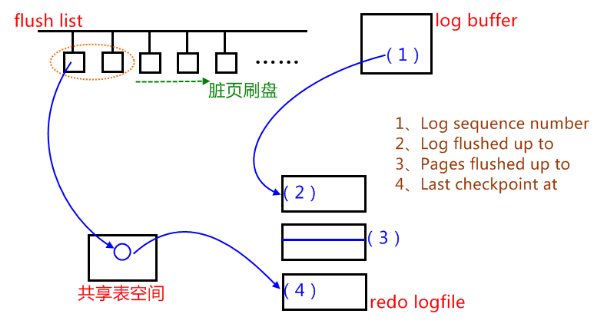
# 背景



**1、我们的日志生成速度？**

1）每天生成多少日志、产生多少redo log

mysql> show global status like 'Innodb\_os\_log\_written';

+-----------------------+--------+

| Variable\_name | Value |

+-----------------------+--------+

| Innodb\_os\_log\_written | 107008 |

+-----------------------+--------+

1 row in set (0.01 sec)

2）如果redolog量大，需要修改如下参数，增加logfile的大小和组数

mysql> show variables like 'i%log\_file%';

+---------------------------+----------+

| Variable\_name | Value |

+---------------------------+----------+

| innodb\_log\_file\_size | 50331648 |

| innodb\_log\_files\_in\_group | 2 |

+---------------------------+----------+

2 rows in set (0.00 sec)

**2、日志写入速度？**

Log buffer有没有满、满的话为什么满

mysql> show variables like 'i%log\_buffer%';

+------------------------+----------+

| Variable\_name | Value |

+------------------------+----------+

| innodb\_log\_buffer\_size | 16777216 |

+------------------------+----------+

1 row in set (0.01 sec)

mysql> show global status like '%log\_pending%';

+------------------------------+-------+

| Variable\_name | Value |

+------------------------------+-------+

| Innodb\_os\_log\_pending\_fsyncs | 0 |

| Innodb\_os\_log\_pending\_writes | 0 |

+------------------------------+-------+

2 rows in set (0.01 sec)

**3、脏页的写入速度？**

1）、Log buffer满了会hang住

2）、Logfile满了不能被覆盖也会hang住

3）、如果脏页写入速度慢的话，logfile满了也不能被覆盖，系统容易hang住，log buffer如果满了的话也容易hang住。

**4、数据库启动时间是多少？**

启动时，默认是要先恢复脏页。当然，能通过参数innodb\_force\_recovery启动控制。

如果innodb\_buffer\_pool很大，32G，极端情况可能有32G的脏页，这个时候如果崩了，恢复的话需要恢复这32G的脏页，时间非常长。

缓冲池的设计目的是为了协调CPU速度与磁盘速度的鸿沟。**页的操作都在在缓冲池中完成的**。如果一条DML语句，如Update或者Delete改变了页中的记录，那么此时页是脏的，即缓冲池中的页的版本比磁盘的要新。数据库需要将最新版本的页刷新到磁盘。

InnoDB采用Write Ahead Log策略来防止宕机数据丢失，即事务提交时，先写重做日志，再修改内存数据页，这样就产生了脏页。既然有重做日志保证数据持久性，查询时也可以直接从缓冲池页中取数据，那为什么还要刷新脏页到磁盘呢？如果重做日志可以无限增大，同时缓冲池足够大，能够缓存所有数据，那么是不需要将缓冲池中的脏页刷新到磁盘。但是，通常会有以下几个问题：

1. 服务器内存有限，缓冲池不够用，无法缓存全部数据
2. 重做日志无限增大成本要求太高
3. 宕机时如果重做全部日志恢复时间过长

事实上，当数据库宕机时，数据库不需要重做所有的日志，只需要执行上次刷入点之后的日志。这个点就叫做Checkpoint，它解决了以上的问题：

1. 缩短数据库恢复时间
2. 缓冲池不够用时，将脏页刷新到磁盘
3. 重做日志不可用时，刷新脏页

重做日志被设计成可循环使用，当日志文件写满时，重做日志中对应数据已经被刷新到磁盘的那部分不再需要的日志可以被覆盖重用。

# 分类

## Sharp CheckPoint

发生在数据库关闭时，会将所有的脏页刷回磁盘。

## Fuzzy CheckPoint

为提高性能，数据库运行时使用Fuzzy CheckPoint进行页的刷新，即只刷新一部分脏页。

**Fuzzy CheckPoint触发时机：**

### Master Thread CheckPoint

差不多以每秒或每十秒的速度，从缓存池脏页列表中刷新一定比例的页，且此过程是异步的，因此不会阻塞其他操作。

1、周期性，读取flush list，找到脏页，写入磁盘

2、写入的量比较小

3、异步，不影响业务

mysql> show variables like '%io\_cap%';

+------------------------+-------+

| Variable\_name | Value |

+------------------------+-------+

| innodb\_io\_capacity | 200 |

| innodb\_io\_capacity\_max | 2000 |

+------------------------+-------+

2 rows in set (0.01 sec)

4、通过capacity能力告知进行刷盘控制

通过innodb的io能力告知控制对flush list刷脏页数量，io\_capacity越高，每次刷盘写入脏页数越多；

如果脏页数量过多，刷盘速度很慢，在io能力允许的情况下，调高innodb\_io\_capacity值，让多刷脏页。

### FLUSH\_LRU\_LIST CheckPoint

因为InnoDB需要保证LRU列表中有一定数量的空闲页可使用，倘若不满足该条件，则会将LRU列表尾端的页移除，若这些页中有脏页，则会进行CheckPoint。该检查被放在一个单独的Page Cleaner线程中进行。

用户可以通过innodb\_lru\_scan\_depth控制LRU列表的可用页数量，默认为1024。

读取lru list，找到脏页，写入磁盘。

mysql> show variables like '%lru%depth';

+-----------------------+-------+

| Variable\_name | Value |

+-----------------------+-------+

| innodb\_lru\_scan\_depth | 1024 |

+-----------------------+-------+

1 row in set (0.01 sec)

此情况下触发，默认扫描1024个lru冷端数据页，将脏页写入磁盘(有10个就刷10，有100个就刷100个……)

### Async/Sync Flush CheckPoint

当重做日志文件不可用的情况下，会强制将一些页刷回磁盘。Async/Sync Flush CheckPoint是为了重做日志的循环使用的可用性。

简单来说，Async发生在要刷回磁盘的脏页较少的情况下，Sync发生在要刷回磁盘的脏页很多时。

这部分操作放入到了Page Cleaner线程中执行，不会阻塞用户操作。

### Dirty Page too much CheckPoint

是指当脏页比例太多，会导致InnoDB存储引擎强制执行CheckPoint。目的根本上还是为了保证缓冲池中有足够可用的页。

比例可由参数innodb\_max\_dirty\_pages\_pct控制。若该值为75，表示当缓冲池中脏页占据75%时，强制CheckPoint。

1、脏页监控，关注点

mysql> show global status like 'Innodb\_buffer\_pool\_pages%t%';

+--------------------------------+-------+

| Variable\_name | Value |

+--------------------------------+-------+

| Innodb\_buffer\_pool\_pages\_data | 2964 |

| Innodb\_buffer\_pool\_pages\_dirty | 0 |

| Innodb\_buffer\_pool\_pages\_total | 8191 |

+--------------------------------+-------+

3 rows in set (0.00 sec)

mysql> show global status like '%wait\_free';

+------------------------------+-------+

| Variable\_name | Value |

+------------------------------+-------+

| Innodb\_buffer\_pool\_wait\_free | 0 |

+------------------------------+-------+

1 row in set (0.00 sec)

1）、Innodb\_buffer\_pool\_pages\_dirty/Innodb\_buffer\_pool\_pages\_total：表示脏页在buffer 的占比

2）、Innodb\_buffer\_pool\_wait\_free：如果>0，说明出现性能负载，buffer pool中没有干净可用块

2、脏页控制参数

mysql> show variables like '%dirty%pct%';

+--------------------------------+-----------+

| Variable\_name | Value |

+--------------------------------+-----------+

| innodb\_max\_dirty\_pages\_pct | 75.000000 |

| innodb\_max\_dirty\_pages\_pct\_lwm | 0.000000 |

+--------------------------------+-----------+

2 rows in set (0.01 sec)

1）、默认是脏页占比75%的时候，就会触发刷盘，将脏页写入磁盘，腾出内存空间。建议不调，调太低的话，io压力就会很大，但是崩溃恢复就很快；

2）、lwm：low water mark低水位线，刷盘到该低水位线就不写脏页了，0也就是不限制。

注意：上面在调整的时候，要关注系统的写性能iostat -x。

# 原理

## LSN

InnoDB引擎通过LSN(Log Sequence Number)来标记版本，LSN是日志空间中每条日志的结束点，用字节偏移量来表示。**每个page有LSN，redo log也有LSN，Checkpoint也有LSN**。可以通过命令show engine innodb status来观察：

---

LOG

---

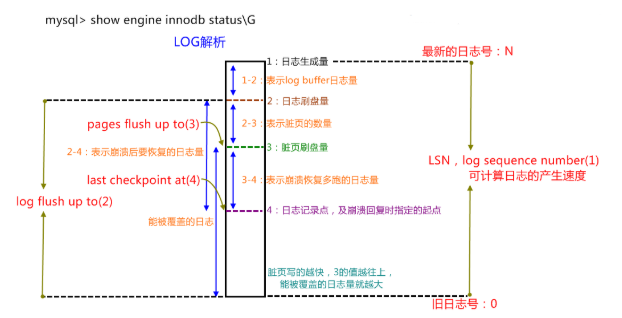
Log sequence number 11102619599

Log flushed up to 11102618636

Last checkpoint at 11102606319

0 pending log writes, 0 pending chkp writes

15416290 log i/o's done, 12.32 log i/o's/second

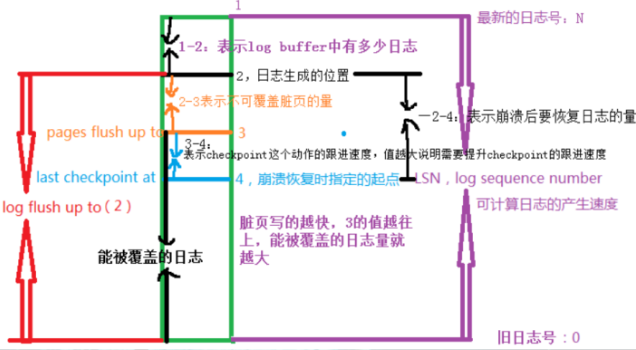


总结：

1. LSN是用来标记版本的
2. LSN是8字节的数字
3. 每个页有LSN，重做日志也有LSN，CheckPoint也有LSN

## Flush刷新

由于Checkpoint和日志紧密相关，将日志和Checkpoint一起说明，详细的实现机制如下：



如上图所示，Innodb的一条事务日志共经历4个阶段：

1）创建阶段：事务创建一条日志；

2）日志刷盘：日志写入到磁盘上的日志文件；

3）数据刷盘：日志对应的脏页数据写入到磁盘上的数据文件；

4）写CKP：日志被当作Checkpoint写入日志文件；

对应这4个阶段，系统记录了4个日志相关的信息，用于其它各种处理使用：

Log sequence number（LSN1）：当前系统LSN最大值，新的事务日志LSN将在此基础上生成（LSN1+新日志的大小）；

Log flushed up to（LSN2）：当前已经写入日志文件的LSN；

Pages flushed up to（LSN3）：当前最旧的脏页数据对应的LSN，写Checkpoint的时候直接将此LSN写入到日志文件；

Last checkpoint at（LSN4）：当前已经写入Checkpoint的LSN；

对于系统来说，以上4个LSN是递减的，即： LSN1>=LSN2>=LSN3>=LSN4.

具体的样例如下（使用show engine innodb status \G命令查看）

---

LOG

---

Log sequence number 1039878815567

Log flushed up to 1039878815567

Pages flushed up to 1039878814486

Last checkpoint at 1039878814486

0 pending log writes, 0 pending chkp writes

5469310 log i/o's done, 1.00 log i/o's/second

# 作用

1、缩短数据库的恢复时间

当数据库宕机时，数据库不需要重做所有日志，因为CheckPoint之前的页都已经刷新回磁盘。只需对CheckPoint后的重做日志进行恢复，从而缩短恢复时间。

2、缓冲池不够用时，将脏页刷新到磁盘

当缓存池不够用时，LRU算法会溢出最近最少使用的页，若此页为脏页，会强制执行CheckPoint，将该脏页刷回磁盘。

3、重做日志不可用时，刷新脏页

不可用是因为对重做日志的设计是循环使用的。重做日志可以被重用的部分，是指当数据库进行恢复操作时不需要的部分。若此时这部分重做日志还有用，将强制执行CheckPoint，将缓冲池的页至少刷新到当前重做日志的位置。