ZFS: What's New

Jeff Bonwick
Oracle

New Stuff Since Last Year

- Major performance improvements
- User Quotas
- Pool Recovery
- Triple-parity RAID-Z
- Deduplication
- Encryption
- zfs diff
- zpool split
- zfs send/recv support for NDMP-based backup
- Read-only import

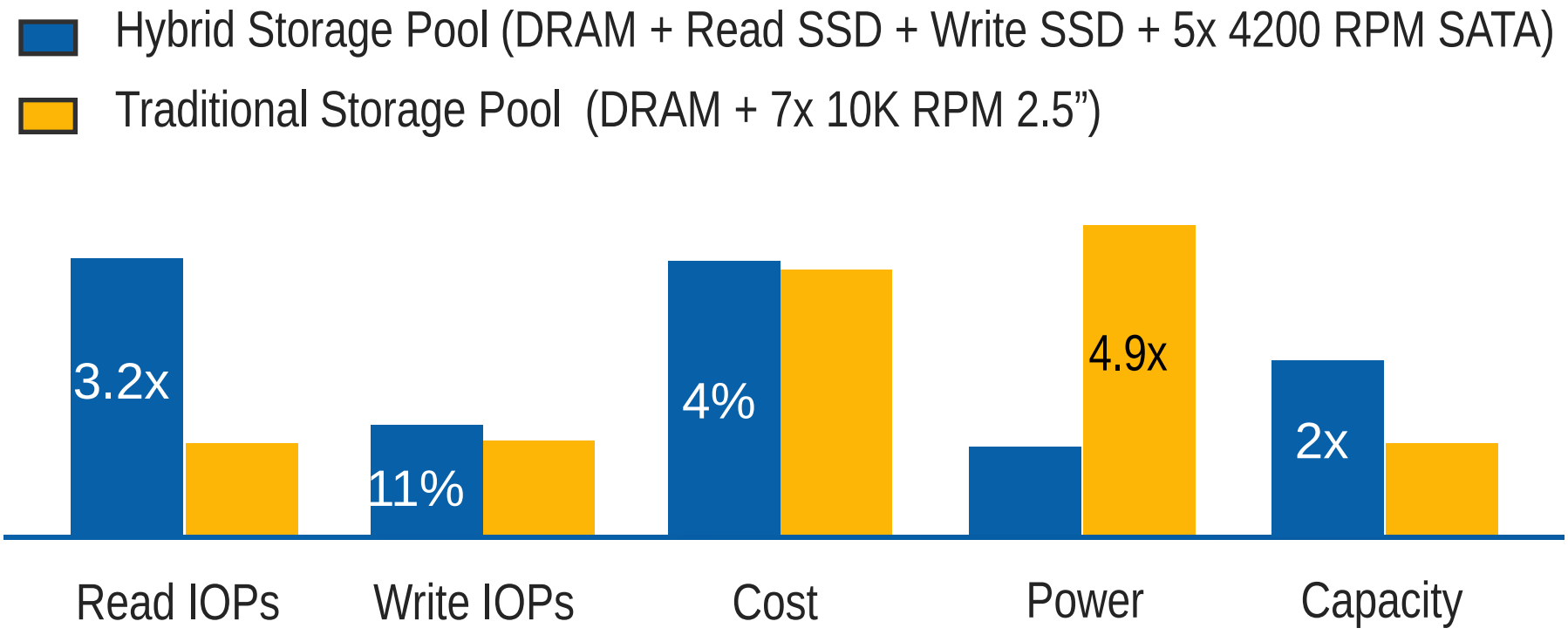
Performance

- Hybrid storage pools
- Better, faster block allocator
- Scrub prefetch
- Raw scrub/resilver
- Zero-copy I/O
- Duty-cycle scheduling class
- Native iSCSI
- Intent log latency/throughput control
- Explicit sync mode control
- RAID-Z / mirror hybrid allocation

ZFS Hybrid Storage Pools

- Separate log devices for fast synchronous writes
  - Enterprise-grade SLC flash SSD
    - Cheaper than NVRAM
    - Easily clustered over standard SAS fabric
- Cache devices for fast random reads
  - Cheap, consumer-grade MLC flash
    - DRAM eviction cache; support very large working sets
    - It's just a cache – failures are OK, no need to cluster
    - Everything is checksummed – no risk of silent errors
- Low-power, high-capacity disks for primary storage

ZFS Hybrid Pool Example

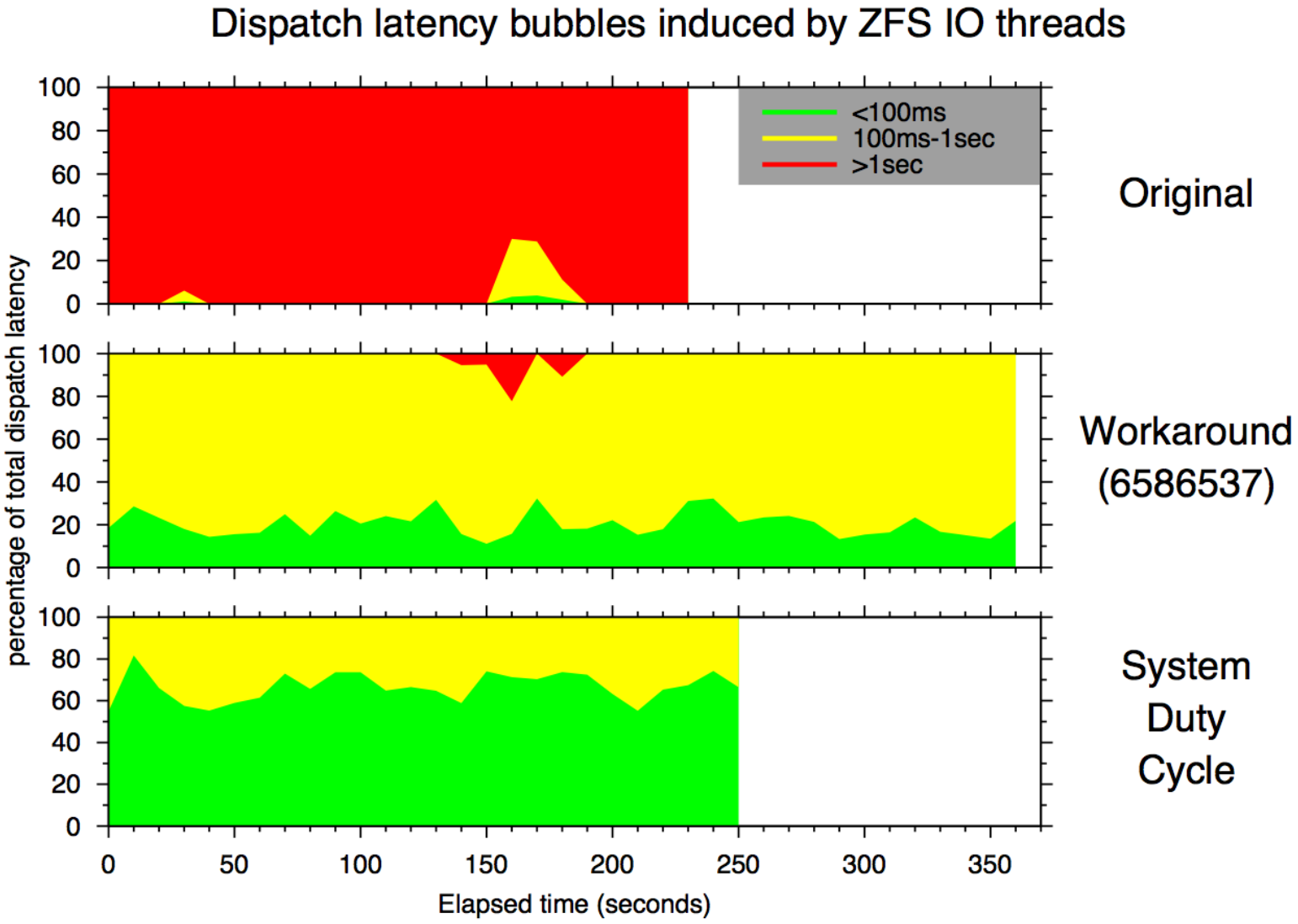


- If NVRAM were used, hybrid wins on cost, too
- For large configs (50T - 1PB+) cost is entirely amortized

Duty-Cycle Scheduling Class

- Problem
  - ZFS transaction group sync can hog the CPU
    - Several seconds when using compression, dedup, etc.
  - Kernel threads run at high priority
  - Result: latency bubbles in web, NFS, etc.
- Solution
  - Scheduling class for CPU-intensive kernel threads
  - Quantized, variable-priority to achieve duty cycle
  - Result: significant reduction in latency with minimal impact on throughput

Duty-Cycle Scheduling Class



ZFS logbias property

- Allows per-dataset log bias for latency vs. throughput
  - Slog devices are a limited resource
    - Synchronous bulk I/O crowds out latency-sensitive I/O
  - Not all sync I/O benefits from lower latency
    - When latency isn't critical, it's cheaper to go to disk
    - With many disks, available bandwidth is far higher
  - For Oracle redo logs, zfs set logbias=latency
  - For Oracle data files, zfs set logbias=throughput
  - Over 30% performance improvement

ZFS sync property

- Allows per-dataset control of synchronous semantics
  - Standard: follows all the usual POSIX rules
    - Calls to fsync(), O_DSYNC write(), etc. commit to stable storage before returning
  - Always: makes every transaction synchronous
    - Actually a performance win for some combinations of hardware and workload – explicit syncs not needed
  - Disabled: makes everything asynchronous
    - Huge performance win on systems without dedicated log devices when sync semantics aren't required

RAID-Z / Mirror Hybrid Allocator

- Use one group of disks as both RAID-Z and mirror
  - RAID-Z for user data and most metadata
    - Provides greatest capacity and highest write throughput
  - Mirror for latency-sensitive, dittoed metadata
    - Provides most read IOPs and lowest read latency
    - Ideal for small, randomly accessed bits of metadata
      - indirect blocks, dnodes, directories, dedup tables
- Some real-world workloads up to 4x faster
  - Copying large files with deduped blocks
  - rm -rf of a giant directory

RAID-Z / Mirror Hybrid Layout

- Most blocks are RAID-Z
- A few are mirrored (red)
- Any adjacent disks can mirror
- No restrictions on placement
- Bit in block pointer indicates which layout was selected

LBA \ Disk	A	B	C	D	E
0	P_0	D_0	D_2	D_4	D_6
1	P_1	D_1	D_3	D_5	D_7
2	P_0	D_0	D_1	D_2	P_0
3	D_0	D_1	D_2	M_0	M_1
4	P_0	D_0	D_4	D_8	D_{11}
5	P_1	D_1	D_5	D_9	D_{12}
6	P_2	D_2	D_6	D_{10}	D_{13}
7	P_3	D_3	D_7	P_0	D_0
8	D_1	D_2	D_3	X	P_0
9	D_0	M_0	M_1	P_0	D_0
10	D_3	D_6	D_9	P_1	D_1
11	D_4	D_7	D_{10}	P_2	D_2
12	D_5	D_8	•	•	•

User Quotas

- For enterprise customers
  - Finer grained answer to "where did my space go"
- For education customers
  - Many users, want quota per user
  - One fs / user is too many (unfortunately)
- User & group quotas with "deferred enforcement"
  - User may go over quota for several seconds (one transaction group) before system notices that they are over quota and returns EDQUOT
- Supports both SMB SIDs and POSIX UIDs/GIDs

User Quota Properties

- New ZFS dataset properties
  - userused@<user> groupused@<group>
  - userquota@<user> groupquota@<group>
  - "zfs get" / "zfs set" like other properties
  - <user> or <group> specified as:
    - Numeric POSIX ID (125829)
    - POSIX name (ahrens)
    - Numeric SID (S-1-123-456-789)
    - SID name (matthew.ahrens@oracle)

User Quota Subcommands

- "zfs userspace" and "zfs groupspace"
  - Display table, one line per user or group, e.g.:

```
TYPE        NAME           USED  QUOTA
POSIX User  ahrens          14M     1G
POSIX User  lling          258M   none
POSIX Group staff         3.75G    32T
SMB User    marks@oracle   103M     5G
```

Pool Recovery: The Problem

- ZFS pool integrity depends on explicit write ordering
  - Some cheap disks and bridges silently ignore it!
  - Result: uberblock written before data it points to
  - Power loss can lead to complete pool failure

Pool Recovery: The Solution

- Recover pool even if devices ignore write barriers
  - Verify integrity and completeness of recent transaction groups during pool open
    - Very fast – uses birth-time-pruned pool traversal
  - If damaged, rollback to previous uberblock
    - Lather, rinse, repeat
  - Rollback made reliable by deferred block reuse
    - Recently freed blocks are never immediately reused, so it's completely safe to assume they're still valid

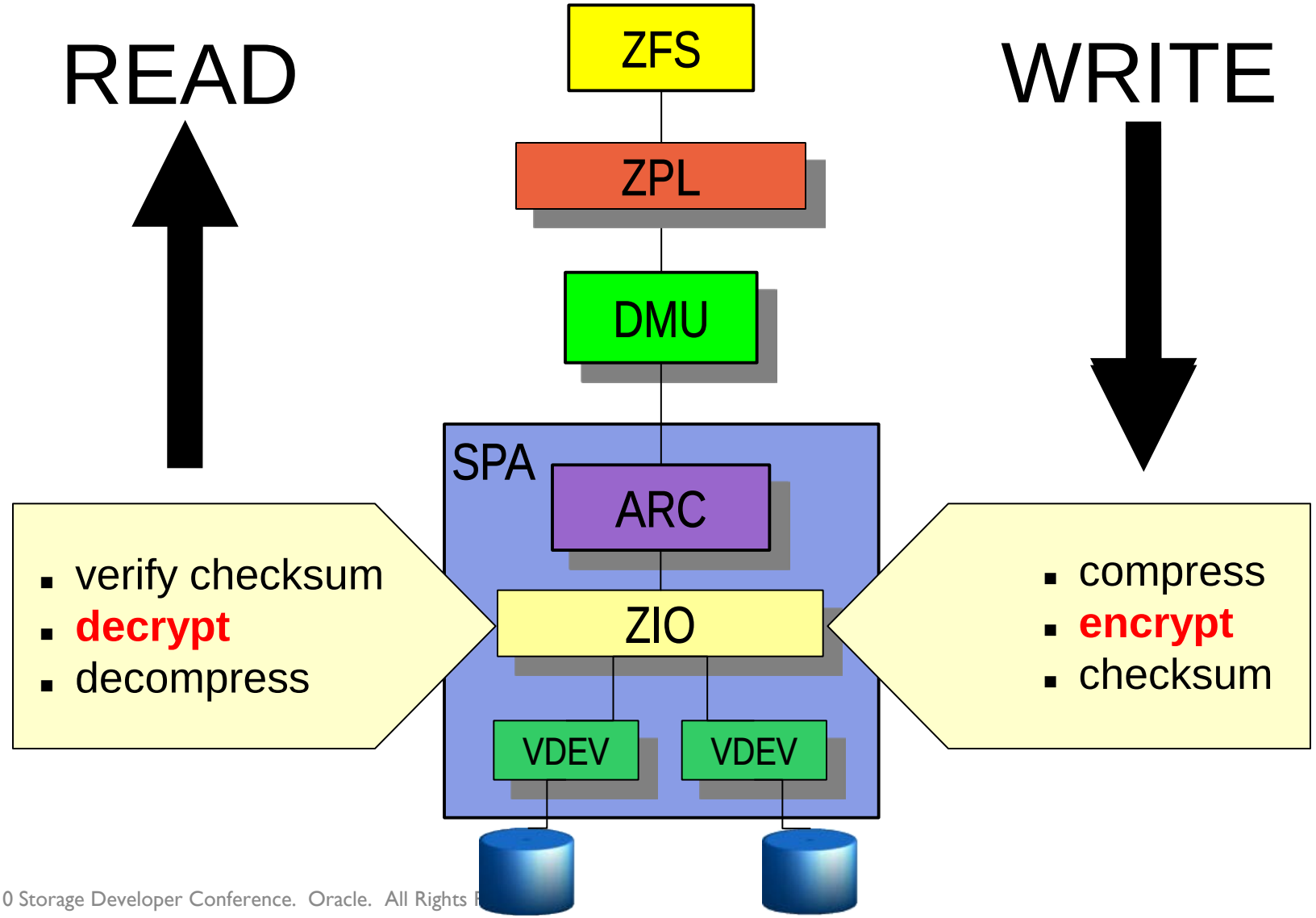
Triple-Parity RAID-Z (RAIDZ3)

- Survives three-disk failure
  - Or, two-disk failure plus occasional bad reads
- Enables bigger, faster, high-BER disks
  - 30-40% of the bits on modern hard disks are ECC
  - Modified HDD zone recording tables would allow:
    - 30-40% higher capacity
    - 30-40% higher bandwidth
    - Many more I/O errors, detected and corrected by ZFS

Dedup

- Only store one copy of identical data blocks
  - Three parts: on-disk, in-core, over-the-wire
- Key applications
  - Virtualization
  - Backup servers
  - Build environments
- Fully integrated with ZFS (i.e. not an add-on)
  - No special hardware
  - No limits on capacity
  - Major performance improvements in recent builds

ZFS Encryption



ZFS Encryption: Design Goals

- Encrypt all data and ZPL metadata (name, owner, etc)
  - All data on zvols can be encrypted
- Allow for secure delete
- Must not require special hardware
  - But should be able to take advantage of it
- Integrate with existing ZFS admin model
- Support mix of ciphertext and cleartext datasets
- Minimize performance overhead
  - Actual cost: 7% for random I/O, 3% for sequential

ZFS Encryption: Key Management

- Dataset encryption requires two different keys
  - A user specified key called the “wrapping” key
  - A randomly generated dataset key wrapped by the user specified key
- This model simplifies such tasks as secure deletion
  - Get rid of the wrapping key and the data is gone
- The wrapped key can also change without changing the user specified wrapping key

ZFS Encryption: Administration

- Dataset encryption must enabled at creation time
  - Specify keysource and encryption algorithm
  - Enables SHA-256 checksum automatically
- Keysource indicates location of the wrapping key
- Two encryption algorithms supported initially
  - AES-128-CCM
  - AES-256-CCM (default when enabled)

```
# zfs create -o encryption=on -o keysource=passphrase,prompt tank/fs
```

zfs diff

- Very efficiently generate the delta between snapshots
- Shows all files that have been added, removed, modified, or renamed

```
# zfs create tank/jeff
# cd /tank/jeff
# echo hello >hello.txt
# echo world >world.txt
# zfs snapshot tank/jeff@hello-world
# echo kitty >kitty.txt
# mv hello.txt goodbye.txt
# rm world.txt
# ls
goodbye.txt   kitty.txt
# zfs diff tank/jeff@hello-world
M   /tank/jeff/
R   /tank/jeff/hello.txt -> /tank/jeff/goodbye.txt
-   /tank/jeff/world.txt
+   /tank/jeff/kitty.txt
```

Other cool stuff

- zpool split
  - Splits a pool of mirrored devices into two distinct, identical pools
  - Useful for disaster recovery, site replication, or instant physical archive
- zfs send/recv support for NDMP-based backup
  - The speed of zfs send/recv
  - The tools you already know and love/hate
- Read-only import
  - Examine pool with guarantee of not altering it