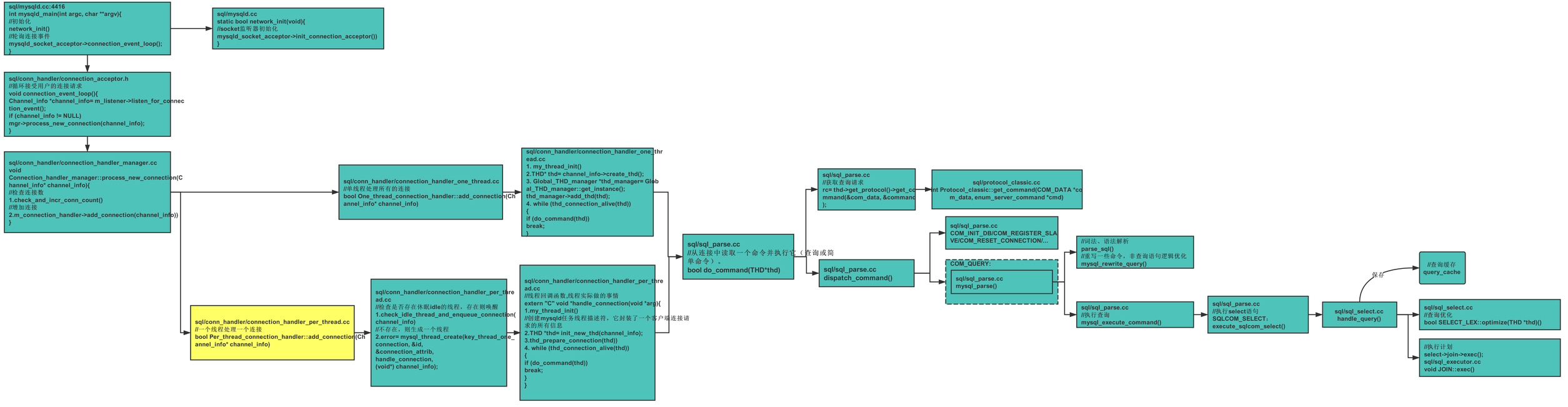
sql/mysqld.cc:4416
int mysqld_main(int argc, char **argv){
//初始化
network_init()
//轮询连接事件
mysqld_socket_acceptor->connection_event_loop();
}
sql/mysqld.cc
static bool network_init(void){
//socket监听器初始化
mysqld_socket_acceptor->init_connection_acceptor())
}
sql/conn_handler/connection_acceptor.h
//循环接受用户的连接请求
void connection_event_loop(){
Channel_info *channel_info= m_listener->listen_for_connection_event();
if (channel_info != NULL)
mgr->process_new_connection(channel_info);
}
sql/conn_handler/connection_handler_manager.cc
void
Connection_handler_manager::process_new_connection(Channel_info* channel_info){
//检查连接数
1.check_and_incr_conn_count()
//增加连接
2.m_connection_handler->add_connection(channel_info))
}
sql/conn_handler/connection_handler_one_thread.cc
//单线程处理所有的连接
bool One_thread_connection_handler::add_connection(Channel_info* channel_info)
sql/conn_handler/connection_handler_one_thread.cc
1. my_thread_init()
2.THD* thd= channel_info->create_thd();
3. Global_THD_manager *thd_manager= Global_THD_manager::get_instance();
thd_manager->add_thd(thd);
4. while (thd_connection_alive(thd))
{
if (do_command(thd))
break;
}
sql/conn_handler/connection_handler_per_thread.cc
//一个线程处理一个连接
bool Per_thread_connection_handler::add_connection(Channel_info* channel_info)
sql/conn_handler/connection_handler_per_thread.cc
//检查是否存在休眠idle的线程，存在则唤醒
1.check_idle_thread_and_enqueue_connection(channel_info)
//不存在，则生成一个线程
2.error= mysql_thread_create(key_thread_one_connection, &id,
&connection_attrib,
handle_connection,
(void*) channel_info);
sql/conn_handler/connection_handler_per_thread.cc
//线程回调函数,线程实际做的事情
extern "C" void *handle_connection(void *arg){
1.my_thread_init()
//创建mysqld任务线程描述符，它封装了一个客户端连接请求的所有信息
2.THD *thd= init_new_thd(channel_info);
3.thd_prepare_connection(thd))
4. while (thd_connection_alive(thd))
{
if (do_command(thd))
break;
}
}
sql/sql_parse.cc
//从连接中读取一个命令并执行它（查询或简单命令）。
bool do_command(THD*thd)
sql/sql_parse.cc
//获取查询请求
rc= thd->get_protocol()->get_command(&com_data, &command);
sql/protocol_classic.cc
int Protocol_classic::get_command(COM_DATA *com_data, enum_server_command *cmd)
sql/sql_parse.cc
dispatch_command()
sql/sql_parse.cc
COM_INIT_DB/COM_REGISTER_SLAVE/COM_RESET_CONNECTION/...
COM_QUERY:
sql/sql_parse.cc
mysql_parse()
//词法、语法解析
parse_sql()
//重写一些命令，非查询语句逻辑优化
mysql_rewrite_query()
sql/sql_parse.cc
//执行查询
mysql_execute_command()
sql/sql_parse.cc
//执行select语句
SQLCOM_SELECT：
execute_sqlcom_select()
sql/sql_select.cc
handle_query()
保存
//查询缓存
query_cache
sql/sql_select.cc
//查询优化
bool SELECT_LEX::optimize(THD *thd)()
//执行计划
select->join->exec();
sql/sql_executor.cc
void JOIN::exec()