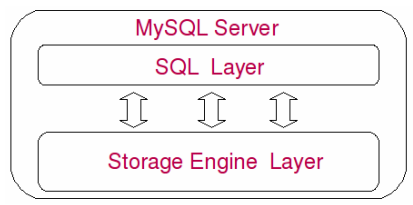
《MySQL性能调优与架构设计》中第二章讲到了MySQL Server 系统架构：

主要是逻辑模块组成和各模块工作的配合。

#########################################

**1 逻辑模块组成**

总的来说，MySQL 可以看成是二层架构，第一层我们通常叫做 SQL Layer，在 MySQL 数据库系统处理底层数据之前的所有工作都是在这一层完成的，包括权限判断，sql 解析，执行计划优化，query cache 的处理等等；第二层就是存储引擎层，我们通常叫做 StorageEngine Layer，也就是底层数据存取操作实现部分，由多种存储引擎共同组成。所以，可以用如下一张最简单的架构示意图来表示 MySQL 的基本架构，如图 2-1 所示：



虽然从上图看起来 MySQL 架构非常的简单，就是简单的两部分而已，但实际上每一层中都含有各自的很多小模块，尤其是第一层 SQL Layer，结构相当复杂的。下面我们就分别针对 SQL Layer 和 Storage Engine Layer 做一个简单的分析。

SQL Layer 中包含了多个子模块，下面我将逐个做一下简单的介绍：

1、初始化模块

顾名思议，初始化模块就是在 MySQL Server 启动的时候，对整个系统做各种各样的初始化操作，比如各种 buffer，cache 结构的初始化和内存空间的申请，各种系统变量的初始化设定，各种存储引擎的初始化设置，等等。

2、核心 API

核心 API 模块主要是为了提供一些需要非常高效的底层操作功能的优化实现，包括各种底层数据结构的实现，特殊算法的实现，字符串处理，数字处理等，小文件 I/O，格式化输出，以及最重要的内存管理部分。核心 API 模块的所有源代码都集中在 mysys 和 strings文件夹下面，有兴趣的读者可以研究研究。

3、网络交互模块

底层网络交互模块抽象出底层网络交互所使用的接口 api，实现底层网络数据的接收与发送，以方便其他各个模块调用，以及对这一部分的维护。所有源码都在 vio 文件夹下面。

4、Client & Server 交互协议模块

任何 C/S 结构的软件系统，都肯定会有自己独有的信息交互协议， MySQL 也不例外。 MySQL的 Client & Server 交互协议模块部分，实现了客户端与 MySQL 交互过程中的所有协议。当然这些协议都是建立在现有的 OS 和网络协议之上的，如 TCP/IP 以及 Unix Socket。

5、用户模块

用户模块所实现的功能，主要包括用户的登录连接权限控制和用户的授权管理。他就像MySQL 的大门守卫一样，决定是否给来访者“开门”。

6、访问控制模块

造访客人进门了就可以想干嘛就干嘛么？为了安全考虑，肯定不能如此随意。这时候就需要访问控制模块实时监控客人的每一个动作，给不同的客人以不同的权限。访问控制模块实现的功能就是根据用户模块中各用户的授权信息，以及数据库自身特有的各种约束，来控制用户对数据的访问。用户模块和访问控制模块两者结合起来，组成了 MySQL 整个数据库系统的权限安全管理的功能。

7、连接管理、连接线程和线程管理

连接管理模块负责监听对 MySQL Server 的各种请求，接收连接请求，转发所有连接请求到线程管理模块。每一个连接上 MySQL Server 的客户端请求都会被分配（或创建）一个连接线程为其单独服务。而连接线程的主要工作就是负责 MySQL Server 与客户端的通信，接受客户端的命令请求，传递 Server 端的结果信息等。线程管理模块则负责管理维护这些连接线程。包括线程的创建，线程的 cache 等。

8、Query 解析和转发模块

在 MySQL 中我们习惯将所有 Client 端发送给 Server 端的命令都称为 query，在 MySQLServer 里面，连接线程接收到客户端的一个 Query 后，会直接将该 query 传递给专门负责将各种 Query 进行分类然后转发给各个对应的处理模块，这个模块就是 query 解析和转发模块。其主要工作就是将 query 语句进行语义和语法的分析，然后按照不同的操作类型进行分类，然后做出针对性的转发。

9、Query Cache 模块

Query Cache 模块在 MySQL 中是一个非常重要的模块，他的主要功能是将客户端提交给MySQL 的 Select 类 query 请求的返回结果集 cache 到内存中，与该 query 的一个 hash 值做一个对应。该 Query 所取数据的基表发生任何数据的变化之后， MySQL 会自动使该 query 的Cache 失效。在读写比例非常高的应用系统中，Query Cache 对性能的提高是非常显著的。当然它对内存的消耗也是非常大的。

10、Query 优化器模块

Query 优化器，顾名思义，就是优化客户端请求的 query，根据客户端请求的 query 语句，和数据库中的一些统计信息，在一系列算法的基础上进行分析，得出一个最优的策略，告诉后面的程序如何取得这个 query 语句的结果。

11、表变更管理模块

表变更管理模块主要是负责完成一些 DML 和 DDL 的 query，如：update，delte，insert，

create table，alter table 等语句的处理。

12、表维护模块

表的状态检查，错误修复，以及优化和分析等工作都是表维护模块需要做的事情。

13、系统状态管理模块

系统状态管理模块负责在客户端请求系统状态的时候，将各种状态数据返回给用户， 像DBA 常用的各种 show status 命令， show variables 命令等，所得到的结果都是由这个模块返回的。

14、表管理器

这个模块从名字上看来很容易和上面的表变更和表维护模块相混淆，但是其功能与变更及维护模块却完全不同。大家知道，每一个 MySQL 的表都有一个表的定义文件，也就是\*.frm文件。表管理器的工作主要就是维护这些文件，以及一个 cache，该 cache 中的主要内容是各个表的结构信息。此外它还维护 table 级别的锁管理。

15、日志记录模块

日志记录模块主要负责整个系统级别的逻辑层的日志的记录，包括 error log，binary

log，slow query log 等。

16、复制模块

复制模块又可分为 Master 模块和 Slave 模块两部分， Master 模块主要负责在Replication 环境中读取 Master 端的 binary 日志，以及与 Slave 端的 I/O 线程交互等工作 。

Slave 模块比 Master 模块所要做的事情稍多一些，在系统中主要体现在两个线程上面。一个是负责从 Master 请求和接受 binary 日志，并写入本地 relay log 中的 I/O 线程。另外一个是负责从 relay log 中读取相关日志事件，然后解析成可以在 Slave 端正确执行并得到和Master 端完全相同的结果的命令并再交给 Slave 执行的 SQL 线程。

17、存储引擎接口模块

存储引擎接口模块可以说是 MySQL 数据库中最有特色的一点了。目前各种数据库产品中，基本上只有 MySQL 可以实现其底层数据存储引擎的插件式管理。这个模块实际上只是一个抽象类，但正是因为它成功地将各种数据处理高度抽象化，才成就了今天 MySQL 可插拔存储引擎的特色。

**2.2.2 各模块工作配合**

在了解了 MySQL 的各个模块之后，我们再看看 MySQL 各个模块间是如何相互协同工作的 。

接下来，我们通过启动 MySQL，客户端连接，请求 query，得到返回结果，最后退出，这样一整个过程来进行分析。

当我们执行启动 MySQL 命令之后， MySQL 的初始化模块就从系统配置文件中读取系统参数和命令行参数，并按照参数来初始化整个系统，如申请并分配 buffer，初始化全局变量，以及各种结构等。同时各个存储引擎也被启动，并进行各自的初始化工作。当整个系统初始化结束后，由连接管理模块接手。连接管理模块会启动处理客户端连接请求的监听程序， 包括 tcp/ip 的网络监听，还有 unix 的 socket。这时候，MySQL Server 就基本启动完成，准备好接受客户端请求了。

当连接管理模块监听到客户端的连接请求（借助网络交互模块的相关功能），双方通过Client & Server 交互协议模块所定义的协议 “寒暄”几句之后，连接管理模块就会将连接

请求转发给线程管理模块，去请求一个连接线程。

线程管理模块马上又会将控制交给连接线程模块，告诉连接线程模块：现在我这边有连

接请求过来了，需要建立连接，你赶快处理一下。连接线程模块在接到连接请求后，首先会

检查当前连接线程池中是否有被 cache 的空闲连接线程，如果有，就取出一个和客户端请求

连接上，如果没有空闲的连接线程，则建立一个新的连接线程与客户端请求连接。当然， 连接线程模块并不是在收到连接请求后马上就会取出一个连接线程连和客户端连接，而是首先通过调用用户模块进行授权检查，只有客户端请求通过了授权检查后，他才会将客户端请求和负责请求的连接线程连上。

在 MySQL 中，将客户端请求分为了两种类型：一种是 query，需要调用 Parser 也就是

Query 解析和转发模块的解析才能够执行的请求；一种是 command，不需要调用 Parser 就可以直接执行的请求。如果我们的初始化配置中打开了 Full Query Logging 的功能，那么Query 解析与转发模块会调用日志记录模块将请求计入日志，不管是一个 Query 类型的请求还是一个 command 类型的请求，都会被记录进入日志，所以出于性能考虑，一般很少打开 Full Query Logging 的功能。

当客户端请求和连接线程“互换暗号（互通协议） ”接上头之后，连接线程就开始处理客户端请求发送过来的各种命令（或者 query），接受相关请求。它将收到的 query 语句转给 Query 解析和转发模块， Query 解析器先对 Query 进行基本的语义和语法解析，然后根据命令类型的不同，有些会直接处理，有些会分发给其他模块来处理。

如果是一个 Query 类型的请求，会将控制权交给 Query 解析器。 Query 解析器首先分析看是不是一个 select 类型的 query，如果是，则调用查询缓存模块，让它检查该 query 在query cache 中是否已经存在。如果有，则直接将 cache 中的数据返回给连接线程模块，然后通过与客户端的连接的线程将数据传输给客户端。如果不是一个可以被 cache 的 query类型，或者 cache 中没有该 query 的数据，那么 query 将被继续传回 query 解析器，让 query解析器进行相应处理，再通过 query 分发器分发给相关处理模块。

如果解析器解析结果是一条未被 cache 的 select 语句，则将控制权交给 Optimizer，也就是 Query 优化器模块，如果是 DML 或者是 DDL 语句，则会交给表变更管理模块，如果是一些更新统计信息、检测、修复和整理类的 query 则会交给表维护模块去处理，复制相关的query 则转交给复制模块去进行相应的处理，请求状态的 query 则转交给了状态收集报告模块。实际上表变更管理模块根据所对应的处理请求的不同，是分别由 insert 处理器、 delete处理器、update 处理器、create 处理器，以及 alter 处理器这些小模块来负责不同的 DML和 DDL 的。

在各个模块收到 Query 解析与分发模块分发过来的请求后，首先会通过访问控制模块检查连接用户是否有访问目标表以及目标字段的权限，如果有，就会调用表管理模块请求相应的表，并获取对应的锁。表管理模块首先会查看该表是否已经存在于 table cache 中，如果已经打开则直接进行锁相关的处理，如果没有在 cache 中，则需要再打开表文件获取锁， 然后将打开的表交给表变更管理模块。

当表变更管理模块“获取”打开的表之后，就会根据该表的相关 meta 信息，判断表的

存储引擎类型和其他相关信息。根据表的存储引擎类型，提交请求给存储引擎接口模块， 调

用对应的存储引擎实现模块，进行相应处理。

不过，对于表变更管理模块来说，可见的仅是存储引擎接口模块所提供的一系列 “标准”接口，底层存储引擎实现模块的具体实现，对于表变更管理模块来说是透明的。他只需要调用对应的接口，并指明表类型，接口模块会根据表类型调用正确的存储引擎来进行相应的处理。

当一条 query 或者一个 command 处理完成（成功或者失败）之后，控制权都会交还给连接线程模块。如果处理成功，则将处理结果（可能是一个 Result set，也可能是成功或者失败的标识）通过连接线程反馈给客户端。如果处理过程中发生错误，也会将相应的错误信息发送给客户端，然后连接线程模块会进行相应的清理工作，并继续等待后面的请求，重复上面提到的过程，或者完成客户端断开连接的请求。

如果在上面的过程中，相关模块使数据库中的数据发生了变化，而且 MySQL 打开了 binlog 功能，则对应的处理模块还会调用日志处理模块将相应的变更语句以更新事件的形式记录到相关参数指定的二进制日志文件中。

在上面各个模块的处理过程中，各自的核心运算处理功能部分都会高度依赖整个 MySQL的核心 API 模块，比如内存管理，文件 I/O，数字和字符串处理等等。

