Programming the Xen Cloud using OCaml

David Scott, Anil Madhavapeddy and Richard Mortier

The Xen Cloud Platform (XCP)[1] is an open-source software distribution that converts clusters of physical computers into many virtual machines, all isolated from each other via the Xen hypervisor. XCP is a large, mature and widely deployed OCaml code-base that is used as mission-critical software on hundreds of thousands of hosts today.

XCP is currently mostly used to run conventional operating systems, such as Linux or Windows. Our next-generation version of XCP bypasses this completely, and permits programmers to compile OCaml code directly into virtual machines that use the low-level APIs provided by Xen, without all the intervening layers of abstraction imposed by virtualisation. This system, under development for 2 years, is based on *Mirage*, an OCaml operating system[2] specialised to run directly on the low-level APIs provided by hypervisors.

In this talk we'll show you how to write OCaml code targetting these new "cloud APIs". You'll be able to progressively remove decades of legacy cruft from your software stack, increasing performance, security, reliability. We provide a comprehensive set of open-source, pure OCaml networking and storage libraries, enabling your application to communicate with the outside world via standard protocols such as SSH and TCP. You'll be able to take advantage of "Software Defined Networking" by configuring state-of-the-art network switches with the OpenFlow protocol while storing application state on disks formatted with standard filesystems such as FAT32.

The public cloud is a vast datacenter, and OCaml is the language of choice for constructing fast, safe and reliable services on it.

Our contributions

We'll describe the Xen Cloud Platform (XCP), an OCaml software stack which connects physical servers running the `Xen` hypervisor together into *Resource Pools*, capable of running 1000s of Virtual Machines (VMs). The OCaml software handles the allocation of physical hardware (RAM, CPUs, PCI devices) to VMs, sets up network and storage links and orchestrates activities such as VM live migration. XCP is a building block used by cloud orchestration layers such as CloudStack[3] and OpenStack.[4]

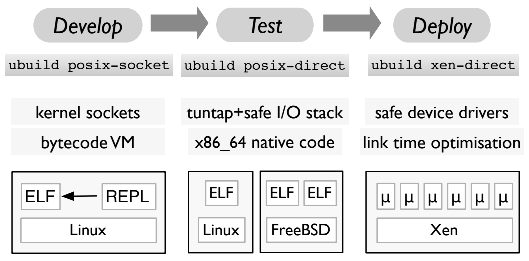


Figure 1: OCaml code can be progressively specialised into standalone cloud microkernels.

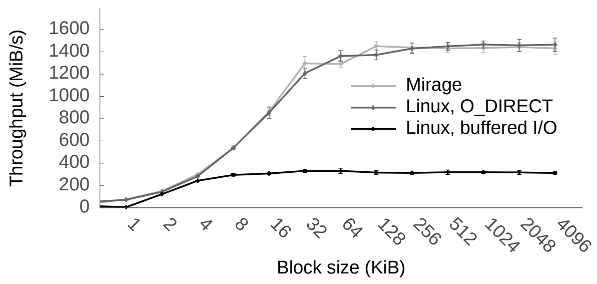
We'll demonstrate Mirage, our OCaml-based programming framework which allows the construction of specialised virtual machines that run directly on `Xen` hypervisors without recourse to a traditional guest OS. Mirage applications are complete software stacks including everything from application logic to low-level device drivers in one single, optimised microkernel binary.

Applications coded with Mirage can be very **small** and boot using just 4MB RAM; Figure 2 compares the number of lines of code in each language in Mirage to other conventional operating systems and applications. Many of the layers in traditional software stacks are are duplicated in a virtual cloud environment; a problem overcome by our approach. Almost all the code in a Mirage kernel is type-safe OCaml, from the device drivers, to the TCP/IP stack, to the application itself.

Applications coded with Mirage are very **fast**, despite all networking and device drivers being written in pure OCaml; Figure 3 shows an application achieving over 10Gb/s disk I/O rates through a fast PCIe SSD storage device, and another application achieving excellent DNS DNS query throughput when pitted against conventional C servers.

Applications coded with Mirage are naturally **distributed**; built-in rendezvous and communication proto-

[1]`http://www.xen.org/product/cloudxen.html`
[2]`http://www.openmirage.org/`
[3]`http://www.cloudstack.org/`
[4]`http://openstack.org/`



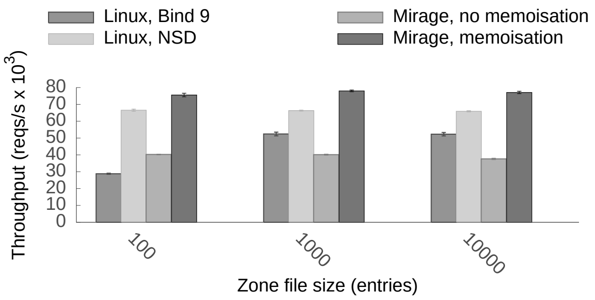


Figure 3: (*left*) A Mirage application using all-OCaml device drivers and zero-copy I/O achieves over 10 Gigabits per second of throughput performing random reads from a PCIe SSD storage device. (*right*) DNS queries-per-second served comparing two Mirage applications against industry standard Bind and NSD running on Linux guests. Addition of memoisation to the Mirage application took barely a dozen lines of OCaml.

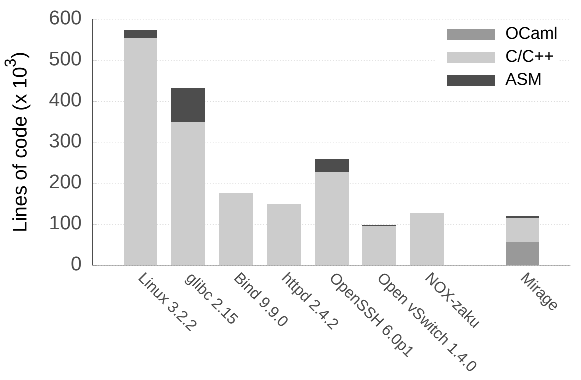


Figure 2: Comparison of the number of lines of code in different languages within the Mirage core compared to contemporary software systems.

cols allow applications to spread themselves across many VMs in the cloud.

Applications coded with Mirage are **feature-rich**; Mirage comes complete with an extensive set of pure OCaml libraries including:

I/O pages : a representation of machine memory pages allowing Mirage applications to perform zero-copy data transfers.

cstruct : a lightweight `camlp4` extension which allows C-style structs to be "attached" to a buffer and fields safely manipulated through the "struct". This is used as the foundation for all network and block protocol implementations, giving high performance while retaining compact notation.

Device drivers : for both disk and network for the `Xen` hypervisor. This allows Mirage guests to work easily and transparently on public clouds such as Amazon EC2.

TCP/IP : supporting multiple application interfaces allowing some applications to manipulate TCP packets efficiently while others simply use a high-level channel abstraction.

SSH : allowing secure remote control of your Mirage application and the ability to *ssh* in to an OCaml toplevel.

DNS : providing both client name resolution *and* a DNS server.

OpenFlow : enabling state-of-the-art hardware and software switches to be dynamically reconfigured. Your OCaml program can become the heart of a "Software Defined Network".

FAT32 : a simple filesystem which allows application state to be stored on disk.

HTTP : a webserver stack (including `camlp4` extensions to directly write XML, JSON and HTML directly).

These libraries are all individually first-class entities; they are designed to also be compiled under UNIX using the normal Lwt toolchain. This permits developers to use a familiar development environment, and only compile to Xen kernels for production. The key difficulty with deploying such microkernels in the past has been managing them without having a full OS available (e.g. logging, shells, etc). We have extended XCP to provide many of these facilities directly as Mirage libraries, thus turning it into a distributed OS that supports microkernel guests.

How you can get involved

We'll show you how to get started and install XCP/Mirage, and demonstrate building a distributed application that is deployed directly to EC2. Anyone with an Amazon account can convert their homepage into their own OCaml kernel. We are also looking for volunteers to port Mirage to more platforms, such as the Raspberry Pi!