A Heterogeneous Key-Value System with Fast Load Balancing

Xiaozhou Li, Raghav Sethi, Michael Freedman (Princeton), David Andersen (CMU), Michael Kaminsky (Intel Labs)

Goal: Cost-Effective Key-Value Store

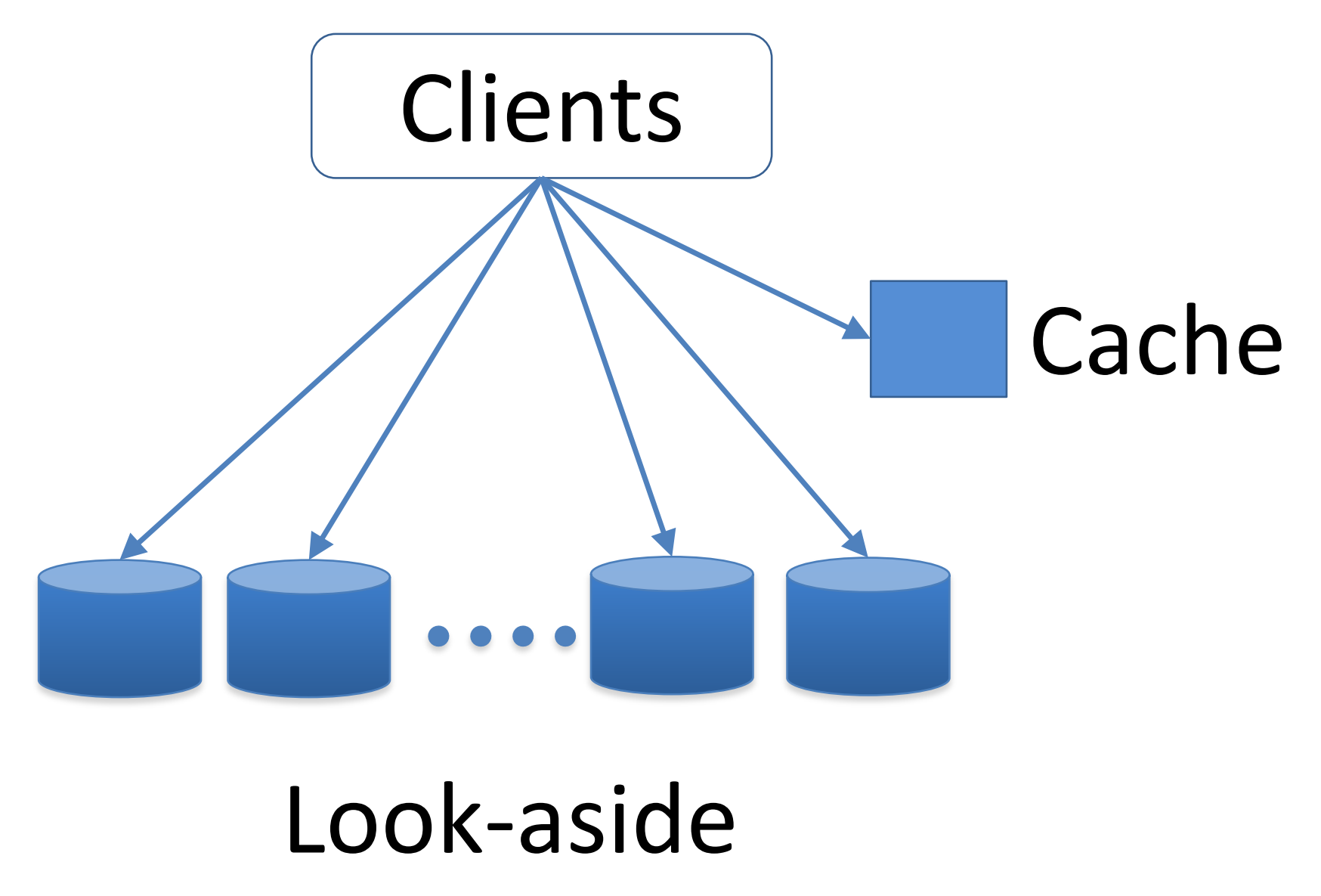
- Build large scale SSD-based key-value storage cluster using resources correctly-provisioned to meet a service-level objective (SLO)
- **Key challenge**: Handle the highly-skewed and rapidly changing real-world workloads with efficient dynamic load balancing

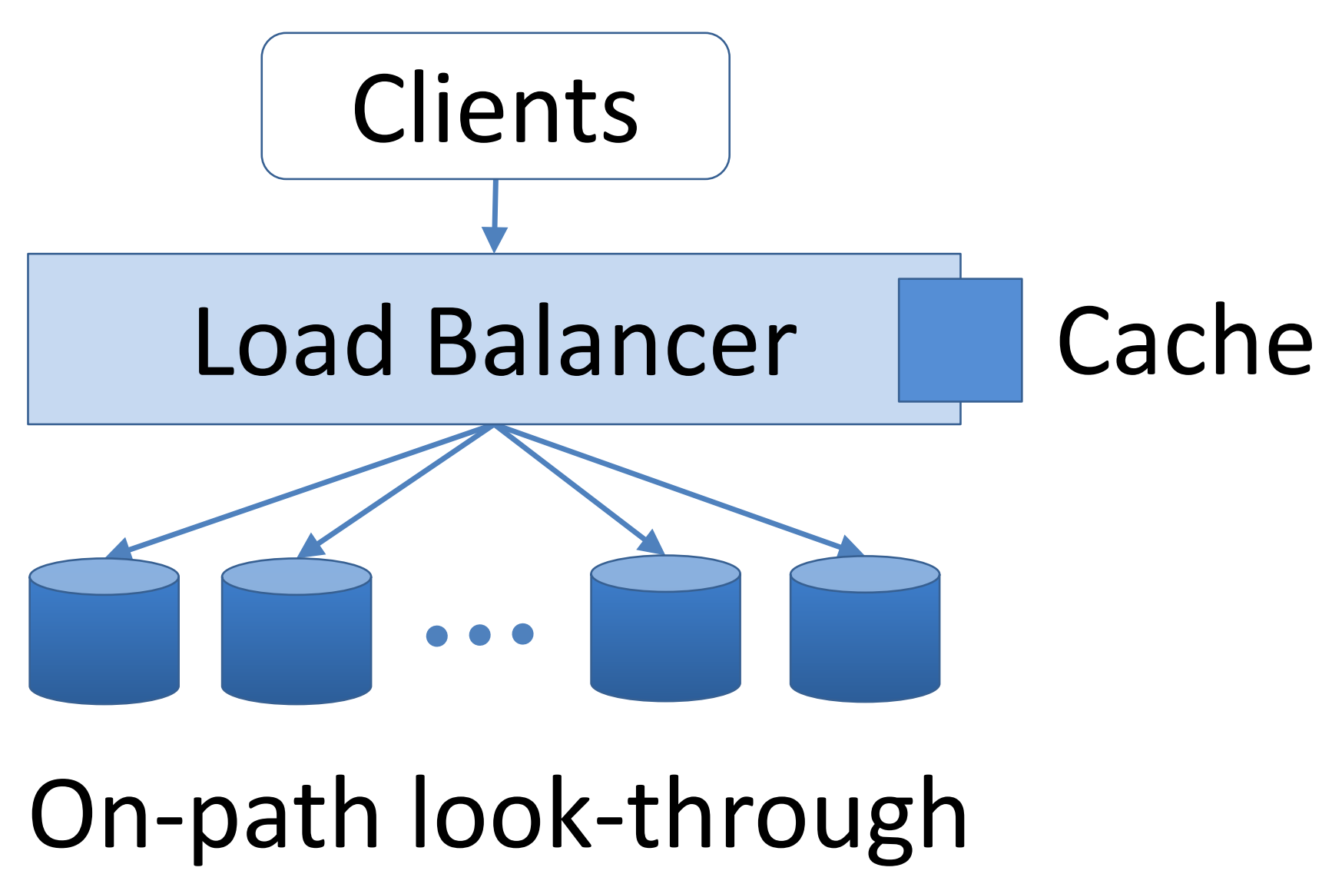
Cache-based Dynamic Load Balancing

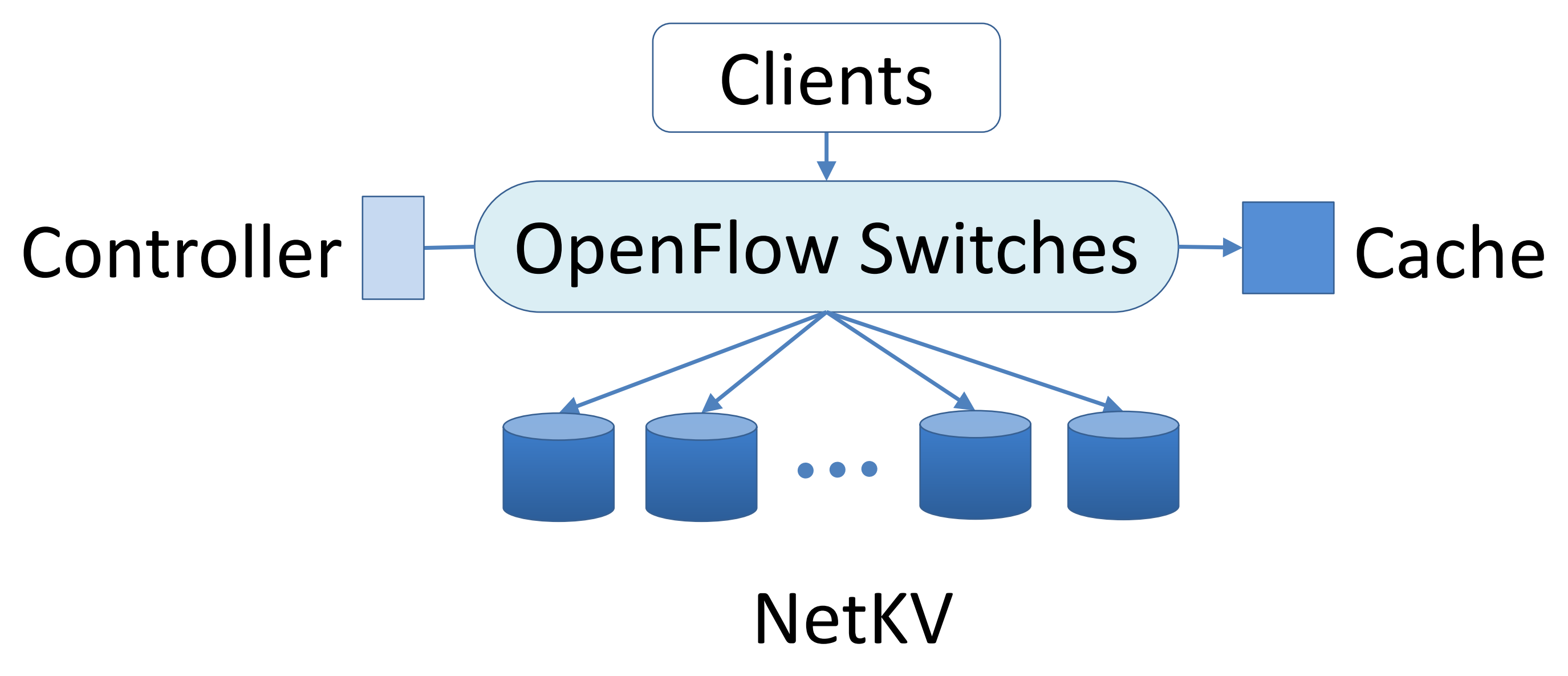
- A small and fast frontend cache can provide good load balance across the backends by only serving the $O(n \log n)$ hottest items, where n is the total number of backend nodes [SOCC'11]
- **Problem:** caching in the data path, introduces system complexity and performance overhead

New Architecture for Efficient Cache-based Load Balancing with Content-based Routing

- Move cache out of the data path by *exploiting SDN* and *deeply optimized switch hardware*.
- Clients encode keys in packet headers. OpenFlow switches maintain forwarding rules for all cached keys, and route requests directly to the cache or backend nodes as appropriate based on content keys.



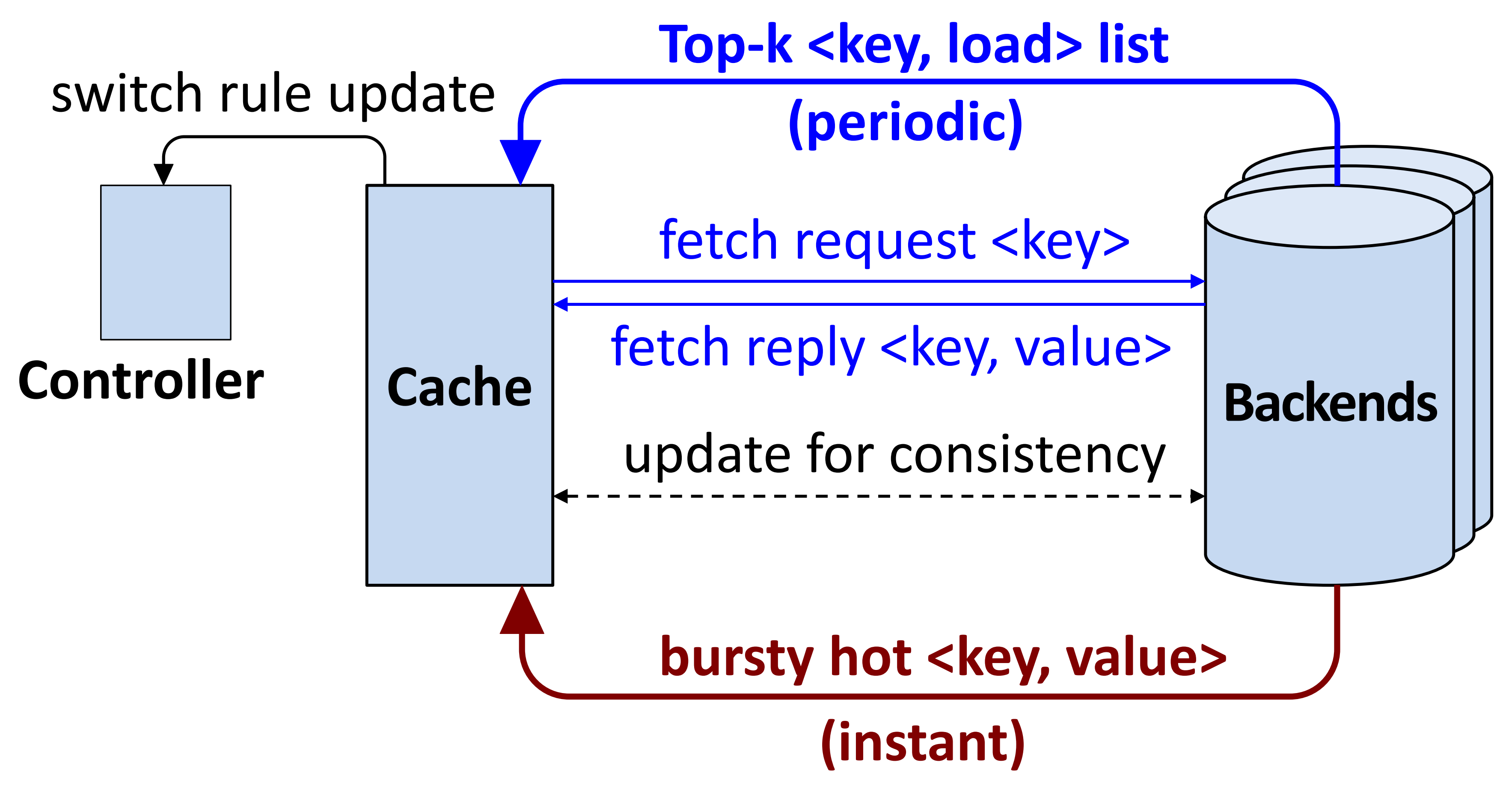




	Look-aside	On-path look-through	NetKV
Client's responsibilities	handle cache misses	nothing (transparent)	encode keys in packet headers
Cache load	100% queries	100% queries	cache hits (likely <30% queries)
Latency with cache miss	three machine transits	two machine transits	one machine transit
Failure points	switches	load balancer, switches	switches
Cache update involves	cache, backends	cache, backends	cache, backends, switches
Cache update rate limit	high	high	low (<10K/s in switch hardware)

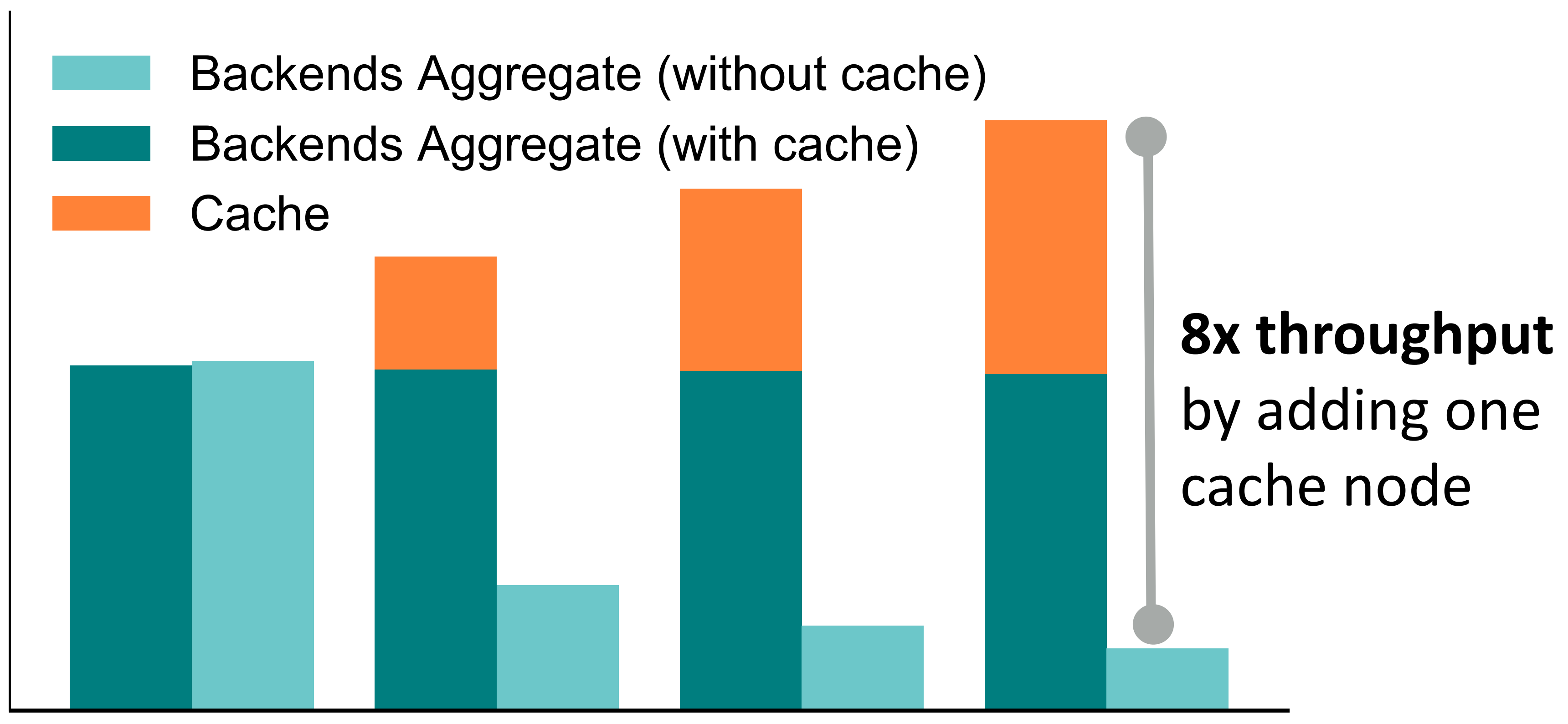
Hybrid Cache Update

Primary goal: minimize unnecessary cache churn



Simulation Results

100 backends, each can serve 200K request per second
Cache size: 5000


Backends Aggregate (without cache)
Backends Aggregate (with cache)
Cache
8x throughput by adding one cache node