

性能优化

目录

1

部署优化

2

MySQL 内部性能优化

3

使用优化

性能优化策略的设计思想

1. 最大化利用资源
2. 最小化使用资源
3. 业务优先

部署优化

1. 单机多实例 vs 单实例
2. 要不要绑定CPU
3. 是否使用 tcmalloc / jemalloc

部署优化：针对固定的业务场景，从编译环节入手提升性能

课堂练习： 以下这段代码，输入 1 亿个偶数和输入 1 亿个奇数的执行时间对比

```
if (a %2 == 0){  
    M++  
}  
else{  
    N++  
}
```

- A. 输入偶数显著快于输入奇数
- B. 输入偶数显著慢于输入奇数
- C. 执行时间差不多

MySQL: 配置文件引导优化 (PGO) $\geq 8.0.19$

-DFPROFILE_GENERATE=ON

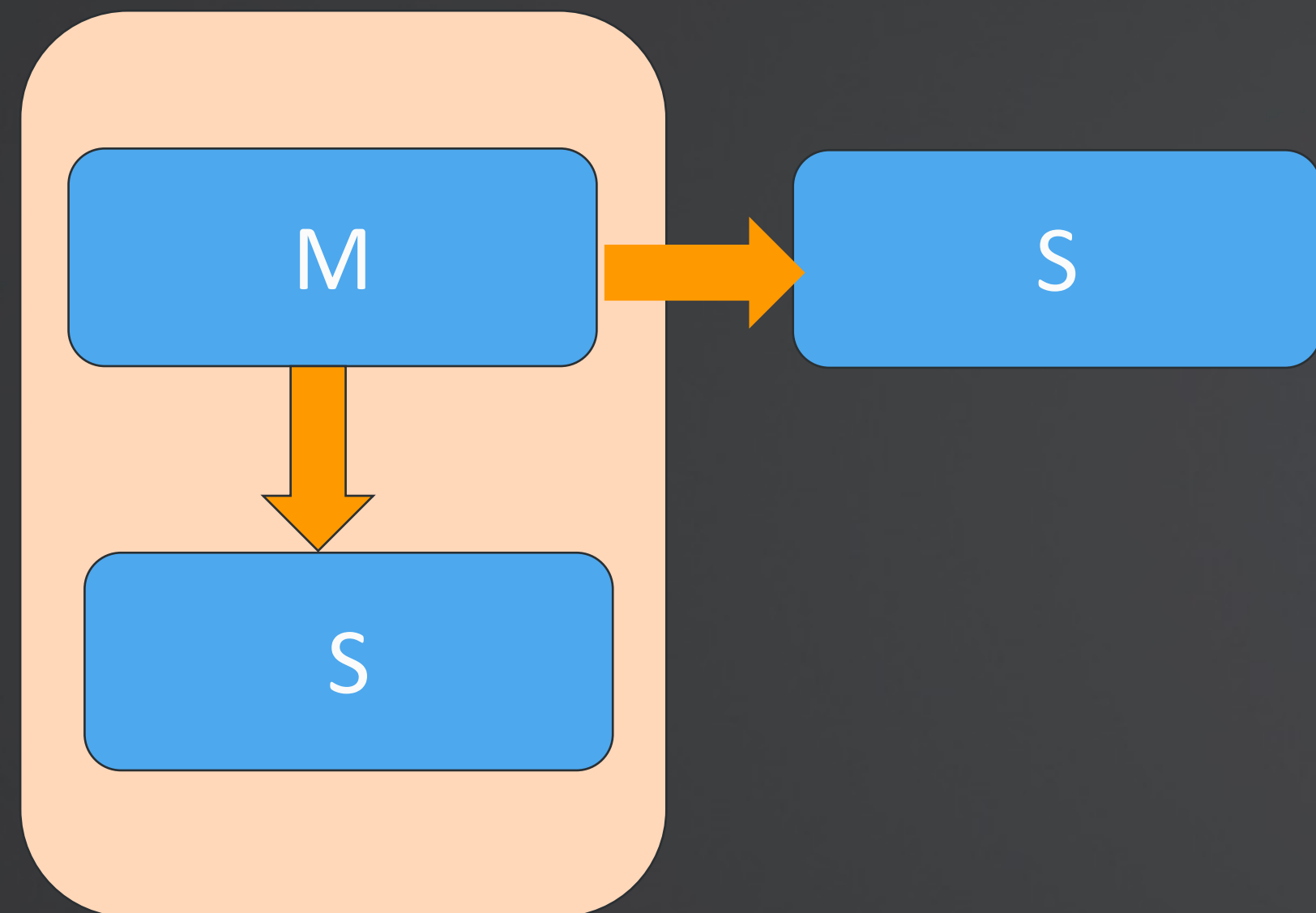
mysqld

train by running test

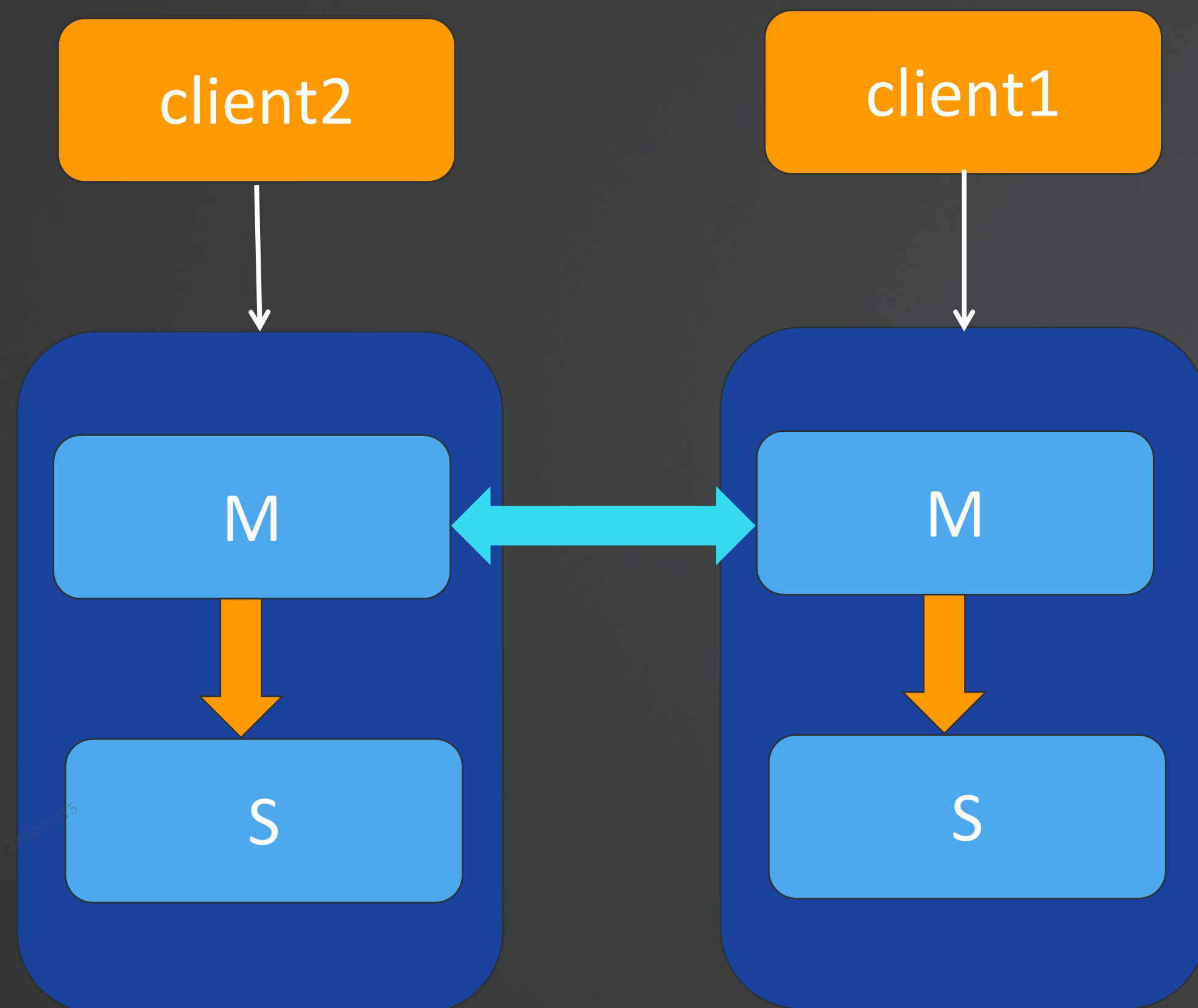
-DFPROFILE_USE=ON

mysqld

部署优化 Semi-sync



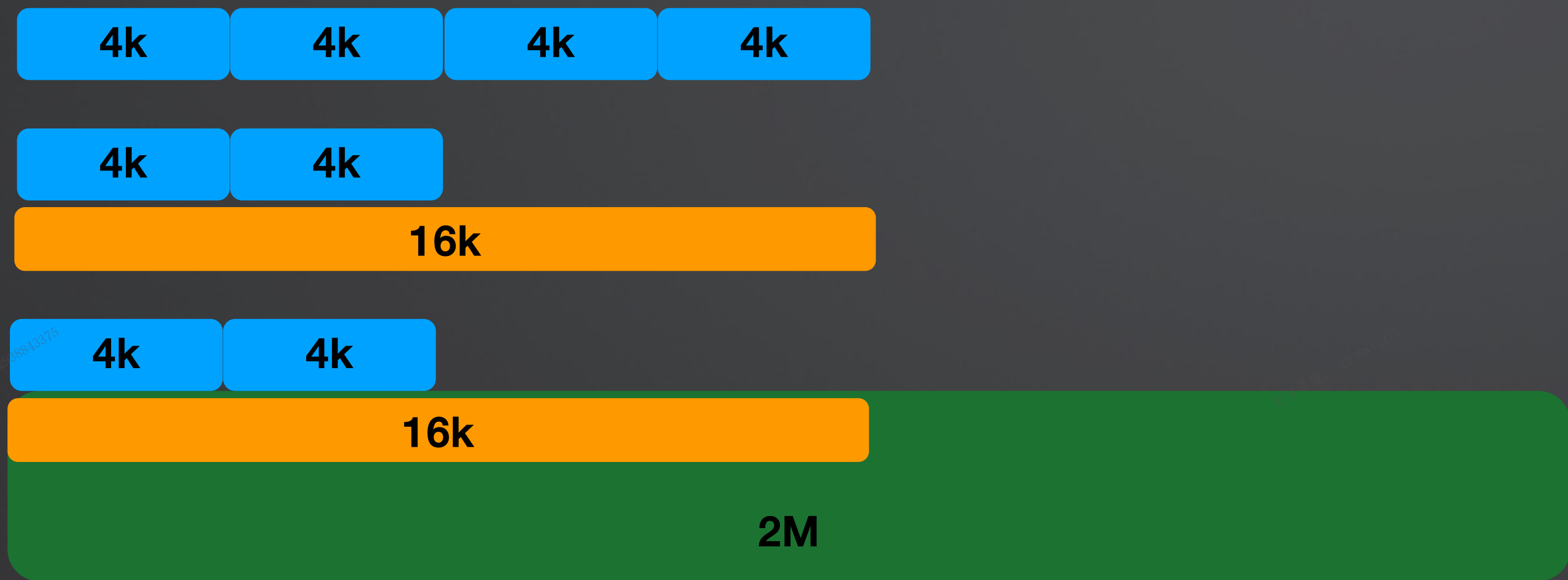
部署优化-单元化



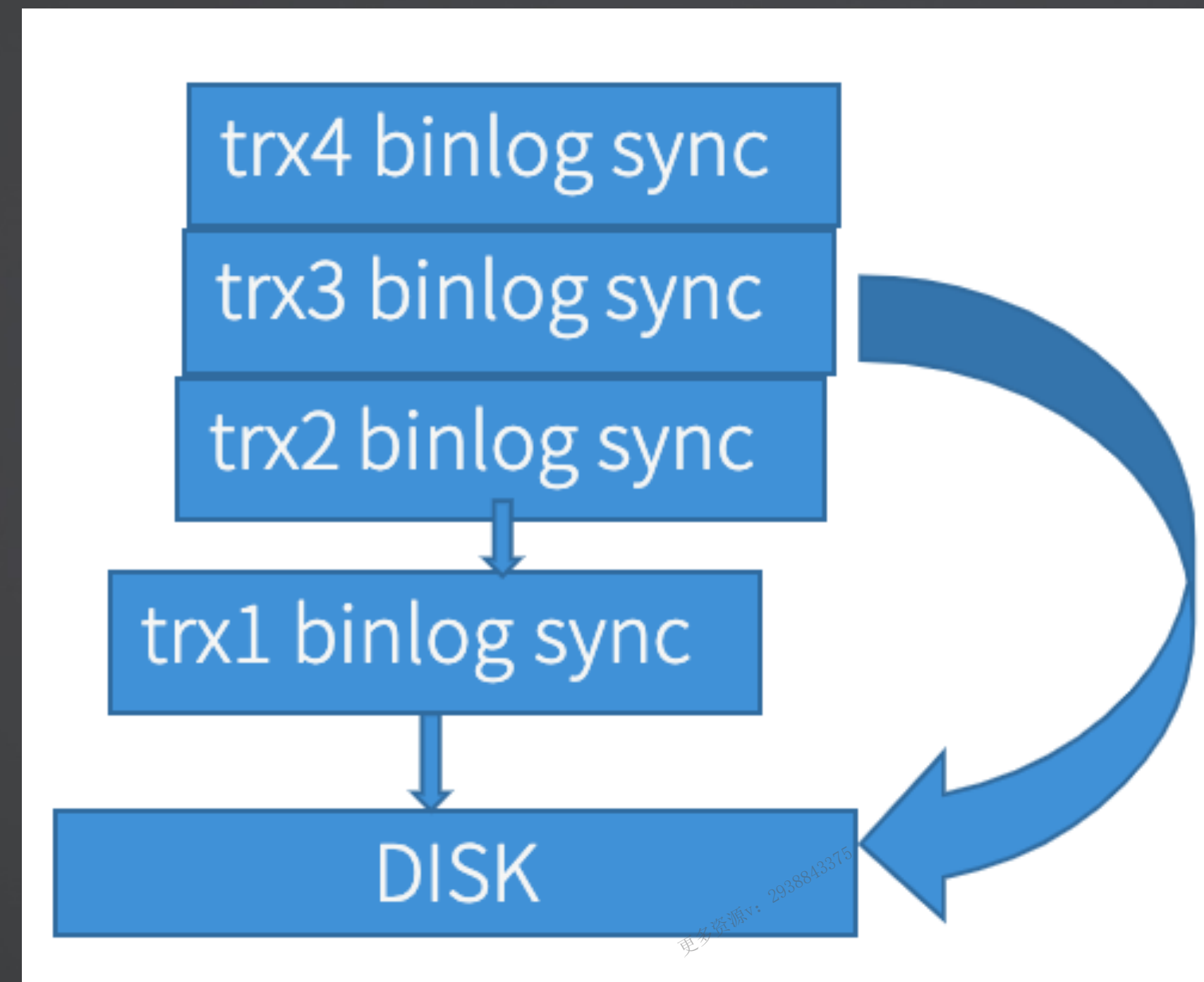
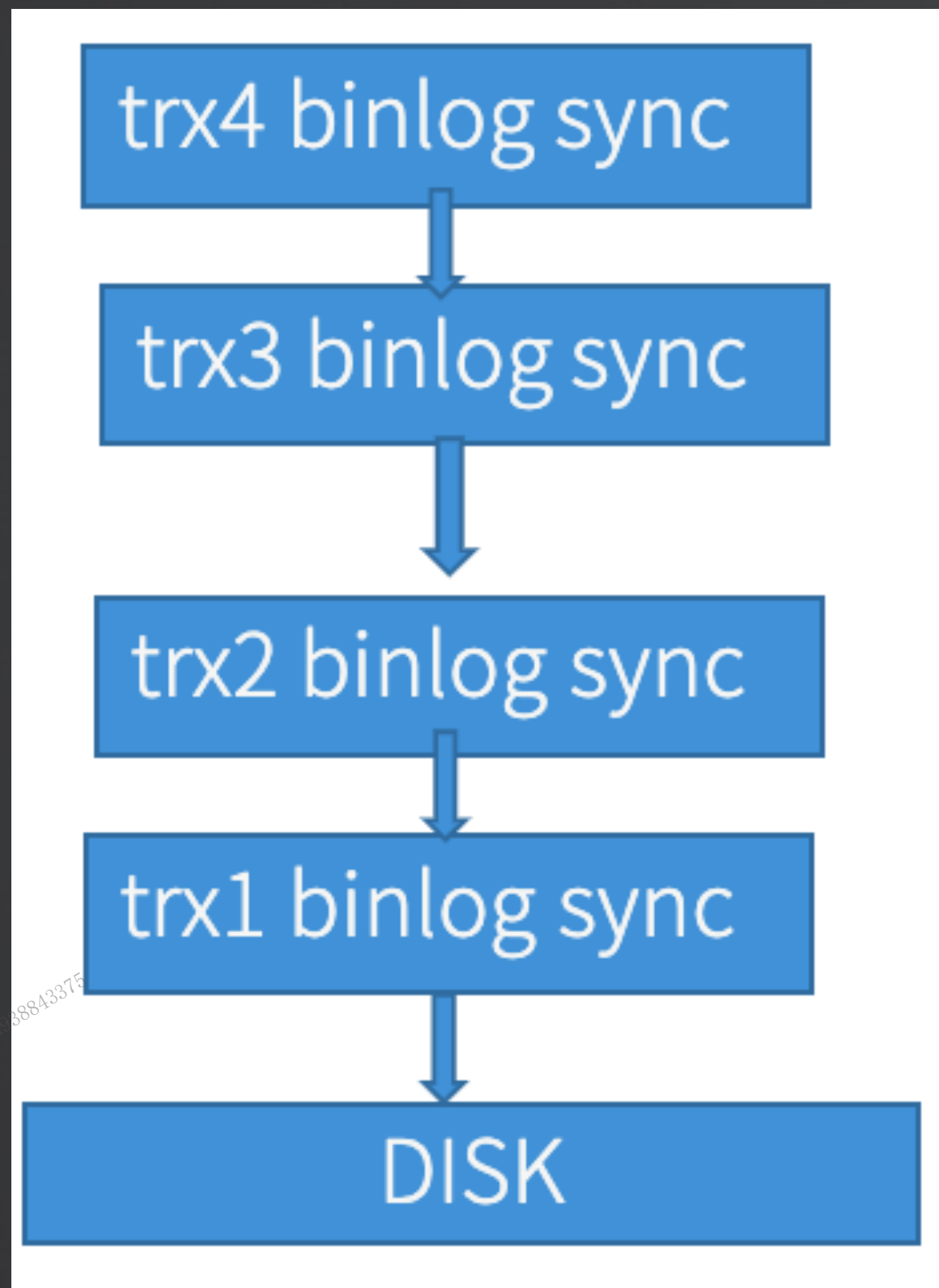
部署优化

binlog/redo log 独立部署

double write buffer -> 原子写
pwrite (16 k)



MySQL 内部优化 -- group commit



MySQL 内部优化--更新短路

MySQL 8.0 RR 隔离级别, row_image=full

session1	session2
drop table t; create table t(id int primary key , a int); insert into t values(1,1); begin; select * from t where id = 1;	
	update t set a=3 where id=1;
update t set a=3 where id=1; select * from t where id = 1; //(1,1)	

session1	session2
drop table t; create table t(id int primary key , a int); insert into t values(1,1); begin; select * from t where id = 1;	
	update t set a=3 where id=1;
set binlog_row_image=minimal; update t set a=3 where id=1; select * from t where id = 1; //(1,3)	

MySQL 内部优化--查询短路

`select count(*) from t where id is null;`

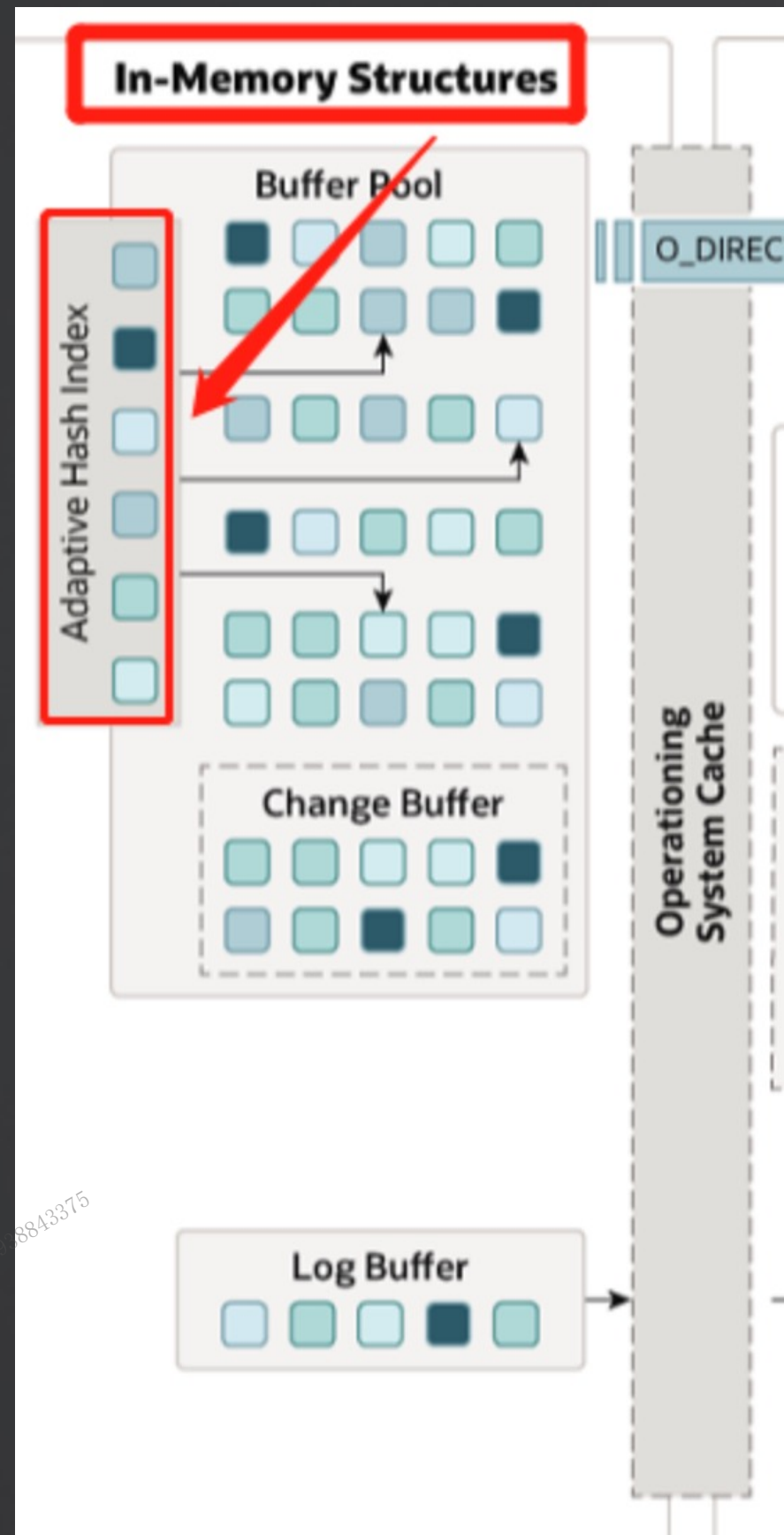
`select distinct id from t where`

`select sum(sleep(0.5)) from t;`

ID
id1
id2

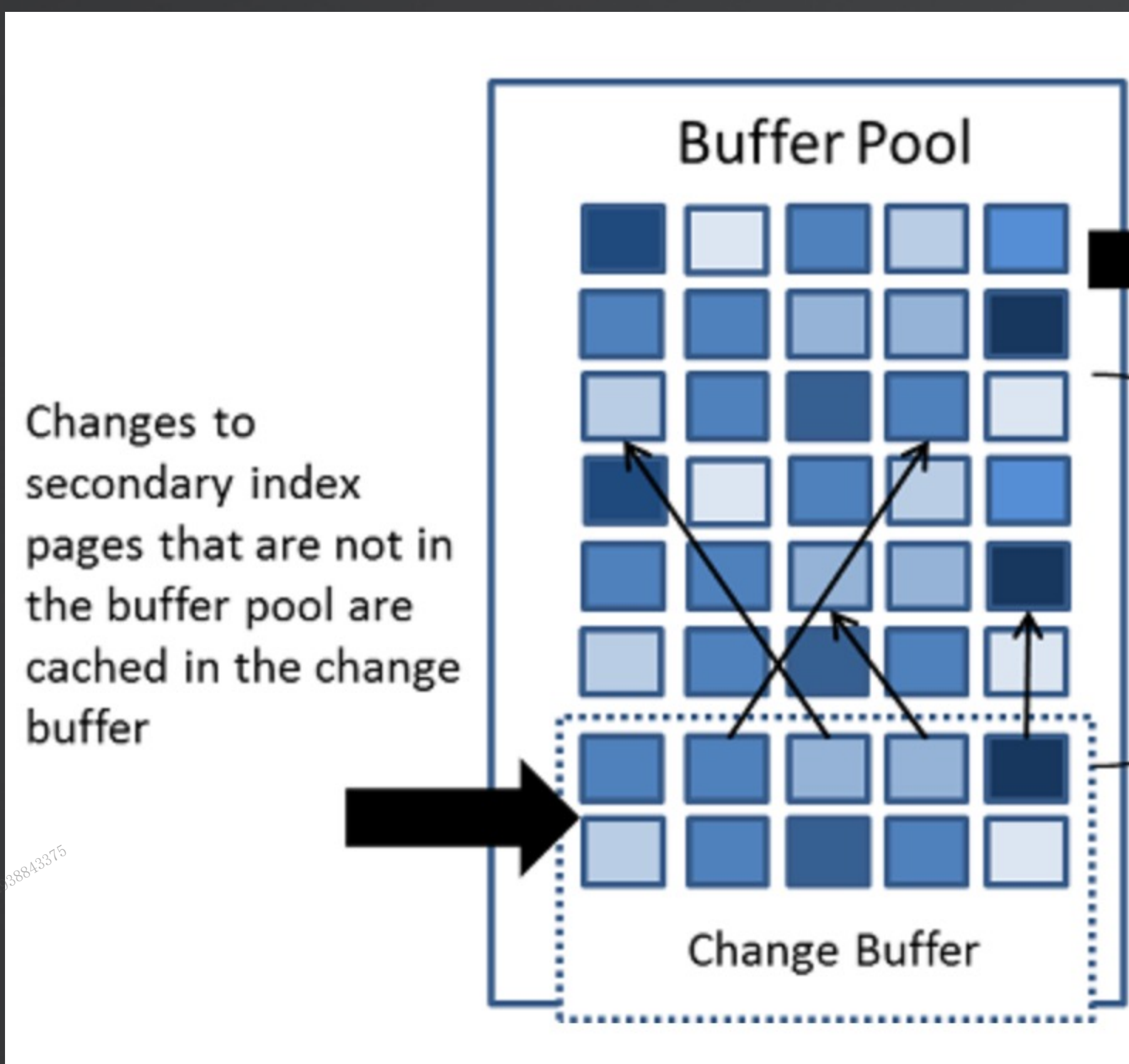
```
id: 1
select_type: SIMPLE
table: t
partitions: NULL
type: index
possible_keys: PRIMARY
key: PRIMARY
key_len: 4
ref: NULL
rows: 2
filtered: 100.00
Extra: Using index
1 row in set, 1 warning (0.00 sec)
```

MySQL 内部优化 - AHI



0.00 hash searches/s, 0.00 non-hash
searches/s

MySQL 内部优化 -- change buffer



课堂练习：SQL 语句的扫描行数怎么算？

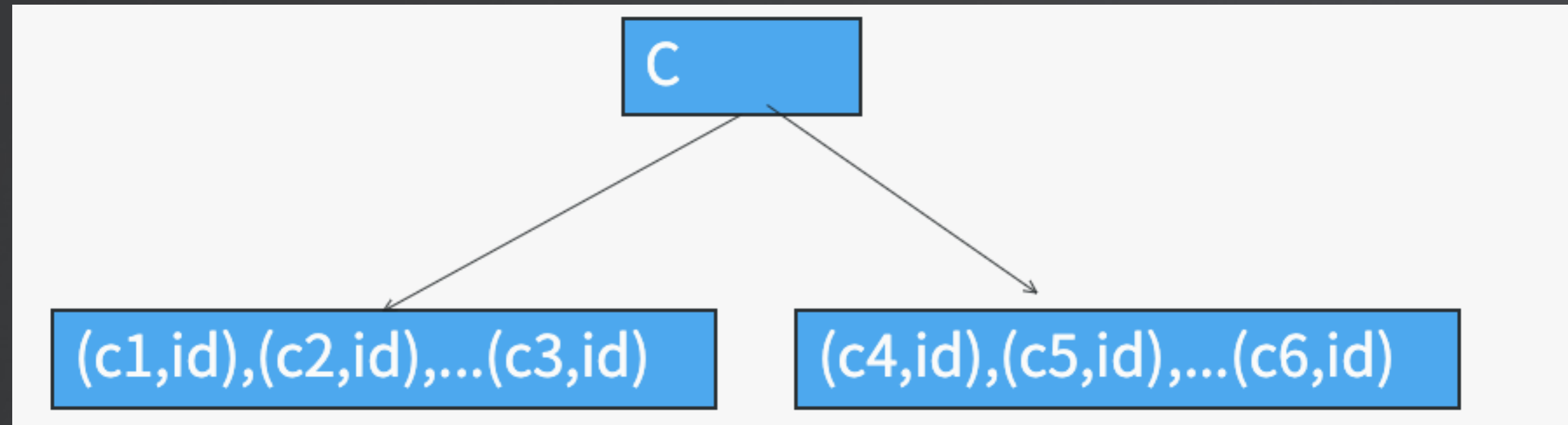
```
CREATE TABLE `t` (  
  `id` int NOT NULL,  
  `c` int DEFAULT NULL,  
  `d` varchar(255),  
  PRIMARY KEY (`id`),  
  KEY `c` (`c`)  
) ENGINE=InnoDB;  
insert into t(id,c) values(1,1),(2,2);
```

版本 MySQL 8.0.39，以下每个语句的 rows_examined 分别是多少

1. select id from t force index(c) where c=1;
2. select id from t force index(primary) where c=1;
3. select count(*) from t;
4. select id, rand() as d from t order by d;

相关知识点

force index(c), 使用索引 c, 直接定位到1行



force index(primary), primary 索引中无法保证 c 是有序的, 因此必须全索引扫描, 扫描行数 2。

相关知识点

$\leq 8.0.16$

Server



InnoDB

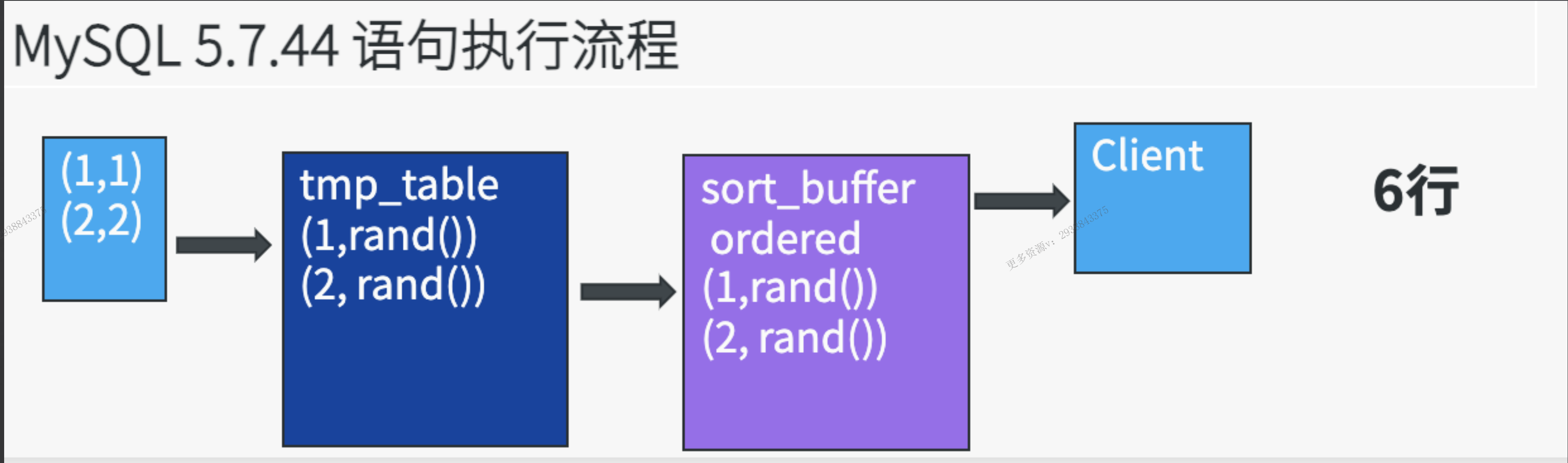
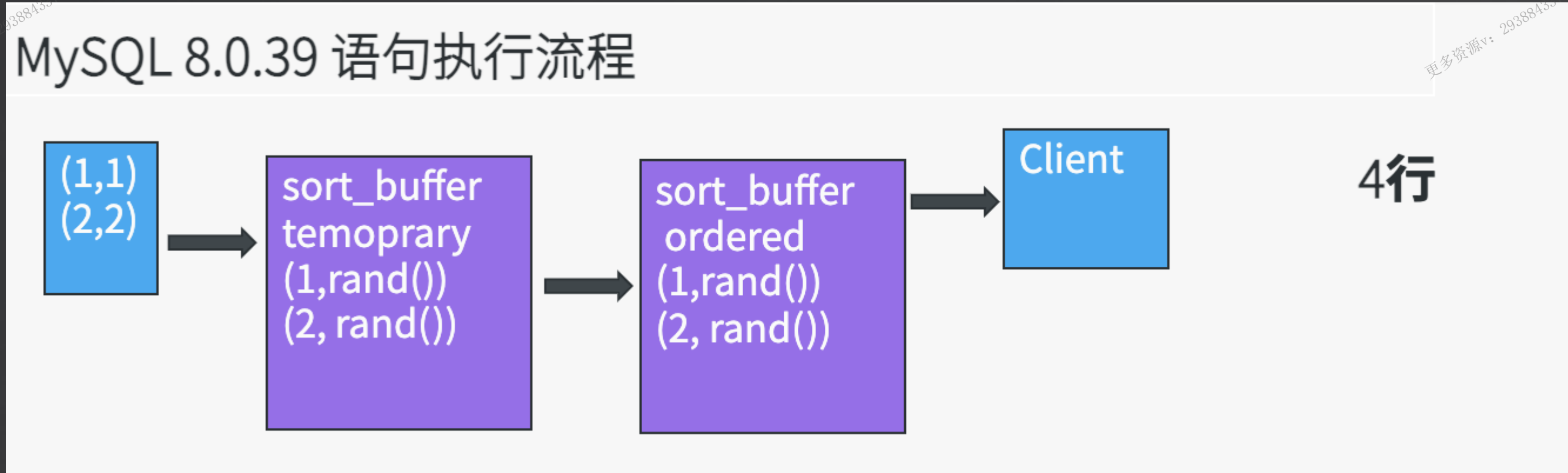
$\geq 8.0.17$

ha_innobase::records

rows_examined=0

8.0.37

相关知识点



课堂练习：SQL 语句的扫描行数怎么算？

```
CREATE TABLE `t` (  
  `id` int NOT NULL,  
  `c` int DEFAULT NULL,  
  `d` varchar(255),  
  PRIMARY KEY (`id`),  
  KEY `c` (`c`)  
) ENGINE=InnoDB;  
insert into t(id,c) values(1,1),(2,2);
```

版本 MySQL 8.0.39 ， 以下每个语句的 rows_examined 分别是多少

1. select id from t force index(c) where c=1;
2. select id from t force index(primary) where c=1;
3. select count(*) from t;
4. select id, rand() as d from t order by d;

MySQL 参数优化-- time_zone

time_zone = system

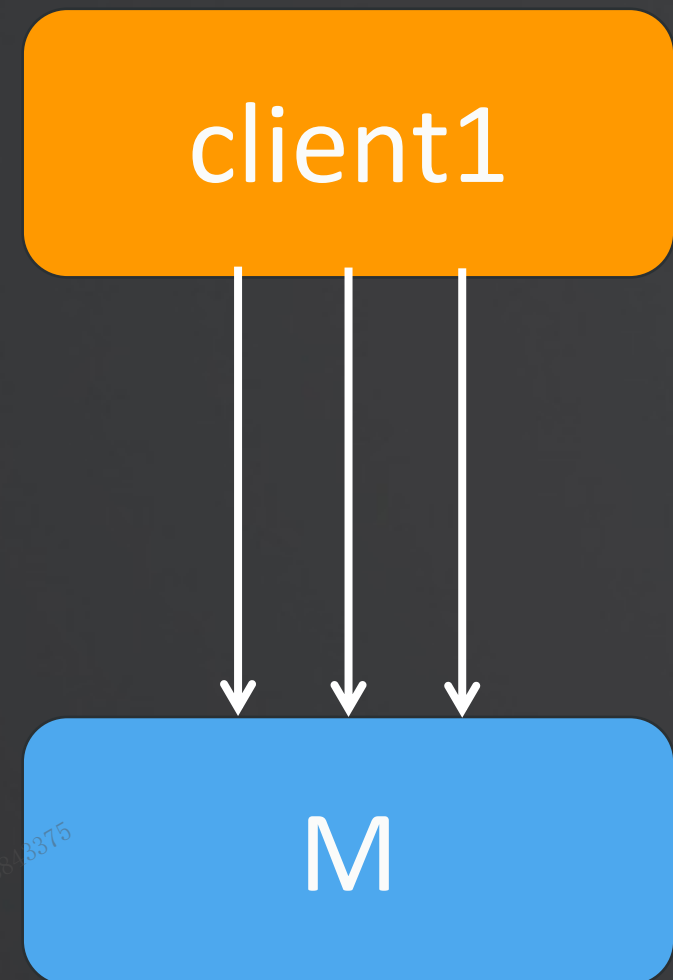


time_zone = +8:00

应用层优化

减少交互次数: CLIENT_MULTI_STATEMENTS

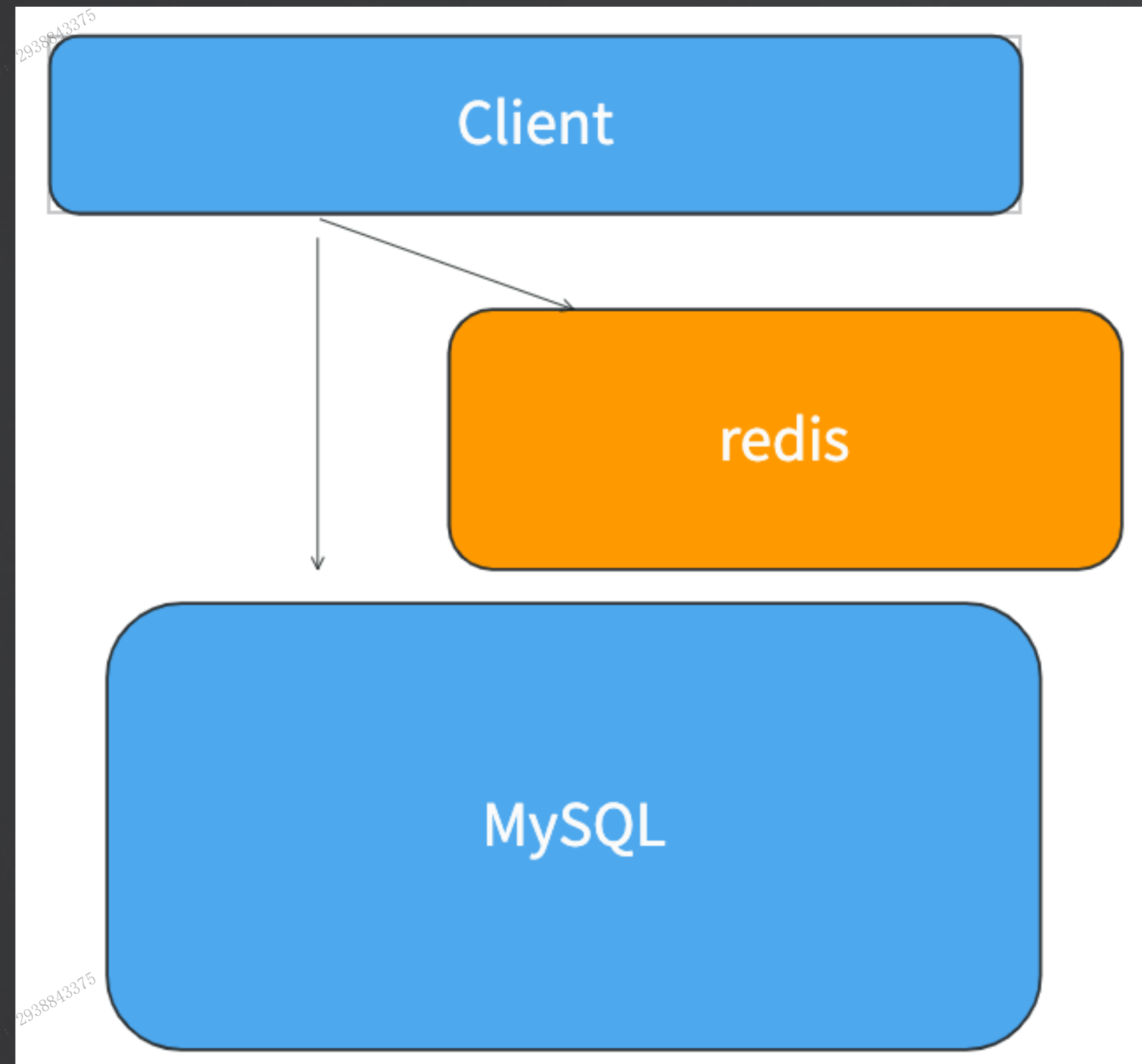
长短连接选择



resetconnection

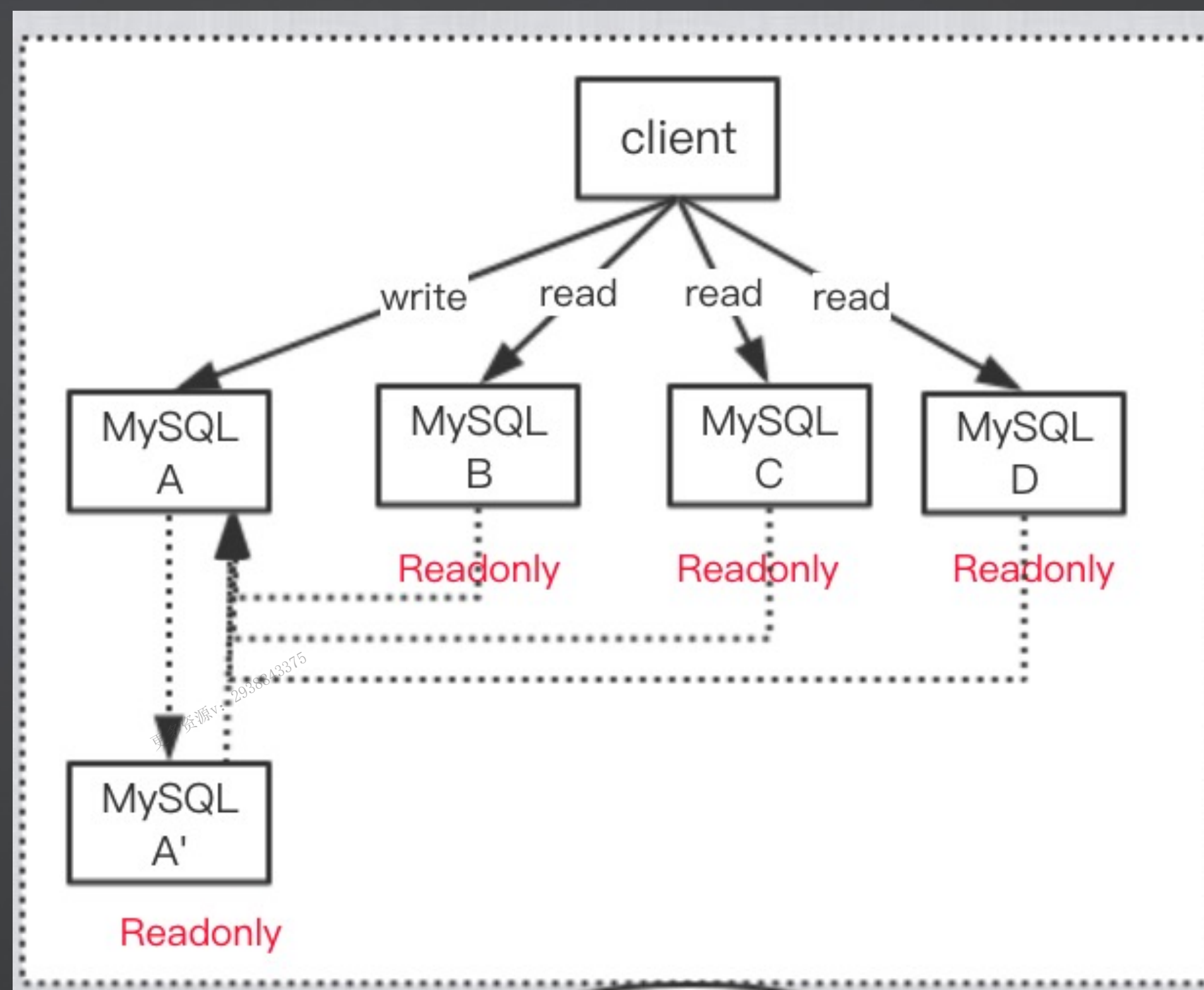
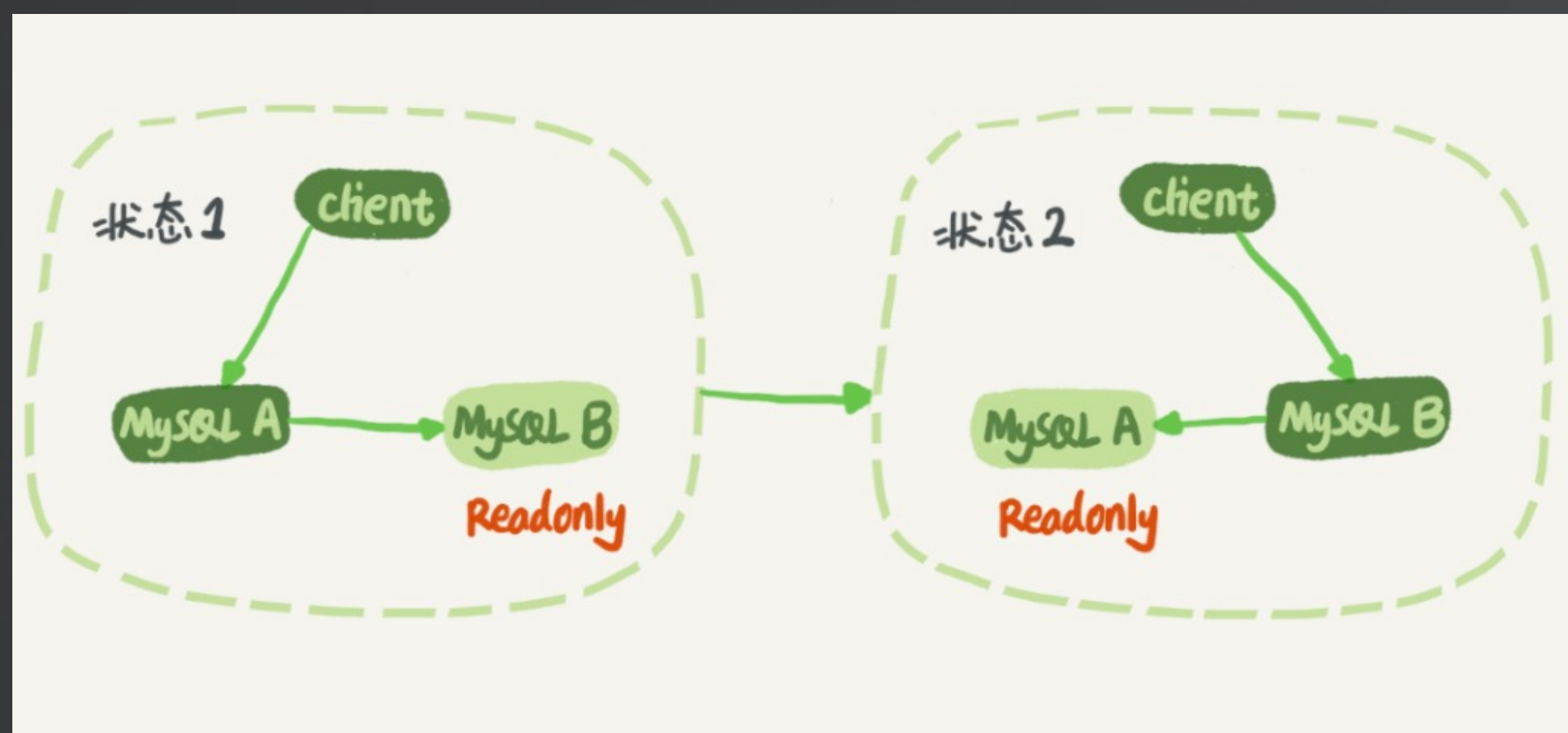
主动重连

应用层优化--Redis



应用层优化 - 异构索引

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？



课堂练习: 用短的停机时间, 删除主键

```
CREATE TABLE `f` (  
  `id` varchar(36) NOT NULL ,  
  `a` varchar(36) DEFAULT ,  
  `b` varchar(36) DEFAULT ,  
  `c` varchar(36) DEFAULT ,  
  `d` varchar(36) DEFAULT ,  
  `e` varchar(50) DEFAULT ,  
  `f` varchar(100) DEFAULT ,  
  `g` datetime DEFAULT NULL COMMENT '行为操作时间',  
  `h` varchar(50) DEFAULT ,  
  `i` char(1) DEFAULT ,  
  `j` varchar(2500) DEFAULT ,  
  `k` varchar(500) DEFAULT ,  
  `l` varchar(500) DEFAULT ,  
  `m` varchar(30) DEFAULT ,  
  `n` varchar(255) DEFAULT ,  
  PRIMARY KEY (`id`) USING BTREE,  
  KEY `ind_d` (`d`) USING BTREE ,  
  KEY `ind_hx` (h,n) USING BTREE ,  
  KEY `file_operate_time` (h) USING BTREE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8  
ROW_FORMAT=DYNAMIC;
```

说明: 业务可以接受停机时间,要求是越短越好,
因此不考虑使用 ost 工具

分析:

对需求本身优化:

1. 这个表有第二个索引,第三个索引可以不需要;
2. 不能直接删除主键, 否则表没有主键,以后问题更大

```
KEY `ind_d` (d) USING BTREE ,
```

```
KEY `ind_hx` (h,n) USING BTREE ,
```

```
KEY `file_operate_time` (h) USING BTREE
```

优化方向:

1. 删除原来的id字段, 重新创建一个id bigint unsigned 的自增字段做主键
2. 重建过程中顺便删除第三个索引

方案1:先写主键索引,后续补普通索引

创建空表

```
CREATE TABLE `new_f` (  
  `id` bigint unsigned auto_increment ,  
  `a` varchar(36) DEFAULT ,  
  `b` varchar(36) DEFAULT ,  
  `c` varchar(36) DEFAULT ,  
  `d` varchar(36) DEFAULT ,  
  `e` varchar(50) DEFAULT ,  
  `f` varchar(100) DEFAULT ,  
  `g` datetime DEFAULT NULL COMMENT '行为操作时间',  
  `h` varchar(50) DEFAULT ,  
  `i` char(1) DEFAULT ,  
  `j` varchar(2500) DEFAULT ,  
  `k` varchar(500) DEFAULT ,  
  `l` varchar(500) DEFAULT ,  
  `m` varchar(30) DEFAULT ,  
  `n` varchar(255) DEFAULT ,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8  
ROW_FORMAT=DYNAMIC;
```

```
insert into new_f select null, a,b,c....n from f;
```

```
alter table new_f add KEY `ind_d` (`d`) USING BTREE ,  
add KEY `ind_hx` (h,n) USING BTREE ;
```

追问：有没有时间更短的方案？

额外信息：这个表 f 的操作模式只有 insert 且 uuid 递增，没有 update。

方案2：减少停机时间

创建空表

```
CREATE TABLE `new_f` (  
  `id` bigint unsigned auto_increment ,  
  `a` varchar(36) DEFAULT ,  
  `b` varchar(36) DEFAULT ,  
  `c` varchar(36) DEFAULT ,  
  `d` varchar(36) DEFAULT ,  
  `e` varchar(50) DEFAULT ,  
  `f` varchar(100) DEFAULT ,  
  `g` datetime DEFAULT NULL COMMENT '行为操作时间',  
  `h` varchar(50) DEFAULT ,  
  `i` char(1) DEFAULT ,  
  `j` varchar(2500) DEFAULT ,  
  `k` varchar(500) DEFAULT ,  
  `l` varchar(500) DEFAULT ,  
  `m` varchar(30) DEFAULT ,  
  `n` varchar(255) DEFAULT ,  
  PRIMARY KEY (`id`),  
  KEY `ind_d` (`d`) USING BTREE ,  
  KEY `ind_hx` (`h`,`n`) USING BTREE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8  
ROW_FORMAT=DYNAMIC;
```

```
select  max(id) from f; //得到M
```

```
insert into new_f select null, a,b,c....n from f where f<=M;
```

停机

```
insert into new_f select null, a,b,c....n from f where f>M;
```

```
rename  f to old_f;
```

```
rename new_f to f;
```

恢复服务

方案3：先并发写主键索引,后续补普通索引

创建空表

```
CREATE TABLE `new_f` (  
  `id` bigint unsigned auto_increment ,  
  `a` varchar(36) DEFAULT ,  
  `b` varchar(36) DEFAULT ,  
  `c` varchar(36) DEFAULT ,  
  `d` varchar(36) DEFAULT ,  
  `e` varchar(50) DEFAULT ,  
  `f` varchar(100) DEFAULT ,  
  `g` datetime DEFAULT NULL ,  
  `h` varchar(50) DEFAULT ,  
  `i` char(1) DEFAULT ,  
  `j` varchar(2500) DEFAULT ,  
  `k` varchar(500) DEFAULT ,  
  `l` varchar(500) DEFAULT ,  
  `m` varchar(30) DEFAULT ,  
  `n` varchar(255) DEFAULT ,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8  
ROW_FORMAT=DYNAMIC;
```


方案3：先并发写主键索引,后续补普通索引

1. 准备数据

```
select * from f order by id limit 5000000 into outfile file1;  
//tail file1 最后一行得到最大id M1  
select * from f order by id where id>M1 limit 5000000 into outfile file2;  
//tail file2 最后一行得到最大id M2  
....  
select * from f order by id where id>M6 limit 5000000 into outfile file2;  
  
// 修改file1文件, 把第一列修改成数字1~500W  
//修改file2文件, 把第一列修改成数字5000001~1000W  
.....
```

2. 多线程执行,每个线程执行load

```
load data file1 into ..  
load data file2 into ..  
...
```

3. 全部insert执行完成后

```
alter table new_f add KEY `ind_d` (`d`) USING BTREE ,  
add KEY `ind_hx` (h,n) USING BTREE ;
```

Q&A

THANKS