DOI: 10.13718/j.cnki.xdzk.2018.06.022

结合 Backfilling 和空闲资源调度的云工作流调度方法①

谭海中[1]，　赵　丽[2]

1. 广州工程技术职业学院 信息工程系，广州 510925；
2. 山西大学 软件学院，太原 030006

摘要：针对云计算中工作流的科学调度问题，提出了一种快速且有效的调度方案. 首先，根据计算速度将所有的资源节点以降序方式排列；然后，调度程序通过深度优先搜索，检查任务之间的依赖关系，并根据截止期限对待执行任务进行加权排序；接着，计算每个待执行任务所使用的资源的时隙. 如果当前可用资源不能满足当前任务，则采用 Backfilling 策略，对该任务所需资源进行预留，并跳到下一个任务执行. 如果当前资源满足当前任务，则执行提出的空闲资源调度(IRS)策略，尽量安排空闲资源来执行该任务. 仿真结果表明：与当前云工作流调度技术相比，本文调度策略具有更低的任务完成时间与任务执行延迟，以及更高的资源利用率.

关　键　词：云计算；工作流调度；Backfilling 策略；空闲资源调度；任务执行延迟

中图分类号：TP393　　**文献标志码**：A　　**文章编号**：1673-9868(2018)06-0149-09

云计算是一种大规模分布式计算模式，具有抽象化、虚拟化、动态可扩展等特性，可以将计算能力、存储资源、平台和服务根据需要通过互联网提供给外部用户[1]. 其中，计算资源通常以虚拟机(Virtual Machine，VM)的形式提供[2]. 多租户是云计算的关键特征之一，因此云提供商必须允许虚拟化和资源共享以实现多租户. 租户的各种需要通常以工作流的方式提交给云供应商，为了有效地执行这些任务并最小化执行成本，需要对工作流进行合理调度. 工作流调度是在分布式资源上映射和管理具有内在关联的任务的过程[3]，它向工作流任务分配适当的资源，保证用户需求可以完整地实现. 目前，常用的调度策略有先来先服务(First Come First Served，FCFS)[4]、短时间作业优先(Shortest Processing Time，SPT)[5]、首次适应(First Fit)、最佳适应(Best Fit)、贪心算法(Greedy)等[6]. 然而，这些策略按照一定的规则进行调度，都存在一些缺陷.

分布式服务的任务调度已被证明是一个 NP 问题，因此，在多项式时间内没有最优解. 启发式算法被广泛应用以便实现接近最优解[7]，例如，文献[8]提出了一种基于开销和时间的启发式算法，以最大限度地减少云环境中调度工作流任务的执行成本、通信成本和总体完成时间. 文献[9]提出了一种基于索引的启发式调度，利用机会调度策略在可用云资源上执行并行任务. 机会调度将低优先级任务分配给间歇可用的服务器，从而最小化等待和迁移成本，然而，基于启发式的调度算法仅适用于特定类型的问题求解(例如，具有简单结构的工作流). 因此，一些学者利用智能优化算法来进行调度，例如，文献[10]提出了一种基于粒子群优化(Particle Swarm Optimization，PSO)的调度算法，在最大限度上降低云计算中工作流应用的执行成本. 然而，这些智能算法非常耗时，对于大型工作流程应用程序效果不佳.

① 收稿日期：2017-06-12
基金项目：山西省基础研究计划项目——青年科技研究基金(2014021039-6).
作者简介：谭海中(1979-)，男，硕士，高级工程师/副教授，主要从事云计算、计算机应用等方面的研究.

为此，本文提出了一种快速且有效的云工作流调度方案. 首先，根据计算速度将所有资源节点以降序方式排列，并根据截止期限对待执行任务进行加权排序；接着，计算每个待执行任务所使用的资源的时隙. 如果当前可用资源不能满足当前任务，则采用 Backfilling 调度策略，将理想的资源用适当的工作流回填，以实现更高的资源利用率，减少等待时间. 如果当前资源满足当前任务，则执行提出的空闲资源调度(Idle Resource Scheduling，IRS)策略，充分利用已调度任务之间的间隙来调度其他任务，从而减少整体的执行周期. 仿真结果表明，提出的调度策略与传统调度策略相比具有较低的任务完成时间、任务执行延迟和较高的资源利用率.

1 云工作流调度模型

1.1 云工作流调度架构

云计算环境中，工作流调度系统架构可分为4个模块，分别为：1) 工作流生成模块：当用户提交一个包含多个任务的任务集时，将任务集分解为多个子任务，并根据子任务之间的依赖关系(通信关系)，将任务集组成一个由有向无环图(Directed Acyclic Graph，DAG)表示的工作流. 同时根据用户信息和服务质量(Quality of Service，QoS)需求，为每个子任务配置一个参数列表，包含用户ID、任务提交时间、任务计算量、任务截止日期等信息，从而将该用户的任务需求构建成一个带有QoS属性的云工作流. 随后，每个租户将工作流应用程序提交给工作流调度模块；2) 调度模块：调度模块根据监视器发送的资源情况和工作流生成模块提交的工作流需求，利用本文设计的调度算法来合理调度工作流任务；3) 任务分析模块：根据用户配置或预定义设置来对资源和任务进行聚类分析；4) 资源监视器模块：在系统初始化时启动，用来发现可用资源和资源特征，并监视资源在系统中的添加/移除情况，创建活跃资源列表. 工作流调度系统架构如图1所示.

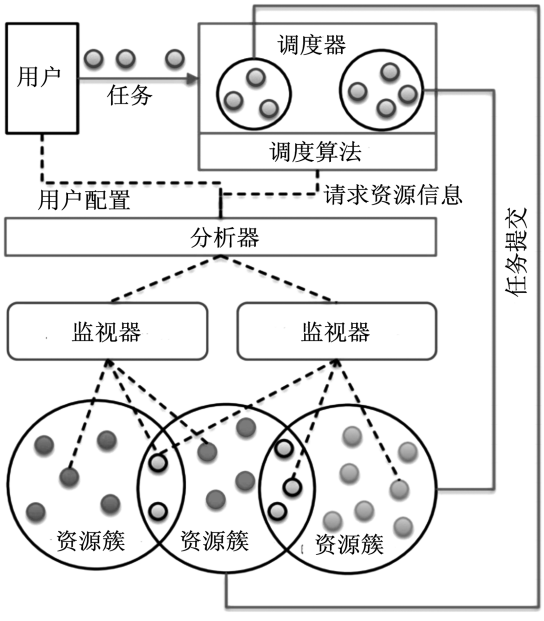

图1 工作流调度系统架构

1.2 工作流表示

云工作流可以建模为有向无环图(Directed Acyclic Graph，DAG)[11]，它由计算任务和传输任务组成. DAG是一个二元组 $G=(V, E)$，其中 $V=\{\tau_i \mid i=1, \cdots, v\}$ 为节点集合，表示任务；而 $E=\{e_{ij} \mid (i, j)\in\{1, \cdots, v\}\times\{1, \cdots, v\}\}$ 为边集合，表示2个计算任务之间的通信链路. 节点上的标签表示计算成本，边上的标签表示通信成本. 工作流任务之间的通信时间由各种因素决定，包括带宽、任务数量和传输数据量. 关键路径(Critical Path，CP)是工作流中最长的路径.

图2以DAG的形式对云工作流进行了表示. 其中从节点 τ_i 到 τ_j 的边 $e(i, j)$ 表示从 τ_j 到 τ_i 的中间结果. 没有前置任务的任务被称为入口任务，而没有后续任务的任务则为出口任务. 2个任务 τ_i 和 τ_j 之间的平均通信开销为 $C(i, j)=\mathrm{data}(i, j)/B$，其中 B 表示平均带宽，$\mathrm{data}(i, j)$ 表示在 τ_i 和 τ_j 之间传输的数据量. 如果 τ_i 和 τ_j 都分配到了相同的资源，则通信成本可忽略不计. 计算成本是资源上任务的执行成本，其值为执行该任务所需的指令总数除以资源的处理能力(以 Mbps/s 为单位).

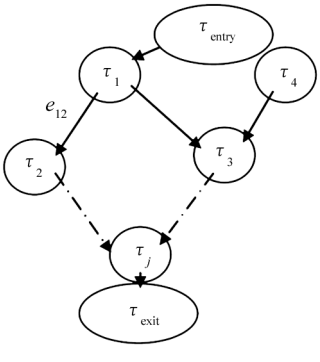

图2 云工作流的DAG表示

IaaS云可以提供一组具有不同处理能力(CPU，RAM，存储)和价格的VM实例类型. VM出租时，需要一定的启动时间才能正确初

始化，以向租户提供服务，因此，在调度和成本估计中考虑了这一时间消耗. 由于本文假设在单个云数据中心上执行工作流应用程序，物理机器之间通过高带宽链路进行互连，因此，数据传输所产生的成本相对于 VM 租赁成本来说可以忽略不计[12]，对本文实验结果没有影响. 此外，考虑到随用随付的付费模式，需要全时间段统计 VM 实例的运行时间.

2　本文设计的工作流调度策略

在本文模型中，给定一组云工作流$\{\omega_l \mid l=1, \cdots, n\}$，$\omega_l \in W$，这些工作流由计算密集型任务组成. 工作流需要在云资源场($1 \leqslant k \leqslant m$)中进行调度，在时刻$P(t)$时提供资源$R_k \in R_\Phi$，因此有$\sum_{R_k \in R_\Phi} Q_{\tau_i \omega_l R_k}$，其中$Q_{\tau_i \omega_l R_k}$表示某个工作流安排在某个资源上，表示为：

$$Q_{\tau_i \omega_l R_k} = \begin{cases} 1 & \text{如果工作流 } \omega_l \text{ 中的任务 } \tau_i \text{ 分配给资源 } R_k \\ 0 & \text{其他情况} \end{cases} \tag{1}$$

在调度过程中，每个租户以(ID, t_l, VM_l, D_l)的形式给出资源需求，并向工作流调度器提交服务请求，其中 ID，t_l，VM_l，D_l 分别表示租户 ID、提交时间、完成所需的 VM 数量以及每个工作流的相关截止日期. 工作流调度程序的目的是在给定的时间约束内(工作流 ω_l 的任务 τ_i 必须在截止时间之前完成执行)，为工作流 ω_l 分配资源 R_k.

在工作流任务执行过程中，只有先完成所有父任务之后，才能执行子任务. 在多租户环境中，许多租户将资源虚拟化并共享. 在这种共享环境中，使工作流应用程序在满足完成时间和延迟最小化的同时，提高资源利用率成为一个具有挑战性的问题.

本文提出了一种空闲资源调度(IRS)策略，并结合 Backfilling 策略构建了一种新型的调度方案，方案的伪代码如算法 1 所示.

算法 1　提出的云工作流调度算法

输入：工作流 W，NewSchedule=0
1. while(所有的工作流都被调度)
2. W ←未调度工作流列表
3. for all $r_k \in R$ do
4. 基于节点的计算速度将 r_k 以降序插入 InitialSchedule(IS)；
5. end for
6. for all $\omega_l \in W$，$r_k \in R$ do
7. 插入所有待执行工作流 $\omega_l \in W$ 到服务队列，并以深度优先搜索方式遍历工作流；
8. for all $r_k \in R$ do
9. schedulegap←寻找 schedulegap(IS)；
10. 根据式(2)计算 schedulegap；
11. end for
12. if (schedulegap 在资源节点 $r_k \in R$ 上不存在，且在所有调度策略中存在 SchedulingPolicy，选择 SchedulingPolicy)then
13. 调用 function Backfilling()；
14. else if(存在 schedulegap 在资源节点 $r_k \in R$ 上)then
15. 调用 function IRS()；
16. end if
17. end for
18. end while

算法的具体流程描述如下所示：

步骤1 算法根据计算速度将所有资源节点以降序方式排列. 这样可以为即将完成的工作流任务尽可能地选择较低成本的资源(第3～5行).

步骤2 调度程序通过深度优先搜索，检查任务间的依赖关系，用以验证各个待调度任务的执行顺序. 当工作流被提交到调度程序时，工作流任务将会被插入服务队列中(第6～7行).

步骤3 待执行任务按照期限要求进行加权排序. 根据式(2)计算适当的调度间隔(schedulegap)，即每个待执行任务所使用的资源节点的时隙(第8～11行). schedulegap为CPU的空闲时间，当现有可用CPU的数量比在特定时间段内工作流所请求的CPU数量更多时，就会发生CPU空闲. 当出现新任务时，调度器会根据任务情况选择更好的调度位置. 对于一组工作流 $\omega_l \in W$，其截止时间为 D_l，其中 $l=1,2,\cdots,n$. 工作流调度可行的充要条件是满足 $SU(t)$. $SU(t)$被定义为在特定时刻 t 执行任务的时间利用率，可以通过下式表示：

$$SU(t)=\sum_{TI_i \in CI(t)} \frac{C_i}{T_i} \leqslant 1 \tag{2}$$

其中，TI_i 表示任务调用，$CI(t)$ 表示当前调用的集合，C_i 表示最坏情况下的计算时间(即资源释放和终止之间的时间最大值). 此外，T_i 表示工作流任务的周期，C_i/T_i 表示在执行工作流任务时花费的总处理时间比例. 计数器在运行时实时跟踪可调度资源的利用率，在每个调度间隙的初始阶段，计数器设置为零($SU=0$). 当调用新的工作流任务时，增加 C_i/T_i. 即如果 $SU+C_i/T_i \leqslant 1$，则在当前时刻加上截止期限并减去 C_i/T_i 后批准调度请求. 如果当前工作流任务调用的时间已经到达了截止时间，则会在每个可调度资源上搜索 schedulegaps.

步骤4 如果没有找到schedulegap，则调度程序使用Backfilling策略[13]来进行工作流调度(12～13行和函数1). Backfilling算法的思想是当队首任务不能立即执行时，先预留其所需的资源，在不影响队首任务开始执行时间下，执行后续任务中可立即执行的任务[14]，这样可以减少闲置CPU数量，提高资源利用率.

在Backfilling调度策略中，工作流将按照截止时间进行排序，并按顺序放置在调度队列中. 如果由于缺少所需的资源而无法调度第一个工作流，则调度程序将根据正在运行的工作流来计算首个工作流的最早启动时间. 随后，为此工作流做出预定，一旦所需要的资源可用，就对预定的工作流执行调度. 如果某一个在队列后面未完成的工作流可以在不影响其他工作流执行的前提下按时完成，则将其在队列中向前移动并提前执行. 因此，为了实现更高的资源利用率，理想的资源用适当的工作流回填.

函数1 Backfilling()

1. 根据工作流的截止时间对其进行排序.
2. for all $\omega_l \in W$, $r_k \in R$
3. 扫描工作流的时空分布图，找到插入点工作流 ω_0；
4. 计算闲置资源数；
5. end for
6. for all 除了 ω_0 之外的 $\omega_l \in W$
7. if(闲置资源数＞该 ω_l 所需资源数 && 该 ω_l 在 ω_0 之前结束)
8. 通过FirstFit策略回填该工作流；
9. end if
10. end for

步骤5 在发现schedulegap的情况下，工作流调度器执行提出的空闲资源调度(IRS)策略(参见第14～15行和函数2). 定期检查所有合适的资源节点，确认在其调度过程中是否存在可以用于新工作流任务执行的调度间隙. 如果任务队列不为空，并且云资源池中存在空闲资源，则工作流调度程序将尝试为资源池中的工作流任务寻找合适的资源. 函数2中的权重(tw)值表示所有新工作流任务所分配权重的最大值，根据式(3)～(5)，通过权重可以设置合适的CPU间隙. tw 定义如下：

$$tw = w_{\text{makespan}} + w_{\text{deadline}} \tag{3}$$

其中：

$$w_{\text{makespan}} = \frac{(ms_{\text{old}} - ms_{\text{new}})}{ms_{\text{old}}} \tag{4}$$

$$w_{\text{deadline}} = \frac{(nd_{\text{new}} - nd_{\text{old}})}{nd_{\text{old}}} \tag{5}$$

其中，ms_{old}和ms_{new}分别表示当前调度任务的完成时间以及新调度任务的完成时间. 此外，nd_{old}和nd_{new}分别表示在任务分配之前和之后在该执行期限内的任务数量. 本文所提出的算法搜索每个可用资源上的所有空隙并选择最佳空隙，如果 $tw>0$，则将当前映射作为最佳调度. 如果在调度开始时没有合适的调度策略，则算法将搜索当前任务 τ_i 和下一个任务 τ_{i+1}之间的间隙. 如果测试了所有的调度位置，还没有找到更好的性能优化位置，则工作流返回到初始位置. 如果 w_{deadline}的值为正，则说明 NewSchedule 工作流中被延迟的任务数量较少.

函数 2　IRS()

1. NewSchedule←InitialSchedule；
2. if ($tw>0$) then
3. for all $\omega_l \in W$，$r_k \in R$ do
4. NewSchedule←工作流任务，找 schedulegap 估计调度位置，即：一个虚拟机已经开始运行；
5. end for
6. else
7. 将工作流移出调度队列；
8. end if

3　实验及分析

3.1　实验配置

使用 CloudSim 云模拟器构建云计算场景，全面评估本文提出的调度策略的性能. CloudSim 可以在单个计算节点上模拟大型云计算的工作环境，包括服务代理、资源配置、数据中心和分配策略等. 根据云服务提供商 Amazon EC2[15] 的实例来构建一个云计算环境，虚拟机配置如表 1 所示. 其中，一个计算单元(CU)计算能力定义为一个主频为 1.0～1.2 GHz 的 Intel Xeon 处理器能力. 每个 CU 的性能峰值可达每秒计算 4.4G 条浮点指令(4.4GFLOPS).

表 1　虚拟机类型及参数

类型	核心数	CU 数	RAM/GB	存储/GB
M1.small	1	1	1.7	1×160
M1.medium	1	2	3.75	1×410
M1.large	2	4	7.5	2×420
M1.xlarge	4	8	15	2×840

截止期限范围通过识别执行单个工作流所需的最小时间来计算. 令 $D_{\min}$表示截止期限范围的最小值，且 $D_{\min}=\min CP(\omega_i)$，$CP(\omega_i)$表示具有最短关键路径的工作流 ω_i 在配备最佳可用资源时的执行时间. 同时，令 $D_{\max}$表示截止期限范围的最大值，为了构建不同的截止期限，随机设定每个工作流的 $D_{\max}$为 2CP、3CP 或 4CP. 其他工作流参数如表 2 所示.

表 2　工作流参数

参　数	值	参　数	值
工作流数量	1～20	带宽	100～1 000 Mbit/s
每个工作流的节点数	3 000	波特率	10 000 Kbit/s

3.2 性能指标

本文定义了用于评估工作流调度策略效率和鲁棒性的性能指标，描述如下所示：

工作流完成时间：完成时间为工作流全部执行完毕所需的时间. 有效的调度策略的主要设计目标是最小化工作流完成时间，提高资源利用率，从而带来更高的吞吐量. 对于工作流中的每个任务，工作时间是指从提交到完成任务所用的时间，即由 $ms=SCT$ 表示. 工作流完成时间 C_W 是指该工作流下所有任务的最大完成时间，

$$C_W=\max_{\tau\in\omega_l}\{SCT(\tau_i)\} \tag{6}$$

因此，完成时间的优化率(OMS)可表示为

$$OMS=\frac{MMS-MS_{\min}}{MMS} \tag{7}$$

其中，MMS 表示平均完成时间，$MS_{\min}$表示最小完成时间.

完成时间标准差(MSD)：这一性能指标用于衡量任务调度策略的鲁棒性. MSD 值越小表明系统越稳定，表达式如下所示：

$$\sigma_m=\sqrt{\frac{1}{N-1}\sum_{a=1}^{N}(X_a-\overline{X})^2} \tag{8}$$

其中，σ_m，X_a，N 和 $\overline{X}$ 分别表示样本完成时间的标准差、样本完成时间，样品数量和完成时间平均值.

完成时间的偏度：这一性能指标测量了完成时间分布的对称性，可以通过 *Fisher-Pearson* 标准化矩系数来计算，表达式如下所示：

$$\gamma(X)=\frac{N}{(N-1)(N-2)}\sum_{a=1}^{N}\left(\frac{X_a-\overline{X}}{\sigma_m}\right)^3 \tag{9}$$

偏度值 γ 为负值表明尾部可能处于完成时间分布的左侧，如果偏度值为正值，则意味尾部处于完成时间分布的右侧.

任务执行延迟：如果任务 τ_i 的执行时间超过了其截止时间，则会执行一个带有权重的延迟惩罚. 最小化平均迟延是调度任务的主要目标，迟延的表达式如下所示：

$$td_i=\sum_{i=1}^{N}[\max(C_i-d_i),0]/v \tag{10}$$

其中，$C_i-d_i\geqslant 0$. 最大迟延为

$$td_i^{\max}=\max_{i=1,\cdots,v}[\max(C_i-d_i,0)] \tag{11}$$

其中，C_i，d_i 和 v 分别表示任务 τ_i 的完成时间、到期时间(即任务到达时间与总处理时间之和)以及工作流中的任务数量.

调度执行时间：此度量衡量工作流调度程序执行工作流任务调度所花费的总时间.

资源利用率：这一度量定义为使用资源的时间百分比，在满足工作流应用程序的最终期限下，需要提高资源利用率. 平均资源利用率表达式如下所示：

$$\overline{RU}=\frac{\sum_{i=0}^{n-1}(RU_{t_i}*(t_{i+1}-t_i))}{t_n} \tag{12}$$

$$RU_i=\frac{R_{\text{act}}}{\min(R_{\text{ava}},R_{\text{req}})} \tag{13}$$

其中，n，R_{act}，R_{ava}和 R_{req}分别表示工作流的长度、使用资源数量、可用资源数量和工作流执行所需的资源.

3.3 结果分析

在上述仿真环境中，将本文提出的方法与传统的 FCFS 策略、SPT 策略和文献[10]提出的基于 PSO 的调度策略进行比较，在工作流完成时间、调度执行时间、资源使用率以及总延迟性能方面进行了分析.

3.3.1 工作流完成时间

比较不同调度策略下的工作流完成时间，平均完成时间如图 3 所示. 可以看出，随着工作流数量的增

加，完成时间会有所增加. 其中，本文所提方案在减少完成时间方面效果最好. 本文所提方案比 FCFS 策略的完成时间平均缩短了 51.3%，比 SPT 策略缩短了 17.4%. 这是因为本文所提方案能够更加有效地利用资源，从而使用更少的时间来完成更多的任务. 另外，本文所提方案与 PSO 调度策略的结果相近，这是因为 PSO 算法是一种高性能的智能优化算法，在一定邻域内能够收敛到最优解. 但由于 PSO 是一种迭代优化的复杂算法，所以计算量较高.

完成时间的偏度衡量了完成时间数据集的不对称程度，完成时间的标准差能够表示调度策略的稳定性. 表 3 统计了平均完成时间、完成时间的偏度和标准差结果，可以看到，完成时间的分布是不对称的，工作负载往往是具有偏差的. 另外，本文所提方案的平均完成时间最短，而 FCFS 策略的平均完成时间最长. 本文所提方案的完成时间标准差也最小，说明了调度策略具有较好的稳定性和鲁棒性.

表 3　不同调度策略的完成时间统计

调度策略	平均完成时间/s	完成时间的偏度	完成时间的标准差
本文所提方案	23 358.6	0.472	583.6
FCFS	24 521.7	−0.136	789.4
SPT	23 563.4	0.463	644.7
PSO	23 401.3	0.487	596.1

3.3.2　调度执行时间

该实验对调度策略的执行时间进行了评估. 在工作流数量分别为 1～20 个情况下，利用各种调度策略对其执行 5 次调度，计算平均调度执行时间，结果如图 4 所示. 可以看出，执行时间随工作流数量的增加呈线性增长. 其中，本文所提方案的平均执行时间明显小于传统 FCFS 策略和 SPT 策略，随着工作流数量的增加，其优势也越来越明显. 本文所提方案比 FCFS 策略执行时间减少了 78.4%，比 SPT 策略减少了约 46.8%. 另外，PSO 调度策略的执行时间也较长，在工作流总数为 20 时，PSO 算法需要迭代将近 150 次才能制定出一个最优调度策略，约花费了 40s 的时间，所以不适用于大规模实时任务调度中.

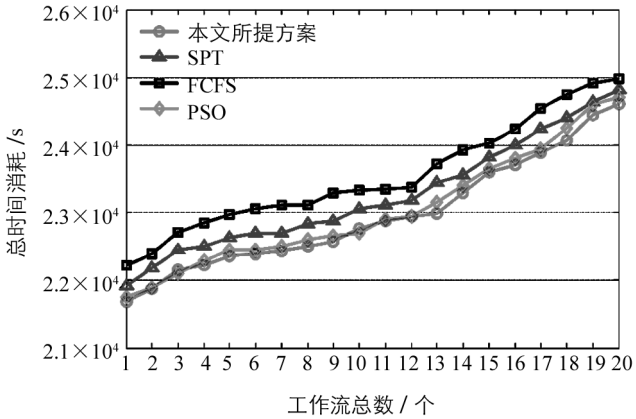

图 3　工作流完成时间

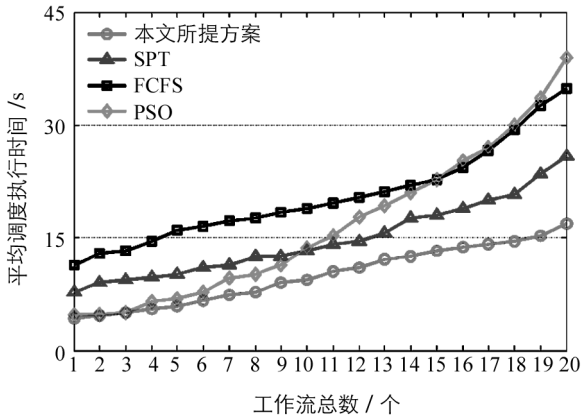

图 4　调度策略的平均执行时间

3.3.3　资源使用率

当资源节点不支持任务所需的所有硬件/软件要求时，资源节点会处于空闲状态. 图 5 描绘了不同调度间隔下，每种调度策略的资源利用率，可以看出，本文所提方案展现出更好的资源利用率. 本文所提方案的资源利用率比 FCFS 策略提高了 17.8%，比 SPT 策略提高了 8.2%. 这是因为本文所提方案总是能够保持系统资源在使用状态，并能有效调度系统的计算资源. 此外，本文所提方案通过提前执行非计划任务来利用空闲资源，以减少成本开销. 另外，由于 PSO 算法能够搜索到较优的调度策略，使其获得了较高的资源使用率，与提出方案的性能类似.

3.3.4　任务执行延迟

平均延迟取决于在可用机器上执行的工作流数量，在工作负载逐渐增加的情况下，延迟将会增加. 这表明，总的延迟程度很大程度上取决于 VM 可用实例的数量和工作负载的强度. 随着负载的增加，延

迟的工作流数变得更多，并且延迟时间可能会变得更长. 图6展示了不同调度策略的平均延迟，可以看出，与FCFS和SPT调度策略相比，本文所提方案的平均延迟值较小. 本文所提方案比FCFS策略的平均延迟降低了约50.7%，比SPT策略降低了约21.6%，这是因为本文所提方案具有更高的资源利用率，且能够更有效地利用调度间隙，所以能够在更短的时间内完成更多的任务，表现出最小的延迟时间. 此外，与PSO调度相比，本文所提方案仍然稍具优势. 这是因为当工作流总数较大时，PSO需要大量迭代才能寻找到最优值，会出现迭代次数超出设定的限制，有时会给出次优的调度策略，这在一定程度上影响了调度性能.

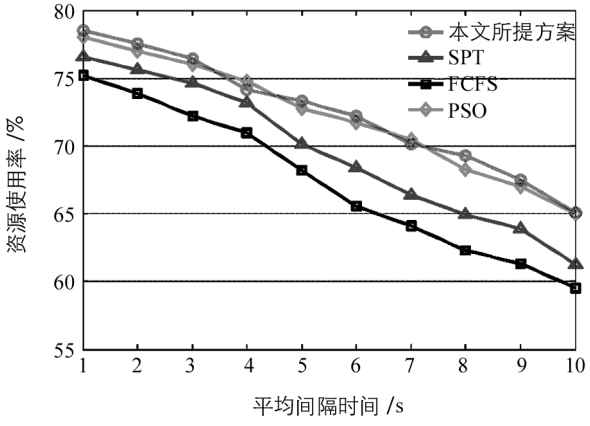

图5 资源使用率

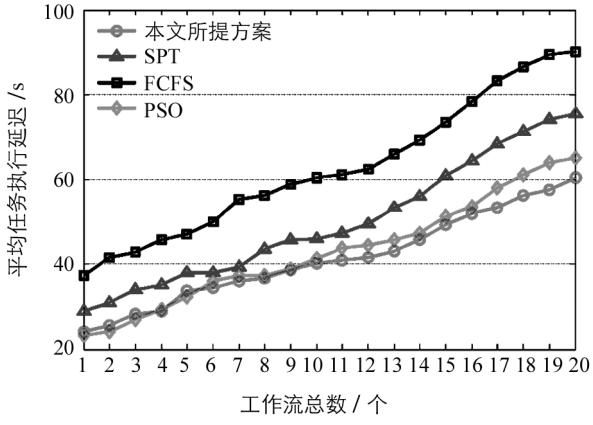

图6 平均任务执行延迟

综上实验表明，本文所提调度策略在各项性能指标上都优于传统的调度策略，且与目前先进的PSO调度策略性能相当. 但在调度执行时间方面比PSO策略更具明显的优势，因此更适合云工作流调度场景应用. 这是因为本文采用了IRS调度策略，并结合了Backfilling策略，IRS充分利用了已调度任务之间的间隙，用于调度其他任务，从而减少了整体的执行周期. 此外，IRS的执行时间比其他传统策略的执行时间要短. Backfilling策略引入了任务回填操作从而减少了资源碎片化的程度.

4 结束语

云计算已经被广泛认为是执行计算和数据密集型工作流的基本计算范例. 本文提出了一种新颖的多租户云计算环境中的工作流调度策略，即空闲资源调度(IRS)策略，充分利用已调度任务之间的间隙来调度其他任务，从而减少整体的执行周期，并结合了Backfilling策略以缩短任务等待时间. 通过与现有传统方法进行比较，证明了本文所提方案的有效性.

参考文献：

[1] 贾冬青，高永芹，陈玉芳. 基于云存储的内容分发技术研究 [J]. 西南师范大学学报(自然科学版)，2016，41(6)：111－118.

[2] 王 准，苏顺开. 多租户云计算中基于ILP模型的虚拟机放置策略 [J]. 湘潭大学(自然科学学报)，2016，38(4)：71－75.

[3] 邓见光，赵跃龙，袁华强. 一种多QoS目标约束的云计算任务调度策略 [J]. 计算机应用研究，2016，33(8)：2479－2482.

[4] 孙景昊，邓庆绪，孟亚坤. GPU上两阶段负载调度问题的建模与近似算法 [J]. 软件学报，2014，25(2)：298－313.

[5] BOBELIN L，MARTINEAU P，ZHAO D，et al. Shortest Processing Time First Algorithm for Hadoop [C]//Cyber Security and Cloud Computing (CSCloud)，2016 IEEE 3rd International Conference on. Beijing，China：IEEE，2016：119－123.

[6] 周 凯. 高性能计算中作业调度技术与集群管理系统的研究 [D]. 镇江：江苏科技大学，2015.

[7] 张 伟，刘三阳. 动态局部搜索差分进化算法 [J]. 西南大学学报(自然科学版)，2015，37(3)：93－98.

[8] BESSAI K，YOUCEF S，OULAMARA A，et al. Bi-Criteria Workflow Tasks Allocation and Scheduling in Cloud Com-

puting Environments [C]//Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. Honolulu, USA: IEEE, 2012: 638－645.

[9] HE T, CHEN S, KIM H, et al. Scheduling Parallel Tasks onto Opportunistically Available Cloud Resources [C]//Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. Honolulu, USA: IEEE, 2012: 180－187.

[10] WU Z, NI Z, GU L, et al. A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling [C]//Computational Intelligence and Security (CIS), 2010 International Conference on. Nanning, China: IEEE, 2010: 184－188.

[11] 张　亮，张曦煌. 一种面向云计算虚拟机资源拓扑结构的任务调度 [J]. 计算机应用研究，2015，32(12)：3738－3741.

[12] 王　波. 基于遗传算法的集群虚拟机资源调度研究 [J]. 西南师范大学学报(自然科学版)，2016，41(3)：107－111.

[13] SURESH A, VIJAYAKARTHICK P. Improving Scheduling of Backfill Algorithms Using Balanced Spiral Method for Cloud Metascheduler [C]//Recent Trends in Information Technology (ICRTIT), 2011 International Conference on. Chennai, Tamil Nadu, India: IEEE, 2011: 624－627.

[14] Sullivan D, Newman A. Extraction and Backfill Scheduling in a Complex Underground Mine [J]. Interfaces, 2014, 44(2): 204－221.

[15] AGMON BEN-YEHUDA O, BEN-YEHUDA M, SCHUSTER A, et al. Deconstructing Amazon EC2 Spot Instance Pricing [J]. ACM Transactions on Economics and Computation, 2013, 1(3): 1－20.

A Cloud Workflow Scheduling Scheme Based on Combining Backfilling and Idle Resource Scheduling

TAN Hai-zhong[1], ZHAO Li[2]

1. *Information Technology Department, Guangzhou Institute of Technology, Guangzhou 510925, China;*
2. *School of Software, Shanxi University, Taiyuan 030006, China*

Abstract: A fast and effective scheduling scheme is proposed in this paper for solving workflow scheduling problems in cloud computing. First, all resource nodes are arranged in a descending order according to the calculation speed. Then, the scheduler evaluates the dependencies between the tasks by depth-first search (DFS) and weights the tasks according to the deadline. After that, the time slot of each resource to be used for the task is calculated. If the currently available resource does not meet the current task, the backfilling policy is used to reserve the resources required for the task and skip to the next task. If the current resource satisfies the current task, the proposed idle resource scheduling (IRS) policy is performed to schedule the idle resource to perform the task. Simulation results show that the proposed scheduling strategy has excellent performance in terms of task completion time, task execution delay and resource utilization.

Key words: cloud computing; workflow scheduling; backfilling mechanism; idle resource scheduling; task execution delay

责任编辑　崔玉洁