



## Linux Mainframe - KVM and the Road of Virtualization

Operating System Research Center  
André Przywara  
Jörg Rödel

May 30th, 2007

- VMware, Xen and KVM – the different approaches
- Paravirtualization – Linux as a Guest
- KVM – The new In-Linux Hypervisor
- The Linux Mainframe – KVM in the future

# Virtualization dictionary

- Host: physical machine
- Guest: emulated or virtualized machine
- Hypervisor: System software that manages virtualization
- Paravirtualization: guest is modified to ease or allow virtualization
- Full virtualization: allows to run unmodified OS kernels
- Dom0: XEN term for privileged guest which helps hypervisor, must be paravirtualized
- DomU: XEN term for unprivileged guest, can be fully or para-virtualized

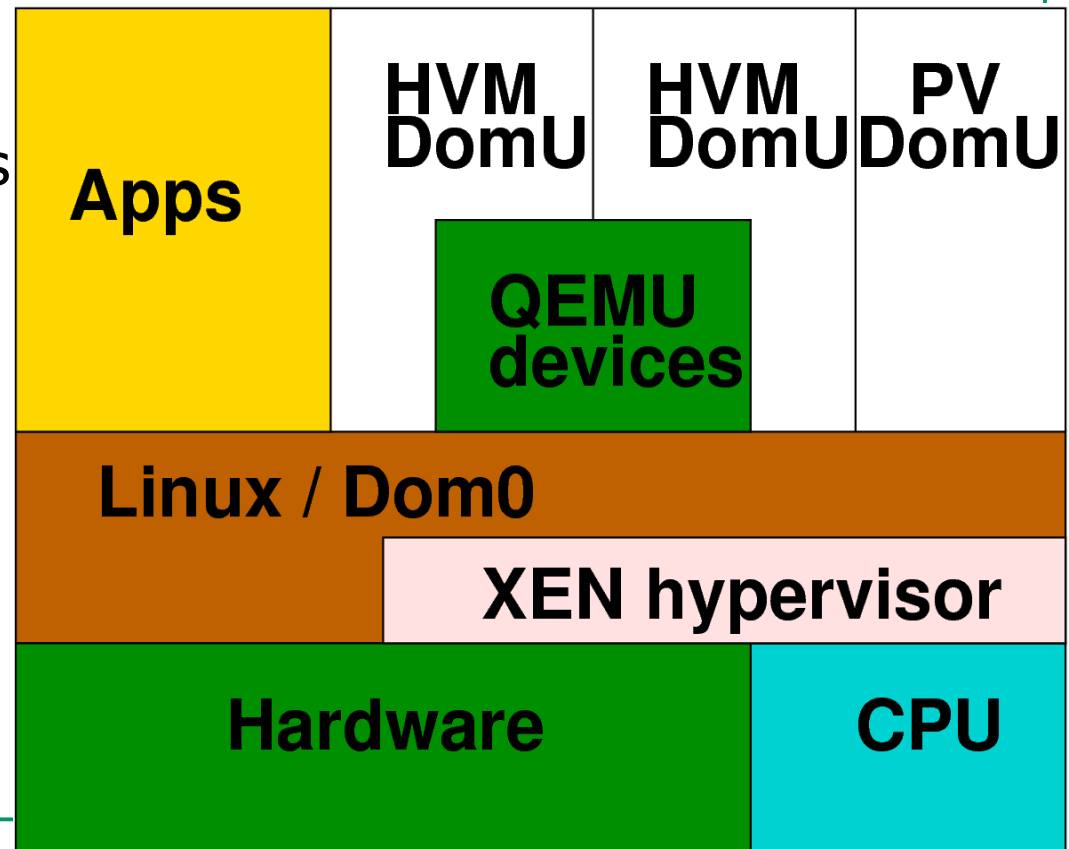
# VMware, Xen and KVM – The different approaches

# The VMware approach

- Very little documentation on real implementation
- VMware uses dynamic binary translation
- Most code will be executed natively, critical sections will be modified and replaced by VMware's replacement code
- Paravirtualized drivers to gain performance and comfort
- Known OS kernel versions are modified in advance
- Already translated code parts are cached
- Many tweaks to improve performance
- Complex solution

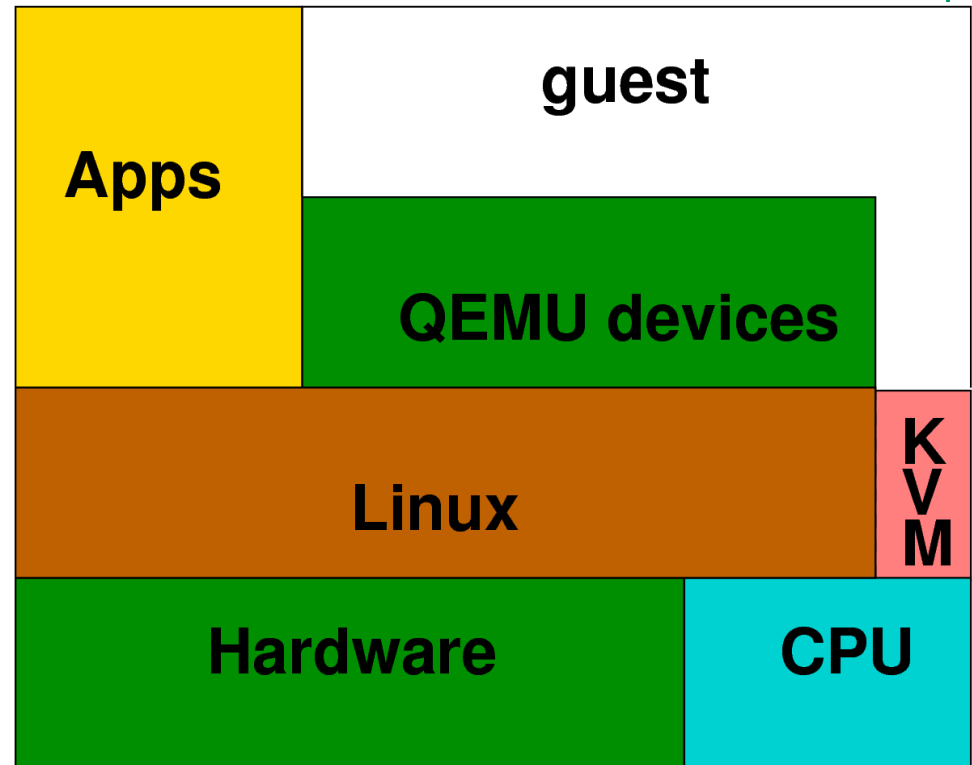
# Paravirtualization with Xen

- The Xen hypervisor is a small layer beneath a modified kernel (Dom0)
- Dom0 controls the hypervisor and provides the hardware drivers
- PV guests do hypercalls for I/O
- Hardware virtualized domains see QEMU's emulated devices



# Full virtualization with KVM

- New development to start Operating System guests directly on top of Linux
- Uses full virtualization, does not require the guest to be modified
- QEMU is used to emulate the hardware devices
- KVM relies on hardware virtualization extensions
- Small and lean implementation



# Comparison

	VMware Server	Xen	KVM
Software complexity	High	Moderate	<b>Low</b>
Installation effort	Fair	High	<b>Low</b>
I/O speed	<b>Fast</b>	Fast/Slow	Slow
Maturity state	<b>Mature</b>	Stable	Unstable
Requires HW virtualization	No*	<b>No/Yes</b>	Yes
Supports HW virtualization	Yes**	<b>Yes</b>	<b>Yes</b>

\* not on Intel hardware for 64bit guests

\*\* only on Intel hardware for 64bit guests

- Weak points of KVM are being addressed
  - Paravirtualized drivers improve I/O performance
  - Paravirtualized execution mode does not require HW virtualization
  - Maturity will improve naturally

# Paravirtualization – Linux as a Guest

# Linux on Xen

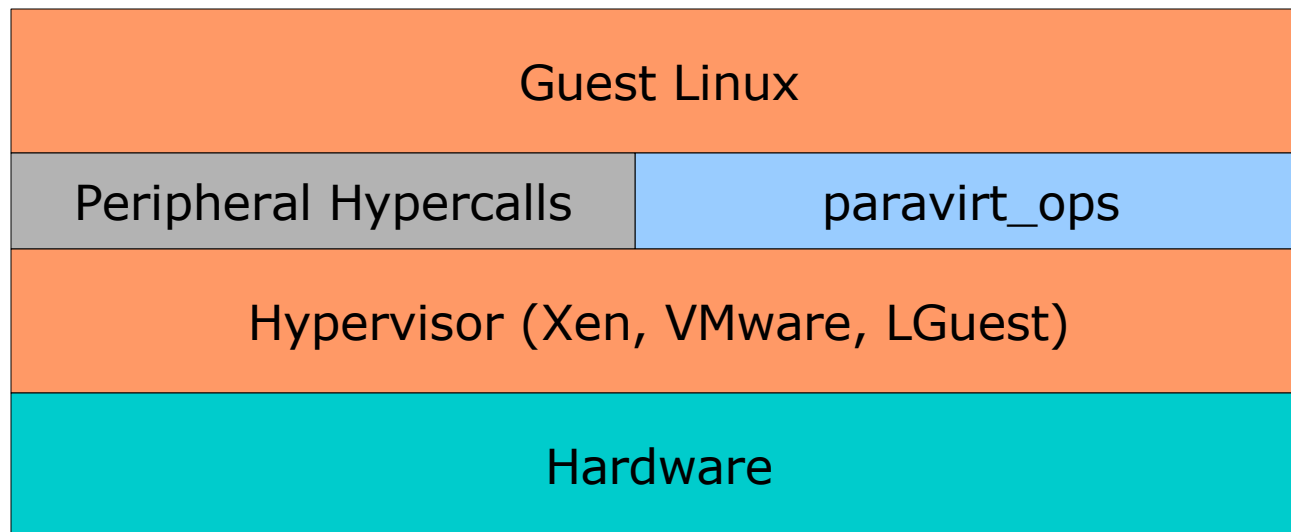
- system critical instructions are replaced with hypercalls
- Hypercalls use a ring buffer to transfer commands and data
- Console, network and block drivers are virtualized on a higher level
- Xen maintains a set of patches against some Linux kernel (currently 2.6.18), keeping pace is a pain

# The Linux paravirt\_ops Interface

- New interface inside the Linux kernel to wrap specific hardware accesses
- These accesses can be redirected to the hypervisor
- Backend can be replaced at runtime to support different hypervisors
- On i386 it contains 92 different operations
- Currently a backend for VMware is in mainline Linux
- Backends for Xen and LGuest are available

# The Linux paravirt\_ops Interface

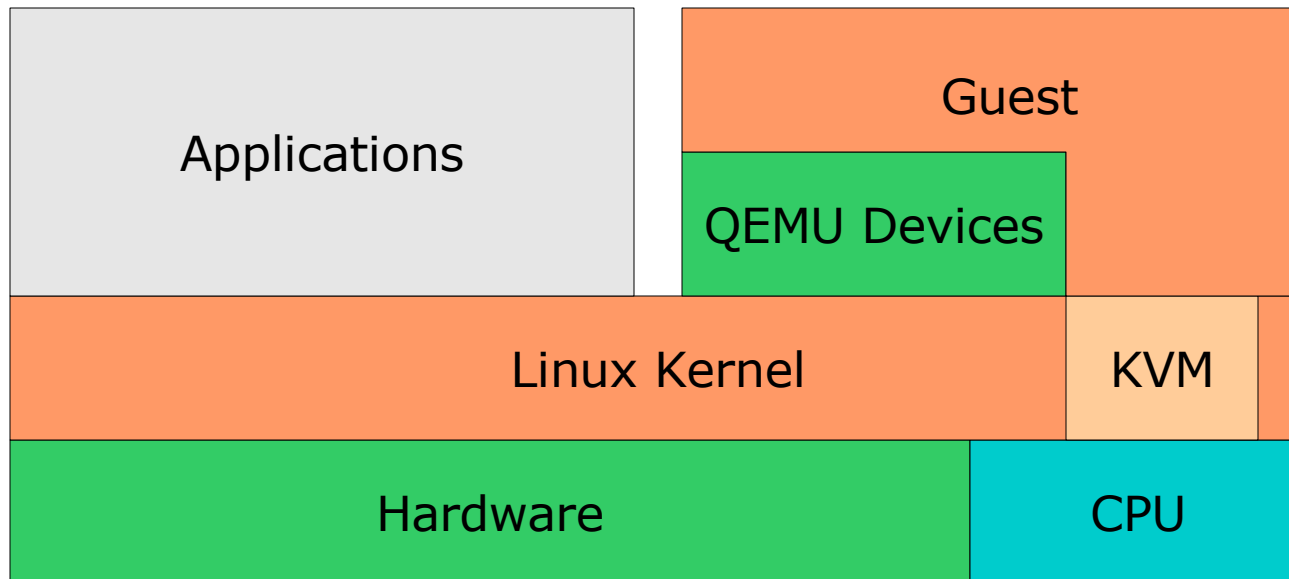
- Implements only CPU and timer specific hardware accesses
- Only mainline for i386 architecture at the moment
- AMD64 implementation is on the way



# KVM – The new In-Linux Hypervisor

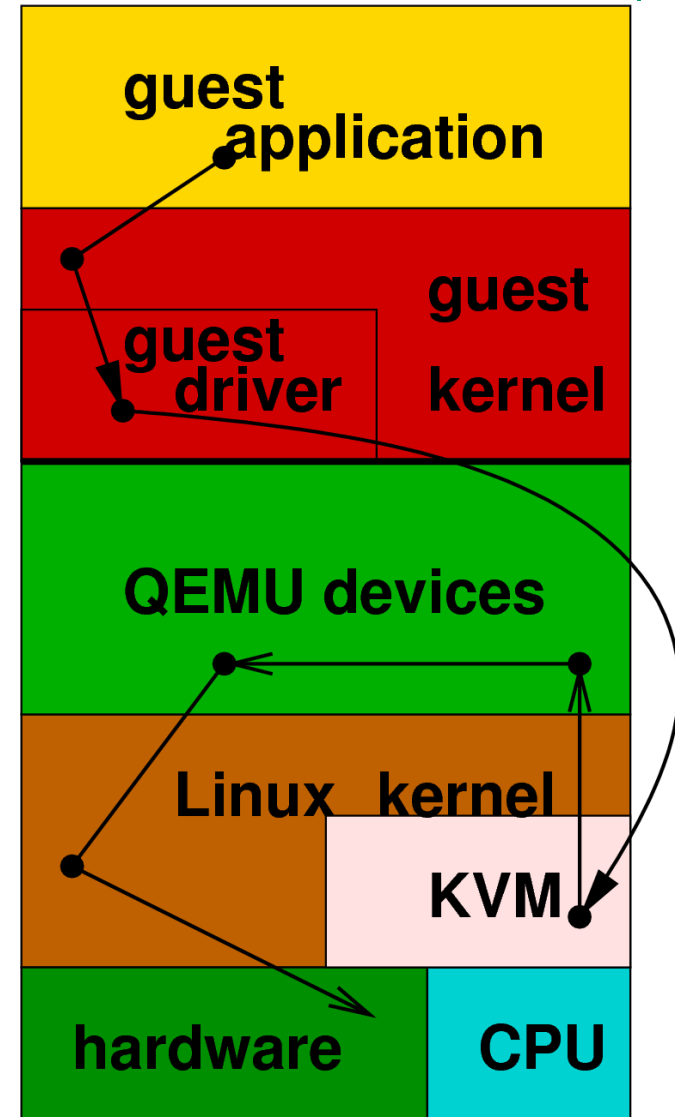
# KVM in the Linux World

- Mainly implemented in the Linux kernel
- Hardware devices emulated by QEMU
- Guest runs as normal application within Linux



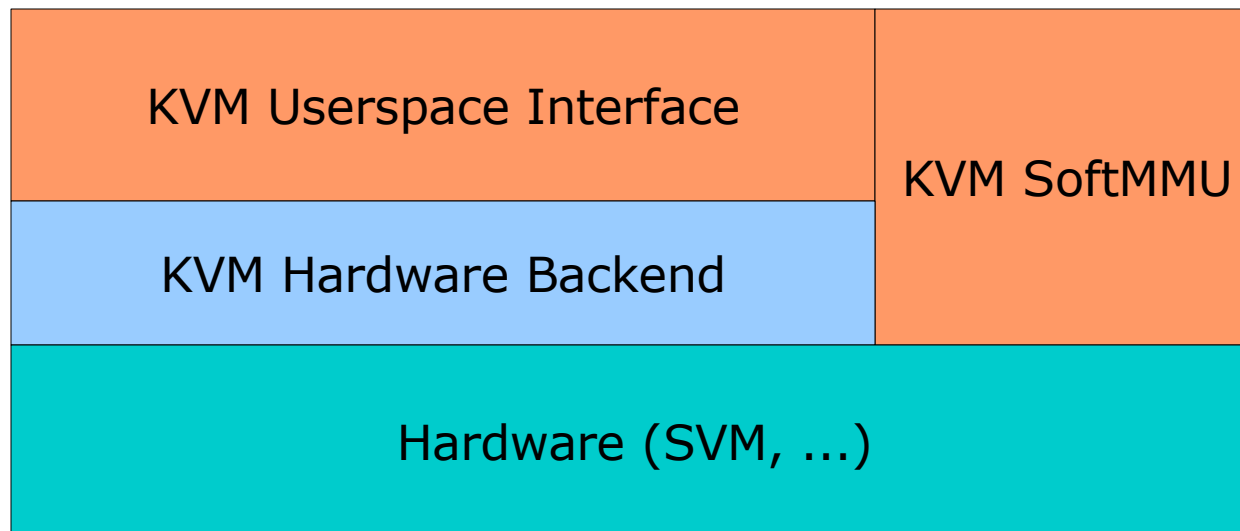
## The QEMU part

- Guest hardware accesses are intercepted by KVM
- Cause exit to Host Mode and userspace
- QEMU emulates hardware behavior of common devices (RTL 8139, PIIX4 IDE, Cirrus Logic VGA)
- Paravirtualized drivers for network and block devices are currently under development



# The Structure of the KVM Kernel Part

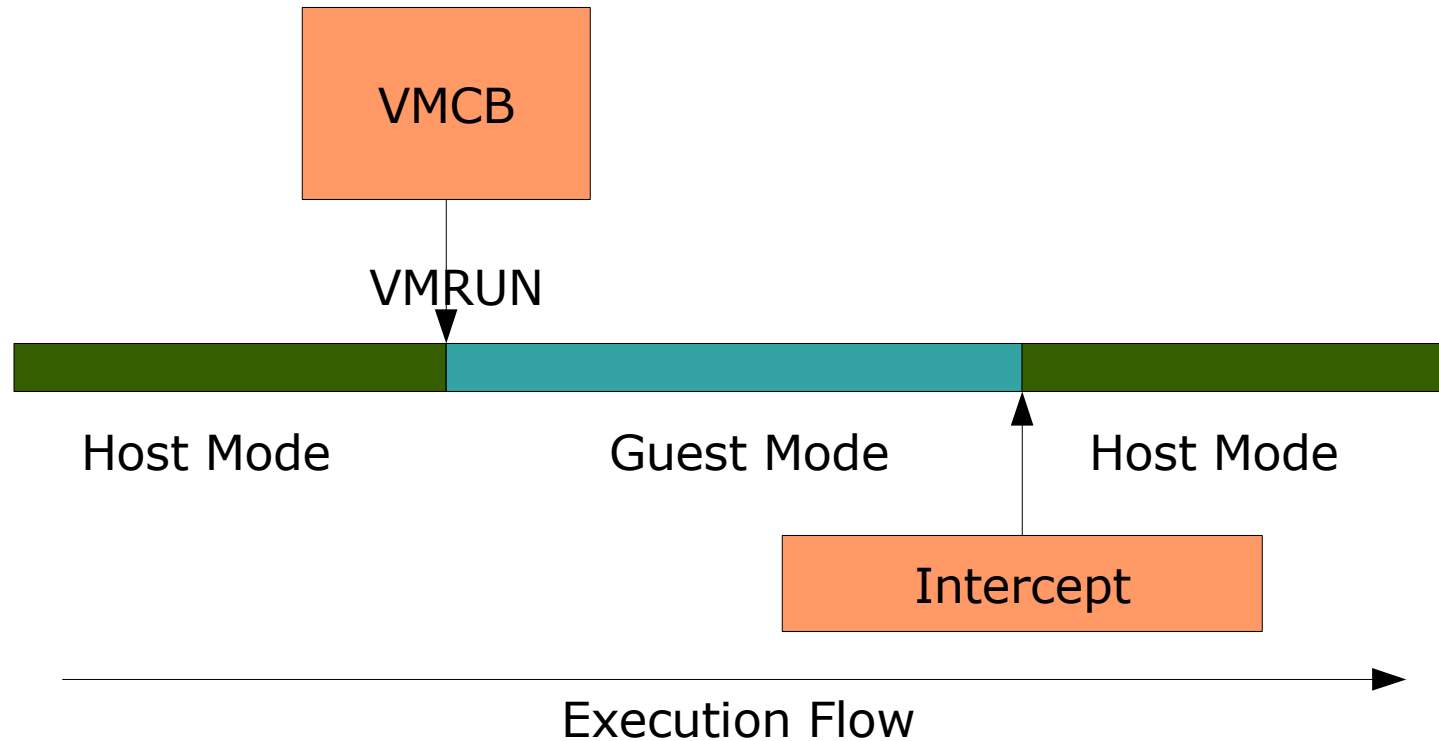
- Implemented as a set of kernel modules
- Has a generic part which implements common things like userspace interface and shadow paging
- Different backends for various hardware (like the AMD virtualization extension - SVM)



# The AMD-V Virtualization Extension

- Is an extension to recent AMD processors to support execution of multiple Operating Systems
- Introduces **Guest Mode** to the processor
- Guest is described with **Virtual Machine Control Block (VMCB)**
- New **VMRUN** instruction to switch processor to Guest Mode
- **Intercepts** switch processor back to **Host Mode**
- Hypercalls as explicit call from the Guest to the hypervisor with **VMCALL** instruction

# AMD-V Execution Example



# KVM and Hardware Virtualization

- Sets up a VMCB for every guest virtual CPU
- KVM switches processor to Guest Mode
- On intercept control is returned to KVM
- Examines if the intercept can be handled in KVM itself
- Otherwise control is passed to QEMU
- After intercept is handled guest is continued

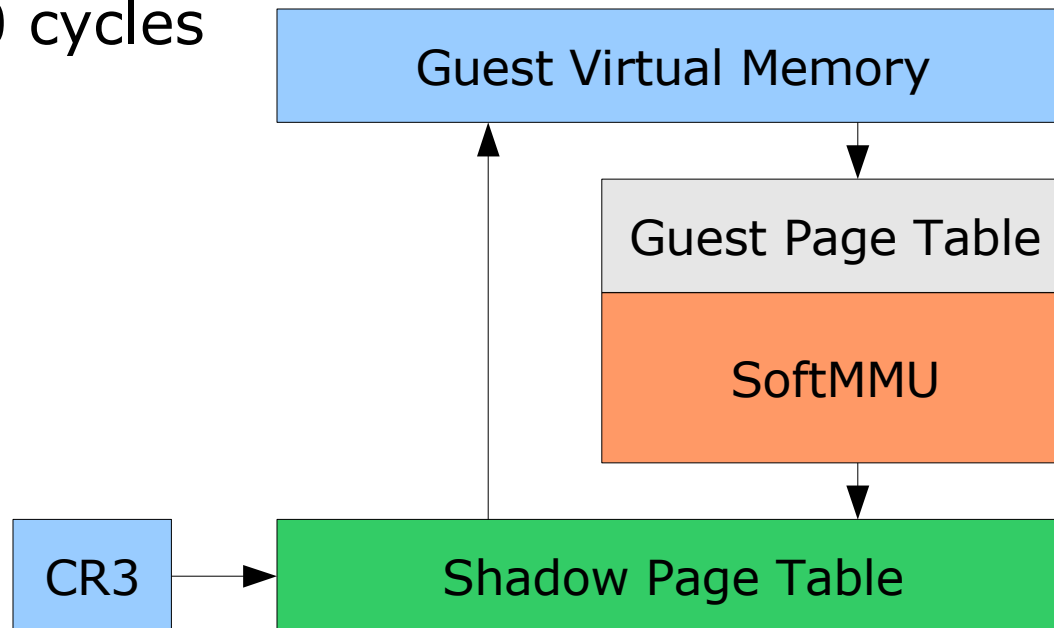
## Current Status of KVM

- Boots most unmodified Operating Systems supported by QEMU
- Runs with nearly full processor speed due to hardware virtualization
- Support for live migration
- I/O is still slow due to device emulation
- Limitations in number of guest processors and guest RAM
- Still under heavy development

# The Linux Mainframe – KVM in the Future

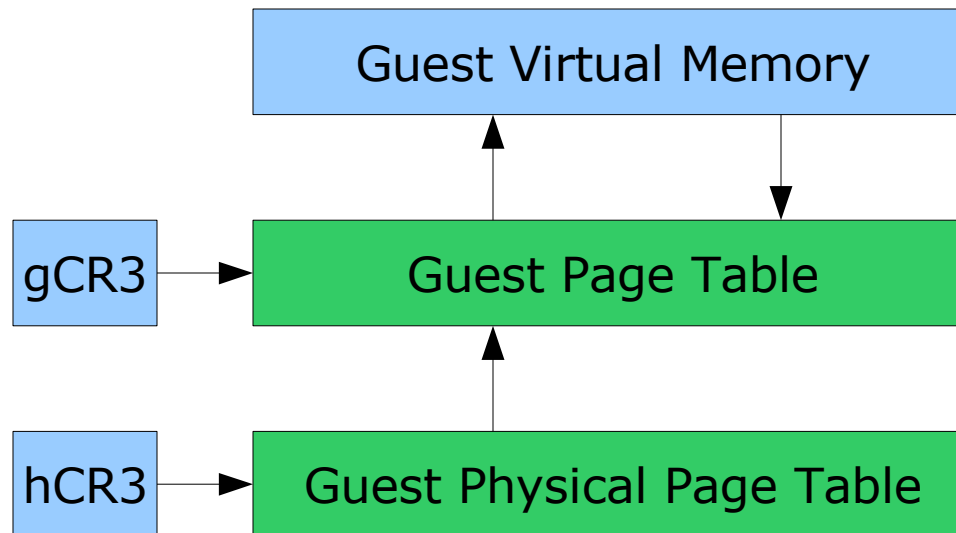
# The KVM SoftMMU

- Guest sets up its own page tables
- KVM has to build a shadow page table for the processors MMU
- Building one entry in the shadow page table costs ~6000 cycles



# Support for Nested Paging

- Building the shadow page table is an expensive operation
- More efficient to provide a page table which maps host physical to guest physical memory
- Feature will be available in AMD K10 cores (Barcelona)

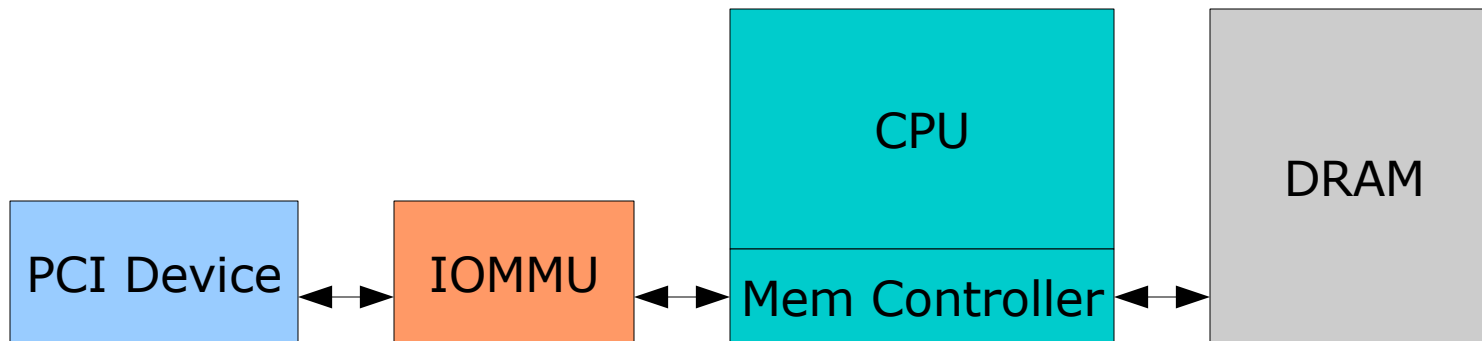


## KVM PCI Access

- Currently KVM guests can not access PCI devices of the host directly
- Most of the PCI card interface can be virtualized (config space, MMIO, IO ports, interrupts)
- Problem exists with DMA because the device only accesses host physical addresses
- For virtualization of DMA the devices need access to guest physical addresses
- Other problems: interrupt sharing, MSI

# PCI DMA with IOMMU

- To translate the device addresses into guest physical addresses another MMU is required
- So called IOMMU was invented for this
- Located between the device and memory to redirect accesses



# KVM Paravirtualization and LGuest

- LGuest is another In-Linux hypervisor and works completely paravirtualized
- Allows to boot a paravirtualized guest Linux under a host Linux (Linux on Linux)
- Currently supports i386 architecture
- It is not included in mainline yet
- Mainly developed by Rusty Russell

## LGuest – Merge with KVM?

- KVM developers are interested in a merge between KVM and LGuest
- This would allow KVM to boot paravirtualized Linux guests
- Lots of code to share between them (e.g. SoftMMU)
- may also share the paravirtualized drivers
- LGuest people seem not to be interested at the moment
- Thus its not sure if they will ever merge

# Conclusions

- KVM looks very promising
- usability depends on your needs and expectations
- lean concept makes installation easy
- currently on the way to production use
- major issues are being addressed

**GOOD LUCK KVM!**

## Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2007 Advanced Micro Devices, Inc. All rights reserved.