Unikernels: Functional Library Operating Systems for the Cloud

Anil Madhavapeddy, University of Cambridge
with a merry crew:

Haris Rotsos, Balraj Singh, Steven Smith
Jon Crowcroft, Steve Hand
(University of Cambridge)
Richard Mortier (University of Nottingham)
Thomas Gazagnaire (OCamlPro)
Dave Scott (Citrix Systems R&D)

openmirage.org @avsm

Monday, 18 March 13
The Cloud Threat Model

- **Guest VM**
  - **Application Logic**
    - (native code, byte code)
  - **Runtime**
    - (OCaml, Java, Python, etc)

- **Control Domain**

- **Userspace Process**
  - **Frontend Device**
    - (net, block, usb, pci, framebuffer)

- **Kernel**
  - **Backend Device**
    - (net, block, usb, pci, framebuffer)

- **Hypervisor**

---

Monday, 18 March 13
The Cloud Threat Model

- **Frontend Device**
  - net, block, usb, pci, framebuffer

- **Runtime**
  - OCaml, Java, Python, etc

- **Application Logic**
  - native code, byte code

- **Backend Device**
  - net, block, usb, pci, framebuffer

- **Guest VM**

- **Userspace Process**

- **Kernel**

- **Hypervisor**

- **Internet**

- **SAN/NAS**

- **Tenants**

- **SDN**

---

openmirage.org

Monday, 18 March 13
Type-safety in the application layer defeats several external threats.
Type-safety in the kernel will make all external I/O safe, but at what cost?
Most attacks never come from within a guest, only from external traffic. So why all these privilege checks?
On the cloud, most appliances have a specialised purpose. E.g. a webserver, or a database VM.

The hypervisor gifts us a **stable hardware interface**. Cures the curse of library operating systems!
Key Design Insights

Userspace Process

Kernel

Hypervisor

Guest VM

Application Logic
(native code, byte code)

Runtime
(OCaml, Java, Python, etc)

Frontend Device
(net, block, usb, pci, framebuffer)

Backend Device
(net, block, usb, pci, framebuffer)

Control Domain

openmirage.org

Monday, 18 March 13
Unikernels!

Virtual machines are UNIX processes “done right” on the cloud

- Userspace Process
- Kernel
- Hypervisor
- Runtime (OCaml, Java, Python, etc)
- Application Logic (native code, byte code)
- Backend Device (net, block, usb, pci, framebuffer)

Monday, 18 March 13
Compiler has to stop at userspace. Every level has a different API, calling convention, and privilege requirements.
Virtual Appliances: specialised

Source Code → Object Files → Network Library → Device Library
Virtual Appliances: specialised

Source Code -> Object Files -> Network Library -> Device Library -> Boot Library
### Appliance Image Size

<table>
<thead>
<tr>
<th>Appliance</th>
<th>Standard Build</th>
<th>Dead Code Elimination</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>0.449 MB</td>
<td>0.184 MB</td>
</tr>
<tr>
<td>Web Server</td>
<td>0.674 MB</td>
<td>0.172 MB</td>
</tr>
<tr>
<td>Openflow learning switch</td>
<td>0.393 MB</td>
<td>0.164 MB</td>
</tr>
<tr>
<td>Openflow controller</td>
<td>0.392 MB</td>
<td>0.168 MB</td>
</tr>
</tbody>
</table>

All configuration and data compiled into the image by the toolchain.

Live migration is easy and fun :-)

---

OpenMirage.org
Unikernels are compact enough to boot and respond to network traffic in real-time.
Simplified Memory Management

Compiled native source code and runtime statically linked with random start offset

- Text and data
- Foreign grants
- Reserved by Xen
- OCaml minor heap
- OCaml major heap

128TB

64-bit virtual address space

120TB

4kB

4kB

8x512kB sectors

openmirage.org

Monday, 18 March 13
All I/O memory gets mapped into a reserved area and can be distinguished thusly.

- text data
- foreign grants
- reserved by Xen
- OCaml minor heap
- OCaml major heap

64-bit virtual address space:
- 128TB
- 120TB

8×512kB sectors

- IP header
- TCP header
- tx data
- rx data
- 4kB

openmirage.org

Monday, 18 March 13
Simplified Memory Management

OCaml heap is contiguous, with simpler write barriers as a result.

The diagram shows a breakdown of memory spaces:
- Text and data
- Foreign grants
- Reserved by Xen
- OCaml minor heap
- OCaml major heap

The diagram also indicates:
- 64-bit virtual address space
- 128TB
- 120TB
- 4kB
- 8x512kB sectors

openmirage.org
Zero-Copy IO Buffer Management

IO Page Pool

alloc

App

GET

HTTP

send

TCP/IP

HTTP

Ethernet

TCP/IP

HTTP

success

garbage collection

Grant Table

reference

request

Ring

openmirage.org

Monday, 18 March 13
let main () =
  lwt zones = read key "zones" "zone.db" in
  Net.Manager.bind (fun mgr dev →
    let src = 'any_addr, 53 in
    Dns.Server.listen dev src zones
  )
DNS Server Performance

- BIND9/UNIX
- NSD/UNIX
- Mirage/nomemo
- Mirage/memo
- NSD/-O0/Xen
- NSD/-O3/Xen

queries per second vs. Zone size (entries)

Zone size (entries): openmirage.org
DNS Server Performance

- BIND9/UNIX
- NSD/UNIX
- Mirage/nomemo
- Mirage/memo
- NSD/-O0/Xen
- NSD/-O3/Xen

Queries per second

Zone size (entries)

openmirage.org

Monday, 18 March 13
DNS Server Performance

- BIND9/UNIX
- NSD/UNIX
- Mirage/nomemo
- Mirage/memo
- NSD/-O0/Xen
- NSD/-O3/Xen

queries per second

<table>
<thead>
<tr>
<th>Zone size (entries)</th>
<th>BIND9/UNIX</th>
<th>NSD/UNIX</th>
<th>Mirage/nomemo</th>
<th>Mirage/memo</th>
<th>NSD/-O0/Xen</th>
<th>NSD/-O3/Xen</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>40000</td>
<td>40000</td>
<td>40000</td>
<td>40000</td>
<td>40000</td>
<td>40000</td>
</tr>
<tr>
<td>10000</td>
<td>60000</td>
<td>60000</td>
<td>60000</td>
<td>60000</td>
<td>60000</td>
<td>60000</td>
</tr>
</tbody>
</table>

openmirage.org

Monday, 18 March 13
DNS Server Performance

queries per second

Zone size (entries)

BIND9/UNIX
NSD/UNIX
Mirage/nomemo
Mirage/memo
NSD/-O0/Xen
NSD/-O3/Xen

100
10000

openmirage.org
DNS Server Performance

queries per second

Zone size (entries)

BIND9/UNIX
NSD/UNIX
Mirage/nomemo
Mirage/memo
NSD/-O0/Xen
NSD/-O3/Xen

Monday, 18 March 13
Summary

- **OCaml is the baseline language for all new code**
  - C runtime is small, and getting smaller.
  - Is fully event-driven and non-preemptive

- **Rewriting protocols wasn’t *that hard***
  - Not necessarily the best research strategy though.
  - But an extremely useful learning experience.
  - Tech transfer is vital.

- **Unikernels fit perfectly on the cloud**
  - Internet protocol building blocks
  - Seamless interop with legacy code through VMs

---

openmirage.org

Monday, 18 March 13
- Pure OCaml code at http://github.com/mirage/ for:
  - Device drivers (netfront/blkfront/xenstore)
  - TCP/IPv4 and DHCPv4
  - HTTP
  - DNS(SEC)
  - SSH
  - OpenFlow (controller/switch)
  - vchan IPC
  - 9P :-)
  - NFS
  - FAT32
  - Distributed k/v store: arakoon.org
Online resources:

- [http://www.openmirage.org](http://www.openmirage.org)
- (Code) [http://github.com/mirage](http://github.com/mirage)
- (O’Reilly Book) [http://realworldocaml.org](http://realworldocaml.org)

- [http://www.cl.cam.ac.uk/projects/ocamllabs](http://www.cl.cam.ac.uk/projects/ocamllabs)
  - funded by industry/academia (Jane Street Capital, Citrix, EU FP7, RCUK/Horizon)
  - Focus: real world functional programming with OCaml
  - Need compiler hackers, protocol heads, PL/type theory systems. Must enjoy open source work!
Reserve Slides

openmirage.org

Monday, 18 March 13
Key Research Questions Now

- **Interoperability --- with billions of VMs out there**
  - A unikernel per-language?
  - Interconnect strategies? Heap sharing?
  - Formal method integration easier or harder?

- **Coordination --- planetary scale computers**
  - Resources are highly elastic now.
  - How to coordinate a million microkernels?
  - “Warehouse Scale Computing”

- **Library Applications --- where are they?**
  - Irminsule, a git-like functional distributed database
  - Beanstalk, a self-scaling web server

Monday, 18 March 13
Optional VM Sealing

- **Single address-space and no dynamic loading**
  - $W^X$ address space
  - Address offsets are randomized at compile-time

- **Dropping page table privileges:**
  - Added *freeze* hypercall called just before app starts
  - Subsequent page table updates are rejected by Xen.
  - Exception for I/O mappings if they are non-exec and do not modify any existing mappings.

- Very easy in unikernels due to **focus on compile-time specialisation** instead of run-time complexity.
How Large is Large?

Lines of code (x 10^3)

- Linux 3.2.2
- glibc 2.15
- Bind 9.9.0
- httpd 2.4.2
- OpenSSH 6.0p1
- NOX-zaku
- Mirage

- OCaml
- C/C++
- ASM

openmirage.org

Monday, 18 March 13
Garbage collected heap management is more efficient in a single address-space environment. Thread latency can be reduced by eliminating multiple levels of scheduling.
Scaling via Parallel Instances

- linux-pv (1 host, 6 vcpus)
- linux-pv (2 hosts, 3 vcpus)
- linux-pv (6 hosts, 1 vcpu)
- xen-direct (6 unikernels)

- Apache/Linux vs. Mirage appliance
- Serving single static page

openmirage.org
Openflow controller performance

- Openflow controller is competitive with Nox (C++), but much more high-level. Applications can link directly against the switch to route their data.

openmirage.org

Monday, 18 March 13
Threads are heap allocated values, so benefit from the faster garbage collection cycle in the Mirage Xen version, and the scheduler can be overridden by application-specific needs.
Microbenchmarks: Block Storage

Throughput (MiB/s) vs. Block size (KiB) for different storage configurations:
- `xen-direct`
- `linux-pv, direct I/O`
- `linux-pv, buffered I/O`

See the graph for a visual representation of how throughput changes with block size for each configuration.

Openmirage.org

Monday, 18 March 13