并行数据库 数文据库 结点故障 对策 (10)

计算机科学1995 Vol. 22№. 5

42-45

无共享并行数据库中结点故障对策

昌月楼 阳国贵

(国防科技大学计算机系 长沙410073)

TP311.13

摘 要 Several methods for dealing with the problems of node failure in shared-nothing parallel database systems are presented in this paper. Also is given the analysis of difference in these methods, concerned with system availability and load balance. It is concluded that the chained declustering method is superior to the others in the two respectives.

关键词 Parallel processing, Database, Fault tolarence, Load balance.

1. 问题的提出

随着信息时代的出现，人们对数据库系统的要求越来越高，既要求它能提供快速联机响应和大的吞吐量，巨大的存贮容量；又要求它能提供高度的容错能力，即在故障情况下能继续工作。

为满足性能方面的要求，人们一方面在传统数据库的基础上采用尽可能先进的平台和研制高效的数据库管理系统；另一方面把目标瞄准了并行数据库，让多处理机同时并行处理同一查询或事务以提高响应速度和加大吞吐量。国外在这方面的研究已开展了十多年了，出现诸如 MCC 的 Bubba[1]，UW 的 Gamma[2]，Tandem 的 NonStop SQL[3]，Teradata 的 DBC/1012[4]的等原型机，以及 nCUBE2上的 Oracle[5]这样的商用系统。国内的研究则处于起步阶段，象曙光、银河等均在朝此方向努力。

并行数据库系统由多个处理机组成，其硬件结构有共享主存、共享磁盘和无共享三种典型模式[6]。由于无共享结构具有系统伸缩性好、对互联网的带宽要求较低等优点，所以目前多数专家都推崇无共享模式，象前面提到的 Bubba，Gamma，NonStop SQL，DBC/1012，nCUBE2/Oracle 等都属于此类。随之而来的要求是，在无共享并行数据库系统中，怎样才能保证该系统的容错能力。即是说，当结点出现故障时，用什么办法使系统继续工作而不致于瘫痪下来。有一些文章把这一问题称之为数据库系统的高可用性(High-availability)[7]问题。

2. 几种结点故障对策

本文论述的结点故障是指无共享的多处理机模式中某结点出现的磁盘或处理机的致命性故障，即是说这些故障引起某结点瘫痪。而要研究的对策是在此不幸情况下怎样才能维持全系统继续工作。此外，下面提到的方法与硬件的拓扑结构无关，即各结点的互联方式可以是总线方式，超立方方式，交叉开关方式，等等。

在并行数据库系统中，对付结点故障的容错手段有两类：一类是数据副本方式；另一类是冗余检验方式。前者是将同一数据形成两个拷贝，任何情况下这两个拷贝必须相同，一个叫主拷贝，一个叫副拷贝。系统正常工作时用主拷贝，出现故障时启用副拷贝。后者只有一份数据，但有一个冗余磁盘保持各盘数据的检验码，出现故障时用无故障盘和检验盘重新产生故障盘的数据。

为了获得高的加速比，一般是将一个表的记录水平分割后散布在各结点的磁盘上进行并行处理，水平分割表的方式主要有三种：循环分割，范围分割和哈希分割。循环分割是每次将 n 个记录依次存放到 n 个磁盘上，循环到全部记录分布完毕；范围分割是一次性将整个表根据划分属性值的范围分成 n 块存放到 n 个盘上；哈希分割是根据划分属性值用 Hash 函数将记录映射到 n 个盘上。下面论述的方法对这三种分割方式都是可行的。

2.1 副本方式

昌月楼 副教授，主要研究领域为数据库、软件工具、并行处理等系统软件。阳国贵 讲师，目前主要研究领域为数据库与知识处理。

这类容错手段又可分为数据镜象，交叉分布和链式分布三种模式。

2.1.1 数据镜象。在单处理机系统中也存在数据镜象。在那里，当主磁盘出现故障时系统启动备份磁盘；但是，如果主机出现故障系统就瘫痪了。在多处理机情况下，要求既要对磁盘容错，也要对处理机容错，所以镜象方式有所不同。图1是 Tandem 的 NonStop SQL 所采用的镜象方式。其中假设表 R 只被分割在两个结点上，R1和 R2为主拷贝，r1和 r2为副拷贝。

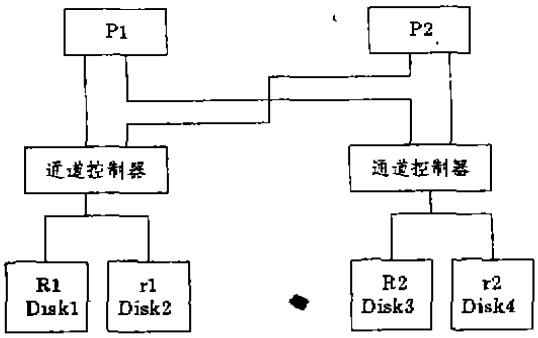

图1 Tandem NonStop SQL 的镜象磁盘结构

这种结构下，磁盘容错是很直观的：假设 Disk1 有故障，则马上可以启动 Disk2继续工作，同时对 Disk1进行"热"替换。为了做到处理机容错，对于每个结点的磁盘访问有两个进程位于两个处理机上，即 P1和 P2上都有访问本结点的 I/O 进程和他结点的 I/O 进程，这样，其中一个处理机有故障时另一个可以"接管"它的工作。当然，对于通道故障，也可用双通道备份的方式进行容错。这种模式在一个磁盘失效时对全系统性能的影响并不明显，而在一个结点（处理机）失效时全系统性能的影响是很糟的，因为另一结点承担了双倍的工作，引起了严重的负载不均。

2.1.2 交叉分布。在交叉分布模式中，一个表的主拷贝以某种水平分割方式分散存放在 n 个结点上，而副拷贝的分布则有所变化。某一个结点上相应于该结点主拷贝的副本再被分成(n－1)个部分分散存放在其余(n－1)个结点上。每个结点都是如此，形成所谓的"交叉"分布。假设 n＝4，则表 R 的交叉分布如图2所示，其中 Ri 为主拷贝，ri. j 为副拷贝，即 ri. j 是 Ri 再分割以后的部分。

结点号	0	1	2	3
主拷贝	R0	R1	R2	R3
副拷贝		r0. 0	r0. 1	r0. 2
	r1. 2		r1. 0	r. 1. 1
	r2. 1	r2. 2		r2. 0
	r3. 0	r3. 1	r3. 2	

图2 交叉分布

这是 Teradata 公司所采取的容错方式。它的一个差别是把这样一组结点叫做一簇，而一个表不只是分布在一个簇中，可以分布在多簇中；另一个差别是一个结点的磁盘可以有多个，但当作一个逻辑盘来处理。

这种模式的一个优点是在任意结点失效的情况下，同一簇中的其余活跃结点可以均摊失效结点的工作，例如结点2失效，由于它的副拷贝 r2. 0，r2. 1，r2. 2分别存放在结点3、结点0和结点1上，从而这些结点可以分担结点2的 I/O 操作。所以这种模式在一个结点失效情况下能很好地均衡负载，这是它优于数据镜象的地方。

2.1.3 链式分布。从图2可以看出，在交叉分布模式中，同一簇中任意两个结点失效将导致系统瘫痪。而在数据镜象模式中若将2n 个节点按图1的方式构成 n 个镜象对，将一个表分布在这 n 个镜象对中，也不存在任意两个结点失效而导致系统瘫痪的问题，只有同一镜象对中的两个结点失效时才会如此。但它又存在一个结点失效时负载不均的问题。

链式分布则可以吸取上述两种模式的优点，一方面，它解决了一个结点失效时的负载不均问题；另一方面，它不是任何两个结点失效都会导致系统瘫痪。链式分布中一个表在 n 个结点上的主副拷贝的分布错开一个结点的位置，其名称也由此而来。图3给出了 n＝4的链式分布结构，即假设表 R 分布在4个结点上。

结点号	0	1	2	3
主拷贝	R0	R1	R2	R3
副拷贝	r3	r0	r1	r2

图3 链式分布

假设结点1失效，乍一看来，R1的 I/O 负载会全部落在结点2上；但是，如果调整一下活跃结点的负载，则可做到负载均匀，如图4所示。

结点号	0	1	2	3
主拷贝	R0	—	1/3R2	2/3R3
副拷贝	1/3r3	—	r1	2/3r2

图4 结点1失效后的负载调节

这样，在任意结点失效的情况下，对于n个结点的系统，其余活跃的每个结点都只增加1/(n－1)的负载。在图4的例子中，结点0的I/O操作除了原来的R0以外增加访问1/3r3，结点2的I/O操作是访问1/3R2和r1，结点3的I/O是访问2/3R3和2/3r2。这样，表R的各部份均能访问到，并能维持并行处理，且能做到负载均匀。

还要注意的是，这种一个结点失效后活跃结点负载的调整不需要数据在结点之间进行迁移。因为每个结点都有正常情况下的主副拷贝部份，在故障情况下要某活跃结点只访问某拷贝的一部份时，只要改变常驻内存控制表中的一些边界值和指针或索引即可。例如，正常读操作时，结点0只访问R0(图3)，而在故障情况下，结点0除读R0外，还要读1/3r3(图4)，而r3原来已在结点0上，不存在数据迁移。

从图3还可以看出，只有当相邻两个结点失效时才会丢失表中的数据从而使系统瘫痪。除此以外的任意两个结点失效均不会引起系统瘫痪，因为表的各部份数据都还完整地保留在活跃结点中。交叉分布和链式分布都可以通过增加结点数来减轻在一个结点失效时活跃结点的负载，但随着结点数的增加，任意两个结点同时失效的机会增加，因此加交叉分布系统的可用率比链式分布系统要低。

2.2 冗余校验方式

这是RAID(Redundant Array of Inexpensive Disks)磁盘阵列技术中采用的方式。在这种情况下，每个结点使用的是一组价格低廉的小磁盘，以高带宽的互连网连接，称作一个磁盘阵列。其中有一个检验盘，其余为数据盘。结点的数据依次存放在数据盘上，而检验盘存放的是各数据盘的校验数据。正常I/O操作访问的是数据盘，当数据盘中有一个出现故障时利用活跃的数据盘和检验盘的数据来恢复丢失的数据。

建立校验数据和恢复丢失数据的原理是这样的：假设A1，A2，…，An和B均为一个字节的数据，用(XOR)表示按位的异或操作，则有如下规则：

若 $B=(\overset{n}{\underset{i=1}{XOR}})Ai$

则 $Aj=[(\overset{n}{\underset{i}{XOR}})Ai](XOR)B \quad j\in[1,n],i\neq j$

例如，A1＝01101101，A2＝11000011

则 B＝A1(XOR)A2＝10101110

于是 A1＝A2(XOR)B＝01101101

A2＝A1(XOR)B＝11000011

如果将数据A1，A2，…，An放在n个数据盘上，将数据B(异或结果)放在检验盘上，则当任何一个数据盘j(j＝1，2，…，n)失效时，可利用余下活跃的(n－1)个数据盘以及检验盘上的数据进行异或操作来恢复盘j的数据。

这种方式的优点是没有数据副本，节省磁盘空间；缺点是在正常写操作和故障情况下读操作均需要作异或运算，多用了CPU时间。当然，随着CPU成本的降低，可用专门的CPU来做这件事，甚至把它集成在RAID系统中。

显而易见，RAID系统的检验盘是至关重要的。如果检验盘失效，则任何数据盘的数据也不能恢复。因此，在实际系统中，检验盘不是固定在一个盘上，而是轮流使用各盘。假设有n个盘组成一个磁盘阵列，则在写数据时前(n－1)个扇区分散在前(n－1)个盘上，第n盘为检验盘；下面再写数据时将第(n－1)盘作为检验盘，数据扇区写在盘1，盘2，…，盘(n－2)，盘n上；再写时将第(n－2)盘作为检验盘……，以此类推。图5给出了n＝4的示意图。

磁盘号	0	1	2	3
BLOCK0	S0.0	S0.1	S0.2	///
BLOCK1	S1.0	S1.1	///	S1.2
BLOCK2	S2.0	///	S2.1	S2.2
BLOCK3	///	S3.0	S3.1	S3.2
BLOCK4	S4.0	S4.1	S4.2	///

图5 4级RAID示意图

其中每三个扇区组成一块(BLOCK)，Si.j表示第i块的第j扇区。阴影区表示检验数据。这样做的好处是可以分散风险。如果将一个专用盘作检验盘，则该盘失效时系统全部数据失去恢复功能。在图4的情况下，某一盘失效时，只有部份数据失去恢复功能。例如盘1失败，则只有第2块数据失去恢复功能，而盘1上的数据可通过其他盘得以恢复。理论上，可以通过增加盘的个数来减小数据丢失的风险和提高I/O操作的并发度。但是，随着磁盘的增加，连接和控制的复杂性又会增加非介质因素的故障率，这是互相牵制的。目前最高的是5级RAID系统。

3. 结束语

上述几种容错办法都能解决一个盘或一个结点

软件需求定义语言 NDRDL

董丽君 费宗铭 朱 鸿 金凌紫
(南京大学计算机软件研究所 南京210093)

摘 要 NDRDL 语言是一种图形化的软件需求定义语言，用于书写软件需求定义。其特点是：形象直观、表达力强、实用性好、可靠性高。本文是该语言的试用文本。

关键词 软件需求定义，软件需求定义语言，图形化。

一、引 言

八十年代以来，南京大学计算机软件研究所对从软件功能规约到可执行代码全过程的自动化技术进行了较为系统的研究，并取得一定进展。先后设计了软件规约语言 GSPEC 和 FGSPEC，研制了 NDAUTO、NDTPS、NDADAS、NDIPS、NDSAIL 等软件自动化系统。然而，从非形式的软件需求定义到形式的功能规约的自动转换仍是软件自动化的一大难题。为此，我们致力于这方面的研究与开发工作，研制基于功能分解的软件需求支撑系统 NDRASS，旨在辅助分析人员实现从需求定义到功能规约的转换。同时，为了书写软件需求定义我们设计了软件需求定义语言 NDRDL。

NDRDL 以结构化数据/功能分析方法为基础，将待解的问题抽象成一系列需要具备的功能，通过表述各功能之间的数据流向、控制流程等关系，来刻画用户对系统的需求。NDRDL 还支持功能分解模型中自顶向下，逐层分解的结构化方法，提供了分层的数据流图、控制流图等设施。

NDRDL 语言具有如下特点：

1)形象直观 采用图形(即数据流图、控制流图、实体关系图)和正文相结合的方法描述软件需求，形象直观，易于理解。

2)表达力强 不仅提供了较强的功能需求的描述设施，而且还提供了描述非功能需求的设施，从而用户可以用 NDRDL 书写完整的软件需求定义。

3)实用性好 在功能需求描述设施的设计上，

失效时系统继续运行的问题。并在不同程度上解决故障情况下的负载均衡问题。相比之下，链式分布在容错和负载均衡方面最优。而数据镜象虽然容错性好，但负载均衡性差；交叉分布虽然负载均衡好，但容错性差。这里指的容错性是指任何两个结点同时失效时系统的可用性。在这种情况下，交叉分布不如数据镜象和链式分布。但是，任何收益都是需要付出代价的。链式分布的负载匀衡是动态调整来实现的，所以在 I/O 操作方面增加了软件的复杂性，需要有一定的算法来支持，但这一点不难做到。而交叉分布的负载均衡是静态的。

RAID 系统的数据校验方式的最大好处是不需要数据副本，减少了磁盘需求，同时还能提供并发 I/O 操作。但它只能对一个磁盘故障容错，并且对处理机故障无能为力。正是由于这一特点，它可以用在单机系统中，也可用在其他模式的多机系统中。

参考资料

[1]Haran Boral et al., Prototyping Bubba, A Highly Parallel Database System, IEEE trans. on Knowledge and data engineering, Vol. 2, No. 1, 1990

[2]David I. DeWitt et al., The Gamma Database Machine Project, same to [1]

[3]Tandem Computers inc., Introduction to Non-Stop SQL, Part no. 82317, March 1987

[4]J. Page, High Performance Database for Client/Server Systems, Applied Imformation Technology 13. Parallel Processing and Data Management, 1992

[5]S. Costicoglou et al., On Benchmarking the ORACLE Parallel Server on nCUBE2. Advances in Parallel & Distributed System. October 6, 1993

[6]杨利、周兴铭、郑若忠，并行数据库系统的体系结构，计算机科学，1994，Vol. 21, No. 4

[7]Patrick Valdnnieg. Parallel Database Systems: Open Problems and New Issues. Distributed and Parallel Databases, 1, 1993