控制与决策
Control and Decision

应用服务器集群能耗与性能平衡的在线实时优化

熊智， 赵悦源， 许建龙， 蔡玲如， 蔡伟鸿

引用本文:
熊智, 赵悦源, 许建龙, 等. 应用服务器集群能耗与性能平衡的在线实时优化[J]. 控制与决策, 2021, 36(11): 2589−2598.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2020.0559

应用服务器集群能耗与性能平衡的在线实时优化

熊 智[1,2†], 赵悦源[1], 许建龙[1,2], 蔡玲如[1,2], 蔡伟鸿[1,2]

(1. 汕头大学 计算机系，广东 汕头 515063；
2. 智能制造技术教育部重点实验室(汕头大学)，广东 汕头 515063)

摘 要: 如何根据负载状况实时优化应用服务器集群的部署,以在能耗与性能之间取得平衡是急需解决的重要问题. 对此,提出一种应用服务器集群能耗与性能平衡的在线实时优化策略,优化目标是最小化能耗与请求丢弃速率的加权值,优化内容包括各服务器的开关和CPU频率. 该策略包括小规模集群优化(SSCOpt)和大规模集群优化(LSCOpt)两种方案:前者定义大量的变量,将集群优化描述成线性混合整数规划问题,然后采用软件包求解;后者通过分析能耗和负载模型的特性定义很少的变量,将集群优化描述成非线性混合整数规划问题,并提出一种基于花朵授粉算法和变量融合的求解算法. 测试结果表明:当集群规模较小时,SSCOpt方案能快速求得全局最优部署;当集群规模较大时,LSCOpt方案能快速求得很好的次优部署.

关键词: 应用服务器集群；能耗性能平衡；实时优化；混合整数规划；花朵授粉算法；变量融合

中图分类号: TP272 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2020.0559 **开放科学(资源服务)标识码(OSID):**

引用格式: 熊智,赵悦源,许建龙,等. 应用服务器集群能耗与性能平衡的在线实时优化[J]. 控制与决策, 2021, 36(11): 2589-2598.

Online real-time optimization of power-performance tradeoff for application server clusters

XIONG Zhi[1,2†], *ZHAO Yue-yuan*[1], *XU Jian-long*[1,2], *CAI Ling-ru*[1,2], *CAI Wei-hong*[1,2]

(1. Department of Computer Science，Shantou University，Shantou 515063，China；2. Key Laboratory of Intelligent Manufacturing Technology of Ministry of Education (Shantou University)，Shantou 515063，China)

Abstract: How to dynamically optimize the deployment of an application server cluster according to the load condition to balance the power and performance is an important issue that must be urgently solved. This paper proposes an online real-time optimization strategy for power-performance tradeoff for application server clusters, which aims to minimize the weighted value of the power and request discarding rate of a cluster. The optimization content involves the on/off state and CPU frequency of each server. The strategy involves two methods: small-scale cluster optimization (SSCOpt) and large-scale cluster optimization (LSCOpt). The SSCOpt defines a large number of variables to describe cluster optimization as a linear mixed integer programming problem, and then uses a software package to solve the problem. By analyzing the traits of power and load models, the LSCOpt defines a small number of variables to describe cluster optimization as a nonlinear mixed integer programming problem, and then proposes an solution algorithm based on flower pollination algorithm and variable fusion. The experimental results show that, the SSCOpt can get the global optimal deployment rapidly when used in small-scale clusters, and the LSCOpt can find a good near-optimal deployment rapidly even when applied to large-scale clusters.

Keywords: application server cluster；power-performance tradeoff；real-time optimization；mix integer programming；flower pollination algorithm； variable fusion

0 引 言

实际的Web系统通常采用Web服务器处理静态请求,采用应用服务器处理动态请求. 动态请求的响应内容是根据请求参数动态生成的,需要消耗较多的服务器资源. 应用服务器集群,以其可扩展、高可靠、高性价比等特性,已成为当前应用最为广泛的一种提高应用服务器性能和可靠性的解决方案,因此,大型电子商务平台、大型社交网站以及SaaS (software as

收稿日期: 2020-05-12；**修回日期:** 2020-09-04.
基金项目: 国家自然科学基金项目(61202366, 61702318)；广东省自然科学基金项目(2018A030313438).
责任编委: 刘德荣.
†通讯作者. E-mail: zxiong@stu.edu.cn.

a service,软件即服务)平台通常都采用应用服务器集群对外提供服务.

应用服务器集群有两个基本需求:一是要提升性能以提高服务质量;二是要节能以降低运营成本和减少排放[1]. 差的服务质量会导致差的用户体验,从而导致用户流失. 特别地,SaaS和电子商务等平台中的用户有很多都是付费获得服务的,并且服务质量的好坏会影响用户的生产和经营,因此,平台必须为他们提供性能有保证的服务. 服务器电能消耗所带来的运营成本已成为各平台运营商的主要开支之一,如果服务定价太高也会导致用户流失. 当前,节能减排是各个行业都要解决的重要问题[2-3],而服务器的巨大能耗给社会和环保带来了沉重的压力,预计到2020年,数据中心的碳排放将接近每年10亿吨,占全球总排放量的4%[4]. 因此,平台必须根据当前的负载状况,动态调整集群中各节点的部署以进行节能和减排. 然而,这两方面的需求是矛盾的,前者要求尽可能增大集群规模,而后者则需要尽可能减小集群规模. 如何根据实际的负载状况实时优化集群部署,以在能耗与性能之间取得平衡是急需解决的重要问题.

响应时间是直接影响用户体验的性能指标之一,因而成为众多相关研究所关注的对象[5-11]. 然而,与批处理作业和并行计算等任务不同,单个Web请求的处理时间很短,响应时间也很短,所以响应时间这一用户体验在很大程度上取决于用户的网络环境,在应用服务器上优化响应时间的意义不大. 由于应用服务器都会限制其并发服务的请求数量,当Web负载很重时,会有请求因为并发数已达到上限而被丢弃. 然而,很多研究采用无限容量的排队模型(例如$M/M/1$或$M/M/n$等)[3,12]对服务器进行建模,从而忽略了请求丢弃率. 过高的请求丢弃率同样会使用户的体验非常糟糕,因此,对于Web应用而言,请求丢弃率也是直接影响用户体验的重要性能指标,需要得到优化. 与响应时间不同的是,请求丢弃率主要取决于应用服务器,可以较准确地进行优化.

CPU动态频率调整和服务节点动态开关是集群优化的重要措施. 基于规划问题的优化方法定义变量表示节点的开关和频率,将集群优化描述成规划问题,然后求解. 该方法因能获得最优或次优的部署而被广泛采用[4,6,13-19]. 集群的优化是一个复杂的带约束的规划问题,如果求解时间过长,将无法做到细粒度的优化,会导致较高的性能违约率[20]. 在规划变量的定义方面,已有的研究大多都是针对单个节点来定义规划变量[13-14,17-18],当集群规模很大时,规划问题的变量数目巨大,难以在线实时求解. 在规划问题的求解方面,有些研究仅采用启发式算法求解[15,18,21],无法求得最优解. 虽然有些研究提出了基于Benders分解[4]和遗传算法[19]等求解方法,但他们未对求解的实时性进行评测. 文献[13]意识到求解计算量大的问题,提出了离线构造“负载-解”的表,然后在线查表求解. 但是,对大规模集群构建该表的计算量巨大,不太可行,因为对单个负载进行求解的计算量本身就巨大,而大规模集群能承担的负载范围很大,表的行数也将巨大.

本文提出一种应用服务器集群能耗与性能平衡的在线实时优化策略,优化的目标是最小化集群功耗与请求丢弃速率的加权值,可通过权值来调整两者的重要程度. 该策略包括SSCOpt(small-scale cluster optimization)和LSCOpt(large-scale cluster optimization)两个方案. SSCOpt方案针对节点的频率区间定义变量,将集群优化描述成线性混合整数规划(mixed integer programming, MIP)问题,然后采用工具包进行精确求解. LSCOpt方案通过分析功耗和负载模型的特性,针对节点型号定义变量,将集群优化描述成非线性MIP问题,然后提出一种基于花朵授粉算法[22]和变量融合的高效求解算法——FPVF(flower pollination, variable fusion).

1 SSCOpt优化方案

本节首先给出相关的定义及符号,然后推导负载和功耗模型,最后介绍SSCOpt方案对集群优化问题的描述和求解.

1.1 定义及符号

定义1 平均请求速率定义为负载.

考虑一个包括m个服务节点的集群,节点记为$S_i(1\leqslant i\leqslant m)$,节点$S_i$的相关参数如表1所示.

表1 服务节点S_i的相关参数

参数	含义
c_i	离散频率的数量
$f_{i,j}$	从低到高的第j个(从1开始)离散频率
$l_{i,j}$	$f_{i,j}$的负载能力(参见定义3)
$p_{i,j}$	$f_{i,j}$的满载功耗(参见定义3)
p_i^{off}	关机时的功耗

CPU的频率调整有两种模式:离散模式和连续模式,前者让CPU工作在一个离散频率上,后者让CPU在相邻的离散频率之间切换以等效工作在连续频率上. 由于后者具有细粒度优化和函数连续性的优势[17],本文采用连续模式.

定义2 将一个周期内节点S_i的CPU频率在两相邻离散频率$f_{i,j}$与$f_{i,j+1}$之间切换,且工作在

$f_{i,j}$ 上的时间与周期的比值等于 $\eta(\eta \in [0,1])$, 定义为频率 $[f_{i,j}, f_{i,j+1}, \eta]$, 将 η 称为切换因子, 将 $\eta f_{i,j} + (1-\eta) f_{i,j+1}$ 称为频率 $[f_{i,j}, f_{i,j+1}, \eta]$ 的等效频率.

定义3 当节点工作在某个频率上时, 在请求丢弃率低到可以忽略(例如低于0.1%)的情况下, 它能承担的最大负载定义为该频率的负载能力, 承担该最大负载时的功耗称为该频率的满载功耗.

将预测的集群负载记为 L, 系统允许的最大请求丢弃率记为 D(请求丢弃率过高会导致用户流失), 集群功耗与请求丢弃速率重要程度的比值记为 $1/\sigma$.

1.2 负载和功耗模型

下面考虑节点 S_i, 假设其等效频率为 $f \in (f_{i,j}, f_{i,j+1})$, 根据定义2可知, 切换因子为

$$\eta = \frac{f_{i,j+1} - f}{f_{i,j+1} - f_{i,j}}. \tag{1}$$

通常CPU有很多离散频率, 且离散频率之间的间隔很小, 再加上频率 f 仅由 $f_{i,j}$ 与 $f_{i,j+1}$ 之间的切换来模拟, 而与其他离散频率没有关系, 因此, 本文采用线性插值来计算频率 f 的负载能力, 即

$$\mathrm{load}_i(f) = \eta l_{i,j} + (1-\eta) l_{i,j+1}. \tag{2}$$

根据等效频率的定义, 频率 f 表示CPU频率在 $f_{i,j}$ 与 $f_{i,j+1}$ 之间切换, 且工作在 $f_{i,j}$ 和 $f_{i,j+1}$ 上的时间分别占 η 和 $1-\eta$, 从而其满载功耗为

$$\mathrm{power}_i(f) = \eta p_{i,j} + (1-\eta) p_{i,j+1}. \tag{3}$$

因此, $\mathrm{load}_i(f)$ 是一系列点 $(f_{i,j}, l_{i,j})$ 的分段线性插值, $\mathrm{power}_i(f)$ 是一系列点 $(f_{i,j}, p_{i,j})$ 的分段线性插值.

1.3 优化问题的描述

针对每个节点的每对相邻离散频率定义3个变量, 具体地, 为节点 S_i 的相邻离散频率 $f_{i,j}$ 和 $f_{i,j+1}$ 定义一个二进制变量 $z_{i,j}$ 以及两个实数变量 $x_{i,j}$ 和 $y_{i,j}$. 如果 $z_{i,j}=1$, 则表示 S_i 在频率 $f_{i,j}$ 与 $f_{i,j+1}$ 之间切换, 并且 $x_{i,j}$ 表示切换因子, $y_{i,j}$ 表示1减去切换因子; 如果 $z_{i,j}=0$, 则规定 $x_{i,j}=y_{i,j}=0$. 因此, $x_{i,j}+y_{i,j}=z_{i,j}$.

对于节点 S_i, 显然至多只有一个 $z_{i,j}$ 等于1, 如果所有 $z_{i,j}$ 都为0, 则该节点关机. 补充定义满足 $z_{i,0} = x_{i,0}+y_{i,0} = 1-\sum_{j=1}^{c_i-1} z_{i,j}$ 的3个变量 $z_{i,0}$、$x_{i,0}$ 和 $y_{i,0}$. 其中: $z_{i,0}$ 为二进制变量, 表示节点 S_i 是否关机; $x_{i,0}$ 和 $y_{i,0}$ 为实数变量. 由于关机时的功耗为 p_i^{off}, 负载为0, 定义 $p_{i,0}=p_i^{\mathrm{off}}, l_{i,0}=0$, 并规定 $y_{i,0}=0$(即如果 $z_{i,0}=1$, 则 $x_{i,0}=1$).

注意到 $z_{i,j}$ 的取值只能为0或1, 且当 $z_{i,j}=0$ 时 $x_{i,j}=y_{i,j}=0$, 故有 $z_{i,j}x_{i,j}=x_{i,j}, z_{i,j}y_{i,j}=y_{i,j}$. 根据式(2)和(3), 以及上述补充定义, 节点 S_i 的功耗和承担的负载(包括关机情况下的)计算如下:

$$\mathrm{power}_i = \sum_{j=0}^{c_i-1}(z_{i,j}(x_{i,j}p_{i,j}+y_{i,j}p_{i,j+1})) = \sum_{j=0}^{c_i-1}(x_{i,j}p_{i,j}+y_{i,j}p_{i,j+1}), \tag{4}$$

$$\mathrm{load}_i = \sum_{j=0}^{c_i-1}(z_{i,j}(x_{i,j}l_{i,j}+y_{i,j}l_{i,j+1})) = \sum_{j=0}^{c_i-1}(x_{i,j}l_{i,j}+y_{i,j}l_{i,j+1}). \tag{5}$$

于是, 集群能耗与性能平衡的优化可以描述成如下规划问题:

$$\begin{aligned}
&\min \sum_{i=1}^{m}\mathrm{power}_i + \sigma\Big(L - \sum_{i=1}^{m}\mathrm{load}_i\Big).\\
&\text{s.t. } (1-D)L \leqslant \sum_{i=1}^{m}\mathrm{load}_i \leqslant L;\\
&\sum_{j=0}^{c_i-1} z_{i,j} = 1,\ \forall i;\\
&x_{i,j}+y_{i,j} = z_{i,j},\ \forall i,j;\\
&z_{i,j} \in \{0,1\},\ \forall i,j;\\
&x_{i,j}, y_{i,j} \in [0,1],\ \forall i,j;\\
&y_{i,0} = 0, \forall i.
\end{aligned} \tag{6}$$

其中: 目标函数由两部分加权构成, 前一部分是集群功耗, 即每秒的耗电量, 后一部分是集群的请求丢弃速率, 即每秒的请求丢弃数, 由权重系数 σ 决定这两部分的相对重要程度; 第1个约束条件右边的小于等于是保证集群的实际负载不超过预测的负载, 左边的小于等于是保证请求丢弃率不超过 D. 式(4)和(5)都是线性的, 并且该规划问题的目标函数和约束条件也都是线性的, 因此, 该规划问题是一个线性MIP问题.

一旦求解该规划问题得到各个 $x_{i,j}$、$y_{i,j}$ 和 $z_{i,j}$ 的值, 便可计算出节点 S_i 的负载为 load_i, 集群的实际负载为 $\sum_{i=1}^{m}\mathrm{load}_i$, 应该丢弃的负载为 $L-\sum_{i=1}^{m}\mathrm{load}_i$. 因此, 当一个请求到达集群前端的分配器后, 分配器以 $\Big(L-\sum_{i=1}^{m}\mathrm{load}_i\Big)\Big/L$ 的概率丢弃该请求, 以 load_i/L 的概率将该请求调度给节点 S_i.

1.4 优化问题的求解

GLPK (GNU linear programming kit, GNU线性规划工具)[23] 是用于建立线性规划和混合型整数规划问题的建模语言工具包, 同时可以用来对模型进行最优化求解. 本文采用GLPK来求解上述线性MIP问

题. 然而,该问题为每个节点的每对相邻离散频率定义3个变量,整数和实数变量的数目分别为$\sum_{i=1}^{m} c_i$和$2\times\sum_{i=1}^{m} c_i$,当集群规模较大时,变量数目巨大,无法在线实时求解. 因此,SSCOpt方案仅能适用于小规模集群的在线实时优化.

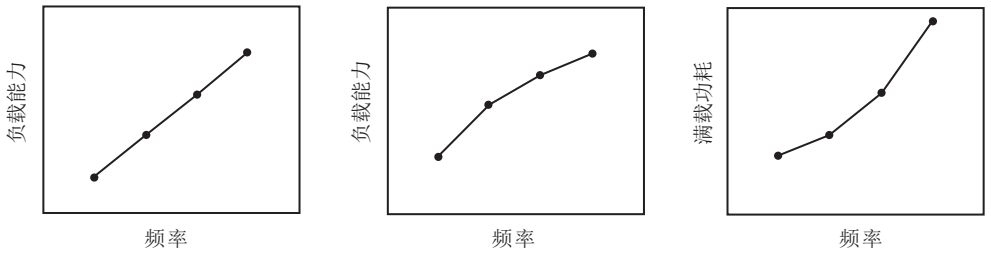
2 LSCOpt优化方案

在实际的集群中,服务器通常是一批一批采购的,同一批采购的服务器中有很多都是同型号(同构)的. 如果能够针对节点型号来定义变量,则变量的数目将大幅度减少. 基于这一思想,本节提出LSCOpt优化方案. 首先通过分析负载和功耗模型的特性说明针对节点型号定义变量的可行性,然后针对节点型号定义变量对集群优化问题进行重新描述,最后提出求解算法.

2.1 负载和功耗模型的特性

先看点$(f_{i,j}, l_{i,j})$的变化趋势. $l_{i,j}$是频率$f_{i,j}$的负载能力,显然,$l_{i,j}$随着$f_{i,j}$的增大而增加. 有两种情况: 第1种情况,系统中绝大部分动态请求仅仅消耗CPU资源,因此$l_{i,j}$与$f_{i,j}$成比例; 第2种情况,系统中绝大部分动态请求除了消耗CPU资源,还需要消耗较多的内存资源,甚至需要访问磁盘和网络,因此,$l_{i,j}$无法随着$f_{i,j}$的增大而线性增加,而是会向下弯曲,即是一个递增的凹函数. 再看点$(f_{i,j}, p_{i,j})$的变化趋势. $p_{i,j}$是频率$f_{i,j}$的满载功耗,相关研究通常将功耗表达成频率的3次或更高次的函数[1,10,16,24],因此,点$(f_{i,j}, p_{i,j})$是一个递增的凸函数.

结合前面的分析,曲线$\mathrm{load}_i(f)$是一系列点$(f_{i,j}, l_{i,j})$的分段线性插值,曲线$\mathrm{power}_i(f)$是一系列点$(f_{i,j}, p_{i,j})$的分段线性插值,$\mathrm{load}_i(f)$和$\mathrm{power}_i(f)$曲线的示意图(假设有4个离散频率)如图1所示.



(a) $\mathrm{load}_i(f)$的情况1 (b) $\mathrm{load}_i(f)$的情况2 (c) $\mathrm{power}_i(f)$

图1 负载能力和满载功耗曲线示意图

根据凹函数和凸函数的性质可以得到如下结论:

命题1 对于相同型号的节点来说,当它们工作在相同的等效频率(且承担相同的负载)时,可以得到集群优化的最优部署.

这是因为,如果两个同型号节点的等效频率分别为f_a和$f_b(f_a \neq f_b)$,现将它们的等效频率都调整为$(f_a+f_b)/2$,则很容易证明:

1)它们的负载能力之和在第1种情况下为

$$2\times\mathrm{load}_i\Big(\frac{f_a+f_b}{2}\Big)=\mathrm{load}_i(f_a)+\mathrm{load}_i(f_b);\quad(7)$$

在第2种情况下为

$$2\times\mathrm{load}_i\Big(\frac{f_a+f_b}{2}\Big)\geqslant\mathrm{load}_i(f_a)+\mathrm{load}_i(f_b).\quad(8)$$

其中: 如果f_a和f_b在同一线性区间中,则式(8)中的等号成立; 如果f_a和f_b不在同一线性区间中,则式(8)中的大于号成立. 即它们的负载能力之和不小于调整之前的.

2)它们的满载功耗之和为

$$2\times\mathrm{power}_i\Big(\frac{f_a+f_b}{2}\Big)\leqslant\mathrm{power}_i(f_a)+\mathrm{power}_i(f_b),\quad(9)$$

即它们的满载功耗之和不高于调整之前的. 因此,调整后的部署要么比调整前更优,要么与调整前持平,从而命题1成立. 但是,如果让相同型号的节点工作在相同的等效频率(且承担相同的负载),则可以针对节点型号定义规划变量,从而大幅度减少变量的数目. 因此,针对节点型号定义变量是可行的.

2.2 优化问题的重新描述

假设集群中的节点有M个型号$T_i(1\leqslant i\leqslant M)$,$T_i$型号节点的数量为$N_i$,其相关参数沿用表1中的符号,只不过将小写字母变成大写. 例如: $F_{i,j}$表示从低到高的第j个离散频率,$L_{i,j}$表示$F_{i,j}$的负载能力,P_i^{off}表示关机时的功耗. 对于T_i型号的节点,根据式(1)~(3),如果其频率$f\in(F_{i,j}, F_{i,j+1})$,则其负载能力和满载功耗分别为

$$\mathrm{Load}_i(f)=\frac{F_{i,j+1}-f}{F_{i,j+1}-F_{i,j}}L_{i,j}+\frac{f-F_{i,j}}{F_{i,j+1}-F_{i,j}}L_{i,j+1},\quad(10)$$

$$\mathrm{Power}_i(f)=\frac{F_{i,j+1}-f}{F_{i,j+1}-F_{i,j}}P_{i,j}+\frac{f-F_{i,j}}{F_{i,j+1}-F_{i,j}}P_{i,j+1}.\quad(11)$$

针对节点型号T_i定义两个变量: 整数n_i和实数f_i, 它们表示有n_i个T_i型号的节点在等效频率f_i上提供服务(其余的N_i-n_i个节点关机), 因此, 集群能耗与性能平衡的优化问题可重新描述成如下规划问题:

$$\begin{aligned}&\min\sum_{i=1}^{M}(n_i\mathrm{Power}_i(f_i)+(N_i-n_i)P_i^{\mathrm{off}})+\\&\quad\sigma\Big(L-\sum_{i=1}^{M}(n_i\mathrm{Load}_i(f_i))\Big).\\&\text{s.t. }(1-D)L\leqslant\sum_{i=1}^{M}(n_i\mathrm{Load}_i(f_i))\leqslant L;\\&\quad n_i\in\{0,\ldots,N_i\},\ \forall i;\\&\quad f_i\in[F_{i,1},F_{i,C_i}],\ \forall i.\end{aligned}\tag{12}$$

上述规划问题中的第1个约束条件比较复杂, 不利于问题的求解. 因此令$l_i=\mathrm{Load}_i(f_i)$, 并用l_i作为新的变量替换掉原来的变量f_i. 由式(10)和(11)很容易将$\mathrm{Power}_i(f_i)$写成l_i的函数, 即

$$\begin{aligned}&\mathrm{Power}'_i(l_i)=\\&\frac{L_{i,j+1}-l_i}{L_{i,j+1}-L_{i,j}}P_{i,j}+\frac{l_i-L_{i,j}}{L_{i,j+1}-L_{i,j}}P_{i,j+1}.\end{aligned}\tag{13}$$

于是, 规划问题(12)变为如下形式:

$$\begin{aligned}&\min\sum_{i=1}^{M}(n_i\mathrm{Power}'_i(l_i)+(N_i-n_i)P_i^{\mathrm{off}})+\\&\quad\sigma\Big(L-\sum_{i=1}^{M}(n_il_i)\Big).\\&\text{s.t. }(1-D)L\leqslant\sum_{i=1}^{M}(n_il_i)\leqslant L;\\&\quad n_i\in\{0,\ldots,N_i\},\ \forall i;\\&\quad l_i\in[L_{i,1},L_{i,C_i}],\ \forall i.\end{aligned}\tag{14}$$

这是一个非线性MIP问题, 但是它的变量数目很少, 整数和实数变量的数目都仅为M.

2.3 FPVF求解算法

花朵授粉算法是2012年提出的一种新型元启发式群体智能优化算法, 其基本思想来源于对自然界花朵自花授粉和异花授粉的模拟, 它采用浮点数编码, 在连续空间进行优化计算. 异花授粉表现为全局搜索方式, 对应全局授粉操作, 而自花授粉表现为局部寻优方式, 对应局部授粉操作, 这两种操作通过转换概率进行控制. 花朵授粉算法具有参数少、结构简单、全局搜索能力强等优点, 因此, 本文采用它来求解非线性MIP问题(14).

已有的研究在采用智能优化算法求解MIP问题时, 都将整数变量和实数变量一起用相同的参数(例如, 缩放因子、交叉概率、变异概率等)进行变异、交叉等操作, 然后对整数变量多增加一个取整(即离散化)操作[25-26]. 然而, 之所以有的变量是整数, 有的变量是实数, 这表明它们在意义、单位、尺度和取值范围等方面是不同的, 因此, 将它们混在一起操作是不合理的. 根据这里非线性MIP问题的特点, 本文提出一种基于花朵授粉算法和变量融合的求解算法FPVF, 它在传统花朵授粉算法的基础上, 在花朵授粉操作之前引入变量结合操作, 在花朵授粉操作之后依次引入变量矫正和变量分离操作.

变量结合操作将n_i与l_i相乘, 结合成k_i. 首先, k_i是有意义的, 它表示T_i型号节点的总负载; 其次, 变量数目减半, 减小了授粉操作的计算开销; 最后, 变量结合之后便于规划问题(14)中第1个约束条件的处理. 变量分离操作通过搜索最优的n_i, 将k_i拆分成n_i与l_i的乘积, 衡量最优的标准是使得目标函数中相关的部分$n_i\mathrm{Power}'_i(l_i)-n_iP_i^{\mathrm{off}}$取得最小值. 由于$n_i$为整数且范围通常不大, 并且需要被拆分的$k_i$的数量也很少, 搜索开销将非常小.

满足约束条件的个体称为可行个体, 否则称为不可行个体. 经过授粉操作后得到的个体, 以及随机产生的初始种群中的个体, 可能是不可行个体, 变量矫正操作采用矫正算法将不可行个体改造成可行个体. 矫正算法必须开销很小, 并且能兼顾迭代的收敛速度和种群的多样性, 因此, 算法结合贪婪和随机思想进行, 具体包括: 1) 当可以关闭节点时, 优先关闭最大能耗效率最低的节点型号; 2) 当需要开启节点时, 优先开启最大能耗效率最高的节点型号; 3) 当需要调高负载时, 随机挑选节点型号; 4) 当需要调低负载时, 挑选最大能耗效率最低的节点型号. 其中节点型号T_i的最大能耗效率定义为

$$E_i=\max_{f\in[F_{i,1},F_{i,C_i}]}\frac{\mathrm{Load}_i(f)}{\mathrm{Power}_i(f)}.\tag{15}$$

由规划问题(14)的最后两个约束条件可知, k_i的可行范围为$k_i\in\{0\}\bigcup[L_{i,1},N_iL_{i,C_i}]$. 矫正算法的流程见算法1.

算法1　矫正算法.

输入: 各节点型号对应的k_i, 集群负载L, 最大请求丢弃率D, 各节点型号对应的参数;

输出: 各节点型号对应的k_i.

01: if $(k_i<0)k_i=0$;

02: else if $(k_i<L_{i,1}/2)k_i=0$;

03: else if $(k_i<L_{i,1})k_i=L_{i,1}$;

04: else if $(k_i>N_iL_{i,C_i})k_i=N_iL_{i,C_i}$;

05: $l=\sum_{i=1}^{M}k_i$;
06: while $(l>L)$\{
07: 从$\{i|0<k_i<l-L\}$中挑E_i最小的i;
08: if (i存在){//关闭所有T_i型号的节点
09: $l=l-k_i$;
10: $k_i=0$;
11: }
12: else {//调低负载
13: 从$\{i|k_i>L_{i,1}\}$中挑选E_i最小的i;
14: if $(k_i-(l-L)<L_{i,1})$ {
15: $l=l-(k_i-L_{i,1})$
16: $k_i=L_{i,1}$;
17: }
18: else {
19: $k_i=k_i-(l-L)$;
20: return;
21: }
22: }
23: }
24: while $(l<L(1-D))$\{
25: 从$\{i|0<k_i<N_iL_{i,C_i}\}$ 中随机选取i;
26: if (i存在){//调高负载
27: if $(L(1-D)-l>N_iL_{i,C_i}-k_i)$\{
28: $l=l+(N_iL_{i,C_i}-k_i)$;
29: $k_i=N_iL_{i,C_i}$;
30: }
31: else {
32: $k_i=k_i+(L(1-D)-l)$;
33: return;
34: }
35: }
36: else {//开启节点型号
37: 从$\{i|k_i=0\}$中挑选E_i最大的i;
38: if $(L(1-D)-l>N_iL_{i,C_i})$\{
39: $l=l+N_iL_{i,C_i}$;
40: $k_i=N_iL_{i,C_i}$;
41: }
42: else {
43: $k_i=L(1-D)-l$;
44: if $(k_i<L_{i,1})$\{ //确保 $k_i\geqslant L_{i,1}$
45: $\mathrm{d}t=L_{i,1}-k_i$;
46: $k_i=L_{i,1}$;
47: 从$\{i|k_i>L_{i,1}\}$中挑k_i最大的i;
48: $k_i=k_i-\mathrm{d}t$;
49: }
50: return;
51: }
52: }
53: }

在算法1中,第01～04行是将各k_i矫正到可行的范围;第06～23行是使得约束条件中的$\sum_{i=1}^{M}(n_il_i)\leqslant L$得到保证;第24～53行是使得约束条件中的$(1-D)L\leqslant\sum_{i=1}^{M}(n_il_i)$得到保证. 算法1中的一些关键地方也给出了相应的注释.

3 测 试

本节用不同规模的集群对两种优化方案进行测试,看其能否求得最优或次优部署,以及求解的效率.

3.1 测试场景

测试用的集群包含4种型号的服务节点,表2给出了它们的频率、功耗和性能数据[13]. 各型号节点关机(挂起到内存)时的待机功耗设为3.5 W. 测试总共涉及3种规模的集群,它们均由这4种型号的节点构成,各型号节点的数量相等,分别为5、20和80台. 假定请求丢弃率的上限D为10 %. 优化控制器的配置为Intel Core i7-4790的CPU和8 G的内存.

表2 各节点型号的相关数据

型号	CPU	离散频率数量	频率/GHz,满载功耗/W,负载能力/(req/s)
1	AMD Athlon 64 3500+	4	(1.0, 74.7, 51.2), (1.8, 95.7, 91.2), (2.0, 103.1, 101.4), (2.2, 110.6, 111.4)
2	AMD Athlon 64 3580+	5	(1.0, 77.0, 55.3), (1.8, 89.0, 96.9), (2.0, 100.0, 107.0), (2.2, 115.0, 115.8), (2.4, 136.0, 124.6)
3	AMD Athlon 64 5000+ (dual core)	6	(1.0, 82.5, 99.4), (1.8, 99.2, 177.4), (2.0, 107.3, 197.2), (2.2, 116.6, 218.0), (2.4, 127.2, 234.6), (2.6, 140.1, 255.2)
4	Intel Pentium-M 1.8 G	7	(0.6, 44.0, 37.3), (0.8, 45.0, 50.0), (1.0, 47.0, 62.4), (1.2, 49.0, 74.4), (1.4, 51.0, 88.4), (1.6, 55.0, 97.6), (1.8, 60.0, 111.4)

图2给出了这4种节点型号的能耗效率(即负载/功耗)曲线. 型号1和型号4的能耗效率随着频率的增加而递增,在最高频率处取得最大值;而型号2和型号3的能耗效率曲线非严格递增,效率最大的频率并非最高频率. 启发式算法尽可能让开启的节点工作在最高频率上,以关闭尽可能多的节点,然而上述情形将使得它们无法求得最优部署. 同时,上述情形还表明,本文测试用的集群较为复杂,很难求得其

最优部署.

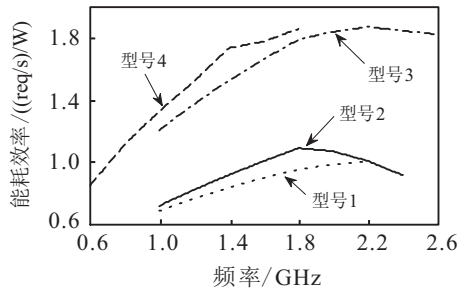


图 2　能耗效率曲线

3.2　SSCOpt方案的优化效果

在每种集群规模下,各节点的最大负载能力之和为该集群能承担的最大负载,记为MaxLoad,下文用负载与MaxLoad的比值来表示集群负载的大小,例如50%的负载是指负载等于MaxLoad×50%. 在每种集群规模下,考虑20%、50%和80%三种负载情况,每种负载情况下又考虑3个σ值的场景,为了具有代表性,选择的3个σ值分别使得最优解对应的请求丢弃率等于D、介于$0\sim D$之间,以及等于0.

对于每个场景,采用GLPK中的glpsol命令求解SSCOpt方案中的线性MIP问题. 在求解过程中glpsol不断输出当前找到的最优解,如果求解过程结束,则最后输出的解将是全局最优解;否则,最后输出的解不一定是全局最优解. 由于这里的求解结果将作为高质量的解供后面的测试参照,这里希望能够得到尽可能优秀的解,因而将最长求解时间设定为10 h. 表3～表5给出了求解结果.

表 3　集群规模为20时的结果(整数和实数变量的数目分别为110和220)

场景	负载	权重σ	求解结果				求解过程结束时间
			集群功耗/W	请求丢弃率/%	目标函数值	求解时间/ms	
1		0.30	347.1	10.00	365.2	3	18 ms
2	20%	0.50	352.5	7.57	375.3	1	10 ms
3		0.70	383.8	0.00	383.8	17	1.2 s
4		0.40	772.2	10.00	832.5	186	1.5 s
5	50%	0.50	804.9	5.14	843.6	7	145 ms
6		0.60	848.5	0.00	848.5	33	130 ms
7		0.50	1 345.2	10.00	1 465.7	8	401 ms
8	80%	0.80	1 377.5	7.87	1 529.3	3	21 ms
9		1.10	1 556.4	0.00	1 556.4	9	20 ms

表 4　集群规模为80时的结果(整数和实数变量的数目分别为440和880)

场景	负载	权重σ	求解结果				求解过程结束时间
			集群功耗/W	请求丢弃率/%	目标函数值	求解时间	
10		0.30	1 387.4	10.00	1 459.7	1.0 s	超时
11	20%	0.45	1 410.0	7.57	1 492.1	67.0 s	67.2 s
12		0.60	1 507.7	0.00	1 507.7	47 ms	406.2 s
13		0.35	3 073.2	10.00	3 284.2	442 ms	超时
14	50%	0.50	3 106.5	8.76	3 370.5	74 ms	超时
15		0.65	3 388.4	0.00	3 388.4	75 ms	超时
16		0.60	5 332.0	10.00	5 910.5	38 ms	超时
17	80%	0.95	5 852.0	3.85	6 205.0	1 ms	18 ms
18		1.30	6 214.0	0.00	6 214.0	36 ms	35.1 s

表 5　集群规模为320时的结果(整数和实数变量的数目分别为1 760和3 520)

场景	负载	权重σ	求解结果				求解过程结束时间
			集群功耗/W	请求丢弃率/%	目标函数值	求解时间	
19		0.35	5 522.7	10.00	5 860.2	260 ms	112.0 s
20	20%	0.55	5 979.3	0.78	6 020.9	778 ms	超时
21		0.75	6 024.6	0.00	6 024.6	1.7 s	超时
22		0.20	12 278.1	10.00	12 760.2	672 ms	超时
23	50%	0.45	12 298.6	9.81	13 362.7	643 ms	超时
24		0.70	13 527.8	0.00	13 527.8	564 ms	超时
25		0.60	21 338.1	10.00	23 652.1	1.2 s	超时
26	80%	0.85	21 367.0	9.86	24 598.6	298 ms	超时
27		1.10	24 841.9	0.00	24 841.9	484 ms	超时

计算集群的优化部署包括两个步骤: 首先是预测下个阶段的负载,然后是求解规划问题得到优化部署. 1 s是一个可接受的在线实时计算优化部署的时间,但其包括预测时间和求解时间两部分,因此,本文认为0.5 s是一个可接受的在线实时求解时间.

在总体上,表3～表5中的结果表明,随着集群规模的增加, SSCOpt方案中线性MIP问题的整数和实数变量数目会增加,求解时间会变长. 下面先看表3. 最后一列的结果表明,在所有9种场景下,求解过程都在2 s内结束,因此所求得的解均是最优解. 再由倒数第2列可知,在所有9种场景下,glpsol均在200 ms内求得了最优解. 因此,对于该规模为20个节点的集群,SSCOpt方案能够做到在线实时优化.

再看表4和表5. 首先,在绝大多数场景下,求解过程都超时了,也就是说glpsol未在10 h内求得最优解;其次,在很多场景下,求解时间都超过了0.5 s. 因此,对于规模大于80个节点的集群,SSCOpt方案无法做到在线实时优化.

3.3 LSCOpt方案的优化效果

3.2小节的实验结果表明,SSCOpt方案能快速求得小规模集群的全局最优部署,因此,本小节仅考虑后2种较大规模的集群,仅在这2种规模下测试LSCOpt方案,测试用的负载和权重系数(即场景)与3.2小节完全一致. 对于LSCOpt方案中的非线性MIP问题,在各种集群规模下整数和实数变量的数目均为4个,其变量数目仅与服务器型号的数量相关,而与集群规模无关.

每次测试让FPVF算法运行0.5 s,求得一个目标函数的最小值. 为了评估该最小值的质量,需要参考对照的标准. 在3.2小节中,GLPK运行了10 h来求解SSCOpt方案中的线性MIP问题,其求得的解本文认为是高质量的,因此用其作为参考标准. 使用OpenMP[27]并行化个体的进化,线程数量为4. FPVF算法的参数设置如下:种群大小为80 (变量个数的10倍);转换概率为0.8[22, 28]. 表6给出了求解结果.

表6 优化效果

规模	场景	负载	权重σ	目标函数值	参考值
80	10	20%	0.30	1 460.2	**1 459.7**
	11		0.45	1 492.1	1 492.1
	12		0.60	1 507.8	**1 507.7**
	13	50%	0.35	**3 284.1**	3 284.2
	14		0.50	3 370.5	3 370.5
	15		0.65	**3 382.8**	3 388.4
	16	80%	0.60	5 910.5	5 910.5
	17		0.95	6 205.0	6 205.0
	18		1.30	6 214.0	6 214.0
320	19	20%	0.35	5 860.2	5 860.2
	20		0.55	6 020.9	6 020.9
	21		0.75	**6 022.0**	6 024.6
	22	50%	0.20	**12 758.3**	12 760.2
	23		0.45	**13 361.4**	13 362.7
	24		0.70	**13 526.6**	13 527.8
	25	80%	0.60	**23 641.4**	23 652.1
	26		0.85	**24 596.2**	24 598.6
	27		1.10	24 841.9	24 841.9

表6中的结果表明,在所有18种场景中,仅在10和12这两种场景下,FPVF算法(在0.5 s内)求得的目标函数值比参考值差(高),即在绝大多数场景下,FPVF算法(在0.5 s内)都能求得比参考值要好的解. 即使在10和12这两种场景下,FPVF算法求得的目标函数值比参考值仅高了不到0.034%. 因此,LSCOpt方案能够应用于大规模集群的在线实时优化.

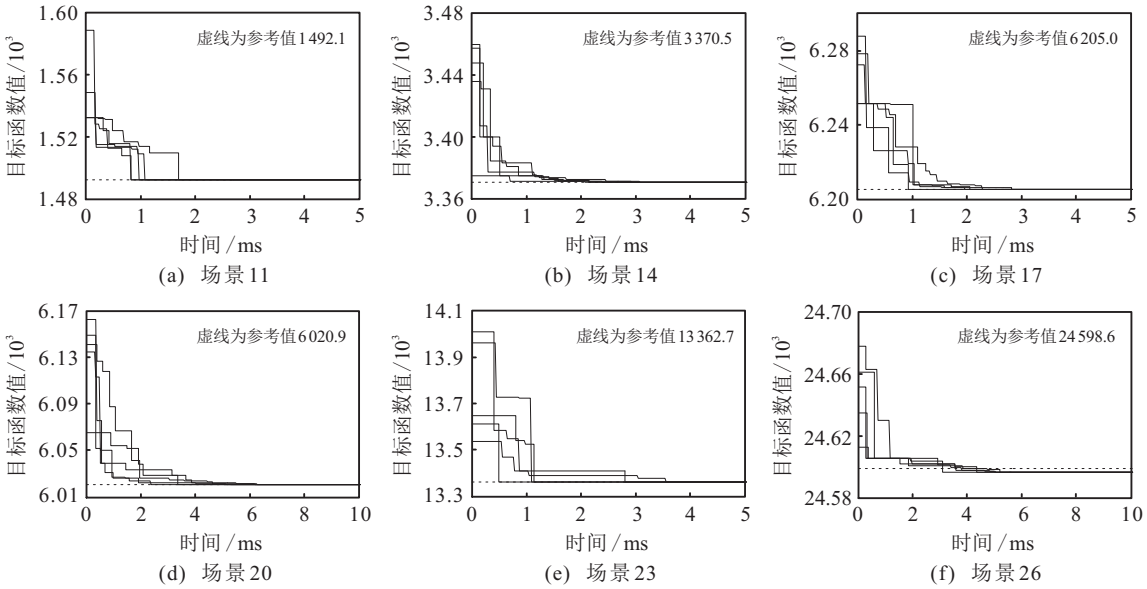


图3 求解效率

3.4 FPVF算法的求解效率

本小节采用前面的场景11、14、17、20、23和26来测试FPVF算法的收敛速度. 在求解过程中,不断记录下当前得到的目标函数值和所对应的求解时间. 每个实验重复5次,使用的参考标准与3.3小节一样. 测试结果如图3所示,图中的虚线表示参考值.

图3中的结果表明: 当集群规模为80时,在所有场景下,FPVF算法都能在5 ms内收敛到不比参考值差的解; 当集群规模为320时,在所有场景下,FPVF算法都能在10 ms内收敛到不比参考值差的解. 可见收敛速度非常快,这是因为规划问题的变量数目少,并且FPVF算法的求解效率高. 另外还可以看到,虽然随着集群规模的增长,规划变量的数目不会增加,但整数变量n_i的范围会变大,使得变量分离操作的开销会略有增加,因而求解时间总体上会略微增长. 此外,图3中的结果还表明,初始种群中的解已经比较优秀了,这是因为初始种群中的个体经过了基于贪婪思想的矫正操作,以及实施了最优搜索的变量分离操作,所以是较优秀的贪婪解.

4 结　论

本文提出了SSCOpt和LSCOpt两种优化方案,分别适用于小规模和大规模的集群. SSCOpt方案针对节点的频率区间定义大量的变量,将集群优化描述成线性MIP问题,然后采用GLPK求解. 实验结果表明,当集群规模较小时,该方案能在200 ms内求得全局最优部署. LSCOpt方案针对节点型号定义很少的变量,将集群优化描述成非线性MIP问题,然后提出一种基于花朵授粉算法和变量融合的高效求解算法FPVF. 实验结果表明,即使应用于大规模集群,该方案也能在10 ms内求得很好的次优部署. 因此,本文提出的优化策略能够在线实时进行. 近年来出现了一些花朵授粉算法的改进研究[29-30],对此,在接下来的研究中将参考这些改进措施,进一步提升FPVF算法的求解效率.

参考文献(References)

[1] 王继业, 周碧玉, 张法, 等. 数据中心能耗模型及能效算法综述[J]. 计算机研究与发展, 2019, 56(8): 1587-1603.
(Wang J Y, Zhou B Y, Zhang F, et al. Data center energy consumption models and energy efficient algorithms[J]. Journal of Computer Research and Development, 2019, 56(8): 1587-1603.)

[2] 朱光宇, 徐文婕. 考虑能耗与质量的机床构件生产线多目标柔性作业车间调度方法[J]. 控制与决策, 2019, 34(2): 252-260.
(Zhu G Y, Xu W J. Multi-objective flexible job shop scheduling method for machine tool component production line considering energy consumption and quality[J]. Control and Decision, 2019, 34(2): 252-260.)

[3] Chang C J, Chang F M, Ke J C. Optimal power consumption analysis of a load-dependent server activation policy for a data service center[J]. Computers and Industrial Engineering, 2019, 130: 745-756.

[4] Kwon S, Ntaimo L, Gautam N. Demand response in data centers: Integration of server provisioning and power procurement[J]. IEEE Transactions on Smart Grid, 2019, 10(5): 4928-4938.

[5] Cheng D Z, Guo Y F, Jiang C J, et al. Self-tuning batching with DVFS for performance improvement[J]. ACM Transactions on Autonomous and Adaptive Systems, 2015, 10(1): 1-32.

[6] Zhou Z, Liu F, Zou R, et al. Carbon-aware online control of geo-distributed cloud services[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(9): 2506-2519.

[7] Cai H R, Cao Q, Sheng F, et al. Montgolfier: Latency-aware power management system for heterogeneous servers[C]. Proceedings of the 35th IEEE International Performance Computing and Communications Conference. Piscataway: IEEE, 2016: 1-8.

[8] Shi X Y, Dong J, Djouadi S M, et al. PAPMSC: Power-aware performance management approach for virtualized web servers via stochastic control[J]. Journal of Grid Computing, 2016, 14(1): 171-191.

[9] Entrialgo J, Medrano R, García D F, et al. Autonomic power management with self-healing in server clusters under QoS constraints[J]. Computing, 2016, 98: 871-894.

[10] 张江强, 赵宁, 刘文奇. 具有两类请求的云计算中心服务器数量的优化[J]. 智能系统学报, 2017, 12(5): 601-607.
(Zhang J Q, Zhao N, Liu W Q. Optimization of the number of servers in a cloud computation center with two demand classes[J]. CAAI Transactions on Intelligent Systems, 2017, 12(5): 601-607.)

[11] Enokido T, Doulikun D, Takizawa M. An energy-aware load balancing algorithm to perform computation type application processes in a cluster of servers[J]. International Journal of Web and Grid Services, 2017, 13(2): 145-169.

[12] 孙健, 廖丹, 李可, 等. 基于排队论的异构数据中心性能及能源管理策略[J]. 电子科技大学学报, 2018, 47(2): 161-168.

(Sun J, Liao D, Li K, et al. A strategy for queuing theory-based performance and energy management in heterogeneous data centers[J]. Journal of University of Electronic Science and Technology of China, 2018, 47(2): 161-168.)

[13] Bertini L, Leite J C B, Mossé D. Power optimization for dynamic configuration in heterogeneous web server clusters[J]. Journal of Systems and Software, 2010, 83(4): 585-598.

[14] Cao J, Li K, Stojmenovic I. Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers[J]. IEEE Transactions on Computers, 2014, 63(1): 45-58.

[15] Mazumdar S, Pranzo M. Power efficient server consolidation for cloud data center[J]. Future Generation Computer Systems, 2017, 70: 4-16.

[16] Atiewi S, Yussof S, Rusli M E B, et al. A power saver scheduling algorithm using DVFS and DNS techniques in cloud computing data centres[J]. International Journal of Grid and Utility Computing, 2018, 9(4): 385-395.

[17] Xiong Z, Zhang Q K, Cai L R, et al. Power-aware dynamic deployment under CPU utilization guarantee for application server cluster[J]. Wireless Personal Communications, 2018, 102(2): 1485-1502.

[18] Hao Y, Cao J, Ma T, et al. Adaptive energy-aware scheduling method in a meteorological cloud[J]. Future Generation Computer Systems, 2019, 101: 1142-1157.

[19] Qi L Y, Chen Y, Yuan Y, et al. A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems[J]. World Wide Web, 2020, 23(2): 1275-1297.

[20] Monteiro A, Loques O. Quantum virtual machine: Power and performance management in virtualized web servers clusters[J]. Cluster Computing, 2019, 22(1): 205-221.

[21] Dong Z Q, Liu N, Rojas-Cessa R. Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers[J]. Journal of Cloud Computing, 2015, 4(1): 1-4.

[22] Yang X S. Flower pollination algorithm for global optimization[C]. Proceedings of the International Conference on Unconventional Computation and Natural Computation. Berlin: Springer, 2012: 240-249.

[23] Free Software Foundation, Inc. GLPK (GNU linear programming kit) [EB/OL]. (2012-06-23) [2020-04-10]. http://www.gnu.org/software/glpk/.

[24] Vasques T L, Moura P, Almeida A. A review on energy efficiency and demand response with focus on small and medium data centers[J]. Energy Efficiency, 2019, 12(5): 1399-1428.

[25] 吕志明, 王霖青, 赵珺, 等. 一种基于多代理模型的混合整数规划优化方法[J]. 控制与决策, 2019, 34(2): 362-368.
(Lyu Z M, Wang L Q, Zhao J, et al. A multi-surrogates algorithm for mixed-integer programming problems[J]. Control and Decision, 2019, 34(2): 362-368.)

[26] Shokrian M, High K A. Application of a multi objective multi-leader particle swarm optimization algorithm on NLP and MINLP problems[J]. Computers and Chemical Engineering, 2014, 60: 57-75.

[27] OpenMP Architecture Review Board. OpenMP application programming interface, version 5.0 [EB/OL]. (2018-11-15) [2020-04-10]. https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf.

[28] Draa A. On the performances of the flower pollination algorithm — Qualitative and quantitative analyses[J]. Applied Soft Computing, 2015, 34: 349-371.

[29] Zhou Y Q, Wang R, Luo Q F. Elite opposition-based flower pollination algorithm[J]. Neurocomputing, 2016, 188: 294-310.

[30] 沈艳军, 杨鑫, 刘允刚. 考虑需求响应的水火电优化调度改进型花朵授粉算法[J]. 控制与决策, 2019, 34(8): 1645-1653.
(Shen Y J, Yang X, Liu Y G. An improved flower pollination algorithm for hydrothermal scheduling incorporating demand response[J]. Control and Decision, 2019, 34(8): 1645-1653.)

作者简介

熊智 (1978–), 男, 副教授, 博士, 从事服务器集群、云计算和大数据应用等研究, E-mail: zxiong@stu.edu.cn;

赵悦源 (1996–), 男, 硕士生, 从事云计算和绿色计算的研究, E-mail: 18yyzhao2@stu.edu.cn;

许建龙 (1982–), 男, 讲师, 博士, 从事分布式计算和面向服务计算的研究, E-mail: xujianlong@stu.edu.cn;

蔡玲如 (1979–), 女, 副教授, 博士, 从事建模与仿真、系统动力学和大数据应用等研究, E-mail: lrcai@stu.edu.cn;

蔡伟鸿 (1963–), 男, 教授, 博士, 从事云计算和信息安全等研究, E-mail: whcai@stu.edu.cn.

(责任编辑：李君玲)