AWOL: An Adaptive Write Optimizations Layer

Alexandros Batsakis, Randal Burns
The Johns Hopkins University
Arkady Kanevsky, James Lentini, Thomas Talpey
Network Appliance(TM)Inc

FAST '08: 6th USENIX Conference on File and Storage Technologies

Outline

- Introduction
- Adaptive Write Scheduling
- Evaluation
- Conclusions

Introduction (1/3)

- Traditionally, reads have been considered more important than write
  - Reads are synchronous
  - Writes are asynchronous

Introduction (2/3)

- However, the synchronous reads depend upon the asynchronous writes, because
  - Dirty pages consume memory that is unavailable for read caching
  - Write traffic to clean pages interferes with read requests

Introduction (3/3)

- When to destage dirty pages to storage ?
  - Use a periodic update policy in which individual dirty blocks are flushed when their age reaches a predefined limit
  - When the number of dirty pages in memory exceeds a certain percentage (*dirty_background_ratio*)

Adaptive Write Scheduling

- A Write-Only Cache
- A Read-Write Cache
- Opportunistic Queuing

A Write-Only Cache (1/2)

- The goal
  - Keep as many dirty buffered pages as possible
- The High-low watermark algorithm
  - When the high threshold is crossed, the memory manager starts writing data out to disk until the percentage of dirty blocks is below the low threshold
- Drawback
  - Both thresholds are time-invariant

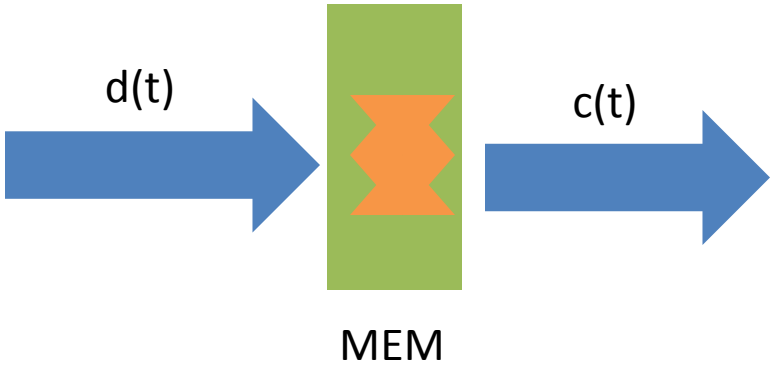
A Write-Only Cache (2/2)

- h(t)
  - The value of the time-variant high watermark
- l(t)
  - The value of the time-variant low watermark

$$h(t) = h(t-1)\frac{d(t-1)}{d(t)}$$

$$l(t) = l(t-1)\frac{d(t-1)}{d(t)}\frac{c(t)}{c(t-1)}$$

- d(t)
  - The rate that processes are dirtying new pages
- c(t)
  - The rate that memory manager is able to clean pages

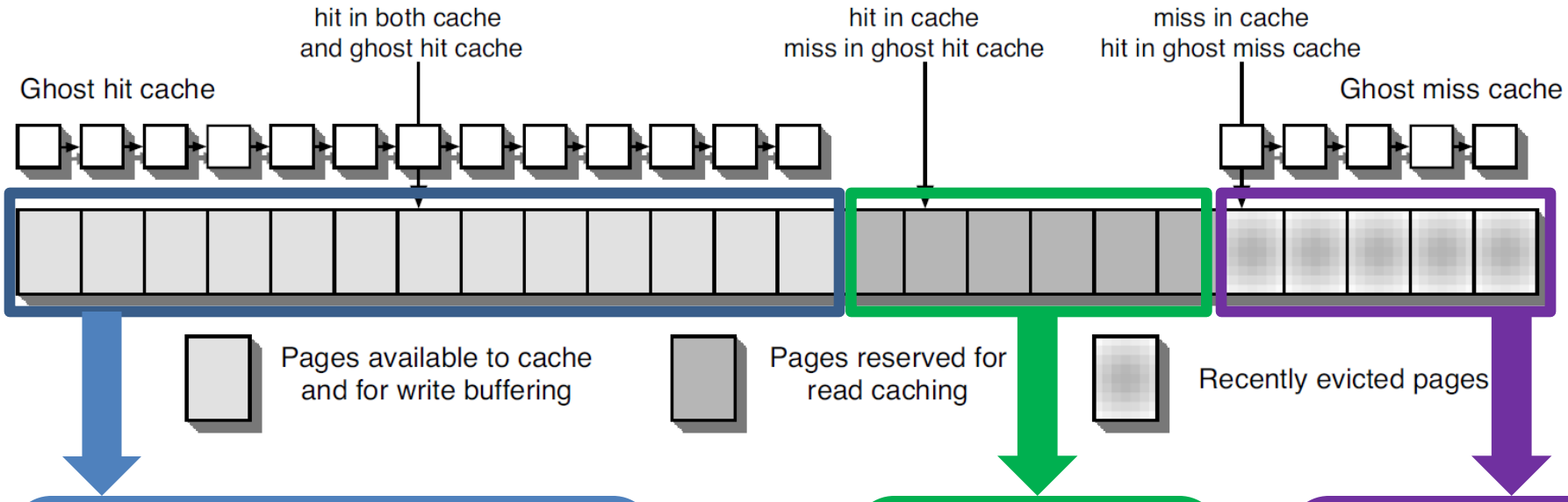


- (1/2)h(t – 1) ≤ h(t) ≤ 2h(t – 1)
- (1/2)l(t – 1)≤l(t)≤ 2l(t – 1)

A Read-Write Cache (1/5)

- The goal
  - Keep as many dirty buffered pages as possible
  - Maximize the read hit rate
- H_{max}
  - The maximum possible occupancy of memory with dirty pages before the write-back starts
- Use two ghost caches
  - Ghost miss cache
  - Ghost hit cache

A Read-Write Cache (2/5)



Contain *all of the write* buffered pages and the most recent/frequent read cache pages

Contain the least recent/frequent read cache pages

Contain the recently evicted cache pages

A Read-Write Cache (3/5)

- Ghost miss cache
  - Hold meta-data information on blocks recently evicted from the cache
  - Record the history of a larger set of blocks than can be accommodated in the actual cache
  - Keep an index of the blocks that were replaced as a result of write buffering only

A Read-Write Cache (4/5)

- The drawback of ghost miss cache
  - The signaling of over-buffering comes too late
- Ghost hit cache
  - Detect the potential negative effects of aggressive write buffering prior to incurring penalties from cache misses

A Read-Write Cache (5/5)

- reads(t)
  - The number of total read requests
- C(t)
  - The number of hits in the page cache
- GH(t)
  - Ghost hit cache
- GM(t)
  - Ghost miss cache

The read requests that fall into the reserved area and in recently evicted pages

$$h_{max}(t) = h_{max}\left(1 - \frac{C(t) - GH(t) + GM(t)}{reads(t)}\right)$$

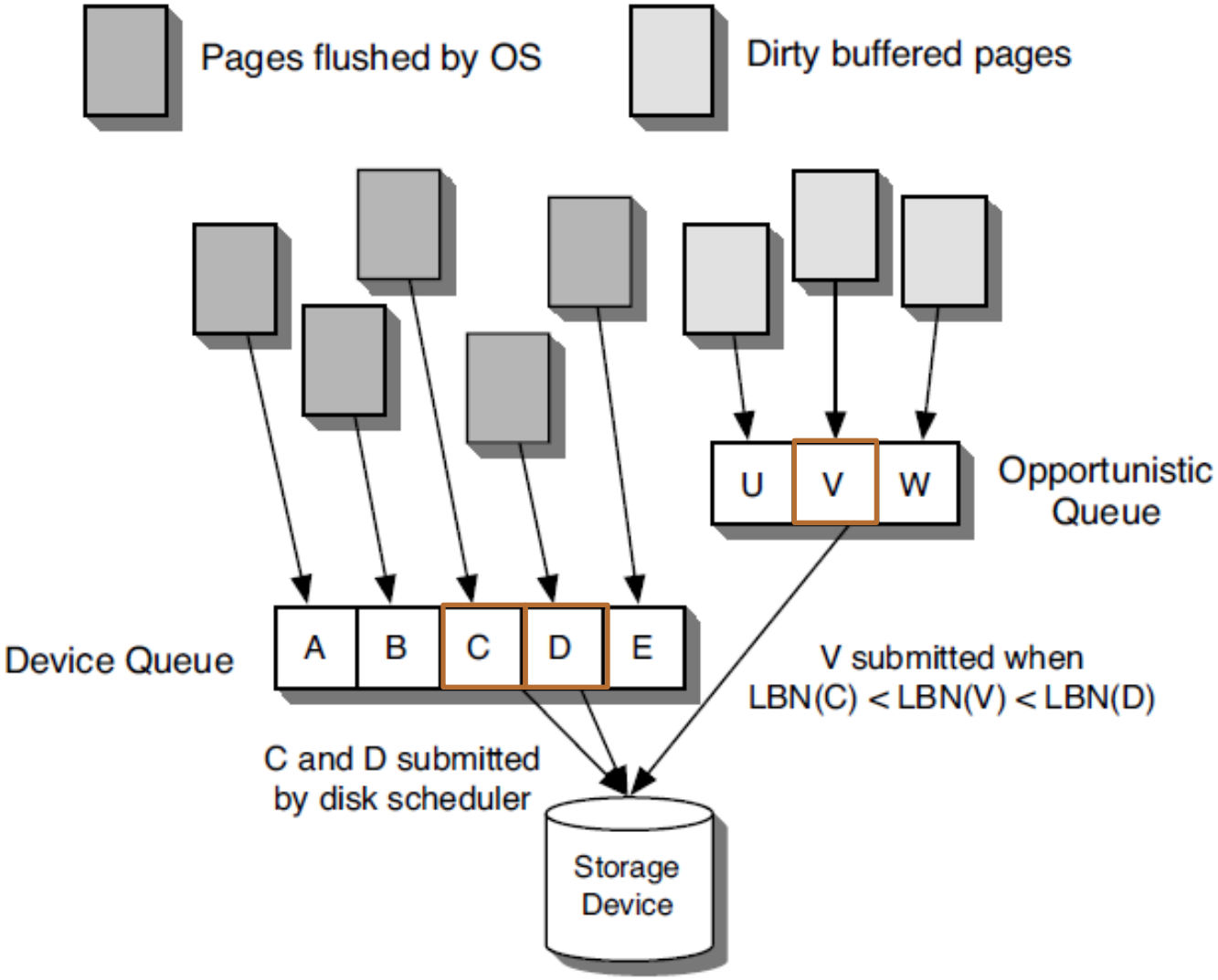
Opportunistic Queuing (1/3)

- Decide what to destage
- In two queues for read and write requests, the I/O scheduler keeps the list of pending I/O requests sorted by logical block number

Opportunistic Queuing (2/3)

- Opportunistic queue
  - Maintain an LBN sorted list of pages that are in memory and have not yet been submitted to the device for write-back

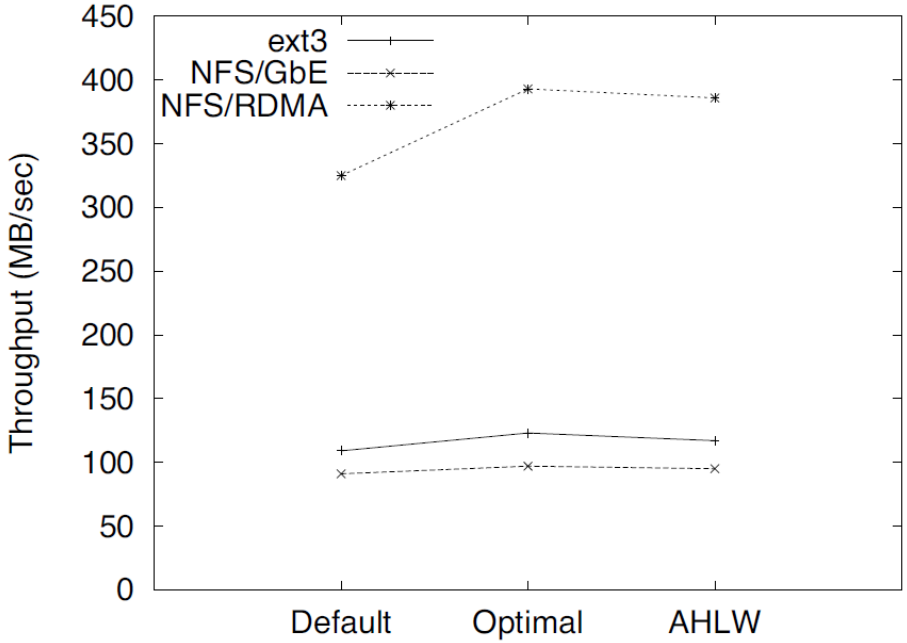
Opportunistic Queuing (3/3)



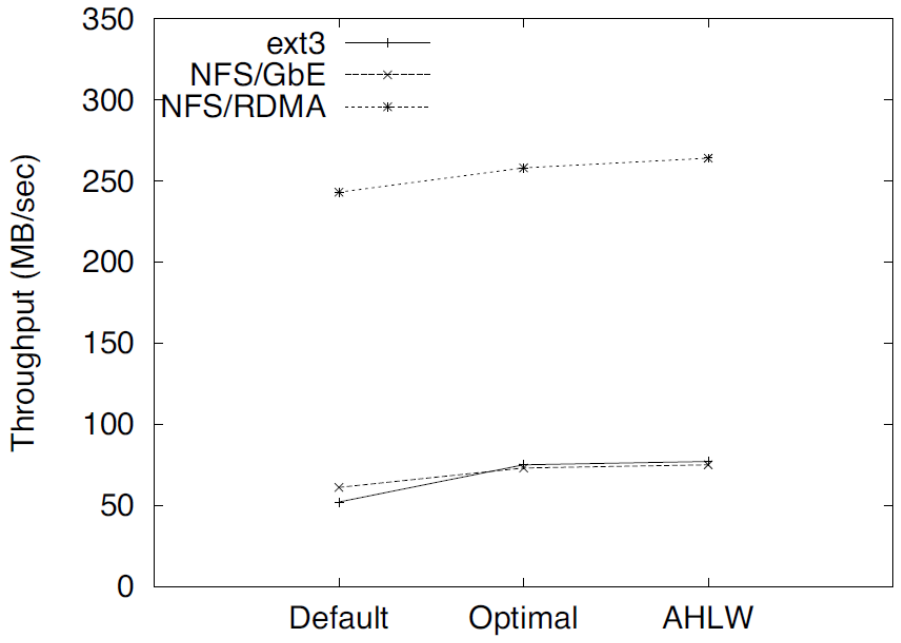
Evaluation

- Environment
  - A dual-core Xeon machine with 2GB of RAM out of which about 1.5GB can be used for the page cache
  - Network File System (NFS) using two different networks: gigabit Ethernet and 10-Gbps Infiniband
  - IOzone microbenchmarks
    - No Reader, Sequential Writer (NRSW)
    - No Reader, Zipf Writer (NRZW)
    - No Reader, Variable Writers (NRVW)
    - Sequential Reader, Sequential Writer (SRSW)
    - Random Reader, Random Writer (RRRW)
    - Zipf Reader, Zipf Writer (ZRZW)
    - Variable Readers, Variable Writers (VRVW)

The Adaptive High-Low Watermark Algorithm

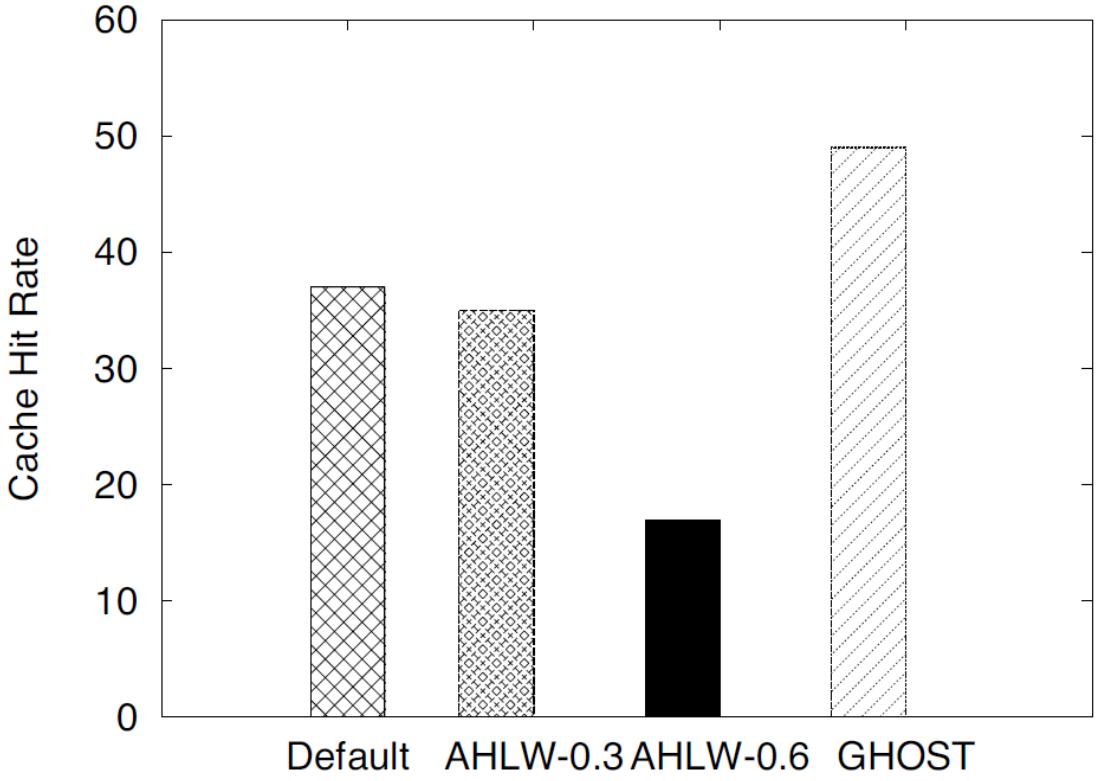


- Throughput of a sequential writer (NRSW)



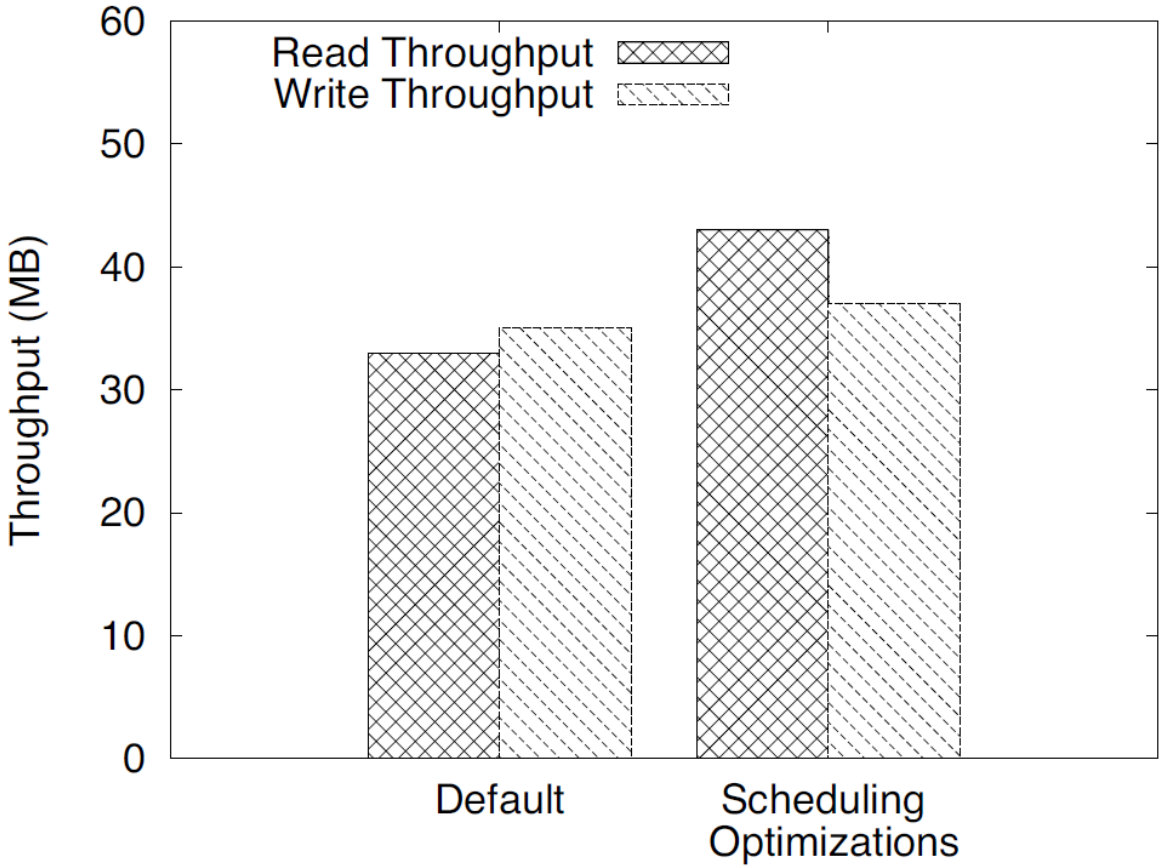
- Throughput of a Zipf-distribution writer (NRZW)

The AHLW Algorithm with Ghost Caching



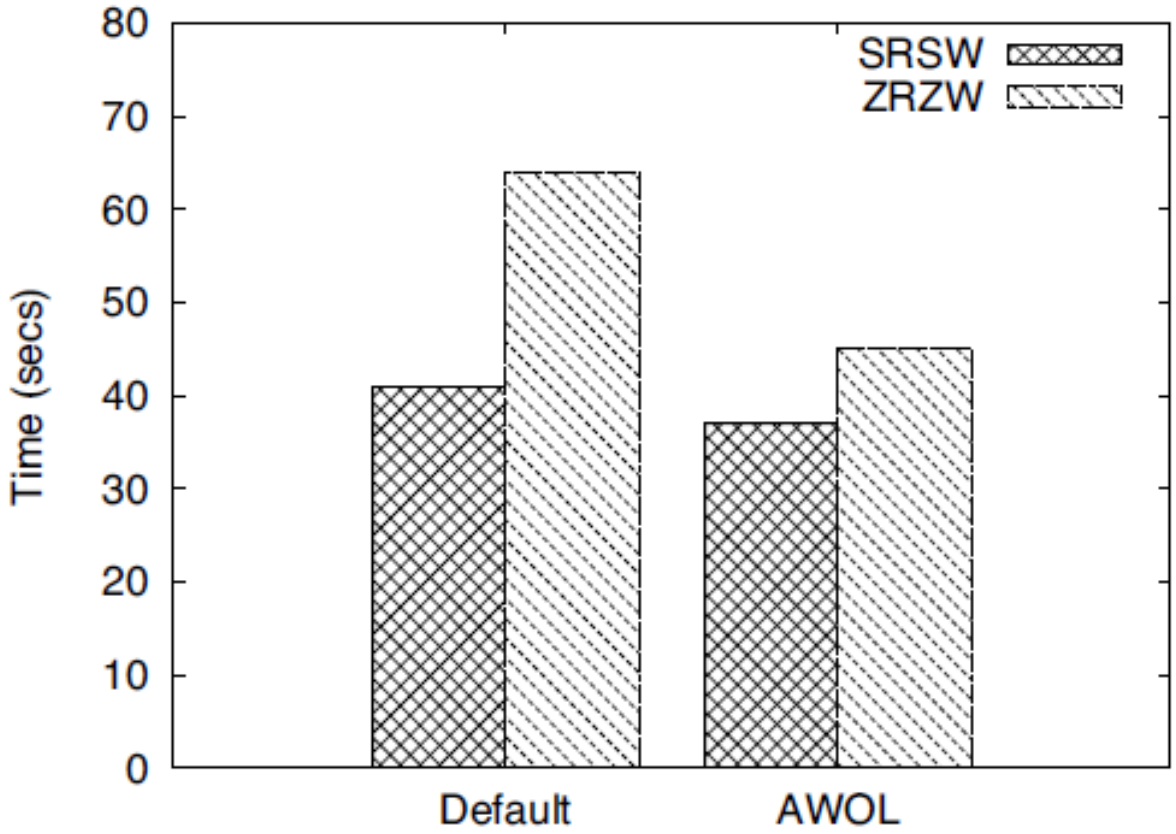
- Comparison of cache hit rates for Default, AHLW and GHOST (ZRZW)

I/O Scheduler Optimizations



- Comparison of the read and write throughput with and without opportunistic queuing (RRRW)

Putting it All together



- Performance of the ZRZW and SRSW benchmarks for Default and AWOL

Conclusions

- The modifications to the memory manager and I/O scheduler enable the system to automatically tune the destaging process, depending on the workload