the *Future* of *Memory* and *Storage* 2024

Flexible Data Placement (FDP):
What Every Storage Architect Should Know

Rory Bolt

KIOXIA America, Inc.

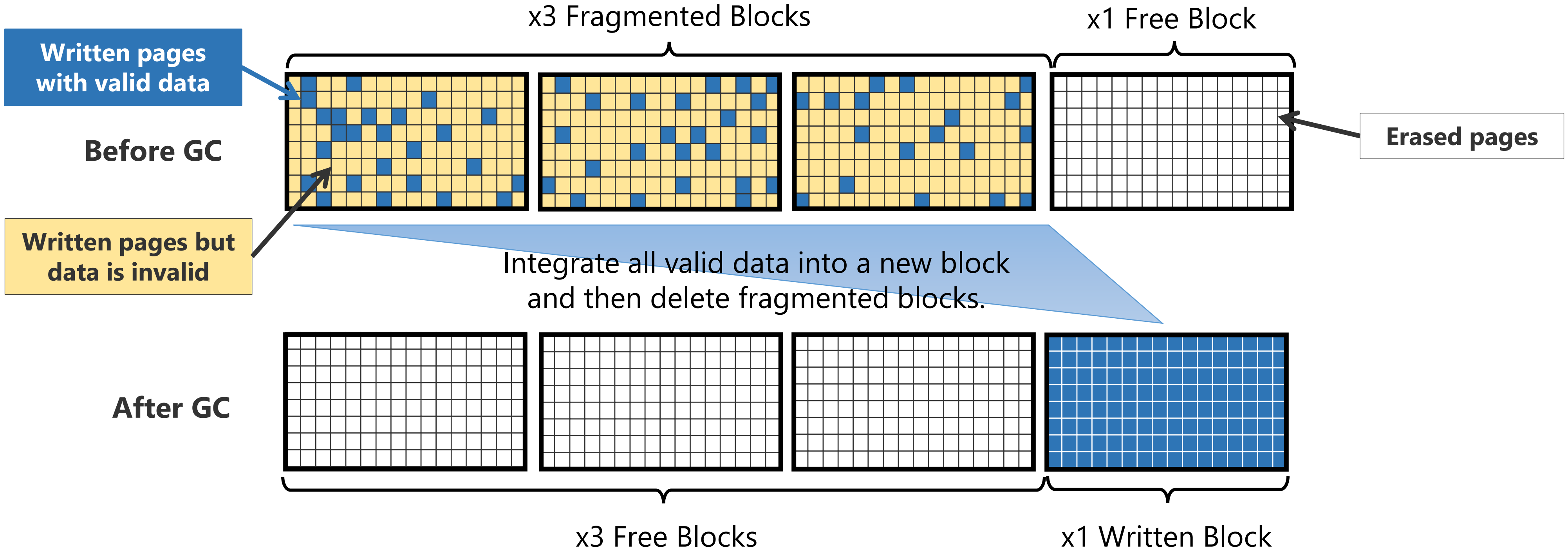
FARP-101-1

August 6, 2024

KIOXIA

Motivation For Flexible Data Placement (FDP)

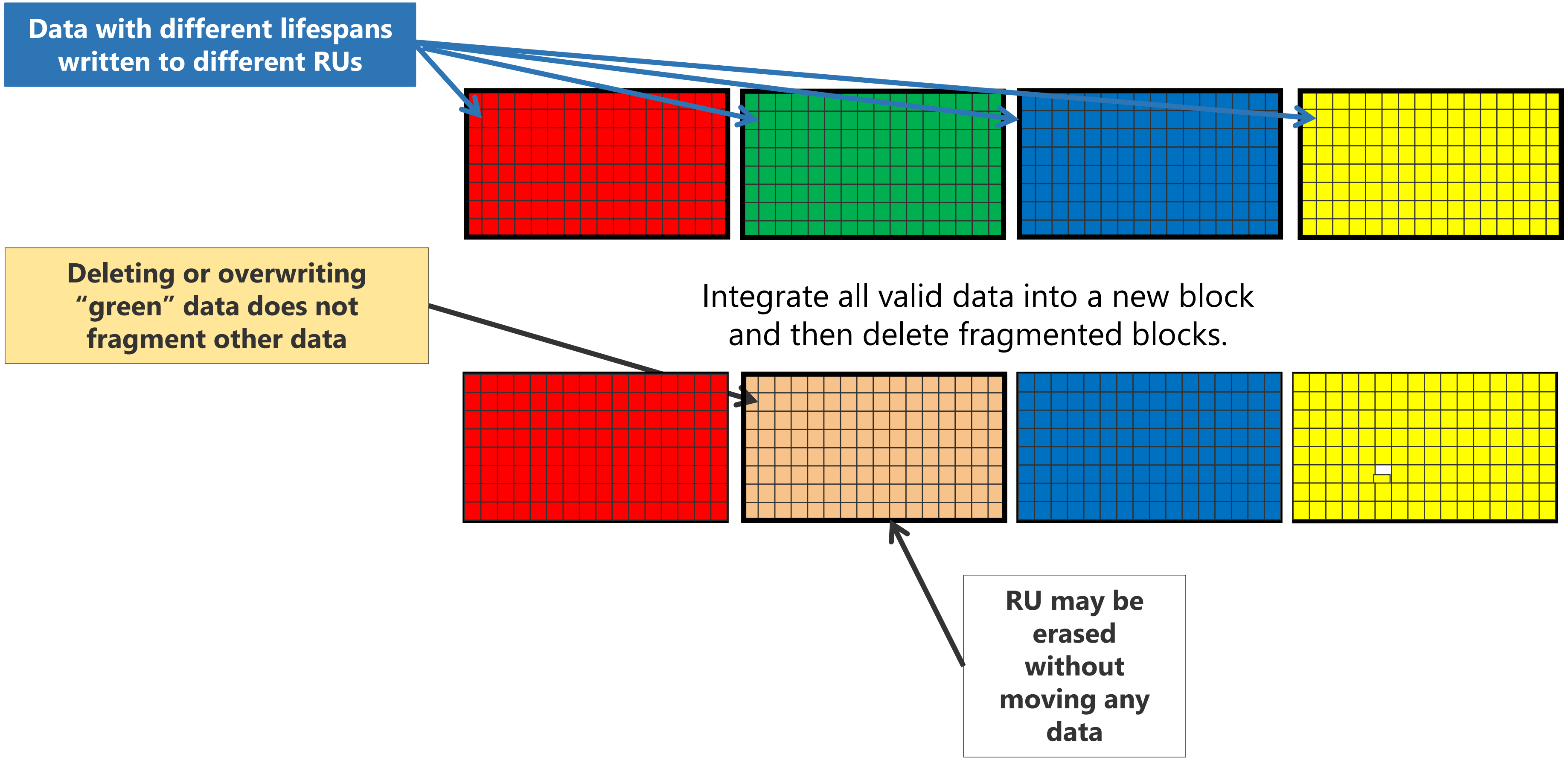
- The difference in erase granularity and program granularity is the fundamental reason for garbage collection (GC), the associated write amplification, and ultimately the desire for controlling data placement


x3 Fragmented Blocks
x1 Free Block
Written pages with valid data
Before GC
Erased pages
Written pages but data is invalid
Integrate all valid data into a new block and then delete fragmented blocks.
After GC
x3 Free Blocks
x1 Written Block

- When Logical Block Addresses (LBAs) are overwritten or trimmed, the associated area of the media is invalidated but cannot be reused
- If any data in the Reclaim Unit (RU) is still valid, it must be copied to a new location, creating write amplification

The Promise of Flexible Data Placement (FDP)

- FDP provides a mechanism to group data that is likely to be discarded or overwritten at the same time to increase the likelihood of reusing a RU without copying any data, thereby minimizing the amount of write amplification. Write amplification affects performance and device lifespan, and is a particular concern for QLC media



Data with different lifespans written to different RUs

Deleting or overwriting "green" data does not fragment other data

Integrate all valid data into a new block and then delete fragmented blocks.

RU may be erased without moving any data

- NOTE: Various internal media management requirements prevent ever truly eliminating write amplification…

Flexible Data Placement Is Highly Configurable

- Extremely flexible specification for controlling how data is organized on flash media
- Devices may implement different configurations, enabling control from the superblock level to individual die
- Flexible Data Placement drives from different vendors may have different capabilities
- Devices are unlikely to support all possible configurations due to Quality Assurance (QA) complexity

- This presentation will focus on superblock level control…
  - Currently seems to be the most popular choice
  - Most, if not all, modern SSDs perform cross-die RAID at the superblock level to enhance device reliability
  - Striping across die maximizes parallelism within the device without additional host overhead
  - Write buffer and power loss protection implications of die-level control

Things To Know When Using Flexible Data Placement

- There are few actual guarantees, and lots of possible implementations…
- Do not make assumptions on aligning data structures to RUs
  - Media characteristics (e.g. Flash block size) vary from vendor to vendor and generation to generation
  - Flash block sizes are rarely a power of 2
  - Flash block sizes are growing…
- Do not rely on data written to a Reclaim Unit Handle (RUH) ending up in the current RU, even if you checked the remaining space and are implementing Queue Depth (QD)=1
- Superblock size (e.g., RU size) can be quite large indeed
- The number of RUHs supported is likely to stay small
  - Each RUH consumes write buffer, which in turn has power loss protection implications
- There is currently no guaranteed interlock between trim and garbage collection
- There are possible race conditions that can result in unnecessary write amplification when QD > 1

Best Practices From A Device Perspective

- If you want to approach a Write Amplification Factor (WAF) of 1.0:
  - Overprovisioning helps (at a cost, of course). This can include “short stroking” in addition to the manufacturer’s factory-configured overprovisioning. This minimizes unnecessary write amplification due to race conditions
- Assuming direct LBA access:
  - Log structured write algorithms and circular buffers generally work well with FDP
    - Log Structured Merge (LSM) of RocksDB is a great example
    - Transaction journals
    - Copy-on-write snapshotting file systems
  - Separate hot and cold data whenever possible
    - Traditionally classifying data at creation is very difficult
    - Copy-on-write storage systems that perform application level garbage collection can do this naturally during garbage collect
  - If you can’t afford to fundamentally change your storage applications, look for low hanging fruit
    - The Pareto Principle usually holds true; just separating out a small portion of your data may yield large results (i.e. metadata is often far more volatile than user data, temp files can be moved to a dedicated namespace with a default placement ID)

KIOXIA