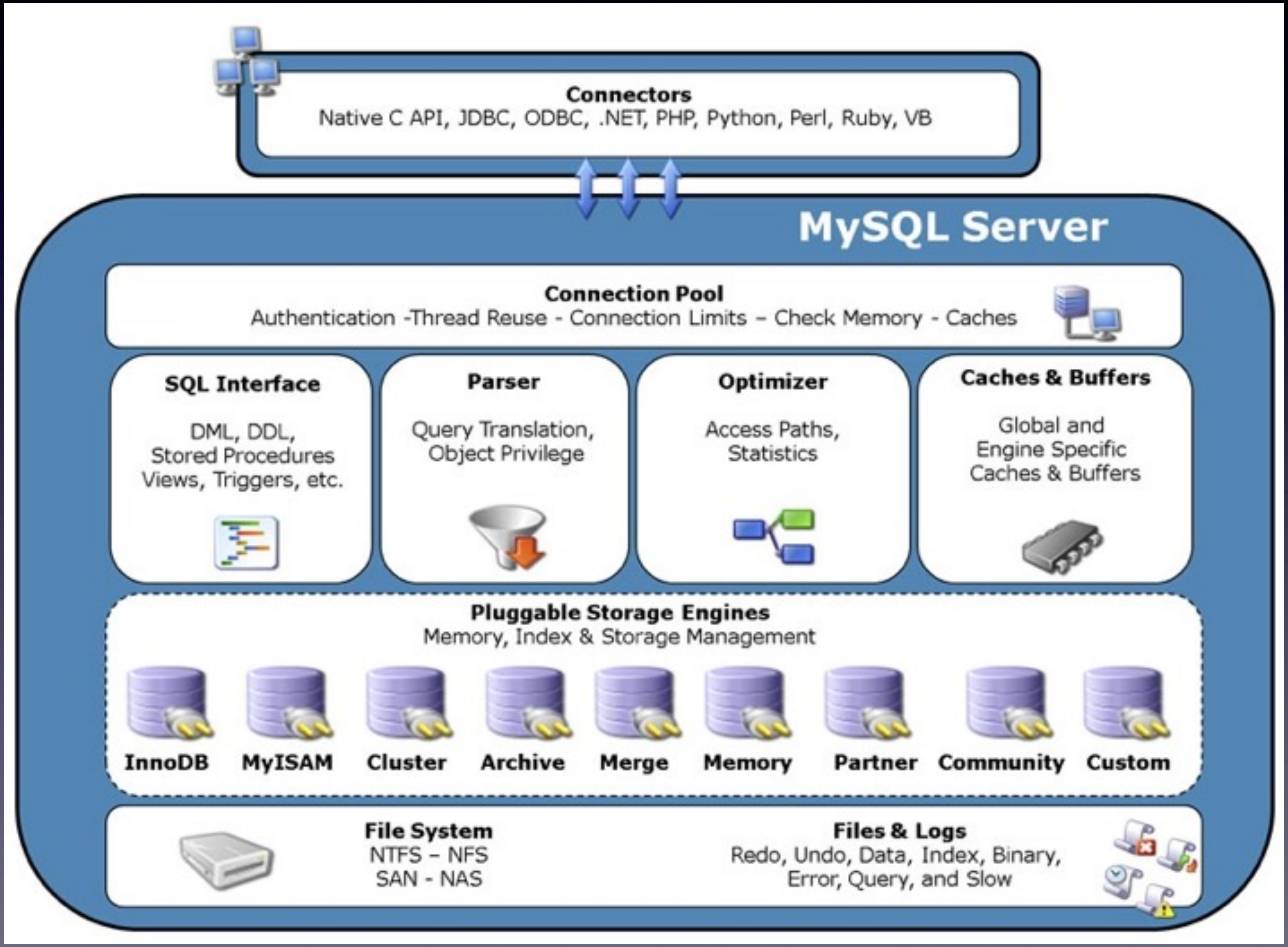
Introduce to MySQL Source Code

娄帅

Architecture



Connectors
Native C API, JDBC, ODBC, .NET, PHP, Python, Perl, Ruby, VB
MySQL Server
Connection Pool
Authentication -Thread Reuse - Connection Limits – Check Memory - Caches
SQL Interface
DML, DDL, Stored Procedures Views, Triggers, etc.
Parser
Query Translation, Object Privilege
Optimizer
Access Paths, Statistics
Caches & Buffers
Global and Engine Specific Caches & Buffers
Pluggable Storage Engines
Memory, Index & Storage Management
InnoDB
MyISAM
Cluster
Archive
Merge
Memory
Partner
Community
Custom
File System
NTFS – NFS
SAN - NAS
Files & Logs
Redo, Undo, Data, Index, Binary, Error, Query, and Slow

Get Source Code

- Git

```
$ git clone https://github.com/mysql/mysql-server.git
$ cd mysql-server
$ git checkout 5.6
```

- Bazzar

```
$ bzr branch lp:mysql-server/5.6 mysql-5.6
```

Directory

- BUILD: build scripts
- client: client programs, such as mysql, mysqladmin
- dbug: DBUG package
- include: header files
- libmysql: generate libmysqlclient.so
- mysql-test: mysql test suites
- mysys: basic system library, structs and algorithm
- plugin: semi_sync, audit, memcached
- sql-bench: MySQL benchmark
- sql: main code， Parser， Optimizer，Executor
- storage：storage engine, MyISAM, InnoDB
- support-files: configuration files and Startup scripts
- vio: Virtual I/O, for tcp, unix socket

Compile

- Dependency

```
cmake, bison, g++, libncurses5-dev
```

- Compile

```
$ mkdir debug && cd debug
$ cmake -DCMAKE_INSTALL_PREFIX=$HOME/mysql-bin  \
-DWITH_DEBUG=1 -DWITH_NDBCLUSTER=0 ..
$ make -j4 && make install
```

Debug

- Install DB

```
$ scripts/mysql_install_db --basedir=$HOME/mysql-bin \
--datadir=$HOME/mysql-bin/var
```

- Modify my.cnf

```
[mysqld]
gdb
debug
basedir = /home/vagrant/mysql-bin
datadir = /home/vagrant/mysql-bin/var
```

- GDB

```
$ gdb --args bin/mysqld --defaults-file=my.cnf
```

DBUG

- Enable DBUG

```
mysql> SET GLOBAL debug = 'debug_options';
mysql> SET SESSION debug = ‘debug_options’;

debug_options = d:t:i:o,/tmp/mysqld.trace
```

- Trace File

```
T@4    : >do_command
T@4    : | >my_net_set_read_timeout
T@4    : | | enter: timeout: 28800
T@4    : | | >vio_socket_timeout
T@4    : | | <vio_socket_timeout 313
T@4    : | <my_net_set_read_timeout 1036
T@4    : | >clear_error
```

Nested-Loop Join

- Init Test Case

```
mysql> create database nested_loop;

mysql> use nested_loop

mysql> create table t1(c1 int);

mysql> create table t2(c2 int);

mysql> insert into t1 values(1),(2),(3);

mysql> insert into t2 values(3),(4),(5);
```

- Test SQL

select * from t1 join t2 on t1.c1 = t2.c2

Nested-Loop Join

- Query Execute Plan

```
mysql> explain  select * from t1 join t2 on t1.c1 = t2.c2;
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | t1    | ALL  | NULL          | NULL | NULL    | NULL |    3 | NULL        |
|  1 | SIMPLE      | t2    | ALL  | NULL          | NULL | NULL    | NULL |    3 | Using where |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
```

- NLJ Algorithm

```
for each row  r1 in t1
    for each row  r2 in t2
        if r1.c1 == r2.c2
            send to client
```

Nested-Loop Join

- Stack

```
> JOIN::exec
| > sub_select         // for each record r1 in t1
| | > handler::ha_rnd_next
| | > evaluate_join_record
| | | > sub_select           // for each record r2 in t2
| | | | > handler::ha_rnd_next
| | | | | > evaluate_join_record       // compare r1.c1 with r2.c2
```

- Time Complexity

O(n1*n2)

Block Nested-Loop Join

- BNL Algorithm

```
for each row  r1 in t1
    store r1 in t2’s join buffer

for each row  r2 in t2
        compare r2 with join buffer
```

- Time Complexity

O(n1/join_buffer_size * n2)

Block Nested-Loop Join

- **Stack**

```
> Join::exec
| > sub_select       // for each record r1 in t1
| | > handler::ha_rnd_next
| | > evaluate_join_record
| | | > sub_select_op         // put to join buffer
| > sub_select       // end store record
| | > sub_select_op
| | | > JOIN_CACHE::join_records // for each record r2 in t2
| | | | > handler::ha_rnd_next
| | | | > JOIN_CACHE_BNL::join_matching_records   // compare r2 with join buffer
```

Hash Join?

- **HJ Algorithm**

```
for each row  r1 in t1
        Hash r1.c1 and Store in HashMap

for each row  r2 in t2
        Hash r2.c2 and compare with t1's HashMap
```

- Time Complexity

O(n1+ n2)

- **MySQL have NO Hash JOIN, Try MariaDB!!!**

Resource

- http://dev.mysql.com/doc/refman/5.7/en/
- http://dev.mysql.com/doc/internals
- https://planet.mysql.com/
- Expert MySQL
- Understanding MySQL Internals
- MySQL核心内幕
- MySQL技术内幕

Q&A