Open-source, Lightweight, Extensible Hypervisor

ANUP PATEL <ANUP@BRAINFAULT.ORG>
Little About Me

• Hypervisor and Linux kernel developer with 12+ years of industry experience
• Post-graduated (Masters) in 2009 from IIT Bombay, India
• Work full-time for Qualcomm as Server virtualization expert
• Maintain Xvisor as hobby project in personal time (since 2010)
• Open source contributions:
  • 3300+ patches in Xvisor (http://xhypervisor.org/)
  • 100+ patches in Linux ARM/ARM64/RISC-V (https://www.kernel.org/)
  • 24+ patches in Linux KVM ARM64 (https://www.kernel.org/)
  • 16+ patches in Atomthreads RTOS (https://atomthreads.com/)
  • Few patches in Xen ARM, QEMU, KVMTOOL, etc
Agenda

- Overview
- Virtualization Infrastructure
- Domain Isolation
- Device Virtualization
- Domain Messaging
- Footprint
- Xvisor for Automotive
- References

**NOTE:** Domains in Automotive world are referred to as **Guests** in Xvisor
Overview
What is Xvisor?

• **XVISOR** = eXtensible Versatile hypervisor

• Xvisor is an open-source GPLv2 Type-1 monolithic (i.e. Pure Type-1) hypervisor

• Community driven open source project  
  (http://xhypervisor.org, xvisor-devel@googlegroups.com)

• 8+ years of development and hardening (since 2010)

• Supports variety of architectures: ARMv5, ARMv6, ARMv7, ARMv7ve, ARMv8, x86_64, and RISC-V (work-in-progress)

• First paper in IEEE PDP 2015 titled “*Embedded Hypervisor Xvisor: A comparative analysis*”
Hypervisor Classification - Traditional

Type 1 Examples: Xvisor, Xen, VMWare ESX Server, Microsoft HyperV, OKL4 Microvisor, etc

Type 2 Examples: Linux KVM, FreeBSD Bhyve, VMWare Workstation, Oracle VirtualBox, etc
Xvisor - Complete Monolithic - Type 1

Xvisor: Extensible Versatile Hypervisor

- Host Hardware
  - Device Drivers (Host HW Access)
  - Host Virtualization
  - Management Terminal
  - VCPUs
  - Orphan VCPUs
  - IOMMU
  - Network, Block, Input, ... Virtualization

- Xvisor Hypervisor

- Guest User Space
  - Guest Kernel
  - VirtIO Frontends

- Guests
  - Guest0
  - Guest1
  - GuestN

- EL0-NS (ARM64)
- EL1-NS (ARM64)
- EL2-NS (ARM64)
Lots of features

• Virtualization Infrastructure:
  • Device tree based configuration
  • Soft real-time pluggable scheduler
  • Hugepages for Guest and Host
  • Tickless and high-resolution timekeeping
  • Host device driver framework
  • Threading framework
  • Runtime loadable modules
  • Management terminal
  • Light-weight filesystem
  • White-box testing
  • ... Many More ...
Lots of features (Contd.)

• **Domain Isolation:**
  • VCPU and Host Interrupt Affinity
  • Spatial and Temporal Memory Isolation

• **Device Virtualization:**
  • Pass-through device support
  • Block device virtualization
  • Network device virtualization
  • Input device virtualization
  • Display device virtualization
  • VirtIO v0.9.5 for Para-virtualization

• **Domain Messaging:**
  • Sharing On-chip Coprocessor
  • Zero-copy Inter-Guest Communication
Virtualization Infrastructure
Three types of device tree (DT):

1. **Host DT:**
   - Device tree which describes underlying host HW to Xvisor
   - Used by Xvisor at boot-time

2. **Guest Xvisor DT:**
   - Device tree which describes Guest virtual HW to Xvisor
   - Used by Xvisor to create Guest

3. **Guest OS DT:**
   - Device tree which describes Guest virtual HW to Guest OS
   - Used by Guest OS at boot-time
Soft Real-time Pluggable Scheduler

- Scheduling entity is a VCPU

- Two types of VCPUs:
  1. **Normal VCPU**: A VCPU belonging to Guest/VM
  2. **Orphan VCPU**: A VCPU belonging to Hypervisor for background processing

- Orphan VCPUs are very light-weight compared to Normal VCPUs

- Scheduler supports **pluggable scheduling policy**, available policies:
  - Fixed priority round-robin
  - Fixed priority rate monotonic

- Scheduling policies are soft real-time

- Scheduler supports multi-processors (or SMP Host)
### Hugepages for Guest and Host

- Xvisor uses Stage1 (regular) page table for “Hypervisor virtual address” to “Host physical address” mappings
- Host hugepages are bigger mappings in Stage1 (regular) page table
- **Host hugepages make Xvisor memory accesses faster**
- Xvisor uses Stage2 (nested) page table for “Guest physical address” to “Host physical address” mappings
- Guest hugepages are bigger mappings in Stage2 (nested) page table
- **Guest hugepages make Guest OS memory accesses faster**
- For ARM64 and x86_64, hugepage sizes are 2M and 1G
Domain Isolation
VCPU and Host Interrupt Affinity

• **VCPU affinity is an attribute of VCPU** specifying Host CPUs on which it is allowed to run

• Using VCPU affinity, we can assign particular Host CPUs for:
  • Guest VCPUs (Normal VCPUs)
  • Xvisor background threads (Orphan VCPUs)

• **Host interrupt affinity is an attribute of Host interrupt** specifying Host CPUs on which it can be processed

• Using Host interrupt affinity, we can assign particular Host CPUs for Host interrupts of a Guest pass-through device

• Host interrupt affinity of per-CPU Host interrupts (such as IPIs) cannot be changed
Spatial and Temporal Memory Isolation

- **Spatial memory isolation** achieved using cache-coloring on last level cache for Guest RAM
- **Temporal memory isolation** achieved using CPU performance monitoring unit (PMU) to control memory access rate by Guest

IEEE ICIT 2018 paper: “Supporting Temporal and Spatial Isolation in a Hypervisor for ARM Multicore Platforms”
Device Virtualization
Device Virtualization Types

• Types of Virtual Devices:
  1. **Software Emulated Device:** Real-world device emulated by hypervisor. **Examples**, Emulated UART 8250, etc.
  2. **Paravirtual Device:** Pseudo-devices emulated by hypervisor which are designed to minimize register programming. **Examples**, VirtIO Net, VirtIO Block, VirtIO Console etc.
  3. **Pass-through Device:** Direct access of host device from Guest/VM. This requires IOMMU support in Host. **Examples**, PCI e1000 network adapter accessed by Guest/VM, SATA AHCI controller accessed by Guest/VM, etc.
  4. **Partial Pass-through Devices:** Access part of a host device from Guest/VM. This requires IOMMU support in Host and Host device should have special support. **Examples**, SRIOV based PCI Network Adapter, GPU with multiple channels, etc.

• All types of virtual devices supported by Xvisor
Pass-through Device Support

- Linux compatibility headers for drivers running in Xvisor
- IOMMU and Interrupt controller virtualization for drivers running in Guest OS
- Access part of device from Guest OS using partial pass-through:
  - Custom driver
  - Custom emulator
Block Device Virtualization

• Consist of:
  1. **vdisk**: Logical entity which gets block read/write requests from Guests. Examples, Storage device emulators, and VirtIO Block backends.
  2. **blockdev**: Logical entity which represents host storage device or a partition on host storage device. Examples, MMC, NAND, SATA, etc.

• **We can attach a blockdev to a vdisk**

**Diagram Description**
- **INTEGRITY/QNX**
  - VirtIO Block Frontend
  - Guest0

- **Android/AGL**
  - VirtIO Block Frontend
  - Guest1

- **RTOS**
  - VirtIO Block Frontend
  - GuestN

- **Device Drivers (Host HW Access)**
- **Disk Virtualization Framework**
- **Host Block Driver (blockdev)**
- **Orphan vCPUs**
- **Xvisor Hypervisor**
- **CPU Virtualization**
- **Host Hardware**
- **Storage Device**

**Diagram Key**
- **Host Hardware**: Physical components
- **Storage Device**: Media for data storage
- **Device Drivers**: Interface for host hardware access
- **Disk Virtualization Framework**: Converts block operations
- **Host Block Driver (blockdev)**: Controls block device access
- **Orphan vCPUs**: Unassigned virtual CPUs
- **Xvisor Hypervisor**: Core virtualization layer
- **CPU Virtualization**: Processor virtualization

**Diagram Notes**
- **Consist of**
  - **vdisk**: Logical entity
    - **Logical entity which gets block read/write requests from Guests**
      - Examples: Storage device emulators, VirtIO Block backends
  - **blockdev**: Logical entity
    - **Logical entity which represents host storage device or a partition on host storage device**
      - Examples: MMC, NAND, SATA, etc.
- **We can attach a blockdev to a vdisk**
Network Device Virtualization

• Consist of:
  1. **netport**: Logical entity capable of consuming and generating packets. Examples host network drivers, NIC emulators, and VirtIO Net backends.
  2. **netswitch**: Logical entity which does packet routing between netports. Various routing policy available: hub, bridge, vlan, etc.

- **Xvisor**: Extensible Versatile Hypervisor

- **Host Hardware**

- **INTEGRITY/QNX**
  - Guest0
  - VirtIO Net Frontend
  - INTEGRITY/QNX

- **Android/AGL**
  - Guest1
  - VirtIO Net Frontend
  - Android/AGL

- **RTOS**
  - GuestN
  - VirtIO Net Frontend

- **Device Drivers (Host HW Access)**

- **Network Virtualization Framework (netswitch)**

- **Host Network Driver (netport)**

- **Orphan vCPUs**

- **CPU Virtualization**

- **Xvisor Hypervisor**
Domain Messaging
Sharing On-chip Coprocessor

- SOCs can have on-chip coprocessors for secured processing
- Linux applications can communicate with on-chip coprocessor using RPMSG character device
- Virtual messaging domain to define a set Guests allowed to communicate with on-chip coprocessor
Zero-copy Inter-Guest Communication

• Achieved using:
  1. **VirtIO RPMSP:**
     • Used for control messages
     • Name-service notifications
  2. **Shared Memory:**
     • Used for actual data transfers
     • Very fast zero-copy
• Linux applications can communicate across Guests using **RPMSG** character device
• Virtual messaging domain to define a set of Guests allowed to communicate
Footprint
# Code Size and Memory Footprint

<table>
<thead>
<tr>
<th>Lines of Code</th>
<th>Comments</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/arm/</td>
<td>7143</td>
<td>20614</td>
</tr>
<tr>
<td>core/</td>
<td>8974</td>
<td>35419</td>
</tr>
<tr>
<td>commands/</td>
<td>1025</td>
<td>10145</td>
</tr>
<tr>
<td>daemons/</td>
<td>147</td>
<td>526</td>
</tr>
<tr>
<td>drivers/</td>
<td>9427*</td>
<td>43922*</td>
</tr>
<tr>
<td>emulators/</td>
<td>3217*</td>
<td>24963*</td>
</tr>
<tr>
<td>libs/</td>
<td>5933</td>
<td>17445</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>35866</strong></td>
<td><strong>153034</strong></td>
</tr>
</tbody>
</table>

* Can be further decreased or increased based on compile-time configuration

<table>
<thead>
<tr>
<th>BLOB</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>.text</td>
<td>969 KB*</td>
</tr>
<tr>
<td>.data</td>
<td>129 KB</td>
</tr>
<tr>
<td>.rodata</td>
<td>329 KB*</td>
</tr>
<tr>
<td>.bss</td>
<td>202 KB</td>
</tr>
<tr>
<td>vmm.bin</td>
<td>1445 KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Runtime Memory</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text memory freed at boot-time</td>
<td>112 KB</td>
</tr>
<tr>
<td>Typical memory usage</td>
<td>21 MB*</td>
</tr>
<tr>
<td>Max VAPOOL limit</td>
<td>32 MB*</td>
</tr>
</tbody>
</table>

**NOTE:** Stats gathered from Xvisor-next on 22nd September 2018 for ARM64
Xvisor for Automotive
Why Xvisor is ideal for Automotive?

• No dependency on any Guest OS for running management tools
• Single software providing complete virtualization solution
• Guest types described using device tree instead of fixed Guest types
• Para-virtualization complying open-standards (such as VirtIO)
• Pass-through (or direct access) device support
• Zero-copy inter-guest communication
• Spatial and temporal memory isolation between Guests
• Low memory footprint with reasonable code size
• Playground for academic research
On-going Work in Xvisor

- Guest image authentication
- VirtIO input
- VirtIO GPU
- Netport Rx/Tx throttling
- Vdisk IO request rate-limiting
- Fixed priority deadline scheduler
- RISC-V support
- Upgrade to VirtIO 1.0 support
- ... And other stuff ...
References
References

• *Embedded Hypervisor Xvisor: A comparative analysis (IEEE PDP 2015)*
  (http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7092793)

• *Xvisor: An open-source, lightweight, embedded hypervisor for your car (FOSDEM 2015)*
  (https://archive.fosdem.org/2015/schedule/event/car_hypervisor/)

• *Xvisor VirtIO CAN: Fast virtualized CAN (ERTS 2016)*
  (http://xhypervisor.org/pdf/Xvisor_VirtIO_CAN_Fast_virtualized_CAN.pdf)

• *Supporting Temporal and Spatial Isolation in a Hypervisor for ARM Multicore Platforms (IEEE ICIT 2018)*
  (https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8342303)

• *Reconciling Security with Virtualization: A Dual-Hypervisor Design for ARM TrustZone (IEEE ICIT 2018)*
  (https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8342303)
Thank You !!!
Backup
Hypervisor Classification - New

- Design of a hypervisor can be further classified based on three aspects:
  1. **CPU virtualization**
     - What part of hypervisor virtualize CPU registers and MMU?
  2. **Host hardware access**
     - What part of hypervisor access host devices (i.e. Host device drivers)?
  3. **Guest IO emulation**
     - What part of hypervisor virtualize peripherals (i.e. Guest I/O devices)?

**IEEE PDP 2015 paper:** “Embedded Hypervisor Xvisor: A comparative analysis”
## Hypervisor Classification - New (Contd.)

<table>
<thead>
<tr>
<th>Complete Monolithic</th>
<th>Partially Monolithic</th>
<th>Micro-Kernelized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single software layer</td>
<td>Extends an existing OS kernel</td>
<td>Micro-kernel providing virtualization</td>
</tr>
</tbody>
</table>

**Main Hypervisor:**
- Host hardware access
- CPU virtualization
- Guest IO emulation

**Remaining Stuff:**
- Optional host hardware access from virtual machine(s)

**Type-1**
- Examples: Xvisor, VMware ESX server

**Type-2**
- Examples: Linux KVM, FreeBSD Bhyve, VMware Workstation, Oracle VirtualBox

**Main Hypervisor:**
- Host hardware access
- CPU virtualization in host OS

**Remaining Stuff:**
- Optional host hardware access from virtual machine(s)
- Guest IO emulation from user-space software

**Main Hypervisor:**
- Basic host hardware access
- CPU virtualization in hypervisor micro-kernel

**Remaining Stuff:**
- Complete host hardware access in management virtual machine(s)
- Guest IO emulation in management virtual machine(s)

**Type-1**
- Examples: Xen, Microsoft HyperV, OKL4 Microvisor
Xen - Micro-kernelized - Type1

Dom0 User Space
Xen Toolstack (Management) + QEMU (Guest IO Emulation)

Dom0 Kernel
Device Drivers (Host HW Access) Xen PV Backends

Dom0 (Guest0 - Control)

DomU User Space

DomU Kernel
Xen PV Frontends

Dom1 (Guest1)

DomU User Space

DomU Kernel
Xen PV Frontends

DomN (GuestN)

CPU Virtualization
Xen Hypervisor
Basic Host HW Access

Host Hardware

EL0-NS (ARM64)
EL1-NS (ARM64)
EL2-NS (ARM64)

XVISOR: EXTENSIBLE VERSATILE HYPERVISOR

36
KVM - Partially Monolithic - Type2

- **Host User Space**
  - Process1
  - ProcessN
  - Host Hardware
  - Device Drivers (Host HW Access)
  - Linux Kernel (Host Kernel)
  - KVM Module (CPU Virtualization)
  - KVM Module Low-Visor (Only for ARM/ARM64)

- **Guest User Space**
  - Guest Kernel
  - Guest0
  - QEMU/KVMTOOL (Guest IO Emulation + VirtIO Backends)
  - VirtIO Frontends

EL0-NS (ARM64)
EL1-NS (ARM64)
EL1-NS (ARM64 non-VHE)
EL2-NS (ARM64 VHE)
EL2-NS (ARM64)

- **Hypervisor Component**