Fast Software Cache Design for Network Appliances

Dong Zhou, Huacheng Yu, Michael Kaminsky, David G. Andersen
Flow Caching in Open vSwitch

Microflow Cache
Exact Match
Single Hash Table
Flow Caching in Open vSwitch

- srcAddr=10.1.2.3, dstAddr=12.4.5.6, srcPort=15213, dstPort=80 → output: 1
- srcAddr=12.4.5.6, dstAddr=10.1.2.3, srcPort=80, dstPort=15213 → output: 2
- srcAddr=12.4.5.6, dstPort=13.1.2.3, srcPort=80, dstPort=15213 → drop

Microflow Cache
Exact Match
Single Hash Table
Flow Caching in Open vSwitch

- Megaflow Cache
  - Wildcard Match without Priority
  - Multiple Masked Tables

- Microflow Cache
  - Exact Match
  - Single Hash Table

Miss
Flow Caching in Open vSwitch

srcAddr=10.0.0.0/8, dstAddr=12.0.0.0/8, srcPort=*, dstPort=* → output: 1
srcAddr=12.0.0.0/8, dstAddr=10.0.0.0/8, srcPort=*, dstPort=* → output: 2
srcAddr=*, dstPort=13.0.0.0/8, srcPort=*, dstPort=* → drop

Megaflow Cache
Wildcard Match without Priority
Multiple Masked Tables

Miss

Microflow Cache
Exact Match
Single Hash Table
Flow Caching in Open vSwitch

- Packet Classifier
  - Multiple OpenFlow Tables

  Miss

- Megaflow Cache
  - Wildcard Match without Priority
  - Multiple Masked Tables

  Miss

- Microflow Cache
  - Exact Match
  - Single Hash Table
Flow Caching in Open vSwitch

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcAddr==10.0.0.0/8, dstAddr==12.0.0.0/8</td>
<td>output:1</td>
</tr>
<tr>
<td>srcAddr==12.0.0.0/8, dstAddr==10.0.0.0/8</td>
<td>output:2</td>
</tr>
</tbody>
</table>

Packet Classifier
Multiple OpenFlow Tables

Miss

Megaflow Cache
Wildcard Match without Priority
Multiple Masked Tables

Miss

Microflow Cache
Exact Match
Single Hash Table
Flow Caching in Open vSwitch

- Cache Hit Rate
- Lookup Latency

Packet Classifier
Multiple OpenFlow Tables

Megaflow Cache
Wildcard Match without Priority
Multiple Masked Tables

Microflow Cache
Exact Match
Single Hash Table

8x!
Basic Cache Design

- **Oversubscription factor** $\alpha = \# \text{keys} / \# \text{entries}$

- Assumption
  - Uniform workload
  - Random eviction
  - $\alpha = 0.95$

- 81% cache hit rate
Cache Design: 
Increase Set-Associativity

4-way set-associative bucket

8-way set-associative bucket

$81 \rightarrow 87\%$ cache hit rate
Cache Design:
More Candidate Buckets

Cuckoo hashing
81 → ~99% cache hit rate
Our Solution: Bounded Linear Probing (BLP)

\[ h(k) \]
\[ h(k') \]

4-way set-associative bucket

2,4 BLP

81 \( \rightarrow \) \(~94\%\) cache hit rate
# Qualitative Comparison

<table>
<thead>
<tr>
<th>Design</th>
<th>Lookup Speed (cache line reads)</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-way set-assoc.</td>
<td>1</td>
<td>~81%</td>
</tr>
<tr>
<td>8-way set-assoc.</td>
<td>1</td>
<td>~87%</td>
</tr>
<tr>
<td>2-4 cuckoo</td>
<td>2 random</td>
<td>~99%</td>
</tr>
<tr>
<td>2-4 BLP</td>
<td>1.5 consecutive</td>
<td>~94%</td>
</tr>
</tbody>
</table>
## Qualitative Comparison

<table>
<thead>
<tr>
<th>Design</th>
<th>Lookup Speed (cache line reads)</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-way set-assoc.</td>
<td>1</td>
<td>~81%</td>
</tr>
<tr>
<td>8-way set-assoc.</td>
<td>1</td>
<td>~87%</td>
</tr>
<tr>
<td>2-4 cuckoo</td>
<td>2 random</td>
<td>~99%</td>
</tr>
<tr>
<td>2-4 BLP</td>
<td>1.5 consecutive</td>
<td>~94%</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Occupancy = 0.71875

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Occupancy = 0.75

Why BLP is Better Than Set-Assoc.?
Qualitative Comparison

<table>
<thead>
<tr>
<th>Design</th>
<th>Lookup Speed (cache line reads)</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-way set-assoc.</td>
<td>1</td>
<td>~ 81%</td>
</tr>
<tr>
<td>8-way set-assoc.</td>
<td>1</td>
<td>~ 87%</td>
</tr>
<tr>
<td><strong>2-4 cuckoo</strong></td>
<td>2 random</td>
<td>~ 99%</td>
</tr>
<tr>
<td><strong>2-4 BLP</strong></td>
<td>1.5 consecutive</td>
<td>~ 94%</td>
</tr>
</tbody>
</table>
## Qualitative Comparison

<table>
<thead>
<tr>
<th>Design</th>
<th>Lookup Speed (cache line reads)</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-way set-assoc.</td>
<td>1</td>
<td>~ 81%</td>
</tr>
<tr>
<td>8-way set-assoc.</td>
<td>1</td>
<td>~ 87%</td>
</tr>
<tr>
<td><strong>2-4 cuckoo</strong></td>
<td><strong>2 random</strong></td>
<td>~ 99%</td>
</tr>
<tr>
<td><strong>2-4 BLP</strong></td>
<td>1.5 <em>consecutive</em></td>
<td>~ 94%</td>
</tr>
</tbody>
</table>
Better Cache Replacement

• Traditional LRU
  – High space overhead
  – CLOCK: 1 bit / key

• Our Solution: Probabilistic Bubble LRU (PBLRU)
PBLRU: Bubbling

Promotion

\[ h(D) \]
PBLRU: Bubbling

Eviction
PBLRU

• Basic bubbling
  – Combines both recency and frequency information

• Probabilistic bubbling
  – We only promote every $n$-th cache hit to reduce the number of memory writes

• Applying to 2-4 BLP
  – We choose a random bucket to apply bubbling
Evaluation

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2× Intel Xeon E5-2660v3 CPUs (2.60GHz)</td>
</tr>
<tr>
<td>DRAM</td>
<td>160 GiB DDR4 Memory</td>
</tr>
<tr>
<td>L3 Cache</td>
<td>2× 24 MiB</td>
</tr>
<tr>
<td>NIC</td>
<td>Intel X520 dual-port 10GbE</td>
</tr>
</tbody>
</table>
15% higher tput
Lookup Latency and Hit Rate

Cache hit rate improvement is not enough to compensate for its higher lookup latency.
Throughput (Skewed)

Skewed ($\theta=0.99$)

- 4-way
- 4-way w/ SIMD
- 8-way w/ SIMD
- 2-4 Cuckoo
- 2-4 BLP
- 2-4 BLP w/ PBLRU

7.5% higher tput
Lookup Latency and Hit Rate

Graph showing the relationship between Oversubscription Factor ($\alpha$) and Lookup Latency (Cycles) or Cache Hit Rate for different cache configurations: 4-way, 4-way w/ SIMD, 8-way w/ SIMD, 2-4 cuckoo-lite, 2-4 BLP, and 2-4 BLP w/ PBLRU.
Summary

- Bounded Linear Probing
- Probabilistic Bubble LRU
- Balance between Cache Hit Rate and Lookup Latency
Thank You!