IBM

NAND Flash Media Management Algorithms

Roman Pletka
IBM Research – Zurich Research Laboratory

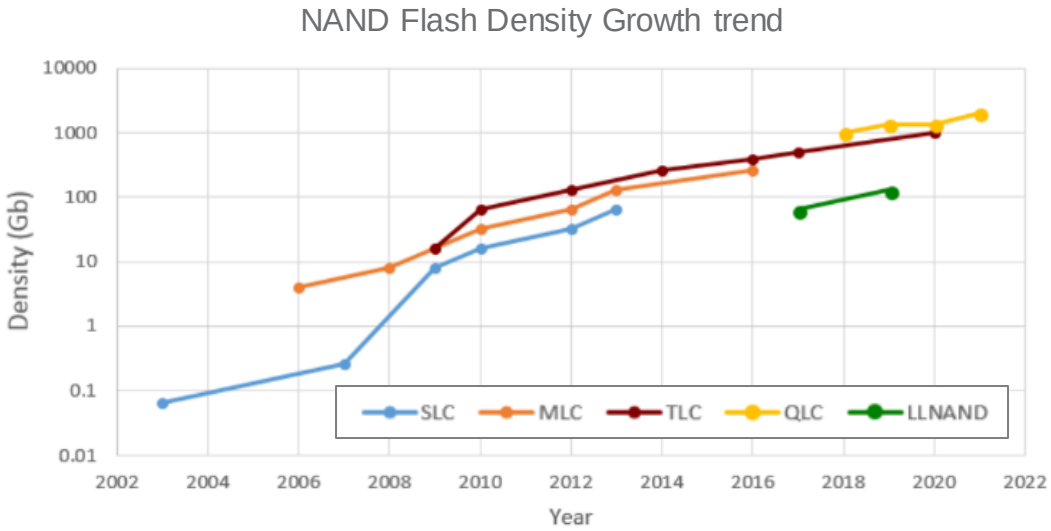
Outline

- Background
  - NAND Flash Scaling Trends
- ECC
  - Hard and Soft Decision Decoding
- Read Voltage Calibration
- Flash management
  - Garbage collection
  - Wear leveling
- Protection against media failures
- Compression
- Summary

NAND Flash density trends

- Flash density will continuously increase by:
  - layer-count scaling beyond 200 layers (2020-2023+)
  - lithography shrinking (2022+)

  ⇒ no need for EUV for 7+ years in flash
- Significant 3D-NAND process, architecture, and cell materials innovations needed for continued scaling beyond 200 layers (yield, cost)



Source: J. Yoon, IBM, FMS2018

NAND Flash impairments

IBM

Impairment	Effect	Mitigation
Program/Erase cycling	Widens and shifts voltage distributions	ECC, read voltage calibration
Retention	Charge loss causing voltage shifting and widening	ECC, read voltage calibration, static wear leveling
Read disturb	Cause slight programming of unselected word-lines	ECC, read voltage calibration
Page program/read vs block erase operations	Requires garbage collection of blocks to reclaim space causing write amplification	Indirection table from logical to physical addresses, dynamic wear leveling, health binning
Media defects	Page, block, plane, or die failures	ECC, variable block stripes with parity, cooperative error recovery with RAID controller

ECC overview

BCH codes

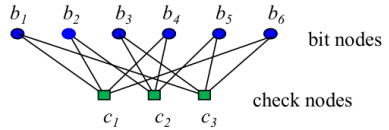
- BCH codes typically support hard-decision decoding only
- Conventional controllers use BCH codes with 40 bit error correction over 1 KB code words
- Today, more advanced BCH-based codes are typically used with larger code word sizes (e.g., TPC, …)
- Error recovery by read retry with different read voltages

LDPC codes

- Defined by a sparse (low density) parity check matrix H

$$H = \begin{array}{c} \begin{matrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} c_1 \\ c_2 \\ c_3 \end{matrix}$$

- Are represented with a bi-partite graph



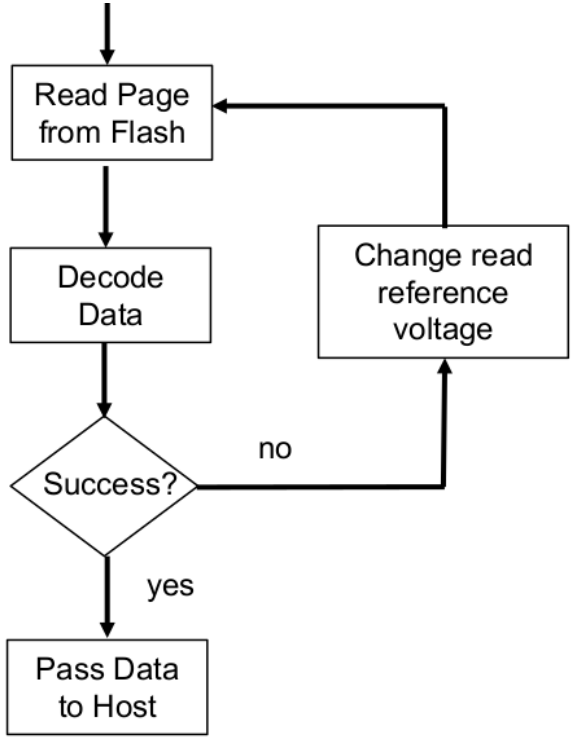
- Support hard and soft decision decoding
- Soft decision decoding requires additional soft information from additional reads

Any ECC code benefits from optimized read threshold voltages!

Read retry algorithm

- Default read reference voltage optimized for typical condition only
- Read retry algorithm cycles through several individual read decoding steps
- Retry steps use read reference voltages optimized for program/erase cycling, retention, read disturb, etc.

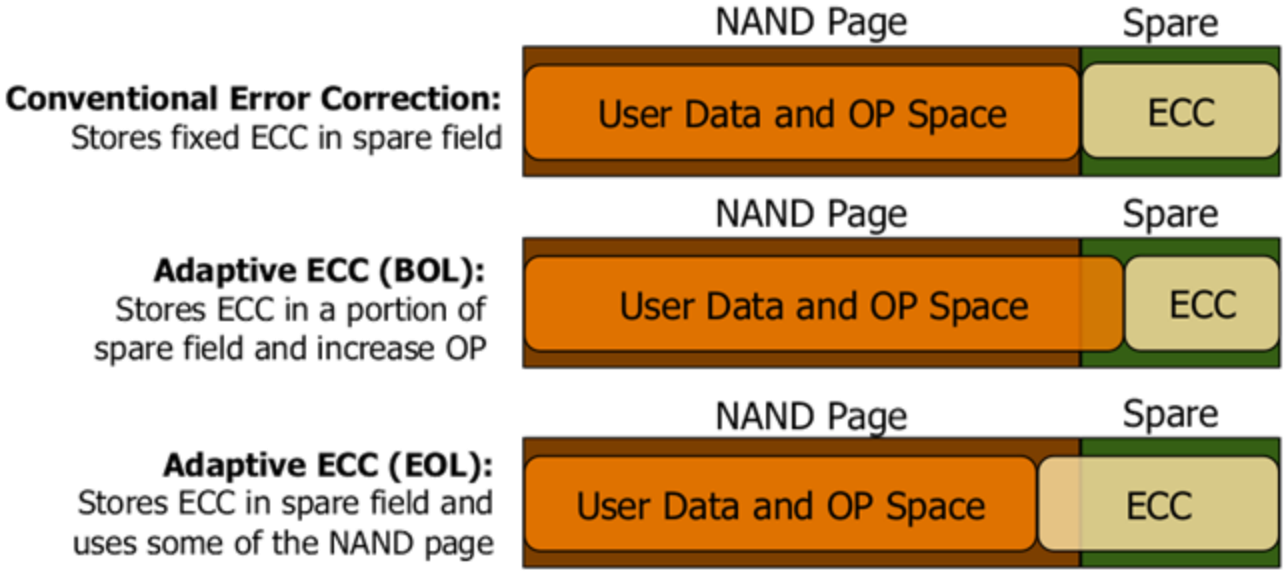
⇨ A controller design should reduce read retry operations by pro-actively calibrating blocks



Source: E. Haratsch, FMS2018

Adaptive code rates

- Beginning of Life: use less ECC to increase overprovisioning
- End of life: increase ECC to maintain reliability



Adaptive ECC allows for more free space @ BOL = More OP and less write amplification

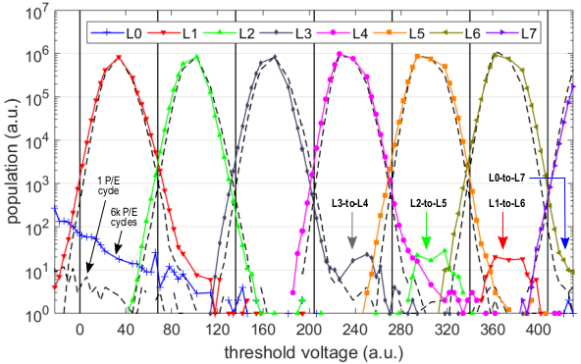
Source: E. Haratsch, FMS2018

Effects on read voltage distributions

Example read voltage distribution effects in 3D TLC NAND Flash:

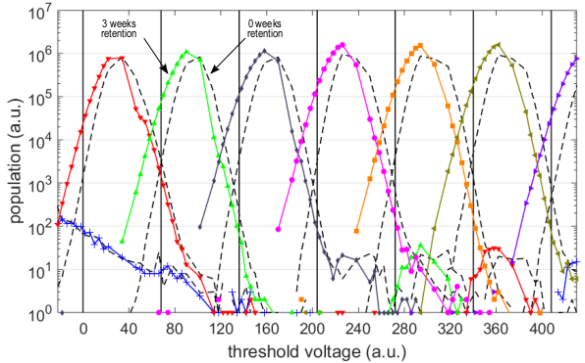
P/E cycling

- 1 vs 6k P/E cycles
- widens tails of distributions,
- presence of 2-step program effects: L0-to-L7



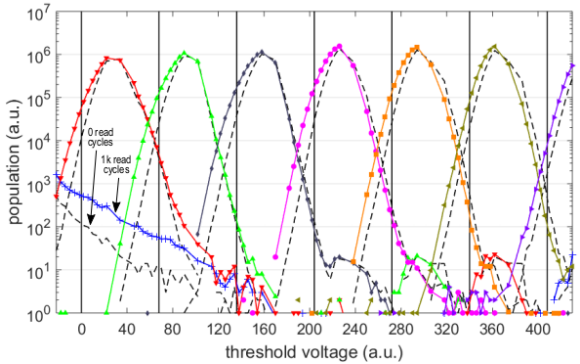
Retention

- 6k P/E cycles + 0 vs 3 week retention
- negative shift and increasing left tails



Read disturb

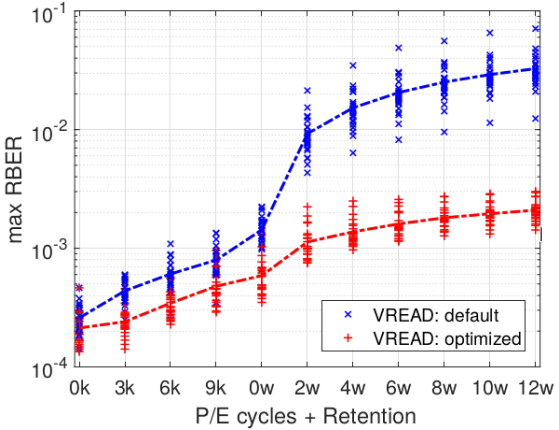
- 9k P/E cycles + 0 vs 1k rd cycl.
- lower levels wider
- moderate negative shift due to short term retention

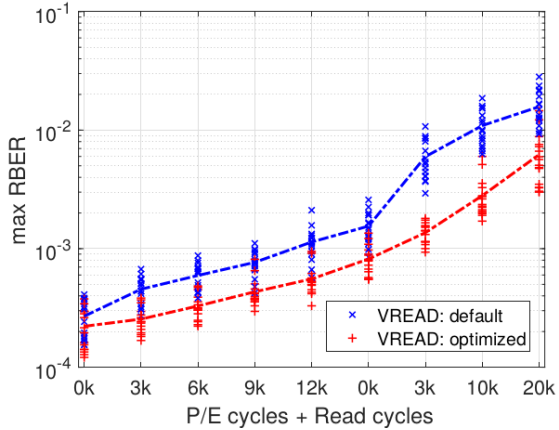


Source: N. Papandreou et al., IRPS 2019

Read voltage calibration

- Read voltage calibration:
  - Requires special access modes to Flash.
  - Extensive characterization is required to determine behavior of read voltage levels under different conditions.
  - Read voltage levels depend on:
    - Number of P/E cycles
    - Number of reads a page has seen since programmed
    - Retention time
    - Individual block/layer/page characteristics
- Block calibration determines optimal read voltage levels continuously in the background. Benefits are:
  - ~3x more endurance
  - Significant reduction of read retry operations
  - No impact on host read and write operations

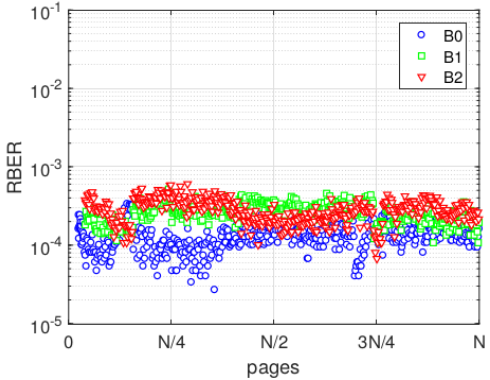


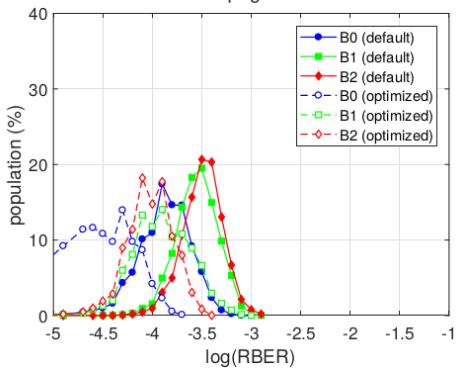


Source: N. Papandreou et al., IRPS 2019

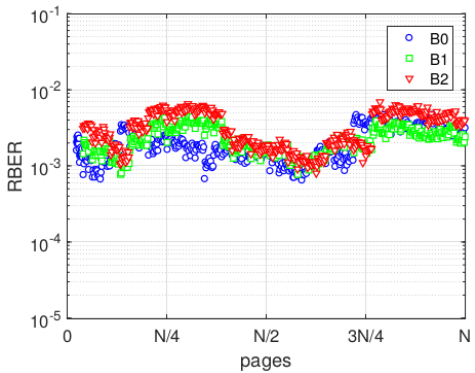
RBER characteristics and page type

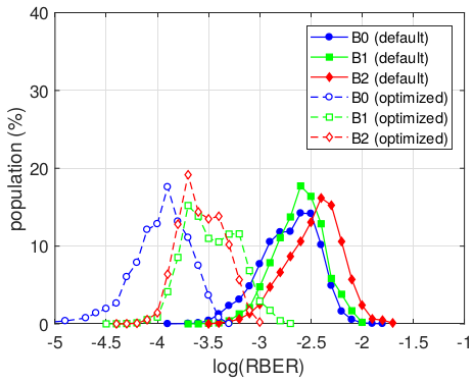
9k P/E cycles



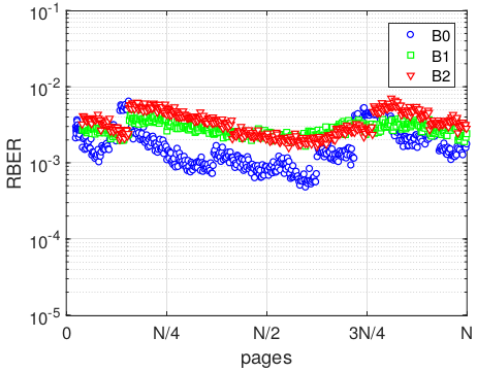


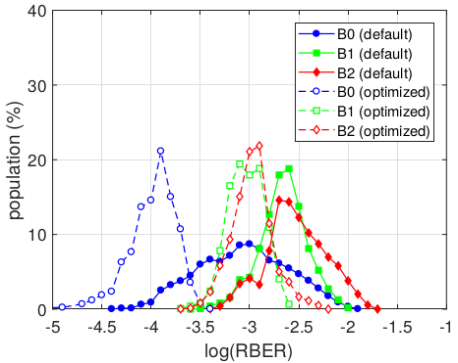
9k P/E cycles + 4 week retention





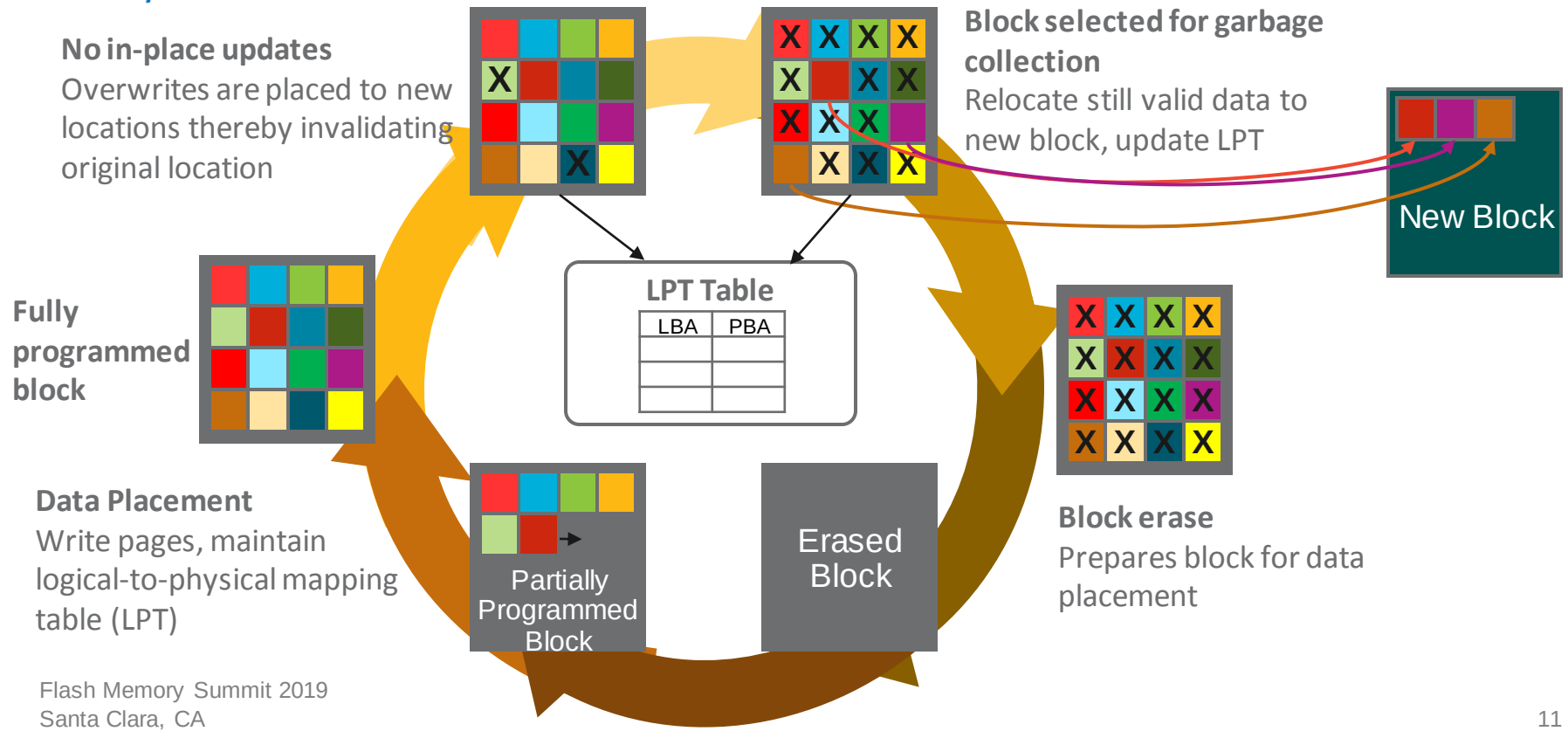
9k P/E cycles + 10k read cycles





Source: N. Papandreou et al., IRPS 2019

The Flash program-erase cycle



Traditional wear leveling (WL)

Dynamic WL

Balance P/E cycles across blocks upon overwrites and relocations. Typically uses the least worn available block to place new data.

Static WL

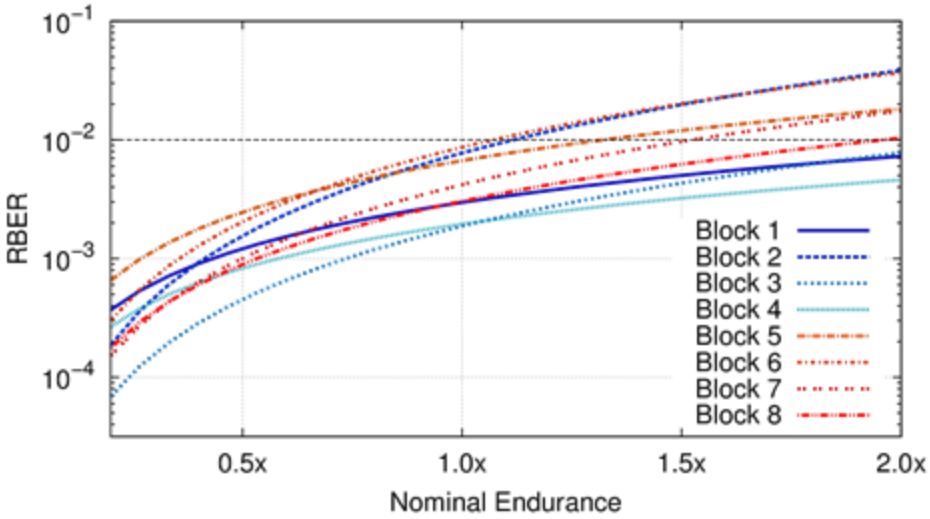
Identifies the least worn blocks holding static data in the background. Still valid data is relocated to another block causing an increase in write amplification.

P/E Cycle-based WL

Balances wear of blocks based on their program-erase cycle count only.

Flash block characteristics

- Significant differences in the RBER over time can be observed in blocks within the same device:
  - Some blocks have almost twice the endurance of others
  - Low RBER at early life does not indicate a good block, and an early high RBER not a weak one



RBER of different flash blocks in the same device as a function of P/E cycles

From WL to Health Binning

Dynamic WL

Balance P/E cycles across blocks upon overwrites and relocations. Typically uses the least worn available block to place new data.

- Introduce data placement with stream segregation
- Use better blocks for hotter data

Static WL

Identifies the least worn blocks holding static data in the background. Still valid data is relocated to another block causing an increase in write amplification.

- Reduce Static WL to address retention and read disturb only
- Relocations instead of block swapping

P/E Cycle-based WL

Balances wear of blocks based on their program-erase cycle count only.

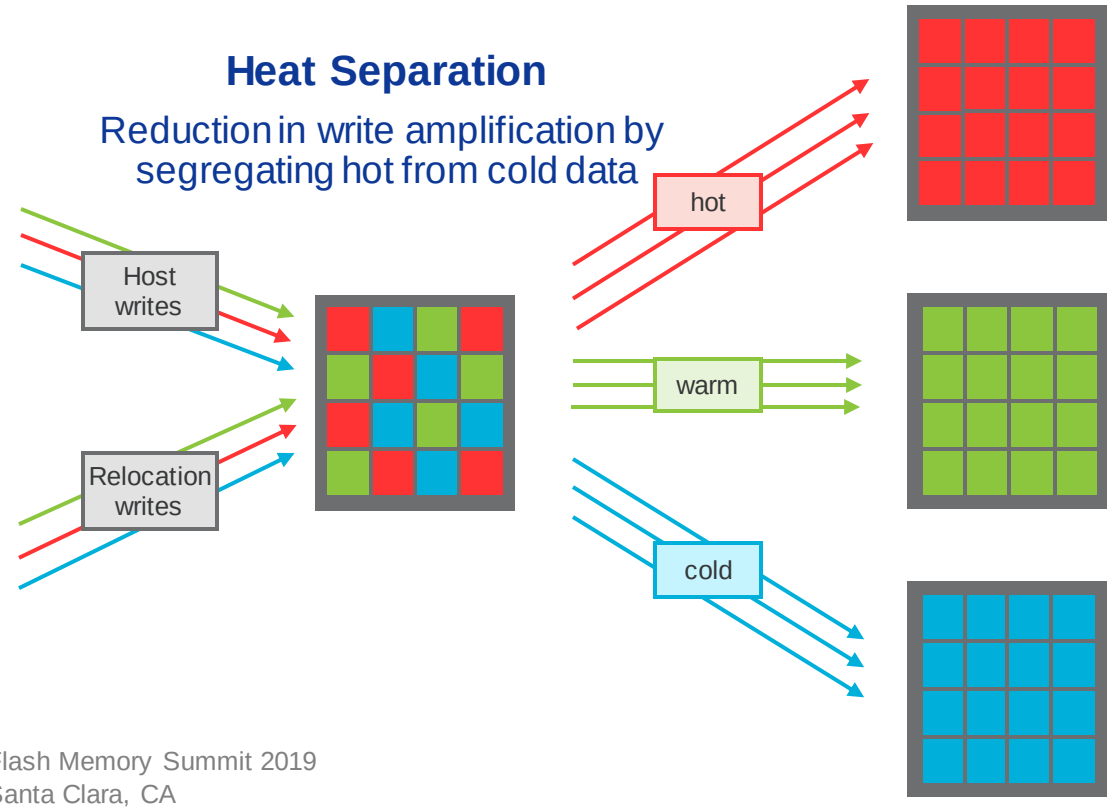
- Background grading of blocks based on RBER
- RBER estimation based on ECC feedback

Endurance gains of up to 80%!

Combining heat separation with health binning

Heat Separation

Reduction in write amplification by segregating hot from cold data



Healthiest Blocks

Health Binning

Determine block health during background operations.

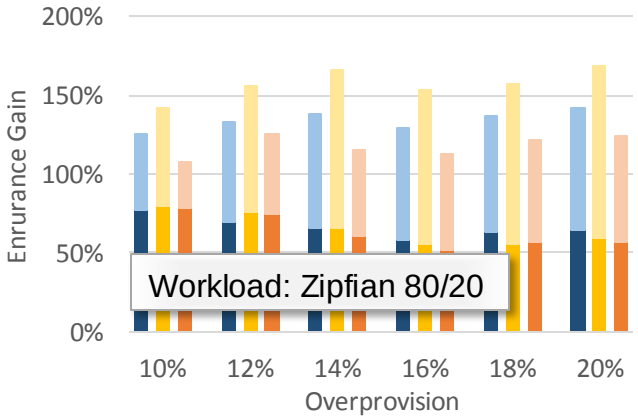
- Hottest data is written to healthiest blocks,
- cold data to less healthy block

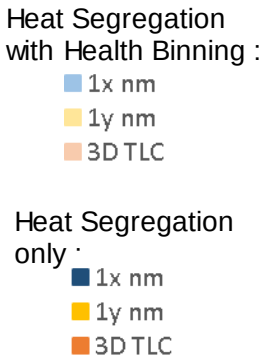
Least Healthy Blocks

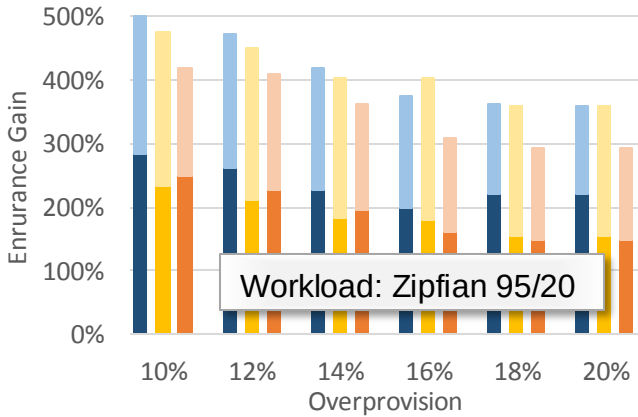
Endurance gains

Endurance gains from write separation and health binning:

- Substantially higher gains for more skewed workloads thanks to reduced GC overhead
- Endurance gains are increasingly dominated by Health Binning with higher over-provisioning as relative gains from write amplification reduction decreases at the same time
- Smaller feature size have higher gains from health binning due to increase in the block variability



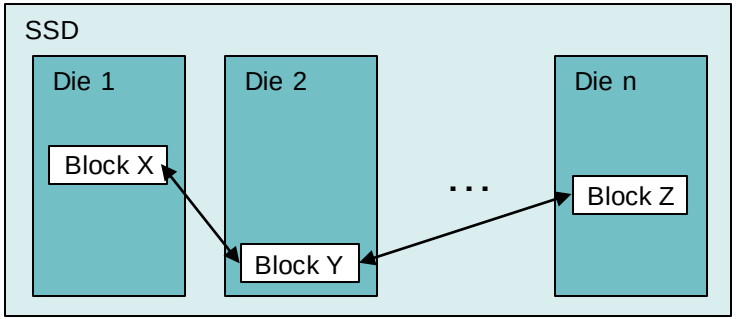


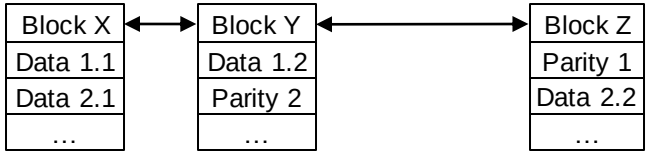


Media failures

- Pages, blocks, planes or the whole die can fail
- ECC cannot recover data from such catastrophic failures

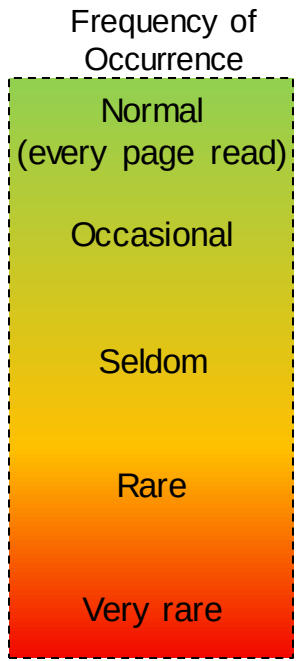
- Need RAID-like protection inside SSD
  - Write data across multiple dies with additional protection
  - Corrects full page, block or die failures when ECC fails
  - RAID stripes can be of variable size





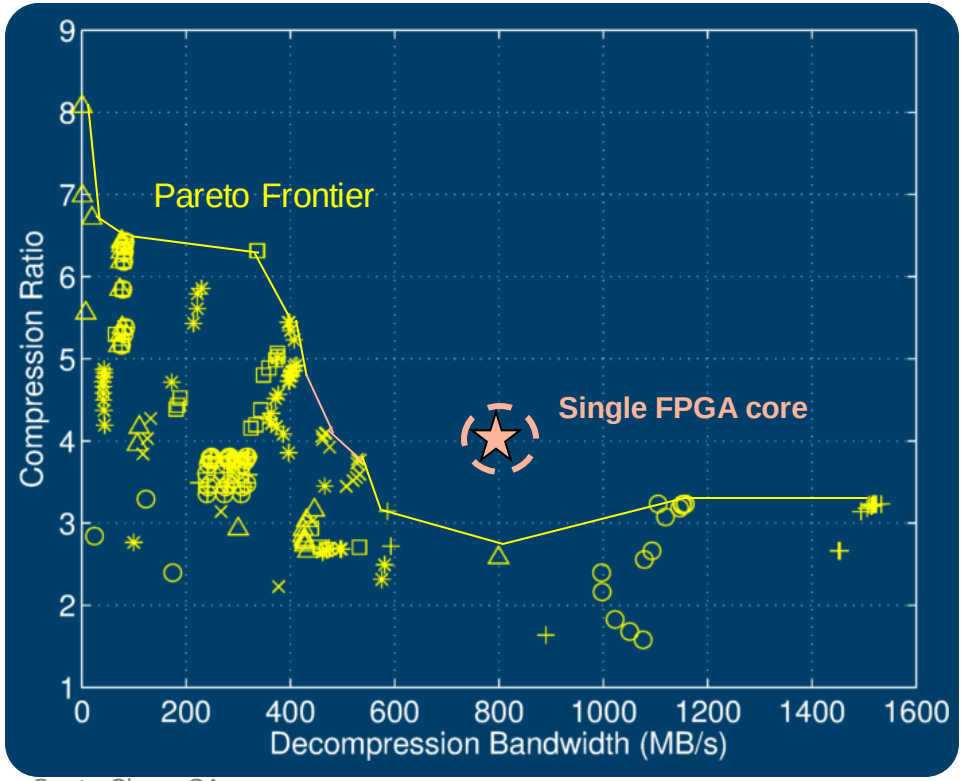
Multi-level error correction

1. Hard decision ECC decoding
2. Progressively apply stronger decoding methods such as
   - soft-decision decoding and signal processing, or
   - use read retry methods
3. Read voltage calibration to reduce effects of P/E cycling, retention, read disturb, etc
4. RAID-like parity among different dies
5. Cooperative system-level RAID and SSD error recovery schemes



Additional error detection using CRC on logical data and page headers

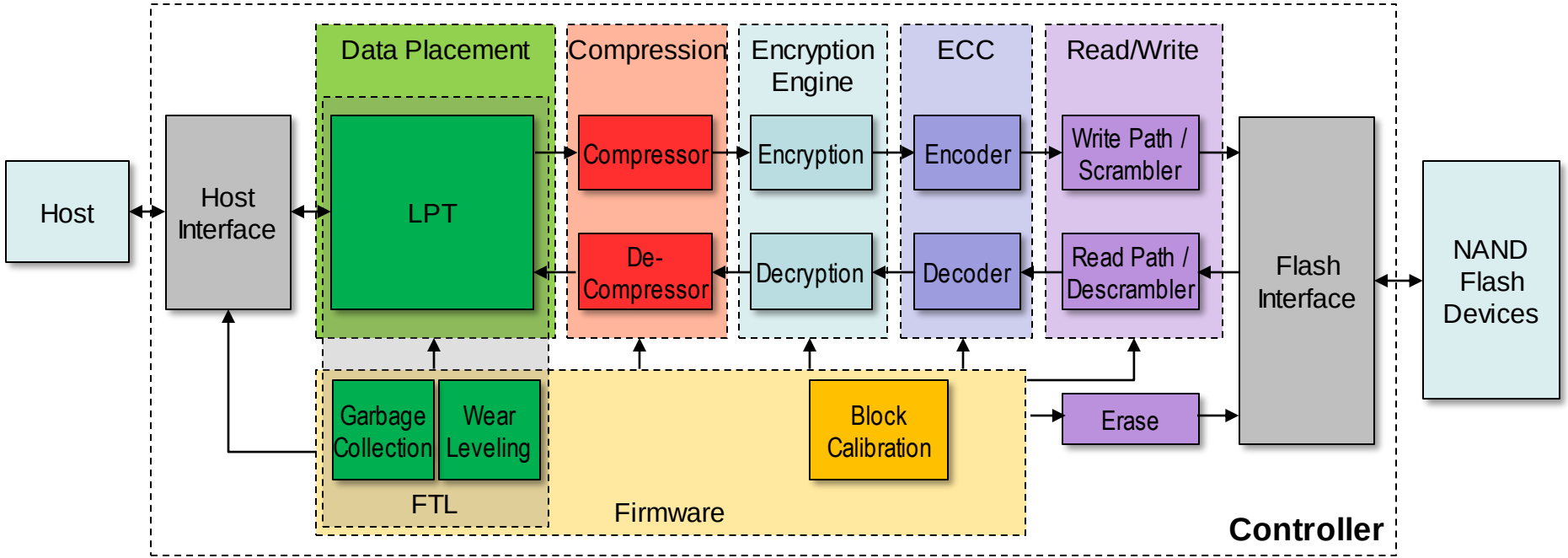
Hardware compression



Hardware compression for IBM FlashSystem™

- Compression reduces internal write amplification
- Low power FPGA implementation
- Multiple compression engines per FPGA core
- Each flash card has its own compression engines
- System can deliver inline compression @ > 1M IOPs
- Built into data path as a streaming engine (no store and forward)
- Based on Lempel-Ziv coding combined with pseudo-dynamic Huffman codes and a 4KiB history buffer

SSD controller block diagram



Conclusion

IBM

- Complexity of SSD controllers keeps increasing with every NAND flash generation
- 3D NAND flash relies on strong ECC algorithms, especially with increasing number of bits per cell.
- Latest memory technology demands intelligent NAND management features for data placement, garbage collection, wear leveling, and block calibration
- Modern controllers use multiple levels of error corrections including strong ECCs, signal processing, read voltage calibration, RAID-like parity schemes

Thank You !