Operating systems

- Naming
- Operating system structure and the kernel
- Processes, inter-process communication
- CPU scheduling
- Input/output
- Memory management
- Virtual memory and paging
- File systems
- Network stack design
- Virtual machines
A modern SoC
A typical rack-scale architecture?

Q. Message latency in this network?
A. \(~0.5\mu s\), or < 10 LLC misses.

⇒ need to think of this as one big machine
Is this a computer?

Still programmed today *mostly* as a classical distributed system
Conclusion: A computer is…

- A large system of heterogeneous cores and other devices
  - CPUs, GPUs, accelerators, network processors, …
- … which access distributed memory
  - Persistent, non-uniform, non-shared, etc.
- … which come and go dynamically
  - Hotplug, power management, partial and/or transient failure
- … which communicate via messages
  - Cache coherence protocols, etc. with significant latency
- … across a complex network
  - Multiple levels of routing, firewalling, address translation
- … and which show unpredictable diversity
- System configurations vary and change rapidly over time
Traditional OS structure
(Linux, Windows, iOS, Android, …)

Main memory: *global* data structures
The traditional OS model…

- Single, large, kernel
- Multithreaded, shared-memory program
- Drivers for most devices
- Surrounded by service daemons
- Written in C
- Optimized for common hardware
- In-kernel coded policies to maximize utilization

… is dead 😞.
So we built a new OS: Barrelfish

- Open-source OS written from scratch at ETHZ
- Support from Microsoft, HP, Huawei, UW, Cisco, Oracle, VMware...
- Goals:
  - Scale to many cores
  - Adapt to different hardware and networks
  - Handle complex, large main memories
  - Keep up with trends
The Barrelfish multikernel model

User space:
- App
- App
- Application
- Application

Operating System:
- OS node
  - state replica
- OS node
  - state replica
- OS node
  - state replica
- OS node
  - state replica

Arch-specific code:
- x86_64
  - CPU
- X86_64
  - CPU
- ARM
  - NIC
- GPU w/
  - CPU features

Async. msgs

Interconnect(s)
Current work: the complexity of modern hardware
A closer look at the OMAP4460

- 6+ heterogen. cores
- shared + private memory
- 5+ Inter-connects
- Devices on different buses
- Interrupt subsystem
There is no uniform view of the system from all cores
Writing correct software…

… means getting all this **right**
- C code is frequently wrong.
- Nice to generate this code, but from what?
- **Proving** software correct requires a **specification** of the hardware
  - But what would it look like?
  - What could be generated from it?
Representing address decoding

- OS code
- Page tables
- Memory allocation
- Proofs
- Runtime checking
- etc.
What kind of hardware do we really want?
What if we had…

- A hardware *research* platform for system software
  - Massively *overengineered* wrt. products
  - Highly *configurable* building block for rackscale and datacenter computing
- Perhaps we can *actually build it* at ETH…
  - Logical next platform for our research
  - Seed to other universities for impact

[ENZIAN]: A computer for systems research at ETHZ
Current prototype
Enzian board

- **ThunderX**
  - 4xDDR4
  - 128GB @ 2133

- **XCVU9P**
  - 4xDDR4
  - 512GB @ 2133
  - 64GB @ 2400

**Connectors and Bandwidths:**
- 4x40Gb/s
- 240Gb/s
- ~400Gb/s
- ~500Gb/s
- 4x100Gb/s
  - Or
  - 16x25Gb/s

**IO Shield**
- QSFP+
- QSFP+
- PCIe x8
- NVMe x4
- NVMe x4
- NVMe x4
- 4xSATA3
- FMC
- PCIe x16
- NVMe x4
- NVMe x4
- NVMe x4
- IO Shield

**Bandwidths:**
- 64Gb/s
- 96Gb/s
- 48Gb/s
- 64Gb/s
- 128Gb/s
- 48Gb/s
- 64Gb/s
- ~400Gb/s
- ~500Gb/s
- 4x100Gb/s
  - Or
  - 16x25Gb/s

**Enzian board Diagram**

---

*Timothy Roscoe, Systems Group*
Department of Computer Science
ETH Zurich