Exploring Caching Policies for Host-side Flash Caches

Bhavesh Singh
143059003

Guided by
Prof. Purushottam Kulkarni

Setting the Context - Centralized Storage

- Storage separate from servers
    - Accessible over the network
    - Exposed as logical disks
- Benefits
    - Storage can scale independently
    - Easy maintenance
- Challenges
    - Network round trips for each read/write.

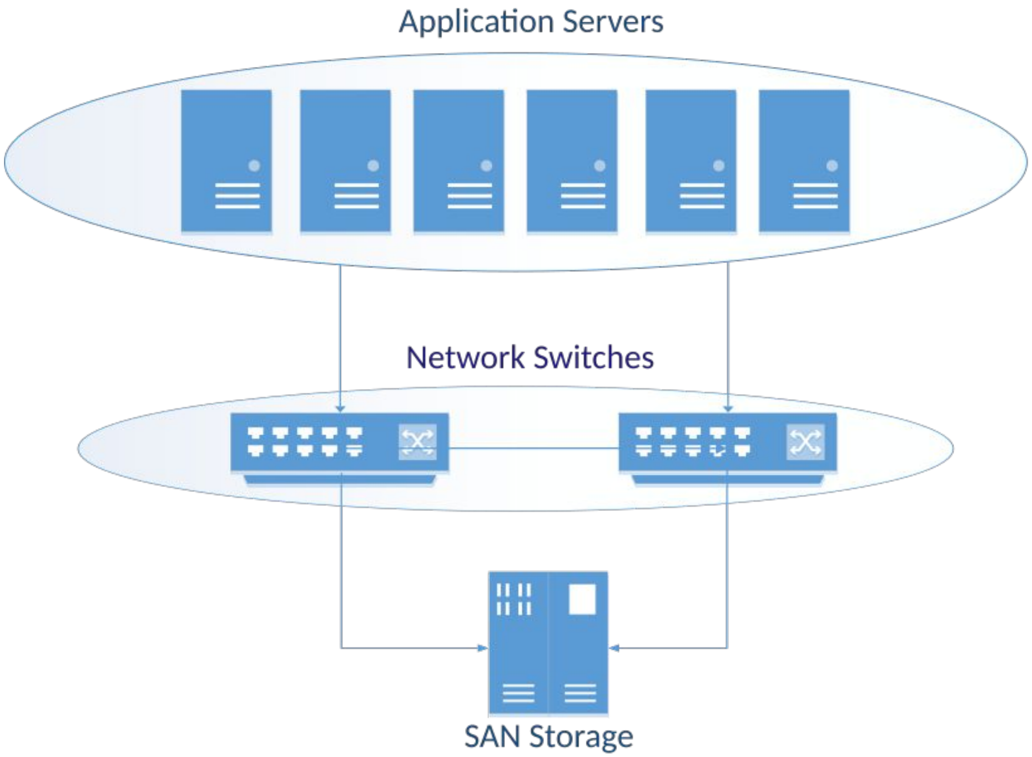


Image credit: blog.cdemi.io/

Setting the Context - Host-side Flash Caches

- Solid state disks used at the host-side as caching layer
- Benefits
  - Save network round trips to networked storage
  - Faster than network access
  - Persistent
- Challenges
  - Caching policy decisions
    - Data consistency

Caching Policies - Write Through

- Ack a write only after writing to storage and cache
- Good for read-heavy workloads
- Poor performance for write-heavy workloads
- Transactional Consistency
    - Each write acknowledged to the application is guaranteed to be on the storage

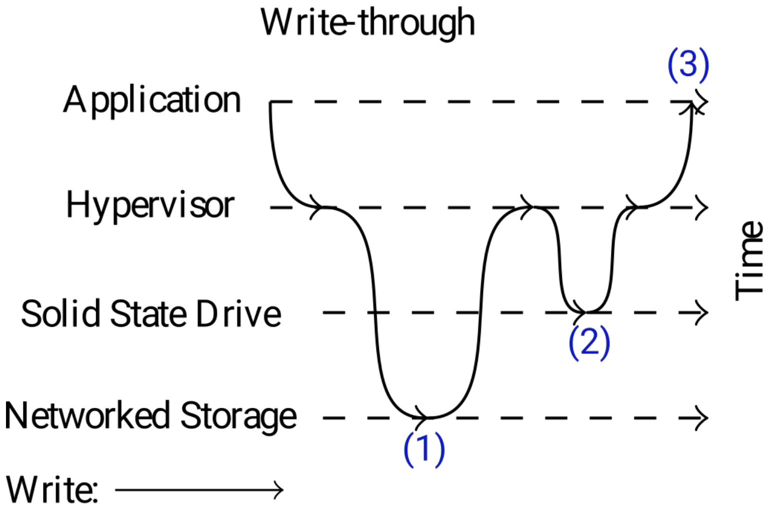


Image credit: [Koller, FAST 13]

Caching Policies - Write Back

- Cache the writes too
- Ack a write after writing to cache
  - Evict on memory pressure
- Wonderful for write-heavy workloads
  - Write coalescing
- No consistency guarantees

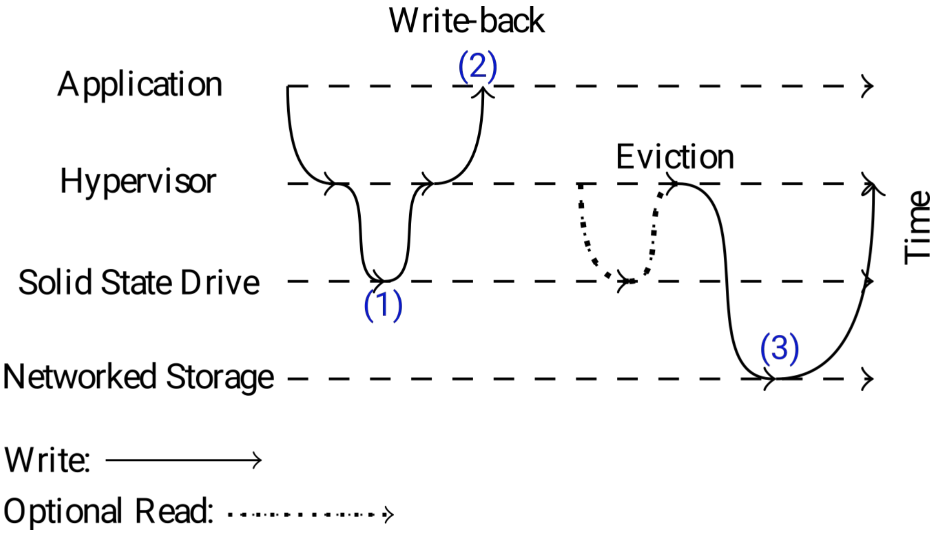
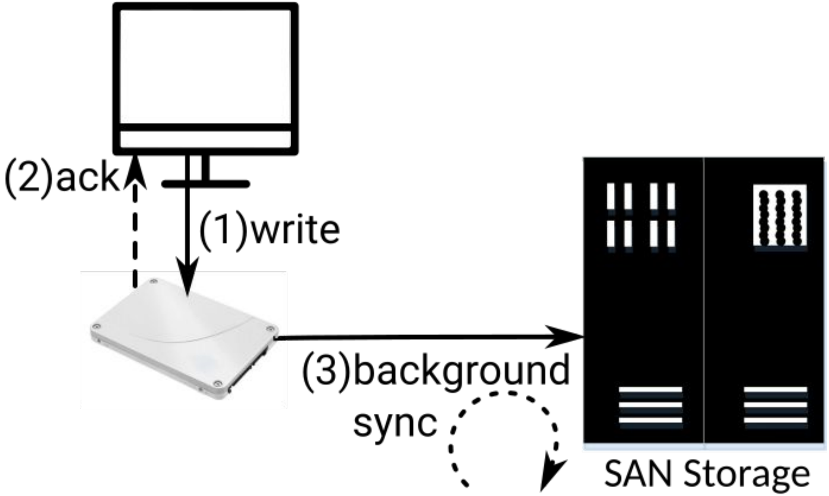


Image credit: [Koller, FAST 13]

Caching Policies - Write Back Variants

- Room for improvement
  - Trade off data staleness for improved write performance
  - Get lower consistency guarantees than write through
  - Batch updates to networked storage in the background



Ordered Write Back [Koller, FAST 13]

- Imposes a partial order on writes
  - Called completion-issue order
  - Ensure writes are evicted in order
- Point-in-time consistency
  - After recovery storage reflects application's view of storage at some point-in-time
- Space overhead
  - Each write to a block mapped to different cache block
- Data lost in case of flash failure

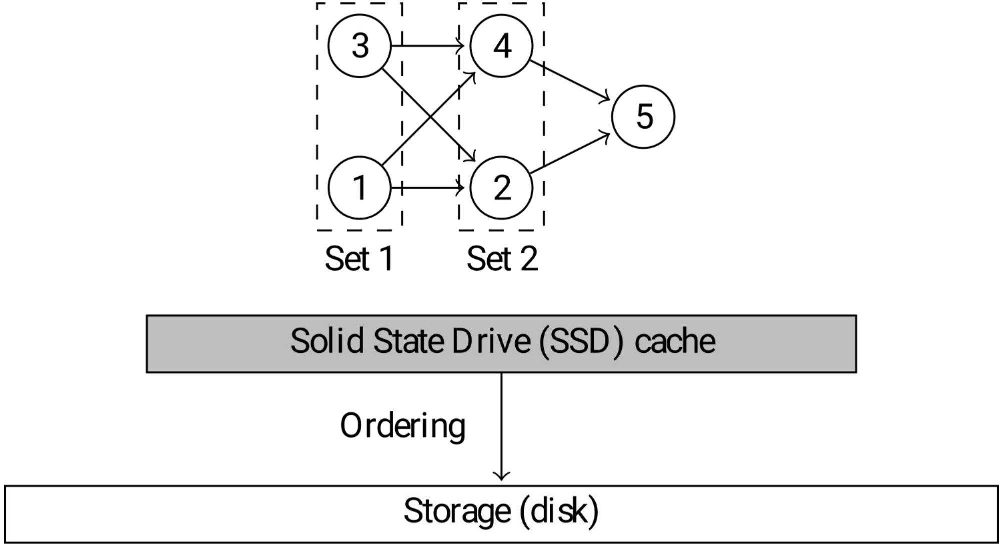


Image credit: [Koller, FAST 13]

Journaled Write Back [Koller, FAST 13]

- Groups updates into journals
  - Journals committed atomically to networked storage
  - Writes within a journal can be coalesced
- Point-in-time consistency
- Data lost in case of flash failure

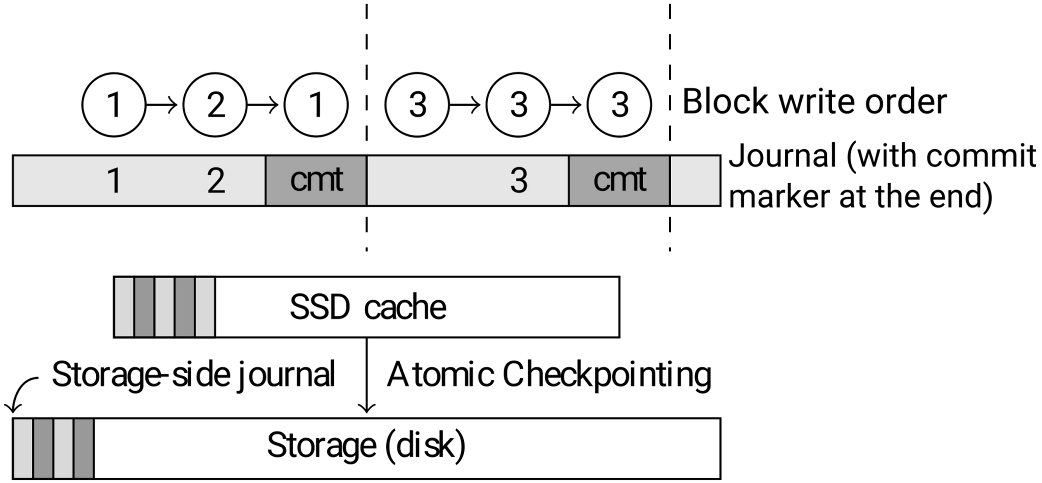
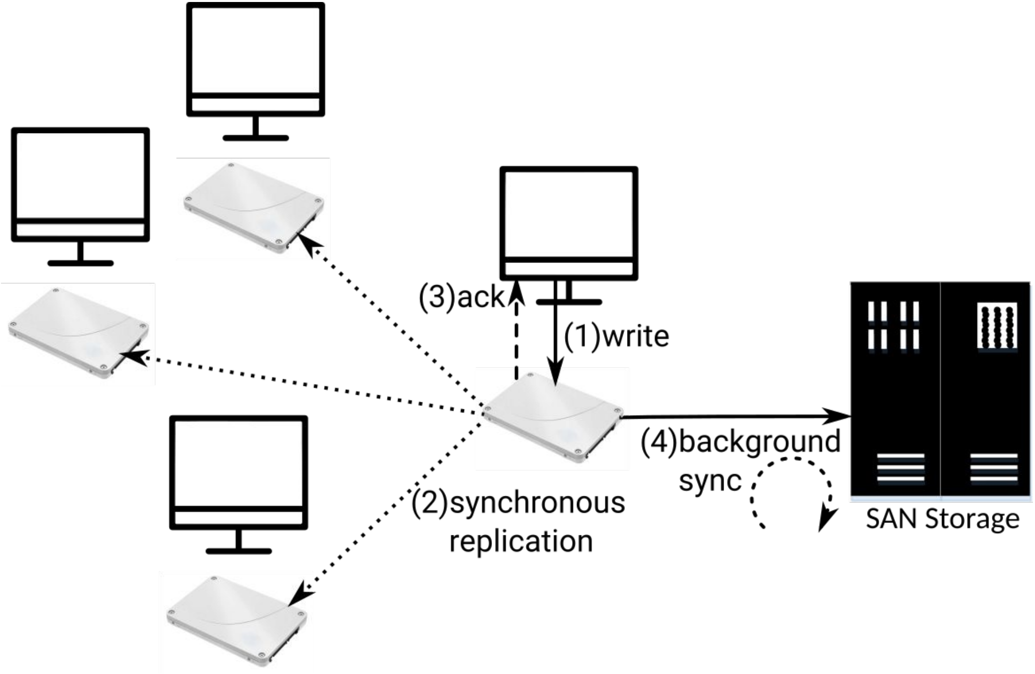


Image credit: [Koller, FAST 13]

Write Back (with Peer Replication) [Bhagwat, FAST 15]

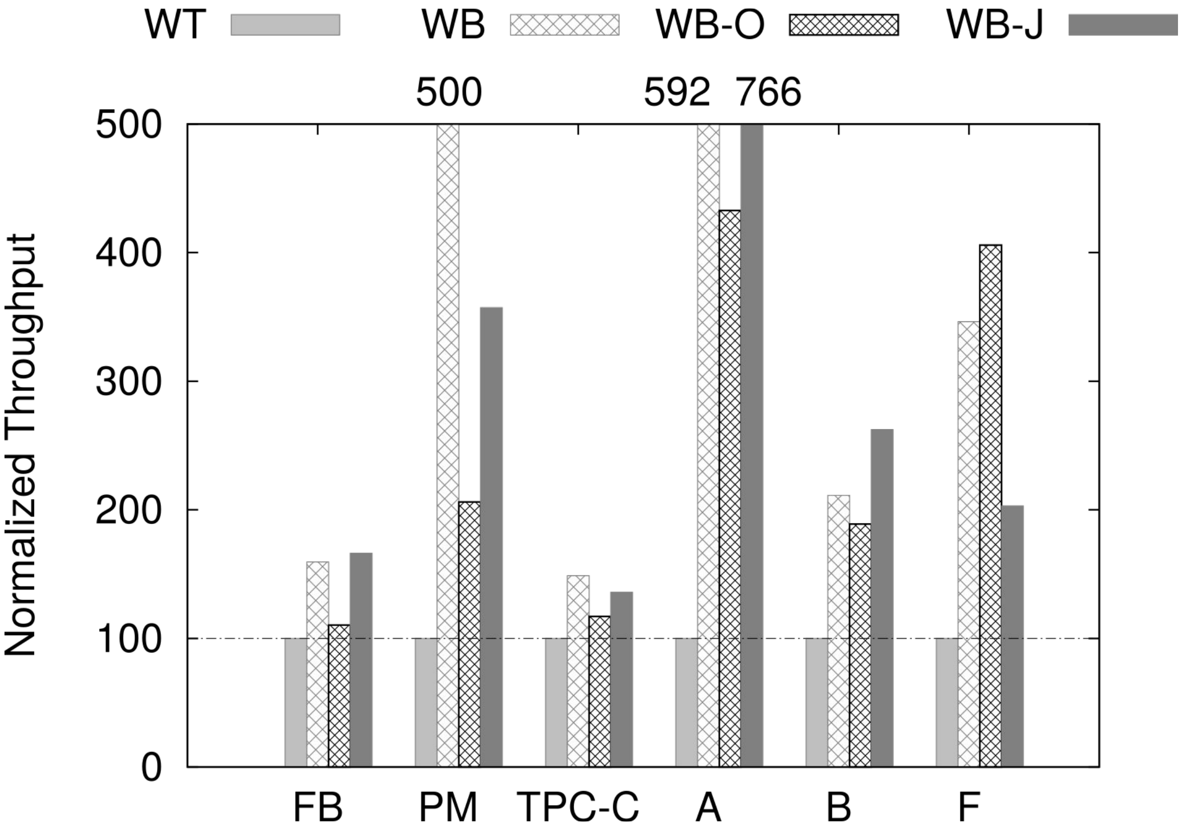
- Adds fault-tolerance
    - Each write synchronously replicated to flash caches of **k** peers
- Point-in-time consistent
- Can tolerate k flash failures and k+1 host failures
- Synchronous replication overhead



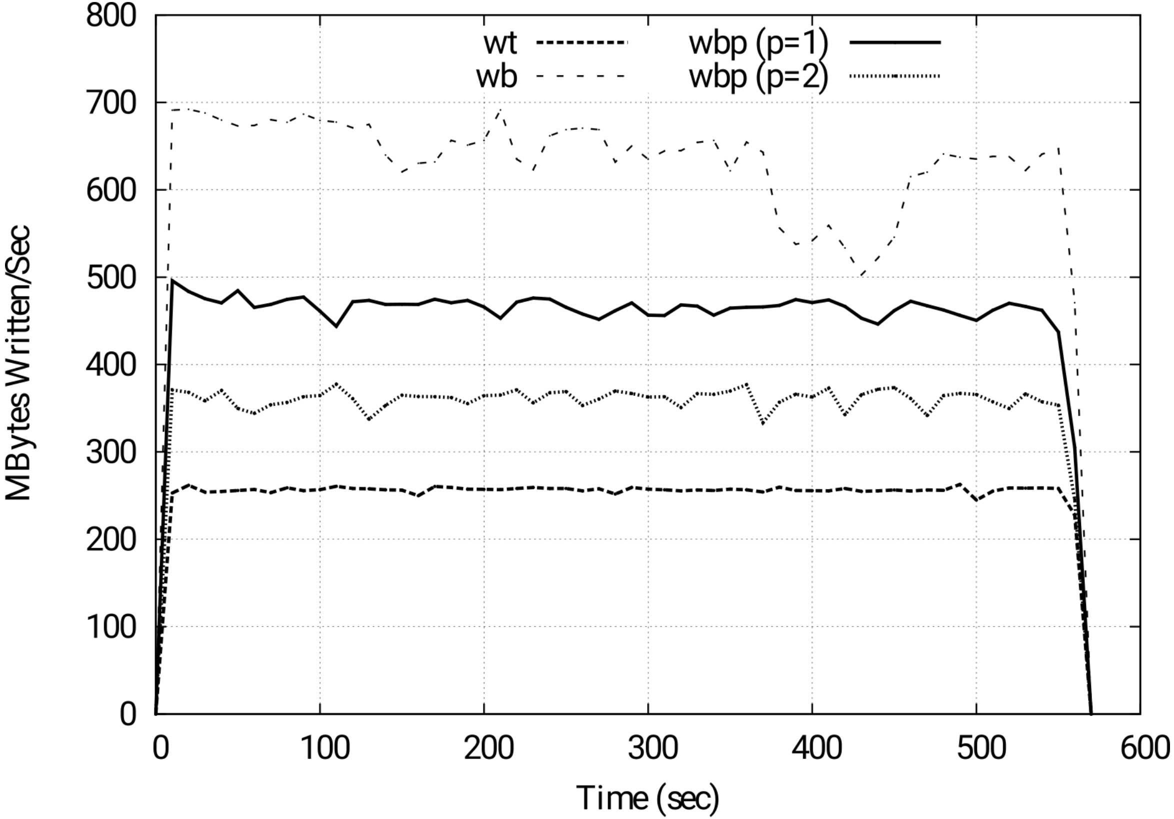
Write Back Flush and Write Back Persist [Qin, ATC 14]

- Flash cache provides consistency guarantees equivalent to the underlying storage
  - Applications only expect data to be persistent at write barriers viz. fsync.
- Application-level consistent
  - After recovery, storage reflects a state which satisfies some application-level constraints

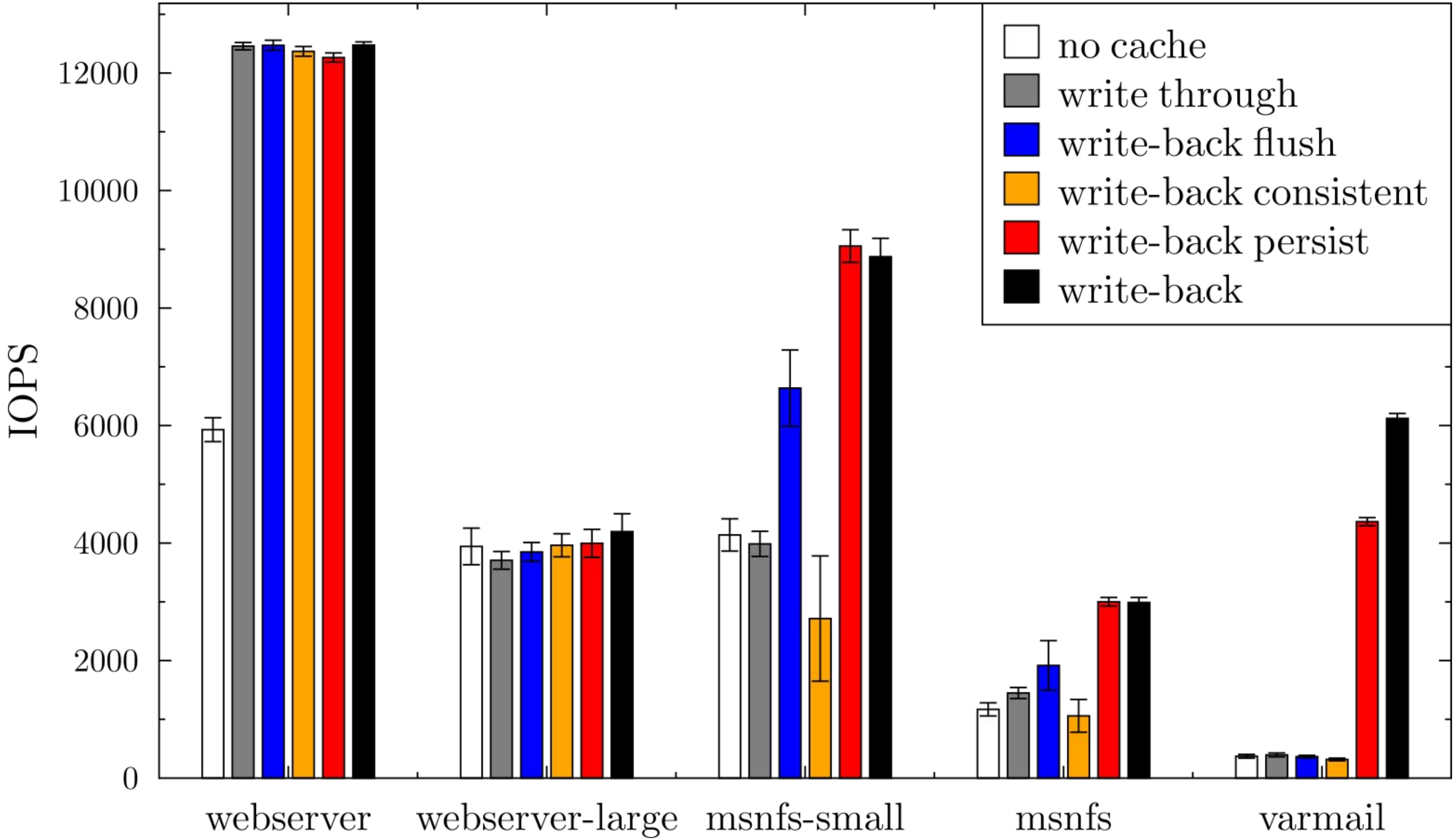
Performance Comparison [Koller, FAST 13]



Performance Comparison [Bhagwat, FAST 15]



Performance Comparison [Qin, ATC 14]



Overview of Caching Policies

Caching Policy	**Consistency Provided**	**Fault Tolerance**
Write through	Transactional	
Write back	Point-in-time	
Ordered write back	Point-in-time	
Journaled write back	Point-in-time	
Write back flush	Application-level	
Write back persist	Application-level	Flash failures only
Write back with peering	Point-in-time	Yes

Can We Do Better?

- GOAL: Be “almost synchronous” while performing fully asynchronously.
- Motivation
  - Using packet-level FEC [Balakrishnan, NSDI 08]
  - Monitor the network and create a global view to help choose which paths to send data on [Al-Fares, NSDI 10].

Packet-level FEC [Balakrishnan, NSDI 08]

Forward Error Correction used to recover from packet losses

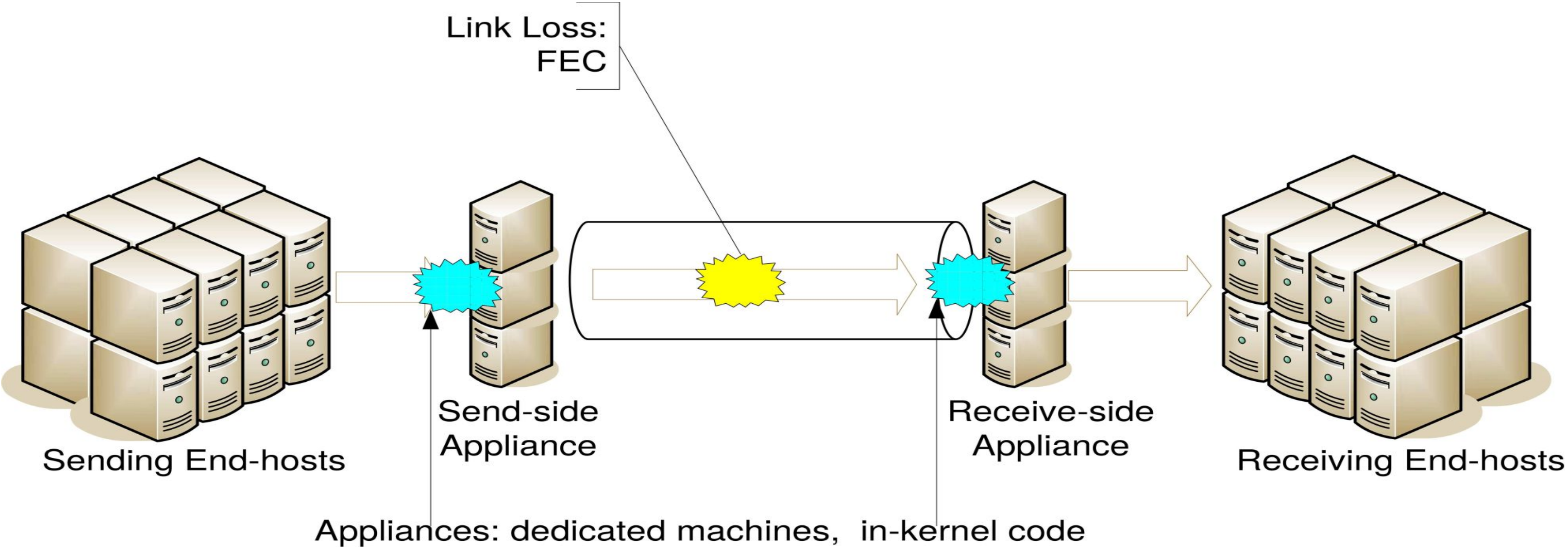


Image credit: [Balakrishnan, NSDI 08]

Global View of Data Center Network [Al-Fares, NSDI '08]

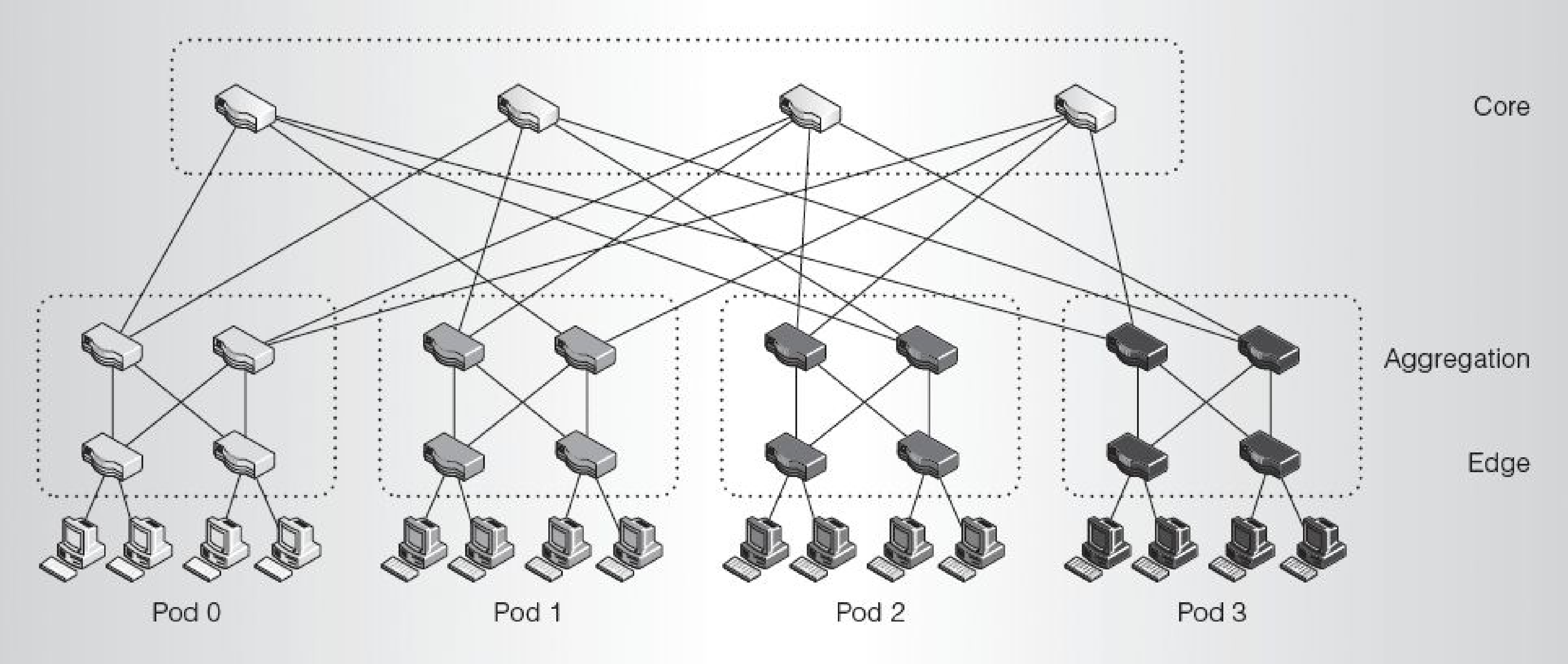
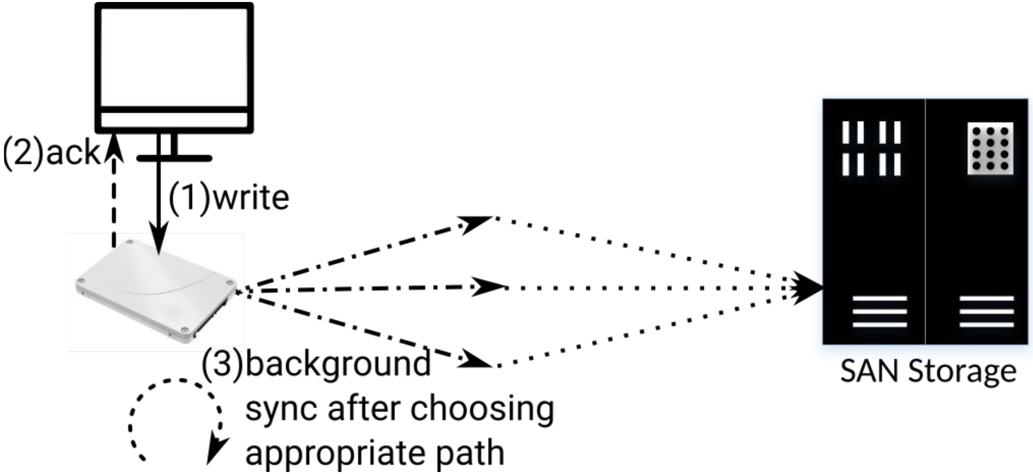


Image credit: Google Image Search

Our Solution - Probabilistic Asynchronous Writes

- Asynchronous
  - Writes ack'ed as soon as persisted on flash and put on the wire.
- Probabilistic
  - Determine the loss probabilities of the packet on various routes to the networked storage and choose best path.
  - Use packet-level FEC to overcome packet losses.



References

1. *Koller, R., Marmol, L., Rangaswami, R., Sundararaman, S., Talagala, N., and Zhao, M.* **Write Policies for Host-side Flash Caches**. *USENIX FAST 13.*
2. *Bhagwat, D., Patil, M., Ostrowski, M., Vilayannur, M., Jung, W., and Kumar, C.* **A Practical Implementation of Clustered Fault Tolerant Write Acceleration in a Virtualized Environment**. *USENIX FAST 15.*
3. *Qin, D., Brown, A. D., and Goel, A.* **Reliable Writeback for Client-side Flash Caches**. *USENIX ATC 14*.
4. *Balakrishnan, M., Marian, T., Birman, K., Weatherspoon, H., and Vollset, E.* **Maelstrom: Transparent Error Correction for Lambda Networks**. *USENIX NSDI 08.*
5. *Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., and Vahdat, A.* **Hedera: Dynamic Flow Scheduling for Data Center Networks**. *USENIX NSDI 10.*

Thank You

And May the Fourth Be With You!
(Today is Star Wars Day)