MySQL体系结构及原理(innodb)

http://blog.csdn.net/longxibendi

宏观认识

六个问题：

1.为什么delete from table_name速度， 3MB/S 左右

alter一张表，有多快呢？rename 呢？

宏观认识

六个问题：

2.为什么innodb 系统io能力为 200

1.本文所有内容，围绕mysql 5.1版本。Innodb版本为plugin版本。

宏观认识

六个问题：

3.为什么机械磁盘的iops <=180

1.为什么移动硬盘，在copy数据，声音很大。有人用ssd的移动硬盘么？U盘没声音？

宏观认识

六个问题：

4.Insert一条记录，会产生物理**io**读操作么？

1.Update呢，会有物理读io操作么..

宏观认识

六个问题：

5.merge 为什么只适用于非唯一索引的**insert**？

INSERT BUFFER AND ADAPTIVE HASH INDEX

Ibuf: size 738, free list len 16770, seg size 17509,
28472919 inserts, 28349854 merged recs, 15294548 merges

宏观认识

六个问题：

6.为什么建立索引的字段，不允许默认为**NULL**？

宏观认识

应用程序　　mysql、dbproxy
OS　linux、windows、unix、mac
硬件　　内存、CPU、外设、系统(数据)总线、网卡
环境　　网络环境、上下游环境

1.每一层出现问题，都有可能导致整个系统异常。
2.系统的处理能力，取决于每一层自身的处理能力。

宏观认识

外存储分类：机械磁盘(磁介质)、电介质存储

存储的性能

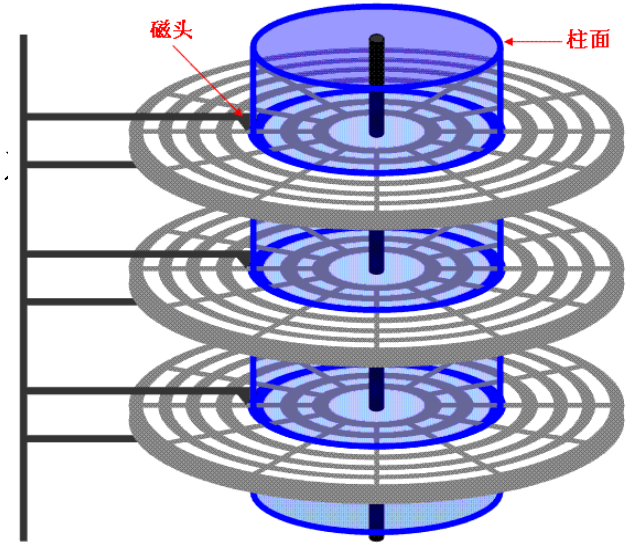
吞吐量　　衡量顺序读、写能力　。每秒，读写量大小。MB/S
IOPS(吞吐率)　　衡量随机读、写能力　。每秒，IO次数。IOPS

OLAP系统，顺序读写需求高。看吞吐量。iqiyi.com。视频网站。
OLTP系统，随机读写需求高。看IOPS。baifubai.com。数据库系统。

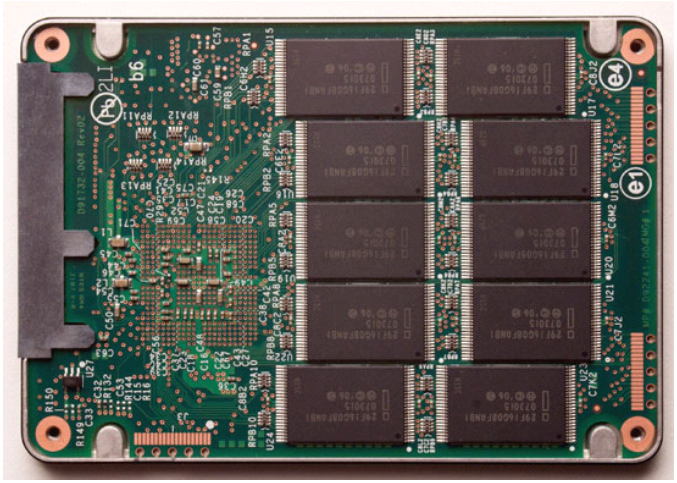
每秒，顺序读写量远远大于随机读写量。
思考：
　　如何测试，随机读写、顺序读写？
　　15K RPM SAS盘，15K/60=250转/秒

1.Innodb为什么要存2份数据。事务日志和ibd?
2.oltp数据库设计只因数据库随机io不给力?

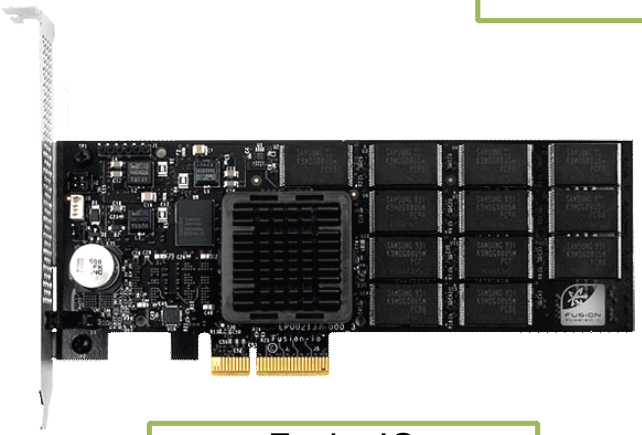
宏观认识



普通机械硬盘
结构图



SSD



FusionIO
iodrive/PCI-E

宏观认识

设备	IOPS	接口
7200 RPM SATA drives	~90 IOPS	SATA II
15k RPM SCSI drives	~180 IOPS	SAS
Intel X25-M G2 (MLC)	~8,600 IOPS	SATA II
ioDrive, a PCI-Express card with Flash	with Flash 140,000 Read IOPS, 135,000 Write IOPS	PCIe
Fusion-io ioDrive Octal	1,180,000+ Random Read/Write IOPS	PCIe

宏观认识

存储的性能

SSD：

- IOPS：随机读35000，随机写5000
- Throughput： 连续读250M，连续写170M
- Latency：75us

磁盘：

- IOPS：随机读 160，随机写 160
- Throughput ： 连续读170M，连续写130M
- Latency：6ms

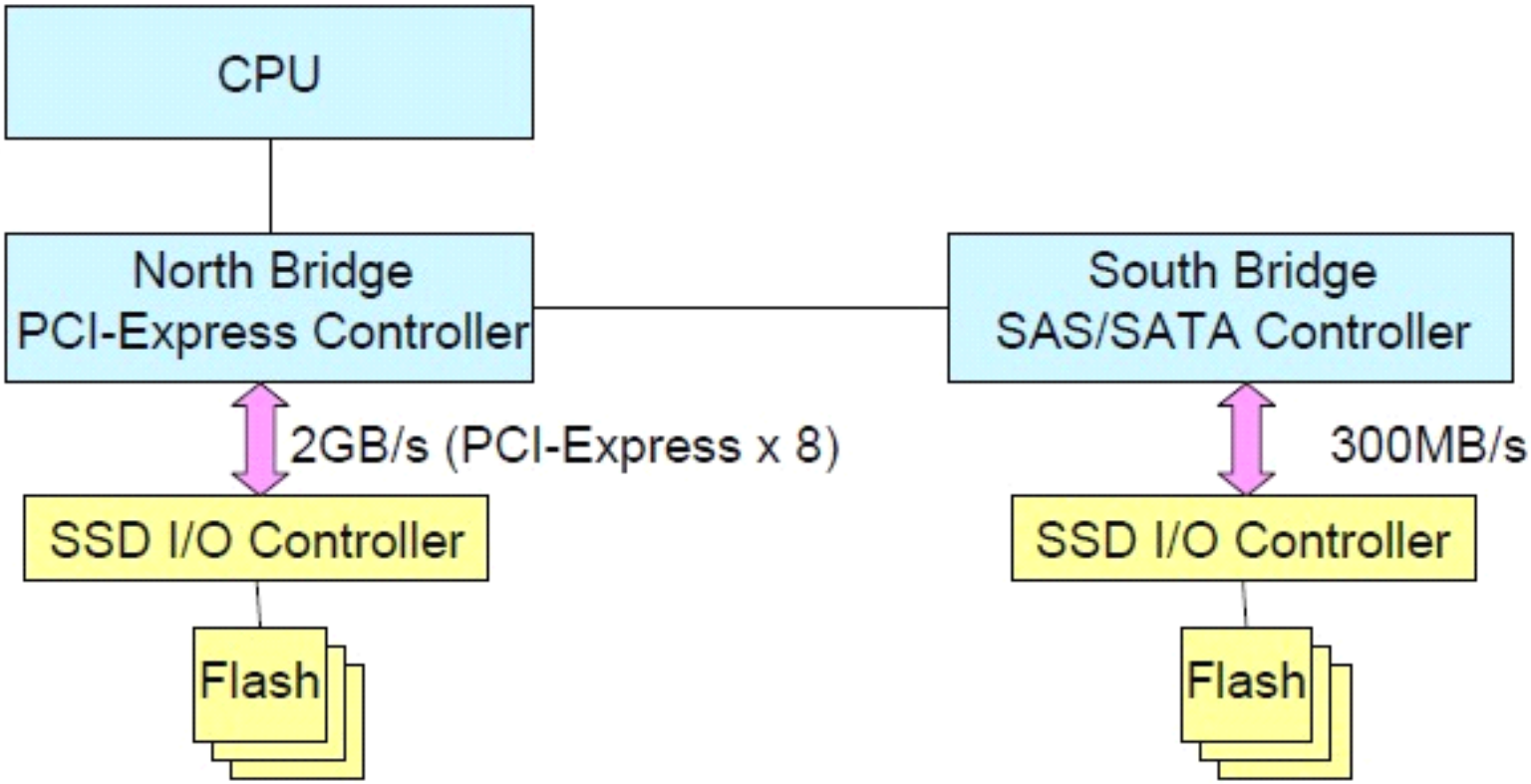
宏观认识

存储的性能

RPM Rotations Per Minute	Rotations Per Second	Rotations Per Mili-second	Full Rotation	Rotational Latency (Half Rotation)	Average Seek Time	IO Time	IOPS
(x)	*(x/60)*	*(x/60,000)*	*(1/ [x/60000])*	*(1/ [x/60000]) / 2*			
				Y	Z	(Y+Z)	(1/[Y+Z])*1000
15,000	**15,000/60**	**15,000/ 60,000**	**4ms**	**2ms**	**4ms**	**6ms**	**167**
10,000	**10,000/60**	**10,000/ 60,000**	**6ms**	**3ms**	**5.15ms**	**8.15ms**	**122**

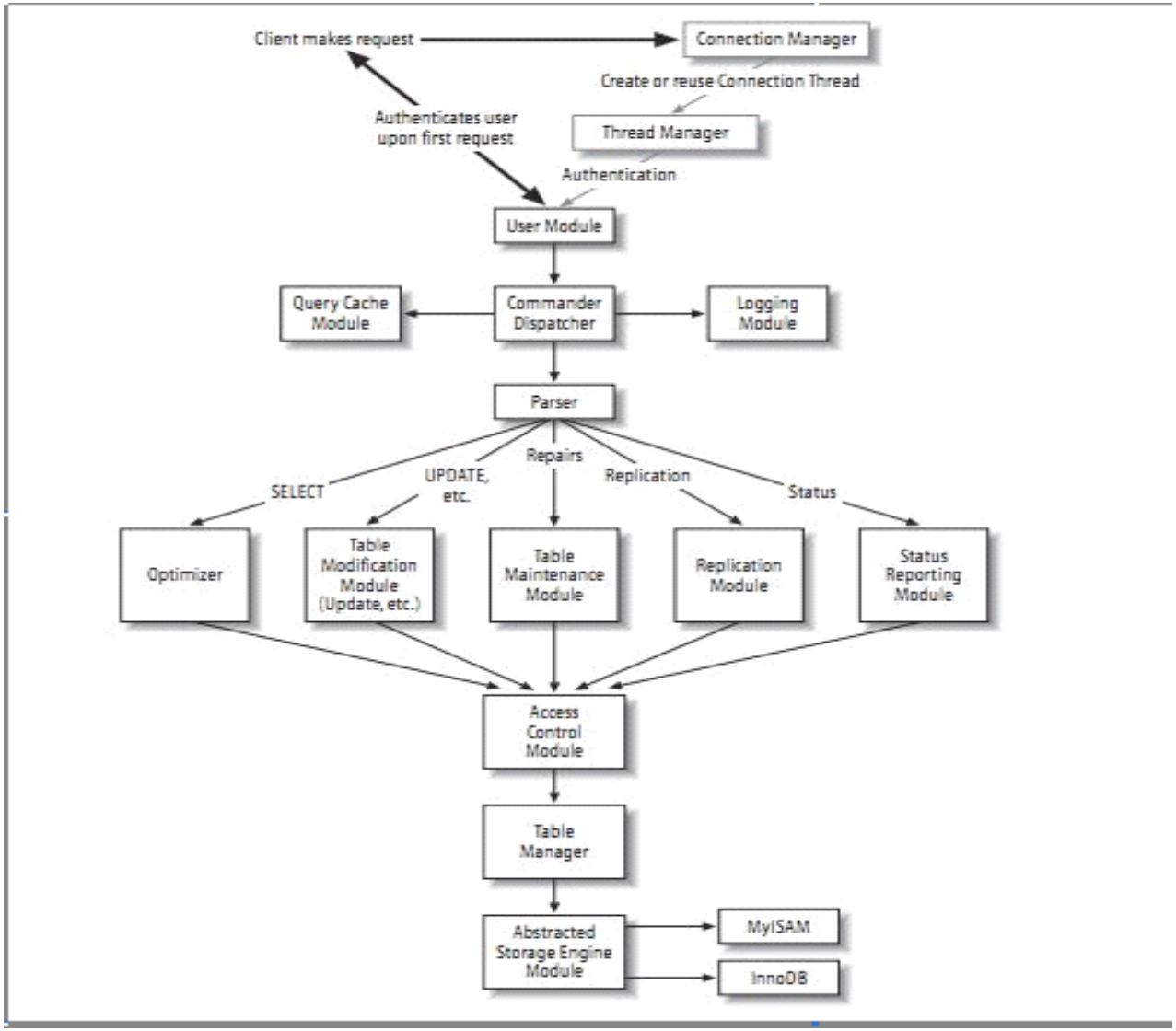
1.10K,2.5寸，sas盘，随机iops，能跑到300么？
2.拍桌子，计算，还是测试？

宏观认识



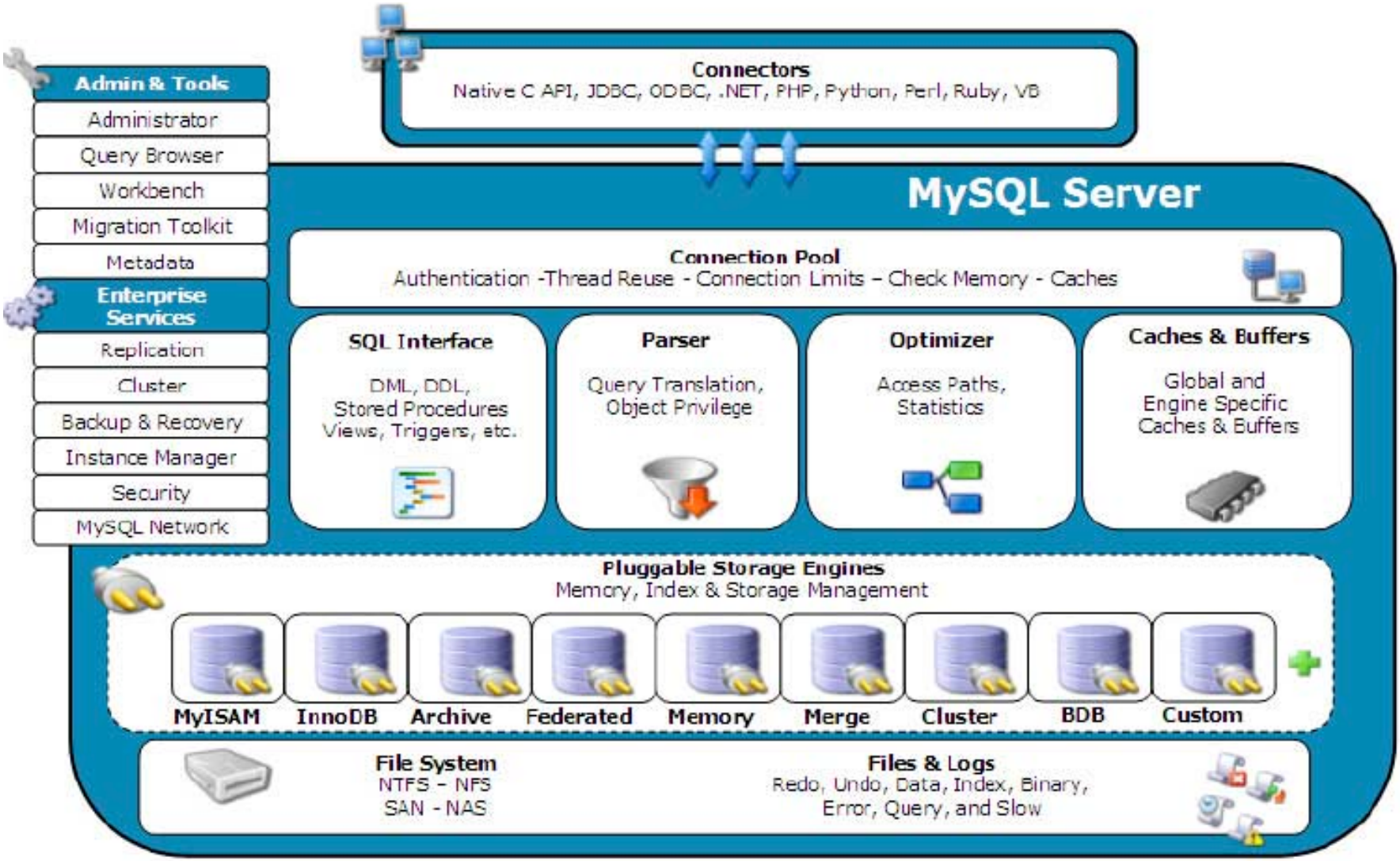
1.存储的读写能力，受限于存储的自身处理能力。
2.就整个系统而言，同样受限于接口的吞吐。
3.write through 模式，5块ssd，做raid5，RAID卡是否能发挥效率，还是会拖后腿，成为瓶颈。

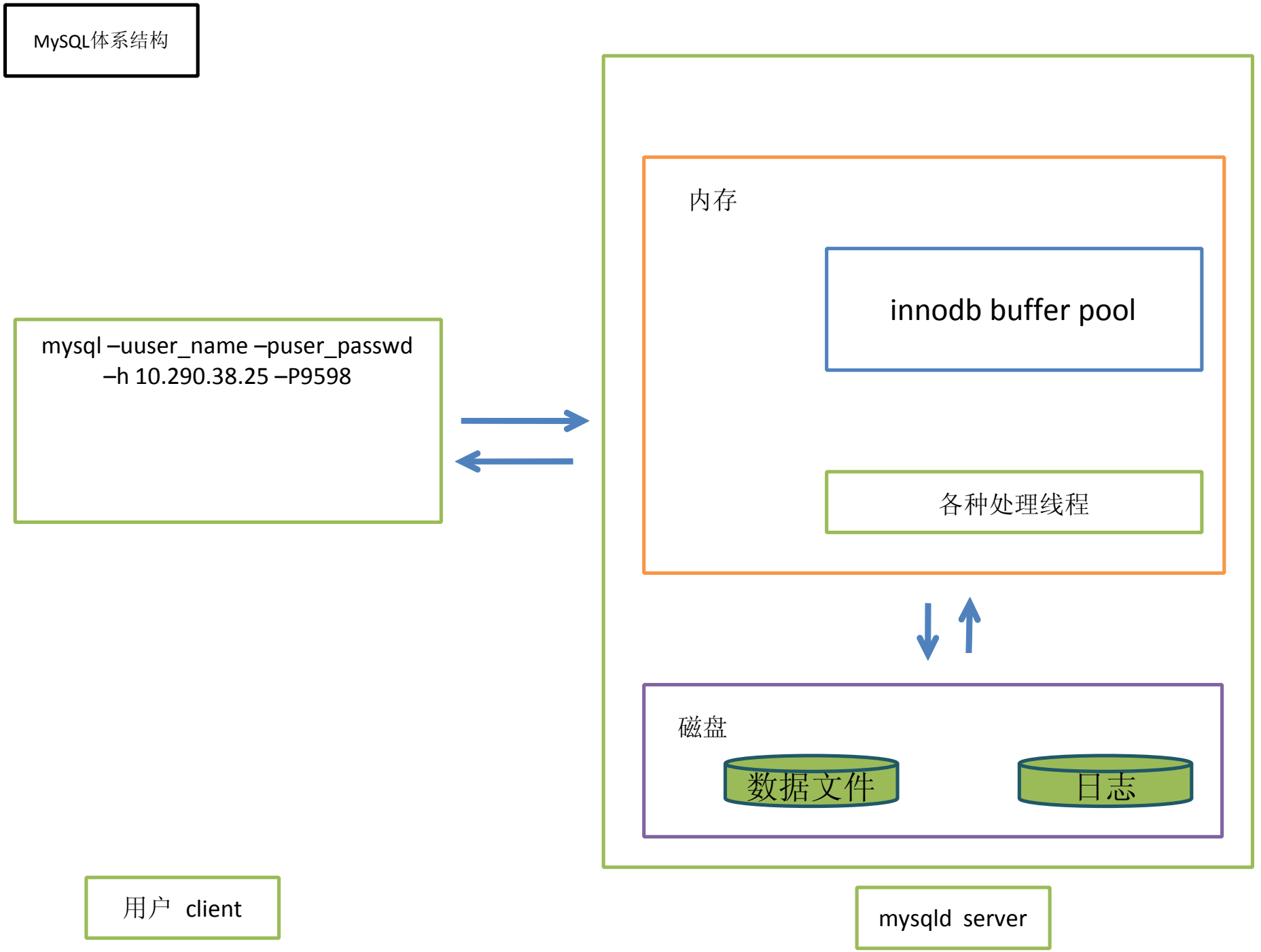
MySQL体系结构



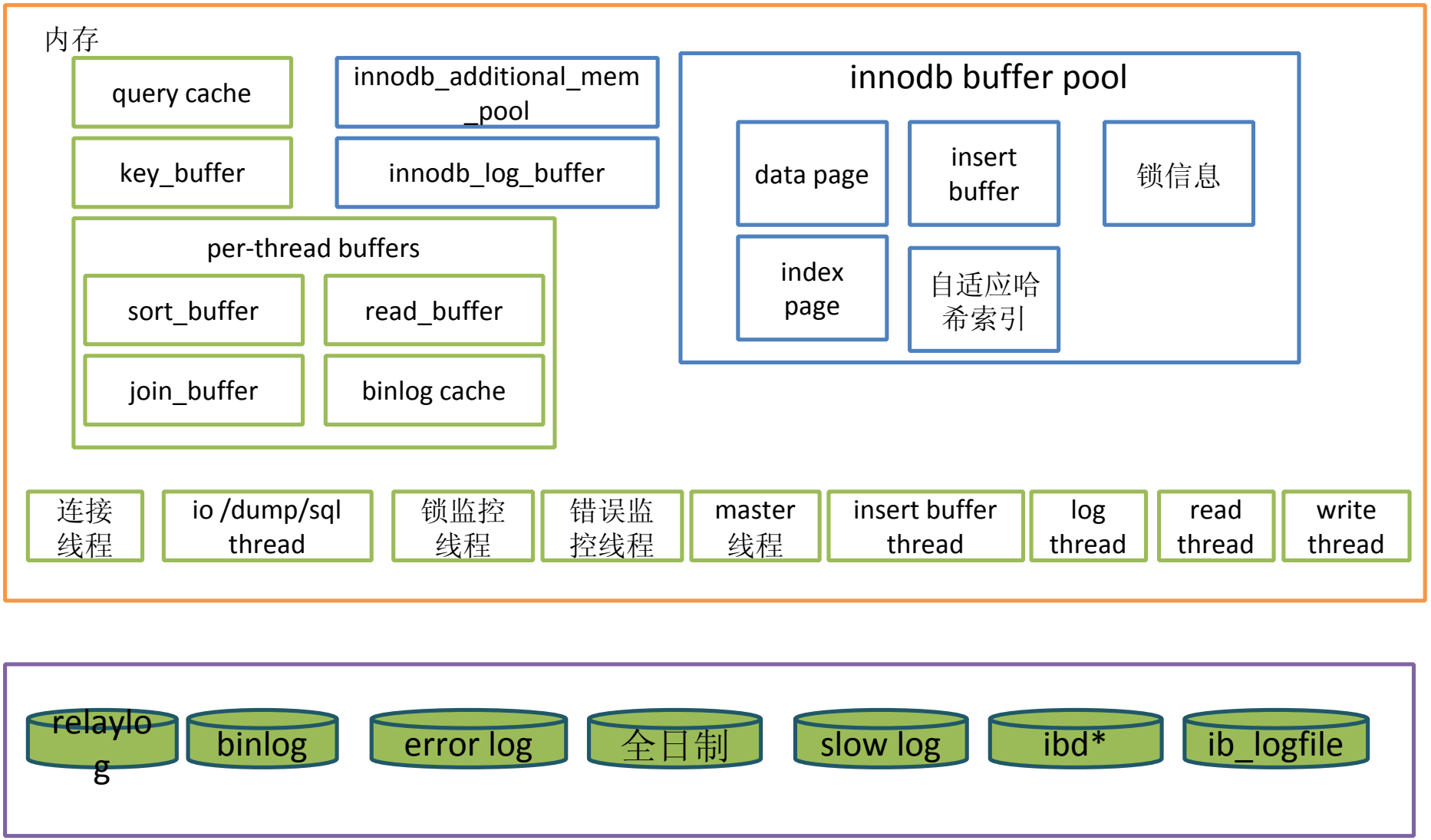
1.SELECT userna FROM fc.userinfo WHERE pa=2;如果query_cache命中; 该sql有没有进入语法解析流程?
2.Innodb做语法解析了么?

MySQL体系结构

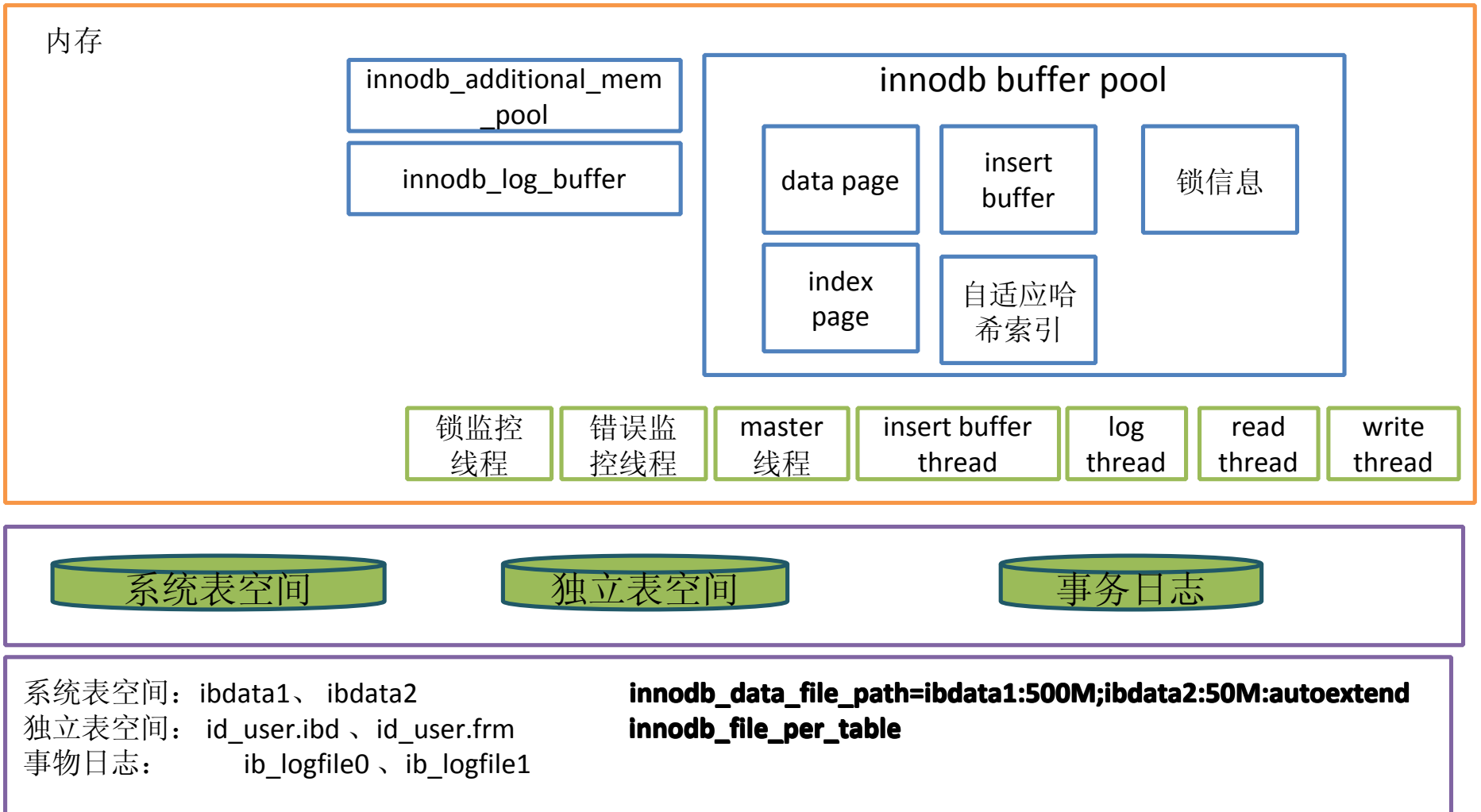

Admin & Tools
Administrator
Query Browser
Workbench
Migration Toolkit
Metadata
Enterprise Services
Replication
Cluster
Backup & Recovery
Instance Manager
Security
MySQL Network
Connectors
Native C API, JDBC, ODBC, .NET, PHP, Python, Perl, Ruby, VB
MySQL Server
Connection Pool
Authentication -Thread Reuse - 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
MyISAM
InnoDB
Archive
Federated
Memory
Merge
Cluster
BDB
Custom
File System
NTFS - NFS
SAN - NAS
Files & Logs
Redo, Undo, Data, Index, Binary,
Error, Query, and Slow


MySQL体系结构
mysql –uuser_name –puser_passwd
–h 10.290.38.25 –P9598
内存
innodb buffer pool
各种处理线程
磁盘
数据文件
日志
用户 client
mysqld server

MySQL体系结构


内存
query cache
innodb_additional_mem_pool
key_buffer
innodb_log_buffer
per-thread buffers
sort_buffer
read_buffer
join_buffer
binlog cache
innodb buffer pool
data page
insert buffer
锁信息
index page
自适应哈希索引
连接线程
io /dump/sql thread
锁监控线程
错误监控线程
master 线程
insert buffer thread
log thread
read thread
write thread
relaylog
binlog
error log
全日制
slow log
ibd*
ib_logfile

MySQL体系结构



系统表空间：ibdata1、 ibdata2
独立表空间： id_user.ibd 、id_user.frm
事物日志： ib_logfile0 、ib_logfile1

innodb_data_file_path=ibdata1:500M;ibdata2:50M:autoextend
innodb_file_per_table

MySQL体系结构

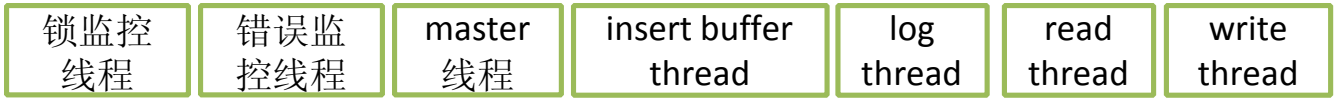
内存

innodb buffer pool

系统表空间：insert buffer、doubel write、数据字典、undo
独立表空间：表数据、索引

1.insert buffer在内存中，还是在磁盘中?
2.聚集索引、非聚集索引在磁盘中？自适应hash在磁盘中？

Innodb有几个线程



Innodb内部，有loop ,background loop ,flush loop ,suspend loop 等后台循环。

```
master_thread(){
        goto loop;
loop:
for (i=0;i<10;i++){
        thread_sleep(1);
        do log buffer flush;
        if (last_one_second_ios<5% innodb_io_capacity)
                do merge 5%   insert buffer
        if (modified_pct>dirty_pct)
                do buffer pool flush 100%
        else if当前所需的dirty page flush速度大于过去20s平均的刷脏页的速度，
                do buffer pool flush 100%
        if (no user activity)
                goto background loop
        }
}
If (last_ten_second_ios<innodb_io_capacity)
        do buffer pool flush 100%
do merge 5%  insert buffer
do log bufer flush;
do full purge; undo page 删除。
If (modified_pct> 70%)
        do buffer pool flush 100%;
else
        do buffer pool flush 10%;
do fuzzy checkpoint
goto  loop
```

background loop:
do full purge ;
do merge 100% insert buffer
If not idle:
goto loop:
else
goto flush loop;

flush loop:
do buffer pool flush 100% dirty page
If (modified_pct>max_dirty_pct)
go to flush loop;
goto suspend loop;

suspend loop:
suspend_thread()
waiting event
goto loop:
}

1.checkpoint到底是干啥的，可不可以没有checkpoint? mysql的checkpoint分类。
2.redo&undo的区别，到底是什么?
3.mysql的停、启过程?

表结构：

```
 CREATE TABLE `test_user_id`
(
  `id` bigint(20) NOT NULL ,
  `UserName` varchar(40) NOT NULL,
   PRIMARY KEY (`id`),
   KEY `index_UserName` (`UserName`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8
```

索引：

Key_name	Column_name	Index_type
PRIMARY	id	BTREE
Index_UserName	index_UserName	BTREE

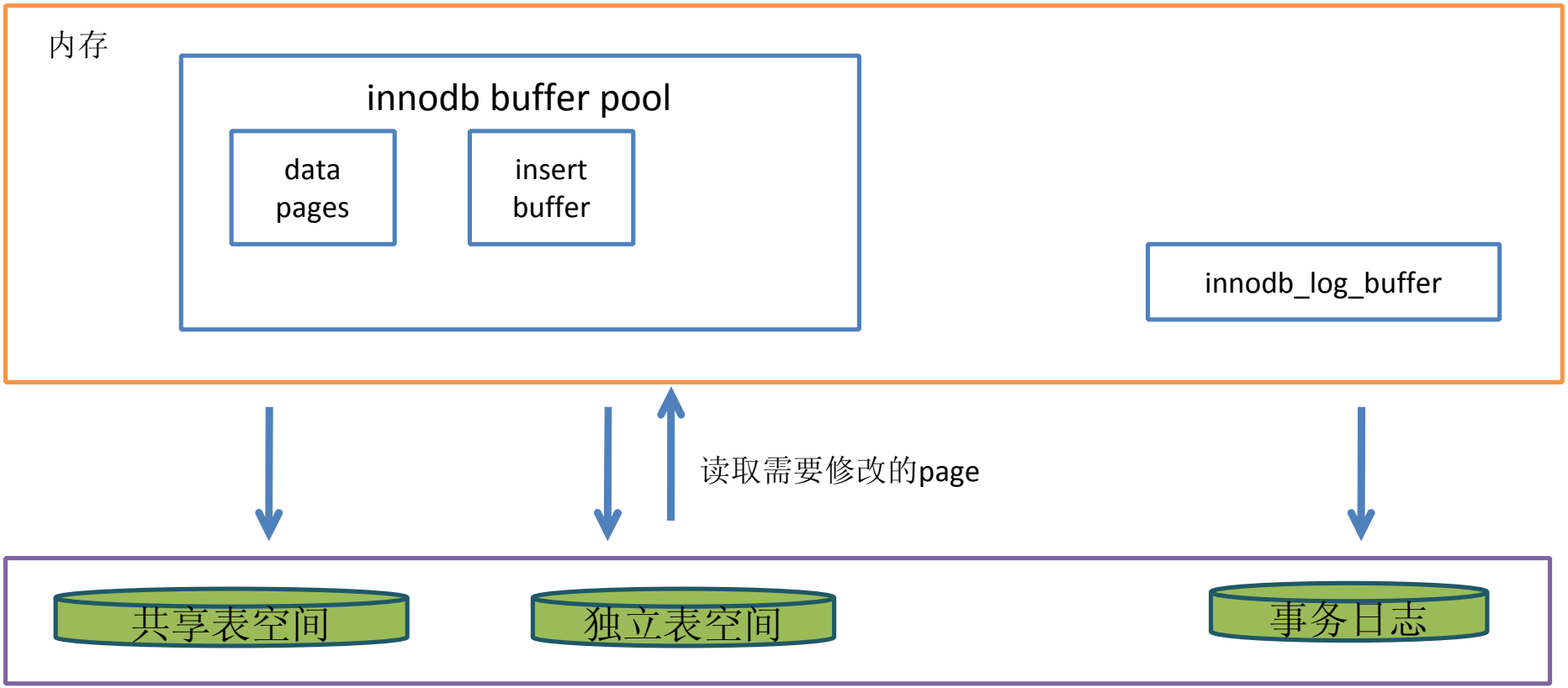
执行下面的语句：

```
UPDATE test_user_id SET  UserName='dbs' WHERE id=168;
INSERT into  test_user_id   values ( 9988,'不停服、不宕机');
```

需要更新 几个索引？

有没有物理io读操作?

MySQL体系结构



1.如果需要修改的page，在内存中(自适应hash)，则直接修改该page即可。否则需要执行一个物理读io，读取page。
2.修改page，并产生undo。
3.事物,commit之后，redo会刷盘。(顺序io) (写事物日志)
4.对于insert操作，因为有KEY `index_UserName` (`UserName`),该索引的更新，可以等待merge。本次写入insert buffer (顺序io) (写系统表空间，insert buffer)。对primary key的insert，与1类似。
5.修改后的page，写入到double write buffer中。(之后刷盘，会顺序io) (写系统表空间)

MySQL体系结构

undo & redo :

start transaction;
UPDATE test_user_id SET UserName='dbs' WHERE id=168; 未提交， 产生**undo**。可用于**rollback**。

commit ; 产生 **redo log** ，可用于重做。

mysql 启动:
1. 读取事物日志，获取最近的checkpoint .

2.遍历系统表空间，获取double_write，检查页面，
同步数据 到独立表空间。

3.遍历该checkpoint后的事物，重做数据。
根据[space_id,page_no]及data,执行redo .
读取事物日志，及表数据文件对应的page到内存。
将事物日志中的data，应用到数据文件中。

4.读取数据字典信息。初始化到内存中。

5.遍历系统表空间，读取undo，执行rollback。

6.启动master 等innodb 线程。

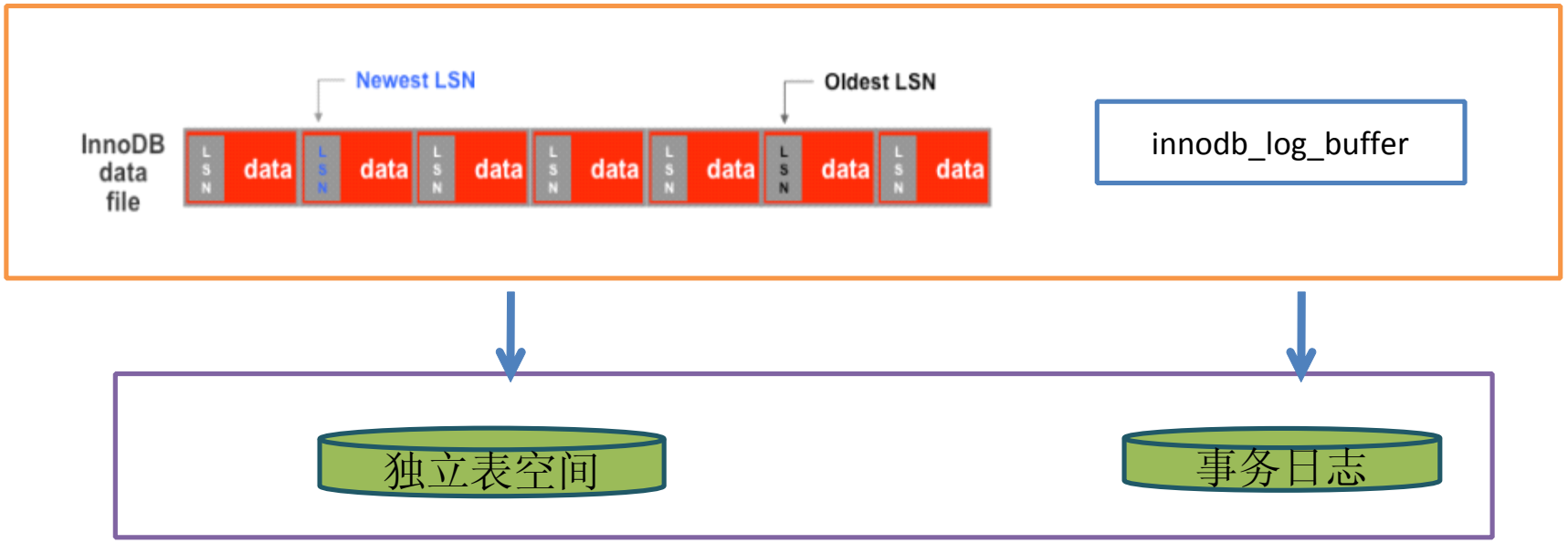
checkpoint at 398 3301091314
LSN 398 3301091315 space_id page_no dataxxxx
LSN 399 1755046031 space_id page_no dataxxxx
LSN 399 2555046031 space_id page_no dataxxxx

MySQL体系结构

mysql checkpoint:

1. sharp checkpoint mysql shutdown使用，会刷所有dirty page到磁盘。

2. fuzzy checkpoint 每10s，执行，只刷oldestLSN之前的page到磁盘。



MySQL体系结构

1.为什么delete from table_name速度， 3MB/S 左右

2.为什么innodb 系统io能力为 200

3.为什么机械磁盘的iops <=180

4.Insert一条记录，会产生物理io读操作么？

5.merge 为什么只适用于非唯一索引的insert？

6.为什么建立索引的字段，不允许默认为NULL？

1. Xtrabackup 的增量备份，能不能每天完成一次增量备份。

MySQL体系结构

1. query_cache 利弊。全局锁。
2. 自适应hash索引的利弊。
3. 多个子系统合并。Innodb_buffer_pool 被不断清理。 innodb_old_blocks_time。
4. 事务日志全局锁，底层并发写为1。
5. 大批量插入/更新大数据，表空间自动扩展，表空间全局锁。

MySQL体系结构

innodb_flush_log_at_trx_commit

当被 设置为0，日志缓冲每秒一次地被写到日志文件，并且对日志文件做到磁盘操作的刷新，但是在一个事务提交不做任何操作。
当这个值为1（默认值）之时，在每个事务提交时，日志缓冲被写到日志文件，对日志文件做到磁盘操作的 刷新。
当设置为2之时，在每个提交，日志缓冲被写到文件，但不对日志文件做到磁盘操作的刷新。尽管如此，在对日志文件的刷新在值为2的情况也每秒发生一次。

我们必须注意到，因为进程安排问题，每秒一次的 刷新不是100%保证每秒都发生。你可以通过设置这个值不为1来获得较好的性能，但随之你会在一次崩溃中损失二分之一价值的事务。如果你设置这个值为0，那么任何**mysqld**进程的崩溃会删除崩溃前最后一秒的事务，如果你设置这个值为2，那么只有操作系统崩溃或掉电才会删除最后一秒的事务。

尽管如此，InnoDB的崩溃恢复不受影响，而且因为这样崩溃恢复开始作用而不考虑这个值。注意，许多操作系统和一些磁盘硬件会欺骗 刷新到磁盘操作。尽管刷新没有进行，你可以告诉**mysqld**刷新已经进行。即使设置这个值为1，事务的持久程度不被保证，且在最坏情况下掉电甚至会破坏InnoDB数据库。在SCSI磁盘控制器中，或在磁盘自身中，使用有后备电池的磁盘缓存会加速文件 刷新并且使得操作更安全。你也可以试着使用Unix命令**hdparm**来在硬件缓存中禁止磁盘写缓存，或使用其它一些对硬件提供商专用的命令。这个选项的 默认值是1。

1. 多级cache。mysql -> 文件系统 ->raid


谢谢