Allocation Performance

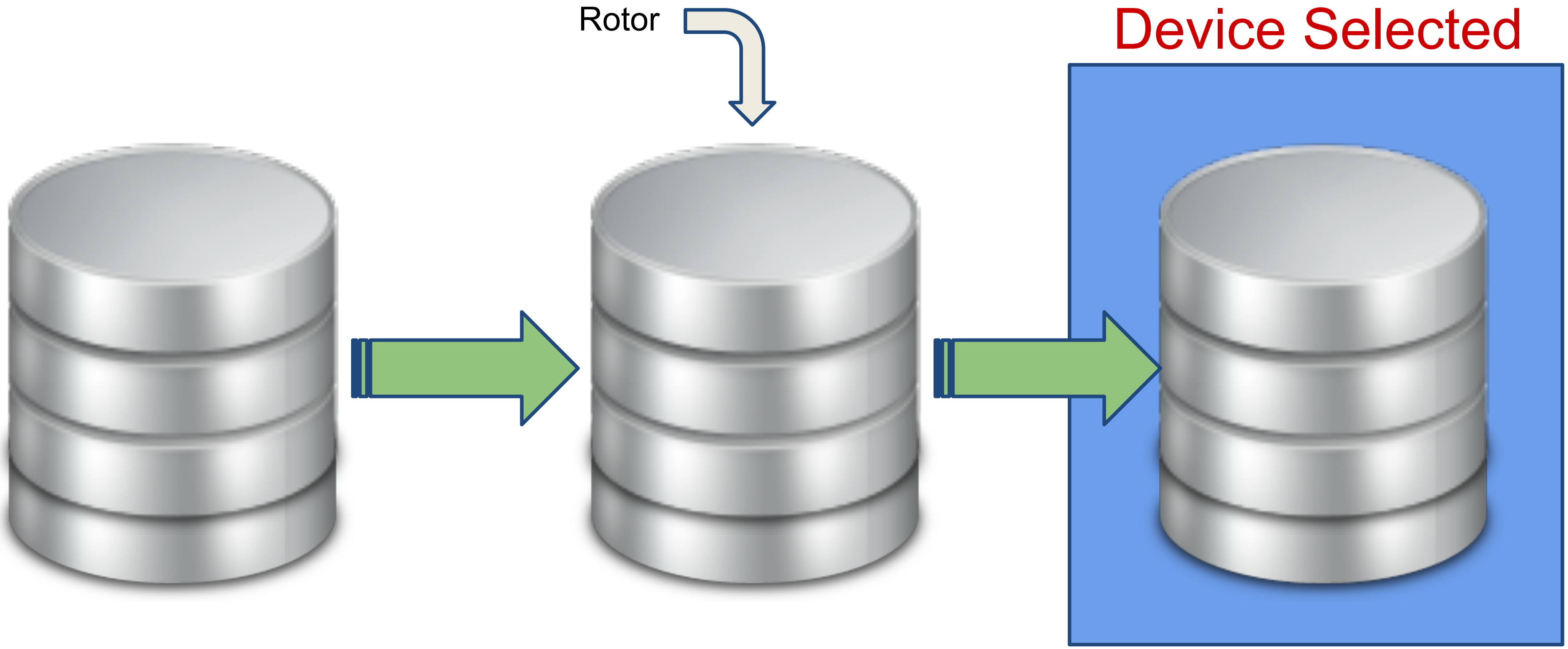
George Wilson
gwilson@delphix.com
@zfsdude

Overview

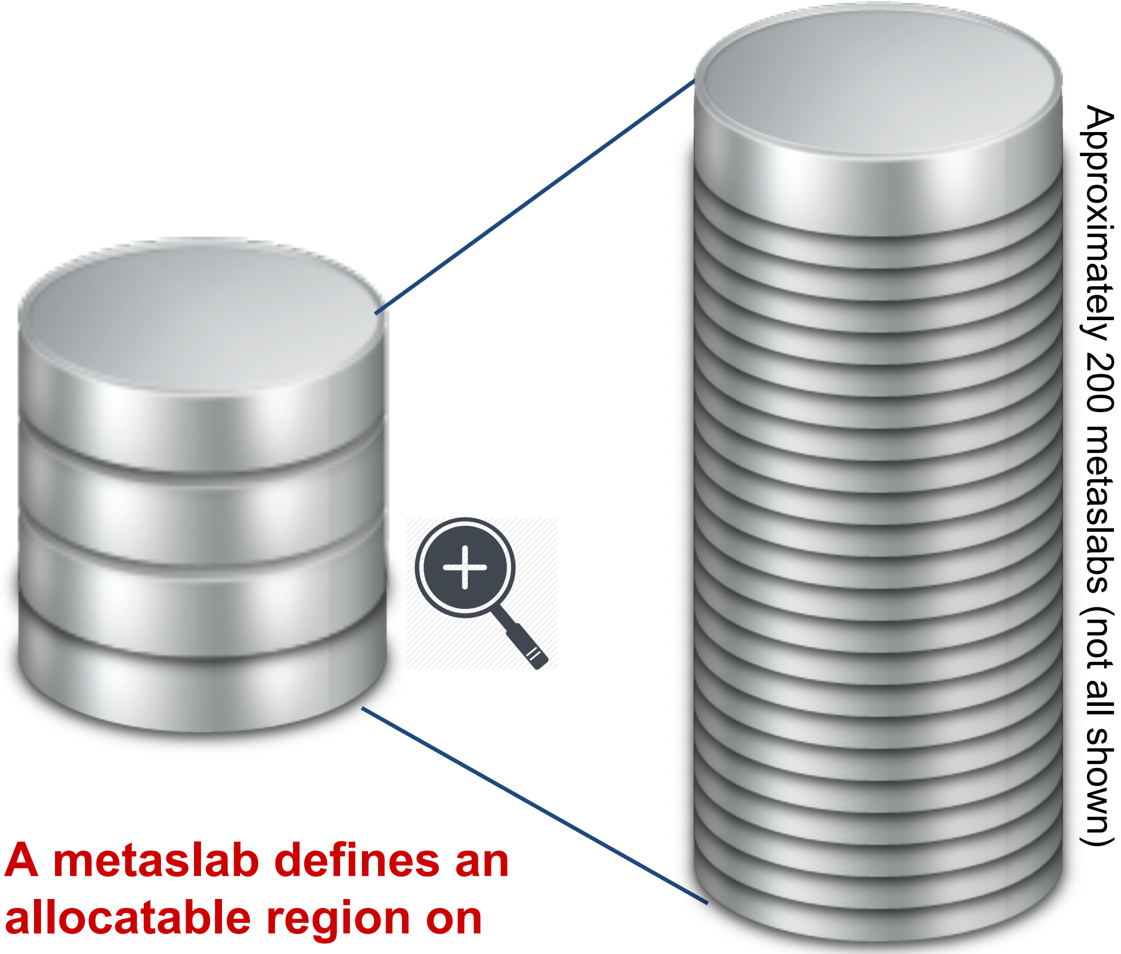
- Allocation Overview
- Recent Performance Improvements
- Upcoming Perf Improvements

Allocation Overview -- Device selection

Devices are visited in round-robin fashion



Allocation Overview -- Metaslab selection



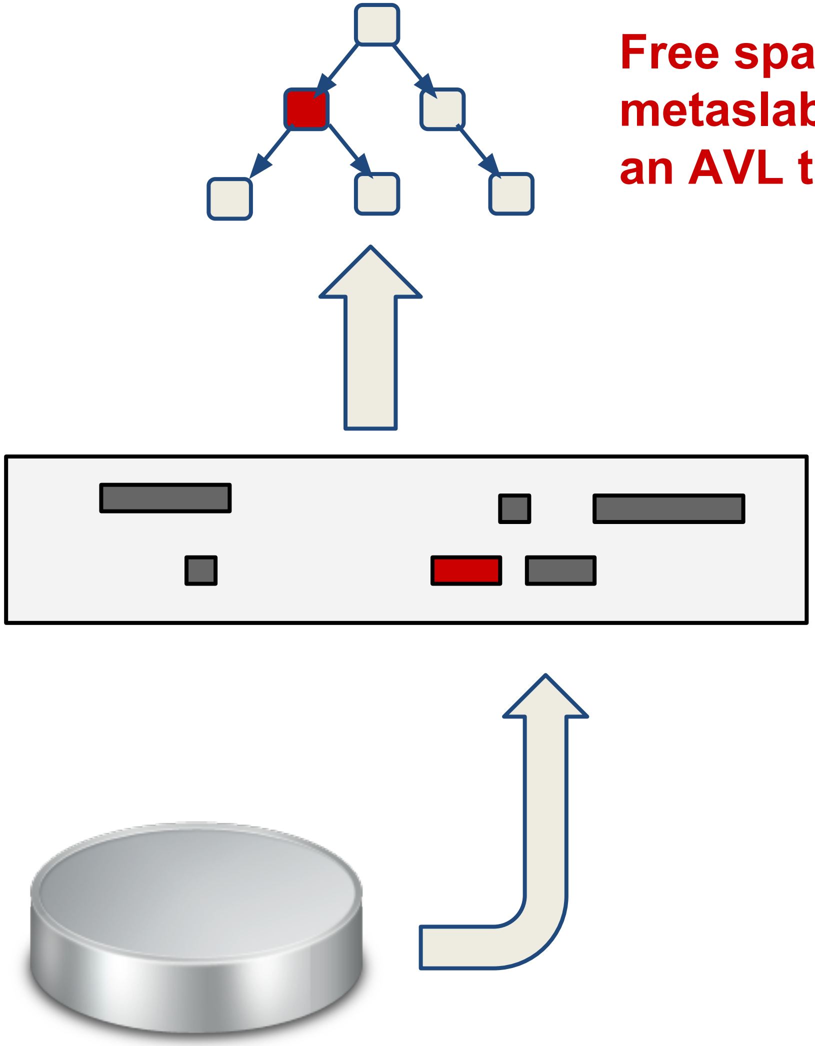
A metaslab defines an allocatable region on a disk

- Metaslabs are given a weight
- Sorted by weight
- Select metaslab with highest weight
- Attempt to allocate from region

What is a metaslab weight?

- Based on the free space of a metaslab
- Free space is “weighted” by the following factors
  - Weight the space down by fragmentation, if pool supports the space map histogram feature
  - Weight the space up by offset
  - Weight it up if the metaslab is currently loaded (i.e. its been recently used)
- The higher the weight the “better” the metaslab

Allocation Overview -- Block selection



Free space within a metaslab is stored in an AVL tree

- Select a free block from the next highest offset that has space (first fit)
- When space is low then pick a block that best fits the the size of the request (best fit)

What are we trying to improve

- Write performance of aged pools
  - Pools fragmentation increases over time
  - Performance suffers as pool nears full capacity
- Frag benchmark
  - Fills the pool to a specified capacity
  - Writes random data to random offsets
  - After benchmark reaches steady state, obtains the average random write IOPS
- Focused Investigation
  - Pool capacities <= 80%
  - Don’t kill performance above 80%

Looking back... 2013 improvements

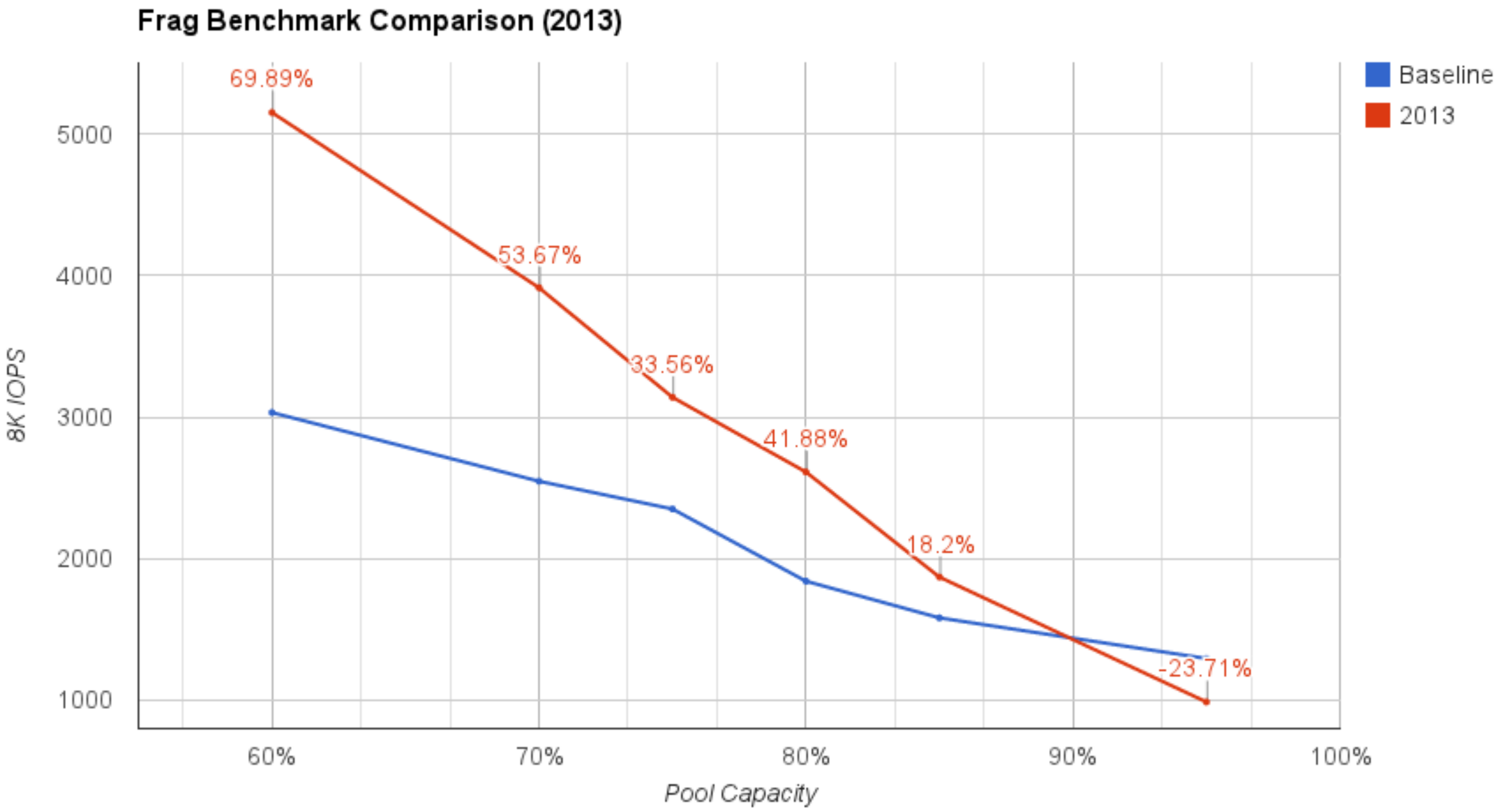
- Device selection
  - zfs_mg_noalloc_threshold
- Metaslab selection (region on that device)
  - improved metaslab preloading
  - space_map histogram
  - fragmentation metric
- Block selection
  - cursor fit allocator

Defining fragmentation

- Segment-based metric
  - 16M or larger segment is 0% fragmented
  - 1K or smaller segment is 100% fragmented
  - 50% fragmentation means majority of free space is comprised of 128K segments
  - Metric is in-core only and may change in the future

```
metaslab      2   offset     40000000   spacemap     52   free      163M
On-disk histogram:              fragmentation 80%
           100%  10   (1K):        2 *
            98%  11   (2K):       11 *
            95%  12   (4K):      749 ***
            90%  13   (8K):    11417 ****************************************
            80%  14  (16K):     1654 ******
            70%  15  (32K):      210 *
            60%  16  (64K):       59 *
            50%  17(128K):        41 *
            40%  18(256K):        24 *
            30%  19(512K):         5 *
            20%  20  (1MB):        1 *
            15%  21  (2MB):        1 *
```

Looking back…



Where are we going?

- Device selection
  - allocation throttle
- Metaslab selection (region on that device)
  - dynamic metaslab selection
- Block selection
  - hole-filling

Dynamic Metaslab Selection

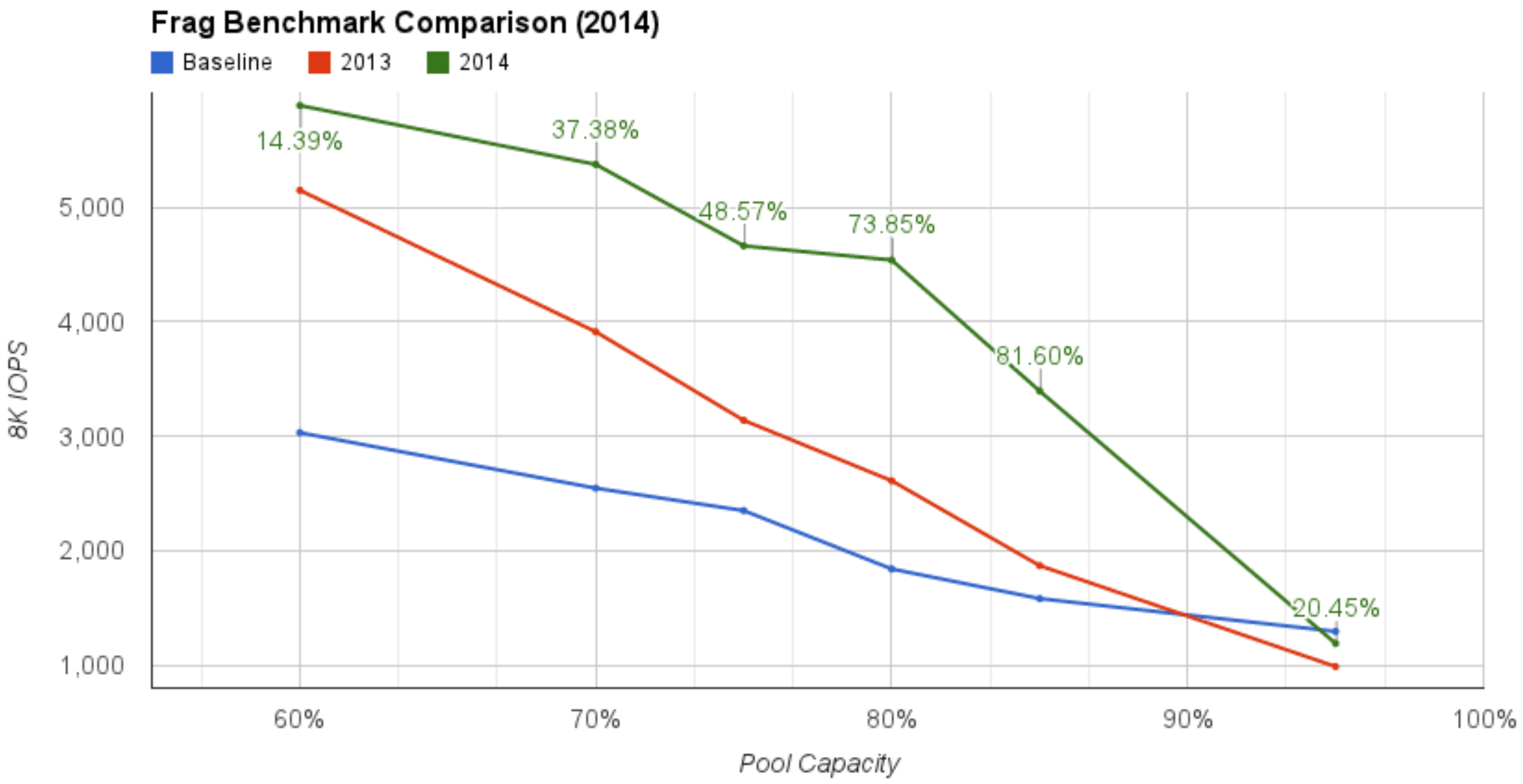
- Change the weight from space to segments
  - Requires space map histogram feature
  - Encodes the largest contiguous region into the weight
  - Metaslabs with larger regions are considered “best”
- Space-based weighting:

```
64      56      48      40      32      24      16      8       0
+-------+-------+-------+-------+-------+-------+-------+-------+
|PS1|                   weighted-free space                     |
+-------+-------+-------+-------+-------+-------+-------+-------+
```

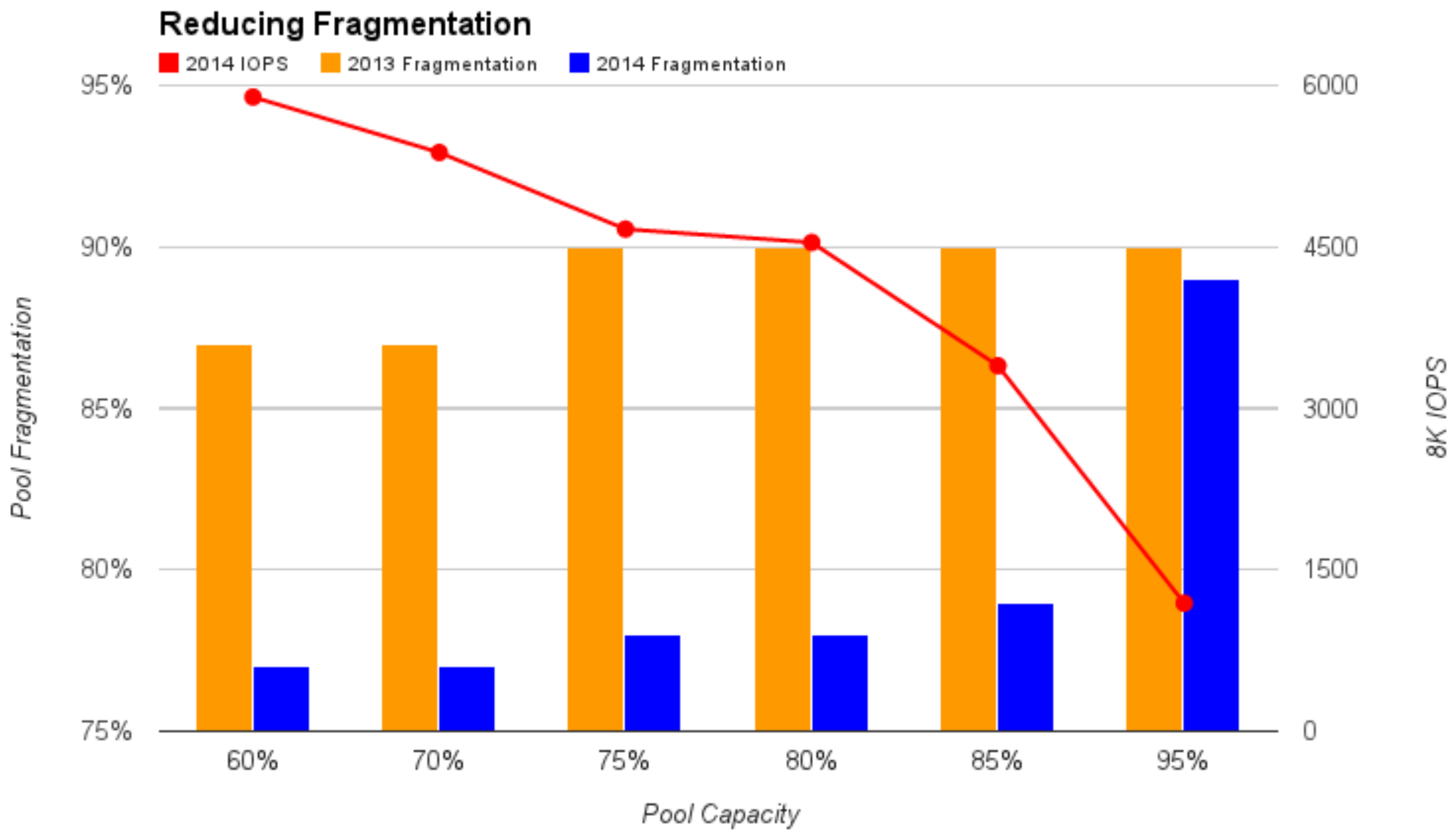
- Segment-based weighting:

```
64      56      48      40      32      24      16      8       0
+-------+-------+-------+-------+-------+-------+-------+-------+
|PS0| idx|             count of segments in region              |
+-------+-------+-------+-------+-------+-------+-------+-------+
```

Frag Performance Results

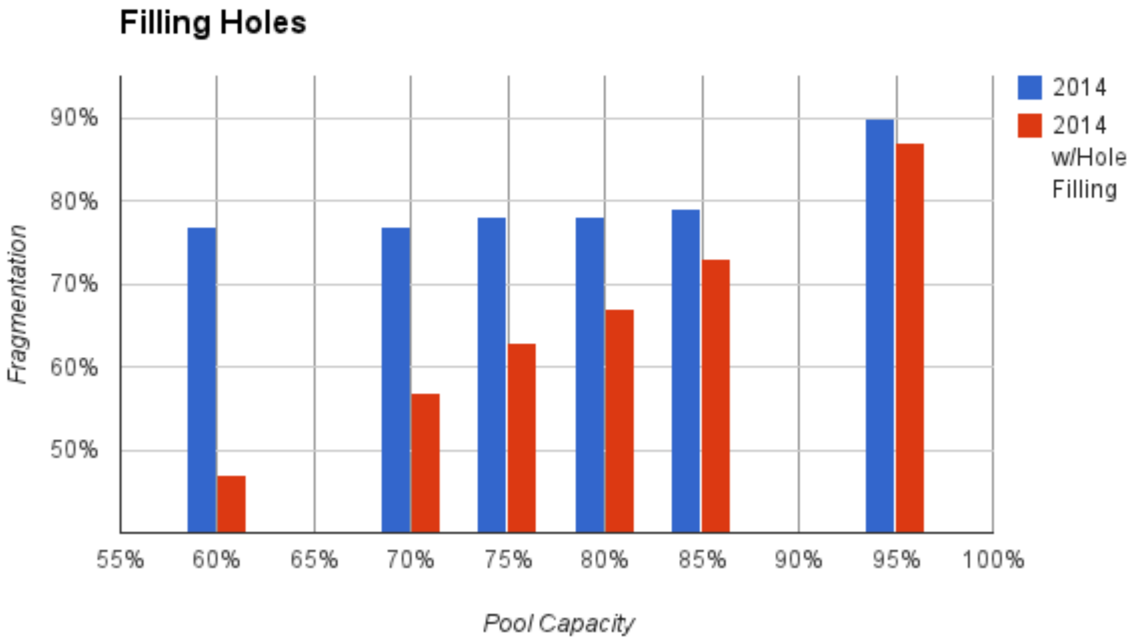


Reducing Fragmentation



Going further

- Hole-filling
  - Metaslabs are sorted by holes
  - Allocate from crappy metaslabs during times of low write activity
  - Preserve pristine metaslabs for heavy write loads



What’s next

- Allocation throttle improvements
- Directed device selection
- Synchronous write improvements

Questions?

Thank You

George Wilson
gwilson@delphix.com
@zfsdude

Backup Slides

Recent changes

- zfs_mg_noalloc_threshold
  - percentage of free space that makes a device eligible for allocations
  - any device that does not have this percentage free is skipped
- Improved metaslab preloading
  - Load more metaslabs before we reach allocation path (avoid reading during writes)

Recent changes

- space_map histogram
  - Maintain on-disk histogram of free segments in power-of-2 buckets
    - Requires pool to be upgrade (new feature flag)
    - space maps have to be upgraded to maintain information (happens when space maps condense)
  - Ability to retrieve histogram of free segments
    - zdb -mm - provide on-disk histogram (requires feature flag)
    - zdb -mmm - add in-core histogram (requires all space maps to be loaded)
    - Running ‘zdb’ fails when pool is busy or mostly full