Are Our OS’es Prepared for Edge Computing?

Pekka Enberg
Introduction (slide #1)

- **Thesis title:**
  - Operating System and Middleware for Internet-of-Things Devices

- **Student information:**
  - Pekka Enberg
  - PhD started at University of Helsinki on January 2017 (part-time)
  - Previous relevant work experience on Linux, KVM, and OSv

- **PhD advisor:**
  - Prof. Sasu Tarkoma

- **Research areas:**
  - Operating systems, edge computing, low-latency networking
What is the problem that you are actually going to solve in your work?
- Address latency bottlenecks in the OSes serving low-latency edge computing use cases like smart spaces in an energy efficient and secure way.

Why is it a problem?
- Extreme edge applications can benefit from lower limits on their tail latencies, and the proposed latency optimizations for current OSes are largely dependent on hardware designed for high end servers.

Positive, startling statement about your work that will address this problem.
- We need to throw away backwards compatibility and design systems that have applications, kernel, and hardware work together to achieve low-latency and energy efficiency in secure way.

What's the consequence of the startling statement?
- We can do more computation at the edge of the network and break the dependency to the centralized cloud.
What is Edge Computing?

- Edge computing is primarily about **low latency** networking
  - Low latency is hard over wide area networks, which makes it difficult to leverage cloud computing for latency sensitive applications like **augmented reality** and **self-driving vehicles**.
  - **Smart spaces** are another edge use case, which requires connectivity between low-power microcontrollers and the cloud.
- Edge computing also needs to be **secure** and **energy efficient**
  - Edge devices are deployed to network edges — not necessarily physical secure or managed by people.
  - Communications technology is forecast to consume, in worst case scenario, up to 50% of global electricity by 2030 [Andrae and Edler, 2015]
Background: Networking Stack Overheads

- The socket API and its typical kernel implementations have various overheads [Rizzo, 2012; Jeong et al., 2014; Marz and Zanden, 2016]:
  - **System calls**: Context switching and processor state pollution.
  - **Per-packet processing costs**: Dynamic memory allocation, heavy kernel data structures (e.g. “socket buffers”)
  - **Lack of connection locality**: Packet processing happening on different processor than where userspace thread is running.
  - **VFS abstraction**: Sockets are also file descriptors, which have various overheads.
  - **I/O multiplexing and event notification**: The epoll and kqueue interfaces report events but require system calls to retrieve event data.
Virtualization techniques are proposed for edge computing provide isolation and multitenancy, but they have a cost on latency and energy efficiency.

Hypervisors provide full isolation but have various overheads, which hurts low latency networking and energy efficiency.

- Hypervisor energy overhead can be as high as 59% to 273% for KVM compared to bare metal [Jin et al., 2012]

Containers have less overheads than hypervisors but provide less isolation because host kernel is shared by all the containers.

- Containers are bound by existing OS interfaces, which prevent optimizations that unikernels can do.
Examples of Edge Devices

<table>
<thead>
<tr>
<th></th>
<th>Raspberry Pi 3 B+</th>
<th>HPE GL20 IoT Gateway</th>
<th>HPE Edgeline EL1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>ARM Cortex A53</td>
<td>Intel Core i5-4300U</td>
<td>Intel Xeon D-1548</td>
</tr>
<tr>
<td></td>
<td>4 x 1.2 GHz</td>
<td>2 x 1.9 GHz</td>
<td>8 x 2.0 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>1 GB</td>
<td>8 GB</td>
<td>64 GB</td>
</tr>
<tr>
<td>Network</td>
<td>~1 GbE</td>
<td>2 x 1 GbE</td>
<td>2 x 10 GbE</td>
</tr>
<tr>
<td>Storage</td>
<td>MicroSD</td>
<td>SSD</td>
<td>NVMe SSD</td>
</tr>
<tr>
<td></td>
<td>32 GB</td>
<td>64 GB</td>
<td>4 TB</td>
</tr>
<tr>
<td>Price</td>
<td>50 EUR</td>
<td>500 - 1000 EUR</td>
<td>3000+ EUR</td>
</tr>
</tbody>
</table>

www.cs.helsinki.fi
Experimental Evaluation: Single-Board Microservers

Image source: https://commons.wikimedia.org/wiki/File:Raspberry-Pi-2-Bare-FL.jpg
What is the request/response latency and goodput on an single-board microserver for a network intensive application?

**Experimental setup:**
- Device: ASUS Tinker Board
- OS: Armbian OS
- System under test: Memcached memory cache system
- Load generator and monitor: Mutilate benchmarking tool
- Mutilate default key and value sizes (30 and 200 bytes, respectively)

**Factors:**
- Varying number of concurrent connections
- Network stack scaling enabled/disabled
- Interrupt-handling NIC not isolated/isolated
Experimental Evaluation: Memcached, 4 Threads, Steering

Concurrent Connections vs. Latency (ms) and Goodput (MB/s)

- p99 latency
- p95 latency
- p90 latency
- Goodput

Latency and Goodput measurements are plotted against the number of concurrent connections, illustrating the performance of Memcached with 4 threads and steering.
Experimental Evaluation: Memcached, 3 Threads, No Steering

![Graph showing latency and goodput over concurrent connections]

- p99 latency
- p95 latency
- p90 latency
- Goodput

Concurrent Connections vs. Latency (ms) and Goodput (MB/s)
Experimental Evaluation: Memcached, 3 Threads, No Steering

Latency reduces by factor of 2
Goodput reduces by factor of 1.2
What workloads and use cases will become important for edge computing?
- Augmented reality, autonomous vehicles, smart spaces, privacy-aware data platforms

What hardware capabilities will become important for edge computing?
- Wireless networking, multiqueue NICs, SR-IOV, GPUs

How will applications be deployed to edge devices?
- Light-weight VMs and unikernels, serverless computing
Summary

- Edge computing is about low latency networking, security, and energy efficiency.
  - Early empirical results indicate that revising OS design and interfaces has high potential to reduce latency on edge devices.

- Expected contributions:
  - Understand limitations of current OS design and interfaces that hinder low latency edge computing.
  - Propose OS design and interfaces that strikes a balance between security, low latency, and energy efficiency.
Thank you!
References