虚拟机热迁移性能优化方案

---张友加

目录

01 虚拟机热迁移功能介绍

02 高业务负载虚拟机热迁移遇到的挑战

03 虚拟机热迁移性能优化方案

04 优化方案总结

05 热迁移数据一致性校验方案

目 录

01 **虚拟机热迁移功能介绍**

02 高业务负载虚拟机热迁移遇到的挑战

03 虚拟机热迁移性能优化方案

04 优化方案总结

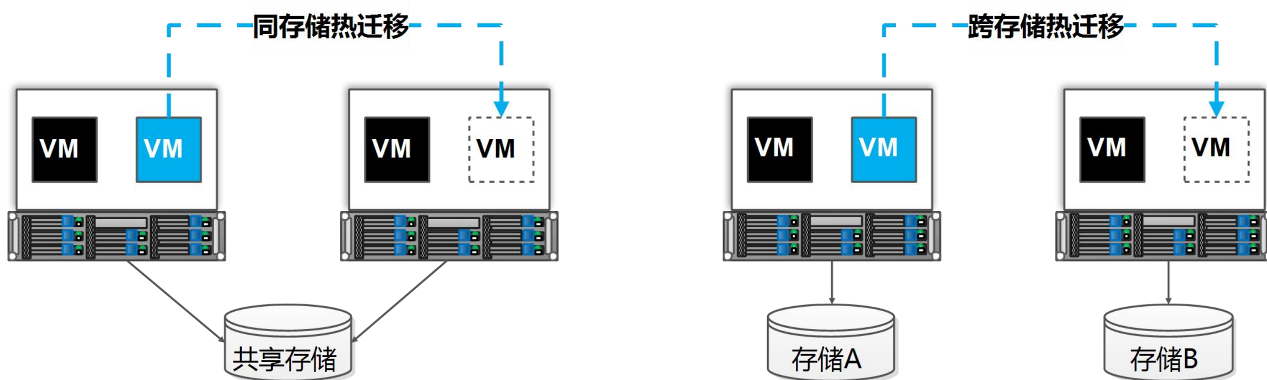
05 热迁移数据一致性校验方案

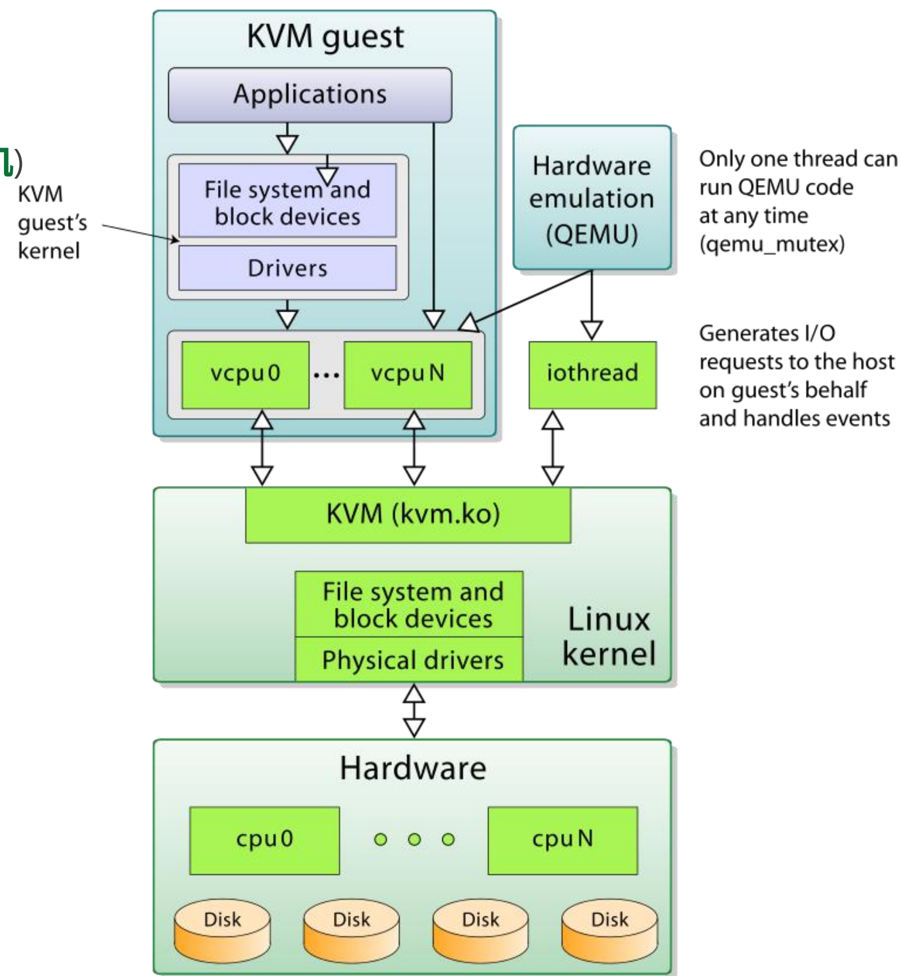
虚拟机热迁移功能介绍

CEC 中国电子 | CESTC 中国系统

功能概述

- 热迁移(Live Migration)是指把虚拟机从一台宿主机(源宿主机)迁移到另一台宿主机(目的宿主机)
- 热迁移过程中，虚拟机业务不会中断；在最后的停机迁移阶段，虚拟机业务会有短暂的卡顿
- 热迁移完成后，虚拟机在目的宿主机上继续运行，用户不会察觉到任何差异
- 热迁移分为同存储热迁移和跨存储热迁移两种类型





虚拟机热迁移功能介绍

价值分析

随着云计算产业的发展，越来越多的高级云业务特性基于虚拟机热迁移功能来实现，虚拟机热迁移功能也逐渐成为虚拟化的一个核心基础功能。

- DRS(分布式资源调度)
- 主机亚健康处置
- 主机停机维护
- 主机滚动热升级
- QEMU热升级
- ……

虚拟机热迁移功能介绍

CEC 中国电子 | CESTC 中国系统

技术原理-热迁移流程

- 虚拟机热迁移流程开始时，会在目的宿主机上启动目的虚拟机
- 源虚拟机与目的虚拟机建立TCP连接，然后进行运行时数据的迁移
- 热迁移过程会涉及到如下几种运行时数据的迁移：
  - 虚拟机的内存数据
  - 虚拟机的磁盘数据(可选，跨存储热迁移时需要)
  - 虚拟机的硬件状态数据(CPU、网卡、显卡等)
- 虚拟机的内存和磁盘数据需要进行多轮迭代迁移，使得剩余数据量逐渐减少
- 当剩余数据量能够在一个可以设定的downtime停机时间内迁移完成时，会暂停源虚拟机，将剩余的数据一次性迁移到目的虚拟机
- 最后，关闭源虚拟机，激活目的虚拟机，至此虚拟机的热迁移操作就完成了

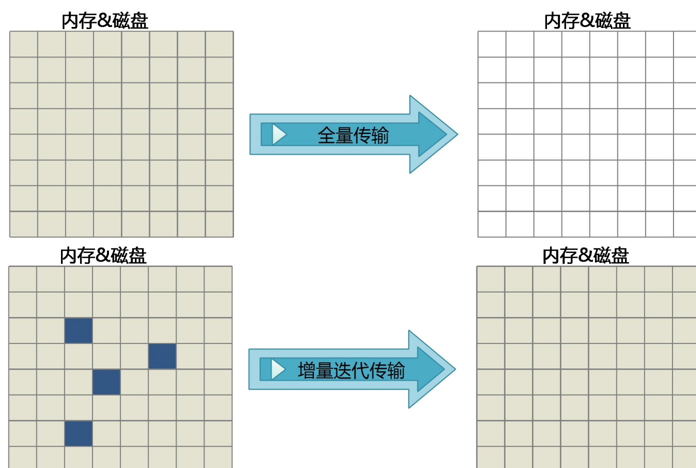
虚拟机热迁移功能介绍

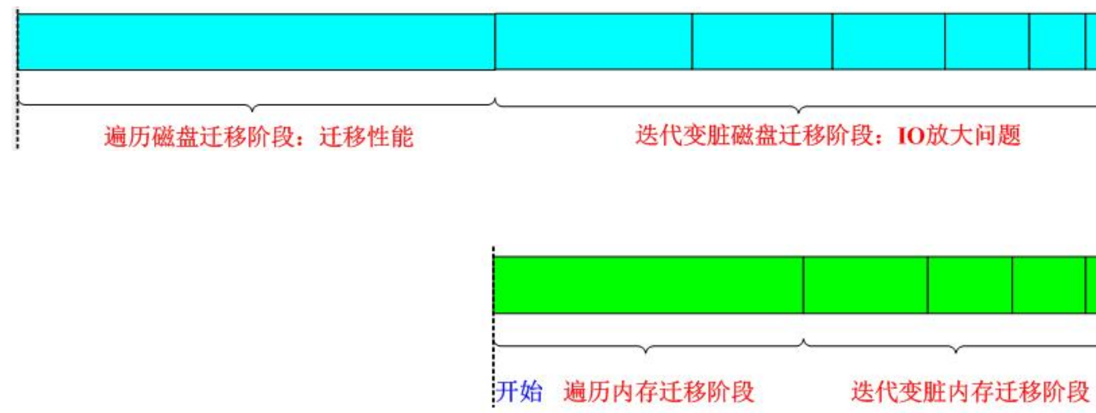
CEC 中国电子 | CESTC 中国系统

技术原理-内存和磁盘数据的迭代迁移

由于虚拟机的内存和磁盘数据量比较大，且在不断变化，无法一次性全部迁移完成，因此需要进行多轮的增量迭代迁移，整个过程可分为三个阶段：

- **全量迁移阶段：**源虚拟机将所有的内存和磁盘的BITMAP都标记脏，然后全量迁移内存和磁盘数据到目的虚拟机。在此期间，源虚拟机在迁移过程中会监控内存和磁盘变化情况，并记录在BITMAP，当所有内存和磁盘数据都迁移完成后，会开始进入**增量迭代迁移阶段**。
- **增量迭代迁移阶段：**源虚拟机持续不断迭代发送内存和磁盘的增量脏数据，在每轮迭代来检查BITMAP有多少数据被标记脏，根据预估带宽和当前迭代的脏数据量，可以计算出花费多久来迁移剩余脏数据，如果在可接受或者设置的downtime停机时间限制内，就会进入**停机迁移阶段**。否则，会继续停留在**增量迭代迁移阶段**。





- **停机迁移阶段：**暂停源虚拟机，将剩余内存和磁盘脏数据一次性迁移过去，然后将所有硬件状态数据迁移过去，最后目的虚拟机被激活运行，源虚拟机被关闭。

目录

01 虚拟机热迁移功能介绍

02 **高业务负载虚拟机热迁移遇到的挑战**

03 虚拟机热迁移性能优化方案

04 优化方案总结

05 热迁移数据一致性校验方案

高业务负载虚拟机热迁移遇到的挑战

问题1：原生QEMU热迁移压缩算法效率低下

原生QEMU热迁移内存数据使用的压缩算法是zlib，其存在如下问题，难以满足高负载场景的需求

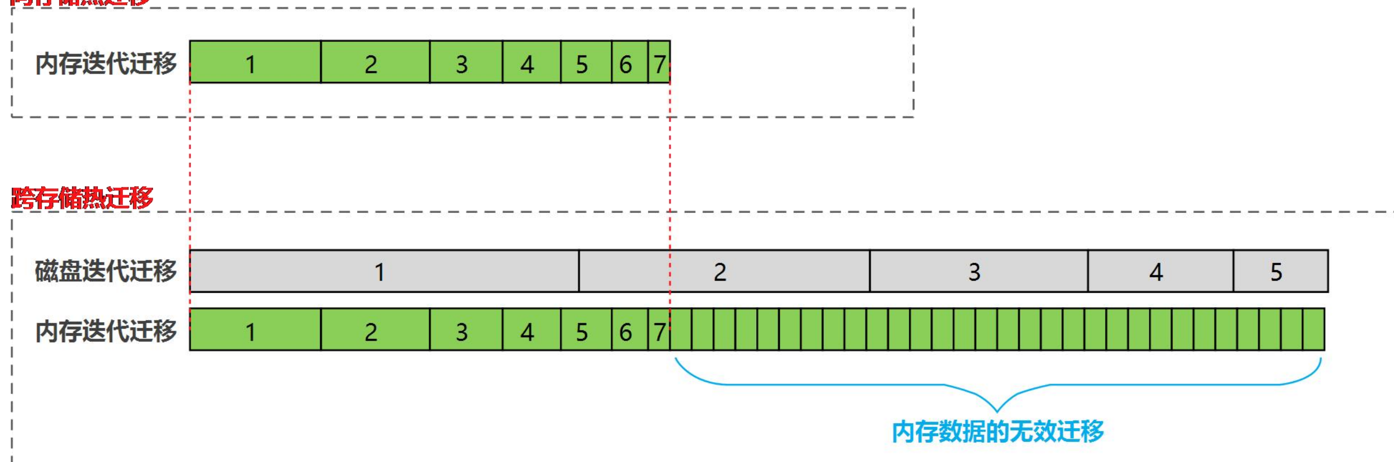
- 单核的压缩性能只有**100MB/s （但压缩率相对较高）**
- 跑满万兆带宽，QEMU压缩线程的CPU消耗高达**1000% （10多个CPU核，压缩线程：解压线程 = 8:2）**

```
[root@worker1 /]# virsh qemu-monitor-command 1 --hmp "info migrate_capabilities"
xbzrle: off
rdma-pin-all: off
auto-converge: off
zero-blocks: off
compress: off
events: on
postcopy-ram: off
x-colo: off
release-ram: off
return-path: off
pause-before-switchover: off
multifd: off
dirty-bitmaps: off
postcopy-blocktime: off
late-block-activate: off
x-ignore-shared: off
validate-uuid: off
background-snapshot: off
```

高业务负载虚拟机热迁移遇到的挑战

问题2：内存和磁盘的热迁移时间片分配问题而导致内存数据的无效迁移

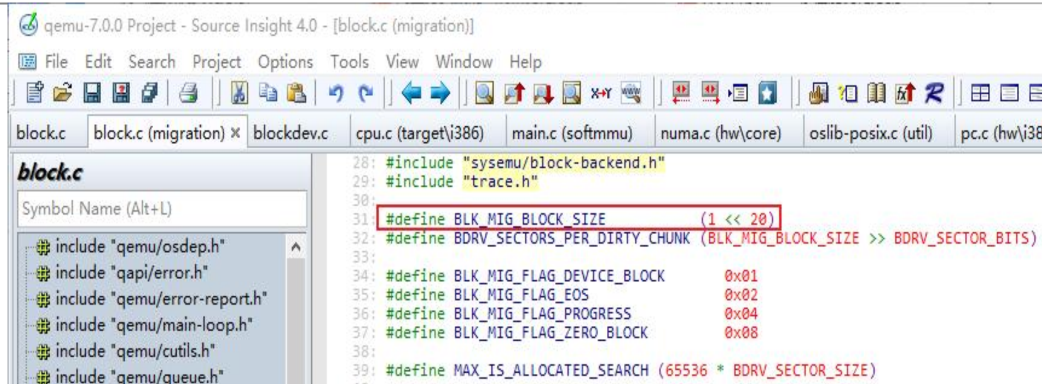
- 虚拟机在进行**跨存储热迁移**时，虚拟机的内存和磁盘两种对象是存在着较大的差异
  - 内存是一个快速设备，其迁移速率和脏数据生成速率可以达到**1GB/s**的级别
  - 磁盘是一个慢速设备，其迁移速率和脏数据生成速率是在**100MB/s**级别
  - 磁盘的总数据量通常是内存是**10倍**左右
- 由于磁盘数据量大，并且迁移速率慢，因此**跨存储热迁移**的整体耗时**同存储热迁移**的几十倍
- 原生QEMU进行跨存储热迁移，进入到增量迭代迁移阶段，内存和磁盘数据是并行迭代迁移，由于对内存热迁移的时间片分配不合理，会带来如下问题：
  - 内存数据热迁移的增量迭代次数会大幅增加(**举例：同存储热迁移场景下进行7次迭代，跨存储热迁移场景可能会变为上百次**)
  - 内存的脏数据生成速率很快，增量迭代次数的大幅增加，导致大量内存数据的无效迁移



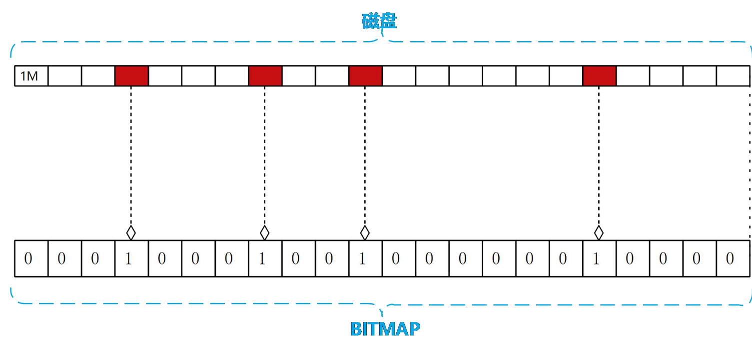
高业务负载虚拟机热迁移遇到的挑战

问题3：磁盘热迁移脏数据放大

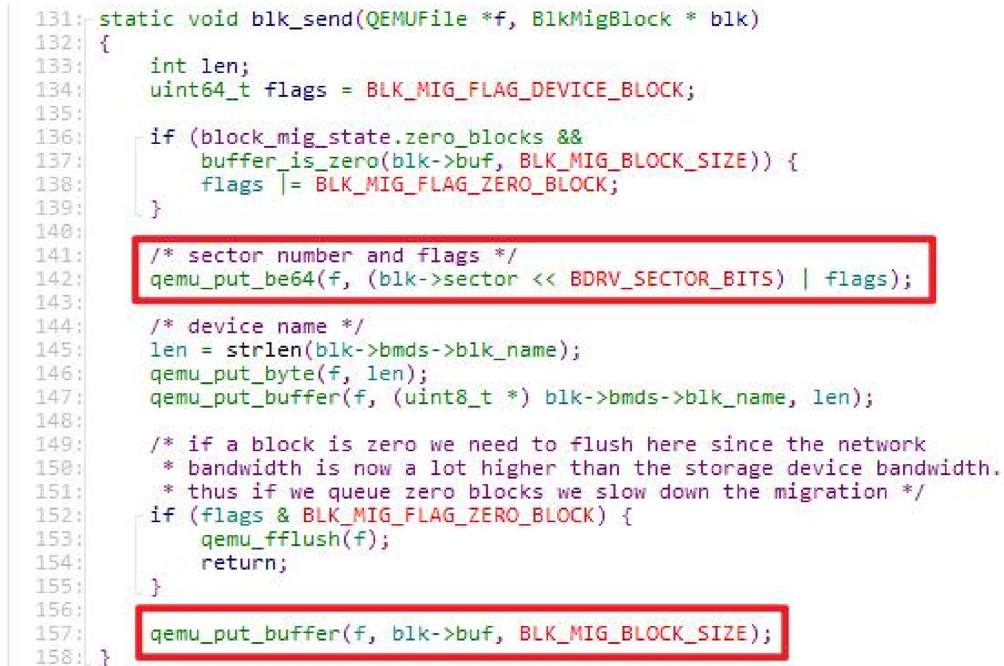
（原生qemu热迁移磁盘，协议设计存在问题：没有传数据大小--被宏定义固定）



- QEMU进行跨存储热迁移，在迭代迁移磁盘数据阶段，通过BITMAP记录磁盘脏数据
- QEMU中记录磁盘BITMAP对应数据块的粒度是1M
- 假设虚拟磁盘的某个数据块有4K数据发生修改，BITMAP会将整个数据块(**1M大小**)标记为脏，需要迁移整个数据块的数据，导致磁盘脏数据迁移放大问题



fio块大小	fio写速率	磁盘脏数据生成速率	磁盘脏数据放大率
4K	2MB/s	380 MB/s	190
8K	**2MB/s**	**185 MB/s**	**92.5**
16K	2MB/s	96 MB/s	48
32K	2MB/s	51 MB/s	25.5
64K	2MB/s	24 MB/s	12
128K	2MB/s	11 MB/s	5.5
256K	2MB/s	4 MB/s	2

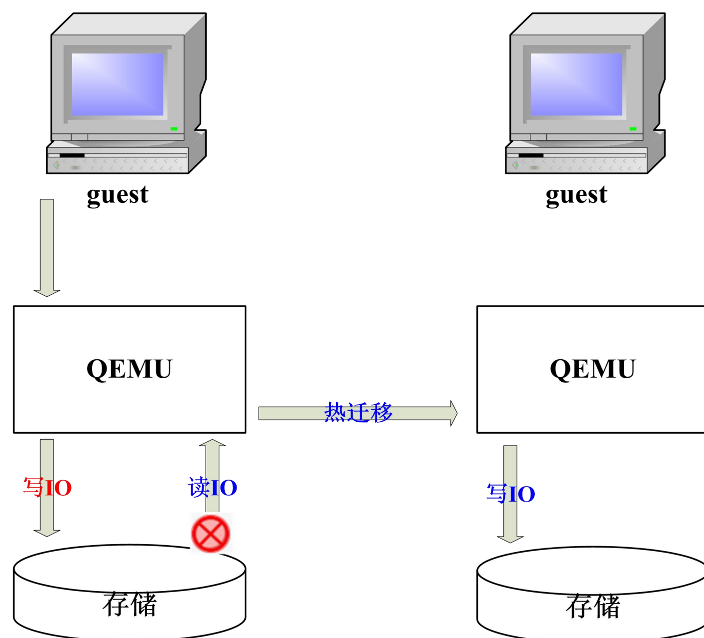


高业务负载虚拟机热迁移遇到的挑战

CEC 中国电子 | CESTC 中国系统

问题4：存储性能瓶颈导致磁盘数据无法迁完

- QEMU进行跨存储热迁移，在迁移磁盘数据时，需要将数据从存储读上来
- **当存储的IO压力较大，读操作就会遇到性能瓶颈**
- QEMU的CPU节流机制，对于IO密集型的业务效果不好，虚拟机的写速率并不会呈线性下降
- 当磁盘脏数据的生成速率超过存储的读速率时，磁盘数据就永远无法迁移完



高业务负载虚拟机热迁移遇到的挑战

CEC 中国电子 | CESTC 中国系统

问题5：磁盘热迁移流程存在CPU空耗

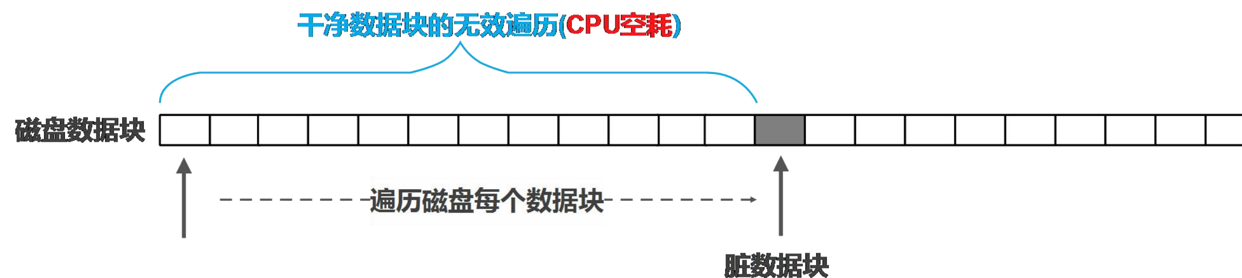
- QEMU进行**跨存储热迁移**，在增量迭代迁移阶段，会遍历磁盘每个数据块，检测其状态是否为脏
- 在磁盘热迁移进入到最后阶段，如果磁盘数据块总量比较多，而剩余脏数据块量比较少时，热迁移线程的CPU大量消耗在**干净数据块的无效遍历**
- 在极端场景下的实测结果，这里会导致**60%以上**的CPU额外消耗

```
static int mig_save_device_dirty(QEMUFile *f, BlkMigDevState *bmds,
                                 int is_async)
{
    BlkMigBlock *blk;
    int64_t total_sectors = bmds->total_sectors;
    int64_t sector;
    int nr_sectors;
    int ret = -EIO;
    size_t mmap_size = 0;

    for (sector = bmds->cur_dirty; sector < bmds->total_sectors;) {
        blk_mig_lock();
        if (bmds_aio_inflight(bmds, sector)) {
            blk_mig_unlock();
            blk_drain(bmds->blk);
        } else {
            blk_mig_unlock();
        }
        bdrv_dirty_bitmap_lock(bmds->dirty_bitmap);
        if (bdrv_dirty_bitmap_get_locked(bmds->dirty_bitmap,
                                         sector * BDRV_SECTOR_SIZE)) {
            if (total_sectors - sector < BDRV_SECTORS_PER_DIRTY_CHUNK) {
                nr_sectors = total_sectors - sector;
            } else {
                nr_sectors = BDRV_SECTORS_PER_DIRTY_CHUNK;
            }
```

遍历磁盘每个数据块

检测数据块状态是否为脏



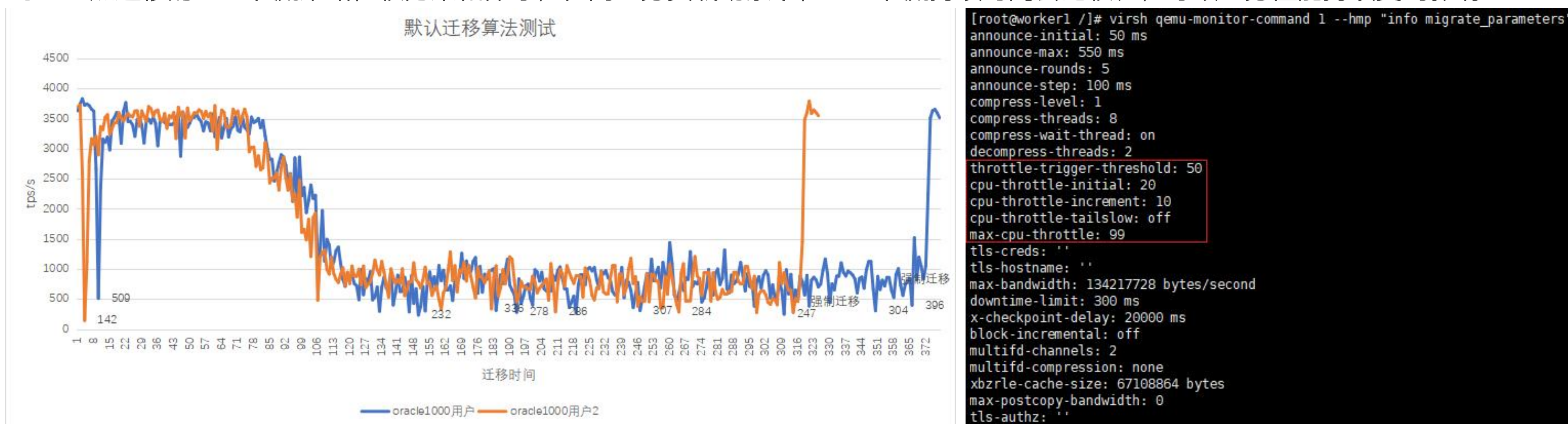
高业务负载虚拟机热迁移遇到的挑战

CEC 中国电子 | CESTC 中国系统

问题6：原生QEMU的CPU节流策略导致业务性能受影响时间较长

虚拟机热迁移进入到增量迭代迁移阶段时，如果内存脏数据的生成速率大于迁移速率时，内存数据永远无法完成迁移，原生QEMU提供了CPU节流机制来尝试解决此问题。

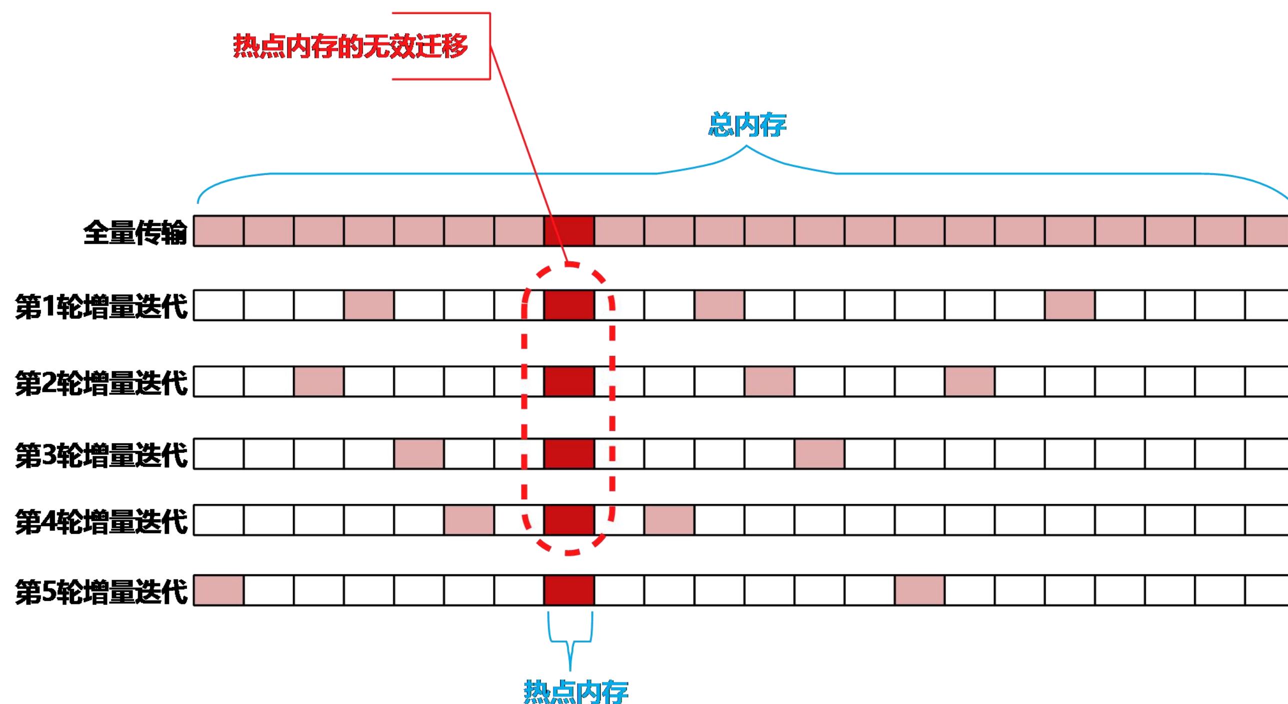
- 原生QEMU的CPU节流运行机制如下：
    - 如果本轮内存脏数据量超过上轮迁移的50%，QEMU会开始对虚拟机进行CPU节流，先削减20%的vCPU执行时间
    - 在每两轮内存迭代迁移开始时，QEMU会根据之前的削减情况，决定后续削减粒度（**削减步进**）
    - 如果新增脏数据仍然过多，QEMU会继续削减10%的vCPU执行时间
    - QEMU最高会削减99%的vCPU执行时间，期望进入**停机迁移阶段**，以完成迁移过程
- 原生QEMU热迁移的CPU节流策略，较为柔和保守，在高业务负载场景下，CPU节流持续时间会比较长，导致业务性能持续受到影响



高业务负载虚拟机热迁移遇到的挑战

问题7：热点内存的无效迁移

- ➢ 根据内存局部性原理，虚拟机业务在运行过程中，会存在大量的热点内存，在短时间内被频繁的访问和修改
- ➢ 虚拟机热迁移进入到增量迭代迁移阶段时，热点内存由于频繁修改，在每轮迭代都会变脏而重复迁移
- ➢ 以下图为例，热点内存在5轮的增量迭代中被重复迁移了5次，而前面4轮其实属于无效迁移，导致带宽和CPU资源的浪费



目录

01 虚拟机热迁移功能介绍

02 高业务负载虚拟机热迁移遇到的挑战

03 **虚拟机热迁移性能优化方案**

04 优化方案总结

05 热迁移数据一致性校验方案

虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

问题1：原生QEMU热迁移压缩算法效率低下

原生QEMU热迁移内存数据使用的压缩算法是zlib，其存在如下问题，难以满足高负载场景的需求

- 单核的压缩性能只有**100MB/s （但压缩率相对较高）**
- 跑满万兆带宽，QEMU压缩线程的CPU消耗高达**1000% （10多个CPU核）**

备选解决方案：

- 优化zlib算法的性能，**效果有限**
- 压缩算法卸载到硬件(加速卡、QAT等)，**成本高、平台受限(arm?)**
- **替换效率更高的压缩算法**

开源社区新进展：

https://patchew.org/QEMU/20231018221224.599065-1-yuan1.liu@intel.com/

[PATCH 0/5] Live Migration Acceleration with IAA Compression

【压缩性能测试对比实例】

```
[root@chroot <host-6c92bfd3e1b6> /home/zyj/lz4 ]# ./compress_test oracle.data 0
begin compress time=1630305167 size=8192 compress_type: zlib(Z_DEFAULT_COMPRESSION)
end compress time=1630305567, cost=400(seconds), compress_size=3737

[root@chroot <host-6c92bfd3e1b6> /home/zyj/lz4 ]# ./compress_test oracle.data 1
begin compress time=1630295523 size=8192 compress_type: zlib(Z_BEST_COMPRESSION)
end compress time=1630295954, cost=431(seconds), compress_size=3732

[root@chroot <host-6c92bfd3e1b6> /home/zyj/lz4 ]# ./compress_test oracle.data 2
begin compress time=1630305586 size=8192 compress_type: zlib(Z_BEST_SPEED)
end compress time=1630305875, cost=289(seconds), compress_size=3940

[root@chroot <host-6c92bfd3e1b6> /home/zyj/lz4 ]# ./compress_test oracle.data 3
begin compress time=1630305981 size=8192 compress_type: lz4_default
end compress time=1630305997, cost=16(seconds), compress_size=4834

[root@chroot <host-6c92bfd3e1b6> /home/zyj/lz4 ]# ./compress_test oracle.data 4
begin compress time=1630306005 size=8192 compress_type: lz4_fast4
end compress time=1630306014, cost=9(seconds), compress_size=5288

[root@chroot <host-6c92bfd3e1b6> /home/zyj/lz4 ]# ./compress_test oracle.data 5
begin compress time=1630306022 size=8192 compress_type: lz4_fast8
end compress time=1630306028, cost=6(seconds), compress_size=5897

[root@chroot <host-6c92bfd3e1b6> /home/zyj/lz4 ]# ./compress_test oracle.data 6
begin compress time=1630306037 size=8192 compress_type: lz4_fast16
end compress time=1630306040, cost=3(seconds), compress_size=6462
```

图3：lz4与zlib性能测试对比数据

虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

方案1：QEMU热迁移压缩算法替换为lz4

➢ 方案概述：

- 将原生QEMU热迁移使用的zlib压缩算法替换为性能更高的lz4压缩算法

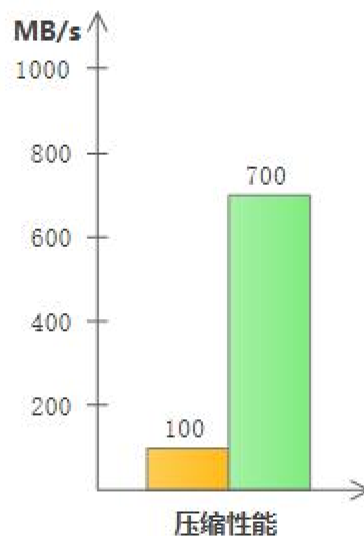
➢ 实现效果：

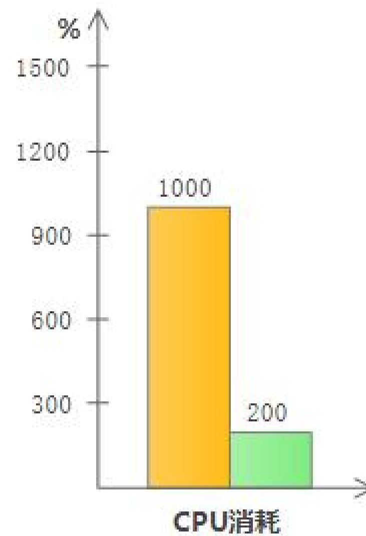
- 单核的压缩性从**100MB/s提升到700MB/s**
- 跑满万兆带宽，QEMU压缩线程的CPU消耗从**1000%降低到200%（压缩线程：解压线程 = 8:2）**
- 压缩效果有一些降低，压缩率从**36%升高到47%**

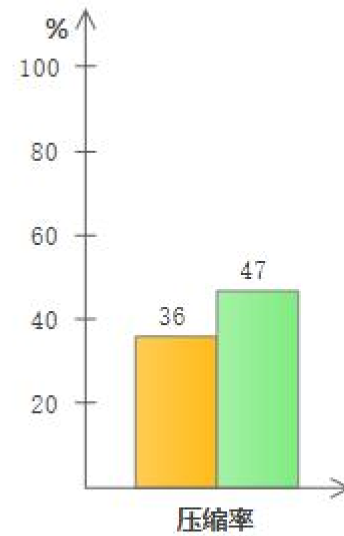
```
[root@worker1 /]# virsh qemu-monitor-command 1 --hmp "info migrate_capabilities"
xbzrle: off
rdma-pin-all: off
auto-converge: off
zero-blocks: off
compress: off
events: on
postcopy-ram: off
x-colo: off
release-ram: off
return-path: off
pause-before-switchover: off
multifd: off
dirty-bitmaps: off
postcopy-blocktime: off
late-block-activate: off
x-ignore-shared: off
validate-uuid: off
background-snapshot: off
lz4: off
```

https://github.com/lz4/lz4

Compressor	Ratio	Compression	Decompression
memcpy	1.000	13700 MB/s	13700 MB/s
LZ4 default (v1.9.0)	**2.101**	**780 MB/s**	**4970 MB/s**
LZO 2.09	2.108	670 MB/s	860 MB/s
QuickLZ 1.5.0	2.238	575 MB/s	780 MB/s
Snappy 1.1.4	2.091	565 MB/s	1950 MB/s
Zstandard 1.4.0 -1	2.883	515 MB/s	1380 MB/s
LZF v3.6	2.073	415 MB/s	910 MB/s
zlib deflate 1.2.11 -1	2.730	100 MB/s	415 MB/s
LZ4 HC -9 (v1.9.0)	**2.721**	**41 MB/s**	**4900 MB/s**
zlib deflate 1.2.11 -6	3.099	36 MB/s	445 MB/s



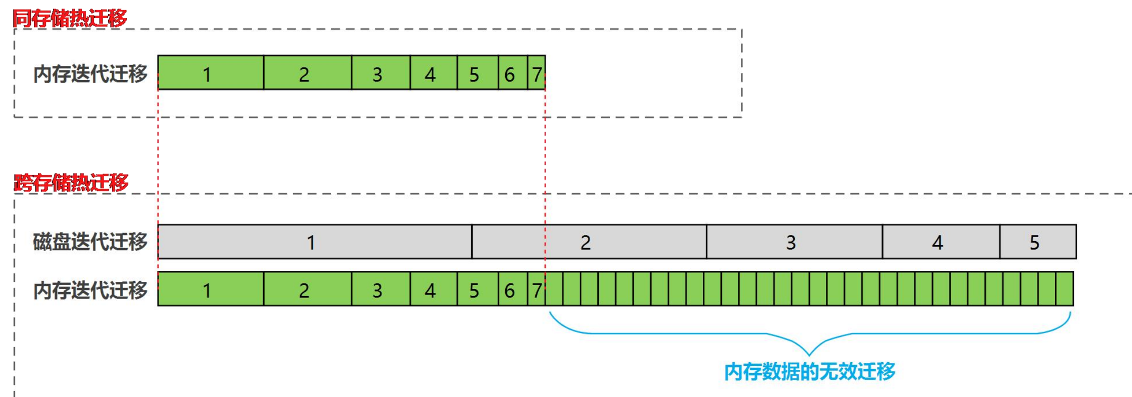




虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

问题2：内存和磁盘的热迁移时间片分配问题而导致内存数据的无效迁移




```
=========== migrate_parameters ===========
announce-initial: 50 ms
announce-max: 550 ms
announce-rounds: 5
announce-step: 100 ms
compress-level: 1
compress-threads: 4
compress-wait-thread: off
decompress-threads: 4
throttle-trigger-threshold: 50
cpu-throttle-initial: 20
cpu-throttle-increment: 10
cpu-throttle-value: 0
max-cpu-throttle: 99
tls-creds: ''
tls-hostname: ''
max-bandwidth: 8589934592 bytes/second
downtime-limit: 1000 milliseconds
x-checkpoint-delay: 20000
block-incremental: off
multifd-channels: 2
multifd-compression: none
lz4-acceleration: 8
ram-save-max-wait: 50
ram-delay-save-percent: 30
bypass-compression: off
mig-block-size: 65536
xbzrle-cache-size: 67108864
max-postcopy-bandwidth: 0
tls-authz: ''
```

方案2：内存热迁移时间片动态调整机制

➢ 内存热迁移时间片动态调整机制，方案概述：

- 对内存热迁移设置可动态调整的时间片，调整范围是0~1024，0表示**不迁**内存，数值越大，所占的时间比例越高
- QEMU进行跨存储热迁移，内存热迁移时间片的动态调整策略如下：
  - 在磁盘数据全量迁移阶段，内存热迁移时间片设为0，不进行任何内存数据的迁移操作
  - 在磁盘数据增量迭代迁移阶段，内存热迁移时间片初始设为最小值1，随着磁盘剩余脏数据的减少，动态提高内存时间片

➢ 实现效果：

- 内存迭代迁移的次数大幅减少，避免了多轮迭代的无效内存迁移
- 在整体迁移耗时不变的情况下，内存数据的迁移量有明显的降低

虚拟机热迁移性能优化方案

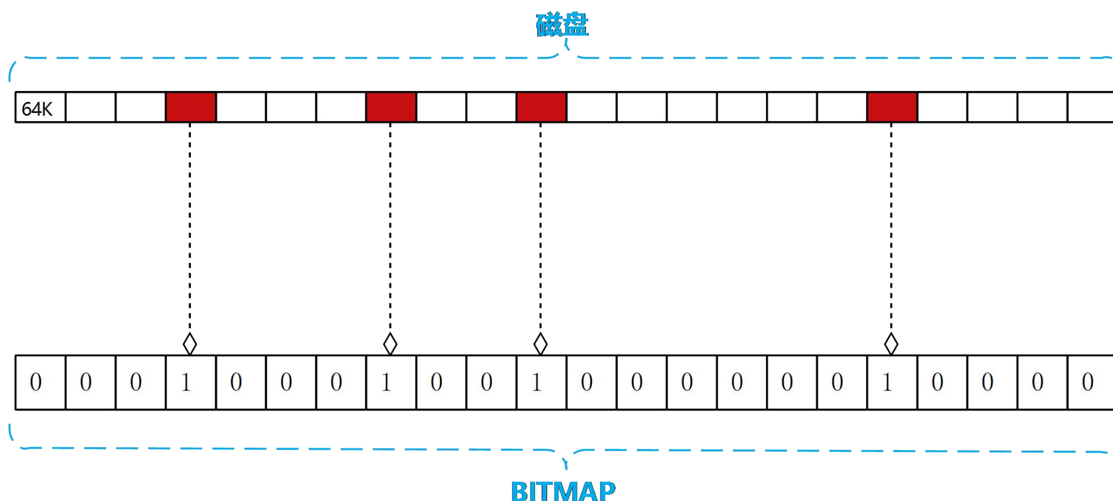
CEC 中国电子 | CESTC 中国系统

问题3：磁盘热迁移脏数据放大

方案3：磁盘热迁移BITMAP数据块粒度调整

➢ 磁盘热迁移BITMAP数据块粒度调整机制，方案概述：

- BITMAP数据块粒度做成迁移前动态可配置
- BITMAP数据块粒度默认值从**1M改为64K**
- 在热迁移开始前，可以根据虚拟机业务类型来动态调整BITMAP数据块粒度




```
========== migrate_parameters ==========
announce-initial: 50 ms
announce-max: 550 ms
announce-rounds: 5
announce-step: 100 ms
compress-level: 1
compress-threads: 4
compress-wait-thread: off
decompress-threads: 4
throttle-trigger-threshold: 50
cpu-throttle-initial: 20
cpu-throttle-increment: 10
cpu-throttle-value: 0
max-cpu-throttle: 99
tls-creds: ''
tls-hostname: ''
max-bandwidth: 8589934592 bytes/second
downtime-limit: 1000 milliseconds
x-checkpoint-delay: 20000
block-incremental: off
multifd-channels: 2
multifd-compression: none
lz4-acceleration: 8
ram-save-max-wait: 50
ram-delay-save-percent: 30
bypass-compression: off
mig-block-size: 65536
xbzrle-cache-size: 67108864
max-postcopy-bandwidth: 0
tls-authz: ''
```

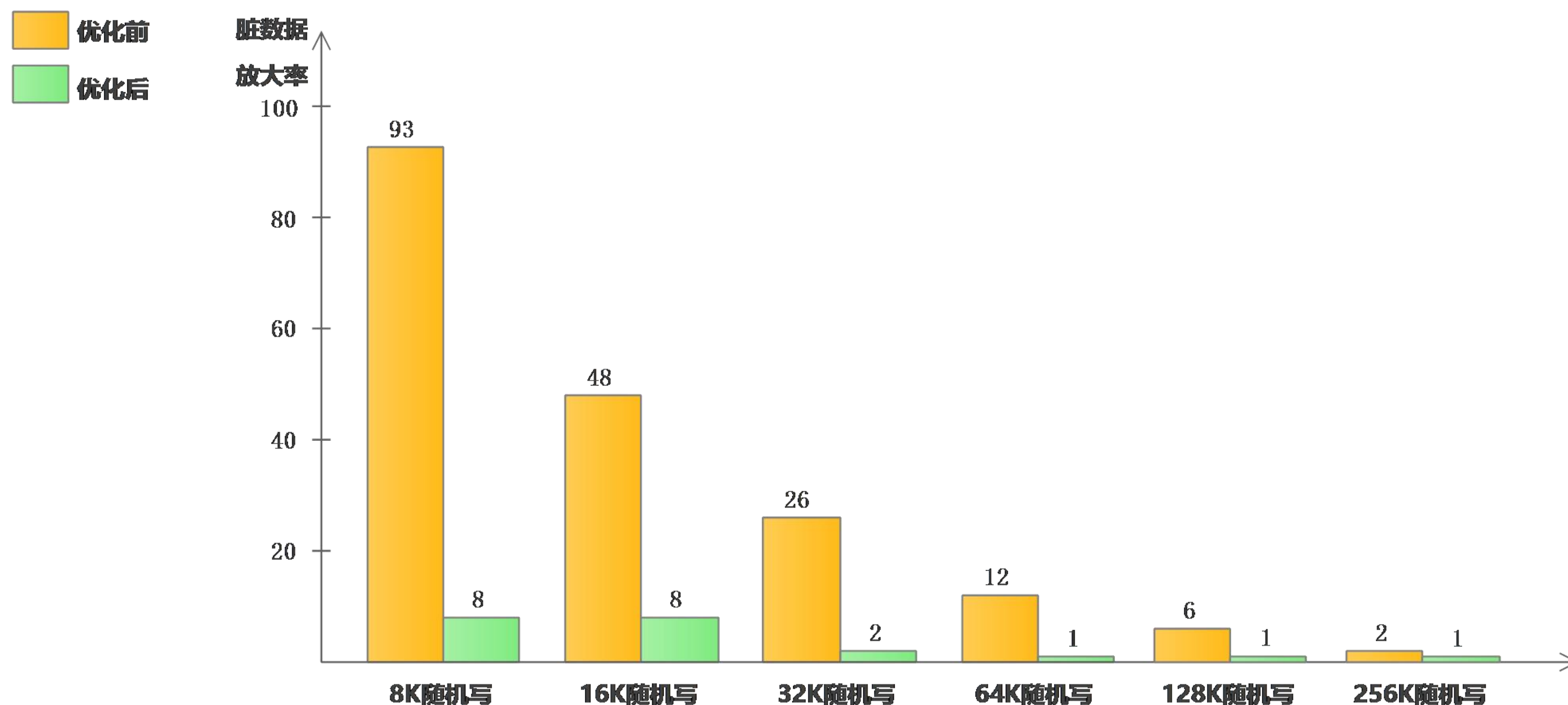
fio块大小	fio写速率	磁盘脏数据生成速率	磁盘脏数据放大率
4K	2MB/s	29 MB/s	14.5
8K	**2MB/s**	**15.2 MB/s**	**7.6**
16K	2MB/s	7.2 MB/s	3.6
32K	2MB/s	3.7 MB/s	1.9
64K	2MB/s	2 MB/s	1
128K	2MB/s	2 MB/s	1
256K	2MB/s	2 MB/s	1

虚拟机热迁移性能优化方案

方案3：磁盘热迁移BITMAP数据块粒度调整

➢ 实现效果：

- 数据块大小在256K以内的随机写场景，磁盘脏数据放大率有明显的下降
- ORACLE数据库场景(8K随机写IO模型)，磁盘脏数据放大率**从93下降到8**



虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

问题4：存储性能瓶颈导致磁盘数据无法迁完

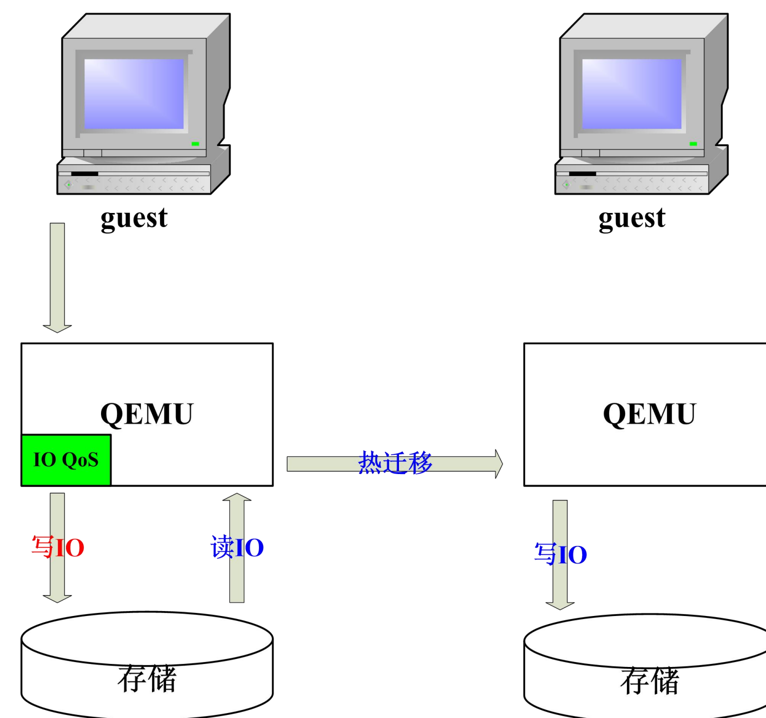
方案4：磁盘热迁移IO QoS动态调整机制

➢ 磁盘热迁移IO QoS动态调整机制，方案概述：

- 磁盘热迁移进入增量迭代迁移阶段，每轮迭代前会计算磁盘脏数据量
- 如果本轮迭代需要迁移的磁盘脏数据量相比以上一轮没有明显减少，就对磁盘写操作进行IO QoS限速
- 如果每轮迭代的脏数据量相比以上一轮仍然没有明显减少，就持逐步降低磁盘写操作的IO QoS限速值

➢ 实现效果：

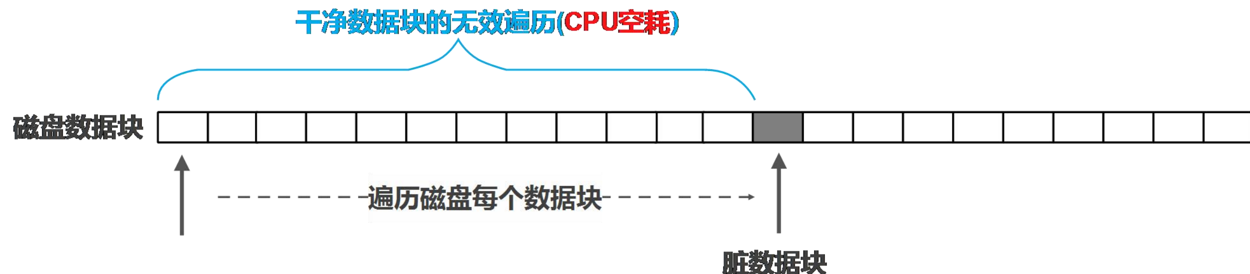
- 之前由于存储性能瓶颈，而无法热迁移完的场景，现在基本都可以完成迁移
- 之前由于虚拟机业务IO压力较大，而无法热迁移完的场景，现在基本都可以完成迁移



虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

问题5：磁盘热迁移流程存在CPU空耗




```
static int mig_save_device_dirty(QEMUFile *f, BlkMigDevState *bmds,
                                 int is_async)
{
    BlkMigBlock *blk;
    int64_t total_sectors = bmds->total_sectors;
    int64_t sector;
    int nr_sectors;
    int ret = -EIO;
    size_t mmap_size = 0;

    for (sector = bmds->cur_dirty; sector < bmds->total_sectors;) {
        blk_mig_lock();
        if (bmds_aio_inflight(bmds, sector)) {
            blk_mig_unlock();
            blk_drain(bmds->blk);
        } else {
            blk_mig_unlock();
        }
        bdrv_dirty_bitmap_lock(bmds->dirty_bitmap);
        if (bdrv_dirty_bitmap_get_locked(bmds->dirty_bitmap,
                                         sector * BDRV_SECTOR_SIZE)) {
            if (total_sectors - sector < BDRV_SECTORS_PER_DIRTY_CHUNK) {
                nr_sectors = total_sectors - sector;
            } else {
                nr_sectors = BDRV_SECTORS_PER_DIRTY_CHUNK;
            }
```

遍历磁盘每个数据块

检测数据块状态是否为脏

方案5：磁盘热迁移时脏数据块的检测机制优化

➢ 方案概述：

- 磁盘热迁移时，对脏数据块的检测方式，由遍历检测改为直接对脏数据块进行**多级检索**
- **多级检索**复用原生QEMU的HBitmap能力

➢ 实现效果：

- 磁盘热迁移，每轮增量迭代总耗时**下降20%**
- 热迁移线程的CPU消耗**下降60%**

```
static int mig_save_device_dirty(QEMUFile *f, BlkMigDevState *bmds,
                                 int is_async)
{
    BlkMigBlock *blk;
    int64_t total_sectors = bmds->total_sectors;
    int64_t sector = bmds->cur_dirty;
    int nr_sectors;
    int i, ret = -EIO;
    size_t mmap_size = 0;
    bool aio_inflight = false;
    int64_t dirty_start = 0;
    int64_t dirty_count = 0;

    //Improvement: by merging device dirty.
    if (bdrv_dirty_bitmap_next_dirty_area(bmds->dirty_bitmap, sector * BDRV_SECTOR_SIZE,
                                          INT64_MAX, SF_BLK_MIG_BLOCK_SIZE,
                                          &dirty_start, &dirty_count)) {
        sector = dirty_start >> BDRV_SECTOR_BITS;
        if (total_sectors - sector < (dirty_count >> BDRV_SECTOR_BITS)) {
            nr_sectors = total_sectors - sector;
        } else {
            nr_sectors = dirty_count >> BDRV_SECTOR_BITS;
        }
        bdrv_reset_dirty_bitmap_locked(bmds->dirty_bitmap,
                                       sector * BDRV_SECTOR_SIZE,
                                       nr_sectors * BDRV_SECTOR_SIZE);
        bdrv_dirty_bitmap_unlock(bmds->dirty_bitmap);
```

虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

问题6：原生QEMU的CPU节流策略导致业务性能受影响时间较长

方案6：CPU节流策略优化

➢ 在热迁移过程中引入**直接节流**和**阶段式节流**两种节流方式，用来替换QEMU默认的节流策略，方案概述：

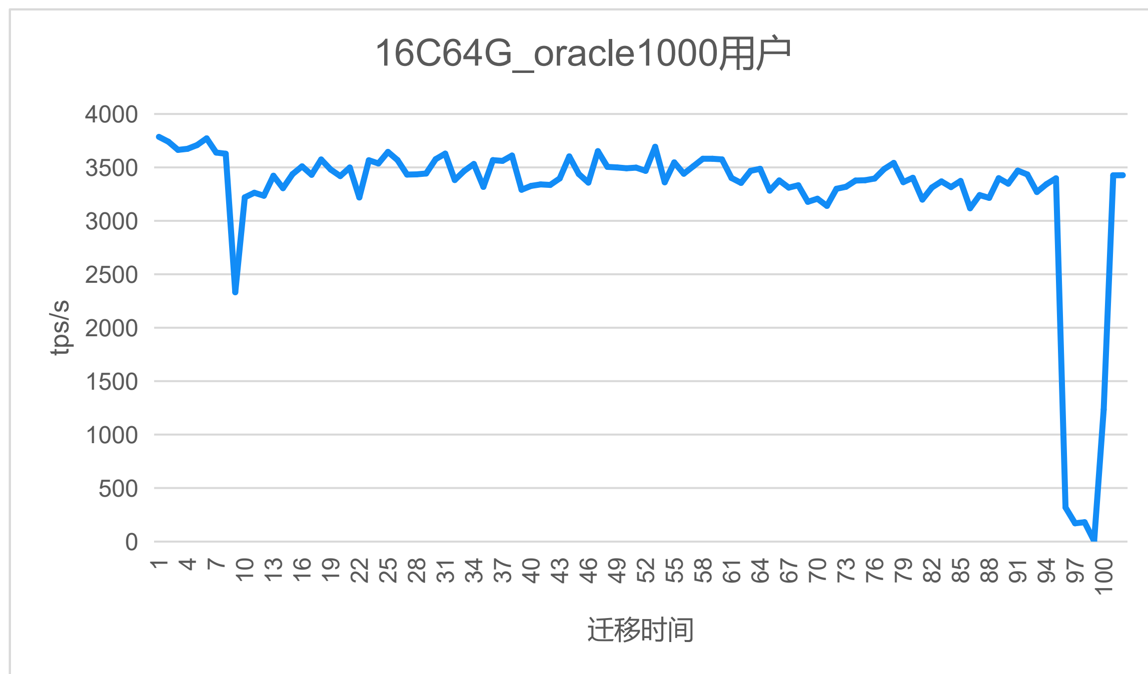
- 循环判断前后两轮脏页变化率是否小于5%，如果是就进入节流模式，否则继续迭代迁移
- 判断条件***剩余脏页 < 平均网络迁移速率 * 5s***是否满足，如果满足就进入**直接节流**，否则进入**阶段式节流**
- **直接节流**的策略为：
  - 90%节流持续3s
  - 99%节流持续3s
  - 如果经过以上**直接节流**仍然无法达到downtime停机条件，这种情况大概率是迁移不完了，后面会进入**阶段式节流**
- **阶段式节流**的策略为：
  - 每一轮都是递进式的逐步增强节流，计算公式为：***节流比例：1-(（平均网络带宽 * 0.2）/ 脏页生成速率 * 上一轮节流比例)***
  - 如果计算的节流比例超过99%，则只执行3s，最后提示用户强制进行切换

虚拟机热迁移性能优化方案

方案6：CPU节流策略优化

➢ Oracle虚拟机，1000用户跑3000 tps的压力测试场景，实现效果：

- 原本无法迁移完的场景，现在能够成功迁移完
- 热迁移过程中，进入CPU节流之前，性能影响5%以内
- 热迁移最后downtime停机阶段，ping网络中断时间**不超过1秒**
- 在热迁移过程中，oracle业务性能低于100 TPS的时间控制在**3s以内**



20:08:48	[1000/100	204833	3215	128
20:08:49	[1000/100	204789	3399	128
20:08:50	[1000/100	204560	3348	128
20:08:51	[1000/100	204401	3470	128
20:08:52	[1000/100	204453	3434	128
20:08:53	[1000/100	204255	3270	128
20:08:54	[1000/100	204066	3343	128
20:08:55	[1000/100	204146	3398	128
20:08:56	[1000/100	198678	318	128
20:08:57	[1000/100	195237	170	128
20:08:59	[1000/100	192126	180	128
20:09:01	[1000/100	188803	4	128
20:09:02	[1000/100	186696	1234	129
20:09:03	[1000/100	186786	3426	129
20:09:04	[1000/100	186816	3427	129

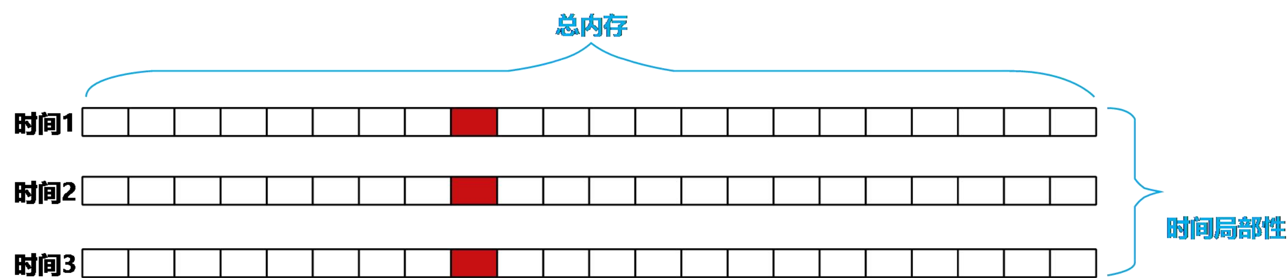
虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

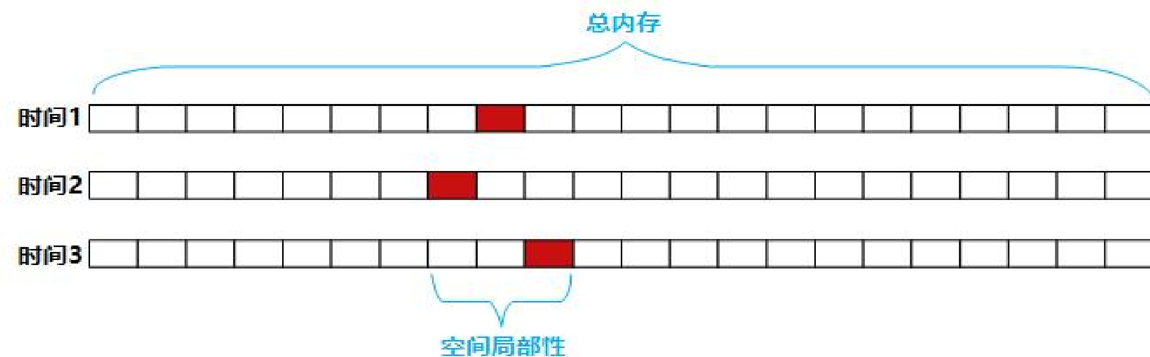
问题7：热点内存的无效迁移

方案7：热点内存脏数据延迟迁移机制

- 内存局部性原理分为：**时间局部性**和**空间局部性**
- 内存的**时间局部性**，指的是内存某个位置被访问后，未来一段时间内，**相同位置**有很大概率会被再次访问



- 内存的**空间局部性：**内存某个位置被访问后，未来一段时间内，**相邻位置**有很大概率会被访问



虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

方案7：热点内存脏数据延迟迁移机制

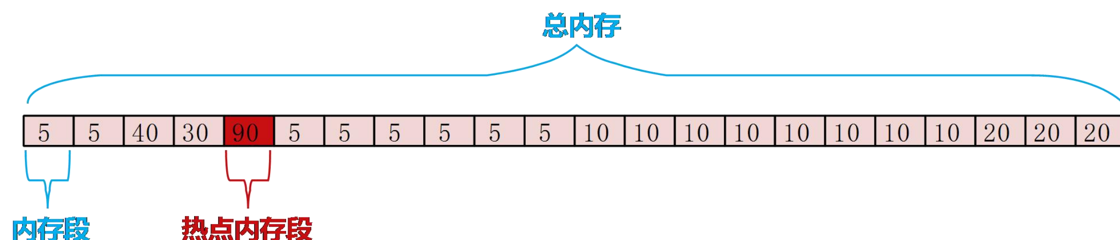
➢ 热点内存脏数据延迟迁移机制，方案概述：

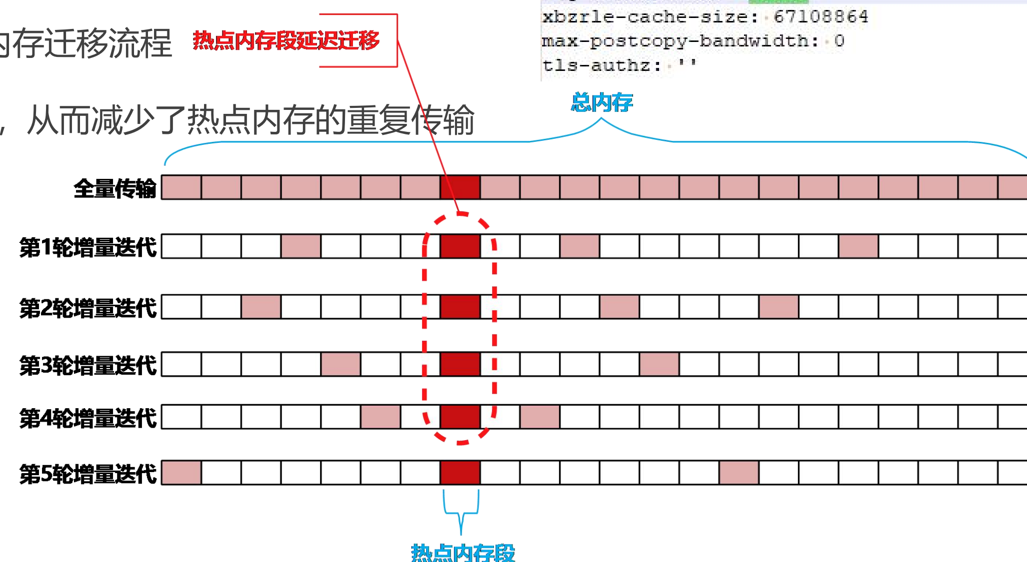
- 热迁移前先设置内存延迟迁移的比例(可以动态调整)
- 把总内存分为N个**内存段**，为每个内存段设置一个**热度值**
- 内存热迁移进入增量迭代迁移阶段后，每轮迭代前统计各内存段的脏页数量，并以此为内存段的**热度值**
- 对各内存段的热度值从大到小排序，找出**热点内存段**(虚拟机频繁写的内存段)
- 按照延时迁移内存的比例，为热度最高的若干个**内存段**设置**延迟迁移标记**
- 在进行内存热迁移时，有**延迟迁移标记**的**内存段**的脏页先不迁移
- 当剩余的内存脏数据量小于总内存量的10%时，关闭延迟迁移功能，走正常的内存迁移流程

```
========== migrate_parameters ==========
announce-initial: 50 ms
announce-max: 550 ms
announce-rounds: 5
announce-step: 100 ms
compress-level: 1
compress-threads: 4
compress-wait-thread: off
decompress-threads: 4
throttle-trigger-threshold: 50
cpu-throttle-initial: 20
cpu-throttle-increment: 10
cpu-throttle-value: 0
max-cpu-throttle: 99
tls-creds: ''
tls-hostname: ''
max-bandwidth: 8589934592 bytes/second
downtime-limit: 1000 milliseconds
x-checkpoint-delay: 20000
block-incremental: off
multifd-channels: 2
multifd-compression: none
lz4-acceleration: 8
ram-save-max-wait: 50
ram-delay-save-percent: 30
bypass-compression: off
mig-block-size: 65536
xbzrle-cache-size: 67108864
max-postcopy-bandwidth: 0
tls-authz: ''
```

热点内存段延迟迁移

➢ **热点内存脏数据延迟迁移机制**：通过算法识别出热点内存段，并设置延迟迁移标记，从而减少了热点内存的重复传输





虚拟机热迁移性能优化方案

CEC 中国电子 | CESTC 中国系统

方案7：热点内存脏数据延迟迁移机制

➢ Oracle虚拟机，1000用户、3000 tps的压力测试场景，实现效果：

- 迁移内存耗时从**120秒减少到85秒**
- 迁移内存总量从**117G减少到75G**
- 热迁移期间，虚拟机的EPT_VIOLATION VM-EXIT事件**减少20%左右**

优化前
优化后

秒
150
120
90
60
30
120
85
迁移内存耗时

GB
150
120
90
60
30
117
75
迁移内存总量

info migrate

```
total ram: 67113544 kbytes
duplicate: 8309328 pages
skipped: 0 pages
normal: 518525 pages
normal bytes: 2074100 kbytes
dirty sync count: 4
page size: 4 kbytes
multifd bytes: 0 kbytes
pages-per-second: 1533480
dirty pages rate: 24524 pages
ram bulk stage: 0
ram delay save percent: 80
ram delay save: on
pagesize_pages: 512 pages
dirty_area_pages: 167936 pages
delay dirty ram: 6819724 pages
save dirty ram: 1704931 pages
new dirty ram: 79019 pages
total new dirty ram: 251772 pages
last sync ram: 8524655 pages
last dirty ram: 8030063 pages
remaining dirty: 7969683 pages
ram total: 16778386 pages
last page: 7440383
last block: mem-node3
   167936[D]   167936[D]        1[D]   167936[D]   167936[D]        1[D]   167936[D]   167936[D]   167936[D]   167936[D]
   167936[D]   167936[D]        1[D]   167936[D]   167936[D]   167936[D]        1[D]   167936[D]        1[D]   167936[D]
   167936[D]        1[D]   167936[D]   167936[D]   167936[D]   167936[D]   167936[D]   167936[D]   167936[D]        1[D]
   167936[D]   167936[D]   167936[D]        1[D]   167936[D]   167936[D]   167936[D]   167936[D]   167936[D]   167936[D]
   167936[D]        0[S]        0[S]        0[S]   116710[S]   167936[S]        1[S]   167936[S]        1[S]     3581[D]
   167936[D]        1[S]   167936[S]   167936[D]   167936[S]   167936[S]   167936[S]        1[S]   167936[S]        0[S]
        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]     1062[D]   167936[D]   167936[D]
   167936[D]        0[S]      899[D]   167936[D]    13611[D]        0[S]        0[S]        0[S]        0[S]        0[S]
        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]
        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]        0[S]   108690[D]
```

目录

01 虚拟机热迁移功能介绍

02 高业务负载虚拟机热迁移遇到的挑战

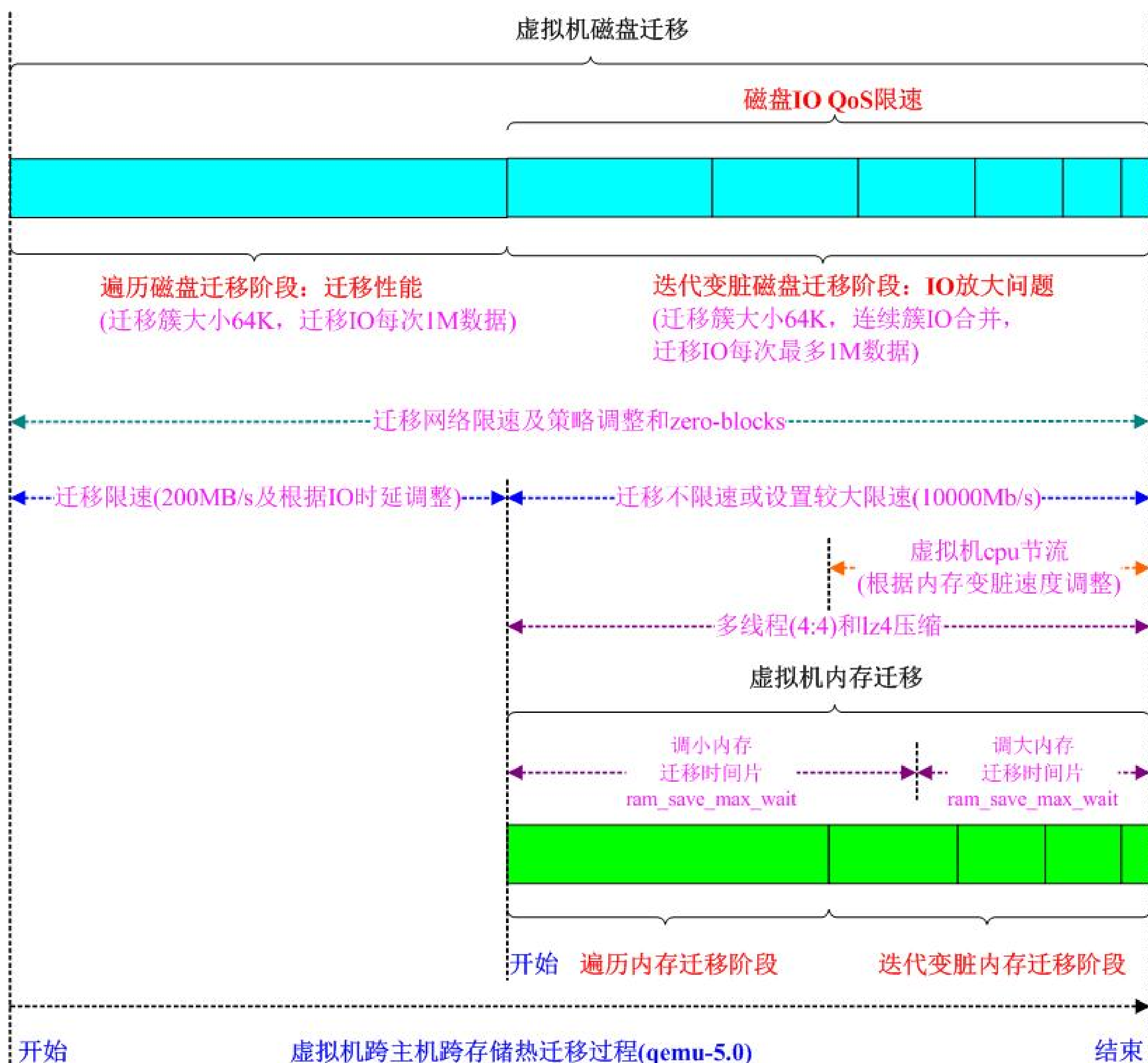
03 虚拟机热迁移性能优化方案

04 **优化方案总结**

05 热迁移数据一致性校验方案

优化方案总结

CEC 中国电子 | CESTC 中国系统



	评价热迁移性能优劣的指标	是否优化
1	迁移初始化时间	否
2	迁移总时间	是
3	迁移总数据量	是
4	迁移停机时间(downtime)	是
5	对业务性能的影响 & 影响时间	是
6	资源消耗(CPU、网络带宽等)	是

优化方案总结

CEC 中国电子 | CESTC 中国系统

测试环境

物理主机：

cpu	Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz	
内存	256G	
网络	存储网万兆网卡，管理网千兆网卡	

虚拟机：

cpu	16C	
内存	64G	2M大页
磁盘	120GB+3*100GB+3*50GB	预分配

数据库:

1000用户

3000 tps

MinDelay	5
MaxDelay	30
InterMinDelay	5
InterMaxDelay	30
Customer Registration	10
Update Customer Details	10
Browse Products	35
Order Products	35
Process Orders	5
Browse Orders	5

优化方案总结

热迁移性能优化方案的实现：2000行左右C代码 + 几百行管控面代码(方便机制与策略分开实现)

同存储热迁移场景：

- 跑满万兆带宽，QEMU压缩线程的CPU消耗从**1000%降低到200%**
- 迁移内存耗时从**120秒减少到85秒**
- 迁移内存总量从**117G减少到75G**
- 热迁移最后downtime停机阶段，ping网络中断时间**不超过1秒，丢包数5个以内**
- 在热迁移过程中，oracle业务性能低于100 TPS的时间控制在**3s以内**

跨存储热迁移场景：

- 基于**内存热迁移时间片动态调整机制**，在整体迁移耗时不变的情况下，内存脏数据的迁移量有明显的降低
- ORACLE数据库场景(8K随机写IO模型)，磁盘脏数据放大率**从93下降到8**，大幅降低了磁盘脏数据传输量
- 基于**磁盘热迁移IO QOS动态调整机制，**之前由于存储性能瓶颈、虚拟机业务IO压力大，导致无法迁移完的场景，优化之后基本都可以完成迁移
- 跨存储热迁移过程中，QEMU热迁移线程的CPU开销**下降了60%**
- 优化前，磁盘写操作的IO QoS限速到**2MB/s**，仍然无法完成磁盘数据迁移；优化后，磁盘写操作的IO QoS限速到**30MB/s**，可以完成磁盘数据迁移。

优化方案总结

CEC 中国电子 | CESTC 中国系统

深信服超融合680版本发布会

高负载业务下的虚拟机热迁移优化，达到业界领先水平

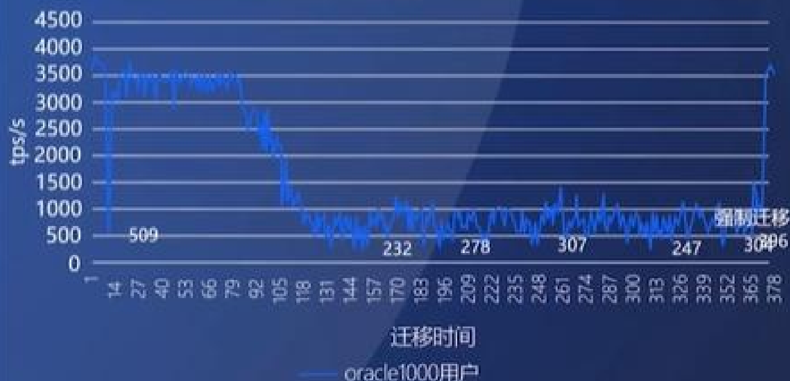
热迁移场景 主机设备维护、集群替换时，将虚拟机在不中断业务情况下迁移至其他主机。

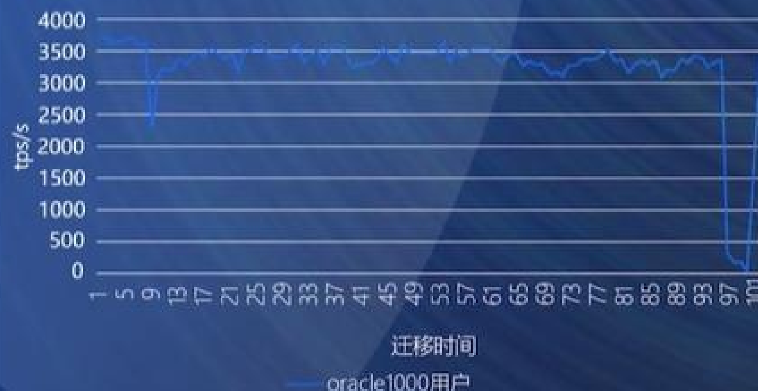
优化前

- CPU节流算法效率低，节流策略较为保守，导致CPU节流持续时间长
- Oracle 1000用户负载下，性能下降70%，业务中断，迁移数百秒无法完成

优化后

- 递进式增强的CPU节流策略，在满足条件下节流3s，然后强制切换完成迁移
- Oracle 1000用户负载下，性能下降仅3s内，业务不中断，100%完成迁移





数据压缩算法优化 zlib 算法 | BITMAP数据块粒度调整 | CPU节流策略优化 | 7项技术优化

CEC 中国电子 | CESTC 中国系统

目 录

01 虚拟机热迁移功能介绍

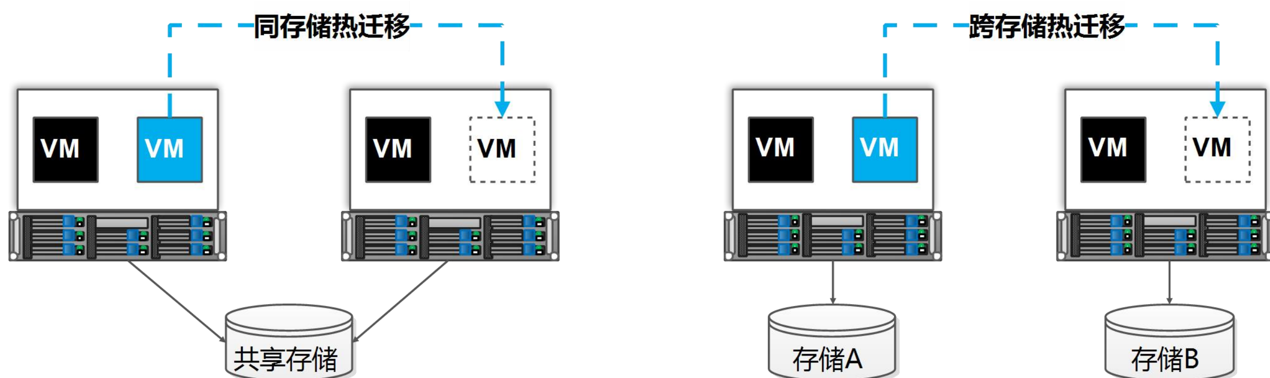
02 高业务负载虚拟机热迁移遇到的挑战

03 虚拟机热迁移性能优化方案

04 优化方案总结

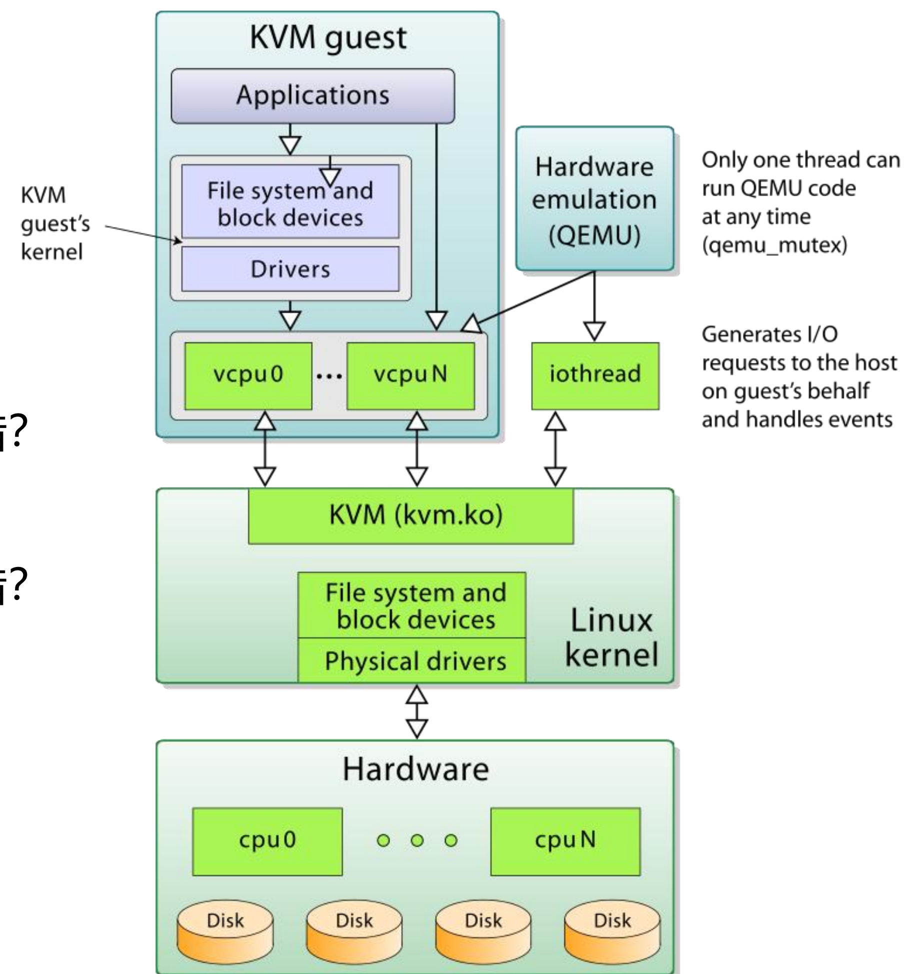
05 **热迁移数据一致性校验方案**

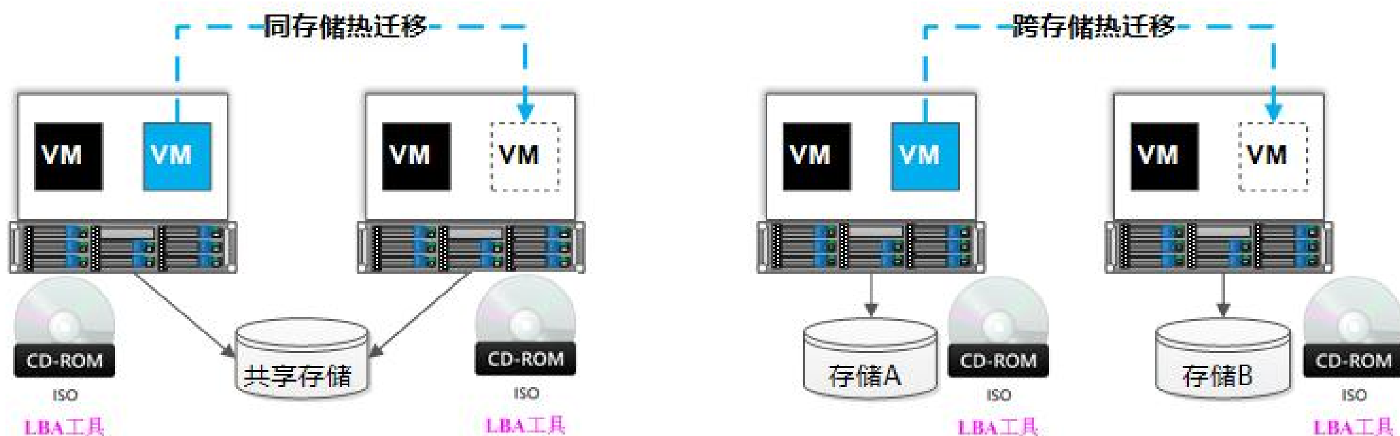
热迁移数据一致性校验方案



(1) 虚拟机热迁移，怎么验证内存数据迁移到目标端虚拟机后，数据没有出错？
(2) 虚拟机热迁移，怎么验证内存数据迁移，源端虚拟机没有漏迁数据？

(3) 虚拟机热迁移，怎么验证磁盘数据迁移到目标端虚拟机后，数据没有出错？
(4) 虚拟机热迁移，怎么验证磁盘数据迁移，源端虚拟机没有漏迁数据？





热迁移数据一致性校验方案

CEC 中国电子 | CESTC 中国系统

1.虚拟机添加hd_write_verify.mem_disk_lba.with.stripe.robin.1M.x86_64.V10.iso，作为系统启动盘；
2.添加数据盘，最多可以添加64个数据盘（一般只需要添加一个数据盘进行测试即可）；
3.运行虚拟机，稳定性与数据一致性自动化测试工具跑起来后，会同时测试校验磁盘与内存数据。
4.对虚拟机进行热迁移，如果虚拟机内运行的自动化测试工具，输出了“BUG”关键字信息，即测试出了LBA问题（出现数据一致性错误）；

热迁移数据一致性校验方案：qemu-2.5热迁移代码原生BUG：排查修复代价大

1. 2020年2月份(新冠疫情爆发期间)，多台虚拟机跑LBA自动化测试系统并进行热迁移，偶现内存热迁移数据不一致问题；
2. 大批量虚拟机跑热迁移(加大主机节点负载)，增加复现概率；
3. 排查手段：LBA工具 + 详细的调试日志 + mprotect抓取篡改内存现场；

```
03-04 15:44:17
[2020-03-04]
一、当前的情况：
1、早上热迁移winpe和linux虚拟机各出现一次lba，调试日志明确记录了这次内存数据被正确发送到目的端并保存，但是guest运行后读对应位置内存时，数据却是错误的。
2、分析了guest的内存，是1M buff中间一个4K内存页损坏，而前后的内存页数据是正确的，与调试日志对得上。
3、目前初步怀疑是目的端虚拟机启动阶段，某个流程把guest内存页4K数据破坏掉，关键阶段的时间线如下，在02:18:00.364045到02:18:09.273347这段时间，内存被破坏。
2020-03-04 02:18:00.364045   目的端qemu接收到对应内存页并保存，调试日志显示此内存页数据是正确的
2020-03-04 02:18:00.464607   内存迁移全部完成
2020-03-04 02:18:09.203232   热迁移完成，目的端虚拟机开始运行
2020-03-04 02:18:09.273347   测试工具检测数据错误，报告出现lba

二、下一步计划
1、在目的端虚拟机启动前的几个阶段，分别将guest内存文件保存下来，这样可以进一步判断内存页数据是在哪个阶段被破坏。
2、添加调试代码，调用mprotect接口将guest物理内存对应的qemu虚拟地址空间设为只读，如果有流程破坏内存，就可以被抓到。

03-04 18:15:11
有一种场景需要注意：
1、现在测试都是使用共享内存，ksm模块不会进行内存回收；
2、如果使用普通内存，ksm是会时进行相同页内存回收的(特别是全0页)，需要在这种场景下跑LBA工具进行热迁移测试，会不会也存在LBA的问题？

三、结论：
1、lba工具测试虚拟机热迁移，发现内存数据不一致；
2、在qemu的热迁移源端和目的端的代码中加详细的调试日志，确认所有内存数据被正确发送到目的端并保存，但是目的端guest运行后读对应位置内存时，LBA工具却发现有些内存数据是错误的；
3、在qemu的热迁移目的端的代码中调用mprotect接口将guest物理内存对应的qemu虚拟地址空间设为只读，在目的端虚拟机运行之前，部分内存数据被篡改并主动触发断言出core；
4、qemu热迁移原生bug，内存热迁移每轮迭代没有对解压缩线程做同步等待，如果解压缩线程执行速度很慢，在内存热迁移已经整体完成后，解压缩线程还在向guest写入旧数据，导致guest内存损坏。

Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `/usr/bin/kvm -id 227587371880 -chardev socket,id=qmp,path=/var/run/qemu-server/'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x00007f5ee420ee70 in inflate_fast () from /lib/x86_64-linux-gnu/libz.so.1
[Current thread is 1 (Thread 0x7f5e503fe700 (LWP 12496))]

(gdb) bt
#0  0x00007f5ee420ee70 in inflate_fast () from /lib/x86_64-linux-gnu/libz.so.1
#1  0x00007f5ee421116e in inflate () from /lib/x86_64-linux-gnu/libz.so.1
#2  0x00007f5ee42150b1 in uncompress2 () from /lib/x86_64-linux-gnu/libz.so.1
#3  0x00007f5ee4215193 in uncompress () from /lib/x86_64-linux-gnu/libz.so.1
#4  0x0000561b5b0b3058 in do_data_decompress (opaque=0x561b5d1453f0) at /home/test/qemu-2.5.1/migration/ram.c:2341
#5  0x00007f5eddc8bb50 in start_thread () from /lib/x86_64-linux-gnu/libpthread.so.0
#6  0x00007f5edd9d5a7d in clone () from /lib/x86_64-linux-gnu/libc.so.6
#7  0x0000000000000000 in ?? ()
```

热迁移数据一致性校验方案

CEC 中国电子 | CESTC 中国系统

存储稳定性测试与数据一致性校验工具和系统： （深信服科技/中国电子云/大道云行/zstack/云和恩墨/麒麟软件 等公司在使用）

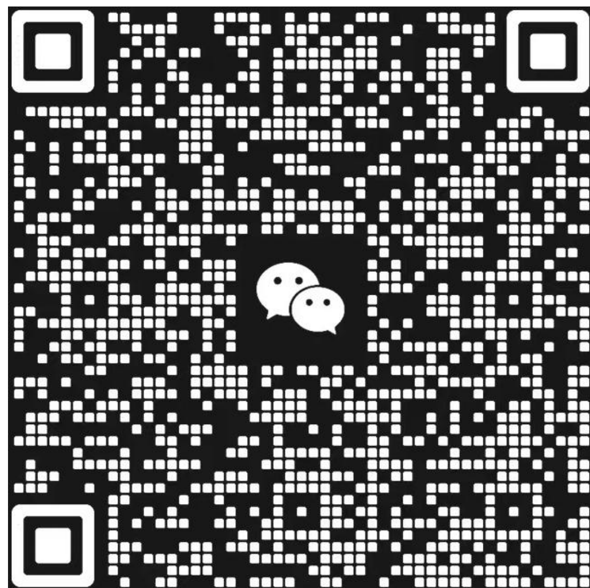
https://github.com/zhangyoujia/hd_write_verify

PPT:

https://github.com/zhangyoujia/hd_write_verify/存储稳定性测试与数据一致性校验工具和系统.pptx

录屏：

https://cloud.tencent.com/developer/video/78756



CEC 中国电子 | CESTC 中国系统

THANKS

中国电子云