·计算机工程与应用·

分布式并行数据库系统中任务分配算法的设计

顾 攀，刘心松，陈小辉，邱元杰，左朝树

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】在分析传统任务分配算法的基础上，设计了一种改进的混合启动自适应任务分配算法。该算法根据系统总体负载变化自适应地选择启动策略，考虑任务对资源的需求和系统中的数据分布，有效地选择节点进行探询，使任务分配的开销最小化。在分布式并行数据库系统DPSQL中的使用表明，该算法比传统算法提供更高的任务执行效率和更好的系统稳定性。

关 键 词 分布式并行数据库系统; 任务分配; 自适应; 混合启动

中图分类号 TP311.133.1 文献标识码 A

Design of an Algorithm for Task Assignment in Distributed Parallel Database System

GU Pan，LIU Xin-song，CHEN Xiao-hui，QIU Yuan-jie，ZUO Chao-shu

(School of Computer Science & Engineering, Univ. of Electron. Sci. & Tech. of China Chengdu 610054)

Abstract Based on the analysis of the algorithms commonly used for task assignment, this paper puts forward an improved both-initiated adaptive algorithm for task assignment. This algorithm adaptively determines initiated strategy according to the system load. In the system, the data requirement of each task and the data distribution are considered.The nodes are effectively chosen so as to minimize the costs for task assignment. This algorithm has been successfully applied in distributed parallel database system DPSQL. It has been shown that the method has better efficiency and stability than the common algorithms for task assignment.

Key words distributed parallel database system; task assignment; adaptive; both-initiated strategy

在分布式并行系统中的任务分配算法可广义地划分为动态、静态和自适应三种[1]。动态任务分配在启动策略上可分为发送者启动、接收者启动和混合启动[2]。自适应算法是一类特殊的动态任务分配算法，根据系统状态信息动态选择任务分配所采用的启动策略、转移策略及选择策略[3]。文献[4]提出了一种启发式稳定算法，刻意忽略了访问远程数据与本地数据的差别。文献[5]明确了应考虑各节点资源分布情况和任务对资源的需求，避免资源的远程访问，却没有根据这一原则提出具体的分配算法。文献[6]提出应任务要求转移和复制数据，当数据副本较大时，付出的代价很大。文献[6]还在算法中考虑了系统负载状况，由于负载轻重的划分未针对平均负载进行，使接收者启动在欠载时延迟过大，发送者启动在超载时效率太低。本文设计了一种改进的混合启动自适应任务分配算法，通过系统平均负载来确定各节点状态，自适应地选择启动策略；并根据数据分布情况对任务进行分配，减少了数据访问的代价，克服了盲目探询引起的不稳定性。该算法已运用于自主研发的分布式并行数据库系统DPSQL中，具有良好的效果。

1 分布式并行数据库系统DPSQL概述

分布式并行数据库系统DPSQL是由多台高性能PC通过高速网络连接在一起构成数据库服务器群，为大量用户提供高性能、高可靠、大数据量数据库服务的分布式并行系统。系统中的各节点协同完成全局数据目录维护、数据副本智能控制、节点状态切换等系统管理工作。对客户端而言，它是一个单一映象的数据库服务器群，隐藏了数据冗余和分布信息。任一节点均可代表服务器群接收用户请求，此后该节点(称为协调节点)根据用户的操作类型、数据的分布情况和系统的负载信息，将待执行的用户任务分配到最合适的节

收稿日期：2004 - 04 - 15

作者简介：顾 攀(1980 –)，男，硕士，主要从事分布式数据库系统等方面的研究.

点(称为执行节点)。执行节点在完成任务后，将执行响应传递给协调节点，最后将结果返回给客户。从上述系统结构及执行过程可以看出，执行节点的选择好坏对于用户任务的执行效率和整个系统的处理开销影响极大。因此，为了高效低价地将协调节点上的任务分配到执行节点，减少任务的执行时间，本文设计了改进的混合启动自适应任务分配算法。

2　改进的混合启动自适应任务分配算法

2.1　相关定义

设服务器系统$\boldsymbol{S}$由n个可用的节点构成，$\boldsymbol{S}=\{S_1,S_2,\cdots,S_n\}$；系统中的数据划分为$p$个数据分片，$\boldsymbol{D}=\{D_1,D_2,\cdots,D_p\}$；系统中共有$m$个任务，$\boldsymbol{T}=\{T_1,T_2,\cdots,T_m\}$；系统中任务等待队列$W$的长度为$Q$；$W_i$对应第$i$个节点待处理任务组成的队列，其长度为$Q_i$，$W=W_1\cup W_2\cup\cdots\cup W_n$，$Q=Q_1+Q_2+\cdots+Q_n$。

定义 1　节点负载L_i：某一节点的CPU队列长度与系统中所有节点的平均CPU队列长度的比值。

设节点S_i(i=1,2,⋯,n)的CPU队列长度为l_i，则节点S_i的负载为：

$$L_i=l_i/\overline{l}=l_i/(\frac{1}{n}\sum_{j=1}^{n}l_j)=nl_i/\sum_{j=1}^{n}l_j \tag{1}$$

定义 2　节点类型向量$\boldsymbol{T}_V$：根据节点负载L确定其取值，记录各节点类型的一维向量，其大小为n。

$$T_V[i]=\begin{cases}-1 & L_i<1-f \\ 0 & 1-f\le L_i\le 1+f \\ 1 & 1+f<L_i\end{cases}\quad\begin{matrix}\text{节点 } S_i \text{ 处于欠载，为接收节点}\\ \text{节点 } S_i \text{ 处于平衡，其负载适中}\\ \text{节点 } S_i \text{ 处于超载，为发磅节点}\end{matrix}\quad\begin{matrix}\text{阈值 } f \text{ 根据系统处理能力}\\ \text{和资源配置等条件设置}\end{matrix}\quad i=1,2,\cdots,n \tag{2}$$

定义 3　数据分布矩阵$\boldsymbol{D}_M$：记录数据分片在系统各节点分布情况的矩阵，其大小为$n\times p$。

$$D_M[i,k]=\begin{cases}1 & \text{节点 } S_i \text{ 上有数据分片 } D_k \text{ 的副本分布}\\ 0 & \text{节点 } S_i \text{ 上无数据分片 } D_k \text{ 的副本分布}\end{cases}\qquad 1\le i\le n,1\le k\le p \tag{3}$$

定义 4　数据分片冗余等级向量$\boldsymbol{R}_V$：记录各个数据分片在系统中冗余等级的一维向量，其大小为p。

$$R_{\boldsymbol{V}}[k]=\sum_{i=1}^{n}D_M[i,k]\qquad k=1,2,\cdots,p \tag{4}$$

定义 5　任务执行开销矩阵$\boldsymbol{A}_M$：记录任务在节点执行所产生开销的矩阵，其大小为$m\times n$。

$$\boldsymbol{A}_M=\{A_M[g,i]\,|\,1\le g\le m,1\le i\le n,A_M[g,i]\text{为任务}T_g\text{在节点}S_i\text{上的执行开销}\} \tag{5}$$

定义 6　任务分配矩阵$\boldsymbol{X}_M$：记录任务在节点中分配情况的矩阵，其大小为$m\times n$。

$$X_M[g,i]=\begin{cases}1 & \text{任务 } T_g \text{ 已分配给节点 } S_i\\ 0 & \text{任务 } T_g \text{ 未分配给节点 } S_i\end{cases}\quad 1\le g\le m,1\le i\le n \tag{6}$$

定义 7　节点任务执行开销数组$\boldsymbol{A}_A$：记录节点所有任务执行开销之和的一维数组，其大小为n。

$$A_A(i)=\sum_{g=1}^{m}(X_M[g,i]A_M[g,i])\qquad i=1,2,\cdots,n \tag{7}$$

定义 8　任务间通讯开销矩阵$\boldsymbol{T}_M$：记录分配于不同节点执行的两个任务间进行信息交换所产生通讯开销的矩阵，其大小为$m\times m$。

$$\boldsymbol{T}_M=\{T_M[g,h]\,|\,1\le g,h\le m,\ T_M[g,h]\text{为任务}T_g\text{和任务}T_h\text{被分配于不同节点执行的通讯开销}\} \tag{8}$$

定义 9　节点任务通讯开销数组$\boldsymbol{C}_A$：记录节点所有任务与其他节点任务通讯开销之和的一维数组，其大小为n。

$$C_A(i)=\sum_{g,h=1}^{m}\left[T_M[g,h]X_M[g,i](1-X_M[h,i])\right]\qquad i=1,2,\cdots,n \tag{9}$$

定义 10　节点任务总开销数组$\boldsymbol{F}_A$：记录节点计算开销与通讯开销之和的一维数组，其大小为n。

$$F_A(i)=A_A(i)+C_A(i)=\sum_{g=1}^{m}(X_M[g,i]A_M[g,i])+\sum_{g,h=1}^{m}\left[T_M[g,h]X_M[g,i](1-X_M[h,i])\right]\qquad i=1,2,\cdots,n \tag{10}$$

定义 11　节点间通讯开销矩阵$\boldsymbol{C}_M$：记录节点间传送单位信息产生通讯开销的矩阵，其大小为$n\times n$。

$$\boldsymbol{C}_M=\{C_M[i,j]\,|\,1\le i,j\le n,C_M[i,j]\text{为节点}S_i\text{和节点}S_j\text{ 间传送单位信息产生的通讯开销}\} \tag{11}$$

定义 12 节点间任务转移开销$P_{g(i,j)}$：某一任务在系统中的任意两点间转移所产生的开销。

设任务$T_g(g=1,2,\cdots,m)$执行代码的数据量为$C_{\text{size}}(T_g)$，任务T_g执行后响应和结果信息的数据量为$R_{\text{size}}(T_g)$，则T_g从节点S_i转移到节点S_j $(1\le i,j\le n)$所产生的开销$P_{g(i,j)}$为：

$$P_{g(i,j)}=C_M[i,j](C_{\text{size}}(T_g)+R_{\text{size}}(T_g)) \tag{12}$$

2.2 算法描述

传统的混合启动自适应任务分配算法在系统超载时，发送者启动部分的探询可能导致系统的不稳定性；在进行任务分配时，未考虑任务对于资源的需求，且未考虑任务分配时的开销最小化问题，容易导致任务的执行效率降低。为此，对混合启动自适应任务分配算法进行改进，改进的算法有以下特点：(1) 结合数据分布信息进行任务分配，满足其对资源的需求；(2) 任务转移后所产生的开销最小，提高了任务执行的效率；(3) 根据系统总负载来控制发送者启动，避免了在高负载时带来的系统不稳定；(4) 利用节点任务等待队列和系统平均任务等待量的关系，使接收者找到更合适的任务；(5) 依据数据分布有选择地探询节点，避免盲目探询带来的开销；(6) 由数据分片的冗余等级灵活确定探询次数上限，防止系统崩溃。

本文设计了发送者启动和接收者启动2个子模块：

1) 发送者启动子模块的详细流程

发送方：

(1) $T_V[i]>0$，节点S_i变为发送者，待分配任务为T_g，需访问数据分片D_k。

(2) 在S中查找所有$T_V[j]<0$，且$D_M[j,k]=1$的节点S_j，计算待分配任务T_g从节点S_i分配到节点S_j后产生的代价为：

$$C_j=F_A[j]+P_{g(i,j)}=\sum_{h=1}^{m}(X_M[h,j]A_M[h,j])+\sum_{e,h=1}^{m}\left[T_M[e,h]X_M[e,j](1-X_M[h,j])\right]+C_M(i,j)[C_{\text{size}}(T_g)+R_{\text{size}}(T_g)] \tag{13}$$

根据C_j的大小选择探询节点，找到其任务总开销较小、在系统中引起的通讯开销较小的节点作为接收节点候选者，加快任务执行速度，减少系统开销。

(3) 取符合条件的各个节点中代价数C_j最小的节点S_j作为探询节点，发送探询消息，若从节点S_j返回的响应为$T_V[j]<0$，执行(4)；否则执行(5)。

(4) 将任务T_g从节点S_i转移到节点S_j，重新计算L_i、Q_i、$T_V[i]$，算法结束。

(5) 根据节点S_j的响应同步更新所有其他节点中节点类型向量$T_V[j]$，重复(2)和(3)；选择另一节点作为探询节点，直至满足以下条件之一：① $\forall h\in\{1,2,\cdots,n\}$，$T_V[h]>-1$；② 探询次数$2\times R_V[k]$次；③ $T_V[i]<1$。

接收方：

(1) 节点S_j接收到节点S_i发送的探询消息后，同步更新节点类型向量中$T_V[i]$的取值。

(2) S_j计算节点负载L_j，并取节点类型向量$T_V[j]$返回给探询节点S_i；若$T_V[j]<0$，执行(3)；否则算法结束。

(3) 准备执行转移的任务T_g，置$X_M[g,j]=1$，重新计算L_j、Q_j、$T_V[j]$、$A_A(j)$、$C_A(j)$、$F_A(j)$，算法结束。

2) 接收者启动子模块的详细流程

接收方：

(1) $T_V[j]<0$，节点S_j变为接收者。

(2) 在S中查找所有$T_V[i]>0$的节点S_i作为探询节点，分别发送探询消息，若从节点S_i返回的响应为$T_V[i]>0$，执行(3)；否则执行(6)。

(3) 向节点S_i探询是否有合适的任务T_g从节点S_i转移到节点S_j，若节点S_i返回有任务T_g可转移且代价最小，执行(4)；否则执行(5)。

(4) 准备执行转移的任务T_g，置$X_M[g,j]=1$，重新计算L_j、Q_j、$T_V[j]$、$A_A(j)$、$C_A(j)$、$F_A(j)$，算法结束。

(5) 重复(2)，选择另一个节点作为探询节点，直至满足以下条件之一：① $\forall h\in\{1,2,\cdots,n\}$，$T_V[h]<1$；② 探询次数达到$n$次；③ $T_V[j]>-1$。

(6) 根据节点S_i的响应，同步更新所有其他节点中节点类型向量关于$T_V[i]$的取值，重复(2)，选择另一节点作为探询节点，直至满足以下条件之一：① $\forall h\in\{1,2,\cdots,n\}$，$T_V[h]<1$；② 探询次数$n$次；③ $T_V[j]>-1$。

发送方：

(1) 节点S_i接收到节点S_j的探询消息后，同步更新节点类型向量中$T_V[j]$的取值。

(2) S_i计算节点负载L_i，取节点类型向量$T_V[i]$的值返回给探询节点S_j，若$T_V[i]>0$，执行(3)；否则算法结束。

(3) 在节点S_i的任务等待队列W_i中查找访问数据分片D_k的任务T_g，使$D_M[j,k]=1$，由式(13)计算将任务T_g从节点S_i转移到节点S_j后产生的代价数C_j，根据C_j的大小选择转移任务，将转移后节点S_j的任务总开销较小、引起的通讯开销较小的任务T_g作为可转移任务，若在节点S_i查找可转移任务T_g成功，执行(4)；否则执行(5)。

(4) 向节点S_j发送任务T_g可转移的消息后，任务T_g从节点S_i转移到节点S_j，重新计算L_i、Q_i、$T_V[i]$，算法结束。

(5) 向节点S_j发送无满足条件任务可转移的消息，算法结束。

根据系统总负载和任务等待量来进行选择。当节点接收一个新任务时，首先根据数据的分布情况和节点的当前状态，判断是否需要转移任务。如果不需转移就在本地进行处理，否则查看系统的总体负载情况。当系统中接收者数目比发送者多时，算法自动调用发送者启动子模块，否则等待接收者启动子模块来匹配任务。当节点的任务等待队列长度不小于系统平均等待队长时，强制调用发送者启动子模块一次。另一方面，当节点完成一个任务时，首先判断节点的当前状态，只有当节点变为接收者时，才考虑申请新的任务。当该节点的任务等待队列长度超过系统平均等待队长时，依据数据的分布情况在等待队列中寻找一个本节点能满足其资源需求的任务，否则调用接收者启动子模块从其他节点的等待队列中申请一个合适的任务。

2.3　性能测试

为测试算法性能，从客户机连接到DPSQL服务器，动态增减用户数量，并对其中一些用户的SQL语句平均执行时间进行记录。设定具体测试环境为：9台P4 2G、512 M内存的PC作为DPSQL服务器；15台P3 600 M、128 M内存的PC作为DPSQL客户机；100 M交换机和100 M网卡；测试数据库共10个，以库为单位进行数据分片，每个数据分片的冗余等级为3～5；每个库中有10 000个表，最大表共有5万条记录，每条记录共20个字段，每个字段的平均长度为300 B。根据系统情况，设置算法中的阈值f为0.2。测试结果如表1所示，系统负载程度为当前负载与系统满载的比值。

表1　本文算法与其他算法下**SQL**语句的平均执行时间　　单位：μs

系统负载程度	发送者启动算法	接收者启动算法	混合启动算法	改进的混合启动算法
0.3～0.6	2 415	6 521	2 532	2 063
0.6～0.8	3 280	4 126	3 025	2 646
0.8～0.95	8 452	4 337	4 530	3 460

测试结果表明，本文的算法比现有的其他算法更能保证和优化系统的整体性能，减少用户任务的执行时间。

3　结 束 语

分布式并行数据库系统以其高性能而备受关注，为用户任务选择高效低价的执行节点极为重要。本文描述了一种改进的混合启动自适应任务分配算法，根据系统总体负载变化自适应地选择启动策略，充分考虑任务执行过程中的计算、通讯开销及不同任务对数据的不同需求。性能测试结果表明，该算法能比传统任务分配算法提供更高的任务执行效率和更好的系统稳定性。

参　考　文　献

[1] 袁道华. 分布式系统负载分布研究综述[J]. 计算机科学, 1994, 21(1): 22-29.
[2] 陈华平, 计永旭, 陈国良. 分布式动态负载平衡调度的一个通用模型[J]. 软件学报, 1998, 9(1): 25-29.
[3] 陈华平, 安　虹, 陈国良. 分布式任务调度算法的仿真环境研究[J]. 中国科学技术大学学报, 1999, 29(4): 421-426.
[4] 钱嘉伟. 稳定分布式调度算法中任务的优化分配[J]. 深圳大学学报, 1998, 15(1): 59-63.
[5] 谢树煜, 杨家海. 分布式计算机系统中的通信机制及任务调度算法设计[J]. 小型微型计算机系统, 1995, 16(12): 1-5.
[6] 吴恒山, 张　翼, 李　东. 一个基于分布式数据库系统的动态负载分配算法[J]. 计算机应用研究, 1999, 16(11): 4-6.

编　辑　黄　莘