TRIM Explained

OpenZFS Developer Summit

Brian Behlendorf
behlendorf@llnl.gov

November 4th, 2019

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

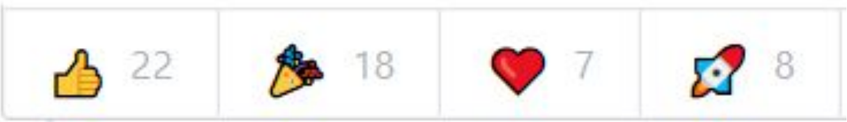
Lawrence Livermore National Laboratory

What is TRIM?

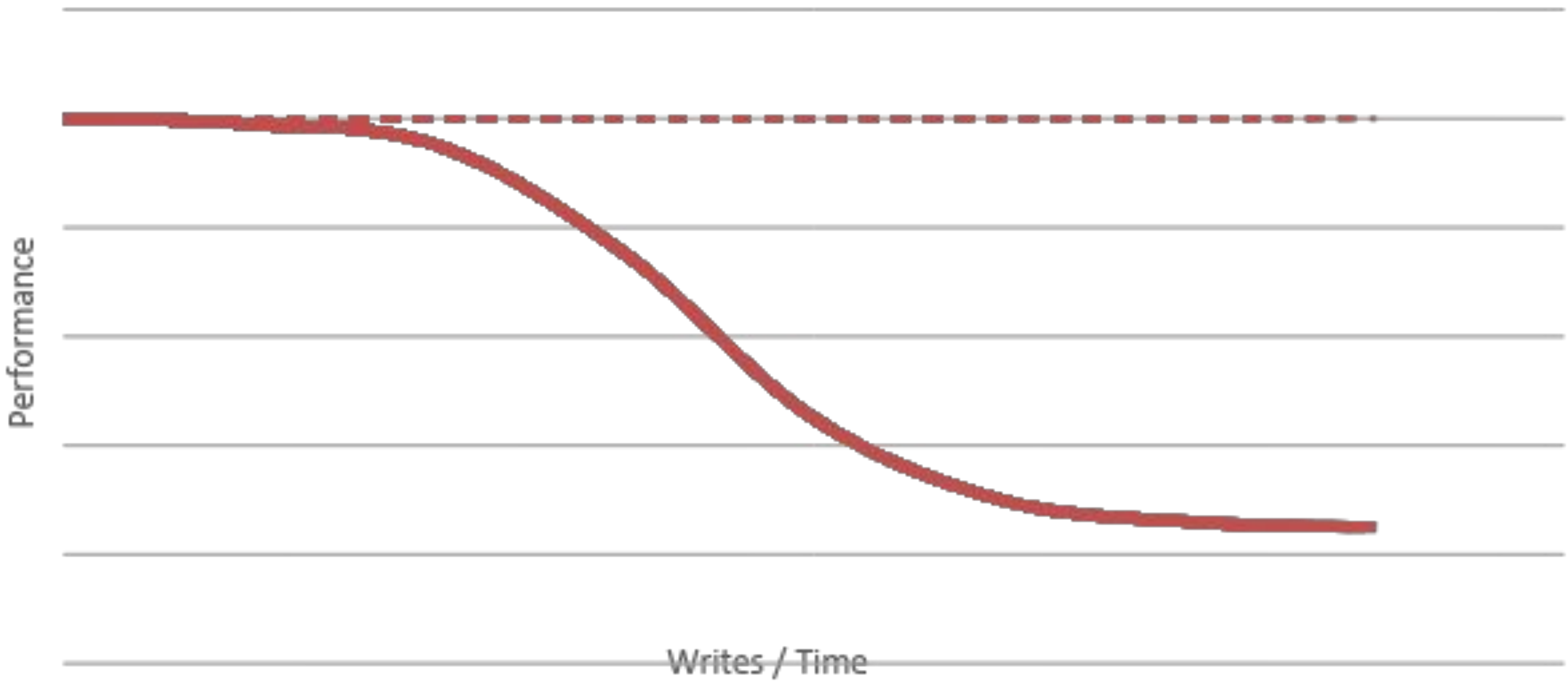
- TRIM is a command which allows the filesystem to notify the storage device which blocks are no longer in use.

- You may know it by a different name:
  - TRIM – ATA command set
  - UNMAP – SCSI command set
  - DISCARD – Linux terminology

- One of the most requested ZFS features
  - https://github.com/zfsonlinux/zfs/pull/8419



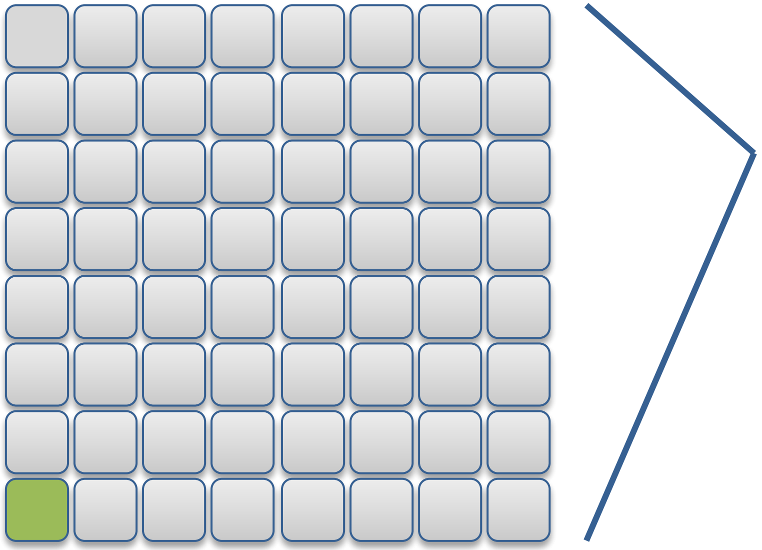
Expected SSD Performance



Decreased Performance Over Time

NAND Limitations and Block Sizes

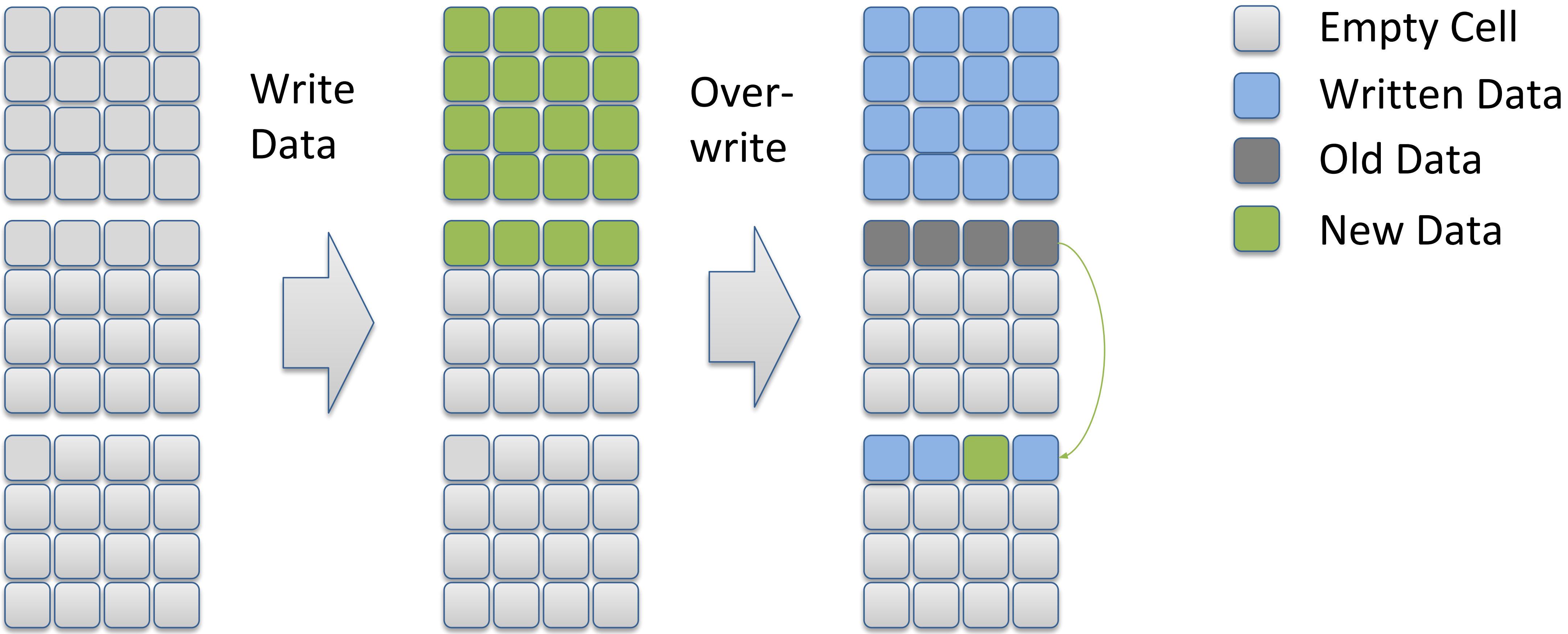
- Only empty pages may be written
- Only entire blocks may be erased
- Erasing a block is slow



256K - 4M Erasure Block

4K - 16K Page

Write Amplification



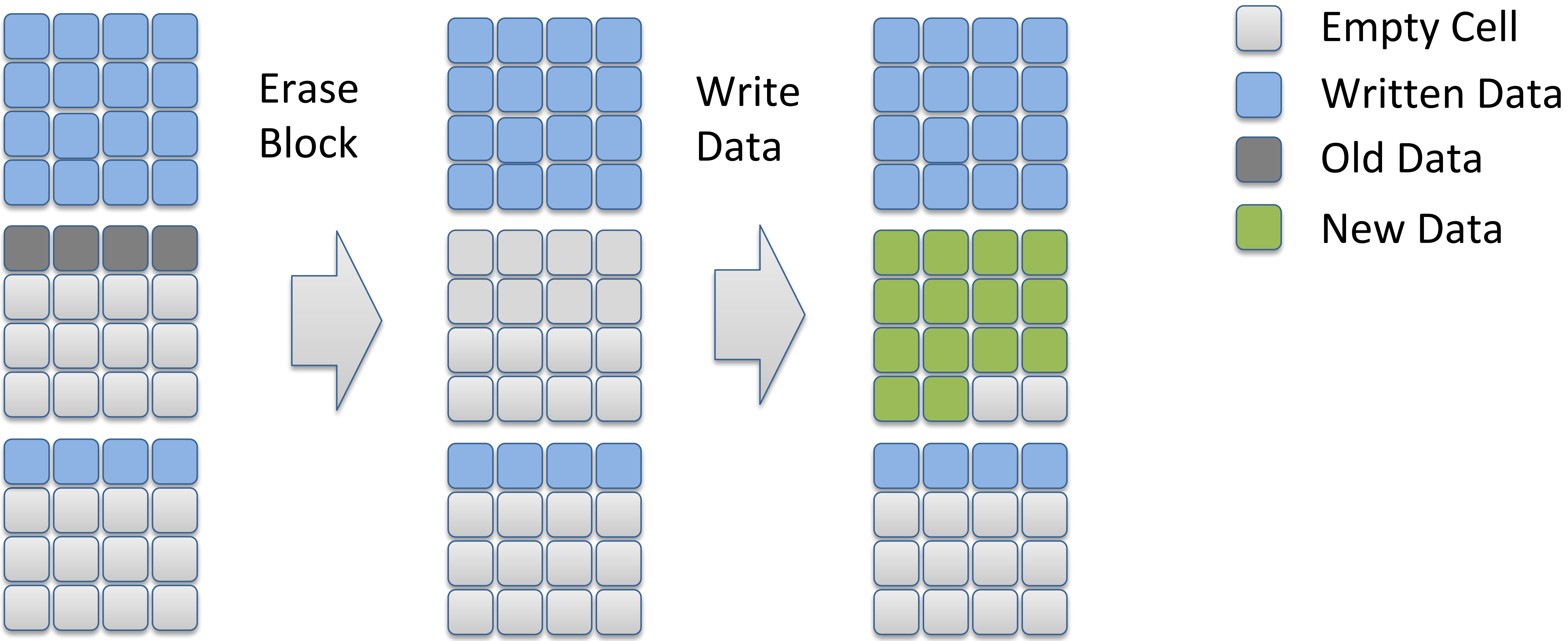
Write Data

Over-write

- ⬜ Empty Cell
- 🟦 Written Data
- ⬛ Old Data
- 🟩 New Data

Writes may be inflated since pages cannot be overwritten

Garbage Collection



Erase Block

Write Data

- ⬜ Empty Cell
- 🟦 Written Data
- ⬛ Old Data
- 🟩 New Data

Full erase/write cycles are expensive

Filesystem Support for TRIM

- Must be implemented in the filesystem; only it knows which sectors are in use
- Filesystems were originally optimized for HDDs; no need to notify device of unused sectors
- Efficient management of SSDs requires TRIM; internally the device can only do so much
- Motivations:
  - Reduced write amplification (fewer writes)
  - Higher write throughput (less read-erase-modify)
  - Increased device longevity (finite erase-write cycles)
- Automatic online TRIM support was added to: Ext4, Btrfs, FAT, JFS, XFS
- But it's tricky, often it is disabled by default due to performance concerns

TRIM must be implemented in the filesystem layer

Revisiting the Design

- Existing versions of TRIM worked well, why another version?
  - FreeBSD - added TRIM in version 9.2 (2012)
  - Nexenta - added TRIM to NexentaStor (2015)

- Design goals:
  - Online TRIM with negligible impact to running applications
  - Interoperates seamlessly with all existing OpenZFS features
  - Avoid introducing any duplicate functionality
  - Long term maintainability
  - Minimize platform specific dependencies

- We can learn from the previous versions

- And… a recent OpenZFS feature enables all of this

Building on the OpenZFS “vdev initialize” feature

- OpenZFS “vdev initialize” feature (aka eager zero)
  - Initializes all unallocated space in the background to prevent a first-access penalty
    - https://github.com/zfsonlinux/zfs/pull/8230
  - Introduced core infrastructure which could be extended for TRIM

- Key existing components:
  - Flexible administrative interface (CLI)
  - Ability to enable/disable new allocations for specified metaslab
  - Walks all unallocated space and submits I/Os for those vdev offsets

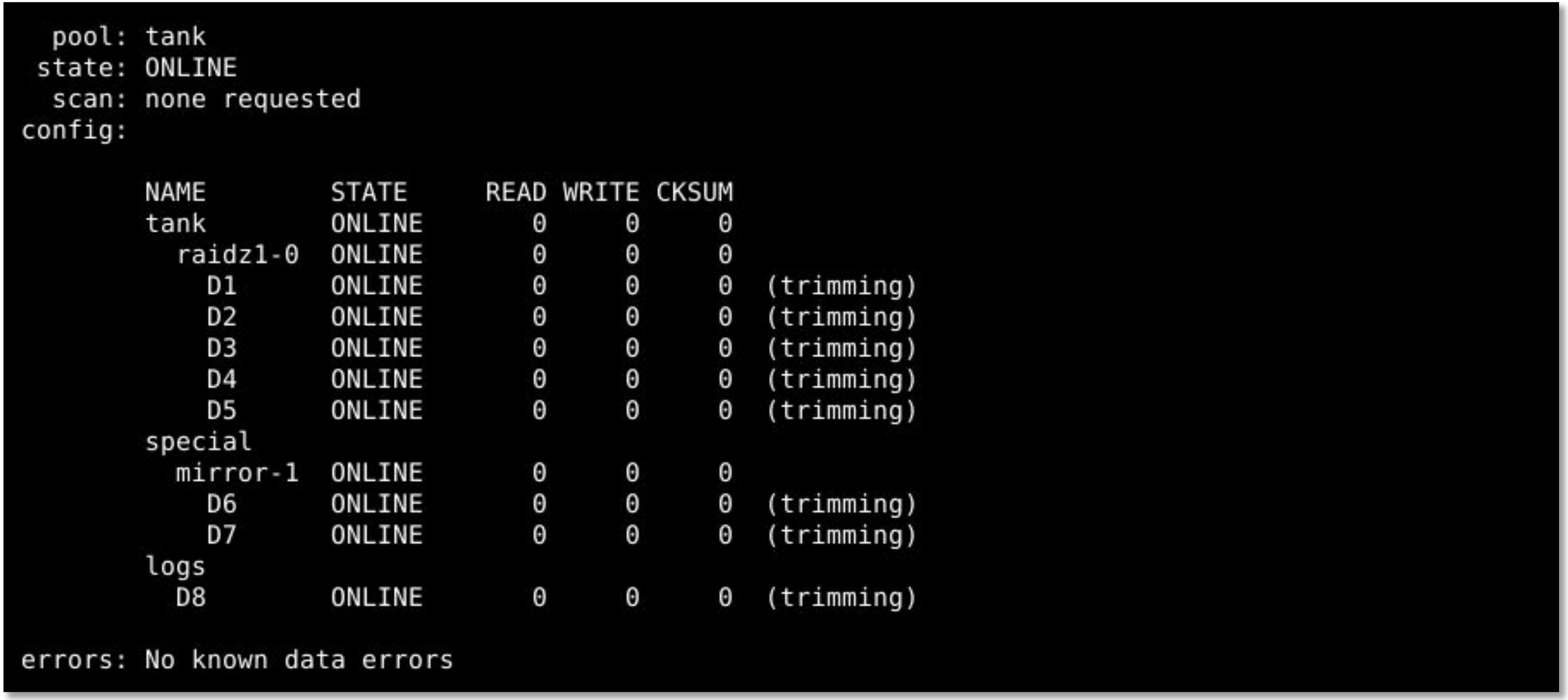
- New work required for TRIM
  - Modify or extend the existing code to be more generic
  - Add TRIM I/Os to the zio pipeline
  - Automatic background TRIM

Manual TRIM – “zpool trim”

```
zpool trim [-d] [-r rate] [-c | -s] pool [device…]
        -d     --secure     Request secure TRIM
        -r     --rate rate  Request specified TRIM rate
        -c     --cancel     Cancel running TRIM
        -s     --suspend    Suspend running TRIM
```

- Initiates an on-demand TRIM for all unallocated space in the pool
- Efficiently issues TRIM I/Os
  - Merges contiguous ranges in to one I/O
  - Skips very small ranges
  - Breaks large ranges into chunks
- Cancel / suspend / resume and in-progress TRIM
- TRIM state is preserved over a reboot (export / import)
- Clear progress reporting via “zpool status [-t]”
- CLI options are consistent with the “zpool initialize” command

“zpool status”

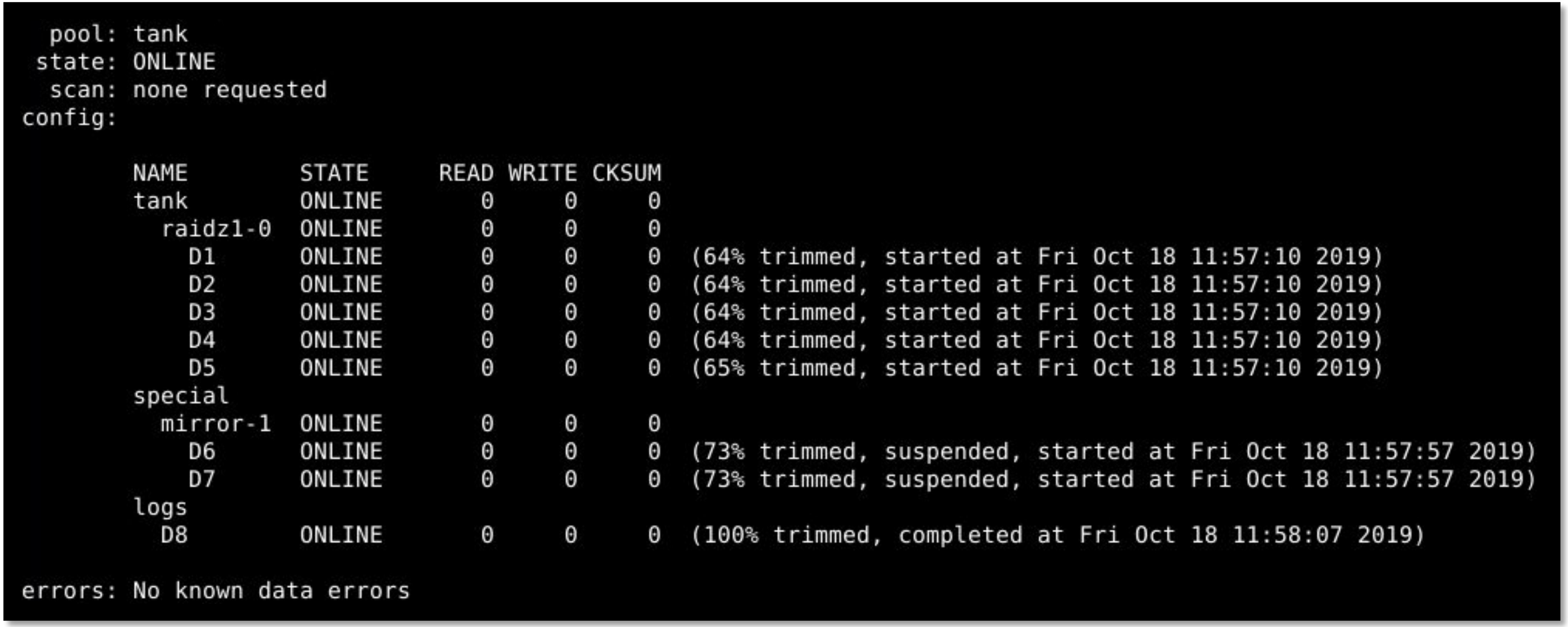


```
  pool: tank
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          raidz1-0  ONLINE       0     0     0
            D1      ONLINE       0     0     0  (trimming)
            D2      ONLINE       0     0     0  (trimming)
            D3      ONLINE       0     0     0  (trimming)
            D4      ONLINE       0     0     0  (trimming)
            D5      ONLINE       0     0     0  (trimming)
        special
          mirror-1  ONLINE       0     0     0
            D6      ONLINE       0     0     0  (trimming)
            D7      ONLINE       0     0     0  (trimming)
        logs
          D8        ONLINE       0     0     0  (trimming)

errors: No known data errors
```

“zpool status –t”



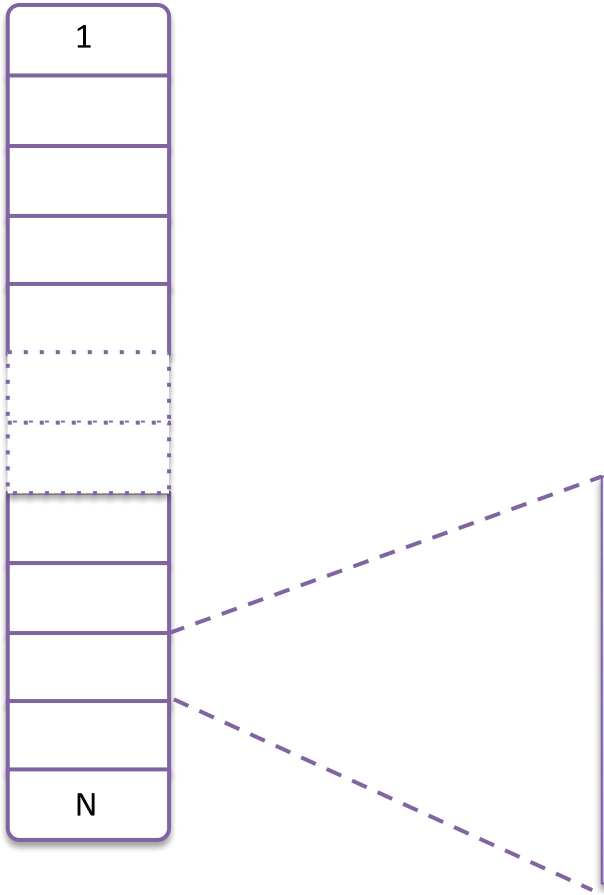
```
  pool: tank
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          raidz1-0  ONLINE       0     0     0
            D1      ONLINE       0     0     0  (64% trimmed, started at Fri Oct 18 11:57:10 2019)
            D2      ONLINE       0     0     0  (64% trimmed, started at Fri Oct 18 11:57:10 2019)
            D3      ONLINE       0     0     0  (64% trimmed, started at Fri Oct 18 11:57:10 2019)
            D4      ONLINE       0     0     0  (64% trimmed, started at Fri Oct 18 11:57:10 2019)
            D5      ONLINE       0     0     0  (65% trimmed, started at Fri Oct 18 11:57:10 2019)
        special
          mirror-1  ONLINE       0     0     0
            D6      ONLINE       0     0     0  (73% trimmed, suspended, started at Fri Oct 18 11:57:57 2019)
            D7      ONLINE       0     0     0  (73% trimmed, suspended, started at Fri Oct 18 11:57:57 2019)
        logs
          D8        ONLINE       0     0     0  (100% trimmed, completed at Fri Oct 18 11:58:07 2019)

errors: No known data errors
```

Metaslabs

- ms_allocatable is used to track the allocatable space
- metaslab_enable() / metaslab_disable()
  - Unavailable for new allocations
  - Up to max_disable_ms metaslabs disabled concurrently
  - Multiple threads may disable the same metaslab



```
struct metaslab {
        …
        range_tree_t        *ms_freed;                      /* already freed this syncing txg */
        range_tree_t        *ms_defer[TXG_DEFER_SIZE];      /* freed in a previous txg */
        range_tree_t        *ms_allocatable;                /* allocatable / free space */
        range_tree_t        *ms_trim;                       /* autotrim ranges */
        …
        uint64_t            ms_disabled;
}
```

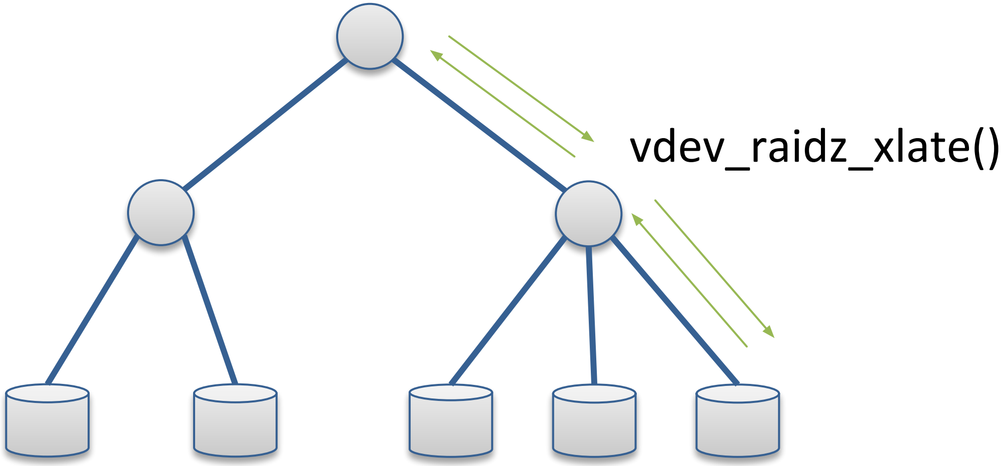
vdev_xlate()

- Translates a logical range to a physical range for the specified vdev

```
void vdev_xlate(vdev_t *vd, const range_seg64_t *logical_rs, range_seg64_t *physical_rs);
```

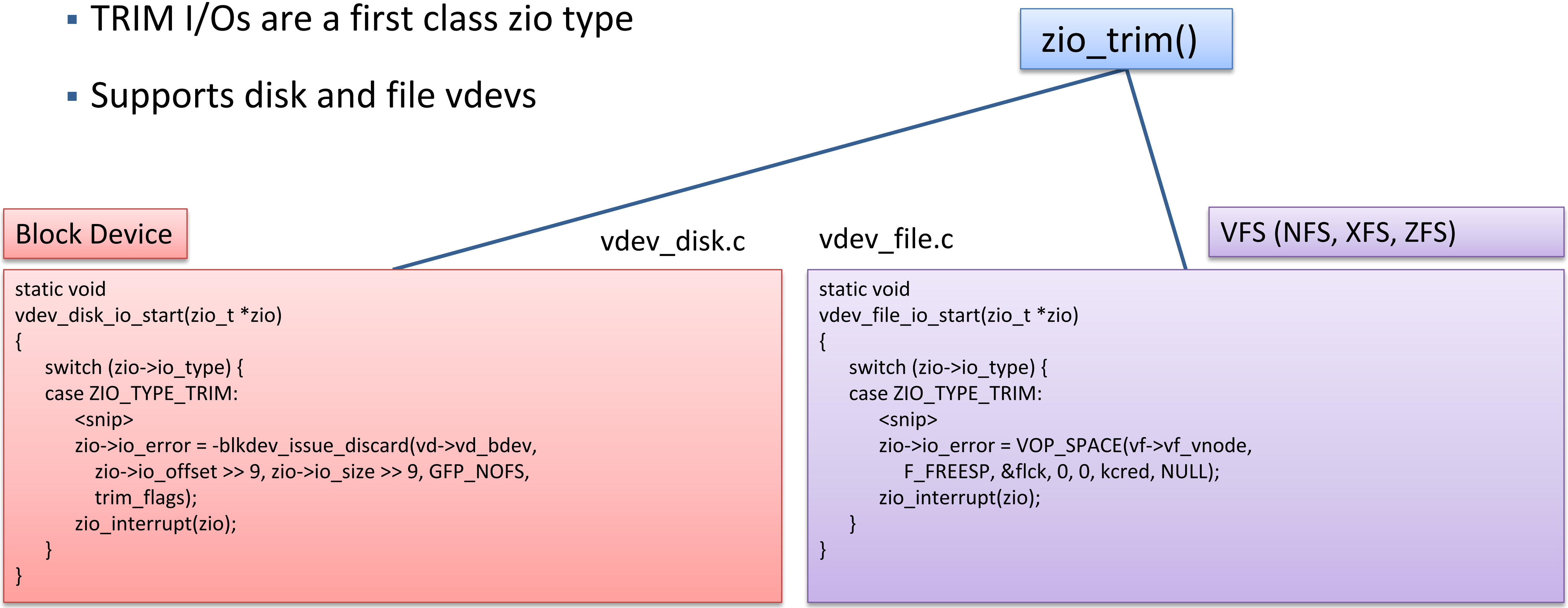
- Strategy:
  - Walks up each parent to the top-level vdev
  - Unwinds calling each parent's translation function
  - Returns the physical range for the vdev

- Translation Functions:
  - Callback: "vdev_ops->vdev_op_xlate()"
    - vdev_raidz_xlate()
    - vdev_default_xlate()



ZIO_TYPE_TRIM

- TRIM I/Os are a first class zio type
- Supports disk and file vdevs

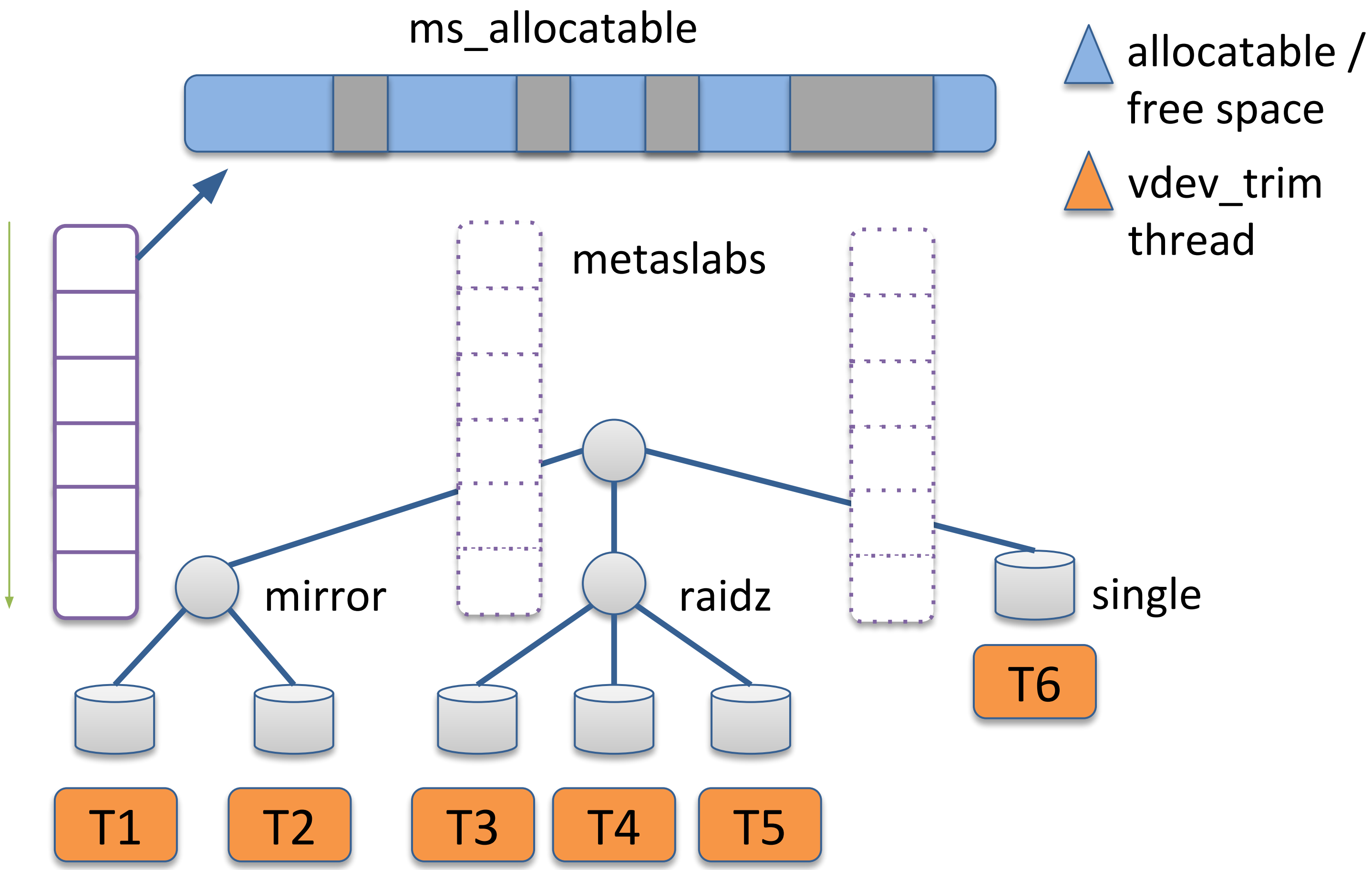


```
static void
vdev_disk_io_start(zio_t *zio)
{
    switch (zio->io_type) {
    case ZIO_TYPE_TRIM:
        <snip>
        zio->io_error = -blkdev_issue_discard(vd->vd_bdev,
           zio->io_offset >> 9, zio->io_size >> 9, GFP_NOFS,
           trim_flags);
        zio_interrupt(zio);
    }
}
```

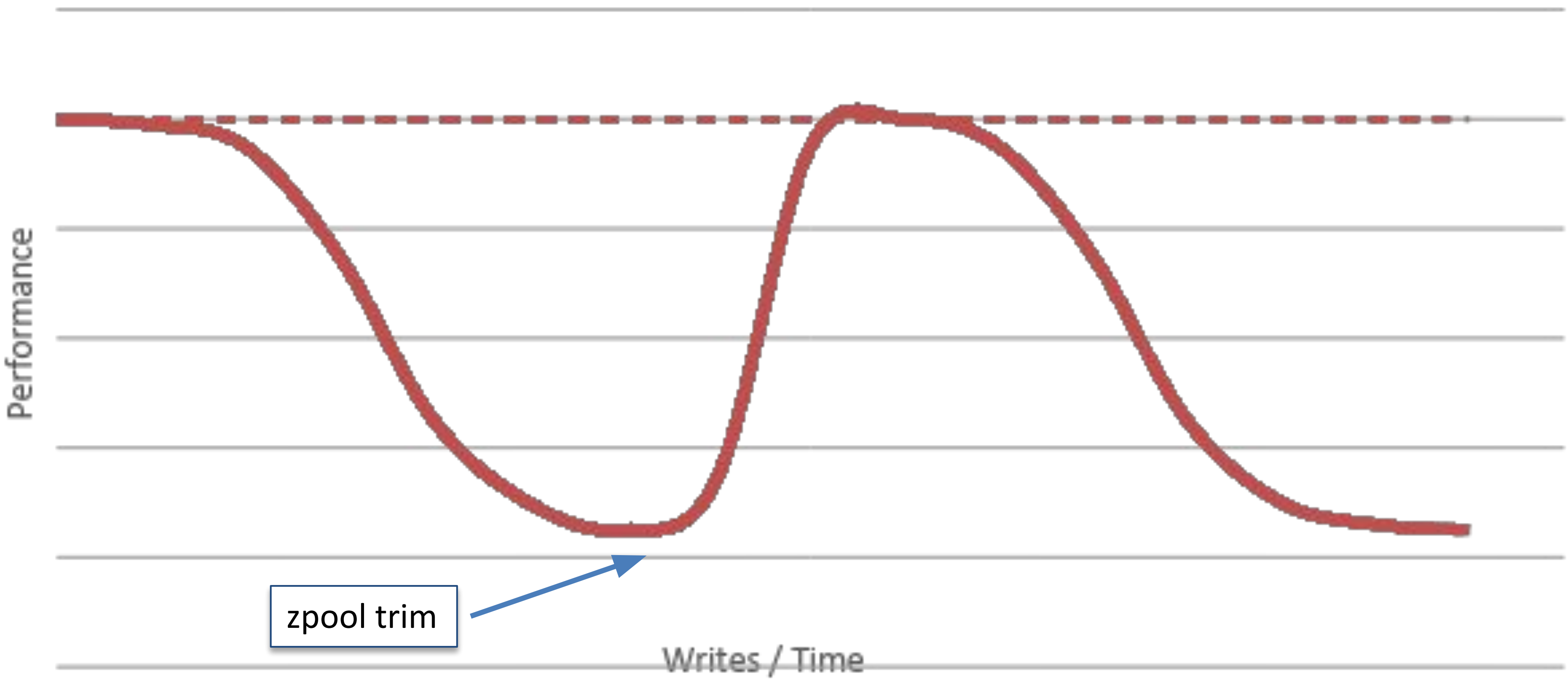
```
static void
vdev_file_io_start(zio_t *zio)
{
    switch (zio->io_type) {
    case ZIO_TYPE_TRIM:
        <snip>
        zio->io_error = VOP_SPACE(vf->vf_vnode,
            F_FREESP, &flck, 0, 0, kcred, NULL);
        zio_interrupt(zio);
    }
}
```

Manual TRIM – “zpool trim”

- One thread per leaf vdev
  - Relatively short lived
- Iterates sequentially over all of the metaslabs:
  - Disable metaslab allocations
  - Issues TRIM I/Os to leaf for all ranges in ms_allocatable
  - Wait for TRIM completion
  - Enable metaslab allocations
- Progress is saved in leaf-ZAP
- Can cancel / suspend / resume



Expected SSD Performance: “zpool trim”



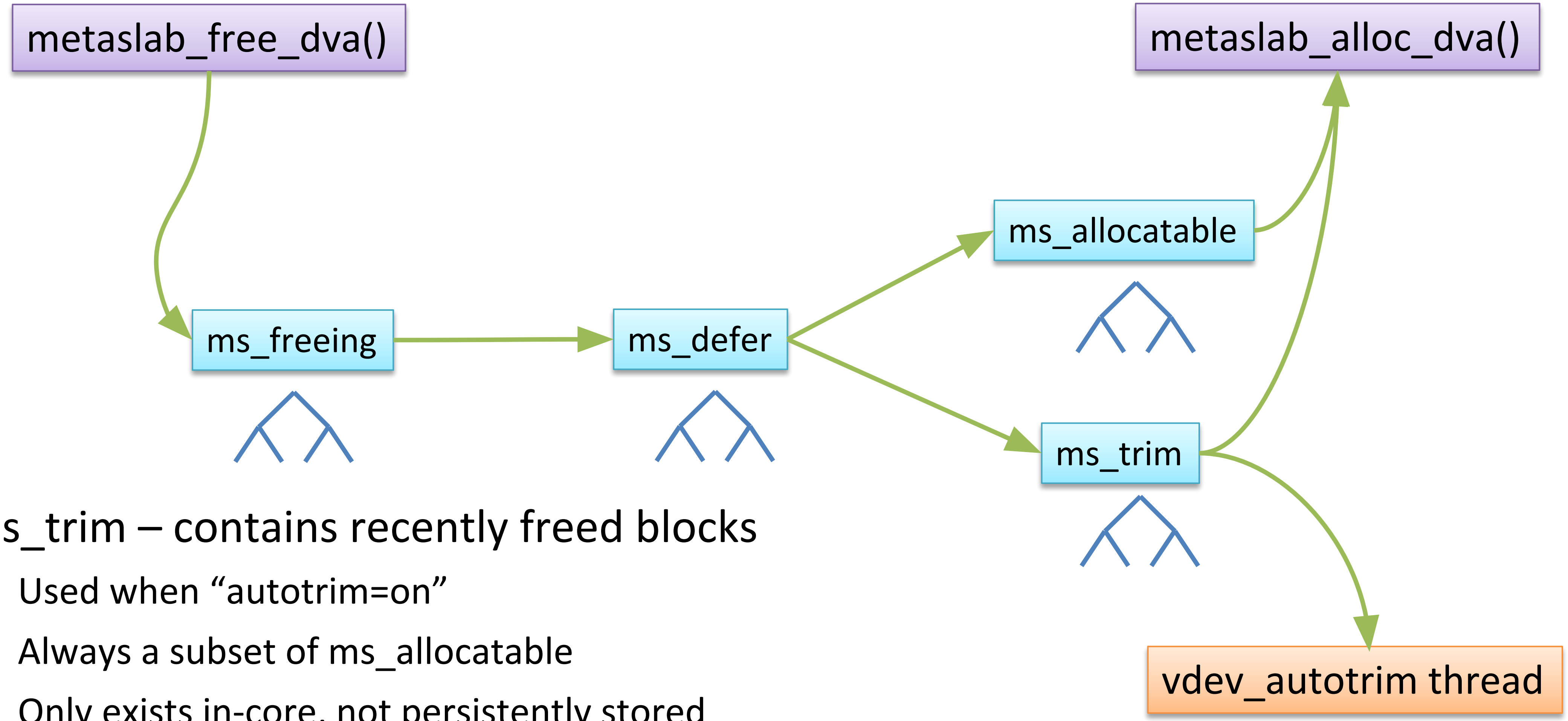
zpool trim

Performance Recovers but Degrades Again

Automatic TRIM

- Continuous background TRIM of all recently freed blocks
- Ensures underlying store always has an up to date mapping of allocated blocks
- Controlled by “autotrim=<on|off>” pool property

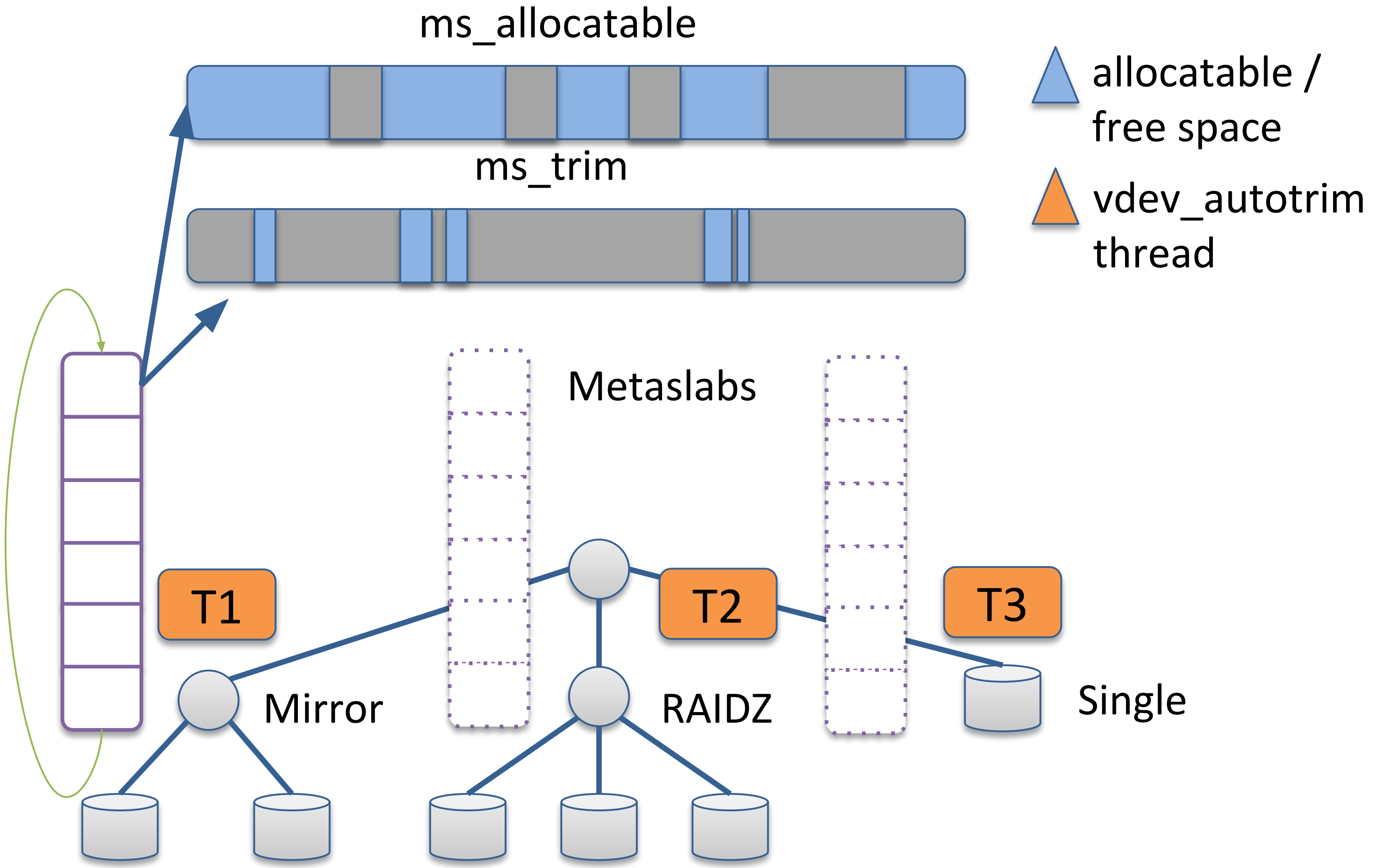
Free Block Life Cycle



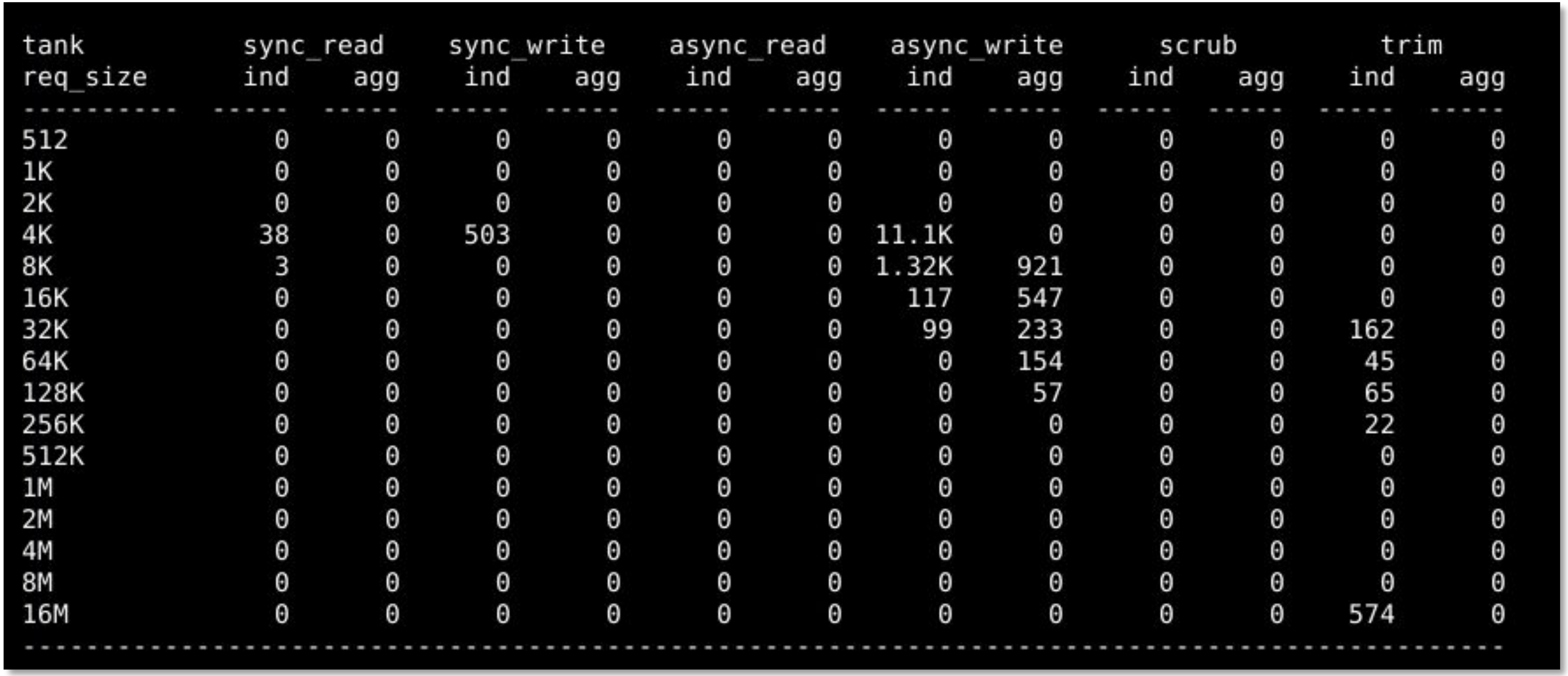
- ms_trim – contains recently freed blocks
  - Used when “autotrim=on”
  - Always a subset of ms_allocatable
  - Only exists in-core, not persistently stored

Automatic TRIM: “autotrim=on”

- One thread per top-level vdev
  - Long running
  - Only disables one metaslab a time
- Continuously iterates over metaslabs
  - Disable allocations
  - Swap and consume ms_trim
  - Issues TRIM I/Os to the children for all ranges in ms_trim
  - Wait for TRIM completion
  - Enable allocations
- Metaslab groups
  - Rate limiting; never forces a txg sync
  - At most one group processed per-txg
  - Allows time for freed block to be merged
  - Controlled by zfs_trim_txg_batch=32

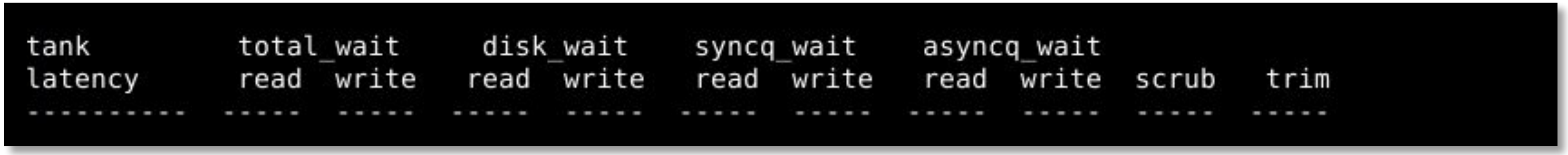


zpool iostat -r

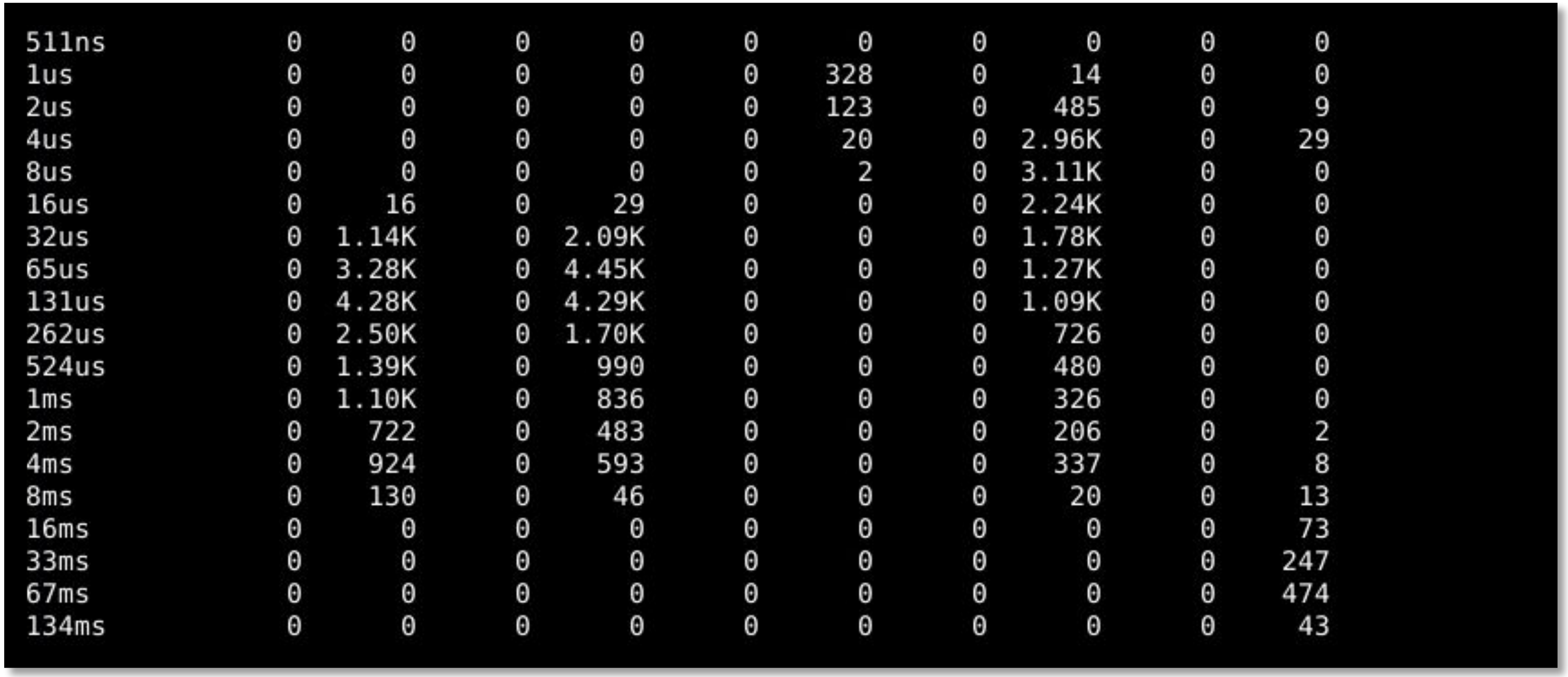


```
tank           sync_read    sync_write    async_read    async_write      scrub         trim
req_size       ind    agg    ind    agg    ind    agg    ind    agg    ind    agg    ind    agg
----------  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----
512             0      0      0      0      0      0      0      0      0      0      0      0
1K              0      0      0      0      0      0      0      0      0      0      0      0
2K              0      0      0      0      0      0      0      0      0      0      0      0
4K             38      0    503      0      0      0  11.1K      0      0      0      0      0
8K              3      0      0      0      0      0  1.32K    921      0      0      0      0
16K             0      0      0      0      0      0    117    547      0      0      0      0
32K             0      0      0      0      0      0     99    233      0      0    162      0
64K             0      0      0      0      0      0      0    154      0      0     45      0
128K            0      0      0      0      0      0      0     57      0      0     65      0
256K            0      0      0      0      0      0      0      0      0      0     22      0
512K            0      0      0      0      0      0      0      0      0      0      0      0
1M              0      0      0      0      0      0      0      0      0      0      0      0
2M              0      0      0      0      0      0      0      0      0      0      0      0
4M              0      0      0      0      0      0      0      0      0      0      0      0
8M              0      0      0      0      0      0      0      0      0      0      0      0
16M             0      0      0      0      0      0      0      0      0      0    574      0
----------------------------------------------------------------------------------------------
```

zpool iostat -w

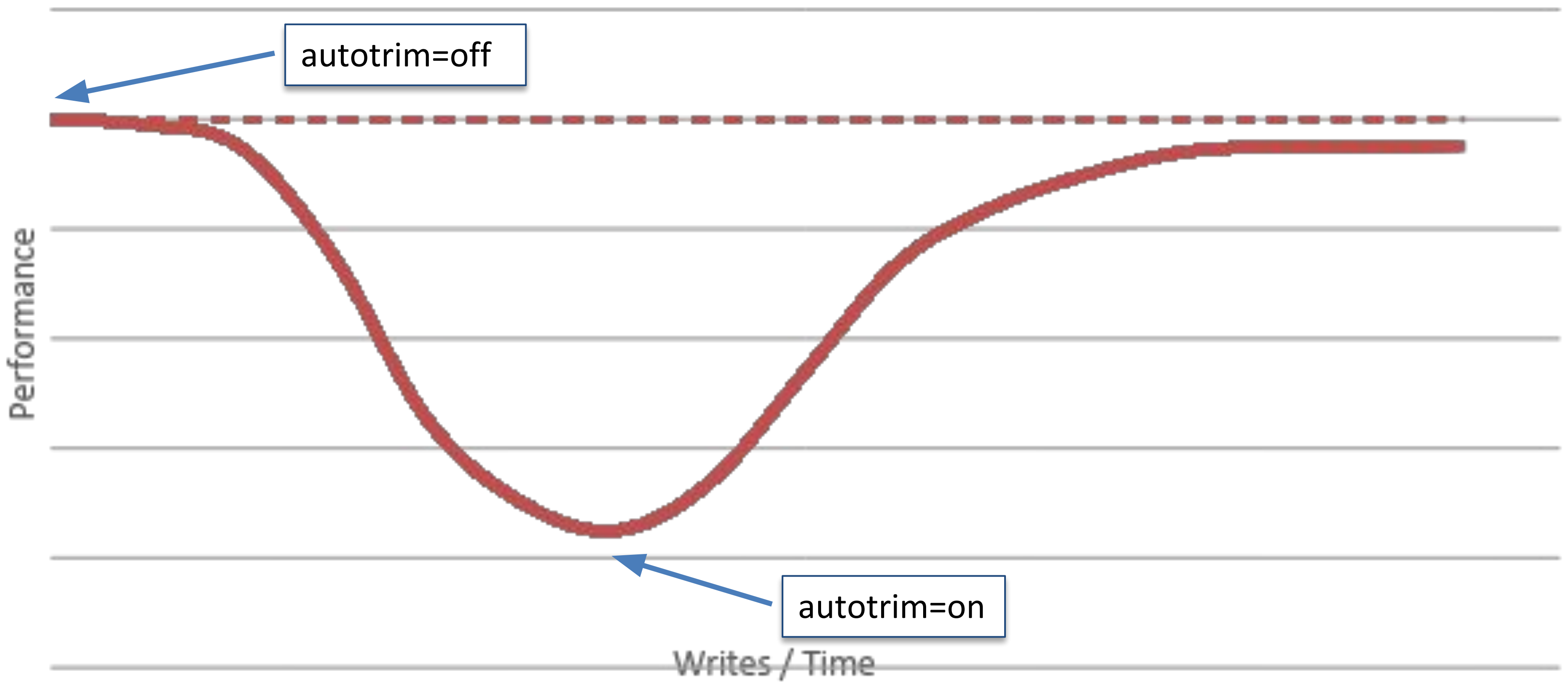


```
tank          total_wait     disk_wait    syncq_wait    asyncq_wait
latency        read  write   read  write   read  write   read  write  scrub   trim
----------  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----
```



```
511ns           0      0      0      0      0      0      0      0      0      0
1us             0      0      0      0      0    328      0     14      0      0
2us             0      0      0      0      0    123      0    485      0      9
4us             0      0      0      0      0     20      0  2.96K      0     29
8us             0      0      0      0      0      2      0  3.11K      0      0
16us            0     16      0     29      0      0      0  2.24K      0      0
32us            0  1.14K      0  2.09K      0      0      0  1.78K      0      0
65us            0  3.28K      0  4.45K      0      0      0  1.27K      0      0
131us           0  4.28K      0  4.29K      0      0      0  1.09K      0      0
262us           0  2.50K      0  1.70K      0      0      0    726      0      0
524us           0  1.39K      0    990      0      0      0    480      0      0
1ms             0  1.10K      0    836      0      0      0    326      0      0
2ms             0    722      0    483      0      0      0    206      0      2
4ms             0    924      0    593      0      0      0    337      0      8
8ms             0    130      0     46      0      0      0     20      0     13
16ms            0      0      0      0      0      0      0      0      0     73
33ms            0      0      0      0      0      0      0      0      0    247
67ms            0      0      0      0      0      0      0      0      0    474
134ms           0      0      0      0      0      0      0      0      0     43
```

Expected SSD Performance: “autotrim=on”

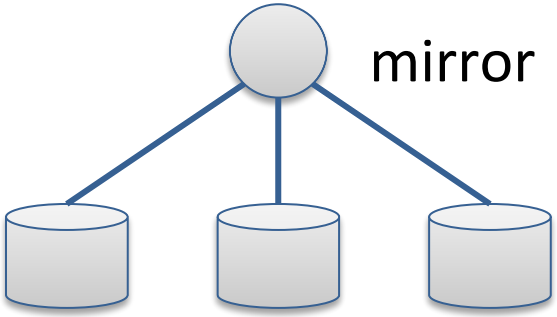


Performance Recovers and is Maintained

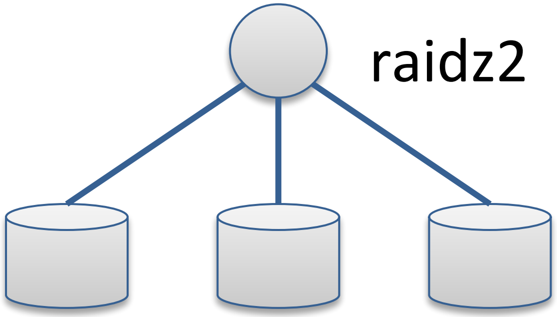
Real Performance Results

- Test case – Total time to copy the Linux kernel source

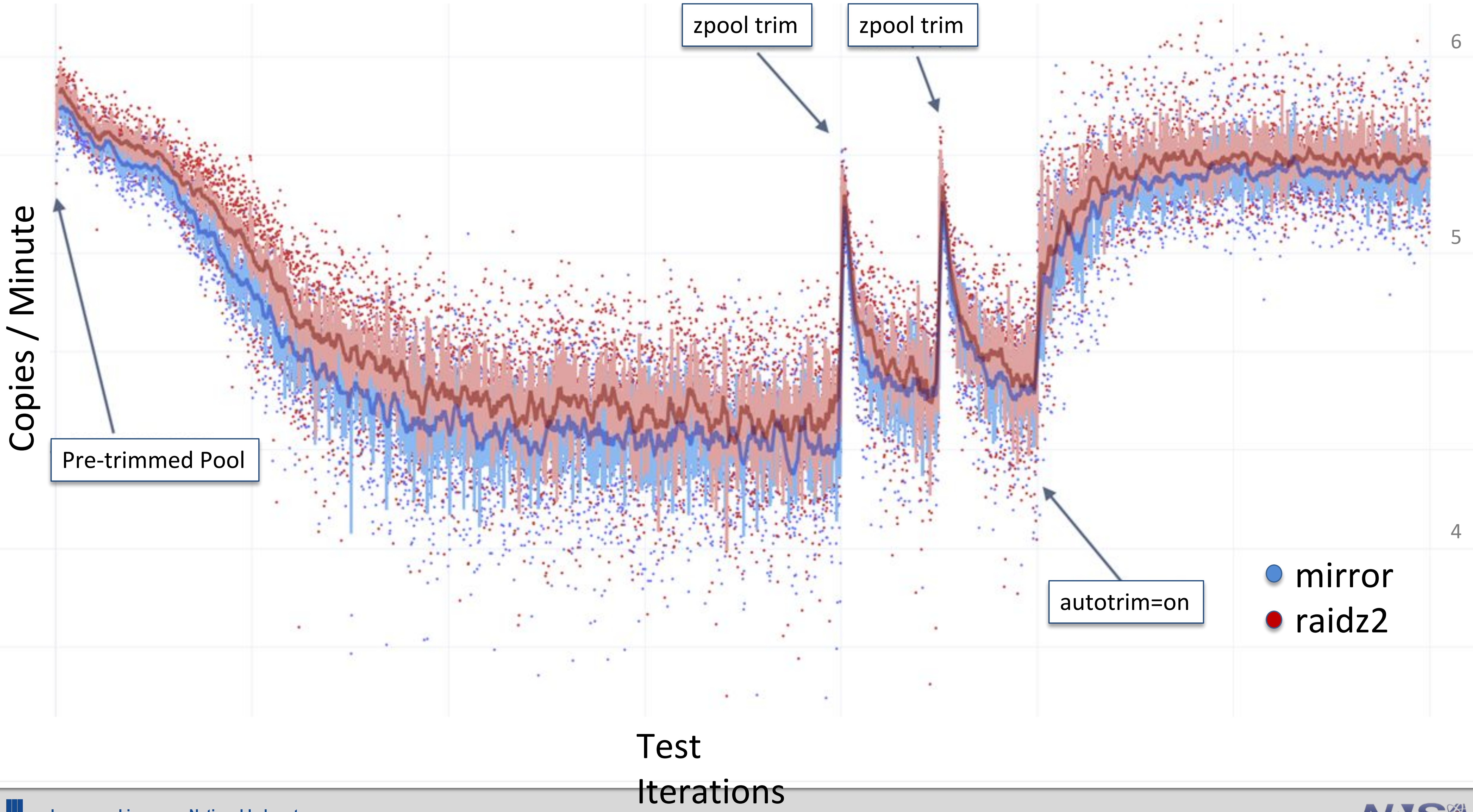
```
Repeat:
    N=$((RANDOM % 200))
    rm -r /testpool/fs/linux-$N
    time (cp -a /tmp/linux /testpool/fs/linux-$N; sync)
```



- Configuration
  - Target pool capacity ~80%
  - RHEL 7.6 - 3.10.0-957.0.0 kernel
  - Mirror (3-devices)
  - RAIDZ2 (1d+2p)



- Hardware
  - 3 Seagate ST800FM0173 devices
  - Dual 12Gb/s SAS attached



zpool trim

zpool trim

Pre-trimmed Pool

autotrim=on

Brian Behlendorf
behlendorf@llnl.gov

- Thank You:
  - Tim Chase
  - George Wilson
  - Matt Ahrens

Lawrence Livermore National Laboratory

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.