Hewlett Packard
Enterprise

# AN AGED CLUSTER FILE SYSTEM: PROBLEMS AND SOLUTIONS

Artem Blagodarenko

# WHAT CUSTOMERS ASK

Why zeroing the position can affect performance to the degree noted?

Can we get a review of the perf data captured in May, see what's causing the very slow e2fsck?

Whether we can enable the large_dir feature here or not ?

Something is not right with the system and it seen on the older OSTs.

This presentations is about Lustre FS with LDISKFS backend

# LDISKFS BLOCK ALLOCATOR

allocation window

unfragmented vs fragmented

Allocator processes whole disk trying to find large continuous range of blocks. Disks become larger, the problem becomes visible.

# SOLUTION – TUNE!

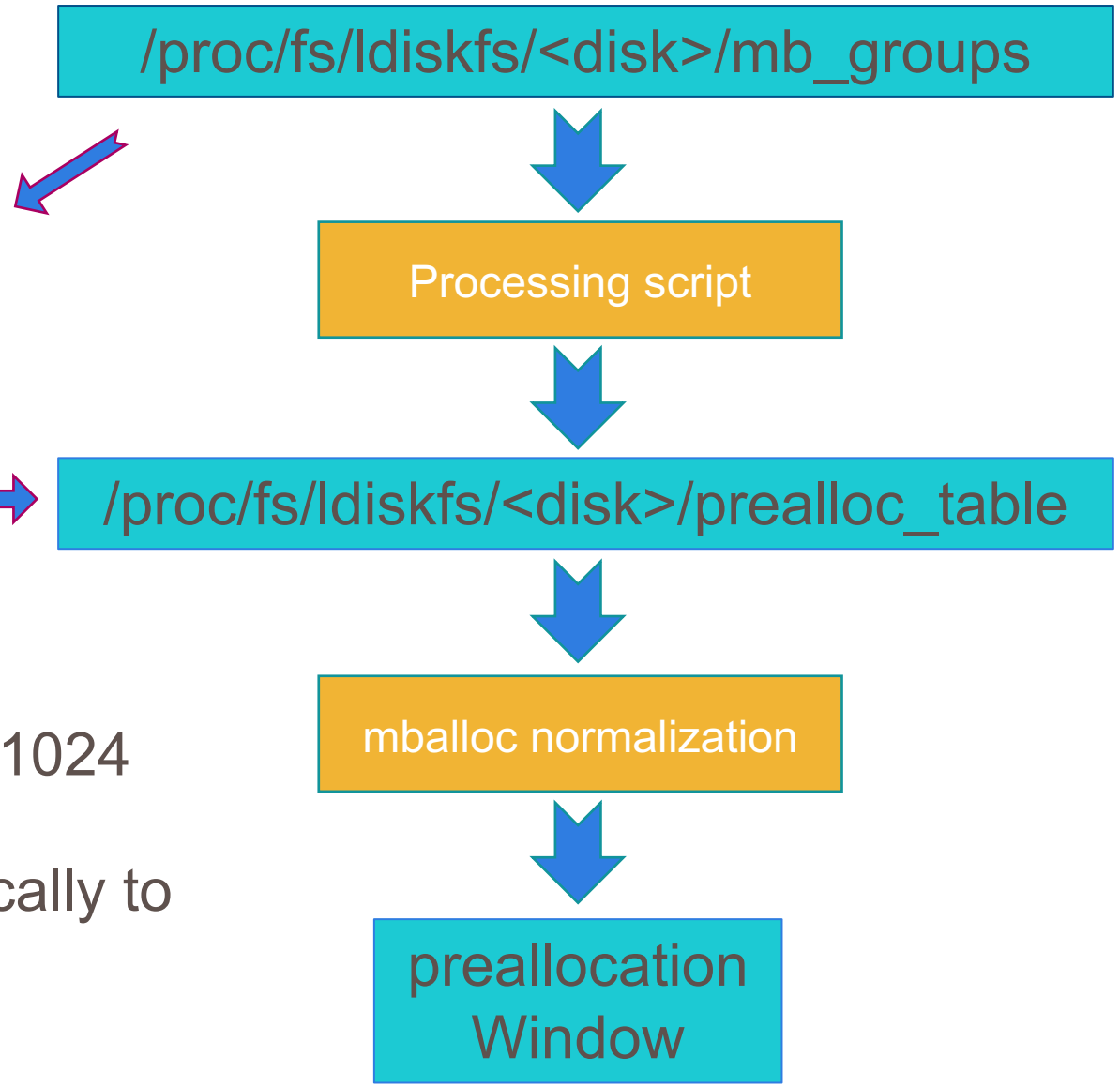
```
     [2^0                          … 2^13 ]
     [ 0 1 1 0 0 1 0 1 1 1 1 0 0 0 ]
     [ 1 0 1 1 2 2 1 1 3 1 1 1 0 0 ]
     [ 0 0 0 0 0 1 0 0 1 0 0 1 0 0 ]

1 2 4 8 16 32 64 128 256 512 1024 2048 4096
```

1 2 4 8 16 32 64 128 256 512 1024

Offset + requested size = 3000

Normalized request 4096 but is limited to 1024

prealloc_table should be adjusted periodically to reflect current state



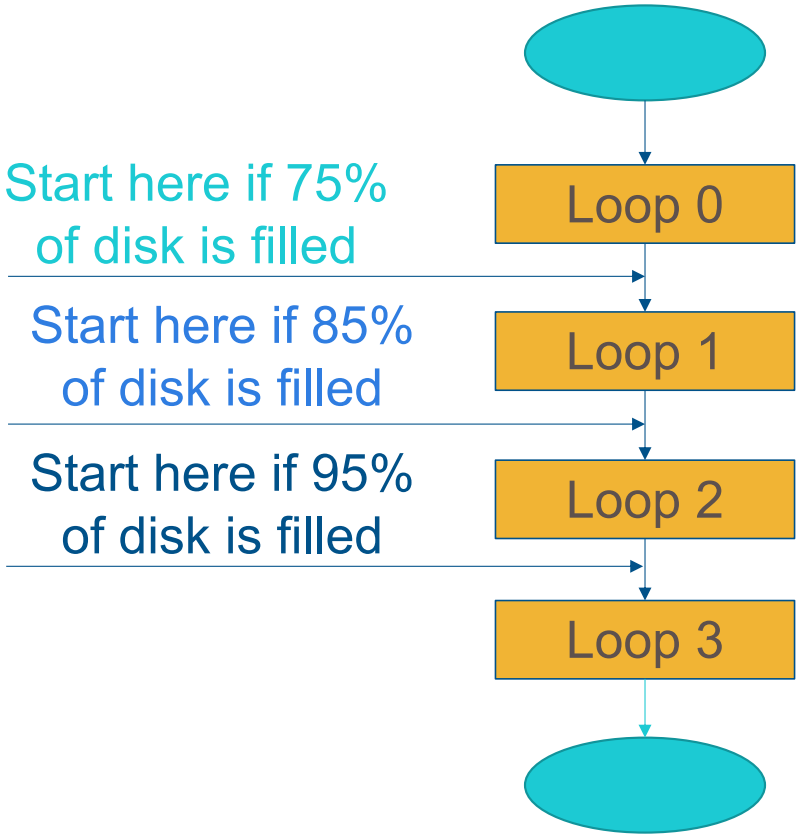
# SOLUTION - SIMPLIFY

## Loops Skipping Solution

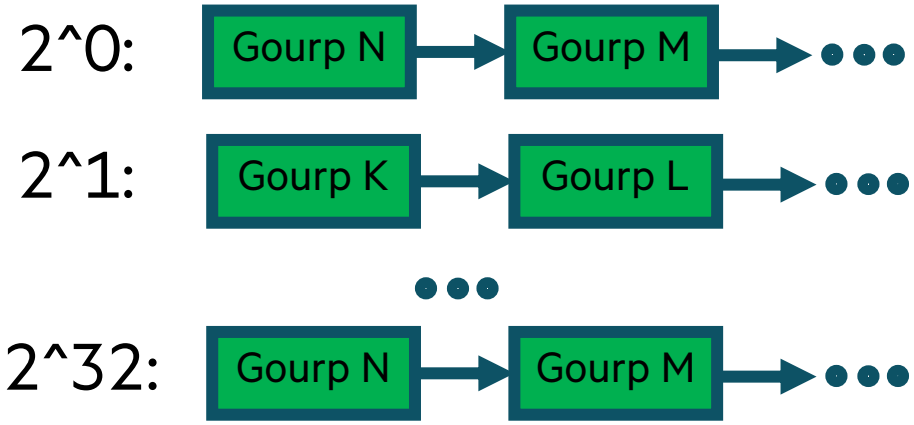
based on FS condition

```
echo "75" > /sys/fs/ldiskfs/loop1/mb_c1_threshold
echo ”85" > /sys/fs/ldiskfs/loop1/mb_c2_threshold
echo ”95" > /sys/fs/ldiskfs/loop1/mb_c3_threshold
```

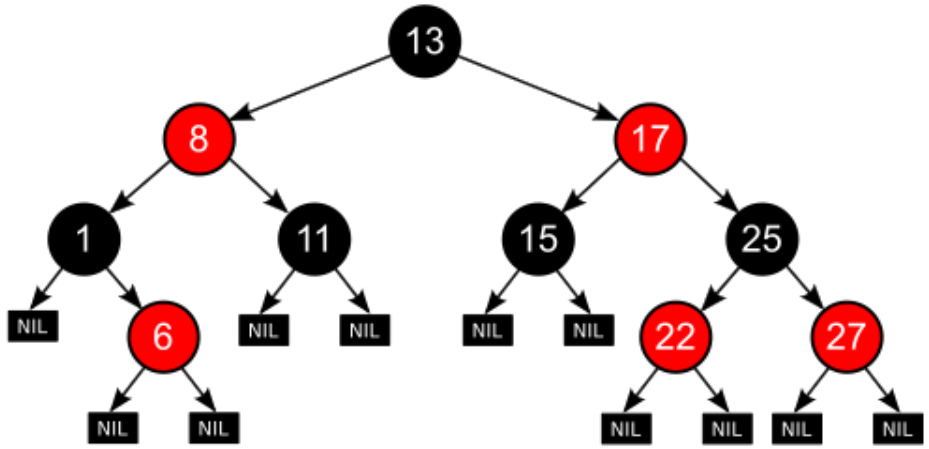
force to skip *useless* loops



# SOLUTION – REWRITE ALLOCATOR




for cr1 there is a list for each order. Get required group for O(1)




for cr2 there is a rb tree of groups sorted by largest fragment size. O(log)

LU-14438

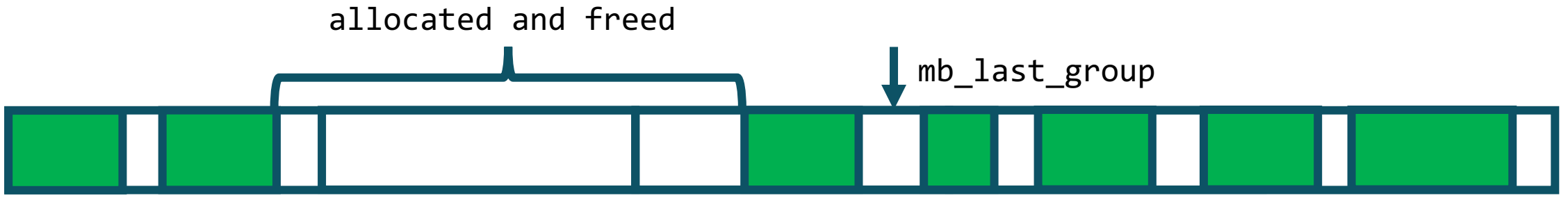
https://www.spinics.net/lists/linux-ext4/msg77184.html

# MB_LAST_GROUP. PROBLEM

pdsh -g oss 'cat /proc/fs/ldiskfs/*/mb_last_group' | sort"

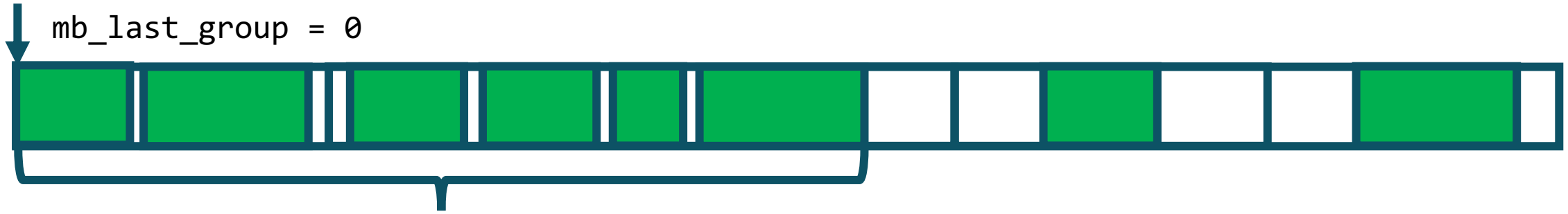
Obdfilter shows 30% performance drop for OSTs with high mb_last_group

Spinning hard drive is faster at the start and slower at the end




```
echo 0 > /proc/fs/ldiskfs/*/mb_last_group
```

# MB_LAST_GROUP. SOLUTIONS




/proc/fs/ldiskfs/*/mb_groups

↓

Heuristic algorithm script

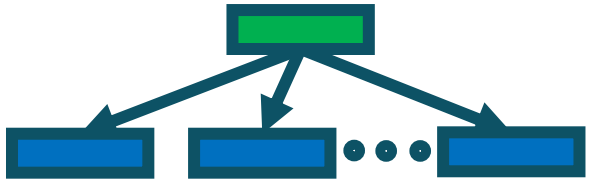
↓

/proc/fs/ldiskfs/*/mb_last_group

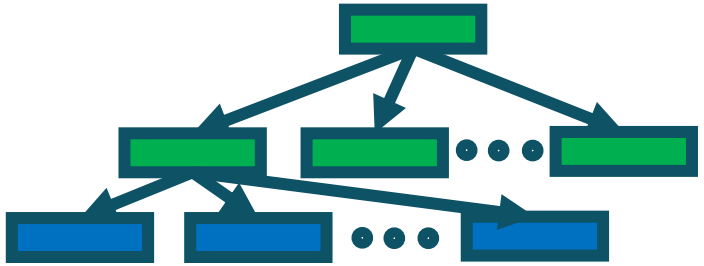
Solution based on new blocks allocator from LU-14438

# LARGE DIRECTORY(LU-11912)

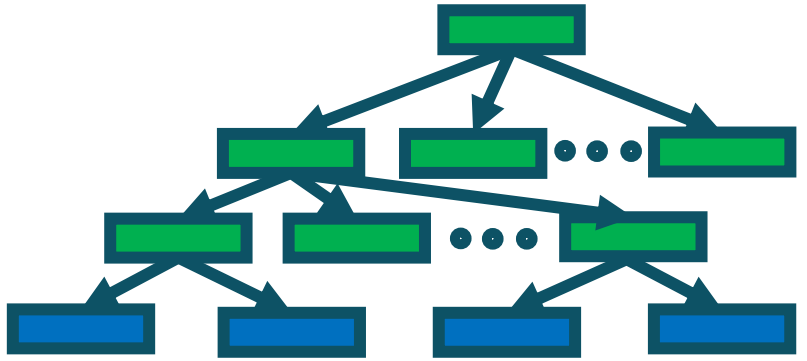
~100K (3MB)



~1M (30MB)



~10M (300MB)



New create of unlink goes to different leaf - random 4kB IOPS

```
Reduce LUSTRE_DATA_SEQ_MAX_WIDTH from ~4B to ~33M to
limit the number of objects under /O/[seq]/d[0..31]
dir on OSTs.
```

Sometimes there is a requirement to have a lot of files in the same directory

# DIRECTORY SHRINK

A directory can only grow

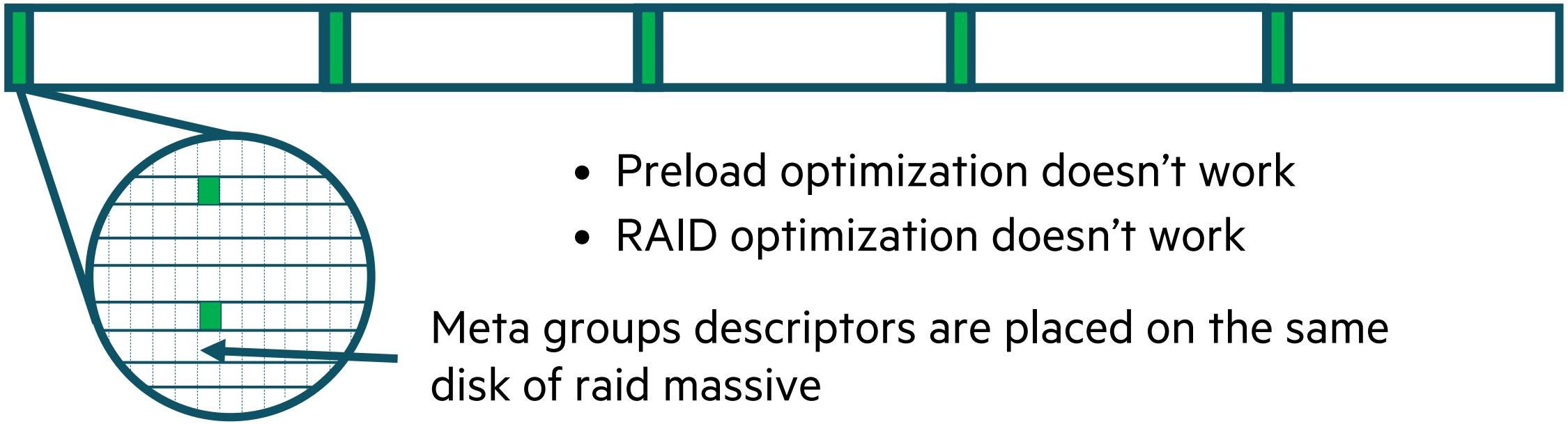
There is a patch “ext4: shrink directory when last block is empty”

e2fsck –fD as workaround

# META_BG AND META GROUPS DESCRIPTORS

Without the meta_bg option all group descriptors are placed in the goup 0

group descriptors are split across a target



- Preload optimization doesn’t work
- RAID optimization doesn’t work

Meta groups descriptors are placed on the same disk of raid massive

# META GROUPS DESCRIPTORS OPTIMIZATION (LU-15002)


The next steps allow to the creation of continuous group descriptors for the first 256TB and use meta_bg for all other groups.

1. Create < 256 TB partition without the meta_bg flag
2. Extend the partition to the whole disk

These steps can be done manually or mkfs can be modified.

To solve this meta_bg problem ext4 and ldiskfs layout must be changed completely

As alternative bigalloc option can be used

# UTILITIES: F2SCK

E2fsck spends 80% of time on pass1

70 files changed, 3335 insertions(+), 393 deletions(-)

LU-8465: Introduce parallel fsck to e2fsck pass1

5x total time reduction

Read inodes in parallel, the fix is still serialized

# UTILITIES: E2IMAGE

```
e2image -Q /dev/md66 /mnt/backup/md66.qcow2
```

+

qemu-nbd -c
/dev/nbd1 ./md66.qcow2
e2fsck -pvf /dev/nbd1
qemu-nbd -d /dev/nbd1

```
#define QCOW_MAX_REFTABLE_SIZE (1024 * MiB)
#define QCOW_MAX_L1_SIZE (1024 * MiB)
```

e2image -r hda1.qcow2 hda1.raw
e2fsck -pvf hda1.raw
with "[PATCH] e2image: fix overflow in l2 table processing”

```
[PATCH] e2image: fix overflow in
l2 table processing
```

or

e2image -r /dev/md66 - | bzip2 -c > /mnt/backup/md66.raw.bz2

# IS IT TIME FOR A WRITECONF?

```
#umount /mnt/fs2mds/
#mount -t lustre -o nosvc,loop /tmp/lustre-mdt1 /mnt/lustre-mds1/

#lctl replace_nids snx11168-MDT0006
10.100.105.3@o2ib4,10.101.105.3@o2ib4001:10.100.105.2@o2ib4,10.100.105.2@o2ib
4000
#lctl replace_nids snx11168-MDT0005  lctl replace_nids snx11168-MDT0005
10.100.105.2@o2ib4,10.100.105.2@o2ib4000:10.100.105.3@o2ib4,10.101.105.3@o2ib
4001

# tunefs.lustre --nolocallogs /tmp/lustre-mdt1
```

# CLEANUP CONFIGURATION FILES

```
#umount /mnt/fs2mds/
 #mount -t lustre -o nosvc,loop /tmp/lustre-mdt1 /mnt/lustre-mds1/
#lctl clear_conf /tmp/lustre-mdt1
```

- Cleans up configuration files stored in the CONFIGS/ directory of any records marked SKIP.
- If the device name is given, then the specific logs for that filesystem (e.g. testfs-MDT0000) is processed.
- Otherwise, if a filesystem name is given then all configuration files for the specified filesystem are cleared.

# QUESTIONS?