MPP大规模并行处理架构详解

面试官：说下你知道的 MPP 架构的计算引擎?

这个问题不少小伙伴在面试时都遇到过，因为对 MPP 这个概念了解较少，不少人都卡壳了，但是我们常用的大数据计算引擎有很多都是 MPP 架构的，*像我们熟悉的 Impala、ClickHouse、Druid、Doris 等都是 MPP 架构*。

采用 MPP 架构的很多 OLAP 引擎号称：**亿级秒开**。

> 本文分为三部分讲解，第一部分**详解 MPP 架构**，第二部分剖析 **MPP 架构与批处理架构的异同点**，第三部分是**采用 MPP 架构的 OLAP 引擎介绍**。

一、MPP 架构

MPP 是系统架构角度的一种服务器分类方法。

目前商用的服务器分类大体有三种：

1. **SMP（对称多处理器结构）**
2. **NUMA（非一致存储访问结构）**
3. ***MPP（大规模并行处理结构）***

我们今天的主角是 MPP，因为*随着分布式、并行化技术成熟应用，MPP 引擎逐渐表现出强大的高吞吐、低时延计算能力，有很多采用 MPP 架构的引擎都能达到"亿级秒开"*。

先了解下这三种结构：

1. SMP

即对称多处理器结构，就是指服务器的多个 CPU 对称工作，无主次或从属关系。**SMP 服务器的主要特征是共享**，系统中的所有资源（如 CPU、内存、I/O 等）都是共享的。也正是由于这种特征，导致了 SMP 服务器的主要问题，即**扩展能力非常有限**。

2. NUMA

即非一致存储访问结构。这种结构就是为了解决 SMP 扩展能力不足的问题，利用 NUMA 技术，可以把几十个 CPU 组合在一台服务器内。NUMA 的基本特征是拥有多个 CPU 模块，节点之间可以通过互联模块进行连接和信息交互，所以，**每个 CPU 可以访问整个系统的内存**（这是与 MPP 系统的重要区别）。但是访问的速度是不一样的，因为 CPU 访问本地内存的速度远远高于系统内其他节点的内存速度，这也是非一致存储访问 NUMA 的由来。

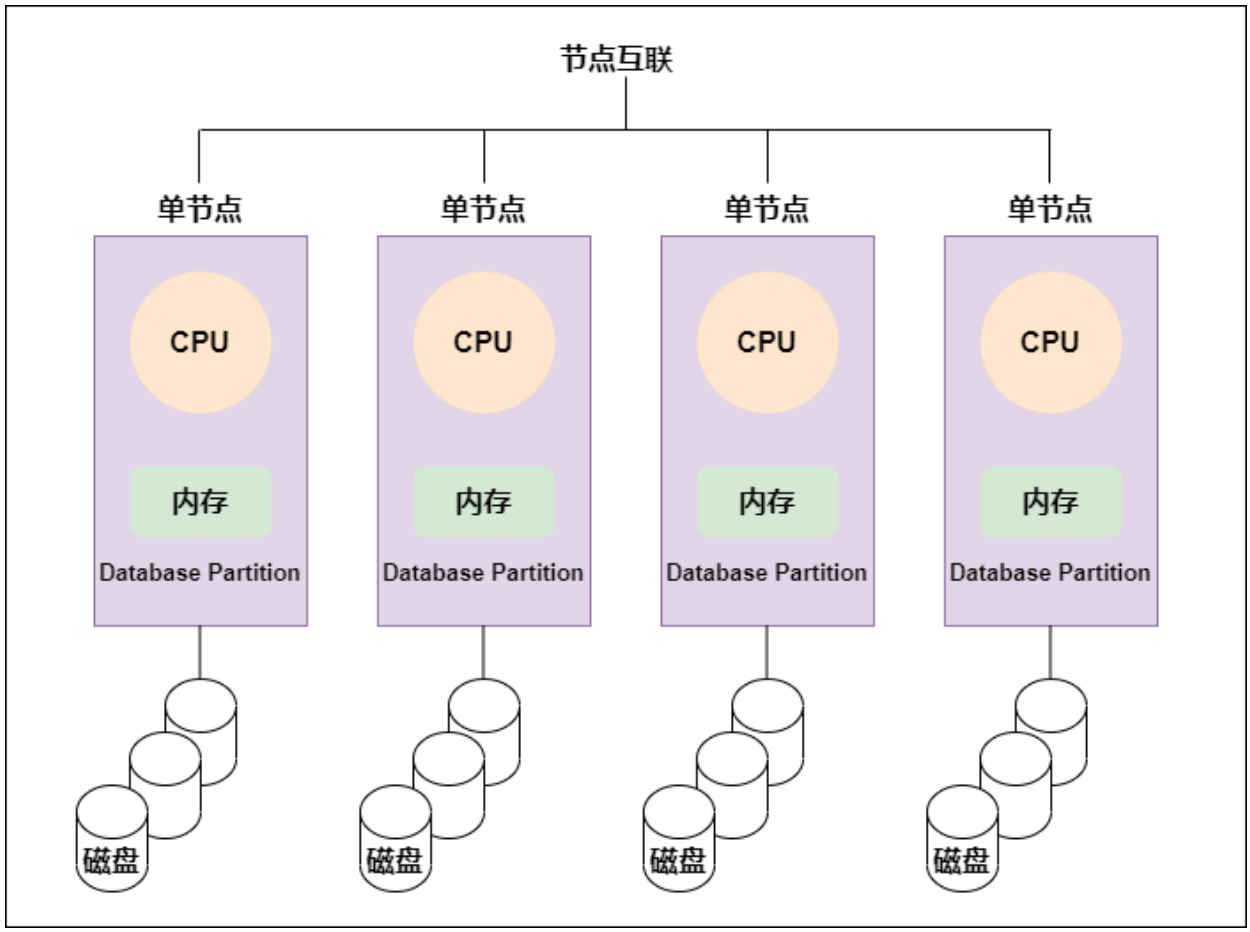
这种结构也有一定的缺陷，由于访问异地内存的时延远远超过访问本地内存，因此，当 CPU 数量增加时，系统性能无法线性增加。

3. MPP

即大规模并行处理结构。MPP 的系统扩展和 NUMA 不同，*MPP 是由多台 SMP 服务器通过一定的节点互联网络进行连接*，协同工作，完成相同的任务，从用户的角度来看是一个服务器系统。每个节点只访问自己的资源，所以是一种**完全无共享（Share Nothing）结构**。

MPP 结构扩展能力最强，理论可以无限扩展。由于 MPP 是多台 SPM 服务器连接的，每个节点的 CPU 不能访问另一个节点内存，所以也不存在异地访问的问题。

MPP 架构图：



MPP 架构

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，这个过程称为**数据重分配**。

但是 MPP 服务器需要一种复杂的机制来调度和平衡各个节点的负载和并行处理过程。目前，一些基于 MPP 技术的服务器往往通过系统级软件（如数据库）来屏蔽这种复杂性。举个例子，Teradata 就是基于 MPP 技术的一个关系数据库软件（这是最早采用 MPP 架构的数据库），基于此数据库来开发应用时，不管后台服务器由多少节点组成，开发人员面对的都是同一个数据库系统，而无需考虑如何调度其中某几个节点的负载。

MPP 架构特征：

- **任务并行执行；**

- **数据分布式存储(本地化);**
- **分布式计算;**
- **高并发，单个节点并发能力大于 300 用户;**
- **横向扩展，支持集群节点的扩容;**
- **Shared Nothing（完全无共享）架构。**

NUMA 和 MPP 区别：

二者有许多相似之处，首先 NUMA 和 MPP 都是由多个节点组成的；其次每个节点都有自己的 CPU，内存，I/O 等；都可以都过节点互联机制进行信息交互。

那它们的区别是什么呢，首先是**节点互联机制不同**，NUMA 的节点互联是在同一台物理服务器内部实现的，MPP 的节点互联是在不同的 SMP 服务器外部通过 I/O 实现的。

其次是**内存访问机制不同**，在 NUMA 服务器内部，任何一个 CPU 都可以访问整个系统的内存，但异地内存访问的性能远远低于本地内存访问，因此，在开发应用程序时应该尽量避免异地内存访问。而在 MPP 服务器中，每个节点只访问本地内存，不存在异地内存访问问题。

二、批处理架构和 MPP 架构

批处理架构（如 MapReduce）与 MPP 架构的异同点，以及它们各自的优缺点是什么呢?

相同点：

首先相同点，**批处理架构与 MPP 架构都是分布式并行处理**，将任务并行的分散到多个服务器和节点上，在每个节点上计算完成后，将各自部分的结果汇总在一起得到最终的结果。

不同点：

批处理架构和 MPP 架构的不同点可以举例来说：我们执行一个任务，首先这个任务会被分成多个 task 执行，**对于 MapReduce 来说，这些 tasks 被随机的分配在空闲的 Executor 上；而对于 MPP 架构的引擎来说，每个处理数据的 task 被绑定到持有该数据切片的指定 Executor 上**。

正是由于以上的不同，使得两种架构有各自优势也有各自缺陷：

- ***批处理的优势：***

对于批处理架构来说，**如果某个 Executor 执行过慢，那么这个 Executor 会慢慢分配到更少的 task 执行**，批处理架构有个*推测执行*策略，推测出某个 Executor 执行过慢或者有故障，则在接下来分配 task 时就会较少的分配给它或者直接不分配。

- ***MPP 的缺陷：***

对于 MPP 架构来说，因为 task 和 Executor 是绑定的，如果某个 Executor 执行过慢或故障，将会导致**整个集群的性能就会受限于这个故障节点的执行速度**(所谓木桶的短板效应)，所以 MPP 架构的最大缺陷就是——**短板效应**。另一点，集群中的节点越多，则某个节点出现问题的概率越大，而一旦有节点出现问题，对于 MPP 架构来说，将导致整个集群性能受限，所以一般实际生产中 **MPP 架构的集群节点不易过多**。

- ***批处理的缺陷：***

任何事情都是有代价的，对于批处理而言，**会将中间结果写入到磁盘中**，这严重限制了处理数据的性能。

- ***MPP 的优势：***

MPP 架构不需要将中间数据写入磁盘，因为一个单一的 Executor 只处理一个单一的 task，因此可以简单直接将数据 stream 到下一个执行阶段。这个过程称为`pipelining`，它提供了很大的性能提升。

举个例子：要实现两个大表的 join 操作，对于批处理而言，如 Spark 将会写磁盘 3 次(第一次写入：表 1 根据`join key`进行 shuffle；第二次写入：表 2 根据`join key`进行 shuffle；第三次写入：Hash 表写入磁盘)， 而 MPP 只需要一次写入(Hash 表写入)。**这是因为 MPP 将 mapper 和 reducer 同时运行，而 MapReduce 将它们分成有依赖关系的 tasks(DAG),这些 task 是异步执行的，因此必须通过写入中间数据共享内存来解决数据的依赖**。

批处理架构和 MPP 架构融合：

两个架构的优势和缺陷都很明显，并且它们有互补关系，如果我们能将二者结合起来使用，是不是就能发挥各自最大的优势。目前批处理和 MPP 也确实正在逐渐走向融合，也已经有了一些设计方案，一旦技术成熟，可能会风靡大数据领域，我们拭目以待！

三、 MPP 架构的 OLAP 引擎

采用 MPP 架构的 OLAP 引擎有很多，下面只选择常见的几个引擎对比下，可为公司的技术选型提供参考。

采用 MPP 架构的 OLAP 引擎分为两类，一类是自身不存储数据，只负责计算的引擎；一类是自身既存储数据，也负责计算的引擎。

1）只负责计算，不负责存储的引擎

1. Impala

Apache Impala 是采用 MPP 架构的查询引擎，本身不存储任何数据，**直接使用内存进行计算**，兼顾数据仓库，具有实时，批处理，多并发等优点。

提供了类 SQL（类 Hsql）语法，在多用户场景下也能拥有较高的响应速度和吞吐量。它是由 Java 和 C++实现的，Java 提供的查询交互的接口和实现，C++实现了查询引擎部分。

Impala 支持共享 Hive Metastore，但没有再使用缓慢的 Hive+MapReduce 批处理，而是通过使用与商用并行关系数据库中类似的分布式查询引擎（由 Query Planner、Query Coordinator 和 Query Exec Engine 三部分组成），可以直接从 HDFS 或 HBase 中用 SELECT、JOIN 和统计函数查询数据，从而大大降低了延迟。

Impala 经常搭配存储引擎 Kudu 一起提供服务，这么做最大的优势是查询比较快，并且支持数据的 Update 和 Delete。

2. Presto

Presto 是一个分布式的采用 MPP 架构的查询引擎，本身并不存储数据，但是可以接入多种数据源，并且**支持跨数据源的级联查询**。Presto 是一个 OLAP 的工具，擅长对海量数据进行复杂的分析；但是对于 OLTP 场景，并不是 Presto 所擅长，所以不要把 Presto 当做数据库来使用。

Presto 是一个低延迟高并发的**内存计算引擎**。需要从其他数据源获取数据来进行运算分析，它可以连接多种数据源，包括 Hive、RDBMS（Mysql、Oracle、Tidb 等）、Kafka、MongoDB、Redis 等。

2）既负责计算，又负责存储的引擎

1. ClickHouse

ClickHouse 是近年来备受关注的开源**列式数据库**，主要用于数据分析（OLAP）领域。

它自包含了存储和计算能力，完全自主实现了高可用，而且*支持完整的 SQL 语法包括 JOIN 等*，技术上有着明显优势。相比于 hadoop 体系，以数据库的方式来做大数据处理更加简单易用，学习成本低且灵活度高。当前社区仍旧在迅猛发展中，并且在国内社区也非常火热，各个大厂纷纷跟进大规模使用。

ClickHouse 在计算层做了非常细致的工作，**竭尽所能榨干硬件能力，提升查询速度**。它实现了单机多核并行、分布式计算、向量化执行与 SIMD 指令、代码生成等多种重要技术。

ClickHouse 从 OLAP 场景需求出发，定制开发了一套全新的高效列式存储引擎，并且**实现了数据有序存储、主键索引、稀疏索引、数据 Sharding、数据 Partitioning、TTL、主备复制等丰富功能**。以上功能共同为 ClickHouse 极速的分析性能奠定了基础。

2. Doris

Doris 是百度主导的，根据 Google Mesa 论文和 Impala 项目改写的一个大数据分析引擎，是一个**海量分布式 KV 存储系统**，其设计目标是支持中等规模高可用可伸缩的 KV 存储集群。

Doris 可以实现海量存储，线性伸缩、平滑扩容，自动容错、故障转移，高并发，且运维成本低。部署规模，建议部署 4-100+台服务器。

Doris3 的主要架构： DT（Data Transfer）负责数据导入、DS（Data Seacher）模块负责数据查询、DM（Data Master）模块负责集群元数据管理，数据则存储在 Armor 分布式 Key-Value 引擎中。Doris3 依赖 ZooKeeper 存储元数据，从而其他模块依赖 ZooKeeper 做到了无状态，进而整个系统能够做到无故障单点。

3. Druid

Druid 是一个**开源、分布式、面向列式存储的实时分析数据存储系统**。

Druid 的关键特性如下：

- **亚秒级的 OLAP 查询分析：采用了列式存储、倒排索引、位图索引等关键技术；**
- **在亚秒级别内完成海量数据的过滤、聚合以及多维分析等操作；**
- **实时流数据分析：Druid 提供了实时流数据分析，以及高效实时写入；**
- **实时数据在亚秒级内的可视化；**
- **丰富的数据分析功能：Druid 提供了友好的可视化界面；**

- **SQL 查询语言；**
- **高可用性与高可拓展性：Druid 工作节点功能单一，不相互依赖；Druid 集群在管理、容错、灾备、扩容都很容易；**

4. TiDB

TiDB 是 PingCAP 公司自主设计、研发的开源**分布式关系型数据库**，是一款同时支持 OLTP 与 OLAP 的融合型分布式数据库产品。

TiDB 兼容 MySQL 5.7 协议和 MySQL 生态等重要特性。目标是为用户提供一站式 OLTP 、OLAP 、HTAP 解决方案。**TiDB 适合高可用、强一致要求较高、数据规模较大**等各种应用场景。

5. Greenplum

Greenplum 是在开源的 PostgreSQL 的基础上采用了 MPP 架构的性能非常强大的**关系型分布式数据库**。为了兼容 Hadoop 生态，又推出了 HAWQ，分析引擎保留了 Greenplum 的高性能引擎，下层存储不再采用本地硬盘而改用 HDFS，规避本地硬盘可靠性差的问题，同时融入 Hadoop 生态。

3）常用的引擎对比

一张图总结下常用的 OLAP 引擎对比：

开源OLAP引擎	优点	缺点	易用性	自身存储
Impala	1. 基于内存，节省大量 I/O； 2. 直接访问HDFS或HBase数据作业调度，速度快。	1. 对内存依赖大； 2. 实践中，分区超过1万，性能严重下降； 3. 只能读取文本文件，而不能直接读取自定义二进制文件。	类SQL语法	否
Presto	1. 基于内存，节省大量 I/O； 2. 能够连接多个数据源，跨数据源连表查询。	1. 对内存依赖大； 2. 不支持UDF。	SQL标准	否
ClickHouse	1. 列存储； 2. 单机性能彪悍； 3. 保留明细； 4. 向量化引擎。	1. 集群在线扩展不佳； 2. 运维成本高。	非标协议接口	是
Doris	1. Mesa+Impala； 2. 主键更新； 3. 支持 Rollup table； 4. 高并发和高吞吐的Ad-hoc查询。	1. 成熟度不足； 2. 应用不广泛。	兼容MySQL协议	是
Druid	1. 实时数据摄入 2. 列式存储和位图索引 3. 多租户和高并发	1. OLAP性能分场景表现差异大； 2. 使用门槛高。	非标协议接口	是
TIDB	1. 100%OLTP + 80%OLAP； 2. 同时明细和举个查询； 3. 支持更新。	1. 非列存储； 2. OLAP能力不足。	SQL标准	是

常见 OLAP 引擎对比

- 发表于: 2021-05-282021-05-28 16:07:22
- 本文为 InfoQ 中文站特供稿件
- 首发地址：https://www.infoq.cn/article/0915a9fbbd1af7347a6deb0bf