Accepted Manuscript

Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems

Hancong Duan, Chao Chen, Geyong Min, Yu Wu

PII: S0167-739X(16)30029-2
DOI: http://dx.doi.org/10.1016/j.future.2016.02.016
Reference: FUTURE 2969

To appear in: *Future Generation Computer Systems*

Received date: 24 November 2015
Revised date: 25 January 2016
Accepted date: 20 February 2016

Please cite this article as: H. Duan, C. Chen, G. Min, Y. Wu, Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems, *Future Generation Computer Systems* (2016), http://dx.doi.org/10.1016/j.future.2016.02.016

Highlights (for review)

- An efficient prediction model based on fractal mathematics is developed.
- An improved ant colony algorithm for optimizing energy consumption is proposed.
- The proposed approach shows excellent energy efficiency and resource utilization.

*Biographies (Text)

Energy-Aware Scheduling of Virtual Machines in Heterogeneous Cloud Computing Systems

Hancong Duan[a,*], Chao Chen[a], Geyong Min[b], Yu Wu[a]

[a] *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China*
[b] *College of Engineering, Department of Mathematics and Computer Sciences, University of Exeter, Exeter, Devon, United Kingdom*

Abstract

With the rapid development of cloud computing, how to reduce energy consumption as well as maintain high computation capacity has become a timely and important challenge. Existing Virtual Machines (VMs) scheduling schemes have mainly focused on enhancing the cluster resource utilization and reducing power consumption by improving the legacy "bin-packing" algorithm. However, different resource-intensive applications running on VMs in realistic scenarios have significant effects on the system performance and energy consumption. Furthermore, instantaneous peak loads may lead to a scheduling error, which can significantly impede the energy efficiency of scheduling algorithms. In this paper, we propose a new scheduling approach named *PreAntPolicy* that consists of a prediction model based on fractal mathematics and a scheduler on the basis of an improved ant colony algorithm. The prediction model determines whether to trigger the execution of the scheduler by virtue of load trend prediction, and the scheduler is responsible for resource scheduling while minimizing energy consumption under the premise of guaranteeing the Quality-of-Service (QoS). Through extensive analysis and simulation experiments using real workload traces from the compute clusters of Google, the performance results demonstrate that the proposed approach exhibits excellent energy efficiency and resource utilization. Moreover, this approach offers an effective dynamic capacity provisioning model for resource-intensive applications in a heterogeneous computing environment and can reduce the consumption of system resources and energy when scheduling is triggered by instantaneous peak loads.

Keywords: Cloud Computing, Resource Allocation, Energy Efficiency, Fractal Prediction Model, Resource-Intensive Applications

*Corresponding author
Email addresses: duanhancong@uestc.edu.cn (Hancong Duan), cqwzchenchao@163.com (Chao Chen), G.Min@exeter.ac.uk (Geyong Min), ThinkingYu2015@163.com (Yu Wu)

1. Introduction

cloud computing has emerged as a new business model of computation and resource storage based on on-demand access to potentially significant amounts of remote data center capabilities [1]. More and more data centers have successively been established because of the rapid development of cloud computing. However, these data centers are running a large number of applications that do not only consume considerable computing and storage capacity, but also consume huge amount of energy. It is reported that data centers in the US accounted for approximately 3% of the total electricity consumption in 2011 [2], and that the operational expenses will exceed the costs to purchase server hardware by 2015 [3]. Motivated by these facts, reducing power consumption and cutting down energy cost has become a primary concern for today's data center operators.

Many existing studies have been made to improve the energy efficiency of data centers. A series of technical methods are proposed for different aspects, including better cooling technology, temperature control, Dynamic Voltage and Frequency Scaling (DVFS) [4], software components such as resource virtualization [5], and load-balancing algorithms. An effective approach for energy saving in data centers is to dynamically adjust the data center capacity and provide resources for the application systems by scheduling algorithms. However, the scheduling algorithm in the field of cloud computing is known to be a challenge, as it requires a careful understanding of resource demand characteristics, as well as through consideration of various energy cost factors. Traditional scheduling algorithms pay more attention to allocating resources fairly, but in most realistic scenarios, different resource-intensive applications running on virtual machines also have an important effect on the system performance and energy consumption. Therefore, these aspects should be taken into account seriously. Cloud computing applications always use multiple types of resources, such as CPUs, memory, network bandwidth, and so on. From this perspective, the ability to schedule a variety of resources becomes indispensable. Finally, because each scheduling is based on the resources required of the virtual machines in the current moment, instantaneous peak loads may lead to a scheduling error, which can significantly impede the energy-saving performance of scheduling algorithms. In order to reserve resources more effectively, a scheduling algorithm needs to consider the load trend of a data center when it makes scheduling decisions.

To address these problems, we aim to design, implement, and evaluate an energy-aware scheduling algorithm that integrates a fractal prediction model for optimizing energy consumption in a cloud computing environment by shutting down redundant machines. In contrast to the existing work based on the legacy "bin-packing" algorithm, we formulate the problem as a convex optimization problem-one that considers resource-intensive applications and a heterogeneous computing environment. Moreover, our algorithm avoids scheduling errors triggered by a specific threshold.

The major contributions of this paper are as follows:

1) We develop an efficient prediction model based on fractal mathematics. The model can assist the algorithm to decide to turn on/off hosts. Through this wayit help to avoid the performance and energy loss which is triggered by instantaneous peak loads on account of scheduling.

2) We propose an improved ant colony algorithm that is used to optimize energy consumption and meet resource-intensive application demands in the heterogeneous cloud computing environment.

The remainder of this paper is organized as follows: Section 2 presents a survey of related work in the current literature and highlights the motivation of this. In Section 3, we describe the overall architecture model of the proposed approach followed by a detailed description of the load indicator, the developed fractal prediction model, and the improved ant colony algorithm applied to the scheduler. Section 4 presents the evaluation for our proposed algorithm and demonstrates its benefits under various working conditions. Finally, the conclusions are drawn in Section 5.

2. BACKGROUND AND MOTIVATION

2.1. RELATED WORKS

This section surveys the related work reported in the current literature to reduce energy consumption and optimize resource utilization in the cloud computing area.

Based on the DVFS technology, Wang *et al.* [6] proposed an algorithm to reduce the energy consumption of parallel jobs. With the premise of not increasing the overall execution time, the algorithm appropriately extends the execution time of non-stressed tasks while lowering the voltage and frequency of processors to reduce energy costs. There is no doubt that this algorithm is an energy-efficient scheduling policy under the multi-core environment. Nevertheless, it does not consider parallel operation for the demand of other resources in addition to the CPU. Hosseinimotlagh [7] proposed a cooperative two-tier energy-aware scheduling approach for real-time tasks to benefit both cloud providers and their customers. However, it does not make full use of the resources of the whole cluster. Similarly, Watanabe *et al.* [8], in contrast, focused on dependency between tasks and proposed new task schedulers for cloud computing aiming to achieve energy savings. By using statistical method, Wu [9] built an energy consumption model of the Cloud Computing system and proposed a virtual machine scheduling algorithm to improve the energy efficiency of the system. In order to save resources, it does have very good effect by using this way. Whereas, task dependency leads to low cloud utilization.

Using an internal energy savings plan in the operating system, Nathuj and Schwan [10] proposed a virtual energy management structure called Virtual Power Management(VPM). It adopts a specific energy-saving measure for the whole system by intercepting hardware instruction requests from the guest operating system. This approach uses the client computer operating system to simplify the mass of the energy-saving decision phase, but it is difficult for the

method to make a comprehensive implication for the whole system. This is because the single guest operating system can only monitor the local execution environment. Therefore, from a global perspective, it is very difficult to guarantee the energy saving effect .Analogously, Liu *et al.* [11] presented an optimization model for task scheduling to minimize energy consumption in cloud computing data centers. The proposed approach was formulated as an integer programming problem to minimize energy consumption by scheduling tasks to a minimum number of servers while keeping the task response time constraints. The number of servers that required to service a homogenous workload is somehow large. However, uneven network load distribution may be caused due to such a greedy approach.

Stoess [12] regarded energy-savings management as an optimization problem coupled with energy constraints. In order to support energy conservation, energy cost constraints are assigned to individual nodes, and the corresponding energy monitoring system is applied. This scheme is based on hard energy-saving constraints that ensure that application efficiency is in place. Thus, it may increase the time of Service-Level Agreement (SLA) violation. At the same time, it does not apply to the current private cloud environment. However, this approach make a contribution to promoting and popularizing the idea of green and low carbon systems. Focusing on reducing energy consumption through migration of virtual machines, Hossain *et al.* [13] proposed a solution which attached a potential score associated with energy-saving to each running virtual machine. Every running time, the algorithm merges the virtual machine with the highest score to the one with the lowest and shuts down redundant physical machines(PMs). Unsurprisingly, this algorithm can reduce energy consumption caused by the migration of virtual machines. However, migrating energy consumption is just a part of the energy source in the cloud computing environment. This algorithm is more suitable as a component of an energy-efficient scheduling algorithm. Besides, a long scheduling delay, which can significantly hinder the performance of some services, can easily occur.

Treating the scheduling of virtual machines as 0-1 knapsack problems, Srikantaiah and Kansal [14] provided a new train of thought for the resource optimization of cloud computing. Based on the same approach, Buyya [15] proposed an energy-saving computing framework that maximizes the advantages of heterogeneous resources using the improved Best Fit Decreasing (BFD) algorithm to allocate resources for virtual machines. The 0-1 knapsack problem can be a good simulation of optimization problems for energy saving or load constraints. However, for multi-objective optimization problems, including task execution efficiency, energy saving, and load balancing, or some other targets, the simulation efficiency will be affected by the competition between multiple targets. Moreover, given that it is difficult to set appropriate weights, the overall effect is not ideal.

Niu *et al.* [16] proposed a Semi-Elastic Cluster (SEC) computing model for organizations to reserve and dynamically resize a virtual cloud-based clusters. Similarly, Kliazovich *et al.* [17] emphasized the role of communication fabric and presented a scheduling solution, named e-STAB, which takes the traffic

requirements of cloud applications into account for energy-efficient job allocation and traffic load balancing in data center. Nevertheless, it is difficult to know how the execution time of each task will change in this situation. Focusing on energy-aware fault tolerant scheduling in public, multi-user cloud systems, Gao *et al.* [18] explored a way to make cloud service provider achieve high error coverage and fault tolerance confidence while minimizing the global energy cost under user deadline constraints. However, from the perspective of global, it is not an optimal scheme to place the virtual machine.

Szymanski [19] explored the use of a recently-proposed Future-Internet network combined with a Maximum-Flow Minimum-Energy routing algorithm to achieve low-latency energy-efficient communications in Global Scale Cloud Computing systems. In order to maintain sustainable Cloud computing facing with ever-increasing problem complexity and big data size, Cao [20] proposed an energy-efficient scientific workflow scheduling algorithm to minimize energy consumption and CO2 emission while satisfying certain Quality-of-Service (QoS). Nonetheless, the method that pursuing low latency, low energy consumption urgently may lead to a variety of errors.

2.2. MOTIVATION

Motivated by the defects in most existing scheduling algorithms, we propose a new scheduler for optimizing energy consumption. Our approach investigates the source of power consumption in the cloud computing environment from a global perspective. It also takes into account the impacts of heterogeneity in the computing environment, as well as different applications resource requirements. Furthermore, we devise a prediction model based on fractal mathematics [21] to assist the algorithm make appropriate decisions to turn the system on/off, making the scheduler more practical. Finally, our approach is also lightweight, independent, and easy to deploy because its implementation is very simple. Meanwhile, this algorithm runs only on the management node.

3. The PROPOSED SCHEDULING APPROACH

This section starts with the presentation of the model of the proposed scheduling approach followed by a detailed description of the load indictor. Then we focus on the algorithm design and implementation.

3.1. MODEL OF THE PROPOSED SCHEDULING APPROACH

The model of the proposed scheduling approach is shown in Fig.1. It consists of four functional modules. The monitoring module is responsible for periodically gathering resource usage from the physical machines in the cluster such as CPU utilization and disk utilization. The prediction model determines whether to trigger the execution of the scheduler through load trend prediction and results evaluation. The scheduler is responsible for migrating virtual machines to the specified hosts according to the instructions from the prediction model. The allocation model is in charge of dispatching a new task to a virtual machine,

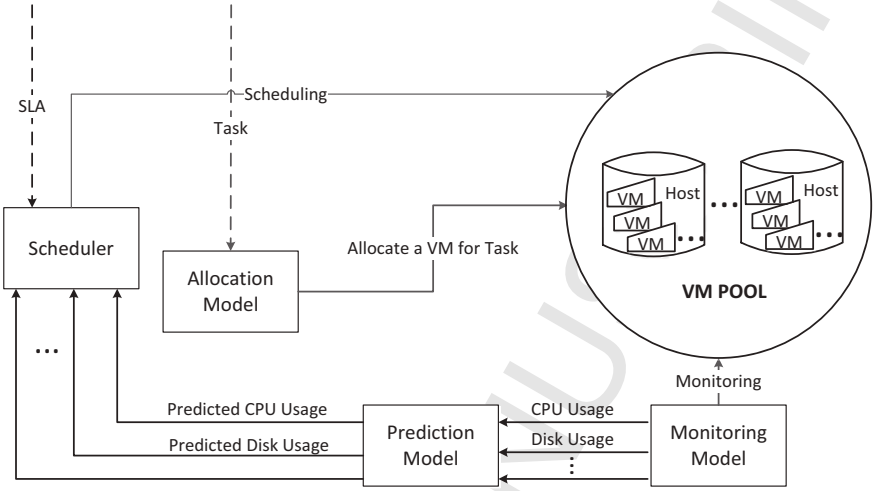

Figure 1: Model of the proposed scheduling approach

and it allocates appropriate resources for the task. The overall workflow of this model can be roughly described as: the prediction model predicts the coming resource utilization of a data center through the data collected by the monitoring model. Then the scheduler schedules virtual machines based on the instructions from the prediction model. The main job of the allocation model is assigning an appropriate virtual machine to a new cloud task produced by clients. Because the prediction model and the scheduler own the core of our algorithm, we will discuss the two module design, implementation and algorithm in detail later.

3.2. LOAD DESCRIPTION

In this study, we focus on two types of resources: CPU and disk I/O bandwidth. However, it has been illustrated that our algorithm has the ability to allocate multiple types of resources. We present a unified and standardized description of the load indicator in order to provide comparability between heterogeneous physical nodes. The CPU load indicator of the ith host machine in the cluster is expressed as $Load_{cpu\_i} = \sum_{k=1}^{N} Usage(k)/Capacity_{cpu\_i}$, where N represents the number of active virtual machines on this host machine, $Capacity_{cpu\_i}$ is its CPU capacity and $Usage(k)$ represents the CPU usage of the kth virtual machine running on this host. CPU capacity is measured by core count. Thus, the CPU load of a cluster, which is made up of M physical machines, can be expressed as $Load_{cpu\_total} = \sum_{i=1}^{M} Load_{cpu\_i}/M$. In the same way, the disk load indicator of the ith host machine in the cluster is expressed as $Load_{io\_i} = \sum_{k=1}^{N} D_k/Total_{io\_i}$ where D_k represents the disk usage

of the kth virtual machine running on this host. $Total_{io\_i}$ is the capacity of disk bandwidth of the host machine. Disk capacity is measured by the data transmission rate (MB/s). Thus, the disk load of a cluster that consists of M physical machines can be expressed as $Load_{io\_total} = \sum_{i=1}^{M} Load_{io\_i}/M$. Finally, the logic resource capacity of the ith single host machine in the cluster is $\mathrm{Res}_{i-\log ic} = \sigma \times Capacity_{cpu\_i} + (1-\sigma)Total_{io\_i}$ where σ stands for the weight factor of the CPU capacity and (1-σ) represents the weight factor of disk I/O bandwidth capacity.

3.3. DESIGN OF PREDICTION MODEL

This section presents the prediction model we developed for predicting the usage of each type of resource. Dinda [22] found that the CPU load had a very strong nature of self-similarity and a long correlation through extensive research. Motivated by this, we used fractal mathematics to predict resource usage in a period of time in the future. In our model, continuous time is divided into a series of discrete time points with a fixed interval like t_{k-i+1}, ... , t_{k-1}, t_k, t_{k+1},..., t_{k+i}. We aim to predict G_{k+1}, ..., G_{k+I}, which represents the usage of the CPU in the cluster at time t_{k+1}, ..., t_{k+i} on the basis of G_{k-i+1}, ..., G_{k-1}, G_k. It can be observed that our model prediction dataset has the same number of values as the history dataset after one prediction. Through this strategy, error scheduling caused by an instantaneous peak load can be prevented. Similarly, we can forecast the disk I/O bandwidth load in the same way.

Thus far, the most effective way to study, parse, and construct fractals is by using an Iterated Function System (IFS). According to the IFS collage theorem, there must be an IFS whose attractor is approximate to a given set. Therefore, we can perform compression transformation to a given set and paste the results together to reconstruct it. Barnsley [23] introduced a practical method to construct IFS, called Fractal Interpolation Theory. With its assistance, all we need do is to apply deterministic iteration to any point and we can obtain the attractor. The attractor line contains the forecast values we need. The affine transformation formula [23] is as follows. Formally, we can set (x, y) to (x_0, y_0), (x_1, y_1), ... , (x_N, y_N) where $x_0 < x_1 < \ < x_N$, and W_n is the attractors of the IFS.

$$W_n \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_n & 0 \\ c_n & d_n \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_n \\ f_n \end{bmatrix} \tag{1}$$

And the following two conditions are satisfied:

$$W_n \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \end{bmatrix} \tag{2}$$

$$W_n \begin{bmatrix} x_N \\ y_N \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{3}$$

From Formulas (1)(2)(3), the following equations can be acquired:

$$\begin{cases} \mathrm{a}_n x_0 + e_n = x_{n-1} \\ \mathrm{a}_n x_N + e_n = x_n \\ \mathrm{c}_n x_0 + d_n y_0 + f_n = y_{n-1} \\ \mathrm{c}_n x_N + d_n y_N + f_n = y_n \end{cases} \tag{4}$$

where d_n is a special parameter in fractal mathematics which is called coordination factor. It is usually specified by the external. In this paper, d_n is the ratio of the average load to the max load in the history dataset. When d_n is given, the other coefficients of W_n can be expressed as:

$$\begin{cases} \mathrm{a}_n = \frac{x_n - x_{n-1}}{x_N - x_0} \\ \mathrm{c}_n = \frac{y_n - y_{n-1}}{x_N - x_0} - d_n \frac{x_N y_0 - x_0 y_N}{x_N - x_0} \\ \mathrm{f}_n = \frac{x_N y_{n-1} - x_0 y_n}{x_N - x_0} - d_n \frac{x_N y_0 - x_0 y_N}{x_N - x_0} \\ \mathrm{e}_n = \frac{x_N x_{n-1} - x_0 x_n}{x_N - x_0} \end{cases} \tag{5}$$

Finally, we apply deterministic iteration to any point to obtain the attractor. The algorithm of the fractal prediction model is intuitively described as follows:

Table 1: Parameters Description

Description	Unit	Symbol
The number of similar days	-	N_s
The number of interpolation points	-	N_p
The ith similar day	-	S^i
The IFS of the ith similar day	-	S^i_IFS
The current day	-	C
The IFS of the current day	-	C_IFS
The attractor in the fractal mathematics	-	A
A random and normalized data	-	R
The IFS with weight	-	IFS_w
The interval between interpolation points	minutes	I
The current time	-	T_{now}
The minimum load in the historical dataset	-	Q_{min}
The maximum load in the historical dataset	-	Q_{max}
The average load in the historical dataset	-	Q_{ave}

Algorithm 1 Fractal Prediction

```
1: procedure BuildIFSforDay(D)
2:     i = 0
3:     beginTime = T_now − (N_p − 1) ∗ I
4:     while beginTime != T_now do
5:         makeHistoricalDataSet(i)
6:         i = i + 1
7:         beginTime = beginTime + I
8:     end while
9:     j = 0
10:    while j! = N_p do
11:        makeNormalizedDataSet(j)
12:        j = j + 1
13:    end while
14:    BuildIFS(D)
15: end procedure
16: procedure FractalPrediction
17:    for i from 0 to N_s step 1 do
18:        BuildIFSforDay(S^i)
19:    end for
20:    BuildIFSforDay(C)
21:    IFS_w = weightProcess(S^i_IFS, C_IFS)(0 ≤ i < N_s)
22:    A = DeterministicIteration(R, IFS_w)
23:    k = 0
24:    while k! = N_p do
25:        getPredictedValue(k)
26:        + + k
27:    end while
28: end procedure
```

Make the current day as a starting point and treat the N_s previous days as similar days. The day just prior to the current day is specially named *thebenchmarkday*. Take I minutes as an interval and the current time as a start, and collect N_p forward interpolation points in the load record of the benchmark day. A historical data set composed of the N_p interpolation points is produced by *makeHistoricalDataSet* procedure. The *makeHistoricalDataSet* procedure normalizes the historical data set using the following formula:

$$\begin{cases} x_i = \frac{T_i - T_{\min}}{T_{\max} - T_{\min}} \\ y_i = \frac{Q_i - Q_{\min}}{Q_{\max} - Q_{\min}} \end{cases} \tag{6}$$

where T_i is the current time and Q_i is the corresponding load. T_{min} and T_{max} represent the minimum and maximum time values, respectively. Accordingly, Q_{min} and Q_{max} are the load values at T_{min} and T_{max}. Then, x_i and y_i denote

the normalized interpolation points. $BuildIFS$ procedure generates IFSs for the current day, benchmark day and all the other similar days. Then, these IFSs are consolidated into one IFS with weight through $weightProcess$ procedure and the weighting factors can be specified by the experience. In general, if a similar day is closer to the current day, the weighting factor of its IFS is bigger. $DeterministicIteration$ function is responsible for doing deterministic iteration to any point using the weighted IFS and get its attractor. The last step is to get the predicted values we need using the attractor. All process is in $getPredictedValue$ function. First, it estimates the maximum and minimum load values according to the historical data, then get y_i at time x_i when you want to predict with the attractor line. Finally, the predicted load can be obtained by doing an inversion operation on Formula 6. If there are more than half of predicted loads that are lower than the predetermined threshold, our prediction model will send a notification to the scheduler that it is necessary for the running virtual machines to be reassigned to hosts for power-saving adjustment.

3.4. DESIGN OF SCHEDULER

The scheduler is based on the improved ant colony algorithm [24]. It has been proven to be a robust, versatile and population based approach which can be used to solve different combinatorial optimization problems. Meanwhile, the ant colony algorithm has been shown to compare favorably with other heuristic algorithms like genetic algorithms, evolutionary programming, and simulated annealing. We make use of the improved ant colony algorithm to allocate running virtual machines to those hosts that have a large capacity of resources while running on low operational power. In this way, we can shut down redundant hosts to reduce energy consumption and improve resource utilization.

To assign N_v virtual machines to N_h host machines, the scheduling algorithm of the scheduler is as follows:

Every artificial ant carries a merged queue called a A^i_MQ, which consists of virtual machines. The merged queue is made up of four queues of virtual machines and the process of $makeMergedQueue$ is described as follows:

We assume that the first queue consists of virtual machines that are running CPU-intensive jobs. The second queue consists of those virtual machines running I/O-intensive jobs. The third queue is made up of the virtual machines running both CPU-intensive and I/O-intensive jobs (hereafter, referred to as WEIGHT jobs), and the virtual machines of the last queue running those jobs whose demands for CPU and I/O are both low (hereafter, referred to as LIGHT jobs). All running virtual machines in the cluster are distributed in the four queues, and the virtual machines of each queue are arranged in accordance with the demand for resources, ranging from largest to smallest. It assumes that $a[m]$ stands for the first queue mentioned above, $b[n]$ represents the second queue, $c[p]$ is the third queue, while $d[q]$ is the last queue. The suffix parameters, m, n, p, q, are not necessarily equivalent. If $a[0]$ is bigger than $b[0]$, the sequence of the merged queue is $c[p_i]$ $(0 \leq p_i \leq p-1)$, $d[q_i]$ $(0 \leq q_i \leq q-1)$, $a[0]$, $b[0]$, $a[1]$, $b[1]$, ... If the virtual machines of the a or b queue are the first to be completed

Algorithm 2 Scheduling Algorithm

procedure INTIALIZATION
  for i from 0 to $N_a - 1$ step 1 **do**
    $A^i\_AL = \{VM^j\}(0 \le j < N_v)$
    $A^i\_HC = \varnothing$
    $A^i\_P = \varnothing$
    makeMergedQueue(A^i)
  end for
end procedure
procedure ANTTASK(x)
  while $|A^x\_MQ|! = 0$ **do**
    for i from 0 to $N_h - 1$ step 1 **do**
      calProbabilityForHost(i)
    end for
    if A^x select H^r $(0 \le r < N_h)$ **then**
      move H^r from A^x_AL to A^x_HC
      for j from 0 to $N_v - 1$ step 1 **do**
        if (VM^j is in A^x_MQ) and ($VM^j\_RR \le H^r\_Ca$) **then**
          add $\{VM^j, H^r\}$ to A^x_P
          delete VM^j from A^x_MQ
          $--|A^x\_MQ|$
        end if
      end for
    end if
  end while
end procedure
procedure IMPROVEDANTCOLONYALGORITHM
  for t from 1 to N_m step 1 **do**
    Intialization()
    for i from 0 to $N_a - 1$ step 1 **do**
      AntTask(i)
    end for
    UpdateAction()
    GetPlan()
  end for
end procedure

Table 2: Parameters Description

Description	Unit	Symbol
The number of ants	-	N_a
The number of hosts	-	N_h
The number of virtual machines	-	N_v
The number of iterations	-	N_m
The ith ant	-	A^i
The AL set of the ith ant	-	A^i_AL
The HC set of the ith ant	-	A^i_HC
The merged queue of the ith ant	-	A^i_MQ
The migration plan of the ith ant	-	A^i_P
The energy consumption produced by the ith ant's migration plan	kW·h	A^i_Co
The ith host	-	H^i
The logic resource capacity of the ith host	-	H^i_Ca
The pheromone of the ith host left by ants	-	H^i_Ph
The jth virtual machine	-	VM^j
The resource requirement of the ith ant	-	VM^j_RR

and added into the merged queue, the rest virtual machines of the other queue are added directly into the merged queue according to the established order.

In the 12th line of Algorithm 2, the ant calculates the probability for hosts as Formula(7):

$$P_j(t) = \begin{cases} \frac{E_j(t)^\alpha \times {F_j}^\beta}{\sum (E_k(t)^\alpha \times {F_k}^\beta)}, j, k \in \text{AL} \\ 0, otherwise \end{cases} \tag{7}$$

where P_j represents the probability that the ant chooses the jth host, E_j is the pheromone concentration of the jth host, F_j is the cost performance of the jth host, α stands for the importance of the pheromone concentration, and β represents the importance of cost performance. AL is the set of hosts that ants can continue to select. These hosts are defined as not falling within the set HC. Meanwhile, their logic resource capacity is larger than the resource demands of the virtual machine at the head of the merged queue. HC is a set of hosts that are already selected by the ants. When an ant selects the jth host, it should place this host in HC and place as many virtual machines as possible on this host.

In the 32nd line of Algorithm 2, the *UpdateAction* procedure updates the global pheromone concentration according to the following formulas:

$$E_j(t+1) = (1-\rho)E_j(t) + \Delta E_j \tag{8}$$

$$\Delta E_j = \sum_{k=1}^{M} \Delta E_j^k \tag{9}$$

$$\Delta E_j^k = \begin{cases} \frac{Q}{P_k}, if\ ant\ \text{k}\ passes\ the\ \text{j}th\ host\ during\ \text{t}\ to\ \text{t+1}\ iteration \\ 0, otherwise \end{cases} \quad (10)$$

ρ in Formula 8 represents the pheromone volatilization rate, and ΔE_j is the pheromone released by all the ants in the period of t iteration. If ant k passes the jth host during t to $t+1$ iteration, it will release $\frac{Q}{P_k}$ pheromone on this host. Otherwise, ant k will not release any pheromone on it. Q is a constant and P_k is the total real power of all hosts that ant k has passed. We assume that ant k has passed through U host machines and ${P_k}^i$ is the real power of the ith host that it has passed.

$$P_k = \sum_{i=1}^{U} P_k^i \quad (11)$$

$$P_k^i = (\frac{T}{L})Load_i + \frac{L-T}{L} \quad (12)$$

T in Formula(12) represents the rated power of the ith host machine, $Load_i$ is its real-time load and L is a coordination factor.

The *GetPlan* procedure selects the optimal scheduling in the loop which owns the minimal actual total power. If the actual total power of this optimal scheduling is smaller than the last cycle, then replace the final optimal scheduling with this one.

4. EXPERIMENTAL EVALUATION AND RESULTS

We have implemented our algorithm model and evaluated the quality of our solution using trace-driven simulation experiments. To make the simulation more targeted and reduce unnecessary complexity in the process of simulation, we developed a light and powerful simulator for cloud computing scheduling algorithms that references CloudSim [25]. However, it must be emphasized that we diminished the scale of both tasks and a data center because we use a "flat" operation entity model. Every task is bound to only one virtual machine, and the resource requirements of each task are set to equal the task usage. This means that a task does not reserve resources when it is applied to the resource provisioning model for resources. In order to reduce the number of operations in the entity model, we adopted some scale reduction measures that have no influence on performance comparisons. In the experiments, the data center is composed of 100 physical machines. We set the CPU capacity of each machine to 0.25, 0.5, 0.75, or 1. This represents the majority of the machines' configurations in the Google cluster. In our simulation experiments, we implemented the greedy First-Fit (FF) scheduling algorithm, the Round-Robin (RR) scheduling algorithm (which are used by some cloud computing platforms), and the Minimum Migration Power[13] (MM), which was proposed in the recent year. In our simulations, the rated power of each machine is generated randomly from 400 to 800 Watt and we set the running power of an idle machine to half of its rated power.

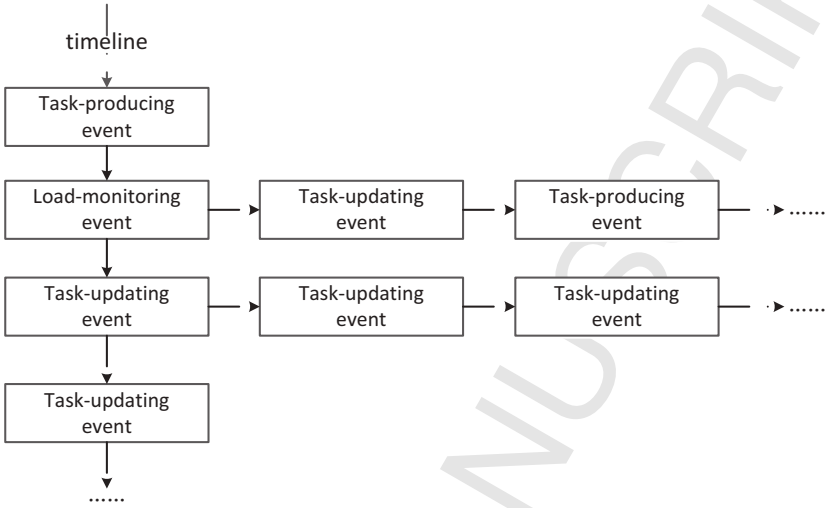

Figure 2: Simulation Core Process

4.1. Simulation Scenarios

Although we reduced the scale of the Google trace data without any distortion to simplify our simulation, many parameters still need to be decided. The simulation scenarios are as follows: the length of simulation is 24 hours. The low threshold of the CPU and disk utilization is set to 70%, and the high threshold is 90%. The system monitoring time interval is 20 minutes. In the improved ant colony algorithm, α is 2, β is 5, L is 2 and ρ is 0.5. The number of ants is 100 and the maximum iteration time is 100. The weighting factor of IFS is set to 0.4, 0.4, and 0.2. The weight coefficient for CPU capacity in logic resources capacity of each host machine is 0.7 and the disk weight coefficient is set to 0.3 accordingly. N_s is specified to 3, I to 20 and N_p to 5 in the fractal prediction model. We conduct our experiments with different combinations of multiple sets of parameters. Finally we decide to give the above combination of parameters because these values can fully display the functionality and performance of our algorithm. Meanwhile, the amount of resource consumed of this algorithm is acceptable under these values.

4.2. Simulation Target

First, we will prove that our prediction model, which is based on fractal mathematics is effective and reasonable. Then we will prove that the algorithm in this paper performs better than the other algorithms in reducing the power consumption of data centers and promoting the resource utilization of host machines. The First-Fit (FF) algorithm, the Round-Robin (RR) algorithm, and the Minimum Migration Power (MM) scheduling algorithm are used to compare with our algorithm.

4.3. Simulation Core Processes

The simulation core process is shown in Fig.2. At first, all the Task-producing and Load-monitoring events within the scope of the simulation interval are mounted to the timeline of the timescheduler according to their timestamp. Then, the timescheduler begins to deal with all kinds of events in order. During this period, with the implementation of Task-producing events, Task-updating events are generated and added to the timeline. The simulation experiment ends until the presupposed simulation time expires.

4.4. Simulation Test And Result Analysis

4.4.1. CPU-load prediction

To verify the validity of the fractal prediction model, CPU load is selected as the input data of this model. However, if the fractal prediction model can predict CPU load effectively, other resources with autocorrelation will be equally as effective.

Fig. 3 shows the comparison between the predictions of CPU usage which adopts fractal mathematics and the real usage. The graph shows that the predicted values are always close to the real ones even during peaks. We use Pearson product-moment correlation coefficient (PPMCC) to measure the trend similarity of real CPU load and predictive value of CPU load. The Pearson correlation coefficient is calculated as follows.

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}} \tag{13}$$

Having r close to 1 means the level of the positive correlation between X and Y is very high. We replace X with the predictions of CPU usage and replace Y with the real CPU usage. Finally, the value of r is very close to 1 ($r \approx 0.95$) which indicates that our fractal prediction model provides an accurate prediction of the load trend for the resources which have a very strong self-similarity and a long correlation.

4.4.2. Energy Consumption Test

Table 3: Energy Consumption Statistics

Index \ Algorithm	PreAntPolicy	FF	RR	MM
Average energy consumption(kwh)	360.54	1512.94	1458.66	923.90
Maximum energy consumption(kwh)	748.22	1790.48	1995.49	1306.06
Minimum energy consumption(kwh)	203.79	1291.93	1156.63	690.48
AVEDEV of energy consumption(kwh)	56.19	71.13	107.57	82.39

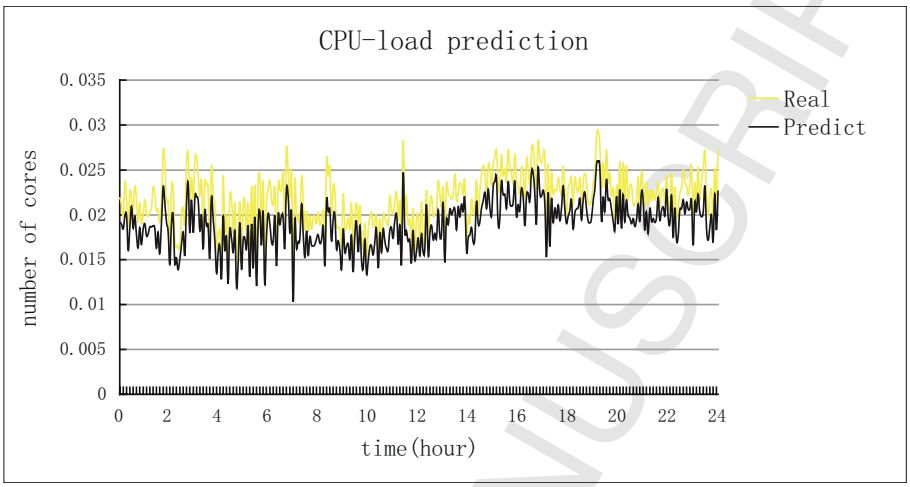

Figure 3: CPU-load prediction

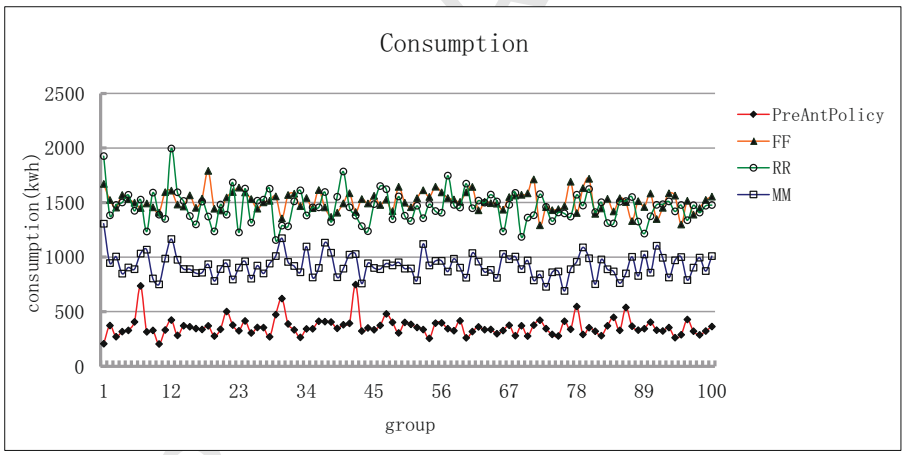

Figure 4: Energy Consumption Contrast Test

Fig. 4 depicts the case of energy consumption for four algorithms. A statistical table of test data is given at the same time. Table 3 shows that our algorithm, nameed *PreAntPolicy*, has the least average energy consumption during the experiment interval than FF, RR as well as MM algorithms. It represents 76.17%, 75.28%, and 60.98% improvement over FF, RR, and MM, respectively. Meanwhile, the maximum and minimum values of *PreAntPolicy* of the energy consumption test are far lesser than the other algorithms which means that *PreAntPolicy* has a very excellent energy saving effect and is highly controllable. The AVEDEV indicator proves that the fluctuation of energy consumption of *PreAntPolicy* is also satisfactory and acceptable. *PreAntPolicy*

Table 4: CPU Utilization Statistics

Index \ Algorithm	PreAntPolicy	FF	RR	MM
Average CPU utilization of one host	0.73	0.67	0.21	0.08
Maximum CPU utilization of one host	0.89	0.75	0.45	0.10
Minimum CPU utilization of one host	0.54	0.57	0.08	0.06
AVEDEV of CPU utilization of one host	0.04	0.03	0.06	0.01

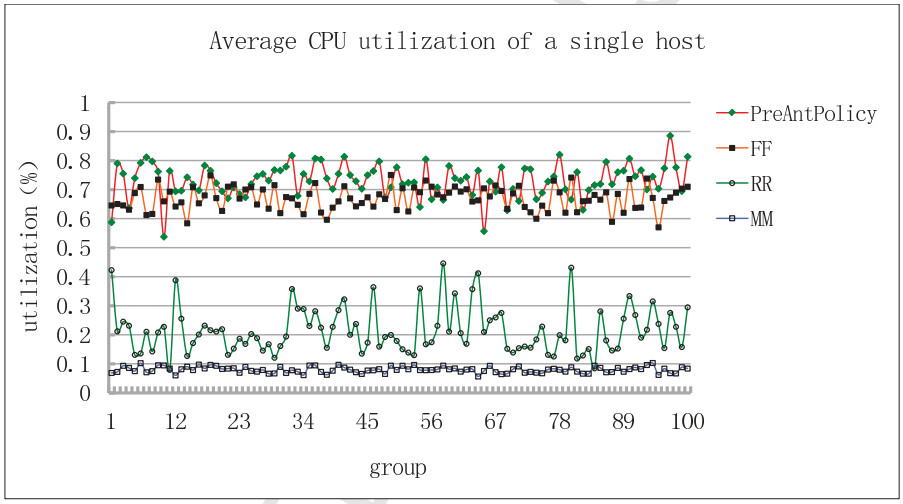

Figure 5: CPU Utilization Contrast Test

performs much better in energy consumption because it takes into account the impacts of heterogeneity in the computing environment, as well as different applications resource requirements. With the help of improved ant colony algorithm, *PreAntPolicy* makes virtual machines running on the hosts as cost-effective as possible, and achieving the combinatorial optimization of energy consumption and resource requirements. The prediction model based on fractal mathematics also helps the algorithms to reserve resources for the future load. However, FF , RR and MM algorithms only provide the assignment of running virtual machines in a fixed manner without taking any combinatorial optimization measures for energy consumption and resource requirements. Simultaneously, no prediction model is supplied for load prediction. At the same time, by the way of making virtual machines running on a small number of hosts which own high cost performance and turning off redundant machines, the average resource utilization of a physical machine is promoted.

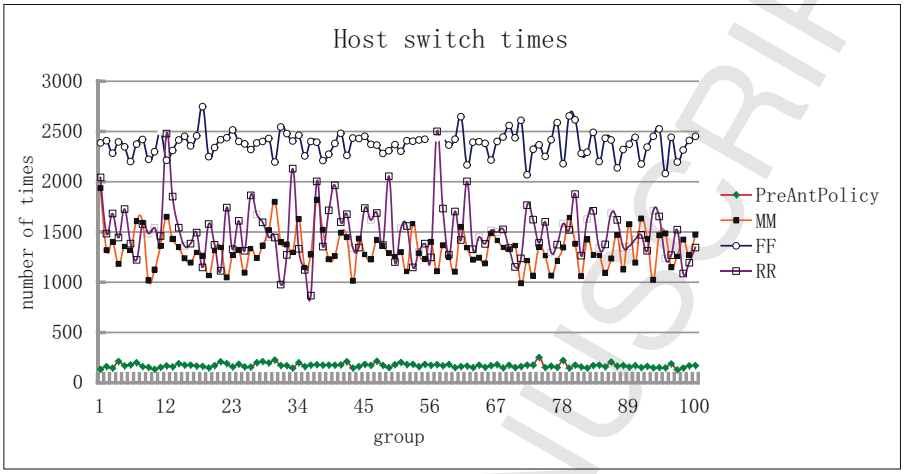

Figure 6: Physical Machine Switch Contrast Test

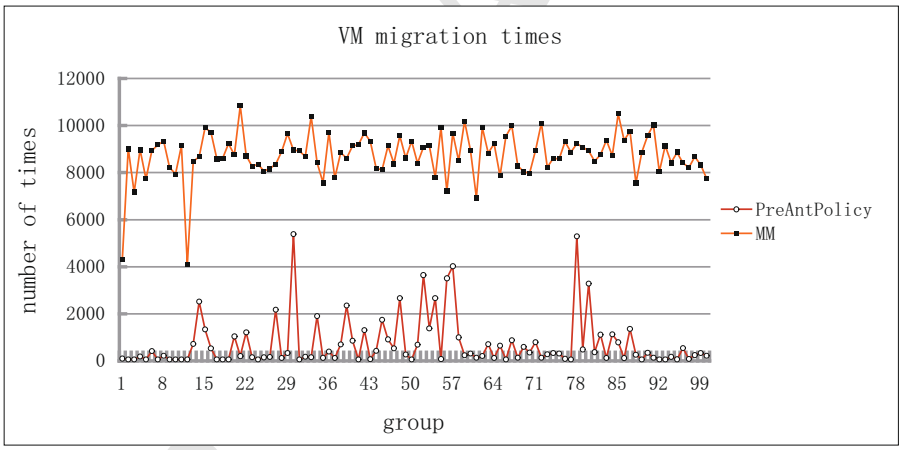

Figure 7: Virtual Machine Migration Times Contrast Test

4.4.3. CPU Utilization Test

Fig. 5 and Table 4 show that the average CPU utilization of *PreAntPolicy* increases by about 6%, 52% and 65% compared to FF, RR, and MM, respectively. Meanwhile, the maximum CPU utilization of *PreAntPolicy* can access 90% which is much better than other algorithms. The minimum CPU utilization of *PreAntPolicy* and AVEDEV is not the best among all the test algorithms. The main reason is FF algorithm places as many virtual machines as possible on the chosen host. It just puts the focus on resource capacity and and its lack of the sight of energy consumption. However, *PreAntPolicy* is more holistic.

Table 5: Physical machine (PM) Switch Times Statistics

Index \ Algorithm	PreAntPolicy	FF	RR	MM
Average PM switch times	169.44	2378.82	1506.29	1326.72
Maximum PM switch times	250.00	2750.00	2501.00	1937.00
Minimum PM switch times	127.00	2070.00	866.00	989.00
AVEDEV of PM switch times	16.00	89.97	210.48	138.06

4.4.4. PM Switch Times Test

Fig. 6 and Table 5 show that *PreAntPolicy* has obvious advantage in the average host switch times against other test algorithms. It represents a decrease of 1157.28, 2209.38, and 1336.85 times over MM, FF, and RR respectively. Using load trend prediction model based on the fractal mathematics, our algorithm can reserve resources for applications intelligently. The difference between our prediction model and others is that our model can predict the load trend but not just a single future load through historic dataset. Through this strategy, error scheduling caused by an instantaneous peak load can be prevented and unnecessary host switches can be avoided.

4.4.5. VM Migration Times Test

Because the FF and RR algorithms do not migrate virtual machines, we need only compare *PreAntPolicy* with MM. From Fig.7 and Table 6 we can find that the average migration time of *PreAntPolicy* is 724.60 and the MM is 8741.46. It means that *PreAntPolicy* has a huge advantage with an average of 8016.86 times less migration compared to the MM. The maximum and minimum values of migration times are far less than MM simultaneously. *PreAntPolicy* reduces the migration number of virtual machines effectively because *PreAntPolicy* considers energy consumption from a global perspective and takes into account the impacts of different applications resource requirements. After every migration triggered by the scheduler of *PreAntPolicy*, all virtual machines run on cost-effective hosts. A cluster is in the energy-saving state right now in which case virtual machines do not need frequent migrations in a short time. However, the MM algorithm just put focus on energy consumption caused by migration. Meanwhile, it does not take the heterogeneous of tasks into consideration. Under this conditions, MM's scheduler is triggered frequently which leads to frequent migration of virtual machines.

5. CONCLUSION

In this paper, we propose an energy-aware scheduling for virtual machines in heterogeneous cloud computing systems. The scheduling approach is mainly composed of a prediction model and a scheduler. Simulation experiments using

Table 6: VM Migration Times Statistics

Index \ Algorithm	PreAntPolicy	MM
Average VM migration times	724.60	8741.46
Maximum VM migration times	5388.00	10842.00
Minimum VM migration times	55.00	4092.00
AVEDEV of VM migration times	733.46	673.69

real traces obtained from production compute clusters of Google were conducted to investigate the energy efficiency of the approach and the experimental results showed that the proposed approach can reduce energy consumption by up to 76.17% compared to first-fit and 75.28% for round-robin, while the QoS has not engaged any significant decline. The PPMCC(0.95) calculated within our test case has shown the validation of the prediction model which assists our scheduler significantly in reducing the number of virtual machine migrations and the number of physical machine switch. Moreover, the scheduler based on the improved ant colony algorithm supports multiple types of resource scheduling which meets the requirements of resource-intensive applications in a real world scenario.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Communications of the ACM 53 (4) (2010) 50–58.

[2] Epa datacenter report congress, `http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final.pdf`.

[3] Microsoft environment-the green grid consortium, `http://www.microsoft.com/environment/our_commitment/articles/green_grid.aspx`.

[4] C.-H. Hsu, W.-c. Feng, A feasibility analysis of power awareness in commodity-based high-performance clusters, in: Cluster Computing, 2005. IEEE International, IEEE, 2005, pp. 1–10.

[5] J. Sahoo, S. Mohapatra, R. Lath, Virtualization: A survey on concepts, taxonomy and associated security issues, in: Computer and Network Technology (ICCNT), 2010 Second International Conference on, IEEE, 2010, pp. 222–226.

[6] L. Wang, S. U. Khan, D. Chen, J. Kolodziej, R. Ranjan, C.-z. Xu, A. Zomaya, Energy-aware parallel task scheduling in a cluster, Future Generation Computer Systems 29 (7) (2013) 1661–1670.

[7] S. Hosseinimotlagh, F. Khunjush, S. Hosseinimotlagh, A cooperative two-tier energy-aware scheduling for real-time tasks in computing clouds, in: Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on, IEEE, 2014, pp. 178–182.

[8] E. N. Watanabe, P. P. Campos, K. R. Braghetto, D. M. Batista, Energy saving algorithms for workflow scheduling in cloud computing, in: Computer Networks and Distributed Systems (SBRC), 2014 Brazilian Symposium on, IEEE, 2014, pp. 9–16.

[9] K. Wu, R. Du, L. Chen, S. Yan, An energy-saving virtual-machine scheduling algorithm of cloud computing system, in: Information Science and Cloud Computing Companion (ISCC-C), 2013 International Conference on, IEEE, 2013, pp. 219–224.

[10] R. Nathuji, K. Schwan, Virtualpower: coordinated power management in virtualized enterprise systems, in: ACM SIGOPS Operating Systems Review, Vol. 41, ACM, 2007, pp. 265–278.

[11] N. Liu, Z. Dong, R. Rojas-Cessa, Task scheduling and server provisioning for energy-efficient cloud-computing data centers, in: Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on, IEEE, 2013, pp. 226–231.

[12] J. Stoess, C. Lang, F. Bellosa, Energy management for hypervisor-based virtual machines., in: USENIX Annual Technical Conference, 2007, pp. 1–14.

[13] M. M. Hossain, J.-C. Huang, H.-H. S. Lee, Migration energy-aware workload consolidation in enterprise clouds., in: CloudCom, 2012, pp. 405–410.

[14] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of the 2008 conference on Power aware computing and systems, Vol. 10, San Diego, California, 2008.

[15] R. Buyya, A. Beloglazov, J. Abawajy, Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges, arXiv preprint arXiv:1006.0308.

[16] S. Niu, J. Zhai, X. Ma, X. Tang, W. Chen, Cost-effective cloud hpc resource provisioning by building semi-elastic virtual clusters, in: Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2013, p. 56.

[17] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, A. Y. Zomaya, Energy-efficient data replication in cloud computing datacenters, in: Globecom Workshops (GC Wkshps), 2013 IEEE, IEEE, 2013, pp. 446–451.

[18] Y. Gao, S. K. Gupta, Y. Wang, M. Pedram, An energy-aware fault tolerant scheduling framework for soft error resilient cloud computing systems, in: Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, IEEE, 2014, pp. 1–6.

[19] T. H. Szymanski, Low latency energy efficient communications in global-scale cloud computing systems, in: Proceedings of the 2013 workshop on Energy efficient high performance parallel and distributed computing, ACM, 2013, pp. 13–22.

[20] F. Cao, M. M. Zhu, Energy efficient workflow job scheduling for green cloud, in: Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, IEEE Computer Society, 2013, pp. 2218–2221.

[21] K. Falconer, Fractal geometry: mathematical foundations and applications, John Wiley & Sons, 2013.

[22] P. A. Dinda, The statistical properties of host load, Scientific Programming 7 (3) (1999) 211–229.

[23] M. F. Barnsley, A. N. Harrington, The calculus of fractal interpolation functions, Journal of Approximation Theory 57 (1) (1989) 14–34.

[24] V. Maniezzo, M. Dorigo, A. Colorni, The ant system: Optimization by a colony of cooperating agents, IEEE Trans. Sys. Man Cybernet (Part B) (1996) 29–41.

[25] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience 41 (1) (2011) 23–50.

Hancong Duan (duanhancong@uestc.edu.cn) received the B.S. degree in computer science from Southwest Jiaotong University in 1995, the M.E. degree in computer architecture in 2005, and the Ph.D. degree in computer system architecture from University of Electronic Science and Technology of China in 2007. He is currently a professor of computer science at the University of Electronic Science and Technology of China. His current research interests include large-scale P2P content delivery network, distributed storage, and cloud computing.

Chao Chen (cqwzchenchao@163.com) received the B.S. degree in computer science and technology (2015) from University of Electronic Science and Technology of China. Currently he is studying toward his M.S. degree at University of Electronic Science and Technology of China and engaged in research and development of distributed network. And He is also interested in cloud computing and distributed storage.

Geyong Min (G.Min@exeter.ac.uk) received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. He is a Professor of High Performance Computing and Networking in the Department of Mathematics and Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, UK. His research interests include Future Internet, Computer Networks, Wireless Communications, Multimedia Systems, Information Security, High Performance Computing, Ubiquitous Computing, Modelling and Performance Engineering.

Yu Wu (ThinkingYu2015@163.com) received the B.S. degree in computer science and technology (2013) from University of Electronic Science and Technology of China. Currently he is studying toward his M.S. degree at University of Electronic Science and Technology of China and engaged in research and development of distributed network. And He is also interested in cloud computing and cloud storage.