GPU Computing in EGI environment using Cloud approach

F. Vella, R. Cefalà and **O. Gervasi**
Dept of Mathematics and Computer Science
University of Perugia, Italy
osvaldo@unipg.it

April 10, 2013
Manchester, United Kingdom

Introduction

- GPU Computing has generated considerable interest in the scientific community
- Some EGI Virtual Organizations are reshaping their applications to exploit the new programming paradigm.
- Cloud Computing implements a transparent use of computational resources
- We provide the access to a GPU environment on EGI using a Cloud approach

Introduction

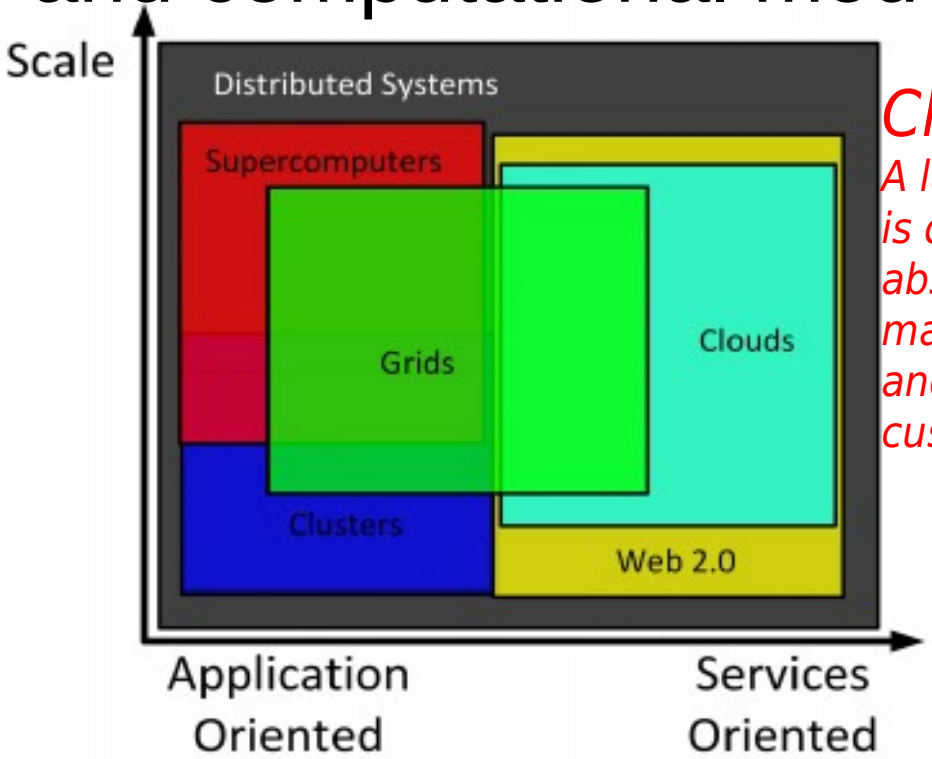
- GPU Computing is pushing the development of distribution models aimed at integrating HPC and HTC resources.
- We provide on-demand execution environments through the joint usage of g-Lite components and the EC2 web-service API.
- The entire Job flow has been defined, enabling the request of GPU resources through JDL parameters, dynamically allocating the resources in a Cloud-like infrastructure

Grid and Cloud

- Grid and Cloud paradigms share some essential driving ideas and overlapping areas which lead to
  - Encapsulate the complexity of hardware resources and make them easily accessible by means of high-level user interfaces
  - Address the intrinsic scalability issues of large scale computational challenges
  - Cope with the need of resources that cannot be hosted locally

Grid and Cloud

- Among the key differences between Grid and Cloud there are those related to abstraction and computational models



Cloud definition:

A large-scale distributed computing paradigm tha is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.

Grid and Cloud

- Clouds enable the users to choose between different computational models suited to match the requirements of a particular application
- This allow to encompass some limits of the batch model of the Grid, where resources are often statically managed and partitioned preventing any possible dynamical arrangement to match the application requirements.
- The workload in Grid is often unpredictable leading to unbalanced distribution on the use of resources and to a reduction of QoS

Grid and Cloud

- Cloud Computing presents still some weak aspects that require further developments which are instead well-established in Grid, i.e:
  - Security
  - Data management
- The more reasonable approach could be an **integration model** that combines the features of both paradigms

Grid and Cloud integration

- Hybrid with batch-dependent cloud-enabled LRMS
  - a single local batch system is used to schedule the jobs on a pool of dynamically provisioned re-sources on public/private clouds
- Hybrid batch independent
  - the local Grid site spawns resources on public/ private clouds on the basis of job requests. The integration is done at the Computing Element level. It enables the creation of multiple virtual clusters, including LRMS.

HPC in EGI: Multi/Manycore

- The opportunity to using heterogeneous resources (CPUs and GPU) as compute units can solve the need of HPC resources.
- An application written in OpenCL or CUDA is inherently parallel but the management of its workflow is simple like a single job.
- The cost of communications among threads is smaller in a single node in respect of an MPI one
- An application as the like would be less dependent from the site peculiarities.
- Not all HPC scientific application can be implemented in a Many/Multicore fashion.

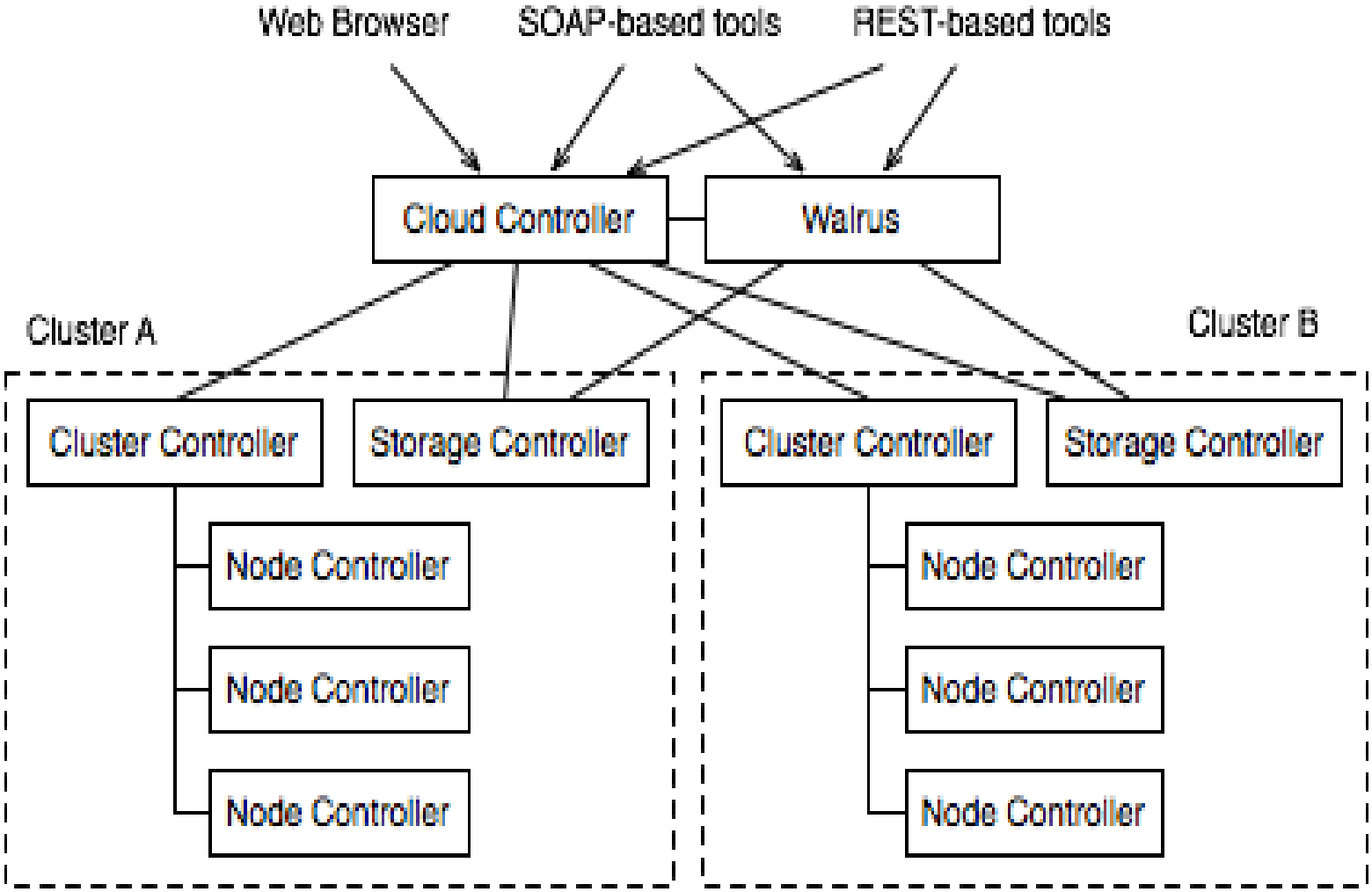
A simple integration model

- A typical Grid site runs an application that “speaks” with the CREAM CE BLAH component and an IaaS provider
- To create a working testbed we chose and implemented a simple infrastructure for the provision of GPU virtual instances using Eucalyptus on top of the Xen Hypervisor
- The interfaces to the Cloud Controller are Amazon EC2 compliant and we used boto API to develop our testbed

A simple integration model

- The physical GPU resources have been virtualized and made available in a Infrastructure as a Service (IaaS) private Cloud.
- A centralized mechanism that interfaces a generic EGI Grid site to an IaaS provider has been implemented to fulfill job requirements.
- The fully working testbed has been built with the adoption of Eucalyptus software system to implement a private Cloud over the cluster.
- We created virtual appliances for OpenCL or CUDA to match job requirements related to Multi/Manycore technology.

IaaS architecture

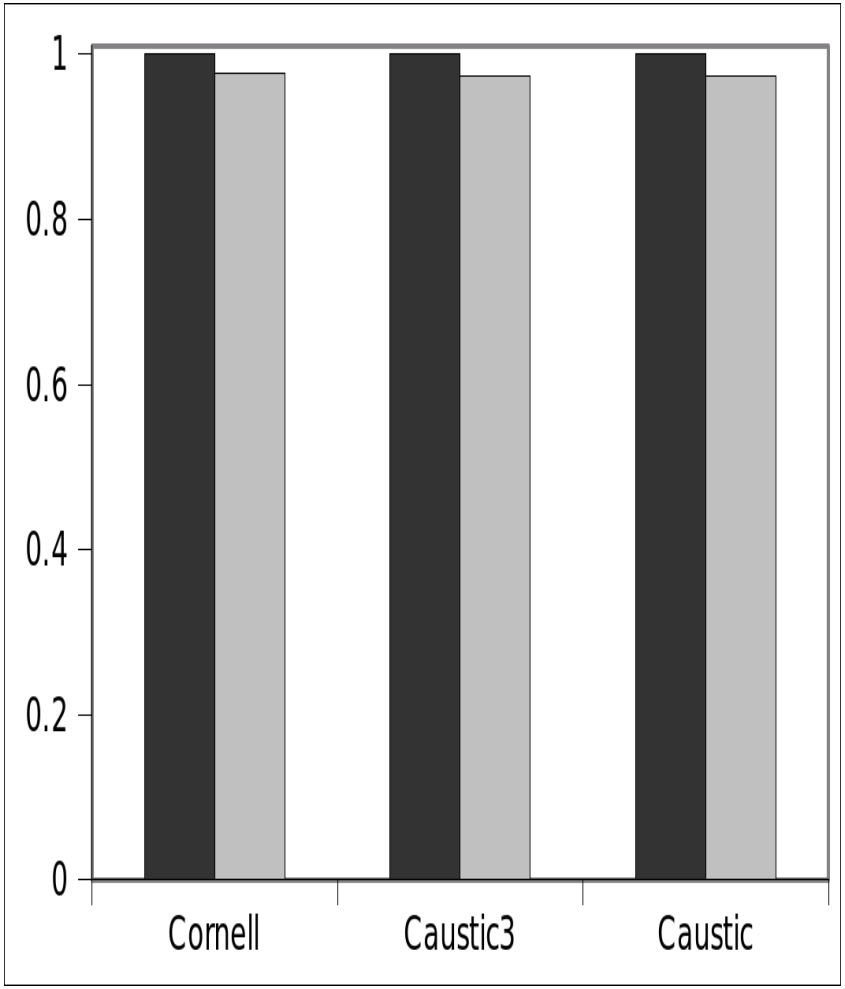


Xen and GPU virtualization

- When hardware prerequisites are met, Xen permits to transparently assign PCIE devices to Virtual Machines.
- A single PCIE resource cannot be shared between Virtual Machines (unless one separate and decouple the graphic driver between front-end and back-end).
- At the present time only a few GPUs officially support passing through Virtual Machines.
- The selection of commodity components, new chipsets, motherboards, CPUs and GPUs require a particular care since some components are not compatible with full hardware virtualization.

OpenCL performances

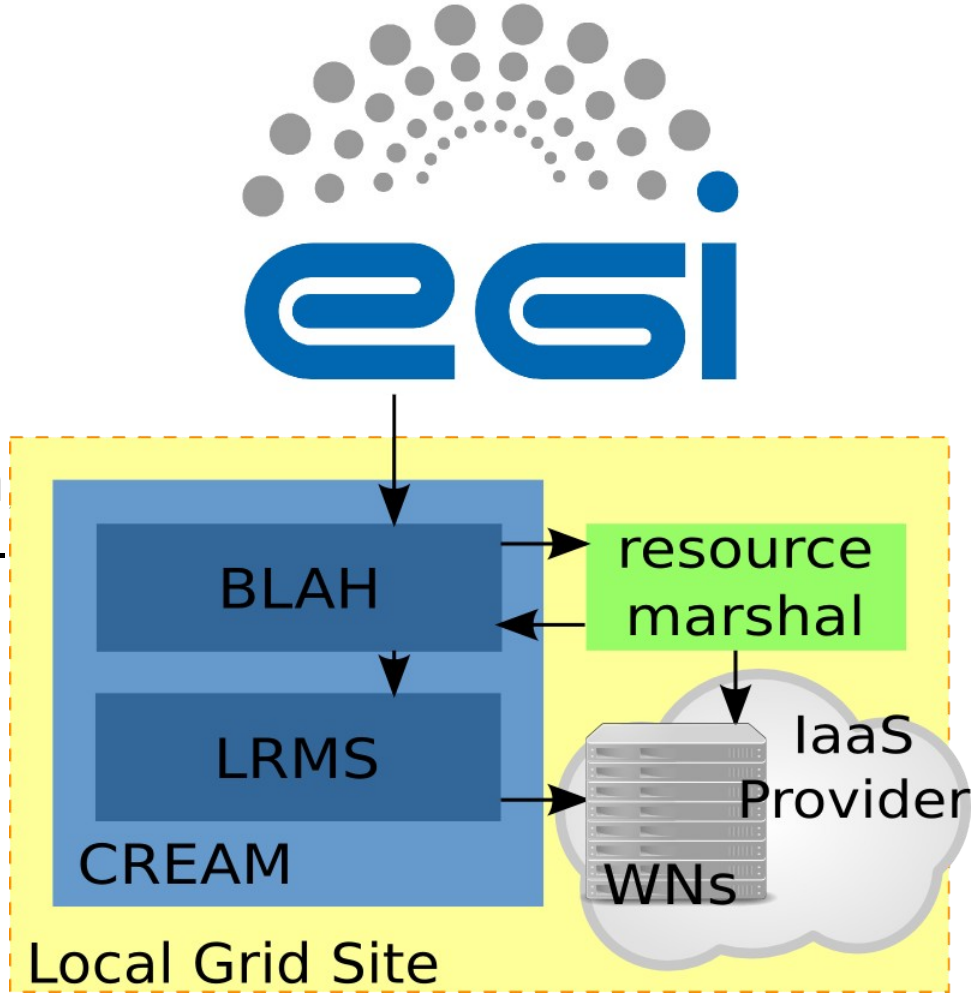
- To assess OpenCL performance gap between a real machine and a virtual one a testbed has been implemented using an **Asus P7P55** LX motherboard with support of both Intel Virtualization Technology (VT-x) and Intel Virtualization Technology for Directed I/O (VT-D) provided by **I7 870** CPU, a 4 cores, 64-bit, x86 part from INTEL. Graphic Adapter: **AMD FireStream 9270**.
- We used Phoronix test suite benchmark
- The virtual GPU is **3% slower**



Black=Real GPU Gray=Virtual GPU

The implemented system

- The main components are: Grid site CE and BLAH, a demon named resource-marshal and an IaaS provider that exposes EC2 interfaces.
- For each job the CE receives, after the BLAH parsing operation we are able to gather all information for that job and marshal its execution.
- To this end we implemented a basic daemon that manages the allocation of instances according to Job information and simple policies



The implemented system

- Job information are used to determine the flavor of the required virtual appliance, specified via GlueSchema in the JDL
- If an instance is already available the job proceeds immediately. To implement such behavior a call to simple client application is interposed to the BLAH XXX_job_submit scripts that inquires the resource-marshal daemon about the availability of resources

```
Type = "job";
executable="/bin/sleep";
[...] CeRequirements=
    "other.GlueHostMainMemoryRAMSize >
    2048 && (Member( "GPU\", other .
     GlueHostApplicationSoftwareRuntimeEnvironment)
    )";
```

- The daemon keeps track of available instances and sends requests of new ones to the IaaS provider via boto EC2 interfaces.
- We haven't applied a strict termination policy for the instances, however the possibilities are:
  - Job termination events can be used to trigger the reclamation of unused instances
  - Grid users proxy lifetime could be bound to the instances lifecycle.

Conclusions

- We presented a system to provide to EGI users a specific on-demand GPU environment to transparently execute jobs on GPU devices
- The proposed system uses a Cloud approach, based on EC2 Compliant Clouds, in order to control the specific GPU-enabled VO environment from EGI middleware interfaces.
- The proposed solution enables the users to run their applications in parallel, exploring the Many/Multicore capabilities of the Grid nodes.
- This innovative approach provides to the users an interesting alternative to MPI for running parallel jobs