软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(9):2049−2058 [doi: 10.3724/SP.J.1001.2011.03879]

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

基于多代理协作的 IT 复杂应用管理任务分解算法*

高 斐+, 邱雪松, 高志鹏, 孟洛明

(北京邮电大学 网络与交换技术国家重点实验室,北京 100876)

Task Decomposition Algorithm for IT Complex Application Management Based on Multi-Agent Collaboration

GAO Fei+, QIU Xue-Song, GAO Zhi-Peng, MENG Luo-Ming

(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

+ Corresponding author: E-mail: pheonixgao@gmail.com

Gao F, Qiu XS, Gao ZP, Meng LM. Task decomposition algorithm for IT complex application management based on multi-agent collaboration. *Journal of Software*, 2011,22(9):2049−2058. http://www.jos.org.cn/1000-9825/3879.htm

Abstract: This paper presents a dynamic hieratical task decomposition algorithm which applies for a management module of complex IT application based on multi-agent collaboration. The algorithm considers the capacity restriction of multi-agent and the dynamicity of management task caused by the variation of management strategy, IT infrastructure and service logic. Meanwhile, it also considers the balance issue of sub tasks after decomposition, which is the load balance issue of the corresponding multi-agent. The algorithm effectively improves the task executing efficiency and stability of multi-agent. Simulation and analysis results show that the algorithm in this paper is more efficient and has steadier load distribution than that of other compared algorithms.

Key words: multi-agent; complex IT application; task decomposition; capacity restriction of multi-agent; load balance factor

摘 要: 基于多代理协作的 IT 复杂应用管理模型,给出了用于 IT 复杂应用管理的代理能力模型及管理任务分解问题的基本原则,进而提出了一种动态多角色的管理任务层级分解算法.算法考虑到多代理的能力限制,以及由管理端策略、IT 基础设施或业务逻辑改变而引起的 IT 复杂应用管理任务的动态性.同时,算法兼顾了分解后子任务的平衡性,即执行子任务的多代理的负载平衡性,能够有效地提高多代理的执行效率和稳定性.仿真结果显示,由该算法所分解的子任务集呈现出了良好的执行效率及稳定的负载平衡性.

关键词: 多代理系统;IT 复杂应用管理;任务分解;多代理能力限制;负载平衡度

中图法分类号: TP311 **文献标识码**: A

IT 应用(如数据库应用、中间件应用和各类特定业务应用等)是 IT 系统的核心内容.完备且高效的 IT 应用

* 基金项目: 国家自然科学基金(60821001, 60973108, 60902050); 国家重点基础研究发展计划(973)(2007CB310703); 国家高技术研究发展计划(863)(2008AA01Z201)

收稿时间: 2010-01-25; 定稿时间: 2010-04-28

管理能够更好地保障IT系统的稳定运行,是IT管理的重点.随着IT系统规模的日益扩大,IT应用呈现出动态性、分布性、异构性等特点(本文称其为 IT 复杂应用).其管理任务中包含对众多物理和逻辑单元的管理,是大量管理活动的集合,且随业务需求等因素动态变化.传统的静态管理方式已经很难适应纷繁复杂、动态多变的 IT 复杂应用管理需求.目前,对于IT复杂应用管理的常用管理方案是基于多代理(multi-agent,简称 MA)[1]协作的管理方案,即根据当前管理任务的特点即时生成 MA,协作完成管理任务.多代理协作的管理方案可以动态灵活地适应于分布性和动态性强的 IT 复杂应用的管理.

基于多代理协作的 IT 复杂应用管理中,在管理功能完整性可保障的前提下,管理任务执行效率将是重点需要考虑的因素.因而,如何能够动态高效地调度IT复杂应用管理任务的协作执行是关键问题.管理任务分解是基于多代理管理任务调度中首先需要解决的问题.管理任务的分解结果将直接影响到管理任务的执行效率、完整性以及稳定性.因此,研究管理任务分解算法具有重要意义.

在 IT 复杂应用管理任务分解的研究中,首要目标是分解后子任务集合执行效率尽可能高,同时还需兼顾:

1) 低干扰性,要求分解后的执行子任务的 MA 所占用的被管对象的资源尽可能地不影响到被管对象的正常运行性能,通常设置 MA 能力上限,MA 资源消耗处于该能力上限内时不会造成被管对象的运行性能的下降,因而任务分解的粒度是受限于 MA 能力;
2) 动态性,IT 复杂应用的管理任务由管理端根据当前管理策略动态生成,这就要求管理任务的分解能够动态适应管理任务的变化;
3) 平衡性,要求分解后执行子任务的 MA 工作负载处于平衡状态,非平衡状态的 MA 将会造成一定程度的资源浪费并且有几率影响 IT 复杂应用的稳定运行.

目前,适应 IT 复杂应用管理任务特点的分解算法的研究仍然是一个难题.

本文针对 IT 复杂应用的管理需求和管理任务的特点,提出一种基于多代理协作的动态任务分解算法.算法考虑到 MA 的能力限制,能够动态适应 IT 复杂应用管理任务由管理端策略、IT 基础设施或业务逻辑改变而引起的变化.同时,本算法兼顾分解后子任务的平衡性,即执行子任务的 MA 的负载平衡性,能够有效地提高 MA 的执行效率和稳定性.

本文第 1 节介绍近年来的研究成果.第 2 节给出基于 MAS 的 IT 复杂应用管理系统模型以及 IT 复杂应用管理任务的描述.第 3 节介绍提出的基于多代理协作的动态任务分解算法.第 4 节介绍本文算法的仿真与分析结果.第 5 节给出结论和后续工作.

1 相关工作

对 IT 复杂应用管理框架和方法的研究,文献[2]提出一种基于多代理协作的通用框架用于 IT 应用自主管理,但该文献没有涉及复杂管理任务的分解和调度,其实际的管理效率和可操作性还存在进一步优化的可能.

对于多代理协作环境下的任务分配和调度问题,目前已在相关领域有许多探索.文献[3]提出了基于多代理联盟的任务分配的分类方法,按照任务需求、资源限制和收益目标 3 个参数将任务分配划分为 5 类,并对每类问题提出了适用的算法.文献[4]提出一种改进型的遗传算法解决多代理协作的任务分配,该算法改进了基因编码和解码时间,能够获得全局最优解.对于执行任务的多代理联盟生成的研究,文献[5]给出了一种对 Agent 联盟快速动态生成的算法,在基于 Agent 合作收益独立性假设的基础上,可以对联盟结构图进行化简,化简后的搜索量减少,联盟生成效率得到改进.上述任务分配算法均假设复杂任务已经完成初步分解,然而复杂应用的管理任务本身的特性要求管理任务的分解和分配不能破坏管理功能的完整性和准确性.因而,复杂应用的管理任务分配和调度首先需要适应复杂应用管理任务特性的分解算法的支持.

在基于多代理协作的任务分解方面的研究中,文献[6]提出一种基于智能规划任务分解和分配的在线算法,运行时由执行任务的代理推理决定完成自身能完成的子任务,并将不能完成的子任务委派给其他代理.文献[7]结合多代理并行执行的特点和网络管理任务本身特点提出一种基于任务依赖关系的分解算法.然而,上述文献中均没有考虑到 MA 的能力限制及分解后子任务的平衡性,不适用于 IT 复杂应用的管理任务分解.

2 系统模型和管理任务描述

2.1 基于MAS的IT复杂应用管理系统模型

通常,基于 MAS 的 IT 复杂应用管理系统模型[2]自下而上分为被管网元(managed element,简称 ME)层、MA 层和网络管理系统(network management system,简称 NMS)层.如图 1 所示,图 1(a)表示管理系统无管理任务的待机状态,图 1(b)表示 NMS 下发管理任务后的运行状态.MA 层中的虚线圈表示一个 MA,被管网元层中的云表示一个 IT 复杂应用型的 ME.

用于 IT 复杂应用管理的 MA 需满足两个特性:

第一,所有代理要共享通用的被管对象的描述信息;

第二,至少存在一个代理与 ME 之间要存在通用的耦合关系,即代理可通过某种手段访问 ME.

本文针对此两个特性,采用层级式 MA[8],由领袖代理(leader agent,简称 LA)和运行时动态创建的工作代理(worker agent,简称 WA)组成.

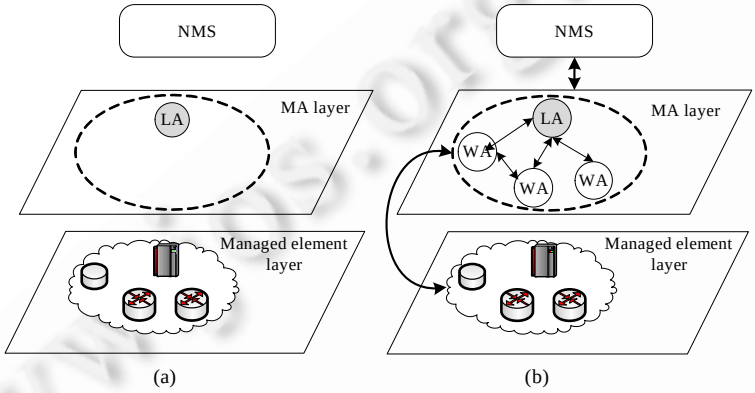


Fig.1 IT application management model based on multi-agent collaboration

图 1 基于 MA 协作的 IT 应用管理模型

在无管理任务时处于等待阶段,MA 环境中只存在 LA,如图 1(a)所示.其中,LA 主要负责与 NMS 通信,与对应的 ME 建立耦合.LA 不参与任务的执行,只负责管理和控制 WA 的生成和运行,是一种行使工厂功能的代理,如图 2 中 LA 的结构所示.本文所提出的任务分解算法也是在 LA 中实现的.当 LA 接收到 NMS 发送的管理任务后,将会依照当前 MA 的能力,对管理任务进行分解,形成可供单一 WA 执行的子任务;随后,按照任务的动态生成具有特定管理角色的 WA,形成协作的 WA 群组,如图 1(b)所示.WA 是执行子任务的工作代理,与 LA 建立通信连接获取控制信息并上报任务执行信息;同时,WA 也需要和有依赖关系的 WA 通信,如图 2 中 WA 结构所示.

2.2 IT复杂应用管理任务描述

通常,IT 复杂应用的管理任务内包含具有不同管理功能的大量管理活动,且不同管理活动可能具有不同的管理功能,需具有不同管理角色的执行代理 WA.如“Oracle 应用”管理任务中的性能监控类的管理子任务“表空间监控”只能由管理 Oracle 应用的 MA 中的具有“Perf”管理角色的代理来执行.管理任务分解过程中需区分管理活动的不同管理功能.

我们用如图 3 所示的管理任务树来描述复杂管理任务.其中,任务目标节点表示其所含子节点中的管理活动全部完成后的目标,并不表示实际的管理活动.管理任务树的第 1 层的目标节点用来表示管理功能目标,下层的目标节点表示子任务目标.

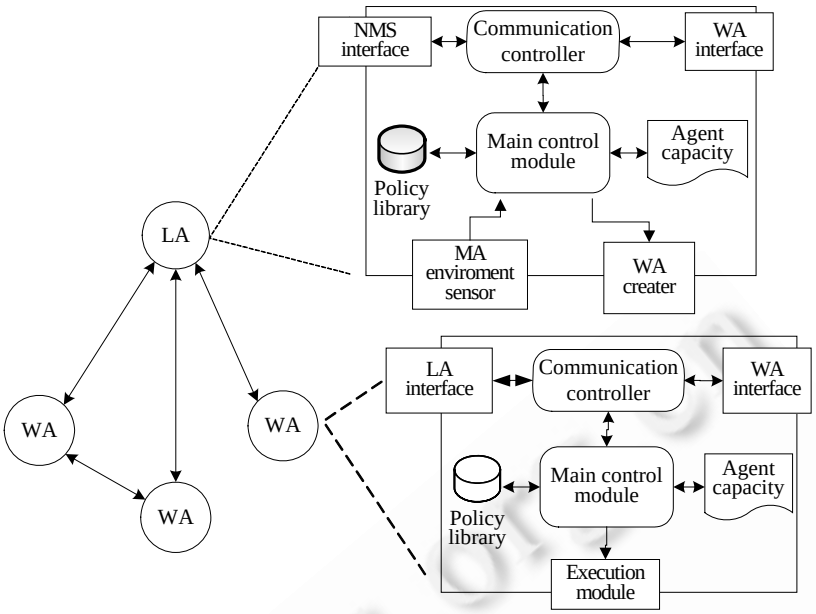


Fig.2 Architecture of LA and WA

图 2 LA 和 WA 的结构

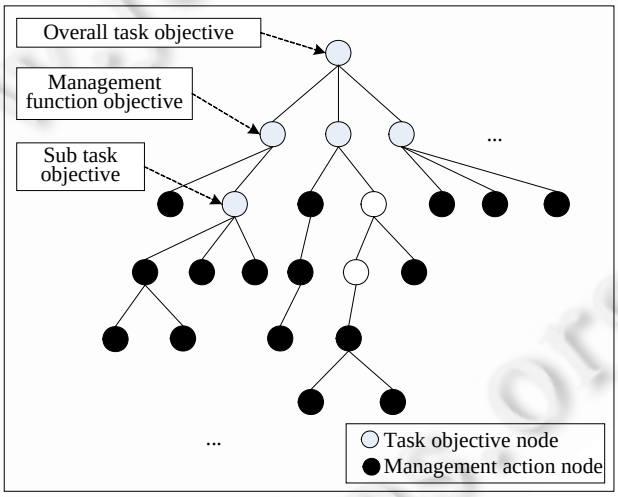


Fig.3 Management task tree of IT complex application

图 3 IT 复杂应用的管理任务树

管理任务的分解过程即是以子任务目标节点为信标,将合适粒度的管理活动集合分离形成子任务[1,9].

其中,子任务可能是一个简单任务或多个管理活动的集合.子任务中管理活动的数量及其依赖关系体现了复杂管理任务的复杂性.在介绍本文所提出的分解算法前,我们先给出如下几个关于复杂任务的定义.

定义 1(管理任务的复杂因子). 我们用 *Ct* 来表示管理任务的复杂因子,*Ct* 包含由管理活动数造成的复杂性因子 *Ca* 与由管理活动依赖关系造成的复杂性因子 *Cd*.

其中,管理活动依赖关系造成的复杂因子 *Ct* 主要由被管 IT 复杂应用的体系架构的层次深度决定,反映在管理任务树上即为任务树深度大.对于企业级的大型 IT 复杂应用一般是分布在 IT 环境中不同的物理设备上,且规

模较大,相对于其管理活动数量,各管理活动的依赖关系对 Ct 的贡献较不明显.在管理精度要求不高的情形下,可近似令 $Ct \approx Ca$.

定义 2(任务分解空间). 任务分解空间是描述复杂任务及其分解子任务的全体线性空间,用$I^{(n)}(T)$来表示.

我们用任务分解空间来描述复杂任务的分解状态,其中,n 表示任务分解的的迭代次数.随任务分解的进行,$I^{(n)}(T)$具有不同的状态.

3 基于多代理协作的动态任务分解算法

基于多代理协作的动态任务分解算法的目标是,根据IT复杂应用管理任务的特点,在有限的代理能力限定条件下,动态求解能够兼顾各子任务负载平衡性[10]并且任务执行效率最高的分解方案,其输出是可供MA中单一代理执行的子任务集合.

代理的能力是检验子任务分解合理性的必要条件,我们首先给出执行子任务的工作代理的能力模型.同时,由于IT复杂应用管理任务的本身特性,任务分解中还需考虑到除代理能力之外的一些边界条件和约束规则.

3.1 代理的能力模型

LA 的能力不影响任务分配的结果和执行效率,因而任务分配过程中需重点考虑执行任务的 WA 的能力 Cap.Cap 包含3个方面的因素:WA的执行效率 U,即执行管理任务中原子性的管理活动所需时间;WA执行任务最大负载 MT;WA执行任务的当前负载 L.Cap 可用三元组 $Cap(U,MT,L)$来表示.

对于每个WA,其执行效率 U 由WA本身的内部构架和WA所拥有的系统资源大小所确定,U 决定了任务分解后的子任务执行效率.WA 所拥有的系统资源是 LA 运行时的动态分配,对于具有不同管理功能的管理活动, LA为相应的WA分配不同的资源,以确保不同管理角色的WA具有相似的执行效率.因而,保证WA群负载平衡即可确保WA群的执行效率处于较优的水平.

WA的最大负载 MT 由NMS端所确定的策略参数、综合考虑IT复杂应用的管理任务特点以及MAS的环境的系统资源所确定.系统运行前可在NMS端调节.

WA执行任务的当前负载 L 与其所执行的子任务的复杂因子 Ct_j 相关,待执行的子任务 Ct_j 越高,L 也相应越高.若 L 过低,则分解所得子任务 Ct_j 过低(即任务分解粒度过大),WA资源存在浪费;若 L 过高,则分解所得子任务 Ct_j 过高(即任务分解粒度过小),WA资源不足以正常完成任务.

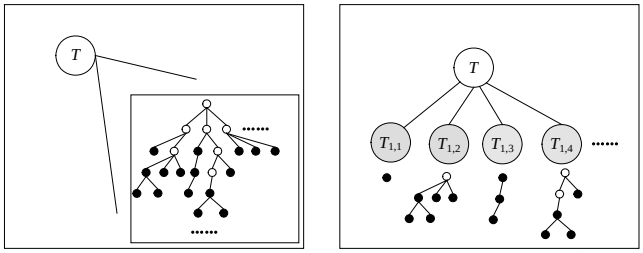
3.2 管理任务分解约束条件

结合复杂IT应用管理任务的特性,管理任务分解过程中需要遵循以下几个约束条件:

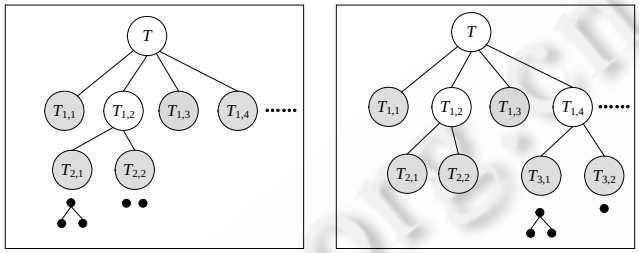
1) 基于代理的独立性:分解后的任务由代理执行时,可以不依赖于其他任务的性质.它一般是结构功能及时序意义上的独立.任务的独立性原则是任务并行执行的基础,也有利于减少任务执行代理之间的协调和通信工作量,提高完成任务的绩效;
2) 任务完整性:任务分解要满足完整性,即每项复杂任务分解后的简单任务的集合应当满足 $T=\bigcup_{i=1}^{n} T_i$;
3) 柔性原则:任务分解得到的结构所具有的在数量、类型等方面的可扩展性.任务分解的柔性便于动态不确定环境下任务的增删和其他局部调整.

3.3 管理任务分解算法

基于MAS的IT复杂应用管理的管理任务可按照如图4所示的层次分解方式迭代分解.图4中,T 表示复杂管理任务,$T_{i,j}$ 表示任务分解过程中出现的第 i 层的第 j 个子任务,灰色子任务表示当前迭代次数分解所得的子任务,即需要由MA中单一WA完成的任务.



(a) Initial state (b) First iteration
(a) 初始状态 (b) 第 1 次迭代



(c) Second iteration (d) Third iteration
(c) 第 2 次迭代 (d) 第 3 次迭代

Fig.4 Example chart of task decomposition

图 4 任务分解示例图

考虑到管理任务分解的原则,任务分解的过程中,每一步都需要在不违反任务分解原则的前提下求解到负载平衡性最好的单步分解方案,其中包括完整性的检验、任务独立性检验以及负载平衡检验,从而得到最终子任务的负载平衡性最好的分解方案.

具体的算法流程图如 5 所示.算法分为如下几个步骤:

1) 任务分解

任务分解开始前,首先按照管理活动所具有的管理功能进行粗分解,形成基于不同管理功能的任务子树集.随后,判断待分解任务是否可分解,即其任务树的根节点是否是目标节点:若不可分,则结束任务分解;若可分则将任务树中复杂因子高于平均复杂因子的任务支树分离.

分离支树的步骤是:检索当前待分解任务中的每个二层节点,计算每个分支子树所含的管理活动数,即任务复杂度 Ct_j.随后,选取 $Ct_j \geqslant Ct/q_i$ 的支树依次分离,其中,q_i 表示当前任务空间 $I^{(i)}(T)$ 中的子任务数,Ct 表示管理任务总复杂度.

此时,若当前子任务数 q_i 超过 MAS 允许的最大 WA 数 $n_{\max}$,则将子任务中 Ct_j 从小依次排列的 $q_i-n_{\max}$ 个子任务聚合.

2) 重建任务空间

根据步骤 1)的分解结果,重新建立当前任务空间,经步骤 1)分解后的任务空间是一系列任务子树(即子任务)的集合.重建过程中,需确保本级任务分解中的子任务完整,即检验当前任务空间 $I^{(i)}(T)$ 内的子任务集满足:

$$\bigcup_{j=1}^{m} T_{i,j} = T.$$

3) 生成所需 WA 群组

LA 根据当前任务空间 $I^{(i)}(T)$ 的子任务的数量生成对应的 WA 群组,并根据子任务的管理功能赋予 WA 相应的管理角色.

4) 子任务的独立性检验

依次验证分解所得子任务是否都可以在对应 WA 的能力范围内完成,即依次验证当前任务空间中的各子

任务是否会造成对应的 WA 的当前负载 $L_T^{(i)}$ 超出最大负载 MT.若不超出,则转步骤 5);否则,选取超出 WA 能力的子任务返回步骤 1)迭代分解.

5) 评估子任务的平衡性

检查执行子任务的 WA 群组的负载是否平衡.若平衡,则转步骤 6);否则,选取负载最大的子任务返回步骤 1)迭代分解.

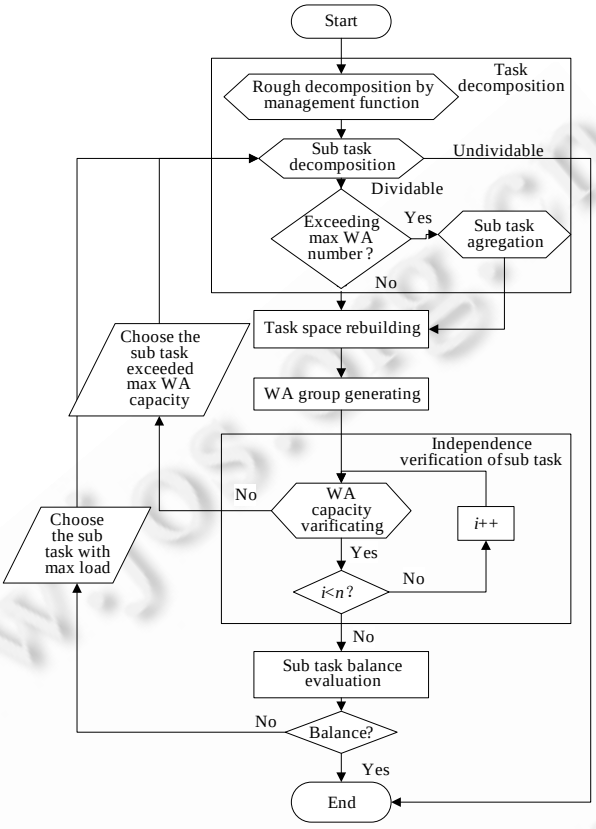


Fig.5 Flow chart of dynamic task decomposition algorithm for IT complex application management

图 5 IT 复杂应用管理任务动态分解算法流程图

WA 群组负载平衡性检验是本分解算法的重要步骤.在一次任务分解完成后,分别对任务分解空间 $\Gamma^{(i)}(T)$ 的子任务 $T_{i,j}$ 进行平衡性的评估,即执行 $T_{i,j}$ 的 WA_j 负载.定义负载影响因子向量 $\lambda=(\lambda_1,\lambda_2,\ldots,\lambda_m)$ 来表示 WA_i 的负载对 MA 的影响重要程度,其中,$\sum_{j=1}^{m}\lambda_j=1$,则 MA 的负载可以表示为公式(1)的形式:

$$L_T=\lambda_1\cdot L(WA_1)+\lambda_2\cdot L(WA_2)+\ldots+\lambda_n\cdot L(WA_n) \tag{1}$$

随后,根据公式(1)计算所得的当前 MA 负载 $L_T^{(i)}$ 考察负载平衡性,其中,i 表示当前第 i 次迭代分解后的 MA 负载.首先定义一个负载平衡系数,表示 MA 中各 WA 的负载的平衡度,记为 G_T,见公式(2):

$$G_T=\sum_{i=1}^{n}\lambda_i\cdot[L_T-L(WA_i)]^2 \tag{2}$$

其中,G_T 越小,表示 WA 群组的负载平衡性越好.当 G_T=0 时,则 WA 群组中的负载绝对平均.

随后,我们定义一个 MA 负载平均饱和度,表示 MA 中 WA 群组的平均负载状况,记为 η_T,见公式(3):

$$\eta_T=[\alpha\cdot L_0+\beta\cdot(L_T-L_0)]/L_0 \tag{3}$$

其中,L_0 是 MA 的初始负载,α,β 是与 WA 的最大负载 MT 以及初始负载 L_0 的值相关的系数.若 η_T=1,则说明 MA

的负载处于饱和状态;若η_T<1,则说明系统处于欠负载状态,目前无需增加新的WA;若η_T>1,则说明MA处于过负载状态,需要增加新的WA,否则WA的执行效率U将会开始下降.对于确定的系统和IT复杂应用,α,β是确定的常量,α,β可由公式(4)确定.其中,η_0是依据具体IT复杂应用的特点所确定的MA系统的负载平均饱和度初值,1−η_0可反映负载平均饱和度的动态范围.

$$\begin{cases}[\alpha\cdot L_0+\beta\cdot(1-L_0)]/L_0=1\\ [\alpha\cdot L_0+\beta\cdot(0-L_0)]/L_0=\eta_0\end{cases} \tag{4}$$

6) 分解终止

任务分解完成,输出结果是由MA中单一WA执行的子任务集合.

4 仿真与分析

4.1 仿真实验建立

为了分析管理任务分解算法的性能,我们建立一个管理任务产生器产生随机的管理任务.

管理任务的随机性主要体现在任务内包含的管理活动的数量及管理任务树的深度.这里,我们考虑到IT复杂应用的管理任务特点,对于管理任务产生器产生的管理任务,我们限定任务树的深度小于等于10,设管理任务树的深度为k(1≤k≤10).

我们以管理任务产生器产生的随机管理任务为输入,对分解算法的分解过程进行仿真.为方便分析分解算法的性能,仿真实验中我们做以下约定以及量化确定分解算法运行的初始参数值:

1) 任务复杂性因子Ct中仅考虑由管理活动数量引起的复杂性因子,即Ct_j=Ca;
2) 单一管理活动执行时间为10ms,MA中允许最大WA数$n_{\max}$=20,单一WA的最大负载MT=5(最多接受5个管理活动的负载量);
3) MA的负载影响因子向量λ=(λ_1,λ_2,…,λ_n)中,设λ_1=λ_2=…=λ_n,为使实验结果较明显,便于比较分析,我们假设MA的初始负载较高,设初始负载L_0=0.5(即2.5个管理活动的负载量),实际系统中通常初始负载L_0都相对较低.同时设η_0=0,由公式(4)可以得到平衡影响因子α,β取值:α=1/2,β=1/2.

4.2 分解算法性能分析

我们通过管理任务产生器分别生成Ca=30,Ca=60,Ca=90的3个管理,按照本文的分解算法进行任务分解.随着分解算法迭代次数的增长,MA中WA的数量增长曲线如图6所示,MA的负载平均饱和度变化曲线如图7所示.

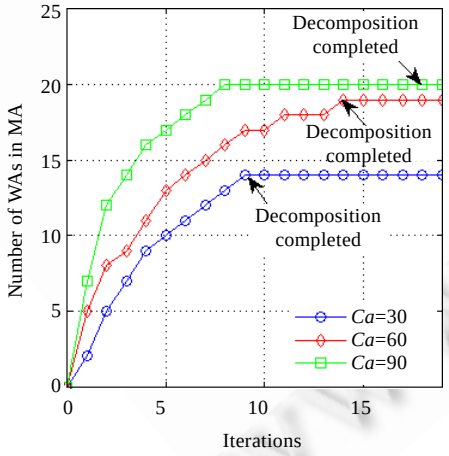


Fig.6 Number of WAs vs. iterations of task decomposition

图6 WA数量随任务分解迭代次数变化曲线

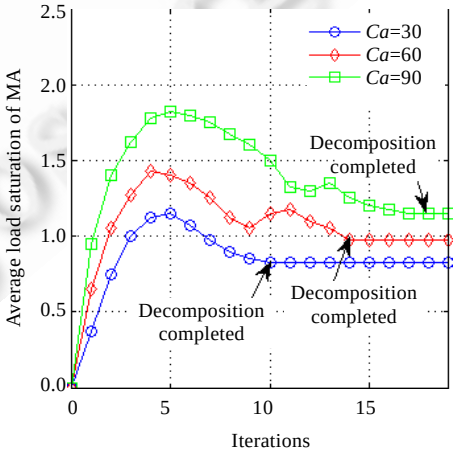


Fig.7 Average load saturation of MA vs. iterations of task decomposition

图7 MA负载平均饱和度随分解迭代次数变化曲线

对于 Ca=30 的管理任务,在第 10 次迭代完成分解,其分解后的 WA 数量稳定在 14 个,MA 负载平均饱和度稳定在 0.7~0.8 之间,MA 处于轻微欠负载状态.对于 Ca=60 的管理任务,在第 14 次迭代完成分解,其分解后的 WA 数量稳定在 19 个,MA 负载平均饱和度稳定在 0.9~1.0 之间,此时,MA 处于满负载状态.对于 Ca=90 的管理任务,在第 19 次迭代完成分解,其分解后的 WA 数量在第 8 次迭代后达到 n_{max};随后的迭代中,WA 数量不再增加,开始汇聚 Ca_j 低的子任务.MA 的负载平均饱和度稳定在 1.2~1.3 之间,MA 处于过负载状态.

总体上观察,本文的管理任务分解算法完成分解的所需迭代次数随任务复杂度 Ca 呈近似线性增大,且总体上以较少的迭代次数即可完成管理任务的分解;同时,能够在 Ca 较大的情况下兼顾平衡性,并将 MA 平均饱和度稳定在合理范围内.

4.3 分解算法结果对比与分析

我们分别对管理任务中管理活动数量在0~110的范围内以10为步长来考察本文的任务分解算法的执行效果.经本文的任务分解算法分解后的管理任务执行时间(即本文分解算法所得解的质量)如图8所示,相对于无负载平衡验证的一般分解算法(如文献[7]中给出的算法),在 Ca<100 的范围内,与理论最短时间比较贴近.其中,理论最短时间是不考虑管理活动的依赖关系将 Ca 个管理活动平均由 n_{max} 个 WA 执行,可近似用 Ca/n_{max} 来计算.当任务复杂度过大(Ca>100),由于本文算法考虑到 MA 能力限制,此时 MA 能力趋于饱和,所得解的执行时间开始超过文献[7]的分解算法.如有需要,可通过增大 MA 能力限制来消除.

同时,如图 9 所示,分解后的管理任务对应的 MA 的负载平均饱和度,在 Ca<100 的范围内也能够稳定地保持在 1.0 附近.而为考虑负载平衡性因素的任务分解算法,则呈现出随机的负载饱和度.总体看来,在一般情况下,IT 复杂应用管理活动中,本文的分解算法显示出了良好的负载平衡性.

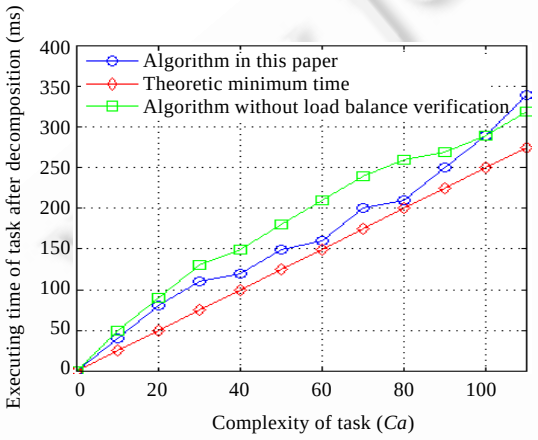


Fig.8 Executing time of task after decomposition vs. complexity of task

图 8 分解后,任务执行时间随任务复杂度变化曲线

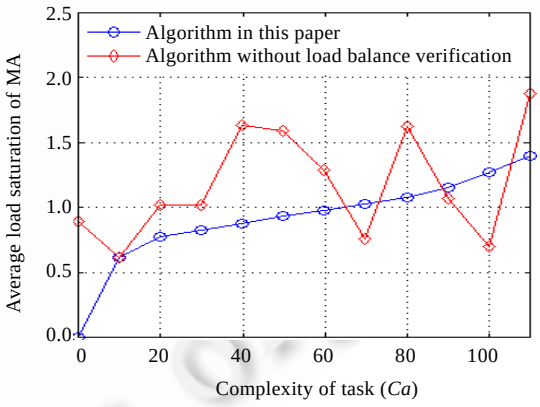


Fig.9 Average load saturation of MA after decomposition vs. complexity of task

图 9 分解后 MA 负载平均饱和度随任务复杂度变化曲线

5 结论与后续工作

本文针对IT复杂应用管理的管理任务分解问题,在基于MAS的应用管理模型之上提出了一种动态多角色的管理任务层级分解算法,并对此任务分解算法进行了仿真和性能分析.本文提出的动态多角色的层级任务分解相对于先前任务分解算法,能更好地适应 IT 复杂应用管理任务的特点,在不同的管理任务中包含的管理活动较大、而管理任务树的深度较小,且执行任务的协作多代理的能力受有限系统资源制约的前提下,维持分解所得子任务执行在较高效的水平;同时,能够兼顾分解所得的子任务的平衡性.经过本算法分解后的子任务集对整

个 MA 的负载平均饱和度,在通常任务量情况下表现出了很好的平稳性.

对于 IT 管理中的复杂应用管理,仅解决管理任务的分解问题还不能完全满足复杂应用管理的管理需求,我们需要进一步考虑任务的分配和调度,任务执行过程中的异常回滚等问题.

目前,研究者越来越开始关注管理系统本身的自配置和优化,如文献[11]就在异构集群的任务调度中引入了调度系统资源自主调节的因素.本文中的代理能力仍然不支持运行时的自适应调节.任务分解算法在进一步考虑到 MA 能力和资源的自配置、自优化等因素的情况下,还有待进一步的研究和优化.

References:

[1] Ferber J, Gutknecht O, Michel F. From agents to organizations: An organizational view of multi-agent systems. In: Proc. of the AOSE 2003. 2003. 214−230.

[2] Lavinal E, Desprats T, Raynaud Y. A generic multi-agent conceptual framework towards self-management. In: Proc. of the NOMS 2006. 2006. 394−403. [doi: 10.1109/NOMS.2006.1687569]

[3] Lau HC, Zhang L. Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity. In: Proc. of the 15th IEEE Conf. on Tools with Artificial Intelligence. 2003. 346−350. [doi: 10.1109/TAI.2003.1250210]

[4] Zhang Y, Li FC. Research on multi-agent dynamic task allocation algorithm and based on dynamic fuzzy set. Acta Electronica Sinica, 2009,37(11):2551−2556 (in Chinese with English abstract).

[5] Zhang XL, Shi CY. A dynamic formation algorithm of multi-agent coalition structure. Journal of Software, 2007,18(3):574−581 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/574.htm [doi: 10.1360/jos180574]

[6] Lopes AL, Betelho LM. Task decomposition and delegation algorithms for coordinating unstructured multi agent systems. In: Proc. of the Complex, Intelligent and Software Intensive Systems 2007. 2007. 209−214. [doi: 10.1109/CISIS.2007.52]

[7] Liu B, Luo JZ, Li W. Task decomposition and scheduling in large-scale network management. Journal on Communication, 2006, 27(3):64−72 (in Chinese with English abstract).

[8] Wooldridge M. An Introduction to Multi Agent Systems. John Wiley & Sons, 2002. 105−126.

[9] Shehory O, Kraus S. Methods for task allocation via agent coalition formation. Artificial Intelligence, 1998,101(1-2):165−200. [doi: 10.1016/S0004-3702(98)00045-9]

[10] Jiang YC, Jiang JC. Contextual resource negotiation-based task allocation and load balancing in complex software systems. IEEE Trans. on Parallel and Distributed Systems, 2009,20(5):641−653.[doi: 10.1109/TPDS.2008.133]

[11] Dutot PF, N'Takpe T, Suter F, Casanova H. Scheduling parallel task graphs on (almost) homogeneous multicluster platforms. IEEE Trans. on Parallel and Distributed Systems, 2009,20(7):940−952. [doi: 10.1109/TPDS.2009.11]

附中文参考文献:

[4] 张瑜,李凡长.基于 DFS 的多 Agent 动态任务分配算法研究.电子学报,2009,37(11):2551−2556.

[5] 张新良,石纯一.多 Agent 联盟结构动态生成算法.软件学报,2007,18(3):574−581. http://www.jos.org.cn/1000-9825/18/574.htm [doi: 10.1360/jos180574]

[7] 刘波,罗军舟,李伟.大规模网络管理中的任务分解与调度.通信学报,2006,27,(03):64−72.

高斐(1982−),男,河南洛阳人,博士,主要研究领域为网络管理,IT 管理.

高志鹏(1980−),男,博士,副教授,CCF 会员,主要研究领域为自然语言处理,网络信息检索.

邱雪松(1973−),男,博士,教授,博士生导师,主要研究领域为网络与业务管理.

孟洛明(1955−),男,教授,博士生导师,主要研究领域为通信网,网络管理,通信软件.