we set up a cluster with two servers, `Nemesis` and `Survivor`, and run the 2 aforementioned Hadoop applications in it. This scenario may seem relatively similar to the *Connected Server* scenario, but it has is a major difference. While in the previous scenario we were the ones controlling the network

[10]Respectively 1.596, 2.128 and 2.794 GHz, according to the results shown in Section III.

[11]We can read the registers rx_bytes, rx_packets, tx_bytes or tx_packets from `/sys/class/net/eth0/statistics`.

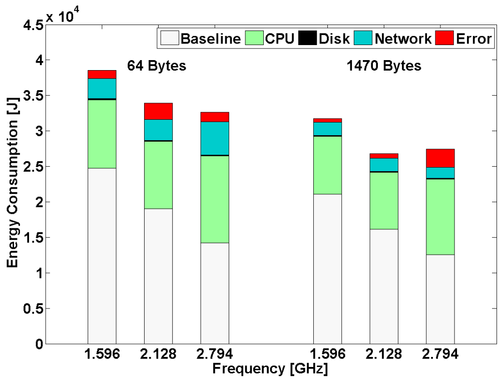


(a) PageRank, sender side.

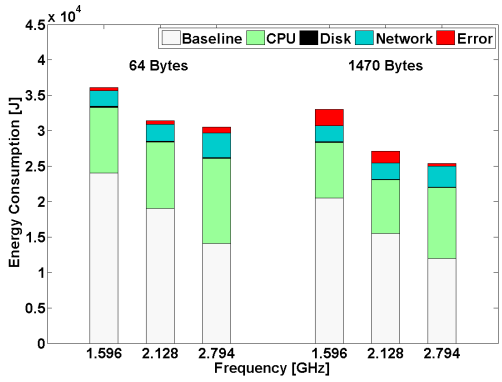


(b) PageRank, receiver side.

Fig. 10: Energy consumption of `Nemesis` running PageRank in the Connected Server scenario, with either small or big packets.



(a) WordCount, sender side.



(b) WordCount, receiver side.

Fig. 11: Energy consumption of `Nemesis` running WordCount in the Connected Server scenario, with either small or big packets.

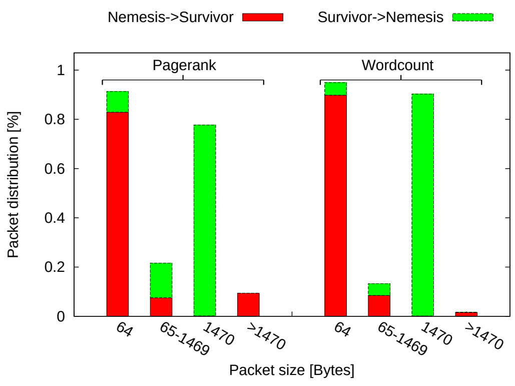


Fig. 12: Distribution of the sizes of the packets exchanged between *Nemesis* and *Survivor* for both PageRank and WordCount in the Cloud scenario.

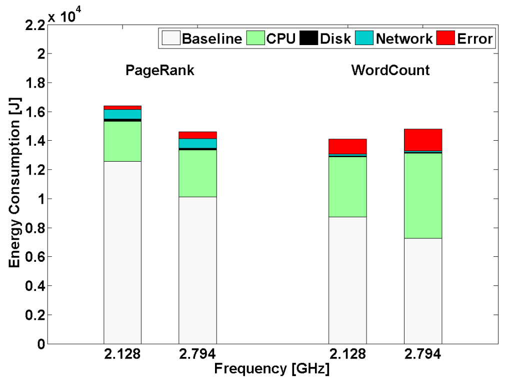
TABLE II: Error measured in the different cases of the *Connected Server* scenario.

Packet Size	Freq	Cases			
		PR - Send	PR - Rec	WC - Send	WC - Rec
64-B	1.596	0.5%	6.0%	2.9%	2.7%
	2.128	2.0%	4.7%	6.4%	1.5%
	2.794	0.5%	2.9%	4.0%	2.9%
1470-B	1.596	0.7%	6.9%	1.6%	6.5%
	2.128	1.1%	6.5%	5.8%	5.8%
	2.794	3.8%	0.3%	0.9%	1.5%

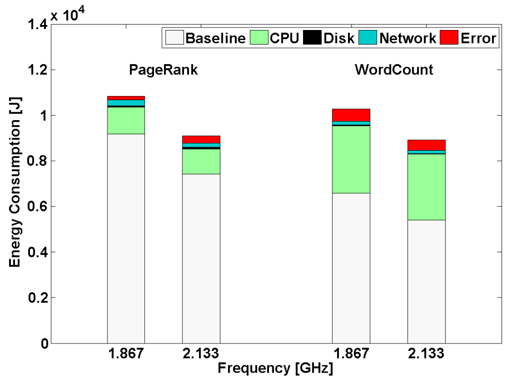
traffic, here the traffic is controlled by Hadoop. Specifically, we know that, in this scenario, there are two main sources of traffic: requesting input data when it is not present in a server, and sending the mapper tasks outputs to the reducer tasks. The only condition we impose in the server to have some control over the traffic is related to this later aspect, we force the reducers to be always in `Nemesis`.

Although we are able to retrieve the total amount of data received or sent by each server, we know neither the size of the packets used nor the rate. Therefore, we can compute neither the sending efficiency η_N^{tx} nor the receiving efficiency η_N^{rx}. In order to be able to compute both the sending and receiving efficiencies we analyze the traffic exchanged by both servers for each one of the applications. Fig. 12 shows the amount of packets of each size that were exchanged by both servers (and the direction of the exchange) for both applications. The results show the vast majority of packets are either small (64 bytes) or big (1470 bytes). Moreover, it shows that most of the packets sent from `Nemesis` to `Survivor` are small packets for both applications, while big packets are sent in the opposite direction.

Given these results, we approximate the energy consumed by the network assuming that all the packets exchanged are of the same size and that the rate is the maximum achievable rate for each packet size according to the results from Section III. For instance, we consider roughly 30 Mbps when Survivor receives 64-Byte packets and roughly 970 Mbps if it sends 1470-Byte packets. These assumptions allow us to compute now η_N^{tx} and η_N^{rx}. The remaining parameters are computed as for the other scenarios, so to determine



(a) Nemesis



(b) Survivor

Fig. 13: Energy consumption of `Nemesis` and `Survivor` in the Cloud scenario.

$\hat{E}_{app}$ and E_{app}. The results are shown in Fig. 13. As in the previous scenarios, errors are relatively low. In particular, the error in `Nemesis` when running PageRank is 3.1% and 1.4% for 2.128 GHz and 2.794 GHz, respectively, and of a 9.7% and a 6.5% for 2.128 GHz and 2.794 GHz when running WordCount. On the other hand, the measured errors for `Survivor` are 3.3% and 3.6% for 1.867 GHz and 2.133 GHz when running PageRank and 5.1% and 5.2%, respectively, when running WordCount.

V. Discussion

We discuss now some of the implications of our results. We start with consolidation as a technique for energy saving. It has been often assumed that the best way of saving energy is by using the highest frequency available and applying consolidation (which is to fill servers as much as possible). This reduces the total number of servers being used, allowing to switch off the rest. This assumption has led to proposing bin-packing based solutions [8], [9], [10], [11]. However, the results presented in Figs. 3(b), 4(b) and 5(b) show that the highest frequency is not always the most efficient one, and this has been found to be true for two different architectures (Intel and AMD). This implies that, by running servers at the optimal amount of load, and the right frequency, a considerable amount of energy could be saved.

A second relevant aspect is the baseline consumption of servers. The results presented for all 3 servers show that their baselines are within a 30-50% of the maximum consumption. Then, it is obvious that more effort has to be done for reducing baseline consumption. For instance, a solution could consist in switching off cores in real time, not just disabling them, or in introducing very fast transitions between active and lower energy states, i.e., to achieve real *suspension* in idle state.

There is another relevant issue related to the CPU load associated to disk and network activity. It can be observed in Fig. 6 that disks do not incur much CPU overhead. In fact, the power used by the CPU plus baseline does not change much across the experiments. Instead, the energy consumed by the CPU due to network operations is even larger than the energy consumed by the NIC (see Fig. 8). Some works [12] have already pointed out that the way packets are handled by the protocol stack is not energy efficient. Our results reinforce this feeling and point out that building a more efficient protocol stack would certainly reduce the amount of energy consumed due to the network.

Finally, it is worth to mention that in this work we have assumed that the power utilization of the RAM memory is included in the *baseline*. The characterization experiments have been run in such a way that there were few memory accesses, so its power utilization did not affect our measurements. However, RAM memory became an uncontrolled source of power utilization in Section IV-C when we validated our proposed model. In fact, all the Hadoop processes that run in the servers consume significant RAM memory. This impacts more significantly the memory used by the cluster's master node, since it runs internal Hadoop processes (such as the NameNode or the JobTracker) whose memory requirement increases with the number of mappers and reducers. This cost is, therefore, paid only in `Nemesis`, the master node of our cluster, and not in `Survivor`, which explains the different accuracy of the model for the two servers. This error is particularly evident when WordCount is run, due to the fact that the required number of mappers for WordCount is larger than for PageRank and, therefore, the RAM required in `Nemesis` increases and so does the uncontrolled energy consumption.

VI. Related Work

There is a large body of work in the field of modeling server energy consumption and its components, both theoretically and empirically. The consumption of servers has been assumed as linear, e.g., by Wang *et al.* [10], Mishra *et al.* [9] or Beloglazov *et al.* [8], who assumed models in which energy consumption mainly depends linearly on CPU utilization. Based on the models, they proposed bin-packing-like algorithms to reduce energy consumption. Other works like the ones from Andrews *et al.* [13] or Irani *et al.* [14] proposed non-linear models, claiming that energy could be saved by running processes at the lowest possible speed. However, we have experimentally shown that current data center servers exhibit non-linear behaviors in terms of energy consumption and that the impact of frequency is not straightforward in modern servers.

Moving to the empirical field, we first classify works in two different groups, depending on whether they consider the effect of frequency in their analysis. We start with works not considering frequency. In this category we find articles proposing models where server components follow a linear behavior, like in [15], [16], [17] or more complex ones, like in [18], [19], [20]. In [16] Liu *et al.* proposed a simple linear model and evaluate different hardware configurations and types of workloads by varying the number of available cores, the available memory, and considering also the contribution of other components such as disks. Vasan *et al.* [17] monitored multiple servers on a datacenter as well as the energy consumption of several of the internal elements of a server. However, they considered that the behavior of this server could be approximated by a model based only on CPU utilization. Similarly, Krishnan *et al.* [15] explored the feasibility of lightweight virtual machine power metering methods and examined the contribution of some of the elements that consume energy in a server like CPU, memory and disks. Their model depends linearly on each of these components. In [19], Economou *et al.* proposed a non-intrusive method for modeling full-system energy consumption by stressing its components with different workloads. Their resulting model is also linear on the utilization of server components. Finally, Lewis *et al.* [20] and Basmasjian *et al.* [18] presented much more complex models which, apart from the contribution of different components of the server, consider extra parameters like temperature and cache misses as well as multiple cores. In particular, Lewis *et al.* [20] reported also an extensive study on the behavior of reading and writing operations in hard disk and solid state drives. We go beyond existing work by showing that, in data centers, non-linear models and a new load metric are required to improve the accuracy of energy consumption estimation. Furthermore, we complement existing studies by showing both individual and joint effects of load, I/O block sizes, network activity and CPU frequencies.

Next we move to the works which also consider frequency in their analysis. Miyoshi *et al.* [21] analyzed the runtime effects of frequency scaling on power and energy. Brihi *et al.* [22] presented an exhaustive study of DVFS using a `cpufrequtils` as we do. Main differences with our work were that they studied four different power management policies under DVFS and centered their study on the relationship between CPU and power utilization. However, they also presented interesting results about disk consumption that match partially our results, showing a flat consumption in reading operations and variations in the writing ones that they attribute to the size of the files being written. Although it was not the main objective of their work, Raghavendra *et al.* [23] performed a per-frequency and core CPU power characterization of two different blade servers. However, they claimed that CPU power depends linearly on its utilization. The main difference with our analysis is that we consider that the load supported by a server increases with the number of active cores and, hence, this load should not be represented in percentage. Gandhi *et al.* [24] published the analysis of global energy consumption versus frequency, based on DVFS and DFS and gave some intuition about the non-linearity of this relation. However, so far there has been no work like ours, i.e., presenting a per-component analysis that allows us to enter into deeper details on the energy versus frequency analysis.

Moreover, there are studies that model the energy consumption behavior for clouds and try to balance the load in order to operate the cluster in its most efficient load-power combination. MUSE [25] is one of the first works that consider a resource management architecture for data centers. Its energy efficient approach dynamically assigns jobs to the servers based on the workload (for CPU and disk) and the potential energy consumption. The authors measure the energy consumption of servers and switches involved in the cluster and conclude that at least 29% of the energy can be saved by MUSE for typical web workloads. In [26] the authors proposed a consolidation algorithm that considers the workloads of the servers in the cloud in order to find the least possible energy consumption point. Their study shows that the energy consumption of a server using variable loads for CPU and disks has an optimal operating point. Given the data from the various servers the algorithm can estimate the ideal load distribution among the servers. The authors in [27] modeled the energy consumption of data centers equipment (i.e., servers, storage, switches) for cloud computing based on existing energy consumption measurements or publicly available data sheets for each of the components (CPU, disk, network, switches). The model estimates the energy consumption per bit from the data center to the user and further analyzes the energy consumption for different types of services, i.e., storage, software, processing. However, existing works on clouds lack experimental inputs on energy consumption. Moreover, not only in our experiments we had a complete control of servers and network and we were able to correlate activity and consumption of different components, but also we unveiled that baseline energy consumption is key to achieve good analytical estimates.

We conclude with some works that also consider frequency but do not model the energy consumption of a server. First of them, the work from Le Sueur *et al.* [28] presented an analysis of the evolution of the effectiveness of DVFS and how it is reduced in the newest and most optimized servers. They show that DVSF might be soon obsoleted by the adoption of ultra low power sleep modes. Ge *et al.* proposed PowerPack [29], a framework that includes a set of toolkits to perform an exhaustive profile of the power utilization of servers and its components. Their analysis is centered in showing the contribution of multicore system to the efficiency of several applications and, hence, no power characterization is presented. Finally, Basmadjian *et al.* [30] published an in deep analysis of the components of a processor and its contribution to the energy consumption of the CPU, shedding some light on the behavior of multicore servers. Some of their conclusions are very relevant to our work, as they show, for instance, that the energy consumption of multiple cores performing parallel computations is not equal to the sum of the power of each of those active cores. Our experiments and model support their findings and shed light on the nature of such effect.

VII. Conclusions

In this work we have reported our measurement-based characterization of energy and power consumption in a server. The results obtained in this work can be enumerated as follows:

1) We have exhaustively measured the power consumed by CPU, disk, and NIC under different configurations, identifying the optimal operational levels, which usually do not correspond to the static system configurations commonly adopted.
2) We found that, besides the *baseline component*, which does not change significantly with the operational parameters, the CPU has the largest impact on energy consumption among all the three components.
3) We also observed that CPU power consumption is neither linear nor concave with the load, i.e., the systems are not *energy proportional*.
4) Our results showed that disk I/O is the second larger contributor to power consumption, although performance changes sensibly with the I/O block size used by the applications.
5) Finally, the NIC activity is responsible for a small but not negligible fraction of power consumption, which scales almost linearly with the network transmission rate.

In general, these results show that most of the energy/power performance figures do not scale linearly with the utilization, in contrast to what is commonly assumed in the literature. We have then shown how to predict and optimize the energy consumed by an application via a concrete example using network activity plus PageRank computation in Hadoop. Our model achieves very accurate energy estimates, obtaining errors below 4.1% on average, and always a within 10%, with respect to the measured total energy consumption.

Finally, as future work, we plan to do an evaluation of the energy profile of multiple typical cloud operations, like instantiation of virtual machines, in an OpenStack deployment. Similarly, using the same applications we ran in our servers, we will study the drawbacks, in terms of energy, of having them ran in VM flavors similar to the typically offered ones by private cloud providers such as Amazon Web Services or Google Cloud.

References

[1] W. V. Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, “Trends in worldwide ICT electricity consumption from 2007 to 2012,” *Computer Communications*, 2014.

[2] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, “Power and performance management of virtualized computing environments via lookahead control,” *Cluster computing*, vol. 12, no. 1, pp. 1–15, 2009.

[3] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, “Making scheduling “cool”: Temperature-aware workload placement in data centers.” in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC ’05. Berkeley, CA, USA: USENIX Association, 2005, pp. 61–75. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247360.1247365

[4] M. Weiser, B. Welch, A. Demers, and S. Shenker, “Scheduling for reduced CPU energy,” in *Mobile Computing*, 1996, vol. 353, pp. 449–471.

[5] J. Arjona Aroca, A. Chatzipapas, A. Fernández Anta, and V. Mancuso, “A measurement-based analysis of the energy consumption of data center servers,” in *Proceedings of the 5th international conference on Future energy systems*. ACM, 2014, pp. 63–74.

[6] IEEE Std. 802.3az, “Energy Efficient Ethernet,” 2010.

[7] P. Castagna, “Having fun with pagerank and mapreduce,” *Hadoop User Group UK talk. Available: http://static.last.fm/johan/huguk-20090414/paolo_castagna-pagerank.pdf*, 2009.

[8] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” *Future Gener. Comp. Sy.*, vol. 28, no. 5, pp. 755–768, 2012.

[9] M. Mishra and A. Sahoo, “On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach,” in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, ser. CLOUD ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 275–282. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2011.38

[10] M. Wang, X. Meng, and L. Zhang, “Consolidating virtual machines with dynamic bandwidth demand in data centers,” in *IEEE INFOCOM*, 2011, pp. 71–75.

[11] L. Nonde, T. E. El-Gorashi, and J. M. Elmirghani, “Energy efficient virtual network embedding for cloud networks,” *Journal on Lightwave Technology*, vol. 33, no. 9, pp. 1828–1849, 2015.

[12] A. Garcia-Saavedra, P. Serrano, A. Banchs, and G. Bianchi, “Energy consumption anatomy of 802.11 devices and its implication on modeling and design,” in *ACM CoNEXT*, 2012, pp. 169–180.

[13] M. Andrews, S. Antonakopoulos, and L. Zhang, “Minimum-cost network design with (dis)economies of scale,” in *IEEE FOCS*, 2010, pp. 585–592.

[14] S. Irani, S. Shukla, and R. Gupta, “Algorithms for power savings,” *ACM Trans. Algorithms*, vol. 3, no. 4, Nov. 2007. [Online]. Available: http://doi.acm.org/10.1145/1290672.1290678

[15] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, “VM power metering: feasibility and challenges,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 3, pp. 56–60, 2011.

[16] C. Liu, J. Huang, Q. Cao, S. Wan, and C. Xie, “Evaluating energy and performance for server-class hardware configurations,” in *IEEE NAS*, 2011, pp. 339–347.

[17] A. Vasan, A. Sivasubramaniam, V. Shimpi, T. Sivabalan, and R. Subbiah, “Worth their Watts? - An empirical study of datacenter servers,” in *IEEE HPCA*, 2010, pp. 1–10.

[18] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, and G. Giuliani, “A methodology to predict the power consumption of servers in data centres,” in *ACM e-Energy*, 2011, pp. 1–10.

[19] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, “Full-system power analysis and modeling for server environments,” in *Proceedings of Workshop on Modeling, Benchmarking, and Simulation*, 2006, pp. 70–77.

[20] A. W. Lewis, S. Ghosh, and N.-F. Tzeng, “Run-time energy consumption estimation based on workload in server systems,” *HotPower’08*, pp. 17–21, 2008.

[21] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar, “Critical power slope: understanding the runtime effects of frequency scaling,” in *ACM ICS’02*, 2002, pp. 35–44.

[22] A. Brihi and W. Dargie, “Dynamic voltage and frequency scaling in multimedia servers,” in *IEEE AINA*, 2013.

[23] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, “No power struggles: Coordinated multi-level power management for the data center,” in *ACM SIGARCH Computer Architecture News*, vol. 36, no. 1. ACM, 2008, pp. 48–59.

[24] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, “Optimal power allocation in server farms,” in *ACM SIGMETRICS*, 2009, pp. 157–168.

[25] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, “Managing energy and server resources in hosting centers,” in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5, 2001, pp. 103–116.

[26] S. Srikantaiah, A. Kansal, and F. Zhao, “Energy aware consolidation for cloud computing,” in *Proceedings of the 2008 conference on Power aware computing and systems*, vol. 10. San Diego, California, 2008.

[27] J. Baliga, R. W. Ayre, K. Hinton, and R. Tucker, “Green cloud computing: Balancing energy in processing, storage, and transport,” *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011.

[28] E. Le Sueur and G. Heiser, “Dynamic voltage and frequency scaling: The laws of diminishing returns,” in *Proceedings of 2010 Usenix HotPower*, 2010, pp. 1–8.

[29] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, “Powerpack: Energy profiling and analysis of high-performance systems and applications,” *IEEE TPDS*, vol. 21, no. 5, pp. 658–671, 2010.

[30] R. Basmadjian and H. de Meer, “Evaluating and modeling power consumption of multi-core processors,” in *IEEE e-Energy*, 2012, pp. 1–10.

Jordi Arjona Aroca is a Ph.D. student at the Universidad Carlos III de Madrid. He received the Telecommunications Engineering degree from the Universidad Politécnica de Valencia and his Masters degree in Telematic Engineering at the Universidad Carlos III de Madrid. As part of his Ph.D. studies, Jordi Arjona Aroca is involved in the field of energy optimization in data centers and data center networking.

Angelos Chatzipapas received his B.S. degree in computer engineering and informatics from University of Patras, Greece, in 2006. He further received his M.Sc. in telematics engineering from University Carlos III in Madrid, Spain, in 2012. Currently, he is a Ph.D candidate at University Carlos III of Madrid and IMDEA Networks Institute. His research interests are in the areas of energy efficient networking, network programming, telecommunications, renewable energy sources and photovoltaics.

Antonio Fernández Anta (M'98-SM'02) is a Research Professor at IMDEA Networks Institute. Previously he was a Full Professor at the Universidad Rey Juan Carlos (URJC) in Madrid and was on the Faculty of the Universidad Politécnica de Madrid (UPM), where he received an award for his research productivity. He was a postdoc at MIT from 1995 to 1997. He has more than 20 years of research experience, with a steady productivity of more than 5 papers per year on average. He is Chair of the Steering Committee of DISC and has served in the TPC of numerous conferences and workshops.

Dr. Vincenzo Mancuso is Research Assistant Professor at IMDEA Networks Institute (Madrid, Spain) since September 2010. He has build his research experience by working with University of Palermo (Italy), from which he received a Ph.D. in Electronic, Computer Science and Telecommunications in 2005, Rice University (Houston, TX, USA), and INRIA Sophia Antipolis (France). His research activities focus on analysis, design, optimization and experimental evaluation of protocols and architectures for efficient wireless networks.