ACRN

Project ACRN: A Comprehensive Overview

Last updated: January 19, 2021

Content

- Introduction
- Architecture
- Value Proposition
- Key Capabilities
- Scenarios

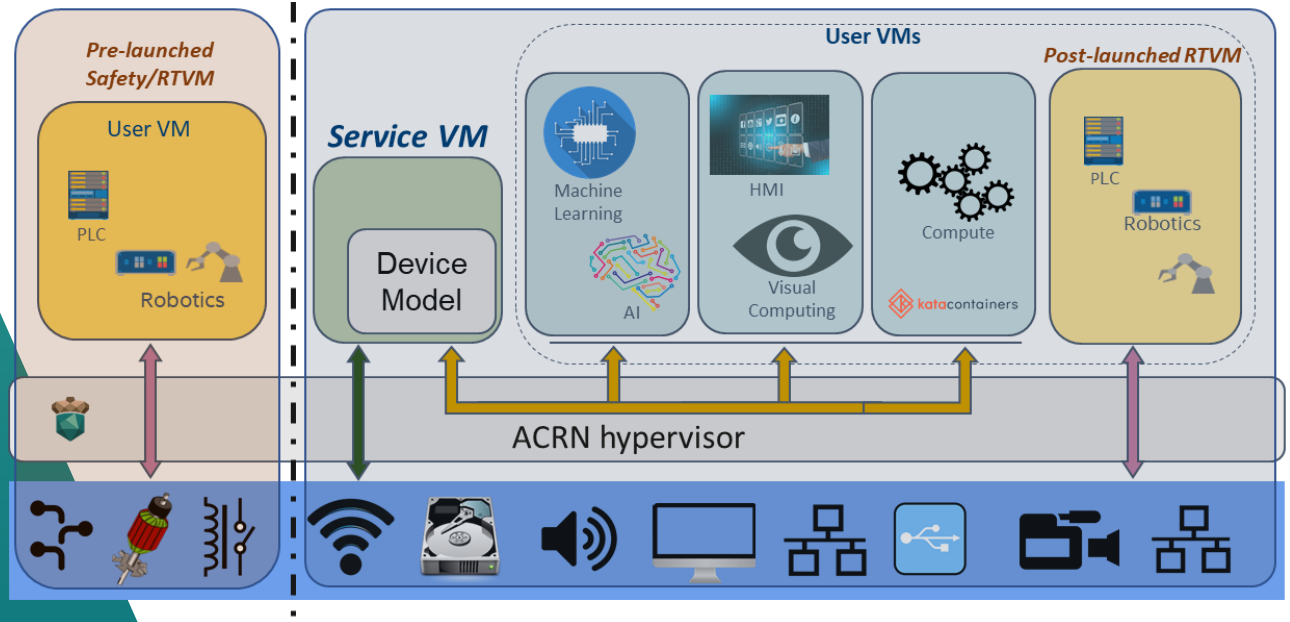
Introduction

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.

- A Linux Foundation Project Launched in March 2018
- Version 1.0 released in May 2019
- Version 2.0 released in June 2020

https://projectacrn.org

Overall architecture



Value Proposition - ACRN

Small Footprint

- Optimized for resource-constrained devices
- Small codebase: less than 40,000 vs. >156,000 lines of code for datacenter-centric hypervisors

Functional Safety & Hard Real-time

- Heterogeneous Workloads Consolidation
- Supports also the most demanding workloads: safety-critical and real-time

Open-source with Flexible License

- Permissive BSD license enables proprietary Guest OS
- Business-friendly to adopt, modify and integrate
- True open source with a vibrant Community

ACRN reduces system deployment complexity, enables heterogeneous architectures, and provide TCO advantages

Key Capabilities

Hard Real-time
Support hard or soft RT VM
Optimized for RT, e.g. no VMExit*, cache isolation

Rich I/O Mediation
Graphics, Audio, USB…
Industry standard Virtio BE/FE drivers

Flexible Architecture
Partition Mode, Sharing mode
Hybrid (mix of partition & share) mode

Various Guest OSes Support
Linux*, Zephyr*, Android*, VxWorks*, Windows*…

Secure Container
Kata Containers enabled for starting isolated and secure containers

Functional Safety
MISRA-C Compliance
FuSa certification targeted

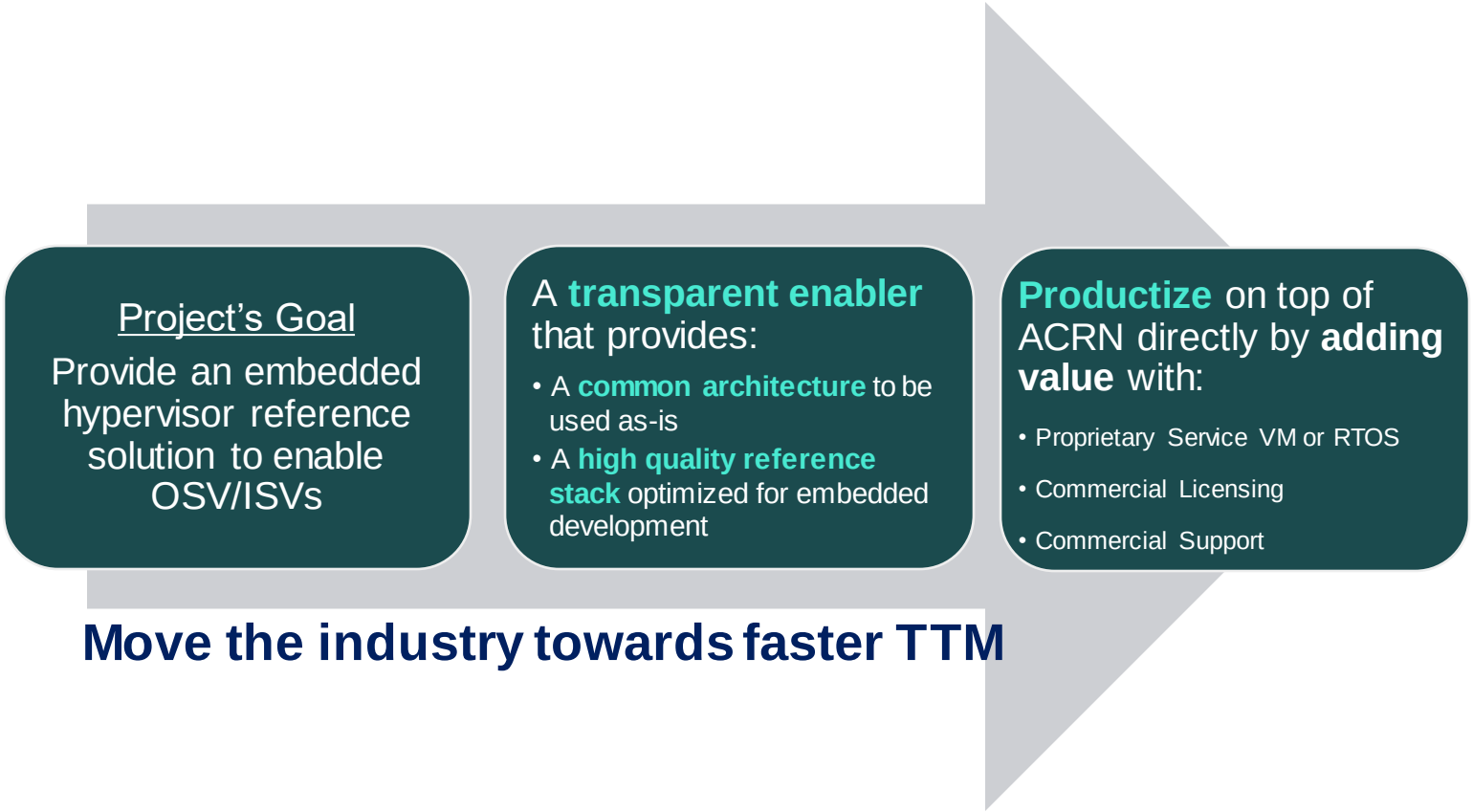
Security & Isolation
Full isolation for mixed criticality workloads
Intel VT backed isolation
Secure boot

Permissive Open Source License
Permissive BSD-3-clause license
Linux Foundation Affiliation

System Manageability
Flexible VM lifecycle Management
Virtualization API supported (libvirt)

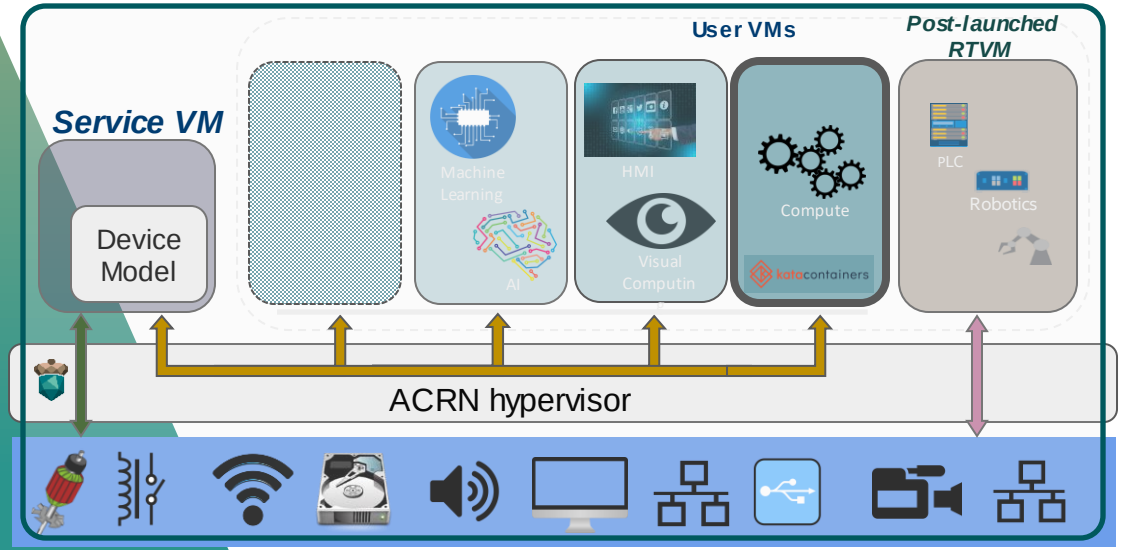
Ease of use
ACRN configuration tool
Rich documentation
Multiple-channel community support

ACRN™ & OSV/ISV Vendors



Move the industry towards faster TTM

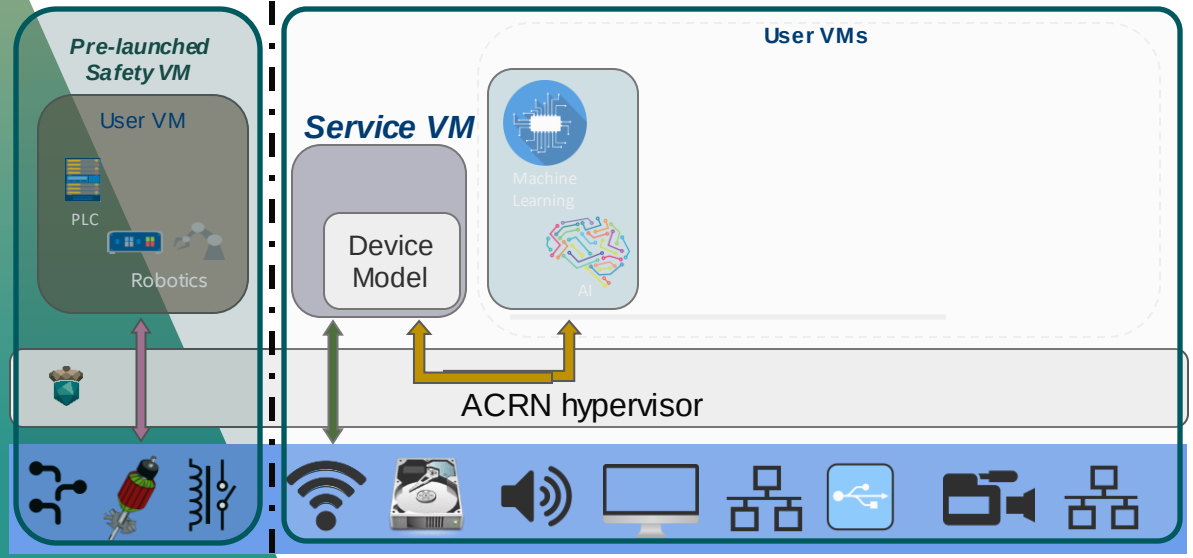
Industry



Key Challenges:

- ❑ **Mixed Workloads:**
  - Real-Time vs non Real-Time
  - Isolation vs Sharing
- ❑ **Real-Time (Hard / Soft)**
  - GBE packet IO control loop < 12us
  - MSI interrupt latency < 4us
  - Cyclictest jitter < 10us
- ❑ **HMI**
  - Windows 10

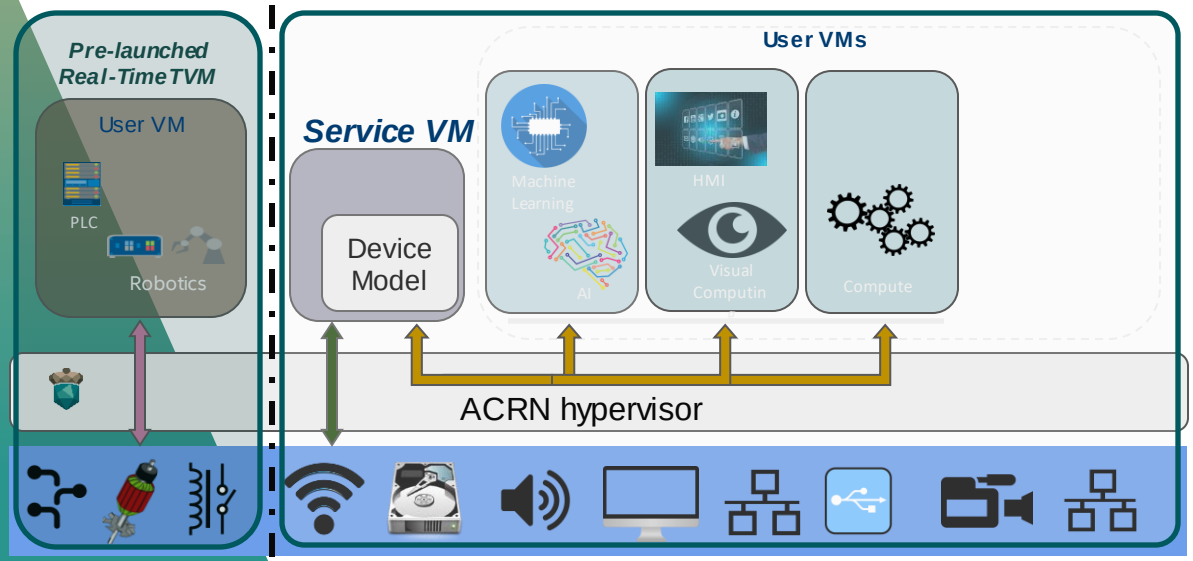
Hybrid



Key Challenges:

- ❑ **Mixed Workloads:**
  - Safety-Critical and normal workloads
  - Isolation and Sharing
- ❑ **Functional Safety**
  - IEC 61508-3 (Industrial)

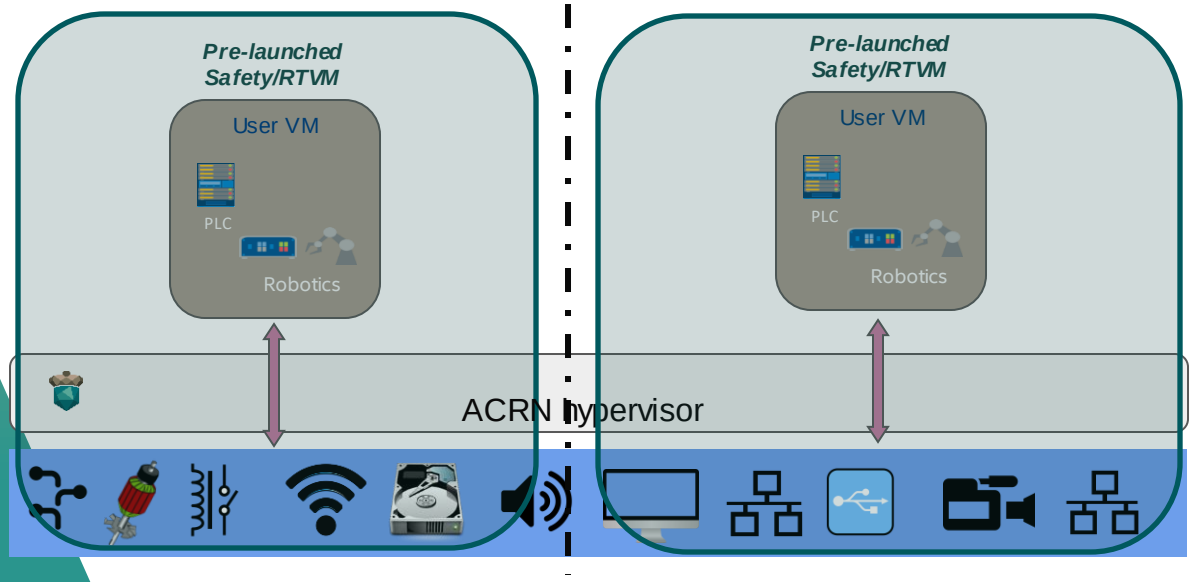
Hybrid Real-Time



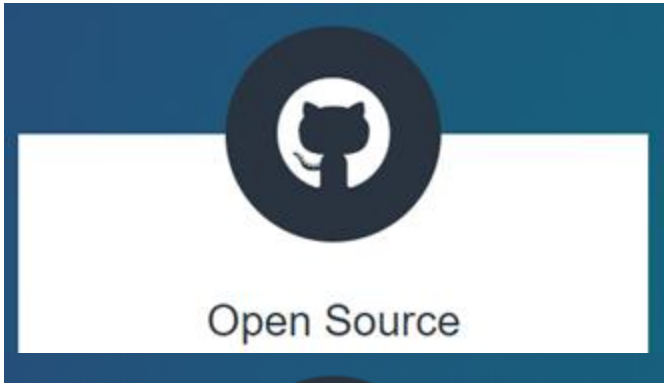
Key Challenges:

- ❑ **Mixed Workloads:**
  - Real-Time and normal workloads
  - Isolation and Sharing
- ❑ **Real-Time (Hard / Soft)**
  - GBE packet IO control loop < 12us
  - MSI interrupt latency < 4us
  - Cyclictest jitter < 10us

Logical Partition

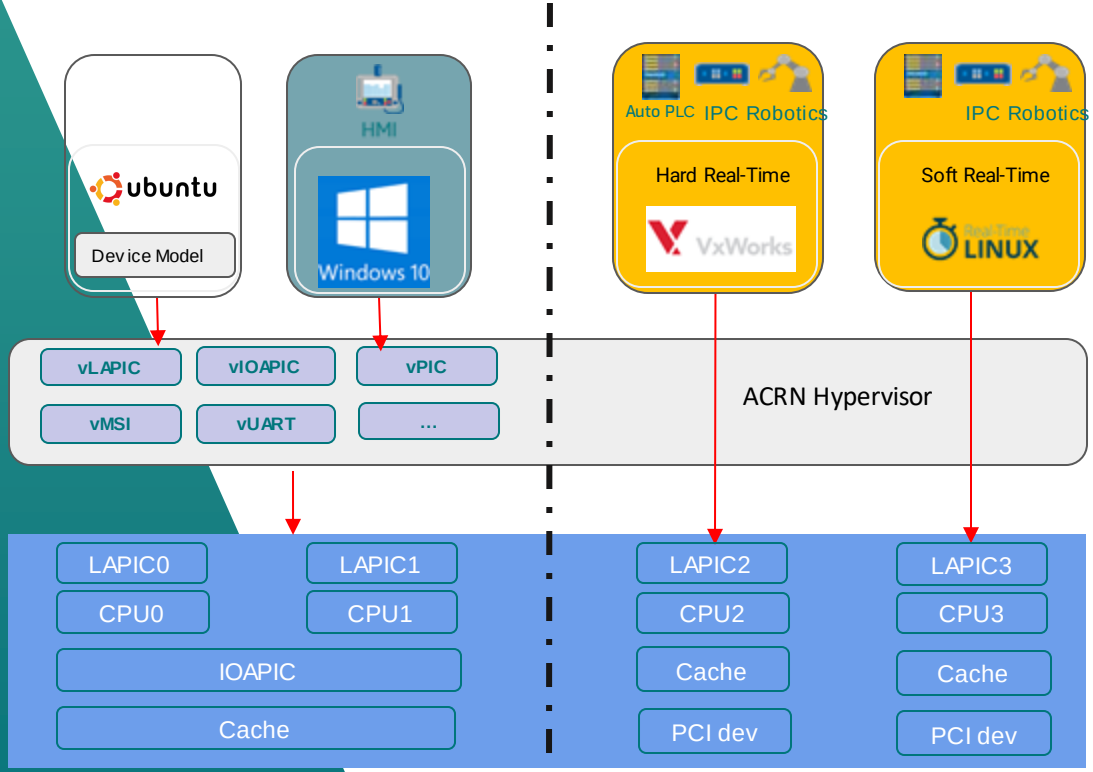


Open Source with Flexible Licensing



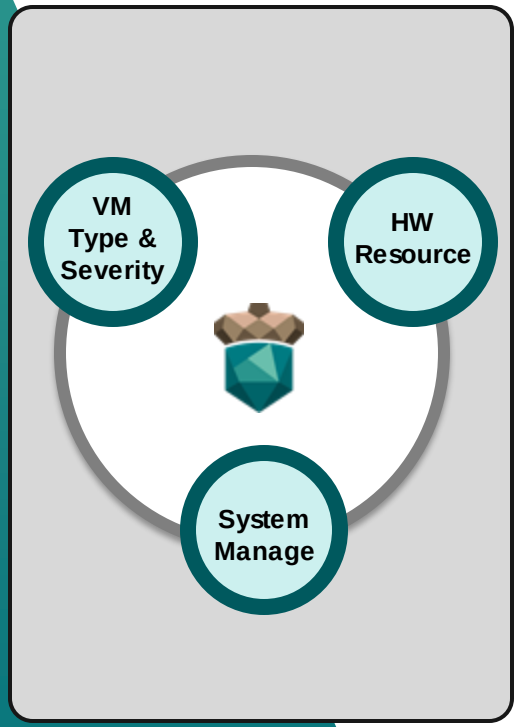
- Scalable support
- Significant R&D and development cost savings
- Transparent open source development model
- Collaborative SW development with industry leaders
- Permissive BSD licensing

Real Time



- ❑ Support hard or soft Real-Time VMs (RTVMs)
- ❑ No VMEXIT during runtime operations
- ❑ Highly optimized for RT:
  - LAPIC passthrough
  - RDT for resources isolation (cache and memory)
  - PCI device passthrough
  - Static CPU assignment
- ❑ Still use EPT and VT-d for VM isolation

System Manageability



VM Type & Severity:

- Load Order: Pre-launched, Service VM, Post-launched
- Category of VM: Service VM, User VM (can be pre-launched or post-launched)
- Severity: Safety VM > Hard RT VM > Soft RT VM > Service VM > Standard VM

HW Resource:

- CPU, memory & cache, devices, etc
- Partitioning or sharing based on VM type & severity

System Management:

- HW resources statically assigned at build time or dynamically assigned during runtime
- ACRN configuration tool: offline tool
- General reference design for VM & system lifecycle management
- Virtualization API: libvirt

System Security

Secure Boot	Isolation	Runtime Security
• Measured Boot • Verified Boot	• Isolation for mixed criticality workloads • Intel Virtualization technologies: VT-x, VT-d • Kata Containers • EPT memory Isolation • Interrupt isolation • Cache Allocation Technology (CAT)	• Virtual TPM • Trusty • Supervisor-Mode Access Prevention (SMAP) • Supervisor-Mode Execution Prevention (SMEP) • Software Guard Extension (SGX) • Dynamic Application Loader (DAL) • Total Memory Encryption (TME)

Rich I/O Mediation

- **I/O device mediators**

GPU	Ethernet	Block	Audio	IPU	I2C	GPIO	Touch	USB
Mediated Passthru	Virtio	Virtio	Virtio	Virtio	Virtio	Virtio	Virtio	Emu.

- **Various security virtualization features**

RPMB	CSE	TPM	Android Trusty	Verify Boot	Seed	SGX
Virtio	Virtio	Emu.	Emu.	Emu.	Emu.	Emu.

- **PCI devices pass-through (VT-d) capability too**

Diverse Guest OSes Supported

Ease of Use

Fast Development

- Short Learning Curve
- Straight-forward coding styles
- Multiple-channel Community (Mailing list, WeChat, TCM, etc)

Rich Documentation

- Getting Started Guide
- Architecture & Design
- Contributing Guides
- Tutorial
- Release notes

Easy Deployment

- Out-of-Box Experience
- VM Configuration Tool
- CPU assignment
- I/O sharing or pass-through
- Pre-defined Configuration
- Rich supported OS types
- Orchestration
- OTA

Flexible License

- BSD license for Hypervisor & Device models
- Dual Licenses for the ACRN Linux kernel drivers

ACRN

Fin