文章编号:1007-6735(2017)05-0505-06 DOI:10.13255/j.cnki.jusst.2017.05.015

云计算环境下基于优先级的IO和网络密集型应用调度策略

麻双克, 周兰凤
(上海应用技术大学 计算机科学与信息工程学院,上海 201418)

摘要:当前云计算环境中,当大于CPU核数的IO和网络密集型应用并发执行时,传统的资源分配策略没有考虑到应用的特性,导致资源利用率偏低,应用执行效率低下.针对这种现状,本文对IO密集型应用和网络密集型应用进行分析,根据它们可量化的特性,提出并设计了基于优先级的IO和网络密集型应用调度策略.针对可量化的小应用提高优先级,获得更大的CPU时间片,让小应用尽早完成,然后将所有CPU时间片分配给大应用,减少进程之间的切换调度,提高了效率.大量实验表明,该策略可以有效提高应用的执行效率,减少资源的消耗.

关键词:云计算;IO密集型;网络密集型;优先级;CPU时间片;应用调度
中图分类号:TP 391.9 **文献标志码**:A

Scheduling Strategy of IO and Network Intensive Applications Based on the Priority in Cloud Environment

MA Shuangke, ZHOU Lanfeng
(*School of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 201418, China*)

Abstract: At present, in cloud environment, when the times of parallel IO and network intensive applications are more than the number of CPU cores, the traditional resource allocation strategy is not taking the characteristics of the applications into account, which leads to the low utilization rate of resources and the low efficiency of applications execution. To solve this situation, the IO intensive applications and network intensive applications were analyzed, and the scheduling strategy for IO and network intensive applications was put forward based on the priority. The priority of small applications was improved to get larger CPU time slices, so that small applications can be completed as soon as possible, and then all the CPU time slices are allocated to large applications, to reduce the process of switching between schedulings so as to improve the efficiency. A large number of experiments show that the strategy can effectively improve the efficiency of the applications execution and reduce the consumption of resources.

收稿日期:2017-04-26
基金项目:国家自然科学基金资助项目(41671402)
第一作者:麻双克(1990-),男,硕士研究生.研究方向:云计算、大数据.E-mail:1204016495@qq.com
通信作者:周兰凤(1966-),女,副教授.研究方向:路径规划、云计算、大数据.E-mail:ifzhou@sit.edu.cn

Keywords: *cloud computing*; *IO intensive*; *network intensive*; *priority*; *CPU time slice*; *application schedule*

随着计算机软件和硬件技术的飞速发展,计算模型层出不穷,继分布式计算、并行计算和网格计算之后,商业界和学术界提出了一种新的计算模型——云计算模型.云计算是一种按需付费的模式,将计算机或服务器等的软硬件资源虚拟化成虚拟机资源,满足不同用户对资源的实时动态需求,集成部署,统一管理,具有廉价、高效、便利的特点[1]. 正是由于它的这种可靠性和便利性,近年来云计算发展迅猛,由此带来的能耗问题也日益突出.根据美国环保署报告[2],2006年美国境内当时有6 000多个云计算数据中心,当年共消耗了总价值高达45亿美元的610亿kW·h的电量.国内同样存在此类能耗问题,2011年,中国联通和电信两大运营商的数据中心年耗电总量约达211亿kW·h[3].各种数据表明,云计算数据中心消耗了大量能源,还有文献表明,一些数据中心的服务器利用效率低下,有效利用率只有11%～50%[4],存在巨大的资源浪费.

提高数据中心的资源利用效率是节约能源的有效途径,优化计算策略是一种行之有效的方法.前人通过对云计算下应用的特征进行分析,根据应用所使用的资源类型和资源比重提出了一种云应用识别模型,其中的IO密集型应用和网络密集型应用是现在云计算中十分普遍的应用,当大量IO和网络密集型应用并发执行时,传统的分配策略没有考虑不同应用的特征,均匀地分配时间片,增加了进程间的切换调度,造成资源的浪费.在此情况下,本文对云计算环境下的IO和网络密集型应用进行分析,分析了其大小可量化的特点,提出并设计了一种基于优先级的IO和网络密集型应用调度模型.

1 相关研究

很多学者对云计算中的基于优先级的资源调度策略进行了研究,Pandey等[5]在云环境下使用优化粒子群算法得到了一种资源分配方式,使得应用的执行时间和进程通信时间最低.Kaur等[6]提出了一种新的基于优先级的抢占调度算法,在绿色云计算中,根据服务器的能量需求和服务器频率可用性,计算服务器被分配到最佳匹配的进程.祝家钰等[7]结合表启发式调度技术和任务复制的思想,提出了基于路径优先权的任务调度算法,实验表明,该算法能获得较短的调度长度.为了满足云计算中服务提供商和服务消费者双方协商的服务等级协议(SLA),林清滢等[8]提出了在云计算环境下采用元调度和本地调度两层框架结构,在每层上采用了多级反馈队列调度算法.谢丽霞等[9]针对云计算数据密集型和计算密集的特点,提出了分层调度策略以实现云计算的服务和资源调度,实验结果表明,所采用的调度有效地提高了资源利用率.郭松辉等[10]提出了一种动态优先级排序的虚拟机I/O调度算法DPS(dynamic priority scheduling),该算法以离差最大化方法计算I/O任务的优先级评估属性权重,对I/O任务优先级进行综合评估,通过引入任务所在虚拟域价值,体现云计算环境下虚拟域重要性差异.

有学者对基于应用资源类型分类的优化调度策略进行了研究.Peng等[11-12]对CPU密集型应用和IO密集型应用进行了研究和判定,发现了CPU密集型和IO密集型应用的特征,在此基础上建立了判别CPU和IO密集型应用的数学模型,并由此提出了CPU密集型应用和IO密集型应用的调度模型.周兰凤等[13]对IO密集型应用中的海量小文件传输提出了一种基于优化服务器缓存区参数机制的打包传输策略,结果表明,该策略能够明显地提高文件传输效率,在一定程度上降低能耗.周兰凤等[14]对IO密集型应用中的大量流媒体文件提出了文件切割打包传输策略,寻求最优打包阈值,在一定程度上解决了打包时间和传输时间的矛盾,缩短文件传输时间.

2 基于优先级的IO和网络密集型应用调度模型

在使用了云应用分类模型对应用进行分类之后,如何有效地对应用进行调度是个亟待解决的问题.本文探讨IO和网络密集型应用调度问题,以达到提高云应用执行效率、减少能源消耗的目的.

2.1 IO和网络密集型应用调度

在云计算环境中,根据云应用所使用的资源类型和资源比重将应用进行分类,可分为CPU密集型应用、内存密集型应用、IO密集型应用、网络密集型应用等.其中,IO密集型应用是消耗IO资源最多的

应用，主要是磁盘读写，如局域网 FTP 传输、dd 写文件等；网络密集型应用是消耗网络资源最多的应用，如在线视频服务、socket 网络传输等. 提高应用的执行效率等同于减少应用的执行时间.

对于 IO 密集型应用，当大于 CPU 核数的多个 IO 密集型应用同时运行时，这些 IO 应用共享计算机 CPU，IO 等资源，CPU 会给这些应用分配时间片，让应用分别执行. 为了使这些 IO 密集型应用更快地完成，对不同的 IO 应用分配不同的优先级是一种优化的调度策略. 对于较小的 IO 应用，提高优先级，分配较大的时间片，让该应用尽快完成. 当较小的应用完成后，将全部的时间片都分配给较大的 IO 应用，让较大的 IO 应用也尽快完成，减少进程的切换调度，从而减少应用执行时间.

对于网络密集型应用，使用和 IO 密集型应用相似的策略. 当大于 CPU 核数的网络密集型应用都共享系统的 CPU、带宽等资源时，为了提高他们的执行效率，对不同应用分配不同的优先级. 对于较小的网络密集型应用，提高优先级，分配较大的时间片，让该应用尽快完成. 当较小的应用完成后，将所有时间片都分配给较大的网络密集型应用，让较大的网络密集型应用也尽快完成，减少进程的切换调度，从而减少应用执行时间.

2.2　Linux 进程优先级和进程调度

Linux 对普通的进程是根据动态优先级进行调度，这也是 Linux 主要的进程调度手段. 而动态优先级是由静态优先级(static_prio)调整而来. Linux 环境下，静态优先级是用户不可见的，隐藏在内核中. 而内核提供给用户一个可以影响静态优先级的接口，那就是 Nice 值，两者关系如下：

$$\text{static\_prio} = \text{MAX\_RT\_PRIO} + \text{Nice} + 20 \quad (1)$$

默认 MAX_RT_PRIO 配置为 100，Nice 值的范围是 $-20 \sim +19$，因而静态优先级范围在 100～139 之间. Nice 数值越大，就使得 static_prio 越大，进程优先级就越低. Linux 普通进程优先级一共有 40 个级别，数字越大，表示进程的优先级越低，默认时进程的优先级是 0. 这个时间片执行完后，一般情况下就会根据它的初始优先级来重新分配时间片，优先级为 +19 时最低，只分配最小时间片 5 ms，优先级为 0 时分配 100 ms 时间片，优先级是 -20 时分配最大时间片 800 ms.

当大于 CPU 核数的多个云应用在虚拟机上同时运行时，一些设计良好的高性能系统，如 nginx 的分配策略没有考虑到应用的特性，均匀地将时间片分配给这几个工作进程，当每个工作进程运行结束一个相同的时间片后，内核需要作进程切换，把即将结束的进程上下文保存下来，打开下一个进程的上下文，造成进程切换频繁，导致应用执行效率低下，资源利用率偏低.

2.3　感知应用“大小”的调度策略

IO 密集型应用和网络密集型应用的大小在一定程度上可以量化. IO 密集型应用通过磁盘来读取和写入数据，网络密集型应用通过网卡来接收和发送数据. 不管是 IO 磁盘资源还是网络带宽资源，IO 密集型应用和网络密集型应用都是感知数据大小的. 因此，对于 IO 密集型应用和网络密集型应用，可以使用相同的调度策略——感知应用“大小”的调度策略. 这里说的“大小”是指这个应用所操作的文件大小. 对于 IO 密集型应用来说，是指该应用需要写入或者读取磁盘的文件大小；对于网络密集型应用来说，是指应用需要从网络接收或者发送的文件大小. 对于 IO 或者网络密集型应用，根据应用的“大小”，设定优先级，对应用分配不同的时间片和资源，从而缩短应用执行时间，提高应用的执行效率.

对于 IO 密集型应用或者网络密集型应用，假设共有 N 个应用，优先级计算方法如下：

对所有的应用按照“大小”，由小到大排序，应用顺序为 $A_1, A_2, A_3, \cdots, A_{N-1}, A_N$，对应的应用大小为 $Size_{A_1}, Size_{A_2}, Size_{A_3}, \cdots, Size_{A_{N-1}}, Size_{A_N}$. Linux 进程优先级共有 40 级，可以分成若干个优先级来设计. 初始化优先级

$$Nice_{A_i} = \begin{cases} \alpha_1,\ Size_{A_i} \leqslant \alpha \sum_1^N Size_{A_i}/N \\ \alpha_2,\ \alpha \sum_1^N Size_{A_i}/N < Size_{A_i} \leqslant \beta \sum_1^N Size_{A_i}/N \\ \alpha_3,\ \beta \sum_1^N Size_{A_i}/N < Size_{A_i} \leqslant \gamma \sum_1^N Size_{A_i}/N \\ \vdots \\ \alpha_n,\ Size_{A_i} > \delta \sum_1^N Size_{A_i}/N \end{cases} \quad (2)$$

式中：$-20 \leqslant \alpha_1 \leqslant \alpha_2 \leqslant \alpha_3 \leqslant \cdots \leqslant \alpha_n \leqslant 19$，$i = 1, 2, 3, \cdots, N$；$\alpha, \beta, \gamma, \delta$ 为比例系数，与样本分布有关，使样本尽量区分开.

在知道优先级计算方法后，对 IO 密集型应用或者网络密集型应用的调度方法如下：

步骤 1　初始化或更新应用的优先级表，对于当前服务器上所有云应用，根据上述优先级计算方

法,计算各个云应用的优先级;

步骤2 Linux系统根据各个云应用优先级来分配CPU时间片执行;

步骤3 当有云应用执行完成后,或者有新来的云应用,转到步骤1重新更新优先级.

3 实验与结果

为了验证上面提出的基于优先级的IO和网络密集型应用的调度策略,做了大量的实验.

3.1 实验环境

实验所使用的平台是开源的云计算平台CloudStack,版本是4.5.1,实验平台环境由3台物理机在局域网里组建而成,1台设为管理节点,1台设为计算节点,1台设为存储节点.主机节点Host的操作系统为Redhat server 5.3,处理器Inter I5 4440@3.10 GHz,内存8 GB DDR3,硬盘容量1 TB.Host上的虚拟机,操作系统为Ubuntu14.04版本,处理器为1 GHz*1,内存为1 GB,网络带宽为150 Mb/s,磁盘转速为7 200 r/min.具体参数如表1所示.

表1 物理机参数表

Tab.1 Physical machine parameters

机器型号	Hostname	内存	硬盘	CPU
WiNSER	Management	8 GB	1 TB	I5-4440@3.10GHz
OPTPLEX760	Storage Node	4 GB	250 GB	E7400@2.80GHz
WiNSER	Host	8 GB	1 TB	I5-4440@3.10GHz

3.2 实验内容

为了验证提出的基于应用大小的优先级调度策略,选取了应用模型中比较典型的IO密集型应用和网络密集型应用来进行验证.使用1核虚拟机来运行IO密集型应用和网络密集型应用.其中,IO密集型应用选取的是Linux下面的dd方法让其在本地生成不同大小的文件;网络密集型应用选取的是Socket传输不同大小的文件(去掉磁盘IO对实验结果的影响,客户端不从磁盘上读取数据).

实验1 在传统策略和本文提出的策略下(传统策略即默认优先级为0,均匀分配时间片),在1核虚拟机上测试使用dd命令生成100 MB文件所用时间,并发生成100 MB和500 MB文件所用时间,并发生成100,500,1 000 MB文件所用时间,记为实验1.1,1.2,1.3,并分别计算传统策略和本文策略所用总时间.

实验2 在传统策略和本文提出的策略下,在1核虚拟机上测试使用Socket传输100 MB文件所用时间,并发传输100 MB和500 MB文件所用时间,并发传输100,500,1 000 MB文件所用时间,记为实验2.1,2.2,2.3,并分别计算传统策略和本文策略所用总时间.

实验设置5个优先级为例,因本实验时间片越大,进程切换越少,应用将更快完成,故取优先级a_1,a_2,a_3,a_4,a_5最小,得到$a_1=-20,a_2=-12,a_3=-4,a_4=3,a_5=11$,比例系数取$\alpha=0.02$,$\beta=0.2,\gamma=1,\delta=2$,$\sum_1^N Size_{A_i}/N$约为500 M.故实验设计采取的优先级表为

$$Nice_{A_i}=\begin{cases}-20, & Size_{A_i}\leqslant 10\ \text{MB}\\ -12, & 10\ \text{MB}<Size_{A_i}\leqslant 100\ \text{MB}\\ -4, & 100\ \text{MB}<Size_{A_i}\leqslant 500\ \text{MB}\\ 3, & 500\ M<Size_{A_i}\leqslant 1\ 000\ \text{MB}\\ 11, & Size_{A_i}>1\ 000\ \text{MB}\end{cases}\quad(3)$$

3.3 实验分析与结果

本文做了大量的实验得到上面的2组实验结果,然后进行分析以验证本文提出的模型.

3.3.1 IO密集型应用的实验结果分析

图1表示使用不同策略创建100 MB文件所需要的时间,可以看出,创建单个文件时,没有进程切换,使用本文策略和传统策略时间差不多,优势不明显.

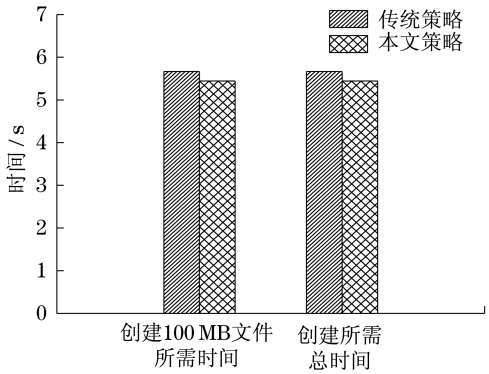

图1 创建100 MB文件不同策略消耗时间图

Fig.1 Consuming time diagram of creating 100 MB files with different strategies

图2表示使用传统策略和本文策略并发创建100 MB文件和500 MB文件所需要的时间,可以看出,进程切换较多,使用本文策略创建100 MB文件优先级高,其执行时间都有所减少,之后时间片都用来创建500 MB文件,执行时间也有所减少,总时间比传统时间减少6.5 s,较有优势.

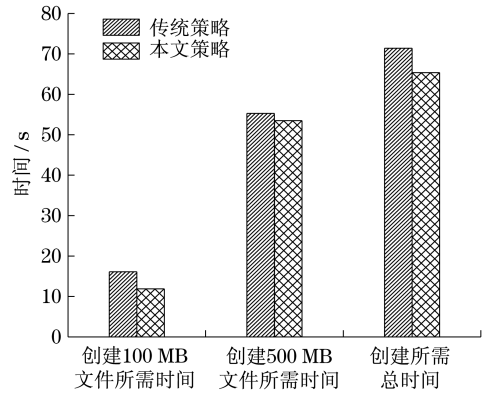

图 2　并发创建 100 MB 和 500 MB 文件不同策略消耗时间图

Fig.2　Consuming time diagram of creating 100 MB and 500 MB files at the same time with different strategies

图 3 表示使用传统策略和本文策略并发创建 100，500，1 000 MB 文件所需要的时间. 可以看出，进程切换很多，小应用执行时间有所减少，对于创建较大文件时间也有明显减少，故使用本文策略减少了进程间切换调度，总时间缩短了近 59.8 s 时间，大大提高了应用执行的效率，具有显著优势.

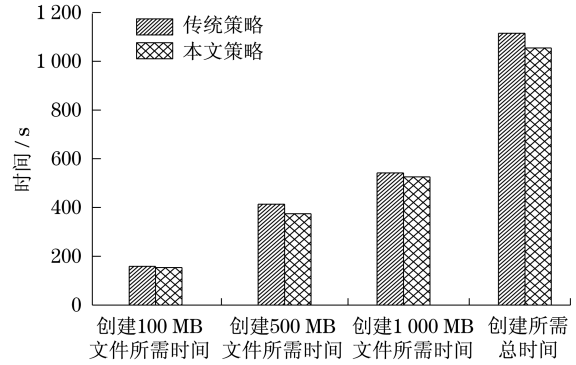

图 3　并发创建 100，500，1 000 MB 文件不同策略消耗时间图

Fig.3　Consuming time diagram of creating 100，500，1 000 MB files at the same time with different strategies

从表 2 可以看出，当 IO 密集型应用个数大于 CPU 核数时，使用本文策略可以有效地缩短小应用的执行时间，而对于大应用，其执行时间也有所减少，从总的执行时间上看，使用本文策略可以有效地减少进程间的切换调度，缩短 IO 密集型应用执行时间，提高 IO 密集型应用的执行效率，而且应用越多、应用越大，本文策略优势越明显.

表 2　实验 1.1，1.2，1.3 使用本文策略缩短总时间和占比

Tab.2　Total reducing time and proportion of Experiment 1.1，Experiment 1.2，and Experiment 1.3 with the strategy proposed

实验标号	使用本文策略缩短总时间/s	缩短总时间占默认策略时间百分率/%
实验 1.1	0.2	3.4
实验 1.2	6.5	8.5
实验 1.3	59.8	5.8

3.3.2　网络密集型应用的实验结果分析

图 4 是使用不同策略传输 100 MB 文件所需要的时间，可以看出，传输单个文件时，没有进程切换，使用本文策略和传统策略时间差不多，优势不明显.

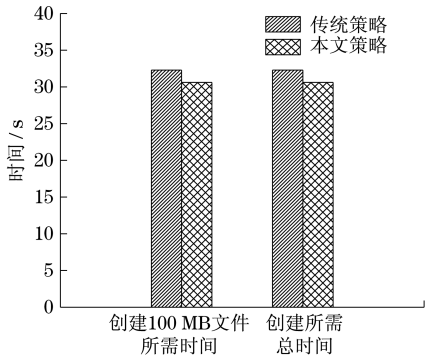

图 4　传输 100 MB 文件不同策略消耗时间图

Fig.4　Consuming time diagram of transferring 100 MB files with different strategies

图 5 表示使用传统策略和本文策略并发传输 100 MB 文件和 500 MB 文件所需要的时间，可以看出，进程切换较多，使用本文策略传输 100 MB 文件和 500 MB 文件时间都有所减少，总时间比默认时间减少 21.7 s，较有优势.

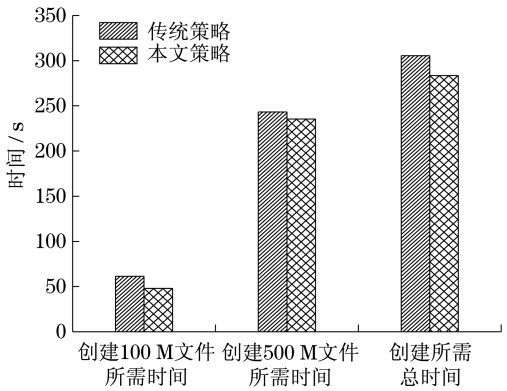

图 5　并发传输 100 MB 和 500 MB 文件不同策略消耗时间图

Fig.5　Consuming time diagram of transferring 100 MB and 500 MB files at the same time with different strategies

图 6（见下页）表示使用传统策略和本文策略并发传输 100，500，1 000 MB 文件所需要的时间，可以看出，进程切换很多，小应用执行时间有所减少，对于传输较大文件时间也有明显减少，故使用本文策略减少了进程间切换调度，总时间缩短了近 81 s 时间，大大提高了应用执行的效率，具有显著优势.

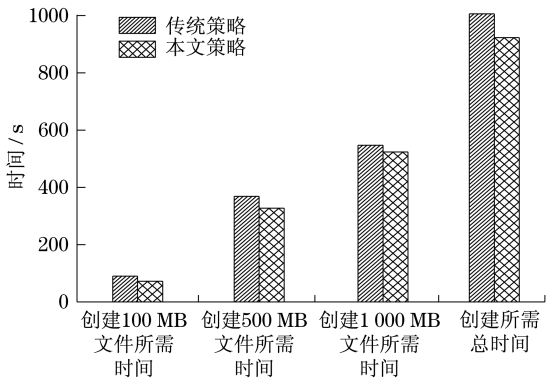

图6 并发传输100,500,1 000 MB文件不同策略消耗时间图

Fig.6 Consuming time diagram of transferring 100,500,1 000 MB files at the same time with different strategies

从表3可以看出,当网络密集型应用个数大于CPU核数时,使用本文策略可以有效缩短小应用的执行时间,一定程度地减少大应用的执行时间,所以,从总的执行时间上看,使用本文策略可以有效地减少进程间的切换调度,缩短网络密集型应用执行时间,提高网络密集型应用的执行效率,而且应用越多、应用越大,本文策略优势越明显.

表3 实验2.1,2.2,2.3使用本文策略缩短总时间和占比

Tab.3 Total reducing time and proportion of Experiment 2.1, Experiment 2.2 and Experiment 2.3 with the strategy proposed

实验标号	使用本文策略缩短总时间/s	缩短总时间占默认策略时间百分率/%
实验2.1	2.2	5.5
实验2.2	21.7	7.2
实验2.3	81.0	8.2

4 结 论

在前人对云计算应用资源消耗特征和比重进行分类的基础上,分析了IO密集型和网络密集型应用大小可量化的特点,对IO和网络密集型应用进行量化,并提出了一种基于应用大小的优先级调度策略.大量实验表明,该策略可以有效地提高云计算下IO和网络密集型应用的执行效率,对于应用繁多、庞大的云计算服务器效果更加明显,不仅提高了云计算应用的执行效率,还提高了云计算中心资源的利用率,减少能源的消耗,为下一步更好、更完善的资源调度策略提供了一定的基础和借鉴.

参考文献:

[1] 刘鹏程,陈榕.面向云计算的虚拟机动态迁移框架[J].计算机工程,2010,36(5):37-39.

[2] KURP P. Green computing[J]. Communications of the ACM,2008,51(10):11-13.

[3] 佚名.云计算"偷吃"巨大能源[J].呼和浩特经济,2012(2):59-60.

[4] BOHRER P,ELNOZAHY E N,KELLER T,et al. The case for power management in web servers[M]// GRAYBILL R,MELHEM R. Power Aware Computing. Boston,MA:Springer,2002:261-289.

[5] PANDEY S,WU L L,GURU S M,et al. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments[C]// Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications. Perth,Australia:IEEE,2010:400-407.

[6] KAUR G,MIDHA S. A preemptive priority based job scheduling algorithm in green cloud computing[C]// Proceedings of the 6th International Conference on Cloud System and Big Data Engineering. Noida:IEEE,2016:152-156.

[7] 祝家钰,肖丹.云计算环境下基于路径优先级的任务调度算法[J].计算机工程与设计,2013,34(10):3511-3515.

[8] 林清滢,陆锡聪,徐林.云计算中面向SLA的作业分层优先级调度策[J].计算机科学,2014,41(6A):316-317.

[9] 谢丽霞,严焱心.云计算环境下的服务调度和资源调度研究[J].计算机应用研究,2015,32(2):528-531.

[10] 郭松辉,龚雪容,王炜,等.一种动态优先级排序的虚拟机I/O调度算法[J].计算机科学,2017,44(1):13-19.

[11] PENG J J, DAI Y C, RAO Y, et al. Research on processing strategy for CPU-intensive application[J]. Journal of Systems Architecture,2016,70:39-47.

[12] PENG J J,RAO Y,DAI Y C,et al. Modeling for I/O intensive applications in cloud computing[C]// 2015 IEEE Symposium on Service-Oriented System Engineering. San Francisco Bay, CA: IEEE, 2015: 229-234.

[13] 周兰凤,赵鹏飞,彭俊杰.基于云环境下一种小文件传输策略研究[J].计算机工程与科学,2016,38(1):20-27.

[14] 周兰凤,孟驰,彭俊杰.一种基于云环境的文件存储策略的研究[J].计算机工程与科学,2016,38(2):262-268.

(编辑:石 瑛)