云中多媒体应用中基于混合DAG的最优任务调度研究

郭雅琼 宋建新

(南京邮电大学江苏省图像传输与处理重点实验室 南京 210003)

摘 要 云计算的平台优势使得它在多媒体应用中得到广泛使用。由于多媒体服务的多样性和异构性,如何将多媒体任务有效地调度至虚拟机进行处理成为当前多媒体应用的研究重点。对此,研究了云中多媒体最优任务调度问题,首先引入有向无环图来模拟任务中的优先级及任务之间的依赖性,分别对串行、并行、混合结构任务调度模型进行任务调度研究,根据有限资源成本将关键路径中任务节点融合,提出一种实用的启发式近似最优调度方法。实验结果表明,所提调度方法能够以最短的执行时间在有限的资源成本下完成最优的任务分配。

关键词 关键路径,有限无环图,任务级调度,启发式调度

中图法分类号 TP37 **文献标识码** A

Optimal Task-level Scheduling Based on Multimedia Applications in Cloud

GUO Ya-qiong SONG Jian-xin

(Jiangsu Key Laboratory of Image Transmission & Processing, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract As an emerging computing paradigm, cloud computing has been increasingly used in multimedia applications. Because of the diversity and heterogeneity of multimedia services, how to effectively schedule multimedia tasks to multiple virtual machines for processing has become one fundamental challenge for application providers. So we studied task-level scheduling problem for cloud based multimedia applications. Firstly, we introduced a directed acyclic graph to model precedence constraints and dependency among tasks in the hybrid structure. Based on the model, we studied the optimal task scheduling problem for the sequential, the parallel, and the mixed structures. Moreover, we combined the task nodes in the critical path according to the cost of limited resources. Lastly, we proposed a heuristic method to perform the near optimal task scheduling in a practical way. Experimental results demonstrate that the proposed scheduling scheme can optimally assign tasks to virtual machines to minimize the execution time.

Keywords Critical path, Directed acyclic graph, Task-level scheduling, Heuristic scheduling

1 研究现状

云计算是分布式计算、并行计算、网格计算、效用计算、虚拟化网络存储等传统计算机和网络技术发展融合的商业实现。在云数据中心,通过管理服务器共享池来提供按需计算、通信和存储等接入服务[1]。尤其是多媒体应用的密集任务需求,使得云计算得到了更加广泛的使用。在基于云平台的多媒体应用[2]中,应用提供商供给一定数量的虚拟机来处理多媒体任务,并迫切需要有效的调度算法将多媒体任务最优地分配给各种VM(virtual machine)进行处理。目前,当多媒体应用配置在云上时,可以用工作流和执行方案来描述。工作流由一组抽象任务集和任务间的依赖组成,描述了应用逻辑上的工作过程 。

多媒体应用可以使用由任务集和任务之间依赖性组成的工作流来描述,根据任务之间的依赖性,进一步制定执行方案。工作流中任务通常是一些抽象的功能模块,处理节点则是应用中任何任务的逻辑执行节点。选择最优的执行方案可以将处理节点有效地分配给不同的任务,工作流和执行方案都可以采用有向无环图DAG来描述。图中的顶点代表了一个任务或处理节点,边缘则代表了任务间的依赖性[3,4]。

通常,云计算虚拟资源调度有两种形式。一种是用户级任务调度,按照当前工作负载将每个应用的用户请求分配给不同的VMs,通过均衡VMs的负载,可以避免云中用户过多引起的堵塞;相对用户级调度来说,任务级调度则趋于更细化的调度工作,通常可以将多媒体应用分解成若干任务集,任务集可以是串行、并行或者混合型 。任务级调度的最终目的就是以最小的总执行时间将任务分配给VMs。而两级调度则是以分级的方式在云中进行。

云中多媒体应用引起了广大学者的关注。文献[5]提出了云媒体中的最优负载调度方法,其利用贪婪算法来优化响应时间和资源成本,从而达到均衡负载的目的。文献[6]提出了一种基于队列模型的资源分配方法,以优化服务的响应时间。文献[7]提出了一种基于突发感知的自适应资源分配算法,即利用突发负载均衡器预测用户需求的变化,根据预测信息进行适应性的调度。但是文献[5-7]都是从用户级考虑调度问题的。文献[8]根据媒体任务QoS属性进行任务调度,利用节点利用率实现各服务节点的高效利用。文献[9]利用启发式调度算法来减少任务处理的成本,但是文献[8,9]没有综合考虑资源成本与响应时间的关系,而本文对此进行了云媒体的任务级调度研究。

郭雅琼(1990—),女,硕士生,主要研究方向为图像处理与多媒体通信,E-mail:Gyqnj0625@126.com;**宋建新**(1959—),男,教授,硕士生导师,主要研究方向为图像处理、多媒体通信、计算机视觉。

云中，有效的任务级调度可以使 VMs 资源最优匹配任务需求，从而提高服务质量。然而，找到最优的任务级调度依然是极具挑战性的工作。首先，在多种任务之间存在预先限制条件，从而限制任务的执行顺序。图 1 显示了视频检索架构[10]。当不同任务分配给 VMs 时，很难满足所有的预先限制条件。其次，多媒体应用有不同的结构，一般分为 3 种：串行、并行和混合结构。串行结构中，任务必须按照顺序执行；并行结构中，任务可以并发执行；混合结构结合串行和并行结构，很难针对不同的结构找到有效的调度方法。最后，VMs 有不同的资源性能。如何将一个任务分配给最优的虚拟机，且满足任务的时变要求，依然具有挑战性。

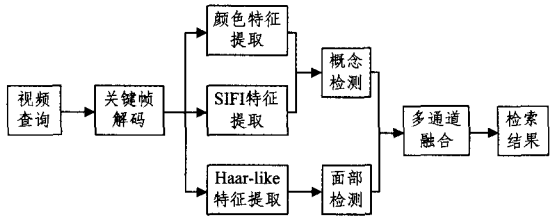


图 1　视频检索架构

在云计算环境中，任务资源调度是一个 NP 完全问题。解决资源调度的方法主要是一些智能算法[11,12]，如模拟退火算法、粒子群算法、蚁群算法和遗传算法等。但是它们的复杂度过高，易受某些启发参数的影响，对云媒体应用场合不太适用。

本文针对云媒体应用中不同任务结构模型，融合关键路径上的节点，采用所提出的启发调度方法，能够缩短系统处理调度问题的时间，从而减少总任务执行时间，提高云计算任务调度的效率。

本文第 2 节描述了多媒体任务调度模型以及不同结构任务调度问题；第 3 节主要讨论了在资源成本限制条件下提出的关键路径融合算法和启发式调度方法；第 4 节进行仿真分析；最后总结全文。

2　不同结构任务调度模型

2.1　任务调度模型及条件

本节中，首先建立任务调度模型。根据多媒体任务之间的预先限制条件，引入有向无环图 DAG（Directed Acyclic Graph），不存在开始和结束在同一点的路径。DAG 表示为：$DAG=(V,E)$，V 代表任务顶点集，E 是边集。假定多媒体应用分解成 K 个任务。DAG 中每个顶点代表一个任务。这样，$V=\{V_1,V_2,\cdots,V_k\}$ 代表任务集。DAG 中边集表示两个任务之间的限制条件。$E_{k',k}$ 代表任务 V_k 只能在 V_k' 之后才能执行。图 2 为视频检索架构的 DAG 图。其中存在 7 个任务，视频查询和检索结果代表源和终点，即 DAG 的开始和结束。

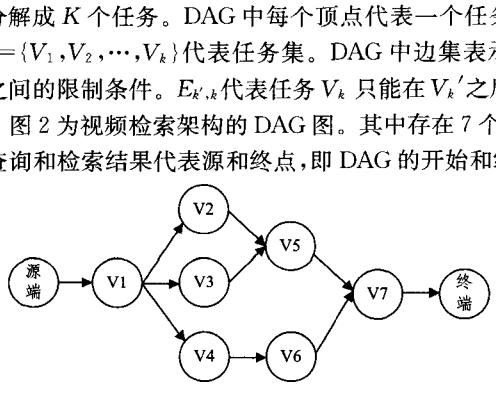


图 2　视频检索架构的 DAG 图

假定有 N 类 VM，每个任务可以在任何 VM 上工作，但是不同类型的 VM 有不同的价格速率和资源性能，从而导致不同的执行时间。t_j^k 代表用 j 类 VM 执行任务 V_k 所需的时间，p_j 是租用 j 类 VM 的资源成本。假定每个任务只能分配给一个 VM 执行。那么，任务调度的最终目的就是将任务最优地分配给虚拟机。引入 $s_j^k\in\{0,1\}$，1 表示任务 V_k 分配给 j 类 VM，反之亦然。本文提出最优的任务调度方案来确定 s_j^k $(V_k\in(V),\forall j=1,2,\cdots,N)$，从而减少总执行时间。

任务的执行时间主要有两部分：与边缘任务通信的传输时间和自身任务的处理时间。本文主要针对这两部分进行优化，通过关键路径结合方法，优化传输过程，从而减少传输时间。关键路径上的传输时间 T_{trans} 由式(1)给出：

$$T_{trans}=t_1+t_2+\cdots+t_e$$
$$t_i=\frac{D_i}{v_i} \tag{1}$$

式中，t_i 是关键路径 CP 边缘节点的 i 传输时间，e 是 CP 上的边缘数目，D_i 是边缘 i 将要传输的数据大小，v_i 是边缘传输速率。由边缘节点连接的任务可以合并为一个逻辑组合任务，这样有效地降低了它们之间的通信（可以看作为 0），把边缘成本比较高的节点结合起来，传输时间也相应地减少。考虑到 CP 可以控制任务的最新完成时间，将两个关键任务组合成一个新任务可以有效缩短传输时间。结合过程可以持续，直到没有相邻节点存在为止，这样形成一个新的逻辑图和 CP，重新调整 CP 上的总传输时间 T_{trans}。边缘结合过程见 3.1节。

除了优化传输时间，还需要优化自身节点的处理时间。CP 上 T_{trans} 所有任务的自身处理时间之和为 T_{exe}。任务 j 的处理时间为 t_j，d 为 CP 上的任务数目。可以得到总任务处理时间：

$$T_{exe}=t_1+t_2+\cdots+t_d$$
$$t_j=\frac{D_j}{v_j} \tag{2}$$

D_j 是数据大小，v_j 是任务 j 的自身执行时间。

在对 CP 上边缘传输时间优化后，引入了一个最终调度条件：

$$\rho_j^k=\frac{\Delta t_j^k}{\Delta C_j^k} \tag{3}$$

V_k 调度至 VM 缩短 Δt_j^k 时间，需要多花费 ΔC_j^k，当 ρ_j^k 取最大值时，将 V_k 重新调度至 j 类 VM。

2.2　串行结构任务调度问题

串行结构中，所有的任务都是按顺序执行，目标就是使总执行时间最少。由前所述，每个任务只能分配给一个 VM。任务 V_k 由 j 类 VM 执行调度，$s_j^k=1$ 和 $s_{j'}^k=1(j'\neq j)$。这样，V_k 总执行时间为 $\sum_{j=1}^{n}s_j^kt_j^k$，即若 V_k 被虚拟机调度，j 类 VM 需要 t_j^k 来执行任务 V_k。由于所有任务都是以串行结构执行，总执行时间是每个任务执行时间之和，因此总执行时间为 $T_{seq}^{tot}=\sum_{k=1}^{K}\sum_{j=1}^{N}s_j^kt_j^k$，处理 V_k 花费的成本为 $\sum_{j=1}^{N}s_j^kp_j$。这样，完成多媒体任务的总资源成本是 $C_{seq}^{tot}=\sum_{k=1}^{K}\sum_{j=1}^{N}s_j^kp_j$。另外，每个任务必须分配给一个 VM 执行。因此，必须满足 $\sum_{j=1}^{N}s_j^k=1(V_k\in\mathbb{V})$。根据以上分析，串行结构的任务调度最优问题可以描述为：

$$\begin{aligned}&\underset{\{s_j^k\}}{\text{Minimize}}\ \sum_{k=1}^{K}\sum_{j=1}^{N}s_j^kt_j^k\\&\text{Subject to}\ \sum_{k=1}^{K}\sum_{j=1}^{N}s_j^kp_j\leqslant C_{upp}\\&\sum_{j=1}^{N}s_j^k=1\\&s_j^k\in\{0,1\},V_k\in\mathbb{V},\forall j=1,\cdots,N\end{aligned} \tag{4}$$

C_{upp} 代表资源成本上限。

2.3 并行结构任务调度问题

本节研究了并行结构的最优任务调度问题，所有任务都是并行结构，因此必须并发执行。根据 2.2 节所述，V_k 的执行时间为 $\sum_{j=1}^{N} s_j^k t_j^k$。并行结构的总执行时间依赖于所有任务的最大执行时间，即 $T_{par}^{tot} = \max_{\{V_k \in V\}} \{\sum_{j=1}^{N} s_j^k t_j^k\}$。总资源成本 $C_{par}^{tot} = \sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k p_j$。另外，$\sum_{j=1}^{N} s_j^k = 1\{V_k \in \mathbb{V}\}$ 保证所有任务都能执行。因此，并行结构的任务调度优化问题可以描述为：

$$\begin{aligned} &\text{Minimize} \max_{\{V_k \in V\}} \{\sum_{j=1}^{N} s_j^k t_j^k\} \\ &\text{Subject to} \sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k p_j \leqslant C_{upp} \\ &\sum_{j=1}^{N} s_j^k = 1 \\ &s_j^k \in \{0,1\}, V_k \in \mathbb{V}, \forall j = 1, \cdots, N \end{aligned} \tag{5}$$

问题(5)的优化是 0-1 整形规划问题。

2.4 混合结构任务调度问题

在混合结构中，一些任务并行处理，另外一些任务进行串行处理，目标是总执行时间最小。图 1 视频检索架构就是一个混合结构应用。混合结构的有向无环图 DAG 从源端到终端有多条路径。每一条路经都必须进行串行处理，不同路经彼此之间又是并行结构。假定一个 DAG 中有 W 条路经，令 Ψ_w 为第 w 条路径上的顶点集合。由前所述，每个任务只能分配给一个 VM。任务 V_k 由 j 类 VM 执行调度，$s_j^k = 1$ 和 $s_{j'}^k = 1 (j' \neq j)$。V_k 的执行时间为 $\sum_{j=1}^{N} s_j^k t_j^k$，即若 V_k 被虚拟机调度，j 类 VM 需要 t_j^k 来执行任务 V_k。这样 Ψ_w 集合中任务执行时间为 $\sum_{k \in \Psi_w} \sum_{j=1}^{N} s_j^k t_j^k$。因此，总任务执行时间为所有路径中最大执行时间，即关键路径的总任务执行时间 $T^{tot} = \max_{\{w \in W\}} \{\sum_{V_k \in \Psi_w} \sum_{j=1}^{N} s_j^k t_j^k\}$，资源成本为 $C^{tot} = \sum_{k \in \Psi_w} \sum_{j=1}^{N} s_j^k t_j^k$。此外，所有任务的处理必须满足 $\sum_{j=1}^{N} s_j^k = 1 (V_k \in \mathbb{V})$。因此，混合结构的最优调度问题描述为：

$$\begin{aligned} &\text{Minimize} \max_{\{w \in W\}} \{\sum_{V_k \in \Psi_w} \sum_{j=1}^{N} s_j^k t_j^k\} \\ &\text{Subject to} \sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k p_j \leqslant C_{upp} \\ &\sum_{j=1}^{N} s_j^k = 1, s_j^k \in \{0,1\}, V_k \in \mathbb{V} \\ &\forall j = 1, \cdots, N; \forall w = 1, 2, \cdots, W \end{aligned} \tag{6}$$

C_{upp} 代表资源成本上限。式(6)所示最优问题是 0-1 整形规划问题，可以通过数学模型计算每一个 s_j^k，但是时间复杂度不能满足实际需求。因此，本文提出启发式调度算法来解决上述最优调度。

3 最优任务级调度

本节主要描述了具体的最优任务调度方法。任务调度问题是 N-P 类难题。沿着任务和 VM 两个方向搜索，可以获得最优解。然而在任务数庞大时，采用枚举法效率极低，因此，本文提出了一种启发式最优调度算法。该方法首先找出关键路径，根据关键路径结合方法将有向图中的依赖性任务组合起来，形成新的有向无环图，减少任务间的传输时间；然后调用本文所提启发式调度方法，在满足资源成本限制条件下，加快关键任务的执行速度，实现对虚拟机的优化分配，执行完成所有任务[13,14]。

3.1 关键路径融合方法

通过遍历整个 DAG 图的所有任务，取最大完成时间的路径为工作流的关键路径。最大完成时间主要考虑了任务节点的执行时间和边缘节点传输时间。而且，工作流存在多条关键路径 CPs(critical paths)。受资源成本的限制，如果要对任务的最终完成时间进行优化，关键路径长度就必须缩短，同时，任务节点的执行时间和边缘传输时间也必须优化。本文通过合并关键路径上的任务，缩短数据传输时间，同时采用启发式算法对运行时间进行优化。关键路径融合的目的就是对 DAG 图中逻辑相邻的任务节点进行归并，合成一个新的组合任务节点。任务调度期间，这些能合并的子任务可以分配到同一服务器的虚拟机(VMs)中，这样，消除了传输延时，并能够节省带宽。

关键路径结合方法：

工作流定义为有向无环图 G。将 DAG 中的边缘按照传输成本进行处理。关键路径上由边缘点组成关键边缘列表，非关键路径上边缘节点组成剩余边缘列表。

1. 如果边缘列表非空，将关键边缘列表中的第一个边缘点赋给当前边缘值 $edge$；

2. 删除关键边缘列表中的第一个边缘点。将当前边缘点连接的两个任务组合成一个新任务，并更新有向无环图 DAG；

3. 返回步骤 1，并重复步骤 1，2，直到边缘列表中的所有边缘点移除完毕。

对于非关键路径上的边缘点组成的剩余边缘列表，采取同样的结合方法，最后调整图中数据的传输时间。

3.2 最优启发式任务级调度算法

所提的算法步骤如下：

1. 计算当每个任务调度到最便宜的 VM 时所需的 C_{tot} 和 T_{tot}。令 $\mathcal{T}^k$ 和 C^k 分别代表 V_k 任务当前执行时间和资源成本，找出关键路径。

2. 并按照 3.1 节合并关键路径上的节点。

3. 若 $C_{tot} < C_{upp}$，进行步骤 4－步骤 7，否则无操作。

4. 对于每个任务 V_k，若 $\mathcal{T}_j^k < \mathcal{T}^k$，则 $\Delta t_j^k \leftarrow \mathcal{T}^k - t_j^k$，$\Delta C_j^k \leftarrow p_j - C_k$，$\rho_j^k \leftarrow \dfrac{\Delta t_j^k}{\Delta C_j^k}$，即 V_k 调度至 VM 需要多花费 ΔC_j^k，但是能缩短 Δt_j^k 时间。

5. 返回步骤 4，直到所有关键任务执行完毕。

6. 按照降序排列 ρ_j^k，选择最大的值，并将 V_k 重新调度至 j 类 VM。更新 $T_{tot} \leftarrow T_{tot} - \Delta t_j^k$，$C_{tot} \leftarrow C_{tot} + \Delta C_j^k$，$\mathcal{T}^k \leftarrow t_j^k$，$C^k \leftarrow \rho_j$。

7. 返回步骤 3，直到循环完毕。

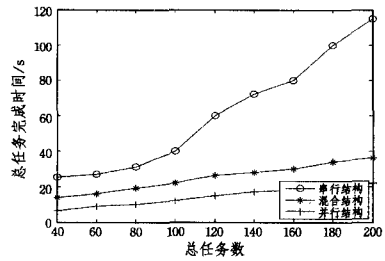
8. 返回 T_{tot}。

4 实验结果分析

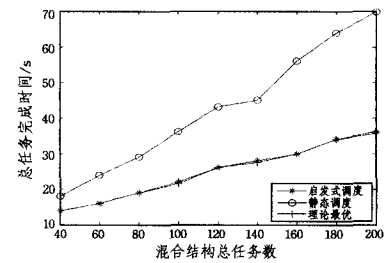
在云模拟仿真平台 Cloudsim 下，本文对所提的最优任务级调度方法进行了仿真评估，包含了数据仿真和实际的多媒体应用。实验中采用了 Amazon EC2 的 VM 配置和价格速率。详细参数见文献[15]。任务数变化范围为 40～200，每个任务的执行时间服从高斯分布。

针对串行、并行、混合型结构，分别采用所提启发式调度算法和静态执行方法对工作流总任务执行时间进行了仿真比

较。静态执行方案中，采用中型标准实例执行所有任务。本文的算法整体来说接近最优，且方便可行。串行结构、并行结构，以及混合结构的总任务执行时间结果如图 3 所示。从对比结果中可以看出，在相同的资源限制条件下，所提的启发式调度方案，理论上接近于最优方案，理论最优通过数学仿真软件实现；相对静态方案来说，可以获得更短的执行时间，且随着任务数目的增多，两种方案之间的差距也更加明显，本文算法更适合大规模任务的执行。静态方案的任务完成时间往往很长，这是因为它只采用了一种 VM 来执行所有任务，任务不能以分布式的方式得到有效处理。另外从图 3(a)中可以看出，采用同样的方案执行所有任务，混合结构往往小于串行结构，大于并行结构，从而证实了程序的加速往往依赖于并行化所占的比例。并行化程度越高，程序的执行越快，但混合结构更适合于工作流应用。



(a)3 种结构对比



(b)混合结构不同算法对比

图 3

另外，本文还采用文献[16]中视频检索结构的工作流进行仿真，所有视频序列和数据库取自 TRECVID2009。如图 4 所示，同样可以看出所提最优方案处理视频任务时执行时间更短，且随着序列数的增大，两者之间的差距变大，说明本文所提方法更适合于大型多任务多媒体应用。

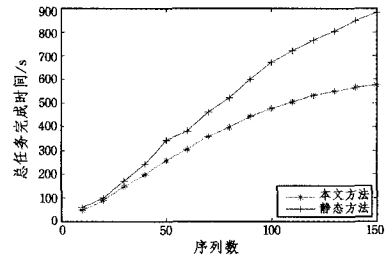


图 4　各方案处理视频任务的时间对比

结束语　本文对云中多媒体应用的最优任务级调度问题进行了研究。首先引入了有向无环图来描述任务之间的优先限制条件，并根据此模型，对 3 种结构中的任务进行了有效的任务优化调度。使得在资源成本限制下，总任务执行时间达到近似最优化。实验表明，在 VMs 资源有限的情况下，所提最优任务级调度方案能够有效满足任务调度实际需求，获得最短的执行时间，更适合于云中多媒体应用。

参 考 文 献

[1] 刘培松. 云计算环境下任务调度和资源分配策略的研究[D]. 上海：华东师范大学，2013

[2] Zhu W, Luo C, Li S. Multimedia cloud computing [J]. IEEE Signal Processing Magazine, 2011, 28(3): 59-69

[3] 晏婧. 适用于实例密集型云工作流的调度算法[J]. 计算机应用，2011, 30(11): 1-3

[4] 晏婧. 云环境下基于 QoS 约束的工作流任务调度算法研究与实现[D]. 重庆：重庆大学，2011

[5] Nan X, He Y, Guan L. Optimization of workload scheduling for multimedia cloud computing [C]// IEEE International Symposium on Circuits and Systems(ISCAS). 2013: 2872-2875

[6] Nan X, He Y, Guan L. Optimal resource allocation for multimedia cloud based on queuing model [C] // IEEE International Workshop on Multimedia Signal Processing(MMSP). 2011: 1-6

[7] Zhang T J, Li J J, et al. Adaptive resource allocation for cloud computing environments under bursty workloads [C] // IEEE Performance Computing and Communications Conference. 2011: 1-8

[8] Hong B, Tang R, Zhai Y. A resources allocation algorithm based on media task Qos in cloud Computing[C]// IEEE Software Engineering and Service Science(ICSESS). 2013: 841-844

[9] Guo L, Zhao S, Shen S. Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm [J]. Journal of Networks, 2012, 7(3): 547-553

[10] Zhao Z, Zhao Y, Gao Z, et al. BUPT-MCPRL at TRECVID 2009 [C]// Proceedings of TRECVID 2009 Workshop. 2009: 1-11

[11] 袁浩. 基于社会力群智能优化算法的云计算资源调度[J]. 计算机科学，2015, 42(4): 206-208

[12] 王波，张晓磊. 基于粒子群遗传算法的云计算任务调度研究[J]. 计算机工程与应用，2015, 51(6): 84-88

[13] Kelley J E. The critical-path method: Resources planning and scheduling [J]. Journal of Industrial Scheduling, 1963, 37(11): 108-111

[14] Gao Y, Ma H, Zhang H. Concurrency Optimized Task Scheduling for Workflows in cloud [C] // IEEE Sixth International Conference on Cloud Computing. 2013: 709-716

[15] Amazon EC2[OL]. http://aws.amazon.com/ec2/

[16] Over P, Awad G M, Fiscus, et al. TRECVID 2009-goals, tasks, data, evaluation mechanisms and metrics [OL]. http://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.org.html

(上接第 407 页)

[6] NelSon M, Lim B, Hutchins G. Fast Transparent Migration for Virtual Machines[C]// Proc. USENIX. 2005

[7] Jayasinghe D, Pu C, Eilam T, et al. Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement[C]// IEEE SCC. 2011: 72-79

[8] Chen Jian-hai, Kebin C, Ye De-shi. AAGA: Affinity-Aware Grouping for Allocation of Virtual Machines[C]// 27th International Conference on Advanced Information Networking and Applications. IEEE Press, 2013

[9] Clark C, Fraser K, Hand S, et al. Live migration of Virtual machine[C]// Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation. Boston, USA, 2005