第10章、InnoDB存储引擎

1.innodb存储引擎体系结构  
2.数据库查询性能提升-缓冲池buffer pool  
3.刷写脏页check point  
4.重做日志-redo log  
5.回滚日志-undo log  
6.DML语句性能提升技术-insert buffer、change buffer  
7.数据写入可靠性提升技术-double write buffer  
8.自适应哈希索引-AHI  
9.innodb预读预写技术  
10.参数文件  
11.错误日志文件  
12.慢查询日志文件  
13.二进制日志文件  
14.InnoDB存储引擎表空间文件  
15.InnoDB存储引擎表空间结构  
16.表碎片整理  
17.表空间文件迁移  
​

**1.innodb存储引擎体系结构**

show engines \G;

default\_storage\_engine | InnoDB

innodb存储引擎体系结构：

内存：

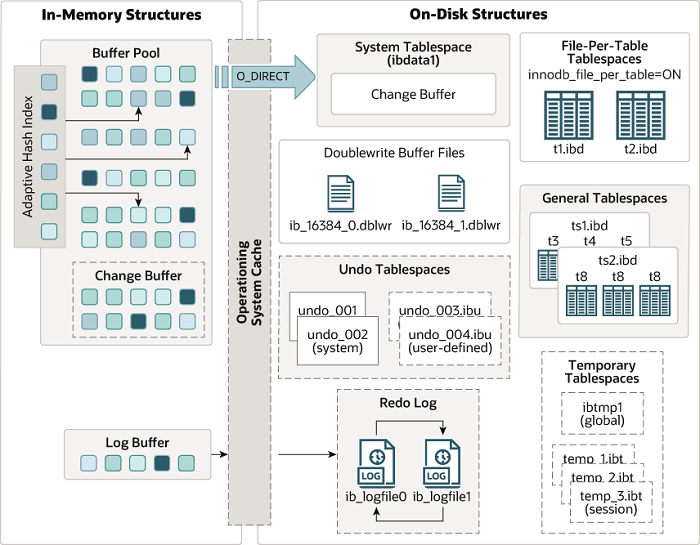
内存：BUFFER POOL、AHI、CHANGE BUFFER、REDO LOG BUFFER...

文件：ibd（独立表空间）、共享表空间、error log、binlog、slow log、general log

线程：master thread、page cleaner thread 、io thread ...

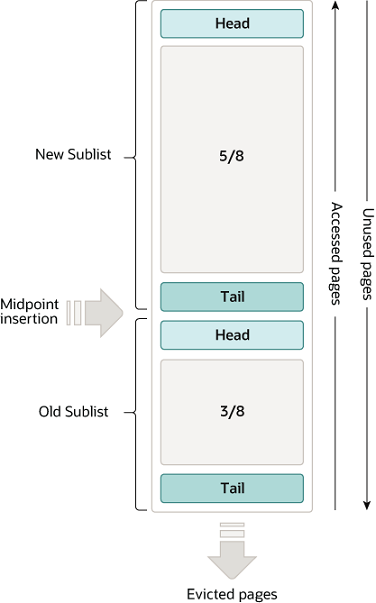
innodb存储引擎的特点：

支持多缓冲区池、ahi、聚集索引、mvcc、自动故障恢复、热备、行锁、事务、外键



**2.提升数据库性能-buffer pool**

<https://www.pianshen.com/article/44151236867/>



①缓冲池作用：缓存磁盘上的数据，提升数据库读写性能

②怎么设置缓冲池大小：

innodb\_buffer\_pool\_size  | 134217728  
​  
• set global innodb\_buffer\_pool\_size=134217728

配置文件

物理内存的70%左右

③如何判断缓冲池大小的设置是合理：

如果free buffers为0，则表明缓冲池设置过小  
• show engine innodb status \G;   
Buffer pool size  
Free buffers  
​  
如果value值大于0，就表明缓冲池设置过小，等待空闲页次数  
• show status like '%buffer\_pool\_wait%';

④缓冲池存放哪些数据：数据页，索引页，锁信息，元数据信息，insert buffer，change buffer

⑤如何管理缓冲池：

数据页是存放在LRU list

是通过LRU算法配合midpoint来管理数据页，最频繁使用的页会被存放到yound list最前端，而最少使用的会被存放old list最尾端；

innodb\_old\_blocks\_pct  | 37      
• innodb\_old\_blocks\_time | 1000

⑥空闲页 free list：

数据库刚启动时，这时分配的page都会存放到free list中，lru list 是空的

⑦脏页列表 flush list

脏页：在lru list中的页被修改，脏页；

⑧数据库缓冲池数据预热

innodb\_buffer\_pool\_dump\_at\_shutdown | ON  
​  
• innodb\_buffer\_pool\_dump\_pct        | 25  
​  
• innodb\_buffer\_pool\_load\_at\_startup | ON

将缓冲池中的数据保存到ib\_buffer\_pool文件中

**3.刷写脏页check point**

磁盘：a=3

内存：a=5 （通过读取或者预热加载到 BP 中，被修改3 -> 5，内存中与磁盘数据不一致，则称为 脏页 ）

①作用：当缓冲池产生脏页，就与磁盘上的数据页产生不一致，就需要将内存的**脏页刷写到磁盘**

**可以释放buffer pool空间**

②什么时候触发checkpoint：

。master thread：每一秒或者每十秒刷写脏页

。buffer pool；innodb\_lru\_scan\_depth | 1024；指定 BP 中一定要有多少空闲页，如果低于指定空闲页，将触发checkpoint

。重做日志不可用，将触发checkpoint

。buffer pool；innodb\_max\_dirty\_pages\_pct | 75.000000；脏页在 BP 中占用超过75%，如果超过指定75%，将触发checkpoint

。innodb\_fast\_shutdown | 1 ；如果关闭数据库将buffer pool中所有的脏页全部刷写到磁盘

**4.重做日志-redo log**

LSN：版本号、字节量

show engine innodb status\G  
​  
BP 中版本号、字节量  
Log sequence number  
​  
rodo log 中版本号、字节量  
Log flushed up to  
​  
刷鞋到磁盘上的版本号、字节量  
Pages flushed up to  
​  
上一次 ckpt 检查点  
Last checkpoint at

redo log细节：

①重做日志是物理日志，顺序写入，性能是比较高的

②redo log buffer刷写到redo log的依据：

。master thread，以每秒的频率刷写到redo log

innodb\_flush\_log\_at\_trx\_commit | 1   
1：表示每次事务提交都会将redo log buffer刷写到redo log   双1标准其中一个 1   更安全  
0：表示每次事务提交不会刷写到redo log，而是一秒后再刷写到redo log   不安全 性能高  
2：表示每次事务提交不会刷写到redo log，而是存放到os cache，等一秒后再刷写到redo log   
什么是 os cache ：在磁盘内有一块区域  
​  
超过redo log buffer空间的 50%，就会将redo log buffer刷写到redo log  
redo log buffer 大小设置  
• innodb\_log\_buffer\_size | 33554432  
​  
磁盘上 redo log 大小。在 ssd 顺序写、性能很高，4G  
• innodb\_log\_file\_size      | 134217728  
​  
环境里面设置多少个 redo log 文件，循环写入  
• innodb\_log\_files\_in\_group | 2

③重做日志是innodb存储引擎层产生的，二进制日志是根据数据库的（无论什么存储引擎都会记录到二进制日志里面）

④重做日志由redo log buffer 和 redo log构成

**5.回滚日志-undo log**

①是什么：是逻辑日志，当对记录做了变更操作就会产生undo记录，会保存变更前旧数据信息；

②存在哪里：回滚段，共享表空间，5.7和8.0 版本后可以指定独立回滚的表空间（undo 表空间）

undo 表空间设置：

在线回收 undo log 空间，开启自动清理功能  
• innodb\_undo\_log\_truncate | on  
​  
设置表空间文件个数  
• innodb\_undo\_tablespaces  | 2  
​  
如果超过最大值，空间就会被释放回收  
• innodb\_max\_undo\_log\_size | 1073741824

③作用：用于保证事务的原子性和一致性，mvcc实现

④undo log分类：insert\_undo,update\_undo

insert\_undo 插入事物

update\_undo 修改删除事物，涉及到其他事物读取旧版本数据信息。

**6.DML语句性能提升技术-insert buffer、change buffer**

**MySQL 写操作性能得到了很大的提升**

insert buffer 用来优化插入语句

change buffer 优化 DML 语句

辅助索引在进行插入操作时，由于需要离散的，随机的获取数据页，导致插入性能降低

insert buffer原理：在插入时，不会直接插入到索引页中，而是放到insert buffer中，然后再按照一定的频率进行insert buffer和索引页的合并操作，通常可以将多个插入操作合并到一个操作中

满足条件：非唯一的普通索引

缺点：当insert buffer还未与索引页合并，发生宕机，可能会造成恢复过程比较慢

查看：

show engine innodb status \G;  
seg size 表示当前 insert buffer 的大小（页的方式）  
free list len 表示空闲页  
size 表示已经合并的记录页的数量  
​  
change buffer 参数  
会进行优化所有的操作 update delete insert  
• innodb\_change\_buffering       | all  
​  
change buffer 最大占 BP 的大小  
• innodb\_change\_buffer\_max\_size | 25

**7.数据写入可靠性提升技术-doublewrite**

两次写：doublewrite；解决部分写失效，提高数据写入的可靠性 保护磁盘数据页，避免意外宕机而出现数据页出现部分写（partial write）写坏的情况，是数据库自身进行的维护

两次写原理：在对缓冲池中的脏页进行刷新时，并不是直接写到数据页中，而是先将数据存放到内存中的doublewrite buffer，每次1M大小顺序写入共享表空间，然后再刷写到数据页；

建议主库开启，从库停止  
• innodb\_doublewrite | ON

**8.自适应哈希索引-AHI**

哈希索引是针对经常访问的索引页面按需构建的。

作用：把缓冲池中热点数据，创建哈希索引，提升**等值查询**的性能

INNODB 会监控对表上的二级索引查找，如果发现某些二级索引被频繁访问，二级索引称为热点数据，将频繁进行访问的索引页，创一个内存HASH表，达到快速访问这些索引的目的

**只支持等值查询**

i\_name(name)

where name='ergou';

where name in();

where name != '';

i\_name\_order\_date(name,order\_date)

where name='';

where name='' and order\_date='';

注意：前缀索引是利用不到 AHI

AHI 开启  
• innodb\_adaptive\_flushing         | ON  
​  
• show engine innodb status \G;  
每秒中使用 hash 索引搜索次数，每秒中没有使用 hash 索引搜索次数  
0.00 hash searches/s, 0.00 non-hash searches/s

**9.innodb预读预写技术**

预写：

刷新邻接页；当刷新脏页时，会检测该页所在的区（区包含连续的64个页 16KB \* 64 = 1M）里所有脏页，会一起刷写到磁盘

innodb\_flush\_neighbors | 1

ssd建议关闭，可能会造成多次刷写脏页

预读：

随机预读：在一个区中，如果有13个热点页（ lru list ，都是在 yound list 1/4 的地方），就会读取整个区

innodb\_random\_read\_ahead    | OFF

线性预读：一个区中有连续56个页都被顺序访问，就会读取整个区

innodb\_read\_ahead\_threshold | 56

**10.参数配置文件**

show variables like '%buffer\_pool%'; 查看配置参数

动态参数：在数据库运行期间允许修改

只读参数：在数据库运行期间不允许修改

set global datadir='./tmp'; ERROR 1238 (HY000): Variable 'datadir' is a read only variable

设置参数：

set [global | session] k=v;

参数配置文件修改

**11.错误日志文件**

查看错误日志文件位置：

show variables like 'log\_error';

log\_error | /data/mysql/mysql3306/logs/error.log

设置：参数配置文件中设置log\_error

清空错误日志文件： > error.log

general log 通用查询日志，正常情况下不开启

| general\_log | OFF | | general\_log\_file | /data/mysql/mysql3306/data/db01.log |

**12.慢查询日志文件**

①开启

slow\_query\_log | ON

slow\_query\_log\_file | /data/mysql/mysql3306/logs/slow.log

long\_query\_time | 0.500000

log\_queries\_not\_using\_indexes | ON

min\_examined\_row\_limit | 100

②分析慢查询日志

。mysqldumpslow -s c -t 10 slow.log

。pt-query-digest --user=root --password=123456 --since=12h --socket=/data/mysql/mysql3306/mysql.sock /data/mysql/mysql3306/logs/slow.log

--host mysql服务器地址  
--user mysql用户名  
--password mysql用户密码  
--socket 指定socket文件  
--limit   限制输出结果百分比或数量，默认值是20,即将最慢的20条语句输出，如果是50%则按总响应时间占比从大到小排序，输出到总和达到50%位置截止。  
--since 从什么时间开始分析，值为字符串，可以是指定的某个”yyyy-mm-dd [hh:mm:ss]”格式的时间点，也可以是简单的一个时间值：s(秒)、h(小时)、m(分钟)、d(天)，如12h就表示从12小时前开始统计。  
--until 截止时间，配合—since可以分析一段时间内的慢查询。

pt-query-digest --user=root --password=123456 --since=‘2021-06-025’ --until=‘2021-06-29’ --socket=/data/mysql/mysql3306/mysql.sock /data/mysql/mysql3306/logs/slow.log

③如何清理大的慢日志文件

。mv slow.log slow.log.bak

。flush slow logs;

。rm -rf slow.log.bak

**13.MySQL二进制日志**

①记录什么

。记录已提交DML事务语句,并拆分为多个事件进行保存

begin;

update ...

insert ...

commit;

。记录DDL、DCL语句

总结：记录对mysql数据库执行更改的所有操作；

②开启并配置二进制日志

log\_bin=/data/mysql/mysql3306/logs/mysql-bin #开启并设置二进制日志保存的目录

binlog\_format=row #设置二进制日志如何保存

row：行记录格式，会记录每一行的操作

优点：数据记录严谨，如主从复制，保证主从数据的一致性

缺点：导致二进制日志文件数据量较大，磁盘性能消耗

statement：语句记录格式，会记录每一个sql语句

优点：只记录sql语句，日志量比较小，节省空间，减少磁盘性能消耗

缺点：可能会导致数据不一致now()

mixed：混合记录模式

max\_binlog\_size 设置二进制文件大小，默认最大是1G；超过最大值，就会新生成一个新的二进制日志文件

binlog\_cache\_size 设置binlog cache大小

sync\_binlog设置事务提交或者ddl、dcl执行后怎么保存binlog

1：每次事务提交后就写到二进制日志文件

2：每2次事务提交后才写到二进制日志文件

0：由系统决定什么时候讲缓存中的二进制日志写到二进制日志文件

binlog\_rows\_query\_log\_events | on （在row格式，会记录sql语句）

expire\_logs\_days=7 设置二进制日志保存的天数

binlog\_do\_db 记录某个库的日志

binlog\_ignore\_db 忽略某个库的日志

③作用

。增量备份

。主从复制

④管理二进制日志

show binary logs; 查看当前数据库实例有哪些二进制日志文件

show master status; 查看当前数据库正在使用的二进制日志文件

show binlog events in 'mysql-bin.000091';

mysqlbinlog --base64-output=decode-rows -vv /data/mysql/mysql3306/logs/mysql-bin.000091

mysqlbinlog --start-position=417 --stop-position=653 /data/mysql/mysql3306/logs/mysql-bin.000091 >/tmp/t3.sql 截取二进制日志文件

⑤利用二进制日志文件恢复误删的表

mysqlbinlog --start-position=259 --stop-position=1087 /data/mysql/mysql3306/logs/mysql-bin.000091 >/tmp/t3\_1.sql

⑥刷新二进制文件

。flush logs;

。重启数据库

。超过binlog最大值

⑦删除二进制文件

。expire\_logs\_days=7

。reset master;

。purge binary logs to 'mysql-bin.000088'

**14.InnoDB存储引擎表空间文件**

共享表空间：ibdata1

存什么：数据字典信息、doublewrite、insert buffer、undo

参数：innodb\_data\_file\_path | ibdata1:11M:autoextend

独立表空间：t1.idb

存什么：每个表中的数据、索引

参数：innodb\_file\_per\_table | ON

undo表空间：

临时表空间：5.6+支持

存什么：通用临时表及sql执行过程中产生的临时表、undo

**15.InnoDB存储引擎表空间结构**

段：由若干个区构成，是逻辑概念，由存储引擎自身去进行管理，无需人工干预；

区：由连续的64个页构成，默认是1M大小；8k 128页；4k 256页

页：默认大小是16k；innodb\_page\_size | 16384；i/o最小的操作单位是页；

压缩表：是基于页进行压缩

alter table supplier row\_format=compressed，key\_block\_size=8;

行：行格式innodb\_default\_row\_format | dynamic

compressed 支持压缩

dynamic 优化大字段记录

行溢出页：id info 2m

**16.表碎片清理**

大量删除操作 DELETE 会造成表碎片

DELETE 事务型删除，DML 语句逐行进行删除，但实质上是将被删除的行做了标记，并没有从磁盘上删除，自增主键不会从1开始

①如何判断表是否有碎片

show table status like '%orders%';  
表示表中是否有碎片，但是不太准确  
Data\_free : 6291456  
  
information\_chema.tables 表下面两个字短  
数据大小  
i\_s.tables.data\_length  
索引大小  
i\_s.tables.index\_length  
  
select (data\_length + index\_length)/1024/1024/1024 from tables where table\_schema = 'DB\_NAME' and table\_name = 'TABLE\_NAME'

计算出来的结果和 Linux 目录文件下的 TABLE\_NAME.ibd 大小进行比对，ibd 文件的值 > 计算出来的结果 = 有碎片

②如何清理碎片

alter TABLE\_NAME orders engine=innodb;  
  
​ optimize TABLE\_NAME orders;

pt-osc 推荐

虽然性能方面不会对 DML 造成影响，但是会产生从库复制延迟。

pt-online-schema-change --user=root --password=123456 --socket=/data/mysql/mysql3306/mysql.sock D=dbt3,t=lineitem --alter "engine=innodb" --execute

**17.表空间文件迁移**

案例：将3306实例上dbt3库下的orders迁移到3307实例下的db库

。在3306，flush table orders for export; #加锁阻止写操作

。在3306，进入数据区；cp orders.\* /root

。在3306，unlock tables; 释放锁

。在3307，执行order建表

show create table orders;  
CREATE TABLE `orders` (  
 `o\_orderkey` int(11) NOT NULL,  
 `o\_custkey` int(11) DEFAULT NULL,  
 `o\_orderstatus` char(1) DEFAULT NULL,  
 `o\_totalprice` double DEFAULT NULL,  
 `o\_orderDATE` date DEFAULT NULL,  
 `o\_orderpriority` char(15) DEFAULT NULL,  
 `o\_clerk` char(15) DEFAULT NULL,  
 `o\_shippriority` int(11) DEFAULT NULL,  
 `o\_comment` varchar(79) DEFAULT NULL,  
 PRIMARY KEY (`o\_orderkey`),  
 KEY `i\_o\_custkey` (`o\_custkey`),  
 KEY `i\_key\_date` (`o\_orderkey`,`o\_orderDATE`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

。在3307，执行alter table orders discard tablespace; 删除独立表空间文件

。在3307：拷贝ibd和cfg文件到3307

scp root@192.168.58.51:/root/orders.ibd ./

scp root@192.168.58.51:/root/orders.cfg ./

。更改用户

chown mysql:mysql orders.\*

。在3307，alter table orders import tablespace; 导入表空间的数据

**18.INNODB存储引擎相关问题总结**

1⃣️简单说下 MySQL 体系结构

**连接池**、**管理工具和服务**、**SQL 接口**、**解析器**、**优化器**、**缓存**、**插拔式存储引擎**、**文件系统**组成

2⃣️ INNODB 存储引擎有哪些特点？与 MyISAM 有哪些区别？

支持多缓冲区池、AHI 、聚集索引、MVCC 、自动故障恢复 CSR 、热备、行锁、事物、外键

3⃣️Buffer Pool 如何设置合适 BP 大小、如何判断 BP 设置是否合理、存放了哪些数据、LRU 算法、预热？

4⃣️WAL日志优先写

5⃣️一条语句的执行流程+故障恢复过程

6⃣️Cache和Buffer是两个不同的概念，简单的说，Cache是加速“读”，而 buffer是缓冲“写”，前者解决读的问题，保存从磁盘上读出的数据，后者是解决写的问题，保存即将要写入到磁盘上的数据。在很多情况下，这两个名词并没有严格区分，常常把读写混合类型称为buffer cache。