并行数据库 体系结构 并行处理

计算机科学 1994Vol. 21 №. 4

42-47

并行数据库系统的体系结构

TP311-13

杨 利 周兴铭 郑若忠

（国防科技大学计算机研究所 长沙 410073）

摘 要 Parallel database systems will be the only Choice for the coming high-performance database systems. There are many new issues raised by the introduction of parallism within a parallel database system. This paper touches on the main issues such as parallel architecture, parallel algorithms, load balance and parallel query optimization. They are the most important things for a parallel database to be successful.

关键词 Parallel database system, Parallel processing, Architecture, Parallel algorithm.

一、引言

进入九十年代以来，越来越多的应用表明，传统的大型计算机系统缺乏支持高性能联机事务处理和复杂查询操作的能力。当今数据库规模的急剧膨胀、数据库工作负载的日益加重，以及新的应用领域的不断出现和成熟，已使传统的大型计算机达到了性能的极限。例如，美国国家专利局的信息数据库的信息量高达 25 太字节（1989 年）[1]，即使使用目前最快的大型机，按每秒处理 100 兆字节的处理速度，要把这个数据库全部检索一遍，也要花费 100 小时。设计支持海量数据和满足实时要求的高性能的数据库系统已经成为数据库研究领域所面临的一项严峻挑战。

像其它计算领域一样，并行处理也许是高性能数据库系统的必由之路。近年来，随着微处理器技术的进步，出现了大规模并行处理的研究热潮。这不仅从技术上，同时也从商业上为并行数据库系统的研究和发展创造了有利条件。按现阶段的技术水平，大规模并行处理机能提供比传统的大型机高得多的性能/价格比。美国著名的 ORACLE 公司在 ORACLE7 中实现的并行服务器就是并行数据库系统的典型代表。ORACLE7 并行服务器在 MPP 结构的 nCUBE2 上创下了运行 TPC－B 最高和每个 TPS 代价最低的两项世界记录[2]。其他一些著名数据库公司也宣布了推出并行数据库系统的计划，如 Sybase system 10 和 DB2 并行服务器。数据库研究界更是热火朝天，因为在多年的专用数据库机研究失败以后，MPP 结构为高性能数据库系统的研究带来了斩新的发展前景。

二、并行处理的结构模型与目标

十年以前，在数据库机热的研究中，人们关注的是各种代价昂贵的专用硬件，象磁泡存储器和固定头盘等设备。然而，这些技术的发展都没有达到人们的期望。专用数据库机因此已退出了历史舞台。而并行数据库系统的含义则是基于常规的商用处理机、常规的 RAM 存储器和常规的移动头盘技术连接的并行处理系统[3]。

2.1 并行性的度量

理想的并行系统应该展示两个关键特征：线性加速（linear speedup）和线性放大（linear scaleup）。前者的含义是，使用 n 倍多的硬件以 1/n 的时间完成同一项任务；后者的含义是，用 n 倍多的硬件以同样的时间完成 n 倍多的任务。形式上：

加速＝T1/Tn

放大＝nT1/Tn′

其中，T1 为在单处理机上执行一项任务所需要的时间；Tn 为同一任务在 n 台处理机上执行所需要的时间；Tn′为在 n 台处理机上执行 n 倍多的任务所需要的时间。当加速为 n 时称为线性加速；当放大为 1 时称为线性放大。

并行数据库系统中存在两种形式的放大，一是

杨利 副教授，博士研究生，主要研究兴趣包括并行处理技术、并行与分布式数据库系统、操作系统与性能分析。周兴铭 学部委员，教授，博士导师，主要研究兴趣包括并行处理技术、大规模并行处理机系统。郑若忠 教授，主要研究兴趣包括数据库及知识库系统、数据库应用系统。

批(batch)放大，二是事务性(transactional)放大。如果在一个 n 倍大的系统中，有 n 倍多的客户在 n 倍大的共享数据库上发出了 n 倍多的请求，这种形式的放大就称为事务性放大。这种应用典型地出现在事务处理和分时系统中；如果在一个 n 倍大的系统中，同一个查询执行在 n 倍大的数据库上，这种形式的放大叫做批放大。批放大多见于复杂查询的决策支持系统。

影响线性加速和线性放大的主要因素有三条：a. 启动时间，启动并行操作所需要的时间。若要启动上千个并行进程，则这一时间很容易左右实际的计算时间；b. 错开(skew)：各并行处理部件所负担的原始数据量或工作量严重不均衡。这时，服务时间取决于并行系统中负载最重的处理部件所花的时间；c. 干扰：当访问共享资源时，每个进程都会对其他进程构成影响，减慢其他进程的执行速度。进程越多，这种干扰越明显。

2.2 并行硬件的结构

理想的并行硬件应该有一台无限快的处理机，并带有一个带宽无限宽和容量无限大的存储器，并且应该无限便宜。如果有这样的机器，就无需加速和放大，也就不需要并行性了。但现在的技术还无法做到这一点。现在的问题是，如何用无限多的速度有限的处理机构造一台速度无限的处理机，用无限多个容量和带宽均有限的存储器构造一个容量和带宽均无限的存储器。这个问题在数学意义上很简单，但在技术上却存在很大难度。

实际上，在系统中每增加一台处理机，都会对系统中的其它处理机构成一定的干扰，也就是使其他处理机的解题速度略微降低。假设这种干扰为 1%，那么当 n＝105 时，该系统的加速最高只能达到 37(加速＝$n/(1+1\%)^{n-1}$)。在这种情况下，一个有一千台处理机系统的效率却只有 4%得到了发挥。

关于如何构成并行系统，存在下面三种结构模式(图 1)：

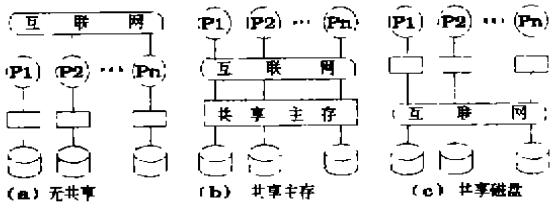

图 1 并行结构的三种模式

(1)共享主存(SM)：系统中的所有处理机都直接连接到一个全局存储器的共享磁盘系统；

(2)共享磁盘(SD)：系统中的所有处理机都有一个私有存储器，但都直接连接到一个共享磁盘系统；

(3)无共享(SN)：系统中的所有处理机都有自己的私有存储器和磁盘系统，没有共享硬件资源。

SM 和 SD 结构的缺点是系统可扩充性不好，干扰较大。SM 中互联网的带宽必需是磁盘带宽和处理机带宽之和，因而也增加了互联网的压力。为减少互联网的压力，每台处理机都配有私有缓冲区。有研究表明，在数据库应用中，这些私有缓冲区的装载和倒空会严重影响处理机的性能[4]。

SD 结构对于数据库的并发读/写操作不十分有效：某台处理机想要更新一个数据，必须先获得该数据的复本，同时必须向其他处理机宣布它的更新意图。只有当它的宣布得到所有处理机认同后，它才能开始读取并更新。这一过程都要通过网络信息交换来完成。

相比这下，SN 结构中的共享资源最少，因而干扰也最小。SN 结构还具有对互联网带宽要求较低和易于扩充规模等优点。此外，SN 结构特别适合顾客-服务员计算模式。

考虑到上述因素，许多数据库专家都认为，在并行数据库的设计中，最好一开始就采用 SN 的结构。实际上，已有很多实验室的和商品化的并行数据库系统采用了 SN 结构，如：Teradata，Tandem Nonstop SQL，NCR，Oracle-nCUBE，Gamma，Bubba，Arbre 等。

三、SQL 软件中的两种并行性

SQL 是标准化的最为流行的关系查询语言。由于 SQL 的非过程化特点和关系运算语义的简洁性，使 SQL 软件中存在着许多被并行化的机会。

首先，把一种关系运算的输出结果导向另一关系运算，使两个以上的关系运算构成流水线，就得到了“流水线并行性”(或称“程序并行性”)。其次，用多台处理机存储器分割大的数据集，使得一个运算被分割成多个相同的关系运算而作用于不同的数据对象，就得到了所谓“划分并行性”(或称“数据并行性”)。

流水线并行性的性能往往受到下列限制：(1)关系流水线通常比较短；(2)某些关系运算在消耗完全部输入关系之前不产生输出，比如聚集运算(aggregate)和排序运算(sort)；(3)某些运算的代价大于其他运算。

相比之下，划分并行性提供了较好的加速和放

大的机会。通过划分关系运算的输入和输出，可以采取分而治之的方法，把大作业转化为多个独立的小作业。这是加速和放大的理想条件。

3.1 数据划分

划分并行性的关键是数据划分。数据划分允许并行数据库系统利用多盘的并行 I/O 带宽。这种方法优于 RAID 方式的盘阵系统，又无需专用硬件。通常存在以下三种数据划分的方法：

(1)循环(round-robin)划分。这是最简单的数据划分方法：输入数据依次循环地映射到顺序盘上。这种划分对于顺序扫描全关系的查询十分有利，不过对于"相关查询"应用就存在问题。所谓"相关查询"是指类似于找出所有具有某一特定属性值的元组的应用。对于这种应用，必须启动所有的盘系统，因此可能造成不必要的启动开销。

(2)散列划分。在散列划分中，每个元组经过散列函数对其某个属性作用后被映射到某个盘上。这种划分利于顺序访问和相关查询。散列划分存在的问题是，它试图分散而不是聚集数据，但在数据库系统的设计中特别关心把相关数据聚集在相近的物理存储位置。比如，在地理数据库中，描述邻近街道的元组应物理上组织在相同磁盘块或相近磁盘块中。

(3)区域划分。把属性值相近的元组聚集在同一划分区间(盘)中。这种做法对于顺序访问、相关查询和聚集数据都是有利的。它存在的问题是下面所要谈到的数据错开(skew)。也就是说，映射到某一分区中的元组个数比其他分区多得多，甚至全部元组都被映射到了一个分区中。采用均匀分布的划分策略能够减轻区域划分所造型的数据错开程度。比如，Bubba 系统中采用的平衡元组访问频度，而不是平衡元组数的策略来划分数据[6]。

3.2 并行算法

数据划分是划分并行的第一步，接下来便是对划分了的数据执行各类并行关系运算。关系运算特别适合于并行处理，因此，近十年来，并行关系运算的研究一直引起人们的极大兴趣[7]。特别是并行连接和并行排序，由于它们的运算代价高且使用频繁而受到特别地重视。但目前提出的并行关系运算基本上都是通过对相应运算并行化，利用增加抽象的"分割-汇合"(split-merge)运算而得到的。以连接运算为例，下面是三种常用的并行算法：

(1)并行嵌套循环法(Nested-Loop)；

(2)并行分类合并法(Sort-Merge)；

(3)并行散列连接法(Hash-join)。

实验表明，新的并行算法能够改善关系运算的性能。所以，人们期待着摆脱并行化串行算法模式的新型关系并行算法的出现。

四、并行数据库中的错开问题

在并行数据库系统中，一般文献谈到的错开形式有两种：数据错开和执行错开。前者是指，由于某种原因(比如数据划分和中间结果特性)，造成了某些处理机要处理的数据量远远超过均值；执行错开是指，由于某种原因，某些处理机在其数据上的处理时间远远超过均值。一般来说，数据错开会导致执行错开。错开问题在 SN 结构的系统中显得特别严重，因为 SN 系统中没有共享系统资源。研究表明，如果不能处理好错开问题的话，将会形成系统瓶颈，使并行数据库系统的性能大大地折扣[8]。特别地，错开对并行连接算法的影响尤为严重，这其中的主要原因是，现有的并行连接算法都是对经典连接算法的简单并行化，因而对错开十分敏感。

由于认识到错开是影响并行数据库性能的主要因素之一，因此，错开问题也成为这几年并行数据库研究中十分活跃的领域。文[9]中更精确地归纳出了以下五种错开：

(1)内函错开：即关系元组中，某些属性值远远超出属性值空间的均值范围。这种形式的错开是数据源的特性，不随算法而变化；

(2)元组放置错开：指关系元组的最初划分分布。例如，使用聚集属性方法划分元组；

(3)选择错开：指同样的选择谓词作用于不同的数据分区所造成的选择率的不同。明显的例子是在分区上执行一个区域选择；

(4)重分布错开：指连接键(join keys)值的分布与重分布函数期待的分布不匹配；

(5)连接结果错开：各分区的连接的选择率呈现不同。

好的算法应能处理每一种错开。

五、处理机调度与系统负载平衡

5.1 I/O、通讯及 CPU 并行度

并行数据库系统中的查询处理要用到四种主要资源：CPU、I/O、主存和互联网。每类资源的并行度要求依赖于这类资源的负载。对于一个给定的应用问题，总存在一个响应时间阈值，在阈值以下，再缩短响应时间已显得不重要了。并行的目的是最大限度地利用有限资源，使响应时间接近阈值。现在的问题

是，怎样利用最小的并行度满足接近响应时间阈值的目标。因为，并行度越小，负载平衡问题越好解决，解题效率越高。相反，并行度越高，单个任务承担的工作量越少，可能使关系模型带来的某些好处被折衷。例如，如果每一任务所处理的数据不足一页，就丧失了顺序预读的优点。另外，当多个并行任务要访问同一页的不同元组时，也增加了 I/O 开销，这种开销会随着并行度的增加而增大。

各种资源的带宽差别很大，但它们的并行度要求之间存在一定的关系。例如，当数据全部放在主存时，任务变为 CPU 受限，因此希望 CPU 的并行度越高越好；但当数据是来自速度较慢的磁盘设备时，任务变成 I/O 受限，对磁盘的并行度要求提高。如果使用了速度更快的磁盘系统，比如盘阵，则对它的并行度要求就会降低。

对于一个应用，若用 L 表示要执行的 CPU 指令条数（通常包括执行 I/O、上锁、求谓词、排序、连接等），I 表示从磁盘读取的数据页面总数，那么，L/I 就刻画了由磁盘每读取一个数据页面而引发的平均 CPU 指令条数。L/I 很好地刻画了一种工作负载，因此可用来研究 CPU 与 I/O 之间的并行度要求和负载平衡（CPU 速度和 I/O 速度是系统参数）。

如果 L/I 表明 CPU 和 I/O 的并行度接近，则无需做什么工作；如果 L/I 表明，I/O 并行度比 CPU 并行度高很多，那么就必须想办法增加 I/O 的并行度。提高 I/O 系统速度（采用新设备）、增加主存容量能减少对 I/O 并行度的要求。

还需指出的是，数据库软件和应用类型直接影响到对并行度的要求。当发现新的更快的数据库算法时，L/I 将减小，因此构成对 I/O 系统更高的并行度要求；反之，当应用问题涉及复杂的关系查询时，L/I 增大，对 I/O 并行度的要求就有所降低。

5.2 处理机调度与任务分配

一般认为，流水线并行性的两个主要问题是处理机调度和运算顺序的选择。处理机调度的主要目的是平衡系统的物理资源负载，最大限度地利用这些资源，缩短单个查询的执行时间和提高全系统的吞吐率。运算顺序的选择主要考虑利用各种软件资源。例如，数据缓冲区就是一种软件资源。即，希望一个（生产者）任务产生的中间结果立即为下一个（消费者）任务利用，否则它必须把中间结果先写入磁盘，而后来的消费者任务必须从磁盘中读取，形成了所谓数据缓冲区抖动现象，增加了不必要的 CPU 开销和 I/O 开销。但简单考虑数据缓冲区立即使用也存在问题。过分考虑减少数据缓冲区的抖动，可能会造成很多任务急于消费缓冲区中的数据，结果每个任务都得不到充分执行，反而又可能引起新的缓冲区抖动。

值得注意的是，由于数据库应用的特点和性能要求，越来越多的现代数据库系统倾向于从操作系统方面接管一部分资源调度权力，包括磁盘空间管理、缓冲区管理、特别是 CPU 的调度权。CPU 调度在并行数据库中是重要的，原因是，并行查询优化器在编译时往往缺乏系统信息，因而必须在动态执行时加以弥补。

六、并行查询优化

数据库查询优化技术一直受到十分的重视。在传统的优化方法中通常采用了代数变换和启发式两种优化措施，其时间复杂性是一个主要问题。并行性的引入又给查询优化带来了新的研究课题。首先，编译时的静态优化缺乏关于系统资源可用性的知识（如可用的缓冲区大小、自由处理机的数目等），因此它产生的执行规划很难充分利用并行流水线资源以达到线性缩短求解时间之目的。解决这个问题的办法是采用动态优化技术。动态优化除了执行编译时的优化外，还建立了运行时支持功能，根据 CPU 负载、中间结果特性、主存使用情况等系统信息动态适应执行规划。很显然，动态优化比静态优化困难得多。其次，并行执行规划的搜索空间变得更加庞大，使动态优化的时间复杂性问题更加严重。

是否存在一种折衷方案，不致于增加太多的复杂性？一种方案是所谓两段静态优化，它的含义是，首先在忽略并行性的前提下对查询进行常规优化，然后，将选中的查询规划做进一步的并行化优化。这种方法也能减小优化的搜索空间，因此在并行化一个现存的 DBMS 时特别有吸引力。

然而，优化是分段进行的，可能产生不了最优执行规划，因为两段优化过程是孤立的。例如，假设第一段优化选择了规划 P1 而拒绝了规划 P2。若 P2 比 P1 的代价昂贵得多，通常我们对并行 P2 不会感兴趣，但当 P1 的资源消耗并不明显好于 P2，而 P2 的并行化又优于 P1 时，我们实际上丢失了 P2。当 P1 使用非划分索引扫描而 P2 使用划分索引扫描时就是如此。或许给顺序优化器某些提示有助于帮助它选择那些易于并行化的规划。

目前的优化所基于的代价模型全都依靠估计，在并行优化特别是在复杂的并行优化时就显得不

够。并行优化需要考虑的另一个问题是高度的数据错开分布。错开可能导致中间结果规模在很大范围内变化，在这种情况下，传统的代价模型已失去意义。

在并行查询中，查询树的选择也与传统的做法不同。右深树(right-deep tree)利于流水线并行，但其结构灵活性差，因而对性能构成限制。相比之下，丛生树(bushy tree)为查询规划的生成提供了更多的灵活性。已经证明，对于分类合并连接算法，丛生树的执行性能可以超过线性树，特别是当查询中涉及的关系数目很大时更是如此。但对于散列连接算法，丛生树结构的执行规划调度要比右深树复杂得多。要达到丛生树充分利用流水线的目的，即使不是不可能，也是非常困难的。因此，需要提出一种更趋合理的查询树结构，使之更利于并行优化的执行规划的产生。比如，有人提出了分段右深树结构。

七、并行恢复与并发控制

并行恢复对于并行数据库系统的坚固性致关重要。当系统中某一结点或磁盘发生故障，其余存活的结点要能够自动探测，并利用日志信息，把故障结点的工作接管过来，从而使系统继续正常运转。并行系统中的每个结点只维持一部分日志信息，必须采取某种措施保证其一致性。

另一方面，并发控制对系统的响应速度影响很大。并行数据库系统支持的并发事务数目要比传统的大型机系统大得多，要获得高的吞吐率，必需采用先进的控制技术。Oracle7 采用了"行级锁"(row-level locking)思想，利用先进的"并行缓冲区管理"和"并行锁管理"技术实现并发控制。这种独特的并发控制机制不会发生并发更新和查询的阻塞，提供了最细粒度的锁控制，保证了数据资源上的竞争最小。并行缓冲区管理程序和并行锁管理程序负责追踪数据块副本的位置和最新状态，使结点间的数据移动最小化，能有效地节约网络带宽。Oracle7 的并行缓冲区管理实际上是一种改进了的多版本控制技术，目前看来是一种比较先进的并发控制技术。

八、总结

并行数据库系统主要有两个目的：一是利用多 CPU 和盘等系统资源极大地提高多用户处理能力；二是利用多 CPU 和盘实现单个查询的并行处理，显著缩短复杂查询的执行时间。简单地说就是查询间并行和查询内并行(inter-queries parallism vs. intra-query parallism)。查询内并行性又分为算子间并行性和算子内并行性(inter-operator vs. intra-operator parallism)。围绕这两个目的，本文从系统结构的角度提出了六个方面的设计问题。

文中讨论比较了三种并行结构模式，指出 SN 结构更适合作为并行数据库系统的平台。但 SN 结构存在的最大问题是不易实现负载均衡。或许综合了 SD 和 SN 两者优点的混合结构更为合理。如日本的超级数据库机 SDC 在一个结点中采用了 SM 结构，构成一种 SM 与 SN 的混合结构。

我们也讨论了数据划分问题。目前较多的研究关注于单属性划分，实际查询通常都是在多个属性上进行，而多维划分研究得还不充分。

除了本文提出的六点问题之外，并行数据库系统还有其它问题需要认真研究。如，关于数据库的并行装填、并行卸出和并行数据重组等实用程序的研究；关于联机短事务与复杂长事务混合应用环境的研究；关于数据库应用程序并行性工具以及巨型数据库的具有增量、可恢复功能的联机备份技术的研究等；其它包括如何把并行数据库应用于分布数据库系统，如何在并行数据库系统中引入诸如知识库技术等。

并行数据库系统的时代已经到来，随着我国数据库应用规模和范围的扩大，可以预料，很快会出现对高性能数据库的需求。并行数据库必然在未来的高性能数据库系统中发挥主要作用。

参考文献

[1] A. R. Hurson et al., Parallel architectures for database systems, Computer Society Press, 1989.

[2] ORACLE nCUBE for Oracle7—— Technology overview, March, 1993.

[3] David DeWitt and Jim Gray, Parallel database systems: The future of high performance database systems. Communication of ACM, Vol. 35, No. 6, June, 1992.

[4] Thakkon, S. S. and Sweiger, M., Performance of an OLTP applications on symmetry multiprocessor system. In Proceedings of the 17th annual International Symposium on Computer Architecture, May, 1990.

[5] Hamid Pirahesh, et al., Parallelism in relational database systems: Architecture issues and design approaches. In Proceedings of the 2nd internation-

计算机科学 1994Vol. 21 No. 4

SQL的现状及其研究与发展

胡志平 麦中凡
（北京航空航天大学计算机系 北京 100083）

摘 要 The research and development of SQL technology is still a focus in the field of database. This paper summarizes the work on SQL standard and its present situation, and discusses the continuous functional modifications of database's improviment in the mean time. Based on most people's ideas, we emphisize the later research on SQL3 about procedure language, type system and query technology.

关键词 SQL, Database, Standard, Relation, Object, Type, Model

一、SQL简介

如果说七十年代关系数据库还处于研制实验阶段，那么到了八十年代大量关系数据库商用产品的面市几乎占据了整个数据库市场，其中的原因在于它们使用了统一的语言界面，那就是，SQL成为了关系数据库的通用语言。

1974年，IBM公司San Jose研究实验室的D. D. Chamberlin在开发SEQUEL-XRM原型时提出了SEQUEL(Structured English Query Language)，1976年又修改成为SEQUEL/2，也就是目前的SQL(Structured Query Language)。SEQUEL/2主要成形于IBM的System R原型中[2]，并在原有基础上大加扩展，其后逐渐完善并最终成为一种完备的结构化数据库查询语言。

到目前为止，SQL语言已广泛用于关系数据库中，起初的SQL也是围绕着关系模型而制定的，SQL语言在关系数据库中主要有四大功能：查询，操纵(Manipulation)，定义，控制[1]，这四大功能基本上构成了关系数据库的功能主体，从七十年代到九十年代，短短的二十年间，关系数据库能广泛地流行起来，与SQL语言的作用是分不开的。

随着现代计算技术的进一步发展，特别是近年来OO技术的兴起，关系模型有了诸多的限制，SQL语言将如何适应新的对象模型成为SQL发展的目标，这也是本文中将要讨论的SQL3所要完成的工作。

二、SQL标准的产生与发展

众所周知，美国国家标准委员会ANSI和国际标准化组织ISO是世界上最具有权威的两大标准化组织，1986年10月，ANSI正式把SQL作为关系数据库的标准语言，并颁布了标准的SQL文本，X3. 135-1986，此后不久，ISO也作出了同样的决定，公布了IS9075，1986标准。这就是我们通常所说的SQL-86。SQL-86为软件制造商提供了一种极大的可能性，那就是无论在那种机器平台上，还是何种数据库系统，都可以采用SQL作为共同的数据存取或标准接口，从而使未来的数据库世界有可能连接成为一个统一的整体。这个前景是十分诱人而意义重大的，因而有

al symposium on database in parallel and distributed systems, July, 1990.

[6] Copeland, G. et al., Data Placement in Bubba. In Proceedings of ACM-SIGMOD international conference on management of data, May, 1988.

[7] Donovan A. Schneidr and David J. DeWitt, A Performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment, In Proceedings of ACM-SIGMOD international conference on management of data, 1989.

[8] M. Seetha Lakshmi and Philip S. Yu, Limiting factors of join performance on parallel processors, In Proceedings of the 5th international conference on data engineering, 1989.

[9] Christopher B. Walton et al., A taxonomy and performance model of data skew effects in parallel joins, In Proceedings of the 17th international conference on VLDB, Sept., 1991.