第八讲：网络模块和派发模块

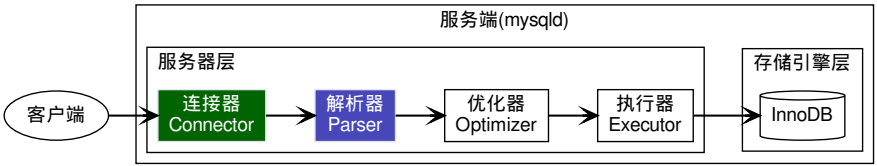
知春路遇上八里桥

<2024-05-30 Thu>

1

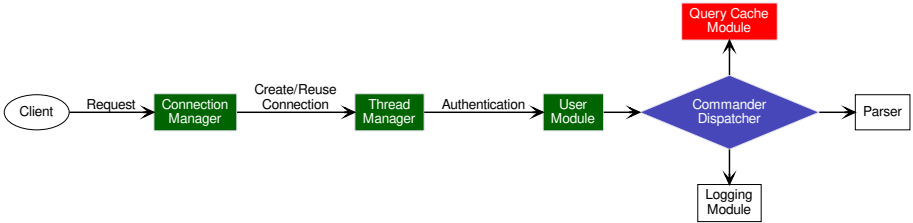
前情提要

执行流程


服务端(mysqld)
服务器层
客户端
连接器
Connector
解析器
Parser
优化器
Optimizer
执行器
Executor
存储引擎层
InnoDB

本节内容

- 连接器
  - ☑ 连接管理器 Connection Manager
  - ☑ 线程管理器 Thread Manager
  - ☑ 用户模块 User Module
- 解析器
  - ☐ 网络模块 Net Module
  - ☐ 派发模块 Commander Dispatcher



2

协议

客户端/服务器的协议

MySQL 的客户端支持以下数据交互协议：

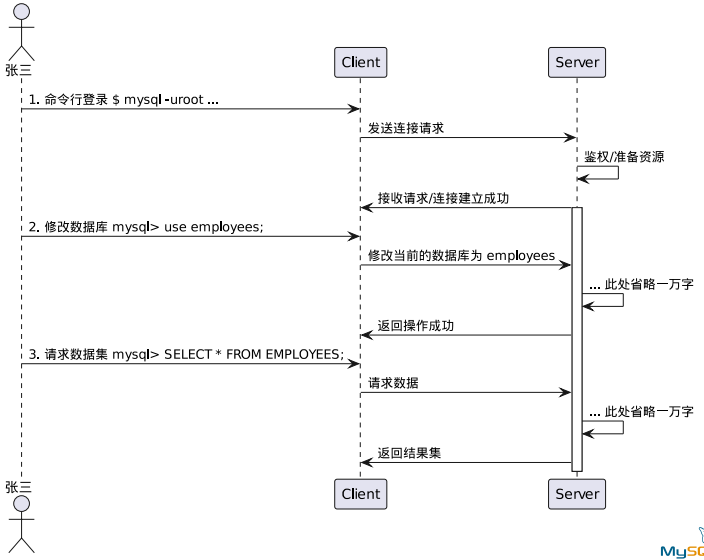
1. Connectors (Connector/C, Connector/J, and so forth)
2. MySQL Proxy
3. Communication between master and slave replication servers

数据交互具体包含以下特性：

1. 使用 SSL 进行透明数据加密（Transparent encryption）
2. 支持透明数据压缩（Transparent compression）
3. 连接阶段 [1] (Connection Phase)，处理兼容性和认证信息的数据交换
   - 在 `check_connection()` 函数中执行握手操作
4. 命令阶段 [2] (Command Phase)，接收客户端命令然后执行
   - 在 `do_command()` 函数中执行用户命令

[1] https://dev.mysql.com/doc/dev/mysql-server/8.0.37/page_protocol_connection_phase.html
[2] https://dev.mysql.com/doc/dev/mysql-server/8.0.37/page_protocol_command_phase.html

什么是命令


张三
Client
Server
1. 命令行登录 $ mysql -uroot ...
发送连接请求
鉴权/准备资源
接收请求/连接建立成功
2. 修改数据库 mysql> use employees;
修改当前的数据库为 employees
... 此处省略一万字
返回操作成功
3. 请求数据集 mysql> SELECT * FROM EMPLOYEES;
请求数据
... 此处省略一万字
返回结果集
张三
Client
Server

Command 枚举类型

- 客户端发送带序列号的包 [3]
  - 序列号 sequence-id [4] ，位于负载 (payload) 的第一个字节
  - 包 packet，负载后续传输的数据
- 所有命令定义见 ⋆ include/my_command.cc ，是 `enum enum_server_command` 枚举类型
- Text Protocol 文本协议
  - `COM_QUERY`
- Utility Commands 管理命令
  - `COM_QUIT COM_INIT_DB COM_FIELD_LIST COM_REFRESH COM_STATISTICS`
  - `COM_PROCESS_INFO COM_PROCESS_KILL COM_DEBUG COM_PING`
  - `COM_CHANGE_USER COM_RESET_CONNECTION COM_SET_OPTION`
- Prepared Statements 准备语句
  - `COM_STMT_PREPARE COM_STMT_EXECUTE COM_STMT_FETCH COM_STMT_CLOSE`
  - `COM_STMT_RESET COM_STMT_SEND_LONG_DATA`
- Stored Programs [5]
  - Multi-Resultset / Multi-Statement

[3] https://dev.mysql.com/doc/dev/mysql-server/8.0.37/page_protocol_command_phase.html
[4] https://dev.mysql.com/doc/dev/mysql-server/8.0.37/page_protocol_basic_packets.html
[5] https://dev.mysql.com/doc/dev/mysql-server/8.0.37/page_protocol_command_phase_sp.html

COM_QUERY 命令类型的负载示例 [6]

Payload

Type	Name	Description
int<1>	command	0x03: COM_QUERY
if **CLIENT_QUERY_ATTRIBUTES** is set {		
int<lenenc>	parameter_count	Number of parameters
int<lenenc>	parameter_set_count	Number of parameter sets. Currently always 1
if parameter_count > 0 {		
binary<var>	null_bitmap	NULL bitmap, length= (num_params + 7) / 8
int<1>	new_params_bind_flag	Always 1. Malformed packet error if not 1
if new_params_bind_flag, for each parameter {		
int<2>	param_type_and_flag	Parameter type (2 bytes). The MSB is reserved for unsigned flag
string<lenenc>	parameter name	String
}		
binary<var>	parameter_values	value of each parameter: **Binary Protocol Value**
}		
}		
string<EOF>	query	the text of the SQL query to execute

[6] https://dev.mysql.com/doc/dev/mysql-server/8.0.37/page_protocol_com_query.html

COM_DATA 联合体

- COM_DATA 定义见 ⋆ include/mysql/com_data.h

```
112 union COM_DATA {
113   COM_INIT_DB_DATA com_init_db;
114   COM_REFRESH_DATA com_refresh;
115   COM_KILL_DATA com_kill;
116   COM_SET_OPTION_DATA com_set_option;
117   COM_STMT_EXECUTE_DATA com_stmt_execute;
118   COM_STMT_FETCH_DATA com_stmt_fetch;
119   COM_STMT_SEND_LONG_DATA_DATA com_stmt_send_long_data;
120   COM_STMT_PREPARE_DATA com_stmt_prepare;
121   COM_STMT_CLOSE_DATA com_stmt_close;
122   COM_STMT_RESET_DATA com_stmt_reset;
123   COM_QUERY_DATA com_query;
124   COM_FIELD_LIST_DATA com_field_list;
125 };
```

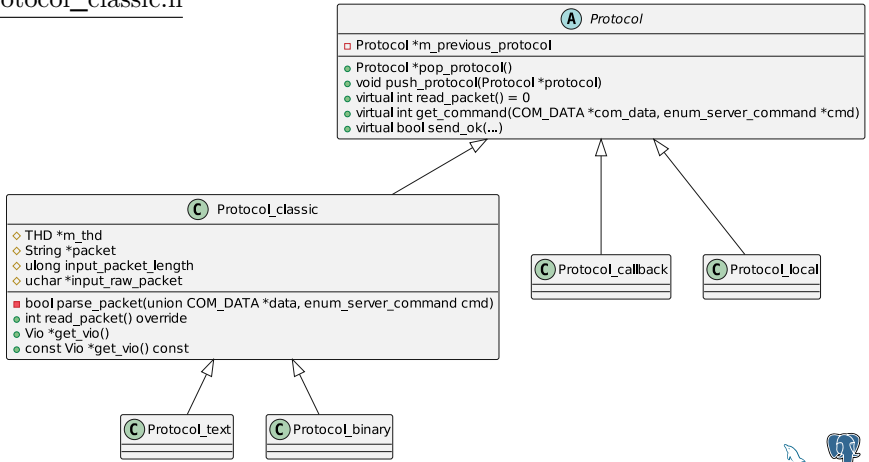
- 它表示不同 Command 所需要关心的数据结构

3

网络模块

Protocol 协议类

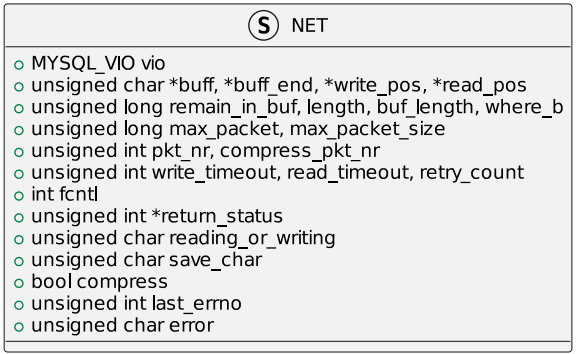
- 抽象类 `Protocol` 实现见文件 ⋆ sql/protocol.h ， 它实现了 MySQL 交互协议 [7]
- 实现类 `Protocol_classic` / `Protocol_text` / `Protocol_binary` 见文件 ⋆ sql/protocol_classic.h



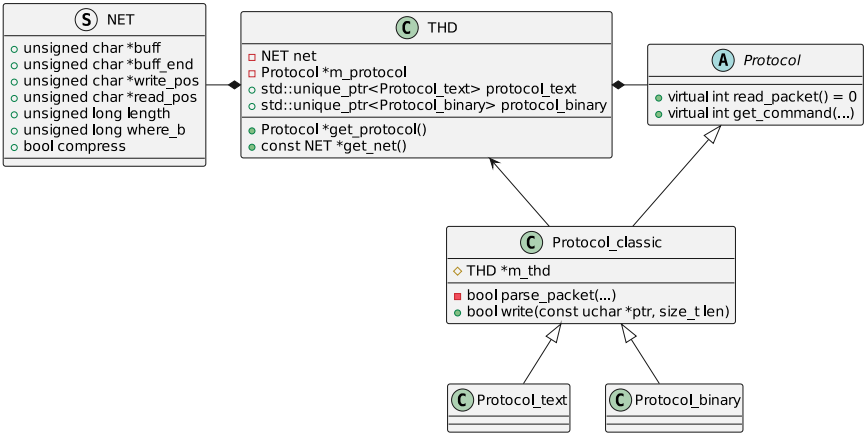
[7] https://dev.mysql.com/doc/dev/mysql-server/8.0.37/page_protocol_basics.html

NET 网络结构体

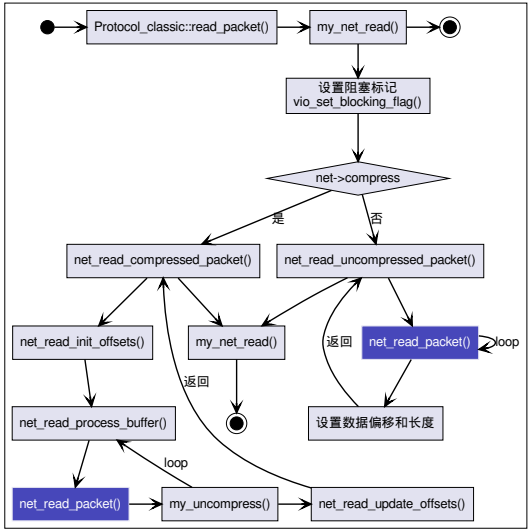
- `vio` 虚拟 IO 层的实现
- 一些缓冲的 buffer 定义
- 当前状态的指针：长度，Buffer 总长度，Buffer 所在位置
- 一些超时记录参数，读超时，写超时，重试超时
- compress 记录是否压缩

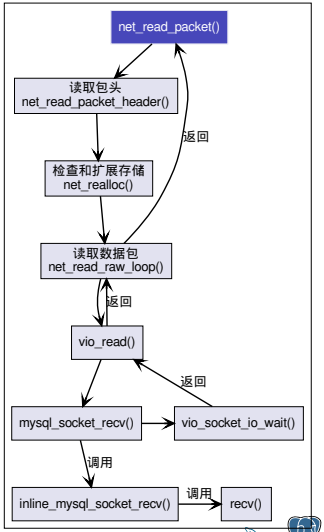


与 THD 之间的关系

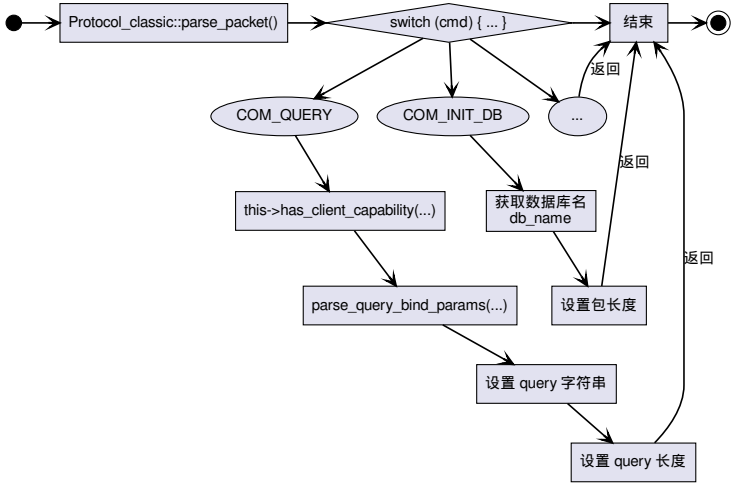

NET
unsigned char *buff
unsigned char *buff_end
unsigned char *write_pos
unsigned char *read_pos
unsigned long length
unsigned long where_b
bool compress
THD
NET net
Protocol *m_protocol
std::unique_ptr<Protocol_text> protocol_text
std::unique_ptr<Protocol_binary> protocol_binary
Protocol *get_protocol()
const NET *get_net()
Protocol
virtual int read_packet() = 0
virtual int get_command(...)
Protocol_classic
THD *m_thd
bool parse_packet(...)
bool write(const uchar *ptr, size_t len)
Protocol_text
Protocol_binary

读取包 read_packet 函数





解析包 parse_packet 函数



4

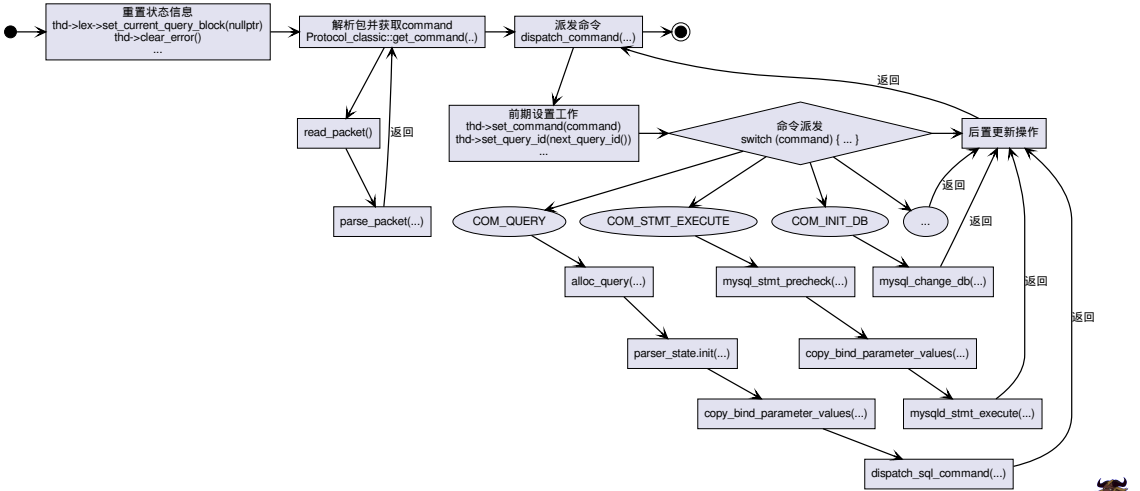
派发模块

do_command 函数

- do_command() 的实际功能
  - 读取一个命令，注意命令可能是 query 或者指令
    1. Query 例如：SELECT * FROM employees;
    2. 指令例如：use dbname / shutdown
  - 然后执行命令，并发送结果
- do_command() 的实现见 ⋆ sql/sql_parse.cc

```
1308 bool do_command(THD *thd) {
1309   bool return_value;
1310   int rc;
  ⋮
1444 out:
1445   /* The statement instrumentation must be closed in all cases. */
1446   assert(thd->m_digest == nullptr);
1447   assert(thd->m_statement_psi == nullptr);
1448   return return_value;
1449 }
```

do_command 执行流程



结束