# 背景

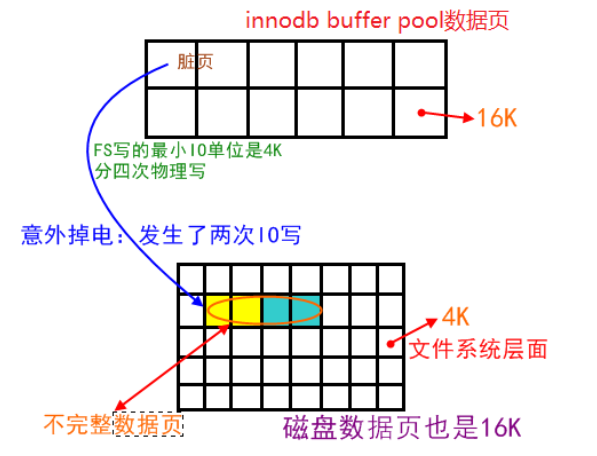
关于IO的最小单位：

1、数据库IO的最小单位是16K（MySQL默认，oracle是8K）

2、文件系统IO的最小单位是4K（也有1K的）

3、磁盘IO的最小单位是512字节

因此，存在IO写入导致page损坏的风险：



一个数据页的大小是16K，假设在把内存中的脏页写到数据库的时候，写了2K突然掉电，也就是说前2K数据是新的，后14K是旧的，那么磁盘数据库这个数据页就是不完整的，是一个坏掉的数据页。即InnoDB的page刷到磁盘上要写4个操作系统block，**在极端情况下(比如断电)不一定能保证4个块的写入原子性**。redo只能加上旧、校检完整的数据页恢复一个脏块，不能修复坏掉的数据页，所以这个数据就丢失了，可能会造成数据不一致，所以需要double write。

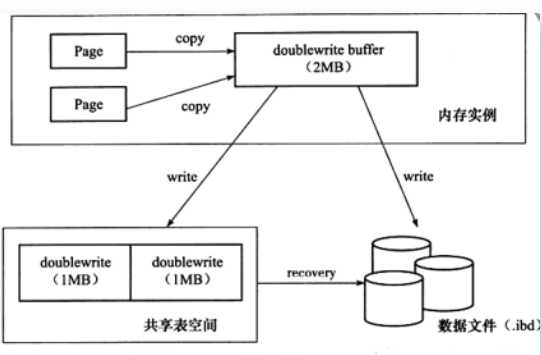
**说明：**

InnoDB的redo日志格式它是逻辑的(即：一个redo记录内容只包含指定的space\_id，page\_no，page\_offset和数据内容，在真正应用的时候才会将redo内容转换为对应的数据页记录),所以在执行redo时候如果那个页面本身就是break page，那么使用redo恢复的数据肯定也是错误的。

所以InnoDB为了保证page是正确的，使用了double write功能，保证page是完整的。

当数据库正在从内存想磁盘写一个数据页时，数据库宕机，从而导致这个页只写了部分数据，这就是部分写失效（partial write），它会导致数据丢失。这时是无法通过重做日志恢复的，因为重做日志记录的是对页的物理修改，如果页本身已经损坏，重做日志也无能为力。

Doublewrite（两次写）提高innodb的可靠性，用来解决部分写失败(partial page write页断裂)。



doublewrite由两部分组成:内存中的doublewrite buffer和磁盘上的共享表空间中的连续页。

在对缓冲池的脏页进行刷新时，并不直接写入磁盘，而是通过memcpy函数将脏页先复制到内存中的doublewrite buffer，之后通过doublewrite buffer分两次顺序的写入共享表空间的物理磁盘，然后马上调用fsync函数，同步磁盘，避免缓冲写带来的问题。在完成doublewrite页的写入后，再将doublewrite buffer中的页写入各个表空间文件中。是否开启doublewrite还需要看具体情况。

# 概述

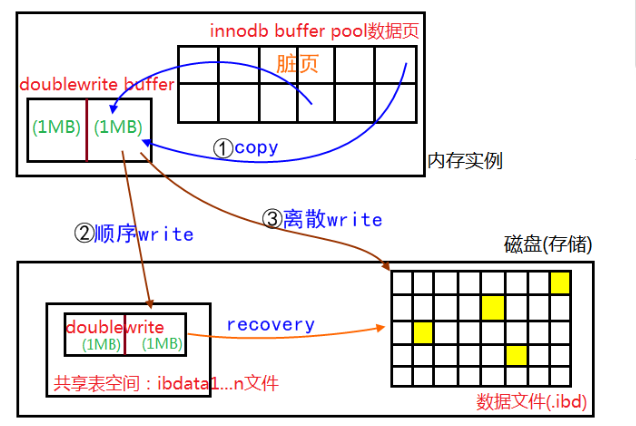
InnoDB使用了一种叫做doublewrite的特殊文件flush技术，在把pages写到date files之前，InnoDB先把它们写到一个叫doublewrite buffer的连续区域内，在写doublewrite buffer完成后，InnoDB才会把pages写到data file的适当的位置。**如果在写page的过程中发生意外崩溃，InnoDB在稍后的恢复过程中在doublewrite buffer中找到完好的page副本用于恢复**。

doublewrite buffer是InnoDB在table space上的128个页（2个区）大小是2MB。为了解决 partial page write问题，**当MySQL将脏数据flush到data file的时候, 先使用memcopy将脏数据复制到内存中的doublewrite buffer，之后通过doublewrite buffer再分2次，每次写入1MB到共享表空间，然后马上调用fsync函数，同步到磁盘上，避免缓冲带来的问题**，在这个过程中，doublewrite是顺序写，开销并不大，在完成doublewrite写入后，再将double write buffer写入各表空间文件，这时是离散写入。

所以在正常的情况下, MySQL写数据page时，会写两遍到磁盘上，第一遍是写到doublewrite buffer，第二遍是从doublewrite buffer写到真正的数据文件中。如果发生了极端情况（断电），InnoDB再次启动后，发现了一个page数据已经损坏，那么此时就可以从doublewrite buffer中进行数据恢复了。

# 原理

## doublewrite



doublewrite由两部分组成，一部分为内存中的doublewrite buffer，其大小为2MB，另一部分是磁盘上共享表空间(ibdata x)中连续的128个页，即2个区(extent)，大小也是2M。

1、当一系列机制触发数据缓冲池中的脏页刷新时，并不直接写入磁盘数据文件中，而是先拷贝至内存中的doublewrite buffer中；

2、接着从两次写缓冲区分两次写入磁盘共享表空间中(连续存储，顺序写，性能很高)，每次写1MB；

3、待第二步完成后，再将doublewrite buffer中的脏页数据写入实际的各个表空间文件(离散写)；(脏页数据固化后，即进行标记对应doublewrite数据可覆盖)

## 崩溃恢复

如果操作系统在将页写入磁盘的过程中发生崩溃，在恢复过程中，innodb存储引擎可以从共享表空间的doublewrite中找到该页的一个最近的副本，将其复制到表空间文件，再应用redo log，就完成了恢复过程。

因为有副本所以也不担心表空间中数据页是否损坏。

**Q：为什么log write不需要doublewrite的支持？**

A：因为redolog写入的单位就是512字节，也就是磁盘IO的最小单位，所以无所谓数据损坏。

# 副作用

1、double write带来的写负载

a. double write是一个buffer, 但其实它是开在物理文件上的一个buffer, 其实也就是file, 所以它会导致系统有更多的fsync操作, 而硬盘的fsync性能是很慢的, 所以它会降低mysql的整体性能。

b. 但是，doublewrite buffer写入磁盘共享表空间这个过程是连续存储，是顺序写，性能非常高，(约占写的%10)，牺牲一点写性能来保证数据页的完整还是很有必要的。

2、监控double write工作负载

mysql> show global status like '%dblwr%';

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

| Variable\_name | Value |

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

| Innodb\_dblwr\_pages\_written | 7 |

| Innodb\_dblwr\_writes | 3 |

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

2 rows in set (0.00 sec)

　　关注点：Innodb\_dblwr\_pages\_written / Innodb\_dblwr\_writes

　　开启doublewrite后，每次脏页刷新必须要先写doublewrite，而doublewrite存在于磁盘上的是两个连续的区，每个区由连续的页组成，一般情况下一个区最多有64个页，所以一次IO写入应该可以最多写64个页。

　　而根据以上系统Innodb\_dblwr\_pages\_written与Innodb\_dblwr\_writes的比例来看，大概在3左右，远远还没到64(如果约等于64，那么说明系统的写压力非常大，有大量的脏页要往磁盘上写)，所以从这个角度也可以看出，系统写入压力并不高。

3、关闭double write适合的场景

　　a. 海量DML

b. 不惧怕数据损坏和丢失

c. 系统写负载成为主要负载

mysql> show variables like '%double%';

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

| Variable\_name | Value |

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

| innodb\_doublewrite | ON |

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

1 row in set (0.04 sec)

　　作为InnoDB的一个关键特性，doublewrite功能默认是开启的，但是在上述特殊的一些场景也可以视情况关闭，来提高数据库写性能。静态参数，配置文件修改，重启数据库。

4、为什么没有把double write里面的数据写到data page里面呢？

a. double write里面的数据是连续的，如果直接写到data page里面，而data page的页又是离散的，写入会很慢。

b. double write里面的数据没有办法被及时的覆盖掉，导致double write的压力很大；短时间内可能会出现double write溢出的情况。

# 使用

在一些情况下可以关闭doublewrite以获取更高的性能。比如在slave上可以关闭，因为即使出现了partial page write问题，数据还是可以从中继日志中恢复。设置InnoDB\_doublewrite=0即可关闭doublewrite buffer。