Coerced Cache Eviction and Discreet-Mode Journaling: Dealing with Misbehaving Disks

Abhishek Rajimwale*, Vijay Chidambaram, Deepak Ramamurthi
Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau

*Data Domain Inc
University of Wisconsin Madison
Disks are not perfect

- Expanding disk fault model
- Latent Sector Errors [Bairavasundaram SIGMETRICS 07]
  - RAID-6
- Block Corruption [Bairavasundaram FAST 08]
  - Checksums
- The disk cache
  - Always trusted so far
Disk Caches

• Disk cache improves performance
  – But at the risk of data loss
• Order of writes issued by file system:
  – A, B, C
• Disks **reorder** writes during destaging:
  – B, A, C
• File systems **flush** the disk cache to ensure correct ordering of writes
  – A, flush, B, flush, C
Problem: Flushing doesn’t work

- Disks can fail to flush data upon request

- One reason: Bugs
  - Errors in the storage stack [Bairavasundaram FAST 08]
  - Improper propagation of error codes [Bairavasundaram FAST 08]
  - Inadequate failure policies [Prabhakaran SOSP 05]
  - Bugs in the firmware [Ghemawat SOSP 03]
Disks can lie!

- Misbehaving disks *ignore or delay* flush requests
- Increases risk for data loss
  - File systems usually blamed for such loss

**Sequential writes**

- Avg time (msec)
- Write size:
  - 4k, 16k, 64k, 128k, 512k, 1m

3/13/12
Disks can lie!

- Evidence from industry experts
  - Microsoft
  - Seagate

- From the `fcntl` man page in Mac OSX:

  `F_FULLFSYNC` Does the same thing as `fsync(2)` then asks the drive to flush all buffered data to the permanent storage device (arg is ignored). This is currently implemented on HFS, MS-DOS (FAT), and Universal Disk Format (UDF) file systems. The operation may take quite a while to complete.

  Certain FireWire drives have also been known to ignore the request to flush their buffered data.
Ordering points are essential

• All modern file systems depend on ordering points
  – Journaling file systems (ext3, ext4)
    • Data before the commit block
  – Copy on write file systems (ZFS)
    • Data before the uber-block

• If ordering points are not enforced:
  – Data corruption
  – Inconsistent file system
Summary

• We present Coerced Cache Eviction (CCE)
  – Write extra data into the cache to evict target blocks
• We show how to characterize 9 SATA disk drive cache
  – Examine the wide range of caching policies
• We implement CCE in ext3
  – Well known journaling file system
• CCE provides stronger enforcement for ordering points
  – At acceptable overheads
Outline

• Motivation
• Background
• Coerced Cache Eviction
• Cache Fingerprinting
• Discreet Mode Journaling
• Evaluation
• Conclusion
File System Background

• Consider deleting a file
  – Removing its directory entry
  – Freeing the space occupied by the file and its metadata

• Journaling file system
  – Makes sure all changes get to disk or none do
  – Groups writes into transactions
  – Writes everything to a log first
  – Checkpoints to disk later
File System Background

• Ext3 file system
  – Semi-modern journaling file system
  – Well known, well understood
• Variants of journaling
  – Data journaling mode
    • Everything (data, metadata) goes to the log first
  – Ordered journaling mode
    • Only metadata is logged
Data Journaling

Memory

Disk Surface

Journal

Fixed locations
Data Journaling

Disk Cache

Disk Surface

Journal

Fixed locations
Outline

- Motivation
- Background
  - Coerced Cache Eviction
  - Cache Fingerprinting
  - Discreet Mode Journaling
- Evaluation
- Conclusion
Coerced Cache Eviction

• Ensures that cache has been truly flushed
• Key idea:
  – Extra writes to flush the disk cache
  – Desired Order of writes: A, B, C
  – With CCE:
    • Write A
    • Write to flush zone
    • Write B
    • Write to flush zone
    • Write C
Coerced Cache Eviction

Memory

Disk Cache

Disk Surface

Journal

Flush Zone

Fixed locations
Coerced Cache Eviction

• Desired properties:
  – High probability of flushing target blocks
  – Low performance overhead

• Need to understand the disk cache to design the flush workload
Outline

• Motivation
• Background
• Coerced Cache Eviction
  • Cache Fingerprinting
• Discreet Mode Journaling
• Evaluation
• Conclusion
Cache Fingerprinting

• Manufacturers don’t expose details about disk caches

• Disk caches can vary in:
  – Read/Write partition size
  – Number of segments
  – Replacement policy

• Poorly characterized in literature
Cache Fingerprinting

• Flush micro-benchmark:
  – Write target block
  – Write varied flush workload – *measure cost*
  – fsync()
  – Read target – *infer eviction*

• Micro-benchmark is repeated
  – Probability of eviction is calculated

• Vary in each workload:
  – Number of writes
  – Amount of data in each write
  – Sequential/Random writes
Cache Fingerprinting

- Eviction fingerprint
  - Probability of eviction is visually shown
  - Darker region indicates higher probability

![Graph showing eviction probability](image-url)
Cache Fingerprinting

- Performance fingerprint
  - Time taken to write flush workload
  - Darker region indicates more time

![Graph showing performance fingerprint](image-url)
Cache Fingerprinting

- Selecting a flush workload:
  - Combine information from both fingerprints
  - High probability of eviction
    - Dark region in eviction fingerprint
  - Low performance cost
    - Light region in performance fingerprint
## Cache Fingerprinting

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Cache (MB)</th>
<th>Capacity (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitachi</td>
<td>8</td>
<td>80</td>
</tr>
<tr>
<td>Hitachi</td>
<td>32</td>
<td>1024</td>
</tr>
<tr>
<td>Samsung</td>
<td>8</td>
<td>250</td>
</tr>
<tr>
<td>Samsung</td>
<td>16</td>
<td>250</td>
</tr>
<tr>
<td>Western Digital</td>
<td>16</td>
<td>320</td>
</tr>
<tr>
<td>Western Digital</td>
<td>64</td>
<td>800</td>
</tr>
<tr>
<td>Seagate</td>
<td>8</td>
<td>250</td>
</tr>
<tr>
<td>Seagate</td>
<td>16</td>
<td>320</td>
</tr>
<tr>
<td>Seagate</td>
<td>32</td>
<td>750</td>
</tr>
</tbody>
</table>
Cache Fingerprinting

Sequential writes may be ineffective at flushing

– Regardless of the size of the write

A number of random writes are required
Cache Fingerprinting

Vertical stripes indicate that the cache is segmented
– Each write, regardless of size, is sent to one segment

Eviction Probability

- 90 – 100%
- 70 – 90%
- 50 – 70%
- 30 – 50%
- 10 – 30%
- 0 – 10%
Cache Fingerprinting

Cache behavior of disks from the same manufacturer is qualitatively similar across their different models.

Eviction Probability

- 90 – 100%
- 70 - 90%
- 50 – 70%
- 30 – 50%
- 10 – 30%
- 0 – 10%
Cache Fingerprinting

• It’s not all good news however:
  – Some caches appear to use random replacement policies
  – For such caches, we cannot evict blocks with 100% certainty
  – A large number of random writes are required to get high eviction probability
# Cache Fingerprinting - Results

<table>
<thead>
<tr>
<th>Drive</th>
<th>Number of writes</th>
<th>Total Data (MB)</th>
<th>Eviction Probability</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitachi 8 MB</td>
<td>1</td>
<td>2.38</td>
<td>100</td>
<td>0.05</td>
</tr>
<tr>
<td>Hitachi 32 MB</td>
<td>1</td>
<td>11</td>
<td>100</td>
<td>0.087</td>
</tr>
<tr>
<td>Seagate 8 MB</td>
<td>256</td>
<td>31</td>
<td>100</td>
<td>0.87</td>
</tr>
<tr>
<td>Seagate 16 MB</td>
<td>128</td>
<td>17</td>
<td>100</td>
<td>0.342</td>
</tr>
<tr>
<td>Seagate 64 MB</td>
<td>128</td>
<td>37</td>
<td>100</td>
<td>0.396</td>
</tr>
<tr>
<td>Samsung 8 MB</td>
<td>128</td>
<td>49</td>
<td>~90</td>
<td>1.328</td>
</tr>
<tr>
<td>Samsung 16 MB</td>
<td>256</td>
<td>128</td>
<td>~90</td>
<td>2.872</td>
</tr>
<tr>
<td>Western Digital 16 MB</td>
<td>1792</td>
<td>19</td>
<td>~90</td>
<td>5.107</td>
</tr>
<tr>
<td>Western Digital 64 MB</td>
<td>256</td>
<td>1</td>
<td>100</td>
<td>7.705</td>
</tr>
</tbody>
</table>
Outline

- Motivation
- Background
- Coerced Cache Eviction
- Cache Fingerprinting
- **Discreet Mode Journaling**
- Evaluation
- Conclusion
Discreet Mode Journaling

• Incorporating CCE into ext3
  – Fingerprint the disk to find optimal flush workload
  – Create flush zone with suitable size
  – Modify ext3 to issue flush zone writes:
    • One at each ordering point
    • # of CCE operations = # of ordering points

• Can be used with any disk:
  – As long as the disk is fingerprinted first
Outline

• Motivation
• Background
• Coerced Cache Eviction
• Cache Fingerprinting
• Discreet Mode Journaling

• Evaluation
• Conclusion
Evaluation

• Goal:
  – CCE provides higher reliability
  – At what cost? Is it practical to use?

• Experimental setup:
  – File system: Ext3
  – Disk: Hitachi 8 MB
  – Journaling mode: Data journaling
    • (See paper for ordered journaling results)
  – Operating system: Linux 2.6.13, Linux 2.6.23
Evaluation

• What we compare:
  – Regular journaling with disk cache turned off
    • “Safe” but slow
    • Disk might not obey command to turn off cache!
  – Regular journaling with disk cache turned on
    • Unsafe but fast
  – Discreet mode journaling
    • Midway option – Safe but with cost
Evaluation

• Benchmarks:
  – OpenSSH
    • copy, untar, configure, make
  – Postmark
    • Simulates a mail server
    • Single threaded
  – Filebench Webserver
    • I/O intensive
  – Filebench Varmail
    • Multithreaded postmark
Evaluation – OpenSSH

Data Journaling Mode

- regular w/o cache
- discreet
- regular w/ cache
Evaluation – Postmark

Data Journaling Mode

- regular w/o cache
- discreet
- regular w/ cache

Time (s)

0 100 200 300 400 500 600 700 800
Evaluation – Filebench Webserver

Data Journaling Mode

Throughput (MB/s)

- regular w/o cache
- discreet
- regular w/ cache

3/13/12
Evaluation – Filebench Varmail

Data Journaling Mode

Throughput (MB/s)

- regular w/o cache
- discreet
- regular w/ cache

3/13/12
Evaluation – Filebench varmail

• Workload writes a small amount of data and calls \texttt{fsync()} repeatedly

• Each \texttt{fsync()} causes 3 CCEs

• Number of optimizations:
  – Incorporate Group Commit in varmail
    • Improves throughput for all modes
  – We use a few other techniques as well (see paper)
Evaluation – Filebench Varmail

Original performance

With optimizations

Throughput (MB/s)

regular w/o cache

discreet

regular w/ cache
Summary

• Coerced Cache Eviction (CCE):
  – Run file systems reliably on top of misbehaving disks
• Characterization of 9 SATA disk caches through fingerprints
• Discreet Mode Journaling:
  – Implementation of CCE for ext3 filesystem
  – Acceptable performance on 3 workloads
    • Only if the cache doesn’t use random replacement
  – High overhead for apps which call \texttt{fsync()} frequently
Conclusion

• Trust in disk is weakening:
  – Latent Sector Errors
  – Block corruption
  – Cache flushing

• Cloud computing systems:
  – Virtualized hardware
  – Large software stack

• Can such hardware be trusted?

• Will coercion be more widely used?
Thank you!

Advanced Systems Lab (ADSL)
University of Wisconsin-Madison
http://www.cs.wisc.edu/adsl