ACRN™: A Big Little Hypervisor for IoT Development

Yu Wang, Intel Open Source Technology Center

Key contributors: Anthony Xu; Jason Chen; Eddie Dong; Bing Zhu; Jack Ren; Hao Li; Kevin Tian
Table of Contents

PART 1: ACRN Overview

PART 2: Security in ACRN

PART 3: Rich I/O Mediation

PART 4: Call for Participation
ACRN™ is a Big Little Hypervisor for IoT Development

ACRN™ is a flexible, lightweight reference hypervisor, built with real-time and safety-criticality in mind, optimized to streamline embedded development through an open source platform.
ACRN Features

- **Small Footprint**
  - Optimized for resource constrained devices

- **Real Time**
  - Low latency
  - Enables faster boot time

- **Built for Embedded IoT**
  - Rich set of I/O mediators to share devices across multiple VMs

- **Adaptability**
  - Multi-OS support for guest systems like Linux and Android

- **Open Source**
  - Permissive BSD licensing

- **Safety Criticality**
  - Project is built with safety critical workload considerations in mind
Virtualization User Cases for IOT

In-Vehicle-Infotainment

Robotics

Precision instrument

Industrial
Architecture Overview

Service VM
- VM Manager
- ACRN Device Model (Mediators)

Linux Guest VM
- User
- Kernel
- virtio FE Drivers
- Keystore

Android Guest VM
- Normal World
  - User
  - Kernel
  - virtio FE Drivers
  - Keystore
- Secure World
  - User
  - Kernel
  - Keystore

VMX non-root operation
- Native Device Driver
- VMX
- VT-d
- EPT
- Hypercalls
  - VM API
  - Trusty API
- ACRN Hypervisor
  - vPIC/vLAPIC/vMSI

VMX root operation
- CSE
- SOC Platform (Apollo Lake etc.)
- Firmware (UEFI, SlimBoot etc.)
ACRN as a Device Hypervisor

- Small footprint

<table>
<thead>
<tr>
<th></th>
<th>KVM</th>
<th>Xen</th>
<th>ACRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>17M</td>
<td>290K</td>
<td>25K</td>
</tr>
</tbody>
</table>

- BSD licensee
- Be able to cherry pick piece of codes into OSV/OEM’s own hypervisor
- Verified boot
- Rich I/O mediators

<table>
<thead>
<tr>
<th>GPU</th>
<th>IPU</th>
<th>CSE</th>
<th>USB</th>
<th>Audio</th>
<th>Ethernet</th>
<th>Block</th>
<th>IOC</th>
<th>Touch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mediated Passthru</td>
<td>Virtio</td>
<td>Virtio</td>
<td>Emu.</td>
<td>Virtio</td>
<td>Virtio</td>
<td>Virtio</td>
<td>Emu.</td>
<td>Virtio</td>
</tr>
</tbody>
</table>
Verified Boot Sequence with SBL

- CSE verifies SBL
- SBL verifies ACRN & SOS Kernel
- SOS kernel verifies DM & vSBL thru dm-verity
- vSBL starts the guest side verification process (reusing the Android verified boot mechanism)

- NOTE: Each user VM has a DM APP instance in SOS
Verified Boot Sequence with UEFI

- UEFI verifies ACRN & OS Bootloader & SOS Kernel
- SOS kernel verifies DM and vSBL thru dm-verity
- vSBL starts the guest side verified boot process

- NOTE: ACRN remains EFI runtime services and boot time services (without interrupt)
Trusty OS virtualization

- Trusty OS is Google released OS for Android secure world which designed to execute in ARM TrustZone mode.

- ACRN hypervisor provide vCPU with different contexts for normal world and secure world. The android OS and Trusty OS can trigger the world switch through hypercall.

- ACRN hypervisor also maintain two EPT tables for different worlds. The secure world memory is invisible for normal world, but not vice versa.
HECI emulator implements a virtio PCIe device to support multiple User OS.

HECI BE will communicate with HECI FE driver to send & receive the HECI messages.

HECI client layer protocol will read/write to SOS MEI cdev directly. And HECI bus messages will emulate in the BE.

*MEI: Intel Management Engine Interface Linux driver; mei_cl_driver: mei client driver
SEED Virtualization

- HV gets pSEED from SBL, which retrieves from CSE through HECI
- Hypervisor implements Key derivation function (HKDF-256) to generate child seeds (vSEED) per request
- Present the derived vSEED to guest VM. Each guest cannot see/derive the other guest’s vSEED
• **IOC (IO controller)** is a bridge of SoC to communicate with Vehicle Bus. It routing of Vehicle Bus signals (for example, extracted from CAN messages) from IOC to the SoC and back, as well as controlling the onboard peripherals from SoC.

• **SOS** owns IOC, but **UOS** may access part features

• **Whitelisted CMDs** from **UOS** may be forwarded / emulated

• Support **Intel IOC controller** only, OEMs may extend
xHCl emulator provides multiple instances of virtual xHCl controller to share among multiple User Oss, each USB port can be dedicatedly assigned to a VM.

xDCI controller can be passed through to the specific user OS with I/O MMU assistance.

DRD device model emulate the APL PHY MUX control logic. The frontend re-use the native Intel USB role driver directly which provides sysfs interface to user space of user OS to switch DCI/HCI role in CarPlay SW.
Other mature I/O mediator

- **Standard virtio devices**
  - virtio storage
  - virtio network
  - virtio console
  - virtio input

- **GPU virtualization**
  - base on Intel Open Source GVT-g technology
<table>
<thead>
<tr>
<th>Area</th>
<th>v0.2@Q2'18</th>
<th>v0.5@Q3'18</th>
<th>V0.8@Q4'18</th>
<th>V1.0@Q1'19</th>
<th>V1.x@2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW</td>
<td>APL NUC (UEFI)</td>
<td>APL NUC (UEFI)</td>
<td>APL NUC (UEFI)</td>
<td>APL NUC (UEFI)</td>
<td>APL NUC (UEFI)</td>
</tr>
<tr>
<td></td>
<td>KBL NUC (UEFI)</td>
<td>KBL NUC (UEFI)</td>
<td>KBL NUC (UEFI)</td>
<td>KBL NUC (UEFI)</td>
<td>KBL NUC (UEFI)</td>
</tr>
<tr>
<td></td>
<td>APL UP2 (UEFI)</td>
<td>APL UP2 (UEFI)</td>
<td>APL UP2 (UEFI)</td>
<td>APL UP2 (UEFI)</td>
<td>APL UP2 (UEFI)</td>
</tr>
<tr>
<td>Hypervisor</td>
<td>VT-x</td>
<td>Virtio (v1.0)</td>
<td>32bit guest</td>
<td>vHost</td>
<td>ARM</td>
</tr>
<tr>
<td></td>
<td>VT-d</td>
<td>Power Management (Px/Cx)</td>
<td>Guest Real mode</td>
<td>Basic Realtime</td>
<td>Advanced Realtime</td>
</tr>
<tr>
<td></td>
<td>CPU static-partitioning</td>
<td>VM management</td>
<td>Android as guest</td>
<td>Power Management</td>
<td>Windows as guest</td>
</tr>
<tr>
<td></td>
<td>memory partitioning</td>
<td>ACRN debugging tool</td>
<td>MISRA C compliance</td>
<td>(S3/S5)</td>
<td>vxWorks as guest</td>
</tr>
<tr>
<td></td>
<td>Virtio (v0.95)</td>
<td>vSBL</td>
<td>Trusty (Security)</td>
<td></td>
<td>SGX (Security)</td>
</tr>
<tr>
<td></td>
<td>VHM</td>
<td>AliOS as guest</td>
<td>SBL boot *</td>
<td></td>
<td>Functional Safety</td>
</tr>
<tr>
<td></td>
<td>EFI boot</td>
<td>Zephyr as guest</td>
<td></td>
<td></td>
<td>compliance</td>
</tr>
<tr>
<td></td>
<td>ClearLinux as guest</td>
<td>Logical partitioning without Service OS</td>
<td></td>
<td></td>
<td>CPU sharing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ARM</td>
</tr>
<tr>
<td>I/O virtualization</td>
<td>Storage</td>
<td>GPU Sharing</td>
<td>Touch sharing</td>
<td>IPU Sharing</td>
<td>HECI sharing (Security)</td>
</tr>
<tr>
<td></td>
<td>Ethernet</td>
<td>GPU Prioritized Rendering</td>
<td>IOC sharing</td>
<td>USB DRD virtualization</td>
<td>CSME/DAL sharing (Security)</td>
</tr>
<tr>
<td></td>
<td>USB host controller (PT)</td>
<td>GPU Surface Sharing</td>
<td>Audio sharing</td>
<td>CarPlay</td>
<td>TPM Sharing (Security)</td>
</tr>
<tr>
<td></td>
<td>USB device controller (PT)</td>
<td>IPU (PT)</td>
<td>USB host controller Sharing</td>
<td></td>
<td>eAVB/TSN Sharing</td>
</tr>
<tr>
<td></td>
<td>Audio (PT)</td>
<td></td>
<td></td>
<td></td>
<td>SR-IOV</td>
</tr>
<tr>
<td></td>
<td>WiFi (PT)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Touch (PT)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GPU Sharing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Call For Action

• Watch, …
  https://github.com/projectacrn/acrn-hypervisor

• … try, …

• … and participate!
  https://lists.projectacrn.org/g/acrn-dev/topics
THINK OPEN
开放性思维
Reference:

• ELC2018 ACRN introduction – Eddie Dong

• Android tamper-resistant anti-replay secure storage solution and its virtualization – Bing Zhu
Backup

- Storage virtualization
- Network virtualization
- GPU virtualization
- Audio virtualization
Storage Virtualization

- Map a host storage area (SAR), i.e., disk / partition / file, as a guest disk
- Map a portion of host SAR (start_LBA, size) as a guest disk
Network Virtualization

Service OS

- Virtual Bridge / Switch
- Native NIC Driver
- Tap / Tun Driver

ACRN Device Model

NIC BE Service

User OS

- Virtio-NIC FE driver
- Guest Virtual NIC

ACRN Hypervisor

External Network
GPU Virtualization

ACRN Hypervisor

Host GPU Driver

App

GPU BE Services

vGPU

GPU

Guest GPU Driver

MPT API

Pass-through

Trap

User OS

User OS

User OS

User OS

User Kernel

User Kernel
Audio Virtualization

**Service OS**
- Audio Apps
  - ALSA lib/Tiny ALSA
- ALSA Core
- SOF Machine Driver
- SOF PCM Driver
- SOF IPC Driver
- DSP Platform Driver

**User OS**
- Audio Apps
  - ALSA lib/Tiny ALSA
- ALSA Core
- SOF Machine Driver
- SOF PCM Driver
- SOF IPC Driver

---

**Virtio Audio BE Service**

**Virtio Audio FE Drivers**

**ACRN Hypervisor**

ALSA: Advanced Linux Sound Architecture

FE driver communicate with IPC driver thru ops callback of platform driver

FE driver forwards IPC commands to BE service thru virtio shared rings

Service OS can directly access the memory of User OS

BE service communicate with IPC driver thru IPC TX/RX interface of IPC driver

---

*SOF: Sound Open Firmware; PCM: Pulse-code modulation; IPC: Inter-Processor Communication*