Linux Kernel Extensions for Open-Channel SSDs

Matias Bjørling

Member of Technical Staff

The Future of “device” FTLs?

- Dealing with flash chip constrains is a necessity
  - No way around the Flash Translation Layer (FTL)
- Embedded FTLs enabled wide SSD adoption - esp. for Client computing:
  - Client: single host, single SSD, low I/O efficiency, wide variety of applications
- Server systems have a much different profile :
  - Server: multi-host, multi-SSD, high I/O efficiency, limited # of applications

The Future of “device” FTLs?

Embedded FTL’s introduce <u>significant limitations for Server compute</u>:

- Hardwire design decisions about data placement, over-provisioning, scheduling, garbage collection, and wear leveling.
- Designed on more or less explicit assumptions about the application workload.
- Introduces redundancies, missed optimizations, and underutilization of resources.

Market-Specific FTL’s: Current Solutions

Limited number of SSDs in the market with embedded FTLs for specific:

- Workloads (e.g., 90% reads)
- Applications (e.g., SQL Server, Key-value stores)

Cost and lack of flexibility for these “hard-wired” solutions is prohibitive:

- What if the workload changes (at run-time)?
- What about new workloads?
- And new applications?

Open-Channel SSD: Overview

- Open-Channel SSDs share control responsibilities with the Host in order to implement and maintain features that typical SSDs implement strictly in the device firmware

* CNEX WestLake is a commercial class Open-Channel SSD controller ASIC

Device information:

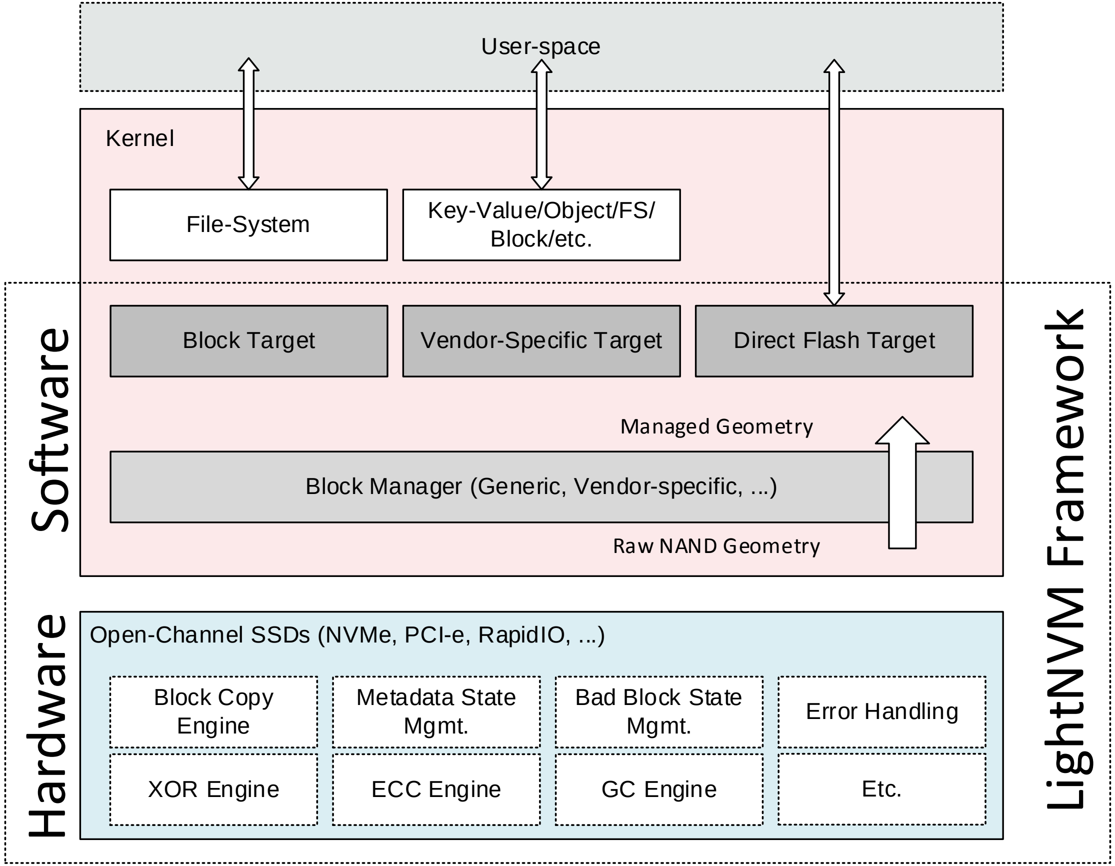
- SSD offload engines & responsibilities
- SSD geometry
  - NAND media
  - Channels, timings, etc.
  - Bad blocks list
  - ECC

Host gains:

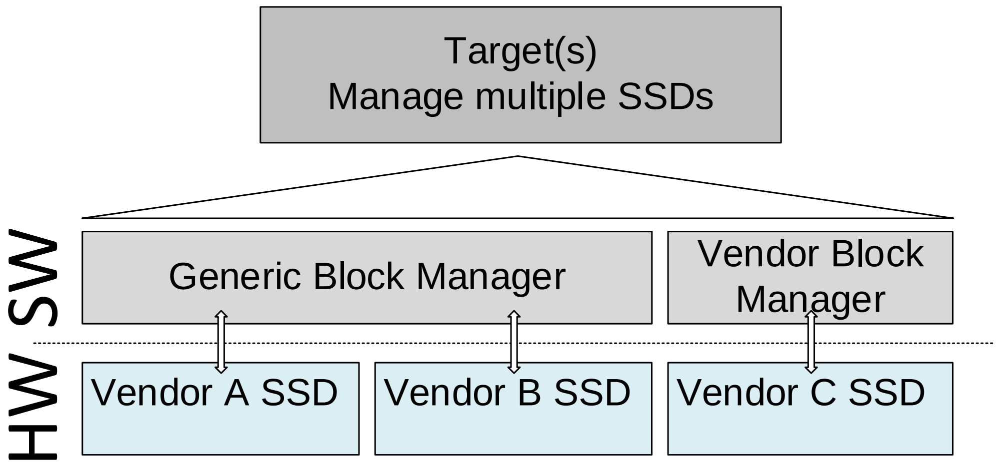
- Data placement
- I/O scheduling
- Over-provisioning
- Garbage collection
- Wear-leveling

Enables Quality of Service

Open-Channel SSD: Architecture



Open-Channel SSD: Configurability



1. Target across SSDs (with different vendor SSDs)
2. Global Garbage Collection
3. Single Address Space

BMs expose a generic interface

Same target (FTL) across SSDs. Same behavior over many SSDs

Open-Channel SSD: More Benefits

- Over-provisioning can be greatly reduced,
  - E.g., 20% lower cost for the same performance
- SSD steady state can be considerably improved
- Predictable latency
  - Reduce I/O outliers significantly

Open-Channel SSD: Host Overhead

Component	Description	Native Latency(us)		LightNVM Latency(us)	
		Read	Write	Read	Write
Kernel and fio overhead	Submission and completion (4K)	1.18	1.21	1.34 (+0.16)	1.44 (+0.23)
Completion time for devices	High-performance SSD	10us (2%)			
	Null NVMe hardware device	35us (0.07%)			
	Common SSD	100us (0.002%)			

SSD: ECC, Translation & Bad block table metadata offloaded to device.

Low overhead neglectible to hardware overhead
0.16us on reads and 0.23us on writes

Open Channel SSDs: Where are they useful?

- Software-defined storage solutions:
  - Storage is managed centrally across multiple Open-Channel SSDs
    - Petabytes of flash
  - Open-Channel SSDs are “software programmable”
    - Versus “Hardware/Firmware configurable”
    - Applications that have <u>specific</u> and/or <u>evolving</u> needs
  - Applications can define their own FTLs based on their workload
  - FTL optimizations that change over time
  - New target (e.g., Customer-specific) implementations
  - Different Application personalities:
    - Transactions, archiving, video processing, backup, vm-aware storage
  - Multi-tenancy environments

Open-Channel SSDs -> Application-driven Storage

Open Channel SSDs: Application-Driven Storage

RocksDB

ceph

PERCONA

FOUNDATIONDB

openstack™

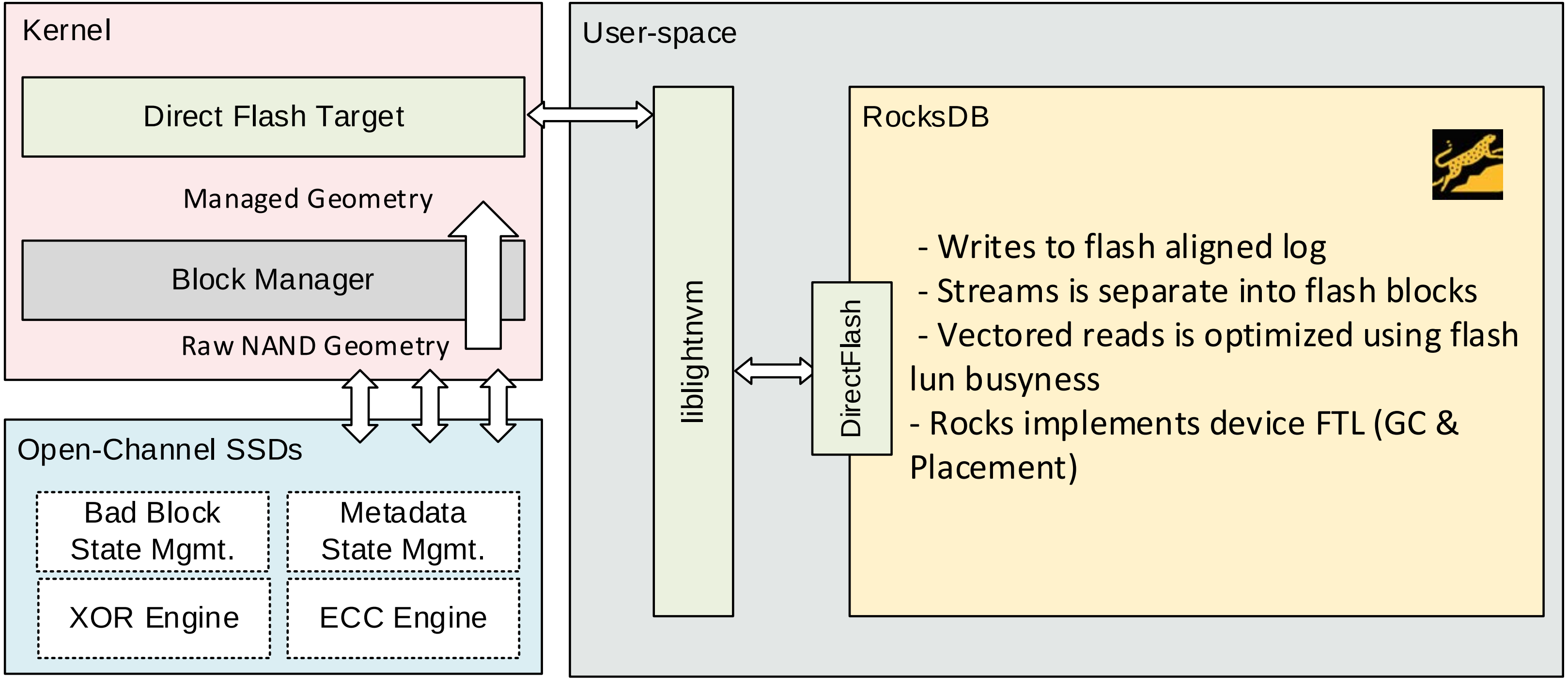
1. How do we support applications that benefit from custom FTLs?

2. What is the role of the OS in this architecture?

3. How can we hide NAND media complexity from the application (and the OS)?

- Application-Driven Storage
  - Generic interface for programmable SSDs to abstract the hardware
  - Avoid multiple layers of translation
  - Leverage optimization opportunities
  - Minimize overhead when manipulating persistent data
  - Make better decisions regarding latency, resource utilization, and data movement (compared to the best-effort techniques seen today)

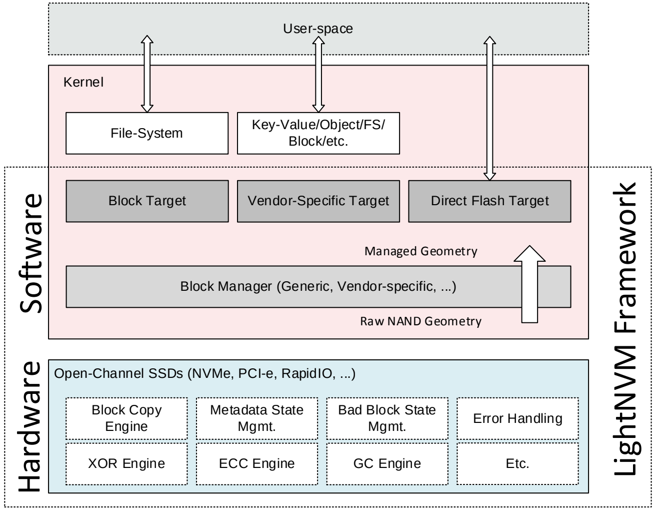
Open Channel SSDs: RocksDB Use-case



Prototype in progress

Kernel Support

- LightNVM: Linux kernel support for Open-Channel SSDs
  - Open, flexible, extensible, and scalable layer for Open-Channel SSDs for the Linux kernel
  - Development: https://github.com/OpenChannelSSD
- Supports multiple block managers and targets



Status

- Pluggable Architecture
  - Block Managers – Generic, Vendor specific,etc
  - Targets – Block, Direct Flash
- Supported drivers:
  - NVMe, Null driver (FTL performance testing and debugging)
- Push into the Linux kernel. v7 posted to LKML (7/7-15).
- Users may extend, contribute, and develop new targets for their own use-cases.
- Direct integration with RocksDB under development.

Thank you

Development: https://github.com/OpenChannelSSD/

Interface Specification: http://goo.gl/BYTjLI

Contact: Matias Bjørling
matias@cnexlabs.com