第 12 讲:查询执行 2

15-445/645 数据库系统(2022 年秋季)

https://15445.courses.cs.cmu.edu/fall2022/

卡内基梅隆大学安迪·帕夫洛

1 背景

之前关于查询执行的讨论假设查询是用单个 worker(即线程)执行的。然而，在实践中，查询往往是与多个 worker 并行执行的。

并行执行为 dbms 提供了许多关键的好处:

- 吞吐量(每秒更多的查询)和延迟(每次查询时间更少)方面的性能提升。
- 从 DBMS 的外部客户的角度增加了响应性和可用性。
- 潜在地降低*总拥有成本*(TCO)。该成本包括硬件采购和软件许可，以及部署 DBMS 的人力开销和运行机器所需的能源。

dbms 支持两种类型的并行性:查询间并行性和查询内并行性。

2 并行数据库与分布式数据库

在并行和分布式系统中，数据库都分散在多个“资源”上，以提高并行性。这些资源可以是计算性的(例如，CPU 核心、CPU 插槽、gpu、附加机器)或存储性的(例如，磁盘、内存)。

区分并行系统和分布式系统很重要。

- **并行 DBMS** 在*并行 DBMS* 中，资源或节点在物理上彼此接近。这些节点通过高速互连进行通信。假设资源之间的通信不仅快速，而且廉价可靠。
- **分布式 DBMS** 在*分布式 DBMS* 中，资源可能彼此相距很远;这可能意味着数据库跨越了世界上不同地方的机架或数据中心。因此，资源在公共网络上使用较慢的互连进行通信。节点之间的通信成本更高，故障不可忽视。

即使一个数据库在物理上被划分为多个资源，但对应用程序来说，它仍然是一个逻辑数据库实例。因此，对单节点 DBMS 执行的 SQL 查询应该在并行或分布式 DBMS 上生成相同的结果。

3 进程模型

DBMS *过程模型*定义了系统如何支持来自多用户应用程序/环境的并发请求。DBMS 由多个*工作人员*组成，这些工作人员负责代表客户端执行任务并返回结果。一个应用程序可能同时发送一个大请求或多个请求，这些请求必须分配给不同的 worker。

DBMS 可以采用两种主要的进程模型:每个工作进程和每个工作线程。第三种常见的数据库使用模式采用嵌入式方法。

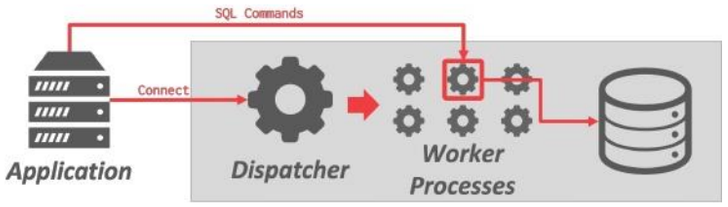

图 1:每个工作者的进程模型

每个工人的进程

最基本的方法是*每个工人的流程*。这里，每个 worker 都是一个独立的 OS 进程，因此依赖于 OS 调度器。一个应用程序发送一个请求，并打开到数据库系统的连接。一些分配器接收到请求后，选择它的一个工作进程来管理连接。应用程序现在直接与负责执行查询所需请求的工作进程进行通信。这个事件序列如图 1 所示。

依赖操作系统进行调度有效地减少了 DBMS 对执行的控制。进一步，该模型依赖共享内存来维护全局数据结构或依赖消息传递，这具有较高的开销。

每个工作进程方法的一个优点是，进程崩溃不会破坏整个系统，因为每个工作进程都在自己的 OS 进程的上下文中运行。

这种进程模型提出了在独立进程上的多个 worker 对同一页面进行大量复制的问题。一个最大化内存使用的解决方案是为全局数据结构使用共享内存，以便它们可以被运行在不同进程中的 worker 共享。

利用每个 worker 进程模型的系统示例包括 IBM DB2、Postgres 和 Oracle。在开发这些 dbms 时，pthreads 还没有成为标准的线程模型。线程的语义因 OS 而异，而 fork()的定义更好。

每个工作线程

现在最常见的模式是*每个工人的线程*。每个数据库系统只有一个具有多个工作线程的进程，而不是由不同的进程执行不同的任务。在这种环境下，DBMS 完全控制任务和线程，它可以管理自己的调度。多线程模型可以使用，也可以不使用分配器线程。每个 worker 模型的线程示意图如图 2 所示。

使用多线程架构提供了某些优势。首先，每次上下文切换的开销更少。此外，共享模型不需要维护。然而，每个工作线程模型并不一定意味着 DBMS 支持查询内并行性。

在过去的 20 年里，几乎所有的 DBMS 都使用了这种方法，包括 Microsoft SQL Server 和 MySQL。IBM DB2 和 Oracle 也更新了他们的模型以提供对这种方法的支持。Postgres 和 Postgres 衍生的数据库在很大程度上仍然使用基于过程的方法。

调度

总之，对于每个查询计划，DBMS 必须决定在何处、何时以及如何执行。相关的问题包括:

应该使用多少个任务?

•它应该使用多少 CPU 内核？

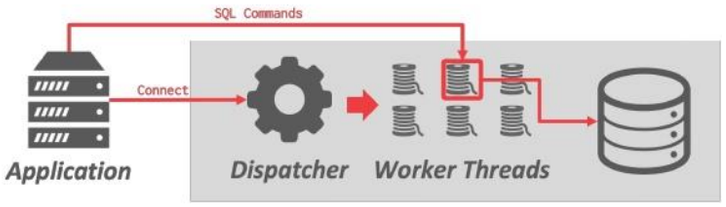

图 2:线程 per Worker 模型

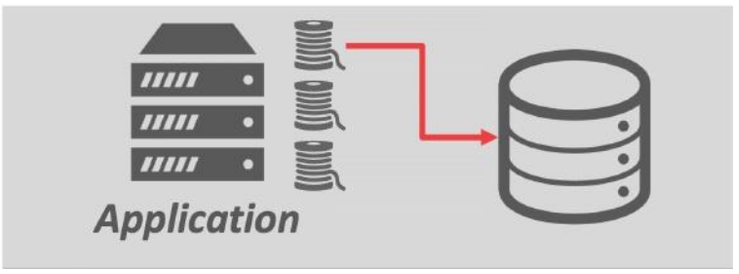

图 3:嵌入式 DBMS 调度

- 任务应该在哪些 CPU 内核上执行?

任务应该将其输出存储在哪里?

在制定有关查询计划的决策时，DBMS **总是**比操作系统知道的更多，因此应该优先考虑。

嵌入式 DBMS

数据库的一种非常不同的使用模式涉及在应用程序的相同地址空间中运行系统，而在客户机-服务器模型中，数据库独立于应用程序。在这种情况下，应用程序将设置在数据库系统上运行的线程和任务。应用程序本身将主要负责调度。嵌入式 DBMS 的调度行为图如图 3 所示。

DuckDB、SQLite 和 RocksDB 是最著名的嵌入式 dbms。

4 查询间并行性

在*查询间并行中*，DBMS 并发地执行不同的查询。因为多个 worker 同时运行请求，所以整体性能得到了提高。这就增加了吞吐量，减少了延迟。

如果查询是只读的，那么查询之间几乎不需要协调。然而，如果多个查询同时更新数据库，则会出现更复杂的冲突。这些问题将在第 15 讲中进一步讨论。

5 查询内并行

在*查询内并行*中，DBMS 并行地执行单个查询的操作。这减少了长时间运行的查询的延迟。

内部查询并行性的组织可以按照*生产者/消费者*范式来考虑。每个操作符既是数据的生产者，也是数据的消费者，这些数据来自于运行在它下面的某些操作符。

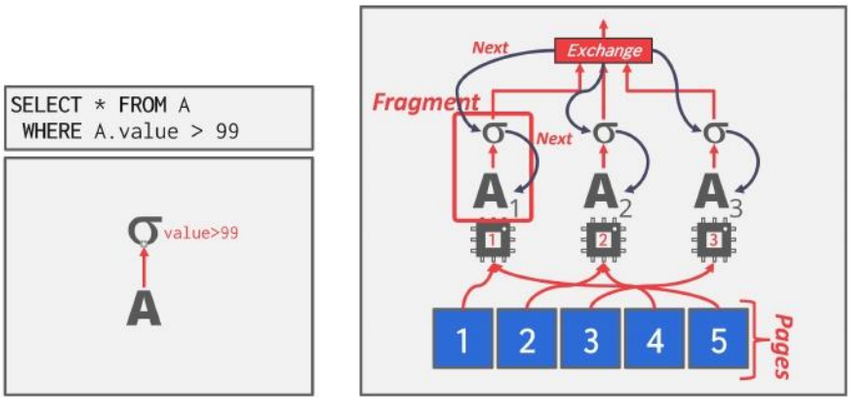

图 4:操作符内并行性—此 SELECT 的查询计划是对 a 进行顺序扫描，并将其馈送到过滤器操作符中。为了并行运行，查询计划被分割成不相交的片段。给定的计划片段由一个不同的 worker 在 A 上操作。交换操作符并发地对所有的片段调用 Next，然后这些片段从它们各自的页面检索数据。

每个关系运算符都存在并行算法。DBMS 既可以让多个线程访问集中的数据结构，也可以使用分区来划分工作。

在内部查询并行性中，有三种类型的并行性:内部操作符，内部操作符和浓密的。这些方法不是互斥的。在给定的工作负载上，以一种优化性能的方式组合这些技术是 DBMS 的责任。

操作符内并行性(水平)

在*操作符内部并行性*中，查询计划的操作符被分解为独立的*片段*，在不同的(不相交的)数据子集上执行相同的功能。

DBMS 将*交换*操作符插入到查询计划中，以合并子操作符的结果。交换操作符阻止 DBMS 执行计划中它上面的操作符，直到它从子操作符接收到所有数据。图 4 显示了一个这样的例子。

一般来说，有三种类型的交易所运营商:

- **汇总:**将多个 worker 的结果合并为单个输出流。这是并行 dbms 中最常用的类型。

  分布式:将单个输入流分割为多个输出流。

- **重新划分:**跨多个输出流重新组织多个输入流。这允许 DBMS 接受以一种方式分区的输入，然后以另一种方式重新分配它们。

操作符间并行(垂直)

在*操作符间并行性*中，DBMS 重叠操作符，以便将数据从一个阶段输送到下一个阶段，而不需要具体化。这有时被称为*流水线并行*。参见图 5 中的示例。

这种方法在*流处理系统*中被广泛使用，流处理系统是指在输入元组流上持续执行查询的系统。

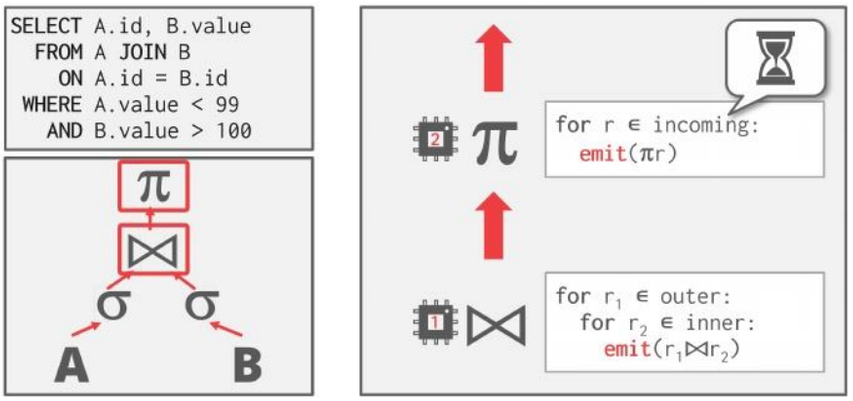

图 5:操作符间并行性—在左侧的 JOIN 语句中，单个工作人员执行连接，然后将结果发送给执行投影的另一个工作人员，然后再次发送结果。

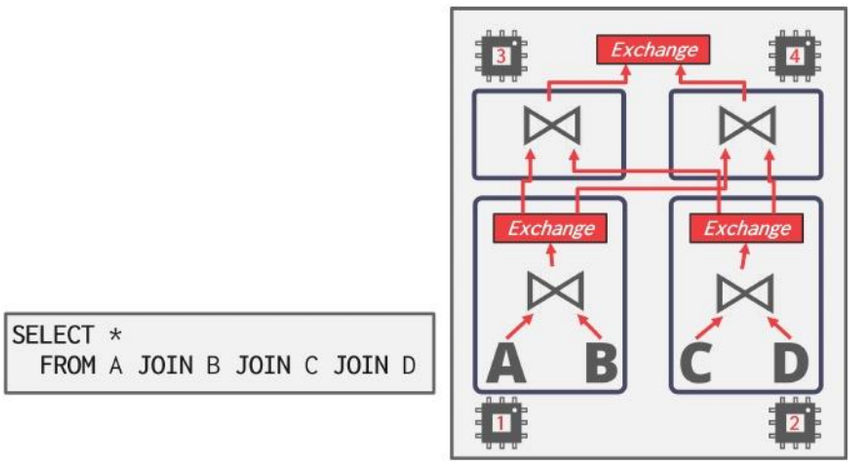

图 6:密集并行性—在三个表上执行 4-way JOIN，查询计划被划分为四个片段，如图所示。查询计划的不同部分同时运行，其方式类似于操作符间并行。

灌木并行性

Bushy parallelism 是操作符内并行和操作符间并行的混合，其中 worker 同时执行来自查询计划不同段的多个操作符。

DBMS 仍然使用交换操作符来组合这些段的中间结果。图 6 展示了一个例子。

I/O 并行性

如果磁盘始终是主要瓶颈，使用额外的进程/线程并行执行查询不会提高性能。因此，能够跨多个存储设备拆分数据库是很重要的。

为了解决这个问题，dbms 使用 I/O 并行来*拆分跨多个设备的安装*。I/O 并行的两种方法是多磁盘并行和数据库分区。

多磁盘并行性

在*多磁盘并行中*，操作系统/硬件被配置为跨多个存储设备存储 DBMS 文件。这可以通过存储设备或 RAID 配置来完成。所有的存储设置对 DBMS 都是透明的，因此工作人员不能在不同的设备上操作，因为DBMS 不知道底层的并行性。

数据库分区

在*数据库分区*中，数据库被划分为不相交的子集，这些子集可以分配给离散的磁盘。有些 dbms 允许指定每个单独数据库的磁盘位置。如果 DBMS 将每个数据库存储在单独的目录中，那么在文件系统级别就很容易做到这一点。更改的日志文件通常是共享的。

*逻辑分区*的思想是将单个逻辑表拆分为不相交的物理段，这些物理段分别存储/-管理。理想情况下，这样的分区对应用程序是透明的。也就是说，应用程序应该能够访问逻辑表，而不关心事物是如何存储的。

分区的两种方法是垂直分区和水平分区。

在*垂直分区*中，表的属性被存储在一个单独的位置(就像列存储)。为了重建原始记录，必须存储元组信息。

在*水平分区*中，表的元组根据一些分区键被划分为不相交的段。有不同的方式来决定如何分区(例如，哈希、范围或谓词分区)。每种方法的功效取决于查询。

我们将在本学期晚些时候讨论分布式数据库时介绍这些方法。