北京万里开源软件有限公司

MySQL核心代码开发经验与贡献分享

何振兴 技术副总经理

DTCC2012

自我介绍

- 2004 年加入北京万里开源软件有限公司，负责Turbolinux企业级服务器产品的研发
- 2007 年负责MySQL中国研发中心
- 2009年开始负责万里开源公司MySQL研发团队，分布式数据库系统、Turbolinux操作系统、云计算平台等产品的研发工作

内容

DTCC2012

北京万里开源软件有限公司

公司介绍

DTCC2012

北京万里开源软件有限公司

- 北京万里开源软件有限公司是一家立足于中国的开源软件产品、解决方案和技术服务提供商
- 致力于在中国推广LAMP架构的企业级应用

Turbolinux

Apache

MySQL

PHP

开放源代码的LAMP架构(即Linux、Apache、MySQL、Perl/Python/PHP编程语言)已经与J2EE和.Net形成三足鼎立之势，成为目前最受欢迎的Web应用和开发环境的组合

DTCC2012

万里开源与拓林思

Great OpenSource（国内厂商）

turbolinux（JP）

控股

北京拓林思软件公司(中外合资)

turbolinux

拓林思与万里开源共享所有产品以及产品著作权

DTCC2012

中国最早的MySQL合作伙伴

- 2005年开始与MySQL AB合作，作为当时中国唯一的认证白金合作伙伴，在全国范围提供MySQL产品销售、咨询与支持服务，为关键的企业级数据库应用提供技术支持及数据保障
- 拥有多位MySQL认证的专业工程师
- 2006年与MySQL AB共同建立了中国研发中心
- Oracle的金牌合作伙伴

ORACLE Gold Partner

“MySQL一直非常关注中国市场的发展，希望与我们的合作伙伴一起为中国的开源事业作出更多的贡献。”

—— MySQL创始人David Axmark

DTCC2012

MySQL代码贡献最大的中国研发团体

万里开源MySQL中国研发团队

- 2006年8月29日，MySQL AB与万里开源共同组建MySQL中国研发中心，参与NDB Cluster 的开发
- 2007年10月，开始参与 Replication 的开发
- 2008年1月，Sun收购MySQL之后，MySQL中国研发中心也随之终止，但保留研发团队，继续MySQL的研发合作
- 2009年4月Oracle收购Sun之后仍继续研发合作，同时开始万里开源自己的数据库相关产品的研发
- 2011年年底，万里开源终止与Oracle在MySQL上的研发合作

万里开源致力于中国企业级市场的MySQL推广

MYSQL推广

DTCC2012

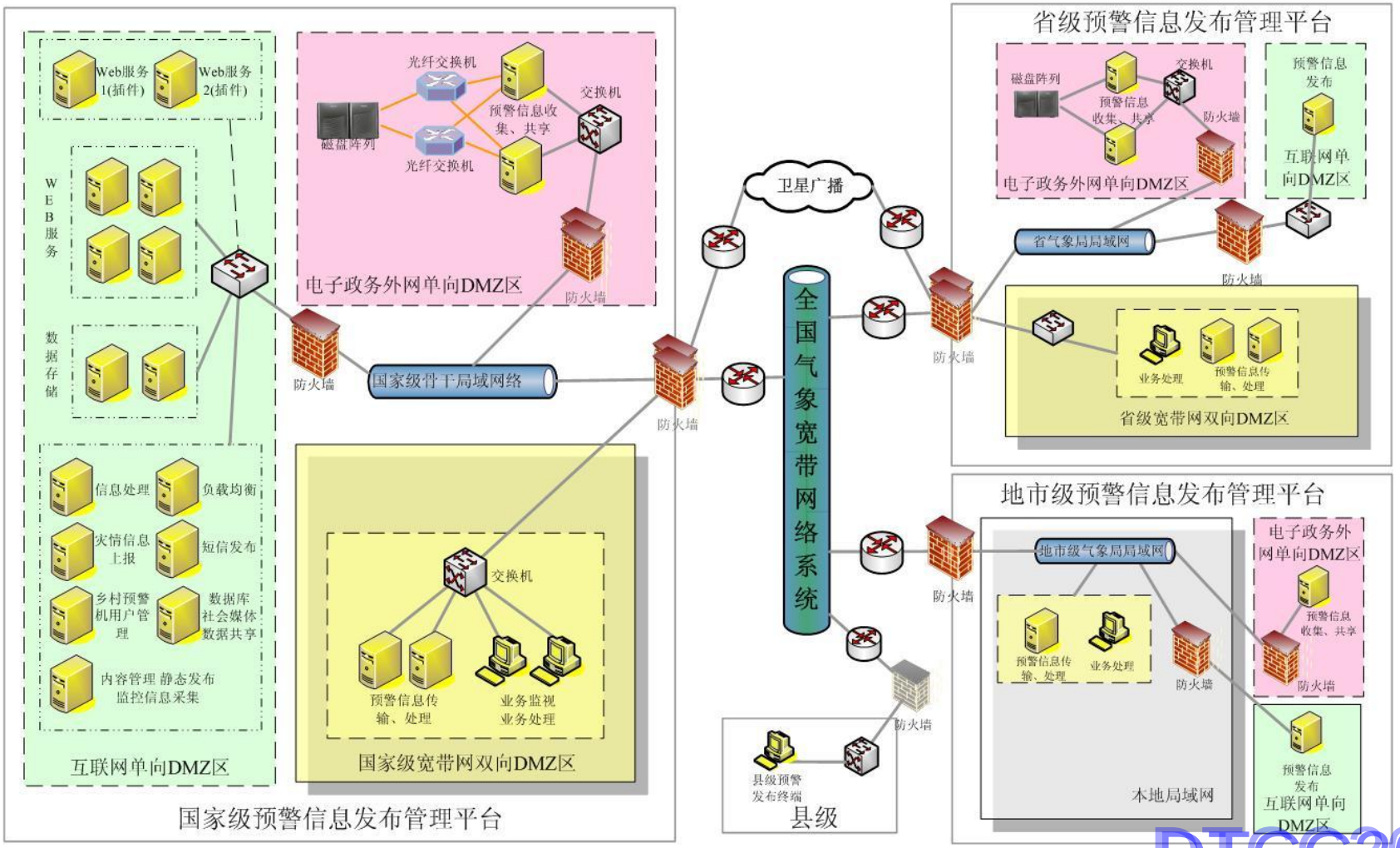
MySQL推广案例

国家突发公共事件预警信息发布系统

- 国家突发公共事件应急体系的重要组成部分
  - 自然灾害
  - 事故灾害
  - 公共卫生事件
  - 社会安全事件
- 整个系统基于B/S架构，国家、省、地市三级部署
  - 1 个国家级突发公共事件预警信息管理发布平台
  - 31 个省级突发公共事件预警信息管理发布平台
  - 342 个地市级突发公共事件预警信息管理发布平台
  - 2379 个县级突发公共事件预警信息管理终端

MySQL推广案例

国家突发公共事件预警信息发布系统总体布局



MySQL推广案例

万里开源推动MySQL中标国家突发公共事件预警信息发布系统

- 国家、省和地市级都包含数据库服务器
- 数据库服务器软件的采购量在700套以上
- 几乎国内外各大数据库软件都参与投标，如 MySQL 、 Oracle、 DB2、Sybase等
- 万里开源在项目开始之初就积极推动MySQL在该项目上的应用，为项目承办方提供广泛深入的MySQL 技术和商务咨询，解除他们对MySQL的担忧，并最终促成MySQL中标国家突发公共事件预警信息发布系统项目的数据库软件

MySQL核心代码开发流程和方法

MYSQL BUG修复

DTCC2012

MySQL BUG修复流程

- QA 验证 Bug
- 分配开发人员，修复（1人），代码审核（2人）
- 开发人员的工作
  - 编写测试用例，用于重现Bug、代码调试和回归测试
  - 调试和分析问题，提出解决方案
  - 修改代码
  - 运行测试用例，通过后提交patch
  - 审核人员对patch进行审核，提出审核意见
  - 根据审核意见修改patch
  - 审核通过后将patch push到bugfixing分支
- 合并到Trunk，技术文档人员更新相应的文档

MySQL测试框架简介

- MySQL 测试框架的基本思想是比较（diff）运行测试产生的输出和预先保存好的预期结果
- 每个测试包含一个测试文件（.test）和一个结果文件（.result）
- 测试文件中可以使用SQL语句和mysqltest指令
- 测试用例依照功能分为多个suite，main suite测试文件放在 mysql-test/t，结果文件放在mysql-test/r；其他suite在mysql-test/suite目录下，如：
- mysql-test/suite/rpl/{t,r}
- mysql-test/suite/binlog/{t,r}

测试用例样本

测试文件 t/sample.test

```
--disable_warnings
DROP TABLE IF EXISTS t1;
--enable_warnings
CREATE TABLE t1 (a INT);
INSERT INTO t1 VALUES(1);
INSERT INTO t1 VALUES(2);
INSERT INTO t1 VALUES(3);
SELECT * FROM t1;
DROP TABLE t1;
```

结果文件 r/sample.result

```
DROP TABLE IF EXISTS t1;
CREATE TABLE t1 (a INT);
INSERT INTO t1 VALUES(1);
INSERT INTO t1 VALUES(2);
INSERT INTO t1 VALUES(3);
SELECT * FROM t1;
a
1
2
3
DROP TABLE t1;
```

运行测试用例

- 使用mysql-test/mysql-test-run.pl 运行测试用例

  $ ./mysql-test-run.pl sample

- 如果运行某个suite下的测试，可以使用--suite选项

  $ ./mysql-test-run.pl --suite=binlog binlog_database

- 可以使用--record选项生成结果文件，结果文件会保存在测试文件对应的结果文件夹中

  $ ./mysql-test-run.pl --record sample

- 脚本mysql-test-run.pl会自动启动mysqld，并将测试文件中的语句发送到mysqld执行，并比较结果

- 使用 --gdb 可以自动启动调试器调试mysqld

  $ ./mysql-test-run.pl –gdb sample

测试Replication功能

- 脚本mysql-test-run.pl支持测试Replication功能，自动启动两个mysqld实例
- 在测试文件中包含 include/master-slave.inc 可以初始化一个主从Replication测试环境
- 通过使用 mysqltest 指令 connect 和 connection 可以在测试文件中连接到master或slave上执行SQL语句

复制测试实例

suite/rpl/t/rpl_drop.test

```
# Testcase for BUG#4552 (DROP on two tables, one of which does not
# exist, must be binlogged with a non-zero error code)
source include/master-slave.inc;
create table t1 (a int);
--error 1051
drop table t1, t2;
--sync_slave_with_master
# End of 4.1 tests
--source include/rpl_end.inc
```

指令 source 用于包含给定文件的内容，通常Replication测试用例都会包含 include/master-slave.inc，该包含文件会初始化主从复制环境，并创建名为master和slave的连接，可使用 connection 指令转换连接，初始连接的是master，可以使用 --connection slave 连接到 slave，之后的语句都将发送到 slave

指令 error 指明下一语句期望产生的错误

指令 sync_slave_with_master 等待slave完成同步

指令推荐使用 -- 格式，以区别于SQL语句

MySQL 测试框架参考资源

- The MySQL Test Framwork, Version 2.0
  - http://dev.mysql.com/doc/mysqltest/2.0/en/index.html
- mysqltest Language Reference
  - http://dev.mysql.com/doc/mysqltest/2.0/en/mysqltest-commands.html
- How to Create Good Tests
  - http://forge.mysql.com/wiki/How_to_Create_Good_Tests
- Test Synchronization
  - http://forge.mysql.com/wiki/MySQL_Internals_Test_Synchronization

Bug#32205

Replaying statements from mysqlbinlog failes with a syntax error, replicates fine

- 现象是使用mysql客户端执行mysqlbinlog dump 出的语句时产生语法错误
- 问题的原因是mysqlbinlog在dump 语句时，会在语句最后添加定界符（/*!*/），但如果原始SQL语句中使用 -- 注释时，就会导致这个定界符也被当成了注释

```
CREATE TABLE t1 (a INT) -- create table;
INSERT INTO t1 VALUES (1);

CREATE TABLE t1 (a INT) -- create table/*!*/
INSERT INTO t1 VALUES (1)/*!*/
```

DTCC2012

Bug#32205 测试用例

```
--source include/have_log_bin.inc


--disable_warnings
DROP TABLE IF EXISTS t1;
--enable_warnings


CREATE TABLE t1 (a INT) -- create table;
INSERT INTO t1 VALUES (1);
FLUSH LOGS;
let $MYSQLD_DATADIR= `select @@datadir`;
--exec $MYSQL_BINLOG $MYSQLD_DATADIR/master-bin.000001 >
      $MYSQLTEST_VARDIR/tmp/binlog_start_comment.binlog
--exec $MYSQL < $MYSQLTEST_VARDIR/tmp/binlog_start_comment.binlog
DROP TABLE t1;
```

MYSQLTEST_VARDIR – 测试用的 var 路径，用于日志、临时文件等

MYSQL_BINLOG – mysqlbinlog 客户端的路径

MYSQL – mysql 客户端的路径

MYSQLD – mysqld 服务器的路径

MYSQL_TEST – mysqltest 客户端的路径

MYSQL_TEST_DIR – 运行测试的mysql-test路径

编译和运行测试用例

- 使用Bazaar获取最新的MySQL源代码

  $ bzr branch lp:mysql-server

- BUILD子目录下有针对各个平台和选项的编译脚本
  - BUILD/compile-pentium-debug-max
  - BUILD/compile-pentium-max
  - BUILD/compile-pentium-debug-max-no-ndb
- 运行测试用例，验证测试用例可以复现bug

  $ cd mysql-test

  $ ./mysql-test-run.pl binlog_start_comment
- 使用--gdb选项运行测试时调试代码

  $ ./mysql-test-run.pl --gdb binlog_start_comment

Bug#32205 修复方法

- 一种方法是分析每个语句，遇到包含 -- 注释的语句时进行特殊处理，比如转换成 /**/ 注释格式
- 另一种办法是在添加定界符之前，先添加一个换行符，将定界符作为单独的一行，而不是加在原来的语句后面

Bug#32205 代码修改（部分）

```
--- a/sql/log_event.cc 2007-12-14 20:05:58 +08:00
+++ b/sql/log_event.cc 2007-12-17 21:13:20 +08:00
@@ -2177,7 +2177,7 @@
  void Query_log_event::print(FILE* file,
  print_query_header(&cache, print_event_info);
  my_b_write(&cache, (uchar*) query, q_len);
- my_b_printf(&cache, "%s\n", print_event_info->delimiter);
+my_b_printf(&cache, "\n%s\n", print_event_info->delimiter);
 }
 #endif /* MYSQL_CLIENT */
```

编译和测试修改后的代码

- 编译修改后的代码

  $ make

- 重新运行测试用例

  $ cd mysql-test

  $ ./mysql-test-run.pl binlog_start_comment

- 运行完整测试，确保修改没有引进regression

  $ ./mysql-test-run.pl --force --parallel=16 --max-test-fail=0

  $ ./mysql-test-run.pl --suite=binlog,rpl --force --parallel=16 --max-test-fail=0

- 所有测试通过后提交代码和测试用例，等待代码审核

万里开源MySQL中国研发团队

MYSQL核心代码贡献

DTCC2012

MySQL中国研发团队的职责

- 参与 NDB Cluster 功能的开发和维护
- 参与 Replication 功能的开发和维护
- MySQL 中文手册的翻译
- 提供本地和全球 MySQL 研发技术咨询

主要贡献一览

- 修复NDB Cluster bug > 100
- 修复Replication bug > 300
- 完成NDB Cluster 新功能（WorkLog） > 10
- 完成Replication 新功能（WorkLog） > 20
- 完成翻译MySQL 5.1 中文参考手册

Replication Show Stopper Bug#26489

- Bug#26489 Corruption in relay log
  - 非常严重的bug（Show Stopper），导致 slave 节点上的 relay log 文件数据损坏，很多重要客户都受到了影响，如Google、爱立信等。
  - 错误出现具有随机性，与网络状况有关，非常难以复现
  - 报告bug后近一年没有任何进展，负面影响极大
  - 中国研发团队在参与Replication研发不久就接受这一挑战，负责修复这个bug
  - MySQL创始人、CTO Michael Widenius （Monty）也亲自参与这个bug的修复，负责审核代码

Bug#26489的原因

- 某个binlog event被分成在两个网络包内发送出去
- 两个包都正常被 slave 接受并保存到 relay log
- 但是由于网络或负载原因，slave 发送的第二个网络包的回复没有能在 master 超时之前到达
- Master 在超时之后重新发送第二个网络包
- Slave 则认为重新发送的第二个网络包为一个新的 binlog event，并写入 relay log

欧洲团队提供的patch

```
--- net_serv.cc.orig    2008-01-25 15:07:49.000000000 +0100
+++ net_serv.cc 2008-01-25 15:20:08.000000000 +0100
@@ -534,13 +534,13 @@
       vio_trace_print(net->vio, "flushing buffer left_length=%lu bytes",
                       left_length);
       memcpy((char*) net->write_pos,packet,left_length);
-      if (net_real_write(net, net->buff,
-                         (size_t) (net->write_pos - net->buff) + left_length)) {
+      size_t const bytes_to_write = (size_t) (net->write_pos - net->buff) + left_length
+      net->write_pos= net->buff;
+      if (net_real_write(net, net->buff, bytes_to_write) {
         vio_trace_print(net->vio,
                         "leave net_write_buff since net_real_write()!=0");
         return 1;
       }
-      net->write_pos= net->buff;
       packet+= left_length;
       len-= left_length;
     }
```

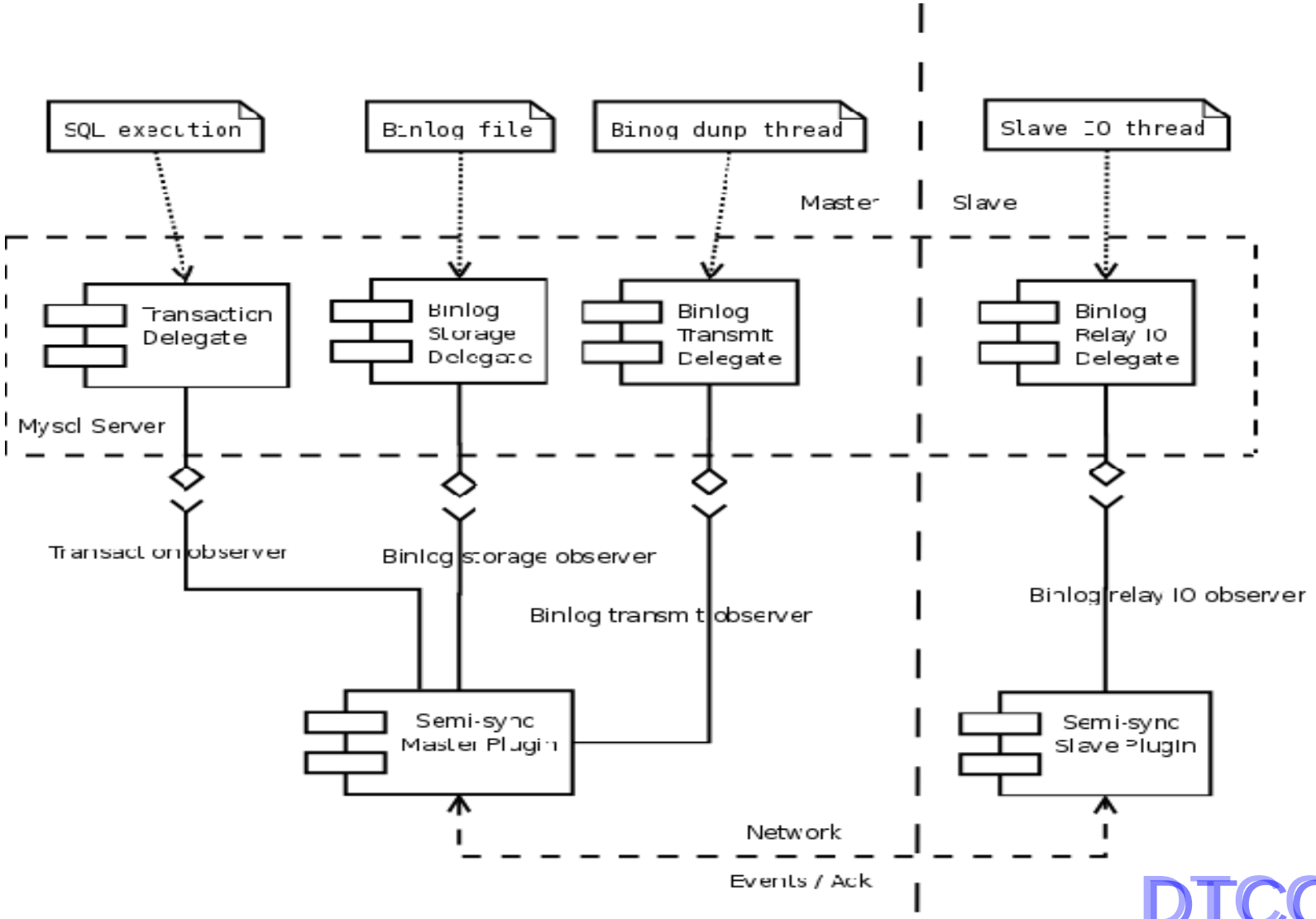
中国研发团队修复bug#26489的patch

```
--- sql_parse.cc.orig   2008-01-28 14:54:29.000000000 +0800
+++ sql_parse.cc        2008-01-28 14:54:40.000000000 +0800
@@ -1166,7 +1166,6 @@
       unregister_slave(thd,1,1);
       /*  fake COM_QUIT -- if we get here, the thread needs to terminate */
       error = TRUE;
-      net->error = 0;
       break;
     }
 #endif
```

新功能举例–Semi-synchronous Replication

- MySQL 5.5最重要的Replication新功能
- 原始patch来自Google，针对MySQL 5.0
- MySQL中国研发团队所做的工作：
  - 移植到MySQL 5.5
  - 设计并添加Replication plugin interface
  - 利用Replication plugin interface来改写Google的patch，将Semi-synchronous功能作为插件，可动态加载和卸载
  - 修改原Google的patch中的bug

Semi-synchronous Replication Plugin 架构图



DTCC2012

新功能举例– Crash-Safe Master/Slave

- 提高Replication功能的可靠性，保证数据的完整性
- 避免由于服务器crash造成的Replication不可用，甚至数据损坏
- Crash-Safe Master
  - 添加binlog自动修复功能，将binlog的修改作为事务的一部分，与事务一同提交、回滚以及恢复
- Crash-Safe Slave
  - 自动修复relay log
  - 将 master.info 和 relaylog.info 文件中的数据保存在相应的MySQL系统表中，对表中数据的修改作为事务执行

新功能举例– Informational Log Event

- 动机是解决Row Event 缺乏原始语句信息的问题，提高Row-based replication 的可读性，便于审计和分析查错等
- 添加一类新的 Informational Log Event，作为所有提供注释、附加信息的 Log Event 的基础类
- 添加Rows_query Log Event，保存Rows Event对应的原始语句
- 在Slave节点上执行Rows Event 时将 Rows_query 中的语句设置为当前语句，保证Rows_query Log Event 能够在Slave上被延续下来

其他开发的新功能一览

- Server UUID
- Replication Checksum
- Time-Delayed Replication
- Remote Binlog Backup
- Row-Based Replication Optimization
- Multi-threaded Slave
- ...

分布式和云数据产品

数据库产品研发

DTCC2012

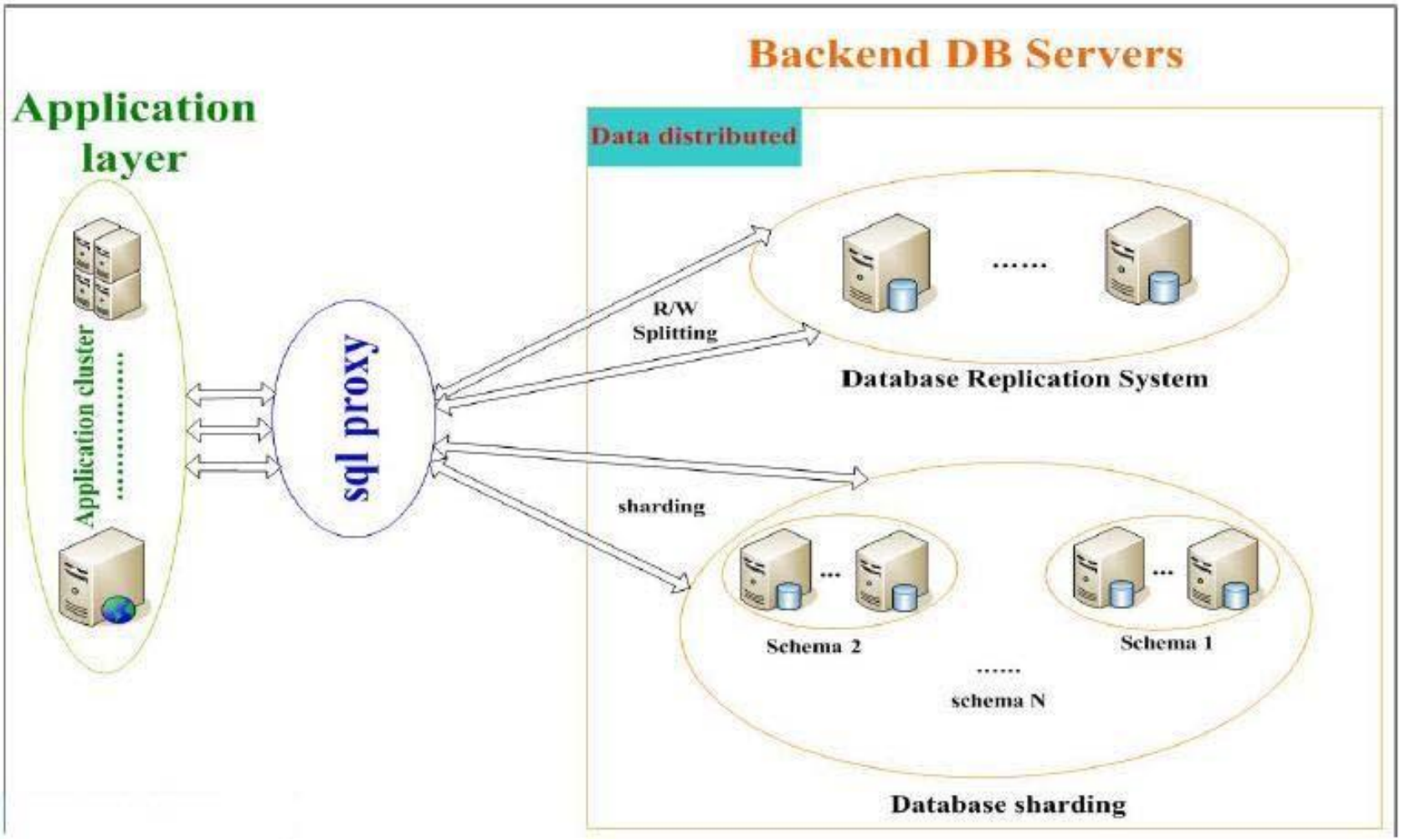
万里开源数据库产品的研发

- 关注MySQL及MariaDB和Percona产品的研发
- 万里开源自身的数据库产品的研发
  - Sqlproxy 分布式数据库系统
  - GreatSQL Cloud 数据库系统

Sqlproxy 分布式数据库

- 基于MySQL通信协议的分布式处理软件
- 解决大数据量，高负载下的数据分布和读写分离
- 类似于MySQl Proxy读写分离，但全部使用C实现，稳定、高效；支持数据分区
- 内置高可用功能，自动处理网络故障和软件故障
- 基于Replication实现自动数据冗余

Sqlproxy 系统架构

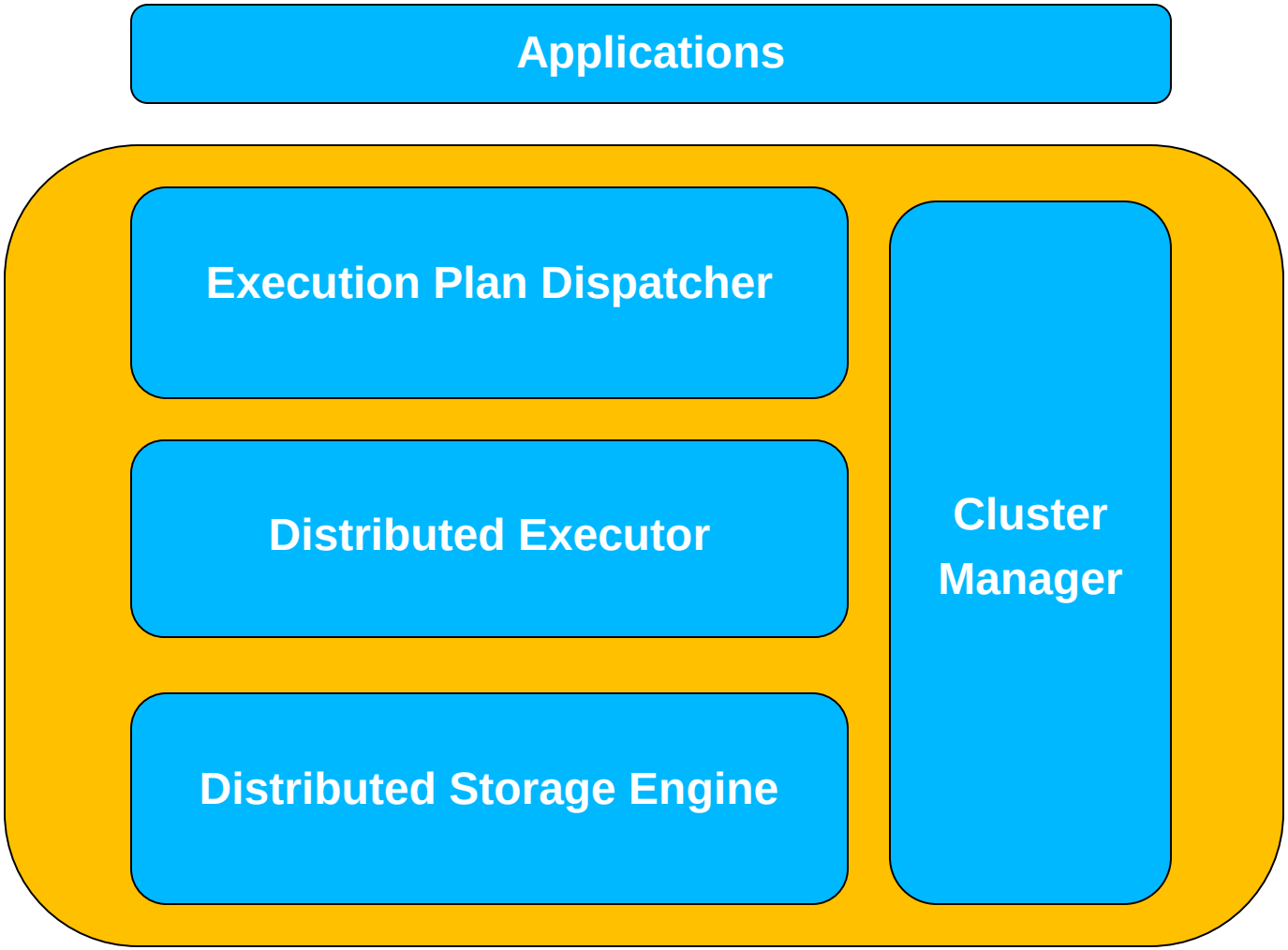


GreatSQL Cloud 数据库系统

- 支持海量数据存储（PB级）
- 支持事务高并发执行
- 高可扩展性，性能可随节点数线性增长
- 高可用性，自动处理节点故障，自动提供数据冗余
- 完全兼容MySQL
- 同时支持SQL和NoSQL
- 支持事务处理
- Share-Nothing架构，无需共享存储
- 同时为OLTP和OLAP优化

DTCC2012

GreatSQL Cloud 架构示意框图



GreatSQL Cloud 设计思想

- 尽量避免使用全局锁，只使用本地锁和区域锁，最大程度实现并发执行
- 最小化数据移动，将执行计划发送到数据所在的节点，对于执行计划产生的中间结果，将根据数据量的大小决定是发送中间结果，还是将需要的数据发送到当前节点
- 基于Share-Nothing架构，不依赖共享存储
- 同时考虑OLTP和OLAP应用的需求
- 针对 SSD 硬盘优化

手机：13910848359
电话：58699916-318
邮箱：hezx@greatopensource.com

北京万里开源软件有限公司

http://www.greatopensource.com

DTCC2012

Great OpenSource