Turning down the LAMP
Software Specialisation for the Cloud

Anil Madhavapeddy
Motivation: Layers

Hardware

Processes

OS Kernel

Threads

Application

Hardware
Motivation: Layers

- Application
- Threads
- Language Runtime
- Processes
- OS Kernel
- Hardware

- Microsoft .net
- Ruby
- Java
- Python
Motivation: Layers

- Application
- Threads
- Language Runtime
- Processes
- OS Kernel
- Hypervisor
- Hardware
Motivation: Security

- Linux Kernel
  - Mar 1994: 176,250 LoC
  - May 2010: 13,320,934 LoC

Most core Internet services still written in C / C++
Approach: Reconstruct

• Most layers are in place for compatibility
  • **Xen**: to run operating systems
  • **Linux**: to run POSIX applications
  • **Processes**: to protect C applications

• If we start again, how much can things be improved?
Choose a new implementation language that:

- Has strong static typing
  - This improves performance (more work at compile time)
  - Reduces run-time bugs (memory safety)
- Has a simple run-time system
  - Essential for a low-level systems language
- Is extensible, e.g. for new methods of parallelization
Language: Objective Caml

• Developed since 1996 in INRIA, France.
• Based on the ML type-system: **type inference**, **static typing**
• Proven in **industry**:
  • Citrix XenServer (virtualization)
  • Jane Street Capital (finance)
  • Skydeck, MLState (web)
• **Extensible** type-system and grammar (FlowCaml, JoCaml, HashCaml)
DNS: Performance of BIND (C) vs Deens (ML)

![Graph showing performance comparison between BIND 9.2.3 and Deens.](chart.png)
DNS: with functional memoisation
MirageOS: Specialised application kernels

- Application Code
  - Threads
  - Language Runtime
  - User Processes
  - OS Kernel
  - Hypervisor
  - Hardware

- Application Code
  - Mirage Kernel
  - Hypervisor
  - Hardware
MirageOS: memory layout, concurrency

Memory
- 64-bit PV layout
- Single process
- Zero-copy I/O to Xen
- 4MB super page mappings

Concurrency
- Cooperative threading and events
- Fast inter-domain communication
- Works across cores and hosts
Language-integrated storage:

```ocaml
type t = { name: string; age: int }

let me = { name = "Anil"; age=31 }

let save () = t_save db me

let get () = t_get ~age:`Gt 30) db
```

**Advantage:** SQLite is fast and simple

**Downside:** interoperability. Object SCSI (Panassus) ?
Mirage: concurrency

Language-integrated concurrency:

```ocaml
let rec loop () =
  printf "hello!\n";
  lwt s = sleep 2.5 in
  loop ()

# val loop : unit -> Lwt.t unit = <fun>
```

**Advantage:** Blocking functions have a special type `Lwt.t`

**Downside:** Extra function call overhead
MirageOS: SQL performance vs PV Linux

![Graph showing SQL performance comparison between MirageOS Xen Guest and Mirage Linux Application for different record sizes.](image-url)
MirageOS: memory performance vs PV Linux
The Future: Multi-scale Operating System

- We produce **highly optimized kernels** from a **portable functional language** code base which can **adapt** to the local hardware.

- Same source code runs efficiently on:
  - **mobile phone** environment (e.g. using Cadmium or ARM)
  - **desktop OS** for development (e.g. using Eclipse IDE)
  - **cloud** for cheap scalability (using Xen kernel backend)
  - and soon GPGPU? FPGA? Intel SCC?
Applications

• **Dust Clouds**
  - Thousands of tiny virtual machines (~100k each)
  - Same price as a few conventional “large” virtual machines
  - Sprinkle them world-wide to run Tor anonymity nodes

• **Self-scaling Services**
  - As load spikes, request more resources dynamically from cloud
  - Detect resource imbalance and “migrate” globally on demand

• **All requires low-latency, high-reliability cloud APIs**
Observations

- Static address space layouts permit **multiple language runtimes** to run simultaneously in one VM container.
  - Alternative to Facebook compiling PHP to C++ using HipHop
- **Partial evaluation** has the potential save huge amounts of energy
  - Already used in systems, e.g. libc/arch/x86_64
- Thinking **multi-scale** instead of **multi-core** is important for OS and language design:
  - Newer multi-core look like multiple hosts in many ways (failure, coherency, communication latency).
Questions?

Open-source:

http://github.com/avsm/melange
http://github.com/avsm/mirage
http://github.com/mirage

Contact:

@ avsm2@cl.cam.ac.uk

avsm