WORKLOAD CHARACTERIZATION OF INTERACTIVE CLOUD SERVICES ON BIG AND SMALL SERVER PLATFORMS

Shuang Chen*, Shay Galon**, Christina Delimitrou*, Srilatha Manne**, and José Martínez*

*Cornell University
**Cavium Inc.
EXECUTIVE SUMMARY

- How to achieve low tail latency for interactive cloud services?
  - Tail latency more important and challenging
  - The entire stack from SW to HW is involved

- Understand how tail latency reacts to application and system changes
  - Quantify how current designs work
  - Get insights on future designs
**Low Latency**

- **Tail latency**
  - e.g., QoS defined as 99\textsuperscript{th} %ile in 1ms

\[0.99^5 = 0.95\]
The entire stack from SW to HW is involved

- Application bottleneck
- Different user cases
- Scalability

- Overhead of virtualization
- SW isolation mechanisms
- Overhead of context switching
- HW isolation mechanisms
- Hyperthreading
CATEGORIZE LC APPLICATIONS

- **By QoS Strictness**
  - us: memcached
  - ms: web server, in-memory database
  - s: persistent database

- **By Statefulness**
  - Stateful: memcached
  - Stateless: web server
SELECTED LC WORKLOADS

- **NGINX**
  - Web server
  - Stateless
  - 99th% in tens of ms

- **Memcached**
  - Key-value store
  - Stateful
  - 99th% in hundreds of us
**Server Architecture**

- **Intel Xeon E5-2699 v4**
  - 22 Cores...
  - 2 Threads/Core
  - L1 I/D: 32/32KB
  - L2: 256KB
  - LLC: 55MB, 20 ways
  - Memory: 128G DDR4
  - NIC: 10Gbps
  - 2.2GHz
  - 14nm
  - $4,115

- **Cavium ThunderX**
  - 48 Cores...
  - 1 Thread/Core
  - L1 I/D: 78/32KB
  - L2: 256KB
  - LLC: 16MB, 16 ways
  - Memory: 128G DDR4
  - NIC: 10Gbps
  - 1.8GHz
  - 28nm
  - $785
STUDIED PARAMETERS

- Application bottleneck
- Different user cases
- Scalability

- Overhead of virtualization
- SW isolation mechanisms
- Overhead of context switching
- HW isolation mechanisms
- Hyperthreading
INPUT LOAD

- **Xeon**
  - Latency (usec) vs. Percentage of Max RPS
  - AVG
  - 99th %ile

- **Memcached**
  - Latency (usec) vs. Percentage of Max RPS

- **ThunderX**
  - Latency (usec) vs. Percentage of Max RPS

- **NGINX**
  - Latency (msec) vs. Percentage of Max RPS

5.2x

5x
**Memcached Latency Decomposition**

- **Request:**
  - NIC
  - RX
  - TCP/IP
  - Epoll_wait
  - Libevent
  - Read
  - Memcache
  - Write
  - TCPIP
  - TX

- **Response:**
  - NIC

**Little user-space processing**

- **Xeon at 10% of max throughput:**
  - RX: 6
  - TX: 3
  - Mem: 1
  - Write: 1
  - Send: 5

- **ThunderX at 10% of max throughput:**
  - RX: 14
  - TX: 4
  - Mem: 5
  - Write: 9
  - Send: 24

**Network delay**

- 2x slower than Xeon

- **Xeon at 90% of max throughput:**
  - RX: 6
  - TX: 782
  - Mem: 1,009
  - Write: 315

- **ThunderX at 90% of max throughput:**
  - RX: 14
  - TX: 1,290
  - Mem: 1,650

**Queuing delay**

- 7
  - 20
  - 24

---

*Introduction • Characterization • Implications*
**STUDIED PARAMETERS**

- Application bottleneck
- **Different user cases**
- Scalability

- Overhead of virtualization
- SW isolation mechanisms
- Overhead of context switching
- HW isolation mechanisms
- Hyperthreading
MEMCACHED VALUE SIZE

- **Xeon**
  - Memory copy
  - Network processing and transmission
  - ThunderX is more sensitive

- **ThunderX**
**NUMBER OF MEMCACHED ITEMS**

- **Xeon**
  - Cache capacity
  - ThunderX is more sensitive

- **ThunderX**

![Graph showing latency (usec) vs. number of items for Xeon and ThunderX.](image-url)
STUDIED PARAMETERS

- Application bottleneck
- Different user cases
- **Scalability**

- Overhead of virtualization
- SW isolation mechanisms
- Overhead of context switching
- HW isolation mechanisms
- Hyperthreading
SCALABILITY

Memcached

- Interrupt handling
- Load imbalance
- Lock contention

NGINX
**STUDIED PARAMETERS**

- Application bottleneck
- Different user cases
- Scalability

- Overhead of virtualization
- SW isolation mechanisms
- **Overhead of context switching**
- HW isolation mechanisms
- Hyperthreading
**CONTEXT SWITCHING**

- Statically spawned threads VS dynamically allocated cores
- ThunderX is more sensitive

Memcached on Xeon

Memcached on ThunderX
STUDIED PARAMETERS

Application

• Application bottleneck
• Different user cases
• Scalability

Resource Manager

Virtualization

• Overhead of virtualization
• SW isolation mechanisms
• Overhead of context switching
• HW isolation mechanisms
• Hyperthreading

OS

Hardware
HYPERTHREADING

- **Reduce the overhead of context switching**
  - Allocate two threads on two hyperthreads

- **Make better use of execution units**
  - Co-locate different applications

<table>
<thead>
<tr>
<th></th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20%</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30%</td>
<td>MN</td>
<td>MN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40%</td>
<td>MN</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60%</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memcached & Nginx on the same hyperthreads

<table>
<thead>
<tr>
<th></th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>M</td>
</tr>
<tr>
<td>20%</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>30%</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>40%</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>50%</td>
<td>MN</td>
<td>MN</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60%</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70%</td>
<td>N</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memcached & Nginx on different hyperthreads
- Reduce network/queuing delays
- Optimize common user cases
- Improve elasticity
  - Lock alternatives
  - Load balance
- Reduce the overhead of virtualization
- Reduce context switching
- Make best use of SW isolation mechanisms
- Big VS Small Cores
- Make best use of HW features