

一种云环境下的科学 workflows 均衡调度算法

吴文兵

(福州外语外贸学院 信息系, 福建 福州 350202)

摘 要: 为了优化云环境中预算约束下的科学 workflows 调度问题, 提出一种 workflows 均衡调度算法 BDWTS. 算法以满足 workflows 预算约束并同步优化执行代价和执行时间为目标, 将 workflows 调度划分为四个阶段: workflows 分级、预算分割、任务选择和实例选择. workflows 分级通过自顶向下的方式对所有 workflows 任务进行分级, 预算分割中设计了六种用户预算在不同 workflows 分级上的子划分方法, 任务选择通过最早开始时间原则选择优先的调度任务, 实例选择则综合考虑时间因子和代价因子选择最优执行资源. 通过仿真实验, 证明了算法可以在预算约束下得到降低 workflows 执行时间和代价的均衡调度方案.

关键词: 云环境; workflows 调度; 均衡调度; 预算分配

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2018)10-0121-06

A Scientific Workflow Trade-off Scheduling Algorithm in Cloud Environment

WU Wen-bing

(College of Information, Fuzhou University of International Studies and Trade, Fuzhou 350202, China)

Abstract: For optimizing scientific workflow scheduling with budget constraint in cloud, a Budget Division Workflow Trade-off Scheduling (BDWTS) algorithm is presented. With synchronous optimization of makespan and cost meeting budget constraint as the objective, the algorithm is divided into four stages during scheduling workflow: workflow leveling, budget division, task selection and instance selection. Workflow leveling partitions the workflow tasks into different levels by the top-down method, budget division allocates the user-defined budget to each defined levels by six strategies, task selection selects the scheduled-priority task by the earliest start time principle, instance selection selects the optimal execution resource by overall considering the time factor and cost factor. It is proved that our algorithm can obtain the trade-off scheduling scheme of the execution time and execution cost under meeting budget constraint by simulation experimental results.

Key words: cloud environment; workflow scheduling; trade-off scheduling; budget distribution

1 引言

科学 workflows 广泛应用于大型复杂科学应用建模, 云计算环境提供的即付即用计算能力及定制式硬件设施, 是调度科学 workflows 应用的一种有效方法^[1]. 与传统 workflows 不同, 科学 workflows 规模更大, 任务的数据计算与通信代价更高, 其本质是实现相互依赖的任务至资源间的映射, 同时要求满足用户定

义的 QoS 约束(截止时间或预算). 传统 workflows 调度算法仅侧重于考虑优化执行效率(执行时间), 未考虑资源代价, 而云是商业化环境, 其资源使用是有偿付费的, 资源能力越高, 价格越高, 此时, 使用不同资源及不同调度方案得到的执行代价和时间是不同的. 因此, 云环境中的 workflows 调度需要同步考虑用时间和代价问题.

相关研究中, 文献[2]提出一种预算约束异构最

早完成时间算法 BHEFT,算法引入当前任务预算因子到未调度任务的剩余预算分配中,与本文不同的是,其任务预算与剩余预算是在逐任务分配的.文献[3,4]则仅侧重于截止时间的约束及截止时间在不同任务或不同分级间的分割,不适用于云环境.文献[5]提出预算感知回溯调度算法实现了多任务工作流的优化调度,文献[6,7]则提出一种预算约束的最小端到端延迟工作流调度算法,通过在关键路径上应用贪婪策略寻找调度方案,然而,以上三个文献中的资源代价模型并不适用于商业云特征.文献[8]提出基于自动扩展方法求解预算约束下的工作流调度,而本文将根据任务优先级采用正比例方式进行预算分割.

不同于以上工作,本文将重点关注于通过预算分割的方式优化工作流的执行代价与执行时间,在工作流分级及在不同分级上赋予待调度任务优先级的方式,实现最优目标实例资源的选择,最终完成科学工作流的代价与时间同步最优的均衡调度.

2 工作流与系统模型

2.1 工作流模型

以有向无循环图 DAG(Directed Acyclic Graph)表示工作流模型,定义为 $G=(T,E)$,其中, $T=\{t_0,t_1,\cdots,t_n\}$ 表示图的顶点代表的任务集, $E=\{e_{i,j} \mid t_i,t_j \in T\}$ 表示图的边集,代表任务间的数据或控制依赖性.

边 $e_{i,j} \in E$ 表示两个任务 t_i 与 t_j 间的有向边, $t_i,t_j \in T$,代表两个任务间的优先顺序约束,其具体含义是:仅当任务 t_i 执行完成后并收到 t_i 的所有数据后,任务 t_j 才可以开始执行.换言之,任务 t_i 是 t_j 的父任务, t_j 是 t_i 的后继或子任务.单个任务可拥有一个或多个父任务和子任务.直到其所有父任务完成后,子任务 t_i 才可以开始执行.

2.2 系统模型

云资源提供者可提供多种类型的实例,拥有不同的CPU、内存、存储和带宽能力,且配置不同的使用价格.本文使用基于Amazon EC2(Amazon Elastic Compute Cloud)的资源模型,其实例类型按需提供,其定价利有即付即用的最小小时计费模型,即:小于一小时的资源使用,均按一小时支付.表1是本文中所使用的资源代价与实例类型的相关参数配置,更新于2016年3月的Amazon EC2数据.

假设云资源提供者可提供无限制异构实例数量,表示为 $P=\{p_0,p_1,\cdots,p_h\}$,其中, h 表示实例类型索引.假设所有实例类型和存储服务处于同一云数据中心内,则不同实例间的平均带宽可视为基本相等的.

表 1 资源实例类型

类型	ECU	内存/GB	代价/\$
m3. medium	3	3.75	0.067
c4. large	8	3.75	0.105
c3. xlarge	14	7.5	0.21
m4. 2xlarge	26	32	0.479
c4. 4xlarge	62	30	0.838
c3. 8xlarge	108	60	1.68

注:ECU:Elastic Compute Unit,弹性计算单元.

3 预算分割工作流均衡调度算法

基于预算分割的均衡工作流调度算法 BDWTS(Budget Division Workflow Trade-off Scheduling)划分为以下四个阶段:

- (1)工作流分级:基于工作流任务的同步需求,将工作流任务进行分级;
- (2)预算分割:将用户定义的预算按六种不同的策略在不同的任务分级间进行分割;
- (3)任务选择:在任务就绪队列中基于优先级原则选择调度任务;
- (4)实例选择:选择最优目标实例执行任务,满足可用预算约束.

3.1 工作流分级

工作流分级的目标是通过任务分级最大化任务执行的并行程度,使得同一分级中的任务与其它分级中的任务不具有依赖性.因此,每个分级可视为包括独立任务的批任务集 BoTs(Bag of Tasks).

定义任务 t_i 的分级为以自顶向下的方式,任务 t_i 至工作流的出口任务间的所有路径的边的最大值,表示为 $N_L(t_i)$.对于出口任务,分级为1,对于其他任务:

$$N_L(t_i) = \max_{t_j \in \text{succ}(t_i)} \{N_L(t_j) + 1\} \tag{1}$$

式中, $\text{succ}(t_i)$ 表示任务 t_i 的直接后继子任务集合.然后,根据任务分级,将所有拥有相同分级的任务组成任务分级集合 TLS(Task Level Sets),则:

$$\text{TLS}(l) = \{t_i \mid N_L(t_i) = l\} \tag{2}$$

式中, l 表示分级数,且 $l \in [1,\cdots,N_L(t_{\text{entry}})]$.

3.2 预算分割

基于工作流的分级结果,预算分割的目标是将用户定义的工作流预算在不同分级任务进行子划分. 本文设计了以下六种预算分割算法:

- (1)随机分割算法 Random:预算随机分配于工作流的不同分级上.
- (2)平均分割算法 Uniform:每个工作流分级得到用户预算的 $1/L$, 其中, L 表示工作流总分级数.
- (3)高度正比例分割算法 Height:每个工作流分级得到的预算分割正比于工作流分级与入口节点的距离,则该距离越小,预算分割越小.
- (4)宽度正比例分割算法 Width:每个工作流分级得到的预算分割正比于该分级中任务的数量.
- (5)区域正比例分割算法 Area:联合 Width 算法和 Height 算法设置每个分级的预算分割.
- (6)全进分割算法 All in:将用户预算全部分配至入口任务,然后将剩余预算依次传递至下一分级. 该算法可视为 Height 算法的精炼算法.

以下通过一个算例说明以上六种算法所描述的预算在不同分级间的分割方法. Random 算法是随机产生预算分割,因此不作额外说明. 图 1 所示是一个拥有 10 个任务的工作流结构,图中左侧一栏是通过公式(1)计算得到的工作流分级值,右侧一栏是从出口任务开始对每一级任务的标记. 该工作流中,最大分级 $N_{\max}=5$. 同时,设置用户预算 $Budget=165$.

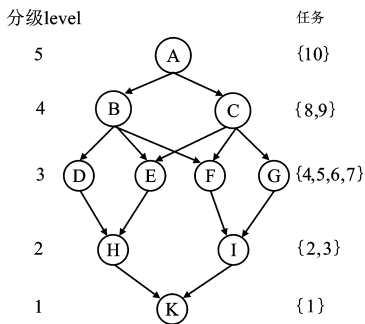


图 1 工作流示例

每种算法得到的预算分割如下所示(表 2 给出预算分割的完整情况):

- Uniform 算法:由于工作流共有 5 个分级,故每个分级获得 $165/5$ 的预算分割.
- Height 算法:每个分级分配一个相对于工作流分级的预算加权值,表示为:
$$L_{\text{weight}} = \sum_{k=1}^{N_{\max}=5} k = 15$$
预算因子 BF(Budget Factor)为:

$$BF = \frac{\text{budget}}{L_{\text{weight}}} = \frac{165}{15} = 11$$

例如:level 4 包括任务 B 和 C,则所分配的预算为 $4 \times BF = 4 \times 11 = 44$.

Width 算法:每个分级得到的预算分割取决于对应分级中的任务数量,此时的预算因子为:

$$BF = \frac{\text{budget}}{\text{tasksnumbers}} = \frac{165}{10} = 16.5$$

例如:level 4 包括两个任务,则所分配的预算为 $2 \times BF = 2 \times 16.5 = 33$.

Area 算法:该算法的预算分割为 Height 和 Width 算法的联合,计算为:

$$L_{\text{weight}} = \sum_{k=1}^{10} k = 55$$

预算因子 BF 为:

$$BF = \frac{\text{budget}}{L_{\text{weight}}} = \frac{165}{55} = 3$$

然后,预算根据图 1 中右栏中的数值之和进行分割. 例如:level 3 的预算分割为 $(4+5+6+7) \times BF = 22 \times 3 = 66$.

All in 算法:总预算分配至 level 5. 调度该分级上的所有任务后,剩余预算传递至下一分级中.

表 2 算法的预算分割情况

	预算分割算法				
	Uniform	Height	Width	Area	All in
Level 5	$165/5=33$	$5 \times BF=55$	$1 \times BF=16.5$	$10 \times BF=30$	165
Level 4	$165/5=33$	$4 \times BF=44$	$2 \times BF=33$	$17 \times BF=51$	0
Level 3	$165/5=33$	$3 \times BF=33$	$4 \times BF=66$	$22 \times BF=66$	0
Level 2	$165/5=33$	$2 \times BF=22$	$2 \times BF=33$	$5 \times BF=15$	0
Level 1	$165/5=33$	$1 \times BF=1$	$1 \times BF=16.5$	$1 \times BF=3$	0

3.3 任务选择

由于任务的分级特征,这表明只有前一分级中的所有任务调度之后,该任务才能开始执行. 而同一分级中的任务间不存在依赖性,因此,均可以准备执行,置入就绪任务队列中. 为了从中选择调度任务,需要为就绪队列中的任务分配优先级. 本文设计一种基于最早开始时间 EST(Earliest Start Time)的优先级任务选择算法.

如图 2 所示的工作流示例,边上的数值表示任务间的数据传输时间. 根据任务 A, B, C 的执行,所有子任务置入就绪队列中. 由于任务 A 和 C 在相同实例上执行,两个任务间的数据传输时间为 0. 然而,任务 B 必须等待接收其父任务的数据才能开始. 任务 D 可以分别在时间 13 和 9 时在实例 p_0 和

p_1 上开始执行. 任务 E 在实例 p_0 上的最早开始时间为 13, 在实例 p_1 上为 14. 因此, 任务 D 赋予更高优先级.

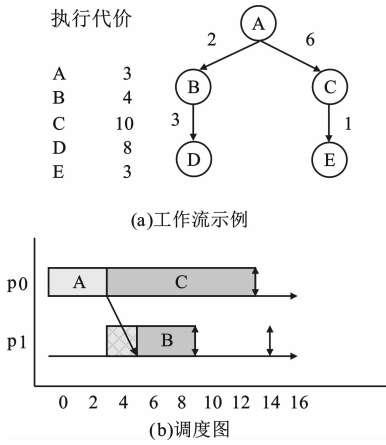


图 2 任务选择示例

任务 t_i 在执行时间最短的实例上的最早开始时间 EST 定义为:

$$EST(t_i) = \begin{cases} 0, & t_i = t_{entry} \\ \max_{t_j \in pred(t_i)} \{EST(t_j) + w_{t_j} + C_{i,j}\}, & \text{否则} \end{cases} \quad (3)$$

式中, w_{t_j} 表示任务 t_j 在最快实例上的执行时间; $pred(t_i)$ 表示任务 t_i 的父任务集合; $C_{i,j}$ 表示任务 t_i 与 t_j 间的传输数据的通信时间.

任务选择从入口任务 t_{entry} 所在的分级开始执行, 第一分级执行之后, 下一分级中的所有任务被置入待调度的就绪队列中.

3.4 实例选择

BDWTS 的目标是在满足用户截止时间的同时最小化工作流执行时间. 每一分级获得的预算分割为该分级上任务所能花费的最大预算. 为了选择执行任务的最优目标实例, 算法通过公式(4)和(5)计算每个任务在各个实例上的执行代价 Cost 集和执行时间 Time 集.

$$Cost_{t_i}^{p_j} = \frac{subBudget - C_i}{subBudget - C_{best}} \quad (4)$$

式中, $subBudget$ 表示该分级的预算分割, C_i 表示当前任务 t_i 调度至实例 p_j 上的代价, C_{best} 表示所有实例中执行当前任务的最小代价.

$$Time_{t_i}^{p_j} = \frac{ECT(max) - ECT(t_i, p_j)}{ECT(max) - ECT(min)} \quad (5)$$

式中, $ECT(t_i, p_j)$ 表示当前任务 t_i 在实例 p_j 上的执行时间, $ECT(max)$ 和 $ECT(min)$ 分别表示在所有实例上执行当前任务的最大和最小时间.

为了寻找最优实例, 利用时间 / 代价均衡因子 TCTF(Time Cost Trade-off Factor) 进行度量, 表示为:

$$TCTF_{t_i}^{p_j} = Time_{t_i}^{p_j} / Cost_{t_i}^{p_j} \quad (6)$$

4 仿真实验

4.1 实验配置

利用 CloudSim 对算法进行仿真分析, 云环境配置为一个数据中心, 六种不同实例类型, 具体配置见表 1 所示. 实例带宽固定设置为 20 MBps, EC2 的处理能力以每秒百万浮点操作数 MFLOPS (Million Floating Point Operations Per Second) 度量.

为了评估 BDWTS 算法中预算分割策略对性能的影响, 设置不同的预算范围, 调度工作流的最小可能预算通过下式计算:

$$Lowset_{cost} = \sum_{t_i \in G} Cost_{t_i}^{p_j} \quad (7)$$

式中, p_j 表示价格最低的实例. 通过将所有任务调度至最低代价的实例上执行即可获得该值. 该方式可以得到执行代价最低的调度方式. 利用该最低代价, 可以计算最小预算为:

$$budget = \alpha \times Lowset_{cost}, 1 < \alpha < 10 \quad (8)$$

设置工作流任务数量为 1 000, 利用 Pegasus 工作流产生器^[9] 创建与科学工作流 LIGO 相似的合成工作流结构作为测试数据源, 其结构如图 3 所示.

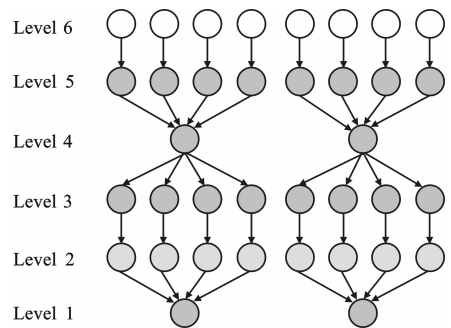


图 3 LIGO 工作流结构图

4.2 结果分析

对于拥有 1 000 个任务 LIGO 工作流, 每一分级中的任务数量、预算分割与剩余预算如表 3 所示. 例如: 利用 Height 算法调度第一分级中的所有任务后, 剩余预算 0.15 被增加至下一分级的预算中, 导致该分级预算为 1.69. 表最后一行表示每种策略的执行时间.

表 3 算法的预算分割情况

预算分割策略											
		Uniform		Height		Width		Area		All in	
	任务数量	预算分割	剩余预算	预算分割	剩余预算	预算分割	剩余预算	预算分割	剩余预算	预算分割	剩余预算
Level 6	229	1.173	0.01	2.01	0.15	1.61	0.01	2.85	0.01	7.035	0.001
Level 5	229	1.173	0.03	1.675	0.01	1.61	0.03	2.11	0.03	0.001	0.001
Level 4	21	1.173	0.05	1.34	0.03	0.14	0.006	0.15	0.01	0.001	0.001
Level 3	250	1.173	0.01	1	0.02	1.75	0.01	1.39	0.02	0.001	0.001
Level 2	250	1.173	0.03	0.67	0.009	1.75	0.02	0.51	0.06	0.001	0.001
Level 1	21	1.173	0.05	0.335	0.02	0.14	0.02	0.003	0.05	0.001	0.001
执行时间		938.97		798.71		798.61		682.6		603.93	

表 4 为不同预算分割策略的实例使用数量情况. 可以看出, All in 算法分配了更小数量的高性能实例, 降低了数据传输代价, 得到了最少的执行时间(表 3). 对于用户而言, 更倾向于寻找满足预算且执行时间更少的调度方案, 而对于资源提供方而言, 更倾向于最大化资源利用率. 尽管不同的预算分割方法下的 BDWTS 算法均可以得到可行解, 但所使用的实例类型和数量是各不相同的. 开启更多实例并不一定会带来更低的执行时间, 相反, 会导致更高的调度开销和低利用率. 因此, 预算分割策略对资源利用率是有直接影响的.

表 4 不同预算分割策略的实例使用情况

预算分割算法					
实例类型	Uniform	Height	Width	Area	All in
m3. medium	1	1	6	2	0
c4. large	6	3	2	2	1
c3. xlarge	6	3	2	3	1
m4. 2xlarge	0	2	2	2	0
c4. 4xlarge	6	2	2	2	0
c3. 8xlarge	0	2	2	2	4
实例数量	19	13	16	13	6
总执行代价	6.985	7.006	7.026	6.968	7.035

表 5 给出了 LIGO 工作流的执行时间. 可以看出, All in 策略在两种性能指标均优于其他算法, 而 Random 策略则是性能最差的. 图 4 是本文算法与 BHEFT 算法在代价上的比较结果, 此时 BDWTS 的预算分割采取 All in 策略, 显然, 通过不同分级上的预算分割的 BDWTS 获得了更低的代价.

5 结束语

为了解决云环境中科学工作流的调度优化问题, 提出一种工作流均衡调度算法. 算法集中于解决

工作流预算约束下的代价与时间优化问题, 算法通过将寻找最优调度方案划分为工作流分级、预算分割、任务选择和实例选择四个阶段, 实现了预算约束下的时间与代价均衡调度. 实验结果证明了算法的有效性.

表 5 执行时间

预算分割算法					
实例类型	Uniform	Height	Width	Area	All in
m3.medium	1	1	6	2	0
c4.large	6	3	2	2	1
c3.xlarge	6	3	2	3	1
m4.2xlarge	0	2	2	2	0
c4.4xlarge	6	2	2	2	0
c3.8xlarge	0	2	2	2	4
实例数量	19	13	16	13	6
总执行代价	6.985	7.006	7.026	6.968	7.035

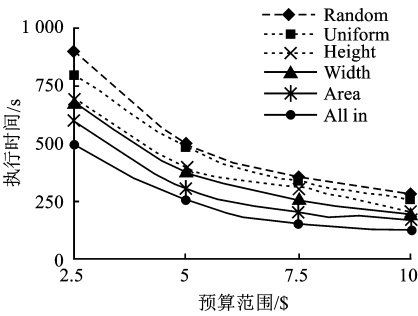


图 4 执行代价

参考文献:

[1] Wu F, Wu Q, Tan Y. Workflow scheduling in cloud: a survey[J]. Journal of Supercomputing, 2015,71(9): 1-46.

[2] Zheng W, Sakellariou R. Budget-Deadline Constrained Workflow Planning for Admission Control[J]. Journal

- of Grid Computing, 2013, 11(4):633-651.
- [3] Arabnejad V, Bubendorfer K. Cost Effective and Deadline Constrained Scientific Workflow Scheduling for Commercial Clouds[C]. In International Symposium on Network Computing and Applications. IEEE, 2016:106-113.
- [4] Arabnejad V, Bubendorfer K, Ng b, et. al, A deadline constrained critical path heuristic for cost-effectively scheduling workflows[C]. In 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing, IEEE, 2015: 242-250.
- [5] Zeng L, Veeravalli B, Li X. Scalestar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud[C]. In 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, 2012:534 - 541.
- [6] Lin X, Wu C Q. On Scientific Workflow Scheduling in Clouds under Budget Constraint [C]. International Conference on Parallel Processing. IEEE, 2013: 90-99.
- [7] Wu C Q, Lin X, Yu D, 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.
- [8] Mao M, Humphrey M. Scaling and Scheduling to Maximize Application Performance within Budget Constraints in Cloud Workflows[C]. In International Symposium on Parallel & Distributed Processing. IEEE, 2013:67-78.
- [9] Juve G, Chervenak A, Deelman E, et al. Characterizing and profiling scientific workflows [J]. Future Generation Computer Systems, 2013, 29(3):682-692.

作者简介:

吴文兵 男, (1968—), 男, 博士, 副教授. 研究方向为云计算调度及其应用. E-mail: wenbing808@163. com.

(上接第 120 页)

- [2] He T, Chin K W, Soh S. On using Wireless Power Transfer to increase the max flow of Rechargeable Wireless Sensor Networks[C]// IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing. IEEE, 2015:1-6.
- [3] Rappaport T S. Wireless Communications: Principles and Practice[M]. Prentice Hall PTR, 1996.
- [4] Xiao Y, Chen H, Li F H. Handbook on Sensor Networks[M]. World Scientific, 2010.
- [5] Ahuja R K, Magnanti T L, Orlin J B. Network flows: theory, algorithms, and applications[J]. Journal of the Operational Research Society, 1993, 45 (11): 791-796.
- [6] Goldberg D E. Genetic Algorithm in Search, Optimization, and Machine Learning[J]. 1989, xiii(7):2104 - 2116.
- [7] Srinivas M, Patnaik L M. Adaptive probabilities of crossover and mutation in genetic algorithms [J]. IEEE Transactions on Systems Man & Cybernetics, 2002, 24(4):656-667.
- [8] Ford L R J, Fulkerson D R. A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem[J]. Canadian Journal of Mathematics, 1955, 9:210-218.
- [9] Zhu C, Liu K, Yu C, et al. Simulation and experimental analysis on wireless energy transfer based on magnetic resonances[C]// Vehicle Power and Propulsion Conference. IEEE, 2008:1-4.
- [10] Heinzelman W R, Chandrakasan A, Balakrishnan H. Energy-Efficient Communication Protocol for Wireless Microsensor Networks [C]// Hawaii International Conference on System Sciences. IEEE Computer Society, 2000:4-7.

作者简介:

李天旭 男, (1995-), 硕士研究生. 研究方向为无线传感器网络、无线充电. E-mail: tianxuli@cumt. edu. cn.

肖 硕 男, (1981-), 博士, 副教授. 研究方向为无线传感器网络.