Dealing with Hardware Heterogeneity Using a Virtualization Framework Tailored to ARM Based Embedded Systems

Embedded Xen

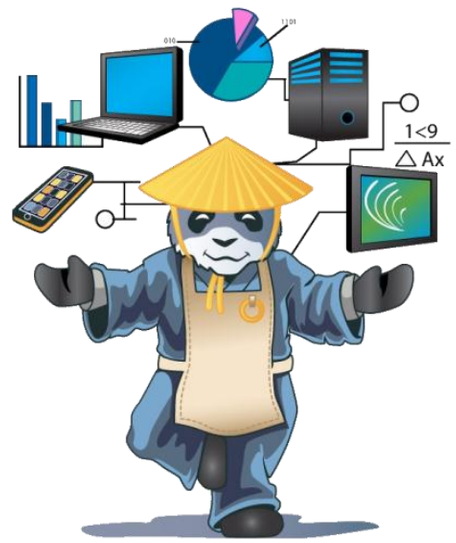
Prof. Daniel Rossier, PhD

HEIG-VD
Institut REDS, Reconfigurable & Embedded Digital Systems
rte Cheseaux 1, 1400 Yverdon-les-Bains
http://www.reds.ch/

ReDS
Reconfigurable & embedded Digital Systems

Outline

- Background
- Overview of *EmbeddedXEN*
- Protection & **Memory** Isolation
- **Domain Interactions**
- **Device Heterogeneity**
- Conclusions & Future Work



Background

- HEIG-VD
  - *University of Applied Sciences in Yverdon-les-Bains, Switzerland (CH)*
  - ***R**econfigurable **E**mbedded **D**igital **S**ystem Institute*
    - *Hardware Design, FPGA*
    - *Embedded Execution Environment, Drivers, BSP, OS/RTOS, etc.*

- *Applied Research & Development*
  - *ARM based* Microcontrolers *(v4, v5, v6, v7)*
  - Low-level interactions between computing cores *(CPU, DSP, FPGA, etc.)*
  - Towards *Cortex-A15 HVM*

Background

- **Embedded Virtualization** on ARM
  - **XEN**: free & easy access to a stable & evolving hypervisor source code
  - Early port of XEN on ARM in 2007 in the context of a Diploma Project
  - Necessity to get a simple, thin, fast, robust, easy-to-deployed virtualization framework
  - Re-use of *Linux* file organization & build system (*Makefiles, scripts, …)*
  - Focus on realtime aspects (*Linux/Xenomai* as RTOS)

- Different sources of inspiration
  - "**Fast Secure Virtualization for the ARM Platform**", *Daniel Ferstay,* Master Thesis, *The University of British Columbia*, 2006
  - **XEN ARM port**, *George G. Davis, MontaVista,* 2007
  - ***Secure* Xen on ARM Project**, *Sang-bum SUH, Samsung,* 2007

Background

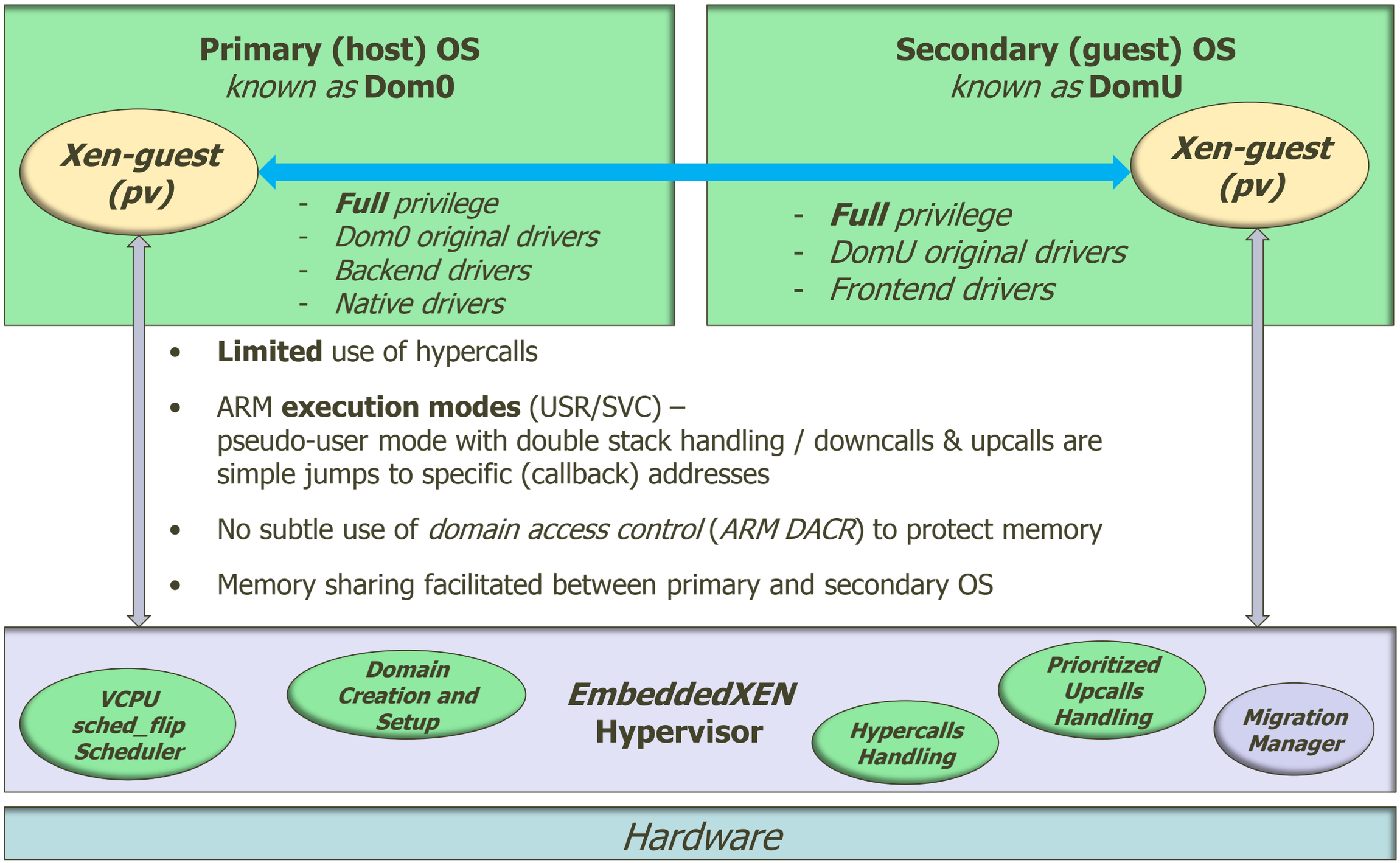
- Focus on **heterogeneity** of embedded devices
  - Idea to **re-use OS & applications** from old devices to recent devices
  - Time-to-market **migration to new hardware** generation
  - Dealing with **various cross-compiled binaries** (ARM v5-v7)
  - Dealing with **different peripherals**
  - **Less emphasis** on security aspects
- **Publicly** available
  - *https://sourceforge.net/projects/embeddedxen*

Background

- Hardware constraints
  - Low latency, reactivity (response time)
  - ARM cores **do not support** virtualization mechanisms
    - ➢ not easy to deal with various levels of execution modes

- Para-virtualization remains attractive
  - **About 30 files** to be (slightly) adapted
  - Low execution overhead
  - Efficient processing of downcalls/upcalls with support of domain interactions
    - ➢ Physical interrupts are quickly processed in *dom0*.

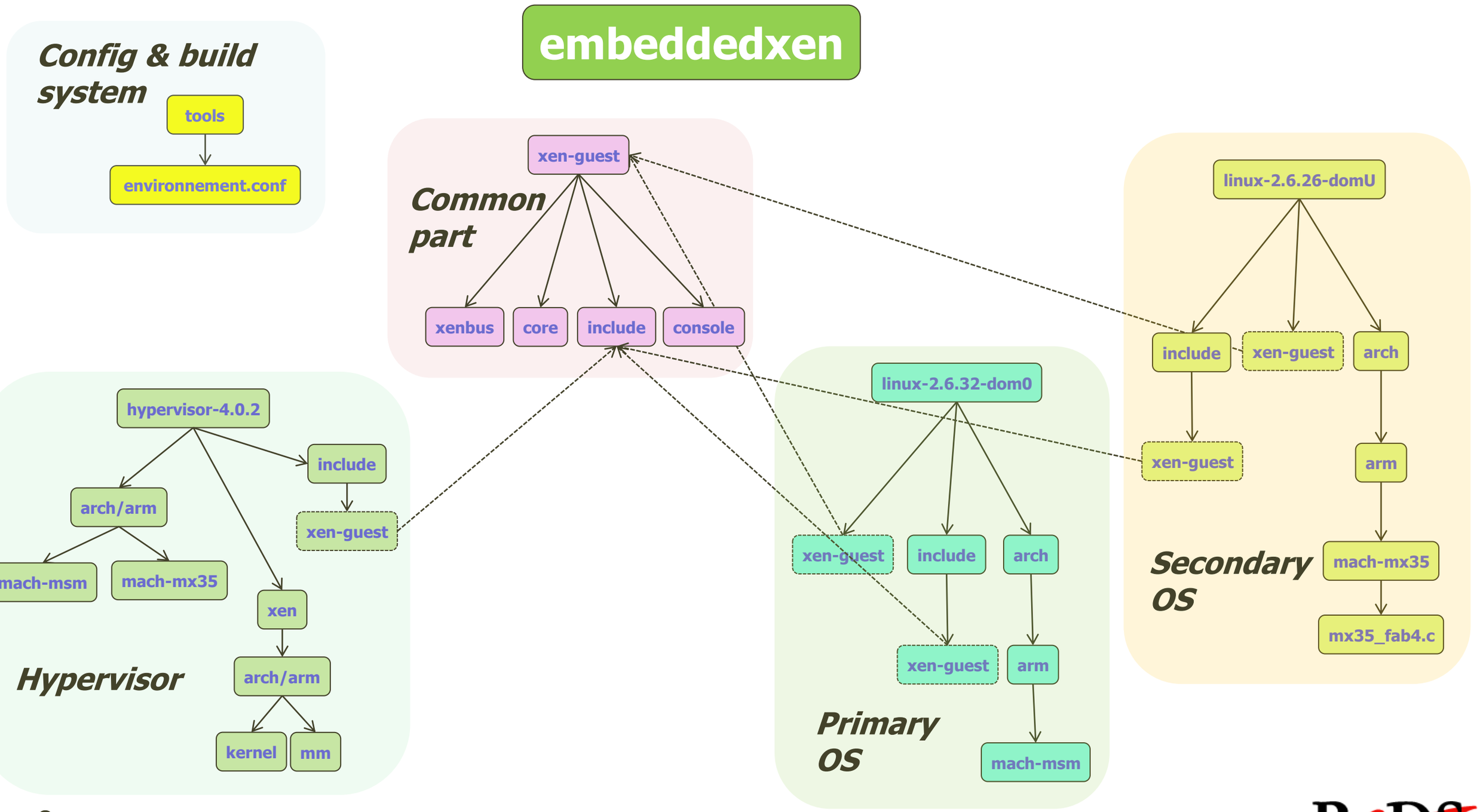
Overview of *EmbeddedXEN*



- **Limited** use of hypercalls
- ARM **execution modes** (USR/SVC) –
  pseudo-user mode with double stack handling / downcalls & upcalls are simple jumps to specific (callback) addresses
- No subtle use of *domain access control* (*ARM DACR*) to protect memory
- Memory sharing facilitated between primary and secondary OS

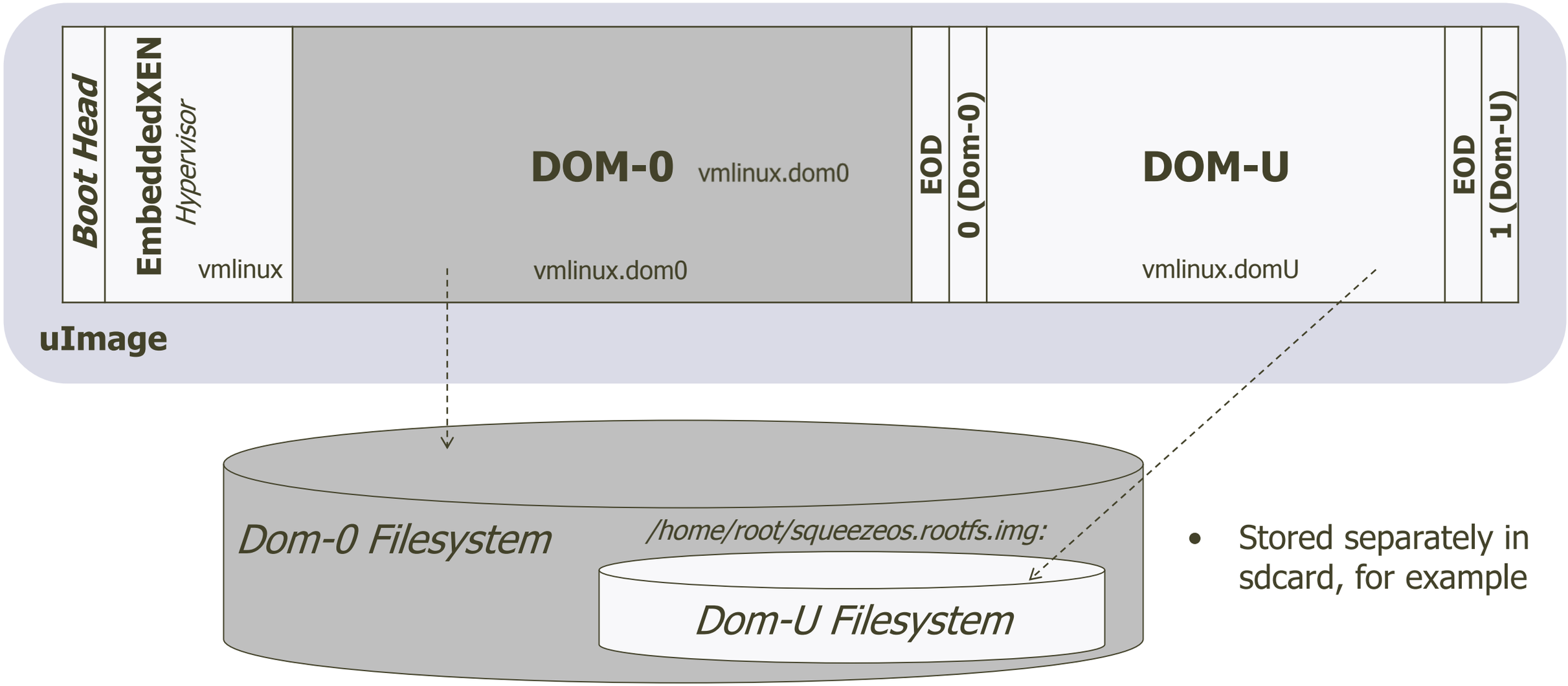
Overview of *EmbeddedXEN*

- *Linux*-like source tree and build system (*Makefiles*)



Overview of *EmbeddedXEN*

- Single binary (multi-kernel) image
- Automatic parsing & image relocation during hypervisor bootstrap



- Stored separately in sdcard, for example

Protection & Memory Isolation

- Memory isolation between domains relies on different address space isolation.
  - No further advanced mechanisms to protect hypervisor and guest memory
  - **The guest OS kernel runs at the same privilege as the hypervisor.**

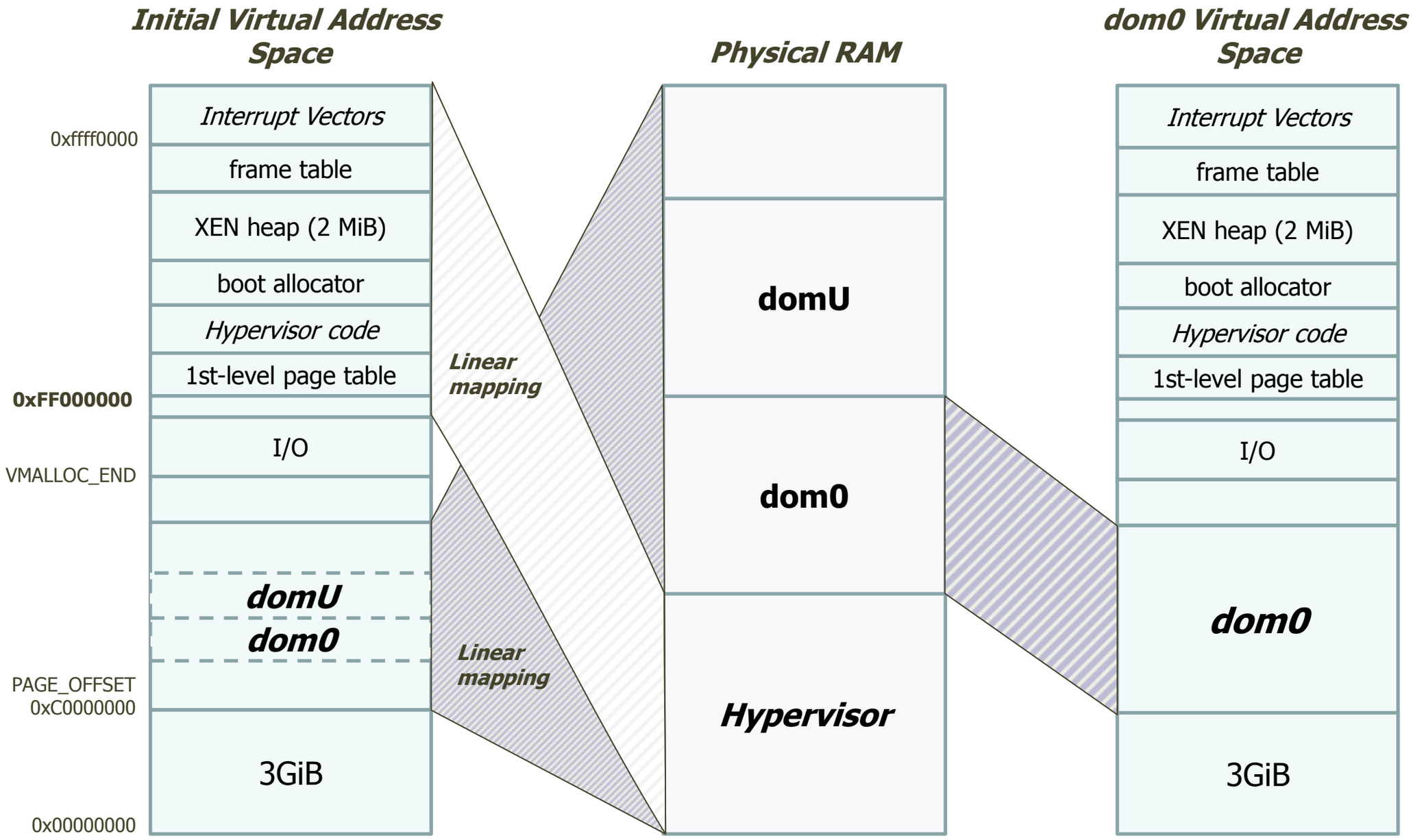
- Each domain receives its own (contiguous) physical memory region during domain set up.

- No pagination of the kernel linear address space is performed.
  - *Paravirt* of memory management is kept minimal.

- Guest OS kernel has access to the whole memory.
  - No protection mechanism for strong isolation of VM (but is it really necessary?)

Protection & Memory Isolation

- Virtual & Physical Memory Layouts

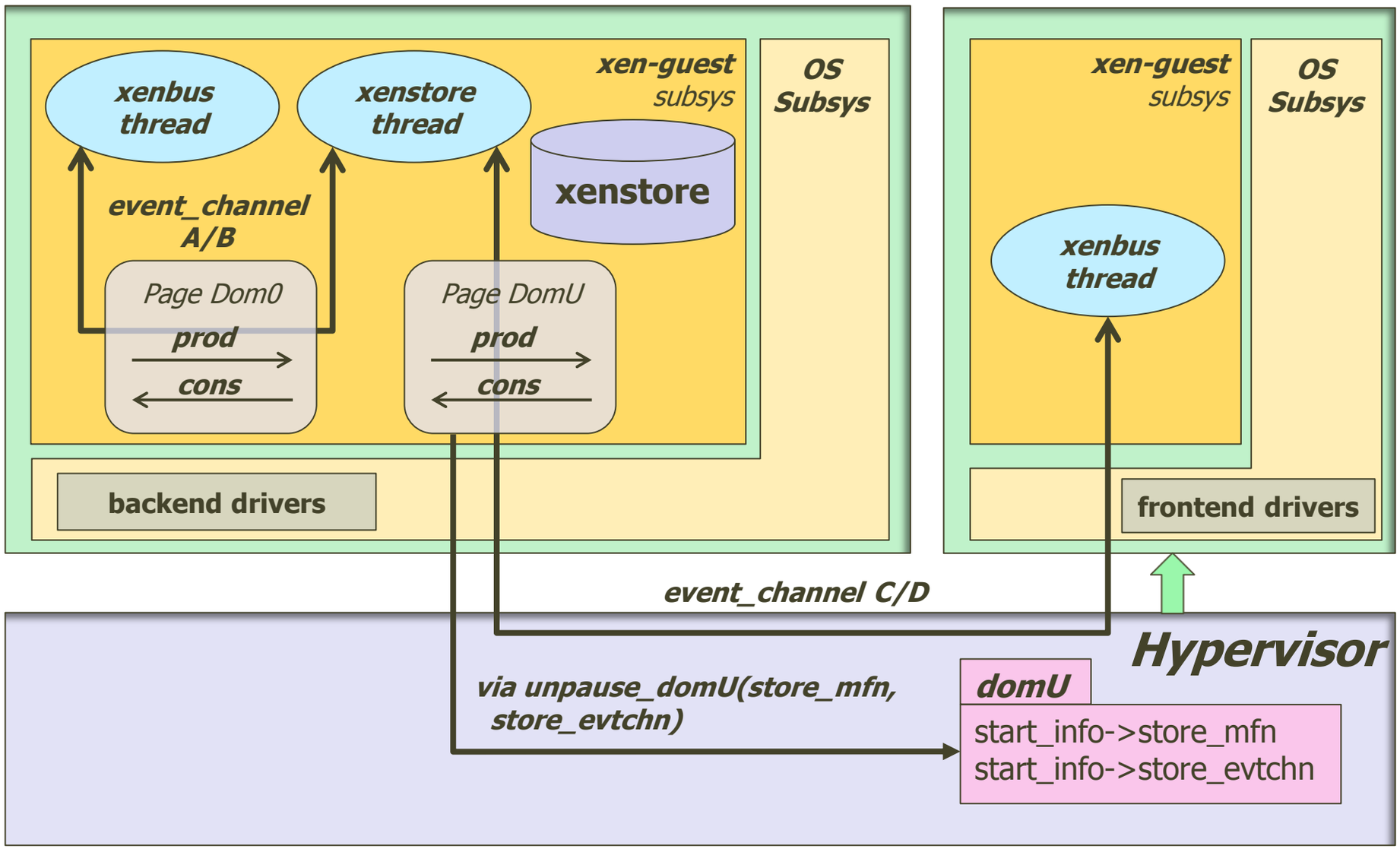


Domain Interactions

- Virtualization of peripherals in *EmbeddedXEN* is quite similar to the existing mechanisms in XEN.
  - Driver split with frontend & backend drivers
  - Communication with *xenbus*
  - Use of grant tables for sharing/copying pages between domains
- However, a revisited (simplified) implementation of these mechanisms have been achieved in *EmbeddedXEN*.
  - XEN store is dynamically allocated at boot time of guest OSes.
  - No user space tools are required to manage XEN store entries or peripherals configs.
  - *Hotplugs* & dynamic configs of peripherals are less relevant to embedded systems.

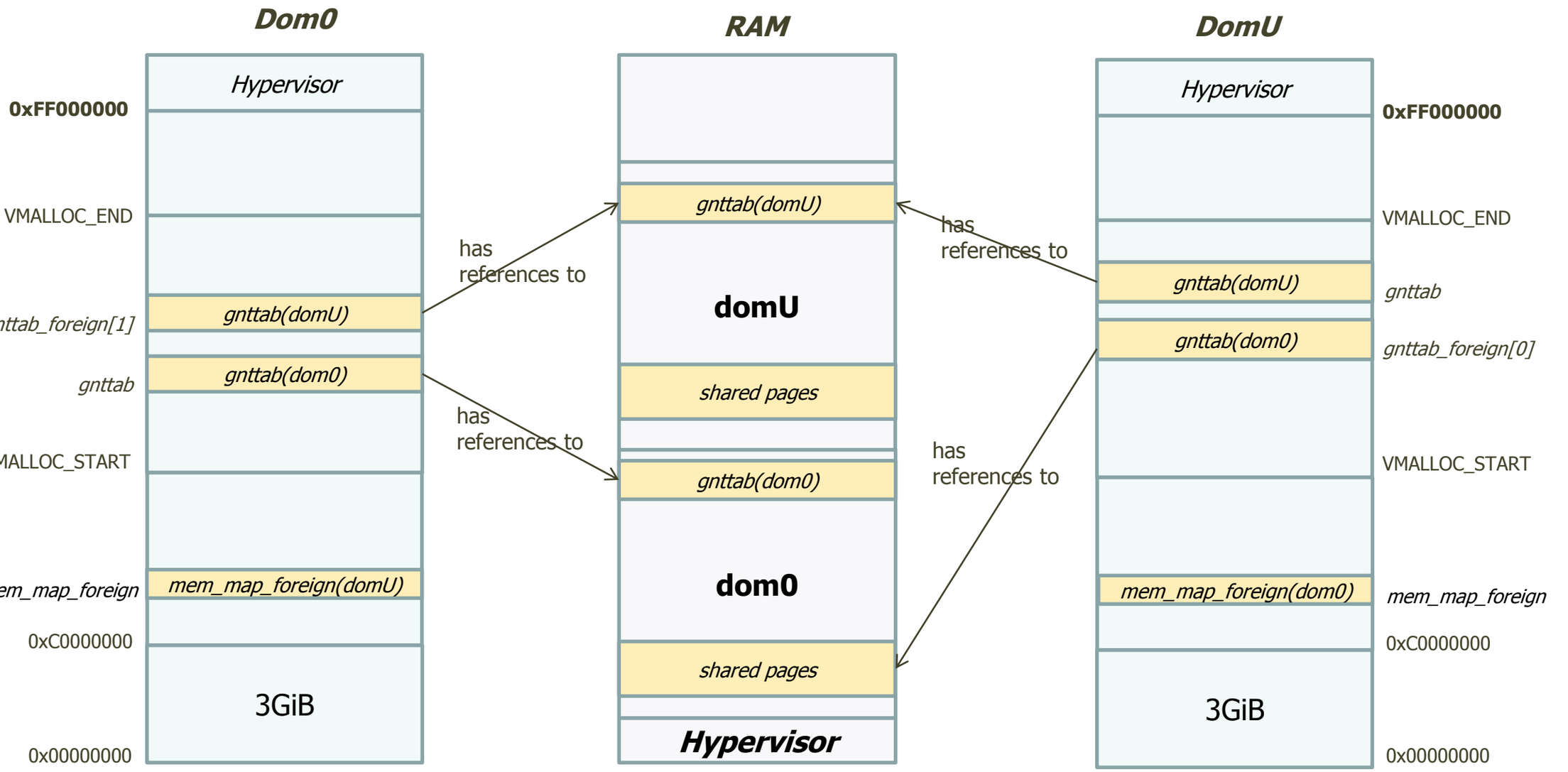
Domain Interactions

- *domU* is passed information during bootstrap under control of *dom0*.



Domain Interactions

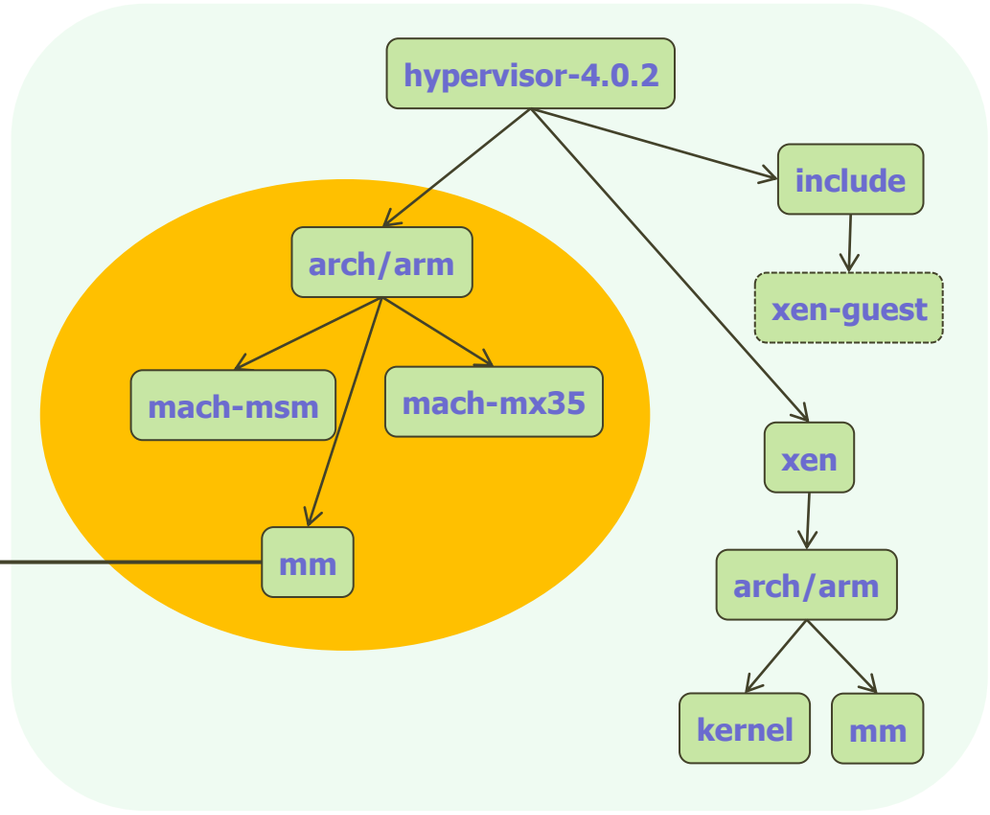
- Grant tables are used in a different way
  - Shared pages are possible only in the *vmalloc'd* area.
  - Kernel linear addresses are not shareable; contents needs to be copied using temporary mappings.



Device Heterogeneity

- Different levels of heterogeneity
  - At CPU level: various instructions sets (locks, cache, etc.), various PTEs (MMU) flags, various co-processors, etc.
    - ➢ Compatibility ensured via hypercalls
  - At peripherals level: not the same hardware
    - ➢ Compatibility ensured via backend driver processing

built-in.o **cache-v6.S** **cache-v7.S** Kconfig Makefile mm.h proc-macros.S **tlb-v6.S** **tlb-v7.S**



Device Heterogeneity

- Example of ARMv6 running on ARMv7 CPU (iMX35 -> HTC Desire HD)

Original version

linux-2.6.26-domU/arch/arm/mm/cache-v6.S:

```
ENTRY(v6_flush_kern_cache_all)
  mov         r0, #0
#ifdef HARVARD_CACHE
  mcr         p15, 0, r0, c7, c14, 0    @ D cache clean+invalidate
#ifdef CONFIG_SMP
  mcr         p15, 0, r0, c7, c5, 0     @ I+BTB cache invalidate
#else
  b           v6_icache_inval_all
#endif
#else
  mcr         p15, 0, r0, c7, c15, 0    @ Cache clean+invalidate
#endif

  mov         pc, lr
```

Paravirt (PV) version

linux-2.6.26-domU/arch/arm/mm/cache-v6.S:

```
.extern xen_flush_kern_cache_all
ENTRY(v6_flush_kern_cache_all)
   b          xen_flush_kern_cache_all
#if 0 /* paravirt */
  mov         r0, #0

  …

  mov         pc, lr
#endif /* 0 */
```

linux-2.6.26-domU/xen-guest/hypervisor.c:

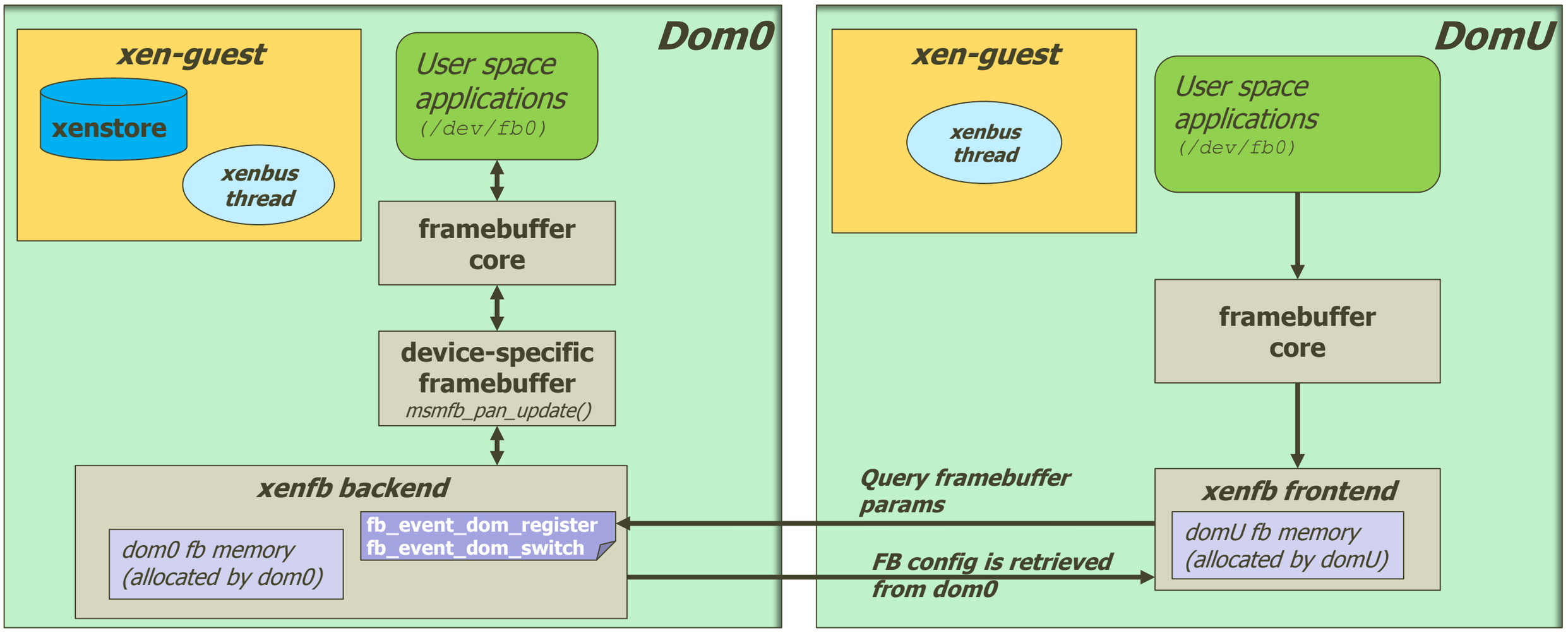
```
void xen_flush_kern_cache_all(void)
{
  struct mmuext_op op;

  op.cmd = MMUEXT_FLUSH_CACHE;
  HYPERVISOR_mmuext_op(&op, 1, NULL, DOMID_SELF);
}
```

hypervisor-4.0.2/arch/arm/mm/cache_v7.S:

```
ENTRY(xen_flush_kern_cache_all)
  stmfd       sp!, {r4-r5, r7, r9-r11, lr}
  bl          xen_flush_dcache_all @ much more complex!!
  mov         r0, #0
  mcr         p15, 0, r0, c7, c5, 0     @ I+BTB cache invalidate
  ldmfd       sp!, {r4-r5, r7, r9-r11, lr}
  mov         pc, lr
ENDPROC(xen_flush_kern_cache_all)
```

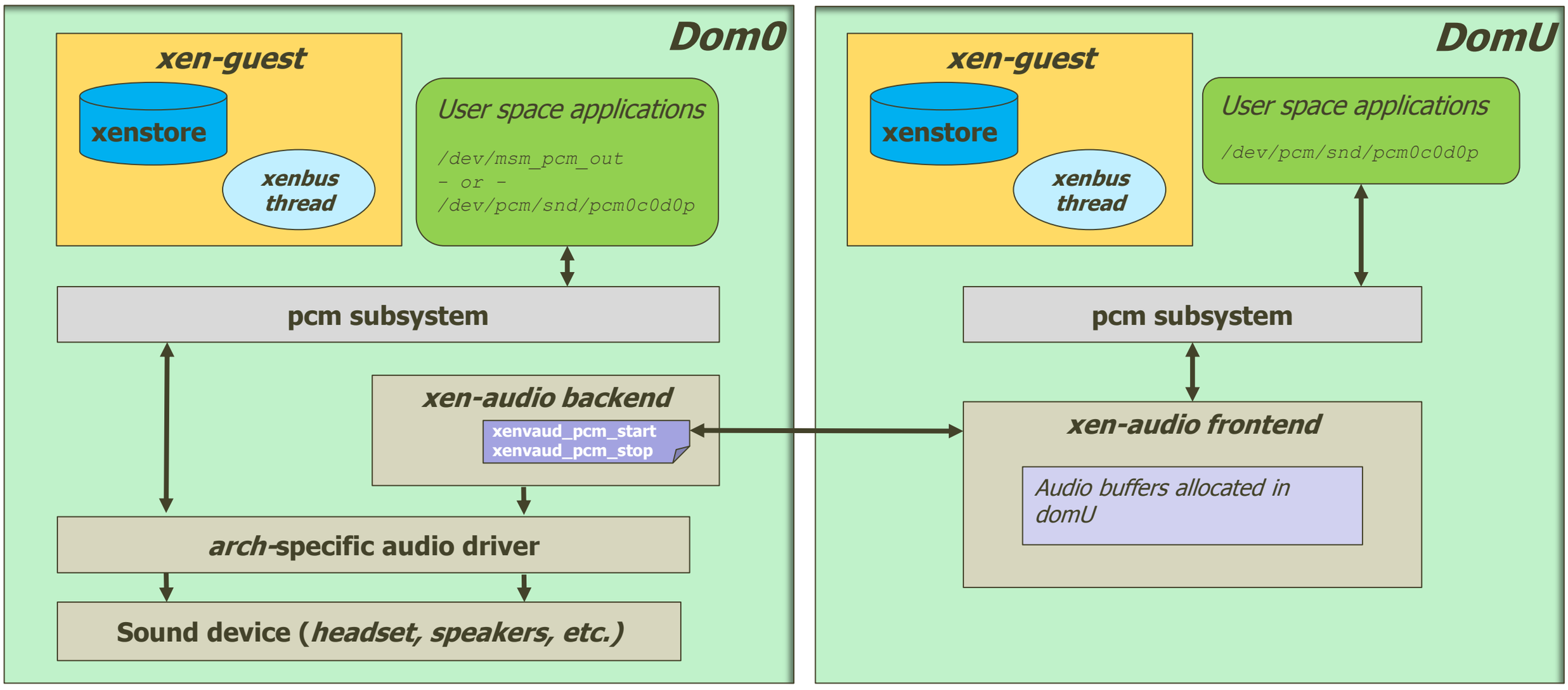
Device Heterogeneity

- Example of *framebuffer device* heterogeneity
- Configuration of framebuffer is retrieved from *Dom0* via *xenbus*



Device Heterogeneity

- Example of *audio device* heterogeneity
- *DomU* audio buffers are accessed from *Dom0* via shared pages.



Conclusions & Future Work

- *EmbeddedXEN* is an embedded virtualization framework which puts emphasis on **efficient** and **heterogeneous hardware**.

- Application environments **can be re-used "as such"** on modern platforms (*Android*-based for example) **taking advantage** of last generation hardware.

- *EmbeddedXEN* relies on the main principles of XEN, with a **revisited lightweight**, but ***less*** secure architecture.

- A **single multikernel binary image**, *easy to deploy on the target platform* **without additional tools**, makes *EmbeddedXEN* well tailored to embedded systems.

Conclusions & Future Work

- Further investigation projects:
  - Elaboration of a *domU* using a **graphical desktop** for user applications
  - Support of **multicore** ARM CPUs (cortex-A9, cortex-A15)
  - **Live migration** of *domU* using remote NFS-filesystem (migration within a cloud)
  - Support of **hard realtime** OS (*RTEMS*-paravirt)

- Thanks for your attention!

- Further information: ***daniel.rossier@heig-vd.ch***