Caching 50.5* + Apache Kafka

COS 518: Advanced Computer Systems
Lecture 10
Michael Freedman

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

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

Caching within datacenter systems

- Load balancers
  - Identical
- Front-end web servers
  - Identical
- DB / backend
  - Partitioned
Caching within datacenter systems

- **Load balancers**
  - Identical

- **Front-end web servers**
  - Identical

- **DB / Backend**
  - Partitioned

**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

**New system / hardware architectures:**

- New opportunities for caching
Be Fast, Cheap and in Control with SwitchKV

Xiaozhou Li
Raghav Sethi
Michael Kaminsky
David G. Andersen
Michael J. Freedman
NSDI 2016

**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

**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

**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
  - **Packet In**
    - L2 table (exact rule per cached key)
    - Packet Out to the cache
  - **TCAM table** (match rule per physical machine)
  - **Packet Out**
Keep cache and switch rules updated

- New challenges for cache updates
  - Only cache the hottest $O(\log n)$ items
  - Limited switch rule update rate
- Goal: react quickly to workload changes with minimal updates

What is pub sub?

- 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

Automatic load balancing
### Topics

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

  Kafka prunes “head” based on **age** or **max size** or **“key”**

  ![Kafka topic diagram](image)

  Producers always append to “tail”
  (think: append to a file)

### Partitions

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

  ![Anatomy of a Topic](image)

### 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
Wednesday:

Welcome to BIG DATA