electronics

MDPI

Article

An Energy Efficient Task Scheduling Strategy in a Cloud Computing System and its Performance Evaluation using a Two-Dimensional Continuous Time Markov Chain Model

Wenjuan Zhao [1], Xiushuang Wang [1], Shunfu Jin [1,*], Wuyi Yue [2] and Yutaka Takahashi [3]

[1] School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China
[2] Department of Intelligence and Informatics, Konan University, Kobe 658-8501, Japan
[3] Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
* Correspondence: jsf@ysu.edu.cn; Tel.: +86-335-8057078

Received: 23 April 2019 ; Accepted: 9 July 2019; Published: 11 July 2019

Abstract: With ongoing energy shortages and rises in greenhouse emissions worldwide, increasing academic attention is being turned towards ways to improve the efficiency and sustainability of cloud computing. In this paper, we present a performance analysis and a system optimization of a cloud computing system with an energy efficient task scheduling strategy directed towards satisfying the service level agreement of cloud users while at the same time improving the energy efficiency in cloud computing system. In this paper, we propose a novel energy-aware task scheduling strategy based on a sleep-delay timer and a waking-up threshold. To capture the stochastic behavior of tasks with the proposed strategy, we establish a synchronous vacation queueing system combining vacation-delay and N-policy. Taking into account the total number of tasks and the state of the physical machine (PM), we construct a two-dimensional continuous-time Markov chain (CTMC), and produce an infinitesimal generator. Moreover, by using the geometric-matrix solution method, we analyze the queueing model in the steady state, and then, we derive the system performance measures in terms of the average sojourn time and the energy conservation level. Furthermore, we conduct system experiments to investigate the proposed strategy and validate the system model according to performance measures. Statistical results show that there is a compromise between the different performance measures when setting strategy parameters. By combining different performance measures, we develop a cost function for the system optimization. Finally, by dynamically adjusting the crossover probability and the mutation probability, and initializing the individuals with chaotic equations, we present an improved genetic algorithm to jointly optimize the sleep parameter, the sleep-delay parameter and the waking-up threshold.

Keywords: cloud computing system; task scheduling; energy conservation; sleep-delay timer; waking-up threshold; Markov chain; cost function; intelligent searching algorithm; joint optimization

1. Introduction

Cloud computing is a paradigm of computing in which dynamically scalable and virtualized resources are provided as a service over the Internet [1–3].

In a cloud environment, there are two key actors: cloud providers and cloud users [4]. The cloud providers hold enormous computing resources in data centers [5]. They lease resources out to the cloud users on a pay-per-use basis. Therefore, the cloud providers want to improve resource utilization and maximize their profit, while the cloud users, who have applications of various loads, attempt to receive service from different cloud providers at the lowest expense possible [6].

How to achieve a higher resource utilization, while at the same time offering a lower cost to the cloud users, is a key part of a cloud provider's management strategy [7,8]. Some researchers have conducted analyses to minimize the construction period and increase the system utilization by scheduling several cloud tasks on different virtual machines (VMs) [9]. However, this scheduling method requires all servers to keep incoming tasks active, leading to dramatic levels of energy consumption and elevating carbon dioxide emissions [10,11]. Energy consumption is estimated to continue rising to approximately 73 billion kWh by 2020 [12].

Therefore, one of the current challenges in cloud computing is to reduce energy consumption while guaranteeing the quality of user experience.

1.1. Related Works

Cloud computing, with its unprecedented computing capability, has become a popular paradigm yet has raised concerns from enterprises [13,14]. Recently, many scholars have carried out fruitful research on cloud management and cloud optimization.

Yang et al. quantified the cloud service performance based on the assumption of a failure recovery [15]. By considering the recovery among processing nodes and communication links, and the precedence constraints of subtask, they derived a probability distribution of service response time. Fellera et al. evaluated a Hadoop performance in both the traditional model of collocated data and computing services model in cloud environments [16]. In addition, they conducted an energy efficiency evaluation of the Hadoop on physical and virtual clusters in different configurations.

Xia et al. [17] presented a queueing network-based performance framework with dynamic voltage scaling with the purpose of conserving power consumption. Cheng et al. [18] presented a method of Minimum Expectation Execution Energy with Performance Constraints (ME^3PC), by which the energy consumption can be effectively conserved under certain performance constraints. Chen et al. [19] proposed a dynamic voltage and frequency scaling scheme by which the most suitable voltage and frequency for the multi-core embedded system could be dynamically forecasted. Aiming to achieve adaptive regulations for different requirements in a cloud computing system, Shen et al. [20] studied a genetic algorithm E-PAGA.

All the aforementioned research has sought to conserve energy consumption, but has ignored the fact that, even though no tasks are being processed, all the virtual machines (VMs) remain awake.

Putting idle VMs in sleep mode is a way of conserving power consumption when the traffic load is light [21]. Kempa [22] investigated a vacation queueing system with a finite buffer in which the transmission is restarted if the number of packets in the buffer reaches a threshold at the epoch when a sleep period ends. For the purpose of reducing the carbon footprint of data centers, Mcbay et al. [23] presented a combined approach using an energy conservation method of dynamic voltage/frequency scaling and sleep mode. To efficiently control the traffic load on each VM, Lawanyashri et al. [24] introduced a vacation mechanism with threshold policy. With this mechanism both the energy consumption and the system cost could be cut down. In a greener cloud computing system, Singh et al. [25] presented a deep-sleep mode in a cloud computing system to conserve energy consumption and improve the resource utility. Putting idle VMs into a sleep state can conserve energy consumption to some extent. However, continually switching VMs between asleep and awake states can cause response penalties.

Based on this research background, we consider an effective strategy to satisfy the response performance of cloud users while also trying to improve the energy efficiency in the cloud computing system.

In making a compromise between the performance degradation and the energy conservation, we need to evaluate the strategy performance and optimize the strategy parameters.

We note that a genetic algorithm is a heuristic method used to search for near-optimal solution in a large solution space [26]. Genetic algorithm originated as an effective tool for function optimization in the 1960s. Since then, considerable research on improving the searching ability of genetic algorithm has

been carried out. To improve the convergence rate of the genetic algorithm, Qiu et al. [27] implemented the rank-based roulette wheel scheme in the selection mimics, where the better individuals have more chance in reproducing offspring. For the purpose of introducing population diversity, Huang et al. [28] randomly and uniformly selected the initial parent chromosome from among the top-tier chromosomes, whereupon the second parent selected from the lower-tier chromosomes. Jiang et al. [29] presented an adaptive genetic algorithm, changing the crossover probability and the mutation probability according to the level of fitness. This method can prevent the genetic algorithm from getting stuck at a local optimal solution.

As mentioned above, the searching ability of a genetic algorithm is greatly influenced by the crossover probability and the mutation probability. This can conceptually be applied to optimizing the system parameters and to enhancing the overall performance for the greening of cloud computing.

1.2. Contributions

This paper is a substantial and appropriate extension of our previous work [30] appearing in the conference proceedings. In the paper of the proceedings [30], we proposed a task scheduling strategy with a sleep-delay timer and a waking-up threshold to satisfy the response performance of cloud users while reducing the energy consumption in cloud computing. In the numerical results, the performance of the system and the effects of the design parameters based on the average sojourn time of tasks and the energy conservation level were evaluated.

However, in this paper, with the advent of energy shortages and a rise in greenhouse gas emissions, we extend our previous work [30] to propose a more effective strategy for the greening of cloud computing. This is to further satisfy the response performance of cloud users on the premise of ensuring a higher energy efficiency in cloud computing by proposing a novel task scheduling strategy with a sleep-delay timer and a waking-up threshold. Moreover, to optimize the parameter settings for the cloud providers, we analyze the system performance, derive the system performance measures, present a cost function and propose an enhanced intelligent searching algorithm using a compromise between different performance measures.

The following summarizes the main contributions of this paper.

(1) We provide additional analyses for our proposed task scheduling strategy, and present the two forms of the state transition based on the relationship between the number of VMs in the system and the waking-up threshold.
(2) We evaluate the system performance of the energy efficient task scheduling strategy with a two-dimensional continuous-time Markov chain (CTMC).
(3) We provide new system experiments with analysis and simulation to investigate the proposed strategy. We also make a comparison between our proposed strategy and conventional strategies.
(4) By considering both the average sojourn time of tasks and the energy conservation level of the system, we establish a cost function to make a compromise between the performance degradation and the energy efficiency.
(5) By dynamically adjusting the crossover probability and the mutation probability, and initializing the individuals with chaotic equations, we present an improved genetic algorithm to jointly optimize the system parameters with the proposed strategy.

The structure of this paper is outlined as follows: In Section 2, we describe the task scheduling strategy and establish the system model accordingly. In Section 3, we derive the system model in the steady state. In Section 4, we give the performance measures and an improved intelligent searching algorithm to minimize the cost of the system. In Section 5, we provide system experiments to investigate the system performance of the proposed strategy. Finally, in Section 6, we draw conclusions.

2. Descriptions of a Task Scheduling Strategy and Establishment of the System Model

In this section, we propose a task scheduling strategy and construct the system model accordingly.

2.1. Description of Task Scheduling Strategy

Cloud services are provided over the cloud computing environment via distributed software and hardware. On a physical machine (PM), several virtual machines (VMs) can be deployed [31]. Even if no tasks need processing, sets of VMs located on one or more PMs will remain awake. As a result, massive amounts of energy are wasted, thus energy consumption has become a special concern for cloud providers.

Generally speaking, the PMs in a cloud data centers are highly configured. To ensure high availability and improve parallel processing capability, each VM hosted on a PM works within its own operating system. This arrangement offers the theoretical foundation for the feasible and practical implementation of a sleep mode on each VM.

Putting idle VMs to sleep is considered to be a useful method of reducing energy consumption in cloud computing [32]. However, the sleep mode may degrade the response performance. Therefore, we propose an energy efficient task scheduling strategy for cloud computing by introducing a sleep-delay timer and a waking-up threshold. Considering the negative effect from the sleep mode, we set a sleep-delay timer on a PM to guarantee the quality of experience for cloud users. When the system becomes empty, i.e., all the VMs are idle, the PM will not go to sleep immediately, but remain awake for a random time length under the control of a sleep-delay timer. Tasks arriving at the system during the sleep-delay period will receive immediate service. If and only if no tasks arrive at the system before the sleep-delay timer expires then the PM will go into periodical sleep, where multiple sleep periods constitute a sleep state.

Frequent state switches will certainly cause additional energy consumption and lead to extra latency. Thus, we set a critical waking-up threshold N to improve the energy efficiency. When a sleep timer expires, if the number of tasks waiting in the system buffer is less than the waking-up threshold N, the PM will keep asleep to save more energy consumption. If not, a new sleep period is initiated, i.e., the PM remains asleep. As a result, the energy consumption of each PM can be efficiently conserved.

For the task scheduling strategy proposed in this paper, a PM is in the awake state, the sleep state, or the sleep-delay state.

Awake State: During the awake state, there is at least one VM busy with task processing. The tasks in the system receive service in accordance with a *first-come first-served* (FCFS) policy.
Sleep-delay State: To extend the awake period and improve the response performance, once all the tasks in the system are completely executed, a sleep-delay timer with a random time length will be started, and all the VMs will remain active within the constraint of the sleep-delay timer. We call this state the sleep-delay state. A new task arriving at the system during the sleep-delay period will receive service promptly. At the epoch when the sleep-delay timer expires, if there are no tasks queueing in the system buffer, the PM will switch into sleep state.
Sleep State: A sleep timer with a random time length will also be started as soon as the PM enters the sleep state. In the sleep state, the power of some accessories, with the exception of the memory, will be cut off, and the tasks in the system will not be served. All the tasks arriving at the system during the sleep state have to wait in the system buffer. When the sleep timer expires, if there are fewer tasks waiting in the system buffer than the threshold N, another sleep period will be started. The time duration of this sleep period is controlled by a new sleep timer. Otherwise, the PM will switch to the awake state, and all the VMs in the PM will wake up and prepare to serve all the tasks queueing in the system buffer.

As a summary, the state transition of the PM with the proposed strategy is plotted in Figure 1.

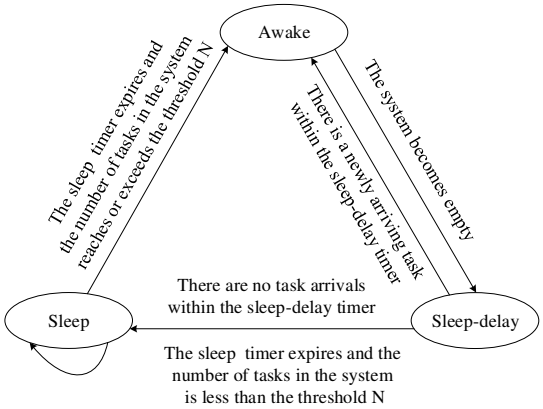

Figure 1. The transition among the three states in the proposed strategy.

2.2. Establishment of System Model

Just as in our previous research [30], we establish a synchronous multiple vacation queueing system with a vacation-delay and an N-policy to model the proposed strategy.

In a cloud data center, there are several PMs. We assume a task is assigned to one of the PMs with equal probability, and assume the behavior of all the PMs in a cloud data center to be stochastically homogeneous [33]. We consider a tagged PM to evaluate the task scheduling strategy. The tasks are assumed to arrive at the system according to a Poisson process with the parameter λ. The execution time of a task by a VM is assumed to have an exponential distribution with the parameter μ. The time length for the sleep-delay timer is supposed to have an exponential distribution with the parameter β. The time length for a sleep timer is supposed to follow an exponential distribution with the parameter θ, i.e., the average time length for a sleep-delay timer is equal to $\frac{1}{\theta}$. The system buffer is supposed to have an infinite capacity.

By $X_t = i$ $(i = 0, 1, \ldots)$, we denote the number of tasks in the system at the epoch t. By $Y_t = j$ $(j = 0, 1, 2)$, we denote the PM state at the epoch t. $j = 0$ means the PM is asleep, $j = 1$ means the PM is awake, and $j = 2$ means the PM is in a sleep-delay state. X_t and Y_t are called the system level and the PM state, respectively. The stochastic behavior of the queueing system under consideration can be given in a two-dimensional continuous-time Markov chain (CTMC): $\{(X_t,\ Y_t),\ t \geq 0\}$. The CTMC is regular irreducible and has an infinite state space as follows:

$$\mathbf{\Omega} = \{(0,0) \cup (0,2) \cup (i,j) : i \geq 1,\ j = 0, 1\}. \tag{1}$$

Under the assumption that the CTMC is positive recurrent, the steady-state distribution $\pi_{i,j}$ is defined as follows:

$$\pi_{i,j} = \lim_{t \to \infty} P\{X_t = i,\ Y_t = j\},\ i = 0, 1, \ldots,\ j = 0, 1, 2. \tag{2}$$

Let us form the row vector $\boldsymbol{\pi}_i$ as

$$\boldsymbol{\pi}_i = (\pi_{i,0}, \pi_{i,1}, \pi_{i,2}),\ i = 0, 1, \ldots. \tag{3}$$

The steady-state probability distribution $\mathbf{\Pi}$ of the CTMC can be partitioned as follows:

$$\mathbf{\Pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots). \tag{4}$$

3. Model Analysis in the Steady State

In this section, we present the state transition of the CTMC and derive the system model in the steady state.

3.1. State Transition

To analyze the CTMC in the steady state, one of the key steps is to construct an infinitesimal generator $\boldsymbol{Q}$. Based on the relationship between the number of VMs k in the cloud computing system under consideration and the waking-up threshold N with our proposed strategy, we give two forms of infinitesimal generators.

Let $\boldsymbol{Q}_{x,y}$ be the sub-matrices of $\boldsymbol{Q}$ for the system level jumping from x $(x = 0, 1, \ldots)$ to y $(y = 0, 1, \ldots)$. For the sake of convenience in presentation, we denote $\boldsymbol{Q}_{x,x-1}$ as $\boldsymbol{B}_x$, $\boldsymbol{Q}_{x,x}$ as $\boldsymbol{A}_x$ and $\boldsymbol{Q}_{x,x+1}$ as $\boldsymbol{C}_x$. Considering the size relationship between the number k of VMs in the cloud computing system and the waking-up threshold N, we discuss in detail $\boldsymbol{B}_x$, $\boldsymbol{A}_x$ and $\boldsymbol{C}_x$ for the following two cases.

Case I: $N \leq k$

(1) When the initial level x is 0, the sub-matrices $\boldsymbol{A}_0$ and $\boldsymbol{C}_0$ are given as follows:

$$\boldsymbol{A}_0 = \begin{pmatrix} -\lambda & 0 & 0 \\ 0 & 0 & 0 \\ \beta & 0 & -(\lambda+\beta) \end{pmatrix}, \quad \boldsymbol{C}_0 = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \lambda & 0 \end{pmatrix}.$$

(2) When the initial level x is 1, the sub-matrices $\boldsymbol{B}_1$, $\boldsymbol{A}_1$ and $\boldsymbol{C}_1$ are given as follows:

$$\boldsymbol{B}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \mu \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}_1 = \begin{pmatrix} -\lambda & 0 & 0 \\ 0 & -(\lambda+\mu) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{C}_1 = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

(3) When the initial level x ranges from 2 to $N-1$, the sub-matrices $\boldsymbol{B}_x$, $\boldsymbol{A}_x$ and $\boldsymbol{C}_x$ are given as follows:

$$\boldsymbol{B}_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & x\mu & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}_x = \begin{pmatrix} -\lambda & 0 & 0 \\ 0 & -(\lambda+x\mu) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{C}_x = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

(4) When the initial level x ranges from N to $+\infty$, the sub-matrices $\boldsymbol{B}_x$, $\boldsymbol{A}_x$ and $\boldsymbol{C}_x$ are given as follows:

$$\boldsymbol{B}_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \min(x,k)\mu & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}_x = \begin{pmatrix} -(\lambda+\theta) & \theta & 0 \\ 0 & -(\lambda+\min(x,k)^{-}) & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\boldsymbol{C}_x = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

From the infinitesimal generator $\boldsymbol{Q}$ in the case of $N \leq k$, we find that all the sub-matrices $\boldsymbol{B}_x$ and $\boldsymbol{A}_x$ are repeated for $x \geq k$, and all the sub-matrices $\boldsymbol{C}_x$ are repeated for $x \geq 1$. For the convenience of presentation, we represent the repetitive sub-matrices $\boldsymbol{B}_x$ $(x \geq k)$, $\boldsymbol{A}_x$ $(x \geq k)$, and $\boldsymbol{C}_x$ $(x \geq 1)$ as $\boldsymbol{B}$, $\boldsymbol{A}$ and $\boldsymbol{C}$, respectively. Thus, the infinitesimal generator $\boldsymbol{Q}$ for **Case I** can be given as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{C}_0 & & & & & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C} & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & \boldsymbol{B}_{N-1} & \boldsymbol{A}_{N-1} & \boldsymbol{C} & & & & \\ & & & \boldsymbol{B}_N & \boldsymbol{A}_N & \boldsymbol{C} & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \boldsymbol{B}_{k-1} & \boldsymbol{A}_{k-1} & \boldsymbol{C} & \\ & & & & & & \boldsymbol{B} & \boldsymbol{A} & \boldsymbol{C} \\ & & & & & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{5}$$

Case II: $N > k$

(1) When the initial level x is 0, the sub-matrices $\boldsymbol{A}_0$ and $\boldsymbol{C}_0$ are the same as those given in **Case I**.
(2) When the initial level x is 1, the sub-matrices $\boldsymbol{B}_1$, $\boldsymbol{A}_1$ and $\boldsymbol{C}_1$ are the same as those given in **Case I**, too.
(3) When the initial level x ranges from 2 to $N-1$, the sub-matrices $\boldsymbol{B}_x$, $\boldsymbol{A}_x$ and $\boldsymbol{C}_x$ can be given as follows:

$$\boldsymbol{B}_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \min(x,k)\mu & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}_x = \begin{pmatrix} -\lambda & 0 & 0 \\ 0 & -\lambda - \min(x,k)\mu & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\boldsymbol{C}_x = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

(4) When the initial level x ranges from N to $+\infty$, the sub-matrices $\boldsymbol{B}_x$, $\boldsymbol{A}_x$ and $\boldsymbol{C}_x$ are given as follows:

$$\boldsymbol{B}_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & k\mu & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{A}_x = \begin{pmatrix} -(\lambda+\theta) & \theta & 0 \\ 0 & -(\lambda+k\mu) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{C}_x = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

From the infinitesimal generator $\boldsymbol{Q}$ in the case of $N > k$, we find that all the sub-matrices $\boldsymbol{B}_x$ and $\boldsymbol{A}_x$ are repeated for $x \geq N$, and all the sub-matrices $\boldsymbol{C}_x$ are repeated for $x \geq 1$. We denote the repetitive sub-matrices $\boldsymbol{B}_x$ $(x \geq N)$, $\boldsymbol{A}_x$ $(x \geq N)$ and $\boldsymbol{C}_x$ $(x \geq 1)$ as $\boldsymbol{B}$, $\boldsymbol{A}$ and $\boldsymbol{C}$, respectively. The infinitesimal generator $\boldsymbol{Q}$ for **Case II** is given as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{C}_0 & & & & & & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C} & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & \boldsymbol{B}_{k-1} & \boldsymbol{A}_{k-1} & \boldsymbol{C} & & & & & \\ & & & \boldsymbol{B}_k & \boldsymbol{A}_k & \boldsymbol{C} & & & & \\ & & & & \ddots & \ddots & \ddots & & & \\ & & & & & \boldsymbol{B}_{N-1} & \boldsymbol{A}_{N-1} & \boldsymbol{C} & & \\ & & & & & & \boldsymbol{B} & \boldsymbol{A} & \boldsymbol{C} & \\ & & & & & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{6}$$

Now, all the sub-matrices in the infinitesimal generator $\boldsymbol{Q}$ have been addressed. From the block-tridiagonal structure of the infinitesimal generator $\boldsymbol{Q}$ in Equations (5) and (6), we note that the two-dimensional CTMC $\{(X_t,\ Y_t),\ t \geq 0\}$ is in fact a type of the quasi birth-and-death (QBD) process.

3.2. Steady-State Probability Distribution

To analyze the two-dimensional stochastic process $\{(X_t,\ Y_t),\ t \geq 0\}$, we need to obtain the minimal non-negative solution to the following matrix quadratic equation.

$$\boldsymbol{R}^2\boldsymbol{A} + \boldsymbol{R}\boldsymbol{B} + \boldsymbol{C} = \boldsymbol{0}. \tag{7}$$

We call the solution $\boldsymbol{R}$ of Equation (7) the rate matrix.

Since the coefficient matrices of Equation (7) are all upper triangular, and all the elements of the third row and the third column are zero, the rate matrix $\boldsymbol{R}$ can be written as follows:

$$\boldsymbol{R} = \begin{pmatrix} r_{11} & r_{12} & 0 \\ 0 & r_{22} & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{8}$$

where r_{ij} is the element that needs to be solved.

Substituting Equation (8) into Equation (7) yields the following set of equations:

$$\begin{cases} k\mu(r_{11}r_{12} + r_{12}r_{22}) + \theta r_{11} - (\lambda + k\mu)r_{12} = 0 \\ k\mu {r_{22}}^2 - (\lambda + k\mu)r_{22} + \lambda = 0 \\ -(\lambda + \theta)r_{11} + \lambda = 0. \end{cases} \tag{9}$$

By solving Equation (9) under the condition that $\rho = \dfrac{\lambda}{k\mu} < 1$ [34], we can derive the minimal non-negative solution $\boldsymbol{R}$ of Equation(7) as follows:

$$\boldsymbol{R} = \begin{pmatrix} \dfrac{\lambda}{\lambda + \theta} & \rho & 0 \\ 0 & \rho & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{10}$$

Based on the analytical result of $\boldsymbol{R}$, we can easily obtain the spectral radius $SP(\boldsymbol{R}) = \max\left(\dfrac{\lambda}{\lambda + \theta}, \rho\right) < 1$. Thus, the QBD is positive recurrent, and the steady-state probability distribution of the CTMC exists.

Let u be the minimum value of the number k of VMs and the waking-up threshold N, i.e., $u = \min\{k, N\}$. Let v be the maximum value of the number k of VMs and the waking-up threshold N, i.e., $v = \max\{k, N\}$. Then, we construct an augmented matrix as follows:

$$\boldsymbol{B}[\boldsymbol{R}] = \begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{C}_0 & & & & & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C} & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & \boldsymbol{B}_{u-1} & \boldsymbol{A}_{u-1} & \boldsymbol{C} & & & & \\ & & & \boldsymbol{B}_u & \boldsymbol{A}_u & \boldsymbol{C} & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \boldsymbol{B}_{v-2} & \boldsymbol{A}_{v-2} & \boldsymbol{C} & \\ & & & & & & \boldsymbol{B}_{v-1} & \boldsymbol{A}_{v-1} & \boldsymbol{C} \\ & & & & & & & \boldsymbol{B} & \boldsymbol{A} + \boldsymbol{R}\boldsymbol{B} \end{pmatrix}. \tag{11}$$

Using the equilibrium equation and the normalization condition, we have the following set of equations:

$$\begin{cases} (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_v)\boldsymbol{B}[\boldsymbol{R}] = \boldsymbol{0} \\ \sum\limits_{i=0}^{v-1} \boldsymbol{\pi}_i \boldsymbol{e} + \boldsymbol{\pi}_v (\boldsymbol{I} - \boldsymbol{R})^{-1} \boldsymbol{e}_1 = 1, \end{cases} \tag{12}$$

where $\boldsymbol{e}$ is a $3v \times 1$ column vector with ones, and $\boldsymbol{e}_1$ is a 3×1 column vector with ones.

Applying the Gauss–Seidel method [35], we obtain $\boldsymbol{\pi}_0$, $\boldsymbol{\pi}_1$, …, $\boldsymbol{\pi}_v$. Note that the CTMC $\{(X_t, Y_t),\ t \geq 0\}$ is a QBD process [36], and we can give the steady-state probability distribution of the CTMC as follows:

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_v \boldsymbol{R}^{i-v},\ i \geq v + 1. \tag{13}$$

By substituting $\boldsymbol{\pi}_v$ obtained in Equation (12) into Equation (13), we get $\boldsymbol{\pi}_i$ $(i = v + 1, v + 2, \ldots)$. Thus, the steady-state distribution $\boldsymbol{\Pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots)$ of the CTMC can be presented numerically.

4. Performance Measures and Genetic Algorithm

In this section, we derive the performance measures and propose an improved intelligent searching algorithm to minimize the cost of the system.

4.1. Energy Model

The sojourn time of a task is defined as the time duration from the epoch a task arrives at the cloud computing system to the epoch that task successfully departs from the cloud computing system. That is to say, by adding the average waiting time of a task in the system buffer and the service time of a task on the VM, we can obtain the sojourn time of a task. Based on the model analysis in Section 3.2, we give the average number L of tasks in the system under the steady state as follows:

$$L = \sum_{i=0}^{\infty} i(\pi_{i,0} + \pi_{i,1} + \pi_{i,2}). \tag{14}$$

Using Little's law [37], we give the average sojourn time W of tasks as follows:

$$W = \frac{1}{\lambda}\left(\sum_{i=0}^{\infty} i(\pi_{i,0} + \pi_{i,1} + \pi_{i,2})\right). \tag{15}$$

Within our proposed strategy, the energy conservation level of the system is defined as the decreased energy consumption per unit time. During both the awake state and the sleep-delay state, the PM consumes energy normally. During the sleep state, energy consumption is reduced. Moreover, at each sleep period completion instant, the listening process and the state transition will consume additional energy.

In deriving the energy conservation level of the system, by C_a we denote the energy consumption per unit time when the VMs are awake or idle in the sleep-delay state, by C_s we denote the energy consumption per unit time when the VMs are asleep, by C_t we denote the additional energy consumption for the VMs to wake up from the sleep state, and by C_l we denote the additional energy consumption when the VMs listen to the cloud computing system. Based on these values, we give the energy conservation level E_v of the system as follows:

$$E_v = \sum_{i=0}^{\infty} \pi_{i,0} \times (C_a - C_s) - \sum_{i=1}^{\infty} \pi_{i,0} \times \theta \times C_t - \sum_{i=0}^{\infty} \pi_{i,0} \times \theta \times C_l. \tag{16}$$

4.2. Genetic Algorithm

The economic analysis of cloud computing systems has recently been focusing increased attention on cloud providers [38,39]. In this section, with an aim to providing an enhanced energy efficient strategy in a cloud environment and maintaining the service level agreement (SLA) between the cloud providers and the cloud users, we establish a cost function for the system model.

Let f_s be the impact factor of the average sojourn time W of tasks on the system cost. Let f_v be the impact factor of the energy conservation level E_v of the system on the system cost. The cost function F is then given as follows:

$$F = f_s W - f_v E_v, \tag{17}$$

where W and E_v are given in Equations (15) and (16), respectively.

Based on the analysis results in Section 4.1, we note that it is difficult to derive the closed-form solutions for the performance criteria. It is also an arduous task to address the strict monotonicity of the cost function. The use of traditional optimization algorithms, such as the stochastic gradient descent method, the Lagrangian duality method, and the Gauss–Newton method, are inappropriate for obtaining the optimal system parameters. For the purpose of jointly optimizing the sleep parameter, the sleep-delay parameter and the waking-up threshold, we turn to an improved intelligent searching algorithm.

A genetic algorithm is a method used to search for the globally optimal solution of objective functions by simulating the natural evolutionary process [40,41]. In a conventional genetic algorithm, both the crossover probability and the mutation probability are set statically. We note that the fixed

crossover probability and the mutation probability will make the genetic algorithm premature and easy to become trapped in a local optimum. To improve the searching speed, and overcome premature and local convergence, we present an improved genetic algorithm by dynamically adjusting the crossover probability and the mutation probability. Furthermore, to make the initialization more diverse, we use chaotic equations to initialize the individuals in the population [42]. The main steps for the improved genetic algorithm are given in Algorithm 1.

Algorithm 1 Improved genetic algorithm to obtain $(N,\theta,\beta)^*$ and $F((N,\theta,\beta)^*)$.

Step 1: Initialize the sleep parameter with the lower bound $\theta_l = 0.1$ and the upper bound $\theta_u = 0.9$, the sleep-delay parameter with the lower bound $\beta_l = 0.1$ and the upper bound $\beta_u = 10$. Initialize the minimum crossover probability $P_{cl} = 0.001$ and the maximum crossover probability $P_{cm} = 0.1$, the minimum mutation probability $P_{ml} = 0.4$ and the maximum mutation probability $P_{mm} = 0.95$. Set the initial number of evolution generation as $gen = 1$, the maximum evolution generation as $gen_{max} = 50$. Set the initial waking-up threshold as $N = 1$, the maximum waking-up threshold as $N_{max} = 50$.

Step 2: Set the population size as $M = 100$, and initialize each individual as $(\theta,\beta)_i^N$, $i \in \{1,2,\ldots,M\}$ in population $\boldsymbol{S}$ by using chaotic equations:

$(\theta,\beta)_1^N = rand(2,1)$

for $i = 2 : M$

$(\theta,\beta)_i^N = r \times (\theta,\beta)_{i-1}^N \times (1 - (\theta,\beta)_{i-1}^N)$

endfor

% $rand(2,1)$ represents a 2×1 dimensional matrix, the value of the elements are random

% between 0 and 1. r is the chaotic factor, $r = 3.85$.

Step 3: For each individual $(\theta,\beta)_i^N$, $i \in \{1,2,\ldots,M\}$, calculate the fitness $F((\theta,\beta)_i^N)$, the selection probability $P((\theta,\beta)_i^N)$ and the cumulative probability $C((\theta,\beta)_i^N)$.

$$F\left((\theta,\beta)_i^N\right) = f_s W\left((\theta,\beta)_i^N\right) - f_v E_v\left((\theta,\beta)_i^N\right),$$

$$P\left((\theta,\beta)_i^N\right) = \frac{F\left((\theta,\beta)_i^N\right)}{\sum_{i=1}^{M} F\left((\theta,\beta)_i^N\right)},$$

$$C\left((\theta,\beta)_i^N\right) = \sum_{j=1}^{i} P\left((\theta,\beta)_i^N\right).$$

% $W((\theta,\beta)_i^N)$ and $E_v((\theta,\beta)_i^N)$ are the average sojourn time of tasks and the energy

% conservation level of system when the combination of the sleep parameter and the

%sleep-delay parameter is $(\theta,\beta)_i^N$, respectively.

Step 4: Calculate the crossover probability P_c and the mutation probability P_m.

$$P_c = P_{cl} - \frac{(P_{cm} - P_{cl}) * gen}{gen_{max}},$$

$$P_m = P_{ml} + \frac{(P_{mm} - P_{ml}) * gen}{gen_{max}}.$$

Step 5: Perform the genetic operation to update $\boldsymbol{S}$.

for $j = 1 : M$

$slen = selection(\boldsymbol{S}, C((\theta,\beta)_j^N))$

% Select two individuals with the maximum cumulative probability for crossing

%and mutating.

$\boldsymbol{S} = crossover(\boldsymbol{S}, slen, P_c)$

% Cross the selected individuals.

$\boldsymbol{S} = mutation(\boldsymbol{S}, slen, P_m)$

% Mutate the selected individuals.

endfor

Step 6: Check the number of evolution generation.

if $gen < gen_{max}$

then $gen = gen + 1$, go to **Step 3**

endif

Step 7: Select the optimal individual among the population S.

$$(\theta,\beta)^N = \underset{i\in\{1,2,\cdots,M\}}{\arg\min} \left\{F\left((\theta,\beta)_i^N\right)\right\},$$

$$F((\theta,\beta)^N) = f_s W((\theta,\beta)^N) - f_v E_v((\theta,\beta)^N).$$

% $(\theta,\beta)^N$ denotes the optimal parameter combination when the waking-up threshold is N.

Step 8: Check the waking-up threshold N.

if $N < N_{max}$

then $N = N + 1$, go to **Step 2**

endif

Step 9: Choose the minimum cost in the $F\left((\theta,\beta)^z\right)$, $z \in [1, N_{max}]$, record the corresponding waking-up threshold N^*, the sleep parameter θ^* and the sleep-delay parameter β^*, constitute the optimal parameter combination $(N,\theta,\beta)^*$.

Step 10: Output the optimal parameter combination $(N,\theta,\beta)^*$ and the minimum system cost $F\left((N,\theta,\beta)^*\right)$.

5. Statistical Results

The innovative point of our proposed strategy is the introduction of a sleep-delay scheme and a waking-up threshold. To demonstrate the effectiveness of task scheduling strategy proposed in this paper, we present results comparisons between the conventional strategy without a sleep-delay scheme [43] and the conventional synchronous multi-sleep strategy without a waking-up threshold [44].

We carried out the system experiments with analysis and simulation to investigate the proposed strategy and validate the system model. All the experiments were performed on a personal computer configured with AMD Ryzen 3 2200, CPU @ 3.50 GHz, 16.00 GB RAM and 4T disk. Referring to [45], we set the experimental parameters as in Table 1.

Table 1. Experimental parameters.

Parameters	Values
Total number k of VMs in the system	20
Task arrival rate λ	0.4 (tasks/ms)
Service rate μ	0.2 (tasks/ms)
Energy consumption level C_a of a busy VM	20 (mW)
Energy consumption level C_s of a sleeping VM	2 (mW)
Energy consumption level C_t of each switching	12 (mJ)
Energy consumption level C_l of each listening	4 (mJ)

In all of the following figures, the analytical results and the simulation results are illustrated as lines and markers, respectively.

Figure 2 illustrates how the average sojourn time W of tasks changes with the sleep parameter θ for the different waking-up thresholds N and the different sleep-delay parameters β. The larger the sleep-delay parameter is, the less likely it is that the tasks arriving at the system after the awake state are served immediately, thus the average sojourn time of tasks is greater. As the sleep parameter increases, the tasks arriving at the system within a sleep period need to wait a shorter time for the completion of the sleep period, thus the average sojourn time of tasks decreases. As the waking-up threshold increases, the tasks have to wait longer in the sleep state, hence the average sojourn time of tasks increases.

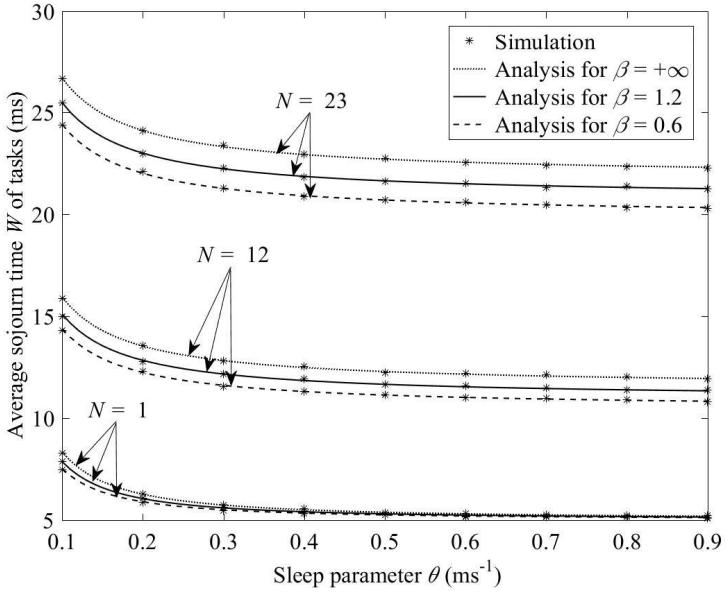

Figure 2. The average sojourn time W of the tasks.

Figure 3 illustrates how the energy conservation level E_v of the system changes with the sleep parameter θ for the different waking-up thresholds N and the different sleep-delay parameters β. The larger the sleep-delay parameter is, the more likely it is that the PM will switch into the sleep state from the sleep-delay state, thus more energy will be conserved. On the other hand, with a larger sleep parameter, the sleep period gets shorter, and the listening frequency increases. Additional energy consumption resulting from listening increases, and the energy conservation decreases. As the waking-up threshold increases, the PM stays in the sleep state longer accumulating tasks, hence the energy conservation level of the system increases.

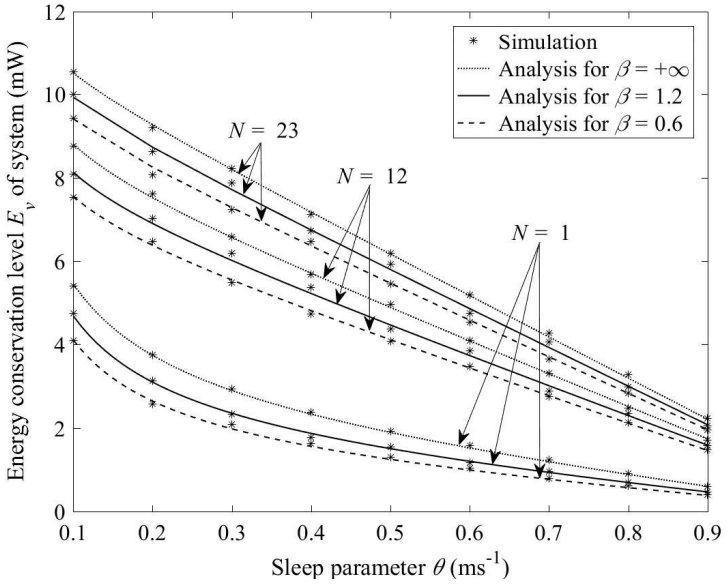

Figure 3. The energy conservation level E_v of the system.

In Figures 2 and 3, the statistical results with $\beta = +\infty$ are for a conventional strategy without a sleep-delay scheme, and the statistical results with $N = 1$ are for a conventional synchronous

multi-sleep strategy without a waking-up threshold. Compared to the conventional strategy without a sleep-delay scheme [43] as the results with $\beta = +\infty$ in Figure 2, we observed that our proposed strategy performs better in guaranteeing the response performance. Compared to the conventional synchronous multi-sleep strategy without a waking-up threshold [44] as the results with $N = 1$ in Figure 3, we observed that our proposed strategy is more effective at reducing energy consumption.

Based on the statistical results shown in Figures 2 and 3, we conclude that the system performance of the proposed strategy is determined by the system parameters in terms of the sleep parameter, the sleep-delay parameter and the waking-up threshold. For the real-time applications, such as the interactive traffic presented in [46–48], the response performance is highly valued, therefore the sleep parameter should be set larger, while the waking-up threshold and the sleep-delay parameter should be set smaller. Conversely, for the non-real-time applications, such as the file transfer presented in [49,50], the energy conservation is urgently required, therefore the sleep parameter should be set smaller, while the waking-up threshold and the sleep-delay parameter should be set larger. Thus, a compromise between response performance and energy conservation should be considered when setting system parameters in our proposed task scheduling strategy.

By applying the parameters used in Figures 2 and 3 into the improved genetic algorithm and setting the impact factors $f_s = 0.3$, $f_v = 0.4$ as an example, we provide numerical results for the optimal parameter combination (N, θ, β) with the different service rates μ in Table 2.

Table 2. The numerical results for the system optimization.

Service Rate μ	Optimal Combination (N, θ, β)	System Cost F
0.07	(6, 0.1468, 7.9699)	4.2246
0.10	(6, 0.1269, 9.8577)	2.7236
0.13	(5, 0.1481, 8.5751)	1.7222
0.16	(4, 0.2306, 6.9526)	0.9801
0.19	(4, 0.1357, 6.1439)	0.4045

6. Conclusions

This study is original in that it put forth an energy efficient task scheduling strategy in a cloud computing system, and studied the compromise between the response performance and the energy conservation level. By introducing the sleep-delay parameter and the waking-up threshold, we firstly proposed a novel sleep mode based task scheduling strategy. The proposed strategy is representative of real-world cloud scenarios. We modeled the proposed task scheduling strategy as a type of vacation queueing system. Based on the model analysis in the steady state, we derived the average sojourn time of tasks and the energy conservation level of system. Taking into account numerous and different cloud services, we were able to extend our presented model to investigate the strategy performance over a more complicated public cloud scenario. Statistical results with analysis and simulation showed that a larger sleep parameter, a smaller waking-up threshold and a smaller sleep-delay parameter may lead to a lower average sojourn time, while a smaller sleep parameter, a larger waking-up threshold and a larger sleep-delay parameter may result in a higher energy conservation level. Correspondingly, we developed a cost function to trade off the different performance measures and gave an improved genetic algorithm to search the optimal parameter combination. Numerical results for the system optimization indicated that an appropriate parameter setting can achieve the minimum cost. The proposed strategy presented in this paper is efficient under the network environment with a heavy load of big data streams. In our future research, we will investigate the task scheduling strategy under the network environment with a light load traffic by considering deep sleep mechanism.

Author Contributions: W.Y. proposed and discussed with Y.T. the energy efficient task scheduling strategy. Then, W.Y. advised on the system modeling, analysis and the creation of thesis, and worked on the completion of this research. S.J. worked to analyze the system model and present the system performance and numerical calculation.

W.Z. and X.W. mainly conducted detailed analysis, and numerically verified the proposed energy efficient task scheduling strategies using numerical analysis and simulation. They also made effort to complete this paper.

Funding: This research was supported in part by grants from NSF (Grant Nos. 61872311 and 61472342), and Hebei Province NSF (Grant No. F2017203141) of China, and was supported in part by MEXT and JSPS KAKENHI (Grant No. JP17H01825) of Japan.

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

References

1. Olokunde, T.; Misra, S.; Adewumi, A. Quality Model for Evaluating Platform as a Service in Cloud Computing. In Proceedings of the International Conference on Information and Software Technologies (2017), Da Nang, Vietnam, 16–19 April 2017; pp. 280–291.
2. Mondal, S.; Das, G.; Wong, E. An Analytical Cost-Optimal Cloudlet Placement Framework over Fiber-Wireless Networks with Quasi-Convex Latency Constraint. *Electronics* **2019**, *8*, 404. [CrossRef]
3. Fatima, A.; Javaid, N.; Butt, A.; Sultana, T.; Hussain, W.; Bilal, M.; Aqeel, M.; Hashmi, R.; Akbar, M.; Ilahi, M. An Enhanced Multi-Objective Gray Wolf Optimization for Virtual Machine Placement in Cloud Data Centers. *Electronics* **2019**, *8*, 218. [CrossRef]
4. Madni, S.; Latiff, M.; Coulibaly, Y. Recent Advancements in Resource Allocation Techniques for Cloud Computing Environment: A Systematic Review. *Clust. Comput.* **2017**, *20*, 2489–2533. [CrossRef]
5. Zhang, Y.; Yao, J.; Guan, H. Intelligent Cloud Resource Management with Deep Reinforcement Learning. *IEEE Cloud Comput.* **2017**, *4*, 60–69. [CrossRef]
6. Abdullahi, M.; Ngadi, M.; Abdulhamid, S. Symbiotic Organism Search Optimization Based Task Scheduling in Cloud Computing Environment. *Future Gener. Comput. Syst.* **2016**, *56*, 640–650. [CrossRef]
7. Ullah, A.; Li, J.; Shen, Y.; Hussain, A. A Control Theoretical View of Cloud Elasticity: Taxonomy, Survey and Challenges. *Clust. Comput.* **2018**, *21*, 1735–1764. [CrossRef]
8. Dechouniotis, D.; Leontiou, N.; Athanasopoulos, N.; Christakidis, A.; Denazis, S. A Control-Theoretic Approach Towards Joint Admission Control and Resource Allocation of Cloud Computing Services. *Int. J. Netw. Manag.* **2015**, *25*, 159–180. [CrossRef]
9. Abdulhamid, S.; Latiff, M.; Bashir, M. On-Demand Grid Provisioning using Cloud Infrastructures and Related Virtualization Tools: A Survey and Taxonomy. *Int. J. Adv. Stud. Comput. Sci. Eng.* **2014**, *3*, 49–59.
10. Li, X.; Garraghan, P.; Jiang, X.; Wu, Z.; Xu J. Holistic Virtual Machine Scheduling in Cloud Datacenters Towards Minimizing Total Energy. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 1317–1331. [CrossRef]
11. Duan, K.; Fong, S.; Siu, S.; Song, W.; Guan, S. Adaptive Incremental Genetic Algorithm for Task Scheduling in Cloud Environments. *Symmetry* **2018**, *10*, 168. [CrossRef]
12. Zakarya, M.; Gillam, L. Energy Efficient Computing, Clusters, Grids and Clouds: A Taxonomy and Survey. *Sustain. Comput. Inform. Syst.* **2017**, *14*, 13–33. [CrossRef]
13. You, C.; Huang, K.; Chae, H. Energy Efficient Mobile Cloud Computing Powered by Wireless Energy Transfer. *IEEE J. Sel. Areas Commun.* **2016**, *2*, 1757–1771. [CrossRef]
14. Hashem, L.; Yaqoob, L.; Mokhtar, S.; Gani, A.; Khan, S. The Rise of "Big Data" on Cloud Computing: Review and Open Research Issues. *Inf. Syst.* **2015**, *47*, 98–115. [CrossRef]
15. Yang, B.; Tan, F.; Dai, Y.; Guo, S. Performance Evaluation of Cloud Service Considering Fault Recovery. In Proceedings of the IEEE International Conference on Cloud Computing, Beijing, China, 1–4 December 2009; Springer: Berlin, Germany, 2009; Volume 5931, pp. 571–576.
16. Fellera, E.; Ramakrishnan, L.; Morin, C. Performance and Energy Efficiency of Big Data Applications in Cloud Environments: A Hadoop Case Study. *J. Parallel Distrib. Comput.* **2015**, *79–80*, 80–89. [CrossRef]
17. Xia, Y.; Zhou, M.; Luo, X.; Pang, S.; Zhu, Q. A Stochastic Approach to Analysis of Energy-Aware DVS-Enabled Cloud Datacenters. *IEEE Trans. Syst. Man Cybern.* **2015**, *45*, 73–83.
18. Cheng, C.; Li, J.; Wang, Y. An Energy-Saving Task Scheduling Strategy Based on Vacation Queueing Theory in Cloud Computing. *Tsinghua Sci. Technol.* **2015**, *20*, 28–39. [CrossRef]
19. Chen, Y.; Chang, M.; Liang, W.; Lee, C. Performance and Energy Efficient Dynamic Voltage and Frequency Scaling Scheme for Multicore Embedded System. In Proceedings of the IEEE 6th International Conference on Consumer and Electronics (2016), Las Vegas, NV, USA, 27–30 January 2016; pp. 58–59.

20. Shen, Y.; Bao, Z.; Qin, X.; Shen, J. Adaptive Task Scheduling Strategy in Cloud: When Energy Consumption Meets Performance Guarantee. *World Wide Web* **2017**, *20*, 155–173. [CrossRef]
21. Khosravi, A.; Andrew, L.; Buyya, R. Dynamic VM Placement Method for Minimizing Energy and Carbon Cost in Geographically Distributed Cloud Data Centers. *IEEE Trans. Sustain. Comput.* **2017**, *2*, 183–196. [CrossRef]
22. Kempa, W. Time-Dependent Analysis of Transmission Process in a Wireless Sensor Network with Energy Efficient Mechanism Based on Threshold Waking up. In Proceedings of the IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (2015), Stockholm, Sweden, 28 June–1 July 2015; pp. 26–30.
23. Mcbay, C.; Parr, G.; Mcclean G. Energy Efficient in Data Center Servers using Optimal Scheduling to Ensure QoS. In Proceedings of the 7th International Conference on Cloud Computing, GRIDs, and Virtualization (2016), Rome, Italy, 20–24 March 2016; pp. 56–60.
24. Lawanyashri, M.; Balusamy, B.; Subha, S. Threshold-Based Workload Control for an Under-Utilized Virtual Machine in Cloud Computing. *Int. J. Intell. Eng. Syst.* **2016**, *9*, 234–241. [CrossRef]
25. Singh, D.; Devgan, M. Task Scheduling with Multilayer Hybrid Energy Efficient Approach in Green Cloud Computing. *Int. J. Sci. Res. Dev.* **2016**, *4*, 814–818.
26. Madni, S.; Abd, L.; Abdullahi, M.; Abdulhamid S.; Usman, M. Performance Comparison of Heuristic Algorithms for Task Scheduling in IaaS Cloud Computing Environment. *PLoS ONE* **2017**, *12*, e0176321. [CrossRef] [PubMed]
27. Qiu, M.; Ming, Z.; Li, J.; Gai, K.; Zong, Z. Phase-Change Memory Optimization for Green Cloud with Genetic Algorithm. *IEEE Trans. Comput.* **2015**, *6*, 3528–3540. [CrossRef]
28. Huang, S.; Jiau, M.; Lin, C. A Genetic-Algorithm-Based Approach to Solve Carpool Service Problems in Cloud Computing. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 352–364. [CrossRef]
29. Jiang, Y.; Jiang, J.; Zhang, Y. A Novel Fuzzy Multiobjective Model using Adaptive Genetic Algorithm Based on Cloud Theory for Service Restoration of Shipboard Power Systems. *IEEE Trans. Power Syst.* **2012**, *27*, 612–620. [CrossRef]
30. Jin, S.; Wang, X.; Yue, W. A Task Scheduling Strategy with a Sleep-Delay Timer and a Waking-Up Threshold in Cloud Computing. In Proceedings of the 13th International Conference on Queueing Theory and Network Applications (2018), Tsukuba, Japan, 25–27 July 2018; pp. 1115–1123.
31. Mevada, A.; Patel, H.; Patel, N. Enhanced Energy Efficient Virtual Machine Placement Policy for Load Balancing in Cloud Environment. *Int. J. Curr. Res. Rev.* **2017**, *9*, 50–53.
32. Khoshkholghi, M.; Derahman, M.; Abdullah, A.; Subramaniam, S.; Othman, M. Energy-Efficient Algorithms for Dynamic Virtual Machine Consolidation in Cloud Data Centers. *IEEE Access* **2017**, *5*, 10709–10722. [CrossRef]
33. Dzheparov, F.; Shestopal, V. Asymptotically Exactly Solvable Models of Processes in Stochastically Homogeneous Disordered Lattice Media. *Theor. Math. Phys.* **2003**, *135*, 549–565.
34. Zhao, Y.; Jin, S.; Yue, W. Performance Analysis of Cognitive Radio Networks for Secondary Users with Slotted Central Control. *Telecommun. Syst.* **2017**, *66*, 689–699. [CrossRef]
35. He, H.; Yuan, D.; Hou, Y.; Xu, J. Preconditioned Gauss-Seidel Iterative Method for Linear Systems. *Int. Forum Inf. Technol. Appl.* **2009**, *1*, 382–385.
36. Yue, D.; Yu, J.; Yue, W. A Markovian Queue with two Heterogeneous Servers and Multiple Vacations. *J. Ind. Manag. Optim.* **2009**, *5*, 453–465.
37. Honnappa, H.; Jain, R.; Ward, A. A Queueing Model with Independent Arrivals, and its Fluid and Diffusion Limits. *Queueing Syst.* **2015**, *80*, 71–103.
38. Nguyen, T.; Gribaudo, M.; Pernici, B. Characterizing Energy per Job in Cloud Applications. *Electronics* **2015**, *6*, 90–114.
39. Chen, Y.; Xie, G.; Li, R. Reducing Energy Consumption with Cost Budget using Available Budget Preassignment in Heterogeneous Cloud Computing Systems. *IEEE Access* **2018**, *6*, 20572–20583. [CrossRef]
40. Peiravi, A.; Mashhadi, H.; Javadi, S. An Optimal Energy-Efficient Clustering Method in Wireless Sensor Networks using Multi-Objective Genetic Slgorithm. *Int. J. Commun. Syst.* **2013**, *26*, 114–126. [CrossRef]
41. Hussain, S.; Matin, A.; Islam, O. Genetic Slgorithm for Energy Efficient Clusters in Wireless Sensor Networks. In Proceedings of the 4th International Conference on Information Technology (2007), Las Vegas, NV, USA, 2–4 April 2007; pp. 147–154.

42. Gandomi, A.; Yang, X. Chaotic Bat Algorithm. *J. Comput. Sci.* **2014**, *5*, 224–232. [CrossRef]
43. Rajagopal, M.; Jayarajan, P.; Dhasarathan, V.; Sivasankaran, V.; Udaiyakumar, R. Performance Analysis of Contention Based Priority Queuing Model using N-Policy Model for Cluster Based Sensor Networks. In Proceedings of the 7th IEEE International Conference on Communication and Signal Processing (2018), Pune, India, 23–24 November 2018; pp. 229–233.
44. Jin, S.; Han, L.; Yue, W. Performance Evaluation for the Power Saving Class Type III with a Sleep-Delay in IEEE 802.16e. In Proceedings of the 4th International Conference on Queueing Theory and Network Applications (2009), Singapore, 29–31 July 2009; pp. 1–6.
45. Jin, S.; Hao, S.; Yue, W. Energy-Efficient Strategy with a Speed Switch and a Multiple-Sleep Mode in Cloud Data Centers. In Proceedings of the 12th International Conference on Queueing Theory and Network Applications (2017), Qinhuangdao, China, 21–23 August 2017; pp. 143–154.
46. Chen, H.; Zhu, X.; Guo, H.; Qin, X.; Wu, J. Towards Energy-Efficient Scheduling for Real-Time Tasks under Uncertain Cloud Computing Environment. *J. Syst. Softw.* **2015**, *99*, 20–35. [CrossRef]
47. Stavrinides, G.; Karatza, H. An Energy-Efficient, QoS-Aware and Cost-Effective Scheduling Approach for Real-Time Workflow Applications in Cloud Computing Systems utilizing DVFS and Approximate Computations. *Future Gener. Comput. Syst.* **2019**, *96*, 216–226. [CrossRef]
48. Tantalaki, N.; Souravlas, S.; Roumeliotis, M. A Review on Big Data Real-Time Stream Processing and its Scheduling Techniques. *Int. J. Parallel Emerg. Distrib. Syst.* **2019**, 1–31. [CrossRef]
49. Kao, B.; Garcia-Molina, H. Scheduling soft real-time jobs over dual non-real-time servers. *IEEE Trans. Parallel Distrib. Syst.* **1996**, *7*, 56–68. [CrossRef]
50. Štefanič, P.; Cigale, M.; Jones, A.; Knight, L.; Taylor, I.; Istrate, C. SWITCH Workbench: A Novel Approach for the Development and Deployment of Time-Critical Microservice-Based Cloud-Native Applications. *Future Gener. Comput. Syst.* **2019**, *99*, 197–212. [CrossRef]