

基于粒子群优化的云 workflow 任务调度

薛 凡^{1,2}, 吴志健¹

(1 武汉大学 计算机学院软件工程国家重点实验室, 湖北 武汉 430072;

2 黄淮学院 创新创业学院, 河南 驻马店 463000)

摘 要: 为了解决云环境中科学 workflow 调度的代价优化问题, 提出一种基于改进粒子群优化模型的科学 workflow 调度算法 CWS_PSO. 该算法将科学 workflow 调度问题形式化为截止时间约束下的调度代价最优化问题, 利用粒子群模型建立了粒子位置与任务调度方案的编码与转换模型, 并通过粒子的迭代进化寻找满足截止时间约束的代价最小化调度方案. 通过科学 workflow 模型的仿真实验, 与同类型调度算法进行了性能比较. 实验结果表明, CWS_PSO 算法不仅可以有效均衡 workflow 的执行跨度与执行代价, 而且能以更高的截止时间约束满意度实现 workflow 执行代价的最小化.

关键词: 云计算; workflow 调度; 粒子群算法; 代价最优化; 约束满意度

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2018)08-0122-06

Cloud Workflow Tasks Scheduling Based on Particle Swarm Optimization

XUE Fan^{1,2}, WU Zhi-jian¹

(1 State Key Lab of Software Engineering, Computer School, Wuhan University, Wuhan 430072, China;

2 International Education College, Huanghuai University, Zhumadian 463000, China)

Abstract: For solving the cost optimization of the scientific workflow scheduling in cloud environment, a scientific workflow scheduling algorithm CWS_PSO based on improved particle swarm optimization (PSO) model is presented in this paper. Our algorithm formalizes the scientific workflow scheduling problem as the scheduling cost optimization problem with the deadline constraint. The encoding and transfer model are constructed between the particle's position and tasks scheduling scheme by PSO model, and searching for the cost optimization scheme satisfying the deadline constraint is processed by the particle's iteration evolution. Through the simulation experimental of scientific workflows model, the performance comparisons are conducted in the same kind of algorithms. Experimental results show that our algorithm could efficiently achieve the trade-off between the workflow execution makespan and the execution cost, and achieve the execution cost minimization with the deadline constraint satisfaction.

Key words: cloud computing; workflow scheduling; particle swarm optimization; cost optimization; constaint satisfaction

1 引言

workflow 调度普遍应用于大规模科学问题和工业控制问题的建模领域中^[1], 其任务执行有着巨大的

数据和计算需求, 不仅任务执行序列间有严格依赖关系, 且执行时间较长, 对高性能计算环境需求更高. 云计算作为最新兴的分布式计算环境, 为科学 workflow 的调度与执行带来了更有效的平台与技术支

撑^[2,3]. 本文中,笔者将重点关注 IaaS 云环境中科学 workflow 调度优化问题.

相关研究中,文献[4]提出一种前向遗传算法,通过可靠性驱动的信誉值评估资源可靠性,实现执行跨度与执行可靠性的多目标优化.文献[5]提出一种动态规划云 workflow 调度算法,有效解决了资源价格变化时任务调度的开销问题.文献[6]考虑不同价格和类型的 VMs,基于定价模型设计了一种 workflow 调度代价最小化算法,尽管可以降低总代价,但结果并非近似最优.文献[7]提出满足期限和预算约束的 workflow 调度算法,目标是最大化任务执行数量,但算法仅考虑了单一 VM 提供环境,忽略了云异构特征.文献[8]与本文较为类似,均研究了 IaaS 云中单 workflow 的调度问题,文献算法基于关键路径,同时考虑了 VM 异构、即付即用和区间定价机制,实现了执行代价最小化的任务调度算法.但是,该算法得到的解并不是全局最优,而仅是在任务层次上的最优,忽略了整个 workflow 的结构和特征.

本文将设计时间复杂度更低的工作流调度算法,满足截止时间同时寻找调度可行解.

2 问题描述

表 1 首先给出相关参数符号及其含义说明.

表 1 符号说明

符号	参数含义
$W = (T, E)$	工作流 DAG 模型
$T = \{t_1, t_2, \dots, t_n\}$	工作流任务集合
E	工作流有向边集合
d_w	用户定义的工作流截止时间
P_{VM_i}	VM_i 的处理能力
C_{VM_i}	VM_i 的单位时间代价
deg_{VM_i}	VM_i 的性能下降比例
$ET_{t_i}^{VM_j}$	任务 t_i 在 VM_j 上的执行时间
I_{t_i}	任务 t_i 的大小
$TT_{t_i t_j}$	父任务 t_i 与子任务 t_j 间的数据传输时间
β	网络带宽
$d_{t_i}^{out}$	任务 t_i 的输出数据大小
$PT_{t_i}^{VM_i}$	任务 t_i 在 VM_i 的运行时间
$R = \{r_1, r_2, \dots, r_n\}$	租用的 VMs 集合
LST_{r_i}	资源 r_i 的租用开始时间
TEC	工作流总执行代价
TET	工作流总执行时间
ST_{t_i}	任务 t_i 的开始时间
ET_{t_i}	任务 t_i 的结束时间

2.1 任务与资源模型

将工作流 $W = (T, E)$ 建立为有向无循环图 DAG, $T = \{t_1, t_2, \dots, t_n\}$ 为任务集, E 为有向边集. 若任务对 (t_i, t_j) 间存在边 t_{ij} , 则表示任务 t_i 与 t_j 间存在数据依赖性, t_i 称为 t_j 的父任务, t_j 称为 t_i 的子任务, 换言之, 子任务在其父任务执行之前不能开始执行. 一种简单的工作流模型如图 1 所示. 每个工作流 W 拥有一个完成截止时间约束 d_w , 限定了所有工作流任务的完成时间期限.

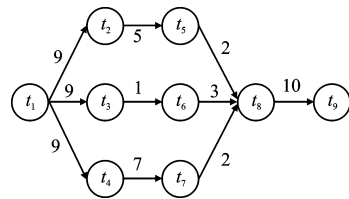


图 1 一种工作流模型

IaaS 云中, 一种 VM 类型 VM_i 可根据能力 P_{VM_i} 和使用代价 C_{VM_i} 定义. 假设对于每一类 VM, 处理能力根据资源的可用 FLOPS(每秒浮点运算操作数)进行定义, 该信息在笔者设计的算法中用来计算给定 VM 上任务的执行时间, deg_{VM_i} 表示 VM 的性能下降比例, VM 的性能变化通过调整租用 VM 的处理能力建立模型. IaaS 云按照单位时间 τ 提供即付即用的资源使用模式, 且任意部分时间的 VM 租用均按一个单位时间收取费用.

令 $ET_{t_i}^{VM_j}$ 表示任务 t_i 在 VM_j 上的执行时间, 由根据浮点运算数 FLOP 定义的任务大小 I_{t_i} 进行计算, 即式(1). 令 $TT_{t_i t_j}$ 表示父任务 t_i 与子任务 t_j 间的数据传输时间, 由式(2)计算, β 为网络带宽. 为了计算 $TT_{t_i t_j}$, 假设 t_i 的输出数据大小 $d_{t_i}^{out}$ 已知, 且工作流运行在同一数据中心, 即 VM 间的带宽相同. 运行在同一 VM 上的任务间的传输时间设为 0. 任务 t_i 在 VM 上的总处理时间 $PT_{t_i}^{VM_j}$ 由公式(3)计算, 其中, k 表示 t_i 作为父任务时与其相连的边的数量, 当 t_i 与 t_j 运行于同一 VM 时, $s_k = 0$, 否则 $s_k = 1$.

$$ET_{t_i}^{VM_j} = I_{t_i} / P_{VM_j} \times (1 - deg_{vm_j}) \quad (1)$$

$$TT_{t_i t_j} = d_{t_i}^{out} / \beta \quad (2)$$

$$PT_{t_i}^{VM_j} = ET_{t_i}^{VM_j} + \sum_1^k TT_{t_i t_j} \times s_k \quad (3)$$

2.2 问题定义

工作流调度最优化问题, 即寻找一种调度方案, 在满足截止时间约束同时, 任务执行总代价达到最小. 令 $S = (R, M, TEC, TET)$ 表示调度方案, R 为资源集, M 为任务与资源间的映射, TEC 为总执行代价, TET 为总执行时间. 图 2 是图 1 所示工作流的

一种调度方案,由于仅需保证子任务在其父任务之后开始执行,调度方案并不唯一。 $\mathbf{R} = \{r_1, r_2, \dots, r_n\}$ 为租用 VMs 集合,每个资源 r_i 拥有一种 VM 类型 VM_{r_i} ,其属性包括租用开始时间 LST_{r_i} 和租用结束时间 LET_{r_i} , M 为一种映射,以四元组形式表示为 $m_{ij}^q = (t_i, r_j, ST_{ij}, ET_{ij})$,映射 m_{ij}^q 可理解为:任务 t_i 被调度到资源 r_j 上运行,其开始执行时间为 ST_{ij} ,完成时间为 ET_{ij} . 公式(4)和(5)分别计算总执行代价 TEC 和总执行时间 TET:

$$TEC = \sum_{i=1}^{|R|} C_{VM_{r_i}} \times [(LET_{r_i} - LST_{r_i}) / \tau] \quad (4)$$

$$TET = \max\{ET_{t_i} : t_i \in T\} \quad (5)$$

则 workflow 调度最优化问题可形式化为:

$$\begin{aligned} \min & TEC \\ s. t. & TET \leq d_w \end{aligned} \quad (6)$$

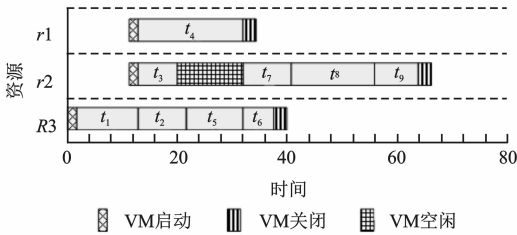


图2 workflow 调度示例

3 粒子群优化

粒子群优化 PSO^[9] 是一种基于动物群体行为的随机化进化方法. PSO 中,一个粒子表示一个个体,可在问题空间中移动,同时代表一个候选解.在给定时间点,粒子运动通过矢量速度定义.每个粒子由位置与速度表示, $pbest$ 表示粒子找到的历史最优解, $gbest$ 表示整个种群找到的历史最优解.粒子拥有适应度,算法迭代中,粒子通过速度和位置的改变朝着 $pbest$ 和 $gbest$ 位置进行进化,直到满足预定的终止条件.算法 1 是 PSO 的伪代码,粒子位置与速度通过下式更新:

$$x_i(t+1) = x_i(t) + v_i(t) \quad (7)$$

$$\begin{aligned} v_i(t+1) = & \omega v_i(t) + c_1 r_1 (x_i^*(t) - x_i(t)) + \\ & c_2 r_2 (x^*(t) - x_i(t)) \end{aligned} \quad (8)$$

式中, ω 为惯性权重; c_i 为加速度因子; $i = 1, 2, r_i$ 为随机数; $i = 1, 2$, 且 $r_i \in [0, 1]$; x_i^* 为粒子 i 的最优位置; x^* 为种群中最优粒子的位置; x_i 为粒子的当前位置. 算法 1 给出 PSO 算法的具体过程.

算法 1 PSO 算法

1. set the dimensions of the particles to d
2. initialize randomly the population of particles
3. while stopping criterion is not met do
4. for each particle do
5. calculate fitness value f
6. if f better than $pbest$ then
7. set $pbest$ to the current fitness value and location
8. end if
9. if f better than $gbest$ then
10. set $gbest$ to the current fitness value and location
11. end if
12. update the position and velocity of the particle
13. end for
14. end while

式(8)中, ω 影响算法收敛性,决定粒子之前的速度对当前速度的影响,定义了粒子局部认知成分与全局群体经验的权衡.其值越大,粒子速度增加越快,更倾向于全局搜索;其值越小,粒子速度会下降,更倾向于局部搜索. c_1, c_2 对收敛性并无关键影响,但恰当取值可防止算法陷入局部最优. c_1, r_1 定义了先前最优位置出现次数,故 c_1 可视为认知参数, c_2, r_2 定义了粒子相对其它邻居的行为,故 c_2 可视为种群参数.

4 CWS_PSO 算法设计

4.1 问题建模

通过 PSO 求解问题包括两个步骤:一是定义问题编码方式,即种群表示;二是定义粒子优劣度量方式,即适应度函数.为了定义编码方式,需要建立粒子的含义和维度.本文的调度场景中,一个粒子表示一个 workflow 及其任务,因此,粒子的维度等同于 workflow 任务数量.粒子维度决定了用来定义空间中粒子的坐标系,如:一个二维粒子的位置可以二维坐标标明,三维粒子的位置由三维坐标标明.图 3 中,粒子代表拥有 9 个任务的 workflow,粒子维度为 9,位置由 9 个坐标决定,即 1~9.

Coord.1	Coord.2	Coord.3	Coord.4	Coord.5	Coord.6	Coord.7	Coord.8	Coord.9
1.2	1.0	2.1	3.0	1.2	1.1	2.0	2.2	2.3

任务与资源映射

$t_1 \rightarrow r_1$	$t_2 \rightarrow r_1$	$t_3 \rightarrow r_2$	$t_4 \rightarrow r_3$	$t_5 \rightarrow r_1$	$t_6 \rightarrow r_1$	$t_7 \rightarrow r_2$	$t_8 \rightarrow r_2$	$t_9 \rightarrow r_2$
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

图3 粒子位置编码

粒子移动范围由执行任务的可用资源量决定,因此,坐标值的变化范围在 0 至初始状态时 VMs 的数量之间.基于此,在粒子位置中每个坐标值的整数

部分对应于资源索引,即表示以特定坐标定义的分配至任务的计算资源.通过这种方式,粒子位置可编码为任务与资源间的映射.以图 3 为例,资源池有三个资源,则每个坐标值的取值范围为 $[0,3]$,坐标 1 对应任务 1,其值 1.2 表明任务分配至资源 1;坐标 2 对应任务 2,其值 1.0 表明任务 2 分配至资源 1;依次类推.

为了缩小搜索空间,笔者设计以下策略:令 P 表示给定工作流中可并行执行最大任务数量,则 PSO 用来寻找近似最优调度方案的初始资源池 R_{initial} 由 P 中每个任务的每类 VM 组成.然后,算法从资源池中选择合适数量 VMs 租用.通过这种方式降低搜索空间大小,同时保证算法可以以并行方式执行所有任务. R_{initial} 的大小设置为 $|P| \times n$, n 表示可用 VMs 类型的数量.

求解约束最优化问题时:当比较解的优劣时,若两种解均可行,则选择适应度更优解作为新解;若一种可行,一种不可行,则选择可行解;如果两种解均不可行,则拥有更小全局约束违例的解作为新的解.本文的约束最优化问题为单约束问题,笔者将解的全局约束违例值定义为解的执行跨度与截止时间之差.则执行跨度更接近于截止时间的解将视为更优粒子.

4.2 调度算法

算法 2 给出了粒子位置与调度方案的转换方法.初始状态下,资源集 R 与任务-资源映射 M 均为空,TEC 和 TET 设为 0,如步骤 3.然后,计算每个任务在每个资源 r_i 上的执行时间, $r_i \in R_{\text{initial}}$,并表示为一个矩阵,行为任务,列为资源, $\text{ExeTime}[i,j]$ 为 t_i 在 r_j 上的运行时间,通过式(1)计算,如步骤 4.接下来即是计算数据传输时间矩阵,该矩阵可表示为 DAG 的加权邻接矩阵, $\text{TransferTime}[i,j]$ 包含任务 t_i 与 t_j 间传输输出数据的时间,通过式(2)计算,如步骤 5.若 $i=j$ 或两任务间没有直接边时,其值为 0.如图 4 为以上两种矩阵示例.

	r_1	r_2	r_3		t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
$\text{exeTime} =$	t_1	2	1	4	$\text{transfer Time} =$	t_1	0	9	9	9	0	0	0
	t_2	4	3	6		t_2	0	0	0	5	0	0	0
	t_3	10	6	15		t_3	0	0	0	0	0	1	0
	t_4	7	4	12		t_4	0	0	0	0	0	7	0
	t_5	8	4	10		t_5	0	0	0	0	0	0	2
	t_6	3	2	7		t_6	0	0	0	0	0	0	3
	t_7	12	7	18		t_7	0	0	0	0	0	0	2
	t_8	9	5	20		t_8	0	0	0	0	0	0	10
	t_9	13	8	19		t_9	0	0	0	0	0	0	0

(a) 执行时间矩阵

(b) 数据传输时间矩阵

图 4 执行时间与数据传输时间矩阵

算法目标是对粒子位置进行解码,并构造调度方案.策略为:算法在位置数组 pos 中的每个坐标 i 上进行不断迭代,并不断更新 R 和 M .首先,算法决定哪些任务与哪些资源需要关联至当前坐标中,并计算其值,该过程通过前文的编码策略完成, achi 为对于任务 t_i 的坐标 i ,其值 $\text{pos}[i]$ 对应资源 $r_{\text{pos}[i]}$, $r_{\text{pos}[i]} \in R_{\text{initial}}$,如步骤 7.如此,可以确定调度方案四元组的前两个成份.下面计算任务的开始时间 ST_{t_i} 和结束时间 ET_{t_i} .

1) ST_{t_i} 计算方法.开始时间 ST_{t_i} 有两种场景.1、任务没有父任务时,只要分配至该任务的 VM 为空闲状态即可开始执行,其值对应于资源 $r_{\text{pos}[i]}$ 的当前租用结束时间,为 $\text{LET}_{r_{\text{pos}[i]}}$,如步骤 8 ~ 9;2、如果任务拥有一个以上父任务,只有等待其父任务完成并发送数据后才可以开始执行.如果该 VM 处于执行状态,则需等待该 VM 空闲才可开始执行,如步骤 10 ~ 12.

2) ET_{t_i} 计算方法. ET_{t_i} 通过任务的总处理时间和开始时间计算,对于计算处理时间 $PT_{t_i}^{\text{rpos}[j]}$,首先需要计算执行与数据传输时间,前者可由 $\text{exeTime}[i, \text{pos}[i]]$ 得到,如步骤 13,后者通过将 t_i 的每个子任务 $t_{\text{child}(i)}$ 的 $\text{TransferTime}[i, \text{child}(i)]$ 的值加至 $r_{\text{pos}[i]}$ 上得到,如步骤 14 ~ 18.这两个值通过公式(3)同步加至 $PT_{t_i}^{\text{rpos}[j]}$,如步骤 19.最终通过从 $PT_{t_i}^{\text{rpos}[j]}$ 中替换 ST_{t_i} 得到 ET_{t_i} ,如步骤 20.

至此,已经得到 $m^{\text{rpos}[j]}_{t_i} = (t_i, r_{\text{pos}[i]}, ST_{t_i}, ET_{t_i})$ 中的所有要素,如步骤 21.但需不断更新与 $r_{\text{pos}[i]}$ 相关的两个参数:一是 VM 的启动时间 $\text{LST}_{r_{\text{pos}[i]}}$,如果 R 中存在资源,则表明开始时间已知,且 $\text{LST}_{r_{\text{pos}[i]}}$ 无需更新;如果此时资源是新资源, t_i 是第一个分配至该资源的任务,则更新 R ,使其包含新资源 $r_{\text{pos}[i]}$,并将 $\text{LST}_{r_{\text{pos}[i]}}$ 设置为任务的开始时间和 VM 的启动时间的较大值,如步骤 23 ~ 24.二是需要更新 $\text{LET}_{r_{\text{pos}[i]}}$,代表资源被调度完成其上最后一个任务的时间.资源 $r_{\text{pos}[i]}$ 的新租用结束时间即为处理 t_i 的时间($PT_{t_i}^{\text{rpos}[j]}$)加上资源调度开始时间 $\text{LST}_{r_{\text{pos}[i]}}$,如步骤 25.

算法完成位置向量中每个坐标的任务处理之后, R 将包含所有租用资源及其开始和关闭时间,同时, M 将包含每个任务与资源的映射方案,以及任务开始和结束时间.基于此,利用公式(4)和(5)计算在当前解中的总执行时间 TET 和总执行代价 TEC,并构造给定粒子位置时的调度方案,如步骤 26 ~ 27.

算法 2 基于 PSO 的工作流调度算法

1. input: tasks set T , initial resource pool $R_{initial}$, an array $pos[|T|]$ representing a particle's position
2. output: schedule solution S
3. initialize schedule components, $R = \emptyset, M = \emptyset, TEC = 0, TET = 0$
4. calculate $ExeTime[|T| \times |R_{initial}|]$
5. calculate $TransferTime[|T| \times |T|]$
6. for $i=0$ to $|T|-1$
7. $t_i = T[i], r_{pos[i]} = R_{initial}[pos[i]]$
8. if t_i has no parents
9. $ST_{t_i} = LET_{r_{pos[i]}}$
10. else
11. $ST_{t_i} = \max(\max\{ET_{t_p} : t_p \in parents(t_i)\}, LET_{r_{pos[i]}})$
12. end if
13. $exe = exeTime[i][pos[i]]$
14. for each child t_c of t_i
15. if t_c is mapped to a resource different to $r_{pos[i]}$
16. $transfer += TransferTime[i][c]$
17. end if
18. end for
19. $PT_{t_i}^{pos[j]} = exe + transfer$
20. $ET_{t_i} = PT_{t_i}^{pos[j]} - ST_{t_i}$
21. $m_{t_i}^{pos[j]} = (t_i, r_{pos[i]}, ST_{t_i}, ET_{t_i})$
22. $M = M \cup \{m_{t_i}^{pos[j]}\}$
23. if $r_{pos[i]} \notin R$
24. $LST_{r_{pos[i]}} = \max(ST_{t_i}, bootTime), R = R \cup \{r_{pos[i]}\}$
25. $LET_{r_{pos[i]}} = PT_{t_i}^{pos[j]} + LST_{r_{pos[i]}}$
26. calculate TEC and TET according to equation (4) and (5)
27. return $S = (R, M, TEC, TET)$

5 实验仿真

5.1 环境配置

利用 WorkflowSim^[10]构建仿真实验评估算法性能,选取四种科学工作流作为测试数据源: Montage, LIGO, SIPHT 及 CyberShake, 四种工作流结构各异^[11],如图 5 所示.同时,为了评估资源异构性对执行时间和执行代价的影响,笔者还基于 CWS_PSO 设计了变种算法 PSO_HOM,该算法仅考虑同质化环境.除此之外,选择 IC-PCP^[7]和 SCS^[5]作为基准算法.

模拟拥有单个数据中心和六种类型 VMs 的云环境,VM 配置参考 Amazon EC2,如表 2 所示.每种应用以三种不同规模工作流配置: small, large 及 extra large,其中,small 平均配置 50 个任务,large 平均配置

100 个任务,extra 平均配置 1 000 个任务. VM 支付单位时间为 1 hour,VM 启动时间设置为 97 sec.

表 2 VMs 类型

类型名称	EC2 单元	处理能力	代价/小时 \$
m1. small	1	4400	0.06
m1. medium	2	8800	0.12
m1. large	4	17600	0.24
m1. xlarge	8	35200	0.48
m3. xlarge	13	57200	0.50
m3. doublexlarge	26	114400	1.00

将 Deadline 分布于最慢执行时间与最快执行时间之间以建立约束松紧变化.最慢执行时间定义为将所有任务均调度至最低价 VM 上执行得到的时间,而最快执行时间则定义为将所有任务调度至最快 VM 上执行得到的时间.然后,将两种时间之差平均划分为五个步长,Deadline1 设为最快(最短)执行时间加一个步长,Deadline2 设为最快执行时间加两个步长,依此类推.其它参数设置如下: $c_1 = c_2 = 2.0, w = 0.5$,最大粒子数为 100.

5.2 结果分析

实验 1 分析工作流截止时间满意度.对于 Montage, IC-PCP 无法满足所有约束, PSO_HOM 对于 Deadline1 的满意度少于 50%,在 Deadline3 和 Deadline4 上达到 100%. CWS_PSO 和 SCS 在约束满意度上是更优的, CWS_PSO 在 Deadline1 和 Deadline2 上均弱小于 100%,而在 Deadline3 和 Deadline4 上达到 100%.对于 SIPHT, IC-PCP 无法满足大部分约束,其性能差于其他算法, SCS, CWS_PSO 和 PSO_HOM 对于 Deadline2、Deadline3 和 Deadline4 的满意度均在 95%以上.对于 LIGO, SCS, CWS_PSO 和 PSO_HOM 的结果是类似的, IC-PCP 对于前三个约束的满意度均为 0,在 Deadline4 上的满意度达到 100%.对于 CyberShake, IC-PCP 在四种约束上均表达最差.对于 Deadline1, SCS 和 CWS_PSO 拥有最低百分例,而比较 IC-PCP,两者在约束宽松时表现更好.对于 Deadline2, CWS_PSO, PSO_HOM 和 SCS 表现相似.总体来说, IC-PCP 差于其他算法, SCS 在大部分情况下满意度达到 100%. CWS_PSO 和 PSO_HOM 与 SCS 类似,大部分情况下满意度均达到 100%,这源于算法考虑了云动态特性和 CPU 性能动态性.

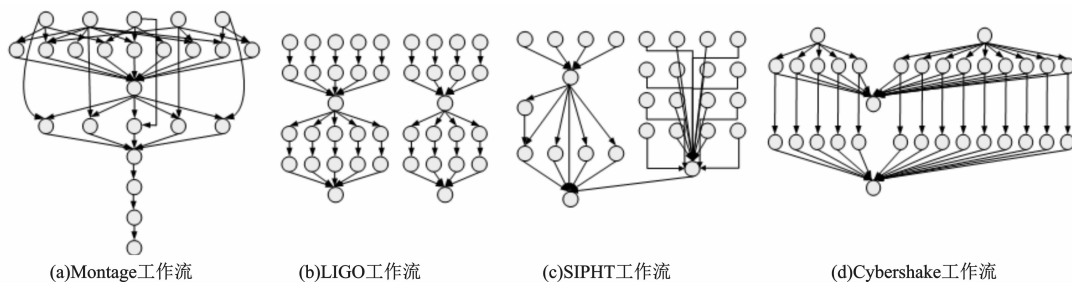


图 5 科学 workflow 模型

实验 2 分析总执行代价情况. 对于 Montage, IC-PCP 的代价在四种约束中均是最低的. 满足约束的算法中, CWS_PSO 在 Deadline1 得到了最低代价, PSO_HOM 在 Deadline2 代价最低, CWS_PSO 在 Deadline3 和 Deadline4 代价最低. 该结果表明, CWS_PSO 在代价上优于 SCS. IC-PCP 代价最低且执行时间最长源于在计算调度时间时没有考虑 VM 性能变化和启动时间. 对于 SIPHT, CWS_PSO 在前三种约束中性能最优. IC-PCP 在四种约束中得到了最低代价, 但是其执行时间在前三个约束中均超过约束, 因此, IC-PCP 在相对宽松约束中性能优于其他算法. 由于 SCS、CWS_PSO 和 PSO_HOM 满足所有约束, 则代价最低的性能最优. 总体来看, CWS_PSO 和 SCS 可以满足截止时间约束. 对于 CyberShake, 面对 Deadline1 时 IC-PCP 代价更高, 此时, IC-PCP 拥有最低且弱高于截止时间的执行跨度. 其他算法拥有更低代价, 但是以无法满足约束的更长执行跨度为代价的. CWS_PSO 和 SCS 的性能相似, 其执行跨度弱高于 IC-PCP, 但代价更低. 对于 Deadline2、Deadline3 和 Deadline4, IC-PCP 的调度更廉价但均无法满足约束, 此时的 CWS_PSO 则拥有最低的代价, 算法性能最好. 除了 Deadline1, 四种算法在该 workflow 中的代价相似, CWS_PSO 在 Deadline2、Deadline3 和 Deadline4 上性能超过其他算法. 对于 LIGO, IC-PCP 的性能与 SIPHT 中类似, 其代价最低, 执行跨度最长. CWS_PSO 和 SCS 均可满足所有截止时间约束, 同时 CWS_PSO 性能更优, 由于它的执行跨度更小, 代价更低.

6 结束语

提出了基于 PSO 的调度算法. 将 workflow 调度最优化问题建立为截止时间约束的代价最小化模型, 模型综合考虑云资源使用的即付即用、异构、弹性及动态性、性能变化及 VM 启动时间, 设计了最优调度方案算法. 结果证明了算法的有效性和可行性.

参考文献:

- [1] G. Juve, A. Chervenak, E. Deelman, et. al, Characterizing and profiling scientific workflows[J], Future Generation Computer System, 2012, 29(3): 682-692.
- [2] Liu L, Zhang M, Lin Y, et. al. A survey on workflow management and scheduling in cloud computing[C]\In 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE, 2014; 837-846.
- [3] Wu F, Wu Q, Tan Y. Workflow scheduling in cloud: a survey[J]. The Journal of Supercomputing, 2015, 71(9): 3373-3418.
- [4] Wang X, Yeo CS, Buyya R, Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm[J]. Future Generation Computer System, 2012, 27(8): 1124-1134.
- [5] 郑敏, 曹健, 姚艳, 面向价格动态变化的云 workflow 调度算法[J], 计算机集成制造系统, 2013, 19(8): 1849-1858.
- [6] M. Mao and M. Humphrey, Auto-scaling to minimize cost and meet application deadlines in cloud workflows [C]\in Proceeding International Conference on High Performance Computing, Network, Storage Anal., IEEE, 2012; 1-12.
- [7] M. Malawski, G. Juve, Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds[C]\in Proceeding on High Performance Computing, Network, Storage Anal., IEEE, 2012; 1-11.
- [8] S. Abrishami, D. Epema, Deadline-constrained workflow scheduling algorithms for IaaS clouds[J], Future Generation Computer System, 2012, 23(8): 1400-1414.
- [9] Clerc M. Particle Swarm Optimization[M]// Fractional Order Darwinian Particle Swarm Optimization. Springer International Publishing, 2016; 149-150.
- [10] Chen W, WorkflowSim: a toolkit for simulating scientific workflows in distributed environments[C]\In IEEE 8th International Conference on E-Science, IEEE, 2012; 1-8.

图,结果如图 5 所示:

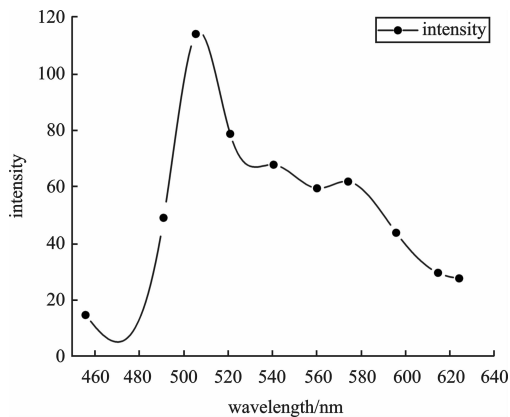


图 5 金纳米球粒子散射光谱图

结合表格数据及粒子散射光谱图分析可知,标准化后的粒子统计光谱在 510 nm 左右产生共振峰,与金纳米球(直径 80 nm)实际散射光谱基本吻合^[8],表明待测样品为金纳米球粒子。

6 结束语

图像处理系统的关键在于系统的一体化集成和功能的实现. 本文为暗场成像系统设计了一种实时图像处理子系统. 基于原有暗场系统,本系统较好地实现了暗场系统的实时图像处理需求. 结合 TMS320C6748 提供的硬件资源及接口,设计了以上各个功能模块. 并在各个功能模块的基础上设计了优越的驱动程序以及图像处理算法. 通过金纳米球粒子暗场图像的处理与分析结果可知,系统功能与性能满足了暗场图像实时处理的要求. 本系统尚有不足,有待改进. 彩色 CMOS 摄像头对暗场图像的采集使得粒子成像的清晰度降低,且敏捷度不足. 可改用黑白 CCD 对图像进行采集,其清晰度及敏捷度相较于彩色 CMOS 都有很大程度地提升^[9]. 总而言之,本系统是一个稳定工作的实时暗场图像采集与处理系统.

参考文献:

- [1] 雷玉堂. 现代摄像机 DSP 图像处理系统的设计[J]. 中国公共安全, 2009 (19): 146-151.
- [2] Van Dijk M A, Lippitz M, Orrit M. Far-field optical microscopy of singlemetal nanoparticles[J]. Accounts of Chemical Research, 2005, 38(7): 594-601.
- [3] Zhang B, Xue L, Qin S, et al. The Experiment and Analysis of Image Acquisition System Based on the Hardware Platform[C]// Proceedings of the 2015 International Conference on Communications, Signal Processing, and Systems. Springer Berlin Heidelberg, 2016: 869-879.
- [4] 朱明, 鲁剑锋, 赵建, 等. 基于 TMS320C6202 DSP 的实时数字图像处理系统的设计[J]. 光学精密工程, 2003, 11(5): 497-501.
- [5] 汪安民. TMS320C674x DSP 应用开发[M]. 北京航空航天大学出版社, 2012.
- [6] 黄德天, 陈建华. DSP 图像处理的程序优化[J]. 中国光学与应用光学, 2009, 2(5): 452-459.
- [7] Russ J C, Matey J R, Mallinckrodt A J, et al. The image processing handbook[J]. Computers in Physics, 1994, 8(2): 177-178.
- [8] 王健, 朱涛, 张续, 等. 表面增强拉曼散射强度与金纳米粒子粒径关系[J]. 物理化学学报, 1999, 15(15): 476-480.
- [9] Hain R, Kähler C J, Tropea C. Comparison of CCD, CMOS and intensified cameras[J]. Experiments in fluids, 2007, 42(3): 403-411.

作者简介:

- 刘国华 男, (1964-), 博士, 教授. 研究方向为生化传感器的设计. E-mail: liugh@nankai.edu.cn.
- 鲁楷滨 男, (1993-), 硕士研究生. 研究方向为生化传感器及其控制电路设计.
- 杜谦 男, (1990-), 博士研究生. 研究方向为传感技术与智能系统设计.
- 肖靖楠 男, (1992-), 硕士研究生. 研究方向为生化传感器的软件设计.

(上接第 127 页)

- [11] M. Rodriguez, R. Buyya, Deadline based resource provisioning and scheduling algorithm for scientific workflows on Clouds[J], IEEE Transactions on Cloud Computing, 2014, 2(2): 222-235.

作者简介:

- 薛凡 男, (1978-), 博士生, 讲师. 研究方向为智能计算、云计算. E-mail: xufam78@163.com.
- 吴志健 男, (1963-), 教授、博导. 研究方向为智能计算、智能信息处理.