计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(7):127−132 [doi: 10.15888/j.cnki.csa.007003]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

云环境下科学工作流的调度算法①

冯复剑

(江苏第二师范学院, 南京 210013)

摘　要: 提出一种云环境下科学工作流的调度算法. 针对已有的调度算法和松弛时间资源分配策略均未考虑“或”控制结构的不足, 给出了关键活动优先级 (Critical Activity Priority, CAP) 的概念; 定义了活动的服务效益比 (Service Benefit Ratio, SBR); 提出了活动松弛时间分配策略; 并从流程定义和实例运行两个层次, 给出了活动截止期限的分配算法. 该项研究成果为解决科学工作流调度过程中的时间-成本优化问题提供了更合适的解决方案.

关键词: 科学工作流; 调度; 云; 服务效益比

引用格式: 冯复剑.云环境下科学工作流的调度算法.计算机系统应用,2019,28(7):127–132. http://www.c-s-a.org.cn/1003-3254/7003.html

Algorithm of Scientific Workflow Scheduling in Commercial Clouds

FENG Fu-Jian

(Jiangsu Second Normal University, Nanjing 210013, China)

Abstract: A scheduling algorithm of scientific workflow in commercial clouds is proposed. To solve the problem of the existing scheduling algorithms and the slack time allocation strategies that do not consider the OR control structure, the critical activity priority (CAP) is defined. Also the service benefit ratio (SBR) and slack time allocation strategy of activity are presented. Then, from the two levels of definition time and running time, the deadline distribution algorithm of activity is proposed. The results of this study provide a more suitable solution for solving the time-cost optimization problem in the scientific workflow scheduling.

Key words: scientific workflow; scheduling; cloud; Service Benefit Ratio (SBR)

近年来随着科学研究越来越依赖于海量数据集的分析和分布式资源的使用, 科学工作流[1–3]得到了巨大的发展. 通过管理分析的复杂性、在分布式资源上执行必要的计算、收集分析结果的信息以及记录和再现科学分析等手段, 科学工作流管理系统提供了一个辅助科学发现过程的环境, 加速了科学进步的步伐.

商业云因其高弹性、专用软硬件基础设施和按需付费的成本模型等特性[4–6], 已经成为进行大规模科学分析的支撑平台. 通过各种云服务, 可以促进工作流的执行. 其执行过程中, 需要考虑一个或多个 QoS 约束, 常见的为时间-成本优化. 工作流调度的时间-成本优化是一个难题, 国内外学者已从不同方面展开了研究并取得了丰富的成果[7–15]. 一些综述文章[16–18]对其进行了较完整的阐述, 这里就不再一一回顾. 为了简洁起见, 仅描述本文解决的问题.

现有的科学工作流调度算法多基于有向无环图 (Directed Acyclic Graph, DAG) 建模, 仅对工作流结构中的顺序和“与”结构适用, 并不支持“或”结构. 这限制了算法的使用范围和性能. 文献[12]提出, 可以将“或”结构转换成“与”结构进行处理. 这是不合理的, 违背了工作流语义. 对于“或”结构中的支路每个流程实例只会选择其中一条, 而“与”结构的所有支路都是同时执行的, 两者是不可以直接转换的. 正因为未考虑“或”结构, 现有的算法存在以下不足:

① 收稿时间: 2018-12-27; 修改时间: 2019-01-18, 2019-02-26; 采用时间: 2019-03-01; csa 在线出版时间: 2019-07-01

(1) 同一层活动分配相同的截止期限不合理

逆向分层调度算法[12](Deadline Bottom Level, DBL) 以及基于此而改进的算法中, 认为同一层的所有活动具有相同的截止期限. 图 1 为一个简单的基于逆向分层的工作流实例, 图中活动节点的上方数字表示选择最快执行服务所需要消耗的时间. 活动 1 完成后, 根据输出条件选择执行支路{2, 3, 4}或者支路{5}. 依据 DBL 算法求得每个活动的时间区间如表 1. 活动 5 的时间区间为[4, 27], 所以可选服务的耗时范围为[7, 23], 假定其执行最慢的服务耗时 23 个单位时间. 基于满足截止期限的前提下选择最低成本的原则, 工作流在实例执行过程中若选择支路{5}将会调用执行时间最长的服务, 因此将消耗掉额外 16 个单位的时间. 由此带来两个问题:

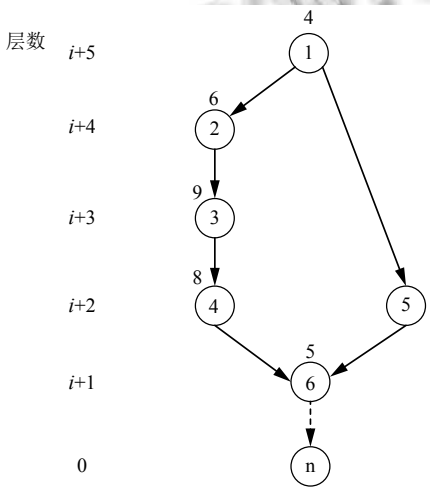

图 1 一个简单的工作流实例

表 1 DBL 活动时间区间

活动编号	开始时间	结束时间 (截止期限)
1	0	4
2	4	10
3	10	19
4	19	27
5	4	27
6	27	32

① 活动 5 本可以在时间点 11 时刻即可结束, 则活动 6 将提前 16 个单位时间开始, 在不改变后续活动既定调度方案的情况下, 将使得整个工作流提前 16 个单位时间结束;

② 将额外消耗的 16 个单位时间分配给后续未开始的活动, 从而获得更优的执行方案.

当这两种情况下带来的效益远远大于单个活动 5 优化带来的效益时, DBL 对截止期限的分配存在弊端.

(2) 松弛时间优化未同时考虑活动控制结构和服务效益

当给定的工作流截止期限大于算法计算得出的截止期限时, 工作流存在松弛时间. 为了提高效益, 需要将这部分时间资源合理地分配给能够带来费用优化的活动. 表 2 为图 1 工作流实例中部分活动对应的候选服务列表.

表 2 候选服务列表

活动编号	服务时间	服务成本
1	4	6
2	6	10
3	9	7
4	8	8
5	7	10
5	9	8
6	5	12
6	7	11

假设现有 2 个单位的工作流松弛时间, 由于只有活动{5,6}有多个候选的服务, 因此只需要对这两个活动进行优化. 根据文献[13]提出的宽裕时间有效分配算法 (Slack-time Effective Allocation, SEA), 活动 5,6 的邻服务级差性价比分别为 1 和 0.5, 因此将松弛时间全部分配给活动 5. 基于“或”结构的特性, 若工作流实际执行过程中选择活动 5 所在的支路是小概率事件, 此时分配的松弛时间将不会带来费用优化, 从而导致时间资源的浪费.

(3) 其它问题

文献 12 中定理 2 认为$BL_{\min} \geqslant L_{CT}$, 其中: $BL_{\min}$是按照逆向分组后求得的工作流最小完工时间, L_{CT}为最小关键路径法计算的完工时间. 该定理不成立, 因为$BL_{\min}$和L_{CT}均根据每个活动的最快服务求得, 若$BL_{\min} > L_{CT}$表示同一工作流在活动执行者相同的条件下通过两种不同方法得到不同的完工时间, 这是不合理的.

为了解决以上的不足, 本文提出一种新的调度算法, 充分考虑工作流控制结构中的顺序、“与”和“或”结构, 并分别从静态调度和动态监控两个阶段对截止期限的分配进行讨论.

1 模型和相关概念

1.1 工作流模型

定义 1. 工作流的 DAG 模型是一个二元组$\langle N,E\rangle$, 其中: $N=\{1,2,3,4,\cdots,n|n\in N^*\}$表示所有的活动集; $E=\{e_{ij}|i,j\in N\}$是活动之间的逻辑关系.

为了讨论方便, 下面给出一些变量的定义及其数学表达式:

(1) $S(i)$: 活动i的候选服务集合, 其大小为$L(i)$.

(2) $S(i)_j$: 活动i的第j个候选服务.

(3) $T(i)_j$: $S(i)_j$所需的处理时间和数据传输时间之和.

(4) $C(i)_j$: $S(i)_j$对应的服务费用和数据传输成本之和.

(5) $D(i)$: 活动i的截止期限.

(6) $TS(i)$: 活动i的松弛时间, 当选定$S(i)_j$后有: $D(i)=T(i)_j+TS(i)$.

(7) $pred(i)/succ(i)$: 活动i的前驱/后继.

为了排除网络等其它客观环境的影响, 假设云服务供应商能够提供无限数量的实例访问, 且不同存储服务之间的带宽基本相同.

1.2 关键活动优先级

为了叙述方便, 下面给出一些符号的语义:

(1) CP: 工作流的最长执行路径, 即关键路径.

(2) $CP(i)$: 活动i是关键活动.

(3) $CP_s(i)/CP_{and}(i)/CP_{or}(i)$: 关键活动$i$.处于顺序支路/“与”支路/“或”支路, 同步活动为关键活动, 且在顺序支路上.

定义 2. 关键活动优先级 (Critical Activity Priority, CAP), 反应活动时间-成本优化对整个流程性能提升的影响程度. 优先级越高, 表示活动的优化带来的影响越大, 反之越小.

关键活动按照其所处控制结构的不同, 相应的优先级不同. 下面分别对 3 种拓扑结构进行阐述.

(1) “与”支路

“与”控制结构中的每条支路需要在同步活动处等待, 当关键活动获得额外的时间资源后, 将延长所在支路的执行时间, 进而影响整个控制块中每条支路的运行. 因此, “与”支路上单个关键活动的优化将会提升整个“与”控制结构的性能.

(2) 顺序支路

顺序支路上的关键活动, 对其分配额外的时间资源只会优化该活动本身, 并不会波及其它活动.

(3) “或”支路

由于“或”结构在实例运行之前无法确定选择哪条支路, 因此, 对“或”支路上的关键活动分配额外的时间资源有可能无法带来效益.

综上所述, 可以得到关键活动的优先级为: $CAP(CP_{and}(i))>CAP(CP_s(i))>CAP(CP_{or}(i))$.

2 调度算法

本文按照如下步骤解决调度问题:

步骤 1 寻找工作流的最小关键路径.

步骤 2 查询可用时隙, 根据每个任务的局部最优解生成最优调度计划.

步骤 3 实时监控流程实例的运行状态.

接下来详细陈述步骤 2–3, 关键路径的寻找通过文献[19]提出的方法实现.

2.1 松弛时间分配策略

定义 3. 服务效益比 (Service Benefit Ratio, SBR), 衡量服务替换所带来效益的大小, 其计算公式为:

$$SBR(i)_{m,n}=\frac{|C(i)_m-C(i)_n|}{|T(i)_m-T(i)_n|}=\frac{\Delta C(i)_{m,n}}{\Delta T(i)_{m,n}},\ L(i)>1 \quad (1)$$

式 (1) 中的m、n分别表示第m、n个候选服务.

对优先级相同的活动集合A, 基于服务效益比的松弛时间分配算法 (Slack Time Allocation Based on SBR, STABSBR) 如下:

算法 1. STABSBR 算法

输入: A,$\forall i\in A$具有相同的活动优先级; 松弛时间TS;
输出: $\{TS(i)\}$;
Begin
1 计算A中每个活动i的下一个候选服务 ($S(i)$按照成本降序排列) 与当前选择服务的$\Delta C(i)_{m,n}$, $\Delta T(i)_{m,n}$, $SBR(i)_{m,n}$;
2 while($A\neq\emptyset$)
3 {
4 从A中去除所有$\Delta T(i)_{m,n}>TS$的活动, 并按照$SBR(i)_{m,n}$降序、$\Delta T(i)_{m,n}$升序排列得到B;
5 if ($B\neq\emptyset$)
6 {
7 $TS(B(0))=\Delta T(B(0))$, $TS=TS-TS(B(0))$;
8 将活动$B(0)$移出A;
9 }
10 }
End

该算法能够将全局额外的时间资源合理的在活动之间进行分配, 从而产生最优的调度方案. 假设现有顺

序串联的两个活动 1 和 2, 其候选服务集分别为 $S(1)=\{(T,C)\}=\{(5,9),(8,8)\}$, $S(2)=\{(T,C)\}=\{(6,10),(8,8)\}$, 且根据工作流执行时间下界求得其松弛时间为$TS(WF)=3$. 从前面的分析可以知道, $SBR(2)=\dfrac{|10-8|}{|6-8|}=1>SBR(1)=\dfrac{|9-8|}{|5-8|}=\dfrac{1}{3}$. 因此, 给活动 2 分配两个单位的松弛时间, 即选择第 2 个候选服务作为执行服务. 剩余的 1 个单位时间资源由于无法提供给活动 1 以满足其最低的时间增量, 将不再继续优化.

2.2 静态调度

静态调度需要考虑活动的拓扑结构, 由关键活动优先级可知对于顺序和“或”结构松弛时间的再分配只影响单个活动, 而“与”结构会影响到整个控制块. 因此, 需要对“与”结构进行单独说明.

图 2 “与”控制结构

以图 2 所示的“与”控制结构为例, $branch(1)$, $branch(2)$...$branch(i)$分别表示每条支路, 假设$branch(i)$是子关键路径. 现根据松弛时间分配策略为某个关键活动m分配了ΔT的松弛时间, 则可求得“与”控制块新的截止期限为$(D(branch(i))+\Delta T)$, 因此其他非关键路径支路获得的松弛时间为$(D(branch(i))+\Delta T-D(branch(n)))$, $1\leqslant n\leqslant i-1$. 此时, 可在每条支路内部按照松弛时间分配策略再对调度进行调整.

考虑工作流控制结构的静态调度算法 (Static Scheduling Considering Workflow Control Structure, SSCWCS) 如下:

算法 2. SSCWCS 算法

输入: $\langle N,E\rangle$, $D(WF)$;
输出: $D(i)$;

```
Begin
1   为每个活动选择执行最快的服务, 以此求得工作流的关键路径CP, 并计算其执行时间下界D(CP)min;
2   if (D(WF)<D(CP)min)
3   {
4       提醒管理者需要修改D(WF);
5   }
6   else if(D(WF)>D(CP)min)
7   {
8       计算工作流的松弛时间TS=D(WF)−D(CP)min;
9       将关键活动按照拓扑结构分类, 分别求得B=[{CPand(i)},{CPs(i)},{CPor(i)},{N−({CPand(i)}∪{CPs(i)}∪{CPor(i)})}];
10      for(i=0;i<4;i++)
11      {
12          将A=B(i),TS分别代入算法 1 进行优化;
13      }
14      if (B(0)≠∅)
15      {
16      计算“与”关键支路新的截止期限D(branch(CPand))′并将其作为整个控制块新的截止期限;
17  for(i=0;i<length(and);i++)
18  {
19          if (branch(i)≠branch(CPand))
20          {
21      TS(branch(i))=D(branch(CPand))′−D(branch(i)), 按照算法 1 对该支路进行调优;
22          }
23  }
24      }
25  }
End
```

2.3 动态监控

由于“或”结构在流程定义阶段无法确定实际运行过程中所选择的支路, 因此静态调度并未对非关键路径支路和关键路径支路设置相同的截止期限. 这就需要在执行过程中, 为“或”结构设置监测点, 根据选择的支路动态的为后续未执行的活动重新生成调度方案.

通过为每个“或”分支活动设置监测点, 进而监听该活动的结束事件, 并根据输出条件判断后续所选择的支路. 若选择的是关键路径支路, 则不更改现有的调度方案, 否则对其进行优化. “或”结构动态监控优化算法 (OR Dynamic Monitoring, ORDM) 如下.

算法 3. ORDM 算法

输入: $\langle N,E\rangle$, $D(WF)$;
输出: $D(i)$;

```
Begin
1   找出所有的“或”控制结构, 并设置“或”分支活动ior_split为监测点;
2   监听ior_split的结束事件, 记录其完成时间TF(ior_split), 并根据输出条件判定后续选择的支路branch;
3   if (branch=branch(CP))
4   {
5       按照静态调度方案对活动进行调度;
6   }
7   else
8   {
9       根据当前的调度方案求得未完成活动的关键路径CP′及其需要消耗的时间D(CP′);
```

10 $D(WF)'=D(WF)-TF(i_{or\_split})$;
11 分别将$CP=CP'$, $D(CP)_{\min}=D(CP')$, $D(WF)=D(WF)'$代入算法 2, 对未完成的活动进行调优;
12 }
End

算法 3 动态监控“或”结构的执行情况, 规避了为所有“或”分支分配相同截止期限而造成效益无法最大化的情况的出现. 本文提出的动态监控思路同样适合于流程执行过程中出现的简单异常, 如服务的延迟、异常导致服务不可用. 此时, 可以把异常节点至结束活动的拓扑当作是新的工作流, 并按照算法 3 中步骤 8–12 重新调度. 其它复杂的异常处理将是下一步研究的对象.

3 实例

本章对前文提出的调度方法进行实例应用. 图 3 是虚拟科学分析过程的 DAG 模型, 表 3 是模型中每个活动对应的候选服务集合, 时间单位为分钟 (min), 费用单位为人民币 (￥). 设定工作流的开始时间为 0, 截止期限为 110.

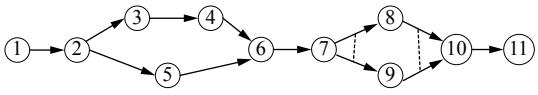

图 3 工作流 DAG 模型

表 3 活动候选服务集合

活动编号	候选服务 (T, C)/(min, ￥)	活动编号	候选服务 (T, C)/(min, ￥)
1	(4,9)	6	(20, 18)
1	(6, 8)	7	(20, 30)
2	(8, 8)	8	(5, 12)
2	(10, 7)	8	(7, 11)
2	(12, 6)	9	(15, 5)
3	(6, 6)	10	(4, 11)
4	(10, 15)	10	(8, 9)
4	(15, 9)	11	(7, 18)
5	(12, 16)	11	(15, 10)

3.1 关键路径时间下界

每个活动选择耗时最短的服务的条件下, 按照如下步骤求得关键路径执行时间的下界.

(1) “与”结构

$T(3,4)_{\min}=T(3)_{\min}+T(4)_{\min}=6+10=16, T(5)_{\min}=12$, $T(3,4)_{\min}>T(5)_{\min}$, 因此关键路径为活动 3,4.

(2) “或”结构

$T(8)_{\min}=5, T(9)_{\min}=15, T(9)_{\min}>T(8)_{\min}$, 所以关键支路为活动 9.

顺序活动处于关键路径上, 因此整个工作流的关键路径由活动 1,2,3,4,6,7,9,10,11 组成, 其执行时间下界为$D(CP)_{\min}=T(1)_{\min}+T(2)_{\min}+T(3,4)_{\min}+T(6)_{\min}+T(7)_{\min}+T(9)_{\min}+T(10)_{\min}+T(11)_{\min}=94$.

3.2 松弛时间分配

工作流的松弛时间$TS(WF)=D(WF)-D(CP)_{\min}=110-94=16$. 将关键活动按照所处的控制结构分类得到 $\{CP_{and}(i)\}=\{3,4\}$, $\{CP_s(i)\}=\{1,2,6,7,10,11\}$, $\{CP_{or}(i)\}=\{9\}$. 根据算法 2 分配松弛时间.

(1) “与”关键活动

活动 3 只有一个候选服务, 已经是最优方案; 活动 4 的$\Delta T=5<16$, 所以其可以获得 5 个单位的额外时间, 此时工作流的剩余松弛时间$TS(WF)=16-5=11$.

(2) 顺序关键活动

同步活动 6,7 只有单个候选服务, 所以$\{\Delta T\}=\{\Delta T(1),\Delta T(2),\Delta T(10),\Delta T(11)\}=\{2,2,4,8\}$, $\{SBR\}=\{0.5,0.5,0.5,1\}$. 因此活动 11 将优先获得 8 个单位的时间资源, 此时工作流剩余的松弛时间为 3 个单位. 活动 1,2 的服务替换产生相同的效益, 因此给活动 1 分配 2 个单位时间.

经过上述的步骤, 工作流剩余 1 个单位的松弛时间, 无法满足其它活动更换服务所需的最小时间变量, 且又因为调整过后的“与”控制结构中活动 5 没有可替换的服务, 调优结束. 工作流优化过后的调度方案如表 4 所示.

表 4 活动执行服务

活动编号	服务 (T, C)/(min, ￥)	活动编号	服务 (T, C)/(min, ￥)
1	(6, 8)	7	(20, 30)
2	(8, 8)	8	(5, 12)
3	(6, 6)	9	(15, 5)
4	(15, 9)	10	(4, 11)
5	(12, 16)	11	(15, 10)
6	(20, 18)		

3.3 动态监控

活动 7 处设置监测点, $TF(7)=T(1,2,3,4,6,7)=75$, 根据其输出条件判断活动 8 所在的支路被选中, 此时活动 8,10,11 组成了新的工作流WF'. 其松弛时间$TS=D(WF)-TF(7)-T(8,10,11)=110-75-24=11$,

$\Delta T(8)=2, \Delta T(10)=4$, 易知活动 8,10 均可分得对应的额外时间资源, 即选择成本最低的服务.

4 结束语

为了更好地解决云环境下科学工作流调度过程中的时间-成本优化问题, 在 DAG 模型中增加了对工作流“或”控制结构的支持, 并提出了关键活动优先级的概念, 此举弥补了现有基于 DAG 的调度算法无法支持“或”结构的不足; 同时, 给出了活动的松弛时间分配策略, 并基于此提出了云环境下科学工作流的调度算法, 该方法由静态调度和动态监控组成. 最后通过虚拟的科学分析过程对上述的算法进行了说明和应用.

参考文献

1 张卫民, 刘灿灿, 骆志刚. 科学工作流技术研究综述. 国防科技大学学报, 2011, 33(3): 56–65. [doi: 10.3969/j.issn.1001-2486.2011.03.013]

2 Barker A, Van Hemert J. Scientific workflow: A survey and research directions. Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics. Gdansk, Poland. 2007. 746-753.

3 Gil Y, Deelman E, Ellisman M, *et al*. Examining the challenges of scientific workflows. Computer, 2007, 40(12): 24–32. [doi: 10.1109/MC.2007.421]

4 Arabnejad V, Bubendorfer K, Ng B, *et al*. A deadline constrained critical path heuristic for cost-effectively scheduling workflows. Proceedings of the IEEE/ACM 8th International Conference on Utility and Cloud Computing. Limassol, Cyprus. 2015. 242–250.

5 Arabnejad V, Bubendorfer K. Cost effective and deadline constrained scientific workflow scheduling for commercial clouds. Proceedings of the IEEE 14th International Symposium on Network Computing and Applications. Cambridge, MA, USA. 2016. 106-113.

6 Arabnejad V, Bubendorfer K, Ng B. Deadline distribution strategies for scientific workflow scheduling in commercial clouds. Proceedings of the IEEE/ACM 9th International Conference on Utility and Cloud Computing. Shanghai, China. 2017. 70-78.

7 Yu J, Buyya R, Tham CK. Cost-based scheduling of scientific workflow applications on utility grids. Proceedings of 1st International Conference on E-Science and Grid Computing. Melbourne, Australia. 2005. 140-147.

8 Yuan YC, Li XP, Wang Q, *et al*. Deadline division-based heuristic for cost optimization in workflow scheduling. Information Sciences, 2009, 179(15): 2562–2575. [doi: 10.1016/j.ins.2009.01.035]

9 Abrishami S, Naghibzadeh M, Epema DHJ. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Generation Computer Systems, 2013, 29(1): 158–169. [doi: 10.1016/j.future.2012.05.004]

10 Topcuoglu H, Hariri S, Wu MY. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3): 260–274. [doi: 10.1109/71.993206]

11 Yu J, Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Scientific Programming, 2006, 14(3-4): 217–230. [doi: 10.1155/2006/271608]

12 苑迎春, 李小平, 王茜, 等. 基于逆向分层的网格工作流调度算法. 计算机学报, 2008, 31(2): 282–290. [doi: 10.3321/j.issn:0254-4164.2008.02.012]

13 郑美光, 胡志刚, 杨柳, 等. 使用宽裕时间有效分配优化工作流逆向分层算法. 小型微型计算机系统, 2016, 37(8): 1639–1644. [doi: 10.3969/j.issn.1000-1220.2016.08.002]

14 龙浩, 梁毅, 邸瑞华. 基于相对效费比的网格工作流调度算法. 计算机集成制造系统, 2010, 16(3): 589–597.

15 彭佳, 谭文安, 孙勇, 等. QoS 约束下的分层工作流调度算法. 小型微型计算机系统, 2015, 36(7): 1444–1448. [doi: 10.3969/j.issn.1000-1220.2015.07.007]

16 Xavier S, Lovesum SPJ. A survey of various workflow scheduling algorithms in cloud environment. International Journal of Scientific and Research Publication, 2013, 3(2): 1–3.

17 Wu FH, Wu QB, Tan YS. Workflow scheduling in cloud: A survey. The Journal of Supercomputing, 2015, 71(9): 3373–3418. [doi: 10.1007/s11227-015-1438-4]

18 Arya LK, Verma A. Workflow scheduling algorithms in cloud environment - A survey. Proceedings of 2014 Recent Advances in Engineering and Computational Sciences. Chandigarh, India. 2014. 1-4.

19 李慧芳, 冯复剑. 时间约束工作流的截止期限管理及其动态监控. 北京理工大学学报, 2011, 31(8): 937–943.