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

Answers to questions (slide #2)

- **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.

Background: Virtualization

- **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.
  - **Light-weight hypervisors** [Manco *et al.*, 2017] and **unikernels** [Madhavapeddy *et al.*, 2013] reduce hypervisor-based virtualization overheads.
  - 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

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

Experimental Evaluation: Single-Board Microservers

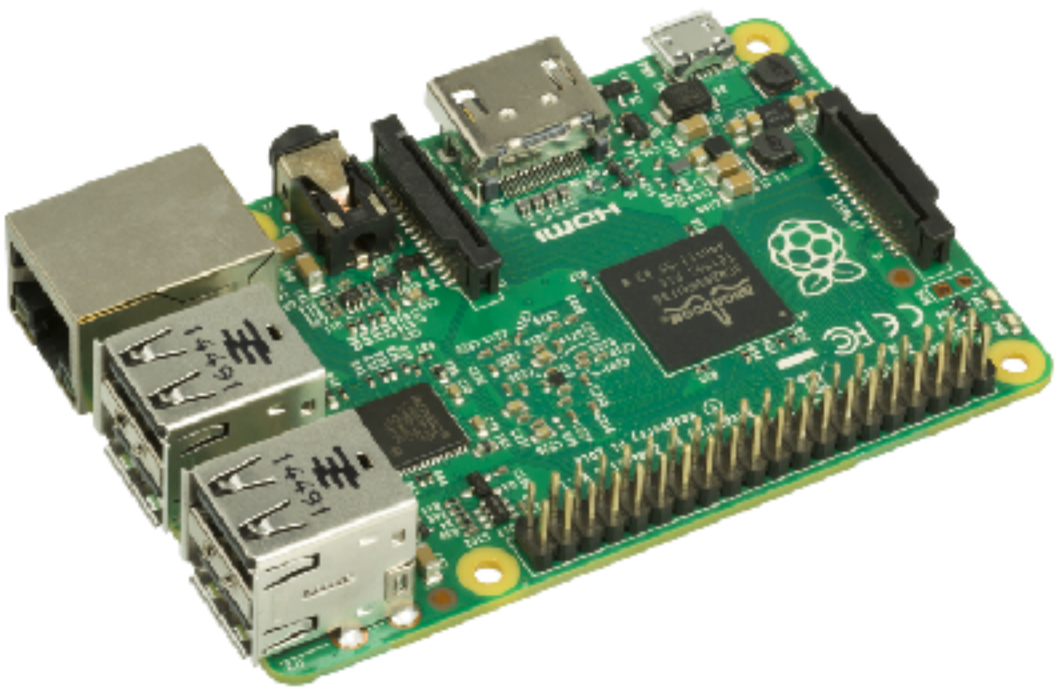
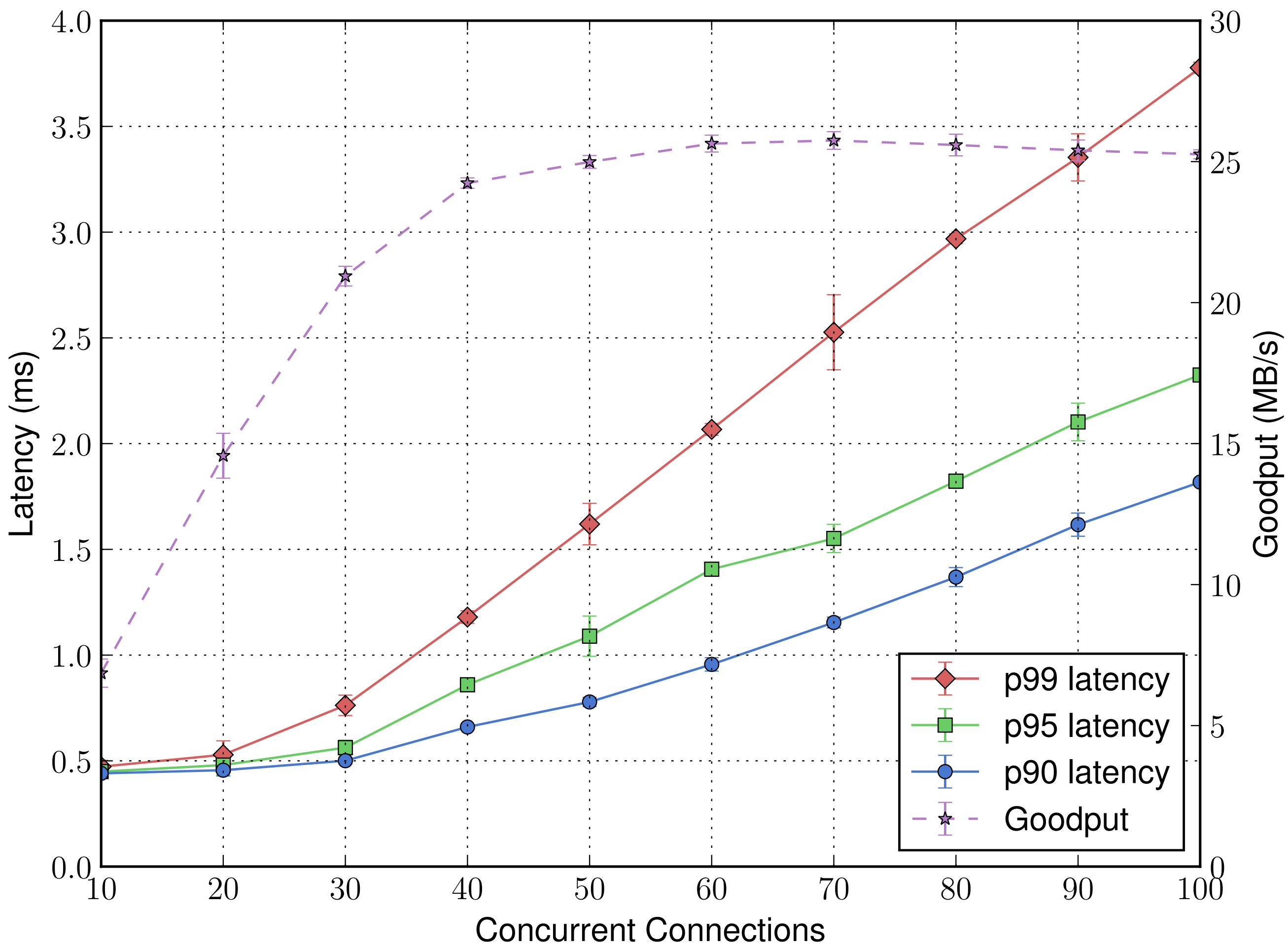


Image source: https://commons.wikimedia.org/wiki/File:Raspberry-Pi-2-Bare-FL.jpg

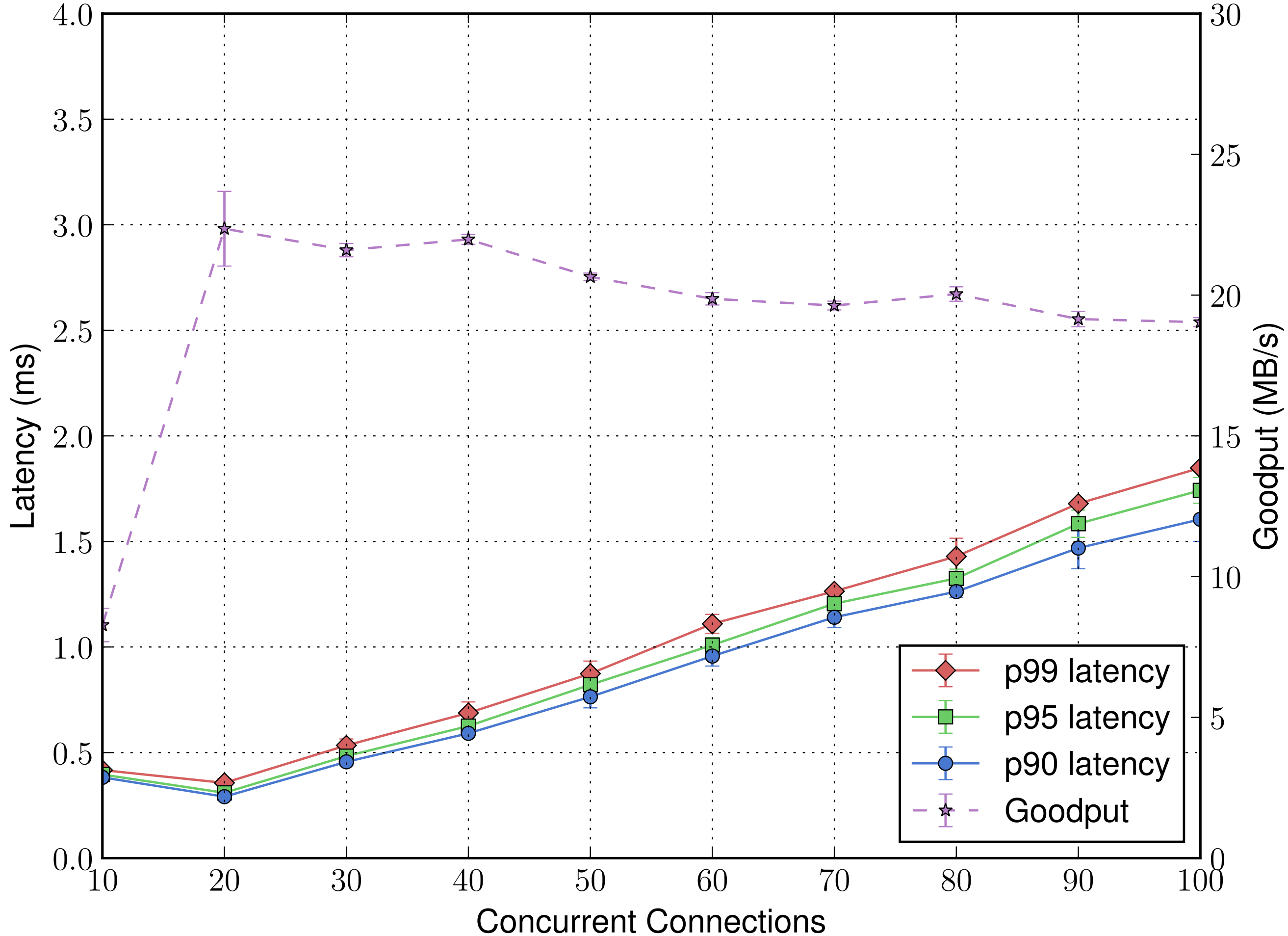
Experimental Evaluation: Setup and Factors

- 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

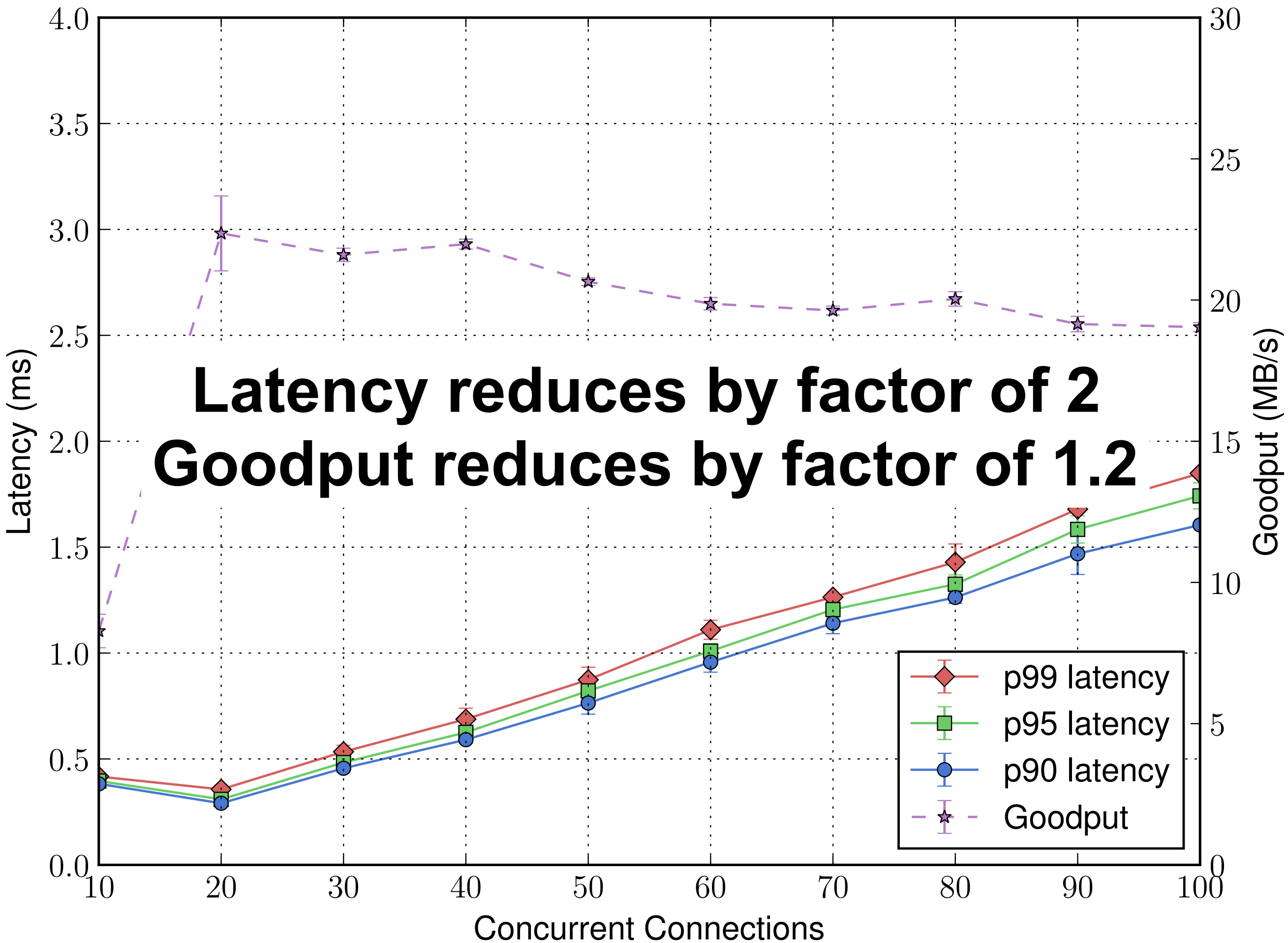
Experimental Evaluation: Memcached, 4 Threads, Steering



Experimental Evaluation: Memcached, 3 Threads, No Steering



Experimental Evaluation: Memcached, 3 Threads, No Steering



Discussion Topics

- **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

- Anders S.G. Andrae and Tomas Edler. On Global Electricity Usage of Communication Technology: Trends to 2030. *Challenges*, 2015.
- Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach, and Akhilesh Singhania. The multikernel: a new OS architecture for scalable multicore systems. In *SOSP*, 2009.
- Ricardo Koller and Dan Williams. Will Serverless End the Dominance of Linux in the Cloud?. In *HotOS*, 2017.
- Filipe Manco, Costin Lupu, Florian Schmidt, Jose Mendes, Simon Kuenzer, Sumit Sati, Kenichi Yasukata, Costin Raiciu, and Felipe Huici. My VM is Lighter (and Safer) than your Container. In SOSP '17, 2017.
- Stephen Marz and Brad Vander Zanden. Reducing Power Consumption and Latency in Mobile Devices Using an Event Stream Model. In *TECS*, 2016.
- Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. Unikernels: library operating systems for the cloud. In *ASPLOS*, 2013.
- Luigi Rizzo. Netmap: A Novel Framework for Fast Packet I/O. In *USENIX ATC*, 2012.
- Eun Young Jeong, Shinae Woo, Muhammad Jamshed, Haewon Jeong, Sunghwan Ihm, Dongsu Han, and KyoungSoo Park. mTCP: A Highly Scalable User-level TCP Stack for Multicore Systems. In *NSDI*, 2014.