用于分布式并行数据库系统的重定向算法

左朝树，刘心松，邱元杰，刘克剑，杨 峰

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】分布式并行数据库系统以高可用性、高效率等特征愈来愈受到人们的关注。针对分布式并行数据库系统的特征，提出了分布式并行重定向算法，实现服务器节点的透明切换。该算法不仅能实现负载均衡以及位置透明性，提高系统效率，而且能保证节点故障时事务不被中断，实现系统的高可用性。

关 键 词 分布式并行数据库； 分布式并行重定向； 可用性； 透明性

中图分类号 TP311.133.1 文献标识码 A

A Redirection Algorithm Applied in Distributed Parallel Database System

ZUO Chao-shu，LIU Xin-song，QIU Yuan-jie，LIU Ke-jian，YANG Feng

(School of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract Distributed parallel database system is increasingly concerned because of its characteristics, high-availability, high-efficiency and so on. Distributed redirection algorithm has been presented to accomplish the transparent switch of the node that a client has created connection with, according to the characteristics of the distributed parallel database system. The algorithm has not only accomplished system load balance and location transparency, and enhanced the system efficiency, but also ensured that the transaction don't be interrupted when the node fault occurred，and enhanced the system availability.

Key words distributed parallel database; distributed parallel redirection; availability; transparency

近几年来，国内外学者针对具体的应用领域或环境提出了多种重定向算法[1-3]。文献[1]提出利用服务器返回的目标地址在客户端重发请求实现重定向；文献[2]利用路由器改写数据包的目标地址实现重定向，达到负载均衡；文献[3]利用重定向表记录web页面的位置及数量，并查找能够提供服务的服务器。这些重定向策略都用于无状态服务，不能满足分布式并行数据库系统的有状态服务，如事务操作重定向。为此，本文提出客户端和服务器相结合的分布式并行重定向算法(Distributed Parallel Redirection Algorithm，DPRA)，不仅有助于实现系统负载均衡，而且能够实现有状态服务的重定向，保证故障时事务不被中断，提高系统可用性。

1 系统概述

分布式并行数据库系统是由多台高性能PC通过高速交换机连接在一起构成的数据库服务器群，是能为

收稿日期：2004－04－16
基金项目：四川省科技攻关基金资助项目(02GG006-018)
作者简介：左朝树(1972－)，男，博士生，主要从事分布式并行处理、系统容错、数据库安全方面的研究.

大量用户提供数据库服务的分布式并行系统。它融合网络、数据库、并行处理等先进技术，提供高效率、高可用性、高存储容量的数据库服务，对外表现为统一整体，其拓扑结构如图1所示。每台PC都是自治系统，相互之间不存在主从之分。为了保证高可用性和高效率，系统中每个数据库都存在一定的冗余度r，并按副本分布算法放置在不同的节点[4]。用户请求数据库服务时，客户端通过已知的服务器地址建立与某个数据库服务器的会话，并创建主代理A_m。用户的所有任务都通过这个会话发送到A_m，并根据任务涉及的数据建立相应的辅代理A_p。A_m收到任务T_i后进行分析，并根据任务类型作相应处理，任务号i=1，2，3，…。如果T_i属于写任务，则将T_i发送给所有A_p并行执行，并回送执行结果给A_m，最后A_m汇总处理后将最终结果返回给用户；如果T_i属于读任务，则由A_m选择一个较优A_p，并将T_i发送到该A_p执行，同样通过A_m将结果返回给用户。数据库操作常使用由多个任务组合在一起构成的事务。为了保证原子性、一致性、独立性和耐久性等特性，事务涉及到的所有A_p都要保持事务状态，直到事务结束为止。

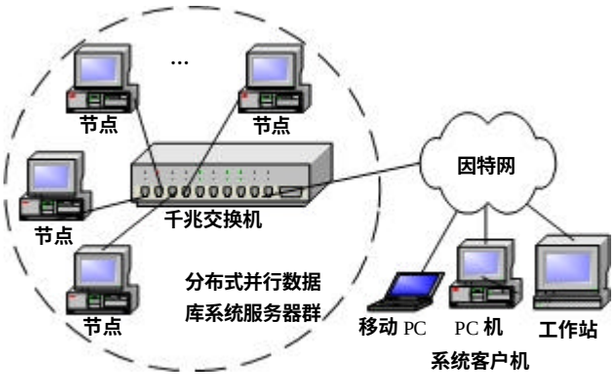

图1　分布式并行数据库系统体系结构

2　分布式并行重定向算法

2.1　算法基本思想

分布式并行数据库系统中，为了实现负载均衡并处理网络或节点故障，重定向算法必须能透明地将用户任务转发给负载较轻或无故障节点，并保证任务及事务正确性。为此，客户端在发送任务以及收到结果时，都要保存足够的状态信息。一旦故障发生，客户端能够根据这些信息与另一特定服务器建立新的会话，并采取相应措施。如果当前任务没有完成，就重发该任务，并重新执行；如果事务在执行过程中，就根据事务执行状态判断需要重新执行的任务，然后将这些任务发送到新的节点重新执行。为了实现负载均衡，一旦A_m发现所在节点负载过重，就在返回结果的同时，请求客户端重定向。整个重定向过程不需要用户参与和干预，完全透明，增加了使用方便性，同时不同客户的重定向过程可以相互独立，并行进行。

2.2　节点优先级

目标节点选择既要考虑数据分布，又要兼顾系统负载均衡，提高系统效率，同时还要避免出现系统抖动。因此，目标节点为拥有该会话A_p的所有服务器节点，用集合N_{Ap}表示。N_{Ap}中，如果完全按照负载大小确定优先级，就可能出现多个负载过重的A_m选择同一个节点N_j，造成N_j负载过重而再次发生重定向，即系统出现抖动。为了避免这种情况，必须引入相应的处理措施。为此，设目标节点N_j的负载为l_j(j =1，2，…，k；k=|N_{Ap}|)，N_{Ap}的所有元素根据负载轻重构成模糊集A，故有：

$$\begin{cases} l_{\min}=\min(l_1,l_2,\cdots,l_k) \\ l_{\max}=\max(l_1,l_2\cdots,l_k) \\ A=\{(m_A(N_j),\ N_j)\mid N_j\in N_{Ap}\} \end{cases}$$

式中　$m_A(N_j)$为模糊集A的隶属函数，根据负载可得$m_A(N_j)=(l_{\max}-l_j)/(l_{\max}-l_{\min})$。

为了避免系统抖动，应尽可能将模糊集A分成每两个元素为一个子集，每个子集按隶属度排序，子集内进行随机排序，从而得到所有目标节点优先级。为了对模糊子集排序，取a=2/k，然后依次取a截集、2a截集等，即可得到根据隶属度排序的多个模糊子集。

2.3　分布式并行重定向协议包括T，A_C，ADDR_DADT和DADT

分布式并行重定向协议(Distributed Parallel Redirection Protocol，DPRP)报文中，T表示数据报类型，长度为3个bit。000为客户端请求建立会话；001为服务器响应客户端的建立会话请求；010为客户端发送任务请求；011为服务器返回任务执行结果；100为服务器请求重定向；101为客户端响应重定向请求；110为服务器创建A_m请求；111为服务器创建A_m响应；A_C表示服务器返回客户端可以重定向的节点数，并决定

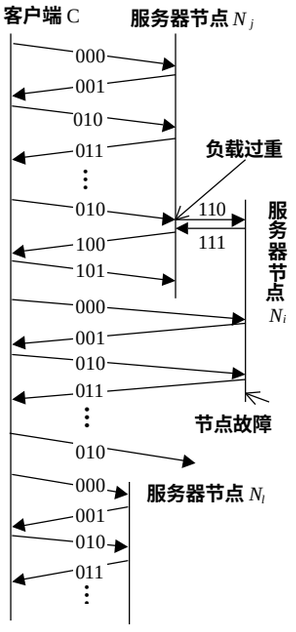

图2 DPRP报文序列

ADDR_DATA的字节数，占5个bit。ADDR_DATA为按照优先级排序的可以重定向的目标节点地址，占4*A_C bit。DATA为任务执行需要传输的数据。在TCP/IP协议栈中，DPRP位于应用层和传输层之间，并对应用层实现透明的重定向。DPRP数据报传输过程如图2所示。图中箭头表示数据传输方向；数字表示数据报类型。客户端C先建立与节点N_j的会话，然后执行用户任务。在任务执行的同时，要根据执行状态缓存任务的相关信息，确保在重定向后任务仅依赖客户端也能够正确完成，并保存A_m返回的已排序的节点表。

在任务执行过程中，有两种情况需要启动重定向过程。1) N_j负载过重。在任务执行的某个时刻l_j过大，则由A_m发出重定向请求并捎带执行结果，选择优先级最高的节点N_i($i\neq j$，且$0<i<k$)创建A_m，并关闭N_j上的A_m。N_i上的A_m等待接收创建会话请求，并建立与特定客户端C的会话。2) N_j节点故障。在客户端C和服务器N_j进行交互的任何时刻，如果检测到节点N_j出现故障，则客户端根据存储的目标节点表，按照优先级选择节点发起重定向请求，直到会话创建成功或最低优先级目标节点会话创建失败为止。执行重定向创建会话后，客户端根据缓存的任务信息，判断任务执行的状态，选择需重新执行的任务，并通过新的会话重新执行，以此保证执行任务的正确性和高效率。

2.4 算法描述

DPRA包括客户端重定向和服务器重定向两部分。客户端重定向负责创建会话、发送任务请求、接收服务器响应、管理目标节点表、缓存任务信息和检测节点故障；服务器重定向负责接收任务请求、确定N_{Ap}优先级、返回执行结果、判断负载是否过重、发起重定向请求并创建新的A_m。

2.4.1 客户端重定向算法

1) 创建会话，如果有需要执行的缓存任务则执行，如果失败，则转至7)；2) 发送任务请求，启动定时器，并缓存该任务；3) 如果定时器超时，没有收到执行结果，则转至7)；4) 终止定时器，向用户返回执行结果，如果报文中捎带有目标节点表则刷新缓存的节点表；5) 处理缓存任务，如果缓存的任务与后续任务耦合度为0，则清除缓存；6) 判断报文类型，如果服务器请求重定向，则转至7)，否则转至2)；7) 从目标节点表中选择最优的未标记的节点，并标记该节点，转至1)。

2.4.2 服务器节点N_j的重定向算法

1) 等待接收用户任务；2) 判断负载是否超过阈值，如果超过阈值，则选择最优节点N_i请求创建新的A_m；3) 执行用户任务，组装重定向协议数据报，如果目标节点优先级已改变，则捎带目标节点地址；4) 判断是否请求创建新的A_m，如果无请求，转至1)；5) 将N_i上A_m的状态置为与当前A_m一致，并结束当前A_m。

3 性能分析

为了说明DPRA的性能，在此分析平均重定向时间，并假定所有任务执行时间相同，用t_e表示；客户端与任何一个节点创建会话的时间相同，用t_s表示，以简化分析过程。

根据算法设计可以看出，DPRA有节点负载过重和节点故障两种触发机制。节点负载过重时，N_j上的A_m在N_i上创建新的A_m，新的A_m拥有当前任务的执行状态，因此不需要重新执行客户端缓存任务，此时平均重定向时间为定值，即创建会话的时间t_s。对于节点故障，平均重定向时间计算较复杂，为了描述方便，用b表示缓存的任务数、t_f表示故障检测时间、R表示节点可靠度、n表示系统节点数。由于每个节点都可能出现故障，因而在执行缓存任务时A_m所在的节点也可能出现故障。按概率可以得到节点故障时的平均缓存任务数为：

$$b' = Rb \tag{1}$$

而平均重定向时间为：

$$\overline{t} = t_f + (t_s + b't_e + t_f)\sum_{i=1}^{n-1}(1-R)^i \tag{2}$$

将式(1)代入式(2)可以得到：

$$\bar{t}=\sum_{i=1}^{n-1}(1-R)^i t_s+Rb\sum_{i=1}^{n-1}(1-R)^i t_e+(1+\sum_{i=1}^{n-1}(1-R)^i)t_f \tag{3}$$

根据式(3)可得，平均重定向时间与执行时间、会话创建时间、故障检测时间及缓存任务数等密切相关，特别是执行时间和缓存任务数对平均重定向时间的影响最大。在分布式并行数据库中，大多数情况下 b=1，只有在事务状态时才会有 b＞1 的情况。因此，在一般情况下，平均重定向时间较短，系统可用性非常高。

4　测试及分析

为测试可用性，设置数据的冗余度为3。通过断开网络、关闭服务器等方式，人为制造节点故障，测试DPRA的正确性；通过增加会话数以及执行任务数，增加节点负载，测试负载过重情况下DPRA的正确性。经多次测试表明，不论节点故障还是负载过重的重定向，结构式询问语句(Structured Query Language，SQL)都能正确执行。为了测试重定向时间，利用测试程序多次模拟测试上述情况。在事务和非事务状态下，测试程序执行相同的SQL以模拟相同执行时间，记录每次重定向时间和缓存SQL数。通过对不同SQL以及不同数目缓存SQL的测试，对测试数据进行分析和整理后得到图3和图4所示的曲线。

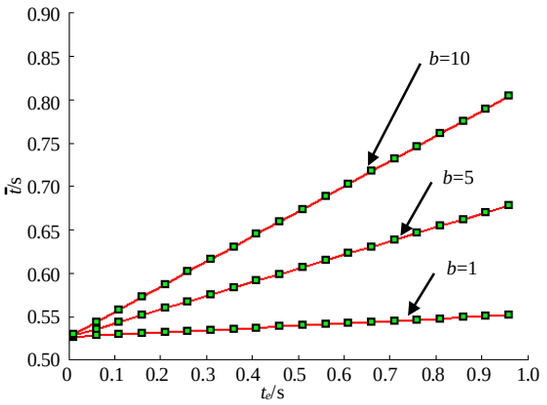

图3　$\bar{t}$ 与t_e的关系

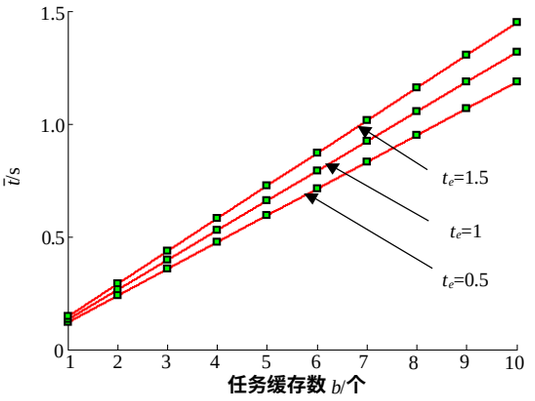

图4　$\bar{t}$ 与b的关系

从图4和图5可看出平均重定向时间$\bar{t}$ 与任务执行时间t_e、任务缓存数b成正比，且b对$\bar{t}$ 的影响最大，其原因是任务缓存数越多，重定向时需要重新执行的任务越多，花费的时间越长；执行时间越长，完成缓存任务的时间就越长。

5　结 论

在分布式并行数据库系统中，通过采用分布式并行重定向算法不仅实现了系统的负载均衡，提高了系统的效率，而且解决了节点故障问题，达到了节点故障透明，特别是故障情况下保证了有状态服务即事务的正确完成。

参 考 文 献

[1] Wenting T, Mutka M W. Load distribution via static scheduling and client redirection for replicated Web servers[C]. In: Proc. 2000 International Workshops on Parallel Processing, Toronto, Canada, 2000. 127-133

[2] Cardellini V, Colajanni M, Philip S Y. Redirection algorithms for load sharing in distributed web-server systems[C]. In: Proc. 19th IEEE International Conference on Distributed Computing Systems, Austin, Texas, 1999. 528-535

[3] Suryanarayanan K, Christensen K J. Performance evaluation of new methods of automatic redirection for load balancing of Apache servers distributed in the Internet[C]. In: Proc. 25th Annual IEEE Conference on Local Computer Networks, Tampa, Florida, 2000. 644-651

[4] 沈海华, 陈世敏, 沈美明, 等. WWW集群服务器的数据副本分布方式研究[J]. 软件学报, 2001, 12(3): 367-371

编　辑　熊思亮