Query Execution

In-memory的DBMS，没有了磁盘IO瓶颈，那么可能存在的优化点在于?

优化目标

减少要执行指令数目

减少每条指令执行的周期

- 分支预测
- 减少Cache Miss

并发执行

- 使用多线程

Operator Operation

- Query Plan Processing
- Scan Sharing
- 物化视图
- Query Compilation
- 向量化Operators
- 并行算法
- UDF

今天的内容

CPU

CPU指令以流水线组织运行

超标量CPU支持多流水线(如果多条指令互相独立,可以并发在单周期执行)

CPU / DBMS PROBLEMS

- Dependencies

如果指令相互依赖，不能够同时在同一条pipeline中执行。

- Branch Prediction

CPU会尝试预测下一个周期执行的是哪个分支，如果预测失败了就会flushh pipeline

对OLAP的DBMS，范围搜索中where操作其实基本上是预测不了的

PROCESSING MODEL

决定DBMS如何执行query计划

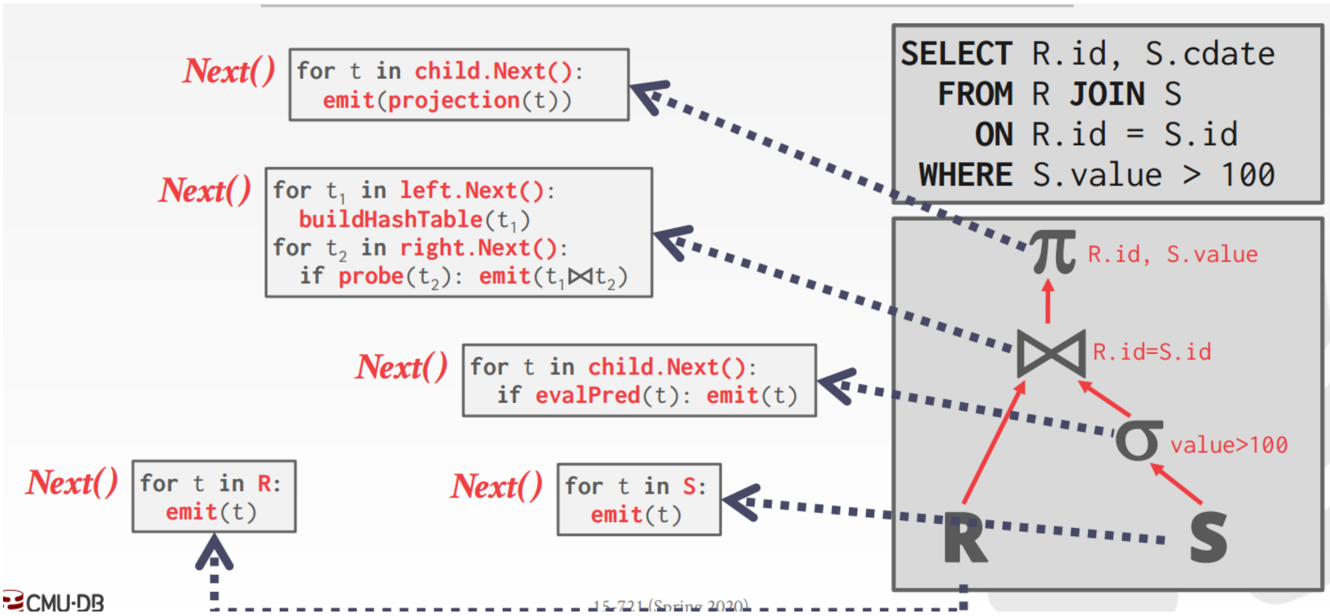
Iterator Model

每一个query plan实现一个`next`函数

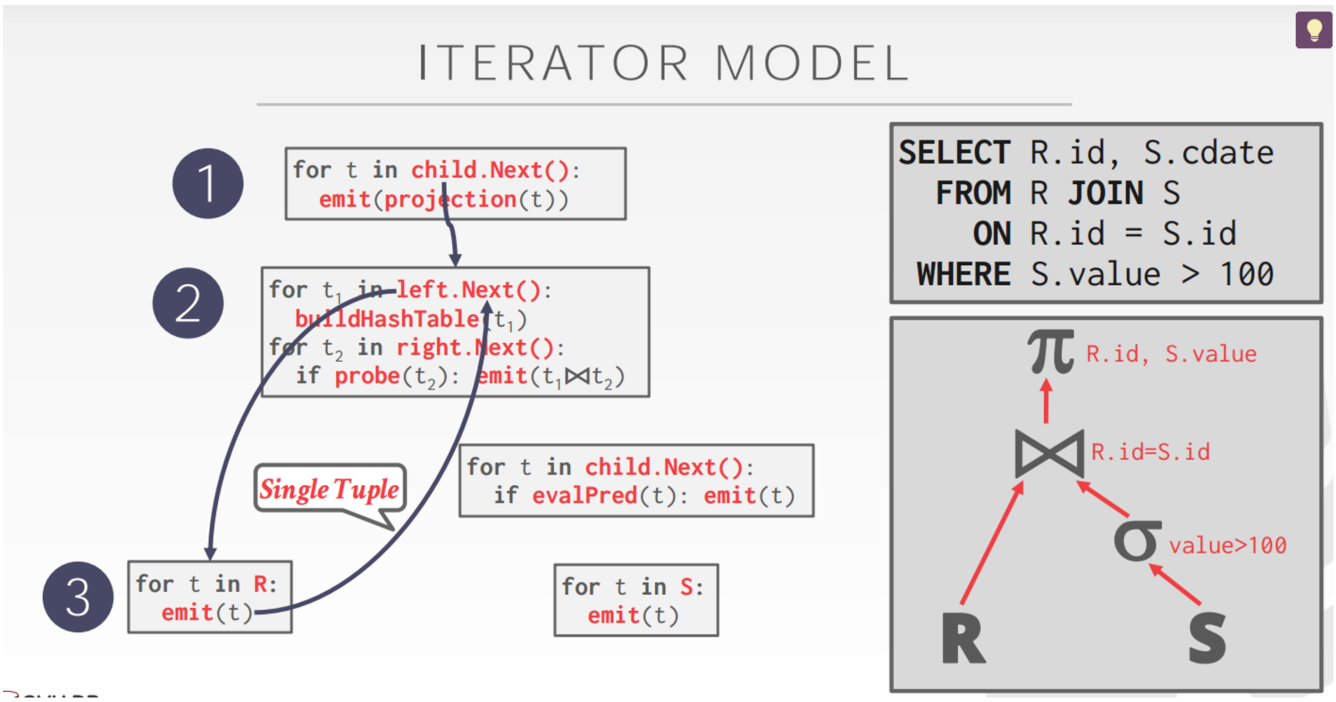
每次调用的时候，operator要么返回一个tuple结果，要么一个null标志没有其他tuple

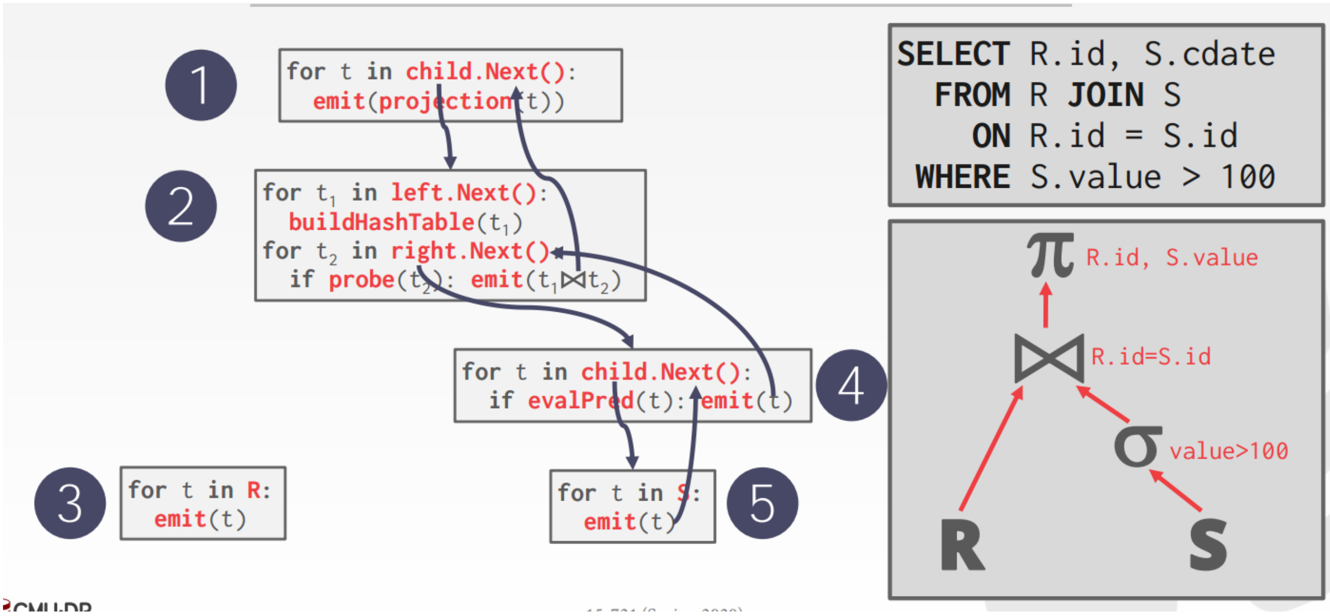
operator会实现一个loop，对其子operator调用next来获得其对应的tuple然后处理他们。

也叫做`Volcano`或者`Pipeline`模型



像下图那样，每个operator对应一个for loop,每次子operator会返回单个tuple的结果

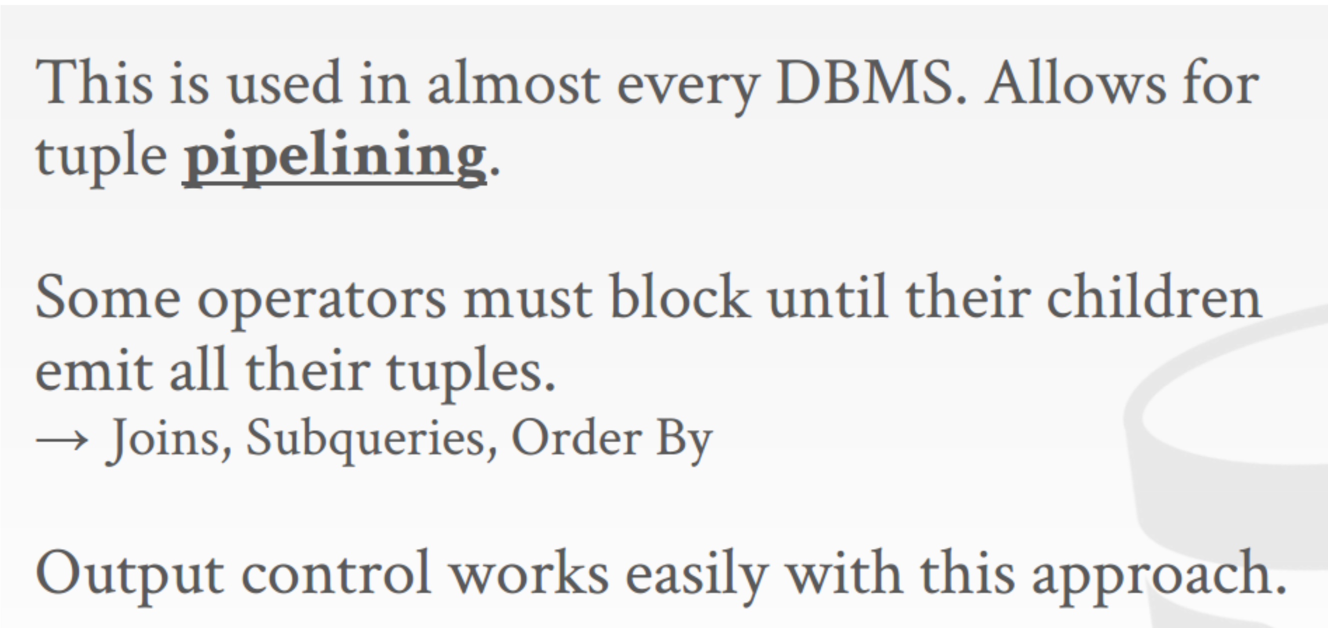




几乎所有的DBMS都会采用这种模型。允许流水线执行tuple

一些算子operator会阻塞直到其子operator输出完所有数据

eg: Join,Subqueries,Order by



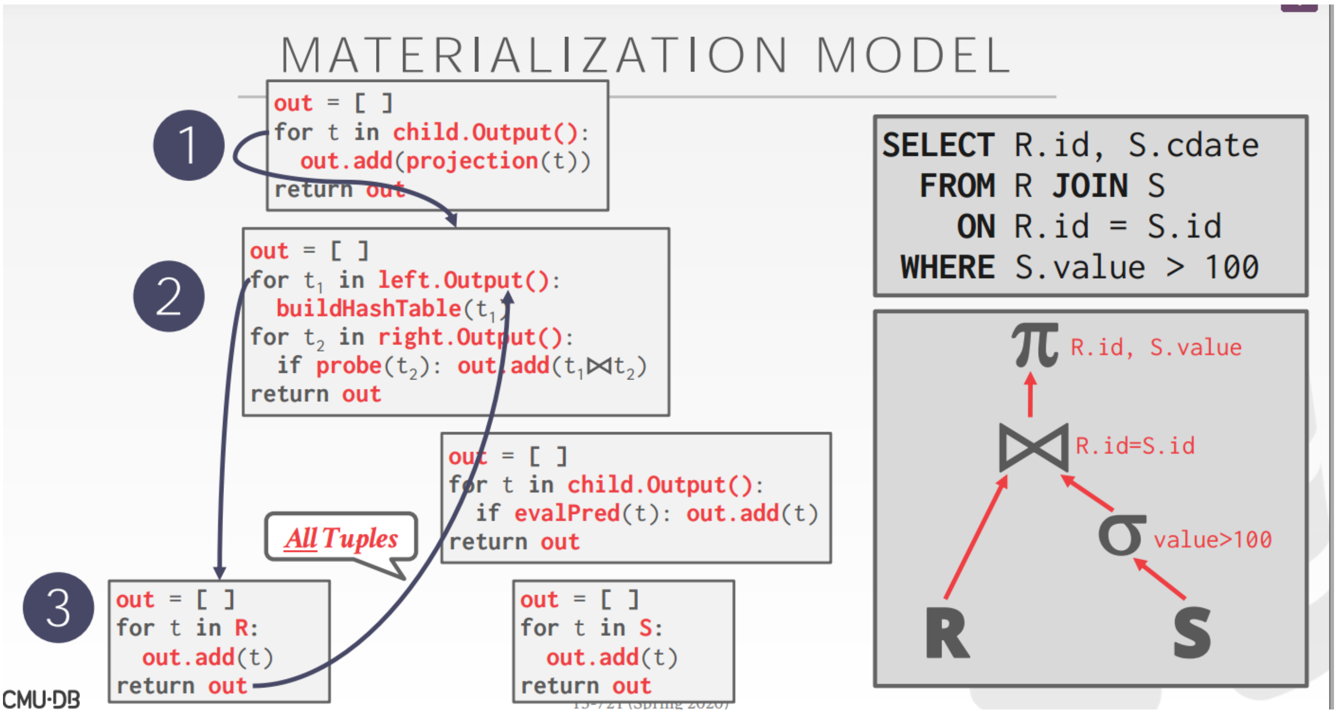
Materializaion Model

每个算子一次性处理所有输入输出

物化指的是把输出结果物化成一个单一的结果

DBMS可以把把hints下移，避免扫描过多的tuples

输出结果可以是全部tuple(行式存储),也可以是某些列(列式存储)



每个operator的`out[]`是buffer，结果会放入buffer中，operator处理完之后会把buffer中的所有结果交付给父operator

适用于OLTP，因为OLTP一次只会访问少量的tulpe结果

- 更少的execution/coordination 开销
- 更少的函数调用

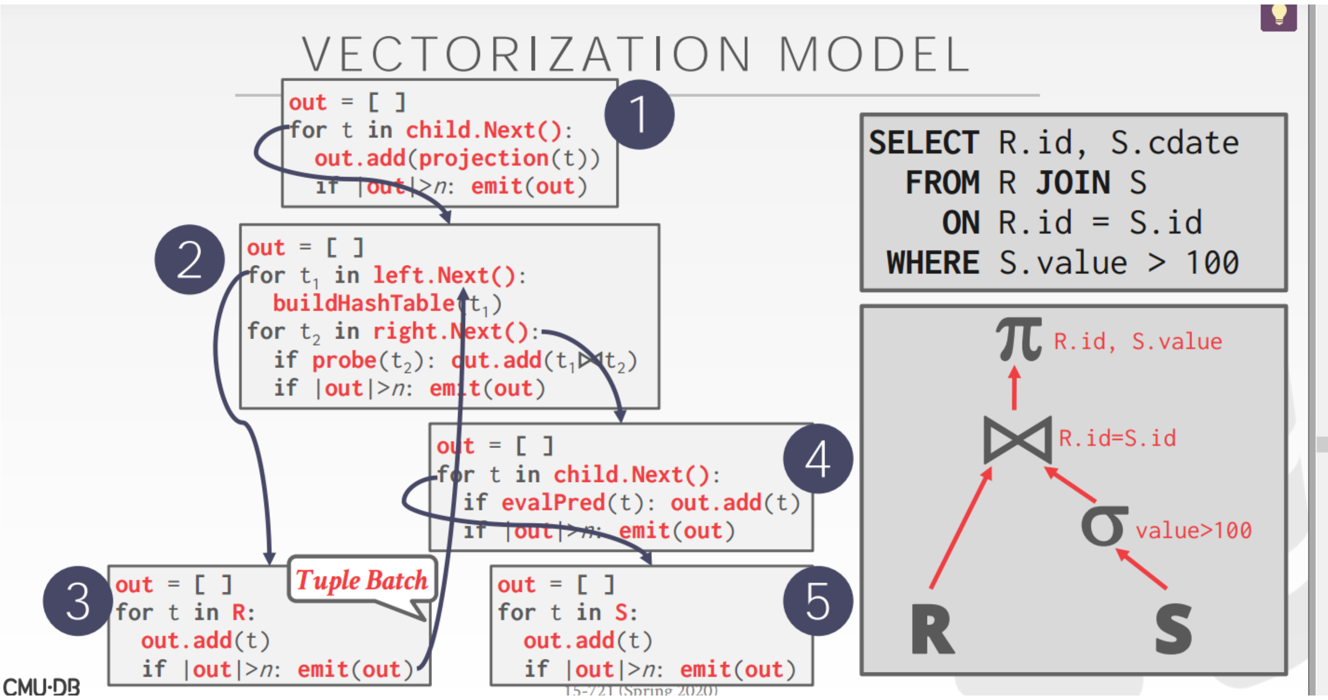
不适用于OLAP，因为OLAP会访问部分列的所有tuple，对结果实时性要求强

Vectorized/Batch Model

与Iterator Model类似，每一个operator会实现一个next函数

每一个operator不会emit单个tuple结果，而是emit批结果

- 每个operator的 内部循环会一次处理多个tuple
- 处理的批的大小取决于硬件和query的性质



适用于OLAP数据库

因为极大地减少了每个operator的调用次数

允许operator使用SIMD(单指令多数据)指令来对tuple做批次处理

生成计划处理方向

方法一： 由上至下

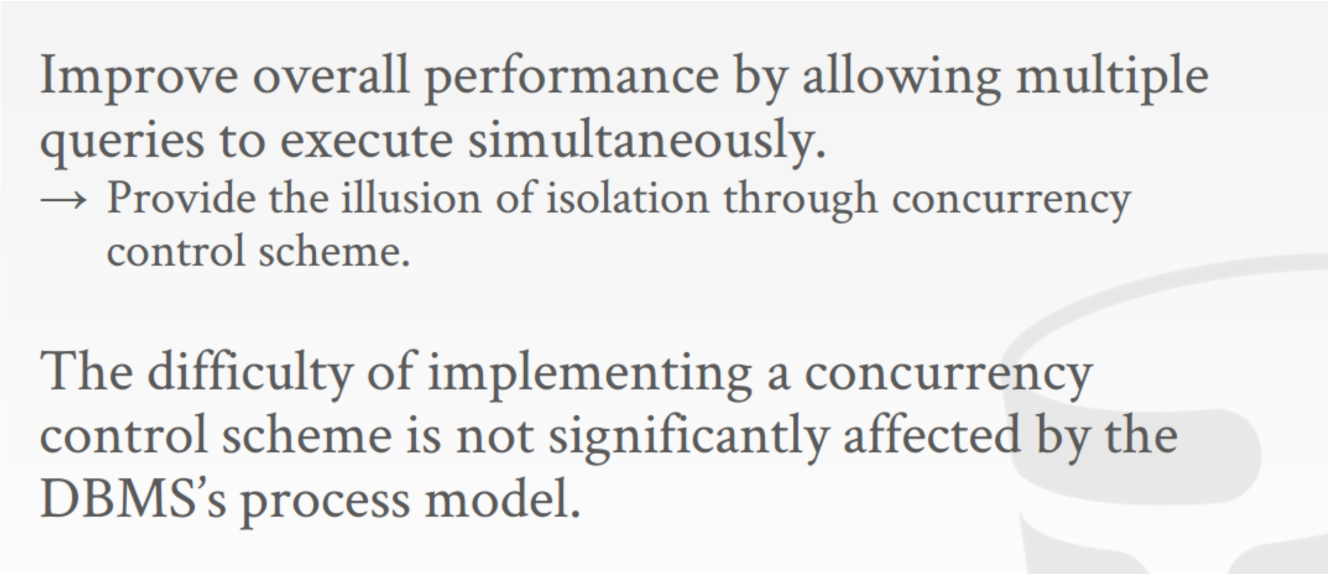
- 从根operator开始，从子operator处理完的数据pull到父节点
- tuple往往通过函数调用来传递

方法二： 由下至上

- 从叶子节点开始，并且把子operator处理的数据push到父operator
- 对缓存和寄存器会有更精细化的管理

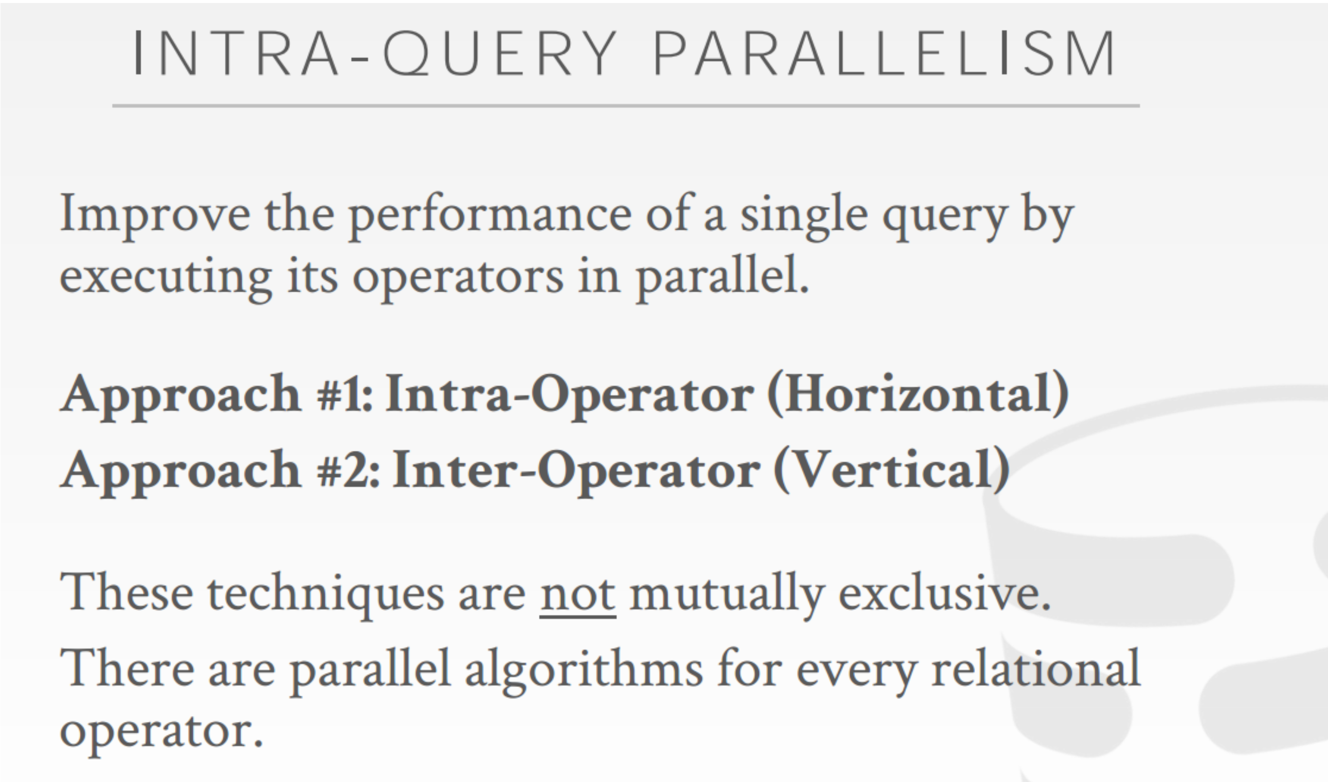
INTER-QUERY 并行

同时允许多条query执行

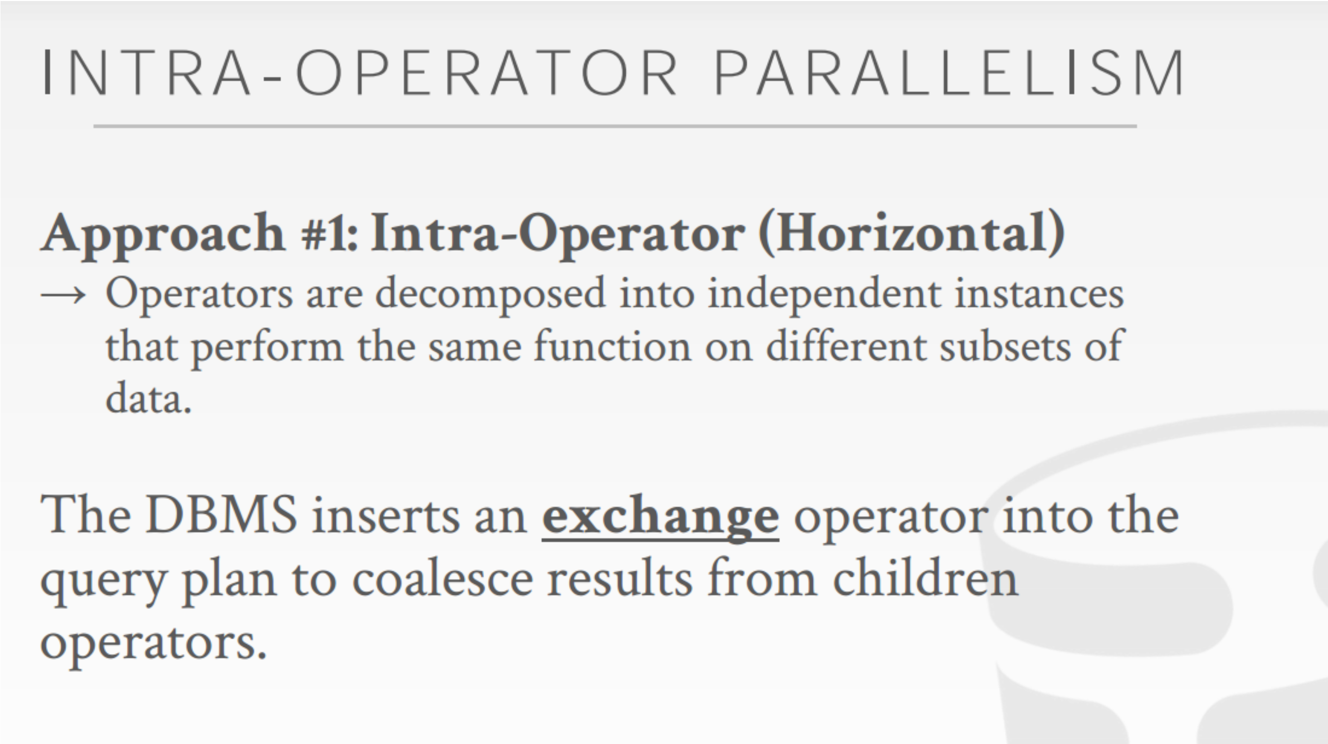


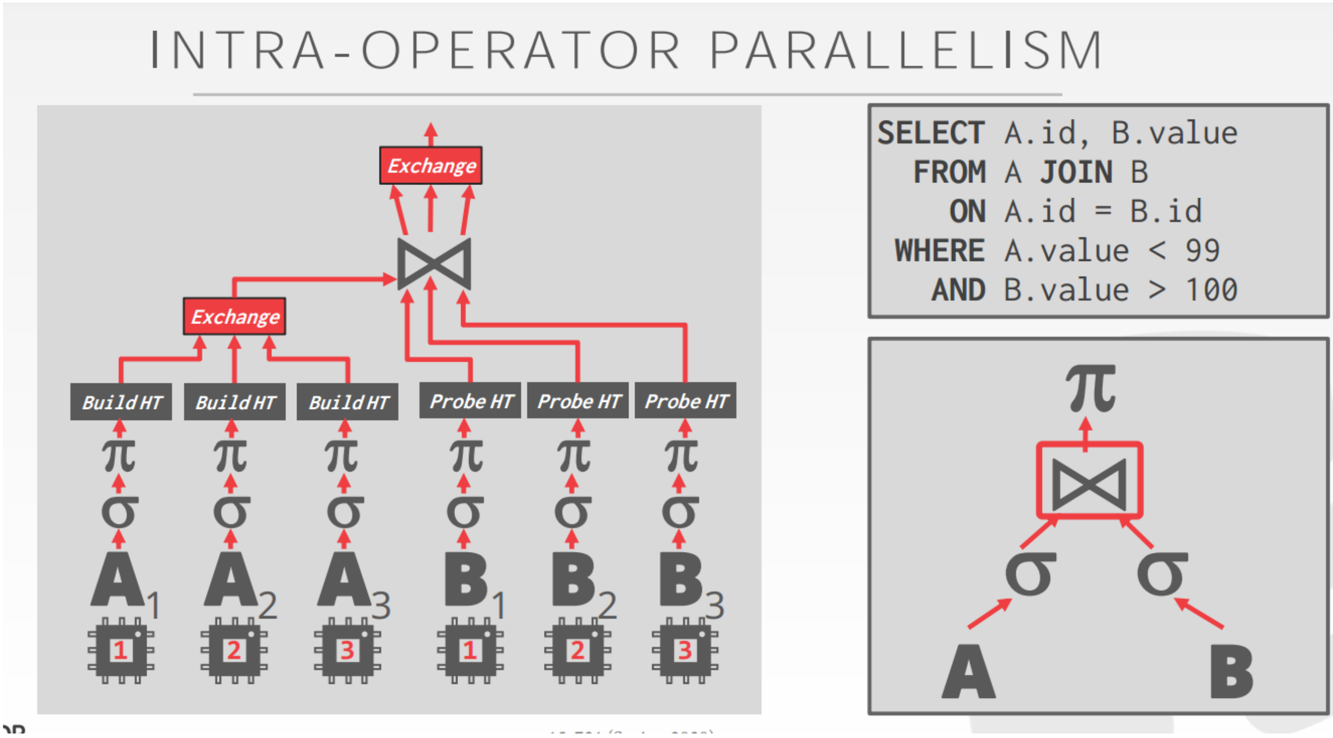
INTRA-QUERY 并行

单条Query内允许并行执行多个operator（SINGLE QUERY PARALLEL OPERATOR）



Intra-Operator

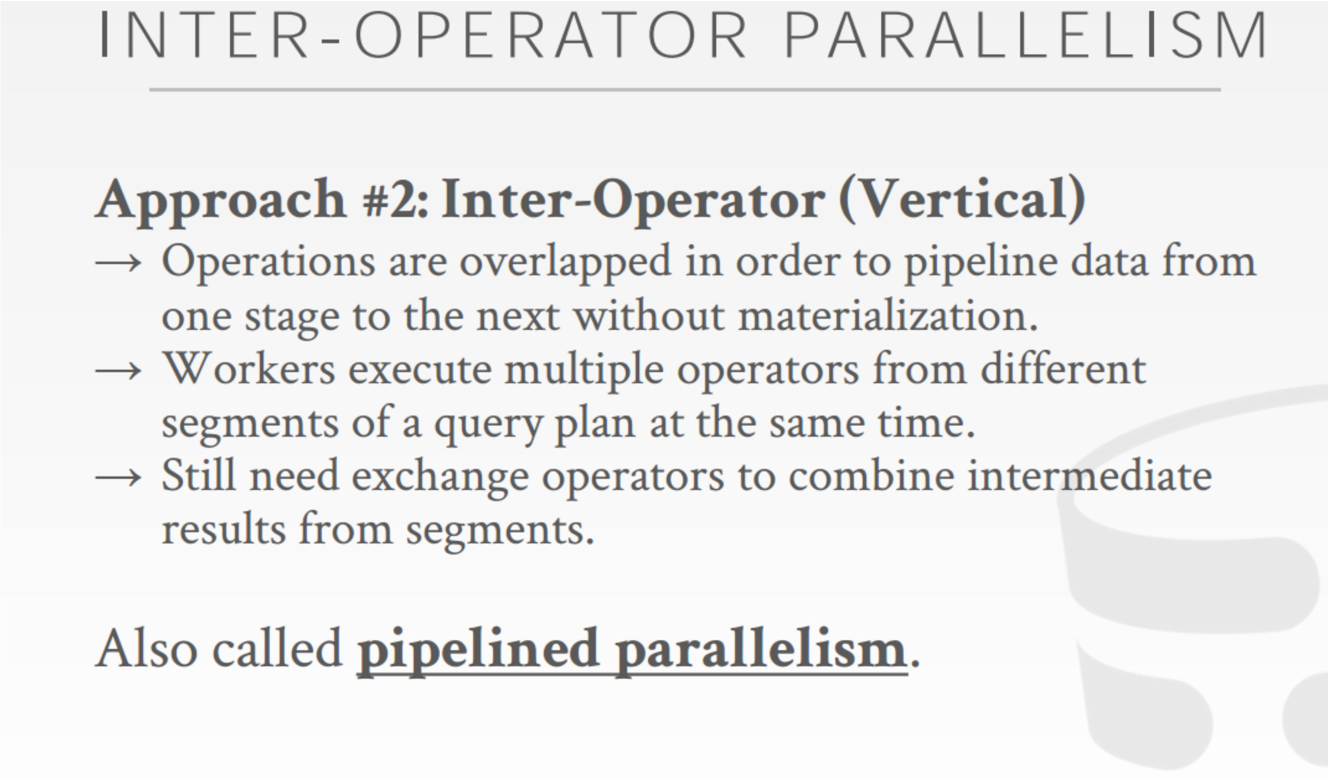


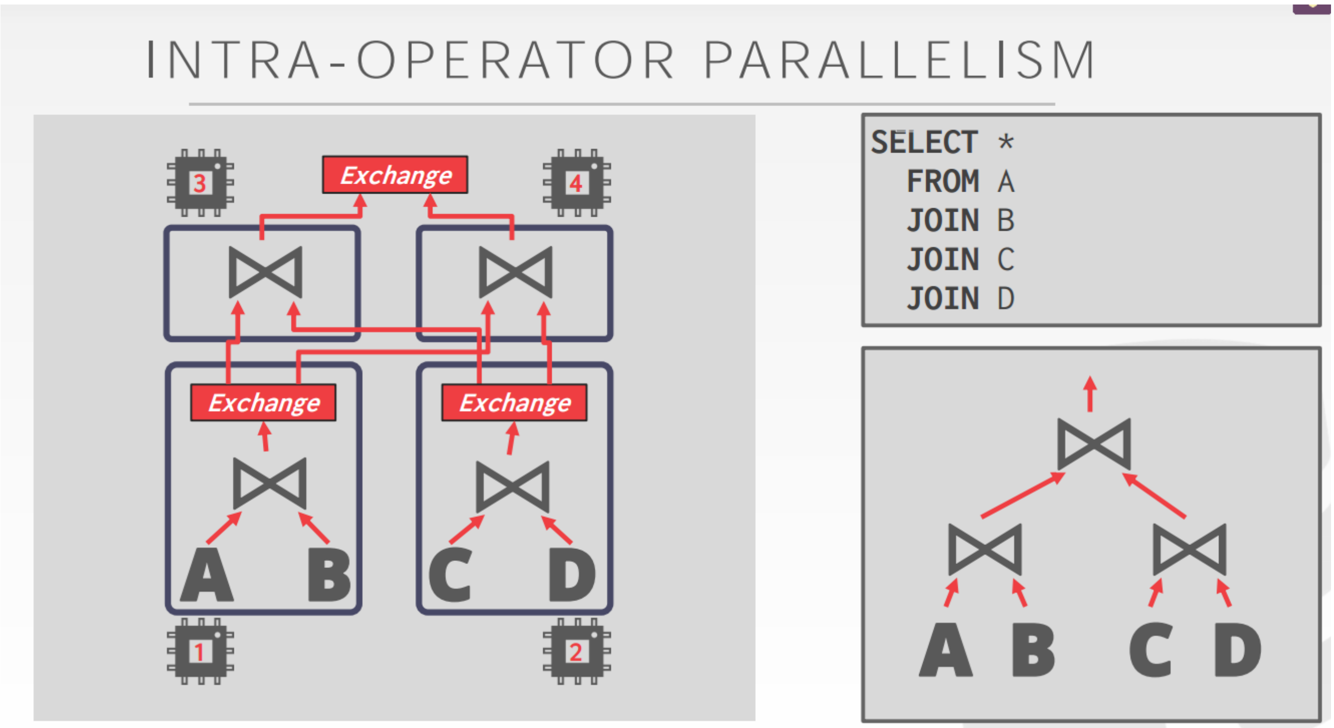


如上图，一个operator的任务可以有多个线程并发执行，DBMS使用`exchange`operator

INTER-OPERATOR

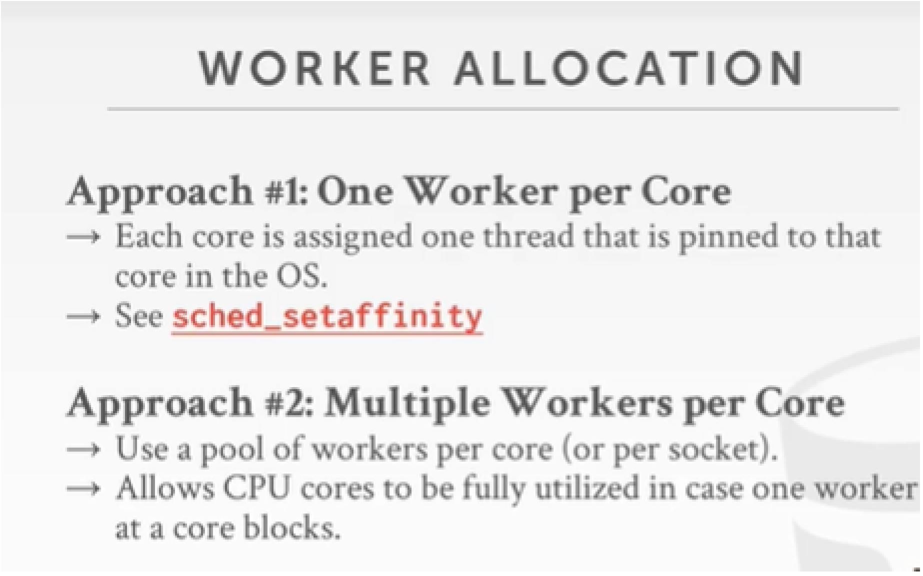
流水线化的并行





那如何给operator分配worker数量?

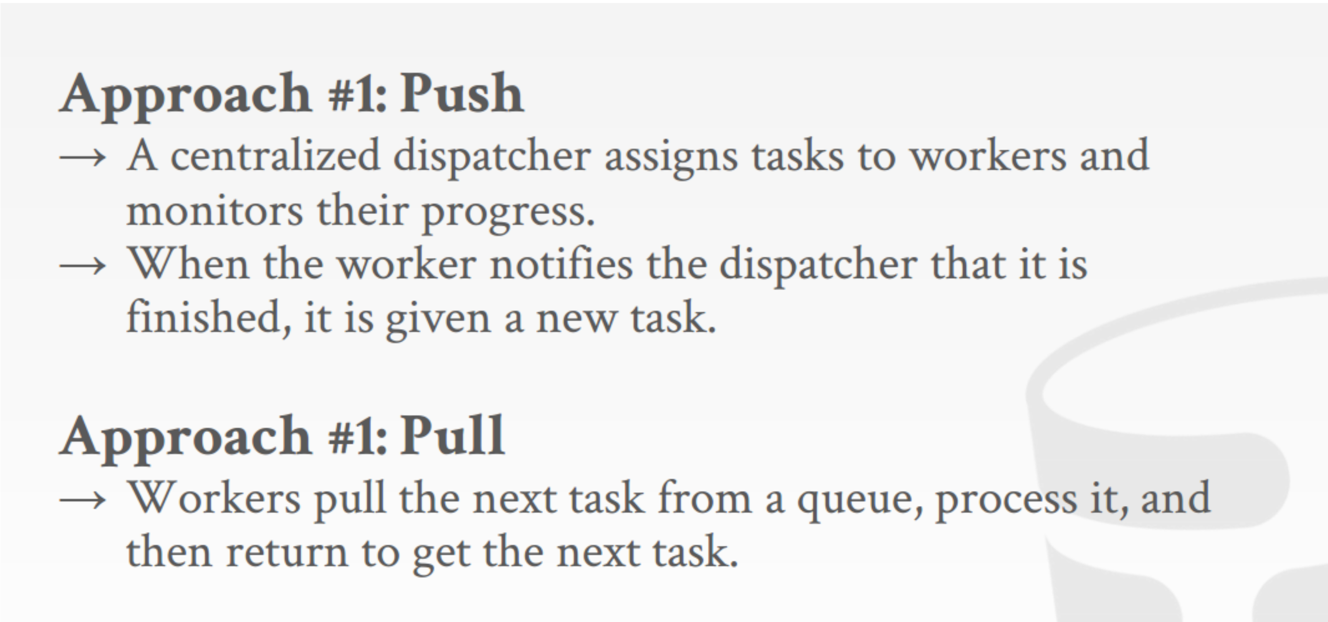
WORKER ALLOCATION



One Worker per Core

Multiple Workers per Core

TASK ASSIGNMENT



总结而言对于OLAP系统来说，vectorized/bottom-up的执行策略是最佳的。