

InnoDB 和线程

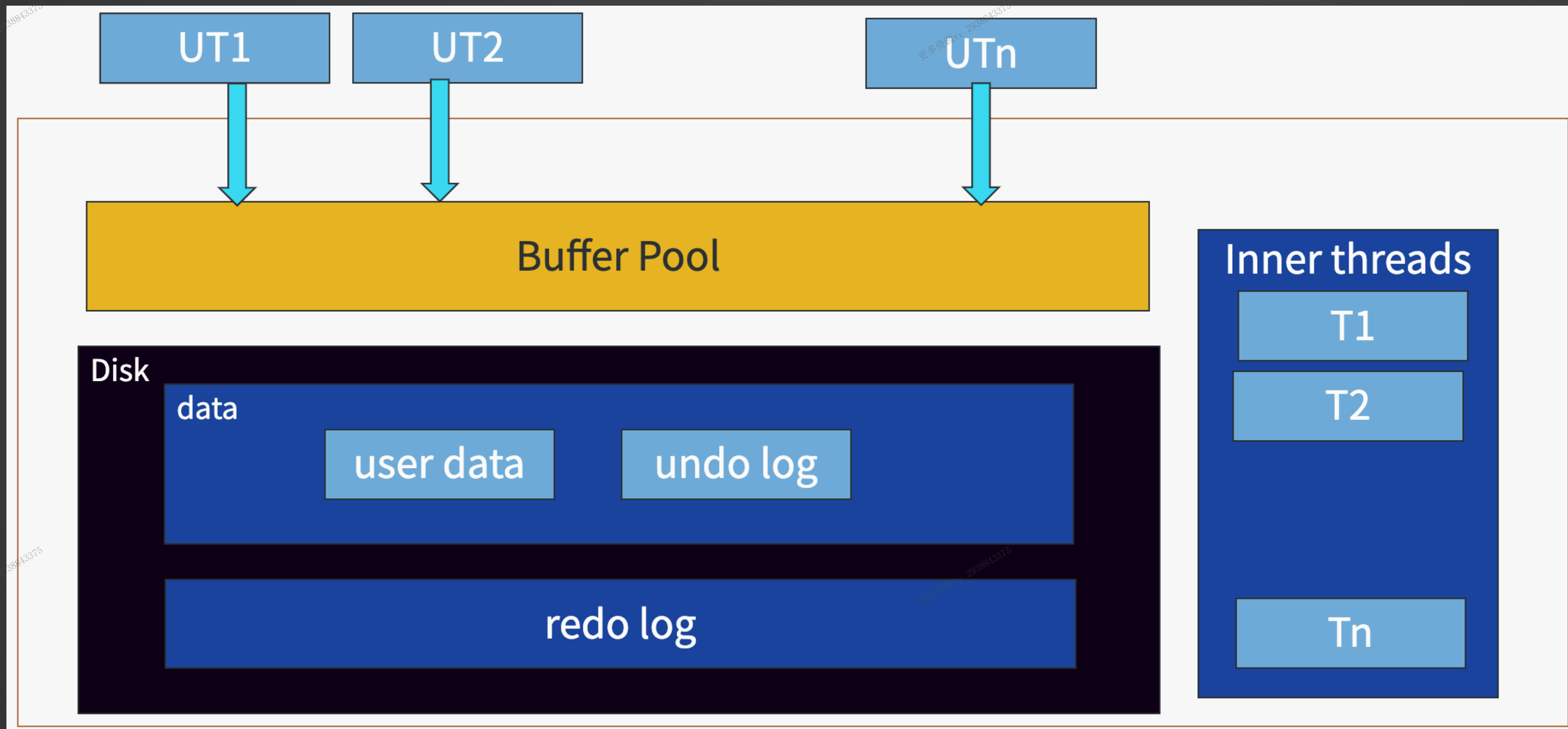
目录

1 MySQL 分层模型

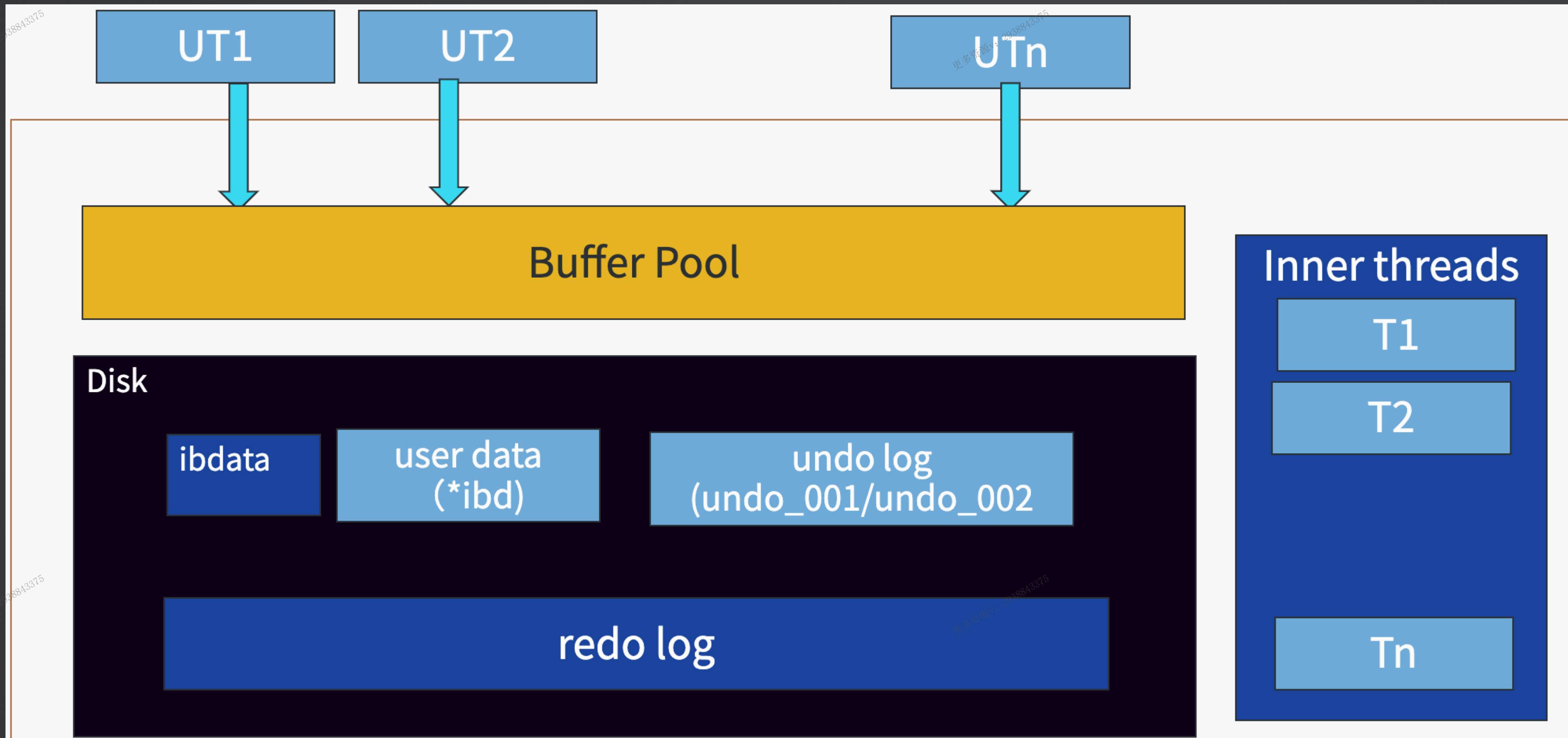
2 Server 层线程

3 InnoDB 线程

基础知识回顾



基础知识回顾



基础知识回顾

```
mysql> select thread_id,name from performance_schema.threads where name like '%innodb%' order by name;
```

thread_id	name
33	thread/innodb/buf_dump_thread
26	thread/innodb/buf_resize_thread
34	thread/innodb/clone_gtid_thread
28	thread/innodb/dict_stats_thread
29	thread/innodb/fts_optimize_thread
3	thread/innodb/io_ibuf_thread
4	thread/innodb/io_read_thread
7	thread/innodb/io_read_thread
6	thread/innodb/io_read_thread
5	thread/innodb/io_read_thread
10	thread/innodb/io_write_thread
11	thread/innodb/io_write_thread
9	thread/innodb/io_write_thread
8	thread/innodb/io_write_thread
13	thread/innodb/log_checkpoint_thread
18	thread/innodb/log_files_governor_thread
14	thread/innodb/log_flush_notifier_thread
15	thread/innodb/log_flusher_thread
16	thread/innodb/log_write_notifier_thread
17	thread/innodb/log_writer_thread
12	thread/innodb/page_flush_coordinator_thread
24	thread/innodb/srv_error_monitor_thread
23	thread/innodb/srv_lock_timeout_thread
27	thread/innodb/srv_master_thread
25	thread/innodb/srv_monitor_thread
35	thread/innodb/srv_purge_thread
36	thread/innodb/srv_worker_thread
37	thread/innodb/srv_worker_thread
38	thread/innodb/srv_worker_thread

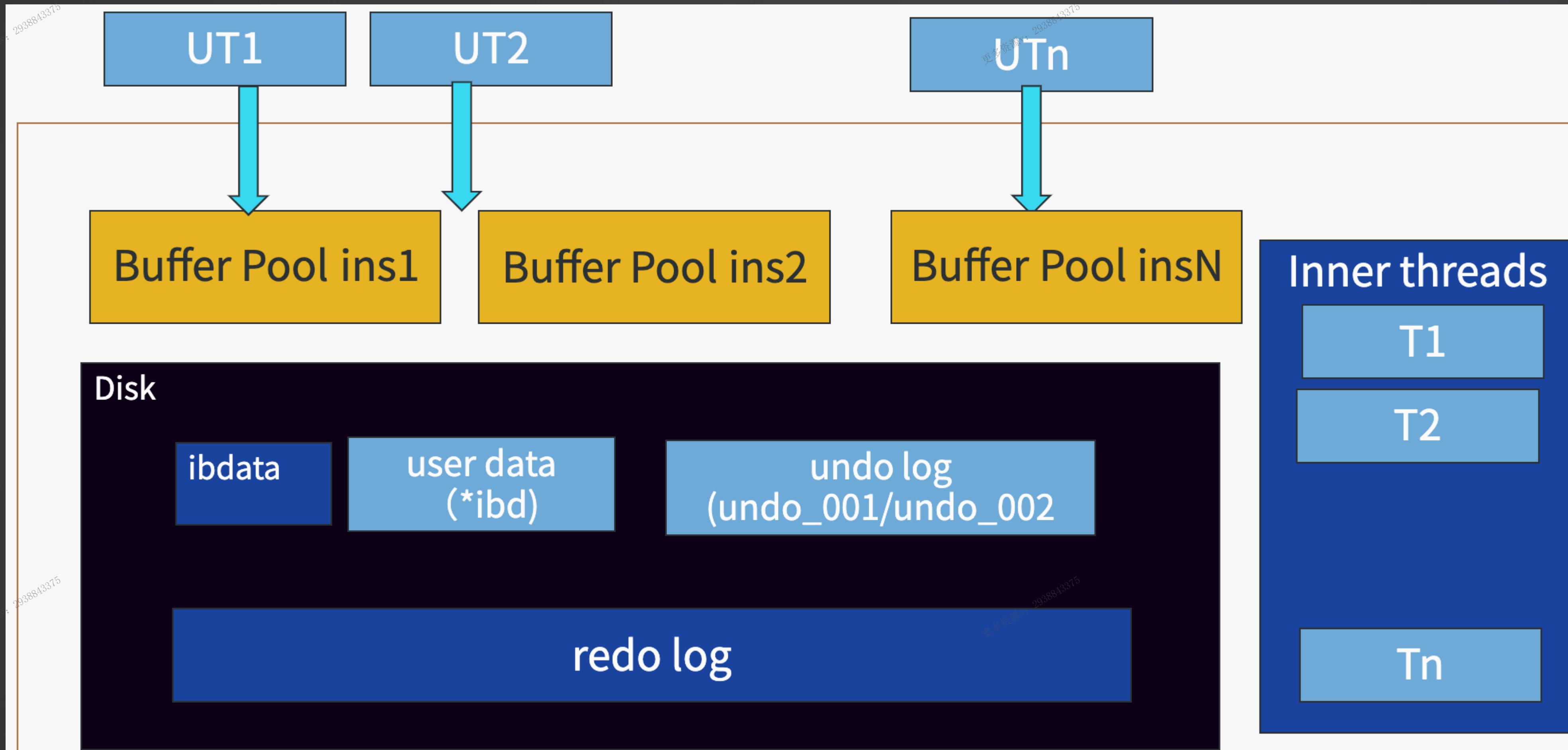
29 rows in set (0.00 sec)

buf_dump_thread

```
mysql> set global innodb_buffer_pool_dump_now=on;
```

```
Query OK, 0 rows affected (0.00 sec)
```

buf_dump_thread



buf_dump_thread:innodb_buffer_pool_dump_pct

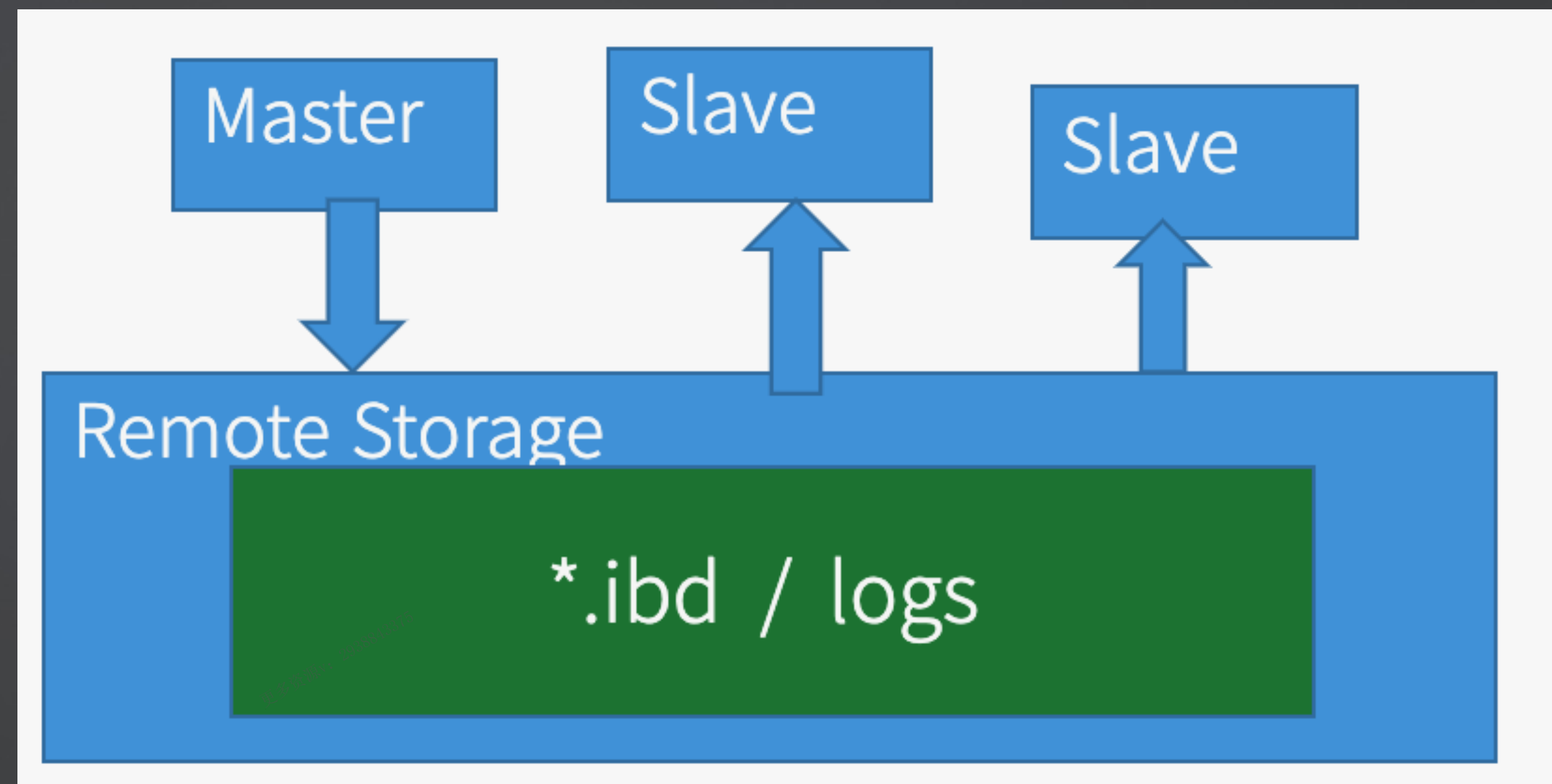
4294967279, 109
4294967279, 227
4294967279, 226
4294967279, 225
4294967279, 223
4294967279, 106
4294967279, 221
4294967279, 220
4294967279, 219
4294967279, 218
4294967279, 276
4294967279, 215

```
/* walk through each buffer pool */  
for (i = 0; i < srv_buf_pool_instances && !SHOULD_QUIT(); i++) {  
    buf_pool_t *buf_pool;
```

```
    for (auto bpage : buf_pool->LRU) {  
        if (n_pages <= j) break;  
        ut_a(buf_page_in_file(bpage));  
  
        dump[j++] = BUF_DUMP_CREATE(bpage->id.space(), bpage->id.page_no());  
    }  
    ut_a(j == n_pages);
```


buf_dump_thread:innodb_buffer_pool_dump_pct

课堂练习：半同步复制的主备架构，主库的 buf_dump 拿到读库去 load，有作用吗？



buf_dump_thread:innodb_buffer_pool_dump_pct

课堂练习：在 buff load 的时候，先执行了 `std::sort(dump, dump + dump_n);` 目的是什么？

```
4294967279, 109  
4294967279, 227  
4294967279, 226  
4294967279, 225  
4294967279, 223  
4294967279, 106  
4294967279, 221  
4294967279, 220  
4294967279, 219  
4294967279, 218  
4294967279, 276  
4294967279, 215
```

课堂练习：动态修改buffer_pool (base 8.0.39)

session1	session2
	begin; select * 大表 ; (假设表数据>bp 当前大小
set global buffer_pool_size = 2* 当前值 A. 无法执行，等待 B. 执行 resize 操作，成功后语句返回 C. 语句返回执行成功，但没有做 resize 工作 D. 语句返回执行成功，并后台启动 resize 工作	

课堂练习：动态修改 buffer_pool (base 8.0.39)

官方文档: Active transactions and operations performed through InnoDB APIs should be completed before resizing the buffer pool

```
2024-08-31T10:10:34.139604Z 0 [Note] [MY-011887] [InnoDB] Will retry to withdraw 1 seconds later.
2024-08-31T10:10:35.139702Z 0 [Note] [MY-011881] [InnoDB] buffer pool 0 : start to withdraw the last 64 blocks.
2024-08-31T10:10:35.139793Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.139815Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.139839Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.139855Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.139874Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.139890Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.139909Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.139931Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.139951Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.139962Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.139980Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.139995Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.140014Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.140025Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.140064Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.140077Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.140105Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.140114Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.140132Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (0/64)
2024-08-31T10:10:35.140142Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 0 pages (0/64).
2024-08-31T10:10:35.140151Z 0 [Note] [MY-011883] [InnoDB] buffer pool 0 : will retry to withdraw later.
2024-08-31T10:10:35.140162Z 0 [Note] [MY-011887] [InnoDB] Will retry to withdraw 2 seconds later.
2024-08-31T10:10:37.140261Z 0 [Note] [MY-011881] [InnoDB] buffer pool 0 : start to withdraw the last 64 blocks.
2024-08-31T10:10:37.141479Z 0 [Note] [MY-013954] [InnoDB] Status code 3: buffer pool 0 : withdrawing blocks. (64/64)
2024-08-31T10:10:37.141517Z 0 [Note] [MY-011882] [InnoDB] buffer pool 0 : withdrew 0 blocks from free list. Tried to relocate 64 pages (64/64).
2024-08-31T10:10:37.141533Z 0 [Note] [MY-011884] [InnoDB] buffer pool 0 : withdrawn target 64 blocks.
2024-08-31T10:10:37.141561Z 0 [Note] [MY-013953] [InnoDB] Status code 3: 100% complete
2024-08-31T10:10:37.141574Z 0 [Note] [MY-013952] [InnoDB] Status code 3: Completed
2024-08-31T10:10:37.141586Z 0 [Note] [MY-013954] [InnoDB] Status code 4: Latching whole of buffer pool.
2024-08-31T10:10:37.141601Z 0 [Note] [MY-013953] [InnoDB] Status code 4: 14% complete
```

课堂练习：动态修改 buffer_pool (base 8.0.39)

官方文档： Once the resizing operation is in progress, new transactions and operations that require access to the buffer pool must wait until the resizing operation finishes

改大 or 改小

ib_io_read

1. 线性预读

innodb_read_ahead_threshold

2. 随机预读

innodb_random_read_ahead

$5 + 64 / 8$

课堂练习：假设表 T 结构：

```
create table t(id int, c int , d varchar(255),e int, primary key(id), index(c));
```

以下哪种情况，会最快触发 ib_io_read 工作？

A. select count(*) from t;

B. select * from t;

C. alter table t add index (e)

3. buf_read_ibuf_merge_pages

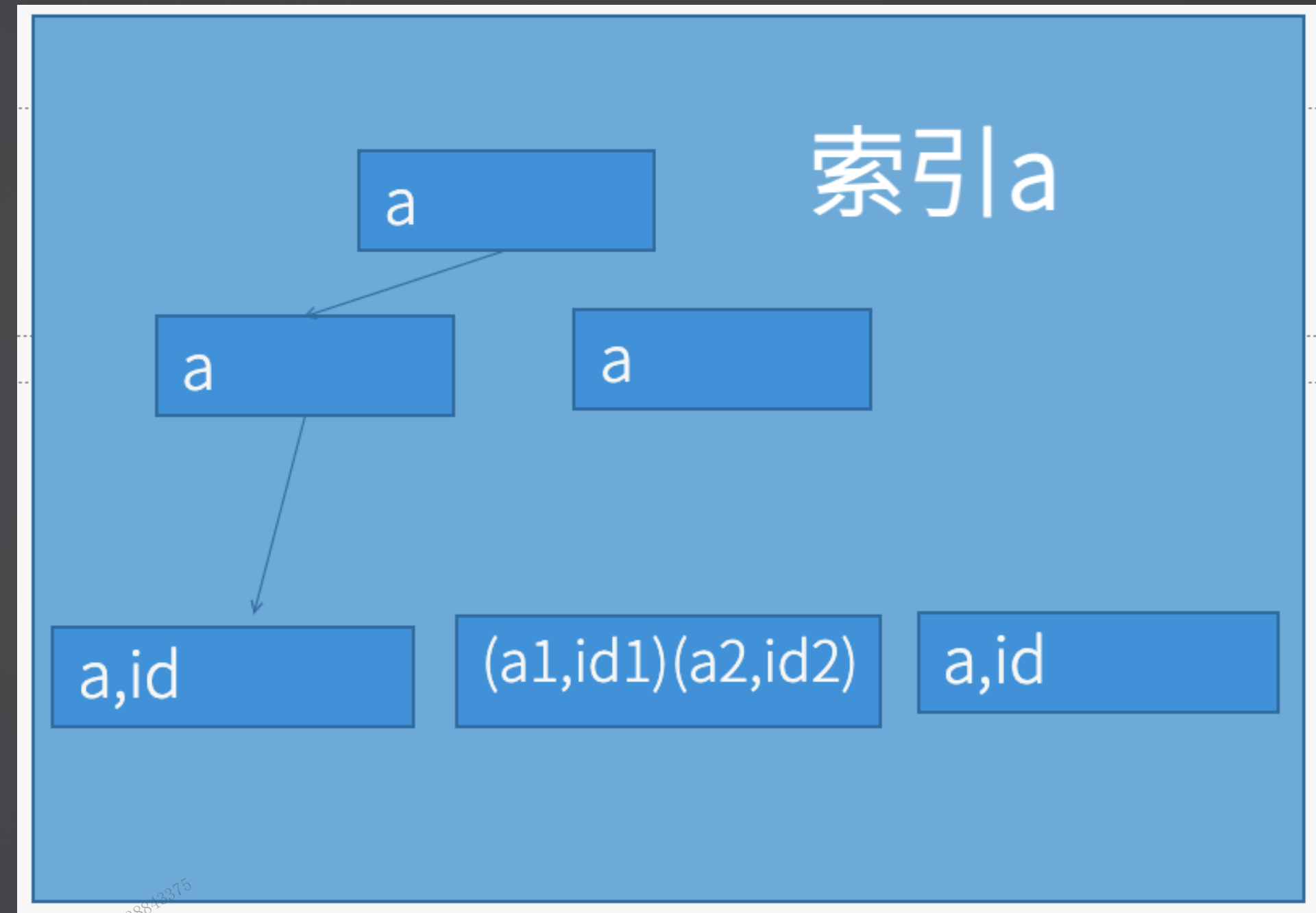
Ibuf (Change Buffer)

什么是 Change Buffer

前身是 Insert Buffer

加速更新，减少 I/O 读

```
3 thread/innoDB/io_ibuf_thread
4 thread/innoDB/io_read_thread
7 thread/innoDB/io_read_thread
6 thread/innoDB/io_read_thread
5 thread/innoDB/io_read_thread
10 thread/innoDB/io_write_thread
11 thread/innoDB/io_write_thread
9 thread/innoDB/io_write_thread
8 thread/innoDB/io_write_thread
```



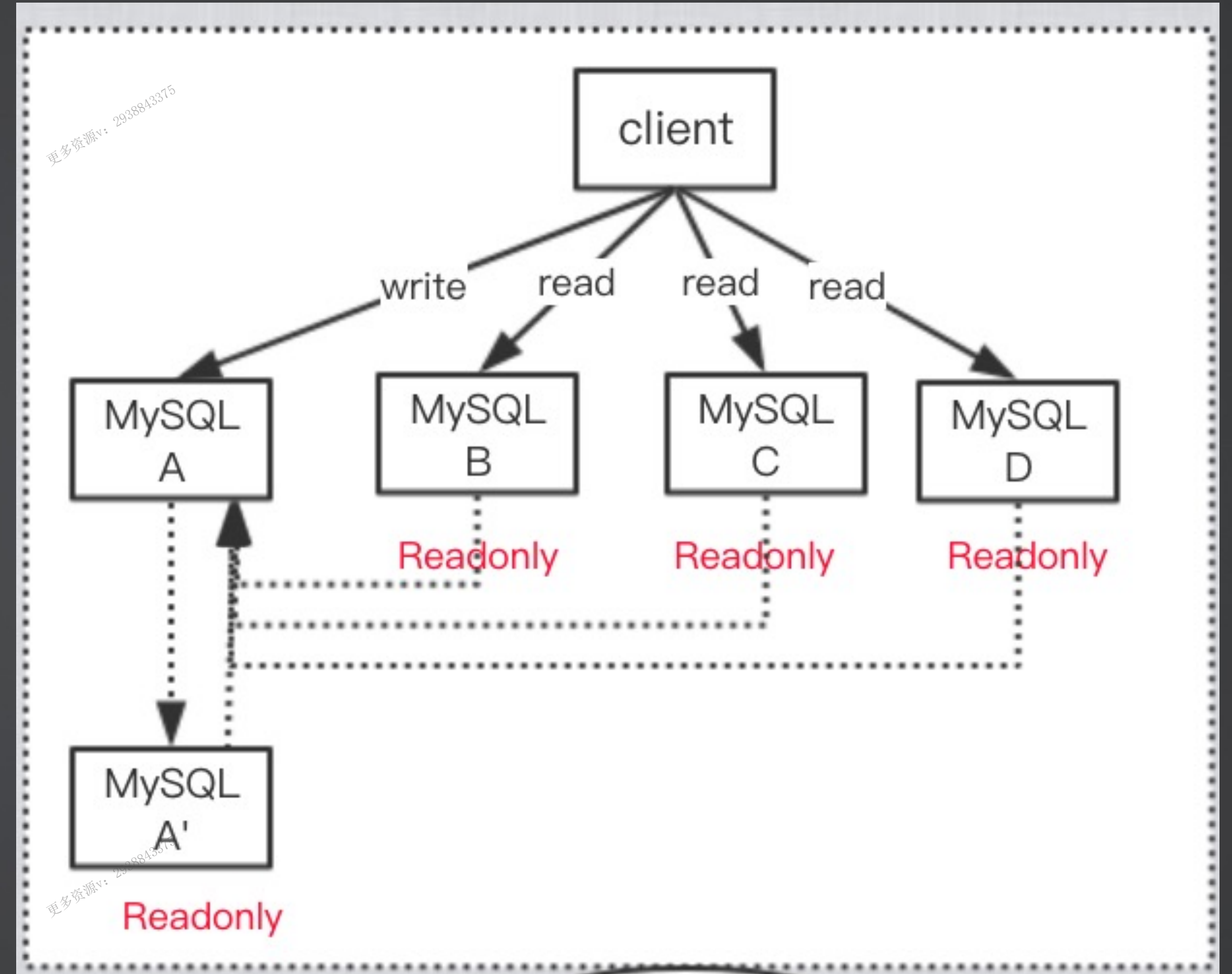
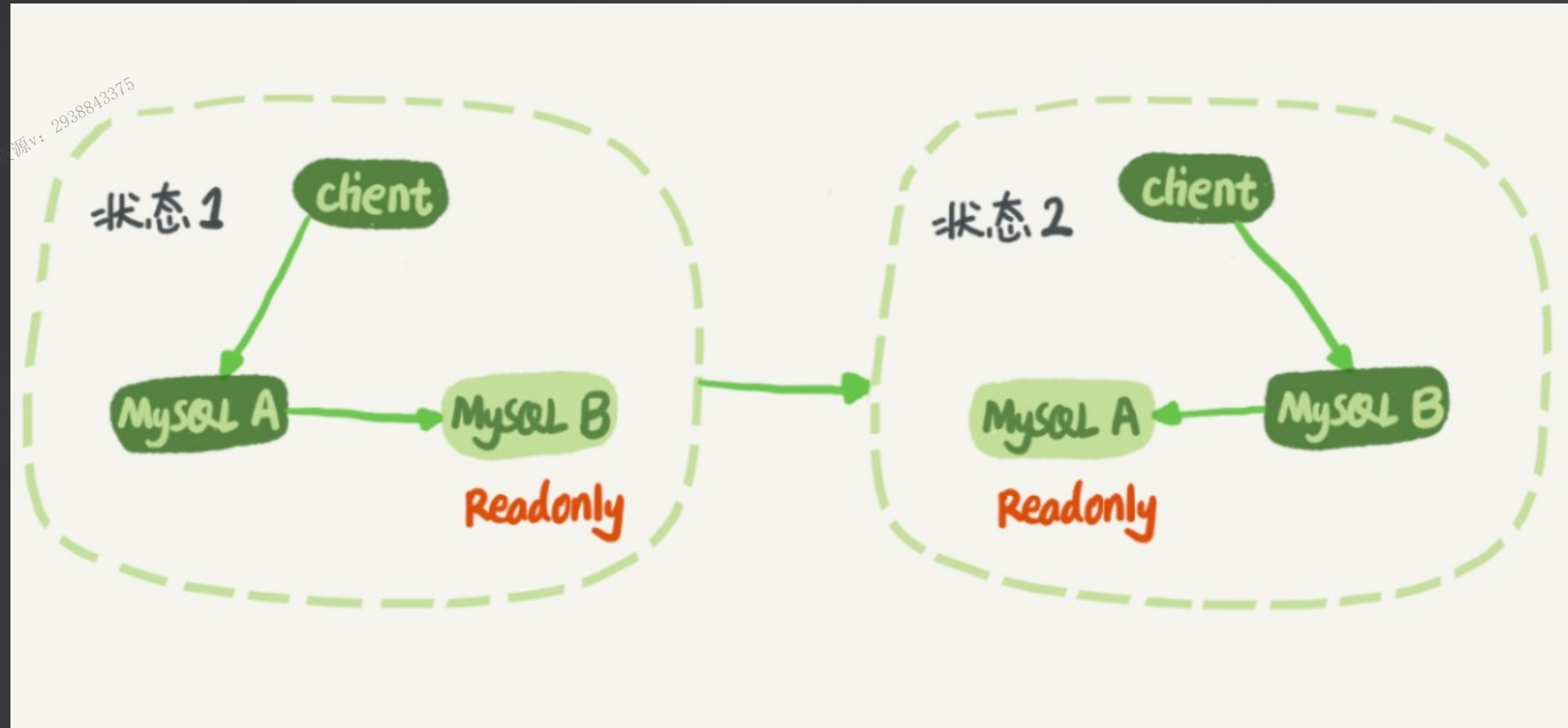
课堂练习：异常掉电后，change buffer 会不会丢失？

Change Buffer 收益最小的场景是什么？

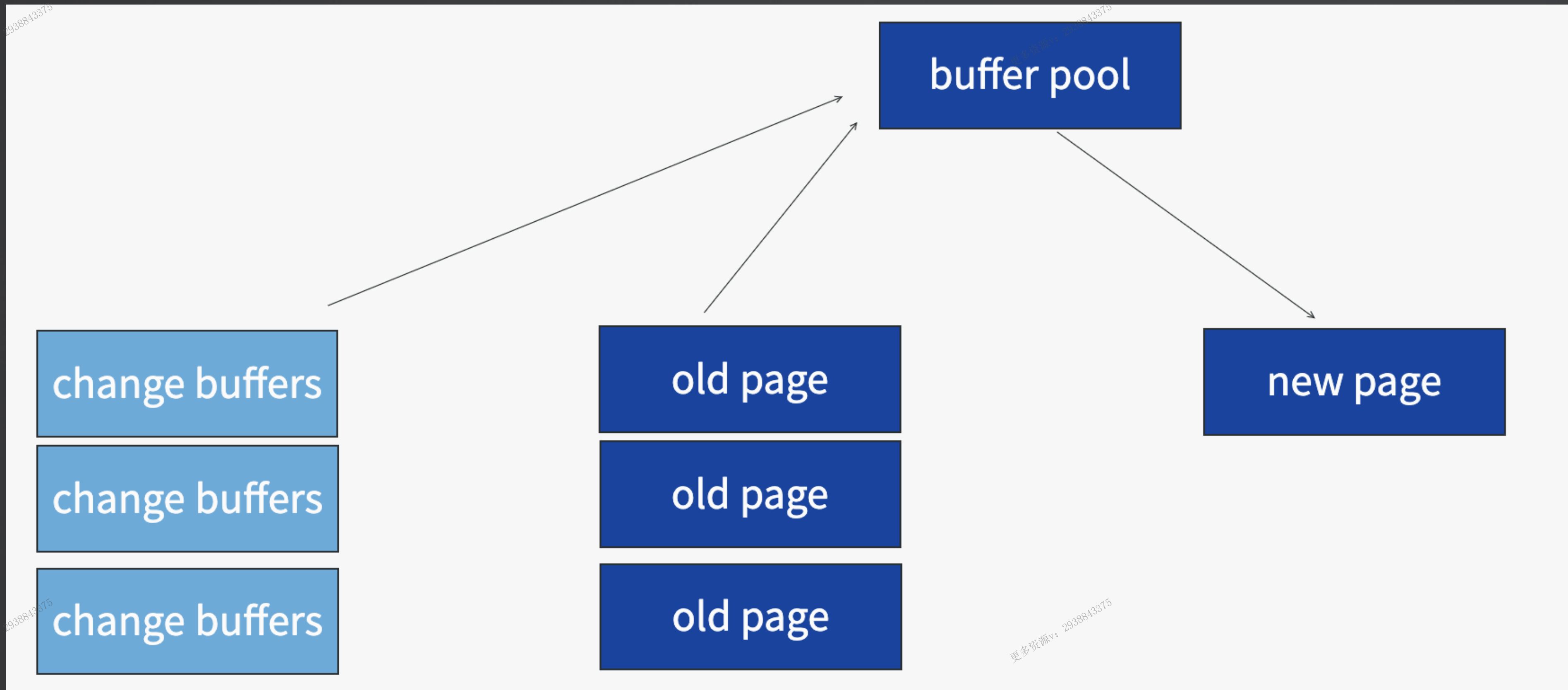
业务使用建议

讨论：主从结构下，为了提升从库的查询性能，又不影响主库的更新性能，单独给从库创建索引，这个方案可行吗？

参考回答



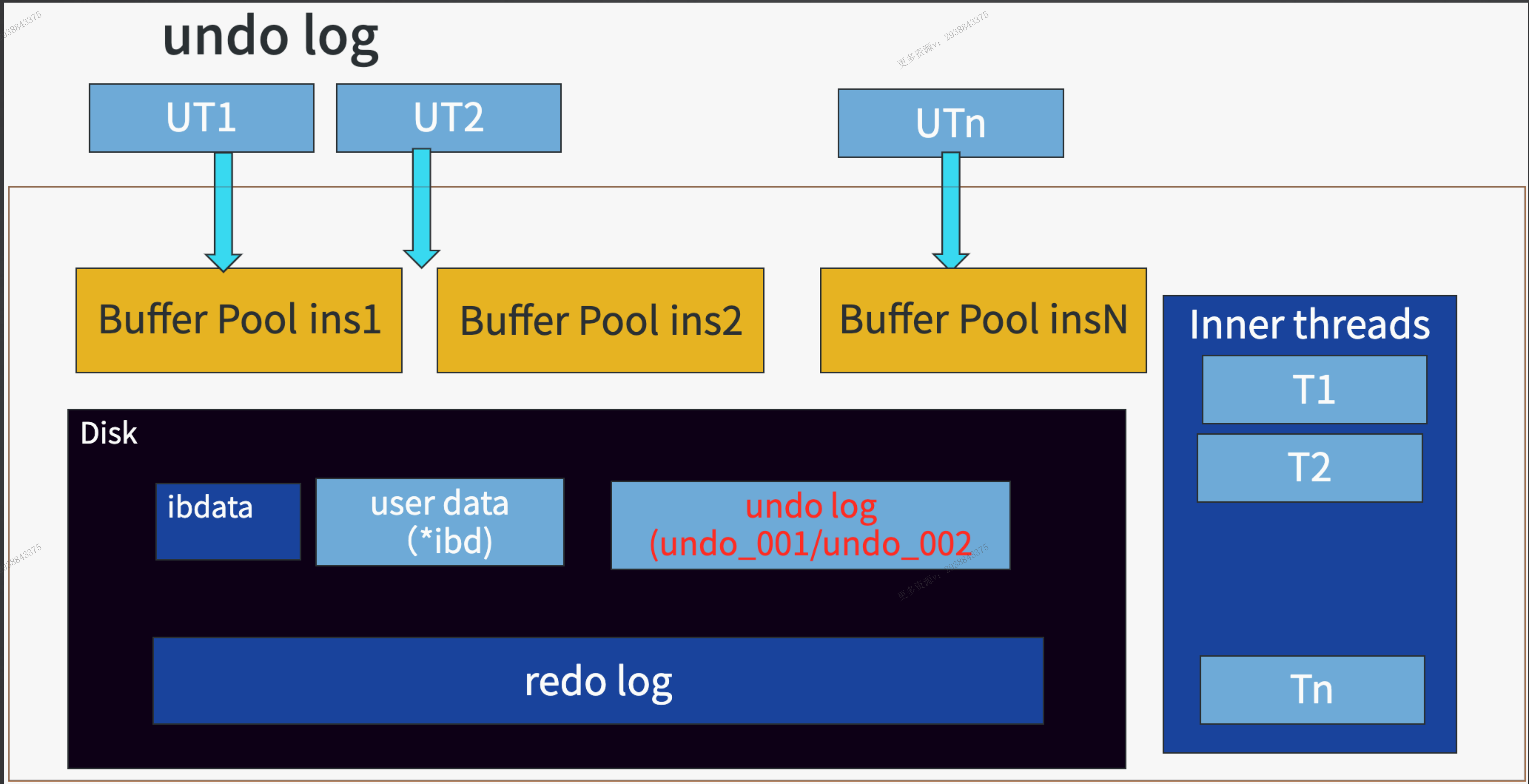
课堂练习：merge 操作需要经过内存吗？



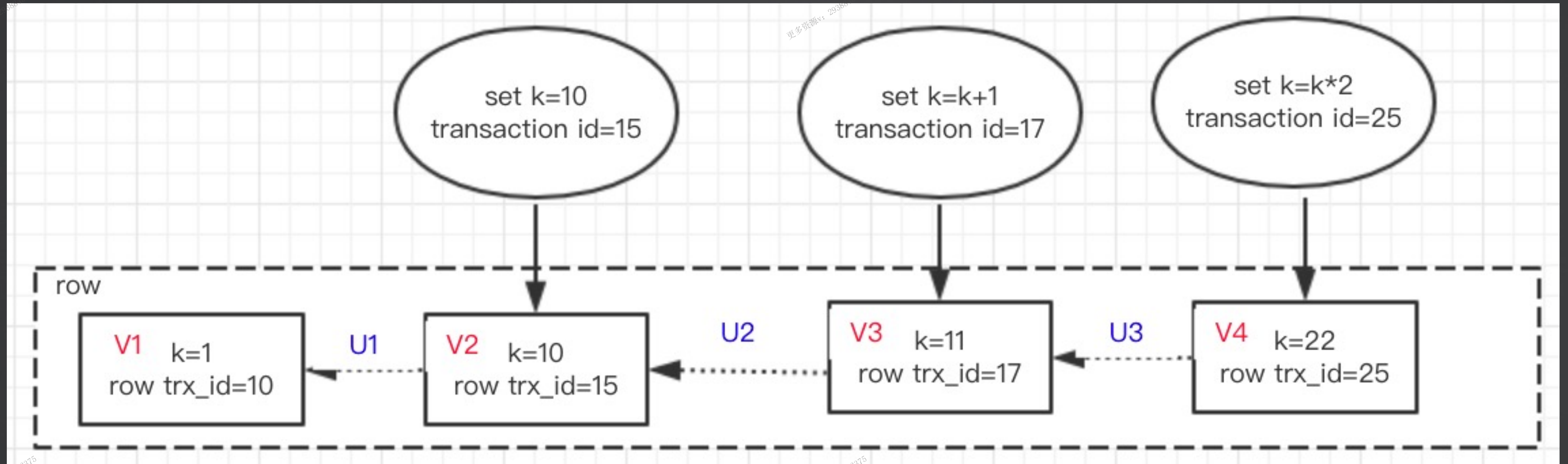
purge thread

35		thread/innodb/srv_purge_thread
36		thread/innodb/srv_worker_thread
37		thread/innodb/srv_worker_thread
38		thread/innodb/srv_worker_thread

purge thread



purge thread - 多版本的物理形态



purge thread

```
-----  
TRANSACTIONS  
-----
```

```
Trx id counter 1549
```

```
Purge done for trx's n:o < 1543 undo n:o < 0 state: running but idle
```

```
History list length 3
```

```
Total number of lock structs in row lock hash table 0
```

```
LIST OF TRANSACTIONS FOR EACH SESSION:
```

```
Old value = 3
```

```
New value = 2
```

```
trx_purge_remove_log_hdr (rseg_hdr=0x7fbd7b354026 "\377\377\377\376", log_hdr=0x7fbd7bab4442 "", mtr=0x7fbd0f7fd280)  
  at /root/dq/mysql-8.0.39/storage/innobase/trx/trx0purge.cc:410
```

```
410 }
```

```
(gdb) bt
```

```
#0  trx_purge_remove_log_hdr (rseg_hdr=0x7fbd7b354026 "\377\377\377\376", log_hdr=0x7fbd7bab4442 "", mtr=0x7fbd0f7fd280)  
  at /root/dq/mysql-8.0.39/storage/innobase/trx/trx0purge.cc:410
```

```
#1  0x0000000004d2076b in trx_purge_truncate_rseg_history (rseg=0x7fbd7e76c900, limit=0x7fbd7e74c2e0) at /root/dq/mysql-8.0.39/storage/innobase/trx/trx0purge.cc:16
```

```
#2  0x0000000004d236ca in trx_purge_truncate_history (limit=0x7fbd7e74c2e0, view=0x7fbd7e74c220) at /root/dq/mysql-8.0.39/storage/innobase/trx/trx0purge.cc:16
```

```
#3  0x0000000004d25576 in trx_purge_truncate () at /root/dq/mysql-8.0.39/storage/innobase/trx/trx0purge.cc:2383
```

```
#4  0x0000000004d25a7b in trx_purge (n_purge_threads=4, batch_size=300, truncate=true) at /root/dq/mysql-8.0.39/storage/innobase/trx/trx0purge.cc:2493
```

```
#5  0x0000000004cd51ef in srv_do_purge (n_total_purged=0x7fbd0f7fd9e0) at /root/dq/mysql-8.0.39/storage/innobase/srv/srv0srv.cc:2953
```

```
#6  0x0000000004cd5a7d in srv_purge_coordinator_thread () at /root/dq/mysql-8.0.39/storage/innobase/srv/srv0srv.cc:3122
```

```
#7  0x0000000004b75508 in std::__invoke_impl<void, void (*)()> (__f=@0xcb8fda8: 0x4cd583d <srv_purge_coordinator_thread()>) at /usr/include/c++/8/bits/invoke.h:95
```

```
#8  0x0000000004b753ae in std::__invoke<void (*)()> (__fn=@0xcb8fda8: 0x4cd583d <srv_purge_coordinator_thread()>) at /usr/include/c++/8/bits/invoke.h:95
```

```
#9  0x0000000004b75070 in std::invoke<void (*)()> (__fn=@0xcb8fda8: 0x4cd583d <srv_purge_coordinator_thread()>) at /usr/include/c++/8/functional:81
```

```
#10 0x0000000004b74c83 in Detached_thread::operator()<void (*)()> (this=0xcb8fdb0, f=@0xcb8fda8: 0x4cd583d <srv_purge_coordinator_thread()>)
```


课堂练习

session1	session2
begin; select * from t limit 1;	
	History list 5 insertt into t2 ... History list ?

课堂练习

session1	session2
begin; select * from t limit 1;	
	History list 10 insertt into t ... //执行1000次 History list 1010 truncate table t //语句现象?

课堂练习

session1	session2
<pre>begin; select * from t2 limit 1;</pre>	
	<pre>History list 10 insertt into t ... //执行1000次 History list 1010 truncate table t //语句现象?</pre>
<pre>select * from t; //语句现象?</pre>	

page_flush_coordinator_thread

脏页 vs 干净页

The innodb_page_cleaners 默认值 4

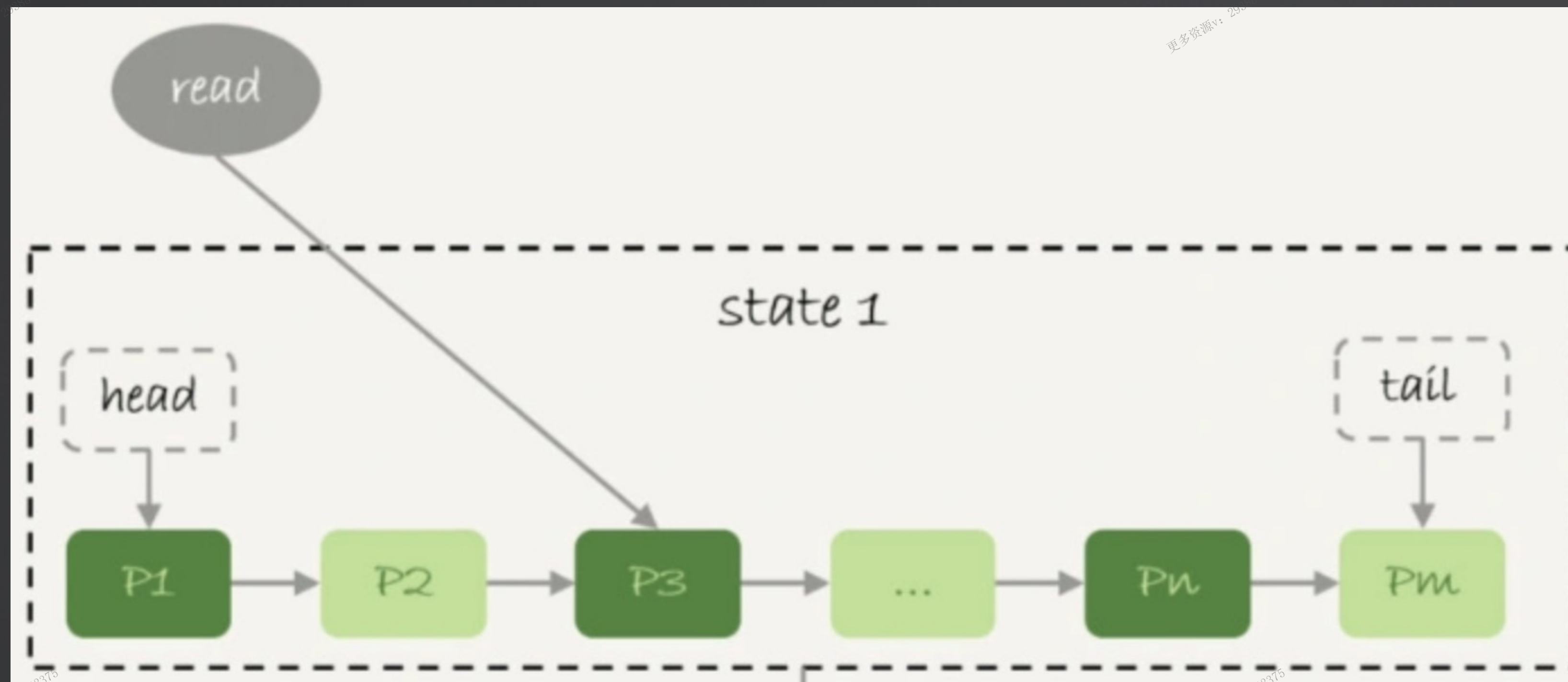
- 不超过 innodb_buffer_pool_instances
- 设置 innodb_buffer_pool_instances =2, 重启

```
12 | thread/innodb/page_flush_coordinator_thread |
13 | thread/innodb/page_flush_thread           |
```

BUFFER POOL AND MEMORY

```
-----
Total large memory allocated 0
Dictionary memory allocated 746192
Buffer pool size      1535
Free buffers          735
Database pages        796
Old database pages    695
Modified db pages 0
Pending reads         0
Pending writes: LRU 0, flush list 0, single page 0
```

课堂练习：查询会触发刷脏页吗？



课堂练习：

机器 I/O 压力大，影响业务响应，应该把 `innodb_io_capacity` 调大还是调小？

课堂练习

假设场景在MySQL8.0的默认配置下，初始化一个空示例和空库，建一个serv表，有31个字段，除了主键id外， a1~a29定义为varchar(255), a30定义为text, 初始化1行数据。

```
1 CREATE TABLE serv( id int(11) NOT NULL AUTO_INCREMENT,  
2 a1 varchar(255) DEFAULT NULL,  
3 a2 varchar(255) DEFAULT NULL,  
4 a3 varchar(255) DEFAULT NULL,
```


课堂练习

在可重复读隔离级别(RR)下，分别执行以下两个场景；

场景1：

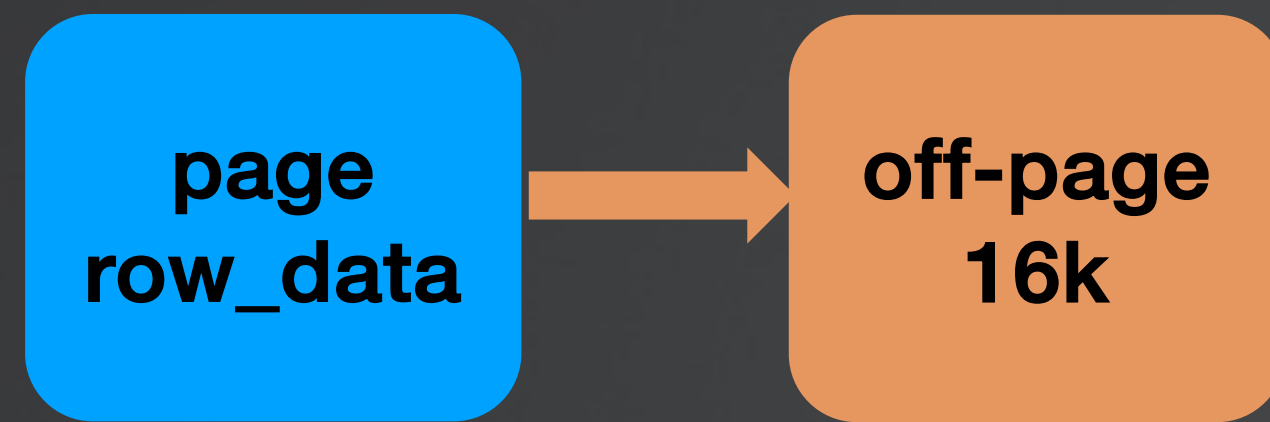
```
1  T0: session1
2  start transaction with consistent snapshot
3
4  T1: session2:
5  loop 5000 times:
6  update serv set a1=repeat('a', 255), a2=repeat('a',255)...a29=repeat('a',255);
7  update serv set a1=repeat('b', 255), a2=repeat('b',255)...a29=repeat('b',255);
8  end loop;
```

课堂练习

场景2:

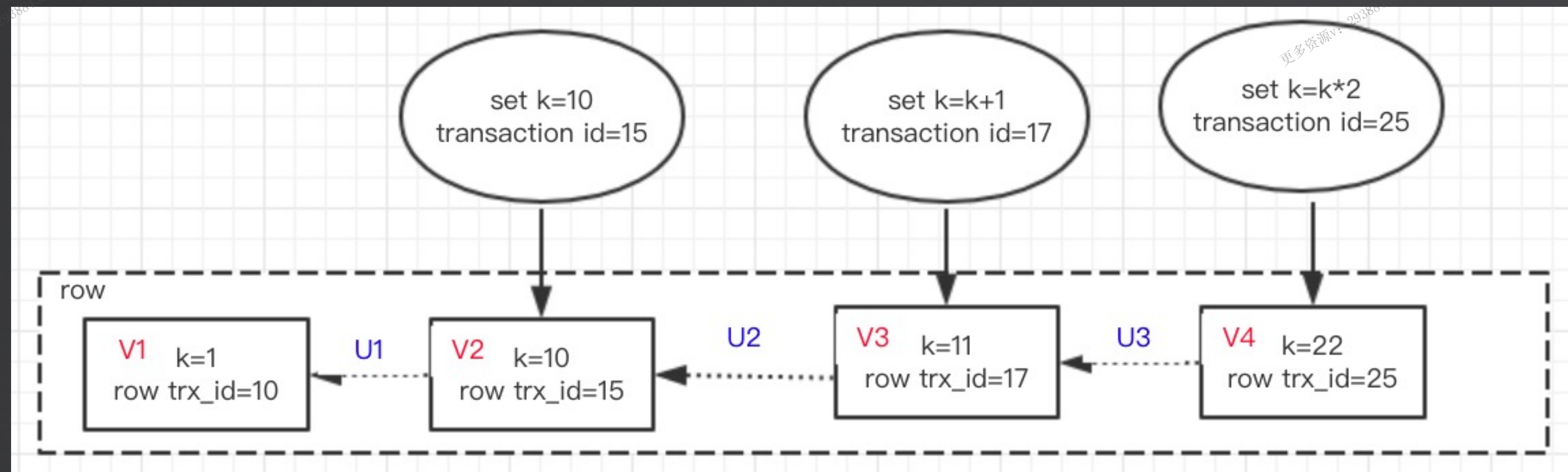
```
1  T0: session1
2  start transaction with consistent snapshot
3
4  T1: session2:
5  loop 5000 times:
6    update serv set a30=repeat('a', 29*255);
7    update serv set a30=repeat('b', 29*255);
8  end loop
```

课堂练习: off-page



1. 页字段总长度超过 8K
2. text/blob 单个字段长度超过 768 字节

课堂练习:



off-page
16k

off-page
16k

off-page
16k

课堂练习

场景2:

```
1  T0: session1
2  start transaction with consistent snapshot
3
4  T1: session2:
5  loop 5000 times:
6    update serv set a30=repeat('a', 29*255);
7    update serv set a30=repeat('b', 29*255);
8  end loop
```


课堂练习：一个正常运行的实例，rm 掉 ibdata，会怎么样？

- A. 直接重启，并正常工作
- B. 直接崩溃，且无法重启正常工作
- C. 服务继续正常使用

Q&A

THANKS