计算机科学 2001Vol. 28№. 8

基于网络集群系统的并行数据库中数据均衡分布研究

Analysis of the Data Balanced Distributing in Parallel DB in COW

杨全胜　徐宏炳　王能斌

（东南大学计算机科学与工程系　南京 210096）

Abstract　Base on the character of the cluster of workstation(COW)and the latest development of the parallel computer, this paper analyzes the data deflexion problem in data distributing of parallel DB in COW. On the basis of this analysis, we get a dynamic data balanced distributing algorithm which has adaptability.

Keywords　Cluster of workstation(COW), Parallel database, Data deflexion

1　引言

随着信息处理技术的不断发展，信息量的急剧膨胀，对数据库系统提出了更高的要求；并行计算机系统的快速发展，使得并行运行环境逐渐趋向成熟，广泛应用于科学计算与研究的众多领域，并行数据库的研究也越来越受到广泛的关注，并已经成为数据库研究中一个重要的研究领域。

并行数据库系统中，数据偏斜问题一直受到大家的重视，因为数据分布的不均衡，会影响整个系统的性能，为此，有不少对数据均衡及数据重组问题的研究。但大多数研究基于 MPP 或者 SMP 型的并行计算机模型，采用一种均分或者类似均分的方法，这种方案对于网络并行集群系统并不一定合适，由于目前大多数网络并行集群中通信延迟问题是不容忽视的，数据均分可能会造成新的数据偏斜，本文将从网络并行集群的特点以及并行系统新的发展的角度提出一种具有一定的环境适应能力的动态数据均衡分配方案。

2　系统运行的环境分析

并行数据库的研究，离不开其运行的特定的并行系统环境，不同结构的并行系统，会使得并行数据库在数据分布、查询优化等各个方面的设计上有不同的特性，为了研究在网络并行集群下的并行数据库数据均衡分布的问题，我们简单分析一下网络集群系统的特点。

网络并行集群系统是通过高速网络，连接数台工作站、高档微机甚至 SMP，构成一个并行运行环境。这种系统投资少、见效快、系统有很灵活的扩展能力，是九十年代中期并行机研究的热点之一。从并行系统的体系结构发展来看，目前可以将并行系统分成四种基本结构，第一种是 Share-Everything(SE)结构，第二种是 Share-Disk(SD)结构，第三种是 Share-Nothing(SN)结构，第四种是 Distributed Share Memory(DSM)结构。目前大多数网络集群系统属于 Share-Nothing 结构，这种系统由高速网络连接多个超级结点机，每个超级结点机带有一个或几个处理器，并有自己的内存和磁盘组，但由于结点机与结点机之间的数据传送速度比结点机内部处理机与处理机之间的传送速度要慢，因此它与一般分布式存储的 MPP 还是有着明显的不同，在数据分布的时候我们既要注意由分布带来的任务并行执行的好处，又要注意由于网络速度问题而引起的大量数据移动的代价问题。另外从发展的角度来看，由于网络技术的不断发展，我们有可能在不久的将来使网络通信速度提高到一个非常快的程度，实际上目前已经有这方面的新进展的报道，这时，在网络集群系统下就可以实现分布式共享存储结构(DSM)。同时，网络并行集群系统有很大的灵活性，其规模是可伸缩的，通信性能又是可以不断升级的，同时随着各结点机主存的扩大，一个大内存的系统会对系统的性能带来巨大的变化，正是由于网络并行集群系统处于发展状态，因此我们在考虑数据分布时，不能仅仅考虑一种结构模式，而应该考虑其在体系结构发展中的适应性。

3　数据分布方案

以下的讨论我们基于同构网络并行集群系统，系统中的超级结点机采用 SMP 机，而同一个结点机中的各 CPU 共享存储器，同时磁盘采用 RAID 技术，并能同时为几个处理器服务，在系统中有一台主机负责控制整个系统的工作、数据的分布和任务的分配都由主机进行，其它的结点机进行具体的数据处理，并有如下

约定：

1. 系统中超级结点机数为N。分别记为J_1、J_2、J_3、…、J_n，J_1是主机。

2. 各超级结点机中的CPU数为M。记为C_{11}、C_{12}、…C_{1M}、C_{21}、…、C_{i1}、…、C_{nm}，其中$1\leqslant i\leqslant N$，$1\leqslant j\leqslant M$。C_{ij}是第i个结点机上的第j个CPU，我们将各超级结点机的存储器划分成M块，每块对应一个CPU，所以C_{ij}也表示第i个结点机上的第j块存储区域。

3. 数据记录的总数为K，每个记录假定为100字节。

4. 设每个结点机主存储器的可使用容量为D个记录容量。

5. T_{MOVE} 一个记录从C_{ik}移到C_{jq}($i\neq q$)所需的时间。T_{NETCOM} 一个记录从J_i到J_k($i\neq k$)所需的时间，T_{IO} 从磁盘读一个记录的时间。

由于T_{MOVE}、T_{NETCOM}和T_{IO}之间存在着差异，当主存储器容量足够大时T_{IO}所占比重相应会减少，这时数据的操作大部分情况下是在主存之间或是网络上交换，所以不可忽略T_{NETCOM}的延迟带来的数据分布上新的偏斜情况，均分数据可能会使得非主机站点在收发数据和处理数据上负载增大。另外，如果参与操作的超级结点机数目增加，在任务并行度提高的同时，网络上传输的代价也在提高，究竟两者增长的规律是什么？用多个结点机并行工作，能否抵消由于网络传输而带来的新的开销？为此，我们可考虑在分布数据的时候，为主机多预留P个记录，将K-P个记录均分到其它结点机上，这一节我们就这一思路给出相应的分配数据算法。在下一节，我们将讨论系统规模发生变化时，究竟需不需要通过预留来提高性能，如何确定更佳的结点机数N'。

一旦确定下P、N'之后，就可以具体实施数据的分配了。数据的分布分为数据初始分布和数据的均衡调整，它们依据的均衡准则都一样，但是具体操作上会有所区别。

初始数据分布是在数据库建立的初始阶段。

算法1

Step1 根据系统当前的性能情况，确定相应的P与N'；当P接近K时转到Step3。

Step2 将K-P的数据均分成N'等份，并将它们分到各个超级结点上。

Step3 将各结点机上的数据均分到各个存储器区域中。

Step4 各结点机将数据分配情况通知主机，主机作好记录。

当数据库系统运行了一段时间以后，或是由于数据规模变化，或是由于系统规模、结构发生变化会使得数据再次出现偏斜，此时就要进行均衡调整。

该算法的主要开销是第二步的数据通信和第三步的数据上盘工作。就某一具体的操作来说，当网络速度增加时，第四步的代价将会不断缩小。由于是初始分布，所以所有数据都必将存到磁盘上，因此第三步的IO代价也将会很大，而且可能是该算法的主要开销。

算法2

Step1 根据系统当前的性能情况，确定相应的P与N'；当P接近K时转到Step5。

Step2 将K-P的数据均分成N'等份，得到K_1、K_2、K_3、…、K_N。

Step3 各结点机将目前数据分配情况通知主机，主机获得各机数据量K'_1、K'_2、K'_3、…、K'_N。令$T_i=K_i-K'_i$。

Step4 将各结点机将数据重新调整，使得其数据量为K_i。

Step5 将各结点机上的数据均分到各个存储器区域中。

Step6 各结点机将数据分配情况通知主机，主机作好记录。

其中第4步可以采用数据平衡的记录移动法[3]。算法2的主要开销是第四步的通信与第五步的存盘的IO开销，但与算法一相比较，由于它移动数据量要少得多，所以其开销也要小得多。

这两个算法并不很复杂，但如何确定N'以及是否需要预留P个记录是很关键的。下面就这个问题做一些讨论。

4 预留数据和参与运算结点的确定

上一节中我们给出了数据初始分布和均衡调整的算法，算法本身并不难，但是算法性能的好坏决定于是否要预留以及参与运算的结点机数N'有没有限制等。P与N'值的指定可以通过T_{MOVE}、T_{NETCOM}、T_{IO}的比例关系，以及M和T的数量来近似获得。数据库操作中主要是选择、排序、连接等操作，我们可以根据上节的分配方案分别为这些操作定义预留数据的并行操作算法，通过对算法的模拟分析来近似求出N'的取值及确定是否预留，然后根据以上几种操作在数据库管理系统中所占操作的频度，综合出最终的N'值和预留方案。

为了说明问题，下面以网络并行集群系统中的合并排序为例，说明一下对N'值的确定和预留判断。

假设主机在分配数据前先预留P个记录，将$K-P$个记录在各结点机中平均分配，这样各从结点机上有$\frac{K-P}{N}$个记录，其中D个在主存中，$\frac{K-P}{N}-D$个记录在磁盘中，初始状态每个CPU负责$\frac{K-P}{MN}$个记录，

主机上有$\frac{K-P}{N}+P=\frac{K+(N-1)P}{N}$个记录初始状态下主机各CPU负责$\frac{K+(N-1)P}{MN}$个记录。

算法3

Step1 主机将各从机的数据通过网络分配到各个从机上。

Step2 各结点机将自己的数据按CPU数M均分到存储器中。

Step3 各个CPU利用排序算法将自己的数据排序。

Step4 各结点机用多个CPU对自己机器上的数据利用并行多路合并排序算法合并排序，结果集中到一个缓冲器中。

Step5 全系统的各结点机利用并行多路合并算法将全部数据合并到主机上，算法结束。

这里的多路合并算法是通过多次两两合并来实现的。为了讨论简单，假设K、M、N均为2的幂。具体算法描述请参考文[4]中的讨论，这里我们将磁盘IO代价也考虑进去。

如果每台机有E个记录大小的主存容量，令

$$D=\begin{cases}E & (E\leqslant\frac{K-P}{N})\\ \frac{K-P}{N} & (E>\frac{K-P}{N})\end{cases}$$

而当$D=\frac{K-P}{N}$时，说明此时对于主机尚有主存空间可以存放预留的P个数据中的部分或全部，所以令$D_1=\begin{cases}E-D & (E-D\leqslant P)\\ P & (E-D>P)\end{cases}$，另外，令$D_0=\frac{K-P}{N}-D$；

Step1的代价为$\frac{(K-P)(N-1)}{N}\times T_{NETCOM}+K\times T_{IO}$

Step2的代价为

$$\frac{(D+D_1)\times(M-1)}{M}\times T_{MOVE}+\frac{(D_0+(P-D_1))\times(M-1)}{M}\times T_{IO}$$

Step3的代价为

$$O(T_{COMP}\times\frac{K+(N-1)\times P}{MN}\log_2(\frac{K+(N-1)\times P}{MN})+T_{MOVE}\times\frac{D+D_1}{M}+T_{IO}\times\frac{D_1+(P-D_1)}{M})$$

Step4的代价为

$$O(T_{COMP}\times\frac{K+(N-1)\times P}{N}\log_2 M+\frac{(D-D_1)\times(M-1)}{M}\times T_{MOVE}+\frac{(D_0+(P-D_1))\times(M-1)}{M}T_{IO})$$

Step5的代价为

$$T_{COMP}\times K\times\log_2 N+T_{NETCOM}\times\frac{(N-1)\times(K-P)}{N}+T_{MOVE}\times((\frac{N-1}{N})\times D\times N+D_1\times\log_2 N)+T_{IO}((\frac{N-1}{N})\times((K-P)-N\times D)+(P-D_1)\times\log_2 N)$$

其中T_{COMP}是CPU完成两个数据比较所用的时间。

为了能够说明问题，我们将该算法和均分数据算法进行比较，有如下的一些图形：

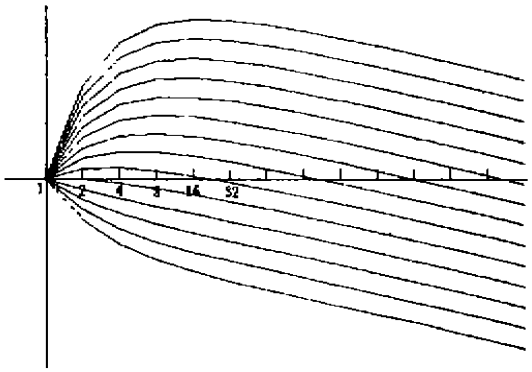

图1

图1中X轴表示N的变化，分别为1，2，4，8，16，…，2^{14}。图上曲线从下到上分别表示T_{NETCOM}和T_{MOVE}比值分别为1，2，3，4，…，14，k＝10000个记录，每个记录100字节，P＝500记录，M＝4，各结点机可用内存容量为1M，这时整个系统的主存容量足够数据存放，所以这种情况下磁盘读写对性能基本没有影响。在这种情况下以比值为6时为例，我们注意到，结点机数在2～16之间预留都有好处，但此时8个结点机获得的性能最好，8就是在这种情况下采用预留方案最佳结点机数，而一旦结点机数超过16，整个系统由于并行结点的增加，抵消了网络延迟带来的额外开销，此时，采用均分数据法会得到更好的性能。那么当主存容量不够时情况又如何呢？

图2

图2中X轴仍然表示N的变化，分别为1，2，4，8，16，…，2^{14}。图上曲线从下到上分别表示T_{NETCOM}和T_{MOVE}比值分别为1，2，3，4，…，14，k＝100000个记录，

每个记录 100 字节，P＝500 记录，M＝4，各结点机可用内存容量为 IM，由于当 N＜16 时系统的主机内存容量不足，因此部分数据需要在磁盘中存储，而 IO 的代价相对网上代价和内存访问代价要大得多，此时预留显然增加了这一部分的开销，均分以后使得数据集变小，又可以并行处理，从而降低 IO 代价，此时宜采用均分法。当 N＞16 以后，由于系统总主存容量大于数据量，此时又呈现出图 1 中讨论的情况，在一段范围内，预留数据将比较好。

另外，通过这两幅图我们还注意到网络速度的提高对网络并行数据库有着很重要的作用，一旦网络速度足够快，或者在 DSM 结构下，预留数据就没有必要了。所以我们认为，预留数据的数据分配方案是在目前网络速度不够快的情况下，利用网络并行环境设计并行数据库时，减少数据分配时数据偏斜的一种权宜之计。实际操作中，我们应该根据网络速度、总内存容量、以及系统中最多可用结点数等因素，合理地确定采用预留数据法还是采用均分数据法，在预留数据法中，再根据相应的计算，确定参与运算的结点机数。

结论 通过上述讨论我们提出了一种在网络并行环境下数据均衡分配的方案。该方案考虑到了目前网络速度与内存存取速度之间的差异带来的数据偏斜问题，提出一种预留数据的分配方案，并经过进一步的探讨，说明在一定的特定条件下，预留数据分配方案确实能收到较好的性能。同时，文中还讨论了在网络并行环境的规模、网络速度、总内存容量变化的情况下，如何合理选择数据分配方案的问题。

参 考 文 献

1 廖朝晖，张鹏，王珊. 并行数据库系统：目标、体系结构及研究方向. 计算机科学，1994，21(6)，23～28

2 Hua K A，Su J X W. Dynamic Load Balancing in Very Large Share-Nothing Hypercube Database Computers. IEEE Trans. On Computer，1993，42(12)：1425～1439

3 金树东，冯玉才，王元珍. 并行数据库系统的数据重组研究. 小型微型计算机系统，1998，19(3)：28～33

4 杨全胜，徐宏炳，王能斌. 网络并行集群中的多路合并算法研究. 计算机工程与科学，1997，19(A1)：94～97

5 Wen Zhaofang. Multiway Merging in Parallel. IEEE Trans. Parallel and Distributed Sysrem，1996，7(1)

6 孙家昶，张林波，等 网络并行计算与分布式编程环境. 北京：科学出版社，1996

7 陈数清. 并行计算机的现状与发展趋势. 中国计算机学会，当代计算机体系结构、操作系统的发展学术研讨会，1996

（上接第 41 页）

7 Pinkerton B. Finding What People Want. Experiences with the WebCrawler. In：Second Intl. WWW Conf.'94，July 1994，Chicago，USA，Oct. 1994. http. // info. webcrawler. com/bp/WWW94 html

8 http. // www. searchenginewatch. com. webmasters/rank. html

9 张卫丰，徐宝文. Web 搜索引擎框架研究. 计算机研究发展，2000，37(3)，376～378

10 张卫丰，徐宝文，周晓宇. Web 页面中的计数器研究. 小型微型计算机，2000，21(10)，1096～1099

11 张卫丰，徐宝文，周晓宇. Web 页面中元素间交互技术研究. 计算机工程，2000，26(8)：62～64

12 张卫丰，徐宝文，许蕾 Web 页面安全性技术初探. 计算机工程与应用，已录用

13 张卫丰，徐宝文，许蕾. 利用 Agent 个性化搜索结果. 小型微型计算机，已录用

14 张卫丰，徐宝文，周晓宇，管宇，许蕾. 基于遗传算法的搜索引擎调度，待发表

15 张卫丰，徐宝文，周晓宇，许蕾 带反馈自适应 Web 搜索引擎研究，待发表

16 张卫丰，徐宝文，周晓宇，许蕾，李东，元搜索引擎结果生成技术研究，待发表

17 张卫丰，徐宝文，周晓宇，李慎之，许蕾. 数据挖掘技术在 Web 预取中的应用研究，待发表

18 Zhang Weifeng，Xu Baowen，Yang Hongji，Chu William C. A Genetic Algorithm Based General Search Engine. In：Proc. of IEEE MSE'2000

19 Zhang Weifeng，Xu Baowen，Chu William C，Yang Hongji. Data Mining Algorthms for Web Pre-Fetching. In. Proc. of The Workshop on the World Wide Web Semantics (WebSem'2000)

20 Xu Baowen，Zhang Weifeng，Chu William C，Yang Hongji. Application of Data Mining in Web Pre-Fetching. In. Proc. of IEEE MSE2000