



Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa



Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers



Jian-ping Luo*, Xia Li, Min-rong Chen

College of Information Engineering, Shenzhen University, Shenzhen 518060, PR China

ARTICLE INFO

Keywords:

Cloud computing
Data center
Resource management
Intelligent algorithm

ABSTRACT

Cloud computing aims to provide dynamic leasing of server capabilities as scalable virtualized services to end users. However, data centers hosting cloud applications consume vast amounts of electrical energy, thereby contributing to high operational costs and carbon footprints. Green cloud computing solutions that can not only minimize the operational costs but also reduce the environmental impact are necessary. This study focuses on the Infrastructure as a Service model, where custom virtual machines (VMs) are launched in appropriate servers available in a data center. A complete data center resource management scheme is presented in this paper. The scheme can not only ensure user quality of service (through service level agreements) but can also achieve maximum energy saving and green computing goals. Considering that the data center host is usually tens of thousands in size and that using an exact algorithm to solve the resource allocation problem is difficult, the modified shuffled frog leaping algorithm and improved extremal optimization are employed in this study to solve the dynamic allocation problem of VMs. Experimental results demonstrate that the proposed resource management scheme exhibits excellent performance in green cloud computing.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Cloud computing is a cost-effective service delivery model that makes IT management and maintenance easy; it can rapidly adapt to changing business needs. Cloud computing can be roughly divided into three categories in accordance with the service type, namely, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) (Gandhi, Harchol Balter, Das, & Lefurgy, 2009). Users can easily avail of these services on a pay-as-you-use basis without any geographical restrictions. Cloud computing is a convenient, necessary, and shared network computing resource with a dynamic configuration (including network, service, storage, and application); it can be managed easily and effectively by the resource manager at minimal management costs (Gandhi et al., 2009). Cloud service providers need to consider the convenience of cloud services without placing too much emphasis on the hardware facilities to extend the computing potential of a computing center. Thus, different cloud users can enjoy cloud computing capabilities easily and efficiently.

The core of the cloud computing environment is the cloud data center, which often consists of a number of highly configured hard-

ware facilities. The computing capability of the data center is the main indicator considered by cloud service providers. With the appearance of increasingly large data centers, the energy consumption of data centers increases as well. This increase in energy consumption affects the environment. For instance, carbon dioxide emissions cause the greenhouse effect and have a serious impact on climate and the environment. These effects will eventually affect the operational benefits of a cloud service provider. Statistics show that the energy consumption of a common data center is approximately equal to that of 25,000 ordinary households (Kaplan, Forest, & Kindler, 2008). The energy consumption of data centers in the United States from 2008 to 2010 was supplied by 10 nuclear power stations. Clearly, more data centers cause higher energy consumption. However, the energy consumption of data centers has rarely elicited attention. Therefore, the computing capacity performance of data centers and the problem of huge energy consumption by these centers must be considered in addressing the resource management optimization issue of data centers. Green computing is the only method to enhance the operational efficiency of data centers and reduce damage to the environment. The designers of these centers should consider the efficient use of computing resources on the premise that the system maintains excellent computing service capacity. The hosts of data centers in idle or low utilization status also consume vast amounts of energy. For instance, the energy consumption of a

* Corresponding author. Tel.: +86 15889669910.
E-mail address: camelrock@126.com (J.-p. Luo).

non-operative server (not turned off) accounts for approximately 70% of the full load energy consumption (Kusic, Kephart, Hanson, Kandasamy, & Jiang, 2009). Similar conclusions can also be found in the literature (Fernández-Montes, Gonzalez-Abril, Ortega, & Lefèvre, 2012).

In reality, hardware facilities in cloud data centers do not remain static for a long time. The state of most hardware facilities often change. First, adding a new server to the system causes the reconfiguration, restoration, or replacement of the existing server. Second, the resource pool often changes the operating status to satisfy the requirement of intermittent resource changes. Third, live migration causes virtual machines (VMs) to achieve rapid reconfiguration and consolidation in different physical nodes to achieve goals, such as load balancing. Fourth, some servers need to transfer VMs to the appropriate servers; shutdown, maintenance, and restart are operated after the live migration, thereby causing some servers to become unavailable. Similarly, some servers need to be temporarily open to address unpredictable access peak outbursts. Access to a server involves a certain degree of uncertainty. In addition, changes could occur within each server, such as changes in processing elements (PEs), memory size, hard disk storage, and bandwidth. Current servers generally support dynamic voltage frequency scaling (DVFS). In DVFS, the server can adjust the operating frequency based on the current load to achieve energy saving. Adopting a dynamic regulatory mechanism is necessary for the dynamics and uncertainty of the resource. Determining the transition state of the source, placing the new applied VMs reasonably, and optimizing the allocation of VMs that violate the service level agreements (SLAs) or VMs in the server with very low CPU utilization are also necessary. These measures are essential to ensure that the entire configuration of the VMs is as optimal as possible. A dynamic regulatory mechanism can be utilized to achieve automatic and dynamic management without managerial or staff intervention. In a cloud environment, the data center is usually presented in an over-provisioned state to meet the uncertain resource application peak, thereby resulting in large amounts of energy waste.

A complete data center resource management scheme is proposed for the Infrastructure as a Service (IaaS) cloud environment in this paper. The proposed scheme can not only guarantee user quality of service (QoS) specified by SLAs but can also achieve maximum energy saving and green computing goals. The main contribution of this study is the proposal of the complete dynamic resource management scheme. Consolidation of resources is achieved by VM migrations technology and low-utilized or idle hosts switched to power saving mode to achieve energy saving while ensuring that SLAs are adhered to. The intelligent method of modified shuffled frog leaping algorithm (SFLA) based on improved extremal optimization (EO) is applied to rapidly and efficiently complete the dynamic allocation of VMs.

This paper is organized as follows. Section 2 provides a brief review of related research on energy conservation and dynamic VM allocation technologies. Section 3 discusses the intelligent resource management scheduling framework. An intelligent hybrid algorithm for the dynamic VM consolidation problem is proposed in Section 4. Section 5 presents the experimental evaluations and result discussions. Section 6 provides the conclusion and suggestions for future work.

2. Related work

DVFS is a means of achieving hardware facilities energy conservation (Chase, Anderson, Thakar, Vahdat, & Doyle, 2001). DVFS, which is based on the different needs of the computing capacity by application program, dynamically adjusts the running frequency and voltage of the chip (for the same chip, the higher the

frequency, the higher the voltage) to achieve energy saving. A low frequency can reduce power. However, energy cannot be saved by simply reducing the frequency because for a given task, energy consumption can only be reduced when the voltage and frequency are lowered simultaneously. The implementation of DVFS depends on the successful prediction of the number and execution time of processing tasks in the server. Clock frequency and voltage do not have a linear relation in a real-time system. Much uncertainty exists among task execution time, energy consumption, and processor voltage. Inappropriate frequent voltage adjustment will degrade processor performance. Predicting the number of tasks is difficult in most cloud environments.

DVFS generally requires power management through BIOS. Circuits designed by different manufacturers differ significantly. Intel, Microsoft, Toshiba, and other companies jointly developed the advanced configuration power interface (ACPI) specification to establish a common power management interface between the operating system and the hardware facilities (Venkatachalam & Franz, 2005). ACPI improves the original APM through BIOS and provides a relatively good power management mode and interface specification in configuration management. ACPI sets a maximum of six power states. Different states correspond to the different power consumptions of the processor, memory, and hard disk. Most processors at present support the states running, idling, sleeping, and closed.

Rusu, Ferreira, Scordino, and Watson (2006) proposed an energy consumption management strategy based on QoS in connection with the server cluster system. The strategy is divided into back-end management and local management. Local management supports DVFS. When the back-end manager detects that the system needs to close or open a server, the local manager controls power by the DVFS module and switches the server into the corresponding state. The system does not utilize live VM migrations technology; it involves the off-line calculation of the back-end server to decide whether to shut down or open the server. Such decision is limited for energy saving.

Lee and Zomaya (2010) proposed an efficient energy management strategy in a distributed cloud computing system. The researchers defined the optimized objective function as a relative superiority (RS) expression according to the relation between task processing time and energy consumption. The RS value for the assigned task is calculated first. The task is then allocated to the server with maximum RS value. This algorithm assumes that all servers are active and in good running condition and ignores the heterogeneity of the system. Only the allocation of the newly added VMs was considered in this study. In the actual process, the adjustment of VMs with SLA violation should be considered as well.

Kusic et al. (2009) defined the problem of energy management in virtual heterogeneous environments as a scheduling optimization problem and applied limited look-ahead control (LLC) during processing. LLC aims to maximize the profit of the resource service providers and minimize energy consumption and SLA violation. The system applies the Kalman filter to estimate the number of future client requests and predict the state of the system to achieve necessary resource integration. However, the system cannot be implemented in the IaaS cloud environment. In addition, the model is extremely complicated. Adjusting a system with 15 nodes requires 30 min. Thus, the model is difficult to apply to large-scale and real-time cloud data centers.

Verma, Ahuja, and Neogi (2008) modeled the energy-aware dynamic arrangements of VMs in the virtual heterogeneous environment and turned them into a continuous optimization problem; the placement of VMs was regarded as the minimum energy consumption and the maximum performance optimization issue in each time frame. The researchers utilized a heuristic algorithm to solve the packing problem and re-allocated the VMs of each time frame by live migration strategy. In their follow-up work

(Verma, Dasgupta, Nayak, De, & Kothari, 2009), the researchers applied static strategy (month and year adjustment), semi-static strategy (day and week adjustment), and dynamic strategy (minute and hour adjustment) to regular adjustment. However, these algorithms do not consider the issue of SLA violation. The system performance diminishes when the load changes, and SLAs cannot be guaranteed.

Berral et al. (2010) studied the VM dynamic consolidation problem on the premise of SLAs by applying machine learning techniques to address the issue of energy consumption. Similarly, Rodero, Viswanathan, and Lee (2012) employed the CPU dynamic voltage scaling technique and virtual machine dynamic integration technology to achieve the energy saving target of a data center. However, these research methods consider only applications in certain occasions, such as high-performance computing and applications with deadline constraints. Kramer, Petrucci, and Subramanian (2012) adopted the column generation technique and server power-saving technology based on CPU dynamic on/off switching and developed an energy-saving scheme for heterogeneous virtual server cluster environments. The scheme provided better results in a relatively short period of time. However, the experiment did not consider the large-scale cluster network environment. Beloglazov, Abawajy, and Buyya (2012) applied MBFD to solve the dynamic allocation issue of VMs. However, the algorithm cannot achieve the dynamic consolidation of VMs with the desired maximum energy conservation.

Virtualization is applied in data centers at present. The systems of these centers allow VMs to perform live migration among physical nodes to achieve improved performance or energy saving. When the resources that VMs use are less than those actually assigned, the VMs can migrate to other server nodes by adjustment and combination. The idle server node would switch to energy-saving mode via the ACPI interface to save energy. The current resource scheduling strategy of cloud data centers focuses on system performance improvement and ensures SLAs. The present study aims to maintain SLAs and save energy. The following issues are addressed. First, excessive energy consumption adjustment of each server reduces the stability of the server. Second, frequently shutting down the server leads to the result that QoS cannot be guaranteed in a dynamic environment. Some VMs cannot obtain sufficient resources during the access peak period because of migration and consolidation. Therefore, the requirements of QoS are not met. Third, in a virtual environment of enlarged numbers host, using an ordinary scheduling algorithm to ensure rapid and effective resource management scheduling is difficult. Ensuring application performance and guaranteeing SLAs pose challenges as well. An effective resource consolidation strategy should achieve energy conservation and green computing as well as address user-specified QoS requirements.

3. Dynamic resource management based on VM consolidation

Most of the existing energy saving technologies at present only consider energy saving and ignore SLA violation issues. Moreover, current research focuses on the allocation of newly added VMs and not on the operational status of the allocated VMs and corresponding server load status.

Compared with existing research, the present study addresses the live dynamic allocation problems of VMs with a scheduling framework based on an intelligent algorithm. The framework applies live migration strategy based on current resource utilization and switches some free resource nodes into power sleeping mode to save energy. These steps can ensure SLAs and meet the QoS requirements of heterogeneous underlying facilities and requests of different types of VMs.

3.1. Scheduling framework

An intelligent dynamic resource management framework based on the IaaS cloud environment is proposed in this paper. As illustrated in Fig. 1, when the user applies VM resources to the cloud system and specifies SLAs, the scheduling framework allocates the applied resources to the appropriate host based on the applied requirements and operating condition of host. The placement of VM must be completed quickly to ensure response speed. The scheduling framework should implement real-time monitoring on the hardware's (host) running conditions and perform dynamic consolidation of VMs based on the host's running condition. Dynamic consolidation with the server is mainly based on two conditions. First, for hosts that cannot meet SLAs, reducing the upper level of the threshold value of processor utilization, transferring some VMs to other hosts to reduce the load, and ensuring that SLAs are met are necessary. Second, for hosts whose utilization load is below a certain low limit, transferring all VMs on this host to another host and switching the host into sleeping state for energy saving are necessary. This dynamic adjustment process should be performed within a certain period of time. VM Scheduling problem is essential for VMs that need to be integrated, and it can be seen as a bin packing optimization problem with variable bin sizes and prices. To solve it we apply the modified SFLA intelligent method based on EO. Unlike the exact method or traditional heuristic algorithm, this method avoids inefficiency and long computation time. The SLAs of the VMs should be guaranteed during the optimization process with minimum energy consumption. The number of VMs that migrate should not be too large because the frequent live migration of VMs affects service performance.

3.2. Scheduling strategy for the newly applied VM

Users can apply new services (VMs) to the data center any time. After reaching SLAs, the cloud data center immediately assigns VMs to the appropriate host. With the proposed resource management scheduling scheme, the newly applied VMs are assigned to the host that can meet SLA requirements and save energy. Allocation is performed with the algorithm while traversing the host list only once, including the advantage of fast response, as shown in Fig. 2.

3.3. Dynamic resource adjustment of the time window

The hardware and software of the host in the cloud data center are in a constant changing environment. The VM service is also in a

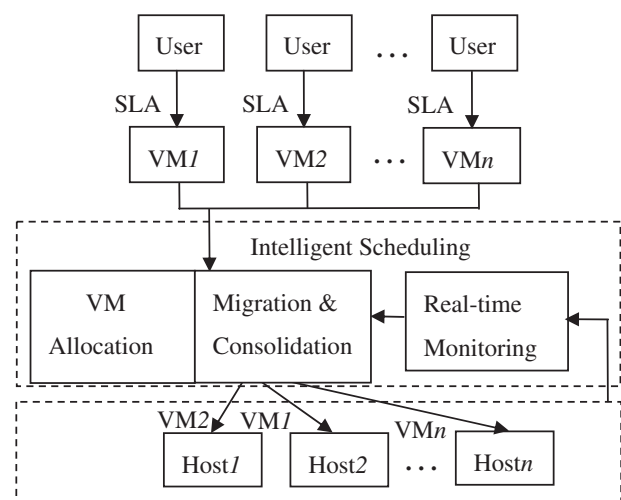


Fig. 1. Intelligent resource management scheduling framework.

changing state. The scheduling framework monitors the host's running condition and performs the dynamic consolidation of VMs based on the running state of the host within a certain period of time. Fig. 3 shows the specific adjustment strategy.

During the dynamic adjustment of each time window, VMs are selected and consolidated from the hosts with SLA violation or low load according to the adjustment strategy to reduce the number of migration. For the selection of hosts with SLA violation, the server service level is matched to the customized SLAs based on Eq. (1) by dynamically adjusting the upper threshold of processor utilization:

$$new_upper_th \propto \frac{R_{SLA}}{r_{SLA}} upper_th. \quad (1)$$

In the equation above, $upper_th$ is the current upper threshold of processor utilization, R_{SLA} is the goal SLAs of the host, and r_{SLA} is the current SLAs of the host. The new threshold of processor utilization is estimated as the arithmetic product of the current threshold and the ratio of the targeted and current SLAs. The new threshold will replace the old one. The $upper_th$ value cannot be decreased or increased limitlessly. If the $upper_th$ value is too large, e.g., approaches 1, user QoS may not be guaranteed because the consolidation of VMs will not occur even when processor utilization is close to 100%; this condition means that the host is overloaded. On the other hand, the $upper_th$ value should not be too small because it would cause high energy consumption. In this study, $upper_th \in [0.5, 0.95]$ was used. The next step is to evaluate the SLA violation of the current host and determine whether it meets the specified SLA requirements. If not, some VMs will be selected and added into the migration list according to Eq. (2). These VMs are ready to migrate from the host:

$$vms = \left\{ V | V \subset hvs, \sum_{h \in hvs} Utlz(h) - \sum_{v \in V} Utlz(v) < upper_th, |V| \rightarrow \min \right\}. \quad (2)$$

In Eq. (2), hvs is the overall VMs on the current host; $\sum_{h \in hvs} Utlz(h)$ is the overall utilization level of the current host processor, and $Utlz(v)$ is the average utilization of VM v within the time window occupying the current processor. VMs that need to be migrated were selected in this study by applying the strategy to minimize number of VM migrations. This strategy aims to reduce system performance degradation caused by the migration of VMs. When the sum of all the processor utilization of the remaining VMs is less than the pre-determined threshold after VM migrations, the specified requirements of SLAs are met. The specific scheduling strategy is shown in Fig. 4.

As shown in Fig. 5, the system transfers all VMs below the lower threshold of processor utilization to decrease the energy consumption. After these VMs are transferred, the host is switched into power-saving sleeping mode.

3.4. Energy consumption estimation of the host

Host energy consumption in the cloud data center is mainly caused by the consumption of the CPU, memory, disk and refrigeration systems, and other hardware modules (Minas & Ellison, 2009). With the increasing popularity of multi-core CPUs, the energy consumption of these CPUs has become a major component of energy consumption. According to Kusic et al. (2009) and Fan, Weber, and Barroso (2007), host energy consumption exhibits an almost linear proportion to CPU energy consumption. Moreover, the energy consumption of an idle host accounts for 70% of

```

BOOL AllocateVmToHost (vm, HostTable)
{
    allocatedHost = NULL;
    minEnergy = MAX;
    // Traverse all the Server Hosts
    for (int i=0; i<len(HostTable); i++){
        host = GetHostFromHostTable(i);
        // Estimate the energy consumption when VM is assigned to specified host
        energy = EstimateEnergyAfterVMToHost (host,vm)
            - GetHostEnergy(host);
        // Meet SLAs requirement and have the minimum energy consumption
        if( HostMeetSLA(host,vm)
            && energy < minEnergy){
            allocatedHost = host;
            minEnergy = energy;
        }
    }
    if ( allocatedHost != NULL ){
        // VM is assigned to the most appropriate server host
        DispatchVmToHost(allocatedHost,vm);
        Return TRUE;
    }
    else
        Return FALSE;
}

```

Fig. 2. Scheduling strategy for the newly applied VM.


```

// Carry out dynamic Scheduling with certain periodic frequency
DynamicScheduling()
{
    vmMigrateTable = NULL;
    for(host: HostTable){
        // Choose VMs from the host of SLA violation and add into the migration
        list
        vms = GetVmToMigrateByViolateSLAs(host);
        AddVmsTovmMigrateTable(vms, vmMigrateTable);
        // Add all VMs of the low-load host into the migration list
        vms = GetVmToMigrateByLowLoad(host);
        AddVmsTovmMigrateTable(vms, vmMigrateTable);
    }
    // Apply the intelligent algorithm to find the appropriate hosts for the VMs
    which need migration and carry out live migration
    IntelligentAlgorithmToDispatch(HostTable, vmMigrateTable);
}

```

Fig. 3. Dynamic adjustment of the time window in the cloud data center.

```

vms GetVmToMigrateByViolateSLAs(host)
{
    vms = NULL;
    // By equation (1), adjust the upper threshold of host utilization
    AdjustThresholdByEquation(1);
    if( GetHostSLA(host) > SLAs){
        // By equation (2), choose VMs which need to be migrated
        vms = GetVmByEquation(2);
    }
    Return vms;
}

```

Fig. 4. VM migrations selection strategy in the SLA violation host.

```

vms GetVmToMigrateByLowLoad(host)
{
    vms = NULL;
    if( GetCPUUtilization(host) < low_threshold)
        vms = GetVmFromHost(host);
    Return vms;
}

```

Fig. 5. VMs of the host with low utilization added into the migration list.

full-load operation energy consumption. The energy consumed by VM migrations also requires consideration. Live VM migrations technology allows fast and flexible reset of VMs in hosts (Beloglazov & Buyya, 2011).

In live migration technology, the image file and real-time data of VMs are stored in network attached storage. Therefore, the VM memory and not the VM itself is copied during migration. The migration time of VMs is equal to the quotient of memory size and network bandwidth. However, the live migration of VMs can

still adversely affect the running services in the VMs. Voorsluys, Broberg, Venugopal and Buyya (2009) discussed the impact of live migration technology on system performance and performed mathematical modeling, which revealed that the loss of performance is mainly caused by migration time and the updated number on the service memory pages of VMs. For regular application services such as Web servers, the live migration process consumes approximately 10% of CPU utilization. Therefore, energy consumption within unit time can be defined in Eq. (3) as follows:

$$E(h) = 0.7E_{\max}(h) + 0.3Utlz(h)E_{\max}(h) + 0.1E_{\max} \sum_{i \in v} T(i). \quad (3)$$

In the equation above, $E_{\max}(h)$ is the energy consumption when host h is in full load. $Utlz(h)$ is the average utilization rate of the host processor within unit time, v is the collection of VM migrations within the unit time window, and $T(i)$ is the migration time of VM i .

3.5. SLA estimation

Achieving QoS, which is often presented by SLAs, is an important requirement for cloud computing systems. SLAs specify the system computing ability, reaction time, and maximum visit capacity, which differ in accordance with different application services. Based on Beloglazov and Buyya (2011), SLAs were evaluated in this study by defining SLA violation (SLAV) within a certain time window as follows:

$$SLAV = \left(\frac{T_v}{T_a} \right) \left(\frac{C_v}{C_a} \right). \quad (4)$$

The first item in Eq. (4) is SLA violation by the overload operation of the host processor. In this item, T_v is the time length of 100% utilization of the host within the time window and T_a is the size of the time window. One hundred percent host utilization means that the host is overloaded. An SLA violation then occurs because responding timely to the new access application is difficult when the host is overloaded. The second item in the equation above is SLA violation caused by performance degradation by live migration. In this item, C_v is the MIPS consumed by migration, which accounts for 10% of the total processor resources in the live migration process. C_a is the applied general MIPS by VMs.

4. VM scheduling based on an intelligent algorithm

Dynamic resource adjustment is implemented within a certain time period. It needs to determine the optimal scheduling scheme for all VMs needed to be allocated in the global scope. Much computation time is required in the calculation with the exact algorithm. Improved EO was applied to modified SFLA (MSFLA) in this study to solve the optimization problem.

4.1. Basic working principle of SFLA

SFLA is an intelligent algorithm inspired by natural organism imitation (Eusuff & Lansey, 2003). This algorithm is based on two search modes, namely, individual meme evolution, which is regarded as the meme carrier in the ethnic group, and global search for information exchange in the entire meme population. SFLA functions by randomly generating a frog population with F number ($P = \{X_1, X_2, \dots, X_F\}$). To solve the t dimension problem, the position of frog i is set to $X_i = [x_{i1}, x_{i2}, \dots, x_{it}]^T$. After generating a frog population, the fitness value $f(X_i)$ of each frog position is calculated and arranged from the maximum to the minimum. The frogs are then allocated into m groups with Eq. (5). Each group has n frogs, and the equation is $F = m \times n$:

$$M_i = \{X_{i+m(l-1)} \in P | 1 \leq l \leq n\}, \quad 1 \leq i \leq m, \quad (5)$$

where M_i is the i th group. A local search is performed in each group. The last frog with the lowest fitness value in each group is updated with Eqs. (6) and (7):

$$D_{i,w}(k) = r(X_{i,s} - X_{i,w}(k)), \quad (6)$$

$$X_{i,w}(k+1) = X_{i,w}(k) + D_{i,w}(k), \quad (7)$$

where $X_{i,w}(k)$ is the frog position with the worst fitness value at the k th iteration of the i th group; $X_{i,s}$ is the best frog position of the

group; r is a random number with $r \in [0, 1]$; and $D_{i,w}(k)$ is the moving distance for the k th iteration of the worst frog. If the fitness value of the new frog position is better than the original fitness value after the update, the new position will replace the old one. Otherwise, the moving distance will be updated with Eq. (8):

$$D_{i,w}(k) = r(X_b - X_{i,w}(k)), \quad (8)$$

where X_b is the best frog position in the entire population. If no improvement is observed after the update, random solution $X_{i,w}$ will be implemented. This operation is repeated in every group until the specified iterations are reached. After the depth search in every group, the entire frog population is re-mixed and sorted. The best frog position is recorded based on the fitness value. The population is then re-divided, and local depth search is performed again. The search ends when the termination condition is satisfied.

4.2. Improvement of SFLA by dynamic adjustment of leaping vision

Frog foraging in SFLA refers to other frogs; it can be obtained by updating Eq. (6). A frog's leaping vision lies between the frog's current position and the position with maximum food in current. However, a position with more food around the place with maximum food may exist in the current search. This position could lie outside the area between the frog's current position and the position with maximum food. The original SFLA updating equation cannot determine this region, thereby limiting the optimization ability and affecting the convergence speed of the algorithm. Eq. (6) for the k th mixed iteration can thus be modified as follows:

$$D_{i,w}(k) = rc(X_{i,s} - X_{i,w}(k)), \quad (9)$$

where c is the leaping vision factor and $c \geq 1$. The new updated equation increases the frog's leaping range for every step, expands the frog's leaping vision, and enhances the optimization capability of the algorithm. The algorithm can expand the range of the search by expanding c . However, c cannot be infinitely expanded; otherwise, the function of the current optimal position will weaken and the algorithm will evolve into a random search algorithm. Therefore, the scope of c is generally set to $1 \leq c \leq 3$. Leaping vision weight w was employed in this study to better control the relation between local exploitation and global exploration in the frog's leaping process. Eq. (9) is modified as follows:

$$D_{i,w}(k) = wrc(X_{i,s} - X_{i,w}(k)). \quad (10)$$

Parameter w , the leaping vision weight, has a significant impact on the convergence behavior of the algorithm. A large parameter w indicates strong global search capability and weak local search capability. Conversely, a small parameter w indicates weak global search capability and strong local search capability. The impact of the previous pace (speed) on the current pace (speed) can be controlled by adjusting w . A linear descending equation was applied in this study as follows:

$$w = w_{\min} + (k_{\max} - k)(w_{\max} - w_{\min})/k_{\max}. \quad (11)$$

Therefore, w gradually descends from w_{\max} to w_{\min} in a linear manner as the iteration progresses. Global exploration and local exploitation can therefore be properly controlled. The exploration capability is strong early in the iteration; a large solution space can be searched, and new regions can be constantly explored. Late in the iteration, the algorithm gradually shrinks to a preferred area for a finer search and increases the convergence speed. Eq. (8) can be updated as follows:

$$D_{i,w}(k) = wrc(X_b - X_{i,w}(k)). \quad (12)$$

This modified SFLA is referred to as MSFLA hereafter.

4.3. MSFLA based on integer encoding for resource scheduling

t VMs were allocated in q hosts. The scheduling scheme was then encoded into

$$X = [x_1, x_2, \dots, x_t], \quad x_i \in [1, q], x_i \text{ is an integer.} \quad (13)$$

The updated equation for MSFLA must be adjusted in this encoding scheme. To neglect group subscript i and iteration k , X_w , which represents the worst frog position in the current group, was used. d_j represents the j th component moving distance of X_w . Therefore, Eq. (10) can be updated into Eq. (14):

$$d_j = wrc(x_{sj} - x_{wj}), \quad (14)$$

where x_{sj} and x_{wj} are the values of the j th component in X_s and X_w , respectively, which can be summarized in the following equation:

$$d'_j = \begin{cases} 0 & x_{sj} = x_{wj}, \\ \frac{d_j}{wc|(x_{sj} - x_{wj})|} & \text{otherwise,} \end{cases} \quad (15)$$

d'_j is employed to determine the value of the corresponding dimension of X_w to be set as x_{sj} or x_{wj} . The larger d'_j is, the larger x_{sj} setting probability is. Moreover, the smaller d'_j is, the larger the x_{wj} setting probability is. A sigmoid function is used to convert d'_j into a probability value between 0 and 1:

$$\text{sig}(d'_j) = \frac{1}{1 + \exp(-d'_j)}, \quad (16)$$

where $\text{sig}(d'_j)$ represents the probability of x'_{wj} to be set as x_{sj} or x_{wj} . The updated equation for MSFLA can be modified as follows:

$$x'_{wj} = \begin{cases} x_{sj}, & r' < \text{sig}(d'_j), \\ x_{wj}, & r' \geq \text{sig}(d'_j) \end{cases} \quad (17)$$

where r' is a randomly generated number ($r' \in [0, 1]$). X'_w is produced in all dimensions by the aforementioned updating process. Similarly, if the individual needs to learn from the global optimal, Eq. (18) can substitute for Eq. (14):

$$d_j = wrc(x_{bj} - x_{wj}), \quad (18)$$

where x_{bj} is the value of the j th dimension of X_b . Given that the encoding of each dimension value is an integer, the coding sequence is the VMs scheduling allocation scheme, which does not need to be decoded.

4.4. Local search with the improved EO

The random solution of MSFLA can enrich group information and maintain sample diversity. However, the algorithm's convergence efficiency is low. A good local search technology can be applied to replace the random solution on the premise of maintaining the diversified samples, thereby improving the search efficiency. Extremal optimization (EO) proposed by Boettcher and Percus (1999) based on the fundamentals of statistical physics and self-organized criticality (SOC) is a new general-purpose local search optimization approach. The evolution in this method is driven by a process where the weakest species in the population is always forced to mutate. EO successively eliminates those worst components in the sub-optimal solutions, and has been successfully applied to many continuous and discrete optimization problems (Jahan, Mohammad, & Akbarzadeh, 2012). Considering that EO has strong local-search ability, in order to further improve the local search ability of MSFLA, we present a novel hybrid algorithm (MSFLA-EO) which makes full use of the exploration ability of MSFLA and the exploitation ability of EO. This algorithm is used to replace the random solution of MSFLA with the improved EO

process. The EO algorithm framework (Boettcher & Percus, 1999) is shown in Fig. 6.

4.5. Evaluation of the improved EO component fitness value in MSFLA-EO

When EO evolves, it becomes an individual consisting of several components. Every component matches the corresponding solution vector. The EO individual and the frog position of MSFLA have the same encoding. The algorithm regards each internal component as having different contributions to the individual quality. The fitness value is defined by the contribution each component provides to the individual objective function value. The smallest fitness value is the worst component. The worst component is mutated in each EO iteration. In this study, every component matches a VM that needs to be migrated. Therefore, the fitness value of the component is as follows:

$$\lambda_i = \begin{cases} \frac{1}{\alpha(\text{Utlz}(h(i)) - \text{upper_th}) + \beta \text{Utlz}(i)} & \text{Utlz}(h(i)) > \text{upper_th}, \\ \frac{1}{\beta \text{Utlz}(i)} & \text{Utlz}(h(i)) \leq \text{upper_th}, \end{cases} \quad (19)$$

where λ_i is the fitness value of component i (VM); $h(i)$ is the target host for VM i to be allocated; $\text{Utlz}(h(i))$ is the processor utilization for target host $h(i)$; $\text{Utlz}(i)$ is the processor utilization for VM i ; and α and β are the weighting parameters. As indicated by the definition of fitness value, for the host whose processor utilization exceeds the upper threshold, the greater the processor utilization is, the smaller the fitness value of VM pre-assigned to the host will be; for the VM on the same host, the greater the processor utilization is, the corresponding smaller the fitness value will be. The component with a small fitness value is normally selected for mutation.

The selection of the mutation component was performed in this study by the power-law probability in the sequence of the fitness value to maintain the global search of the algorithm and to avoid the local optimum. t VMs must be migrated. The selected probability of each VM has the following allocation in accordance with the sequence of the fitness value from largest to smallest:

$$p(r) \propto r^{-\tau} \quad 1 \leq r \leq t, \tau \geq 0. \quad (20)$$

4.6. Establishment of the component mutation solution

Similar scheduling strategies exist in the component mutation process and newly applied VMs as shown in Fig. 2; that is, the configured hosts meet the SLA requirements and maximize energy saving.

4.7. Runtime complexity and iteration number of EO

For a scheduling problem that involves t VMs and q hosts, in each iteration of EO, the runtime complexity of fitness evaluation for components is $O(t)$, the runtime complexity of selection for the mutation component is $O(t)$, and the complexity of the allocation that places the component (VM) that needs to be mutated (migrated) on one host is $O(q)$. Therefore, the total runtime complexity is $O(N(2t + q))$, where N is the iteration number of EO. Computational complexity increases with the increase in the N value. Obviously, a trade-off exists between computational cost and convergent performance. According to our experimental observation, the N value should not be too large in our hybrid algorithm (MSFLA-EO). The value of iteration number N is set to 2 in our research.

1. Randomly generate an individual $X = (x_1, x_2, \dots, x_t)$. Set the optimal solution $X_{best} = X$. Objective function value is $C(X)$.
2. For the "current" individual X
 - a. evaluate the fitness λ_i for each decision variable $x_i, i \in \{1, 2, \dots, t\}$;
 - b. find x_j satisfying $\lambda_j \leq \lambda_i, i = 1, 2, \dots, t$, i.e., x_j creates the "worst fitness";
 - c. get X' where x_j must change its state, X' is the neighborhood of X ;
 - d. accept $X = X'$ unconditionally;
 - e. if $C(X) < C(X_{best})$, then $X_{best} = X$;
3. Repeat Step 2 as long as desired.

Fig. 6. EO algorithm framework.

5. Experiment and analysis

CloudSim toolkit, a cloud computing simulation platform for simulations (Buyya, Ranjan, & Calheiros, 2009) launched by Grid Laboratory of the University of Melbourne and the Gridbus project in 2009, was utilized in this study. This toolkit adopts the GridSim programming model to support cloud computing research and development, provide the characteristics of cloud computing, and support resource management and scheduling simulation. The cloud information service of CloudSim and DataCenterBroker served as the core of simulation scheduling to realize resource discovery and information exchange. User-developed scheduling algorithms can be implemented in DataCenterBroker to simulate the scheduling algorithm. The latest version supports the energy-aware simulated experience of hosts.

The simulation cloud platform was configured with 1000 hosts, including 500 ProLiant DL360 G4p hosts (configured with 3400 MHz * dual-core, 6 GB memory, 1 GB network bandwidth) and 500 ProLiant ML110 G3 hosts (configured with 3000 MHz * dual-core, 4 GB memory, 1 GB network bandwidth). The energy consumption of each server was calculated with Eq. (3) according to average result of benchmark testing statistics in the fourth quarter of 2010 (SPECpower, 2010) with E_{max} value of 259 W. The task load data of the VMs were obtained from the CoMon Project, which is one of the monitoring facilities of PlanetLab (Park & Pai, 2006). The data mainly contains CPU utilization in over 1000 VMs of 500 different areas around the world tested every 5 min. We have chosen 10 days from the workload traces collected during March and April 2011. The characteristics of the data for each day are shown in Table 1. Each VM load on the test set was considered a new VM application during testing. In the simulation, the data center should be able to allocate resources and run the service. An intelligent scheduling framework was adopted for the intelligent scheduling of all VM loads. SFLA, MSFLA, and MSFLA-EO were utilized to compare scheduling. Under similar initial conditions, the local parameters of the three scheduling algorithms in SFLA are as follows: $F = 100$, $m = 10$, $n = 10$ and the number of internal iterations inside groups is 10. In MSFLA and MSFLA-EO, $c = 1.5$, $w_{min} = 0.8$, $w_{max} = 2.5$, and $k_{max} = 1000$; in MSFLA-EO, $\tau = 1.5$. If the optimal solution does not improve after mixed iteration for

Table 1

Workload Data Characteristics (CPU Utilization).

Instance	Date	Number of VMs	Mean (%)	St. dev. (%)
p1	03/03/2011	1052	12.31	17.09
p2	06/03/2011	898	11.44	16.83
p3	09/03/2011	1061	10.70	15.57
p4	22/03/2011	1516	9.26	12.78
p5	25/03/2011	1078	10.56	14.14
p6	03/04/2011	1463	12.39	16.55
p7	09/04/2011	1358	11.12	15.09
p8	11/04/2011	1233	11.56	15.07
p9	12/04/2011	1054	11.54	15.15
p10	20/04/2011	1033	10.43	15.21

continuous $G = 300$ times, a convergence condition will develop and the algorithm will be exited. Each scheduling scheme was executed 40 times, from which the average value will be obtained. A comparison was conducted in terms of SLAV and energy consumption in the data center to evaluate the performance of the algorithm. The experimental results are shown in Tables 2–4, Figs. 7 and 8.

Table 2 present the simulation results of the scheduling algorithms for p1 (i.e., March 3, 2011). SLAV in the first column shows the performance of the different algorithms obtained by testing the SLAV value from 1% to 5% ($\times E - 3$). In the second column, LTH is the lower threshold utilization of the host. Each SLAV value contains LTH from 0.1 to 0.5. The first scheduling algorithm is Non Power-Aware (NPA) policy, this policy does not apply any power aware optimizations and implies that all hosts run at 100% CPU utilization and consume maximum power all the time (Beloglazov et al., 2012). The second is DVFS mode (Chase et al., 2001) in which all VMs does not have any reset option or consolidation processing. The third strategy is for resource adjustment strategy (proposed in this paper). In this strategy, VMs are allocated by standard SFLA. The fourth strategy allocates VMs with the improved SFLA (MSFLA). The fifth strategy is the dynamic allocation strategy integrated with EO (MSFLA-EO). Each algorithm was tested by different SLAV and LTH combinations. In this manner, the value of energy consumption (kWh) and the VM migrations (Migr.) number can be obtained.

Table 2

Comparison Results of Several Scheduling Strategies for P1.

SLAV ($\times E-3$) (%)	LTH	NPA		DVFS		SFLA		MSFLA		MSFLA-EO	
		Energy (kWh)	Migr.	Energy (kWh)	Migr.	Energy (kWh)	Migr.	Energy (kWh)	Migr.	Energy (kWh)	Migr.
1	0.1	3455	0	856	0	186	13,250	153	14,125	145	14,293
	0.2	3455	0	856	0	175	14,586	145	14,243	135	14,452
	0.3	3455	0	856	0	164	14,889	143	14,991	134	15,213
	0.4	3455	0	856	0	168	15,864	141	15,722	138	15,338
	0.5	3455	0	856	0	171	15,912	145	15,874	143	15,864
2	0.1	3455	0	856	0	134	14,841	117	15,101	106	14,867
	0.2	3455	0	856	0	125	15,547	107	15,246	102	15,146
	0.3	3455	0	856	0	126	16,403	103	15,894	99	15,534
	0.4	3455	0	856	0	128	16,900	105	16,025	100	15,861
	0.5	3455	0	856	0	141	17,055	108	16,856	107	16,030
3	0.1	3455	0	856	0	124	15,455	107	15,852	102	15,662
	0.2	3455	0	856	0	115	15,965	98	15,951	97	15,753
	0.3	3455	0	856	0	113	16,819	96	16,023	93	16,142
	0.4	3455	0	856	0	117	17,210	103	17,409	104	16,852
	0.5	3455	0	856	0	128	17,521	108	17,476	104	17,013
4	0.1	3455	0	856	0	112	15,768	105	15,815	99	15,928
	0.2	3455	0	856	0	106	16,212	95	15,895	95	16,353
	0.3	3455	0	856	0	103	16,976	95	16,853	93	16,448
	0.4	3455	0	856	0	103	17,328	99	17,001	98	16,657
	0.5	3455	0	856	0	109	17,639	104	17,548	101	17,106
5	0.1	3455	0	856	0	109	16,037	98	16,346	94	16,127
	0.2	3455	0	856	0	103	16,326	94	16,657	87	16,328
	0.3	3455	0	856	0	99	16,934	92	16,850	87	16,669
	0.4	3455	0	856	0	99	17,246	90	16,952	89	16,920
	0.5	3455	0	856	0	104	17,730	97	17,159	95	17,228

Table 3

Scheduling Performance of the Different Algorithms under Different Parameter G.

G	SFLA		MSFLA		MSFLA-EO	
	Time (S)	Energy (kWh)	Time (S)	Energy (kWh)	Time (S)	Energy (kWh)
100	7.2	122	5.8	101	2.5	91
200	9.9	109	10.7	96	4.2	87
300	12.8	99	16.1	91	5.7	86
500	20	95	25.2	90	7.8	86
800	32.6	93	41.1	89	10.3	85
1000	41.7	91	51.9	89	12.5	85

Table 4

Energy consumption performance of the different algorithms. The bold values are the best results.

Algorithm	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
OODA	104.21	76.43	88.22	107.4	92.65	137.9	112.92	105.8	95.65	77.84
	104.21	76.43	88.22	107	92.7	138	112.9	105.8	95.65	77.84
APA-VMP	94.21	72.29	78.73	100.8	86.58	107.5	106.9	97.29	89.13	78.68
	93.01	71.52	76.5	95.3	84.4	106	104.3	95.57	87.5	76.08
MSFLA	94.25	71.98	83.26	99.27	85.25	106.6	103.28	96.5	88.05	75.22
	91.35	70.18	81.7	97.8	83.1	104	101.2	94.55	86.14	73.14
MSFLA-EO	91.89	70.64	79.41	94.24	83.36	102.9	98.49	92.17	82.33	68.15
	90.1	68.95	78.14	93.2	82.3	102	97.5	90.83	81.17	67.68

The table shows that the energy consumption of the cloud data center is greatly reduced by the resource adjustment strategy proposed in this paper. Energy consumption accounts for only about 4% of the NPA strategy when the VM allocation strategy based on MSFLA-EO is adopted. Hence, this strategy reduces energy consumption significantly. Our experimental conditions, which include 1000 hosts, are different from the actual hardware facilities of CoMon Project. Thus, the percentage (about 4%) between our proposed scheme and the NPA strategy for energy consumption is different from the CPU mean utilization (about 10%) shown in Table 1. When energy consumption is at its lowest (87 kWh), 5% $\times E-3$ SLA violation is still maintained. This maintained value

indicates that the system maintains very good quality of service as it saves energy.

According to the experimental data, compared with SFLA, MSFLA and MSFLA-EO can improve system performance. The leaping vision and weighting factors in MSFLA allow for better control of the frog leaping search and enhance the efficiency of the algorithm. For instance, when the equation is $SLAV = 2\% \times E-3$, $LTH = 0.4$, the total number of VM migrations required by the two allocation strategies (SFLA and MSFLA) are approximately the same. However, the allocation efficiency of MSFLA is higher; energy consumption is reduced by 17% $((128 - 105)/128)$. Moreover, the fine local search strategy of EO improves the scheduling

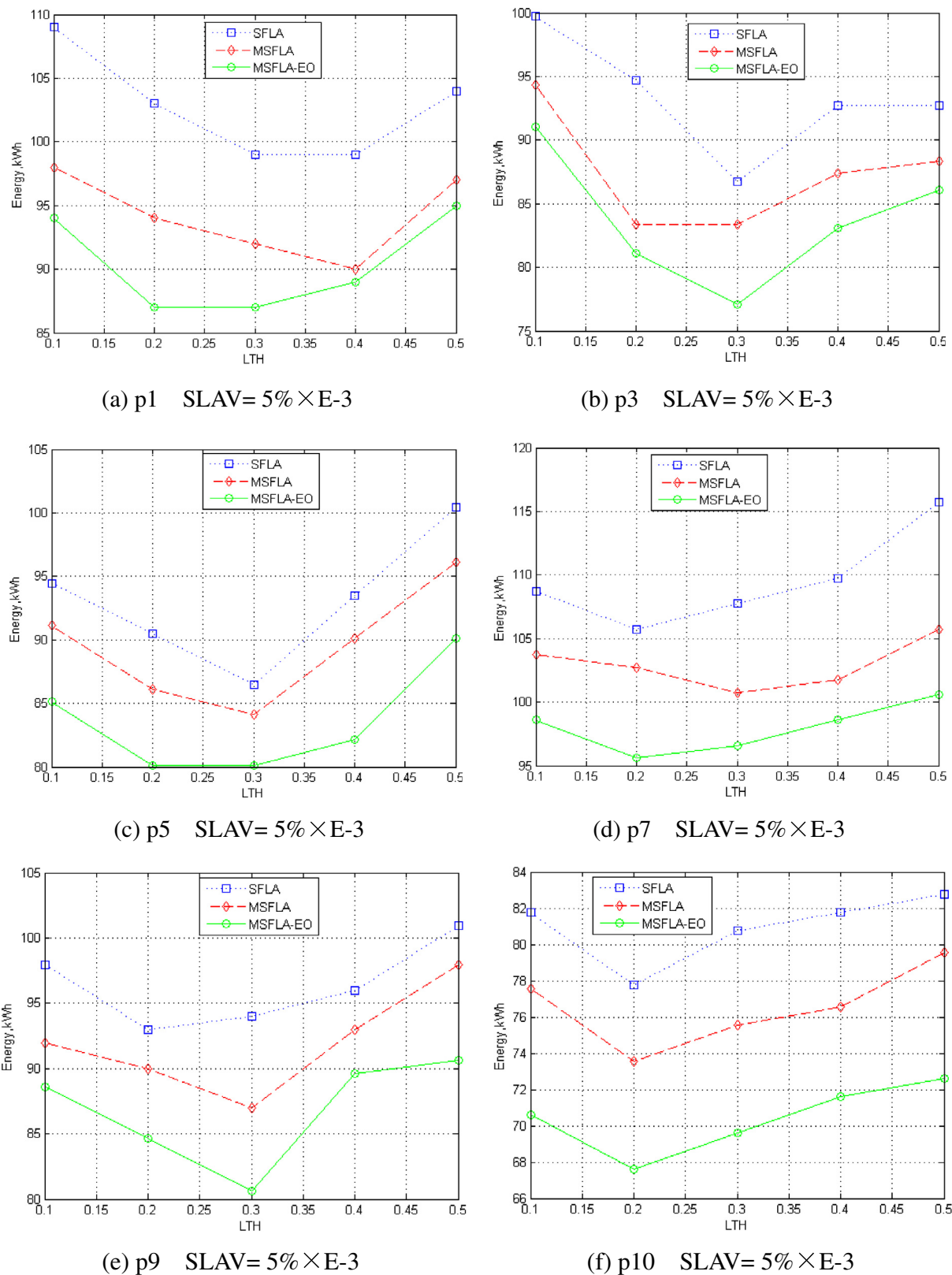


Fig. 7. Energy consumption performance of different algorithms for different instances (p1, p3, p5, p7, p9, and p10).

efficiency of MSFLA and saves energy when the number of VM migrations is similar. For instance, when the parameters are $SLAV = 5\% \times E-3$, $LTH = 0.3$, MSFLA-EO consumes minimal energy (87 kWh).

Fig. 7 shows the energy consumption performance of the different algorithms for different instances (p1, p3, p5, p7, p9, and p10) under the condition that $SLAV$ is equal to $5\% \times E-3$. MSFLA-EO outperforms the other two algorithms, i.e., SFLA and MSFLA. The

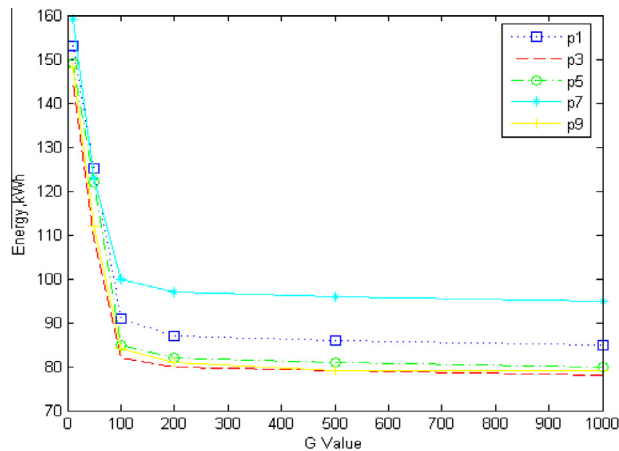


Fig. 8. The trend of changes of obtained solutions by MSFLA-EO with different parameter G.

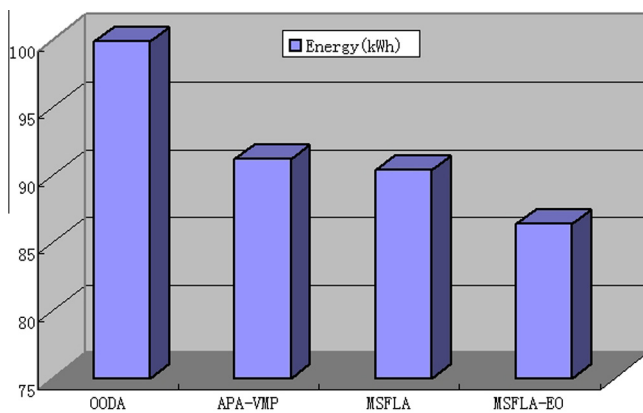


Fig. 9. Mean energy consumption performance of the different algorithms.

scheduling strategy of MSFLA-EO consumes the minimum energy in all the cases. Fig. 7 also shows that LTH is not too small. Although LTH can avoid putting too many hosts in the sleeping state when its value is extremely small, the problem of high energy consumption arises. The LTH value should not be too large because it would cause frequent switching between sleeping and waking states as well as increased energy consumption. An LTH value that is large and approaches the upper threshold causes difficulty in assigning VMs. Generally, the LTH value should be set between 0.2 and 0.3.

In the aforementioned algorithms, when the current optimal solution shows no improvement for continuous G times mixed iteration, the algorithm is assumed to have achieved convergence and is exited. The scheduling scheme in Table 3 shows the average execution time and energy consumption performance for all instances in different parameter G . Fig. 8 shows the trend of changes in the solutions obtained by MSFLA-EO with different parameter G for instances p1, p3, p5, p7, and p9. The testing equation is $SLAV = 5\% \times E-3$, $LTH = 0.3$. Different parameter G values will have different results from the different algorithms. The iterations of the algorithm increase significantly with the increase in G . Likewise, the algorithm scheduling optimization results improve with the increase in scheduling time. At the same G value, MSFLA-EO provides an excellent scheduling result within a short period of time. When $G = 300$, MSFLA-EO can implement complete scheduling at an average of 5.7 s and achieve a final energy consumption of 86 kWh. Obviously, there is tradeoff between the

computational cost and the convergent performance. The G value in this study was usually set between 200 and 300 for MSFLA-EO to achieve the desired scheduling effect. The scheduling algorithm is performed periodically. A complete scheduling configuration is produced for each execution. The scheduling center performs scheduling according to the scheduling configuration. For hosts that need to be reconfigured, VM migrations or switching state is performed. Therefore, the execution time of the algorithm must be less than the scheduling period for the successful performance of proposed scheduling scheme.

Beloglazov and Buyya (2011) introduces an energy-aware algorithm called Optimal Online Deterministic Algorithms (OODA) to address the scheduling problem in cloud data centers. This algorithm applies the minimum migration time (MMT) strategy to select VMs that need to be migrated and uses the MBFD (Beloglazov et al., 2012) algorithm to address the VM placement problem. This algorithm also applies the optimal strategy and simulates certain inputs under the CloudSim platform for the instances p1–p10. With the Median Absolute Deviation (MAD) host overloading detection method, the derived mean SLAV is $4.12\% \times E-3$. An experiment was conducted in this study under similar testing conditions (with the same testing platform and input condition) to compared MSFLA-EO with this algorithm. The algorithm target SLAV is set to $4.12\% \times E-3$ for better comparison. Jeyarani, Nagaveni, and Vasanth Ramc (2012) proposes adaptive power-aware virtual machine provisioner (APA-VMP), which makes use of swarm intelligence methodology to detect and track the changing optimal target servers for VM placement very efficiently. The scenario is experimented by novel self-adaptive particle swarm optimization for VM provisioning, which makes possible use of the power saving states of idle servers and instantaneous workload on the operational servers. To compare the effectiveness of these algorithms, APA-VMP replaced MSFLA-EO in our experiments to address the VM allocation problems under same experimental conditions. The experimental results of comparison are shown in Table 4. Each instance was executed 40 times. The average energy consumption values and the best results (bold) are recorded for each algorithm. Fig. 9 shows the mean energy consumption values obtained by each algorithm for all instances.

Statistical tests were implemented as experimental tests to investigate the quality of the results. We applied Friedman's test (García, Molina, Lozano, & Herrera, 2009) to determine if global differences exist in the results for each algorithm. The average energy consumption values of each algorithm were considered. All computations were performed with the statistical software SPSS. The χ^2 -value and p -value of Friedman's test are 24.6 and $0.18 \times E-3$, respectively. Given that the p -value of Friedman's test is lower than the level of significance considered ($\alpha = 0.05$), significant differences exist among the observed results of each algorithm in all the instances. Post-hoc statistical analysis can help detect concrete differences among algorithms. Given that non-parametric tests do not require explicit conditions to be conducted, it is recommended that the sample of results be obtained following the same criterion; that is, compute the same aggregation (average, mode, etc.) over the same number of runs for each algorithm and problem. In this study, The Wilcoxon test (García et al., 2009), which is a non-parametric test, was utilized to perform individual comparisons between two algorithms (pairwise comparisons). The p -value associated with a comparison was obtained by normal approximation for the Wilcoxon T statistic. Therefore, a p -value lower than the level of significance α indicates that obvious differences exist between two algorithms.

Tables 5 summarizes the results of the Wilcoxon test. It displays the sum of rankings obtained in each comparison and the associated p -value. APA-VMP and MSFLA outperform OODA because the p -values are very small. The statistical significance of

Table 5

Wilcoxon test for pairwise comparisons.

	APA-VMP vs OODA	MSFLA vs OODA	MSFLA-EO vs OODA	MSFLA vs APA-VMP	MSFLA-EO vs APA-VMP	MSFLA-EO vs MSFLA
R+	1	0	0	5.5	1	0
R-	6.000	5.500	5.500	5.500	6.000	5.500
p-value	.007	.005	.005	.093	.007	.005

Table 6

Performance of MSFLA-EO for Different Host Scales.

The number of host		Time(S)	Energy (kWh)
ProLiant DL360 G4	ProLiant ML110 G3		
500	500	4.2	87
1000	1000	11.7	92
2500	2500	29.5	96
5000	5000	116.5	102

combining pairwise comparisons is provided by $p = 1 - \prod_{i=1}^{k-1} (1 - pH_i)$ (García et al., 2009). We can deduce that MSFLA-EO is better than the other three algorithms with a p -value of $p = 1 - (1 - 0.005)(1 - 0.007)(1 - 0.005) = 0.0169$. Therefore, we conclude that MSFLA-EO outperforms OODA, APA-VMP, and MSFLA with level of significance $\alpha = 0.02$; however, MSFLA does not outperform APA-VMP under this significance level.

The performance of the proposed algorithm under different host scales was also evaluated. As shown in Table 6, the number of hosts is 1000, 2000, 5000, and 10,000 for each case (the numbers of ProLiant DL360 G4 and ProLiant ML110 G3 are equal). The parameters are SLAV = $5\% \times E-3$, LTH = 0.3, and $G = 200$. The other parameter settings were retained. All instances (p1–p10) were tested, and the mean results are shown in Table 4. The overall energy consumption exhibits minimal change with the increase in the number of hosts. This result shows that the proposed algorithm can accurately and effectively assign VMs to the appropriate hosts. Therefore, the hosts are maintained in the best energy consumption condition (most of the hosts are switched to the sleeping power mode) and energy saving is achieved.

6. Conclusion

Given that the main purpose of cloud computing is to help consumers manage unexpected demands for resources, data centers have become inherently dynamic. In the IaaS cloud, instantaneous resource provisioning on demand as well as resource reclamation occur when the customer no longer requires the resources. Considering the thousands of hosts in a cloud data center, the heuristic method can provide a better, more reasonable solution within a short duration compared with conventional methods. An energy-aware resource allocation scheme that involves the dynamic consolidation of VMs was presented and evaluated in this study. Performance models were built to help explore the trade-offs between QoS and energy savings. The primary contribution of the proposed resource management scheme is the dynamic consolidation of the host's resource by VM migration technology as well as switching idle or low-utilized hosts into the power saving mode for energy conservation while guaranteeing adherence to SLAs. A novel hybrid algorithm (MSFLA-EO), which maximizes the exploration capability of MSFLA and the exploitation capability of EO, was developed to address the dynamic allocation problem of VMs effectively. The improved algorithm not only enhances the SFLA frog leaping vision but also improves local search capability. It can also be easily applied to other combinatorial optimization problems.

Humans have become increasingly conscious of the environment. Recent studies show that data centers represent a large

and rapidly growing energy consumption sector of the economy and are a significant source of CO₂ emissions. Reducing greenhouse gas emissions is the key energy policy focus of many countries around the world. Both computer research community and industry focus on green computing to make efficient use of computing systems and reduce their environmental and social impact. Hardware techniques can be utilized to solve energy problems; however, in many cases, these techniques require software intervention to achieve optimum results. The proposed data center resource management scheme can not only ensure user QoS (specified by SLAs) but can also maximize energy saving for green computing. The experimental results show that the proposed method results in a substantial reduction in energy consumption in cloud data centers. For resource providers, the optimal allocation of VMs will result in high utilization of resources, and therefore, reduced operational costs. End users will benefit from the reduced prices.

In our future work, we plan to investigate more complex workload models, where the dynamic behavior of the data center can change dramatically and the resource management scheme must be able to quickly respond to these changes. We will focus on designing and implementing power-aware software frameworks encompassing dynamic prediction service using adaptive provisioning service and the multi-phase scheduling model, which can allocate resources effectively, to improve power conservation in modern data centers. Several researchers have addressed the related fields (Yang, 2013). On the other hand, we have investigated the problem of energy-aware dynamic consolidation of VMs according to the current CPU utilization. However, to allow for better VM placement optimization, VMs should be reallocated according to the current utilization of multiple system resources, including the CPU, RAM, and network bandwidth. A generic cloud computing environment (IaaS) should be built to serve multiple applications for multiple users, which create mixed workloads and complicate the workload characterization. Our future research will be directed toward the development of fast and effective algorithms for VM placement optimization across multiple resources for large-scale systems.

Acknowledgments

The authors would also like to thank the anonymous referees for their helpful comments and suggestions on improving the presentation of this paper. This work is supported by the National Natural Science Foundation of China under Grant Nos. 61301298, 61171124 and 61005049 and the Scientific Research and Development Foundation of Shenzhen under Grant Nos. JCYJ20120613161222123 and JCYJ2013032911062410.

References

- Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware source allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 5(28), 755–768.
- Beloglazov, A., & Buyya, R. (2011). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 1–24.
- Berral, J. L., Goiri, I., Nou, R., Juli, F., Guitart, J., Gavalda, R., & Torres, J. (2010). Towards energy-aware scheduling in data centers using machine learning. In *Proceedings*

- of the 1st International Conference on Energy-Efficient Computing and Networking, Passau, Germany (pp. 215–224).
- Boettcher, S., & Percus, A. G. (1999). Extremal optimization: Methods derived from co-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA* (pp. 101–106).
- Buyya, R., Ranjan, R., & Calheiros, R. N. (2009). Modeling and simulation of scalable cloud computing environments and the CloudSim Toolkit: Challenges and opportunities. In *Proceedings of the seventh high performance computing and simulation conference (HPCS 2009, ISBN: 978-1-4244-4907-1), Leipzig, Germany* (pp. 21–24). New York, USA: IEEE Press.
- Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., & Doyle, R. P. (2001). Managing energy and server resources in hosting centers. In *Proceedings of the 18th ACM symposium on operating systems principles* (pp. 103–116). New York, NY, USA: ACM.
- Eusuff, M., & Lansey, K. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resource Plan and Management*, 129(3), 10–25.
- Fan, X., Weber, W. D., & Barroso, L. A. (2007). Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on computer architecture (ISCA 2007)* (pp. 13–23). New York, NY, USA: ACM.
- Fernández-Montes, A., Gonzalez-Abril, L., Ortega, J. A., & Lefèvre, L. (2012). Smart scheduling for saving energy in grid computing. *Expert Systems with Applications*, 39, 9443–9450.
- Gandhi, A., Harchol Balter, M., Das, R., & Lefurgy, C. (2009). Optimal power allocation in server farms. In *Proceedings of the 11th int. joint conference on measurement and modeling of computer systems* (pp. 157–168). New York, USA: ACM.
- García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6), 617–644.
- Jahan, M. V., Mohammad, R., & Akbarzadeh, T. (2012). Extremal optimization vs. learning automata: Strategies for spin selection in portfolio selection problems. *Applied Soft Computing*, 12(10), 3276–3284.
- Jeyarani, R., Nagaveni, N., & Vasanth Ramc, R. (2012). Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence. *Future Generation Computer Systems*, 28, 811–821.
- Kaplan, J. M., Forest, W., & Kindler, N. (2008). *Revolutionizing data centre energy efficiency*. McKinsey & Company (White paper).
- Kramer, H., Petrucci, V., & Subramanian, A. (2012). A column generation approach for power-aware optimization of virtualized heterogeneous server clusters. *Computers & Industrial Engineering*, 63(3), 652–662.
- Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., & Jiang, G. (2009). Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1), 1–15.
- Lee, Y. C., & Zomaya, A. Y. (2010). Rescheduling for reliable job completion with the support of clouds. *Future Generation Computer Systems*, 26, 1192–1199.
- Minas, L., & Ellison, B. (2009). *Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers*. Intel Press.
- Park, K. S., & Pai, V. S. (2006). CoMon: A mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Operating Systems Review*, 40(1), 74–80.
- Rodero, I., Viswanathan, H., & Lee, E. (2012). Energy-efficient thermal-aware autonomic management of virtualized HPC cloud infrastructure. *Journal of Grid Computing*, 10(3), 447–473.
- Rusu, C., Ferreira, A., Scordino, C., & Watson, A. (2006). Energy efficient real-time heterogeneous server clusters. In *Proceedings of the IEEE real-time and embedded technology and applications symposium, San Jose, USA* (pp. 418–428).
- The SPECpower benchmark results for the fourth quarter of 2010 [EB/OL]. http://www.spec.org/power_ssj2008/results/res2010q4/.
- Venkatachalam, V., & Franz, M. (2005). Power reduction techniques for microprocessor systems[J]. *ACM Computing Surveys*, 37, 195–237.
- Verma, A., Ahuja, P., & Neogi, A. (2008). pMapper: Power and migration cost aware application placement in virtualized systems. In *Proceedings of the ninth ACM/IFIP/USENIX international conference on middleware (Middleware 2008)* (pp. 243–264). Leuven, Belgium: Springer.
- Verma, A., Dasgupta, G., Nayak, T. K., De, P., & Kothari, R. (2009). Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 USENIX annual technical conference, San Diego, CA, USA* (pp. 28–38).
- Voorsluys, W., Broberg, J., Venugopal, S., & Buyya, R. (2009). Cost of virtual machine live migration in clouds: A performance evaluation. In *Proceedings of the first international conference on cloud computing (CloudCom 2009)* (pp. 105–125). Beijing, China: Springer.
- Yang, S. Y. (2013). A novel cloud information agent system with web service techniques: Example of an energy-saving multi-agent system. *Expert Systems with Applications*, 40, 1758–1785.