概念：

所有的行为都可以归结为写数据、读数据。

Mysql倾向于使用Scale Out，SQLServer倾向于使用Scale Up。

Mysql是云中最流行的数据存储方式。

innodb\_flush\_log\_at\_trx\_commit和sync\_binlog两个参数是控制Mysql磁盘写入策略以及数据安全的关键参数。

Server：

Usage：

mysqld --verbose --help

mysqld --character-set-server=utf8 --collation-server=utf8\_general\_ci --lower\_case\_table\_names=1

安装：

Docker：

docker run  -e MYSQL\_ROOT\_PASSWORD=123456 -p 3306:3306 -d mysql:5.7.33 --character-set-server=utf8 --collation-server=utf8\_general\_ci --lower\_case\_table\_names=1

docker run -it --rm mysql:5.6.39 --verbose --help

CentOS：

/usr/bin/mysql\*ƒ

/usr/sbin/mysql\*

/etc/init.d/mysql/

/usr/share/mysql/

/etc/my.cnf

/var/lib/mysql/

1. configuration

The MySQL server has many operating parameters, which you can change at server startup using command-line options or configuration files (option files). It is also possible to change many parameters at runtime. For general instructions on setting parameters at startup or runtime, see refer to “Server Command Options”, and “Server System Variables”.

Config File:

my.cnf－位置不确定

[mysqld]

performance\_schema

skip-name-resolve

slow-query-log=TRUE

slow\_query\_log\_file=/path/to/file

#The default value is host\_name-slow.log

#表名小写

lower\_case\_table\_names=1

#字符集

character-set-server=utf8

collation-server=utf8\_general\_ci

#core num \* 2

thread-concurrency=16

back\_log=500

#关闭QueryCache

query\_cache\_type=0

query\_cache\_size=0M

max\_connectoins=1000

max\_allowed\_packet=512M

#读缓存大小

read\_buffer\_size=4M

#随机读缓存大小

read\_rnd\_buffer\_size=16M

#排序缓存大小

sort\_buffer\_size=4M

thread\_cache\_size=64

#批量插入数据缓存大小，默认为8M

bulk\_insert\_buffer\_size=64M

#启用线程池, 5.7

thread\_handling="pool-of\_thread"

#可以设为可用内存的70-80%

innodb\_buffer\_pool\_size=6G

innodb\_buffer\_pool\_instances=6

#文件IO线程数，一般为4

innodb\_file\_io\_threads=4

#对I/O影响较大，对CPU几乎不影响

innodb\_flush\_log\_at\_trx\_commit=0

[server]

TimeZone：

#时区格式

#UTC偏移量

+ 00：00'，'+ 10：00'或'-6：00'

#指定时区模式

'Europe / Helsinki'，'US / Eastern'或'MET'

系统时区：(SYSTEM)

#说明

代表当前系统的时区，该变量为只读变量，不能通过命令行修改。

The system time zone. When the server starts, it attempts to determine the time zone of the host machine automatically and uses it to set the [system\_time\_zone](https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_system_time_zone) system variable. The value does not change thereafter.

To explicitly specify the system time zone for MySQL Server at startup, set the TZ environment variable before you start [**mysqld**](https://dev.mysql.com/doc/refman/5.6/en/mysqld.html). If you start the server using [**mysqld\_safe**](https://dev.mysql.com/doc/refman/5.6/en/mysqld-safe.html), its [--timezone](https://dev.mysql.com/doc/refman/5.6/en/mysqld-safe.html#option_mysqld_safe_timezone) option provides another way to set the system time zone. The permissible values for TZ and [--timezone](https://dev.mysql.com/doc/refman/5.6/en/mysqld-safe.html#option_mysqld_safe_timezone) are system dependent. Consult your operating system documentation to see what values are acceptable.

#check

SELECT @@system\_time\_zone;

show global variables like ‘system\_time\_zone’

全局时区：（默认为SYSTEM）

#set by command

SET GLOBAL time\_zone = '+8:00';

SET GLOBAL time\_zone = 'Europe/Helsinki';

SET @@global.time\_zone='+00:00';

#set by conf

default\_time\_zone='+00:00'

#check

SELECT @@global.time\_zone;

show global variables like ‘time\_zone’

会话时区：

#set by command

SET time\_zone = 'Europe/Helsinki';

SET time\_zone = "+00:00";

SET @@session.time\_zone = "+00:00";

#set by url

?serverTimezone=GMT%2B8

#check

SELECT @@session.time\_zone;

show variables like ‘time\_zone’

Server Command Options:

—<option>[=value]

Options:

--performance\_schema

--slow-query-log

--slow\_query\_log\_file=<value>

Server System Variables:

介绍：

The MySQL server maintains many system variables that indicate how it is configured. Each system variable has a default value. System variables can be set at server startup using options on the command line or in an option file. Most of them can be changed dynamically while the server is running by means of the SET statement, which enables you to modify operation of the server without having to stop and restart it. You can refer to system variable values in expressions.

There are several ways to see the names and values of system variables:

To see the values that a server will use based on its compiled-in defaults and any option files that it reads, use this command:

mysqld --verbose —help

To see the values that a server will use based on its compiled-in defaults, ignoring the settings in any option files, use this command:

mysqld --no-defaults --verbose —help

To see the current values used by a running server, use the SHOW VARIABLES statement.

SHOW VARIABLES LIKE 'perf%’;

SHOW VARIABLES LIKE 'slow%';

SHOW VARIABLES LIKE 'query\_cache%';

Global Variable：

Show：

SHOW GLOBAL VARIABLES;

SHOW GLOBAL VARIABLES LIKE 'perf%’;

Set：(有些变量为只读变量，只能通过conf配置，不能通过command设置)

SET GLOBAL time\_zone = '+8:00';

SET @@global.time\_zone='+00:00';

Variables:

slow query:

set slow-query-log=on

set slow\_query\_log\_file=''

sqlmode:

sql\_mode:

ONLY\_FULL\_GROUP: 不能select超出groupby的字段（未使用聚合函数）

\*\*\*\*\*

Session Variable：

Show：

SHOW SESSION VARIABLES;

SHOW SESSION VARIABLES LIKE 'perf%’;

Set：

SET time\_zone = 'Europe/Helsinki';

SET time\_zone = "+00:00";

SET @@session.time\_zone = "+00:00";

Variables:

set slow-query-log=on

set slow\_query\_log\_file=''

Server Status Variables:

介绍：

The MySQL server maintains many status variables that provide information about its operation. You can view these variables and their values by using the SHOW [GLOBAL | SESSION] STATUS statement. The optional GLOBAL keyword aggregates the values over all connections, and SESSION shows the values for the current connection.

Global Status：

Usage：

SHOW GLOBAL STATUS;

SHOW GLOBAL STATUS LIKE 'Qcache%';

SHOW GLOBAL STATUS LIKE '%wait%';

SHOW GLOBAL STATUS LIKE '%row\_lock%';

SHOW GLOBAL STATUS LIKE '%table\_lock%';

Variables:

Select\_full\_join........... 94

Select\_full\_range\_join..... 0

Select\_range............... 25547

Select\_range\_check......... 0

Select\_scan................ 893514

Slow\_queries............... 1

Sort\_merge\_passes..... 588

Sort\_range............ 289

Sort\_rows............. 154366

Sort\_scan............. 318

#Table\_locks\_immediate 指的是能够立即获得表级锁的次数，而Table\_locks\_waited指的是不能立即获取表级锁而需要等待的次数。如果Table\_locks\_waited的值比较高，则说明存在着较严重的表级锁争用情况。

Table\_locks\_immediate.......... 15173726

Table\_locks\_waited............. 0

Table\_open\_cache\_hits.......... 15177644

Table\_open\_cache\_misses........ 6980

Table\_open\_cache\_overflows..... 5255

Innodb\_row\_lock\_current\_waits..... 0

Innodb\_row\_lock\_time.............. 920

Innodb\_row\_lock\_time\_avg.......... 41

Innodb\_row\_lock\_time\_max.......... 80

Innodb\_row\_lock\_waits............. 22

Sessino Status：

Usage：

SHOW SESSION STATUS;

Variables:

2. 基准测试

基准测试不仅是用来解决业务问题的一种实践行动，也是一种很好的学习方法。学习如何将问题分解成可以通过基准测试来获得答案的方法，就和在数学课上从文字题目中推导出方程式一样。首先正确地描述问题，之后选择合适的基准测试来回答问题，设置基准测试的持续时间和参数，运行测试，收集数据，分析结果数据。

如果你还没有做过基准测试，那么建议至少要熟悉sysbeach。可以先学习如何使用oltp和fileio测试。

性能即是完成某件任务所需要的时间度量，换句话说，性能即响应时间。

性能优化就是在一定的工作负载下尽可能的降低响应时间。

无法测量就无法有效地优化。在对系统进行性能剖析前，必须先要能够进行测量，这需要系统可测量化的支持。

性能剖析是测量和分析时间花费在哪里的主要方法。有两种类型的性能剖析：基于执行时间的分析和基于等待的分析。基于执行时间的分析研究是什么任务的执行时间最长，基于等待的分析则是判断任务在什么地方被阻塞的时间最长。

2. system vars

#查看事务隔离级别

SELECT @@tx\_isolation;

#查看AutoCommit状态

SELECT @@autocommit;

#关闭AutoCommit

SET autocommit = 0;

SET autocommit = OFF;

#启用AutoCommit

SET autocommit = 1;

SET autocommit = ON;

#查看max\_allowed\_packet

select @@max\_allowed\_packet

#查看max-connectoins

select @@max-connectoins

#设置启用查询sql日志表

SET GLOBAL log\_output = 'TABLE';

SET GLOBAL general\_log = 'ON';

select \* from mysql.general\_log;

3. system dbs

sys

statement\_analysis

schema\_tables\_with\_full\_table\_scans

statements\_with\_full\_table\_scans

x$schema\_tables\_with\_full\_table\_scans

x$statements\_with\_full\_table\_scans

mysql

user

help\_topic

information\_schema

USER\_PRIVILEGES

SCHEMATA

SCHEMA\_PRIVILEGES

TABLES

TABLE\_PRIVILEGES

COLUMNS

COLUMN\_PRIVILEGES

VIEWS

TRIGGERS

INNODB\_TRX(事务)

INNODB\_LOCKS(锁)

SELECT b.\*,a.\* FROM information\_schema.INNODB\_TRX AS a LEFT OUTER JOIN information\_schema.INNODB\_LOCKS AS b ON a.`trx\_id` = b.`lock\_trx\_id`;

INNODB\_LOCK\_WAITS(锁等待)

SELECT r.trx\_state wating\_trx\_state,r.trx\_id waiting\_trx\_id,r.trx\_mysql\_thread\_Id waiting\_thread,r.trx\_query waiting\_query, b.trx\_state blocking\_trx\_state,b.trx\_id blocking\_trx\_id,b.trx\_mysql\_thread\_id blocking\_thread,b.trx\_query blocking\_query

FROM information\_schema.innodb\_lock\_waits w INNER JOIN information\_schema.innodb\_trx b ON b.trx\_id = w.blocking\_trx\_id INNER JOIN information\_schema.innodb\_trx r ON r.trx\_id = w.requesting\_trx\_id

GLOBAL\_VARIABLES

INNODB\_PRINT\_ALL\_DEADLOCKS

performance\_schema

users

threads

Client：

mysql [-h localhost|\*\*\*\*\*\*\*] -P [port] -u [root|xxx] [-p ‘password']

1. show command

help show

show [xxx]

show databases

show tables — on current db

show columns from [table]

show index from [table]

show table status like ‘pattern'

show warnings

show errors

show engine innodb status(包含死锁信息)

show processlist

show table status like ‘%’

2. status command

status

show current connection info

3. create db

create schema my\_database collate utf8\_general\_ci;

3. create user

create user <‘name'>@<‘domain'> identified by <‘password’>

% in domain means for any host

4. select user / drop user

select host,user,password from mysql.user

drop user <username>

5. set password

set PASSWORD = PASSWORD('123456')

6. create database

create database <db\_name>;

CREATE DATABASE <db\_name> DEFAULT CHARSET utf8 COLLATE utf8\_general\_ci;

6. grant command

赋予权限（创建用户）：

语法：

GRANT privileges ON object TO user;

**方式一：**

grant <privileges> on <object> to '<user>'@'<domain>' identified by '<password>';

flush privileges;

privilege: \*.\*，all privileges，select，insert，update，delete，create，drop

object: \*.\*, <db>.\*

domain：%，localhost，192.168.1.132

快捷方式:

grant all privileges on \*.\* to 'root'@'%' identified by 'password';

flush privileges;

**方式二：**

CREATE USER 'default\_user'@'%' IDENTIFIED BY '123456';

GRANT ALL ON \*.\* TO 'default\_user'@'%';

删除权限：

语法：

REVOKE privileges ON object FROM user;

查看权限：

show grants for user;

7. import

source #file#

mysql -u #user# -p #pass# #db# < file.sql

8. export

mysql -u #user# -p #pass# #db# > file.sql

9. reset root password

update mysql.user set password=PASSWORD('') where user=root;

flush privileges;

10. set default charset

不加 "GLOBAL" 的话就只对本次会话有效

SET [GLOBAL] character\_set\_client = utf8;

SET [GLOBAL] character\_set\_connection = utf8;

SET [GLOBAL] character\_set\_database = utf8;

SET [GLOBAL] character\_set\_results = utf8;

SET [GLOBAL] character\_set\_server = utf8;

10. show opentables

show open tables where in\_use > 0

10. other commands

describe [table]

delimiter #

exit

11. explain

explain <sql>

12. show profile

set profiling = 1; #open profile

........ exec sqls

show profiles; #show profiles for all sql queries

SHOW PROFILE FOR QUERY [Query\_Id] #show detail for a query

LOCK and UNLOCK TABLES syntax:

LOCK TABLES table\_name [READ | WRITE];

UNLOCK TABLES;

SQL of MySql:

函数:

NOW(), VERSION(), USER(), LAST\_INSERT\_ID()，UUID(), CONNECTION\_ID()

语法：

SELECT:

LIMIT <num>

OFFSET <num>

DATABASE：

CREATE DATABASE <db>

DROP DATABASE IF EXISTS <db>

提示：(显示锁定)

FOR UPDATE

LOCK IN SHARE MODE

事务：

#prepare

SET session TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SET session autocommit=0;

#step 1

BEGIN; or START TRANSACTION;

#inspect current connection

select connection\_id();

#inspect current trax

SELECT tx.\* FROM information\_schema.innodb\_trx tx

WHERE tx.trx\_mysql\_thread\_id = connection\_id();

#inspect current locks

SELECT tx.trx\_id, lc.\* FROM information\_schema.innodb\_trx tx

LEFT JOIN information\_schema.INNODB\_LOCKS AS lc ON tx.trx\_id = lc.lock\_trx\_id

WHERE tx.trx\_mysql\_thread\_id = connection\_id();

#step 2 execute dml

#step 3

COMMIT; or ROLLBACK;

变量：

SET @abc = 123;

select \* from <table> where <column> = @abc;

Insert Or Update:

// Replace 删除后重建

REPLACE INTO abc(a, b, c, d, e)

 SELECT a, b, c, d, e, f

 FROM \*\*\*

 WHERE \*\*\*

//Insert Or Update

INSERT INTO abc(a, b, c, d, e)

 SELECT a, b, c, d, e, f

 FROM \*\*\*

 WHERE \*\*\*

ON DUPLICATE KEY UPDATE a = VALUES(a), b = VALUES(b), c = VALUES(c), d = VALUES(d), e = VALUES(e);

Utilities：

mysqldump::export db

mysqldump -h <host> -u username -p -all-database

mysqldump -h <host> -u username -p dbname [tablename] > exportfile

mysqldump -h <host> -u username -p -d --add-drop-table > export file

#just structure:

mysqldump --routines -h <host> -u username -p  -d dbname [tablename] > exportfile

#just data:

mysqldump -h <host> -u username -p  -t dbname [tablename] > exportfile

mysqladmin

GUI：

MySQL Workbench

Sequel Pro

JDBC Driver:

Configuration properties define how Connector/J will make a connection to a MySQL server. Unless otherwise noted, properties can be set for a DataSource object or for a Connection object.

Configuration properties can be set in one of the following ways:

Using the set\*() methods on MySQL implementations of java.sql.DataSource (which is the preferred method when using implementations of java.sql.DataSource):

com.mysql.jdbc.jdbc2.optional.MysqlDataSource

com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource

As a key-value pair in the java.util.Properties instance passed to DriverManager.getConnection() or Driver.connect()

As a JDBC URL parameter in the URL given to java.sql.DriverManager.getConnection(), java.sql.Driver.connect() or the MySQL implementations of the javax.sql.DataSource setURL() method. If you specify a configuration property in the URL without providing a value for it, nothing will be set; for example, adding useServerPrepStmts alone to the URL does not make Connector/J use server-side prepared statements; you need to add useServerPrepStmts=true.

性能统计：

数据1:

表：

main：20w rows

id(auto\_increment)[\*]

ext\_id[\*]

\*\*\*\*\*

main\_a：20w rows

id(auto\_increment)[\*]

ext\_id

\*\*\*\*\*

main-b：20w rows

id(auto\_increment)[\*]

ext\_id

\*\*\*\*\*

main\_c：20w rows

id(auto\_increment)[\*]

ext\_id

\*\*\*\*\*

main\_d：20w rows

id(auto\_increment)[\*]

ext\_id

\*\*\*\*\*

main\_e：556 rows

id(auto\_increment)[\*]

ext\_id

\*\*\*\*\*

SQL:

SELECT \* FROM main as m

LEFT JOIN main\_a AS ma ON m.ext\_id = ma.ext\_id

LEFT JOIN main\_b AS ma ON m.ext\_id = mb.ext\_id

LEFT JOIN main\_c AS ma ON m.ext\_id = mc.ext\_id

LEFT JOIN main\_d AS ma ON m.ext\_id = md.ext\_id

LEFT JOIN main\_e AS ma ON m.ext\_id = me.ext\_id

WHERE m.ext\_id = '\*\*\*\*'

问题：

CPU使用率比价高

解决：

调查发现，只有main表在ext\_id字段建立了索引，其余表均只有主键索引，执行单条命令发现在600ms左右，这就意味着一条SQL需要占用600ms\* 50%的CPU时间（有网络开销），如果一秒钟有100个并发请求的话就需要600ms \* 100 \* 50%的cpu时间，也就是30s的CPU时间（30核）。

通过为其余表的ext\_id字段建立索引，查询时间缩短到了40ms，如果1秒种有100个并发请求的话就需要40ms \* 100 \* 50%的cpu时间，也就是2s的cpu时间（2核）

Percano：

Percano XtraBackup：

Percona Server：

Debian/Ubuntu下安装：

*Debian* and *Ubuntu* packages from *Percona* are signed with a key. Before using the repository, you should add the key to **apt**. To do that, run the following commands as root:

$ apt-key adv --keyserver keys.gnupg.net --recv-keys 1C4CBDCDCD2EFD2A

Add this to /etc/apt/sources.list, replacing VERSION with the name of your distribution:

deb http://repo.percona.com/apt VERSION main

deb-src http://repo.percona.com/apt VERSION main

Remember to update the local cache:

$ apt-get update

After that you can install the server and client packages

$ apt-get install percona-server-server-5.6 percona-server-client-5.6