基于博弈论的云资源调度算法

徐 飞[1,2] 王少昌[1] 杨卫霞[1]
(西安工业大学计算机科学与工程学院 西安 710021)[1] (西北工业大学航海学院 西安 710072)[2]

摘 要 在云环境下的大数据中心中,虚拟机数目和虚拟机的负载会随着用户和应用的需求而时常发生变化。虚拟机需要进行动态资源调整,及时移除系统中的热点资源,从而达到整个系统的负载均衡。通过对云资源分配的理论研究,获取到 First-Fit 贪心算法和 Round Robin 轮询算法等。将它们应用到一些云系统中虽然能够在短时间内解决问题,但存在资源利用率和负载均衡等方面的问题。文中提出一种基于博弈论的 FUTG(Fairness-Utilization Tradeoff Gme)云资源调度算法。该算法打破了固定数量的资源分配瓶颈,将 QoS 因素纳入考量范围,解决了资源利用率以及资源分配的公平性这两个优化目标的资源调度问题。仿真实验结果表明,FUTG 算法能够显著提高动态资源调度的有效性和动态负载下资源使用的执行效率。

关键词 云资源调度,服务质量,博弈论,FUTG,动态负载
中图法分类号 TP18 **文献标识码** A

Cloud Resource Scheduling Algorithm Based on Game Theory

XU Fei[1,2] WANG Shao-chang[1] YANG Wei-xia[1]
(School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China)[1]
(School of Marine Engineering, Northwestern Polytechnical University, Xi'an 710072, China)[2]

Abstract In a large data center in a cloud environment, the number of virtual machines and the load of virtual machines change frequently with the needs of users and applications. The virtual machines need to make dynamic resource adjustments to remove hotspot resources in the system in time and implement load banlancing for the entire system. Now through theoretical research on cloud resource allocation, we have obtained such applications as First-Fit greedy algorithm and Round Robin polling algorithm that can be applied to some cloud systems to solve problems in a short time, but they have the problems of resource utilization and load. Therefore, this paper proposed a fuzzy-future-memory tradeoff (GMO) cloud resource scheduling algorithm based on game theory. The algorithm breaks a fixed number of resource allocation bottlenecks, takes QoS into consideration, and solves problems of resource utilization and resource allocation fairness. Simulation results show that FUTG algorithm can significantly improve the effectiveness of dynamic resource scheduling and the efficiency of resource usage under dynamic load.

Keywords Cloud resources dispatch, QoS, Game theory, FUTG, Dynamic load

云计算是一种新型的服务模式,是计算机技术和网络技术的融合,是未来信息社会的关键技术和手段。它把存储在数据集群上的大数据计算资源进行集中统一管理和分配,而云计算的核心问题就是资源管理。在云计算中,科学的调度资源会对任务的时间和服务质量造成显著的影响,调度策略直接决定了资源分配方式和资源的使用效率。云计算系统存在于一个动态环境中,用户的群体非常庞大,并且每个服务对云资源的请求都处于不断变化的状态。另一方面,云计算中的虚拟化技术弱化和屏蔽了物理硬件的复杂性,具有高内聚、低耦合的特点。云计算的多资源共享和按需资源供应导致云资源调度问题成为典型的非确定多项式(Non-deterministic Polynominal, NP)问题。

博弈论是数学理论研究的一个方向,目前已成为经济学中一种重要的研究方法。其主要通过研究多个博弈参与者之间理性的相互作用来解决决策问题,寻找最优解。博弈论只要有五大基本要素:博弈参与者、策略集、收益、信息以及纳什均衡。博弈论根据不同的分类方法又可分为静态和动态博弈、合作和非合作博弈、完全信息和非完全信息博弈。

目前已有的相关工作中,主要用博弈论方法解决分布式并行环境下的资源调度问题。Xin 等[1]运用非合作博弈模型解决云资源管理问题,求解纳什均衡解来获取云中多租户资源获取的最佳方案;但它主要研究的是云中资源定价问题。Osorio 等[2]简要描述了当前云资源分配策略,并给出了一个云资源分配框架。Carroll 等[3]提出基于博弈论的资源组合框架,在供应云服务阶层动态创建虚拟组织,通过最大化收益提高云服务阶层的收益;但每个云供应商无法准确计算出各组织的利益,因此信息有限,无法做出正确决策。Mohammad 等[4]提出用博弈方法来解决多提供商云环境的资源管理问

本文受国家自然科学基金(51179156),陕西省教育厅科学研究项目计划(15JK1364)资助。
徐 飞(1980-),男,博士,副教授,主要研究方向为中间件、分布式计算,E-mail:29112462@qq.com;王少昌(1991-),男,硕士生,主要研究方向为云机器人;杨卫霞(1993-),女,硕士生,主要研究方向为云机器人。
万方数据

题，他们得出了合作博弈比非合作博弈的分布式资源分配博弈模型更适合于混合云或云联盟环境的结论。张晞等[5]主要对云计算虚拟拓扑结构的任务调度算法进行研究。丁丁等[6]基于用户行为对云资源调度问题进行研究，提出基于用户行为反馈的资源调度机制。史宝鹏等[7]根据医疗系统对资源的不同需求提出 IB-Choose 资源调度策略。目前广泛使用的启发式资源调度算法有基于遗传算法、模拟退火算法、粒子群算法的资源调度算法以及这些算法的改进算法[8-11]。启发式算法的优点是能自适应并很好地为 NP 难资源调度问题寻找最优或者近似最优解；但算法本身和建模过程比较复杂，而且对于不同问题，收敛性也不同。

除了上面介绍的资源调度算法外，还有一些综合和改进的调度算法，如在原有调度算法的基础上加入优先级、信任机制等约束算法，也有在原有基础之上进行改进，以提高算法的负载均衡性能并降低算法的能耗开销等[12]。这些算法是关注固定数量的资源分配问题，没有考虑到客户的服务质量问题。因此，本文的研究主要从用户的服务质量入手，将 QoS 加入云资源调度算法优化的约束条件，对 FUTG 算法进行优化和改进，达以到提高用户满意度的目标。

本文第 1 节介绍博弈论模型；第 2 节重点介绍 FUTG 算法的设计和实现；第 3 节通过仿真实验对比 EFFD 算法、谷歌集群和 FUTG 算法的资源利用率以及在不同阈值下的用户满意度，充分显示出 FUTG 算法的可行性和优越性；最后为结果分析总结以及拓展。

1 博弈论模型

博弈模型建立的基本假设：每个博弈参与者都是理性的，并且追求自身利益的最大化，每个参与者需要考虑自身的知识信息和其他参与者的行为预期。从博弈类型来看，云资源调度博弈模型一般可以分为 3 类：1)非合作博弈，每个参与者都独立与其他参与者选择自己的行动；2)合作博弈，博弈中的参与者以结盟和合作形式行动；3)半合作博弈，参与者选择一位参与者合作。

假设博弈的参与者是每天可用的物理机 m_i，云数据中心有 M 台不同规格的物理机$\{m_1,m_2,\cdots,m_M\}$，每台物理机 m_i 对应的可用资源数量为$\vec{R}(m_i)$，物理机的计算能力为 $MIP(m_i)$，网络带宽为 $Band(m_i)$，对云资源调度中心而言，某一个决策时刻共有 L 个任务，每个子任务 l 需要 V_l 类型的虚拟单元，资源向量为$\vec{r_i}=(r_{i1},r_{i2},\cdots,r_{iK})$，工作负载为 $Workload_i$，任务最大响应时间为 RT_i，成本预算为 $Cost_i$，资源价格向量 $Price=(Price_1,Price_2,\cdots,Price_K)$，用户的可选策略是从云数据中心的可用物理机中选取合适的主机来创建自己需要的虚拟单元类型。如图 1 所示，可以表示为一个资源供应量 $S(m_i)=(s_1(m_i),s_2(m_i),\cdots,s_i(m_i))$，$s_i\in N$ 表示创建子任务 i 所需要虚拟单元类型的数量，物理机的资源分配策略可表示为资源分配矩阵 $A(m_i)$。

$$A(m_i)=\begin{pmatrix}\bar{a}_1(m_i)\\ \bar{a}_2(m_i)\\ \cdots\\ \bar{a}_n(m_i)\end{pmatrix}=\begin{pmatrix}\bar{a}_{11}(m_i)&\bar{a}_{12}(m_i)&\cdots&\bar{a}_{1K}(m_i)\\ \bar{a}_{21}(m_i)&\bar{a}_{22}(m_i)&\cdots&\bar{a}_{2K}(m_i)\\ \cdots&\cdots&\cdots&\cdots\\ \bar{a}_{n1}(m_i)&\bar{a}_{n2}(m_i)&\cdots&\bar{a}_{nK}(m_i)\end{pmatrix}\tag{1}$$

万方数据

其中，$A_i(m_i)=(a_{i1}(m_i),\cdots,a_{ik}(m_i),\cdots,a_{iK}(m_i))$表示物理机 m_i 中各类型资源分配给子任务 i 的数量，其中 $a_{ik}(m_i)=s_i(m_i)\cdot r_{ik}$。

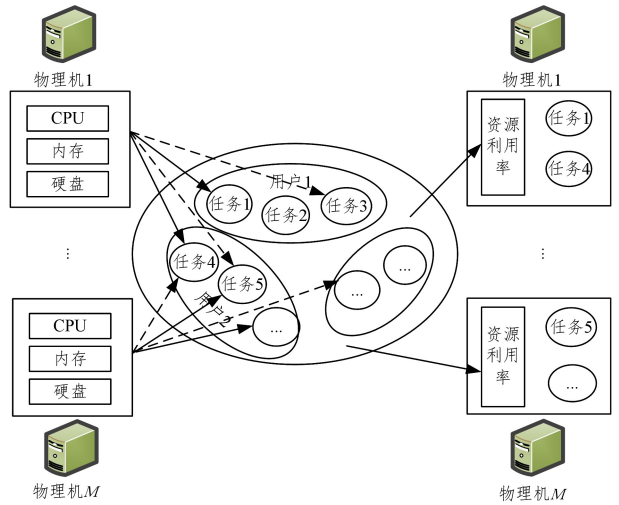

图 1 云资源调度拓扑结构示意图

云供应商资源博弈(PRAG)可以用一个四元组表示 $PRAG=\{M,Req,S,Utility\}$。其中，M 为博弈的参与者，就是所有拥有可用物理资源的物理机$\{m_1,m_2,\cdots,m_M\}$；Req 是云中的请求矩阵；$S=\{S(m_i)\mid i=1,2\cdots,M\}$是博弈参与者的备选策略；$Utility=\{Utility^{(m_i)}\mid i=1,2,\cdots,M\}$是收益函数，用以计算参与博弈的实体物理机选定的博弈策略，即每个参与者可以获得的利益。

如何最合理且有效地利用数据中心资源，满足用户 QoS 要求并提高资源使用效率，减少资源浪费，是最值得关注的问题。云资源博弈的数学模型可用如式(1)、式(2)表示：

$$Max_i\text{-}Utility^{(m_i)}$$

$$\text{s.t.}\begin{cases}s_i(m_i)\in N\\ \sum_i s_i(m_i)\cdot r_{ik}\leqslant R_k(m_i)\\ 0\leqslant ERT_i\leqslant RT_i\\ 0\leqslant\sum_k a_{ik}\cdot price_k\leqslant Cost_i\end{cases}\tag{2}$$

第一个约束条件表示为一个整体规划问题，第二个约束条件表示每个物理机中分配的虚拟单元的总资源不超过该物理机的可用资源总量。第三个约束条件表示每个用户对每个子任务的实际响应时间在约束中的最大响应时间范围内。第四个约束条件表示成本开销在 QoS 约束中的预算范围内。通过博弈模型中收益函数的设计，其优化算法能为每个物理机求解到合适的可用资源调度策略 $S(m_i)$。收益函数是对云中心的资源利用率、资源分配的公平性以及多类型优化目标的权衡。

为了从用户的角度研究云计算环境下的资源调度问题，对用户需求的分析至关重要。考虑到云计算运行过程中用户的实际需求，本文从最低需求和用户偏好两个方面对用户的差异性需求进行建模，通过使用二元组将用户提交的任务 t_i 表示为 $t_i=(tr_i,tp_i)$，式中 tr_i 表示用户任务 t_i 的最低需求，tp_i 表示为用户任务 t_i 的偏好。tr_i 和 tp_i 均可在用户提交任务时得到。

用户在实际应用中的需求往往是多方面的，包括时间性需求、安全性需求、可靠性需求、信任性需求、消费需求等。失

一般性,假设用户需求的维度是 d,则用户任务的最低需求 tr_i 定义为 $tr_i=[tr_{i1},tr_{i2},\cdots,tr_{id}]$。以此类推,则有 L 个用户任务的需求可表示为任务需求矩阵 $TR_{L\times d}=[tr_1\ \ tr_2\ \ \cdots\ tr_L]^T$,$TR$ 反映用户任务在各个维度上的最低需求。

同理,用户偏好 tp_i 可定义为 $tp_i=[tp_{i1},tp_{i2},\cdots,tp_{id}]$,则所有 L 个用户任务的偏好可以表示为任务偏好矩阵 $TP_{L\times d}=[tp_1\ \ tp_2\ \ \cdots\ tp_L]^T$,$TP$ 反映了用户对各个需求的重视程度。

在实际应用中,不同的任务需求具有不同的表达形式和数值范围。为综合评价需要,对各个任务需求的值进行规一划处理,使得所有任务需求的值具有可比性。可以采用最小-最大规格化方法对任务需求矩阵 TR 中任务需求的取值进行线性变换,消除不同量纲对调度结果的影响。归一化处理后,任务需求矩阵 TR 中的每一个 tr_{ik}($1\leqslant i\leqslant L$,$1\leqslant k\leqslant d$)都将满足 $0\leqslant tr_{ik}\leqslant 1$。

用户偏好的本质是用户对不同任务需求的关注程度存在差异。根据用户自身的实际需求,更加关注其中的某些任务需求而不是重视其他需求,这是实际应用中的常态。因此,需要考虑用户任务需求的所有维度,对每个任务需求的重视程度进行衡量。本文考虑采用层次分析法(Analytic Hierarchy Process,AHP)对各个任务需求偏好进行两两对比,尽可能降低由于主观判断不一致造成的影响,使用户的偏好信息量化比较得以体现和传递,从而实现对用户的按需服务。经过处理,任务偏好矩阵 TP 中每个 tp_{ik}($1\leqslant i\leqslant L$,$1\leqslant k\leqslant d$)在云任务 i 和用户需求维度 k 中都能满足条件。

2　公平性-有效性权衡博弈算法

公平性-有效性权衡博弈算法(Fairness-Utilization Tradeoff Gme,FUTG)把某个资源决策时刻的整个调度过程模拟化为一个扩展博弈,构成云服务的每台物理机都被视为博弈中的参与者,并且每台物理机中的虚拟机创建方案是参与者的备选方案。每个博弈参与者 m_i 都希望得到使自身利益最高的资源调度策略,所以一个资源调度问题就转换为如下公式所示的最优化问题。

$$\begin{aligned}&Max\text{-}Utility^{(m_i)}(A)\\&\text{s. t. } a_{ik}(m_i)\geqslant 0\\&\sum_i a_{ik}(m_i)\leqslant R_k(m_i)\\&0\leqslant tp_{ik}\leqslant 1\\&\text{s. t. } \sum_{k=1}^{d} tp_{ik}=1\end{aligned}\tag{3}$$

其中,$A^*=\{A^*(m_1),A^*(m_2),\cdots,A^*(m_M)\}$表示为资源调度博弈的纳什均衡解,即也可以表示为:

$$\forall i,Utility^{(m_i)}(A^*(m_1),A^*(m_2),\cdots,A^*(m_i),\cdots,A^*(m_M))>Utility^{(m_i)}(A^*(m_1),A^*(m_2),\cdots,A^1(m_i),\cdots,A^*(m_M))$$

云资源调度目标函数所求解为非合作博弈的博弈纳什均衡解,所以就是最优的资源调度方案。

FUTG 算法的流程如下:

1. Initialization:combinLists,selectedServerList,selection[M+1]
2. 步骤 1:预先组合阶段
3. //云数据中心拥有可用资源的物理机为博弈者
4. PM←{1,…,i,…,M}
5. For each physical server m_i do
6. List any possible coordinate placement combinations of this server to be fulfilled by different types of VMs without exceeding the capacity in the combinList$_i$
7. combinLists.add(combinList$_i$) // 对于每台物理机,统计待创建虚拟机类型可能的组合方案
8. 步骤 2:参与者策略空间生成
9. for each physical server m_i do
10. Pick up top g of combinations $O^{(m_i)}=\{com_1,com_2,\cdots,com_g\}$ and calculate min($O^{(m_i)}$)
11. //每个 $com_g(m_i)$可以转化为分配矩阵 $A_g(m_i)$
12. 基于 TR、TP、匹配阈值 θ 和分配矩阵 $A_g(m_i)$判断
13. 更新用户任务 t_i 的 TR 和 TP
14. 步骤 3:扩展博弈有向树生成
15. The orginal array $[\min(O^{(1)}),\cdots,\min(O^{(M)})]$ is rearranged in a non-decreasing order with indices $[l_1,\cdots,l_M]$ such that $\min(O^{(l_1)})\leqslant\cdots\leqslant\min(O^{(l_M)})$ //按照 min(O(m))值对物理机进行非递减排序
    The game players take action as the order of $[l_1,\cdots,l_M]$ //物理机排序依次选择博弈策略
16. 步骤 4:求解子博弈纳什均衡策略

FUTG 云资源调度算法代码如下:

1. max[x]←argMax$_y$Utility$^{(l_M)}$[x][y]
2. selection[l_{M-1}]←argMax$_x$U$^{(l_{M-1})}$[x][max[x]]
3. selection[l_M]←argMax$_x$[max[x]]
4. Add l_{M-1} to the selectedServerList
5. for each physical server m_i from l_{M-2} to 1 do
6. Add up the total amount of resources for physical servers in selected ServerList
7. for each strategy $A_g(m_i)$ of physical server m do
8. Calculate the ske(m_i) if $A_g(m_i)$ is chosen //计算每种博弈分配方案对应的偏度值
9. Add up the total allocated resource=$A_g(m_i)+\varphi$
10. for i:=1 to L do
11. if sim(tr_i,$A_g(m_i)$)$\geqslant\theta$ then //计算 tr_i,$A_g(m_i)$之间的相似性
12. AR_i←r_j
13. Calculate the v(A) //计算每种博弈分配方案的公平偏差值
14. utilityCalculation(ske(m_i),v(A)) //生成收益矩阵
15. end for
16. selection[m_i]←argMax$_x$(Utility$^{(m_i)}$[x])
17. Add server m_i to selectedServerList
18. end for
19. The best strategy of each player m_i in selection[m_i] can be represented as an allocation matrix A(m_i), and $A^*=\{A^*(m_1),A^*(m_2),\cdots,A^*(m_M)\}$ //子博弈纳什均衡解可以转成分配矩阵表示形式

3　仿真实验

本文使用 Google Cluster Trance 作为数据中心工作负载的参考数据。如表 1 所列,Google Cluster Trance 收集了谷歌集群在 10 天内工作参数的变化,包括用户信息、工作类型、工作数量、子任务数量、子任务开始和结束时间、子任务资源请求参数、资源利用率等、CPU 需求量、内存需求量、硬盘需求量的范围,每天的物理机的工作数量以及每天的任务量都明确给出,同时也获取到任务的运行时间范围。实验有以下假设:1)实验只考虑两种数据类型,即 CPU 和内存;2)用户请求中给出了所有任务类型对应的虚拟机类型;3)云平台会在

万方数据

每个决策之前预先判断共有多少可用于资源可用调配。

表 1 数据中心工作负载参数

	工作数量	任务数量	任务运行时间/s	CPU需求	内存需求/GB	硬盘需求/GB
1	9	350	[0,78]	[0,4]	[0,3.5]	[0,37]
2	7	90	[0,92]	[0,4]	[0,7]	[0,7]
3	14	109	[0,85]	[0,4]	[0,10]	[0,80]
4	3	95	[10,125]	[0,4]	[0,6.5]	[0,22]
5	7	230	[0,45]	[0,8]	[0,5]	[0,17]
6	8	102	[0,71]	[0,4]	[0,10]	[0,16]
7	4	90	[20,100]	[0,4]	[0,3.5]	[0,21]
8	18	196	[0,70]	[0,4]	[0,5]	[0,27]
9	14	101	[0,60]	[0,8]	[0,3]	[0,10]
10	30	237	[10,104]	[0,4]	[0,11]	[0,100]

图 2 和图 3 给出了 EFFD 算法、谷歌集群和 FUTG 算法资源利用率的数据对比情况。从图 2 中可以看出：EFFD 算法中 CPU 的利用率为 50%左右，谷歌集群大部分处于 40%～50%之间，但是 FUTG 算法的 CPU 利用率一半以上都超过 50%，有的甚至达到 67%。图 3 显示了内存利用率情况，EFFD 算法的内存利用率保持在 40%左右，谷歌集群的内存利用率在 40%到 50%之间，而 FUTG 算法的内存资源利用率绝大多数都在 50%以上。

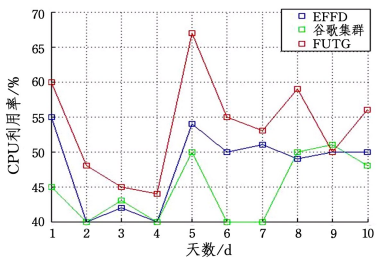

图 2 CPU 利用率的对比

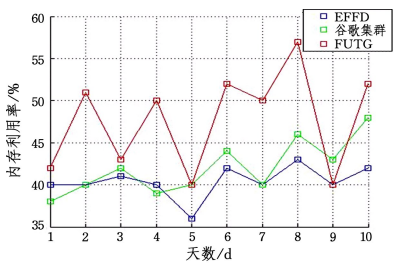

图 3 内存利用率的对比

用户满意度表征了用户对调度结果的满意程度。本文中用调度结果和用用户需求之间的匹配程度来衡量用户满意度。用户需求则是根据云计算过程中用户的实际需求，从最低需求和用户偏好两方面考量。

$\theta=0.2$ 时，用户的满意度如图 4 所示。

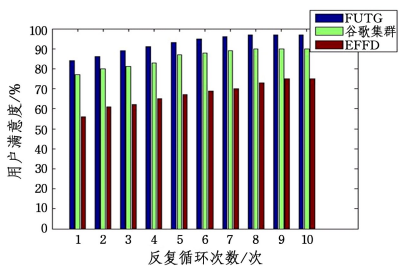

图 4 用户满意度的对比($\theta=0.2$)

$\theta=0.5$ 时，用户的满意度如图 5 所示。

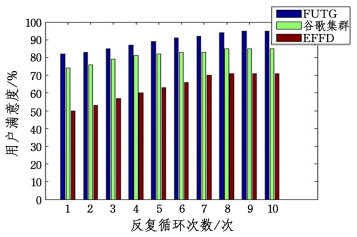

图 5 用户满意度的对比($\theta=0.5$)

$\theta=0.8$ 时，用户的满意度如图 6 所示。

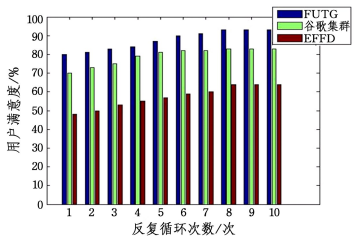

图 6 用户满意度的对比($\theta=0.8$)

图 4－图 6 展示了匹配阈值 θ 分别为 0.2，0.5，0.8 条件下，3 种算法在反馈循环次数变换时的用户满意度变化情况。从 3 个图中可以看出，FUTG 算法的用户满意度普遍高于 80%，在阈值为 0.2 时更是达到 97%。相比之下，谷歌集群的用户满意度在 80%左右，而且随着阈值的增加，用户满意度有所下降。EFFD 算法的用户满意度最低，在阈值为 0.8、反复循环次数为一次的情况下低于 50%。

结束语 本文研究了 FUTG 的算法，并通过一组仿真实验模拟了谷歌集群机制负载收集实验结果。与 EFFD 算法和 Google 管理机制的对比验证得出 FUTG 算法在动态负载下各类资源利用的高效性。从 3 组阈值之间的对比和每组内部 3 种算法之间的对比看出：FUTG 算法的用户满意度更高，更符合实际云服务需求。云计算中的资源具有异构性和自治性的特点，而且往往是分布式的，云计算中的用户需求也千差万别，这些都使得云计算环境下的资源调度问题更加复杂。因此，如何在云计算资源中获取用户真实资源需求，实现高效、经济、准确的资源调度，是当前迫切需要解决的问题，这就要求研究人员根据云计算的发展趋势做出正确分析，提出新的方法。

参 考 文 献

[1] XIN J，KWONG K Y，YONG Y. Competitive Cloud Resource Procurements via Cloud Brokerage[C]// Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science. IEEE，2016：355-362.

[2] OSORIO I，ZAVERI H，FREI MG，et al. Epilepsy：the intersection-of-neurosciences，biology，mathematics，engineering and physics [M]. CRC，2015.

[3] CARROLL T E，GROSU D. Formation of virtual organizations in grids：a game-theoretic approach[J]. Concurrency and Computation：PracticeandExperience，2017，22(14)：1972-1989.

[4] HASSAN M，SONG B，HUB E N. Game-based distributed resource allocation in horizontal dynamic cloud federation platform [M]// Algorithms and Architectures for Parallel Processing. Berlin：Springer，2011：194-205.

[5] 张晞. 云计算环境下改进的虚拟机资源调度算法研究[J]. 科技通报，2018，34(2)：155-158

[6] 丁丁，艾丽华，罗四维，等. 基于用户行为反馈的云资源调度机制

万方数据

[J]. 系统工程与电子技术，2018，40(1)：209-216.
[7] 史宝鹏，段迅，孔广黔，等. 医疗云平台资源调度策略研究[J]. 计算机工程，2017，43(8)：44-48，55.
[8] MITTAL A，KAUR P D. Genetic based QoS task scheduling in cloud-upgrade genetic algorithm[J]. International Journal of Grid and Distributed Computing，2015，8(4)：145-152.
[9] PANDITD，CHATTOPADHYAYS，CHATTOPADHYAY M，et al. Resource allocation in cloud using simulated annealing[C]// Proc. of the International Conference on Applications and Innovations in Moble Computing. 2014：21-27.
[10] 张永强，徐宗昌，呼凯凯，等. 基于私有云和改进粒子群算法的约束优化求解[J]. 系统工程与电子技术，2016，38(5)：1086-1092.
[11] SINGH J，MISHRA S. Improved ant colony load balancing algorithm in cloud computing[J]. International Journal of Computers and Technology，2015，4(3)：5636-5644.
[12] 李智勇，陈少淼，杨波，等. 异构云环境多目标 Memetic 优化任务调度方法[J]. 计算机学报，2016，39(2)：377-390.
[13] WANG W，LI B，LIANG B. Dominant resource fairness in cloud computing systems with heterogeneous servers[J]. arXiv：1308.0083，2013.
[14] MATTHEWS J N，ANDERSON T E. 22nd symposium on Operating systems principles[C]// ACM Symposium on Operating Systems. Principles，ACM，2009：261-276.
[15] 齐平，王福成，王必晴. 一种基于图模型的可信云资源调度算法[J]. 山东大学学报(理学版)，2018，53(1)：63-74.
[16] 王涛，杨喆. 数据中心中云计算资源调度算法的浅入分析[J]. 自动化技术与应用，2018，37(1)：47-48，59.
[17] 王琛，汤红波，游伟，等. 一种 5G 网络低时延资源调度算法[J]. 西安交通大学学报，2018(4)：1-7.
[18] 徐昕. 基于博弈论的云计算资源调度方法研究[D]. 上海：华东理工大学，2015.
[19] 李超，戴炳荣，旷志光，等. 云计算环境下基于改进遗传算法的多维约束任务调度研究[J]. 小型微型计算机系统，2017，38(9)：1945-1949.
[20] 张素芹，徐飞. 多目标约束条件的云计算资源调度算法仿真[J]. 价值工程，2017，36(22)：216-218.

(上接第 286 页)

图 13　视频 4 测试效果

4.2　对比实验结果分析

对 4 个视频中的 1700 帧视频图像进行检测，检测结果如表 1 所列。实验统计表明，虽然仍存在误检和漏检等现象，但完整的功能模块对火灾的识别率达到了 97%以上。

表 1　火焰检测结果

视频	总视频帧数	火焰帧数	非火焰帧数	漏检帧数	漏检率/%	误检帧数	误检率/%
视频 1	450	407	43	6	1.3	3	0.7
视频 2	225	225	0	2	0.9	—	—
视频 3	425	425	0	2	0.5	—	—
视频 4	600	0	600	—	—	13	2.1

结束语　本文提出了一种基于动静态特征的视频火灾检测算法，并给出了仿真实现的具体流程。分析视频图像中的静态特征，得到疑似火焰图像，再通过动态特征进一步判断其是否为火焰。实验结果表明本文的方法能够去除一些由静态特征分析产生的非火焰的干扰，从而降低了误检率，具有一定的实用性。为了提高计算效率，本文的动静态特征采用固定算法及固定阈值，在进一步的研究中可以考虑自适应特征提取算法及阈值。

参 考 文 献

[1] CELIK T，DEMIREL H. Fire detection using statistical color model in video sequences[J]. Journal of Visual Communication & Image Representation，2007，18(2)：176-185.
[2] ÇELIK T，DEMIREL H. Fire detection in video sequences using a generic color model[J]. Fire Safety Journal，2009，44(2)：147-158.
[3] HORNG W B，PENG J W，CHEN C Y. A new image-based real-time flame detection method using color analysis[C]// Networking，Sensing and Control，2005. IEEE，2005：100-105.
[4] LIU C B，AHUJA N. Vision Based Fire Detection[C]// International Conference on Pattern Recognition. IEEE Computer Society，2004：134-137.
[5] TÖREYIN B U，DEDEO ĞLU Y，GÜDÜKBAY U，et al. Computer vision based method for real-time fire and flame detection[J]. Pattern Recognition Letters，2006，27(1)：49-58.
[6] JENIFER P. Effective visual fire detection in video sequences using probabilistic approach[C]// International Conference on Emerging Trands in Electrical & Computer Tehcnology. IEEE，2011.
[7] LAFARGE F，DESCOMBES X，ZERUBIA J. Textural kernel for SVM classification in remote sensing：application to forest fire detection and urban area extraction[C]// IEEE International Conference on Image Processing. IEEE，2005：III-1096-9.
[8] KO B C，CHEONG K H，NAM J Y. Fire detection based on vision sensor and support vector machines[J]. Fire Safety Journal，2009，44(3)：322-329.
[9] ZHAO J H，ZHANG Z，HAN S Z，et al. SVM based forest fire detection using static and dynamic features[J]. Computer Science & Information Systems，2011，8(8)：821-841.
[10] CHO B H，BAE J W，JUNG S H. Image Processing-Based Fire Detection System Using Statistic Color Model[C]// International Conference on Advanced Language Processing and Web Information Technology. IEEE，2008：245-250.
[11] SHAO J，WANG G，GUO W. An image-based fire detection method using color analysis[C]// International Conference on Computer Science and Information Processing. IEEE，2012：1008-1011.
[12] PÉTERI R，FAZEKAS S，HUISKES M J. DynTex：A comprehensive database of dynamic textures[J]. Pattern Recognition Letters，2010，31(12)：1627-1632.
[13] 邵婧，王冠香，郭蔚. 基于视频动态纹理的火灾检测[J]. 中国图象图形学报，2013，18(6)：38-44.
[14] 许宏科，房建武，文常保. 基于亮度与火焰区域边缘颜色分布的火焰检测[J]. 计算机应用研究，2010，27(9)：3581-3584.

万方数据