Key-value store cluster with programmability in NICs and switches

Antoine Kaufmann, Naveen Kr. Sharma
Scaling Up a Key-Value Store

- Implementing load-balancing today:
  - Client chooses destination explicitly
  - Load balancer (server, or dedicated core)

- Problem:
  - Client based: dynamic load balancing is hard
  - Load balancer: does not scale with bandwidth

- Key Idea: Use flexible network hardware to steer packets
  - NICs: FlexNIC, Intel RRC, Netronome
  - Switches: RMTs, Intel Flexpipe
Packet Steering: Status Quo

- Problem:
  - Lock contention
  - Poor cache utilization
Scaling to Multiple Cores

- **Idea:** NIC steers packets to cores based on hash over key
- **Implementation:**
  - Place key in IPv6 src/dst address fields
  - Use receive-side scaling in NIC to steer packets to cores
  - NIC also provides RSS hash to Software, use for hash table lookup
Resulting Throughput

![Bar chart showing resulting throughput for Key-based and Flow-based methods. The x-axis represents the number of cores (1 to 5), and the y-axis represents throughput (Mops). The chart shows that as the number of cores increases, the throughput also increases for both Key-based and Flow-based methods.](image-url)
Scaling to Multiple Servers

- **Idea:** Switch steers packets to servers based on hash over key

- **Implementation:**
  - Again key in IPv6 src/dst address fields
  - Use Link Aggregation over server ports on switch

<table>
<thead>
<tr>
<th># Servers</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests/s</td>
<td>320,017</td>
<td>633,712</td>
<td>950,914</td>
</tr>
<tr>
<td>Scale-up</td>
<td>1x</td>
<td>1.98x</td>
<td>2.97x</td>
</tr>
</tbody>
</table>