Al-Dulaimy et al. Journal of Cloud Computing: Advances, Systems and Applications (2016) 5:6
DOI 10.1186/s13677-016-0055-y

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Power management in virtualized data centers: state of the art

Auday Al-Dulaimy[1*], Wassim Itani[2], Ahmed Zekri[1,3] and Rached Zantout[4]

Abstract

Cloud computing is an emerging technology in the field of computing that provides access to a wide range of shared resources. The rapid growth of cloud computing has led to establishing numerous data centers around the world. As data centers consume huge amounts of power, enhancing their power efficiency has become a major challenge in cloud computing. This paper surveys previous studies and researches that aimed to improve power efficiency of virtualized data centers. This survey is a valuable guide for researchers in the field of power efficiency in virtualized data centers following the cloud computing model.

Keywords: Cloud computing, Data center, Virtualization, Power management, Power efficiency

Introduction

Cloud computing is an emerging model that delivers services over the Internet by providing access to a wide range of shared computational resources hosted in data centers. The growth of cloud computing model has led to establishing numerous data centers around the world that consume huge amounts of power. Therefore, eliminating any waste of power in cloud data centers is very necessary. This can be achieved by observing how power is delivered to the data centers' resources, and how these resources are utilized to serve users' jobs. Hence, the need for improving the existing resource allocation and management algorithms in cloud data centers, as well as proposing new ones, is highly required. This paper presents previous research works that aimed to improve power efficiency of virtualized data centers. It is a valuable guide for understanding the state of the art of how to manage the power consumed in such environments. Additionally, it leads to suggesting and adding new proposals for further enhancement.

This paper is structured as follows. The motivations and objectives of the paper are stated in the next section. Power and Energy section defines the terminology used throughout the paper. Power Consumption section explains the sources of power consumption. The section titled Power Efficiency Metrics lists and describes some power efficiency metrics. State of the Art in Power Management section describes the power management techniques in details. Finally, conclusions are drawn in the last section.

Motivations and objectives

Recently, numerous data centers were established around the world. These data centers consume large amount of energy.

In general, the consumed energy amount is resulting in:

1) Operating costs,
2) Carbon dioxide (CO_2) emissions.

This amount was estimated to be between 1.1 and 1.5 % of the total electricity use in 2010. It has increased by 56 % from 2005, and it will continue to increase in a rapid manner unless advanced energy efficient resource management algorithms are proposed [1, 2]. Beside, CO_2 emissions of the Information and Communication Technology (ICT) industry were accounted to be 2 % of the total global emissions. However, the CO_2 emissions affect the global warming [1].

Addressing the problem of high energy use is a significant issue due to its financial and environmental effects. So, it is important to improve the resource allocation algorithms and proposing new management approaches

* Correspondence: auday.aldulaimy@gmail.com; a.aldulaimy@student.bau.edu.lb
[1]Department of Mathematics & Computer Science, Beirut Arab University, Beirut, Lebanon
Full list of author information is available at the end of the article

which aim to enhance the power efficiency in the data centers.

This survey is a guideline for researchers in designing the energy aware algorithms that execute the users' jobs in cloud data centers.

Power and energy

In order to fully understand the relation between power and energy, and to comprehend their management techniques, some related terminology must be identified. So, fundamental terms are defined in this survey [3]:

Charge: *charge* is the quantity of electricity responsible for electric phenomena, expressed in Coulomb (C).

Current: *current* is the flow of electric charge transferred by a circuit per time unit, measured in Amperes (A):

$$a = \frac{\Delta c}{\Delta t} \tag{1}$$

Voltage: *voltage* is the work or energy required to move an electric charge, measured in Volt (V):

$$v = \frac{\Delta w}{\Delta c} \tag{2}$$

Power: *power* is the rate at which the work is performed by the system, measured in Watt (W):

$$P = \frac{\Delta w}{\Delta t} \tag{3}$$

Accordingly, power is calculated by multiplying the element current by element voltage:

$$P = \frac{\Delta w}{\Delta t} = \frac{\Delta w}{\Delta c} * \frac{\Delta c}{\Delta t} = v * a \tag{4}$$

Energy: *energy* is the total amount of work performed over a period of time, measured in Watt-hour (Wh):

$$E = P * \Delta t \tag{5}$$

From (4) and (5), it is obvious that both power and energy are defined in terms of the work that a system performs. It is very important to note the difference between power and energy. The reduction of power consumption does not necessarily lead to a reduction of the amount of energy consumed. For example, lowering the CPU performance by decreasing its voltage and/or frequency can result in a decrease in power consumption. In this case, completing the program execution may take a longer time. However, the consumed energy may not be decreased even by decreasing power consumption [1].

As discussed in detail in the following sections, power consumption can be reduced by applying Static Power Management (SPM), Dynamic Power Management (DPM), or even by applying both solutions to the system.

Power consumption

Generally, the power that any system consumes consists of two main parts [1, 4]: Static Power Consumption (SPC) and Dynamic Power Consumption, as shown in Fig. 1. A description of these two terms is presented in the following two sections.

Static Power Consumption (SPC)

Static power consumption is the power consumed by the system components. SPC is caused by leakage currents of the active circuits in the powered system. It is independent of clock rates and does not rely on usage scenarios. Instead, it is fundamentally specified by the type of transistors and the technology applied to the processor of the system.

The reduction of SPC requires reducing the leakage current, and this can be done in three ways [5]:

1) Reducing the supplied voltage. The renowned technique that has been applied to system components (e.g. CPUs, cache memories) is called Supply Voltage Reduction (SVR),
2) Reducing the size of the circuit in the system, either by designing circuits with fewer transistors, or by cutting the power supplies to idle components to reduce the effective transistor count,
3) Cooling the system by applying cooling technologies. Cooling technologies can reduce the leakage power by allowing circuits to work faster as electricity encounters less resistance at lower temperatures. Also, they eliminate some negative effects of high temperatures, specifically the

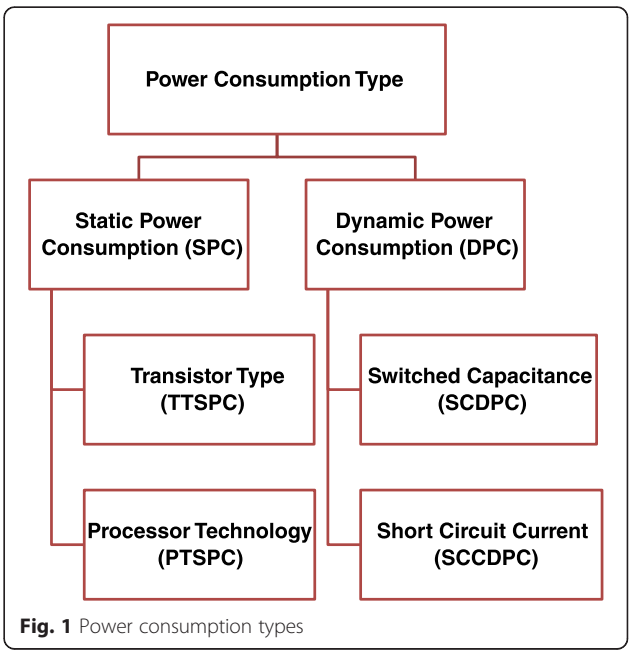

Fig. 1 Power consumption types

degradation of a chip's reliability and life expectancy.

The above three mentioned methods require improving the low-level system design, which lead to power reduction in SPC.

Dynamic Power Consumption (DPC)

Power consumption in this type results from the usage of system components. DPC is resulted by circuit activity. Mainly, DPC depends on clock rates, I/O activity, and the usage scenario. There are two sources of DPC; Switched Capacitance (SCDPC) and Short Circuit Current (SCCDPC) [5].

1) *SCDPC*: *SCDPC* is the major source of DPC. In this case, the consumed power is a byproduct of charging and discharging capacitors in circuits.
2) *SCCDPC*: *SCCDPC* is the minor source of DPC. The consumed power here results from the current switching between transistors. It approximately causes about 10–15 % of the total power consumption. However, this amount cannot be reduced without violating the system.

So, DPC can be defined as [1, 5–7]:

$$P_{Dynamic} = aCV^2f \tag{6}$$

where α represents the switching activity in the system, C is the physical capacitance, V is the voltage, and f is the CPU clock frequency of the system. The values of α and C are determined in the system design stage.

The DPC can be reduced by four methods [5]:

1) Reducing the switching activity.
2) Reducing the physical capacitance which depends on low level design parameters such as transistors sizes.
3) Reducing the supply voltage.
4) Reducing the clock frequency.

The core idea of the widely used DPM technique, called Dynamic Voltage and Frequency Scaling (DVFS), relies on a combined reduction of the supply voltage and/or clock frequency in a dynamical manner. This technique scales down the CPU performance by decreasing the voltage and frequency of the CPU when it is not fully utilized [8]. Most CPUs in modern systems (e.g. mobile, laptop, desktop, and server systems) support DVFS techniques.

Table 1 is a summary of the power consumption types [1, 4-5].

Table 1 A summary of power consumption types

Consumption Type	SPC	DPC
Results from:	The system components	The usage of the system components
Source:	The type of transistors and processor technology.	The short circuit current and switched capacitance.
Reason:	The leakage currents that are present in any active circuit of the powered system.	The circuit activity, Usage scenario, Clock rates, and I/O activity
Reduction:	Reducing the supplied voltage, Reducing the size of the circuits, Cooling the computer system.	Reducing the physical capacitance, Reducing the switching activity, Reducing the clock frequency, Reducing the supply voltage.

Power efficiency metrics

Different metrics are used to measure the power efficiency in data centers. Cloud providers have to use one or more metrics to estimate consumed power and overall performance. The researchers assessed the impact of applying their proposed strategies and algorithms in power management of data centers. These metrics can be classified as [9]:

1) Resource usage metrics: refer to the utilization of a certain resource (e.g. CPU, memory, bandwidth, storage capacity, etc.), concerning a component (node) or a set of components (node-group, rack).
2) Heat-aware metrics: use temperature as the main indicator for the behavior of a specific data center.
3) Energy-based metrics: refer to the amount of energy consumption during a certain period of time.
4) Impact metrics: that are used to assess the performance of data center in environmental and economic terms.

The following are examples of some metrics with their definitions:

i. *Total Energy Consumption*: *Total energy consumption* refers to the total power consumed by the data center over a certain period of time. It was measured in (W/h) and defined as:

$$TE_{DC} = \int_{t1}^{t2} TP_{DC}(t)dt \tag{7}$$

while TE_{DC} is the total energy consumed over a certain period of time, TP_{DC} is the total power consumed at a specific time.

ii. *Power Usage Effectiveness* (PUE): *Power Usage Effectiveness* refers to the ratio of the total power consumption in the data center and the power used by the IT equipment. PUE becomes the industry-preferred metric for measuring infrastructure energy efficiency for data centers, it is measured as [10]:

$$PUE_{DC} = \frac{TE_{DC}}{TE_{IT}} \quad (8)$$

iii. *Carbon Emission* (CE): *carbon emission* gives an indication about the amount of CO_2 emission in the data center. It uses Carbon Emissions Factor (CEF) and converts the total energy consumed to CO_2 emissions metric. CEF is a function of the participation of the different energy sources (e.g. carbon, gas, wind, solar, biomass, nuclear, etc.) which affect total electricity generation and the efficiency of conversion. Thus, this factor is different from one data center to another. CE is measured in kg CO_2 and can be defined as:

$$CE = TE_{DC} * CEF \quad (9)$$

iv. *Electricity Cost (EC)*: *electricity cost* converts the total energy consumed in the data center to a cost value by multiplying it by the price of electricity that differs from one data center to another. EC is measured in currencies (e.g. $, €, and £) and can defined as:

$$EC = TE_{DC} * Price \quad (10)$$

State of the art in power management

Many research works have been done in the area of power management. Power management techniques can be divided into two main parts as shown in Fig. 2.

These techniques are discussed in more details in the next two sections.

Table 2 is a summary of the power management types [5, 11-14].

Static Power Management (SPM)

SPM determines the efficiency of the usage of the hardware components (e.g. CPU, memory, disk storage, network devices, and power supplies). The reduction in SPM is permanent. It includes all the optimization methods that are applied at the design of the logic, circuits, and architecture levels of the system [5].

Logic level optimization attempts to optimize the power of the switching activity of combinational and sequential circuits.

Circuit level optimization reduces the power of the switching activity of individual logic gates and transistors. This is usually done by utilizing a sophisticated gate design and transistor sizing.

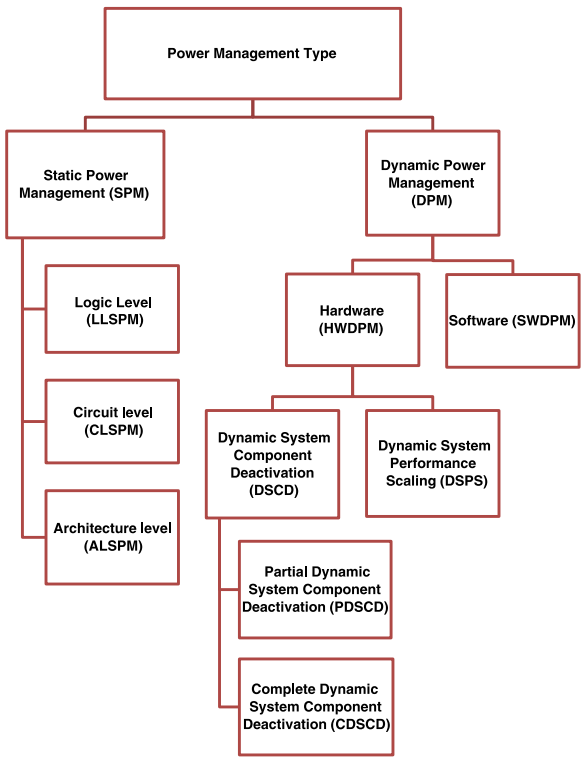

Fig. 2 Power management types

Architecture level optimization is achieved by mapping a high-level problem specification to a low-level design.

Low-power components are designed to save power and keep the system at an acceptable level of performance.

Many studies and SPM techniques have been proposed. The authors in [11] defined a novel system architecture, called fast array of wimpy nodes (FAWN). FAWN designed for low-power data intensive computing. In the proposed architecture, low-power CPUs combined with small amounts of local flash storage. The authors stated that such combination provides an efficient parallel data access in the system. In [12], the authors evaluated FAWN experimentally on various workloads. The results showed that the nodes with low-power CPUs are more efficient than conventional high-performance CPUs from the energy perspective. In [13], the authors defined a novel architecture, called Gordon, of

Table 2 A summary of power management types

Management Type	SPM	DPM
Level:	At the design of logic, circuits, and architecture levels	At the knowledge of the resource usage and application workloads
Reduction:	It is permanent	It is temporary
Implementation:	At the hardware level	At the hardware and software levels

low-power data-centric applications. Gordon can reduce the power consumption by utilizing low-power processors and flash memory. More details about the SPM techniques are available in [14].

In addition to the optimization in hardware-level system design, it is very important to carefully consider the implementation of programs that are to be executed on the system. Inaccurate software design can adversely affect the performance and may lead to power loss, even with perfectly designed hardware. Therefore, the code generation, the instructions used in the code, and the order of these instructions must be carefully selected, as they affect performance as well as power consumption.

Generally, it is impossible to analyze power consumption caused by any software at the hardware level [1, 15, 16]. However, the power management solution applied at the hardware level design is not within the scope of this survey.

Dynamic Power Management (DPM)

DPM optimizes energy consumption by utilizing a convenient knowledge of:

1) The available resources in the system and their usage.
2) The application workloads to optimize energy consumption.

Techniques in DPM allow a dynamic adjustment of the system's power states based on the system's current behavior. Additionally, they assume that the prediction of the workloads requirements is possible passable, which enables the adaptation of the future system behavior leading to perform the appropriate actions according to these requirements.

Thus, the reduction in DPM is temporary in the system, and it would last for an indefinite period of time according to the available resources and current workloads.

These techniques are distinguished by the level they are applied: hardware level or software level, as further explained in the next two sections.

Hardware Level Solutions

DPM techniques that are applied at the hardware level aim at reconfiguring the systems dynamically by designing methodologies to provide the requested services with a minimum number of active components or a minimum load on such components. The DPM techniques at a hardware level can selectively turn off the idle system components or reduce the performance of the partially unexploited ones. Also, it is possible to switch some components, such as the CPU, between active and idle modes to save energy [17, 18].

DPM at a hardware level can be divided into:

1) Dynamic System Component Deactivation (DSCD): which is subdivided into:
2) Dynamic System Performance Scaling (DSPS): DSPS techniques adjust the performance of the components dynamically according to the system state and the resources demands. These can be applied to system components to support the dynamic adjustment of their performance in a proportional manner with the power consumption. In addition to a complete deactivation approach, some components in the system (e.g. CPU) support increase or decrease in clock frequency along with adjustments of the supply voltage when the resource is not fully utilized. An example of DSPS is the DVFS technique [19] that is widely adopted in modern systems.
   i. Partial Dynamic System Component Deactivation (PDSCD): techniques are based on the idea of clock gating of parts of an electronic component.
   ii. Complete Dynamic System Component Deactivation (CDSCD): techniques are based on the idea of complete disabling for the components during periods of inactivity.

Software Level Solutions

A hardware level solution is very sophisticated. It is difficult to implement any modification or reconfiguration at this level. Therefore, shifting to the software level solutions is highly demanded.

There have been some proposed solutions for managing power consumption, such as Advanced Power Management (APM) performed by Basic Input/Output System (BIOS), firmware based, and platform-specific power management systems. But these solutions are hardware and platform-dependent.

The first attempt to address a software solution was made in 1996, when the first version of the Advanced Configuration and Power Interface (ACPI) was proposed. ACPI was a platform-independent interface. It improved the existing power and configuration standards for hardware devices and allowed operating systems to control power management and efficiently configured the hardware platform they ran on. ACPI has been widely accepted as a standard that brings DPM into the operating system (OS) control of all aspects of power management and device configuration [20, 21].

ACPI defines a number of power states. These states can be enabled in the system at runtime. Also, it gives software developers the ability to leverage the flexibility in adjusting the system's power states [1].

The states which are relevant in the context of DPM are:

i. *C-states*: *C-states* are the CPU power states C0, C1, C2, and C3. They denote the Operating State, Halt, Stop-Clock, and Sleep Mode respectively. Recently, deep power down states (e.g. C4, C5, C6, and C7) were introduced to define different levels of lower activity.
ii. *P-states*: *P-states* are the power-performance states when the processor is operating. P-states can be one of several states, and each state represents a specific combination setting of DVFS values. They are implementation-dependent, but P0 is always the highest performance one. If there is implementation-specific limit of n, then P_1 to P_n are successively lower performance states.

As mentioned earlier, data centers consume huge amounts of electrical power. Although DVFS technique can provide an efficient direction in managing power consumption of the CPU, more power reduction is required. The server consumes over 50 % of its actual peak power and up to 70 % in some cases, even when it is completely idle [22]. Switching PMs off is the only possible way to eliminate their SPC. These circumstances led to propose some solutions which are suit the data centers environment. Those solutions aimed to consolidate the workload to fewer PMs and deactivating the idle ones. The consolidation is a complicated problem. The performance of the applications can be affected from unnecessary consolidation. Therefore Quality of Service (QoS) requirements restricts consolidation. In general, QoS are defined in terms of Service Level Agreement (SLA) between cloud users (or their brokers) and cloud providers.

Many studies and approaches have been dedicated to enhance the power-efficiency in virtualized data centers.

Management techniques that take into account the concept of virtualized systems were first explored by Nathuji and Schwan in 2007 [23]. The authors examined ways of integrating power management mechanisms into the virtualized data center environments and presented a power management model for such environments, called VirtualPower, which controlled the power consumption of underlying platforms. A new technique, called "soft resource scaling", was applied in the model of that study. This technique emulated hardware scaling by providing less time for the VM to utilize a resource. It is a very efficient technique when hardware scaling is not supported. VirtualPower provided a set of virtualized power states, called VirtualPower Management (VPM) states, which were taken into account in all management actions. VirtualPower was able to modify model-specific registers (MSRs) in the PM and change the power states. Thus, an abstraction of (VPM) channels could be created. The channels then delivered guest VM power management actions as a set of 'soft' states that provided a virtual layer for application-aware management.

In the same year, Nathuji, Isci, and Gorbatov [24] pioneered in exploiting platform heterogeneity to improve the power management, while taking into account the emergence of the virtualization concept. They defined an intelligent workload allocation system that efficiently mapped workloads to the best fitted resources in heterogeneous platforms. The model consisted of three components: platform/workload descriptors, power/performance predictor, and allocator. The workload descriptor consisted of modules labeled by attributes, and a list of values for these attributes was provided. The platform descriptor consisted of individual modules representing system components that were used to convey information according to the PM's power management and hardware capabilities. The power/performance predictor used these descriptors to estimate the performance and power savings in the data center. Finally, the allocator used these predictions to map workloads to a specific type of platform. The allocator of the study evaluated the power efficiency tradeoffs of assigning a workload to many kinds of platforms, while each kind of platforms was associated with a cost metric. Jobs of the workloads were queued according to the values of the cost metric. Then, a mapping process was performed based on this queue with priority given to the job with the highest cost.

In [25], the authors investigated several previous solutions to the problem of high power consumption in data centers. Until then, there had not been any corresponding work on coordinating all these solutions. Hence, the authors characterized the existing solutions and classified them into hardware/software and local/global levels.

Then, the authors proposed their solution which was a model of multiple feedback controllers at various levels. The solution implemented in a distributed manner. Being the core level of that model, the Efficiency Controller (EC) was implemented in the system that served as a "container". This container was used as a reference to the controller depending on the desired part of its capacity. There was also a sensor to compare the value with the actual utilization of the container. By managing the actual resource utilization in the system and the reference value of the container, EC dynamically "resized the container". EC also monitored previous/former resource utilization and adjusted a processor P-state accordingly in order to match estimated demand. In this case, the consumed power was adapted to the total resource demand of the workloads in real time fashion. The second controller in that model, called Server Manager (SM), implemented the server level power capping. It measured the consumed power in the server and compared it with the power budget. The third controller, called Enclosure Manager (EM), implemented enclosure-level power

capping. It monitored the total power consumption of the code enclosure and compared it with an enclosure-level power budget periodically. The forth controller, which implemented the group-level power capping, was called Group Manager (GM). It worked at either the rack level or the data center level. GM function is to compare the actual power consumption of the group with the group power budget. The final controller in the model was the Virtual Machine Controller (VMC). The function of VMC is to collect the resource utilizations of the individual VMs in the system and performed a new VM-to-Servers mapping to minimize the total power based on the utilization information. That model provided a feedback control loop to federate multiple power management solutions at different levels. However, the proposed architecture needs an extension to include coordination with other solutions in the performance domains.

The authors investigated the design and implementation of a power-aware application placement model under heterogeneous virtualized server environment in [26]. When matching the application containers to the PMs, the placement model considered the power and migration costs.

The authors divided this work into two parts: The first part presented methods to implement the placement of cost-aware application on real servers, while the second one presented an application placement framework, called (pMapper), to solve the problem of high power consumption and reduce this amount as much as possible. The architecture of that framework consisted of three managers (Performance Manager, Power Manager, and Migration Manager) and an Arbitrator. Performance Manager monitored the behavior of the workload and rearranged the VMs while taking into account both the current resource requirements and SLAs. It consulted a knowledge base for an application performance in addition to the cost of its VM migration from one machine to another. Power Manager monitored the current power consumption and adjusted hardware power states. It used the power model in the knowledge base to determine the placement, and thus could estimate the power for every placement scenario and suggest the server throttling. Migration Manager estimated the cost of moving from a given placement to a new one and issues instructions for VM live migration to consolidate the overall VMs of the application. Arbitrator, which was the central intelligence part in pMapper, received the estimations from the three managers, configured a space for the VM placements, computed the best VMs sizes, and implemented an algorithm to choose the best VM placement. Once the Arbitrator decided on the new configuration, the three managers executed the following operations respectively: VM sizing, server throttling and live migration.

The authors in [27] implemented a dynamic resource provisioning of VMs for web applications in virtualized server environments. In this study, the provisioning problem was defined as sequential optimization under uncertainty. The authors proposed a framework called Limited Look-ahead Control (LLC) to solve this problem. The framework monitored the process of VMs provisioning and calculated the switching costs resulting from this provisioning, then it encoded the corresponding cost value in the optimization problem. In order to reduce power consumption and maintain the SLA that was defined in term of the processing rate for each application, the work proposed an online controller to decide the number of PMs and VMs to be allocated to each job in the workload. In this case, it is possible to turn the PMs on and off upon the controller's decision which may vary according to workload types.

In [28], the authors designed a trace-based workload placement controller to optimize the VMs allocation based on historical information, while maintaining specific QoS requirements. Then, they proactively mapped the workload periodically to the PMs. The work traded off between the consumed power of the required resources and the number of migrations that may occur after VMs placement. This approach was based on the fact that the historic traces of any application offer a model for describing the future application behavior. Traces were used to decide how to consolidate the VMs workloads to the PMs. The workload trace-based approach was considered a proactive controller that caused the VMs of the workloads migration among PMs in order to consolidate the system periodically. In this work, the optimization algorithm was enhanced by reducing the total number of migrations that occurred during successive control intervals. In addition to the proactive controller, the study also introduced a reactive migration controller. It detected the overload/under-load conditions in the PM and initiated the VMs migration. It dynamically added and removed PMs to maintain a balance of supply and demand resources by turning PMs on and off. The controller's act was based on the real-time data collected from the resource usage. However, during the workload placement approach, this work did not consider the effect of cache sharing on power efficiency.

The authors in [29] proposed novel techniques for placement and power consolidation of VMs in the virtualized data centers. Most hypervisors (e.g. Xen and VMware) provide administrators that are able to adjust the minimum number of the required resources (min) and the maximum number of allowable resources for a VM (max). Values such as (min) and (max) are very useful in ensuring the intelligent distribution of resources between different workloads. Hence, this study leverages

min, max and the shares parameters supported in virtualization technologies. The proposed techniques provided a tradeoff mechanism for power and performance when running heterogeneous workloads in data centers. They involved sorting physical machines (PMs) in an increasing order of the power cost per unit of capacity, wherein the objective function included power consumption and the utilization resulted from the execution of a specific VM, which was a priori assumption. The placement strategy then is to place all the VMs at their maximum resource requirements in a first-fit manner. An amount of 10 % of the capacity is leaved as a spare to handle any future growth of the resource usage.

The authors in [30] proposed a solution that facilitated coordination between the virtualization concept and the loosely couple platforms in data centers. This solution, known as vManage, provided an execution policy for better virtualization management. The design of vManager, which consisted of registry and proxy mechanisms, had several features such as a unified monitoring over platform and virtualization domains, the coordinators are able to interface with existing management controllers, easy portable across different hardware platforms with an independent implementation, and flexibility and extensibility in allowing new management solutions to participate in a "plug-and-play" manner. vManager is based on the concept of estimating "stability." In other words, it is based on the probability that a proposed VM reallocation will stay efficient for an appropriate future period of time. The predictions of future resource demands are computed using a probability function. This study provided an example of stability management and coordinated VM placement in the data center called the stabilizer, which is a plug-in' component. The VM placement policy considers the platform requirements as well as the requirements of the VM including CPU, memory, and network constraints. However, the proposed solution needs to be extended for larger scale data centers.

In [31], the authors tackled the problem of optimizing resource allocation for multitier applications in consolidated server environments. This was achieved by exploring a runtime cost-sensitive adaptation engine that weighed the potential benefits of automatic reconfiguration and their corresponding costs. The study suggested an offline cost model for each application to decide when and how to reconfigure the VMs. The model estimated the cost according to the changes in the utility for the application, which was a function of the response time for that application. This model improved the application response time and ameliorated the period over which the system remained functional in the new VM configuration.

In order to optimize resource allocation in data centers, the authors in [32] presented a multi-tiered resource scheduling scheme in that environment. This scheme, named RAINBOW, automatically provided on-demand capacities via resources flowing that indicated which resources were released by some VMs to be allocated to other VMs. The resource flowing was modeled using optimization theory and was resolved by the Simplex Method. Based on this model, the authors proposed a scheduling algorithm to optimize resource allocation and to ensure performance of some critical services, leading to reduction in consumed power. The proposed scheduling algorithm had three levels. The first was Application-Level scheduler: which dispatched requests across the VMs by applying VM migration; the second was Local-Level scheduler: which allocated resources of a PM to VMs according to their priorities; and the third was Global-level scheduler: which controlled the flow of resources among the applications.

In order to meet the QoS in this study, the resources were allocated to the applications according to the application priorities. In case of limited resources, the resources of low priority applications would be allocated to critical ones. The authors stated that performance of critical applications was guaranteed using this scenario. However, there is a need to analyze the effect and overhead caused by each tier of the proposed multi-tiered resource scheduling.

In [33], Stillwell, Schanzenbach, Vivien, and Casanova demonstrated the utilization of resource allocation management systems in virtualization technology for sharing parallel computing resources among competing jobs. The model focused on HPC applications. The authors defined the resource allocation problem considering a number of underlying assumptions and determined the complexity of each one. They also proposed a more general approach by eliminating some assumptions. The problem of resource allocation was formally defined as Mixed Integer Programming (MIP) model. The design of this model was based on two assumptions: first, the application required only one VM instance, and second, the computational power and memory requirements needed by the application were static. However, estimating accurate job resource requirements for jobs is a weak point in this work.

In [34], the authors investigated the problem of high power consumption in cloud data centers. They proposed an energy-aware resource provisioning that mapped the resources to the applications while meeting the SLA between the provider and the users. The study proposed visions, challenges, principles, and an architectural framework for energy-efficient model in virtualized data center environments. It focused on providing resources dynamically and how allocation algorithms could be improved via managing consumed energy among various data center infrastructures. The proposed

framework consisted of three levels; *User level*: at this level, users or their brokers submit their service requests to the cloud. *Allocator level*: which acts as an interface between users and the cloud infrastructure. *Data center level*: represents the VMs and PMs.

The VM allocation process was divided into two parts. The first part was receiving new requests for VM provisioning. All VMs were sorted in decreasing order according to their current utilization, and then allocated each VM to a PM that expanded the minimum amount of increment in the consumed energy. The second part was optimizing the current allocation of VMs, which was further divided into two steps: selecting the VMs to be migrated and placing the selected VMs on new PMs.

In that study, the selection of the migrating VMs was heuristically achieved. Four heuristics were used; the first one was based upon setting an upper utilization threshold for PMs. The VMs were allocated to a PM when the placement process kept the total CPU utilization below that threshold. The other three heuristics were based on the idea of setting two thresholds for utilization, upper and lower. The total CPU utilization by all VMs had to remain between the setting two thresholds.

The four heuristics relied on three policies: the first was migrating the least number of VMs to minimize migration overhead. The second was migrating VMs that had the lowest usage of CPU to maintain utilization. The third policy was selecting the necessary number of VMs based on a uniformly distributed random variable.

In [35], the authors analyzed the cost of energy in virtualized data center environments. The virtualization concept inspired the authors to propose an energy-efficient framework dedicated to cloud architecture, and they called it Green Open Cloud (GOC). GOC was proposed for the next generation of Cloud data centers that support extra facilities, such as advanced reservation. GOC has the ability to aggregate the workload by negotiating with users. In this case, the idle PMs can be switched off for a longer period of time without the need for further negotiations with the users.

In [36], the paper presented the design and implementation of two VM management solutions: Distributed Resource Scheduler (DRS) and Distributed Power Management (DPM).

DRS managed the allocation of physical resources to a set of VMs by mapping these VMs to PMs. Additionally; it performed intelligent load balancing in order to enhance the performance. DRS provided a "what-if" mode to handle any changes in workloads or PM configuration. DRS solution performed four key resource-management operations: Computes the amount of resources requested by a VM based on the reservation, periodically balanced load across PMs by performing VM migrations, saved power by benefiting from DPM, and performed initial placement of VMs onto PMs.

DPM surpassed DRS in its ability to reduce the consumed power by consolidating VMs onto fewer number of PMs. DPM is able to power a PM off when the CPU and memory resources have low utilization. At the same time, it is able to power a PM on appropriately when demand on resources increases, or in order to maintain the constraints.

The researchers in [37] investigated the resource fragments which resulted from the imbalanced use of the PM resources. They mentioned that the problem of VMs to PMs placement should be solved according to a resource-balanced strategy. To characterize the resource usage of each PM, the authors proposed a multi-dimensional space model, while each dimension of the space corresponds to one dimensional resource. The whole space is partitioned into three domains, each with particular features, to elucidate the appropriateness of resource utilization for each VM placement process. The proposed model can be used as a guide in designing the resource-balanced VM placement algorithms. Based on this model, the researchers proposed their own energy efficient VM placement algorithm and called it (EAGLE). EAGLE was based on a tradeoff at each time-slot between balancing multi-dimensional resource utilization and reducing the total number of PMs during VMs placement. EAGLE checks the next resource usage state for each available PM and chooses the most suitable one. A new PM could be turned on to avoid any excessive resource fragments. This would decrease excessive resource fragments and further reduce the number of PMs on the long run. This algorithm resulted in a better utilization of resources, introduced less resource fragments and saved energy.

The trade-off between energy efficiency and SLA constraints were analyzed in [38]. The authors studied the users' utilization patterns and introduced a dynamic resource provisioning mechanism to over-allocate capacity in cloud data centers. The core concept of the proposed mechanism was to employ the resource utilization patterns of each user to eliminate any potential waste in utilization that might result from overestimation of resources requests. The over-allocate algorithm in this mechanism considered two different parameters: the predicted resource utilization based on historical data, and dynamic occupation to determine the maximum number of resources that could be allocated over the actual PM capacity. The PM capacity was calculated based on the cost-benefit analysis of deploying a specific instance into a particular server. It also allowed consolidating additional VMs in the same PM. A compensation mechanism to adjust resource allocation in cases of underestimation was also discussed in this study.

In his thesis [1], Anton Beloglazov proposed novel algorithms for distributed dynamic consolidation of VMs in virtualized cloud data centers. The thesis reduced amount of the total energy consumption under different workload requirements. Energy consumption was reduced by dynamically switching PMs on and off to meet the current resource demand. The author suggested a data center model and applied a VM placement algorithm that worked in this model. The data center model consisted of numerous PMs and had two types of managers to coordinate the VM management. The local manager residing on each PM as a module of the virtual machine management (VMM) and the global manager that resided as a master for a specific number of PMs. The decision to place the VM on a specific PM was made individually according to the communication between the local manager on the PM and the global managers.

The thesis also presented a novel approach that optimally solved the problem of host overload detection by maximizing mean inter-migration time. This approach was based on a Markov chain model and worked for any fixed workload and a given state configuration.

The authors in [39] investigated the VM provisioning as an essential technique in cloud computing. VM provisioning refers to providing VMs upon users' requests. The work proposed a power-aware VM provisioning model for both hard and soft real-time services. A real-time service (such as financial analysis, distributed databases, and image processing) was presented as real-time VM requests. It included many tasks, and each task was defined by some parameters. Therefore, when users made their requests to the cloud computing environment, appropriate VMs were allocated for executing those requests. Brokers were responsible for finding VMs for the users' requests. The requirements of the requested VMs were called Real-Time Virtual Machine (RT-VM) in this paper. Each RT-VM V_i included three parameters: utilization u_i, Million Instruction Per Second (MIPS) m_i , and deadline d_i. Such requirements imply that the real-time service is guaranteed if the allocated VM keeps providing a processing capacity of $u_i \times m_i$ amount by the specified deadline d_i. After defining their power model, the authors proposed a power aware framework including the following five steps: Requesting a VM, generating the RT-VM from real-time applications, requesting a real-time VM, mapping the physical processors, and finally, executing the real-time applications.

This study suggested the variable w_i as the remaining service time. The initial value of w_i is defined by $u_i \times m_i \times (d_i - t_s)$, at the submission time t_s. If V_i is provided with q_i MIPS rate for the period t_p, then, the value w_i is decreased by $q_i \times t_p$. In such case, V_i finishes its service when w_i becomes zero.

When a datacenter receives a RT-VM request from users or their brokers, it returns the price of providing the RT-VM service if it can provide real-time VMs for that request. Then, the users or brokers can select the VM with the minimum price among available VMs provided by the datacenters. Thus, the provisioning policy in this work was selecting the processing element with the minimum price to maximize user/broker profits. If the process element is able to schedule V_i, it estimates provisioning energy and cost. As the provisioning policy aimed to provide a lower price for the user, the proposed algorithm in this paper discovered the minimum-price processor. For the same price, less energy is preferable because it produces higher profit. Finally, a VM is mapped if V_i is schedulable on the datacenter. However, in this study, the soft real-time VM provisioning did not considered.

The authors in [40] stated the previous real time job scheduling algorithms were running in uncertain cloud environments. Those algorithms assumed that cloud computing environments were deterministic, and there were statistical pre-computed schedule decisions to be followed during the schedule execution. So this study introduced the interval number theory to describe the uncertainty of the cloud computing environment and the impact of uncertainty on the scheduling quality in a cloud data center. Accordingly, a novel scheduling algorithm, called Proactive and Reactive Scheduling (PRS), was presented. It dynamically exploited proactive and reactive scheduling methods for scheduling real-time jobs.

The proactive scheduling was used to build baseline schedules depending on redundancy, where a protective time cushion between jobs' finish time lower bounds and their deadlines was added to guarantee job deadlines. The reactive scheduling was triggered to generate proactive baseline schedules in order to account for various disruptions during the course of executions.

Some strategies were presented to scale the system's computing resources up and down according to workload to improve resource utilization and to reduce energy consumption for the cloud data center. These strategies were proposed to treat the following five events as disruptions:

1) a new job arrives;
2) the system becomes overloaded;
3) a new urgent job arrives or the waiting jobs become urgent;
4) a VM finishes a job;
5) some VMs' idle time exceeds the pre-established threshold.

However, estimating job execution time is main factor in the scheduling model proposed in this work. So, improving the precision of estimated job execution time may lead to better scheduling decisions.

The authors in [41] presented two scheduling algorithms for precedence-constrained parallel VMs in a virtualized data center. The proposed algorithms used a new insertion policy to insert VMs among previously scheduled ones. The new policy inserted VMs into already switched on low utilized PMs to increase their utilization, thus reducing the total number of switched on PMs that served the VMs, and therefore enhancing energy efficiency.

The first algorithm, called Virtualized Homogeneous Earliest Start Time (VHEST), was an extension of the well-known HEFT algorithm. HEFT scheduled VMs according to the non-overlapping insertion policy. VHEST was modified to use overlapping insertion policy to minimize the makespan.

It had two major stages: VM selection and PM Selection. At the first stage, VMs were sorted according to their priority, and then the VM with the highest priority was selected to be placed on PM. At the second stage, the selected VM was placed in the best PM that minimized the VM's start time by applying an overlapping-insertion policy.

The second algorithm, called Energy-Aware Scheduling Algorithm (EASA), solved a multi-objective problem. It improved the utilization of PMs and minimized the makespan. EASA also had two major stages: Local optimization and Global Optimization. The local optimization stage improved the utilization of the PMs. The global optimization stage reduced the number of switched on PMs by switching off the underutilized ones. However, the work did not support heterogeneous data centers.

An energy-aware resource provisioning framework for cloud data centers was proposed in [42]. The main functions of the proposed framework can be summarized into three points:

1) Predicting the upcoming number of the (VM) requests that would arrive to the cloud data center in a certain future period, associated with the requirements for each VM. The prediction approach relied upon monitoring past workload variations during a period of time. It combines machine learning clustering and stochastic theory to predict the number of the upcoming VM requests along with required resources associated with each request.
2) Estimating the number of PMs needed in the data center that will serve the upcoming users' requests. This estimation is based on the predictions of such requests.
3) Turning the unneeded PMs in the data center to the sleep mode by applying intelligent power management decisions in order to reduce the consumed energy.

In addition to the previous studies, various optimization methods such as, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithms (GA)) were used to improve resource utilization and reduce energy consumption in the virtualized data centers.

In [43], the researchers presented a framework to manage the VM placement in an IaaS environment. They defined an initial VM placement strategy and proposed multi-objective optimization algorithm based on (ACO) to determine the initial VMs placement. The proposed algorithm was an optimization method. It was able to achieve an optimal solution through efficient convergence by the constantly updated pheromone. The optimal solution was selected from a set of solutions using the exclusion method.

In [44], the authors designed a distributed ACO-based algorithm for solving the VM consolidation problem (VMCP). The algorithm iterated over a finite number of cycle. At each cycle, an item was selected for each ant to be packed in a bin. If the bin did not have enough space according to defined constraints, another bin was selected. At the end of each cycle, the best solution found was saved and pheromone evaporation was triggered to achieve the VM consolidation.

In [45], the authors proposed a multi-objective ACO to solve the problem of VM placement. The formulated multi-objective VM placement problem represented a permutation of VM assignment. The goal was to efficiently obtain a set of non-dominated solutions that reduced total power consumption resulting from resource wastage. This proposed model has two phases; Initialization phase: where the parameters and the pheromone trails were initialized. Iterative phase: in this phase, all VM requests were sent to the ants to start assigning VMs to the selected PMs. This was done by using a pseudo-random-proportional rule, which described the way each ant selected a particular next one VM pack into its current PM. Ants moved towards the most promising VMs based on information about the current pheromone concentration during the movement. Local and global pheromones were frequently updated. A local pheromone update was performed once an artificial ant built a movement. Then, a global update was performed with each solution of the current Pareto set after all ants had constructed their solutions.

The thesis in [46] focused on the IaaS cloud service model. This model offered compute infrastructure to cloud consumers by provisioning VMs on-demand. The

thesis investigated the challenge of designing, implementing, and evaluating an energy-efficient IaaS cloud management system for private clouds. The author proposed four contributions:

1) IaaS Cloud Management system (Snooze): It was based on a self-configuring hierarchical architecture and performed VM management for distributed large-scale virtualized data centers by splitting the data center into independently managed groups of PMs and VMs. Snooze provided a holistic energy-efficient VM management solution. Particularly, it integrated a power management mechanism which automatically detected idle PMs, transitioned them into a power-saving state, and woke them up on demand.
2) VM Placement via ACO: an ACO-based VM placement algorithm was used to solve the problem of considering only a single resource to evaluate the PM load and VM resource demands while ignoring the other resources. In addition, ACO appealed to the VM placement problem due to its polynomial time worst-case complexity and convenience of parallelization.
3) VM Consolidation via Ant Colony Optimization: VM consolidation algorithms are required in order to enable continuous consolidation of already placed VMs on fewer PMs. This consolidation helped avoid resource fragmentation and further increases the data center resource utilization. Therefore, the researcher proposed a consolidation algorithm based on ACO to achieve both scalability and high data center utilization by applying VM consolidation.
4) Fully decentralized consolidation system based on an unstructured peer-to-peer network.

In [47], the authors proposed a VM consolidation scheme that focused on balanced resource utilization of servers across different computing resources with the goal of minimizing power consumption and resource wastage. This study presented an adaptation and integration of the ACO met heuristic with a balanced usage of computing resources. The degree of imbalance in the current resource utilization of a PM was captured and represented as a multi-dimensional server resource utilization, and then resources utilization are balanced using vector algebra.

The study in [48] proposed a PSO-based algorithm, which could successfully reduce the energy cost and the time for searching feasible solutions. The authors presented an environment of heterogeneous multiprocessors, which is similar to the environment of cloud data center, and they proposed a job to the processor assignment model that would work in that environment. During assigning jobs to the processors, the velocity of the particles (that represented the jobs) determined their positions. This velocity will affect the overall convergence of the PSO algorithm and the efficiency of the algorithm's global searching. The particle's position updates present the next position of the job. As the particle position updates, it indicated that it needed to adjust the number of processors that fit its requirements. Then, the proposed algorithm optimized the most feasible solution to in order to reduce energy consumption by assigning jobs to new processors or by exchanging their corresponding processors. However, there are many constraint conditions in the work, reducing them will make the work more convenient for solving the problem of real-time job scheduling.

In [49], the authors proposed a genetic algorithm for power-aware (GAPA) scheduling to find the optimal solution for the problem of VM allocation. In the proposed algorithm, a tree structure was used to encode chromosome of an individual job. The fitness function of GA calculated the evaluation value of each chromosome. The tree had three levels; Level 1: Consisted of a root node that did not have a significant meaning, Level 2: Consisted of a collection of nodes that represented a set of PMs, Level 3: Consisted of a collection of nodes that represented a set of virtual machines. Using this model, each instance of tree structure showed the VM to PM Allocation. However, the computational time of the GAPA is high, also deadline of jobs did not considered in the work.

In [50], the authors proposed a distributed parallel genetic algorithm (DPGA) of placement strategy for VMs deployment on cloud platform. The proposed algorithm had two stages: It executed the genetic algorithm in parallel and in a distributed manner on several selected PMs in the first stage to get several solutions. Then, it continued to execute the genetic algorithm of the second stage with solutions obtained from the first stage as the initial population. A relatively optimal job to VM mapping was obtained as a result of the second stage. The fitness value of GA chosen here was performance per watt.

In [51], the authors introduced a power efficient resource allocation algorithm for jobs in cloud computing data centers. The developed approach was also based on GA. Resource allocation was performed to optimize job completion time and data center power consumption. It considered a static scheduling of independent jobs on homogeneous single-core resources. The proposed algorithm, called Non-dominated Sorting Genetic Algorithm II (NSGA-II), was applied to cloud environments to explore space solutions and efficiently search for the optimal solution. The data center was modeled as three-tier fat-tree layers architecture: access, aggregation, and core layers. The access layer provided

Table 3 Studies of power efficiency improvement in virtualized data centers

Ref.	Virtualized Data Center		Environment		All Workload Types	Power Aware Scheduling		Resources					Approach		Scheduling		Technology/Method
	Single	Multiple	Homo	Hetero		Consider Cost	Consider Time	CPU	Memory	Storage	Network.	Cooling System	SW	HW	Offline	Online	
[23]	√		√		√			√					√	√		√	DVFS, SW-based Model, VM Consolidation, On/Off Switching.
[24]	√			√	√			√					√	√		√	DVFS, SW-based Model, VM Consolidation, On/Off Switching.
[25]	√		√		√	√		√					√	√		√	DVFS, SW-based Model, VM Consolidation, On/Off Switching.
[26]	√			√	√	√		√					√	√		√	DVFS, SW-based Model, VM Consolidation, On/Off Switching.
[27]	√		√		√			√	√				√			√	WS-based Model, On/Off Switching.
[28]	√		√		√			√	√				√			√	VM Consolidation, On/Off Switching.
[29]	√		√					√					√	√		√	DVFS, SW-based Model.
[30]	√		√		√		√	√	√		√		√		√		SW-based Model, VM Consolidation
[31]	√		√		√	√	√	√	√				√		√		SW-based Model
[32]	√		√		√			√	√				√			√	SW-based Model
[33]	√			√				√					√			√	SW-based Model, VM Consolidation
[34]	√			√	√	√		√					√	√		√	DVFS, SW-based Model.
[35]	√		√		√			√					√			√	SW-based Model, On/Off Switching.
[36]	√		√		√			√	√				√			√	SW-based Model, VM Consolidation, On/Off Switching.
[37]	√		√		√			√					√			√	SW-based Model, VM Consolidation.
[38]	√		√		√			√					√			√	SW-based Model, VM Consolidation, On/Off Switching.
[1]		√		√	√			√					√	√		√	DVFS, SW-based Model, VM Consolidation, On/Off Switching.
[39]	√			√	√	√		√					√	√		√	DVFS, SW-based Model.
[40]		√	√		√		√	√	√				√			√	SW-based Model.
[41]	√			√	√		√	√					√			√	SW-based Model, VM Consolidation.
[42]	√			√	√			√	√				√			√	SW-based Model.
[43]	√		√		√		√	√	√		√		√				ACO
[44]	√			√	√		√	√	√	√			√			√	ACO, VM Consolidation
[45]	√		√		√			√	√				√			√	ACO, VM Consolidation
[46]	√			√	√			√					√		√		ACO, VM Consolidation, On/Off Switching.
[47]	√		√		√			√	√		√		√			√	ACO, VM Consolidation, On/Off Switching.
[48]	√			√	√		√	√					√			√	PSO
[49]	√		√		√			√					√			√	GA
[50]	√		√		√	√		√					√			√	GA
[51]	√		√		√		√	√			√		√			√	GA

connection to servers which were arranged into racks with each rack being served by a single Top of the rack switch. Two fitness functions were used here: task completion time and data center power consumption. When the execution of the algorithm was completed and optimal Pareto solutions were obtained, it became possible to fine tune the trade-off between power consumption and execution time. Then, by using a procedure called ranking, the population of solutions were sorted heuristically into different non-domination levels. This procedure was repeated for every solution creating different groups or non-domination; an integer value called rank was assigned to each non-domination level. When applying selection and sorting, NSGA-II was able to deal with constraints. The solution with less constraint violation had a better rank.

Table 3 [1, 23-51] summarizes all the studies and techniques illustrated in this survey. The table compares the studies from many perspectives. It shows if the studies are applied in single or multiple data centers, Homogenous or heterogeneous environment, Specific or all types of jobs are used, if any other parameters are considered with enhancing the energy efficiency, the involved resources in enhancing the energy efficiency, HW or SW approach is applied with the study, the scheduling is online or offline, and finally, what is the used technology in each study?

Conclusion

By studying the research works mentioned in this survey paper, it can be concluded that:

1) In order to improve energy efficiency in DPM software solutions, the only two issues that have to be investigated in cloud data centers are: jobs or tasks to VMs allocation, and VMs to PMs placement.
2) The utilization of the PMs is a very important factor to be considered in proposing the resource management solutions. The PMs utilizations affect the energy efficiency in the cloud data centers. In general, the best energy efficiency occurs at optimal utilization and it drops as utilization decreases.
3) VM migrations usually occur when there is over/under utilization of the resources. Extra VM migration may affect the whole system performance, leading to further power consumption. So, VM management (e.g VM allocation and VM placement) is a very critical process that should be optimally done to avoid unnecessary VM migration.
4) Before proposing the energy aware scheduling algorithms, it is important to understand the capacity of the resources and the types of services provided by the cloud data centers to avoid any waste of the available resources capacities.
5) When proposing the energy aware algorithms, identifying the behaviors of cloud users' requests and common workloads patterns would improve the system performance, and therefore, enhance energy efficiency in the cloud data centers.
6) Although modern advances in hardware technologies have reduced energy consumption to some extent, many software approaches have been proposed for further improvements. The two directions must be considered as complementary approaches, and applying both of them (hardware and software) in any proposed model leads to more reduction in energy consumption in the cloud data centers.

We do believe that this survey is a good guideline for researchers in designing the energy aware algorithms that execute the users' jobs in cloud data centers.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

Auday explained the taxonomy and methodology, carried out the study on power efficiency in cloud data centers, and drafted the manuscript. Wassim discussed, revised and added sections to the manuscript. Ahmed and Rached provided notes and revised the manuscript and gave a useful feedback. All authors read and approved the final manuscript.

Acknowledgements

The authors wish to thank Dr Lama Hamandi for offering insightful suggestions and providing many references.

Author details

[1]Department of Mathematics & Computer Science, Beirut Arab University, Beirut, Lebanon. [2]Department of Electrical & Computer Engineering, Beirut Arab University, Beirut, Lebanon. [3]Department of Mathematics & Computer Science, Faculty of Science, Alexandria University, Alexandria, Egypt. [4]Department of Electrical & Computer Engineering, Rafic Hariri University, Beirut, Lebanon.

Received: 14 November 2015 Accepted: 5 April 2016
Published online: 27 April 2016

References

1. Beloglazov A (2013) Energy-efficient management of virtual machines in data centers for cloud computing, PHD thesis. Department of Computing and Information Systems, The University of Melbourne
2. Koomey J (2007) Estimating total power consumption by servers in the us and the world, Lawrence Berkeley National Laboratory, Technical Report
3. Dorf R, Svoboda J (2010) Introduction To Electric Circuits, 8 edn. John Wiley & Sons Inc, USA; ISBN: 978-0-470-52157-1
4. Orgerie A, Assuncao M, Lefevre L (2014) A survey on techniques for improving the energy efficiency of large scale distributed systems. ACM Comput Surv 46(4):1–35
5. Venkatachalam V, Franz M (2005) Power reduction techniques for microprocessor systems. ACM Comput Surv 37(3):195–237
6. Burd T, Brodersen R (1996) Processor design for portable systems. J VLSI Signal Processing 13(2–3):203–221
7. George J (2005) Energy-optimal schedules of real-time jobs with hard deadlines, Msc Thesis. Texas A&M University, Texas
8. Barroso LA, Hölzle U (2007) The case for energy-proportional computing. Computer 40(12):33–37
9. Cupertino L, Costa GD, Oleksiak A, Piatek W, Pierson J, Salom J, Sisó L, Stolf P, Sun H, Zilio T, part B (2015) Energy-efficient, thermal-aware modeling and

simulation of data centers: the coolemall approach and evaluation results. Ad Hoc Netw 25:535–553
10. Avelar V, Azevedo D, French A (2012) PUE: a comprehensive examination of the metric
11. Andersen D, Franklin J, Kaminsky M, Phanishayee A, Tan L, Vasudevan V (2009) FAWN: A Fast Array of Wimpy Nodes. In: 22nd ACM Symposium on Operating Systems Principles
12. Vasudevan V, Andersen D, Kaminsky M, Tan L, Franklin J, Moraru I (2010) Energy-efficient cluster computing with FAWN: workloads and implications. In: 1st International Conference on Energy-Efficient Computing and Networking
13. Caulfield A, Grupp L, Swanson S (2009) Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. In: 14th international conference on Architectural support for programming languages and operating systems
14. Valentini G, Lassonde W, Khan S, MinAllah N, Madani S, Li J, Zhang L, Wang L, Ghani N, Kolodziej J, Li H, Zomaya A, Xu C, Balaji P, Vishnu A, Pinel F, Pecero J, Kliazovich D, Bouvry P (2013) An overview of energy efficiency techniques in cluster computing. Clust Comput 16(1):3–15
15. Tiwari V, Ashar P, Malik S (1993) Technology mapping for low power. In: 30rd design automation conference
16. Su CL, Tsui CY, Despain AM (1994) Saving power in the control path of embedded processors. IEEE Design & Test of Computers 11(4):24–31
17. Benini L, Bogliolo A, Micheli G (2000) A survey of design techniques for system-level dynamic power management. IEEE Transact Very Large Scale Integration Systems 8(3):299–316
18. Kuo C, Lu Y (2015) Task assignment with energy efficiency considerations for non-DVS heterogeneous multiprocessor systems. ACM SIGAPP Appl Comput Rev 14(4):8–18
19. Snowdon D, Ruocco S, Heiser G (2005) Power management and dynamic voltage scaling: myths and facts. In: The workshop on power aware real-time computing
20. Duflot L, Levillain O, Morin B (2009) ACPI: Design Principles and Concerns. In: the 2nd International Conference on Trusted Computing. Berlin
21. Toshiba, Compaq, Intel, Microsoft and PhoenixLTD (2013) Advanced configuration and power interface specification : Revision 5.0a
22. Fan X, Weber W, Barroso L (2007) Power provisioning for a warehouse-sized computer. In: the 34th annual International symposium on computer architecture
23. Nathuji R, Schwan K (2007) VirtualPower: coordinated power management in virtualized enterprise systems. ACM SIGOPS Oper Syst Rev 41(6):265–278
24. Nathuji R, Isci C, Gorbatov E (2007) Exploiting platform heterogeneity for power efficient data centers. In: The 4th International conference on autonomic computing
25. Raghavendra R, Ranganathan P, Talwar V, Wang Z, Zhu X (2008) No power struggles: coordinated multi-level power management for the data center. SIGARCH Compr Architecture News 36(1):48–59
26. Verma A, Ahuja P, Neogi A (2008) pMapper: power and migration cost aware application placement in virtualized systems. In: the 9th ACM/IFIP/USENIX International Conference on Middleware
27. Kusic D, Kephart J, Hanson J, Kandasamy N, Jiang G (2009) Power and performance management of virtualized computing environments via lookahead control. Clust Comput 12(1):1–24
28. Gmach D, Rolia J, Cherkasova L, Kemper A (2009) Resource pool management: reactive versus proactive or let's be friends. Comput Netw 53(17):2905–2922
29. Cardosa M, Korupolu M, Singh A (2009) Shares and utilities based power consolidation in virtualized server environments. In: the 11th IFIP/IEEE International Symposium on Integrated Network Management
30. Kumar S, Talwar V, Kumar V, Ranganathan P, Schwan K (2009) vManage: loosely coupled platform and virtualization management in data centers. In: the 6th International conference on autonomic computing
31. Jung G, Joshi K, Hiltunen M, Schlichting R, Pu C (2009) A cost-sensitive adaptation engine for server consolidation of multitier applications. In: the ACM/IFIP/USENIX International conference on middleware
32. Song Y, Wang H, Li Y, Feng B, Sun Y (2009) Multi-tiered on-demand resource scheduling for Vm-based data center. In: the 9th IEEE/ACM International symposium on cluster computing and the grid
33. Stillwell M, Schanzenbach D, Vivien F, Casanova H (2009) Resource allocation using virtual clusters. In: the 9th IEEE/ACM International symposium on cluster computing and the grid
34. Buyya R, Beloglazov A, Abawajy J (2010) Energy-efficient management of data center resources for Cloud computing: a vision, architectural elements, and open challenges. In: the International conference on parallel and distributed processing techniques and applications
35. Lefevre L, Orgerie A (2010) Designing and evaluating an energy efficient cloud. J Super Comput 51(3):352–373
36. Gulati A, Holler A, Ji M, Shanmuganathan G, Waldspurger C, Zhu X (2012) VMware distributed resource management: design, implementation, and lessons learned. VMware Tech J 1(1):45–64
37. Li X, Qian Z, Lu S, Wu J (2013) Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in data center. Mathematical Computer Modeling 58(5–6):1222–1235
38. VIKRAM R, NEELIMA A (2013) Resource over allocation to improve energy efficiency in real-time cloud computing data centers. Int J Advanced Trends in Compr Sci Eng 2(1):447–453
39. Kim K, Beloglazov A, Buyya R (2011) Power-aware provisioning of virtual machines for real-time cloud services. Pract Experience Concurrency Computation 23(13):1491–1505
40. Chen H, Zhu X, Guo H, Zhu J, Qin X, Wu J (2015) Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment. J Syst Softw 99:20–35
41. Ebrahimirad V, Goudarzi M, Rajabi A (2015) Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers. J Grid Computing 13(2):233–253
42. Dabbagh M, Hamdaoui B, Guizaniy M, Rayes A (2015) Energy-efficient resource allocation and provisioning framework for cloud data centers. IEEE Trans Netw Serv Manag 12(3):377–391
43. Ma F, Liu F, Liu Z (2012) Multi-objective optimization for initial virtual machine placement in cloud data center. Journal of Information & Computational Science 9(16):5029–5038
44. Esnault A (2012) Energy-Aware Distributed Ant Colony Based Virtual Machine Consolidation in IaaS Clouds. Distributed, Parallel, and Cluster Computing [cs.DC]. HAL ID:dumas-00725215 version 1
45. Gao Y, Guan H, Qi Z, Hou Y, Liu L (2013) A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 79(8):1230–1242
46. E. Feller (2013) Autonomic and Energy-Efficient Management of Large-Scale Virtualized Data Centers, PhD Thesis. University of Rennes, ISTIC
47. Ferdaus M, Murshed M, Calheiros R, Buyya R (2014) Virtual Machine Consolidation in Cloud Data Centers Using ACO Metaheuristic. In: Euro-Par, the 20th International Conference of Parallel Processing. Porto, Portugal; Springer International Publishing, 306-317
48. Zhang W, Xie H, Cao B, Cheng A (2014) Energy-aware real-time task scheduling for heterogeneous multiprocessors with particle swarm optimization algorithm. Math Probl Eng 2014:Article ID 287475
49. Quang-Hung N, Nienz P, Namz N, Tuong N, Thoa N (2013) A genetic algorithm for power-aware virtual machine allocation in private cloud. Lecture Notes in Compr Sci 7804:170–179
50. Dong Y, Xu G, Fu X (2014) A distributed parallel genetic algorithm of placement strategy for virtual machines deployment on cloud platform. Sci World J 2014:Article ID 259139
51. Portaluri G, Giordano S, Kliazovich D, Dorronsoro B (2014) A power efficient genetic algorithm for resource allocation in cloud computing data centers. In: IEEE 3rd International conference on cloud networking