云计算中一种多DAG工作流可抢占式调度策略

孙 月[1,2] 于 炯[1,3] 朱建波[1]
(新疆大学信息科学与工程学院 乌鲁木齐 830046)[1] (68302部队 渭南 714000)[2]
(新疆大学软件学院 乌鲁木齐 830008)[3]

摘 要 为解决多用户工作流调度过程中的公平性问题,提高资源利用率,满足不同用户DAG工作流的不同QoS需求,提出了抢占式多DAG工作流动态调度模型。该算法将DAG工作流按照QoS需求进行优先级划分,采用高优先级作业优先占有资源的原则调度作业。相同优先级DAG工作流的任务依据带有启发性信息的slowdown进行资源抢占,进一步提高了作业调度的公平性;对于不同优先级的作业调度,提出了基于阈值的回填算法,该算法在保证作业调度公平的同时提高了资源利用率。

关键词 多DAG调度,优先级,抢占式,公平性,回填
中图法分类号 TP301.6 **文献标识码** A

Preemptive Scheduling for Multiple DAGs in Cloud Computing

SUN Yue[1,2] YU Jiong[1,3] ZHU Jian-bo[1]
(College of Information Science and Technology, Xinjiang University, Urumqi 830046, China)[1]
(68302 Troops of PLA, Weinan 714000, China)[2]
(College of Software, Xinjiang University, Urumqi 830008, China)[3]

Abstract There are different QoS demands on DAG workflow for different users. Three layers multiple DAG dynamic scheduling model was proposed to solve the fairness and the utilization rate problems. In the algorithm, priority division is in the light of the different QoS demand of DAG, and the high priority DAG gives preference to the resources. The same priority DAGs can preempt the resources according to the slowdown with the heuristic information, improving the fairness of the workflow scheduling. In the circumstances of high priority DAGs giving preference to resources, the resource utilization rate must be considered. Modified Backfill algorithm was introduced based on threshold, improving the resources utilization rate with the guarantee of high priority job in preference to resources.

Keywords Multiple DAGs scheduling, Priority, Preemptive, Fairness, Backfill

1 引言

工作流[1,2]作为计算机支持的协调工作的一部分,有效反映了流程中各项任务的逻辑约束关系,它被广泛应用于规划、企业工程、IT应用体系等领域,其描述通常用一个有向无环图DAG(Directed Acyclic Graph)来表示,而工作流的管理其关键技术之一就是调度策略。对于单个DAG的工作流调度策略[3-4]的研究取得了很大进展,其中一类是静态调度策略,它假设作业的调度不会失败,计算资源的可用性及计算能力不随时间改变,调度过程中每个任务结点的计算代价和数据传输代价是估计信息,因此这类策略的调度过程不具有鲁棒性[5],典型算法有HEFT[6]、Sufferage等;另一类是动态调度策略,它允许作业或资源在调度过程中动态变化,如DLS[7]、GridFlow[8]、GrADS[9]、SR[5]。

单DAG调度算法大部分是静态算法,不能适应多DAG调度的动态特性,也不能直接应用于多DAG调度。Honig和Schiffmann[10]为了解决单DAG静态调度算法应用于多DAG效率低的问题,提出了将多个DAG合并再调度的方法,但这种方法存在各DAG调度不公平的问题。Zhao和Sakellariou[11]考虑了多个DAG调度的公平性问题,提出了基于度量因子Slowdown的公平性算法。Yu Zhi-feng和Shi Wei-song[12]提出了针对不同时刻多DAG的动态调度模型,但是新的DAG到达后就会出现权值小的DAG一直占有计算资源的不公平性问题。DAG工作流的QoS衡量参数[13-14]主要有:时间、费用、安全性、可靠性和优先级等,上述这些应用于多工作流调度的策略并没有考虑用户的QoS需求。在云计算环境中,为公平使用资源,提高资源的利用率,多DAG工作流的调度[15,16]中必须考虑每个用户的不同需求。

文中提出了基于优先级的抢占式多DAG工作流动态调度策略,即实现不同优先级的作业抢占资源,同优先级的作业

到稿日期:2013-04-05 返修日期:2013-08-20 本文受新疆维吾尔自治区自然科学基金项目(2011211A011),国家自然科学基金项目(61262088,61063042)资助。

孙 月(1985-),男,硕士生,主要研究方向为云计算、网格与分布式计算,E-mail:sunyue@xju.edu.cn;于 炯(1964-),男,博士,教授,主要研究方向为网络安全、网格与分布式计算;朱建波(1987-),男,硕士生,主要研究方向为云计算、网格与分布式计算。

依据作业调度进展也可以抢占资源；为提高资源利用率，不同优先级作业调度使用基于阈值的回填策略。

2 基于优先级的抢占式多 DAG 工作流动态调度

2.1 调度模型

为了满足不同用户的 QoS 需求，实现基于优先级的多用户作业调度，将优先级队列分为高优先级可抢占队列、高优先级非可抢占队列和基本优先级队列 3 类。工作流子任务间存在约束关系，当一个任务执行完毕，其直接后继任务才能进入任务池准备被调度。调度机将高优先级 DAG 的任务优先调度到任务队列，低优先级 DAG 的任务只能使用回填策略调度。当资源空闲时，任务队列中的第一个任务将会被调度。抢占式多 DAG 的动态调度模型如图 1 所示，由 4 部分组成：优先级队列、任务池、任务队列、调度器。

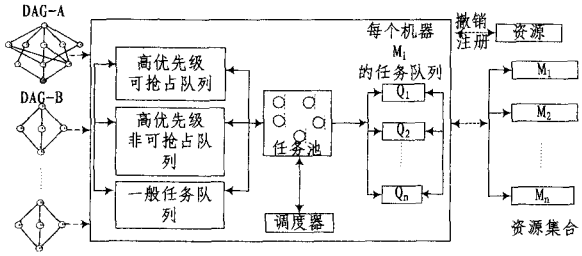


图 1 抢占式多 DAG 工作流动态调度模型

此模型中，调度器负责 DAG 工作流优先级的识别，优先级队列、任务队列和计算资源的管理，任务调度。其调度时机一个是当有任务执行完毕释放资源时，该任务的直接后继进入任务池，调度器根据当前时间更新每个 DAG 的 Slowdown，然后将任务池中的任务依据调度策略调度到相应任务队列，等待资源可用；另一个是当有用户提交新的作业时，其 Entry Node 结点任务直接进入任务池，调度器根据当前时间进行作业调度。

在优先级队列中，高优先级可抢占队列中的 DAG 具有最高优先级，优先使用资源，可以抢占后两类作业的任务资源；高优先级非可抢占队列中的 DAG 具有先得到响应的权利，但不能抢占低优先级作业的任务资源；基本优先级队列中的 DAG 具有最低的优先级。资源抢占不仅在不同优先级 DAG 作业之间发生，而且同优先级的 DAG 作业之间根据延迟因子 Slowdown 也进行资源抢占，不能抢占正在使用的资源。

为了实现任务的迁移和系统负载的平衡，提高对资源计算能力的动态变化和可用性的容忍度，模型中引入任务队列。新的资源注册时，调度器为其新建、维护一个任务队列；资源撤销时，对应任务队列中的任务全部重新放入任务池，包括正在计算但没有完成的任务。

2.2 调度策略

一般一个 DAG 工作流可以表示为 $G=(V,E)$，其中 V 是任务结点集合，E 是任务结点间有向边的集合。每个有向边 $(i,j)\in E$ 表示任务结点 n_i 和 n_j 之间的先后执行关系，即 n_j 必须在 n_i 执行完毕以后才能开始执行。***Data*** 是一个 $v\times v$ 的矩阵，矩阵中的元素 $data_{i,j}$ 表示结点 n_i 向 n_j 需要传递的数据量。如果任务结点 n_i 和 n_j 分配在同一个资源上，则认为 $data_{i,j}=0$。Q 为全拓扑方式连接在一起的 q 个处理器组成的集合。设 $\boldsymbol{W}$ 是 $v\times q$ 的计算时间矩阵，其中的元素 w_{ij} 表示结点 n_i 在机器资源 q_j 上的期望运行时间。

多 DAG 调度过程中的一个 DAG 的 Makespan 和它单独使用资源的 Makespan 分别表示为 $M_{multi}(a)$ 和 $M_{own}(a)$。根据文献[11]的定义，Unfairness 是用来衡量一个多 DAG 调度算法 S 调度的不公平程度的重要指标，表示为：

$$Unfairness(S)=\sum_{\forall a\in A}|Slowdown(a)-AvgSlowdown| \tag{1}$$

$$AvgSlowdown=\frac{1}{|A|}\sum_{\forall a\in A}Slowdown(a)$$

$$Slowdown(a)=\frac{M_{own}(a)}{M_{multi}(a)} \tag{2}$$

式中，A 为多个 DAG 的集合，$AvgSlowdown$ 是所有 DAG 的 Slowdown 平均值。$|A|$ 表示集合 A 的基。

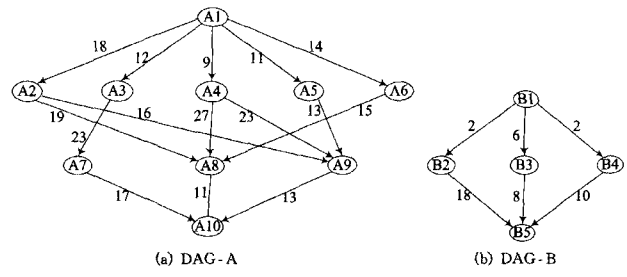
2.2.1 同优先级 DAG 的任务调度

文献[11]中的 Slowdown 是基于多 DAG 调度的当前时间来衡量各个作业的执行进度的，它适用于静态调度策略，而云计算中资源的可用性及计算能力都是动态变化的。本文中任务被调度后进入相应资源的任务队列中，虽然此任务没有占有计算资源，但是此时的作业进度已经比没有被调度的情况要快，而基于当前调度时间的 Slowdown 并不能真实反映此时各个作业的调度的调度进展。虽然任务队列中的任务还没有得到资源，但是调度系统已经试图对现有调度顺序分配资源。为了提高调度的公平性，将每个资源对应任务队列中的所有任务作为搜索空间，引入启发性信息因子[17] DelayTime(a)，更新 slowdown。通过任务队列中未执行的任务，让系统根据调度策略来决定当前作业的后继任务是否有必要进行资源的抢占，以调整整个作业的调度，保证使用资源的公平性(Preemptive Fairness)，将这种保证同优先级作业调度公平的策略简称为 P-Fairness 策略。

$$Slowdown(a)=1-DelayTime(a)/ct$$
$$DelayTime(a)=M_{multi}-M_{own} \tag{3}$$

式中，ct 表示当前调度时间，$DelayTime(a)$ 表示工作流 a 和其它 DAG 工作流共享计算资源时的调度时间与 a 单独使用计算资源所需调度时间相比的延迟量。事实上，当 $ct=M_{multi}$，很容易证明式(2)、式(3)是等价的。当用 $DelayTime(t)$ 来反映 a 的调度进程时(t 是 a 的任务)，$DelayTime(t)<0$，这就表示与 a 单独使用资源相比，任务 t 的调度是提前的；同理，$DelayTime(t)>0$ 时，表示与 a 单独使用资源相比，虽然 t 还没有被执行，但是任务 t 的调度已经延迟。

下面通过图 2 中两个工作流实例 DAG-A 和 DAG-B 来说明 P-Fairness 策略。假设两个 DAG 中每个任务在每个机器资源上的执行时间如表 1 所列，那么按照 $rank_u$[6] 定义可以得到每个任务的向上权值，如表 2 所列。



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

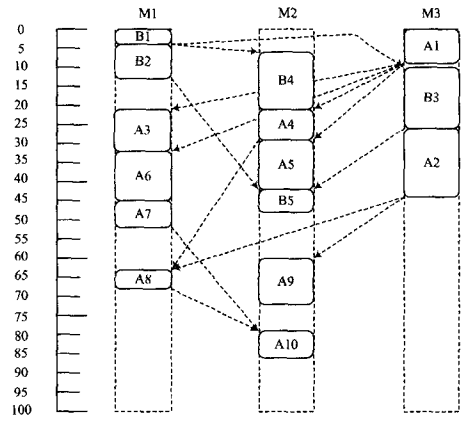
图 2 两个 DAG 工作流实例图

表 1　DAG-A 和 DAG-B 中各任务在机器上的执行时间

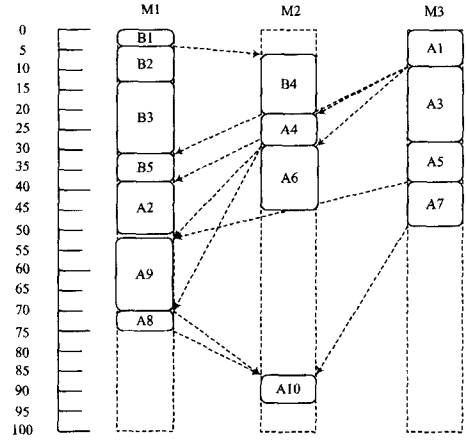
DAG	DAG-A											DAG-B				
n_i	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	B_1	B_2	B_3	B_4	B_5
M_1	14	13	11	13	12	13	7	5	18	21	4	9	18	21	7
M_2	16	19	13	8	13	16	15	11	12	7	5	10	17	15	6
M_3	9	18	19	17	10	9	11	14	20	16	6	12	16	19	5

表 2　两个 DAG 的任务向上权值表

DAG	DAG-A										DAG-B				
ni	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5
Ranku	108	77	80	80	69	63.33	42.67	35.67	44.33	14.67	42	34.33	31	34.33	6



(a) AvgSlowdown=0.840 Unfairness=0.180



(b) AvgSlowdown=0.904 Unfairness=0.087

图 3　DAG-A 和 DAG-B 分别用 Fairness、P-Fairness 算法调度结果

图 3(a)是基于 Zhao 的 Fairness 算法对 DAG-A 和 DAG-B 的调度结果，在 $t1=26$ 时刻，B5 已经等待调度执行，但 B5 一直处于等待状态，得不到资源，显然这对 DAG-B 来说是不公平的。图 3(b)是基于文中提出的 P-Fairness 策略的调度结果。B1 完成时，后继任务 B2、B4、B3 进入任务池，调度器将其分别调度到 M1、M2、M3；但 A1 完成后，其直接后继任务(A3、A4、A2、A5、A6)进入工作池调度($t1=9$)，$SD_{DAG\text{-}A}=SD_{DAG\text{-}B}=1$(DAG a 的 SlowDown 简写为 SD_a)，但 DAG-A 的 Makespan 大于 DAG-B 的 Makespan，所以 DAG-A 的任务优先得到资源，即 A3、A4、A2、A5、A6 优先得到调度，抢占任务 B3 的资源。基于 B2 的完成时间 $t2=12$，由式(2)得 $SD_{DAG\text{-}A}=SD_{DAG\text{-}B}=1$，但 DAG-B 已经比单独占有资源时变慢，没有考虑到 B4、B3 的调度情况。由式(3)得：$SD_{DAG\text{-}A}=1-(12^1-12^2)/t2=1$，$12^2$ 对应 DAG-A 单独使用资源时在 t_2 时刻的作业调度状态：A3 完成 15.8%，12^1 对应 DAG-A 与其它作业共享资源时完成对应任务比例的时间。$SD_{DAG\text{-}B}=1-(40-12)/t2=-1.3$，12 对应 DAG-B 单独使用资源时在 $t2$ 时刻的调度状态：B3 完成12.5%，40 对应 DAG-B 共享资源时 B3 完成 12.5%的期望时间。因此，基于 $t2=12$ 时刻的调度，DAG-B 具有先占有资源的权利，对于队列中未执行的任务 B3 重新调度，抢占 DAG-A 中 A2 的资源，A2 被放入任务池，等待重新调度。

2.2.2　不同优先级 DAG 的任务调度

资源的任务队列 Q_n 中的任务能被执行，必须满足两个条件：一是此任务在队列的头部，二是此任务需要的父结点的数据传送完毕。当资源出现时隙 *slot* 时，只有执行时间小于时隙的任务才能进行利用。由于任务的约束关系和 *slot* 本身较小，时隙得到利用的几率也较小。为充分利用 *slot*，提高资源利用率，在不同优先级的作业调度时，使用基于阈值的 Backfill[18] 策略对 *slot* 进行调整，其取值为[0～C]，简称 T-Backfill(Backfill Strategy Based on Threshold)策略。

阈值的定义为：

$$C=\text{size}(slot)/n \tag{4}$$

其中，size(*slot*)表示 *slot* 的大小，n 为计算资源的数目。调度器将任务池中高优先级作业的任务调度到任务队列后，对任务池中低优先级作业的任务使用 T-Backfill 策略调度。使用回填算法的任务可以通过阻塞整个资源队列来调整 *slot* 的大小，这样在保证公平使用资源的同时，确保提高资源的利用率。

2.3　算法描述

对本文提出的基于 P-Fairness 和 T-Backfill 策略的多 DAG 抢占式调度算法(Multiple DAGs Preemptive Scheduling，简称 MPS)描述如下。

初始化：

1. S_{org}：每个 DAG 独自占有资源用 HEFT 算法时的 Makespan；
2. R：计算资源的集合；
3. JP：任务池中任务的集合，当一个任务执行完毕其直接后继任务才能添加进 JP，Entry Node[11] 直接添加进 JP；
4. U：所有 DAG 集合；

抢占式调度策略：

1. While U≠∅
2. 基于当前时间更新每个 DAG 的 SlowDown；
3. Sort U：按照任务的 DAG 优先级由高到低排序，优先级相等时按照 DAG 的 SlowDown 由低到高排序，SlowDown 相等时按照剩余任务的 Makespan 由高到底排序；
4. If b∈U 并且 b 在 U 中顺序发生变化
5. 撤销资源队列中优先级低于 b 的所有作业的任务，重新放入任务池；
6. while JP≠∅
7. a←U 中的第一个未完成的 DAG；

```
8.    for each t∈JP 并且 t∈a
9.      对 t 使用 T-Backfill 策略;
10.     If 使用 T-Backfill 成功调度
11.         continue;
12.     else
13.         使用 HEFT 将 t 调度到 Q 对应任务队列上;
14.         If t 是最后一个任务
15.             标记 a 完成;
16.   end for
17.  end while
18. end while
```

3 实验及分析

本部分应用 MPS、MMHS[19] 和 RoundRobin 算法，通过平均调度时间、不公平度和资源利用率对每种算法的调度结果进行比较，分析基于优先级的可抢占式动态调度策略的性能优劣。实验中随机产生的 DAG 各个参数如表 3 所列。

表 3 实验中随机产生的 DAG 各个参数的取值

顶点个数	5—30	任务数目
每个任务处理数据量	2—11	M
DAG 宽度	0.1—0.8	宽度与并行度成正比
密度	0.2—0.8	相邻两层边的密度
规则性	0.2—0.8	不同层任务分布情况
传输和计算时间比率	0—3	通信与计算代价比率

第一组实验如图 4、图 5 所示，所有 DAG 优先级相同，每个 DAG 间隔 5 个时间单位提交到调度系统，调度器将其调度到 5 个计算资源上的实验结果。从实验结果可以得出：MPS 算法的调度时间与 MMHS、RoundRobin 两种算法的调度时间相差无几，其平均波动范围为[−7.0%，10.5%]；在不公平度指标上，MPS 算法比另两种算法平均降低了 15.5%。

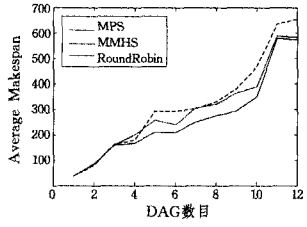


图 4 同优先级 DAG 平均执行时间比较结果

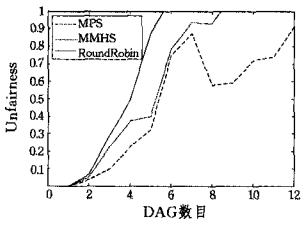


图 5 同优先级 DAG 不公平度比较结果

第二组实验如图 6—图 8 所示，DAG 优先级不同，每个 DAG 间隔 5 个时间单位提交到系统，调度器将其调度到 3 个计算资源上的实验结果。调度时间的波动范围与上述分析一致，资源平均利用率与另外两种算法相比，平均提高 7.9%，但 Unfairness 指标在有 3 个 DAG 时就迅速上升到 1，原因在于低优先级的 DAG 任务长时间得不到资源，由于低优先级的 DAG 长时间等待，导致 Unfairness 指标迅速上升。

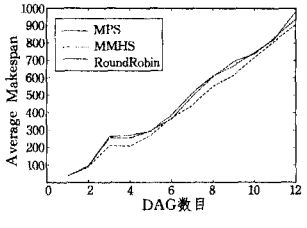


图 6 不同优先级 DAG 平均执行时间比较结果

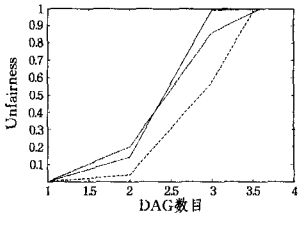


图 7 不同优先级 DAG 不公平度比较结果

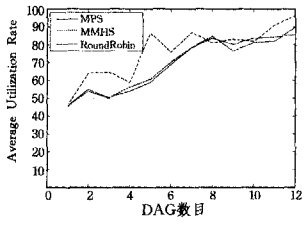


图 8 不同优先级 DAG 资源利用率比较结果

基于上述两组的实验分析，设计最后一组实验为：12 个随机产生的 DAG 在 3 个机器上调度，其中第一个为高优先级可抢占 DAG，到达时间以 5 个时间单位递增。其余的 11 个 DAG 为随机产生的高优先级非可抢占 DAG 和基本优先级 DAG，其在 $t=0$ 时刻同时到达来验证模型与算法。

实验结果如图 9、图 10 所示，高优先级可抢占 DAG 的 $M_{own}=36$，M_{own} 为[36，44]；Unfairness 指标为[0.82，1]。通过跟踪分析工作流的调度过程，该 DAG 的 Unfairness 出现“跳跃”现象，原因在于某个作业的任务占用资源后不可以被抢占，即使是高优先级可抢占 DAG 的作业也不能抢占低优先级作业任务正在使用的资源，但是当此任务执行完释放资源后，便一直可以被高优先级可抢占 DAG 的作业占用，所以会继续保持 Unfairness 指标不变。

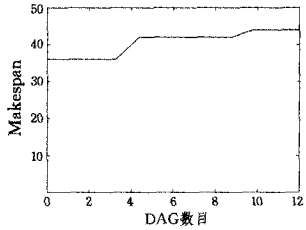


图 9 高优先级可抢占 DAG 的 Makespan 结果

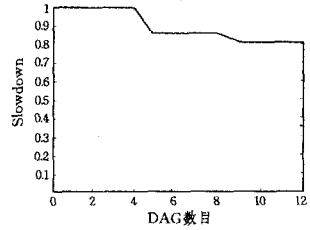


图 10 高优先级可抢占 DAG 的 Slowdown 结果

结束语 针对单 DAG 工作流静态调度存在的不足和多 DAG 工作流调度中的问题，本文提出了云计算环境下抢占式多 DAG 工作流的动态调度模型，模型中任务池和任务队列的引入对工作流中任务的计算和通信的代价与实际差距具有一定的冗余度，方便任务的迁移和负载平衡。并提出依据 P-Fairness 和 T-Backfill 策略的 MPS 算法，通过实验结果的分析比较表明，此算法明显提高了作业的调度公平性和资源利用率。这种调策略既保证了同优先级 DAG 之间调度的公平性，又满足了用户的不同优先级工作流任务的不同需求，提高了系统资源利用率。

参 考 文 献

[1] Gil Y, et al. Examining the Challenges of Scientific Workflows [J]. Computer, 2007, 40(12): 24-32

[2] 黄震春. SCO-GADL：一种用于科学计算的网格工作流描述语言[J]. 计算机科学，2011，38(6)：28-30，34

[3] 桑莉莉. 一种网格工作流动态调度算法[J]. 计算机系统应用，2009，7：45-47

[4] Yu Jia, Buyya R. A Taxonomy of Workflow Management Systems for Grid Computing [J]. SIGMOD Record, 2005, 34(3): 44-49

[5] Sakellariou R, Zhao He-nan. A Low-Cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems[J]. Scientific Programming, 2004, 12(4): 253-262

(下转第 168 页)

问，构造了一个宽容的类型系统；在独立于调度模型的情形下证明了类型系统的可靠性。本质上说，本文的方法借助于静态分析提高了类型检查的能力。

在将来的工作中我们将扩展本文方法的应用范围，研究更为接近实际的并发程序语言，包括在语言中引入方法调用、指针等语法构件；开发相应的类型检查算法。进一步研究在基于类型理论的方法中更多地融入静态分析的思路，提高类型检查的精确度。

参 考 文 献

[1] Sabelfeld A, Myers A C. Language-based information-flow security[J]. IEEE Journal on Selected Areas in Communications, 2003, 21(1): 5-19

[2] Smith G. Improved typings for probabilistic noninterference in a multi-threaded language[J]. J. Comput. Secur., 2006, 14(6): 591-623

[3] Russo A, et al. Closing internal timing channels by transformation[C]// Proceedings of the 11th Asian computing science conference on Advances in computer science: secure software and related issues 2007. Tokyo, Japan: Springer-Verlag, 2007

[4] Terauchi T. A Type System for Observational Determinism [C]// Computer Security Foundations Symposium, 2008. CSF '08. IEEE 21st. 2008

[5] Russo A, Sabelfeld A. Securing interaction between threads and the scheduler in the presence of synchronization[J]. Journal of Logic and Algebraic Programming, 2009, 78(7): 593-618

[6] Zdancewic S, Myers A C. Observational determinism for concurrent program security [C] // Computer Security Foundations Workshop 2003. Proceedings 16th IEEE. 2003

[7] Sabelfeld A. The Impact of Synchronisation on Secure Information Flow in Concurrent Programs[C] // Revised Papers from the 4th International Andrei Ershov Memorial Conference on Perspectives of System Informatics. Akademgorodok, Novosibirsk, Russia: Springer-Verlag, 2001: 225-239

[8] Volpano D, Smith G. Probabilistic noninterference in a concurrent language[C] // Computer Security Foundations Workshop 1998. Proceedings 11th IEEE. 1998

[9] Huisman M, Worah P, Sunesen K. A temporal logic characterisation of observational determinism [C] // Proceedings of the 19th IEEE Workshop on Computer Security Foundations. 2006

[10] Mantel H, Sands D, Sudbrock H. Assumptions and Guarantees for Compositional Noninterference [C] // Computer Security Foundations Symposium 2011 IEEE 24th. 2011

[11] Boudol G, Castellani I. Noninterference for concurrent programs and thread systems[J]. Theoretical Computer Science, 2002, 281 (1/2): 109-130

[12] Barthe G, Nieto L P. Formally verifying information flow type systems for concurrent and thread systems[C]// Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering 2004. ACM: Washington DC, USA, 2004: 13-22

(上接第 148 页)

[6] Haluk T, Salim H, Wu M Y. Performance-effective and Low-complexity Task Scheduling for Heterogeneous Computing[J]. Parallel and Distributed Systems, 2002, 13(3): 260-274

[7] Gilbert C S, Edward A L. A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures[J]. Parallel and Distributed Systems, 1993, 4(2): 75-87

[8] Cao J W, Stephen A J, Sunhash S, et al. GridFlow: Workflow Management for Grid Computing[C] // 3rd IEEE International Symposium on Cluster Computing and the Grid. Tokyo, IEEE Computer Society, May 2003: 198-205

[9] Berman F, et al. New Grid Scheduling and Rescheduling Methods in the GrADS Project[J]. Parallel Programming, 2005, 33 (2): 209-229

[10] Hönig U, Schiffmann W. A Meta-algorithm for Scheduling Multiple DAGs in Homogeneous System Environments[C]// Parallel and Distributed Computing and Systems. Dallas, IEEE Computer Society, November 2006: 147-152

[11] Zhao He-nan, Sakellariou R. Scheduling Multiple DAGs onto Heterogeneous Systems[C] // 20th International Parallel and Distributed Processing Symp. Piscataway: IEEE, 2006

[12] Yu Zhi-feng, Shi Wersong. A Planner-Guided Scheduling Strategy for Multiple Workflow Applications [C]// International Conference on Parallel Processing, Portland, 2008. IEEE Computer Society, September 2008: 1-8

[13] 孔维梁，刘清堂，杨宗凯，等. 基于动态 QoS 的 Web 服务组合[J]. 计算机科学，2012，39(2)：268-272

[14] Amudha T, Dhibyaprabha T T. QoS Priority Based Scheduling Algorithm and Proposed Framework for Task Scheduling in a Grid Environment [C] // International Conference on Recent Trends in Information Technology, Chennai, 2011. IEEE, June 2011: 650-655

[15] Ankur K, Soo-young L. A Stochastic Approach to Estimating Earliest Start Times of Nodes for Scheduling DAGs on Heterogeneous Distributed Computing Systems [C]//19th International Parallel and Distributed Processing Symposium, Denver, 2005. IEEE Computer Society, April 2005: 1530-2075

[16] Kwok Y, Ahmad I. On Multiple Processor Task Scheduling Using Efficient State Space Search Approaches[J]. Parallel and Distributed Computing, 2005, 65(12): 1515-1532

[17] 王万森. 人工智能原理及其应用(第 2 版)[M]. 北京：电子工业出版社，2005：115-120

[18] 付云虹. 基于 backfill 的并行计算作业调度算法研究[D]. 长沙：湖南大学，2007

[19] 田国忠，肖创柏，徐竹胜，等. 异构分布式环境下多 DAG 工作流的混合调度策略[J]. 软件学报，2012，23(10)：2720-2734