
epam
GENIVI AMM
Hypervisor workshop
GENIVI®

<epam>

Virtualization for Multi-core, SoC peripheral hardware and special-purpose CPUs

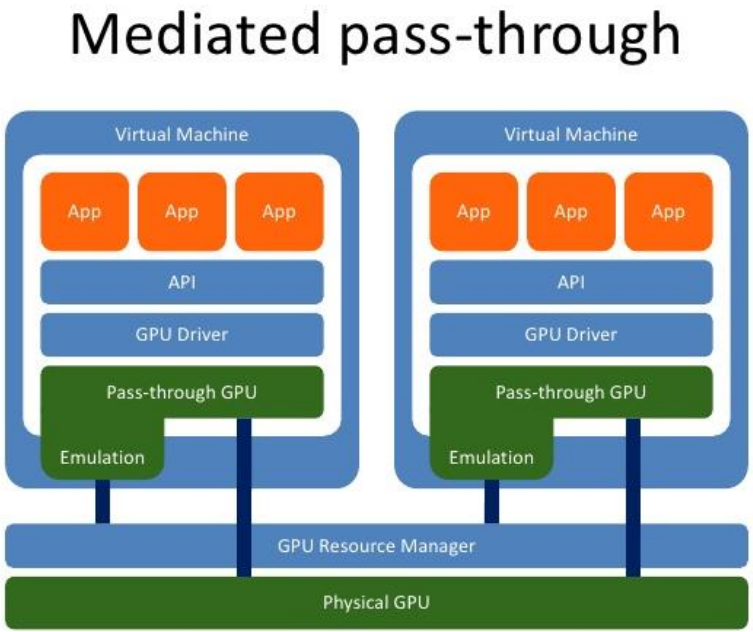
Peripherals sharing

- When no sharing is needed...
  - Direct pass through (IPMMU only)
  - Full virtualization (emulation)

- Para-virtualization
  - Centralized control over HW
  - Complex control logic (mixing, composing)
  - Lower performance
- Hardware-assisted virtualization
  - IPMMU needed
  - Only control logic supported by HW
  - High performance

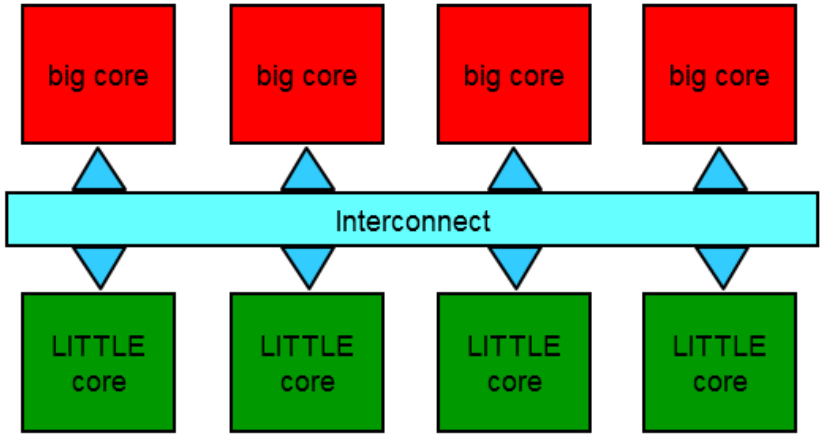
Co-processors support

- Co-processor = DSP, GPU, IPU, …
  - Data-intensive computing scenarios
  - Memory shared with main CPU
  - SMMU controlled by main CPU
  - Has firmware
  - No native virtualization support



Heterogeneous multicore

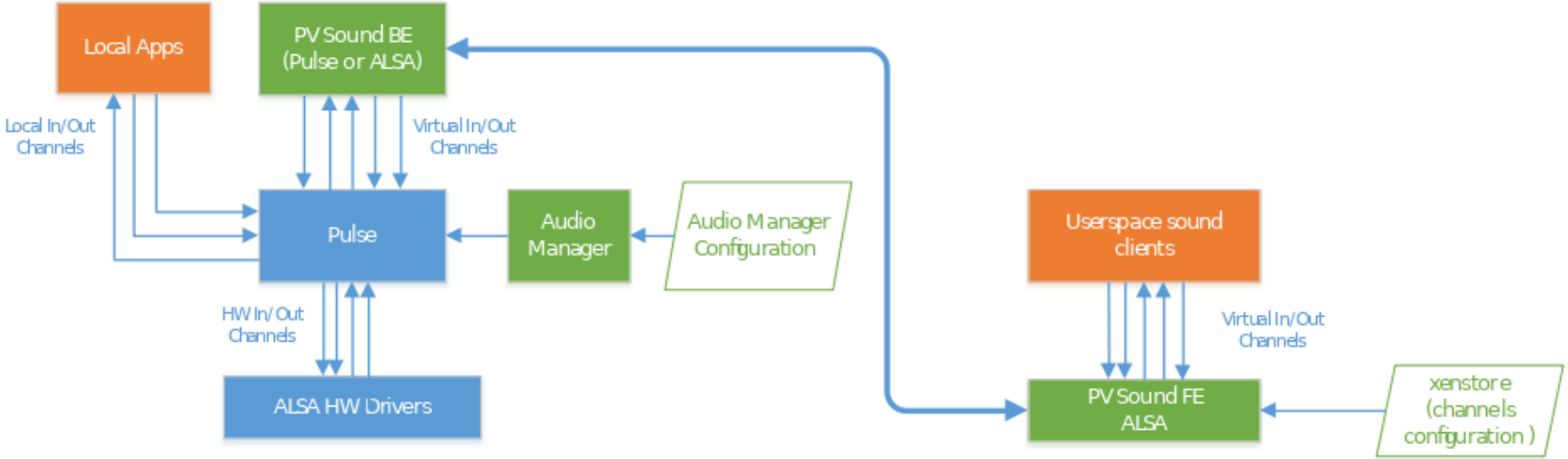
- E.g. Arm big.LITTLE
- vCPU <-> pCPU ?
  - Performance
  - RT constraints (soft or firm)
  - Energy-aware scheduling (EAS)



<epam>

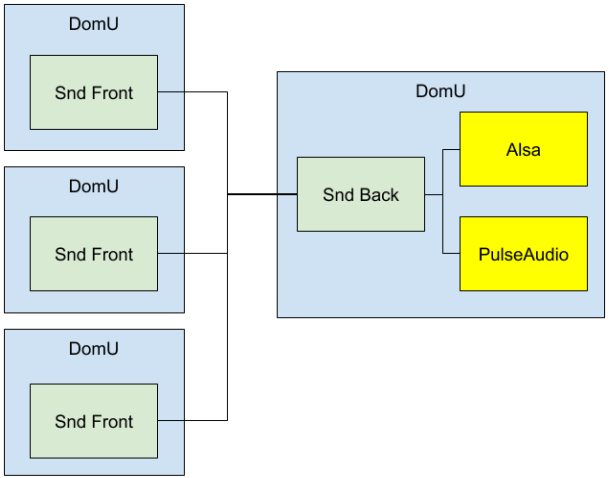
Audio system design with HVs

Audio architecture



Audio backend interface

- Alsa
  - Has:
    - Route different streams to different HW devices
    - Mix few streams to the one HW device with dmix plugin
    - Volume control per HW device but not per audio stream
  - Need to be done:
    - Audio manager
    - Per stream volume control
- Pulse
  - Has:
    - Route different streams to different HW devices
    - Mix few streams to the one HW device with dmix plugin
    - Volume control per HW device and per stream
    - Automatic static ducking
  - Need to be done:
    - Audio manager



GENIVI Audio Manager

- Pros:
  - The core part is lightweight and simple
  - Extendable functionality with plugins
  - Rich sets of policies based on triggers, conditions and actions
  - Controls external HW
  - Routes different boards, different audio domains
- Cons:
  - Not so many plugins are currently available (generic control plugin, routing ALSA plugin and POC of routing pulse plugin)
  - Example control plugin is a bit complicated
  - For normal usage application should communicate with AudioManager through DBus, GENIVI Common API or custom command plugin

Implementation for Xen

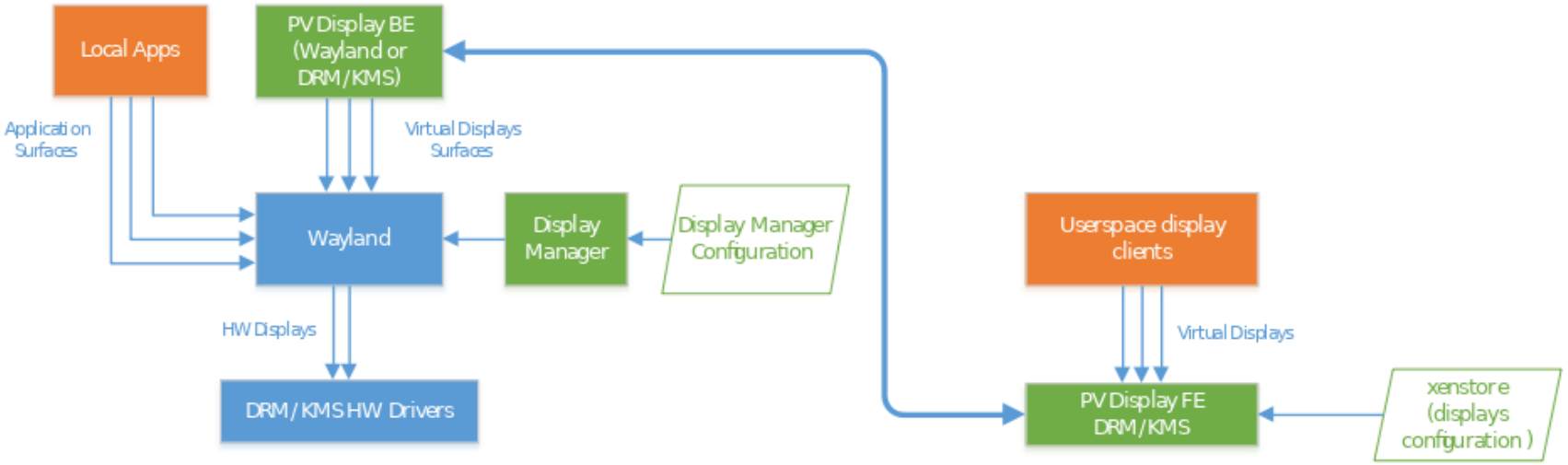
- PV Sound protocol for Linux kernel https://git.kernel.org/pub/scm/linux/kernel/git/xen/tip.git/tree/include/xen/interface/io/sndif.h?h=for-linus-4.17&id=cd6e992b3aab072cc90839508aaf5573c8f7e066
    - Already in kernel, updates planned for 4.17
- PV Sound frontend https://lkml.org/lkml/2018/4/16/76
    - Upstreamed!
- PV Sound backend for Xen https://github.com/xen-troops/snd_be
- GENIVI Audio Manager extensions with Pulse https://github.com/xen-troops/AudioManagerPlugins

Graphics/GPU Sharing

Graphics Sharing & Distributed HMI

- GPU sharing
  The GPU can be used from multiple operating systems, so it is shared.
  Concurrent access to the physical GPU has to be controlled by the hypervisor, hardware or other means which are implementation specific.
- Display sharing
  The physical display can be shared across multiple operating systems. HW compositor unit composites final display buffer from HW Layers of each OS.
  This requires virtualization of the display controller hardware.
    - Studied technologies:
        - *Since this is mostly hardware specific we have not (yet) looked at it much.*
        - Layer-Management might be seen as a *related* software abstraction, but intended a for single-system.

Display sharing



Implementation for Xen

- PV DRM protocol for Linux kernel https://www.mail-archive.com/linux-kernel@vger.kernel.org/msg1650261.html
  - Just upstreamed to kernel!
- PV DRM frontend https://cgit.freedesktop.org/drm/drm-misc/tree/drivers/gpu/drm/xen
  - Upstreamed!
- PV Display & Input backend for Xen https://github.com/xen-troops/displ_be
- GENIVI Wayland IVI extensions

Coprocessor (incl. GPU) sharing in Xen

- [Xen-devel] [RFC] Shared coprocessor framework https://lists.xenproject.org/archives/html/xen-devel/2016-10/msg01966.html