The INFN experience with virtualized computing services and perspectives for integration of Grid and Cloud services

Davide Salomoni

INFN-CNAF

About this talk

- It's a summary of INFN activities on Grid/Cloud + Virtualization.
  - Virtualization WG
  - Grid/Cloud Integration WG
- Thanks to all those who contributed to this
  - A.Chierici, R.Veraldi, A.Italiano, A.Ghiselli, V.Venturi, M.Orrù, M.Cecchi, D.Rebatto, M.Sgaravatto, L.Zangrando, and others

Outline

- Problem context
- Components of a grid-aware solution
  - VM's
  - Local layer: VM managers and dynamic instantiation
  - Extension to CE's and WMS'
- Generalizing the architecture toward a grid and cloud integration

Problem context

- Running jobs in dynamic environments, i.e. on computing resources “created” at job execution time
  - Interacting with resources through standard grid interfaces. Ideally, the mechanism should be as transparent as possible.
  - Run experiment software (not touched *per se* by this talk, as it normally lives in software areas) on various O/S versions – across different VOs or even within a given VO
  - Provide strictly homogeneous environments to users requiring them: e.g., ETICS people need to run their building and testing code on systems where they know exactly what is installed.
  - Efficiency optimization: proper resource selection *may* bring about billing/accounting advantages.
- But then, could we also re-use our (HEP) resources and provide access to them through simpler interfaces, à la EC2?
  - Some (typically small) experiments / customers might not want or need to use grid interfaces
  - Can we 1) offer differentiated access methods (grid *and* cloud) and 2) optimize resource usage?

To share or not to share?

Sharing Is the Root of All Contention

Sharing requires waiting and overhead, and is a natural enemy of scalability

By Herb Sutter, Dr. Dobb's Journal
feb 13, 2009
URL:http://www.ddj.com/hpc-high-performance-computing/214100002

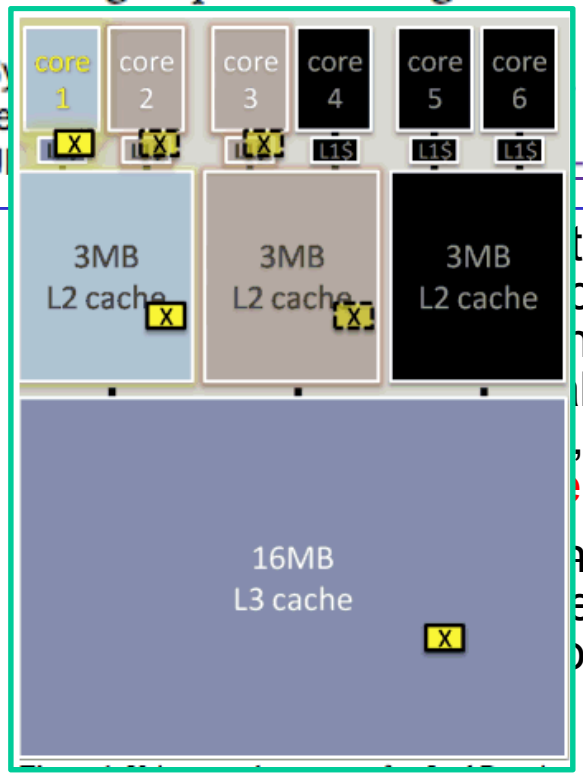
- This is obviously valid at the resource configuration level ("can't we simply statically dedicate resources?") and at the software layer (ever tried e.g. to run multiple Python threads in a system with more than one core? See also Vincenzo's talk). But it's also valid at the hardware layer (e.g. many-core systems sharing memory, latencies induced by L1/L2/Ln cache contention) →All sharing is bad – even of "unshared" objects.
- On the other hand, ~~wasting~~ leaving resources idle is something not always desirable, given also increasing needs by many key customers and, at the same time, decreasing (or, at best, constant) money.

To share or not to share?

Contention writing to the **same** shared object

Contention

Sharing requires waiting and overhead, and is a natural enemy of scalability



Contention writing to ***nearby*** shared objects a.k.a. *false sharing*

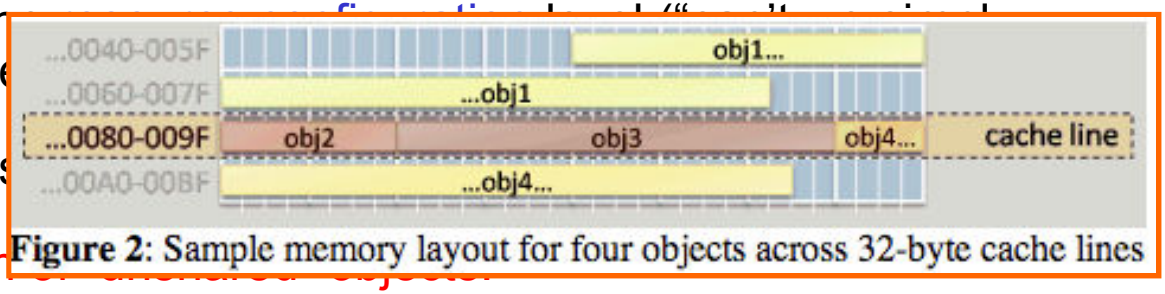


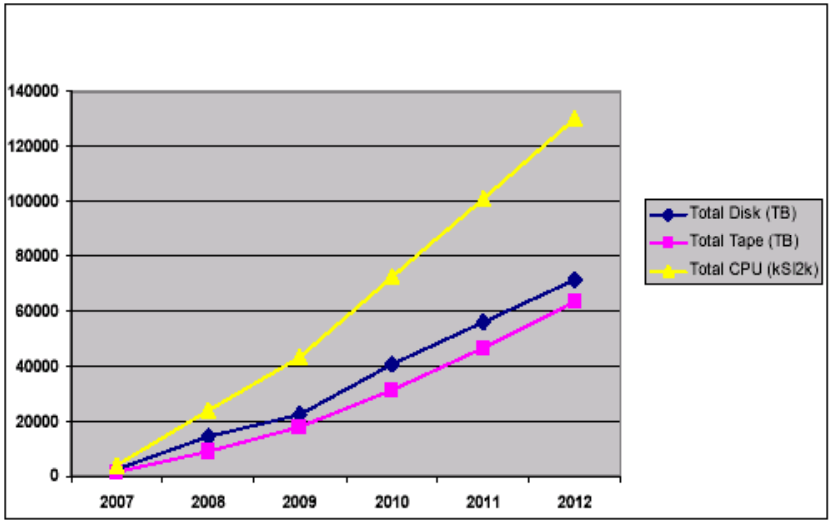
Figure 2: Sample memory layout for four objects across 32-byte cache lines

- ~~ng~~ leaving resources idle is something not always easing needs by many key customers and, on the or, at best, constant) money.

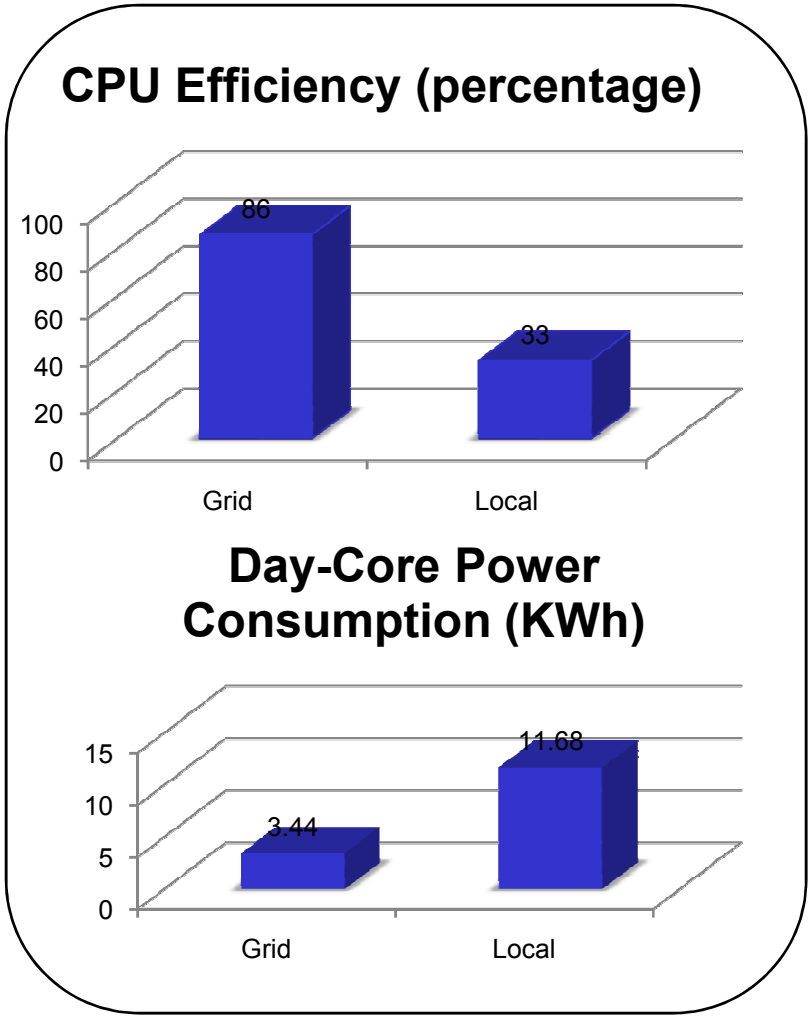
Sharing or not?

- Customer requirements

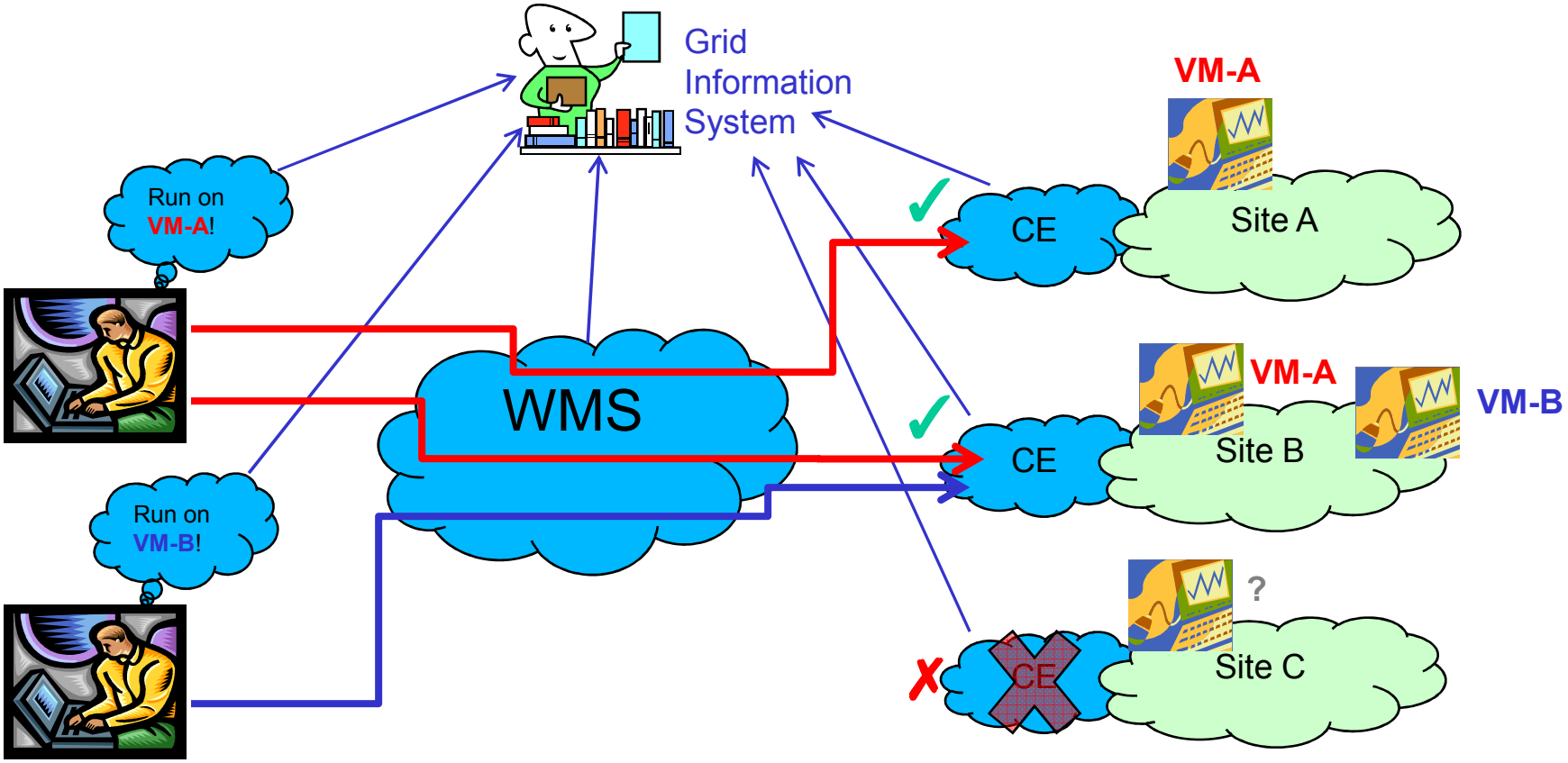
ATLAS forecasts(2006)



- Typical efficiency & power consumption patterns for shared vs. non-shared farms (data by A.Ciampa, INFN-Pisa – time period 1 year, 05/08-05/09)



Grid-aware dynamic selection of resources



Outline

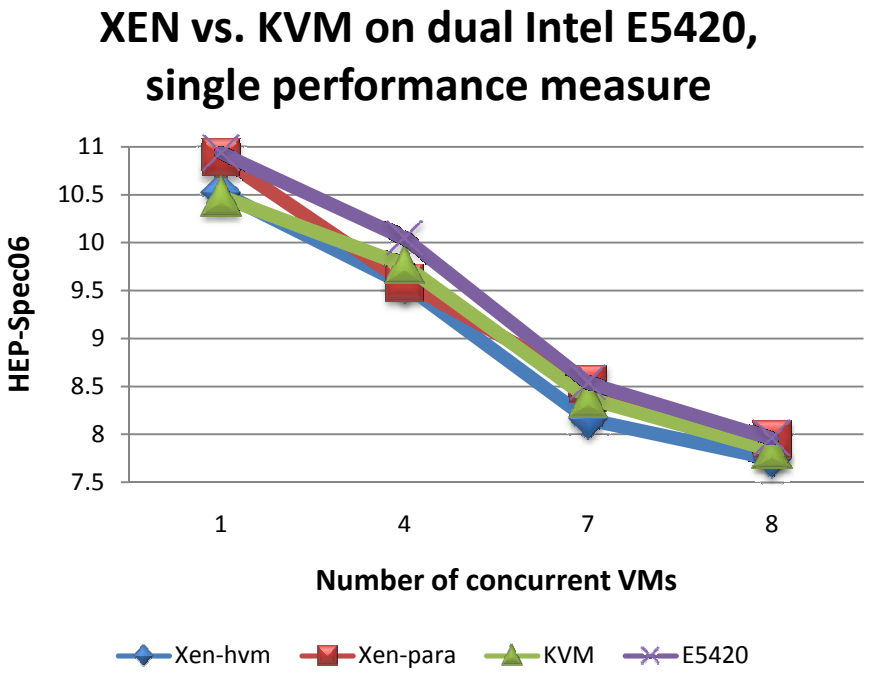
- Problem context
- Components of a grid-aware solution
  - VM's
  - Local layer: VM managers and dynamic instantiation
  - Extension to CE's and WMS'
- Generalizing the architecture toward a grid and cloud integration

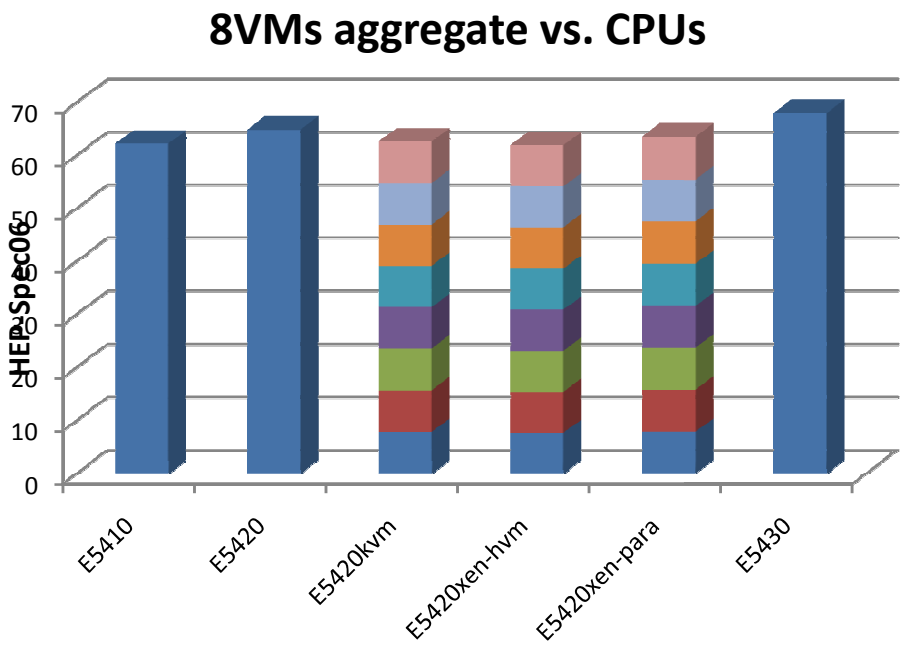
Basic building block: VM's

- An important activity within INFN has been to verify the merits of VM's and of competing solutions through quantitative tests
- XEN (para-virtualized and HVM) vs. KVM, using a non-virtualized machine as the baseline
  - 1 blade, dual E5420, 16GB ram, 10k sas disk via LSI logic raid controller (raid0)
  - Xen-para VM specs: 1 vcpu, 2 GB ram, disk on a file
  - Xen-hvm VM specs: 1 vcpu, 2GB ram, disk on a file, "netfront" network driver
  - KVM VM specs: 1 vcpu, 2GB ram, disk on a file, e1000 network driver emulation (non-virtio tests)
  - Host OS: SL 5.2 x86_64, kernel 2.6.18-92.1.22.el5
  - VM OS: SLC 4.5 i386, kernel 2.6.9-67.0.15.EL.cern

Benchmarking VM's

- Plots by A.Chierici/R.Veraldi, INFN-CNAF
  - CPU tests using HEP-SPEC06 and PI calculation
  - Network and disk access tests not shown here – see the "XEN vs. KVM" talk by R.Veraldi at HEPiX '09





Summary of results

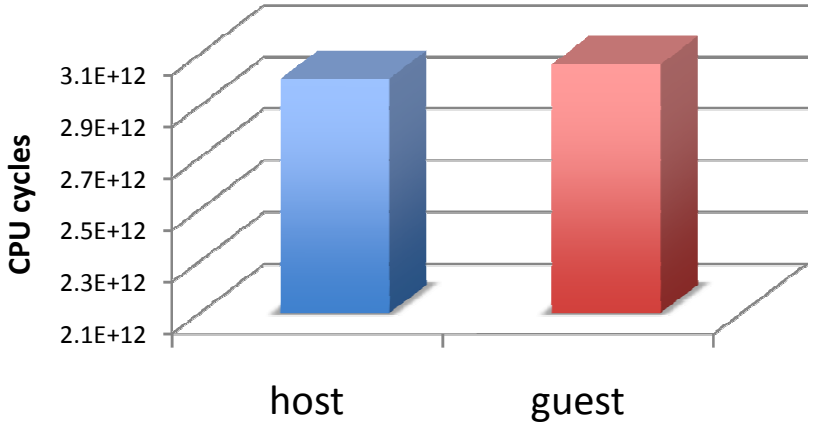
- HEP-SPEC

Virtualization Technology	% loss from non emulated CPU (E5420, 8vm)
E5420kvm	3,42
E5420xen-hvm	4,55
E5420xen-para	2,02
E5410 vs. E5420	4,07

Summary of results

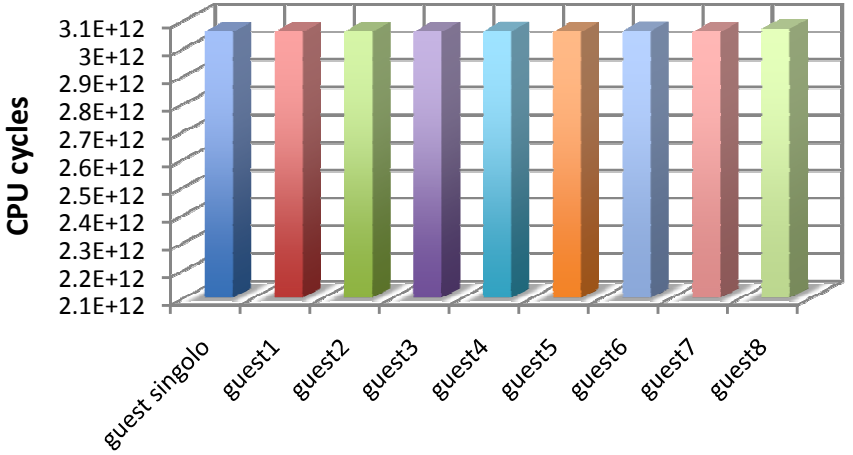
- PI tests
  - Computes PI with n arbitrary digits, returns the number of CPU cycles through an external function using the RDTSC instruction

KVM host vs. guest



Difference < 2%

8 concurrent guest

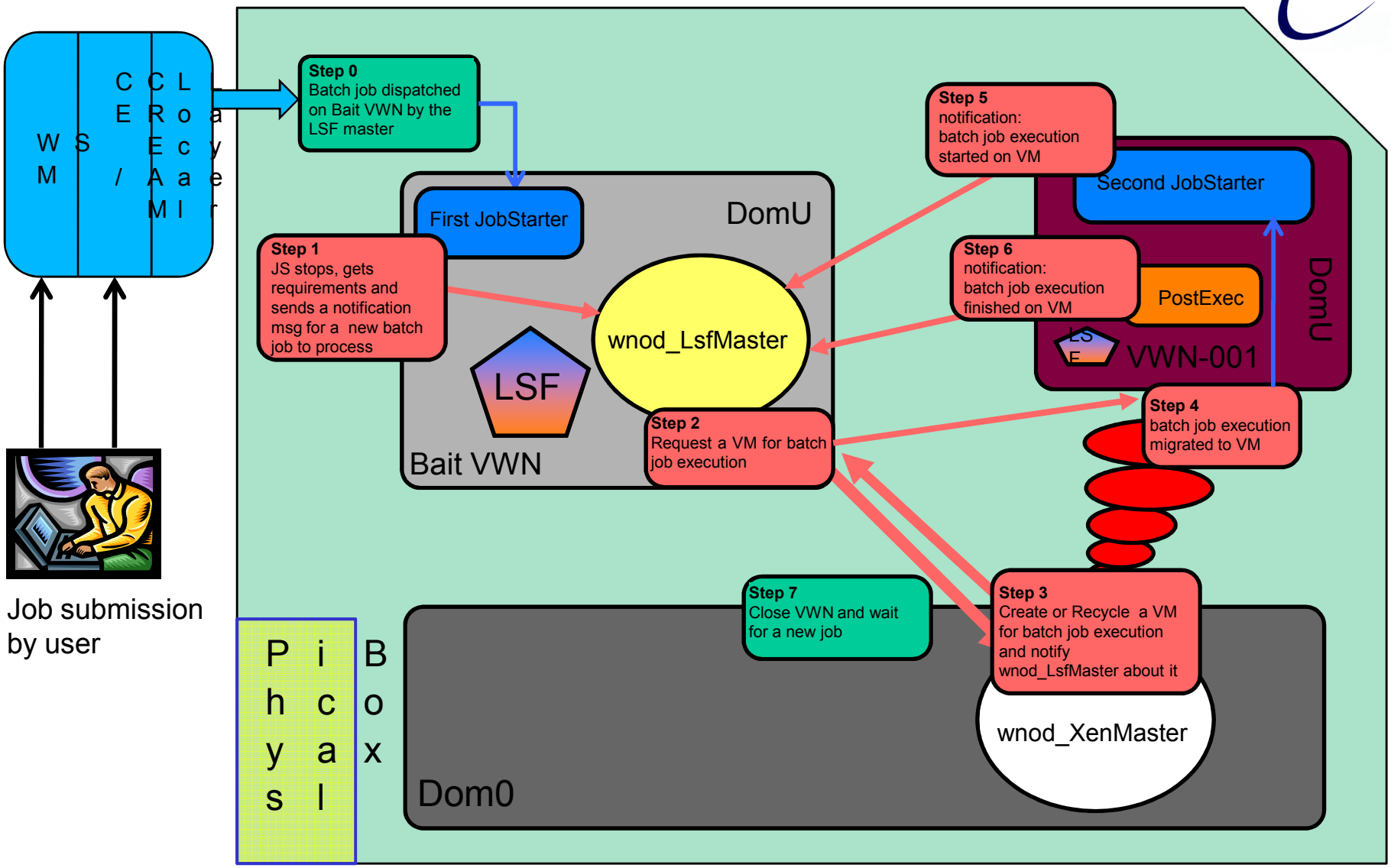


8 cores, balanced performance, each guest uses 1 core at 100%

Outline

- Problem context
- Components of a grid-aware solution
  - VM's
  - Local layer: VM managers and dynamic instantiation
  - Extension to CE's and WMS'
- Generalizing the architecture toward a grid and cloud integration

Local Layer Example: CNAF's WNOD (Worker Nodes On Demand)



Some WNOD details

- VM installation and configuration managed by Quattor
- Underlying batch system is Platform LSF
- Can be extended to support requests for multiple cores
- Successfully migrated from XEN to KVM
  - Several simplifications were possible, e.g. introduction of snapshots, less overhead in the WNOD code
- Currently extensively testing interaction with GPFS
  - No major problems encountered (short of clock synchronization, now achieved through a cron hack using ntpdate – paravirtualized clock to be tested, see talk by A.Arcangeli)
- VM startup time currently at about 120 sec, mostly driven by GPFS
- About 20k jobs executed, using 3 different VM types, max of 14 VM on an 8-core physical hardware (for testing purposes)

Outline

- Problem context
- Components of a grid-aware solution
  - VM's
  - Local layer: VM managers and dynamic instantiation
  - Extension to CE's and WMS'
- Generalizing the architecture toward a grid and cloud integration

Elements toward a solution

- User requirements must be [selected by the user and] carried forward along the WMS → CE → LRMS chain and ultimately processed by the dynamic VM manager.
  - Use the so-called *Forward Requirements*
  - This works fine with the latest WMS & CREAM CE releases
- In particular, we chose to use an existing Glue Schema 1.x attribute:
  - SRTE : Software RunTime Environment

SRTE Pros, Cons, and Use

- ✓ It's an existing Glue 1.x attribute: no need to introduce new things, no compatibility issues
  - It may be used to store VM data (and, hence, select them)
- ✗ The semantic of SRTE entries is not univocally determined
  - For instance, SRTE might be used (is currently being used) to identify which resource centers to select (e.g., "tier1" or "tier2"). This is only useful at the WMS match-making level, i.e. this is not really a forward requirement, so it is irrelevant here.
  - How would an engine interpret the elements published into the SRTE attribute? → need to disentangle ambiguity.
-  Use it through a convention:
  - Dynamic VM are identified via SRTE using the syntax VM_TYPE:xxx
  - Yes, it is somewhat fragile, and it works today

SRTE in practice

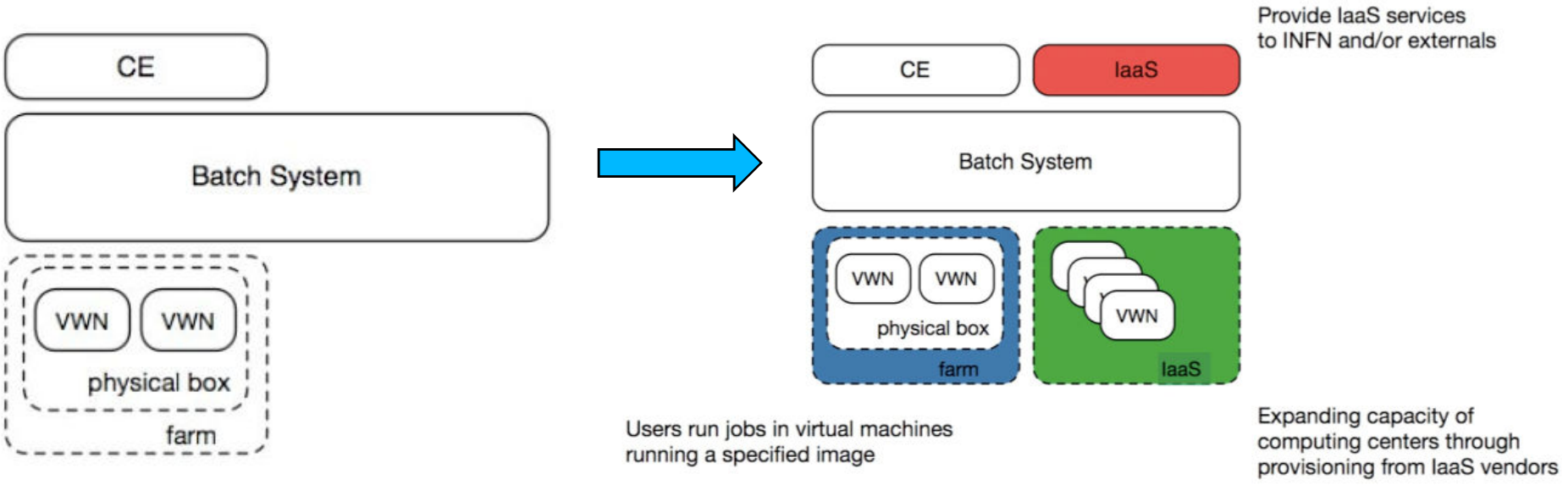
Attribute Description	Value
objectClass (12 values)	
GlueSchemaVersionMajor	**1**
GlueSchemaVersionMinor	**3**
GlueSubClusterUniqueID	**ce02-lcg.cr.cnaf.infn.it**
GlueChunkKey	GlueClusterUniqueID=ce02-lcg.cr.c
GlueHostApplicationSoftwareRunTimeEnvironment (16 values)	
GlueHostApplicationSoftwareRunTimeEnvironment	GLITE-3_0_0
GlueHostApplicationSoftwareRunTimeEnvironment	INFN-T1
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_1_0
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_1_1
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_2_0
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_3_0
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_3_1
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_4_0
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_5_0
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_6_0
GlueHostApplicationSoftwareRunTimeEnvironment	LCG-2_7_0
GlueHostApplicationSoftwareRunTimeEnvironment	R-GMA
GlueHostApplicationSoftwareRunTimeEnvironment	SF00MeanPerCPU_810
GlueHostApplicationSoftwareRunTimeEnvironment	SI00MeanPerCPU_1610
GlueHostApplicationSoftwareRunTimeEnvironment	VM_TYPE:dteam_img_slc44
GlueHostArchitecturePlatformType	i686
GlueHostArchitectureSMPSize	2
GlueHostBenchmarkSF00	810
GlueHostBenchmarkSI00	1610
GlueHostMainMemoryRAMSize	4096
GlueHostMainMemoryVirtualSize	8000
GlueHostNetworkAdapterInboundIP	FALSE
GlueHostNetworkAdapterOutboundIP	TRUE
GlueHostOperatingSystemName	Scientific Linux CERN
GlueHostOperatingSystemRelease	3.0.6
GlueHostOperatingSystemVersion	SL
GlueHostProcessorClockSpeed	2200
GlueHostProcessorModel	Opteron
GlueHostProcessorVendor	AMD
GlueInformationServiceURL	ldap://ce02-lcg.cr.cnaf.infn.it:2170
GlueSubClusterLogicalCPUs	0
GlueSubClusterName	ce02-lcg.cr.cnaf.infn.it
GlueSubClusterPhysicalCPUs	0
GlueSubClusterTmpDir	/tmp
GlueSubClusterWNTmpDir	/tmp

- Selection:
  - Technical note: SRTE is a sub-cluster attribute – it would maybe make more sense to find a cluster-level attribute.
  - Add specific requirement to the JDL:
    ```
    requirements  =
    Member("VM_TYPE:dteam_img_slc44",other.GlueHostA
    pplicationSoftwareRunTimeEnvironment)
    ```
- Future:
  - Use of Glue 2.x
  - Formalization of VM identification
  - Definition and standardization of key VM parameters, possibly including consumable resources (like RAM, floating licenses)
  - Standardization of the interface toward the LRMS – LSF, Torque, GridEngine, etc.

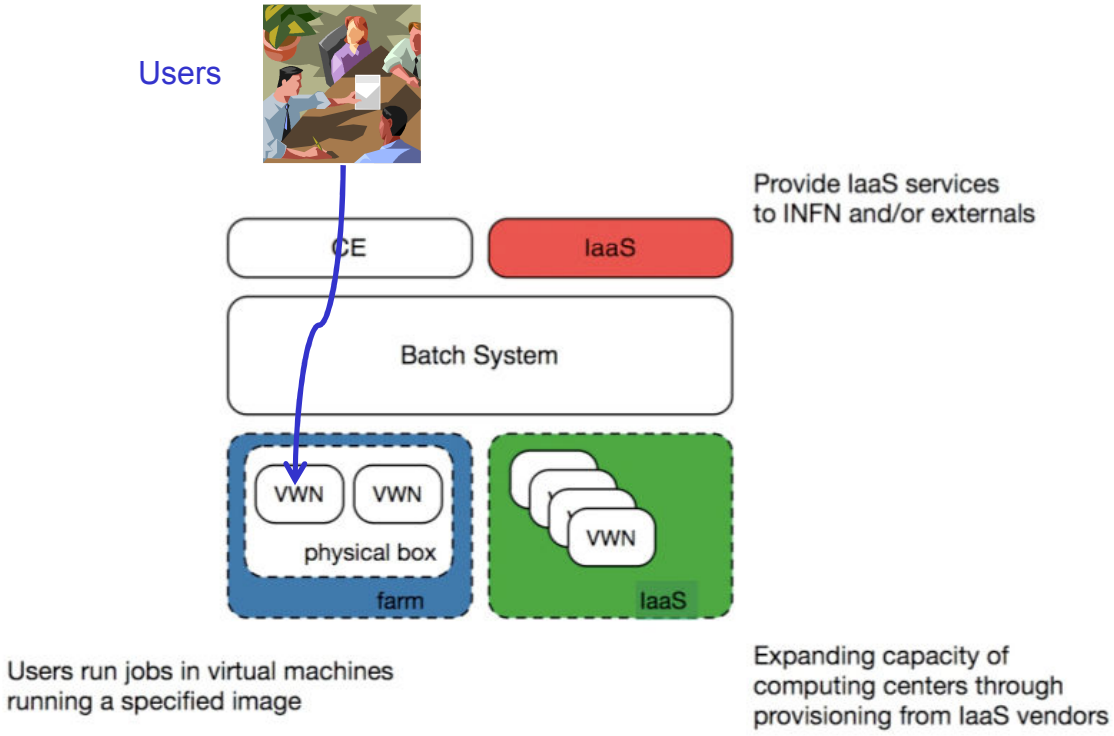
Outline

- Problem context
- Components of a grid-aware solution
  - VM's
  - Local layer: VM managers and dynamic instantiation
  - Extension to CE's and WMS'
- Generalizing the architecture toward a grid and cloud integration

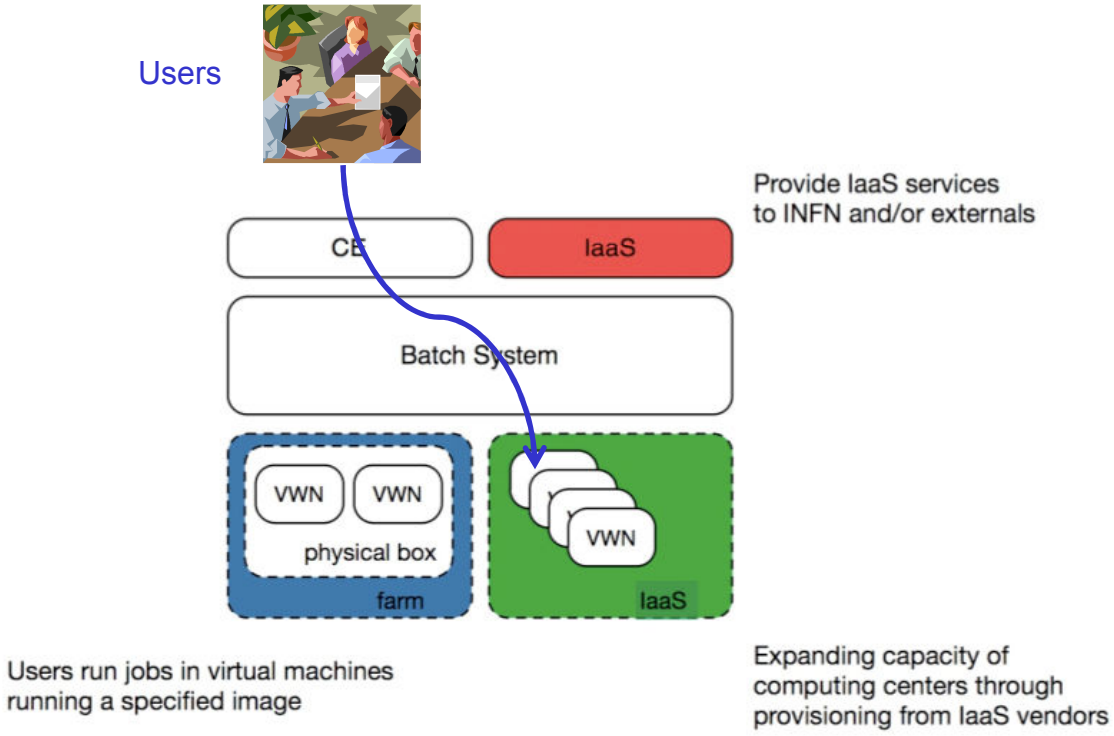
Integration in a slide



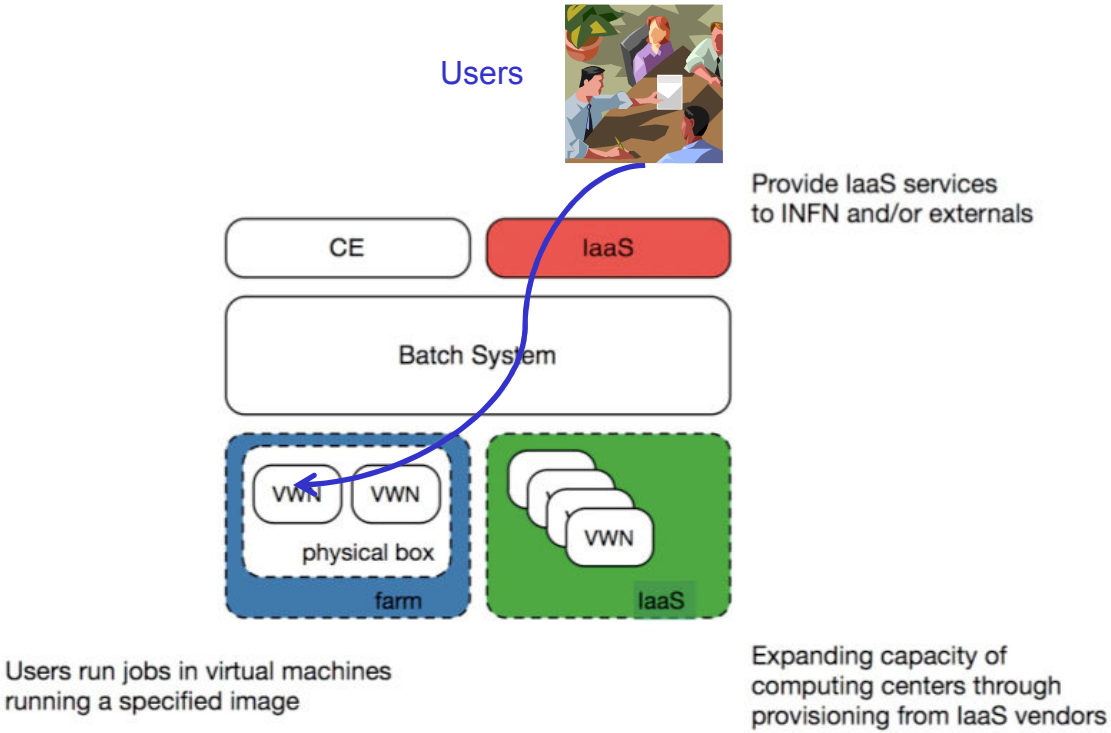
Path 1: Grid-aware Virtual Provisioning



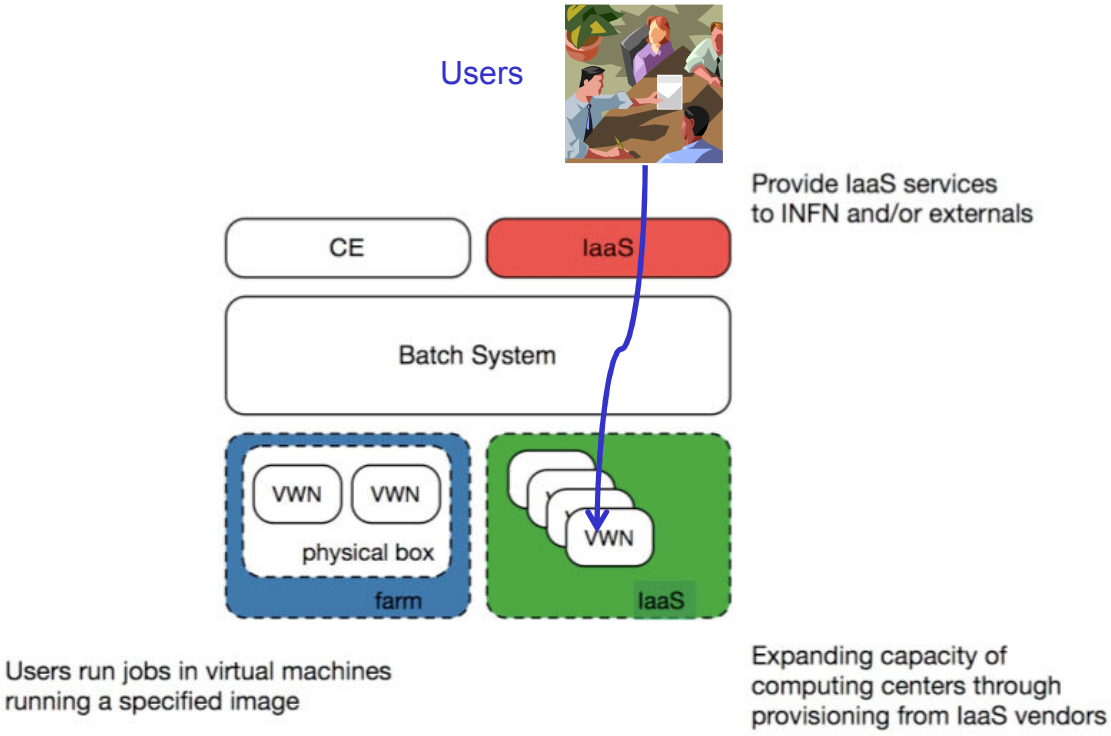
Path 2: Extending Grid-aware Virtual Provisioning



Path 3: Resource Provisioning through EC2-like Mechanisms



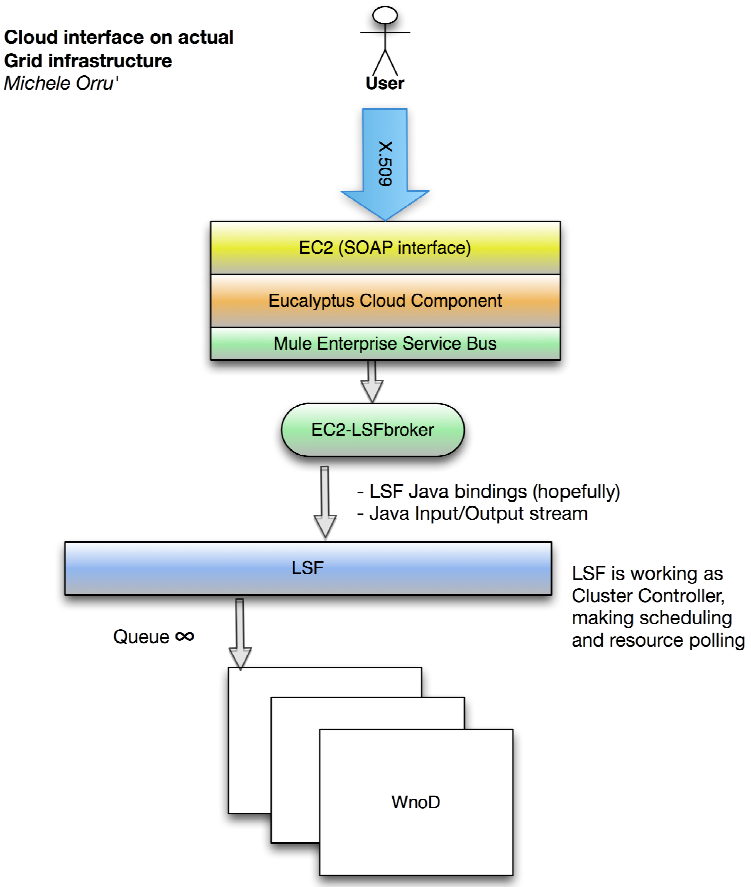
Path 4: Extending EC2-like Resource Provisioning



Integrating the Cloud

- Building on Eucalyptus
  - An open-source cloud computing platform using Amazon EC2 Web Services; the Amazon EC2-api-tools (command line client) are fully compatible with Eucalyptus, and used as the preferred method to interact with it.
- Building on our WNOD solution
- Working on replacing Eucalyptus' VMcontrol scheduler with LSF
  - Goal: migrate resource provisioning and polling to LSF, maintaining compatibility with the rest of the EC2 API's, without modifying the Web Services (WSDL) interface on top of it

Cloud over existing ("Grid") resources



- Open points:
  - Network isolation (dynamic VLANs), public vs. private addresses
  - Proper configuration of the LSF layer is obviously critical
    - QoS / resource reservation mechanisms may be needed
  - Billing

Thanks

- Questions welcome
- Offline: email to Davide.Salomoni@cnaf.infn.it