Talk

1. Motivation
2. Physical Page Addressing (PPA) I/O Interface
3. The LightNVM Subsystem
4. pblk: A host-side Flash Translation Layer for Open-Channel SSDs
5. liblightnvm
6. RocksDB integration
7. Demonstrate the features of Open-Channel SSDs
Today’s Solid State Drives

Throughput

Intel P3608

Intel P4500

Micron P320H

Writes (GB/s)

Reads (GB/s)

Capacity

Source: Toshiba Roadmap at Flash Memory Summit 2016
Mixed I/O Workloads

- 0% writes and latency is consistent
- Larger outliers on increased writes
- 20% writes makes big impact on read latency
- 50% writes can make SSDs as slow as spinning drives...
Writes are buffered

Avoiding read and write collisions are not enough

Write Indirection & Missing State

Buffered Writes due to media writes are at page-granularity

Read/Write Interface makes Data placement + Buffering = Best Effort

Copyright © 2016 CNEX Labs | Proprietary and Confidential
Solid-State Drives and Non-Volatile Media

Transform R/W/E to R/W

Manage Media Constraints ECC, RAID, Retention

Responsibilities

Flash Translation Layer
Media Error Handling
Media Retention Management

Host Interface

Parallel Units

Media Controller

Channel X
Channel Y

Read/Write

Read/Write/Erase

NAND

Read (50-100us)
Write (1-10ms)
Erase (3-15ms)

Tens of Parallel Units!
Host Storage Data Placement Inefficiencies

Avoiding read and write collisions is not enough

Log-on-Log

1. Log-structured Database (e.g., RocksDB)
   - Metadata Mgmt.
   - Address Mapping
   - Garbage Collection
   - pread/pwrite

2. VFS

3. Log-structured File-system
   - Metadata Mgmt.
   - Address Mapping
   - Garbage Collection

4. Block Layer
   - Read/Write/Trim

5. Solid-State Drive
   - Metadata Mgmt.
   - Address Mapping
   - Garbage Collection

Log-structured implementations

Data placement information is lost = Increasing Write amplification and GC
Open-Channel SSDs

I/O Isolation
Enable I/O isolation between tenants by allocating your SSD into separate parallel units.

Predictable Latency
No more guessing when an IO completes. You know which parallel unit is accessed on disk.

Data Placement & I/O Scheduling
Manage your non-volatile memory as a block device, through a file-system or inside your application.
Open-Channel SSD

Solid-State Drive

Host Interface

Responsibilities

Flash Translation Layer

Media Error Handling

Media Retention Management

Media Controller

Parallel Units

Channel X

Channel Y
Building the Physical Page Addressing (PPA) Interface

- Expose geometry of device
  - Logical or physical representation are supported
    - Expose a set of physical LUNs as one logical or expose individually
  - Performance
    - Media Timings (Read/Write/Erase)
  - Media-specific metadata
    - Out of band size
  - Controller features
    - E.g., Device-side media buffering vs host-side media buffering

- Hierarchical Address Space
  - Encode internal geometry of the device into the address

- Read/Write/Erase
  - Vector I/Os to easily access the address space
  - Fail fast
    - In the case data are available elsewhere, no need to
Encode Geometry in Address Space

Channels -> Parallel Units -> Planes -> Blocks -> Pages -> Sectors

Solid State Drive

Media Controller

Channels:
- Channel 1
- Channel N

Parallel Units:
- PU 1..M

Planes

Blocks

Pages

Sectors

Linear Address Space

Encode Geometry into the address space

0

Max LBA -1

CH₀PU₀  CH₀PU₁  ...  CHₓPUₙ

Max LBA -1
Vector I/O Access

- Large overhead if I/Os is separately issued
- Introduce vector I/O interface to enable host to submit I/Os to multiple LUNs using one command
- Higher throughput using parallel units
- Vector Read/Write/Erase using scatter/gather LBA list
First Iteration: Version 1.2

- Released 2H2016
- Implemented as a vendor specific command set
- Vector interface Support (PPA Read/Write/Erase + Raw access without ECC applied)
- Detailed information about media data placement
  - Data placement
  - Lower/Upper page placement
  - SLC mode
  - Multi-plane operations
  - Media Scrambler control
- Media management executed on the host
- Bad block management by the host
- Results in LightNVM paper is based on this version
- Good for research, bad for production SSDs
Lessons Learned

**Warranty to end-users**
Direct access must be monitored to easily determine faulty use.

**Media characterization**
Is complex and performed for each type of NAND memory – Abstract the media to a "clean" interface.

**Write buffering**
For MLC/TLC media, write buffering is required. Decide if in host or in device.

**Application-agnostic wear leveling is mandatory**
Enable statistics for host to make appropriate decisions.
Adapted to customer requests

- Simplified
  - Log-structured. Write sequentially within block
  - Minimal write size, optimal write size.

- Media-agnostic
  - Work on NAND as well as PCM and other next generation memories

- Exposes as a traditional SSD. Reads are at sector-granularity, while particular write and erase rules must be respected.
  - Requirements to be exposed through the Zones interface introduced for SMR drives
Media Feedback

- Why a feedback channel? How?

Host

- FTL/File-Systems/Applications
- NVMe Device Driver

Open-Channel SSD

- Retention Policy
- Read Retry
- High ECC
- Wear-leveling

Read/Write

NVMe AER
Accessing an Open-Channel SSD (1)

What’s different compared to a traditional SSD?

- Geometry?
  - Presents a set of blocks (also known as zones/chunks)
  - Timings

- Reads?
  - Sector-sized reads
  - Bonus: Vector interface defines extra error codes such as reading a non-written page, and hints if a page has high ECC
Accessing an Open-Channel SSD (2)

- **Writes?**
  - Requirement: write sequential within a chunk
  - Can fail. An error does not mean the hole disk is bad. Only the particular write unit is bad.
  - Write data to next page within chunk (or skip chunk depending on error code)

- **Erases?**
  - Can fail. An error does not mean the hole disk is bad. Just means the block is bad and data should be rewritten to a new page.
LightNVM Architecture

1. NVMe Device Driver
   - Detection of OCSSD
   - Implements PPA interface
2. LightNVM Subsystem
   - Generic layer
   - Core functionality
   - Target management (e.g., pblk)
3. High-level I/O Interface
   - Block device using pblk
   - Application integration with liblightnvm
Flash Translation Layers

- **Data Placement**
  - Physical to logical address mapping (L2P)
  - Mapping table in host
  - Other metadata

- **I/O Scheduling**

- **Error handling**
  - Erase & Write errors

- **Targets today**
  - pblk (Kernel 4.12)
  - liblightnvm – RocksDB (Kernel 4.11)
pblk: Physical Block Device

- Fully associative mapping table (4KB)
- L2P recovery
- Host-side cache
  - Deal with write/read media constrains
- Cost-based GC
- Adaptive rate-limiter: user I/O vs GC I/O
- Multi-instance support
- Sysfs support: debug/configuration
- Error handling:
  - Bad block management
  - Re-mapping for grown bad blocks
pblk: Architecture

User I/O threads

- ... 
- generic_make_rq
- read
- write
- 1. Reserve space in buffer
- 2. Copy user data
- 3. Save write context
- 4. Complete I/O to block layer

GC I/O thread

Respect Media Constrains

Ring Write Buffer

Context
User Data

Configurable Mapping Strategy

Submission Path

1. Map buffer L2P in current line
   - Update L2P table on wrap-up
2. Map metadata for previous line
3. Map erase for next line
4. Submit I/O set

Completion Path

1. Update buffer pointers
2. Deal with W/E errors

Open-Channel SSD

Line 0
Line 1
Line 2
Line N

P0  P1  P2  PN

Metadata

L2P Lookup
pblk: Data Placement

- Fully associative L2P table
  - 4KB granularity (1GB per 1TB)

- Pre-populated bitmap encoding map
  - Bitmap encodes bad blocks and metadata
  - Save expensive calculations on fast path (division/modulus)

- L2P mapping is decoupled from I/O scheduling
  - Simple to add new mapping strategies
  - Simplifies error handling
  - Does not necessarily affect disk format
  - Default:
    - Stripe across channels and LUNs to optimize for throughput
    - Metadata at beginning and ending of each line
pblk: I/O Scheduling

- **Goals**
  - Fully utilize the bandwidth of the media
    - 1 core (E5-2620, 2.4GHz) can move ~3.7GB/s
  - Minimize impact of reaching steady state
  - Rate-limit user and GC I/O according to the device’s capacity

- **Single write thread**
  - Submits user write I/Os as buffer entries are mapped
  - Submits write I/Os for previous line metadata
    - Align with user data to minimize disturbances
  - Submits erase I/Os for next line
    - Align with user data to minimize disturbances
    - Distribute price of erasing across all lines
pblk: Garbage Collection

- Cost-based recycling mode based on valid sector count
- Dedicate X LUNs in a line for GC
  - Rate-limit user I/O as a function of device’s capacity
  - GC dedicated write buffer (share percentage of the buffer capacity)
  - Separate hot/cold data on a line
- Two GC modes are available:
  - Move data using the host’s CPU
    - Several GC read thread
    - Copy data to the GC buffer
    - Submit GC I/O in write thread
  - Optional: Use vector copy command
    - Move data using the controller
pblk: Recovery

- Per line metadata

- smeta
  - Mark line as “open” when it is allocated
  - Give line a sequence number
  - Create a reverse line list
  - Store the LUNs forming the line
  - Store active write LUNs

- emeta
  - Replicate smeta for consistency
  - Store updated bad block bitmap for line
  - Store L2P portion for line (lba list)
  - Store valid sector count (VSC) for all lines

- Per page metadata
  - 16 bytes per 4KB
  - Store lba mapped to 4KB sector in OOB area (8 bytes)

- Recovery: Scan all lines and reconstruct L2P in order - first closed lines, then open line
Experimental Evaluation

- CNEX Labs Open-Channel SSD
  - NVMe
  - Gen3x8
  - 2TB MLC NAND

- Geometry
  - 16 channels
  - 8 PUs per channel (Total: 128 PUs)

- Parallel Unit Characteristics
  - Page size: 16K + 64B user OOB
  - Planes: 4, Blocks: 1,067, Block Size: 256 Pages

- Performance:
  - Write: Single PU 47MB/s
  - Read: Single 108MB/s, 280MB/s (64K)

Evaluation
- Base
- Interface Flexibility
  - Limit # Active Parallel Write Units
  - Predictable Latency
  - Multi-Tenant Workloads
Base Performance

Throughput & Latency

Increases with parallelism

RR slightly lower due to scheduling conflicts

Request I/O Size

Throughput (GB/s)

Latency (ms)
A priori knowledge of workload. E.g., limit to 400MB/s Write
Limit number of Active PU Writers, and achieve better read latency

- Single Read or Write Perf.
- Mixed Read/Write
- 256K Write QD1
- 256K Read QD16

Read Perf. at 200MB/s
Write latency increases, and read latency reduces
Predictable Latency

- 4K reads during 64K concurrent writes
- Consistent low latency at 99.99, 99.999, 99.9999
pblk: Multi-tenant Isolation

- Mitigate noisy neighbor effect on different pblk instances
- Scales with number of channels
  - 16 channels in HHHL LightNVM SDK
- Does not require changes to the application
**liblightnvm**

- User space library enabling direct access to OCSSD from user-space
- BSD license – compatible with open/closed source LightNVM implementations
- 2 Interfaces
  - Command line:
    - Direct I/O submission
    - Supports tools to manage OCSSD
  - C API: Allows applications to implement their own FTL
    - Query OCSSD geometry
    - I/O submission: Both vector I/O and scalar I/O (SMR-like)
RocksDB Architecture with liblightnvm

- Use RocksDB LSM as the FTL
  - Tree nodes (files) control data placement on physical media
  - Sstables, WAL, and MANIFEST on Open-Channel SSD - rest in normal FS (e.g., pblk)
  - Garbage collection takes place during LSM compaction – no extra movement

- LightNVM interfaces OCSSD
  - Provides an API for applications
  - Divide geometry through ICTLs or sysfs
  - I/O submission through IOCTLs

- In-progress work
Conclusion

- New interface that provides
  - I/O Predictability
  - I/O Isolation
  - Puts the host in the front seat of data placement and I/O scheduling

- PPA Specification is open and available for implementors

- Active community using OCSSDs both for production and research
  - Multiple drives in development within SSD vendors
  - Multiple papers already on Open-Channel SSDs that shows how this interface can improve workloads

- Fundamental building blocks are available:
  - Initial release in Linux kernel 4.4.
  - User-space library (liblightnvm) support with Linux kernel 4.11.
  - Pblk will be upstream with Linux kernel 4.12.

- The right time to dive into Open-Channel SSDs
  - More information available at: http://lightnvm.io
Backup Slides
RocksDB & OLTP/OLAP

- Traditional block device using pblk.
- Reducing Active Writers -> Reduce latency spikes
- OLTP/OLAP: Similar TPS and Latency due to being CPU bound
**Geometry-encoded Address Space**

- Encode geometry in address space
  - Reserve X bits for each level
  - Non-power of two configuration has "holes"
  - Another approach is to lay out blocks back-to-back. At the cost of extra calculations.

- Example address space (128 Pus – total 29 bits)

<table>
<thead>
<tr>
<th>Level</th>
<th>2TB &quot;Physical&quot; NAND SSD</th>
<th>2TB &quot;Logical&quot; NAND SSD</th>
<th>2TB &quot;Logical&quot; PCM SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels</td>
<td>16 -&gt; 4 bits</td>
<td>16 -&gt; 4 bits</td>
<td>16 -&gt; 4 bits</td>
</tr>
<tr>
<td>PUs/LUNs</td>
<td>8 -&gt; 3 bits</td>
<td>8 -&gt; 3 bits</td>
<td>2 -&gt; 1 bit</td>
</tr>
<tr>
<td>Planes</td>
<td>2 - &gt; 1 bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blocks</td>
<td>1024 -&gt; 10 bits</td>
<td>1024 -&gt; 10 bits</td>
<td></td>
</tr>
<tr>
<td>Pages</td>
<td>512 -&gt; 9 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sectors (4K)</td>
<td>4 -&gt; 2 bits</td>
<td>4096 -&gt; 12 bits</td>
<td>16777216 -&gt; 24 bits</td>
</tr>
</tbody>
</table>

Logical Block Address (LBA)

<table>
<thead>
<tr>
<th>Channel</th>
<th>PU/LUN</th>
<th>Block</th>
<th>Plane</th>
<th>Page</th>
<th>Sector</th>
</tr>
</thead>
</table>

Physical Page Address (NAND)

<table>
<thead>
<tr>
<th>Channel</th>
<th>PU/LUN</th>
<th>Block</th>
<th>Page</th>
<th>Sector</th>
</tr>
</thead>
</table>

Physical Page Address (NAND with Sector-sized Pages)

<table>
<thead>
<tr>
<th>Channel</th>
<th>PU/LUN</th>
<th>Block</th>
<th>Sector</th>
</tr>
</thead>
</table>

Physical Page Address (PCM with Sector-sized Pages)

<table>
<thead>
<tr>
<th>Channel</th>
<th>PU/LUN</th>
<th>Sector</th>
</tr>
</thead>
</table>

MSB | LSB
Geometry Identification

- **Physical geometry**
  - Address format, #Channels, #LUNs, #Blocks per LUN, Minimal Write Size, Optimal Write Size, ...

- **Performance**
  - Read/Write/Erase Timings – For predictability calculation

- **Media-specific metadata (if needed)**
  - Distance between shared pages (MLC -> Lower/Upper, TLC -> Lower/Middle/Upper)

- **Controller functionalities**
  - Vector copy, Write/Erase Suspend, ...
Media Failure Opportunities

- **Bit errors**
  - Increase in bit flips as MLC/TLC/QLC due to retention, pe cycles, and media quality. SSD performs error recovery up to a given error rate guarantee (e.g., $10^{-17}$)

- **Read disturbs**
  - Multiple reads to the same cell leaks current to nearly cells. Cause error rates to go up for nearby cells. SSD may provide an event when a single sector or a group of sectors must be refreshed.

- **Write/Erase error**
  - Error may be reported from media when programming/erasing. User must take appropriate actions

- **Die failure.**
  - Device or host recovery (as in traditional storage systems)
Access Constrains

- One operation concurrent per LUN (Read/Write/Erase)
- Erase is at block granularity
- Write is a page granularity
- Read is at sector granularity
- Writes
  - Sequential within a block (or media specific order)
  - If media is MLC/TLC. Lower/Middle pages cannot be read before their upper counter parts has been written. Host assumes that reads can occur right after a write. A write buffer is required.
CNEX Labs, Inc.

Teaming with NAND Flash manufacturers and industry leaders in storage and networking to deliver the next big innovation for solid-state-storage.