Using Allocation Classes

ZFS* User Conference

Don Brady, Intel Corp

Legal Information

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://www.intel.com/content/www/us/en/software/intel-solutions-for-lustre-software.html.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

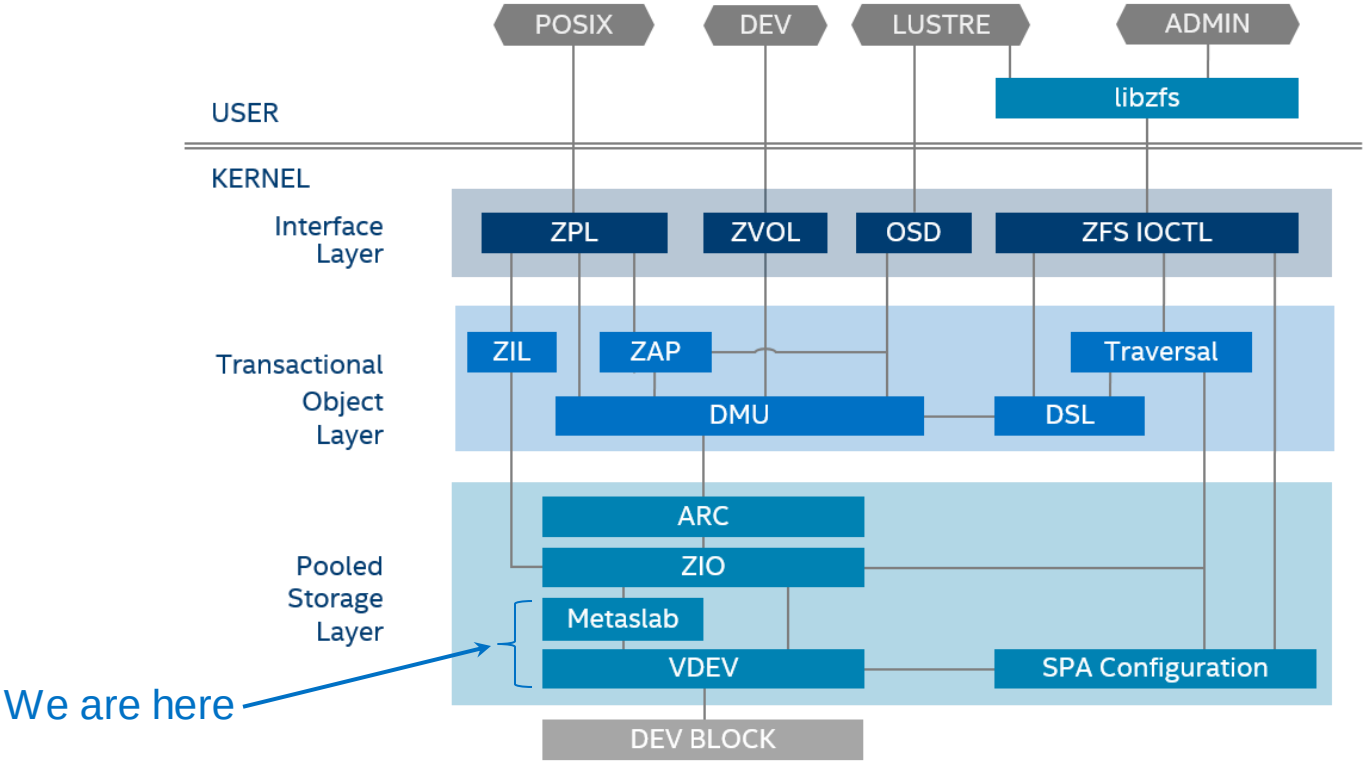
Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

Using Allocation Classes

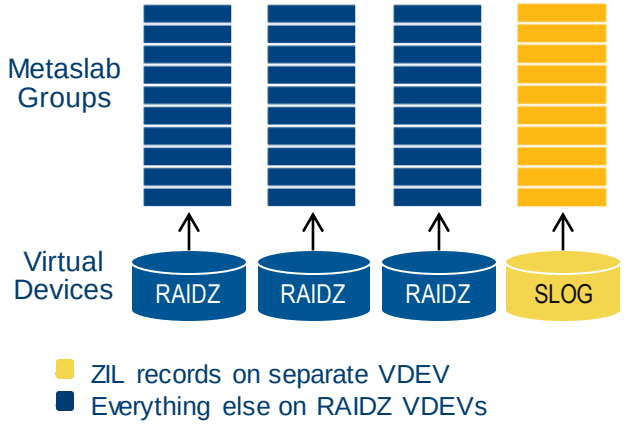
- SLOG Background
- SLOG Extensions
- Allocation Classes
- Management CLI

ZFS Block Allocation



Separate Log (SLOG)

- ZIL data uniqueness
  - Highly transient (don't worry about holes)
  - Want to involve all metaslabs
  - Want low latency (roving first-fit and wrap)
- SLOG allows different devices and allocation policies for ZIL data record
- Everything else uses primary VDEVs
- Can be stripped and/or mirrored

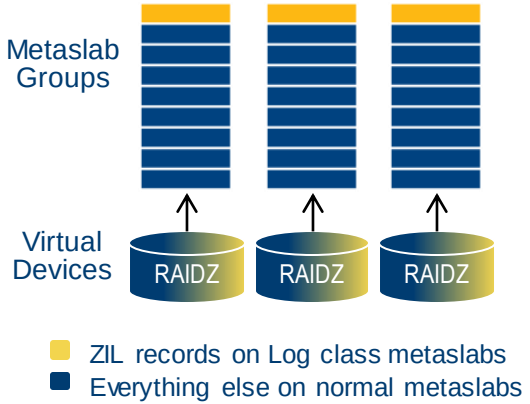


New SLOG Variation

- What if using a virtual LUN, or due to hardware constraints cannot have a dedicated SLOG?
- Still beneficial to isolate the transient ZIL data from all other allocations

Solution:

- Set aside one metaslab per top-level VDEV to service log data
- Introduces new concept called *VDEV Segregation*



Example: `zpool create -o segregate_log=on dozer raidz sda sdb sdc sdd sde`

Allocation Classes
(Variation #2)

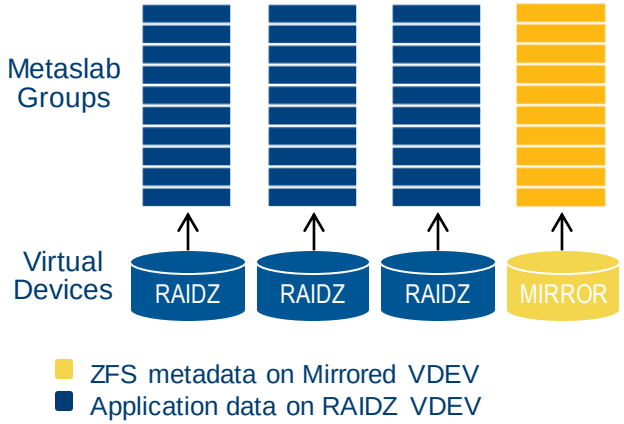
In addition to ZIL data, what other categories might we want to isolate?

- dedup, metadata, small blocks
- Two ways to opt in
  - *Dedicated* VDEVs (just like SLOG)
  - *Segregated* VDEVs
- feature@allocation_classes

Class	Designation
`Normal`	Any block type
`Log`	ZIL records
`Dedup`	DDT blocks
`Metadata`	Pool/Filesystem Metadata
`Small Blocks`	Small block sizes (32K)

Dedicated Class VDEV

- Enables different allocation policies for each allocation class
- All VDEV metaslabs dedicated to a specific allocation class
  - Application data uses RAIDZ VDEV
  - Metadata or small files use mirrored VDEV
- Opt-in during create with a class designation:
  - log
  - metadata
  - smallblks



Dedicated VDEV CLI

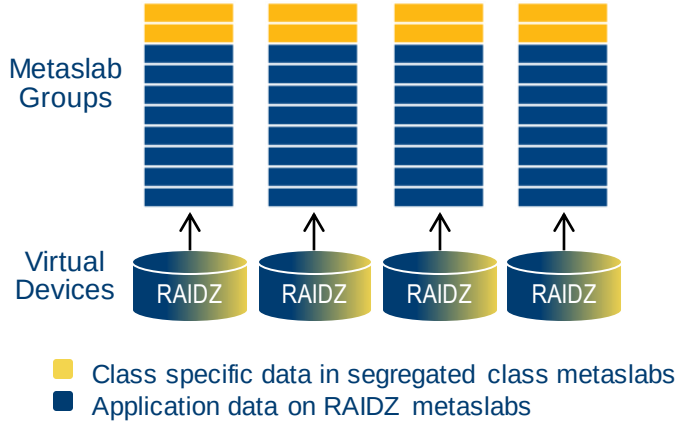
Create

```
zpool create <pool> <main-vdev>… <alloc-class> mirror <disks>…
```

```
# zpool create dozer \
  raidz1 sda sdb sdc sdd sde \
  raidz1 sdf sdg sdh sdi sdj \
  metadata mirror sdk sdl \
  log mirror sdm sdn
```

Segregated VDEV

- When separate devices for classes not feasible
- Set aside some metaslabs for specific class
- Application data is allocated on different metaslabs from the ZFS metadata
- Opt-in with a pool property:
  - segregate_metadata
  - segregate_smallblks
  - segregate_log
- dRAID -- metadata metaslabs are backed by mirror while normal class is backed by RAID



Segregated VDEV CLI

Create

```
zpool create <pool> -o segregate_[class]=on raidz2 <disks>…
```

```
# zpool create dozer \
  -o segregate_metadata=on \
  raidz1 sda sdb sdc sdd sde \
  raidz1 sdf sdg sdh sdi sdj
```

Management CLI

Listing VDEVs

```
$ zpool status demo
  pool: demo
 state: ONLINE
  scan: none requested
config:
        NAME                    STATE     READ WRITE CKSUM
        demo                    ONLINE       0     0     0
          raidz1-0              ONLINE       0     0     0
            sda                 ONLINE       0     0     0
            sdb                 ONLINE       0     0     0
            sdc                 ONLINE       0     0     0
            sdd                 ONLINE       0     0     0
            sde                 ONLINE       0     0     0
          metadata:mirror-1     ONLINE       0     0     0
            sdf                 ONLINE       0     0     0
            sdg                 ONLINE       0     0     0
```

Management CLI

Checking Segregate Property

```
$ zpool list -o name,segregate_log,segregate_metadata,segregate_smallblks
NAME        SEGREGATE_LOG  SEGREGATE_METADATA  SEGREGATE_SMALLBLKS
demo                  off                  on                   on
```

Management CLI

Listing Class Allocation Stats

```
$ zpool list -C demo
                     Generic Blocks          Small Blocks      Metadata Blocks
NAME          SPACE  ALLOC  USING  SPACE  ALLOC  USING  SPACE  ALLOC  USING
------------  -----  -----  -----  -----  -----  -----  -----  -----  -----
demo          3.62T  46.2G  1.24%   696G  22.7G  3.26%   232G  1.26G  0.54%
  raidz1-0    3.62T  46.2G  1.24%   696G  22.7G  3.26%   232G  1.26G  0.54%
```

Management CLI

Observing Metaslabs

```
# zdb -m demo
Metaslabs:
        vdev          0   segregate
        metaslabs   145   offset              spacemap          free          class
        ---------------   ------------------  ---------------   ------------  --------
        metaslab      0   offset            0   spacemap     75   free   32.0G  smallblk
        metaslab      1   offset    800000000   spacemap     74   free   32.0G  smallblk
        metaslab      2   offset   1000000000   spacemap     73   free   32.0G  smallblk
        metaslab      3   offset   1800000000   spacemap      4   free   20.5G  smallblk
        . . .
        metaslab     29   offset   e800000000   spacemap      5   free   16.0G    normal
        metaslab     30   offset   f000000000   spacemap      6   free   16.0G    normal
        metaslab     31   offset   f800000000   spacemap     10   free   17.8G    normal
        metaslab     32   offset  10000000000   spacemap      0   free     32G    normal
        metaslab     33   offset  10800000000   spacemap      0   free     32G    normal
        metaslab     34   offset  11000000000   spacemap      0   free     32G    normal
        . . .
        metaslab    144   offset  48000000000   spacemap      0   free     32G    normal
```

Management CLI

Observing Metaslabs

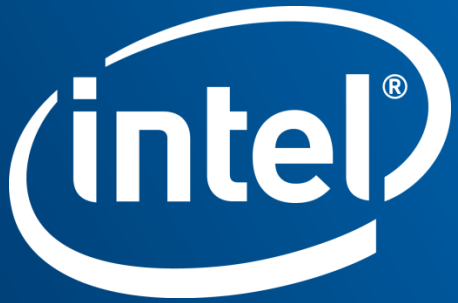
```
# zdb -mm [-P] demo

        Allocation Summary:       33180160 allocated
                  metadata:    2633216
                smallblks:   30546944
                    dedup:          0
                  generic:          0
```

Progress Update

- Work-in-progress pull request posted for community feedback

  https://github.com/zfsonlinux/zfs/pull/5182

- Intel CORAL project: using allocation classes with dRAID


intel®