软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(10):2720−2734 [doi: 10.3724/SP.J.1001.2012.04198]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

异构分布式环境下多 DAG 工作流的混合调度策略*

田国忠[1,2+], 肖创柏[1], 徐竹胜[1], 肖 霞[1]

[1](北京工业大学 计算机学院,北京 100124)

[2](新疆工程学院 计算机工程系,新疆 乌鲁木齐 830091)

Hybrid Scheduling Strategy for Multiple DAGs Workflow in Heterogeneous System

TIAN Guo-Zhong[1,2+], XIAO Chuang-Bai[1], XU Zhu-Sheng[1], XIAO Xia[1]

[1](College of Computer Science, Beijing University of Technology, Beijing 100124, China)

[2](Department of Computer Engineering, Xinjiang Institute of Engineering, Urumqi 830091, China)

+ Corresponding author: E-mail: tiangz@emails.bjut.edu.cn

Tian GZ, Xiao CB, Xu ZS, Xiao X. Hybrid scheduling strategy for multiple DAGs workflow in heterogeneous system. *Journal of Software*, 2012,23(10):2720−2734 (in Chinese). http://www.jos.org.cn/1000-9825/4198.htm

Abstract: Recent research in multiple DAG workflows in heterogeneous systems have been making progress and have solved some problems, but fail to classify the multiple DAGs, according to the demand of the performance asked by the varied DAG workflow and also fail to address the scheduling multiple DAGs workflow with multiple priorities submitted at different times. To solve these problems, the paper presents a new model of multiple DAGs management system for multiple DAGs workflow with multiple priorities and an adjustment method to the previous Fairness algorithm to solve the fairness issue in scheduling multiple DAGs with the same priorities submitted at different times. In addition, the study also proposes an implementation method of the Backfill algorithm for multiple DAGs with different priorities to improve utilization rate of resource, and then, based on the new model and the two methods, propose a hybrid strategy for scheduling multiple DAGs with multiple priorities submitted at different times. These experimental results show that it is possible to meet different requirements of DAGs submitted at different times and to improve utilization rate of a resource. In addition, the results about scheduling two-DAGs show a significant “Trail Ending” principle, which is valuable for academic study and application.

Key words: multiple DAGs scheduling; multiple priorities; fairness; slots

摘 要: 关于多个 DAG 工作流在异构分布式环境下调度的研究近来有了新的进展,也解决了一些问题,但现阶段还没有考虑和解决根据不同类型 DAG 的需求按优先级进行分类,以及对不同时间到达的多个不同优先级 DAG 进行调度的问题.为解决这些问题,针对各用户对 DAG 工作流的 QoS 需求的不同,在对不同用户的 DAG 工作流进行优先级划分的基础上,首先提出了一种新的调度模型,并改进了已有的公平调度算法,解决在不同时间上被提交的具有相同优先级的多个 DAG 工作流之间调度的公平性问题.为了提高资源利用率和高优先级 DAG 尽可能小地受低优先级 DAG 的影响,又提出了一种适用于多个不同优先级 DAG 之间调度的 Backfill 算法.在新的系统模型和这两种算法的基础上,提出了一种混合调度策略.实验结果表明,这种混合调策略能够兼顾不同时间到达的多个不同类型 DAG 调度需求和资源利用率的改善.另外,通过实验发现了关于两个 DAG 调度所特有的“拖尾”规律,具有进一步研

* 基金项目: 国家高技术研究发展计划(863)(2009AA01Z437); 北京市自然科学基金重大项目(4110001); 国家自然科学基金(60863003, 61063042)

收稿时间: 2011-04-04; 修改时间: 2011-07-21; 定稿时间: 2012-02-15

究和应用的价值.

关键词: 多 DAG 调度;多优先级;公平性;时隙
中图法分类号: TP311 文献标识码: A

有向无环图 DAG(directed acyclic graph)是工作流常用的一种描述方式[1],而工作流系统中最重要的一项关键技术就是 DAG 调度.近年来,关于单个 DAG 在异构分布式环境下的调度研究已经取得了很大进展[2−8],但这些算法不能直接运用于多 DAG 的调度[9],针对多个 DAG 工作流调度的研究还处于探索阶段.现有相关的多 DAG 调度模型与算法尽管提出和解决了一些重要的问题,但对更为复杂情况下的多 DAG 调度,如多个用户可能会在不同时间提交 DAG,且用户对 DAG 执行时间的要求可能差异较大,如何处理好已被部分调度执行的 DAG 和新到达 DAG 中各任务之间的关系,以更好地兼顾多个 DAG 之间调度的公平性和资源利用率的改善等问题,还没有得到较好的解决.为此,本文提出了适用于异构分布式环境下多个不同 DAG 随机提交的多优先级 DAG 调度模型和算法.

本文第 1 节阐述背景及相关工作.第 2 节提出新的多 DAG 调度系统模型.第 3 节介绍本文算法和策略.第 4 节给出实验说明和相关工作的比较分析.第 5 节是结论.

1 背景及相关工作

Kertész 等人[10]将工作流管理平台分为两大类:如果多用户同时以协作的方式监控、控制和执行同一个任务图,则称为 MC 类型;如果所管理运行的多用户之间的 DAG 任务图互相独立,则称为 MI 类型,即多 DAG 工作流系统(多租户 DAG 工作流系统).针对多 DAG 工作流的调度,目前已有的研究解决了多 DAG 工作流调度相关的一些问题,其中一个简单的办法是将所有的 DAG 放入一个队列里按顺序调度,将队列中的一个 DAG 利用单 DAG 调度方法将所有任务调度完毕后,再调度下一个 DAG,但这种方法不能充分利用分布式环境中可利用的资源.Honig 等人[11]为了改善资源的利用率和提高多 DAG 任务调度的吞吐量,提出了将多个 DAG 合成一个复合 DAG 后,再利用单 DAG 调度方法来调度合成的 DAG,但这种方法存在各 DAG 之间调度的不公平性问题. Zhao 和 Sakellariou[9]在 Honig 方法的基础上,首次提出了多 DAG 调度中存在的多个 DAG 之间的公平性问题,为此提出了关于多 DAG 调度公平性的定义方法和 Fairness(公平)算法.简单来说,该算法是要先调度多个 DAG 中相对滞后的那个 DAG 中权值最大的任务.具体做法是,在每次调度一个任务后,比较各 DAG 的 *SlowDown* 值来决定下一次将调度哪一个 DAG 中的任务,从而较好地解决了多个 DAG 调度的公平性问题.实验数据表明,他们的 Fairness 算法能够使得多 DAG 调度的公平程度最接近于 Kleinrock 提出的理论极限值[12],但该算法没有进一步考虑不同用户在不同时间提交 DAG 的情况.在很多情况下,不同用户提交 DAG 的时间是随机的,不一定是在同一时间提交,系统如果让先到达的 DAG 等待其他 DAG 到来后同时调度,显然会降低资源利用率和 DAG 的吞吐量.Yu 和 Shi[13]为了调度不同用户在不同时间提交的多个 DAG 和 Honig 等人[11]的合成方法中可能出现的不公平性调度问题,提出了动态的调度模型和策略.采用的方法是,将所有具备执行条件的任务根据任务向上权值的大小按降序或升序放入调度队列中,如果这些具备执行条件的任务来自同一个 DAG,则调度队列中的任务按降序排列,否则按升序排列,从而避免早到达的 DAG 中权值小的任务得不到调度的问题.然而,这种方法仍会造成不同程度的不公平问题,因为如果后续被提交的 DAG 任务权值总是低于那些先来 DAG 中未被调度的任务权值,那么先来的 DAG 剩余任务将得不到调度,所以只适合于多个 DAG 之间结构和复杂程度相近的情况.

可以看出,这些相关的研究工作虽然提出并解决了一些重要问题,但都还没有很好地解决在 DAG 复杂多样和用户需求不同等情况下随机提交的多个 DAG 调度的公平性和资源利用率的改善问题.这是一个新的问题,需要新的模型和算法来解决.

在异构分布式环境下,用户提交的 DAG 工作流一般由一个工作流管理系统统一进行管理和调度[11,14,15].从系统管理方面来看,工作流管理系统负责处理用户工作流的提交、资源管理和分配、数据移动和工作流的调度,并且尽可能地提高资源的利用率和工作流任务的吞吐量.而从用户方面看,不同的用户有着不同的 QoS 需求.在

网格和云计算环境下,资源的提供者会根据用户使用资源的情况向用户索取相应的费用,所以一些工作流的用户会尽量将费用支出控制在一个预算范围里[16−19],而对执行时间的要求并不是很严格.另外,在一些分布式的实时应用中,如金融市场分析、雷达系统、多媒体等应用的多个任务之间也存在先后限制关系,并且整个实时 DAG 工作流的执行期限有着较为严格的时间限制,如果不能在规定的时间内完成工作流任务,执行结果将会失去意义[20].因此,不同用户对所提交 DAG 的执行期限要求的严格程度是不一样的,将这些不同 QoS 需求的 DAG 按照执行期限的严格程度进行优先级的划分是必要的.但目前在这些关于多 DAG 调度的研究[9−11,13]中,还没有由于根据不同用户的 QoS 需求不同以及为了进一步提高资源利用率而将多个不同的工作流进行优先级划分,从而造成了一定程度上资源利用率的不足.本文针对不同用户提交的 DAG 工作流有不同的 QoS 需求,主要是对执行时间和支出费用的要求不同,将用户提交的 DAG 根据调度优先级水平从高到低分为以下 3 类:RT(实时)-DAG,它对整个 DAG 的执行期限有着较为严格的时间限制;TO(时间优化)-DAG,它要求最小化执行时间,而不考虑费用支出;CO(费用优化)-DAG,它要求在最小化执行费用支出条件下,执行时间尽可能地短.对多 DAG 调度系统来说,在用户已经提交的多个 DAG 进行资源分配调度后,仍然会有新的 DAG 随时到达,需要系统进行调度,这个时候,调度系统应该能够对 DAG 的优先级加以识别,并根据这些 DAG 的优先级情况来决定是否需要将较早到达并已经分配资源的 DAG 中的部分任务撤销,处理好这些已被部分调度执行的 DAG 和新到达的 DAG 中各任务之间的关系,在兼顾公平性和提高资源利用率的基础上,尽可能地满足各种类型用户的需求.为此,本文提出了一个新的多 DAG 调度管理系统的模型——一种适用于不同优先级 DAG 之间调度的 Backfill 算法、一种适用于同级 DAG 之间调度的改进 Fairness 算法和基于这两种算法基础上的 MMHS(hybrid scheduling strategy for multiple DAGs workflow with multiple priorities,多优先级多 DAG 混合调度)调度策略.

2 多优先级多 DAG 调度系统模型

本文提出的新多优先级多 DAG 工作流管理系统模型如图 1 所示,它由 3 部分组成,分别是:多 DAG 工作流接收与优先级判别子系统——Receiver & Priority Identification;多 DAG 工作流调度子系统——Multi-DAG Scheduler 和对应于资源 M_i 的等待执行任务队列组——Q_{W-M_i} :Set of Queue of Tasks to Be Executed on M_i.在多租户 DAG 工作流系统中,不同用户会在不同时间提交 DAG.这些 DAG 到达后,由工作流接收与优先级判别子系统来管理和识别新 DAG 的优先级,并与较早时间到达的正在被调度执行的 DAG 优先级进行比较,为多 DAG 工作流调度子系统调度提供调度信息.然后,由多 DAG 工作流调度子系统根据本文提出的混合调度策略(详见第 3.3 节)将这些多 DAG 的任务按照执行顺序分配至系统的第 3 部分,即待执行任务队列组 Q_{W-M_i} 中.每次资源 M_i 执行完一个任务后,依次从与其相对应的 Q_{W-M_i} 中取下一个任务执行.

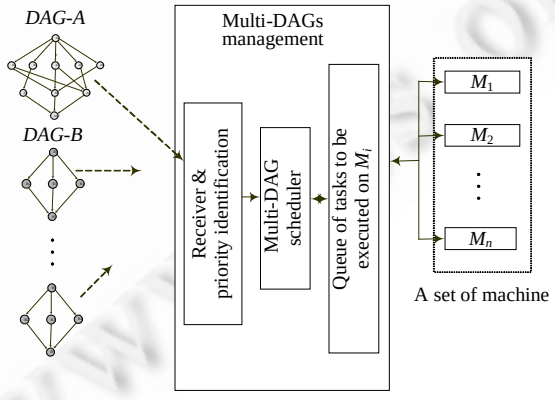
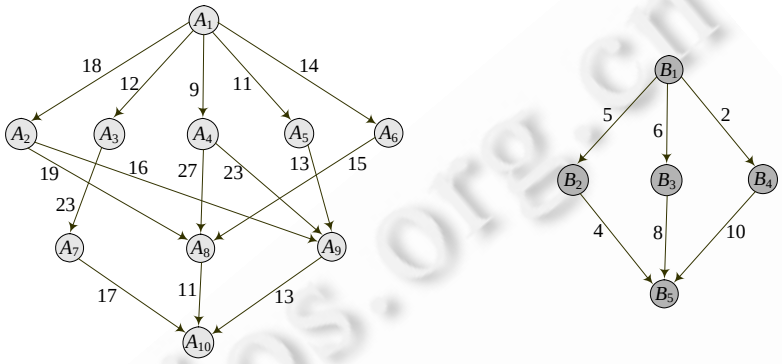


Fig.1 Management system model for DAGs with multiple priorities

图 1 多优先级 DAGs 管理系统模型

本文关于 DAG 工作流任务图的表示、定义和向上权值 $rank_u(n_i)$等与文献[2,13]的有关定义和方法一致,这里不再重复.在本文的多租户多 DAG 系统模型中,资源的提供者是根据用户任务的运行时间向用户收取费用的.对 CO-DAG 类型的用户来说,首要的 QoS 需求是整个 DAG 的执行费用最低.如果 p_{M_i} 表示资源 M_j 的单位时间的租用价格,则任务 n_i 在资源 M_j 上所发生的费用为 $E_{ij} = p_{M_i} \cdot w_{ij}$ (w_{ij} 表示任务结点 n_i 在机器资源 M_j 上的估计运行时间,参见文献[2]).

下面通过图 2 的两个工作流实例 *DAG-A* 和 *DAG-B* 的调度来说明本文所提出的调度方法和策略.这两个 DAG 的复杂度、结构和各项参数基本与文献[2,9]中的实例一样,机器资源数为 3 个.假设两个 DAG 中每项任务在每个机器资源上的执行时间见表 1,那么计算可得到每个任务的向上权值 $rank_u(n_i)$,见表 2.



(a) *DAG-A* (b) *DAG-B*

Fig.2 Two example of DAG-based workflow

图 2 两个 DAG 工作流的实例

Table 1 Computation time of *DAG-A* and *DAG-B* on each machine

表 1 *DAG-A* 和 *DAG-B* 中各任务在机器上的执行时间

DAG	*DAG-A*										*DAG-B*				
n_i	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	B_1	B_2	B_3	B_4	B_5
M_1, p_{M_i} =1	14	13	11	13	12	13	7	5	18	21	4	9	18	21	7
M_2, p_{M_i} =1.2	8	19	13	8	13	16	15	11	12	7	5	10	17	15	6
M_3, p_{M_i} =1.1	16	18	19	17	10	9	11	14	20	16	6	11	16	19	5

Table 2 Rank of each task of the two DAGs in Fig.2

表 2 两个 DAG 的任务向上权值表

DAG	*DAG-A*										*DAG-B*				
n_i	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	B_1	B_2	B_3	B_4	B_5
Rank	107.67	77	80	80	69	63.33	42.67	35.67	44.33	14.67	42	20	31	34.33	6

本文的模型和算法除了要利用以上基本的定义和参数以外,还用到另一个重要定义,即公平性.多 DAG 调度的一个重要功能是通过一定的方法确定各 DAG 中各任务的调度和执行顺序,既涉及到同一 DAG 内前后有依赖关系任务的调度顺序,也涉及到分属不同DAG没有依赖关系任务的调度顺序,这些调度顺序会直接影响各个 DAG 的执行时间;而后一种调度顺序是否公平,也会直接影响到各个 DAG 的平均执行时间和调度系统整体性能的好坏.如果调度不考虑公平性,可能往往会因为多个同级 DAG 之间的结构差异较大而使得任务量小的 DAG 完成时间比任务量大的完成时间要晚,造成明显的不公平.如第 1 节提到的文献[11,13]中的方法,仅根据任务的权值来确定调度顺序,很可能会造成先到达的 DAG 部分任务因为后续的 DAG 任务权值总是很大而得不到调度.所以,除了资源利用率,DAG 的执行时间、公平性也是衡量多 DAG 调度算法性能的重要指标.Zhao 和

Sakellariou[9]首次提出了衡量这种公平性的方法,基于这种定义方法上的调度算法,能够达到更好的公平性.以下是文献[9]中关于公平性的表述和定义:

由于一个 DAG a 要与其他 DAG 争用同一组机器,所以 a 的 *Makespan*(从提交 DAG a 开始到 DAG a 的最后一个任务执行完毕所用的时间)很可能比 a 单独使用这组机器的 *Makespan* 要长,那么这两个 *Makespan* 可分别被表示为 $M_{multi}(a)$和 $M_{own}(a)$.根据文献[10]的定义:$Slowdown(a)=M_{own}(a)/M_{multi}(a)$,那么某种调度算法 S 的不公平程度 $Unfairness(S)=\sum_{\forall a\in A}|Slowdown(a)-AvgSlowdown|$,其中,$A$ 为给定的多个 DAG 的集合;*AvgSlowdown* 是所有 DAG 的 *Slowdown* 平均值,即 $AvgSlowdown=\frac{1}{|A|}\sum_{\forall a\in A}Slowdown(a)$, $|A|$表示集合 A 的基.$Unfairness(S)$是可以用来衡量一种多 DAG 调度算法 S 调度的不公平程度的重要指标,本文将沿用这一定义,并将对其 Fairness 算法进行改进.

3 新的方法与混合调度策略

3.1 多DAG调度Fairness算法的改进

如前所述,Zhao 的 Fairness 算法不能处理不同时间到达的多个 DAG,也不适用于多优先级的多个 DAG 的应用情况.为了能够将任何时间到达的新 DAG 与已经全部预分配资源且部分任务已经执行的 DAG 同时公平地进行调度,本文提出了对 Fairness 算法的改进算法 E-Fairness.以下将通过图 2 和图 3 来讨论改进方法.

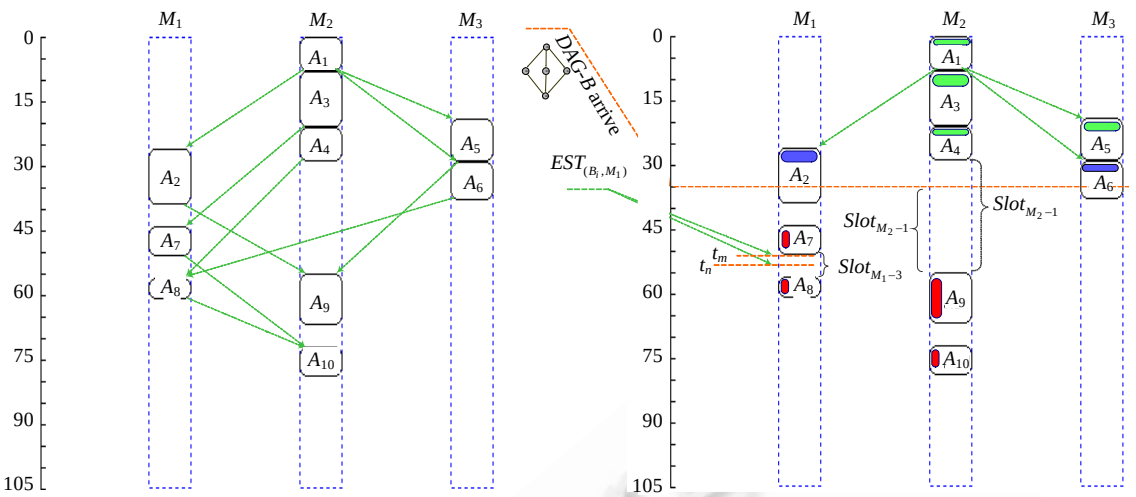
如果系统在 0 时刻只有图 2 中的 *DAG-A* 到达请求调度,利用 Topcuoglu 的 HEFT 算法[2]对 *DAG-A* 进行调度并执行,且在整个 *DAG-A* 调度执行过程中没有任何其他 DAG 到达,那么 *DAG-A* 的调度结果如图 3(a)所示.$M_{own}(DAG\text{-}A)=79$,$M_{multi}(DAG\text{-}A)=79$,$Slowdown(DAG\text{-}A)=1$.但是,如果当 *DAG-A* 在调度执行过程中,新的 *DAG-B* 到达,假设它的到达时间 $B_{ar\text{-}t}=35$,且 *DAG-B* 的优先级与 *DAG-A* 相同,那么如图 3(b)所示,*DAG-A* 中的所有任务可以分为 3 部分:(1) 执行完毕的任务组(A_1,A_3,A_4 和 A_5);(2) Q_{W-M_i} 中等待执行的任务组(A_7,A_8,A_9 和 A_{10});(3) 机器 M_i 上正在执行的任务组(A_2,A_6).由于机器上正在执行的任务具有不可剥夺性[2],所以在 M_1 和 M_3 上的 A_2 和 A_6 这两个任务不能被撤销,但是在 Q_{W-M_i} 中等待执行还未被执行的任务组(A_7,A_8,A_9 和 A_{10})能够被撤销并且可以与新到达的 *DAG-B* 一起进行公平调度.考虑到原有 *DAG-A* 部分任务的撤销和重新调度可能会导致数据传输问题,因为撤销的未被执行的任务组中的任务有可能重新被调度到一个新的机器资源上,这时已经执行完毕的父母任务结点的数据是否能够及时送达是非常重要的.为了解决这个问题,本文提出的多 DAG 系统模型要求:当一个有后继结点的任务完成后,它的执行结果数据必须向每个机器发送.另外,改进的 Fairness 算法的重要的一个方面就是对新 DAG 到达后时隙的调整.由于 Fairness 调度策略基于 HEFT 方法,在本例中,$B_{ar\text{-}t}=35$,B 中的所有任务和 A 中要重新调度的任务中任何一个任务的执行时间不能早于这个时间,所以在图 3(b)中,机器 M_2 上的第一个时隙(浅灰色 $Slot_{M_2-1}$)应该调整为新的可用时隙(深灰色 $Slot_{M_2-1}$),它的开始时间应被重置为 $B_{ar\text{-}t}$.

另外,Zhao 的 Fairness 算法中首先要对每个 DAG 的 *Slowdown* 值初始化为 0,所有 DAG 调度的初始顺序为各个 DAG 单独调度时 *Makespan* 大小的降序排列.只有 DAG 被调度 1 次,DAG 的 *Slowdown* 值发生变化后,才能实现多个 DAG 之间的公平性调度.对 Fairness 另一重要调整的步骤是,新到达的 DAG 必须首先被调度 1 次.在本例中,$A_{ar\text{-}t}=0$,$B_{ar\text{-}t}>0$,针对所有 *DAG-A* 中被撤销的任务和所有 *DAG-B* 中的任务,新到达 B 中向上权值最大的 B_1 任务首先被调度 1 次.

最后一个要调整的步骤是,计算新达到 DAG 中第 i 个任务 $T_{DAG(new)-i}$ 被调度后的 $Slowdown(T_{DAG(new)-i})$的计算要考虑到新达到 DAG 的到达时间 $DAG(new)_{ar\text{-}t}$,即

$$Slowdown(T_{DAG(new)-i})=M_{own}(T_{DAG(new)-i})/(M_{multi}(T_{DAG(new)-i})-DAG(new)_{ar\text{-}t}).$$

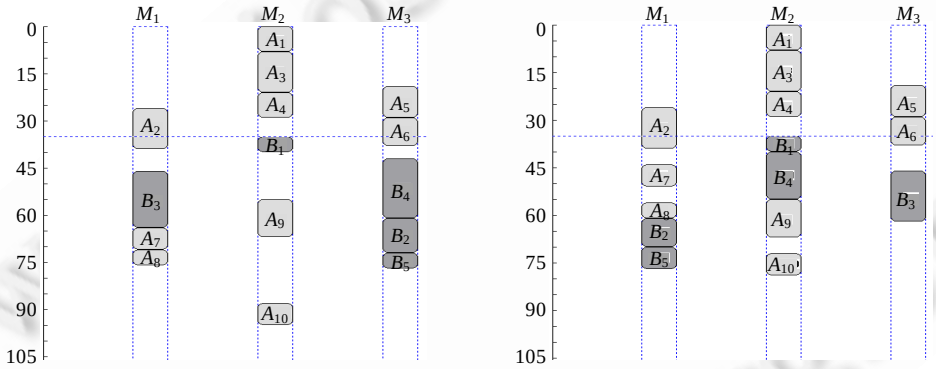
最后的调度结果和 Gannt 图分别如表 3 和图 4(a)所示.



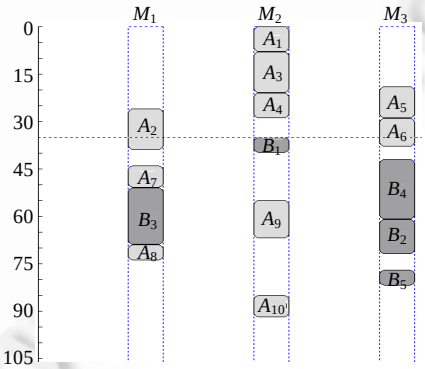
(a) *DAG-A* 的单独调度结果　　(b) *DAG-B* 在 35 时刻到达时 *A* 的任务状态和时隙举例

Fig.3　A example of *DAG-A* and *DAG-B* scheduling

图 3　*DAG-A* 和 *DAG-B* 调度举例



(a) E-Fairness 算法　　(b) Backfill 算法



(c) Round-Robin 算法

Fig.4　Gannt chart of scheduling *DAG-A* and *DAG-B* of 3 algorithms

图 4　3 种算法调度 *DAG-A* 和 *DAG-B* 的 Gannt 图

Table 3 Results of scheduling *DAG-A* and *DAG-B* of 3 algorithms ($B_{ar\text{-}t}$=35)

表 3 3 种算法调度 *DAG-A* 和 *DAG-B* 结果比较($B_{ar\text{-}t}$=35)

算法	E-Fairness(*B* 优先级=*A* 优先级)	Backfill(*B* 优先级<*A* 优先级)	Round-Robin
两个 DAG 的任务调度次序	$A_1, A_3, A_4, A_2, A_5, A_6, B_1, A_9, B_4, B_3, B_2, B_5, A_7, A_8, A_{10}$.	$A_1, A_3, A_4, A_2, A_5, A_6, A_9, A_7, A_8, A_{10}, B_1, B_4, B_3, B_2, B_5$.	$A_1, A_3, A_4, A_2, A_5, A_6, B_1, A_9, B_4, A_7, B_3, A_8, A_{10}, B_2, B_5$.
资源利用率	0.526 32	0.607 60	0.543 48
Makspan(*A*+*B*)	95.000 00	79.000 00	92.000 00
Unfairness	0.090 50	0.077 92	0.007 16
Makespan A	95	79	92
Makespan B	42	42	47

上述将 Fairness 算法调整为 E-Fairness 算法的主要调整步骤可以概括为以下几点:

当 $DAG(new)_{ar\text{-}t}>0$ 时($DAG(new)_{ar\text{-}t}$ 表示新 DAG 的到达时间):

(1) 撤销 Q_{W-M_i} 中所有未被执行的任务,以便与新到达 DAG 的任务一起进行调度;

(2) 修改相关的时隙;

(3) 新到达的 *DAG*(*new*)中向上权值最大的任务首先被调度 1 次;

(4) 新到达 *DAG*(*new*)的某个任务 *i* 被调度后 *Slowdown* 值计算如下:

$$Slowdown(T_{DAG(new)-i})=M_{own}(T_{DAG(new)-i})/(M_{multi}(T_{DAG(new)-i})-DAG(new)_{ar\text{-}t}).$$

3.2 多DAG的Backfill算法的实现

在图 3 中,同样地,如果 $B_{ar\text{-}t}$=35,但 *B* 的优先级不同于 *A*,那么应该采用 Backfill 算法来确定 *A* 和 *B* 的调度关系,而不能采用 E-Fairness 算法,因为优先级高的 DAG 可以中断先到达的并被正在调度的低优先级 DAG,不同优先级 DAG 之间比较调度顺序上的公平性没有意义.当 *B* 的优先级高于 *A* 时,应该撤销 *A* 中未被执行的任务组中的任务,并首先将资源按照 HEFT(最早完成时间)算法将所有机器分配给 *B*,然后将 *A* 中的被撤销的任务插入到 *B* 所留下的时隙中.如果 *B* 的优先级低于 *A*,那么 *A* 中的所有任务仍然按照在 0 时刻的调度方案进行调度,并将 *B* 中所有任务按照 HEFT(最早完成时间)算法匹配到各个机器 M_i 上时隙链表 S_{M_i} (按照时隙的开始时间排列)上,并将这些任务插入到 Q_{W-M_i} 中等待执行.当新 *DAG-B* 到达时,正如图 3(b)所示,相关的时隙应该被修改为可用时隙.在调度过程中,其他一些需要调整时隙的情况举例说明如下:

如在图 3(b)中, $EST_{(B_i,M_1)}$ 表示任务 B_i 的所有父母任务结果数据的最晚送达到机器 M_1 上的时刻,即任务 B_i 的最早开始时间.设 t_m 和 t_n 分别是 $EST_{(B_i,M_1)}$ 两个可能的取值,且 t_m 等于时隙 $Slot_{M_1-3}$ 的开始时刻,t_n 是时隙 $Slot_{M_1-3}$ 的等分时刻.假设 *B* 中某个任务 B_i 的长度恰好是 $Slot_{M_1-3}$ 的 1/3,且任务 B_i 被匹配到时隙队列 S_{M_1} 中的时隙 $Slot_{M_1-3}$ 上.那么,

- 当 $EST_{(B_i,M_1)}=t_m$时,$Slot_{M_1-3}$ 被 B_i 占用后仍然是 1 个时隙,只是时隙长度缩短了 1/3;
- 当 $EST_{(B_i,M_1)}=t_n$时,$Slot_{M_1-3}$ 被 B_i 占用后, $Slot_{M_1-3}$ 将会被分成两个更小的时隙.

当 $B_{ar\text{-}t}$=35,*B* 的优先级低于 *A* 时,采用 Backfill 算法的调度结果和 Gannt 图分别如表 3 和图 4(b)所示.

本文也利用了传统的 Round-Robin(轮转)算法对图 2 中两个 DAG 进行了调度.同样,如果 $B_{ar\text{-}t}$=35,则首先撤销 Q_{W-M_i} 中 *DAG-A* 的还未被执行的任务组(A_7,A_8,A_9 和 A_{10})原始分配方案,然后将 *DAG-B* 中所有任务与 *A* 中这些被撤销的任务一起通过 Round-Robin 算法进行调度,不考虑 DAG 的优先级关系.即 *A* 和 *B* 之间的先后调度关系由 Round-Robin 算法来确定,而每个 DAG 内部的任务间调度顺序和资源选择仍与 E-Fairness 和 Backfill 算法一样,通过 HEFT 方法来确定.利用 Round-Robin 算法的调度结果和 Gannt 图分别如表 3 和图 4(c)所示.从这两个 DAG 实例的 3 个算法调度图中可以看出,当 $B_{ar\text{-}t}$=35、*B* 优先级<*A* 优先级时,如果选用 Backfill 算法,不仅资源利用率比其他两种算法要好.更重要的是:如果 *A* 属于第 1 类 DAG,*B* 的到来并不会影响 *A* 的严格的执行时间期限要求.

另外,对于 CO-DAG 这一类型的 DAG,虽然不是本文讨论的重点,但这里也要必要作一些简要说明.图 4 是利用 3 种算法对图 2 中两个 DAG 实例进行调度的 Gannt 图,这 3 种算法都采用 HEFT 方法来确定 DAG 内部任

务间调度顺序和资源选择.但是,如果 *DAG-B* 属于 CO-DAG 类型,那么 *DAG-B* 中的任务就不能通过 HEFT 方法来调度,因为这一类型的 DAG 用户要求在费用支出最低的情况下,*Mackspan* 尽可能地短.所以,对 *DAG-B* 中的任务应该选用那些能够使 $E_{ij}=p_{M_i}\cdot w_{ij}$ 最小化的机器,本文称这种方法为 LE(lowest-expenditure)方法.

3.3 多优先级多DAG的混合调度策略——MMHS

正如上述算法 E-Fairness 和 Backfill 中所提到的,这两种算法在多优先级的多 DAG 调度中适用于不同的情况:E-Fairness 适合于确定相同优先级的 DAG 之间的调度关系,而 Backfill 适合确定不同优先级的 DAG 间的调度关系.因此如算法 1 所示,本文提出了适用于多优先级多 DAG 调度在不同时间到达系统的混合调度策略——MMHS.

算法 1.

While ($DAG_{new}\neq\varnothing$) **do** /*当新 DAG 到达*/

计算 DAG_{new} 中所有任务的向上权值 Rank;

If ($P_{new}\geqslant\max\{P_{scheduling}\}$) /*$P_{scheduling}$ 表示集合 $U_{scheduling}$ 中元素优先级的集合($U_{scheduling}$ 是正在调度运行的 DAG 的集合,并按照优先级从大到小排序),P_{new} 表示新达到 DAG 的优先级)*/

根据新 DAG 的到达时间修改相关时隙;

撤销 Q_{W-M_i} 中所有任务; /*正在运行 DAG 的所有任务包括对应于机器 M_i 的 Q_{W-M_i} 中未被调度的任务、已经执行完毕的任务和正在 M_i 上执行的任务*/

If ($P_{new}>\max\{P_{scheduling}\}$)

根据 HEFT 算法调度 DAG_{new} 的任务,并将 DAG_{new} 的任务依次放入 Q_{W-M_i};

While ($U_{scheduling}\neq\varnothing$) **do**

计算和修改所有机器上的时隙链表 S_{M_i};

a_{peer}←从 $U_{scheduling}$ 选取优先级最高的 DAG(注:最高级的可能有多个);

If (a_{peer} 中的 $DAG\notin CO\text{-}DAG$)

a_{peer} 中多个同级 DAG 间的调度关系采用 E-Fairness 算法,并采用 HEFT 方法的 Backfill 算法将这些任务匹配到 S_{M_i},并将任务依次放入 Q_{W-M_i}

Else

a_{peer} 中多个同级 DAG 间的调度关系采用 E-Fairness 算法,并采用 LE 方法的 Backfill 算法将这些任务匹配到 S_{M_i},并将任务依次放入 Q_{W-M_i};

$U_{scheduling}=U_{scheduling}-a_{peer}$;

End while

Else

If ($P_{new}=\max\{P_{scheduling}\}$)

a_{peer}←依次从 $U_{scheduling}$ 选取优先级最高的 DAG(注:最高级的可能有多个);

a_{peer} 中多个同级 DAG 间的调度关系采用 E-Fairness 算法

While ($U_{scheduling}\neq\varnothing$) **do**

计算和修改对应于每个机器的时隙链表: S_{M_i}

a_{peer}←依次从 $U_{scheduling}$ 选取优先级最高的 DAG;

If (a_{peer} 中的 $DAG\notin CO\text{-}DAG$)

a_{peer} 中多个同级 DAG 间的调度关系采用 E-Fairness 算法,并采用 HEFT 方法的 Backfill 算法将这些任务匹配到 S_{M_i},同时将任务依次放入 Q_{W-M_i}

Else

a_{peer} 中多个同级 DAG 间的调度关系采用 E-Fairness 算法,并采用 LE 方法的 Backfill 算法将这些任务匹配到 S_{M_i},同时将任务依次放入 Q_{W-M_i};

$U_{scheduling}=U_{scheduling}-a_{peer}$

End while

Else

If ($P_{new}<\max\{P_{scheduling}\}$)

仅撤销 Q_{W-M_i} 中优先级小于或等于新到达 DAG 优先级的 DAG 中的任务;

保留所有优先级大于新 DAG 的 DAG 中任务的调度方案,并将这些 DAG 从 $U_{scheduling}$ 中删除;

While ($U_{scheduling}\neq\varnothing$) **do**

计算和修改所有时隙链表 S_{M_i};

$a_{peer}\leftarrow$在 $U_{scheduling}$ 中选取优先级与新 DAG 相同的 DAG 和新达到的 DAG;

If (a_{peer} 中的 $DAG\notin CO\text{-}DAG$)

a_{peer} 中多个同级 DAG 间的调度关系采用 E-Fairness 算法,并采用 HEFT 的 Backfill 算法将这些任务匹配到 S_{M_i},同时将任务依次放入 Q_{W-M_i}

Else

a_{peer} 中多个同级 DAG 间的调度关系采用 E-Fairness 算法,并采用 LE 方法的 Backfill 算法将这些任务匹配到 S_{M_i},同时将任务依次放入 Q_{W-M_i};

$U_{scheduling}=U_{scheduling}-a_{peer}$

End while

End while

4 实验结果与分析

本节通过大量的实验数据,着重从资源利用率 *UR*、DAG 的 *Makespan* 和不公平程度 *Unfairness* 这 3 项指标将 MMHS 策略和相关的算法(E-Fairness,Backfill 和 Round-Robin)进行性能比较.

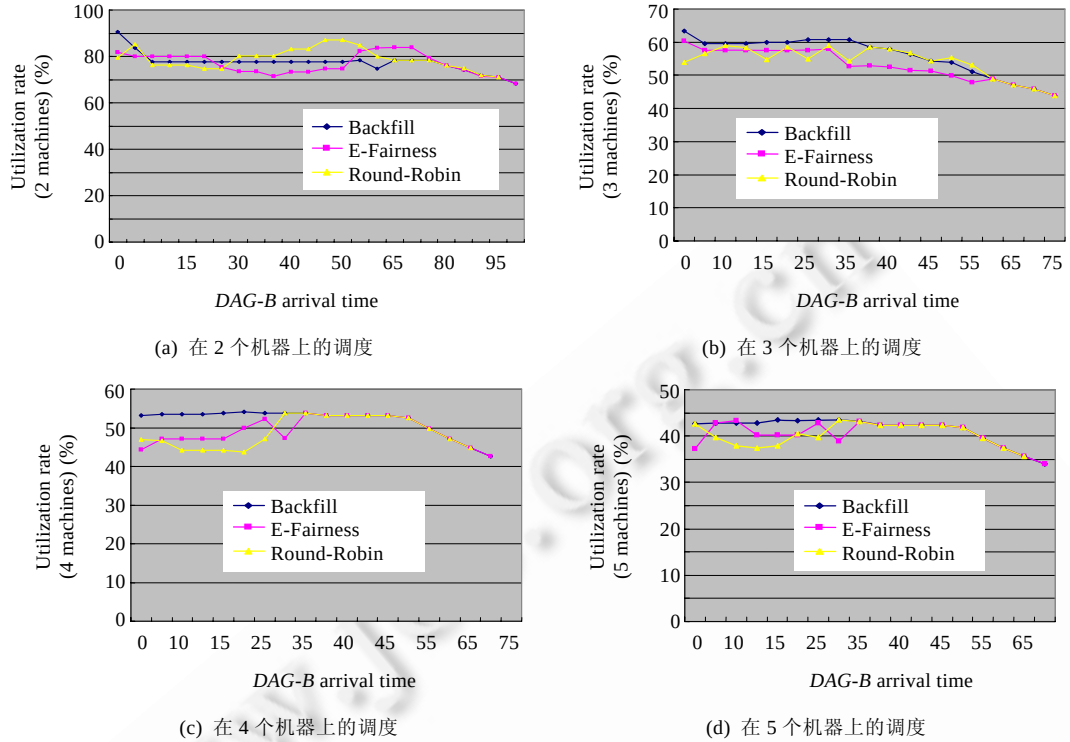
其中,第 4.1 节是针对上文第 2 节中给出的两个 DAG 实例调度实验的统计和分析.由于针对两个 DAG 的调度,MMHS 调度策略只需要在 E-Fairness 和 Backfill 之间进行选择,所以只需对 E-Fairness,Backfill 和 Round-Robin 这 3 种算法在不同条件下作性能比较即可.第 4.2 节给出了针对随机产生的两种及两种以上 DAG 的调度实验的实验方法、结果分析和性能比较.对于调度两个以上的 DAG,MMHS 将会根据新达到 DAG 的优先级在 E-Fairness 和 Backfill 两种算法之间进行切换.而其他 3 种算法:E-Fairness,Backfill 和 Round-Robin 将不考虑 DAG 的优先级,分别单独使用,直至将所有 DAG 调度完毕.

4.1 相关的两个DAG的调度实验

针对图 2 所示的两个 DAG 实例,我们分别采用 E-Fairness,Backfill 和 Round-Robin 这 3 种算法在不同机器数目上(2~6 machines)进行这两个 DAG 的调度.我们在 0 时刻使用 HEFT 调度 *DAG-A*;而 *DAG-B* 到达时间从 0 时刻开始,以平均 5 个时间单位递进增加 *DAG-B* 的到达时间来模拟 *DAG-B* 的动态到达.在这部分实验的参数中,表 1 中两个 DAG 中的任务在各机器上的执行时间仅用于机器数目为 3 的情况;而对其他几种数目机器的实验情况,虽然各任务在每个机器上的执行时间有所不同,但各任务的平均执行时间与表 1 中的平均值相同.

图 5 显示了在 *DAG-B* 不同到达时间上,分别在 2 个~5 个机器上,3 种调度算法的 *UR* 性能指标情况.从机器的不同数目情况来看,3 种算法的利用率都随着机器数目的增加而下降;但随着机器数目的增加,利用率下降的速度变慢.从另一个角度看,这样的结果也表明了文献[21]所提出的结论:机器数目的增加并不一定会减小 DAG 的调度时间长度.从 *B* 的不同到达时间来看,随着 *B* 到达时间的增加,Backfill 算法的 *UR* 变化幅度都小于其他两

种算法.另外,从图 5 中可以看出,尽管图 5(a)中的情况要复杂一些,但总体来看,Backfill 算法的利用率要好于 E-Fairness 和 Round-Robin.



(a) 在 2 个机器上的调度 (b) 在 3 个机器上的调度

(c) 在 4 个机器上的调度 (d) 在 5 个机器上的调度

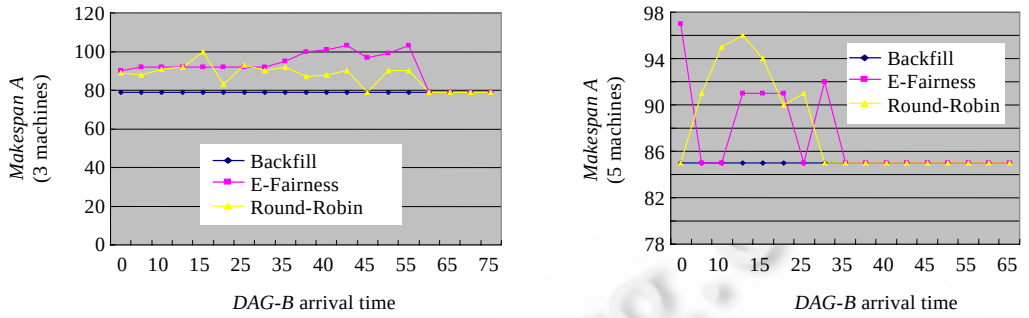
Fig.5 *UR*, at different *DAG-B* arrival time, scheduled on 2~5 machines, respectively

图 5 在 *DAG-B* 不同到达时间上,3 种调度算法的 *UR*(分别在 2~5 个机器上的情况)

进一步来看,如果假设 *A* 属于 RT-DAG 类型并且 *B* 属于 TO-DAG 类型,即 *B* 的优先级小于 *A* 的优先级,根据我们的混合调度策略 MMHS:if (P_{new}<max{$P_{scheduling}$}),保留那些优先级大于新到达 DAG 的所有 DAG 的任务分配方案.那么,当 *B* 到达时,MMHS 将保留 *A* 中的所有任务在 0 时刻的分配方案,而将 *B* 匹配到各个机器的时隙上.我们比较了 3 种算法的 *Makespan A*,如图 6 所示,采用 Backfill 算法,*DAG-A* 的 *Makespan A* 能够避免受到 *B* 的影响,较好地保证了 *DAG-A* 用户对执行期限的较为严格的限制.在下文的图 9 中显示了在 *DAG-B* 不同到达时间上,分别在 2 个~6 个机器上 3 种调度算法的相关性能指标情况.可以看出,在公平性指标上,Backfill 算法比其他两种算法相对要差.所以如果 *A* 和 *B* 的优先级相同,那么公平性比效率更为重要.因此,当 *B* 到达时,MMHS 会将 *A* 中未执行的任务和 *B* 的所有任务一起利用 E-Fairness 算法进行调度,以达到较好的公平性.

值得注意是,所有两个 DAG 的调度实验统计结果都会出现“拖尾”规律.从图 5~图 8 可以看出:如果 *B* 的到达时间超过某个时间点以后,尽管在这个时间点上(以下称这个时间点为 *TET*(拖尾时间 trail ending time)), *A* 中的有些任务还没有开始调度执行(可结合表 1、图 3 和图 4 来看),但相关性能指标都会不同程度地呈现出“拖尾”现象.这就意味着:对结构确定的两个 DAG,如果当第 1 个 DAG 被调度后,第 2 个 DAG 在某个时间(*TET*)到后才到达,尽管第 1 个 DAG 还有部分任务未被调度执行,并且利用不同的调度算法对新到达 DAG 的所有任务和旧 DAG 的剩余任务调度顺序也都不一样,但最终这些算法的各项性能指标会表现为一致.随着机器数目的增加,*TET* 会提前,并且当机器数目到达一定量时,*TET* 会停止提前而稳定在某个值上.如图 5(a)~图 5(d)所示,*TET* 的值分别是 80,60,35 和 35.这一规律可以从另外的角度作定性分析:如果 *B* 的到达时间越晚,尤其是在 *A* 中多数任务已被调度执行后 *B* 才到达,那么两个 DAG 之间的相互影响程度就越小.一种极端情况是,当 *A* 已经执行完

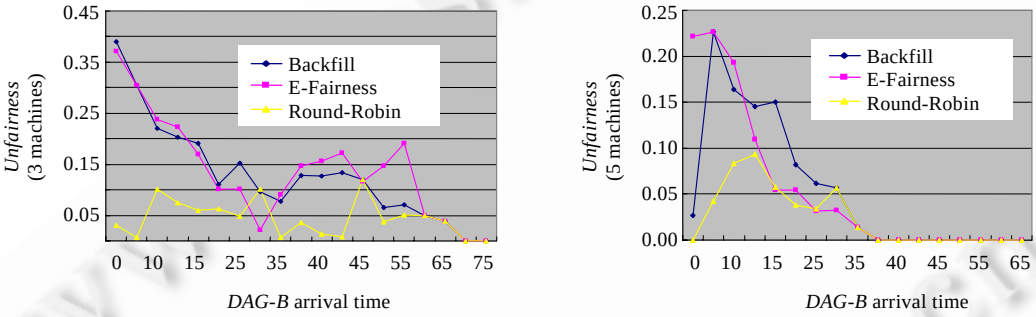
所有任务后 *B* 才到达,所有不同的算法就等同于依次用 HEFT 单独调度两个不同的 DAG,这两个 DAG 会互不影响.随着机器数目的增加,*TET* 会有所提前,这也说明,随着机器数目的增加,多个 DAG 之间的影响程度也会随之降低.我们也做了大量的其他两个 DAG 的调度实验,均出现了同样的“拖尾”规律.限于篇幅,这里不再罗列所有实验结果.



(a) 在 3 个机器上的调度 (b) 在 5 个机器上的调度

Fig.6 *Makespan A*, at different *DAG-B* arrival time, scheduled on 3 and 5 machines, respectively

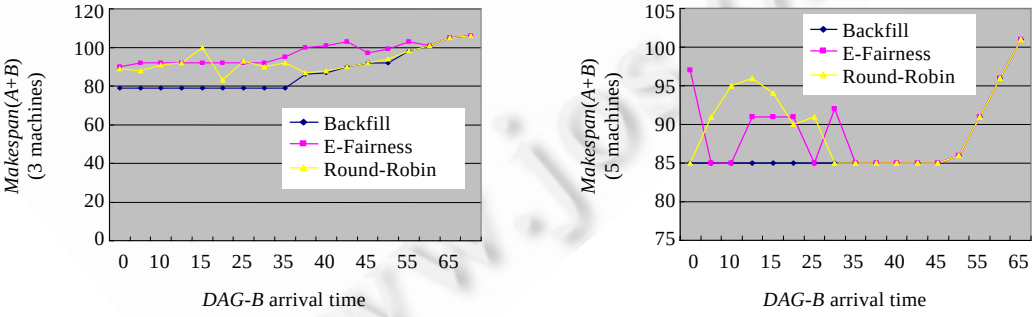
图 6 在 *DAG-B* 不同到达时间上,3 种调度算法的 *Makespan A*(分别在 3 个和 5 个机器上的情况)



(a) 在 3 个机器上的调度 (b) 在 5 个机器上的调度

Fig.7 *Unfairness*, at different *DAG-B* arrival time, scheduled on 3 and 5 machines, respectively

图 7 在 *DAG-B* 不同到达时间上,3 种调度算法的 *Unfairness*(分别在 3 个和 5 个机器上的情况)

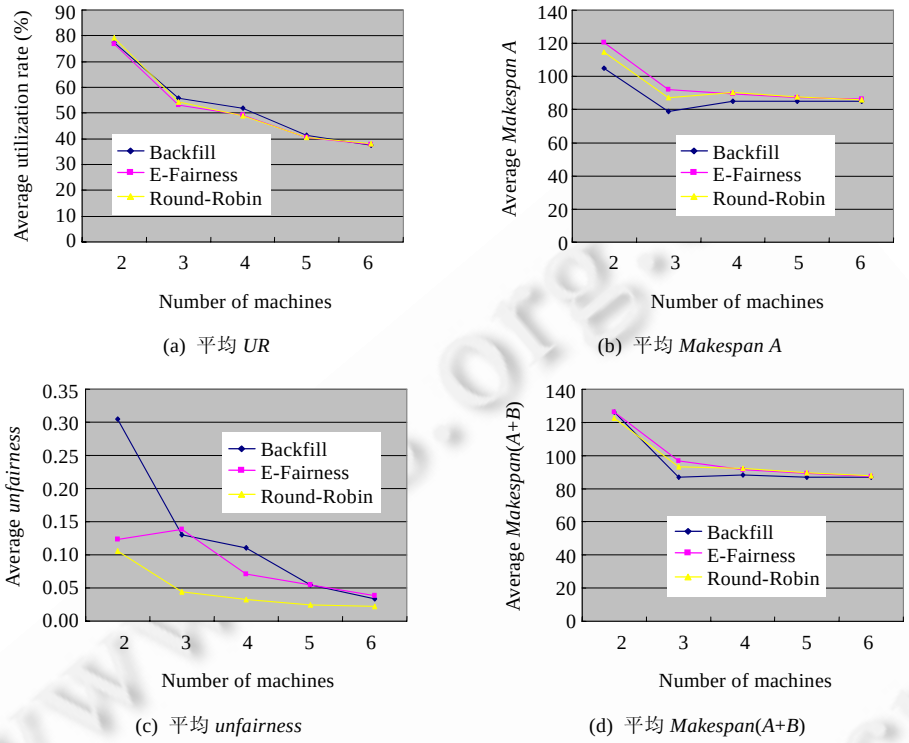


(a) 在 3 个机器上的调度 (b) 在 5 个机器上的调度

Fig.8 *Makespan*(*A*+*B*), at different *DAG-B* arrival time (scheduled on 3 and 5 machines, respectively)

图 8 在 *DAG-B* 不同到达时间上,3 种调度算法的 *Makespan*(*A*+*B*)(分别在 3 个和 5 个机器上的情况)

这一“拖尾”规律在一些应用中是有价值的.如在某些特定的多 DAG 系统中,当执行一个 DAG 时,不允许受到任何其他 DAG 的影响时,可以采用先来先服务的简单方法,即调度完先来 DAG 的所有任务后再调度后到达的 DAG,那么,该系统的 *TET* 可以基于历史记录,通过一定的学习方法获得.一旦 *TET* 被确定,那么调度系统可以不必等待先到达的 DAG 中所有任务执行完毕才开始调度后到达的 DAG,这样可以大大提高这类多 DAG 系统的调度效率.



(a) 平均 *UR* (b) 平均 *Makespan A*

(c) 平均 *unfairness* (d) 平均 *Makespan*(*A*+*B*)

Fig.9 Average values for the related metrics at different *DAG-B* arrival time, scheduled on 2~6 machines

图 9 在 *DAG-B* 不同到达时间上,3 种调度算法的各性能指标平均值(分别在 2 个~6 个机器上的情况)

4.2 多个随机DAG的调度实验结果分析

本节是针对多个随机产生的不同时间到达的多优先级 DAG 调度的相关实验参数和实验结果.

相关的实验参数如下:

(1) 实验中随机的 DAG 是根据表 4 中构成 DAG 各个参数的取值区间按照均匀分布产生.

(2) 3 种 DAG 类型(RT-DAG,TO-DAG,CO-DAG)的比例和 DAG 平均到达时间间隔见表 5.

(3) 每个任务的估计计算时间 w_{ij} 和 $\overline{w}_i$ 的确定方法与文献[2]的方法一样.

根据以上参数,我们模拟了在不同的 DAG 到达时间间隔,在机器数目分别为 2~6 上,DAG 并行调度个数分别为 2,4,6,8 和 10 的调度实验.针对每种情况做 30 次,并且每次的 DAG 根据相关参数随机产生.然后,统计了当 DAG 个数相同时,所有情况中各个性能指标的平均值,实验结果如图 10 所示.

Table 4 Range of parameters of random DAGs

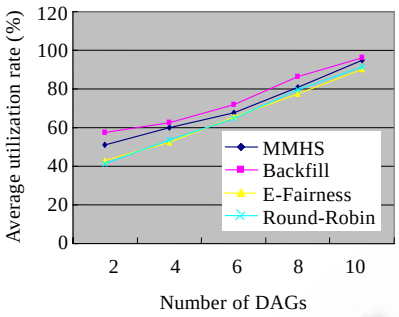
表 4 实验的随机 DAG 参数的取值范围

顶点(任务)个数	5~30
顶点出度	2~5
传输时间和任务计算时间比率:*CCR*	0.1~2
边长$c_{(i,j)}$	3~50

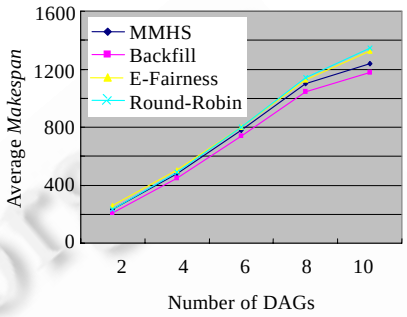
Table 5 Ratio of 3 types of DAG and interarrival time of DAG

表 5 3 种类型 DAG 的比例和 DAG 到达时间间隔

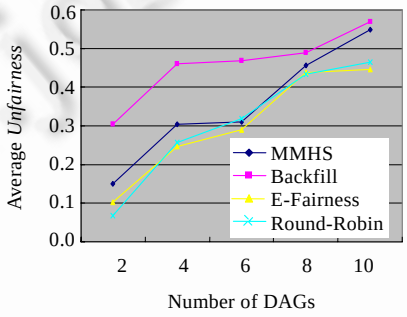
并行调度的DAG个数	RT:TO:CO (每种DAG类型的个数)	平均到达时间间隔 (实验的间隔时间以此为均值按Poisson分布产生)
2	1:1:0	10, 30, 50, 70, 90
4	1:2:1	10, 30, 50, 70, 90
6	1:2:3	10, 30, 50, 70, 90
8	5:2:1	10, 30, 50, 70, 90
10	2:5:3	10, 30, 50, 70, 90



(a) 平均利用率



(b) 平均 *Makespan*



(c) 平均 *UnFairness*

Fig.10 Average values for the related metrics
(different arrival interval under different number of DAGs on 2~6 machines)

图 10 相关指标的平均值(分别在 2~6 个机器上的不同到达时间不同 DAG 个数)

由图 10 可以看出,MMHS 的平均利用率 *UR* 高于 E-Fairness 和 Round-Robin,但是低于 Backfill;而平均不公平性低于 Backfill.也可以看到,MMHS 的相关指标总是落在 Backfill 和 E-Fairness(或 Round-Robin)之间,并且当某种类型的 DAG 在 DAG 总数中所占比例较大时,例如当 DAG 的个数是 8 时,RT:TO:CO=5:2:1,RT 类型的 DAG 占多数,那么 MMHS 的性能便接近于 E-Fairness.这是因为,MMHS 在同级 DAG 间总是采用 E-Fairness 算法,而在不同级 DAG 间的调度关系选用 Backfill 算法,所以同级的 DAG 比例越大,那么 MMHS 会更多地选择 E-Fairness 算法,因此调度性能也更接近于 E-Fairness.另外,实验结果也表明,在 MMHS 混合调度策略下,RT 类型的 DAG 平均执行时间也比分别单独使用 Backfill 和 E-Fairness 算法要低.

5 结 论

针对分布式环境下多个不同时间到达的 DAG 工作流的调度,本文做了以下几方面的工作:首先,为了提高机器资源的利用率和满足不同用户对 DAG 工作流的不同 QoS 需求,提出了多 DAG 工作流的调度优先级这一

概念,并将其划分为若干等级.为了能够适应这种多优先级多 DAG 工作流的调度,进一步提出了一种新的调度模型.由于多优先级 DAG 的调度既涉及同级 DAG 的调度关系,也涉及不同级 DAG 之间的调度关系,因此无论是现有的 Fairness(公平)算法,还是其他算法,都不能单独用于多优先级多 DAG 的调度.现有的 Fairness 虽然适合于相同优先级 DAG 之间调度关系的确定,但未能解决多个 DAG 在不同时间上提交被调度的问题,因此,本文提出了 Fairness 算法的改进方法和步骤.另外,本文为了提高资源的利用率和减小优先级较高的 DAG 受到低优先级 DAG 的影响,又提出了一种适合于不同优先级的多个 DAG 之间调度的 Backfill 算法.最后,在改进的 Fairness 算法和 Backfill 算法的基础上,提出了一种混合调度策略,较好地解决了异构分布式环境下多优先级多 DAG 工作流在不同时间到达时的调度问题.最后的实验结果表明:与单独使用公平调度算法或 Backfill 算法相比,这种混合调策略既较好地保证了相同优先级 DAG 之间调度的公平性,也提高了资源利用率.另外,对所进行的实验进行统计后,发现了关于两个 DAG 调度所特有的“拖尾”规律和现象,有着较好的研究和应用价值.

References:

[1] Yuan YC, Li XP, Wang Q, Wang KJ. Time optimization heuristics for scheduling budget-constrained grid workflows. Journal of Computer Research and Development, 2009,46(2):194−201 (in Chinese with English abstract). http://crad.ict.ac.cn:81/CRAD/ePublish/Search/Search.asp

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

[3] Tao Y, Gerasoulis A. DSC: Scheduling parallel tasks on an unbounded number of processors. IEEE Trans. on Parallel and Distributed Systems, 1994,5(9):951−967. [doi: 10.1109/71.308533]

[4] Kwok Y-K, Ahmad I. On multiprocessor task scheduling using efficient state space search approaches. Journal of Parallel and Distributed Computing, 2005,65(12):1515−1532. [doi: 10.1016/j.jpdc.2005.05.028]

[5] Kwok Y-K, Ahmad I, Jun G. FAST: A low-complexity algorithm for efficient echeduling of DAGs on parallel processors. In: Proc. of the ’96 Int’l Parallel Processing. Piscataway: IEEE, 1996. 150−157. [doi: 10.1109/ICPP.1996.537394]

[6] Yuan D, Yang Y, Liu X, Chen J. A data placement strategy in scientific cloud workflows. Future Generation Computer Systems, 2010,26(8):1200−1214. [doi: 10.1016/j.future.2010.02.004]

[7] Byun E, Choi S, Baik M, Gil J, Park C, Hwang C. MJSA: Markov job scheduler based on availability in desktop grid computing environment. Future Generation Computer Systems, 2007,23(4):616−622. [doi: 10.1016/j.future.2006.09.004]

[8] Tian GZ, Yu J, He JS. Towards critical region reliability support for grid workflows. Journal of Parallel and Distributed Computing, 2009,69(12):989−995. [doi: 10.1016/j.jpdc.2009.04.010]

[9] Zhao HN, Sakellariou R. Scheduling multiple DAGs onto heterogeneous systems. In: Proc. of the 2006 20th Int’l Parallel and Distributed Processing Symp. (IPDPS 2006). Piscataway: IEEE, 2006. 14. [doi: 10.1109/IPDPS.2006.1639387]

[10] Kertész A, Sipos G, Kacsuk P. Brokering multi-grid workflows in the P-GRADE portal. In: Proc. of the Parallel Processing (Euro-Par 2006), Vol.4375. Berlin: Springer-Verlag, 2007. 138−149. [doi: 10.1007/978-3-540-72337-0_14]

[11] Hönig U, Schiffmann W. A meta-algorithm for scheduling multiple DAGs in homogeneous system environments. In: Proc. of the 2006 18th Int’l Parallel and Distributed Computing and Systems. Piscataway: IEEE, 2006. 147−152.

[12] Bennett JCR, Zhang H. WF^2Q: Worst-Case fair weighted fair queueing. In: Proc. of the 15th Annual Joint Conf. of the IEEE Computer Societies (INFOCOM’96), Networking the Next Generation. Piscataway: IEEE, 1996. 120−128. [doi: 10.1109/INFCOM.1996.497885]

[13] Yu ZF, Shi WS. A planner-guided scheduling strategy for multiple workflow applications. In: Proc. of the Int’l Parallel Processing-Workshops (ICPPW 2008). Piscataway: IEEE, 2008. 1−8. [doi: 10.1109/ICPP-W.2008.10]

[14] Fahringer T, Prodan R. ASKALON: A grid application development and computing environment. In: Proc. of the 6th IEEE/ACM Int;l Workshop on Grid Computing. Piscataway: IEEE, 2005. 122−131. [doi: 10.1109/GRID.2005.1542733]

[15] Dagman. Condor 7.4.2 released, 2010. http://www.cs.wisc.edu/condor/dagman/

[16] Yu J, Buyya R. Budget constrained scheduling of workflow applications on utility grids using genetic algorithms. In: Proc. of the Workshop on Workflows in Support of Large-Scale Science (WORKS 2006). Piscataway: IEEE, 2006. 1−10. [doi: 10.1109/WORKS.2006.5282330]

[17] Buyya R, Abramson D, Venugopa S. The grid economy. Proc. of the IEEE, 2005,93(3):698−714. [doi: 10.1109/JPROC.2004.842784]

[18] Stuer G, Vanmechelena K, Broeckhovea J. A commodity market algorithm for pricing substitutable grid resources. Future Generation Computer Systems, 2007,23(5):688−701. [doi: 10.1016/j.future.2006.11.004]

[19] Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 2009,25(6):599−616. [doi: 10.1016/j.future.2008.12.001]

[20] Stavrinides GL, Karatza HD. Scheduling multiple task graphs with end-to-end deadlines in distributed real-time systems utilizing imprecise computations. Journal of Systems and Software, 2010,83(6):1004−1014. [doi: 10.1016/j.jss.2009.12.025]

[21] Kwok Y-K, Dynamic AI. Critical-Path scheduling: An effective technique for allocating task graphs to multiprocessors. IEEE Trans. on Parallel and Distributed Systems, 1996,7(5):506−521. [doi: 10.1109/71.503776]

附中文参考文献:

[1] 苑迎春,李小平,王茜,王克俭.成本约束的网格工作流时间优化方法.计算机研究与发展,2009,46(2):194−201.

田国忠(1971－),男,江苏常州人,博士生,副教授,主要研究领域为并行与分布计算,高性能集群计算.

徐竹胜(1986－),男,硕士生,主要研究领域为并行计算,图形图像处理.

肖创柏(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机网络安全,信号处理,云计算,数据挖掘.

肖霞(1989－),女,硕士生,主要研究领域为并行与分布计算.