Flash Caching on the Storage Client

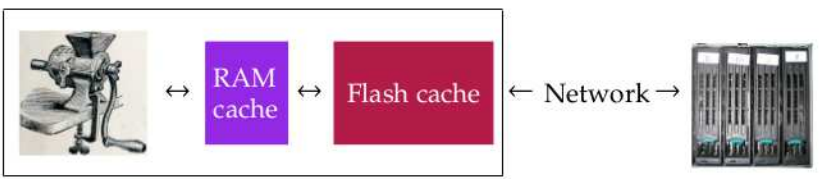
David A. Holland, Elaine Angelino,
Gideon Wald, Margo I. Seltzer

June 26, 2013

Client-Side Caching

Today flash is mostly used on the server side.

We looked at the client side of the network:



This matches a number of real-life environments.

Advantages: reduce latency and filer load

Considerations

Can the flash be write-through?

How tightly do we have to integrate the flash cache?

Does the flash cache need to survive crashes?

What (else) about cache consistency?

What We Did

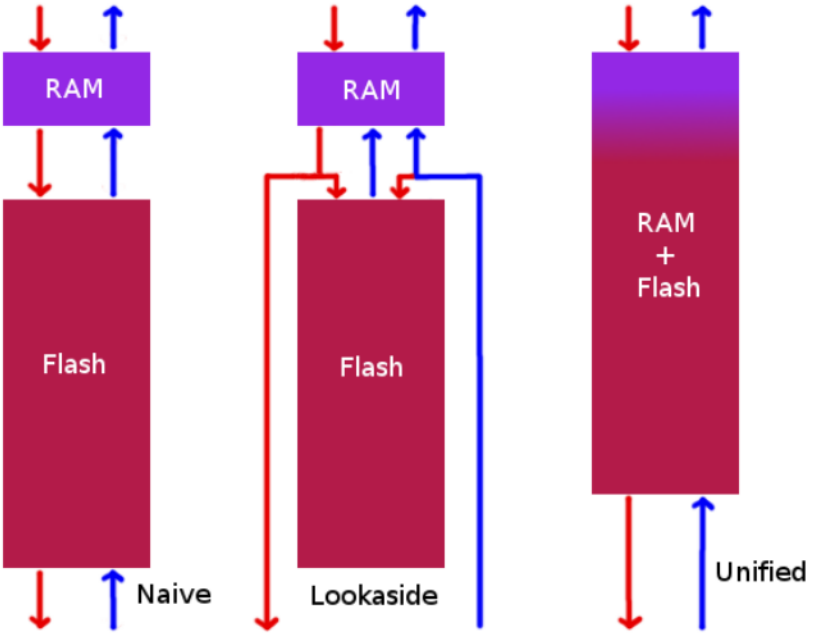
Because the potential design space is enormous, we turned to trace-driven simulation.

We validated the simulator with real traces.

Our results are from generated traces.

Cache Knobs

- RAM and flash sizes
- RAM and flash writeback policy
  - `s` - synchronous write-through
  - `a` - asynchronous write-through
  - `p`*N* - periodic *N*-second syncer
  - `n` - none, capacity evictions only
- Cache architecture
  - naive, lookaside, or unified


RAM
Flash
Naive
RAM
Flash
Lookaside
RAM
+
Flash
Unified

Hardware Parameters

- Filer read-ahead performance (90% by default)
- Low-level timings
  - RAM (fixed)
  - Flash
  - Network (gigabit)
  - Filer

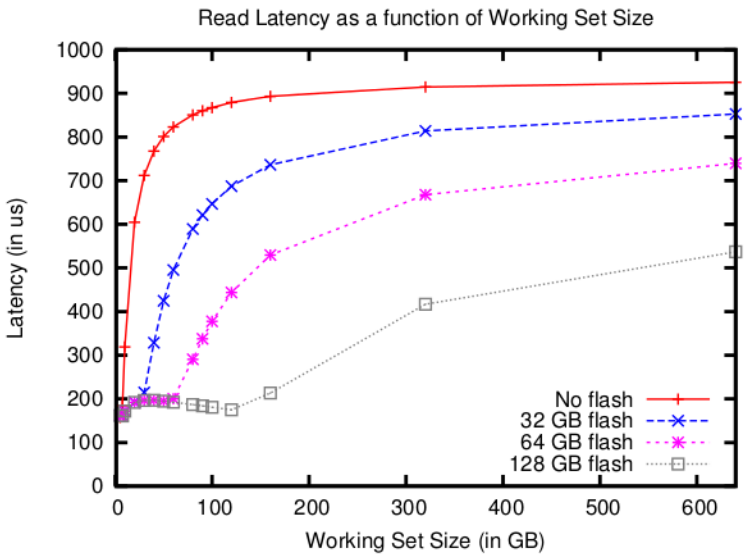
Workload Parameters (defaults)

- Total size of everything on the file server (1 TB)
- Number of working sets (1) and their size (60, 80 GB)
- Number of client hosts (1) and threads (8)
- Fraction of I/O outside the working set (20%)
- Fraction of writes (30%)
- Total I/O volume (pegged to working set size)

Results Outline

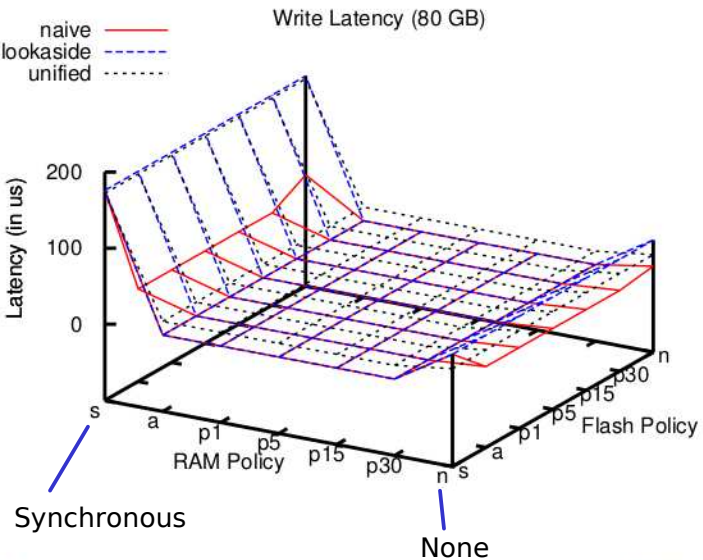
- Is the flash cache a win?
- What should the writeback policy be?
- Is the naive architecture good enough?
- Does the cache need to be persistent?
- What about cache consistency?
- Anything else...

1. Yes, the flash cache is a win.



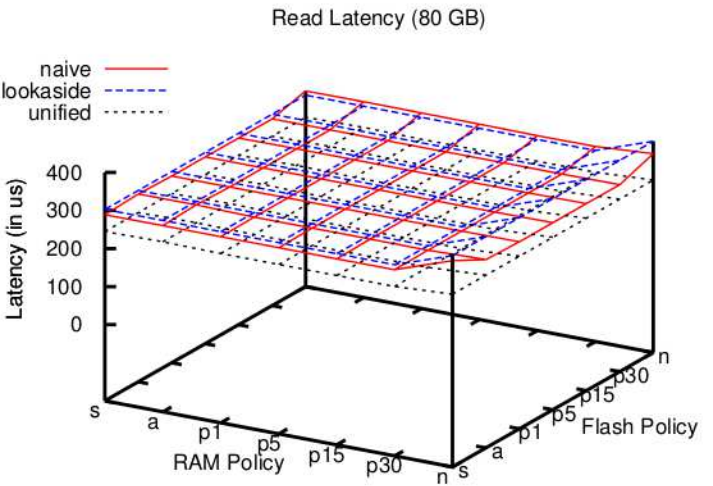
8 GB RAM
30% writes

2. Writeback policy doesn't matter.

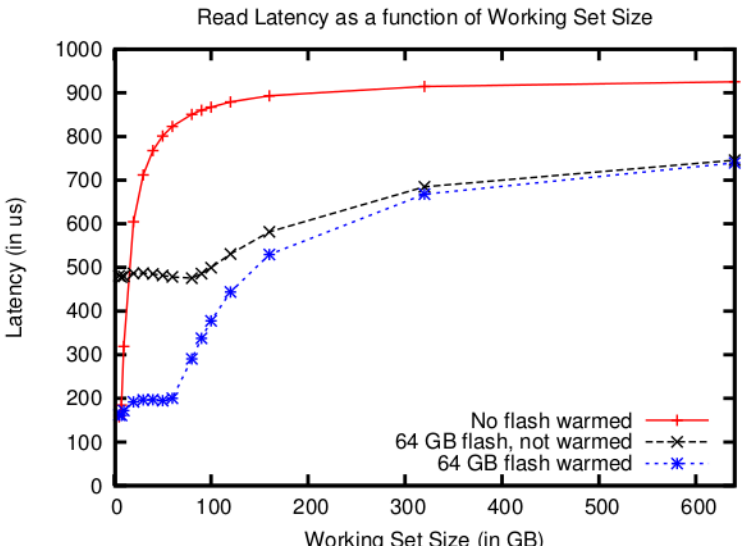


8 GB RAM
64 GB flash
30% writes

3. The naive architecture is fine.

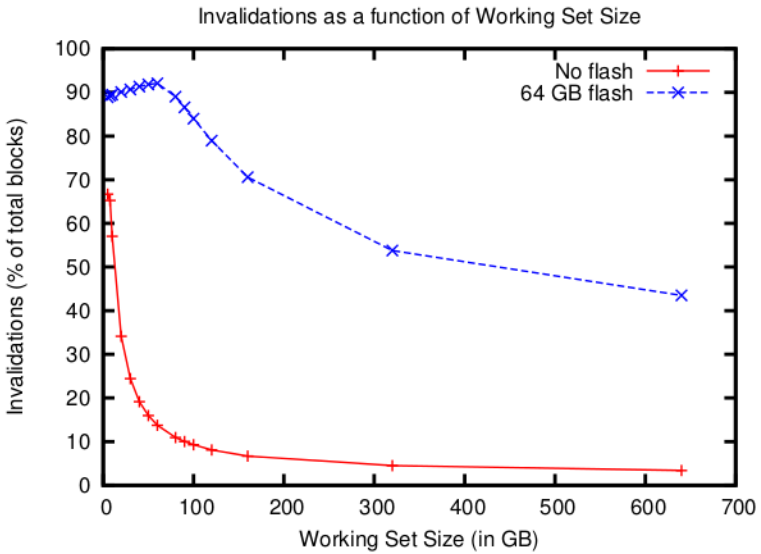


4. Persistence is nice but not critical.



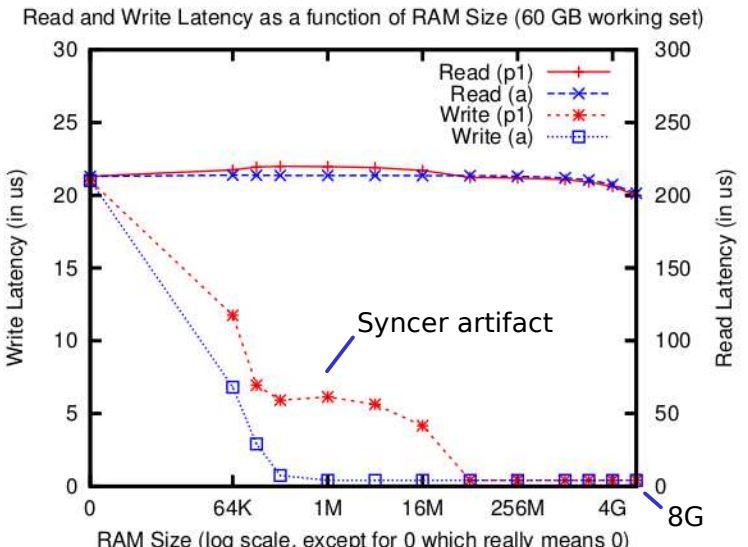
8 GB RAM
30% writes

5. Consistency does matter.



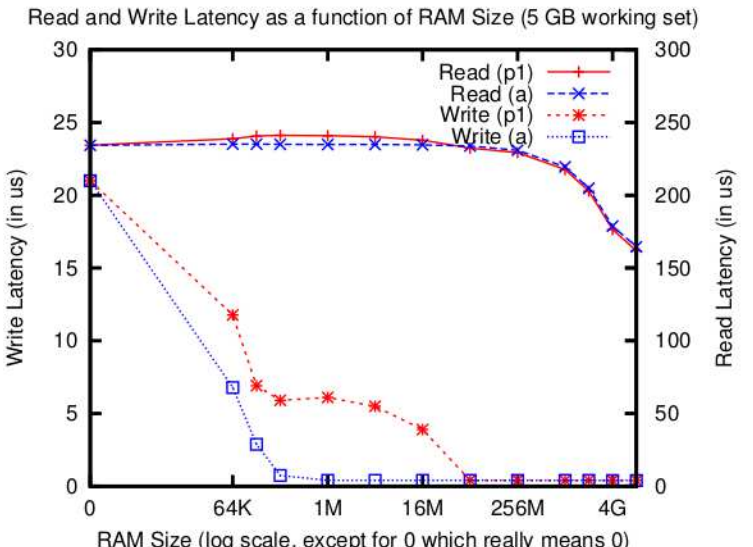
2 hosts
1 working set
8 GB RAM
30% writes

6. Use your RAM for other stuff.



30% writes
20% I/Os
from whole
file server

...maybe even for small workloads.



Conclusions

- Client-side flash caches are a pretty big win.
- It is ok for the cache to be write-through.
- The cache doesn't need to be integrated with the file system.
- Persistence isn't necessary but seems worthwhile.
- Some open consistency issues remain for shared data.

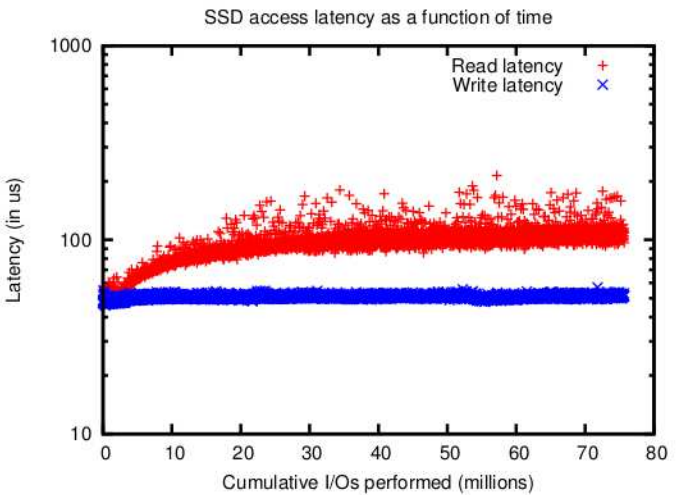
Flash Caching on the Storage Client

David A. Holland dholland@eecs.harvard.edu
Elaine Angelino elaine@eecs.harvard.edu
Gideon Wald gideon.wald@gmail.com
Margo I. Seltzer margo@eecs.harvard.edu

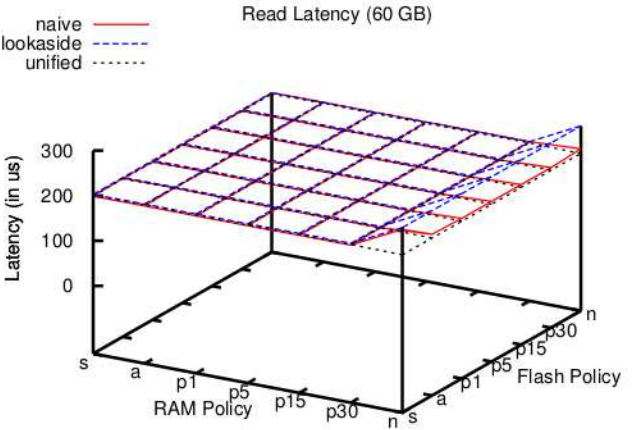
Timing Parameters

Parameter	Value
RAM read	400 ns / 4K block
RAM write	400 ns / 4K block
Flash read	88 μs / 4K block
Flash write	21 μs / 4K block
Network base latency	8.2 μs / packet
Network data latency	1 ns / bit
File server fast read	92 μs / 4K block
File server slow read	7952 μs / 4K block
File server write	92 μs / 4K block
File server fast read rate	90%

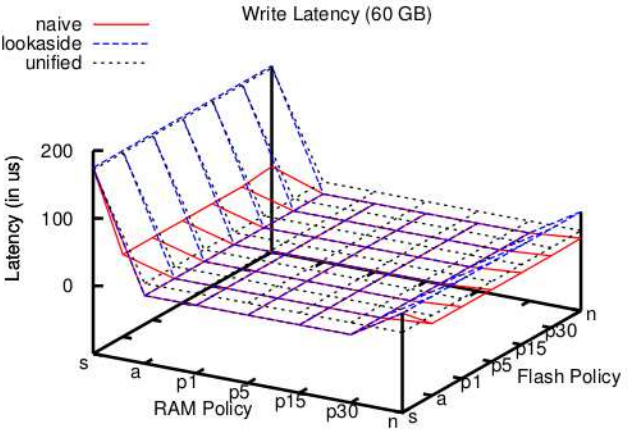
Flash device access latency



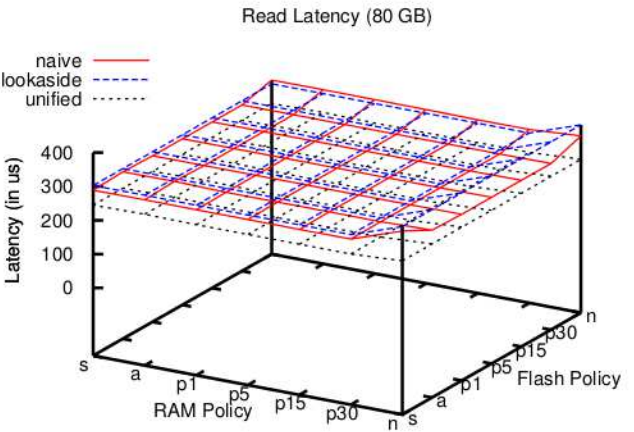
Read latency / policy (60GB trace)



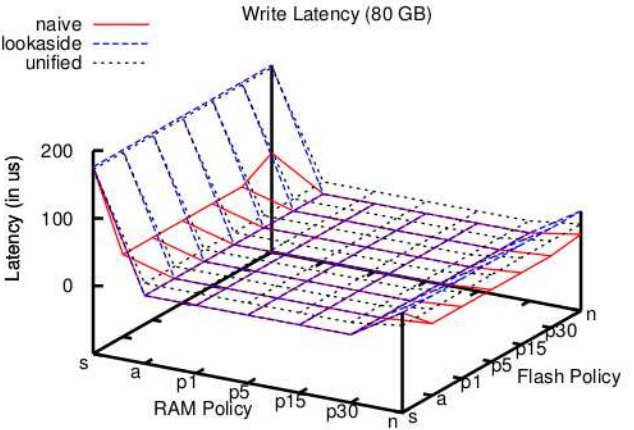
Write latency / policy (60GB trace)



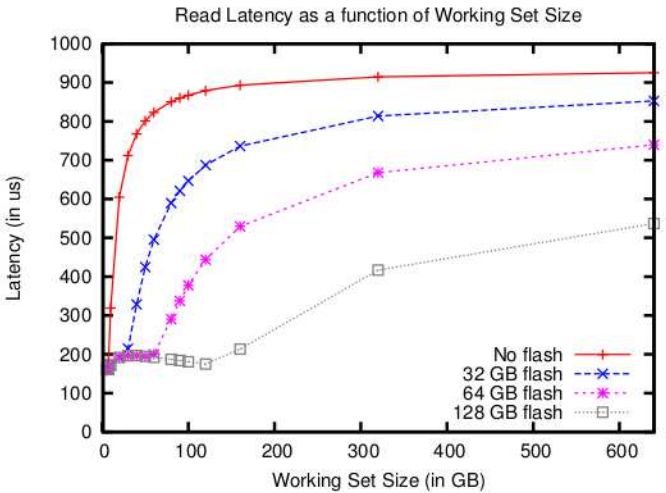
Read latency / policy (80GB trace)



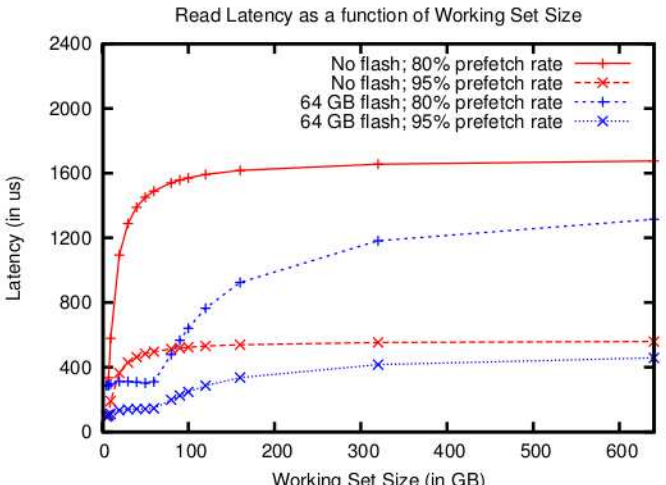
Write latency / policy (80GB trace)



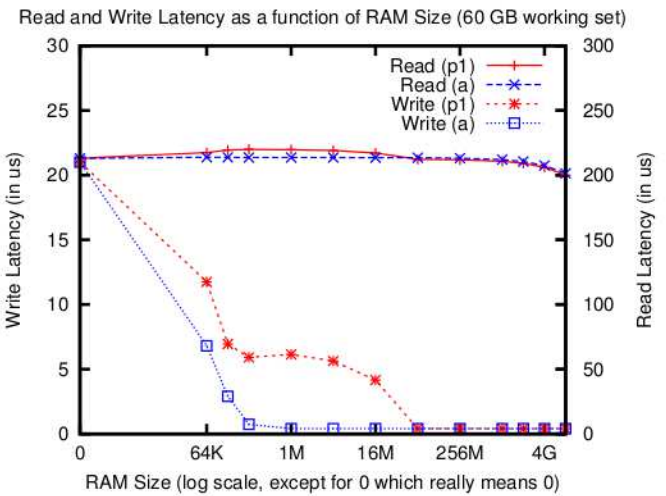
Read latency / WSS (Flash sizes)



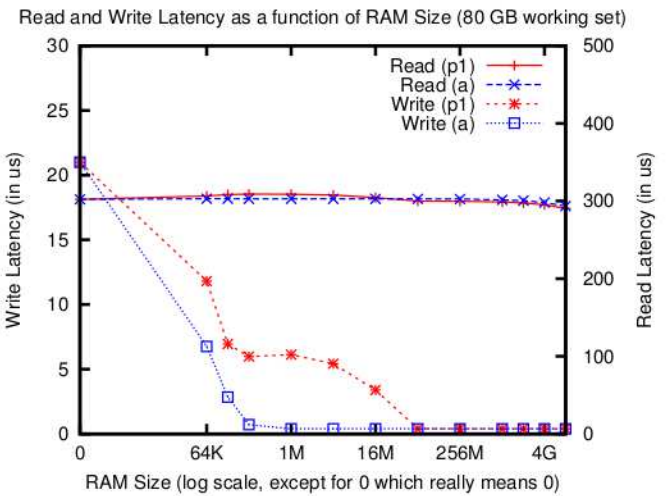
Read latency / WSS (Prefetch)



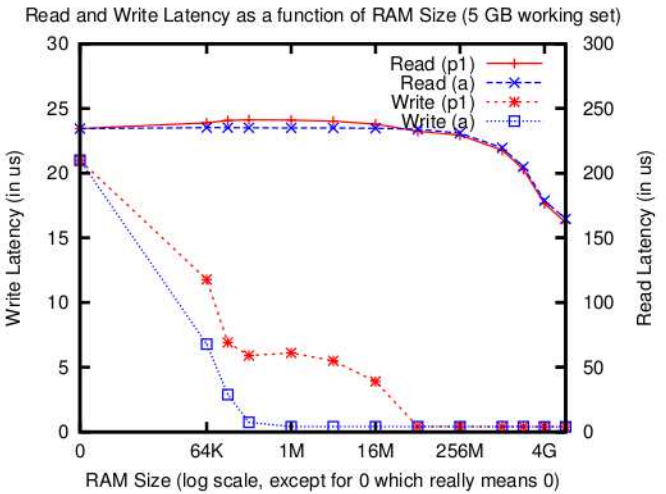
Latency / RAM size (60GB trace)



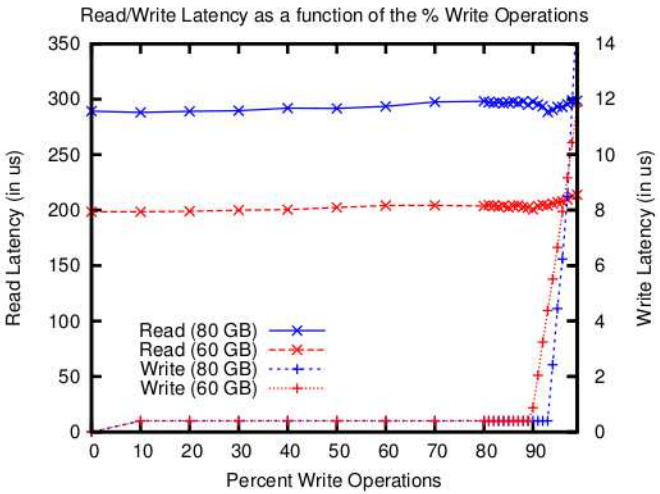
Latency / RAM size (80GB trace)



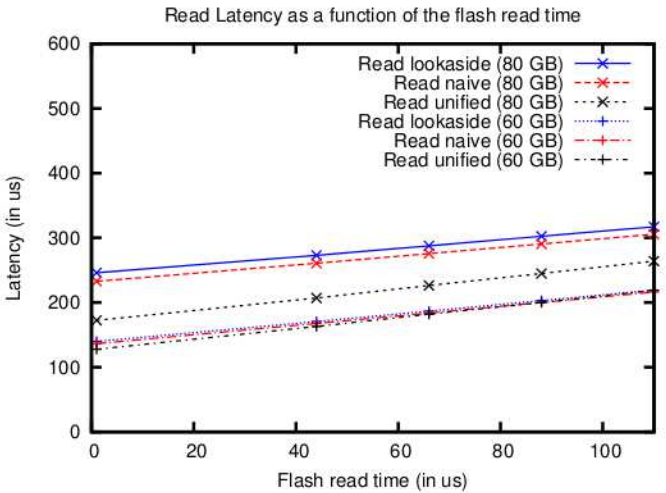
Latency / RAM size (5GB trace)



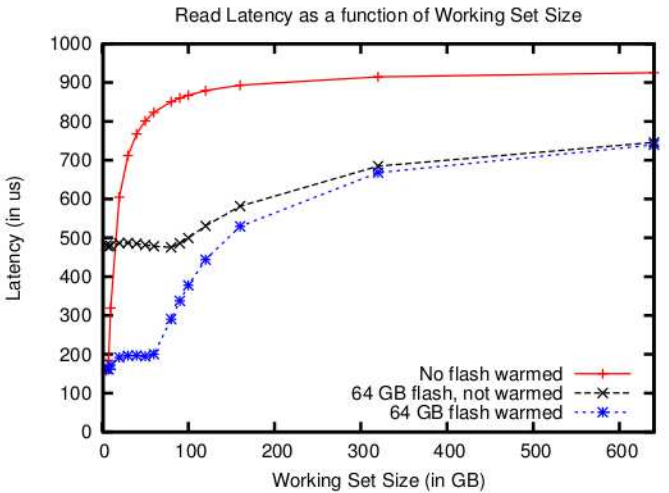
Latency / write percentage



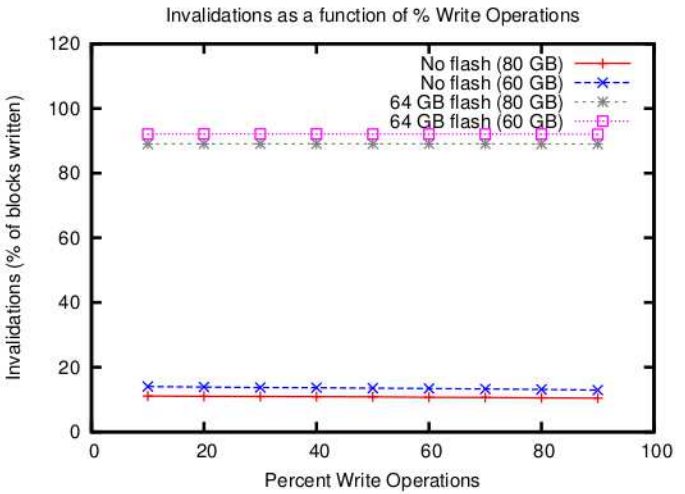
Read latency / flash read time



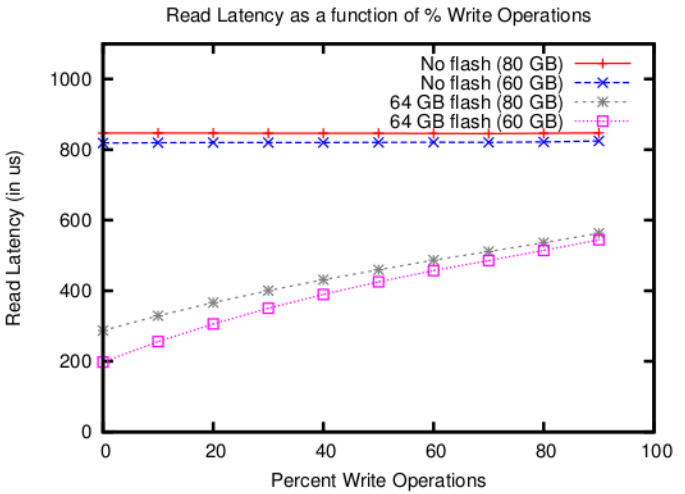
Read latency / WSS (Persistence)



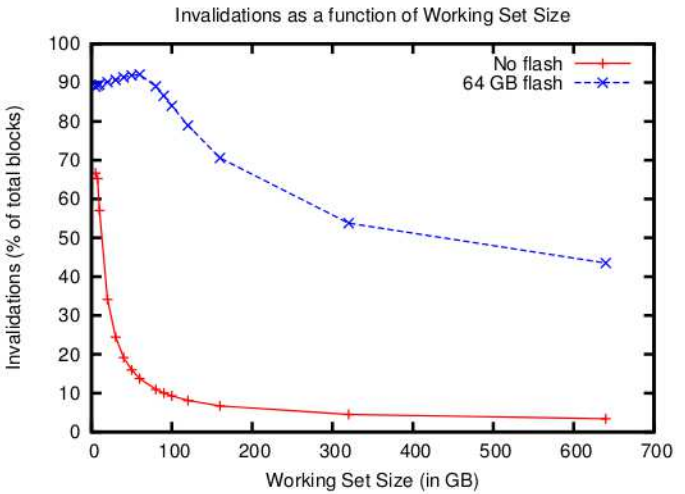
Invalidations / W% (Consistency)



Read latency / W% (Consistency)



Invalidations / WSS (Consistency)



Read latency / WSS (Consistency)