SSD Architecture Challenges with DRAM

Dan Helmick, PhD
Principal Architect

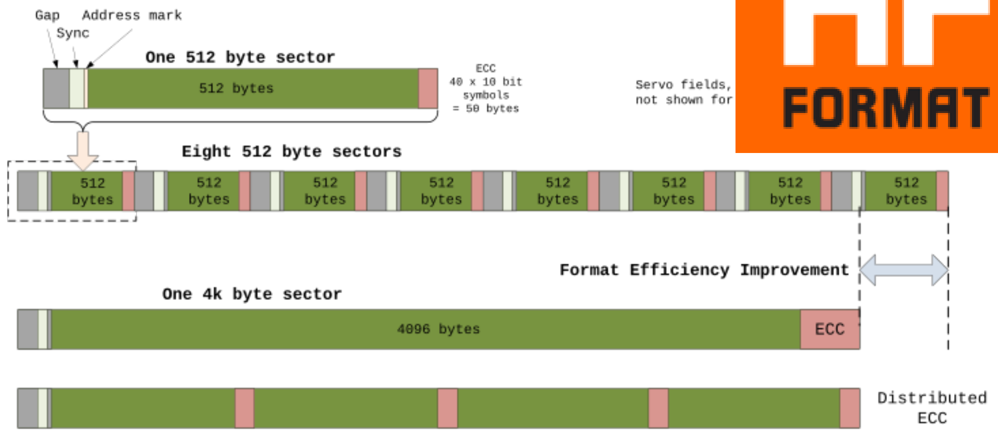
Outline

- What, Why, and How do SSDs IU?
- Solving the IU Capacity Limit
- Customer Ecosystem for Large IUs

What, Why, and How do SSDs IU?

History of the Indirection Unit (IU) – HDD Sectors

- Beginning of Time: Cylinder-Head-Sector (CHS)
  - The “Open Channel” of HDDs had variable sector sizes
  - 512B grew to dominate the marketplace through the 1980’s
- Early 1990’s: 512B sector sizes were the industry norm
  - SASI and ATA leading the transition away from CHS
- 1998: The Advanced Format Technology Committee aligned the industry on 4KiB sector sizes
  - ECC efficiency gains were a primary motivator to increase sector size
  - Alignment to OS memory page size was a primary reason for selecting 4KiB
- 2007-2008: Enterprise SSDs start
  - Fusion IO and other vendors begin shipping volume SSDs
- 2010: First 4Kn (4KiB native) HDD ships
  - Continued to emulate 512B



Credit: Wikipedia

Comparison of 512- and 4096-byte sector formats[7]

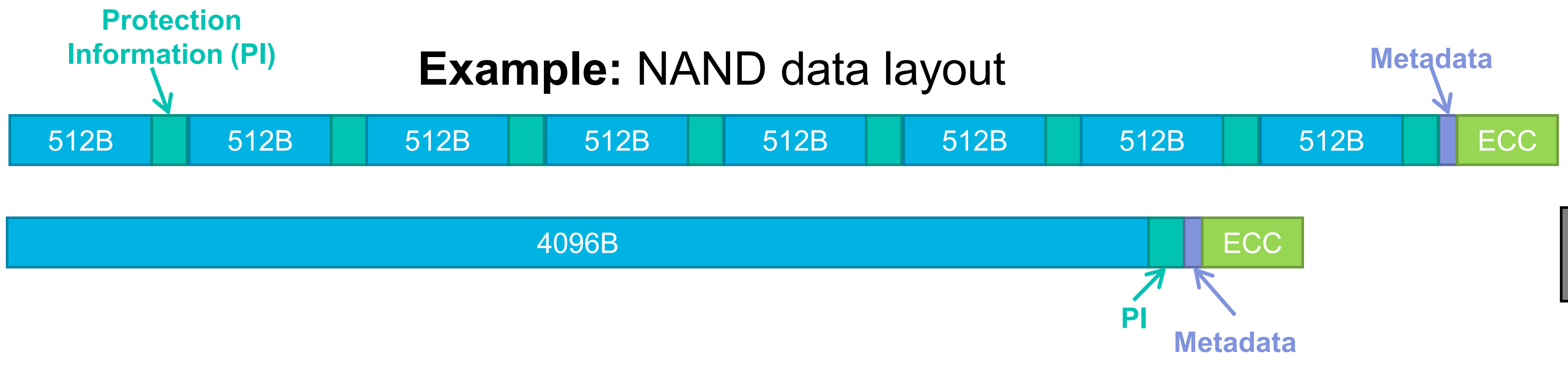
Description	512-byte sector	4096-byte sector
Gap, sync, address mark	15 bytes	
User data	512 bytes	4096 bytes
Error-correcting code	50 bytes	100 bytes
Total	577 bytes	4211 bytes
Efficiency	88.7%	97.3%

512-byte emulated device sector size

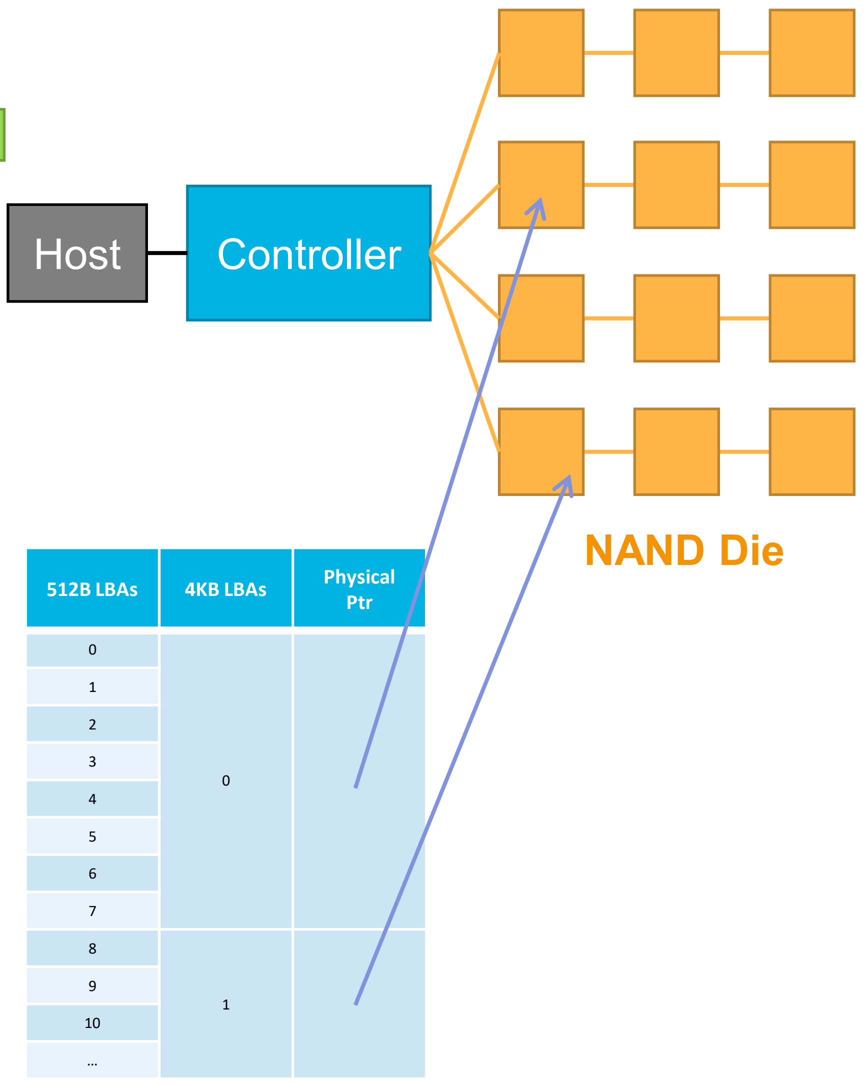
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Physical sector 1								Physical sector 2							

Note: I use Sector, Block, and LBA interchangeably in most of this presentation

Reasoning for SSD's IU



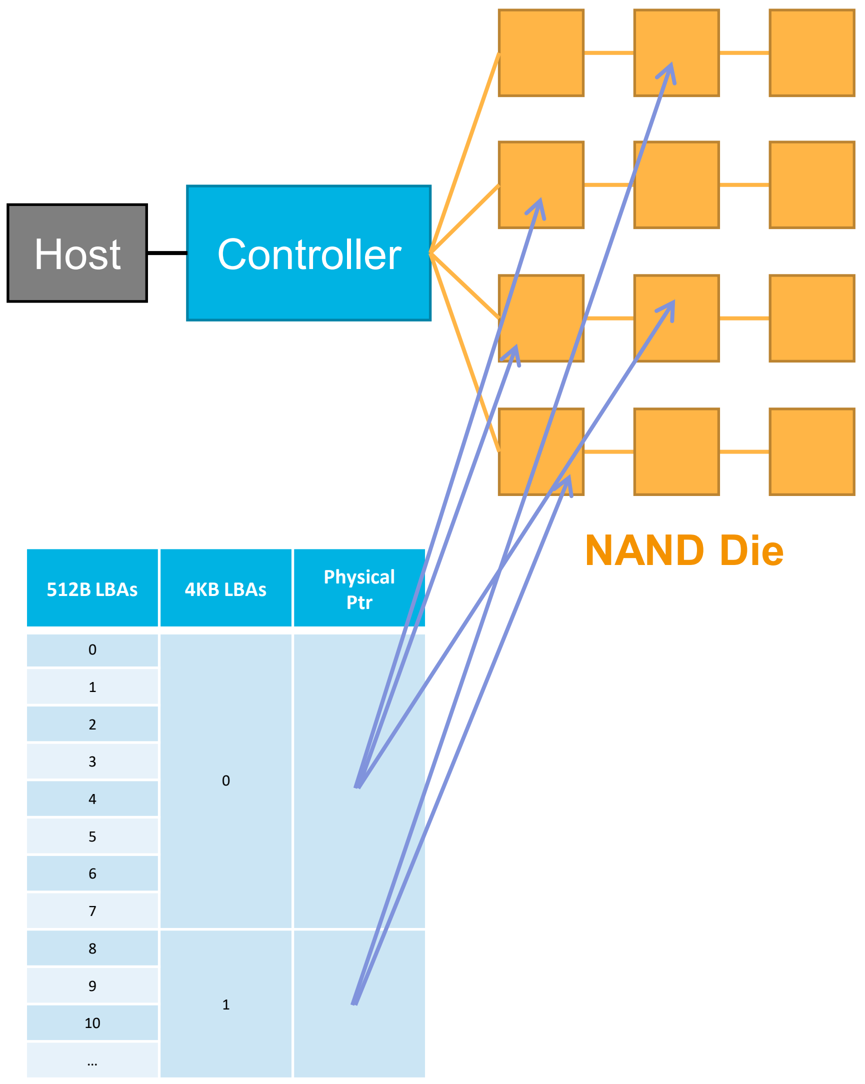
- SSDs compared to HDDs
  - Had no legacy 512B implementation
  - Knew 4KiB would eventually dominate market
  - Emulating 512B with 4KiB native was an easy decision
- Writes not aligned with the IU will incur a Read-Modify-Write (RMW) performance penalty
  - Drive must read the missing data before it can rewrite a completely contiguous IU
  - RMW path is challenging for performance optimizations
- But SSDs need to do Garbage Collection (GC)
  - Implement GC based on the Indirection Unit (IU)
  - IU = The group of data written and tracked together by an SSD
  - Emulating 512B reduces look-up table size and leverages a 4KiB data path



512B LBAs	4KB LBAs	Physical Ptr
0	0	
1		
2		
3		
4		
5		
6		
7		
8	1	
9		
10		
…		

Logical-to-Physical Table

- Logical-to-Physical (L2P) Table
  - Translates the Logical Block Address (LBA) to Physical Address
  - Required because
    - Data cannot be read and programmed back into the same place
    - Discontiguous incoming data is written physically contiguous in a log-structured manner
- L2P Look-ups
  - L2P table may be an array
    - For 4KiB LBA: Physical_Address = L2P[LBA]
    - For 512B LBA: Physical_Address = L2P[bitshift(LBA,-3)]
  - Note: This presentation ignores Namespaces.
- Garbage Collection (GC)
  - Able to move the data and update only the Physical Address of the L2P table



512B LBAs	4KB LBAs	Physical Ptr
0	0	
1		
2		
3		
4		
5		
6		
7		
8	1	
9		
10		
…		

Units

- Units are of critical importance in this presentation
  - We cross Decimal and Binary depending on NAND, DRAM, SSD capacity, LBA size, etc.
  - Various standards overlap in terminology
    - Ex: kilobyte
- This presentation uses IEC and Metric only.
  - "Memory" units are not used even though "4KB" LBA size is industry norm
- Further
  - 1b = 1 bit
  - 1B = 1 Byte = 8b
  - Capitalization will also be propagated appropriately
- Examples
  - 4KiB = 4.096kB = 4096B
  - 1GiB = 1024^3B = ~1.07GB

Multiple-byte units							
Decimal			Binary				
Value	Metric		Value	IEC		Memory	
1000	kB	kilobyte	1024	KiB	kibibyte	KB	kilobyte
1000^2	MB	megabyte	1024^2	MiB	mebibyte	MB	megabyte
1000^3	GB	gigabyte	1024^3	GiB	gibibyte	GB	gigabyte
1000^4	TB	terabyte	1024^4	TiB	tebibyte	TB	terabyte
1000^5	PB	petabyte	1024^5	PiB	pebibyte	–	
1000^6	EB	exabyte	1024^6	EiB	exbibyte	–	
1000^7	ZB	zettabyte	1024^7	ZiB	zebibyte	–	
1000^8	YB	yottabyte	1024^8	YiB	yobibyte	–	
1000^9	RB	ronnabyte	1024^9	–		–	
1000^{10}	QB	quettabyte	1024^{10}	–		–	

Physical Capacity vs Logical Capacity

- Over Provisioning (OP%) = (Physical Capacity – Logical Capacity) / (Logical Capacity) * 100%

- Historically
  - Industry used the Power of 2 to Power of 10 based systems as the OP
    - Example
      - (128Gib die) * (8 die) = 128GiB Physical Capacity
      - (128GiB - 128GB) / 128GB = (137GB-128GB)/128GB = 7% OP
  - Logical capacity was further confused by bad EBs impacting the logical capacity
    - Solved by Industry alignment and IDEMA capacity points

- Some industry standard logical capacity points with their presumed physical capacities
  - …120GB (128GiB), 240GB (256GiB), 480GB (512GiB), 960GB (1TiB), 1.92TB (2TiB), 3.84TB (4TiB), 7.68TB (8TiB), 15.36TB (16TiB), 30.72TB (32TiB), 61.44TB (64TiB), 122.88TB (128TiB), …

- 7% OP is a "Marketing" OP shorthand
  - NAND die size doesn't strictly follow Powers of 2
    - Ex: TLC NAND
  - Extra EBs per die might be used to help yield.
  - Actual OP can be an engineering decision per vendor per generation

L2P Size Consuming DRAM

- 4KiB = LBA Size = IU Size
- 960GB = SSD Nominal Capacity Point
  - "1TB" in simple terms
  - 234,421,141 LBAs per IDEMA ~= 960.2GB
- 32b per IU tracked
- DRAM consumed by L2P table
  - 234,421,141 * 32b = 937.7MB

Let's do IDEMA for extra accuracy

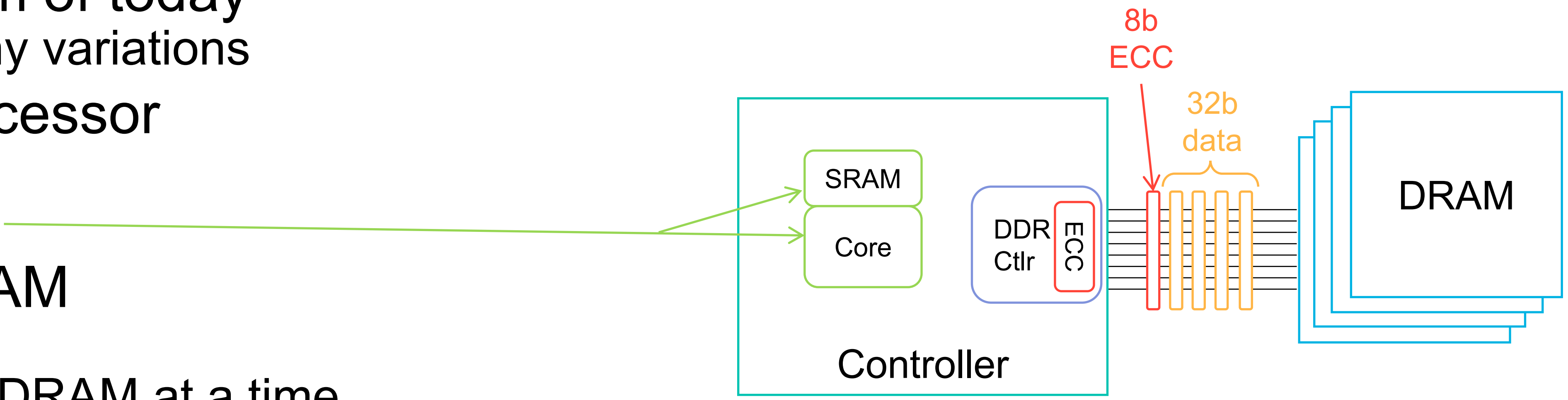
Not exactly the often quoted "1000:1"

- Real Ratio of Storage Capacity to DRAM Capacity?
  - (Size of the storage tracked)/(Size of the tracking unit)
  - 4096B/32b = 1024:1

Conclusion:
DRAM cost can be a SSD cost impactor, and it depends on IU size

Constraints of an Embedded Environment

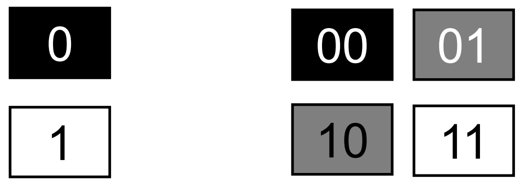
- Example system of today
  - There are many variations
- Embedded Processor
  - 32b processor
  - 32b SRAM
- Embedded DRAM
  - 8b interface
  - 40b of data to DRAM at a time
    - 4x 8b of data
    - 1x 8b of ECC
  - Yes, new LPDDR5 standards begin to solve these DRAM problems
    - But 32b interactions with DRAM and compute are still the focus for this presentation
- Contrasting with Hosts
  - 64b Addressing and processing

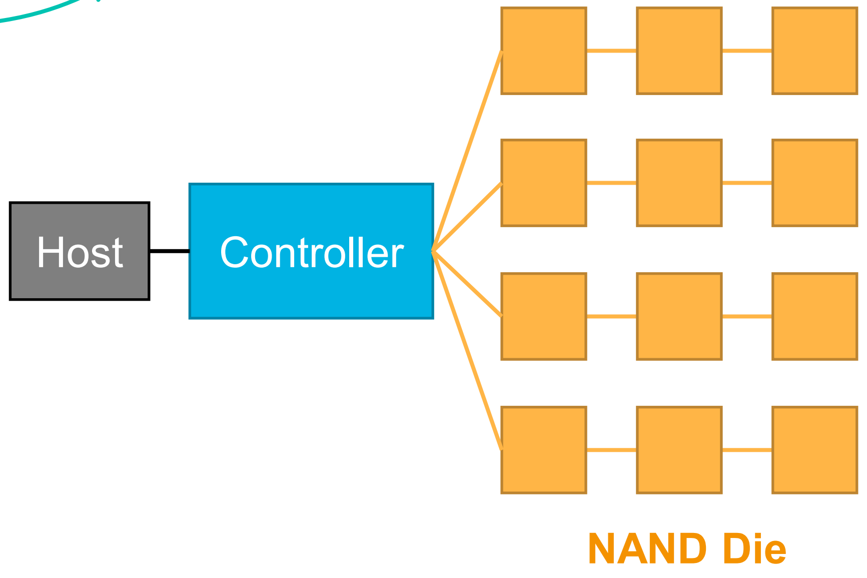


Conclusion: SSDs are a 32b world

Physical Addressing Options in SSDs

- Compact Simple Math
  - Maximum number of locations trackable
    - 1b can track 2^1=2 locations
    - 2b can track 2^2=4 locations
    - ...
    - 32b can track 2^{32} locations
  - Requires a translation to actual data location
    - 0000 0000h -> Die 0, Plane 0, EB 0, WL 0, Page 0, Sector 0
    - 0000 0001h -> Die 0, Plane 0, EB 0, WL 0, Page 0, Sector 1
    - …
    - Translation is not always simple. Ex:
      - TLC has 3 pages which is not a power of 2
      - EBs per die might not be a power of 2





- Bits Assigned a Physical Meaning
  - An Example SSD on the Right:

Assigned Meaning	Die				...	Page		Sector	
bit position	0	1	2	3	...	28	29	30	31
Value									

- Inefficiencies can result from assigned meanings
  - Representing 12 Die with 4 bits
  - Representing TLC pages with 2 bits

Conclusion: 2^{32} maximum trackable locations
Reality is less than the max

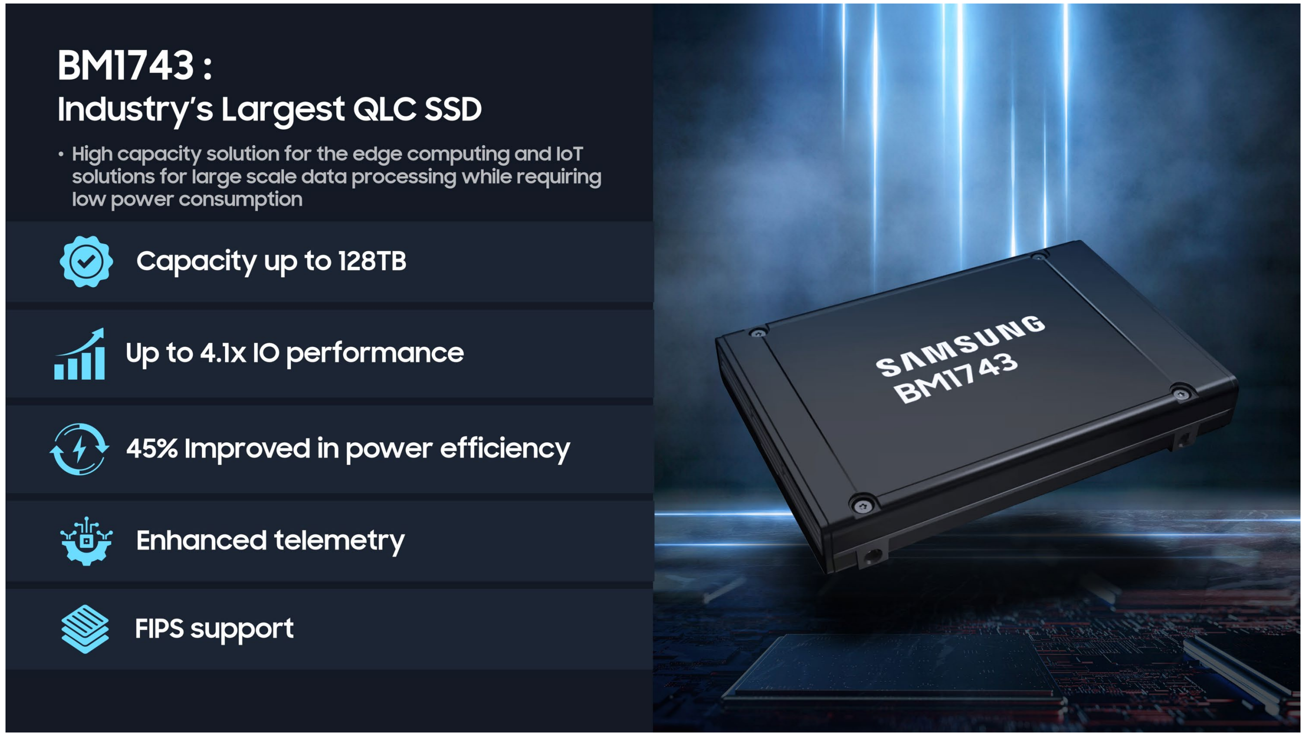
Applying Constraints to SSDs

- Maximum Trackable SSD Capacity
  - IU size = 4KiB
  - Trackable physical locations = 2^{32}
  - 4KiB * 2^{32} = 16TiB

SSDs will top out at **15.36TB** after accounting for OP

Samsung FMS Announcement

128TB



BM1743 :
Industry's Largest QLC SSD

- High capacity solution for the edge computing and IoT solutions for large scale data processing while requiring low power consumption

Capacity up to 128TB

Up to 4.1x IO performance

45% Improved in power efficiency

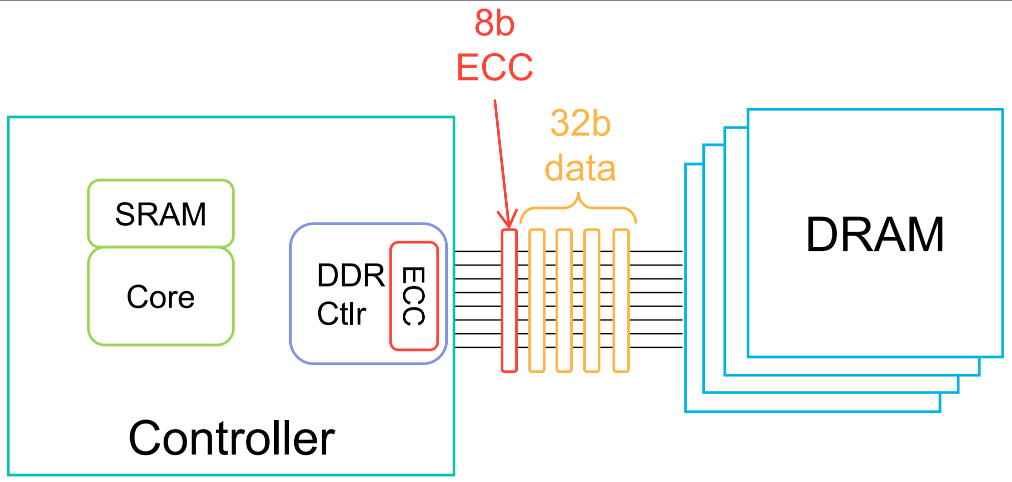
Enhanced telemetry

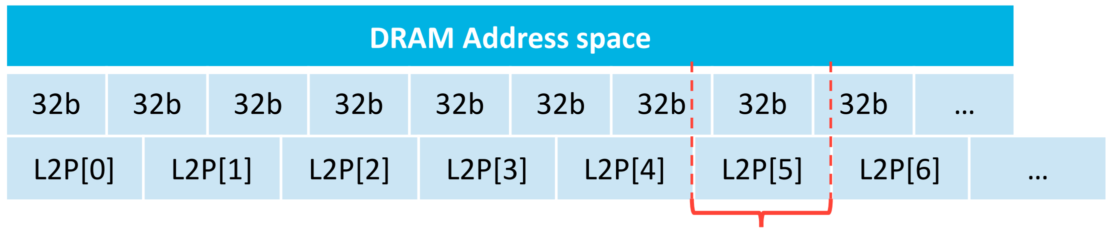
FIPS support

Solving the IU Capacity Limit

Increase Bits per L2P Entry

- Incremental bit increase for L2P Entry Size
  - 33b enables 32TiB max SSD capacities, 34b enables 64TiB, etc.
  - DRAM capacity for L2P table grows incrementally with L2P entry size
    - Example: 4096B/33b = 992:1
  - DRAM accesses are still optimized for 32b
- Compact L2P entries next to each other?
  - DRAM Read Flow
    1. Physical_Ptr_Head = Read 32b
    2. ECC(Physical_Ptr_Head)
    3. Physical_Ptr_Tail = Read 32b
    4. ECC(Physical_Ptr_Tail)
    5. L2P[N] = concatenate [Physical_Ptr_Head Physical_Ptr_Tail]
  - Compaction and shifting challenges
    - Every increase in SSD capacity requires new DRAM compaction and shifting
    - Extending this further means some L2P entries may need 3 DRAM reads if the 32b group lands in the middle of the L2P entry
- Enable 2x 32b DRAM reads?
  - 31b of unused space
  - DRAM capacity for L2P table = 4096B/64b =512:1
  - Increase L2P entry by 1 bit, but double DRAM cost
- Tradeoff and focus on 40b entries?
  - Enables up to 4PiB max SSD capacities
  - Compaction problems are simplified to focus on a common 8b offset
    - HW accelerations assisting the compactions can be more limited
  - DRAM capacity for L2P table = 4096B/40b = 819:1

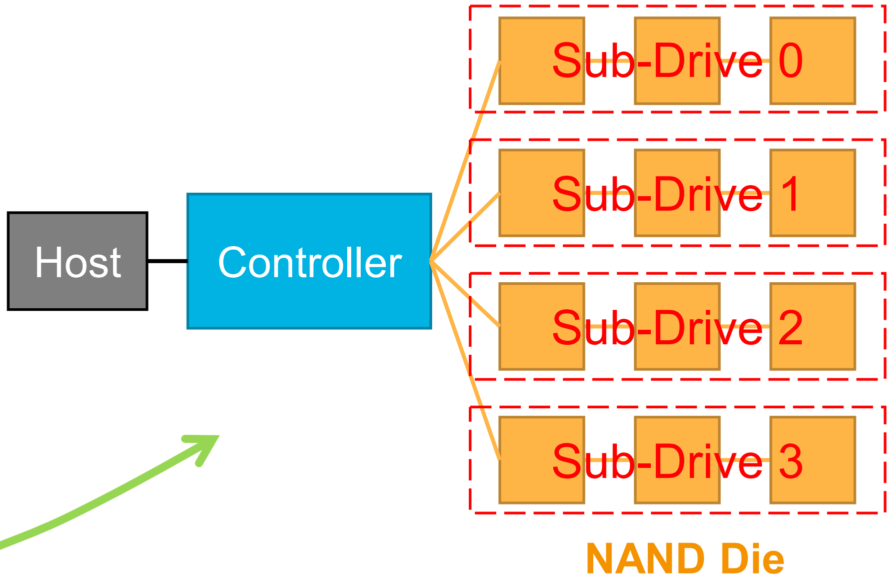




Summarized:
- Cost and Performance – Concatenation challenges in FW and DDR Controller
- Cost – DRAM size increases
- Latency – Additional DRAM reads
- Performance – Potential DRAM congestion

Wear Leveling Sub-Drives

- Stripe/Hash Data into Sub-Drives based on the LBA
  - Many striping/hashing schemes can exist.
- Implement a mini-L2P table within each Sub-Drive
- Challenge:
  - Wear Leveling/Balancing within each Sub-Drive is easy
  - Wear Leveling/Balancing between Sub-Drive
    - Balancing wear between Sub-Drive is possible with invention
    - May not be acceptable to some customers
    - But if each Sub-Drive is 16TiB of capacity, there is a significant capability to manage the wear over the life of the drive
      - Is it a realistic scenario for a user to write only to 1 Sub-Drive?
- L2P Entry Size Savings
  - LBA bits saved = X
  - Sub-Drive Count = 2^X
  - Example choices
    - Sub-Drive routing = Hash (LBA, Sub-Drive count)
    - Local_Sub_Drive_LBA = LBA>>X
    - Illustration has 4 Sub-Drives and saves 2 bits



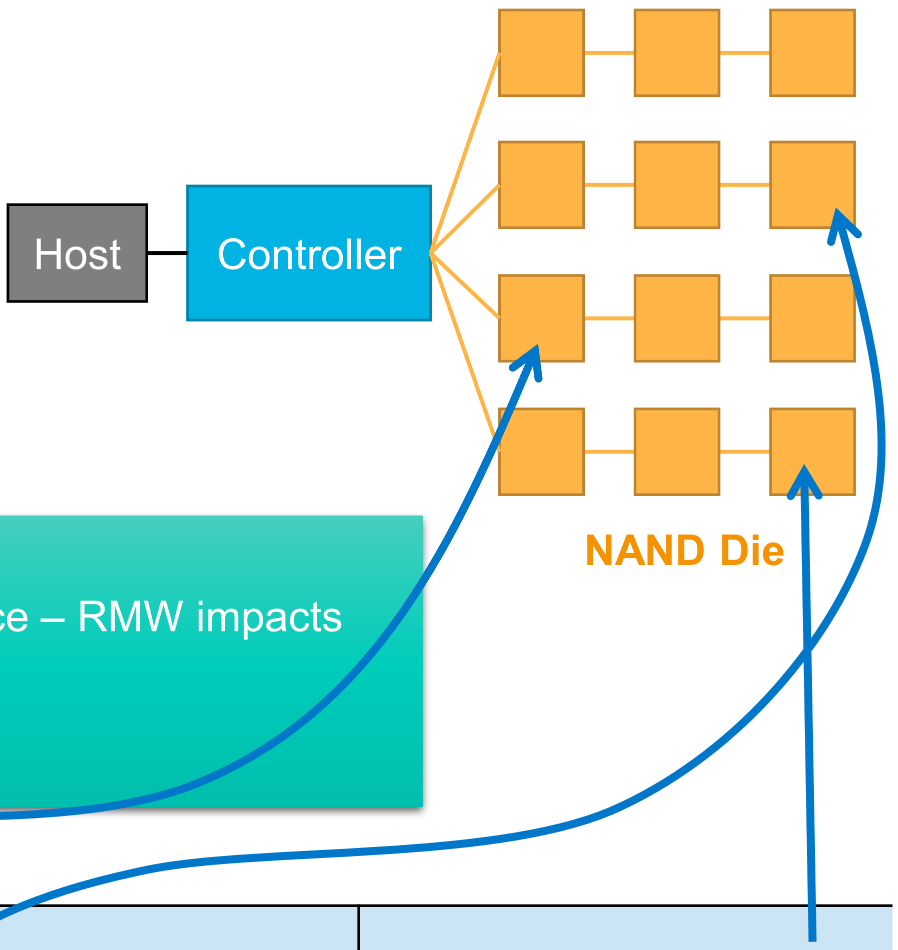
Summarized:
- Endurance – Lacks Global Wear Leveling
- Performance – Risk to lose parallelism across sub-drives
- Extensibility – Global NAND decisions can be complex or slow

Increasing IU size

- Industry standard: Group Sequential LBAs
  - Similar solution to the emulated 512B LBAs
- Advantages
  - Enables informed hosts to start shaping traffic
  - Each doubling of IU size ➔ halving of DRAM needed for L2P table
- Disadvantages
  - Misaligned Writes or Writes smaller than an IU cause RMW performance impact
- 8KiB IUs diagrammed
  - NAND:DRAM ratio = 2048:1

Summarized:

- Performance & Endurance – RMW impacts for small IOs
- Good cost improvements
- Legacy compliant



Physical Ptr																																										
4KiB LBAs	0								1								2								3								4								…	
512B LBAs	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	…

Mixing Solutions

- All 3 can be used together in any combination
  - Wear Leveling (WL) Sub-Drives and larger IUs is most reasonable discussion

- Provides a mix of benefits and disadvantages. Example:
  - Keep the WL Sub-Drives as large as possible for maximum endurance and media management
  - Increase IU only as needed to minimize RMW performance impact

Conclusions on IU Sizes

- IU Size increases are already happening
- Due to the continued increasing SSD Capacities, IU Sizes cannot be capped
  - Enterprise Customer base continues to demand
    - Low Cost
    - Optimized Endurance
    - High Performance
    - Tight Latencies
  - An "Advanced Format" effort to agree on the next IU size for the next 10 years is challenged

IU Sizes will Continue to Increase

Customer Ecosystem for Large IUs

If IU Sizes are going to continue increasing, how do we deal with this?

Matching the LBA Sector Format to the IU

For

- Precedent for 4KiB IUs matching the sector size
- System overhead reductions
  - Likely very minor efficiencies available through various components
- Potential Advantages in ECC efficiency
  - Current SSDs do not take advantage of this
  - ECC advantages are always available with IU increases

Conclusion: Let's examine other options

Against

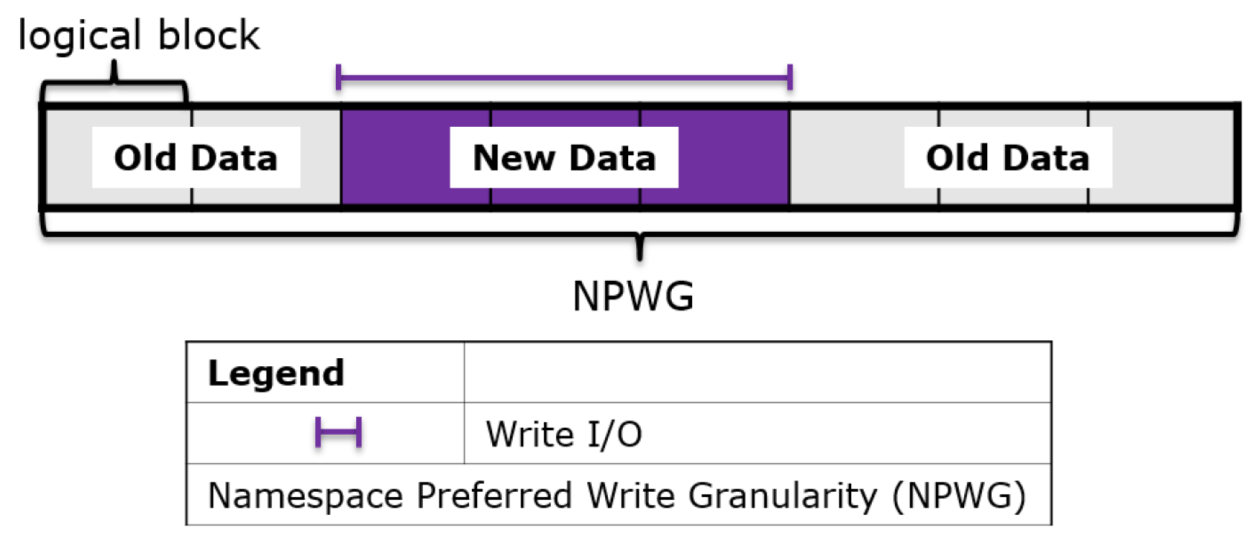
- Significant 512B usage after years of transition time
  - 4KB format intro in 1998
  - Native 4KB drives in 2010
- Supporting diverse LBA sizes is very challenging
  - IU size can vary depending on the capacity, vendor, and SSD generation
- May require new Protection Information (PI) standards at very large sector sizes
- Sector Size changes are entwined in many additional systems
  - Ex: Memory Pages
- Risk of small command performance degradations
  - Ex: QD1 4KiB Random Reads by an end application because entire SSD LBA must be read, ECC decoded, and transferred together before getting the 4KiB requested

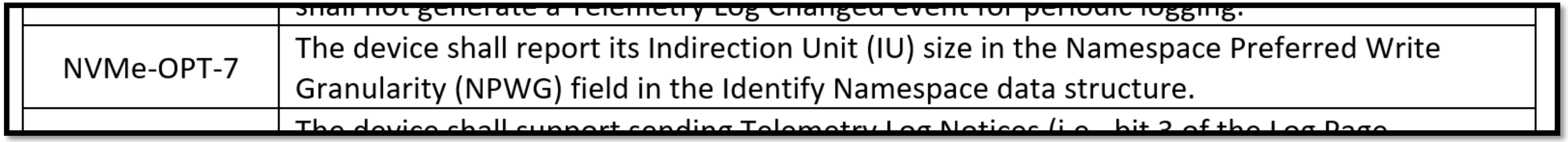
Leveraging NVMe® and OCP Specifications

- NVMe® – Optimal Performance Parameters (OPTPERF)
  - Defines the IU size to be NPWG – as clearly as a specification can define the physical parameters
  - Illustrates a 512B LBA on 4KiB IU
  - Highlights the concerns for RMW impacts

- OCP – Data Center NVMe SSD Specification
  - Can add physical definitions
  - Solves the confusion that NVMe cannot

Figure 141: Non-conformant Write Impact

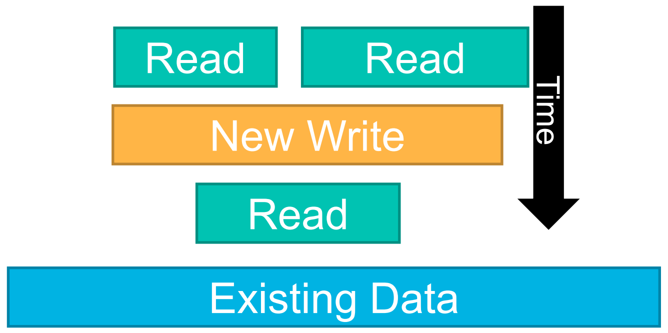


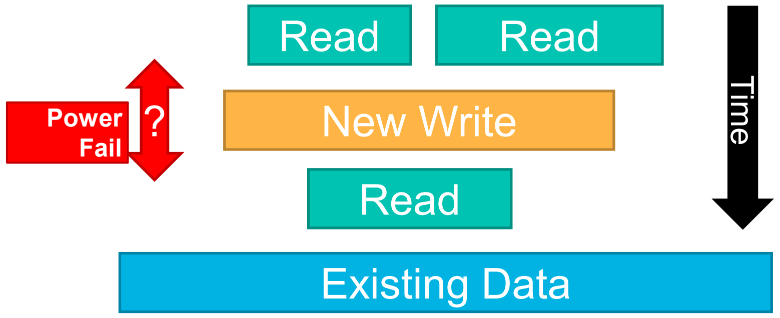


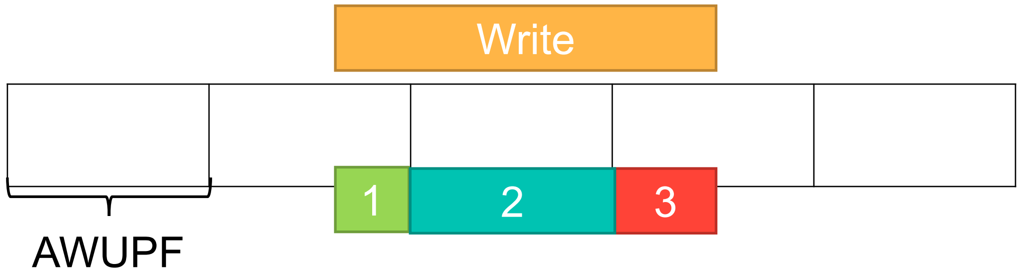
	shall not generate a Telemetry Log Changed event for periodic logging.
NVMe-OPT-7	The device shall report its Indirection Unit (IU) size in the Namespace Preferred Write Granularity (NPWG) field in the Identify Namespace data structure.
	The device shall support sending Telemetry Log Notices (i.e., bit 3 of the Log Page

NVMe® Atomics

- Atomic Write Unit Normal (AWUN) & Namespace Atomic Write Unit Normal (NAWUN)
  - Data consistency unit for commands in flight
  - Any reads in flight concurrent with a write (<=AWUN or <=NAWUN) will receive only new data or only old data
- Atomic Write Unit Power Fail (AWUPF) & Namespace Atomic Write Unit Power Fail (NAWUPF)
  - Data consistency unit across power fail
  - For writes <=AWUPF or <=NAWUPF experiencing a powerfail, the write completes in entirety or not at all.
- Multiple Atomicity Mode
  - Enables a command crossing AWUPF/NAWUPF to complete in portions
  - MAM enables potential optimizations: Data consistency unit across power fail enables a Host to check which portions of a command completed

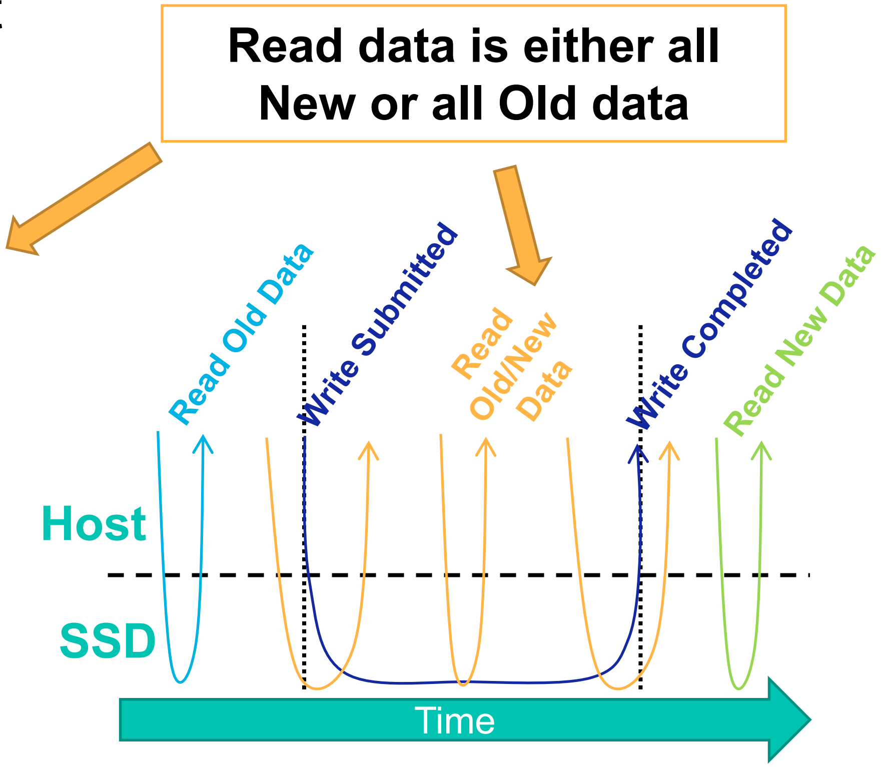






Host's usage of Atomics

- Enables system Coherence and Consistency across multiple host compute resources
  - Host must select the Coherence and Consistency models and implement appropriate rules
  - Coherence
    - Requirements
      - Write Propagation: Changes to data in cache are propagated fully and completely through SSD
      - Transaction Serialization: Writes must be seen by all processors in the same order
    - Due to race conditions and multi-threading in SSD, Host must barrier around the Write to close this gap
  - Consistency
    - Requires
      - All processors are consistent in the order they see a write occur
    - Host can place boundaries on behaviors around Write to enforce Consistency
- Coherence and Consistency Requirements
  - Eventual consistency database models have significantly relaxed requirements on many parts of stored data
  - However, File System and Database metadata are often much stricter.
    - Ex: FS progress within an enclosure must be power fail safe



Conclusion:
- AWUPF enables increasing the Write size of the minimum allowed IOs that are Coherent and Consistent
- **AWUPF>=NPWG=IU Size** means all Host IOs can now remove RMW exposure

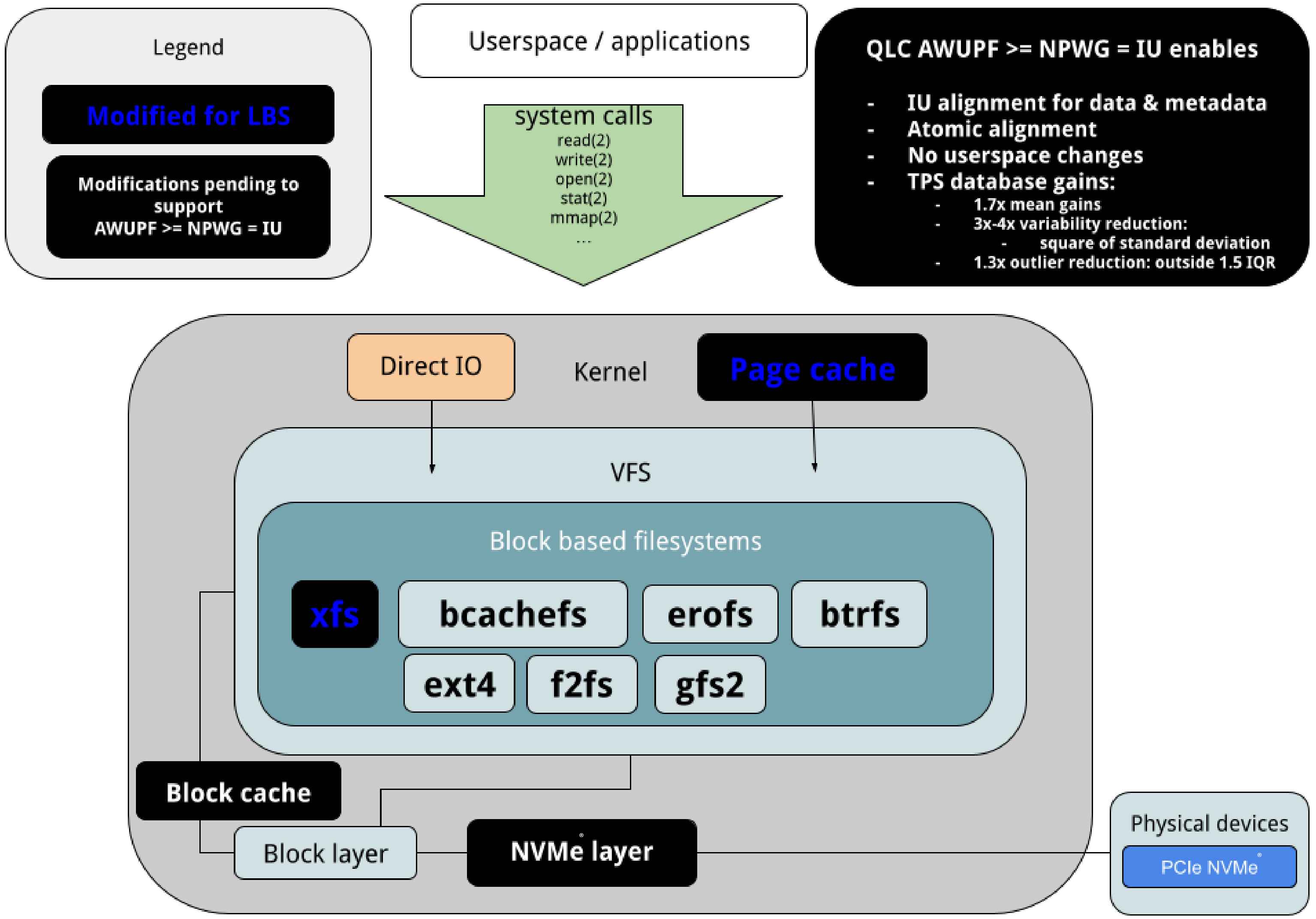
Enabling Large Block Sizes (LBS) in Host OS

- Samsung GOST has been leading a Large Block Size (LBS) effort in Linux
  - Evaluate the OS changes that best facilitate QLC and large Indirection Unit (IU) SSDs
  - Quantify SW impacts of each option and potential gains for Host and SSD
  - Propagate those changes through all layers

- LBS conclusions
  - Increasing LBA Sector Size is difficult
  - Putting new requirements on Atomic Power Fail is the best solution for enabling large IUs
    - AWUPF >= NPWG = IU
    - Minimal ecosystem impacts with maximized advantages
    - No backward compatibility concerns
  - Multiple Atomicity Mode might be interesting for some customers to evaluate, but has not helped current SW changes.

How LBS helps Large IUs

- All Host SW can pick up these advantages "for free"
  - Changes are encapsulated (Block layer, Page Cache, File System, etc).
- I/O alignment determinism is an option
  - Opt-in to using the sector size
  - Aligns Host I/O with atomics
- Backward compatibility is maintained



How LBS helps with IU Alignment: blkalgn

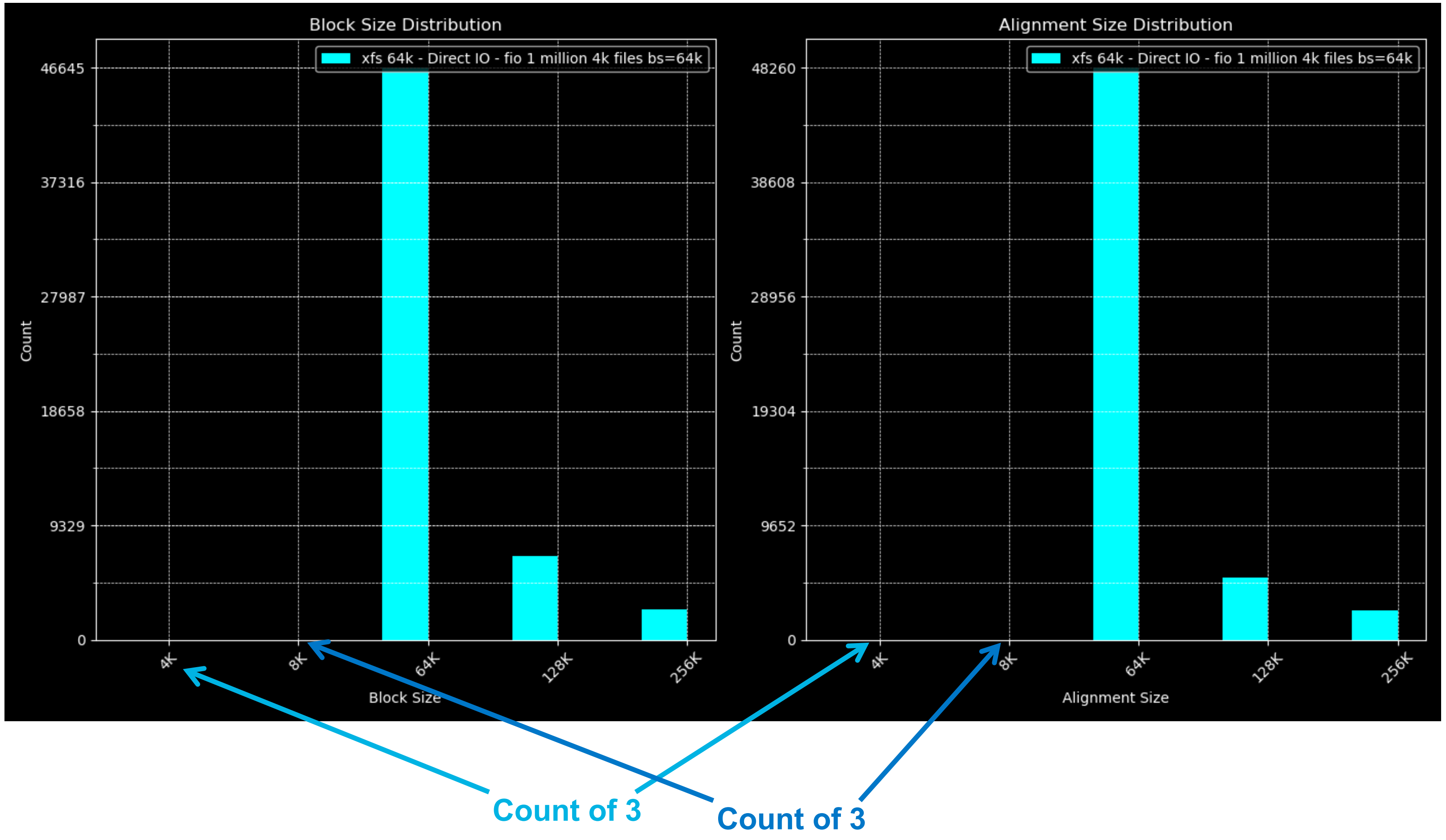
Workload	IU	FS	fs block size	fs sector size	Write type	LBS	Worst Case Workload WAF
1 million 4 KiB files fio bs=512	16 KiB	XFS	4 KiB	4 KiB	Buffered I/O	No	1.6802359343
	64 KiB	XFS	4 KiB	4 KiB	Buffered I/O	No	5.4081575535
	64 KiB	XFS	64 KiB	4 KiB	Buffered I/O	Yes	1.0000008551
	16 KiB	XFS	**Plotted on next slide**		fered I/O	Yes	1.0
	64 KiB	XFS	64 KiB	4 KiB	Direct I/O	Yes	1.0000098099
	64 KiB	Btrfs on XFS	4 KiB on 64KiB	4 KiB	Direct I/O	Layered	1.0001400657
MySQL 16 KiB db page size 12 hour sysbench	16 KiB	EXT4	4 KiB	4 KiB 16 KiB cluster	Direct I/O	No	1.0121850379
	16 KiB	XFS	16k KiB	4 KiB	Direct I/O	Yes	1.0000000220
MySQL 64 KiB db page size 12 hour sysbench	64 KiB	XFS	64 KiB	4 KiB	Direct I/O	Yes	1.0000023285
	64 KiB	XFS	64 KiB	4 KiB	Direct I/O	Yes + Block folios	1.0000019071

```
> wget https://raw.githubusercontent.com/dkruces/bcc/lbs/tools/blkalgn.py -O /usr/local/bin/blkalgn
> chmod 755 /usr/local/bin/blkalgn
> apt-get install python3-bpfcc python-is-python linux-headers-$(uname -m)
> blkalgn --disk nvme0n1 --ops Write --json-output example.json
```

LBS Results Visualized

- FIO doing 512B Writes
- IU = 64KiB
- FS = XFS with 64KiB block size
- SSD and FS = 4KiB Sector Size
- LBS enabled
- WAF = 1.0000098099

- Counts of 3 for
  - 4KiB sized
  - 8KiB sized
  - 4KiB aligned
  - 8KiB aligned
  - *Perhaps they're the same 6 I/Os*



A Final Comment: Why the inequality on AWUPF?

- Recommending: AWUPF**>=**NPWG=IU size

- The “Greater than or Equal to” is needed to allow more design freedom for vendors
  - Often AWUPF is designed into the HW of the SSD’s Controller
  - IU Size can vary with capacity while the Controller is constant

- Independent of the IU size, Hosts are free to increase interaction sizes all the way up to AWUPF
  - Optimizations possible for hosts to move up to Minimum(AWUPF, NPWG) if the smaller IO sizes are deemed beneficial when AWUPF exceeds NPWG.

Conclusions

- IU Sizes will continue to increase
- **Recommended** Customer Requirements to Optimize Performance of Large IU SSDs
  - AWUPF>=NPWG=IU
  - Optional suggestion of MAM=1
- Impacts
  - Backwards Compatible with legacy SW and legacy SSDs
  - LBS has been part of the Linux-next development tree since the end of August, and is expected to be part of the upcoming v6.12 Linux kernel release in Q4 2024
- Come learn more on putting these learnings into practice!
  - Hyung Seuk Kim is presenting “Impact of High Capacity SSDs and QLC on Storage System - Issues to be Resolved” on Wednesday at 3:35PM