# Long term analysis in HEP: Use of virtualization and emulation techniques


Yves Kemp

DESY IT


First Workshop on Data Preservation and Long Term Analysis in HEP, DESY 26.1.2009

# Outline

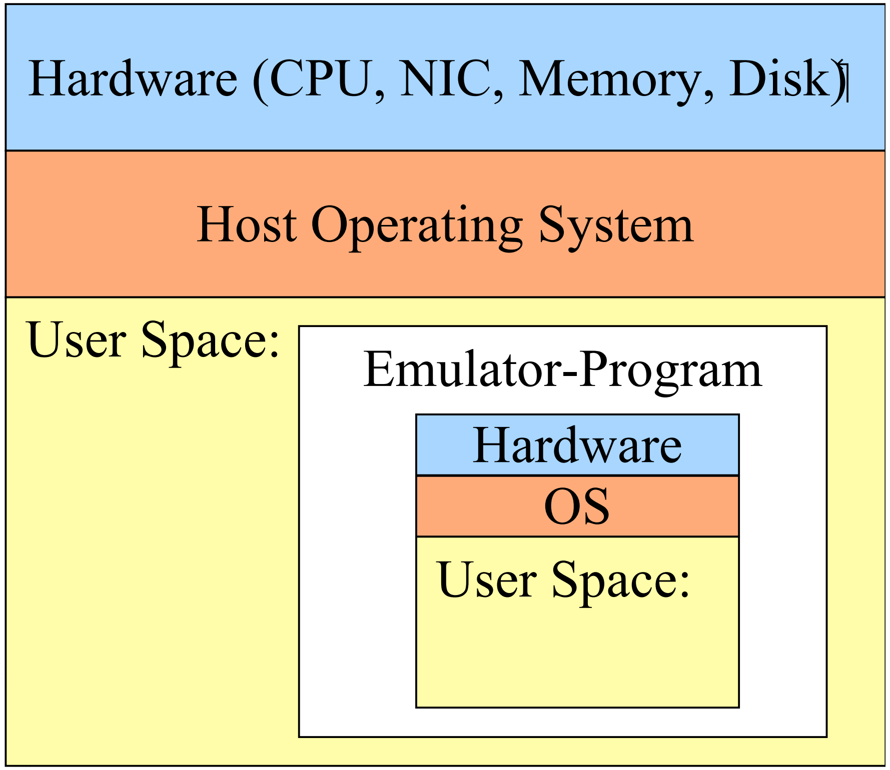
- Why virtualization and emulation?
- What is available now?
  - Products
  - Systems and workflows
- Who is working on what?
  - Some projects
- Two possible scenarios and discussion

# Why virtualization and emulation?

- Why virtualization and emulation?
  - To analyze old data with old software on old OS platform, with new hardware
  - “Keep alive” necessary services (Cond DB, Wiki (Docu), CVS?…)
- Old software only runs on old OS
  - Compilers, libs,…
- Old OS not supported by new hardware
  - Or even complete platform change
- Use virtualization, emulation or similar techniques to enable running of the old OS, and analysis software

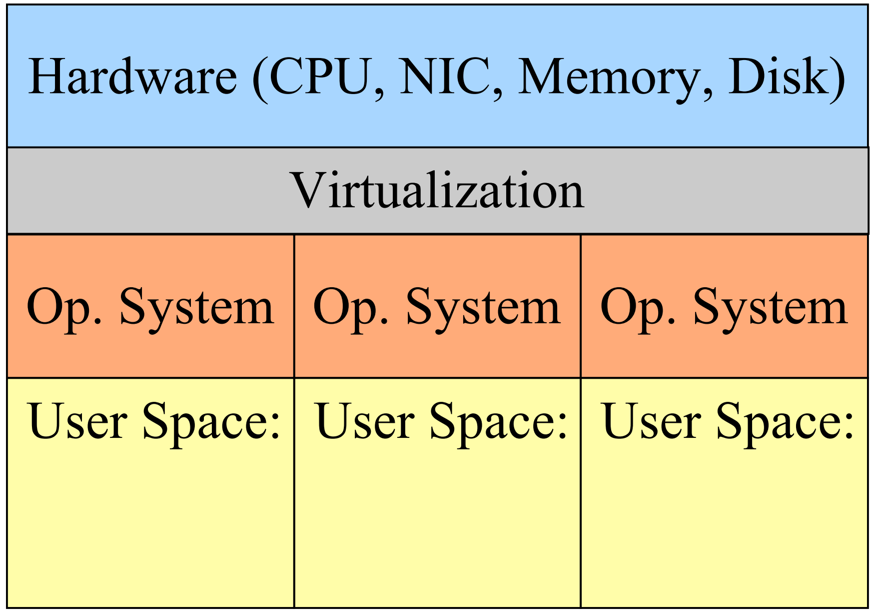
# Some definitions: Emulation

**An emulator duplicates (provides an emulation of) the functions of one system using a different system, so that the second system behaves like (and appears to be) the first system.**



# Some definitions: Virtualization

**Virtualization describes methods, which allow the resources of a computer system to divide or to combine.**




**→Virtualization can be achieved through emulation**

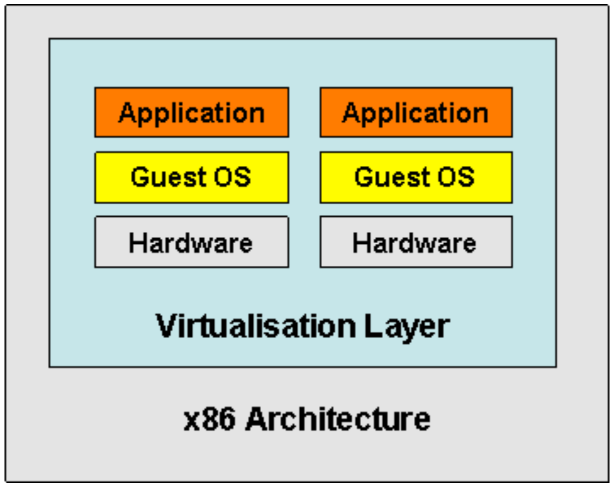
**Running of different OS more a "by-product"**
**Some virtualzation techniques are not able to do so (FreeBSD Jails)**

# Technology: "Bare metal"

Full Virtualisation, e.g. VMWare ESX (ESXi)

- Virtualization Layer is directly installed on the server hardware
- It is optimized for some certified hardware components

- Allows emulation of hardware components for the VMs by near-native performance
- Provides features like memory ballooning, over-commitment of RAM, live migration …
- Supports up to 128 powered-on Virtual Machines
- (Was?) relatively expensive

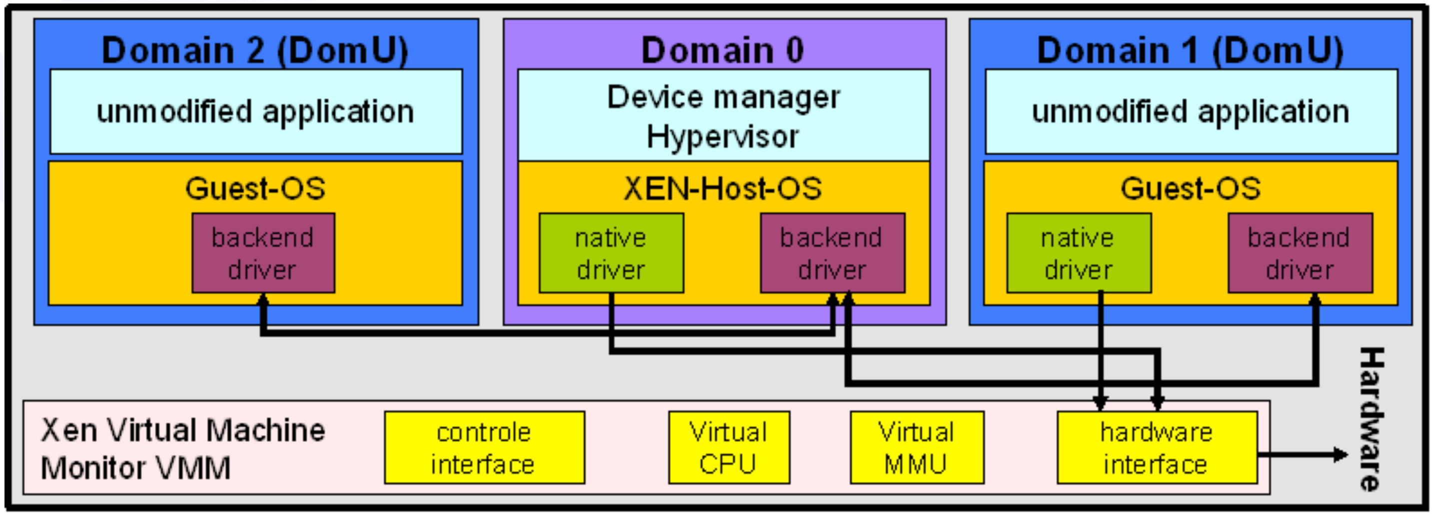



Schematic overview of VMWare ESX-Server

# Techology: Paravirtualization

## Para Virtualisation, e.g. XEN

- different hardware components are not fully emulated by the host OS. It only organises the usages → Small loss of performance
- layout of a Xen based system: Privileged host system (Dom0) and unprivileged guest systems (DomUs)
- DomUs are working cooperatively!
- guest-OS has to be adapted to XEN (Kernel-Patch), but not the applications!



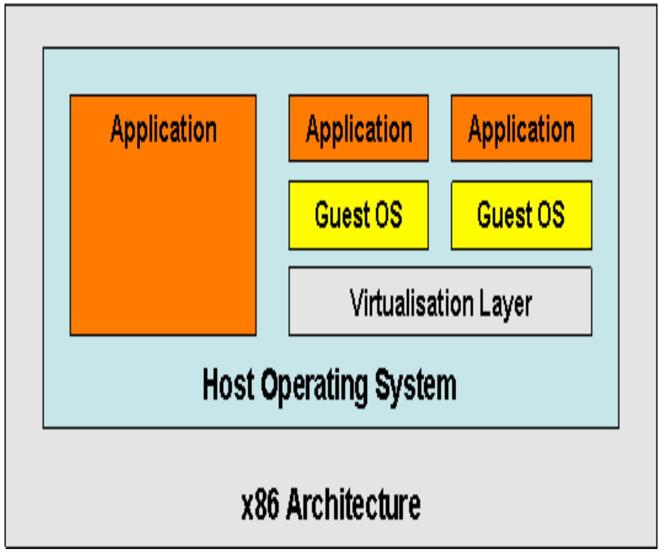
# Techology: Full virtualization

Full Virtualisation, e.g. VMware Server (formerly GSX)

- The host OS emulates all hardware components except for the CPU for the VM

  → VM becomes independent from host configuration and can be used on different host systems
- VM is stored and run in files
- VMs contain native OS and are completely isolated …

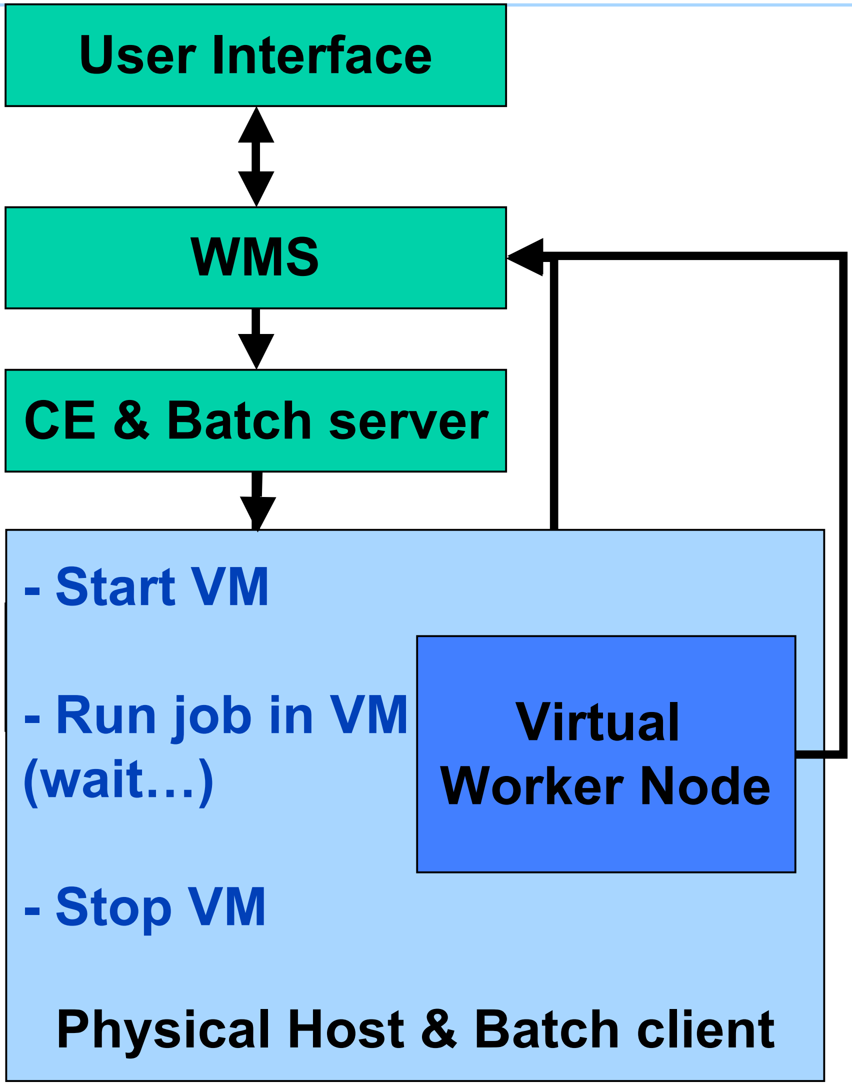
… but such hardware emulations cost performance




Schematic overview of VMWare GSX-Server

# Batch virtualization

- Cluster with multiple OS needed
- Situation at many universities
- Sites supporting many experiments
- Project: Desy, U. Karlsruhe, FHTW Berlin, SUN/SGE

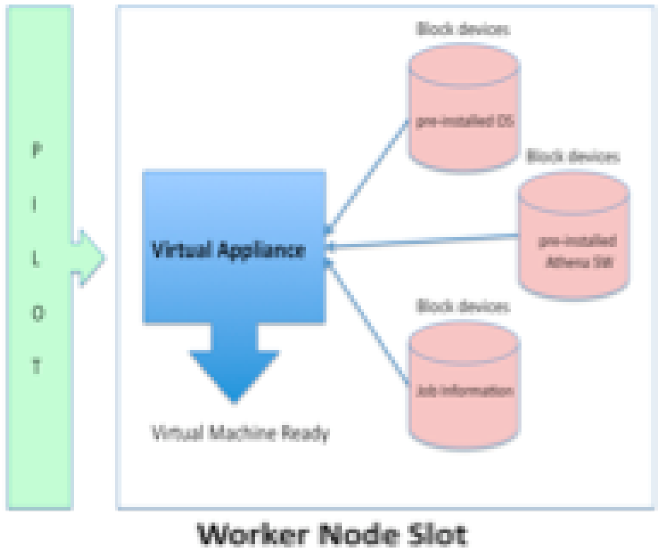


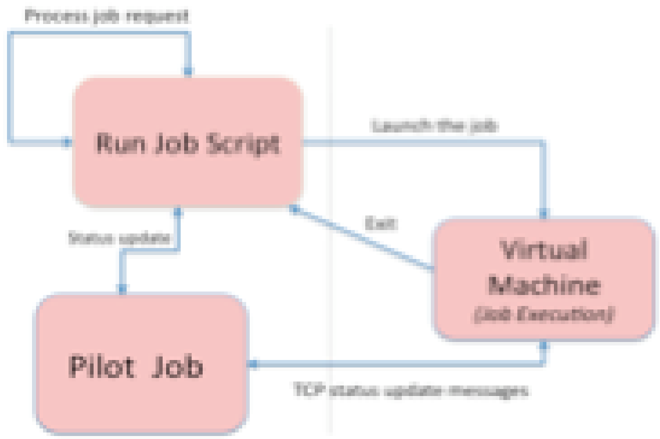
# Virtual Panda Pilot Project

- Assumption: VO takes care of backend system
  - Here: Panda: Atlas pilot job system
- Network IO performed outside of virtual machine
  - Speeds up execution
  - Possibility of decoupling analysis from data access?
- User can select images from local repository

Omer.Khalid@cern.ch

On-Demand Contextualization



# OS management @ CERN

- CERN openlab works (among other) on image generation
- Libfsimage: Python library of Linux image file system generation routines for the OS:
  - Debian, Ubuntu, CentOS, SL CERN, Fedora
- OS Farm: creates VM images and Virtual Appliances
  - Web interface
  - Uses caching mechanisms for speedup
- Content-Based Transfer
  - efficiently transfer VM image data
  - Identify common blocks in FS using checksums

H. Bjerke e.a, VHPC 08

CernVM
SoFTware Appliance

# 1.0 Release

- Available now for download from
  - http://rbuilder.cern.ch/project/cernvm/releases
- Can be run on
  - Linux (KVM, Xen,VMware Player, VirtualBox)
  - Windows(WMware Player, VirtualBox)
  - Mac (Fusion, Parallels, VirtualBox)
- Release Notes
  http://cernvm.web.cern.ch/cernvm/index.cgi?page=ReleaseNotes
- HowTo
- http://cernvm.web.cern.ch/cernvm/?page=HowTo

- Appliance can be configured and used with ALICE, LHCb, ATLAS (and CMS) software frameworks

**Slide from ACAT 2008, Predrag Buncic**

# Cloud Computing: My personal view

- **Cloud Computing relies on Virtualization techniques**
  - **Virtualization products and products around it will evolve at a raising pace in future**
- **Cloud Computing is a technology, like Cluster Computing**
  - **"Cloud Computing is Cluster Computing, with the difference, that the cluster is not in your cellar, but in Amazon's cellar"**
  - **However: New standards appear. My hope is, they will be open, and widely supported.**
- **Cloud Computing *alone* will not solve the accessibility of data for analysis**

# Interoperability and image formats

- Xen boosted virtualization usage in Linux world
  - Xen not part of vanilla kernel
  - Currently patched kernels shipped by major distributors (e.g. Red Hat)
- KVM integrated into kernel
  - Red Hat plans to migrate to KVM as virtualization format
- VMware another major player
- Need for interoperability…
  - E.g. libvirt: API for e.g. Xen & KVM
- … and exchangeable image formats
  - E.g. Open Virtualization Format (OVF): open standard for packaging and distributing virtual appliances

redhat acquires QUMRANET

# Virtualization / Emulation overhead

- Virtualization presents a moderate overhead
  - Difficult to give one single number: ~5-50%
- Emulation can have orders of magnitude overhead
- … and then there is Moore’s law…

→Today’s emulation of C64 and the like are much faster than the original C64

(e.g. Amiga forever running on EeePC)

# Two possible future scenarios

- My personal view
- Even probably incomplete
- Open for discussion
- Everything will be different anyhow:-)

# Scenario 1: "Freezing"

- At the end of the experiment:
  - Datasets closed, final reprocessing done
  - Software framework stable
- Virtual image of the OS with software is done
  - Important: Use a standardized format, like OVF
- Necessary services like Cond DB.:
  - Either integrated into images
  - Or also frozen into another image
- Data access:
  - Either maintain the old protocol/interface
  - Or use high-level protocols
- Running analysis in 20NN (with NN >> 09):
  - Start the whole ensemble of VMs

# Scenario 2: Test-driven migration

- Start during running experiment
  - Or even before, when designing software framework
- Define tests
  - In the beginning on MC data, later real data
  - Certain code, running on certain data, yields certain result (e.g. $M_{top}$=172.4 GeV/$c^2$)
- Have an automated machinery, which regularly compiles code for different OS / architectures, and runs the tests
- If test fails (e.g. compilation or execution fails, or result divergent)
  - Manual intervention: understand (and fix) problem

→ Such automated tests are usually performed using virtualization techniques and workflows

# Discussion:

- **Pro Freezing**
  - One-time effort, very small maintenance outside of analysis phase
  - Also allows software w/o code (but fails with DRM)
- **Pro Test-driven migration**
  - Usability and correctness of code is guaranteed at every moment
  - Data accessibility and integrity can be checked as well
  - Fast reaction to standard/protocol changes
  - General code quality can improve, as designed for portability and migration
- **Cons Freezing**
  - Rely on certain standards and protocols
  - Potential performance problems
- **Cons Test-driven migration**
  - Needs long-time intervention, more man-power and resources needed
  - Some knowledge of the frameworks must be passed to maintainers

# Summary & Outlook

- **Virtualization (and emulation) important in today's IT world**
- **Cloud  Computing will push virtualization even more**
- **Many ongoing projects around virtualization in HEP field already**
- **Virtualization will be necessary in some scenarios of long-term analysis**
- **BUT: Virtualization *alone* will not be sufficient**

**I would like to thank for their ideas and contributions: Havard Bjerke, Volker Büge, Predrac Buncic, Omer Khalid, Marcel Kunze, Markus Schulz, Sven Sternberger**