FarReach: Write-back Caching in Programmable Switches

Siyuan Sheng¹, Huancheng Puyang¹, Qun Huang², Lu Tang³, and Patrick P. C. Lee¹

¹The Chinese University of Hong Kong  ²Peking University  ³Xiamen University

USENIX ATC 2023
 Writes in Key-value Stores

ongan in production key-value storage workloads

• 20% of Twitter’s Twemcache clusters are write-intensive
• Facebook’s RocksDB for AI services has 92.5% of read-modify-writes

Challenges for high write performance

• High round-trip latencies in transmission, queuing, and processing
• Skewness introduces imbalanced server loads
Programmable switches can help improve write performance
• Switch OS controls multi-pipeline data plane
• Each pipeline has multiple stages with stateful memory

Write-back policy: caches popular write records in switch without immediately updating servers
Challenges

- Performance challenge
  - Scarce switch resources require offloading cache management to controller → high controller-to-switch latency

- Availability challenge
  - Synchronization between switch and servers is required to keep latest records available

- Reliability challenge
  - Latest records may be lost in switch failures under write-back caching
Our Contributions

- **FarReach**, a fast, available, and reliable in-switch write-back cache
  - Non-blocking cache admission for fast access
  - Available cache eviction
  - Crash-consistent snapshot generation and zero-loss recovery

- Prototype implementation
  - P4-based in-switch cache and RocksDB-based servers

- Tofino switch evaluation
  - Up to 6.6× throughput gain under 128 simulated servers

- Open-source FarReach prototype
FarReach architecture

- In-switch cache absorbs writes with cache hits
- Controller performs cache management through switch OS
- Carefully co-design control and data planes
Problem of Cache Admission

- Suppose that a request triggers cache admission
  - Subsequent writes arrive at switch before admission
  - Blocking subsequent writes undermines I/O performance
  - Absorbing subsequent writes in switch undermines availability
Non-blocking Cache Admission

- Process subsequent writes in server without blocking
  - Mark admitted record as “outdated” as server is latest
  - Conservatively forward subsequent reads to server for availability
  - Mark admitted record as “latest” as early as possible
Problem of Cache Eviction

- Under write-back policy
  - Evicted record is latest yet not updated to server
  - Controller loads evicted record to server for persistent storage

- Subsequent writes arrive at switch during cache eviction
  - Processing without synchronization undermines availability
  - Synchronization by controller incurs large overhead
Available Cache Eviction

- Associate additional in-switch metadata to evicted record
  - Mark evicted record as “to-be-evicted”
  - Load evicted record to server before removing it from switch
  - Mark “to-be-evicted” record as “outdated” and forward writes to server
  - Process reads by switch if “latest” or server if “outdated”
Problem of Reliability

- Under write-back policy
  - Cached records are latest yet not updated to servers
  - Latest in-switch records are lost after switch failures

- Controller loads cached records for snapshots
  - Subsequent writes arrive at switch during snapshot generation
  - Updating cache records incurs inconsistent snapshots
Crash-consistent Snapshot Generation

- Send original cached record for each first write
  - Controller replaces overwritten records for consistency

- Two-phase algorithm
  - Controller triggers snapshot generation
  - Controller loads cached records and switch sends original ones
Zero-loss Recovery

- Limitation of snapshot generation
  - Snapshot generation avoids data loss before the latest snapshot
  - Cached records after the latest snapshot are not protected

- Client-side record preservation
  - Clients preserve copies of cached records after the latest snapshot
  - Controller notifies clients to release the snapshotted records

- Replay-based recovery
  - Replay writes of the latest records to update servers
  - Replay admission decisions to recover in-switch cache
Evaluation

➢ Methodology
  • Simulate tens of servers by server rotation for server-side storage
  • Compile P4 in a Tofino switch for in-switch cache
  • Baselines: NoCache and NetCache [Jin et al., SOSP’17]

➢ Experiments
  • YCSB core workloads to evaluate throughput, latency, and scalability
  • Synthetic workloads to evaluate impact of different parameters
  • Performance of snapshot generation and crash recovery time
  • Hardware resource usage
Throughput Analysis

- Simulate 16 servers by server rotation
- Larger throughput especially for workload A
  - In-switch write-back cache reduces server-side load

![Throughput Analysis Diagram]

- Thpt (MOPS)
- Load: A, B, C, D, F
- NoCache, NetCache, FarReach
Scalability

- Use workload A (skewed and write-intensive)
  - Simulate 16 to 128 servers by server rotation
- Throughput gain is up to 6.6× under 128 simulated servers
  - In-switch write-back cache balances server-side load
Performance of Snapshot Generation

- Dynamic workload patterns
  - Bandwidth includes snapshot generation and cache management

- Similar throughput and limited control-plane bandwidth
Conclusion

- FarReach, a fast, available, and reliable in-switch write-back cache
  - Non-blocking cache admission
  - Available cache eviction
  - Crash-consistent snapshot generation with zero-loss recovery

- Tofino switch evaluation on YCSB and synthetic workloads

- Source code:
  - [http://adslab.cse.cuhk.edu.hk/software/farreach](http://adslab.cse.cuhk.edu.hk/software/farreach)
Thank You!

Q & A