计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20151174
54(4):872-885,2017

基于协作相容性的工作流任务分配优化方法

胡海洋[1,2,3] 姬朝配[1] 胡 华[1,2] 葛季栋[3]
[1](杭州电子科技大学计算机学院 杭州 310018)
[2](复杂系统建模与仿真教育部重点实验室(杭州电子科技大学) 杭州 310018)
[3](计算机软件新技术国家重点实验室(南京大学) 南京 210046)
(huhaiyang@hdu.edu.cn)

Method for Optimizing Task Allocation in Workflow System Based on Cooperative Compatibility

Hu Haiyang[1,2,3], Ji Chaopei[1], Hu Hua[1,2], and Ge Jidong[3]
[1](*College of Computer Science, Hangzhou Dianzi University, Hangzhou* 310018)
[2](*Key Laboratory of Complex Systems Modeling and Simulation (Hangzhou Dianzi University), Ministry of Education, Hangzhou* 310018)
[3](*State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing* 210046)

Abstract Task allocation strategy has an important influence on the performance efficiency of workflow system. When allocating tasks among executors, it needs to consider both the capability of each executor and the cooperative compatibility between the executors. Traditional methods for assigning tasks usually only consider the technical skills of executors and ignore the social cooperation compatibility among the executors. Although a few of research works have considered the social cooperation compatibility, they fail to consider how to maintain load balancing among executors when allocating the tasks. Based on the workflow log, cooperative compatibilities among executors are modeled and computed. The relations of interaction tasks are also taken into account. By analyzing the current workload of each executor, a multi-objective joint optimization framework for maintaining load balancing and maximizing the cooperative compatibility among executors is proposed. In this framework, when a new task is assigned, the current workload of each executor that can perform this task will be analyzed and its cooperation capability to other executors that have been assigned those tasks having interactions with this new task will be computed. Several corresponding algorithms are designed for optimizing different objectives and their time complexity is analyzed. Extensive

收稿日期:2015-12-22;**修回日期**:2016-04-28
基金项目:国家自然科学基金项目(61572162,61272188,61572251);浙江省哲学社会科学重点研究基地项目(14JDXX04YB);江苏省自科学基金项目(BK20131277);中央高校基本科研业务费专项资金项目(021714380004);南京大学计算机软件新技术国家重点实验室开放基金项目(KFKT2014B15);南京大学计算机软件新技术国家重点实验室创新项目(ZZKT2013B14)
This work was supported by the National Natural Science Foundation of China (61572162, 61272188, 61572251), the Foundation of Key Research Base for Philosophy and Social Sciences of Zhejiang Province (14JDXX04YB), the Natural Science Foundation of Jiangsu Province of China (BK20131277), the Fundamental Research Funds for the Central Universities (021714380004), the Foundation of State Key Laboratory for Novel Software Technology of Nanjing University (KFKT2014B15), and the Innovation Project of State Key Laboratory for Novel Software Technology of Nanjing University (ZZKT2013B14).

experiments are conducted for comparing the proposed methods which demonstrate the correctness and effectiveness of our approaches.

Key words workflow; task assignment; cooperation compatibility; load balancing; task interaction

摘 要 工作流系统中任务分配策略将对其系统运行性能有很大的影响,在分配任务时不仅需要考虑执行者对相应任务的熟悉度,还需分析执行者之间配合协作的默契程度.传统研究工作在进行工作流任务分配时缺乏对执行者工作负载、执行者之间协作相容性的综合考虑.为了实现有效的任务分配,首先通过分析历史日志的信息,对执行者间的协作相容性进行分析计算,在此基础上综合考虑执行者当前的任务负载,提出了基于协作相容性的、负载均衡式任务分配模型,并给出了多目标联合优化的任务分配方法,可提高整个流程实例的执行效率,并保持执行者间的负载均衡.提出 4 种相应的算法,并分析了算法的时间复杂度,进行了系统性的对比实验,评估了所提出方法的正确性和有效性.

关键词 工作流;任务分配;协作相容性;负载均衡;任务交互

中图法分类号 TP311

在工作流调度中,各任务由工作流引擎调度系统中的资源来操作完成.由于不同的任务分配策略将对工作流管理系统的性能有很大的影响[1],因此需要制定良好的任务分配优化策略,将各任务分配给合适的资源.根据应用领域的不同,工作流系统中的资源可以是人力资源、设备仪器资源、应用程序或网络资源等.其中人力资源在工作流系统中起着重要的作用,他们一般是指具有特定技能的任务执行者,通过相应的角色(role)彼此配合与协作,在工作流系统中调度各类计算资源完成任务.在现代企业中,任务执行者常可承担多类角色用于完成多种任务[2],其对完成不同类型任务常可具有不同的熟悉程度,并且不同人员之间配合协作的默契程度也存在着差异.

任务执行者完成任务的技能熟悉度、彼此间协作的默契度对整个业务过程顺利、高效执行起着重要作用.然而,现有的任务分配算法仅考虑候选执行者的专业能力、兴趣、经验、负载等,忽略了工作流中任务交互时执行者之间的协作相容性.这样的协作相容性可以定义为:“和其他人的凝聚力、熟悉度,和谐、配合度等[2]”.例如一个典型的医疗索赔流程(如第 2 节中的图 1 所示),几个任务以确定的顺序分配给几个角色执行.当一个角色完成分配的任务并提交给下一个任务的执行者执行时,2 个执行者之间可能会存在一些交互以询问和验证一些信息.在这样的任务交互过程中,执行者之间的协作相容性影响着工作的高效性.例如有 2 个员工甲、乙均可完成某个任务,并且甲的个人能力强于乙,然而甲对公司里其他员工的配合并不默契,当工作流中的任务需要员工之间进行交互时,甲的整体工作效率可能反而低于乙的整体效率.

此外,在工作流系统实际过程中,由于存在着多个流程实例,并且各实例到达时间有一定的随机性,候选执行者的工作列表中常存在多个等待处理的任务.在此情形下,一个满负载的执行者很难及时完成所分配的工作流任务.由于执行者当前所有的任务负载情况将对分配任务的最后完成时间有着很大的影响,因此工作流系统在分配任务过程中需考虑到各个任务执行者当前的工作负载情况,即尽可能将任务分配给轻负载(light-loaded)的执行者,从而提高整个工作流系统的性能.

现有的一部分研究工作通过将任务执行者的专业能力、任务成功率、兴趣度、经验等因素转化为模糊数[3-5],并对各影响因素分配一个权重因子,在任务分配时选择综合分数最高的候选者进行分配,但没有认识到任务间存在交互的情形下执行者之间协作的配合度影响.而有些研究尽管考虑了任务分配时上下文环境中任务执行者的一些社会关系、配合度等对任务执行时间的影响[6-9],然而却仅局限于连续的 2 个任务之间的配合度,同时也没有考虑到多实例同时到达的情况下任务执行者工作负载方面的影响.

为了解决以上问题,本文引入了执行者间协作相容性对任务分配影响的内容,通过结合历史日志的信息,对执行者间的协作相容性及任务负载进行了分析计算.在此基础上,给出了任务分配问题的优化建模,提出了基于协作相容性与负载均衡的任务分配算法,提高了整个流程实例的执行效率.

1 相关工作

在工作流调度中基本上任务分配策略是基于角色或组织模型[1]. 它仅仅考虑了资源的角色而没有区分自动型任务和用户型任务. 研究者们通过人的属性扩展了这一概念,例如文献[2]基于流程任务间可能存在的交互,通过一个启发式算法,在任务分配时选择整个实例中合作协作相容性之和最大的执行者执行任务. 文献[3]提出了一种多准则评估模型算法,根据候选者的能力、社会关系和任务属性进行任务分配,但对社会关系如何影响候选者的能力问题没有说明. 文献[4]中提出了一种自主工作流任务分配策略,为任务分配带来了更多的自主权.

文献[5]在研究任务分配时考虑了更多因素(如任务重要性、任务类型、能力、负载、经验),并用模糊理论讨论了关于每一因素权重的大小表示该因素的影响程度;文献[6-7]证明了基于优化社会关系的任务团队组建,如团队的凝聚力等,将使工作流中的任务更加高效地执行;文献[8]提出了一种基于社会关系矩阵处理多实例的任务分配问题;文献[9]采用隐Markov 建模,通过基于候选者任务执行能力和日志中挖掘的连续任务的交互关系,并通过 Viterbi 算法解决任务分配问题;文献[10]提出了一种基于Q 学习的任务分配算法,且在任务分配时考虑了社会关系的影响,缩短了实例的平均执行时间. 然而,以上 5 种方法考虑的社会关系仅仅局限于连续任务的前置执行者,忽略了工作流中非连续任务之间存在交互时社会关系的影响,也未能充分考虑到执行者间的负载均衡.

此外文献[11]在任务分配时不仅考虑到当前待分配任务的候选者,也考虑整个实例执行中合作候选者间的依赖关系,优先选择历史合作次数最多的执行者. 然而,他们都没有考虑到候选者的负载情况. 文献[12]在任务分配时不仅考虑了任务候选者的个体属性,而且考虑了工作流中前一任务执行者对当前候选者的影响,改进了传统的最短处理时间、最短完成时间、平均工作负载和最短工作列表这 4 种任务分配算法,但同样没有充分考虑到工作流任务分配中的执行者工作负载均衡问题. 文献[13-15]提出的动态任务分配策略,主要考虑了众包(crowdsourcing)执行环境中任务执行候选者在任务分配时的当前状况,具有一定的现实意义. 文献[16]将机器学习算法用于工作流事件日志,在任务执行时根据算法生成的分类器推荐一个合适的执行者,以半自动的方法减少手动员工分配. 文献[17-19]提供了解决任务分配问题的一个基本框架,但忽略了任务候选者之间社会关系的影响,也没有考虑到同时到达多实例时的情形.

综上所述,在任务分配时大多数的任务分配算法仅考虑候选执行者的个体能力属性,没有认识到工作流中不同任务交互时执行者之间的协作相容性对流程性能的影响. 本文提出的基于协作相容性的任务均衡分配算法,不仅考虑了执行者当前的工作负载,而且考虑了流程实例运行中任务交互时候选者之间的协作相容性,更好地体现了工作流系统的实际运行情况.

2 基于协作相容性的任务均衡分配模型

2.1 问题描述和基本框架

本文通过一个具体实例来阐释相关问题. 如图 1 所示的一个医疗保险索赔流程[2],该工作流程涉及索赔的受理(receiving)、材料的验证(validating)、理算(settlement)、审批签字(approving)和付款(payment)等任务处理过程,每个任务由不同角色执行的同时又有不同的交互.

该流程运行时,1)客服代表(customer staff)对医疗保险索赔进行记录并将由审查人员(reviewer)进行理赔调查或体检;2)评估人员(evaluator)根据审查人员所提供的信息进行索赔理算(settlement);3)经理(manager)对收到的赔偿款项进行签字确认;4)财务人员(accountant)将相应款项转入客户账户.

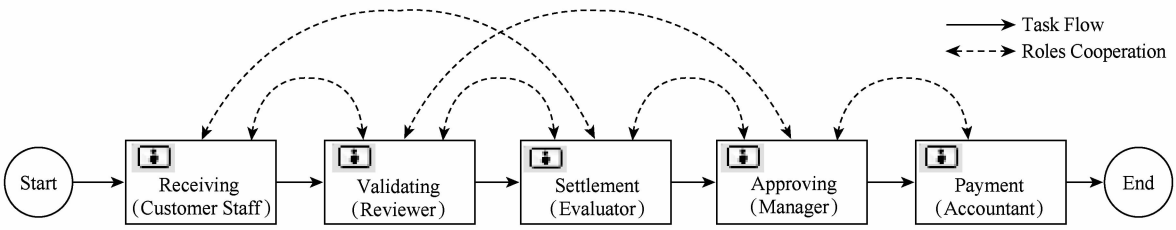

Fig. 1 A process flow of medical insurance claim

图 1 一个医疗保险索赔流程

由于每一任务的角色可能处于不同的物理位置,使得任务执行过程中协作相容性的需求较大.在图1中,各任务上面的圆弧状虚线,表示执行不同任务的角色可能需要进行信息的交互.例如,收到索赔后,审查员可能需要客户服务代表澄清关于某些缺失的索赔信息(如发生事故的确切位置或事件的时间丢失).同样,评估员可能需要咨询审查员关于事故信息的更多细节.最后,财务会计人员可能会在付款之前咨询审查员说明一些支付选项不明的信息(如签字无效等).

因此,这里我们考虑的任务分配问题的场景如下:1)我们要考虑候选任务执行者的任务负载情况,即在分配时优选相对轻载的执行者;2)由于整个实例中不同任务可能存在交互的情况,不同执行者合作时所需的时间开销不同,因此任务分配时还需考虑执行者之间的协作相容性.任务执行者之间的协作相容性越高,交互时所需的时间越少.由于执行者可具备多种角色,即具有执行多种任务的能力,若任务分配时仅考虑优化协作相容性(我们假设一个执行者与自身的协作相容度最大),会导致多个任务分配给同一个执行者,这样大大增加该执行者的工作负载,因此,这2个优化目标之间存在一定的冲突.为了解决这个问题,我们提出了一个基于协作相容性的任务均衡分配基本框架(如图2所示).

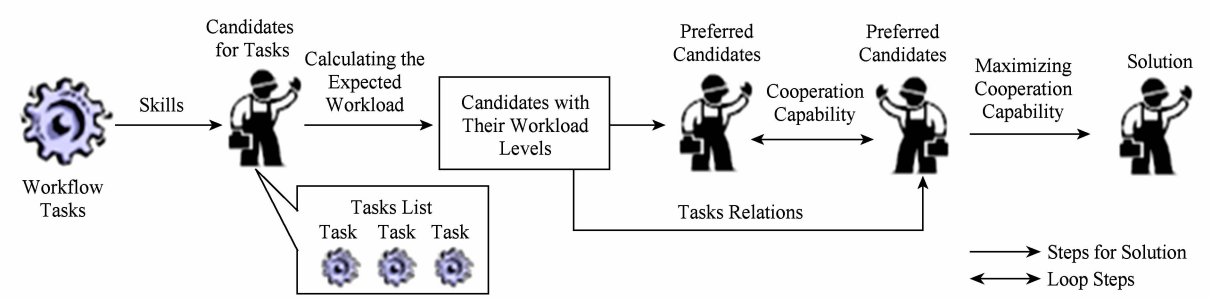

Fig. 2 Framework for tasks allocation based on cooperation capability

图2 基于协作相容性的任务均衡分配基本框架

基于该框架,任务分配的确定过程如下:

1) 根据任务候选执行者之间相对预测负载大小(相关定义在2.2节中给出),将候选执行者划分为轻负载、中负载和重负载3个区间.同一区间内的候选者具有相似的负载.而在具体应用中,对候选者的相对预测负载区间数量和参数的设置,可根据工作环境的实际状况进行灵活调整;

2) 选择轻载区间内的候选者作为待分配任务的新候选者集合;

3) 根据具体任务之间的交互情形,从每一个任务的新候选者集合中,选择一个具有能力执行该任务且同时能最大化交互任务中执行者之间的协作相容性.

在工作流任务分配过程中,本文所做4点假设:

1) 工作流中所有任务候选者之间的协作相容性独立于具体的任务,且执行者之间的协作相容性具有对称性.

2) 执行者间的协作相容性与交互时所需时间成正比,即2个任务执行者间的协作相容性越好,任务执行时交流信息花费的时间越小.

3) 工作流运行过程中执行者总负载包括2部分:任务执行负载、与其他执行者交互所需的负载.其中,任务执行负载是当前预分配任务及执行者的工作列表中所有等待执行任务的负载.

4) 执行者对其工作列表中的任务进行处理时采用"先进先出"方式.

2.2 最优任务分配模型

根据图2中的基本框架,本节将给出具体的计算模型;为了方便描述和分析,表1给出了本文所使用的一些基本符号与定义.

我们用执行者完成其工作队列中所有任务所需的时间来表示他的工作负载.

定义1. 执行者 u_k 当前工作列表中等待处理的任务集为 TA_k,且执行任务 $T_i \in TA_k$ 所需要时间为 t_k^i,若工作列表中等待处理的任务 T_i 有 n_i 个,则 u_k 的当前负载为

$$w_{\text{cur}}(u_k) = \sum_{T_i \in TA_k} n_i t_k^i.$$

定义2. 设执行者 u_k 当前负载为 $w_{\text{cur}}(u_k)$,若任务 T_i 即将分配给他,则 u_k 的预测负载为

$$w_{\text{pred}}(u_k) = w_{\text{pred}}(u_k) + t_k^i.$$

定义3. 设待分配的任务 T_i 的候选执行者集合为 $CE_i = \{u_k\}$,且执行 u_k 的预测负载为 $w_{\text{pred}}(u_k)$,则将候选执行者的负载进行归一化处理后可得到

u_k 的相对预测负载为

$$\hat{w}_{\text{pred}}(u_k) = w_{\text{pred}}(u_k) \Big/ \sum_{u_k \in CE_i} w_{\text{pred}}(u_k).$$

令 $\bar{w}=\frac{1}{m}\sum_{k=1}^{m} w_{\text{cur}}(u_k)$ 为各执行者的平均工作负载,则执行者间的负载均衡情况可通过标准差来表示:

$$WL = \Big(\sum_{k=1}^{m}(w_{\text{cur}}(u_k)-\bar{w})^2/m\Big)^{1/2}. \quad (1)$$

令 A_{ik} 标识任务 T_i 是否被分配给执行者 u_k, cp_{ij} 标识任务 T_i 与 T_j 需要交互,则工作流任务分配时执行者间发生交互时的总协作相容度可表示为

$$CW = \sum_{k=1}^{m}\sum_{v=1}^{m}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} A_{ik}A_{jv}cp_{ij}cw_{kv}. \quad (2)$$

我们的目标是在分析执行者任务负载均衡的基础上,进一步通过使待分配任务的执行者与流程中各交互任务的执行者之间总体协作相容性最大,从而提高系统运行的效率. 该任务分配问题可以描述成一个多目标整数线性规划问题:

$$\begin{aligned} &\max(CW, WL^{-1}),\\ \text{s.t. } &\sum_k A_{ik}=1,\ i=1,2,\cdots,n,\\ &A_{ik}\leqslant X_{ik},\ 1\leqslant i\leqslant n,\ 1\leqslant k\leqslant m,\\ &X_{ik}\in\{0,1\},\ 1\leqslant i\leqslant n,\ 1\leqslant k\leqslant m, \end{aligned} \quad (3)$$

其中,对 WL^{-1} 值的最大化即相当于对 WL 值的最小化. 约束条件 $\sum_{k=1}^{m} A_{ik}=1$ 表明工作流实例中每个任务 T_i 都应该分配唯一的执行者 u_k. 约束条件 $A_{ik}\leqslant X_{ik}$ (X_{ik} 的定义如表 1 所示)表明每个任务必须被分配给有能力完成它的候选执行者.

尽管式(3)是一个整数的线性规划问题,可以使用数学优化工具(如 CPLEX)计算得出最优的分配方法,然而由于问题的规模,这是非常耗时的. 因此,在后面我们首先给出计算执行者间协作相容性的方法,然后提出贪心算法,求解任务分配过程的局部最优解.

Table 1 Definitions of Notations

表 1 相关符号与定义

Notation	Definition
$Task=\{T_i\}$	The set of tasks in workflow system
$U=\{u_i\}$	The set of executors in workflow system
$MCP=\{cp_{ij}\}$	The set of interaction tasks with $cp_{ij}\in\{0,1\}$; if T_i has interaction with T_j, $cp_{ij}=1$; otherwise $cp_{ij}=0$
$MX=\{X_{ik}\}$	The set of roles that an executor can take on, where $X_{ik}\in\{0,1\}$; if executor u_k can take on the role of performing task T_i, $X_{ik}=1$; otherwise $X_{ik}=0$
$MCW=\{cw_{ij}\}$	The set of cooperation capability among executors where $cw_{ij}\in[0,1]$ denotes the amount of cooperation capability between executor u_i and u_j.
t_k^i	The time needed by executor u_k for completing task T_i
$m=\lvert U\rvert$	The number of executors
$n=\lvert Task\rvert$	The number of tasks
$MA=\{A_{ik}\}$	The set of task assignments in which if task T_i is assigned to executor u_i, $A_{ik}=1$; otherwise $A_{ik}=0$
$W_{\text{L}}, W_{\text{M}}, W_{\text{H}}$	$W_{\text{L}}, W_{\text{M}}$ and W_{H} represents the set of executors that currently have light, medium and heavy amount of workload, respectively.

2.3 执行者间协作相容性

当工作流中的任务之间需要交互时,任务执行者间的高协作相容性可以加快他们信息的交流,提高任务执行的速度. 一些电子商务网站如 Epinions.com, Slashdot.org 等通过询问用户记录成员之间的协作相容性来建立员工之间的信任网络或黑名单. 然而,由于员工之间的协作相容性属于个人隐私,从而使得这种通过访问的方式来记录彼此间的协作相容性是非常不合适的.

根据任务执行者合作的多样性,本文主要通过工作流日志里执行者间协作完成流程的时间来度量他们的协作相容性,主要思想如下:对于发生交互的任意 2 个任务,计算进行协作的 2 个执行者之间平均吞吐时间与最小执行时间的差值,相对于 2 个任务中最多与最少的执行时间的差值比值,则该比值越小,协作相容性越大. 协作相容性计算为

$$cw_{kv}=1.0-\frac{t_{\text{avg}}-t_{\min}}{t_{\max}-t_{\min}}\omega, \quad (4)$$

其中,cw_{kv} 表示 u_k, u_v 的协作相容性; t_{avg} 表示 u_k, u_v 配合时执行流程的平均吞吐时间; $t_{\min}$ 表示流程的最小完成时间; $t_{\max}$ 表示流程的最大完成时间; $0<\omega<1$. 显然,由式(4)所得的任务执行者之间的协作相容性

取值范围为 $cw_{kv}\in[0,1]$. 例如,假设基于图 1 的一个流程日志解析得到的部分执行信息如表 2 所示:

Table 2 Parts of Log Information

表 2 部分日志信息

CaseID	Receiving	Validating	Settlement	Time Needed /min
1	Mary	Jack	John	10
2	Mary	Carl	John	12
3	Mary	Jack	John	11

由表 2 可得,流程的平均吞吐时间为 $t_{avg}=(10+12+11)/3=11$ min,Mary 和 Jack 配合时的流程平均完成时间为 $(10+11)/2=10.5$ min;流程的最大完成时间为 12 min,最小完成时间为 10 min;取 $\omega=0.8$,根据式(4)得:Mary 和和 Jack 的协作相容性为 0.8. 同理可得,Mary 和 Carl 的协作相容性为 0.2.

3 任务分配策略

为了阐述本文所给算法的适用场合和相关特点,我们首先给出 3 种单目标的贪心算法,分别针对候选执行者的期望任务负载(expected workload)最小化、流程中所有任务完成的期望完成时间(expected completed time)最短以及基于预测负载的协作相容性最大化;然后在此基础上提出了联合优化执行者负载均衡及执行者间协作相容性的相关算法. 在后面的实验中,我们将从不同的角度分析对比这 4 种算法.

3.1 期望任务负载最小化策略

执行者的期望任务负载最小化算法 ESWL 的主要思想是:每次在进行任务分配时,遍历所有具有执行该任务能力的候选执行者,从中选择期望任务负载最小的候选者执行相应任务. 设在单位时间内系统分配给 u_k 的平均任务数为 m_k(m_k 值可通过日志分析的方法近似获取[20]),令执行者 u_k 当前待完成的任务集为 TA_k,则在 u_k 完成任务集 TA_k 的过程中,系统新分配给 u_k 的新任务数近似为 $m_k\sum_{T_i\in TA_k}t_k^i$,即 u_k 的期望任务负载为 $\lambda_k=|TA_k|+m_k\sum_{T_i\in TA_k}t_k^i$.

下面给出期望工作负载最小化算法 ESWL 的执行过程:

算法 1. ESWL 算法.

输入:执行者角色集合 $MX=\{X_{ik}\}$;

输出:任务分配策略集 $MA=\{A_{ik}\}$.

① $MA=\varnothing$;
② FOR each task $T_i\in Task$ DO
③ $exp\_workload=MAX\_INT$;$k=0$;
④ FOR each $u_j\in U$ DO
⑤ IF($X_{ij}=1$)
⑥ 计算 u_j 的期望任务数 λ_j;
⑦ IF($\lambda_j<exp\_workload$)
⑧ $exp\_workload:=\lambda_j$;
⑨ $k:=j$;
⑩ END IF
⑪ END IF
⑫ END FOR
⑬ $A_{ik}=1$;
⑭ END FOR

ESWL 算法对流程中的每一个待分配任务,需要遍历所有的候选者,因此,其时间复杂度为 $O(mn)$,其中 n 是流程中任务的个数,m 是所有执行者数.

3.2 期望完成时间最短化策略

对于流程的期望完成时间最短算法 ESCT,其主要思想是:在任务分配时,遍历所有具有执行该任务能力的候选者,考察当前的执行者完成及其所携带的工作列表,计算新任务完成的期望时间,期望时间最短的执行者将被挑选出,并将该任务分配给它. 由于执行者 u_k 可承担的任务集为 $Task_k=\{T_i|X_{ik}=1,i=1,2,\cdots,n\}$,则 u_k 完成任务的平均时间为

$$\bar{t}_k=\frac{1}{|Task_k|}\sum_{T_i\in Task_k}t_k^i,$$

若 u_k 当前待完成的任务集为 TA_k,考虑到当 u_k 完成 TA_k 中任务时系统还会分配新任务,则 u_k 完成任务的期望时间为

$$\gamma_k=\sum_{T_i\in TA_k}t_k^i+\bar{t}_k m_k\sum_{T_i\in TA_k}t_k^i.$$

下面给出期望完成时间最短化算法 ESCT 的执行过程:

算法 2. ESCT 算法.

输入:执行者角色集合 $MX=\{X_{ik}\}$;

输出:任务分配策略集 $MA=\{A_{ik}\}$.

① $MA=\varnothing$;
② FOR each task $T_i\in Task$ DO
③ FOR each $u_j\in U$ DO
④ $exp\_time=MAX\_INT$;$k=0$;

⑤ IF($X_{ij}=1$)
⑥ 计算 u_j 完成任务的期望时间 γ_j;
⑦ IF($\gamma_j<exp\_time$)
⑧ $exp\_time:=\gamma_j$; $k:=j$;
⑨ END IF
⑩ END IF
⑪ END FOR
⑫ $A_{ik}=1$;
⑬ END FOR

同样地,该算法对流程中的每一个任务,也需遍历所有的候选者.因此,时间复杂度为 $O(nm)$,其中 n 是流程中任务的个数,m 是所有候选者的个数.

3.3 协作相容性最大化策略

我们首先根据式(4)计算出执行者之间的协作相容性;然后在文献[2]的基础上,给出协作相容性最大化算法 MCW 的主要步骤如下:在任务分配时,遍历所有具有执行该任务能力的候选者,考察当前的执行者与所有交互任务的执行者协作相容性,从中选择协作相容性最大的执行者分配该任务.

下面给出协作相容性最大化算法 MCW 的算法伪码:

算法 3. MCW 算法.

输入:执行者角色集合 $MX=\{X_{ik}\}$、任务交互集合 $MCP=\{cp_{ij}\}$、执行者协作相容集合 $MCW=\{cw_{kv}\}$;

输出:任务分配策略集 $MA=\{A_{ik}\}$.

① FOR each $u_k\in U$ DO
② FOR each $u_v\in U$ DO
③ 由式(4)计算 u_k 与 u_v 间协作相容性 cw_{kv}的值;
④ END FOR
⑤ END FOR
⑥ FOR each $T_i\in Task$ DO
⑦ $max\_coop:=0$; $v:=0$;
⑧ FOR each u_k DO
⑨ IF($X_{ik}=1$)
⑩ $max\_coop[k]:=0$;
⑪ FOR each $T_j\in Task\wedge T_j\neq T_i$ DO
⑫ IF ($cp_i=1$ 且 T_j 未分配)
/*循环所有交互且未分配任务*/
⑬ 在集合中找出与任务 T_i 的执行者 u_k 之间 cw_{kv} 值最大的候选者 u_v;
⑭ $max\_coop[k]:=max\_coop[k]+cw_{kv}$;
⑮ END IF
⑯ END FOR
⑰ END IF
⑱ END FOR
⑲ $k^*:=\arg\max_{1\leqslant k\leqslant m}(\{max\_coop[k]\})$;
⑳ $A_{ik^*}:=1$;
㉑ FOR each $T_i\in Task\wedge T_j\neq T_i$ DO
㉒ IF($cp_i=1$ 且 T_j 尚未分配)
㉓ 找出与 u_{k^*} 之间 cw_{k^*v}最大的执行者 u_v, $A_{jv}:=1$;
㉔ END IF
㉕ END FOR
㉖ END FOR

如上所述,该算法对流程中的每一待分配任务,需要遍历所有的候选者及任务集合,用以找到与所有交互任务执行者之间的协作相容性.因此,时间复杂度是 $O(mn^2)$.

3.4 联合优化策略

我们将考虑在执行者负载相对均衡的基础上,且使得执行者间整体协作相容性最优的任务分配算法.在设计这样的分配算法 MCLB 时,需要考虑到流程中存在任务交互的情形,因此,算法需要 3 个输入集合:任务交互集合 $MCP=\{cp_{ij}\}$、执行者协作相容集合 $MCW=\{cw_{kv}\}$及执行者角色集合 $MX=\{X_{ik}\}$.在任务分配过程中,我们目标是在保持执行者间工作负载相对均衡的基础上,最大化整个流程中交互任务间的执行者协作相容性.为了实现这个目标,我们首先通过一个映射函数,将执行者之间的协作相容性值映射为任务交互时花费的时间,即对 2 个执行者而言,若他们的协作相容性越高,则彼此交互所需的时间越少.设 2 个执行者 u_k 和 u_v 分别执行任务 T_i 和 T_j,他们之间协作相容性为 cw_{kv},则他们交互时间开销可表示为

$$tc(T_i^k,T_j^v)=(1-cw_{kv})(t_k^i+t_v^j)/\beta, \tag{5}$$

其中,β 是协作相容性对时间映射的比例因子.对于任一执行者 u_k,若考虑将任务 T_i 分配给他,则任务 T_i 完成的预测时间为

$$w_{\text{pred}}(u_k)=w_{\text{pred}}(u_k)+\sum_{j=1}^{n}\sum_{v=1}^{m}X_{jv}A_{jv}cp_{ij}tc(T_i^k,T_j^v). \tag{6}$$

由于在任务分配过程中,当分配 T_i 时,可能存在着其他任务尚未被分配给任何执行者,因此,利用式(6)计算 T_i 分配策略时,我们仅考虑 T_i 与那些已分配好执行者的任务间的交互情形.

下面简述 MCLB 算法的执行流程:1)当分配流程实例中一个新任务 T_i 时,统计所有具备承担该任务角色的那些后续执行者,通过定义 2、定义 3 计算他们当前的负载及相对预测负载值,依据相对预测负载值的大小依次将其放到对应的轻负载执行者集合 $W_L=\{u_k\mid\widehat{w}_{pred}(u_k)\in[0,\theta_L^+)\}$、中负载集合 $W_M=\{u_k\mid\widehat{w}_{pred}(u_k)\in[\theta_L^+,\theta_M^+)\}$和重负载 $W_H=\{\widehat{w}_{pred}(u_k)\mid\widehat{w}_{pred}(u_k)\in[\theta_M^+,1)\}$集合中(其中 θ_L^+,θ_M^+ 分别为系统设置的工作负载区间阈值,且 $\theta_L^+<\theta_M^+$). 2)判断新任务与流程其他任务间是否有交互,若无,则将该任务分配给相对预测负载值较小的候选者;否则,对流程中待分配的每个任务,依次遍历 W_L 与 W_M 集合,找到可以执行 T_i 的候选者并且遍历所有与 T_i 需要交互任务,考察它们所有可能处于 W_L 或 W_M 中的候选执行者,计算 T_i 的候选执行者与这些候选执行者间的协作相容性总和. 3)找出可最大化全局协作相容性的任务候选者,并分配任务 T_i,并将实例中与 T_i 有交互的其他尚未分配的任务分配给对应的执行者. 重复以上步骤,直到所有任务已全部分配.

下面给出面向负载均衡的、最大化整体协作相容性的任务分配算法 MCLB 伪码:

算法 4. MCLB 算法.

输入:执行者角色集合 $MX=\{X_{ik}\}$、任务交互集合 $MCP=\{cp_{ij}\}$、执行者协作相容集合 $MCW=\{cw_{kv}\}$;

输出:任务分配策略集 $MA=\{A_{ik}\}$.

① FOR each $u_j\in U$ DO
② 由定义 3 计算 $\widehat{w}_{pred}(u_k)$;
③ END FOR
④ 根据$\{\widehat{w}_{pred}(u_k)\}$,分别生成 W_L,W_M 和 W_H;
⑤ IF (!$IsExistCoop(MCP)$) /* 判断是工作流程是否需要任务交互 */
⑥ FOR $T_i\in Task$ DO
⑦ 利用 ESWL 算法找出当前期望负载最小的后续执行者 u_k;
⑧ $A_{ik}:=1$;
⑨ END FOR;
⑩ ELSE
⑪ FOR each $T_i\in Task$ DO
⑫ $max\_coop:=0$;$v:=0$;
⑬ FOR each $u_k\in W_L\cup W_M$ DO
⑭ IF($X_{ik}=1$)
⑮ $max\_coop[k]:=0$;
⑯ FOR each $T_i\in Task\wedge T_j\neq T_i$ DO
⑰ IF ($cp_i=1$ 且 T_j 尚未分配)
/* 循环所有交互且未分配任务 */
⑱ 在集合 $W_L\cup W_M$ 中考察所有 T_j 的候选执行者$\{u_v\}$,找出 cw_{kv}值最大的候选者 u_v;
⑲ $max\_coop[k]:=max\_coop[k]+cw_{kv}$;
⑳ END IF
㉑ END FOR
㉒ END IF
㉓ END FOR
㉔ $k^*:=\arg\max_{1\leqslant k\leqslant m}(\{max\_coop[k]\})$;
㉕ $A_{ik^*}:=1$;
㉖ FOR each $T_i\in Task\wedge T_j\neq T_i$ DO
㉗ IF($cp_i=1$ 且 T_j 尚未分配)
㉘ 考察集合 $W_L\cup W_M$ 中所有 T_j 的候选执行者$\{u_v\}$,找出 cw_{k^*v}值最大的候选者 u_v;
㉙ $A_{jv}:=1$;
㉚ END IF
㉛ END FOR
㉜ END FOR
㉝ END IF

函数 $IsExistCoop(MCP)$判断输入的流程中是否存在任务交互. 若流程中不存在任务交互时,时间复杂度为 $O(nm)$;当任务间存在交互时,该算法在任务分配时,对每一待分配任务,在遍历相对轻载的候选者集合时,还需在该可能候选者的基础上遍历其他所有可能交互的任务. 因此,算法的时间复杂度是 $O(m^2n^2)$,其中 n 是任务的个数,m 是所有候选者的个数.

3.5 基于最优模型分配的一个例子

为了便于阐述上述方法,现给出了基于图 1 场景的一个简单例子. 假设图 1 中每一任务的角色、执行者如表 3 所示:

Table 3 Task Roles and Candidates

表 3 任务角色及候选者信息

Task	Role	Candidate
Receiving	Customer Staff	Mary, Susan
Validating	Reviewer	Jack, Carl
Settlement	Evaluator	John, Beth
Approving	Manager	Tony, Sam
Payment	Accountant	Clare, Lin

其中,基于图 1 的任务交互矩阵如表 4 所示:

Table 4 Matrix of Task Interaction

表 4 任务交互矩阵

Task	Receiving	Validating	Settlement	Approving	Payment
Receiving	0	1	1	0	0
Validating	1	0	1	1	0
Settlement	1	1	0	1	0
Approving	0	1	1	0	1
Payment	0	0	0	1	0

设执行者之间计算所得的协作相容性矩阵为表 5 所示:

Table 5 Matrix of Cooperation Capability

表 5 协作相容性矩阵

Role (Task)	Customer Staff (Receiving)		Reviewer (Validating)		Evaluator (Settlement)		Manager (Approving)		Accountant (Payment)	
	Mary	Susan	Jack	Carl	John	Beth	Tony	Sam	Clare	Lin
Mary	1.0	0.9	0.8	0.2	0.8	0.4	0.9	0.3	0.5	0.3
Susan	0.9	1.0	0.4	0.7	0.3	0.8	0.9	0.2	0.2	0.7
Jack	0.8	0.4	1.0	0.8	0.2	0.9	0.8	0.3	0.2	0.9
Carl	0.2	0.7	0.8	1.0	0.8	0.5	0.4	0.6	0.8	0.4
John	0.8	0.3	0.2	0.8	1.0	0.9	0.3	0.9	0.7	0.2
Beth	0.4	0.8	0.9	0.5	0.9	1.0	0.8	0.2	0.3	0.8
Tony	0.9	0.9	0.8	0.4	0.3	0.8	1.0	0.8	0.7	0.4
Sam	0.3	0.2	0.3	0.6	0.9	0.2	0.8	1.0	0.1	0.8
Clare	0.5	0.2	0.2	0.8	0.7	0.3	0.7	0.1	1.0	0.9
Lin	0.3	0.7	0.9	0.4	0.2	0.8	0.4	0.8	0.9	1.0

基于上面前提条件,则在运行时有如下情形:

1) 初始时所有候选者的工作列表为空,即任务候选者的待执行任务的负载为 0;当到达第 1 个流程实例时,流程中所有任务的待候选者均为空闲(即都在轻载集合中). 这时,通过最优分配模型,得到总体最大化协作相容性的任务分配情况如下:{receiving: Mary, validating: Jack, settlement: Beth, approving: Tony, payment: Clare}. 总体协作相容性为:4.4.

2) 假设在某一时刻新的流程实例到达时,所有任务的候选者工作列表中都有等待完成的任务,即一定的工作负载,且通过估算执行者的预测负载及相对预测负载值,划分的新候选集合如下:W_L = {Mary, Susan, Carl, Clare, Lin}; W_M = {John, Beth, Tony}; W_H = {Jack, Sam}.

在任务分配时,首选我们考虑的是负载,将对应的轻载集合作为新任务的候选执行者集合. 因此,首先从 W_L 集合中搜索具备执行该任务角色的候选者,直到对应轻载集合搜索完为止,若未发现具备承担该任务角色的执行者,再依次搜索 W_M 与 W_H. 例如在分配任务 receiving 时,由于该任务的 2 个候选执行者均在 W_L 集合中,因此,候选执行者集合就为 W_L 中的 Mary, Susan. 而在搜索任务 validating 的候选者集合时,W_L 中有一个可能的候选者 Carl,则任务 validating 的候选执行者集{Carl}. 通过以上候选执行者集合的确定,然后针对分配的可能情况,选择任务交互时执行者间协作相容性最大的进行相应分配,结果如下:{receiving: Susan, validating: Carl, settlement: Beth, approving: Tony, payment: Clare},总体协作相容性为 3.9.

4 实 验

本节对 ESWL, ESCT, MCW, MCLB 这 4 种算法进行实验比较来分析各个方法的特点与性能. 仿真采用的工作流顺序模型如图 3 所示:

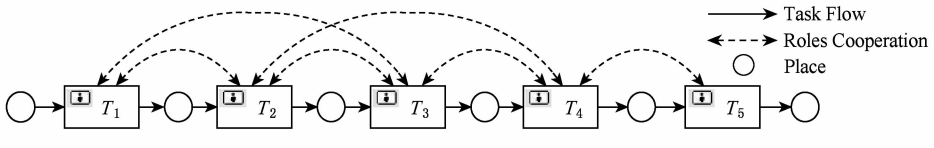

Fig. 3 Workflow model

图 3 工作流模型

在实验中,我们根据相对预测负载值设置轻负载、中负载、重负载区间,即 $W_L=[0,0.34)$,$W_M=[0.34,0.67)$,$W_H=[0.67,1)$. 首先使用 ESWL 算法,产生相应的工作流日志. 根据所产生的工作流日志信息,计算任务执行者之间的协作相容性和执行者任务平均完成时间,将协作相容性值存入协作相容性矩阵;在工作流实例不同到达概率的情况下,使用不同的任务分配算法和相应能力配置进行仿真实验. 对每一实例到达的概率,使用每种算法进行 100 次的仿真,并使用相同的实例队列和迭代次数在其他算法进行仿真,采取平均 100 次的仿真结果作为最终的结果进行分析. 实验中工作流实例到达的概率服从二项分布.

各任务的处理时间设置如表 6 所示. 在实验中,执行者可承担的角色设置为 2 种情况:任务执行者仅具备单一角色、任务执行者可具备多个角色的情况(如表 7、表 8 所示);流程中任务间的交互情形又分为 2 种情况:任务间存在交互、任务间不存在任何交互(如表 9、表 10 所示). 通过结合执行者能力、负载以及任务交互的特点,仿真实验了所有 4 种可能情况下的结果.

Table 6 Processing Time of Tasks

表 6 任务处理时间 min

Task	Processing Time
T_1	9
T_2	12
T_3	17
T_4	10
T_5	6

Table 7 Each Executor Has a Single Role

表 7 任务执行者仅具备单一角色

Executor	T_1	T_2	T_3	T_4	T_5
u_1	1				
u_2	1				
u_3		1			
u_4		1			
u_5			1		
u_6			1		
u_7				1	
u_8				1	
u_9					1
u_{10}					1

Table 8 Each Executor Can Have Several Roles

表 8 任务执行者可具备多个角色

Executor	T_1	T_2	T_3	T_4	T_5
u_1	1				
u_2	1	1			
u_3		1	1		
u_4	1	1			
u_5			1		
u_6			1		
u_7				1	
u_8				1	1
u_9					1
u_{10}					1

Table 9 Tasks Have No Interactions

表 9 任务间无交互情形

Task	T_1	T_2	T_3	T_4	T_5
T_1	0	0	0	0	0
T_2	0	0	0	0	0
T_3	0	0	0	0	0
T_4	0	0	0	0	0
T_5	0	0	0	0	0

Table 10 Tasks Have Interactions

表 10 任务间存在交互情形

Task	T_1	T_2	T_3	T_4	T_5
T_1	0	1	1	0	0
T_2	1	0	1	1	0
T_3	1	1	0	1	0
T_4	0	1	1	0	1
T_5	0	0	0	1	0

任务无交互情况下是在表 7 和表 8 配置下,结合表 9 进行仿真实验. 实验结果如图 4 所示. 通过观察图 4(a)可以看出,4 种算法下工作流实例完成时

间几乎相同. 这是由于在一个任务执行者仅具备单一角色且无任务交互时,其他 3 种算法的本质上是一样的,都是基于最小负载进行任务分配的,能够达到任务的均衡分配. MCW 算法的任务执行者是随机分配的,没有考虑候选执行者实际的负载情形. 通过观察图 4(b),MCLB 算法和 ESCT 算法完成时间几乎相同,ESWL 和 MCW 算法略差且具有交叉性. 这是由于在一个任务执行者可具备多个角色下,ESWL 算法仅以期望任务负载作为任务分配的立足点,并不能正确地反映候选者的实际任务负载. 仿真的结果表明,在没有任务交互的情况下,MCLB 算法仍然取得了较好的性能.

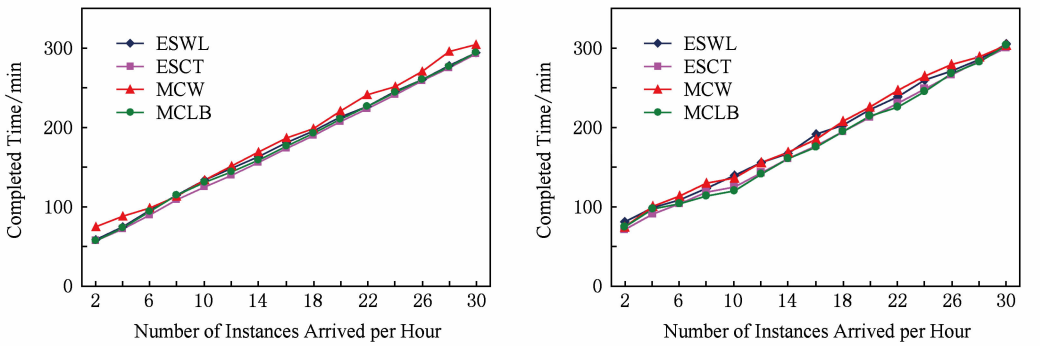

Fig. 4 Completed time of workflow instances with no task interactions

图 4 无任务交互情况下工作流实例的完成时间

我们现在考虑在工作流实例中任务有交互情况下,实验结果如图 5 所示. 通过观察图 5 可以看出,在 2 种场景下,MCLB 算法的结果都是最好的,MCW 算法最差. 这是由于存在任务交互时,MCLB 算法不仅考虑了任务的负载,还考虑了交互任务执行者的协作相容性,一定程度上缩短了任务交互时的花费时间. 而 ESCT 算法仅考虑了期望完成时间,ESWL 算法仅仅考虑期望任务负载. 尽管 MCW 算法考虑了任务候选者与前置所有任务候选者的协作相容性,但却忽略了多实例同时到达的场景,也即忽略了任务负载的影响. 同时,从图 5(a)和图 5(b)的结果对比中可以发现,MCLB 算法在后一种场景下的效果,要比前一种场景下的优势更大,这是由于实验中任务候选者增多时任务选择性变得更多.

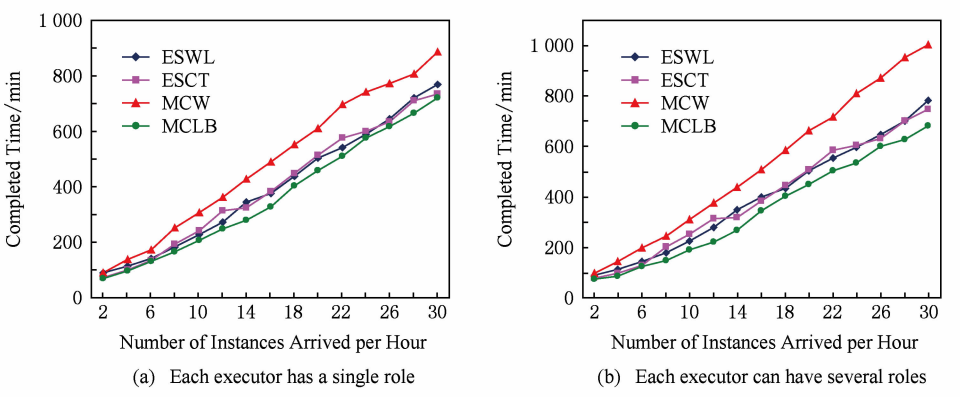

Fig. 5 Completed time of workflow instances having task interactions

图 5 存在任务交互情况下工作流实例的完成时间

下面我们考察任务执行者之间的协作相容性对工作流实例处理时间的影响. 当工作流实例中的各任务间无交互情形时,4 种任务分配算法中工作流实例的平均处理时间总是大小相等,即约为 54 min;而当工作流任务存在交互情形时(如图 6 所示),4 种任务分配算法的执行时间是不同的. 使用 MCW 算法和 MCLB 算法时,工作流实例的平均处理时间总是优于其他 2 种算法. 其中,图 6(a)中,MCLB 算法下工作流实例的平均处理时间比 ESCT 算法约少 2 min,比 ESWL 算法约少 3 min. 图 6(b)中,MCLB 算法比 ESCT 算法约少 2.5 min,比 ESWL 算法约少 1.3 min. 相比之下,MCW 算法比 ESCT 算法约

少 5 min,比 ESWL 算法约少 6 min. MCW 算法下的平均处理时间最小的原因在于,它总是能够使当前任务执行者的协作相容性最大,尤其在任务执行者具备多个角色时,会将连续的多个交互任务分配给同一执行者. 同时,通过将图 6(a)和图 6(b)的结果对比可以发现,在任务执行者可具备多个角色的情况下,使用 ESCT 算法得到的效率比 ESWL 算法要差. 这是由于 ESCT 算法不能根据一个任务执行者可具备多个角色这一特点进行灵活地分配,造成前面任务候选者的选择严重影响后面其他任务的候选者选择情况.

我们现在分析在 4 种任务分配算法下任务执行者的负载均衡,其中执行者的总负载为所分配任务的总执行时间加上实际运行中交互时的花费时间. 图 7 中给出了相关的实验结果. 从图 7(a)结果可以看出,当任务执行者仅具备单一角色时,MCLB 算法能够使任务所有候选者的总负载相对均衡,而其他 3 种算法下任务候选者的负载情况相差较大. 例如任务 T_3 的候选者 u_5,u_6,MCLB 算法下 u_5 比 u_6 多 3.15%,ESWL 算法下多 13.26%,ESCT 算法下多 23.74%,而 MCW 算法下 u_5 比 u_6 少 68.04%. 从图 7(b)结果可以看出,一个任务执行者可具备多个角色时,MCLB 算法同样能够均衡全局任务执行者的负载,而其他 3 种算法下任务的执行者负载不能保持相对均衡. 例如 T_4 的所有候选者有 u_7,u_8,T_5 的所有候选者有 u_8,u_9,u_{10}. 尽管 u_8 可以同时执行 2 个任务,但 MCLB 算法下 2 个任务所有候选执行者的负载相对于其他算法,仍然具有较好的均衡效果.

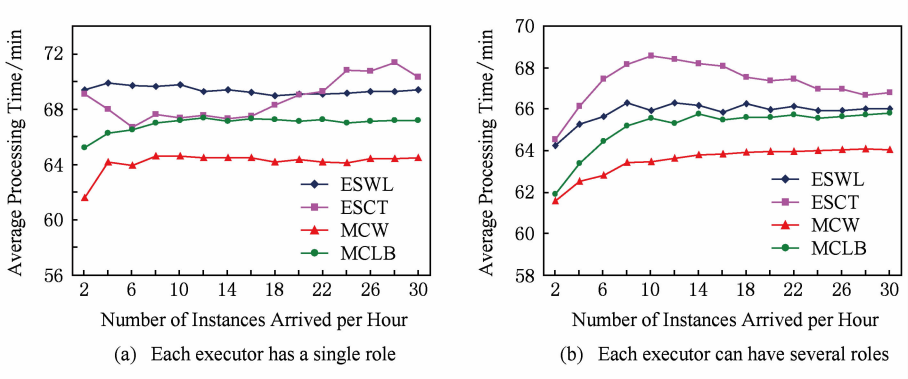

Fig. 6 Average processing time of a workflow instance having task interactions

图 6 存在任务交互情形时工作流实例的平均处理时间

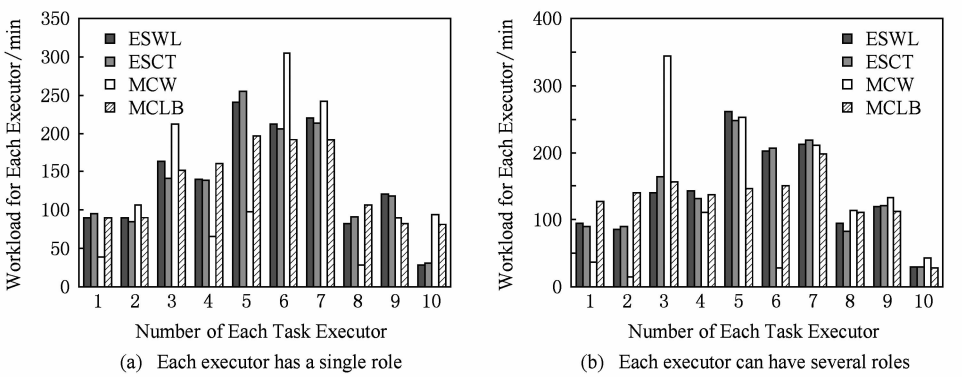

Fig. 7 Workload for each executor when 20 instances having task interactions arrived per hour

图 7 每小时到达 20 个实例时任务交互情况下执行者的工作负载

5 结 论

本文研究了基于协作相容性的工作流任务分配问题及算法. 通过对工作流中任务之间的交互与否以及执行者之间的协作相容性对工作流性能的影响进行建模,并在考虑负载均衡的基础上,通过将交互任务的执行者协作相容性整体最大化以实现最终的任务分配,减少了流程实例的平均吞吐时间,提高了工作流的整体性能. 通过实验得出,基于协作相容性的任务均衡分配方法对工作流中是否存在交互任务的情况均有良好的执行效率.

由于执行者之间协作相容性大小涉及的因素可能有很多,因此本文计算协作相容性的方法有待进一步细化.同时,由于具体的工作流流程有顺序、选择、并行结构,而本文只关注了顺序结构.因此,未来的研究包括3个方面:1)通过考虑更多的因素,进一步完善协作相容性的计算方法;2)考虑工作流中具有选择、并行结构时任务执行者间协作相容性的影响;3)考虑任务间存在转换概率时的协作相容性概率期望值.

参 考 文 献

[1] Aalst W M P, Van Hee K M. Workflow Management: Models, Methods, and Systems [M]. Cambridge, MA: MIT Press, 2004

[2] Kumar A, Dijkman R, Song M. Optimal resource assignment in workflows for maximizing cooperation [G] // LNCS 8094: Proc of the 11th Int Conf on Business Process Management. Berlin: Springer, 2013: 235-250

[3] Shen M, Tzeng G H, Liu D R. Multi-criteria task assignment in workflow management systems [C] //Proc of the 36th Annual Hawaii Int Conf on System Sciences. Piscataway, NJ: IEEE, 2003: 1-9

[4] Chen Chuanbo, Zhao Weiwei. Strategy for a task assignment of workflow system [J]. Journal of Huazhong University of Science and Technology: Natural Science Edition, 2005, 33(6): 20-22 (in Chinese)
(陈传波, 赵伟伟. 一种自主工作流任务分配策略[J]. 华中科技大学学报: 自然科学版, 2005, 33(6): 20-22)

[5] Xiao Zhengjin, He Qinming, Chen Qi. A multilevel model of task assignment in fuzzy situations of workflow [J]. Journal of Computer Research and Development, 2007, 44(2): 302-309 (in Chinese)
(肖郑进, 何钦铭, 陈奇. 模糊环境中工作流任务分配的多级模型[J]. 计算机研究与发展, 2007, 44(2): 302-309)

[6] Jordan M H, Feild H S, Armenakis A A. The relationship of group process variables and team performance a team-level analysis in a field setting [J]. Small Group Research, 2002, 33(1): 121-150

[7] Sanders K, Nauta A. Social cohesiveness and absenteeism the relationship between characteristics of employees and short-term absenteeism within an organization [J]. Small Group Research, 2004, 35(6): 724-741

[8] Bajaj A, Russell R. AWSM: Allocation of workflows utilizing social network metrics [J]. Decision Support Systems, 2010, 50(1): 191-202

[9] Lin S, Luo Z, Yu Y, et al. Effective team formation in workflow process context [C] //Proc of the 2013 Int Conf on Cloud and Green Computing. Piscataway, NJ: IEEE, 2013: 508-513

[10] Yu Yang, Wang Ying, Liu Xingmei, et al. Workflow task assignment strategy based on social context [J]. Journal of Software, 2015, 26(3): 562-573 (in Chinese)
(余阳, 王颖, 刘醒梅, 等. 基于社会关系的工作流任务分派策略研究[J]. 软件学报, 2015, 26(3): 562-573)

[11] Yang H, Wang C, Liu Y, et al. An optimal approach for workflow staff assignment based on hidden Markov models [G] //LNCS 5333: Proc of OTM 2008 Workshops. Berlin: Springer, 2008: 24-26

[12] Xu J, Huang Z, Yu Y, et al. A performance analysis on task allocation using social context [C] //Proc of the 2nd Int Conf on Cloud and Green Computing. Piscataway, NJ: IEEE, 2012: 637-644

[13] Ho C J, Vaughan J W. Online task assignment in crowdsourcing markets [C] //Proc of the 26th AAAI Conf on Artificial Intelligence. Menlo Park, CA: AAAI, 2012: 45-51

[14] Ho C J, Jabbari S, Vaughan J W. Adaptive task assignment for crowdsourced classification [C] //Proc of the 30th Int Conf on Machine Learning. New York: ACM, 2013: 534-542

[15] Björnson E, Jorswieck E. Optimal resource allocation in coordinated multi-cell systems [J]. Foundations and Trends in Communications and Information Theory, 2013, 9(2): 113-128

[16] Liu Y, Wang J, Yang Y, et al. A semi-automatic approach for workflow staff assignment [J]. Computers in Industry, 2008, 59(5): 463-476

[17] Huang Z, Lu X, Duan H. A task operation model for resource allocation optimization in business process management [J]. IEEE Trans on Systems, Man and Cybernetics, Part A: Systems and Humans, 2012, 42(5): 1256-1270

[18] Menon A, Tamuz O, Gulwani S, et al. A machine learning framework for programming by example [C] //Proc of the 30th Int Conf on Machine Learning. New York: ACM, 2013: 187-195

[19] Huang Z, Aalst W M P, Lu X, et al. Reinforcement learning based resource allocation in business process management [J]. Data and Knowledge Engineering, 2011, 70(1): 127-145

[20] Han Rui, Liu Yingbo, Wen Lijie, et al. A probabilistic approach to analyze and adjust time constraints in workflow management System [J]. Journal of Computer Research and Development, 2010, 47(1): 157-163 (in Chinese)
(韩锐, 刘英博, 闻立杰, 等. 工作流管理系统中一种概率性分析和调整时间约束的方法[J]. 计算机研究与发展, 2010, 47(1): 157-163)

Hu Haiyang, born in 1977. PhD and Professor. His main research interests include workflow system and parallel computing.

Hu Hua, born in 1964. Professor and PhD supervisor. His main research interests include workflow system and database theory.

Ji Chaopei, born in 1987. Master. His main research interests include workflow system and business process management.

Ge Jidong, born in 1978. PhD and associate professor. His main research interests include workflow system and business process management.

2017 年《计算机研究与发展》专题(正刊)征文通知

——边缘计算

伴随着计算机硬件和网络技术的发展,计算模式从大型主机计算演进到 C/S 模式的网络计算,再到云计算.然而,基于云计算的电子商务平台、评级服务、搜索引擎、在线社交网络等集中式服务采集个人在线行为和社交数据而易导致隐私泄露;从用户到云的完全授权的应用程序和系统控制,要求客户端到云的单方面信任,阻碍了用户之间建立更细粒度的信任;同时,随着物联网、移动互联网的发展,云计算亦失去和浪费了现代个人设备的计算,通信和存储能力;最后,云计算亦难以满足强实时需求、大数据量交互和处理、本地化位置相关计算的需求.边缘计算(Edge Computing)作为一种新的范式应用而生,将计算应用,数据和服务的前沿从集中式节点推向网络边缘,其中计算和存储资源放置在互联网的边缘,靠近移动设备,传感器和最终用户,形成了云计算有益的补充,通过资源协同获得更好的用户体验.

当前,学术界和工业界已经针对边缘计算模式的体系结构、关键理论与科学问题、技术挑战等开展探索,并形成许多初创成果,推动了边缘计算的深入研究.美国自然科学基金会于 2016 年首次以边缘计算为主题设立研究计划资助科学家研究,IEEE 和 ACM 于 2016 年联合发起召开了首届边缘计算研讨会.2016 年 11 月 30 日,中国边缘计算产业联盟成立;通信领域的 5G 标准正在制定与形成中,边缘计算相关标准也将成为未来 5G 标准的重要组成部分.

《计算机研究与发展》将于 2017 年 12 月出版“边缘计算”专题,欢迎相关领域的专家学者和科研人员踊跃投稿.现将专题论文征集的有关事项通知如下.

征文内容

本专题包括(但不限于)下列主题:

- 边缘计算的计算模型
- 边缘计算的体系结构
- 边缘计算的资源管理与协同
- 边缘-中心协同与任务卸载
- 边缘计算中的资源虚拟化
- 物联网应用的边缘计算模式
- 无线网络边缘计算中的频谱与资源联合优化
- 边缘计算中 QoS 和 QoE 保障机制
- 边缘计算中的跨层优化
- 边缘计算的性能评价与优化
- 边缘计算的信任管理、安全与隐私保护
- 边缘计算的应用示范

投稿要求

1) 论文应属于作者的科研成果,数据真实可靠,具有重要的学术价值与推广应用价值,未在国内外公开发行的刊物或会议上发表或宣读过,不存在一稿多投问题.作者在投稿时,需向编辑部提交版权转让协议.
2) 论文应包括题目、作者信息、摘要、关键词、正文和参考文献,论文一律用 Word 排版,论文格式请参考《计算机研究与发展》近期文章.
3) 论文需附通信作者的联系方式、联系地址及 E-mail 信息.
4) 论文请通过期刊网站(http://crad.ict.ac.cn)进行投稿,并在作者留言中注明“边缘计算 2017 专题”(否则按自由来稿处理).

重要日期

征文截稿日期:2017 年 9 月 24 日
录用通知日期:2017 年 11 月 10 日
最终稿提交日期:2017 年 11 月 17 日
出版日期:2018 年 2 月或 3 月

特邀编委

邓晓衡 教授 中南大学 dxh@csu.edu.cn
李东升 教授 国防科学技术大学 dsli@nudt.edu.cn
吴 帆 教授 上海交通大学 fwu@cs.sjtu.edu.cn

联系方式

编 辑 部:crad@ict.ac.cn,010-62620696,010-62600350
通信地址:北京 2704 信箱《计算机研究与发展》编辑部 邮政编码:100190