JITSU: JUST-IN-TIME SUMMONING OF UNIKERNELS

CS 5523 – OPERATING SYSTEMS

Ruta Dandekar
(pcf266/@01614154)

Network Latency

- Network Latency is a delay that occurs in data communication over a network.
- There are many benefits of cloud hosting like, centralised management, high availability etc. but still it is prone to network latency.
- The physical separation between data centers imposes network latency.
- Thus, it can have impact on recent technologies like augmented reality (e.g. Google Glass) and voice control (e.g. Apple’s Siri).

How it can be resolved?

- It can be mitigated by using Just-In-Time Summoning of Unikernel (JITSU).
- It is a lightweight, multi-tenant isolated, etc. virtual machine (VM).
- It provides a service that launches unikernels in response to network traffic by limiting boot latency (few hundred milliseconds).

WHAT ARE UNIKERNELS?

- It is an executable image.
- It can be executed on a hypervisor without the need for separate operating system.
- It contains all the operating system functions required by the operations.
- It contains the collections of system libraries that contains all the core capabilities and application codes.
- They can boot and respond to traffic in real-time.

JITSU System.

- The widely deployed Xen hypervisor enforces isolation between multiple tenants sharing physical machines.
- It recently added support for building an embedded cloud system of distributed low-power devices, deployed near users, able to host applications delivering real-time services directly via local networks.
- Jitsu, a system for securely managing multi-tenant networked applications on embedded infrastructure.
- Jitsu re-architects the Xen toolstack by overcoming current limitations that prevent Xen from becoming an effective platform for building embedded clouds.

MirageOS

- In Jitsu they have used the open-source MirageOS written in OCaml, a statically type-safe language that has a low resource footprint and good native code compilers for both x86 and ARM.
- A particular advantage of using MirageOS when working with Xen is that all the toolstack libraries involved are written entirely in OCaml making it easier to safely manage the flow of data through the system and to eliminate code that would otherwise add overhead.

Xen/Arm Unikernels.

- Bringing up MirageOS unikernels on ARM required detailed work mapping the libOS model onto the ARM architecture.
- They have described :

  Booting MirageOS unikernels on ARM.

  Memory management requirements.

  Device virtualization support.

Booting MirageOS unikernels on ARM

- Xen Boot Library.: The first generation of unikernels such as MirageOS, OCaml, were constructed by forking *Mini-OS*, a tiny Xen library kernel that initializes the CPU, displays console messages and allocates memory pages. These embedded libraries are both security-critical (they run in the same address space as the type-safe unikernel code) and difficult to audit.
- Fast Booting on ARM. : They then ported Mini-OS to boot against the new Xen ARM ABI. The Xen domain builder allocates a fresh virtual machine descriptor, assigns RAM to it and loads the kernel at the offset 0x8000 (32KB). Execution begins with the r2 register pointing to a Flattened Device Tree (FDT). The FDT approach is much simpler than x86 booting, where it supports multiple modes (paravirtualized, hardware-assisted and hybrids).

Continued….

Some assembler code then performs basic boot tasks:

- Configuring the MMU, which handles mapping virtual to physical memory addresses.
- Turning on branch prediction.
- Setting up the exception vector table, defining how to handle interrupts and deal with various faults such as reading from an invalid memory address.
- Setting up the stack pointer and jumping into the Carch_init function for the remainder of execution. The final step is to jump into the OCaml code section and begin executing application logic.

Memory management

- Once the MirageOS/ARM unikernels has booted, it runs in a single address space without context switching. However, the memory layout under ARM is significantly different from that for x86.
- Under the ARM virtualization extensions, there are two stages to converting a virtual memory address (used by application code) to a physical address in RAM, both of which go through translation tables.
- The first stage is under the control of the guest VM, where it maps the virtual address to what the guest believes is the physical address – the Intermediate Physical Address (IPA).
- The second stage, under the control of Xen, maps the IPA to the real physical address.

Device Virtualization

- On Xen/x86 it is possible to add virtual devices by two means: pure PV devices that operate via a split-device model and emulated hardware devices that use the qemu device emulator to provide the software model.
- MirageOS includes OCaml library implementations of the Xen PV protocols for networking and storage

The Jitsu Toolstack

Jitsu is described in three phases, each of which progressively reduces end-to-end latency.

- The traditional Xen toolstack is serialised leading to large boot times due to long pauses between actual boot activity. They reduce these boot times by reducing this blocking behaviour and speeding up various boot components.
- They have described optimisation of the inter-VM communications protocol via conduits, to support direct shared memory communication between named endpoints. Conduits eliminate the need to use local networking to communicate between Jitsu and unikernels, further driving down latency.
- They have introduce the Synjitsu directory service that masks boot latency to external clients by handling the initial stages of TCP handshake.

Optimising Boot Times

- Xen’s domain builder creates the initial VM kernel image. Most of its work is to initialize and zero out physical memory pages, thus guests with less memory are naturally built more quickly.
- As unikernels require such small amounts of memory to boot (8MB is plenty), they have an advantage over modern Linux distributions which typically require at least 64MB and are often recommended 128MB or more.

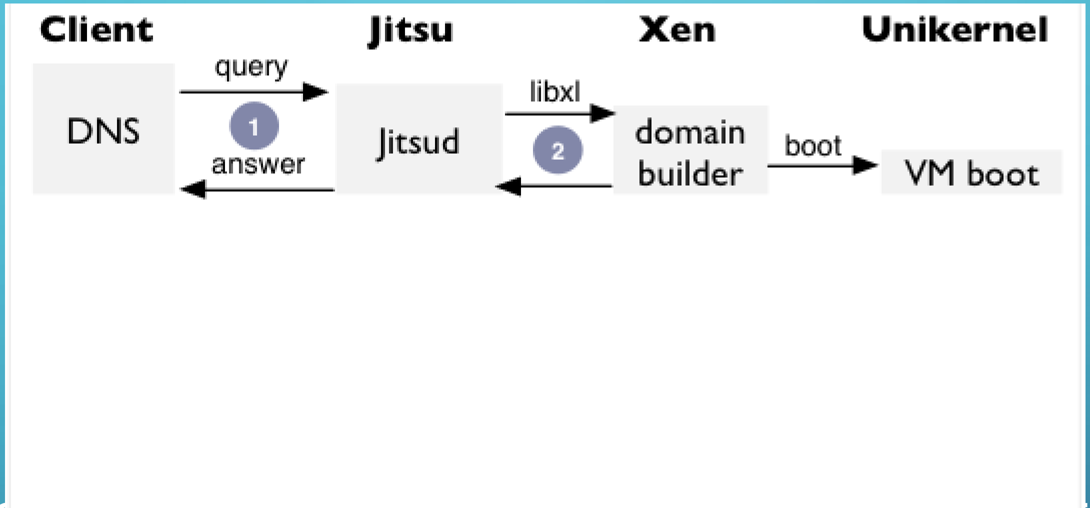
Communication Conduits

- It establishes zero-copy shared-memory pages between peers.
- It helps VMs to assembles in orrder to discover named peers.
- It hooks into higher level name services like DNS.
- Conduit is designed to be compatible with the vchan library for inter-VM communication.

JITSU Directory Service.

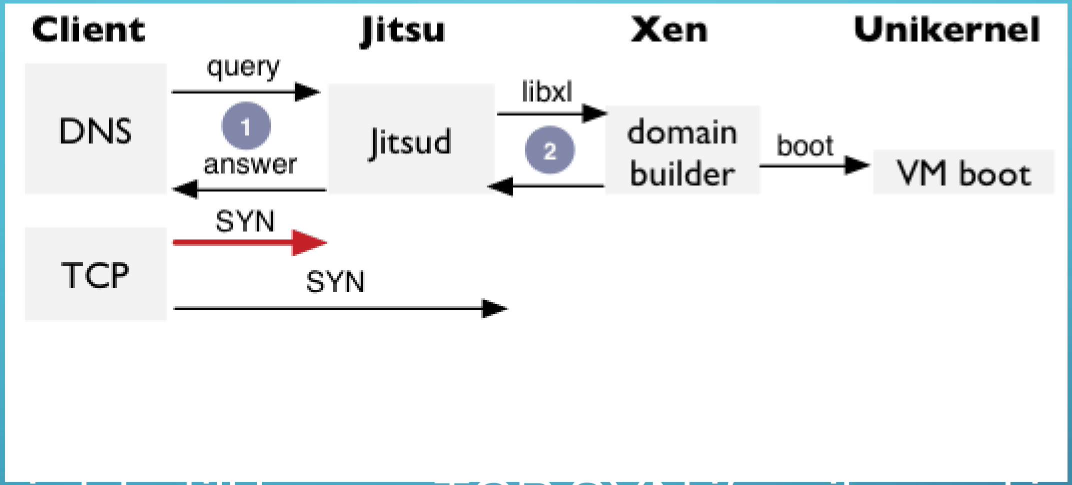
- The goal is to ensure that unikernels are launched and halted in real-time in response to network requests.
- Jitsu VM launches at boot time to handle name resolution.
- When a request arrives for a live unikernel, Jitsu returns the appropriate endpoint
- If the unikernel is not live, Jitsu boots it, and acts as proxy until the unikernel is ready

Masking Boot Latency



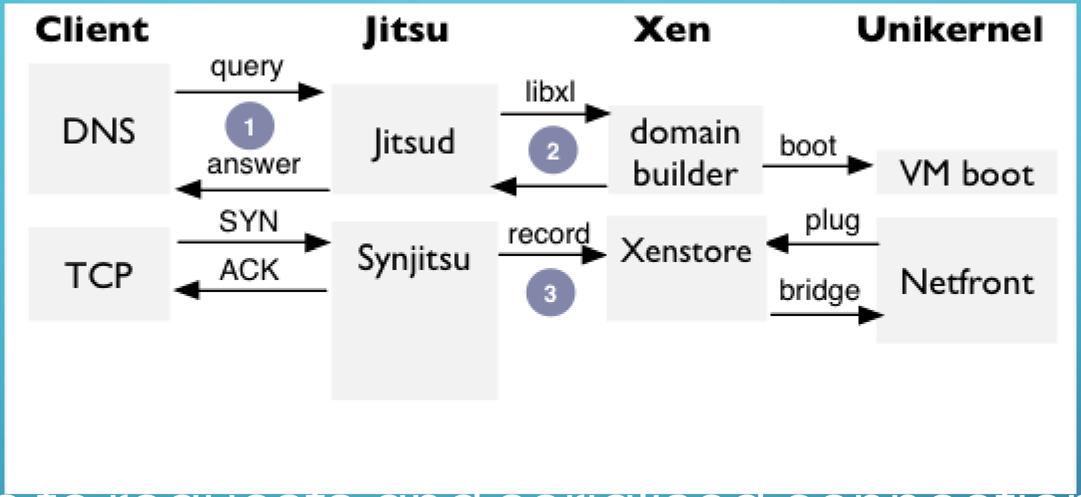
The Jitsu toolstack listens for DNS requests and boots the relevant unikernel and responds immediately.

Continued….



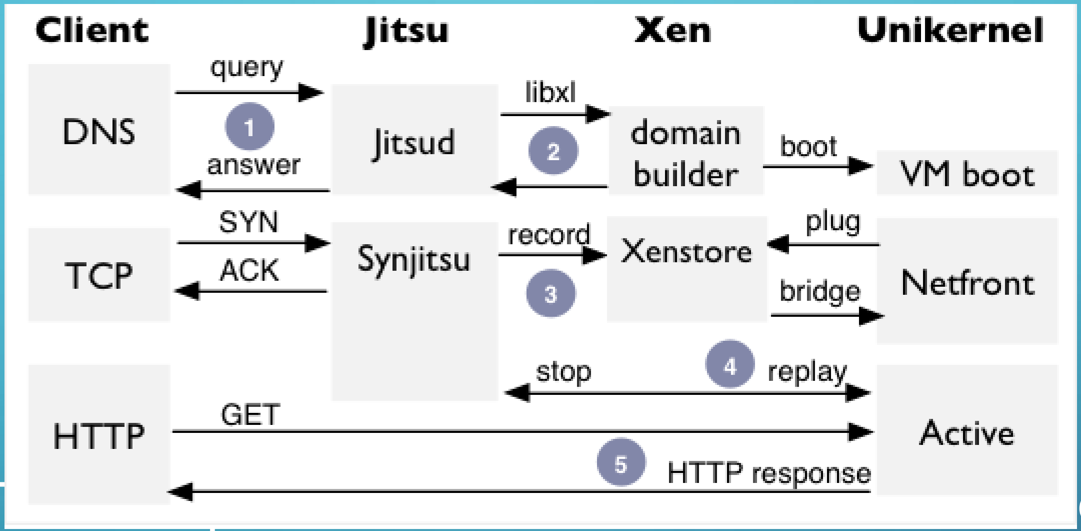
But a fast client might still lose a TCP SYN if unikernel isn't ready, thus causing SYN retransmits (slow!).

Continued….



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

Continued…



- By buffering T eplaying, Synjitsu parallelises connection setup and unikernel boot.
- Jitsu optimisations bring boot latency down to ~30—45 ms (x86) and ~350—400 ms (ARM).

Evaluation

- Throughput: They have previously carried out a fuller analysis of unikernel throughput with various protocol. They found that network throughput remains acceptable
- Datapath latency: Jitsu minimises excess bridging and latency.
- Power Usage: A key facet of their contribution is that by using ARM-based devices, power consumption is significantly reduced, to the extent that they become acceptable to run 24/7 in a domestic environment.
- Security: To evaluate end-to-end security properties they looked for critical security bugs eliminated by use of isolation via type-I hypervisor and memory-safe language to build minimal VM appliances.

Conclusion

They have presented Jitsu, a low latency toolstack for Xen/ARM that uses memory-safe unikernels to serve applications with significantly greater levels of isolation and security than currently achieved on modern embedded devices.

Thank You