Towards Application Driven Storage

Javier González, Matias Bjørling, Philippe Bonnet

Lund Linux Con 2015
Solid State Drives

Overview

- Thousands of IOPS and low latency (<1ms)
- Hardware continues to improve
  - Parallel architecture
  - Larger flash chips
- Abstractions have allowed to rapidly replace Hard-Drives
- Complexity is handled in embedded software (FTL)
Embedded FTLs
No Future

- Dealing with flash chip constraints is a necessity
  - No way around the Flash Translation Layer (FTL)

- FTLs have enabled wide SSD adoption, but they introduce critical limitations that are visible today:
  - Hardwired data placement policies, over-provisioning, scheduling, garbage collection, wear-leveling.
  - Based on more or less explicit assumptions about the application workload.
  - Redundancies, missed optimizations, and resource underutilization
Market Specific FTLs
Workarounds Today

- SSDs on the market with embedded FTLs targeted at specific workloads (e.g., 90% reads) or applications (e.g., SQL Server, KV store)
- The FTL is optimized for a given application and workload
- But…
  - Who can afford these hardwired optimization?
  - Who can afford these hardwired optimization?
  - What about new workloads? And new applications?
What can we wish for?

RocksDB

How to engage SSD vendors (flash, chips)?

Which role does the OS play in this architecture?

Application Driven Storage

- Avoid redundancies
- Leverage optimization opportunities
- Minimize overhead when manipulating persistent data
- Make better decisions regarding latency, resource utilization, and data movement
1. Open-Channel SSDs: The kernel exposes a single address space through a host-based FTL

2. Applications can implement their own FTL and bypass the kernel for the IO flow
Open-Channel SSDs

Overview

- Host is the king:
  - Data placement
  - IO scheduling
  - Over-provisioning
  - Garbage collection
  - Weal-leveling
  - ...

- The host knows:
  - SSD geometry
    - NAND idiosyncrasies
    - Die geometry (blocks & pages)
    - Channels, timings, etc.
    - Bad blocks
    - Error-correction codes
  - Features and responsibilities
Open-Channel SSDs
Kernel Integration

- LightNVM: Linux kernel support for Open-channel SSDs
  - Development: https://github.com/OpenChannelSSD/linux

- Generic core features for host-based SSD management:
  - List of free, in-use, and bad blocks
  - Handling of flash characteristics (features and responsibilities)

- Targets that expose a logical address space
  - Can be tailored (e.g., key value stores, file systems)

- Round-robin with cost-based GC FTL (rrpc)

- Status:
  - Second patch revision posted to LKML
  - HW support: Memblaze eBlaze, a number of stealth hardware startups, IIT Mandras FPGA-based NVMe implementation.
LightNVM
Architecture

User-space

Kernel

VFS

File-systems

Page Target

Key-Value Target

Block Layer

Open-Channel SSD Integration

NVMe

PCle-based

SATA/SAS

Null device

Open-Channel SSDs
Programmable SSDs

Design Space

Requirements:
- Programming SSDs should not be reduced to adding functions on top of a generic FTL
- Programmable SSDs should contribute to streamlining the data path

Design Space:
- Programmability / Complexity
- Host-based / Embedded
Programmable SSDs

AppNVM Vision

- Applications describe declaratively the characteristics of their IO flows
- AppNVM makes applications closer to storage
  - Applications can describe their workload and communicate it to the SSD by means of rules
  - The kernel acts as a control plane and transforms this rules into SSD actions to guarantee that the expected characteristics are met with optimal resource utilization.

- Storage Quality of Service
  - Admission control combined with placement/scheduling policies managed by centralized controller
AppNVM Architecture

Data Plane

Userspace

App 2

App 1

Rule Engine

Flash LUN

Flash LUN

Network

Channels

(e.g., other SSDs)

Device

Control Plane

Kernel

Controller

Rules

Mapping Tables

Config
AppNVM

Control Plane

- A controller sets up SSD actions for a given IO flow, based on application request.

- How does the address space look like?
  - Defines how SSD state is defined and exposed to host
  - Quantizes flows of IOs

- Which actions?
  - Managing flash constraints: data placement, over-provisioning, scheduling
  - Functions to minimize data & metadata movement
AppNVM
Data Plane

- The data plane directly connects applications to storage via SSD actions
- Relies on mechanisms that bypass the OS for flows of IO: NVMe, RDMA, RapidIO
  - Millions of IOPS (throughput)
  - Predictable latency
Future Work

‣ LightNVM:
  - Get LightNVM upstream (close, but still some work to do)
  - Implement LightNVM support in more platforms. Today:
    • QEMU (Keith Busch's qemu-nvme branch)
    • A number of hardware prototypes
  - Gain traction via LightNVM and Open-Channel SSDs
    • NAND vendors, chip vendors, SSD vendors

‣ AppNVM (initial prototype)
  - Controller + rule engine embedded on SSD + set of rules
  - Application re-design based on IO flow abstraction:
    • Distributed file systems
    • DBMS storage engines (key-value and SQL)