Opportunistic Storage Maintenance

George Amvrosiadis, Angela Demke Brown, Ashvin Goel
University of Toronto

UNIVERSITY OF TORONTO

Importance and challenges of storage maintenance

- Storage maintenance task characteristics
  - Run periodically in the background
  - Access large amounts of data
  - Operate in the user or kernel level
- Maintenance tasks offer critical guarantees

Guarantee	Periodic maintenance task
Reliability	Scrubbing, Write Verification
Availability	Backup, Data reorganization
Performance	Layout optimization
Security	Virus scanning
Storage Efficiency	Deduplication, Garbage collection

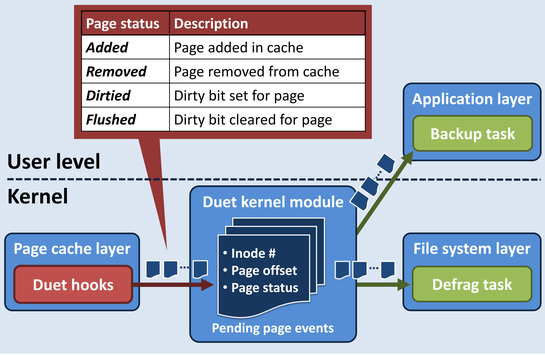
- **Problem:** maintenance impacts applications
  - Causes cache pollution, longer disk seek times
- **Solution:** Schedule tasks during idle times
- **Challenge:** Too little idle time available
  - Less idle time in the cloud
  - Maintenance takes too much time
    - Full backups are performed every 1-4 days
    - 10 hours to scan an enterprise 6TB HDD
  - **Too many tasks, working independently**
    - **Total I/O proportional to number of tasks**

The Opportunistic Storage Maintenance Model

Goal: Reduce maintenance I/O by enabling tasks to work synergistically

- Maintenance tasks often access same data
  - Caching should be able to exploit data reuse
- **Problem:** Cached data is replaced before reuse
- **Insight:** Tasks can process data in any order
- **Approach:** Adapt task processing to operate on cached data first

The Duet Framework



- Duet exposes page cache info to tasks
  - Tracks changes to the status of cached pages
- Tasks poll to receive page events
  - Use events to process data more efficiently
- **Example:** file defragmentation task
  - Uses Added, Removed page events to track cached file pages
  - Processes files with most cached pages first

```
1  Algorithm example_task()
2    while True do
3      /* Process files opportunistically */
4      while True do
5        duet_fetch(sid, events)
6        prioqueue_update(sid, pq, events)
7        file ← prioqueue_dequeue(pq)
8        handle_file(file)
9      end
10     file ← pick_next_file()
11     handle_file(file)
12   end
1  Procedure handle_file(file)
2    if duet_check_done(sid, file) then return
3    process_file(file)
4    duet_set_done(sid, file)
```

Fetch any new page events

Use events to update priority queue of files with most cached pages

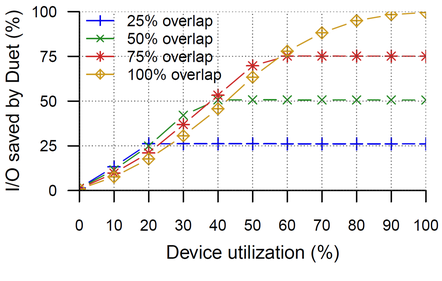
Pick file in task's standard order

Ensure file hasn't been processed before

Evaluation Results

Single task alongside workload

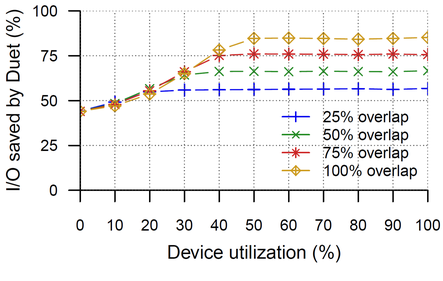
(Backup)



- Maintenance I/O is reduced based on:
  - Data overlap with workload, higher device utilization

Running multiple tasks together

(Scrubbing, Backup, and Defragmentation)



- Tasks can piggyback on one another
  - Running 3 tasks together reduces I/O by up to 80%

Duet exploits all opportunities to save I/O ⟶ Less idle time is needed for maintenance

Contact
George Amvrosiadis
Dept. of Computer Science, University of Toronto
gamvrosi@cs.toronto.edu www.cs.toronto.edu/~gamvrosi

Duet source code
github.com/gamvrosi/duet