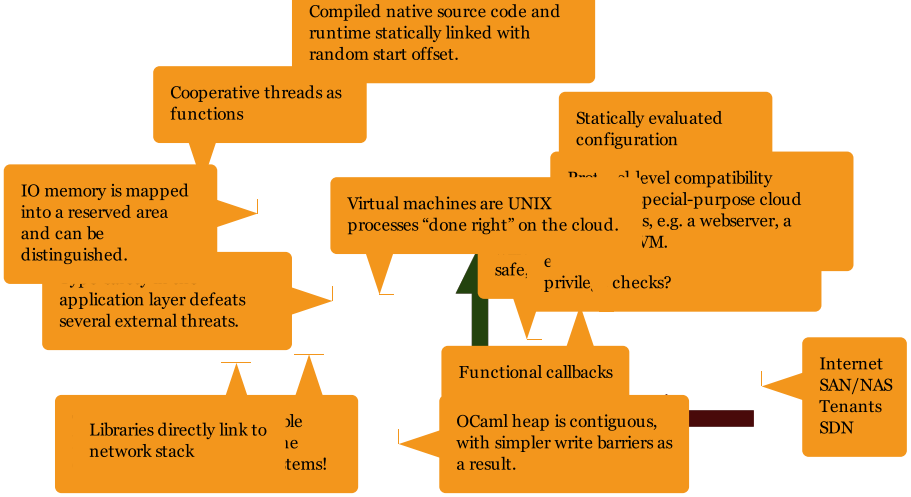
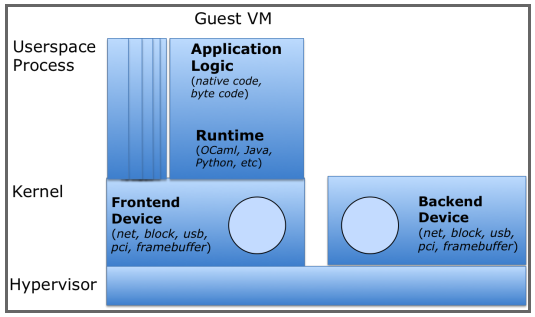


MIRAGE: EXTREME SPECIALISATION OF CLOUD APPLIANCES

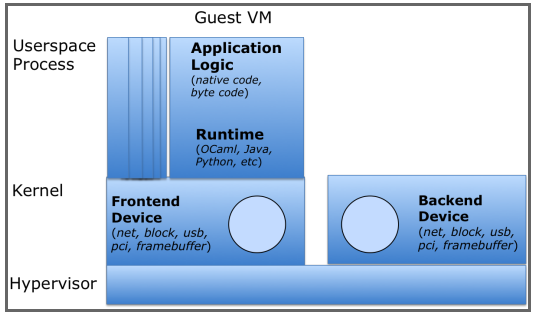
Anil Madhavapeddy University of Cambridge **@avsm**
Richard Mortier University of Nottingham **@mort___**
Dave Scott Citrix Systems R&D **@mugofsoup**

http://openmirage.org
http://decks.openmirage.org/oscon13/
http://www.youtube.com/watch?v=2Mx8Bd5JYyo

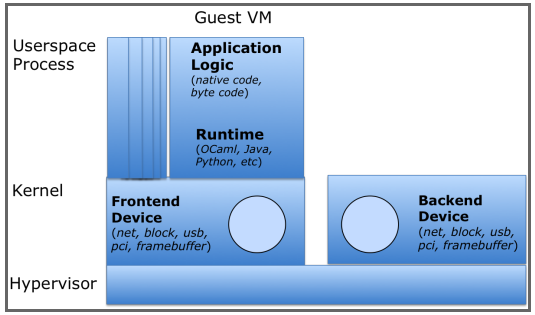
THE CLOUD THREAT MODEL



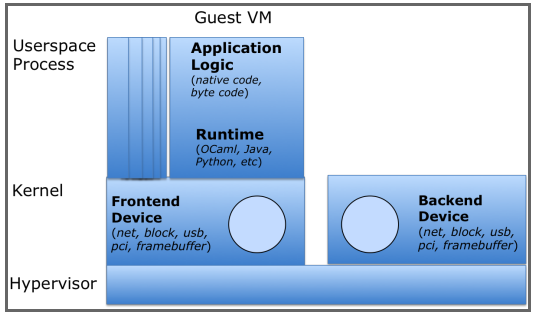
THE CLOUD THREAT MODEL



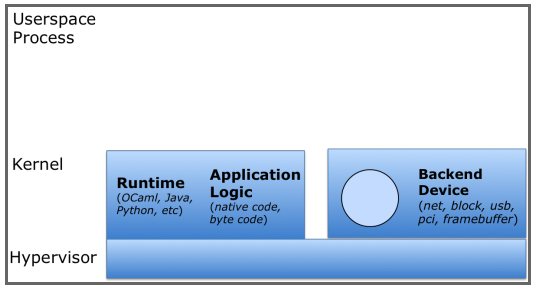
THE CLOUD THREAT MODEL



KEY DESIGN INSIGHTS



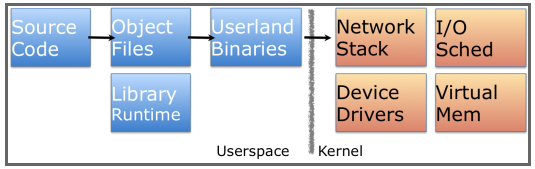
UNIKERNELS!


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

CONTRIBUTIONS

- The unikernel approach to building single-purpose appliances
  - Library OS + high level programming interface
  - Single-address space layout
- Evaluation of these techniques using a functional programming language (OCaml)
  - Benefits of type-safety need not damage performance
  - Static typing + modules = high level manipulation
- Language extensions for systems programming in OCaml

CURRENT VIRTUAL APPLIANCES

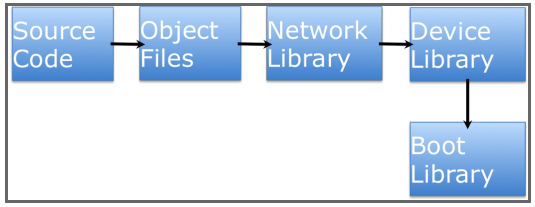


Compiler has to stop at userspace.

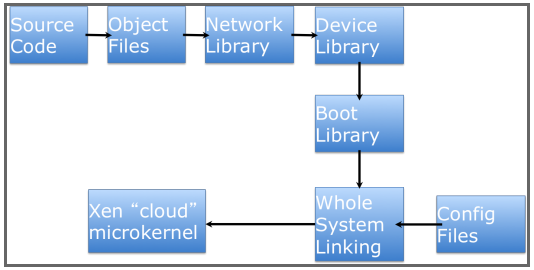
Every level has a different API, calling convention, and privilege requirements.

SPECIALISED VIRTUAL APPLIANCES

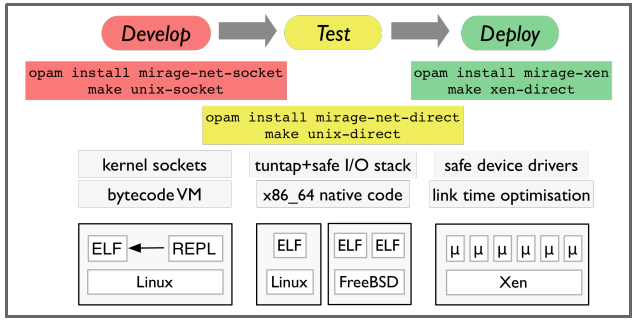
SPECIALISED VIRTUAL APPLIANCES



SPECIALISED VIRTUAL APPLIANCES



PROGRESSIVE SPECIALISATION



MICROBENCHMARKS!

Unikernel appliances are:

- Smaller, both in kLoC & image size
- Boot faster, *at packet round-trip time timescales*
- High performance
- Type-safe

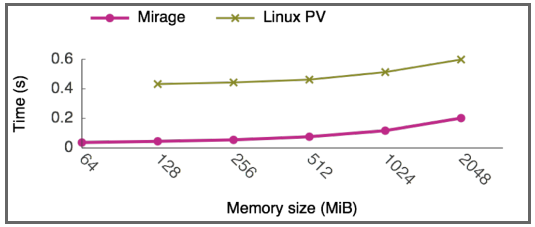
APPLIANCE IMAGE SIZE

Appliance	Standard Build	Dead Code Elimination
DNS	0.449 MB	0.184 MB
Web Server	0.674 MB	0.172 MB
Openflow learning switch	0.393 MB	0.164 MB
Openflow controller	0.392 MB	0.168 MB

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

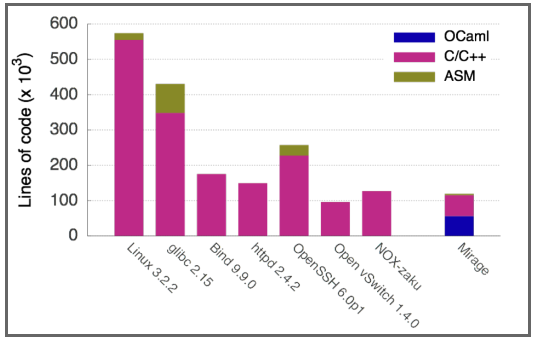
Live migration is easy and fun :-)

BOOT TIME

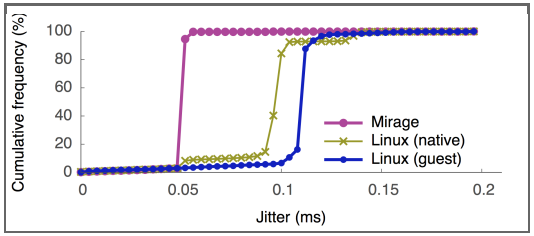


Unikernels are compact enough to boot and respond to network traffic in real-time.

HOW LARGE IS LARGE?

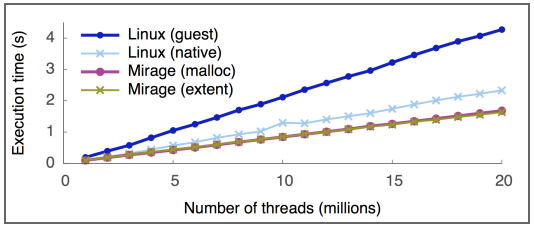


EVENT DRIVEN CO-THREADS



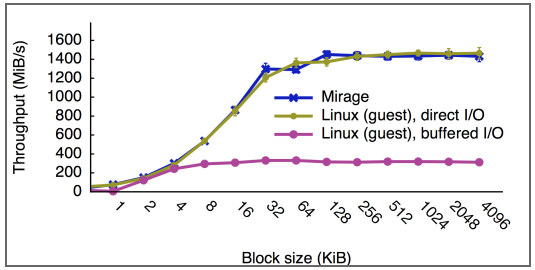
Garbage collected heap management is more efficient in a single address-space environment. Thread latency can be reduced by eliminating multiple levels of scheduling.

THREAD SCALING SINGLE INSTANCE



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.

BLOCK STORAGE



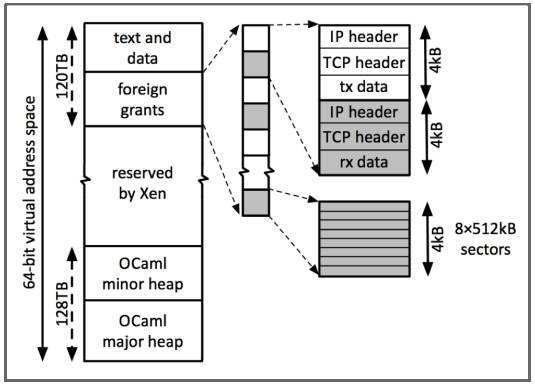
*Additionally, reading **from** a Mirage NAS-like storage VM achieves 942MiB/s buffered, and 1.8GiB/s unbuffered.*

TECHNIQUES

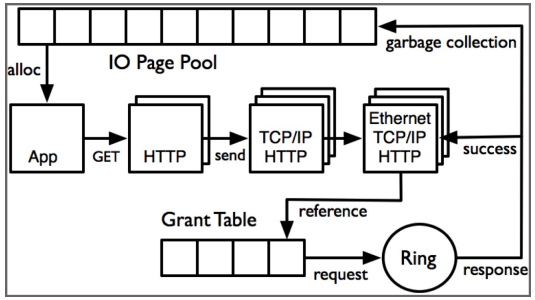
Several implementation techniques give rise to these benefits:

- Simplified memory management
- Zero-copy IO buffer management
- Hypervisor security extension for *VM sealing* (`W^X`)

MEMORY MANAGEMENT SPECIALISED!



BUFFER MANAGEMENT ZERO-COPY IO



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*

MACROBENCHMARKS

We have implemented several larger appliances.

We discuss *deens*, our DNS server in detail here.

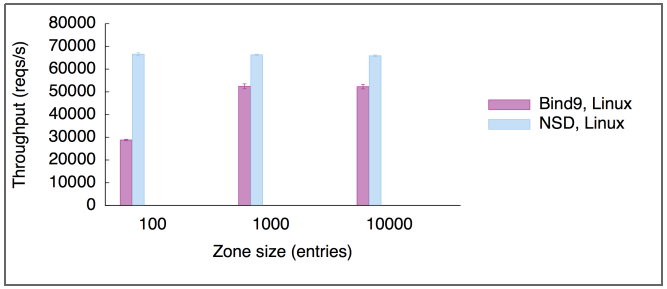
We also have:

- a simple webserver,
- an OpenFlow Switch, and
- an OpenFlow Controller.

DNS SERVER CODE

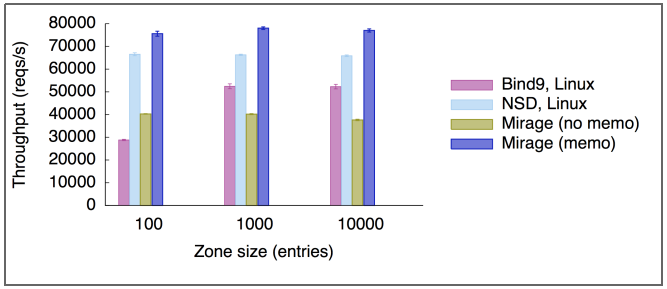
```
let main () =
  lwt zones = read key "zones" "zo
  Net.Manager.bind (fun mgr dev ->
    let src = `any_addr, 53 in
    Dns.Server.listen dev src zone
  )
```

DNS SERVER PERFORMANCE BASELINE



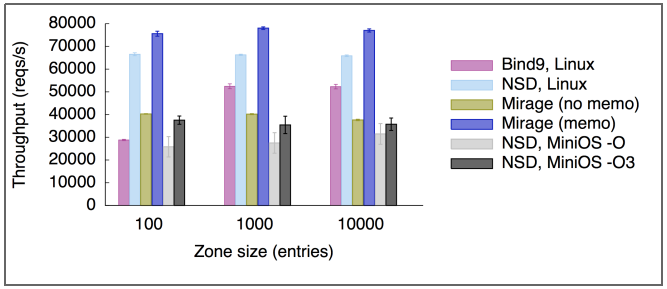
*Baseline figures, running **Bind** (standard) and **NSD** (high performance). NSD achieves around 70 kreqs/s.*

DNS SERVER PERFORMANCE MIRAGE



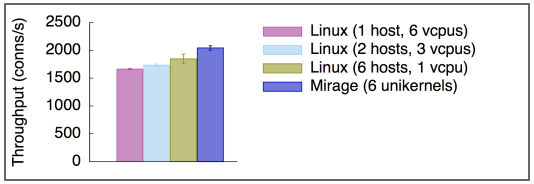
*Comparing against **Mirage** appliance, with and without simple memoisation. This **algorithmic** optimisation added just 6 lines of code.*

DNS SERVER PERFORMANCE C/MINIOS



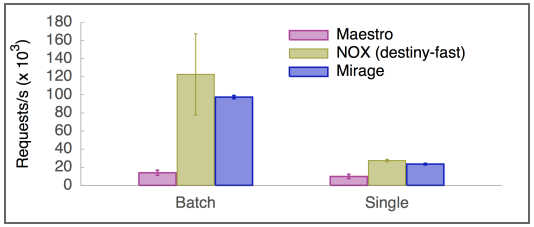
A rudimentary C-based appliance linking NSD directly against MiniOS. Poor user-space library performance vastly outweighs language effects.

SCALING VIA PARALLEL INSTANCES



Request throughput for serving a simple static page using Apache on Linux vs. a Mirage appliance. Rather than pay the cost of interlocking for thread-level parallelism, we scale by running many instances of the Mirage appliance.

OPENFLOW CONTROLLER



OpenFlow controller is competitive with NOX (C++), but much higher level. Applications can link directly against the switch to route their data.

SUMMARY

- **OCaml is the baseline language for all new code**
  - Our 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

PURE OCAML CODE GITHUB.COM/MIRAGE

- Device drivers
  - netfront
  - blkfront
  - xenstore
- Networking/Communication
  - IP/TCP/UDP/DHCP/DNS(SEC) (v4)
  - HTTP, SSH, OpenFlow (controller/switch)
  - vchan IPC
- Storage
  - NFS, FAT32
  - **http://arakoon.org** distributed k/v store
  - 9P :-)

MIRAGE ONLINE

- Website, **http://www.openmirage.org**
- Code, **http://github.com/mirage**
- O'Reilly OCaml book, **http://realworldocaml.org**

- OPAM package manager
  - Allows constraints to be applied to package installation
  - **Very** useful for managing assembly of the many small OCaml modules that construct a Mirage appliance
  - **http://opam.ocamlpro.com**

KEY RESEARCH DIRECTIONS

- **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

WE'RE HIRING! MUST ENJOY OPEN SOURCE!

- OCamlLabs, Cambridge, UK
  - **http://www.cl.cam.ac.uk/projects/ocamllabs**
  - Real world functional programming with OCaml
  - Need compiler hackers, protocol heads, PL/type theory systems

- Networks-as-a-Service, University of Nottingham, UK
  - **http://bit.ly/13sBjjC**
    (2 posts, 3 years, deadline **Friday August 2nd**)
  - Joint with University of Cambridge & Imperial College
  - Network virtualisation for millions of microkernel apps
  - Need Mirage hackers, network protocol experts, graph theorists

HTTP://TWITTER.COM/MORT___

Thanks! Questions?

(…and please rate the talk!)