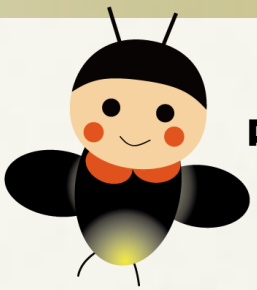


中华数据库
行业协会

2014中华数据库与运维安全大会

官方网址：www.zhdba.com

深入MySQL源码 – Step By Step

网易杭研——何登成

自我简介

- 何登成
  - 网易——杭州研究院；
- **工作领域**
  - 数据库引擎/分布式数据库/分布式KV
- **技术领域**
  - 数据库/分布式/并发编程/性能优化
- **联系方式**
  - 微博：何_登成
  - 博客：何登成的技术博客

- 开始，我想把MySQL中我所了解的源码文件，一个个拿出来分析。但是转头一想，这不太切实际！

- 因此，此分享的内容改为：总结我个人在过去几年内学习MySQL，开发自己的引擎的一些经验，走过的一些路，希望对大家有所帮助；

Outline

- Step 0
  - 心理准备；
  - 知识准备；
  - 工具准备；
- Step 1
  - 亲自动手，编译一个MySQL；
  - 阅读MySQL Internal文档；
  - 亲自验证文档中所有的知识；
  - 掌握MySQL基本架构；
- Step 2
  - 亲自解释一个关于MySQL的疑惑；
  - 理清一个MySQL功能的实现细节；
  - 好记性不如烂笔头；
  - 实现一个简单的Patch、修复一个小Bug；
  - 构建属于自己的知识体系；
- Step 3
  - 注重MySQL技术圈子的收集；
  - 跟踪MySQL每个发行版和Bugs；
  - 定期更新自己的前期知识，自我纠错；
  - 注重发散知识的积累，挑战更大的难度；
  - Keep on going；
- 个人推荐的相关书籍

Step 0 —— 心理准备

* 相对于其他的前期准备，心理准备是最重要的，你心里已经准备好去挑战MySQL这个百万行的开源系统了吗？

* 不妨问问自己以下的问题
  * 做这件事，对我有什么意义？对公司有什么意义？
  * 我能够花多少时间和精力？
  * 我有没有**兴趣**一直坚持下去？我碰到困难是不是经常退缩？
  * 我是不是一个有着强烈**好奇心**的人？

* 一个忠告
  * 千万不要想一口气吃成一个胖子！
  * MySQL，不是一个短期能够吃透的系统！

Step 0 —— 知识准备

- C/C++编程经验
  - 相信大部分人都会或多或少有这方面的经验；
  - InnoDB引擎：　C
  - MySQL Server：　C++
- 数据库理论知识
  - 什么是RDBMS？ACID？Transaction？Index？Log？...
  - 相对于C/C++编程经验，有数据库理论知识方面的储备的人，相信会少一些；
  - 但，RDBMS是一个成熟的系统，有各种经典书籍/大学课程，想学不难；
- 英文阅读
  - 不仅仅是MySQL，所有做技术的，都应该有意识的提高自己的英文阅读理解能力；

Step 0 —— 知识准备(续)

- 就我个人而言，接触MySQL数据库之前
  - 7年数据库内核研发经验
    - 研究生期间，实验室与神舟通用合作，进行国产Oscar数据库的研发。我有幸参与其中，做过事务/锁/索引/并发控制等模块的研发，积累了大量经验；
  - 两年数据库运维
    - 第一份工作，阿里B2B Oracle DBA，跟随前辈们学习了大量Oracle数据库的先进理念；

Step 0 —— 工具准备

- 工欲善其事，必先利其器
  - 做任何事，都需要有专用称手的工具
  - 挑选最适合自己的操作系统/编译工具/开发工具/源码阅读工具/...
  - 我个人喜欢的工具
    - 操作系统：Windows
    - 开发工具：Windows Visual Studio
    - 源码阅读：Source Insight
  - 我个人常用的工具
    - 操作系统：Linux
    - 开发工具：vi　gcc/g++/gdb
    - 源码阅读：vi

Step 1 —— 编译你的MySQL

- 想学MySQL源码？
  - 想。
  - 编译过没？
  - 没有...
  - 为什么不编译一个？
  - 太难...
- 学习MySQL源码的第一步
  - 你必须亲自动手，编译出一个属于你自己的MySQL可运行版本；
  - 不知道怎么编译？网上有太多的相关资料...
  - 如何编译MySQL？Linux下；Windows下；
- 运行自己编译出的版本
  - 简单的建表，I/U/D/S ...

Step 1 —— 阅读Internal文档

* 在Google上搜索MySQL Internal，第一篇的链接就是：MySQL Internals Manual；搜索InnoDB Internal，第一篇则是：InnoDB Architecture and Internals；
  * 这两个都是必须看的；
  * 如果你用了baidu，那当我没说...
* 此外
  * MySQL Reference Manual 一定要看!!!

MySQL Internals Manual

Abstract

This is the MySQL Internals Manual.

Document generated on: 2014-04-29 (revision: 215)

For legal information, see the Legal Notice.

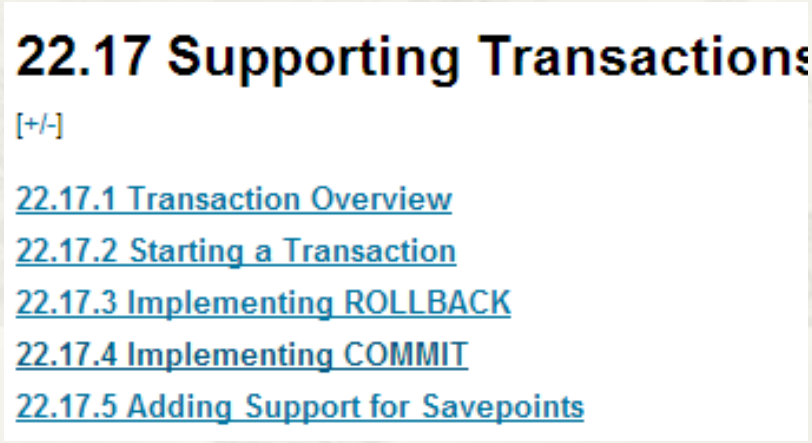
Table of Contents [+/-]

Step 1 —— 阅读Internal文档(续)

* 个人经验
  * 由于工作关系(我做的是自主研发的存储引擎)，我在阅读MySQL Internals Manual的过程中，重点关注其Chapter 22 Writing a Custom Storage Engine；
  * 当然，根据每个人工作重心的不同，关注的点也会有所不同，例如：
    * 对MySQL Binlog & Replication关注的，可以看Chapter 18 & 19；
    * 对MySQL协议关注的，可以看Chapter 14；

Step 1 —— 验证每一个知识点

- 古人云：纸上得来终觉浅，绝知此事要躬行；
- 我认为：不要相信别人说的！别人的再好，都是别人的，看过之后，一定要自己亲自验证，验证后才是你自己的；
- 个人经验
  - 书中，博客中获取的每一个知识点，都亲自进行验证；
  - 源码阅读 + 跟踪调试
  - 例如：MySQL Internals Manual，Chapter 22.17，内容右图所示；



Step 1 —— 验证每一个知识点(续)

Ha_innodb.cc

commit

- commit_cond
- commit_cond_key
- commit_cond_m
- commit_cond_mutex_key
- commit_threads
- commit_threads_m
- commit_threads_m_key
- innobase_commit
- innobase_commit
- innobase_commit_concurrency
- innobase_commit_low
- innobase_commit_low
- prepare_commit_mutex
- innobase_commit_by_xid
- innobase_commit_by_xid
- innobase_commit_concurrency_val
- prepare_commit_mutex_key
- innobase_commit_concurrency_ini
- innobase_commit_concurrency_ini
- trx_has_prepare_commit_mutex
- trx_owns_prepare_commit_mutex_s

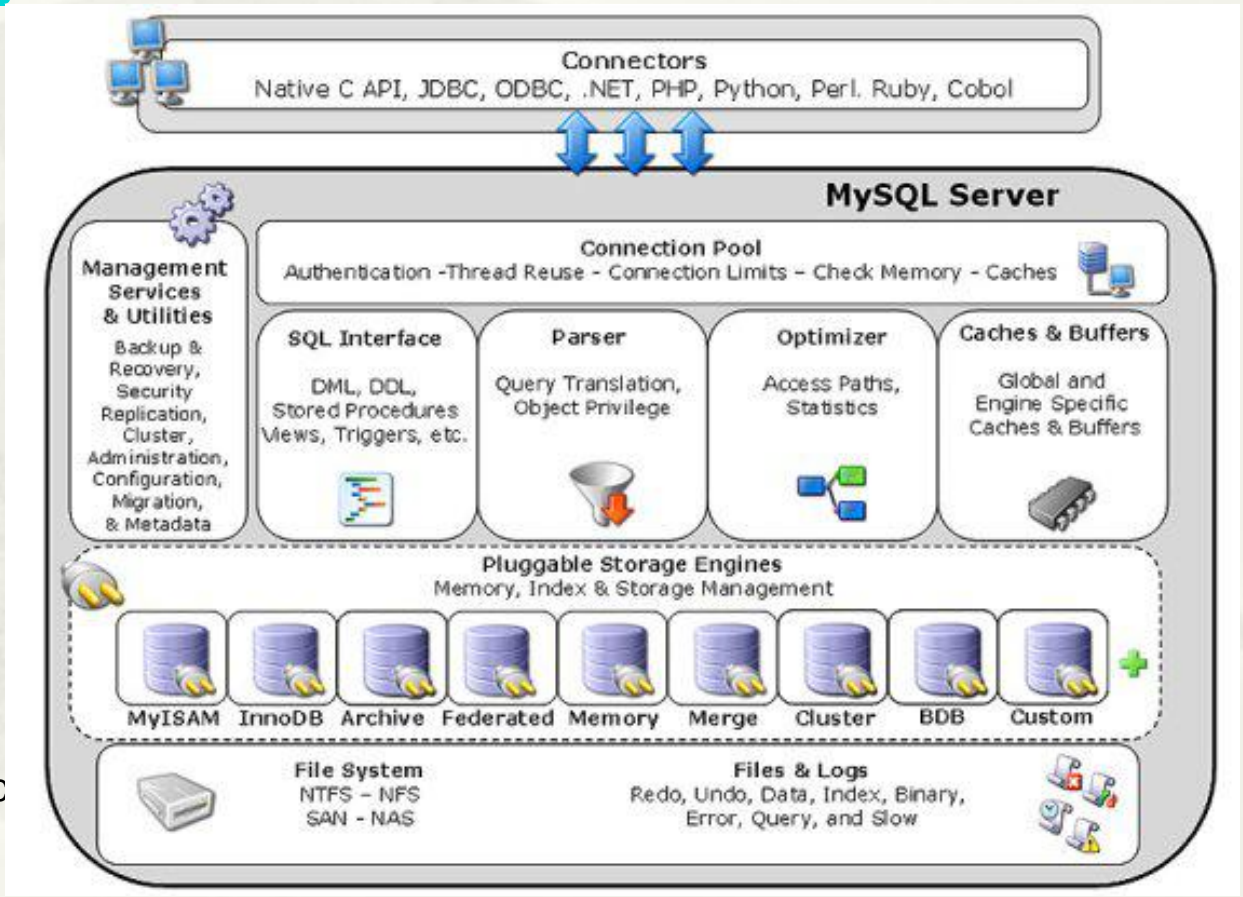
```
02680:
02681: /*****************************************************************//**
02682: Commits a transaction in an InnoDB database or marks an SQL statement
02683: ended.
02684: @return  0 */
02685: static
02686: int
02687: innobase_commit(
02688: /*============*/
02689:         handlerton *hton,  /*! < in: Innodb handlerton */
02690:     THD*     thd,      /*! < in: MySQL thread handle of the user for whom
02691:                        the transaction should be committed */
02692:     bool     all)      /*! < in:  TRUE - commit transaction
02693:                            FALSE - the current SQL statement ended */
02694: {
02695:     trx_t*       trx;
02696:
02697:     DBUG_ENTER("innobase_commit");
02698:     DBUG_ASSERT(hton == innodb_hton_ptr);
02699:     DBUG_PRINT("trans", ("ending transaction"));
02700:
02701:     trx = check_trx_exists(thd);
02702:
02703:     /* Since we will reserve the kernel mutex, we have to release
02704:     the search system latch first to obey the latching order. */
02705:
02706:     if (trx->has_search_latch) {
02707:         trx_search_latch_release_if_reserved(trx);
02708:     }
02709:
02710:     /* Transaction is deregistered only in a commit or a rollback. If
02711:     it is deregistered we know there cannot be resources to be freed
02712:     and we could return immediately.  For the time being, we play safe
02713:     and do the cleanup though there should be nothing to clean up. */
02714:
02715:     if (!trx_is_registered_for_2pc(trx) && trx_is_started(trx)) {
02716:
02717:         sql_print_error("Transaction not registered for MySQL 2PC, "
02718:                 "but transaction is active");
02719:     }
02720:
02721:     if (all
02722:         || (!thd_test_options(thd, OPTION_NOT_AUTOCOMMIT | OPTION_BEGIN))) {
02723:
02724:         /* We were instructed to commit the whole transaction, or
02725:         this is an SQL statement end and autocommit is on */
02726:
02727:         /* We need current binlog position for ibbackup to work.
02728:         Note, the position is current because of
02729:         prepare_commit_mutex */
```

Step 1 ——掌握MySQL基本架构

* 插件式存储引擎架构

* MySQL Server
  * Parser
  * Optimizer
  * SQL Interfaces
  * ...

* Storage Interface
  * InnoDB: ha_innodb.c
  * MyIsam: ha_myisam.c



Step 2 ——亲自解释一个关于MySQL的疑惑

- 例如：一个困扰很多DBA的参数：innodb_flush_log_at_trx_commit；
  有三个取值：0, 1, 2，默认为1；
  但这个参数究竟该怎么理解？？
  - 通过前面的阅读，你已经知道事务的提交(innodb)，通过ha_innodb.cc的innobase_commit()函数来处理，那跟踪此函数一定能够找到该参数的使用；
  - ha_innodb.cc::innobase_commit()->innobase_commit_low()->TrxOtrx.c::trx_commit_for_mysql()->trx_commit_off_kernel();
  - trx_commit_off_kernel()函数中，看到了一个变量：srv_flush_log_at_trx_commit，看其如何处理？

```
if (trx->flush_log_later) {
    /* Do nothing yet */
    trx->must_flush_log_later = TRUE;
} else if (srv_flush_log_at_trx_commit == 0) {
    /* Do nothing */
} else if (srv_flush_log_at_trx_commit == 1) {
    if (srv_unix_file_flush_method == SRV_UNIX_NOSYNC) {
        /* Write the log but do not flush it to disk */

        log_write_up_to(lsn, LOG_WAIT_ONE_GROUP,
                FALSE);
    } else {
        /* Write the log to the log files AND flush
        them to disk */

        log_write_up_to(lsn, LOG_WAIT_ONE_GROUP, TRUE);
    }
} else if (srv_flush_log_at_trx_commit == 2) {

    /* Write the log but do not flush it to disk */

    log_write_up_to(lsn, LOG_WAIT_ONE_GROUP, FALSE);
} else {
    ut_error;
}

trx->commit_lsn = lsn;
```

Step 2 ——亲自解释一个关于MySQL的疑惑(续)

- 参数 = 0
  - 提交时，对日志不做处理；
- 参数 = 1
  - 调用log_write_up_to函数，第三个参数为TRUE；
- 参数 = 2
  - 同1，但是第三个参数为FALSE；
- log_write_up_to第三个参数分析
  - flush_to_disk
  - 控制日志是否刷出到磁盘：fsync
- 总结
  - 0 最快；1最慢；
  - 1能够保证事务提交后数据不丢失；
  - 1的性能好坏，取决于磁盘处理fsync调用的能力；
  - Done！

```
/*****************************************************//**
This function is called, e.g., when a transaction wants to commit. It checks
that the log has been written to the log file up to the last log entry written
by the transaction. If there is a flush running, it waits and checks if the
flush flushed enough. If not, starts a new flush. */
UNIV_INTERN
void
log_write_up_to(
/*============*/
    ib_uint64_t lsn,      /*! < in: log sequence number up to which
                          the log should be written,
                          IB_ULONGLONG_MAX if not specified */
    ulint       wait,     /*! < in: LOG_NO_WAIT, LOG_WAIT_ONE_GROUP,
                          or LOG_WAIT_ALL_GROUPS */
    ibool       flush_to_disk)
                          /*! < in: TRUE if we want the written log
                          also to be flushed to disk */
```

Step 2 ——理清一个MySQL功能的实现细节

- 前面提到的，解释一个参数的疑惑，只是小Case，更大的挑战，就是理清一个MySQL功能模块的实现细节；
- MySQL有哪些功能模块？
  - Parser & Optimizer
  - Binlog & Replication
  - Handler Interface
  - InnoDB
    - Transaction & Lock
    - MVCC
    - Log & Crash Recovery
    - Buffer Pool
    - I/O
    - ...
  - ...
- 牢记一点：简单先行；到目前为止，我也不能说理解了MySQL Optimizer的实现☹

Step 2 —— 好记性不如烂笔头

- 当你在解释一个疑问点，或是分析一个功能实现的时候；
- 你应该把你的收获记录下来；
- 千万不要相信你的记忆力，再牛逼的记忆力也抵挡不住时间的流逝；
- 笔记可以做的很粗糙，可以做的只有你一个人能看得懂，但是笔记一定要有；
  - 当然，后续你可以把这些笔记好好加以整理，分享给大家，甚至是出书均可；

Step 2 ——好记性不如烂笔头(续)

* 本人的一些技术方面的分享集合

MySQL/InnoDB

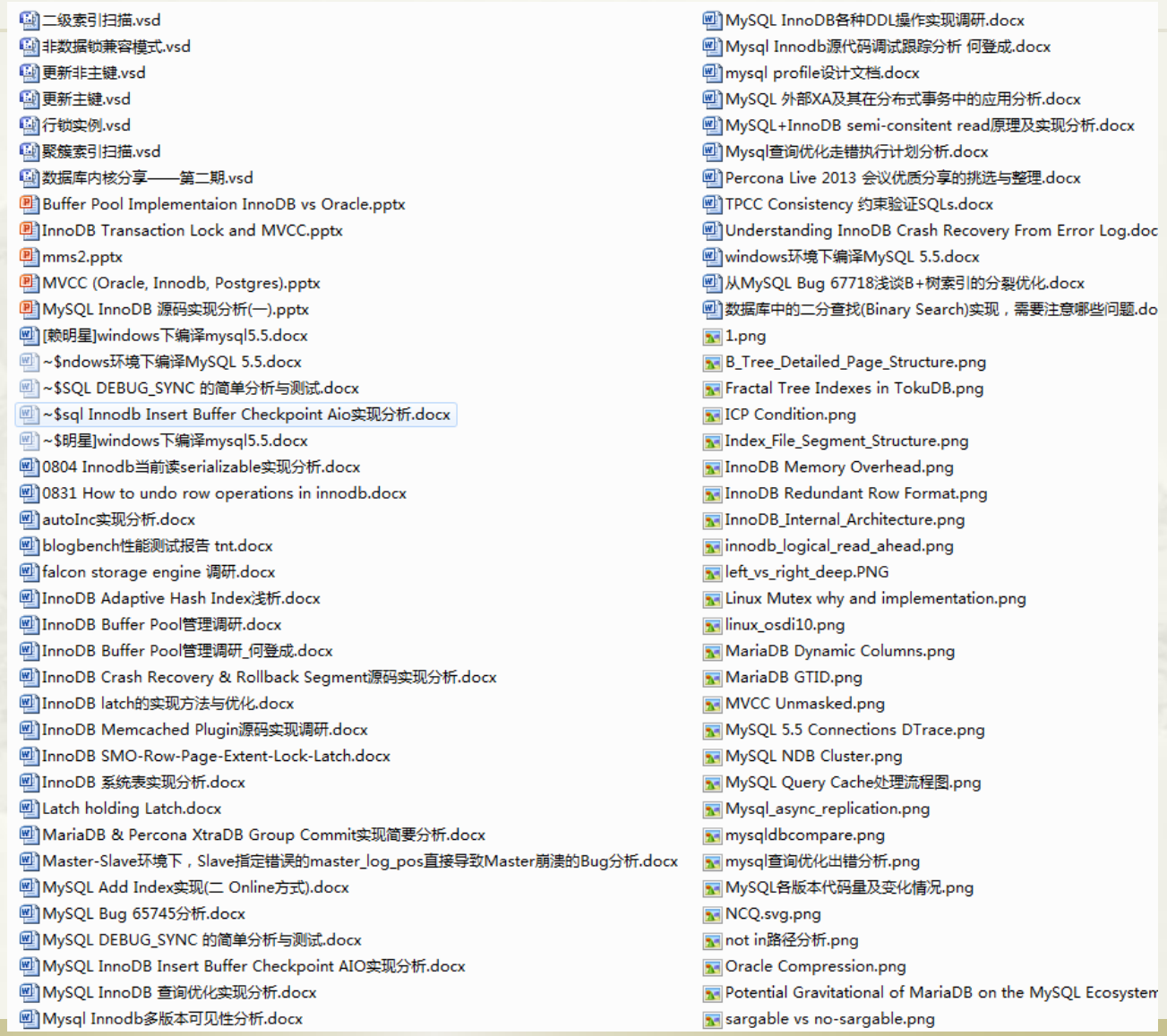
- **MySQL InnoDB 源码实现分析(一):** 微盘地址 Slideshare地址
- **MVCC (Oracle, Innodb, Postgres):** 微盘地址 Slideshare地址
- **InnoDB Transaction Lock and MVCC:** 微盘地址 Slideshare地址
- **Buffer Pool Implementaion InnoDB vs Oracle:** 微盘地址 视频地址 Slideshare地址
- **MySQL查询优化浅析:** 微盘地址 Slideshare地址
- **InnoDB 日志 回滚段 & 崩溃恢复 实现详解:** 微盘地址 视频地址 Slideshare地址
- **InnoDB Page Structure(InnoDB索引页面结构详解):** 微盘地址
- **InnoDB Log Block Structure(InnoDB日志块结构详解):** 微盘地址
- **MySQL Row Format(MySQL行格式详解):** 微盘地址

Oracle

- **Oracle RAC资源管理算法与Cache-Fusion实现浅析:** 微盘地址 Slideshare地址

Step 2 —— 好记性不如烂笔头(续)

* 本人未完全公开的一些笔记
  * 部分如右图所示；



Step 2 ——实现一个简单的Patch

- 到这个阶段，你已经成功掌握了MySQL的一个功能实现细节。开始跃跃欲试，此时我的建议，是尝试为MySQL写一个简单的/有用的Patch，练练手；
- MySQL分支——Percona，在其创始初期，所做的最大贡献，就是将MySQL中众多通过代码写死的参数，通过用户可配置的形式展示出来，典型例子：
  - 早期MySQL InnoDB中，将磁盘的IOPS能力，写死在代码中：200 IOPS；
  - 哪怕你为MySQL配置了很好的磁盘阵列/SSD，MySQL也无法充分发挥此能力；
  - Percona的做法，将它设置为用户可配置参数，极小的改动，收获极大的效益；

```
static MYSQL_SYSVAR_ULONG(io_capacity, srv_io_capacity,
  PLUGIN_VAR_RQCMDARG,
  "Number of IOPs the server can do. Tunes the background IO rate",
  NULL, NULL, 200, 100, ~0L, 0);
```

Step 2 ——修复一个简单的Bug

- 虽然MySQL已经在各大公司，成功使用了10年以上；但说真的，MySQL目前仍旧不是一个完美无Bug的系统；
- MySQL在运行，压力测试，各种功能测试的测试中，虽然概率不大，但仍旧可能会出现Core Dump，陷入死循环等Bugs；
- 如果不幸，碰到了这样的Bug，你该怎么办？
  - 首选，去MySQL的官方Bugs库查询此Bug，看在哪个版本被修复；
  - Bugs库未找到，新的Bug？提交Bug，等待MySQL官方响应；
  - 或者将Bug提交给响应更快的MariaDB，Percona；
  - 或者，你已经有了MySQL源码经验，你可以亲自尝试定位并修复此Bug！

Step 2 —— 修复一个简单的Bug(续)

- 个人经验
  - 一个月前，团队做MySQL的Master-Slave测试时，shutdown MySQL Master竟然无法正常关闭；
  - gdb挂载master mysqld，发现系统处于ret= mysql_bin_log.wait_for_update_bin_log(thd, heartbeat_ts)函数内，一直处于等待状态；
  - 观察发现， thd->killed 标识位已经被设置，按理说根本就不应该进入等待函数；
  - 再进一步分析，发现原有实现，在检测thd->killed 标识位的时机上存在问题；

经过分析，这是一个两个线程并发产生的问题
线程1：关闭数据库线程
线程2：等待 wait_for_update_bin_log 线程

线程1的流程如下：
（1）将所有thd的kill标志位设上
（2）遍历所有thd，将那些注册过信号量并正进行等待的线程唤醒

线程2的流程如下：
（1）检测thd的kill标志位，如果设置上了退出
（2）如果没有设置kill标志位，那么注册当前thd的信号量和等待量
（3）进入等待

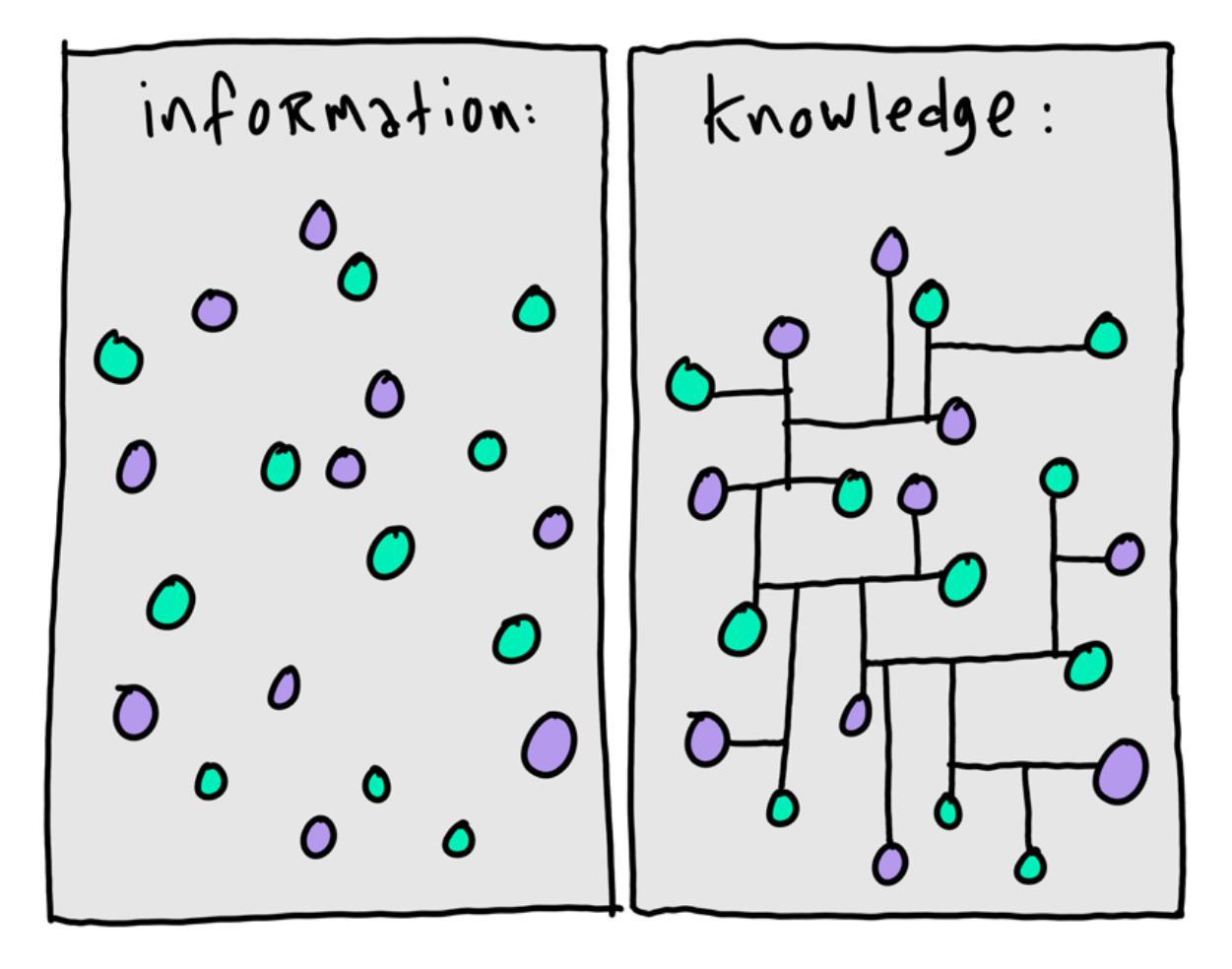
如果按照线程2（1），线程1（1），线程2（2），线程2（2），线程2（3）这样的顺序执行就可能造成测试中出现的问题。

解决方案：
在线程2（2）之后加一个步骤，再检测一次thd的kill 标志位，如果被设置了，那么退出。

  - 注：MySQL Bug http://bugs.mysql.com/bug.php?id=70237 官方未响应；

Step 2 ——构建属于自己的知识体系

- 现在，关于MySQL，你已经有了零散的了解；
- 接下来要做的，就是把这些零散的知识组装起来，形成自己的知识体系。



Step 2 ——MySQL Server Architecture

- MySQL Server Architecture
  - 《Understanding MySQL Internals》. PP-27
  - 了解MySQL源码，尤其是MySQL Server层面的实现，这本书必看，而且需要边看边做实验，边看边跟踪调试代码；

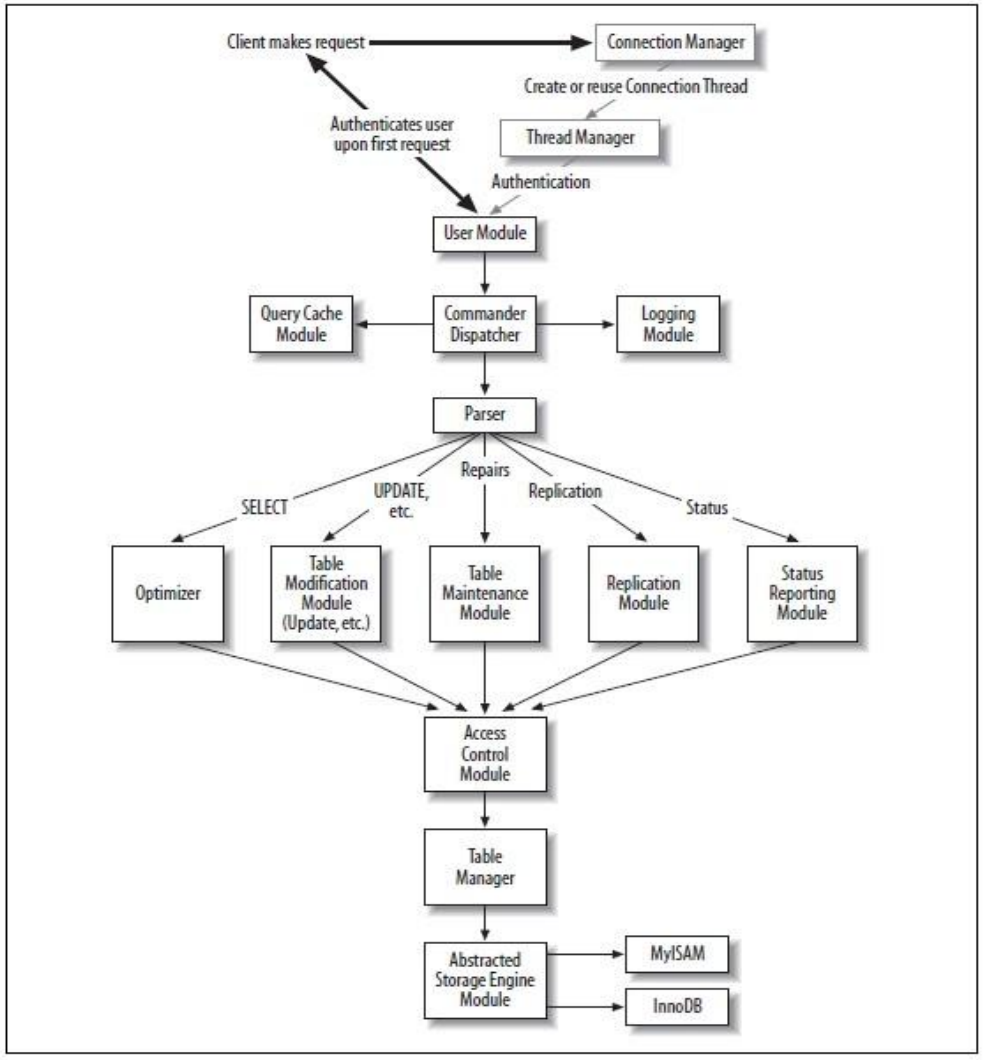
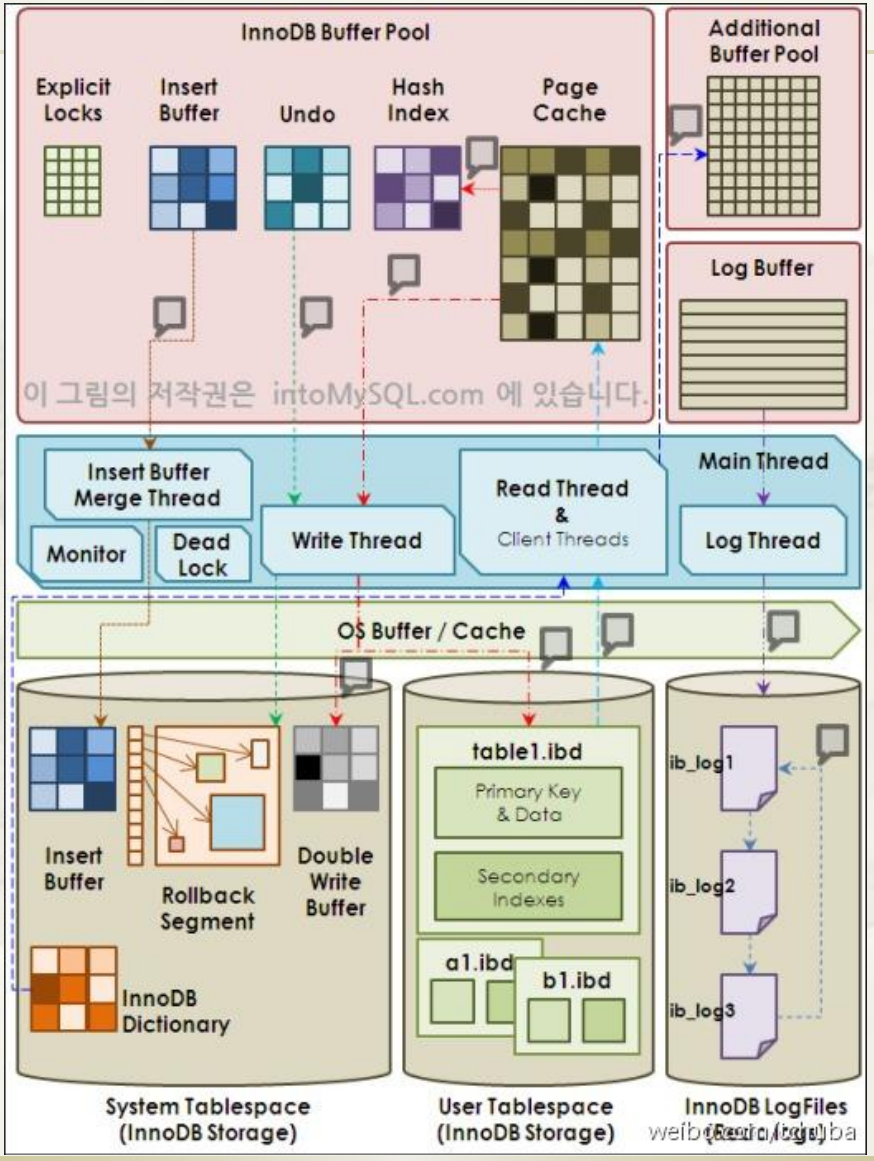


Figure 1-1. High-level view of MySQL modules

Step 2 —— InnoDB Architecture



Step 3 ——注重MySQL技术圈子的收集

- 每一种技术，都有自己的圈子，MySQL也不例外。平时，就要注重MySQL圈子的收集与融入；
- 有哪些做得好，又注重分享的公司：Oracle MySQL，MariaDB，Percona，Google，FB，Twitter，Taobao，NetEase...
- 有哪些值得关注的个人：Mark Callaghan、Jeremy Cole、Dimitri、Peter Zaitsev、Yoshinori Matsunobu ...
- 微博上有哪些值得关注的账号： @姜承尧、@淘宝丁奇、@plinux、@那海蓝蓝 ...
- 业界有哪些好的会议：Percona Live、FOSDEM、MySQL Connect ...
- 哪里去提问和找答案：Google、StackOverflow ...

Step 3 ——跟踪MySQL每个发行版和Bugs

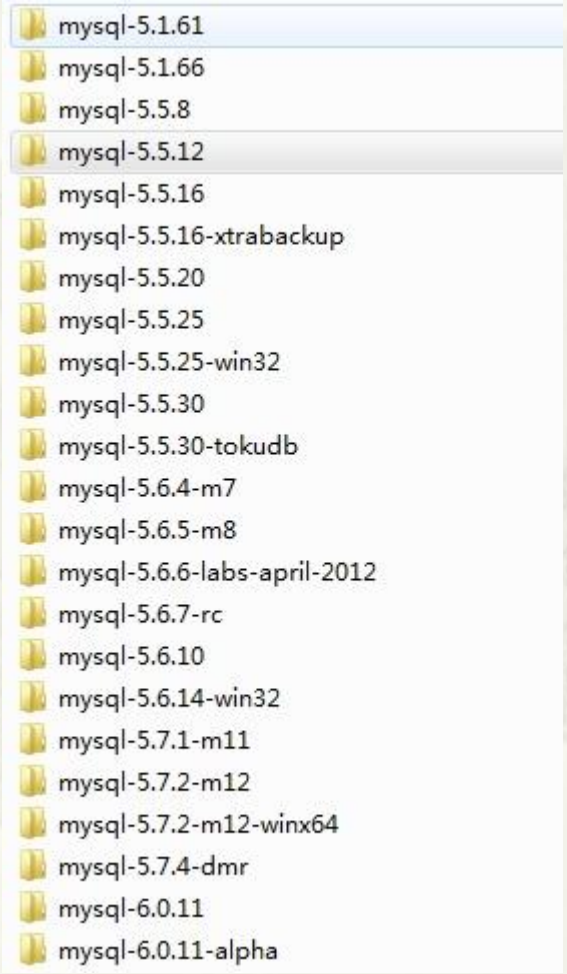
- 关于这一点，个人曾经跟霸爷@淘宝褚霸 有过一次讨论。
  - 霸爷说：在过去几年，跟踪Erlang，把Erlang语言每个版本/每个提交的变更都看了一遍；
  - 我说我没有这么执着，但是我也把MySQL 5.1以来各版本的Release Notes、Bug Fix都跟踪过，新增的功能，都阅读过代码，调试跟踪分析过其实现；
- 要想深入了解MySQL，或者是其他的系统，你需要做到几年如一日的跟踪其发展，才能真正的做到不落伍；
- 一个关于Oracle DBA的典故；
  - 曾经有一位Oracle DBA，被Oracle公司请去对其公司内的Oracle研发人员做关于Oracle系统的培训☺

Step 3 ——跟踪MySQL每个发行版和Bugs(续)

- 哪些地方可以获取这些资料？
- WorkLogs
  - MariaDB：https://mariadb.atlassian.net/secure/Dashboard.jspa
  - MySQL： https://dev.mysql.com/worklog/
  - Percona：https://launchpad.net/percona-server
- Bug 库
  - MySQL Bugs Home：http://bugs.mysql.com/
  - Percona Bugs Home：https://bugs.launchpad.net/percona-server
- 各发行版本
  - 历史版本：http://downloads.mysql.com/archives/community/
  - 当前版本：http://dev.mysql.com/downloads/mysql/

Step 3 ——跟踪MySQL每个发行版和Bugs(续)

* 我的经验



Step 3 ——定期更新自己的前期知识

- 随着对MySQL系统理解的深入，此时应该定期回过头来看看自己早期整理的笔记，撰写的文章，相信我，你一定会发现很多错误，嗤之以鼻的想法。
- 无须沮丧，这说明你的能力提高了，更正他们。

- 个人经验
  - 就InnoDB的锁实现一个功能，近三年内，每当有点新的思路，想法，我就会去重新做测试，看代码，不断纠正自己的想法。

  - 最近的一篇文章：《MySQL 加锁处理分析》

  - 但在我现在看来，这篇文章中仍旧有不正确的地方...

Step 3 ——注重发散知识的积累

- 看懂MySQL源码不是最终目标，当你觉得很多你看懂了，你就会有新的追求，此时，也就意味着需要积累新的知识；
- 对MySQL的并发处理不满意？Kernel_mutex？
  - 需要学习并发编程的相关知识；
- 对MySQL单线程复制不满意？延迟严重？
  - 需要学习MySQL现有复制的实现，进行多线程改造；
- 对MySQL压缩功能不满意？
  - 了解业界成熟的压缩算法，尝试实现并替换；
- 对InnoDB引擎不满意？
  - 自己做一个引擎，你需要进一步了解其他数据库/NoSQL/NewSQL的优点；
- ...

Step 3 —— Keep on going

- 做了前面那么多，你仍旧是在MySQL的圈子之中，如果你有更大的想法，那么是时候跳出这个圈子，寻求更大的挑战。

- 个人经验
  - 在完成多个TNT引擎的版本开发之后，除了继续做这个引擎，我也开始向MySQL之外延生；
  - 分布式数据库系统DDB
  - 分布式KV系统/Redis
  - 感想
    - 以一个数据库的基础，去做这些系统难吗？相信我，真的不难！

Step 3 ——写在最后的建议

- 能坚持到/看到这里的，那绝壁是真爱！！
- 赠送两个小小的建议☺

- 建议一：从handler出发
  - MySQL插件式引擎，连接MySQL Server与各种存储引擎的，是其Handler模块 —— hanlder模块是灵魂；
  - 以InnoDB引擎为例，从ha_innodb.cc文件出发，理解其中的每一个接口的功能，能够上达MySQL Server，下抵InnoDB引擎的内部实现；

- 建议二：不放过源码中的每一处注释
  - MySQL/InnoDB源码中，有很多注释，一些注释相当详细，对理解某一个函数/某一个功能模块都相当有用；

个人推荐的相关书籍

- 首先，以下推荐的书籍，我都保证是自己看过的，有的看过不止一遍；

- MySQL. 《MySQL Reference Manual》
- Baron Schwartz, Peter Zaitsev, Vadim Tkachenko. 《High Performance MySQL, 3rd Edition》
- Sasha Pachev. 《Understanding MySQL Internals》
- J. M. Hellerstein, M. Stonebraker, J. Hamilton. 《Architecture of a Database System》
- Jonathan Lewis. 《Oracle Core: Essential Internals for DBAs and Developers》
- Jonathan Lewis. 《Cost-Based Oracle Fundamentals》
- Steve Adams. 《Oracle8i Internal Services for Waits, Latches, Locks, and Memory》
- Oracle. 《Oracle Data Server Internals： Oracle DSI》
- 姜承尧. 《MySQL技术内幕：InnoDB存储引擎》

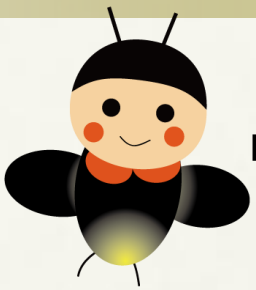
个人推荐的相关博客

- 以下的这些MySQL相关的博客，都是个人订阅，并且每天关注更新的；
- 有时间与经历，建议将这些博客中过去的博文，都看一遍；

- Planet MySQL http://planet.mysql.com/
- Mark Callaghan http://mysqlha.blogspot.com/
- Jeremy Cole http://blog.jcole.us/
- Percona http://www.mysqlperformanceblog.com/
- Oracle InnoDB https://blogs.oracle.com/mysqlinnodb/
- Morgan Tocker http://www.tocker.ca/
- Dimitri KRAVTCHUK http://dimitrik.free.fr/blog/index.html
- Yoshinori Matsunobu http://yoshinorimatsunobu.blogspot.com/
- ...

2014年11月中华架构师大会预告

演讲主题	演讲嘉宾	公司名称	职位/职称
待定	朱超	360	中间件研发负责人
TFS技术架构及运维	张友东	阿里云	TFS研发负责人
待定	黄俊	国药集团	常务副总经理
golang实时消息推送架构实战	毛剑	金山网络	移动游戏技术经理
MyCAT之前世今生	吴治辉	惠普中国	系统架构师
雪球的架构实践	王栋	雪球财经	CTO
待定	刘建平	热璞科技	技术总监



中华数据库
行业协会

中华数据库行业协会

官方网站：www.zhdba.com

官方微信平台：zhdba2014

官方微博：中华数据库行业协会ZHDBA

技术交流QQ群：91596001