·先进计算与数据处理· 文章编号:1000-3428(2019)09-0040-09 文献标志码:A 中图分类号:TP393

云环境下基于预算分配的科学工作流调度研究

张继炎,郑汉垣

(龙岩学院 数学与信息工程学院,福建 龙岩 364012)

摘 要: 云环境下的科学工作流部署不同于传统的独立任务调度,需同步考虑调度代价与时间问题。为此,提出基于预算分配的科学工作流调度方法,将工作流任务与虚拟机资源间的映射求解分为预算分配和资源提供与调度 2 个阶段。为优化预算使用,设计基于快优先的预算分配算法(FFTD)和基于慢优先的预算分配算法,实现预算在各任务间的子分配。基于任务最早完成时间的降序排列进行任务选择,在虚拟机可重用的情况下根据单个任务的子预算进行资源分配,保证工作流任务的顺利调度。引入 5 种常规类型的科学工作流进行实验,测试算法在不同类型工作流结构和不同预算约束下的性能,结果表明,FFTD 算法在 72%、88%、84% 的实验场景中相比 BDT-AI 算法具有更高的虚拟机资源利用率、预算约束满足率以及更短的调度时间,综合性能更优。

关键词: 云计算; 科学工作流; 预算分配; 工作流调度; 资源调度

开放科学(资源服务)标志码(OSID):

中文引用格式: 张继炎,郑汉垣. 云环境下基于预算分配的科学工作流调度研究[J]. 计算机工程,2019,45(9):40-48.

英文引用格式: ZHANG Jiyan, ZHENG Hanyuan. Research on scientific workflow scheduling based on budget allocation in cloud environment[J]. Computer Engineering, 2019, 45(9): 40-48.

Research on Scientific Workflow Scheduling Based on Budget Allocation in Cloud Environment

ZHANG Jiyan, ZHENG Hanyuan

(School of Mathematics and Information Engineering, Longyan University, Longyan, Fujian 364012, China)

【Abstract】 The scientific workflow deployment in the cloud environment is different from the traditional independent task scheduling, and the scheduling time and cost should be considered simultaneously. To address the problem, a scientific workflow scheduling method based on budget allocation is proposed. The mapping between workflow tasks and virtual machine resources is divided into two stages: budget allocation, and resource provision and scheduling. In order to optimize budget usage, a budget allocation algorithm based on fast-priority, called FFTD, and budget allocation algorithm based on slow-priority, called SFTD, are designed to achieve sub-allocation of budget among tasks. The task selection is performed based on the descending order of the earliest completion time of the task, and the resources are allocated according to the sub-budget of the single task when the virtual machine is reusable, thereby ensuring smooth scheduling of the workflow task. Five kinds of conventional types of scientific workflows are introduced to test the performance of the algorithm under different types of workflow structures and different budget constraints. The results show that the FFTD algorithm has shorter scheduling time and higher virtual machine resource utilization and satisfaction rate of budget constraints than the BDT-AI algorithm in 72%, 88% and 84% experimental scenarios, and the overall performance is better.

【Key words】 cloud computing; scientific workflow; budget allocation; workflow scheduling; resource scheduling

DOI: 10.19678/j.issn.1000-3428.0052687

0 概述

在科学实验中通常涉及收集海量数据进行实验仿真,这些科学实验通常表达为工作流格式(即由相互关联的多重计算任务组成)。在科学工作流中,任务代表不同的计算需求,而任务间的有序连接则代表数据间的关联性,它们需要大量的资源在合理的时间内进行数据处理,云计算平台这种分布式计算

基金项目: 福建省自然科学基金(2015J01587);龙岩学院产学研创新基金(LC2016005)。

作者简介: 张继炎(1978—),男,实验师、硕士,主研方向为物联网、云计算;郑汉垣,教授、博士。

收稿日期: 2018-09-17 **修回日期:** 2018-10-29 **E-mail:** 3455393742@qq.com

环境是当前进行工作流任务处理与部署的常用平台。云计算通过不同配置(虚拟机类型)的虚拟机(Virtual Machine,VM)租用提供计算资源,虚拟机可以按需租用,并根据提供者制定的账单定期向用户索取费用,如:Amazon EC2 中的虚拟机以 1 h 作为付费的账单周期,换言之,即使虚拟机使用为 1.5 h,也将以 2 h 的账单付费。这种灵活性和弹性使云环境成为执行科学工作流的理想平台,而虚拟机数量和类型也易于调整,进而匹配工作流任务的特征和数量。然而,高灵活性和易于扩展资源数量的能力会导致 2 种服务质量需求间的冲突:时间和代价,即虚拟机处理能力越强,代价也越高。因此,在工作流调度时对虚拟机类型和数量进行决策的过程中必须考虑时间与代价的均衡问题[1]。在已有研究中,调度算法多数集中于在截止期限内完成工作流的情况下最小化执行代价。本文工作主要集中于优化资源的利用,在满足预算约束的同时最小化工作流的调度时间。

1　相关研究

云环境中的科学工作流部署已有广泛研究,多数算法集中于 2 种 QoS 参数:时间和代价,该类算法的主要目标是满足截止期限的约束并最小化资源的租用代价,典型工作如:文献[2]提出一种满足期限约束的代价最小化粒子群优化算法;文献[3]提出一种局部关键路径代价优化算法;文献[4]提出一种多层次代价优化调度算法;文献[5]提出一种期限分布比例算法,优化期限约束下的调度代价;文献[6]提出一种个体向量最优化学习策略进行云服务代价的优化;文献[7]利用迭代复合局部搜索方法,求解期限约束的工作流调度成本最小化问题。另外有一部分工作集中于满足预算的约束最小化工作流的调度长度/时间,如局部关键路径预算均衡算法 PCP-B^2[8]。该算法将工作流结构划分为多条管道式的局部关键路径,并在最大化预算利用的情况下寻找最优的执行资源类型。PCP-B^{2}算法以一个时间单位作为资源的定价模型,这也是账单周期的最通用模型。关键贪婪算法[9]利用类似的策略,在剩余预算仍可用的情况下通过迭代提取和初始化调度方案优先对能力较强的虚拟机进行利用,实现效率的优化。相同优化目标的算法还有 PSO 算法[10]和遗传算法[11],均提出一种静态预算分配方案,完成工作流调度时间的最小化操作。这些算法均需要依赖主动式元启发式方法计算近似最优的调度解。本文设计的自适应动态部署算法,基于系统运行时的状态对调度部署与资源提供做出相应决策。

BAGS 算法[12]是一种预算约束算法。该算法将工作流划分为处于同一工作流层次上的包任务,并基于在线预算分配策略动态执行资源提供与任务部署,使就绪任务能够尽早执行。相比本文算法,BAGS 算法利用一种细粒度的资源账单计费时间(1 min),而该时间小于任务的平均执行时间。BDT 算法[13]利用类似的机制将任务合并至相同的工作流层次上,预算在每个层次上进行分配,并将剩余预算移至下一层次,该算法以 1 h 作为账单周期,而忽略了虚拟机的性能变化对部署优化结果带来的影响。此外,该算法以任务是否就绪作为任务的调度起点,而本文部署方式为只要所有父任务完成且所有数据已输入,无论所在层次,该任务可以立即执行。

2　系统模型

将工作流建模为有向无循环图(Directed Acyclic Graph,DAG),由不存在循环结构的有向边构成。工作流 W 由任务集合 $T=(t_1,t_2,\cdots,t_n)$ 和有向边集合 $E=(e_{12},e_{13},\cdots,e_{mn})$ 组成,每条边 e_{ij} 代表任务的父节点 t_i 与任务的子节点 t_j 间的数据依赖性,即节点 t_j 需要在节点 t_i 完成后才开始执行。令 S_t 表示任务的大小,以百万指令数(Millions of Instructions,MI)度量。

虚拟机(Virtual Machine,VM)以按需计价模型租用,即以账单周期 bp 索价,任意账单周期内的部分使用将以最邻近的账单周期量计价。云环境下存在若干虚拟机类型 vmt_i,它们拥有不同的处理能力 PC_{vmt} 和不同账单周期下的使用代价 c_{vmt}。处理能力 PC_{vmt} 以每秒百万指令数 MIPS 度量,这与任务大小 S_t 的单位对应。假设虚拟机性能随时间变化,PC_{vmt} 值为已知性能的最大值。一个任务的运行时间定义为 RT^t_{vmt},可利用式(1)计算得到:

$$RT^t_{vmt}=S_t/PC_{vmt} \tag{1}$$

考虑一种全局共享存储系统进行工作流任务间的数据共享,每个任务从全局共享存储中得到的输入数据为 D^t_{in},产生的输出数据为 D^t_{out}。每一台虚拟机的带宽为 B_{vmt},全局存储系统的数据读取和写入速率分别为 GS_{read} 和 GS_{write}。带宽与 I/O 速率基于运行时间 t 下事务数量 Tr 的改变而改变,即:

$$B_{vmt}(t)=B_{vmt}/Tr_t \tag{2}$$

$$GS_{\text{read}}(t)=GS_{\text{read}}/Tr_t^{\text{read}} \tag{3}$$

$$GS_{\text{write}}(t)=GS_{\text{write}}/Tr_t^{\text{write}} \tag{4}$$

从全局共享存储系统输入数据至虚拟机的时间为:

$$T^{D^t_{\text{in}}}_{vmt}=D^t_{\text{in}}/B_{vmt}+D^t_{\text{in}}/GS_{\text{read}} \tag{5}$$

数据输出时间为:

$$T^{D^t_{\text{out}}}_{vmt}=D^t_{\text{out}}/B_{vmt}+D^t_{\text{out}}/GS_{\text{write}} \tag{6}$$

假设全局共享存储系统与虚拟机位于同一区域内,则区域内的数据传输代价为 0,一个任务的总体处理时间为:

$$PT^t_{vmt}=RT^t_{vmt}+T^{D^t_{\text{in}}}_{vmt}+T^{D^t_{\text{out}}}_{vmt} \tag{7}$$

当使用类型为 vmt 和单个账单周期 pb 下价格为 c_{vmt} 的虚拟机时,任务代价包括资源准备延时 T_{pdelay} 和移除延时 T_{ddelay},即:

$$c^t_{vmt}=\lceil(PT^t_{vmt}+T_{\text{pdelay}}+T_{\text{ddelay}})/pb\rceil\times c_{vmt} \tag{8}$$

3 算法设计

基于预算分配的科学工作流调度算法由 2 个部分组成：

1）预算分配：工作流部署在费用上的预算将分配于单个个体任务上，本文设计基于快优先的预算分配算法（FFTD）和基于慢优先的预算分配算法（SFTD）。

2）资源提供与调度：基于任务的最早完成时间的降序排列进行任务选择，在虚拟机可重用的情况下根据单个任务的子预算进行资源提供。

3.1 预算分配

本质上，预算大小决定了工作流的调度与部署过程。在预算允许的情况下，选择最快资源执行任务可以降低由于性能变化（影响执行时间）导致超过虚拟机账单周期的概率。任务执行时间的推迟不仅会带来账单周期的逾期，而且会导致代价的增加。因此，粗粒度云资源租用账单周期的存在将使预算分配问题变得尤其重要。在某些情况下，基于任务的运行时间估算而不考虑账单周期下的预算分配将导致预算不充分，剩余预算不足以满足新虚拟机的提供需求。

考虑图 1 中的示例，假设云系统中有 2 种类型虚拟机 small 和 large，small 类型虚拟机的利用代价为 $1/h，large 类型虚拟机的利用代价为 $3/h。假设代价越高，虚拟机处理任务的能力越强。现有由 7 个任务组成的工作流，每个任务的运行时间估算为 $RT_A=100$ s、$RT_B=400$ s、$RT_C=400$ s、$RT_D=200$ s、$RT_E=100$ s、$RT_F=100$ s、$RT_G=100$ s，工作流总体执行预算为 $7。若基于每个任务的运行时间与总运行时间进行预算的正比例分配而忽略虚拟机的账单周期，得到各任务的预算子分配如图 1（b）所示，此时对于任务 A、E、F 和 G 的预算将出现不足。由于 A 和 E 是工作流的入口任务，因此若其子预算不足以进行资源提供，则工作流将得不到执行。

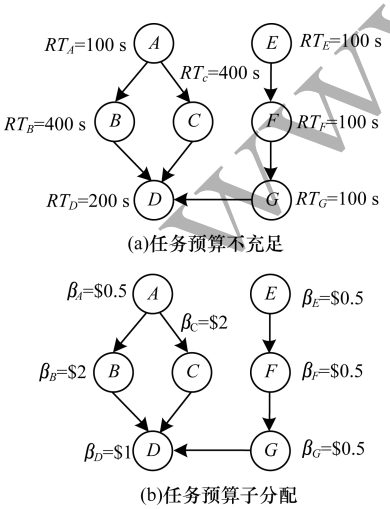

图 1 任务预算不充足及子分配示例

假设基于任务的运行时间排序进行预算分配，任务从处于工作流第一层次的入口任务开始执行，然后执行下一层次的子任务，依此类推。那么，可利用自项向下的 DTL 方法以工作流的入口任务作为起点进行层次划分（如式（9）所示），代替自底向上的 DBL 方法中以出口任务为起点进行层次划分。

$$level(t)=\begin{cases}0, Pred(t)=\varnothing\\ \max\limits_{p\in Pred(t)} level(t)+1, \text{其他}\end{cases} \tag{9}$$

其中，$Pred(t)$ 为任务 t 的父任务集合，$level(t)$ 表示任务 t 所在层次。

考虑图 1 中的示例，DTL 分配任务 A 和 E 在相同层次 $level(1)$ 中，而 DBL 则从任务 D 开始分配在层次 $level(1)$ 中。相应地，DBL 中任务 A 的层次为 $level(3)$，而任务 E 处于不同的层次 $level(4)$ 中。工作流需要从入口任务开始处理，由于需基于层次执行任务，因此将任务分配在不同层次可能导致任务执行的延时。此外，为决定层次中任务的次序，算法需要基于任务最早完成时间（EFT）的降序对任务进行排序，即：

$$EFT(t)=\begin{cases}PT^{t}_{vmt}, Pred(t)=\varnothing\\ \max\limits_{p\in Pred(t)} EFT(t)+PT^{t}_{vmt}, \text{其他}\end{cases} \tag{10}$$

此时，将图 1 工作流排序为：$A[level(1)]\rightarrow E[level(1)]\rightarrow F[level(2)]\rightarrow B[level(2)]\rightarrow C[level(2)]\rightarrow G[level(3)]\rightarrow D[level(4)]$，那么预算分配算法需要在考虑任务的估算运行时间和每种虚拟机类型代价的情况下顺序地将预算分配至每个任务。进一步地，预算分配考虑的虚拟机资源以每个账单周期索价，即分配至单个任务的子预算等于一个完整账单周期时间的代价。然而，可能存在部分任务无法得到子预算的分配。对于这部分任务，算法将延迟任务执行直到它们可重用空闲虚拟机资源为止。预算分配算法具体过程如算法 1 所示。

算法 1 预算分配算法

```
1. Procedure DISTRIBUTIONBUDGET(β,T)
2. S = task's estimated execution order
3. or each task t ∈ T do
4. allocateLevel(t,1)
5. initiateBudget(0,t)
6. for each level l do
7. T_l = set of all tasks in level l
8. sort T_l based on ascending EFT
9. put(T_l,S)
10. while β >0 do
11. t = S. poll
12. vmt = chosen VM type//FFTD or SFTD
13. allocateBudget(C^t_vmt,t)
14. β = β - C^t_vmt
```

算法 1 根据虚拟机类型选择 FFTD 和 SFTD 算法进行预算分配。FFTD 算法在工作流预算可负担的情况下选择最快虚拟机类型执行任务。由于算法将最快资源分配至最早任务,其子任务重用较快资源的概率将会增加,因此有利于在空闲虚拟机的可用性存在一定延时的情况下减少任务的处理时间。相比而言,SFTD 算法选择价格最低(性能最差)的虚拟机类型执行任务。在所有任务已分配子预算后若仍有预算剩余,算法将在剩余预算允许的情况下租用更快的虚拟机资源执行任务。SFTD 算法可确保大部分任务得到预算分配,使任务无须等待账单周期内的空闲虚拟机资源。在 2 种算法中,将资源分配至最早(就绪)任务也可确保工作流的顺利执行。

3.2　资源提供与调度

在子预算分配至每个任务后,算法将从工作流的入口任务开始处理,并基于 EFT 的降序排列将其放入优先级队列。然后,在任务的子预算允许的情况下向其提供一个最快的虚拟机资源类型。只要存在空闲虚拟机,算法寻找包含任务输入数据的虚拟机列表并对在账单周期内拥有空闲时间的虚拟机进行分配,使其以最小风险完成任务执行。风险最小是指若同时拥有的两个虚拟机在不引发一个新账单周期的情况下均无法完成任务,则算法选择代价低的虚拟机。若拥有任务输入数据的虚拟机不可用,则算法将以同样的标准分配任意空闲虚拟机。资源提供与调度过程如算法 2 所示。

算法 2　资源提供与调度算法

1. procedure SCHEDULEQUEUETASKS(q)
2. sort q by ascending EFT
3. sb = spare budget
4. while q is not empty do
5. t = q. poll
6. vm = null
7. delayFlag = false
8. if there are idle VMs then
9. VM_{idle} = set of all idle VMs
10. VM_{input} idle = set of vm ∈ VM_{idle} that have t's input data
11. vm = vm ∈ VM_{idle}^{input} that can finish t with minimum risk of incurring a new billing period
12. if vm = null then
13. vm = vm ∈ VM_{idle} that can finish t with minimum risk of incurring a new billing period
14. else
15. vmt = cheapest VM type
16. if t. budget < C_{vmt}^{t} then
17. delayFlag = true
18. if delayFlag = false then
19. vmt = fastest VM type within t. budget
20. if there are faster VM type than vmt AND sb is enough then
21. vmt = leaseFasterVMT( )
22. vm = provisionVM(vmt)
23. scheduleTask(t,vm)

在完成一个任务后,算法更新预算分配,调整可用预算。同时,保留重用空闲虚拟机的任务,未使用的子预算作为剩余预算。该剩余预算可用于预算分配更新,也可在资源提供与调度阶段租用更快的虚拟机资源。预算分配更新过程算法 3 所示。

算法 3　预算分配更新算法

1. procedure UPDATEBUDGET(T)
2. t_f = completed task
3. T_c = set of t ∈ T that are children of t_f
4. β_c = total sum of t. budget, where t ∈ T_c
5. T_u = set of unscheduled t ∈ (T − T_c)
6. β_u = total sum of t. budget, where t ∈ T_u
7. sb = spare budget
8. if $C_{vmt}^{tf} \leqslant (t_f.\ budget + sb)$ then
9. $sb = (t_f.\ budget + sb) - C_{vmt}^{tf}$
10. else
11. $debt = C_{vmt}^{tf} - (t_f.\ budget + sb)$
12. $\beta_c = \beta_c - debt$
13. DISTRIBUTIONBUDGET(β_c, T_c)
14. if $\beta_c < 0$ then
15. $\beta_u = \beta_u + \beta_c$
16. DISTRIBUTIONBUDGET(β_u, T_u)

3.3　实例分析

本节以图 1 所示工作流结构示例详细阐述 2 种预算分配方法下工作流部署算法的思想。假设每种虚拟机类型的 PC_{vmt} 与 c_{vmt} 为线性正比例关系,FFTD 和 SFTD 得到的预算分配结果分别如图 2(a)、图 2(b)所示,对应的任务部署结果如图 3(a)、图 3(b)所示。

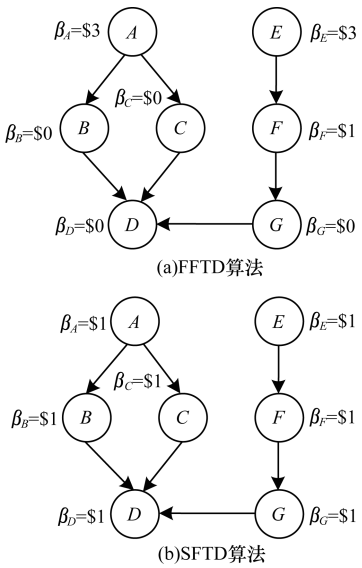

图 2　2 种算法下的预算分配结果

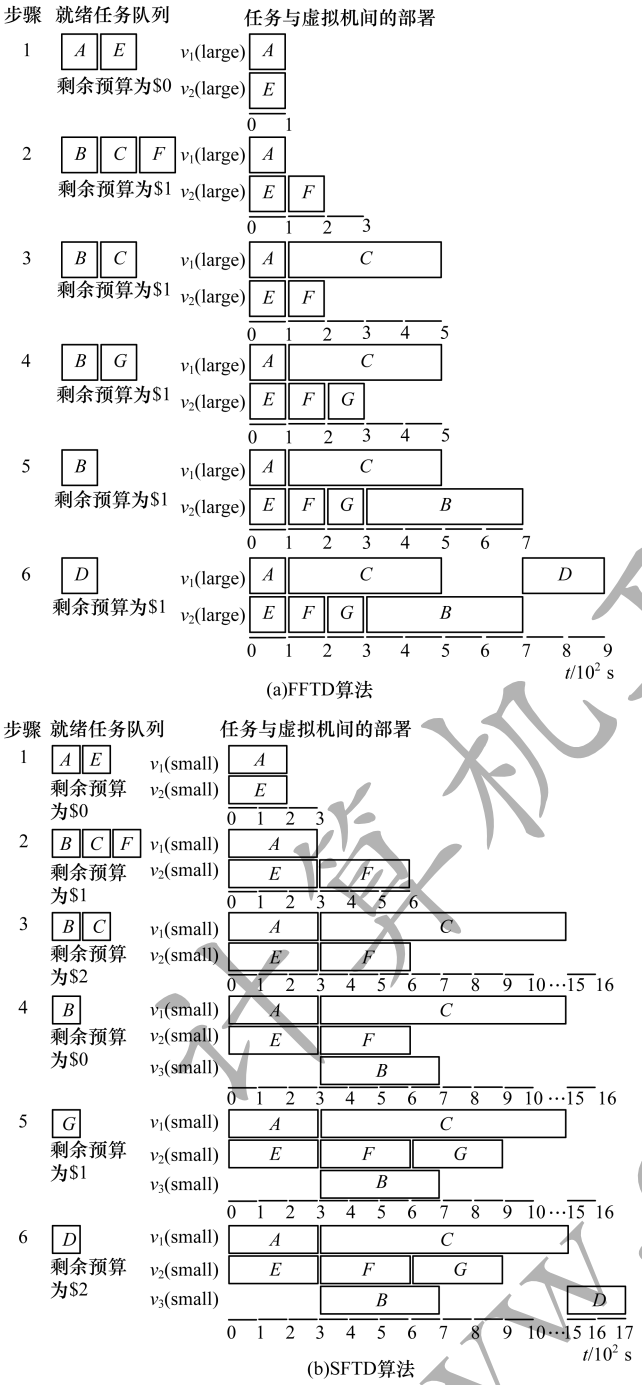

图 3 任务部署示例

可以看出,在步骤 1 ~步骤 3 中,除了虚拟机类型与剩余预算不同,FFTD 和 SFTD 的部署结果是相似的,由于分配至入口任务的子预算不同导致所选虚拟机类型不同。在 FFTD 的步骤 4 中,算法延迟了任务 B 的执行,由于其子预算为 \$0。任务 F 完成后,任务 G 为就绪任务,算法优先执行 G,由于 G 相比 B 具有更小的 EFT。然后,任务 G 重用 v_2(large),而 B 在 G 完成后也重用了 v_2(large)。在 B 和 C 完成后,D 为就绪任务,D 重用 v_1(large)。最后,FFTD 花费代价为 \$6,剩余预算为 \$1,调度时间为 900 s。在 SFTD 的步骤 4 中,算法调度任务 B,并基于其子预算提供一个新的虚拟机类型,原因在于此时拥有足够的剩余预算为 B 租用更快的虚拟机类型。F 完成后,G 为就绪任务,可重用 v_2(small)。然后,B 和 C 完成后,D 为就绪任务,并可重用 v_3(large)。最后,SFTD 花费代价为 \$5,剩余预算为 \$2,调度时间为 1 700 s。

算法由预算分配和资源提供与调度 2 个步骤组成,假设以工作流 $G(T,E)$ 作为输入,该工作流拥有 n 个任务和 e 条边。由于 G 是有向图,因此其最大边数为 $((n-1)(n-2))/2 \approx O(n^2)$。预算分配过程中首先需要对任务划分层次并进行预算初始分配,这需要遍历所有工作流任务及其连接边,该步骤的时间复杂度为 $O(n+e) \approx O(n^2)$;然后对各层次中的任务按 EFT 进行降序排列,时间复杂度为 $O(n)$;最后按 FFTD 或 SFTD 进行预算重分配,时间复杂度为 $O(n)$;资源提供与调度过程具有两层嵌套循环,外层循环遍历 n 个任务,内层循环遍历 m 个虚拟机资源,其时间复杂度为 $O(nm)$;预算更新步骤的时间复杂度为 $O(nm)$。通常,$m < n$,假设资源数量最多为 n,则资源提供与调度过程的时间复杂度为 $O(n^2)$。综合以上分析,算法最差时间复杂度为 $O(n^2)$。

4 性能测试

为评估算法性能,利用不同科学领域中的合成工作流结构进行实验仿真。Montage(天文学)工作流可以利用输入图像集产生天空色彩影像,其任务多为 I/O 密集型,且不要求过高的 CPU 处理能力。LIGO(天体物理学)工作流用于发现引力波,基本由高内存需求的 CPU 密集型任务构成。SIPHT(生物信息学)工作流用于自动化搜索生物领域中的基因编码,多数任务具有较高的 CPU 和较低的 I/O 占用。Epigenomics(生物学)工作流用于生物学中的基因测序,多数为 CPU 密集型任务。CyberShake(地震学)用于产生和合成地震时的震动波纹图,主要由具有大内存和 CPU 需求的数据密集型任务构成。这5 种工作流在任务依赖性和任务运行时间上具有不同的结构和特征,其产生方式和具体描述可参见文献[14]。

本文实验应用不同的预算间隔,假设最小预算设置为将工作流所有任务运行于代价最低的虚拟机类型上得到的代价。基于最小预算,通过下式定义10 种不

同的预算间隔:$budget = \alpha \times min_{budget}$,$0 < \alpha < 11$,其中,$\alpha = 1$ 对应于最小的预算,$\alpha = 10$ 对应于最大的预算。利用 CloudSim 仿真平台[15]建模 IaaS 云环境。该环境配置 1 个数据中心和 4 种类型的虚拟机。虚拟机的具体配置参数如表 1 所示。虚拟机的 CPU 性能与价格对应于 Amazon EC2 所提供的 c4 类型实例,2 种参数间存在线性关系。所有虚拟机类型的账单周期设置为 1 h,延时设置为 97 s[16]。

表 1　虚拟机类型和价格

类型	内存/GB	vCPU 数量	ECU 数量	价格/($ · h^{-1})
small	3.75	2	7	1
medium	7.50	4	14	2
large	15.00	8	28	4
xlarge	30.00	16	56	8

4.1　算法性能

本节评估算法在代价和调度时间方面的性能。代价性能通过调度代价/预算来评估算法在满足预算约束上的能力,因此若该比值大于 1,则表明代价大于预算;若该比值等于 1,则表明代价等于预算;若该比值小于 1,则表明代价小于预算。此外,每个预算间隔下的实验将重复 100 次并取其均值结果。

基准算法选取为 BDT 算法[13]。该算法是一种动态的基于层次的预算分配算法,优化目标与本文算法相似。BDT 算法基于最早开始时间(Earliest Start Time,EST)进行任务调度,并引入代价/时间均衡因子(TCTF)计算在相应类型的虚拟机上进行任务执行时的均衡率,然后在资源提供阶段选择具有最大均衡值的资源进行任务调度。BDT 利用可用预算执行同一层次中的所有任务,并将剩余预算留至下一层次。若预算不足以租用新的虚拟机资源,则算法会推迟任务执行,并迫使任务尽可能重用空闲虚拟机。BDT 算法引入多种预算分配机制,其中 All-In 策略的预算分配性能最优,因此以 BDT-AI(All-In)作为基准算法。

实验 1　预算约束满足率评估。图 4 给出了不同工作流类型和预算下的代价/预算比值。对于 Montage 工作流(见图 4(a)),SFTD 在最严格预算间隔中的性能优于其他算法,原因在于该算法在资源提供选择阶段选择价格最低的虚拟机类型。在剩余预算的情况下,算法性能相同,具有相同的代价/预算比值,这与所选的粗粒度(1 h)账单周期相关。从 β_2 开始所有算法得到的调度长度与账单周期相隔较远,这表明虚拟机性能的变化未完全影响调度代价。尽管 FFTD 和 BDT-AI 的预算约束性能接近,但 FFTD 的调度时间仍优于 BDT-AI。

图 4(b)给出了 LIGO 工作流的实验结果,可以看出由于第一个预算间隔过于紧密,导致所有算法均无法满足预算约束。在其他情况下,SFTD 算法得到了更低的代价/预算比值,而另外 2 种算法也满足相应的预算约束。

图 4(c)给出了 SIPHT 工作流的实验结果。第一个预算间隔下所有算法均出现预算约束违例,而 FFTD 的违例边界最小,说明超出预算部分最少。此外,从调度代价与预算间的差值可以看出,虚拟机性能变化会影响算法满足预算的能力。综合而言,FFTD 算法在 SIPHT 工作流中的预算约束性能优于另外 2 种算法。

图 4(d)显示了 CyberShake 工作流的实验结果。可以看出,基本上在每种情况下算法均无法满足预算约束,仅有 BDT-AI 算法在最后 3 种预算间隔下可以满足预算,原因在于:CyberShake 工作流为数据密集型工作流,涉及大量 I/O 活动,这会导致任务完成时间变长,代价相应增加。同时,从 SFTD 的结果可以看出,代价/预算比值会随着虚拟机数量的增加而增加,导致巨大的 I/O 数据转换及更大的开销。尽管算法在估算任务处理时间时均考虑了数据传输时间,但仍可能存在无法预测的开销所带来的网络流量拥塞。此外,运行虚拟机数量正比于 I/O 活动数量,因此相比其他算法拥有更多虚拟机的 SFTD 算法受网络拥塞的影响更大。FFTD 和 BDT-AI 的代价/预算比值会随着预算变得宽松而减小,而 SFTD 会随着预算增加而分配更多的虚拟机,FFTD 和 BDT-AI 则可以利用剩余预算配置更快的虚拟机资源类型。

图 4(e)给出了 Epigenomics 工作流的实验结果。对于较紧密的预算间隔,SFTD 具有更高的代价/预算比值,随着预算的增加,该比值会逐渐减小,原因在于:SFTD 提供的虚拟机数量在预算间隔内基本维持常量,且在预算增加时会选择更快的虚拟机,而其他算法则恰好相反。类似于 CyberShake、Epigenomics 工作流中的多数任务为 I/O 和 CPU 密集型,预算增加时提供更多虚拟机并不会改进 FFTD 和 BDT-AI 的性能。然而,FFTD 在所有实验情况下的预算性能均为最优。

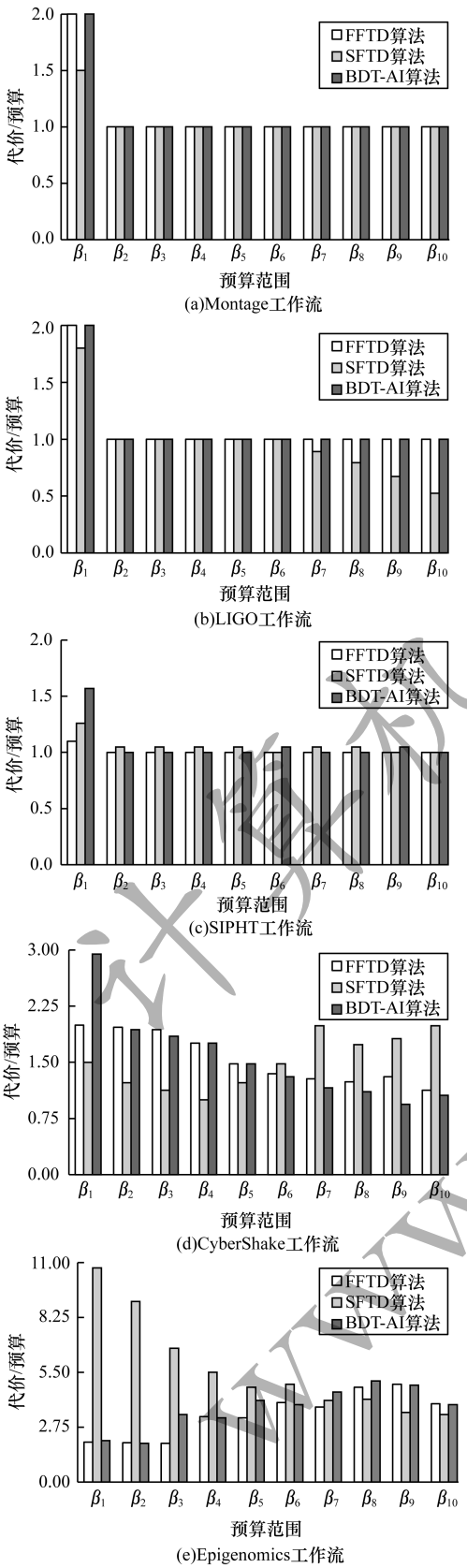

图 4 FFTD、SFTD 和 BDT-AI 在 5 种工作流中的调度代价/预算对比

综合以上场景,FFTD 在代价/预算比值性能上有 88% 以上的实验结果均等于或优于 BDT-AI,除去算法得到的代价与预算相等的情况,相比 BDT-AI,FFTD 和 SFTD 得到的约束满足率高于 62% 。BDT-AI 唯一优于 FFTD 的情形是在 CyberShake 工作流中,在该工作流中 BDT-AI 算法在 90% 的场景中实验结果更优。

实验 2 调度时间评估。图 5(a)为 Montage 工作流的实验结果。在前半部分预算间隔下算法间的差异不大,这主要与 Montage 工作流的任务特征有关,其任务多为 I/O 型任务而非 CPU 型任务,因此选择更多的虚拟机类型并不会对调度时间产生较大影响。然而,随着预算的增加,后半部分预算间隔的增加使得算法得到的调度时间明显不同,原因在于:SFTD 在运行过程中过多的虚拟机提供数量导致了主要的性能下降,而 Montage 工作流并非 I/O 密集型工作流类型,I/O 活动导致的拥塞并不严重。在约 70% 的场景下 FFTD 可以得到比 BDT-AI 更短的调度时间,而 2 种算法在代价/预算比值上具有相似的性能。

图 5(b)是 LIGO 工作流的实验结果。尽管图形趋势与 Montage 相似,但算法间的不同比较明显。FFTD 在所有场景下调度时间最短。同样在图 5(c)显示的 SIPHT 工作流中也有相同的结果,可以看出 FFTD 的调度时间相比 BDT-AI 更加稳定。

图 5(d)是 CyberShake 工作流的实验结果。FFTD 在 60% 场景下具有比 BDT-AI 更短的调度时间,而 SFTD 表现最差,原因在于:CyberShake 是一种数据密集型工作流,涉及较多数据传输,而 SFTD 随着预算增加会提供更多虚拟机,导致性能变差。

图 5(e)是 Epigenomics 工作流的实验结果。调度时间趋势与其他工作流明显不同,主要由于该工作流同时由 CPU 和 I/O 密集型任务构成,提供较低数量的虚拟机就可执行该工作流。同时,FFTD 也具有较短的调度时间。

综合以上场景,FFTD 相比 SFTD 和 BDT-AI 算法在 84% 的场景中具有更短的调度时间,在满足预算约束指标且 FFTD 与 BDT-AI 具有相同性能的情况下,FFTD 在 80% 场景中调度时间更短,平均可以降低 34% 的调度时间。同时,在 FFTD 具有更低的代价/预算比值时(更长调度时间),该算法在 93% 的场景下调度时间比 BDT-AI 更短,平均可降低 18% 的调度时间,综合性能更优。

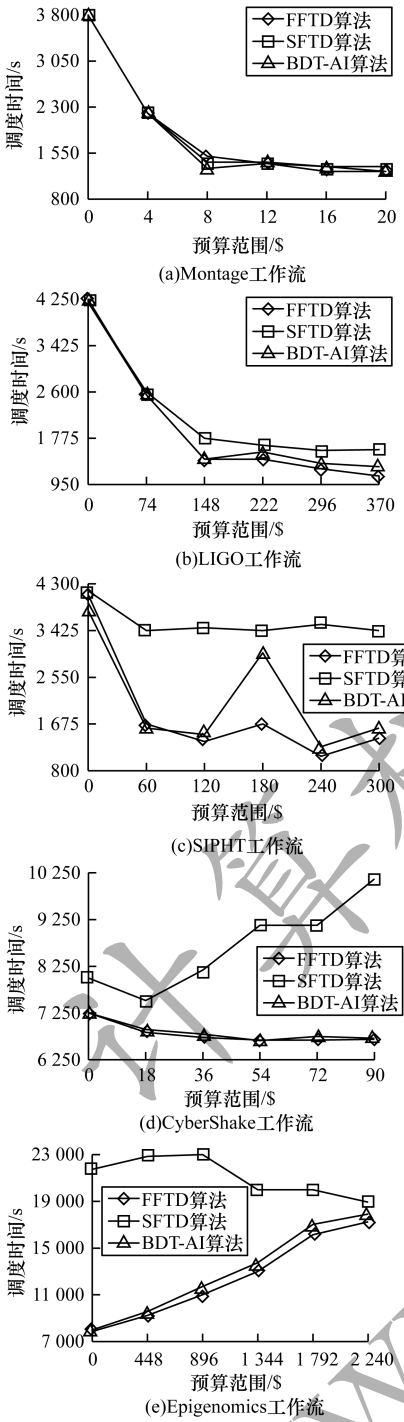

图 5　FFTD、SFTD 和 BDT-AI 在 5 种工作流中的调度时间对比

4.2　虚拟机资源利用率

高虚拟机资源利用率表明算法可以有效进行任务与虚拟机的映射,充分利用虚拟机的空闲时间槽。虚拟机资源利用率指标适用于评估算法处理粗粒度云资源账单周期下的性能,实验结果如图 6 所示。对于 Montage 工作流,FFTD 相比 BDT-AI 在 50% 场景下具有更高的虚拟机资源利用率。对于 LIGO,FFTD 在所有场景下均具有最高的虚拟机资源利用率。但 FFTD 在 90% 场景下相比 BDT-AI 具有更高的虚拟机资源利用率。在 CyberShake 工作流中,BDT-AI 在 60% 场景中相比 FFTD 具有更高的虚拟机资源利用率,从而验证了实验 1 和实验 2 结果的正确性。在 Epigenomics 工作流中,FFTD 在 80% 场景下相比 BDT-AI 具有更高的虚拟机资源利用率。

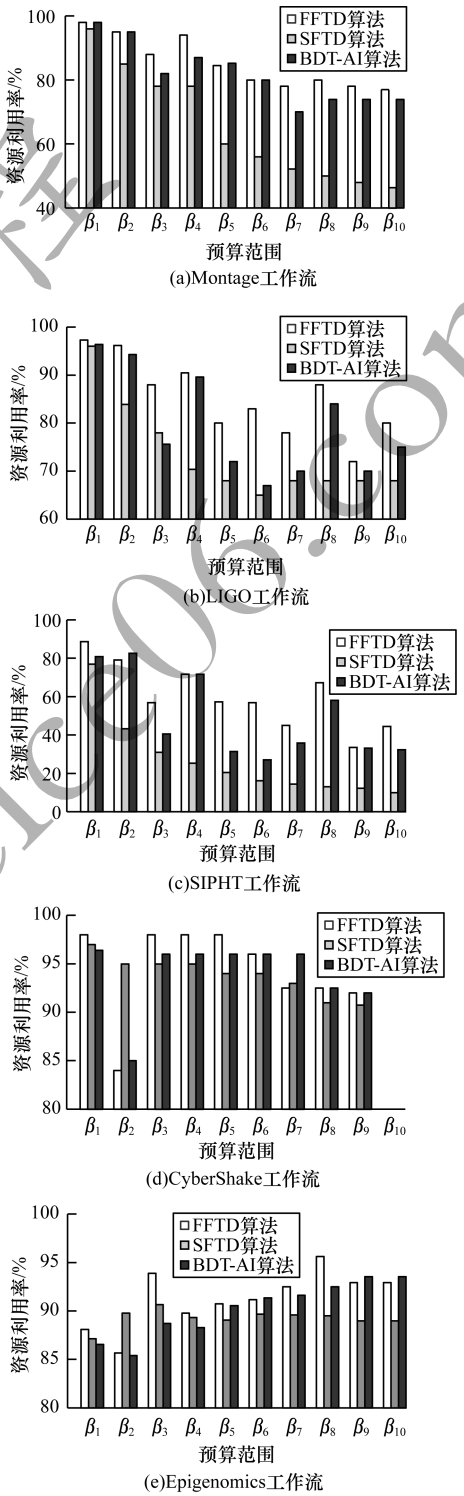

图 6　FFTD、SFTD 和 BDT-AI 在 5 种工作流中的虚拟机资源利用率对比

综合以上场景,在 72% 的场景下,FFTD 在虚拟机资源利用率指标上具有比 BDT-AI 更好的性能。相比 BDT-AI 算法,FFTD 和 SFTD 算法平均可以提高 27% 的虚拟机资源利用率。CyberShake 工作流情况较特殊,其原因在于:BDT-AI 的任务层次划分策略较适用于数据密集型工作流,更符合 CyberShake 工作流的层次特征。

从分析结果可以看出,本文算法在多数仿真场景下实现了性能优化,但也存在一些不足:1)FFTD 和 SFTD 算法在预算分配上采用不同机制,会导致得到的预算约束满意度和调度时间之间的不匹配,FFTD 的综合效果相对更好,但其并未提供一种可量化的方法衡量这 2 个指标的综合性能表现;2)FFTD 和 SFTD 算法在处理不同结构类型的工作流时表现出较大的性能差异,尤其在部分工作流类型中甚至性能差于 BDT-AI 算法,因此在算法可扩展性方面需要进一步提高灵活性,使其适用于工作流结构及任务密度的变化。

5 结束语

为解决云环境下粗粒度账单周期的工作流部署问题,提出一种基于预算分配的工作流调度算法。以在满足预算的前提下最小化调度时间为目标,将工作流部署分解为预算分配和资源提供与调度进行细分求解。在预算分配阶段,设计基于快优先和慢优先的预算分配算法,得到预算在各任务间的子分配。在资源提供与调度阶段,基于任务的最早完成时间的降序排列进行任务选择,在虚拟机可重用的情况下根据单个任务的子预算进行资源提供。实验结果表明,本文算法在多数实验场景中,在代价/预算、调度时间及虚拟机资源利用率 3 项指标上均得到比对比算法更好的性能。下一步将在约束条件和优化目标相同的情况下,研究多工作流动态负载环境下的工作流调度与部署问题。

参考文献

[1] RODRIGUEZ M A,BUYYA R. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments[J]. Concurrency and Computation Practice and Experience, 2016, 29(8): 1-20.

[2] 曹斌,王小统,熊丽荣,等. 时间约束云工作流调度的粒子群搜索方法[J]. 计算机集成制造系统,2016,22(2):372-380.

[3] ABRISHAMI S,NAGHIBZADEH M. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds[J]. Future Generation Computer Systems, 2013,29(1):158-169.

[4] MALAWSKI M, FIGIELA K, BUBAK M, et al. Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization[J]. Scientific Programming,2015,23(1):5-11.

[5] ARABNEJAD V,BUBENDORFER K,NG B. Deadline distribution strategies for scientific workflow scheduling in commercial clouds[C]//Proceedings of IEEE/ACM International Conference on Utility and Cloud Computing. Washington D. C. ,USA:IEEE Press,2017: 137-146.

[6] 沈虹,李小平. 带准备时间和截止期约束的云服务工作流调度算法[J]. 通信学报,2015,36(6):183-192.

[7] 王宏欣,张跃. 带截止期约束的多模态云服务工作流调度[J]. 小型微型计算机系统, 2016, 37(11): 2437-2442.

[8] WU Fuhui, WU Qingbo, TAN Yusong, et al. PCP-B: partial critical path budget balanced scheduling algorithms for scientific workflow applications[J]. Future Generation Computer Systems, 2016, 60(2): 22-34.

[9] WU Chase,LIN Xiangyu,YU Dantong,et al. End-to-end delay minimization for scientific workflows in clouds under budget constraint[J]. IEEE Transactions on Cloud Computing,2015,3(2):169-181.

[10] WANG Xiaotong, CAO Bin, HOU Chenyu, et al. Scheduling budget constrained cloud workflows with particle swarm optimization[C]//Proceedings of 2016 IEEE Conference on Collaboration and Internet Computing. Washington D. C. ,USA:IEEE Press,2016: 219-226.

[11] VERMA A, KAUSHAL S. Budget constrained priority based genetic algorithm for workflow scheduling in cloud[C]//Proceedings of the 5th International Conference on Advances in Recent Technologies in Communication and Computing. Washington D. C., USA:IEEE Press,2013:216-222.

[12] BUYYA R. Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods[M]. New York,USA:ACM Press,2017.

[13] ARABNEJAD V,BUBENDORFER K,NG B. Deadline distribution strategies for scientific workflow scheduling in commercial clouds[C]//Proceedings of IEEE/ACM International Conference on Utility and Cloud Computing. Washington D. C. ,USA:IEEE Press,2017: 137-146.

[14] JUVE G, CHERVENAK A, DEELMAN E, et al. Characterizing and profiling scientific workflows[J]. Future Generation Computer Systems, 2013, 29(3): 682-692.

[15] CALHEIROS R,RANJAN R,BELOGLAZOV A,et al. CloudSim:a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J]. Software Practice and Experience,2011,41(1):23-50.

[16] MAO Ming,HUMPHREY M. A performance study on the VM startup time in the cloud[C]//Proceedings of IEEE International Conference on Cloud Computing. Washington D. C. ,USA:IEEE Press,2012:423-430.

编辑 陆燕菲