Caching 50.5* + Apache Kafka

COS 518: *Advanced Computer Systems*
Lecture 10

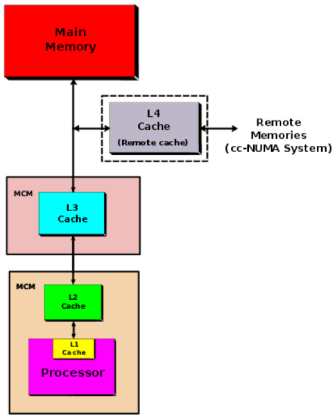
Michael Freedman

* Half of 101

Basic caching rule

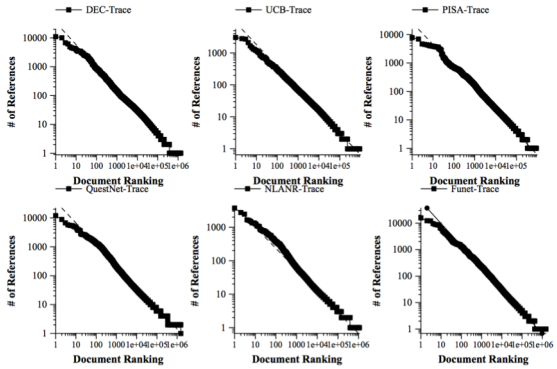
- Tradeoff
  - Fast: Costly, small, close
  - Slow: Cheap, large, far
- Based on two assumptions
  - Temporal location: Will be accessed again soon
  - Spatial location: Nearby data will be accessed soon

Multi-level caching in hardware



https://en.wikipedia.org/wiki/Cache_memory

Caching in distributed systems



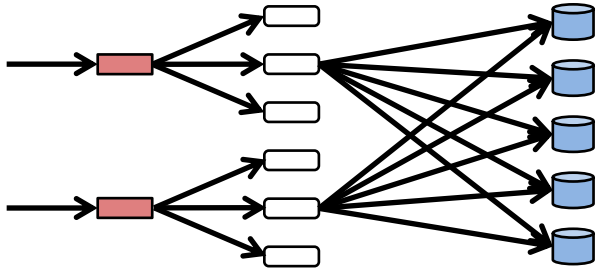
Web Caching and Zipf-like Distributions: Evidence and Implications
Lee Breslau, Pei Cao, Li Fan, Graham Phillips, Scott Shenker

Caching common in distributed systems

- Web
  - Web proxies at edge of enterprise networks
  - "Server surrogates" in CDNs downstream of origin
- DNS
  - Caching popular NS, A records
- File sharing
  - Gnutella & flooding-based p2p networks

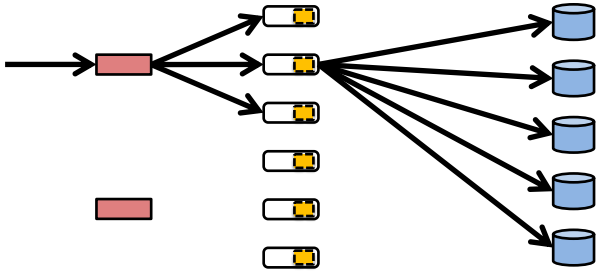
5

Caching within datacenter systems



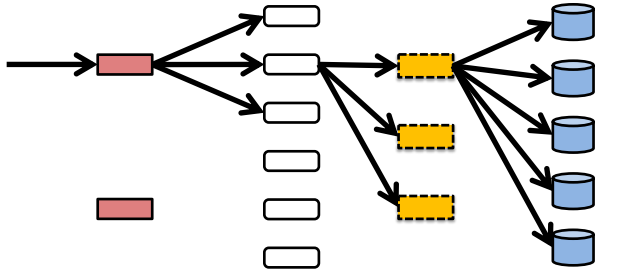
6

Caching within datacenter systems



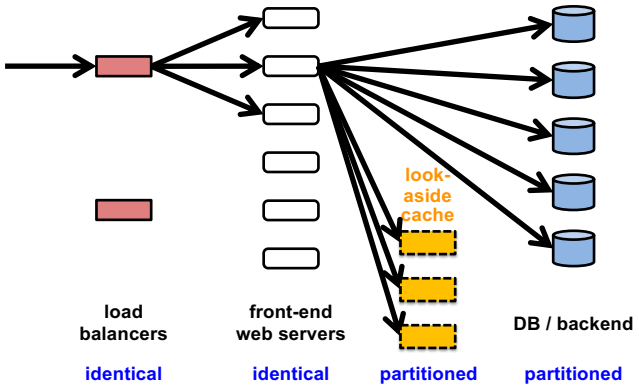
7

Caching within datacenter systems



8

Caching within datacenter systems



9

Cache management

- Write-through
  - Data written simultaneously to cache and storage
- Write-back
  - Data updated only in cache
  - On cache eviction, written “back” to storage

10

Caching within datacenter systems

```
function get_foo(foo_id)
    foo = memcached_get("foo:" . foo_id)
    return foo if defined foo

    foo = fetch_foo_from_database(foo_id)
    memcached_set("foo:" . foo_id, foo)
    return foo
end
```

11

New system / hardware architectures:

New opportunities for caching

12

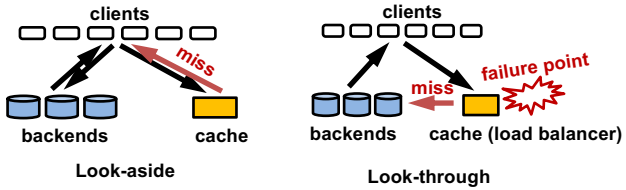
Be Fast, Cheap and in Control with SwitchKV

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

NSDI 2016

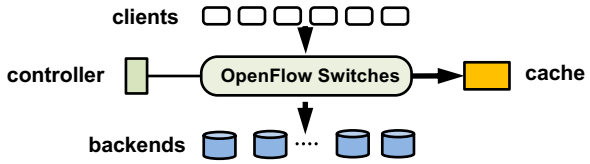
PRINCETON UNIVERSITY
intel
Carnegie Mellon University

Traditional architectures: High-overhead for skewed/dynamic workloads



- 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

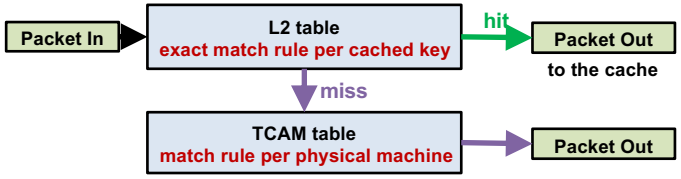


Switches route requests directly to the appropriate nodes

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

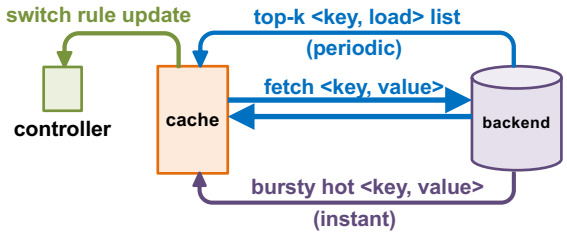
Exploit SDN and switch hardware

- 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



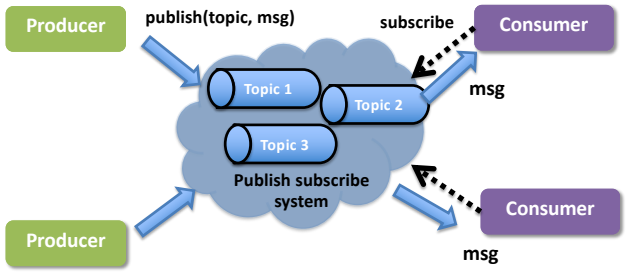
Keep cache and switch rules updated

- 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**



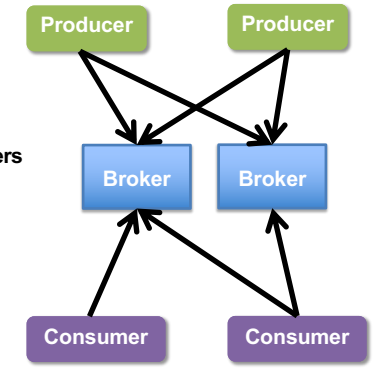
Distributed Queues & Apache Kafka

What is pub sub ?



Automatic load balancing

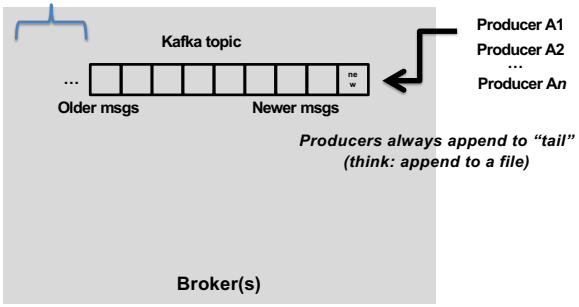
- Parties
  - **Producers** write data to **brokers**
  - **Consumers** read data from **brokers**
- Data stored in **topics**
  - **Topics** split into **partitions**
  - **Partitions** are **replicated** for failure recovery



Topics

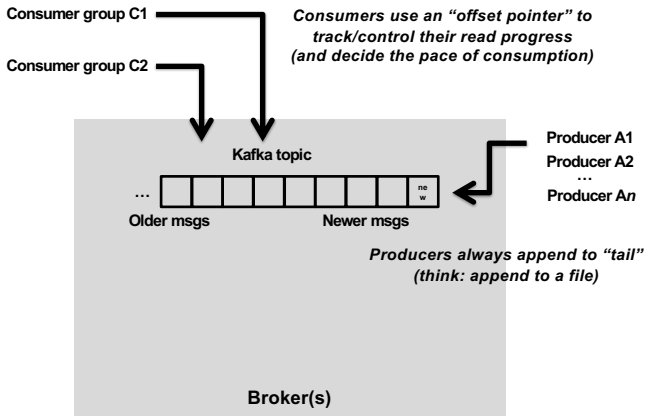
- **Topic:** name to which messages are published

Kafka prunes "head" based on age or max size or "key"



21

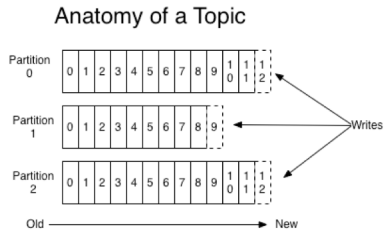
Topics



22

Partitions

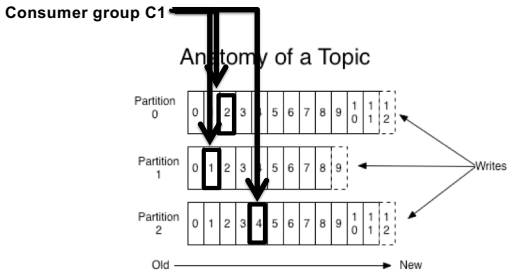
- A topic consists of **partitions.**
- **Partition:** ordered + immutable sequence of msgs, continually appended to
- Number of partitions determines max consumer parallelism



23

Partition offsets

- **Offset**: messages in partitions are each assigned a unique (per partition) and sequential id called the *offset*
  - Consumers track their pointers via *(offset, partition, topic)* tuples



24

Wednesday:

Welcome to BIG DATA

25