SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

Array Level Steady State Detection for ZFS Storage Servers

A Case Study

Ryan McKenzie
iXsystems

Agenda

- Problem Statement
- Previous Work
- Proposed Solution
- Implementation
- Observations and Results

Problem Statement

- Performance Engineers want data to be:
  - Repeatable – won't vary on test order/time
  - Applicable – accurate reflection of how the system should perform under the given load
- Storage systems and their components/layers are complex.
- Test time is very limited compared to customer run time

Problem Statement

- Goals:
  - Steady State Detection (this talk)
    - To prevent measurement in a volatile state
  - Preconditioning (ongoing work)
    - To induce an appropriate steady state on the storage array under test as quickly as possible
    - Rudimentary approach is to just run target workload until steady state is detected

Previous Work

- Array level steady state detection can be viewed as multiple layers on top of single device SSD steady state detection…
  - SNIA has a widely accepted and applied solution for that problem

Solid State Storage (SSS) Performance Test Specification (PTS)

- From v2.01 (Feb. 2018)

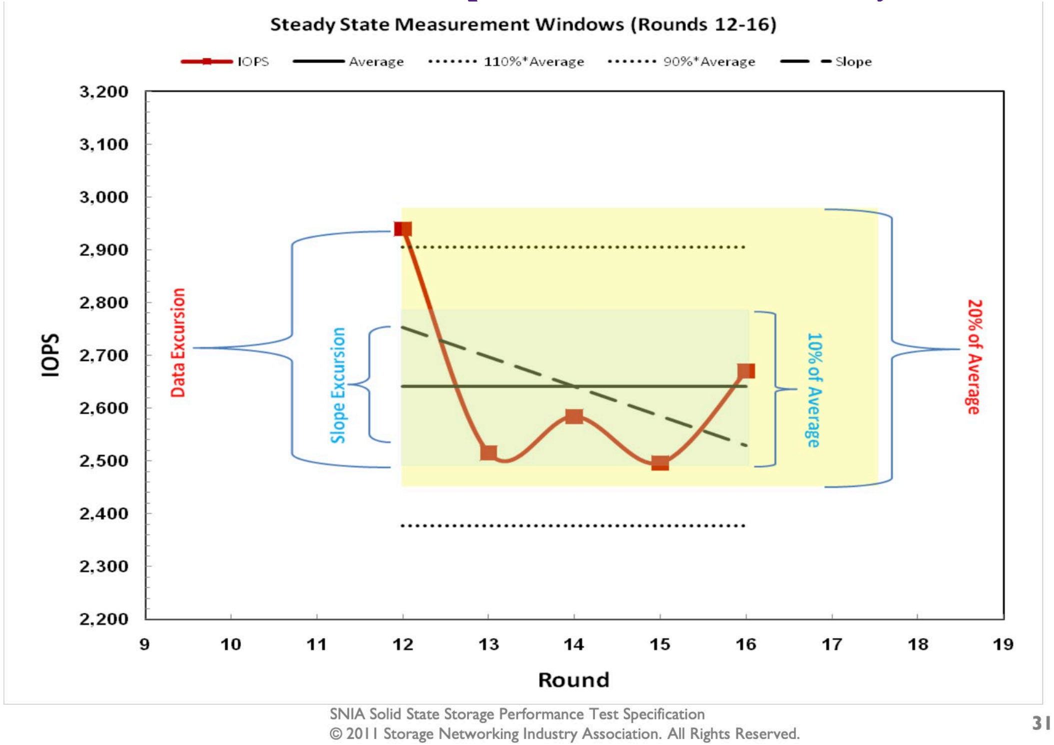
2.1.24 **Steady State:** A device is said to be in Steady State when, for the dependent variable (y) being tracked:

a) Range(y) is less than 20% of Ave(y): Max(y)-Min(y) within the Measurement Window is no more than 20% of the Ave(y) within the Measurement Window; and

b) Slope(y) is less than 10%: Max(y)-Min(y), where Max(y) and Min(y) are the maximum and minimum values on the best linear curve fit of the y-values within the Measurement Window, is within 10% of Ave(y) value within the Measurement Window.

https://www.snia.org/sites/default/files/technical_work/PTS/SSS_PTS_2.0.1.pdf

SNIA Solid State Storage Performance Test Specification (Easen Ho, 2011)



https://www.snia.org/sites/default/files/HoEasen_SNIA_Solid_State_Storage_Per_Test_1_0.pdf

Proposed Solution

- Select multiple metrics to gather from a ZFS storage server under load
- Subject each metric to the SNIA steady state determination of range and slope

Proposed Solution: Metrics

- ARC Metrics
  - ARC size
  - ARC MRU/MFU makeup
  - ARC data/metadata makeup
  - ARC hit ratio

Proposed Solution: Metrics

- L2ARC Metrics (if applicable)
  - L2ARC size
  - L2ARC write rate
  - L2ARC hit ratio

Proposed Solution: Metrics

- ZFS Metrics
  - Pool performance (OPS)
  - Pool performance (bandwidth)
  - Pool Performance (avg op latency)

Proposed Solution: Metrics

- All metrics must "do no harm" if collection fails
  - Currently failure state is "steady"

- All non-performance metrics are normalized to a percentage
  - i.e. ARC size is expressed as

```
blocks in ARC / max blocks in ARC
```

Implementation

Implementation: Metrics

- NOTE: In a test environment, we have the luxury of knowing:
  - Which pool(s) active dataset resides
  - Active dataset size
  - Caches were clear before this test

Implementation: Metrics

- ARC size
  - kstat.zfs.misc.arcstats.size / kstat.zfs.misc.arcstats.c_max
- ARC MRU/MFU makeup
  - arcMRU/(arcMRU+arcMFU)
  - arcMRU = kstat.zfs.misc.arcstats.mru_size
  - arcMFU = kstat.zfs.misc.arcstats.mfu_size

Implementation: Metrics

- ARC data/metadata makeup
  - arcDATA/(arcDATA+arcMETA)
  - arcDATA = kstat.zfs.misc.arcstats.data_size
  - arcMETA = kstat.zfs.misc.arcstats.metadata_size
- ARC hit ratio
  - arcHITS/(arcHITS+arcMISS)
  - arcHITS = kstat.zfs.misc.arcstats.hits
  - arcMISS = kstat.zfs.misc.arcstats.misses

Implementation: Metrics

- L2ARC size
  - kstat.zfs.misc.arcstats.l2_asize / L2CAP
  - L2CAP = sum(cache device capacities from “diskinfo”)
- L2ARC write rate
  - L2writes / vfs.zfs.l2arc_write_max
  - L2writes = kstat.zfs.misc.arcstats.l2_write_bytes this interval – last interval
  - NOTE: this value can be > 100% due to boost!

Proposed Solution: Metrics

- L2ARC hit ratio
  - l2arcHITS/(l2arcHITS+l2arcMISS)
  - l2arcHITS = kstat.zfs.misc.arcstats.l2_hits
  - l2arcMISS = kstat.zfs.misc.arcstats.l2_misses

Proposed Solution: Metrics

- ZFS Metrics
  - Get two intervals of zpool iostat:
    - zpool iostat -l <pool> <interval> 2
  - Pool performance (OPS)
    - “operations” (read + write), normalized to kOPS
  - Pool performance (bandwidth)
    - “bandwidth” (read + write), normalized to MiB/s

Proposed Solution: Metrics

- ZFS Metrics
  - Get two intervals of zpool iostat:
    - zpool iostat -l <pool> <interval> 2
  - Pool Performance (avg op latency)
    - “total_weight” (read + write), normalized to ms

Proposed Solution: Metrics

```
# zpool iostat -l tank 2 2
              capacity     operations     bandwidth    total_wait     disk_wait    syncq_wait    asyncq_wait  scrub   trim
pool        alloc   free   read  write   read  write   read  write   read  write   read  write   read  write   wait   wait
----------  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----  -----
tank        4.80T   122T     15    904   120K  39.5M    3ms    1ms    3ms  750us  177us  890ns  651us  656us      -      -
tank        4.80T   122T  43.4K      0   231M      0    1ms      -    1ms      -   53us      -  283us      -      -      -
```

First line of zpool iostat is cumulative since boot!

Implementation: Engine

- A python Thread class instance
  - Upon creation thread will preload constants like L2ARC capacity, ARC max, etc…
  - When started, thread will spend a tunable amount of time loading metrics
  - When steady state is declared, can signal benchmark to begin measurement
  - Can be stopped early using Thread event

Implementation: Engine

- Collect all metrics every INTERVAL seconds
  - Default 15s, tunable
- Once WINDOW of intervals is collected, begin calculating range and slope for steady state
  - Default 20 intervals, tunable
  - Recommend window at least as long as benchmark measurement period

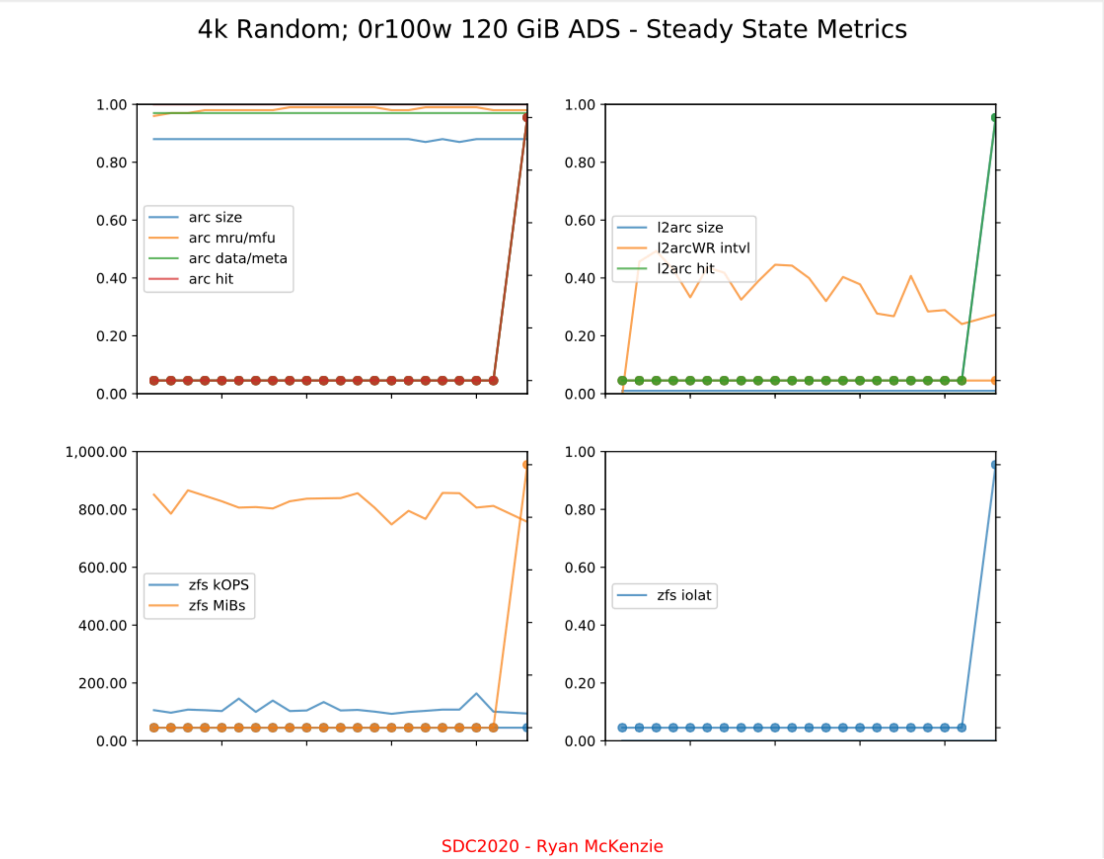
Results

Solution Under Test

- FreeBSD+OpenZFS based storage server
  - 512G main memory
  - 4x 1.6TB NVMe drives as L2ARC
    - Enterprise-grade dual port PCIeG3
  - 142 2TB SAS HDDs in 71 mirror vdevs
  - 1x 100GbE NIC (optical QSFP+)
  - Tested with various iSCSI, NFSv3, and SMB workloads (will present NFSv3)

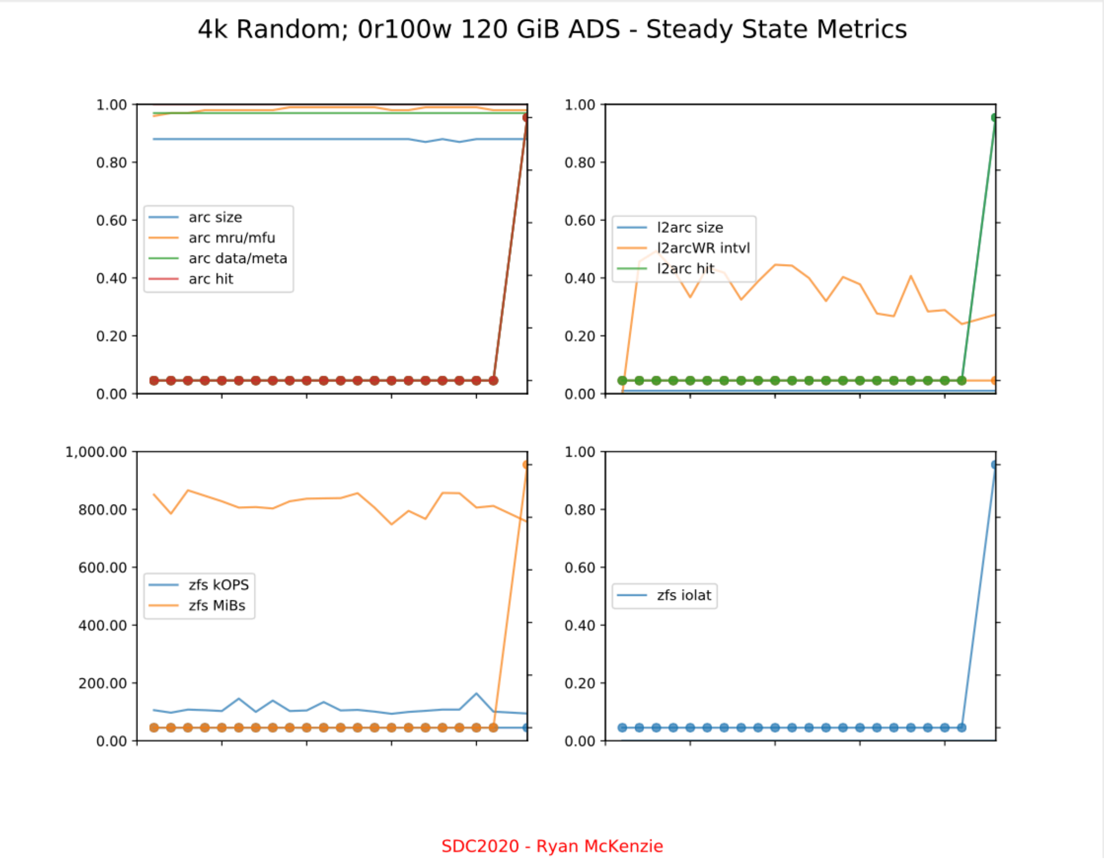
Charting Methodology

- Lines represent calculated value of each metric
- Corresponding color dotted line represents Boolean “steady” for each metric
- For this small ADS, system is steady quickly, as soon as the 20-interval window is loaded

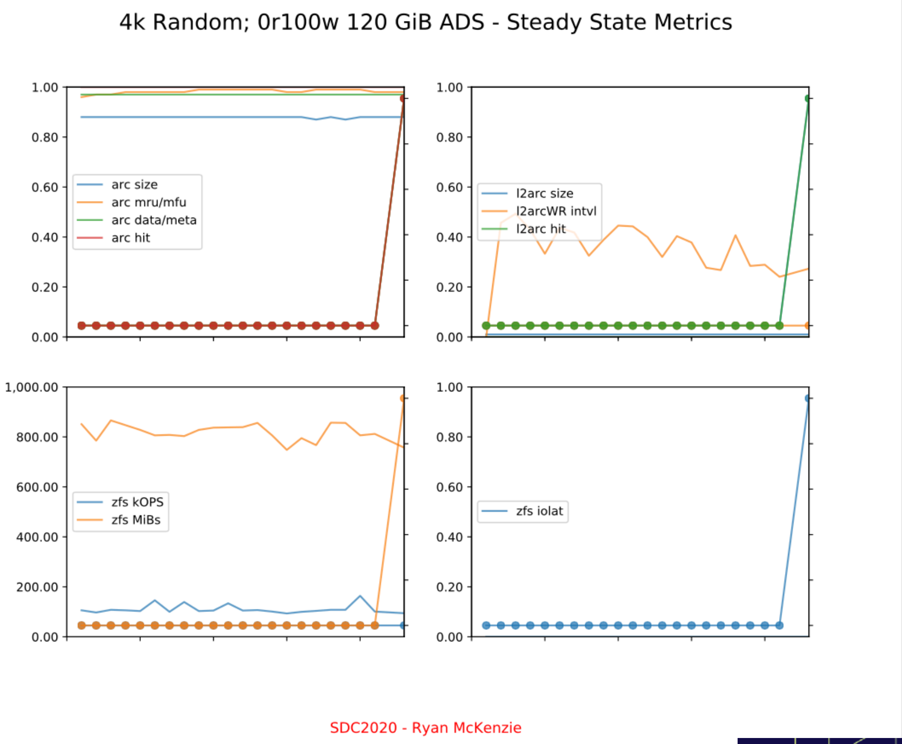


Charting Methodology

- Note that during this test cycle, something was wrong with ZFS latency parsing
- You can see that the metric failure state is positive (steady)

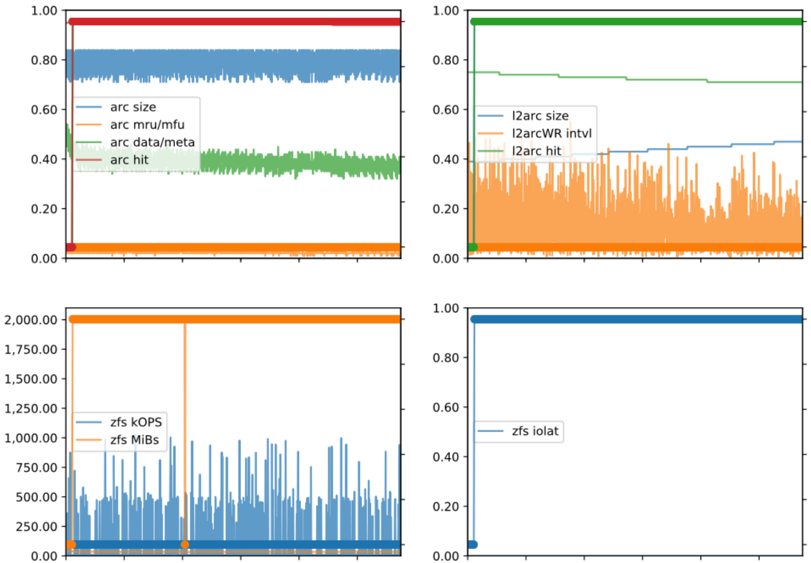


Random 4k Small ADS – Read vs Write



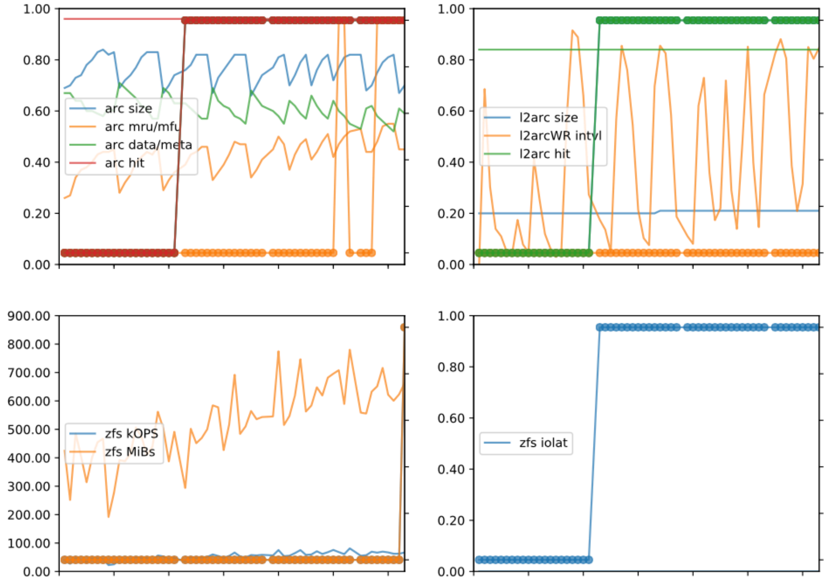
Random 4k Large ADS – Read vs Write

4k Random; 100r0w 4800 GiB ADS - Steady State Metrics



SDC2020 - Ryan McKenzie

4k Random; 0r100w 4800 GiB ADS - Steady State Metrics



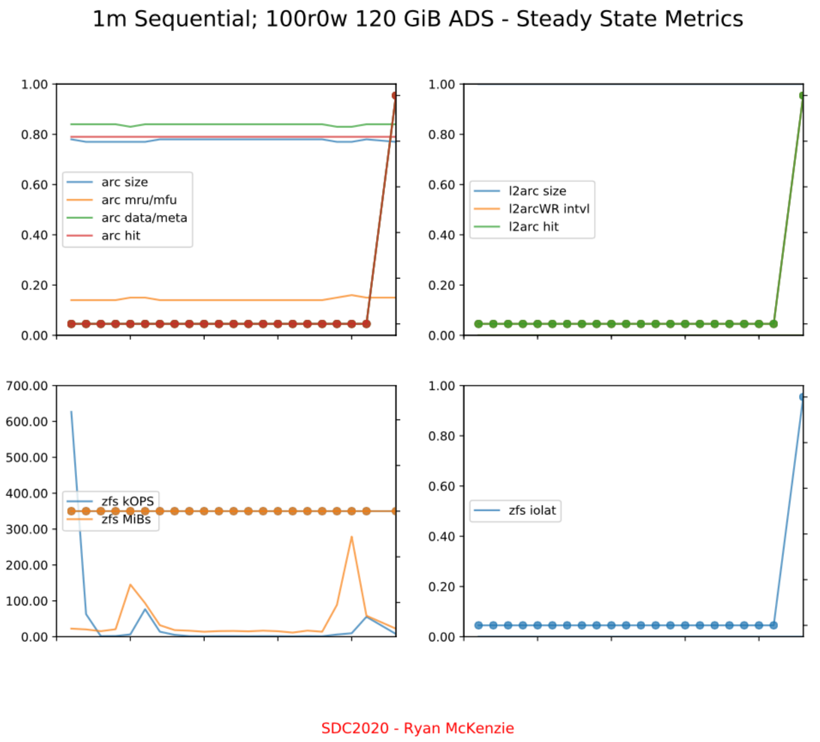
SDC2020 - Ryan McKenzie

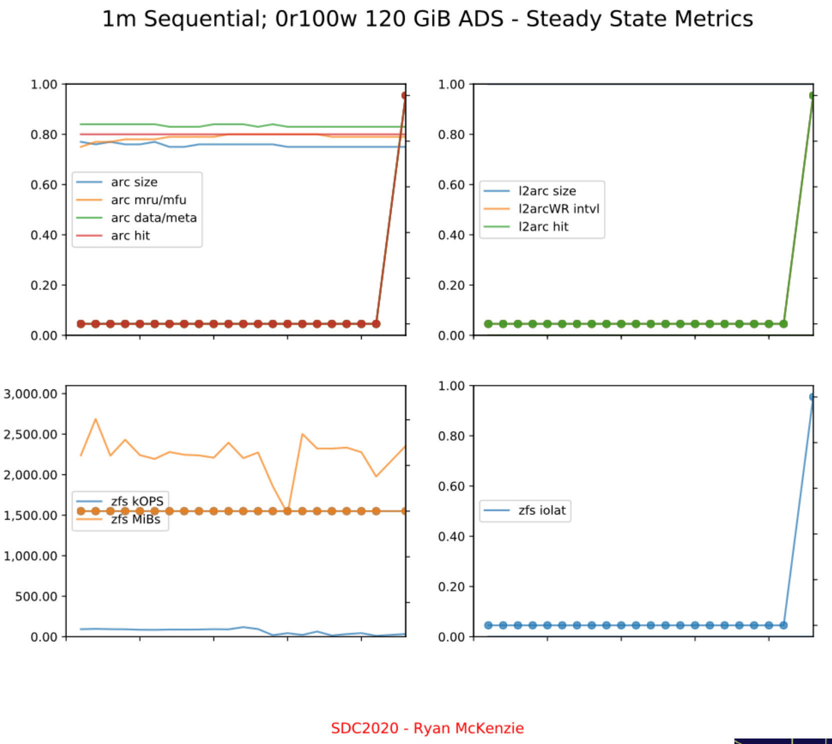
Random 4k Midsize ADS – 70/30 Mix

- Note decreasing MRU over time, slope not significant enough to violate steady state detection
- Note that steady state metrics are mostly satisfied early on, but L2ARC warm heuristic is met much later

SDC2020 - Ryan McKenzie

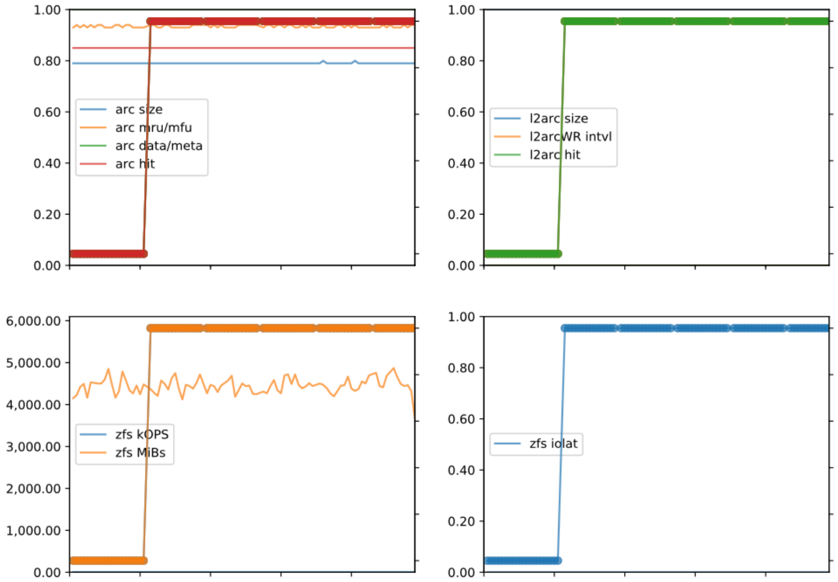
Sequential 1m Small ADS Read vs Write





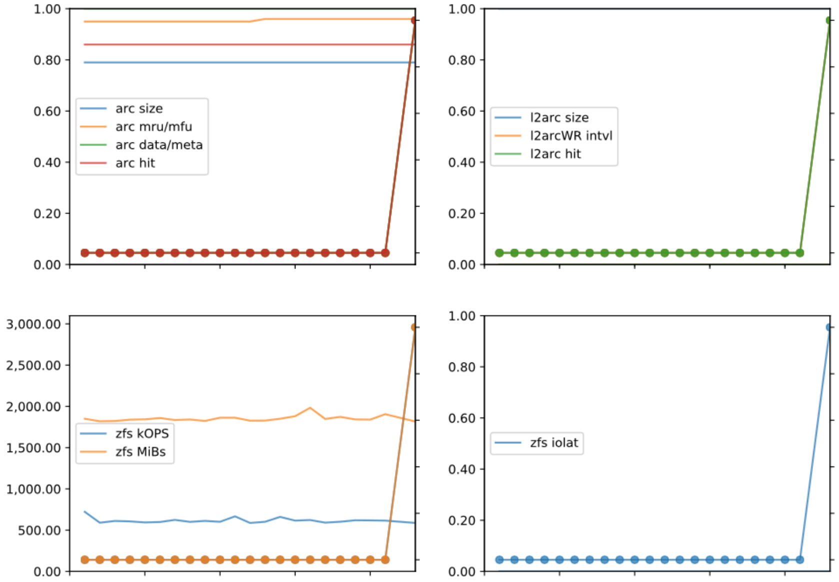
Sequential 1m Large ADS Read vs Write

1m Sequential; 100r0w 4800 GiB ADS - Steady State Metrics



SDC2020 - Ryan McKenzie

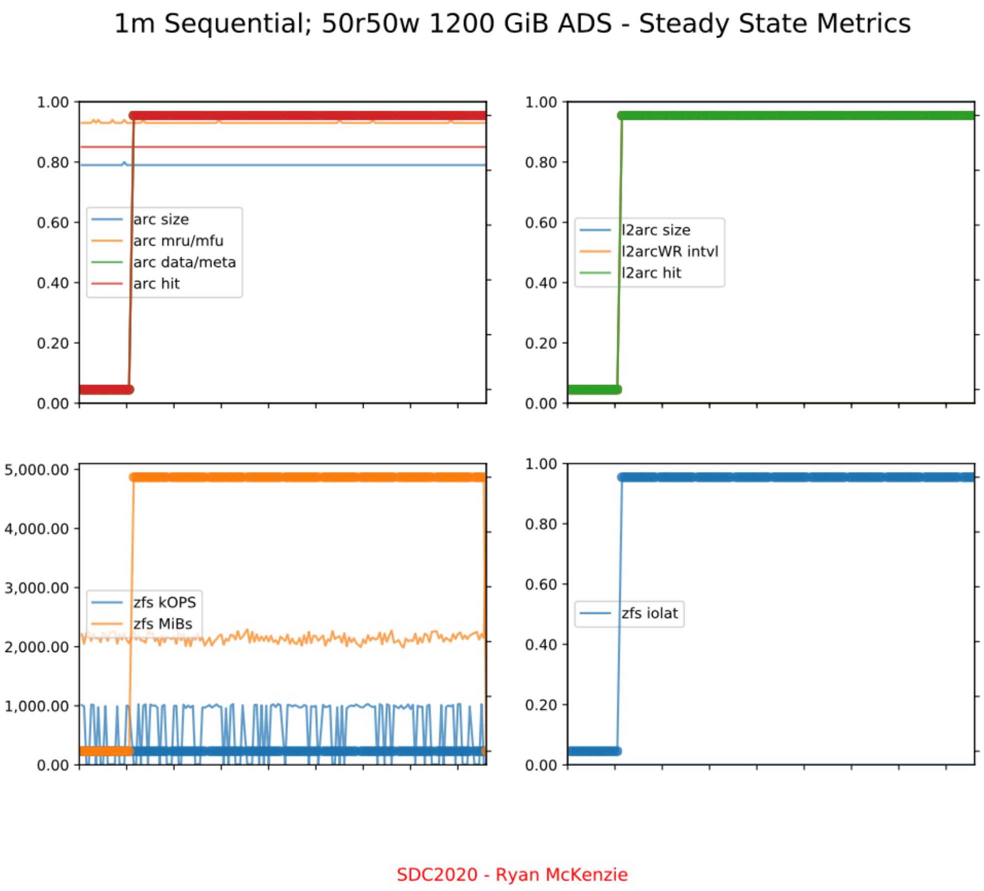
1m Sequential; 0r100w 4800 GiB ADS - Steady State Metrics



SDC2020 - Ryan McKenzie

Sequential 1m Midsize ADS – 50/50 Mix

- Sequential workload, even with 1.2 TiB active data and mixed r/w is not very exciting…
- Note that in the absence of L2ARC, the overall number of metrics and the number of metrics needed to pass are reduced by 3



Observations: Challenges

- Workload Matters!
  - Large block sequential workloads seem very stable, especially with no L2ARC
  - Writes are somewhat unstable for random workloads with L2ARC, bursts correlate to L2ARC write bursts

Observations: Challenges

- Workload Matters!
  - Small ADS that fits in ARC is very stable, io path deterministic
  - Large ADS always takes longer to reach steady state, but only significantly so with L2ARC in the pool

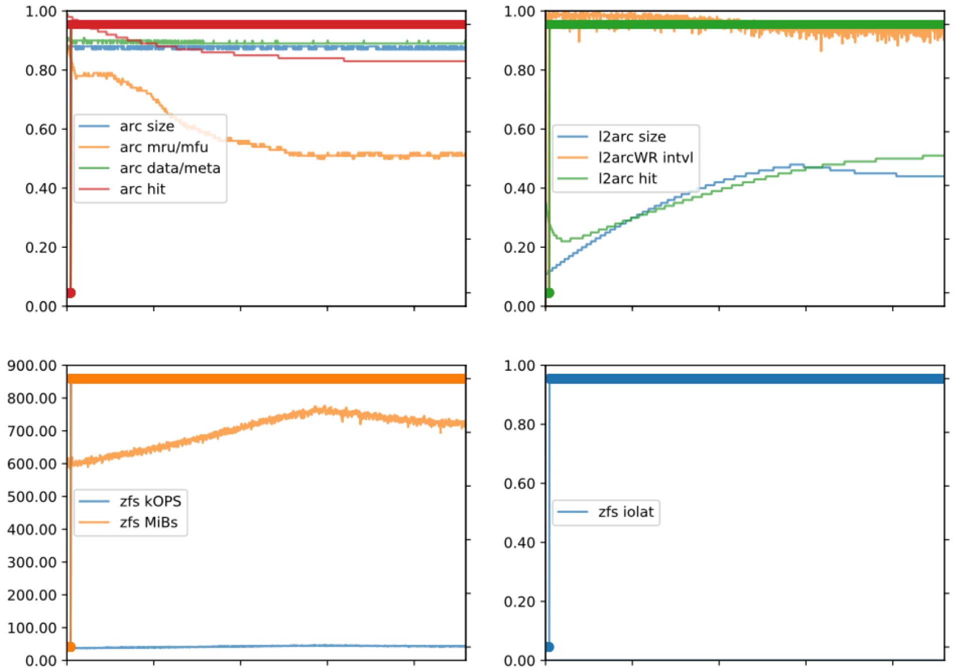
Observations: Challenges

- Some metrics may never reach a steady state!
  - L2ARC write rate metric never reached steady state in these tests (may be removed in future versions)

Observations: Challenges

- Long and slow warming of L2ARC
  - Slope may show as “steady” but performance still not at peak.
  - Can work on this with better preconditioning

32k Random; 70r30w 4800 GiB ADS - Steady State Metrics



SDC2020 - Ryan McKenzie

Optimization: Heuristics

- If 7 of 10 metrics (or 4 of 7 when no L2ARC) are satisfied, we declare system to be in steady state
  - Tunable

- If metrics are satisfied, we still wait until at least 85% of the ADS is ”in cache”
  - Unless ARC and L2ARC are both 80% full or higher
  - Tunable

Overall Results

- Reduction in wait time for deliverables
  - Test Engineer Time reduced by 1/3
    - no more checking intermediate results and deciding on steady state manually, especially with large L2ARC
  - System Utilization more than doubled to 18-20 hours per day unattended

Overall Results

- Better Repeatability
  - Anomalous measurements nearly eliminated
- Better Baselines/Comparability
  - Now in use for 2 complete product cycles

- Still room for Improvement!

Future Work: Metrics

- A better way to collect pool op latency
- Remove L2ARC writes metric
- Add metric(s) for avg. device busy for different pool device classes
  - i.e. data, SLOG, cache, fusion

Thank You!

References

Solid State Storage (SSS) Performance Test Specification (PTS) - V2.01

- www.snia.org/sites/default/files/technical_work/PTS/SSS_PTS_2.0.1.pdf

SNIA Solid State Storage Performance Test Specification (Easen Ho, 2011)

- www.snia.org/sites/default/files/HoEasen_SNIA_Solid_State_Storage_Per_Test_1_0.pdf

"ARC: A Self-Tuning, Low Overhead Replacement Cache"

- Nimrod Megiddo and Dharmendra S. Modha
- 2nd USENIX Conference on File Storage Technologies (2003)
- www.usenix.org/conference/fast-03/arc-self-tuning-low-overhead-replacement-cache

"Activity of the ZFS ARC", Brendan Gregg's Blog (January 9, 2012)

- dtrace.org/blogs/brendan/2012/01/09/activity-of-the-zfs-arc/

"ZFS L2ARC", Brendan Gregg's Blog (July 22, 2008)

- www.brendangregg.com/blog/2008-07-22/zfs-l2arc.html

Please take a moment
to rate this session.

Your feedback matters to us.