MySQL 介绍

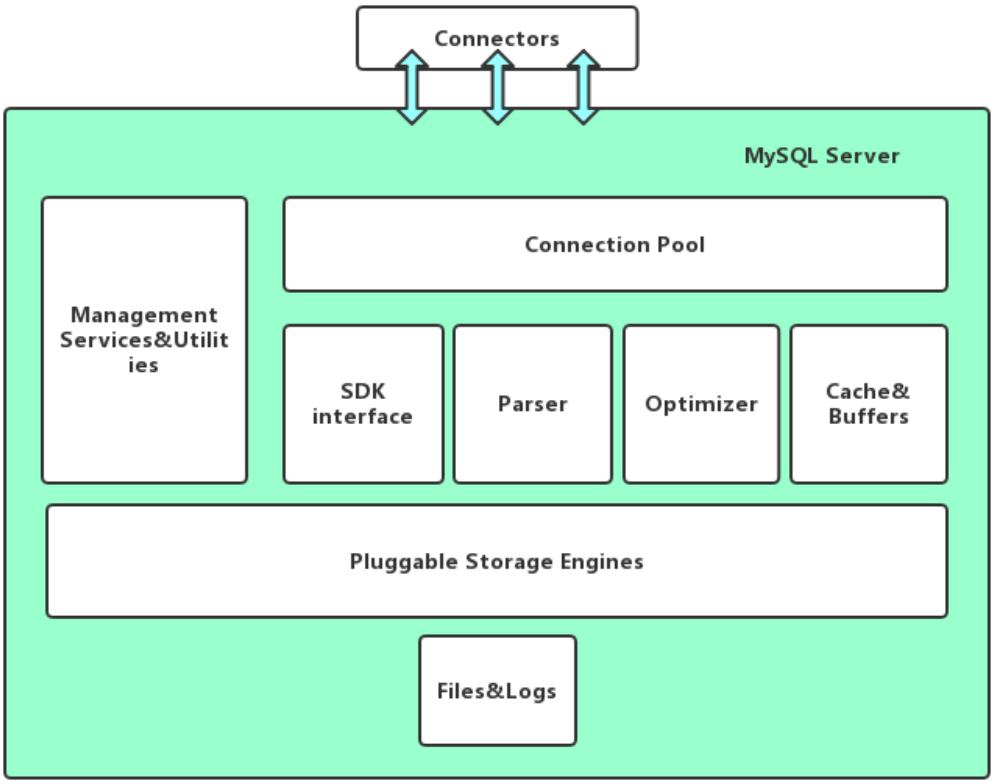
MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL 是最好的 RDBMS（Relational Database Management System，关系数据库管理系统）应用软件。

MySQL 是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

由于其社区版的性能卓越，搭配 PHP 和 Apache 可组成良好的开发环境。

MySQL 工作原理



图：MySQL 内部架构

MySQL 内部架构解析：

1. Connectors

客户端和连接服务，主要完成一些类似于连接处理、授权认证及相关的安全方案。与其

他编程语言

中的 SQL 语句进行交互，如 php、java 等。

2. Management Services & Utilities

系统管理和控制工具。

3. Connection Pool

连接池。作用：管理缓冲用户连接，线程处理等需要缓存的需求。

4. SQL Interface

SQL 接口。作用：接受用户的 SQL 命令，并且返回用户需要查询的结果。

5. Parser

解析器。作用：SQL 命令传递到解析器的时候会被解析器验证和解析。

主要功能：

1) 将 SQL 语句分解成数据结构，并将这个结构传递到后续步骤，后面 SQL 语句的传递和处理就是基于这个结构的；

2) 如果在分解构成中遇到错误，那么就说明这个 sql 语句是不合理的，语句将不会继续执行下去。

6. Optimizer

查询优化器。作用：SQL 语句在查询之前会使用查询优化器对查询进行优化（产生多种执行计划,最终数据库会选择最优化的方案去执行,尽快返回结果），采用“选取-投影-联接”策略进行查询。

7. Cache&Buffers

查询缓存。如果查询缓存有命中的查询结果，查询语句就可以直接去查询缓存中取数据。

这个缓存机制是由一系列小缓存组成的。比如表缓存，记录缓存，key 缓存，权限缓存等。

8. Engine

存储引擎。存储引擎是 MySQL 中具体的与文件打交道的子系统。也是 MySQL 最具有特色的一个地方。MySQL 的存储引擎是插件式的。它根据 MySQL AB 公司提供的文件访问层的一个抽象接口来定制一种文件访问机制（这种访问机制就叫存储引擎）。

存储引擎真正的负责了 MySQL 中数据的存储和提取，服务器通过 API 与存储引擎进行通信。不同的存储引擎具有的功能不同，这样我们可以根据自己的实际需要进行选取。

SQL 语句执行过程：

数据库通常不会被直接使用，而是由其他编程语言通过 SQL 语句调用 MySQL，由 MySQL 处理并返回执行结果。

首先程序的请求会通过 MySQL 的 Connectors 与其进行交互，收到请求后，会暂时存放在连接池（Connection Pool）中并由处理器（Management Services & Utilities）管理。当该请求从等待队列进入到处理队列，管理器会将该请求丢给 SQL 接口（SQL Interface）。SQL 接口接收到请求后，它会将请求进行 hash 处理并与缓存中的结果进行对比，如果完全匹配则通过缓存直接返回处理结果；否则，需要完整的走一趟流程：

1. 由 SQL 接口丢给后面的解释器（Parser），解释器会判断 SQL 语句正确与否，若正确则将其转化为数据结构。

2. 解释器处理完，便来到后面的优化器（Optimizer），它会产生多种执行计划，最终数据库会选择最优化的方案去执行，尽快返回结果。

3. 确定最优执行计划后，SQL 语句此时便可以交由存储引擎（Engine）处理，存储引擎将会到后端的存储设备中取得相应的数据，并原路返回给程序。

MySQL 集群

1. MySQL 集群

MySQL 集群本质上是一个存储方案，其具有不共享(shared-nothing)、分布式节点架构的特性，可以提较好的供容错性，并提高性能。

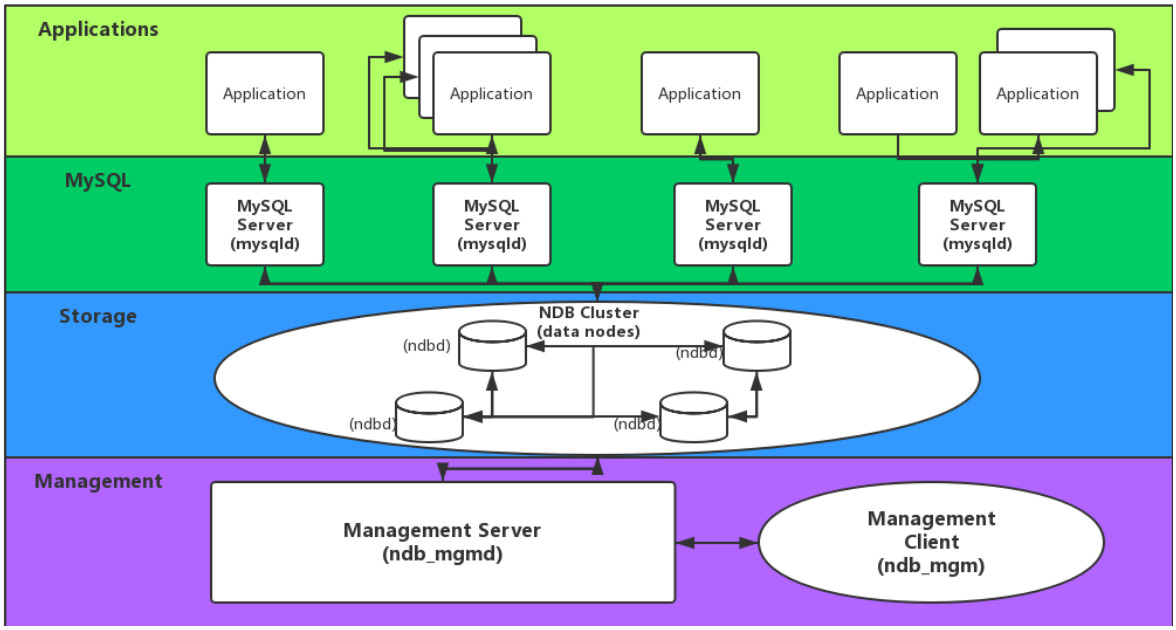
在数据更新的过程中， 使用 **读已提交隔离级别（read-committedisolation)** 保证所有节点的数据一致性，使用 **两阶段提交机制（two-phasedcommit)** 保证所有节点都有相同的数据（如果任何一个写操作失败，则更新失败）。

不共享的对等节点可以令集群中任意一台服务器上的更新操作，立即在其他服务器上可见。

在传播更新过程中使用一种复杂的通信机制，这一机制专用来提供跨网络的高吞吐量。

通过对多个 MySQL 服务器实现分配负载，从而最大程度地达到提高性能的目的，并通过在不

同位置存储数据以保证高可用性和冗余。



2. 数据存储

Mysql Cluster 采用同步复制实现数据节点组内的**主从同步**，以此来保证组内节点数据的一致性。同步过程主要通过 **两阶段提交机制** 实现：

第一阶段：准备提交

1) 执行提交语句时，事务将会被 Master(主)发送给所有的 Slave(从)，每个 Slave 开始准备提交事务。

2) Slave 准备事务完毕后向 Master 发送 OK 或 ABORT 消息，告知 Master 事务的准备情况。

第二阶段：通过收到的消息判断提交或中止

3) Master 接收所有 Slave 发送的 OK 或 ABORT 消息。

Master 如果收到所有 Slave 的 OK 消息，就会向所有 Slave 发送提交消息，执行提交事务；

Master 如果收到来自任何一个 Slave 的 ABORT 消息，就会向所有 Slave 发送 ABORT

消息，取消提交事务。

4) Slave 接收 Master 发送的提交或取消请求。

Slave 收到提交请求，执行提交事务，并向 Master 发送事务已提交的确认；

Slave 收到取消请求，则会撤销所有改变并释放所占用的资源，中止事务，并向 Master 发送事务已中止的确认。

5) Master 收到所有 Slave 的确认后，则将报告该事务被提交（或中止），然后继续处理下一个事务。

由上可知，同步复制共需 4 次消息传递，所以 Mysql Cluster 在数据更新的速度上比单机 Mysql 要慢。为了避免速度过慢，Mysql Cluster 一般要求运行在千兆以上的局域网内，节点可以采用双网卡，节点组之间采用直连方式进行连接。

每个节点组保存的数据是不同的，对 Cluster 进行扩容、增加数据节点组时，数据更新速度会变快，也可以减少锁定。

Mysql Cluster 中的索引列全部都存储于主存中，其他非索引列可以存储在内存中或通过建立表空间的方式存储到磁盘上。一旦出现数据改变(insert,update,delete 等)，改变的记录则会被写入重做日志，然后通过检查点定期将数据写入磁盘。

由于重做日志采取异步提交的形式进行提交，所以如果发生故障，在故障期间内或许会有少量事务丢失。为了避免这种情况，MySQL 集群提供延迟写入功能，通过配置延迟写入，就可以在故障发生时完成检查点写入，避免丢失最后一个检查点。由于采用同步数据复制，所以一般来说，如果单个数据节点发生故障，不会导致任何数据丢失。

3. MySQL 集群的优点和缺点

- **优点：**

高可用性

高吞吐量、低延迟

自动失效切换速度快

支持在线扩容，可扩展性强

灵活的分布式体系结构，没有单点故障

- **缺点：**

  备份、恢复不方便

  部署、管理、配置很复杂

  占用磁盘空间大，内存大

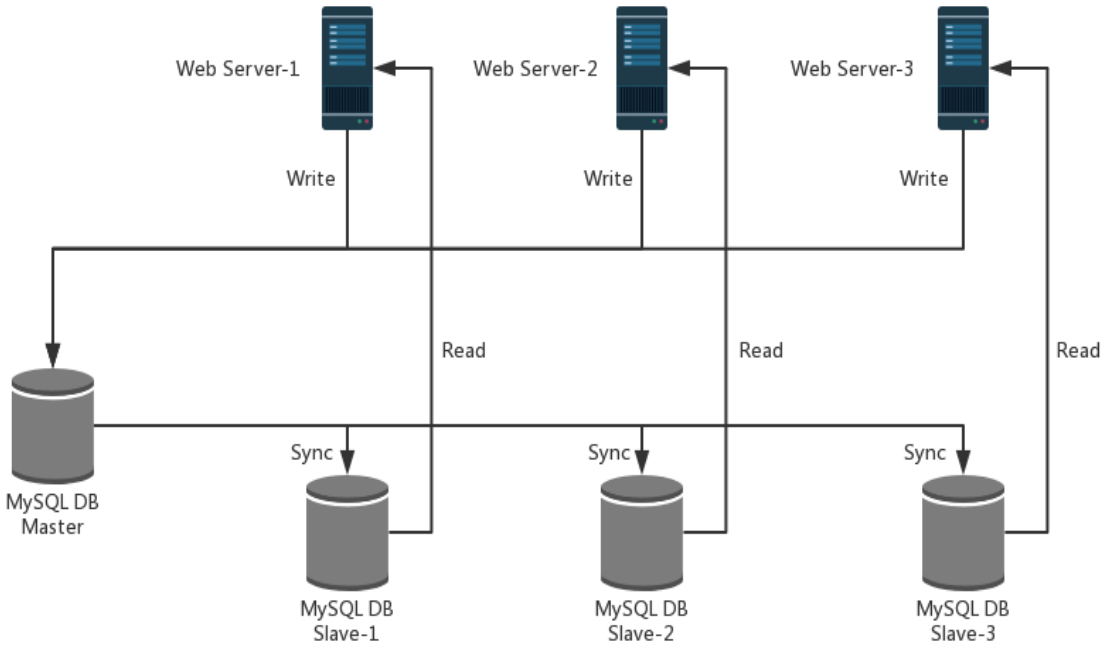
  存在很多限制，比如：不支持外键

  重启时，数据节点将数据 load 到内存需要很长时间

读写分离

一台 Mysql 作为独立的数据库无法满足实际需求。在实际生产环境中，不论是安全性，高可用性，还是高并发性等各个方面，单独一台 Mysql 都不足以满足业务需求。

故而现在的做法多为：通过主从复制(Master-Slave)的方式来实现数据同步，再通过读写分离(MySQL-Proxy)来提升数据库的并发负载能力。使用这种方式进行开发部署，可以较好的解决业务需求。



读写分离的原理：

让 Mater 处理增、改、删(INSERT、UPDATE、DELETE)操作，而 Slave 处理查询(SELECT)操作。

可以通过 MySQL-Proxy 实现读写分离，进行 MySQL-Proxy 读写分离至少需要有以下配置：

1. 数据库 Master 主服务器 x1
2. 数据库 Slave 从服务器 x1
3. MySQL-Proxy 调度服务器 x1

通过以下操作，可以实现读写分离，下列操作都是在 MySQL-Proxy 调度服务器上进行的。

1. MySQL 的安装与配置
2. 检查系统所需软件包
3. 编译安装 Lua
4. 安装配置 MySQL-Proxy
5. 配置并使用 rw-splitting.lua 读写分离脚本

完成以上操作后，即可测试读写分离效果。暂时关闭主从复制功能，连接 MySQL-Proxy，插

入任意记录。查询将发现没有新记录。分别登录到主从服务器查询，发现主服务器有插入记录，从服务器没有。通过验证，MySQL 的读写分离已经达成，所有的写操作均在 Master 完成。这样做，就避免了多个数据库同时写入可能造成的数据库不一致性。

此外，所有的读操作分摊给了各个 SIave，以此减轻数据库压力。

跨地区容灾

这里简单介绍一下使用双机热备进行跨地区容灾。

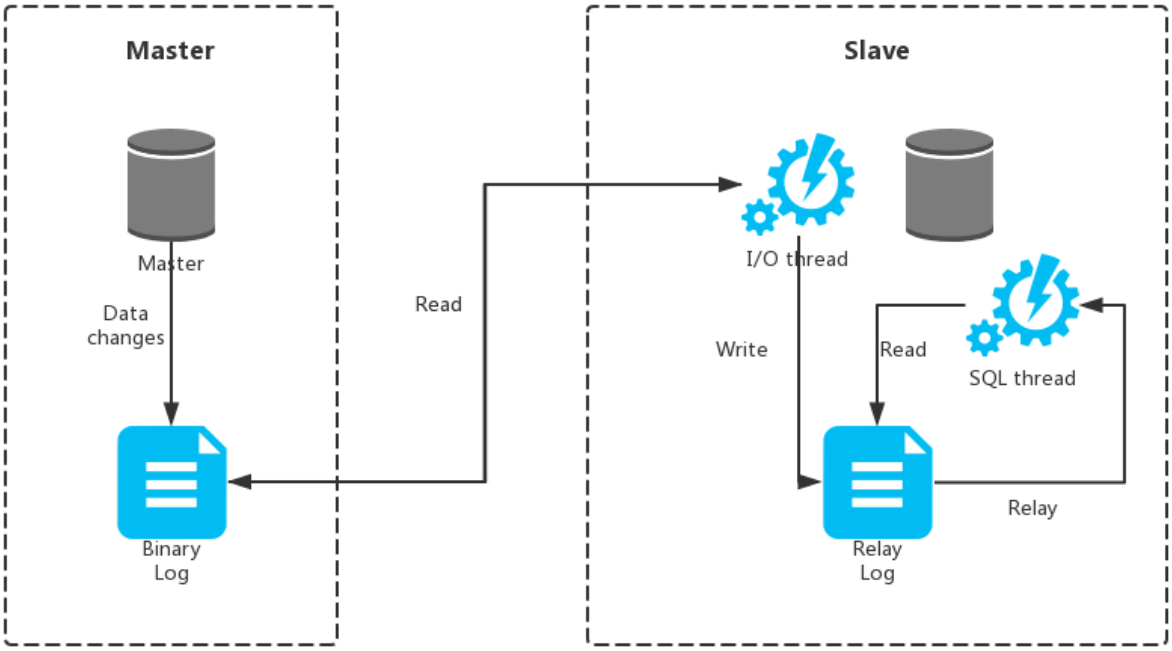
双机热备，简单来讲，就是令两个数据库保持状态自动同步。对于处于双机热备当中的数据库，对其中任何一个进行操作都可以自动同步到另外一个，这样，就保持两个数据库当中的数据始终一致。

这种方式有两个好处：

1. 异地容灾，其中一个坏了可以切换到另一个。
2. 负载均衡，可以将请求分摊到其中任何一台上，提高网站吞吐量。

备份工作原理：

两个原始数据相同的数据库 A 和 B，A 数据库中执行过的 sql 语句在 B 数据库里也同步执行一遍，通过这样，A、B 数据库就可以一直保持同步。



上图所示是一个主从复制的图示，演示了从一个主服务器(Master)把数据同步到从服务器(Slave)的过程。双机热备中，再做一次从服务器到主服务器的数据同步即完成了主主互相复制。

对于一个 Mysql 服务器， 一般有两个线程负责复制和被复制。当开启复制之后：

1. 主服务器 Master，把所有改动都写入 Binary Log 中。

2. 从服务器 Slave 读取 Master 的 Binary Log，写入中继日志 Relay Log，而后本地的 sql 线程会负责读取中继日志 Relay Log，并执行一遍。

至此，主服务器上的更改就同步到从服务器上了。

在 Mysql 中可以查看当前服务器的主、从状态，查看当前服务器的 Binary（作为主服务器时）状态和位置以及其 Relay Log（作为从服务器时）的复制进度和状态。