Persistent Memory and Linux: New Storage Technologies and Interfaces

Ric Wheeler Senior Manager Kernel File and Storage Team Red Hat, Inc.



Overview

- What is Persistent Memory
- Challenges with Existing SSD Devices & Linux
- Crawl, Walk, Run with Persistent Memory Support
- Getting New Devices into Linux
- Related Work



What is Persistent Memory?



Persistent Memory

- A variety of new technologies are coming from multiple vendors
- Critical feature is that these new parts:
 - Are byte addressable
 - Do not lose state on power failure
- Critical similarities are that they are roughly like DRAM:
 - Same cost point
 - Same density
 - Same performance



Similarities to DRAM

- If the parts are the same cost and capacity of DRAM
 - Will not be reaching the same capacity as traditional, spinning hard drives
 - Scaling up to a system with only persistent memory will be expensive
 - Implies a need to look at caching and tiered storage techniques
- Same performance as DRAM
 - Will press our IO stack which already struggles to reach the performance levels of current PCI-e SSD's

Challenges with Existing SSD Devices & Linux





The Linux I/O Stack Diagram (version 1.0, 2012-06-20) http://www.thomas-krenn.com/en/oss/linux-io-stack-diagram.html Created by Werner Fischer and Georg Schönberger License: CC-BY-SA 3.0, see http://creativecommons.org/licenses/by-sa/3.0/

Early SSD's and Linux

- The earliest SSD's look like disks to the kernel
 - Fibre channel attached high end DRAM arrays (Texas Memory Systems, etc)
 - S-ATA and SAS attached FLASH drives
- Plugged in seamlessly to the existing stack
 - Block based IO
 - IOP rate could be sustained by a well tuned stack
 - Used the full block layer
 - Used a normal protocol (SCSI or ATA commands)

PCI-e SSD Devices

- Push the boundaries of the Linux IO stack
 - Some devices emulated AHCI devices
 - Many vendors created custom drivers to avoid the overhead of using the whole stack
- Performance challenges
 - Linux block based IO has not been tuned as well as the network stack to support millions of IOPS
 - IO scheduling was developed for high latency devices



Performance Limitations of the Stack

- PCI-e devices are pushing us beyond our current IOP rate
 - Looking at a target of 1 million IOPS/device
- Working through a lot of lessons learned in the networking stack
 - Multiqueue support for devices
 - IO scheduling (remove plugging)
 - SMP/NUMA affinity for device specific requests
 - Lock contention
- Some fixes gain performance and lose features

Tuning Linux for an SSD

	Terminal		×
File Edit View Search	Terminal Help		
bash-4.2\$ ls /sys/bl add_random discard_granularity discard_max_bytes discard_zeroes_data hw_sector_size iosched iostats logical_block_size bash-4.2\$	ock/sda/queue/ max_hw_sectors_kb max_integrity_segments max_sectors_kb max_segments max_segment_size minimum_io_size nomerges nr_requests	optimal_io_size physical_block_size read_ahead_kb rotational rq_affinity scheduler write_same_max_bytes	

- Take advantage of the Linux /sys/block parameters
 - Rotational is key
 - Alignment fields can be extremely useful
 - http://mkp.net/pubs/storage-topology.pdf
- Almost always a good idea not to use CFQ

Device Driver Choice

- Will one driver emerge for PCI-e cards?
 - NVMe: http://www.nvmexpress.org
 - SCSI over PCI-e: http://www.t10.org/members/w_sop-.htm
 - Vendor specific drivers
 - Most Linux vendors support a range of open drivers
- Open vs closed source drivers
 - Linux vendors have a strong preference for open source drivers
 - Drivers ship with the distribution no separate installation
 - Enterprise distribution teams can fix code issues directly

Challenges Cross Multiple Groups

- Developers focus in relatively narrow areas of the kernel
 - SCSI, S-ATA and vendor drivers are all different teams
- Block layer expertise is a small community
- File system teams per file system
- Each community of developers spans multiple companies



Crawl, Walk, Run



Persistent Memory & Byte Aligned Access

- DRAM is used to cache all types of objects file system metadata and user data
 - Moving away from this model is a challenge
 - IO sent in multiples of file system block size
 - Rely on journal or btree based updates for consistency
 - Must be resilient over crashes & reboots
 - On disk state is the master view & DRAM state differs
- These new devices do not need block IO



Crawl Phase

- Application developers are slow to take advantage of new hardware
 - Most applications will continue to use read/write "block oriented" system calls for years to come
 - Only a few, high end applications will take advantage of the byte addressable capabilities
- Need to hide the persistent memory below our existing stack
 - Make it as fast and low latency as possible!



Crawling Along

- Block level driver for persistent memory parts
 - Best is one driver that supports multiple types of parts
- Enhance performance of IO path
 - Leverage work done to optimize stack for PCI-e SSD's
 - Target: millions of IOP's?
- Build on top of block driver
 - Block level caching
 - File system or database journals?
 - As a metadata device for device mapper, btrfs?



Crawl Status

- Block drivers being developed
- Block caching mechanisms upstream
 - Bcache from Kent Overstreet in 3.10
 - http://bcache.evilpiepirate.org
 - A new device mapper dm-cache target
 - Modular policy allows anyone to write their own policy
 - Reuses the persistent-data library from thin provisioning
- PMDB block device from Intel Research for persistent memory
 - https://github.com/linux-pmfs

Walk Phase

- Modify existing file system or database applications
 - Can journal mechanism be reworked to avoid barrier operations and flushes?
 - Dynamically detect persistent memory and use for metadata storage (inodes, allocation, etc)?
- Provide both block structured IO and byte aligned IO support
 - Block for non-modified applications
 - Memory mapped accessed for "power" applications

Walk Phase Status

- File systems like btrfs have looked at dynamically steering load to SSDs
 - More based on the latency than on persistence
- Looking at MMAP interface to see what changes are needed
- Need to educate developers about the new parts & get parts in community hands



Run Phase

- Develop new APIs at the device level for consumption by file and storage stack
- Develop new APIs and a programming model for applications
- Create new file systems that consume the devices API's and provide the application API
 - Nisha's talk on NVM interfaces Wed at 3:00PM
- Modify all performance critical applications to use new APIs



Run Status

- Storage Network Industry Association (SNIA) Working Group on NVM
 - Working on a programming model for this class of parts
 - http://snia.org/forums/sssi/nvmp
- Several new file systems for Linux being actively worked on
 - Intel Research;s PMFS at https://github.com/linux-pmfs/pmfs.git
 - FusionIO's directFS based on atomic writes
- Might be as painful as multi-threading or teaching applications to use fsync()!

Getting New Devices into Linux



What is Linux?

- A set of projects and companies
 - Various free and fee-based distributions
 - Hardware vendors from handsets up to mainframes
 - Many different development communities
- Can be a long road to get a new bit of hardware enabled
 - Open source allows any party to write their own file system or driver
 - Different vendors have different paths to full support
 - No single party can get your feature into distributions

Not Just the Linux Kernel

- Most features rely on user space components
- Red Hat Enterprise Linux (RHEL) has hundreds of projects, each with
 - Its own development community (upstream)
 - Its own rules and processes
 - Choice of licenses
- Enterprise Linux vendors
 - Work in the upstream projects
 - Tune, test and configure
 - Support the shipping versions

The Life Span of a Linux Enhancement

- Origin of a feature
 - Driven through standards like T10, SNIA or IETF
 - Pushed by a single vendor
 - Created by a developer or at a research group
- Proposed in the upstream community
 - Prototype patches posted
 - Feedback and testing
 - Advocacy for inclusion
- Move into a community distribution like Fedora
- Shipped and supported by an enterprise distribution

The Linux Community is Huge

- Most active contributors in 3.8 kernel (changesets):
 - No affiliation 12.8%
 - Red Hat 9.0%
 - Intel 8.7%
 - Unknown 7.4%
 - LINBIT 4.8%
 - Linaro 4.6%
 - Texas Instruments 4.0%
 - Vision Engraving Systems 3.5%
 - Samsung 3.3%
 - SUSE 2.5%
- Statistics from: http://lwn.net/Articles/536768

Linux Storage & File & MM Summit 2013



Resources & Questions

- Resources
 - Linux Weekly News: http://lwn.net/
 - Mailing lists like linux-scsi, linux-ide, linux-fsdevel, etc
- SNIA NVM TWG
 - http://snia.org/forums/sssi/nvmp
- Storage & file system focused events
 - LSF workshop
 - Linux Foundation & Linux Plumbers Events

