# FarReach: Write-back Caching in Programmable Switches

Siyuan Sheng[1], Huancheng Puyang[1], Qun Huang[2], Lu Tang[3], and Patrick P. C. Lee[1]
[1]The Chinese University of Hong Kong, [2]Peking University, [3]Xiamen University
Source code: http://adslab.cse.cuhk.edu.hk/software/farreach/

## Motivation

- **Goal: Design a fast, available, and reliable in-switch write-back caching framework to improve key-value store performance under skewed write-intensive workloads**

- **Skewed write-intensive workloads** become dominant in recent production key-value stores
- Write requests suffer from **long round-trip times** (RTTs) and **server-side imbalanced load**
- **Programmable switches** can cache hot records to reduce RTTs and balance server-side load

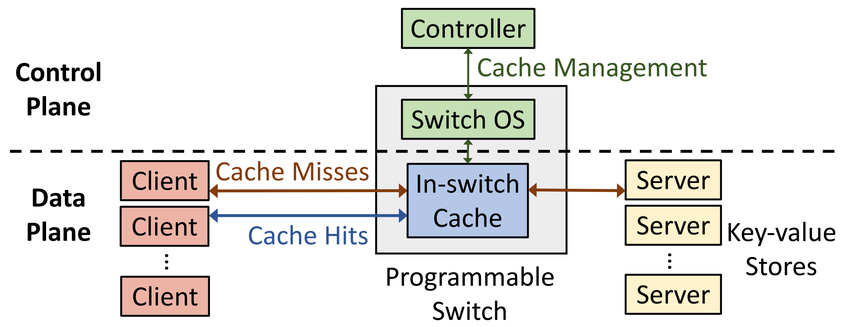
## Challenges

- In-switch write-back caching is subject to three challenges as follows:

- 1. **Performance challenge**: Controller is required for cache management due to switch limitations, yet with I/O performance degradation due to slow control-plane processing
- 2. **Availability challenge**: Keeping latest records available to clients incurs synchronization overhead
- 3. **Reliability challenge**: Cached records not updated to servers can be lost after switch failures

## Overview

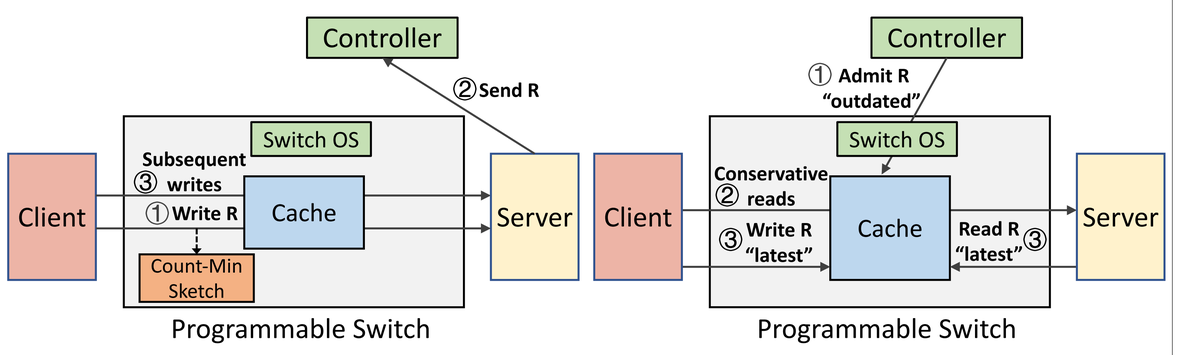
- **Architecture**: Data plane accesses or updates cached records for traversed requests with cache hits, while control plane manages in-switch cache (including cache admission and eviction)



FarReach's Architecture

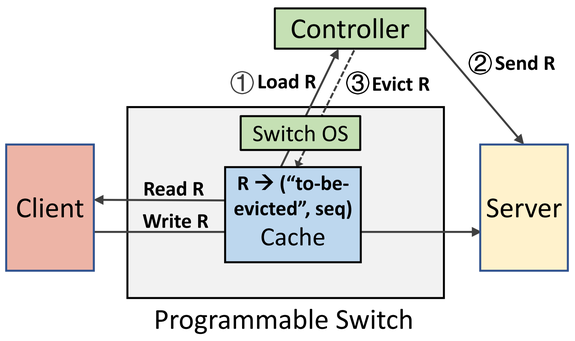
## Design

- **Non-blocking cache admission** (for a hot record R):
  - Before admitting R, switch forwards subsequent writes for R to the server for **non-blocking processing**
  - After admitting R, switch conservatively forwards reads to server until writes or read responses mark R as "latest"



(a) Before cache admission (b) After cache admission

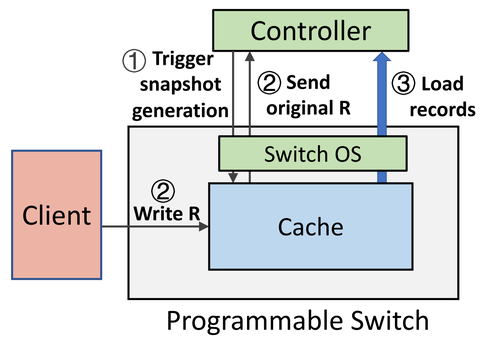
**Non-blocking Cache Admission**



**Available Cache Eviction**

- **Available cache eviction** (for a to-be-evicted record R):
  - For writes for R, switch marks R as "outdated" and forwards writes to server
  - For reads for R, switch returns R to clients if R is "latest", or forwards reads to the server if R is "outdated", to **keep the latest version of R always available**

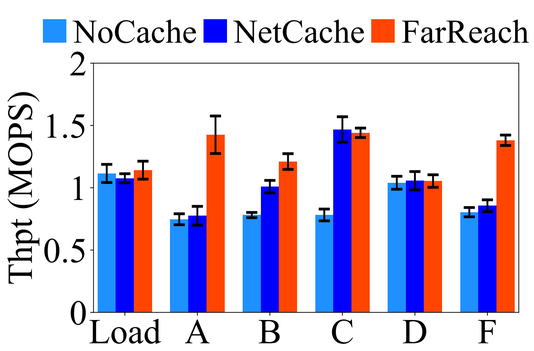
- **Crash-consistent snapshot generation** (with two phases):
  - In the first phase, controller notifies the switch to trigger snapshot generation
  - In the second phase, switch sends each original cached record (say, R) before the first write to controller; after loading all cached records, controller reverts the overwritten records with the original ones **for crash consistency**



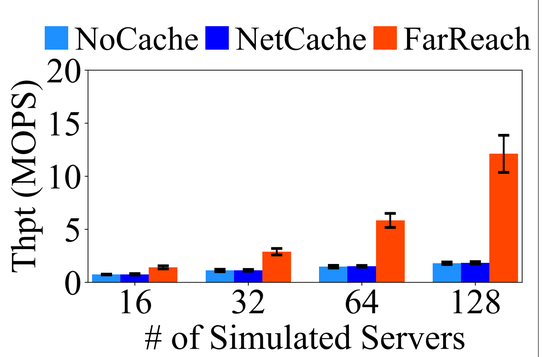
**Crash-consistent Snapshot Generation**

## Evaluation

- Prototype **FarReach** on a two-pipeline Tofino switch
- **Throughput analysis**: Increase throughput by up to 91% and 84% (workload A w/ 50% reads and 50% writes)
- **Scalability analysis**: Achieve up to 6.6× throughput gain under 128 simulated servers



**Throughput Analysis**



**Scalability Analysis**