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

  twitter  Pinterest  airbnb  box

  facebook

• Facebook: 1000s servers, 5 billion requests per second
Memcached Hit-rate Impacts Performance

• +1% cache hit-rate $\rightarrow$ 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?

Memcached Server

Memcached Client

Memcached Server

Get(key)

Set(key, value)

Memcached Server
How Does Memcached Work?

Memcached Server

Get(key)
Set(key, value)

Memcached Client

Response(value)
ACK

Memcached Server

Memcached Server
### Memcached Server: Sounds Familiar?

<table>
<thead>
<tr>
<th></th>
<th>CPU Cache</th>
<th>Memcached Server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory Management</strong></td>
<td>Fixed-sized entries</td>
<td>Fixed-sized slabs</td>
</tr>
<tr>
<td></td>
<td>First Come First Serve</td>
<td>First Come First Serve</td>
</tr>
<tr>
<td><strong>Eviction Policy</strong></td>
<td>LRU</td>
<td>LRU</td>
</tr>
<tr>
<td><strong>Resource Allocation</strong></td>
<td>First Come First Serve</td>
<td>First Come First Serve</td>
</tr>
<tr>
<td>Across Applications</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Not the Same!

<table>
<thead>
<tr>
<th></th>
<th>CPU Cache</th>
<th>Memcached Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request size</td>
<td>Even</td>
<td>Highly variable</td>
</tr>
<tr>
<td>Available resources and complexity</td>
<td>Constrained</td>
<td>Unconstrained</td>
</tr>
<tr>
<td>Workload predictability</td>
<td>Predictable</td>
<td>Unpredictable</td>
</tr>
</tbody>
</table>
Dynacache: Dynamic Cache Controller

- Memcached Controller
  - Hit-rate curve estimation
  - Request distribution
- Memcached Server
- Memcached Server
- Memcached Server
- Memcached Client
Dynacache: Dynamic Cache Controller

- Memcached Controller
  - Change Memory Allocation
  - Change Eviction Policy

- Memcached Server
- Memcached Server
- Memcached Server

- Memcached Client
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

Key 1
35 bytes

Empty Page

Empty Page
Memory Allocation in Memcached

- Key 1: 35 bytes
- 64 bytes slab class
- Key 1
- Empty Page
Memory Allocation in Memcached

Key 2
202 bytes

Key 3
183 bytes

64 byte slab class
- Key 1

256 byte slab class
- Key 2
- Key 3
Memory Allocation in Memcached

- Key 4: 33 bytes
- Key 5: 31 bytes
- Key 10: 18 bytes

64 byte slab class:
- Key 1
- Key 7
- Key 4
- Key 8
- Key 5
- Key 9
- Key 6
- Key 10

256 byte slab class:
- Key 2
- Key 3

...
Separate Eviction Queue for Each Slab Class

Key 11
23 bytes

Key 12
44 bytes

64 byte slab class
Key 1  Key 4
Key 5  Key 6
Key 7  Key 8
Key 9  Key 10

256 byte slab class
Key 2
Key 3
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

Slab Class Allocation: Application 3, Memcached

- % Memory Allocated by Default Slab Allocation
- % Memory Allocated by Optimal Slab Allocation
Hit-rate Curve Profiling

Application 19, Slab Class 9

CDF

Number of Items in LRU Queue
Dynamically Optimize Memory Allocation Using Hit-rate curves

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{s} f_i h_i(m_i, e) \\
\text{subject to} & \quad \sum_{i=1}^{s} m_i \leq M
\end{align*}
\]

\[f\] – 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

![Bar chart showing speedup for various application IDs]
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: Rajashekhari 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

Dynacache Controller

Classifier

Low Entropy

Low Entropy

High Entropy

Profiler

Optimize

Memcached Server

LRU Queues

App 1, Slab 0

App 3, Slab 6

App 7, Slab 3

Client Requests

Server Responses

Dynacache Controller

App 7 Hitrate Curve

App 7 Optimal Slab Alloc.
Evaluation

Requests per Second (1000s)

Throughput

Overhead

0.00%  2.00%  4.00%  6.00%  8.00%  10.00%  12.00%  14.00%  16.00%  18.00%  20.00%  22.00%  24.00%  26.00%  28.00%  30.00%  32.00%  34.00%  36.00%  38.00%  40.00%