Performance evaluation of Linux Discard Support

(Overview, benchmark results, current status)

Red Hat
Lukáš Czerner

February 12, 2011

redhat.

Part I

Discard Background

Agenda

redhat.

Solid-State Drive

- Flash memory block device
- Wear-leveling needed
- Firmware = black box

redhat.

ATA TRIM Command

- Helps handle garbage collection overhead
- Subsequent READ of TRIM'ed blocks
  1. Read data should NOT change between READ's
  2. Read data should NOT be retrieved from data previously written to any other LBA.
- As long as the device has enough free pages to write to we do not necessarily need it.
- In a nutshell: TRIM command tells the device what LBA'a is not used by the OS anymore.

redhat.

Thin Provisioning

- Unlike in traditional storage, there is no fixed one-to-one logical bock to physical storage mapping
- More efficient use of storage space
- Block reclamation interface needed

redhat.

SCSI UNMAP / WRITE SAME

- Storage space reclamation interface
- Subsequent READ of unmapped blocks
  1. Read data should NOT change between READ's
  2. Read data should NOT be retrieved from data previously written to any other LBA.
- Unlike with SSD's we can not afford to wait until we run out of space for reclamation.

redhat.

Linux Discard Implementation

- Abstraction for the two underlying specifications:
  1. ATA TRIM Command
  2. SCSI UNMAP / WRITE SAME
- API for user-space
  - BLKDISCARD ioctl
  - Added with v2.6.27-rc9-30-gd30a260
- API for File Sytems
  1. `sb_issue_discard()`
  2. `blkdev_issue_discard()`

redhat.

Part II

Discard Performance

Agenda

redhat.

What do we need to find out ?

- Does discard really work ? Is it reliable ?
- How fast/slow is it ?
- Is there any difference between devices from different vendors ?
- What is the ideal discard size ?
- SSD performance degradation

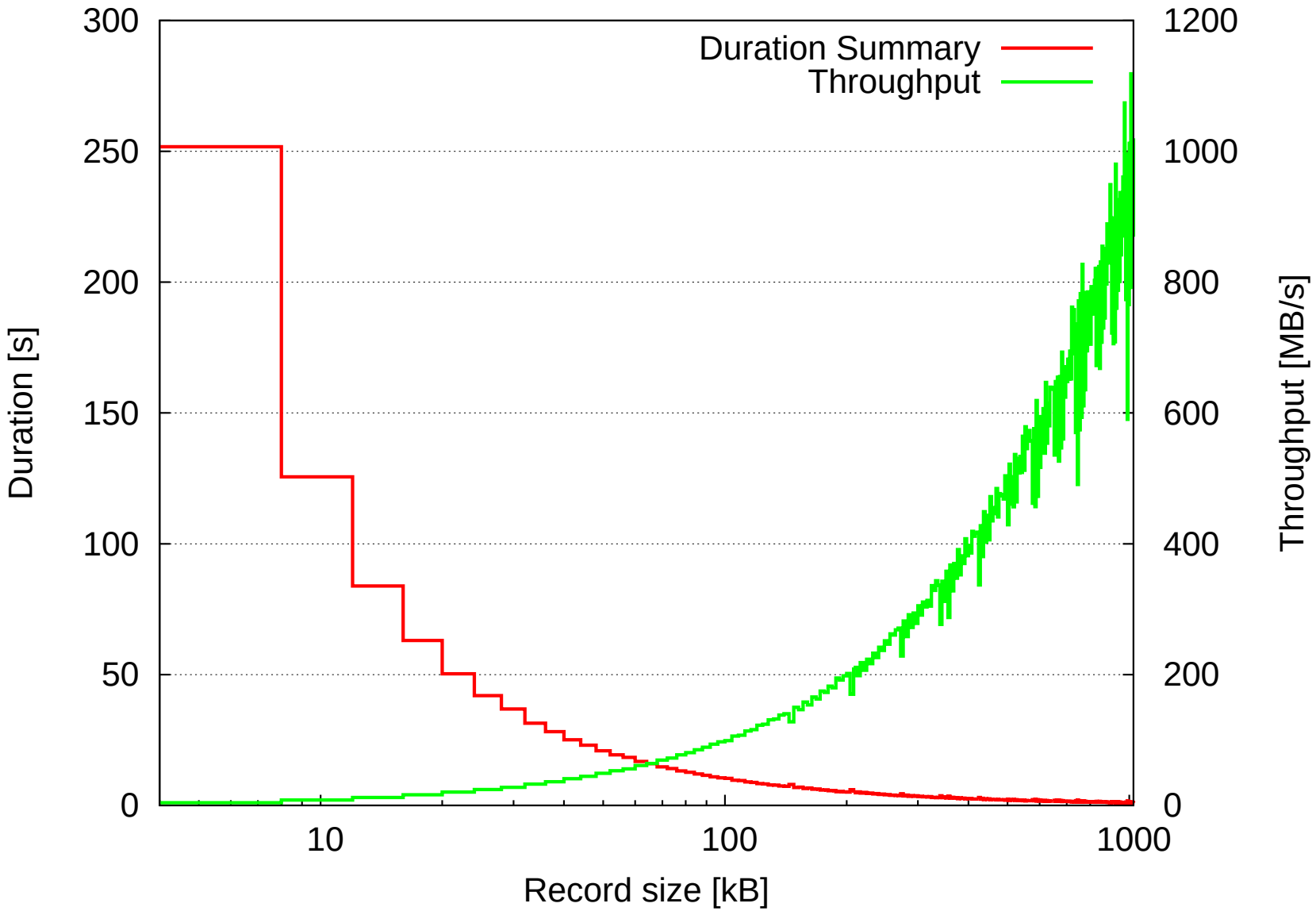
redhat.

How do we test it ?

- BLKDISCARD ioctl()
- Automatic discard of different ranges
- Different discard patterns
  1. sequential performance
  2. random IO peformance
  3. discard already discarded blocks
- test-discard - discard benchmarking tool
  - http://sourceforge.net/projects/test-discard/
- impression - filesystem aging tool

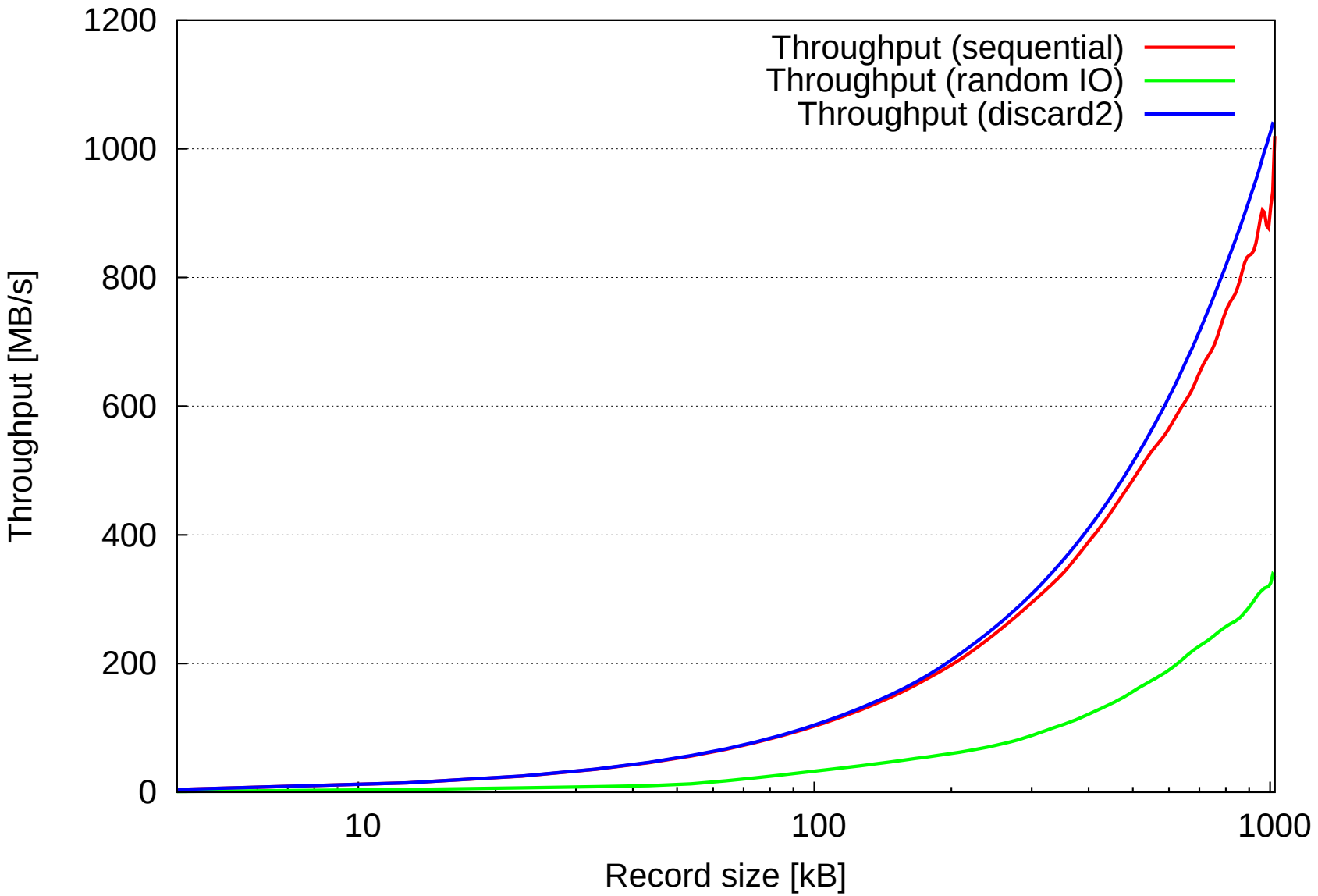
redhat.

Sequential discard performance



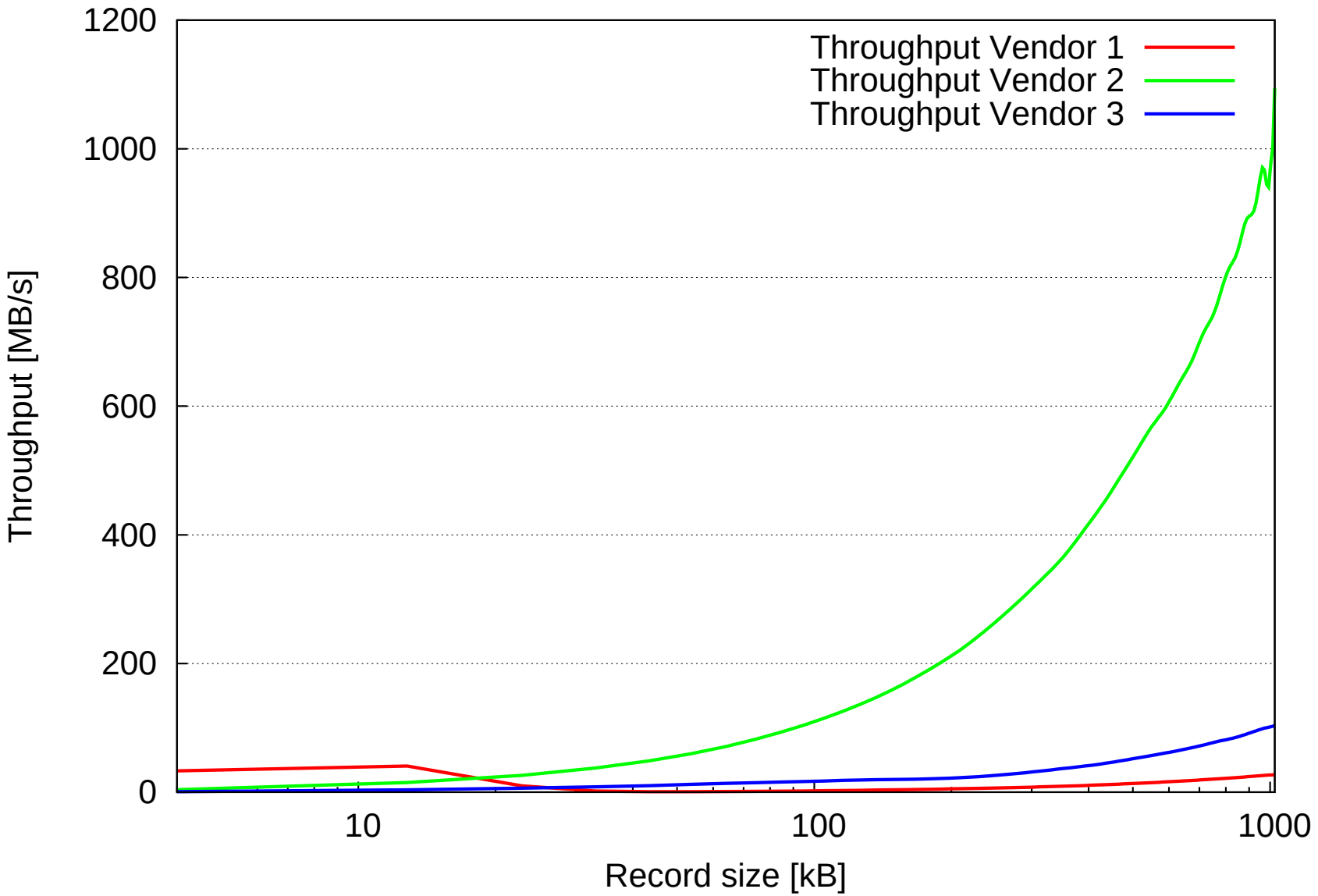
redhat.

Different modes comparison



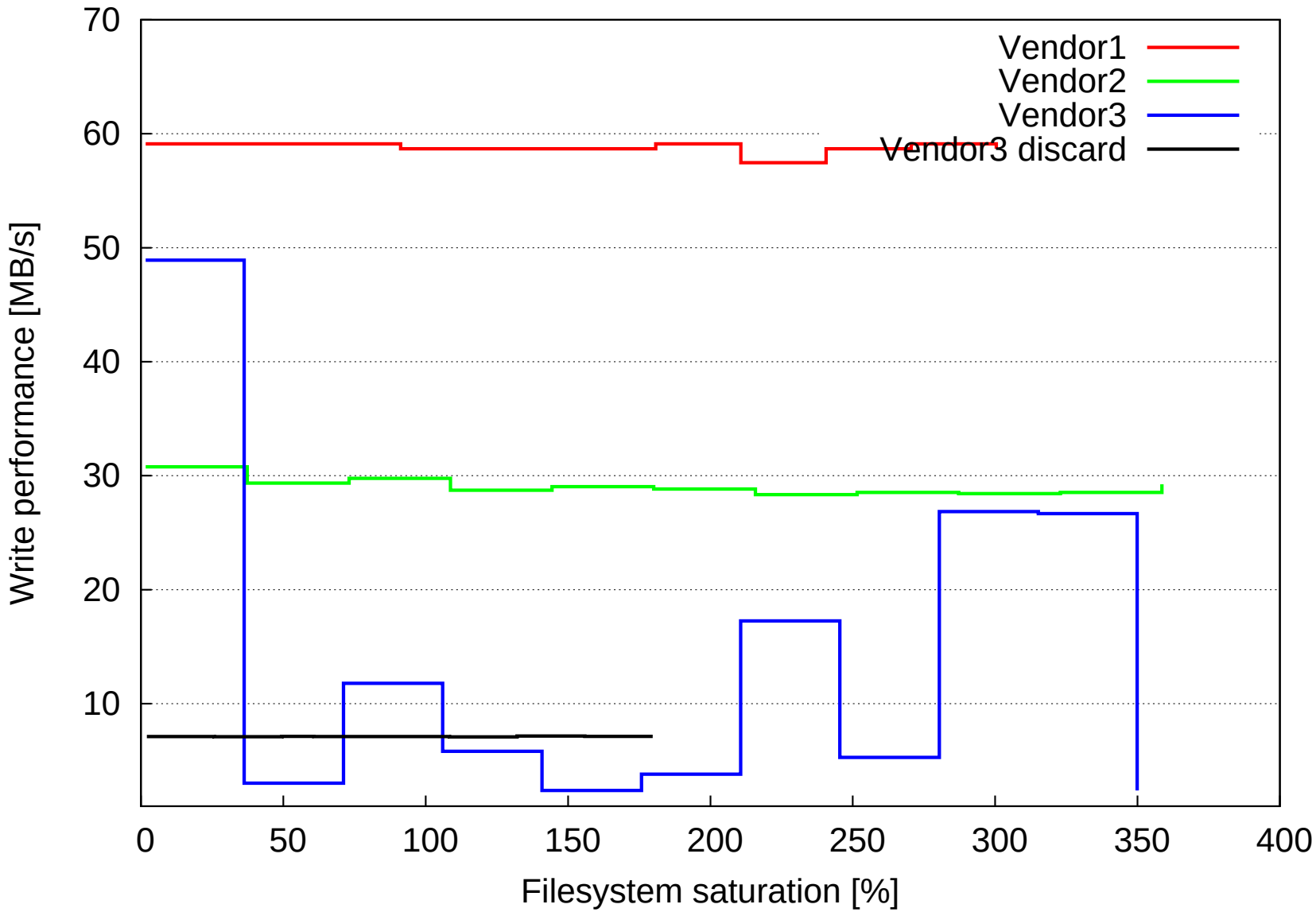
redhat.

Difference between various vendors



redhat.

SSD performance degradation



redhat.

Part III

Discard Support for Linux File systems

redhat.

Agenda

6 Periodic Discard

7 Discard Batching

8 Different Approach

redhat.

Periodic discard

- Easy to implement
- File system support
  1. ext4 (v2.6.27-5185-g8a0aba7)
  2. btrfs (since upstream)
  3. gfs2 (v2.6.29-9-gf15ab56)
  4. fat, swap, nilfs
- `mount -o discard /dev/sdc /mnt/test`
- TRIM is non-queueable command - implications ?

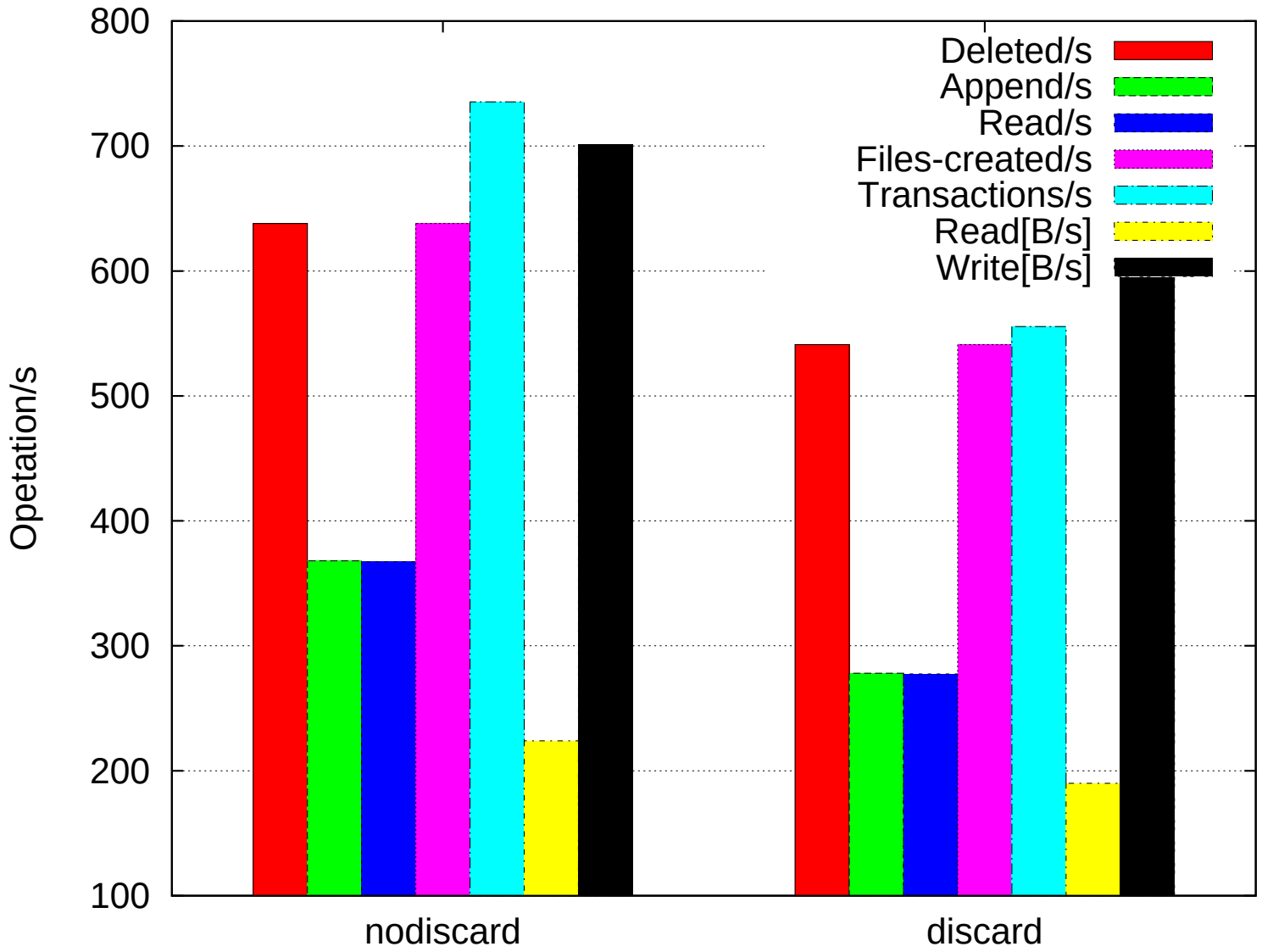
redhat.

Benchmarking periodic discard

- Expectations ?
- Testing methodology
  1. Metadata intensive load
  2. Load with removing files
  3. Reasonable file size distribution
- Discard-kit
  1. Using PostMark
  2. http://sourceforge.net/projects/test-discard/files/

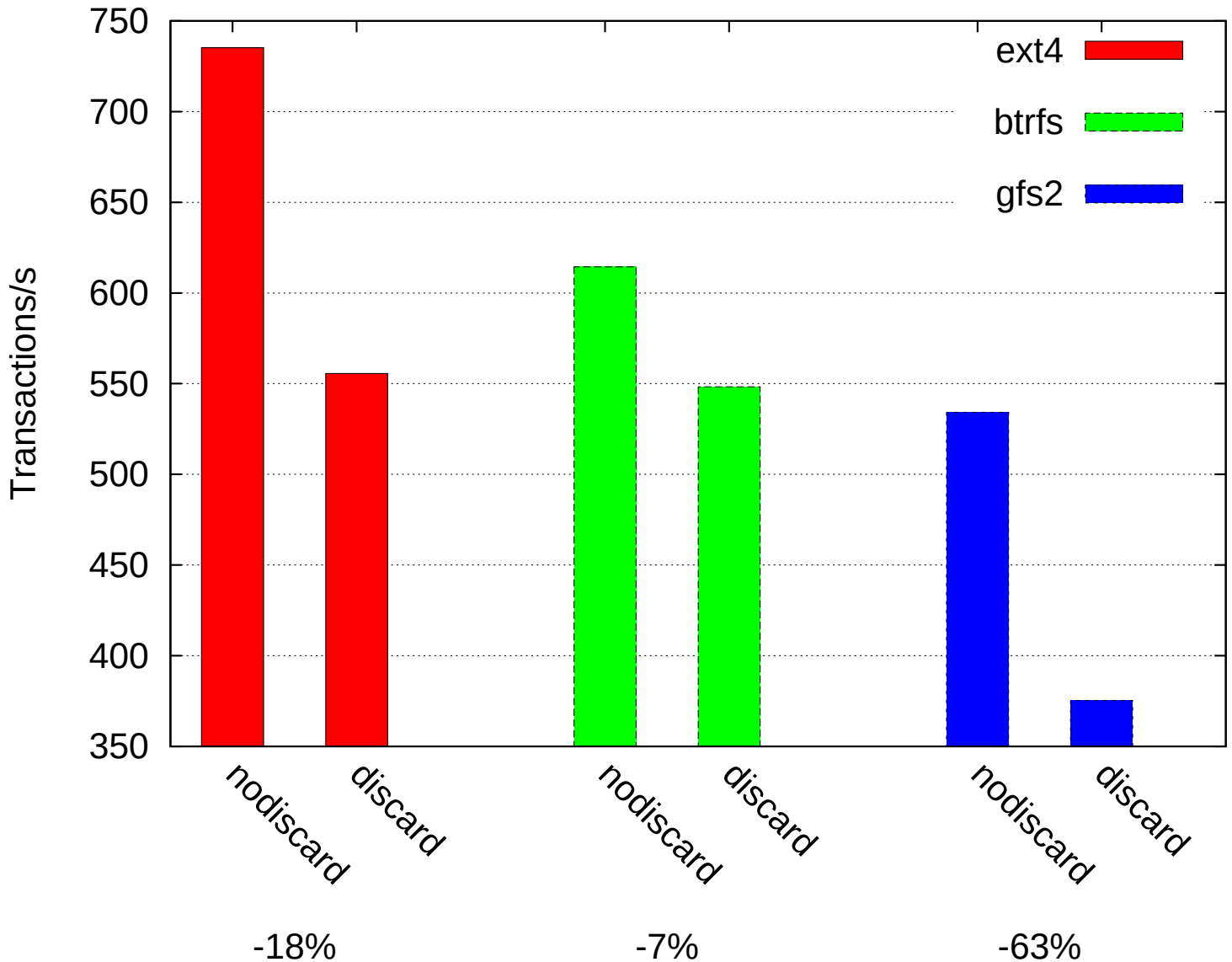
redhat.

Ext4 performance (18% hit)



redhat.

Performance with various file systems



redhat.

Discard Batching - The idea

- Fine-grained discard is not necessarily needed
- Small extents are slow
- With time, freed extents tends to coalesce
- Disadvantages
  1. There is a price for tracking freed extents
  2. Discarding already discarded blocks should be easy, but...
  3. Daemon (in-kernel, user-space) needed.
  4. File system independent solution would most likely be pain to do right (if possible).

redhat.

Batched discard support

- File system specific solution
- Provide ioctl() interface - FITRIM
- Do not disturb other ongoing IO too much
  1. Prevent allocations while trimming
  2. How to handle huge filesystem ?
- File system support
  1. ext4 (v2.6.36-rc6-35-g7360d17)
  2. ext3 (v2.6.37-11-g9c52749)
  3. xfs (v2.6.37-rc4-63-ga46db60)

redhat.

FITRIM ioctl

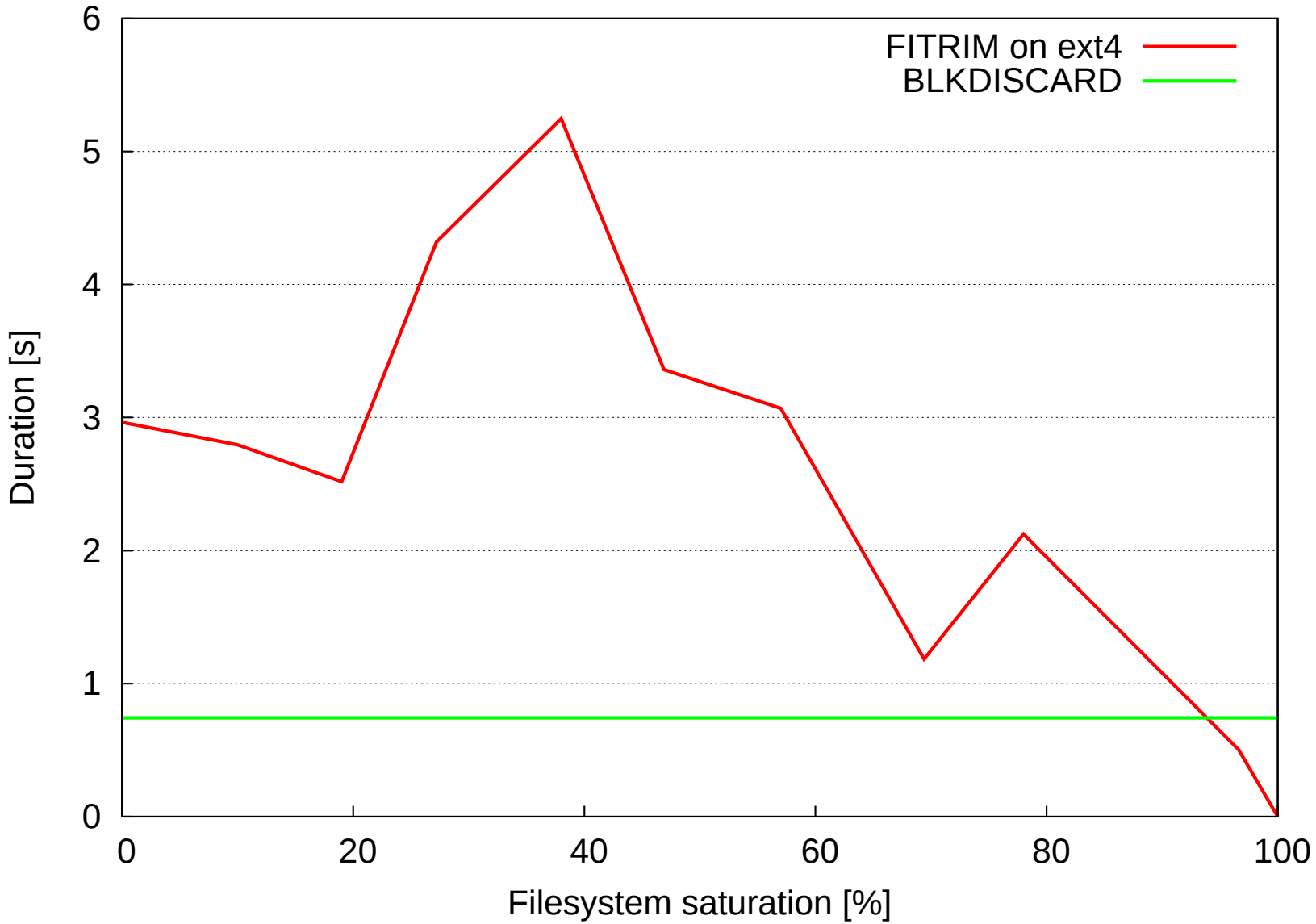
- Ioctl with one RW parameter defined in `linux/fs.h`

```
struct fstrim_range {
__u64 start;
__u64 len;
__u64 minlen;
}
```

- fstrim tool
  - http://sourceforge.net/projects/fstrim/
- util-linux-ng
  - Since v2.18-165-gd9e2d0d

redhat.

Batched discard benchmark results



redhat.

Alternative approach

- It is always a compromise
- The future of SSD's and thinly provisioned LUN's (???)

redhat.

Part IV

Discard Support in user-space

redhat.

Agenda

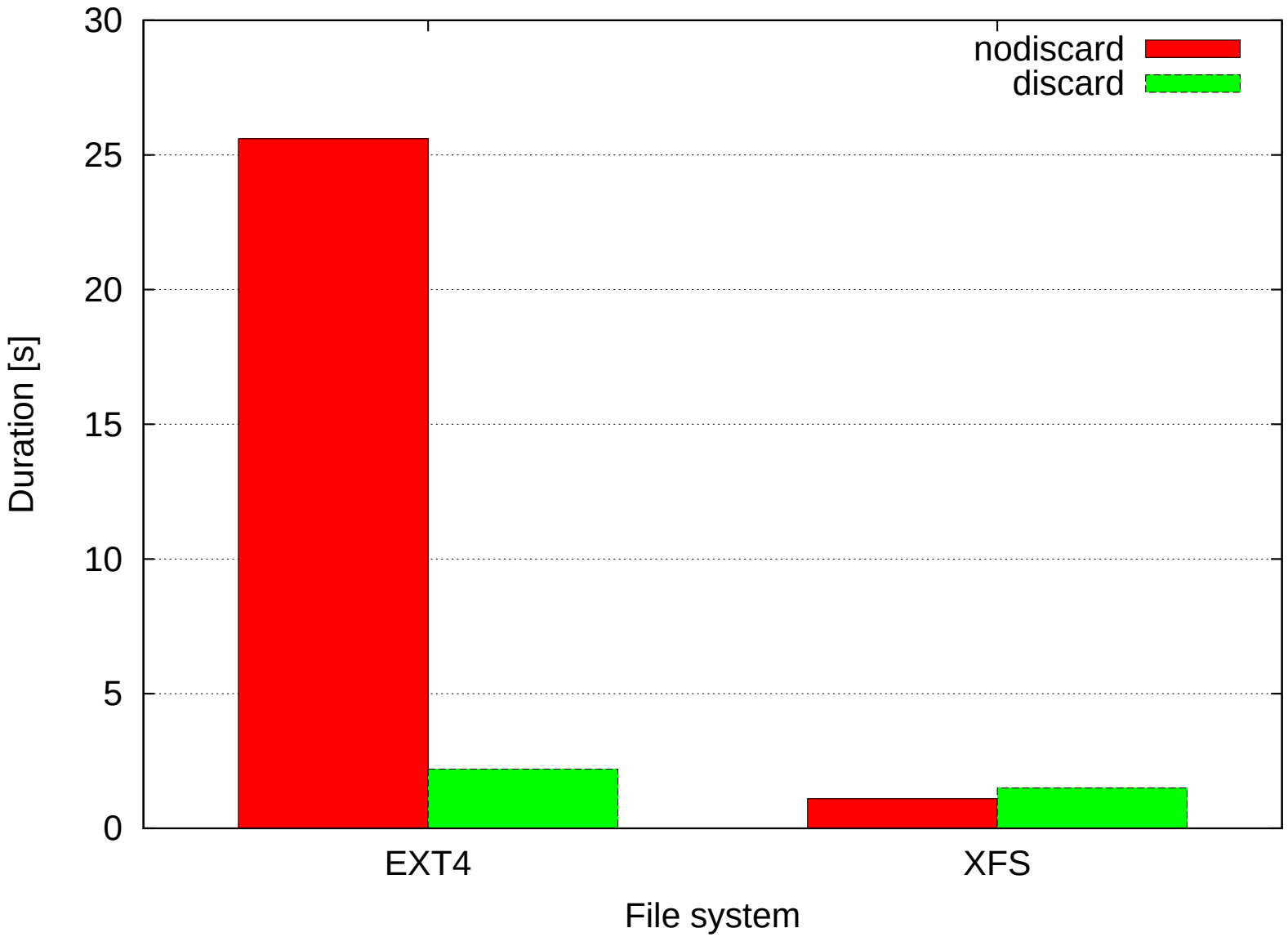
redhat.

Discard in e2fsprogs tools

- Using BLKDISCARD ioctl()
- mke2fs
  1. Refresh SSD's garbage collector
  2. discard zeroes data - significant speed boost
  3. `mkfs.ext4 -E discard /dev/sdc`
- e2fsck
  1. After the last check discard free space
  2. Non detected file system errors ? oops
  3. `fsck.ext4 -E discard /dev/sdc`
- resize2fs
  1. Refresh SSD's garbage collector
  2. discard zeroes data - significant speed boost
  3. `resize2fs -E discard /dev/sdc`

redhat.

File system creation



redhat.

Fstrim tool

- Very simple tool to invoke FITRIM ioctl on mounted file system
- Stand-alone tool
  - http://sourceforge.net/projects/fstrim/
- Since v2.18-165-gd9e2d0d part of util-linux-ng

redhat.

Part V
Summary

Summary

- Linux Discard support is a abstraction for underlying specification
- Exported via BLKDISCARD ioctl to user-space and blkdev_issue_discard() for filesystems
- Discard testing kit (Discard-kit)
  1. test-discard
  2. PostMark
- Filesystem support
  1. Fine grained (online) discard - `mount -o discard`
  2. Batched discard support - fstrim from util-linux-ng
- Support in user-space utilities
  1. Filesystem creation (mkfs)
  2. e2fsprogs - mkfs,e2fsck,resize2fs
  3. xfsprogs - mkfs
  4. fstrim

The end.

Thanks for listening.

Useful links

- http://sourceforge.net/projects/fstrim/
- http://sourceforge.net/projects/test-discard/
- http://people.redhat.com/lczerner/discard/