IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 11, 2019, accepted July 4, 2019, date of publication July 8, 2019, date of current version July 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927406

Online Power-Aware Deployment and Load Distribution Optimization for Application Server Clusters

ZHI XIONG, YU CHENG, LINGRU CAI, AND WEIHONG CAI
Department of Computer Science and Technology, Shantou University, Shantou 515063, China

Corresponding author: Zhi Xiong (zxiong@stu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61202366, in part by the Natural Science Foundation of Guangdong Province under Grant 2018A030313438 and Grant 2018A030313889, in part by the Special Funds for Discipline and Specialty Construction of Guangdong Higher Education Institutions under Grant 2016KTSCX040, and in part by the Science and Technology Planning Project of Guangdong Province under Grant 2019B010116001 and Grant 2016B010124012.

ABSTRACT The energy conservation of application server clusters is a pressing problem. In this paper, we propose an online power-aware deployment and load distribution optimization strategy for application server clusters, whose objective is to minimize the cluster's power while ensuring that the server's CPU utilization is not higher than a preset value. The strategy includes two schemes: a mixed integer linear programming (MILP)-based scheme and a mixed integer non-linear programming (MINLP)-based scheme. The former formulates the cluster optimization problem as a MILP problem and adopts a toolkit to solve it. When the cluster scale is small, it can find the global optimal solution quickly. So, the MILP-based scheme is applicable to small-scale clusters. In the MINLP-based scheme, we first formulate the cluster optimization problem as a non-linear programming problem, and then design a method to reduce the number of variables and reformulate it as an MINLP problem. We finally propose an efficient solution method based on the flower pollination algorithm. Due to the small number of variables and the high solution efficiency, the solution method can quickly obtain a high-quality solution, so the MINLP-based scheme can be applied to large-scale clusters. The experimental results demonstrate the effectiveness of our strategy.

INDEX TERMS Application server cluster, power-aware deployment, load distribution, CPU utilization, optimization.

I. INTRODUCTION

In a web system, dynamic requests (e.g., the requests for ''.php'' or ''.jsp'' files) are usually separated from static requests (e.g., the requests for ''.png'' or ''.html'' files) and handed by an application server. Dynamic requests usually cannot be cached and need to consume more server resources. An application server cluster is a widely used technique for improving the performance of application servers. Application server clusters have become the infrastructure for large web-based applications, and many are deployed in datacenters to supply various services.

Server clusters consume a great deal of energy and thus incur large operational costs [1]–[3]. The Natural Resources Defense Council estimated that the consumption of datacenters in USA reached 91 billion kWh in 2013 and will increase to roughly 140 billion kWh by 2020 [4]. So the energy conservation of a cluster is a pressing problem that must be addressed. Application server clusters are usually designed and built according to the peak load, but the normal loads are far below the peak load. Moreover, application server clusters must supply quality-assured service because poor service may lead to customer churn. Therefore, the cluster's deployment and (corresponding) load distribution must be dynamically optimized according to the varying load online to reduce the cluster's power consumption while assuring good Quality of Service (QoS).

Optimizing cluster deployment based on a programming problem is a popular approach. This approach defines variables to denote the server's on/off state and frequency selection, uses QoS assurance as a constraint, takes power minimization as the objective, and then obtains a constrained

The associate editor coordinating the review of this manuscript and approving it for publication was Irfan Ahmed.

programming problem and solves the problem. However, previous studies have had two types of limitations.

(i) Indicator of QoS. Most previous studies [5]–[14] focus on the server's response time but ignore the server's CPU utilization. This is not advisable for application server clusters for three reasons. First, it is difficult to ensure the response time precisely in a practical multi-tier web system [15]. Second, CPU-intensive business logic processing is often the bottleneck [7] in multi-tier applications, so CPU utilization is an important performance indicator that reflects the load state of an application server. Finally, for a server, high utilization usually achieves high power-efficiency, so the optimization results of these studies make the CPU fully loaded or near-fully loaded. This is dangerous because even a minor load burst may make the server unresponsive. Therefore, the server's utilization must stay within a reasonable range [16].

(ii) Real-time solution of the programming problem. Cluster optimization problems are usually constrained non-linear programming problems that are a type of NP-hard problem. The previous studies all define variable(s) for each server in their optimization problems. When applied to large-scale clusters, it is difficult to solve these problems online because of the large number of variables. So some studies mainly use heuristic or greedy algorithms to solve their optimization problems. However, they cannot obtain the optimal solution. Some studies design ingenious solution methods, but they do not evaluate these methods in real time. In addition, the solution methods proposed in some studies lack practical feasibility. We will discuss these in detail in Section II.

To address these limitations, we propose an online power-aware optimization strategy for application server clusters, whose optimization content involves the on/off state of each server, the CPU frequency of each running server, and the load distribution. Our optimization assumes QoS assurance, and the QoS objective is that the CPU utilization of each running server is not more than a given value. A schematic of the strategy is illustrated in Fig. 1.

Our optimization strategy includes two schemes: a MILP (Mixed Integer Linear Programming)-based scheme and a MINLP (Mixed Integer Non-Linear Programming)-based scheme. The MILP-based scheme formulates the cluster optimization problem as a MILP problem and adopts GLPK (GNU Linear Programming Kit) [17] to solve the problem. It is applicable to small-scale clusters. In the MINLP-based scheme, we first formulate the cluster optimization problem as a NLP (Non-Linear Programming) problem, reformulate it as a MINLP problem to reduce the number of variables, and then propose an efficient FPA (Flower Pollination Algorithm)-based solution method for the MINLP problem. The scheme can be applied to large-scale clusters. Experimental results demonstrate the effectiveness of our strategy. The main contributions of this paper are:

- In our optimization strategy, we focus on server's utilization and the server's utilization is kept at a reasonable level.

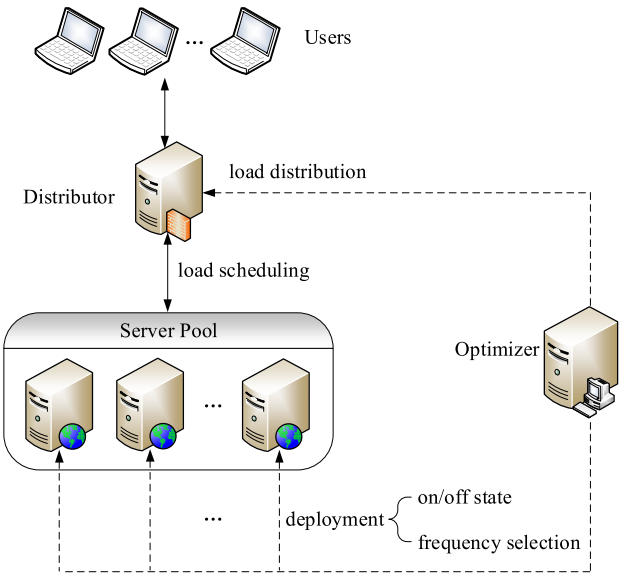


FIGURE 1. Schematic of our strategy.

- We propose two optimization schemes for application server clusters. The MILP-based scheme improves the work in [18] and is applicable to small-scale clusters, and the MINLP-based scheme can be applied to large-scale clusters.
- In the MINLP-based scheme, we propose a hypothesis. Based on the hypothesis and analysis, we define variables for each server model (but no each server), so the number of variables is substantially reduced. Moreover, aimed at the trait of the MINLP problem, we propose an FPA-based solution method. Due to the small number of variables and the high solution efficiency, the FPA-based solution method can obtain a high-quality solution quickly, so the MINLP-based scheme can be applied to large-scale clusters.
- We present our evaluation results in detail. The results validate the feasibility of the hypothesis and the effectiveness of our two schemes.

The remainder of this paper is organized as follows. The related studies are discussed in Section II. Sections III and IV propose the MILP- and MINLP-based schemes, respectively. The experiments and analysis are given in Section V. Finally, we conclude in Section VI.

II. RELATED STUDIES

Server Dynamic Switch On/Off (DSO) and CPU Dynamic Frequency Scaling (DFS) are two effective power optimization measures for clusters. Instead of power-off, suspending-to-RAM is used in practice to shorten the server's startup time [18]. Several studies depend only on server DSO [6], [12], [19]–[22], CPU DFS [8], [14], or the load distribution [23], [24] to optimize the cluster power. However, these methods cannot obtain the best optimization results.

The energy conservation of a cluster should be carried out under the precondition of QoS assurance. As an indicator

of QoS, most previous studies focus on the server's response time but ignore the server's CPU utilization. As discussed in the previous section, this is not advisable for application server clusters. Moreover, CPU utilization and response time are quite distinct from each other, and they are not symmetrical metrics, so response time cannot be simply changed to CPU utilization in their approaches. Although [25] uses utilization as a QoS indicator, it is based on divide-and-conquer (it does not consider server DSO and CPU DFS together) and heuristic methods, so it cannot obtain the best optimization result. Reference [23] imposes punishment on the servers whose utilization is greater than 80% when calculating the power efficiency but does not ensure the server's utilization.

In the solution method for the cluster optimization problem, [19], [26]–[28] use heuristic or greedy algorithms on the whole. Reference [8] first reduces the number of variables by half through a theoretical analysis and then uses a binary search; [5] uses Generalized Benders Decomposition; [18] and [16] use GLPK and CPLEX, respectively; and [14] proposes a bisection method and several heuristic methods. However, these studies do not evaluate the efficiency of their solution methods. Reference [26] uses a genetic algorithm to solve the optimization problem, but it does not evaluate the algorithm in large-scale clusters. To ensure online optimization, [18] presents a method to "first build a load-solution table offline, then look up the table online"; however, the method lacks operability because the table has to be rebuilt when the servers in the cluster change. Reference [11] proposes a distributed solution method based on dual decomposition, but it does not evaluate the efficiency of the method. Moreover, the servers have already taken on a heavy workload; if they also take part in solving the optimization problem, it is difficult to ensure a real-time solution.

In addition, the optimization objects of many related studies [21], [29] are homogenous clusters, but heterogeneous clusters are much more common in practice.

III. MILP-BASED OPTIMIZATION SCHEME

Reference [18] formulated the cluster optimization problem as a MILP problem. However, it has two shortcomings: (i) it lets servers be fully utilized but does not involve CPU utilization assurance; and (ii) it ignores the standby power when a server is turned off. In this section, we improve the work in [18] and propose a new MILP-based optimization scheme that solves the above two shortcomings. In our two schemes, we let all of the cores of the multi-core CPU work at the same frequency to reduce the complexity of the optimization problem.

A. DEFINITION AND SYMBOLS

We first define load capacity. When a server works at a certain frequency with 100% CPU utilization, the load (namely, the request arrival rate) it can bear is defined as the load capacity of the frequency.

Consider a cluster that consists of c servers S_i ($0 \le i \le c-1$). The related parameters of server S_i are given in Table 1.

TABLE 1. Parameters of server S_i.

Parameter	Explanation
f_i	The number of discrete frequencies
$f_{i,j}$	The jth discrete frequency (from low to high), $1 \le j \le f_i$
$p_{i,j}^{busy}$	The busy power of $f_{i,j}$
$p_{i,j}^{idle}$	The idle power of $f_{i,j}$
$l_{i,j}$	The load capacity of $f_{i,j}$
$p_i^{standby}$	The standby power when the server is suspended to RAM

Suppose that the upper limit of the CPU utilization is Φ and that the predicted cluster load is $L^{predict}$.

B. MIXED INTEGER LINEAR PROGRAMMING (MILP) PROBLEM

We define three variables for each frequency of each server: the binary variable $z_{i,j}$ denotes whether S_i works at $f_{i,j}$ or not, the real variable $x_{i,j}$ denotes the CPU busy ratio (namely, CPU utilization) when S_i works at $f_{i,j}$, and the real variable $y_{i,j}$ denotes the CPU idle ratio when S_i works at $f_{i,j}$. Note that we must ensure that $x_{i,j}$ is not greater than Φ.

To describe the server's on/off state along with the CPU frequency selection, we supplementally define the frequency $f_{i,0}$ that denotes that S_i is turned off. Moreover, we let $p_{i,0}^{busy} = p_{i,0}^{idle} = p_i^{standby}$ and $l_{i,0} = 0$. Accordingly, we define three variables: $x_{i,0}$, $y_{i,0}$, and $z_{i,0}$.

Obviously, if $z_{i,j} = 1$ (i.e., S_i works at $f_{i,j}$), then $x_{i,j} + y_{i,j} = 1$. If $z_{i,j} = 0$ (i.e., S_i does not work at $f_{i,j}$), we specify $x_{i,j} = y_{i,j} = 0$. Therefore, $x_{i,j} + y_{i,j} - z_{i,j} = 0$. Then, the cluster optimization problem can be formulated as the following problem:

$$\min \sum_{i=0}^{c-1} \sum_{j=0}^{f_i} \left(x_{i,j} p_{i,j}^{busy} + y_{i,j} p_{i,j}^{idle} \right), \tag{1}$$

$$\text{s.t.} \sum_{i=0}^{c-1} \sum_{j=0}^{f_i} x_{i,j} l_{i,j} = L^{predict}; \tag{2}$$

$$\sum_{j=0}^{f_i} z_{i,j} = 1, \quad \forall i \in \{0, \ldots, c-1\}; \tag{3}$$

$$x_{i,j} + y_{i,j} - z_{i,j} = 0, \tag{4}$$

$$z_{i,j} \in \{0, 1\}, \tag{5}$$

$$x_{i,j} \in [0, \Phi], \tag{6}$$

$$y_{i,j} \in [0, 1], \quad \forall i \in \{0, \ldots, c-1\}, \ \forall j \in \{0, \ldots, f_i\}. \tag{7}$$

In the problem,

- $x_{i,j}$, $y_{i,j}$, and $z_{i,j}$ are variables. The on/off state and frequency selection of S_i are determined by $\sum_{j=0}^{f_i} z_{i,j} j$. For example, 0 denotes that S_i is power-off (standby), and 2 denotes that S_i works at the 2nd frequency (namely, $f_{i,2}$). The load of S_i is given by $\sum_{j=0}^{f_i} x_{i,j} l_{i,j}$.
- $\sum_{j=0}^{f_i} \left(x_{i,j} p_{i,j}^{busy} + y_{i,j} p_{i,j}^{idle} \right)$ is the power of S_i, and (1) is the total power of all servers (namely, the cluster power). Equation (6) is the constraint for CPU

utilization assurance. Therefore, the optimization objective is to minimize the cluster power under the precondition of CPU utilization assurance.

C. SOLUTION METHOD FOR THE MILP PROBLEM

The problem expressed by (1)-(7) is a MILP problem. The GNU Linear Programming Kit (GLPK) is a software package intended for solving large-scale LP (Linear Programming), MILP, and other related problems. It uses the branch-and-bound algorithm with Gomory's mixed integer cuts for (mixed) integer problems. In theory, we can use the GLPK to solve the MILP problem. However, the problem defines variables for each frequency of each server, so the number of variables is enormous, especially if the cluster scale is large. Faced with large numbers of binary variables and real variables, the GLPK cannot ensure a real-time solution. Therefore, the MILP-based scheme is only applicable to small-scale clusters.

IV. MINLP-BASED OPTIMIZATION SCHEME

To ensure the real-time solution of the cluster optimization problem, we start with two aspects: (i) reducing the number of variables, and (ii) designing an efficient solution method. In this section, we first define fewer variables and formulate the cluster optimization problem as a Non-Linear Programming (NLP) problem. We then propose a hypothesis. Based on the hypothesis, we redefine the variables (the number of variables is substantially reduced) and transform the NLP problem into a MINLP problem. Finally, we propose an FPA-based method to solve the MINLP problem.

A. NON-LINEAR PROGRAMMING (NLP) PROBLEM

We still use the symbols defined in Subsection III(A). We define the variable l_i, $l_i \geq 0$, to denote the load assigned to S_i. If $l_i = 0$, then S_i is turned off. Because the maximum load capacity of S_i is l_{i,f_i} and the upper limit for CPU utilization is Φ, we must ensure that $l_i \leq l_{i,f_i}\Phi$.

Because the server power is a cubic function of the CPU frequency [5], [30], [31] but a linear function of the CPU utilization [18], [31], we adopt the following practice—letting a server work at the lowest possible frequency (with a high CPU utilization) as long as its CPU utilization is not higher than Φ. The feasibility of this practice is verified by our experiments in Subsection V(B). Therefore, S_i works at the frequency $f_{i,k}$, where the integer k satisfies $l_{i,k-1}\Phi < l_i \leq l_{i,k}\Phi$, $k \in \{1, \ldots, f_i\}$ (we supplementally define $l_{i,0} = 0$). Then, its CPU utilization is $l_i/l_{i,k}$. According to the linear relationship between power and CPU utilization [18], [31], the power of S_i is

$$power_i(l_i) = \begin{cases} p_{i,k}^{idle} + \dfrac{l_i}{l_{i,k}}\left(p_{i,k}^{busy} - p_{i,k}^{idle}\right), & l_{i,k-1}\Phi < l_i \leq l_{i,k}\Phi \\ p_i^{standby}, & l_i = 0. \end{cases} \tag{8}$$

It is easy to prove that $power_i(l_i)$ is a piecewise linear function and is discontinuous at $l_{i,j}\Phi$ and 0. The cluster optimization problem can be formulated as the following problem:

$$\min \sum_{i=0}^{c-1} power_i(l_i), \tag{9}$$

$$\text{s.t.} \sum_{i=0}^{c-1} l_i = L^{predict}; \tag{10}$$

$$l_i \in [0, l_{i,f_i}\Phi], \quad \forall i \in \{0, \ldots, c-1\}. \tag{11}$$

In the problem,

- l_i $(0 \leq i \leq c-1)$ are variables. The load of S_i is directly given by l_i, and the on/off state and frequency selection of S_i are derived from l_i.
- Equation (9) is the cluster power, and (11) is the constraint for CPU utilization assurance. Therefore, the optimization objective is still to minimize the cluster power under the precondition of CPU utilization assurance.

The problem expressed by (9)-(11) is a discontinuous NLP problem. It only defines one variable (l_i) for each server, and its number of variables is far less than that of the MILP problem expressed by (1)-(7). However, when the cluster scale is large, the number of variables is still enormous. It is difficult to solve a large-scale discontinuous NLP problem. Therefore, to ensure the real-time solution, we must further reduce the number of variables of the NLP problem.

B. MIXED INTEGER NON-LINEAR PROGRAMMING (MINLP) PROBLEM

In a practical cluster, it is difficult to guarantee that all servers are the same. However, we usually purchase, replace, or upgrade a batch of servers together, so many servers are the same. We treat homogeneous servers as the same server model. Generally, the number of server models is more than an order of magnitude lower than the number of servers in a practical cluster. If we can define variables for each server model (but not for each server), number of variables will be sharply reduced.

TABLE 2. Parameters of server model M_i.

Parameter	Explanation
N_i	Number of servers
F_i	The number of discrete frequencies
$F_{i,j}$	The jth discrete frequency (from low to high), $1 \leq j \leq F_i$
$P_{i,j}^{busy}$	The busy power of $F_{i,j}$
$P_{i,j}^{idle}$	The idle power of $F_{i,j}$
$L_{i,j}$	The load capacity of $F_{i,j}$
$P_i^{standby}$	The standby power when the server is suspended to RAM

Assume that there are C server models M_i $(0 \leq i \leq C-1)$. The related parameters of server model M_i are given in Table 2.

For server model M_i ($0 \le i \le C-1$), (8) can be rewritten as follows:

$$Power_i(l) = \begin{cases} P_{i,k}^{idle} + \dfrac{l}{L_{i,k}}\left(P_{i,k}^{busy} - P_{i,k}^{idle}\right), & L_{i,k-1}\Phi < l \le L_{i,k}\Phi \\ P_i^{standby}, & l = 0. \end{cases} \tag{12}$$

We now examine the servers of the same model. We still take M_i as an example.

- According to (12), if multiple (e.g., w ($2 \le w \le N_i$)) servers work in an identical load range $(L_{i,k-1}\Phi, L_{i,k}\Phi]$ (i.e., they work at the same frequency—$F_{i,k}$), the sum of their powers ($P_{\sum}$) is a linear function of the sum of their loads ($L_{\sum}$). Therefore, we can let these servers bear the same load $\frac{L_{\sum}}{w}$ and consume the same power $\frac{P_{\sum}}{w}$ because this will neither change the sum of their loads and the sum of their powers nor violate the QoS assurance.
- Servers usually should work at the highest cost-effective load. However, there is an equality constraint for the sum of the loads (namely, (10)); therefore, servers cannot "choose" their load freely. Even so, they will still tend to work in a few highly cost-effective load ranges. Hence, we propose the following hypothesis: the running servers work at a maximum of two different load ranges $(L_{i,k-1}\Phi, L_{i,k}\Phi]$ (i.e., a maximum of two different frequencies). It is worth emphasizing that under this hypothesis, we may not be able to obtain the global optimal solution; nevertheless, we can obtain at least a very good near-optimal solution (this is verified by our experiments in Subsection V(F)). However, based on this hypothesis and the first point, we can conveniently define variables for each server model, which substantially reduces the number of variables.

Based on these two points, we divide the servers of each server model into three groups. The servers in the first group bear the same load, the servers in the second group also bear the same load (note that it could be equal to or not equal to the load of the first group), and the servers in the third group are turned off. Below, we redefine the variables and rephrase the NLP problem expressed by (9)-(11).

For server model M_i ($0 \le i \le C-1$), we define four variables: $n_{i,1}$, $l_{i,1}$, $n_{i,2}$, and $l_{i,2}$. Specifically, $n_{i,1}$ and $n_{i,2}$ are two integers in the range of $[0, N_i]$, and $l_{i,1}$ and $l_{i,2}$ are two real numbers in the range of $(0, L_{i,F_i}\Phi]$. They denote that the servers are divided into three groups; there are $n_{i,1}$ servers in the first group, and they all work with load $l_{i,1}$; there are $n_{i,2}$ servers in the second group, and they all work with load $l_{i,2}$; and the remaining $N_i - n_{i,1} - n_{i,2}$ servers belong to the third group, and they are turned off. Then, the NLP program expressed by (9)-(11) can be rephrased as the following MINLP problem:

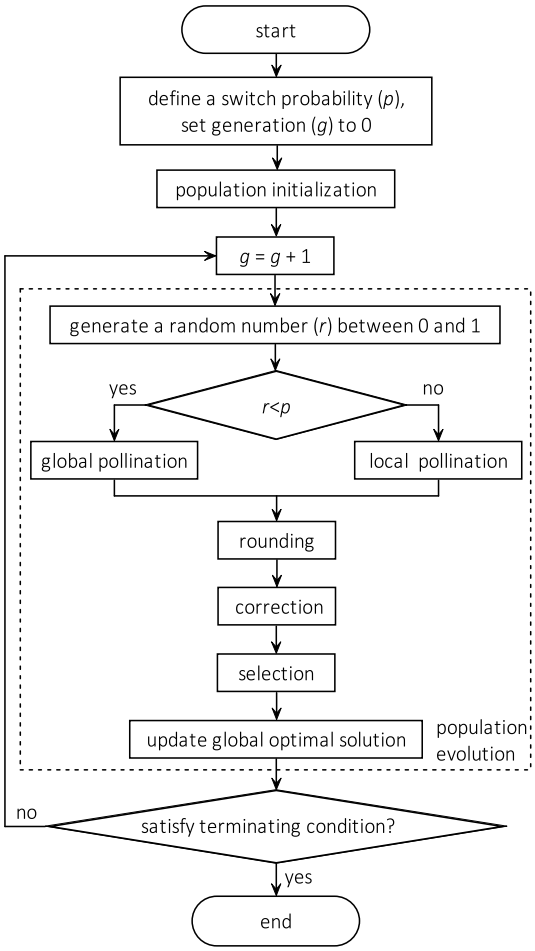


FIGURE 2. Flow diagram of the FPA-based solution method.

$$\min \sum_{i=0}^{C-1}\left(n_{i,1}Power_i\left(l_{i,1}\right) + n_{i,2}Power_i\left(l_{i,2}\right) + \left(N_i - n_{i,1} - n_{i,2}\right)P_i^{standby}\right), \tag{13}$$

$$s.t. \sum_{i=0}^{C-1}\left(n_{i,1}l_{i,1} + n_{i,2}l_{i,2}\right) = L^{predict}; \tag{14}$$

$$l_{i,1} \in (0, L_{i,F_i}\Phi], \quad l_{i,2} \in (0, L_{i,F_i}\Phi], \tag{15}$$

$$n_{i,1} \in \{0, \ldots, N_i\}, \quad n_{i,2} \in \{0, \ldots, N_i\}, \tag{16}$$

$$n_{i,1} + n_{i,2} \le N_i, \quad \forall i \in \{0, \ldots, C-1\}. \tag{17}$$

C. FPA-BASED SOLUTION METHOD FOR THE MINLP PROBLEM

Because of their good flexibility and generality, intelligent optimization algorithms are effective solution methods for complicated NLP problems, and they have been widely used to solve various optimization problems in different fields. However, their solution efficiency and quality are greatly influenced by the algorithm parameters. Fewer parameters may be beneficial to ensure the stability of the algorithm.

The Flower Pollination Algorithm (FPA) is a population-based optimization algorithm that was proposed in 2012 by

Algorithm 1 Correction Operator

1: //for (15), (16), and (17)
2: **if**($n_{i,1}$, $l_{i,1}$, $n_{i,2}$, $l_{i,2}$, or $n_{i,1}+n_{i,2}$ is smaller/bigger than its lower/upper bound ($i \in\{0,\ldots,C\text{-}1\}$))
3: set it to its lower/upper bound;
4: // for (14)
5: $l=\sum_{i=0}^{C-1}\left(n_{i,1}l_{i,1}+n_{i,2}l_{i,2}\right)$;
6: **while** ($l > L^{predict}$) {
7: select *i, j* from $\{i, j|n_{i,j} > 0, i \in\{0, 1, \ldots, C\text{-}1\}, j \in\{1, 2\}\}$ whose $n_{i,j}l_{i,j}$ is minimal;
8: if ($n_{i,j}l_{i,j} < l - L^{predict}$) {// turn off the server group
9: $l = l - n_{i,j}l_{i,j}$;
10: $n_{i,j} = 0$; $l_{i,j} = 0$;
11: }
12: **else** {// only need to reduce the server load of the server group
13: $l_{i,j} = (n_{i,j}l_{i,j} - (l - L^{predict}))/n_{i,j}$;
14: **return**;
15: }
16: }
17: **while** ($l < L^{predict}$) {
18: select *i, j* from $\{i, j\,|n_{i,j} > 0, n_{i,1}+n_{i,2} < N_i$ or $l_{i,j} < L_{i,F_i}\Phi, i \in\{0, 1, \ldots, C\text{-}1\}, j \in\{1, 2\}\}$
19: whose $l_{i,j}/Power_i\left(l_{i,j}\right)$ is minimal;
20: **if** (*i, j* do not exist) {// need to turn on server(s) of a new server model
21: select *i* from $\{i|n_{i,1}+n_{i,2}=0, i \in\{0, 1, \ldots, C\text{-}1\}\}$ whose $\max\{x/Power_i(x), x \in(0, L_{i,F_i}\Phi)\}$ is the highest;
22: j= 1;
23: }
24: **if** ($L^{predict} - l$ is big enough, such that $n_{i,j}$ can be adjusted to $N_i - n_{i,3-j}$, and $l_{i,j}$ can be adjusted to $L_{i,F_i}\Phi$) {
25: $load = n_{i,j}l_{i,j}$;
26: $n_{i,j} = N_i - n_{i,3-j}$; $l_{i,j} = L_{i,F_i}\Phi$;
27: $l = l + n_{i,j}l_{i,j} - load$;
28: }
29: **else** {// only need to increase $n_{i,j}l_{i,j}$. Meanwhile, we minimize $n_{i,j}$.
30: $load = n_{i,j}l_{i,j} + L^{predict} - l$;
31: $n_{i,j} = \text{ceil}(load/(L_{i,F_i}\Phi))$; $l_{i,j} = load/n_{i,j}$;
32: **return**;
33: }
34: }

Yang [32]. It is based on the pollination process of flowering plants and includes two main operators: global pollination and local pollination. The two pollinations are controlled by a switch probability. Other than the population size, the switch probability is the only parameter in the FPA, which allows the FPA to be easily manipulated. So we adopt the FPA to solve the MINLP problem expressed by (13)-(17).

Each individual is expressed as ($n_{0,1}$, $l_{0,1}$, $n_{0,2}$, $l_{0,2}$, $n_{1,1}$, $l_{1,1}$, $n_{1,2}$, $l_{1,2}$, …, $n_{C-1,1}$, $l_{C-1,1}$, $n_{C-1,2}$, $l_{C-1,2}$). The optimization variables in the FPA are real numbers, so we add a rounding operator for the integer variables $n_{i,1}$ and $n_{i,2}$, and use the *ceil()* and *floor()* functions at random. After the pollination and rounding operator, it is nearly impossible for individuals to satisfy the equality constraint in (14). Therefore, we add a correction operator to correct individuals, which makes the individuals satisfy all of the constraints in (14)-(17). The flow diagram of the FPA-based solution method is presented in Fig. 2.

To prevent introducing too much computational overhead, the correction operator is based on a greedy strategy; that is, we reduce the number of running servers to as few as possible. Algorithm 1 gives the pseudocode of the correction operator. In the algorithm, lines 6-16 handle the situation in which the total load is higher than the predicted load. In this situation, we first select a running server group whose total load is minimal. If possible, we turn off the server group and then continue the correction operation; otherwise, we only need to reduce the server load of the server group. Lines 17-34 handle the situation in which the total load is lower than the predicted load. In this situation, we first select a running server group whose power efficiency (defined as the ratio of the load to the power) is minimal, and the group scale or server load can be increased. If such a server group does not exist, we need to turn on server(s) of a new model, and we select the model whose maximum power efficiency is the highest. If possible, we maximize the group scale and server load of the server

TABLE 3. Power and load capacity data of the four server models.

Model	CPU	Frequency (GHz)	Idle power (W)	Busy power (W)	Load capacity (req/s)
0	AMD Athlon 64 3500+	1.0, 1.8, 2.0, 2.2	66.6, 73.8, 76.9, 80.0	74.7, 95.7, 103.1, 110.6	51.2, 91.2, 101.4, 111.4
1	AMD Athlon 64 3580+	1.0, 1.8, 2.0, 2.2, 2.4	68.0, 77.0, 74.0, 79.0, 85.0	70.0, 89.0, 100.0, 115.0, 136.0	55.3, 96.9, 107.0, 115.8, 124.6
2	AMD Athlon 64 5000+ (dual core)	1.0, 1.8, 2.0, 2.2, 2.4, 2.6	65.8, 68.5, 70.6, 72.3, 74.3, 76.9	82.5, 99.2, 107.3, 116.6, 127.2, 140.1	99.4, 177.4, 197.2, 218.0, 234.6, 255.2
3	Pentium-M 1.8G	0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8	42.0, 43.0, 43.0, 44.0, 45.0, 47.0, 49.0	44.0, 45.0, 47.0, 49.0, 51.0, 55.0, 60.0	37.3, 50.0, 62.4, 74.4, 88.4, 97.6, 111.4

group and then continue the correction operation; otherwise, we only need to increase the load of the server group.

To ensure a real-time solution, the FPA must also reduce the overhead and improve the efficiency. In the global pollination operator, the FPA needs to draw a step vector (each individual needs a step parameter in the pollination equation [32]) that obeys a Lévy distribution. This is an expensive operation. In our implementation, we generate a batch of step vectors beforehand offline and then randomly select one of these vectors online. In addition, the evolution of each individual is separated and independent, so we adopt OpenMP [33] to parallelize the population evolution process.

V. NUMERICAL EXPERIMENTS

As described in the previous sections, almost all previous studies focus on the cluster's power optimization under response time assurance but not under CPU utilization assurance, so their optimization objectives are different than ours. Moreover, most do not consider real-time optimization. Although a few studies have identified this issue and proposed methods to address it, their methods lack practical feasibility. Therefore, we do not compare our strategy to those methods. Instead, our experiments focus on:

- The reasonability of the practice and hypothesis in our MINLP-based scheme.
- Whether our MILP- and MINLP-based schemes can find the global optimal or a high-quality solution in an acceptable amount of time.
- How the two schemes perform in clusters with different scales.

It is worth emphasizing that our MILP-based scheme is an improved scheme of [18].

A. EXPERIMENT SCENARIO

Table 3 shows the power and load capacity data of the four server models supplied in [18]. The standby power of each server model is assumed to be 3.5 W. The clusters used for the experiments consist of the four server models. We consider four different cluster scales: 20, 40, 80, and 120 servers, and the numbers of servers of each model are equal. The upper limit of the CPU utilization is set to 85%. All experiments are run on a PC with an Intel Xeon E3-1230 v3 CPU, 8 GB of memory, and the Windows 7 64-bit operating system.

B. POWER AND POWER EFFICIENCY CURVE

For each server model, we plot the function curve between the power and load of each discrete frequency in the same figure, as shown in Fig. 3. The results show that

1) A load may correspond to multiple combinations of frequency and power.
2) In these combinations, the lower the frequency is, the lower the power is.

To better understand these observations, we consider the load of 60 req/s in the first subfigure. It corresponds to three combinations of "(frequency, power)", namely (1.8 GHz, 88.2 W), (2.0 GHz, 92.4 W), and (2.2 GHz, 96.5 W); the first combination has the lowest power and is the most cost-effective. According to these observations, we should let a server work at the lowest possible frequency as long as its CPU utilization meets the QoS goal, as discussed in Subsection IV(A). Based on the above analysis, the function curve of $Power_i(l)$ is marked with circles in each subfigure. The discontinuity of $Power_i(l)$ causes our optimization problem to have many local optimal solutions.

The power efficiency (namely, $l/Power_i(l)$) curve of each server model is given in Fig. 4. We can see that

1) The power efficiency is also discontinuous.
2) The power efficiency generally has an increasing trend, but it is not a monotonously increasing function.
3) Some server models achieve the maximum power efficiency at the highest load (e.g., server models 0 and 3), whereas some server models achieve the maximum power efficiency at a higher load but not at the highest load (e.g., server models 1 and 2).

Many studies only use greedy optimization algorithms that concentrate the load on some servers and make them as fully loaded as possible while turning off as many other servers as possible. However, observations 2) and 3) may cause the solution of the greedy algorithm to be unsatisfactory. In addition, observation 3) indicates that the clusters used in our experiments are "complicated", and it is difficult to obtain the global optimal solutions of their optimization problems.

C. NUMBER OF VARIABLES

In the remainder of this paper, we refer to the problems expressed by (1)-(7), (9)-(11), and (13)-(17) as the MILP, NLP, and MINLP problems, respectively. Table 4 lists the

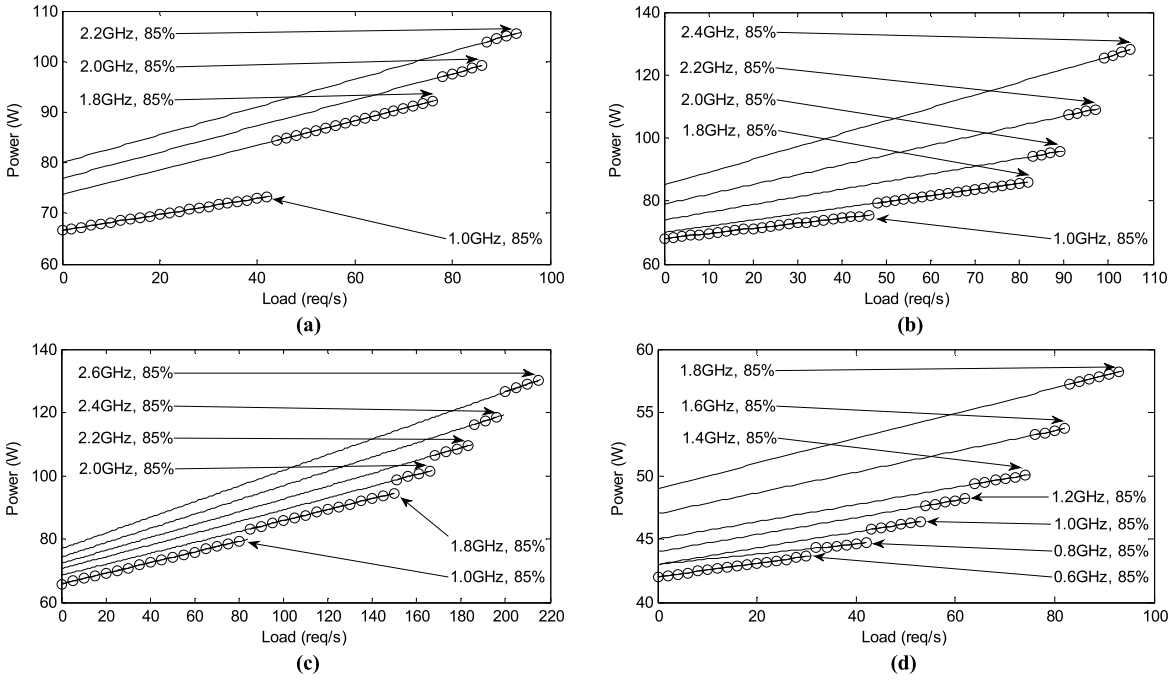


FIGURE 3. Power curves: (a) Server model 0; (b) Server model 1; (c) Server model 2; (d) Server model 3.

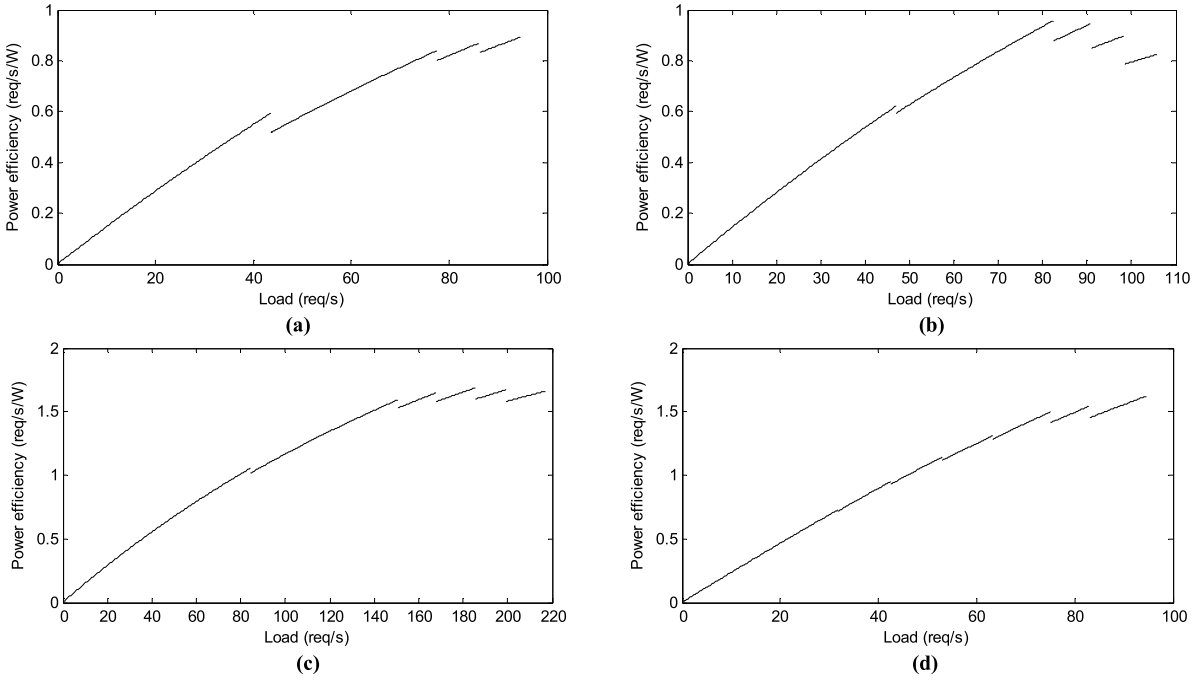


FIGURE 4. Power efficiency curves: (a) Server model 0; (b) Server model 1; (c) Server model 2; (d) Server model 3.

numbers of variables of the three problems with different cluster scales. As shown in the table,

1) The numbers of variables of the MILP and NLP problems increase with the cluster scale because they define variable(s) for each server.
2) The number of variables of the MINLP problem is determined solely by the number of server models because it defines variables for each server model.

D. APPLYING THE MILP-BASED SCHEME IN A SMALL-SCALE CLUSTER

We examine the first cluster scale in Table 4. The maximum load of the cluster is $MaxLoad = (111.4 + 124.6 + 255.2 + 111.4)*5*0.85 = 2561.05$ req/s, where 111.4, 124.6, 255.2 and 111.4 are the maximum load capacity of each server model respectively (see Table 3); 5 is the number of servers of each model; and 0.85 is the upper limit of the

TABLE 4. Numbers of variables of the three problems for clusters of different scales.

Cluster scale (number of servers)	Number of servers of each model	Variable number (real number, integer)		
		MILP problem	NLP problem	MINLP problem
20	5	260, 130	20, 0	8, 8
40	10	520, 260	40, 0	8, 8
80	20	1040, 520	80, 0	8, 8
120	30	1560, 780	120, 0	8, 8

TABLE 5. Results of solving the MILP problem using GLPK.

Scene	Cluster load (of *MaxLoad*)	Minimal power (W)	End time	Server frequency selection—$\sum_{j=0}^{f_i} z_{i,j}j$
1	10%	222.0	0.1 s	0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 7 0
2	20%	371.4	0.5 s	0 0 0 0 0 0 0 0 0 0 4 3 3 0 0 0 0 0 0 0
3	30%	514.2	0.2 s	0 0 0 0 0 0 0 0 0 0 4 4 5 0 5 0 0 0 0 0
4	40%	660.2	0.1 s	0 0 0 0 0 0 0 0 0 0 4 4 4 0 4 7 0 0 7 7
5	50%	812.1	0.3 s	0 0 0 0 0 0 0 0 0 0 4 4 4 4 3 7 7 7 0 7
6	60%	967.5	0.1 s	0 0 0 0 0 0 0 0 0 0 6 6 6 5 6 7 7 7 7 7
7	70%	1223.1	0.3 s	0 0 0 0 0 0 2 0 2 2 6 6 6 6 6 7 7 7 6 7
8	80%	1486.9	0.3 s	0 3 0 0 0 2 2 2 2 2 6 6 6 6 6 7 7 7 7 7
9	90%	1768.8	0.5 s	3 0 3 3 2 2 2 2 2 2 6 6 6 6 6 7 7 7 7 7

TABLE 6. Results for 10 servers of each model.

Scene	Cluster load (of *MaxLoad*)	Solution result			End time
		Computation time	Minimal power (W)	Server frequency selection—$\sum_{j=0}^{f_i} z_{i,j}j$	
10	10%	9.999 s	441.4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	24.9 s
11	20%	66.557 s (1.1 m)	730.2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 0 0 4 0 0 0 0 0 0 7 7 0 0 0 0 0 7	175.3 s
12	30%	22,706.803 s (6.3 h)	1028.4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 5 5 4 0 0 5 4 4 4 0 0 0 0 0 0 0 0 0 0	8.0 h (timeout)
13	40%	45.288 s	1320.4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 0 4 4 4 0 0 7 7 7 0 0 7 7 7 0	96.1 s
14	50%	0.487 s	1618.3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 5 4 4 4 4 4 6 7 7 7 7 7 7 0 0 0 7	59.0 s
15	60%	0.223 s	1935.1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 6 5 6 6 6 6 6 5 6 7 7 7 7 7 7 7 7 7 7	1.0 s
16	70%	10.132 s	2442.4	0 0 0 0 0 0 0 0 0 0 2 2 2 0 2 0 0 2 2 0 6 6 6 6 6 6 6 6 6 5 7 7 7 7 7 7 7 7 7 7	18.7 s
17	80%	1.971 s	2966.8	0 0 0 0 0 0 0 4 0 0 2 3 3 3 3 3 3 3 2 3 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7	4.2 s
18	90%	5.575 s	3513.4	0 4 0 4 4 4 4 0 4 4 2 2 2 2 2 2 3 2 2 2 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7	11.8 s

CPU utilization (see Subsection V(A)). We consider nine load scenes—10%, 20%, …, 90% of *MaxLoad*. For each load, we use the standalone solver GLPSOL in GLPK (version 4.65) to solve the MILP problem. During the solution process, it constantly outputs the current optimal solution that has been found. If the solution process ends, the last output is the global optimal solution; otherwise, the last output is not necessarily the global optimal solution. The results are given in Table 5.

For each scene in Table 5, the solution process ends at the time listed in the fourth column, so the minimal power listed in the third column is the global optimum; i.e., we can

TABLE 7. Results for 20 servers of each model.

Scene	Cluster load (of *MaxLoad*)	Solution result			End time
		Computation time	Minimal power (W)	Server frequency selection—$\sum_{j=0}^{f_i} z_{i,j}j$	
19	10%	0.203 s	874.0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 4 4 0 4 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7	8.0 h (timeout)
20	30%	0.204 s	2047.4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 4 4 4 4 4 4 4 4 4 4 4 0 5 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7	8.0 h (timeout)
21	50%	0.296 s	3231.0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 7 7 7 7 0 7 7 7 7 7 7 0 7 7 0 0 0 7 7	18,234.0 s (5.1 h)
22	70%	/[a]	/[a]	/[a]	8.0 h (timeout)
23	90%	639.217 s (10.7 m)	7052.8	4 4 4 4 0 4 0 0 0 0 0 4 4 4 4 4 4 4 4 4 2 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 6	8.0 h (timeout)

[a]GLPK gives no solution within 8 hours.

TABLE 8. Results for 30 servers of each model.

Scene	Cluster load (of *MaxLoad*)	Solution result			End time
		Computation time	Minimal power (W)	Server frequency selection—$\sum_{j=0}^{f_i} z_{i,j}j$	
24	10%	15.908 s	1309.5	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 0 5 4 4 5 0 0 0 0 0 0 0 6 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8.0 h (timeout)
25	30%	0.215 s	3078.7	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 0 5 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6	8.0 h (timeout)
26	50%	0.612 s	4849.1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 0 0 7 0 0 0 0 0 7 0	8.0 h (timeout)
27	70%	0.481 s	7320.2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 2 0 0 0 0 0 0 0 2 2 2 2 5 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 6 7 7	8.0 h (timeout)
28	90%	/[a]	/[a]	/[a]	8.0 h (timeout)

[a]GLPK gives no solution within 8 hours.

find the global optimal solution within half a second or less. Therefore, when the cluster scale is small, we can directly solve the MILP problem using GLPK to optimize the cluster. We discuss the column ''server frequency selection'' in the next subsection.

E. APPLYING THE MILP-BASED SCHEME IN LARGE-SCALE CLUSTERS

In this subsection, we consider the last three cluster scales in Table 4. Similarly, we can determine their *MaxLoad*s. For each cluster scale, we use nine or five load scenes to test the MILP-based scheme. There are a total of 19 scenes, and the solution time is limited to 8 hours. Tables 6, 7, and 8 give the test results.

We first examine the columns ''computation time'', ''minimal power'', and ''end time'' in Tables 6-8. The results show that:

1) There are 9 scenes (8 in Table 6 and 1 in Table 7) in which GLPK can find the optimal solution (in 8 hours), and there are 10 scenes in which GLPK cannot find the optimal solution (in 8 hours).
2) There are only $10 * 4 = 40$ servers in scene 12, but GLPK fails to find the optimal solution.
3) In scenes 22 and 28, GLPK does not give any solution.
4) In the last 10 large-scale scenes (scenes 19-28), GLPK finds the optimal solution in only one scene (scene 21).

Therefore, when the cluster scale is large, GLPK is usually unable to obtain the optimal solution of the MILP problem

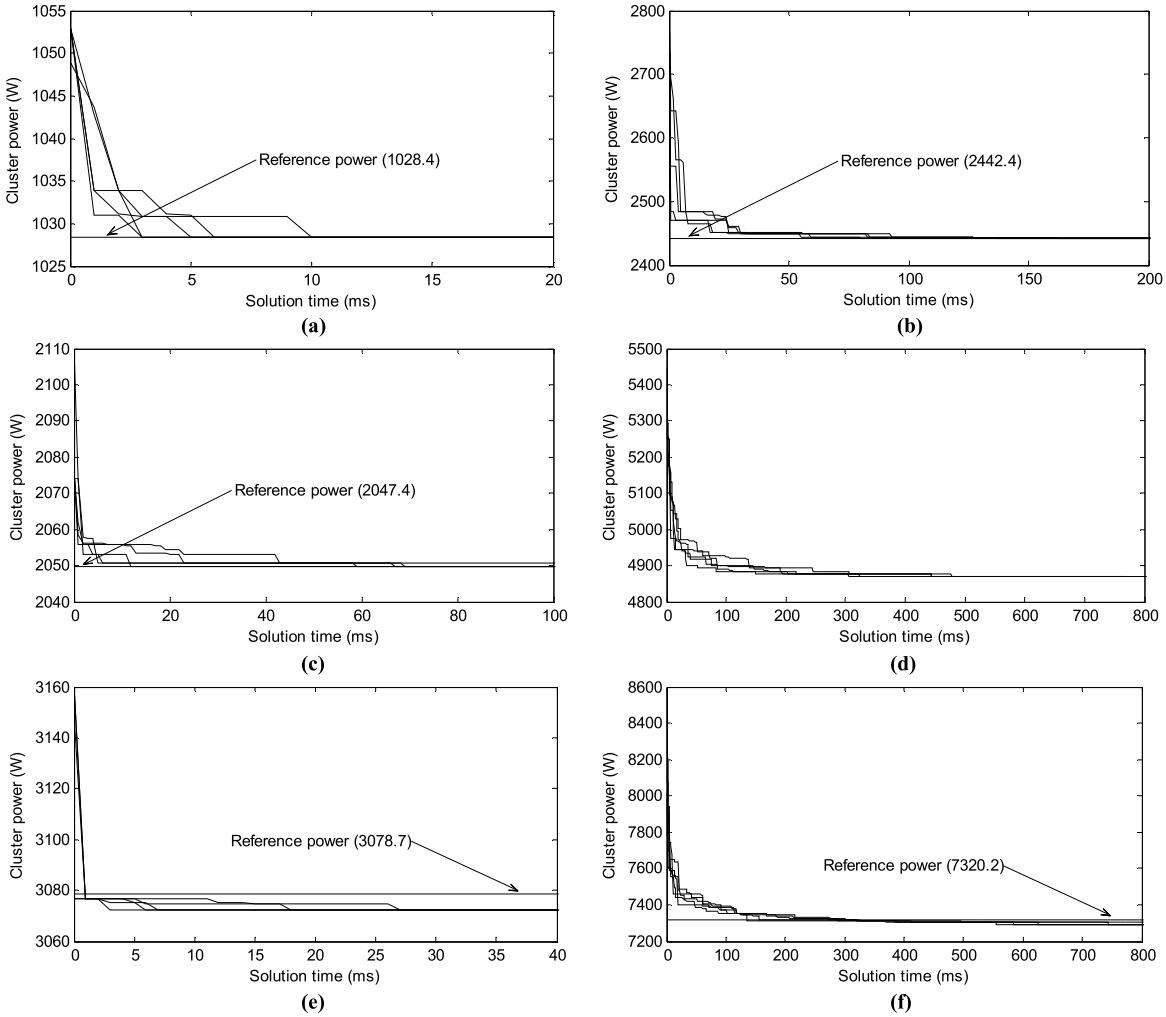


FIGURE 5. Solution efficiency of the MINLP-based scheme: (a) 10 servers of each model, 30% *MaxLoad* (scene 12); (b) 10 servers of each model, 70% *MaxLoad* (scene 16); (c) 20 servers of each model, 30% *MaxLoad* (scene 20); (d) 20 servers of each model, 70% *MaxLoad* (scene 22); (e) 30 servers of each model, 30% *MaxLoad* (scene 25); (f) 30 servers of each model, 70% *MaxLoad* (scene 27).

and sometimes even gives no solution, so the MILP-based scheme cannot be applied to the online optimization of large-scale clusters.

Next, we examine the column ''server frequency selection'' in Tables 5-8, which lists the frequency selection of each server. For example, 0 denotes that the server is power-off (standby), and 2 denotes that the server works at the 2nd frequency (from low to high). There are a total of 18 scenes (9 in Table 5, 8 in Table 6, and 1 in Table 7) in which GLPK finds the optimal solution in 8 hours. Of these 18 scenes, the optimal solution satisfies the hypothesis proposed in Subsection IV(B) in 17 scenes (all except scene 14). This indicates that the hypothesis is tenable in the majority of scenes.

F. SOLUTION QUALITY OF THE MINLP-BASED SCHEME FOR LARGE-SCALE CLUSTERS

We use the same 19 scenes as in Subsection V(E) to test the MINLP-based scheme. One second is an acceptable online solution time. In each scene, we run the FPA-based solution method for one second to obtain a minimal power. To evaluate the quality of the obtained minimal power, we need a reference standard. In each scene, we use the minimal power of the MILP problem that is found by GLPK within 8 hours as a reference power (the column ''minimal power'' in Tables 6-8) and deem it a high-quality solution. Four threads are used for parallelization in the MINLP-based scheme. According to [32] and [34], in the FPA, 0.8 is a good switch probability for most applications; therefore, the switch probability is set to 0.8 in our FPA-based solution method. In addition, the population size is set to 160. The results are shown in Table 9.

Table 9 shows that

1) In most scenes (all except scene 14 and 20), the MINLP-based scheme obtains a solution that is equal to or better than the reference standard.
2) In scene 14 and 20, although the obtained minimal power is more than the reference power, their difference is very small; the relative difference is only 0.07% and 0.1%, respectively.

TABLE 9. Solution quality of the MINLP-based scheme.

Scene	Cluster scale	Cluster load (of *MaxLoad*)	Minimal power (W)	Reference power (W)
10	10	10%	441.4	441.4[a]
11	servers	20%	730.2	730.2[a]
12	for each	30%	1028.4	1028.4
13	model	40%	1320.4	1320.4[a]
14		50%	1619.4	**1618.3**[a]
15		60%	1935.1	1935.1[a]
16		70%	2442.4	2442.4[a]
17		80%	2966.8	2966.8[a]
18		90%	3513.4	3513.4[a]
19	20	10%	**870.2**	874.0
20	servers	30%	2049.7	**2047.4**
21	for each	50%	3231.0	3231.0[a]
22	model	70%	**4868.0**	/[b]
23		90%	**7026.8**	7052.8
24	30	10%	**1308.4**	1309.5
25	servers	30%	**3072.2**	3078.7
26	for each	50%	**4848.0**	4849.1
27	model	70%	**7293.9**	7320.2
28		90%	**10,540.2**	/[b]

[a]The reference power is global optimal.
[b]The reference power is missing because GLPK gives no solution within 8 hours.

In a word, the MINLP-based scheme can find a high-quality solution within one second. It is worth noting that in scene 14, "1618.3 W" is the (global) optimal minimal power. As previously discussed, scene 14 is the only scene in which the optimal solution does not satisfy the hypothesis proposed in Subsection IV(B). In this case, even the optimal solution does not satisfy the hypothesis, but we still can obtain a very good near-optimal solution online based on the hypothesis.

G. SOLUTION EFFICIENCY OF THE MINLP-BASED SCHEME FOR LARGE-SCALE CLUSTERS

In this subsection, we use six scenes (scenes 12, 16, 20, 22, 25, and 27; three cluster scales and two load scenes for each scale) from the previous subsections to evaluate the solution efficiency of the MINLP-based scheme. When using our FPA-based method to solve the MINLP problem, we constantly record the current minimal power and the corresponding calculation time during the solution process. Each test is repeated five times. As with Subsection V(F), we use the minimal power of the MILP problem that is found by GLPK within 8 hours as a reference standard and deem it a high-quality solution (the reference standard for scene 22 is missing). The results are shown in Fig. 5. The dotted line in each subfigure is the reference standard. We can see that our FPA-based solution method has high efficiency, and it can converge to a high-quality solution in 300 ms or less.

According to the analysis in this and the previous subsections, the MINLP-based scheme can be applied to the online optimization of large-scale clusters.

VI. CONCLUSION

In this paper, we propose an online power-aware optimization strategy for application server clusters, whose goal is to minimize the cluster's power consumption while ensuring that the server's CPU utilization is not higher than a preset value. The optimization content involves the on/off state of each server, the CPU frequency of each running server, and the load distribution. The strategy includes two schemes: a MILP-based scheme and a MINLP-based scheme.

The MILP-based scheme formulates the cluster optimization problem as a MILP problem and uses GLPK to solve the problem. When the cluster scale is small, GLPK can find the global optimal solution quickly. However, when the cluster scale is large, the number of variables of the MILP problem will be enormous, and GLPK cannot obtain a satisfactory solution in a short amount of time and sometimes gives no solution. Therefore, the MILP-based scheme is only applicable to the online optimization of small-scale clusters.

In the MINLP-based scheme, we first formulate the cluster optimization problem as an NLP problem, whose number of variables is less than that of the MILP problem. We then propose a hypothesis to further reduce the number of variables. Based on the hypothesis, we redefine the variables and transform the NLP problem into a MINLP problem. The test results show that (i) the optimal solution satisfies the hypothesis in the majority of scenes; and (ii) even though the optimal solutions of some scenes may not satisfy the hypothesis, we still can obtain very good near-optimal solutions based on the hypothesis. Finally, we propose an FPA-based solution method for the MINLP problem that can quickly obtain a high-quality solution. Due to the small number of variables and the high solution efficiency, the MINLP-based scheme can be applied to the online optimization of large-scale clusters. Experimental results demonstrate the feasibility and validity of the two schemes.

We plan to study how to dynamically adjust the interval between two optimizations and extend our strategy to virtualized servers in future work.

ACKNOWLEDGMENT

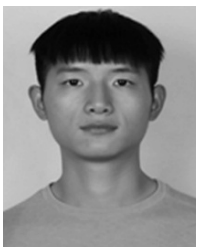
The authors would like to thank the anonymous reviewers for their comments and suggestions on improving the manuscript.

REFERENCES

[1] Z. Zhou, Jemal H. Abawajy, F. Li, Z. Hu, M. U. Chowdhury, A. Alelaiwi, and K. Li, "Fine-grained energy consumption model of servers based on task characteristics in cloud data center," *IEEE Access*, vol. 6, pp. 27080–27090, 2018.
[2] N. K. Sharma and G. R. M. Reddy, "Multi-objective energy efficient virtual machines allocation at the cloud data center," *IEEE Trans. Services Comput.*, vol. 12, no. 1, pp. 158–171, Jan./Feb. 2019.
[3] M. Zakarya, "Energy, performance and cost efficient datacenters: A survey," *Renew. Sustain. Energy Rev.*, vol. 94, pp. 363–385, Oct. 2018.
[4] J. Whitney and P. Delforge, "Data center efficiency assessment," Natural Resour. Defense Council, New York, NY, USA, White Paper IP: 14-08-A, 2014.
[5] P. Wang, Y. Qi, and X. Liu, "Power-aware optimization for heterogeneous multi-tier clusters," *J. Parallel Distrib. Comput.*, vol. 74, no. 1, pp. 2005–2015, 2014.
[6] J. Entrialgo, R. Medrano, D. F. García, and J. García, "Autonomic power management with self-healing in server clusters under QoS constraints," *Computing*, vol. 98, no. 9, pp. 871–894, 2016.
[7] X. Shi, J. Dong, S. M. Djouadi, Y. Feng, X. Ma, and Y. Wang, "PAPMSC: Power-aware performance management approach for virtualized Web servers via stochastic control," *J. Grid Comput.*, vol. 14, no. 1, pp. 171–191, 2016.

[8] 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.
[9] T. Enokido, D. Doulikun, and M. Takizawa, "An energy-aware load balancing algorithm to perform computation type application processes in a cluster of servers," *Int. J. Web Grid Serv.*, vol. 13, no. 2, pp. 145–169, 2017.
[10] M. Jiang, X. Luo, T. Miu, S. Hu, and W. Rao, "Are HTTP/2 servers ready yet?" in *Proc. 37th IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 1661–1671
[11] Y. Tian, C. Lin, and K. Li, "Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing," *Cluster Comput.*, vol. 17, no. 3, pp. 943–955, 2014.
[12] M. E. Gebrehiwot, S. Aalto, and P. Lassila, "Near-optimal policies for energy-aware task assignment in server farms," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2017, pp. 1017–1026.
[13] D. Cheng, Y. Guo, C. Jiang, and X. Zhou, "Self-tuning batching with DVFS for improving performance and energy efficiency in servers," *ACM Trans. Auton. Adapt. Syst.*, vol. 10, no. 1, 2015, Art. no. 6.
[14] K. Li, "Optimal power and performance management for heterogeneous and arbitrary cloud servers," *IEEE Access*, vol. 7, pp. 5071–5084, 2019.
[15] Z. Xiong, Q. Zhang, L. Cai, C. Zhu, and W. Cai, "Power-aware dynamic deployment under CPU utilization guarantee for application server cluster," *Wireless Pers. Commun.*, vol. 102, no. 2, pp. 1485–1502, 2018.
[16] Y. Cho and Y. M. Ko, "Energy efficiency of data center operating practices: Server clustering, powering on/off, and bang–bang control," *Networks*, vol. 71, no. 2, pp. 107–119, 2018.
[17] *GLPK (GNU Linear Programming Kit)*. Accessed: Jun. 9, 2019. [Online]. Available: http://www.gnu.org/software/glpk/
[18] L. Bertini, J. C. B. Leite, and D. Mossé, "Power optimization for dynamic configuration in heterogeneous Web server clusters," *J. Syst. Softw.*, vol. 83, no. 4, pp. 585–598, 2010.
[19] Z. Dong, N. Liu, and R. Rojas-Cessa, "Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers," *J. Cloud Comput.*, vol. 4, no. 1, p. 5, 2015.
[20] F. Yao, J. Wu, S. Subramaniam, and G. Venkataramani, "WASP: Workload adaptive energy-latency optimization in server farms using server low-power states," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jun. 2017, pp. 171–178.
[21] Y. Deng, Y. Hu, X. Meng, Y. Zhu, Z. Zhang, and J. Han, "Predictively booting nodes to minimize performance degradation of a power-aware Web cluster," *Cluster Comput.*, vol. 17, no. 4, pp. 1309–1322, 2014.
[22] A. L. Gouhar and A. Ali, "Power savings in servers infrastructure under backup and availability constraints," in *Proc. Int. Conf. Adv. Comput. Sci.*, 2018, pp. 1–8.
[23] W. Lin, W. Wang, W. Wu, X. Pang, B. Liu, and Y. Zhang, "A heuristic task scheduling algorithm based on server power efficiency model in cloud environments," *Sustain. Comput. Inform. Syst.*, vol. 20, pp. 56–65, Dec. 2018.
[24] D. Duolikun, T. Enokido, A. Aikebaier, and M. Takizawa, "Energy-efficient dynamic clusters of servers," *J. Supercomput.*, vol. 71, no. 5, pp. 1642–1656, 2015.
[25] L. S. Sousa, J. C. B. Leite, and O. Loques, "Green data centers: Using hierarchies for scalable energy efficiency in large Web clusters," *Inf. Process. Lett.*, vol. 113, nos. 14–16, pp. 507–515, 2013.
[26] K. Bilal, A. Fayyaz, S. U. Khan, and S. Usman, "Power-aware resource allocation in computer clusters using dynamic threshold voltage scaling and dynamic voltage scaling: Comparison and analysis," *Cluster Comput.*, vol. 18, no. 2, pp. 865–888, 2015.
[27] S. Mazumdar and M. Pranzo, "Power efficient server consolidation for cloud data center," *Future Gener. Comput. Syst.*, vol. 70, pp. 4–16, May 2017.
[28] A. M. Al-Qawasmeh, S. Pasricha, A. A. Maciejewski, and H. J. Siegel, "Power and thermal-aware workload allocation in heterogeneous data centers," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 477–491, Feb. 2015.
[29] L. Piga, R. A. Bergamaschi, M. Breternitz, and S. Rigo, "Adaptive global power optimization for Web servers," *J. Supercomput.*, vol. 68, no. 3, pp. 1088–1112, 2014.
[30] N. M. Asghari, M. Mandjes, and A. Walid, "Energy-efficient scheduling in multi-core servers," *Comput. Netw.*, vol. 59, pp. 33–43, Feb. 2014.
[31] J. Song, T. Li, Z. Yan, J. Na, and Z. Zhu, "Energy-efficiency model and measuring approach for cloud computing," *J. Softw.*, vol. 23, no. 2, pp. 200–214, 2012.
[32] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Proc. Int. Conf. Unconventional Comput. Natural Comput.* (Lecture Notes in Computer Science), vol. 7445, 2012, pp. 240–249.
[33] (Nov. 2018). *OpenMP Application Programming Interface, Version 5.0, OpenMP Architecture Review Board*. [Online]. Available: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf
[34] A. Draa, "On the performances of the flower pollination algorithm—Qualitative and quantitative analyses," *Appl. Soft Comput.*, vol. 34, pp. 349–371, Sep. 2015.

ZHI XIONG received the Ph.D. degree in communication and information system from Wuhan University, China, in 2006. He is currently an Associate Professor with the Department of Computer Science and Technology, Shantou University, China. His research interests include server cluster, green computing, and cloud computing.



YU CHENG received the B.E. degree in information and computing science from Changsha University, China, in 2017. He is currently pursuing the M.E. degree in computer technology with Shantou University, China. His research interests include server cluster and information system security.

LINGRU CAI received the Ph.D. degree in system engineering from the Huazhong University of Science and Technology, China, in 2010. She is currently an Associate Professor with the Department of Computer Science and Technology, Shantou University, China. Her research interests include modeling and simulation, system dynamics, and game theory.

WEIHONG CAI received the Ph.D. degree in communication and information system from the South China University of Technology, China. He is currently a Professor with the Department of Computer Science and Technology, Shantou University, China. He is also the Director of the Engineering and Technology Research Center of Digital Content Management of Guangdong Province. His research interests include cloud computing, information security, and network communication.