Unobtrusive Storage Maintenance

George Amvrosiadis

Supervisory Committee
Angela Demke Brown, Ashvin Goel, Bianca Schroeder, Ryan Johnson

External Committee
Jason Flinn, Ding Yuan, Eyal de Lara
What is storage maintenance?

• Insurance for your storage system!
  – Pay upfront to avoid paying a lot later

• Maintenance tasks offer various guarantees

<table>
<thead>
<tr>
<th>Guarantee</th>
<th>Maintenance task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Scrubbing, Read-after-Write</td>
</tr>
<tr>
<td>Availability</td>
<td>Backup</td>
</tr>
<tr>
<td>Performance</td>
<td>Layout optimization</td>
</tr>
<tr>
<td>Security</td>
<td>Virus scanning</td>
</tr>
</tbody>
</table>

Joint work with: Nosayba El-Sayed, Ioan Stefanovici, Andy Hwang, Bianca Schroeder, Medha Bhadkamkar
Maintenance is essential

- Hardware failures are unpredictable
  E.g., HDD failure rates differ by operating environment, model

- Storage failures are expensive
  E.g., US consulates skip backups, no visas for 2 weeks after failure
When should maintenance be run?

• **Problem:** maintenance tasks can significantly impact foreground applications (workload)
  – Run alongside workload and access lots of data
    E.g., process all files, scan every block
  – Consequence: Increased seeks, cache pollution

• **Currently:** perform during scheduled downtime
Challenges of storage maintenance

• Downtime is expensive
  – Hard to provision: 1 in 5 backups fail due to limited downtime [ATC 2016]
  – Hard to schedule: Multiple time zones in the cloud

• Too much maintenance
  – 10 hours to scan 6TB hard drive today
  – Enterprises run full backups every 1-4 days [ATC 2015]
  – Too many tasks: backup, virus scanning, ...
Overview

• Storage maintenance in the field
• Scheduling maintenance I/O
  – Challenge: downtime is hard to provision, schedule
• Opportunistic storage maintenance
• Ongoing and future work
Scheduling maintenance I/O

- Perform maintenance when device is idle
- Need to avoid collisions with workload requests
  - Predict start/end of idle periods
- Find right predictor through I/O trace analysis
  - I/O traces: MSR and HP, 77 disks, diverse workloads
Characteristics of idleness (1)

**Long tails:** Majority of idle time in few periods [Riska09]

**Predictor:** Wait for a fixed time, stop on collision
Characteristics of idleness (2)

**Periodicity:** Repeating patterns in disk traffic

**Predictor:** Autoregression – *Fire if prediction > threshold*
Optimal Oracle: always picks X% largest intervals

Prediction threshold

Waiting threshold

Predictor evaluation

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

- Oracle
- Autoregression
- Waiting
Overview

- Storage maintenance in the field
- Scheduling maintenance I/O
- **Opportunistic storage maintenance**
  - Challenge: Too many tasks, too little time
- Ongoing and future work

*Joint work with: Angela Demke Brown, Ashvin Goel*
Too many tasks \[SOSP 2015\]

• Tasks often access the same data
  – E.g., backup and defrag the same file system
  – Caching should be able to exploit data reuse

• **Problem:** cached data gets replaced before reuse
  – Each task accesses more data than fits in cache
  – Tasks process data independently
Reducing maintenance I/O

- Insight: correctness is not tied to processing order
  - E.g., backing up file $f_1$ before, or after $f_2$ doesn’t matter
- Opportunistically process data cached by other tasks

<table>
<thead>
<tr>
<th>Application</th>
<th>Virus scanner</th>
<th>Backup process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_3$</td>
<td>$f_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$f_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$f_3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
</tr>
<tr>
<td>$f_2$</td>
</tr>
<tr>
<td>$f_3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
</tr>
<tr>
<td>$f_2$</td>
</tr>
<tr>
<td>$f_3$</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

How can a task find out what’s in the OS cache?
Exposing cache information

- Operating systems cache data at page granularity
- Need to expose which pages are cached

1) We track changes to the status of cached pages
2) tasks register interest
3) tasks poll for events

<table>
<thead>
<tr>
<th>Page status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added</td>
<td>Added in cache</td>
</tr>
<tr>
<td>Removed</td>
<td>Removed from cache</td>
</tr>
<tr>
<td>Dirtied</td>
<td>Dirty bit set</td>
</tr>
<tr>
<td>Flushed</td>
<td>Dirty bit cleared</td>
</tr>
</tbody>
</table>

Tell me about...
- Added
- Removed

Occurring at...

Page event
- Inode #
- Page offset
- Page status
The Duet Framework

User-level

Kernel

Page cache layer

Duet hooks

Page status changes

Duet kernel module

Pending page events

Application layer

Backup task

Filesystem layer

Defrag task

Page events

How do tasks use page events?
Using page cache events

• Tasks operate on **items** of different granularities, e.g. *files, extents, blocks, segments, …*

• Page events expose cached portion of an item

• Example: file defragmentation task
  – Uses *Added, Removed* events to track cached file pages
  – Prioritizes processing of files with most cached pages

• **Opportunistic processing reduces required I/O**
Evaluation

Scrubbing, backing up, de-fragmenting **concurrently**

- Tasks can piggyback on one another
- Less idle time needed for maintenance
Overview

• Storage maintenance in the field
• Scheduling maintenance I/O
• Opportunistic storage maintenance
• Ongoing and future work
Opportunistic cluster computing

• **Problem:** factor cache residency in scheduling

• **Quartet:** use Duet events in Hadoop, Spark
  – Jobs consist of tasks consuming chunks of input
  – Prioritize tasks interested in data cached across cluster

• Early evaluation: 54% *increase* in cache hit rate
Granular file notifications

• **Problem:** Notification frameworks offer file events
  – Applications have to scan files for changes (chunking)
  – File synchronization app: 92% of time spent chunking

• **Goal:** Finer granularity using Duet events
  – Focus chunking on offsets with Duet Modify events

• Early evaluation: 38% reduction in chunking time
Future work

• Opportunistic cluster computing
  – Evaluate against (simulated) real workloads
  – Augment real applications, e.g. learning algorithms

• Granular file notifications
  – Study file access patterns in the field

• Support data cached via direct or remote I/O
  – Handle application-managed caches, e.g. buffer pools
  – Reuse data brought over the network