文章编号:1671-4598(2020)12-0217-05 DOI:10.16526/j.cnki.11—4762/tp.2020.12.045 中图分类号:TP391 文献标识码:A

一种自适应粒子群算法在云资源调度中的应用

张娟芝，段中兴，熊福力
（西安建筑科技大学 信息与控制工程学院，西安 710055）

摘要：通过资源调度优化提升云计算的效率并降低数据中心能耗是云计算领域的主要研究内容之一；针对粒子群算法在云计算资源调度应用中算法初期收敛速度快、后期收敛速度缓慢、易陷入局部寻优的缺点，提出了一种自适应改进的粒子群算法用于云计算资源调度问题的研究，该算法通过自适应改进粒子的个体学习因子和社会学习因子，以提高算法的全局探索能力，使得粒子逼近更优解；实验结果表明：自适应粒子群算法不仅具备良好的收敛性和全局寻优能力，同时能够大幅度降低云资源调度中任务队列的总完成时间。

关键词：自适应；粒子群算法；局部寻优；云资源

Application of an Adaptive Particle Swarm Algorithm in Cloud Scheduling

Zhang Juanzhi, Duan Zhongxing, Xiong Fuli
(College of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710055, China)

Abstract: Improving the efficiency of cloud computing and reducing the energy consumption of data center is one of the main research contents in cloud computing. Aiming at the shortcomings of particle swarm optimization (PSO) in cloud computing resource scheduling applications, an adaptive improved PSO algorithm is proposed for cloud computing resource scheduling problems. The algorithm improves the global exploration ability of the algorithm and makes the particles approach the better solution. The experimental results show that the adaptive PSO not only has good convergence and global searching excellent ability, and can greatly reduce the total completion time of the task queue in cloud resource scheduling.

Keywords: adaptive; particle swarm algorithm; local optimization; cloud resources

0 引言

随着物联网及5G技术的快速发展及大规模商用，未来时代必定是云计算的时代。这对云计算在高速网络和高性能计算系统方面有了更高的要求[1]。云计算是一种基于虚拟化技术的新型计算模式，它将大量用网络连接的不同位置及空间的数据中心的物理计算资源构成一个虚拟的资源池，并根据用户对计算能力、存储空间、网络带宽及其他信息服务的需求而实现统一的资源调度。因此，快速对各种计算资源进行合理的调度是云计算须处理的关键点及难点所在。

近几年大数据及云计算产业在全世界的兴起及普及，国内外学者对于云计算数据中心的研究方向已经从传统数据中心资源调度优化所关注的应用的稳定性、数据的安全可靠性等方面[2]。云计算发展过程中，资源分配机制始终是数据中心效能优化中亟待解决的问题，也是云计算领域内的研究热点[3]。

目前国内外大多数研究资源调度算法都是基于蚁群算法（ACO，ant colony algorithm）、QoS（quality of service）网格管理算法、粒子群算法（PSO，particle swarm optimization）、遗传算法（GA，genetic algorithm）、布谷鸟算法等智能算法[4]。对云计算而言，合理且高效地调度高度并发的任务十分必要。在这种情况下，分布式计算（distributed computing）、网格计算（grid computing）等混合演进形成了现如今较为成熟的云计服务模式和商业模型[5-6]。Keshanchi[7]等人使用改进遗传算法（GA，genetic algorithm）在云环境中进行任务调度，其中GA采用精英选择的方法防止早熟收敛。基于粒子群算法（PSO，particle swarm optimization）等智能算法的云计算资源调度与优化也成为了研究的热点[8]。如Mathiyalagan[9]等人引入了适用于网格计算编程的粒子群优化算法以发展系统的能力；Srikanth[10]通过粒子群群优化引入任务调度用于生成程序。Pandey[11]等人提出了作为一种可以基于粒子群进行优化的调度启发式教学方法，以降低总执行成本。该研究对比了PSO和最佳资源选择（BRS，best resource selection algorithm）算法，PSO与BRS相比节省了三倍的成本，结果

收稿日期:2020-05-21； **修回日期**:2020-05-29。
基金项目:国家自然科学基金项目(61473216)；陕西省教育厅科学研究计划项目(17JK0459)；西安建筑科技大学基础研究项目(ZR18049)；陕西省自然科学面上项目(2020JM—489)。
作者简介:张娟芝(1993-)，女，陕西西安人，硕士研究生，主要从事数据中心资源调度方向的研究。
段中兴(1969-)，男，湖南茶陵人，博士，教授，硕士生导师，主要从事智能控制、智能检测方向的研究。
通讯作者:熊福力(1974-)，男，黑龙江肇东人，副教授，硕士生导师，主要从事人工智能与系统优化、生产计划与调度优化、智能建筑方向的研究。

表明 PSO 效果更佳。Feng[12] 等人简要介绍了资源分配危机，计划采用粒子群优化算法，通过引入帕累托优势理论来解决这个问题。该理论的依据是每个任务的总任务执行时间，资源预留和 QoS 来研究资源的多目标优化问题。

综上，虽然粒子群算法在云计算领域已经有所应用，但是在解决资源调度问题中存在易陷入局部寻优的缺陷，因此本文对常规粒子群算法中的个体学习因子和社会学习因子进行自适应改进得到一种自适应粒子群优化算法，使得算法全局寻优能力明显提升，且云计算资源任务队列的总完成时间明显下降。

1 问题描述与调度模型

1.1 问题描述

本文所研究的云计算资源调度问题以中小型企业私有云平台的物理资源配置及小规模数量的用户任务为研究对象，进行私有云平台的资源调度优化问题的研究。

图 1 展示出了云计算数据中心资源调度中两阶段任务调度的模型，其中包括各种类型的客户终端（智能手机、PDA、平板电脑、PC 等）、web 访问控制服务器、服务器集群等。每个客户端生成一个或多个任务（例如加载文件）。每个任务 J_j（$j=1, 2, \cdots, m$）由两个操作组成：执行和传输。这两种操作都是在同一台服务器 M_i（$i=1, 2, \cdots, n$）上进行的。设 $T_j^{(1)}$ 是执行所需的时间，$T_j^{(2)}$ 是传输所需的时间。网络将所有客户端通过访问控制服务器连接起来。访问控制服务器将任务分配给可用的物理服务器。根据任务规格，服务器按特定顺序排列。最后服务器通过网络执行任务。

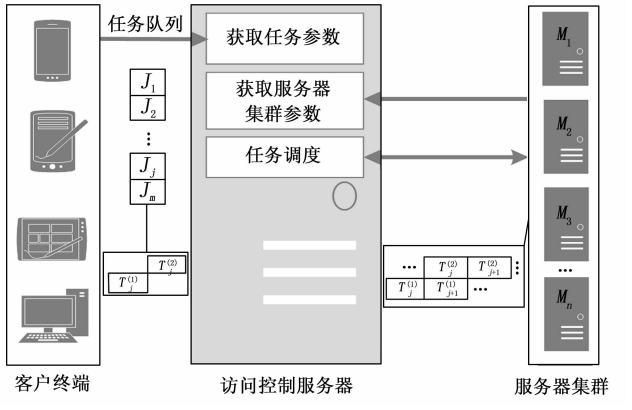

图 1 云数据中心资源调度模型图

云计算资源调度每个阶段的步骤和含义可以简单描述如下：

1）提供虚拟机请求：在云计算环境下，用户根据自身需要，将计算、存储及带宽相关的虚拟机的任务需求通过云服务供应商提供的网络链接将任务提交给云计算平台的资源调度管理系统，由资源调度系统根据客户任务需求进行调度。

2）找到最佳资源点：云计算资源调度系统根据资源调度策略及算法，结合物理服务器资源的实际使用情况和虚拟机任务请求的信息，在云数据中心找到可以匹配的计算资源，并将虚拟机请求分配给该物理资源。

3）执行调度任务：在匹配到最佳资源后，进行对应的虚拟机资源配置，然后由物理服务器执行用户所需的任务队列，待一项任务或一个任务队列完成后，物理机将任务执行结果及自身状态信息反馈给访问控制器。

1.2 资源调度模型

资源调度任务调度的目标是任务队列的完成时间最小化，它涉及分配以及排序两个问题。上述问题描述如下：

$$\begin{cases} M = \{M_1, M_2, \cdots, M_i, \cdots, M_n\} \\ J = \{J_1, J_2, \cdots, J_j, \cdots, J_m\} \end{cases} \tag{1}$$

其中：M 为 n 台相同的并行的物理服务器，J 是一组在两阶段任务调度中处理的 m 个任务。

$$J_j = \{T_j^{(1)}, T_j^{(2)}\}, (T_j^{(1)}, T_j^{(2)} \geqslant 0,\ j = 1, 2, \cdots, m) \tag{2}$$

将整个计划定义为 s，其中包两个部分：分配和排序。为了表示 s 中的分配关系，引入了一个分配矩阵 $\boldsymbol{A}=[A_{ij}]$ 来表示 J 到 M 的映射：

$$A_{ij} = \begin{cases} 1, & J_j \text{ 被分配给 } M_i \\ 0, & J_j \text{ 未被分配给 } M_i \end{cases} \tag{3}$$

分配矩阵可以表示所有可能的映射，并且每个映射都是唯一表示的。由于每个任务只能由一台服务器处理，如下：

$$\sum_{i=1}^{n} A_{ij} = 1 \tag{4}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij} = m \tag{5}$$

实际上，方程（4）和（5）意味着，在每一列中，只有一个元素的值为 1。$C(s)$ 是完成任务执行并传输的时间中最大值，如下：

$$\overline{C}(s) = max\{C_i(s)\}, i = 1, 2, \cdots, n \tag{6}$$

任务调度的目标是最小化 $\overline{C}(s)$。对于 J_j，($j=1, 2, \cdots, m$)，设 $T_{bj}^{(1)}$ 和 $T_{bj}^{(2)}$ 是执行和传输的起始时间，并设 $T_{ej}^{(1)}$ 和 $T_{ej}^{(2)}$ 分别为结束时间，它们受到下列关系的制约：

$$t_{ej}^{(1)} = t_{bj}^{(1)} + T_j^{(1)} \tag{7}$$

$$t_{ej}^{(2)} = t_{bj}^{(2)} + T_j^{(2)} \tag{8}$$

现在，调度优化问题是：

$$min\ \overline{C}(s) \tag{9}$$

$$t_{bj}^{(2)} \geqslant t_{ej}^{(1)} \tag{10}$$

$$a_{ij}t_{ej}^{(1)} \leqslant a_{ik}t_{bk}^{(1)} \text{ or } a_{ik}t_{ek}^{(1)} \leqslant a_{ij}t_{bj}^{(1)} \tag{11}$$

$$a_{ij}t_{ej}^{(2)} \leqslant a_{ik}t_{bk}^{(2)} \text{ or } a_{ik}t_{ek}^{(2)} \leqslant a_{ij}t_{bj}^{(2)} \tag{12}$$

$$i = 1, 2\cdots, n; j, k = 1, 2\cdots m; j \neq k$$

在这个问题中，公式（10）意味着，对于每个任务，只有在执行完成后才开始传输；约束（11）和（12）确保服务器执行和传输。对于分配给同一台服务器的所有任务，例如，一个新任务中的（J_k）的执行（或传输）只在执行（或传输）f 之后才开始，或者以前的工作（J_j）已经完成。分析需要两个假设：所有服务器都是相同的，具有相同的网络知识和相同的接收请求的机会[11]。

2　算法设计

2.1　粒子群算法

PSO 由于其无可比拟性和广泛利用范围内的充足性而变得流行。该算法基于鸟类的行为而工作。部分应用程序利用 PSO 来处理 NP－Hard 问题，如调度和资源分配问题。在 PSO 中，每个优化问题的解都是搜索空间中的一只鸟，称之为“粒子”。所有的粒子都具有一个位置向量和速度向量，并可以根据目标函数来计算当前的所在位置的适应值[12]。粒子的位置和速度更新如下：

$$P(j,1:n) = mod(round(p(j,1:n) + vel(j,:)),n_1) + 1 \tag{13}$$

$$v = e * v(i) + k_1 * A * (p_{best\_p}(i) - p(i)) + k_2 * B * (g_{best\_p} - p(i)) \tag{14}$$

式中，e 为惯性因子，k_1、k_2 分别为每个粒子的个体学习因子和社会学习因子，A、B 为两个取值范围为（0，1）的随机数，$p_{best\_p(j)}$ 为第 j 个粒子的个体历史最优解，g_{best_p} 为全局最优解，$p_{(j)}$ 为第 j 个粒子当前解。

2.2　粒子群算法自适应改进

常规粒子群算法在算法初期收敛速度快，但在后期收敛速度缓慢，易陷入局部寻优的缺陷。本文提出了一种自适应改进的粒子群算法。在粒子群算法中个体学习因子 k_1 和社会学习因子 k_2 通过影响粒子的自我认知与社会认知从而影响粒子的运行轨迹等[13]。本文通过对这两个因子进行自适应改进，从而提高算法的全局探索能力，保存种群多样性，并不断迭代更新，直到满足停止条件。具体自适应改进如下：

在自适应过程中，其取值与目标函数和适应度函数有关，具体表示如下：

$$k1 = \begin{cases} \dfrac{k_{1\max}+k_{1\min}}{2} + \dfrac{(k_{1\max}-k_{1\min})}{2}\tanh\left(\dfrac{F_m - F_{avg}}{F_{\max}-F_{avg}}\pi\right), & F_m \geqslant F_{avg} \\ k_{1\max}, & F_m < F_{avg} \end{cases} \tag{15}$$

$$k2 = \begin{cases} \dfrac{k_{2\max}+k_{2\min}}{2} + \dfrac{(k_{2\max}-k_{2\min})}{2}\tanh\left(\dfrac{F_m - F_{avg}}{F_{\max}-F_{avg}}\pi\right), & F_m \geqslant F_{avg} \\ k_{2\max}, & F_m < F_{avg} \end{cases} \tag{16}$$

式（15）、（16）中，$F_{\max}$ 为最大适应度值，F_{avg} 为平均适应度值，F_c 为适应度中的较大值，F_m 为适应度中的较小值，$k_{1\max}$、$k_{1\min}$、$k_{2\max}$、$k_{2\min}$ 分别为个体学习因子和社会学习因子的上、下限。为了使两个因子的变化在极点处较为平缓的变化，本文使用了双曲正切函数来构造两个学习因子的自适应律。分别对个体学习因子和社会学习因子进行了自适应调整，使得当个体适应度趋于最大值时，该自适应律可提高个体和社会学习因子；当个体适应度趋于最小值时，该自适应律可降低两个学习因子。

自适应粒子群算法具体步骤如下：

1）粒子编码：

本文中粒子位置的编码长度为任务数量，即粒子的维度等于调度的任务数。设有 m 个任务，云资源环境下有 n 个物理资源，则粒子可编码为式（17）所示的 n 维向量：

$$\boldsymbol{P} = \{p_1, p_2, p_3 \cdots p_m\} \tag{17}$$

式中，$1 \leqslant P_i \leqslant n$，$(1 \leqslant i \leqslant m)$，粒子每一维坐标表示一个任务编号，任意一维分量 P_i 表示分配给该任务的资节点编号。

图 2　粒子编码示意图

2）种群初始化：

设粒子种群数量为 Pop，m 个待分配任务数量，n 个物理资源数量。初始化产生 Pop 个粒子，每个粒子的位置由向量 $\boldsymbol{P}$ 表示，第 j 个粒子为：

$$P_j = \{p_{j1}, p_{j2}, p_{j3} \cdots p_{jm}\} \tag{18}$$

式中，$1 \leqslant P_{ji} \leqslant n$，表示任务 j 分配到第 P_i 号物理机上。初始化时将 P_{ji} 的值取为 1～n 之间的随机数。粒子速度由向量 v 表示：

$$v_j = \{v_{j1}, v_{j2}, v_{j3}, \cdots, v_{jm}\} \tag{19}$$

式中，$1 \leqslant j \leqslant Pop$，$-n \leqslant v_{ji} \leqslant n$，$v_{ji}$ 在（$-n$，n）之间随机取值作为初始化值。

3）适应度计算：

根据模型定义的目标函数，并取目标函数值的倒数为适应度函数值。根据式（15）和式（16）设置个体学习因子和社会学习因子。

4）位置和速度更新：

根据式（15）～（16）对粒子进行位置和速度的更新。

5）粒子群更新：

对每个粒子，将其适应度值与其自身经过的最好位置作比较，如果较好，则将其作为当前的最好位置 p_{best}；同样对每个粒子，将其适应度值与其自身经过的最好速度作比较，如果较好，则将其作为当前的最好速度 g_{best}。

自适应粒子群算法流程如图 3 所示。

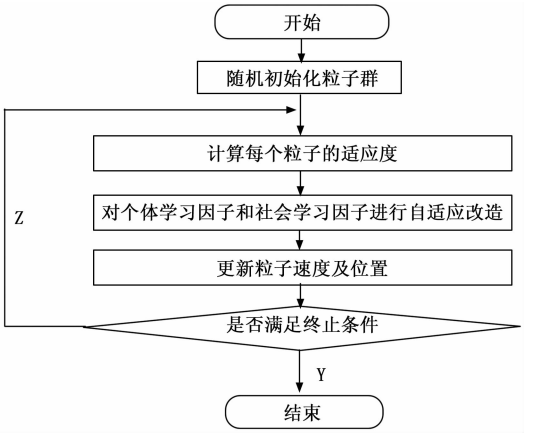

图 3　自适应粒子群算法示意图

3 实验设计及分析

3.1 队列任务执行时间设定

3.2 实验方法及步骤

3.2.1 实验方法

本文基于 Matlab 2016a 平台，完成了粒子群算法及自适应粒子群算法仿真程序的编写。

在实验条件既定的情况下，通过两种算法分别在 n（n=10）个相同配置的物理资源、m（m=15、30、45）个队列任务的情况下的算法仿真运行结果进行分组记录及结果分析。

当 m=30、45 时，待分配的任务以 m=15 时，表 1 给出的各任务进行成倍增加，各任务的执行完成时间保持不变。

表 1 各任务执行时间表

Time/ms	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
$T_j^{(1)}$	92	54	19	78	80	59	114	92
$T_j^{(2)}$	103	48	36	88	72	39	84	79
Time/ms	J_9	J_{10}	J_{11}	J_{12}	J_{13}	J_{14}	J_{15}	
$T_j^{(1)}$	42	38	37	76	29	88	51	
$T_j^{(2)}$	54	45	58	65	34	80	43	

3.2.2 实验步骤

拟通过算法仿真程序的多次运行并记录相关运行数据，以降低算法运行过程中偶然性的性能优劣趋势，以获取最优的算法最优解，分如下几步进行：

1）Matlab 算法仿真程序编写：根据算法设计及流程图，采用 Matlab 编程语言分别进行粒子群及自适应粒子群算法程序的编写；

2）粒子群算法仿真数据采集：分别在 3 组不同物理资源及任务队列数量条件下，粒子群算法仿真程序分别运行 30 次，记录粒子群算法的目标函数值及迭代次数；

3）自适应粒子群算法仿真数据采集：分别在 3 组不同物理资源及任务队列数量条件下，自适应粒子群算法仿真程序分别运行 30 次，记录自适应粒子群算法的目标函数值及迭代次数；

4）实验数据整理计算：计算粒子群算法、自适应粒子群算法 30 次运行数据中，目标值（即任务队列完成时间）的最大值、最小值及标准偏差。

3.3 实验对比分析

对粒子群算法和自适应粒子群算法的实验结果进行了分析，表 2、表 4 给出了当 e=0.6，Pop=50，物理资源数 n=10，待分配任务数 m 分别为 15、30、45 时分别仿真 30 次，两种算法资源调度中任务队列完成时间，单位为 ms；表 3、表 5 给出了两种算法各个规模任务下总目标函数的最大值、最小值、平均值及标准差。

图 5、图 6、图 7 分别为粒子群、自适应粒子群两种算法在任务数分别为 15、30、45 的情况下的仿真结果对比。

从图 4～6 中可以看出，当物理资源数量 n=10，待分配任务数量 m 分别为 15、30、45 时，粒子群算法曲线收敛

表 2 粒子群算法实验数据表

运行次数	任务数		
	15	30	45
1	221	315	412
2	218	312	399
3	223	322	417
4	214	323	403
5	212	324	400
6	224	319	401
7	212	308	397
8	210	312	408
9	216	317	412
10	213	320	409
11	222	321	414
12	217	307	403
14	221	309	401
15	214	314	412
16	213	310	405
17	216	321	415
18	219	312	403
19	222	307	401
20	214	312	410
21	206	317	398
22	216	300	401
23	219	322	407
24	220	324	409
25	223	310	402
26	205	304	403
27	208	322	405
28	212	319	409
29	214	314	405
30	218	308	404

表 3 粒子群算法实验结果分析表

任务数	15	30	45
最大值	224	324	417
最小值	205	300	397
平均值	215.9	314.7	405.7
标准差	5.0	6.4	5.3

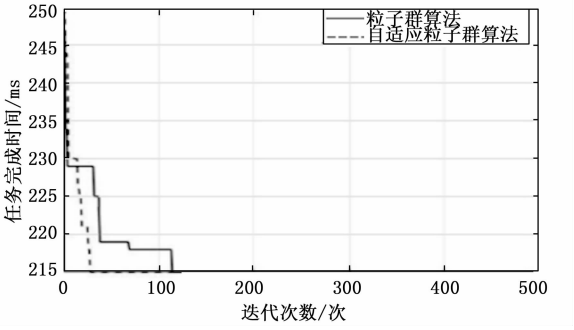

图 4 任务为数 15 时算法对比图

表 4　自适应粒子群算法实验数据表

运行次数	任务数		
	15	30	45
1	229	277	341
2	235	272	333
3	252	264	337
4	213	270	329
5	203	263	324
6	212	272	314
7	245	279	338
8	222	276	329
9	230	273	346
10	235	271	336
11	238	249	332
12	213	280	334
14	219	264	331
15	230	261	328
16	216	272	333
17	225	271	336
18	223	268	324
19	221	277	332
20	213	272	341
21	230	269	324
22	212	262	328
23	199	271	332
24	210	276	323
25	235	274	334
26	216	269	328
27	228	279	331
28	195	272	332
29	204	274	336
30	215	282	334

表 5　自适应粒子群算法实验结果分析表

任务数	15	30	45
最大值	214.9	264.5	325.7
最小值	252	282	346
平均值	195	249	314
标准差	13.3	6.8	6.3

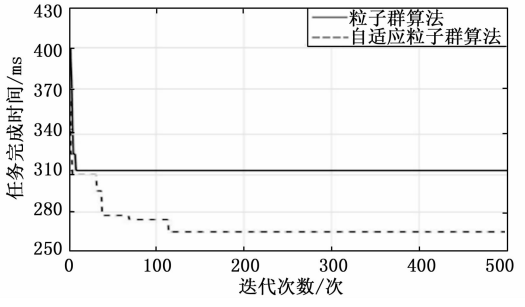

图 5　任务为数 30 时算法对比图

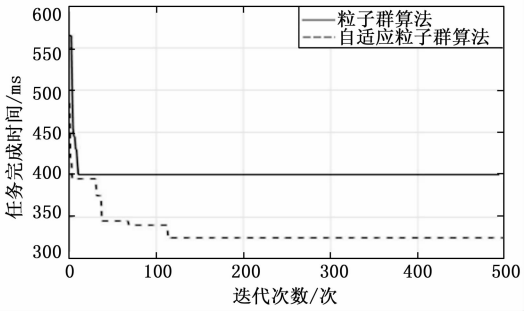

图 6　任务为数 45 时算法对比图

粒子群迭代次数约 20 次左右即趋于稳定，速度明显高于自适应粒子群算法，表现出了易陷入局部寻优的缺点。自适应粒子群算法迭代次 100 次后，逐步趋于稳定，在一定程度上提升了算法的搜索能力，实现全局寻优。同时对于目标函数任务队列总完成时间而言，自适应粒子群算法资源调度中任务队列的总的完成实验平均时间为分别为 214.9 ms、264.5 ms、325.7 ms，较粒子群算法的总的完成时间 215.9 ms、314.7 ms、405.7 ms，具有大幅度的提升。

4　结束语

本文基于云资源两级调度模型，以两阶段资源调度为研究对象，设定了以任务队列完成时间为目标的任务调度策略及数学模型。提出了一种自适应粒子群算法用于解决粒子群算法在处理云资源调度过程中算法过早收敛从而难以全局寻优的缺陷。

实验表明：本文提出的自适应粒子群算法不仅具备良好的收敛性和全局寻优能力，同时能够大幅度降低云资源调度中任务队列的总完成时间，具备良好的性能。用于企业内部私有云或混合云平台的云资源调度应用中，可以提高任务执行效率并降低云计算平台的电力能耗。

参考文献：

[1] 祝　旭. 绿色云计算数据中心能耗资源调度优化关键技术研究 [J]. 无线互联科技，2019，16 (11)：118－119.

[2] 罗慧兰. 基于 Wi－Fi 与 Web 的云计算资源调度算法研究 [J]. 计算机测量与控制，2017，25 (12)：150－152，176.

[3] 张　露，尚艳玲. 云计算环境下资源调度系统设计与实现 [J]. 计算机测量与控制，2017，25 (1)：131－134.

[4] 郝　亮. 面向能耗优化的云计算资源调度算法研究 [D]. 哈尔滨：哈尔滨工业大学，2015.

[5] 黄伟建，郭　芳. 基于烟花算法的云计算多目标任务调度 [J]. 计算机应用研究，2017，34 (6)：1718－1720，1731.

[6] 钟　猛. 云计算资源调度研究及改进 [D]. 赣州：江西理工大学，2015.

[7] Keshanchi B，Souri A，Navimipour N J. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues：formal verification，simulation，and statistical testing [J]. Journal of Systems and Software，2017 (124)：1－21.

(下转第 226 页)