文章编号：0253-374X(2021)08-1192-10

DOI：10.11908/j.issn.0253-374x.20519

基于集聚系数的工作流切片与多云优化调度

王鹏伟，雷颖慧，赵玉莹，章昭辉

（东华大学 计算机科学与技术学院，上海 201620）

摘要：已有的工作流云调度研究，通常将任务和云资源一一对应，难以解决由于频繁的数据通信而带来的完工时间上升、成本增加以及可能的故障风险等问题。因此，为减轻任务间数据通信对完工时间和成本的影响，提出了一种基于集聚系数的工作流切片与多云优化调度解决方案。通过聚类算法对工作流进行初步切片，引入集聚系数来判断和优化切片效果，并在寻找调度方案的过程中根据云实例的实际情况动态地调整切片结果。实验结果表明，所提方案能够有效地减少工作流中因大量数据通信而带来的高昂成本和完工时间。

关键词：工作流调度；切片；集聚系数；多云

中图分类号：P312　　**文献标志码**：A

Clustering Coefficient-Based Workflow Slicing and Multi-Cloud Scheduling

WANG Pengwei, *LEI Yinghui*, *ZHAO Yuying*, *ZHANG Zhaohui*

(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

Abstract: Workflow scheduling in multi-cloud environment is a research hotspot and challenge in recent years. The dependencies in workflow are usually represented by the transmission of data, which also determines the execution order of tasks. Existing studies for workflow scheduling usually map each task to a different cloud resource, which is difficult to solve the problems of increasing make-span and cost, and the possible failure risk caused by frequent data communication. In order to reduce the impact of data communication between tasks, this paper proposes a workflow slicing and multi-cloud scheduling solution based on clustering coefficient. Preliminary slicing of workflow is conducted by using a clustering algorithm, and the clustering coefficient is introduced to evaluate and optimize the slicing effect. In the process of finding the optimal scheduling solution, the slicing result is adjusted dynamically according to the actual situation of cloud instances. Experimental results show that the proposed method can effectively reduce the high cost and make-span caused by large amount of data communications in workflow.

Key words: workflow scheduling; workflow slicing; clustering coefficient; multi-cloud

工作流是一组具有依赖性的任务组成的用于完成特定功能的集合。部署在计算机上的工作流，其依赖性通常由数据的传输表示，并且决定了任务的执行顺序。不同的工作流具有不同的拓扑结构，如管道、数据分布和数据聚合，也有不同的资源需求，包括中央处理器(central processing unit，CPU)、内存、输入/出(Input/Output，I/O)等。不同需求所要求的资源类型也不同，在CPU密集的工作流中，任务需要更多的时间来实行计算。在内存密集型的工作流中，任务需要较高的物理内存使用量。而对于I/O密集型工作流，它需要更多的时间执行I/O操作。由于复杂的工作流(如科学工作流)规模大、任务量多、结构复杂，而云计算这种灵活按需的使用模式则很好地为其提供了一种更加高效的运行环境。

云工作流结合了云计算和工作流的诸多特点，是普通工作流往云计算环境下进行迁移所得。云工作流调度是指用户所提交的工作流任务分配到合适的计算资源上执行，并根据资源的使用量即时地支付相应费用。在云计算环境中执行工作流调度时，有两个问题需要考虑，一个是资源的调配，包括确定工作流需要的云资源类型和数量，这意味着需要确定租用多少台实例、它们的类型，以及何时启动，何

收稿日期：2020-12-14

基金项目：国家自然科学基金项目(61602019)、东华大学“励志计划”项目(LZB2019003)、上海市“科技创新行动计划”高新技术领域项目(19511101802)、中央高校基本科研业务费专项资金项目

第一作者：王鹏伟(1984—)，男，副教授，工学博士，主要研究方向为云计算与边缘计算、服务计算、大数据等.
E-mail：wangpengwei@dhu.edu.cn

论文拓展介绍

时关闭等问题。还有一个问题是实际的调度或任务分配阶段，将每个任务都映射到最合适的资源上。在现有的研究中，通常结合了这两个问题一起考虑。

随着工作流规模和计算量的不断增长，它们对分布式基础设施的需求也逐渐加大，如何在分布式环境中有效地调度和部署工作流，是一个值得研究的问题。由于能够提供近乎“无限”资源的特性，云计算可以令个人或组织在不需要构建基础设施的前提下，按照需要获取、配置和使用云资源，并按使用进行付费。根据云资源的特性，为了使成本、性能等指标达到最优，需要进行良好的调度和优化，特别是当多个指标需要同时优化时，这个问题变得更具挑战性。

工作流调度，不管在哪种环境下执行，其目的都在于得到一个好的调度方案，将任务和资源做一个映射，保证工作流的成功执行，并优化某些指标。已有的工作流调度相关研究，根据求解方式主要可以分为基于给定约束[1-3]、基于帕累托解[4-6]和基于权重[7-8]的三类方法。这些相关研究大多将单独的任务和云实例匹配起来，一个任务分配到一个实例上，一个实例可能也只被分配到一个任务。而工作流是一个有数据依赖性的结构，这么做忽略了频繁的数据通信可能带来的成本和时间上升，以及故障增加的风险。为此，本文中提出了一种基于集聚系数的工作流切片与多云优化调度框架（clustering coefficient-based workflow fragmentation and scheduling，CWFS）。该框架首先采用聚类的方式将工作流初步切分成若干个子工作流，然后利用集聚系数来优化调整切片结果。在优化调度的过程中，根据云实例的实际情况，利用集聚系数动态地调整工作流切片并完成调度。

1　示例场景与问题提出

工作流由于其任务之间存在数据传输而具有依赖性。在工作流调度过程中，将工作流中的任务和云计算资源匹配，不仅要注意任务本身的执行需求，也要注意任务之间的数据通信带来的消耗。对于一些工作流，尤其是数据通信密集型的工作流来说，频繁或者大量的数据通信都会影响总完工时间和执行成本。此外，如果有频繁数据传输关系的任务或者大数据量的任务部署在不同的云资源上，在数据传输过程中，更容易出现错误，导致发生故障重传的概率更大，容易影响后面的任务执行。

图1展示了一个包含15个任务的工作流，任务之间有数据传输的关系。由于工作流的依赖性，任务要想开始必须等到它的父任务将数据传输完成。以任务7为例，任务7的父任务是任务3和任务4，分别要传输2 000个单位和10个单位的数据。而任务3和任务4有共同的父任务：任务1，其要分别传输1 000单位和30单位的数据给任务3和任务4。那么对于任务7来说，要想开始执行，必须等到任务1执行完毕后传输数据给任务3、4，再等到任务3、4执行完毕后传输数据给任务7。在这个过程中，任务1和任务3、任务3和任务7之间传输的数据量远远大于任务1和任务4、任务4和任务7之间的数据量。这也就意味着，假设任务所在云实例传输带宽相等，任务1执行完毕后，任务3等待数据传输的时间是任务4的33倍。而任务7要想开始，等待任务3传输数据的时间是等待任务4传输时间的200倍。可以看出，有大量的时间被浪费在了等待其中一个父任务传输数据的时间。如果将任务1、任务3、任务7放在同一个云实例上，那么任务7需要的数据传输时间只是任务1到任务4，与任务4到任务7的数据传输时间之和。而此时，一共只有40个单位的数据传输，显然会大大减少数据传输带来的成本和时间消耗。

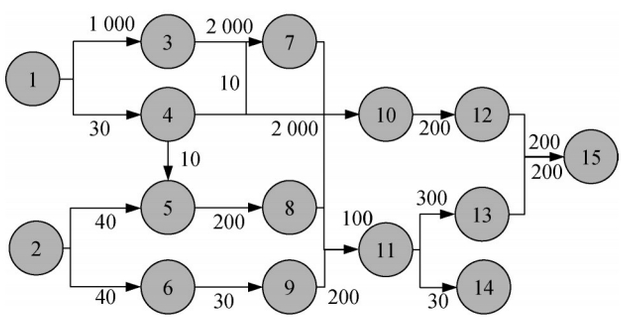


图1　一个有15个任务的工作流

Fig. 1　A workflow with 15 tasks

因此，在进行工作流的优化调度时，减少工作流任务在不同资源之间的数据传输来保证完工时间满足截止期约束，以及其他的一些优化目标，是一个值得关注的研究方向。从上述场景中可以看出，需要将工作流切分成若干个子工作流，使得子工作流内数据传输尽量频繁，子工作流之间的数据传输尽量少，即子工作流内聚性强，外联性弱。在已有的研究中，研究者们通常会为每一个任务分配一个独立的云资源，任务在不同云之间频繁的数据通信不利于充分地利用云资源，以及更好地节约成本和时间。

工作流切片要求带来的效果类似于聚类：类内

元素相似程度高,类外元素相似程度低。因此可以将工作流切片建模成一个聚类问题,利用任务间的数据量来衡量任务间的关系。另外,如果只考虑工作流切片,那么切片结果有可能超过实际的云实例可用负载。以图1为例,当任务1、3、7被划分为一个子工作流时,可以使其内部数据传输量大,外部少。但是考虑到执行任务的计算需求和云实例的承载能力,如果没有一个合适的云实例能够容纳这个子工作流,那么必须要对它再次进行调整。因此在进行优化调度过程中,依然需要对切片的结果进行动态调整,使得云资源的利用率达到一个合理的范围。

2 用于工作流切片的集聚系数

集聚系数概念诞生于图论,表示一个图中顶点的聚集程度。不同于判断类性能的标准,集聚系数更关注节点间的密度。当节点间关系紧密时,那么它们的集聚系数就会变高。如果一个节点相连的节点之间有关系,表示这个群体相互之间比较紧密,那么也会有一个比较高的集聚系数。集聚系数有局部集聚系数和平均集聚系数,前者给出了单个节点的度量,可以判断图中每个节点附近的集聚程度,后者旨在度量整个图的平均集聚性。在近期研究中,集聚系数被广泛应用于社交网络分析、可视化网络安全分析、小世界网络分析等领域。Murray等[9]引入集聚系数来描述小世界模型。集聚系数可以用于判断聚类的效果,Zhong等[10]使用其来解决词关系的聚类问题。

工作流因为任务间有数据传输的关系,将数据传输量大的任务划分成一个子工作流,这样子工作流之间的数据传输就会减少。因此可以采用聚类的方法进行工作流的切片研究。很多聚类算法都基于一些给定的值来进行聚类,这要求使用者必须拥有该领域的一定先验知识。动态的聚类算法可以根据实际情况来分析模型结构来给出k值,但是缺少一个良好的标准来判断类内元素的整体相似度。为此,引入图论中的集聚系数,来帮助判断子工作流内部和外部的聚集程度。

给定一个工作流的有向无环图模型(direct acyclic graph,DAG)$G=(T,E)$,其中T是n个任务的集合$t=\{t_1,t_2,\cdots,t_n\}$,E是任务之间依赖关系的集合,$E=\{e_{i,j}|i,j=0,1,2,\cdots,m\}$。依赖关系$e_{i,j}$表示了任务$t_i$和任务$t_j$的依赖约束,它意味着任务$t_j$必须等到任务$t_i$执行完毕才能开始执行。如果两个任务之间有数据传输的行为,那么认为DAG图中这两个任务之间有一条边。每个任务与其他任务连接的边的集合用N来表示。N_i表示与任务t_i连接的边的集合。$|N_i|$表示集合的度,即边的数量。任务t_i的局部集聚系数LC_i是它的相邻任务之间的边的数量与它们所有可能存在边的数量的比值。

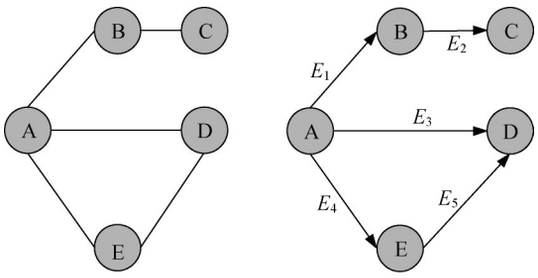
对于无向图来说,n个顶点之间最大的边数是$\frac{n(n-1)}{2}$,那么它的局部集聚系数计算方式如式(1)所示,其中$\mathrm{Neighbor}_i$表示由与任务t_i相连的任务所形成的边的集合,即$\mathrm{Neighbor}_i=\{e_{j,k}|e_{i,j},e_{i,k}\in N_i,e_{j,k}\in E\}$。图2a是一个无向无权图(图中,A~E表示顶点),以任务t_A举例,与它有边关系的任务是t_B、t_D、t_E,因此$N=\{e_{\mathrm{A,B}},e_{\mathrm{A,D}},e_{\mathrm{A,E}}\}$,其度为3。而在三个相连的任务中,只有任务$t_\mathrm{D}$、$t_\mathrm{E}$之间有边的关系,因此$\mathrm{Neighbor}_i=\{e_{\mathrm{D,E}}\}$。那么任务$t_\mathrm{A}$的局部集聚系数$\mathrm{LC}_\mathrm{A}=\frac{2}{3\times(3-1)}=\frac{1}{3}$。

$$\mathrm{LC}_i=\frac{2|\mathrm{Neighbor}_i|}{|N_i|(|N_i|-1)} \tag{1}$$

工作流转换成的DAG图是一个有向图,任务间数据的传输可以视为边的权重。权重对于集聚系数的计算影响颇大。不同的权重定义所表示的集聚系数紧密程度也不同。如果权重代表的是距离,那么权重越小顶点间的关系更为紧密;如果权重表示的是关系值,那么权重越大顶点间的关系越紧密。本文中研究的问题,需要将数据量大的任务聚集在一起,因此任务间数据量越大,表示它们越应该聚集在一起,它们的关系应该更紧密。因此,工作流任务的局部集聚系数可以定义为式(2)。

$$\mathrm{LC}_i=\frac{|\mathrm{Neighbor}_i|}{|N_i|(|N_i|-1)}\cdot\frac{\mathrm{Weight}_i}{|N_i|} \tag{2}$$

工作流是一个有向图,因此其完全图的边的数量是$n\times(n-1)$。$\mathrm{Neighbor}_i$和N_i的定义同式(1)中相同,但是这里两个定义都是针对有向图,$e_{\mathrm{A,B}}\neq e_{\mathrm{B,A}}$。$\mathrm{Weight}_i$表示集合$N_i$中的权重和,$\mathrm{Weight}_i=\sum_{e_{i,j}\in N_i}|e_{i,j}|+\sum_{e_{j,i}\in N_i}|e_{j,i}|$。以图2a中的任务$t_\mathrm{A}$为例,与它有边的关系的任务是$t_\mathrm{B}$、$t_\mathrm{D}$、$t_\mathrm{E}$,因此$N=\{e_{\mathrm{A,B}},e_{\mathrm{A,D}},e_{\mathrm{A,E}}\}$,其度为3。而在三个相连的任务中,只有任务$t_\mathrm{D}$、$t_\mathrm{E}$之间有边的关系,因此$\mathrm{Neighbor}_i=\{e_{\mathrm{E,D}}\}$,那么根据式(2),任务$t_\mathrm{A}$的局部集聚系数为$\mathrm{LC}_i=\frac{1}{6}\times\frac{E_1+E_3+E_4}{3}$。$E_1$~$E_5$表示权重。



a 无向无权图　　b 有向无环有权图

图2　无向无权图和有向无环有权图

Fig. 2　An undirected unweighted graph and a directed acyclic weighted graph

从式(1)和(2)中可看出,无向无权图的任务局部集聚系数总是在0～1之间。0表示附近任务之间没有抱团的关系,而1表示附近任务之间联系紧密,接近完全图。有向有权图中,当任务与其相邻任务接近完全图,并且任务间的权重趋近于$+\infty$时,任务的局部集聚系数值趋近于$+\infty$。得到任务的局部集聚系数还不足以判断整个图的聚集情况。Watts和Strogatz定义了平均集聚系数[11],通过计算平均值来得到整个图的集聚程度,即$\mathrm{AvgC}=\frac{1}{n}\sum_{i=1}^{n}\mathrm{LC}_i$。

将工作流切成若干个子工作流后,根据子工作流内部任务之间的数据聚集程度和子工作流之间的数据聚集程度判断切片的质量。假设工作流切分后,有子工作流A,$G_A=\{T_A,E_A\}$和B,$G_B=\{T_B,E_B\}$。子工作流A的集聚程度如式(3)所示。其中,t_i表示第i个任务,T_A表示子工作流A内包含的任务集合,$e_{i,j}$表示任务t_i和t_j之间的边,E_A表示子工作流A内部包括的边的集合,LC_i表示任务t_i的局部集聚系数。

$$\mathrm{intro}_A=\frac{\sum_{t_i\in T_A,e_{i,j}\in E_A}\mathrm{LC}_i}{|T_A|}\tag{3}$$

仅靠子工作流内部的集聚程度无法判断工作流切片的合理性,需要对子工作流之间的集聚程度进行判断。下面定义类间集聚系数[10],公式(4)是对子工作流A而言,其与子工作流B之间的类间集聚系数。inter(A,B)强调的是A受B关联的类间集聚系数。如果是B受A关联的类间集聚系数,那么计算公式变为inter(B,A)。

$$\mathrm{inter}(A,B)=\frac{\sum_{t_i\in T_A,e_{i,j}\in U}\mathrm{LC}_i}{|T_A|},\ U=E_A\cup E_B\tag{4}$$

图3是一个被切分成4个子工作流的工作流。虚线圈起的部分表示被切分的子工作流。虚箭头表示子工作流之间的数据传输,实箭头表示子工作流内部的数据传输。对于子工作流A来说,如果$\mathrm{inter}(A,B)$大于intro_A,说明子工作流A和子工作流B的类间集聚程度要大于子工作流A的类内集聚程度,将那些与子工作流A有边的关系但是属于子工作流B的任务划分给子工作流A可以提高A的集聚程度。同理,如果$\mathrm{inter}(B,A)$大于intro_B,则说明把子工作流A中那些与子工作流B有边的关系的任务划分给子工作流B更好。当$\mathrm{inter}(A,B)\geqslant\mathrm{intro}_A$和$\mathrm{inter}(B,A)\geqslant\mathrm{intro}_B$同时成立时,说明子工作流$A$和子工作流$B$各自的类内集聚程度都没有它们受另一个子工作流关联的类间集聚程度强,因此将它们合并成一个新子工作流,可以获得更高的集聚程度。同理,如果子工作流A可以被分成A_1和A_2两个子工作流,且有$\mathrm{inter}(A_1,A_2)<\mathrm{intro}_{A_1}$和$\mathrm{inter}(A_2,A_1)<\mathrm{intro}_{A_2}$同时成立。那么说明与子工作流$A$相比,切分成的两个新子工作流有更高的集聚程度。

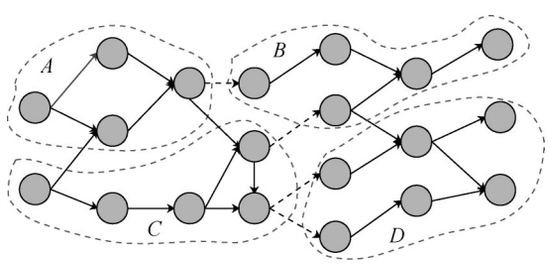


图3　一个被切分的工作流

Fig. 3　A segmented workflow

通过不断地优化调整工作流切片结果,可以保证子工作流内的数据依赖较强,子工作流间的数据依赖程度较弱。

3　工作流切片与优化调度框架

3.1　总体框架

CWFS的目的是通过将工作流切分成若干个子工作流,再使用优化算法为它们找到合适的云实例,在减少数据传输依赖的情况下,找到满足优化目标的调度方案。CWFS包括两个部分:基于集聚系数的工作流切片和基于切片的优化调度。整体框架如图4所示。

基于集聚系数的工作流切片分为两步,首先使用聚类算法,根据任务间的数据通信量对工作流进行一个初步的切片,将通信量较大的任务聚成一个类。得到初步的切片结果后,根据第4节中关于集

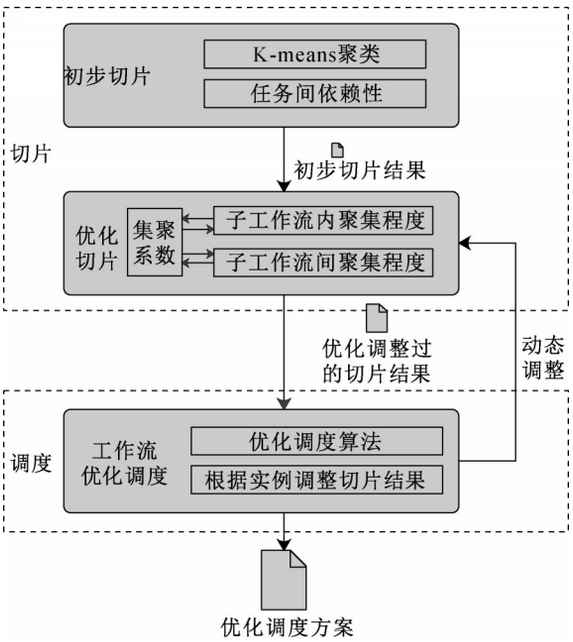


图4 CWFS整体框架

Fig. 4 Framework of CWFS

聚系数的相关定义,以及对切片内部和切片之间紧密度的判断公式,对工作流初步切片结果进行优化调整,使得调整后的子工作流内聚性强,外联性弱,从而得到一个基于集聚系数的工作流切片结果。

在工作流切片过程中,只考虑了任务间的通信情况,而没有考虑实际的云实例承载能力。因此,在基于切片的优化调度过程中,使用启发式算法进行寻优时,根据云实例的实际情况,可能会出现子工作流超过云实例的承载能力或者只占云实例承载能力的一小部分,造成无法找到合适的实例或者造成浪费的现象。因此在工作流优化调度时,需要根据实际情况动态地调整切片结果,使其可以找到合适的调度方案。CWFS框架的输出是一个工作流-云计算资源的调度方案。

3.2 基于集聚系数的工作流切片

3.2.1 初步切片

工作流是一组有依赖关系的任务集合。根据任务之间的数据依赖将工作流切分成若干个子工作流,其过程类似于聚类。本节采用K均值聚类算法(K-means)来进行工作流的初步切片工作,主要包括以下步骤:

(1) 导入工作流,并且将它转换成拥有一个虚拟入口任务(t_{entry})和虚拟结束任务(t_{exit})的DAG图。

(2) 根据实验工作流的大小,指定合适的k值,从工作流的任务集合中,随机选择k个任务作为初始的聚类中心(不包括两个虚拟任务)。

(3) 遍历工作流的每一个任务,计算它们和聚类中心之间的数据传输量。如果一个任务到每一个聚类中心都没有直接的数据传输,那么通过计算它的父任务们的数据传输情况来判断它们应该属于哪一类,并且将它们分别加入数据量最大的聚类中心所在的类中。

(4) 当所有任务都加入到某个类中后,每个类重新计算聚类中心。计算类中每个任务到其他任务的数据量,将最大的那个任务定义为新的聚类中心,一轮迭代结束。

(5) 判断新的k个聚类中心和上一轮迭代相比是否有变化,有则跳转步骤(3);若否,则迭代结束,进行步骤(6)。

(6) 得到工作流初步切片结果,输出k个子工作流集合。

3.2.2 切片优化

采用K-means算法进行初步切分时,k值的选择是基于使用者自己的经验和判断,无法证明划分成k个子工作流是合适的。可能某些工作流中有些任务需要被划分出去,而有些子工作流需要合并成一个新的工作流,凭自身经验很难直接判断这一点。为此,优化的工作流切片要求在初步切片后采用合理的方法对初步切片结果进行合理的调整,因此要使用第2节提到的用于工作流切片的集聚系数。如第2节所述,有两个衡量的标准——类间集聚系数和类内集聚系数,可以帮助衡量工作流切片后的结果,判断它们是否需要进一步切分或合并。下面首先给出两个定义。

定义1 子工作流分割 :现有子工作流A,如果满足下列条件,那么子工作流A将切分成子工作流A_1和子工作流A_2。

如果$\text{intro}_A < \text{inter}(A_1, A_2)$成立,同时$\text{intro}_A < \text{inter}(A_2, A_1)$也成立。那么子工作流$A$可以被分割成新的子工作流$A_1$和子工作流$A_2$。

定义2 子工作流合并 :现有子工作流A和B,如果两者满足下列条件,那么子工作流A和B将合并成一个新的子工作流$(A+B)$。

如果$\text{inter}(A,B) \leqslant \text{intro}_{A+B}$成立,同时$\text{inter}(B,A) \leqslant \text{intro}_{A+B}$也成立。那么子工作流$A$和子工作流$B$可以合成为一个新的子工作流$(A+B)$。

切片优化就是对工作流进行初步切分后,利用定义1和定义2对初步切分后的子工作流集合不断地进行优化调整的过程。迭代的终止条件可以根据设置的标准或者迭代次数来确定。每次迭代分为子

工作流分割和子工作流合并两部分操作。首先是对导入的工作流进行一个初始切片的工作，得到初步切分好的子工作流集合后，给每个子工作流添加“0”的标记。

分割操作过程为：依次从未处理的子工作流集合中取出一个子工作流，对其进行预分割操作，预分割即为依次将子工作流分为两份，利用类内集聚系数和类间集聚系数来判断预分割结果是否可行；如果满足定义1，则此次分割操作可以进行，并且将两个新得到的子工作流标记改为“1”；如果不可行则放弃此次分割操作，并且给该子工作流的状态改为“1”；接着继续从标记“0”的工作流中拿出下一个子工作流进行处理。当未处理的子工作流集合为空时，表示这轮迭代的分割操作已经全部结束，进入合并操作。

合并操作过程如下：取出一个标记为“1”的子工作流，计算它的类内集聚系数，以及该工作流和其关联子工作流之间的类间集聚系数，判断它们是否满足定义2；如果满足则对两个子工作流进行合并，并且将合并后的新子工作流的状态变为“2”；依次取集合中的子工作流，直到集合为空；此时迭代中的合并操作也已经完成。如果此轮迭代满足了迭代停止条件，那么将输出一个经过切片优化调整的子工作流集合；如果不满足，则跳转到分割操作继续进行切片工作。

4　基于切片的工作流多云优化调度

为了降低工作流调度过程的数据通信量，减少完工时间，在对工作流进行切片的基础上，本节中给出一种基于切片和基于免疫粒子群优化算法(immune-based particle swarm optimization, IMPSO)[12]的工作流多云优化调度方法。

工作流模型用DAG图表示，其中$G=(T,E)$，$T=\{t_1,t_2,\cdots,t_n\}$表示工作流任务集合，每个任务的参数为$t_i=(\mathrm{id},\mathrm{namespace},\mathrm{name},\mathrm{size})$；其中namespace为任务所属的工作流名称，name为任务名称，size是任务需要的计算量。边的集合$E=\{e_{i,j}|i,j=0,1,2,\cdots,N\}$，其中$e_{i,j}=(\mathrm{orig},\mathrm{dest},\mathrm{data})$，分别表示父任务、子任务和它们之间传输的数据量。选择的依然是云实例类型，不对更底层的结构进行考虑。云实例$\mathrm{instance}_i=(\mathrm{id},\mathrm{speed},\mathrm{cost},\mathrm{bandwidth},\mathrm{bdprice},\mathrm{type})$，其中，speed表示云实例计算能力；cost是一个时间单位内的成本；bandwidth为带宽；bdprice为单位传输量的传输价格；type为实例类型。用户将工作流提交到调度器进行处理时，对于切片后的子工作流，调度器将它们调度在同一资源节点上，此时子工作流内部的传输成本和时间均为0。因此子工作流sub_i的总成本和总完工时间分别为：

$$\mathrm{COST}(\mathrm{sub}_i)=\sum_{j=1}^{n}(\mathrm{EC}(\mathrm{t}_j,\mathrm{I}),t_j\in\mathrm{sub}_i \tag{5}$$

$$\mathrm{MAKESPAN}\left(\mathrm{sub}_i\right)=\max_{t_j\in\mathrm{sub}_i\ \mathrm{and}\ \mathrm{CHILD}(t_j)=\varnothing}\mathrm{FT}\left(\mathrm{t}_j\right) \tag{6}$$

由于不需要计算传输成本，因此子工作流sub_i的总成本即为各个任务的执行成本的总和。$\mathrm{EC}(t_j,I)$是任务在实例上执行的成本；子工作流的完工时间由其内部最后执行的任务的结束时间决定，值得注意的是，子工作流和工作流不同，为了方便计算，通常会在工作流开始和结束的位置分别加一个虚拟的任务，因此工作流的开始和结束任务只有一个。而切片的子工作流可能会有多个可以同时开始的任务和多个结束任务。因此，子工作流的完工时间应是结束时间最晚的那个任务。而公式(6)中的$\mathrm{CHILD}(t_j)=\varnothing$指的是在该子工作流内部没有子任务，并不一定在整个工作流中没有子任务。任务的结束时间由公式(7)和(8)决定，同样式(7)中，$\mathrm{PRED}\left(t_i\right)=\varnothing$指的是任务在子工作流内没有父任务。

$$\mathrm{ST}(t_i)=\begin{cases}\max\limits_{\mathrm{t}_j\in\mathrm{pred}(t_i)}\{\mathrm{ST}\left(t_j\right)+\mathrm{ET}\left(t_j,I\right)+\mathrm{TT}(e_{j,i})\},\ \mathrm{PRED}\left(t_i\right)=\varnothing\\ \max\limits_{t_j\in\mathrm{pred}(t_i)}\left\{\mathrm{ST}\left(t_j\right)+\mathrm{ET}\left(t_j,I\right)\right\},\ \mathrm{PRED}(t_i)\neq\varnothing\end{cases} \tag{7}$$

$$\mathrm{FT}\left(t_i\right)=\mathrm{ST}\left(t_i\right)+\mathrm{ET}\left(t_j,I\right) \tag{8}$$

由此，对于整个工作流W来说，总的成本和时间可以计算得到：

$$\mathrm{COST}(W)=\sum_{i=1}^{m}\left(\mathrm{COST}\left(\mathrm{sub}_i\right)\right)+\sum_{j=1}^{k}\mathrm{TC}_j \tag{9}$$

$$\mathrm{MAKESPAN}(W)=\max_{\mathrm{sub}_j\subseteq W}\mathrm{MAKESPAN}(\mathrm{sub}_i) \tag{10}$$

因此，工作流调度的优化目标如下所示：

$$\begin{aligned}&\min.\quad \mathrm{COST}(W),\mathrm{MAKESPAN}(W)\\&\mathrm{s.t.}\quad \mathrm{MAKESPAN}(W)\leqslant D\end{aligned} \tag{11}$$

IMPSO方法引入了免疫机制，在粒子寻找全局

最优解的过程中,不断地进行免疫操作来加强粒子的寻解能力。关于IMPSO方法的详细介绍可参考文献[12]。将工作流切片结果作为IMPSO的输入时,由于切片过程没有考虑到真实的云实例的承载能力。可能存在切片超过云实例的容量的情况。因此,在使用IMPSO进行优化调度的过程中,依然需要动态地对切片结果进行调整。在寻优过程中,粒子寻找到的解是一个切片-实例关系对,如果切片过大找不到合适的云实例类型,那么根据提出过的抗体和抗原之间的亲和力公式,粒子的某一维和当前最优粒子之间的距离是无穷大的,那么根据公式计算得到的亲和度就会为0;这意味着,该粒子中存在需要分割的工作流切片,即需要调用工作流切片模块来调整。

图5展示了基于切片和IMPSO算法的工作流多云优化调度方法的流程图。对于待执行的工作流进行基于集聚系数的工作流切片后,调用IMPSO算法进行调度。进行粒子的初始化后,计算抗体和抗原间的亲和度。如果切片的大小超过了实际的云实例负载,现有的云实例类型无法为切片找到合适的选择,那么将调用基于集聚系数的切片方法,重新动态地调整切片结果。直到所有切片都能找到合适的实例,重新计算各抗体的亲和度,并且加入免疫操作,包括抗体克隆、抗体变异等。经过若干轮的迭代,直到找到合适的抗体,算法结束。输出为一个优化调度的解决方案。

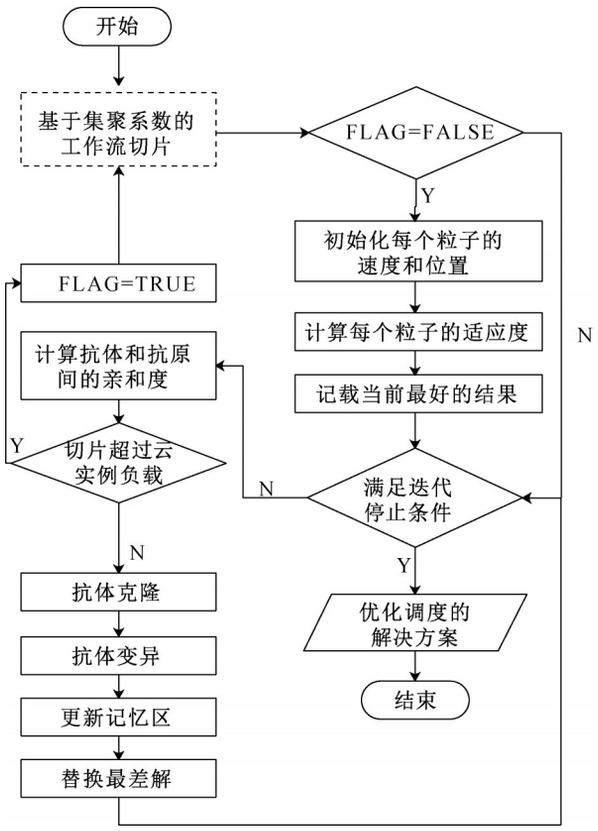


图5 基于切片和IMPSO的工作流多云优化调度

Fig. 5 Multi-cloud workflow scheduling based on slicing and IMPSO

5 实验与分析

5.1 实验设置

为了验证所提方法的有效性,使用真实的工作流Cybershake[13]、Epigenomics和Montage[14]来进行性能评估,实验选用任务数量为100的工作流进行实验。

实验中采用真实的亚马逊AWS EC2云计算实例类型参数,如表1所示。设置的时间间隔为1小时。每个云实例的容量设置为100 000单位,负载达到80 %即视为负载已满。在实验中,使用属性ECU代表实例的速度。

表1 云实例参数

Tab. 1 Cloud instance parameters

实例类型	计算能力/(Mbps)	花费/美元	带宽/(Mbps)	单位传输价格/(美元·Mbps^{-1})
$type_1$	1.0	0.12	10	1.0
$type_2$	1.5	0.195	15	1.8
$type_3$	2.0	0.28	30	3.5
$type_4$	2.5	0.375	10	1.2
$type_5$	3.0	0.48	20	2.3
$type_6$	3.5	0.595	25	2.5
$type_7$	4.0	0.72	15	1.8
$type_8$	4.5	0.855	30	3.5
$type_9$	5.0	1.0	30	3.5
$type_{10}$	5.5	1.25	20	2.3

实验环境的配置为:Core (TM) i5 3.40 GHz, 16 GB RAM, Windows 10, Java 2 Standard Edition V1.8.0。

5.2　实验结果

本文中提出的CWFS框架与IMPSO结合的方法称为CWFS-IM，将CWFS与遗传算法结合的方法称之为CWFS-GA。下面比较和分析CWFS-IM、CWFS-GA、遗传算法GA、免疫粒子群优化算法IMPSO这4种方法的性能。IMPSO的相关参数设置为：学习因子和PSO中的速度更新的惯性因子设置为：$\omega=0.5$; $c_1=2$; $c_2=2$。种群大小和迭代次数被设置为100。结合免疫机制的研究和实验，与免疫有关的参数机制如下：记忆单元容量为种群的一半，随机生成的新粒子数量设置为种群的1/10。所需的克隆的大小被控制在大约两倍的种群数量。CWFS和遗传算法的种群规模设置为100，迭代次数设置为100，交叉概率P_c和变异概率P_m分别为0.5和0.5。动态切片的k值设置成工作流任务数的1/10。

本文选择了3种真实工作流，分别为Cybershake、Epigenomics和Montage。其中Cybershake是数据密集型工作流，执行时有大量的数据传输工作，该工作流是南加州地震中心用于表征地震灾害的工具；Epigenomics是计算密集型工作流，用于自动执行各种基因组测序操作，相对于计算来说，数据传输不是特别多；Montage用于根据输入的图像来创建天空的工作流，其特点是需要大量的I/O，任务间有频繁的通信需求。三种工作流的任务间数据传输情况各不相同，与自身的计算需求相比，Cybershake有最多的数据传输需求，Montage次之，Epigenomics最少。实验分别使用GA、IMPSO、CWFS-GA、CWFS-IM 4种方法对这三个工作流进行调度，4种方法分别运行200次，每10次进行求平均得到一次运行时间。图6中分别展示了Cybershake、Epigenomics和Montage在4种调度方法下的完工时间。

可以看出，在这4种方法中，CWFS-IM和CWFS-GA的效果比IMPSO、GA好，这说明基于集聚系数的切片方法可以有效地减少完工时间。另外，对比这三张图可以看出，对于资源需求和特点不同的工作流，CWFS带来的提升也不尽相同。对于Cybershake来说，CWFS的效果最好，这是由于Cybershake执行过程中有大量的数据传输请求，这会带来大量的数据传输时间。进行合理的工作流切片后，可以将这部分传输时间节省掉。而对于

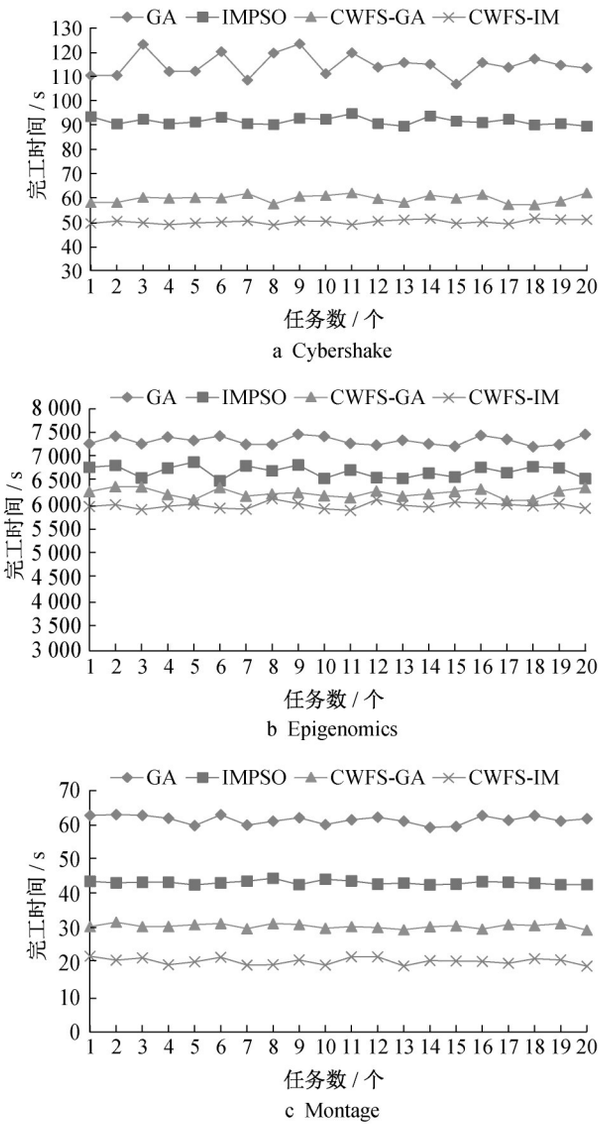


图6　三个工作流在4种调度方法下的完工时间

Fig. 6　Make-span of three workflows in four scheduling methods

Epigenomics来说，这项提升没有Cybershake那么明显，因为Epigenomics工作流更侧重于计算资源需求。

图7中分别展示了三个工作流在4种调度方法下的执行成本。

从图中可以看出，CWFS可以有效地降低成本。不同工作流成本降低的效果和完工时间类似。对于通信需求多的Cybershake工作流来说，CWFS可以带来明显的成本下降效果，因为它节省了大量的数据通信所造成的成本。Montage工作流的成本也有明显下降，而对于Epigenomics工作流来说，效果没有前两者那么明显。

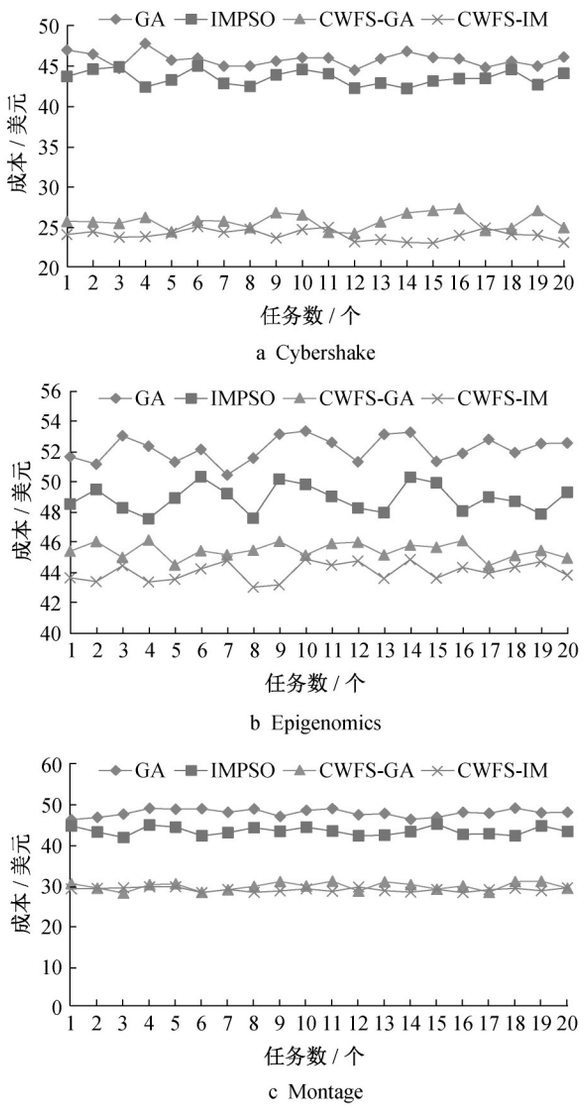


图7 三个工作流在4种调度方法下的执行成本

Fig. 7 Execution cost of three workflows in four scheduling methods

6 结论

在面向多云的工作流优化调度研究中,对于一些数据传输量较多或者数据通信较为频繁的工作流来说,由任务间的依赖性而带来的传输成本和时间消耗不容忽视。为解决该问题,本文提出了先对工作流进行切片,然后再进行优化调度的思路。首先根据工作流切片的实际需要,引入集聚系数概念,并定义了类内聚集系数和类间集聚系数,进而给出了对子工作流进行分割与合并的判断标准;在此基础上,提出了一个基于集聚系数的工作流切片与多云优化调度框架。引入集聚系数来优化调整工作流的切片结果,并在寻优求解过程中根据可用云实例的实际承载能力,动态地调整切片结果。最后,使用三种不同类型的科学工作流,利用真实的云实例信息进行实验,结果表明,相较于对比方法,所提方能够有效减少完工时间和成本。

作者贡献说明:

王鹏伟:论文的提出、构思、方法、写作、修改、验证、分析、校对与编辑。

雷颖慧:论文的数据准备、实验、初稿撰写、验证与分析。

赵玉莹:论文的初稿撰写、图表与格式排版。

章昭辉:论文的阅读、校对与编辑。

参考文献:

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

[2] LIN B, GUO W Z, CHEN G L, *et al*. Cost-driven scheduling for deadline-constrained workflow on multi-clouds [C]// Proceedings of 2015 IEEE International Parallel and Distributed Processing Symposium Workshop. Piscataway: IEEE, 2015: 1191-1198.

[3] RODRIGUEZ M A, BUYYA R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds[J]. IEEE Transactions on Cloud Computing, 2014, 2(2): 222.

[4] DURILLO J J, NAE V, PRODAN R. Multi-objective workflow scheduling: An analysis of the energy efficiency and makespan tradeoff [C]//2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. Delft: IEEE, 2013: 203-210.

[5] XU X, FU S, LI W, *et al*. Multi-objective data placement for workflow management in cloud infrastructure using NSGA-II [J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2020, 4(5): 605.

[6] XU X, MO R, DAI F, *et al*. Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud [J]. IEEE Transactions on Industrial Informatics, 2020, 16(9): 6172.

[7] SHI T, MA H, CHEN G. A seeding-based GA for location-aware workflow deployment in multi-cloud environment [C]// 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington: IEEE, 2019: 3364-3371.

[8] BARIKA M, GARG S, CHAN A, *et al*. Scheduling algorithms for efficient execution of stream workflow applications in multicloud environments[J]. IEEE Transactions on Services Computing, DOI: 10.1109/TSC.2019.2963382.

[9] MURRAY J, WETTIN P, PANDE P P, *et al*. Sustainable wireless network-on-chip architectures [M]. Cambridge, MA: Morgan Kaufmann, 2016.

[10] ZHONG M J, DING Z J, SUN H C, *et al*. A self-learning

clustering algorithm based on clustering coefficient [C]// International Conference on Web Information Systems Engineering. Thessaloniki: Springer International Publishing, 2014: 79-94.

[11] WATTS D J, STROGATZ S H. Collective dynamics of 'small-world' networks[J]. Nature, 1998, 393(6684): 440.

[12] WANG P W, LEI Y H, Agbedanu P, *et al*. Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm[J]. IEEE Access, 2020, 8: 29281.

[13] GRAVES R, JORDAN T H, CALLAGHAN S, *et al*. CyberShake: A physics-based seismic hazard model for southern California[J]. Pure and Applied Geophysics, 2011, 168(3/4): 367.

[14] BERRIMAN G B, DEELMAN E, GOOD J C, *et al*. Montage: A grid-enabled engine for delivering custom science-grade mosaics on demand[C]//Optimizing Scientific Return for Astronomy through Information Technologies. Glasgow: International Society for Optics and Photonics, 2004, 5493: 221-232.

(上接第1143页)

[9] PIOTR B, JERZY M. Euro III / Euro IV emissions: A study of cold start and warm up phases with a SI (Spark Ignition) engine [R]. Detroit:SAE, 1999.

[10] 郝吉明，傅立新，贺克斌. 城市机动车排放污染控制：国际经验分析与中国的研究成果[M]. 北京：中国环境科学出版社，2001.
HAO Jiming, FU Lixin, HE Kebin. Vehicle emissions control: analysis of international experience and research findings in China [M]. Beijing: China Environmental Science Press, 2001.

[11] 刘皓冰. 基于行驶工况的城市机动车尾气排放研究[D]. 上海：同济大学，2013.
LIU Haobing. Estimating urban vehicle emissions based on the driving cycles [D]. Shanghai: Tongji University, 2013.

[12] 单肖年. 基于行驶工况的车辆能耗与排放评估方法研究[D]. 上海：同济大学，2018.
SHAN Xiaonian. Vehicles energy and emissions estimation based on driving cycles [D]. Shanghai: Tongji University, 2018.

[13] 李孟良，冯玉桥，秦孔建，等. 北京市轻型在用车实际道路排放特征分析[J]. 武汉理工大学学报(交通科学与工程版)，2011，35(2)：237.
LI Mengliang, FENG Yuqiao, QIN Kongjian, *et al*. Real-world emission characteristics of in-use light-duty vehicles in Beijing [J]. Journal of Wuhan University of Technology (Transportation Science & Engineering), 2011, 35(2): 237.