机群系统中基于 PVM 的智能化任务分配和负载平衡模型 [1]

陈锡明 * **　　卢显良 *

*(电子科技大学计算机系　成都　610054)
**(电子对抗国防科技重点实验室　成都　610036)

摘　要　该文提出和建立了一种机群系统中基于 PVM 的智能化任务分配和负载平衡模型，它在传统机群系统任务分配和负载平衡方法的基础上引入知识处理和多代理人 (agent) 系统等人工智能方法来进行知识管理、知识交互、任务协同分配及节点间负载平衡等，具有良好的性能和可伸缩性、可扩充性、自适应性等，是大规模机群系统中任务分配和负载平衡的一种理想方法.

关键词　任务调度，负载平衡，agent(代理人)，知识

中图号　TN919.2

1 引　言

随着互连网络规模的扩大和通信速度的提高，如何更加充分地利用机群系统中的空闲或轻负载工作站来实现大规模并行处理，是一个有着广阔前景和重要意义的课题. PVM (Parallel Virtual Machine，并行虚拟机环境) 是一种相对成熟的基于消息传递的网络并行计算环境，具有通用性强、使用方便等优点. 但其中的任务分配和负载平衡却存在许多不足，在任务分配方面，PVM 采用轮转法 (Round-Robin) 一次性分配任务，因一个节点可能同时属于多个 PVM 虚拟机及其上还有其他用户进程存在等，各节点上的任务分配并不均匀，协作进程间的运行时间差异大、性能降低；PVM3.3 及其以后版本通过加入资源管理器 (RM) 来进行负载平衡，但其依据却是本节点某特定 PVM 虚拟机上的任务数，没有考虑一个节点同时属于多个 PVM 虚拟机和 CPU 及内存利用率等因素，负载平衡效率依旧较低. 本文在对 PVM 的分析和研究基础上，提出并初步实现了一种机群系统中基于 PVM 的智能化任务分配和负载平衡模型——ITALBM(IntelligentTask Allocating and Load Banlancing Model)，以此统一替换节点上所有 PVM 虚拟机的资源管理器 (RM)，通过引入知识处理和分布式人工智能 (DAI) 中多 agent 系统等方法，建立一种智能化的任务分配和负载平衡模型，使各节点能尽量根据自己的负载和所掌握的其他节点的负载等知识高效率地迁出或迁入任务，实现任务的并行分配和节点间负载平衡.

机群系统中的任务调度和负载平衡是实现网络并行计算的一个重要方面，很多学者在其上面做出了杰出的研究. 我国南京大学的谢立等学者在 80 年代末到 90 年代初就提出和建立了分布式智能化任务调度模型 [1]；文献 [2] 提出了一种基于工作站的分布式系统负载分布和平衡支持模型；齐红、鞠九滨等实现了工作站上协作任务的调度 [3]；陈华平等则提出和建立了分布式动态负载调度的一个通用模型 [4]. 本文与以上文献的区别是：(1) 使用的负载指标和知识的构成方式不同；(2) 本系统通过引入分布式人工智能中的多 agent 系统来处理知识的交互和任务的协同分配，具有更强的可扩充性和自适应性.

2 任务和负载

ITALBM 将每个节点的任务分为非 PVM 任务和 PVM 任务两类，后者记为 TK_{PVM}. 因同一节点上不同用户可能有不同的 PVM 虚拟机，TK_{PVM} 又按其用户号 UID 区分为 $TK_{PVM\text{-}UID1}$，

1 1999-10-25 收到，2000-04-09 定稿
国家部级九五预研基金资助

$\mathrm{TK_{PVM\text{-}UID2}}$，$\cdots$，$\mathrm{TK_{PVM\text{-}UID}n}$. 按照这种任务划分法，在进行任务迁移时不但要考虑准备迁入结点传统意义上的资源负载情况，如 CPU 利用率等，还需考虑 $\mathrm{TK_{PVM}}$ 任务的数量和驻留时间，因为一个 $\mathrm{TK_{PVM}}$ 任务可能跟其他节点的同类任务构成协同任务，按照协同调度的思想，多个节点上的协同任务应尽量同时运行，以利于同步和通信 [5] . 多个 $\mathrm{TK_{PVM}}$ 任务驻留在同一节点上不但使各 $\mathrm{TK_{PVM}}$ 任务平均响应时间加长，而且可能造成对网络通信端口的挤占，使协同任务不能同时执行. ITALBM 将每个节点的负载分为两类：资源负载率 RL 和任务负载率 TL . 其中， RL 为 CPU, 内存，通信信道利用率 η_c，η_m，η_n 按照其对应权值 w_c，w_m，$w_n(w_c+w_m+w_n=1)$ 构成的一个值域范围为 [0,1] 的单调增函数，此处取为 $\mathrm{RL}=\eta_c w_c+\eta_m w_m+\eta_n w_n$. TL 为 $\mathrm{TK_{PVM\text{-}UID}}$ 任务的数量及其在该节点上驻留时间的一个单调增函数. 设 $\mathrm{TK_{PVM\text{-}UID1}},\mathrm{TK_{PVM\text{-}UID2}},\cdots,\mathrm{TK_{PVM\text{-}UID}n}$ 的任务个数分别为 $L_1,L_2,\cdots,L_n$，用 $\mathrm{TK'_{PVM\text{-}UID}}j$ 表示 $\mathrm{TK_{PVM\text{-}UID}}j$ 任务队列上的第 i 个任务，函数 $t(x)$ 表示任务 x 在节点上的驻留时间，则 TL 可表示为

$$\mathrm{TL}=\Big(\sum_{i=1}^{L_1}t(\mathrm{TK^{i}_{PVM\text{-}UID1}})+\sum_{i=1}^{L_2}t(\mathrm{TK^{i}_{PVM\text{-}UID2}})+\cdots+\sum_{i=1}^{L_n}t(\mathrm{TK^{i}_{PVM\text{-}UID}}_n)\Big)/(\mathrm{CPU\text{-}MIPS})$$

其中 CPU-MIPS 为本节点 CPU 运算速度，以 MIPS(每秒百万指令) 为单位. ITALBM 中对 RL, TL 分别引入一个输出值为 low, median 和 high(均属于 [0,1]) 三者之一的模糊函数 $f_1(\mathrm{RL})$，$f_2(\mathrm{TL})$ 作为衡量 RL ， TL 高低和进行任务迁移与负载平衡的标准，如表 1 所示.

表 1 ITALBM 中一个结点所处的负载状态及该节点上任务迁入迁出准则

节点负载状态	$f_1(\mathrm{RL})$	$f_2(\mathrm{TL})$	任务迁移准则
轻负载状态	low 或 median	low	可迁入 $\mathrm{TK_{PVM}}$ 任务
重负载状态	high	median	寻找合适节点，将 $\mathrm{TK_{PVM}}$ 任务从本节点迁出
	median 或 high	high	
中负载状态	除以上情况的 $f_1(\mathrm{RL})$, $f_2(\mathrm{TL})$ 组合		既不迁出也不允许迁入 $\mathrm{TK_{PVM}}$ 任务

3 ITALBM 中的知识及知识表达方式

ITALBM 采用分布式任务调度和负载平衡方式. 这种方式中，一个节点关于局部范围内其他节点负载信息的拥有量和准确度是调度成功和接近全局最优的关键. 为此， ITALBM 为每个节点建立了一个包含其他节点负载知识等的知识库，利用非经典逻辑中的时态逻辑和模糊逻辑来建立知识的置信度，使重负载结点能尽量根据自己或委托节点 (见 3.3 节) 知识库中有关其他节点的负载知识直接选取一个轻负载且置信度最大的结点来转移任务；使轻负载结点能主动向其他结点发布负载信息、学习其他节点负载知识并向重负载节点申请迁入任务等.

ITALBM 中的知识分为先天知识、静态负载知识和动态负载知识. 先天知识指机器体系结构、 CPU 速度等节点固有的知识. 静态负载知识指节点周期性或临时性运行过程中的负载规律知识，表 2 列出了某重点实验室 15 台工作站周期性统计负载规律. 临时性负载规律知识是由节点主人出差或其他原因引起的短期内对周期性负载规律知识的更改. 静态负载知识由节点主人填写，通过节点间信息交互传播或直接从知识库管理器中录入. 动态负载知识指节点当前资源负载率、任务负载率等运行过程中获取的有关节点的负载信息. 这三种知识的使用顺序是：若存在关于某节点的动态负载知识，则只使用该知识；反之，则使用静态负载知识. 先天知识在重负载节点选择任务迁移节点时且存在多个满足条件的节点时才使用.

表 2 某重点实验室 15 台工作站 3 个月内统计运行规律知识 (以每星期为单位)

日期	起始时间	结束时间	f_1(RL)	置信度	f_2(TL)	置信度
星期一至星期五	8:00	12:00	high	0.81	low	0.74
星期一至星期五	12:01	14:00	low	0.84	low	0.90
星期一至星期五	14:01	18:00	high	0.78	low	0.78
星期一至星期五	18:01	24:00	low	0.90	low	0.89
星期六至星期日	0:01	24:00	low	0.95	low	0.95

ITALBM 中的知识主要由变量、事实、规则、知识库等知识实体进行描述．变量是知识中各种概念的抽象，如当前时间、当前资源负载率等．对变量赋予一定的值及置信度后便形成事实。规则用来表达静态知识，如：

if(DATE=SUNDAY) and (TIME>22.00) then host1.S-RL=low 0.95; host1.S-TL=low 0.95

其含义为：若时间为星期天 22:00 以后，则节点 host1 的资源和任务负载率均为 low，置信度为 0.95．其中，置信度 cf∈ [0,1]， cf 越大表示肯定程度越强．知识库主要包含三个部分：存储本节点和本节点所知的其他节点动态负载知识的动态负载子知识库、存储本节点和本节点所知的其他节点静态负载知识的静态负载子知识库以及存储本节点和本节点所知的其他节点先天知识的先天知识子知识库。

4 基于多 agent 系统的任务调度和负载平衡模型

agent 是当前 DAI 的热点之一 [6-8]．一般认为， agent 是指在一定环境下持续自主地运行的计算实体，它能作用于自身和环境，并能对环境作出反映，具有自主性、感知性和交互性．agent 能对其目标进行决策，能有选择地采纳别的 agent 的建议对外界信息进行主动或被动的获取和学习，能要求别的 agent 完成一定任务或者拒绝其提出的请求等．通过对 agent 和机群系统任务分配与负载平衡特点及行为的研究，我们认为，采用多 agent 系统能够很好地反映建立在知识处理之上的机群系统中基于 PVM 的任务分配和负载平衡的特点和行为，尤其能够很好地反映节点间知识的交互行为，并显著提高任务分配和负载平衡的效率及自适应性、可伸缩性等。

定义 1 ITALBM 中 agent 是一个可被描述为六元组 (aid， CI， CE， E， T， S) 的计算机程序，其中 aid 为其标识符；CI 为其内部特征，包括各种内部知识和经学习获取来的外部知识；CE 为其外呈现的特征，包括类别、所需要的服务、所能提供的服务以及与外部交互的接口等，其中与外部交互的接口是本 agent 通过预先定义的协议与其他 agent 进行通信，协同完成一定任务的基础。此处， agent 之间的通信协议是一个自行开发的基于 TCP/IP 的专用消息传递语言—— ITALBM-ACL． ITALBM 中简单地将消息的到来称之为事件 E， T 为事件处理过程集合， S 为状态集合，即 agent 感受到事件并调用相应事物处理过程后所处的状态，是对 agent 运行特征的反映．

ITALBM 中 agent 分为管理 agent 和应用 agent，目前均为反应式 agent．前者管理所在域范围内各 agent 的建立、撤消；建立、撤消和维护所在域范围内各 agent 的外部特征，为这些 agent 的协作提供支持；建立、撤消和维护本 agent 的各种内部特征。应用 agent 则完成特定领域内的事物处理．各 agent 之间按照具体的应用构成层次关系或协作关系．在具有协作关系的 agent 中，一个 agent 可以通过 agent 之间的专用通信语言 ITALBM-ACL 请求另一个 agent 为其服务或向其提供服务．按这种结构组成的多 agent 系统如图 1 所示．

各部分功能简介如下：

(1) 机群管理 agent: 整个系统一个．用于管理和监测机群系统中每个节点 agent 的建立、退出、网络失效等；收集和管理每个节点 agent 的外部特征，并将这些外部特征存储在自己的

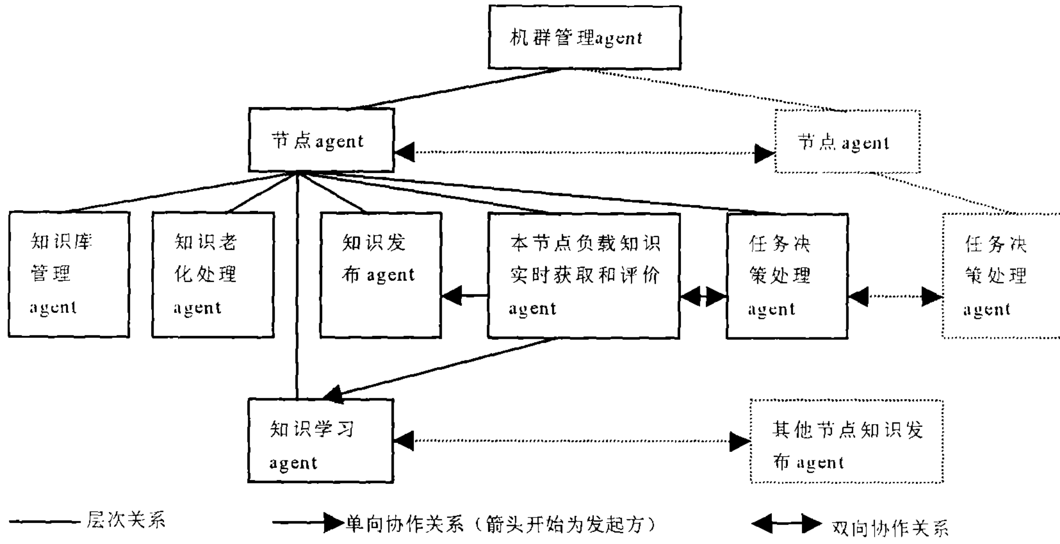

图 1 ITALBM 中 agent 系统组成结构

内部特征中；构造机群系统节点信息表 (host-info-table)(每个节点占一项，供所有节点 agent 继承使用) 等.

(2) 节点 agent: 每个节点一个，继承机群管理 agent 的内部特征，用于建立和保持与机群管理 agent 之间的联系并进行相应信息交互、建立和控制处于其下面层次的各 agent 并协调他们之间的活动等 (其中知识库管理 agent 由用户建立并通知节点 agent，接受后者的协调)．节点 agent 的内部特征还包括：节点知识库、节点所处的负载状态 (如表 1) 以及在共享内存中维持的本节点所知当前机群系统中轻负载和重负载置信度大于某个阈值的几个节点的黑板等．节点 agent 之间具有委托 (commit) 关系，即一个持续处于重负载状态的节点 A，经过一段时间后，A 的动态负载子知识库可能因老化而失去实际意义 (见知识老化处理 agent)，静态负载子知识库则因本节点负载重而没有学习到新知识，这时 A 的节点 agent 可以向其他节点 agent 发出 COMMIT-TO 消息，欲使其成为 A 的委托 agent，以便在任务调度时能使用其知识库 (委托节点应是轻负载节点，在不断地进行知识更新和学习 (见知识学习 agent)，因而其拥有的关于系统中其他节点的知识更丰富并更具实时性)．若存在一个轻负载节点 B，其节点 agent 接到 COMMIT-TO 消息后可以回送 COMMIT-ACK 消息响应而成为 A 的委托 agent．节点 agent 之间的委托关系具有：(a) 多对一映射性，即多个重负载节点 agent 可能拥有同样的一个委托节点 agent，当受委托的节点数大于一定阈值时，该委托节点 agent 成为一个专门提供网络上其他节点知识信息的咨询服务节点 agent; (b) 传递性，即被委托节点 agent 可能由于负载变重而将委托关系转移给新的轻负载节点 agent．

(3) 知识库管理 agent: 提供一个进行先天知识、静态负载知识录入、查询、修改、删除等操作的人机接口，并进行相应的语法、语义校验和规则冲突、一致性检查等.

(4) 知识老化处理 agent：动态负载子知识库中存储的本节点和其他节点的动态负载知识，其置信度都有一个随时间流逝而降低的过程，称之为知识的老化．当其置信度降低到一定阈值以下 (比如 0.25) 时，该知识变成不可用，并从动态负载子知识库中删除 (若在黑板上还需从黑板中删除)．动态负载知识置信度降低公式暂设为： $\mathrm{cf}(t)=\mathrm{cf}(t_0)f(t-t_0)$，其中 $f(t-t_0)$ 暂

取为： $f(t-t_0)=1/[\frac{(t-t_0)}{180}+1]$. t 和 t_0 的单位为秒，其中 $\mathrm{cf}(t_0)$ 为该动态负载知识被获取时的置信度， t_0 为获取时刻， $\mathrm{cf}(t)$ 为 t 时刻置信度， $t-t_0$ 为驻留时间。

(5) 知识学习 agent: 其功能是当节点处于轻负载时对系统中其他节点各种知识的学习，以扩充本节点各子知识库，为任务决策处理 agent 提供任务调度和负载平衡决策支持。学习方法主要有两种： (a) 等待其他节点知识发布 agent 提供； (b) 向周围节点 (设为 B ，本节点设为 A) 发出显式的 REQ-K 消息，请求 B 在可能的情况下提供特定子知识库的知识 (默认值为提供 B 所拥有的各子知识库的知识) 。 ITABLM 中一个节点 i 的周围节点定义为在系统节点信息表 host-info-table 中处于第 i 项前后的各 n 个节点 (n 可调，暂设为 $n=\lfloor\frac{M}{2}\rfloor$ ， M 为节点数) 。 B 的响应方式为：若 B 处于轻负载状态，回复 SEND-K 消息，返回所要求的知识；若 B 处于中负载状态，回复 SEND-K 消息，只返回 B 节点的动态负载知识；若 B 处于重负载状态，由于其知识发送 agent 已经停止工作 (见下面本节点负载知识获取和评价 agent) ，连接将会超时， A 可以由此得到关于 B 动态负载知识的如下断言并写进 A 的知识库和黑板中： f_{B1}(RL)=high 0.9; f_{B2}(TL)=high 0.5 ， 0.9 和 0.5 为置信度。当 A 向所有周围节点发送完 REQ-K 消息后且仍处于轻负载状态则将等待一段时间 λ 后再发 REQ-K 消息 (λ 的暂设为 5s) 。为了减少网络通信开销，其他节点先天知识和静态负载知识可以直接从本节点知识库管理器 agent 中输入。

一个节点 A 的知识学习 agent 新学习到的周围节点动态负载知识与 A 中原有周围节点动态或静态负载知识，有一个融合处理的过程，融合结果将放入 A 的动态负载子知识库中，必要时还将放在黑板上。其融合算法如下： (a) 若 A 的动态负载和静态负载子知识库中均无节点 B 的负载知识存在，则将学习到的 B 的动态负载知识直接写入到 A 的动态负载子知识库中 (若置信度高于某个阈值，还应将其放在 A 的黑板上) ；

否则，若 A 的动态负载子知识库中有 B 的动态负载知识存在，则将其与新学习到的 B 的动态负载知识一起融合，转 (b);

否则，将 A 的静态负载子知识库中有关 B 的静态负载知识与新学习到的 B 的动态负载知识一起融合，转 (b); (b) 若新旧两种知识断言相同，只是置信度不同 (设分别为 cf_1、cf_2) ，则综合负载知识断言仍然相同，置信度为 $(\mathrm{cf}_1+\mathrm{cf}_2)/2$ ；如关于节点 B 的新旧两种知识分别为： f_{B1}(RL)=low, $\mathrm{cf}_1=0.80$ 和 f_{B1}(RL)=low, $\mathrm{cf}_2=0.40$ ，则融合结果为 f_{B1}(RL)=low, cf=0.60 ；

否则取新学习到的知识为融合结果，置信度亦取新知识的置信度；

(6) 知识发送 agent: 其作用是： (a) 当本节点刚刚加入系统时向其他节点的知识学习 agent 发送各种知识； (b) 应其他节点知识学习 agent 的要求根据本节点负载状态发送本节点知识； (c) 轻负载时向周围节点发送本节点知识 (处理方法与知识学习 agent 中的 (b) 刚好相反) 。这里的“发送”只是先发送“ ADV-MY-K ” (我想发送知识) 和知识类型消息，得到可以发送的响应时再传送具体知识，若连接超时，可以得到与 (5) 中连接超时相似的判断。

(7) 本节点负载知识实时获取和评价 agent: 周期性地或应本节点任务决策处理 agent 的要求获取本节点当前负载状态信息。对于后者，直接以“ (重负载状态 1) ”的输出方式给出本节点当前负载状态；对于前者，需要将获取到的当前负载状态与记录的历史负载状态进行比较，确认节点持续处于同一状态的次数超过一定阈值 Count(默认值 Count=3 ，获取周期为 5s) 时方才以消息 Load-Status 后加当前负载状态的方式给出本节点负载状态，并同时传送给任务决策处理 agent 、知识发送 agent 和知识学习 agent 。若 Load-Status 后携带的是重负载状态，则任务决策处理 agent 要考虑迁出任务，知识发送 agent 和知识学习 agent 要暂时停止工作进入休眠以减轻节点负载；若 Load-Status 后携带的是轻负载状态，则若知识发送 agent 和知识学习 agent 已进入休眠，要重新唤醒其开始工作，任务决策处理 agent 则需考虑迁入任务。

(8) 任务决策处理 agent: 其作用是： (a) 接收并处理本节点或其他节点 PVM 虚拟机发送来的增加一个任务、删除一个任务消息； (b) 接收并处理本节点负载知识获取和评价 agent 发送来的 Load-Status 消息； (c) 接收并处理其他节点 agent 发送来的 Req-Mout-Task(请求迁出一个任务)、 Req-Min-Task(请求迁入一个任务) 等消息； (d) 任务决策处理 agent 之间的其他交互消息。因这些消息可能引起任务的迁出或迁入，在处理这些消息时最主要的根据就是本节点当前负载状态和本节点知识库中关于其他节点的负载知识，若本节点被标记为“已委托”，则以委托节点知识库中关于其他节点的负载知识为准。下面简单地列出当 SM-ADDTASK(增加一个任务) 消息到来时任务决策处理 agent 查询到本节点为重负载或接收到 Load-Status 消息且后面的负载信息为重负载时决定迁出一个任务 (若有任务可迁出) 的算法描述，当节点处在轻负载状态时申请迁入一个任务的算法与此有类似之处，在此不再赘述。

算法　迁出一个任务 Req-Mout-Task(taskid)

(1) (1.1) 若本节点被标记为已“委托”，则

(1.1.1) 从委托节点动态负载子识库中选择满足迁入任务且置信度最大的轻负载节点 B(动态负载知识以事实表达，可以直接进行比较选择，下同);

(1.1.2) 若找到满足上述条件的 B ，则转 (2); 否则从委托节点静态负载子知识中选择满足上述条件的 B(静态负载知识以规则表达，需以当前日期、时间等事实为基础进行推理选择，下同)，转 (2);

否则以 (1.1.1) ， (1.1.2) 相同的方式从本节点知识库中选择满足迁入任务且置信度最大的轻负载节点 B;

(1.2) 若没找到满足条件的 B ，则算法停止　/* 等待其他轻负载节点主动请求任务 */

(2) 若存在多个 (>1) 满足上述条件的节点，则

(2.1) 若 A 中或其委托节点中有这些节点的先天知识存在，则从先天知识中选择一个 CPU 执行速度最快的节点作为 B ，转 (3); 否则任选一个满足上述 (1.1.1) 迁入条件的节点作为 B ，转 (3);

(3) 向 B 发送 Req-Mout-Task 消息 (请求迁出一个任务到 B) 并等待返回消息

/* B 的任务决策处理 agent 接收此消息后对自己的负载进行评价，得出是否可以迁入任务的判断，并向 A 的任务决策处理 agent 返回 Acc-Min(接受迁入) 或 Deny-Min(拒绝迁入) 消息 */

(4) 若 B 返回的消息为 Acc-Min ，则

(4.1) 从 A 的就绪任务队列首部选择一个任务迁入 B 中 (A 的就绪任务队列按 FIFO 方式存放，按照 Blumofe 的分析 [9]，重负载节点进行任务迁移时，选择就绪任务队列首部的任务进行迁移效率更高)。

(4.2) 将 A 或其委托节点的相应知识库中关于 B 的动态负载知识置为： f_1(RL)=high 置信度为 0.95; f_2(TL)=high 置信度为 0.95 以避免使用该知识库的节点重复地向 B 迁移任务. (ITABLM 中增加了部分 PVM 函数，并对原部分 PVM 函数做了修改，其中就包括退出一个 PVM 任务的函数 pvm-exit( )，修改后的 pvm-exit( ) 函数当发现本任务是由另外一个节点 A 迁移而来的时，将所在节点 B 的动态负载知识及时返回给 A 的任务决策处理 agent ，以便 A 及时得到 B 的动态负载信息并修改上述关于 B 动态负载知识的设置)。否则

(4.3) 若 B 返回的消息为 Deny-Min, 则判定 B 的： f_1(RL)=high 置信度为 0.7; f_2(TL)=high 置信度为 0.75;

(4.4) 若连接超时，则判定 B 的： f_1(RL)=high 置信度为 0.9; f_2(TL)=high 置信度为 0.5;

(4.5) 转 (1) 继续执行

5 实验及性能评价

ITALBM 是我们实验已久的一个机群系统中智能化任务分配和负载平衡模型，目前已在基于 PVM 的机群系统上得到了初步的实现，最终目的是经过修改用于广泛的 NOW(Network of Workstations) 模型．在基于 PVM 的机群系统上的实现方法是：分别实现了以上各 agent ，并以统一的任务决策处理 agent 替换了本节点上所有 PVM 虚拟机中的资源管理器 RM ，特别是对产生一个任务 SM-SPAWN 、增加一个任务 SM-ADDTASK 、删除一个任务 SM-DELTASK 等消息的响应处理函数做了重大的修改，使之转移至本节点任务决策处理 agent ，由后者进行具体的任务调度处理 (指定在某台工作站上产生一个任务的情形除外)，并增加了部分 PVM 函数和对原部分 PVM 函数做了修改，如对 pvm-exit( ) 函数的修改以实现节点间动态负载信息及时交互的目的等． agent 之间的通信语言采用的是自行开发的基于 TCP/IP 协议基础上的系统专用消息传递语言 ITALBM-ACL ．

为了验证 ITALBM 的性能，我们在某重点实验室进行了一段时间的跟踪实验．实验环境为： 15 台工作站通过 100M 快速以太网相连，其中 Sun Sparc 5 和 Sun Sparc 20 各两台，其余为运行 SCO UNIX 或 LINUX 的 Digital PC5000 微机，共有 13 台工作站的运行规律与表 2 较为相似．实验内容是：将每台工作站的运行规律知识通过知识库管理器 agent 输入到每个节点的静态负载子知识库中，将各工作站逐次加入到系统中，并配置到多个不同的 PVM 虚拟机中，在每个 PVM 虚拟机中含有主 pvmd 的节点上启动 PVM 程序 rangen ，对各种配置和运行情况进行比较． rangen 为 SPMD 程序，其功能是利用随机数并行产生算法产生 $\min(20,m)$ 万个互不相同的随机数，流程如下：

(1) 读入参数 m(若为主 rangen 任务，从参数表中读；否则从 pvm 缓冲区中读);

(2) 若为主 rangen 任务，则

(2.1) i=0;

(2.2) 循环执行下列语句，直到 $m<0$

(2.2.1) pvm-spawn(“ rangen ”， (char **)0 ， 0 ， “ ”， 1 ， & sub-tid[++i]); /* 产生第 i 个 rangen 子任务，并由本节点任务决策处理 agent 决定在哪个节点上运行 */

(2.2.2) $m=m-20$; $n=\min(20,m)$;

(2.2.3) pvm-pkint(& i, 1, 1) /* 指定刚产生的 rangen 子任务为主 rangen 的第 i 个子任务，并通过传递 i 来保证来所有工作站上产生的随机数互不重叠 */

(2.2.4) pvm-pkint(& n, 1,1); /* 将 n 作为 m 传递给第 i 个 rangen 子任务 sub-tid[i]; */
否则产生 $\min(20,m)$ 万个随机数，并以子文件的方式写入本地硬盘中．

实验结果如图 2 所示．其中“ PVM 虚拟机数 =3 ”表示系统中共有 3 个 PVM 虚拟机，每台加入系统的工作站均同时属于这 3 个虚拟机．当机群系统中只有一台工作站时， 3 个 PVM 虚拟机的主 pvmd 均在该工作站上运行；当有两台工作站时，其中一台上有两个 PVM 虚拟机的主 pvmd ，另一台上为一个；当有 3 台或 3 台以上工作站时，这 3 个 PVM 虚拟机的主 pvmd 分别分布在其中 3 台工作站上．

以上实验结果表明，(1) ITALBM 对运行规律性较强，且大部分工作站处于空闲状态时 (如上面测试时间 >18: 00) 有着很好的加速比； (2) 在相同情况下，当系统中大部分节点均处于重负载时， ITALBM 效率明显优于发送者驱动，稍差于接收者驱动；当系统中大部分节点处于轻负载时， ITALBM 效率高于发送者驱动和接收者驱动；当各节点负载分布不均匀时 ITALBM 效率稍高于接收者驱动，明显高于发送者驱动．

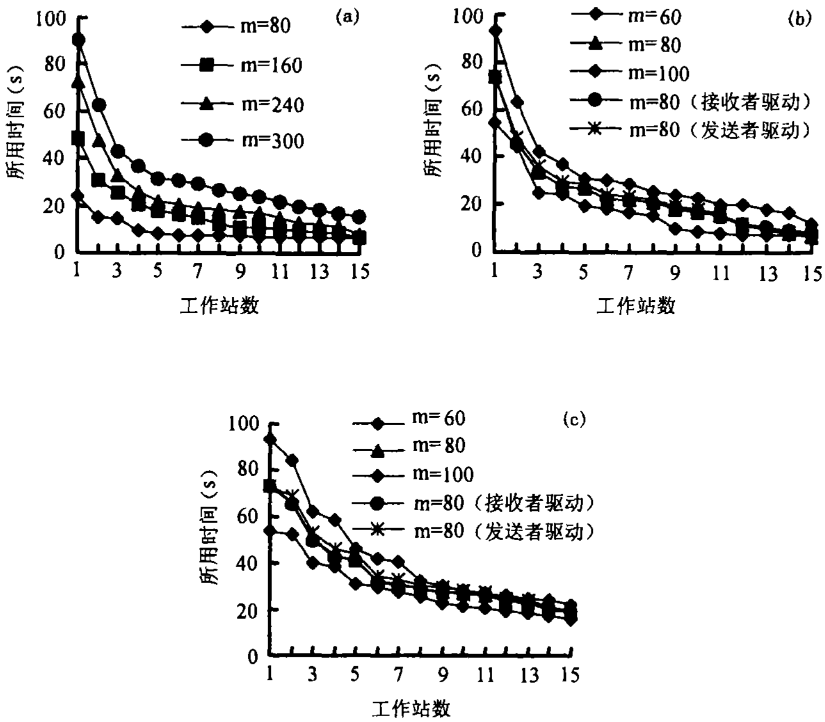

图 2　ITALBM 中程序 rangen 在不同时刻和不同 PVM 虚拟机个数情况下的运行时间及其与接收者驱动和发送者驱动情况下的比较
(a) PVM 虚拟机数 =1 时间：18:00 以后；　(b) PVM 虚拟机数 =3 时间：18:00 以后
(c) PVM 虚拟机数 =3 时间：13:00–17:00

6 结 束 语

智能化任务调度和负载平衡是实现基于机群系统的大规模并行处理系统的一个重要发展方向。ITALBM 通过将节点负载区分为资源负载率 f_1(RL) 和任务负载率 f_2(TL)、引入多 agent 系统和增加、修改 PVM 功能函数的方式得到了一个初步的实现。实验表明，这种方法总体效率优于接收者驱动和发送者驱动，且具有良好的可伸缩性、可扩充性和自适应性。进一步的研究包括：节点中各 agent 的深入研究、agent 之间通信和知识交互方式的改进、以及将 ITALBM 修改用在广泛的 NOW 环境中等。

参 考 文 献

[1] Chen Jun, Xie Li, *et al.*, A model for intelligent task scheduling in a large distributed system, ACM Operating System Review, 1990, 24(10), 26–33.

[2] D. Arredondo, M. Errecalde, S. Flores, Load distribution and balance support in a workstation-based distributed system, ACM Operating System Review, 1997, 32(4), 46–59.

[3] 齐 红，鞠九滨，工作站网络上协作任务的调度，软件学报，1998, 9(1), 14–17.

[4] 陈华平，计永昶，陈国良，分布式动态任务负载平衡调度的一个通用模型，软件学报，1998, 9(1), 25–29.

[5] 陆桑璐，谢 立，一个动态自适应的迁移和协同调度模型，软件学报，1997, 8(10), 752–759.

[6] 姚 郑，高 文，面向 Agent 的程序设计风范，计算机科学，1995, 22(6), 7–11.

[7] 李建民，王建华，李 毅，多智能体系统的一种合作机制，计算机研究与发展，1998, 35(2), 133–139.

[8] 樊晓聪，徐殿祥，郑国梁，Agent 的行为置信度及其修改规则，计算机学报，1998, 21(1), 40–46.

[9] R. D. Blumofe, D. S. Park, Sheduling large-scale parallel computations on network of workstations, Technical Report, Massachusetts Institutue of Technology, 1994, 1–4.

AN INTELLIGENT TASK ALLOCATING AND LOAD BANLANCING MODEL BASED ON PVM IN CLUSTER OF WORKSTATION

Chen Ximing* ** Lu Xianliang*

*(*Dept. of Computer, University of Electronic Sci. and Tech. of China, Chengdu* 610054, China)

**(*National Electronic War Laboratory, Chengdu* 610036, China)

Abstract This paper proposed and created an intelligent task allocating and load balancing model based on PVM in cluster of workstations—ITALBM, which introduced knowledge processing and multi-agent in DAI to the traditional task allocating and load banlancing to perform knowledge administrating, communicating and to coodinately allocate the nodes' tasks, and further to balance the nodes' load. Through this way, better performance, scalability, extensibility and adapibility are gained.

Key words Task allocating, Load balancing, Agent, Knowledge

陈锡明： 男，1970 年生，博士，研究方向为分布式并行处理，负载平衡，人工智能.

卢显良： 男，1944 年生，教授，博士生导师，主要研究方向为操作系统，计算机网络，分布式并行处理.