RED HAT :: CHICAGO :: 2009

SUMMIT

FOLLOW US:

TWITTER.COM/REDHATSUMMIT

TWEET ABOUT US:

ADD #SUMMIT AND/OR #JBOSSWORLD TO THE END OF YOUR EVENT-RELATED TWEET

RED HAT :: CHICAGO :: 2009
SUMMIT

Optimize Storage Performance with Red Hat Enterprise Linux

Mike Snitzer <snitzer@redhat.com>
Senior Software Engineer, Red Hat
09.03.2009

presented by

Agenda

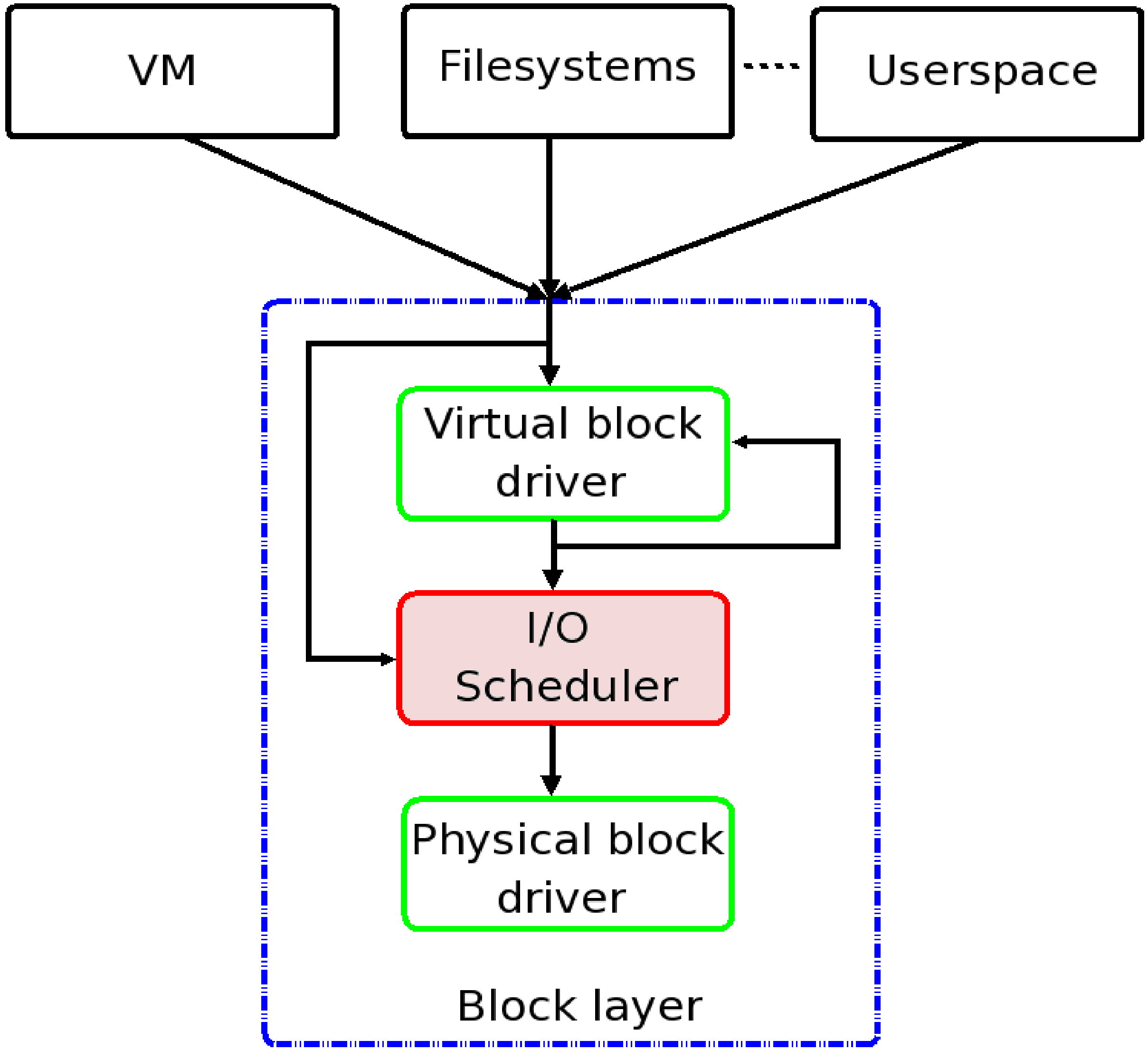
- Block I/O Schedulers
- Linux DM Multipath
- Readahead
- I/O Topology
- Benchmarking and Analysis
- Conclusion
- Questions

RED HAT :: CHICAGO :: 2009
SUMMIT

Block I/O Schedulers

presented by

Block I/O Schedulers – Overview



“Artwork” inspired by http://lwn.net

Block I/O Schedulers – Complete Fair Queuing (CFQ)

- CFQ is the default I/O scheduler in RHEL
- Does best job over widest range of workloads
- One queue for each process submitting I/O
  - Threads within a process get separate queues
- Round-robin among queues that have the same priority
  - Ensures fairness among competing processes
  - Priority is determined by scheduling class and priority level
- slice_idle determines how long CFQ will wait for additional requests to arrive in a queue before switching to the next queue
  - Provided workload is not seeky and application is I/O-bound
  - echo $N > /sys/block/$DEVICE/queue/iosched/slice_idle

Block I/O Schedulers – Complete Fair Queuing (CFQ)

- Offers various I/O nice levels similar to CPU scheduling
- Three scheduling classes with one or more priority levels
  - Real-time (RT) - highest priority class, can starve others
  - Best-effort (BE) – default scheduling class
  - Idle - class that runs if no other processes need the disk
- Priority levels (0 -7) in the RT and BE scheduling classes
  - I/O priority level is derived from CPU scheduling priority
    - io_priority = (cpu_nice + 20) / 5
- See man: ionice (1), ioprio_get (2), ioprio_set (2)
- Refer to: Documentation/block/ioprio.txt

Block I/O Schedulers – Deadline and Noop

- Deadline
  - Attempts to ensure that no request is outstanding longer than its expiration time; read requests have a shorter expiration
  - Maintains 4 queues: Read/Write Sorted, Read/Write FIFO
    - Pulls requests off the sorted queues in batches to minimize seeks; fifo_batch controls sequential batch size
    - Services Read or Write queues if request at respective head expires; expiration times checked after each batch
  - Refer to: Documentation/block/deadline-iosched.txt
- Noop
  - Performs merging but avoids sorting and seek prevention
  - Frequently recommended if using high-end array

Block I/O Schedulers – Choosing wisely

- Can select the default I/O scheduler and override per device
  - elevator={cfq|deadline|noop} on kernel command line (grub)
  - echo {cfq|deadline|noop} > /sys/block/$DEVICE/queue/scheduler
- Deadline vs CFQ
  - CFQ generally outperforms deadline on writes
  - Deadline better on reads for server workloads
  - If running server workloads like: NFS server, iSCSI target, KVM (cache=off)
    - Try CFQ w/ slice_idle=0 to improve CFQ read performance; get closer to deadline read performance
    - Future kernel work will solve this by using shared IO contexts for workloads that interleave reads among multiple threads

RED HAT :: CHICAGO :: 2009 SUMMIT

Linux DM Multipath

presented by

Linux DM Multipath – Blacklist Configuration

- Multipath should only be interacting with appropriate devices
  - Device blacklist can be established in /etc/multipath.conf, default:

```
devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
devnode "^hd[a-z][[0-9]*]"
```

  - To check all invalid devices are blacklisted run: multipath -ll -v3

```
dm-0: blacklisted
dm-1: blacklisted
.
.
.
sda: bus = 1
sda: dev_t = 8:0
sda: size = 156250000
sda: vendor = ATA
sda: product = WDC WD800JD-75MS
sda: h:b:t:l = 0:0:0:0
sda: path checker = readsector0
     (config file default)
```

Linux DM Multipath – Filter Configuration

- “user_friendly_names yes” - simplifies LVM filtering of mpath devices but different nodes won't have the same device names

```
mpath0 (360060160ce831e00645e9544df08de11)
[size=50 GB][features="1 queue_if_no_path"][hwhandler="1 emc"]
\_ round-robin 0 [prio=2][active]
 \_ 0:0:1:0 sdg  8:96   [active][ready]
 \_ 1:0:1:0 sds  65:32  [active][ready]
\_ round-robin 0 [enabled]
 \_ 0:0:0:0 sda  8:0    [active][ready]
 \_ 1:0:0:0 sdm  8:192  [active][ready]
```

- LVM should only allow use of multipath devices and non-mpath devices (e.g. root on /dev/sda2) in /etc/lvm/lvm.conf:

```
filter = [ “a|/dev/sda2|”, “a|/dev/mapper/mpath.*|”, “r|.*|” ]
```

Linux DM Multipath – Device Configuration

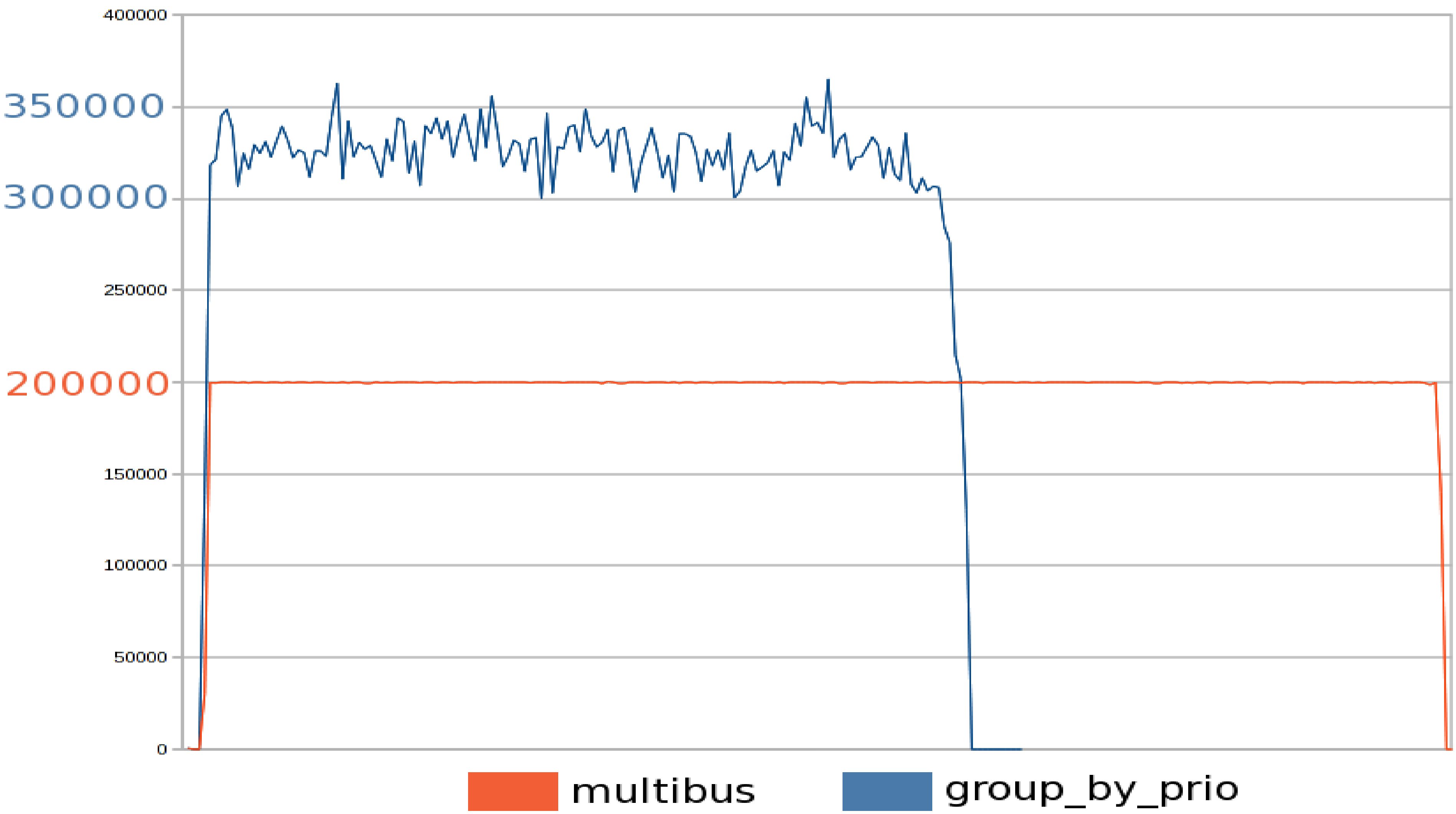
- Developers maintain hardware-specific multipath tuning in multipathd's internal configuration table (hwtable)
- User overrides and extensions are possible by adding custom entries to the 'devices' section of /etc/multipath.conf
  - See man: multipath.conf (5)
  - Consult hardware vendor about appropriate custom entries if you have doubts about DM multipath's support for your hardware
  - Contact Red Hat support if changes are needed
- Show multipathd's active config with:

  'show config' in "multipathd -k" shell

RED HAT :: CHICAGO :: 2009 SUMMIT

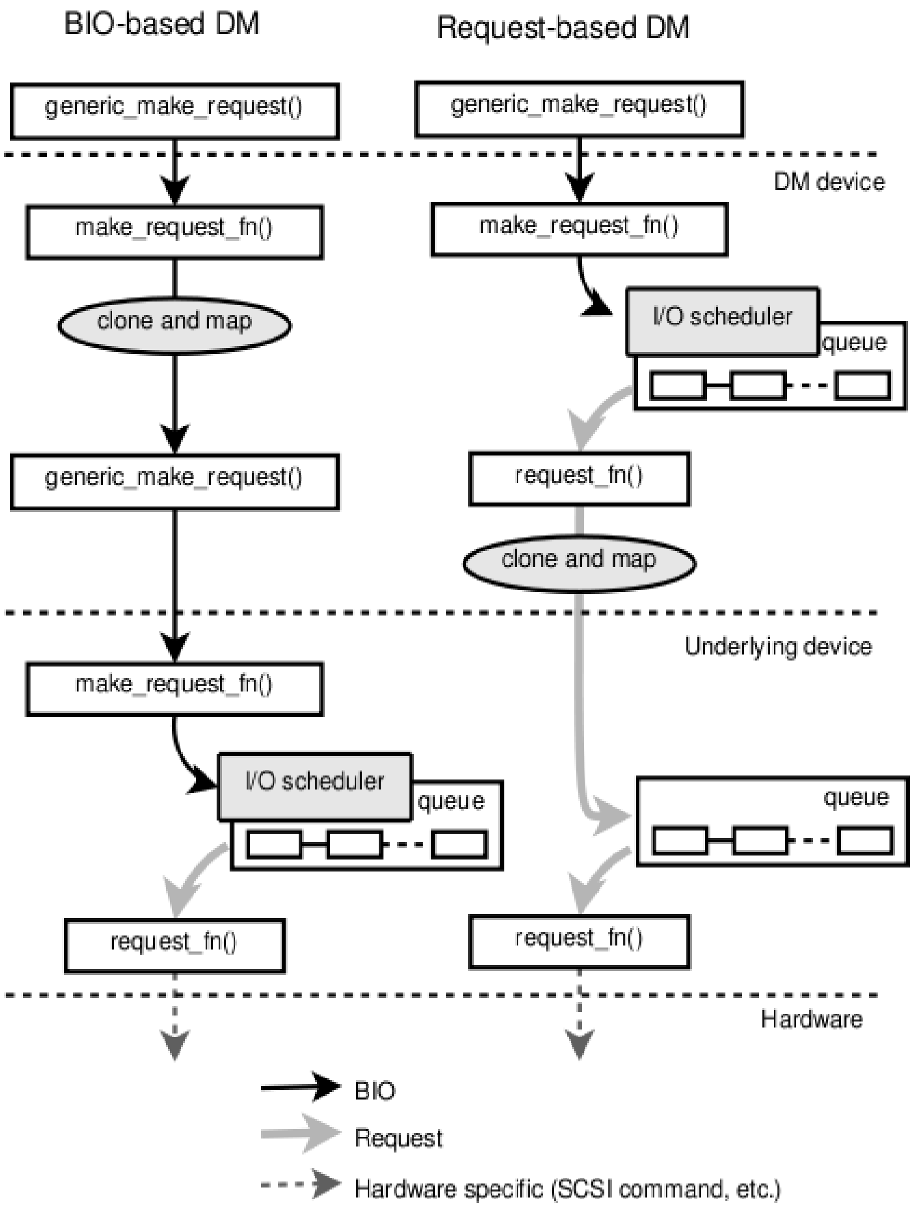
Linux DM Multipath – Proper configuration matters

Improved throughput of ALUA array with proper path_grouping_policy



Linux DM Multipath – Future improvements

- Linux >= 2.6.31 switches DM multipath from BIO-based to request-based
- Improves efficiency by moving multipath layer below the I/O scheduler
  - Reduces total number of requests dispatched even when switching paths frequently (small rr_min_io)
- Improves error-handling by providing DM with more information about SCSI errors
- Adds dynamic load-balancing with 2 new path-selectors:
  - "queue-length" and "service-time" in addition to "round-robin"



RED HAT :: CHICAGO :: 2009
SUMMIT

Readahead

presented by

Readahead – Configuring

- Readahead attempts to improve performance of sequential file reads by reading the file into memory before the app requests it
- Query a device's readahead with: blockdev --getra $DEVICE
- Set a device's readahead with: blockdev --setra $N $DEVICE
  - Caveat: setting readahead too aggressively can waste memory and hurt performance
- LVM inherits readahead from underlying PV when creating LV
  - Change LV's readahead with:

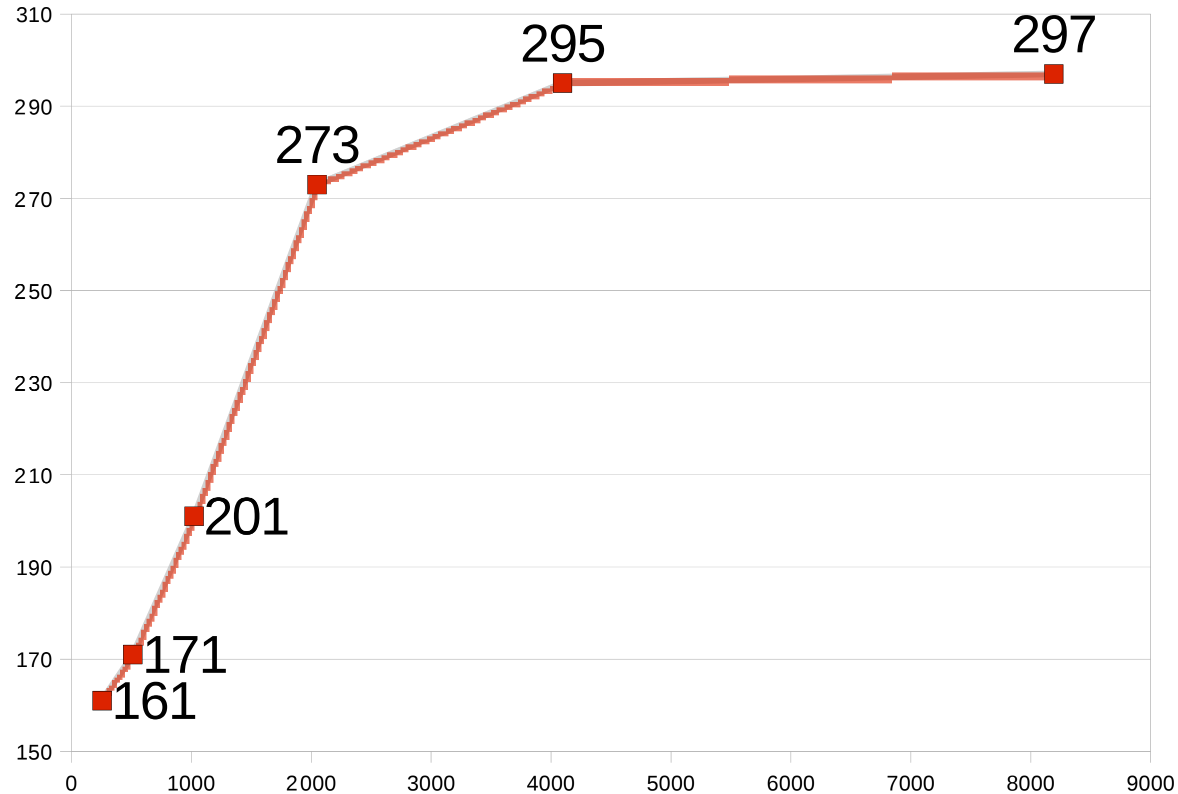
    lvchange -r {ReadAheadSectors|auto|none} ...
    - “auto” allows the kernel to pick a suitable value, e.g.:

      stripe_width=1024K, kernel's readahead=2*1024K
    - “none” is the same as 0

Readahead – Performance impact

512B Sectors	MB/s
256	161
512	171
1024	201
2048	273
4096	295
8192	297

13GB sequential IO (dd w/ bs=128k)



512B Readahead Sectors

RED HAT :: CHICAGO :: 2009
SUMMIT

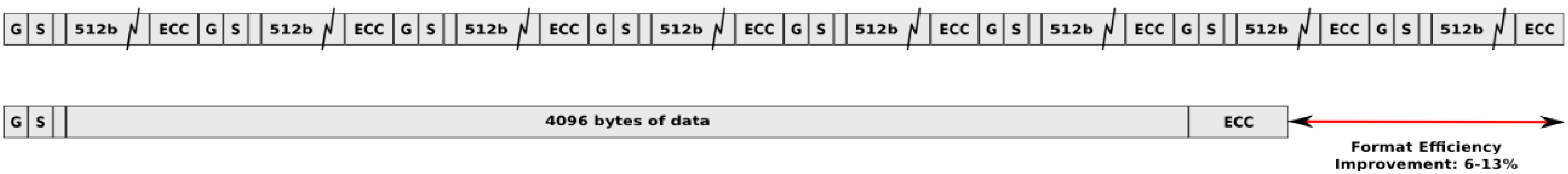
I/O Topology

presented by

I/O Topology – Quest for increased drive capacity

- Each sector on current 512 byte sector disks is quite a bit bigger than 512 bytes because of fields used internally by the drive firmware

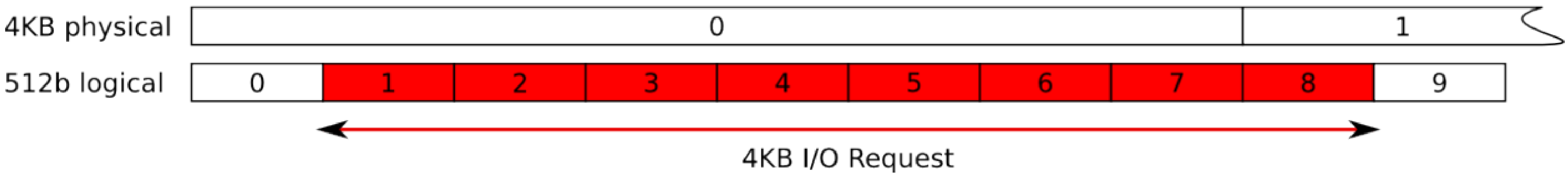
- The only way to increase capacity is to reduce overhead associated with each physical sector on disk



- Top: 8 x 512B sectors, each with overhead, needed to store 4KB of user data
- Bottom: 4KB sector drives can offer the same with much less overhead

I/O Topology – Transitioning to 4KB

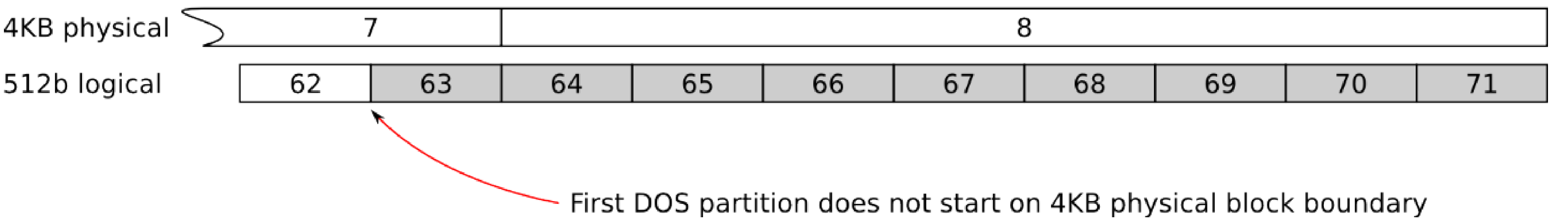
- 4K sector drives **may or may not** accept unaligned IO
- If they do accept unaligned IO there will be a performance penalty
  - Vendors will support a legacy OS with drives that have a 512B logical blocksize (external) and 4K physical blocksize (internal)
  - Misaligned requests will force drive to perform a read-modify-write



- Vendors working on techniques to mitigate the R-M-W in firmware
  - Without mitigation, the drop in performance is quite significant due to an extra revolution; inducing latency and lowering IOPS

I/O Topology – Alignment

- DOS partition tables default to putting the first partition on LBA 63



- Desktop-class 4KB drives can be formatted to compensate for DOS partitioning
  - sector 7 is the lowest aligned logical block, the 4KB sectors start at LBA -1, and consequently sector 63 is aligned on a 4KB boundary
  - Linux >= 2.6.31 allows partition tools, LVM2, etc to understand that this compensation is being used (alignment_offset=3584 bytes), from:

    /sys/block/$DEVICE/alignment_offset

I/O Topology – Performance I/O hints

- Linux >= 2.6.31 also provides the ability to train upper storage layers based on hardware provided I/O hints
  - Preferred I/O granularity for random I/O
    - minimum_io_size - the smallest request the device can perform w/o incurring a hard error or a read-modify-write penalty (e.g. MD's chunk size)
  - Optimal sustained I/O size
    - optimal_io_size - the device's preferred unit of receiving I/O (e.g. MD's stripe width)
- Available through sysfs:

  /sys/block/$DEVICE/queue/minimum_io_size

  /sys/block/$DEVICE/queue/optimal_io_size

I/O Topology – How it is made possible in Linux

- It all starts with the SCSI and ATA protocols
  - The standards have been extended to allow devices to provide alignment and I/O hints when queried
  - Not all hardware will “just work” -- help vendors help you
- Linux now retrieves the alignment and I/O hints that a device reports
  - Uniform sysfs interface works for all Linux block devices!
- Linux DM and LVM2 have been updated to be “topology-aware”
  - Linux MD, XFS, and libblkid are also “topology-aware”; more to come
- Thanks to Martin K. Petersen for implementing Linux's I/O Topology support (and for much of the content and all diagrams in this section!)

RED HAT :: CHICAGO :: 2009
SUMMIT

Benchmarking and Analysis

presented by

Benchmarking and Analysis – General advice

- Benchmark each layer in the I/O stack from the bottom up
- Use target application workload to help select appropriate synthetic benchmarks
- After establishing baseline with synthetic benchmarks the most important benchmark is the target application
- Buffered I/O throughput benchmarks must perform more I/O than RAM can cache
- Clean caches before each iteration of buffered I/O throughput benchmarks:
    - Remount FS or Reboot system
    - Drop caches: echo 3 > /proc/sys/vm/drop_caches
        - Refer to: Documentation/sysctl/vm.txt

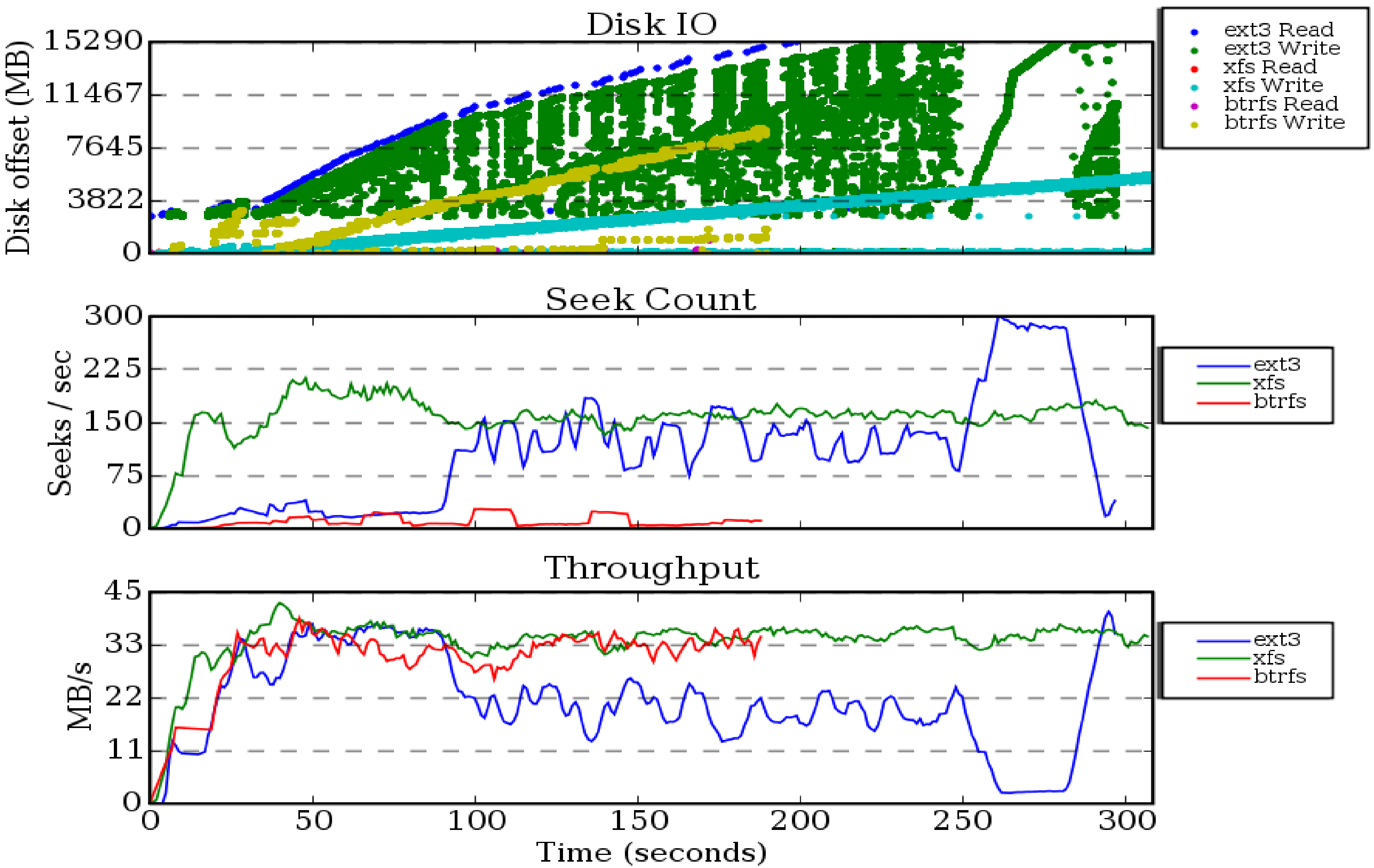
Benchmarking and Analysis – Benchmarking tools

- dd: test buffered and direct IO, provided by coreutils rpm
  - buffered vs direct IO (iflag/oflag=direct avoids page cache)
- fio (Flexible IO tester): http://freshmeat.net/projects/fio/
  - Works on both block devices and files
  - Maintained by Jens Axboe (maintainer of Linux's Block layer)
- ffsb (Flexible Filesystem Benchmark): http://sf.net/projects/ffsb/
- tiobench (threaded i/o tester): http://tiobench.sourceforge.net/
- IOzone: http://www.iozone.org
- fs_mark (simulate mail servers): http://fsmark.sf.net/
- fsx: part of the LTP: http://ltp.sourceforge.net/tooltable.php
- compilebench (fs aging): http://oss.oracle.com/~mason/compilebench/

Benchmarking and Analysis – Analysis tools

- iostat: analyze CPU and I/O statistics, provided by coreutils rpm
  - Useful to run in conjunction with benchmark or target application
- blktrace: generate traces of the I/O traffic on block devices
  - Provides visibility of very detailed I/O event trace information (I/O request sizes, dispatches, merges, etc).
  - blkparse: reads blktrace events to produce human-readable output
  - Google for “blktrace user guide”
- Seekwatcher: generates graphs from blktrace output
  - Helps visualize I/O patterns and performance
  - Maintained by Chris Mason – the lead developer of Btrfs
  - http://oss.oracle.com/~mason/seekwatcher/

Benchmarking and Analysis – Seekwatcher output



RED HAT :: CHICAGO :: 2009 SUMMIT

Conclusion

Conclusion

Linux storage performance tuning is nuanced but quite approachable if you take a bottom up approach

- Careful selection of I/O scheduler and associated tuning
- Properly filter and configure multipath LUNs
- Tune readahead
- Leverage “I/O topology-aware” Linux and associated utilities
- Benchmark all layers to assess impact of various tunings

Slides available here:

http://people.redhat.com/msnitzer/snitzer_rhsummit_2009.pdf

QUESTIONS?

TELL US WHAT YOU THINK:
REDHAT.COM/SUMMIT-SURVEY

Appendix

RED HAT :: CHICAGO :: 2009
SUMMIT

I/O Topology

presented by

I/O Topology – Physical and Logical sectors

- Distinction between Physical and Logical sector size can be visualized as the Firmware (internal) and OS (external) sector size respectively
  - Enterprise-class: physical=logical=4K; misaligned IO not allowed
  - Desktop-class: physical=4K, logical=512; misaligned IO allowed
  - /sys/block/$DEVICE/queue/physical_block_size
  - /sys/block/$DEVICE/queue/logical_block_size
- The SCSI and ATA protocol extensions that make distinction possible:
  - SCSI: physical block size and alignment via READ CAPACITY(16)
  - ATA: physical block size in IDENTIFY word 106, alignment in IDENTIFY word 209
  - SCSI block commands spec provides "Block Limits VPD page" to report performance I/O hints

Linux MD and LVM

Linux MD and LVM – MD chunk size

- MD chunk size governs the unit of I/O that is sent to each raid member
  - Relevant for MD raid levels: 0, 4, 5, 6, 10
  - MD's default chunk size is 64K
  - Should always be > 8-16K to avoid drive firmware's readahead cutoff; otherwise sequential reads suffer
  - Smaller (32-64K) for sequential I/O from a single client
  - Larger (256-512K) for random I/O from single client or multiple clients doing sequential I/O
- Customer case-study, MD raid5 performance:
  - Using 4K * 6, had 30MB/s; dropped to 8MB/s under load
  - Using 256K * 6, consistently saw 110MB/s to 170MB/s

Linux MD and LVM – LVM on MD

- Each LVM2 PV has a number of physical extents of a fixed size (physical extent size, or PE size). The PE size must always be a power of 2. The default is 4 MB and it must be at least 1 KB.
- LVM on MD performs best if the underlying raid is using 2^N data disks:
  - Raid5: 2^N+1 drives (e.g 3, 5, 9, etc).
  - Raid6: 2^N+2 drives (e.g 4, 6, 10, etc).
- Make certain that the start of an LVM2 PV's data area (pe_start) is aligned on a full MD stripe width boundary:
  - chunk_size=64K * 4 data disks, stripe_width=256K
    - RHEL 5: pvcreate --dataalignment 256K ...
    - RHEL 4: pvcreate --metadatasize $((256-4))K