Exploring the trade-off between performance and energy consumption in cloud infrastructures

D. Kanapram
DITEN - University of Genoa
Via all'Opera Pia 13, 16145 Genoa, Italy
Email: divyakrisdk@gmail.com

G. Lamanna
Infocom Srl
Piazza Alessi 2, 16128 Genova, Italy
Email: guerino.lamanna@infocomgenova.it

M. Repetto
CNIT – RU of Genoa
Via all'Opera Pia 13, 16145 Genoa, Italy
Email: matteo.repetto@cnit.it

***Abstract*—Emerging fog and mobile edge computing paradigms will create distributed pervasive virtualization environments, where computing, storage, and networking resources will be deployed at the network boundary in a capillary way. To effectively tackle the large dynamic fluctuations in workload engendered by user-centric services, effective energy management schemes must be in place to modulate power consumption according to the actual processing load in each installation. In this respect, service orchestrators and multi- and cross-cloud energy management systems need proper tools to understand how power consumption would change with different placement decisions, both in the single as well as in federated clouds.**

In this paper, we describe a framework for exploring the trade-off between performance and energy consumption. Our work builds on the availability of both resource usage and power consumption measurements in Cloud Management Software, and makes proper correlation between these values to effectively support energy-efficiency strategies. We describe the implementation of energy monitoring framework in OpenStack, which leverages available features in the Ceilometer component.

I. Introduction

With the constant improvement in thermal design of data centers, electricity drawn by IT equipment remains one of the major costs for cloud operators; as a matter of fact, the Power Usage Effectiveness[1] (PUE) of modern data centers is very close to the unit [1]–[3]. Hence, to further improve efficiency of data centers focus should be given to effective management of IT equipment.

Efficient usage of IT equipment tackles the well-known non-linearity between performance and power consumption of both computing and networking devices [4]. Beside improving hardware efficiency, other mechanisms have been proposed to modulate the power consumption of single devices, including voltage and frequency scaling, low-power idle, and stand-by states [5]–[9]. In a cloud environment, where many devices are present and resources are virtualized, workload consolidation on a small cluster of servers has proven to be the most effective strategy to exploits such mechanisms, allowing to minimize the total consumption of the entire infrastructure in a coordinated way.

[1]Power Usage Effectiveness is the ratio between the total power drawn by the data center, including air conditioning, lightning, electricity distribution losses, and the power drawn by IT equipment. A PUE close to 1 indicates that energy is mostly used for computation, hence a better efficiency of the data center.

Workload consolidation is going to be even more important in distributed clouds like fog and edge computing, which envision many small deployments of IT equipment at the network edge. In fact, with the grow of user-centric services, the workload is expected to follow user distribution and mobility patterns, hence leading to uneven, unsteady, non-uniform workload among the peripheral installations. In this scenario, any orchestration and consolidation algorithm needs precise information about the trade-off between power consumption and performance in each installation, to effectively take decisions on placement and migration of Virtual Machines.

In this paper, we sketch a framework for correlating resource usage with energy consumption. It aims at providing a clear picture that draws how different allocation and placement decisions affect and impact power consumption of single servers, virtual machines, or the whole data center. This is going to be very useful in practice for workload consolidation and power management algorithms, by supplying more dynamic and precise information than current models. We also describe the current implementation effort in OpenStack, building on available information on CPU usage and integrating them with power consumption measurements.

The paper is organized as follows. We review related work in Section II. We introduce the architecture of our framework in Section III. We give more details about energy monitoring in cloud infrastructure in Section IV, which represents one one of the main aspects in our framework. We describe the current implementation stage in Section V, basically consisting in integration of power consumption information in Ceilometer. Finally, we give our conclusions and envision future actions in Section VI.

II. Related work

Workload consolidation aims at reducing the number of active servers, building on the rule of thumb that few heavy-loaded servers consume less power than many lightly-loaded servers. Many algorithms have been proposed for this purpose, which either consider the power consumption of computing resources only (e.g., [10]–[12]) or take the combination of computing and networking resources (e.g., [9], [13]–[15]). The common denominator in all approaches is a need for clear and precise understanding of how power consumption changes when the workload is shifted among the servers; as a matter

of fact, using the smallest possible number of servers does not guarantee to achieve the overall minimal energy consumption [16].

Usually, simple linear or step-wise models are used to describe the trade-off between CPU usage and power consumption, since CPU is the main source of power consumption [17]. However, these models cannot take into account the complexity brought by the presence of multiple CPUs, multiple cores, power-management features (voltage/frequency scaling, low-power idle, etc.) [17]. Further, servers from different vendors have different characteristics, and even the behavior of the same server changes with aging [18]. Statistical models have been proposed for both CPUs and network cards, which better capture the dynamic of real hardware [19]; however, they are complex to manage and to be properly tuned for different devices.

For what concerns data sources, resource usage is reported by every Cloud Management Software (CMS). In addition, extensions to OpenStack Ceilometer also allow to collect power measurements.

OpenStack already includes an IPMI agent[2], which collects sensor data (including voltage and current measurements) on individual compute nodes by leveraging the *ipmitool*[3] utility, which is commonly used for IPMI control on various Linux distributions; power consumption is listed among available Ceilometer metrics[4].

The Kwapi framework [1] implements a modular architecture to collect data from heterogeneous power meters, including legacy wattmeters, outlet power strips (usually called Power Data Unit – PDU), and IPMI cards embedded in most modern servers; the framework already includes drivers for common protocols, like SNMP, IPMI, and serial links (see Section IV).

Simple correlations between CPU utilization and power consumption was already built with the above tools, which highlighted the non-linear behavior and the irregularities due to multiple cores and power management [17]. Such work mainly aimed at validating linear models, but it has never been turned into a functional framework for exposing power-usage correlations to optimization algorithms.

III. System Architecture

The main purpose of our framework is to correlate performance and energy consumption in cloud infrastructures, hence providing more accurate information than most of currently used models. In this respect, the framework includes a *Green Cloud Abstraction Agent* (GCAA), which should be used by energy-aware orchestration and placement algorithms to understand how much energy is required to achieve given performance, and vice-versa (see Fig. 1). It provides an abstraction

[2]OpenStack IPMI agent. URL: https://docs.openstack.org/admin-guide/telemetry-data-collection.html#telemetry-ipmi-agent

[3]IPMItool provides a simple command-line interface to IPMI-enabled devices. URL: https://sourceforge.net/projects/ipmitool/

[4]IPMI based meters for Ceilometer. URL: https://docs.openstack.org/admin-guide/telemetry-measurements.html

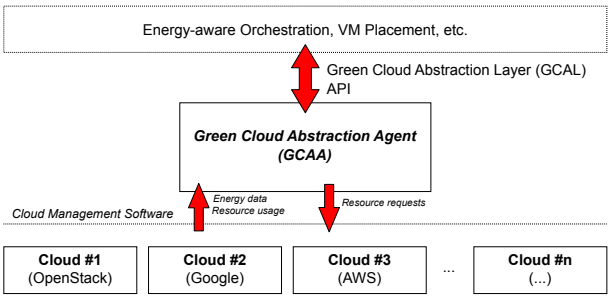

Fig. 1. Green Cloud Abstraction Agent provides an abstraction of the underlying energy behavior.

of the underlying energy profiles, primitives, and behaviors, with respect to both single or multi clouds environments; this abstraction is accessible through an API that we indicate as *Green Cloud Abstraction Layer* (GCAL). Indeed, the GCAA is not expected to expose straight power consumption information of the underlying infrastructure; rather, its main purpose is a high-level abstraction of energy aspects, including the trade-off between resource usage and energy. At the bottom, the GCAA accesses power consumption and performance metrics through legacy and enhanced interfaces of Cloud Management Software (CMS).

Our purpose is to extend the *Green Abstraction Layer* (GAL), an energy-aware interface defined by ETSI standard ES 203 237 for network devices [20]. The GAL offers a framework for information exchange between power-managed data-plane entities and control processes. It enables energy management protocols to determine consistently which power-management capabilities are available at the data-plane, their potential effects on both energy consumption and network performance, and how to interact with them. The ambition is to extend the concept to the cloud, hence to define an interface that provides a hierarchical view of the cloud's internal organization, which abstracts the power management primitives and the local control policies, so that entities at each layer of the tree manage and orchestrate the underlying entities, and expose a synthetic and aggregated set of operating characteristics and available configurations to higher levels.

Through the GCAL interface, high-level orchestrators and optimization engines would be aware whether power-management features are available in underlying cloud infrastructures, how energy consumption changes in response to different allocations of VMs and different workloads, and so on. This information could be used to make the most efficient usage of available cloud installations, as well as for advanced control features such as integration with intermittent power sources (such as renewables).

The internal architecture of the GCAA is depicted in Fig. 2 and reflects the above design considerations.

At the bottom of GCAA architecture, cloud API drivers provide interfaces to different CMS (e.g., OpenStack, Amazon Web Services). Such interface is used to retrieve data about resource usage (CPU, disk, network card, number of

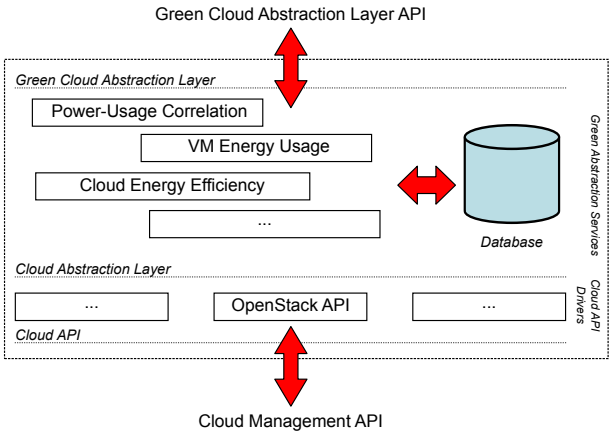

Fig. 2. GCAA architecture.

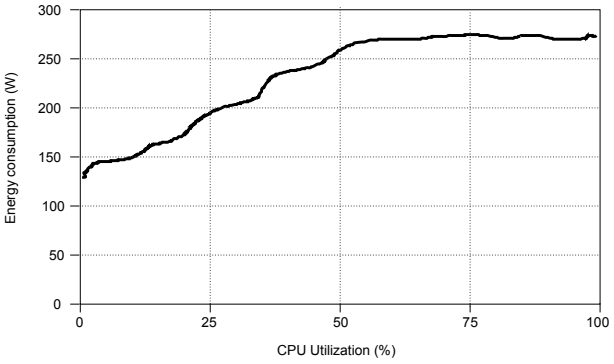

Fig. 3. Example of graphical representation of the trade-off between power consumption and CPU utilization.

VMs, placement of VMs, etc.), energy usage (voltage, current, power), and to request virtual resources (mostly, VM instances). The Cloud Abstraction Layer is an internal layer that provide a common interface to heterogeneous cloud technologies.

At the core of the GCAA architecture, several processing functions will abstract energy-related aspects of the underlying cloud infrastructures; we refer to them as "green services". For example, the "Power Usage Correlation" function will build power-to-usage curves for each physical server in the infrastructure (similar to what shown in Fig. 3) [17]; this curve could be used by energy-aware placement algorithms when evaluating the impact of migrating VMs internally to a single cloud. The "VM Energy Usage" will estimate the power consumption ascribable to virtual resources, e.g., for billing purposes [21]. "Cloud Energy Efficiency" will be responsible to provide an aggregate estimation of how the power consumption of the infrastructure changes when virtual resources are added/removed; it could be used by green orchestrator when evaluating the most efficient placement of VMs among several clouds available. Other services can be envisioned; the common database enables re-usage of information among different functions.

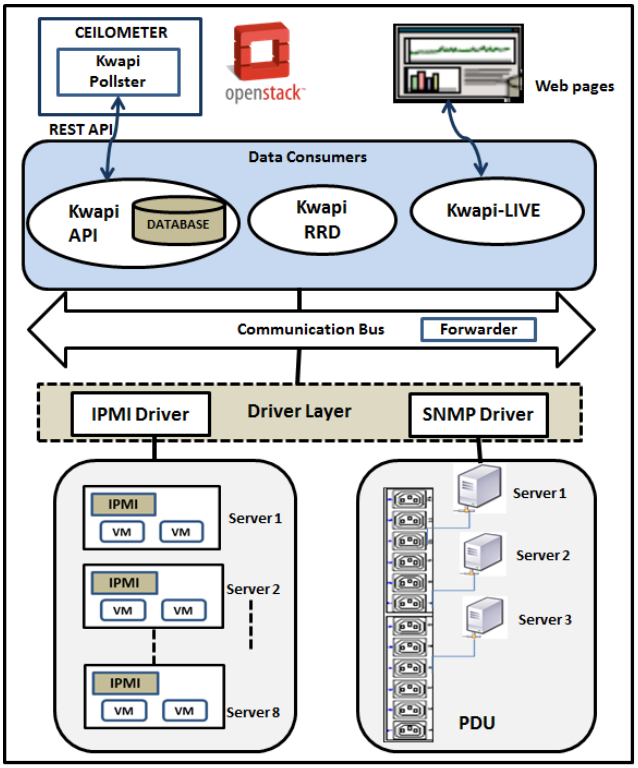

Fig. 4. Kwapi modular architecture. Drivers are used to collect data from heterogeneous power meters.

IV. Monitoring Infrastructure

One of the basic functions of the GCAA is to retrieve information about resource and energy usage. Resource usage is commonly provided by every CMS, so the challenging task is to collect data about power consumption. To this aim, we set up an energy monitoring infrastructure integrated in OpenStack.

The main objective in our framework is to collect measurements from heterogeneous components, hence retrieving data from different meters, with different communication protocols, and with different data formats. To this aim, we use Kwapi [1], a modular framework for monitoring power consumption and publishing data in Ceilometer [22], which is the well-known realization of the Telemetry service in OpenStack. Kwapi includes drivers to query commodity power meters over SNMP, serial/usb wattmeters, and the IPMI[5] interface available in many recent computing boards; this covers most hardware deployed in existing data centers, including legacy servers with a single motherboard as well as blade servers.

The Kwapi architecture[6] for energy monitoring is shown in Figure 4. At the bottom, specific drivers hidden the details of the underlying metering technology and provide a uniform

[5]The Intelligent Platform Management Interface (IPMI) provides management and monitoring capabilities of computer systems over the network, without support from the firmware or the Operating System

[6]Kwapi System Architecture. URL: http://kwapi-g5k.readthedocs.io/en/0.3-4/architecture.html.

Fig. 5. Kwapi message format for data sent on the Communication Bus.

and common interface for collecting data inside the Data Consumer, which buffers data. Data Consumers are then used by different kind of applications (e.g., Kwapi Pollster in Ceilometer, graph visualization).

A. Kwapi Drivers

Drivers are responsible for collecting power consumption data of the resources being monitored. Drivers are threads initialized by a driver manager with a set of parameters loaded from a configuration file. These parameters are used to query the meters (e.g. IP address and port number) and determine the sensor ID to be used in the collected metrics. Collected measurements are represented in JavaScript Object Notation (JSON) format, so they can be easily parsed to get the data. The role of driver threads are:

1) Setting up wattmeters and probes;
2) Listening and decoding received data;
3) Appending signatures to the measurements, and publishing them to forwarders.

Fig. 5 shows the message format that a driver publishes on the bus. We use two kinds of drivers in our testbed. The SNMP driver retrieve power consumption values by using the SNMP protocol and unique OIDs, i.e., "Object Identifiers, which are addresses used to identify devices and their statuses. These drivers can work with any device that supports the SNMP protocol. The IPMI driver query the IPMI interface for information about power consumption. We modified the original driver, which makes use of the command line tool '*ipmitool*', because it reported wrong values for our Intel server boards. We use instead the command line tool '*ipmi-oem*', a special version of ipmi software tailored to the Intel server architecture.

B. Kwapi Communication Bus

The Communication Bus implements a publish/subscribe mechanisms where drivers are publishers and consumers are subscribers. Each consumer must subscribe to drivers from which it should receive data. ZeroMQ is the current brokerless messaging framework that implements the Kwapi Communication Bus. Optional Forwarders are installed on the same servers as consumers, and replicate the same data to multiple consumers that have subscribed for them. This allows to reduce network traffic and bandwidth requirements. In practice, Forwarders behave like special consumers on the Communication Bus, in between real consumers and drivers.

C. Kwapi Data Consumers

Data Consumers collect measurements from the Communication Bus, verify the signature, and process it. There are different Data Consumers already provided by the Kwapi architecture, conceived to feed different external applications.

We make use of the REST API Data Consumer, which implement a REST-based interface to power measurements. This Consumer subscribes for data, computes the number of Kwh of each driver probe, adds a timestamp, and stores the measurement. Internally, this component has a Collector that keeps a buffer of recent measurements, including identifier, type, value, unit; data are periodically cleaned to avoid persistence of removed probes. The REST API is periodically queried by Kwapi Pollster to feed the Ceilometer database.

Other Data Consumers are available. For example, the RRD consumer builds Round-Robin Database files from received measurements, and generates graphs that show the energy consumption over a given period, with additional information about min/average/max energy consumption, cost estimations, etc. The Live consumer is a visualization plugin that provides a web interface with power consumption and network traffic graphs.

D. Kwapi Pollster

The uppermost component in the Kwapi framework is the Pollster. It is a python module developed according to a specific pattern that enables to add energy and power information in the measurement dataset exposed by Ceilometer. When this Pollster is periodically triggered by Ceilometer, it queries the Kwapi API Consumer for data in the last time interval, and returns such data for insertion into the internal database.

The Pollster provides a token for authentication, which is verified by the REST API Consumer before sending the requested data. For each probe, it creates a counter object and publishes it on the Ceilometer bus. We are mainly interested in two metrics:

- a cumulative counter for *Energy*, in kWh;
- a gauge counter for *Power*, in Watts.

E. Access to data

Data stored in Ceilometer can be accessed by standard Telemetry API. The Telemetry service provides a REST API, from which the collected samples can be retrieved, like the list of meters, alarm definitions and so on. Example: the HTTP message "*GET /v2/meters*" returns all known meters, based on the data recorded so far. The response message can be either in JSON or XML format.

In our framework, this interface will be used by the GCAA to get data on resource usage and power consumption.

V. Implementation

Till now, the implementation of the GCAA framework has mainly focused on retrieving power consumption information. We have already deployed the monitoring infrastructure, and we are currently able to retrieve power measurements from both Raritan PX3-5260R PDU (via SNMP) and Intel IPMI S2600KP blades though the Kwapi framework. The polling interval is 2 seconds, which is enough for the matter of a consolidation algorithm, though we can change this interval

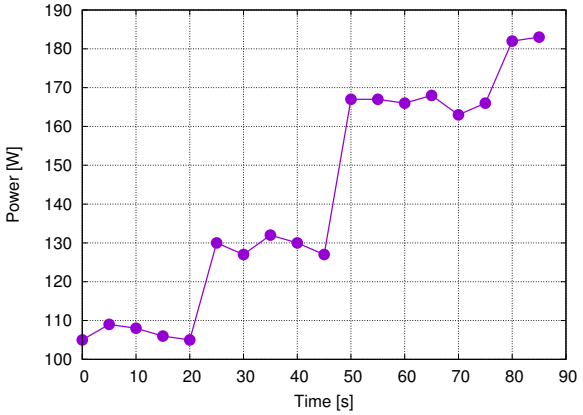

Fig. 6. Power consumption measured for increasing workload. A stress test was conducted to incrementally saturate 1, 4 and 8 CPUs out of 8 available on Intel server board S2600KP.

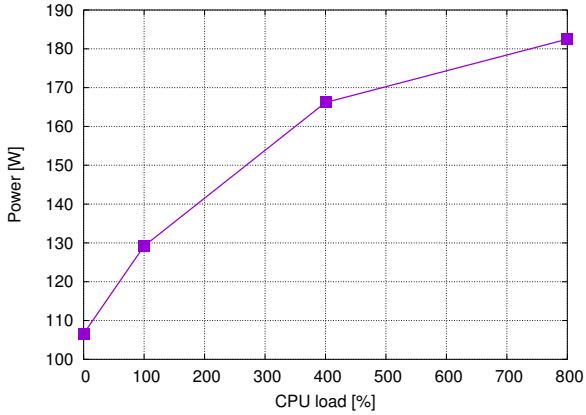

Fig. 7. Example of power-usage correlation, with a typical convex behavior. We consider 100% as full usage for a single CPU, so 800% corresponds to full load for the 8-CPUs system.

according to the requirement. An example of measurements is shown in Fig. 6; in this test, we incrementally increased the workload every 5 seconds so to saturate 1, 4, 8 CPUs out of 8 available on the board. As expected, the power consumption is a stepwise curve.

The GCAA is written in python; MySQL is going to be used as database for collecting information. Initially, we will implement a communication driver for OpenStack, to access energy-related information published by the Kwapi framework.

The first green service will be the Power-Usage Correlation, which is the main information required to drive a consolidation algorithm like the one we proposed in [9]. An indicative example of outcome is shown in Fig. 7, built on the same data used for Fig. 6. Here we get a very raw and coarse correlation between power consumption and system load. For example, we cannot infer from this data whether two half-loaded CPUs are more efficient than one single full-loaded CPU. Or we cannot understand how power consumption changes when using one or multiple cores in the same or different CPUs. The implementation of the Power-Usage Correlation service will process data in a more accurate way, building a multi-dimension view of how power consumption changes with respect to different utilization of all available cores. In addition, the usage metric should enable easy and direct comparison between servers with different CPU models and clock frequencies.

VI. Conclusions and Future Work

In this paper, we have described an architecture for high-level abstraction of energy aspects in cloud infrastructures. Such abstraction is conceived to model how different management actions (mainly in terms of different allocation in workload) affects power consumption with different granularity levels (e.g., single servers or entire clouds). To this purpose, we already envisioned a number of complementary services that are worth for green orchestration and optimization engines (e.g., power-usage correlation, cloud energy efficiency, VM energy usage).

Our next steps will be the definition of a proper model to correlate power and resource usage for single servers. The purpose is to take into account the multi-dimensionality given by the presence of multiple CPUs and cores, and their different usage patterns; this information may be used by system administrators for energy-aware workload consolidation. As a long term objective, we plan *i)* to provide energy abstraction for the whole consolidated cloud, to support consolidation algorithms that act in multi-clouds environments; and *ii)* to estimate power consumption that can be ascribed to specific software instances, which can be used for billing purposes or to select the most efficient software in mono-tenant environments, such as Network Function Virtualization domains.

Acknowledgment

This work was supported in part by the European Commission under the projects ARCADIA (contract no. 607881) and INPUT (contract no. 644672).

References

[1] F. Rossigneux, J.-P. Gelas, L. Lefèvre, and M. D. de Asunção, "A generic and extensible framework for monitoring energy consumption of openstack clouds," in *4th IEEE International Conference on Sustainable Computing and Communications (SustainCom)*, Sydney, Australia, Dec. 3–5, 2014.

[2] "Two-phase immersion cooling – a revolution in data center efficiency," Whitepaper, October 2015. [Online]. Available: http://multimedia.3m.com/mws/media/1127920O/2-phase-immersion-coolinga-revolution-in-data-center-efficiency.pdf

[3] "Energy efficiency," Open Compute Project web pages. [Online]. Available: http://www.opencompute.org/learn/energy-efficiency/

[4] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, December 2007.

[5] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in data center networks," *Computer Communications*, vol. 40, pp. 1–21, March 2014.

[6] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 2, pp. 223–244, May 2011.

[7] C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed," *International Journal of Network Management*, vol. 15, no. 5, pp. 297–310, Sep.-Oct. 2005.

[8] R. Bolla, R. Khan, and M. Repetto, "Assessing the potential for saving energy by impersonating idle networked devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, p. 16761689, March 2016.

[9] A. Carrega and M. Repetto, "Exploiting novel software development paradigms to increase the sustainability of data centers," in *Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2016)*, Shanghai, China, Dec. 6^{th}–9^{th}, 2016.

[10] A. Beloglazov and R. Buyya, "OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 5, pp. 1310–1333, April 2015.

[11] F. Ahmad and T. N. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems (ASPLOS XV)*, Pittsburgh, PA, USA, Mar.13–17, 2010, pp. 243–256.

[12] G. A. Geronimo, J. Werner, C. B. Westphall, C. M. Westphall, and L. Defenti, "Provisioning and resource allocation for green clouds," in *The Twelfth International Conference on Networks (ICN 2013)*, Seville, Spain, Jan. 27–Feb. 1, 2013.

[13] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, January 2013.

[14] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, "Joint virtual machine assignment and traffic engineering for green data center networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 3, pp. 107–112, December 2013.

[15] H. Shirayanagi, H. Yamada, and K. Kono, "Honeyguide: A VM migration-aware network topology for saving energy consumption in data center networks," in *IEEE Symposium on Computers and Communications (ISCC)*, Cappadocia, Turkey, Jul. 1–4, 2012, pp. 460–467.

[16] 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 (HotPower'08)*, San Diego, CA, USA, Dec. 7, 2008.

[17] V. Cima, B. Grazioli, S. Murphy, and T. Bohnert, "Adding energy efficiency to Openstack," in *Sustainable Internet and ICT for Sustainability (SustainIT)*, Madrid, Spain, Apr. 14–15, 2015, pp. 1–8.

[18] M. E. M. Diouri, O. Gl uck, L. Lefèvre, and J.-C. Mignot, "Your cluster is not power homogeneous: Take care when designing green schedulers!" in *2013 International Green Computing Conference (IGCC)*, Jun. 27th-29th, 2013.

[19] W. Dargie and J. Wen, "A probabilistic model for estimating the power consumption of processors and network interface cards," in *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Melbourne, Australia, Jul. 16th–18th, 2013, pp. 845 – 852.

[20] "Green Abstraction Layer (GAL); power management capabilities of the future energy telecommunication fixed network nodes," ETSI ES 203 237, March 2014, version 1.1.1.

[21] C. Gu, H. Huang, and X. Jia, "Power metering for virtual machine in cloud computing-challenges and opportunities," *IEEE Access*, vol. 2, pp. 1106–1116, September 2014.

[22] "Openstack telemetry," web site. [Online]. Available: https://wiki.openstack.org/wiki/Telemetry