JITSU: JUST-IN-TIME SUMMONING OF UNIKERNELS

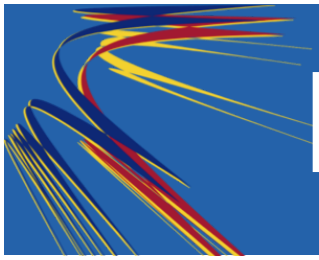
Anil Madhavapeddy University of Cambridge @avsm

Magnus Skjegstad University of Cambridge @MagnusSkjegstad

on behalf of: Thomas Gazagnaire, David Scott, Richard Mortier, Thomas Leonard, David Sheets, Amir Chaudhry, Jon Ludlam, Balraj Singh, Jon Crowcroft, Ian Leslie

http://openmirage.org/
http://decks.openmirage.org/nsdi2015/


THE IOT SPRING



FASTER THAN LIGHT?

Many network services suffer as *latency* increases, e.g.,

- Siri
- Google Glass

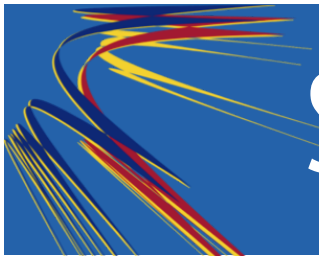
...to say nothing of how they operate when disconnected.

So let's move the computation closer to the data and reduce dependency on a remote cloud

THE PAST YEAR

- **Heartbleed**: 17% of *all* Internet secure web servers vulnerable to a single bug. Described as "catastrophic" by Bruce Schneier.
- **ShellShock**: CGI, Web, DHCP all vulnerable to code execution. Millions of sites potentially vulnerable.
- **JP Morgan**: 76 million homes and 8 million small businesses exposed in a single data breach.
- **Target**: 40 million credit cards stolen electronically.

System security is in a disastrous state, and seemingly getting worse with IoT.



STRONGER THAN STEEL?

We earlier noted the many recent network security problems:

- Heartbleed
- Shellshock

...and such bugs will reoccur, now in our homes, cars, fridges

So let's build fundamentally more robust edge network services

- **VMs are the strongest practical isolation on physical devices**
  - *But resource heavy on embedded devices*
  - *Long boot times and management overheads*
- **Containers are really easy to use**
  - *But isolation is poor due to wide interfaces*
  - *Often requires disk I/O to boot*

Can we eliminate tradeoff between latency and isolation at the edge?

MEANWHILE, IN YOUR CAR...

instrument dashboard." The infotainment system is by nature open and Internet-connected and thus open to outside attacks, while other mission-critical driving instrument systems cannot afford to be compromised. Installing the two systems on the same embedded device can yield up to several hundreds of dollars in saving per unit, and ultimately tens of millions of dollars in savings per year.

-- embedded-computing.com via @whitequark

THE UNIKERNEL APPROACH

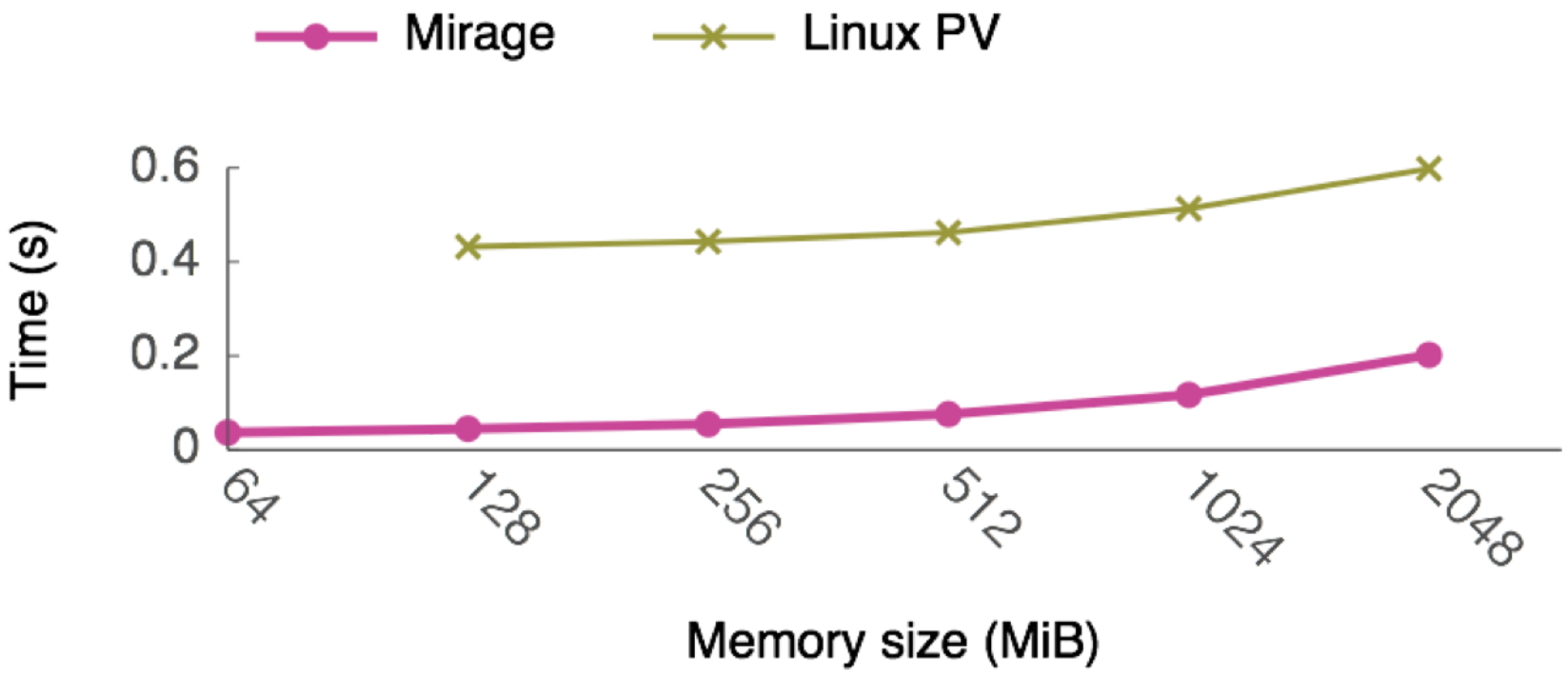
Unikernels are specialised virtual machine images compiled from the full stack of application code, system libraries and config

This means they realise several benefits:

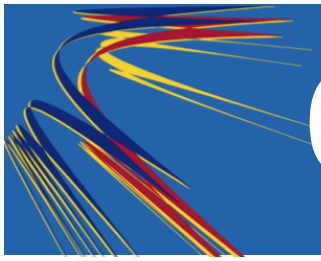
- **Contained**, simplifying deployment and management.
- **Compact**, reducing attack surface and boot times.
- **Efficient**, able to better use host resources.

REAL TIME BOOT

Unikernels can boot and respond to network traffic in real-time.



See Also: HotCloud 2011, ASPLOS 2013, Communications of the ACM Jan 2014



CONTRIBUTIONS

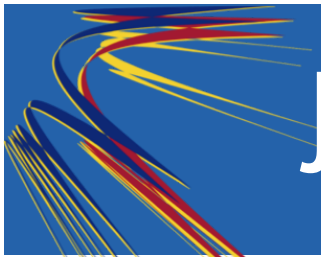
Built platform support required for ARM cloud deployments:

- **Ported unikernels to the new Xen/ARMv7 architecture**
  - *Runs VMs on commodity ARM hardware (Cubieboard)*
  - *Type-safe, native code down to the device drivers*
- **Constructed Jitsu toolstack to launch unikernels on-demand**
  - *Race-free booting of unikernels in response to DNS*
- **Evaluated against alternative service isolation techniques**
  - *E.g. Docker containers*

These slides were written using MirageOS on Mac OS X:

- They are hosted in a **2MB Xen unikernel** written in statically type-safe OCaml, including device drivers and network stack.
- Their application logic is just a **couple of source files**, written independently of any OS dependencies.
- Running on an **ARM** CubieBoard2, and hosted on the cloud.
- Binaries small enough to track the **entire deployment** in Git!

ARTIFACT: MIRAGE OS 2.0

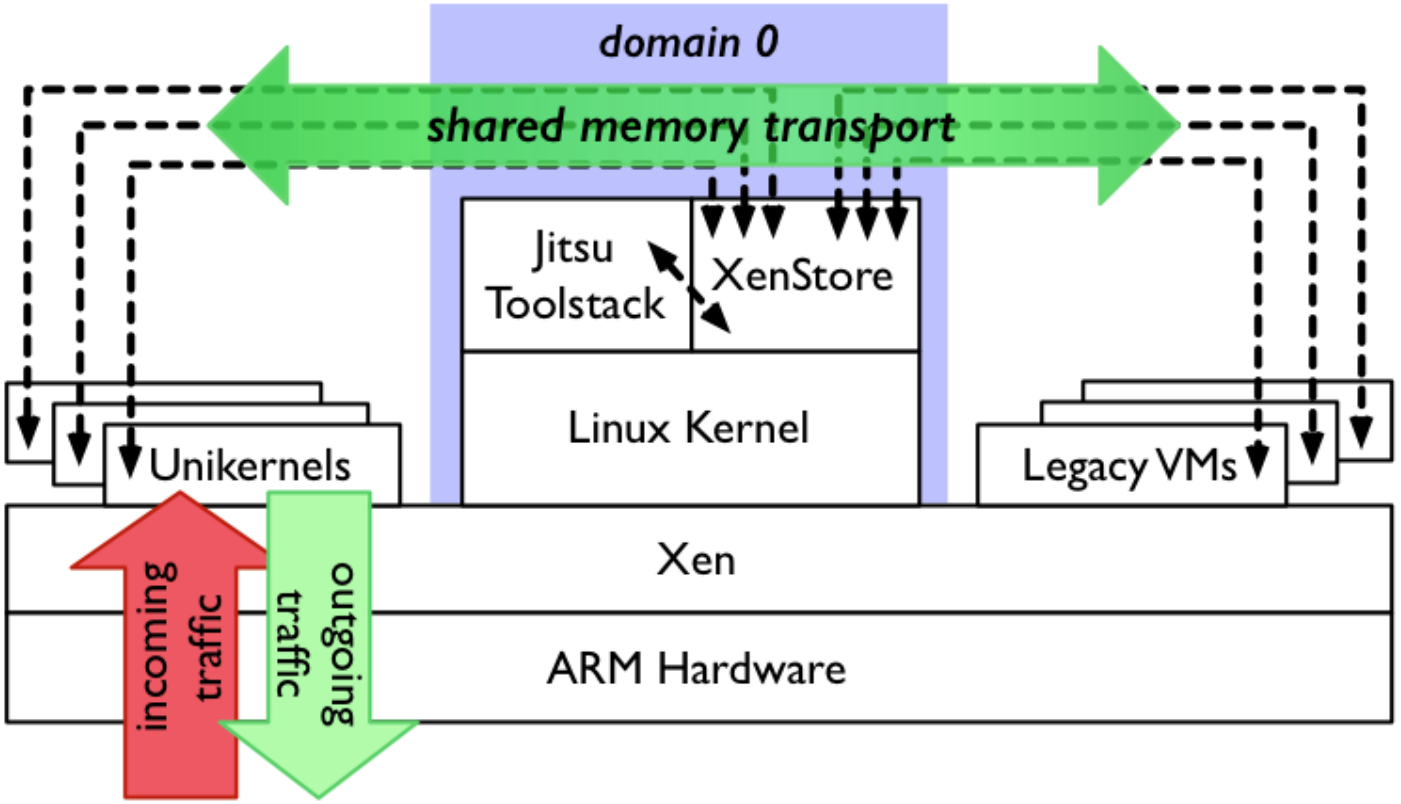


JUST-IN-TIME SUMMONING

A toolstack to launch unikernels on-demand with low latency:

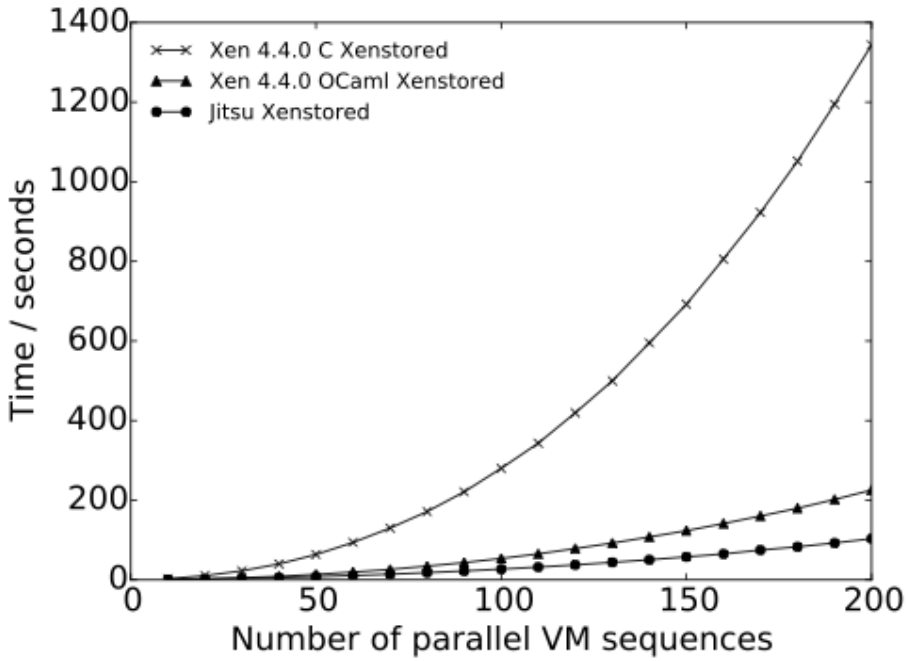
- **Performance improvements** to Xen's boot process & toolstack
  - *Are VMs fundamentally too slow for real-time launch?*
  - *Currently: 3-4s to boot a Linux VM on ARM*
- **Conduit**, shared-memory communication between unikernels
  - *Low-latency toolstack communications*
  - *Currently: loopback TCP over bridge*
- **Synjitsu** and the Jitsu Directory Service
  - *Launch services on-demand in real time*

JITSU ARCHITECTURE



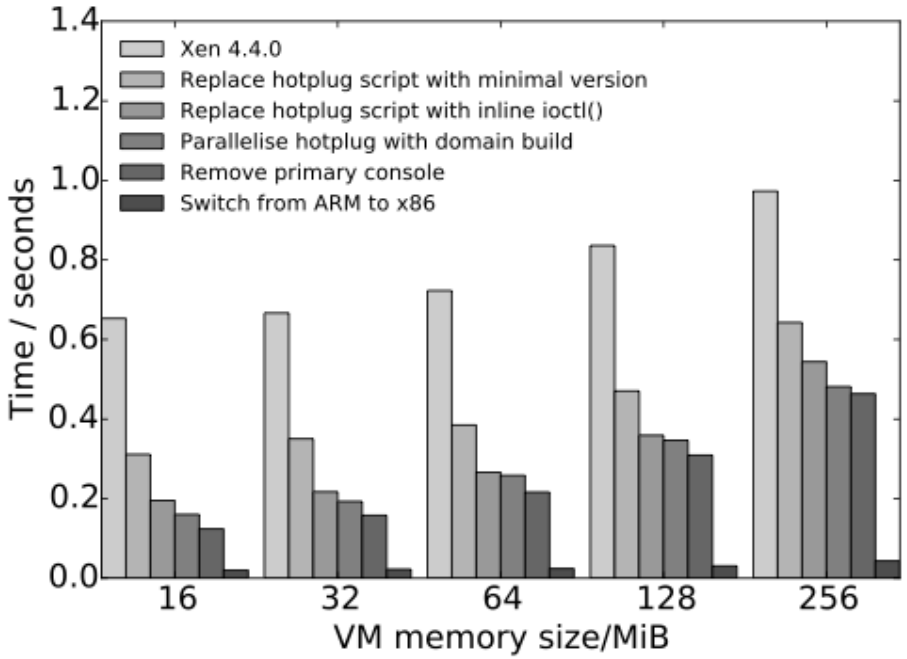
- Required a **new "MiniOS" for Xen/ARMv7** architecture.
  - *Removal of `libc` reduces attack surface and image size*
  - *Vast majority of networking code in pure OCaml*
- Xen PV driver model only – **no hardware emulation**
  - *ARM does not need all the legacy support of Xen/x86!*
- Much less CPU available, so need to optimise toolstack
  - *Linux VM takes 3-4s to boot on Cubieboard*

PARALLEL BOOT

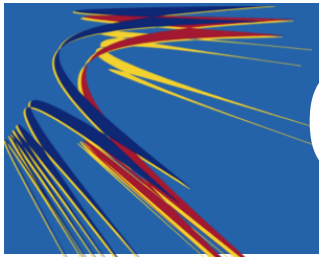


Improving inter-VM XenStore coordination database had scaling problems with concurrency conflicts, resolved via custom merge functions.

DESERIALISATION



Methodical elimination of forking crimes such as dom0 shell scripts



CONDUIT

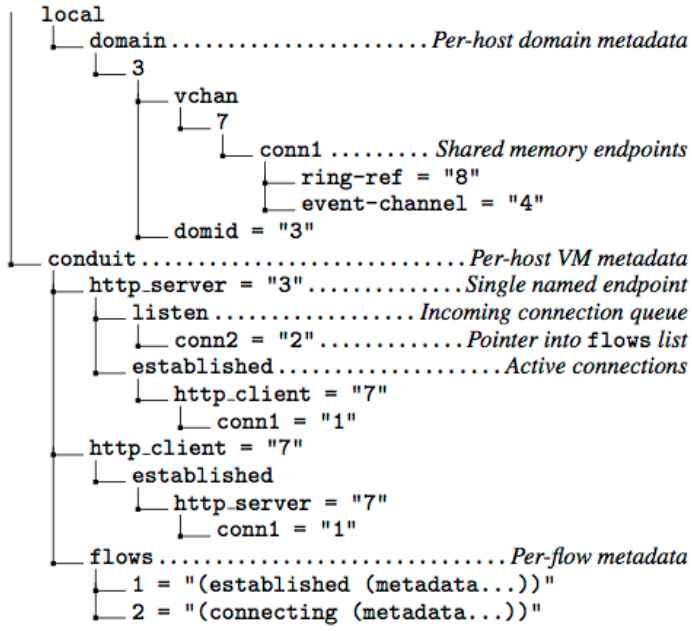
- Establishes **zero-copy shared-memory** pages between peers
  - Xen grant tables map pages between VMs (`/dev/gntmap`), synchronised via event channels (`/dev/evtchn`)
- Provides a **rendezvous facility** for VMs to discover named peers
  - Also supports unikernel and legacy VM rendezvous
- Hooks into higher-level **name services** like DNS
- Compatible with the **`vchan`** inter-VM communication protocol

Code: https://github.com/mirage/ocaml-conduit

RENDEZVOUS

- XenStore acts as an incoming connection queue
- Client requests are registered in a new `/conduit` subtree
- Client picks port and writes to the target `listen` queue
- Connection metadata (grant table, event channel refs) is written into `/local/domain/domid/vchan`

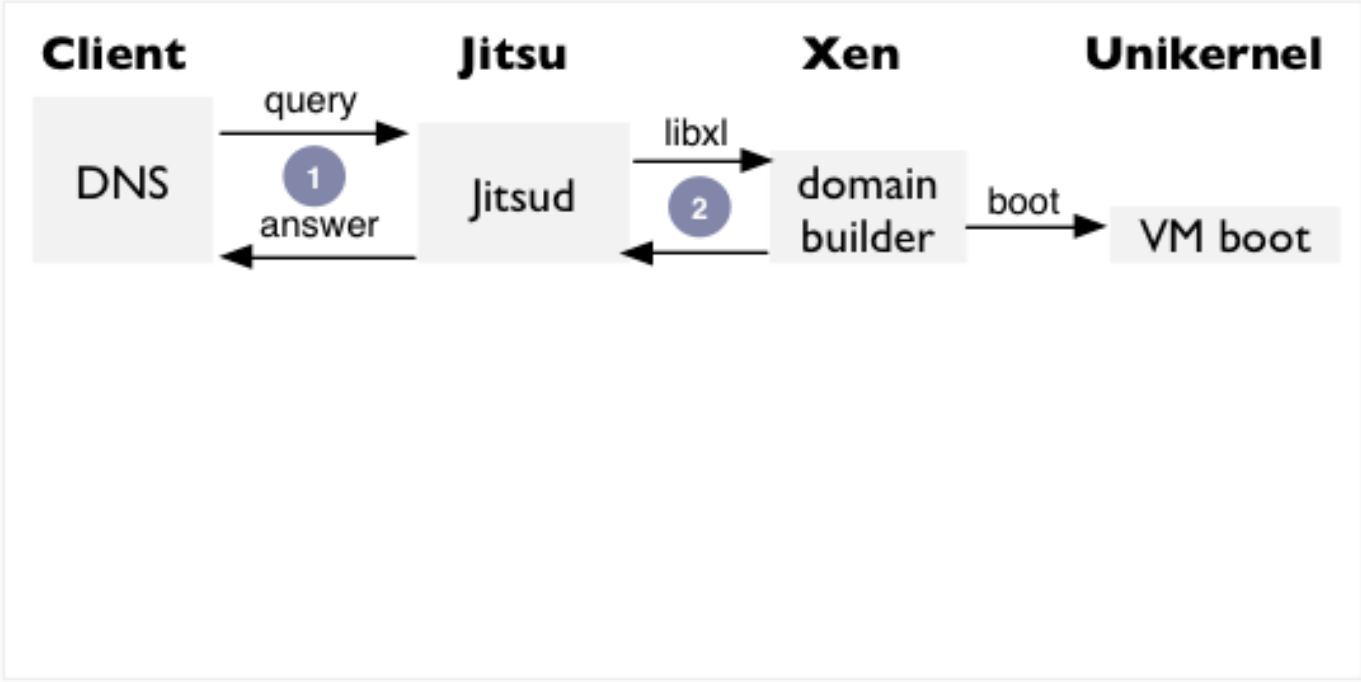
...and the data flows



Performs the role of Unix's `inetd`:

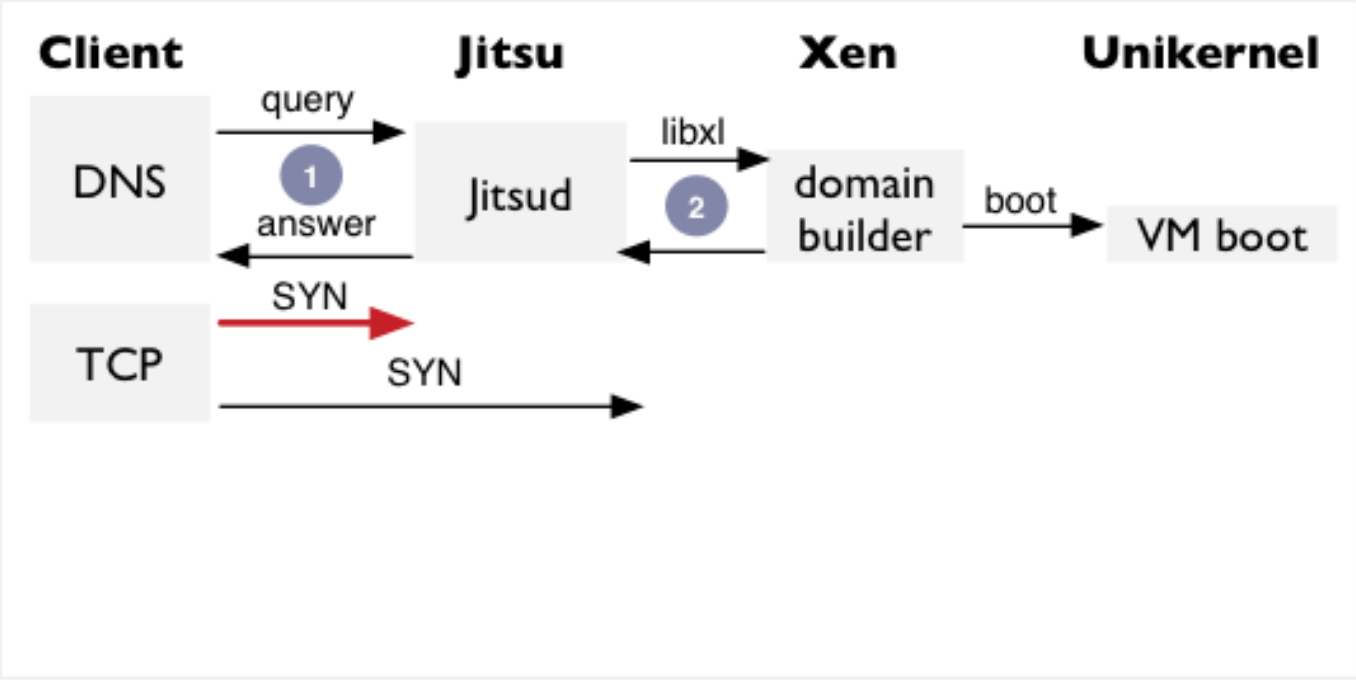
- Jitsu VM launches at boot time to handle name resolution (whether local via a well known `jitsud` Conduit node in XenStore or remote via DNS)
- When a request arrives for a live unikernel, Jitsu returns the appropriate endpoint
- If the unikernel is not live, Jitsu boots it, and acts as proxy until the unikernel is ready

MASKING BOOT LATENCY



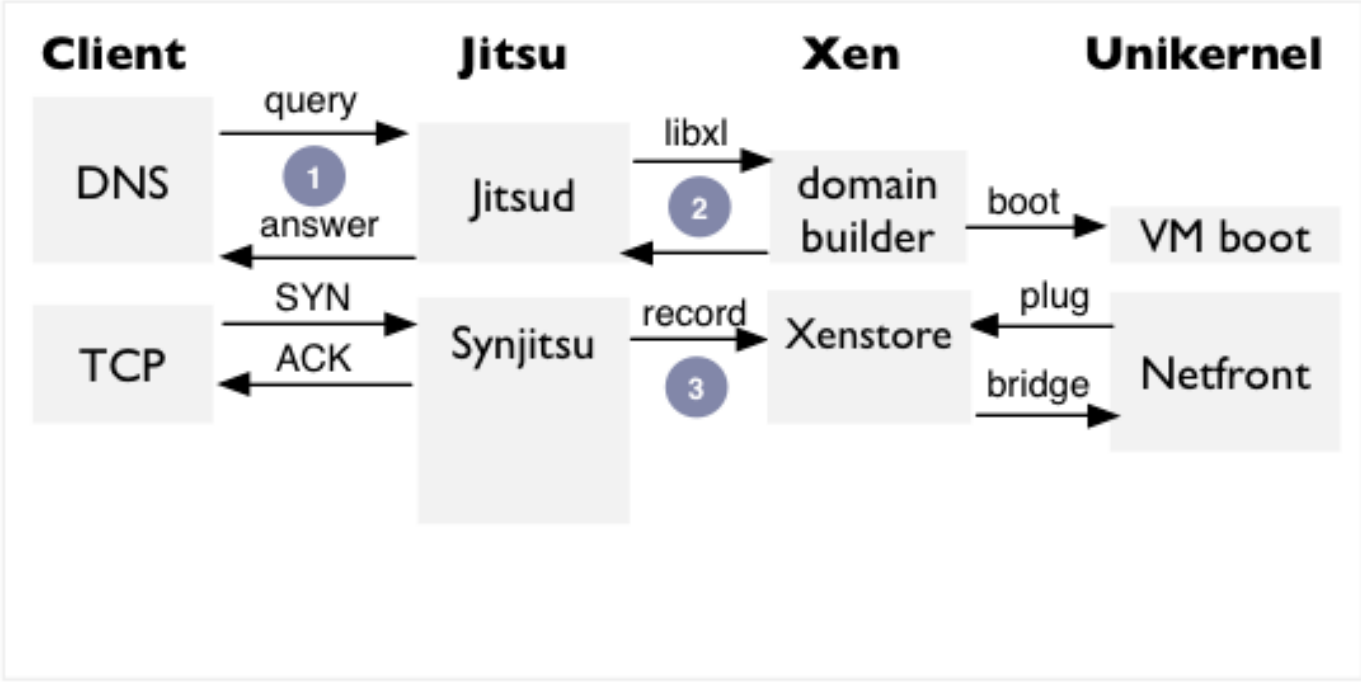
The Jitsu toolstack listens for DNS requests and boots the relevant unikernel and responds immediately.

MASKING BOOT LATENCY



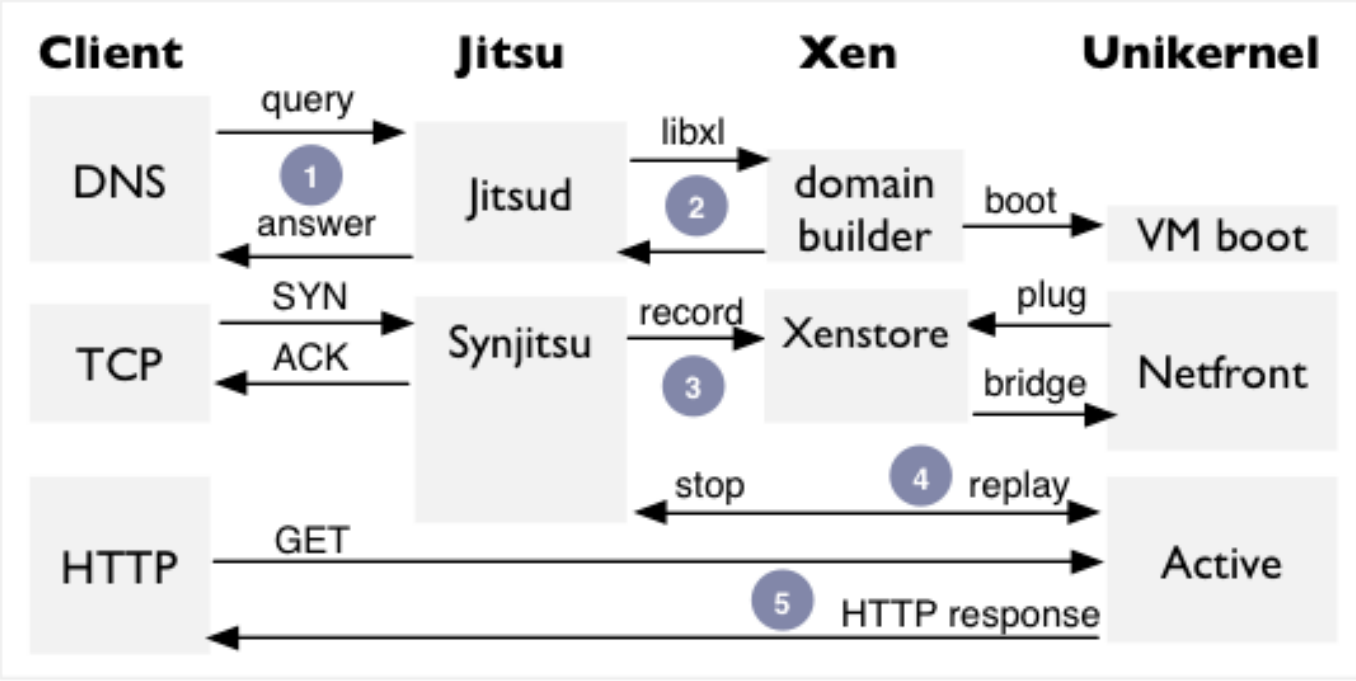
But a fast client might still lose a TCP SYN if unikernel isnt ready, thus causing SYN retransmits (slow!).

MASKING BOOT LATENCY



Synjitsu responds to requests and serialises connection state until VM is ready and network plugged in.

MASKING BOOT LATENCY

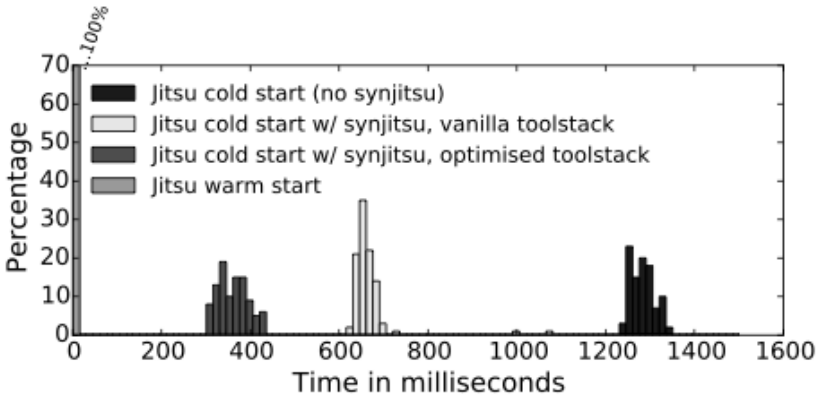


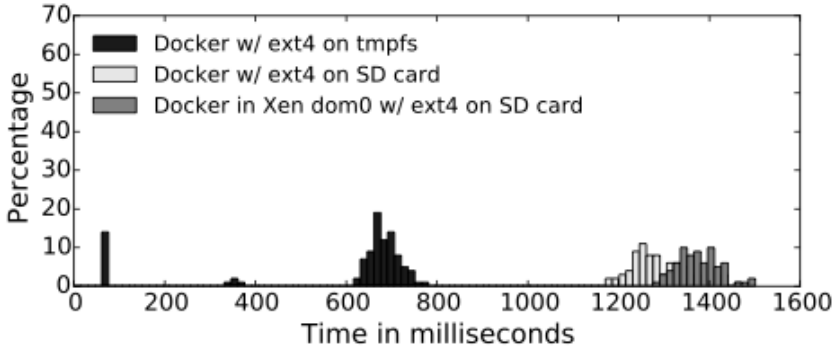
By buffering TCP requests into XenStore and then replaying, Synjitsu parallelises connection setup and unikernel boot

MASKING BOOT LATENCY

Jitsu optimisations bring boot latency down to **~30—45 ms** (x86) and **~350—400 ms** (ARM).

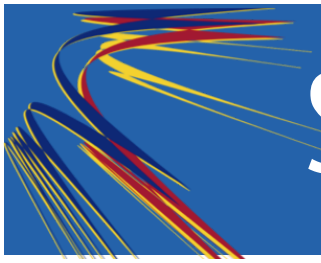
- Docker time was 1.1s (Linux), 1.2s (Xen) from an SD card
- Mounting Docker's volumes on an `ext4` loopback volume inside of a `tmpfs` reduced latency but often terminated early due to many buffer IO, `ext4` and `VFS` errors





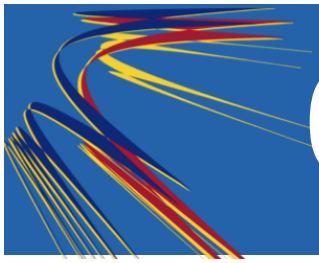
Walkthrough of the key functionality with and without Synjitsu:

https://www.dropbox.com/s/ra5qib321d53nfi/nsdi_screencast.mov



SUMMARY

- **Xen/ARM is here!** Good way to run embedded experiments.
  - *GitHub build scripts: mirage/xen-arm-builder*
  - *GitHub libraries: protocol code at openmirage.org*
  - Robust existing Xen tools all continue to work.
  - Jitsu optimises away a lot of latency at the edge.
- **No fundamental drawback to VMs vs containers**
  - Unikernels competitive with containers on embedded
  - Shipping out specialised type-safe code is practical
  - Not touching disk while booting further improves latency



ONGOING WORK

- **Multiprotocol Synjitsu**
  - Extend to the TLS handshake to pipeline secure connections
  - Add vanilla TCP load balancing support
- **Wide area redirection**
  - DNS proxy to redirect to cloud if ARM node is down
  - First ARM cloud hosting via Scaleway
- **More platforms**
  - Integrating rump kernels to boot without Xen
  - Working with UCN partners to provide home router platform for future deployments

A Linux Foundation Incubator Project lead from the University of Cambridge and Citrix Systems.

Featuring blog posts on new features by:

Amir Chaudhry, Thomas Gazagnaire, David Kaloper, Thomas Leonard, Jon Ludlam, Hannes Mehnert, Mindy Preston, Dave Scott, and Jeremy Yallop.

Thanks for listening! Questions?
Contributions very welcome at openmirage.org
Mailing list at mirageos-devel@lists.xenproject.org