Journal of Cloud Computing: Advances, Systems and Applications

SpringerOpen

This Provisional PDF corresponds to the article as it appeared upon acceptance. Fully formatted PDF and full text (HTML) versions will be made available soon.

Cloud Computing and Its Interest in Saving Energy: the Use Case of a Private Cloud

Journal of Cloud Computing: Advances, Systems and Applications 2012, **1**:5 doi:10.1186/2192-113X-1-5

Robert Basmadjian (robert.basmadjian@uni-passau.de)
Hermann De Meer (demeer@uni-passau.de)
Ricardo Lent (r.lent@imperial.ac.uk)
Giovanni Giuliani (Giuliani@hp.com)

ISSN 2192-113X

Article type Research

Submission date 20 January 2012

Acceptance date 18 May 2012

Publication date 8 June 2012

Article URL http://www.journalofcloudcomputing.com/content/1/1/5

Cloud computing and its interest in saving energy: the use case of a private cloud

Robert Basmadjian[1*]
Hermann De Meer[1]
Ricardo Lent[2]
Giovanni Giuliani[3]

[1]University of Passau, Innstrasse 43, 94032 Passau, Germany
[2]Department of Electrical and Electronic Engineering, Imperial College, London SW7 2BT, UK
[3]HP Italy Innovation Center, Milano, Italy

Email: Robert Basmadjian - Robert.Basmadjian@uni-passau.de;
Hermann De Meer - demeer@uni-passau.de;
Ricardo Lent - r.lent@imperial.ac.uk;
Giovanni Giuliani - Giuliani@hp.com;

*Corresponding author

Abstract

Background

Cloud computing data centres, due to their housing of powerful ICT equipment, are high energy consumers and therefore accountable for large quantities of emissions. Therefore, energy saving strategies applicable to such data centres are a very promising research direction both from the economical and environmental stand point.

Results

In this paper, we study the case of private cloud computing environments from the perspective of energy saving incentives. However, the proposed approach can also be applied to any computing style: cloud (both public and private), traditional and supercomputing. To this end, we provide a generic conceptual description for ICT resources of a data centre and identify their corresponding energy-related attributes. Furthermore, we give power consumption prediction models for servers, storage devices and network equipment. We show that by applying appropriate energy optimization policies guided through accurate power consumption prediction models, it is possible to save about 20% of energy consumption when typical single-site private cloud data centres are considered.

Conclusion

Minimizing the data centre's energy consumption, on one hand acknowledges the potential of ICT for saving energy across many segments of the economy, on the other hand helps ICT sector to show the way for the rest of the economy by reducing its own carbon footprint. In this paper, we show that it is possible to save energy by studying the case of a single-site private cloud data centres. We believe that through the federation of several cloud data centres (both private and public), it is possible to minimize both the energy consumption as well as CO_2 emissions.

Keywords

Private cloud computing data centre, Modelling, IT resources, Power and energy consumption

Background

Motivation

Energy consumption of data centres is becoming a key concern for their owners: *energy costs* (fuel) continue to rise and *CO_2 emissions* related to this consumption are relevant. In 2007, Gartner [1] estimated that the global impact of the ICT sector (considering PCs, servers, cooling, fixed and mobile telephony, local area network, office telecommunications and printers) is 2% of the global CO_2 emissions, which is approximately the same as fuel consumption from the airline industry.

Amazon [2] has evaluated its data centre expenses (see Figure 1), showing that server costs account for 53%, while energy related costs totaling 42% (direct power consumption 19% plus amortized power and cooling infrastructure 23%).

Figure 1 Amazon's monthly expenses distribution. Amazon's evaluation of its data centre expenses showing that server costs account for 53%, while energy related costs totaling 42%

Therefore, saving money in the energy budget of a data centre, without sacrificing Service Level Agreements (SLA) is an excellent incentive for data centre owners, and would at the same time be a great success for *environmental sustainability*. This aspect needs to be highlighted, since it's absolutely not frequent in environment-related problems to have a solution satisfying all stake holders.

Main focus

One of the latest trends in IT is cloud computing [3], where *public* and *private* deployment models [4] are commonly used to differentiate one cloud provider from another. The former is made available to the general public on pay-per-use basis, whereas the latter is operated solely for internal users of organizations.

Since *elasticity* [4] is one of the key aspects of a cloud service, the cloud provider gives the impression of having an infinite set of resources. In fact, large public cloud providers have end users potentially spread over the globe and can capitalize on statistical compensation for large number of service requests; practically speaking, since cloud resources are frequently allocated and also released (as side effect of the pay-per-use billing mode), during the time lots of allocations and de-allocations tend to compensate keeping fluctuations under reasonable control. On the other hand, private clouds, instead, have generally a much smaller number of users (e.g. the employees of a corporation) and the provider needs to size its ICT infrastructure for the peak usage. Since the variety of usage patterns is limited inside a close community, it's quite likely that the utilization rate of the ICT resources can vary a lot between night and day and/or week days and week-ends. Therefore, private cloud providers need to keep a larger capacity buffer compared to public providers, and suffer more from load variations. In addition, it has been noticed that some public cloud providers are starting to offer discounted prices for certain time slots, when they foresee a resource usage gap – something similar to the concept of "last minute"tickets for travelers – where private providers don't have this chance!

Given the above-mentioned differences in the ICT resources' utilization patterns between private and public cloud providers, it is obvious that more evident advantages are foreseen to save energy in former than in latter where the owners –in case of low load –instead of saving energy and costs, can decide to attract additional business by lowering the prices to fill the utilization gap. Private cloud provider –in case of low load –has the possibility to see the whole infrastructure optimized to run the load with the lowest energy consumption, while preserving the SLAs with respect to their users.

In summary, there are lots of incentives for any cloud provider – being public or private – to save energy. The opportunities for public cloud providers might depend on alternative business conducts, while private cloud providers will likely get clear benefits.

Contributions and results

In general, energy savings can be achieved in data centres through optimization mechanisms whose main objective is to minimize the energy consumption. However, in order that these optimization mechanisms can take the most suitable energy-saving decisions, the existence of accurate power consumption prediction models becomes primordial. Consequently, one of the major contributions of this paper is devoted to a detailed description of power consumption prediction models for ICT resources of data centres which are presented in Section Power consumption prediction models. Note that the details on the optimization algorithms used for our use-case study are out of the scope of this paper and interested readers can refer to [5].

The architecture of our energy-saving mechanism is based on a three-step control loop: Optimization, Reconfiguration and Monitoring. The whole *state* of the data centre is continuously *monitored*. As a matter of fact, another major contribution of this paper is dedicated to a detailed description of the state of a data centre in terms of ICT resources with their relevant energy-related attributes and interconnections which is introduced in Section Data centre schema. This state is periodically examined by the *Optimization* module in order to find alternative software application and services deployment configurations that allow saving energy. Once a suitable energy-saving configuration is detected, the loop is closed by issuing a set of actions on the data centre to reconfigure the deployment to this energy-saving setup.

Monitoring and *Reconfiguration* modules in Figure 2 interact with the data centre monitoring and automation frameworks to perform their tasks. The *Optimization* module ranks the candidate target configurations, identified by applying energy-saving policies without violating existing SLAs, with respect to their power consumption that are predicted by the *Power Calculator* module. The accuracy of the predictions of this module is crucial to take the most appropriate energy-saving decisions; in fact it has the responsibility to forecast the power consumption of a data centre *after* a possible reconfiguration option. In the rest of this paper, our energy-saving mechanism is called "plug-in" simply because it runs agnostically on top of data centres' automation and monitoring frameworks.

Figure 2 Proposed control loop for energy optimization. The different components and their interactions of our energy-aware "plug-in". We call it "plug-in" simply because such a set of software components run agnostically on top of data centre automation and monitoring frameworks

Finally, we demonstrate the results obtained through single-site optimization policies of a private cloud computing data centre. In Section Evaluation and results these results are presented which illustrate the possibility of saving about 20% on the total consumption of ICT equipment inside a private cloud computing data centre. It is worthwhile to note that our power consumption prediction models as well as state description of a data centre are generic enough that they are suitable to energy-saving optimization mechanisms applicable to any computing style being traditional, super and cloud computing.

Related work

Servers

In the existing literature, the power consumption of a server has been modelled in two different ways: *offline* and *online*. In the former case, SimplePower [6], SoftwareWatt [7] and Mambo [8] estimate the power consumption of an entire server. However, these models use analytical methods based on some low-level information such as number of used CPU cycles. The major advantage is that they provide high accuracy. Nevertheless, the offline nature of such models requires extensive simulation, which results in a significant amount of time for estimating the power consumption. Consequently, these models are infeasible for predicting the power consumption of highly dynamic environments like cloud computing data centres.

To overcome this problem, an online (run-time) methodology is proposed [9, 10]. Such models are based on the information monitored through performance counters [11]. These counters keep track of activities performed by applications such as amount of accesses (e.g. to caches) and switching activities within processors. The total power dissipation of a server is computed as the power consumption of each activity. However, these counters in certain processors (e.g. AMD Opteron) can report only four out of 84 events [12]. Therefore, such models are unable to predict accurate power consumption in real-life cases.

Another run-time methodology is to use high-level information as the one proposed by [13]. These authors assumed that processors are the main contributors to the total server's power consumption. Thus, a linear model based on the processor's utilization is proposed. However, such a model suffers from a significant inaccuracy as the server's power consumption is not exactly linear [14]. The key reason is that the impact of other components (e.g. multiple level

caches, RAM, I/O activities) and their interactions are not considered. To prevent this problem, [10] designed a component level model. Through this approach, a calibration phase is performed before predicting the power consumption of a server. During this phase, this model analyzes the system parameters (e.g. CPU utilization, hard disk I/O rate) influencing on its power consumption. However, implementing such a model within a data centre (having homogenous and/or heterogenous resources) is very difficult since it needs calibration whenever a new hardware is installed within the existing servers. Since the component level is flexible for modelling a generic server, we have also adopted the same approach as [10]. In contrast to this model, which provides one linear model for the whole servers, our approach designs different models for different components based on their behaviors. Another distinguishing property is that our approach does not need calibration phase.

It is worthwhile to note that above mentioned approaches, which depend on low-level information in order to predict the power consumption, are not appropriate in real-life case simply because the underlying monitoring systems of the data centres are not able to provide the low-level information that these approaches require. Consequently, in this paper we identified the most-relevant energy-related attributes of ICT resources to which the monitoring systems of data centres typically provide the necessary information.

Storage devices

The storage devices range from a single hard disk to SAN (Storage Area Network) devices, which consume a significant amount of power. Several studies [15–18] were dedicated to devise models for individual hard disks, where the power consumption of a disk is predicted based on its states such as *seek*, *rotation*, *reads*, *writes* and *idle*. With the emergence of Disk arrays (e.g. RAID), the above mentioned approaches need to be adapted to deal with several hard disks instead of one. Several models [19–21] for RAID have been proposed. [19] proposed STAMP (Storage Modeling for Power) by mapping the front-end workload to back-end disk activities, and the power consumption of the back-end activities is computed. A major concern is that this model ignores the power of some activities such as spin-up and spin-down [21]. [20] generalized the model of individual hard disks to RAID where the overall power of RAID is the sum of power consumed by each individual hard disk within RAID. However, this model does not take into account the power of RAID controller's processor and memory. [21] addressed this problem by proposing a model (MIND), which computes the power of RAID by considering controller's power and disk activities such as idle, sleep, random, sequential and cache accesses.

The key drawback of these RAID models is their implementation in the real world such as data centres due to the low level input parameters. For instance, hard disk's state (e.g. idle, accessing, startup) and operation mode (read/write operations) are usually not available at data centre level, where a monitoring system provide information only about average read and write rates over a time period (e.g. per second). Furthermore, the total power consumption of SAN devices, which are used for storage within data centres, cannot be computed through RAID models since SAN has also processing (e.g. CPU, RAM) and network components (I/O ports) in addition to hard disks. In this paper, we propose a model, which can be applied to any type of storage devices. In contrast to the existing models, our approach requires information such as read/write rate, which is usually available within data centres monitoring systems.

Network equipment

There has been a great deal of attention to devise power consumption models for routers and switch fabrics. The power consumption of the integrated Alpha 21364 router and the IBM InfiniBand 8-port 12X switch is modelled in [22] illustrating that buffers are the largest power hoggers in routers. A crossbar switch, if present, consumes less but still significant power, and arbiter power is negligible under high network load. It is shown, that also the router micro-architecture has a huge impact on router power consumption. [23] suggest a framework to estimate the power consumption on switch fabrics in network routers. They state that the power consumption on switch fabrics is mainly determined by the internal node switches, the internal buffer queues and the interconnect wires. [24] present a power and area model according to network-on-chip architectures. They suggest ORION 2.0 that uses a set of architectural power models for network routers to perform early-stage network-on-chip power estimation.

Data centre schema

The ICT resources (e.g. servers, storage devices and network equipment) with their relevant energy-related attributes and interconnections are represented by the *data centre schema*. The data centre operator identifies all the equipment that the site is composed of and becomes responsible for creating and editing an instance of such a schema. Note that some forms of automated discovery of resources and exporting them to the *schema instance* might be provided, however the data centre operators are responsible for validating their configuration.

Given the complexity and heterogeneity of the data centre infrastructure, we derive the *data centre schema* by decomposing the modelling process into 5 phases: ICT resources, network topology, server, storage, and services modelling.

ICT resources modelling

Figure 3 illustrates the UML class diagram of ICT resources for data centres. Each “Site” can host a set of “Data centre”s where sites can be scattered geographically. The main energy-related attribute of a “Site” is *PUE* [25] which stands for Power Usage Effectiveness and is used to indicate the efficiency in terms of energy usage of the data centre facilities. This metric is used for evaluating the impact of software load (e.g. virtual machines) on global energy consumption. The attribute *CUE* (Carbon Usage Effectiveness) can be used to perform optimisations based on CO_2 emissions (out of the scope of this paper). Each “Data centre” is based on a specific computing style indicated through the *computingStyle* attribute which belongs to one of the following three categories: Traditional computing, Supercomputing and Cloud computing. Note that the corresponding attribute is useful in order to enable the optimization policies suitable to each computing style.

Figure 3 ICT resources UML class diagram. The figure, which is a Unified Modeling Language design, describes in detail the different ICT resources that can be found inside a site consisting of several data centres

---

Inside each data centre, there are ICT equipment which can be organized either inside “Rack”s or in independent cases such as single box stands, generally with tower form factor

(“Tower Server”), in addition to box-like network devices such as routers and switches (“Box Network”class). “Framework Capabilities” class describes energy-related controlling capabilities of the management and automation tools available to the data centre to be managed energy-wise. The term controlling capabilities refers to all possible actions (e.g. power off/on equipment, migrate software load, etc.) applied to the data centre ICT resources and carried through by the framework. As a matter of fact, in the rest of this paper *frameworkID* attribute is used by every class that needs to identify its corresponding controlling capabilities.

A rack is a framework used typically to hold several different ICT resources: rack-mountable servers are represented by the “Rackable Server”class, enclosures to host blade form factor servers (“Blade Server”class) are depicted by the “Enclosure”class. Typically “Tower Server”s and “Rackable Server”s have independent Power Supply Units (“PSU”class) and cooling system (“Cooling System”class), whereas “Blade Server”shares the PSU and cooling system from its enclosure. Note that all these three classes representing servers are generalizations of a single parent “Server”class (see Section Server modelling). Furthermore, racks contain typically a set of Power Distribution Units (“PDU”class): in most cases they are passive devices simply used to connect the different power plugs of the rack elements; in some other cases they can also be active and perform power measurements and switch on/off functions. Storage Area Network devices (“SAN”class) are generally mounted inside racks and have independent Power Supply Units and cooling system. SAN is a dedicated device that provides network access to consolidated, block level storage. Finally, network devices such as routers and switches can also be mounted inside racks (“Rackable Network” class) whose specifics are explained in Section Network topology modelling. Note that in the rest of this paper *measuredPower* and *computedPower* optional attributes in most of the classes are used in order to record the power consumption of the corresponding resource obtained respectively through a dedicated power meter and power consumption prediction models of Section Power consumption prediction models. Note that these two parameters serve for the model refinement as they are useful to compare measured and computed values to check and refine the power consumption prediction models. Finally, *powerIdle* and *powerMax* are respectively used to denote the idle and the fully active power consumption of the corresponding class.

Network topology modelling

Figure 4 illustrates the network topology UML class diagram which depicts a clearer overview of how network nodes in a data centre are connected to each other. A class “Network”consists of number of entities of “Network Node”and “Flow”. Flows are allocated to the various nodes based on the network decisions. These allocations are dynamic and can vary during the lifetime of a flow to optimise the energy consumption accordingly. Therefore, the two defined entities of “Network Node”and “Flow”shape the “Network”so that based on certain criteria a “Flow”can be detached from one “Network Node”and can be attached to another. The energy efficient policies that make such decisions are performed based on the attributes from both classes of “Flow”and “Network Node”, which will be elaborated further in the followings.

Figure 4 Network topology UML class diagram. The figure, which is a Unified Modeling Language design, provides a detailed description of how network nodes in a data centre are connected to each other

As illustrated, the “Flow”class describes an end-to-end communication occurring within the data centre or between one network node within the data centre and another node on an external network (for example the Internet). This class’ attributes include the communication end points, which can be expressed by the *source* and *destination Network Node* addresses. The “Flow”class also includes an attribute *bandwidth* required by a communication which provides an indication of the expected flow throughput for traffic engineering purposes.

A “Network Node”is an abstract class, which represents the entities defined as routers, switches, servers, and so on. Each “Network Node”can have a number of communication ports (“Network Port”class), however a port can only be associated to a single “Network Node”. Moreover, each network port is associated to a class “Link”that connects this port to another port in the network. The class “Network Node”possesses of the following attributes: *processingBandwidth* refers to the maximum number of packets that can be processed by the node per second. The processing bandwidth can be usually found in router’s/switches specification data sheets or can be measured. *forwardFlag* indicates whether the node is able to forward packets and is used to differentiate end hosts from routers and switches. Note that each “Network Node” is equipped with Power Supply Units (“PSU” class) and cooling system (“Cooling System” class) such as fans whose energy-related attributed are presented in Section Server modelling.

The class “Network Port”defines a port on the “Network Node”, which can be any of the variants as depicted in Figure 4: e.g. Serial PPP, VPN, ATM, e80211x, Ethernet, Optical FDDI and Tunnel. The most relevant energy-related attributes of the “Network Port” class are the followings: *lineCapacity* denotes the nominal transmission rate of the port (typical values are 10 Mbps for Ethernet, 100 Mbps for Fast Ethernet, and 1 Gbps for Gigabit Ethernet), whereas *portID* provides a unique identifier to the port. *powerMax*, *powerIdle* and *lineCapacity* are used to capture the power consumption behaviour of the port. *trafficIn* and *trafficOut* represent respectively the packet throughput in and out of the port. *bufferSize* and *bufferOccupancy* describe all together buffer characteristics and management policies in use within the port that are needed to estimate QoS metrics. *trafficIn* and *trafficOut* are required to compute the actual power consumption based on the power consumption model. *bufferSize*, and *bufferOccupancy* are used to compute the delay for the traffic forwarding to the corresponding port. Finally, the “Link” class models the propagation medium associated with its corresponding “Network Port”. Its attributes are the *propagationDelay*, which defines the time required to physically move a bit from two end points, and the *bitErrorRate*.

Server modelling

Figure 5 depicts the Server UML class diagram where “Server”class represents an abstraction for a generic server computing system, such that the different specializations used in *data centre schema* (“Tower Server”, “Rackable Server”, and “Blade Server”classes) are distinguished by their physical form factor.

Figure 5 Server UML class diagram. The figure, which is a Unified Modeling Language design, illustrates in detail the different components such as processors, mainboard, memories, hard disks that a server can be composed of

---

Typically, a server consists of a “Mainboard”and runs several software applications such as “Native Operating System” and “Native Hypervisor” (see Section Services modelling). The mainboard is the central printed circuit board in server computing system and holds many of the crucial components of the system, while providing connectors for other peripherals. Its *memoryUsage* attribute (counterpart of free space) denotes the overall usage (in GB) of all the attached memory modules whose value should be kept up-to-date through the data centre’s monitoring system. The followings are the main components attached to the “Mainboard”: Central Processing Units (“CPU” class), Random Access Memories (“RAMStick”), Network Interface Cards (“NICs”), hardware RAIDs (“HardwareRAID”) and Storage Units (“StorageUnit”).

With the advent of modern processors, a “CPU”consists of more than one “Core” where each such core can have its own “Cache” depending on the *level* (e.g. Level 1). Furthermore, it is also possible that certain cores of a processor share the same cache (e.g Level 3). The most relevant energy-related attributes of “CPU” are: *architecture* indicates the processor’s manufacturer (e.g. Intel, AMD, etc.) each having different power consumption behaviour, *cpuUsage* denotes the utilization (load) of the processor whose value should be kept up-to-date through the data centre’s monitoring system. *DVFS* (Dynamic Voltage and Frequency Scaling) is an attribute used to indicate whether the corresponding server’s energy-saving mechanisms (e.g. Intel SpeedStep, AMD Cool’n’Quiet, etc.) is enabled or not. *lithography* and *transistorNumber* denote respectively the size in nanometres as well as the number of transistors (in the order of millions) of the processor which are used for idle power consumption prediction purposes.

Each “Core”operates on a *frequency* (in GHz), scaled dynamically[a] between minimum and maximum ones (*frequencyMin* and *frequencyMax*), and *voltage. coreLoad* represents the utilization of the corresponding core whose value should be kept up-to-date through the data centre’s monitoring system. *totalPstate* and *lastPstate* indicate respectively the total number of P-states (e.g. 2 or more) as well as the most recent P-state of the core (e.g. P_0, P_1, etc.). Note that these two parameters are used to estimate the *frequency* of a server’s processor core for data centres whose monitoring system can not provide up-to-date dynamic values for this attribute. The implementation details of guessing the frequency is not covered in this paper.

The “RAMStick”class has several attributes relevant to power consumption estimation: *voltage* reflects the supply voltage under which the memory module operates which is highly dependent on the *type* (e.g. DDR_1, DDR_2, DDR_3, etc.), whereas *size* and *frequency* indicate respectively the size (in GB) and frequency (in MHz) of the memory, *vendor* denotes the manufacturer (e.g. KINGSTON, HYNIX, etc), *bufferType* shows the type of the memory module in terms of buffer technology (e.g. fully buffered, buffered, registered, and unbuffered). It is worthwhile to mention that values of all the above-mentioned attributes are provided by the manufacturer’s data sheet.

Several “Storage Unit”s can be attached to a “Server”either directly through its “Mainboard”or by means of a dedicated “Hardware RAID”device. Additional information regarding storage modelling is provided in Section Storage modelling. As mentioned previously, “Tower Server”s and “Rackable Server”s are equipped with their own Power Supply Units (“PSU”class) and cooling systems (“Cooling System”) which can be either a “Water Cooler”or a “Fan”. The followings are the most relevant energy-related attributes of a

"PSU": *efficiency* (in percentage) indicates the amount of loss of the power supplied to the components of the server, which is highly related to the *load*. Note that values of efficiency for the corresponding loads can be extracted from the manufacturer's data sheet.

Finally, inside the "Fan"class, *depth* denotes the depth (in meter) of a fan, whereas *maxRPM* and *powerMax* indicate respectively the maximum rotations per minute and power consumption of the fan. All the above-mentioned attributes can be found inside manufacturer's manuals. *actualRPM* shows the current rotation speed of the fan whose value should be kept up-to-date through the data centre's monitoring system.

Storage modelling

In this section, we present the storage modelling both for the case of storage units attached directly to servers as well as the case of Storage Area Network (SAN) devices. A generic UML class diagram for these cases is introduced in Figure 6.

Figure 6 Storage UML class diagram. The figure, which is a Unified Modeling Language design, provides an overview of the way storage devices (e.g. hard disks) as well as SAN devices can be connected to servers

Server storage

Left part of the UML class diagram of Figure 6 illustrates the server storage modelling where the "Storage Unit"class represents the abstraction for all kinds of disk-like devices providing the physical storage for data. "Storage Unit"s can be directly connected to a "Server"through its "Mainboard"or by means of a "Hardware RAID"controller that provides the different *levels* of RAID support to servers. We consider both traditional disks with revolving platters ("Hard Disk"class) and solid state disk ("Solid State Disk") as possible "Storage Unit"devices. Note that attributes for "Server" and "Mainboard" classes are described in Section Server modelling.

"Storage Unit"class energy-related attributes are the followings: *maxReadRate* and *maxWriteRate* indicate respectively the maximum read and write rates of the disk which are computed in terms of the transferred size per second (MB/s). The values for above-mentioned attributes can be extracted from the manufacturer's data sheet. *readRate* and *writeRate* indicate respectively the actual read and write rates of the disk which are expressed in terms of MB/s as mentioned previously. The values for both of these attributes should be kept up-to-date through the data centre's monitoring system.

Each "Hard Disk"has the following different energy-related attributes: *rpm* indicates the rotation per minute of the disk, *platters* denotes the number of platters, whereas *AAM* presents whether the hard disk is equipped with Automatic Acoustic Adjustment feature. For the "Solid State Disk", *powerByRead* and *powerByWrite* denote respectively the power consumed by read and write operations. The distinction here is due to the fact that read and write operations in solid state disks have different power consumption behaviour.

"Storage Unit"s can also be attached logically to a "Server". Such a functionality is provided by means of "Logical Unit"s abstraction of "SAN" devices called "LUN", whose details are

covered in Section SAN storage. Each “LUN” class has the following attributes: *LUNRef* is used to reference the corresponding logical unit of a SAN device, whereas *readRate* and *writeRate* have the same definition as for the case of “Storage Unit” class.

SAN storage

A Storage Area Network (SAN) is a dedicated device that provides network access to consolidated, block level storage. SAN architectures are alternative to storing data on disks attached to servers or storing data on Network Attached Storage (NAS) devices connected through general purpose networks - which are using file-based protocols.

Right part of Figure 6 illustrates the SAN devices UML class diagram. Typically, a “SAN” device consists of more than one (usually two) “PSU” and “Fan” for redundancy purposes, several fiber channel (FC) and Ethernet Network Interface Cards (“FiberchannelNIC”and “EthernetNIC” classes) and a set of “Storage Unit”s. Furthermore, “Storage Unit”s are logically consolidated through “Logical Unit”s abstraction such that a “Storage Unit” is a member of one and only one “Logical Unit”. “Server”s access a “Logical Unit” through a unique logical unit number reference (“LUN” class).

Each “SAN” class has the following energy-relevant attributes: *networkTrafficIn* and *networkTrafficOut* have the same definition as their counterparts *trafficIn* and *trafficOut* of Section Network topology modelling. On the other hand, *RAIDLevel* of “Logical Unit” class shows the level (e.g. RAID 0, 1, 5, 10, etc.) of the RAID being used with the corresponding logical unit. As a matter of fact, each logical unit can be considered as a separate RAID controller. Furthermore, *stripeSize* shows the size of the RAID protocol’s stripe. *numberOfRead* and *numberOfWrite* denote respectively the number of read and write operations performed per second. All the other attributes of “Logical Unit” class have the same definition as their counterparts of the “Storage Unit” class of Section Server storage.

Services modelling

By the term *services* we mean any type of software applications which directly or indirectly generate load on ICT resources of a data centre. Consequently, all software components, either at application or system level, independently from the computing style (e.g. traditional, supercomputing or cloud computing) of the data centre, can be modelled as load generator whose UML class diagram is introduced in Figure 7. Note that such a model is necessary for power optimization algorithms that seek to distribute and potentially move software application load (e.g. virtual machine) to the computing resources which consume less power and still satisfy the required Service Level Agreements (SLA).

Figure 7 Services UML class diagram. The figure, which is a Unified Modeling Language design, shows the different software components (e.g. applications, operating systems, virtual machines, etc) that can run on physical servers

---

Physical servers (“Server”class) execute the software structured and layered as depicted in Figure 7. Above the hardware level, at start-up a physical server bootstraps either a traditional operating system (“Native Operating System”) or a virtualization hypervisor (“Native

Hypervisor"). Some virtualization hypervisors need to run above an operating system ("Hosted Hypervisor").

Both "Native Hypervisor" and "Hosted Hypervisor" can run "Virtual Machine"s, which implement a software equivalent environment of a hardware server (with some limitations). The actual power consumption of the "Virtual Machine"increases with the increase of required number of processing resources such as *numberOfCPUs*, *actualCPUUsage* (load imposed on the processor(s)), *actualStorageUsage* (size in GB), *actualDiskIORate* (MB/s), *actualMemoryUsage* (size in GB) and *actualNetworkUsage* (packets/s or MB/s) that is being used. Note that the values of these attributes should be kept up-to-date through the data centre's monitoring system. The *resourceClassID* is used to point out to the required resources of the virtual machine in terms of CPU, memory, disk I/O, etc. This information is useful for the power optimization algorithms to ensure SLA about resource utilization is not violated when moving the VM from one host to another. The *frameworkID* is used in order to discover the appropriate framework actions (e.g. migrate, pause, resume) applicable to the virtual machines. In cloud computing environment, each virtual machine typically belongs to a specific type which is identified by a unique name (e.g. m1.small, m1.medium, c1.large, etc). Each such type specifies the required resources of the virtual machine in terms of CPU, memory and disk. *cloudVmType* is used to identify the appropriate predefined virtual machine types whereas *cloudVmImage* indicates the installed operating system.

"Virtual machine"s typically boot a "Hosted Operating System", which might contain – depending on the case – specialized drivers to operate on virtualized devices.

The UML class diagram of Figure 7 describes "Operating System"class as a generalization for a traditional native or hosted operating systems (OS), and for native hypervisors: they all share the "boot on "relation with respect to a physical server or a virtual machine. *systemRAMBaseUsage* indicates the amount of memory allocated by the operating system. In addition, an operating system in general contains multiple "File System" types), might support - inside the kernel - the software implementation of raid features ("Software RAID") and the software implementation of network devices, for example virtual switches ("Software Network"). The power consumption of "File System"increases with the increase of *fragmentation* factor and with the decrease of the *free* available size with respect to the total capacity (*size*) as the disk needs more power to locate the position of the data. An important energy-related attribute for "Software Network"is *switchFabricType* as this indicates the type of the network device the software is emulating (switch or router).

The typical software packages ("Software Application" class) can run either on a "Native Operating System" and/or on a "Hosted Operating System": some applications don't have problems in any execution environment, while others are not able to run in virtualized mode, thus the distinction in the model. The actual power consumption of the "software application "increases with the increase of number of processing resources such as *NumberOfCPUs*, *actualCPUUsage* (load imposed on the processor), *actualStorageUsage* (size in GB), *actualDiskIORate* (MB/s), *actualMemoryUsage* (size in GB) and *actualNetworkUsage* (packets/s or MB/s) that the application is using. Note that the values of these attributes should be kept up-to-date through the data centre's monitoring system.

Power consumption prediction models

In this section, we introduce the power consumption prediction models of the most relevant ICT resources of data centres such as servers, storage devices and networking equipment (e.g. routers or switches). Note that such power consumption models are the cornerstone of energy optimization algorithms by providing them with detailed insights regarding the power consumption of the aforementioned ICT resources in different workload deployment configurations.

Server

The power consumption of a server is broken down into two parts: *idle* and *dynamic*. The former is computed while the server is idle with no activities, whereas the latter is calculated when the server is performing certain computations. As a matter of fact, it is necessary to model both aspects for different components of a server illustrated in Figure 5, in order to have a deeper understanding on the power consumption.

Processor

It was shown in [13] that processors are the most prominent contributors (about 40%) to the overall power consumption of servers. Furthermore, the power consumption of processors can be due to either its idle state (no utilization) or dynamic state while carrying out certain computations. With the advent of multi-core processors (e.g. dual-core, quad-core, etc.) as well as their corresponding energy-efficient mechanisms (e.g. Intel SpeedStep, AMD Cool'n'Quiet), several techniques (e.g. Dynamic Voltage and Frequency Scaling - DVFS) were introduced that save energy especially when the processor is in idle or low utilization states.

Idle power consumption

The idle power consumption of a processor can be determined by using the following well known equation [26] derived from Joule's and Ohm's laws:

$$P = I * V, \tag{1}$$

where P denotes the power (Watt), I represents the electric current (Ampere) and V indicates the voltage. Equation (1) can be adopted in order to compute the idle power consumption at core level by assuming that each core contributes equally to the overall idle power consumption of a processor:

$$P_i = I_i * V_i, \tag{2}$$

where P_i, I_i and V_i denote respectively the power, current and voltage of the corresponding core i. Furthermore, by analyzing the current I_i and voltage V_i relationship, we derive the following second order polynomial (using the curve-fitting methodology) to model the current leakage:

$$I_i = \alpha V_i^2 - \beta V_i + \gamma, \tag{3}$$

where $\alpha = 0.114312$ $((V\Omega)^{-1})$, $\beta = 0.22835$ (Ω^{-1}) and $\gamma = 0.139204$ $(\frac{V}{\Omega})$ are the coefficients such that V and Ω denote respectively the voltage and resistance. It is worthwhile to note that these values are derived based on results obtained from a power meter [27] while analyzing a quad-core processor with energy-efficient mechanisms (e.g. DVFS) deactivated.

With the emergence of energy saving mechanisms (e.g. Intel SpeedStep, AMD Cool'n'Quiet), the idle power consumption of a core (processor) decreases. This is achieved by decreasing the voltage and frequency (DVFS) of a core. In order to demonstrate such an impact, we propose the following model:

$$P_{r_i} = \delta_i P_i, \tag{4}$$

where δ_i is the factor for reduction in the power consumption P_i of core i (see Equation (2)), whereas P_{r_i} represents the reduced power consumption of core i. It is worth pointing out that δ_i can vary depending upon the corresponding energy saving mechanisms where each of such mechanism has its own particular features (the detailed modelling of δ_i is out of the scope of this paper). To this end, in this paper we provide values of δ_i for multi-core Intel processors in Table 1. Certain processors, for instance, Intel dual- and quad-core processors do not possess C-states. Hence, the energy reduction factor for processors (e.g. hexa-core) having such states is significantly different from the others.

Table 1 Values of δ_i for Intel Processors

Processor type	δ_i
Intel Xeon dual-core E5502	0.942
Intel Xeon quad-core E5540	0.728
Intel Xeon hexa-core X5650	0.316

The values of the reduction factor δ_i for different types of Intel processors

Given a processor of n cores with a specific energy-saving mechanism, then its idle power consumption is given by:

$$P_{CPU_{idle}} = \sum_{i=1}^{n} P_{r_i}, \tag{5}$$

where P_{r_i} is introduced in Equation (4).

Dynamic power consumption

The power consumption of the i^{th} core of a multi-core processor due to performing certain computations is given by:

$$P_i^{'} = P_{max} \frac{L_i}{100}, \tag{6}$$

where $P_i^{'}$ denotes the dynamic power consumption of the core i having a utilization (load) of L_i, whereas P_{max} indicates the maximum power consumption due to 100% utilization. It is worthwhile to note that Equation (6) is derived based on the well known linear utilization-based model of [13] for single-core processors.

The maximum power consumption P_{max} is computed by adopting the following well known CMOS [28] circuits power consumption equation:

$$P_{max} = V_{max}^2 * f_{max} * C_{eff}, \tag{7}$$

where V_{max} and f_{max} denote respectively the voltage and frequency at maximum utilization, whereas C_{eff} [28] indicates the effective capacitance which includes the capacitance C and switching activity factor $\alpha_{0\rightarrow 1}$.

Given a processor of n cores with no specific energy-saving mechanisms enabled, then its dynamic power consumption is given by:

$$P_{CPU_{dynamic}} = \sum_{i=1}^{n} P_i^{'}, \tag{8}$$

where $P_i^{'}$ is introduced in Equation (6).

In fact, there are certain factors which play a major role in reducing the overall power consumption of multi-core processors of Equation (8). Among those, the followings are two main techniques through which the power consumption of multi-core processors can be reduced: Energy saving mechanisms such as Intel SpeedStep and AMD Cool'n'Quiet decrease power utilization of a core by controlling its clock speed and voltage dynamically. In the idle mode or when the utilization (load) of a core (processor) is low, the clock speed is reduced to minimize its power dissipation. Resource sharing (e.g. L2-cache in case of certain Intel multi-core processors) reduces the power consumption. We believe that this assumption is true due to the fact that sharing L2-cache with other cores minimizes the cache miss ratio. As a consequence, less communication takes place with the memory (e.g. to fetch new instructions), which contributes in reducing the total power consumption of cores.

Due to these power reduction mechanisms, the power consumption of a multi-core processor is always less than the one of Equation (8). In order to cope with this overestimation, we introduce a power reduction factor (δ') to Equation (8) in the following manner:

$$P_{CPU_{dynamic}} = \delta'(\sum_{i=1}^{n} P_i^{'}), \tag{9}$$

where the detailed modelling of δ' is not covered in this paper. However, it is worthwhile to mention the following facts related to the modelling of δ':

1. The reduction factor δ' changes by modifying the frequency of the processor.
2. The number of active (a utilization of more than 1%) cores of the processor has an impact on the reduction of power consumption.

Table 2 illustrates a sample value of δ' for an Intel quad-core processor having different number of 100% loaded cores as well as clock frequencies ranging from 2.0 to 2.5 GHz. As we can notice, the higher the frequency is, the possibility of reducing power becomes more evident. Furthermore, more power is saved as the number of active cores increases. These are due to the fact that the pure summation of the power consumption of the cores given by Equation (8) is always greater than the total actual power consumption. As a matter of fact, this difference becomes important for increasing frequencies and number of active (100% loaded) cores. Hence the reduction factor δ' becomes more obvious for those cases.

Table 2 Values of δ' based on Frequency and Number of 100% Loaded Cores

Processor type	F[GHz]	Voltage	δ'		
			2 cores	3 cores	4 cores
Intel Xeon quad-core E5540	2.0	1.104	0.94	0.93	0.92
	2.5	1.104	0.70	0.71	0.70

The values of the reduction factor δ' for different frequencies of Intel quad-core processor. The reduction factor also changes based on fully loaded cores

Total power consumption

The total power consumption of a multi-core processor is defined by:

$$P_{CPU} = P_{CPU_{idle}} + P_{CPU_{dynamic}}, \tag{10}$$

where $P_{CPU_{idle}}$ and $P_{CPU_{dynamic}}$ are introduced in Equations (5) and (8) respectively.

Memory

A Random Access Memory (RAM) consumes power when it is idle refreshing the ranks holding certain stored data as well as while accessing the memory ranks to perform either read or write operations. In this paper, we focus on Synchronous Dynamic RAM DDR_3 technology due to the fact that most modern data centres' (including our real-world testbed) servers are equipped with such type of memory modules.

With respect to the idle power consumption of DDR_3 memory, we derive a model starting from the following well known equation [26] :

$$P = I * V, \tag{11}$$

where P denotes the power (Watt), I represents the electric current (Ampere) and V indicates the voltage. On the other hand, it was shown in [29] that there is a linear relationship between the current I and voltage V when the supplied voltage is between 0 and 2V (which is typically the case for DDR_3 memory). Hence the current can be expressed in the following manner:

$$I = c * V, \tag{12}$$

where based on our observations, we noticed that c takes a value of 13×10^{-5}. Taking Equations (11) and (12) into account, the power consumption of a DDR_3 memory module for a given frequency f (in MHz) and size s (in GB) can be rewritten in the following way:

$$P(f, s) = c * V^2. \tag{13}$$

Consequently, in order to reflect the impact of frequency on the idle power consumption, Equation (13) can be written as:

$$P(s) = f * c * V^2. \tag{14}$$

Furthermore, in order to show the influence of size (in GB) on the idle power consumption, Equation (14) can be written as:

$$P = s * f * c * V^2. \tag{15}$$

Given a set of n DDR_3 memory modules, then their idle power consumption is expressed as:

$$P_{RAM_{idle}} = \sum_{i=1}^{n} s_i * f_i * c * V_i^2, \tag{16}$$

where s_i, f_i and V_i denote respectively the *size* (in GB), frequency (in MHz) and the voltage (volts) of a specific DDR_3 memory module i, whereas c takes a value of 0.00013.
Concerning the dynamic power due to accessing the memory, there is always only one active operating rank per channel regardless of the number of memory modules or module ranks in the system. As a matter of fact, such a power is always constant during access and independent of the operation type (read or write) as well as size and is given by:

$$P_{RAM_{dynamic}} = 9.5 \text{ Watt}, \tag{17}$$

Then based on Equations (16) and (17), the overall power consumption of a Random Access Memory is given by:

$$P_{RAM} = P_{RAM_{idle}} + \gamma * P_{RAM_{dynamic}}, \tag{18}$$

where $\gamma \in [0, 1]$ whose details are covered next. Since certain monitoring systems (including the one of our real-world testbed) can not provide information about how often the memory modules are accessed, then we adopted the following technique to derive values for γ:

1. If the processor is in idle state performing no activity, then we assume that the memory modules are also in idle state ($\gamma = 0$).
2. If the processor is carrying out certain computations (utilization of more than 1 %), then we adopt a probabilistic approach in modelling γ, such that the more total memory is in use, the higher in probability that a memory access is performed:

$$\gamma = \frac{memoryUsage}{\sum_{i=1}^{n} s_i}, \tag{19}$$

where n and s_i are defined in Equation (16) and *memoryUsage* is introduced in Section Server modelling.

Hard disk

Typically, the power consumption of a hard disk can be broken down into three major parts: *startup*, *idle*, and *accessing* modes, where each such mode has different power consumption behaviour. The disk is in startup mode when all of its mechanical and electrical components are activated. On the other hand, the disk is in idle mode when no activity (read or write) is carried out, whereas it is in accessing mode while performing read or write operations.

Based on our observations performed on different families of hard disks, we noticed that the idle mode power consumption can be further split into three states: *idle*, *standby* and *sleep*. Moreover, we noticed that the power consumption due to standby and sleep states is quite identical and it is in average 10% of the idle state power consumption. This is due to the fact that during standby and sleep states, the disk's mechanical parts are significantly shutdown. Then, the idle mode power consumption of the hard disk is given by:

$$P_{HDD\_idle} = P_{idle}(\alpha + 0.2 * \beta), \tag{20}$$

such that $\alpha \in [0, 1]$ indicates the probability that the disk is in idle state, $\beta \in [0, 1]$ denotes the probability that the disk is in standby and sleep states (the values of α and β are given later such that $\alpha + \beta = 1$), whereas P_{idle} is the idle state power consumption provided by the manufacturer's data sheet. Furthermore, we observed that the startup and accessing modes' power consumption is respectively in average 3.7 and 1.4 times more than that of the idle state power consumption. Then, the power consumption of the hard disk is given by:

$$P_{HDD} = x * 1.4 * P_{idle} + y * P_{HDD\_idle} + z * 3.7 * P_{idle}, \tag{21}$$

such that $x, y, z \in [0, 1]$ denote respectively the probability that the disk is in accessing, idle and startup modes, whereas P_{idle} is the idle state power consumption provided by the manufacturer's data sheet.

Since certain monitoring systems (including the one of our real-world testbed) can not provide information about when each hard disk switches from startup, to idle and accessing modes, then we adopted the following two techniques in order to derive values for x, y and z such that $x + y + z = 1$:

1. If the average operation size (MB/s) of reads and writes per second is zero ($readRate = writeRate = 0$), then we assume that the disk is in its idle mode ($x = z = 0$ and $y = 1$).

2. If the average operation size (MB/s) of reads and writes per second is not zero, then we adopt a probabilistic approach in modelling the mode changes such that:

   - If $readRate > 0$ and $writeRate > 0$, then $x = \frac{readRate + writeRate}{maxReadRate + maxWriteRate}$,
   - If $writeRate = 0$, then $x = \frac{readRate}{maxReadRate}$,
   - If $readRate = 0$, then $x = \frac{writeRate}{maxWriteRate}$,

Note that *readRate*, *maxReadRate*, *writeRate* and *maxWriteRate* are introduced in Section Storage modelling, whereas $y = 0.9 * (1 - x)$ and $z = 0.1 * (1 - x)$. Finally, in order to derive values of $\alpha, \beta \in [0, 1]$ for the idle mode power consumption, we adopted the following probabilistic approach:

1. If $0 < y \leq 0.3$, then we set $\alpha = 0.9$ and $\beta = \frac{0.1}{2}$.

2. If $0.3 < y \leq 0.6$, then we set $\alpha = 0.5$ and $\beta = \frac{0.5}{2}$.

3. If $0.6 < y \leq 1$, then we set $\alpha = 0.1$ and $\beta = \frac{0.9}{2}$.

We can notice from the above equations that the more the hard disk is in idle mode ($y \simeq 1$), the higher is the probability that it will remain in standby and sleep states.

Network interface card

Network interfaces, which connect a server to one or more networks, normally operate at a fixed line rate and add both physical and link layer functionalities that contribute to increase the total power consumption of a server. At any given time, a network interface will be either in idle mode or actively transmitting or receiving packets. If $P_{NIC_{idle}}$ is the power of the idle

interface and $P_{NIC_{dynamic}}$ is the power when active either (or both) receiving or transmitting packets, the total energy consumption of an interface will be given by:

$$E_{NIC} = P_{NIC_{idle}} T_{idle} + P_{NIC_{dynamic}} T_{dynamic}, \tag{22}$$

where T_{idle} is the total idle time and $T_{dynamic}$ denotes the total active time in an observation period $T = T_{idle} + T_{dynamic}$, such that T_{idle} and $T_{dynamic} > 0$. The average power P_{NIC} during period T is:

$$P_{NIC} = \frac{(T - T_{dynamic})P_{NIC_{idle}} + P_{NIC_{dynamic}} T_{dynamic}}{T} = P_{NIC_{idle}} + (P_{NIC_{dynamic}} - P_{NIC_{idle}})\rho \tag{23}$$

where $\rho = \frac{T_{dynamic}}{T}$ is the channel utilization (also known as the normalized link's load). Both time periods and power values would depend on the particular network technology employed.

It is interesting to note that the choice of network technology could affect, to varying degrees, the utilization of other computer system components and in particular processor (CPU). For example, in serial point-to-point (PPP) communications, the CPU is normally used to execute a significant number of communication-related operations (e.g., frame checking and protocol control). These operations can easily increase the dynamic power consumption of the CPU. On the other hand, embedded network implementations, such as InfiniBand, can move much of the communication work to the embedded architecture. To include this network-technology dependent behaviour into our model, consider parameter L_i (CPU load) in Equation (6) as resulting from two components: $L_i = L_i^{'} + \gamma\rho$, where $L_i^{'}$ and $\gamma\rho$ correspond to non-network and network dependent CPU load respectively. Parameter γ ($\gamma \geq 0$) models the impact of a given network technology on CPU load based on network utilization ρ. Small γ values can account for the minimal impact that embedded network architectures could cause to CPU load whereas larger γ values could be used to model the higher CPU dependency of other network interfaces.

Mainboard

The aggregated power consumption of the mainboard consists that of its constituent components attached to it and is given by the following equation:

$$P_{Mainboard} = \sum_{i=1}^{l} P_{CPU} + P_{RAM} + \sum_{j=1}^{m} P_{NIC} + \sum_{k=1}^{n} P_{HDD} + c, \tag{24}$$

where P_{CPU}, P_{RAM}, P_{NIC}, and P_{HDD} are given respectively by Equations (10), (18), (23), and (21), whereas c is constant related to the mainboard's own power consumption. Note that technically it is challenging to compute the power consumption of the mainboard. Hence, statistical values for c can be derived based on the server type (e.g. tower, rackable, and blade), which is reflected by means of *powerIdle* and *powerMax* attributes of the "Mainboard" class in Section Server modelling.

Fan

The power consumption of a fan changes from one Rotation Per Minute (RPM) to another: i.e. the higher the RPM is, the more power it consumes. Consequently, a model is derived

starting from the following well known formula[b] for the power consumption of a fan:

$$P = d_p * q, \tag{25}$$

where P denotes the power consumption (Watt), d_p indicates the total pressure increase in the fan (Pa or N/m^2), and q represents the air volume flow delivered by the fan (m^3/s). Hence, replacing d_p by $\frac{F}{A}$ and q by $\frac{V}{t}$ in Equation (25), we obtain:

$$P = \frac{F}{A} * \frac{V}{t}, \tag{26}$$

where F, A, V and t denote respectively the force (N), area (m^2), volume (m^3) and time (seconds). By a simple simplification of volume V and area A, we obtain the following equation:

$$P = \frac{F * d}{t}, \tag{27}$$

where F denotes the force (N), d indicates the depth of the fan (m) and t represents the time (seconds). Based on our observations performed on a set of fans, we found out that F is proportional to the square of the RPM:

$$F = c * RPM^2. \tag{28}$$

By taking into account Equations (27) and (28), the power consumption model for the fan is given by:

$$P_{Fan} = \frac{c * RPM^2 * d}{3600}. \tag{29}$$

where RPM denotes the actual rotation per minute of the fan (*actualRPM* in Section Server modelling) whose value should be kept up-to-date through the monitoring system. It is worthwhile to note that for a given fan, the value of c remains constant. As a matter of fact, we compute the value of c based on Equation (29):

$$c = \frac{3600 * P_{max}}{RPM_{max}^2 * d}, \tag{30}$$

where P_{max} and RPM_{max} denote respectively the maximum power and rotations per minute of the fan whose values can be extracted, in addition to the depth d, from its manufacturer's data sheet.

Power supply unit

As its name indicates, PSU supplies power to the numerous components of a server. In general, the power consumed inside the PSU itself (loss) is highly dependent on its efficiency: the higher the PSU is in efficiency, the less power it consumes. To this end, the PSU manufacturers provide the efficiency range with respect to a given PSU load. Hence, we compute the power consumption of a PSU having an *efficiency* of e, in the following manner:

1. If the data centre's monitoring system provides information at the PSU level (*measuredPower* of "PSU" class in Section Data centre schema), then the power

consumption is given by the following equation:

$$P_{PSU} = \frac{measuredPower * (100 - e)}{100}.$$

2. If the data centre's monitoring system provides information only at the server level (*measuredPower* of "Server" class in Section Data centre schema), then we assume that this *measured power* of the server is evenly distributed among its n PSUs (having similar efficiency) providing power to the components, and compute the power consumption by the following equation:

$$P_{PSU} = \frac{(\frac{measuredPower}{n}) * (100 - e)}{100}.$$

3. If the data centre's monitoring system provides no information neither at the server level nor at the PSU level, then we compute the power consumption by the following equation:

$$P_{PSU} = (\frac{P_{Mainboard} + P_{Fan}}{n * e}) * 100 - (\frac{P_{Mainboard} + P_{Fan}}{n}),$$

such that $P_{Mainboard}$ and P_{Fan} are introduced in Equations (24) and (29) respectively, whereas n denotes the number of PSUs and e their efficiency (assuming that its identical for all the installed PSUs).

Server power consumption

Given a server composed of a mainboard, several fans and power supply units as illustrated in Figure 5, then we compute its power consumption in the following manner:

1. If the server is of type *Blade*, then its power consumption is given by the following equation:

$$P_{Blade} = P_{Mainboard}. \tag{31}$$

2. If the server is of type *Tower* or *Rackable*, then its power consumption is given by the following equation:

$$P_{Tower\_Rackable} = P_{Mainboard} + \sum_{i=1}^{l} P_{Fan} + \sum_{j=1}^{m} P_{PSU}, \tag{32}$$

such that $P_{Mainboard}$, P_{Fan} and P_{PSU} are respectively given by Equations (24), (29), and Section Power supply unit.

SAN devices

Given a SAN device whose UML class diagram is depicted in Figure 6, then its power consumption is presented by the following equation:

$$P_{SAN} = P_{SAN_{idle}} + P_{SAN_{dynamic}}, \tag{33}$$

such that $P_{SAN_{idle}}$ and $P_{SAN_{dynamic}}$ denote respectively the idle (no activity) and utilization dependent power consumptions. Moreover, the idle power consumption of the SAN devices is given by:

$$P_{SAN_{idle}} = \sum_{i=1}^{n} P_{HDD\_idle} + \sum_{j=1}^{m} P_{ENIC_{idle}} + \sum_{k=1}^{l} P_{FCNIC_{idle}} + c, \tag{34}$$

where n denotes the total number of installed hard disks whose idle power consumption is given by Equation (20), m and l indicate respectively the total number of Ethernet Network Interface Cards (NIC) and Fiber Channel NICs having an idle power of $P_{ENIC_{idle}}$ and $P_{FCNIC_{idle}}$ given by manufacturer's data sheet, whereas c is a constant value representing the idle power consumption due to the mainboard and its attached components other than those mentioned above. Statistical values for c can be configured by *powerIdle* attribute of the SAN devices introduced in Section SAN storage. It is worthwhile to note that most of the real-life cases, such SAN devices rarely go to sleep or standby modes. As a matter of fact, we set $\alpha = 1$ and $\beta = 0$ for Equation (20).

On the other hand, the dynamic power consumption of a SAN device is as follows:

$$P_{SAN_{dynamic}} = \sum_{o=1}^{r} P_{LU(o)} + \sum_{p=1}^{s} P_{ENIC_{dynamic}} + \sum_{q=1}^{t} P_{FCNIC_{dynamic}}, \tag{35}$$

where p and q denote respectively the total number of Ethernet and Fiber Channel NICs whose dynamic power consumption is given by Equation (23), whereas o indicates the total number of Logical Units of a SAN device whose dynamic power consumption is:

$$P_{LU(i)} = \left(\frac{NbR_i}{NbR_i + NbW_i}\right) \sum_{i=1}^{r_i} P_{HDD} + \left(\frac{NbW_i}{NbR_i + NbW_i}\right) \sum_{j=1}^{w_i} P_{HDD}, \tag{36}$$

such that NbR_i and NbW_i denote respectively the total number of read and write operations performed per second which are represented by *numberOfRead* and *numberOfWrite* attributes of "Logical Unit" class in Section SAN storage, whereas P_{HDD} is the power consumption of the corresponding hard disk introduced in Equation (21). Since it is not possible for monitoring systems of data centres to provide accurate information regarding whether the last performed operation is read or write, then we adopt a probabilistic approach in Equation (36) by using the number of read and write operations performed per second in order to guess which operation has more dominance. Note that such a guess is important for RAID protocols since the number of involved hard disks differ from read to write.

In order to compute the number of involved hard disks r_i due to performing a RAID protocol read operation at Logical Unit i, we apply the following technique:

$$|R_i| = \frac{\lambda_i}{NbR_i}, \tag{37}$$

where λ_i denotes the read rate (MB/s) of the Logical Unit i represented by *readRate* attribute of Section Server storage and $|R_i|$ indicates the average read operation size (MB/read) per

second. Based on Equation (37), the number of involved hard disks is given by:

$$r_i = \frac{|R_i|}{stripeSize}, \tag{38}$$

where *stripeSize* denotes the size of the RAID protocol's stripes specified by its level. The same methodology can be used in order to compute the number of involved hard disks w_i for write operations. It is important to note that the different involved hard disks for a given operation are picked up randomly since this is purely RAID level's protocol dependent. As a matter of fact, we assume that the hard disks attached to a given Logical Unit i have similar power consumption behaviour.

Network equipment

Given that the power consumption P_{NEQ} of a network equipment will be mainly driven by the internal switching state and buffer utilization, it is sensible to assume that its overall power consumption will be linked to its total packet switching throughput (λ'):

$$P_{NEQ} = \gamma' + \Phi(\lambda'), \tag{39}$$

where γ' represents the power consumed by the equipment without workload and $\Phi(.)$ a function that determines the level of power used by a given packet switching throughput. The exact form of function $\Phi(.)$ will be determined by the implementation technology. In practical terms, the contribution of $\Phi(.)$ to the total power consumption of a regular (embedded) network equipment is expected to be small. However, it is also expected that this trend will change in the future as "greener"implementations are introduced.

Evaluation and results

Testbed environment and configuration

The testbed[c] under investigation provides a computational environment implementing cloud computing for Infrastructure as a Service (IaaS) platform. It is the most basic cloud service typology, where virtual infrastructure resources (e.g. CPU, memory, storage devices) are provided to users on dynamic and scalable basis. It is worthwhile to note that the testbed is based upon a Lab-grade infrastructure fully resembling (in a smaller scale) both the configuration and functional capabilities of actual production-grade IaaS implementations, being private or public.

Hardware configuration

The hardware equipment consist of two racks each hosting an *HP Blade System C3000* [30] enclosure where equivalent ISS (Industry Standard Server) blade servers are mounted inside. In addition to the four free slots, each enclosure bears 4 blade servers (a total of 8 in the testbed) belonging to *HP ProLiant BL460c G6* [31] series which are half-height blades, configured as in Table 3.

Table 3 Servers' Hardware Configuration

Processor	Dual CPU, quad-core Intel Xeon E5540, 2.53 GHz 8 MB L3 cache
Memory	24 GB (6 x 4 GB DIMMs) DDR_3
Hard Disk	Two hot plug hard drives 2 x 300 GB
Network	Dual-port 10 gigabit Ethernet adapter NC532m

The hardware characteristics of the servers for the corresponding testbed regarding

Inside each enclosure, the servers are interconnected through an *HP Virtual Connect* Ethernet Module. The two racks, in turn, are interconnected through an external Ethernet switch. Power supply is bundled with the enclosure, through 6 high efficiency (90%) 1200W *HP Common-Slot* Power Supply Units. Cooling is provided by 6 *HP Active Cool 100* fan units, also directly installed in the enclosures.

Finally, energy measurement is performed by an HP hardware component named *iLO* (Integrated Lights-Out), accessible through the *Insight Control* software suite. *iLO* offers the ability to read real-time electrical power consumption down to single server level.

Software configuration

The testbed's architecture consists of the following five software components:

1. The Cloud Controller (CC),
2. The Node Controller (NC),
3. Energy-aware plug-in,
4. The Power and Monitoring Collector (PMC),
5. The client (end user) system.

The Cloud Controller (CC) is the component hosting the core cloud management functions, i.e. it's the application server (Front End) where the cloud web services actually reside. These services are triggered by any end user request, asking to activate or deactivate a set of computational resources, identifiable as virtual machines. Furthermore, the CC software is deployed onto a physical server, as typically done to duly keep under control the response time to client requests. This software runs on a Red Hat Enterprise Linux (RHEL) 5.5 operating system instance.

The five Node Controllers (NC) are the physical machines providing virtualized environment to cloud platform clients. They are the physical servers on which virtual machines are created and instantiated by the CC, initialized with the software image selected by the client within the service catalog, and finally made exclusively accessible to the requesting client for its own usage through the network. The instantiated virtual machines can be de-instantiated upon clients' request containing a "terminate instance "command, and their used resources are released by the CC. Both events (instantiation and de-instantiation of a virtual machine) are captured by the energy-aware plug-in which on its turn triggers certain optimization algorithms to minimize the energy consumption of the testbed. The Node Controller software, like the Cloud Controller (CC), runs on a RHEL 5.5 operating system instance. The virtual

machines instantiated in response to client requests are created and deployed by a XEN hypervisor, and typically host Linux images (e.g. Ubuntu, Suse, Red Hat, etc.).

The energy-aware plug-in (described briefly in Section Contributions and results) resides altogether on a dedicated virtual machine, running on the VMware ESX 4.0 hypervisor. The Power and Monitoring Collector (PMC) is implemented by a customized version of *collectd*. *Collectd* is an open source Linux daemon able to collect, transfer and store performance data of computers and network equipment. For our testbed, specific *collectd* agents have been developed and implemented, to interface with *iLo* and acquire power measurement data. Like the Cloud Controller, even the PMC is deployed on a physical server.

Finally, the client systems are emulated by a custom software tool, generating sequence of requests that faithfully replay the interaction among a group of observed users from a real life context and the cloud IaaS infrastructure. The client load simulation is deployed inside virtual machines running an Ubuntu image, whose execution is scheduled and coordinated by a custom component running on the same VMware node where the energy-aware plug-in is also deployed.

Testing methodology

A cloud computing IaaS load is by definition fairly unpredictable in the sense that its instantaneous computational load fluctuates arbitrarily between zero and maximum available capacity of the physical resources. This unpredictability in load is due to accommodating requests coming from a group of users, without being constrained into a static planning of the infrastructural capacity. To this regard, finding a suitable testing methodology is challenging due to the lack of upfront clue on the actual usage pattern of the environment.

To overcome this methodological problem, the activities (over a period of 6 months) of a real cloud computing IaaS environment were traced and the system parameters of each active physical and virtual resource were monitored. Then from these observations, a collection of repeating test sequences and usage patterns were extracted that altogether provides an exhaustive representation of system states worth experimenting our optimization algorithms and measuring their actual results. To this end, a custom workload simulator was designed and developed. This tool can generate a sequence of actions and direct them to the Cloud Controller (CC), creating the required workload snapshots in order to enact energy-aware optimization algorithms and measure the achieved results with the best significance.
The testbed is equipped with a data logger component, storing all the details of the energy-aware plug-in activities, along with the measured energy and the corresponding timestamps. After the end of the proof of concept, the log files were extracted from the system, and carefully analyzed to take out of them a perceptible track of the actual user activities performed and logged. As a final outcome of this hindsight, we obtained crisp and content-relevant activity profiles of 7 different usage patterns. The chosen profiles span a sufficient timeframe and content to get a significant variance of activity profiles, and a sufficient amount of dynamical context changes (new activities and tasks, high load versus night timeframes). These profiles serve as the basis for designing and implementing the workload patterns enacted by the simulator tool whose details are covered next.

Workload generation

Based on the observations of the real case user activity profiles as explained in previous section, a basic set of activity types is identified typically replicable on a weekly basis. The analysis elucidated the existence of overall three basic activity aggregation types, detectable in different profiles per user and per virtual machine:

1. Steady tasks: spawn a virtual machine, and keep it running for a medium-long period of time, with a basically constant level of resource usage (e.g. CPU, memory, storage device); typical cases were complex software application development tasks.

2. Spiky tasks: spawn a virtual machine, intensively use it for a short term period (e.g. a quick debug on an application), then suddenly release it to the IaaS environment.

3. Rippling tasks: spawn a virtual machine, and keep it running for a medium-long period of time, with a fairly variant pattern of resource usage; this typology can be associated, for instance, to data management/reporting activities, or to some particular tasks ran in collaboration.

After identifying the above-mentioned three basic activity types, the next step was to configure the workload simulation tool in order to generate a realistic sequence of system actions (create and de-instantiate virtual machines) able to replicate as faithful as possible these recovered patterns.

A snapshot of a workload profile is shown in Figure 8. It is worthwhile to mention the fact that time schedules have been squeezed into a 7:1 or 7:2 compression factor. As a matter of fact, all the tests of one week long are performed in a single day or 2 days time slot, ensuring the full execution of the test plan. Before going full speed with the test campaign, we ran a single-spot, full week test on a selected sample profile, followed by the time squeezed test on the same profile, to make sure that the time compression didn't bring up any bias or alteration to the observed system behaviour.

Figure 8 An example of workload profile on a weekly basis. The figure illustrates the workload profile of our experiments performed in a single day or 2 days time slot

Numerical results

Power consumption predictions

Before performing our tests related to the energy optimization, it was necessary to validate the accuracy of the power consumption prediction models of Section Power consumption prediction models. To this end, we carried out observations both for the idle and dynamic power consumptions of the blade servers whose hardware configuration is presented in Table 3.

Idle power consumption predictions

Figure 9 illustrates the power consumption of the blade servers both obtained from our power monitoring tool *iLo* and our power consumption prediction models (PCM). We can notice that both have identical power consumption which is due to the fact that we were able to configure the mainboard power consumption (see *powerIdle* in Section Server modelling) appropriately. In Table 4, we present the idle power consumption breakdown on the component-basis.

Figure 9 Idle power consumption of blade servers. The figure shows the idle power consumption of blade servers obtained from a power monitoring tool (iLo) and developed power consumption models (PCM)

Table 4 Idle Power Consumption Prediction Breakdown

Component	**Consumption** (Watt)
Processors	33 Watt
Memories	14 Watt
Hard Disks	3 Watt
Mainboard	70 Watt
Total	120 Watt

This table indicates the idle power consumption of the investigated server expressed in terms of its constituent components

Dynamic power consumption predictions

In order to better understand the power consumption behaviour of our blade servers under different load patterns, we performed tests (1) by explicitly fixing the frequency of the processor to its maximum and (2) by letting the operating system to configure it automatically (e.g. on-demand governor) based on certain OS-related mechanisms ensuring performance and energy efficiency (e.g. Intel SpeedStep). To identify the trend of the power consumption for a server, we adopted the *lookbusy*[d] software tool, which allows to generate synthetic workload on a server in a tractable way, based on a wide set of parameters (e.g. CPU usage, memory usage, IO operations, etc.). The methodology followed for testing a server was the following:

1. Set the server's power management policy (e.g. performance, on-demand, etc.).
2. Measure power consumption from *iLO* with CPU in idle state.
3. While the CPU utilization is less than 100%:
   (a) Increment by 20% the workload on the server.
   (b) Wait for a 10-minute period, to let the server and the power metering system to reach a stable situation.
   (c) Measure the power consumption.

Such a measurement was repeated while simulating also memory usage with the *lookbusy* tool, to assess the impact of memory usage on the power consumption.

Figure 10 illustrates the power consumption of the blade servers with dynamic setting (e.g. on-demand governor) both obtained from the power monitoring tool *iLo* and our power consumption prediction models (PCM). The horizontal axis represents the load percentage (utilization) of all the cores: i.e. for a quad-core processor, a load of 20% reflects the fact that all the four cores are 20% loaded. We can notice that both iLo and PCM have quite identical results. However between 60% and 80% load, PCM suffers from an error of at most 13%. We believe that this error is due to inaccuracy of the information provided by the monitoring system of our testbed concerning the frequency. In order to validate our argument, another set of similar observations was performed by fixing the frequency of the processor to the

maximum whose results are demonstrated in Figure 11. It is clear in this figure that if the appropriate frequency and voltage parameters are provided, then the power consumption of *"iLo"* and "PCM" is almost identical.

Figure 10 Power consumption of blade servers with dynamic setting. The figure demonstrates the power consumption of blade servers obtained from a power monitoring tool (iLo) and developed power consumption models (PCM), by increasing the load of the processors by increments of 20%. The frequency of the processors are configured dynamically by means of the operating system in order to achieve the required performance

Figure 11 Power consumption of blade servers with maximum frequency setting The figure demonstrates the power consumption of blade servers obtained from a power monitoring tool (iLo) and developed power consumption models (PCM), by increasing the load of the processors by increments of 20%. The frequency of the processors are set to its maximum manually through the BIOS

Energy optimization tests

The actual energy optimization results were achieved by applying, the same workload of Section Workload generation in identical conditions, to the testbed once without energy-aware plug-in and the second time with the plug-in up and running. Figure 12 illustrates the number of active virtual machine instances tracked across a 24-hour timeframe, which is a compressed image of a 7-day pattern, generated by the execution of the workload profile as described in Section Workload generation. We can notice from Figure 13 that the energy consumption of the testbed is more flat than the corresponding instance curve of Figure 12. This is due to the fact that no energy optimization policies of any type were applied to the testbed. On the other hand, we can identify in Figure 14 the clear difference with respect to the energy consumption when no energy optimization policies are applied. More precisely, we can observe the substantial rippling and fluctuations which follows up the variations of workload. Also, at any specific instance, the testbed's power consumption does not bypass 1 KWatt whereas it is clear that in the case of no energy-aware plug-in, the testbed's average power and energy consumption is more than 1 KWatt.

Figure 12 Number of virtual machine instances. The figure shows the total number of virtual machine (VM) instances over the time of 24 hours test, representing a 7 days load pattern, compressed with a factor of 7 into a single full day

Figure 13 Power measures of the testbed without energy-aware plug-in. The figure illustrates the total overall power consumption of the testbed executing the workload without energy-aware plug-in

Figure 14 Power measures of the testbed with energy-aware plug-in. The figure demonstrates the total overall power consumption of the testbed executing the workload with our energy-aware plug-in

Finally, in order to ensure that the energy-aware plug-in[e], whose main task is to minimize the energy consumption, itself does not consume significant amount of power, we computed its consumption. It turns out that the plug-in itself consumes 3% (6 Watt out of 187 Watt) of the overall power of the server running this VM. Thus, we can conclude that the energy saving achieved through our energy-aware plug-in is not offset by side effects due to its own induced additional consumption. Figures 15 and 16 summarizes the energy as well as average power consumption of the testbed with and without energy-aware plug-in. We can notice in Figure 16 that the least savings (16.27%) can be achieved in the beginning days of the week whereas the most savings (19.76%) can be realized among the end days of the week (weekends). Thus Figure 15 represents the total global savings (17.98%) that is accomplished through our energy-aware plug-in. Note that the results presented in this section are obtained by taking the average of five independent observations (run of the workload) with a confidence interval of 99% such that the power measurement unit has itself an accuracy of 1%.

Figure 15 The global overview of the testbed results. The figure provides a general comparison (in KWh), of the total energy consumption of the testbed, between with and without energy aware plug-in

Figure 16 The day range overview of the testbed results. The figure provides a comparison (in KWh), of the total energy consumption of the testbed, between with and without energy aware plug-in, based on the days of the week

Conclusion and perspectives

Cloud computing being private or public is becoming more and more primordial in IT sector due to the numerous advantages (see Section Main focus) it gives to its end users. To cope with the high user demands, data centres having cloud computing style possess myriad of ICT resources. Most of the cases, this over-provision of resources, which serves to respect the Service Level Agreements that the data centres have with their end users, leads to astronomical numbers with respect to energy consumption. To this end, power and energy consumptions of data centres have become an issue recently due to economical and ecological reasons. In this paper, we study the case of a private cloud computing data centre from the energy efficiency perspectives, and show that there are incentives to save energy. To this respect, we described in the form of UML class diagrams the ICT resources with their most-relevant energy related attributes. Furthermore, we provided generic power consumption prediction models for servers, storage devices and network equipment. Note that our proposed methodology is generic enough so that it encompasses any computing style: traditional, cloud and super computing. Finally, in order to validate the energy optimization policies of [5], we performed evaluations in a real-life private cloud computing data centre and showed that it's possible to save energy almost 20% when only single-site is taken into account.

As a future work, it is also interesting to investigate the federation of several data centres and propose new optimization policies which take into account the following two main objectives:

1. Minimizing the energy consumption of data centres.
2. Minimizing the CO_2 emissions of data centres.

With respect to item (1), our current optimization policies take into account minimizing the overall energy consumption of the data centre while not violating any of its SLAs. When item (2) is concerned, new optimization policies should be devised that take into account the availability of green energy so that the overall CO_2 emissions are minimized by taking advantage of the concept of data centre federation.

In the end, our approach has the following two-dimensional benefits:

1. For the data centre businesses:
   (a) Reduction of costs and therefore prices.
   (b) Marketing options for green services.
   (c) Provision of potential energy legislation.
2. For the data centre end users: reduction of cost for services.

Endnotes

[a]Based on the required performance and energy saving needs.

[b]http://www.engineeringtoolbox.com/fans-efficiency-power-consumption-d_197.html

[c]Hewlett-Packard Italy Innovation Center located in Milan

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

[e]Implemented in the form of a virtual machine

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

RB conceived of the study, participated in its design, carried out the studies related to the power consumption models of servers and storage devices as well as drafted the document. HdM participated in the technical discussions and reviewed the document. RL participated in the design of the study related to the network part and devised the corresponding power and energy consumption models. GG conceived of the study, participated in its design and coordination, carried out statistical analysis and helped to draft the manuscript. All authors read and approved the final manuscript.

Authors' information

Robert Basmadjian

Has been working at the University of Passau, Germany, as a Post Doctorial fellow since 2009. He holds an M.Sc. and Ph.D. in Computer Science from the University of Toulouse. His research interests are replication in distributed systems, mathematical modelling of systems, and energy-efficiency in large-scale systems. He participates in the EU funded projects ALL4Green, FIT4Green and EuroNF.

Hermann de Meer

Is currently appointed as Full Professor at the University of Passau, Germany, and as Honorary Professor at University College London, UK. He is director of the Institute of IT Security and Security Law (ISL) at the University of Passau. His main research interests include IT security and resilience, virtualization and energy efficiency, complex and self-organizing systems, peer-to-peer systems, quality of service and performance modeling, Internet protocols, home networking, and mobile computing. Hermann de Meer has led several nationally and internationally funded projects on Performance Modeling and Computer Networking. He currently holds several research grants funded by the Deutsche Forschungsgemeinschaft (DFG) and by the EU (FP6 and FP7).

Ricardo Lent

Is a Research Fellow in the Intelligent Systems and Networks group at Imperial College London. He holds an M.Sc. and Ph.D. in Computer Science from the University of Central Florida, an M.Sc. in Telecommunications from Universidad Nacional de Ingenieria and a B.Sc. in Electronic Engineering from Universidad Ricardo Palma. His prior experience includes work as a Principal Network Engineer for industry and as a Visiting Assistant Professor at the University of Central Florida. His research is focused on self-aware networks in both the wired and wireless domains and other leading topics in computer networks and distributed systems. Dr. Lent has participated in many UK and EU funded research projects as a post-doctoral fellow and as a Co-PI, including UK EPSRC Self-Aware Networks and Quality of Service, EU FP6 CASCADAS, EU FP7 DIESIS, and EU FP7 FIT4Green.

Giovanni Giuliani

Graduated in Electronics Engineering with honors in 1982 at Politecnico in Milan(Italy), he has over 25 years of working experience in R&D, IT Consulting and System Integration in major Computer Companies (Olivetti, Digital, Compaq, HP). He has covered various positions in SW Engineering until 1997, working on Operating System development and Manufacturing Framework products. He has been nominated Consulting Engineer by Digital Corporate Engineering Board in 1992. In parallel to his job in the industry, Giovanni Giuliani performed university teaching and research activities from 1988 to 1993 at Engineering and Computer Science universities in Milan and Como. Starting from 1998 - as Solution Architect in Consulting and System Integration organization in Digital, and then, due to the mergers, Compaq and finally HP - he delivered several complex projects in Manufacturing, Government, Finance market, then exploring the area of Mobility for various sectors. Back in 2005 he joined the HP Italy Innovation Center, where he has been involved in several research projects funded by European Commission in the area of mobility, collaboration, cloud computing and Green-IT. Since 2009 he's leading the HP Italy Cloud Computing Initiative, that has the goal of helping customers to understand Cloud Computing, evaluate possible impacts and benefits, "hands-on"experiment Cloud Computing technologies in a real lab, start Cloud Computing Proof of Concept and Pilot projects.

Acknowledgements

The research leading to these results was supported by the European Community's 7^{th} Framework Programme in the context of the FIT4Green project [32].

References

1. Newsroom G (2007) Gartner Estimates ICT Accounts for 2 Percent of Global CO2 Emissions. Tech rep. [http://www.gartner.com/it/page.jsp?id=503867]

2. Hamilton J: Cooperative Expendable Micro-Slice Servers (CEMS): Low Cost, Low Power Servers for Internet-Scale Services. Tech. rep., Amazon Web Services http://mvdirona.com/jrh/talksandpapers/jameshamilton_cems.pdf

3. Vaquero LM, Rodero-Merino L, Caceres J, Lindner M (2009) A break in the clouds: towards a cloud definition. SIGCOMM Comput Commun Rev 39:50–55.

4. Mell P, Grance T (2009) The NIST Definition of Cloud Computing. Tech. rep., National Institute of Standards and Technology, Information Technology Laboratory.

5. Quan DM, Basmadjian R, deMeer H, Lent R, Mahmoodi T, Sannelli D, Mezza F, Dupont C (2011) Energy Efficient Resource Allocation Strategy for Cloud Data Centres. In Proceedings of the 26th Int'l Symposium on Computer and Information Sciences (ISCIS 2011), pp 133–142.

6. Vijaykrishnan N, Kandemir MT, Irwin MJ, Kim HS, Ye W (2000) Energy-driven integrated hardware-software optimizations using SimplePower. In Proceedings of the 27th annual international symposium on Computer architecture, pp 95 –106.

7. Gurumurthi S, Sivasubramaniam A, Irwin MJ, Vijaykrishnan N, Kandemir M (2002) Using complete machine simulation for software power estimation: the SoftWatt approach. In Proceedings of Eighth International Symposium on High-Performance Computer Architecture, pp 141–150.

8. Shafi H, Bohrer PJ, Phelan J, Rusu CA, Peterson JL (2003) Design and validation of a performance and power simulator for PowerPC systems. IBM J Res Dev 47:641–651.

9. Lewis A, Ghosh S, Tzeng NF (2008) Run-time energy consumption estimation based on workload in server systems. In Proceedings of the 2008 conference on Power aware computing and systems

10. Economou D, Rivoire S, ChristosKozyrakis (2006) Full-system power analysis and modeling for server environments. In Workshop on Modeling Benchmarking and Simulation (MOBS).

11. Berrendorf R, Ziegler H (1998) The Performance Counter Library: A Common Interface to Access Hardware Performance Counters on Microprocessors. Tech rep, FZJ-ZAM-IB-9816:1–58, http://www2.fz-juelich.de/jsc/docs/printable/ib/ib-98/ib-9816.pdf

12. West P (2008) Core Monitors: Monitoring. Master's thesis The Florida State University

13. Fan X, Weber WD, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In Proceedings of the 34th annual international symposium on computer architecture

14. Saravana M, Govidan S, Lefurgy C, Dholakia A (2009) Using on-line power modeling for server power capping

15. Zedlewski J, Sobti S, Garg N, Zheng F, Krishnamurthy A, Wang RY (2003) Modeling Hard-Disk Power Consumption. In Proceedings of the 2nd USENIX Conference on File and Storage Technologies

16. Molaro D, Payer H, Moal DL (2009) Tempo: Disk Drive Power Consumption Characterization and Modeling. In IEEE 13th International Symposium on Consumer Electronics, pp 246–250.

17. Hylick A, Sohan R, Rice A, Jones B (2008) An Analysis of Hard Drive Energy Consumption. In IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, pp 1–10.

18. Greenawalt P (1994) Modeling power management for hard disks. In In Proceedings of 2nd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp 62–66.

19. Allalouf M, Arbitman Y, Factor M, Kat R, Meth K, Naor D (2009) Storage modeling for power estimation. In Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference

20. Sivathanu S, Ungureanu C, Liu L (2010) Modeling the Performance and Energy of Storage Arrays. In *International Green Computing Conference*, pp 229–242.

21. Liu Z, Zhou J, Yu W, Wu F, Qin X, Xie (2011) MIND: A black-box energy consumption model for disk arrays. *International Green Computing Conference and Workshops* **0**:1–6.

22. Wang HS, Peh LS, Malik S (2003) A Power Model for Routers: Modeling Alpha 21364 and InfiniBand Routers. IEEE Micro **23**:26–35.

23. Ye TT, Benini L, Micheli GD (2002) Analysis of Power Consumption on Switch Fabrics in Network Routers. In Proceedings of the 39th Design Automation Conference (DAC, pp 524–529.

24. Kahng AB, Li B, Peh LS, Samadi K (2011) ORION 2.0: A Power-Area Simulator for Interconnection Networks. IEEE Transactions on Very Large Scale Integration (TVLSI). http://hdl.handle.net/1721.1/67492.

25. Hass J, Froedge J (2009) Usage and Public Reporting Guidelines for The Green Grid's Infrastructure Metrics PUE/DCiE. Tech. rep., The Green Grid

26. Meade RL, Diffenderfer R (2003) Foundations of Electronics: Circuits & Devices. Thomson Delmar Learning, Clifton Park, New York, ISBN: 0-7668-4026-3

27. ZES ZIMMER: 1 to 8 Channel Precision Power Analyzer LMG500 http://www.zes.com/english/products/one-to-eight-channel-precision-power-analyzer-lmg500.html.

28. Chandrakasan PA, Brodersen RW (1995) Minimizing Power Consumption in CMOS Circuits. Tech. rep., University of California at Berkeley. http://bwrc.eecs.berkeley.edu/publications/1995/\\Min_pwr_consump_CMOS_crct/paper.fm.pdf

29. van der Bijl HJ (1919) Theory and Operating Characteristics of the Themionic Amplifier. In Proceedings of the IRE (Institute of Radio Engineers), pp 97–126

30. HP Enclosure http://h18000.www1.hp.com/products/blades/components/enclosures/c-class/c3000/?jumpid=reg_R1002_USEN

31. HP Blade http://h10010.www1.hp.com/wwpc/us/en/sm/WF02d/3709945-3709945-3328410.html?jumpid=in_r2515_us/en/smb/psg/psc404redir-ot-xx-xx-/chev/

32. Basmadjian R, Bunse C, Georgiadou V, Giuliani G, Klingert S, Lovasz G, Majanen M (2010) FIT4Green: Energy aware ICT Optimization Policies. In Proceedings of the COST Action IC0804 on Energy Efficiency in Large Scale Distributed Systems - 1st Year

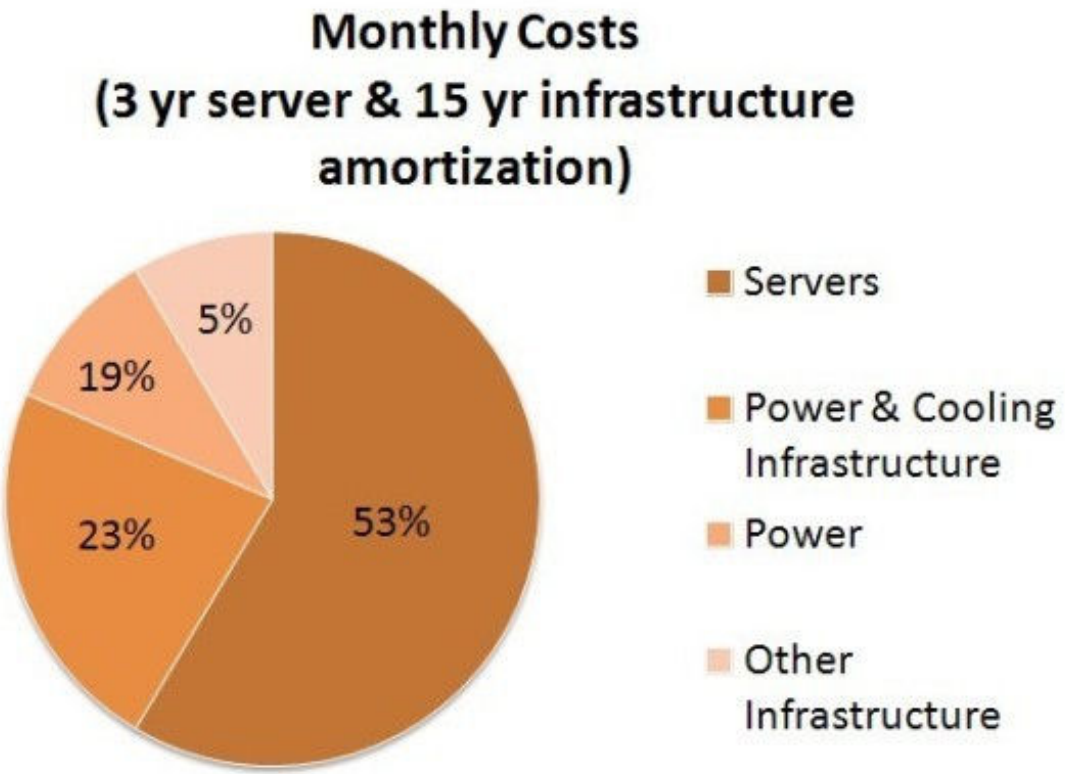

Monthly Costs
(3 yr server & 15 yr infrastructure amortization)
5%
19%
23%
53%
Servers
Power & Cooling Infrastructure
Power
Other Infrastructure

Figure 1

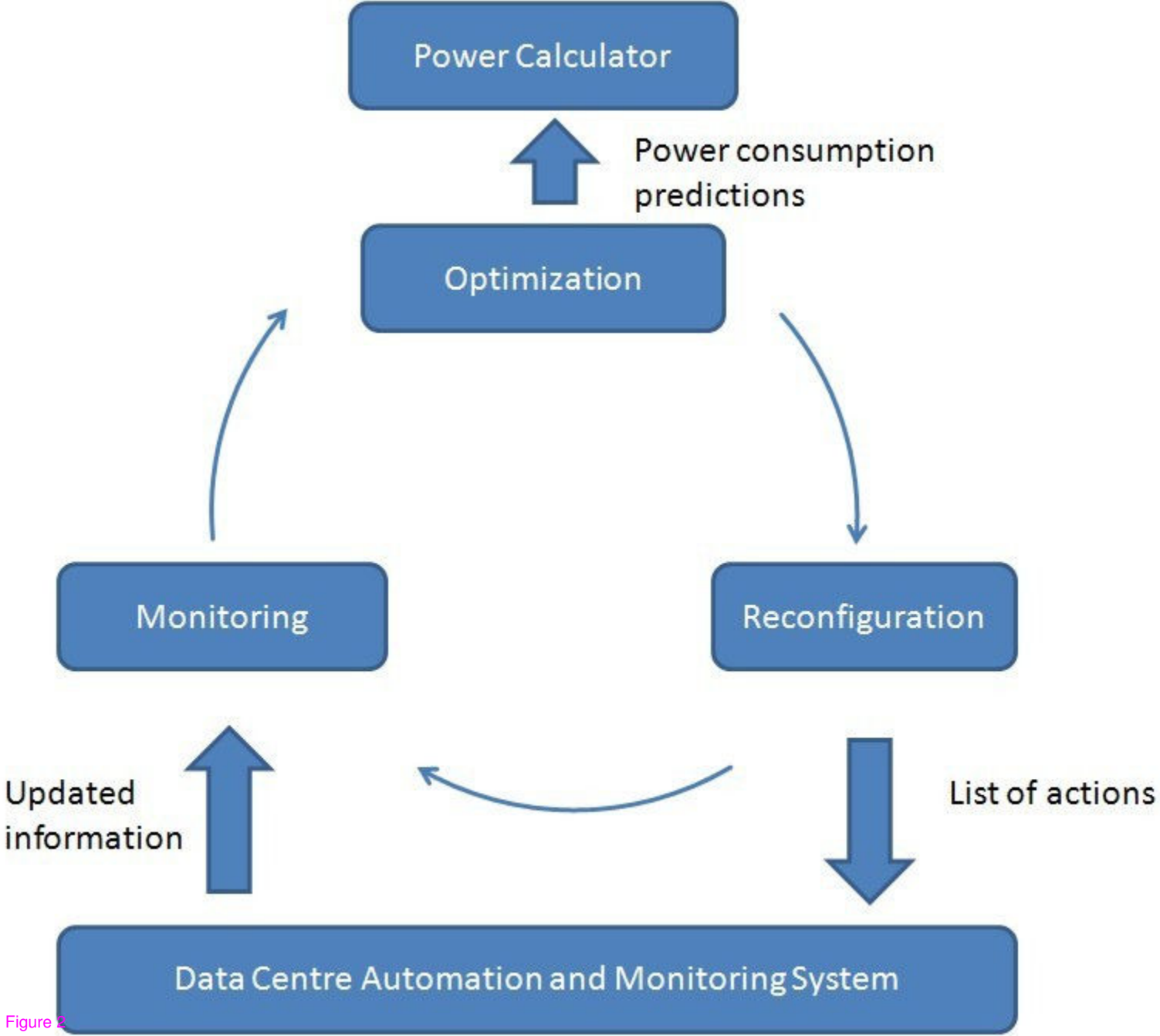

Power Calculator
Power consumption predictions
Optimization
Monitoring
Reconfiguration
Updated information
List of actions
Data Centre Automation and Monitoring System

Figure 2

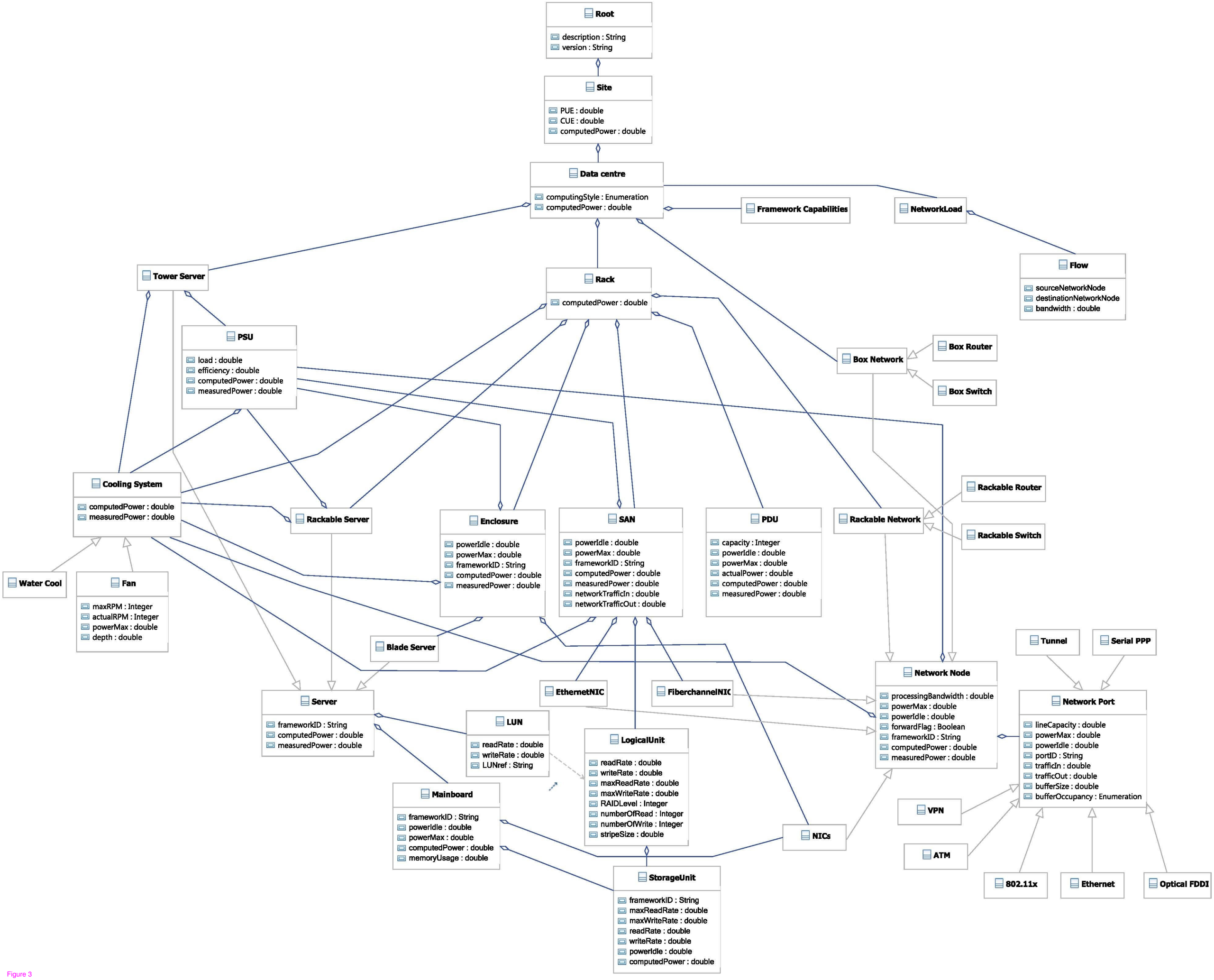

Root
description : String
version : String
Site
PUE : double
CUE : double
computedPower : double
Data centre
computingStyle : Enumeration
computedPower : double
Framework Capabilities
NetworkLoad
Flow
sourceNetworkNode
destinationNetworkNode
bandwidth : double
Tower Server
Rack
computedPower : double
PSU
load : double
efficiency : double
computedPower : double
measuredPower : double
Box Network
Box Router
Box Switch
Cooling System
computedPower : double
measuredPower : double
Rackable Server
Enclosure
powerIdle : double
powerMax : double
frameworkID : String
computedPower : double
measuredPower : double
SAN
powerIdle : double
powerMax : double
frameworkID : String
computedPower : double
measuredPower : double
networkTrafficIn : double
networkTrafficOut : double
PDU
capacity : Integer
powerIdle : double
powerMax : double
actualPower : double
computedPower : double
measuredPower : double
Rackable Network
Rackable Router
Rackable Switch
Water Cool
Fan
maxRPM : Integer
actualRPM : Integer
powerMax : double
depth : double
Blade Server
Server
frameworkID : String
computedPower : double
measuredPower : double
EthernetNIC
FiberchannelNIC
LUN
readRate : double
writeRate : double
LUNref : String
LogicalUnit
readRate : double
writeRate : double
maxReadRate : double
maxWriteRate : double
RAIDLevel : Integer
numberOfRead : Integer
numberOfWrite : Integer
stripeSize : double
Mainboard
frameworkID : String
powerIdle : double
powerMax : double
computedPower : double
memoryUsage : double
NICs
StorageUnit
frameworkID : String
maxReadRate : double
maxWriteRate : double
readRate : double
writeRate : double
powerIdle : double
computedPower : double
Network Node
processingBandwidth : double
powerMax : double
powerIdle : double
forwardFlag : Boolean
frameworkID : String
computedPower : double
measuredPower : double
Tunnel
Serial PPP
Network Port
lineCapacity : double
powerMax : double
powerIdle : double
portID : String
trafficIn : double
trafficOut : double
bufferSize : double
bufferOccupancy : Enumeration
VPN
ATM
802.11x
Ethernet
Optical FDDI

Figure 3

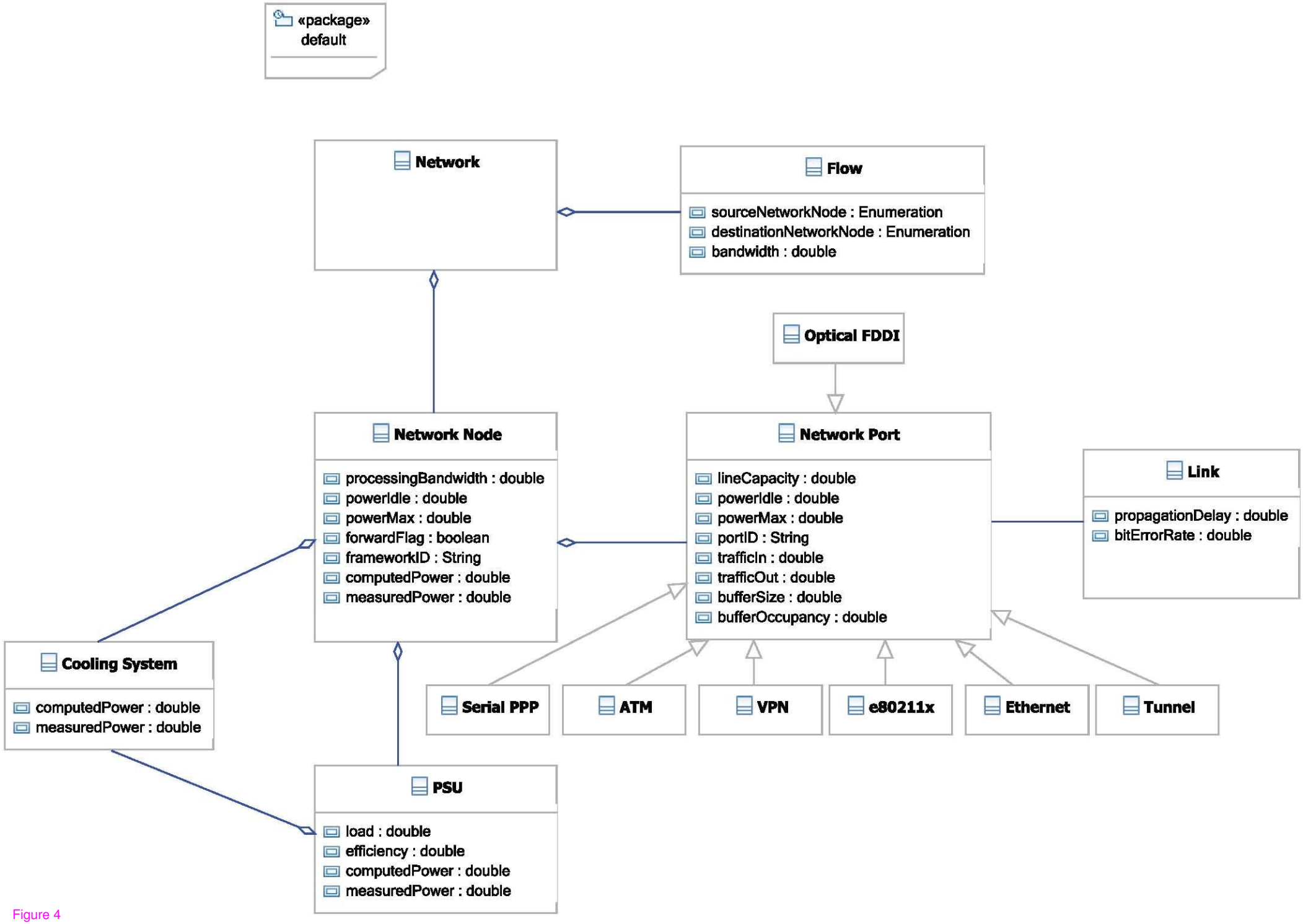

«package»
default
Network
Flow
sourceNetworkNode : Enumeration
destinationNetworkNode : Enumeration
bandwidth : double
Optical FDDI
Network Node
processingBandwidth : double
powerIdle : double
powerMax : double
forwardFlag : boolean
frameworkID : String
computedPower : double
measuredPower : double
Network Port
lineCapacity : double
powerIdle : double
powerMax : double
portID : String
trafficIn : double
trafficOut : double
bufferSize : double
bufferOccupancy : double
Link
propagationDelay : double
bitErrorRate : double
Cooling System
computedPower : double
measuredPower : double
Serial PPP
ATM
VPN
e80211x
Ethernet
Tunnel
PSU
load : double
efficiency : double
computedPower : double
measuredPower : double

Figure 4

«package»
server

Hard Disk
- rpm : Integer
- platters : Integer
- AAM : boolean

Storage Unit
- maxReadRate : double
- maxWriteRate : double
- powerIdle : double
- readRate : double
- writeRate : double
- computedPower : double
- frameworkID : String

Solid State Disk
- powerByRead : double
- powerByWrite : double

RAID
- computedPower : double
- level : Integer

Core
- voltage : double
- frequency : double
- frequencyMin : double
- frequencyMax : double
- frameworkID : String
- coreLoad : double
- computedPower : double
- totalPstates : Integer
- lastPstate : Integer

Cache
- level : Integer
- size : Integer
- computedPower : double

RAMStick
- voltage : double
- size : double
- type : Enumeration
- frequency : double
- bufferType : Enumeration
- vendor : Enumeration
- computedPower : double

Hardware RAID

CPU
- frameworkID : String
- DVFS : boolean
- lithography : Integer
- powerIdle : double
- powerMax : double
- architecture : Enumeration
- computedPower : double
- cpuUsage : double
- transistorNumber

NICs

Mainboard
- frameworkID : String
- powerIdle : double
- powerMax : double
- computedPower : double
- memoryUsage : double

Native Operating System

Native Hypervisor

Fan
- maxRPM : Integer
- actualRPM : Integer
- powerMax : double
- depth : double

Cooling System
- computedPower : double
- meauredPower : double

Server
- frameworkID : String
- computedPower : double
- measuredPower : double

Water Cooler

PSU
- load : double
- efficiency : double
- computedPower : double
- measuredPower : double

Rackable Server

Tower Server

Blade Server

Figure 5

«package»
storage

Cache
- level : Integer
- size : double
- computedPower : double

RAID
- computedPower : double
- level : Integer

Storage Unit
- maxReadRate : double
- maxWriteRate : double
- powerIdle : double
- readRate : double
- writeRate : double
- comptedPower : double
- frameworkID : String

Hard Disk
- rpm : Integer
- AAM : boolean
- platters : Integer

Software RAID
- name : String

Hardware RAID

Solid State Disk
- powerByRead : double
- powerByWrite : double

Logical Unit
- RAIDLevel : Integer
- maxReadRate : double
- maxWriteRate : double
- readRate : double
- writeRate : double
- numberOfRead : Integer
- numberOfWrite : Integer
- stripeSize : double

FiberchannelNIC

Network Node
- processingBandwidth : double
- powerIdle : double
- powerMax : double
- forwardFlag : boolean
- measuredPower : double
- comptedPower : double
- frameworkID : String

Mainboard
- frameworkID : String
- powerIdle : double
- powerMax : double
- computedPower : double
- memoryUsage : double

LUN
- LUNRef : Integer
- readRate : double
- writeRate : double

EthernetNIC

Server
- frameworkID : String
- measuredPower : double
- computedPower : double

SAN
- computedPower : double
- measuredPower : double
- powerIdle : double
- powerMax : double
- frameworkID : String
- networkTrafficIn : double
- networkTrafficOut : double

Cooling System
- computedPower : double
- measuredPower : double

PSU
- load : double
- efficiency : double
- measuredPower : double
- comptedPower : double

Fan
- maxRPM : Integer
- actualRPM : Integer
- powerMax : double
- depth : double

Figure 6

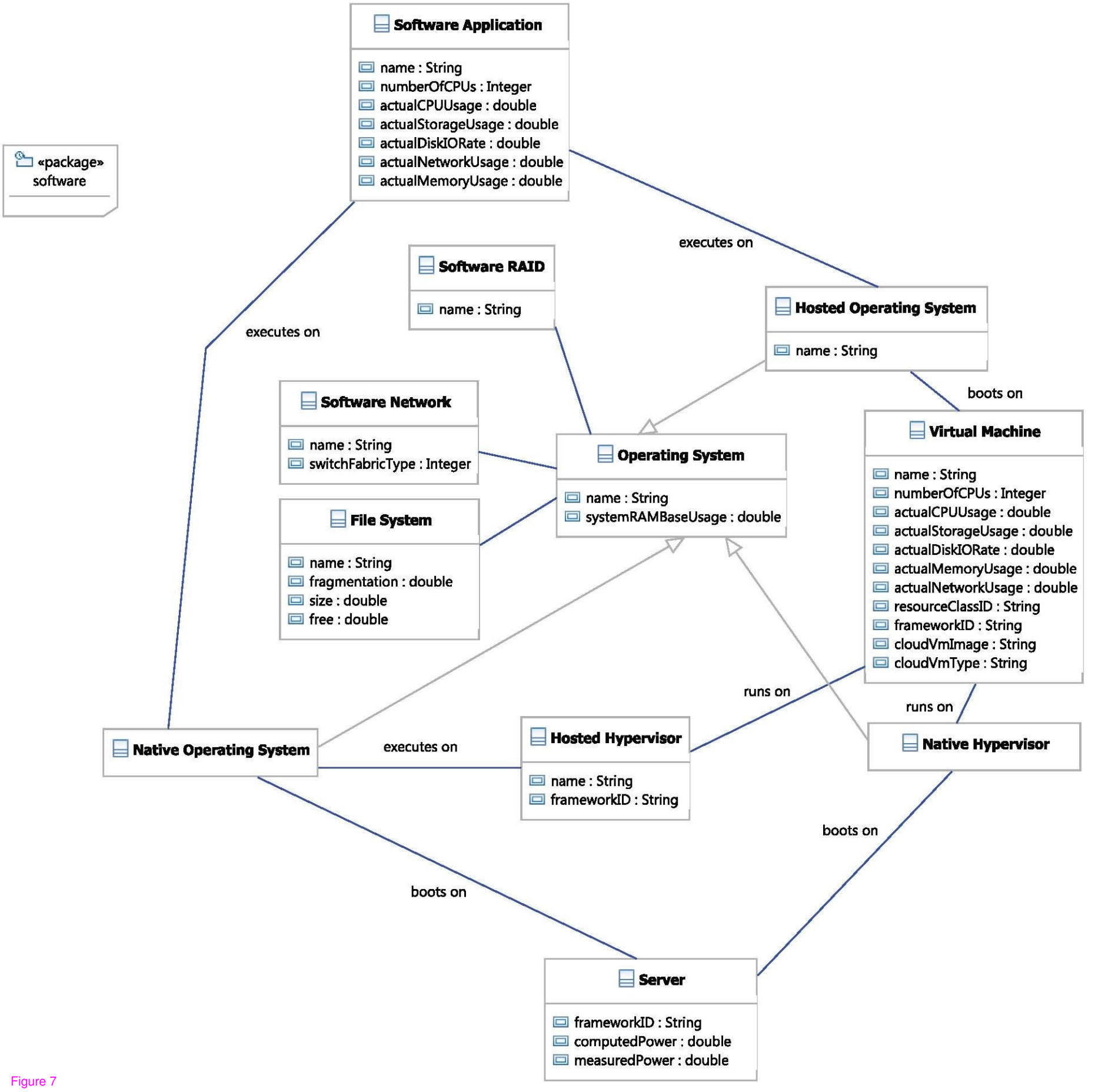

«package»
software
Software Application
name : String
numberOfCPUs : Integer
actualCPUUsage : double
actualStorageUsage : double
actualDiskIORate : double
actualNetworkUsage : double
actualMemoryUsage : double
executes on
executes on
Software RAID
name : String
Hosted Operating System
name : String
boots on
Software Network
name : String
switchFabricType : Integer
Operating System
name : String
systemRAMBaseUsage : double
Virtual Machine
name : String
numberOfCPUs : Integer
actualCPUUsage : double
actualStorageUsage : double
actualDiskIORate : double
actualMemoryUsage : double
actualNetworkUsage : double
resourceClassID : String
frameworkID : String
cloudVmImage : String
cloudVmType : String
File System
name : String
fragmentation : double
size : double
free : double
runs on
runs on
Native Operating System
executes on
Hosted Hypervisor
name : String
frameworkID : String
Native Hypervisor
boots on
boots on
Server
frameworkID : String
computedPower : double
measuredPower : double

Figure 7

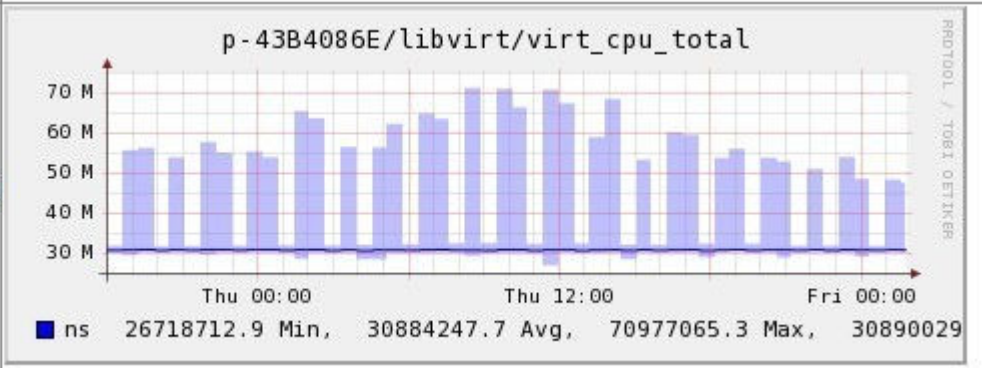
VirtCpuTotal	

Figure 8

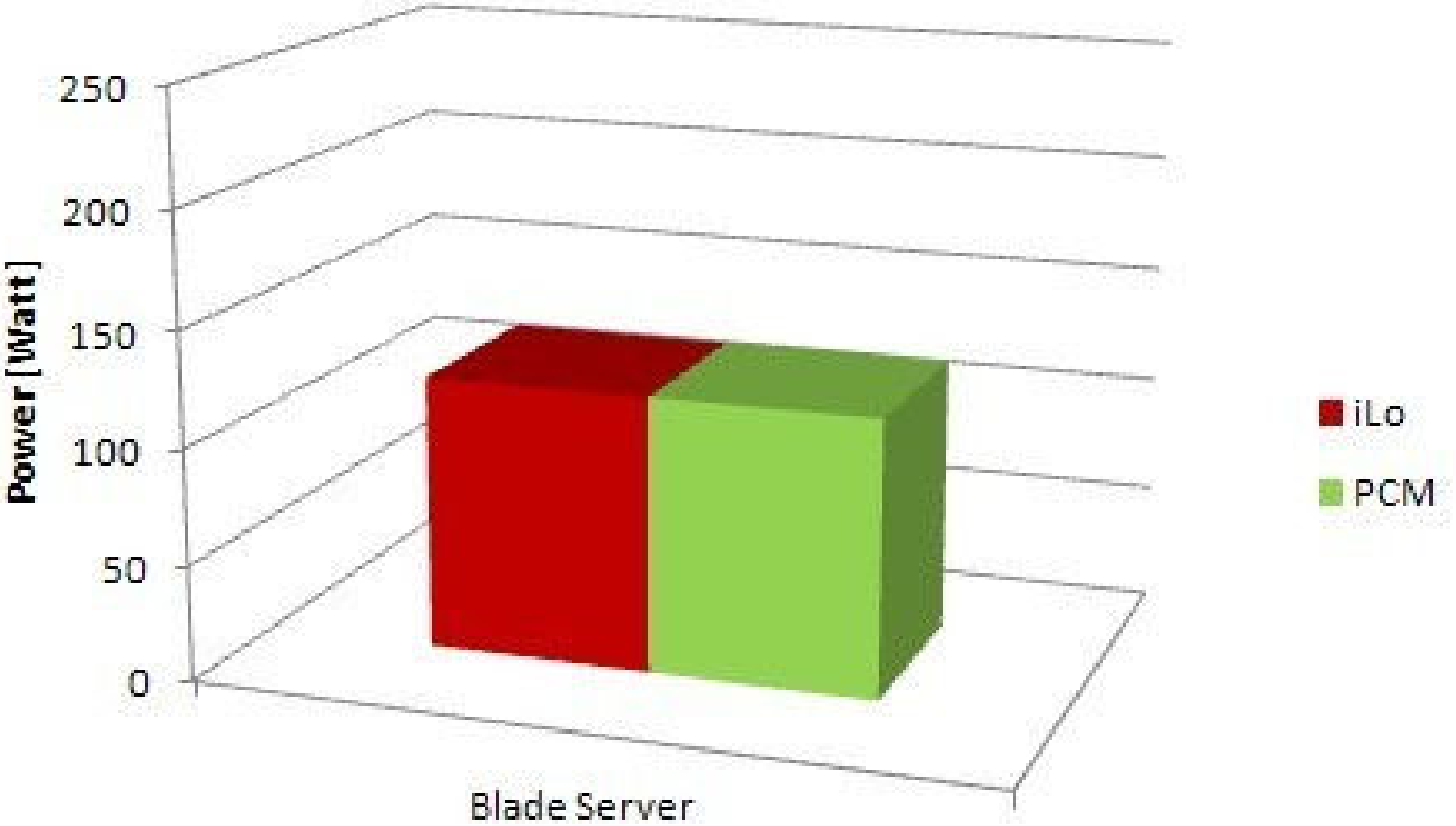


Figure 9

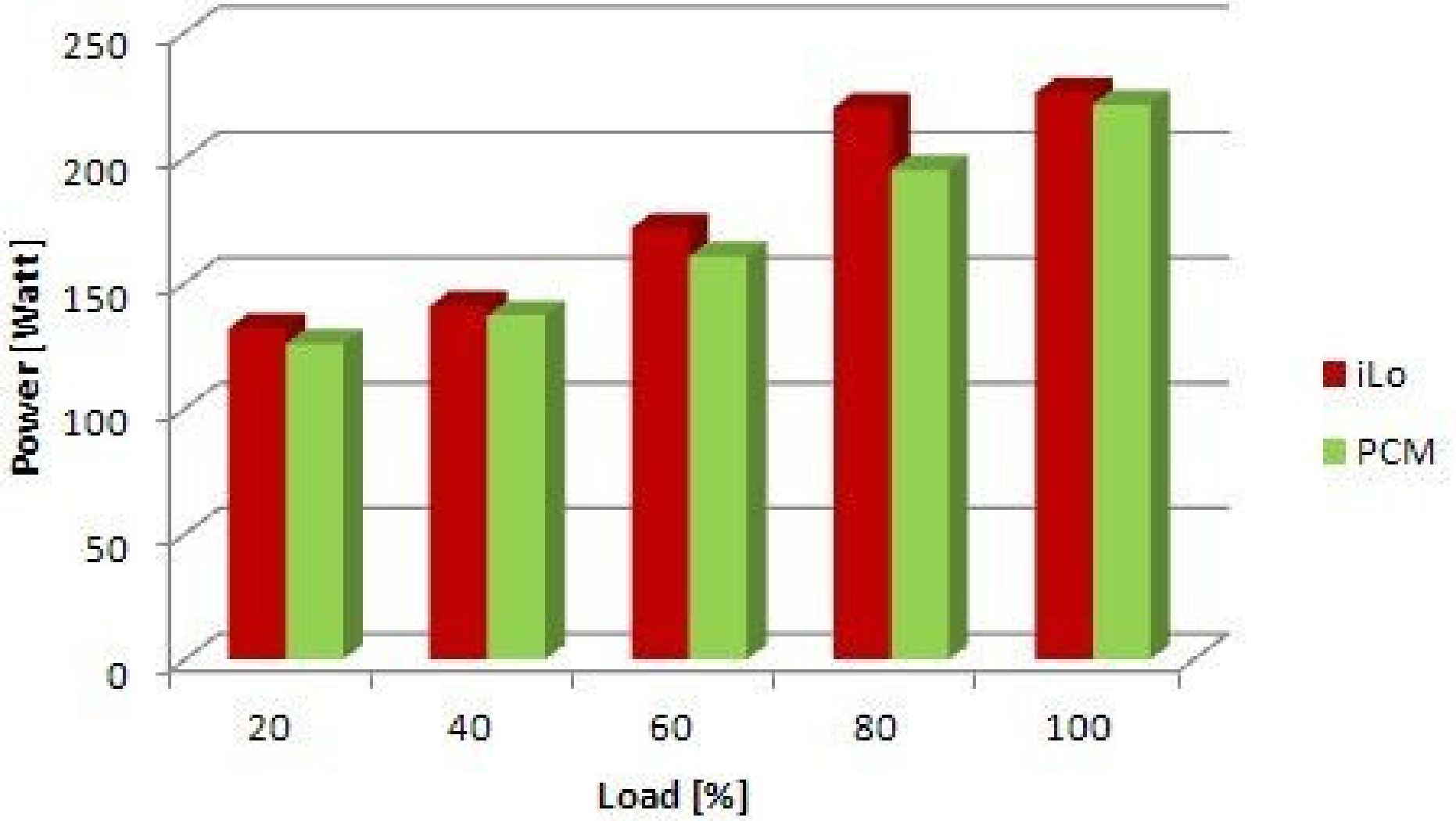


Figure 10

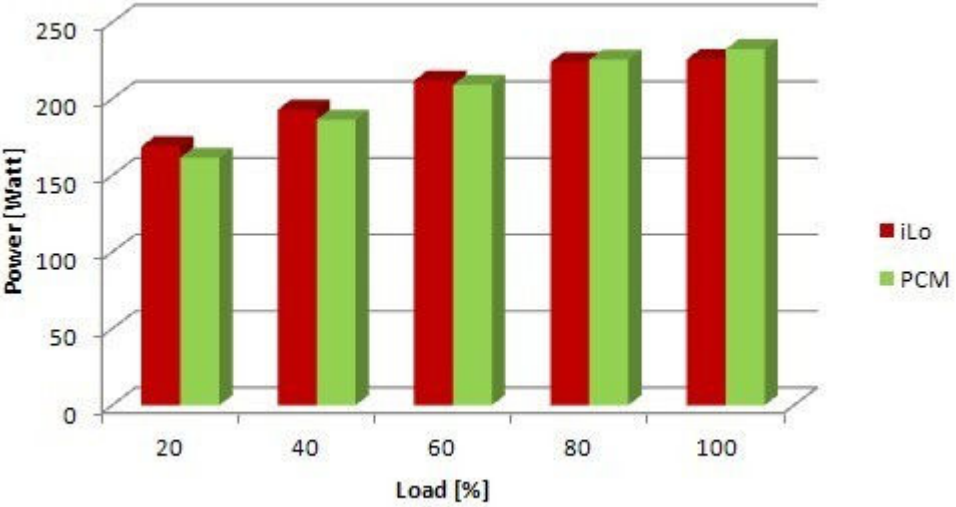


Figure 11

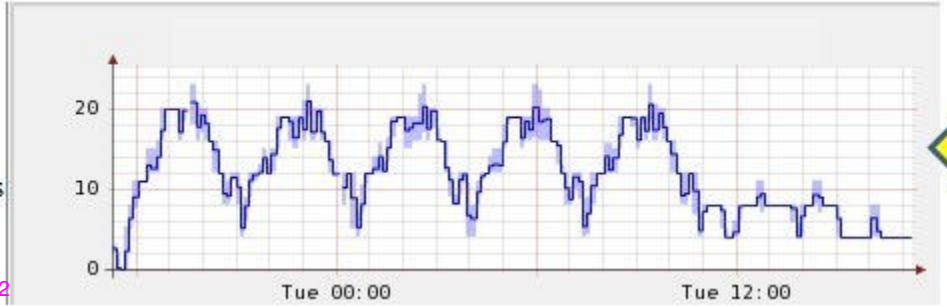

Instances
20
10
0
Tue 00:00
Tue 12:00
24 hours run
representing
1 week load

Figure 12

Figure 13

Watt

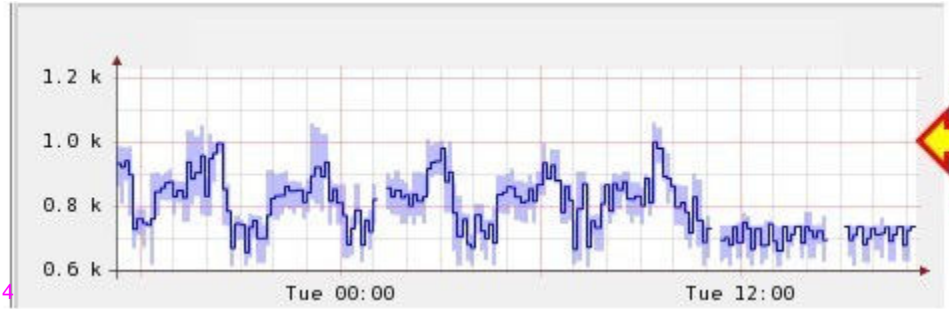


Figure 14

Configuration	Total Energy Consumption (KWh)	Average Power (Watt)	Energy Saving (KWh)	Energy Saving (%)
without energy-aware plug-in	24,758	1,031.58		
with energy-aware plug-in	20,306	846.08	4,452	**17.98%**

Figure 15

Configuration	Energy BEGIN	Saving % BEGIN	Energy CENTRAL	Saving % CENTRAL	Energy END	Saving % END
without energy-aware plug-in	8,391		8,493		7,874	
with energy-aware plug-in	7,026	**16.27%**	6,962	**18.03%**	6,318	**19.76%**

Figure 16