



# Flexible Data Placement: State of the Union

**August 2023**

**Chris Sabol, Google**

**Ross Stenfort, Meta**

**Mike Allison, Samsung**

# Write Amplification Overview

What is Write Amplification (WA)?

- When the host sends write data to the device it is additional data that is written to the media
- Write Amplification Factor (WAF) = media written data/ host written data

WAF = 2.5 Example

- Host writes 1 MB
- Device writes 2.5 MB to the media
  - Media Writes = 1 MB Host Data + *Additional 1.5 MB Garbage Collected Data*
  - Extra Media Reads to enable host write = *1.5 MB*

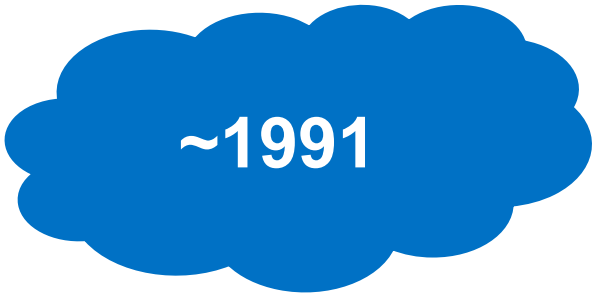
# Why is Write Amplification Undesired

- Write Amplification results in additional:
  Media Reads/ Writes affecting performance/ QoS
  Flash media writes causing non-host induced media wear
  Additional power needed to perform the additional reads/writes

- Random Write example:

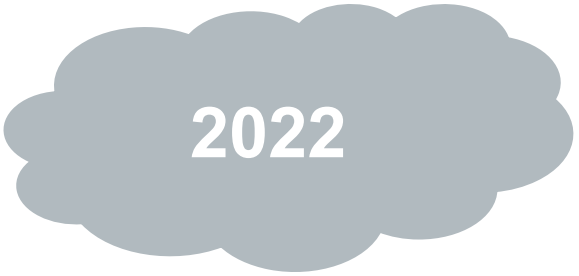
| Write Amplification Factor | Write Performance |
|---|---|
| 1 | Random Write = Sequential Write |
| 5 | Random Write = 20% Sequential Write |

# Write Amplification Improvement Timeline




**NAND Based SSDs**
Solution #1: Overprovisioning


**Host Provides SSD LBA Hints**
Solution #2: TRIM/Deallocate




**Host Provides Data Placement Hints**
Solution #3: Flexible Data Placement (NVMe TP 4146)

# How Flexible Data Placement Developed

How did Flexible Data Placement come about?

- Google Write Amplification Investigation Result
  - Data placement on media is key
  - SMART FTL Proposal
- Meta Write Amplification Investigation Result
  - Data placement on media is key
  - Direct Placement Mode Proposal

➢ Google & Meta merged their independent learnings into Flexible Data Placement (FDP) merging the best features of each proposal to enable best industry solution

What is the status of this in NVM Express?

- TP4146 is fully ratified
- Link: https://nvmexpress.org/wp-content/uploads/NVM-Express-2.0-Ratified-TPs_12122022.zip

# FDP:  High Level Overview

- Enables host to provide hint where to place data via virtual handle/pointer
- Device changes:
  - Places data in super block based on hint rather than choosing it's own super block
  - Advertises size of super block
- What functionality does not change
  - Read
  - Write (Optional write handle added)
  - Deallocate/TRIM
  - Security
- Backwards compatibility
  - FDP may be enabled/disabled on standard devices
  - Applications are not required to understand FDP to get benefits
  - Applications which understand FDP have increased benefits

# FDP: Open Source Activities

**Goal:** Support FDP through a full upstream I/O Path

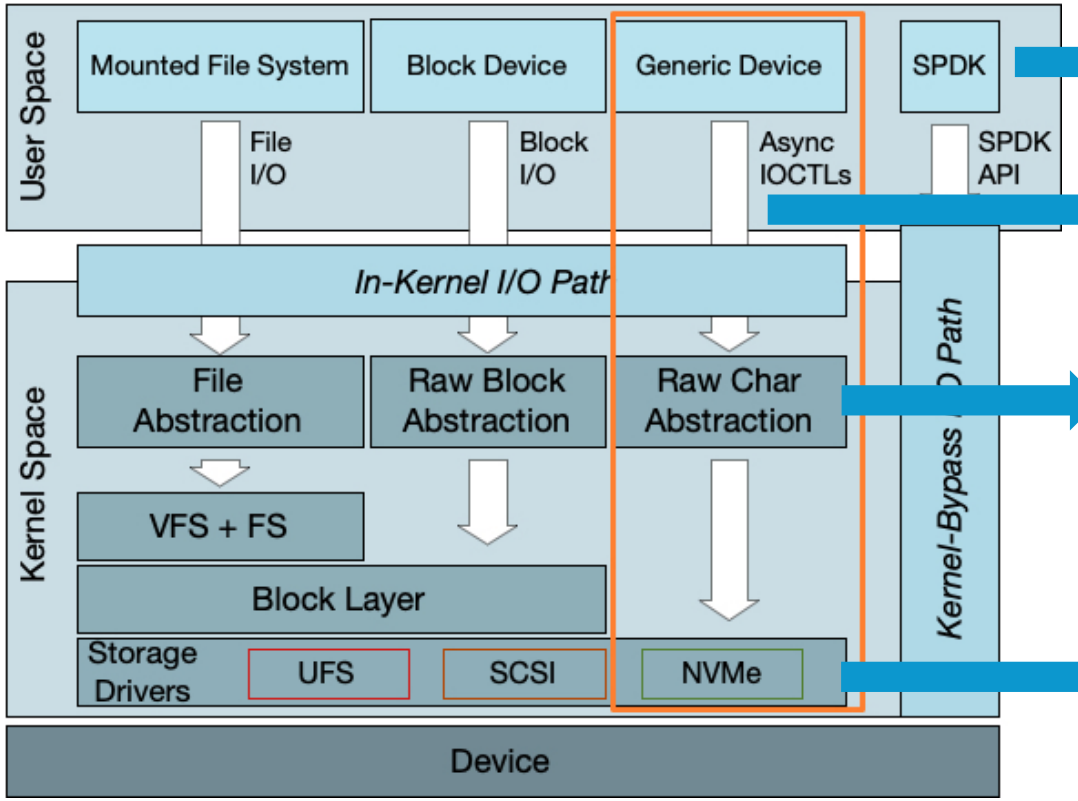
**Current Support**

| | | |
|---|---|---|
| Linux Kernel | Full Support through I/O Passthru | **Upstream since 5.19** |
| xNVMe | Full Support | **Upstream since v0.7** |
| QEMU | FDP Emulation<br>*Validation of host stack. No simulation (e.g. WAF, performance) | **Upstream since v8.0** |
| Fio | Basic support for RU and RUH<br>Working on improving generic trim in io_uring | **Upstream**<br>**Ongoing** |
| nvme-cli | Support for FDP commands and log pages | **Upstream** |
| Cachelib | Ongoing effort to reduce WAF through FDP | **Ongoing** |

# Linux Kernal:  I/O Passthru

- Enable any NVMe device to be available in Linux
  - Remove limitation of block layer
- Implemented in io_uring
  - Same scalability and performance
- Support end-to-end innovation for new interfaces though in-kernel I/O
- Support I/O stacks both in Cloud and Enterprise environments
  - I/O Passthru in SPDK though xNVMe
- Upstream since 5.19

## Architecture



# State of the Union Summary

| | | |
|---|---|---|
| **Specification** | Flexible Data Placement | NVM Express Ratified |
| **Open Source Activities** | Linux Kernel | Upstream since 5.19 |
| | xNVMe | Upstream since v0.7 |
| | QEMU | Upstream since v8.0 |
| | Fio | Upstream + Ongoing io_uring |
| | nvme-cli | Upstream |
| | Cachelib | Ongoing |

- FDP eco-system is ramping ***quickly***

# Thank You


*Architected for Performance*





# TP4146a Flexible Data Placement (FDP)

**Sponsored by NVM Express™ organization, the owner of NVMe® Family of Specifications**

# Speaker

Mike Allison

SAMSUNG

# Agenda

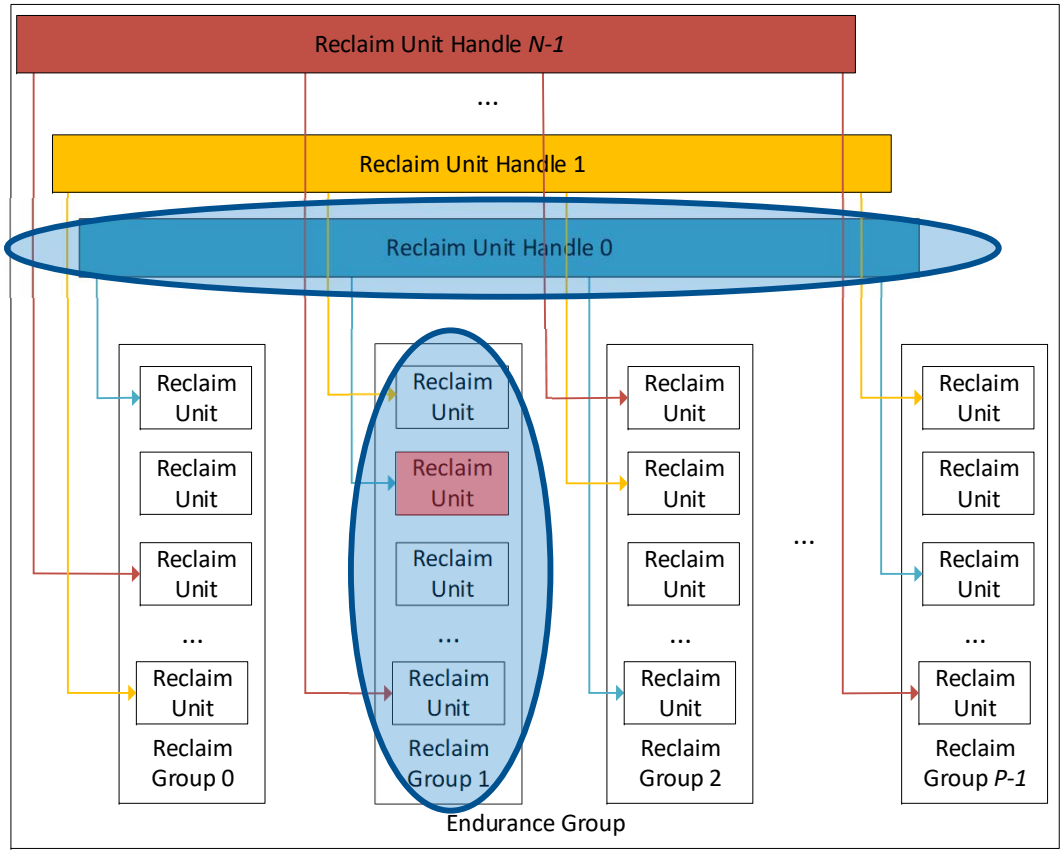
- Benefit
- Architecture
- Namespace Creation
- Streams vs. FDP vs. Zoned Namespaces (ZNS)

# FDP Benefit

- Path to an SSD with a Write Amplification of 1 that supports Random Writes
  - HOST not knowing the what and when an SSD garbage collects
    - Reduces the host write to the drive
    - Affects performance
    - Costs $$$
- Needed a mechanism that works across all namespaces
  - Both Streams and Zoned namespaces are per namespace
  - HOST can choose to or not to mixed namespace written to NAND blocks
- Backwards Compatible
  - Need a single drive with previous NVM Command Set behavior that enables FDP to be enabled
  - Need to be able to allow the HOST to choose what software to update and when
  - If the $$$ to update SW is greater that SSD WAF savings, then do not update the SW

# Storage Entities

- An FDP configuration consists of:
  - One or more Reclaim Units (RUs)
  - One or more Reclaim Groups (RGs)
  - One or more Reclaim Unit Handles (RUHs) that reference to a Reclaim Unit in each RG
- An Endurance Group that supports FDP
  - Supports one or more FDP configurations
- A Host enables specific FDP configuration in an Endurance Group
- Write commands allowed to specify and an Reclaim Group and Reclaim Unit Handle that indicates the Reclaim Unit to place the LBAs



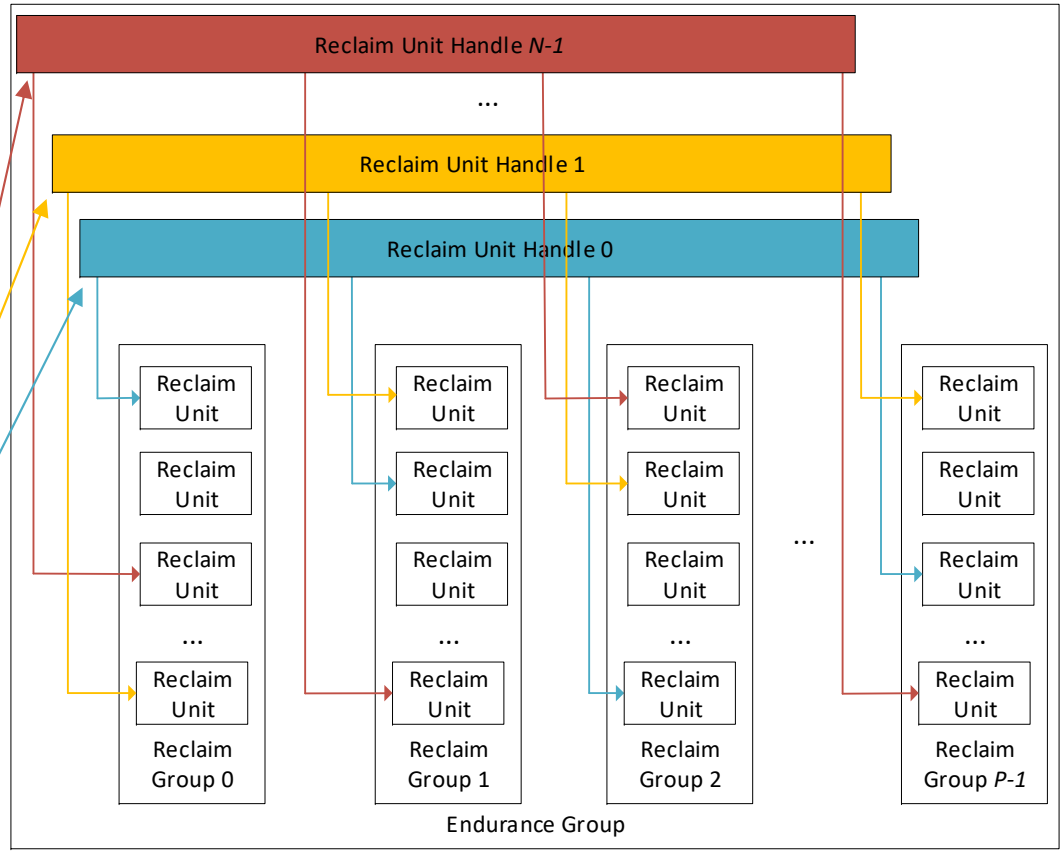
# Namespace Creation with FDP

Namespace Management command

- Specify the Endurance Group with FDP enabled
- Host may provide a Placement Handle List

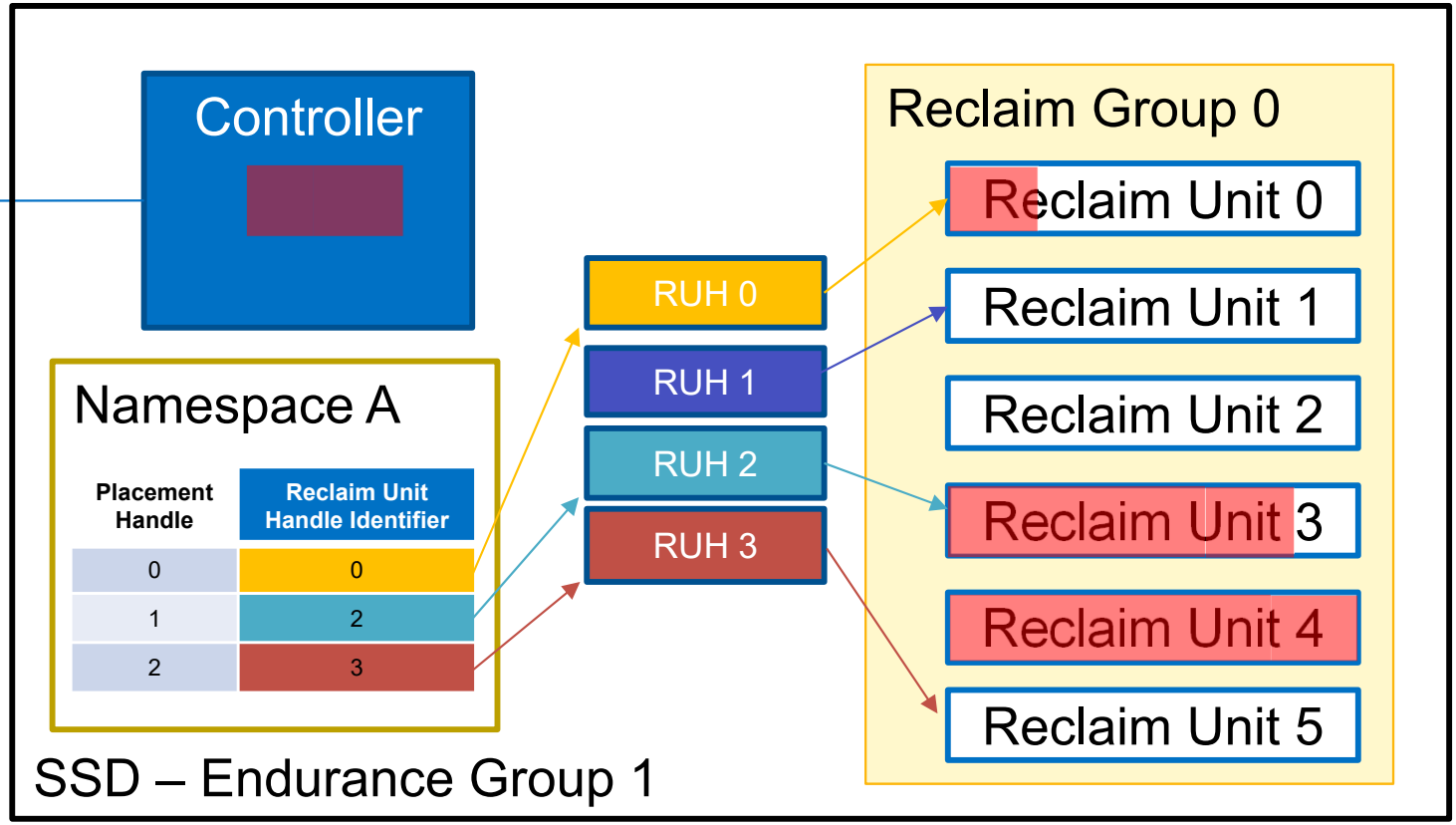
Namespace Scoped Values for VM isolation

| Placement Handle | Reclaim Unit Handle Identifier |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 2 | N-1 |



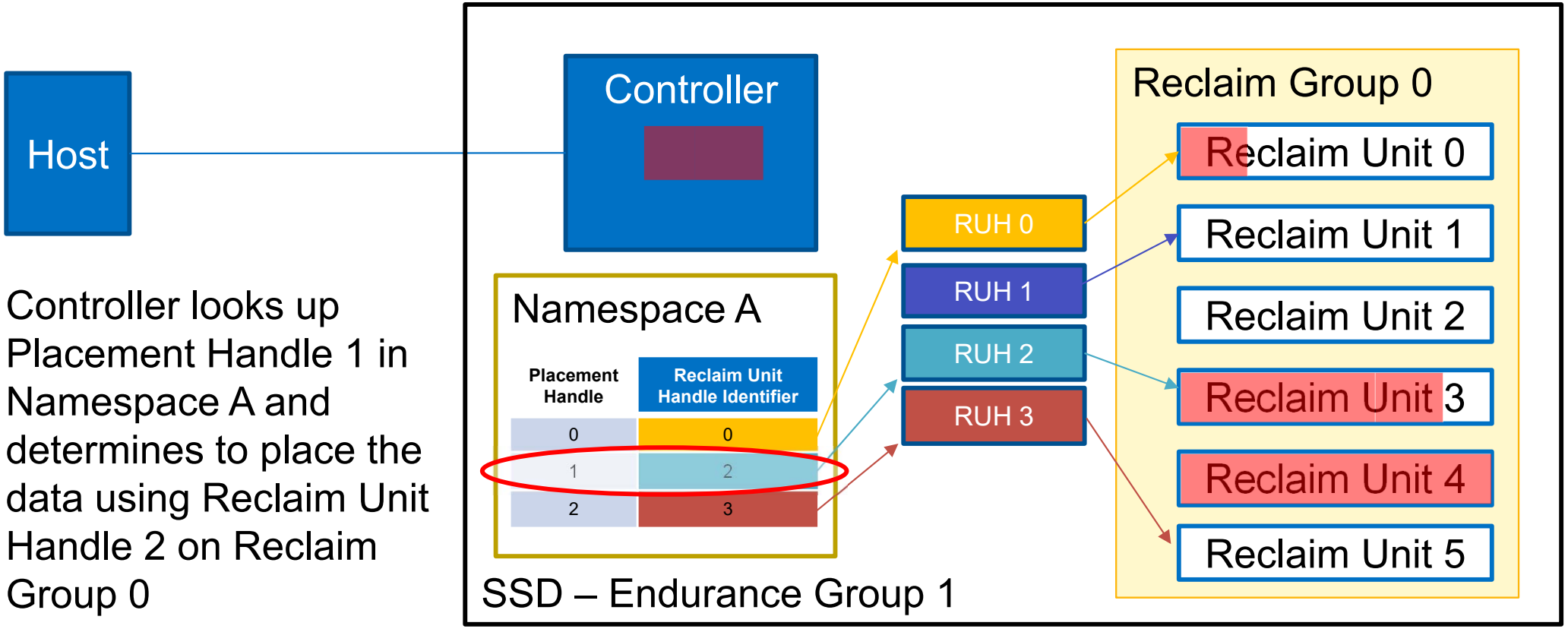
# Writing to a Namespace with FDP

Host issues a Write command to Namespace A specifying Placement Handle 1 and Reclaim Group 0



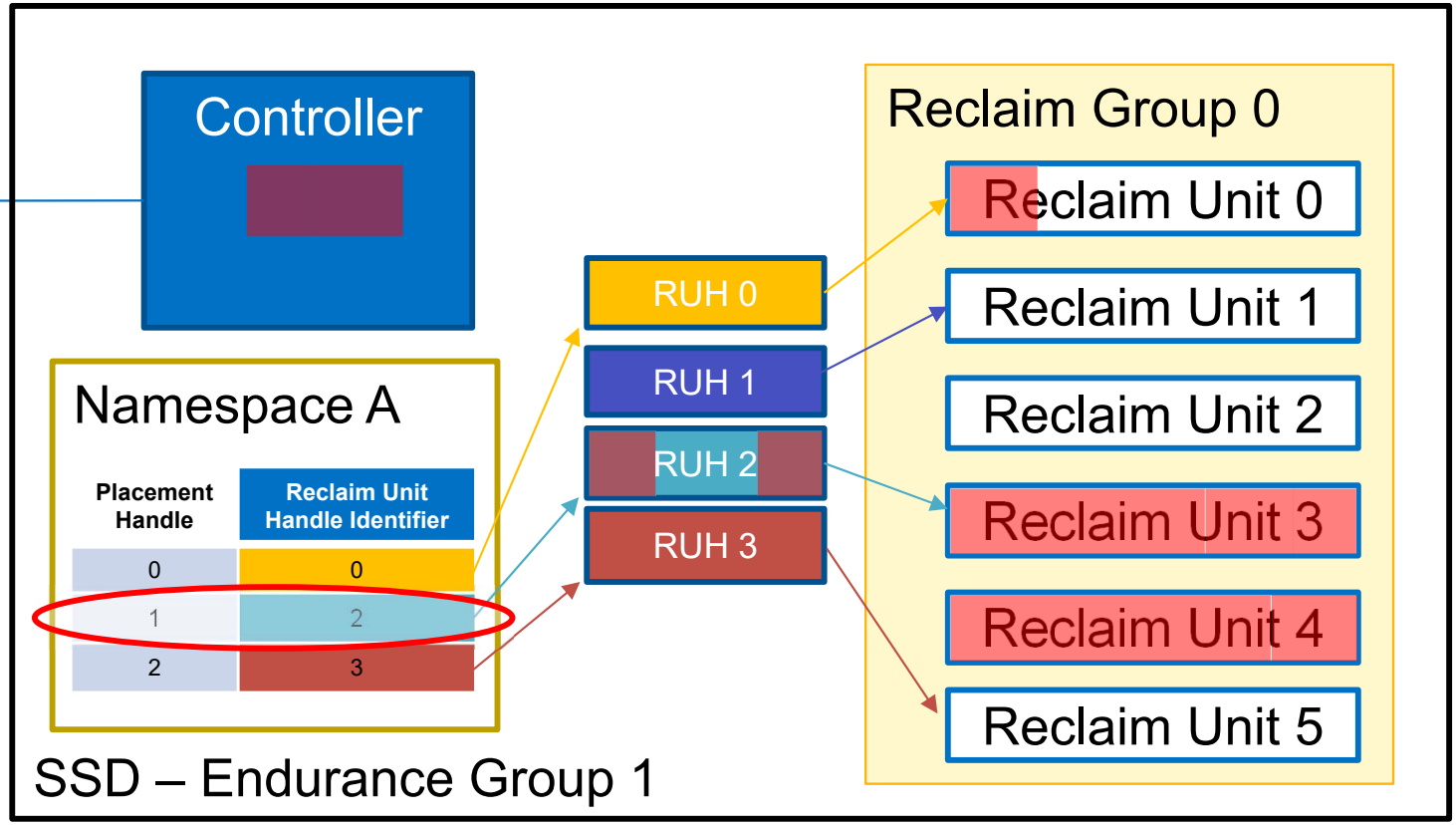
# Writing to a Namespace with FDP

Controller looks up Placement Handle 1 in Namespace A and determines to place the data using Reclaim Unit Handle 2 on Reclaim Group 0



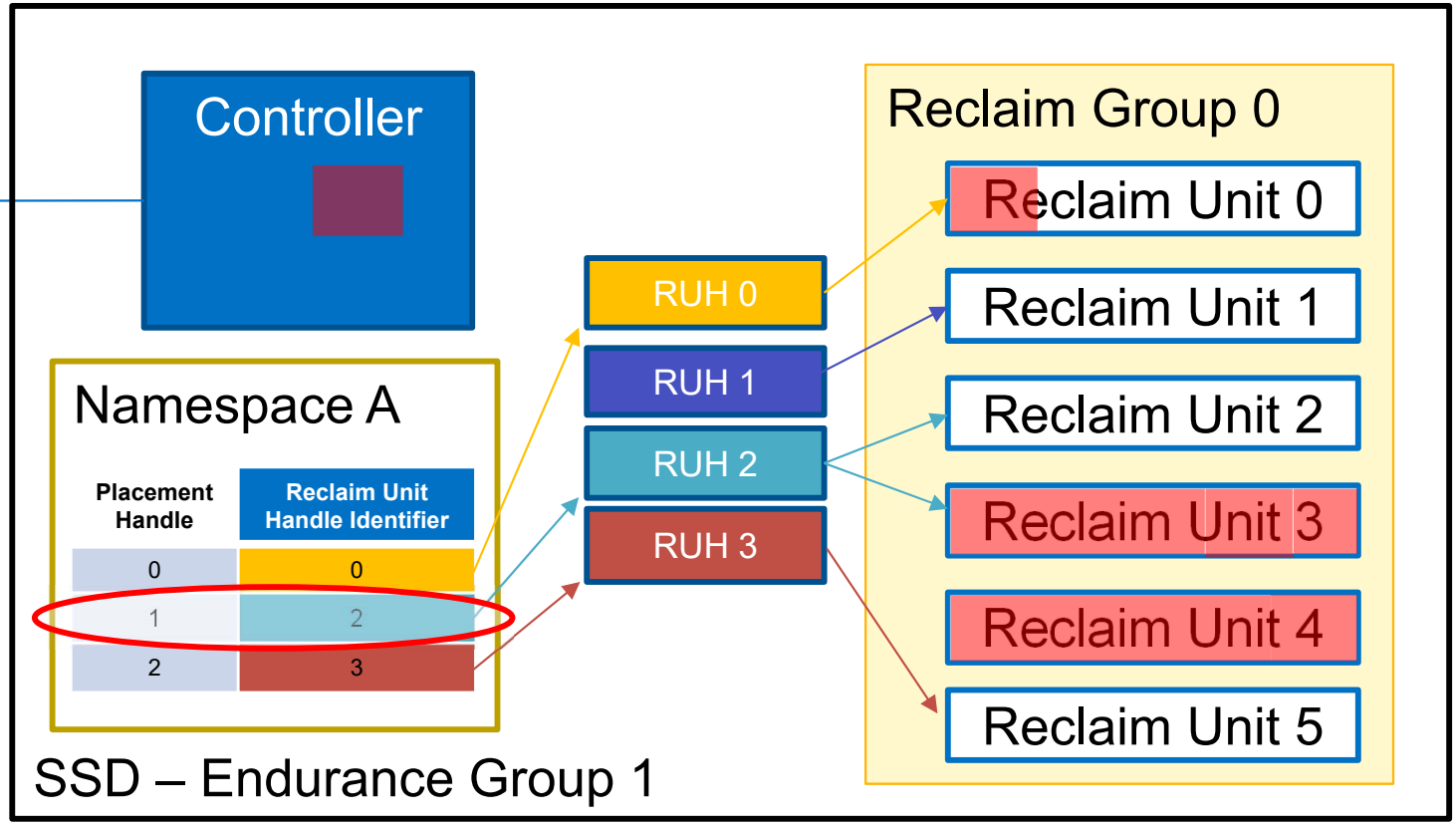
# Writing to a Namespace with FDP

Controller uses Reclaim Unit Handle 2 to places a portion of the data in Reclaim Unit 3 in Reclaim Group 0 filling Reclaim Unit 3 to capacity



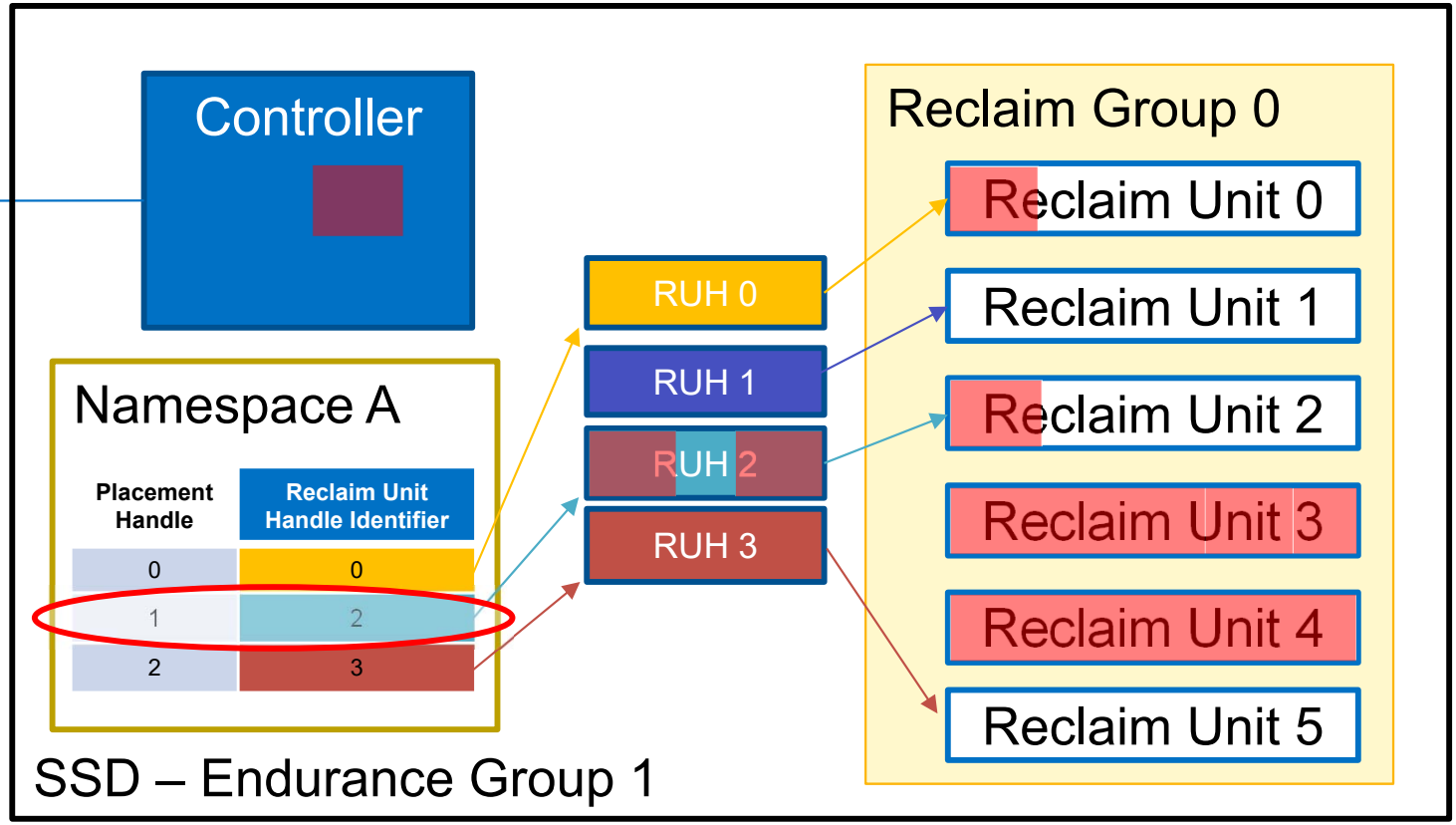
# Writing to a Namespace with FDP

Controller Reclaim Unit Handle 2 to reference the empty Reclaim Unit 2



# Writing to a Namespace with FDP

Controller uses Reclaim Unit Handle 2 to place the remaining portion of the data in Reclaim Unit 2 in Reclaim Group 0



| Streams | Flexible Data Placement | Zoned Namespaces |
|---|---|---|
| Non-conforming writes are not logged | Non-conforming writes are logged | Error on Write |
| Known alignment only after format | Commands available to host to stay aligned | Always aligned by interface rules |
| WAF = 1 achievable without feedback | WAF = 1 achievable with feedback | WAF = 1 guaranteed |
| Backwards compatible | Backwards compatible | Not Backwards compatible |
| No information that controller moved user data | Post logging that controller moved user data | Notification for Host to move user data |
| Placement identifier not tied to LBA | Placement identifier not tied to LBA | Placement identifier is the LBA |
| Stream Granularity Size (SGS) | Reclaim Units | Zones |
| SGS capacity = SGS Size | Reclaim Unit capacity = Reclaim Unit size | Zone capacity <= Zone size |
| No Host metadata per SGS | No Host metadata per Reclaim Unit | Host metadata per Zone |
| Namespace capacity defines # SGS | Endurance Group capacity defines # Reclaim Units | Namespace Capacity defines # zones |
| Sequential, Random, and Over Write | Sequential, Random, and Over Write | Sequential Write |
| Writes allowed to cross Boundaries | Writes allowed to cross Boundaries | Writes not allowed across Boundaries |
| QD > 1: LBA known at Write Submission | QD > 1: LBA known at Write Submission | QD > 1: LBA known at Write Completion<br>Zone Append command |
| Stream written by a single namespace | Reclaim Unit written by one or more namespaces | Zone written by a single namespace |
| API is Stateless | API is Stateless | API is Stateful |
| Requires Full FTL Table | Requires Full FTL Table | Full FTL Table not required |
| Dynamic write resource allocation | Static write resource allocation | Dynamic write resource allocation |

# Questions?


*Architected for Performance*