Taking Linux File and Storage Systems into the Future

Ric Wheeler
Director
Kernel File and Storage Team
Red Hat, Incorporated

Overview

- Going Bigger
- Going Faster
- Support for New Hardware
- Current Areas of Focus
- Resources & Questions

Going Bigger

Storage and Servers Continue to Grow

- File system need to support ever larger storage devices
  - Individual S-ATA disks are now 6TB
  - New Shingled (SMR) drives will be even larger!
- Storage arrays, hardware RAID cards and software LVM combine drives into an even larger block device
  - Normal shelf of drives is 12 drives
  - Allows 10 data drives (with 2 parity) for RAID6
  - 40-60 TB per shelf!

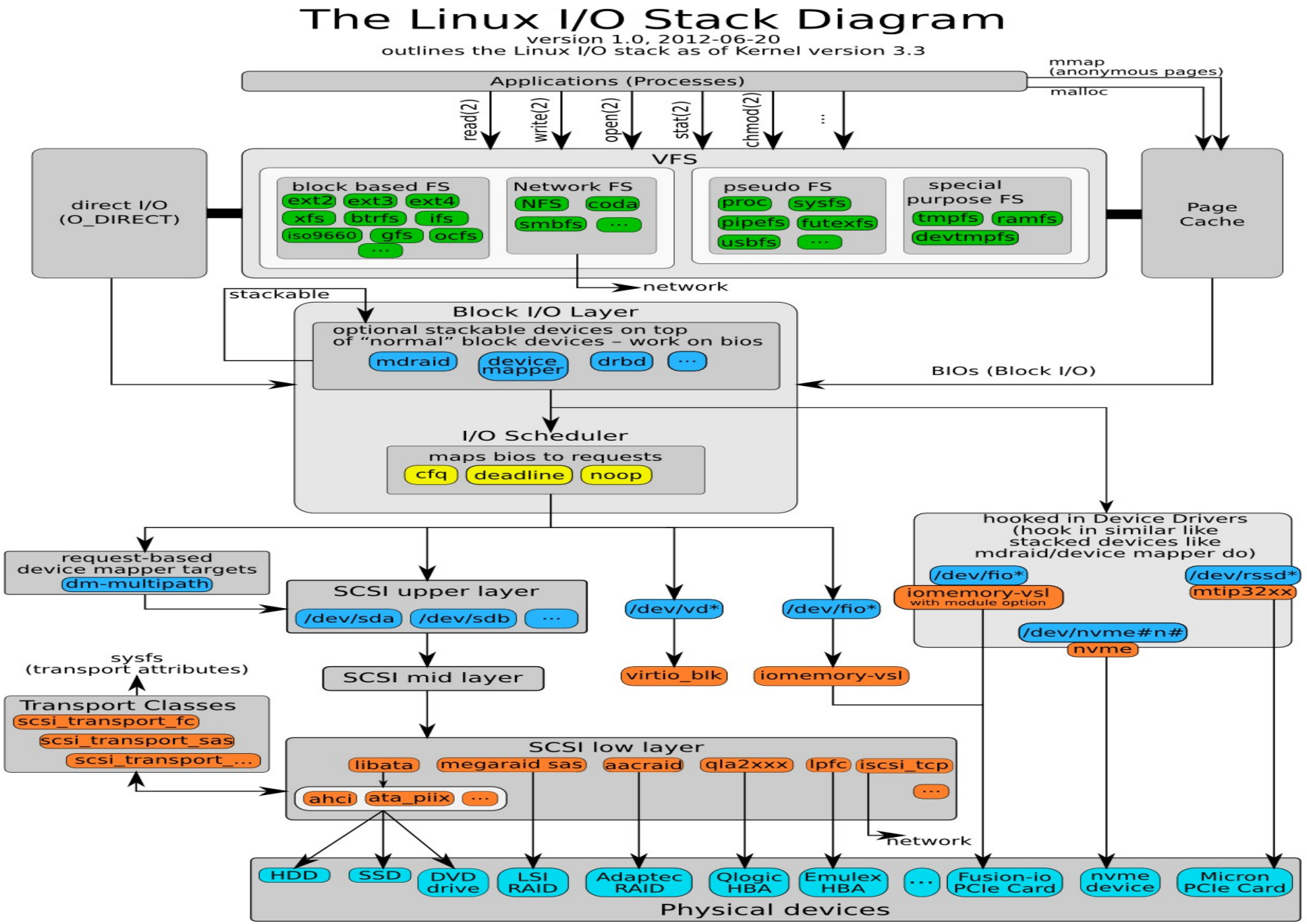
Why Use a Single, Large File System?

- A single file system is easy for users and applications
  - Space is in a common pool
- A single file system can perform better than multiple file systems
  - Rotating storage must minimize disk head movements
  - Carving up a single S-ATA drive or RAID set with S-ATA makes disk heads jump between file systems

Challenges with a single file system

- System operations take a lot longer
  - Backup and restore scale with the size
  - File system repair can take a very long time
- Larger file systems can require larger servers
  - Doing a file system repair on a 100TB file system pulls in a lot of metadata into DRAM
  - Must use servers with sufficient DRAM to prevent paging
- Metadata can be a high overhead
  - Keeping size of structures down is critical when talking about millions or billions of files per file system!

Going Faster



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

Block Level Caching Schemes

- Bcache from Kent Overstreet
  - http://bcache.evilpiepirate.org
- A new device mapper dm-cache target
  - Simple cache target can be a layer in device mapper stacks.
  - Modular policy allows anyone to write their own policy
  - Reuses the persistent-data library from thin provisioning
- Vendor specific caching schemes

Support for New Hardware

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 reach 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
  - IO performance scales with the number of parts
  - Will press our IO stack to reach the maximum performance of PM

Persistent Memory & Byte Aligned Access

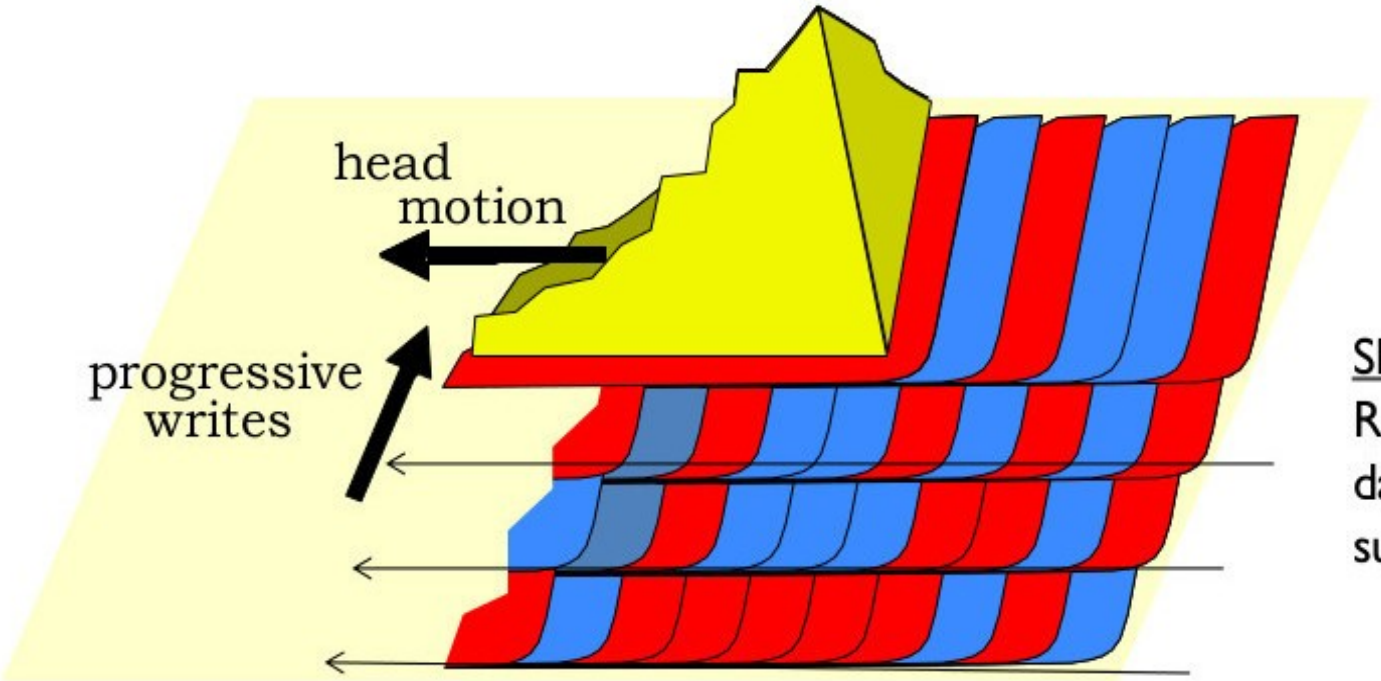
- DRAM is used to cache all types of objects – file system metadata and user data
  - Moving away from this model is a challenge
  - IO sent in multiples of file system block size
  - Rely on journal or btree based updates for consistency
  - Must be resilient over crashes & reboots
  - On disk state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

SMR Overview

- A new areal density enabling technology called Shingled Magnetic Recording (SMR)
  - Industry vendors are working collaboratively on external interfaces
  - Vendors will differentiate on implementations
- SMR alters throughput and response time
  - Especially for random write IO
- Industry is looking for feedback from the Linux community on T10 proposals

What is Shingled Magnetic Recording (SMR)?

SMR write head geometry extends well beyond the track pitch in order to generate the field necessary for recording. Tracks are written sequentially in an overlapping manner forming a pattern similar to shingles on a roof.



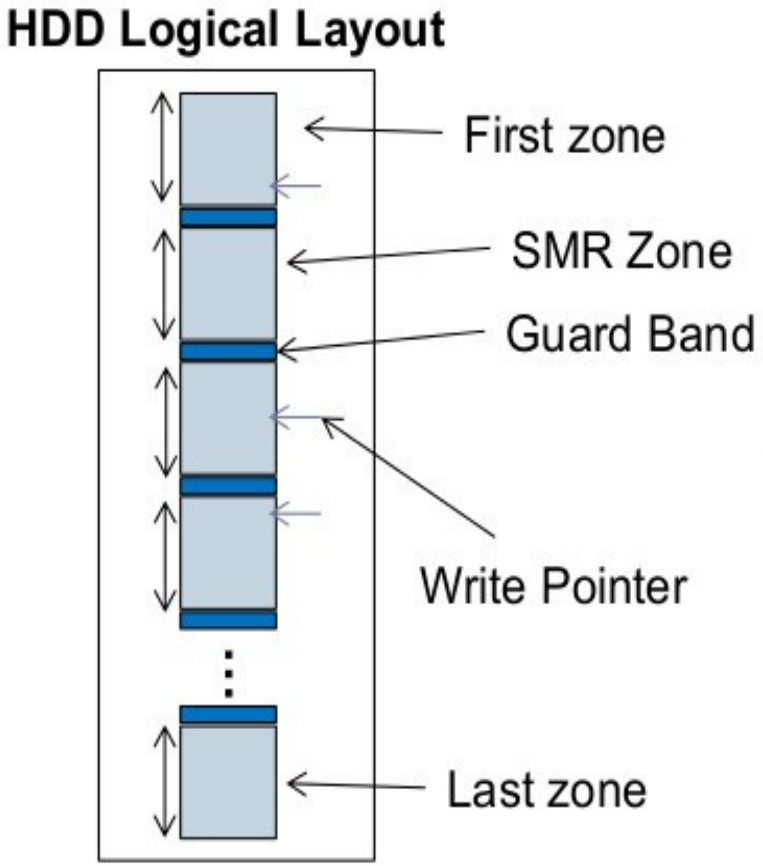
SMR Constraint:
Rewriting a given track will damage one or more subsequent tracks.

Wood, Williams, et al., IEEE TRANSACTIONS ON MAGNETICS, VOL. 45, NO. 2, FEBRUARY 2009

SMR Drive Write Bands

- Random write enabled bands
  - Might not exist at all on some implementations
  - Could be first and last band
  - Place to store metadata, bitmaps, etc
- Sequential write bands
  - Can be written only in order
  - Write pointer is tracked per band
  - Full band reset done when a band's data is all stale

SMR Interface Principles



- SMR drives may be divided into zones: Random Write and SMR
- A Random Write zone is a range of LBAs that may be written randomly
  - The device provides media in this area that is optimized for random writing.
  - This type of zone may raise the cost of the device or lower the overall capacity of the device
- An SMR zone is a zone that should be written sequentially
  - The drive maintains a write pointer for each zone
    - If the initiator does not write starting at the write pointer, ***then the SMR zone rules have been violated***
  - The drive may provide a mechanism to retrieve the write pointer
    - Important for surprise power removal and data recovery operations
    - May be a part of the zone map described below
  - The device provides a mechanism to reset the write pointer to the beginning of the zone

Current Area of Focus

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

Scaling Up File Systems

- Support for metadata checksumming
  - Makes file system repair and corruption detection easier
- Support for backpointers
  - Btrfs can map sector level errors back to meaningful objects
  - Let's users turn map an IO error into knowledge about a specific file for example

Making BTRFS Ready for Enterprise Users

- Slowing the inclusion of new features
  - Focus on bug fixing and performance enhancements
  - Fixing static analysis reported bugs
- Chris Mason is releasing a new, more powerful version of the btrfs user space tools
- Extensive enterprise vendor testing
  - Focus on most promising use cases

Ease of Use

- Linux users have traditional been given very low level tools to manage our storage and file systems
  - Very powerful and complicated interface
  - Well suited to sophisticated system administrators
- Too complicated for casual users
  - Exposes too much low level detail
  - User must manage the individual layers of the stack

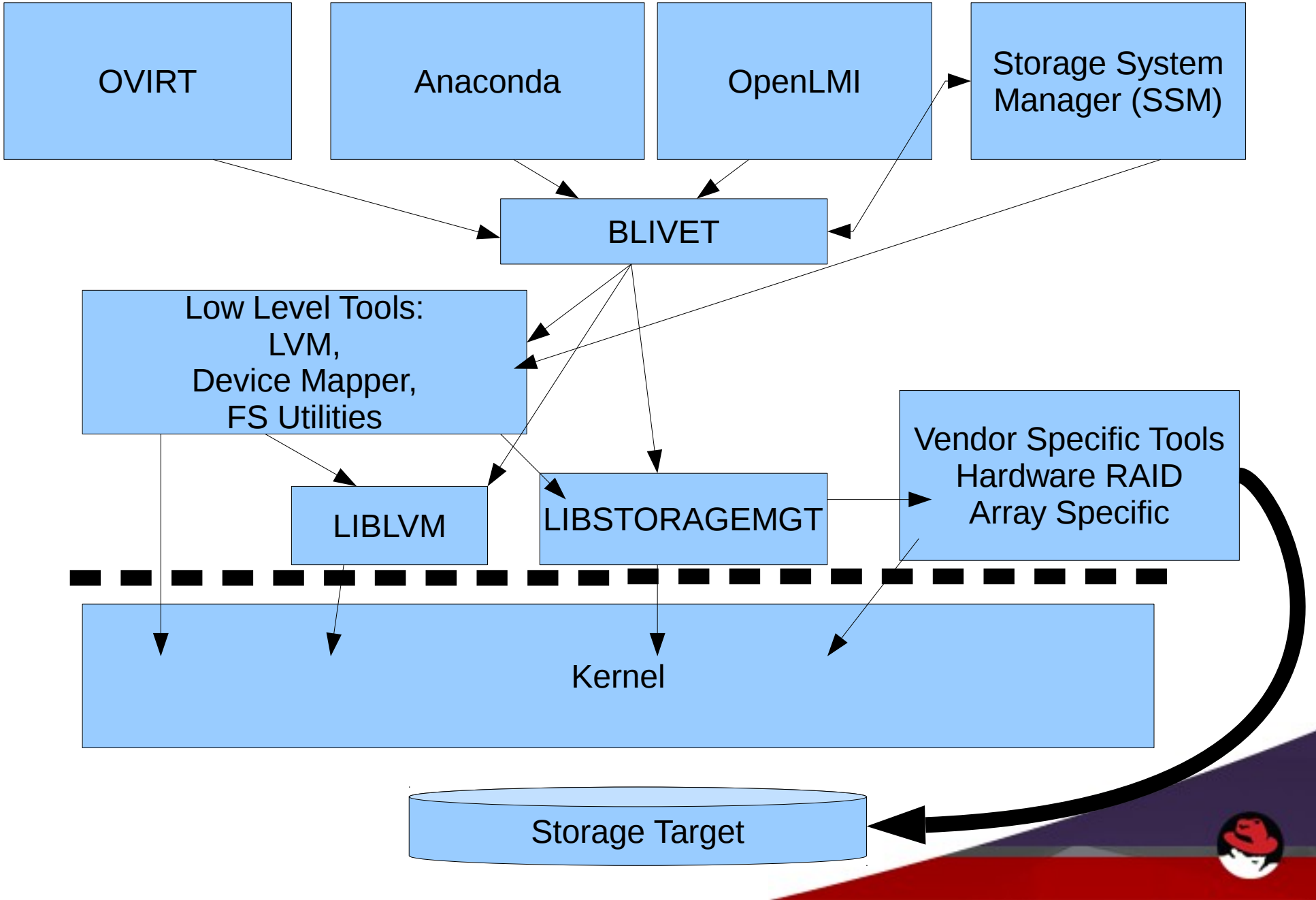
High Level Storage Management Projects

- Storage system manager project
  - CLI for file systems
  - http://storagemanager.sourceforge.net
- openlmi allows remote storage management
  - https://fedorahosted.org/openlmi/
  - http://events.linuxfoundation.org/images/stories/slides/lfcs2013_gallagher.pdf
- Ovirt project focuses on virt systems & their storage
  - http://www.ovirt.org/Home
- Installers like yast or anaconda

Low Level Storage Management Projects

- Blivet library provides a single implementation of common tasks
  - Higher level routines and installers will invoke blivet
  - https://git.fedorahosted.org/git/blivet.git
  - Active but needs documentation!
- libstoragemgt provides C & Python bindings to manage external storage like SAN or NAS
  - http://sourceforge.net/p/libstoragemgmt/wiki/Home
  - Plans to manage local HBA's and RAID cards
- Liblvm provides C & Python bindings for device mapper and lvm
  - Project picking up after a few idle years

Future Red Hat Stack Overview



Getting Read for Persistent Memory

- 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!

Persistent Memory Current Work

- Block level driver for persistent memory parts
  - Best is one driver that supports multiple types of parts
  - Multiple, very early efforts
- 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?

Persistent Memory Standards

- Storage Network Industry Association (SNIA) Working Group on NVM
  - Working on a programming model for PM parts
  - http://snia.org/forums/sssi/nvmp
- New file systems for Linux being actively worked on
  - Will be as painful as multi-threading or teaching applications to use fsync()!
  - Fsync() live on
    - Volatile data lives in CPU caches and needs flushed

Handling Drive Managed SMR

- Have the drive vendors simply hide it all from us!
- Vendors need to add hardware and software
  - Must to virtual to physical remapping similar in some ways to SSD vendors
  - Will increase the costs of each drive
  - Hard to get the best performance
- Some vendors are shipping these SMR drives today
- No changes needed for Linux or other OS platforms

Host Aware SMR

- Host is aware of SMR topology at some layer
  - Avoids sending writes that break the best practices for SMR
  - Write that violates SMR have unpredictable (probably low!) performance
- Allows drives to minimize hardware/software and have reasonable performance as long as most writes behave
- Works with existing operating system stack

Restricted SMR

- Host is aware of SMR topology
  - Only write IO's that follow the SMR rules are allowed
  - Non-obedient writes will be rejected by the SMR drive
- Minimal hardware and software needed on the SMR drive
  - All of the effort is shifted to the operating system stack
- Would not work with existing operating systems and file systems in most cases

SMR Ongoing Work

- Make an SMR device mapper target to hide device from file systems
  - Might support the restricted mode SMR
- Tweak and tune existing file systems to write mostly sequentially
  - Would be the quickest path forward
  - Would support host aware

Bringing SMR and PM Together

- Use the persistent memory as a block level caching device
  - SMR drive (or RAID stripe of SMR drives) as high capacity bulk storage
- Persistent memory only client machines
  - Access bulk data via NFS or iSCSI on high capacity servers with SMR drives
- Application level combinations

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