一孔之见：不同测试场景下遇到的反作用力

李焱 Homer Li 2024.11

感受到设计者在左右为难中取舍

相比很多其它的文件系统, 也能看到Lustre已经是非常高效率的平衡了

使用各种的功能或者功能之间的协同, 往往都伴随代价

是进亦忧, 退亦忧。然则何时而乐耶？其必曰.....

Performance TAX

ZFS is an example because open source
Some erasure coding distributed systems have higher amplification

It's hidden behind a black box, but you can still sense its impact

Performance tax I

Mdadm raid5

fio --rw=randwrite --name 555 --bs=64K --filename=/dev/md10

fio --rw=randwrite --name 555 --bs=128K --filename=/dev/md10

Device	r/s	w/s	rMB/s	wMB/s	rrqm/s	wrqm/s	%rrqm	%wrqm	r_await	w_await	aqu-sz	rareq-sz	wareq-sz	svctm	%util
sdg	2766.00	5619.00	171.62	338.00	41050.00	80878.00	93.69	93.50	1.24	0.94	8.72	63.54	61.60	0.12	97.20
sdi	2798.00	5857.00	172.94	336.52	41383.00	80253.00	93.67	93.20	1.26	0.99	9.35	63.29	58.83	0.11	97.40
sdh	2606.00	5923.00	161.40	348.44	38697.00	83258.00	93.69	93.36	1.22	0.96	8.89	63.42	60.24	0.11	97.80

Device	r/s	w/s	rMB/s	wMB/s	rrqm/s	wrqm/s	%rrqm	%wrqm	r_await	w_await	aqu-sz	rareq-sz	wareq-sz	svctm	%util
sdg	0.00	7577.00	0.00	472.51	0.00	113416.00	0.00	93.74	0.00	1.15	8.74	0.00	63.86	0.11	82.10
sdi	0.00	7577.00	0.00	472.38	0.00	113398.00	0.00	93.74	0.00	1.21	9.16	0.00	63.84	0.11	82.20
sdh	0.00	7576.00	0.00	472.51	0.00	113371.00	0.00	93.74	0.00	1.18	8.96	0.00	63.87	0.11	82.30

Performance tax I

Trace the stripe code

echo "func handle_stripe_dirtying +p" >>
/sys/kernel/debug/dynamic_debug/control

```
[ 9803.659309] for sector 1214603256 state 0x2041, rmw=2 rcw=0
[ 9803.659312] for sector 1214620160 state 0x2041, rmw=2 rcw=0
[ 9803.659315] for sector 1214620168 state 0x2041, rmw=2 rcw=0
[ 9803.659319] for sector 1214620176 state 0x2041, rmw=2 rcw=0
[ 9803.659321] for sector 1214620184 state 0x2041, rmw=2 rcw=0
[ 9803.660245] for sector 1214620192 state 0x41, rmw=2 rcw=1
[ 9803.660247] Read_old block 0 for Reconstruct
[ 9803.660249] for sector 1214620200 state 0x41, rmw=2 rcw=1
[ 9803.660250] Read_old block 0 for Reconstruct
[ 9803.660252] for se
[ 9803.660253] Read_o
[ 9803.660256] for se
[ 9803.660257] Read_o
[ 9803.660258] for se
[ 9803.660264] Read_o
```

```
--7.72%--analyse_stripe
|
--4.22%--md_end_io_acct
          |
           --2.60%--end_bio_bh_io_sync
                     |
                     |--1.48%--end_page_writeback
                     |          |
                     |           --1.35%--test_clear_page_writeback
                     |
                      --0.67%--end_buffer_async_write
|
 --1.22%--handle_stripe_dirtying
           |
            --0.63%--schedule_reconstruction
```

```
if ((rcw < rmw || (rcw == rmw && conf->rmw_level != PARITY_PREFER_RMW)) && rcw > 0) {
        /* want reconstruct write, but need to get some data */
        int qread =0;
        rcw = 0;
        for (i = disks; i--; ) {
                struct r5dev *dev = &sh->dev[i];
                if (!test_bit(R5_OVERWRITE, &dev->flags) &&
                    i != sh->pd_idx && i != sh->qd_idx &&
                    !test_bit(R5_LOCKED, &dev->flags) &&
                    !(test_bit(R5_UPTODATE, &dev->flags) ||
                      test_bit(R5_Wantcompute, &dev->flags))) {
                        rcw++;
                        if (test_bit(R5_Insync, &dev->flags) &&
                            test_bit(STRIPE_PREREAD_ACTIVE,
                                     &sh->state)) {
                                pr_debug("Read_old block "
                                         "%d for Reconstruct\n", i);
```

Performance tax II

Recordsize = 1MiB, cache miss

echo 3 > /proc/sys/vm/drop_caches

dd if=test_file1 of=/dev/null bs=4K skip=1 count=1

4096 bytes (4.1 kB, 4.0 KiB) copied, 0.00436596 s, 938 kB/s

dd if=test_file1 of=/dev/null bs=4K skip=99 count=1

4096 bytes (4.1 kB, 4.0 KiB) copied, 0.00379674 s, 1.1 MB/s

dd if=test_file1 of=/dev/null bs=1M skip=99 count=1

1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.0061811 s, 170 MB/s

pool	alloc	free	read	write	read	write
----------	-----	-----	-----	-----	-----	-----
test_tank	4.35G	182T	8	0	1.00M	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	291	0	1.49M
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	8	0	1.00M	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	294	0	1.44M
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	8	0	1.00M	0
test_tank	4.35G	182T	0	304	0	1.44M
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0

Performance tax III

Notrunc rewrite cause RMW, cache miss

Recordsize = 1MiB for view, echo 3 > /proc/sys/vm/drop_caches

rm -f test_file; dd if=/dev/zero of=test_file bs=1M count=1024

1073741824 bytes (1.1 GB, 1.0 GiB) copied, 1.69116 s, 635 MB/s

dd if=/dev/zero of=test_file bs=1M count=1024

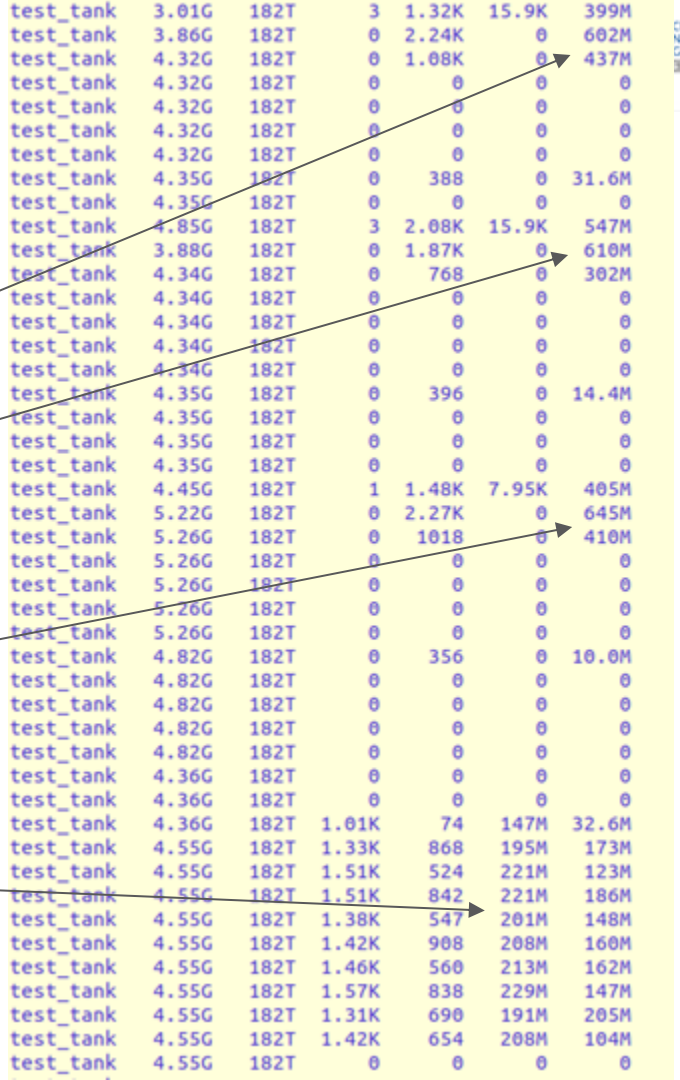
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 1.67787 s, 640 MB/s

dd if=/dev/zero of=test_file bs=1M count=1024 conv=notrunc

1073741824 bytes (1.1 GB, 1.0 GiB) copied, 1.65156 s, 650 MB/s

dd if=/dev/zero of=test_file bs=4K count=262144 conv=notrunc

1073741824 bytes (1.1 GB, 1.0 GiB) copied, 9.66428 s, 111 MB/s



test_tank	3.01G	182T	3	1.32K	15.9K	399M
test_tank	3.86G	182T	0	2.24K	0	602M
test_tank	4.32G	182T	0	1.08K	0	437M
test_tank	4.32G	182T	0	0	0	0
test_tank	4.32G	182T	0	0	0	0
test_tank	4.32G	182T	0	0	0	0
test_tank	4.32G	182T	0	0	0	0
test_tank	4.35G	182T	0	388	0	31.6M
test_tank	4.35G	182T	0	0	0	0
test_tank	4.85G	182T	3	2.08K	15.9K	547M
test_tank	3.88G	182T	0	1.87K	0	610M
test_tank	4.34G	182T	0	768	0	302M
test_tank	4.34G	182T	0	0	0	0
test_tank	4.34G	182T	0	0	0	0
test_tank	4.34G	182T	0	0	0	0
test_tank	4.34G	182T	0	0	0	0
test_tank	4.35G	182T	0	396	0	14.4M
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.35G	182T	0	0	0	0
test_tank	4.45G	182T	1	1.48K	7.95K	405M
test_tank	5.22G	182T	0	2.27K	0	645M
test_tank	5.26G	182T	0	1018	0	410M
test_tank	5.26G	182T	0	0	0	0
test_tank	5.26G	182T	0	0	0	0
test_tank	5.26G	182T	0	0	0	0
test_tank	5.26G	182T	0	0	0	0
test_tank	4.82G	182T	0	356	0	10.0M
test_tank	4.82G	182T	0	0	0	0
test_tank	4.82G	182T	0	0	0	0
test_tank	4.82G	182T	0	0	0	0
test_tank	4.82G	182T	0	0	0	0
test_tank	4.36G	182T	0	0	0	0
test_tank	4.36G	182T	0	0	0	0
test_tank	4.36G	182T	1.01K	74	147M	32.6M
test_tank	4.55G	182T	1.33K	868	195M	173M
test_tank	4.55G	182T	1.51K	524	221M	123M
test_tank	4.55G	182T	1.51K	842	221M	186M
test_tank	4.55G	182T	1.38K	547	201M	148M
test_tank	4.55G	182T	1.42K	908	208M	160M
test_tank	4.55G	182T	1.46K	560	213M	162M
test_tank	4.55G	182T	1.57K	838	229M	147M
test_tank	4.55G	182T	1.31K	690	191M	205M
test_tank	4.55G	182T	1.42K	654	208M	104M
test_tank	4.55G	182T	0	0	0	0

High-frequency deletion and creation operations on a large number of small files

Lustre destroy OST orphan objects

Frequent creation and deletion of numerous small files on a Lustre filesystem

These OSTs orphan objects are still awaiting distributed transaction completion with the LLOG

The filesystem will be full.....

The cluster transaction

MDS(Read journal and send the clear instruction to OSS)

OSS(Destroy objects)

MDS (commit and update journal)

```
# lfs df -i
UUID                      Inodes       IUsed       IFree IUse% Mounted on
lustre-MDT0000_UUID       26214400         434    26213966   1% /mnt[MDT:0]
lustre-OST0000_UUID        7680000     2884287     4795713  38% /mnt[OST:0]
lustre-OST0001_UUID        7680000     3022276     4657724  40% /mnt[OST:1]
lustre-OST0002_UUID        7680000     3022249     4657751  40% /mnt[OST:2]

filesystem_summary:       14111622         434    14111188   1% /mnt

# lfs df -i
UUID                      Inodes       IUsed       IFree IUse% Mounted on
lustre-MDT0000_UUID       26214400         434    26213966   1% /mnt[MDT:0]
lustre-OST0000_UUID        7680000     2880183     4799817  38% /mnt[OST:0]
lustre-OST0001_UUID        7680000     3018045     4661955  40% /mnt[OST:1]
lustre-OST0002_UUID        7680000     3018008     4661992  40% /mnt[OST:2]

filesystem_summary:       14124198         434    14123764   1% /mnt
```

Destroy OST orphan objects

MDS: osp_precreate_cleanup_orphans
ptlrpc_send_new_req
osp_sync_thread
llog_cat_process
osp_sync_process_queues
osp_sync_process_committed
llog_cancel_arr_rec

OSS: ptlrpc_server_handle_request
tgt_request_handle
ofd_destroy_hdl
osp_sync_new_unlink64_job

```
int osp_sync_init(const struct lu_env *env, struct osp_device *d)
{
        struct task_struct      *task;
        struct osp_sync_args    *args;
        DECLARE_COMPLETION_ONSTACK(started);
        int                      rc;

        ENTRY;

        d->opd_sync_max_rpcs_in_flight = OSP_MAX_RPCS_IN_FLIGHT;
        d->opd_sync_max_rpcs_in_progress = OSP_MAX_RPCS_IN_PROGRESS;
        d->opd_sync_max_changes = OSP_MAX_SYNC_CHANGES;
```

Increased the Lustre parameters

(osp_precreate.c:874:osp_precreate_cleanup_orphans()) lustre-OST0000-osc-MDT0000: going to cleanup orphans since [0x100000000:0x39a61c:0x0]
depending on current load (num of changes being synced to OST) the callback can suspend awaiting for some new conditions
lustre-OST0000-osc-MDT0000: 0 changes, 4043 in progress, 0 in flight, 53 done
(llog.c:235:llog_cancel_arr_rec()) Canceling 4043 records, first 32391 in log [0x2a:0x1:0x0]

Destroy OST orphan objects

```
enum llog_op_type {
        LLOG_PAD_MAGIC          = LLOG_OP_MAGIC | 0x00000,
        OST_SZ_REC              = LLOG_OP_MAGIC | 0x00f00,
        /* OST_RAID1_REC        = LLOG_OP_MAGIC | 0x01000, never used */
        MDS_UNLINK_REC          = LLOG_OP_MAGIC | 0x10000 | (MDS_REINT << 8) |
                                  REINT_UNLINK, /* obsolete after 2.5.0 */
        MDS_UNLINK64_REC        = LLOG_OP_MAGIC | 0x90000 | (MDS_REINT << 8) |
                                  REINT_UNLINK,
```

OSS# perf probe -m /lib/modules/4.18.0-477.27.2.el8_lustre.x86_64/extra/lustre-osd-ldiskfs/fs/osd_ldiskfs.ko --add "osd_obj_del_entry dentry_name=dird->d_name.name:string name=name:string"

```
ll_ost01_001  4187 [002] 158121.824475: probe:osd_obj_del_entry: (ffffffffc1bcd320) dentry_name="d1" name="24112641"
ll_ost01_006  4238 [003] 158121.824573: probe:osd_obj_del_entry: (ffffffffc1bcd320) dentry_name="d21" name="24106325"
ll_ost01_006  4238 [003] 158121.825020: probe:osd_obj_del_entry: (ffffffffc1bcd320) dentry_name="d14" name="26417422"
```

OSS# awk 'BEGIN{printf "0x%x\n", or(0x10600000,0x90000,lshift(36,8),4)}'
0x10692404
OSS# perf probe -m /lib/modules/4.18.0-477.27.2.el8_lustre.x86_64/extra/lustre/fs/osp.ko --add "osp_sync_new_unlink64_job lrh_type=h->lrh_type"

```
osp-syn-2-0  3714 [003] 350782.951588: probe:osp_sync_new_unlink64_job: (ffffffffc201e2a1) lrh_type=0x10692404
osp-syn-0-0  2505 [005] 350782.951671: probe:osp_sync_new_unlink64_job: (ffffffffc201e2a1) lrh_type=0x10692404
```

Test DNE

Performance degraded within cross-MDT

	Move	Remove time	Inode number
2.15.5 remote_rename=1 between 2 mdt	2x + (remote MDT, in single MDS)	7x +	2x (parent link)
2.15.5 remote_rename=0 between 2 mdt	3x + (remote MDT, in single MDS)	1x	1x
2.15.5 not pass remote mdt	1x (local mdt)	1x	1x

```
 * When lookup item under striped directory, we need to locate the master
 * MDT-object of the striped directory firstly, then the client will send
 * lookup (getattr_by_name) RPC to the MDT with some slave MDT-object's FID
 * and the item's name. If the system is restored from MDT file level backup,
 * then before the OI scrub completely built the OI files, the OI mappings of
 * the master MDT-object and slave MDT-object may be invalid. Usually, it is
 * not a problem for the master MDT-object. Because when locate the master
 * MDT-object, we will do name based lookup (for the striped directory itself)
 * firstly, during such process we can setup the correct OI mapping for the
 * master MDT-object. But it will be trouble for the slave MDT-object. Because
 * the client will not trigger name based lookup on the MDT to locate the slave
 * MDT-object before locating item under the striped directory, then when
 * osd_fid_lookup(), it will find that the OI mapping for the slave MDT-object
 * is invalid and does not know what the right OI mapping is, then the MDT has
 * to return -EINPROGRESS to the client to notify that the OI scrub is rebuiding
 * the OI file, related OI mapping is unknown yet, please try again later. And
 * then client will re-try the RPC again and again until related OI mapping has
 * been updated. That is quite inefficient.
```

Move and remove timeout
Manual mode get the best performance

LU-10329: For DNE filesystems where there are large numbers of remote entries, for example once LU-4684 is landed to restripe directories, or DNE2 striped directories with many renames, the size of the REMOTE_PARENT_DIR may become very large.

Then in 2.17+ start using those directories (which would be incompatible for pre-2.16 servers, or servers not patched as above)

Empty file overhead

Lies, damned lies, and statistics !

/dev/sdb1 lustre 124G 5.5M 114G 1% /test/test_mdt0

/dev/sdb1 ext4 211G 28K 200G 1% /ext4_mount

MDT Default options: -J size=4096 -I 1024 -i 2560

Inode size: 1024 Bytes

Bytes-per-inode: 2560 Bytes : 1024 + 1536 = 2560 Bytes

Ext4/OST ldiskfs default: 65536 Bytes, lower inode number

cat /sys/class/block/sdb1/size (qemu-scsi 512 Bytes/sector)

450555904

450555904*512=230684622848 Bytes=214.842 GiB (RAW size)

Filesystem 1kB-blocks Used Available Use% Mounted on

/dev/sdb1 129855968 5624 118586448 1% /test/test_mdt0

Filesystem Inodes IUsed IFree IUse% Mounted on

/dev/sdb1 90112000 263 90111737 1% /test/test_mdt0

/dev/sdb1 14082048 11 14082037 1% /ext4_mount

test_mdt0/test_mdt0 48498425 336 48498089 1% /test/test_mdt0

tune2fs -l /dev/sdb1 (format lustre)

block count: 56319488

inode counts: 90112000

90112000*1536=138412032000 (data overhead)

90112000*1024=92274688000 (inode overhead)

138412032000+92274688000=230686720000 Bytes

Lustre MDT is more cost-effective for storing metadata than large data blocks. it appears that the DOM incurs higher storage costs

Lustre MDT capacity

/dev/sdb1 lustre 124G 5.5M 114G 1% /test/test_mdt0

test_mdt0/test_mdt0 lustre 206G 3.3M 206G 1% /test/test_mdt0

Filesystem Inodes IUsed IFree IUse% Mounted on

test_mdt0/test_mdt0 324392039 303615623 20776416 94% /test/test_mdt0

/dev/sdb1 90112000 87112506 2999494 97% /test/test_mdt0

Lustre 2.15.5 ZFS 2.1.15-1	Ldiskfs MDT(static)	ZFS MDT (dynamic)
64 Bytes attr	0 (inline)	0 (inline)
256 Bytes attr	0 (inline)	512 Bytes (spill block)
8x ACL	0 (inline)	32 Bytes
4000 Bytes attr	8192 Bytes	9326 Bytes
8000 Bytes attr	8192 Bytes	17524 Bytes
AVG inode size (with OST overhead, calculate by the RAW MDT size)	2582 Bytes	4096 Bytes
AVG inode size (with OST overhead, calculate by the RAW MDT size)	2582 Bytes	624 Bytes

Eg: DoM, FLR....., More bytes in the EA(eg:trusted.lov)

ZFS limit metadata size by quota

The ldiskfs about 2X or better performance than ZFS MDT

Calculate raidz ZPOOL capacity

- 10x 20,000,588,955,648 bytes Raidz3 2.1.11 7+3
  - zpool create test_tank -o ashift=12 raidz3 /dev/sd{a..j}

```
tart          End             Size          File system  Name                 Flags
048s          39063631871s    39063629824s               zfs-60b858fa60405af1
9063631872s   39063648255s    16384s
```

- 39063629824*512=20000578469888 (Qemu BLK)
- Align 256KiB
  - floor(20000578469888 / (256 * 1024)) * 256 * 1024=20,000,578,469,888
- 4 x 256KiB vdev label + 3.5 MiB boot load region
  - 20000578469888-4*262144-3.5*1024*1024=20,000,573,751,296

- 10x HDDs
  - 20000573751296*10=asize
- zdb -m test_tank | grep metaslab |head -1
  - metaslabs 11641 offset spacemap free
- 2 ^ 34 = 17179869184 bytes
  - $aszie/2^34=11641
- 2 ^ 34*11641=199990857170944

```
zdb -m test_tank | grep metaslab |head -1
  metaslabs 11641   offset                spacemap          free

zdb -C test_tank | grep -E 'metaslab_shift|asize'
             metaslab_shift: 34
             asize: 200005737512960
zpool list -p -o name,size,alloc,free test_tank
NAME                    SIZE     ALLOC              FREE
test_tank   199990857170944   1523712   199990855647232
```

Calculate raidz ZPOOL capacity

- Minimal write raidz3 : P(3)+1=4
- 128KiB=32x4KiB
  - 32 data/48 all = 2/3
  - floor(0.66667*512)/512=341/512 = 0.666015625(vdev_deflate_ratio)
  - 199990857170944*0.666015625= 133197035732992
  - space reservation 128GiB
    - 133197035732992-128*1024^3
      - =133059596779520
  - 1014816+133059595764704
    - =133059596779520
  - 133059596779520/20000588955648
    - 66.52%

```
# zfs list -p
NAME         USED            AVAIL     REFER  MOUNTPOINT
test_tank  1014816  133059595764704    261888  /test_tank
```

PPPDDDDDDD 7+3

PPPDDDDDDD 7+3

PPPDDDDDDD 7+3

PPPDDDDDDD 7+3 (if 8+3, no padding)

PPPDDDD 4+3 to PPPDDDDX 4+4

Because 47%4 = 3; 3<4 (1+P)

(47+1)%4 == 0 , add a padding block

ZFS import failed

First, you can backup each block device by ddrescue

zpool import 8251205345118591 94

cannot import 'tank': I/O error, Recovery can be attempted

by executing 'zpool import -F tank'. A scrub of the pool is strongly recommended after recovery

zpool import -n -F 8251205345118591 94

echo $?

0

zpool import -F 8251205345118591 94

Allow rollback to arbitrary txg

Patch

```
# zdb -e -ul tank  | grep txg                                          #Get txg id from zpool
# zdb -ul /dev/disk/by-id/scsi-3500xxx-part1 |grep "txg "             #Get txg id from the device

# zpool import -o readonly=on -d /dev/disk/by-id -T $old_txg tank     #scan all data

# echo 0 > /sys/module/zfs/parameters/spa_load_verify_data            #Skip data check
# zpool import -o readonly=on -o cachefile=none -d /dev/disk/by-id -m  -T $old_txg tank    # check metadata and import
or
# zdb -ec -t $old_txg tank                                            # check metadata by zdb
```

Force import mode

```
# echo 0 > /sys/module/zfs/parameters/spa_load_verify_metadata
# echo 0 > /sys/module/zfs/parameters/spa_load_verify_data
# zpool import -o readonly=on -o cachefile=none -d /dev/disk/by-id -m tank

# echo 1 > /sys/module/zfs/parameters/zfs_max_missing_tvds (echo 2 or echo 3)
# zpool import -o readonly=on -o cachefile=none -d /dev/disk/by-id -m tank

#with the old txg id
# zpool import -n -o readonly=on -o cachefile=none -d /dev/disk/by-id -m  -T $old_txg tank
```

After recovery

The data is OK

cksum corrupted ?

```
        NAME                      STATE     READ WRITE CKSUM
        ost_9                     ONLINE       0     0 3.90K
          raidz3-0                ONLINE       0     0 7.80K
            scsi-35000cca2940159cc  ONLINE       0     0     0
            scsi-35000cca29400b9d0  ONLINE       0     0     0
            scsi-35000cca29400010c  ONLINE       0     0     0
            scsi-35000cca2940114e8  ONLINE       0     0     0
            scsi-35000cca29400bc44  ONLINE       0     0     0
            scsi-35000cca29401052c  ONLINE       0     0     0
            scsi-35000cca294008f94  ONLINE       0     0     0
            scsi-35000cca294004a44  ONLINE       0     0     0
            scsi-35000cca2940085b0  ONLINE       0     0     0
            scsi-35000cca294011600  ONLINE       0     0     0
            scsi-35000cca294010420  ONLINE       0     0     0
            scsi-35000cca29400bd88  ONLINE       0     0     0
            scsi-35000cca29400d938  ONLINE       0     0     0
            scsi-35000cca26fdaf820  ONLINE       0     0     0
            scsi-35000cca26fde8b0c  ONLINE       0     0     0
            scsi-35000cca26fd648d4  ONLINE       0     0     0
            scsi-35000cca26fdef764  ONLINE       0     0     0
            scsi-35000cca26fe049e0  ONLINE       0     0     0
            scsi-35000cca26fe04b10  ONLINE       0     0     0

errors: Permanent errors have been detected in the following files:

        ost_9/ost_9:/O/0/d28/42401724
        ost_9/ost_9:/O/0/d27/42401755
        ost_9/ost_9:/O/0/d29/42401757
        ost_9/ost_9:/O/0/d30/42401758
```

SAS phy error, see you soon

- Verify the error path for the log mapping to the JBOD sub-expander is accurate
  - Replace the faulty component
- SAS PHY error issue reported by BMC event log (NEW Feature)
  - Easy to monitor from BMC in the future

206	40	0x00000573	0x00000574	0x00000000	0x00000000
207	41	0x00000005	0x00000001	0x00000000	0x00000000
208	42	0x0000008d	0x0000008d	0x00000000	0x00000000
209	43	0x00000005	0x00000001	0x00000000	0x00000000
210	44	0x00000005	0x00000002	0x00000000	0x00000000
211	45	0x0000000c	0x0000000c	0x00000000	0x00000000
212	46	0x00000000	0x00000000	0x00000000	0x00000000
213	47	0x00000016	0x00000016	0x00000000	0x00000000

- Another vendor limit the number of connections to a JBOD
  - Multiple OSS accessing a single JBOD can sometimes trigger PHY errors
    - Eg: high-availability
  - Limiting access to single OSS can prevent PHY errors
    - Test many times
  - The others no this issue in the same case and the same OEM hardware

Replace some characters not display

```
[root@localhost 10:27 ~]# tail -n 1 test.log
0 Voltage Core 0 12V Rail Loc: left-PSU 12.34  OK s o quit)
[root@localhost 10:27 ~]# tail -n 1 test.log | awk '{print $NF}'
OK
[root@localhost 10:27 ~]# tail -n 1 test.log | hexdump -C
00000000  30 20 56 6f 6c 74 61 67  65 20 43 6f 72 65 20 30  |0 Voltage Core 0|
00000010  20 31 32 56 20 52 61 69  6c 20 4c 6f 63 3a 20 6c  | 12V Rail Loc: l|
00000020  65 66 74 2d 50 53 55 20  31 32 2e 33 34 20 50 72  |eft-PSU 12.34 Pr|
00000030  65 73 73 20 6f 20 71 75  69 74 29 08 08 08 08 08  |ess o quit).....|
00000040  08 08 08 08 08 08 08 08  20 08 20 4f 4b 20 0a     |........ . OK .|
0000004f
[root@localhost 10:28 ~]# tail -n 1 test.log | cat -v
0 Voltage Core 0 12V Rail Loc: left-PSU 12.34 Press o quit)^H^H^H^H^H^H^H^H^H^H^H^H^H^H ^H OK
```

Bad output

hexdump -ve '1/1 "%02X "' test.log | sed -e 's/50 72//g' | /usr/bin/xxd -r -p

Dump text to hex data -> replace hex text data -> convert to binary from hex text data

This vendor command line add a lot of bad string and these bad string could not be display in terminal

Auto replace G-list table

```
=== START OF INFORMATION SECTION ===
Vendor:               WDC
Product:              WUH721816AL5204
Revision:             C232
Compliance:           SPC-5
User Capacity:        16,000,900,661,248 bytes [16.0 TB]
Logical block size:   512 bytes
Physical block size:  4096 bytes
LU is fully provisioned
Rotation Rate:        7200 rpm
Form Factor:          3.5 inches
Logical Unit id:      0x5000cca2a1765480
Serial number:        2CJ32DBP
Device type:          disk
Transport protocol:   SAS (SPL-3)
Local Time is:        Mon Sep  2 08:47:25 2024 HKT
SMART support is:     Available - device has SMART capability.
SMART support is:     Enabled
Temperature Warning:  Enabled

=== START OF READ SMART DATA SECTION ===
SMART Health Status: OK

Grown defects during certification <not available>
Total blocks reassigned during format <not available>
Total new blocks reassigned <not available>
Power on minutes since format <not available>
Current Drive Temperature:     39 C
Drive Trip Temperature:        85 C

Manufactured in week 02 of year 2021
Specified cycle count over device lifetime:  50000
Accumulated start-stop cycles:  8
Specified load-unload count over device lifetime:  600000
Accumulated load-unload cycles:  1345
Elements in grown defect list: 473

Error counter log:
           Errors Corrected by           Total   Correction     Gigabytes    Total
               ECC          rereads/    errors   algorithm      processed    uncorrected
           fast | delayed   rewrites  corrected  invocations   [10^9 bytes]  errors
read:          0        1         0         1   18002504     261438.682           0
write:         0      670         0       670     738827     106727.972         391
verify:        0        0         0         0       9101          0.000           0

Non-medium error count:        0

No Self-tests have been logged
```

Manual reassign

blk_update_request: I/O error, dev sda, sector 16782858

SATA

hdparm --yes-i-know-what-i-am-doing --repair-sector 16782858 /dev/sdb

hdparm --read-sector 16782858 /dev/sdb

/dev/sdb:

reading sector 16782858: succeeded

hdparm --write-sector 16782858 /dev/sdb

re-writing sector 16782858: succeeded

Or

dd if=/dev/sdb bs=4096 skip=16782858 count=1 iflag=direct

Background long Failed in segment --> 7 43458 3253726808 [0x3 0x11 0x0]

SAS

sg_verify --lba=3253726808 /dev/sdk

Descriptor format, current; Sense key: Medium Error

Additional sense: Unrecovered read error

Descriptor type: Information: 0x00000000c1efee58

Descriptor type: Sense key specific: Actual retry count: 361

sg_reassign --address=3253726808 /dev/sdk

sg_reassign --grown /dev/sdk #(G-list table)

sg_verify --lba=3253726808 /dev/sdv

THANKS 谢谢 

Owned by All, Completed by All and Shared by All

共有 · 共为 · 共享