# 现网ceph10.2.10版本出现slow\_req问题分析

## 1.问题背景

IP: 172.31.66.17

IM客服现网环境，版本如下：



## **2.问题现象**

IM客户现网环境概率偶现slow request，影响客服业务使用，当前已将有怀疑的host节点的osd全部down掉，本次抓取是2025年0321凌晨，重新拉起异常osd进行复现slow req现象，并抓取响应数据进行分析。

## **3.定位方案**

### 3.1 安装atop工具

现网环境需要安装atop工具，建议推广到块存储云主机所有集群，帮助分析当时系统资源使用情况

### **3.2 后台检测抓取异常osd需要的数据**

1.讯飞云存储监控第一时间捕获到异常osd

2.后台触发脚本一键搜集相关信息，执行脚本内容如下：

d=$(date "+%Y-%m-%d\_%H-%M-%S")

cd /root/nvme9\_trace

blktrace -d /dev/nvme9n1 -w 15

blkparse -i nvme9n1 -d nvme9n1.blktrace\_${d}.bin > /dev/null

btt -i nvme9n1.blktrace\_${d}.bin > nvme9n1\_${d}.txt

ceph daemon osd.109 perf dump > 109\_perf\_dump\_${d}.txt

ceph daemon osd.109 dump\_ops\_in\_flight > 109\_dump\_ops\_in\_flight\_${d}.txt

ceph daemon osd.109 dump\_historic\_ops > 109\_dump\_historic\_ops\_${d}.txt

cd /root/nvme10\_trace

blktrace -d /dev/nvme10n1 -w 15

blkparse -i nvme10n1 -d nvme10n1.blktrace\_${d}.bin > /dev/null

btt -i nvme10n1.blktrace\_${d}.bin > nvme10n1\_${d}.txt

ceph daemon osd.110 perf dump > 110\_perf\_dump\_${d}.txt

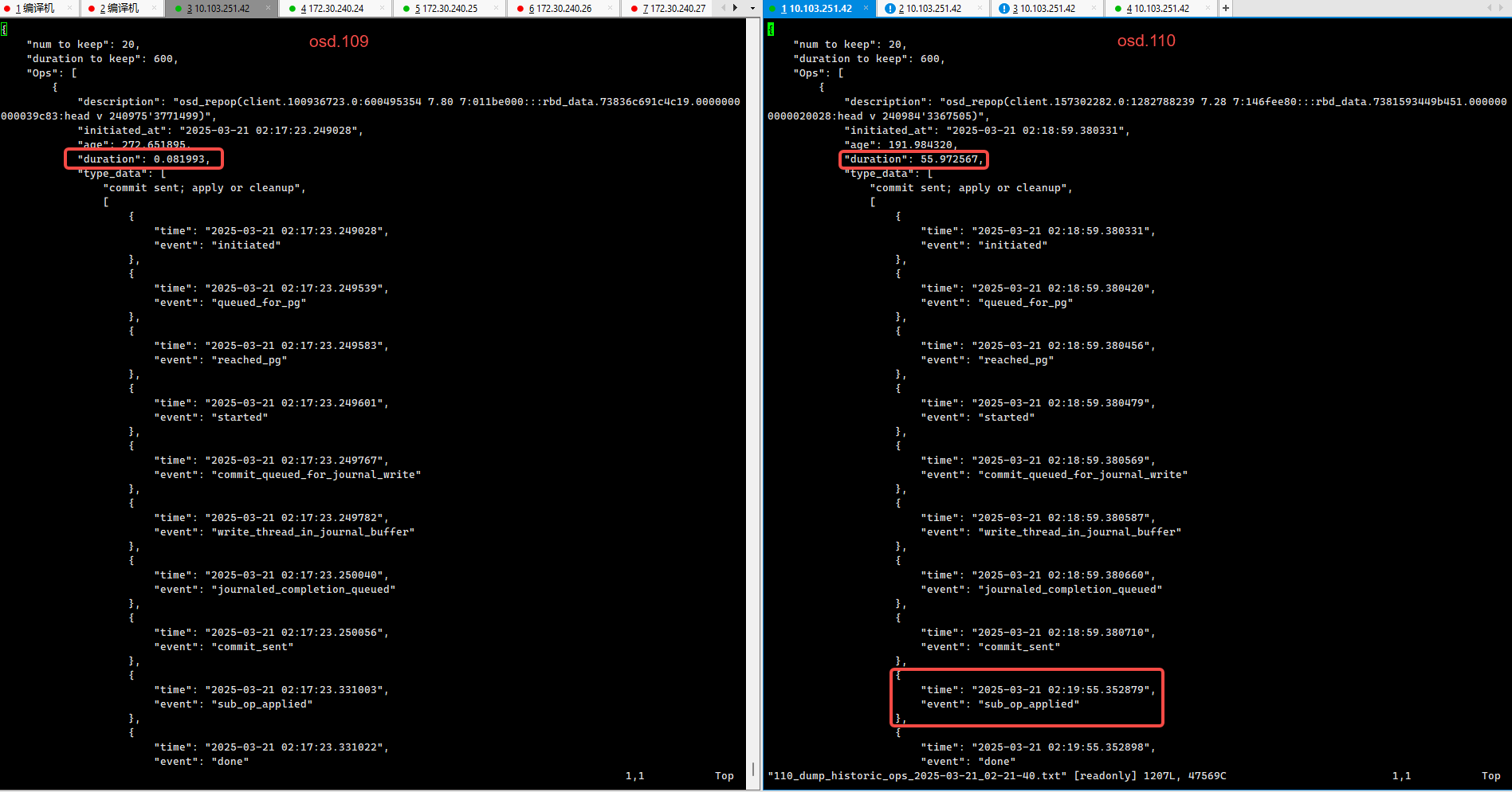
ceph daemon osd.110 dump\_ops\_in\_flight > 110\_dump\_ops\_in\_flight\_${d}.txt

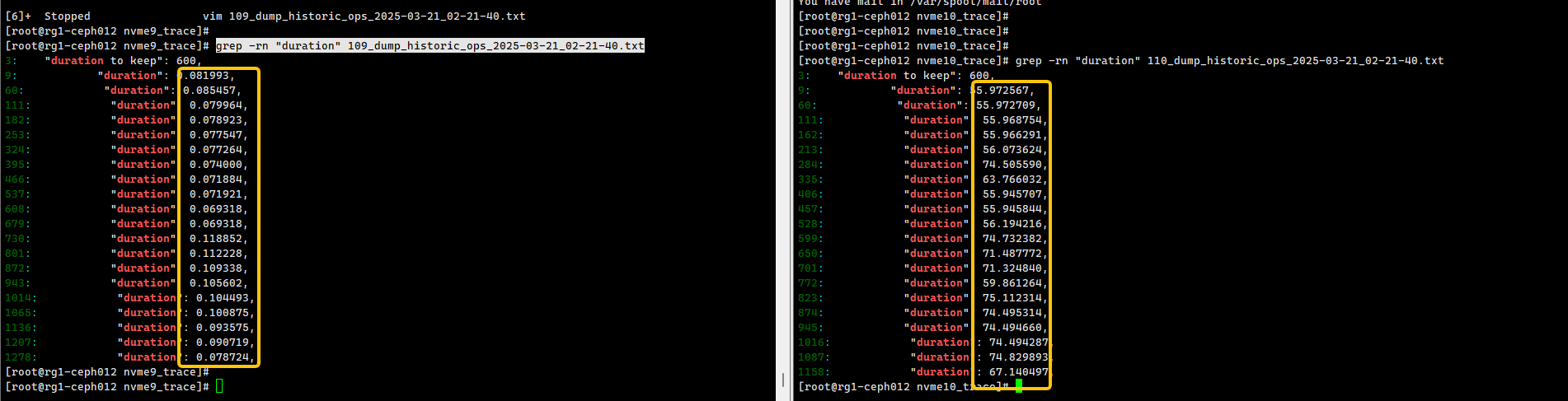
ceph daemon osd.110 dump\_historic\_ops > 110\_dump\_historic\_ops\_${d}.txt

注意：该脚本是已知nvme9、nvme10两块盘有异常，所以需要修改该脚本针对监控捕获有异常的osd进行创建目录，并收集对应osd的信息

## **4.定位分析**

1.现网执行的动作是重新拉起osd109、osd110两个，进行对比分析，先看ceph daemon osd.x dump\_historic\_ops命令输入结果如下：





可以看到osd.109时延是正常的，没有超过slow op的阈值，所以osd.109是正常拉起的，但是osd.110明显出现时延较大的op，这些较慢的处理中，有一些是副本OSD写数据比较慢，有一些是OSD.110 本身写数据比较慢，这里暂时先不考虑其他OSD，先关注OSD.110，通过第一张截图，可以明确大时延主要在写从osd的store层，data部分较慢

OSD.110 op\_applied 慢的时延拆分

{

"description": "osd\_op(client.100936723.0:600499694 7.b482b635 rbd\_data.73836c691c4c19.0000000000039c86 [] snapc 0=[] ack+ondisk+write+known\_if\_redirected e240990)",

"initiated\_at": "2025-03-21 02:20:33.701458",

"age": 97.663194,

"duration": 74.829893, ------------------------------------------> 耗时74.829s

"type\_data": [

"commit sent; apply or cleanup",

{

"client": "client.100936723",

"tid": 600499694

},

[

{

"time": "2025-03-21 02:20:33.701458",

"event": "initiated"---------------------------------------> op初始化

},

{

"time": "2025-03-21 02:20:33.701662",

"event": "queued\_for\_pg" ---------------------------> 塞到pg队列的时间

},

{

**"time": "2025-03-21 02:20:51.539799", 耗时18s**

"event": "reached\_pg" -------------------------------------> osd从pg队列取出op

},

{

"time": "2025-03-21 02:20:51.539843",

"event": "started" --------------------------------> pg 开始处理io

},

{

"time": "2025-03-21 02:20:51.539882",

"event": "waiting for subops from 128,140" -------------------------------------> 组织发送给其他两个从副本OSD消息

},

{

"time": "2025-03-21 02:20:51.540640",

"event": "commit\_queued\_for\_journal\_write" -----------------------------------> 发送到filestore层 队列

},

{

"time": "2025-03-21 02:20:51.541776",

"event": "write\_thread\_in\_journal\_buffer" ---------------------------------------> filestore层write\_thread\_entry线程从队列取出op

},

{

"time": "2025-03-21 02:20:51.541869",

"event": "journaled\_completion\_queued" ------------------------------------------>filestore 日志线程落盘日志

},

{

"time": "2025-03-21 02:20:51.851083",

"event": "op\_commit" -------------------------------------------------> 本地写完日志

},

{

**"time": "2025-03-21 02:21:10.093071", 耗时19s**

"event": "sub\_op\_commit\_rec from 128" -----------------------------------------------------> 从副本OSD.128日志落盘后发送消息给主OSD

},

{

**"time": "2025-03-21 02:21:10.093114",**

"event": "sub\_op\_commit\_rec from 140" -----------------------------------------------------> 从副本OSD.140日志落盘后发送消息给主OSD

},

{

"time": "2025-03-21 02:21:10.093150",

"event": "commit\_sent"

},

{

**"time": "2025-03-21 02:21:48.531305",耗时38s**

"event": "op\_applied" ---------------------------------------------------------------------> 本地写数据落盘完成

},

{

"time": "2025-03-21 02:21:48.531351",

"event": "done"

}

]

]

},

该op有三处较慢的处理io

{

**"time": "2025-03-21 02:20:51.539799", 耗时18s**

"event": "reached\_pg" -------------------------------------> osd从pg队列取出op

},

OSD从pg队列取出的时延较长，可能的原因有队列排队严重，前面的io处理较慢，这个就要看第一个op处理慢的原因，如果不是这个OSD慢，是从OSD慢，就要看从OSD的dump信息，但是从目前的数据推测大概率还是写盘的时延较慢；另外一个点从htop上看，节点CPU并不忙，甚至非常不忙，可以增大线程数提高并发处理能力；

{

**"time": "2025-03-21 02:21:10.093071", 耗时19s**

"event": "sub\_op\_commit\_rec from 128" -----------------------------------------------------> 从副本OSD.128日志落盘后发送消息给主OSD

},

从副本写日志落盘完成后发送给主OSD慢，这个有几种可能性，要么是网络，网卡达到瓶颈上限，需要存储前端网、存储后端网物理分开，但是似乎不太可能，如果凌晨网卡都达瓶颈，那么白天业务基本扛不住，要么就是从OSD本身写日志慢，这个就要捕获从OSD当时的dump信息，但是推测应该是本身写慢； 还有一种就是CPU使用率太低，通过增加线程数来提升CPU效率

{

**"time": "2025-03-21 02:21:48.531305",耗时38s**

"event": "op\_applied" ---------------------------------------------------------------------> 本地写数据落盘完成

},

这个时延占大头，本地写数据落盘慢，所以推测前面两个也是从OSD落盘慢，但是从目前已有的信息上暂时看不出更细化的差异，这个时延点是filestore层调用\_write写接口进行写盘，内核文件系统处理完调用CB回调，在回调的point点，基本认为是内核xfs文件系统本身的问题

1. 继续查看对应osd的perf dump文件，说明一下，这份perf dump和上面op dump信息不能够对应起来看，因为上面是针对每一个具体的op 利用optracker dump出的信息，而perf dump是通过在IO栈上打点获取的数据，可以大致比较

文件：110\_perf\_dump\_2025-03-21\_02-21-40.txt

Filestore层打点统计

"filestore": {

"journal\_queue\_ops": 0,

"journal\_queue\_bytes": 0,

"journal\_ops": 50,

"journal\_bytes": 1933312,

"journal\_latency": { -------------------------------------------------------------> 主OSD写日志落盘完成时延，5.875e-4， 时延不大，和dump信息相一致

"avgcount": 186767,

"sum": 109.742034801

},

"journal\_wr": 172048,

"journal\_wr\_bytes": {

"avgcount": 172048,

"sum": 113833414656

},

"journal\_full": 0,

"committing": 0,

"commitcycle": 255,

"commitcycle\_interval": {------------------------------------------------------> filestore数据写入内核xfs文件系统sync刷盘周期

"avgcount": 255,

"sum": 4396.883290641

},

"commitcycle\_latency": { --------------------------------------------------> filestore数据写入内核xfs文件系统sync刷盘时延，可以不用管，但是时延在31.9ms

"avgcount": 255,

"sum": 8.152394633

},

"op\_queue\_max\_ops": 50,

"op\_queue\_ops": 50,

"ops": 186767,

"op\_queue\_max\_bytes": 104857600,

"op\_queue\_bytes": 1785838,

"bytes": 113196469285,

"apply\_latency": {

"avgcount": 186717, ----------------------------------------> 主OSD从进入fiestore层到写数据落盘完成的时延 974ms，基本上非常高

"sum": 181916.401610857

},

"queue\_transaction\_latency\_avg": { --------------------------------------------> filestore层处理平均耗时36.6ms，基本超过slow req限制，此时未必写完成数据

"avgcount": 186767,

"sum": 6837.302249750

}

},

"finisher-JournalObjectStore": {

"queue\_len": 0,

"complete\_latency": {

"avgcount": 186767,

"sum": 0.681426386

}

},

"finisher-filestore-apply-0": {

"queue\_len": 0,

"complete\_latency": {

"avgcount": 39611,

"sum": 19.079791698 ---------------------------------------------------------------------------> 主OSD写盘时延，平均较低

}

},

"finisher-filestore-ondisk-0": {

"queue\_len": 3,

"complete\_latency": {

"avgcount": 39640, ---------------------------------> 主OSD写日志完成后放入finish queue队列，finsher\_thread\_entry线程再从队列取出op进行后续的处理，这个耗时和日志落盘没关系，日志此时已经落盘完成，但是该时延为22ms，相对较高

"sum": 900.532658592

}

},

"leveldb": {

"leveldb\_get": 67920,

"leveldb\_transaction": 240535,

"leveldb\_get\_latency": {

"avgcount": 67920, -----------------------------------> db获取元数据， 时延在58ms

"sum": 3963.823474942

},

"leveldb\_submit\_latency": {

"avgcount": 240280, --------------------------------------> 写db提交很快，内存操作

"sum": 2.406057323

},

"leveldb\_submit\_sync\_latency": {

"avgcount": 255,

"sum": 0.193181048

},

"leveldb\_compact": 0,

"leveldb\_compact\_range": 0,

"leveldb\_compact\_queue\_merge": 0,

"leveldb\_compact\_queue\_len": 0

},

"osd": {

"op\_wip": 20,

"op": 43925,

"op\_in\_bytes": 87420928,

"op\_out\_bytes": 2489215712,

"op\_latency": {

"avgcount": 43925, -----------------------> OSD层面统计op平均时延在291ms

"sum": 12787.194220548

},

"op\_process\_latency": {

"avgcount": 43925, --------------------------> 从op\_wq队列取出后的处理时延

"sum": 437.078821946

},

"op\_prepare\_latency": {

"avgcount": 43928,------------------------->op进行副本事务上下文的处理

"sum": 343.705675716

},

"op\_r": 42608,

"op\_r\_out\_bytes": 2489215712,

"op\_r\_latency": {

"avgcount": 42608, --------------------------------------> 读io时延，平均在281ms

"sum": 11983.449590064

},

"op\_r\_process\_latency": {

"avgcount": 42608,

"sum": 11.239124881

},

"op\_r\_prepare\_latency": {

"avgcount": 42611,

"sum": 12.476820576

},

"op\_w": 1317,

"op\_w\_in\_bytes": 87420928,

"op\_w\_rlat": {

"avgcount": 821,

"sum": 29.947825524

},

"op\_w\_latency": {

"avgcount": 1317, ------------------------------------>osd层面写时延在610ms

"sum": 803.744630484

},

"op\_w\_process\_latency": {

"avgcount": 1317, -----------------------------------> 写op\_wq队列取出后的处理时延 323ms

"sum": 425.839697065

},

"op\_w\_prepare\_latency": {

"avgcount": 1317, ----------------------------------->写 op进行副本事务上下文的处理 251ms

"sum": 331.228855140

},

"subop": 9518,

"subop\_in\_bytes": 241973375,

"subop\_latency": {

"avgcount": 9518, ----------------------------> 从OSD时延365ms

"sum": 3482.888498679

},

"subop\_w": 9518,

"subop\_w\_in\_bytes": 241973375,

"subop\_w\_latency": {

"avgcount": 9518, --------------------------------> 从OSD写时延365ms

"sum": 3482.888498679

},

"recoverystate\_perf": { -----------------------------------------------------> pg 数据恢复过程中各状态时延，也比较高，但是这个流程比较明确

"initial\_latency": {

"avgcount": 112,

"sum": 0.742941138

},

"started\_latency": {

"avgcount": 361,

"sum": 225828.383082052

},

"reset\_latency": {

"avgcount": 473,

"sum": 3230.242098842

},

"peering\_latency": {

"avgcount": 34,

"sum": 267.927564952

},

"repnotrecovering\_latency": {

"avgcount": 207,

"sum": 77630.470572274

},

"repwaitrecoveryreserved\_latency": {

"avgcount": 1,

"sum": 0.041201393

},

"repwaitbackfillreserved\_latency": {

"avgcount": 105,

"sum": 93905.199031808

},

"reprecovering\_latency": {

"avgcount": 96,

"sum": 23283.947827708

},

"activating\_latency": {

"avgcount": 34,

"sum": 973.065655777

},

"replicaactive\_latency": {

"avgcount": 95,

"sum": 151551.933826059

},

"stray\_latency": {

"avgcount": 439,

"sum": 75852.706478900

},

"getinfo\_latency": {

"avgcount": 34,

"sum": 0.630159315

},

"getlog\_latency": {

"avgcount": 34,

"sum": 17.247730223

},

有几点结论：

queue\_transaction\_latency\_avg 是filestore层处理平均时延，基本都在36.6ms，这个时延是store层写完成日志以及其他io流转时延，不包括数据落盘时延， 增加线程数看看；

apply\_latency是数据落盘时延，这个和本地xfs文件系统有关系，是否和nvme盘未做分区有关系呢？怀疑是这样，建议做分区试试

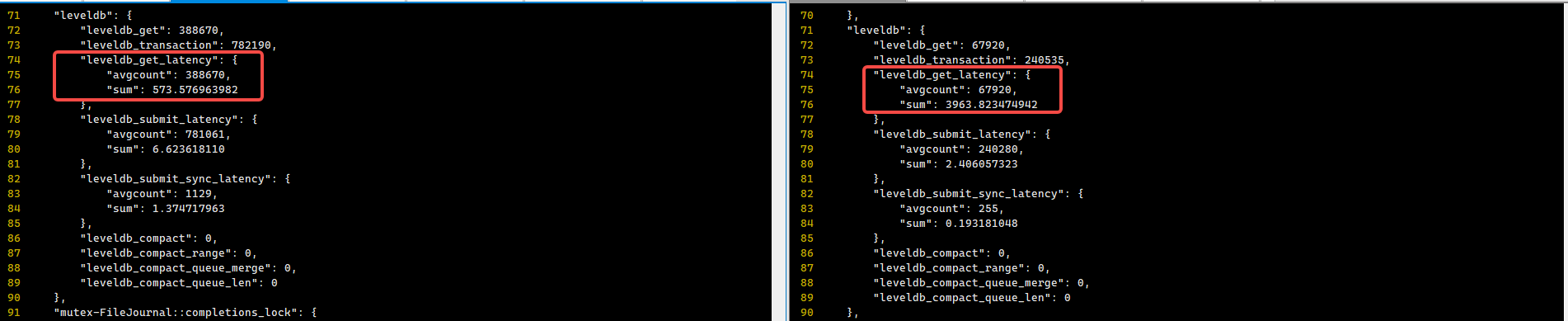
finisher-filestore-ondisk-0 线程处理耗时较长，说不通，此时已经日志落盘，op进入finisher队列后执行CB回调的时延，先通过增加线程数提高并发处理能力试试，如果增加线程数未有明显变化，这里就要再展开细分析一下；

leveldb\_get\_latency 获取元数据时延较长，基本可以认定是cache miss，需要增大db cachesize

op\_r\_latency 读请求不是造成本次io slow op的原因，可能和当时集群数据迁移或均衡有关系，需要控制数据迁移或均衡的速率

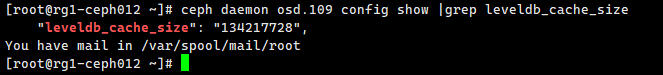
3.其他分析

下面是leveldb时延，正常来说，ceph10.2.10版本 filesstore层普通io不会写db，元数据omap会记db

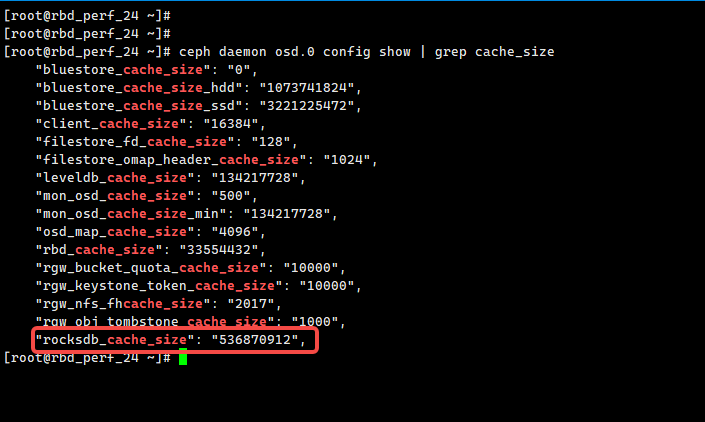


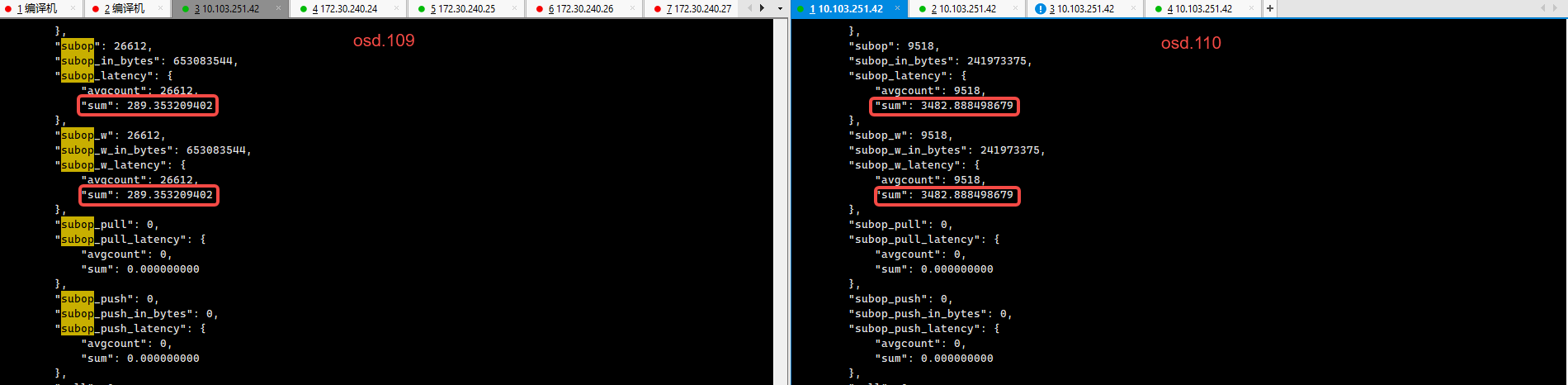
Leveldb get操作时延较高， 单io时延为0.0583s = 58.3ms

然后查看cache\_size，如下图所示：



db cache 太小，需要增大，参考ceph15.2.17版本rocksdb，建议增大为512M，如下图所示：





左侧是osd.109, 右侧是osd.110，可以看到subop latency

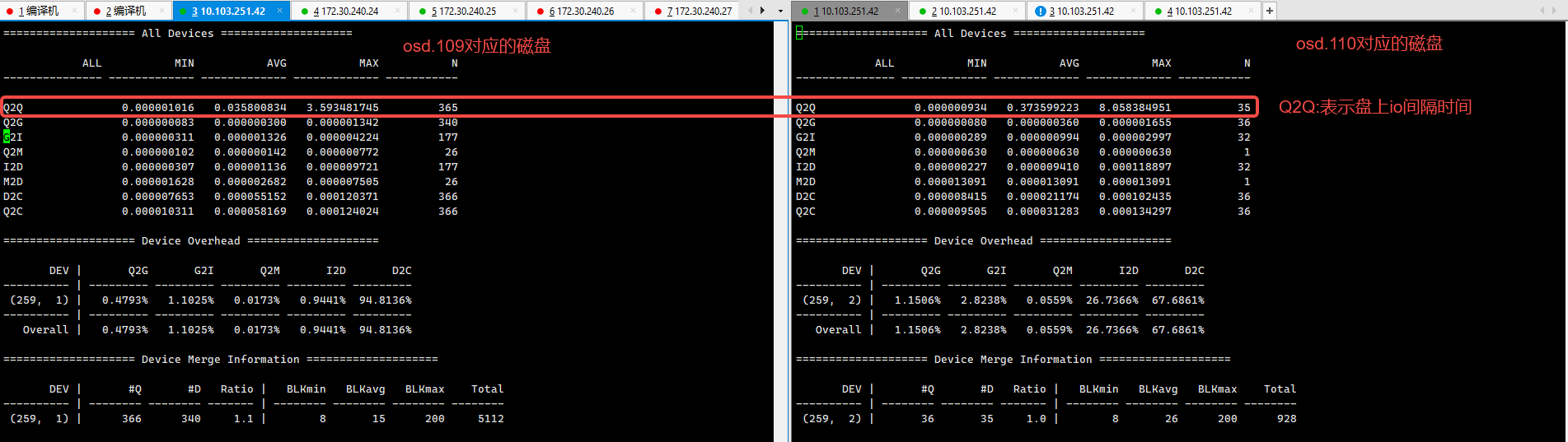
Osd.109从osd一共是写26612个io，耗时是289.353209402s， 单个io耗时0.01087s=10.87ms

osd.110一共是写9518, 耗时是3482.888498679s, 单个io耗时0.36592s=365ms

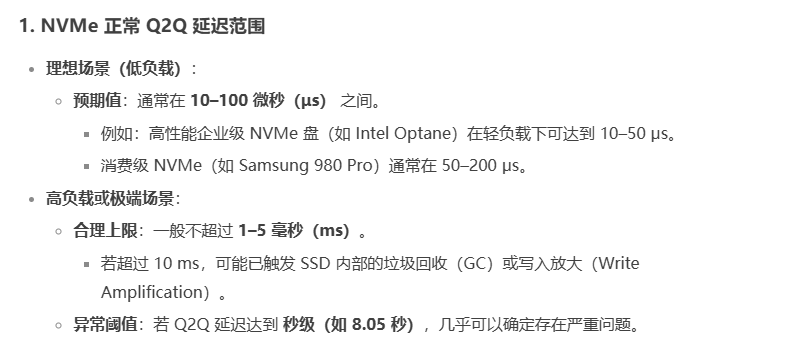
可以看到osd.110比osd.109 从osd写时延高了一个数量级，非常明显；注意的是，ceph daemon osd.x dump\_historic\_ops 和perf dump命令不能对应起来看，dump\_historic\_ops是确定的一个op时延，perf dump是该osd上处理的所有op的均值，只能大致对应来看。

从以上perf dump数据来看，osd.110写db和写data的时延相对osd.109来说，都是比较高的，从osd统计来看，osd.110承载的io业务并不比osd109的多，所以不太可能是osd.110 业务负载高导致，而且所有的软件配置都是一样的，也排除因软件配置不同导致问题，从OSD.110 perf dump上无法再继续分析subop慢的原因。

4.继续查看两个盘本身的处理时延

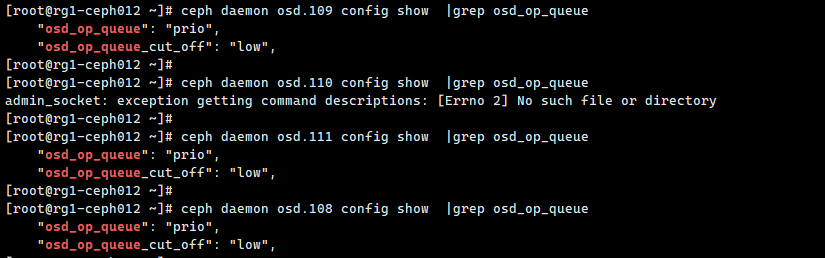


从Q2Q的数据来看，似乎是异常的，这是deepseek给出的结论：



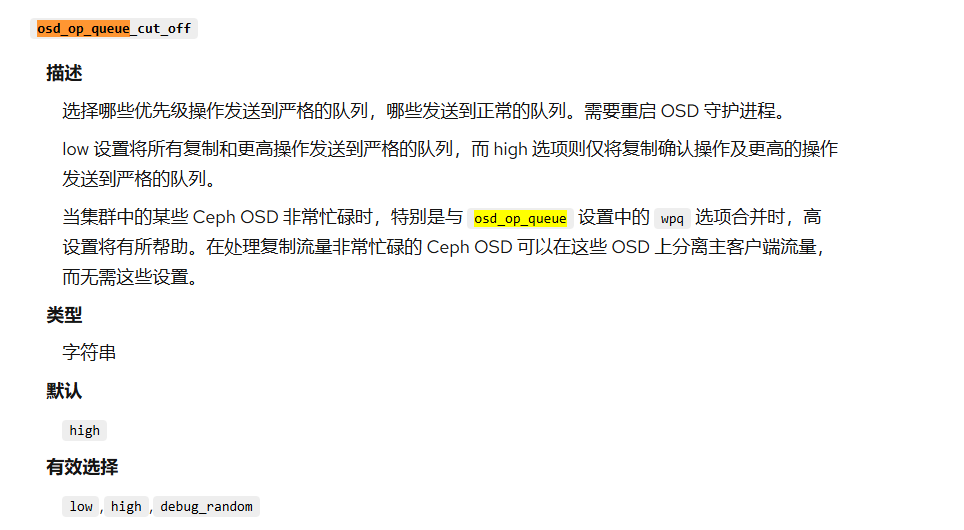
但是从Q2C、D2C数据来看，又似乎磁盘本身没问题，Q2C表示请求到完成的整个时延，D2C表示磁盘驱动从接收到op到处理完成的时延，所以这两组数据来看，又是矛盾的。

4.继续分析ceph osd配置，发现有如下差异

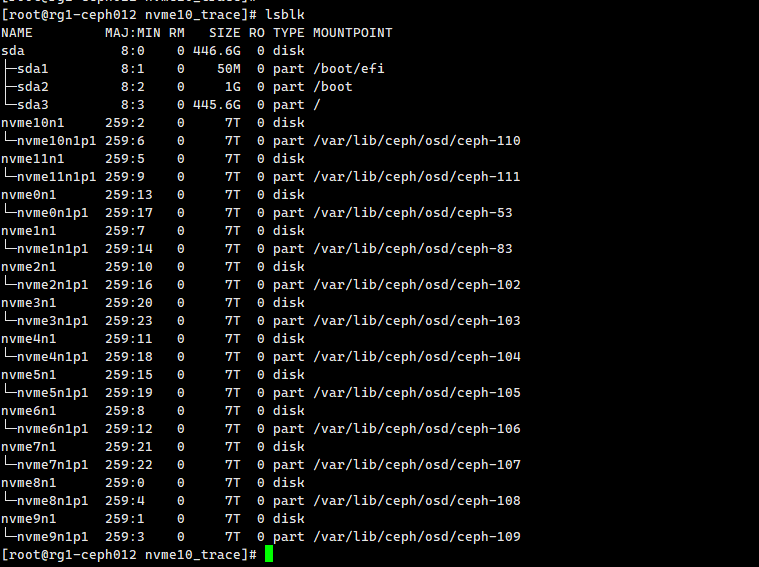


其中ceph osd的两个配置项如下:





从盘的分区挂载看如下：



Nvme盘未做分区，全部用来写data、db、wal，未必是盘的问题，怀疑是对盘本身的用法有问题，最好是根据分区大小，给nvme ssd做分区，data部分、db部分、wal部分

## **5.定位结论**

1.NVME盘建议做data、db、wal分区，大头应该还是盘未做分区的原因

2.修改ceph配置，请见6.解决方案 (可能提升不明显或有限)

3.调整数据均衡速度 (减少其他IO的影响)

4.通过blk数据看盘本身没啥问题

## **6.解决方案**

其中1-10 可以优先操作，看下效果。

1.现网集群节点统一安装atop命令

根因现网优先级酌情安装即可。

2.debug\_leveldb日志级别调为0/5

命令行修改ceph daemon osd.109 config set debug\_leveldb 0

修改ceph文件

3.修改ceph osd\_op\_queue、osd\_op\_queue\_cut\_off配置改为如下参数



4.修改leveldb cache大小leveldb\_cache\_size



由134217728改为536870912

命令行修改ceph daemon osd.109 config set leveldb\_cache\_size 536870912

ceph文件修改

5.修改osd io线程数，提高cpu使用率



建议改为4试试

动态修改: ceph daemon osd.109 config set osd\_op\_threads 4

永久修改:ceph.conf文件修改

6.调整filestore队列深度



filestore\_queue\_max\_ops 调为100

filestore\_queue\_max\_bytes调为1048576000

7.调整网络并发



调整ms\_dispatch\_throttle\_bytes为1048576000

8.调整leveldb 缓存文件大小



调整leveldb\_write\_buffer\_size为16777216

9.调整filestore 数据落盘线程数量



可以调整为2试试

10.调整filestore 写日志线程数量



可以调整为2试试

1. 最好重新部署nvme盘分区，区分data部分、db部分、wal部分(可能周期较长，部署方案分区大小规格、数据迁移策略等需要另外考虑)

因为现网是基于filestore，db分区和数据分区可以放到一起，暂时不需要单独做分区，目前仅针对journal做单独分区，大小暂定10GB