基于改进混合蛙跳算法的云工作流负载均衡调度优化

徐 俊 项倩红 肖 刚
(浙江工业大学计算机科学与技术学院 杭州 310023)

摘 要 在实例密集型和开放的云环境下,工作流调度通常面临着廉价和优质资源被频繁调用的问题,导致调度效率低下,云环境稳定性遭到破坏。此外,区别于一般的任务调度,工作流任务之间通常具有关联依赖性,极大地提高了任务分配的复杂度。针对目前大多数云工作流调度中存在虚拟机间负载不均衡的现象,首先提出一种工作流分层调度模型,按任务优先级进行层级划分,将优先级相近且相互独立的任务置于同一层级,通过分层执行任务来有效缓解虚拟机的负载压力。其次,基于混合蛙跳算法进行改进,采用时间贪心算法来优化初始种群,以提高搜索效率;并增加对局部最优个体的重建策略来跳出局部最优,增强全局搜索能力。最后,将改进后的混合蛙跳算法(ISFLA)应用于云工作流调度,通过 WorkflowSim 仿真平台来模拟工作流调度的真实场景,并将改进后的混合蛙跳算法与传统的混合蛙跳算法及粒子群算法进行对比,从负载均衡度、工作流整体完成时间和搜索效率 3 个方面进行评价。实验结果表明,在迭代相同次数后,ISFLA 的负载均衡度最优,并且随着任务数的增加,其值最先趋于稳定;同时,在工作流整体完成时间上,ISFLA 也显著低于其他算法;在搜索效率方面,由于使用贪心算法提高了初始种群质量,ISFLA 的搜索耗时大幅缩短。

关键词 云工作流,任务调度,负载均衡,局部最优,混合蛙跳
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/jsjkx.181001866

Load Balancing Scheduling Optimization of Cloud Workflow Using Improved Shuffled Frog Leaping Algorithm

XU Jun XIANG Qian-hong XIAO Gang
(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract In instance-intensive and open cloud environments, workflow scheduling always suffers from frequent calls of the cheap and high-quality resources, resulting in poor scheduling efficiency and disruption of stability. In addition, unlike general task scheduling, workflow tasks usually have associated dependencies, which greatly increase the complexity of task assignment. Aiming at the imbalance of load between cloud virtual machines, a workflow hierarchical scheduling model was proposed, which is hierarchically divided according to task priorities so as to alleviate virtual machine load pressure. Besides, to optimize the shuffled frog leaping algorithm(ISFLA), the time greedy strategy is applied to initialize population, as a result, improving the search efficiency. Then, by enhancing the position of best solutions locally, a reconstruction strategy was put forward to go out of the dilemma of local optimum. Finally, the experimental results in cloud workflow scheduling show that the improved shuffled frog leaping algorithm can optimize load balance degree and is more effective in task processing as well as searching compared with the traditional shuffled frog leaping algorithm and particle swarm optimization.

Keywords Cloud workflow, Task scheduling, Load balancing, Local optimum, SFLA

1 引言

云工作流指面向云计算的工作流[1],结合云计算和工作流,云平台的开放性使其具有巨大的用户群、负载实时动态改变、与顶层资源紧密结合等特点,通过云基础架构,以“按需付费”的模式为用户提供低成本和高可靠性的资源及服务。云工作流调度即云环境下工作流的资源分配过程[2],用户可借助云的优势降低工作流的运行成本,提高工作效率。然而,在

到稿日期:2018-10-08 返修日期:2019-03-18 本文受国家自然科学基金(61573316),浙江省重大科技专项(2014C01048),浙江省重点研发计划项目(2018C01064)资助。

徐 俊(1979—),男,硕士,高级实验师,主要研究方向为服务计算,E-mail:xujun@zjut.edu.cn;项倩红(1994—),女,硕士生,主要研究方向为云计算、云调度;肖 刚(1965—),男,博士,教授,CCF 会员,主要研究方向为云计算、云制造,E-mail:xg@zjut.edu.cn(通信作者)。

调度过程中,不合理的资源分配方案将直接影响云环境的稳定性和用户满意度。因此,对云工作流进行合理的资源调度具有十分重要的研究意义[3]。

云工作流调度将相互间具有依赖关系的工作流任务映射到虚拟机资源上执行[4]。通常,任务调度已被认为是一个NP-Hard问题[5],而区别于一般的任务调度,工作流中任务间的强关联依赖性大大增加了调度分配的复杂度。随着问题规模的增大,当大量实例密集型任务同时到达时,会造成高效和优质的虚拟资源被频繁地调用,这种负载不均衡现象会带来响应时间过长、资源利用率不高、云环境稳定性遭到破坏等问题。针对该现状,本文研究云工作流的均衡调度问题,通过解决负载不均衡现象来提高调度质量并缩短工作流完成时间。

迄今为止,国内外研究学者已经提出了许多启发式和元启发式算法,并将其应用于云工作流调度。文献[6]提出一种基于动态关键路径的工作流调度算法,通过计算任务图中的关键路径来确定有效映射,为关键路径中的任务分配优先级,通过时间估算使得所有任务在更短的时间内完成;但它的应用场景局限于任务间无强依赖关系的情况,且未考虑任务间传输时间的影响。文献[7]立足于多用户多业务的开放环境,针对虚拟机负载不均衡的问题,提出了一种面向多用户负载感知的选择模型来实现动态负载环境下的有效调度,较好地适应虚拟机负载的变化,从而实现均衡调度;但针对实例密集型任务连续到达的情况,效果欠佳。文献[8]在满足用户截止时间的约束下,针对最小化调度费用问题提出了云工作流调度的粒子群搜索方法,利用关键路径进行粒子初始化和搜索阶段的筛选处理策略,在搜索精度和效率上都有较大提升;但会造成廉价和优质的虚拟资源被频繁调用,从而导致负载失衡。文献[9]针对负载不均衡问题提出了一种模拟蜜蜂采蜜原理的负载均衡策略,基于负载均衡评价模型来缩短队列中任务的排队长度,以达到有效均衡虚拟机负载的目标,平均执行时间和队列中任务的等待时间均显著缩短;但该策略主要运用于平衡非抢占式独立任务,未考虑工作流任务中的优先级关系。文献[10]针对工作流任务运行时不平衡和依赖性不平衡的问题,提出了水平任务平衡方法和垂直任务平衡方法,有效缩短了工作流应用的运行时间。文献[11]考虑任务总完成时间和负载均衡度,提出了一种基于禁忌搜索的负载均衡任务调度优化算法,运用禁忌搜索和贪心原则,在优化完成时间的同时提升负载均衡的性能;但算法的复杂度较高,实际应用下的搜索效率有待验证。

本文将混合蛙跳算法(SFLA)[12]作为基本研究思路。它是一种全新的启发式群体进化算法,由 Eusuff 和 Lansey[12]为解决组合优化问题最先提出。SFLA 结合了模因演算算法和粒子群优化算法的优点,与其他算法相比,具有计算速度快、全局寻优能力强、调整参数少等优点,在解决 NP-hard 问题时有一定的优势,在网络优化、任务调度、流水车间调度、聚类和推荐系统等领域备受青睐。但混合蛙跳算法由随机方法生成初始种群,会造成随机生成的初始解在可行域中分布不均,导致全局寻优能力变弱,并且其只对各子群中最差的青蛙位置进行更新,而未考虑更新最好的青蛙位置,会造成所求的解未必是全局最优,易陷入局部最优的缺陷。文献[13]利用改进混合蛙跳算法求解旅行商问题,在局部搜索过程中引入调整序的思想,并在全局信息交换过程中增加变异操作,相比于遗传算法和粒子群算法,具有更好的搜索性、稳健性。文献[14]运用蛙跳算法来解决多模式资源约束下的项目调度问题,但当搜索规模较大时易陷入局部最优。文献[15]针对云工作流调度问题提出了一种增强混合蛙跳算法,其弥补了易陷入局部最优的缺陷,但未考虑任务间的依赖交互关系。

针对上述问题,本文提出一种改进混合蛙跳算法(ISFLA)来解决云工作流任务的均衡调度问题。结合分层执行任务的思想,我们对基本混合蛙跳算法进行改进,在局部搜索的过程中增加对最优个体的重建操作,弥补了其在搜索规模较大时计算复杂度高、易陷入局部最优的缺陷,同时提高了搜索的精度和效率。

2 云工作流调度

2.1 云工作流

本文主要针对云环境下开放和资源异构[16-17]等特征,研究工作流任务(Tasks)的调度问题。工作流是任务流程的计算模型,其结构表示各个任务协同执行的过程,一个节点代表一个具体可执行的单元任务。通常,云工作流可用有向无环图描述,任务节点和边分别表示工作量和任务间传递数据的大小。在描述之前,先对下文涉及的术语进行归纳,如表1所列。

表1 符号和参数的定义

Table 1 Symbol and parameter definition

符号和参数	定义
T	任务集合
t_i	第 i 个任务
VM	虚拟机集合
h_j	第 j 台主机
vm_{jk}	第 j 台主机的第 k 个虚拟机
l	工作流的第 l 层任务组
MI_i	任务 t_i 的总指令长度
$VMIP_{jk}$	虚拟机 vm_{jk} 的指令执行速度
$data_i$	任务 t_i 的输出数据量
$netwidth_{jk,ql}$	vm_{jk} 到 vm_{ql} 的传输带宽
β	任务优先级
γ	任意单元任务的调度方案
$D_c(t_i)$	任务的计算需求
$D_s(t_i)$	任务的存储需求
η	置信度

2.1.1 云资源

云资源是由云提供商向用户所提供的具有计算和存储能力的服务资源。云中主机可定义为 $H=\{h_1,h_2,\cdots,h_m\}$(m 为主机总数),每个主机由若干个虚拟机组成,则虚拟机集合可定义为:

$$VM=\{\{vm_{11},\cdots,vm_{1m_1}\},\cdots,\{vm_{n1},\cdots,vm_{nm_n}\}\}$$

其中,m_j 表示主机 h_j 中虚拟机的个数。我们用虚拟机的指

令执行速度(VMIPS)和存储空间(VRAM)来描述虚拟机的性能,不同的虚拟机在这两方面的性能表现有所差异。

2.1.2　工作流

工作流由相互间具有依赖关系的单元任务组成,可用有向无环图(Directed Acyclic Graph, DAG)进行形式化描述[16]。$T=\{t_1,t_2,\cdots,t_N\}$表示由 N 个节点组成的任务集合,每个任务都由任务属性、执行时间、输入数据量和输出数据量等信息组成。t_{entry}表示没有父节点的起始任务,t_{exit}表示没有子节点的结束任务。E 表示有向边集合,$E=\{(t_i,t_j,data_{ij})\mid t_i,t_j\in T\}$:$(t_i,t_j,data_{ij})$表示任务 t_i 和 t_j 间具有依赖性,任务 t_i 是 t_j 的前驱任务,$data_{ij}$ 表示任务 t_i 传递给 t_j 的数据量大小。

2.2　云工作流调度建模

云工作流调度问题的实质是建立多个任务与虚拟资源间的映射关系。对于任意任务的调度方案 γ,定义 $\gamma_{i,jk}$ 表示任务 t_i 和虚拟机 vm_{jk} 之间的分配关系,其中 $\gamma_{i,jk}\in\{0,1\}$。任务 t_i 分配在虚拟机 vm_{jk} 上执行时,$\gamma_{i,jk}=1$;否则,$\gamma_{i,jk}=0$。每个任务必定会调度到某一虚拟机上执行,即 $\sum_{j=1}^{m}\sum_{k=1}^{m_j}\gamma_{i,jk}=1$,$j\in\{1,2,\cdots,m\}$,$k\in\{1,2,\cdots,m_j\}$。

为了缓解多个任务同时到达所造成的负载不均衡现象,本文提出一种云工作流分层调度模型,如图 1 所示,该模型由工作流解析、资源感知和任务调度分析 3 部分组成。当用户提交工作流任务后,工作流解析模块将具有关联依赖性的任务进行优先级分层,每个层组由若干个相互独立的单元任务组成,只有当高优先级层中的任务都执行完毕后才可进入下一层任务;资源感知模块中的资源监控器负责实时监控,并选择可用资源;任务调度分析器负责将各层中的单元任务逐一调度到合适的虚拟机上执行,通过均衡虚拟机间的负载缩短工作流完成时间。

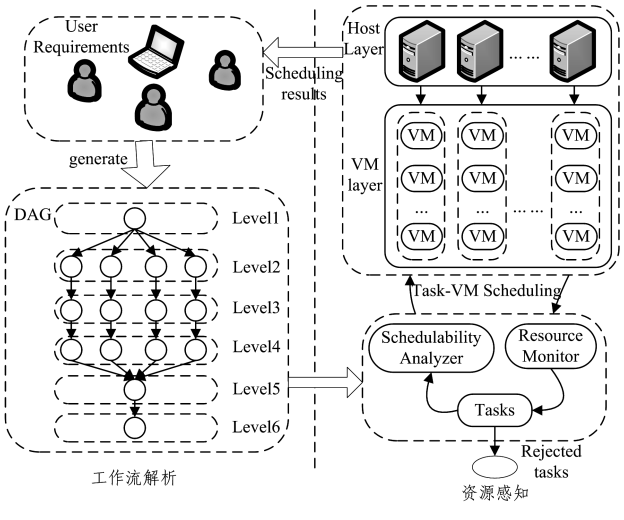


图 1　云工作流调度模型

Fig. 1　Model of workflow scheduling in clouds

2.2.1　工作流完成时间

工作流的完成时间由任务执行时间和传输时间两部分组成。

(1)任务执行时间

假设任务 t_i 的指令长度为 MI_i,虚拟机 vm_{jk} 的指令执行速度为 $VMIPS_{jk}$,则虚拟机 vm_{jk} 执行任务 t_i 的预期完成时间为 $ET(t_i,vm_{jk})$[9]:

$$ET(t_i,vm_{jk})=\frac{MI_i}{VMIPS_{jk}} \tag{1}$$

任务 t_i 的平均执行时间可定义为:

$$\overline{ET(t_i)}=\sum_{j=1}^{m}\sum_{k=1}^{m_j}\frac{ET(t_i,vm_{jk})}{m*m_j} \tag{2}$$

(2)传输时间

任务 t_i 是 t_j 的直接前驱任务,任务 t_i 和 t_j 分别被分配给虚拟机 vm_{jk} 和 vm_{ql} 执行,那么数据从虚拟机 vm_{jk} 到 vm_{ql} 的传输时间可用 $DTT(t_i,t_j)$定义:

$$DTT(t_i,t_j)=\frac{data_i}{netwidth_{jk,ql}} \tag{3}$$

则从 t_i 到 t_j 的平均传输时间为:

$$\overline{DTT(t_i,t_j)}=\frac{data_i}{\overline{netwidth}} \tag{4}$$

(3)工作流完成时间

工作流根据优先级关系分解成 L 层,整个工作流的执行时间由层级内任务的执行时间和层间传输时间两部分组成[17],可定义为:

$$makespan=\sum_{l=1}^{L-1}\overline{DTT(t^l,t^{l+1})}+\sum_{l=1}^{L}\sum_{j=1}^{\lceil\frac{|ET_l|}{x_l}\rceil}\max\bigcup_{a=1+(j-1)x_l}^{x_l\times j}ET(t_a^l) \tag{5}$$

其中,$\sum_{j=1}^{\lceil\frac{|ET_l|}{x_l}\rceil}\max\bigcup_{a=1+(j-1)x_l}^{x_l\times j}ET(t_a^l)$表示第 l 级任务的最长执行时间;$ET(t_a^l)$表示位于第 l 级第 a 个任务的执行时间;L 表示工作流的所有层数;$\sum_{l=1}^{L-1}\overline{DTT(t^l,t^{l+1})}$表示整个工作流层间的传输时间。

2.2.2　负载均衡度

对于虚拟机 vm_{jk},它的负载 LB_{jk} 为分配给它的所有任务的预期完成时间。LB_{jk} 越大,说明虚拟机 vm_{jk} 的负载越高。LB_{jk}^l 定义为:

$$LB_{jk}^l=\sum_{i=1}^{l_n}\gamma_{i,jk}\times ET(t_i,vm_{jk}) \tag{6}$$

对于第 l 层,将 l_n 个相互独立的任务调度到 M 个虚拟机上的平均负载定义为 l_n 个任务的总指令长度与 M 个虚拟机总指令执行速度之商:

$$\overline{LB^l}=\sum_{i=1}^{l_n}MI_i/\sum_{j=1}^{m}\sum_{k=1}^{m_j}VMIPS_{jk} \tag{7}$$

对于任意调度方案 γ,结合式(6)和式(7),给出第 l 层虚拟机的总负载均衡度[9]为:

$$LB_\gamma^l=\sqrt{\frac{1}{M}\times\sum_{j=1}^{m}\sum_{k=1}^{m_j}(LB_{jk}^l-\overline{LB^l})^2} \tag{8}$$

则整个工作流执行时的负载均衡度为:

$$LB_r=\frac{1}{N}\sum_{l=1}^{N}LB_\gamma^l \tag{9}$$

负载均衡度 LB_r 的数值越小,表明云工作流执行过程中各虚拟机之间的负载越均衡。

3 ISFLA 算法概述

混合蛙跳算法(SFLA)主要运用池塘中青蛙寻找食物的行为来有效模拟优化调度的过程[18],由局部搜索和全局信息交换两个关键步骤组成。由于云环境的搜索空间较大,并且随机生成初始种群会造成搜索优解的时间花费较高,且所得到的最好解未必是全局最优,容易陷入局部最优的困境[15]。针对上述缺陷,本文对 SFLA 算法进行改进,提出一种改进混合蛙跳算法(ISFLA)来解决云工作流的均衡调度问题。

3.1 局部搜索

青蛙种群中各族群局部搜索的目的是通过使模因信息在局部个体间传递来优化局部个体,使得搜索迭代一定次数后局部最优趋向于全局最优[13]。因此,如何运用合理的策略选择一定数目的青蛙构成子族群将至关重要。为此,本文设计如下更新策略。

设 X_{lb} 为一个子群中适应度最好的候选解,X_w 为子群中适应度最差的候选解,X_{gb} 为整个种群中适应度最好的候选解。对每个子群进行局部搜索,则子群中适应度最差的青蛙的更新策略[12]为:

$$S=\begin{cases}\min\{\text{int}[\text{rand}(X_{lb}-X_w)],S_{\max}\}, & X_{lb}-X_w\geqslant 0\\ \max\{\text{int}[\text{rand}(X_{lb}-X_w)],-S_{\max}\}, & X_{lb}-X_w<0\end{cases} \tag{10}$$

$$X_w'=X_w+S \tag{11}$$

其中,S 表示青蛙个体的调整矢量,$S_{\max}$ 表示青蛙个体允许改变的最大步长,X_w'表示更新后青蛙的位置。

根据此策略对最差青蛙的位置进行更新,同时为了防止青蛙过早地陷入局部最优,并加快收敛速度,本文还增加了对最好青蛙的位置进行重建的步骤,即不仅要对子群中最差青蛙的位置进行更新,还要调整最优青蛙的位置。最优青蛙位置更新过程的伪代码描述如算法 1 所示。

算法 1 子群内最好青蛙的位置更新

```
Input:常量 a
Output:更新后的 X_lb
1. begin
2. 计算 X_lb的适应度 f_lb;X_gb的适应度 f_gb;X_w 的适应度 f_w
3. 设定一个增量值 s=a * rand()
4. for each dimension d in individual X_lb
5.      new_value()=original_value()+s
6.      重新计算 X_lb的适应度 f_n
7.      if fitness f_n is not better than f_lb then
8.          new_value(d)=original_value(d)
9.      end if
10.     next d
11. end for
12. return frog X_lb
13. end
```

3.2 全局信息交换

全局信息交换有助于收集各子群搜索到的局部信息,通过模因在全局中传递获得新的全局最优解的方向。当所有子群经过一定次数的局部搜索后,将子群中的青蛙个体混合在一起,然后按适应度降序排列后重新划分子群,青蛙个体间的模因信息得到充分传递后继续进行局部搜索,如此反复,直到满足收敛条件为止(收敛到最优解或达到最大进化代数)。

根据混合蛙跳算法的局部搜索和全局信息交换过程,增加对局部最优个体的更新策略。改进后的混合蛙跳算法(ISFLA)的伪代码如算法 2 所示。

算法 2 ISFLA 算法

```
Input:常量 a;子族群的个数 m
Output:全局最优解
1. begin
2. 设置青蛙的维度 d
3. 初始化种群 P,计算每只青蛙的适应度
4. do
5.  将种群 P 按每只青蛙的适应度降序排列,得出 X_gb
6.  将种群 P 划分为 m 个子族群
7.  for each 子族群
8.      计算 X_lb, X_w
9.      根据算法 1 更新 X_lb, X_w
10.  end for
11.  混合更新后的青蛙个体
12. while(不满足终止条件)
13. end
```

4 基于 ISFLA 的云工作流均衡调度

4.1 任务优先级分层

将上述改进的混合蛙跳算法应用于云工作流调度,在调度前先采用分层调度模型对工作流中的任务进行优先级分层。

任务 t_i 的向上优先级表示 t_i 到终止任务 t_{exit} 的最长距离,即从 t_{exit} 开始,通过向上遍历任务图递归地计算每个任务的向上优先级,定义为 $\beta_{\text{up}}(t_i)$:

$$\beta_{\text{up}}(t_i)=\max_{t_j\in Suc(t_i)}(\overline{DTT(t_i,t_j)}+\overline{ET(t_i)}+\beta_{\text{up}}(t_j)) \tag{12}$$

其中,$Suc(t_i)$为 t_i 的直接后继任务集合。

任务优先级分层的伪代码如算法 3 所示。

算法 3 任务优先级分组算法

```
Input:工作流中的任务列表 G
Output:分层后的任务列表 G_k
1. begin
2. 扫描任务列表 G
3. if G is Executable
4.  then accept G
5. end if
6. 根据式(1)、式(2)计算每个任务 t_i 的平均执行时间 ET(t_i)
7. 根据式(3)、式(4)计算任务 t_i,t_j 所对应虚拟机间传输的平均耗时 DTT(t_i,t_j)
8. 根据式(12),从终止任务 t_exit开始向上遍历任务图,计算每个任务的向上优先级 β_up
9. 根据 β_up倒序排列所有任务
10. k=1;G_k={ };add t_entry to G_k
```

11. for (each t_i in a descending order of β_{up})
12. if (($\exists t_j \in G_k$) && (t_i depends on t_j))
13. then k++;G_k={}; //新建任务层 G_k
14. add t_i to G_k;
15. end for
16. end

当工作流到达时,先对其所有任务进行扫描,接受可执行工作流;拒绝不可执行工作流(第 2—5 行)。按向上优先级和依赖关系对任务进行分层(式(12)),处于同一层中的任务相互独立。对于任务 t_i,若它与层 G_k 中的其他任务无关,则将其添加到层组 G_k 中;否则,创建一个新层 G_{k+1},并将 t_i 加入其中(第 6—15 行),以保证同一优先级层组中的任务相互独立。

4.2　基于优先级的云工作流调度

根据本文所提的问题场景和数学模型,求解负载均衡度最优的工作流调度方案。基于工作流任务完成时间(2.2.1 节)和负载均衡度(2.2.2 节)的定义,本文提出一种基于 ISFLA 的工作流任务均衡调度算法,其设计流程如图 2 所示。

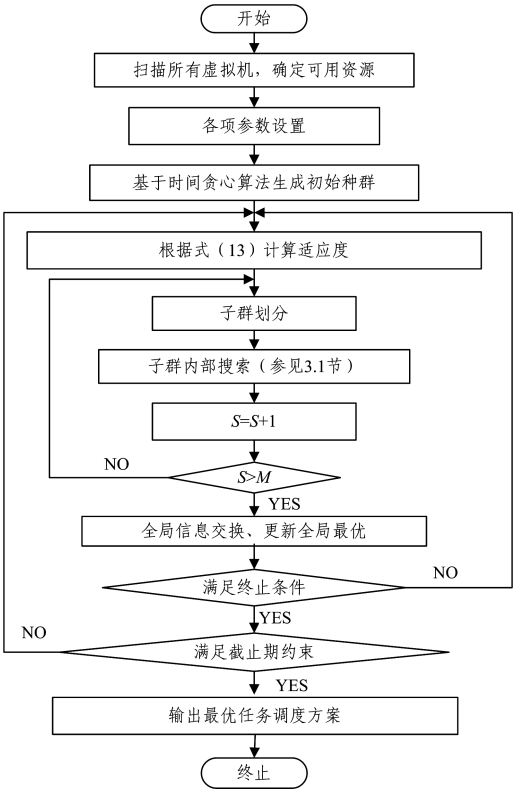


图 2　基于 ISFLA 的云工作流调度算法的流程图

Fig. 2　Workflow scheduling flow chart based on ISFLA

(1)调度方案编码

将具有 N 个单元任务的工作流分配到 M 个虚拟资源上,每只青蛙的位置代表着一种任务分配方案。第 k 只青蛙的位置可用 N 维向量 $\boldsymbol{F}(k)$ 表示,即 $\boldsymbol{F}(k)=(M_k^1, M_k^2, \cdots, M_k^N)$。青蛙的维数代表工作流的任务数 N,虚拟机的数量 M 决定了青蛙被允许移动的搜索空间。因此,每个维度的坐标值可在 0～M 之间。图 3 表示 8 个任务的编码方案,假设每个任务可被分配到 5 台虚拟机上,则青蛙在每个维度上可搜索的空间在 0～5 之间,对坐标值向上取整即可得到任务最终被分配到的虚拟机,例如第 7 维的坐标值为 2.2,向上取整为 3,表示第 7 个任务被分配到虚拟机 vm_3 上。

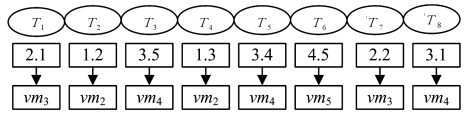


图 3　编码方案

Fig. 3　Coding scheme

(2)种群初始化

针对云工作流调度存在搜索规模大、任务协调困难,以及运用随机法生成初始种群会带来算法搜索时间长、效果欠佳的问题,本文采用基于时间贪心的方法生成初始种群[19]来缩短搜索时间,提高搜索效率。

基于时间贪心算法,将任务 t_i 的指令长度 MI_i 进行降序排列,得到集合 C_{MI},将虚拟机 vm_{jk} 的指令执行速度 $VMIPS_{jk}$ 进行升序排列,得到集合 C_{VMIPS},根据 C_{MI} 中的每个元素与 C_{VMIPS} 中所有元素的商 $ET(t_i, vm_{jk})$ 生成预期完成时间的二维矩阵 $\boldsymbol{Mtx}_{time}$。从行号 $i=1$ 开始,每次都将任务 t_i 自矩阵最后一列对应的虚拟机往前分配,依次计算分配给每个虚拟机的负载 LB_{jk} 与预期执行时间 $ET(t_i, vm_{jk})$ 加权后的值 Eva,将 Eva 最优的虚拟机分配给任务 t_i,若存在多个 Eva 相等的情况,则选择执行任务数最少的虚拟机,直到所有任务分配完成。通过调整计算 Eva 时虚拟机负载 LB_{jk} 和预期执行时间 $ET(t_i, vm_{jk})$ 的权重值,重复执行上述方法,直到生成指定数量的初始解。

(3)适应度函数计算

本文的研究目标是云工作流调度的负载均衡最小化。根据负载均衡度的定义(2.2.2 节),将适应度函数定义为:

$$f=\frac{L}{\sum_{l=1}^{L} LB_\gamma^l} \tag{13}$$

即负载均衡度越小,适应度值越大。

根据本文问题模型,给出云工作流负载均衡调度算法的伪代码,如算法 4 所示,青蛙子群的划分、局部信息搜索以及全局信息交换的具体实现参考第 3 节。

算法 4　基于 ISFLA 的云工作流调度算法

Input:子族群的个数 m;
Output:工作流任务的最佳分配方案
1. begin
2. 扫描所有虚拟机,排除无效虚拟机
3. 多次运用时间贪心算法生成初始种群 P
4. 根据式(13)计算适应度函数 f
5. do
6.　将种群 P 按照每只青蛙的适应度降序排列
7.　计算 X_{gb} 的适应度 f_{gb};X_w 的适应度 f_w
8.　将种群 P 划分为 m 个子族群
9.　for each 子族群
10.　执行算法 2
11. end for
12. 混合更新后的子族群

13. while(不满足终止条件)

14. return 最佳分配方案

15. end

5 实验分析

5.1 实验环境

为有效地模拟云工作流任务调度，本文在 WorkflowSim[20] 平台上进行模拟仿真实验。WorkflowSim 通过对现有的 CloudSim 模拟器进行扩展，能有效模拟资源异构的云环境[21]。为保证对比的公平性，将实验环境统一为：Mac OSX 10.11.3 EI Capitan 系统、主频 2.7 Hz、内存 8 GB，编程及调试环境为 MyEclipse2014。为排除偶然性，每种方法独立重复实验 30 次，并取平均值作为最终实验结果。本实验在工作流和虚拟资源方面的环境配置如下。

5.1.1 工作流

借助重力物理学领域的 LIGO[16] 工作流为实验对象，其结构模型如图 4 所示。实验中，工作流的任务数为 500～4000 的随机数，每个任务的指令长度随机产生，取值范围为[500,2000]。

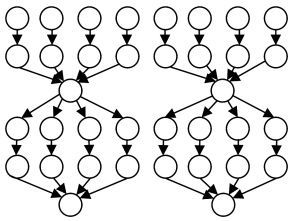


图 4 LIGO 工作流结构

Fig. 4 Structure of LIGO workflow

5.1.2 虚拟资源

本文构建了一个具有 6 个主机的模拟环境，每个主机由若干个虚拟机组成，其中虚拟机的数量服从均值为 6 的正态分布。参考 Google Cloud[22] 的分配机制，虚拟机的计算速度在 500～1500 之间随机选取，虚拟机之间的带宽为 500～1.5 之间的随机数，相同主机的虚拟机间的带宽可视为∞；并且虚拟机参数与主机间应满足如下条件：

$$\sum_{k=1}^{m_j} VMIPS_{jk} \leqslant MIPS_j \tag{14}$$

$$\sum_{k=1}^{m_j} VRAM_{jk} \leqslant RAM_j \tag{15}$$

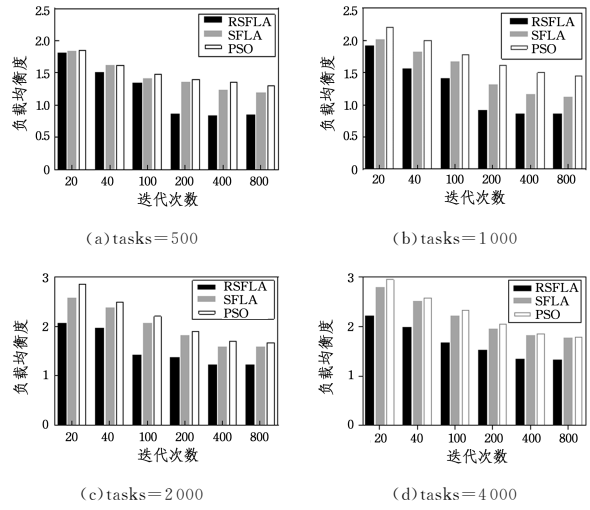
即同一主机上所有虚拟机属性的总和应小于主机。

5.2 仿真实验结果分析

为验证本文所提调度模型和算法的有效性，在上述实验环境中，将 ISFLA 与 SFLA[15,23] 和 PSO[24] 进行对比，采用负载均衡度[25]、工作流整体完成时间[26] 和搜索效率[27] 作为评价指标。

5.2.1 负载均衡度

本节实验通过改变迭代次数、任务数来探索采用 3 种不同算法时负载均衡度的变化，结果如图 5、图 6 所示。

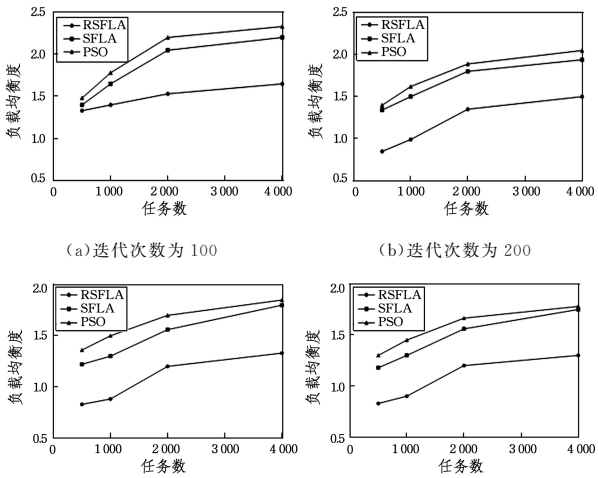


(a)tasks=500 (b)tasks=1000

(c)tasks=2000 (d)tasks=4000

图 5 迭代次数对负载均衡度的影响

Fig. 5 Effect of number of iterations on load balancing



(a)迭代次数为 100 (b)迭代次数为 200

(c)迭代次数为 400 (d)迭代次数为 800

图 6 任务数对负载均衡度的影响

Fig. 6 Effect of number of tasks on load balancing

从迭代次数的角度进行对比，将任务数分别设置为 500，1000，2000，4000，探索迭代次数对负载均衡度的影响。从图 5 可以看出，当任务数相同时，随着迭代次数的增加，负载均衡度呈下降趋势。当迭代次数在 200 左右时，ISFLA 算法就基本趋于稳定，即负载均衡度不再随迭代次数的增加而减小；而其他两种算法在迭代次数为 400，甚至 800 时，才逐渐趋于稳定。这是由于 ISFLA 算法在生成初始解的阶段通过运用基于时间贪心的算法，有效排除了表现较差的个体，缩小了搜索范围；此外，ISFLA 算法对局部最优个体的重建策略可有效优化调度结果，减少最大迭代次数。

从图 6 可以看出，当迭代次数分别设置为 100，200，400 和 800，固定迭代次数不变，任务数按 500，1000，2000，4000 变化时，采用 ISFLA 算法求得的负载均衡度明显低于其他两种算法；并且随着任务数的增加，采用 ISFLA 算法求得的实验结果更趋向于稳定，即负载均衡度没有出现大幅增加。这说明对于任务数较多的场景，ISFLA 算法的调度结果更优，能够更有效地均衡虚拟机的负载。

5.2.2　工作流完成时间

工作流完成时间的长短可以从侧面反映出任务调度的负载均衡情况,负载均衡度越高,说明在部分虚拟机上等待的队列越长,从而表现为工作流整体完成时间延长。本节实验保持迭代次数不变,通过改变任务数来对比采用不同算法时工作流完成时间的变化情况,实验结果如图 7 所示。

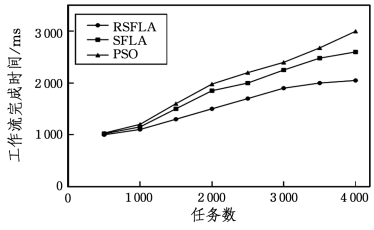


图 7　工作流任务数对完成时间的影响

Fig. 7　Impact of the number of workflow tasks on the makespan

从图 7 中可以直观地看出,运用 ISFLA 算法求解所得调度方案的工作流完成时间要比其他算法短。在任务数少于 1000 时,虚拟资源相对充沛,不存在超负载现象,因此运用 3 种算法求得的时间差异很小。但随着任务数的增加,超负载现象越来越严重,采用 ISFLA 算法所得调度方案的工作流任务完成时间要显著少于其他两种算法。这是由于 ISFLA 算法采用分层调度模型执行任务,有效地均衡了虚拟机间的负载,降低了负载均衡度,从而缩短了工作流的整体完成时间。

5.2.3　搜索效率

在种群初始化阶段生成的初始种群的质量将影响 ISFLA 在后续局部搜索阶段中的搜索时间,进而影响 ISFLA 算法的全局搜索效率。为了验证运用时间贪心算法生成的初始种群的质量优于采用随机算法生成的初始种群,本节实验将对比以上两种策略生成初始种群后 ISFLA 算法在局部搜索阶段所需搜索迭代时间与任务数的关系,实验结果如图 8 所示。

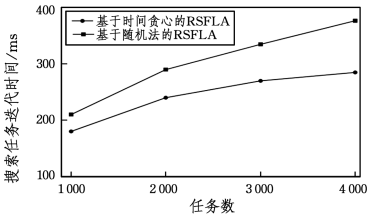


图 8　采用两种初始化方法的局部搜索效率的比较

Fig. 8　Comparison of search efficiency between two initialization methods

从图 8 中可以看出,当迭代次数相同时,随着工作流任务数的增加,两种方法在局部搜索阶段所需的搜索迭代时间延长,但基于时间贪心算法的 ISFLA 的搜索时间增速更趋于平缓。纵向对比来看,当任务数相同时,运用贪心策略生成初始种群后,ISFLA 所需的搜索迭代时间远短于随机方法。这是由于贪心策略生成的初始种群的质量更优,可以有效排除一些表现较差的个体,从而缩短了搜索迭代时间,加快了收敛速度。

结束语　针对云环境下开放和资源异构的特征,本文对云工作流任务的均衡调度进行研究。首先,提出了一种基于任务优先级的分层调度模型,通过分层执行任务有效减轻了虚拟机负载并缩短了工作流完成时间。然后,针对具体的调度过程,提出了一种基于改进混合蛙跳的云工作流任务负载均衡调度算法,算法改进具体表现在增加对局部最优个体的重建操作,弥补了传统方法易陷入局部最优的缺陷。其次,运用时间贪心算法替代随机方法生成初始种群,有效排除了较差的个体,从而提高了搜索迭代的效率,加快了收敛速度。最后,通过 LIGO 工作流实例在 WorkflowSim 平台上进行模拟仿真实验,与混合蛙跳算法、粒子群算法进行对比实验,结果表明,本文所提调度模型和改进的 ISFLA 算法在解决云工作流均衡调度的问题上更有优势。但本文主要关注的是负载均衡度以及工作流完成时间,未考虑用户其他 QoS 因素对工作流任务调度的影响,这些问题将在下一步的工作中进行研究。为了实现云工作流的均衡调度,本文所提的云工作流负载均衡调度优化方法具有一定的时间开销,因此,时间成本也是下一步重点考虑的因素。另外,所提的云工作流负载均衡调度优化方法是基于现有的调度框架实现的,虽已获得较大的成效,但仍有提升的空间。

参 考 文 献

[1] CHAI X Z, CAO J. Cloud Computing Oriented Workflow Technology[J]. Journal of Chinese Computer Systems, 2012, 33(1): 90-95. (in Chinese)
柴学智,曹健. 面向云计算的工作流技术[J]. 小型微型计算机系统,2012,33(1):90-95.

[2] ZHU Z, ZHANG G, LI M, et al. Evolutionary Multi-Objective Workflow Scheduling in Cloud[J]. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(5): 1344-1357.

[3] CHEN H K, ZHU J H, ZHU X M, et al. Resource-Delay-Aware Scheduling for Real-Time Tasks in Clouds[J]. Journal of Software, 2017, 54(2): 446-456. (in Chinese)
陈黄科,祝江汉,朱晓敏,等. 云计算中资源延迟感知的实时任务调度方法[J]. 软件学报,2017,54(2):446-456.

[4] ZHENG H S, YU D J, ZHANG L. Multi-QoS Cloud Workflow Scheduling Based on Firefly Algorithm and Dynamic Priorities [J]. Computer Integrated Manufacturing Systems, 2017, 23(5): 963-971. (in Chinese)
郑宏升,俞东进,张蕾. 基于萤火虫算法和动态优先级的多 QoS 云工作流调度[J]. 计算机集成制造系统,2017,23(5):963-971.

[5] ULLMAN J D. NP-complete scheduling problems [M]. Academic Press, Inc. 1975.

[6] RAHMAN M, VENUGOPAL S, BUYYA R. A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids[C]// IEEE International Conference on E-Science and Grid Computing. IEEE Computer Society, 2007: 35-42.

[7] ZHU Y, LI W, LUO J Z. Multi-User Oriented Load-Aware Dy-

namic Service Selection Model[J]. Journal of Software, 2014, 25(6):1196-1211. (in Chinese)
朱勇,李伟,罗军舟. 一种面向多用户的负载感知动态服务选择模型[J]. 软件学报,2014,25(6):1196-1211.

[8] CHEN W, DA S R F, DEELMAN E, et al. Using imbalance metrics to optimize task clustering in scientific workflow executions[J]. Future Generation Computer Systems, 2014, 46(1): 69-84.

[9] DHINESH B L D, KRISHNA P V. Honey bee behavior inspired load balancing of tasks in cloud computing environments[J]. Applied Soft Computing Journal, 2013, 13(5): 2292-2303.

[10] TAWFEEK M A, EL-SISI A, KESHK A E, et al. Cloud task scheduling based on ant colony optimization[C]// International Conference on Computer Engineering & Systems. IEEE, 2014: 64-69.

[11] SUN L Y, LING M, ZHU P, et al. Load balancing Task Scheduling Algorithm Based on Tabu Search in Cloud Computing[J]. Journal of Chinese Computer Systems, 2015, 36(9): 1948-1952. (in Chinese)
孙凌宇,冷明,朱平,等. 云计算环境下基于禁忌搜索的负载均衡任务调度优化算法[J]. 小型微型计算机系统,2015,36(9):1948-1952.

[12] EUSUFF M, LANSEY K, PASHA F. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization[J]. Engineering Optimization, 2006, 38(2): 129-154.

[13] LUO X H, YANG Y, LI X. Modified shuffled frog-leaping algorithm to solve traveling salesman problem[J]. Journal on Communications, 2009, 30(7): 130-135. (in Chinese)
罗雪晖,杨烨,李霞. 改进混合蛙跳算法求解旅行商问题[J]. 通信学报,2009,30(7):130-135.

[14] WANG L, FANG C. An effective shuffled frog - leaping algorithm for multi-mode resource-constrained project scheduling problem[J]. Information Sciences, 2011, 181(20): 4804-4822.

[15] KAUR P, MEHTA S. Resource provisioning and workflow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm[M]. Academic Press, 2017.

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

[17] THANT P T, POWELL C, SCHLUETER M, et al. A Level-Wise Load Balanced Scientific Workflow Execution Optimization Using NSGA-II[C]// IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. ACM, 2017: 882-889.

[18] EUSUFF M M, LANSEY K E. Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm[J]. Journal of Water Resources Planning & Management, 2003, 129(3): 210-225.

[19] CHEN H, WANG F, NA H, et al. User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing [C]// Parallel Computing Technologies. IEEE, 2013: 1-8.

[20] CHEN W, DEELMAN E. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments[C]// IEEE, International Conference on E-Science. IEEE, 2013: 1-8.

[21] ACCORSI R, STOCKER T. Discovering Workflow Changes with Time-Based Trace Clustering[M]// Data-Driven Process Discovery and Analysis. Berlin: Springer, 2017: 154-168.

[22] ALAM M, SHAKIL K A, SETHI S. Analysis and clustering of workload in google cluster trace based on resource usage[C]// Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES). IEEE, 2016: 740-747.

[23] DENG Y, CHENG X H. A heterogeneous multiprocessor task scheduling algorithm based on SFLA[C]// World Automation Congress. IEEE, 2016: 1-5.

[24] CHEN W N, ZHANG J. A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints[C]// IEEE International Conference on Systems, Man, and Cybernetics. IEEE, 2012: 773-778.

[25] TAO X L, WEI Y, WANG Y. A Load Balancing Method Based on Hierarchy and Multi-agent for Cloud Computing Platform [J]. Acta Electronica Sinica, 2016, 44(9): 1068-1077. (in Chinese)
陶晓铃,韦毅,王勇. 一种基于分层多代理的云计算负载均衡方法[J]. 电子学报,2016,44(9):1068-1077.

[26] WANG L. Research and implementation of task scheduling algorithm in cloud environment[D]. Chengdu: University of Electronic Science and Technology of China, 2016. (in Chinese)
王玲. 云计算下任务调度算法的研究与实现[D]. 成都:成都电子科技大学,2016.

[27] WANG Y W, GUO Y F, LIU W Y, et al. A Task Scheduling Method for Cloud Workflow Security[J]. Joural Computer Research and Development, 2018, 55(6): 66-75. (in Chinese)
王亚文,郭云飞,刘文彦,等. 面向云工作流安全的任务调度方法[J]. 计算机研究与发展,2018,55(6):66-75.