XEN ON THE I.MX8 PROCESSOR

Robbie VanVossen

Embedded Systems Engineering

Agenda

- Hypervisor Overview
- Xen Overview
- Xen System Configuration
- Controlling System Performance
- Xen Performance
- Example Systems
- Xen on i.MX 8

Hypervisor Overview

A general overview of what a hypervisor is, its purpose, and its function

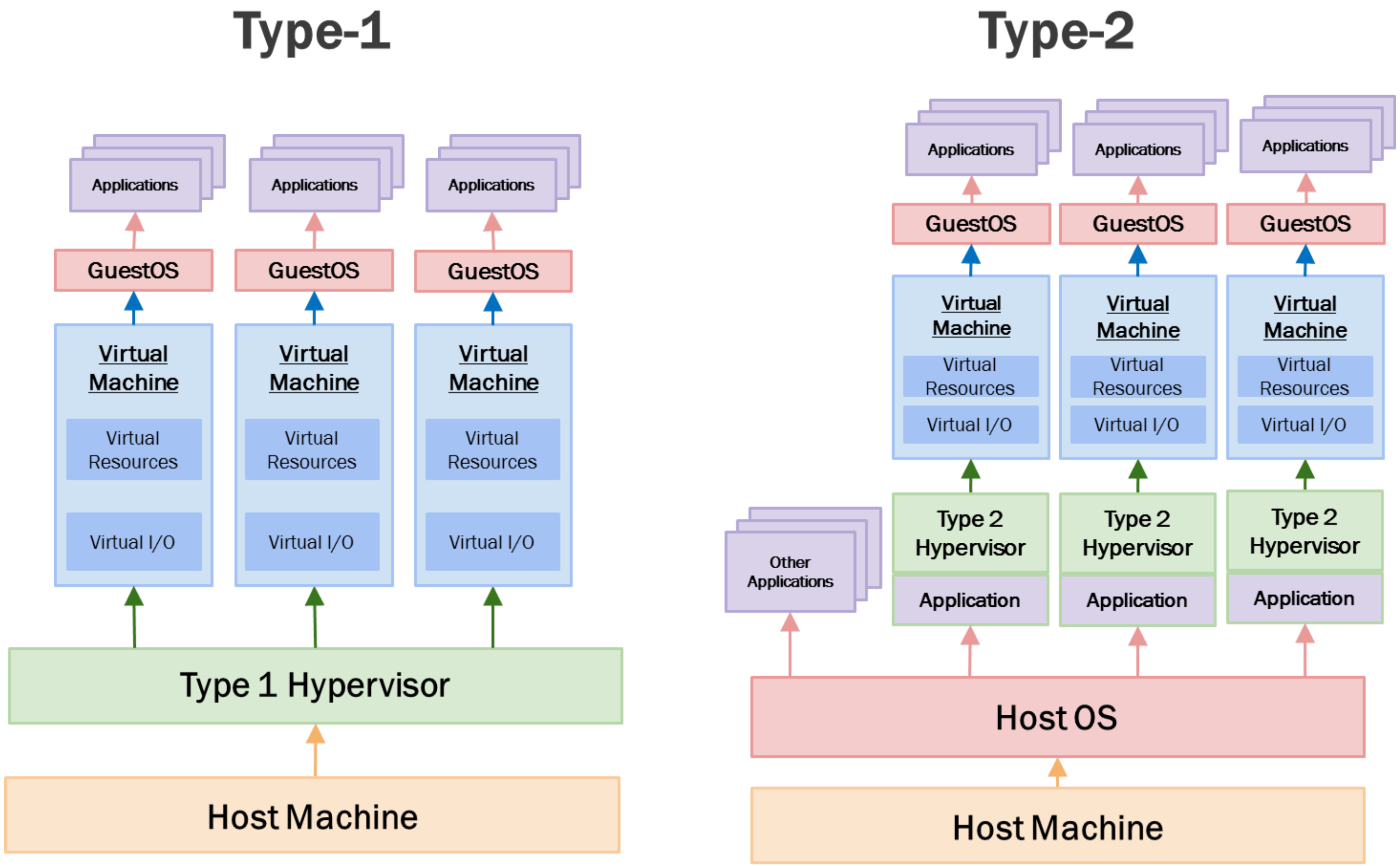
Hypervisor

- **Allocates and partitions** system resources such as RAM, CPU time across all cores, and I/O
- **Enforces isolation** of domains and their resources, such as CPU, memory, and I/O
- **Starts, stops, and configures** all of the guest domains (virtual machines) in the system
- **Configures inter-domain communication** via shared devices and memory while maintaining isolation

Hypervisor Benefits

- Increase **resource utilization**
- **Isolate** applications and operating systems
  - Isolation can provide both **security** and **safety**
- Reduce the **SWaP-c** of a system
- Improve **code portability**
  - Between both existing systems and future hardware
- Ease **legacy system migration**
- Increase **reliability** by providing **redundancy**

Type-1 vs Type-2 Hypervisor Diagram

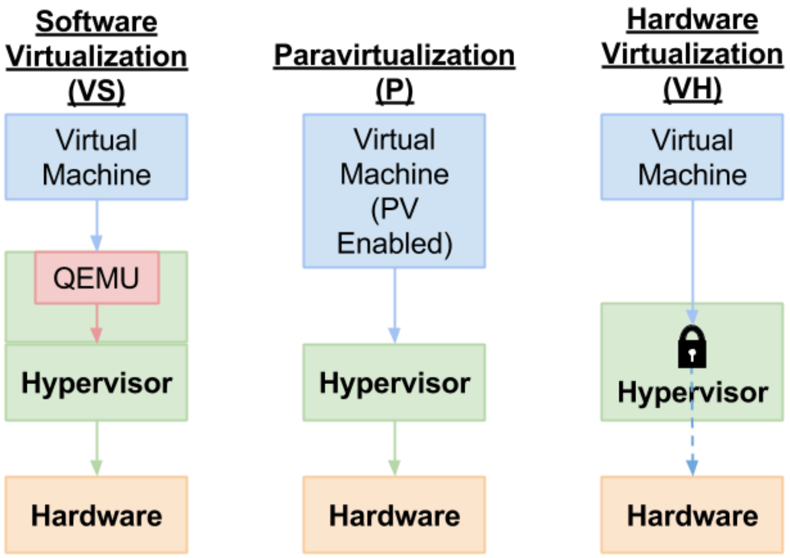


Type-1 vs Type-2 Hypervisor

- An embedded system should always use a Type-1 HV
  - Much lower system resource overhead
  - Highly efficient, can achieve near native performance
  - True isolation of domains at hardware level
  - Very fine grain control of **all** system resources
- A Type-2 HV is typically only used by desktops and servers
  - Useful for running a 2nd OS different from the base OS
    - Legacy OS for old software
    - Different OS for proprietary software (Windows + Linux, Mac + Windows, etc)
  - Generally simpler to use than Type-1 with the cost of high resource overhead, limited customizability, and limited controls

Virtualization Techniques

- **Software Virtualization** is the most portable, but is complicated
- **Paravirtualization** is less complex, but is less portable



- **Hardware Virtualization** is the least complex and most efficient, but has limited portability as it requires support from the hardware itself

DornerWorks
where hardware and software design meet

Xen Overview

How Xen implements these hypervisor concepts and how they apply to ARM

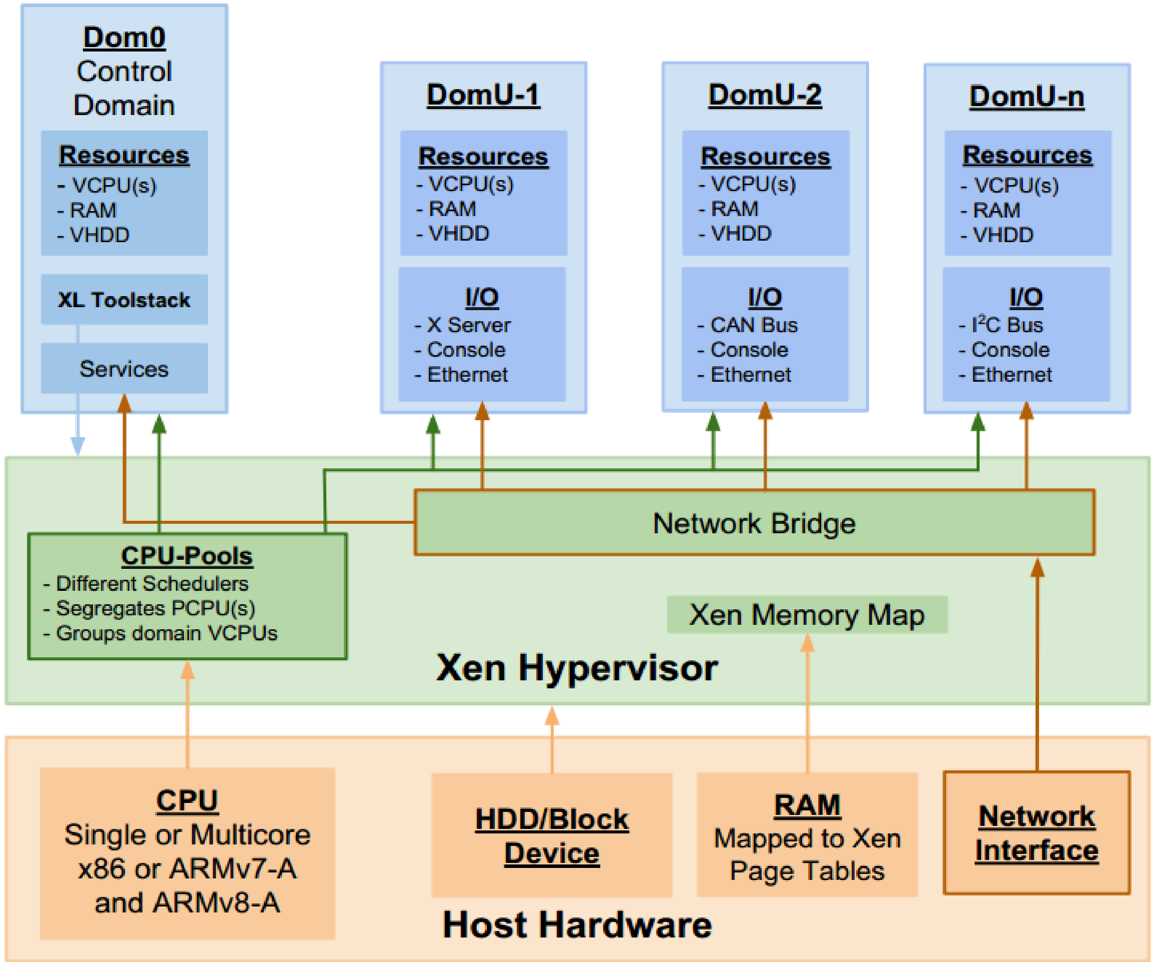
Xen Hypervisor Overview

- Xen is a Type-1 Hypervisor
- Open Source project started over 12 years ago
  - GPLv2 License, same as Linux Kernel
- Initially developed for desktop and servers
- Port to ARM embedded processors began in 2008
  - DW Involved in early stage of embedded development (Navy SBIR)
  - Fully functional prototypes for ARMv7a with VE released in 2011
  - Xen for ARM was incorporated into mainline Xen in 2013
- Long track record with major companies including Amazon Web Services, AMD, Bromium, Calxeda, CA Technologies, Cisco, Citrix, Google, Intel, Oracle, Samsung, and Verizon

Xen Terminology

- Some basic Xen/Hypervisor terminology:
  - **Resources** - All of the available hardware
  - **pCPU** - Physical CPU residing on the motherboard or SoC
  - **vCPU** - Virtual CPU, a schedulable processor unit used by a domain
  - **Scheduler** - The algorithm used by the hypervisor to give guests time on the pCPU's
  - **Dom0** - The special "privileged" domain used to configure Xen
  - **DomU** - An "unprivileged" guest domain in Xen
- For a large glossary of terms and concepts: http://wiki.xenproject.org/wiki/XenTerminology
  - Note that some terms and features referenced in the wiki may be applicable to x86-only Xen

Example Xen System



Virtualization as Implemented by Xen

Poor Performance
Scope for Improvement
Optimal Performance

P = Paravirtualized
VS = Software Virtualized (QEMU)
VH = Hardware Virtualized

Shortcut	Mode	With	Disk and Network	Interrupts & Timers	Emulated Motherboard, Legacy Boot	Privileged Instructions, Page Tables
HVM / Fully Virtualized	HVM		VS	VS	VS	VH
PV	PV		P	P	P	P
PVH	PV	pvh=1	P	P	P	VH

Xen on ARM

- Xen on ARM uses a blend of these techniques known as **PVH**
- This blend provides an optimized balance between complexity, efficiency, and portability

Poor Performance
Scope for Improvement
Optimal Performance

P = Paravirtualized
VS = Software Virtualized (QEMU)
VH = Hardware Virtualized

Shortcut	Mode	With	Disk and Network	Interrupts & Timers	Emulated Motherboard, Legacy Boot	Privileged Instructions, Page Tables
PVH	PV	pvh=1	P*	VH	NA	VH

* Devices can be passed through to avoid any paravirtualization

ARM Hardware Virtualization

- Exception Level (EL)
  - **EL3**: Secure Monitor
  - **EL2**: Hypervisor
  - **EL1**: Operating System
  - **EL0**: Application
  - Each EL has its own copy of certain registers (e.g. MMU, SP)
- Virtualized Registers
  - e.g. Virtual Timer
- Configurable privilege for specific register access and specific instructions
  - e.g. TLBI VAE1 (TLB invalidate by VA, EL1)

Components of a Xen based System

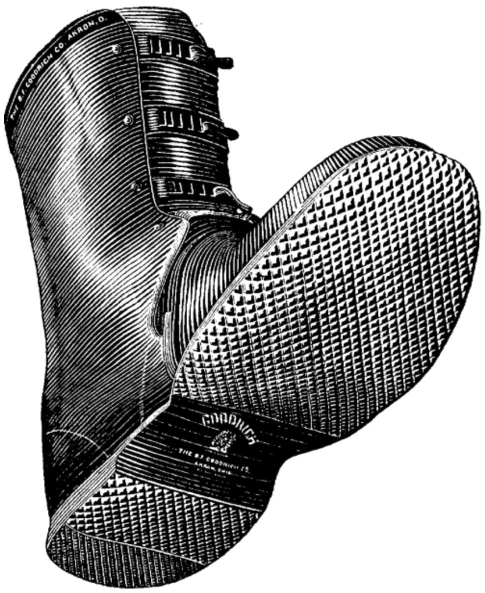
- A full solution is much more than the hypervisor itself!
  - Xen Kernel
  - Device Tree Blob (DTB)
    - This DTB is used by Xen and Dom0 Linux
    - Contains Xen bootargs and SMMU `mmu-masters` entries (discussed later)
  - Dom0 Kernel
    - Drivers for Xen hypercalls (used by toolstack)
    - Backend Drivers (if needed)
  - Dom0 FileSystem
    - Xen ToolStack (xl)
  - Guest
    - Kernel (or bare metal application)
      - Frontend drivers (if needed)
      - Xen specific drivers (for shared mem, etc)
    - Guest File System (if needed)
    - Guest Configuration File
    - Partial DTB (if needed; for pass-through devices)

Xen System Configuration

Configuring the Xen system components to start Xen and dom0 to launch your guest OSes and bare metal applications

Boot Sequence

1. FSBL
   - Initiates System Controller(SC), executes U-Boot
2. U-Boot
   - Loads the Xen kernel, Dom0 Linux kernel, and DTB into RAM and starts execution of Xen
3. Xen
   - Reads in the system DTB (which also contains the Xen bootargs) and configures the systems resources
   - Executes the Dom0
4. Dom0 Kernel
   - Privileged domain, executes toolstack commands and uses guest configuration file to boot up DomU's
5. DomU
   - Executes system guest operating systems, applications, and functions as configured



System Device Tree

- System dtb is generated by a dts (device tree source) file and loaded on startup by U-boot
  - Formatted and used in the same way as a standard Linux device tree
- The same DTB is used by Xen and Dom0 Linux (slight changes for Dom0)
- Relevant entries include:
  - Xen bootargs
    - Example: `xen,xen-bootargs = "console=dtuart dtuart=serial0 dom0_mem=512M bootscrub=0 maxcpus=4 dom0_max_vcpus=1 dom0_vcpus_pin timer_slop=0";`
    - Full documentation of all Xen bootargs: http://xenbits.xen.org/docs/unstable/misc/xen-command-line.html
  - SMMU mmu-masters entries
- Full documentation of device trees is available here: http://elinux.org/Device_Tree_Usage

Dom0 Kernel and xl Toolstack

- The dom0 Linux kernel must be configured for Xen when it is compiled, this includes adding:
  - Drivers for Xen hypercalls (used by toolstack)
  - Backend Drivers for PV devices (if needed)
- The toolstack used by dom0 to configure Xen is known as `xl`
  - The `xl` toolstack is a user-space program that is run in dom0 Linux
  - Commands are available to control many aspects of the system, we call many of these aspects in the following slides
  - Full documentation is available in the Xen Man pages: http://xenbits.xen.org/docs/4.6-testing/man/xl.1.html

Dom0 File System

- Dom0 File System can be located on: SD Card, RAM Disk, NFS, SATA
- Uses any Linux compatible format
  - Prefer ext4, ext3, ext2, etc
- In addition to Linux system files, the FS must also contain:
  - Xen xl toolstack
  - Any backend drivers needed for PV (hvc is minimum for console)
  - Standard Linux drivers for all the board devices
  - May optionally contain guest kernels and/or FSes
- Where/How it is configured:
  - The file system is set by the Dom0 bootargs (located in system DTB); must point to the correct partition

Domain Memory Assignment

- Domains are given permission to access a particular range of memory addresses
  - Xen does this by configuring the EL2 MMU
  - Each guest has its own physical memory location
    - But the guest is only aware of the Intermediate address that Xen sets up via the EL2 MMU
  - The guest then sets up the EL1 MMU
- Minimum memory segment size

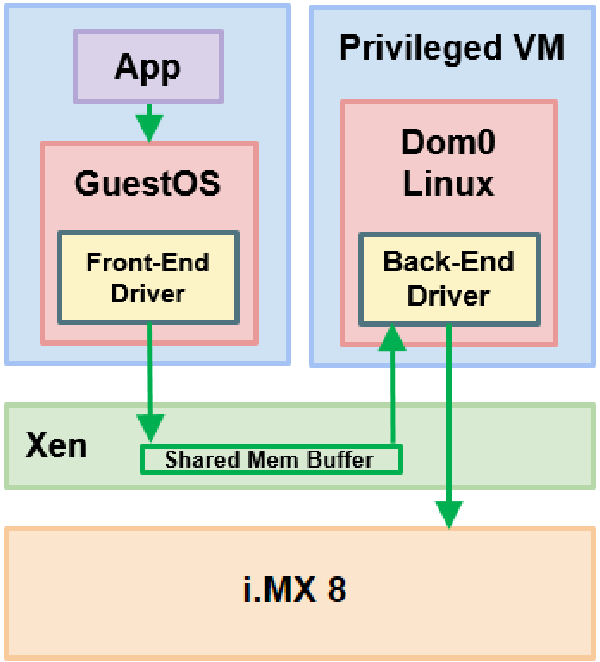
- Where/How it is configured:
  - Dom0 memory size is set via xen bootargs
  - Guest memory size is set in the guest config file
  - Configuration file entries
    - memory
    - maxmem

Controlling System Performance

Xen tools and features for achieving desired performance

Input/Output (Paravirtualization)

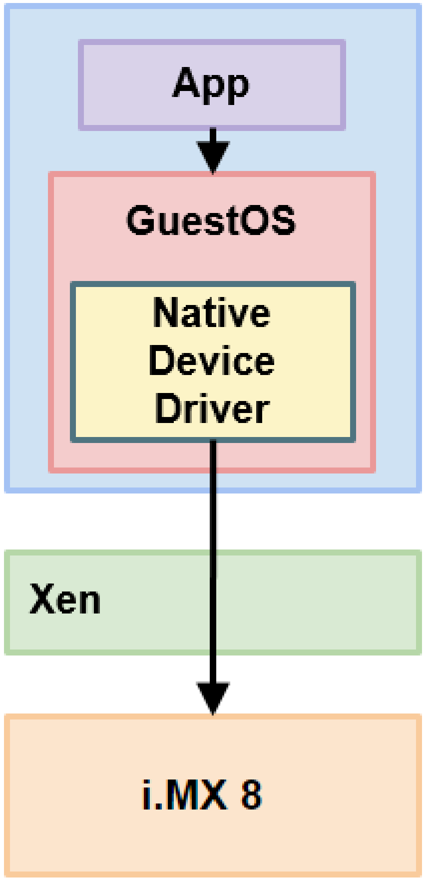
- Split Driver
  - Use if **multiple guests** need to access a particular device
- A single privileged guest has sole, direct access to device
  - Privileged guest provides backend device driver
- Each guest using the device talks to backend driver in the privileged guest through its own frontend driver



DornerWorks
where hardware and software design meet

Input/Output (Pass-Through)

- Pass-Through Driver
  - Generally, use if **only one** guest needs access to a particular device (In most cases)
- Utilizes the SMMU
  - System DTB sets up devices as SMMU `mmu-masters`
    - These can be passed-through
  - The guest config file tells Xen to allow the device to be passed through to that guest
- The guest controls the device directly
  - This means guest can use the original unaltered device driver as if running directly on HW



Multicore and Scheduling

- Schedulers
  - **Credit**: Default Xen scheduler; a "fair time" algorithm similar to default Linux scheduler
  - **RTDS**: Soft Real-time scheduler that uses deferrable server algorithm to assign vCPUs
  - **ARINC653**: Soft Real-time scheduler, assigns vCPUs according to ARINC653 specification
- CPU Pinning
  - Assign a vCPU to run on specific pCPUs **only**
- CPU Pools
  - Group pCPUs to be managed by an scheduler instance
  - Allows for multiple schedulers to be run at the same time
    - This flexibility allows the system to be configured to run SMP or Heterogeneous or Homogeneous AMP as needed

i.MX 8 SC Partitioning

- Pros
  - No need for Emulation/Virtualization SW
  - Partitioning cannot be compromised due to SW
  - Avoids some ASIL complications
- Cons
  - Fixed limit of HW resources
  - Vendor Dependent feature
  - Just delays the inevitable need to address ASIL cert
- Combined Use Cases
  - HW partitioning as additional assurance
  - HW partitioning first, SW Partitioning as needed

Xen Performance

Xen performance/overhead on an ARM platform

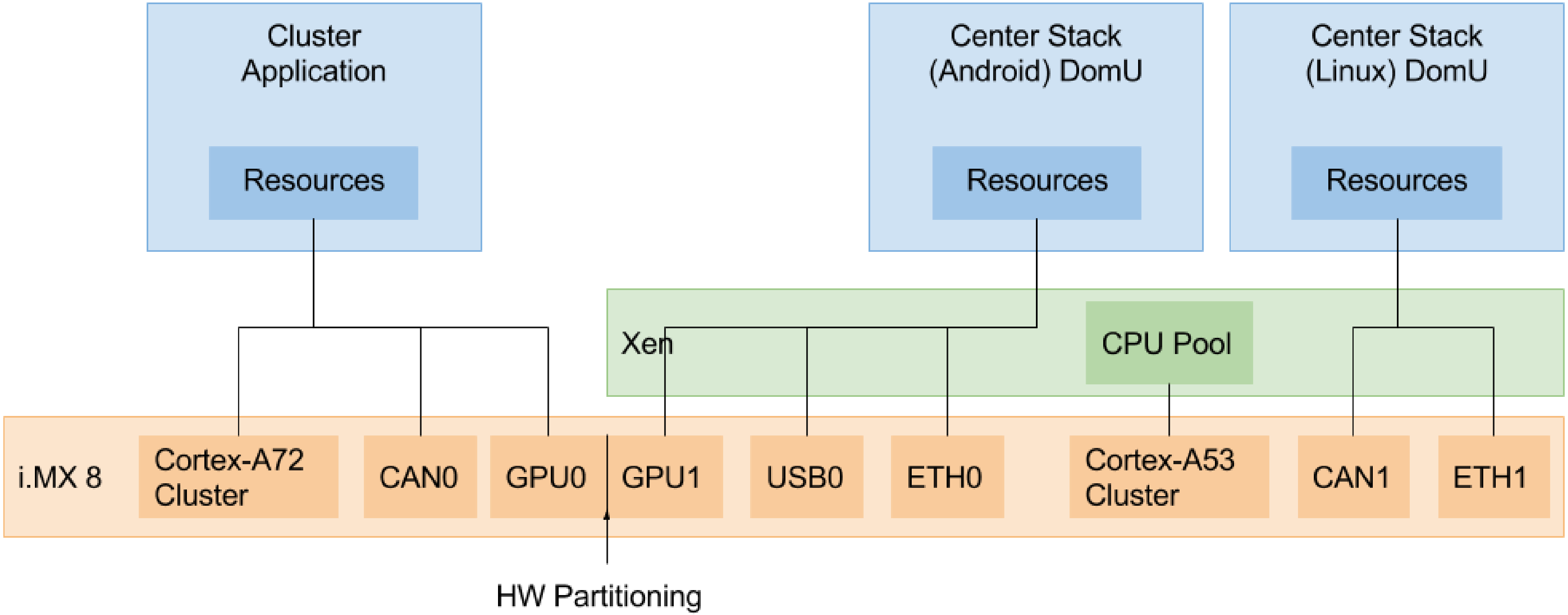
Xen Performance

- Quad Core Cortex-A53
- Boot times
  - Xen 0.8 sec, *dom0* 5.1 sec
- Interrupt Latency
  - 2.3 µsec
- Context Switch Overhead
  - ~0% to 0.6%
- For more information see: http://schd.ws/hosted_files/xensummit2016/39/Embedded%20Xen%20Perf.pdf

Example Systems

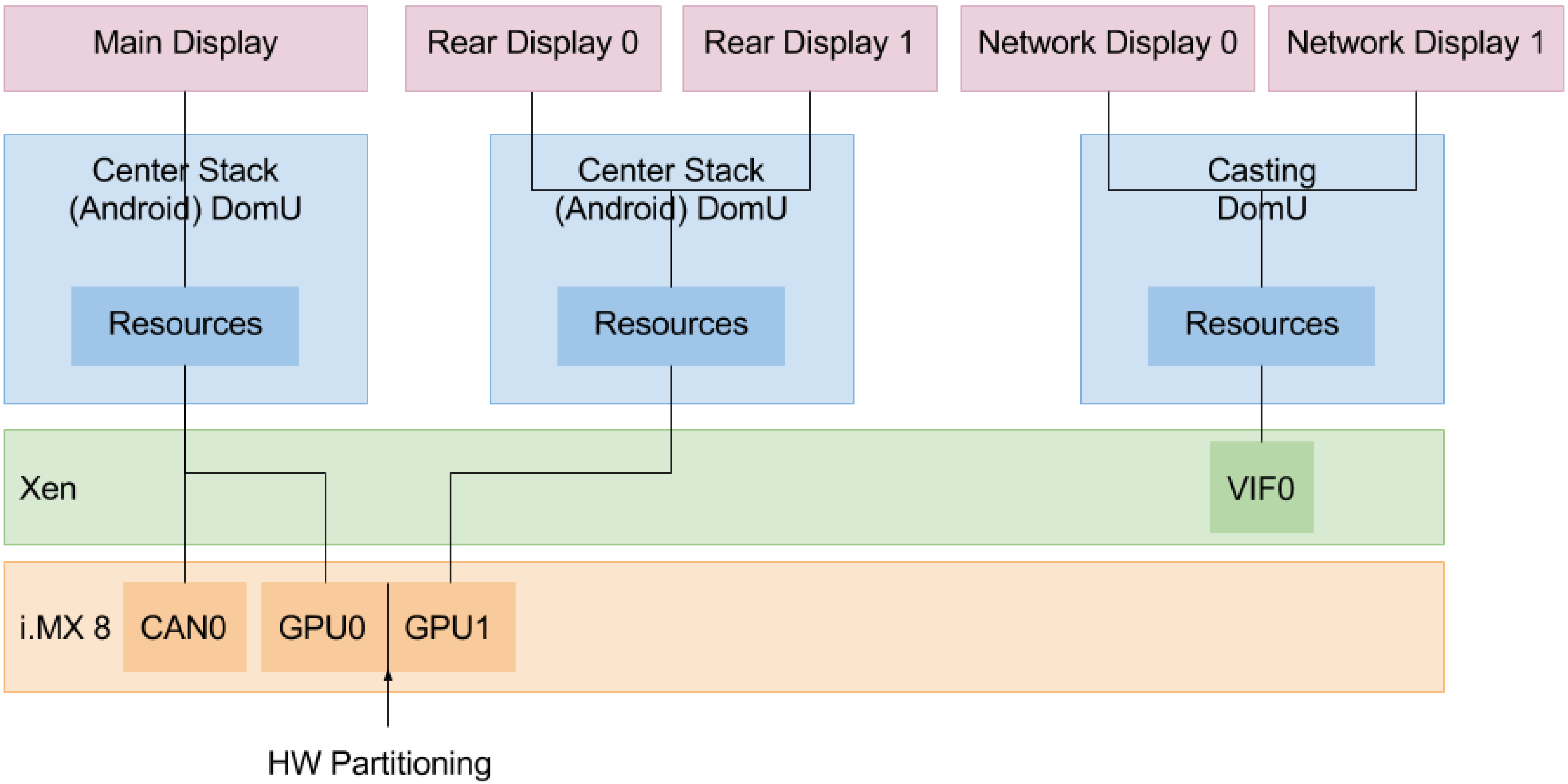
Examples on how Xen can be used in automotive applications

Cluster/Center Stack Example

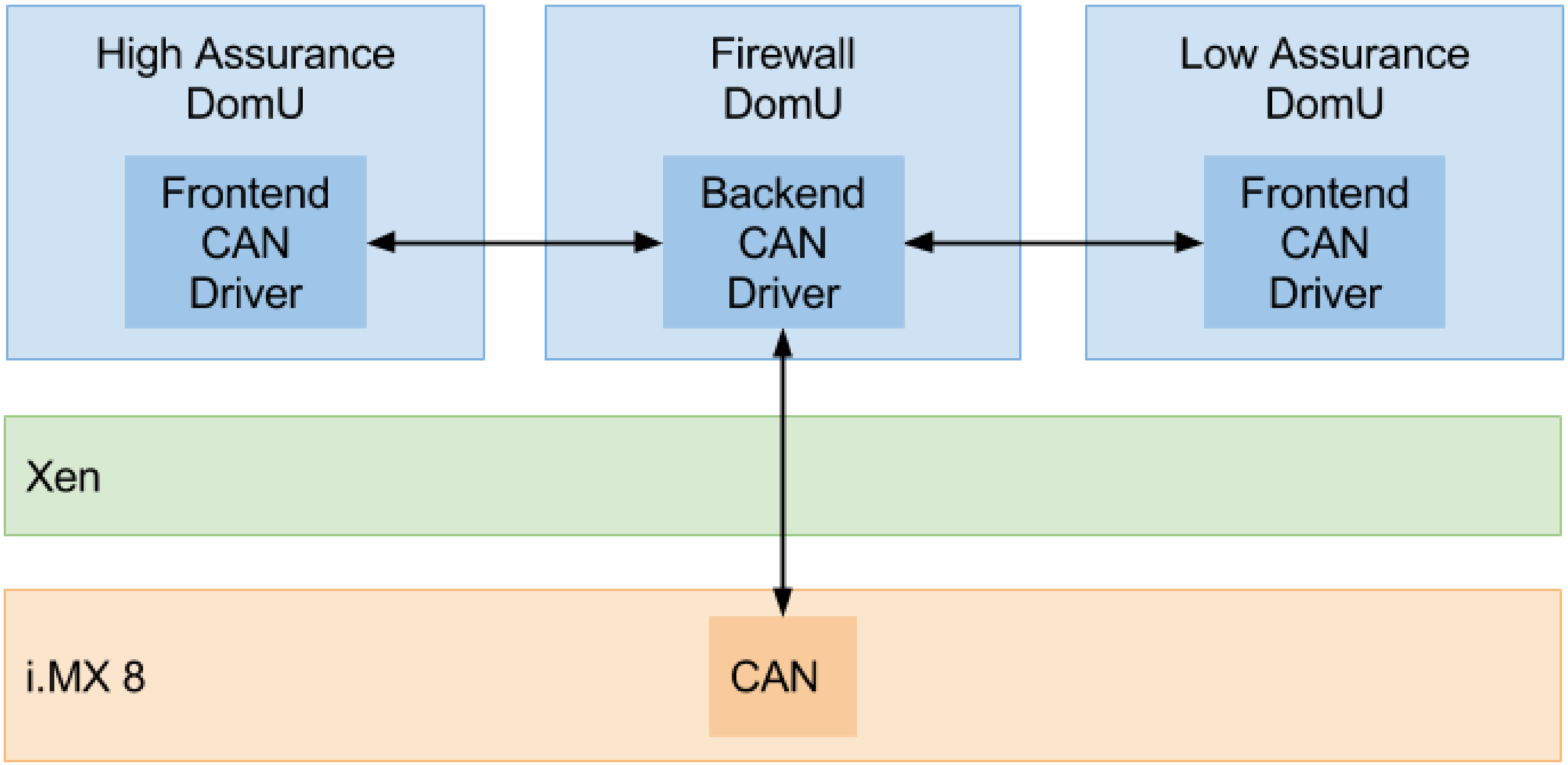


DornerWorks
where hardware and software design meet

Entertainment Example



CAN Firewall Example



DORNERWORKS
where hardware and software design meet

Xen on i.MX 8

Current status and future plans of Xen on the i.MX 8

Xen i.MX 8 Current Status

- Xen 4.7 is running on A53 cluster
- Linux 4.1.30 Control Domain (Dom0)
  - Graphics (GPU, DPU, FBs)
  - Networking
  - SD Card
- Linux 4.1.30 guest with PV IO
  - VIF, VBD, hvc
- Demonstration

Xen i.MX 8 Challenges

- Make sure Dom0 ends up in 32-bit Space
  - Needed so DMA works
  - Difficult due to Xen allocator
- User Space Application DMA
  - Implement new function for Dom0 in Linux Kernel
- Integrating xl toolstack build into Yocto
- Disable SC interaction for Guests

Xen i.MX 8 Planned Features

- Pass-through of half of graphics to a single guest
- Xen and Guests running heterogeneously on Cortex-A72 cluster (big.LITTLE)
- Provide a Xen Distribution for i.MX 8
- More guests in Demo & Distro

Xen i.MX 8 Distribution

- Continued Updates & Support
  - Free and Paid Support options
- Components
  - Xen Kernel
  - Dom0 Linux Kernel & File System
  - System Device Tree
  - Example guests & configuration files
    - Linux, FreeRTOS, BareMetal Guest

Questions

?