

MySQL Buffer Pool的“防暴”机制，让你的数据库内存永不“社恐”

原创 DB哥 DB哥 2025年07月23日 07:01 安徽

关注DB哥微信公众号「DB哥」**免费学**DBA级MySQL视频教程【149课时】



DB哥
10年数据库救火队老炮 | 用实战教你少熬三年夜。遇到数据库别慌，DB哥专治数据库各种...
206篇原创内容

公众号

请加DB哥个人微信：dbelder 邀请加入MySQL高级教程学习答疑群

在DB哥学习答疑群你可以享受到下面的福利：

- MySQL高级课程问题答疑，提问时请提供问题相关信息；
- DB哥每周1个职场MySQL生产数据库问题解决名额，仅工作时间；
- DB哥不定期分享生产数据库架构、优化、安全案例；
- 提供DB哥每篇技术文章最后遗留问题的答案；

各位看官，今天咱不聊风花雪月，聊点数据库界的“惊悚片”——全表扫描。

想象一下这个场景：你精心策划了一场高端相亲大会（Buffer Pool），邀请的都是经过层层筛选、炙手可热的精英数据（热点数据页）。大家眉来眼去，内存命中率蹭蹭上涨，查询速度那叫一个丝滑流畅，宛如德芙巧克力。

突然！后台大门被“哐当”一声撞开！一个练习2年半的程序员小张，手舞足蹈地冲进来，高喊：“给我查！查最近一年所有用户的登录记录，不管3721，我全都要！”——好家伙，这就是传说中的 `SELECT * FROM dbbro_user_logs;` 全表扫描！

刹那间，成千上万、平时无人问津的“用户登录记录”数据页（很多可能是N年陈酿的老数据），像潮水般涌入会场！它们热情洋溢，横冲直撞，瞬间就把那些正在优雅交谈的“精英数据”（比如高频访问的用户信息、商品详情）挤得东倒西歪，甚至直接被挤出会场（淘汰出Buffer Pool）！

内存命中率暴跌！查询性能卡成PPT！数据库服务器CPU风扇开始哀嚎！

一场本应甜蜜高效的“相亲会”，瞬间变成了混乱不堪的“春运火车站”。这就是传说中的“全表扫描污染Buffer Pool”惨案！Buffer Pool这个“社恐”的内存空间，最怕的就是这种不讲武德的“人海战术”。

那么问题来了：MySQL，这位数据库界的“老江湖”，是如何巧妙布局，在相亲大会门口设置“安检隔离区”，让这些“扫表大军”有序进场，避免它们把真正重要的“嘉宾”踩在脚下，从而保护我们宝贵的内存命中率和查询性能的呢？

答案，就藏在 Buffer Pool的核心算法——改进版LRU链表和神奇的midpoint机制 之中！且听我慢慢道来，包你笑出腹肌，哦不，是学出真知！

一：Buffer Pool——数据库的“相亲大会”会场

- 核心概念：** Buffer Pool，简称BP，是InnoDB存储引擎在内存中开辟的一块至关重要的区域。它的核心使命就是 缓存磁盘上的数据页（Data Page）和索引页（Index Page）。想象成就是我们的“相亲大会”会场。数据页就是来相亲的“嘉宾”。
- 为什么重要？** 内存访问速度比磁盘快N个数量级！如果“嘉宾”（数据页）就在会场（BP）里，数据库引擎（媒婆）就能瞬间找到它（高命中率），安排“相亲”（查询）效率贼高（高性能）。如果不在，就得苦哈哈地去磁盘仓库翻找（磁盘I/O），慢得让人心碎（低性能）。
- 内存命中率：** 这是衡量BP效率的核心KPI！。命中率越高，说明会场里的“嘉宾”越受欢迎（被访问得多），查询速度越快。我们的终极目标就是让这个命中率无限接近100%！全表扫描这种“人海战术”最怕的就是它瞬间拉低这个KPI。

二：传统LRU算法——简单的“按资排辈”与它的致命缺陷

- LRU是啥？** Least Recently Used（最近最少使用）。这是计算机科学里最经典的缓存淘汰算法之一。
- 工作原理（相亲大会版）：** 想象一个长长的嘉宾队列（链表）。新来的嘉宾（新数据页）插在队伍最前面（链表头），这里是“C位”，聚光灯下，最容易被媒婆（引擎）看到。每次有嘉宾被成功“相中”（数据页被访问），ta就会被请到队伍最前面（移到链表头），重新成为焦点。当会场坐满了（BP满了），要请新嘉宾进来时，就淘汰掉队伍最后面（链表尾）那个最久没人搭理的“小透明”（最近最少使用的数据页）。
- 缺陷暴露（全表扫描的破坏力）：**
 - “人海”淹没“精英”：** 全表扫描会瞬间加载大量数据页。按照传统LRU，这些新页会疯狂插队到链表头。好家伙！一下子就把原本在头部的真正的热点数据（精英嘉宾）给挤到了后面甚至淘汰出局！

2. “昙花一现”的过客： 全表扫描加载的这些数据页，大部分在扫描完这次查询后，可能就再也不会被访问了！它们是典型的“一次性过客”。但它们却霸占了宝贵的C位，把真正的“常青树”嘉宾挤走了。等扫描结束，会场里塞满了一堆“过气网红”，真正的精英却流落在外。后续查询性能？可想而知，惨不忍睹！内存命中率？断崖式下跌！
3. 污染（Pollution）： 这种现象就被形象地称为 Buffer Pool污染。宝贵的缓存空间被这些“低价值”的、只访问一次（或几次）的数据页污染了，无法有效服务于真正的热点数据。

三：InnoDB的智慧——引入“隔离观察区”(Midpoint Insertion)

MySQL的InnoDB引擎工程师们（相亲大会组委会）一看这不行啊！必须改革！于是，他们祭出了大杀器：改进版LRU链表 + Midpoint插入策略。核心思想：对“扫表大军”实行“隔离观察”政策！

- 分区管理：5:3的“阶级”划分
 - Young Region (新生代/热区 – 占5/8)： 这里是真正的“VIP核心区”！住在这里的都是 高频被访问、炙手可热的精英数据页。媒婆（引擎）优先在这里找对象（数据）。这里是保证高命中率的关键战场。
 - Old Region (老生代/冷区 – 占3/8)： 这里是“新人隔离观察区”或者说“潜在股待定区”。所有新来的嘉宾（新数据页），不管你是何方神圣（包括全表扫进来的），统统先给我进这个区待着！ 想进VIP区？得经过考核！
 - 整个嘉宾队列（LRU链表）被 严格地按照5:3的比例分割 成两个区域：
 - 关键参数： 这个分区比例是固定的（5:3），由MySQL内部机制管理，通常不需要用户调整。它体现了设计者的权衡：给新人（新数据/冷数据）足够的机会（3/8空间），同时确保核心热区（5/8空间）不被轻易污染。
 - 新规一：入口设在“隔离区”（Midpoint Insertion）
 - 新嘉宾（新数据页）加入会场（BP）时，不再享有直接插队C位的特权！而是被 礼貌地（或者说强制性地）引导到 Young区 和 Old区 的交界处——我们称之为 `LRU_old` 或 `midpoint`。这个点，就是Old区的头部。
 - 效果： 全表扫描进来的海量“嘉宾”，不再是直接冲击VIP区，而是被有序地安排在了“隔离观察区”（Old区）的入口处排队。VIP区（Young区）的精英们暂时安全了！
 - 新规二：“隔离观察期”与晋升机制（`innodb_old_blocks_time`）
 - 这个参数定义了“隔离观察期”的长度（单位：毫秒）。默认值是1000毫秒（即1秒）。这个值非常关键！
 - 光隔离还不够。万一隔离区里真藏着一个未来的超级巨星（被频繁访问的数据页）呢？或者，如何识别出那些“扫表大军”里的“一次性过客”？
 - 核心机制： InnoDB给Old区的“观察生”们设定了一个 关键的“考察期”。
 - 关键参数： `innodb_old_blocks_time`
 - 晋升规则（相亲大会考核标准）：
 - 淘汰规则： 当会场满了（BP满了），需要淘汰嘉宾时，优先淘汰Old区尾部（最久未被访问且未达到晋升条件的）的“观察生”。Young区尾部的“过气VIP”也可能被淘汰，但优先级低于Old区的尾部。
1. 当媒婆（引擎）想找某个在“隔离观察区”（Old区）的嘉宾（数据页）聊天（访问它）时，系统会立刻检查这位嘉宾 在观察区已经待了多久（Page第一次被放入Old区的时间戳 vs 当前时间）。
2. “待够时间” (> `innodb_old_blocks_time`)： 如果这位嘉宾在Old区 待的时间超过了设定的考察期（默认1秒），那么恭喜ta！说明ta很可能不是“一次性过客”（因为如果是扫描，通常在极短时间内连续访问完就抛弃了），而是有潜力成为VIP的！ta会被 立刻提拔（移动到Young区的头部），进入核心圈！
3. “待不够时间” (< `innodb_old_blocks_time`)： 如果这位嘉宾在Old区 待的时间还没到考察期（比如，刚进来就被访问了），那么ta 原地不动！ 继续在Old区待着。为什么？ 这很可能就是全表扫描的特征！扫描是顺序的、快速的访问大量数据页。一个数据页刚被加载进Old区（midpoint），紧接着就被扫描的SQL访问到了，时间间隔极短（远小于1秒）。这种“快餐式访问”不被认为是“有潜力”，不值得晋升VIP区，以免挤走真正的精英。它就在Old区待着，等着后续如果真被频繁访问（超过考察期后）再晋升，或者等着被淘汰。

四：实战演练——看改进LRU如何智斗“扫表大军”

场景再现： 程序员小张再次手滑，执行了那个可怕的 `SELECT * FROM dbbro_user_logs WHERE login_time ;`

改进LRU下的“防暴”流程：

1. “大军压境”： 海量的 `dbbro_user_logs` 表的数据页被从磁盘加载。
2. “隔离安置”： 每一个新加载的数据页，都被 严格安置在LRU链表的 `midpoint` 位置，即Old区的头部。VIP区（Young区）风平浪静，精英数据们依然在高效地进行着它们的“相亲活动”（处理其他查询）。
3. “闪电访问”： 全表扫描是顺序访问。当扫描线程访问到刚刚加载到Old区头部的某个数据页时（间隔极短，远小于1秒）。

- “考核瞬间”：InnoDB引擎检查：该页在Old区才待了几毫秒？远小于innodb_old_blocks_time (1000ms)！
- “原地不动”：引擎判定：此乃典型的“扫表特征”！访问无效（对晋升而言）！该数据页保持原位不动（仍在Old区头部）。它不会去冲击Young区。
- “快速掠过”：扫描继续，访问下一个数据页（也是刚加载到Midpoint的），同样因为访问间隔太短，被判定为扫描，原地不动。
- “潮水退去”：全表扫描结束。
- “隔离区清理”：此时，Old区头部塞满了刚刚扫描进来的这些dbbro_user_logs数据页。关键来了：因为它们只在扫描时被短暂访问过一次（且访问发生在“考察期”内），它们没有被晋升到Young区！它们就像一群在隔离区短暂停留后就匆匆离开的访客。
- “精英无损”：Young区（VIP区）的热点数据页（比如dbbro_user_info, dbbro_product的核心页）几乎没有被影响！它们依然占据着C位，高效地服务着后续的常规查询。
- “自然淘汰”：当后续有新的查询需要加载其他数据页时，或者BP需要腾空间，优先淘汰的就是Old区尾部那些“一次性”的dbbro_user_logs扫描页。因为它们既不是热点（只访问了一次），又没资格晋升VIP。
- “命中率坚挺”：内存命中率只是短暂地因为加载磁盘页而略有波动，但核心热点数据依然在缓存中，整体查询性能影响被降到最低！避免了灾难性的暴跌。

效果总结：改进的LRU + Midpoint + innodb_old_blocks_time这套组合拳，相当于给全表扫描这种“人海战术”设置了一个缓冲区（Old区）和一个冷静期（innodb_old_blocks_time）。有效隔离了“扫表大军”对核心热区的冲击，保护了宝贵的内存命中率和查询性能。Buffer Pool这个“相亲会场”从此告别“社恐”，变得从容不迫！

五：技术细节与最佳实践

- 监控你的Buffer Pool：
 - youngs/s：每秒从Old区晋升到Young区的页数。正常业务应有一定值。全表扫描期间可能为0。
 - old-young/s：每秒在Old区被访问但未晋升（即在考察期内访问）的页数。全表扫描期间这个值会暴增！
 - Pages made young / not young：累计值。对比可以看晋升比例。
 - buffer pool hit rate：最直接的命中率指标！守护它！
 - SHOW ENGINE INNODB STATUS\G：查看BUFFER POOL AND MEMORY部分。关注：
 - information_schema.innodb_metrics / performance_schema：提供更细粒度的监控指标。
- 理解innodb_old_blocks_time的调整（谨慎！）：
 - 默认1000ms (1秒) 是一个经过验证的、对绝大多数场景合理的值。它能有效过滤掉大部分全表扫描和类似的大规模顺序访问。
 - 何时调大？如果你的业务有非常大量且访问模式非常随机的数据加载，并且你确信这些新加载的数据有很大比例会很快（超过1秒后）成为热点，那么可以适当调大此参数（比如2000ms），让更多新页有晋升机会。但这会增加扫描污染的风险，需极其谨慎并密切监控命中率！
 - 何时调小？如果你的扫描操作本身非常慢（比如扫描一个超级大表），扫描过程中访问一个Old区页的时间间隔很容易超过1秒，导致大量扫描页被错误晋升，反而污染Young区。此时可以适当调小此参数（比如500ms），让它们更难晋升。但这可能让一些真正的、访问间隔略短于1秒的潜力新星失去晋升机会。同样需谨慎！
 - 黄金法则：99%的情况下，保持默认值1000ms是最佳选择！不要轻易动它，除非你有非常明确的性能问题、详实的监控数据支撑，并理解调整后的所有后果。


```
1  -- 符合规范的包含'dbbro'的表名示例 (MySQL 8.0)
2  CREATE TABLE dbbro_user_logs (
3      id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT '唯一ID',
4      user_id INT UNSIGNED NOT NULL COMMENT '用户ID',
5      login_time DATETIME(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6) COMMENT '登录时间',
6      ip_address VARCHAR(45) COMMENT 'IPv4或IPv6地址',
7      device_info VARCHAR(255) COMMENT '设备信息',
8      INDEX idx_login_time (login_time) COMMENT '加速按时间范围查询' -- 关键索引
9  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci COMMENT='用户登录日志表'
10 -- 模拟全表扫描的查询 (虽然我们有索引，但假设条件导致没走成或者就是SELECT *)
11 EXPLAIN SELECT * FROM dbbro_user_logs WHERE YEAR(login_time) = YEAR(NOW());
12 -- 检查Buffer Pool状态 (部分关键信息)
13 SHOW ENGINE INNODB STATUS\G
14 -- 关注输出中的 'BUFFER POOL AND MEMORY' section
```

谢谢大家的关注、点赞、分享！
如有疑问，可以留言，DB哥看到后会及时回复，也可以加DB哥微信交流

END

-- 加入「DB哥数据库帮」

DB哥微信：dbelder

 DB哥数据库帮专属福利
▸ 授人以渔

关注DB哥微信公众号「DB哥」**免费学**DBA级MySQL视频课程【149课时】





DB哥
10年数据库救火队老炮 | 用实战教你少熬三年夜。遇到数据库别慌，DB哥专治数据库各种...
206篇原创内容


公众号


▸ 技术辅助

- 1

 10年数据库救火队老炮 | 用实战教你少熬三年夜
- 2

 亲手调优3000+故障库 | 企业级数据库架构
- 3

 库崩了？锁死了？SQL慢如🐢？CPU100%
- 4

 别慌，DB哥专治数据库各种“不调”！

▸ 背锅侠租赁

临时工小张随时待命：

- 1

`UPDATE salary SET bonus =0;-- 小张干的！`

帮规：

- 1

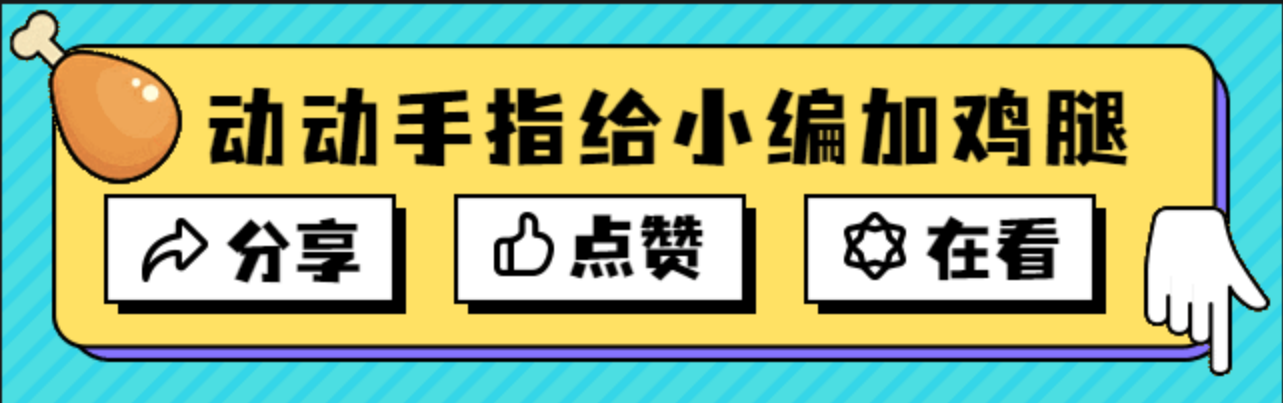
1.不准在生产环境执行UPDATE不带WHERE，否则罚用触控板代替鼠标一周
- 2

2.删库后不跑路，否则罚用Windows XP装 MySQL5.0（不兼容也要装）
- 3

3.必须用 JOIN 代替子查询，否则罚直播用子查询实现复杂报表（不许用JOIN！）
- 4

4.生产环境执行DDL必须测试，否则罚胸口碎大石（罪名：惊动监控告警）
- 5

5.不用SELECT * 横扫全表，否则罚罚抄《索引优化十诫》100遍（用毛笔写SQL语句）



DB哥

 喜欢作者

DB哥讲数据库 · 目录

< 上一篇

别傻了！用SELECT 1检测MySQL健康？数据库的“诈尸”大戏正在上演！

下一篇 >

我的MySQL在USE库时卡成了PPT？真相竟是...（不是网速的锅！）

