Overprovisioning in All-Flash Arrays

Bill Radke
Skyera, Inc

SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Abstract

- Overprovisioning in All-Flash Arrays
  - This session will appeal to Data Center Managers, Development Managers, and those that are seeking a fundamental understanding of the performance and capacity tradeoffs involved in All-Flash arrays.
  - The session will look at how NAND operate, how overprovisioning effects that, and what optimal OP for different applications might be.
  - The presentation is intended to be high-level and is not intended for design engineers, but rather provides a overview of system operations appropriate for IT professionals.

Overprovisioning in All-Flash Arrays

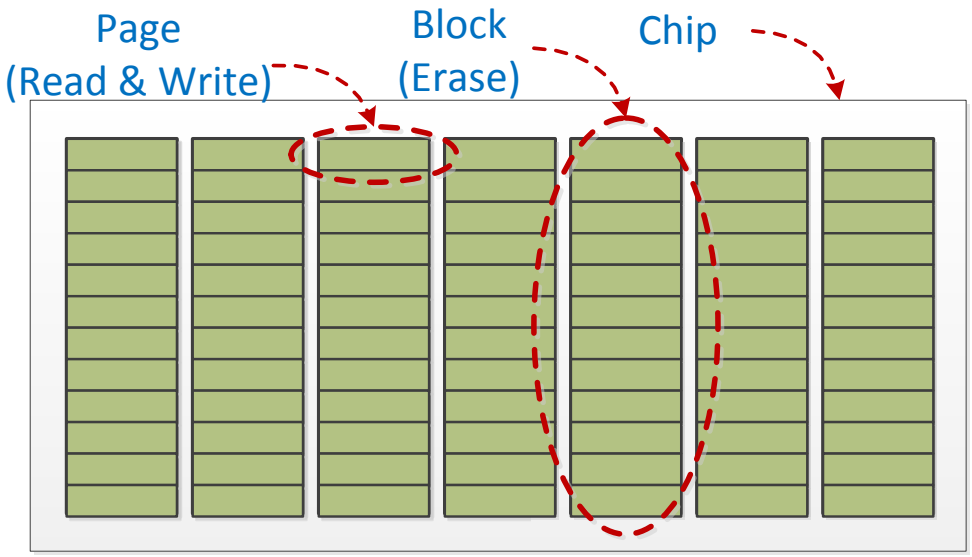
- NAND and Overprovisioning
  - NAND interfaces
  - Flash Translation Layer (FTL)
  - Block and Table Based Management
  - OP and Capacity
- OP's Effect on Performance
  - Garbage Collection and Write Amplification
  - Optimal OP
- All-Flash Arrays
  - AFA and Hybrid Arrays
  - Applications

NAND and Overprovisioning

- NAND are program/erase devices with read/write interfaces
  - SATA, SAS, PCIe, FC, IB, MMC are all read/write
- FTL's convert from one set of commands to another
  - FTL's manage conversion on block or page basis
- Overprovisioning makes that possible
  - More OP makes it more efficient

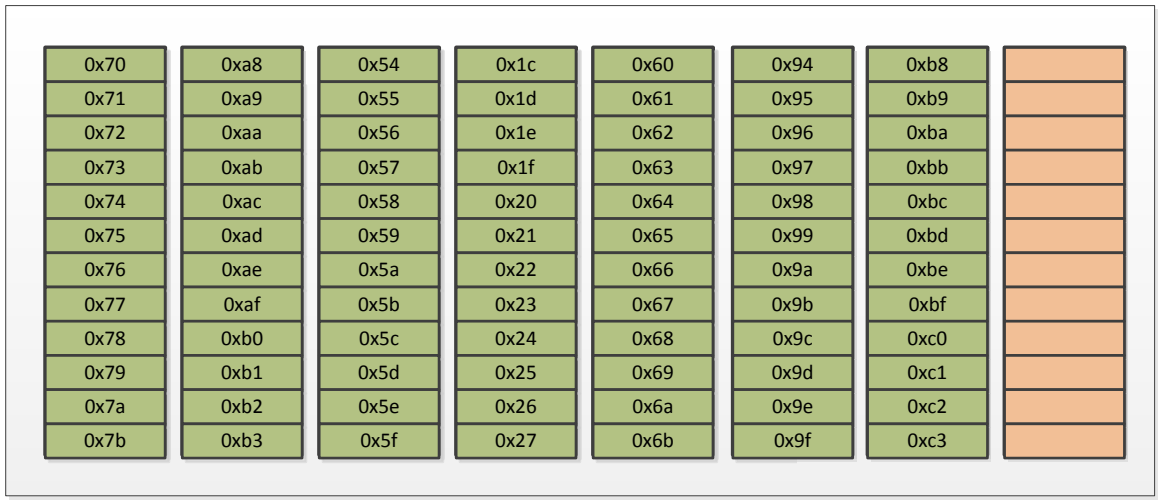
NAND and Overprovisioning: Program/Erase Devices

- Cannot overwrite data in place
- New data is written to empty blocks
- FTL notes invalidation of older entries



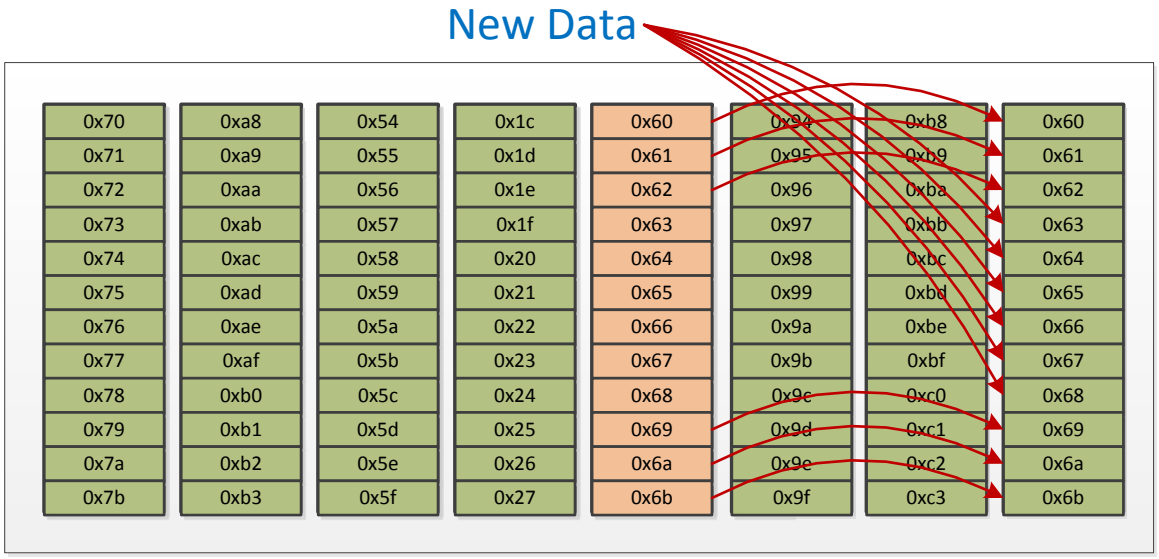
NAND and Overprovisioning: Block-based Management

- NAND can be managed on a block basis
  - Each block contains a region of addresses
  - Any update to a given block would cause the entire block to be rewritten
- Most useful if writes are at least the size of a block
  - Pictures and video, for example



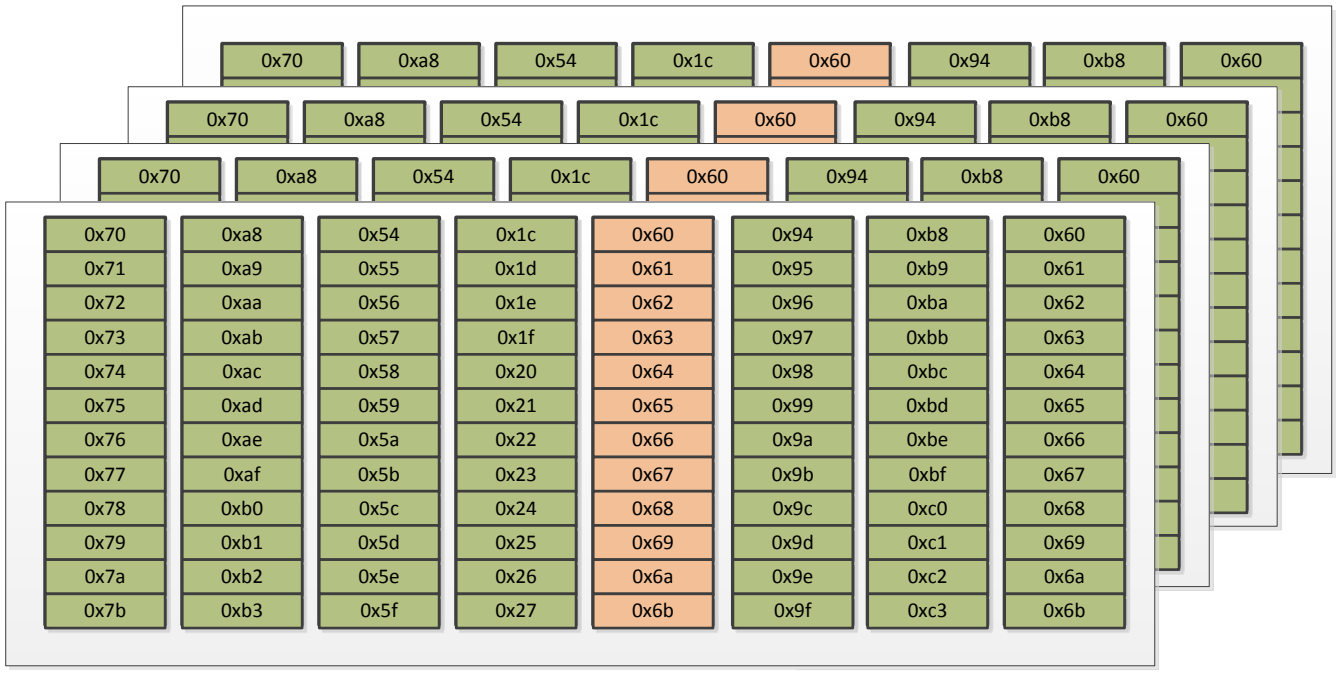
NAND and Overprovisioning: Block-based Management

- New and old data may have to be merged
  - Old data from old block, new data to be written
- Only requires 1 spare block for OP
- In this case, an equal amount of copying and writing



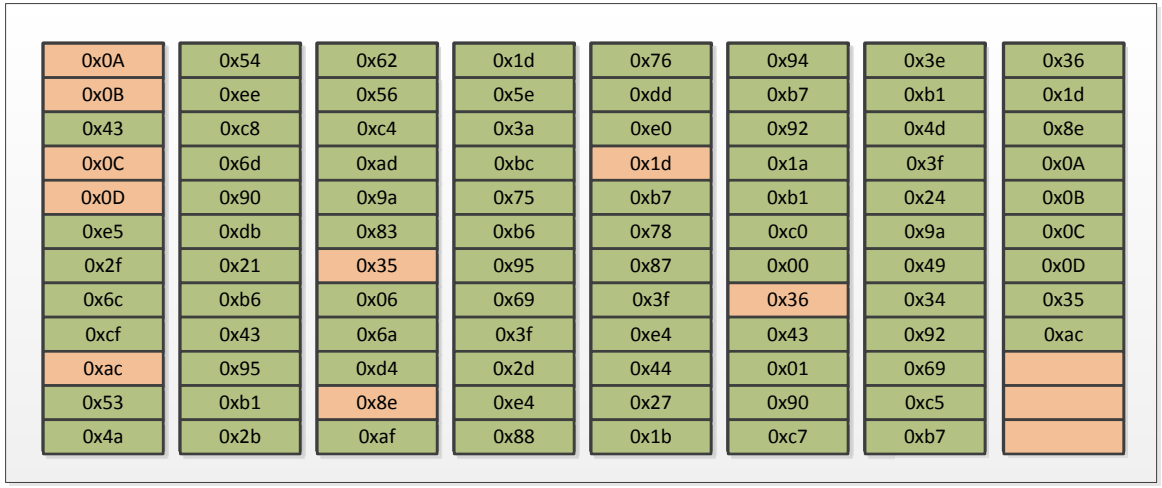
NAND and Overprovisioning: Block-based Management

- Becomes less efficient as data size becomes smaller
- Becomes less efficient as blocks become bigger
  - Grouping Flash together has higher performance, and higher effective block size
- All-Flash Arrays cannot use Block-based Management



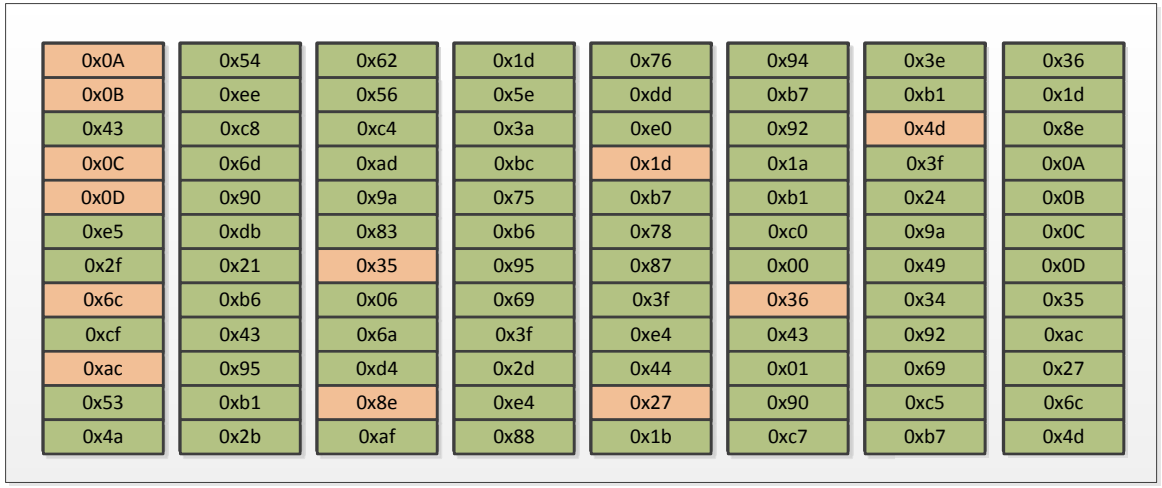
NAND and Overprovisioning: Page-based Management

- Each page of user data is tracked by the FTL
- A single block may contain many different files
  - Usually written at the same time
- New data writes invalidate older data
  - Invalid data shown in orange: some locations unwritten



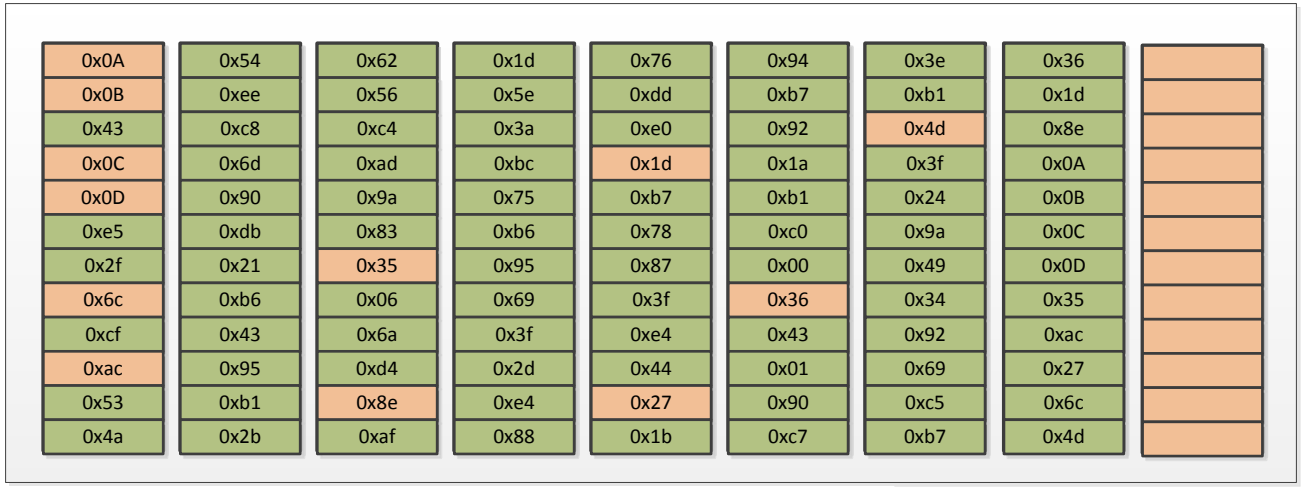
NAND and Overprovisioning: Garbage Collection

- As data is overwritten or trimmed, blocks become empty
- When a block is needed, the remaining data from the most-empty block is copied
- That block can then be erased and reused



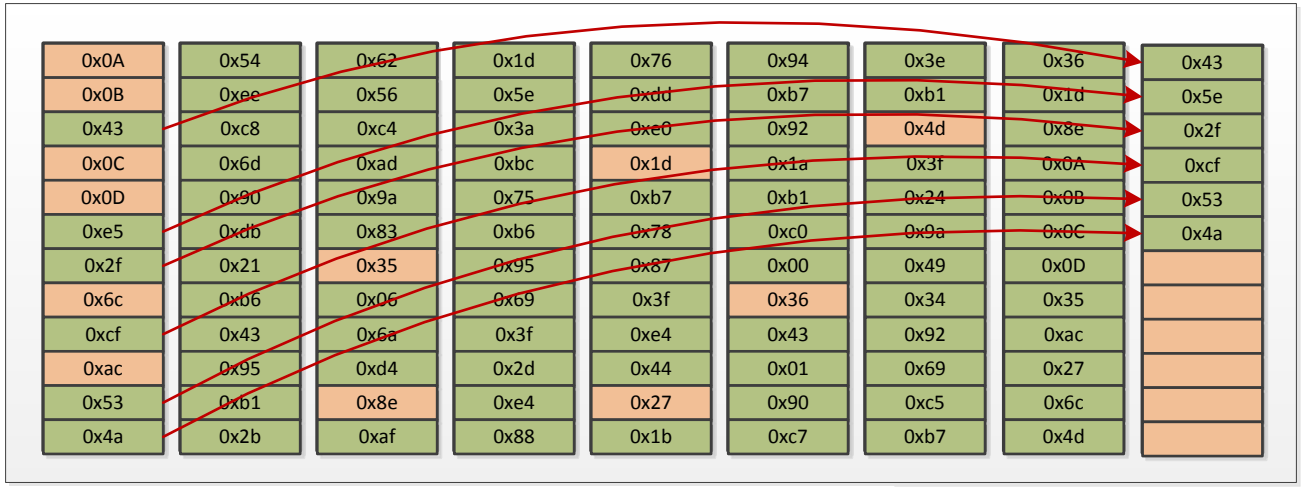
NAND and Overprovisioning: Garbage Collection

- When the data region is full, another block must be found
  - Kept in reserve during the write operations
- Orange blocks are counted as Overprovisioning
  - Both unused blocks, and portion of blocks with invalid data



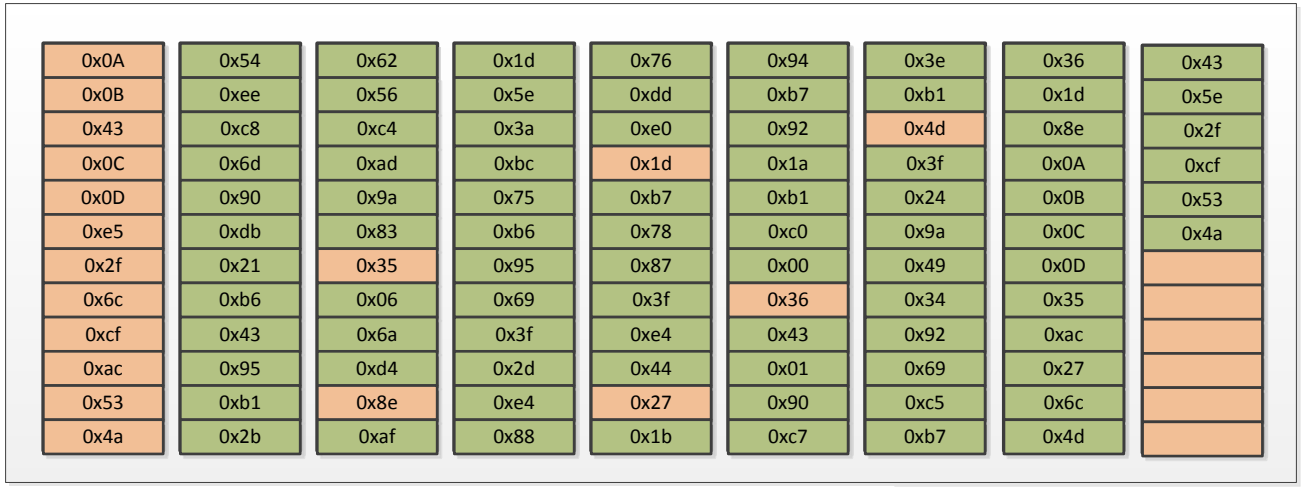
NAND and Overprovisioning: Garbage Collection

- Remaining valid data from least-valid block is copied to the previously empty block
  - This process leads to security issues
- FTL must update the location of the valid copy



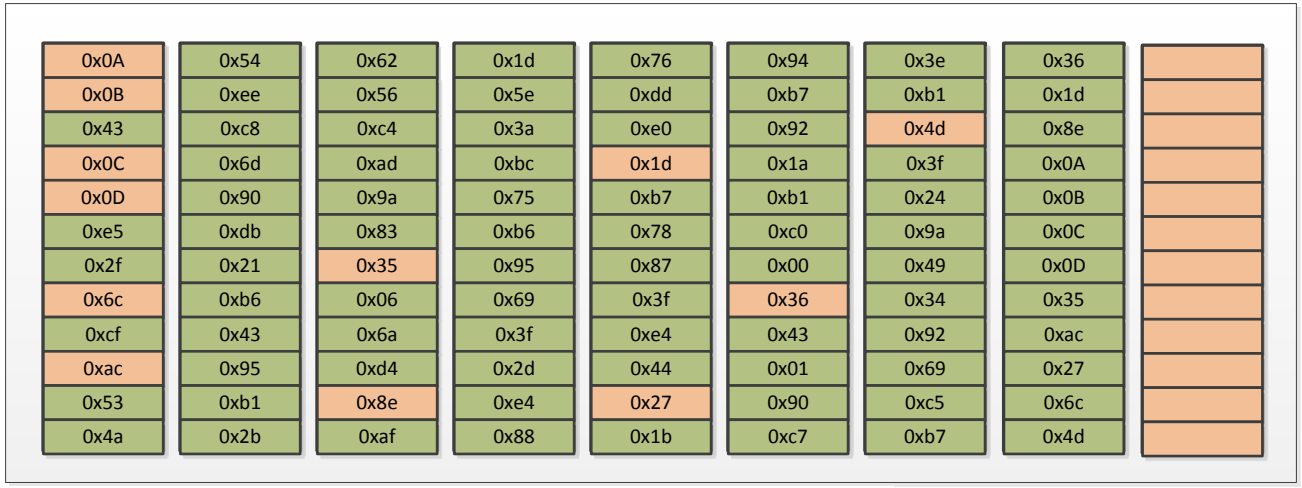
NAND and Overprovisioning: Garbage Collection

- The least-valid block can now be erased
- After GC, we have 1 ½ empty blocks
  - Started GC with 1 empty block
  - Using two blocks of OP



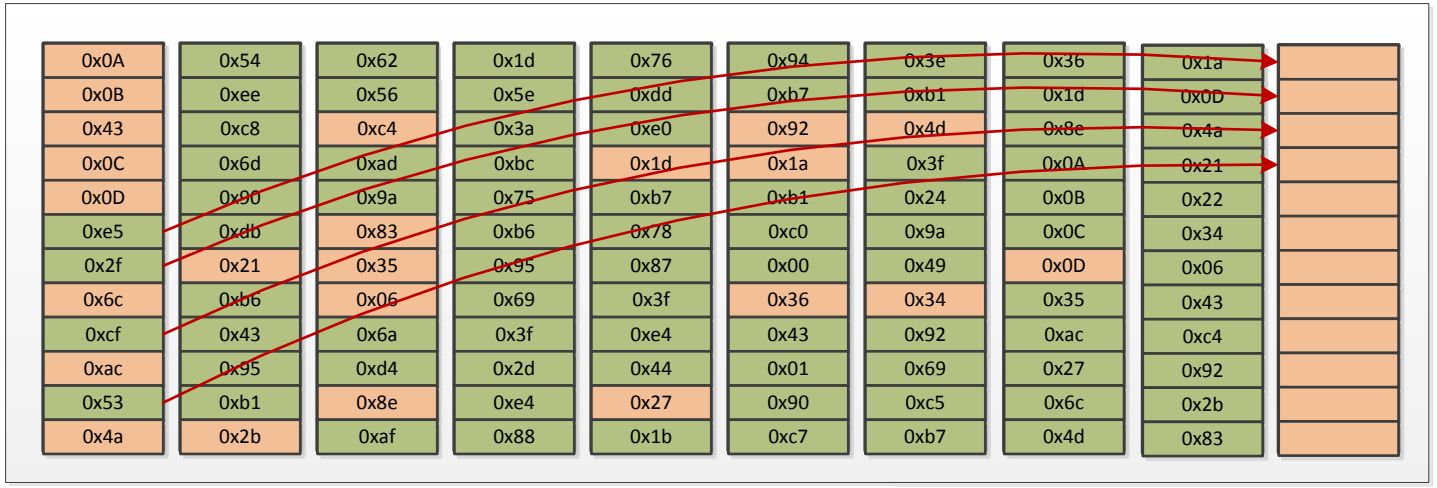
NAND and Overprovisioning: GC Overhead

- If more blocks are available, garbage collection may be delayed
- Instead of copying into empty block, allow host to write data to it
- Decreases the amount of valid data in the least-valid block



NAND and Overprovisioning: GC Overhead

- After one more block is written, less copying is required
  - There are 1 2/3 free blocks after GC
- Adding another is likely to decrease copying further
- Generally, more OP means less GC



NAND and Overprovisioning: Capacity

- Extra space is the difference between NAND and user capacity
- This space comes from a number of sources
  - NAND page and die sizes slightly larger than Po2
  - Difference between base-2 and base-10
  - Extra chips or blades
  - Lowering the stated capacity
- Extra space is consumed by ECC, bad blocks, RAID, FW, tables, and OP
  - Several layers of CRC, ECC, and RAID may be present
  - Space can be reallocated unexpectedly

NAND and Overprovisioning: Capacity

- State capacity could mean one of several things
  - User capacity, with or without compression
  - User capacity and OP, usually if OP is configurable
  - NAND capacity, requiring understanding of ECC overheads
- 100 TB of “raw” capacity
  - After ECC (5%), RAID-5 (3%), RAID-5 (12%), Overprovisioning (30%), and FW & Tables (5%): 60 TB of formatted capacity
- 100 TB of “unformatted” capacity
  - After Overprovisioning (20-40%) and FW & Tables: 79-68 TB of formatted capacity
- 100 TB of “compressed” capacity
  - Must remove compression factor from capacity: 20-50 TB of formatted capacity

NAND and Overprovisioning: Capacity

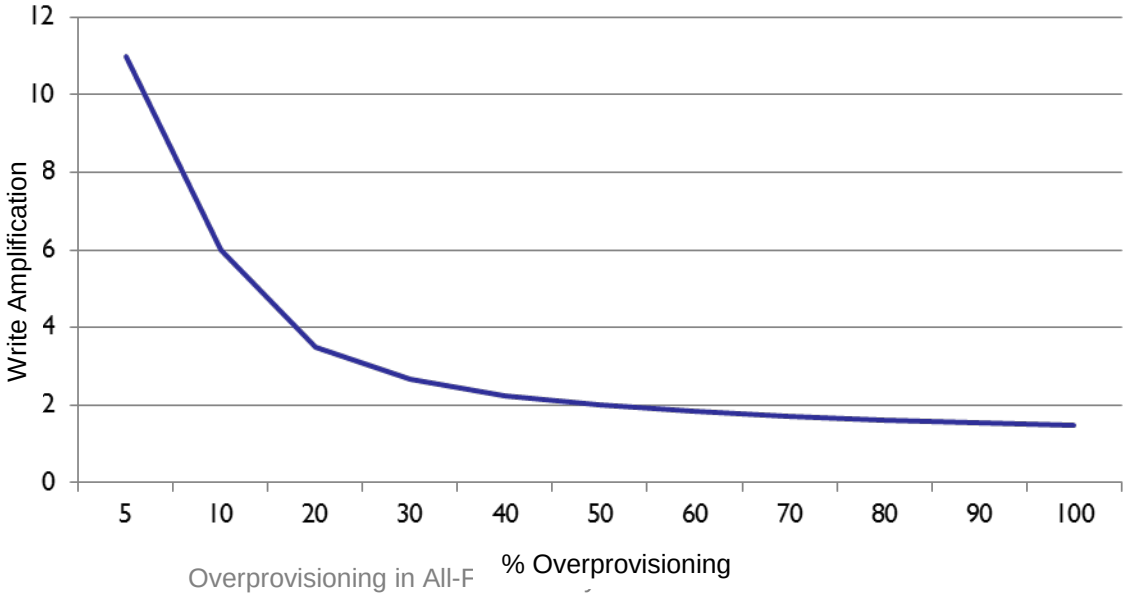
- Device vendors are paid for the user capacity
  - Previous values all estimates: your mileage may vary
  - Memory dedicated to other purposes should be hidden
  - If not, cost must be recomputed
- Goal is to maintain performance with minimal cost
  - Need to meet MB/s goal
  - Need to meet product lifetime goal
- OP is expensive

OP's Effect on Performance

- Write performance is dependent upon amount of Garbage Collection
  - With 10% OP, a system may copy 5B of GC data for every byte of write
  - This corresponds to a Write Amplification of 6
- Large Write Amp can cause endurance problems
  - NAND can be worn down by extra writes
- More OP decreases Garbage Collection writes
  - Expensive way to increase write rate
  - Other factors also influence GC
- OP does not effect read performance

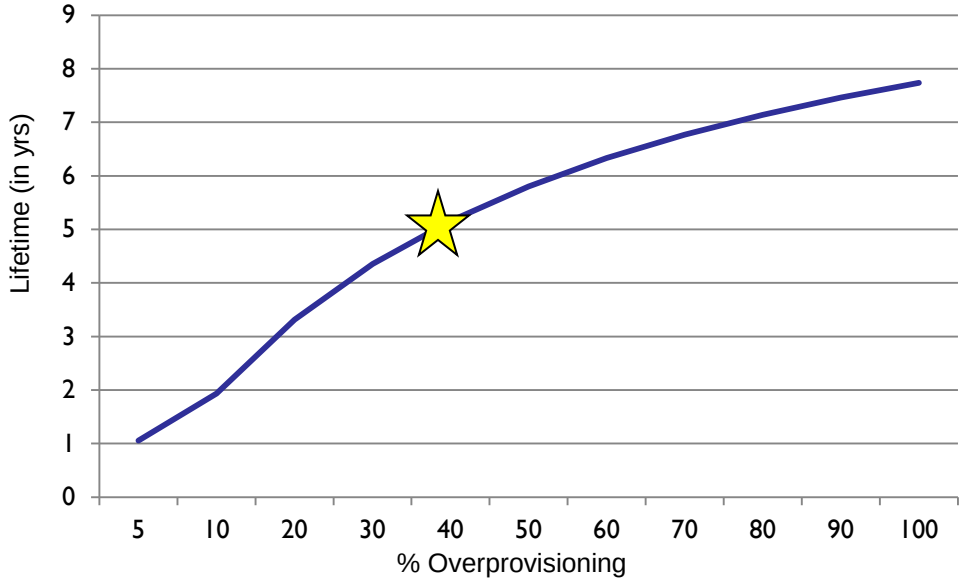
OP's Effect on Performance: Write Amplification

- Until about 50% OP, most writes are caused by Garbage Collection
  - With small OP (5%-10%), 90% of writes can be GC
- These are worst-case benchmark results
  - Small-sized, random writes



OP's Effect on Performance: Lifetime

- A flash can be cycled to death without careful system management
  - Graphs shows an example 1MIOPS system with 50 TB capacity using high-endurance NAND
- A five-year life would require 40% OP

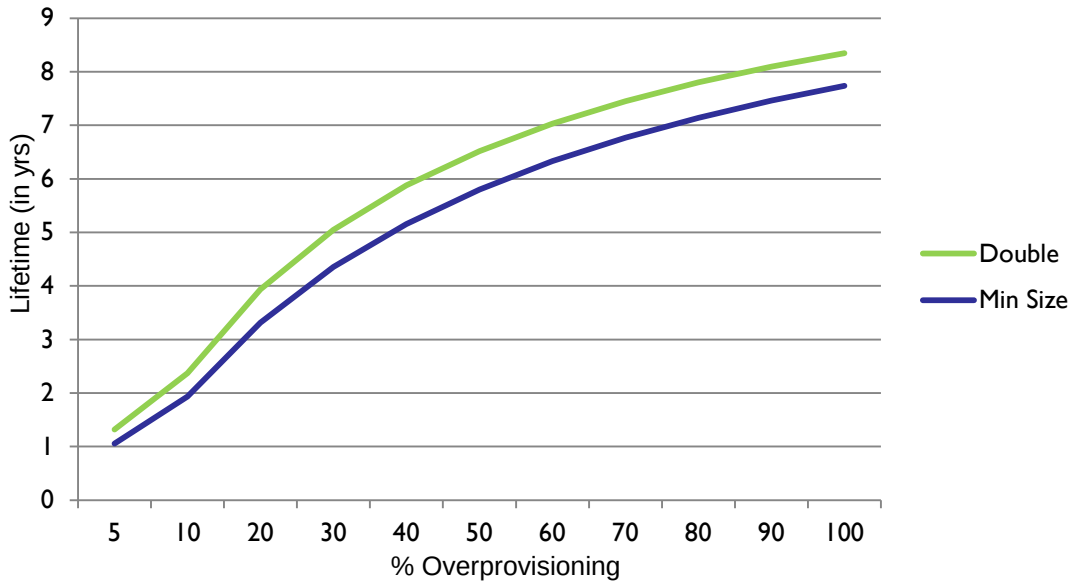


OP's Effect on Performance: Data Size

- Larger data sizes make page-based GC more efficient
  - Increases the chances of an entire block being invalid
- Many applications use larger than minimal data sizes
  - Most operations are small
  - Most data is transferred during large operations
- The toughest case is isolated, page-sized operations
  - Random operations over a limited range are similar to large blocks

OP's Effect on Performance: Data Size

- Larger data sizes make page-based GC more efficient
  - Increases the chances of an entire block being invalid
- Optimal OP for 5 year life decreases from 40% to 30%
  - Relatively small decrease in data size

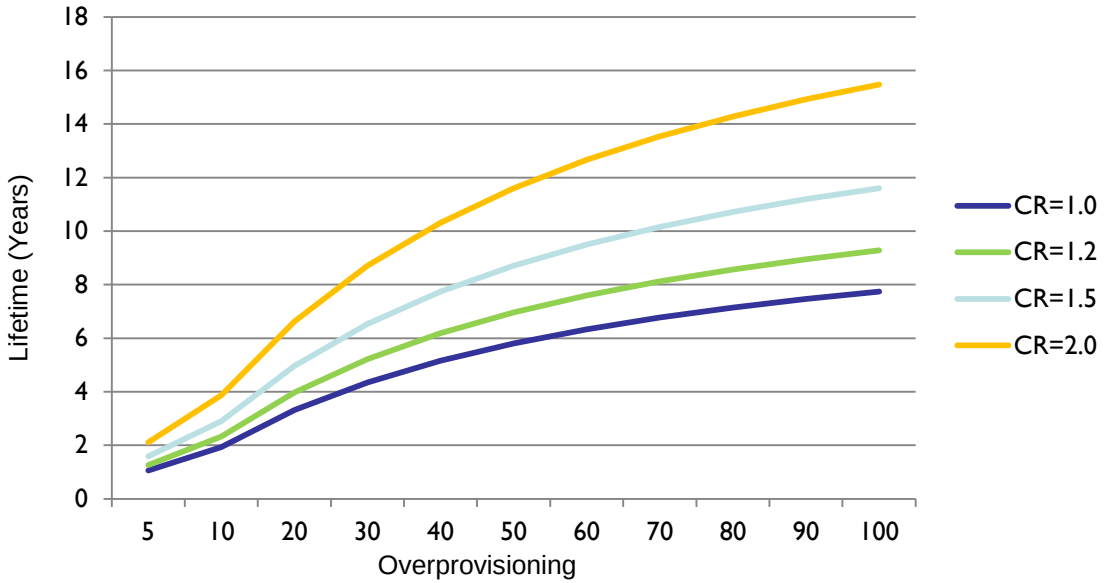


OP's Effect on Performance: Compression & Dedup

- Decreases the amount of written user data
  - Correspondingly decreases the GC
- Radically increases the system complexity
  - Hardware required for comp, dedup
  - Firmware to track byte locations, sizes
  - Lots of nasty edge cases for both
- Effectively increases the capacity of the device
  - Depending upon the compression ratio

OP's Effect on Performance: Compression & Dedup

- Decreased NAND traffic from Compression & Dedup decreases cycling and wear
  - Our hypothetical 40% goes to 15%
- Dangerous to base OP reduction on compression
  - Overprovisioning usually based on worst-case expected traffic

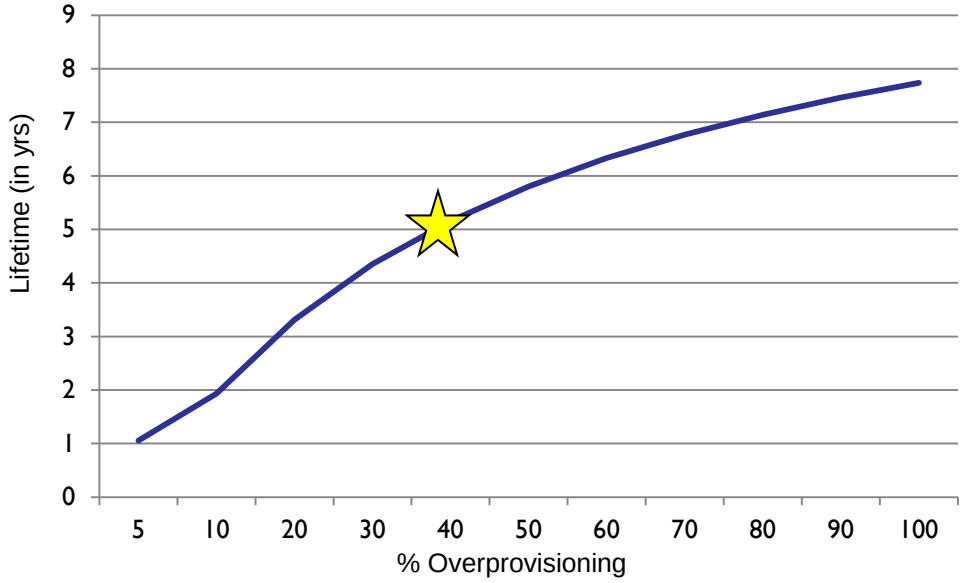


OP's Effect on Performance: Data Organization

- GC cost can be significantly decreased if data is not random
  - Many benchmarks are random
- If data which is invalidated together is grouped, less copying is needed
- Many heuristics exist to improve performance
  - Based on traffic patterns of applications

OP's Effect on Performance

- For a given lifetime or performance, certain OP is necessary
  - That value can be based on worst-case scenario
- The necessary value can be informed by average transfer size, compression and compressibility, and organization

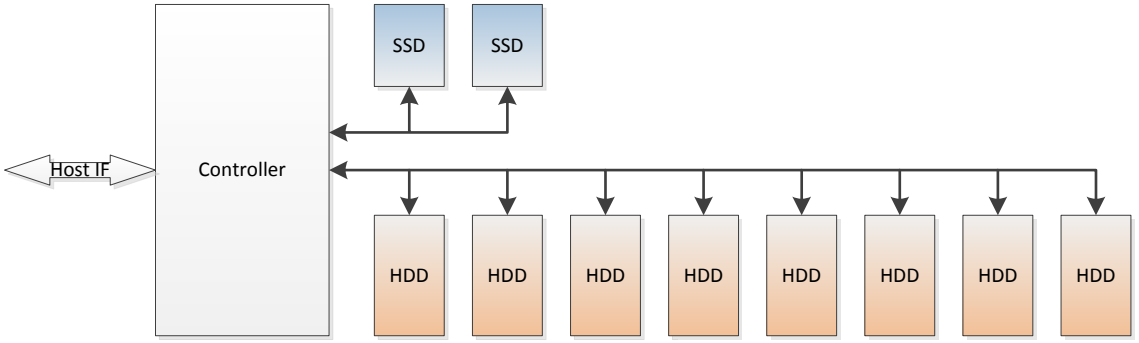


All-Flash Arrays

- Traditionally Enterprise-SSD's have been used for caches in hybrid system
  - Caches are very high traffic
  - Traffic is filtered before caching
- AFA are structurally different from Hybrid-Caching systems
  - Allowing for higher performance and density at the same cost
- All-Flash Arrays see more of the traffic
  - There is (almost) always patterns which can be exploited

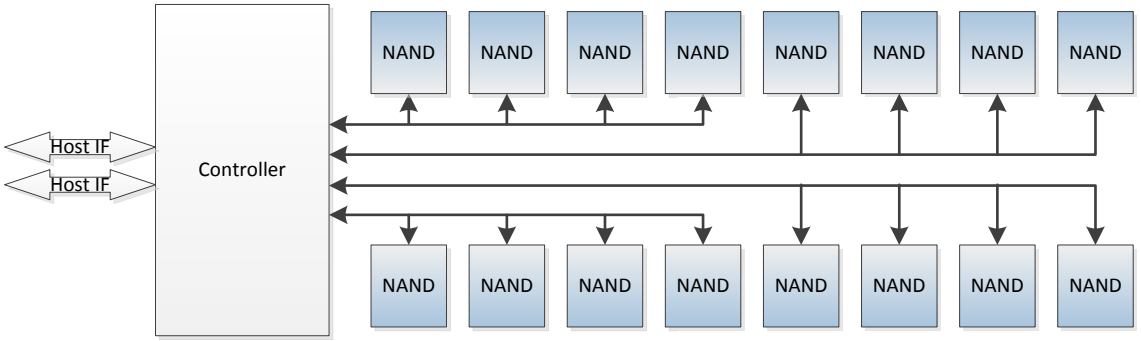
All-Flash Arrays: Hybrid-Caches and AFA

- Traditionally SSD's have been used for caches in hybrid system
  - SSD's high speed used to offset long HDD latency
  - Small SSD capacity has small cost impact
  - SSD as percentage of total capacity grew over time
- Controller firmware responsible for moving data into SSD's
  - Trying to keep as much data as possible in fast SSD's



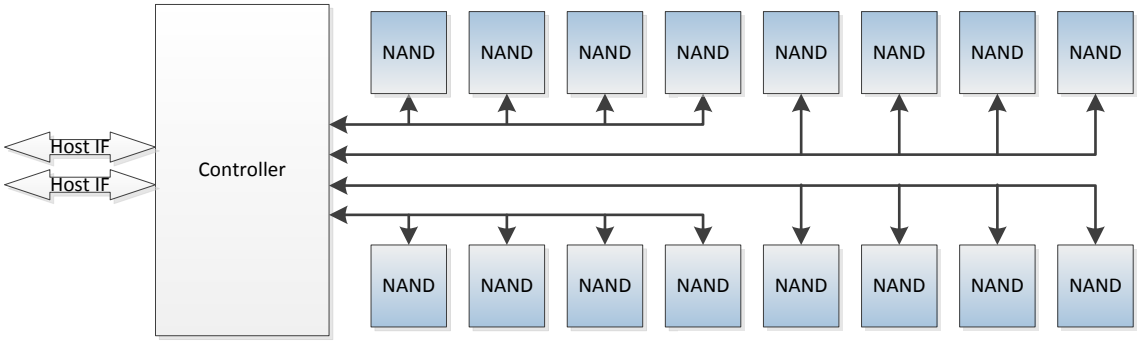
All-Flash Arrays: Hybrid-Caches and AFA

- Hybrid systems easily transitioned to SSD Arrays
  - As the percent in SSD approached 100%
- Transition increased simplicity and performance
- AFA can manage Flash directly or using SSD's
  - Performance and density limited by using HDD-based standards



All-Flash Arrays: Hybrid-Caches and AFA

- All-Flash Arrays see more of the traffic
  - Can take advantage of differences in traffic patterns
- AFA are much faster than HDD or hybrid drives
  - Have achieved price parity with HDD
- Cost of AFA is dominated by the cost of Flash



All-Flash Arrays: Traffic Considerations

- Most applications do not require maximal OP
- Some hybrid-caching applications would be worst-case

Application	Xfer Size	Writes	Comp
Hybrid Caching	Small	High	Medium
Video Streaming	Large	Low	Low
Web Search	Medium	Medium	Medium
Database Processing	Small	Medium	Medium

All-Flash Arrays: Performance Tradeoffs

- The amount of OP should be kept as small as necessary
  - For both performance and endurance goals
  - Flash cost is paramount
- That necessary amount is smaller in All-Flash Arrays than in hybrid caches
  - No longer subject to worst-case traffic, except benchmarks
- There are effects which are not captured here
  - Underlying NAND technology and handling
  - Efficiency of firmware

Conclusion

- Overprovisioning is a necessary response to NAND die architecture
  - Part of the structure used to convert to Flash protcol
- More is better, if everything else is equal
  - Improves performance and reduces wear
  - More OP is more expensive
- Price of OP may be hidden in system cost
  - Need to be careful about capacity and cost
- Know your application
  - Optimal OP depends upon data size, compression, access pattern, and firmware behaviour

Attribution & Feedback

The SNIA Education Committee thanks the following individuals for their contributions to this Tutorial.

Authorship History

Bill Radke / Aug 24, 2013

Updates:
Bill Radke / Sept 24, 2013

Additional Contributors

Please send any questions or comments regarding this SNIA Tutorial to ***tracktutorials@snia.org***