Experience on running an integrated virtualized cloud/grid infrastructure in production mode at the INFN Tier-1

WLCG Collaboration Workshop – 7-9 July 2010, London

Peter Solagna, INFN Padova

Outline

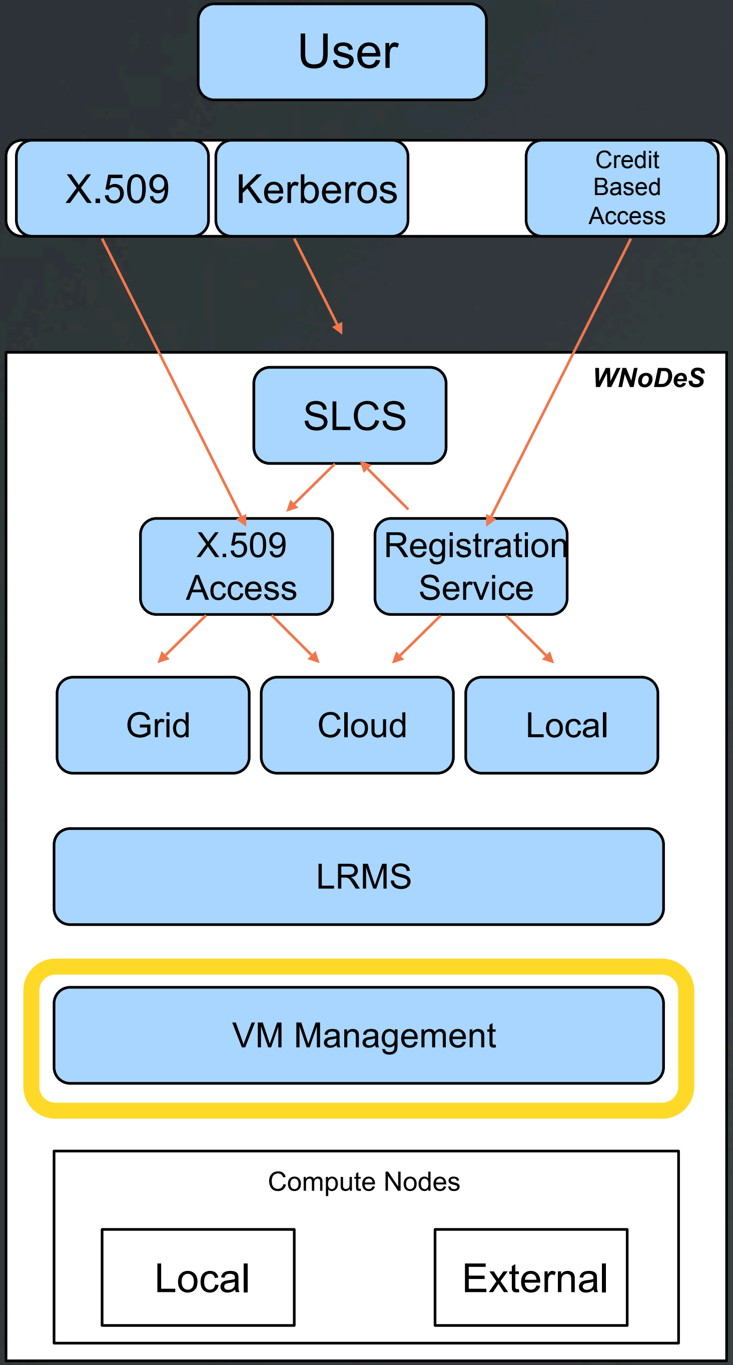
- WNoDeS architecture
- Issues raised by using a large virtualized cluster
  - Batch System Scalability
  - CPU performance
  - I/O (disk and network) performance and tuning
  - Interaction between VMs and distributed file systems (e.g. GPFS)
- Cloud access
  - Web Application overview
  - Developements

WNoDeS

- Worker Node on Demand Service
- WNoDeS is being developed by INFN, and it's in production at the INFN Tier-1 computing Centre. It is built around a tight integration with a LRMS (batch system).
  - On-demand virtual resource provisioning
  - Full integration with existing computing resource scheduling, policing, monitoring and accounting workflows
  - Support for users to select and access WNoDeS-based resources through

  Grid, Cloud interfaces, or also through direct job submissions.
  - *Everything as a Service*, where *Everything* may be hardware, software, data, platform, infrastructure

WnoDeS Architecture



User Access Methods

Authentication Gateway

Access Layer

Interface Layer

Policy Dispatching Layer

VM Layer

Resources

VM Layer

- VMM: KVM
- *Bait*: special virtual machine running on each computing resource
  - Publishes the max number of job slots for that resource
  - Creates (ask the VMM to create) an appropiate Virtual Worker Node
  - Dispatches the job to be executed on the newly created VWN
- A set of Python scripts handles communication between the LRMS, the KVM VMM, the bait, and the VWN:
  - to accept jobs from the LRMS on the bait;
  - to trigger creation, destruction, or suspension of a KVM VM via the KVM VMM;
  - and to keep job state information between the VWN and the bait.

Batch System Scalability

- Batch System Scalability performance is crucial! WnoDeS works as long as the LRMS can manage all the resources
- Additional VWNs registered in the INFN Tier-1 LSF cluster in April 2010 (currently, 4000 LSF clients are in the cluster)
  - Performance problems with LSF
- Tuning of Master LSF operating system
  - Max. Socket number, open files...
- Tuning of LSF scheduler parameters (with direct Platform collaboration)
  - Job Dispatch Interval : 60s ⟶ 30s
  - Job Checking Interval : 30s ⟶ 15s
  - Job Accepting Interval: 60s ⟶ 30s

Batch System Scalability

- lsf.conf parameters:
  - Enabled: LSB_MAX_PROBE_SBD = 64
  - Updated: LSB_MAX_JOB_DISPATCH_PER_SESSION = 1024
- lsf.params
  - Updated: MAX_SBD_CONNS = 4200
  - Defined: CONDENSE_PENDING_REASONS = Y
- Ongoing talks with Platform to grow further
  - 400 8-cores machines are planned to be added to WnoDeS
  - 20k nodes clusters will be soon a real use case for batch systems
    - Multi-cluster systems

CPU performance

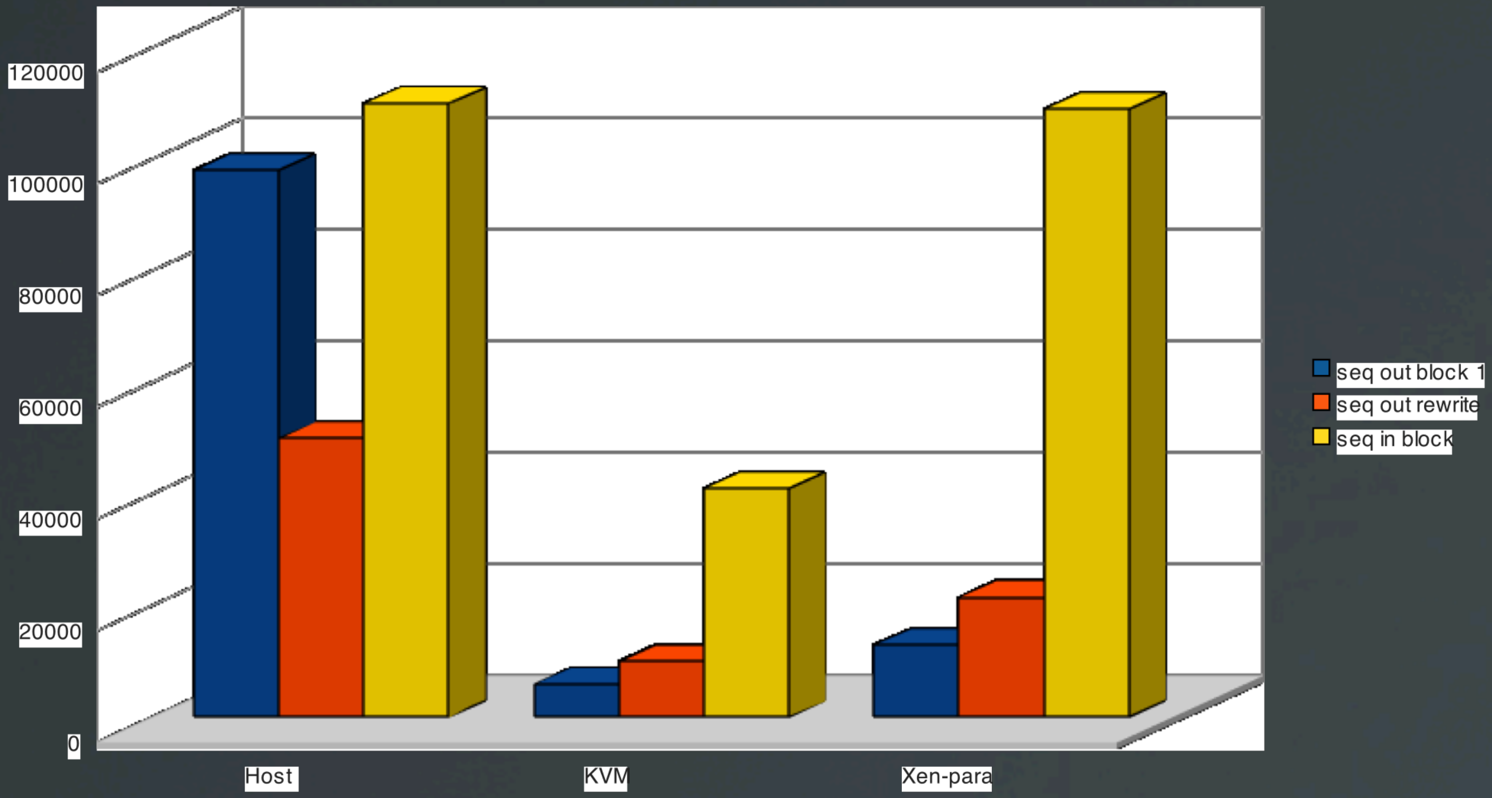
- HEP_SPEC KVM performance test
  - Standard configuration: RH5.4+KVM
    - Integer operations
  - Intel 4 cores Clovertown CPUs
    - Virtual vs Physical Machine: 4~5% performance loss
  - Intel 4 cores Nehalem E5520 CPUs (2009)
    - Virtual vs Physical Machine: >15% performance loss
- KVM configuration fix (RH developers support)
  - Disable EPT (for integer operation, shadow page tables are faster):
    - rmmod kvm-intel
    - modprobe kvm-intel enable_ept=0
  - Performance loss is now acceptable: ~5%
  - Ongoing test with Transparent Hugepage Tables hugetlbfs not yet included in official RH distribution

CPU model identification

- By default guest machine has no clue about the hardware configuration of the master
  - For accounting reasons the CPU model is a necessary information
  - Alice wants the CPU model information as a VO requirement
- Inject the CPU information using libguestfs:
  - Hypervisor downloads the needed VM image
  - Hv writes into a file its own CPU informations, and injects it using libguestfs into the VM image file system.
  - As the VM starts the software running on it can find the CPU informations in the file, put in a known location

Disk I/O Performance

- I/O (disk or network) can still be a weak point in virtualization.



Network I/O

- Standard KVM configuratin has poor network performances:
  - dstat dump on an average loaded VM (10-20 MB/s):
    - IO Wait + Software IRQ + Hardware IRQ > 90%

```
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw

 5   1   0  88   1   5|   0     0 |6102k  172k|   0     0 | 478   235
14   1   0  52   2  31|   0     0 |  16M  445k|   0     0 |1117   379
12   1   0  58   5  24|   0     0 |  14M  415k|   0     0 |1062   346
 7   3   0  55   7  28|   0     0 |  15M  420k|   0     0 |2251   308
 9   1   0  70   7  13|   0     0 |  10M  291k|   0     0 |1098   245
```

  - Flood ping vs a loaded machine like the above:

```
--- 131.154.203.223 ping statistics ---
1006 packets transmitted, 929 received, 7% packet loss, time 11374ms
rtt min/avg/max/mdev = 0.210/325.721/1145.855/308.654 ms, pipe 98, ipg/ewma 11.317/122.548 ms
```

    - 7% packets lost, caused by highly loaded CPU
    - Packets lost can be a huge problems for distributed file systems
- Performances improve dramatically using virtio_net for network
  - Tested network I/O peak: 900 Mbit/

Network I/O tuning

- Switch a sl53_x86_64 guest machine from e1000 driver to virtio:
  - edit /etc/modprobe.conf and change alias eth0 e1000 in alias eth0 virtio_net
  - disable kudzu on boot: chkconfig kudzu off
  - After rebooting if your machine is configured to get address via dhcp, it will load the new module and get the ip automatically.
- Switch a sl4x_x86_64 guest machine from e1000 driver to virtio:
  - edit /etc/modprobe.conf and remove the line alias eth0 e1000
  - disable kudzu on boot: chkconfig kudzu off
  - remove the line starting with HWADDR under /etc/sysconfig/network-scripts/ifcg-eth0
  - After rebooting if your machine is configured to get address via dhcp, it will load the new module and get the ip automatically.
- The virtio config. is automatically managed by WNoDeS depending on the VM it is starting.

Distributed File Systems and VMs

- GPFS is the file system used at INFN Tier-1
- General issues of distributed file systems on large clusters
  - Planned and ”well done” nodes startup or shutdown
  - For performance reasons, WNoDeS mounts on a VM only the GPFS file systems actually needed by that VM
    - On 6 July ~3800 Alice jobs were running in the Tier-1 cluster, mounting the same file system
  - Due to the big number of nodes the cluster must be fine tuned.
  - Network performance is crucial! No packets loss is allowed
- Currently the IBM official maximum tested GPFS cluster size limits are:

GPFS for Linux (Multiplatform and x86 Architecture)	3794
GPFS for AIX	1530
GPFS for Windows (within an existing GPFS cluster)	64
GPFS for Linux (Multiplatform and x86 Architecture) and GPFS for AIX	3906

- To deploy bigger clusters is highly suggested to contact IBM for direct support in configuration and performance tuning of GPFS

Time Sync

- The virtual machines time syncronization is mandatory for a number of applications, like GPFS and networked applications in general
- From Red Hat docs:
  - Using the para-virtualized clock with Red Hat Enterprise Linux guest, for certain Red Hat Enterprise Linux guests, additional kernel parameters are required:

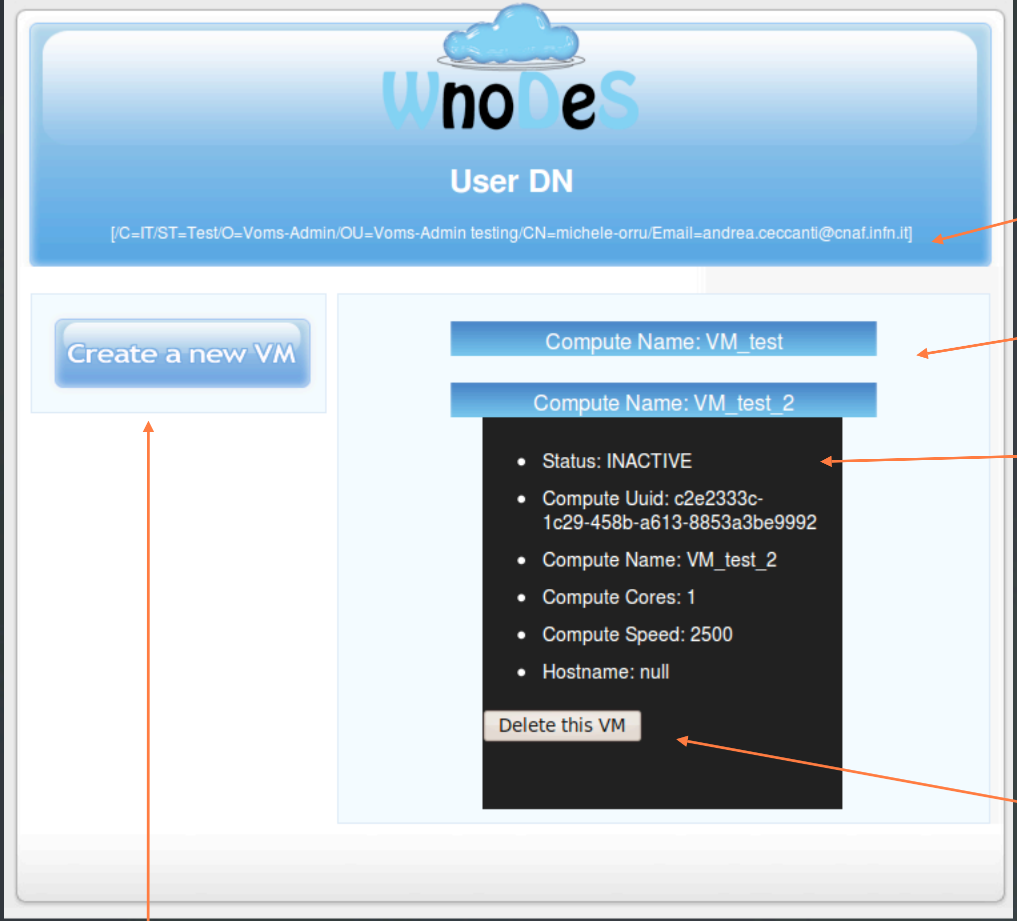
Red Hat Enterprise Linux	Additional guest kernel parameters
5.4 AMD64/Intel 64 with the para-virtualized clock	Additional parameters are not required
5.4 AMD64/Intel 64 without the para-virtualized clock	divider=10 notsc lpj=n
4.8 AMD64/Intel 64	notsc divider=10

The Cloud layer

- It is used to expose a second service that coexist with the Tier-1 grid infrastructure
  - The same pool of resources used for grid or local jobs can be used for Cloud
- RESTful web service
  - Jax-RS, Spring, Hybernate, Tomcat
- X.509 certificates for authentication and autorization
- Accessible via HTTP client like curl or a simple alpha stage Web Application (currently not public)

Web Application

Main View



User DN

[/C=IT/ST=Test/O=Voms-Admin/OU=Voms-Admin testing/CN=michele-orru/Email=andrea.ceccanti@cnaf.infn.it]

Compute Name: VM_test

Compute Name: VM_test_2

- Status: INACTIVE
- Compute Uuid: c2e2333c-1c29-458b-a613-8853a3be9992
- Compute Name: VM_test_2
- Compute Cores: 1
- Compute Speed: 2500
- Hostname: null

User authenticated by browser certificate

Owned Virtual Machines collapsable list

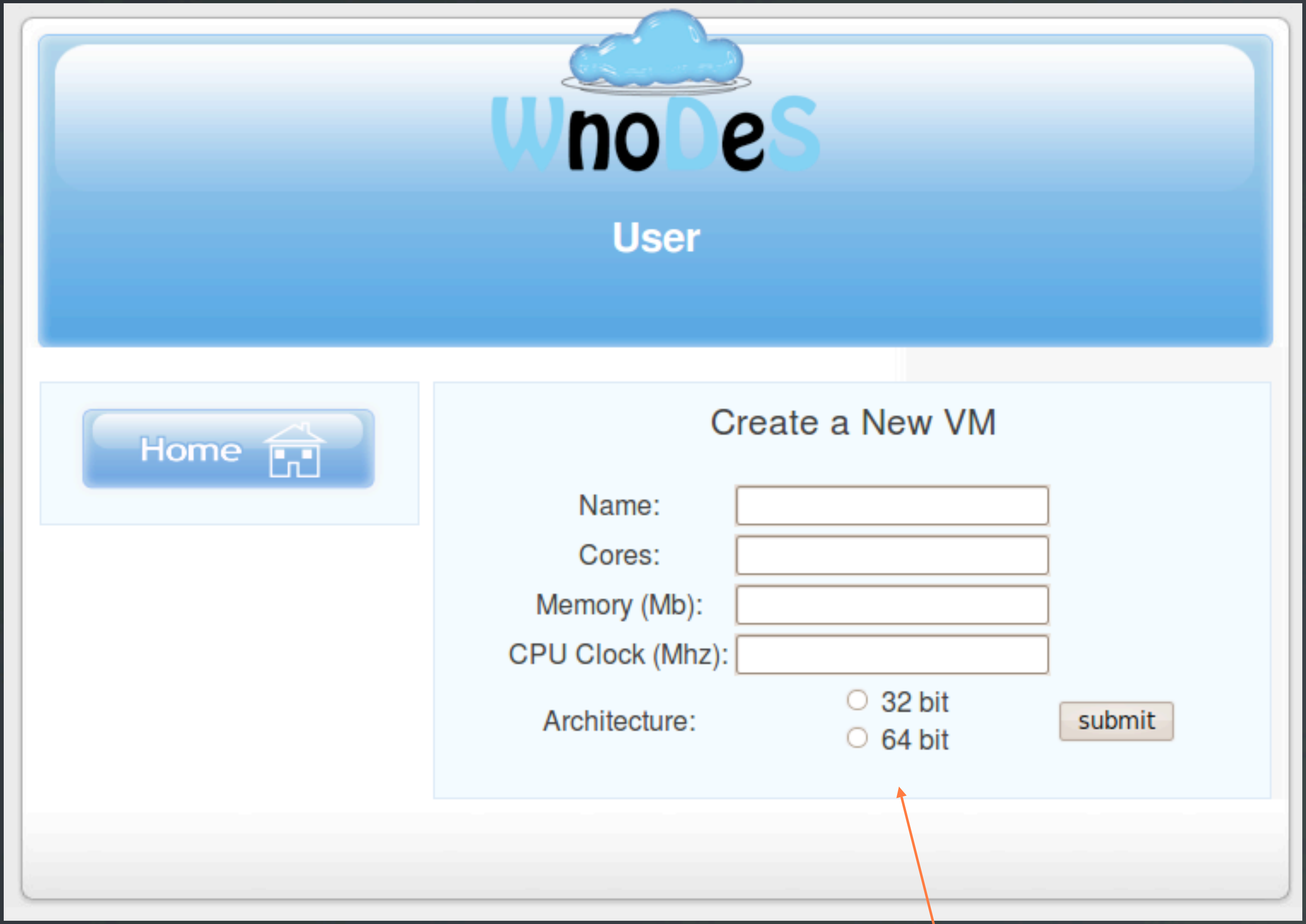
Virtual machine information
Dynamic parameters like
Status and Hostname are
Got with Ajax calls (not refresh
needed)

Delete the virtual machine

Link to the VM creation form

Web Application

Virtual Machine Creation Form



Create a New VM

Name:

Cores:

Memory (Mb):

CPU Clock (Mhz):

Architecture: ○ 32 bit ○ 64 bit

Soon the options will be:
SL4 32bit
SL5 64bit

Web Application Developements

- Ongoing
  - Add the SL4/32 and SL5/64 VM images
  - Automatic generation of ssh keys to be injected into the created VM, in order to let the user connect as root into his own virtual machine
    - The user will be able to use his self-generated public ssh key
- Close future
  - User authentication and authorization using VOMS and ARGUS services

Contacts

- WNoDeS site:
  - http://web.infn.it/wnodes/
- WnoDeS mail list:
  - wnodes-request@lists.infn.it
- Me:
  - peter.solagna@pd.infn.it