RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Virtualisation @ ALICE HLT

Stefan Böttger

Kirchhoff Institut für Physik

Ruprecht-Karls-Universität Heidelberg

RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Content

1. Introduction
2. The Problems with exploiting the Cluster
3. Our Solution based on Virtualisation
4. Practical Issues
5. Results / Outlook

Introduction

- Scientific computing requires powerful hardware
- Peak performance vs. efficiency

- Virtualisation as Enabler technology:

Getting “more“ out of the given resources

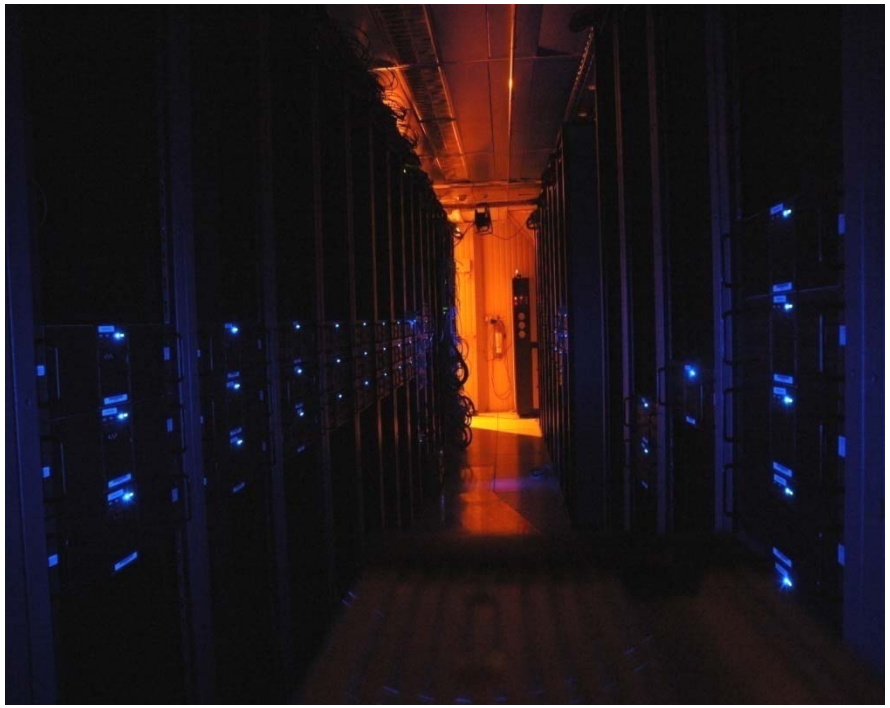
- Availability, redundancy, usage of CPU cycles, management

--- We won´t sell you something:
Is virtualisation more than a buzz word to us ? ---

RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

HLT Cluster

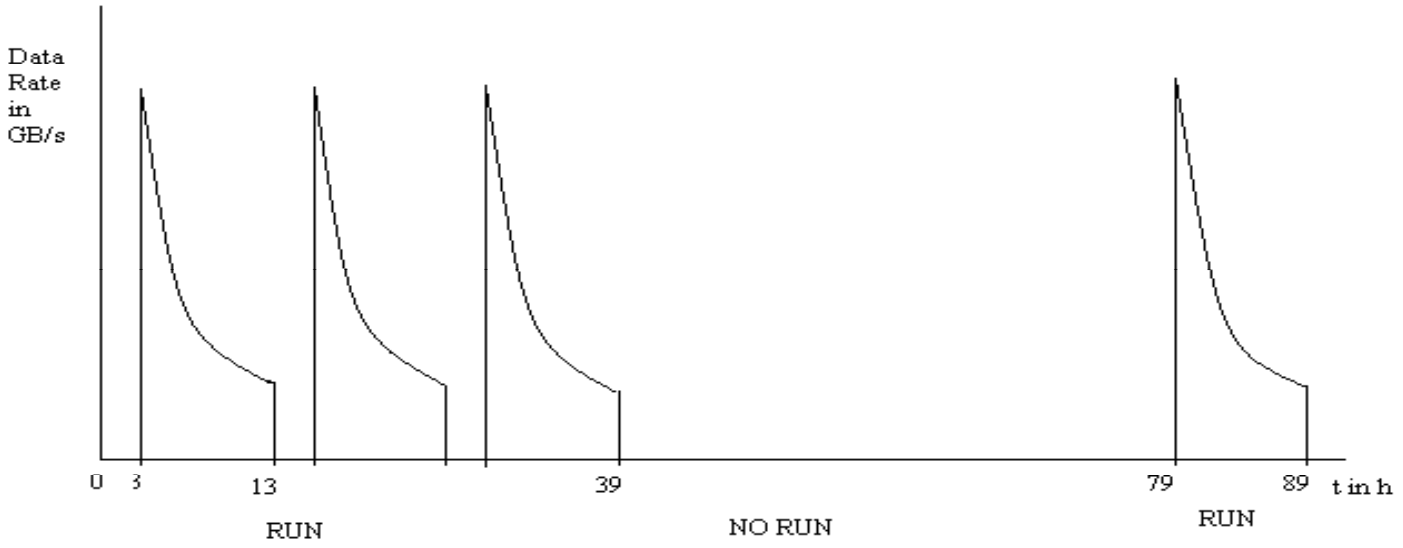
- Commodity hardware cluster
  - –dual/quad core opterons, 8GB RAM
  - –currently 100 nodes, up to 900 nodes planned till 2009
  - –Ethernet interconnects, Quadrics being installed
  - –Linux (Ubuntu6.06) OS
  - –CHARM, HRORC PCI-Cards



- Current usage
  - –on-line data processing
    (High Level Trigger@Alice)
  - –off-line data processing
    (ALiEN Grid)

Usage Scheme of HLT

- No constant data rate: inner-run gaps & inter-run gaps



Our Goal:

1. Dedicate all resources needed to on-line processing
2. Do off-line processing whenever possible

➢ Compute as many results as possible per time

Usage Options

The easy way: Use only the inter-run time gaps for off-line

- off-line requires different OS
- wasted CPU-cycles
- inter-run time needed for maintenance

We want: Usage of inner-run time gaps also

- different OS requirements
- fully dedicated resources
- clean separation needed
- gaps too short for processing a whole job
- checkpointing not supported by off-line

RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Our Approach

What we do:

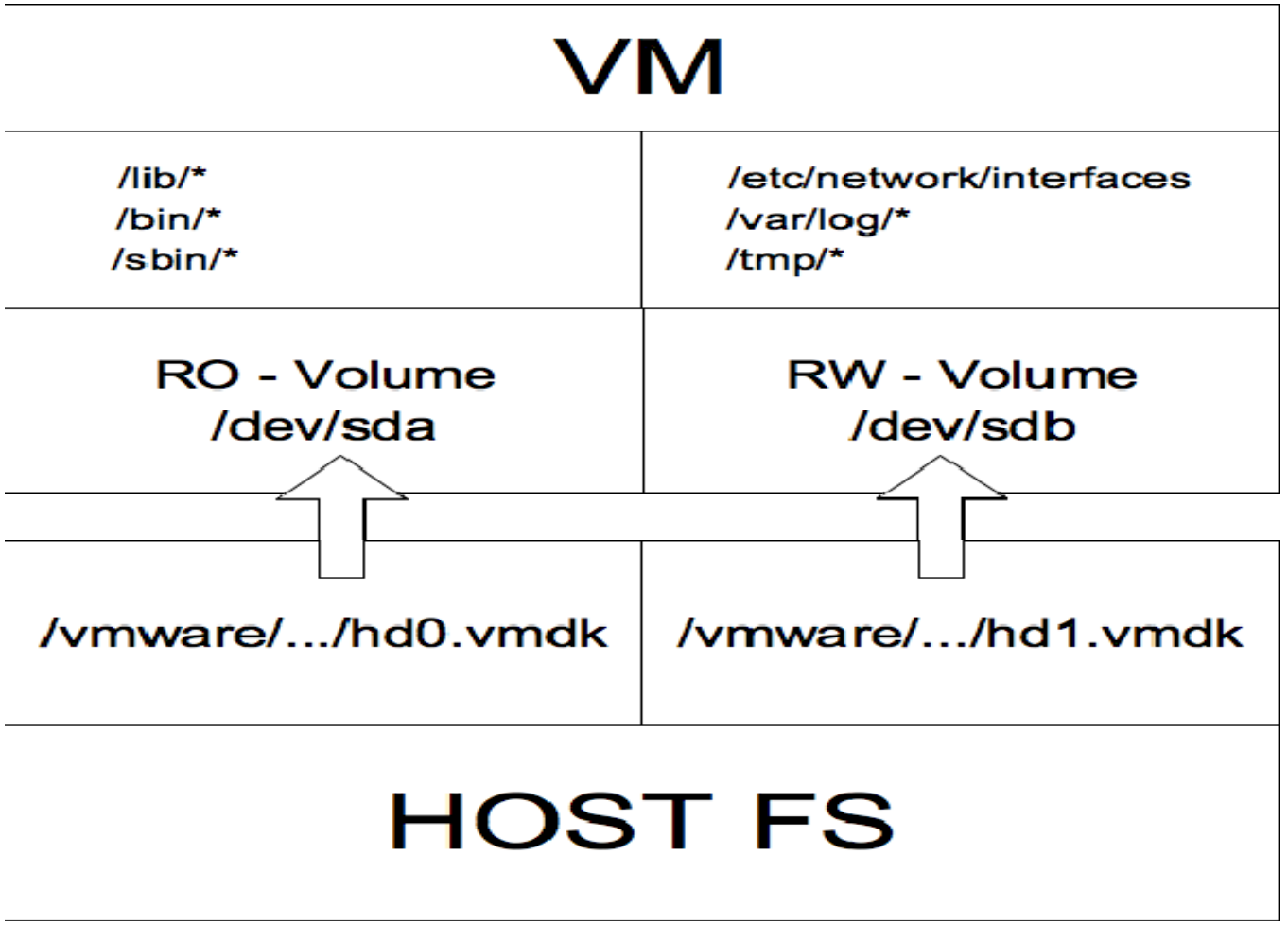
- nativ Ubuntu runs on-line computations (87 nodes)
- on-line jobs distributed by a proprietary application (PubSub)

- virtualised Scientific Linux (VMware Server) runs off-line computations (65 virtual nodes, 3GB RAM, 2 cores)
- off-line jobs distributed by SGE batch scheduler

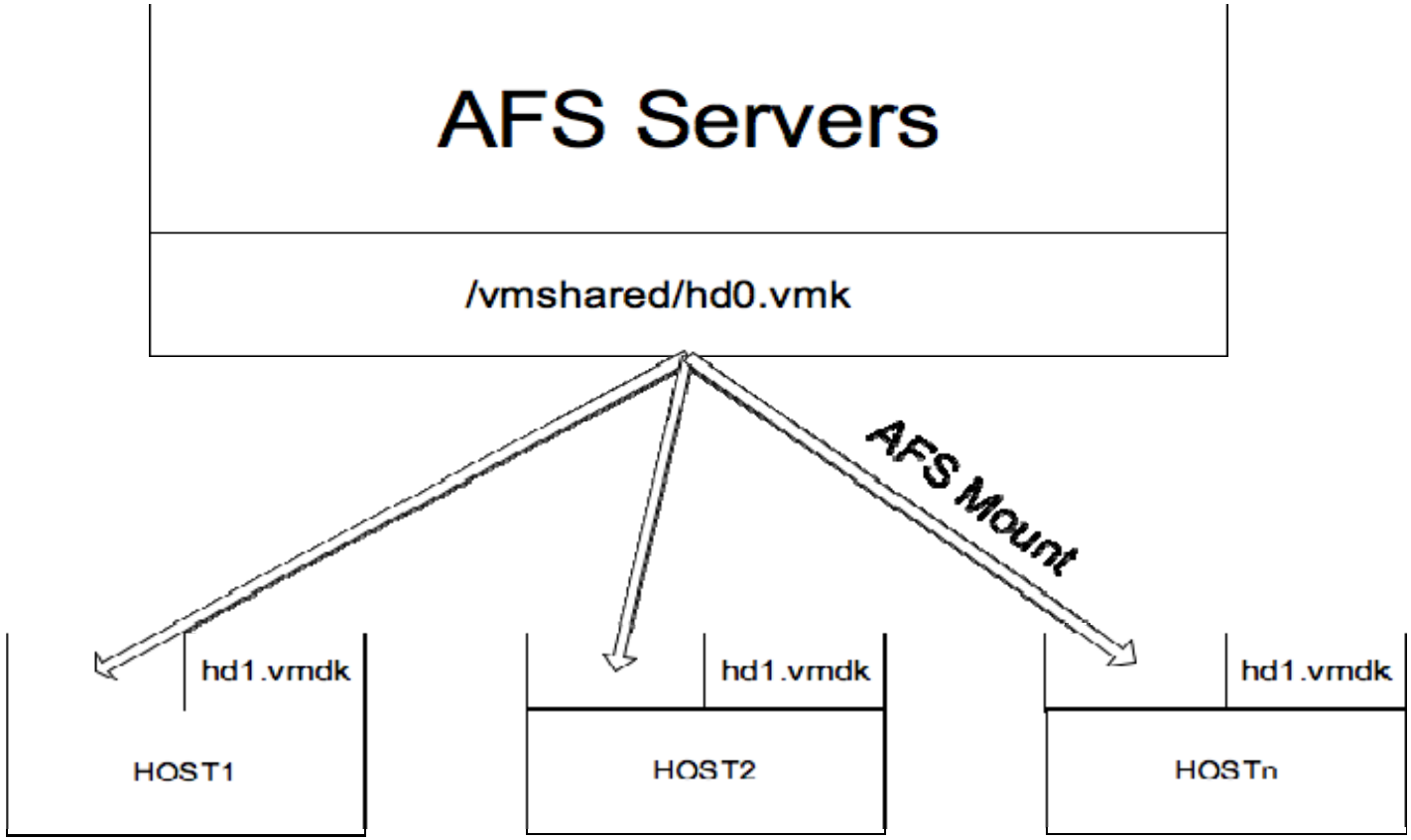
- VMs are suspended when data-rate is to rise
- migrated to a special server if necessary (off-line heartbeat)
- VMs are resumed when data-rate drops

Practical Issues I

- RO / RW Volume Separation



Practical Issues II



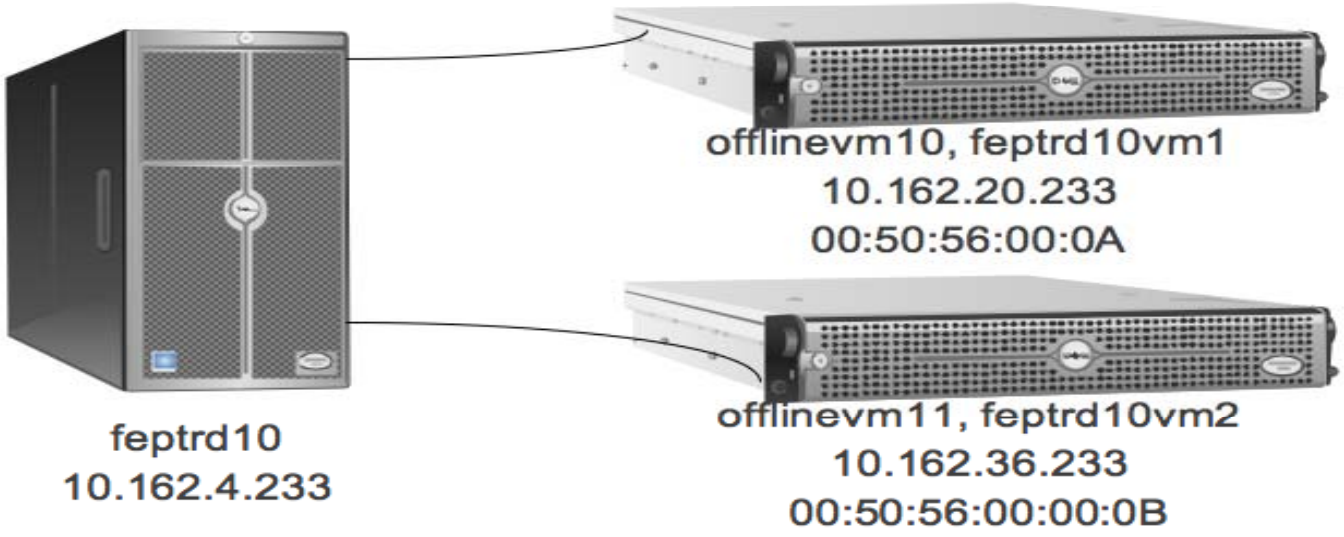
- single RO Volume for all vms
- specific RW Volumes for each vm
- centralised OS update possible

Practical Issues III

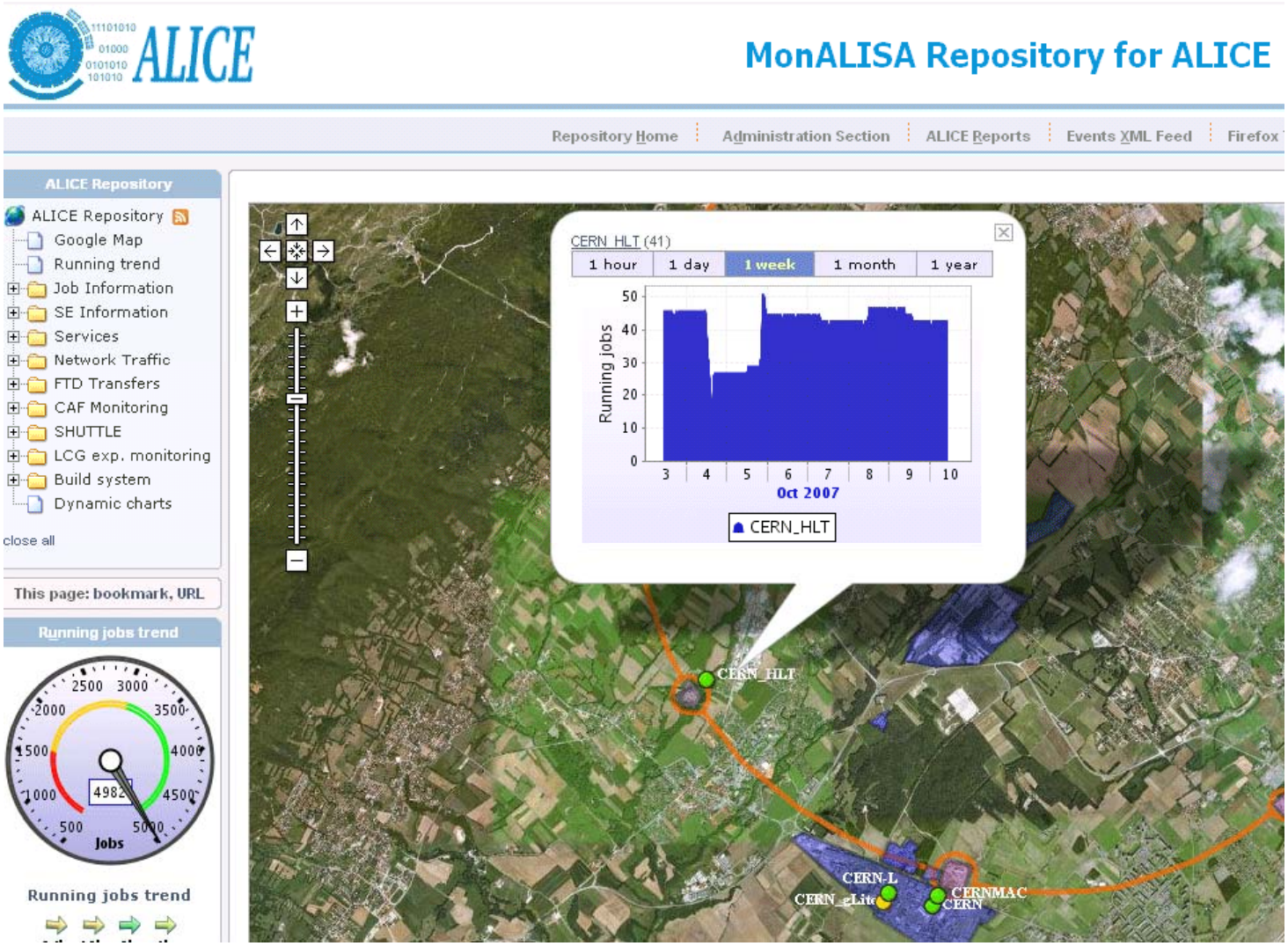
VM Addressing

- IP (first vm on host) = HostIP + 0.0.16.0
- IP (second vm on host) = HostIP + 0.0.32.0

- dns-name (vm) = offlinevm + toString(i)
- dns-alias (vm) = dns-name (host) + “vm“ + toString(j) (# of vm on host)
- mac address (vm) = 00:50:56:00:00:00 + i



Off-line Grid GUI



Results

- VM vs. Nativ computing efficiency for cpu/io-affine physics application (t needed for one found track)

	Exec t in sec	
	VM	Nativ
app1 with high I/O Load	2156	1959
app2 with high CPU Load	3585	3164

- eased change management through centralised vm hosting
- checkpointing works and saves computing time (maintenance & inner-run time gaps)
- suspend/migration of vms from one host to another takes too much time

Outlook

- parallel on-line /off-line processing
- XEN instead of VMWare Server (?)

- sensible migration schemes yet to be identified

- currently VM management done manually (scripted)
- next step: use system-management tool (SysMES) to automise VM management based on rules