Melange, Mirage and Jitsu

A series of papers published by Anil Madhavapeddy

Melange, Mirage and Jitsu

- Melange: Creating a “Functional” Internet – EuroSys 2007
- Unikernels: Library Operating Systems for the Cloud (Mirage) – ASPLOS 2013
- Jitsu: Just-In-Time Summoning of Unikernels – NSDI 2015

Summary of these works

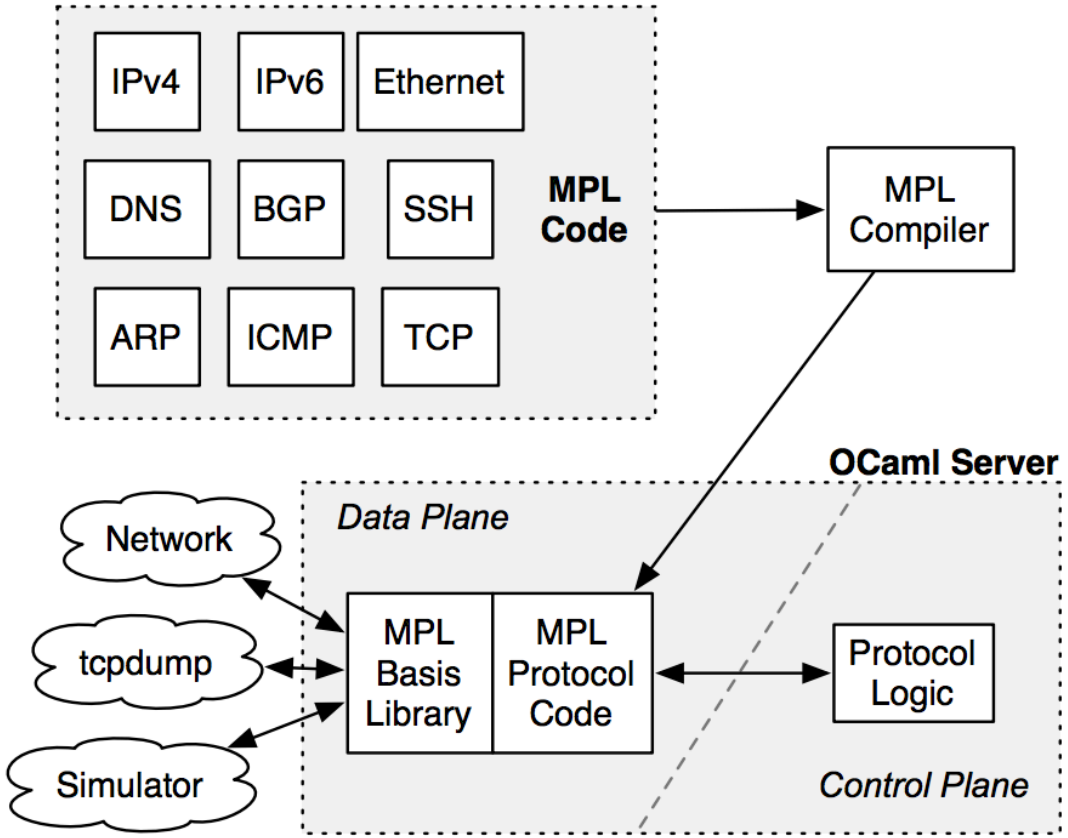
- Use old-school functional programming language (OCaml) to construct real-world Internet applications.
  - Break a stereotype.

- Create a new isolation abstraction - unikernel.
  - Run on virtual machine, better resource isolation than container.
  - Small image footprint, lightening fast boot-up time.

Melange

- Recreate existing Internet applications using OCaml.
- They build a SSH server and a DNS server in OCaml.
- It is surprising that the new applications have better performance.

Inside Melange



Inside Melange

- The architecture is nothing special compared to a regular server.
- The only exception is the MPL (Meta Packet Language) library.
  - MPL is a DSL (Domain specific language).
  - Helps Melange to preserve type safety.
- Melange is type safe.

Melange Performance

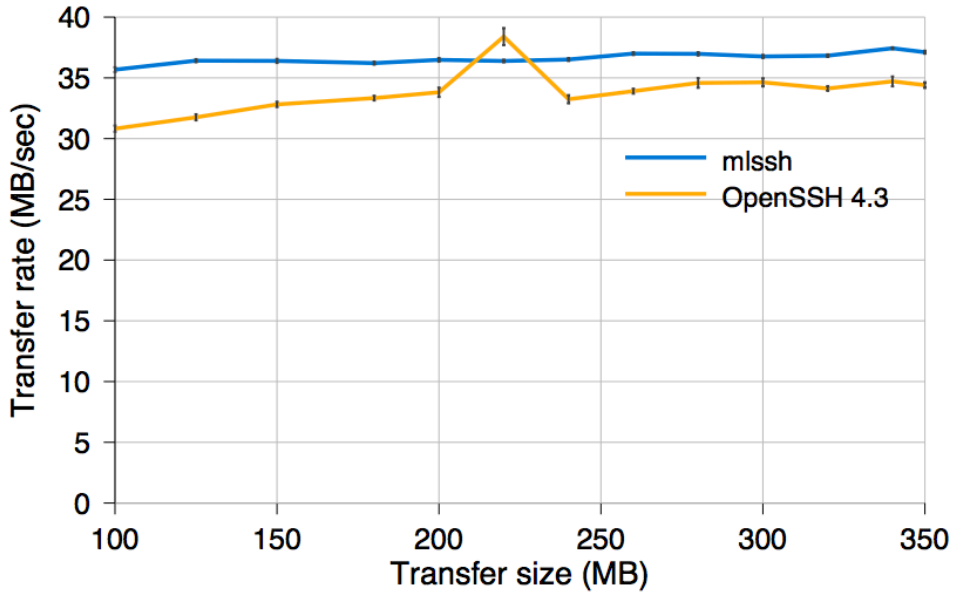


Figure 8: Throughput of OpenSSH vs MLSSH with encryption and message hashing disabled *(higher is better)*.

Melange Profiling

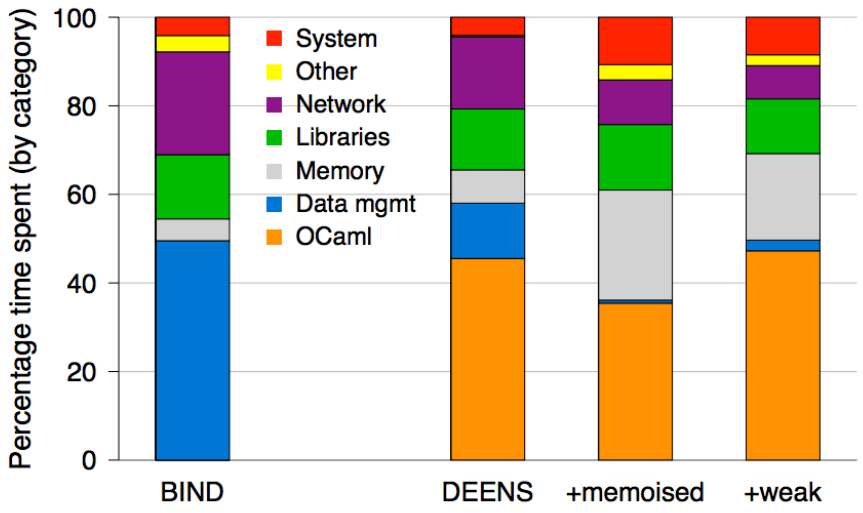


Figure 15: Normalised profiling results for the DNS servers, showing how each application spends its time serving queries.

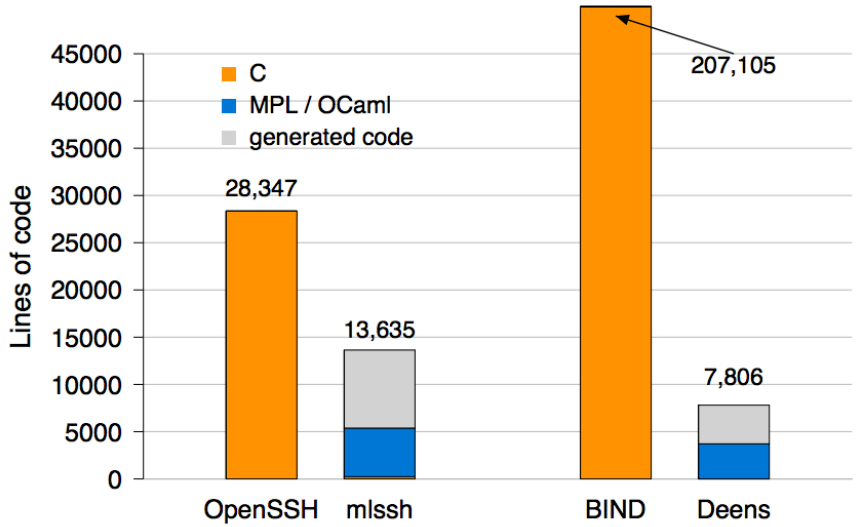


Figure 16: Relative code sizes for MPL/OCaml and C code *(lower is better)*.

Mirage, Unikernels for the Cloud

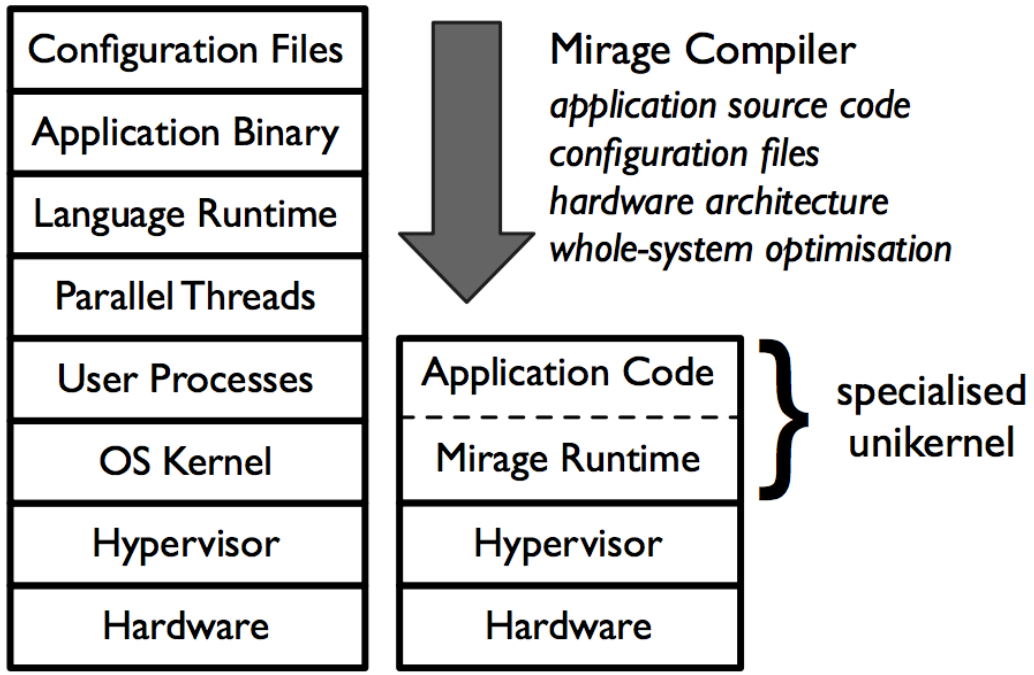
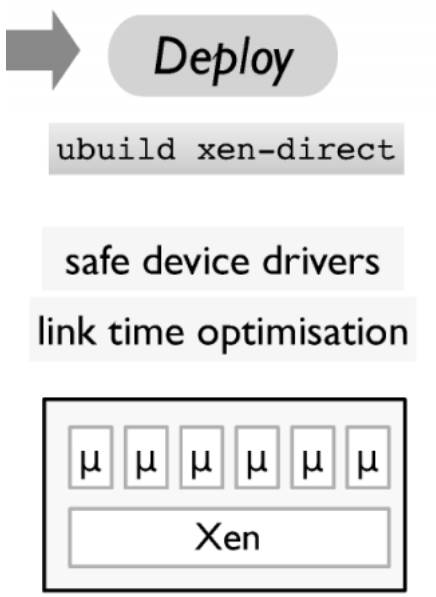
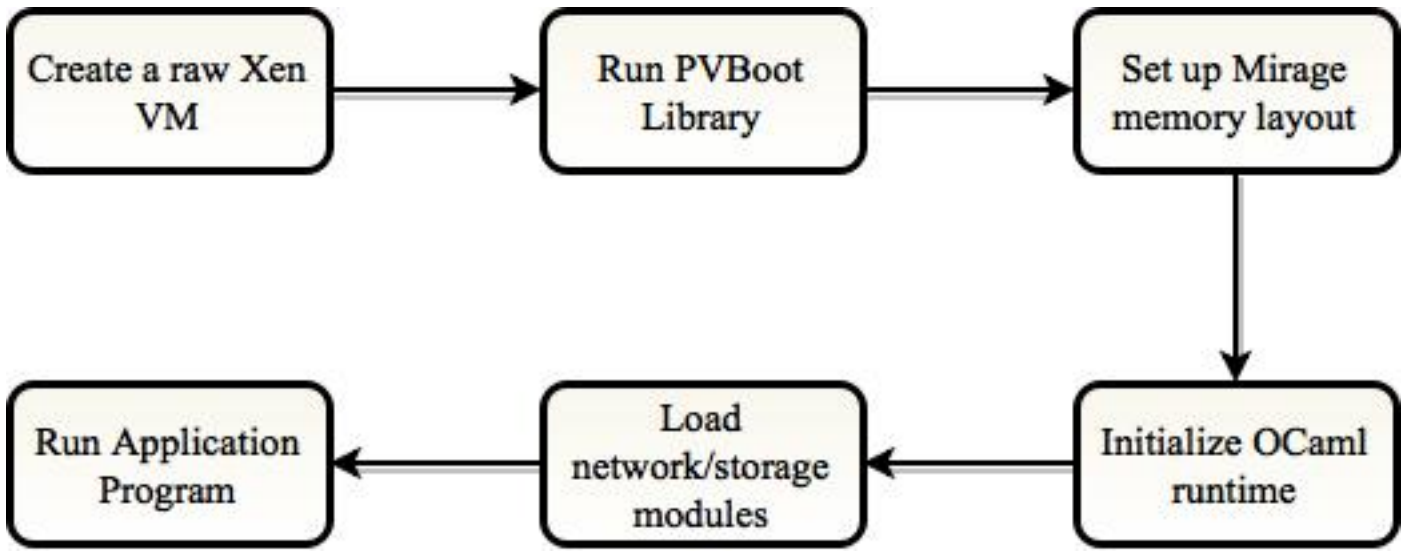


Figure 1: Contrasting software layers in existing VM appliances *vs.* unikernel's standalone kernel compilation approach.

Mirage, Unikernels for the Cloud



Mirage Bootup



Mirage Bootup

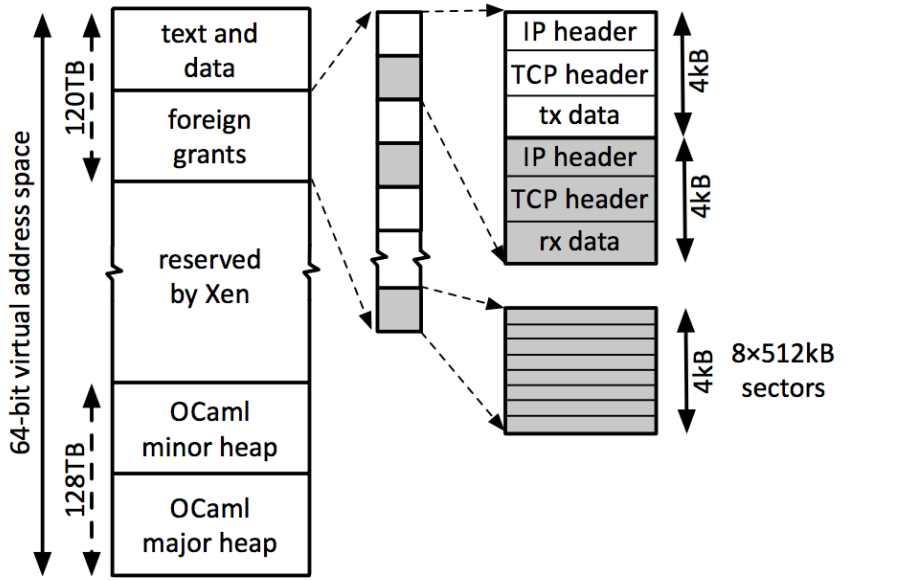
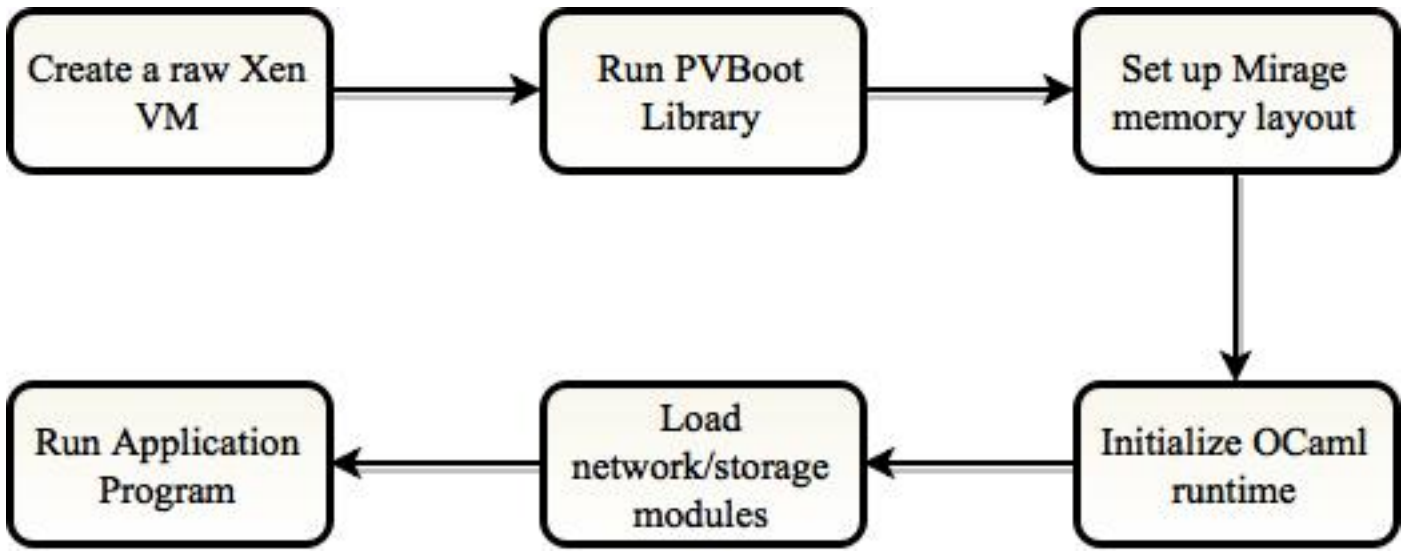


Figure 2: Specialised virtual memory layout of a 64-bit Mirage unikernel running on Xen.

Mirage Bootup



Mirage Bootup time

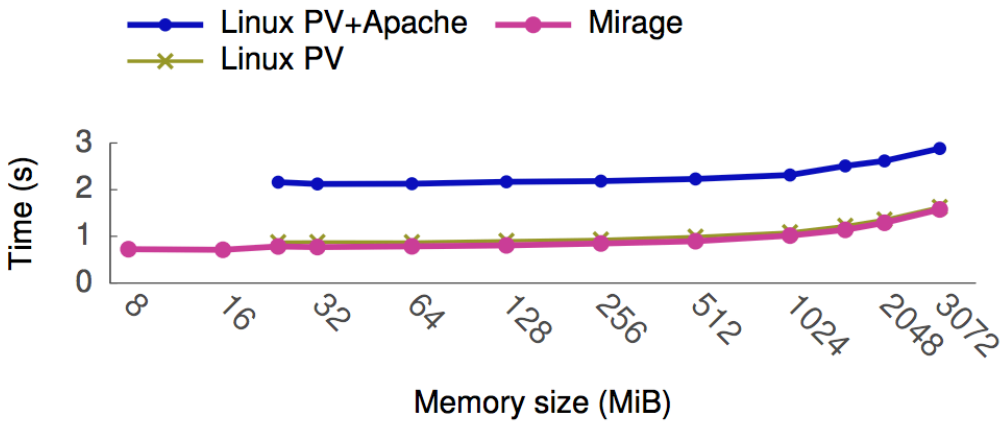


Figure 5: Domain boot time comparison.

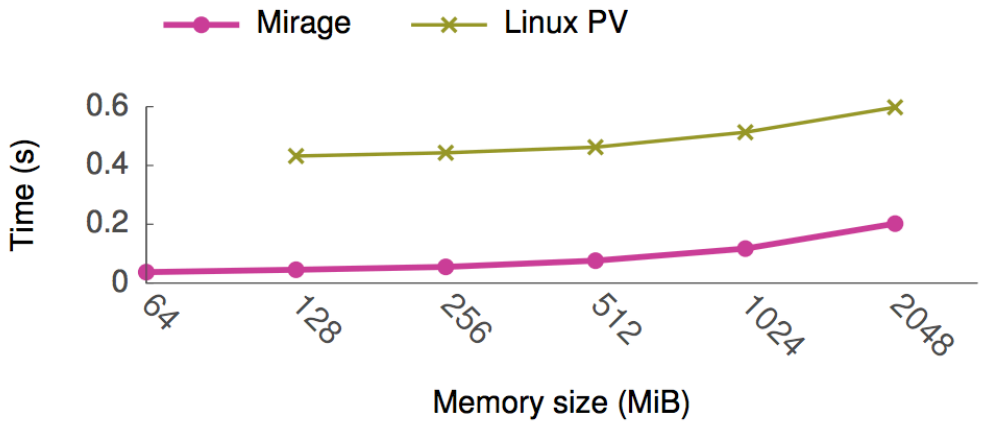


Figure 6: Boot time using an asynchronous Xen toolstack.

Mirage Performance

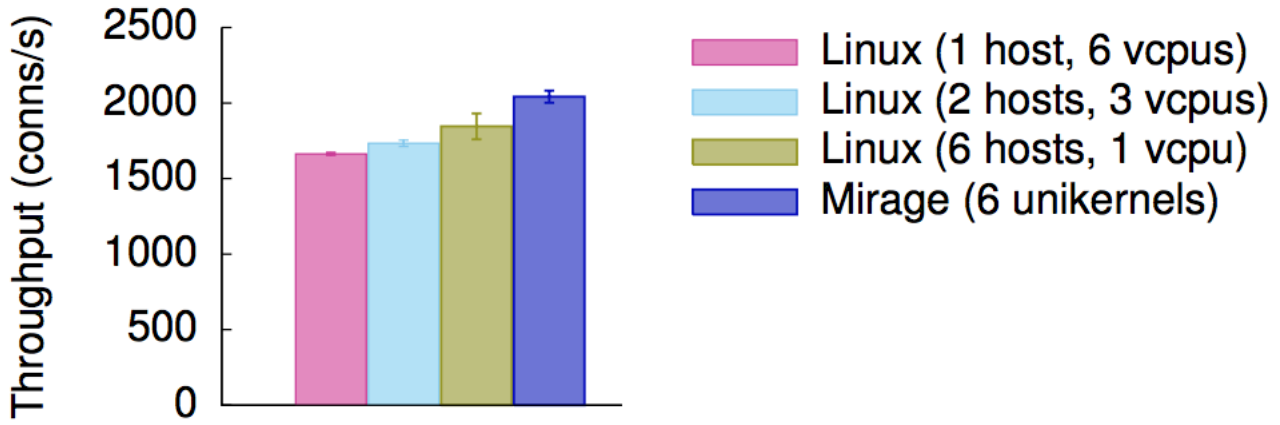


Figure 13: Static page serving performance, comparing Mirage and Apache2 running on Linux.

Pros of Mirage

- Type safe.
- Small memory footprint, lightening fast bootup time.
- Good scalability.
- Good performance.

Cons of Mirage

- Simple. 1 VM - 1 vCPU - 1 thread.
- No locks, no multi-threaded program.
- Hard to learn.

Cons of Mirage

```
$ cat hello/unikernel.ml
open Lwt.Infix

module Hello (Time : Mirage_time_lwt.S) = struct

  let start _time =

    let rec loop = function
      | 0 -> Lwt.return_unit
      | n ->
        Logs.info (fun f -> f "hello");
        Time.sleep_ns (Duration.of_sec 1) >>= fun () ->
        loop (n-1)
    in
    loop 4

end
```

Jitsu

- Deploy on-demand VMs on embeded system (ARM platform).

Jitsu Motivation

- Embeded system requires small power consumption.
- The resource is limited.
- Real-time processing.

Jitsu Overview

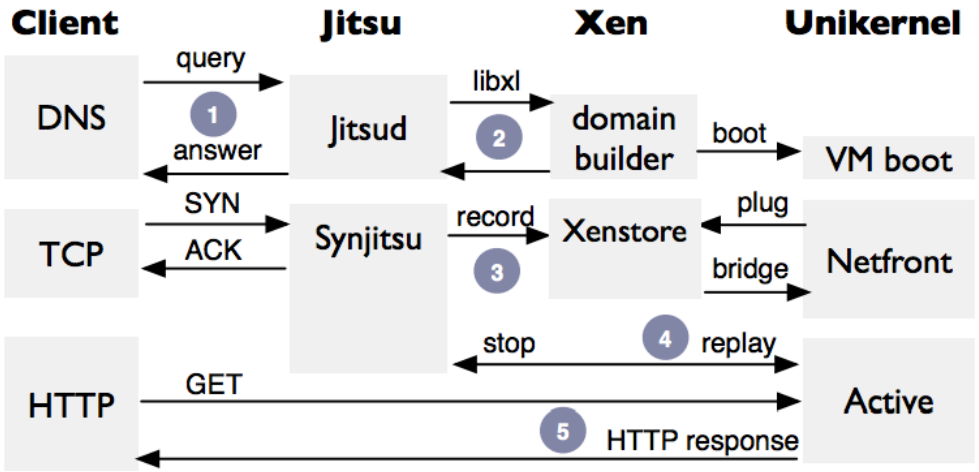


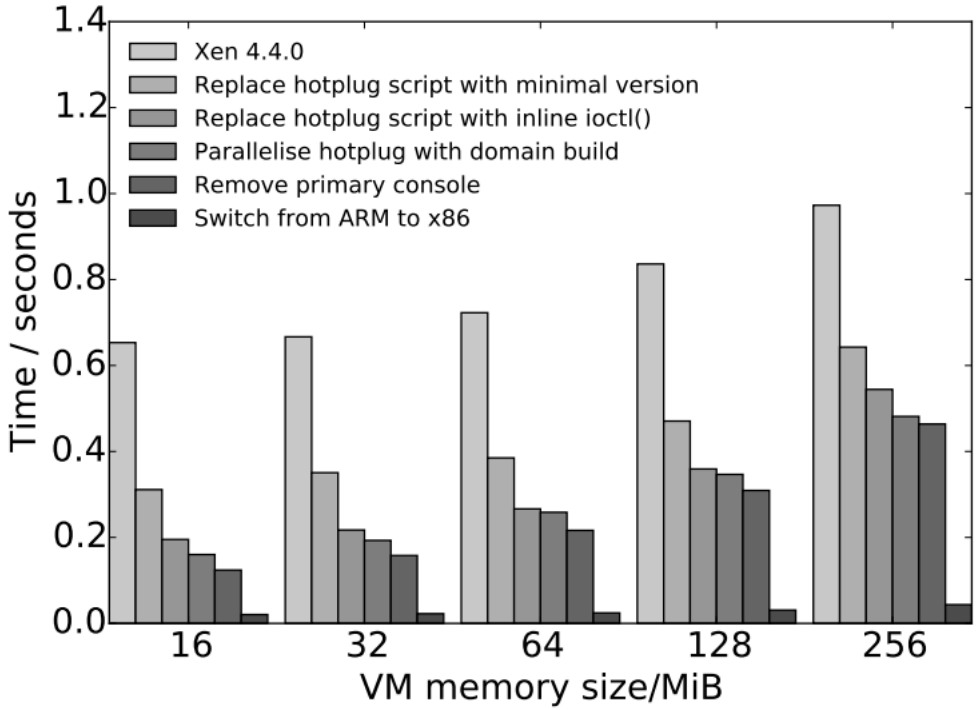
Figure 6: How Jitsu masks boot latency. ❶ DNS request triggers unikernel launch; ❷ response sent when domain building completes but before networking is active; ❸ TCP requests are buffered into XenStore until bridging is setup; and ❹ the active unikernel replays the buffered connections before ❺ directly serving traffic.

Jitsu Optimization

- Optimize boot time.
  - Domain building.
  - Parallel device attachment.

Jitsu Optimization

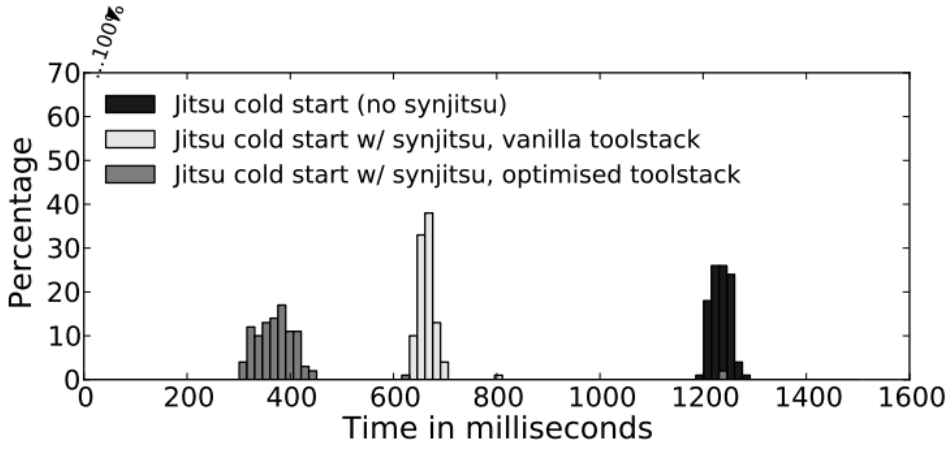
- Optimize boot ti
  - Domain buildi
  - Parallel device



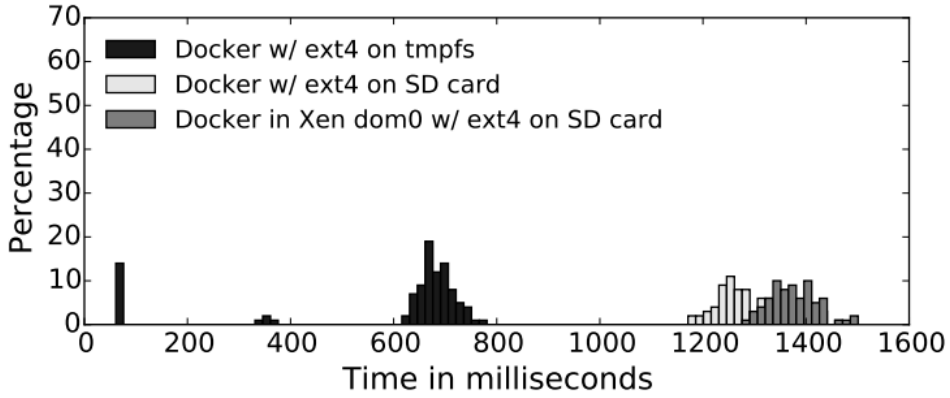
Jitsu Optimization

- Optimize boot time.
  - Domain building.
  - Parallel device attachment.
- Communication conduits.
  - In-memory communication channel between hypervisor and the unikernel VM.
- The Jitsu Directory Service

Jitsu Boot Performance



(a) Using Jitsu from a cold start (booting a new unikernel) without Synjitsu, with Synjitsu and the ordinary toolstack, and with Synjitsu and the optimised toolstack.



(b) Using Docker: direct on Linux with RAM filesystem (`tmpfs`) and SD card, and on Xen in dom0.

Conclusion

- Functional programming language can build fast real-world applications while maintaining type safety.
  - NFV needs type safety because NF directly manipulates packet buffer.

- Unikernel provides an abstraction that boots faster than VM and has better isolation than container.
  - This is very important for NFV.