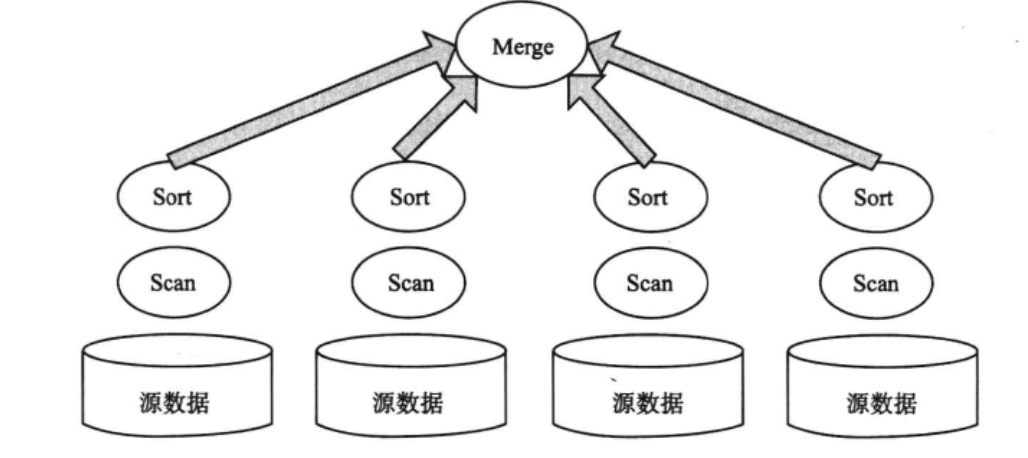
# 概述

# 架构

并行数据库往往采用MPP（Massively Parallel Processing，大规模并行处理）架构。MPP架构是一种不共享的结构，每个节点可以运行自己的操作系统、数据库等。每个节点内的CPU不能访问另一个节点的内存，节点之间的信息交互是通过节点互联网络实现的。

如图所示，将数据分布到多个节点，每个节点扫描本地数据，并由Merge操作符执行结果汇总。

**MPP Merge操作符：**



常见的数据分布算法有两种：

范围分区（Range Partitioning）：按照范围划分数据。

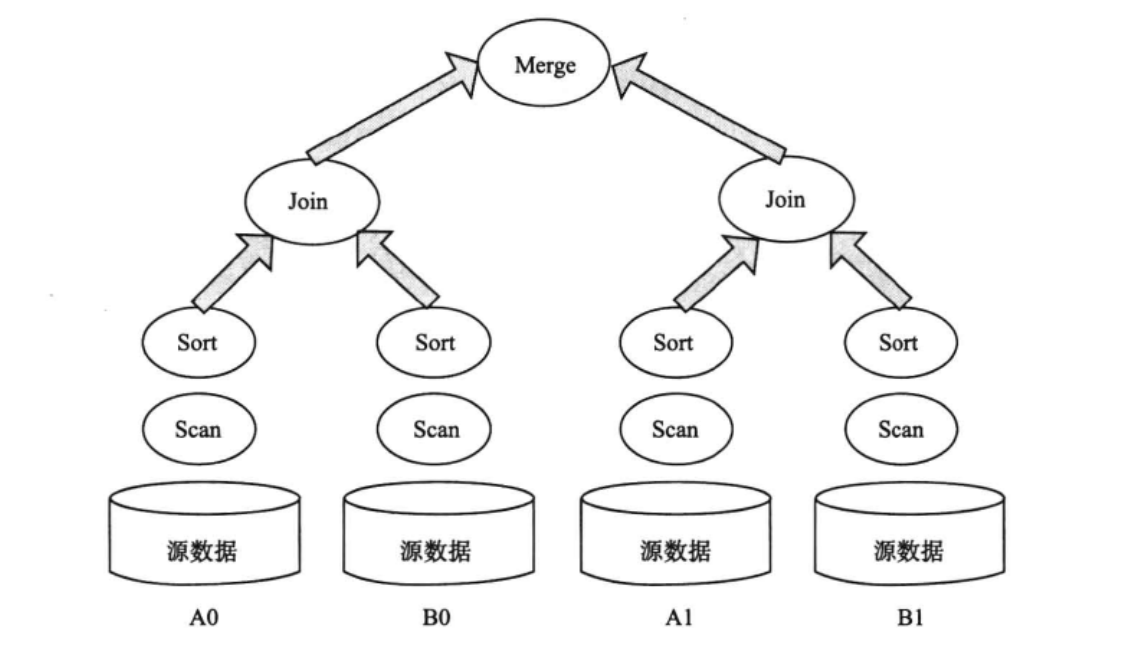
哈希分区（Hashing）：根据哈希函数计算结果将每个元祖分配给相应的节点。

Merge操作符：系统中存在一个或多个合并节点，它会发送命令给每个数据分片请求相应的数据，每个数据分片所在的节点扫描本地数据，排序后回复合并节点，由合并节点通过merge操作符执行数据汇总。**Merge操作符是一个统称，涉及的操作可能是limit、order by、group by、join等**。这个过程相当于执行一个Reduce任务个数为1的MapReduce作业，不同的是，这里不需要考虑执行过程中服务器出现故障的情况。

如果Merge节点处理的数据量特别大，可以通过Split操作符将数据划分到多个节点，每个节点对一部分数据执行group by、join等操作后再合并最终结果。

如图所示，假如需要执行“select \* from A,B where A.x=B.y”，可以分别根据A.x和B.x的哈希值将表A和B划分为A0、A1和B0、B1。由两个节点分别对A0、B0以及A1、B1执行join操作后再合并join结果。

**MPP Split操作符：**



并行数据库的SQL查询和MapReduce计算有些类似，可以认为MapReduce模型是一种更高层次的抽象。由于考虑问题的角度不同，并行数据库处理的SQL查询执行时间通常很短，出现异常时整个操作重做即可，不需要像MapReduce实现那样引入一个主控节点管理计算节点，监控计算节点故障，启动备份任务等。

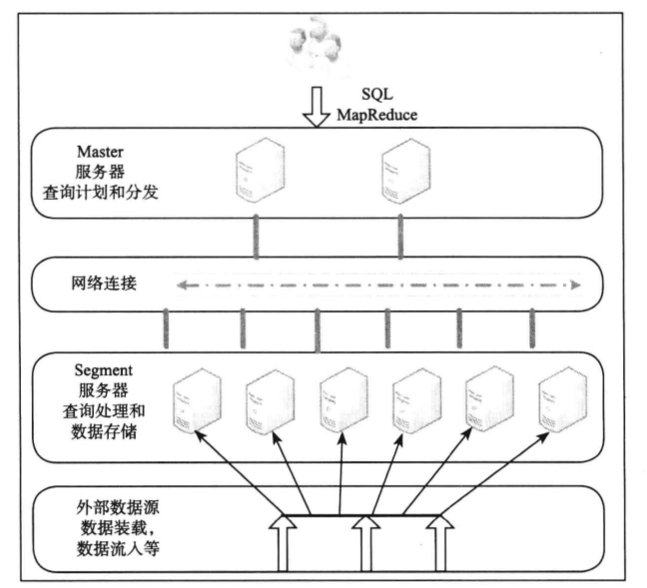
# 分类

## EMC Greenplum

Greenplum是EMC公司研发的一款采用MPP架构的OLAP产品，底层基于开源的PostgreSQL数据库。

### 整体架构

Greenplum系统主要包含两种角色：Master服务器（Master Server）和Segment服务器（Segment Server）。在Greenplum中每个表都是分布在所有节点上的。Master服务器首先对表的某个或多个列进行哈希运算，然后根据哈希结果将表的数据分布到Segment服务器中。整个过程中Master服务器不存放任何用户数据，只是对客户端进行访问控制和存储表分布逻辑的元数据。



Greenplum支持两种访问方式：SQL和MapReduce。用户将SQL操作语句发送给Master服务器，由Master服务器执行词法解析、语法分析，生成执行计划树，并将查询请求分发给多台Segment服务器。每个Segment服务器返回部分结果后，Master服务器会进行聚合并将最终结果返回给用户。除了高效查询，Greenplum还支持通过数据的秉性装载，将外部数据秉性装载到所有的Segment服务器。

### 并行查询优化器

Greenplum的并行查询优化器负责将用户的SQL或者MapReduce请求转换为物理执行计划。Greenplum采用基于代价的查询优化算法（cost-based optimization），从各种可能的查询计划中选择一个代价最小的。Greenplum优化器会考虑集群全局统计信息，例如数据分布，另外，除了考虑单机执行的CPU、内存资源消耗，还需要考虑数据的网络传输开销。

Greenplum除了生成传统关系数据库的物理运算符，包括表格扫描（Scan）、过滤（Filter）、聚集（Aggregation）、排序（Sort）、联表（Join），还会生成一些并行运算符，用来描述查询执行过程中如何在节点之间传输数据。

广播（Brodcast，N:N）：每个极端节点将目标数据发送给所有其他节点。

重新分布（Redistribute，N:N）：类似MapReduce中的shuffle过程，每个计算节点将目标数据重新哈希后分散到所有其他节点。

汇总（Gather，N:1）：所有的计算节点将目标数据发送给某个节点（一般为Master服务器）。

## HP Vertica

Vertica是Michael Stonebraker的学术研究项目C-Store的商业版本，并最终被惠普公司收购。Vertica在架构上与OceanBase有相似之处。

### 混合存储模型

Vertica的数据包含两个部分：ROS（Read-Optimized Storage）以及WOS（Write-Optimized Storage），WOS的数据在内存中且不排序和加索引，ROS的数据在磁盘中有序且压缩存储。后台的“TUPLE MOVER”会不断地将数据从WOS读出并网ROS更新（同时完成排序和索引）。Vertica的折中设计和OceanBase很相似，ROS对应OceanBase中的ChunkServer，WOS对应OceanBase中的UpdateServer。由于后台采用“BULK”的方式批量更新，性能非常好。

### 多映射（Projections）存储

### 列式存储

Vertica中的每一列数据独立存储在磁盘的连续块上。查询数据时，Vertica只需要读取那些需要的列，而不是被选择的行的所有的列数据。

### 压缩技术

## Google Dremel

### 系统架构

### Dremel vs MapReduce