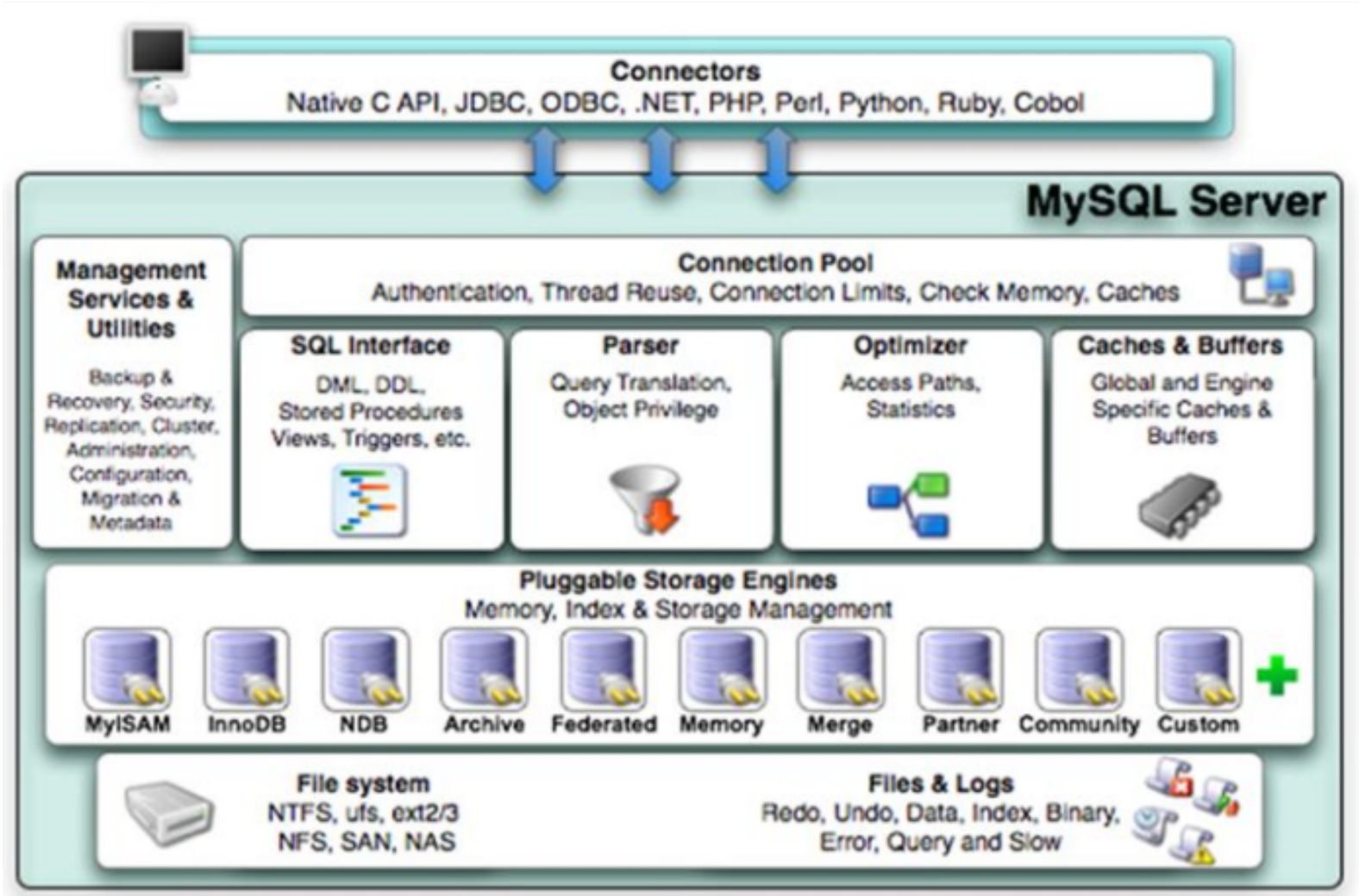
MySQL运行机制原理&架构

1.MySQL知识普及：

> MySQL是一个开放源代码的关系数据库管理系统。
>
> MySQL架构可以在多种不同场景中应用并发挥良好作用。主要体现在存储引擎的架构上，插件式的存储引擎架构将查询处理和其它的系统任务以及数据的存储提取相分离。

2.MySQL逻辑架构：



MySQL Server 逻辑架构图

1).最上层：

> 最上层是一些客户端和连接服务，包含本地的sock通信和大多数基于客户端/服务端工具实现的类似于tcp/ip的通信，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，在该层上引用了线程池的概念，为通过认证安全接入的客户端提供线程。同样在该层上可以实现基于ssl的安全链接。服务器也会为安全接入的每个客户端验证它所具

有的操作权限。

2).第二层:

第二层架构主要完成大多数的核心服务功能。如sql接口，并完成缓存的查询。sql的分析和优化 以及部分内置函数的执行。所有跨存储引擎的功能也在这一层实现，如过程，函数等。在该层，服务器会解析查询并创建相应的内部解析树，并对其完成相应的优化如确定查询表的顺序，是否利用索引等。最后生成相应的执行操作。如select语句，服务器还会查询内部的缓存。如果缓存空间足够大，这样就解决大量读操作的环境中能够很好的提升系统的性能。

3).存储引擎层:

存储引擎真正的负责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进行选取。

4).数据存储层:

主要是将数据存储在运行于裸设备的文件系统之上，并完成于存储引擎的交互。

3.并发控制和锁的概念:

当数据库中有多个操作需要修改同一数据时，不可避免的会产生数据的脏读。这时就需要数据库具有良好的并发控制能力，这一切在MySQL中都是由服务器和存储引擎来实现的。

解决并发问题最有效的方案是引入了锁的机制，锁在功能上分为共享锁(shared lock)和排它锁(exclusive lock)即通常说的读锁和写锁。当一个select语句在执行时可以施加读锁，这样就可以允许其它的select操作进行，因为在这个过程中数据信息是不会被改变的这样就能够提高数据库的运行效率。当需要对数据更新时，就需要施加写锁了，不在允许其它的操作进行，以免产生数据的脏读和幻读。锁同样有粒度大小，有表级锁(table lock)和行级锁(row lock)，分别在数据操作的过程中完成行的锁定和表的

锁定。这些根据不同的存储引擎所具有的特性也是不一样的。

MySQL大多数事务型的存储引擎都不只是简单的行级锁，基于性能的考虑，他们一般在行级锁基础上实现了多版本并发控制(MVCC)。这一方案也被Oracle等主流的关系数据库采用。它是通过保存数据中某个时间点的快照来实现的，这样就保证了每个事务看到的数据都是一致的。详细的实现原理可以参考《高性能MySQL》第三版。

4.事务：

简单的说事务就是一组原子性的SQL语句。可以将这组语句理解成一个工作单元，要么全部执行要么都不执行。默认MySQL中自动提交时开启的（start transaction）

操作事务：

```
1start transaction;
2select ...
3update ...
4insert ...
5commit;
```

注意：默认MySQL中自动提交是开启的：

```
mysql> show variables like 'autocommit';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit    | ON    |
+---------------+-------+
1 row in set (0.00 sec)
```

事务具有ACID的特性：

原子性：

事务中的所有操作要么全部提交成功，要么全部失败回滚

比如你从取款机取钱,这个事务可以分成两个步骤:1划卡,2出钱.不可能划了卡,而钱却没出来.这两步必须同时完成.要么就不完成.

> 一致性:
>
> > 数据库总是从给一个一致性的状态转换到另一个一致性的状态
> >
> > 例如,完整性约束了a+b=10,一个事务改变了a,那么b也应该随之改变.不管数据怎么改变。一定是符合约束
>
> 隔离性:
>
> > 一个事务所做的修改在提交之前对其它事务是不可见的
> >
> > 两个以上的事务不会出现交错执行的状态.因为这样可能会导致数据不一致.
>
> 持久性:
>
> > 一旦事务提交，其所做的修改便会永久保存在数据库中。

事务的隔离级别:

> READ UNCOMMITTED(读未提交):
>
> > 事务中的修改即使未提交也是对其它事务可见
>
> READ COMMITTED(读提交):
>
> > 事务提交后所做的修改才会被另一个事务看见，可能产生一个事务中两次查询的结果不同。
>
> REPEATABLE READ(可重读):
>
> > 只有当前事务提交才能看见另一个事务的修改结果。解决了一个事务中两次查询的结果不同的问题。
>
> SERIALIZABLE(串行化):
>
> > 只有一个事务提交之后才会执行另一个事务。

查询并修改隔离级别:

```
mysql> show variables like 'tx_isolation';
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| tx_isolation  | REPEATABLE-READ |
+---------------+-----------------+
1 row in set (0.00 sec)

mysql> set tx_isolation = 'READ-COMMITTED';
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like 'tx_isolation';
+---------------+----------------+
| Variable_name | Value          |
+---------------+----------------+
| tx_isolation  | READ-COMMITTED |
+---------------+----------------+
1 row in set (0.00 sec)
```

死锁：

> 两个或多个事务在同一资源上相互占用并请求锁定对方占用的资源，从而导致恶性循环的现象。
>
> 对于死锁的处理：MySQL的部分存储引擎能够检测到死锁的循环依赖并产生相应的错误。InnoDB引擎解决的死锁的方案是将持有最少写锁的事务进行回滚。
>
> > 为了提供回滚或者撤销未提交的变化的能力，许多数据源采用日志机制。例如：sql server使用一个预写事务日志，在将数据应用于（或提交到）实际数据页面前，先写在事务日志上。但是，其他一些数据源不是关系型数据库管理系统，他们管理未提交事务的方式完全不同。只要事务回滚时，数据源可以撤销所有未提交的改变，那么这种技术可用于事务管理。

5.MySQL存储引擎及应用方案：

> MySQL采用插件式的存储引擎的架构，可以根据不同的需求为不同的表设置不同的存储引擎。
>
> 如：

```
mysql> use mysql
Database changed
mysql> show table status like 'user'\G
*************************** 1. row ***************************
           Name: user
         Engine: MyISAM
        Version: 10
     Row_format: Dynamic
           Rows: 6
 Avg_row_length: 54
    Data_length: 328
Max_data_length: 281474976710655
   Index_length: 2048
      Data_free: 0
 Auto_increment: NULL
    Create_time: 2014-01-11 15:31:06
    Update_time: 2014-01-11 15:31:07
     Check_time: NULL
      Collation: utf8_bin
       Checksum: NULL
 Create_options:
        Comment: Users and global privileges
1 row in set (0.00 sec)
```

相关字段介绍：

Name：显示的是表名

Engine：显示存储引擎，该表存储引擎为MyISAM

Row_format：显示行格式，对于MyISAM有Dynamic、Fixed和Compressed三种。非别表示表中有可变的数据类型，表中数据类型为固定的，以及表是压缩表的环境。

Rows：显示表中行数

Avg_row_length：平均行长度（字节）

Data_length：数据长度（字节）

Max_data_length：最大存储数据长度（字节）

Data_free：已分配但未使用的空间，包括删除数据空余出来的空间

Auto_increment：下一个插入行自动增长字段的值

Create_time：表的创建时间

Update_time：表数据的最后修改时间

Collation：表的默认字符集及排序规则

> Checksum：如果启用，表示整个表的实时校验和
>
> Create_options：创建表示的一些其它选项
>
> Comment：额外的一些注释信息，根据存储引擎的不同表示的内容也不胫相同。

> 常用MySQL存储引擎介绍：
>
> > InnoDB引擎：
> >
> > > 将数据存储在表空间中，表空间由一系列的数据文件组成，由InnoDb管理
> > >
> > > 支持每个表的数据和索引存放在单独文件中(innodb_file_per_table)；
> > >
> > > 支持事务，采用MVCC来控制并发，并实现标准的4个事务隔离级别，支持外键。
> > >
> > > 索引基于聚簇索引建立，对主键查询有较高性能。
> > >
> > > 数据文件的平台无关性，支持数据在不同的架构平台移植
> > >
> > > 能够通过一些工具支持真正的热备，如XtraBackup等；
> > >
> > > 内部进行自身优化如采取可预测性预读，能够自动在内存中创建bash索引等
>
> MyISAM引擎：
>
> > MySQL5.1默认，不支持事务和行级锁
> >
> > 提供大量的特性如全文索引、空间函数、压缩、延迟更新等
> >
> > 数据库故障后，安全恢复性
> >
> > 对于只读数据可以忍受故障恢复，MyISAM依然非常适用
> >
> > 日志服务器的场景也比较适用，只需插入和数据读取操作

> 不支持单表一个文件，会将所有的数据和索引内容分别存放在两个文件中
>
> MyISAM对整张表加锁而不是对行，所以不适用写操作比较多的场景
>
> 支持索引缓存不支持数据缓存

文件出处：http://www.linuxidc.com/Linux/2014-04/99721.htm