一种基于交流成本的全球软件任务调度方法 *

钟南海[1,2], 刘大鹏[1], 肖俊超[1]
(1. 中国科学院 软件研究所 互联网技术软件实验室, 北京 100190;2. 中国科学院 研究生院, 北京 100049)

摘 要: 在全球软件开发中,由于时区、地理位置、文化和语言等各种因素,交流和协作变得非常困难,如果在进行任务调度的时候不考虑交流对整个项目所造成的影响,则有可能使整个项目开发的总成本增加,从而给项目带来很大的风险。通过采用基于交流成本的任务调度方法,在项目初期就考虑交流风险,并对任务进行调度,从而能有效减少该风险对项目可能造成的损失。通过一个示例项目将该方法与传统的基于阶段的方法进行对比,说明了交流成本对整个项目成本的确有很重要的影响,并且使用基于交流成本的任务调度方法能有效降低项目开发的总成本。

关键词: 全球软件开发; 交流成本; 任务调度
中图分类号: TP311.52 **文献标志码**: A **文章编号**: 1001-3695(2010)08-2942-06
doi:10.3969/j.issn.1001-3695.2010.08.036

Communication-cost based task scheduling method in global software development

ZHONG Nan-hai[1,2], LIU Da-peng[1], XIAO Jun-chao[1]
(1. *Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China*; 2. *Graduate School, Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract: In global software development, communication and coordination become more difficult than traditionally co-located software development, because the global teams may be distributed in different locations, which may cause time-zone, language, culture differences, etc. If the impact of such inefficient communication is not considered, the overall project cost may be very high because of a high communication cost. In order to reduce the possible damage which the communication risk may cause to the global software development project, this risk should be considered when scheduling tasks. This paper proposed a communication-cost based method to support task scheduling to allocate tasks to different employees in distributed locations, which could effectively reduce the communication risk and made the overall project cost a low level. This method was compared with a phased-based task scheduling method with an example global software development project, and it proves that it really can effectively reduce the overall project development cost.

Key words: global software development; communication cost; task scheduling

0 引言

在全球软件开发模式下,各开发小组分布在不同的国家或位于不同的地理位置,他们通过协作完成同一个项目或系统[1]。全球软件开发模式具有缩短产品进入市场的时间、降低总成本、充分利用资源等优势[2],因此越来越多的软件产品都开始采用该模式进行开发。但是,在全球软件开发中,由于存在地理位置分散、时区不同、文化和语言差异等因素,不同站点间的交流和协作变得非常困难[2,3]。低效的交流会造成生产率下降及产品开发时间的延长[4],从而使开发成本增加,因此交流问题成为全球软件开发中一个不得不面对的挑战和风险。

交流主要发生在具有依赖关系的任务的执行人员之间。如果两个任务相互独立,则可以认为执行人员之间不存在交流[5]。为减少交流对全球软件开发所造成的影响,除了采取更加有效的交流方式(如邮件、电话、视频会议等)外,在进行任务调度时也应该考虑采取适当的策略减小交流成本对整个项目成本的影响。

为解决上述问题,本文提出一种在全球软件开发环境下基于交流成本的任务调度方法。该方法根据任务之间的依赖关系,在进行任务分配时考虑人员位于不同站点之间的时区、文化差异等因素,对任务进行合理调度,从而有效提高人力资源利用率,降低开发成本,确保项目成功。

1 相关工作

1.1 全球软件开发中的任务调度方法

Jalote 等人[6]提出了一个基于任务图关键路径的启发式算法,并将分布式环境下的一些关键因素及描述人的能力的资源模型作为其任务调度模型的输入,最后为指定的项目生成一个项目计划。由于该方法基于任务图的关键路径,并且在进行

收稿日期: 2010-01-05; **修回日期**: 2010-02-27 **基金项目**: 国家自然科学基金资助项目(90718042,60903051);国家"863"高技术研究发展计划基金资助项目(2007AA010303);国家"973"重点基础研究发展计划基金资助项目(2007CB310802)

作者简介:钟南海(1984-),男,湖北武汉人,硕士研究生,主要研究方向为软件过程建模、软件过程仿真(zhongnanhai@itechs.iscas.ac.cn);刘大鹏(1981-),男,山东威海人,博士,CCF 会员,主要研究方向为软件过程建模、软件过程仿真;肖俊超(1978-),男,吉林珲春人,助理研究员,博士,主要研究方向为软件过程建模、软件过程仿真.

任务分配时还考虑资源的使用,因此能最大限度地利用资源,并减少项目的执行时间。但是,该方法并没有考虑执行任务的人员之间的交流情况,因此虽在一定程度上提高了资源利用率,却有可能造成项目开发成本的增加。

Lamersdorf 等人[7]提出了一种能系统地将分布式环境下的各种因素进行综合考虑,并为任务分配提供决策支持的模型。该模型使用一种随机化算法对一系列场景进行仿真,最后返回一组排好序的分配列表,提供给项目经理进行选择。该方法能够较好地预计到项目可能带来的各种风险,并能根据不同的项目目标(如质量、成本、时间)来生成不同的任务分配方案,从而为项目经理提供决策支持。但是该方法只考虑了站点的各种因素,对于站点内人员的能力没有考虑,因此对任务不能进行更细致的分配。

1.2　全球软件开发中的交流成本模型

Ramasubbu 等人[8]主要分析了团队组织因素和软件的复杂性对交流成本的影响,并将建立的交流成本模型应用到软件开发当中,所以该方法能帮助项目经理在分布式开发环境中制定一个动态的协作或交流策略来进行软件的迭代开发,但是它并没有考虑全球化软件开发中各种基本因素(如时区、地域、文化差异等)对交流产生的影响。

Espinosa 等人[5,9]建立了一个基于时区和地域的二元交流模型,该模型主要考虑不同的两个人分别在相同时区不同地方、相同时区相同地方、不同时区相同地方以及不同时区不同地方四种情况下在完成存在依赖关系的两个任务时的各种交流情况。该模型虽然考虑了时区、地域因素对成本的影响,但是对文化差异因素却缺少考虑。

2　交流成本模型

本文主要考虑任务分配时单个人与人之间的交流情况,认为语言差异是文化差异的一部分,因此这里只考虑全球软件开发的三个基本因素:地理位置、时区差别、文化差异对交流产生的影响。Espinosa 的交流成本模型描述了全球开发中的两个人在各种不同情况下的交流情形,但是该模型只考虑了时区和地域因素,并没有考虑文化差异因素对交流产生的影响,而文化差异在全球软件开发的协作交流中却产生着越来越重要的影响,因此本文在 Espinosa 模型的基础上增加了对文化差异因素的描述,得到本文所使用的全球软件开发环境下的交流成本模型。

Espinosa 将交流成本分为协作成本 C_c 和脆弱性成本 V_c 两部分,协作成本 C_c 主要包括使用通信工具的通信成本和由于等待造成的延迟成本,脆弱性成本 V_c 是交流障碍导致的信息不畅造成的成本,因为不畅的信息可能造成对请求的反复确认、对任务进行重做以及更长等待时间等问题。

假设现在存在两个人员 a 和 b,其中 a 位于站点 A,b 位于站点 B,a 执行任务 T_0,b 执行任务 T_1,且 T_1 依赖于 T_0,则 Espinosa 模型对应的交流成本可以使用如下公式来表示:

$$\mathrm{CommCost}(a,T_0,b,T_1)=C_c+V_c$$

其中:CommCost 函数表示交流成本,协作成本和脆弱性成本分别按如下方式计算

$$C_c=l(A,B)+2m(A,B)+P_d$$

$$V_c=P_u\times(0.3\times R_w\times P_c+C_c)$$

相关参数说明如下:

a)P_c 表示 a 完成任务 T_0 所花费的成本,P_d 表示 b 等待时的延迟成本。

b)$l(A,B)$ 表示 A、B 两个站点维持一条通信连接的成本,该成本与交流的方式有关,忽略面对面交流的成本,设同步方式的交流成本使用 $ls(A,B)$ 表示,异步方式的交流成本使用 $la(A,B)$ 表示,并且 a 与 b 交流时可能同时采用两种交流方式,同步交流的概率为 p,异步交流的概率为 q,则 $l(A,B)=p\times ls(A,B)+q\times la(A,B)$,且 $p+q=1$。

c)$m(A,B)$ 表示 A、B 两个站点发送一条通信信息的成本,ma 表示异步交流的成本,ms 表示同步交流的成本,则同样的,$m(A,B)$ 可表示为

$$m(A,B)=p\times ms(A,B)+q\times ma(A,B),p+q=1$$

d)P_u 表示在不同情况下信息不畅的概率,如面对面情况 $P_u(F)=0.1$,地理位置分散情况 $P_u(D)=0.3$,时区差异情况 $P_u(T)=0.5$,地理位置不同且时区存在差异情况 $P_u(DT)=0.7$。

e)R_w 表示对于发生信息不畅情况下任务需要重做的概率。

在计算脆弱性成本时,Espinosa 模型并没有考虑文化差异因素,因此需要对 V_c 进行扩展,加入文化差异的因素。假设文化差异越大,则交流障碍越明显,造成的脆弱性成本越高。$cd(A,B)$ 表示 A、B 两个站点的文化差异,则 V_c 扩展后表示如下:

$$V_c=P_u\times(0.3\times R_w\times P_c+C_c)/(1-cd(A,B))$$

其中:文化差异函数 $cd(A,B)=(|A.\mathrm{PDI}-B.\mathrm{PDI}|+|A.\mathrm{IDV}-B.\mathrm{IDV}|+|A.\mathrm{UAI}-B.\mathrm{UAI}|)/300$。

PDI、IDV、UAI 为 Hofstede 五维文化模型[10]中的三个维度:

a)PDI: power distance index, 权利距离指数,表示一种文化对社会不平等的忍耐度。

b)IDV: individualism/collectivism, 个人主义指数,主要反映一个集体的凝聚力。

c)UAI: uncertainty avoidance index,不确定性避免能力指数,表示对发生未知情况的一种容忍能力。

Hofstede 模型是被最为广泛使用的文化模型。Borchers[11]将该文化模型应用到软件工程中,并且只选取了其中 PDI、IDV、UAI 三个维度,来说明文化差异对软件工程也有很大的影响。本文也选取了这三个维度,用于表示文化差异,说明其对交流所造成的影响:文化差异越大,交流越困难,可能造成的成本越高。

$l(A,B)$、$m(A,B)$ 和 $cd(A,B)$ 三个函数值都是由 a、b 所在的站点 A 和 B 决定的,$l(A,B)$ 和 $m(A,B)$ 取值情况可以由项目经理进行输入,而文化差异根据上面的表达式进行计算。

3　全球软件开发环境下基于交流成本的任务调度方法

本文提出的在全球软件开发环境下基于交流成本的任务调度方法主要分为四个部分,如图 1 所示。

a)任务集合:软件项目中所有任务的集合。每个任务由一组属性来描述,包括任务规模、任务类型、该任务的前置任务集合等。图中的箭头表示任务间的依赖关系。

b)人员集合:软件项目中所有人员的集合。人员分布在不同的站点,每个人员由一组属性来描述,包括技能、角色、站点等。

c)调度引擎:根据软件项目的任务集合和人员集合中的信息,计算交流成本,并根据交流成本对任务进行调度,生成相应的分配列表。

d)分配列表:其为调度引擎的输出,即所有任务的分配情况,最后可以根据该分配情况生成表示项目计划的甘特图。

下面对该方法的各个部分进行详细描述。

3.1 任务集合

任务集合包含所有需要完成的任务,其表示形式如下:

$$TaskSet = \{Task_1, Task_2, \cdots, Task_i, \cdots, Task_n\}$$

其中:TaskSet 表示任务集合,每一项表示一条任务,n 为任务数目。每个任务 Task 表示如下:

$$Task = (ID, name, description, size, taskType, needRole, PreTasks, assigned, finished, terminal)$$

其中:

a)ID 是任务的惟一标志;

b)name 表示任务的名称;

c)description 是对任务的描述信息;

d)size 表示任务的规模,如编码类任务 10KLOC;

e)taskType 表示任务的类型,如 Java、C ++ 等;

f)needRole 表示执行任务人员的角色,如开发人员(Dev)、项目经理(PM)、需求人员(Req)、质量保证人员(QA)等;

g)PreTasks 表示该任务的前置任务集合,只有当所有前置任务完成后,该任务才能执行;

h)assigned 表示该任务是否已分配出去;

i)finished 表示任务是否已完成;

j)terminal 表示该任务是否为结束任务,如果为真,则该任务完成标志着整个项目的结束;在一个任务集合中有且仅有一个终节点。

由于任务之间存在前后序的依赖关系,相互依赖的两个任务可以使用一条有向边连接,由所有的任务集合以及表示依赖关系的有向边,最后会形成一个有向无环图,这个图称为任务图[7,12]。例如一个任务集合 TaskSet 中有三个任务 $Task_1$、$Task_2$、$Task_3$,分别表示如下:

$$Task_1 = (1, "需求", "任务 1", 20\ 个, "UML", "Reg", \{\}, false, false, false)$$

$$Task_2 = (2, "编码", "任务 2", 15KLOC, "Java", "Dev", \{Task1\}, false, false, false)$$

$$Task_3 = (3, "测试", "任务 3", 20\ 个, "Test", "QA", \{Task_2\}, false, false, true)$$

任务 3 为标志项目结束的任务,依赖于任务 2,而任务 2 依赖于任务 1,则所形成的任务图如图 2 所示。

3.2 人员集合

人员集合包含所有参与该项目的人员,在全球化软件开发环境中,不同的开发站点分布在不同的地方,而这些人员分别位于这些分散的站点中,具有不同的时区、地理位置、语言和文化。人员集合描述如下:

$$EmployeeSet = \{Employee_1, Employee_2, \cdots, Employee_i, \cdots, Employee_m\}$$

其中:Employee 表示项目当中的一个人员,m 表示人员数目,对每个人员的描述为

$$Employee = (ID, name, description, ProductivitySet, salary, timezone, location, country, ActRoles, available)$$

其中:

a)ID 是人员的惟一标志。

b)name 为人员的姓名。

c)description 为对该人员的描述信息。

d)ProductivitySet 为该人员的生产率集合,由于一个人员可能拥有多种技能,每个不同的技能对应的生产率也是不一样的。这里将一个人员的生产率描述为一个集合 ProductivitySet,具体表述为如下形式:

$$ProductivitySet = \{(skill_1, productivity_1), (skill_2, productivity_2), \cdots, (skill_i, productivity_i), \cdots, (skill_k, productivity_k)\}$$

其中:$(skill_i, productivity_i)$ 表示一条生产率信息;$skill_i$ 表示该人员的第 i 个技能;$productivity_i$ 表示人员对应于这条技能的生产率数值;k 表示该人员具有 k 种技能。

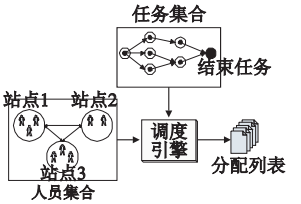

图1 GSD环境下进行任务调度的框架图

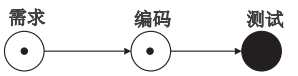

图2 任务图示例

e)salary 表示该人员的薪水,这里使用时薪表示,用于计算完成一个任务需花费的开发成本。

f)timezone 为该人员所在地方的时区,用于描述不同人员的时区差异。

g)location 为该人员所在站点的地理位置,用于描述不同人员的地域差异。

h)country 为该人员所在的国家,不同的国家具有不同的文化,所以可以通过国家信息来描述文化的差异。

i)ActRoles 表示该人员所能担任的角色,如一个人可能同时担任多个角色,如项目经理、开发人员等。

j)available 表示该人员是否空闲,当该人员去执行一个任务的时候,available 为假,空闲时为真。

另外,虽然一个地理位置(location)即可确定其时区信息(timezone)以及判断所属于的国家(country),但是为了计算方便,依然加入 timezone 字段用来计算不同人员之间的时区差异,加入 country 字段用来选择一个站点对应的文化模型,从而计算他们的文化差异。

3.3 调度引擎

下面先介绍在调度时需要采用的优先级函数,然后介绍调度引擎在调度时整个算法的流程。

3.3.1 优先级函数

优先级函数用于计算一个调度对(Task,Employee)的优先值,是调度算法的核心。该函数表示如下:

$$Priority(Task, Employee) = f(Task, Employee) + g(Task, Employee)$$

本文将全球软件开发中的人员交流看做分布式环境下多处理器的一种通信情况,并参考了一种有限连接的多处理器系

统上的任务分配方法[12]。优先级函数由两部分组成:

a) f(Task,Employee) = Prepriority(Task)/maxprepriority

f 函数用来表示任务需要被提前完成的可能性,f 值越大表明该任务在项目中需要被提前完成的可能性越高。其中:Prepriority(Task)表示任务在任务图中对应的节点距离任务图中终结点的最长路径的路径长度,maxprepriority 表示所有任务中最大的 Prepriority 值。对于独立的任务(即该任务不依赖于任何任务且没有任务依赖于该任务),可以单独执行甚至在项目最开始就执行,因此可以设置为最高优先级。

b) g(Task,Employee) = CutCost(Task,Employee)/(CutCost(Task,Employee) + ProdCost(Task,Employee))

该函数用来表示交流成本与开发成本的比值情况,如果该值越大,则花费同等程度的开发成本,需要较少的交流成本。其中:ProdCost(Task,Employee)表示 Employee 完成任务所花费的开发成本,可使用如下方式进行计算

ProdCost(Task,Employee) = (Task. size/productivity) × Employee. salary

其中:productivity 是该 Employee 具有执行 Task 技能的生产率。CutCost(Task,Employee)用来计算如果由 Employee 来执行任务 Task 而不由其他人去执行所产生的成本减少值。对于一个调度对(Task$_i$,Employee$_j$),CutCost 函数可按如下方式进行计算:

$$\mathrm{CutCost}(\mathrm{Task}_i,\mathrm{Employee}_j)=\sum_{k=1}^{r}(mco(i,i_k)-co(i,i_k,j))$$

上式表示 Task$_i$ 有 r 个前置任务,Employee$_j$ 在执行任务 Task$_i$ 的时候需要跟所有完成 Task$_i$ 的前置任务的人员进行交流。设 Task$_{ik}$ 为其中一个前置任务,且完成该任务的人员为 Employee$_{jk}$,则 Employee$_j$ 与 Employee$_{jk}$ 的交流成本可使用前面提到的 CommCost 函数求得,且 $co(i,i_k,j)$ 可按如下方式计算:

$$co(i,i_k,j)=\mathrm{ProdCost}(\mathrm{Task}_i,\mathrm{Employee}_j)+\mathrm{CommCost}(\mathrm{Employee}_j,\mathrm{Task}_i,\mathrm{Employee}_{jk},\mathrm{Task}_{ik})$$

另外,$mco(i,i_k)$ 表示当前所有具有执行任务 Task$_i$ 能力的 K 个人员中函数 $co(i,i_k,j)$ 的最大值,可表示为

$$mco(i,i_k)=\max\{co(i,i_k,j),1\leqslant j\leqslant K\}$$

3.3.2　调度过程

调度引擎的算法流程如图 3 所示。

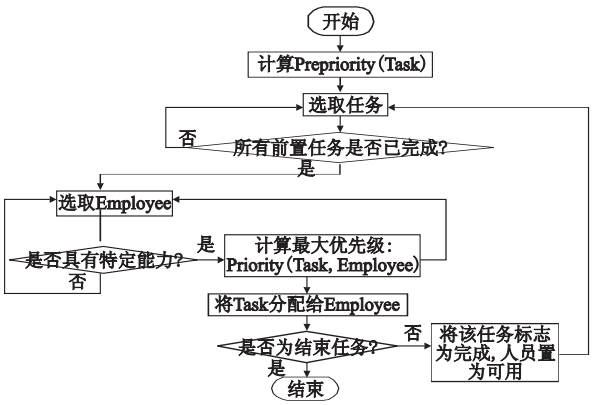

图3　调度算法流程图

具体过程如下:

a)计算所有任务的 Prepriority 值。

b)选取一个未被分配的任务,即该任务 finished 和 assigned 均为 false。

c)如果该任务存在前置任务,判断其所有前置任务是否均已完成,如果没有,回到步骤 b)继续选取任务;如果所有前置任务均完成(即 finished 为 true)或者该任务不存在任何前置任务,进入下一步。

d)从人员集合中选择一个空闲的人员,即该人员 available 为 true。

e)判断该人员是否具有执行当前任务的能力,即具有对应的技能和角色,如果人员不具备相应能力,则回到 d)继续选取;否则,进入下一步。

f)设 c)选取的任务为 Task,e)选取的人员为 Empolyee,使用 Priority 函数计算该调度对(Task,Employee)的优先级;回到 b)继续选取,直到所有任务集合均已遍历一遍,并记录了最大的 Priority 值以及具有该优先值的调度对为(Task$_i$,Employee$_j$)。

g)将 Task$_i$ 分配给 Employee$_j$ 执行,并将 Task$_i$ 的 assigned 置为 true,Employee$_j$ 的 available 置为 false。

h)当 Employee$_j$ 完成任务 Task$_i$,将 Employee$_j$ 的 available 状态置为 true,Task$_i$ 的 finished 状态置为 true;如果 Task$_i$ 为结束任务,即 Task$_i$. terminal = true,则整个项目结束。

4　示例研究

下面通过一个示例项目来演示本文方法的使用过程和效果。示例项目是一个 C/S 架构的信息查询系统开发,主要包括服务器和客户端部分,服务器部分使用 C ++ 语言进行开发,客户端部分使用 Java 语言进行开发。

4.1　任务数据

示例项目包括 12 个任务,分成三种类型:a)需求、设计类任务,这类任务使用 UML 建模语言进行开发,需要具有需求开发(Req)角色的人来完成,包括 Task$_1$ ~ Task$_3$;b)开发类任务,这类任务的开发语言有 C ++ 和 Java,此类任务需要由具有产品开发(Dev)角色的人来完成,包括 Task$_4$ ~ Task$_9$;c)测试、集成类任务,这类任务的类型使用 Test 来标志,需要具有质量保证(QA)角色的人来完成,包括 Task$_{10}$ ~ Task$_{12}$,其中 Task$_{12}$ 标志项目的结束。另外,类型 a)任务规模的单位是设计用例数,类型 b)任务规模的单位是源代码千行数,类型 c)任务规模的单位是测试用例数。所有任务的主要数据如表 1 所示。

表 1　任务集合数据

ID	名称	规模	类型	前置任务	角色
1	需求开发(RD)	200	UML	NULL	Req
2	服务器设计(SD)	150	UML	Task1	Req
3	客户端设计(CD)	180	UML	Task1	Req
4	服务器开发 1(SDEV1)	4	C ++	Task2	Dev
5	服务器开发 2(SDEV2)	5	C ++	Task2	Dev
6	服务器开发 3(SDEV3)	6	C ++	Task2	Dev
7	客户端开发 1(CDEV1)	3	Java	Task3	Dev
8	客户端开发 2(CDEV2)	4	Java	Task3	Dev
9	客户端开发 3(CDEV3)	5	Java	Task3	Dev
10	服务器测试(STE)	100	Test	Task4 Task5 Task6	QA
11	客户端测试(CTE)	120	Test	Task7 Task8 Task9	QA
12	系统集成(SI)	80	Test	Task10 Task11	QA

根据任务之间的依赖关系形成如图 4 的任务图,实心圆为

该项目结束标志。

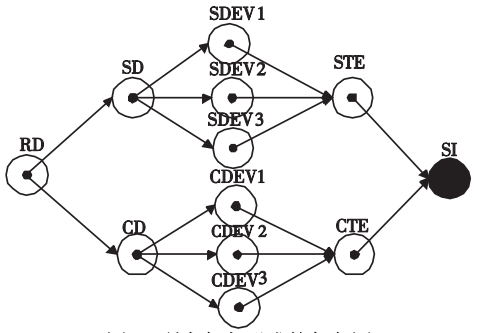

图4　所有任务形成的任务图

4.2　人员数据

本次示例项目中包含三个站点,分别来自美国加利福尼亚州(Site-A)、巴西里约热内卢(Site-B)和中国北京(Site-C)。Site-A 有两个开发人员,位于 GMT-04:00 时区,Site-B 有三个开发人员,位于 GMT-03:00 时区,Site-C 有三个开发人员,位于 GMT+08:00 时区。不同站点的薪水情况是不一样的,同一个站点内的人员薪水也可能不同,薪水统一以时薪计算,单位为美元,美国最高,中国次之,巴西最低。每个人可能具有多个技能或担任不同角色,不同的技能对应不同的生产率。对于需求或设计类型的技能,生产率单位为用例数每小时;开发类型的任务,生产率单位为代码行每小时;测试或集成类型的任务,生产率单位为测试用例数每小时。所有人员的主要数据如表 2 所示。

表 2　人员集合数据

姓名	国家	时区	地域	时薪	生产率	角色
Nanhai	中国	+08:00	北京	8	[Java, 120] [C++, 50]	Dev
Chenhao	中国	+08:00	北京	7	[Java, 100] [C++, 60] [UML, 2] [Test, 3]	Dev Req QA
Sufeng	中国	+08:00	北京	7	[Java, 100] [C++, 80] [Test, 3]	Dev QA
Fernando	巴西	-03:00	里约热内卢	6	[C++,80] [UML,1] [Test, 5]	Req QA
Ronaldo	巴西	-03:00	里约热内卢	5	[Java,100] [Test,4]	Dev QA
Bailly	巴西	-03:00	里约热内卢	5	[C++,80] [UML,1] [Test, 2]	Dev Req QA
Tom	美国	-04:00	加州	12	[Java,120] [C++,100] [UML, 10]	Dev Req
James	美国	-04:00	加州	10	[Java,100] [C++,80] [UML,8]	Dev Req

4.3　基础数据

在调度前,项目经理需要根据实际使用交流工具的情况设置基础数据,主要包括两个成本函数的取值情况:每天维持一个通信连接的成本 $l(a,b)$ 和发送一条通信信息的成本 $m(a,b)$。两个成本函数与站点的基本属性(如时区、地域等因素)、交流时采取的通信方式(如同步、异步等)以及各站点采用的通信工具(如邮件、视频会议系统等)有关。本示例项目的基础数据如表 3 所示,单位为美元,其中,C 表示China(中国),B 表示Brazil(巴西),A 表示America(美国)。

表 3　各成本函数取值情况

(a,b)	la	ls	ma	ms
(A,B)	120	200	10	30
(A,C)	150	250	15	50
(B,C)	100	150	5	20

4.4　调度结果

本文开发了一个实现该方法来处理全球软件开发的任务调度问题的原型工具 GTASys(GSD task allocation system)。将任务集合数据和人员集合数据以及基础数据输入 GTASys,运行该工具得到的调度结果如图 5 所示。

采用这种调度方法,项目最后花费的总成本为 $ 4 080,其中交流成本 $ 590。另外,整个项目开发时间为 21.5 d。

4.5　对比分析

将使用本文方法得到的任务调度结果与传统的基于阶段(phase-based)的任务调度策略进行比较。主要从两方面进行对比,一个是项目进度,一个是项目开发总成本。项目进度可根据最后生成的甘特图进行比较,开发总成本主要由开发成本和交流成本两部分组成。开发成本用于计算员工执行任务的成本,交流成本主要用于不同站点的人员进行交流,这里忽略组内交流的成本。

4.5.1　基于阶段的任务调度方法

在基于阶段的任务调度方法中,将软件项目分成不同的阶段,并将不同阶段的任务分配给不同的站点。比如,国际化的公司经常将一些测试或语言本地化等任务外包给工资水平较低的国家公司或该国的代理机构,从而达到减少成本的目的。

该示例项目按照基于阶段的任务调度方法进行调度,将类型 a)任务分配给美国,类型 b)任务分配给中国,类型 c)任务分配给巴西。每次分配都尽量考虑较优的方案,即对于多个候选的同类型的任务,将规模较大的任务分配给一个空闲的、具有特定能力且对应生产率较高的人。例如“需求开发”和“服务器设计”,两种均为 UML 类型的任务,任务分配到美国站点,且“需求开发”规模较大(200 > 150),当前美国站点中虽然 Tom 和 James 都具备 UML 设计能力且均空闲,但是因为 Tom 在 UML 设计方面具有较高的生产率(10 > 8),所以将“需求开发”分配给 Tom,而将“服务器设计”分配给 James。由于开发类有六个任务,而中国站点只有三个人员,所以将一个人力资源分配给一个任务的时候,另外一个需要该人力资源的任务需要等待前一个任务完成并释放对应资源后,才能得到执行,如图 6 中的“服务器开发 1”和“客户端开发 1”都是由 Chenhao 来完成,两个任务间虽然不存在依赖关系,但是“客户端开发 1”任务需要等待“服务器开发 1”任务完成后才能执行。基于这种调度方法,最后得到的调度结果如图 6 所示。

4.5.2　成本比较和分析

项目开发的总成本包括执行任务的开发成本和由于交流

造成的交流成本。交流成本主要发生在完成存在依赖关系的两个任务的人员之间,如图 5 中的 Chenhao 与 Tom 等和图 6 中的 Tom 与 James 等。根据两种不同的任务分配方案最后计算出的项目成本如表 4 所示。

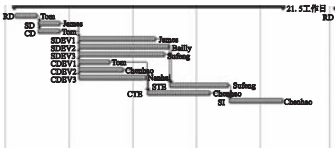

图5　使用本文任务调度方法生成的甘特图

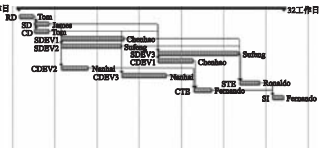

图6　使用按阶段的任务分配方法生成的甘特图

表 4　成本对比表

对比项	基于阶段	基于交流成本
开发成本	$ 3 056	$ 3 490
交流成本	$ 1 530	$ 590
总成本	$ 4 586	$ 4 080
交流成本所占比例	33.4%	14.5%

从表 4 可以看出,虽然使用本文策略造成的开发成本较高,但是交流成本要比传统的方法小很多,而且整个项目的总成本也较低。在全球软件开发中,如果忽略了交流的影响,有可能使整个项目的开发成本提高。而且,从这个表格也可以看出,传统的任务分配策略会使交流成本在整个项目开发中占据较大的比重,如果在一个更大的项目中依然使用这种方法进行任务分配,则可能造成更高的总开发成本。

4.5.3　项目进度比较和分析

通过对比图 5 和 6 可以知道,将传统的按阶段进行任务分配的方法应用到分布式软件开发中还可能造成较长的项目开发时间(32 d > 21.5 d)。虽然这里在使用传统方法分配任务时,每次在每一个站点内都采用较优的策略,但是局部较优的策略并不能带来全局最优的结果,因为这个策略会将多个任务分配到一个站点,而当该站点人力资源比较紧张时,必然出现由于人力资源的使用而造成的等待,从而造成项目开发时间的增加。如图 6,整个项目开发的时间主要花在开发类任务上,这是由于将开发类任务全部分配给中国这个站点,而该站点只有三个人员,所以平均每个人员需要完成两个任务,这样一来,本来不存在依赖关系的任务之间形成了由于人力资源依赖而造成的间接依赖关系。例如 Chenhao 负责完成两个任务:"服务器开发 1"和"客户端开发 1",本来这两个任务并不存在依赖关系,但是"客户端开发 1"需要等待 Chenhao 完成"服务器开发 1"这个任务后,才能得到执行。而这个时候,仍然有位于其他站点的人员处于空闲状态,但该方法并没有对这些人员进行调度。

使用本文采取的方法从全局进行考虑,对空闲的且具有相应能力的人员进行调度,从而充分提高了人力资源利用率,减少任务间由于资源使用而造成的间接依赖,保证项目完成的进度。

5　结束语

本文提出了一种在全球软件开发环境下基于交流成本的任务调度策略,从任务间的依赖关系出发,在每次调度时不仅考虑人员具有的能力以及可用性,还综合考虑人员所处站点的各种因素,如时区、地域、语言、文化等,从而尽量减少人员间低效的交流,努力降低由交流造成的高成本并提高人力资源利用率。随着全球化合作的加深,全球软件开发也不断流行,项目经理在进行任务分配前可以使用该方法对任务进行一次模拟分配,从生成的任务分配结果或项目计划考虑方案的可行性,从而在项目初期就能为项目经理提供较好的决策支持。

未来工作主要包括两个方向:一是如何提供一种合理的方法能让项目经理更好地对任务进行划分,从而得到比较合适的任务集,以方便后面的调度;二是从全球化软件开发的公司中采集更多的数据,对调度模型中的评价函数进行优化,从而得到更佳的调度模型,使得到的任务调度结果更有实用价值。

参考文献:

[1] SETAMANIT S, WAKELAND W, RAFFO D. Using simulation to evaluate global software development task allocation strategies: research sections [J]. Software Process: Improvement and Practice, 2007, 12(15): 491-503.

[2] SETAMANIT S, RAFFO D. Identifying key success factors for globally distributed software development project using simulation: a case study[C]//Proc of International Conference on Making Globally Distributed Software Development a Success Story. Berlin, Heidelberg: Springer-Verlag, 2008: 320-332.

[3] RAFFO D, SETAMANIT S. A simulation model for global software development project[C]//Proc of International Workshop on Software Process Simulation and Modeling. 2005.

[4] MOCKUS A, WEISS D M. Globalization by chunking: a quantitative approach [J]. IEEE Software, 2001, 18(2): 30-37.

[5] ESPINOSA J A, CARMEL E. Modeling coordination costs due to time separation in global software teams [C]//Proc of International Conference on Software Engineering. 2003.

[6] JALOTE P, JAIN G. Assigning tasks in a 24-hour software development project[C]//Proc of the 11th Asia-Pacific Software Engineering Conference. Washington DC: IEEE Computer Society, 2004: 309-315.

[7] LAMERSDORF A, MUNCH J, ROMBACH D. A decision model for supporting task allocation processes in global software development [M]//Product-Focused Software Process Improvement. Berlin Heidelberg: Springer-Verlag, 2009: 332-346.

[8] RAMASUBBU N, MEHRA A, MOOKERJEE V. Modeling coordination in offshore software development [C]//Proc of PACIS2009. 2009.

[9] ESPINOSA J A, CARMEL E. The effect of time separation on coordination costs in global software teams a dyad model [C]//Proc of the 37th Annual Hawaii International Conference on System Sciences. 2004.

[10] HOFSTEDE G. Culture's consequences: international differences in work-related values [M]. CA: Sage, Beverly Hills, 1980.

[11] BORCHERS G. The software engineering impacts of cultural factors on multi-cultural software development teams [C]//Proc of International Conference on Software Engineering. Washington DC: IEEE Computer Society, 2003: 540-545.

[12] KOBAYASHI S. Proposal and evaluation of task allocation method for limited connected multiprocessor system [J]. Systems and computers in Japan, 1997, 28(4): 58-67.