Stackelberg Game Approach for Energy-Aware Resource Allocation in Data Centers

Bo Yang, Zhiyong Li, *Member, IEEE*, Shaomiao Chen, Tao Wang, and Keqin Li, *Fellow, IEEE*

Abstract—Data centers hosting distributed computing systems consume huge amounts of electrical energy, contributing to high operational costs, whereas the utilization of data centers continues to be very low. Moreover, a data center generally consists of heterogeneous servers with different performance and energy. Failure to fully consider the heterogeneity of servers will lead to both sub-optimal energy saving and performance. In this study, we employ game theoretic approaches to model the problem of minimizing energy consumption as a Stackelberg game. In our model, the system monitor, who plays the role of the leader, can maximize profit by adjusting resource provisioning, whereas scheduler agents, who act as followers, can select resources to obtain optimal performance. In addition, we model the problem of minimizing average response time of tasks as a noncooperative game among decentralized scheduler agents as they compete with one another in the sharing resources. Several algorithms are presented to implement the game models. Simulation results demonstrate that the proposed technique has immense potential to improve energy efficiency under dynamic work scenarios without compromising service level agreements.

Index Terms—Data centers, dynamic capacity provisioning, energy efficiency, game theory

1 Introduction

1.1 Motivation

Large data centers (DCs) as a cost-effective platform for hosting large-scale Internet application, such as video distribution networks, and content delivery networks etc., enjoy economies of scale by amortizing long-term capital investments over a considerable number of machines. However, this DCs also incur enormous energy costs. The energy consumption of DCs is expected to grow to 8 percent of the global electricity supply by 2020 [1]. A 2 percent reduction in energy cost for a few large companies (such as Akamai Technologies) can translate to over a million dollars in cost saving [2]. Therefore, energy efficiency in DCs has attracted increasing interests from both the research community and the industry. Consequently, reducing energy consumption has become a primary concern for DC operators.

In practice, most modern DC servers are often utilized by only approximately 10 to 50 percent on average[2]; that is, they are rarely fully utilized during daily workloads. Moreover a distinct observation is that a server consumes at least 60 percent of its peak power when it is active, even without serving any request. By contrast, this figure drops to only 5 W when the server is in sleep mode [1], [3]. Thus, the most effective energy-saving strategy for DCs operators should be to adjust the number of active machines in a DC dynamically to reduce energy consumption while satisfying the quality of service (QoS) requirements specified by end users via service level agreements (SLAs). Therefore, DC operators are expect to place some unnecessary servers in sleep or off mode to improve the utilization of DCs when the incoming request volume is low, such as during off-peak hours.

However, a trade-off frequently exists between energy consumption and degraded processing performance. Switching off servers can reduce energy consumption, but doing so also decrease the service capacity and may consequently incur high scheduling delay. To address this issue and to promote green computing, DC resources should be allocated to satisfy QoS for end-users while achieving high energy savings.

In addition, DC users are naturally required to minimize the response time for their tasks, and DC operators are required to maximize their revenues by attracting users and by improving resource utilization. However, achieving these requirements is difficult in distributed systems such as DCs, where no central authority is available to control the allocation, whereas users or scheduler agents are free to act in a selfish manner.

1.2 Our Contributions

In this study, we introduce the stackelberg game theoretic approach to address aforementioned issues. Game theory provides a natural paradigm to design decentralized mechanisms [27], which can help obtain an in-depth analytical understanding of the service provisioning problem of DCs. In our model, the scheduler agents, who act as followers

- *B. Yang is with the College of Computer Science and Electronic Engineering, Hunan University, National Supercomputing Center in Changsha, Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha 410082, China and the Department of Information and Management of Hunan University of Finance and Economics, Changsha 410205, China. E-mail: bo_yang@hnu.edu.cn.*
- *Z. Li, S. Chen, and T. Wang are with the College of Computer Science and Electronic Engineering of Hunan University, National Supercomputing Center in Changsha, Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha 410082, China. E-mail: {zhiyong.li, chensm1987}@hnu.edu.cn, wangtao1999@foxmail.com.*
- *K. Li is with the College of Computer Science and Electronic Engineering, Hunan University, National Supercomputing Center in Changsha, and the Department of Computer Science, State University of New York, New Paltz, NY 12561. E-mail: lik@newpaltz.edu.*

Manuscript received 11 Aug. 2015; revised 31 Jan. 2016; accepted 25 Feb. 2016. Date of publication 3 Mar. 2016; date of current version 16 Nov. 2016. Recommended for acceptance by I. Brandic.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2016.2537809

who receive requirements from a set of users, can decide which servers will process their requirements to minimize response time. Accordingly, the provider, who act as a leader who provisions resources to scheduler agents, can determine how to configure its own servers to reduce energy consumption.

We formulate the decentralized computation allocation decision making problem among multiple scheduler agents and one provider as a stackelberg game. The problem analyzes the limits of energy saving and strategies for configuring the right set of servers that should be switched on and off (sleep) to economize energy consumption, while maintaining the necessary service performance for applications use the computing resources in DCs.

In particular, we model the problem of minimizing average response time of tasks as a noncooperative game among scheduler agents as they compete with one another in sharing resources. Given the decentralized characteristics of noncooperative game theory, the self-organizing feature of scheduler agents can be added automatically into DCs. Consequently, the heavy burden of complex centralized management can be alleviated.

In addition, a distinct observation made in this study is that the size of users requirements fluctuates periodically over time. Thus, the leader should adjust configuration to react to such changes to realize energy- performance tradeoff. To address this problem, an algorithm is proposed for the leader to determine which resources to switch on or off while simultaneously ensuring that their customers will receive acceptable QoS. In particular, we convert the problem into a binary programming problem to obtain the list of the optimal resource configurations for various load rate conditions in a DC. According to our approaches, the provider only aims to retrieve the right configurations from the stored list depending on the current load rate, rather than solve the problem directly again during runtime. The results of the simulated experiments demonstrate that our approaches are effective and efficient.

The rest of this paper is organized as follows. We first discuss related works in Section 2, and then introduce the system model in Section 3. We provide the formulation of the resource allocation problem and present the detailed description of the proposed algorithms in Sections 4 and 5, respectively. We discuss the simulation results that demonstrate the effectiveness of our approach in Section 6. Finally conclusions are drawn in Section 7.

2 Related Works

According to Moore's law, power consumption in computer systems has increased at an exponential rate for decades. The increasing energy consumption will cause severe economic, ecological, environmental, and technical problems. Developing high-performance and energy-efficient computing systems and DCs has gained increasing interest and importance.

Intuitively, the operator would provision less server or network resources to save power. In practice, however, a trade-off exists between energy saving and scheduling delay. Considering this trade-off, numerous studies on strategies or enabling technologies that aim to optimize power-performance tradeoff have been conducted, and an explosively growing body of literature has been developed for energy-efficient computing [3], [4], [5], [6], [7].

In recent years, numerous dynamic capacity provisioning (DCP) schemes, which actively or passively adapt the provisioning of servers based on current or predicted future loads, have been widely utilized in the power-performance trade-off of DCs. Loads can be typically consolidated to fewer servers during off-peak hours, and unused servers can be temporarily disabled by conducting specific actions, such as entering sleep mode. In [8], [9], [10], [11], [12] , such strategies were adopted to reduce power consumption. [8], [9] proposed a set of heuristics that aimed to conserver power by dynamically reallocating virtual machines (VM) and switching off idle ones. Decisions were made according to a set of fixed thresholds that indicated VM utilization. the authors exploited request admission control and bounds on CPU utilizations to prevent SLA from being violated. In [11], Xiao et al. presented a system that used virtualization technology to allocate DC resources dynamically based on application demands by optimizing the number of servers used. They also designed a load predictive algorithm that could accurately capture future resource usages of applications without looking inside VMs. Following the same line of research, [10], [12] devised online control algorithms based on predictive technology and then dynamically adjusted the number of machines to minimize total energy consumption and scheduling delay according to the predictive results at runtime. Their works have generally focused on predicting the future request of an end-user, and on using centralized strategies to implement the allocation of incoming content requests and turning machines on and off. These previous studies have achieved significant progress in improving DC energy efficiency. However they increase the complexity of the resource management of DCs because of the centralized nature of control algorithms that have been adopted in DCs. Furthermore, these studies have given minimal attention on efficiently allocating workload among DC servers to achieve optimal resource utilization, and consequently, minimize mean response time while reducing energy consumption.

Among studies on saving power, the dynamic voltage and frequency scaling (DVFS) technology has played an important role in power management schemes for DCs. Technology is capable of dynamically adjusting the operating voltage or frequency of a CPU according to its load with respect to a number of discrete operating points. In this manner, the power consumption of a CPU will change according to its load. Numerous studies on DVFS in DCs are available [1], [3], [4], [13], [14], [15]. For example, the authors of [13], [14], [15] explored power-performance tradeoff by fixing one factor and minimizing another, from the perspective of optimal load distribution, that is, power-constrained performance optimization and performance-constrained power optimization across DCs. However, such technologies have two major drawbacks. First, they are designed to remain at the component level, such as, an individual CPU or a cooling fan. However, CPUs no longer dominate server power consumption like they used to. Hence, the effectiveness of DVFS has decreased. Second, although they introduce partial energy proportionality to

the servers, the idle power consumption of a DVFS-enabled server remains considerably higher than the value when the server is in sleep mode [1], [3], [5].

To overcome the aforementioned drawbacks, dynamic provisioning, request management and DVFS have been jointly employed [16], [17], [18], [19], [20]. These schemes follow similar power saving strategies. Future load is initially estimated, and then the servers are dynamically provisioned. Accordingly, minimal yet sufficient amounts of servers are activated to process the presented requests, whereas the remaining ones are either turned off or put in sleep mode. Meanwhile, requests are consolidated and mapped onto DVFS-enabled active servers. Although these schemes exhibited significant improvements in saving energy, they still gave insufficient focus on processing performance.

Recently, game theoretic approaches which are employed to manage DC resources, have gained increasing attention in both the industrial world and the academe. The authors of [21], [22], [23], [24] formulated the load balancing problem in heterogeneous distributed systems as a noncooperative game or cooperative game among users. Several mechanisms based on game theory were employed to balance the loads of the distributed systems and to minimize the response time. [25] formulated the resource allocation game under the proportional-share mechanism and studied the efficiency and fairness of the equilibrium in this game. Evidently, these schemes focus on improving processing performance or efficiency without considering energy consumption. In [26], León et al. modeled the problem of minimizing energy consumption while allocating resources to networked applications as a Stackelberg leadership game in DCs. They also developed strategies for determining the right set of computing nodes that should be switched on and off to minimize energy consumption, while maintaining the right level of service. Their scheme considered the trade-off between workload allocation efficiency and energy consumption, and adopted the decentralized allocation strategy to alleviate the burden of system management. However their scheme disregarded the heterogeneity of performance-energy trade-off in servers and overall processing performance in DCs.

Bi-level programming was originally described as a Stackelberg game [34], [40] and was later generalized by several researchers. An important feature of this game is the hierarchical relationship between two autonomous, and possibly conflicting, decision makers. To date, the Stackelberg game has been extensively studied and adopted in various fields, such as in congestion control, revenue maximization, network design, and cooperative transmission [35], [36], [37], [38], [39], [41]. These fields employed a noncooperative or a cooperative game framework among followers according to the selfish interests of individual followers, which reach Nash equilibrium relative to the strategy of the leader. In this manner, a multi-agent system can be granted a degree of autonomy which helps improve reliability and reduce the complexity of centralized management in practice. From the preceding discussions, the Stackelberg game appears appropriate for resource management and optimization in DCs.

TABLE 1
Notations

Symbol	Meaning
A	Set of scheduler agents
N	Set of severs
n	Number of scheduler agents
m	Number of servers
i	Subscript of scheduler agents
j	Subscript of servers
A_i	Scheduler agent i
N_j	Server j
λ_i	Average arrival rate at scheduler agent i
λ_j	Average arrival rate at server j
λ	Total arrival rate of tasks
μ_j	Average process rate at server j
μ_j^i	Available serving rates of server j for scheduler agent i
ρ_{ij}	Probability of scheduler agent i to dispatch a task to server j
ρ_i	Assigned probability vector of scheduler agent i
p^{busy}	Dynamic power of server
p^{idle}	Static power of server
P_j	Utilization power of server j
β_j	Utilization of server j
T_{QoS}	QoS negotiated between users and operator
α	Energy price
γ	Partition granularity of the aggregate power of the system
b_j	State of server j, 1 is active, and 0 is off or sleep
k	Fraction of the total power partitioned by γ
l	Available number of the servers

3 System Model

We introduce the system model in this section. For the convenience of the readers, the major notations used in this paper are listed in Table 1.

3.1 DC Model

Following previous works, decentralized strategies were adopted in our model to improve the reliability of the system and to prevent access bottleneck. In such manner, when failures arise on an access point, other normal access points can also provide services to users. As shown in the Fig. 1, in DCs, there are a set of heterogeneous servers connected by an underlying communication network.

We assume that the system consists of some users, n scheduler agents, m servers and a system monitor. The set of scheduler agents is denoted by a vector $A = \{A_1, A_2, \ldots, A_n\}$. The set of servers is denoted by a vector $N = \{N_1, N_2, \ldots, N_m\}$. Without losing generality, the number of users will more than the number of scheduler agents in the system.

With the preceding classification, we can model a DC using the hierarchical relationship between users, scheduler agents, and system monitor. Each user may send tasks to a scheduler agent or to more than one scheduler for processing. Meanwhile, a scheduler agent receives tasks from a set of users and decides which servers will process these tasks. The system monitor responds to perform the provisioning of service by turning the servers on and off in DC. The monitor also periodically releases the information of each server to scheduler agents, and thus, scheduler agents can decide which servers will execute their tasks, while minimizing response time.

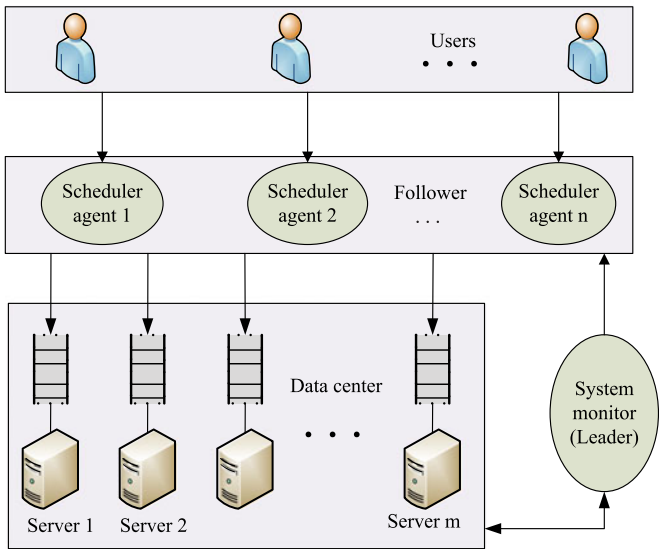


Fig. 1. DC model.

Workload heterogeneity can significantly affect on resource allocation decisions. Hence, numerous studies [7], [42], [44], [45] have discussed the problem from different perspectives. Similar to literature [13], [14], [43], [45], we use the exponential distribution function to model the service time of user tasks. Each user generates tasks in terms of a Poisson process and independently of other users. The task execution requirements (measured by the number of instructions to be executed) are independent and identically distributed (i.i.d.) exponential random variables r with mean $\overline{r}$. Each server is modeled as an $M/M/1$ queuing system and maintains a queue with infinite capacity for waiting tasks. The first-come-first-served queuing discipline is adopted. A server has execution speed s (measured by the number of instructions executed in one unit of time), and the execution times on the server are i.i.d. exponential random variables $x = r/s$ with mean $\overline{x} = \overline{r}/s$. Hence, the average service rate of the server, that is, the average number of service requests can be completed by the server in one unit of time, and can be denoted by $\mu = 1/\overline{x}$.

Let λ_j denote the arrival rate being sent by scheduler agents at server j that executes the tasks at an average service rate μ_j. According to Little's law, the expected response time at server j is denoted by $l(\lambda_j)$ and given by

$$l(\lambda_j) = \frac{1}{\mu_j - \lambda_j}. \tag{1}$$

For stability, we generally use the condition in which tasks must not be generated faster than the system can process them. Therefore, the inequality $\sum_{j=1}^{m} \lambda_j < \sum_{j=1}^{m} \mu_j$ must hold; otherwise, the queues at the servers will build up to infinity and the expected response time will be infinite.

Notably, the service time of the tasks with general distributions is considered, and thus, the $M/G/1$ queuing model is appropriate for those tasks. However, a simple closed-form solution for the $M/G/1$ that will help us to understand the effect of task size variability on mean response time remains unavailable [46].

3.2 Power Model

Ideally, the power consumption of a server should be proportional to its workload to realize power saving. That is, no power should be consumed by an idle server. However, such an objective remains theoretical because the current trend in performance-driven hardware design is the opposite; difficulties are expected in convincing hardware vendors to shift to energy-driven design.

In a DC, the energy curve of a server can be characterized in terms of server workload by

$$P = p^{idle} + \beta p^{busy}, \tag{2}$$

where β is the utilization of a server, and its value is the ratio of the number of tasks accepted and the processing capacity at a server; β is formulated by $\beta = \lambda/\mu$. p^{idle} represents the static power of a server, and is a fixed cost in maintaining a server that is powered on and ready to perform work; and p^{busy} is the dynamic fraction of the power consumed when a server is fully utilized.

From the previous representations, several technologies or strategies, such as DVFS and DCP, are employed to reduce energy consumption in DCs. However, given that idle power consumption is still existent, only partial energy proportional is delivered with such technologies. From this fact, idle consumption is not negligible in DCs.

Hence we argue that high saving can be achieved in large-scale infrastructure if we consider the dynamic provisioning problem of servers with the scheduling decisions based on the current or predicted future workload.

4 Allocation Strategy of Scheduler Agent

Following the discussion in previous sections, the scheduler agents are regarded as the followers who respond to accommodate the tasks generated by users. Without losing generality, all the scheduler agents are considered selfish individuals, and the strategies they used are intended to maximize their own utility. In such cases, the problem faced by scheduler agent A_i is to decide on how to assign tasks to the servers such that each server will operate optimally.

4.1 Task Allocation Formulation

Given that the decentralized strategy is adopted in our model, we assume that each scheduler agent distribute its tasks to the servers independently depending on a probability profile. Thus agent A_i should find probability ρ_{ij} assigned tasks to server j to minimize the expected execution time of its tasks. The probability of agent A_i is denoted by vector $\rho_i = \{\rho_{i1}, \rho_{i2}, \ldots, \rho_{im}\}$, and $\sum_{j=1}^{m} \rho_{ij} = 1$. We assume that the arrival rate of scheduler agent A_i is denoted by λ_i. Once ρ_{ij} is determined, scheduler agent i will send tasks to server j at a rate λ_{ij} given by $\lambda_{ij} = \lambda_i \rho_{ij}$; that is, the rate that is equal to the aggregated tasks of scheduler agent A_i multiplied by the allocation probability for server j. Moreover, the available processing capacity of each server for scheduler agent A_i is denoted by vector $S_i = \{\mu_1^i, \mu_2^i, \ldots, \mu_m^i\}$, where $\mu_j^i = \mu_j - \sum_{k=1,k\neq i}^{n} \rho_{kj}\lambda_k$ indicates the available processing rate at server j for scheduler agent i .

Hence, the overall expected responded time at scheduler agent A_i is given by

$$T(\rho_i) = \sum_{j=1}^{m} \frac{\rho_{ij}}{(\mu_j^i - \rho_{ij}\lambda_i)}. \quad (3)$$

Under the previous assumption, scheduler agents are selfish and contest the sharing of computation resources provisioned by the system monitor. The decision of scheduler agent A_i depends on the strategy profile of other scheduler agents, and thus, we formulate the preceding problem as a decentralized game among scheduler agents. The vector $\rho = \{\rho_1, \rho_2, \ldots, \rho_n\}$ is called the strategy profile of the task allocation game.

Definition 4.1. *The decentralized tasks allocation game is composed of a set of players, a set of strategies, and the best response over a set of strategy profiles:*

1) *Players: n scheduler agents.*
2) *Strategies: feasible tasks allocation strategies of each scheduler agent.*
3) *Best response: each scheduler agent prefers strategy profile ρ to strategy profile ρ' if and only if $T(\rho) < T(\rho')$.*

Evidently, the most commonly used solution concept for such games is that of the Nash equilibrium that we consider in this study.

Definition 4.2. *A strategy profile is a Nash equilibrium of the decentralized task allocation game if no player at equilibrium ρ_* can further reduce its overall response time by unilaterally changing its strategy; that is,*

$$T(\rho_i^*, \rho_{-i}^*) < T(\rho_i, \rho_{-i}^*).$$

The Nash equilibrium has a beneficial self-stability property, such that the scheduler agents at equilibrium can achieve a mutually satisfactory solution and no agent has the incentive to deviate. Furthermore, given that the overall expected response time functions are continuous, convex and increasing, a unique Nash equilibrium exists in our task allocation game [28].

In summary, all the scheduler agents play the best response strategies toward one another at the Nash equilibrium. Accordingly, the overall expected responded time is minimized. Hence, given a set of the available processing rates at each server, the best response of an agent is to find the distributed probability of a set of tasks. In particular, the best response is the solution for the following optimization problem (labeled P1):

$$Minimize \;\; T(\rho_i) = \sum_{j=1}^{m} \frac{\rho_{ij}}{(\mu_j^i - \rho_{ij}\lambda_i)}, \quad (4)$$

subject to the constraints

$$\sum\nolimits_{j=1}^{m} \rho_{ij} = 1, \quad (5)$$

$$\rho_{ij} \geq 0, \quad (6)$$

$$\sum\nolimits_{i=1}^{n} \rho_{ij}\lambda_i < \mu_j. \quad (7)$$

Equation (5) represents the conservation constraint, which ensures that the sum of tasks assigned to each server by scheduler agent A_i is equal to its total tasks arriving within a tasks allocation period. Constraint (6) assures that the distribution probability of all tasks always satisfies nonnegativity. Finally, Constraint (7) guarantees the stability of servers, given that the response time of the tasks queuing to run will be infinite if the available processing rate is lower than the arrival rate at a server according to queuing theory.

4.2 Follower Best Response Algorithm

In this study, we assume that the Constraint (7) is always satisfied at the Nash equilibrium because the total arrival rate does not exceed the total processing rate of the system. Accordingly, Constraint (7) can be ignored. Constraints (5) and (6) have continuous first partial derivatives and $T(\rho_i)$ is a convex function in ρ_{ij}. This information implies that the first-order Kuhn-Tucker conditions are necessary and sufficient for optimality. Hence, problem P1 can be translated into a Lagrangian with Lagrange multipliers and can be deduced toward the optimal solution.

In [21], an algorithm called BEST-REPLY was presented to address similar problems efficiently. In our work, we directly use this algorithm to compute the best response of a follower. For the convenience of the readers, the complete details of the algorithm are provided (labeled Algorithm 1 in this paper).

Algorithm 1. BEST-REPLY(S_i, λ_i)[21]: the best response algorithm for scheduler agent i

Require:
Available serving rates of each server for scheduler agent i: $S_i = \{\mu_1^i, \mu_2^i, \ldots, \mu_m^i\}$.
The total tasks arrival rate of scheduler agent i: λ_i.
Ensure:
$\{\rho_{i1}, \rho_{i2} \ldots, \rho_{im}\}$, the tasks allocation probability of scheduler agent i.

```
Sort the servers in decreasing order of their available processing rates, μ_1^i ≥ μ_2^i ≥ ··· ≥ μ_m^i.
k ← m
while (Σ_{j=1}^{k-1} μ_j^i − λ_i) / √(Σ_{j=1}^{k-1} μ_j^i) ≥ √μ_k^i do
    k ← k − 1
end while
t ← (Σ_{j=1}^{k} μ_j^i − λ_i) / √(Σ_{j=1}^{k} μ_j^i)
for j = 1 to m do
    if j ≤ k then
        ρ_ij ← (μ_j^i − t√μ_j^i) / λ_i
    else
        ρ_ij ← 0
    end if
end for
return {ρ_i1, ρ_i2 ..., ρ_im}
```

In our work, the Algorithm 1 is distributed in the sense that each scheduler agent searches the set of distribution probability that minimizes the overall response time, when the number of servers provisioned by the system monitor has been determined. Following an iterative process of updating the vector of the distribution probability, the preceding algorithm will converge to an efficient equilibrium, and then the optimal distribution probability set of each scheduler agent can be obtained. In this manner, we can draw an effective conclusion, which is described in the following paragraphs.

Theorem 4.1. *At equilibrium, servers to which tasks are assigned to be executed will have the same average response time if each scheduler agent uses the Algorithm 1 to distribute tasks on the servers.*

Proof. Let $\theta \geq 0, \eta_j \geq 0,\ j = 1, 2, \ldots, m$ denote the Lagrange multipliers. The Lagrangian of problem P1 is given as follows:

$$L(\rho_{i1}, \ldots, \rho_{im}, \theta, \eta_1, \ldots, \eta_m) = \sum_{j=1}^{m} \frac{\rho_{ij}}{(\mu_j^i - \rho_{ij}\lambda_i)} - \theta\Big(\sum\nolimits_{j=1}^{m} \rho_{ij} - 1\Big) - \sum\nolimits_{j=1}^{m} \eta_j \rho_{ij}.$$

The first-order Kuhn-Tucker conditions and constraints are given as follows:

$$\partial L/\partial \rho_{ij} = \frac{\mu_j^i}{(\mu_j^i - \rho_{ij}\lambda_i)^2} - \theta - \eta_j = 0,$$

subject to the constraints

$$\partial L/\partial \theta = \sum\nolimits_{j=1}^{m} \rho_{ij} - 1 = 0,$$

$$\eta_j \rho_{ij} = 0, \quad \eta_j \geq \rho_{ij} \geq 0, j = 1, \ldots, m.$$

Hence, the following result can be achieved with respect to Lagrange multiplier θ:

$$\frac{\mu_j^i}{\big(\mu_j^i - \rho_{ij}\lambda_i\big)^2} \begin{cases} = \theta, & if\ \rho_{ij} > 0 \\ \geq \theta, & if\ \rho_{ij} = 0. \end{cases}$$

The first item on the left side of the preceding expression is the first partial derivative of Equation (3) for ρ_{ij}. This item implies that each scheduler agent has the same marginal value on machines where all scheduler agents place tasks but has higher marginal value on machines where all scheduler agents do not distributed any tasks at equilibrium. The aforementioned expression will be $1/\mu_j^i$ if $\rho_{ij} = 0$; that is, the available processing rate $\mu_j^i = \mu_j - \sum_{k=1,k\neq i}^{n} \rho_{kj}\lambda_k$ at server j for scheduler agent i is equal to the reciprocal of the response time of server j before scheduler agent i assign any task to server j. Therefore, by intuitively considering Algorithm 1, the server with higher available processing rate will have the priority to be assigned tasks until its available processing rate is lower than that of the others. Consequently, servers that are assigned with tasks will provision the same available processing rate for any scheduler agent; that is, they will have the same average response time (not exceeding constant θ) when the system reaches equilibrium.

This remark completes the proof.

5 Energy Management of DCs

In our model, the system monitor is regarded as the leader who has the privilege to move first and is expected to maximize its profit by operating DCs to provision computation service for users. Evidently, reducing energy consumption can improve the profit of the leader. Hence, the system monitor can consider a resources provisioning strategy that utilizes the fewest servers to guarantee the SLA while putting the unused ones to sleep or off mode.

5.1 Energy Optimization Formulation

With the improvement of the management level of DCs, energy consumption will occupy a higher proportion of the entire operating cost of DC. Hence, the energy saving is the chief objective considered by us, whereas other factors are disregarded. The primary difficulty of the system monitor is how to determine which resources should be switched on and off while provisioning an acceptable QoS to their customers. In particular, the challenge is finding the solution for the following optimization problem:

$$MaximizeP(b_1, b_2, \ldots, b_m) = rev - \sum\nolimits_{j=1}^{m} \alpha b_j \big(P_j^{idle} + \beta_j P_j^{busy}\big), \tag{8}$$

where rev denotes the total revenue of the system; α is the per unit price of energy; and b_j is a binary variable, which indicates the status of the jth server. Notably, a value of 1 indicates "on", whereas other value indicates "off".

From theorem 1, we learn that the average response times of the available servers are ideally the same. In addition, the utilization of these servers will be increased as much as possible by the provider unless their response time exceeds the acceptable QoS to reduce energy consumption. In particular, the average response time of each available server approximates $T_{QoS} = 1/(\mu_j - \lambda_j)$ in extreme cases. Thus the arrival rate in server j is decided by $\lambda_j = \mu_j - 1/T_{QoS}$, the utilization of the server can be deduced as

$$\beta_j = \frac{\lambda_j}{\mu_j} = \frac{\mu_j - 1/T_{QoS}}{\mu_j} = 1 - \frac{1}{\mu_j T_{QoS}}. \tag{9}$$

We also learn that the per unit price of energy α is a constant and the revenue of the system is fixed because the expected number of arrival tasks are constant during a running period. Hence, the problem can be translated into deciding the value of each b_j. that is, the problem is to select which servers to run to maximize profit while maintaining SLA. This problem can be formulated by the following optimization equation (labeled P2):

$$Minimize\ P(b_1, b_2, \ldots, b_m) = \sum\nolimits_{j=1}^{m} b_j \big(P_j^{idle} + \beta_j P_j^{busy}\big), \tag{10}$$

subject to the constraints:

$$l(\lambda_j) \leq T_{QoS}, \quad \forall j \in \{1, 2, \ldots, m\}, \tag{11}$$

$$b_j \in \{0, 1\}, \quad \forall j \in \{1, 2, \ldots, m\}. \tag{12}$$

The performance guarantee for users is set by these constraints, which promise that the QoS approved by users and DC operator are not violated.

5.2 Energy Optimization Algorithm

The processing rate and power of all the servers in a DC are different from one another because of the heterogeneity of their capability and performance. Hence, directly resolving problem P2, which is an NP-hard problem, is difficult.

However, the essence of problem P2 is to determine an appropriate combination of servers to minimize energy consumption. We are inspired by dynamic programming that tackles similar problems. Accordingly, problem P2 can be converted into an alternative optimization problem that addresses the maximization of the processing rate for every available power given by the distinct proportion of aggregate power of the system. The following paragraphs provide a full representation of our routines.

First, on the basis of queuing theory, the Constraint (11) of problem P2 can be converted into an alternative approximate representation denoted by $\sum_{j=1}^{m} b_j\mu_j \geq \lambda + \sum_{j=1}^{m} b_j/T_{QoS}$, where λ is the total arrival rate of all the tasks. Then, the β_j in Equation (10) is substituted into Equation (9). The constraint of problem P2 is used as the objective, and the original objective becomes the constraint. The formulation is presented as follows (labeled P3):

$$Maximize \sum_{j=1}^{m} b_j\mu_j, \tag{13}$$

subject to the constraints:

$$\sum_{j=1}^{m} b_j\left(P_j^{idle} + \left(1 - \frac{1}{\mu_j T_{QoS}}\right)P_j^{busy}\right) = k\,, \tag{14}$$

$$k \in \left\{\frac{1}{\gamma}\sum_{j=1}^{m} P_j, \frac{2}{\gamma}\sum_{j=1}^{m} P_j, \ldots, \frac{\gamma}{\gamma}\sum_{j=1}^{m} P_j\right\}, \gamma \in \mathbb{N}, \tag{15}$$

$$b_j \in \{0,1\}, \quad \forall j \in \{1,2,\ldots,m\}, \tag{16}$$

where γ is the partition granularity of the aggregate utilization power of the system, the P_j denotes the utilization power of server j and is decided by equations (2) and (9), and k is one of the powers partitioned by γ in a DC. For example, the aggregate utilization power is 9 and the γ is set to 3, then the possible values of k are 3, 6 and 9.

Considering that k can have different power values, γ subproblems are clearly presented in P3. That is, the objective of problem P3 is to minimize processing performance for each given power k by determining an appropriate combination of the servers. Following the dynamic programming method [29], we obtain an insight into the problem, as given by the following claim.

Claim 5.1. k^*, which is the optimal value of problem P2, exists. In problem P3, if we let $k = k^*$, then the optimal solution $\{b_1, b_2, \ldots, b_m\}$ for problem P3 must be the optimal solution for problem P2.

Inspired by this conclusion, we first solve the optimal solution of problem P3 for each parameter k. Then, we determine the optimal value of problem P2 in terms of the set of optimal solutions achieved from problem P3. In particular, we search for the least k^*, which guarantees that the optimal value of problem P3 is not lower than $\lambda + \sum_{j=1}^{m} b_j/T_{QoS}$ under $k = k^*$.

The 2-tuple $<k, l>$ defines a subproblem of P3, in which k is a given power value, and $l \in [0, m]$, indicates that the number of available servers in this subproblem is between 0 and l. $M(k, l)$ denotes a set of machines, which is the optimal solution for subproblem $<k, l>$. For example, if server N_j is selected in the optimal solution of subproblem $< k, l >$, then server $N_j \in M(k, l)$ and $b_j = 1$. $s(k, l)$ denotes the optimal value of subproblem $< k, l >$; that is, $s(k, l)$ is the maximal processing rate for subproblem $< k, l >$. Consequently, the relations among them are given by the following descriptions.

If $P_j \leq k$, $M(k - P_j, l - 1)$, is an available solution and $s(k, l-1) < s(k - P_j, l - 1) + \mu_j$, then

$$M(k, l) = M(k - P_j, l - 1) \cup \{N_j\};$$

otherwise,

$$M(k, l) = M(k, l - 1).$$

The detailed procedure for solving the subproblems is summarized by Algorithm 2 who is called $LeaderBestResponse(LBR)$.

Algorithm 2. LeaderBestResponse(P, μ): Leader's best response algorithm

Require:
Serving rates of each server: $\mu = \{\mu_1, \mu_2, \ldots, \mu_m\}$.
The utilization power of each server: $P = \{P_1, P_2, \ldots, P_m\}$.
Ensure:
the set of solution of all sub problems: M.
the optimal value of all sub problems: s.

$M \leftarrow \phi\,,\ \ s \leftarrow 0$
$\gamma \leftarrow power\ partition\ granularity$
$Power \leftarrow \sum_{j=1}^{m} P_j$
for $j = 1$ *to* m **do**
  $k \leftarrow P_j$
  while $k \leq Power$ **do**
    if $s(k, j-1) < s(k - P_j, j - 1) + \mu_j$ **then**
      $s(k, j) \leftarrow s(k - P_j, j - 1) + \mu_j$
      $M(k, j) \leftarrow M(k - P_j, j - 1) \cup \{N_j\}$
    else
      $s(k, j) \leftarrow s(k, j - 1)$
      $M(k, l) \leftarrow M(k, l - 1)$
    end if
  end while
  $k \leftarrow k + Power/\gamma$
end for
return M, s

The computation complexity of this algorithm is $O(m\gamma)$. Hence, the overhead of this algorithm is acceptable. After executing this algorithm, two results can be obtained. One is M, which comprises the optimal configuration of all the subproblems; and the other is s, which consists of the optimal value of all the subproblems.

5.3 Stackelberg Game Algorithm

As mentioned earlier, the result of executing the LBR algorithm is only influenced by power and the available processing rate of each server in a DC. Moreover, the operators cannot frequently change the physical structure of DCs; otherwise they will have no capability to receive any benefit from their investments. Therefore, the LBR algorithm does not need to be executed repeatedly. This algorithm will only need to be executed again when the physical composition of the system is changed to update the optimal solution and the value. Hence, the result of the algorithm can be stored in a disk after completing the update.

The system monitor aims to retrieve only the result list of the algorithm to determine the optimal solution during runtime.

After obtaining the optimal result of each subproblem, the next step is to decide which solution of which subproblem is the best configuration scheme under the current operating environment. When the previous steps have all been completed, the leader advertises its optimal configuration to all its followers. Sequentially, the followers employ the Algorithm 1 to distribute their tasks according to the information. Algorithm 3 summarizes the tasks allocation process between the leader and the followers through the stackelberg game approach.

Algorithm 3. Stackelberggame(P, μ)

Require:
Serving rates of each server: $\mu = \{\mu_1, \mu_2, \ldots, \mu_m\}$.
The utilization power of each server: $P = \{P_1, P_2, \ldots, P_m\}$.
The appointed average responsible time: T_{QoS}.

1: $M, s \leftarrow LeaderBestResponse(P, \mu)$
2: Obtain the aggregate arrival rates of all tasks: λ
3: $Power \leftarrow \sum_{j=1}^{m} P_j$
4: $k \leftarrow Power/\gamma$
5: $num \leftarrow |M(k,m)|$ the number of servers
6: **while** $s(k,m) \leq \lambda + num/T_{QoS}$ **do**
7: $k \leftarrow k + Power/\gamma$
8: $num \leftarrow |M(k,m)|$ the number of servers
9: **end while**
10: Configure the system by $M(k,m)$ and advertise those information to all scheduler agents
11: **repeat**
12: **for** all scheduler agent i **do**
13: Obtain $S_i\{\mu_1^i, \mu_2^i, \ldots, \mu_m^i\}$ by inspecting each machine ($\mu_j^i \leftarrow \mu_j - \sum_{k=1,k\neq i}^{n} \rho_{kj}\lambda_k$)
14: $\rho_i \leftarrow BEST - REPLY(S_i, \lambda_i)$
15: Set the tasks assigned probability of scheduler agent i by ρ_i
16: **end for**
17: **until** convergence

Once given the overall arrival rate of tasks and the maximum average response time negotiated by the users and the provider in a DC, the optimal configuration scheme of the system can be derived by retrieving the result list of the LBR algorithm. Then the servers that are out of the scheme will be put in off or sleep mode to reduce the energy consumption of the system. The overhead of Algorithm 3 mainly focuses on the computation of the distribution probability, which is the allocation strategy profile determined by scheduler agents for the tasks received from users. Therefore, considering the external loop, the computation complexity of the $StackelbergGame$ algorithm is $O(nm\log(m))$. After several iterations, this algorithm will converge to an efficient equilibrium, and then terminates.

In general, the arrival rate of tasks can fluctuate over time, so we need to employ a suitable prediction model to forecast the future requirement. Some predicting models, such as the autoregressive model and the autoregressive integrated moving average model have been demonstrated to be effective in predicting workload arrival rate [12], [30]. Accordingly, the models can be adopted to predict the workload arrival rate. Furthermore, we ignore the cost in time of moving tasks between servers. Recent studies [31], [32] on live migration in several VMs exhibit migration latencies at the millisecond level, which supports our decision to disregard this cost.

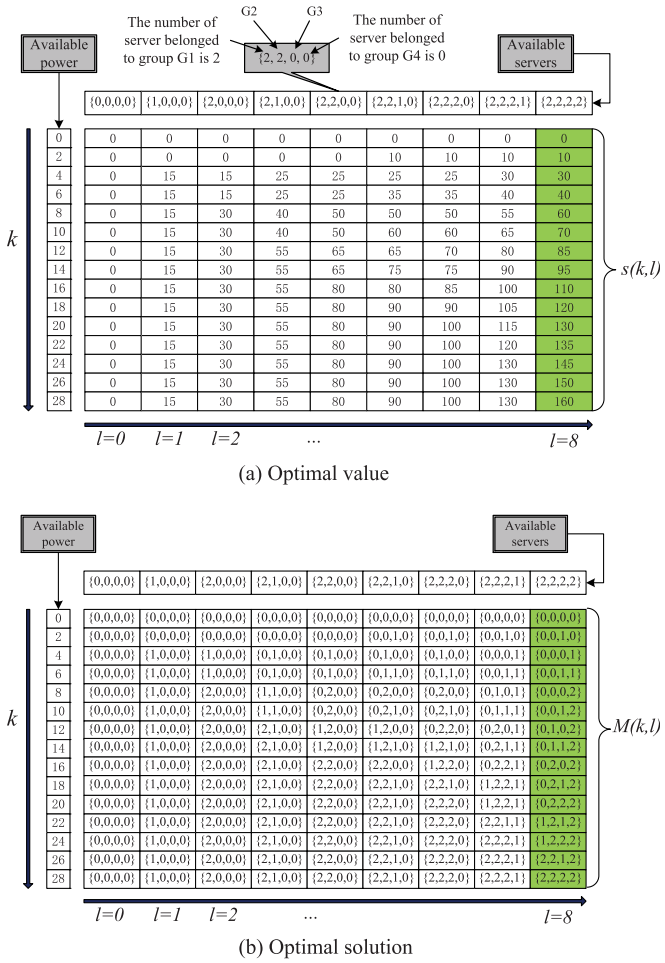


Fig. 2. Example for our methods.

5.4 Example

To help understand our methods, an example is presented in this section. We assume that the appointed average response time T_{QoS} is less than 0.5 time unit. Let 3-tuple $G = < c, p, q >$ define a group in which servers have same parameters c , p, and q, which denotes the service rate, utilization power and quantity of servers, respectively. We assume that four available groups exist: $G1 = <15, 4, 2>$, $G2 = <25, 4, 2>$, $G3 = <10, 2, 2>$, and $G4 = <30, 4, 2>$. Therefore, the aggregate processing capability is 160 units, and the total utilization power and number of servers is 28 units and 8, respectively. The partition granularity γ is set to 14.

The executing results of Algorithm 2 in case the aforementioned parameters are used are provided in Fig. 2. In Fig. 2a, the first row represents available server combinations, in which the server is jointed with the system to provision service for the users in turn. The first column denotes the available power divided by the partition granularity γ; and each cell of the other columns denotes an optimal value $s(k,l)$, which corresponds a given power k and the number of available servers l. In the last column, the available

TABLE 2
Servers Configurations

Model	Service rate (tasks/s)	Idle power (watts)	Full power (watts)	Num of machines
Type1	82	49.6	272	100
Type2	10	15.9	45.1	500
Type3	57	122	580	150
Type4	14	63.2	236	500

servers are set to {2,2,2,2}, which implies that all the servers are available and that each cell of the column highlighted in green denotes a final optimal value associated with a given power. For example, when a given power k is 24 or 12, the maximal service capabilities provisioned by the system are 145 and 85, respectively.

Similarly, Fig. 2b represents the optimal solutions $M(k,l)$ under different given power k and the number of available servers l with Algorithm 2. The last column highlighted in green describes the optimal solution in the case of all servers being available; that is, this column records final provisioning strategies. For example, when a given power k is 24 or 12, the optimal combinations of the servers provisioned by the system are $M(24,8)=\{2,2,1,2\}$ and $M(12,8)=\{0,1,0,2\}$, respectively.

In the following, the Algorithm 3 is performed using the results of Algorithm 2. We assume that the aggregate arrival rate of tasks is 100 units. Accordingly Algorithm 3 can determine that the optimal solution is $M(16,8)=\{0,2,0,2\}$ because $s(16,8)=110$ depending on the results of Algorithm 2 is exactly higher than $\lambda+|M(16,8)|/T_{QoS}=100+4/0.5$. Then the system monitor configures the system through $M(16,8)$. Tasks executed on the servers that will be set to sleep mode or will be turned off are migrated to other servers that are kept active in the next period. Then, the scheduler agents can obtain the information of available servers advertised by the system monitor and can distribute their tasks on the servers using Algorithm 1.

6 Evaluation

In this section, we evaluate the performance of our methods under various scenarios and compare it with alternative approaches.

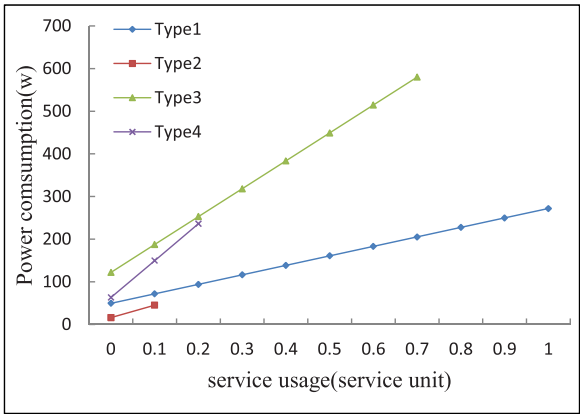


Fig. 3. Server energy consumption.

TABLE 3
Scaling factors f_i for Each Scheduler Agent

schedule agents	1	2-3	4-6	7-10	11-12
f_i	0.2	0.12	0.02	0.1	0.5

6.1 Configurations

While performing our experiment on a large-scale real-life DC, several servers will be switched on or off over time. Consequently, uncertain risks that may result in the instability and may cause enormous economic losses for both provider and other users are present. Therefore, we simulate a heterogeneous DC composed of a mixture of servers from multiple manufacturers and models in our experiments. Their parameters are elaborated as follows.

The system consists of 1250 simulated servers that reflect the characteristics of four types of servers: QuantaGrid D51B-2U, NEC Express5800, IBM x3755 M3, and Dell PowerEdge R610. The parameters of these servers are obtained from the SPECpower_ssj2008 benchmark, which is available to the public through the Standard Performance Evaluation Corporation website [33]. These machines are manufactured by various enterprises using different technologies in the last five years.

We present the characteristics of the simulated servers in Table 2. The five sets of parameters of the servers are listed in this table. The second column provides the processing rate of the simulated servers. The third column shows the idle power of the servers. The forth column indicates the full power as the server runs under full utilization. The last column provides the number of each type of server. A substantial difference exists in the ratio of the energy to the performance of each servers. Fig. 3 shows energy consumption as a function of server usage. It illustrates the importance of considering machine heterogeneity when scheduling tasks to reduce energy consumption.

A total of 12 scheduler agents are present in the system. The total load rate λ is determined by the product of the system utilization and the aggregate processing rate of the system, i.e., the 60 percent total load rate means that the utilization of the system is 60 percent. The arrival rate of each scheduler agent is equal to the product of the scaling factor f_i and the total load rate λ. It is denoted with $\lambda_i=\lambda f_i$, where the scaling factors f_i are provided in Table 3.

In our evaluation experiments, the partition granularity of power γ is decided by

$$\left\lceil \sum_{j=1}^{m} P_j/5 \right\rceil,$$

where 5 is the approximate value of the GCD of the utilization power of the four types of servers. In addition, our method is labeled as STG, and several other methods are implemented for comparison, as follows: (1) a none power-aware algorithm (called NPA), which turns on all the servers during runtime and assigns tasks to each server in proportion to its processing rate, is primarily used as a lower bound for energy saving; (2) the DVFS method discussed in [13], [14] (called DVFS); (3) the DCP scheme shown in [10], [11] (denoted as DCP), which always provides machines in a greedy manner by turning them on in descending order of

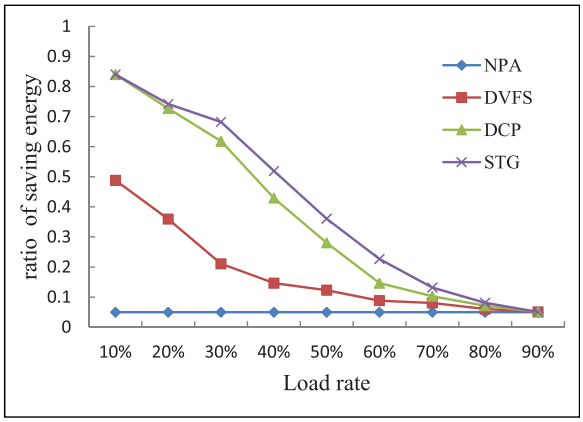


Fig. 4. Ratio of energy saving versus load rate.

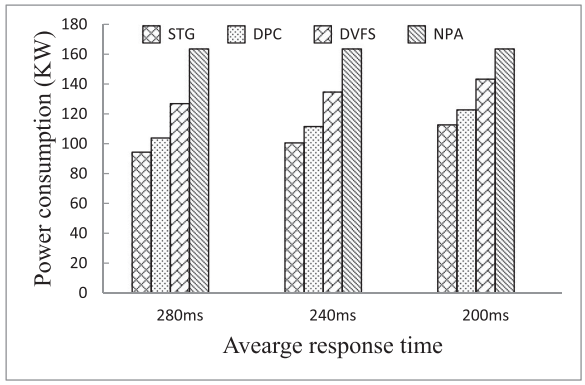


Fig. 5. Average response time versus energy.

energy efficiency (e.g., always turning on Type1 machines first). To simplify the evaluation, we ignore uncompleted tasks and communication delays in the system while making decisions.

6.2 Effect of Load Rate

An important issue is the influence of load rate on energy saving. To study this issue we vary the total load rate of the system from 10 to 90 percent on a simulated heterogeneous distributed system. Then, we perform experiments to measure the effect of energy consumption with NPA, DVFS, DCP, and STG. In the experiment, the parameters of the simulated servers are provided by previous configurations, and the maximal average response time is set to 280 ms. For easy comparison, we set the evaluation metric of energy saving as the ratio of the actual power consumption between these schemes and the NPA scheme. this ratio is characterized by

$$1 - P_{actual}/P_{NPA},$$

where the P_{actual} represents the actual power consumption of each scheme, and the P_{NPA} is viewed as the benchmark power consumed by NPA scheme, thus the higher value means more energy saving.

The Fig. 4 shows the results of the experiment, when the load rate of the system is varied from 10 to 90 percent. As expected, given that all the servers are active under the NPA scheme, this scheme incurs the highest energy cost, and can be regarded as the baseline algorithm. In additional, the DCP scheme which provisions resources with respect to the current requirements, is similar to our scheme. Hence, the results of the two schemes are close in the experiment. However, the Fig. 4 illustrates that our method is superior to the DCP scheme, primarily because DCP does not optimize the supply combination of the servers and tasks allocation.

The results presented in the Fig. 4 indicate that DVFS technology can also reduce energy consumption. However, because of the existence of static power, the energy-saving effect of this scheme can only achieve half of the performance of our method at low utilization (load rates range from 10 to 40 percent). Furthermore, when the load rate of the system is higher than 70 percent, more servers are employed to provision service for users to prevent violating the SLA. Therefore, fewer machines can be scaled and the effect of the energy saving gradually narrows down among the four schemes.

6.3 Effect of Average Response Time

Intuitively, in order to promote the performance of the system, the operator has to provision more machines or scale up the voltage or frequency of machines to reduce the average response time of tasks. However, the reduction of the average response time of the tasks and energy saving is conflicting. Therefore, we explore the relationship between energy saving and the average response time in the four scheduling schemes in the section.

In this experiment, the parameters of the simulated servers are the same as those in the previous experiment, and the load rate of the system is given at 50 percent of the total processing capacity of the system. Fig. 5 shows the power requirement for different schemes at three levels of the average response time: 280, 240, and 200 ms. Notably, the four schemes always satisfy the average response time in the three levels. The plot shows the NPA scheme yields almost the same consumption power at three levels of the average response times. The main reason is that NPA always turns on all the servers to provision service for users without any power-aware strategy.

The results presented in Fig. 5 show that the consumption power of the STG, DCP, and DVFS schemes increases with the decrease in average response time. This means that the three schemes supply more computation capacity with the increasing demands of users. Essentially, they only provision so those minimal yet sufficient service rates to prevent the negotiated SLAs from violating for the minimization of energy consumption.

To a certain extent, the strategies of STG and DCP are similar; that is, they generally consolidate the workload to fewer servers, and then put the unused servers in sleep or off mode. However, the STG scheme significantly outperforms DCP and DVFS in our evaluation. Using STG scheme the consumption power is less than using other schemes in all levels of the average response times. The main reason for this result is that STG scheme distributes the tasks with the aid of Algorithm 1, which can achieve the approximate optimal average response time using less number of machines. Futhermore, given that the STG scheme combines machines with Algorithm 2, its energy consumption is less than that of DCP while maintaining the same average response time.

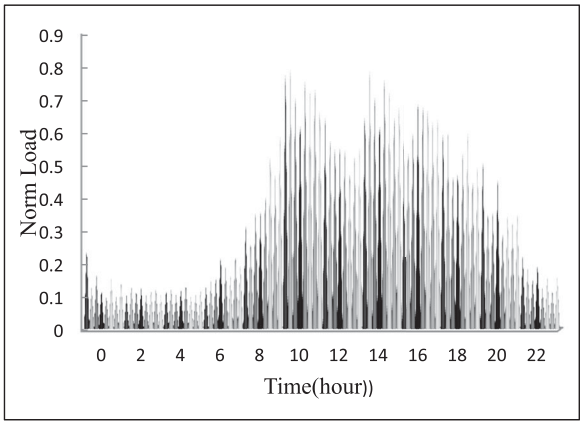


Fig. 6. Workload trace.

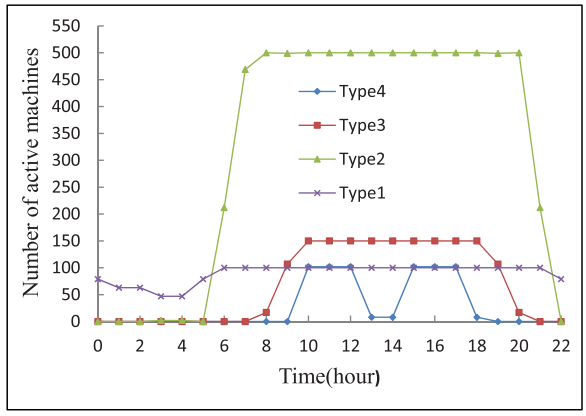


Fig. 8. Number of machines used by STG.

6.4 Effect of Dynamic Load

To validate and evaluate our method further, the statistical data of the actual workload trace are drawn from the China National Supercomputer Center in Changsha. By analyzing and normalizing the statistical data, the final results that reflect the periodic fluctuation of the workload at the center are presented in Fig. 6. In this figure, the maximum load rate is approximated to be 80 percent of the service capacity of the cluster. In this section, we examine the dynamic capability of the four schemes under the simulated situation presented in Fig. 6.

Without losing generality, we assume that the computational resources of the system always satisfy the requirement of the users, and the overall workload of the system also varies periodically in our experiment environment. In addition, the maximum average response time is set to 280 ms. In order to facilitate comparison, we disregard the delays of turning servers on and of the predicting strategies for load fluctuations when using the DCP and STG schemes in our simulated experiments.

In the Fig. 7, the average response time of NPA is the shortest, and the curve of average response time of NPA fluctuates with the change of the system load rate. The main reason is that all the servers of the system are active and ready to provision service for users at any time when operator employs NPA scheme that is a no power-aware strategy.

The results presents in the Fig. 7 show that the curves of DFVS, DCP and STG are close. The experimental results just coincide with our expectation. It reveals that the three schemes do not supply more computation capacity than the demands of the current workload in order to maximize energy saving without violating the negotiated SLAs. Hence, the computation capacity provisioning for users exactly satisfy or approach the maximal average response time in the schemes.

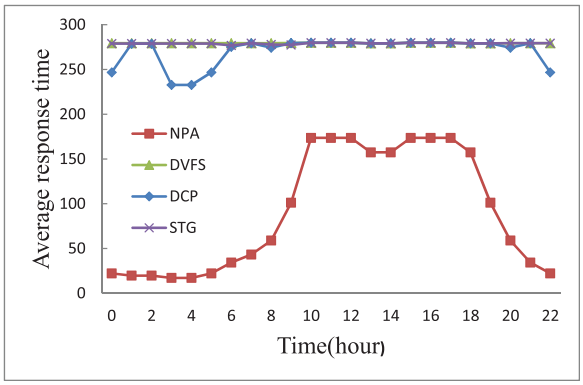


Fig. 7. Average response time.

Specially, the average response time of DCP fluctuates by the change of load rate and is less than that of DVFS and STG when the load rate is relative lower, e.g., the load rate is 20 percent. The reason is that DCP method prefers the machines having higher energy-performance efficiency without considering their power. It ignores the fact that energy-performance efficiency only represents the performance per unit energy rather overall processing rate. Accordingly, the computation capacity provisioning for users may exceed the essential demands of the current workload because of adopting higher energy efficiency servers having higher processing rate.

Fig. 8 shows the number of active machines used in the STG algorithm. This algorithm always minimizes energy costs by combining the machines having different energy efficiencies, rather than prioritizing machines with more energy efficiency to pick up. By contrast, to reduce the overall energy consumption of DC, the DCP prefers to select more energy efficient machines. Therefore, this approach always turns on machines in decreasing order of energy efficiency. However, note that the static power of these servers with high energy efficiency may be more than that of servers with low energy efficiency, e.g., the energy efficiency of Type1 is superior to that of type2, but its static power is also higher than Type1. Therefore, under the DCP approach, energy consumption may not be the least in some cases.

Following the preceding discussions, the energy consumption of STG will not exceed that of DCP when these approaches maintain identical QOS performance. Fig. 9 shows the average energy consumption of using the four methods. STG incurs the lowest energy costs, which correspond to a 5.1 percent reduction in energy cost compared with DCP.

6.5 Convergence of Algorithm

The number of scheduler agents can significantly influence the convergence of our method. In this study, we also evaluate the relationships between the number of scheduler agents and the convergence of the proposed scheme. In our evaluation, the parameters of the simulated servers are set

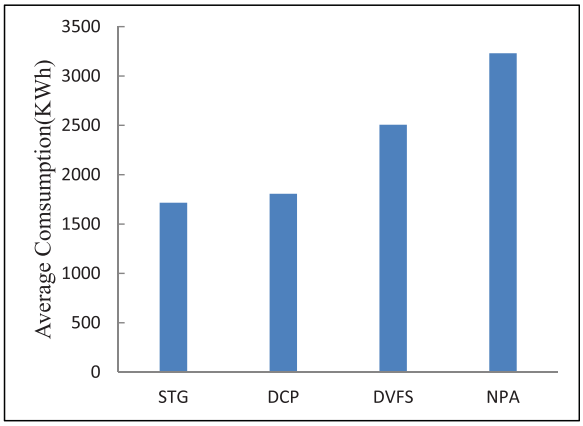


Fig. 9. Comparison of energy consumption.

by the configurations presented in Section 6.1. We measure the number of iterations needed to reach equilibrium for the system with 60 percent load rate and a variable number of scheduler agents (from 5 to 30), in which the deviation of stopping iterations is set to $\varepsilon < 0.001$ and the amount of initial tasks assigned on each server is proportional to their computation capacity in our evaluation. The experimental results show that our method exhibit similar convergence performance with that of the method proposed in [21]. This similarity validates the effectiveness of our scheme.

7 Conclusion

In this study, we investigated the power-performance tradeoff problem in allocating DC resources. We also presented a game-theoretic framework and associated algorithms to reduce energy consumption as much as possible while meeting the minimum performance requirements specified via SLAs. The presented algorithms have relatively low complexity and distribution execution characteristic, and thus, they can be easily implemented to improve the reliability and robustness of system. The effectiveness of our approach was assessed by performing simulated experiments in a real prototype environment. A comparison with popular technologies demonstrated that our approach could outperform alternative methods achieving better energy saving while maintaining the same processing performance. To a certain extent, our work is significant to promoting energy saving in DCs. Our methodology can be applied to other resource allocation models.

In the future, we plan to explore operating cost optimization and the VM allocation among multiple cloud providers while considering the heterogeneity of workloads from the perspective of the operator. We are also interested in investigating energy saving for geographically distributed DCs as an extension of our work.

Acknowledgments

The authors are grateful to three anonymous reviewers for their constructive comments. The work reported in this paper was supported by the National Natural Science Foundation of China (Grant No. 61173107), the National High Technology Research and Development Program of China (Grant No. 2012AA01A301-01), the Special Project on the Integration of Industry, Education and Research of Guangdong Province, China (No. 2012A090300003) and the Science and Technology Planning Project of Guangdong Province, China (No. 2013B090700003). Zhiyong Li is the corresponding author.

References

[1] C. Ge, Z. Sun, and N. Wang, "A survey of power-saving techniques on data centers and content delivery networks," *IEEECommun. Surveys Tuts.*, vol. 15, no. 3, pp. 1334–1354, Jul./Sep. 2013.

[2] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis Lectures Comput. Archit.*, vol. 8, no. 3, pp. 1–154, 2013.

[3] F. Kong and X. Liu, "A survey on green-energy-aware power management for datacenters," *ACM Comput. Surveys*, vol. 47, no. 2, pp. 30–67, 2015.

[4] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *J. Supercomput.*, vol. 60, no. 2, pp. 268–280, 2012.

[5] A. Rahman, X. Liu, and F. Kong, "A survey on geographic load balancing based data center power management in the smart grid environment," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 214–233, Jan./Mar. 2014.

[6] M. Guzek, P. Bouvry, and E.-G. Talbi, "A survey of evolutionary computation for resource management of processing in cloud computing [review article]," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 53–67, May. 2015.

[7] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2867–2876, Nov. 2014.

[8] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[9] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, 2010, pp. 577–578.

[10] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah, "Hybrid resource provisioning for minimizing data center SLA violations and power consumption," *Sustainable Comput. Informat. Syst.*, vol. 2, no. 2, pp. 91–104, 2012.

[11] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.

[12] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 14–28, Mar. 2014.

[13] K. Li, "Optimal power allocation among multiple heterogeneous servers in a data center," *Sustainable Comput. Inform. Syst.*, vol. 2, no. 1, pp. 13–22, 2012.

[14] J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 45–58, Jan. 2014.

[15] J. Cao, H. Kai, K. Li, and A. Y. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[16] B. Guenter, N. Jain, and C. Williams, "Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning," in *Proc. IEEE INFOCOM*, 2011, pp. 1332–1340.

[17] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: Data center energy-efficient network-aware scheduling," *Cluster Comput.*, vol. 16, no. 1, pp. 65–75, 2013.

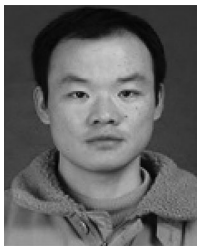
[18] T. Imada, M. Sato, Y. Hotta, and H. Kimura, "Power management of distributed web savers by controlling server power state and traffic prediction for QoS," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2008, pp. 1–8.

[19] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proc. 5th USENIX Symp. Netw. Syst. Des. Implementation*, 2008, pp. 337–350.

[20] S. Wang, J. J. Chen, J. Liu, and X. Liu, "Power saving design for servers under response time constraint," in *Proc. IEEE 22nd Euromicro Conf. Real-Time Syst.*, 2010, pp. 123–132.

[21] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," *J. Parallel Distrib. Comput.*, vol. 65, pp. 1022–1034, 2005.

[22] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "A cooperative game framework for QoS guided job allocation schemes in grids," *IEEE Trans. Comput.*, vol. 57, no. 10, pp. 1413–1422, Oct. 2008.
[23] R. Subrata and A. Y. Zomaya, "Game-theoretic approach for load balancing in computational grids," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 1, pp. 66–76, Jan. 2008.
[24] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "Cooperative power-aware scheduling in grid computing environments," *J. Parallel Distrib. Comput.*, vol. 70, no. 2, pp. 84–91, 2010.
[25] M. Feldman, K. Lai, and L. Zhang, "The proportional-share allocation market for computational resources," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 8, pp. 1075–1088, Aug. 2009.
[26] X. León and L. Navarro, "A Stackelberg game to derive the limits of energy savings for the allocation of data center resources," *Fut. Gen. Comput. Syst.*, vol. 29, no. 1, pp. 74–83, 2013.
[27] R. B. Myerson, *Game Theory*. Cambridge, MA, USA: Harvard Univ. Press, 2013.
[28] P. J. Reny, "On the existence of pure and mixed strategy Nash equilibria in discontinuous games," *Econometrica*, vol. 67, no. 5, pp. 1029–1056, 1999.
[29] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1962.
[30] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," in *Proc. ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, 2009, pp. 51–60.
[31] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated control of multiple virtualized resources," in *Proc. ACM Eur. Conf. Comput. Syst.*, 2009, pp. 13–26.
[32] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in *Proc. 18th ACM Int. Symp. High Performance Distrib. Comput.*, 2009, pp. 101–110.
[33] Spec power benchmarks. (2015). [Online]. Available: http://www.spec.org/power_ssj2008/results/power_ssj2008.html.
[34] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Ann. Oper. Res.*, vol. 153, no. 1, pp. 235–256, 2007.
[35] U. Bhaskar, L. Fleischer, and E. Anshelevich, "A stackelberg strategy for routing flow over time," *Games Econ. Behav.*, vol. 10, no. 1, pp. 192–201, 2013.
[36] Y. Chu, F. You, J. M. Wassick, and A. Agarwal, "Integrated planning and scheduling under production uncertainties: Bi-level model formulation and hybrid solution method," *Comput. Chem. Eng.*, vol. 72, pp. 255–272, 2015.
[37] T. Roughgarden, "Stackelberg scheduling strategies," *SIAM J. Comput.*, vol. 33, no. 2, pp. 332–350, 2004.
[38] B. Wang, Z. Han, and K. Liu, "Distributed relay selection and power control for multiuser cooperative communication networks using stackelberg game," *IEEE Trans. Mobile Comput.*, vol. 8, no. 7, pp. 975–990, Jul. 2009.
[39] S. Maharjan, Q. Zhu, Y. Zhang, S. Gjessing, and T. Basar, "Dependable demand response management in the smart grid: A stackelberg game approach," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 120–132, Mar. 2013.
[40] H. Von Stackelberg, Marktform Und Gleichgewicht, Berlin, Germany: Springer, 1934.
[41] G. E. Asimakopoulou, A. L. Dimeas, and N. D. Hatziargyriou, "Leader-follower strategies for energy management of multi-microgrids," *IEEE Trans. Smart Grid*, vol. 4, no. 4, pp. 1909–1916, Dec. 2013.
[42] T. D. Braun, H. J. Siegel, A. A. Maciejewski, and Y. Hong, "Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions," *J. Parallel Distrib. Comput.*, vol. 68, no. 11, pp. 1504–1516, 2008.
[43] H. Goudarzi, M. Ghasemazar, and M. Pedram, "SLA-based optimization of power and migration cost in cloud computing," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2012, pp. 172–179.
[44] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. 3rd ACM Symp. Cloud Comput.*, 2012, Art. no. 7.
[45] G. Lee, B.-G. Chun, and R. H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud," in *Proc. HotCloud*, 2011, pp. 1–5.
[46] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge, U.K.: Cambridge Univ. Press, 2013.



Bo Yang received the MSc degree in computer science and technology from Hunan University, China, in 2005. He is currently working toward the PhD degree at Hunan University, China. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, game theory, and mobile computing.

Zhiyong Li received the MSc degree in system engineering from the National University of Defense Technology, Changsha, China, in 1996 and the PhD degree in control theory and control engineering from Hunan University, Changsha, China, in 2004. Now, he is a full professor with Hunan University, member of China Computer Federation. His research interests include embedded computing system, visual object tracking, dynamic multi-objective evolutionary algorithm and tasks scheduling optimization in cloud computing. He obtained several awards from academic organizations and conferences, such as the Champion of the Future Challenge: Intelligent Vehicles and Beyond, FC'09, which was hosted by the National Natural Science Fund Committee of China in 2009. He is a member of IEEE.

Shaomiao Chen received the MSc degree in computer science and technology from Hunan University, Changsha, China, in 2014. He is currently working toward the PhD degree at Hunan University. His research interests include parallel computing, evolutionary computation and scheduling optimization.

Tao Wang received the PhD degree in computer application from Central South University, in 2014. He is an assistant professor at Hunan University. . His research interests include evolutionary computing, complex system and network

Keqin Li is a SUNY distinguished professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published more than 390 journal articles, book chapters, and refereed conference papers, and has received several Best Paper Awards. He is currently or has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *Journal of Parallel and Distributed Computing*. He is Fellow of IEEE.