xen.org

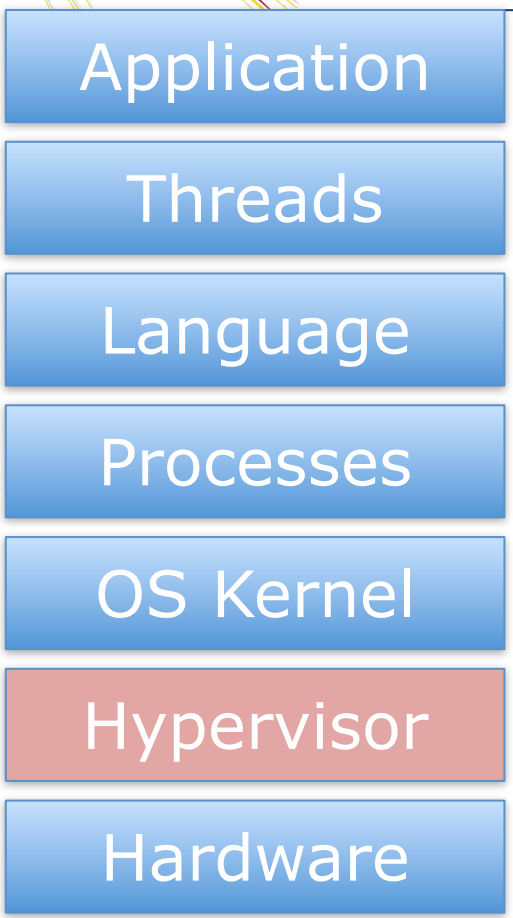
Mirage: OCaml Appliances for Xen Clouds

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

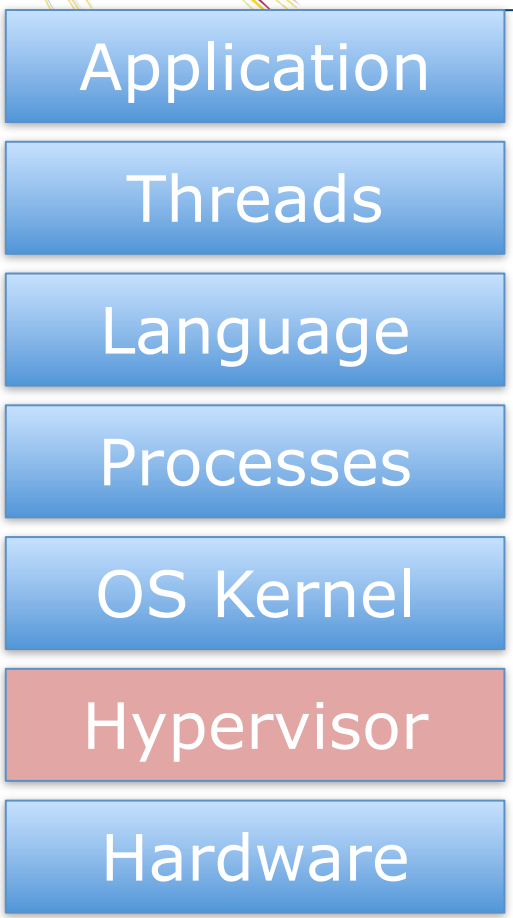
@avsm

Modern Stacks are Too Large

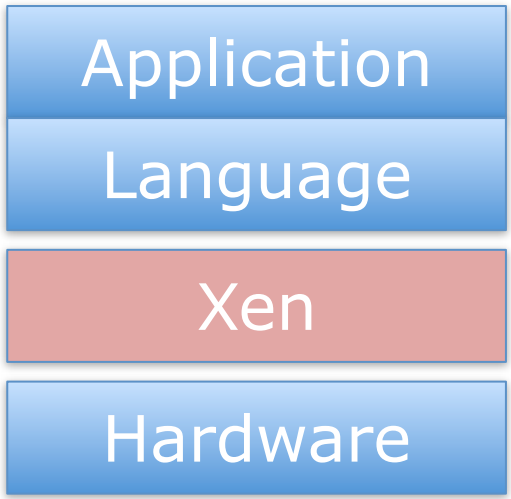


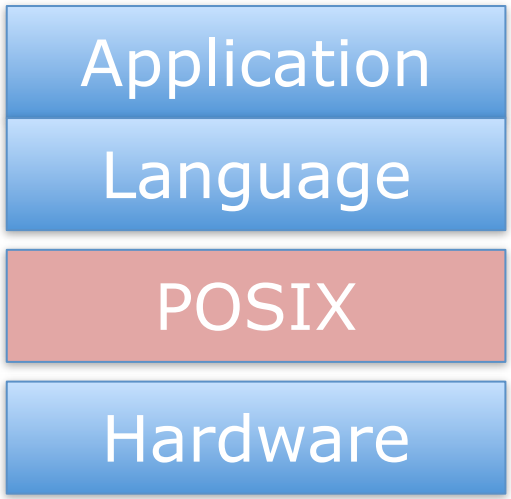
- Millions of lines of code & configuration
- Why build for clouds as for desktops?
- We can simplify!

Modern Stacks are Too Large

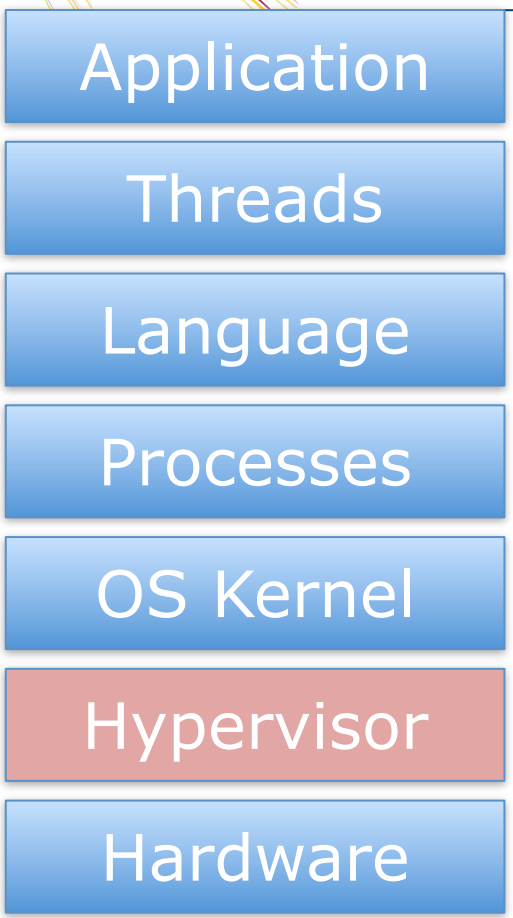


- Millions of lines of code & configuration
- Why build for clouds as for desktops?
- We can simplify!

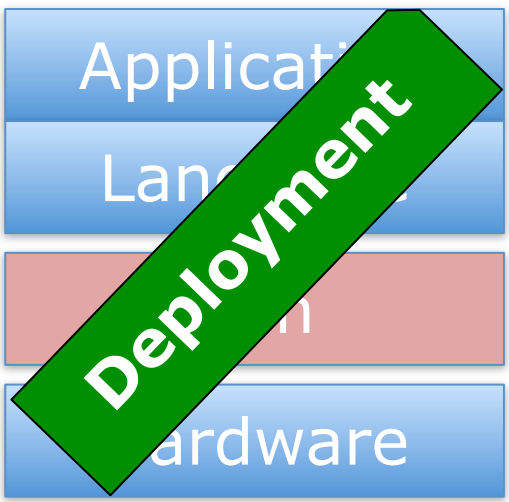


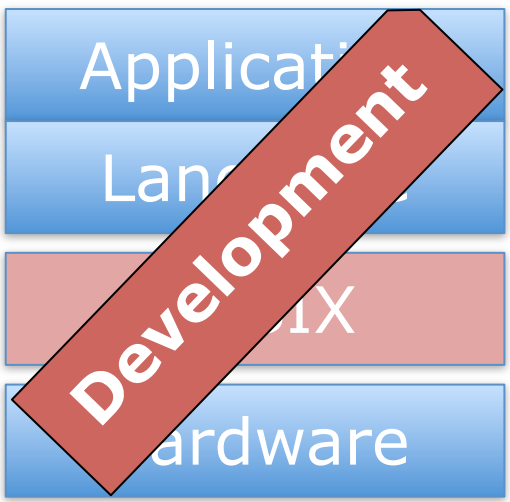


Modern Stacks are Too Large

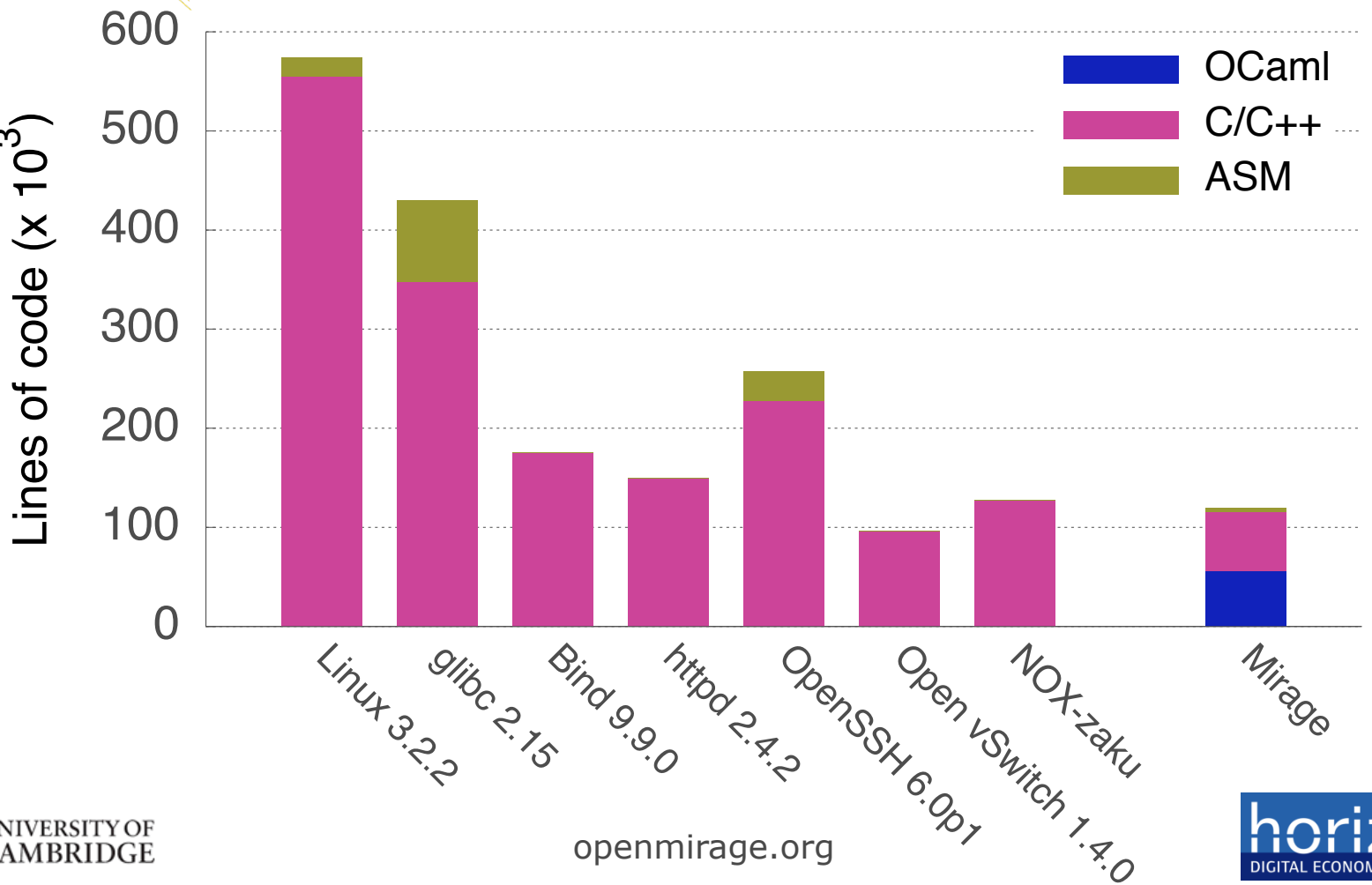


- Millions of lines of code & configuration
- Why build for clouds as for desktops?
- We can simplify!





How Large is Large?



UNIVERSITY OF CAMBRIDGE

horizon DIGITAL ECONOMY RESEARCH

Why Do We Care?

- Critical memory bugs still occur *regularly in mature software*

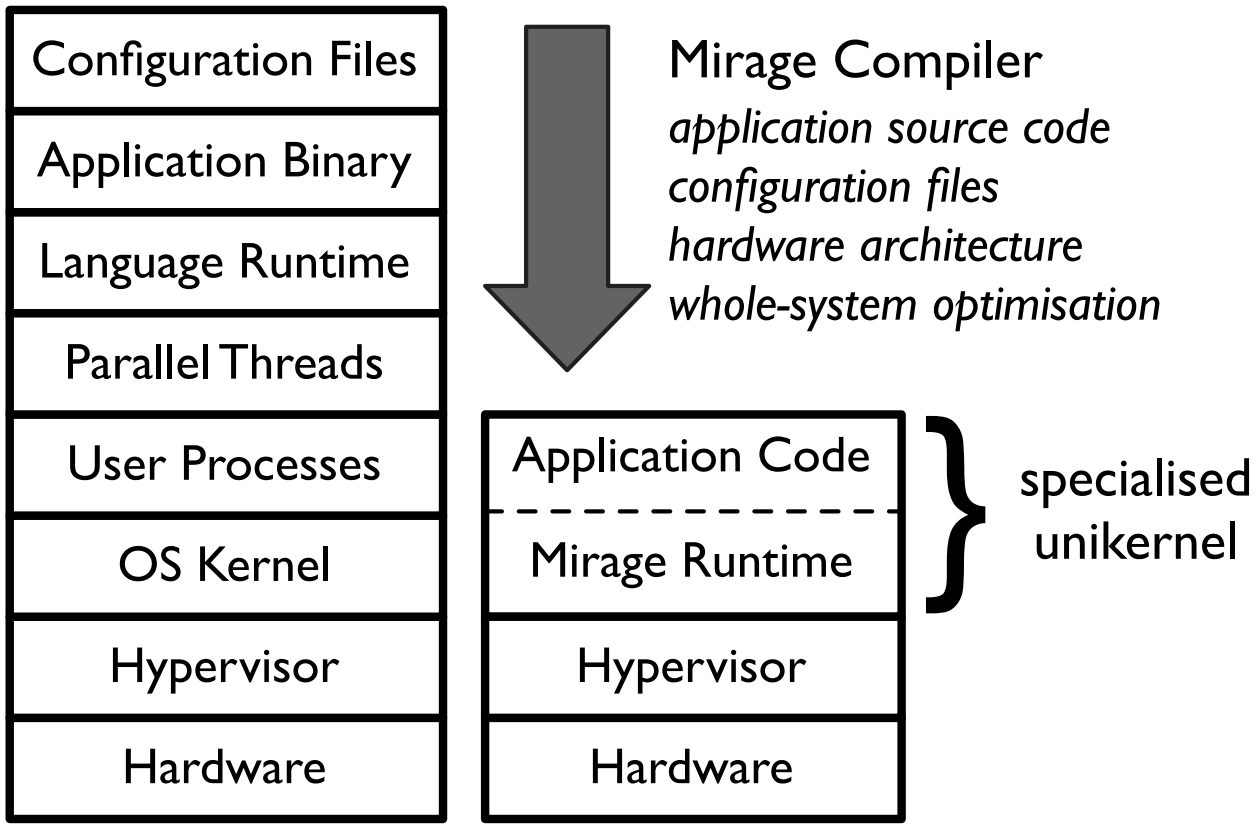
- **CVE-2012-1182 – Samba**
  - RPC code generator overflow
  - Variable containing buffer length checked independently of variable used to allocate memory for buffer
  - *Leads to root exploit*

- **CVE-2012-2110 – OpenSSL**
  - Integer conversion bug mixed with realloc wrappers
  - Unsigned cast to signed, but realloc'd buffer not clamped
  - *Leads to heap corruption*

The Cloud Threat Model

- **Attack from without, not within**: trust hypervisor, but not I/O traffic or other VMs.
- VMs are no longer multi-user, but primarily **single-purpose deployments**
- …and they are in a ***multi-tenant datacenter***
- …and they are ***always network connected***

Sealed Appliances



Key Features of Mirage

- **Static typing**
  - Pragmatic functional language (OCaml)
  - Large set of libraries provided, some used in XCP already

```
Net.Manager.bind
 (fun mgr dev →
   let src = 'any_addr, 53 in
   Dns.Server.listen dev src zones
 )
```

Key Features of Mirage

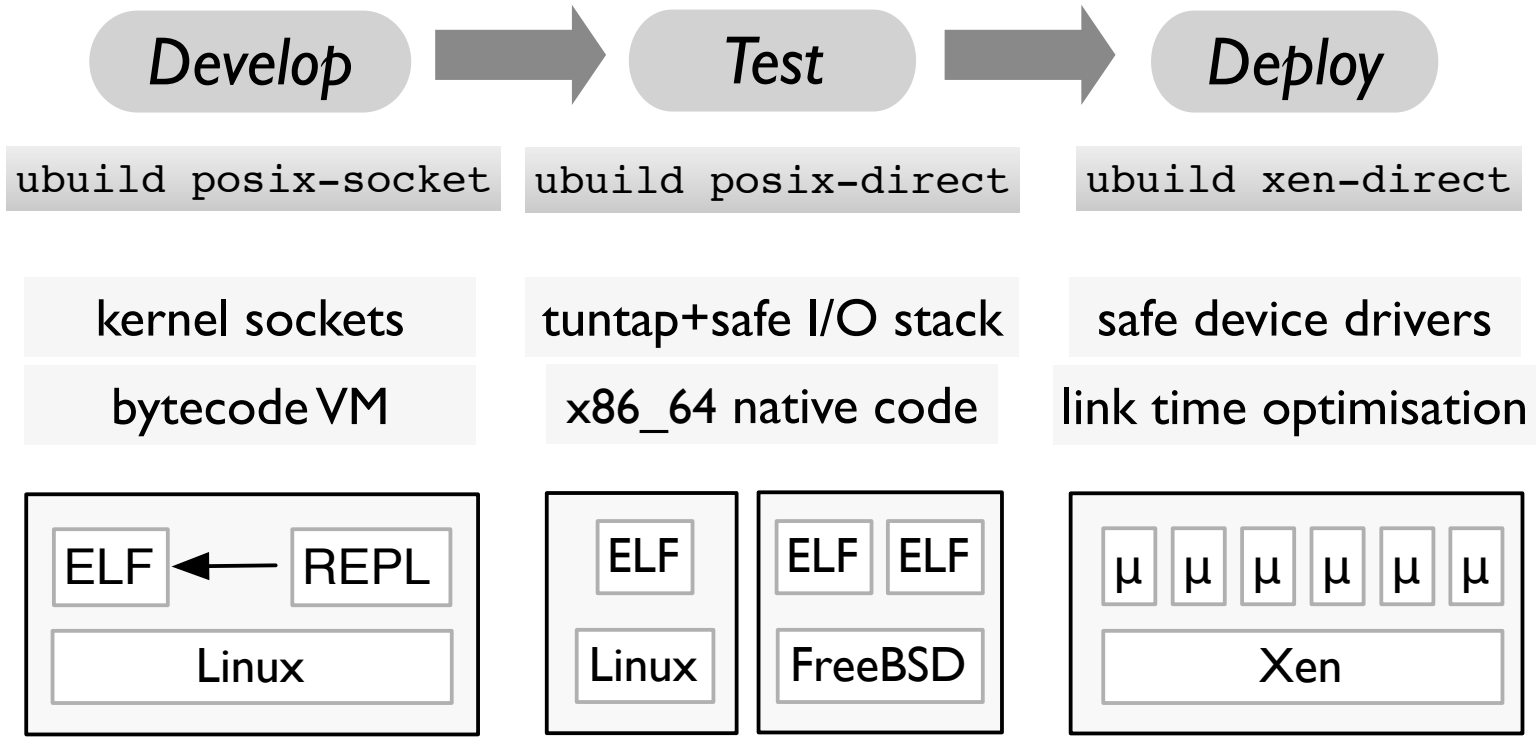
- **Static typing**
  - Pragmatic functional language (OCaml)
  - Large set of libraries provided, some used in XCP already
- **Cooperative concurrency**
  - Wrapped up in *Lwt* syntax extensions
  - Threads encapsulated and hidden within typed modules

```
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)
```

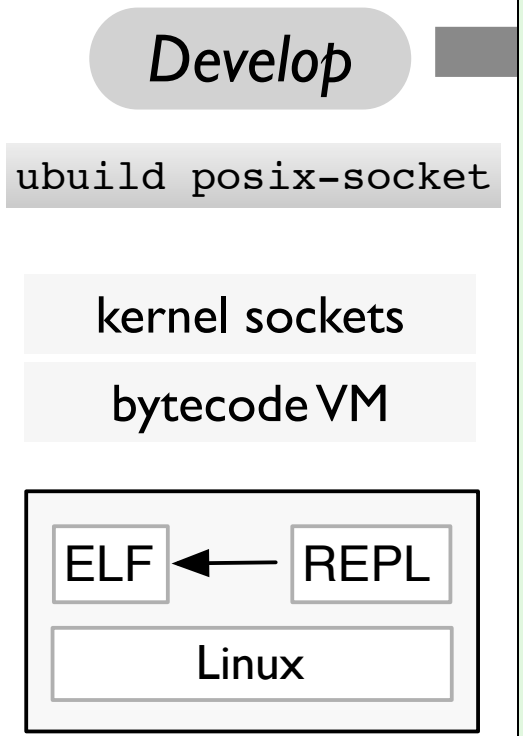
Key Features of Mirage

- **Static typing**
  - Pragmatic functional language (OCaml)
  - Large set of libraries provided, some used in XCP already
- **Cooperative concurrency**
  - Wrapped up in *Lwt* syntax extensions
  - Threads encapsulated and hidden within typed modules
- **Library-based operating system**
  - Fully re-entrant functional libraries
- **No dynamic loading**
  - Configuration evaluated at compile-time, and *sealed*
  - Recompile and redeploy to reconfigure

Progressive Specialization



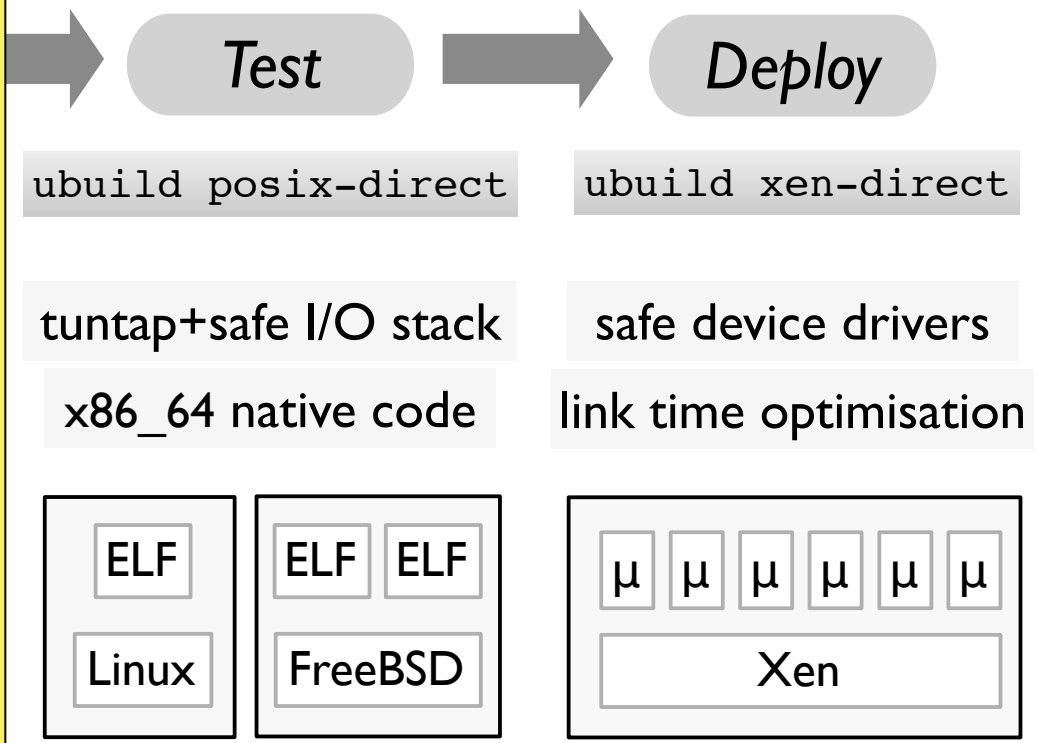
Progressive Specialization



- **Development under UNIX:**
  - full debugging environment (bytecode, debugger, gdb)
  - can use kernel sockets or tuntap networking and IO to isolate bugs
  - interactive REPL for editor integration and other niceties.

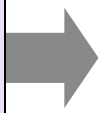
Progressive Specialization

- **Testing:**
  - Simulation backend using NS3/MPI
  - Emulate your code at scale before deploying it



Progressive Specialization

- **Xen output kernel is specialised:**
  - "operating system" is a set of libraries (e.g Netfront, Blkfront, TCP) linked with small C runtime.
  - Configuration files are evaluated at this stage, and image must be recompiled to reconfigure.
  - Data files can also be linked in directly, if relatively small.
  - Dead-code elimination is across whole OS image.

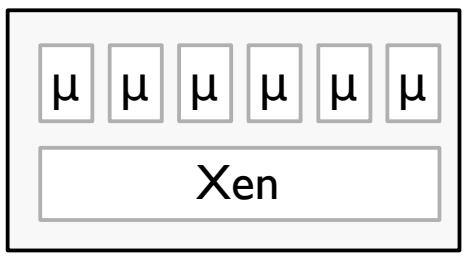


Deploy

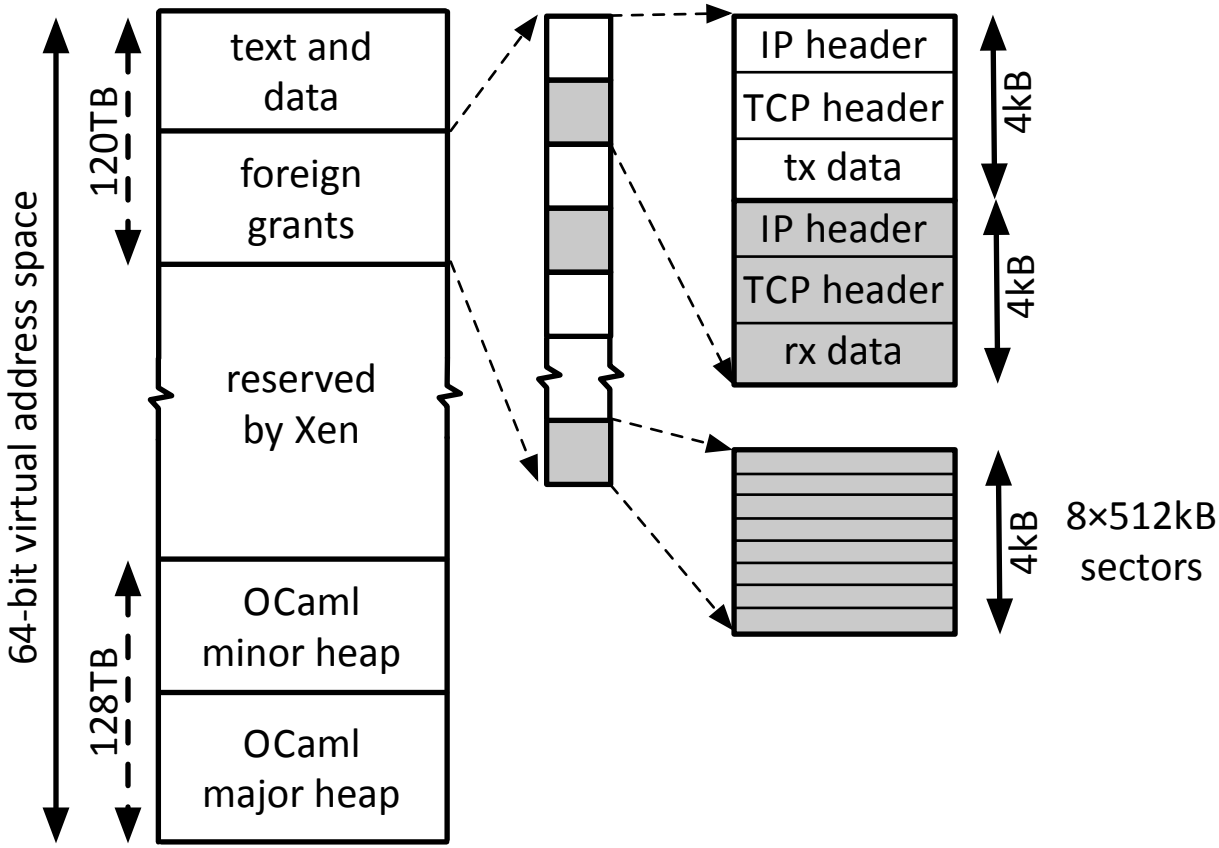
```
ubuild xen-direct
```

safe device drivers

link time optimisation



Simplified Memory Management

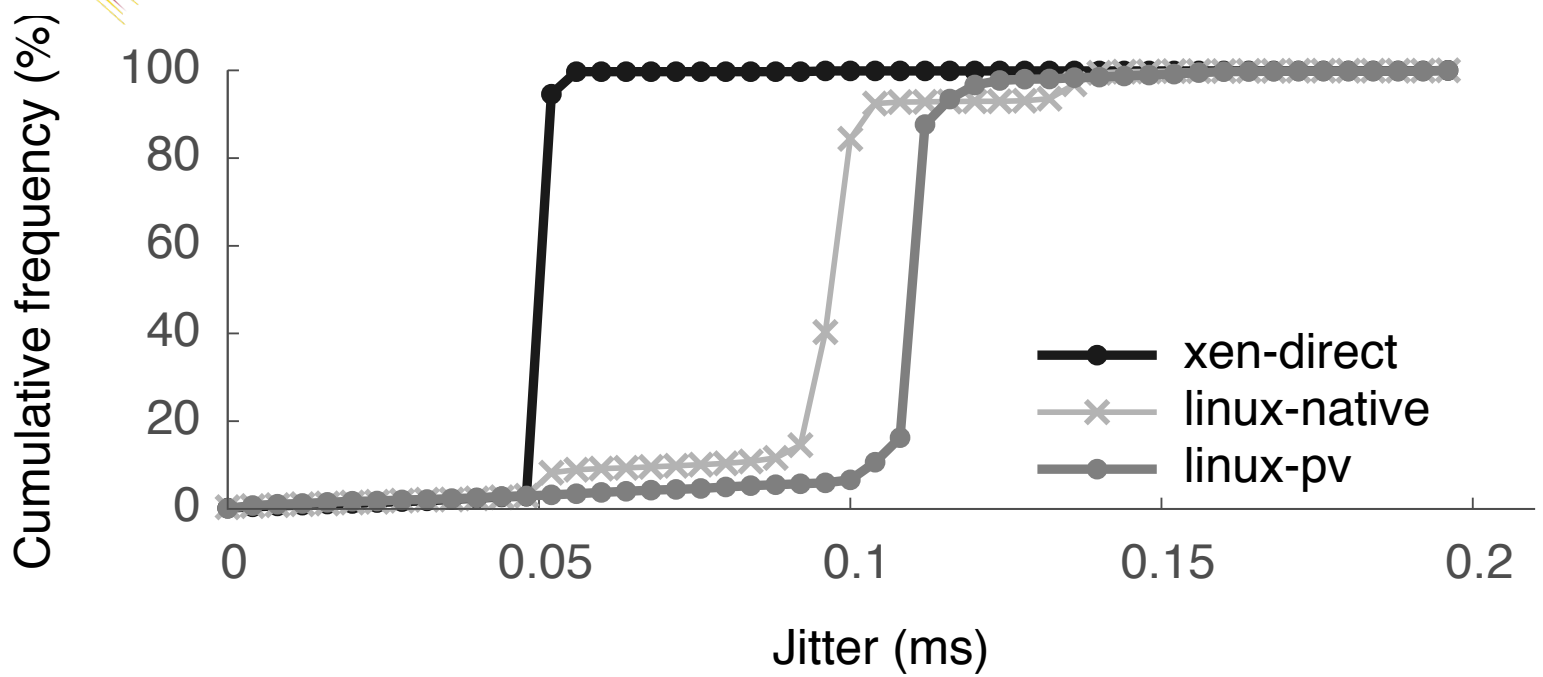


Optional **VM Seal** Hypercall

- **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.

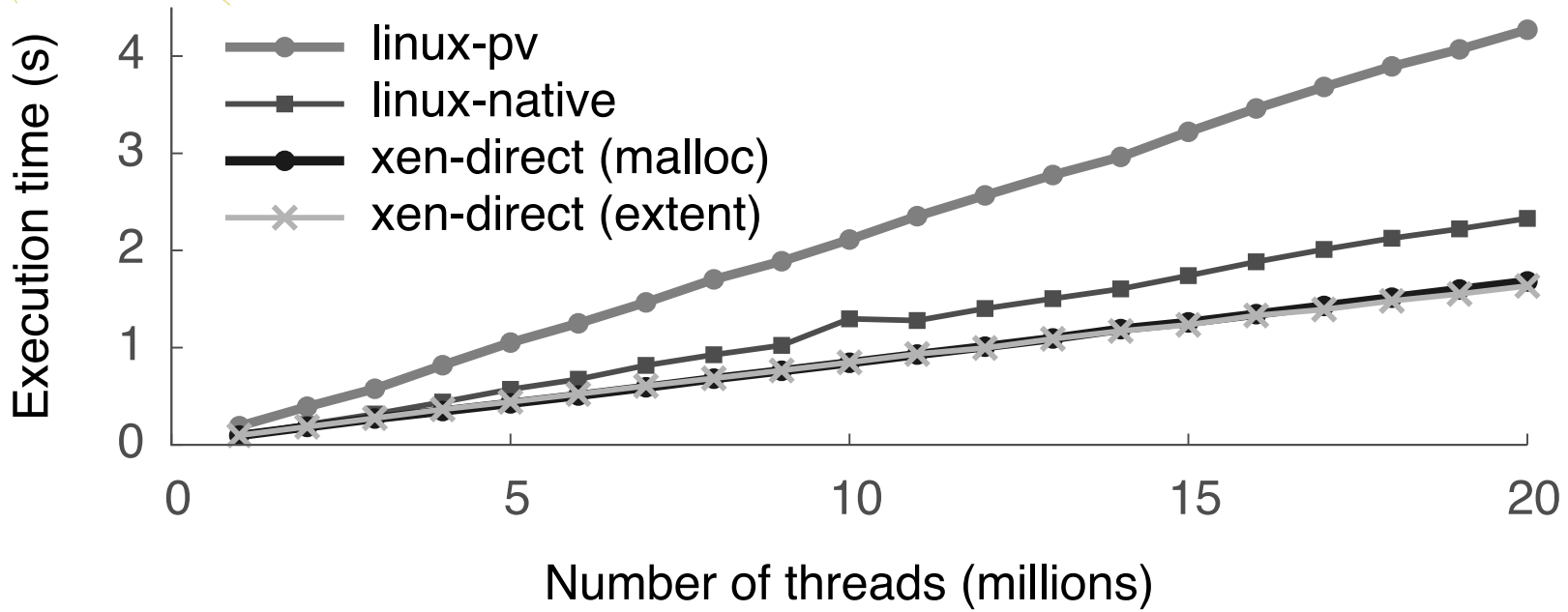
- Very easy in unikernels due to **focus on compile-time specialisation** instead of run-time complexity.

Event Driven co-threads



The microkernel is event-driven, with no pre-emption at all. Graph shows CDF of thread wakeup latency for a Mirage VM running directly on Xen, vs native or PV Linux userspace.

Single-instance Thread Scaling



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.

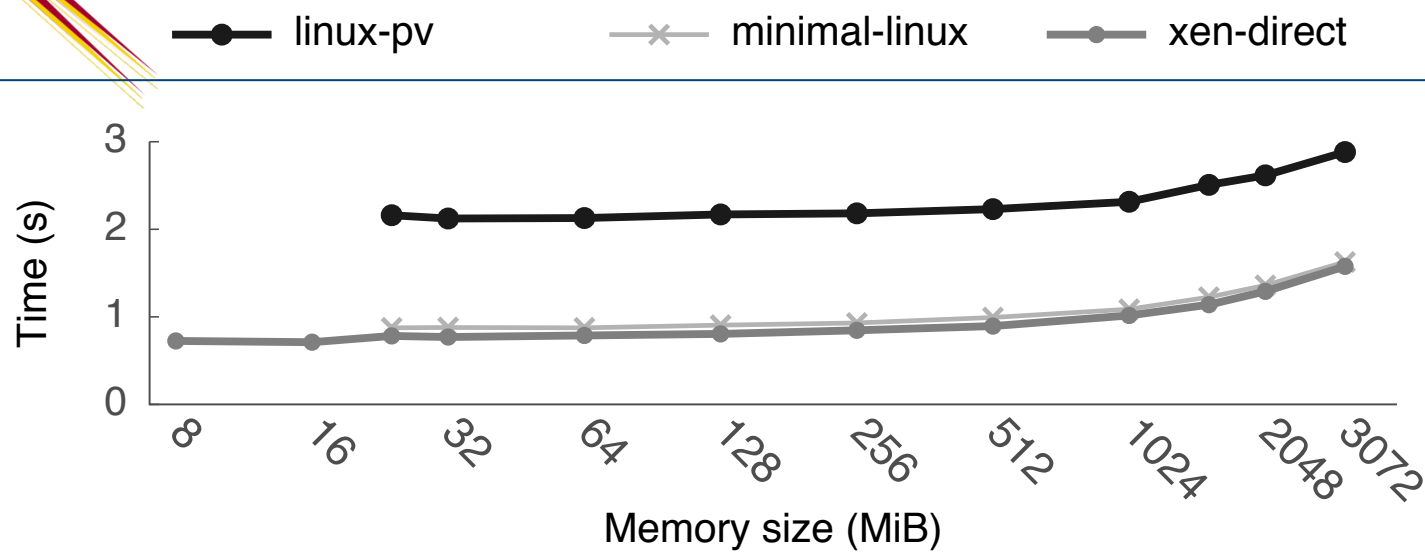
Appliance Image Size

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, so **no separate VBD required** beyond the PV kernel.

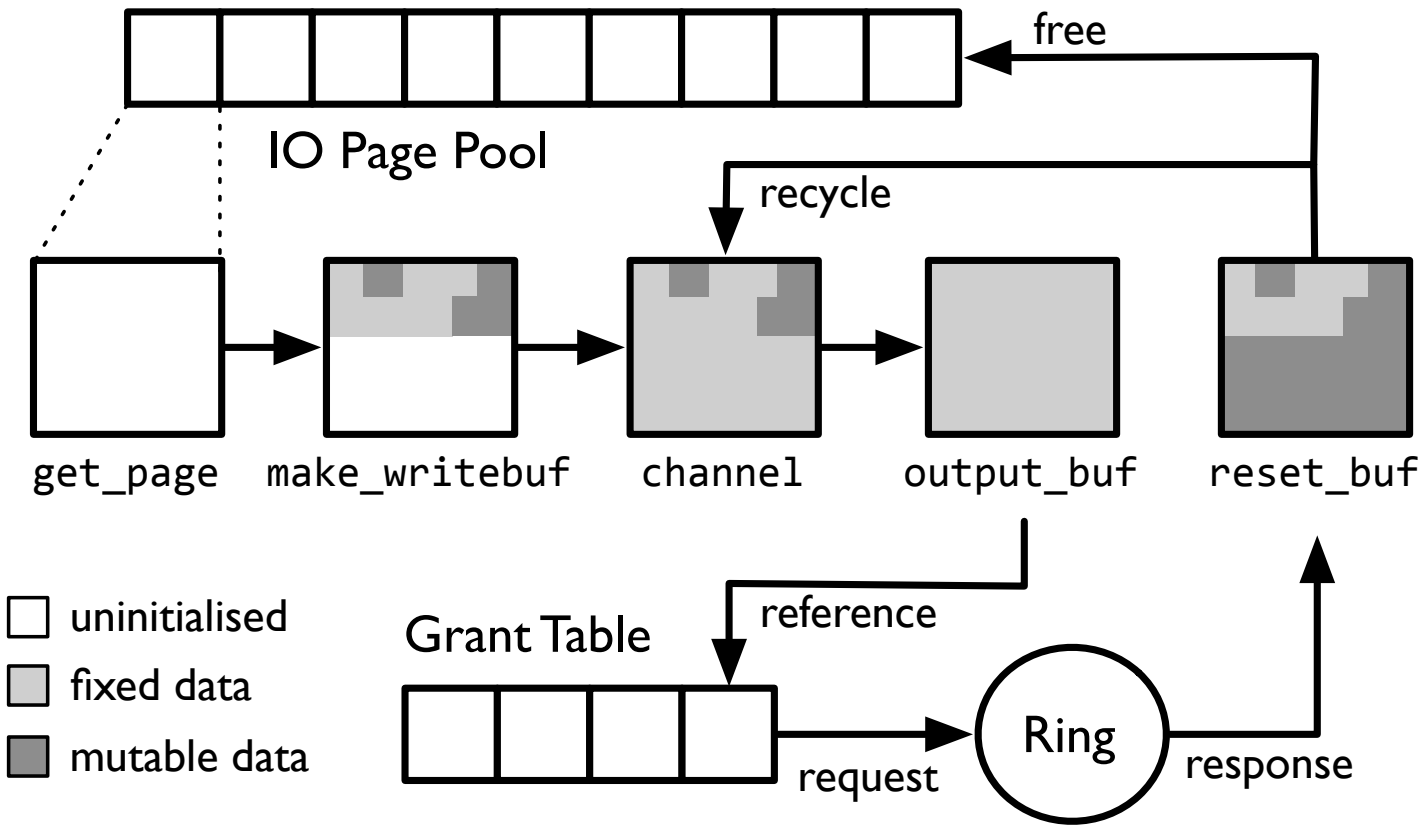
Live migration is easy and fun :-)

Microbenchmarks: Boot Time

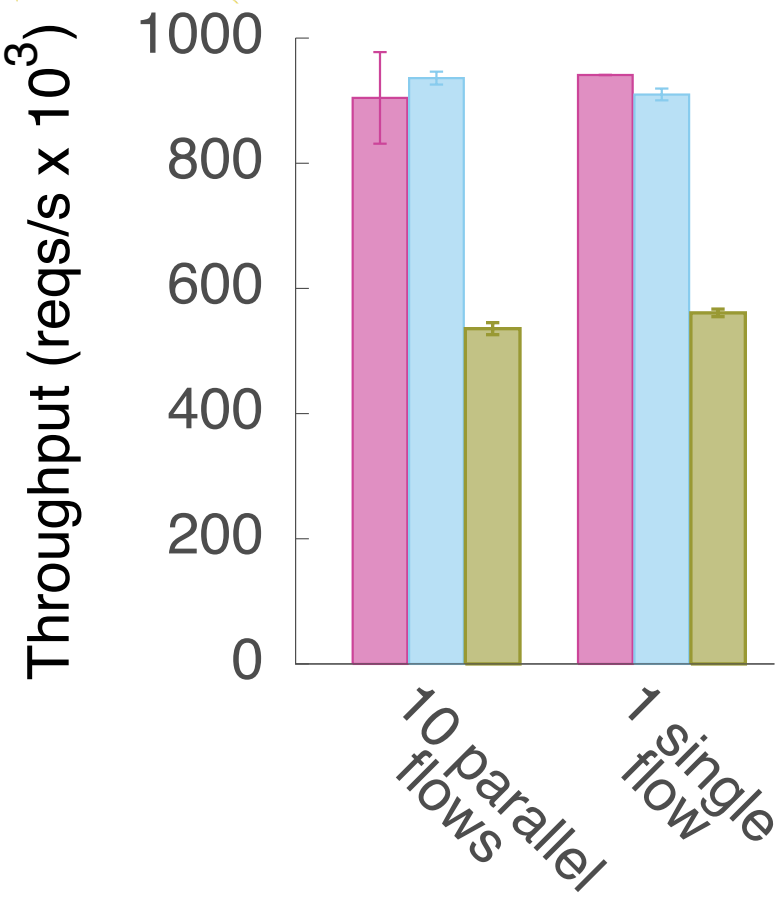


- *Minimal Linux* is a custom initrd which directly calls ioctls to send a UDP packet. The *Linux-PV* is more realistic.
- The *Xen-Direct* is a standard Mirage VM, and is limited by dom0 toolstack latency (XCP in this case).

Zero-Copy IO Buffer Management



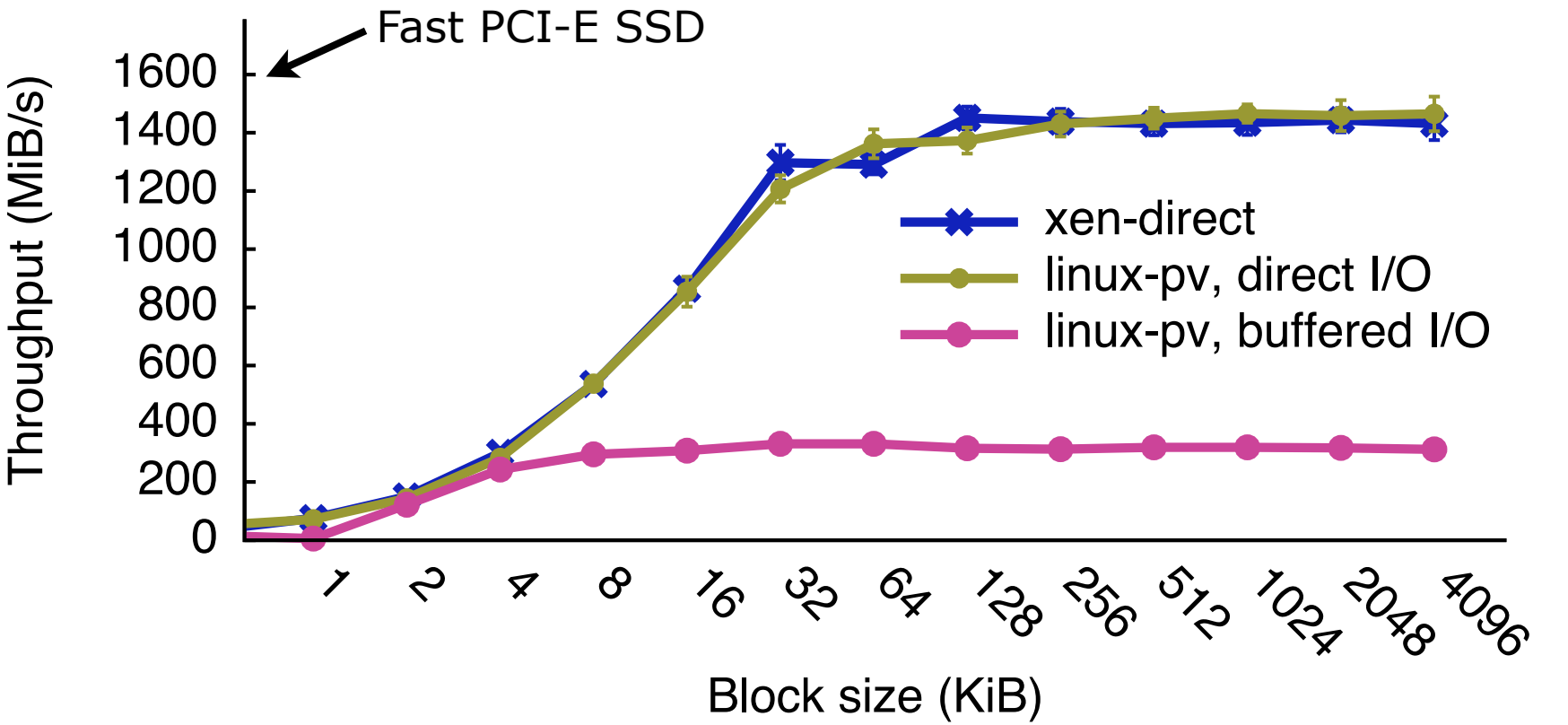
Microbenchmarks: TCP



linux-pv, tx & rx
linux-pv tx, xen-direct rx
xen-direct tx, linux-pv rx

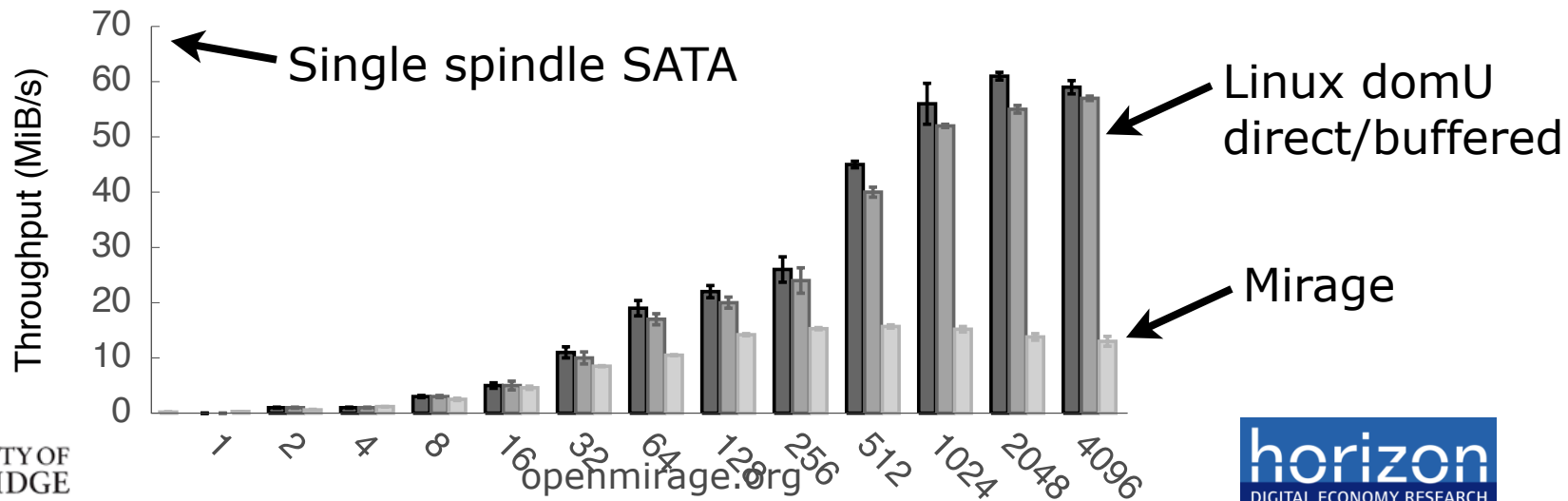
- Simple throughput test with all offload turned off to stress CPU
- Performance bug in TX path
  - …being fixed (ACKs being sent out of order)

Microbenchmarks: Block Storage



Microbenchmarks: Block Storage

- **Same Ring API as Net**
  - Unlike POSIX (libaio and so on)
  - Caching is controlled via libraries, not API
- **No reordering in the front-end**
  - How do we know what the backend storage is?



UNIVERSITY OF CAMBRIDGE

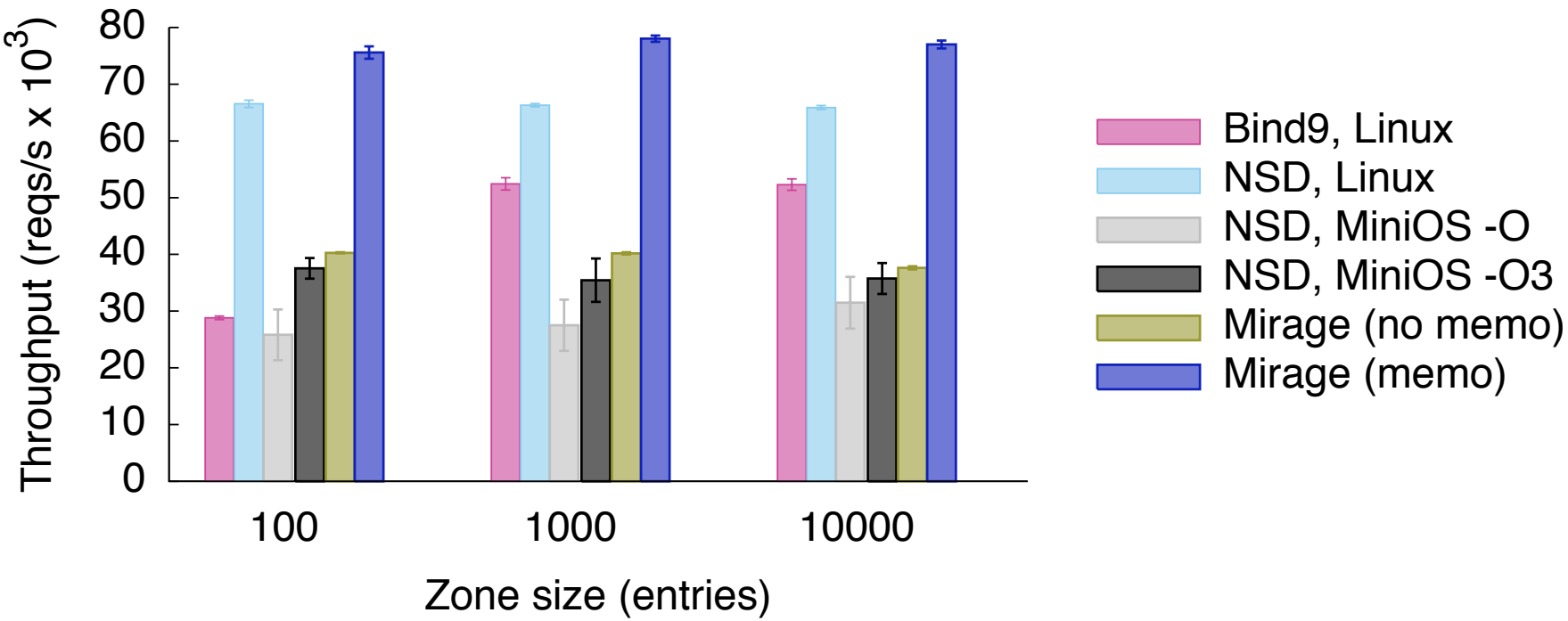
horizon DIGITAL ECONOMY RESEARCH

Microbenchmarks: Summary

- Mirage is comparable to or exceeds Linux PV performance
  - I.e., **the functional programming language overheads are offset via single-address space specialization and careful buffer management.**

- But what about in some more "realistic" scenarios?
  - DNS server appliance
  - HTTP server scaling across 6 vCPUs
  - OpenFlow Controller and Switch implementations

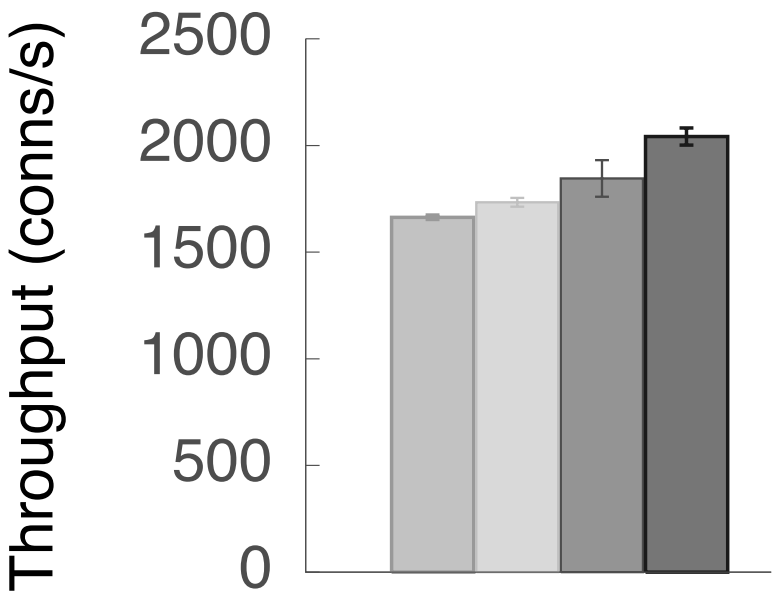
DNS Server Performance



DNS Server Performance

```
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
  )
```

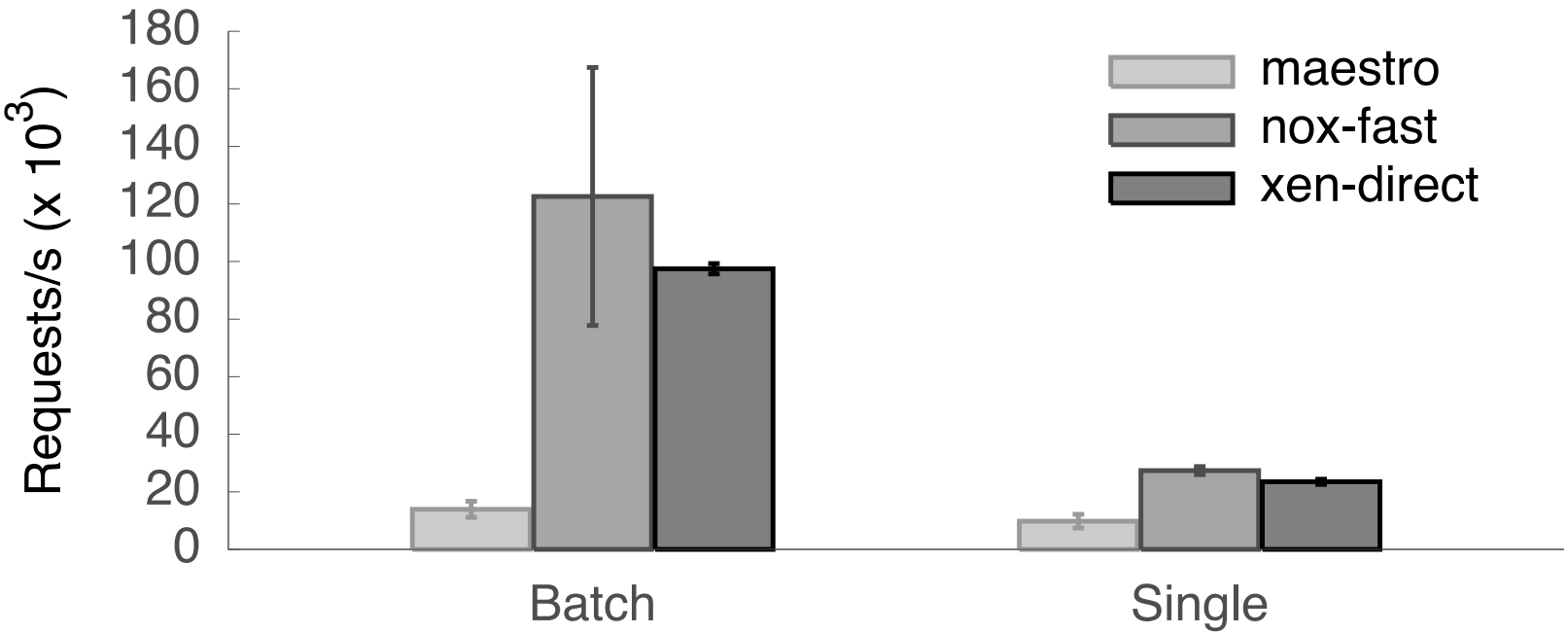
Scaling via Multiple Instances



- 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

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.

Design Discussion

- **Service domains should be built this way**
  - Debugging and moving from dom0/domU is a lot easier.
  - Dependencies are explicitly managed.
  - Resource control can be fine-grained.
  - Very useful for unit testing Xen features!
- **Which language?**
  - Several viable alternatives: HalVM (Haskell), GuestVM (Java), ErlangOnXen. More code sharing worthwhile?
  - Protocol implementations take time to mature.
  - Working on Mirage/XCP integration via "project Windsor"
- **Stub domains?**
  - Less practical for stub domains, due to hardware devices. Turn Linux into a library OS?

Mirage Roadmap

- **Developer Release for Q3 2012**
  - Monolithic repository at http://github.com/avsm/mirage
  - Adding pkg mgmt: http://github.com/OCamlPro/opam

```
$ opam init git://github.com/mirage/opam-repository
$ opam remote -add dev git://github.com/mirage/opam-repo-dev
$ opam switch mirage-3.12.1-xen
$ opam install mirage-www
```

Mirage Roadmap

- **Developer Release for Q3 2012**
  - Monolithic repository at http://github.com/avsm/mirage
  - Adding pkg mgmt: http://github.com/OCamlPro/opam

```
$ opam init git://github.com/mirage/opam-repository
$ opam remote -add dev git://github.com/mirage/opam-repo-dev
$ opam switch mirage-3.12.1-xen
$ opam install mirage-www
```

- **New architectures ongoing:**
  - **FreeBSD** kernel: http://github.com/pgj/mirage-kfreebsd
  - **Javascript**: http://ocsigen.org/js_of_ocaml
  - **ARM/MIPs64/rPI:** via FreeBSD kmod
  - **Simulation:** NS3/OpenMPI functional simulation

Mirage Roadmap

- **Developer Release for Q3 2012**
  - First release will have pure-OCaml implementations of:
    - 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

Mirage Roadmap

- **Online resources:**
  - http://www.openmirage.org
  - http://tutorial.openmirage.org
  - https://lists.cam.ac.uk/mailman/listinfo/cl-mirage
  - (draft in Q4 2012) http://realworldocaml.org
- **Offline resources:**
  - Find me at the hotel pool