DistCache: Provable Load Balancing for Large-Scale Storage Systems with Distributed Caching

Zaoxing (Alan) Liu

Zhihao Bai, Zhenming Liu, Xiaozhou Li, Changhoon Kim,

Vladimir Braverman, Xin Jin, Ion Stoica

Johns Hopkins University · Carnegie Mellon University · Berkeley University of California · Barefoot Networks

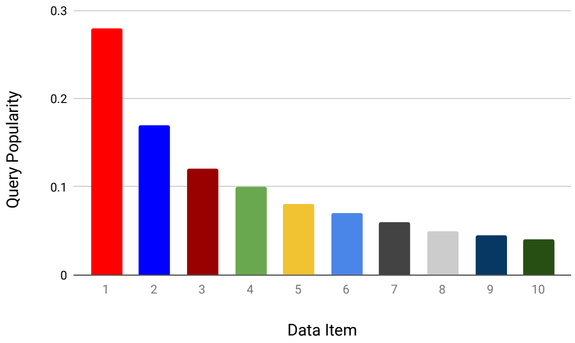
Large-scale cloud services need large storage clusters

- Major cloud services serve billions of users.

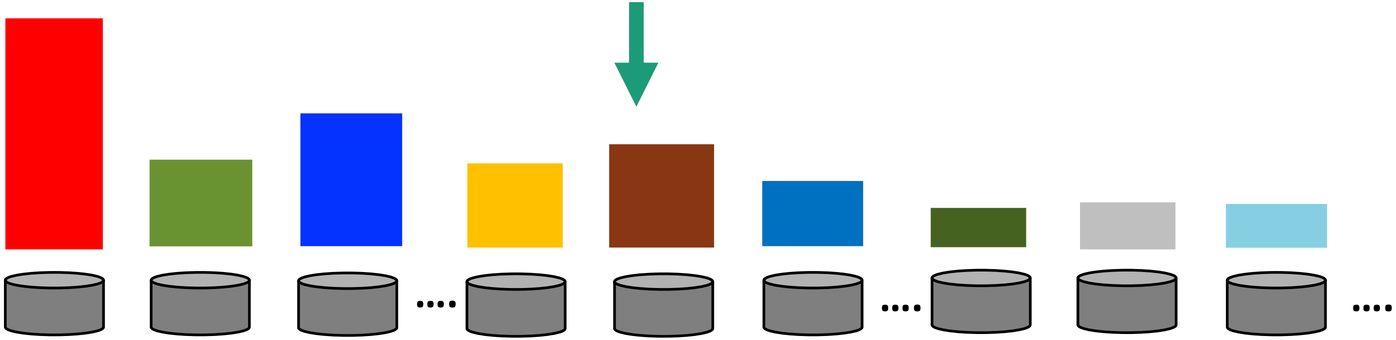
Large datacenter clusters

Storage servers have load imbalance issue

- Typical workloads [Sigmetrics’12]:
  - Highly skewed.
  - Dynamic.



Server load



The skewness of the workload brings imbalance.

Solutions to mitigate the load imbalance

- Consistent hashing and related.
  - Do not handle dynamic and skewed workloads.

- Data migration or replication.
  - Large system and storage overhead.
  - High cache coherence cost.

- Front-end cache as a load balancer.
  - Low update overhead.
  - Work for arbitrary workloads.

Solutions to mitigate the load imbalance

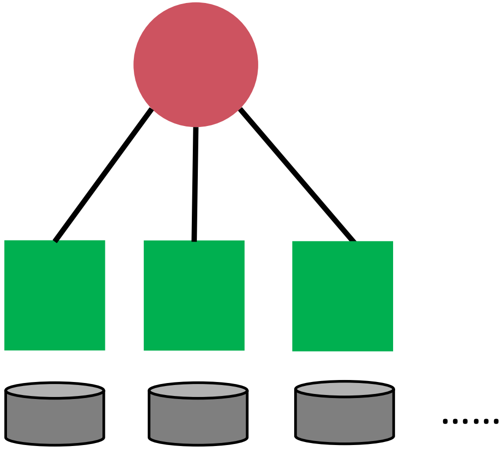
- Consistent hashing and related.
  - Do not handle dynamic and skewed workloads.

- Data migration or replication.
  - Large system and storage overhead.
  - High cache coherence cost.

- Front-end cache as a load balancer.
  - Low update overhead.
  - Work for arbitrary workloads.

Prior work: Fast, small cache alleviates load imbalance

Cache hottest O(n log n) items [SoCC'11]



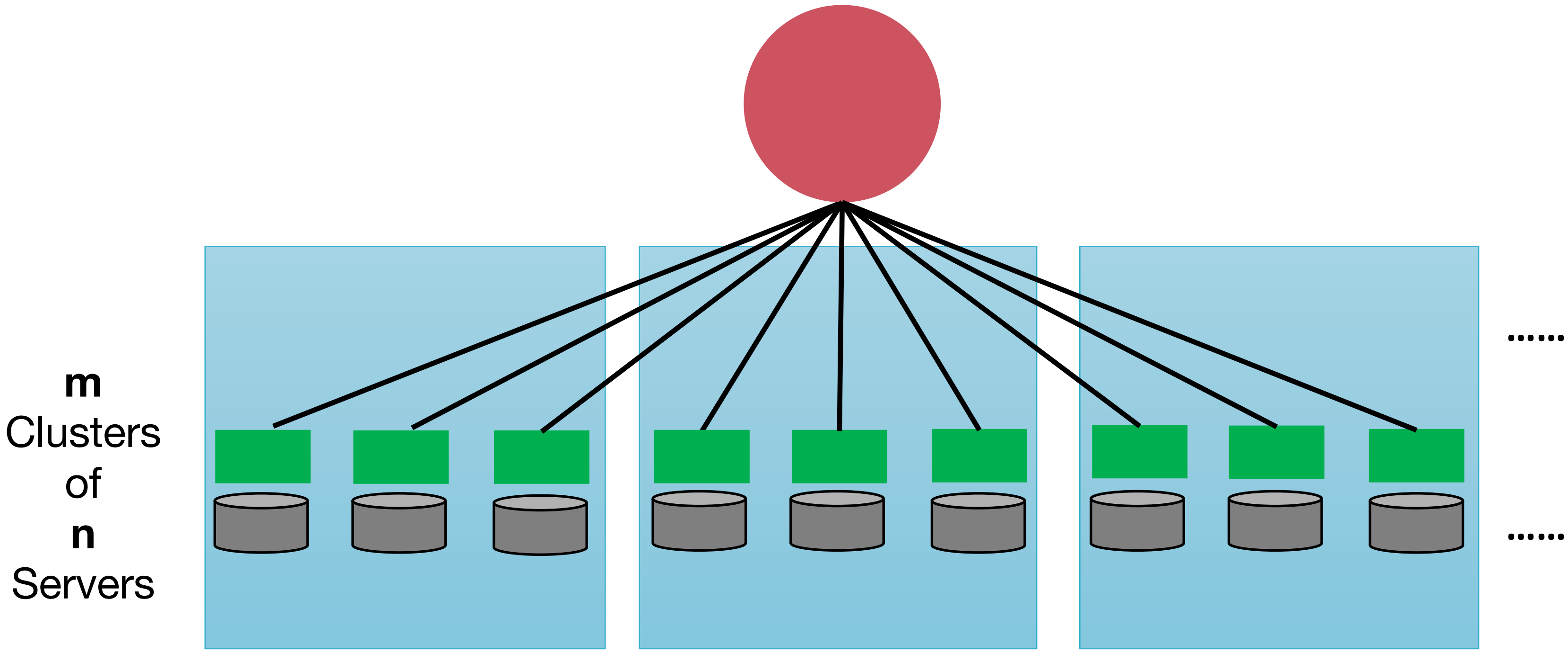
Server load
is balanced

n servers

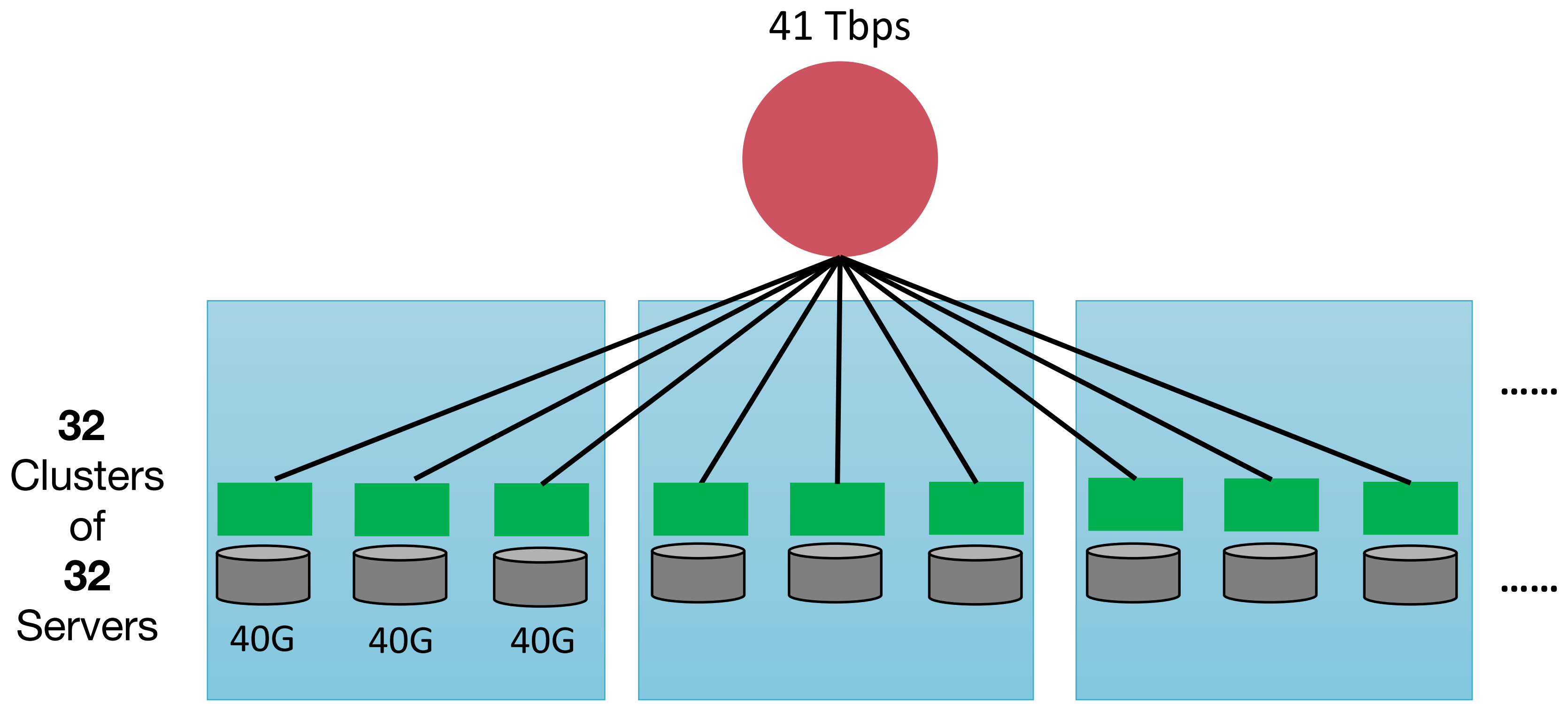
Application to cluster-scale:
[NSDI'16, SOSP'17]

A cache node brings load balancing in a cluster.

Strawman: Big, fast cache for inter-cluster load balancing

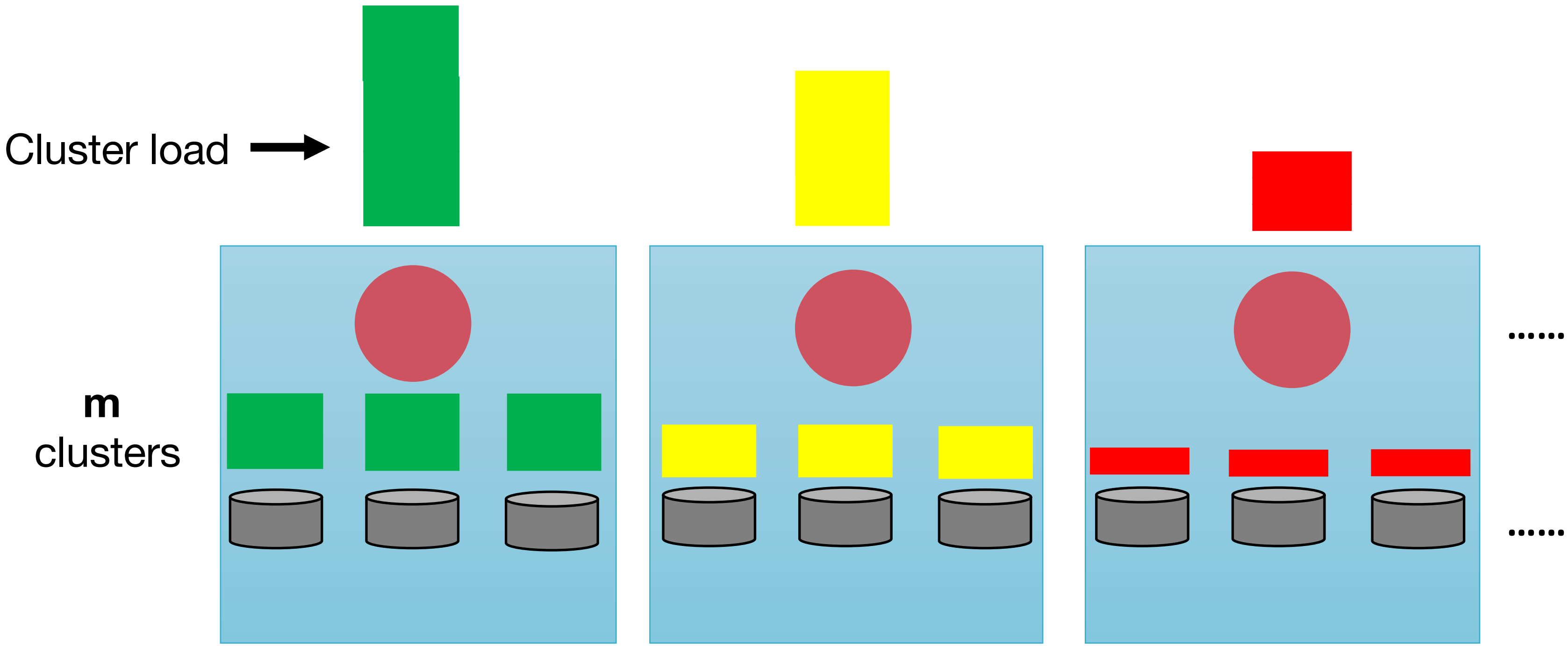


One “Big” cache is infeasible



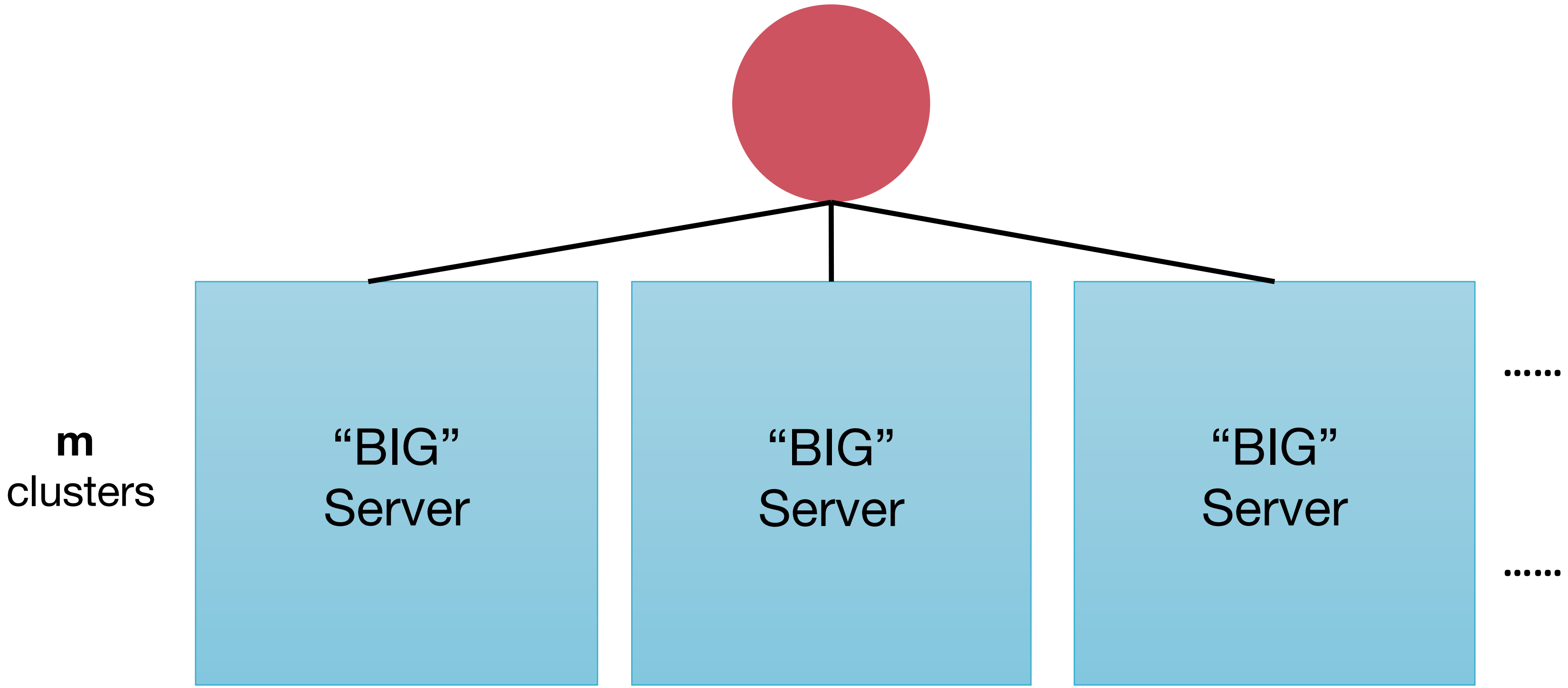
One big cache is not scalable.

First, balance the load within each cluster



Second, balance the load between clusters

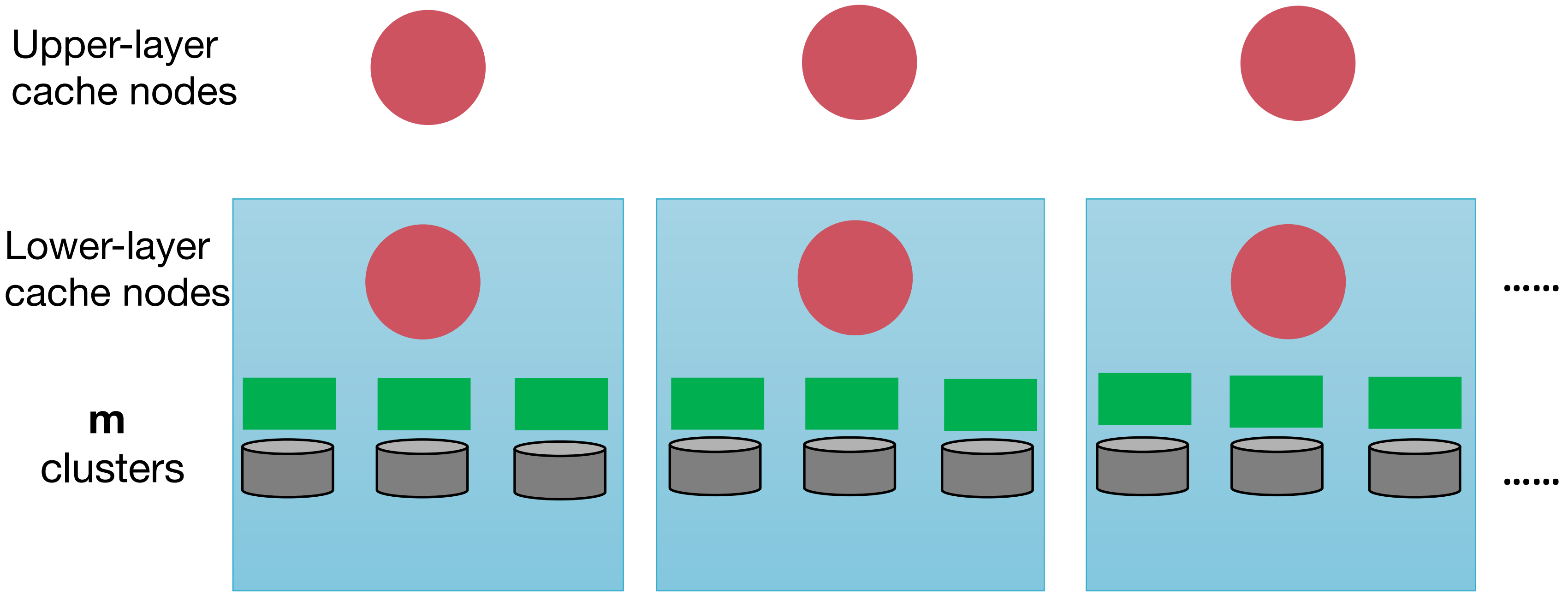
Cache hottest O(m log m) items.



We need to avoid using big node anywhere.

DistCache: Distributed caching as load balancer

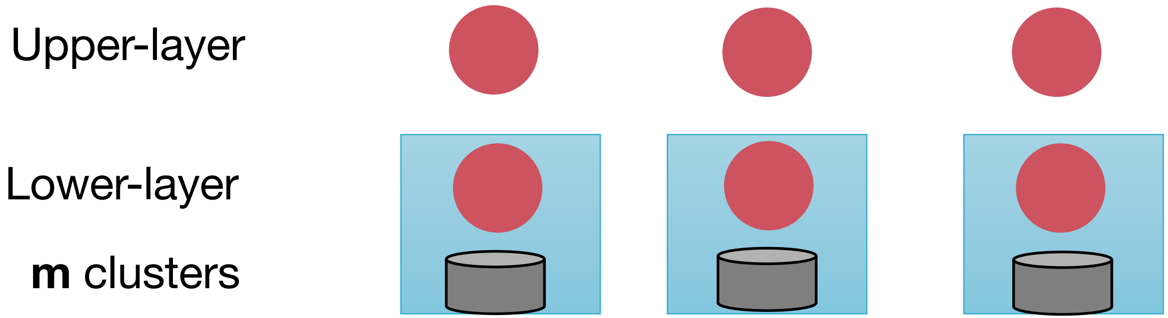
Cache hottest O(m log m) items.



Provable, Practical, General mechanism.

Natural goals on a distributed caching mechanism

Ideally, DistCache should be as good as
"**one big cache**" to absorb O(m log m) hottest items.

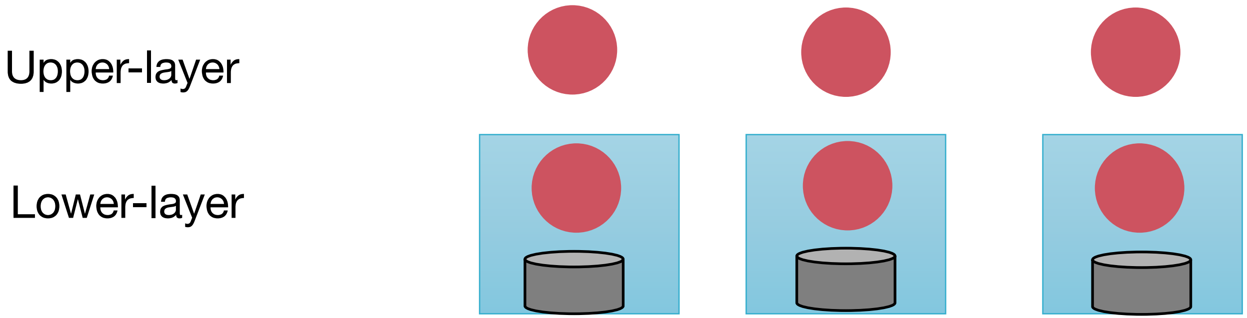


Natural goals on a distributed caching mechanism

Ideally, DistCache should be as good as
“**one big cache**” to absorb O(m log m) hottest items.

To achieve “one big cache”:

- Support **ANY** query workload to hottest O(m log m) items.
- Each cache node is **NOT** overloaded.
- Keep cache coherence with **MINIMAL** cost.



Design Challenges of DistCache

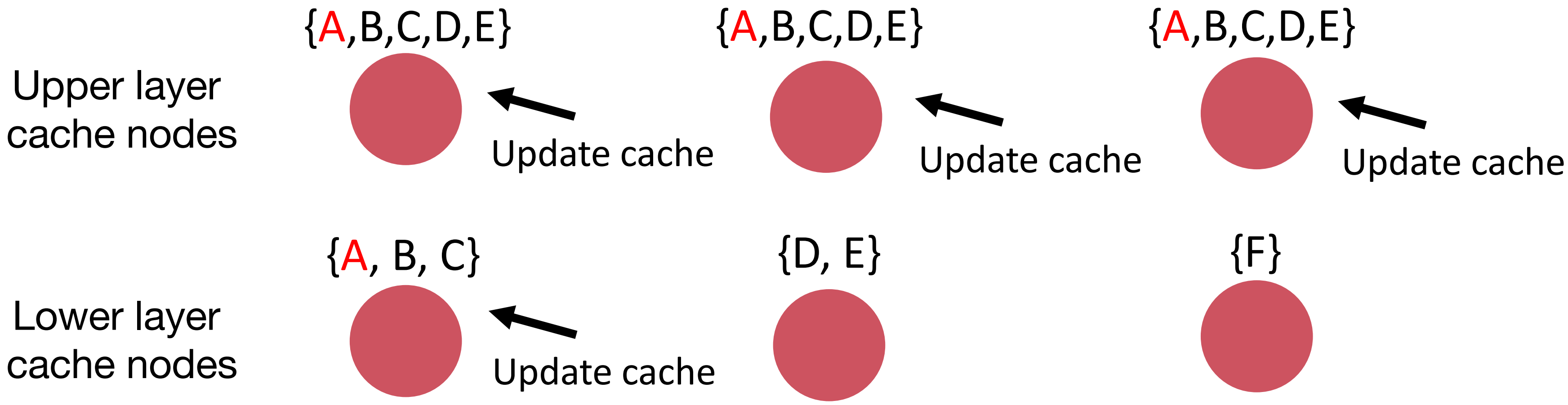
- Challenge #1: How to allocate cached items?
  - Do not overload any cache node.
  - Do not incur high cache coherence cost.
- Challenge #2: How to query the cached items?
  - Provide best and stable cache query distribution.
- Challenge #3: How to update the cached items?
  - Two-phase update to ensure cache coherence.

Design Challenges of DistCache

- Challenge #1: How to allocate cached items?
  - Do not overload any cache node.
  - Do not incur high cache coherence cost.
- Challenge #2: How to query the cached items?
  - Provide best and stable cache query distribution.
- Challenge #3: How to update the cached items?
  - Two-phase update to ensure cache coherence.

Challenge #1: How to allocate the cached items?

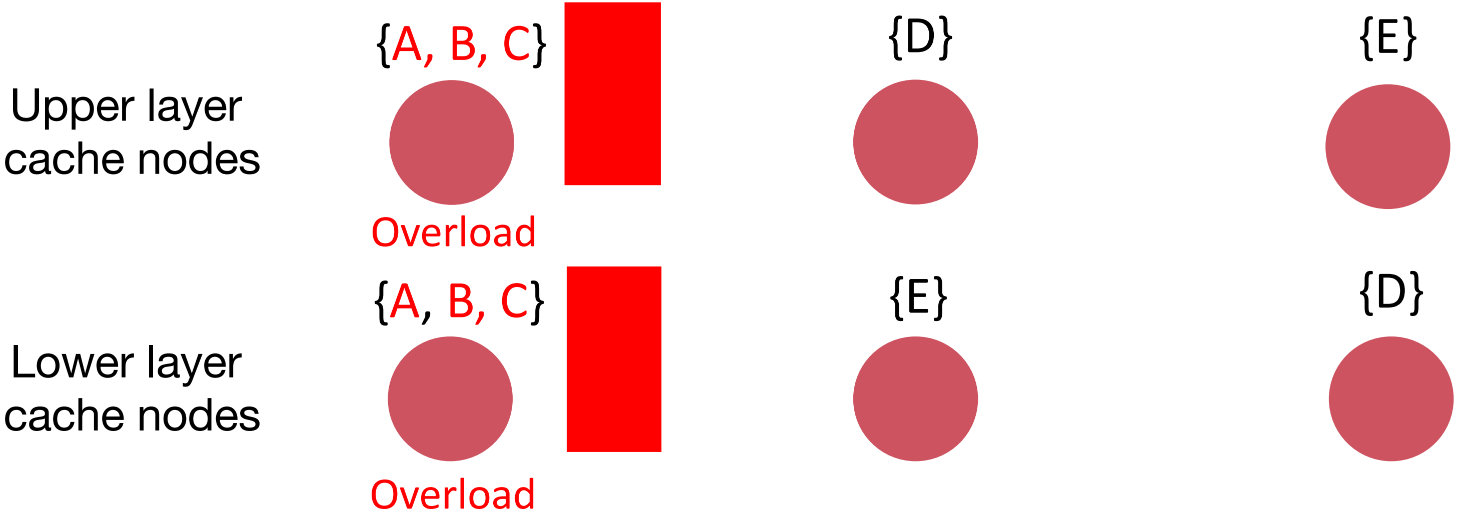
Strawman Sol #1: Cache-Replication



Cache-Replication incurs high cache coherence cost.

Challenge #1: How to allocate the cached items?

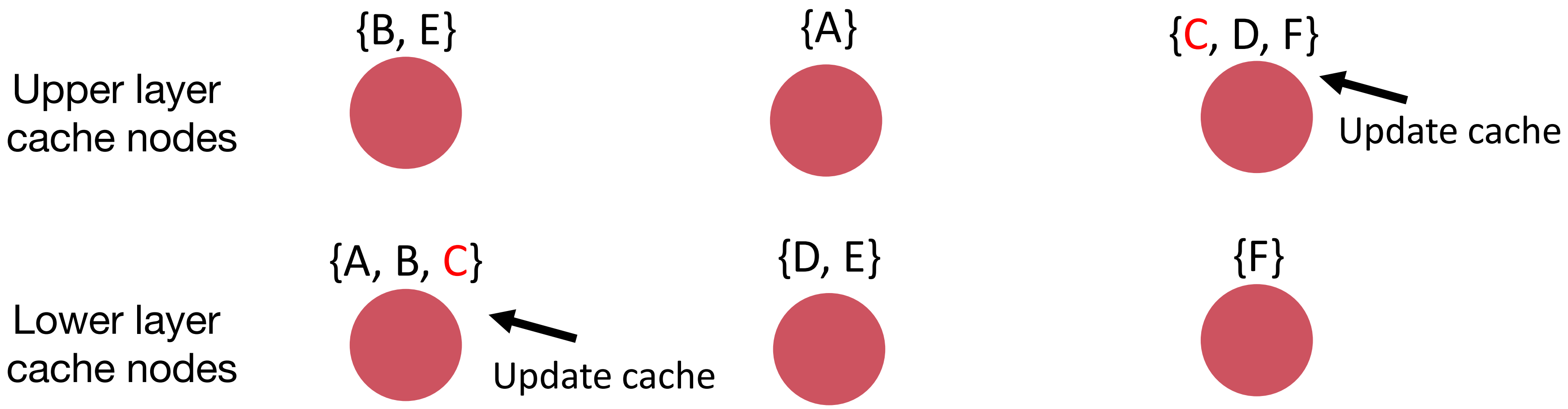
Strawman Sol #2: Cache-Partition



Cache-Partition could put too many hottest items into the same cache node.

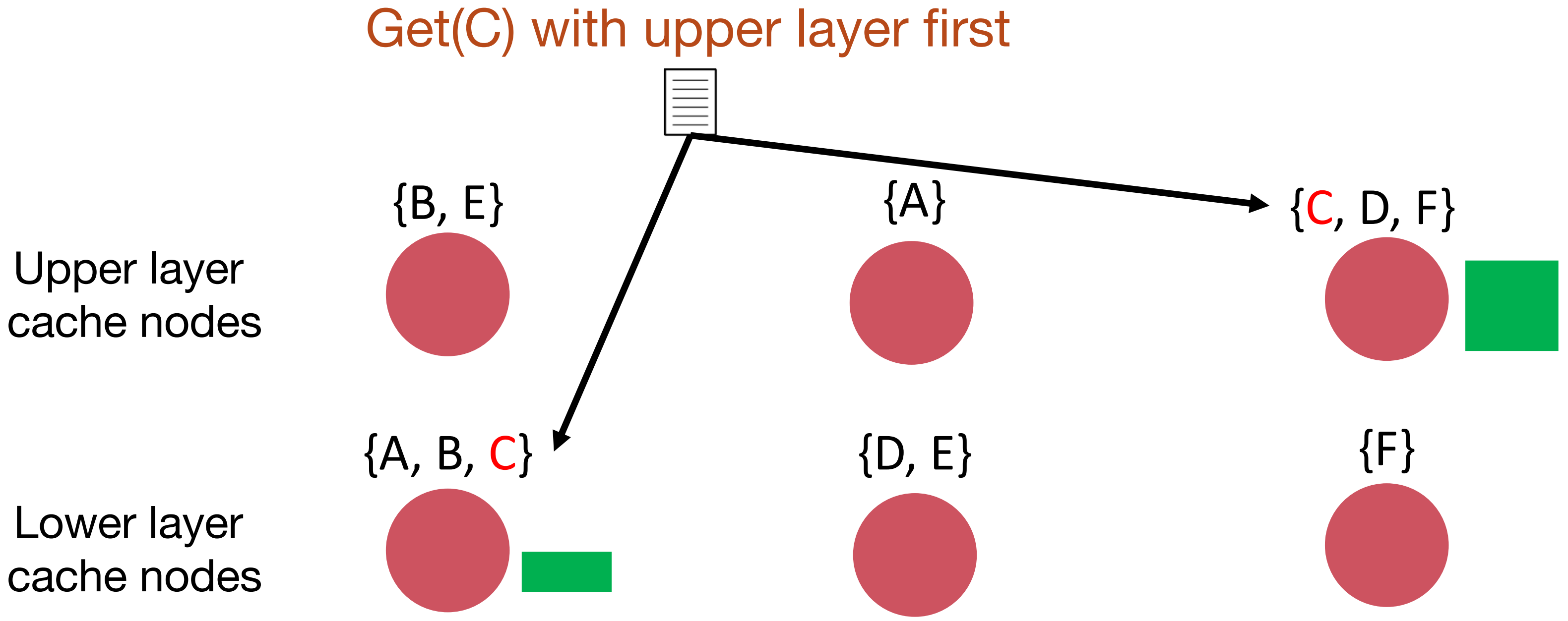
Independent hashes to allocate the cached items

Two independent hashes H1 and H2 to allocate hot items



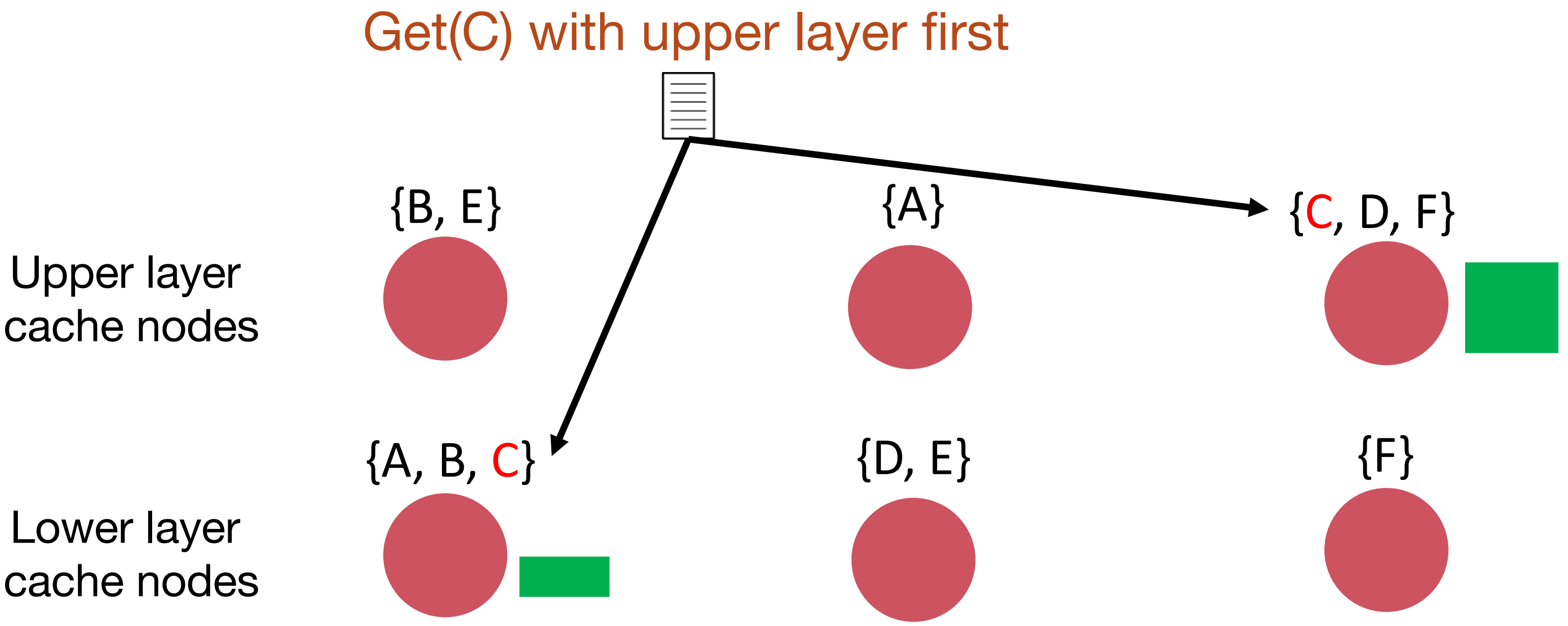
- Stable and best cache allocation.
- Small cache coherence cost.

Challenge #2: How to query the cached items?



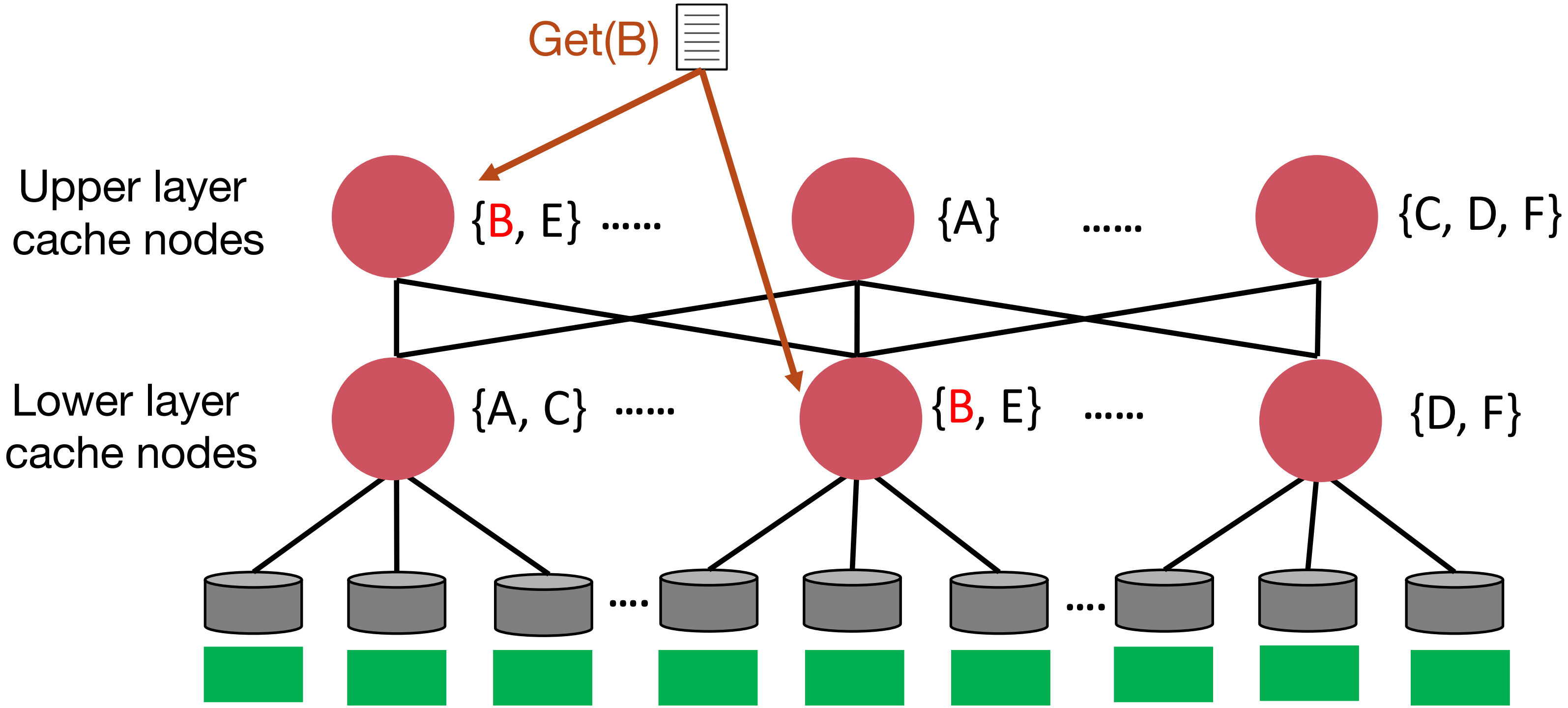
Querying item with upper layer first does not guarantee best throughput.

Power-of-two-choices to query the cached items



Power-of-two-choices to route the queries guarantee stable throughput.

Putting together: DistCache



- Independent hashes to allocate cache items.
- Power-of-two-choices of current cache loads to route queries.

Theoretical Guarantee behind DistCache

For m storage clusters:

- DistCache absorbs any query workload to the hottest O(m log m) items.

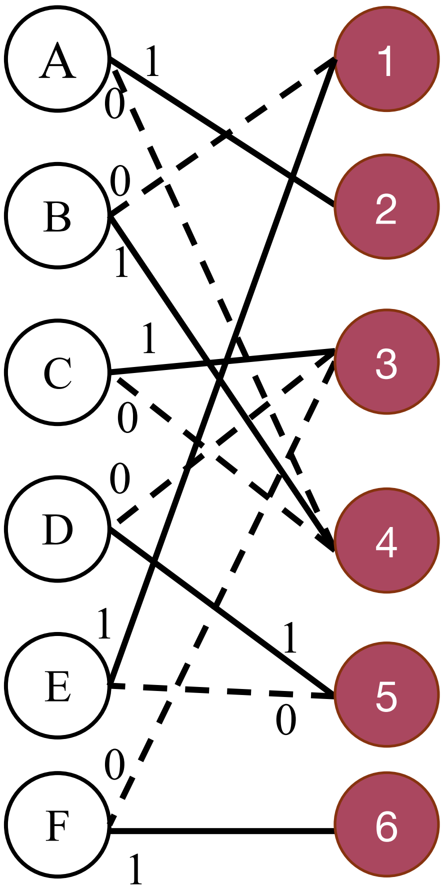
with the following condition:

- Query rate for a single item is no larger than ½ of one cache node's throughput. (No more half of a cluster!)

Proof Sketch: Convert to a perfect matching problem

Proofs leverage tools from **expander graph**, **network flow**, and **querying theory**

Hottest items



Upper layer cache nodes

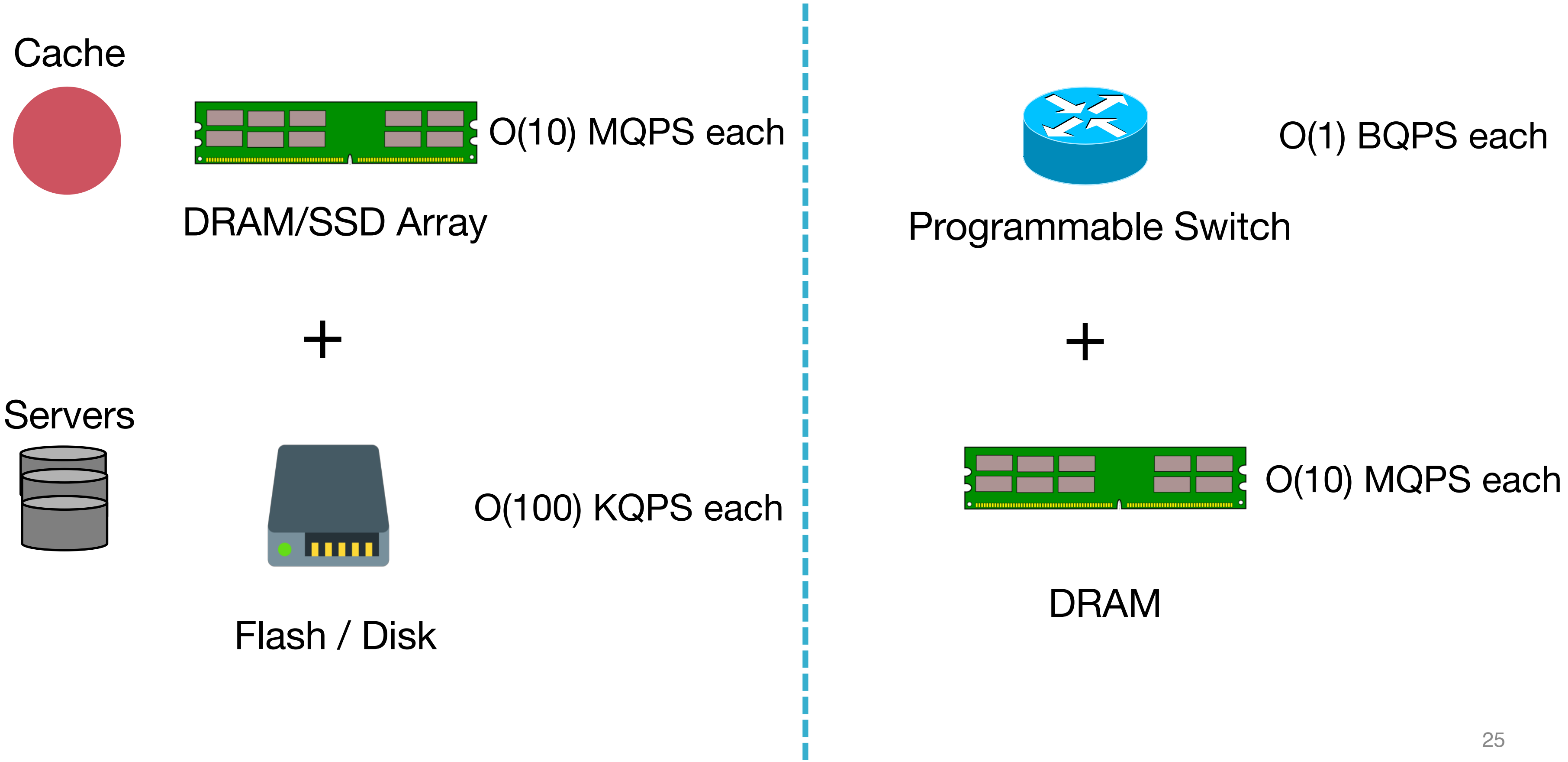
Lower layer cache nodes

Our PoT query can find a perfect match for any query workload distribution.

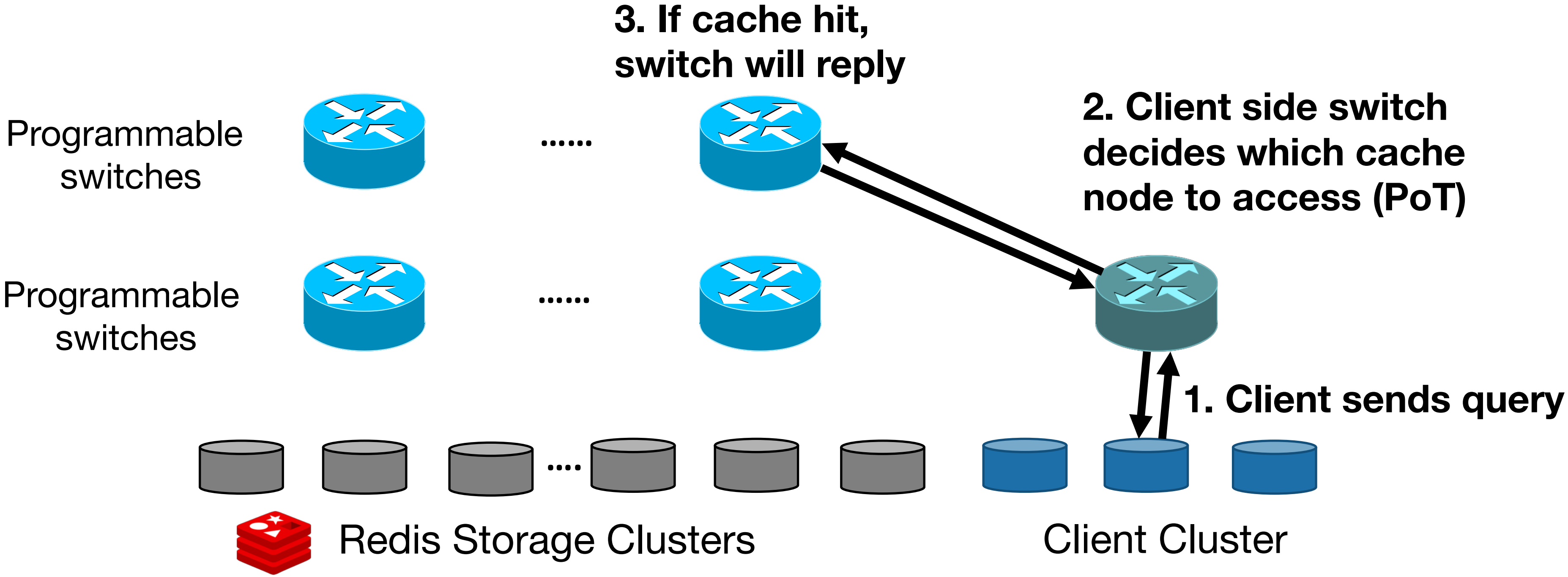
Remarks of the DistCache Analysis

- The numbers of cache nodes in two layers can be different as long as **m** isn't too small.
- The throughput of cache nodes can be different.
- Aggregated throughput is almost same as "big cache".

Example Deployment Scenarios of DistCache

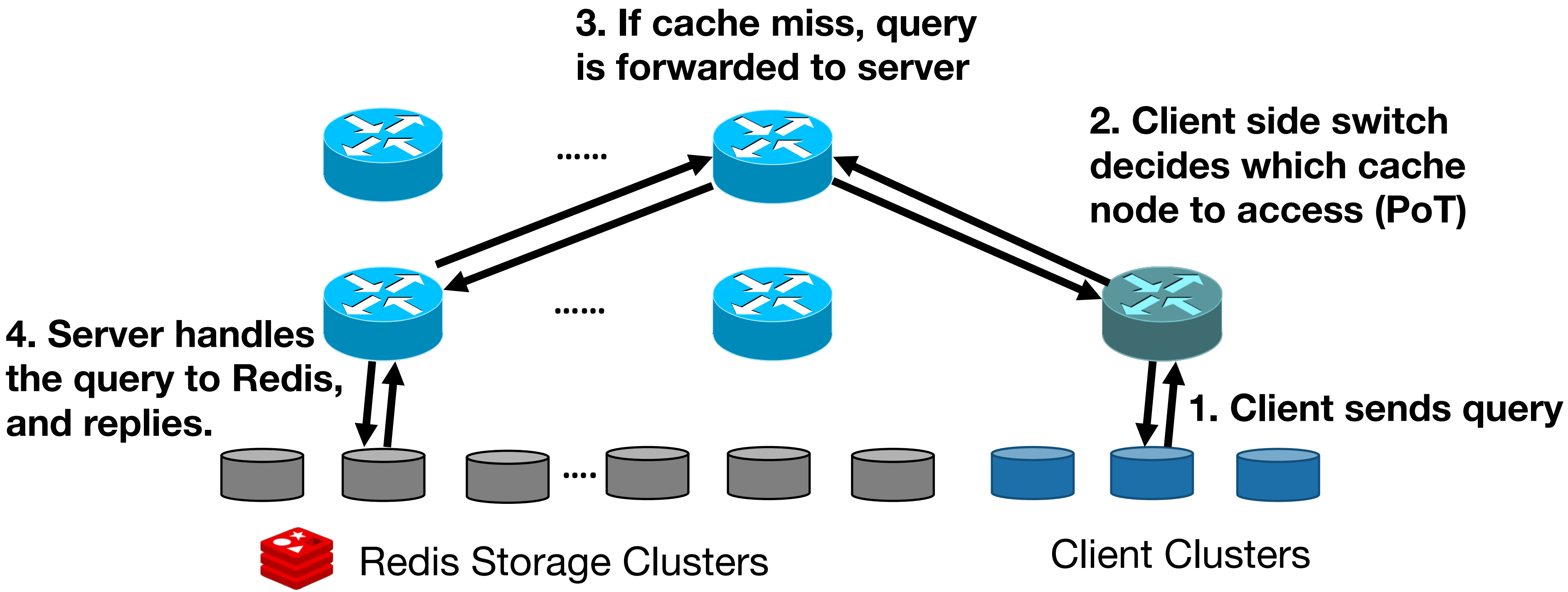


Case Study: Switch-based distributed caching



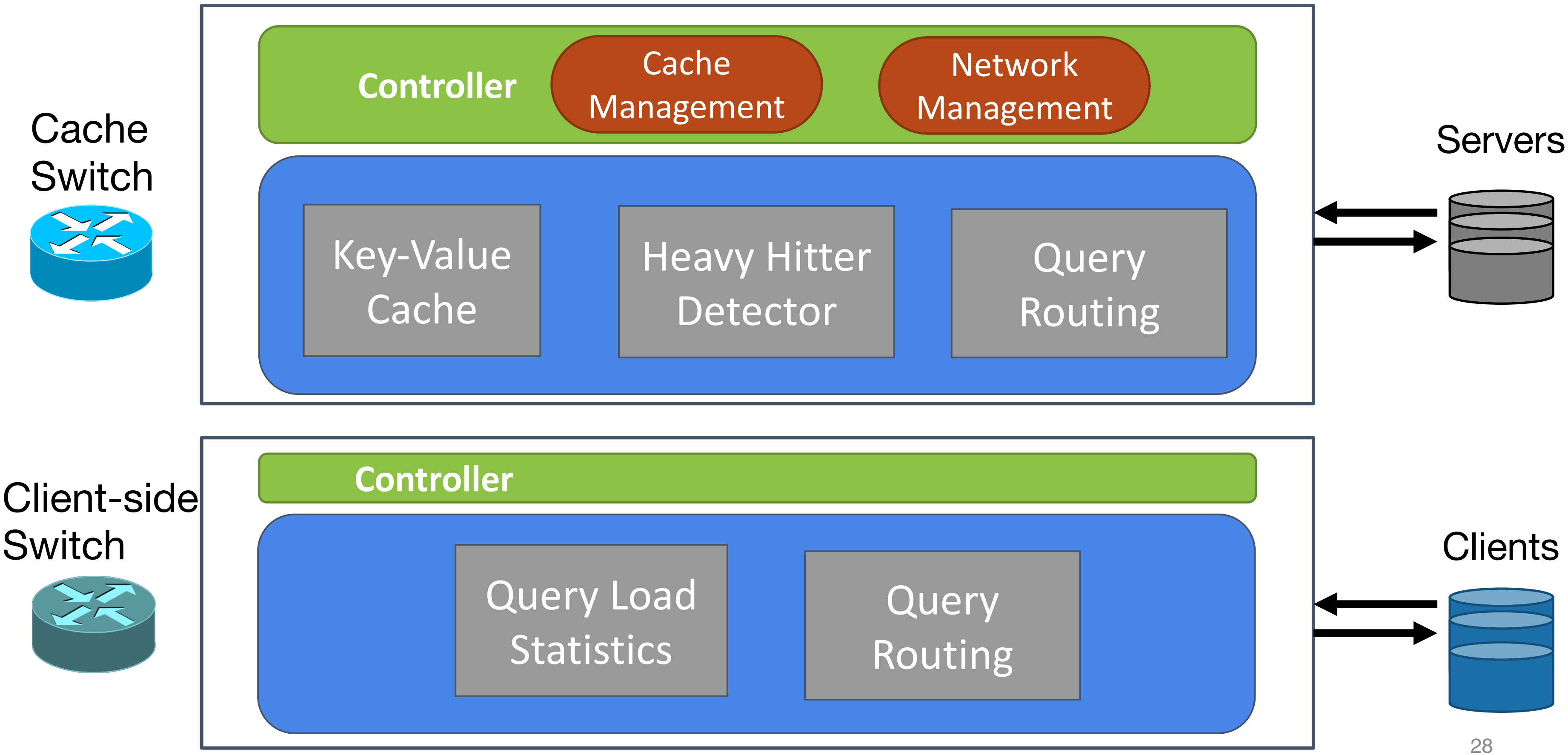
When cache hit, cache switch will reply the query immediately.

Case Study: Switch-based distributed caching



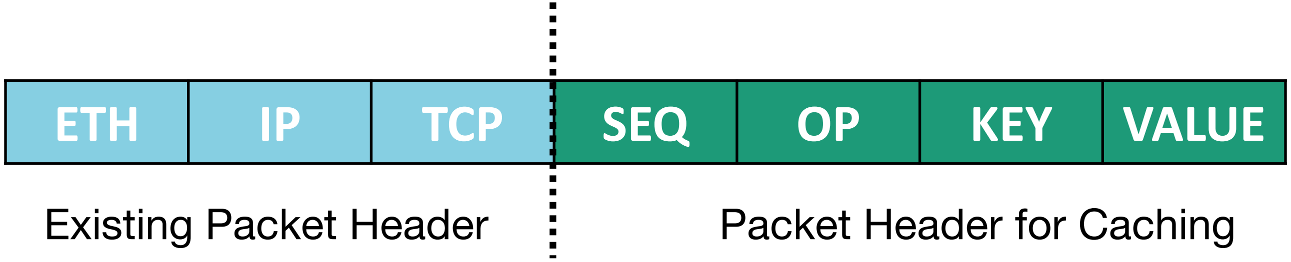
When cache miss, query is handled by the server.

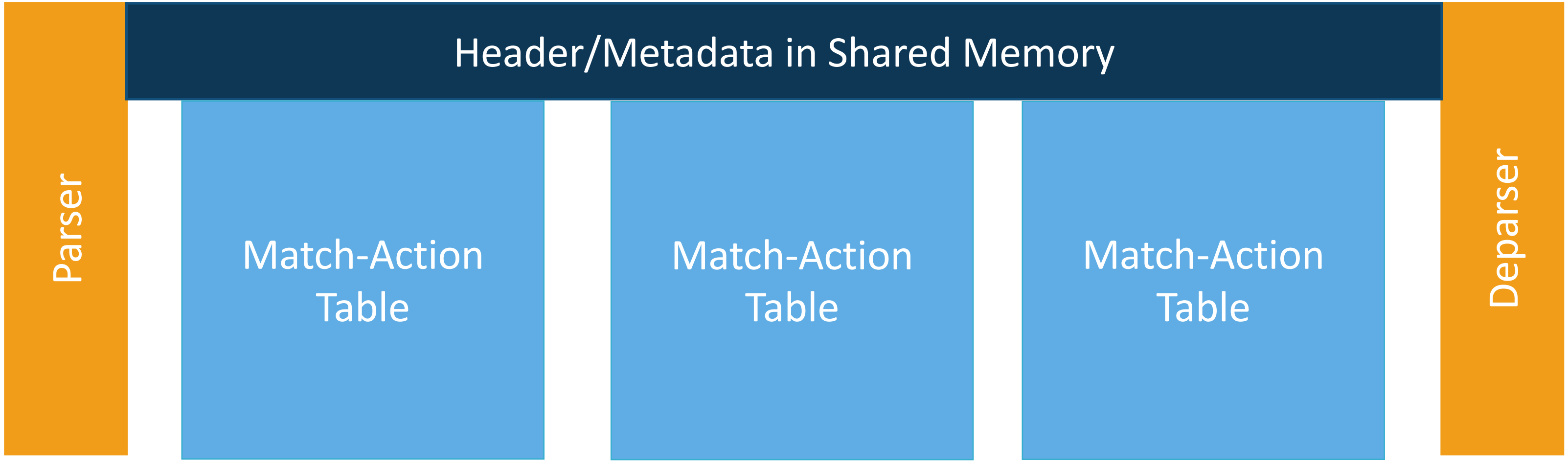
Implementation Overview



P4: Programmable Protocol-Independent Packet Processing

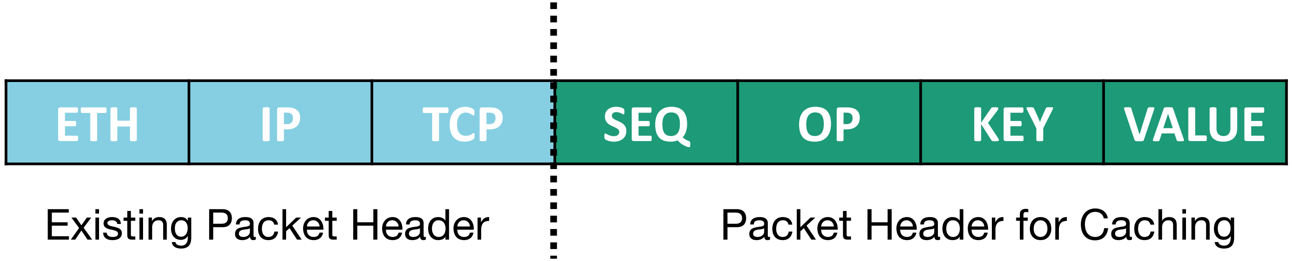
User-defined
Packet Format:

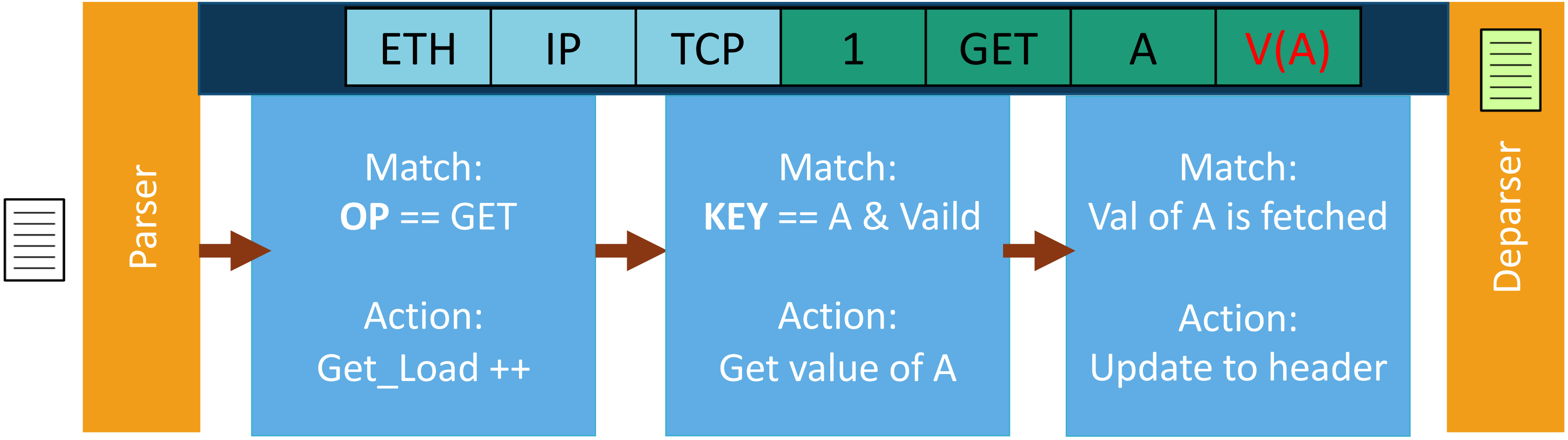




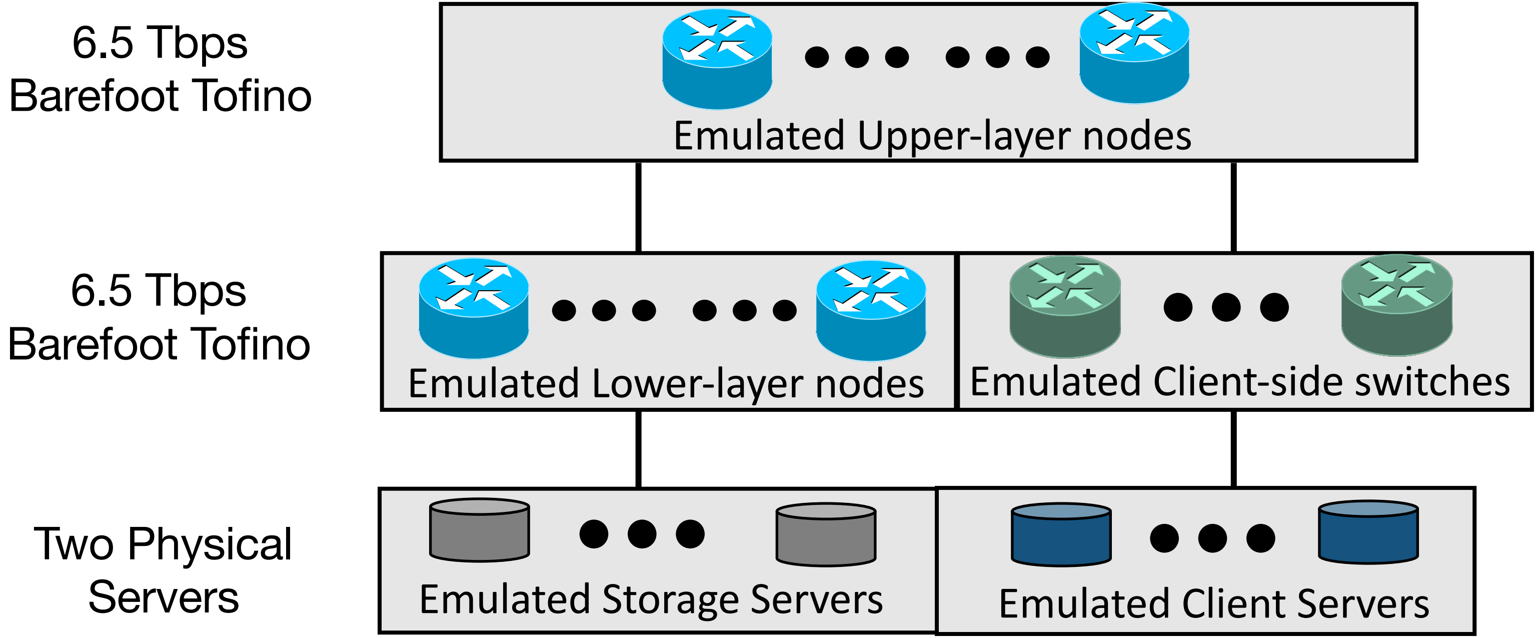
P4: Programmable Protocol-Independent Packet Processing

User-defined Packet Format:





Evaluation Setup



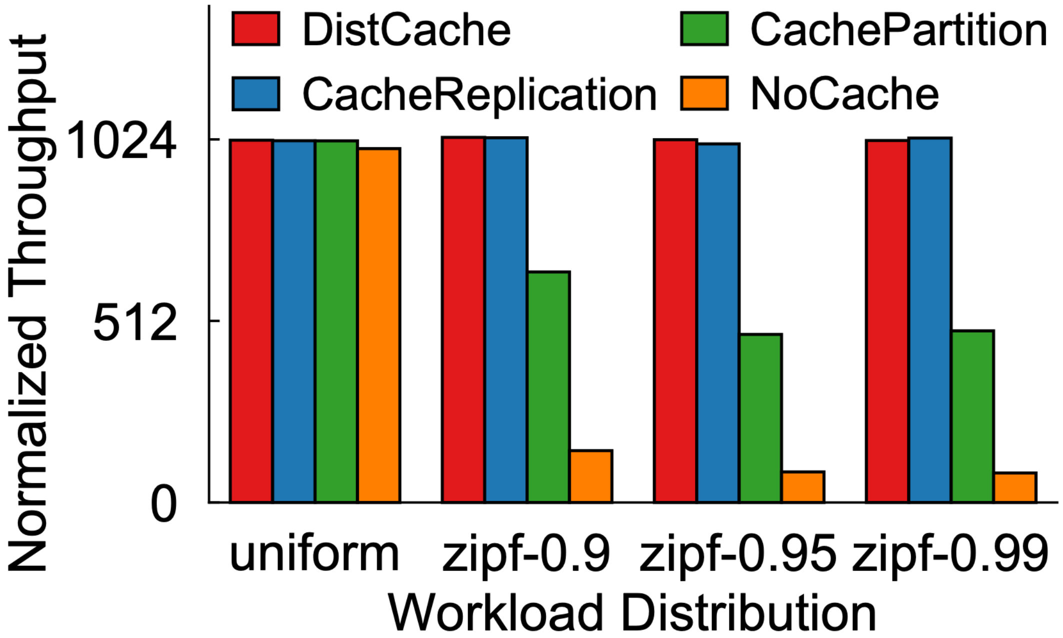
- Baselines: NoCache, Cache-Partition, Cache-Replication.

Evaluation Takeaways

- For read queries, DistCache works as good as Cache-Replication.

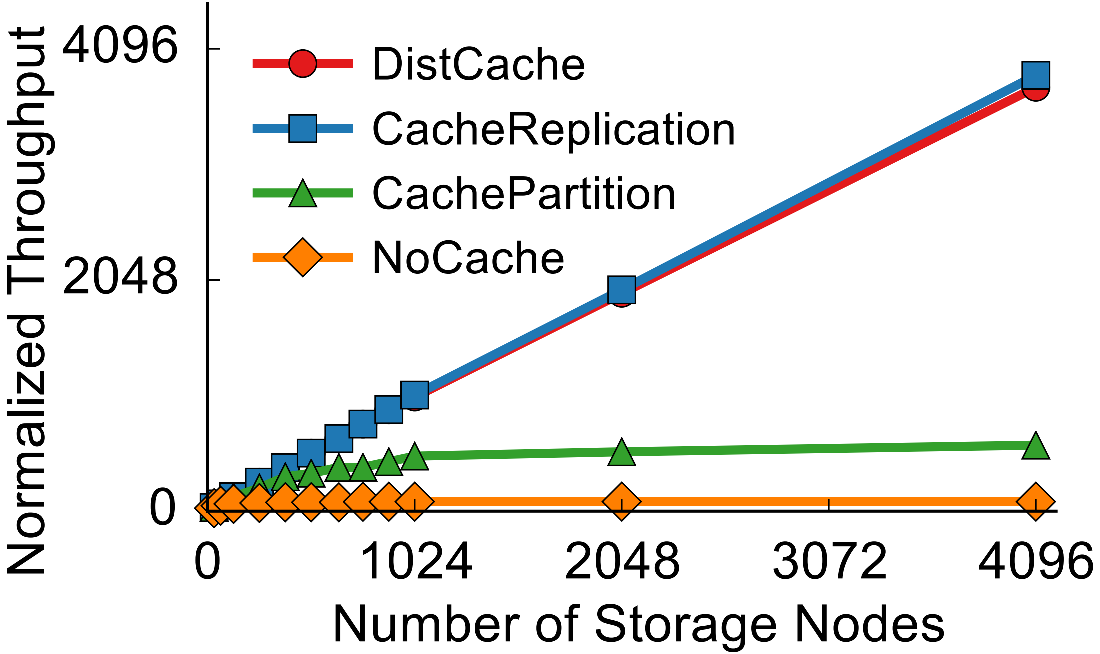
- For write queries, DistCache has performed significantly better:
  - When write ratio (<0.3), better throughput.
  - When write ratio (>0.3), as good as Cache-Partition.

DistCache balances the loads of different clusters



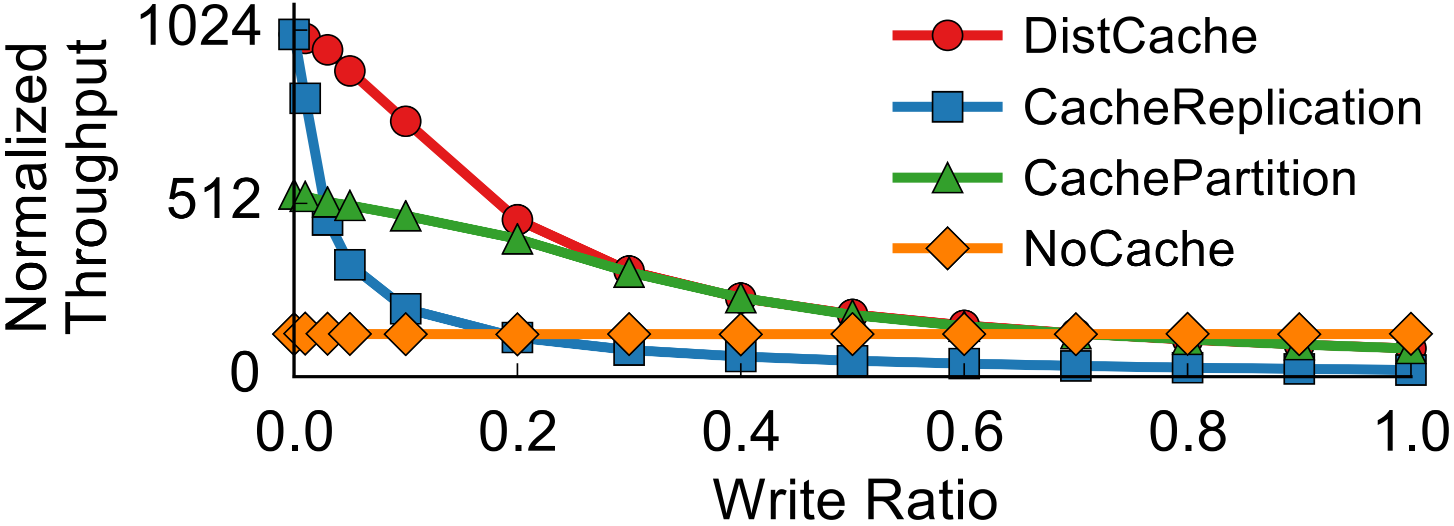
DistCache offers nearly perfect throughput for skewed workloads

DistCache scales linearly with the number of nodes



DistCache can support very large storage clusters.

DistCache incurs small cache coherence cost



Under Zipf-0.99 workload, DistCache offers best write throughput.

Conclusions

- DistCache is a general distributed caching mechanism to ensure load balancing crossing many storage clusters.
- DistCache requires simple primitives (independent hashing, power-of-two-choices routing).
- DistCache provides near-perfect throughput with rigorous theoretical guarantees.