Dynacache: Dynamic Cloud Caching

Asaf Cidon, Assaf Eisenman,
Mohammad Alizadeh and Sachin Katti



Stanford University



MIT

Memcached Driving Web Application Performance

- Web-scale applications heavily reliant on memory caches



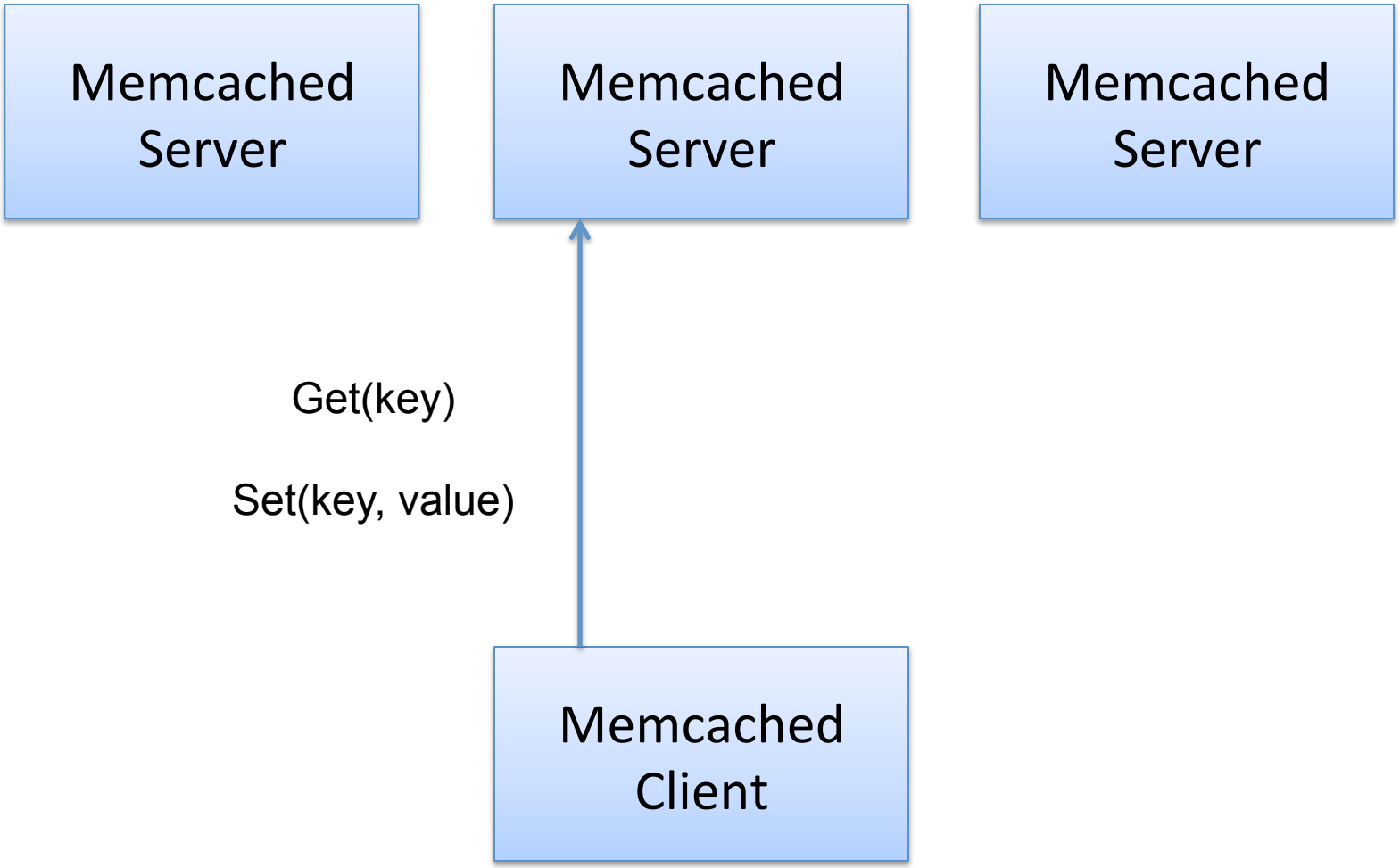
- Facebook: 1000s servers, 5 billion requests per second

Memcached Hit-rate Impacts Performance

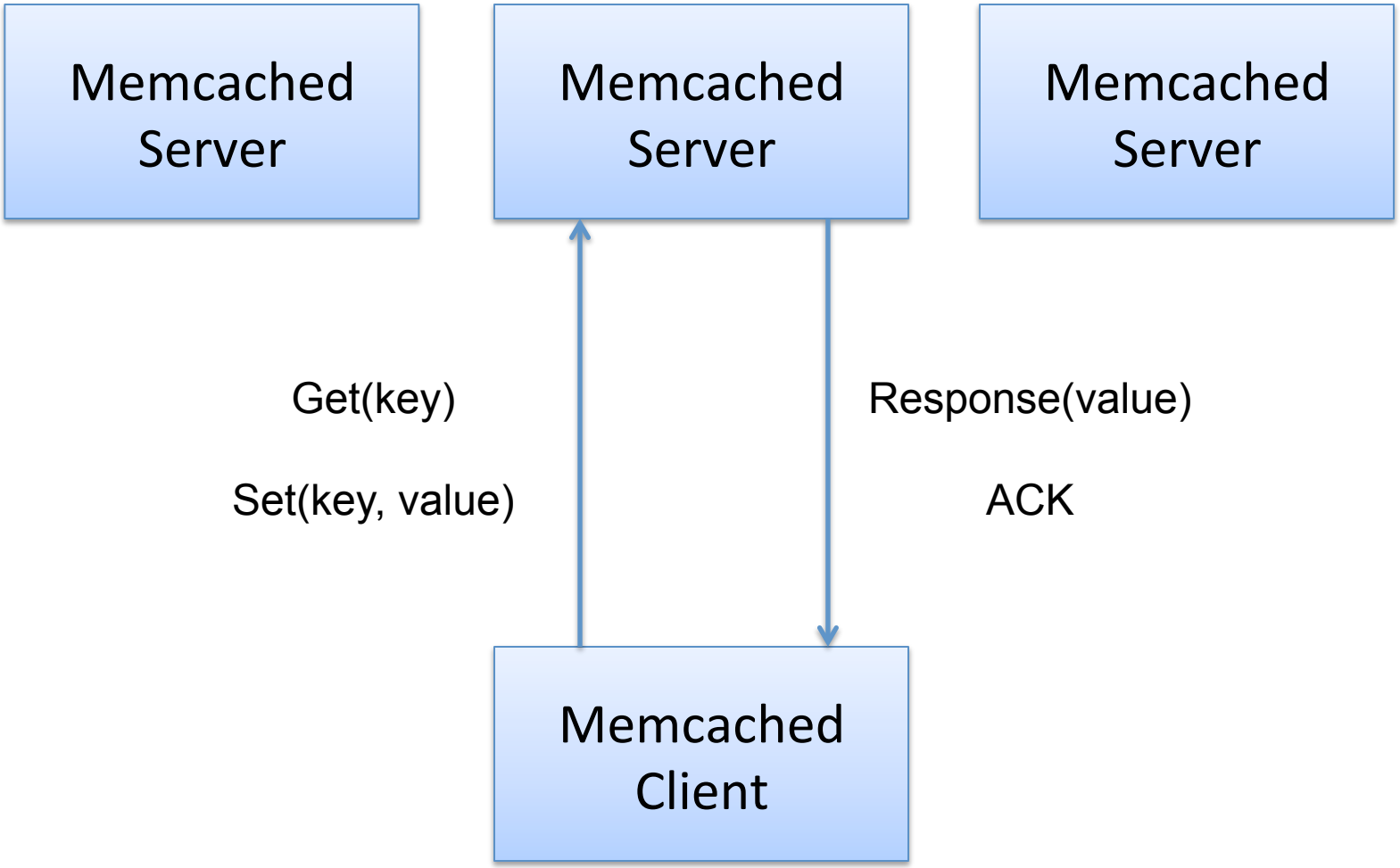
- +1% cache hit-rate → 35% speedup
  - Old latency: 374 µs
  - New latency: 278 µs
  - Facebook study: Atikoglu et al [Sigmetrics '12]

- End-to-end speedup even greater
  - User queries wait on hundreds of requests

How Does Memcached Work?



How Does Memcached Work?



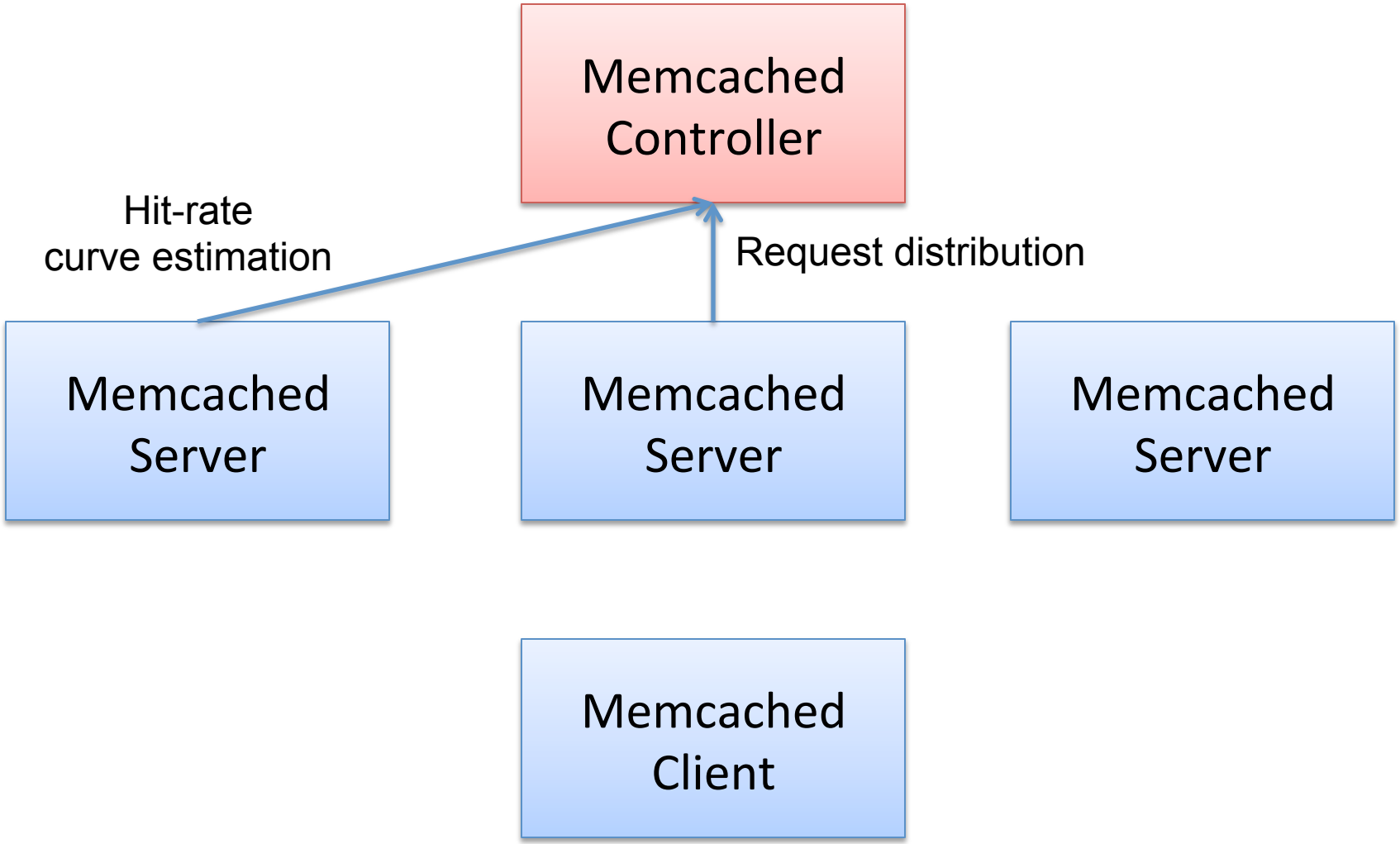
Memcached Server: Sounds Familiar?

	CPU Cache	Memcached Server
Memory Management	Fixed-sized entries First Come First Serve	Fixed-sized slabs First Come First Serve
Eviction Policy	LRU	LRU
Resource Allocation Across Applications	First Come First Serve	First Come First Serve

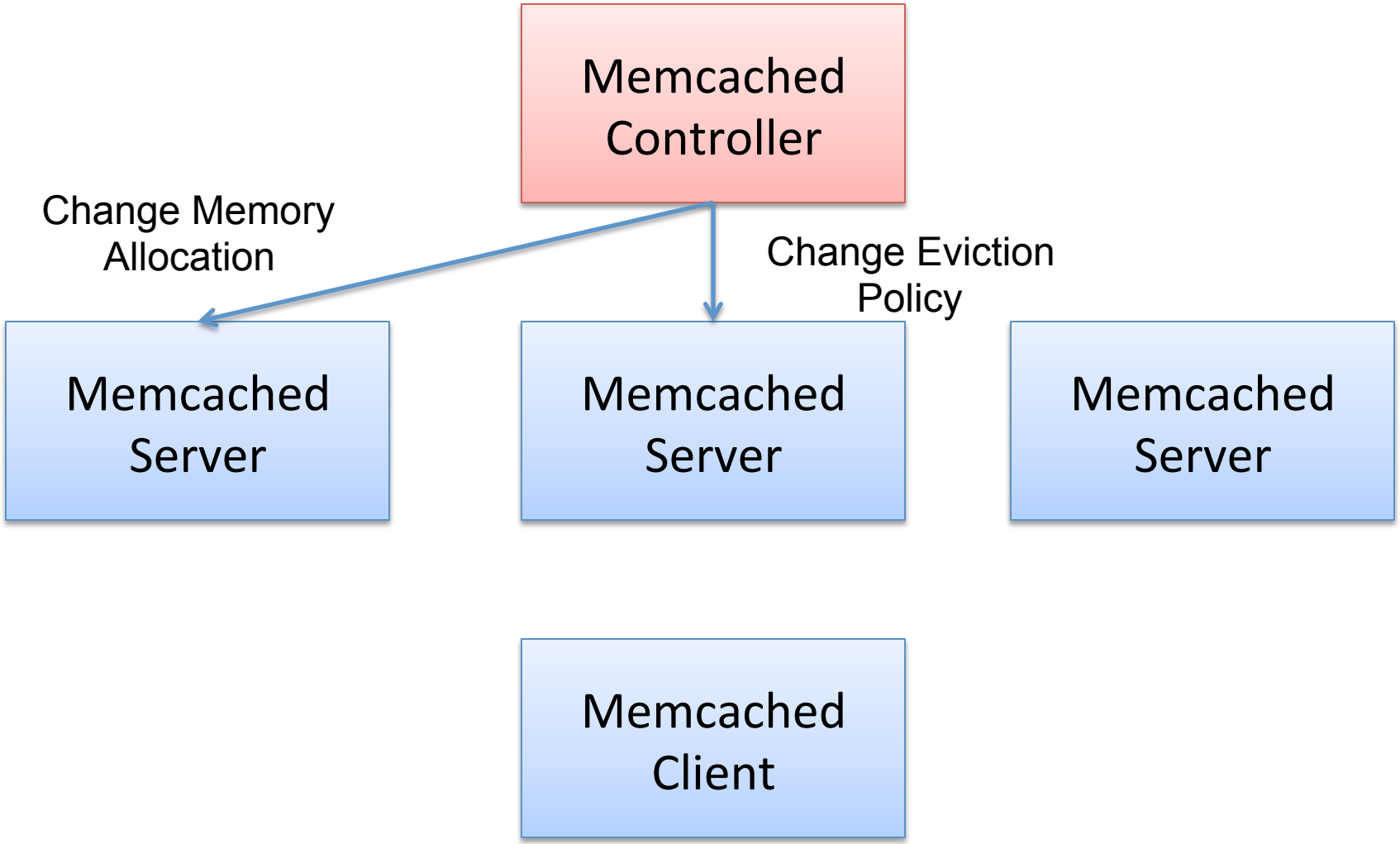
Not the Same!

	CPU Cache	Memcached Server
Request size	Even	Highly variable
Available resources and complexity	Constrained	Unconstrained
Workload predictability	Predictable	Unpredictable

Dynacache: Dynamic Cache Controller



Dynacache: Dynamic Cache Controller



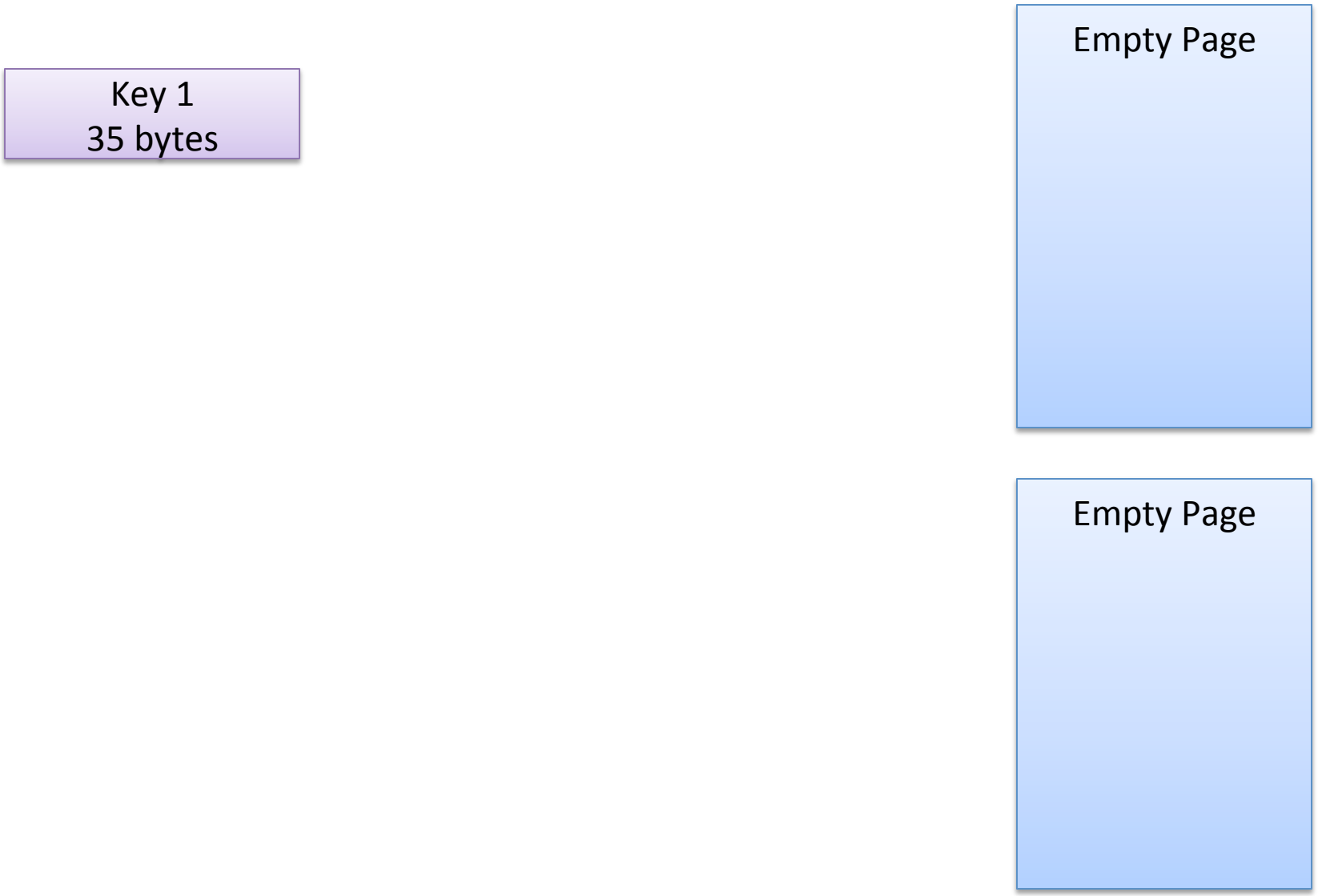
Opportunities for Dynacache

1. Dynamic memory allocation
2. Dynamic eviction policy
3. Dynamic resource allocation across applications
4. …

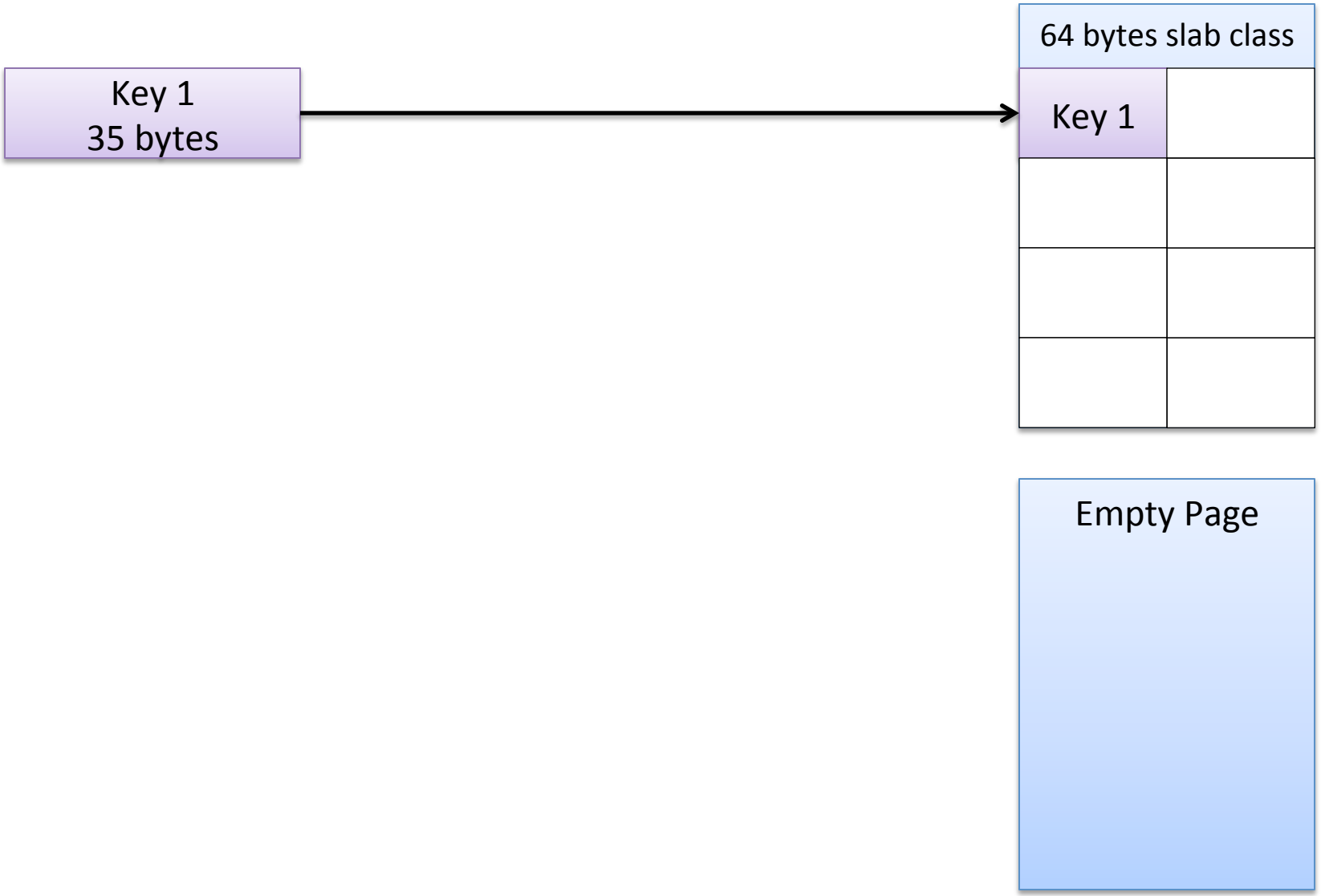
Opportunities for Dynacache

1. **Dynamic memory allocation**
2. Dynamic eviction policy
3. Dynamic resource allocation across applications
4. …

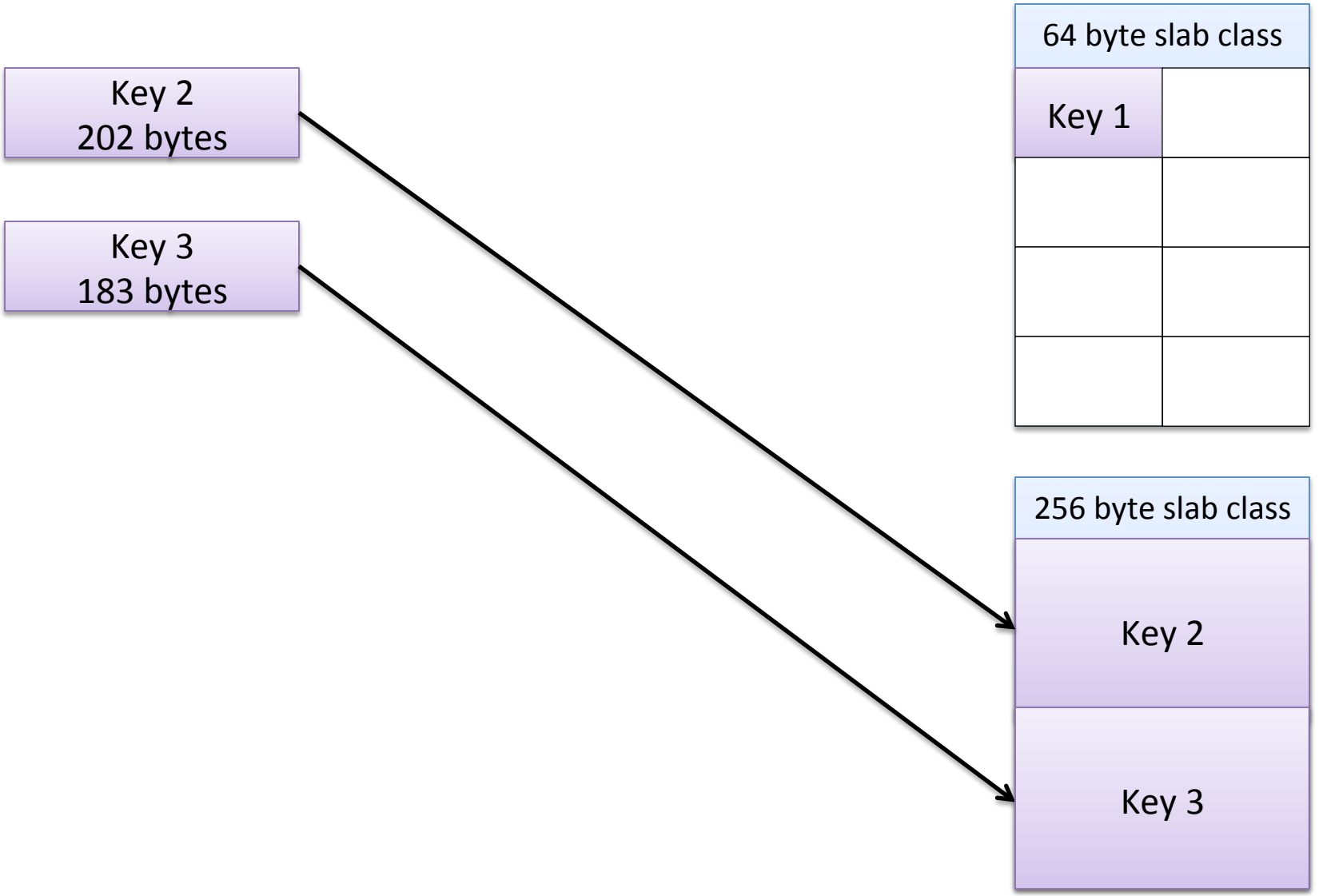
Memory Allocation in Memcached



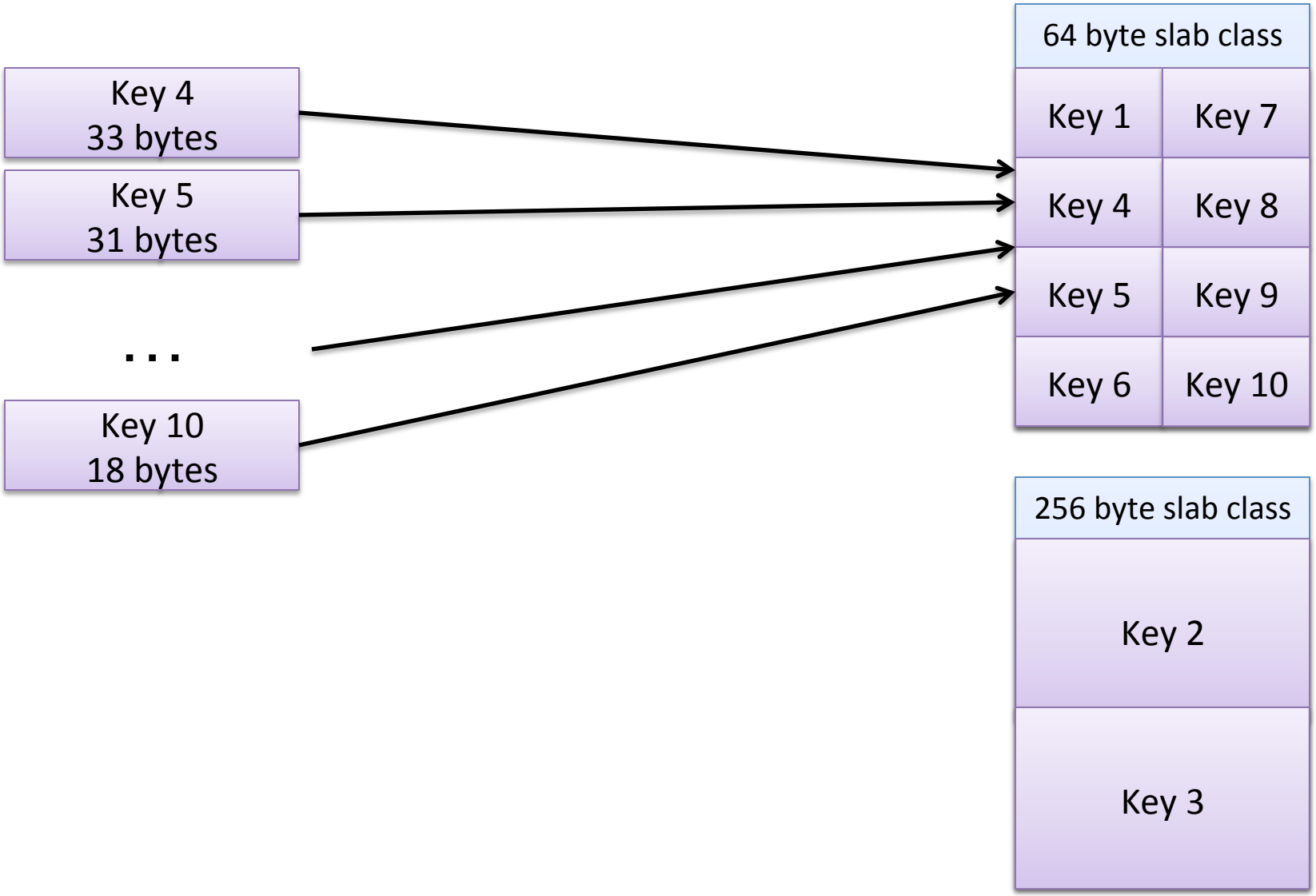
Memory Allocation in Memcached



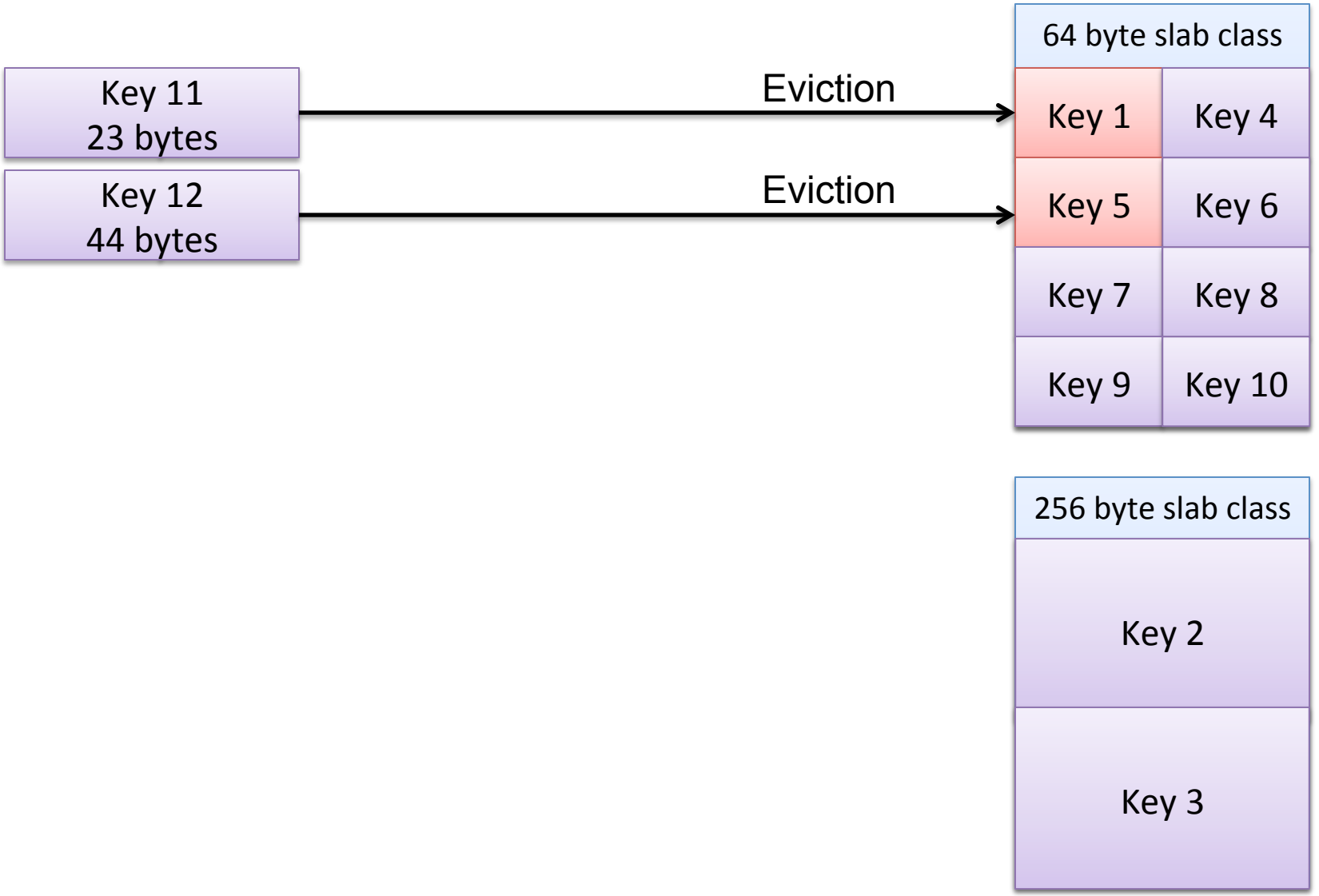
Memory Allocation in Memcached



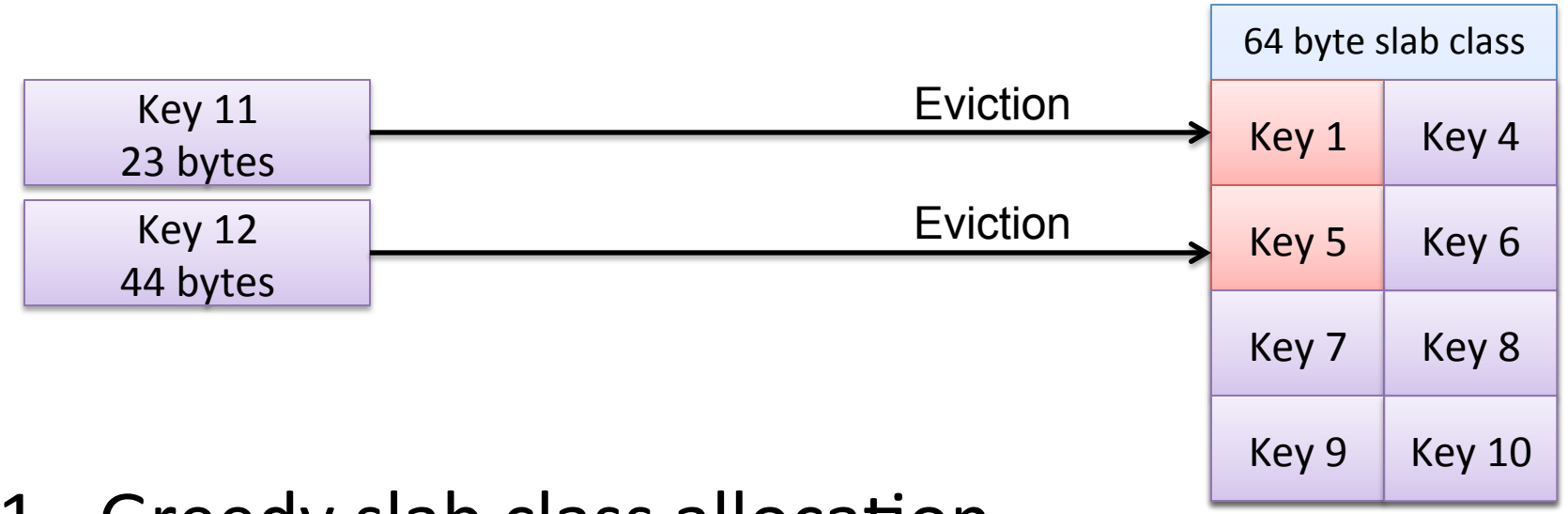
Memory Allocation in Memcached



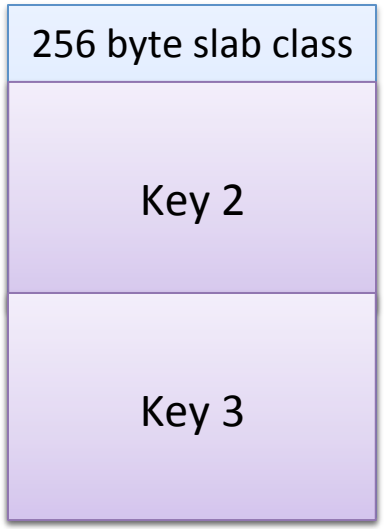
Separate Eviction Queue for Each Slab Class



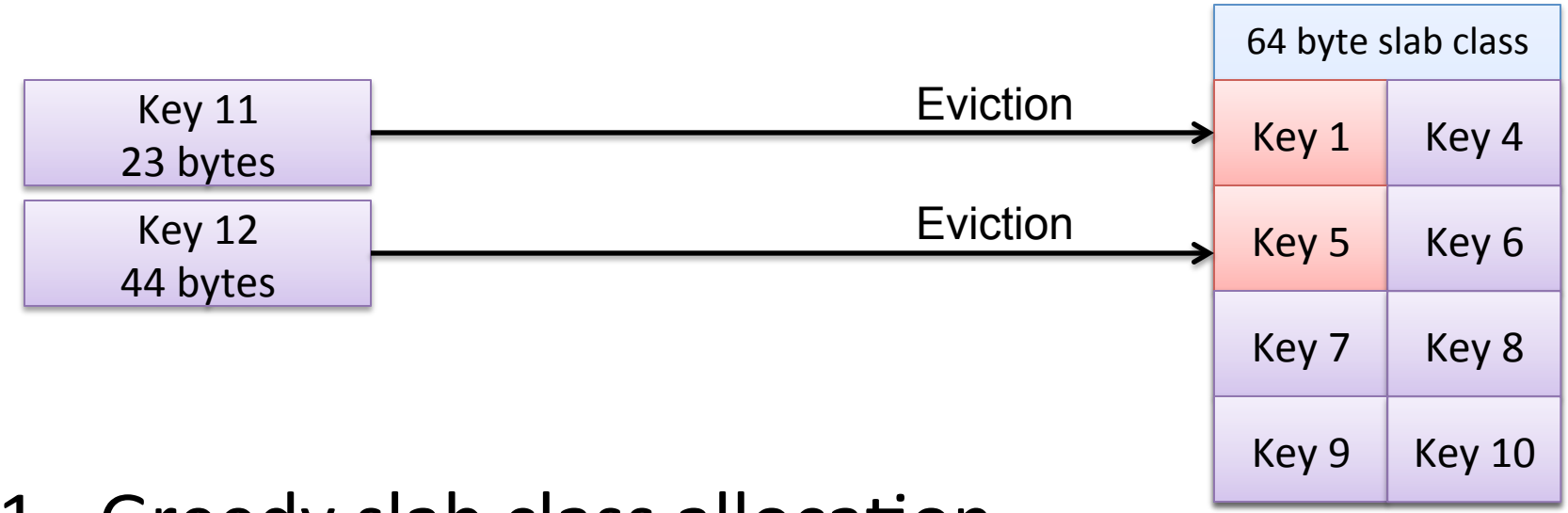
Problems with Memory Allocation



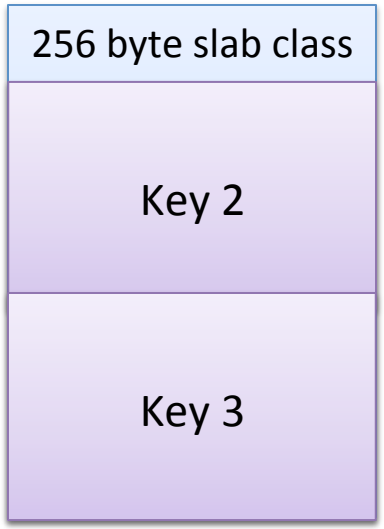
1. Greedy slab class allocation favors large slab classes



Problems with Memory Allocation



1. Greedy slab class allocation favors large slab classes
2. "Slab calcification" when request sizes change over time

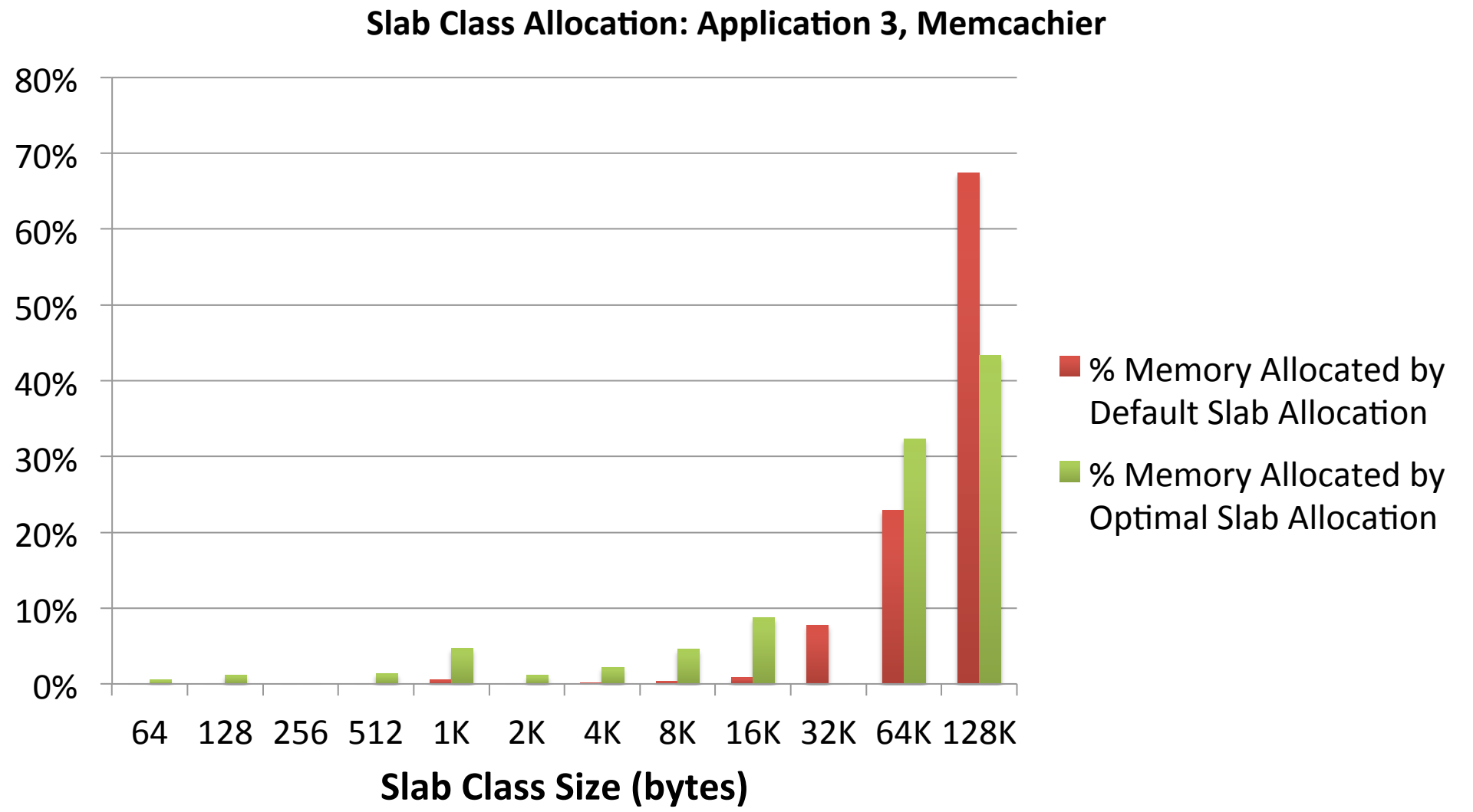


MemCachier Dataset

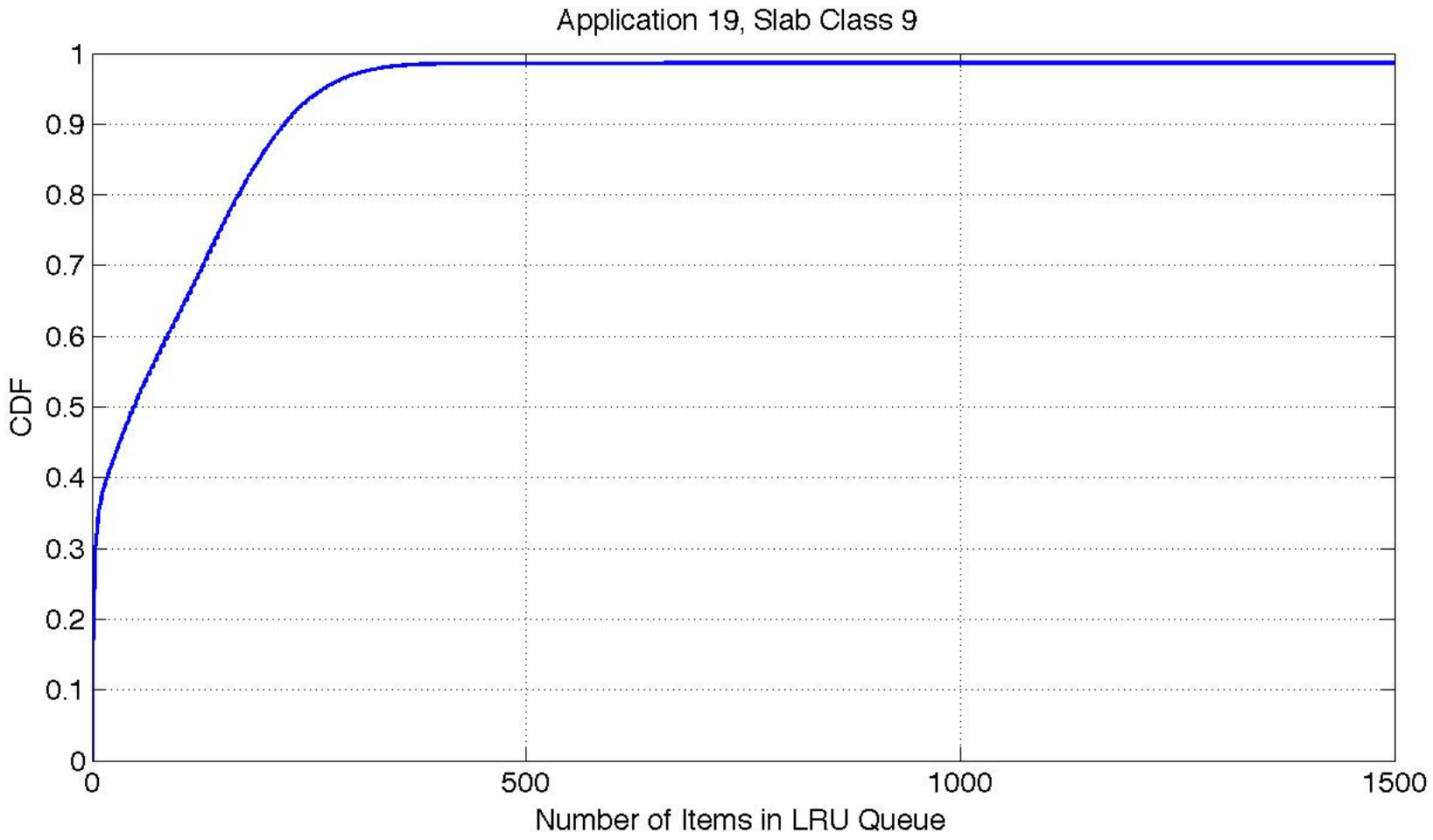
- Weeklong trace taken from MemCachier
  - 30 Memcached servers
  - 490 applications
  - Each application has its own pages
  - Each slab class has its own eviction queue



Application 3: Greedy Allocation Favors Large Slabs Classes



Hit-rate Curve Profiling



Dynamically Optimize Memory Allocation Using Hit-rate curves

$$\underset{m}{\text{maximize}} \quad \sum_{i=1}^{s} f_i h_i(m_i, e)$$

$$\text{subject to} \quad \sum_{i=1}^{s} m_i \leq M$$

ƒ – frequency of requests

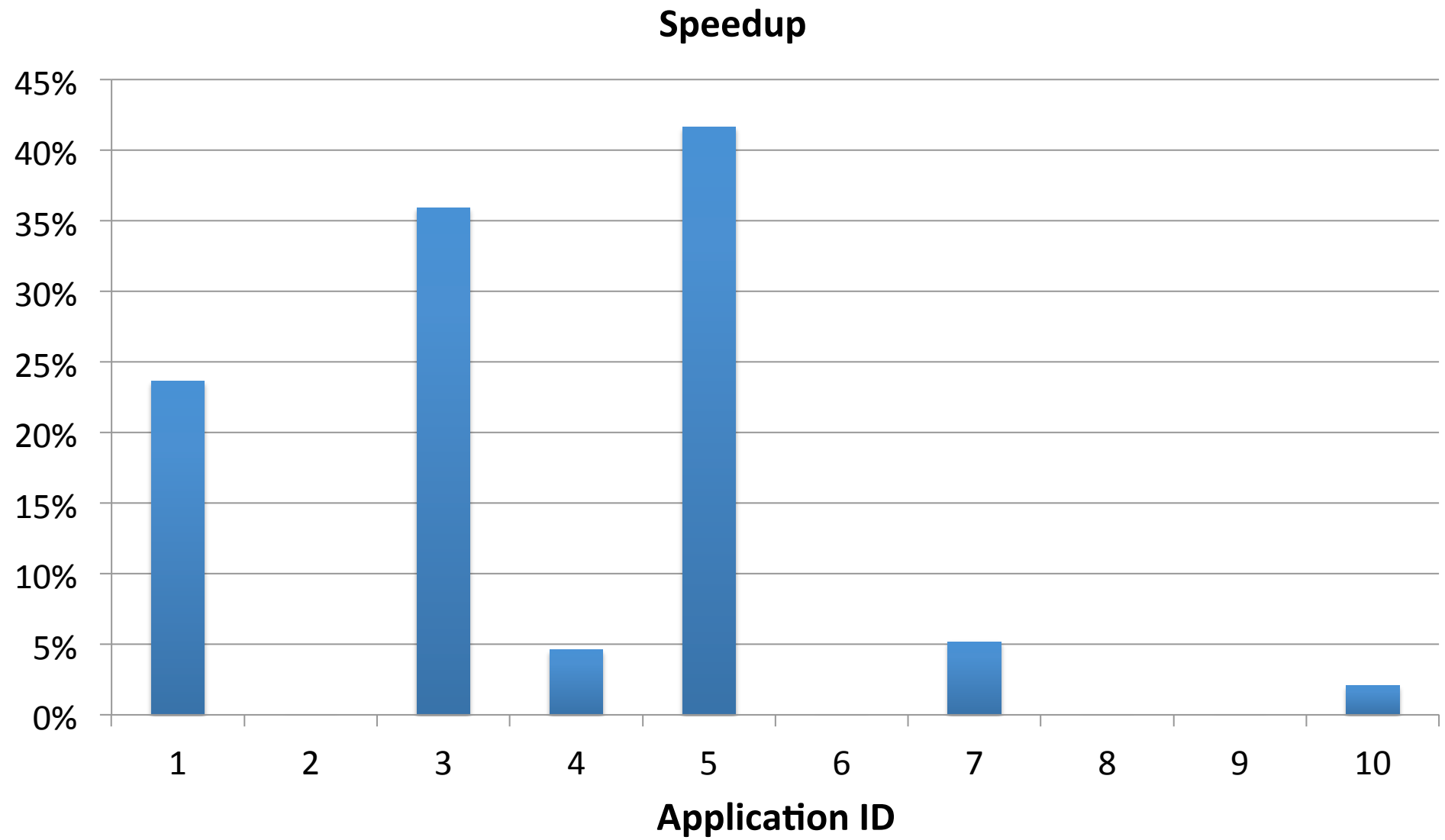
h – hit-rate of requests

m – memory allocated to slab class

e – eviction policy

M – memory allocated to application

Potential of Dynamic Memory Allocation in Memcachier



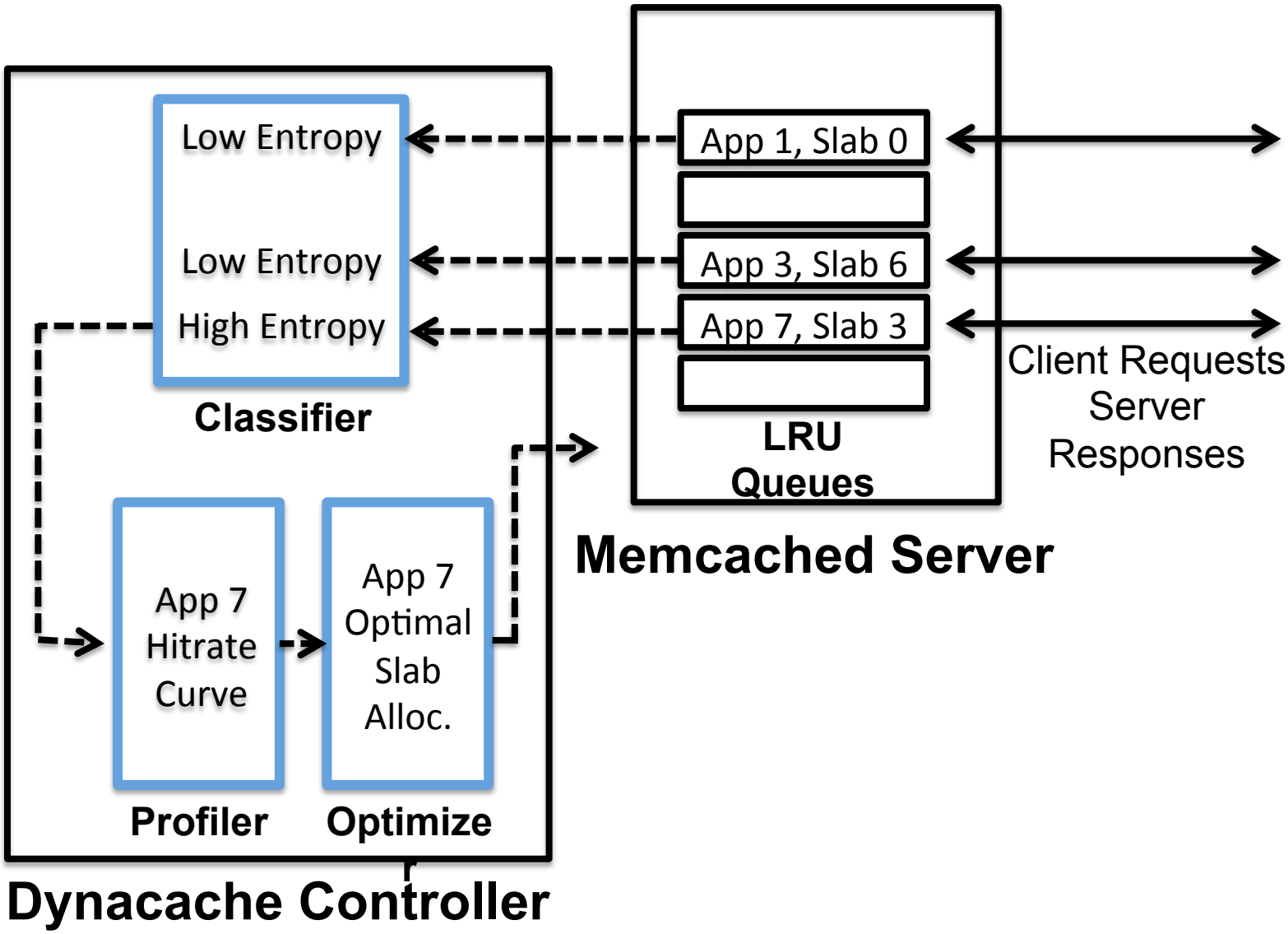
Related Work

- Optimizing memory allocation across applications based on hitrate curves
  - Mimir: Saemundsson et al [SOCC '14]
- Memcached client
  - McRouter: Likhtarov et al [Facebook blog '14]
- Rebalancing slabs to reduce slab calcification
  - Twitter: Rajashekhar et al [Twitter blog '12]
  - Facebook: Nishtala et al [NSDI '13]

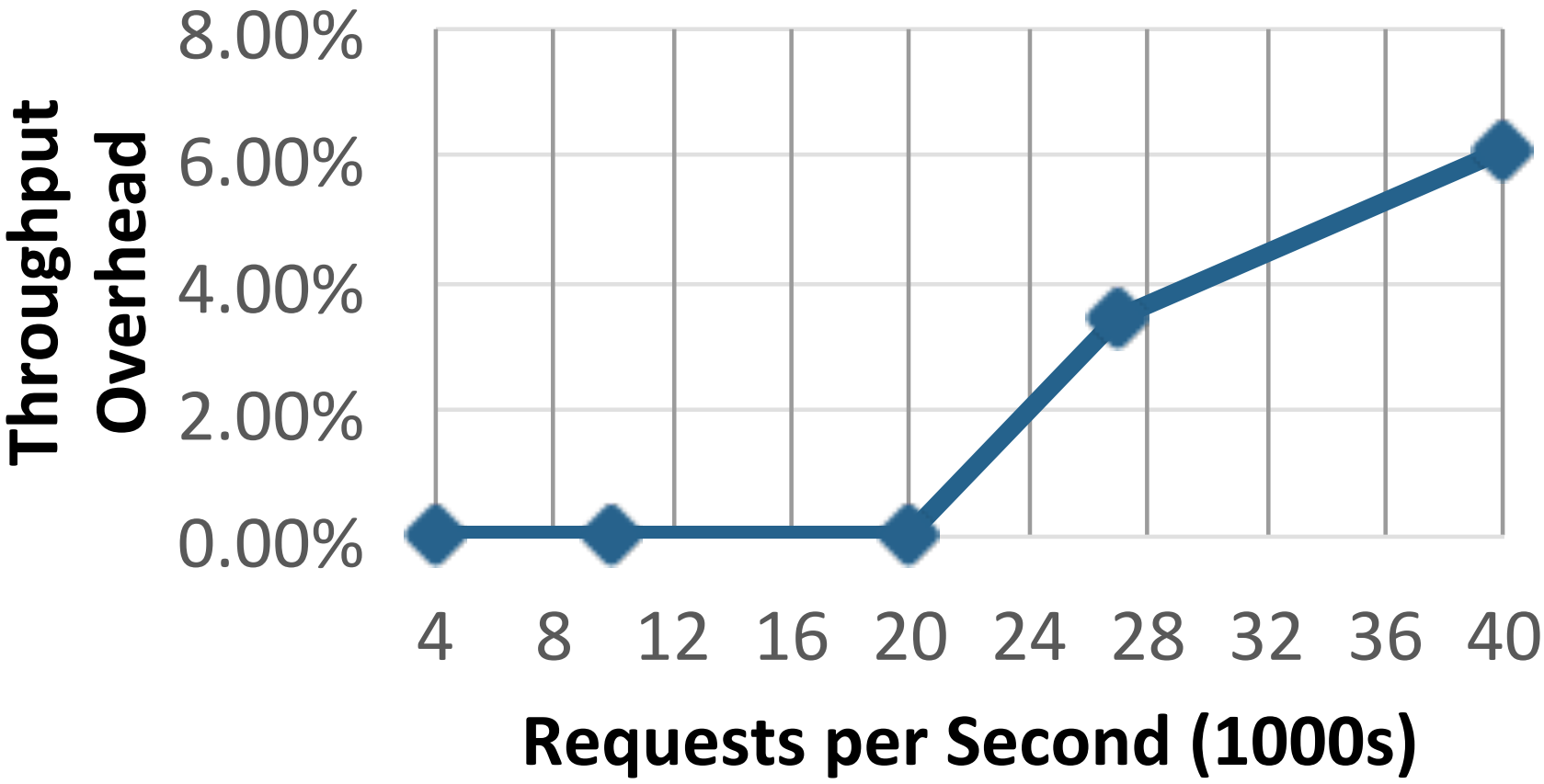
Summary

- Web applications heavily reliant on Memcached
- Memcached's policies are static and greedy
  - Resource allocation
  - Eviction policy
- Web applications are dynamic and variable
- Dynacache is dynamic cache controller that can dynamically adapt resource allocation and eviction policy

Dynacache Prototype Implementation



Evaluation


Throughput Overhead
8.00%
6.00%
4.00%
2.00%
0.00%
4
8
12
16
20
24
28
32
36
40
Requests per Second (1000s)