HotStorage 2022

ScalaRAID: Optimizing Linux Software RAID System for Next-Generation Storage

Shushu Yi, Yanning Yang, Yunxiao Tang,
Zixuan Zhou, Junzhe Li, Chen Yue,
Myoungsoo Jung, Jie Zhang

CHASELab
Computer Hardware And System Evolution Laboratory

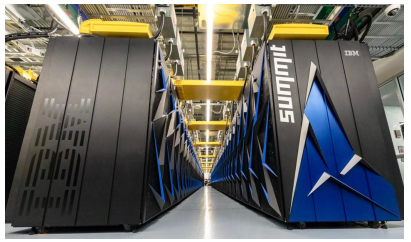
PEKING UNIVERSITY

KAIST

Motivation: Why We Need Software RAID?

- **RAID is widely used in multiple domains**
  - Superb performance, capacity, and reliability

Datacenters



Supercomputers

Clouds

- **H/W RAID card can't carry high-speed SSDs**

RAID card: 16 GB/s[1]

x 28

SSD: 560 MB/s [2]

x 2 !

SSD: 7 GB/s [3]

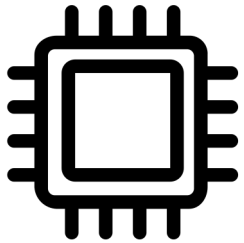
- **Software RAID is a better candidate**
  - More suitable for next-generation storage
  - More flexible and convenient

Array of many high-speed SSDs

?

Software engine

&

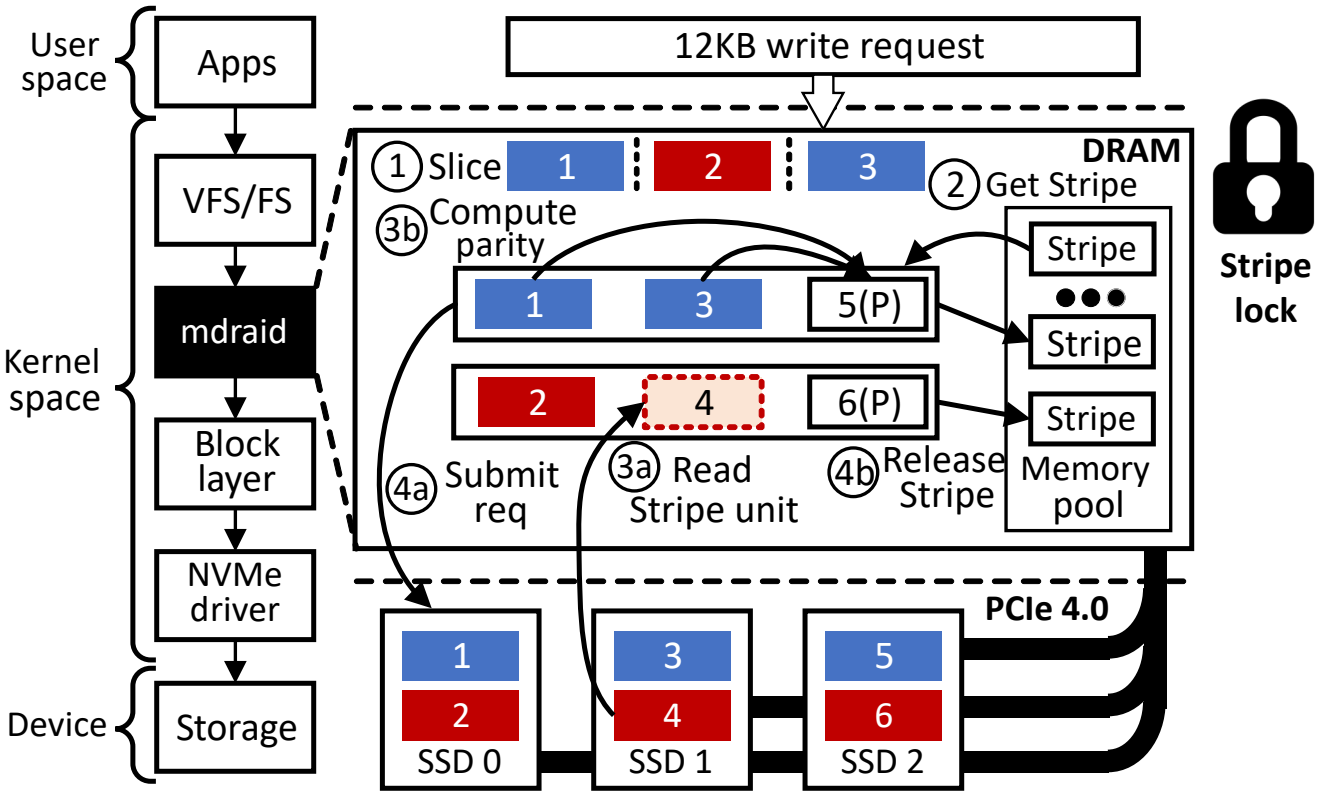


Host CPUs

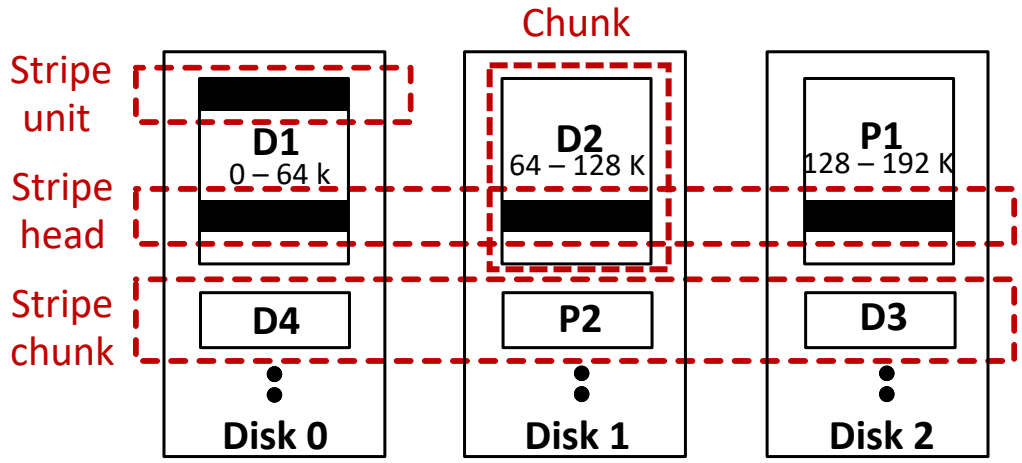
[1] Hight Point SSD7101A-1. https://www.highpoint-tech.com/product-page/ssd7101a-1
[2] Samsung 870EVO. 2021. https://www.samsung.com/us/computing/ memory-storage/solid-state-drives/870-evo-sata-2-5-ssd-250gb-mz-77e250b-am
[3] Samsung 980Pro. 2020. https://www.samsung.com/us/computing/memory-storage/solid-state-drives/980-pro-pcie-4-0-nvme-ssd-1tb-mz-v8p1t0b-am/

Background: Review of Linux Software RAID

- **mdraid layer in Linux storage stack**
  - Manage underlying block devices
  - Provide I/O services for VFS/FS

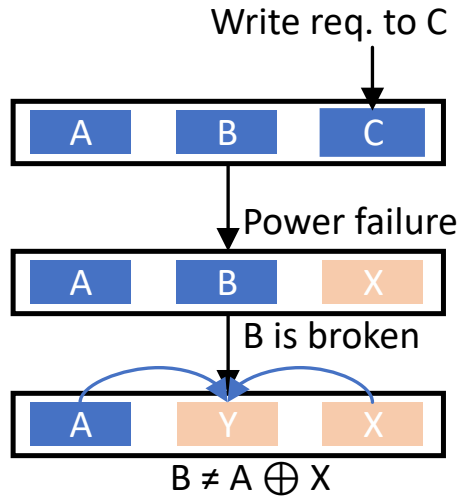


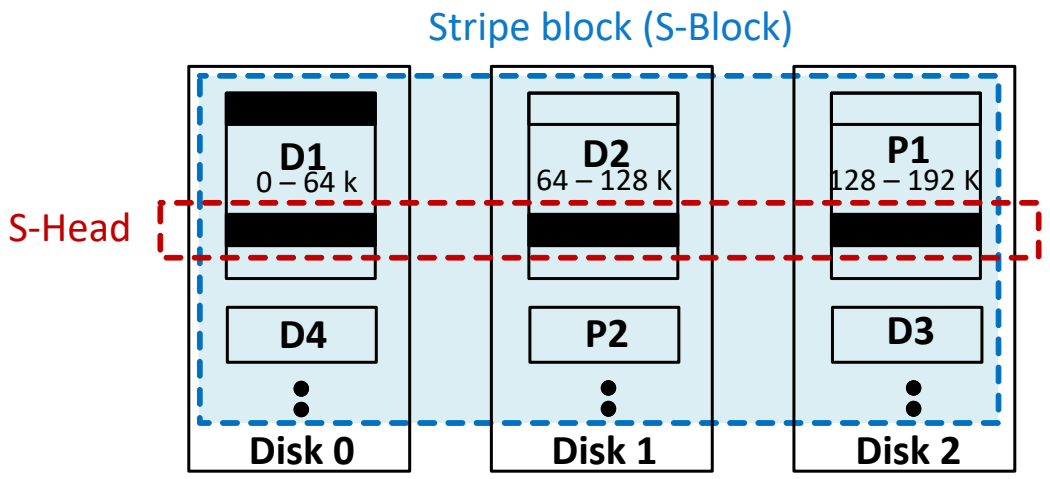
- **Data organization of a RAID 5 system**
  - Chunk & stripe chunk (S-Chunk)
  - Stripe unit (S-Unit) & stripe head (S-Head)

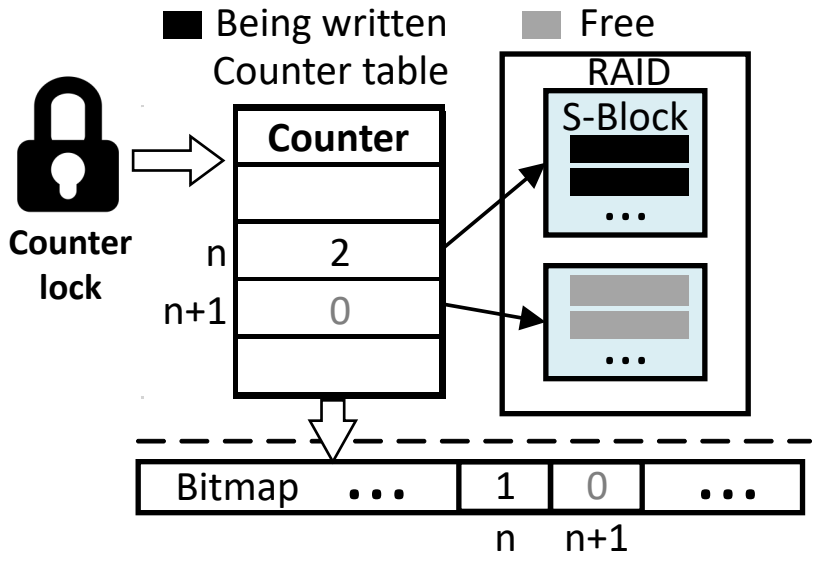


Background: Crash Consistency Guarantee of mdraid

- **Issue: crash consistency of mdraid**
  - E.g., power failure occurs in the process of chunk write;
  - Chunk becomes uncertain, which is harmful to the fault tolerance.
- **Default solution: bitmap mechanism**
  - A group of S-Heads are clustered as Stripe block;
  - Counter records # of S-Heads in an S-Block that are being written;
  - Table is flushed back to member disks in batches.

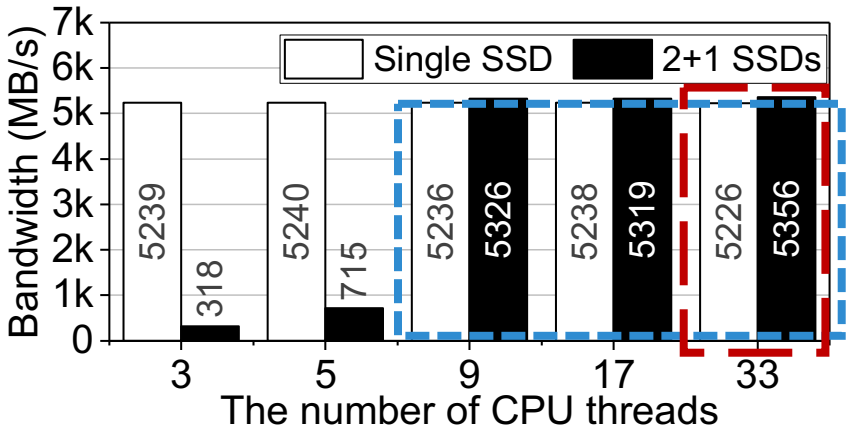


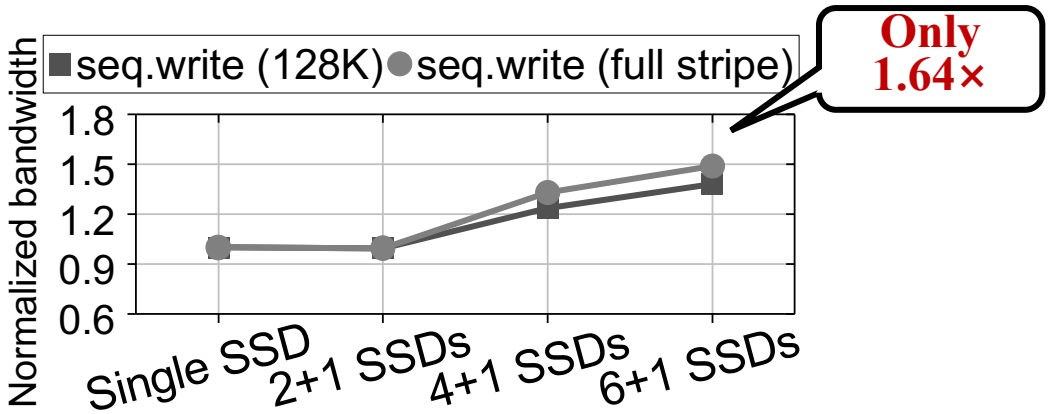




Challenge: Is mdraid Scalable?

- **Bandwidth cannot scale as the # of CPU threads increases**
  - Write bandwidth saturates when the # of CPU threads reaches 9;
  - Cannot exceed the bandwidth of a single SSD even with 33 threads.
- **Peak throughput cannot scale as the # of SSDs increases**
  - Increasing the # of SSDs slightly improve throughput;
  - Performance of RAID (6+1 SSDs) slightly exceeds that of single SSD.





Unfortunately, mdraid is not scalable!

Challenge: What's the Problem?

Let's take a closer look at the software overheads of storage stack!

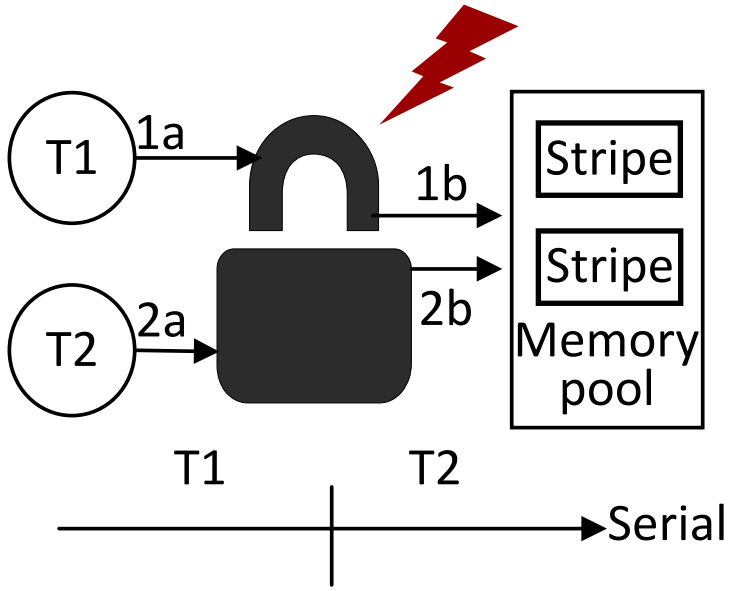
30.8% !!!

3.5

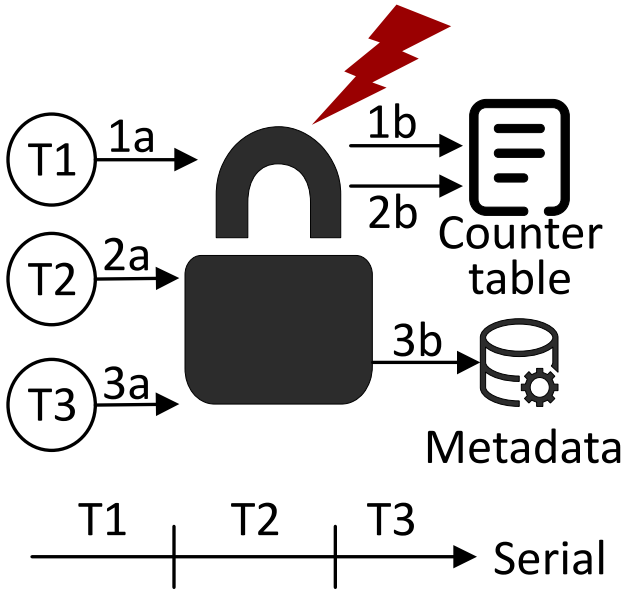
Locks become the main penalty of mdraid write operations!

Challenge: What's the Problem?

- **Details of the lock mechanism**



Stripe lock



Counter lock

CPU threads are serialized in front of the ill-designed locks!

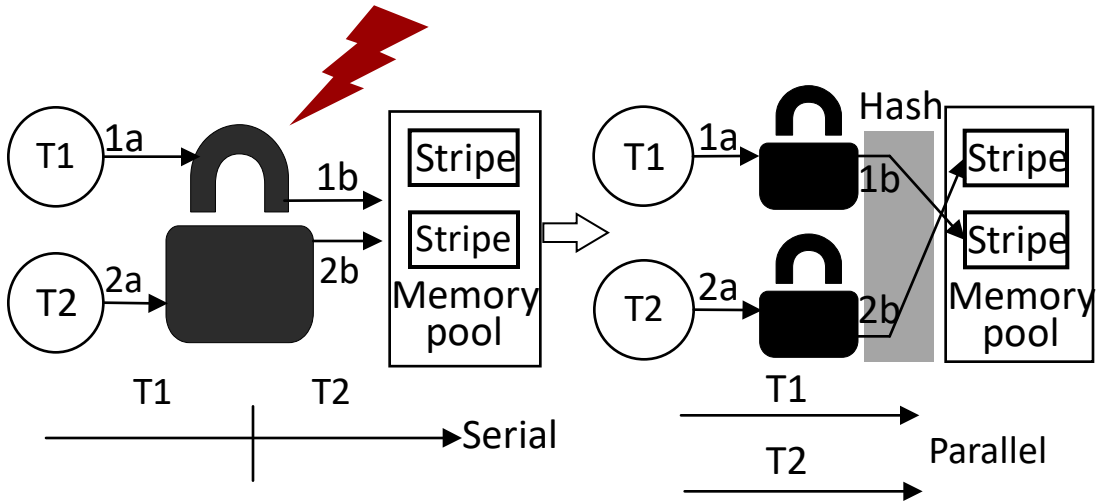
Our Solution: ScalaRAID

- **Fine-grained lock mechanism to maximize parallelism**
  - Refine the scope of lock management
  - Increase the # of locks with minor overheads

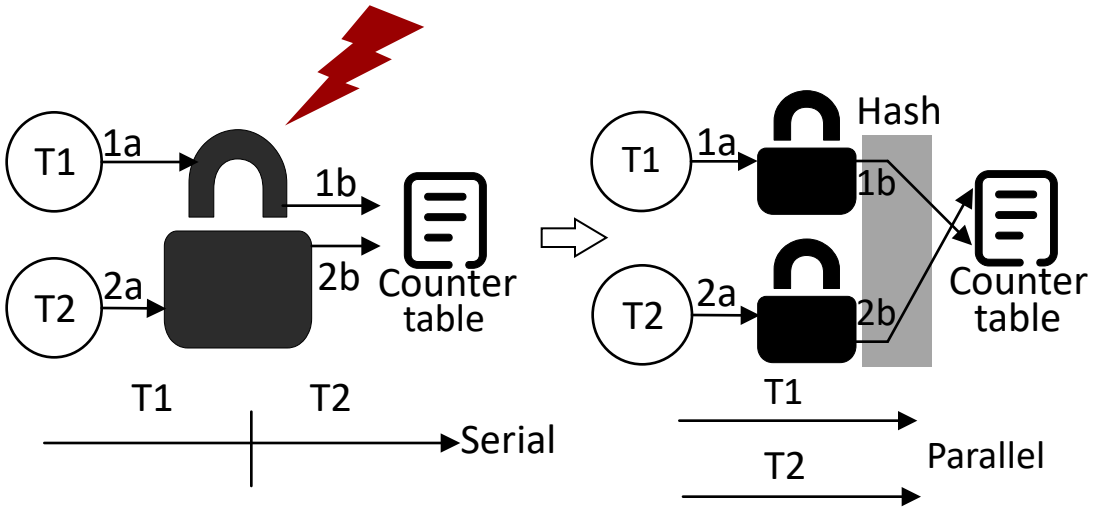
- **Customized data structure to avoid collisions**
  - Different types of locks to manage data and metadata
  - Scatter segments across the entire address range

Fine-grained lock mechanism to maximize parallelism

Idea: Multiple locks to manage different scope of resources.



More Stripe locks



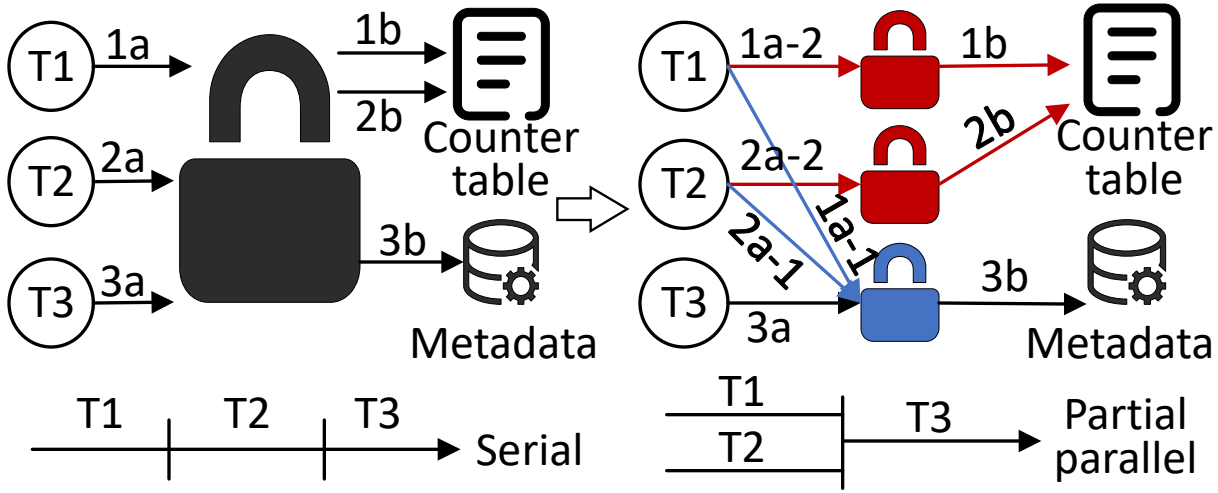
More counter locks

Customized data structure to avoid collisions

Counter lock is used to manage both data and metadata. 🤔

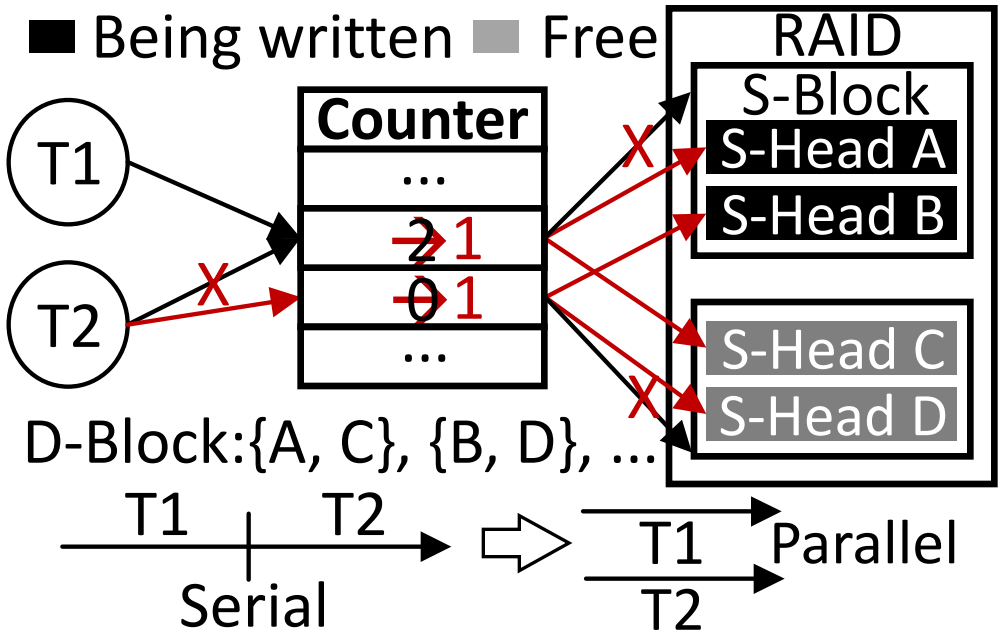
mdraid rarely updates metadata (e.g., RAID shrinking). 👀

Readers-writer lock!



Customized data structure to avoid collisions

- More locks cannot prevent CPU threads from competing for the same counter.
- Use distributed blocks (D-Block) to replace centralized stripe blocks.



Experimental Setup

- **Experimental platforms**
  - NVMe SSDs: 7,000/5,000 MB/s (R/W)
  - # of worker threads = # of fio thread[1]
  - 128 Stripe locks & 16,384 counter locks

Hardware	CPU: 26 core / 2.2 GHz, with hyperthreading
	Memory: 8 × 16 GB DDR4 DIMM
	SSDs: up to 7 × 1TB Samsung 980 Pro
Software	OS: Ubuntu 20.04
	Kernel: Linux v5.11.0
Test tools	Fio v3.16
	Perf v5.11
	mdadm v4.1

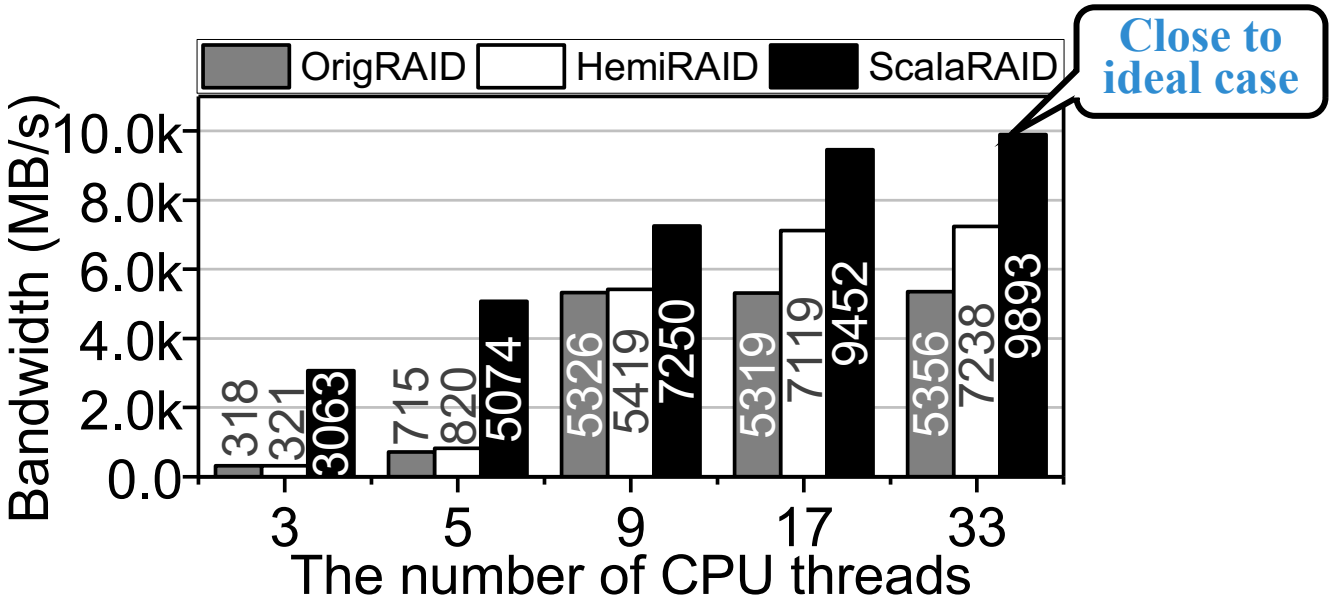
System configurations

- **Three schemes for comparison**
  - OrigRAID: adopting the default configurations of mdraid;
  - HemiRAID: increasing the number of Stripe locks to 128;
  - ScalaRAID: equipping every counter with a counter lock and employ D-Block.

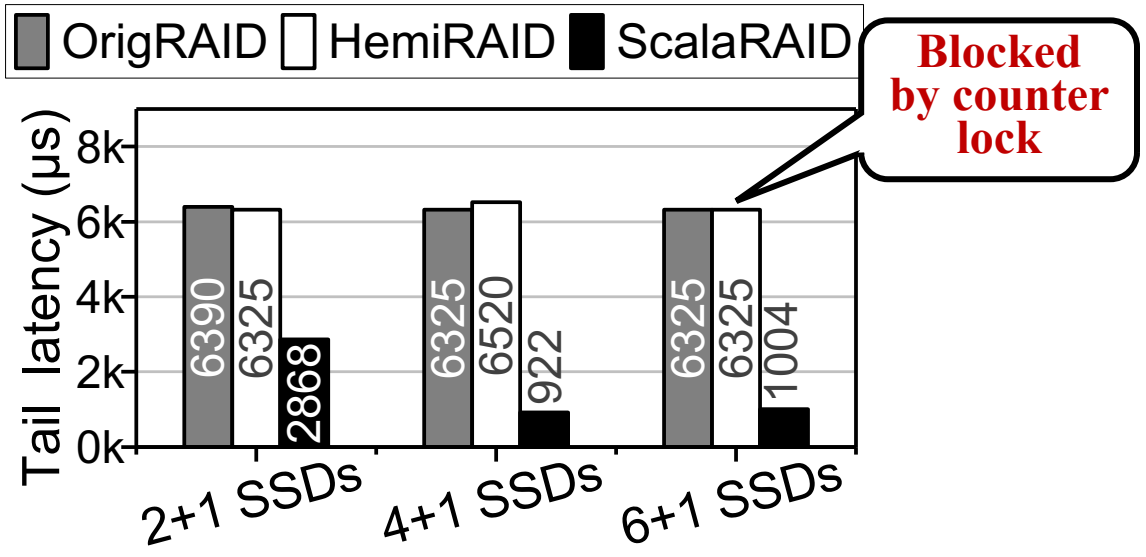
[1] Nikolaus Jeremic, Helge Parzyjegla, and Gero Muehl. 2016. Improving random write performance in homogeneous and heterogeneous erasure-coded drive arrays. ACM SIGAPP Applied Computing Review 15, 4 (2016), 31–53

Performance Comparison

- **Bandwidth of sequential write**



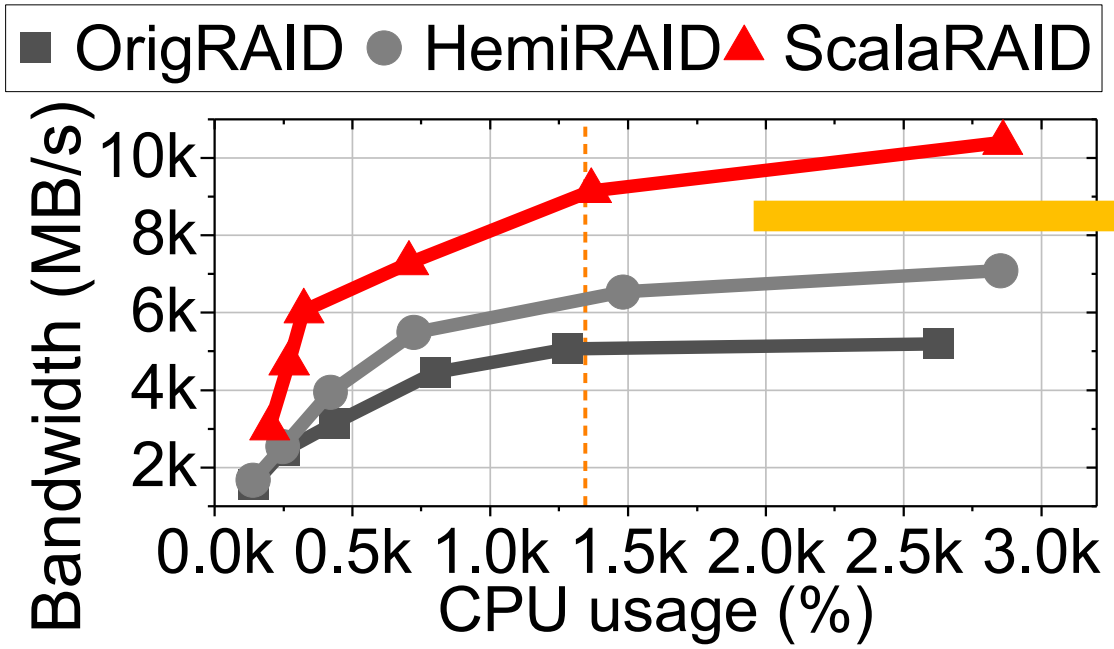
- **99.99th latency of sequential write**

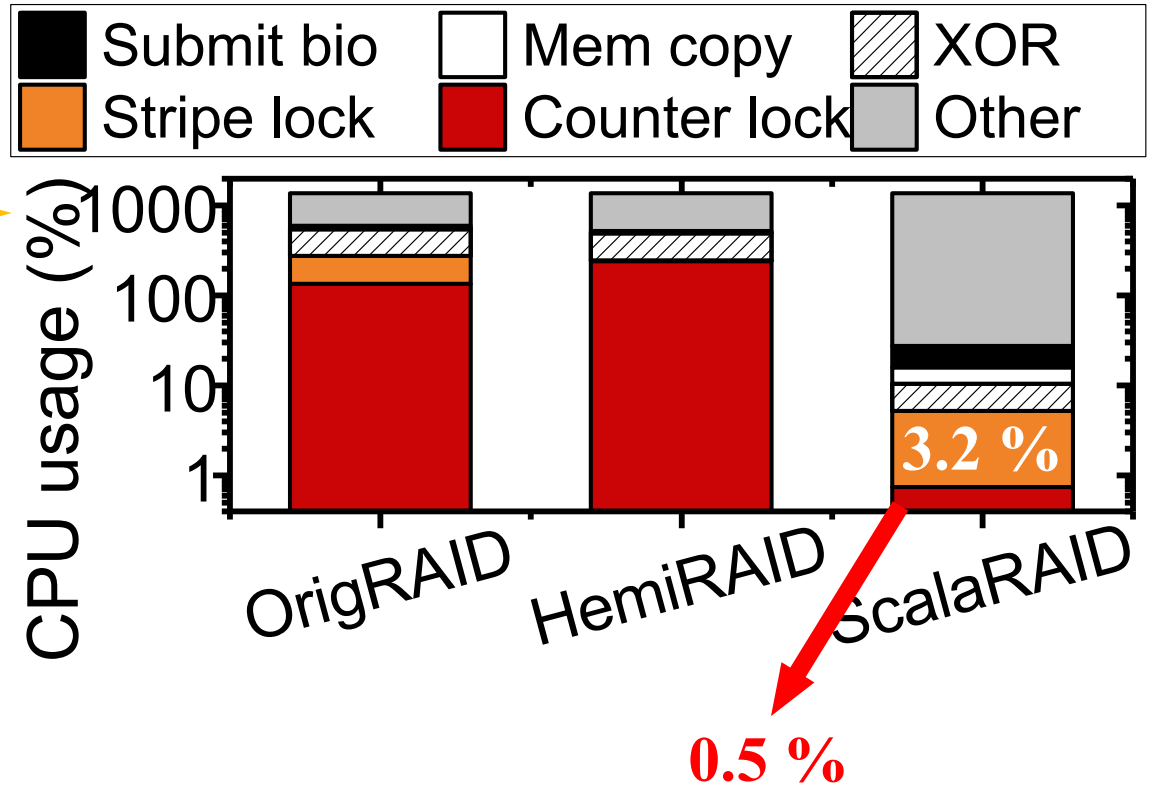


ScalaRAID improves bandwidth by 89.4% while decreasing 99.99th latency by 85.4%!

CPU Overhead Analysis

ScalaRAID achieves high bandwidth under the same CPU usage.





Conclusion

ScalaRAID achieves scalable performance for next-generation storage by optimizing the lock mechanism.

- Deeply analyze the CPU overheads of Linux software RAID;
- Propose fine-grained locks to maximize thread-level parallelism;
- Propose Customized data structures to avoid collisions;
- Improve throughput by 89.4% while decreasing 99.99th latency by 85.4%.

HotStorage 2022

Thank you!

ScalaRAID: Optimizing Linux Software RAID System for Next-Generation Storage

https://github.com/ChaseLab-PKU/ScalaRAID

Shushu Yi

firnyee@gmail.com

CHASELab

Computer Hardware And System Evolution Laboratory

PEKING UNIVERSITY

KAIST