Be Fast, Cheap and in Control with SwitchKV

Xiaozhou Li
Raghav Sethi
Michael Kaminsky
David G. Andersen
Michael J. Freedman

PRINCETON UNIVERSITY
intel
Carnegie Mellon University

Goal: fast and cost-effective key-value store

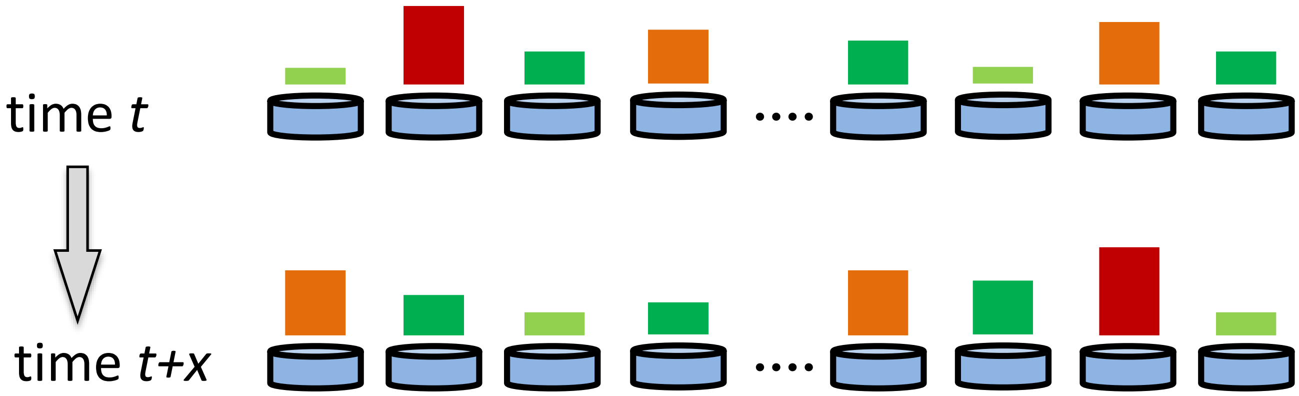
- Target: cluster-level storage for modern cloud services

  …

  - Massive number of small key-value objects
  - Highly skewed and dynamic workloads
  - Aggressive latency and throughput performance goals

- This talk: **scale-out flash-based storage** for this setting

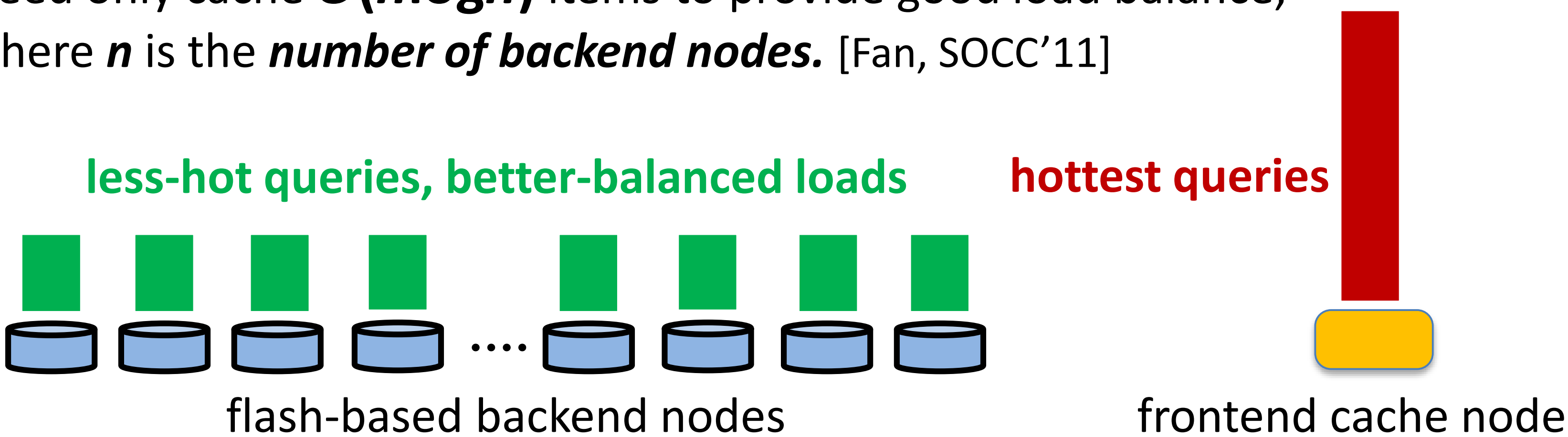
Key challenge: dynamic load balancing



- How to handle the **highly skewed and dynamic workloads**?
- Today's solution: data migration / replication
  - system overhead
  - consistency challenge

Fast, ***small*** cache can ensure load balancing

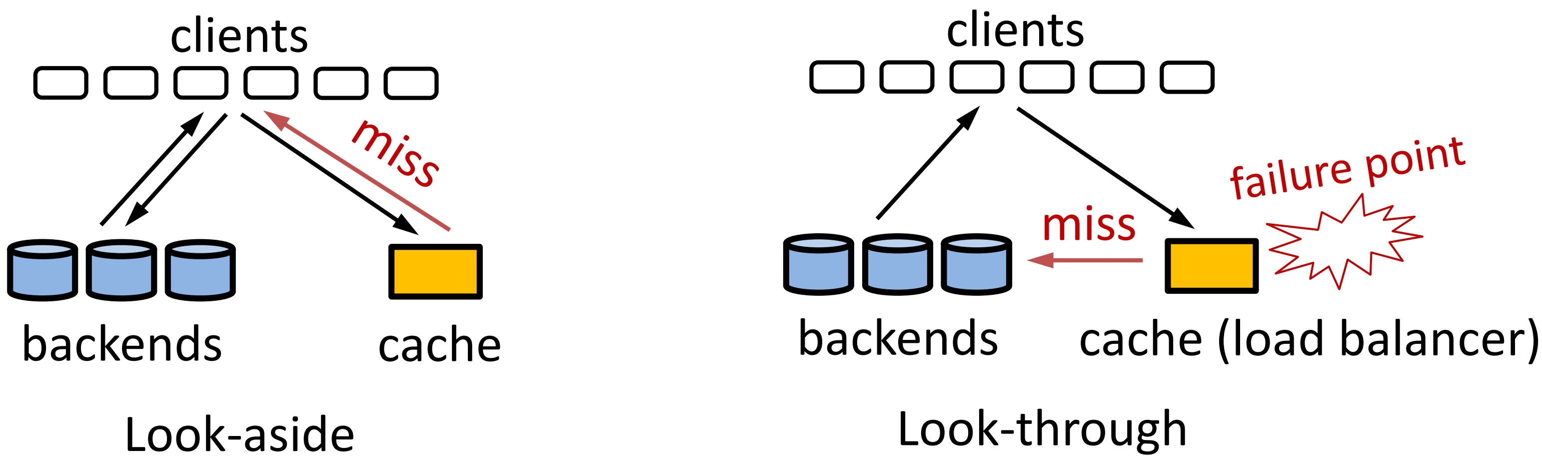
Need only cache ***O*(*nlogn*)** items to provide good load balance, where ***n*** is the ***number of backend nodes.*** [Fan, SOCC'11]



E.g., 100 backends with hundreds of billions of items + cache with 10,000 entries

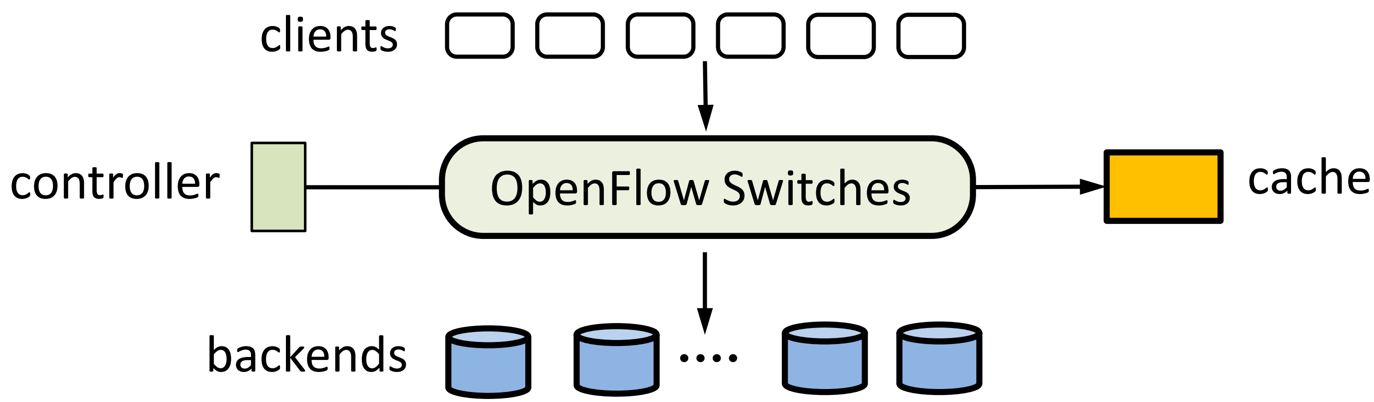
- How to efficiently serve queries with cache and backend nodes?
- How to efficiently update the cache under dynamic workloads?

High overheads with traditional caching architectures



- Cache must process all queries and handle misses
- In our case, cache is small and hit ratio could be low
  - Throughput is bounded by the cache I/O
  - High latency for queries for uncached keys

SwitchKV: content-aware routing

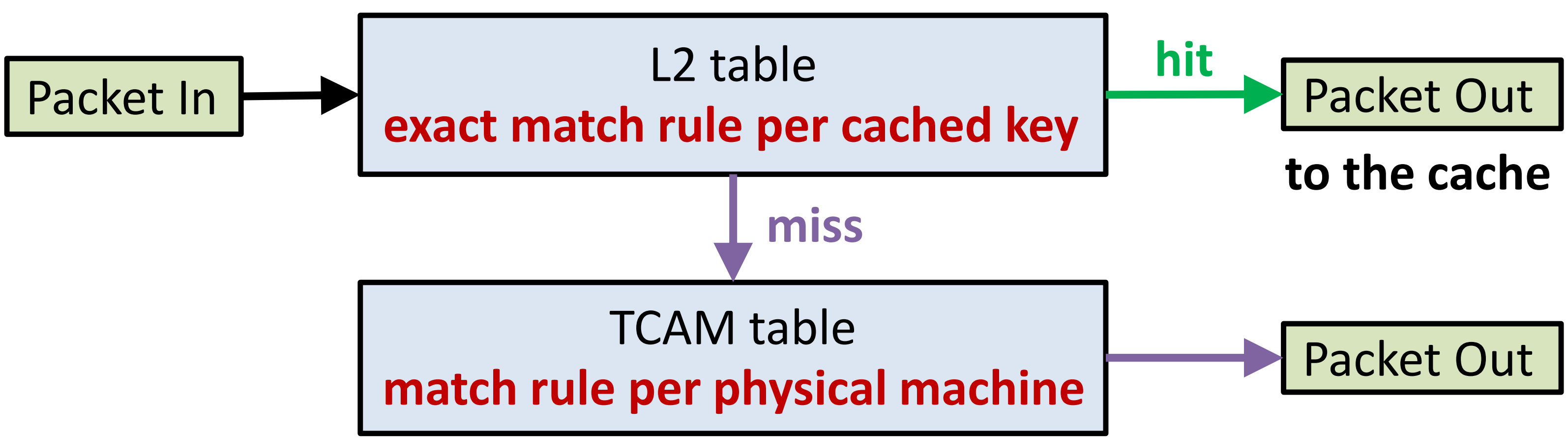


Switches route requests directly to the appropriate nodes

- Latency can be minimized for all queries
- Throughput can scale out with # of backends
- Availability would not be affected by cache node failures

Exploit SDN and switch hardware

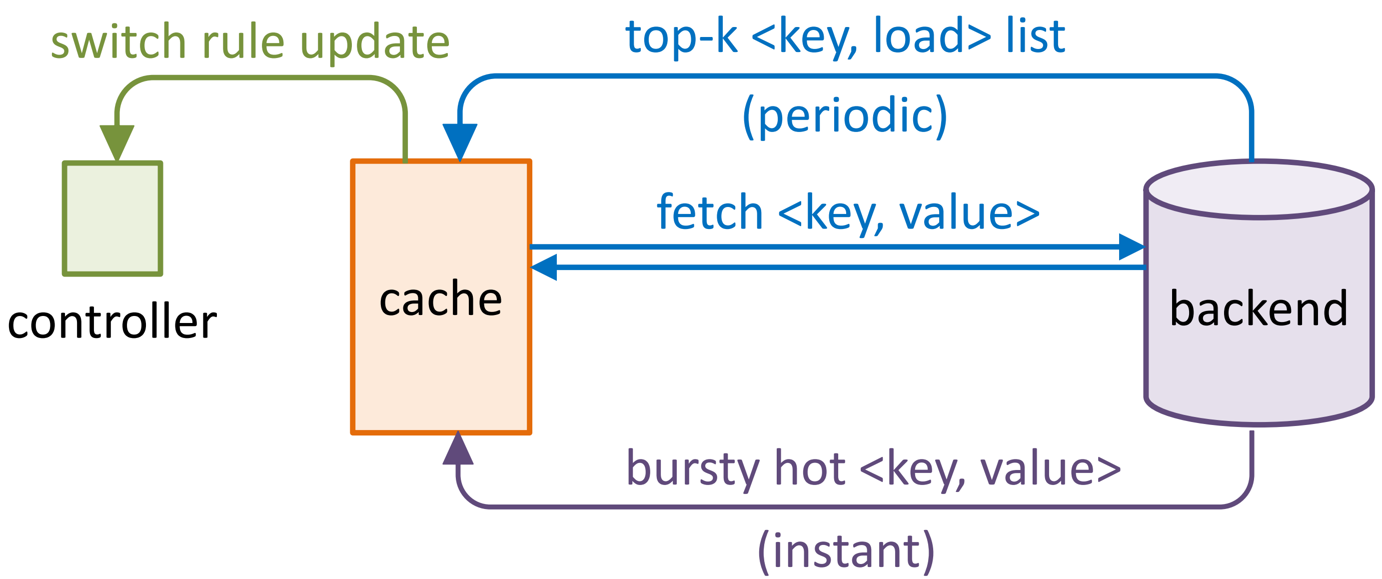
- Clients encode key information in packet headers
  - Encode **key hash in MAC** for read queries
  - Encode destination **backend ID in IP** for all queries
- Switches maintain forwarding rules and route query packets



Keep cache and switch rules updated

- New challenges for cache updates
  - Only cache the hottest O(*n*log*n*) items
  - Limited switch rule update rate

- Goal: **react quickly** to workload changes with **minimal updates**



Evaluation

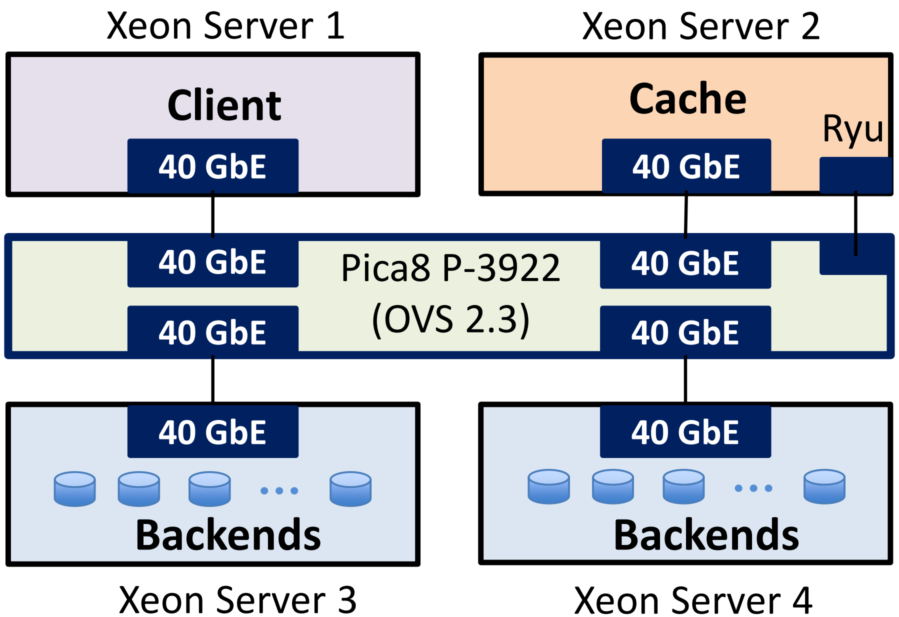
- How well does a fast small cache improve the system load balance and throughput?
- Does SwitchKV improve system performance compared to traditional architectures?
- Can SwitchKV react quickly to workload changes?

Evaluation Platform

Reference backend

- 1 Gb link
- Intel Atom C2750 processor
- Intel DC P3600 PCIe-based SSD
- RocksDB with 120 million 1KB objects
- 99.4K queries per second

Evaluation Platform

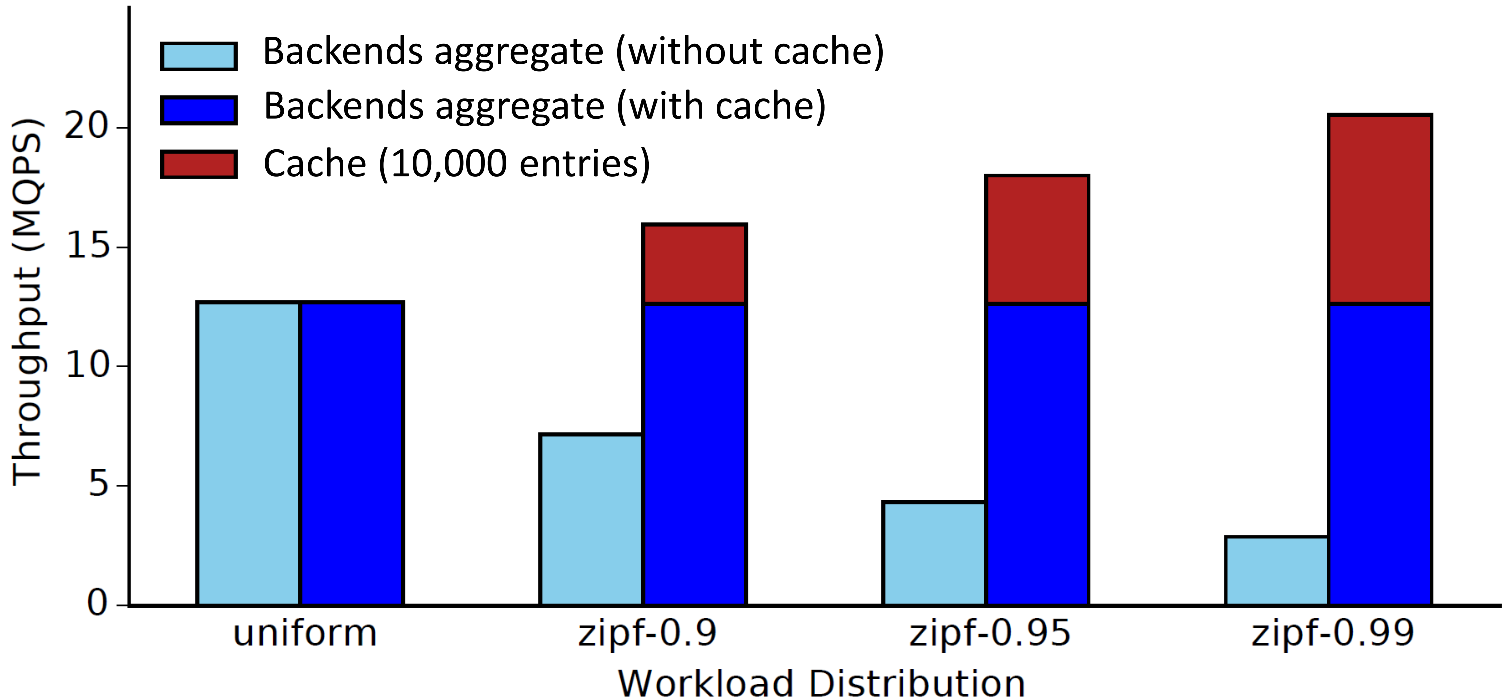


# of backends	128
backend tput	100 KQPS
keyspace size	10 billion
key size	16 bytes
value size	128 bytes
query skewness	**Zipf 0.99**
cache size	**10,000 entries**

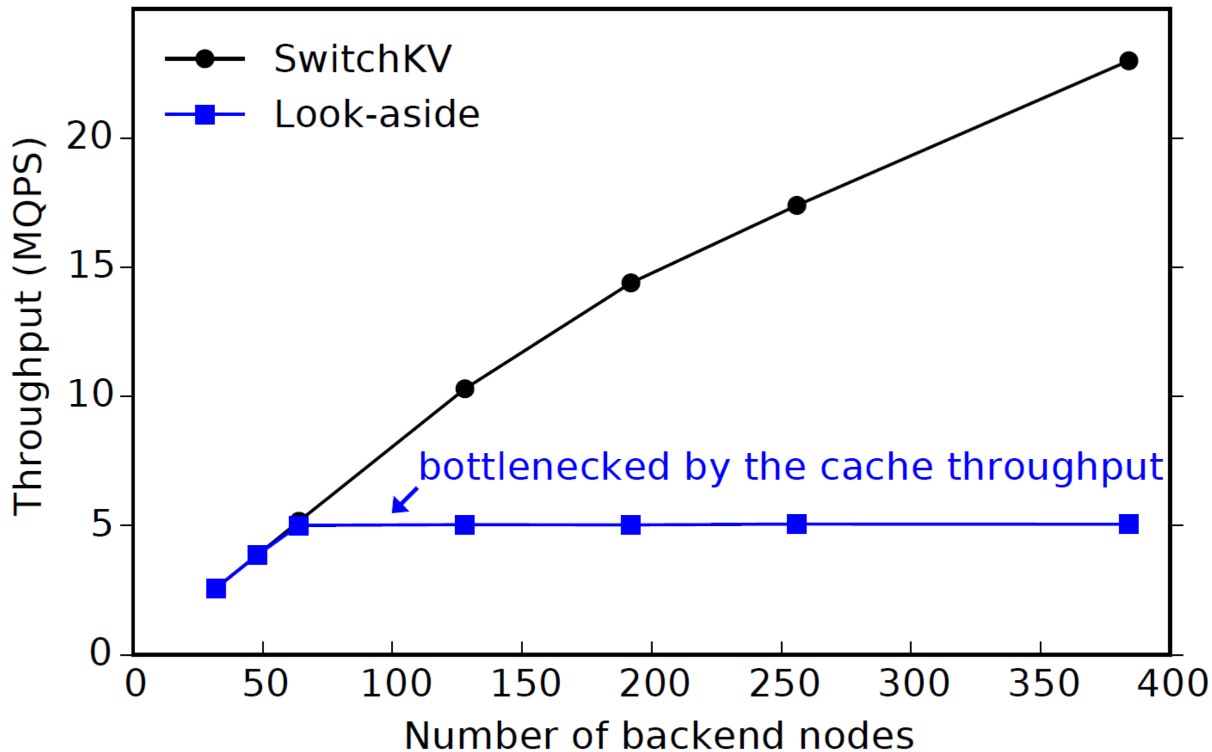
Default settings in this talk

- Use Intel DPDK to efficiently transfer packets and modify headers
- Client adjusts its sending rate, keep loss rate between 0.5% and 1%

Throughput with and without caching

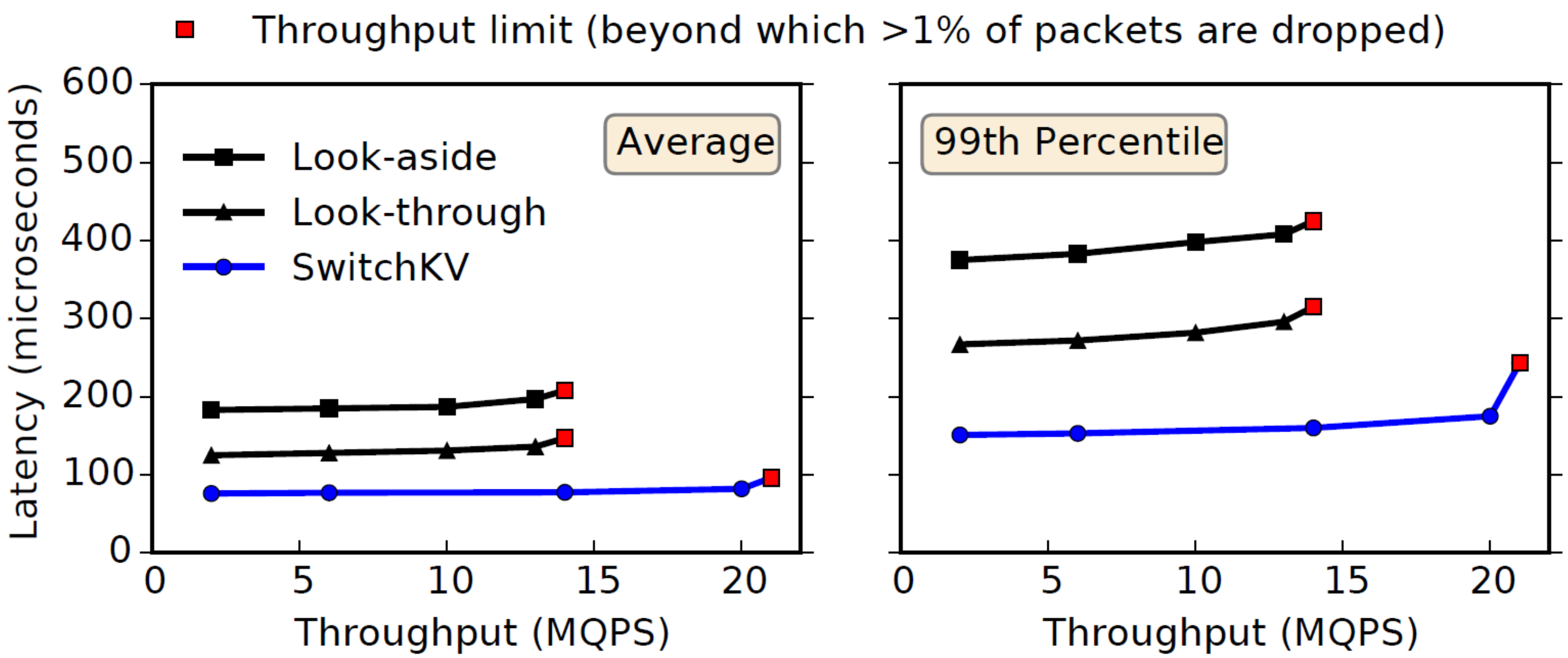


Throughput vs. Number of backends

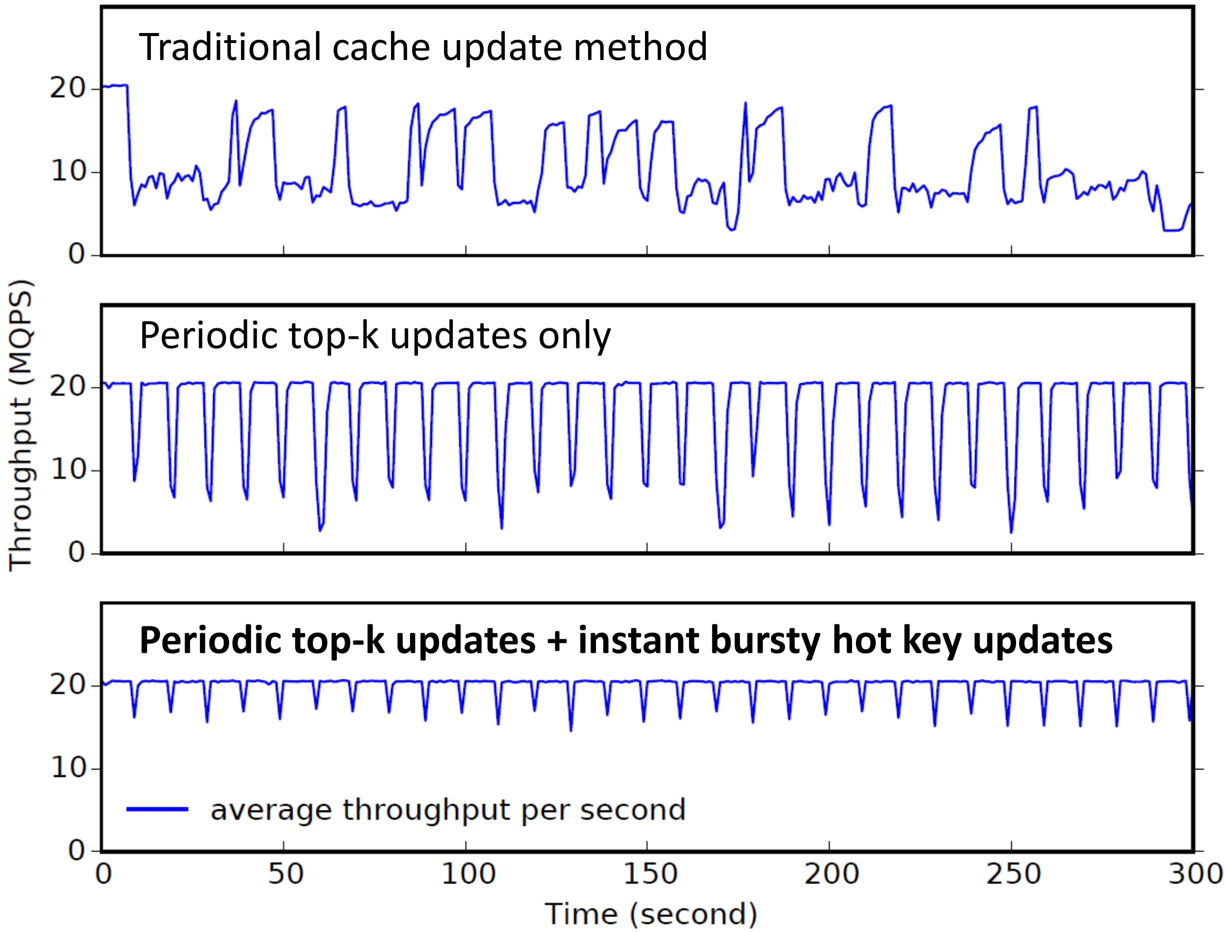


backend rate limit: 50KQPS, cache rate limit: 5MQPS

End-to-end latency vs. Throughput



Throughput with workload changes



Make 200 cold keys become the hottest keys every 10 seconds

Conclusion

SwitchKV: high-performance and cost-efficient KV store

- Fast, small cache guarantees backend load balancing
- Efficient content-aware OpenFlow switching
  - Low (tail) latency
  - Scalable throughput
  - High availability
- Keep high performance under highly dynamic workloads