STORAGE DEVELOPER CONFERENCE
SDC 21
BY Developers FOR Developers
Virtual Conference
September 28-29, 2021

FlexAlloc: a lightweight building-block for user-space data management.

Jesper Devantier (Denmark) Samsung Electronics

Joel Granados Electronics (Denmark) Samsung

Adam Manzanares (USA) Samsung Electronics

Agenda

- 1. Context – Jesper
  - Recap the foundational shifts
  - Review options for storage applications today
- 2. Design – Joel
  - Describe the design of FlexAlloc
- 3. Demonstration – Adam

Context

How it all started...

- **Introduction**
- Along came flash storage
- Our options today

Introduction

- Start – NVMe-native write-path, minimal overhead.
  Read-only access via file system interface.
  - No free abstractions
- Insight: our storage software layers:
  - Handle many responsibilities
  - Are large, complex, well-designed
    - … but their foundations date to the era of spinning drives and the block interface abstraction.
- All-or-nothing proposition
  - Can we be compositional? Can we move faster this way?

- Introduction
- **Along came flash storage**
- Our options today

And along came flash

- Benefits
  - Higher parallelism, lower IO latency, better support for random read/write IO
- Challenges
  - Overwrite not supported
    - must erase between writes
  - Erase granularity > read/write granularity
    - Garbage-collection
  - Limited durability → wear-levelling

Hardware Changes (Along came flash storage)

- SATA →PCIe using NVMe
- Up to 64k queues with 64k commands/queue
  - Each CPU can have its own set of SQ's and associated CQ's
  - Can poll or use interrupts
- Command Sets
  - Multiple interfaces supported
  - Expanded block interfaces (zoned namespaces)
  - Alternative interfaces: KV, computational storage(*)

Software Changes (Along came flash storage)

- OS-level (Linux)
  - blk-mq
    - One SQ/CPU-core*/device
  - io_uring
    - SQ's, CQ's, shared memory
- User-space storage: SPDK
  - Message-passing > locks
  - Polling > interrupts
  - No system calls
  - Zero-copy

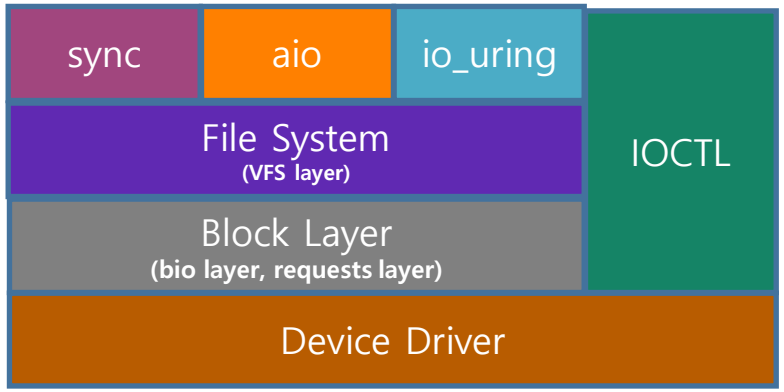
Summary (Along came flash storage)

- NAND Flash has and is profoundly impacting all layers of the storage stack
- We are seeing *a redistribution of responsibilities* across the firmware, kernel and user-space applications level
- NVMe Support multiple interfaces
  - Supports traditional and emerging storage interfaces
  - Some expand the block interface, others supplant it

- Introduction
- Along came flash storage
- **Our options today**

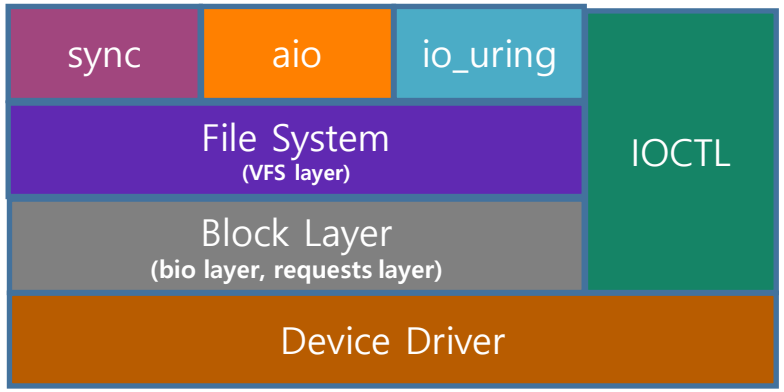
Context (Our options today)

- File system handles allocation, organizing blocks into files and directories
- IO interfaces: sync/aio/io_uring
- Device management: IOCTL, e.g.
  - libnvme, libzbd, blkdiscard, ...



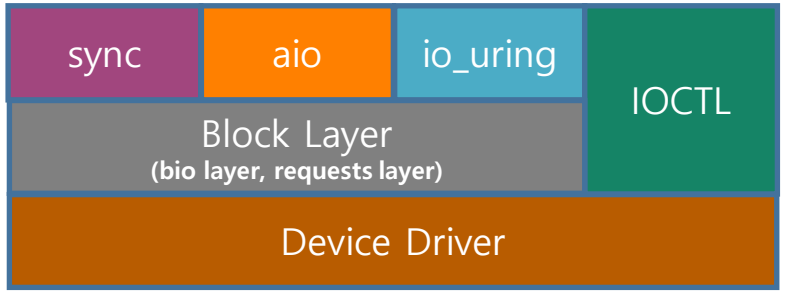
File systems (Our options today)

- General-purpose allocator
  - Files may grow/shrink arbitrarily
  - → fragmentation, metadata overhead, aging
- Provides an abstraction over the block device
  - May lack support for new developments (e.g. ZNS)



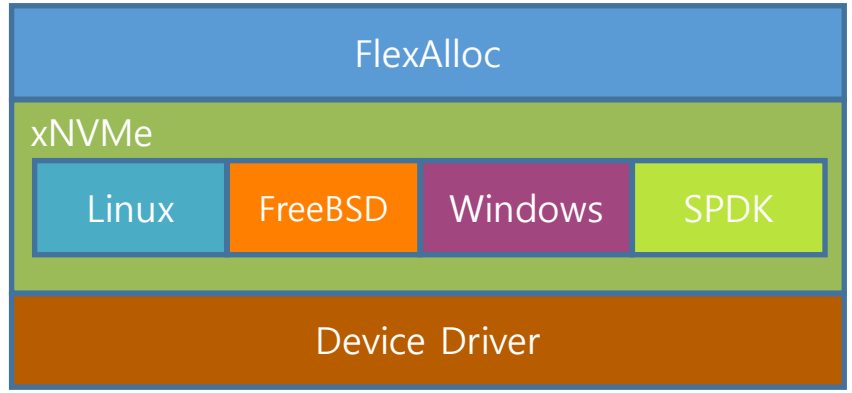
Use the raw block device? (Our options today)

- Greater control
- Must devise a data-placement scheme
- Block interface insufficient today:
  - e.g. especially ZNS devices require extensive use of specialized commands exposed as IOCTL's.
- Still required to pick appropriate IO API's depending on needs



Introducing FlexAlloc (Our options today)

- Handles allocation & data-placement
  - No other assumptions
  - Replaces the file system layer
- Object-store interface
- Supports CNS *and* ZNS devices
- User-defined object sizes – but fixed



FlexAlloc Design

FlexAlloc inside out...

- **Motivation**
- Design
- Characteristics
- Related Projects

FlexAlloc (Motivation)

- Provides versatility for top layers (xNVMe)
  - Different Backends
  - Different OSs
- Layered storage stack
  - Compression
  - Naming infrastructure
- Decisions are not assumed
  - Do one thing and do it well

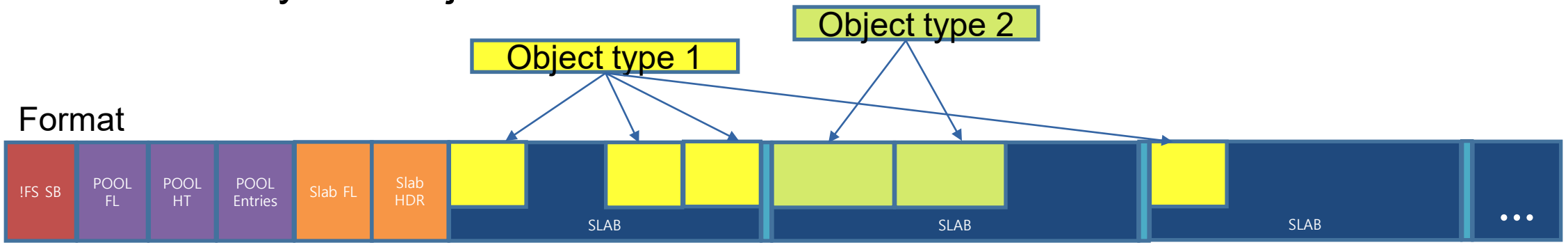
- Inspired by Bonwick's* slab allocator concepts
- Minimal metadata.

** The Slab Allocator: An Object-Caching Kernel Memory Allocator, Jeff Bonwick, Sun Microsystems*

- Motivation
- **Design**
- Characteristics
- Related Projects

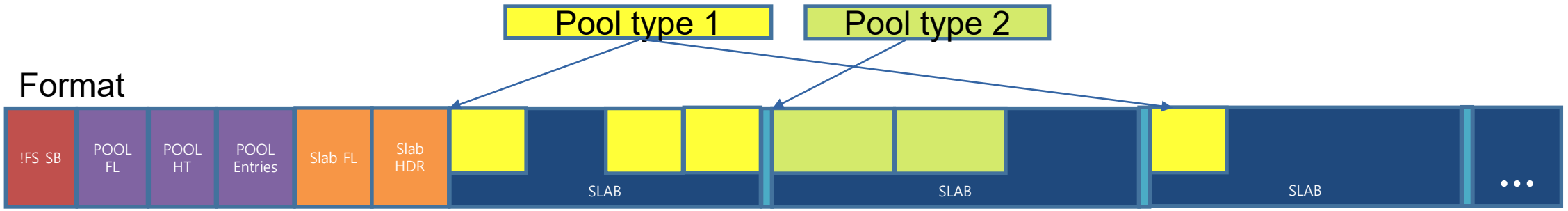
FlexAlloc (Design)

- Slabs
  - Contiguous set of logical blocks
  - Size decided at format time
  - Holds a set of objects
  - Have minimal metadata
  - Uniform segment of disk
  - Contain only one object size
- Objects
  - Contiguous sets of blocks
    - Cannot be fragmented
  - Placed into Slabs back to back
  - Gets out of the way
    - Application defined data



FlexAlloc (Design)

- Pools
  - Grouping of same objects sizes
  - Keeps track of the slabs related to each object
- Metadata
  - At the start of the device
  - A small portion in Slabs



- Motivation
- Design
- **Characteristics**
- Related Projects

FlexAlloc (Characteristics)

- Object size if fixed at creation time
  - Objects do not fragment
  - Objects cannot change max size, less metadata overhead
  - Reduces (eliminates) external fragmentation
  - Give control for internal fragmentation
- Slab size is fixed at format time
  - Know how much metadata we need for slabs
  - configured to optimally support the application's needs
- Max number of pools if fixed at format time
  - Know how much metadata we need for pools
  - configured to optimally support the application's needs

FlexAlloc (Characteristics)

- Quick object ID based lookup
- Versatile building-block for custom storage applications
- Supports CNS and ZNS devices
- Supports multiple operating systems and emerging storage backends through xNVMe
- layer for efficient data placement using an object storage interface

- Motivation
- Design
- Characteristics
- **Related Projects**

FlexAlloc (Related Projects)

- BlobStore (SPDK)
  - Designed as a thin building block
  - It is a block allocator
  - (D) Disk format is based on clusters
  - (D) Grows blobs dynamically
- BlueStore(Ceph)
  - Designed as an object store
  - Depends on RocksDB
  - Sits on top of raw block device
  - (D) More than an allocator (compresses, checksums)

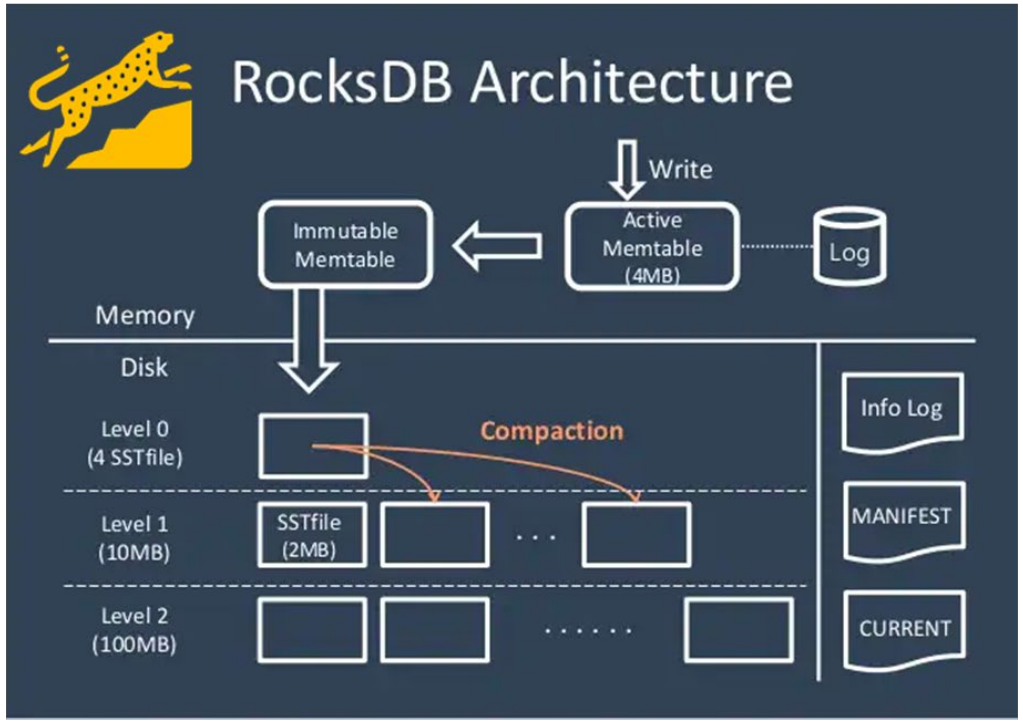
Demonstration

Apps (RocksDB) + ZNS + FlexAlloc

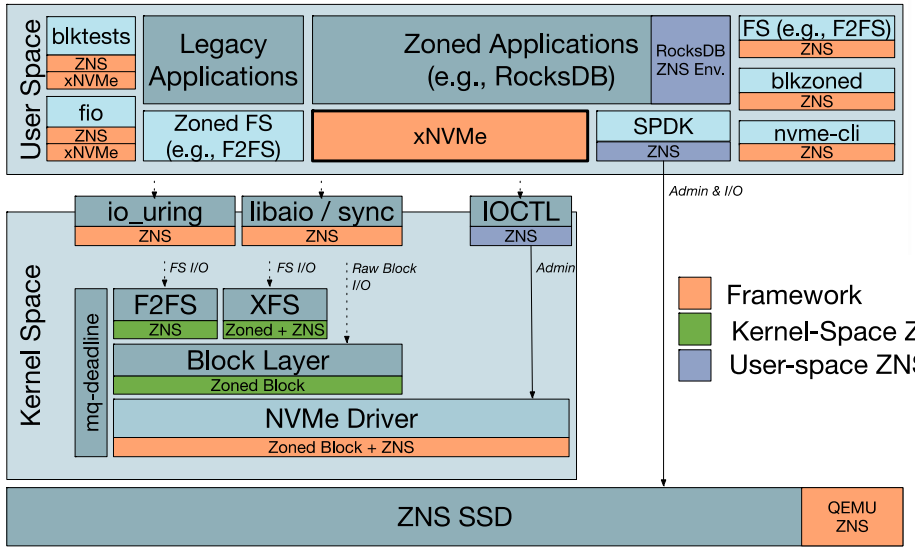
- **FlexAlloc & ZNS**
- Demo

Sequential Writes In The Wild (FlexAlloc & ZNS)

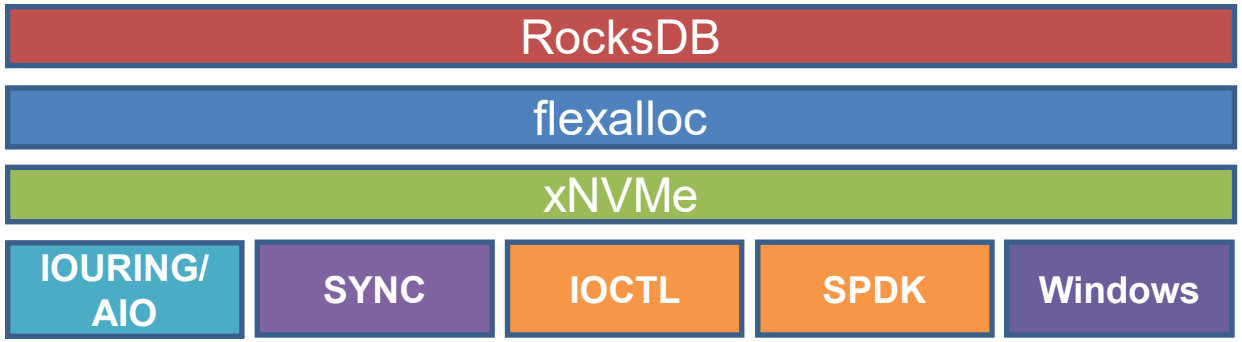
- RocksDB
  - Embeddable KV store
  - Leverages DRAM to organize writes
    - Writes KV pairs in SEQUENTIAL order SEQUENTIALY to storage
  - POSIX based storage interface (env)
    - Pluggable component
- Zoned Namespace (ZNS)
  - Extension to NVMe
  - Namespace is broken into contiguous LBA range (ZONE)
    - Mapped to SSD Dies
  - Writes must be at tail of Zone
    - SEQUENTIAL



https://www.slideshare.net/meeeejin/rocksdb-compaction



Bridge Between ZNS & RocksDB (FlexAlloc & ZNS)



- Flexalloc implements support for ZNS
  - Currently aligns slabs and objects to zones
  - Manages open/closing of zones
- RocksDB flexalloc env developed
  - Developed independently of flexalloc ZNS support
  - Minor changes to support ZNS
- Read-only fuse FS available

+

- FlexAlloc & ZNS
- **Demo**

Flexalloc DEMO

Summary

Summary

- Flexalloc – lean and flexible block allocator
  - Building block for user space file systems
  - Supports ZNS
- Leverags xNVMe
  - Provides flexibility for device access
- Rocksdb integration prototyped
  - Along with read only fuse mount

Please take a moment to rate this session.

Your feedback is important to us.

