energies

MDPI

Article

Lifetime-Aware Cloud Data Centers: Models and Performance Evaluation

Luca Chiaraviglio [1,*], Antonio Cianfrani [1], Marco Listanti [1], William Liu [2] and Marco Polverini [1]

[1] Department of Information, Electronics and Telecommunications, University of Rome Sapienza, Rome 00185, Italy; antonio.cianfrani@uniroma1.it (A.C.); marco.listanti@uniroma1.it (M.L.); marco.polverini@uniroma1.it (M.P.)

[2] Department of Information Technology and Software Engineering, School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, 55 Wellesley Street East, Auckland 1010, New Zealand; william.liu@aut.ac.nz

* Correspondence: luca.chiaraviglio@uniroma1.it; Tel.: +39-644585371; Fax: +39-64744481

Academic Editor: Silvio Simani
Received: 3 March 2016; Accepted: 13 June 2016; Published: 18 June 2016

Abstract: We present a model to evaluate the server lifetime in cloud data centers (DCs). In particular, when the server power level is decreased, the failure rate tends to be reduced as a consequence of the limited number of components powered on. However, the variation between the different power states triggers a failure rate increase. We therefore consider these two effects in a server lifetime model, subject to an energy-aware management policy. We then evaluate our model in a realistic case study. Our results show that the impact on the server lifetime is far from negligible. As a consequence, we argue that a lifetime-aware approach should be pursued to decide how and when to apply a power state change to a server.

Keywords: data-centers (DCs); server lifetime models; energy and lifetime trade-off; failure models

1. Introduction

The traffic volume towards data-centers (DCs) has been constantly increasing [1] during the last ten years. This is due to the fact that the "cloud" paradigm has gained importance in the Internet. As an example, different big players (like Google, Amazon, Dropbox) are offering cloud storage, thus allowing both single users and whole organizations to move data from/to the cloud. In this context, the DC architecture and the management of the servers inside the DC has gained a significant role. In particular, service providers managing DCs are facing two contrasting issues: (i) the requests from users need to be served in a reasonable bounded delay; and (ii) the energy costs for managing the DC have to be reduced. In the literature, both these issues have been extensively studied (e.g., [2,3]), focusing both on intra-DC solutions (*i.e.*, involving the servers inside a single DC), or inter-DC ones (*i.e.*, considering as a whole the set of DCs managed by the cloud provider).

When the objective functions like the minimization of delay or energy are pursued, different power states are applied to the servers of the DC. More in depth, a subset of servers may be even put in sleep mode (SM) to maximize the electricity savings [4]. Although the benefits from changing the server power state in a DC are clear (*i.e.*, electricity cost reduction or performance improvement), the implications of applying such policies in real DC implementations have been only marginally studied. In this context, several question arise, such as: What is the impact of power management policies on the lifetime on servers? Is it possible to define a model for the lifetime of a server subject to power state changes? Specifically, the lifetime is defined as the amount of time between one failure and the following one [5]. When the server lifetime is decreased, its failure rate is increased, thus negatively impacting the user's quality of experience (QoE). On the other hand, the related costs incurred for

fixing the failed servers may also erode the electricity saved by the DC service provider. The scope of this work is therefore to investigate this impact by proposing a primary model of the lifetime for a set of servers subject to power state changes. Moreover, we evaluate the proposed model in different case-studies driven by energy minimization and/or delay minimization goals. Our results indicate that the server's lifetime should be carefully considered when power management policies are applied—*i.e.*, how and when to apply a server power state in a DC. We believe that this work could pave a new way to further investigation. In particular, we show that the lifetime is also heavily influenced by the hardware (HW) parameters used to build the servers, and therefore a careful measurement of these parameters in a working scenario is of mandatory importance. Additionally, we believe that new power management algorithms considering user performance, energy minimization, and server lifetime should be developed. We leave both these aspects for future work. The rest of the paper is organized as follows. Related works are presented in Section 2. The main effects related to temperature which impact server lifetime are summarized in Section 3. Section 4 then presents our model for estimating the server lifetime. Section 5 reports the scenario adopted to evaluate our model. Results, obtained from a realistic case study, are detailed in Section 6. Finally, Section 7 reports the conclusions and the future work.

2. Related Work

The closest paper to this work is [5], in which authors propose a simple model for evaluating the impact of energy-aware policies on the lifetime of cellular and backbone network devices. However, the research in cellular and backbone devices is generally different than the one developed for the server case. For example, the total power consumption of network devices hardly scales with the load [6], while the server machines tend to be more power proportional. In this work, we go three steps further by:

(i) focusing on a DC scenario;
(ii) proposing a completely new model that accounts for transitions among different power states (in other words, not only a SM and full power like in [5]);
(iii) evaluating the impact of applying a management policy for the servers in a DC.

We believe that these improvements are of fundamental importance for understanding the energy, quality of service (QoS), and lifetime trade-off in cloud DCs. Our model is rather general, and, to the best of our knowledge, is representative for the server case. However, the physical phenomena affecting the lifetime of microelectronics components (like chip packages) are common in both backbone, cellular, and server devices. Moreover, we also add that current trends in backbone and cellular network devices foresee the exploitation of commodity HW (thanks to the introduction of new paradigms like Software Defined Networking [7] and Network Function Virtualization [8]). This HW will likely operate like server machines, so we envision the application of the model presented in this paper even in these cases.

In general, the variation of power (and consequently of temperature), has an impact on device lifetime. More in depth, in [9] authors report temperature measurements of disks, memories, and components in DCs, showing that the largest impact on failure rates is due to the variation of temperature rather than its increase. Moreover, the authors show that the failure rate of HW components is linear with temperature rather than exponential. Finally, a linear dependency of power with temperature is shown. In our work, we have considered all these aspects. In particular, we have proposed a failure model taking into account power variation. Additionally, we have assumed that the failure rate increases linearly with temperature.

The management of jobs between different DCs may pursue different goals. Specifically, in [10] authors exploit the variation of electricity prices in order to reduce the associated costs in DCs. More in depth, the servers are geographically distributed among different zones. In this scenario, the job requests are moved among the different DCs in order to exploit the DC(s) experiencing the lowest

electricity prices. Additionally, in [11] authors take into account capacity provisioning in cloud DCs with the goal of reducing the brown energy—*i.e.*, energy generated from non-renewable sources. Finally, in [12] authors consider the consumption of water for dispatching jobs in geographical DCs. All of these works do not consider the impact of the proposed policies on the failure rate of the servers in each DC. In contrast to them, here we focus on the impact of a policy to manage jobs in cloud DCs on server lifetime.

Reducing power consumption in a single DC is a hot topic. In the past years, different works have targeted this problem (for example, [13] for a detailed survey). In particular, Beloglazov and Buyya [4] have considered the consolidation of virtual machines (VMs) in order to bring substantial energy savings while ensuring reliable QoS. Additionally, Heller *et al.* [14] and Fanf *et al.* [15] targeted the energy reduction of the networking infrastructure. All these works prove the efficacy and the efficiency of these solutions in terms of energy, without considering the impact on the failure rate due to power state transitions. Our work is instead devoted to the investigation of the server lifetime in each DC of the cloud provider. Although there are several objectives (*i.e.*, user QoE, energy inside the DC, energy among a set of DCs, electricity costs, brown energy, water consumption) that can be pursued during the management of the DC, we believe that the impact of failure rate should be also controlled. In the next section, we therefore investigate the main physical phenomena affecting the server failure rate and consequently its lifetime.

3. Physical Factors Affecting Server Lifetime

A first order model to compute the failure rate γ^T of a device given its temperature T is the Arrhenius law [16]:

$$\gamma^T = \gamma^0 \mathrm{e}^{-\frac{E_\mathrm{a}}{KT}} \ (1/\mathrm{h}) \tag{1}$$

where γ^0 is the failure rate estimated assuming a very high temperature, E_a is the activation energy (*i.e.*, the minimum energy needed to activate the failure variation), and K is the Boltzmann constant. Interestingly, when the temperature is decreased, the failure rate is also decreased. Although more detailed models have been proposed in the literature (e.g., [17]) all of them predict a decrease in the failure rate when the temperature is reduced. This means that if the reduction of temperature was the only effect taken under consideration, keeping the server in the lower power states for the longest amount of time would be of benefit for its lifetime. However, we stress that Equation (1) is just a first-order approximation. For example, according to [9], failure rates exhibit a linear dependency with temperature (rather than exponential). Therefore, we will consider this aspect when defining a model for the server failure rate.

In the following, we consider the second effect impacting the lifetime—*i.e.*, the temperature variation. In this case, a device may suffer strain and fatigue when temperature conditions change, especially when this happens in a cyclical fashion (*i.e.*, from one temperature to another). The Coffin–Manson model [18] describes the effects of material fatigue caused by cyclical thermal stress. The predicted failure rate $\gamma^{\Delta T}$ due to the thermal cycling effect is then expressed as:

$$\gamma^{\Delta T} = \frac{f^{\mathrm{TC}}}{N^f} \ (1/\mathrm{h}) \tag{2}$$

where f^{TC} is the frequency of thermal cycling and $\gamma^{\Delta T}$ is the estimated failure rate. The term N^f is the number of cycles to failure, and it is commonly denoted as:

$$N^f = C_0(\Delta T - \Delta T_0)^{-q} \tag{3}$$

where ΔT is the temperature variation of the cycle, ΔT_0 is the maximum admissible temperature variation without a variation in the failure rate, C_0 is a constant material dependent, and q is the Coffin–Manson exponent. From this model, we can clearly see that the more often the server

experiences a temperature variation, the higher its failure rate will be. In the following, we therefore build a model to capture this effect and also to consider the impact of temperature decrease reported in Equation (1).

4. Server Lifetime Model

We first consider the model for a single server, then we detail the steps to extend this model to the whole DC under consideration. In particular, we assume that the total period of time under consideration is denoted as T, divided in time slots. Each time slot $k \in T$ is denoted by a duration Δ_k. Moreover, we assume $\sum_{k=1}^{|T|} \delta_k = T$—*i.e.*, the time needed to perform transitions is negligible with respect to the time slot duration. The power consumption of the server at time slot k is denoted as $P(k)$, which is equal to 0 if the server is put in SM, or a value between $P^{\min}$ and $P^{\max}$ if the server is active. $P(k)$ takes into account the whole power consumed by the server. This term scales linearly with the server load—*i.e.*, it is $P^{\min}$ for the minimum load and may be increased up to $P^{\max}$ for the maximum load. When the load is equal to zero, we assume that the server is in SM—*i.e.*, $P(k) = 0$.

We then consider the impact of power state transitions. Specifically, when the server load is varied across two consecutive time slots, a power state change of the whole server is triggered. Let us denote $\delta_P(k)$ as the absolute difference of the power consumption at k-th time slot and the one at the $(k-1)$-th time slot—*i.e.*, $\delta_P(k) = |P(k) - P(k-1)|$. Moreover, let us denote $\delta_{P^{\max}}$ as the maximum difference in power consumption for active states—*i.e.*, $\delta_{P^{\max}} = |P^{\max} - P^{\min}|$.

We then introduce the binary variables used in our model. In particular, we denote $x(k)$ as the SM state setting at time k:

$$x(k) = \begin{cases} 1 & P(k) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Moreover, we introduce the binary variable $y(k)$, which takes value equal to one if the server has experienced a SM-active transition between the current time slot and the previous one, zero otherwise:

$$y(k) = \begin{cases} 1 & (P(k) = 0 \text{ and } \delta_P(k) > 0) \text{ or } (P(k-1) = 0 \text{ and } \delta_P(k) > 0) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Note that an alternative definition of $y(k)$ is $y(k) = x(k-1) \oplus x(k)$, where $\oplus$ is the exclusive disjunction (XOR) operator.

Finally, we introduce binary variable $z(k)$, which takes value one if the server has experienced an active power transition between the current time slot and the previous one, zero otherwise.

$$z(k) = \begin{cases} 1 & P(k) > 0 \text{ and } P(k-1) > 0 \text{ and } \delta_P(k) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We then consider a metric called acceleration factor (AF), introduced in [5]. The AF is denoted as the ratio between the current failure rate (*i.e.*, by applying power management policies) and a fixed reference failure rate (which we assume to be the case in which the server is always kept at full power). Therefore, the AF is simply a normalized failure rate. The failure rate can be obtained by multiplying the AF for the reference failure rate. Specifically, the AF takes into account the lifetime decrease/increase introduced by energy-aware policies. In particular, if the AF is lower than one, the lifetime is increased compared to a full power situation. On the contrary, if the AF is higher than one, the lifetime is decreased. We refer the reader to [5] for a detailed explanation of this metric. In this work, we will decompose the AF in different sub-terms, due to the following effects:

(i) SM state setting;
(ii) active power setting;
(iii) power state transition between the active states (not involving SM);
(iv) power state transitions between an active state and a SM state.

The total AF is the sum of the AF due to power states AF_s and the AF due to transitions AF_t:

$$AF = AF_s + AF_t \tag{7}$$

In particular, AF_s depends solely on the amount of time spent in each power state and the AF of that state. At the same time, AF_t depends solely on the frequency of power state transitions and the penalties for the transitions.

More in depth, AF_s can be decomposed in the following way:

$$AF_s = AF_s^{off} + AF_s^{active} \tag{8}$$

where AF_s^{off} accounts for the AF in SM over the period T, while AF_s^{active} takes under consideration the AF of active transitions over the period T. In particular, AF_s^{off} can be expressed as follows:

$$AF_s^{off} = \sum_{k=1}^{|T|} AF^{off} x(k) \frac{\Delta_k}{T} \tag{9}$$

where AF^{off} is the AF of the server in SM (without considering the time period). Thus, AF_s^{off} is the AF in SM, weighed by the amount of time the server spends in SM. Focusing on AF_s^{active}, we can express it in the following way:

$$AF_s^{active} = \sum_{k=1}^{|T|} (1 - x(k)) \frac{\Delta_k}{T} \left[\frac{(1 - AF^{off})}{(P^{max} - P^{min})} (P(k) - P^{min}) + AF^{off} \right] \tag{10}$$

In particular, we have considered that the AF due to active state is extracted by interpolation between the minimum amount of power P^{min} and the maximum one P^{max}. The intuition suggests in fact that the server temperature is lower when the power is decreased. Therefore, we also expect a lower AF. Additionally, we assume that the AF at maximum power is equal to one. (Equation (10) may also be rewritten to include Equation (9), but we prefer to keep both of them in order to evaluate them separately in the results section.) The reasons for choosing a linear dependency of AF with power are as follows:

(i) the power tends to increase linearly with temperature (as reported by [9]);
(ii) the exponential model presented in Equation (1) tends to over-estimate the failure rate increase with temperature.

Therefore, a more realistic model is the linear one.

In the next step, we consider the AF due to transitions, expressed as:

$$AF_t = AF_t^{off} + AF_t^{active} \tag{11}$$

where AF_t^{off} is the AF due to an SM-active state transitions, and AF_t^{active} is the AF due to transition between active power states. In particular, we define AF_t^{off} as:

$$AF_t^{off} = \sum_{k=1}^{|T|} y(k) \frac{\chi^{off}}{T} \tag{12}$$

where χ^{off} is a weight parameter for the frequency of SM-active transitions $\sum_{k=1}^{|T|} y(k) \frac{1}{T}$. This parameter depends solely on the HW used to build the device. Additionally, we define AF_t^{active} as:

$$AF_t^{active} = \sum_{k=1}^{|T|} z(k) \frac{W \chi^{off} \delta_P(k)}{\delta_P^{max} T} \tag{13}$$

where W is a scale parameter (lower than 1), $\frac{\delta_P(k)}{\delta_P^{\max}}$ is the frequency weight for a variation of power equal to $\delta_P(k)$, and $\sum_{k=1}^{|T|} \frac{z(k)}{T}$ is the frequency of transitions between the active states. The parameter W is inserted since we expect a lower impact on failure rates when variations among active states are performed, comparing to the case in which transitions involve SM.

Thus, we can see that the server AF is governed by four types of parameters:

(i) χ^{off}, AF^{off}, and W, which depend only on the HW used to build the device;
(ii) $x(k)$ and $y(k)$, which are instead related to how long and how frequent SM is set;
(iii) $z(k)$, which is governed by the transitions between power states; and
(iv) T, which is the total period of time under consideration.

Although the HW parameters are fixed given the server, it is possible to play with the other variables, which are instead linked to the power state transitions, the power state duration, and the total period of time under consideration. Note that in this work we have considered the server as a whole by defining a model for the complete server. The application of our model to single server components is a future work. Clearly, there are components that are critical for the life of the entire server—*i.e.*, if one of these components fails, then the server will not able to serve traffic. These components include the memory, the CPU, and the hard disk. Therefore, in order to estimate the HW parameters of our model, one can concentrate on the single components and then extract:

(i) the failure rate in each power state;
(ii) the number of cycles to failure.

Then, by following the procedure reported in [5], it is possible to estimate the HW parameters for the single components. Clearly, the total server AF can be defined as the average of the AF of the single components or alternatively another function—e.g., the maximum AF.

Up to this point, we have focused on the single server. Without loss of generality, we can extend the AF to the whole set of servers of the DC. More formally, let us denote with D the set of servers of the DC, whose cardinality is $\mathcal{D} = |D|$. AF^w is the AF of w-th server, which is defined by Equation (7). Then, we can express the average DC AF as:

$$AF^{\text{avg}} = \sum_{w=1}^{\mathcal{D}} \frac{AF^w}{\mathcal{D}} \tag{14}$$

Figure 1 reports the main steps to obtain the AF^{avg} for a given DC. In particular, we start from traffic requests from users, and from different scenario constraints (like the number of servers and server capacity). Then, we run a scheduling algorithm to associate each request to the server. Given the request–server associations, we can estimate the server's power consumption for each time slot k. Then, we compute variables $x(k)$, $y(k)$, and $z(k)$ for each server. We then plug these variables, together with the power state $P(k)$ and the HW parameters χ^{off}, AF^{off}, and W in the AF model of Equation (14) in order to compute the average DC AF^{avg}.

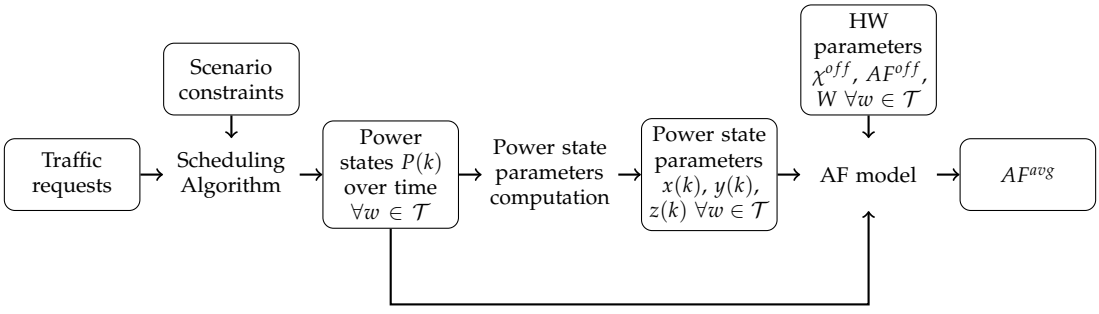

Figure 1. Main steps to compute the average AF in the data center (DC). HW: hardware.

Table 1 reports the main notation introduced so far.

Table 1. Main notations. SM: sleep mode.

Symbol	Measurement Unit	Description
T	h	Total time under consideration
Δ_k	h	Duration of time slot k
k	-	k-th time slot
$P(k)$	Watt	Server power consumption at time slot k
$\delta_P(k)$	Watt	Absolute difference between power consumption $P(k)$ in the k-th time slot and power consumption $P(k-1)$ in the $(k-1)$-th time slot
$P^{\max}$	Watt	Server maximum power consumption in active mode
$P^{\min}$	Watt	Server minimum power consumption in active mode
$\delta_{P^{\max}}$	Watt	Absolute difference between $P^{\max}$ and $P^{\min}$
$x(k)$	-	Binary variable: 1 if $P(k) = 0$, 0 otherwise
$y(k)$	-	Binary variable: 1 if $(P(k) = 0$ and $\delta_P(k) > 0)$ or $(P(k-1) = 0$ and $\delta_P(k) > 0)$, 0 otherwise
$z(k)$	-	Binary variable: 1 if $P(k) > 0$ and $P(k-1) > 0$ and $\delta_P(k) > 0$, 0 otherwise
AF	-	Acceleration factor
AF_{s}	-	Acceleration factor due to power states
AF_{t}	-	Acceleration factor due to power states transitions
$AF_{\mathrm{s}}^{\mathrm{off}}$	-	Acceleration factor due to SM state observed over T
AF^{off}	-	Acceleration factor in SM
$AF_{\mathrm{s}}^{\mathrm{active}}$	-	Acceleration factor due to active states observed over T
$AF_{\mathrm{t}}^{\mathrm{off}}$	-	Acceleration factor due to SM-active state transitions observed over T
$AF_{\mathrm{t}}^{\mathrm{active}}$	-	Acceleration factor due to active states transitions observed over T
χ^{off}	1/h	Frequency weight parameter for a transition between SM and active power
W	-	Scale factor for χ^{off}

5. Scenario Description

In this section, we describe the scenario assumed to evaluate our model. Specifically, we first detail the adopted scheduling algorithm, and then we present the data set under consideration.

5.1. Algorithm Description

In this work, we want to evaluate the impact that an energy saving strategy has on the lifetime of the physical machines of a DC (the evaluation of the impact on the lifetime on a set of VMs is left as future work). To do this, we focus our attention on GreFar [11], a scheduling algorithm for geo-distributed DCs. In particular, GreFar is able to reduce the energy cost while keeping the jobs delay under control over a long time interval.

The working scenario consists of M DCs located in different places, far enough between each other to experience different electricity costs. It is assumed that there is a central scheduler that collects all job requests at a given time t. Job requests can be grouped in J different types, depending on the amount of required physical resources (e.g., CPU speed, memory, *etc.*). Jobs can be immediately processed or they can be put in a queue waiting for availability of processing resources or a decrease of the electricity cost.

Each DC hosts a set of physical machines (or servers) that can be different in terms of HW deployed and power consumption requirements. Occasionally, at a given time t, a subset of servers could be unavailable to process new jobs for different reasons (*i.e.*, processing of priority jobs, server failures, software upgrades, *etc.*). Considering a long budgeting period T, the goal of GreFar is to minimize the electricity cost while limiting the average delay experienced by the jobs in the queue.

GreFar works in an online fashion—*i.e.*, it makes its scheduling decisions based on the current input parameters, without making any assumption about the future, such as job arrivals, electricity cost, and server availability. When a new job request arrives, GreFar can decide to process it and, in this case, it can also decide which DC will serve the request (*i.e.*, typically the one that is experiencing the lowest electricity cost), or it can put the request in the queue, waiting for a more convenient time to start the processing. In order to limit the queue length, at each time slot (where the definition of time

slot is the largest time interval in which it is possible to accurately know the electricity cost and the number of arrivals) GreFar tries to minimize the following objective function:

$$Vg(t) + q(t) \tag{15}$$

where $g(t)$ is the total cost experienced to power on all the working physical machines in all the servers at time t, and $q(t)$ is the queue length—*i.e.*, the number of jobs waiting to be processed at time t. V is called the cost–delay parameter and plays a crucial role in GreFar:

(i) if it is high enough, then the scheduler will prefer to reduce the cost, leaving the queue length growing and waiting for time intervals in which the electricity cost is convenient before starting to process jobs; or
(ii) if it is low, then GreFar will prefer to reduce the delay experienced by the jobs in the queue instead of optimizing the costs.

The peculiarity of GreFar is that this algorithm is $O\left(\frac{1}{V}\right)$-optimal with respect to average cost against the optimal T-step lookahead policy [11] (*i.e.*, solving the optimization problem by assuming a knowledge of the future in the next T time slots), while the queue length is bounded by $O(V)$.

5.2. Data Set Description

In the simulation we consider a time period of one month, divided in time slots of 1 h, for a total of 744 time slots. At the beginning of each time slot, the number of job requests, the electricity cost, and the number of unavailable servers are assumed to be known. More in detail, the electricity cost for each DC has been generated according to the real traces of [19]. Jobs are classified in nine different types, each of which is characterized by its own CPU request and arrival rate. For job requests, we use the same traces of [11].

Like in [11], we consider four geo-distributed heterogeneous DCs, each hosting 200 physical machines. Here we suppose that there are four server types. Each server type is characterized by a maximum power consumption, an idle power consumption, and a CPU speed. We refer the reader to [11] for more details about the servers' power consumption models.

Over this data set we have run GreFar, and the obtained results have been used as input for our lifetime model.

6. Performance Evaluation

We first consider the number of power state changes and their duration. Unless otherwise specified, we set a value of V equal to one for the algorithm. This value is a trade-off between delay minimization ($V = 0$) and energy reduction ($V > 0$). Moreover, we consider a time period T equal to 750 h. We first focus on the transition between active power and SM, as it is the most critical one in terms of AF increase (and consequently lifetime reduction). Figure 2 reports the number of power state changes involving SM and the total number of servers in SM for a single DC (the same analysis repeated on the other DCs produced similar results). Clearly, the total number of servers in SM varies over time without any daily periodicity, since the requests arrive in a random order, due to the fact that the users are spread around the world. Therefore, there is not a clear zone where traffic is low (e.g., at night). Hence, the number of servers in SM frequently changes with time, and consequently also the number of power state transitions. For example, it is possible to put most of the servers in SM at time 398 h, while shortly after they are all active. This behavior introduces a lot of transitions, which we expect to impact the lifetime.

To give more insight, Figure 3 reports the number of state changes *versus* server ID, differentiating between: (i) all transitions and (ii) transitions involving a SM state. Interestingly, although the server can exploit power proportional states, most power states are either in SM or at full power, resulting in the majority of state changes involving a transition between these two states. Moreover, we can see

that more than 50% of servers are experiencing quite a lot of transitions—*i.e.*, more than 200 over 750 h, which corresponds to more than 6.5 transitions per day. Additionally, Figure 4 reports the time in SM (normalized to T) *versus* server ID. We can see that more than 50% of servers are in SM for less than 15% of the time, in order to reduce the delay for requests.

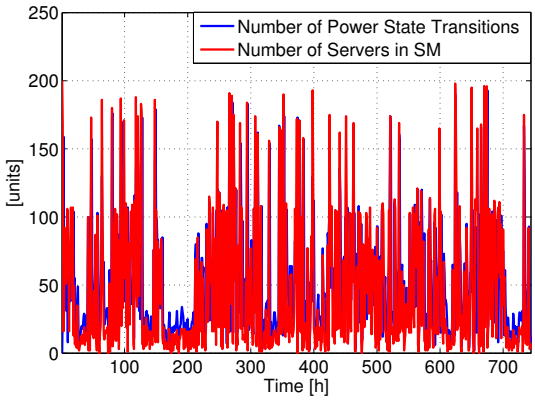

Figure 2. Number of power state changes *vs.* time.

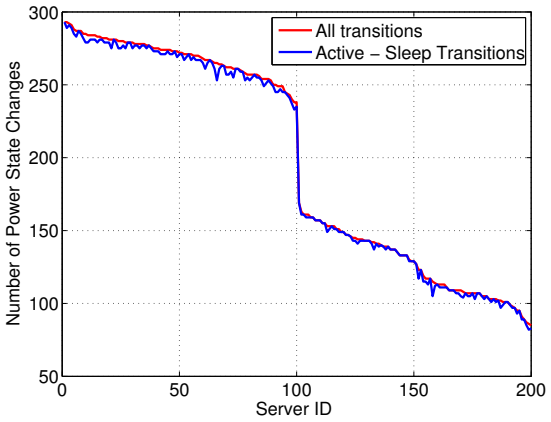

Figure 3. Number of power state changes *vs.* server ID.

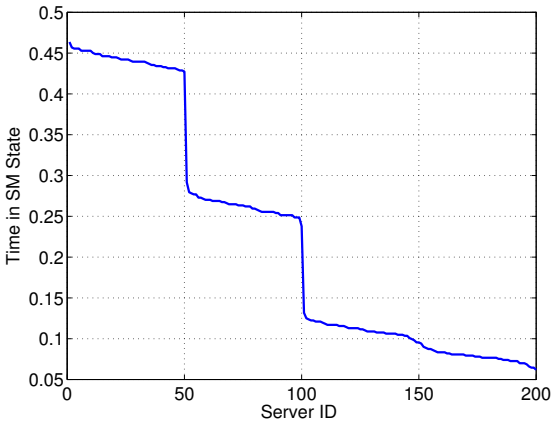

Figure 4. Time in SM *vs.* server ID.

We then apply our lifetime model to the DC taken under consideration. In particular, we incrementally compute AF, AF_s^{off}, AF_s^{active}, AF_t^{off}, AF_t^{active} for each time period $l \in T$ (*i.e.*, from $k = 0$ to $k = l$) for each server. Unless otherwise specified, we assume that the servers have the same HW components. Therefore, χ^{off}, AF^{off}, and W are the same for all the servers. In particular, we initially set $AF^{off} = 0.5$, $\chi^{off} = 2$ (cycles/h), and $W = 1$. The reasons for these settings are as follows:

(i) we consider a gain in the lifetime when the device is put in SM (or the active power is reduced);
(ii) we tend to penalize power state transitions involving SM;
(iii) we assume that the penalties of active power state transitions are much lower compared to the active-SM penalty.

Figure 5 reports AF, AF_s^{off}, AF_s^{active}, AF_t^{off}, AF_t^{active} *versus* time. The colored areas represent confidence intervals (assuming a 95% confidence level), computed over the whole set of servers belonging to the DC. Interestingly, we can clearly see that the total AF increases with time, passing from values lower than 1 to more than 1.4, meaning that at the end of the considered time period, the lifetime has been reduced by more than 40% compared to the case in which the servers always work at full power. Looking more in detail at the AF components, we can see that both AF_s^{off} and AF_s^{active} are lower than 1, meaning that their contribution tends to increase the lifetime. In particular, AF_s^{active} is lower than one since the server is not always at full power. Additionally, AF_s^{off} is much lower than AF_s^{active}, due to the fact that SM is not set for all time periods, and AF^{off} is lower than the AF at full power (which is equal to 1). On the contrary, the negative effect on the total AF is brought by the transitions. Although AF_t^{active} is almost close to 0 (due to the fact that there are few transitions between active power states), AF_t^{off} notably increases with time, due to the fact that active-SM transitions are accumulated in the DC. In the long term, AF_t^{off} becomes the predominant term in the AF, thus bringing to a lifetime decrease with respect to the reference case, in which the servers are always at full power.

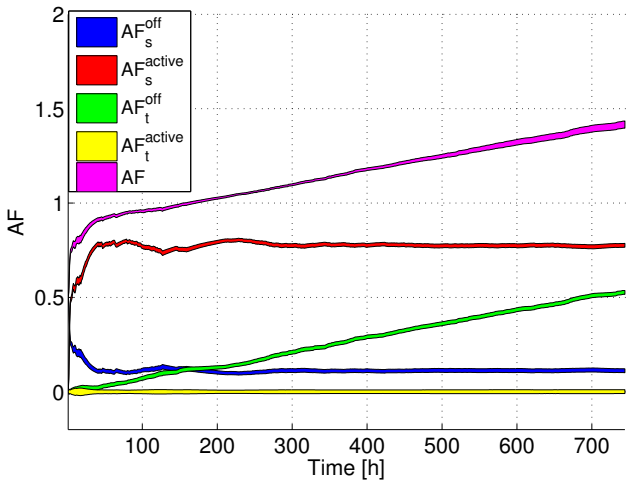

Figure 5. AF components *vs.* time.

In the following step, we consider the variation of the HW parameters AF^{off} and χ^{off}. Figure 6 reports the AF *versus* time averaged over the servers in the DC. Interestingly, the lower is AF^{off}, the lower is also the total AF. This is due to the fact that different servers are put in SM, and consequently their AF tends to decrease when AF^{off} is decreased. However, we can see that χ^{off} plays a crucial role in determining the final AF. In particular, the higher χ^{off} is, the higher the term in the AF due to power state variations is also. Consequently, the final AF tends to be higher than one, resulting in a lifetime decrease. From this figure, we can clearly see that lifetime-aware servers (*i.e.*, devices built with low AF^{off} and low χ^{off}) tend to limit this decrease, despite the fact that different power state transitions take place.

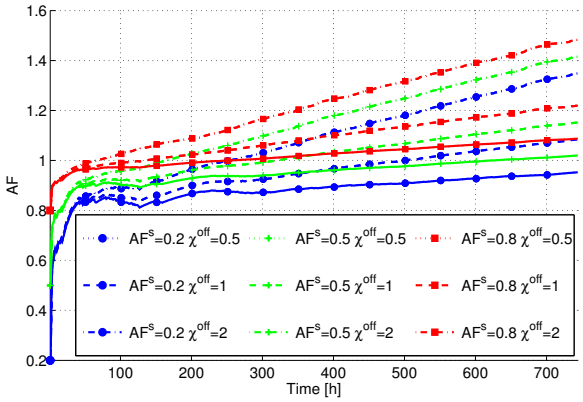

Figure 6. HW parameters variation *vs.* time.

To give more insight, we have computed the AF *versus* time by differentiating between the different DCs and the different server types, as reported in Figure 7. We recall that each server type is characterized by a maximum power consumption, an idle power consumption, and a CPU speed. In particular, we can see that the AF exhibits minor variations across the DCs and the server types, suggesting that the pattern for power state transitions and power state setting is similar between them.

In the following, we consider the impact of setting a delay minimization objective or an energy reduction strategy in the algorithm by changing the values of the V parameter.

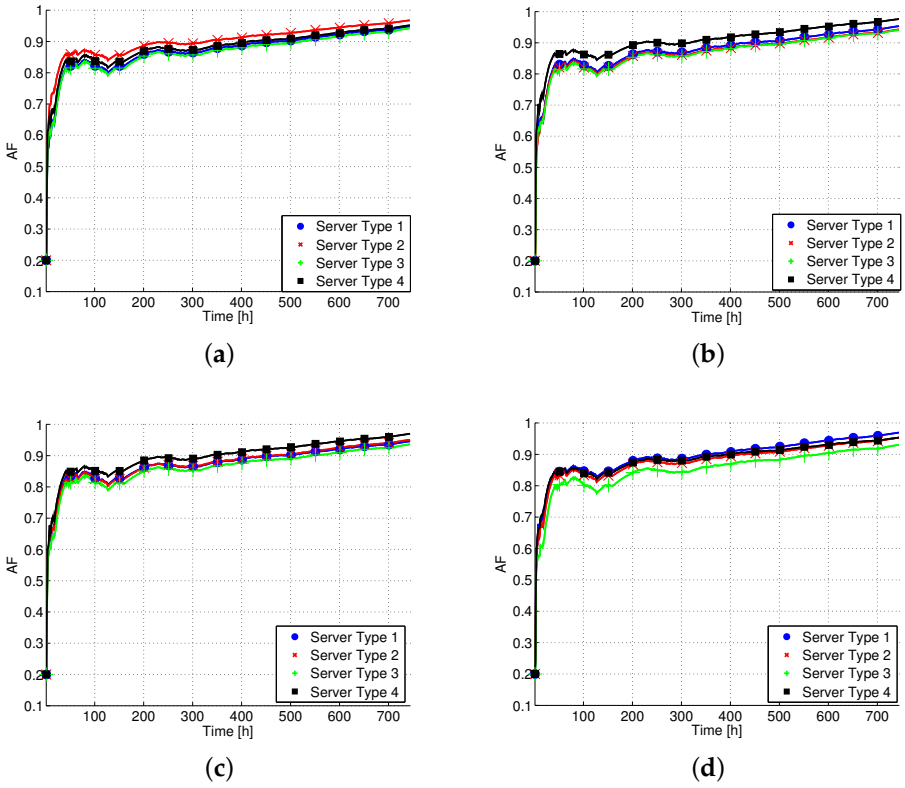

Figure 7. AF *vs.* time considering the different DCs and the different server types: (**a**) DC 1; (**b**) DC 2; (**c**) DC 3; and (**d**) DC 4.

6.1. Delay Minimization Impact

We first consider the case in which $V = 0$—*i.e.*, the delay tends to be minimized. Figure 8 reports the number of power state changes involving SM and the total number of servers in SM *versus* time. In contrast to the previous case (reported in Figure 2), the number of SM power states and the transitions are pretty low, most of the servers being always active during the considered time period. More in depth, if we consider the number of power state changes *versus* server ID (reported in Figure 9), we can see that more than 50% of servers experience less than or equal to 70 transitions during T, corresponding to less than 2.2 transitions per day. Moreover, we can clearly see that no transitions between active states are experienced, suggesting that a transition always implies a passage between full power and SM (or *vice-versa*).

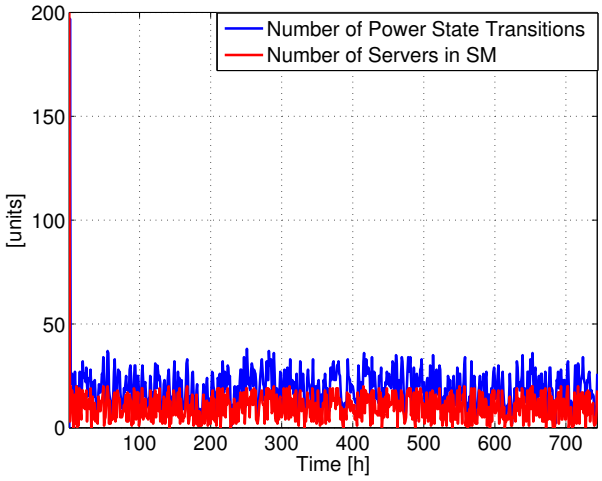

Figure 8. Number of power state changes *vs.* time ($V = 0$).

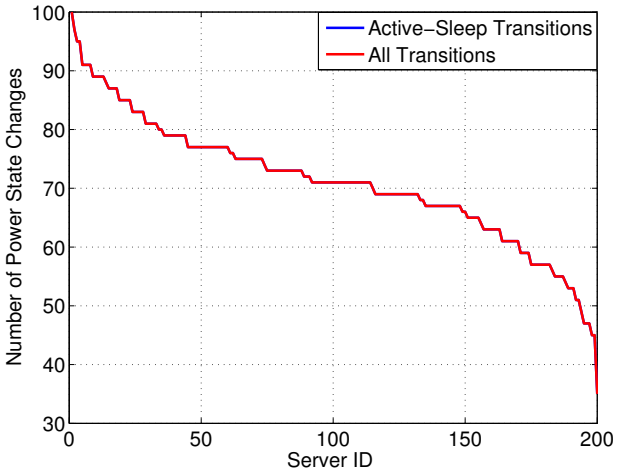

Figure 9. Number of power state changes *vs.* server ID ($V = 0$).

We then apply our model also in this case. Figure 10 reports the variation of the HW parameters AF^{off} and χ^{off}. Interestingly, since most of the servers are always powered on, the resulting AF is close to one. However, also in this case we can see that as χ^{off} increases, the AF tends to be higher than one at the end of the considered time period.

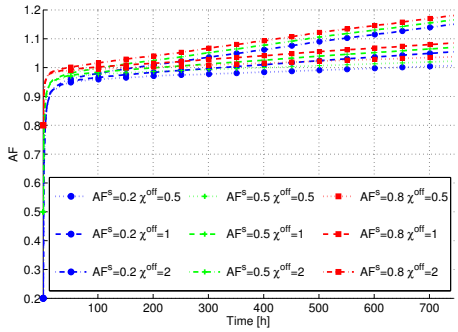

Figure 10. HW parameters variation *vs.* time ($V = 0$).

6.2. Energy Minimization Impact

In the last part of our work, we consider the impact of setting an energy-minimization strategy in the DC. When $V = \infty$, the energy is minimized. However, the delay is unbounded and therefore all the servers are always put in SM in each time slot, independent of the amount of traffic requests. To overcome this issue, we set $V = 1000$ in order to reduce energy while also considering the delay.

Figure 11 reports the number of power state changes and the number of servers in SM *versus* time. Interestingly, most servers are always put in SM, and they are activated in order to serve the traffic requests. Consequently, there are different transitions introduced. By considering the power state changes *versus* the server ID (reported in Figure 12), we can see that around 50% of servers do not experience any transitions. However, for the subset of servers changing power state, we can see that even transitions between active power states are experienced.

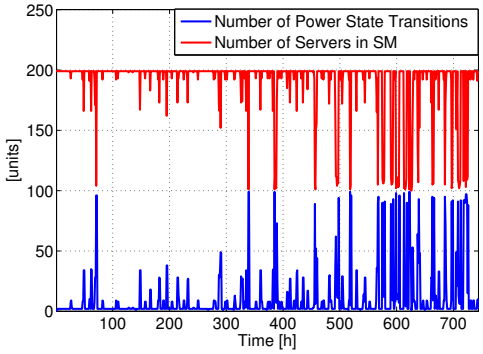

Figure 11. Number of power state changes *vs.* time ($V = 1000$).

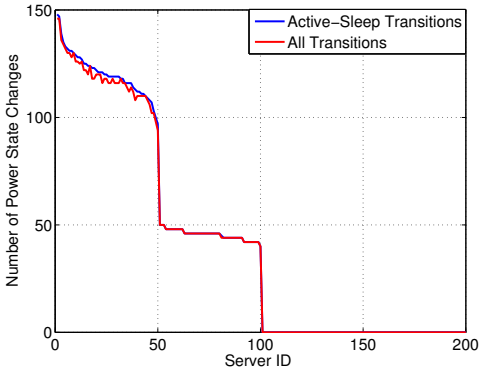

Figure 12. Number of power state changes *vs.* server ID ($V = 1000$).

Figure 13 reports the AF *versus* time in this scenario, considering a variation of the HW parameters AF^{off} and χ^{off}. Interestingly, in this case the AF is always lower than one, leading to a lifetime increase with respect to the reference case, in which the servers are kept at full power . In particular, although the AF tends to increase when AF^{off} is increased (as expected), this effect is counterbalanced by the fact that many servers are always in SM, resulting in a low AF. In fact, if we consider the different AF components (reported in Figure 14 for the case $AF^{\text{off}} = 0.5$, $\chi^{\text{off}} = 2$ (cycles/h), and $W = 1$), we can clearly see that the largest contribution on the final AF is brought by $AF_{\text{s}}^{\text{off}}$ in this case, resulting in a lifetime increase.

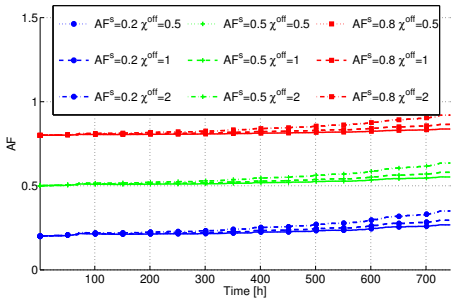

Figure 13. HW parameters variations *vs.* time ($V = 1000$).

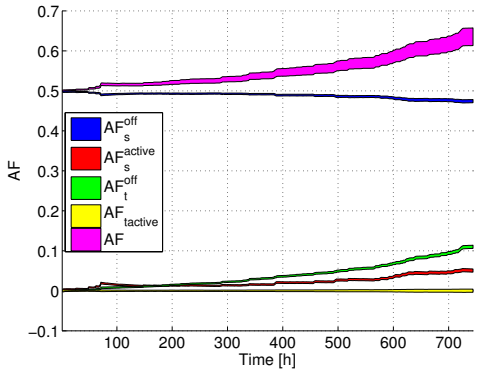

Figure 14. AF components *vs.* time ($V = 1000$).

6.3. Discussion

In the following, we will briefly discuss the main issues emerging from our work.

6.3.1. Impact of Scenario

In this work, we have evaluated a server lifetime model in cloud DCs. Clearly, the presented results depend on the specific scenario taken under consideration. In particular, we have considered the same number of servers deployed in each DC. However, if the number of deployed servers is not the same across the set of DCs, we may expect differences in the AF experienced by the servers in each DC. For example, when the number of deployed servers in a specific DC is above the average number of deployed servers per DC, we may expect that different servers may be put in SM, thus decreasing their AF. On the contrary, when the number of deployed servers is lower than the average, then almost all the servers will always be in active mode, thus increasing their AF. Moreover, the workload characteristics may impact the power usage, and consequently the results.

In addition, the considered time slot duration (*i.e.*, 1 h) is sufficient to capture temperature variations triggered by power state variations. In general, temperature is more inertial than power state change and it is accumulated from continuous work state, regardless of power increase or

decrease. In other words, temperature variation is less than power state change. For example, although processor frequency is decreasing, the processor temperature will increase after working for a while. In our case, this phenomenon may appear when very short time slot durations are taken into account (*i.e.*, in the order of seconds or a minute). As a consequence, the presented model may not be valid in these cases, since the temperature will not change with power. We leave the investigation of models tailored to shorter time slot changes as future work.

6.3.2. Impact of Hardware Components Failures

A server is composed of different HW components—such as processors, disks, memory, *etc.*—which may have different failure patterns. Our work aims at the definition of a lifetime model for the whole server (and not for each component). In particular, when a server fails, there is a QoS degradation for users (almost independent of the specific component that fails). As a consequence, we are interested in the lifetime expressed as the mean time of the two failure events, independent of the components that have triggered the failures. We leave the investigation of more specific models tailored to specific components as a future work.

6.3.3. Other Factors Influencing Failures

Except for the normal worn-out parts, server failures may be triggered by different factors, including: overheating, defects in the chip, operation outside the design specifications, power surge/fluctuation, over-voltage, and overheating. For example, focusing on hard disks, temperature, impact, exposure to water or high magnetic fields, or dust environment may also contribute to its failures. Our model instead takes into account only the failures triggered by power state changes. Nevertheless, all these other failures (which do not depend on the power state changes) can be taken into account in the reference failure rate—*i.e.*, the failure rate achieved by keeping a constant power, introduced in the definition of the AF metric. Our model then reports the variation of lifetime with respect to the reference failure rate.

6.3.4. Data Center Thermo-Mechanical Physical Design

The study of the materials' properties as they change with temperature is the scope of the thermo-mechanical analysis. A branch of this analysis is devoted to the DC physical design. In this context, this work falls within the scope of modulated temperature conditions with static force applied. Actually, we assume that no force is applied on the material, and the resulting AF is solely the result of power (and hence temperature) variations. In general, predicting the lifetime of electronic components is not a trivial task, and different works in the literature detail experimental data of electronic components' lifetime. For example, the authors of [20] focus on solder joint interconnections, which are considered to be the weaknesses of microelectronic packaging. In particular, the authors conducted an accelerated temperature cycling in a thermal chamber, where temperatures range between a minimum and a maximum value. Then, after the thermal cycling was completed for different numbers of cycles, two samples were extracted, prepared, and then analyzed with a scanning electron microscope to characterize the microstructures and to collect failure data. The authors clearly show that the failure probability (called unreliability) increases with the number of cycles. Focusing then on the AF models, the authors of [21] provide a comprehensive overview for joint solders. In our case, we are characterized by relatively short dwell times (*i.e.*, the time between reaching a given power state from another one) and pretty large variations of power (especially when passing from/to SM). Therefore, plastic strain-based fatigue models [21] (like the Coffin–Manson one) can be exploited. We recognize, however, that creep strain-based fatigue models [21] may be assumed when considering transitions between active power states (especially when occurring between the highest power states). Actually, one of possible future research activities will be to measure the AF for a set of server machines and then to follow the methodology proposed by [21] for selecting the applicable fatigue models (and their parameters).

Finally, it is worth mentioning that an optimized lifetime-aware management of DCs could also have implications on the provisioning of air conditioning [22], as well as reducing active redundant air conditioning systems.

7. Conclusions and Future Work

We have proposed a primary model for the evaluation of the lifetime of servers subject to management policies in cloud DCs. The model takes into account two different effects impacting the lifetime: (i) the reduction of power (which tends to increase the lifetime); and (ii) the variation among power states (which instead tends to decrease the lifetime). We have considered the possibility of changing the active power or exploiting a deep SM. We have evaluated our model considering a realistic scenario. Our results indicate that the lifetime is heavily influenced by the HW parameters used to build the servers, as well as the management policy taken into account to change the servers' power states. As a next step, we plan to perform measurements to better estimate the HW parameters, and to validate the proposed model by means of HW testing on real servers. Additionally, we plan to study management policies taking into account DC energy, user QoE, and server lifetime. Finally, we will consider the application of our model to single server components, and to short time slot durations.

Acknowledgments: The research leading to these results has received funding from the Sapienza Awards LIFETEL. The authors would like to thank Robert Birke and Lydia Y. Chen from IBM research for the fruitful discussions, and the reviewers for their valuable feedback.

Author Contributions: This work was led by Luca Chiaraviglio. Preparation of the manuscript has been performed by Luca Chiaraviglio. Revision of the manuscript has been performed by Antonio Cianfrani, Marco Listanti and Marco Polverini. Simulations have been performed by Marco Polverini. Analysis of the results has been perfomed by Luca Chiaraviglio and Marco Polverini.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco Global Cloud Index. Available online: http://www.cisco.com/c/en/us/solutions/service-provider/global-cloud-index-gci/index.html (accessed on 20 July 2015).
2. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; *et al*. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58.
3. Barroso, L.A.; Clidaras, J.; Hölzle, U. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synth. Lect. Comput. Archit.* **2013**, *8*, doi:10.2200/S00516ED2V01Y201306CAC024.
4. Beloglazov, A.; Buyya, R. Energy efficient resource management in virtualized cloud data centers. In Proceedings of the IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, 17–20 May 2010; pp. 826–831.
5. Chiaraviglio, L.; Wiatr, P.; Monti, P.; Chen, J.; Lorincz, J.; Idzikowski, F.; Listanti, M.; Wosinska, L. Is green networking beneficial in terms of device lifetime? *IEEE Commun. Mag.* **2015**, *53*, 232–240.
6. Chabarek, J.; Sommers, J.; Barford, P.; Estan, C.; Tsiang, D.; Wright, S. Power Awareness in network design and routing. In Proceedings of the IEEE INFOCOM, Phoenix, AZ, USA, 15–17 April 2008.
7. McKeown, N. Software-defined networking. In Proceedings of the Keynote Talk at IEEE INFOCOM, Rio de Janeiro, Brazil, 21 April 2009.
8. Jain, R.; Paul, S. Network virtualization and software defined networking for cloud computing: A survey. *IEEE Commun. Mag.* **2013**, *51*, 24–31.
9. El-Sayed, N.; Stefanovici, I.A.; Amvrosiadis, G.; Hwang, A.A.; Schroeder, B. Temperature management in data centers: Why some (might) like it hot. *ACM SIGMETRICS Perform. Eval. Rev.* **2012**, *40*, 163–174.
10. Qureshi, A.; Weber, R.; Balakrishnan, H.; Guttag, J.; Maggs, B. Cutting the electric bill for internet-scale systems. *ACM SIGCOMM Comput. Commun. Rev.* **2009**, *39*, 123–134.
11. Polverini, M.; Cianfrani, A.; Ren, S.; Vasilakos, A.V. Thermal-aware scheduling of batch jobs in geographically distributed data centers. *IEEE Trans. Cloud Comput.* **2014**, *2*, 71–84.
12. Islam, M.; Ren, S.; Quan, G.; Shakir, M.; Vasilakos, A. Water-constrained geographic load balancing in data centers. *IEEE Trans. Cloud Comput.* **2015**, doi:10.1109/TCC.2015.2453982.

13. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768.
14. Heller, B.; Seetharaman, S.; Mahadevan, P.; Yiakoumis, Y.; Sharma, P.; Banerjee, S.; McKeown, N. Elastictree: Saving energy in data center networks. In Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, USA, 28–30 April 2010; Volume 10, pp. 249–264.
15. Fang, W.; Liang, X.; Li, S.; Chiaraviglio, L.; Xiong, N. Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Comput. Netw.* **201**3, 57, 179–196.
16. Laidler, K.J. *Chemical Kinetics, 3/E*; Harper Collins: New York, NY, USA, 1987.
17. De Salvo, B.; Ghibaudo, G.; Pananakakis, G.; Reimbold, G.; Mondond, F.; Guillaumot, B.; Candelier, P. Experimental and theoretical investigation of nonvolatile memory data-retention. *IEEE Trans. Electron. Devices* **1999**, *46*, 1518–1524.
18. *Failure Mechanisms and Models for Semiconductor Devices*; JEDEC Publication JEP122-C; JEDEC Solid State Technology Association: Arlington, VA, USA, 2006.
19. Market Information System. Available online: http://nodal.ercot.com/docs/pd/mis/index.html (accessed on 20 July 2015).
20. Mi, J.; Li, Y.F.; Yang, Y.J.; Peng, W.; Huang, H.Z. Thermal cycling life prediction of Sn-3.0 Ag-0.5 Cu solder joint using type-I censored data. *Sci. World J.* **2014**, *2014*, doi:10.1155/2014/807693.
21. Lee, W.W.; Nguyen, L.T.; Selvaduray, G.S. Solder joint fatigue models: Review and applicability to chip scale packages. *Microelectron. Reliab.* **2000**, *40*, 231–244.
22. Patel, C.D.; Sharma, R.K.; Bash, C.E.; Beitelmal, M.H. Energy flow in the information technology stack: Introducing the coefficient of performance of the ensemble. In Proceedings of the ASME 2006 International Mechanical Engineering Congress and Exposition, Chicago, IL, USA, 5–10 November 2006.