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
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.

[Graph showing read latency as a function of working set size for different flash cache sizes: 0 GB (red), 32 GB (blue), 64 GB (magenta), 128 GB (gray). Each curve represents a different cache size, with lower latency for larger cache sizes.]
2. Writeback policy doesn’t matter.

![3D plot of write latency with different policies and RAM sizes.]

- 8 GB RAM
- 64 GB flash
- 30% writes

- Synchronous
- None
3. The naive architecture is fine.
4. Persistence is nice but not critical.

![Graph](image.png)

**Read Latency as a function of Working Set Size**

- No flash warmed
- 64 GB flash, not warmed
- 64 GB flash warmed

8 GB RAM
30% writes
5. Consistency does matter.

In the figure, the relationship between invalidations as a function of working set size is shown for two scenarios:

- **No flash**
- **64 GB flash**

### Key Observations:

- **2 hosts**
- **1 working set**
- **8 GB RAM**
- **30% writes**
6. Use your RAM for other stuff.

Graph: Read and Write Latency as a function of RAM Size (60 GB working set)

- **Read (p1)**
- **Read (a)**
- **Write (p1)**
- **Write (a)**

**Syncer artifact**

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

**8G**
...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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM read</td>
<td>400 ns / 4K block</td>
</tr>
<tr>
<td>RAM write</td>
<td>400 ns / 4K block</td>
</tr>
<tr>
<td>Flash read</td>
<td>88 µs / 4K block</td>
</tr>
<tr>
<td>Flash write</td>
<td>21 µs / 4K block</td>
</tr>
<tr>
<td>Network base latency</td>
<td>8.2 µs / packet</td>
</tr>
<tr>
<td>Network data latency</td>
<td>1 ns / bit</td>
</tr>
<tr>
<td>File server fast read</td>
<td>92 µs / 4K block</td>
</tr>
<tr>
<td>File server slow read</td>
<td>7952 µs / 4K block</td>
</tr>
<tr>
<td>File server write</td>
<td>92 µs / 4K block</td>
</tr>
<tr>
<td>File server fast read rate</td>
<td>90%</td>
</tr>
</tbody>
</table>
Flash device access latency

SSD access latency as a function of time

Latency (in us)

Cumulative I/Os performed (millions)

Read latency
Write 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)

Read Latency as a function of Working Set Size

Latency (in us)

Working Set Size (in GB)

- No flash; 80% prefetch rate
- No flash; 95% prefetch rate
- 64 GB flash; 80% prefetch rate
- 64 GB flash; 95% prefetch rate

Usenix ATC / June 26, 2013
Latency / RAM size (60GB trace)

Read and Write Latency as a function of RAM Size (60 GB working set)

- Read (p1)
- Read (a)
- Write (p1)
- Write (a)

Write Latency (in us)

Read Latency (in us)

RAM Size (log scale, except for 0 which really means 0)
Latency / RAM size (80GB trace)
Latency / RAM size (5GB trace)

Read and Write Latency as a function of RAM Size (5 GB working set)

- Read (p1)
- Read (a)
- Write (p1)
- Write (a)

Write Latency (in us)

Read Latency (in us)

RAM Size (log scale, except for 0 which really means 0)
Latency / write percentage

Read/Write Latency as a function of the % Write Operations

- Read (80 GB)
- Read (60 GB)
- Write (80 GB)
- Write (60 GB)

Percent Write Operations

Read Latency (in us)

Write Latency (in us)
Read latency / flash read time

Read Latency as a function of the flash read time

- Read lookaside (80 GB)
- Read naive (80 GB)
- Read unified (80 GB)
- Read lookaside (60 GB)
- Read naive (60 GB)
- Read unified (60 GB)
Read latency / WSS (Persistence)

Read Latency as a function of Working Set Size

Latency (in us)

Working Set Size (in GB)

No flash warmed
64 GB flash, not warmed
64 GB flash warmed
Invalidations / W% (Consistency)

Invalidations as a function of % Write Operations

- No flash (80 GB)
- No flash (60 GB)
- 64 GB flash (80 GB)
- 64 GB flash (60 GB)

Invalidateations (% of blocks written)

Percent Write Operations

Usenix ATC / June 26, 2013
Read latency / W% (Consistency)
Invalidations / WSS (Consistency)
Read latency / WSS (Consistency)