STOW: Spatially and Temporally Optimized Write Caching Algorithm

Binny S. Gill, Michael Ko, Biplob Debnath, Wendy Belluomini
IBM Almaden Research Center, University of Minnesota
Prior Art: Write Cache Algorithms

- An eviction problem (like read caches)
- Goal: Keep the disk heads busy for the least time
- Some exploit temporal locality
  - To reduce number of destages
  - LRU, CLOCK, FBR, LRU-2, 2Q, LRFU, LIRS, MQ, ARC, CAR
- Some exploit spatial locality
  - Apply temporal locality rules to larger units
  - Tracks (multiple pages), stripes (multiple tracks)
- Some create spatial locality via reordering
  - To reduce the average cost of destages
  - SSTF, SATF, SCAN, CSCAN, LOOK, VSCAN, GSTF, WSTF
- Some do all of the above: WOW (earlier work)
WOW Algorithm

CLOCK

SORTED CLOCK

CSCAN

Head and Tall are glued to form a WOW
Is there more to it?

The 5 properties a good write cache serving disks needs to have:

- Harness temporal locality
- Create spatial locality
- Maintain free space
- Distribute the write load uniformly over time
- Also serve read hits
What about the Destage Rate?

- Most cache research revolves around the eviction or destage order problem

- *Destage rate* is under-studied, but surprisingly is extremely important for performance

- If you can tame the destage rate, there is another gold mine beyond the benefits of WOW

- We had to invent a new destage order (STOW) to control the destage rate

- STOW becomes the first write caching algorithm to explicitly allow a good destage order and a good destage rate = a powerful combination
Write Cache Tutorial: How to get it wrong?

- **Ignore RAID Parity Groups while destaging**
  - We need to destage all members of the same parity group together to the RAID array, not spread out in time
  - Simple but important
  - WOW already groups based on RAID stripes
Tutorial: Destage rate = as quickly as you can

Destage Order = WOW

SPC1- Like Workload
Tutorial: Destage rate = as quickly as you can only when the cache occupancy reaches a fixed Threshold

Destage Order = WOW

Destage rate toggles between none and full force

SPC1-Like Workload
Tutorial: Destage with Linear Thresholding

- **Destage Rate**
- **Cache Occupancy**
  - 100% Full
  - High Threshold
  - Current Occupancy
  - Low Threshold
  - 0% Full
Tutorial: Destaging with Linear Threshold

Linear threshold cannot keep cache away from 100% full

“Spikes” are due to long time spent in sequential and random regions

Time spent at 100% is bad. Spikes make write burst absorption and destage rate suffer.
Separate Random and Sequential data

Spikes are gone .. now there are two active areas on the disk platters => destage order suffers
Getting Warmer: Add hysteresis to the destages

- **SeqQ**
- **RanQ**

Split 50-50

- **HysteresisCount** = 128 * number of spindles in RAID array

But what if workload has no sequential or random?

Focus on one region of the disk platters for some time before moving to the next region => minimize the negative impact on destage order
STOW: Adapting the size of RanQ and SeqQ

- Queue sizes are adapted according to workload
  - DesiredSeqQSize -= : Whenever a second write happens in a RAID stripe in RanQ
  - DesiredSeqQSize += n * |RanQ|/|SeqQ| : Where, n = number of spindles in array
    - Whenever there is a break in the LBA sequence of destages from SeqQ
  - If |SeqQ| > DesiredSeqQSize, then destage from SeqQ, else destage from RanQ
### STOW vs Competition

Sized are dynamically adapted according to real-time marginal utilities.

<table>
<thead>
<tr>
<th></th>
<th>CSCAN</th>
<th>LRW</th>
<th>WOW</th>
<th>STOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial Locality</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Temporal Locality</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Scan Resistance</td>
<td>No</td>
<td>No</td>
<td>Little</td>
<td>Yes</td>
</tr>
<tr>
<td>Stable D'estage Rate</td>
<td>No</td>
<td>Little</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Stable Occupancy</td>
<td>No</td>
<td>Little</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Experimental Setup

Full Backend = All Disk Capacity Targeted
Partial Backend = Outer 1% of disk capacity targeted

SPC-1 Like Benchmark
STOW: No more spikes in cache occupancy

RAID 5 Partial Backend: target 3500 IOPS, threshold: 70/40
Full Backend: Throughput vs. Response Time

![Graph showing throughput and response time for different RAID configurations](image)
Partial Backend: Throughput vs. Response Time

- STOW (H/L:90/80)
- STOW (H/L:70/40)
- WOW (H/L:90/80)
- WOW (H/L:70/40)
- CSCAN (H/L:90/80)
- CSCAN (H/L:70/40)
- LRW (H/L:90/80)
- LRW (H/L:70/40)

RAID 5

- Avg. Response Time (ms)
- IOPS

Key:
- 160%
- 24%
- 12%
Vary the spread between high and low thresholds
Vary the cache size

RAID 5, Full Backend: Target 1050 IOPS ; H/L : 90/80
Summary

- Tackling both destage order and destage rate = powerful write cache algorithm

- **STOW**
  - Leverages temporal locality
  - Creates spatial locality
  - Maintains steady free space to absorb write bursts
  - Destages uniformly
  - Protects Random data from Sequential bursts
  - Dynamically adapts the sizes of the sequential and random portions of the cache to maximize throughput

- **STOW > WOW > (LRW, CSCAN)**

- **Is there still more to it? :)**