计算机科学 2003Vol. 30№. 10

基于规则的分层负载平衡调度模型

李冬梅[1] 施海虎[2] 顾毓清[3]

(北京林业大学计算机科学与技术系 北京 100083)[1] (中国科学院软件研究所 北京 100080)[2]

A Hierarchical Load Balancing Scheduling Model Based on Rules

LI Dong-Mei[1] SHI Hai-Hu[2] GU Yu-Qing[2]

(Department of Computer Science and Technology, Beijing Forest University, Beijing 100083)[1]

(Software Institute, Chinese Acadimy of Sciences, Beijing 100080)[2]

Abstract On a massively parallel and distributed system and a network of workstations system, it is a critical problem to increase the utilization efficiency of resources and the answer speed of tasks by using effective load balancing scheduling strategy. This paper analyzes the scheduling strategy of dynamic load balancing and static load balancing, and then proposes a hierarchical load balancing scheduling model based on rules. Finally, making some comparisons with other scheduling models.

Keywords Rules, Hiberarchy, Load balancing, Scheduling model

1 引言

随着计算机技术和网络技术的发展，大规模并行分布处理系统，尤其网络工作站机群系统得到了广泛应用。但是随着分布式技术研究的深入，在某一时刻，一些工作站的负载极重而另外一些工作站的负载极为空闲所造成的网络资源浪费现象十分突出。如何采取有效的负载平衡调度策略来平衡各工作站的负载，从而提高整个系统资源的利用率，已成为人们的研究热点[1~5]。

通常的负载平衡分析方法分为动态和静态两种。当用户提交的任务的大小、任务间的通信量以及各工作站的负载情况已知或估算给定时，负载平衡问题就是如何将任务均匀分配给各工作站，使系统效率最高。此时工作站的任务量、任务间的通信量都为定值，不随时间变化，故称之为静态负载平衡调度策略。当任务量、任务间的通信开销以及各工作站的负载情况未知或不确定时，需要人为地对各变量进行预测或动态分析，以便在执行过程中动态分配负载。此时每台工作站上的任务是动态产生的，并且每台工作站的负载大小是动态变化的，故称之为动态负载平衡调度策略。

静态负载平衡调度策略只要进行一次任务划分与工作站分配，并一直保持这个划分和分配到任务完成，其优点是可以得到最优负载平衡结果，其缺点是在现实网络环境中很难做到。另外，因为静态调度服务器需要掌握各工作站的实时负载变化情况，随着系统规模的增大，静态负载平衡调度效率快速下降。这方面的代表性工作有文[6,7]。

动态负载平衡调度策略是根据系统局部范围内的一些负载信息来进行负载平衡，可以根据系统负载的变化情况适当调整，更能反映分布式系统的实际情况。它的最大优点是具有良好的可扩展性，但其缺点也十分明显，调度结果只能得到部分的平衡，并且参与调度的系统也是局部的。另外，动态负载平衡调度通常忽略了工作站可能出现故障的情况，而在大规模分布式系统中，系统的不稳定是难以避免的。此外，动态负载平衡调度也往往忽略了任务之间的通信量，这样在通信量较大的分布式系统中将增大整个系统的负载，进而造成整个系统的不稳定。这方面的代表性工作有文[8~10]。

根据静态负载平衡调度与动态负载平衡调度的这些特点，本文提出了一种分层负载平衡调度模型，这是静态调度与动态调度的混合模型，也是一种通用模型，当层次结构自底向上缩为单层扁平结构时，它退化为静态调度模型，当层次结构自顶向下塌为单层扁平结构时，它退化为动态调度模型。本文第2、3节在讨论了负载平衡调度的一般模型基础上，提出了一个分层负载平衡调度模型，第4节给出了该模型的实现规则。文章最后对该模型与静态调度策略、动态调度策略进行了分析比较。

2 负载平衡调度的一般模型

负载平衡调度的一个重要目标就是提高系统性能，即缩短用户任务的平均响应时间。一项任务的响应时间依赖于其所运行的工作站上的负载。负载平衡的另一个重要目标是均匀地、充分地利用整个系统的资源。这个目标与前者是一致的，资源的使用越平衡，任务的响应时间就越短。负载平衡调度算法需要解决下列问题：什么时候启动负载平衡调度？选择哪些任务进行调度？负载平衡调度的源结点和目标结点是哪些？转移多少任务？

负载平衡调度的一般模型可以用一个四元组(E,T,L,S)来表示。其中，因子E表示分布式系统的网络环境(Environment)，因子T表示用户提交的任务(Task)集，因子L表示系统的负载(Load)评价，因子S表示系统所采用的调度策略(Strategy)。

1)*分布式系统的网络环境E* 因子E既要描述组成分布式系统工作站的功能与性能特性，又要描述工作站所构成的分布式系统的拓扑结构。分布式系统中的工作站很难做到必

李冬梅 硕士，讲师，主要研究领域为数据库与网络工程，并行计算与系统软件。施海虎 博士，副研究员，主要研究领域为分布式人工智能，软件工程，知识工程。顾毓清 研究员，主要研究领域为软件工程。

须是同构的，功能上可以区分图形工作站、科学计算工作站、I/O 工作站等，性能上可以区分单处理机工作站、多处理机工作站、r-处理机工作站。本文假设分布式系统是异构的。

分布式系统的拓扑结构可以表示成一个图结构。根据负载平衡调度算法的不同，图的构成原则各不相同。参见图 1，本文假设，图的结点表示工作站，两结点间的边表示两台工作站是物理邻接的，整个系统的拓扑结构是由若干相对独立的任务调度组组成的层次结构，组内的工作站两两邻接。任务调度组是负载平衡调度的基本单位，每项分布并行计算任务在任务调度组内完成。由于组内的工作站两两物理邻接，因此可以忽略并行计算任务之间的通信量。

2）任务集 T　分布并行计算任务分为两种模式，一种是基本模式，即将一个大计算量的任务分解为一些相互独立的子任务，在一组工作站上并行执行，最后将子任务结果汇集为总的结果。另一种协作模式，即在计算过程中，子任务间需要同步，交换中间结果，因而要求基本上同时开始，并以同样速度执行。另外，任务集 T 还要考虑其它因素，例如，在给定的时间段内，用户提交的任务总量、任务的提交频率、先后次序、任务类型等。

3）负载评价 L　影响负载的因素非常复杂，如任务到达数、任务离开数、瞬时队列长度、平均队列长度、平均响应时间、平均伸量系数（任务从到达到离开的时间与在此阶段收到的服务时间之比）、CPU 利用率、未完成的工作量、可用资源（如内存等）情况、交付任务的要求等，其中有些信息是得不到或不准确的。在所有这些负载指标中，运行队列任务数的效果最好[11]。为简单起见，本文将任务队列中的任务数作为描述负载的唯一参数，事实上，如果采集过多参数，则会因增加额外开销而得不到所希望的性能改善。本文设置两个阈值 α、β，将负载状态划分为 4 种类型，即空载、轻载、适载和重载，任务量 0、(0,α)、[α,β]、(β,∞)分别表示空载、轻载、适载和重载。设 L(x)表示 x 的负载状态，那么，L(x)∈{“空载”,“轻载”,“适载”,“重载”}，其中 x 既可以是一台计算机，也可以是一个网络系统。

4）调度策略 S　本文采用文[7]给出的多处理机系统的负载平衡调度算法 MPLBSA(MultiProcessor Load Balancing Scheduling Algorithm)作为静态负载平衡调度算法。

根据一次负载平衡调度的启动者来划分，动态负载平衡调度主要可分为发送者启动、接收者启动和对称启动等 3 种。发送者启动策略由发送者来触发负载分配，当一个工作站成为发送者时，主动寻找接收者来接收自己负载。接收者启动策略由接收者触发负载分配，空闲结点逐个向邻接结点请求任务。对称启动，即前两种的综合，要求发送者和接收者均参与负载分配的活动，在系统轻载时，发送者启动有效，反之接收者启动有效。

3　分层负载平衡调度模型描述

3.1　分层负载平衡调度模型的主要思想

考虑一个只含 5 家公司的简单社会模型，每家公司下设 3 个业务部门。人员组成上，每家公司设立 1 名总经理、3 名部门经理，每个部门由 20 名技术人员组成，合计 300 名工作人员、20 名管理人员。它们的工作模式是，公司与公司之间是既合作又竞争的动态协作关系，而公司内部则进行层次化集中式管理，即总经理负责协调部门之间的工作分工，部门经理统一安排部门内部技术人员的工作任务。显然，这种模型完全符合现代企业的协作工作模式。将上述模型引入分布式系统，得到一个分层负载平衡调度模型。

3.2　分层负载平衡调度模型描述

在描述分层负载平衡调度模型 HLBSM(Hierarchical Load Balancing Scheduling Model)之前，首先给出几个概念。

1）HLBSM 系统：采用 HLBSM 模型的分布式系统称为 HLBSM 系统。

2）工作站：指 HLBSM 系统中最终参与执行用户任务的计算机。

3）任务调度组：任务调度组即 HLBSM 系统中执行负载平衡调度动作的基本单位，根据负载平衡调度策略的不同分为静态任务调度组与动态任务调度组；静态任务调度组由一台调度服务器和若干调度结点组成；动态任务调度组只包含一些动态调度结点。

4）任务调度层：包含若干相互独立的任务调度组。

考虑一个 n(>0)层负载平衡调度模型 n-HLBSM 系统（参见图 1），每个任务调度层由 $l_i(i=1,\cdots,n)$个任务调度组组成，每个任务调度组包含 $m_{ij}(i=1,\cdots,n,j=1,\cdots,l_i)$台调度结点。我们把 n-HLBSM 系统的所有计算机组成的集合记为：$\Omega=W\cup SS$，其中，集合 W 表示工作站(Workstation)集，集合 SS 表示调度服务器(Scheduling Server)集，则，$|W|=\Sigma m_{ij}$，$|SS|=\Sigma l_i-1$。

n-HLBSM 系统具有如下特征：

1）第 1～n－1 任务调度层均采用静态负载平衡调度策略，第 n 任务调度层采用动态负载平衡调度策略。当 n＝1 时，既可以采用静态负载平衡调度策略，也可以采用动态负载平衡调度策略。

2）第 i∈[1,n－1]任务调度层的调度服务器由第 i 任务调度层的调度结点担任。

3）工作站作为调度结点位于第 1 任务调度层。

4）第 n 任务调度层是顶层任务调度层，只包含一个任务调度组，由于采用动态负载平衡调度策略，因此该层没有调度服务器。

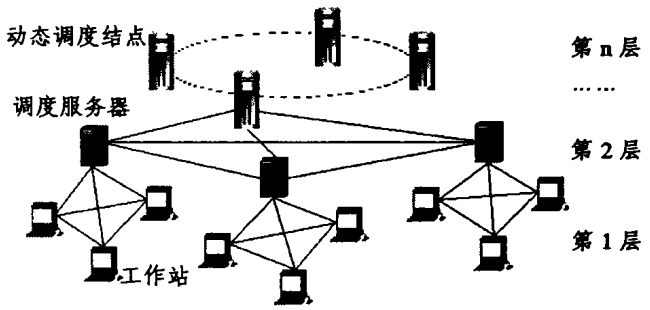

图 1　分层负载平衡调度模型

在 3.1 节的简单社会模型中，$n=3,l_1=15,l_2=5,l_3=1$，$m_{1\cdot1}=\cdots=m_{1\cdot15}=20,m_{2\cdot1}=\cdots=m_{2\cdot5}=3,m_{3\cdot1}=5$。即该模型共分 3 层，第 1 任务调度层有 15 个任务调度组，每个任务调度组含 20 台工作站和 1 台调度服务器；第 2 任务调度层有 5 个任务调度组，每个任务调度组含 3 个调度结点和 1 台调度服务器；第 3 任务调度层为顶层，只有 1 个任务调度组，5 个调度结点。第 1～2 任务调度层采用静态负载平衡调度策略，第 3 任务调度层采用动态负载平衡调度策略。

4　分层负载平衡调度模型的实现原理

4.1　分层负载平衡调度模型的图结构

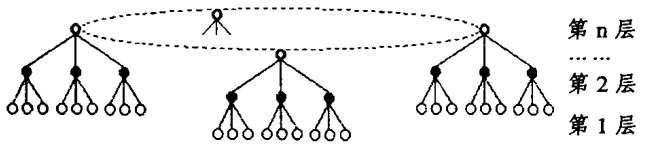

图 2　分层负载平衡调度模型的图结构 G

n-HLBSM 的图结构如图 2 所示。根据负载平衡调度动作的功能不同，n-HLBSM 系统中的所有计算机分为 3 类，即工作站、调度服务器、动态调度结点，它们分别对应图 2 中的空结点、黑结点、虚结点。显然，该图具有如下特征：

1) n-HLBSM 的图结构是由若干棵树组成的森林。

2) 空结点、黑结点、虚结点分别对应树的叶结点、中间结点、根结点。

3) 在每一棵树内部实行静态负载平衡调度策略。

4) 在仅由树根组成的子图中采用动态负载平衡调度策略。

4.2　动态任务调度消息类型

n-HLBSM 模型的顶层实行动态负载平衡调度，采用接收者驱动、发送者驱动或对称启动调度策略。无论执行哪一种调度策略，调度结点之间都需要相互交换负载信息。接收者驱动策略由轻载结点向邻接的重载结点发送轻载信息，请求任务；发送者驱动策略由重载结点向邻接的轻载结点发送重载信息，执行任务迁移。动态任务调度消息主要有下列两种类型：

1) $I_{sender}(x)$消息：表示结点 x 的当前负载状态为重载，希望把部分任务迁移出去。

2) $I_{receiver}(x)$消息：表示结点 x 的当前负载状态为轻载，愿意替其它结点分担任务。

4.3　数据结构

任务调度结点在参与负载平衡调度过程中，需要相互交换负载状态变化信息，例如哪些结点空闲、哪些结点重载等，并由任务调度结点记录保存，作为负载平衡调度的依据，这样就避免了负载信息的反复交换，从而减少通讯开销，也不会给整个系统增加额外的负担与打扰。保存负载平衡调度信息的数据结构包括：

1)任务队列 TQ(Task Queue)：用来记录任务调度结点待求解的用户任务，按照 FIFO 的原则添加任务，根据网络负载的实时状况对这些任务进行统一调度和执行。用 TQ(x)表示任务调度结点 x 的任务队列。

2)负载信息表 LT(Load Table)：用于存放当前任务调度组中所有任务调度结点的负载信息，调度服务器根据该表存放的信息对任务队列 TQ 中的任务进行负载平衡调度。

3)发送者表 ST(Sender Table)：用来保存 $I_{sender}(x)$消息，记录当前系统中重载结点的信息，专用于动态负载平衡调度策略。

4)接收者表 RT(Receiver Table)：用来保存 $I_{receiver}(x)$消息，记录当前系统中轻载结点的信息，专用于动态负载平衡调度策略。

4.4　实现原理

n-HLBSM 系统包含 3 类结点，即工作站、调度服务器、动态调度结点。由于它们参与负载平衡调度的作用各不相同，因此实现方法也不一样。

工作站用任务队列 TQ 来记录待处理的用户任务，主要负责执行用户提交的任务。它将任务调度服务器根据统一调度结果安排的任务添加到任务队列 TQ 中；当接收到用户提交的任务或者完成某项任务而引起负载情况发生变化时，它将把负载变化情况及时通知任务调度服务器。

调度服务器用一张表和一个队列记录负载、用户任务信息。任务队列 TQ 保存该任务调度组需要处理的用户任务。负载信息表 LT 记录任务调度组内部的负载状态信息，时刻监视和更新负载信息表，将本组的负载变化情况及时通知高一层任务调度服务器，并根据负载信息表的实时负载状况统一调度任务队列中的任务。

动态调度结点兼具静态调度和动态调度的职能，它维持一个队列和三张表。任务队列 TQ 记录待求解的用户任务，既可以采用静态调度也可以采用动态调度。发送者表 ST 和接收者表 RT 记录动态调度组中各结点的负载状态信息。负载信息表 LT 记录静态调度组的负载状态信息。

5　分层负载平衡调度模型的实现规则

n-HLBSM 系统的负载平衡调度动作是通过事件触发不同类型的规则来实现的。根据负载平衡调度策略的不同，规则类型可以分为静态调度规则和动态调度规则两类。相对于静态调度策略，动态调度策略需要处理动态任务调度消息事件。

根据消息内容的差异，即用户任务的接收与调度、负载状态变化情况、动态调度策略中任务请求与发送，规则类型又可以划分为任务消息处理规则、负载消息处理规则、动态调度消息处理规则。本文假设 n-HLBSM 系统的每一台计算机均可以接收来自系统用户的任务。另外，只有当某台工作站完成某项任务或者计算机接受新任务时，才会引起系统的负载信息发生变化，对于前者，负载变轻信息的传递是自底向上的过程。

各类系统结点处理事件的方法各不相同，下面按工作站、1 层调度服务器、$i \in [2, n-2]$层调度服务器、n－1 层调度服务器(即动态调度结点)等 4 种类型，给出 n-HLBSM 系统的主要实现规则。

本文的规则用谓词逻辑表示，为方便讨论，下面定义部分常量、谓词、函数：

常量定义：

1) user：表示 n-HLBSM 系统的用户；

2) workstation：表示 n-HLBSM 系统的工作站；

3) DD_node：表示 n-HLBSM 系统的动态调度结点(Dynamic Scheduling node)；

4) L_load：表示“轻载(Low load)”负载状态；

5) H_load：表示“重载(High load)”负载状态；

谓词定义：

6) G(c,0)：表示结点 c 作为调度服务器所在的任务调度组(Group)；

7) G(c,1)：表示结点 c 作为调度结点所在的任务调度组；

8) S(c)：表示结点 c 作为调度结点所在任务调度组的调度服务器(Server)；

9) N(c)：表示结点 c 作为调度服务器所在任务调度组的调度结点集(Node)；

10) T(x)：表示由 x 提交或迁移过来的任务；

函数定义：

11) Send(x,y)：往结点 x 发送动态调度消息 y；

12) TransferL(x):当前结点将自身的当前负载变化情况传递给结点 x;

13) TransferT(x,t):当前结点将任务 t 迁移给结点 x;

14) Do(x):执行任务队列 x 的头任务;

15) Add(x,y):将新任务 y 添加到任务队列 x 的队尾;

16) MPLBSA(TQ(x)):调用静态负载平衡调度算法 MPLBSA,在以 x 为调度服务器的任务调度组中,将 x 的任务队列进行负载平衡调度。

5.1 工作站的负载平衡调度规则

工作站需要处理二类事件,即任务的接收与处理事件和负载状态变化事件。

1)对于工作站 c,它接收的任务 t 既可能来自系统用户,也可能是 S(c)执行静态负载平衡调度的结果。它有 3 条任务消息处理规则:

Rule01. $t \in T(S(c)) \rightarrow Add(TQ(c),t)$。

Rule02. $(t \in T(user)) \wedge (L(c)=\text{"L\_load"}) \rightarrow Add(TQ(c),t) \wedge TransferL(S(c))$。

Rule03. $(t \in T(user)) \wedge (L(c) \neq \text{"L\_load"}) \rightarrow TransferT(S(c),t)$。

2)对于工作站 c,当 c 处理完成某项任务时,将引起自身负载状况发生变化。它有二条负载消息处理规则:

Rule04. $TQ(c) \neq \Phi \rightarrow Do(TQ(c)) \wedge TransferL(S(c))$。

Rule05. $TQ(c)=\Phi \rightarrow TransferL(S(c))$。

5.2 1 层调度服务器的负载平衡调度规则

1 层调度服务器需要处理二类事件,即任务的接收与处理事件和负载状态变化事件。

1) 对于 1 层调度服务器 c,它接收的任务 t 既可能来自于用户,也可能来自于工作站,同时也可能来自于 S(c)。它有 3 条任务消息处理规则:

Rule06. $t \in T(S(c)) \rightarrow MPLBSA(Add(TQ(c),t))$。

Rule07. $((t \in T(user)) \vee (t \in T(worksation))) \wedge (L(G(c,0))=\text{"L\_load"}) \rightarrow MPLBSA(Add(TQ(c),t)) \wedge TransferL(S(c))$。

Rule08. $((t \in T(user)) \vee (t \in T(workstation))) \wedge (L(G(c,0)) \neq \text{"L\_load"}) \rightarrow TransferT(S(c),t)$。

2) 对于 1 层调度服务器 c,它接收的负载变化消息既可能来自于工作站,也可能来自于 S(c)。它有二条负载消息处理规则:

Rule09. $TQ(c) \neq \Phi \rightarrow MPLBSA(TQ(c)) \wedge TransferL(S(c))$。

Rule05. $TQ(c)=\Phi \rightarrow TransferL(S(c))$。

5.3 i∈[2,n−2]层调度服务器的负载平衡调度规则

i 层调度服务器需要处理二类事件,即任务的接收与处理事件和负载状态变化事件。

1) 对于 i 层调度服务器 c,它接收的任务 t 既可能来自于用户,也可能来自于 N(c),也可能来自于 S(c)。它有三条任务消息处理规则:

Rule06. $t \in T(S(c)) \rightarrow MPLBSA(Add(TQ(c),t))$。

Rule10. $((t \in T(user)) \vee (t \in T(N(c)))) \wedge (L(G(c,0))=\text{"L\_load"}) \rightarrow MPLBSA(Add(TQ(c),t)) \wedge TransferL(S(c))$。

Rule11. $((t \in T(user)) \vee (t \in T(N(c)))) \wedge (L(G(c,0)) \neq \text{"L\_load"}) \rightarrow TransferT(S(c),t)$。

2) 对于 i 层调度服务器 c,它接收的负载变化消息既可能来自于 N(c),也可能来自于 S(c)。它有二条负载消息处理规则:

Rule09. $TQ(c) \neq \Phi \rightarrow MPLBSA(TQ(c)) \wedge TransferL(S(c))$。

Rule05. $TQ(c)=\Phi \rightarrow TransferL(S(c))$。

5.4 n−1 层调度服务器的负载平衡调度规则

n−1 层调度服务器需要处理 3 类事件,即任务的接收与处理事件和负载状态变化事件,与此同时,它又作为 n 层负载调度结点参与动态负载平衡调度,还需要处理动态调度消息。

对于 n−1 层调度服务器 c,它接收的任务 t 既可能来自于用户,也可能来自于 N(c),也可能来自于动态调度结点。它有三条任务消息处理规则:

Rule12. $t \in T(DD\_node) \rightarrow MPLBSA(Add(TQ(c),t))$。

Rule13. $((t \in T(user)) \vee (t \in T(N(c)))) \wedge (L(G(c,0)) \neq \text{"H\_load"}) \rightarrow MPLBSA(Add(TQ(c),t))$。

Rule14. $((t \in T(user)) \vee (t \in T(N(c)))) \wedge (L(G(c,0))=\text{"H\_load"}) \rightarrow (x \in RT(c) Send(x, I_{sender}(c))$。

对于 n−1 层调度服务器 c,它接收的负载变化消息只可能来自于 N(c),它有二条负载消息处理规则:

Rule15. $TQ(c) \neq \Phi \rightarrow MPLBSA(TQ(c))$;

Rule16. $(L(G(c,0))=\text{"L\_load"}) \rightarrow \forall x \in ST(c) Send(x, I_{receiver}(c))$。

n−1 层调度服务器 c 作为动态调度结点,需要处理来自其它动态调度结点的动态任务调度消息 I。它有 4 条动态任务调度消息处理规则:

Rule17. $((I=I_{sender}(x)) \wedge (L(c)=\text{"L\_load"})) \rightarrow Send(x, I_{receiver}(c))$。

Rule18. $((I=I_{sender}(x)) \wedge (L(c) \neq \text{"L\_load"})) \rightarrow add(ST(c),x)$。

Rule19. $((I=I_{receiver}(x)) \wedge (L(c)=\text{"H\_load"})) \rightarrow \exists t \in TQ(c) Send(x,t)$。

Rule20. $((I=I_{receiver}(x)) \wedge (L(c) \neq \text{"H\_load"})) \rightarrow add(RT(c),x)$。

6 三种调度模型间的对比分析

分层负载平衡调度模型 n-HLBSM 具有如下特点:

首先,n-HLBSM 是一个通用模型,综合应用了负载平衡调度的两种策略,既体现了静态调度策略的最优化特性,又体现了动态调度策略的实时性。当 n-HLBSM 自顶向下塌为单层结构时,该模型退化为动态调度模型;当 n-HLBSM 自底向上缩为单层结构时,该模型退化为静态调度模型。

其次,n-HLBSM 可以克服动态负载平衡调度模型的缺点。一般的动态调度算法往往忽视了任务调度组中各调度结点之间的任务通信量,并且忽略了调度结点可能出现故障的情况,这样在通信量较大的分布式系统中将增大整个系统的负载,进而造成整个系统的不稳定。而在 n-HLBSM 的第 1 任务调度层中,各个任务调度组相对独立,一方面,由于同一任务调度组内的工作站物理相邻、功能相近,它们之间的任务通信量可以忽略不计;另一方面,工作站的工作状态时刻得到调度服务器的"监视",当某台工作站由于某种原因出现故障时,调度服务器能够马上"侦知"异常情况并作出反应,将它的任

务迁移给其它工作站。

最后，n-HLBSM 具有很好的可扩展性。因为 n-HLBSM 的第 n 任务调度层实行动态调度策略，每个任务调度结点只要知道邻接任务调度结点的局部系统的信息，就能够执行负载平衡调度策略，这样就能够适应分布式系统不断扩展的需要，符合条件的任何任务调度结点都可以随时成为第 n 任务调度层的成员。可扩展性可以克服静态调度策略的固有缺点，即要求调度服务器掌握所有任务调度结点的当前负载情况和工作状态，这种要求的另一缺点是随着系统规模的扩大，静态调度算法的效率将明显下降。

结束语 近年来，负载平衡调度问题的研究成果层出不穷，人们采取不同的方法，从不同的角度进行了多方位的探讨。分布式任务分配算法大致可分为：基于图论的分配算法、整数规划方法、启发式算法等；负载平衡调度可分：集中式调度和分布式调度；负载平衡调度模型可分为静态负载平衡调度和动态负载平衡调度。本文提出的分层负载平衡调度模型 HLBSM 在易实现性、可扩展性、最优性、智能性和容错性等综合特性方面优于其它模型。文章给出了 HLBSM 系统的实现规则，今后将进一步研究实现智能 HLBSM 系统。

参 考 文 献

1 Willebeek-LeMair M H. Strategies for Dynamic Load Balancing on Highly Parallel Computers. IEEE Transactions on Parallel and Distributed System, 1993,4(9):979～993

2 Liao C-J. Tree-Based Parallel Load-Balancing Methods for Solution-Adaptive Finite Element Graphs on Distributed Memory Multicomputers. IEEE Transactions on Parallel and Distributed System, 1999,10(4):360～370

3 Hui C-C, Chanson S T. Hydrodynamic Load Balancing. IEEE Transactions on Parallel and Distributed System, 1999,10(11):1118～1137

4 Das S K. Parallel Processing of Adaptive Meshes with Load Balancing. IEEE Transactions on Parallel and Distributed System, 2001,12(12):1269～1279

5 Chow K-P, Kwok Y-K. On Load Balancing for Distributed Multiagent Computing. IEEE Transactions on Parallel and Distributed System, 2002,13(8):787～801

6 秦忠国，姜弘道. 静态负载平衡问题的表示与算法. 计算机科学，1998,25(2):95～97

7 李冬梅，施海虎. 多处理机系统的负载平衡调度算法. 软件学报，2003 (6)

8 林成江，李三立. 一种可适应的分布式动态负载平衡策略及其仿真. 计算机学报，1995,18(10):721～739

9 陈华平. 分布式动态负载平衡调度的一个通用模型，软件学报，1998,9(1):25～28

10 钟求喜，谢涛，陈火旺. 任务分配与调度的共同进化方法. 计算机学报，2001,24(3):308～314

11 鞠九滨，杨鲲，徐高潮. 使用资源利用率作为负载平衡系统的负载指标. 软件学报，1996,7(4):238～243

(上接第 10 页)

全部相同。

5.3 基于递归维的上翻操作的实现

设 e_f 为递归维成员，首先将多维数据集$\{\langle e_1,e_2,\cdots,e_{n-1},e_f,m\rangle\}$中的$\{\langle e_f,m\rangle\}$按照$\{\langle e_1,e_2,\cdots,e_{n-1}\rangle\}$进行分组，然后在每一个分组中对表达式 EXP 进行求值，算法[16]如下：

从左到右一次扫描表达式中的每一个符号，每遇一维成员 e，将 e.m 值压入栈顶暂存起来(对于 null 值的处理见定义 4.2.3)；每遇一个运算符，就从栈顶弹出相应的运算对象进行相应的运算，并将结果压回栈中；每遇一个 ↑e，输出$\langle e_1,e_2,\cdots,e_{n-1},e,m'\rangle$到结果集，$m'$为栈顶值。

基于递归维的上翻操作难于直接用 SQL 实现，因此我们采用'过程化的 SQL'或其它能够编写"存储过程"的语言来实现。

结论 本文将递归维概念引入多维数据模型，并对相关的多维操作进行了扩展，提高了模型对多维数据的表达能力和操作能力。本文提出了两个适用于递归维的、用于取代聚集函数的、输出为集合关系的汇总函数，用于支持基于递归维的对于多维数据集的上翻操作，提高了多维复杂计算的能力。由于对象关系数据库中所特有的对象标识引用也使得基于递归维的导航式的操作更为高效。

参 考 文 献

1 Pendse N. What is OLAP?. http://www.olapreport.com/fasmi.htm

2 Cabibbo L, Torlone R. A Logical Approach to Multidimensional Databases. In:Proc. of the EDBT 1998

3 Agrawal R, Gupta A, Sarawagi S. Modelling Multidimensional Databases. In:Proc. of the ICDE 1997

4 Li C, Wang X S. A data model for supporting on-line analytical processing. In: Proc. Conf. on Information and Knowledge Management, Nov. 1996

5 Gyssens M, Lakshmanan L V S. A Foundation for MultiDimensional Databases. In:Proc. of the VLDB 1997

6 Vassiliadis P. Modeling multidimensional databases, cubes and cube operations. In: Proc. of 10 th SSDBM 1998, Capri

7 Lehner W. Modeling Large Scale OLAP Scenarios. EDBT'98

8 Nguyen T B, Tjoa A M, Wagner R R. An Object Oriented Multidimensional Data Model for OLAP. In: Proc. of the First Intl. Conf. on Web-Age Information Management (WAIM'00), Shanghai, China, June 2000

9 Jagadish H V, Lakshmanan L V S, Srivastava D. What can hierarchies do for data warehouses? VLDB99

10 Ragged Dimension Support . http://msdn.microsoft.com

11 Pedersen T B, Jensen C S. Multidimensional Data Modeling for Complex Data. In:Proc. ICDE' 99

12 Reasoning about Summarizability in Heterogeneous Multidimensional Schemas. Carlos A. Hurtado, Alberto O. Mendelzon, ICDT 2001. 375～389

13 Niemi T, Nummenmaa J, Thanisch P. Logical multidimensional database design for ragged and unbalanced aggregation hierarchies. In:D. Theodoratos, et al. eds. DMDW'2001

14 Analysis Service. http://msdn.microsft.com/

15 Essbase. http://www.essbase.com/

16 蒋立源，康慕宁. 编译原理. 西北工业大学出版社，1999. 197

17 袁霖，李战怀. 属性维概念及其操作的研究. 计算机科学，2003(6)