LightNVM: The Linux Open-Channel SSD Subsystem

Matias Bjørling, Javier González, and Philippe Bonnet

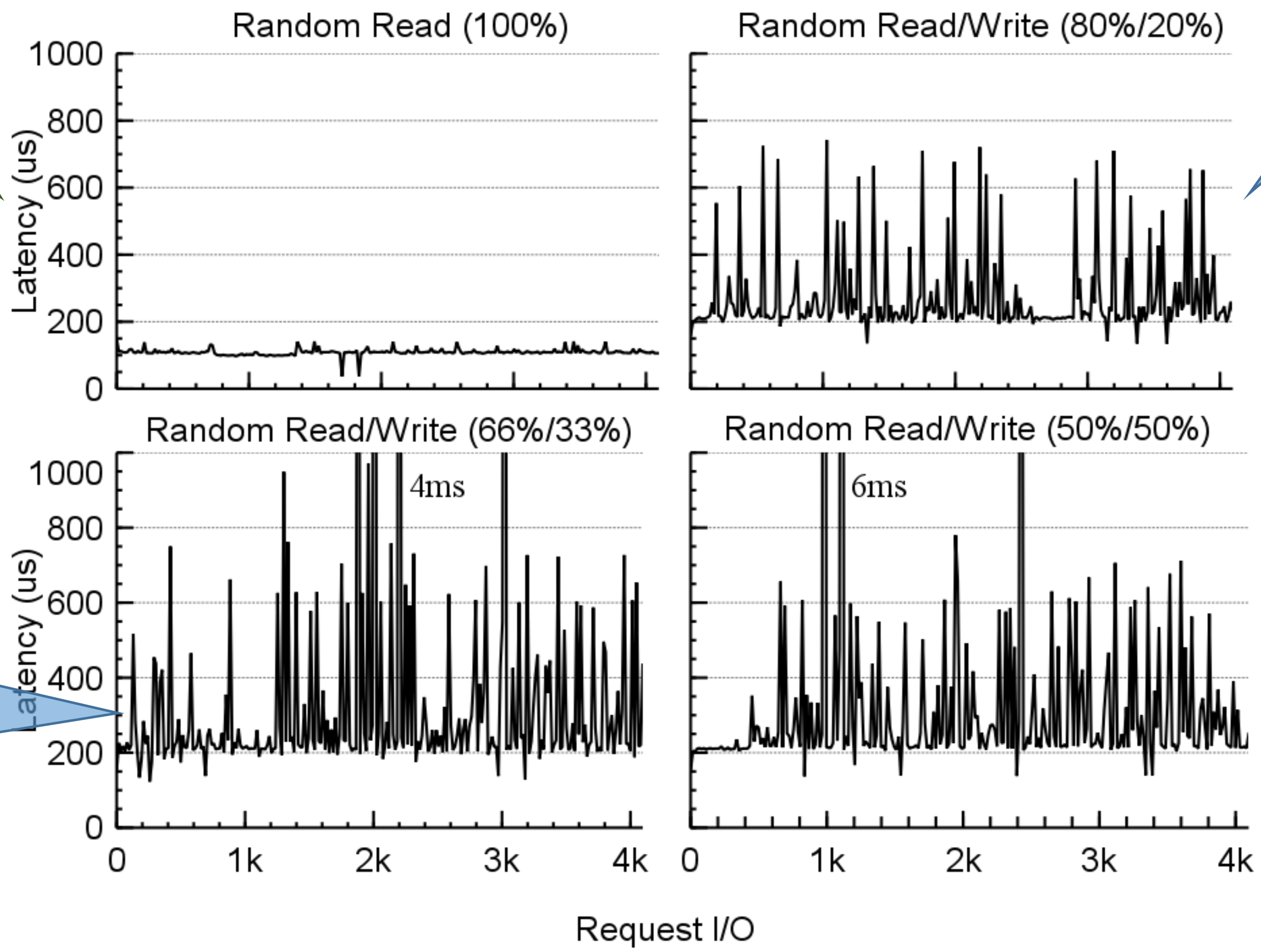
CNEXLABS

IT UNIVERSITY OF CPH

I/O Predictability and Isolation

0% writes and latency is consistent

20% writes makes big impact on read latency



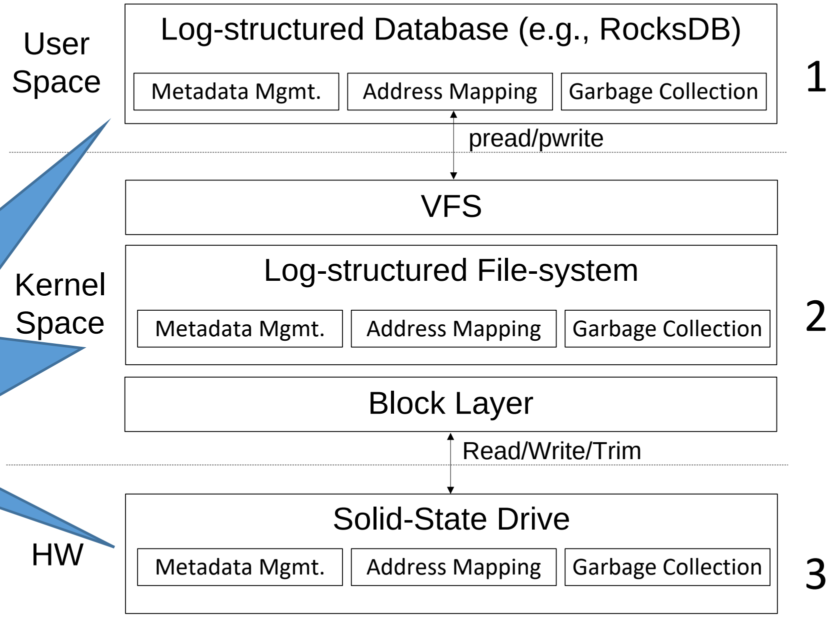
50% writes can make SSDs as slow as spinning drives...

I/O Performance is unpredictable due to writes being buffered

Log-on-log, Indirection, and Narrow I/O

Even if Writes and Reads does not collide from application
Indirection and loss of information due to a **Narrow I/O interface**

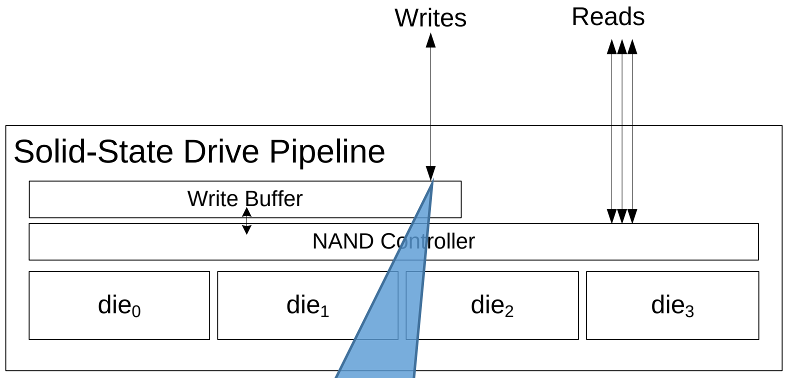
Log-on-Log



FTL-like implementation at multiple layers

Not able to align data on media = Write amplification increase + extra GC

Write Indirection & Lost State

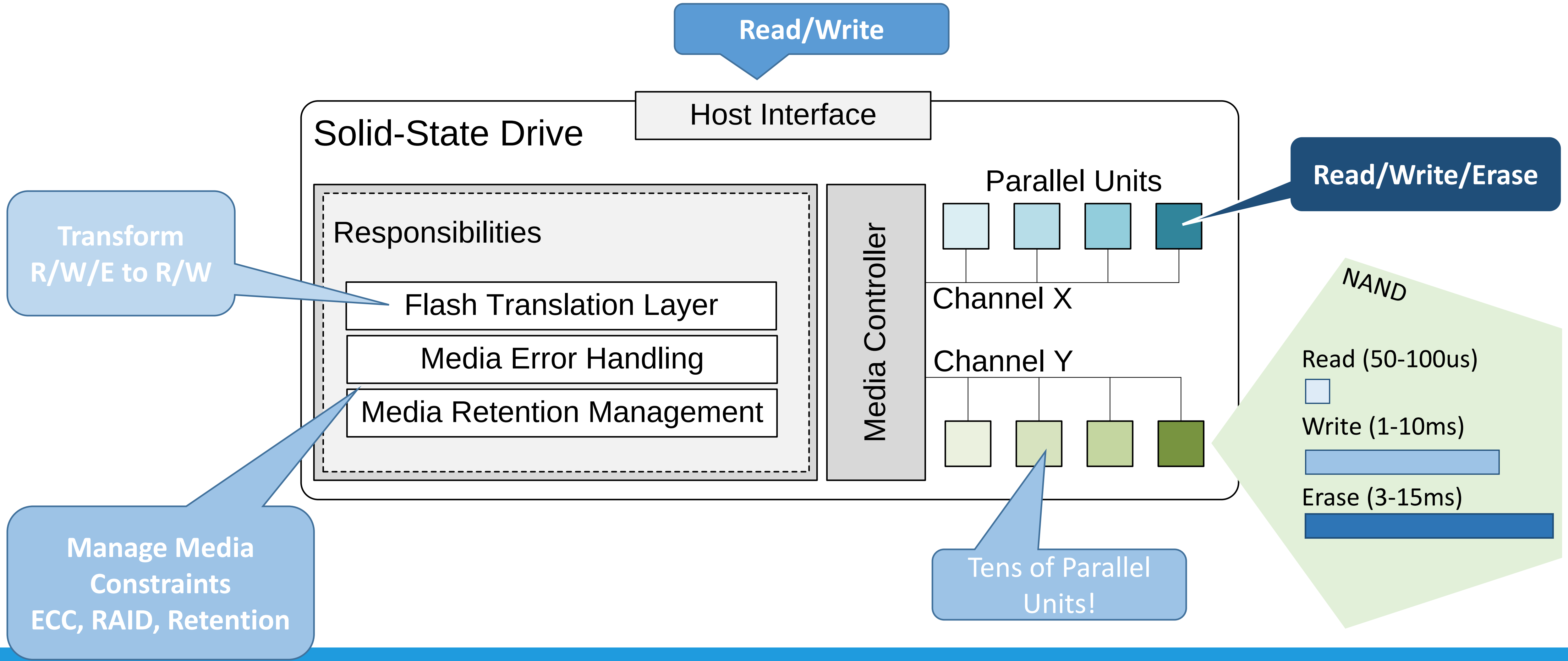


Writes Decoupled from Reads

Read/Write Interface makes Data placement + Buffering = **Best Effort**

Host does not know SSD state due to the narrow I/O Interface

Solid-State Drives and Non-Volatile Media



New Storage Interface that provides

- **Predictable I/O**
- **I/O Isolation**
- **Reduces Write Amplication**
- **Removal of multiple log-structured** data structures
- **Intelligent data placement** and **I/O scheduling decisions**
- **Make the host aware of the SSD state** to make those decisions

Contributions

1. Physical Page Addressing (PPA) I/O Interface
2. The LightNVM Subsystem
3. pblk: A host-side Flash Translation Layer for Open-Channel SSDs
4. Demonstrate the effectiveness of this interface

Physical Page Addressing (PPA) Interface

- Expose geometry of the SSD
  - Logical/Physical geometry
  - Performance
  - Media-specific metadata (if needed)
  - Controller functionalities
- Hierarchical Address Space
  - Encode geometry into the address space
- Vector I/Os
  - Read/Write/Erase

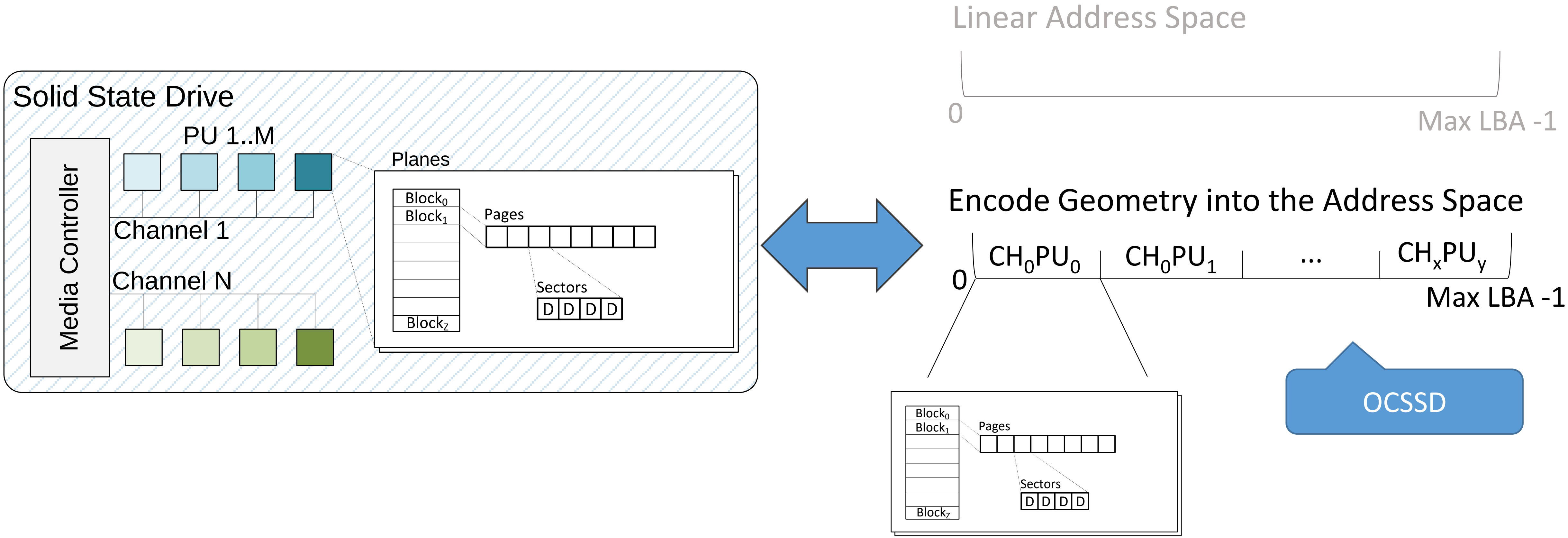
Up to the SSD vendor

Encode parallel units into the address space

Efficient access to the given this new address space

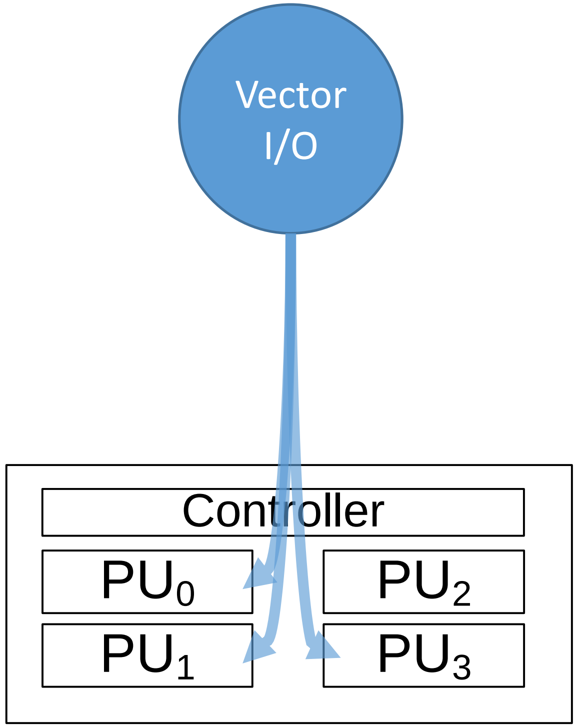
Encode Geometry in Address Space

Channels **->** Parallel Units **->** Planes **->** Blocks **->** Pages **->** Sectors



Vector I/O Access

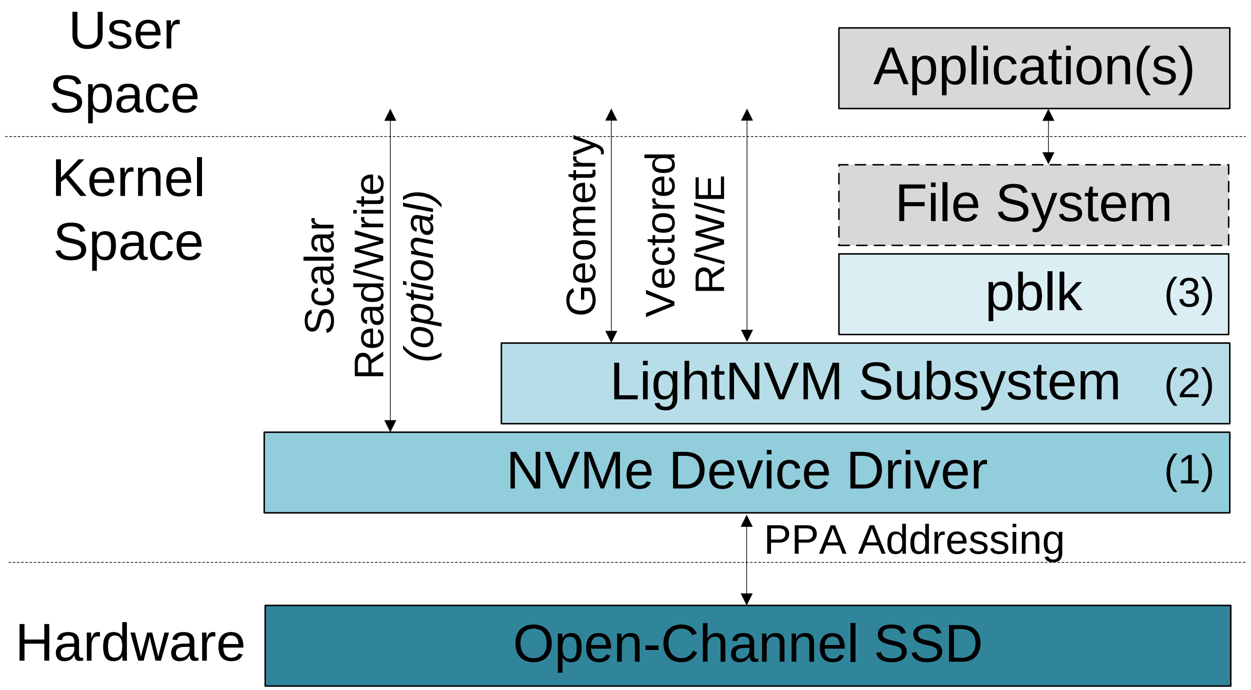
- Obtain higher throughput through parallel units
- Large overhead if I/Os is separately issued
- Introduce vector I/O interface to enable host to submit I/Os to multiple PUs using one command
- Vector Read/Write/Erase using scatter/gather address list



#	LBA
0	CH_0, PU_0, Sector 120
1	CH_0, PU_3, Sector 64
2	CH_0, PU_1, Sector 212

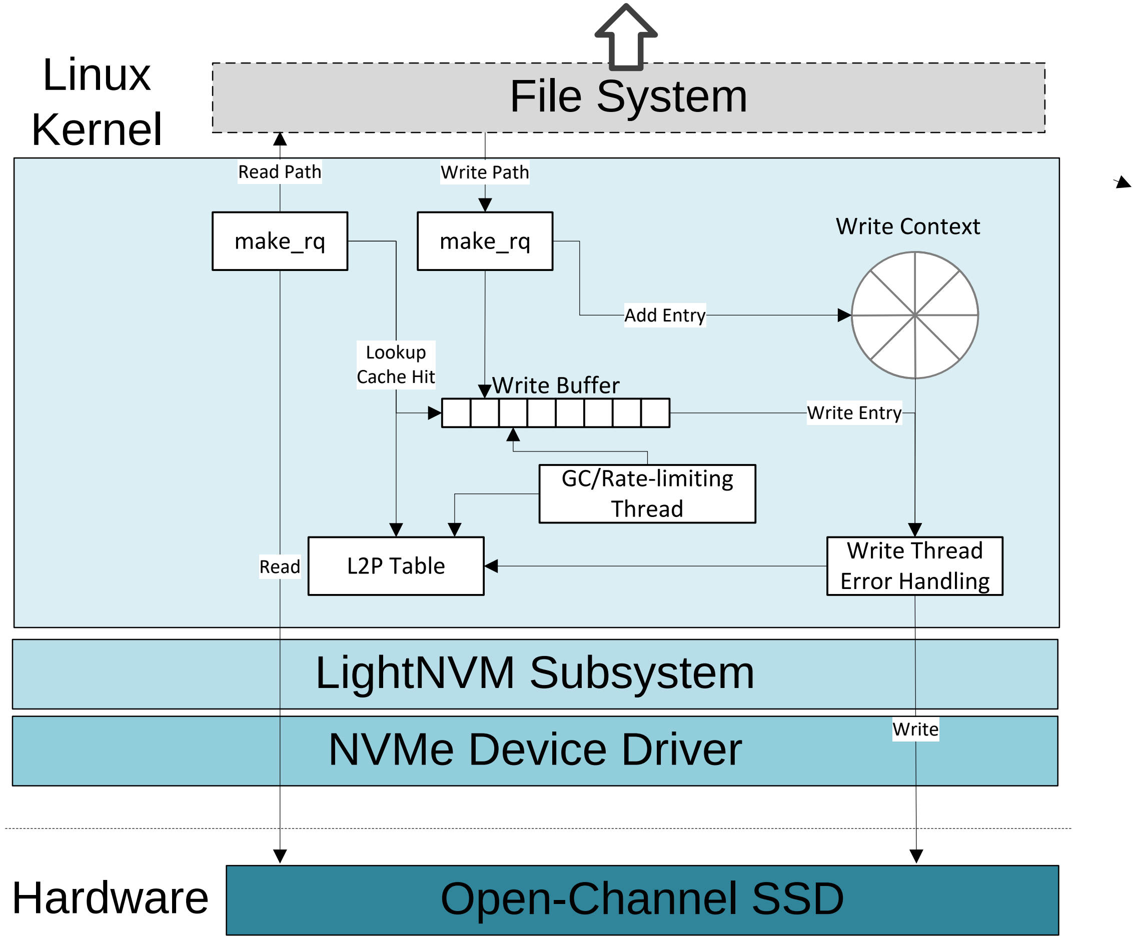
LightNVM Architecture

1. NVMe Device Driver
   - Detection of OCSSD
   - Implements PPA interface
2. LightNVM Subsystem
   - Generic layer
   - Core functionality
   - Target management (e.g., pblk)
3. High-level I/O Interface
   - Block device using pblk
   - Application integration with liblightnvm



Host-side Flash Translation Layer - pblk

- Mapping table
  - Sector-granularity
- Write buffering
  - Lockless circular buffer
  - Multiple producers
  - Single consumer (Write Thread)
- Error Handling
  - Media write/erase errors
- Garbage Collection
  - Refresh data
  - Rewrite blocks



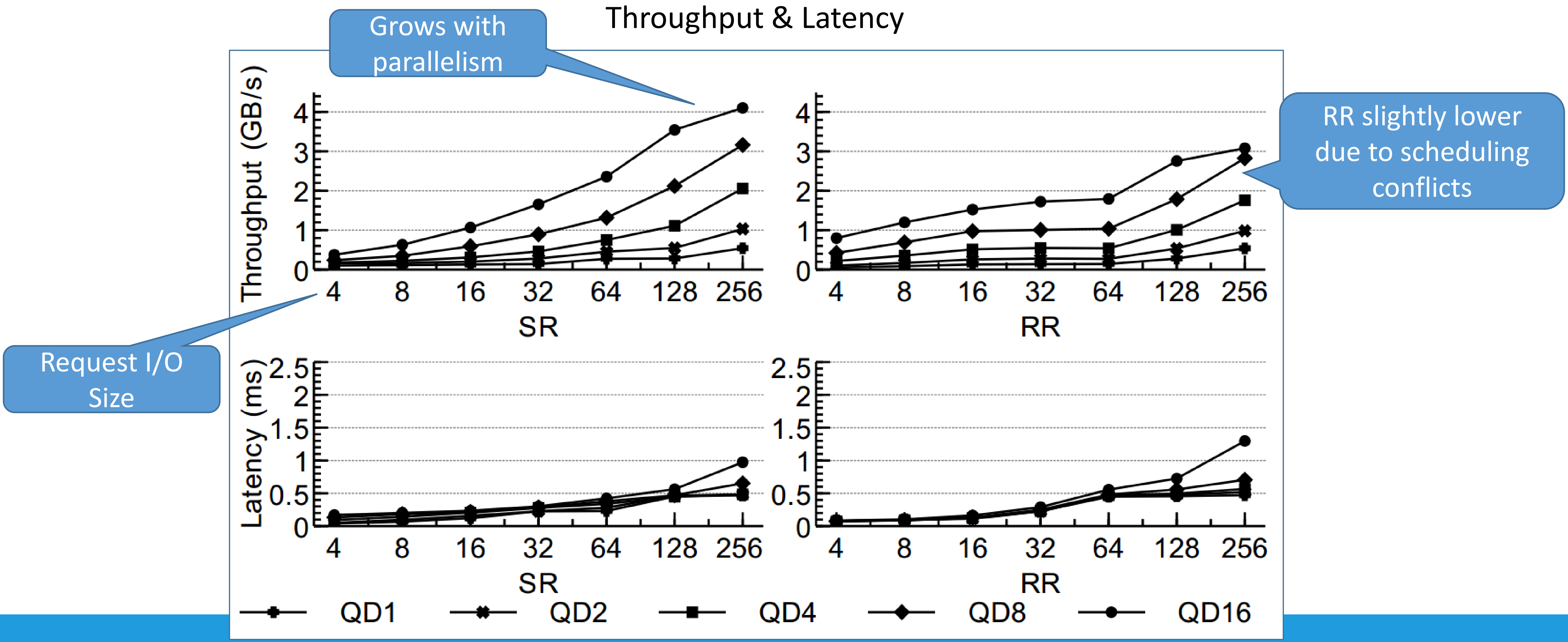
Experimental Evaluation

- CNEX Labs Open-Channel SSD
  - NVMe
  - PCIe Gen3x8
  - 2TB MLC NAND
- Geometry
  - 16 channels
  - 8 PUs per channel (Total: 128 PUs)
- Parallel Unit Characteristics
  - Page size: 16K + 64B user OOB
  - Planes: 4, Blocks: 1.067, Block Size: 256 Pages
- Performance:
  - Write: Single PU 47MB/s
  - Read: Single 108MB/s, 280MB/s (64K)

Evaluation

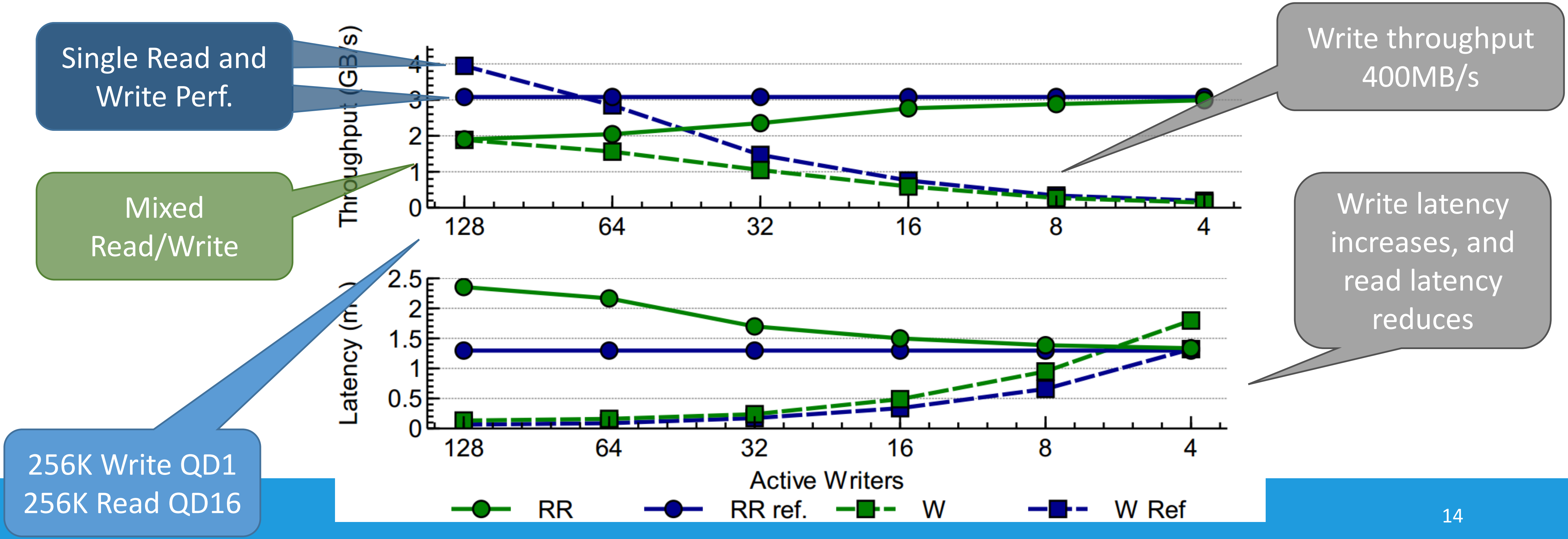
- Sanity check & Base
- Interface Flexibility
  - Limit # Active Parallel Write Units
  - Predictable Latency

Base Performance using Vector I/O



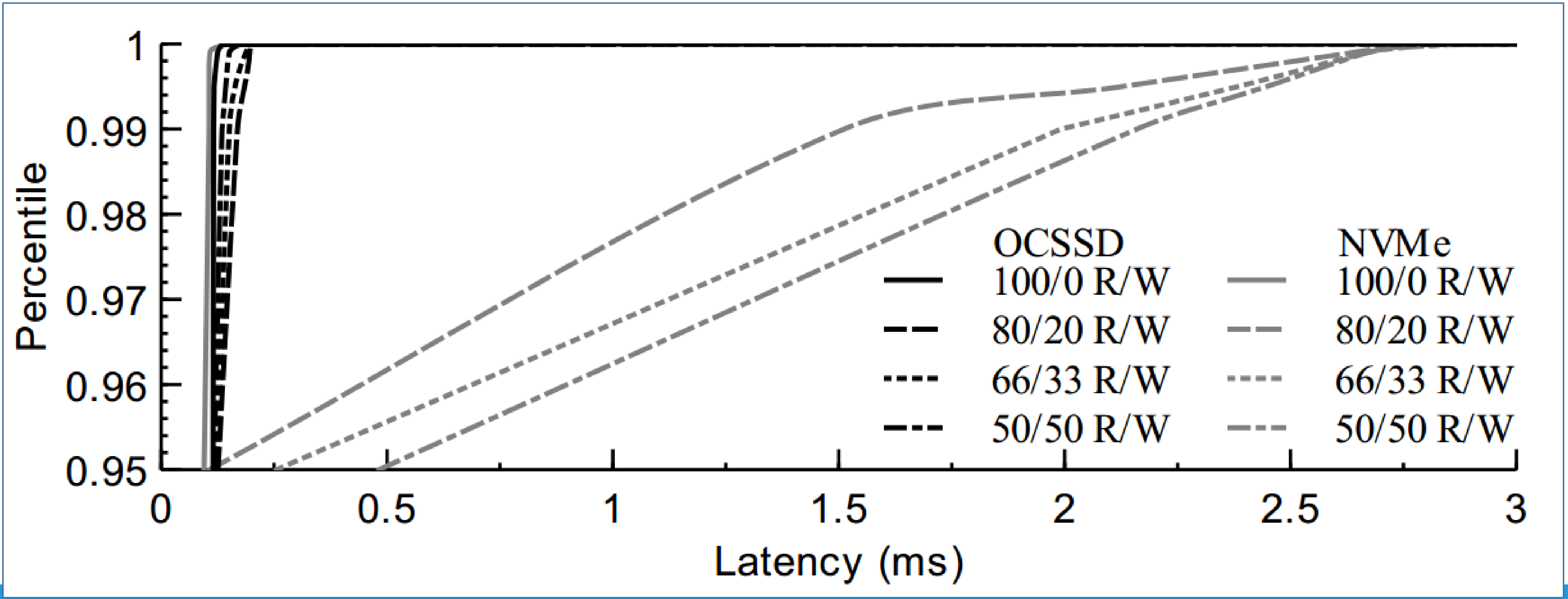
Limit # Active Writers

- A priori knowledge of workload. E.g., limit to 400MB/s Write
- Limit number of Active PU Writers, and achieve better read latency



Predictable Latency

- 4K reads during 64K concurrent writes
- Consistent low latency at 99.99, 99.999, 99.9999



Lessons Learned

1. **Warranty to end-users** – Users has direct access to media
2. **Media characterization is complex** and performed for each type of NAND memory – Abstract the media to a ”clean” interface.
3. **Write buffering** – For MLC/TLC media, write buffering is required. Decide if in host or in device.
4. **Application-agnostic wear leveling is mandatory** – Enable statistics for host to make appropriate decisions.

Conclusion

- Contributions
  - Physical Page Addressing (PPA) I/O Interface
  - The LightNVM Subsystem
  - pblk: A host-side Flash Translation Layer for Open-Channel SSDs
  - Demonstrate the effectiveness of the interface
- Linux kernel subsystem for Open-Channel SSDs
  - Initial release in Linux kernel 4.4.
  - User-space library (liblightnvm) support with Linux kernel 4.11.
  - Pblk upstream with Linux kernel 4.12.
- Physical Page Addressing Specification is available
- The right time to dive into Open-Channel SSDs
  - More information available at: http://lightnvm.io