OpenZFS

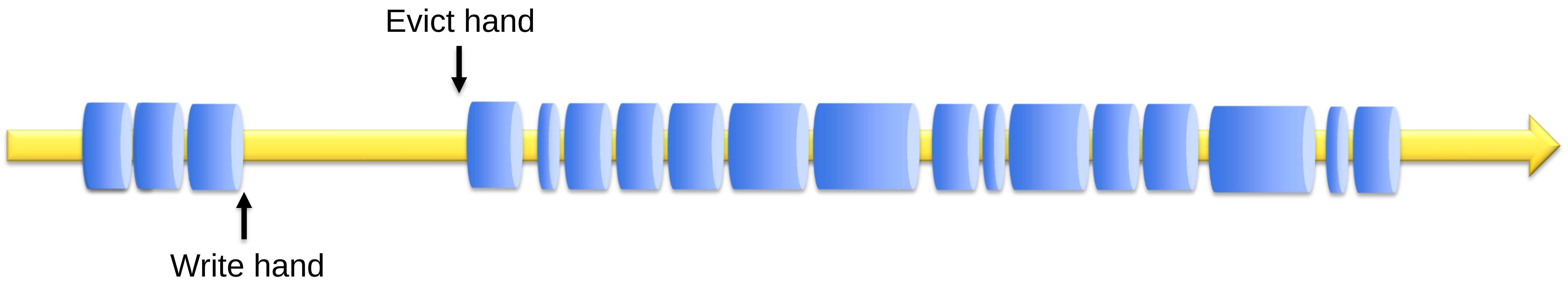
Persistent L2ARC

●●●

George Amanakis
gamanakis@gmail.com

L2ARC

- L2ARC caches buffers from ARC
- Rotary implementation
  1. Writes sequentially on device
  2. Evicts previously written buffers
  3. Writes new ones
  4. Repeat
- Loops from the beginning if cache device is full



How can we make the L2ARC persistent?

OpenZFS

- Enabling persistence means we need to restore the header entries of L2ARC buffers in ARC
- **L2ARC log blocks:** L2ARC metadata containing buffer header entries
  - Written on disk every 1022 L2ARC buffers

l2arc_log_blk_phys_t	
uint64_t	lb_magic
l2arc_log_blkptr_t	lb_prev_lbp
uint64_t	lb_pad[7]
l2arc_log_ent_phys_t	lb_entries[1022]

l2arc_log_ent_phys_t	
dva_t	le_dva
uint64_t	le_birth
uint64_t	le_prop
uint64_t	le_daddr
uint64_t	le_complevel
uint64_t	le_pad[2]

uint64_t le_prop
logical size
physical size
compression algo
data content type
protected status
prefetch status

Storing buffer header entries: L2ARC log blocks

l2arc_log_ent_phys_t

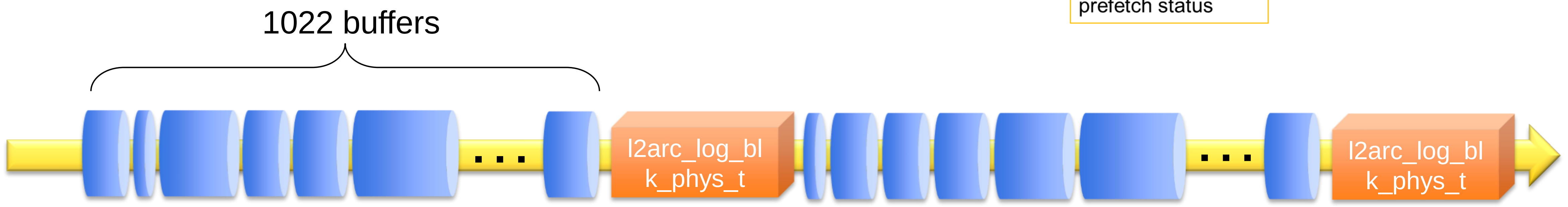
dva_t	le_dva
uint64_t	le_birth
uint64_t	le_prop
uint64_t	le_daddr
uint64_t	le_complevel
uint64_t	le_pad[2]

uint64_t le_prop

logical size
physical size
compression algo
data content type
protected status
prefetch status

l2arc_log_blk_phys_t

uint64_t	lb_magic
l2arc_log_blkptr_t	lb_prev_lbp
uint64_t	lb_pad[7]
l2arc_log_ent_phys_t	lb_entries[1022]

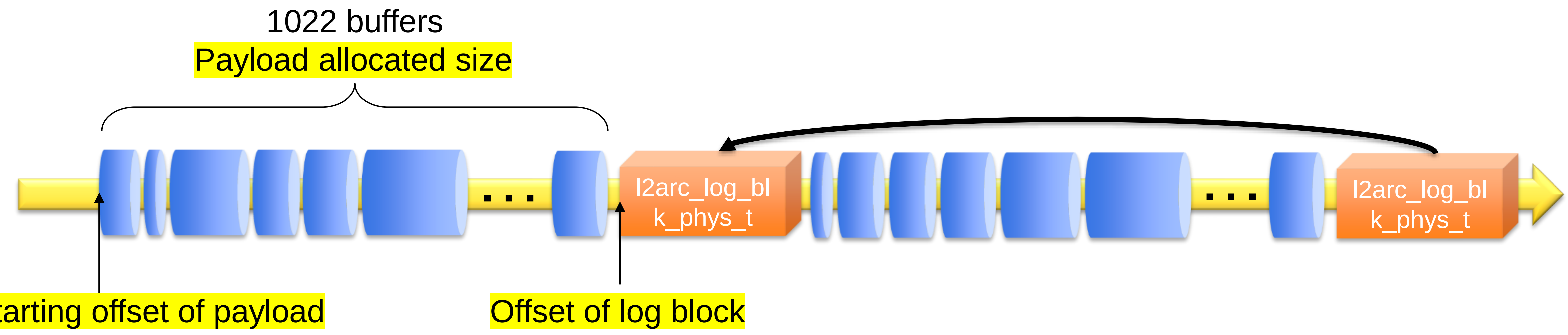


Keeping track of L2ARC log blocks: L2ARC log block pointers

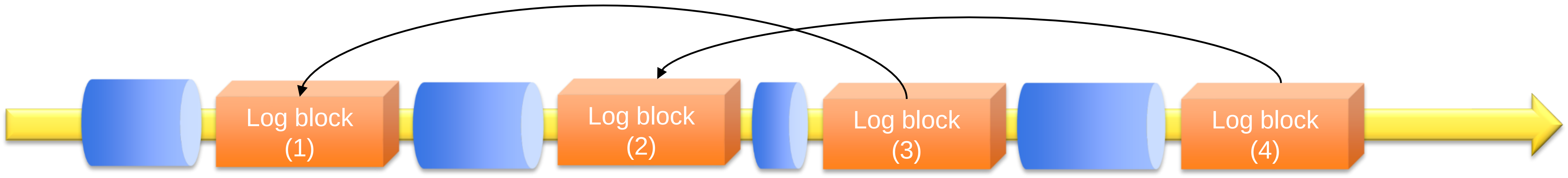
l2arc_log_blk_phys_t	
uint64_t	lb_magic
l2arc_log_blkptr_t	lb_prev_lbp
uint64_t	lb_pad[7]
l2arc_log_ent_phys_t	lb_entries[1022]

l2arc_log_blkptr_t	
uint64_t	lbp_daddr
uint64_t	lbp_payload_asize
uint64_t	lbp_payload_start
uint64_t	lbp_prop
zio_cksum_t	lbp_cksum

uint64_t lbp_prop
logical size
physical size
compression algo
cksum algorithm



Performance optimization: two interleaved chains of log blocks



Rebuild process

- Issue a sync read to read log block (4)
- Issue an async read to read the log block (3)
- Decompress and restore log block (4)
- Check if log block (3) has been read…

Performance

- consumer grade SATA SSD
- Intel Xeon E5-2667v2
- 64GB L2ARC device, ~100GB logical size
    - 2.8 sec with 1 chain
    - 1.9 sec with 2 interleaved chains (~32% faster)

- L2ARC rebuild is done asynchronously with respect to pool import
- No buffers are written to the cache device until the rebuild has been completed

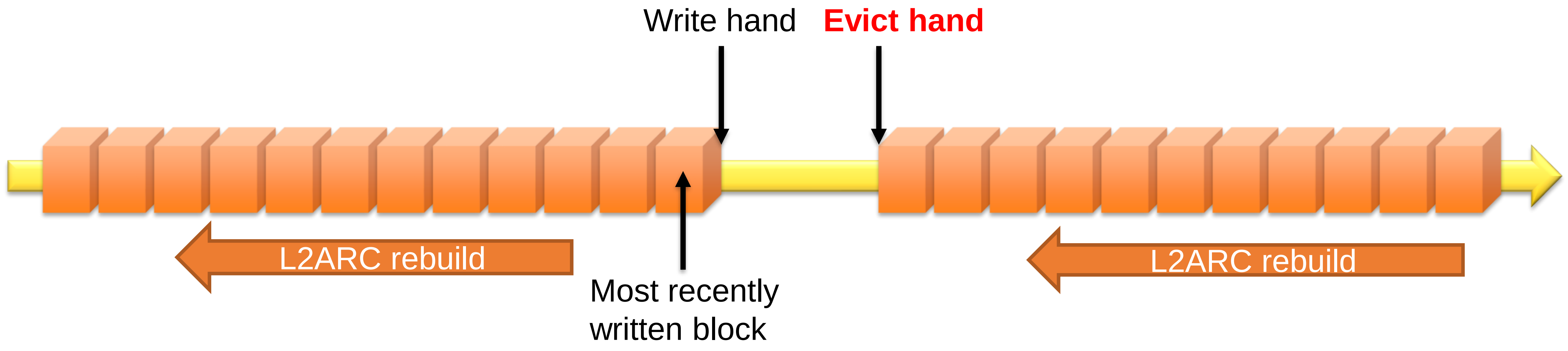
How does the L2ARC rebuild start?

- **L2ARC device header**
  - Is updated each time a log block is written to the cache device
  - Contains pointers to the two **most recently written** log blocks

l2arc_dev_hdr_phys_t	
uint64_t	dh_magic
uint64_t	dh_version
uint64_t	dh_spa_guid
uint64_t	dh_vdev_guid
uint64_t	dh_log_entries
uint64_t	dh_evict
uint64_t	dh_flags (first pass)
uint64_t	dh_lb_asize
uint64_t	dh_lb_count
l2arc_log_blkptr_t	dh_start_lbps[2]
…	
zio_eck_t	dh_tail

When does the L2ARC rebuild stop?

- Stop if the rebuild if we reach the **evict hand**
- The evict hand is stored in the device header
- The range between the write and the evict hand may have been zeroed-out if L2ARC TRIM is enabled



ZFS module parameters

- *l2arc_rebuild_enabled*
  - Controls whether ZFS will attempt to rebuild the L2ARC
  - Log blocks are still written on the device
  - Defaults to 1 (true)

- *l2arc_rebuild_blocks_min_l2size*
  - Disables the writing of L2ARC log blocks on the device
  - Beneficial for small devices
  - Defaults to 1GB

## zdb

# zdb -lll cachedevice

```
------------------------------------
L2ARC device header
------------------------------------
    magic: 6504978260106102853
    version: 1
    pool_guid: 2721483723720346099
    flags: 1
    start_lbps[0]: 310026240
    start_lbps[1]: 298739200
    log_blk_ent: 87
    start: 4194816
end: 1467482112
    evict: 4194816
    lb_asize_refcount: 50176
    lb_count_refcount: 28
    trim_action_time: 0
    trim_state: 0

log_blk_count:     28 with valid cksum
                   0 with invalid cksum
log_blk_asize:     50176
```

```
------------------------------------
L2ARC device log blocks
------------------------------------
lb[    1]    magic: 5498692020116080708
|            daddr: 310026240
|            payload_asize: 11285504
|            payload_start: 298740736
|            lsize: 65536
|            asize: 1536
|            compralgo: 15   <- LZ4
|            cksumalgo: 7    <- FLETCHER4
|

lb[    1]    le[    1]    DVA asize: 131072, vdev: 0, offset: 348140032
|                         birth: 18
|                         lsize: 131072
|                         psize: 131072
|                         compr: 2   <- Off
|                         complevel: 0
|                         type: 1    <- Buffer content: Data
|                         protected: 0
|                         prefetch: 0
|                         address: 298740736
|                         ARC state: 1
```

arcstats / zpool history

```
...
l2_size                           4     320079872
l2_asize                          4     305926656
l2_hdr_size                       4     230592
l2_log_blk_writes                 4     28
l2_log_blk_avg_asize              4     2215
l2_log_blk_asize                  4     50176
l2_log_blk_count                  4     28
l2_data_to_meta_ratio             4     3762
l2_rebuild_success                4     1
l2_rebuild_unsupported            4     0
l2_rebuild_io_errors              4     0
l2_rebuild_dh_errors              4     0
l2_rebuild_cksum_lb_errors        4     0
l2_rebuild_lowmem                 4     0
l2_rebuild_size                   4     318228480
l2_rebuild_asize                  4     305782784
l2_rebuild_bufs                   4     2436
l2_rebuild_bufs_precached         4     30
l2_rebuild_log_blks               4     28
...
```

# zpool history -i pool

```
2020-09-17.16:53:18 [txg:7775] L2ARC rebuild
successful, restored 28 blocks

2020-09-17.16:53:18 lt-zpool import tst3 -d
/home/user/vdevs
```

Questions?

Persistent L2ARC

- Status:
  - Done!
  - In master branch and upcoming OpenZFS 2.0!

- Acknowledgments:
  - Saso Kiselkov (Nexenta)
  - Yuxuan Shui
  - Jorgen Lundman

  Everybody who reviewed code!
  - Matt Ahrens
  - Brian Behlendorf
  - Ryan Moeller
  - Kjeld Schouten-Lebbing
  - George Wilson