Practical Scrubbing
Getting to the bad sector at the right time

George Amvrosiadis
Bianca Schroeder
University of Toronto

Alina Oprea
RSA Laboratories

The Security Division of EMC
Hard disk errors & Scrubbing
What could go wrong?
What could go wrong?

- Spindle failure
- Head crashed/stuck, lube build-up
- Electrical failure
- Firmware bug
- PCB failure
- Surface abnormalities

Tuesday, June 26, 2012
What could go wrong?

What could go wrong?

No Redundancy + LSE = Data Loss

Exp. errors during reconstruction

2012 (4x4TB)  
Expected: 0.96 LSEs

~2015 (4x12TB)  
Expected: 2.88 LSEs

Single-parity RAID (3+P) Array Capacity (TB)

Steven R. Hetzler, “System Impacts of Storage Trends: Hard Errors and Testability”. In USENIX ;login:, v.36/3

Tuesday, June 26, 2012
• **Goal:** Detect LSEs timely to enable recovery

• **How:** Background process verifying sector contents
  - **Detection speed:** verify sectors at high frequency
  - **Verification cost:** avoid delaying workload requests

• **Previous Work:** Focus on detecting LSEs fast

• **Cost of scrubbing?** Practical questions raised:
  - How do I *implement* a scrubber? How do I *configure* it to find LSEs fast?
  - When should I scrub, to *minimize impact* on the system?
Implementation & Configuration
Data scrubbing

• **Option 1**: Use READs to verify data integrity
  - **Overhead**: Data transfer cost, Cache pollution
  - **Correctness**: Might not check medium surface

• **Option 2**: Use VERIFY firmware command
  - **Caveat**: Treated as scheduling barriers
  - **Solution**: Disguise scrubber’s VERIFYs as READs
Data scrubbing

• **Option 1:** Use READS to verify data integrity
  - **Overhead:** Data transfer cost, Cache pollution
  - **Correctness:** Might not check medium surface

• **Option 2:** Use VERIFY firmware command
  - **Caveat:** Treated as scheduling barriers
  - **Solution:** Disguise scrubber’s VERIFYs as READS
System Overview

Filesystem Layer

Generic Block Layer

Scrubbing Framework

I/O Scheduler

On-disk cache

Hard disk
Finding the sweet spot between scrub throughput and service delay

Parameter 1: VERIFY request size

- Marginal increase in throughput for requests ≥ 4 MB
- Seeking cost dominant for requests ≤ 64 KB
- Higher service delay

Fujitsu SCSI, 36GB, 10K RPM
Fujitsu SAS, 73GB, 15K RPM
Hitachi SAS, 300GB, 15K RPM

Service time (msec)

16K 32K 64K 128K 256K 512K 1M 2M 4M 8M 16M

VERIFY command size (bytes)

Tuesday, June 26, 2012
Staggered scrubbing guarantees fast LSE detection, but: seeking overhead never evaluated in implementation

Parameter 2: Scrubbing Order

- **Staggered scrubbing** (fast LSE detection) [Oprea, Juels FAST'10]
- **Sequential scrubbing** (used in the field today)

Can we afford this?
HDD capacities: CAGR +100%
Data transfer speed: CAGR +40%
Parameter 3: Number of Regions

Limit seeking between regions, but retain frequent disk passes

more, smaller regions!

fewer, larger regions?!?

Performance of two approaches equated

Slower LSE detection (longer disk passes)

Lower throughput (more seeking)

Scrubbing throughput (MB/sec)

Number of regions

(Staggered) Hitachi SAS, 300GB, 15K RPM
(Sequential)
(Staggered) Fujitsu SAS, 73GB, 15K RPM
(Sequential)
System Overview

Filesystem Layer

Generic Block Layer

Scrubbing Framework

I/O Scheduler

On-disk cache

Hard disk

VERIFY size: [64KB, 4MB]

Regions: ≥ 128

Tuesday, June 26, 2012
Minimizing Impact
System Overview

Filesystem Layer

Generic Block Layer

I/O Scheduler

On-disk cache

Scrubbing Framework

Verify size: [64KB, 4MB]

Regions: ≥ 128

Hard disk

Verify

Verify

Verify

Verify

Verify

Verify

Verify

Region

Tuesday, June 26, 2012
Background Scheduling

• **Fire VERIFY requests only when disk otherwise idle**
  - Previous work: *Focus on unobtrusive READS/WRITES* [Lumb'02, Bachmat'02]

• **Avoid collisions with workload requests**
  - Start time: *When should we start firing VERIFYs?*
  - Stop time: *When do we stop to avoid collision?*

• **Statistical analysis of idleness**
  - I/O traces: 2 systems, 77 disks, diverse workloads [SNIA, IOTTA repository]
Property: Long Tail - *Majority of idle time in few idle intervals* [Riska '09]

- **Fraction of largest idle intervals**
- **Fraction of total idle time**

- **Home**, **Lunch Break**, **Processing b/w requests**

**Idleness & Long Tails**
**Property:** Long Tail - *Majority of idle time in few idle intervals* [Riska '09]

**Predictor:** Waiting - *Fire past threshold*

---

**Idleness & Long Tails**

<table>
<thead>
<tr>
<th>Time</th>
<th>READ</th>
<th>WRITE</th>
<th>READ</th>
<th>VERIFY</th>
<th>VERIFY</th>
<th>READ</th>
<th>WRITE</th>
</tr>
</thead>
</table>

*Waiting Threshold (T_w)*
Idleness & Long Tails

Property: Long Tail - Majority of idle time in few idle intervals [Riska ’09]

Predictor: Waiting - Fire past threshold, stop only on collision

Waiting Threshold ($T_w$)
**Property:** Periodicity - *Repeating patterns in disk traffic*

**Predictor:** Autoregression - *Fire if prediction > threshold, don’t stop*
Predictor evaluation

Fraction of idle intervals picked by predictor vs. Fraction of idle time utilized by predictor

Optimal
Predictor evaluation

Oracle: always picks $X\%$ largest intervals
Predictor evaluation

Fraction of idle intervals picked by predictor

Fraction of idle time utilized by predictor

Prediction threshold ($T_p$)

Waiting threshold ($T_w$)

Oracle

Autoregression

Waiting

Tuesday, June 26, 2012
Fine-tuning the wait

**Verify request size**

**Waiting threshold**

- Scrubbing throughput (MB/s)
- Average delay per workload I/O request (ms)

- (512KB, 32ms)

Tuesday, June 26, 2012
Fine-tuning the wait

Complete Fair Queueing (CFQ):
- Default Linux I/O scheduler
- Always waits 10ms before declaring disk idle

Verifying request size

Waiting threshold

Scrubbing throughput (MB/s)

Average delay per workload I/O request (ms)

Best effort

Larger thresholds

CFQ I/O scheduler

Worst possible

512Kb
4096Kb
64Kb

Tuesday, June 26, 2012
System Overview

Filesystem Layer

Generic Block Layer

Scrubbing Framework

I/O Scheduler

Verify size: [64KB, 4MB]

Regions: \( \geq 128 \)

Verify request size

Waiting threshold

Time

On-disk cache

Hard disk

Tuesday, June 26, 2012
Also in the paper

• Scrubbing framework implementation
  - Open-sourced* framework simplifying development of scrubbing algorithms
  - Why VERIFY is implemented incorrectly in ATA drives
  - Scrubbing impact evaluation in implementation using synthetic/realistic workloads

• Detailed statistical analysis
  - Detection of longer tails than reported in previous work
  - Characterization of periods in traces
  - TPC-C benchmark: idle time distribution unrepresentative of OLTP workloads