PROCEEDINGS OF SCIENCE

EGI federated platforms supporting accelerated computing

Paolo Andreetto
INFN, Sezione di Padova
Via Marzolo 8, 35131 Padova, Italy
E-mail: Paolo.Andreetto@pd.infn.it

Jan Astalos
Institute of Informatics Slovak Academy of Sciences
Bratislava, Slovakia
E-mail: Jan.Astalos@savba.sk

Miroslav Dobrucky
Institute of Informatics Slovak Academy of Sciences
Bratislava, Slovakia
E-mail: Miroslav.Dobrucky@savba.sk

Andrea Giachetti
CERM Magnetic Resonance Center
CIRMMP and University of Florence, Italy
E-mail: giachetti@cerm.unifi.it

David Rebatto
INFN, Sezione di Milano
Via Celoria 16, 20133 Milano, Italy
E-mail: David.Rebatto@mi.infn.it

Antonio Rosato
CERM Magnetic Resonance Center
CIRMMP and University of Florence, Italy
E-mail: rosato@cerm.unifi.it

Viet Tran
Institute of Informatics Slovak Academy of Sciences
Bratislava, Slovakia
E-mail: viet.ui@savba.sk

Marco Verlato[1]
INFN, Sezione di Padova
Via Marzolo 8, 35131 Padova, Italy
E-mail: Marco.Verlato@pd.infn.it

Lisa Zangrando
INFN, Sezione di Padova
Via Marzolo 8, 35131 Padova, Italy
E-mail: Lisa.Zangrando@pd.infn.it

While accelerated computing instances providing access to NVIDIA™ GPUs are already available since a couple of years in commercial public clouds like Amazon EC2, the EGI Federated Cloud has put in production its first OpenStack-based site providing GPU-equipped instances at the end of 2015. However, many EGI sites which are providing GPUs or MIC coprocessors to enable high performance processing are not directly supported yet in a federated manner by the EGI HTC and Cloud platforms. In fact, to use the accelerator cards capabilities available at resource centre level, users must directly interact with the local provider to get information about the type of resources and software libraries available, and which submission queues must be used to submit accelerated computing workloads. EU-funded project EGI-Engage since March 2015 has worked to implement the support to accelerated computing on both its HTC and Cloud platforms addressing two levels: the information system, based on the OGF GLUE standard, and the middleware. By developing a common extension of the information system structure, it was possible to expose the correct information about the

[1]Speaker

PoS(ISGC2017)020

accelerated computing technologies available, both software and hardware, at site level. Accelerator capabilities can now be published uniformly, so that users can extract all the information directly from the information system without interacting with the sites, and easily use resources provided by multiple sites. On the other hand, HTC and Cloud middleware support for accelerator cards has been extended, where needed, in order to provide a transparent and uniform way to allocate these resources together with CPU cores efficiently to the users. In this paper we describe the solution developed for enabling accelerated computing support in the CREAM Computing Element for the most popular batch systems and, for what concerns the information system, the new objects and attributes proposed for implementation in the version 2.1 of the GLUE schema. For what concerns the Cloud platform, we describe the solutions implemented to enable GPU virtualisation on KVM hypervisor via PCI pass-through technology on both OpenStack and OpenNebula based IaaS cloud sites, which are now part of the EGI Federated Cloud offer, and the latest developments about GPU direct access through LXD container technology as a replacement of KVM hypervisor. Moreover, we showcase a number of applications and best practices implemented by the structural biology and biodiversity scientific user communities that already started to use the first accelerated computing resources made available through the EGI HTC and Cloud platforms.

International Symposium on Grids and Clouds 2017 -ISGC 2017-
5-10 March 2017
Academia Sinica, Taipei, Taiwan

1. Introduction

The EGI infrastructure is a federation of 20 cloud providers and over 300 data centres, spread across Europe and worldwide. As of November 2016, EGI offers more than 730,000 CPU-cores for High Throughput Compute, 6,600 cores for cloud compute, and more than 500 PB of data storage. The infrastructure operations have been supported since 2010 by two European projects: EGI-InSPIRE (2010-2014) and EGI-Engage (2015-2017). In 2012 a EGI Working Group was set up to assess the interest of EGI community (both users and resource providers) about General-Purpose Graphics Processing Units (GPGPUs, shortened to GPUs in the rest of the article). A survey completed in September 2012 [1] showed that:

- 92.9 % of the users would be interested in accessing remote GPU based resources through a computing infrastructure
- 30.2 % of the resource centres were already providing GPU resources, and 56.3 % planned to do in the incoming 24 months.

In the following years Accelerated Computing has become increasingly popular. The term refers to a computing model used in scientific and engineering applications whereby calculations are carried out on specialized processors (known as accelerators) coupled with traditional CPUs to achieve faster real-world execution times. Nowadays accelerators are highly specialized microprocessors designed with data parallelism in mind, and more in general other than GPUs they include Xeon™ Phi™ coprocessors, based on Intel™ Many Integrated Core (MIC) architecture, and specialized Field Programmable Gate Array (FPGA) PCIe cards. They allow to reduce execution times by offloading parallelizable computationally-intensive portions of an application to the accelerators while the remainder of the code continues to run on the CPU.

Unfortunately, in the current implementation of the EGI infrastructure, there is neither way to describe/discover this kind of resources, nor to allocate them together with CPU cores to efficiently execute user jobs. A dedicated task of the EGI-Engage project was therefore designed to address these issues, with the ultimate goal to provide a complete new Accelerated Computing platform for EGI. The task activities started in March 2015, with two main objectives: to implement accelerated computing support in the information system, based on the OGF GLUE standard [2] schema evolution; to extend the current EGI HTC and Cloud platforms introducing middleware support for accelerator cards. User communities, grouped within EGI-Engage in entities called Competence Centres (CCs), had a driving role in setting the requirements and providing a number of use cases. In particular, LifeWatch CC captured the requirements of the Biodiversity and Ecosystems research community for deploying GPU based e-infrastructure services supporting data management, processing and modelling for Ecological Observatories. It provided an Image Classification Deep Learning Tool as use case. MoBrain CC instead captured the requirements of the Structural Biology scientific domain, aiming at deploying portals for bio-molecular simulations leveraging GPU resources. Their use cases involved popular molecular dynamics packages like AMBER and GROMACS, and 3D bio-molecular structure model fitting software like PowerFit and DisVis.

The paper is organized as follows. Section 2 describes the development work carried out to enhance the EGI HTC platform in order to fully exploit accelerator cards. Section 3 illustrates

how the support to GPUs, based on PCI pass-through virtualisation technology, has been introduced in the EGI Federated Cloud by properly configuring and tuning the OpenStack and OpenNebula cloud frameworks. Section 4 shows a number of applications that firstly used the accelerated platform to enhance their performance, and finally Section 5 provides a summary and the future perspectives.

2. The HTC Accelerated Platform

A snapshot of the EGI Grid Information System taken in October 2015 [1] has shown that the 75% of the 567 Compute Elements (CEs) composing the EGI HTC platform is based on CREAM (Computing Resource Execution And Management): a web service based CE for the UMD/gLite grid middleware [3]. CREAM has been since years the most popular grid interface in EGI for a number of Local Resource Management Systems (LRMS): TORQUE, LSF, SGE, Slurm, HTCondor. Nowadays, most recent versions of these LRMS provide native support to GPUs, MIC coprocessors and other accelerators, meaning that computing nodes hosting these cards and managed by one of the above LRMS can be selected by specifying the proper LRMS directives and parameters. This suggested the definition of a work plan to implement an HTC Accelerated Platform based on an enhanced CREAM CE in the following steps:

1. Identification of the relevant GPU/MIC/FPGA related parameters supported by any LRMS and abstract them to useful JDL attributes.
2. Implementation of the needed code changes in the relevant CREAM components.
3. Extension of the current version 2.1 draft GLUE schema for describing the accelerator information.
4. Development of information providers according to the extended GLUE 2.1 draft specifications.
5. Test and certification of the enhanced CREAM CE prototype.
6. Release of a CREAM CE updated version with full support for accelerators.

Considering that TORQUE is the most popular choice of LRMS in EGI for the CREAM CE instances, we started to work with a testbed composed by 3 nodes, each one with 2 Intel™ Xeon™ E5-2620 v2 and 2 NVIDIA™ Tesla™ K20m GPUs, and a server hosting TORQUE 4.2.10 (compiled with NVIDIA™ NVML libraries) with MAUI 3.3.1 as scheduler. The testbed was made available by CIRMMP team at the University of Florence, in Italy.

2.1 Job submission

User job submission towards EGI HTC platform typically happens through VO level services like DIRAC [4] or gLite-WMS [5], which act as resource broker to select the best suited CE where to execute the user job, according to job requirements specified in the JDL file. The Job Description Language (JDL) is a high-level, user-oriented language based on Condor classified advertisements (classads) for describing jobs to be submitted to a resource broker or directly to the CREAM CE service. We'll consider here only direct job submission to a CREAM CE, through the `glite-ce-job-*` command line. Being the JDL an extensible language [6] the user is allowed to use whatever attribute for the description of a request without incurring in errors from the JDL parser. However, only a certain set of attributes, that we will refer as "supported attributes" from now on, is taken into account by the CREAM CE service. Figure 1

below shows a high level architecture of the CREAM CE service. It is essentially a grid gateway with a Web Service interface to allow remote authorized (via VOMS service) users to submit jobs to a cluster of compute nodes managed by one of the popular LRMS listed above.

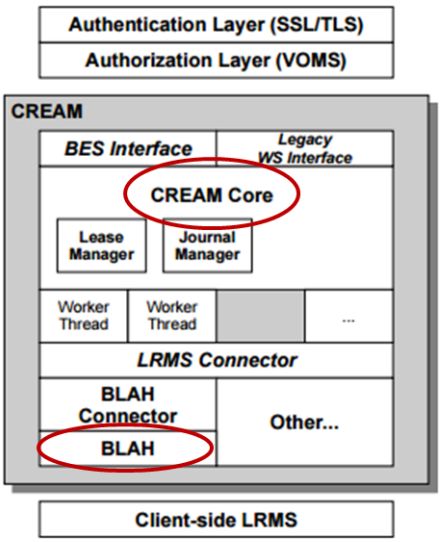

Figure 1: High level architecture of CREAM CE service. Circles highlight the components to be enhanced to support accelerated computing.

The CREAM core is a Java component where JDL supported attributes are defined and parsed from the user request. The BLAH (Batch Local Ascii Helper) component is an abstraction layer for submission and control of job requests to a local batch system [7]. It is a daemon written in C language coupled to BASH scripts that allow to parse the JDL supported attributes and translate them in the proper parameters to be used as input arguments to the specific directives corresponding to a given LRMS.

The testbed set up at CIRMMP was used to implement the first prototype of a GPU-enabled CREAM CE in December 2015. Starting from the local TORQUE/Maui job submission specific syntax, like e.g. the one for a job requesting one node with one GPU:

```
$ qsub -l nodes=1 -W x='GRES:gpu@1'
```

the new JDL attribute “GPUNumber” was defined and the CREAM core and BLAH components were patched in order to allow the CREAM CE to understand the new attribute and properly map it into the suitable TORQUE/Maui directive. This way the user JDL file containing the line GPUNumber=1 would imply the job to be enqueued in the LRMS until a worker node hosting at least one GPU card becomes available. Unfortunately, native support to GPUs or other accelerator cards of TORQUE LRMS coupled with Maui [8], a popular open source job scheduler whose support was discontinued in favour of its commercial successor Moab, was found very limited and could not allow to abstract other interesting attributes at JDL level. On the other hand, the integration of CREAM CE with Moab, which implements full support to accelerators was not in the scope of this activity.

A deeper analysis of latest versions of the other CREAM supported LRMS, namely HTCondor, LSF, Slurm and SGE, allowed instead to identify two additional JDL attributes that

could be defined and implemented to exploit the more advanced native accelerator support of these batch systems:

- The attribute "GPUModel" was defined to target worker nodes with a given model of GPU card.
- The attribute "MICNumber" was defined to target worker nodes with a given number of MIC coprocessors.

The implementation and test of a new CREAM CE prototype supporting the two additional attributes was carried out since March 2016 thanks to the collaboration of four EGI resource centres that deployed the prototype besides their production CREAM CE on top of their cluster managed respectively by HTCondor at GRIF/LLR data centre, Slurm at ARNES data centre, LSF at INFN-CNAF data centre, SGE and Slurm at Queen Mary University of London (QMUL) data centre.

Figure 2 below shows an example of JDL file describing a structural biology job targeting a worker node hosting at least one NVIDIA™ Tesla™ K80 GPU. It was submitted to the QMUL grid site where the CREAM CE prototype has been deployed as interface to their production Slurm LRMS. Figure 3 displays an extract of its gres.conf and slurm.conf files showing that two worker nodes among the ones forming the entire cluster are equipped respectively with 8 CPU cores plus one NVIDIA™ Tesla™ K40c, and 32 CPU cores plus four NVIDIA™ Tesla™ K80 GPUs.

```
[
  executable = "disvis.sh";
  arguments = "10.0 2";
  stdoutput = "out.txt";
  stderror = "err.txt";
  inputSandbox = { "disvis.sh" ,"O14250.pdb" , "Q9UT97.pdb" , "restraints.dat" };
  outputsandboxbasedesturi = "gsiftp://localhost";
  outputsandbox = { "out.txt" , "err.txt" , "results.tgz"};
  GPUNumber=1;
  GPUModel="teslaK80";
]
```

Figure 2: Example of JDL file targeting a worker node with one NVIDIA™ Tesla™ K80 GPUs.

```
NodeName=cn456 Name=gpu Type=teslaK40c File=/dev/nvidia0
NodeName=cn290 Name=gpu Type=teslaK80 File=/dev/nvidia[0-3]

NodeName=cn456 CPUs=8 Gres=gpu:teslaK40c:1 RealMemory=11902 Sockets=1 CoresPerSocket=4…
NodeName=cn290 CPUs=32 Gres=gpu:teslaK80:4 RealMemory=128935 Sockets=2 CoresPerSocket=8…
```

Figure 3: Excerpt of Slurm configuration files gres.conf and slurm.conf where GPU support is implemented.

The implementation of GPU/MIC related JDL attributes assumed that the local batch systems already configured the GPU/MIC support according with the official LRMS specific documentation for HTCondor [9], LSF [10] and Slurm [11]. Of course, to let the user know if QMUL grid site has GPUs on board, and which vendor/model is available, the information system has to be properly extended.

For what concerns the applications run in the worker node, the usual grid way to distribute software via CVMFS tool could not be used because of the complex software dependencies. In fact, GPU specific application software components generally depend on the drivers of a given

GPU card, and different driver versions for the same card can be installed on different worker nodes at the various resource centres. To avoid such conflicts the applications were built inside a number of Docker containers, each one in turn built with a different NVIDIA™ driver. Six containers with the latest six more recent NVIDIA™ drivers were made available in the Docker Hub for the structural biology applications in collaboration with INDIGO-DataCloud project [12], in order to allow the exploitation of the three grid sites hosting NVIDIA™ GPUs at CIRMMP, QMUL and ARNES. To avoid security issues, and then releasing the requirement to have the Docker engine installed and running on each worker node, the Udocker tool was used. Udocker is a basic user tool developed by INDIGO-DataCloud to execute simple Docker containers in user space without requiring root privileges. Figure 4 shows the script acting as executable in the JDL file example of Figure 2. After landing on the worker node, the driver information is collected and used through the Udocker tool to pull the right Docker image from which to create the container. The application is then executed inside the container.

```
#!/bin/sh
driver=$(nvidia-smi | awk '/Driver Version/ {print $6}')
export WDIR=`pwd`
git clone https://github.com/indigo-dc/udocker
cd udocker
./udocker.py pull indigodatacloudapps/disvis:nvdrv_$driver
rnd=$RANDOM
./udocker.py create --name=disvis-$rnd indigodatacloudapps/disvis:nvdrv_$driver
mkdir $WDIR/out
./udocker.py run --hostenv --volume=$WDIR:/home disvis-$rnd disvis
/home/O14250.pdb /home/Q9UT97.pdb /home/restraints.dat -g -a $1 -vs $2
-d /home/out
./udocker.py rm disvis-$rnd
./udocker.py rmi indigodatacloudapps/disvis:nvdrv_$driver
cd $WDIR
tar zcvf results.tgz out/
```

Driver identification

Install Udocker tool

Pull DisVis image

Create the container

Run the container executing DisVis

Figure 4: Example of script for running DisVis application on the grid using GPU driver dependent Docker containers.

2.2 Information System

In order to introduce the concept of accelerator device in a Grid Computing Service an enhancement of the GLUE 2.0 schema is required. The proposed changes, as encompassed in the GLUE 2.1 draft, consist on:

- The definition of a new GLUE object, the Accelerator Environment, representing a set of homogeneous accelerator devices.
- New attributes and associations for existing objects such as Computing Manager, Computing Share and Execution Environment.

Two reasons lead to considering a new GLUE object instead of extending the Execution Environment described in GLUE 2.0 [13]:

- Many accelerator types, each one with its own specific features, can be installed in or linked to an environment. The outcome of extending the environment with many

complex attributes would have been cumbersome and very far from the inner meaning of the GLUE specification.

- A many-to-many relationship between accelerator object and environment must be taken into consideration in order to support external GPU appliances. In that case the Accelerator Environment is completely decoupled, not only from the logical point of view but also from the physical one, from an Execution Environment.

An Accelerator Environment reports the number of physical accelerators, the amount of memory available and any detail about the hardware such as the vendor, the model and the clock speed. The information published with an Accelerator Environment object must be considered immutable or statically defined. It changes only if a hardware restructuring occurs.

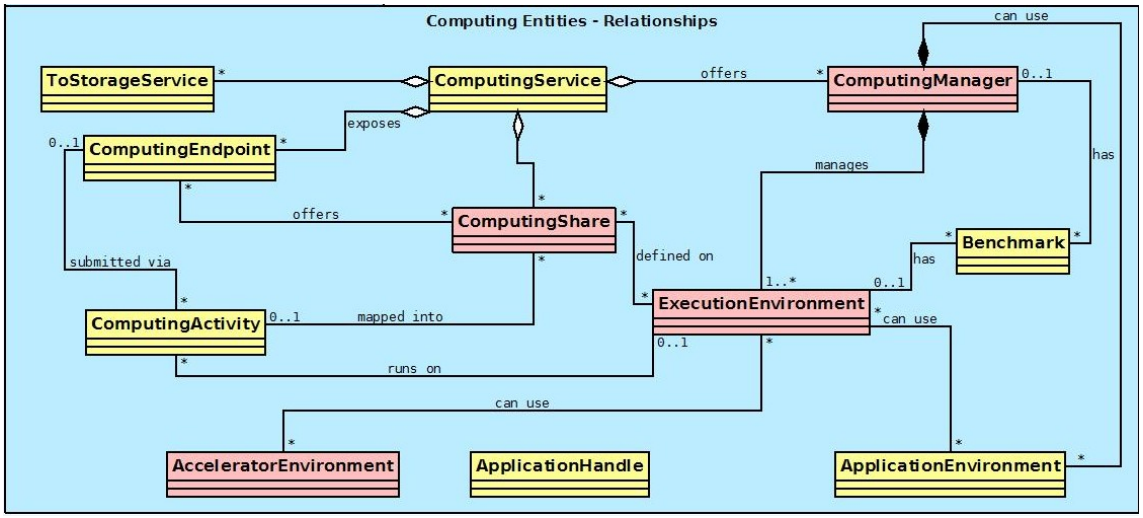

Figure 5: UML diagram of GLUE 2.1 draft Computing Entities. In red the classes involved in the description of the accelerators

The main concept behind the new defined attributes in Computing Share and Computing Manager is the accelerator slot. It represents the minimum amount of a GPU resource that can be allocated to a job. Since a GPU can be efficiently shared among multiple processes, the definition of accelerator slot may be quite complex. In accordance to the meaning of the related JDL attribute, GPUNumber, previously described the accelerator device is considered to work in "exclusive execution mode". Each process has the control of an accelerator device. The information published with the new attributes of Computing Share and Computing Manager can vary according to the resource usage, for example the number of free accelerator slots for a Computing Share, or the runtime configuration of a batch system, such as the total number of accelerator slots for a Computing Manager.

The information about the accelerator driver or generally any related software tool can be published through the Application Environment object, as declared in the GLUE 2.0 schema. It is not necessary to modify the definition of Application Environment in order to have an association with one or more Accelerator Environments. Since there is a many-to-many association between the Application Environment and the Execution Environment, the latter can be used as a bridge for expressing the relationship between the driver and the device.

The conceptual model of the Grid Computing Service with the accelerator information is being defined in GLUE2.1 draft and shown in Figure 5.

The framework adopted by the EGI HTC platform for managing the information system is the Berkeley Database Information Index (BDII) [14].

The BDII is structured as a hierarchy of nodes, with the "top BDII" one at the root level, referencing all the information provided by the other nodes, and the "resource BDII" node at the bottom, collecting data from the grid resources. Gathering information is carried out by the resource BDII in two different ways:

- Reading statically defined descriptions of resources.
- Running a set of executables, called information providers, and catching the output.

The Accelerator Environment is published by the resource BDII using a statically defined description. For a CREAM CE node such a description is built by the deployment and configuration system, based on the Puppet suite. Figure 6 shows an excerpt from the Puppet configuration file (hiera) which describes an accelerator device installed into a worker node.

```
creamce::hardware_table :
    environment_001 : {
        ce_cpu_model : XEON,
        ce_cpu_speed : 2500,
        # other definitions for the worker node
        accelerators : {
            acc_device_001 : {
                type : GPU,
                log_acc : 2,
                phys_acc : 2,
                vendor : NVIDIA,
                model : "Tesla K20m",
                memory : 5119
            }
        }
    }
```

Figure 6: Excerpt of a Puppet configuration file for a GPU installed in a worker node

Every attribute related to the accelerator slots and published within Computing Manager and Computing Share objects is calculated by an information provider, executed by the resource BDII. Many batch systems integrate the support for information retrieval from popular accelerator devices like NVIDIA™ GPU devices. The TORQUE management system, for example, can identify the structure of the NVIDIA™ devices installed in the worker nodes, and many parameters reporting the quality of service, GPU and memory usage. However in many cases the support is not able to give the complete control over all the required details of an accelerator device. Besides it is more advisable to avoid the development of many information providers, each one batch system specific, if a common solution can be identified. For NVIDIA™ devices the common solution consists on an information provider which is able to:

- Run the NVIDIA™ System Management Interface (nvidia-smi) on each active worker node.
- Aggregate the data harvested according to the GLUE 2.1 schema.

Since the executable does not depend on any batch system specific feature, the solution is portable. The main drawback is that it requires a further configuration of both the resource BDII

node and each worker node. Any command of nvidia-smi is run through a secure channel, established with openssh tools, and with normal user privileges, therefore accounts and public keys must be distributed over the cluster.

2.3 Accounting

The APEL (Accounting Processor for Event Logs) [15] is the fundamental tool for the CPU usage accounting infrastructure deployed within EGI. As a log processing application, it interprets logs of CREAM CE and its underlying LRMS to produce CPU job accounting records identified with grid user identities. APEL then publishes accounting records into a centralised repository at a Grid Operations Centre (GOC) for access from the EGI Accounting Portal. The functions of log files parsing, record generation and publication are implemented by the APEL Parser, APEL Core, and APEL Publisher component respectively. The APEL developers were therefore involved in the GPU accounting discussion. Their analysis concluded that a little effort to modify APEL Parser and Core functions had been required if the LRMS would be able to report the GPU usage attributable to each job in their log files. Unfortunately, a detailed analysis of the job accounting records reported in the log files of TORQUE, LSF, Slurm, HTCondor and SGE showed that they don't contain GPU usage information. For the most recent NVIDIA™ GPUs, the NVIDIA™ Management Library (NVML) allows through the NVIDIA™ System Management Interface (nvidia-smi) tool to enable per-process accounting of GPU usage using Linux PID, as shown in the example output of Figure 7:

```
$ nvidia-smi --query-accounted-apps=pid,gpu_serial,gpu_name,gpu_utilization,time --format=csv
pid, gpu_serial, gpu_name, gpu_utilization [%], time [ms]
44984, 0324713033232, Tesla K20m, 96 %, 43562 ms
44983, 0324713033232, Tesla K20m, 96 %, 43591 ms
44984, 0324713033096, Tesla K20m, 10 %, 43493 ms
44983, 0324713033096, Tesla K20m, 10 %, 43519 ms
```

Figure 7: Per-process GPU utilization accounting example using nvidia-smi tool

Regrettably, this functionality is not yet integrated in any of the considered LRMS. In principle, developing suitable prologue/epilogue scripts for any given LRMS would allow to implement per-job accounting starting from the per-process GPU accounting provided by NVML. However, the estimated effort for developing and sustaining in the long term such kind of solution was considered not affordable by the APEL team with their current and future planned funds. The accounting of GPU usage in the EGI HTC platform was therefore temporarily abandoned, pending its possible native support in future versions of the considered LRMS.

3. The Cloud Accelerated Platform

KVM with PCI pass-through virtualisation technology for GPUs is rather mature, but maximum one VM can be attached to one physical card, as shown on the left of Figure 8. Other virtualisation technologies that allow to share the physical GPUs among many virtual machines are available, e.g. NVIDIA™ GRID™ vGPU for XenServer and VMWare hypervisors, SR-IOV based AMD™ MxGPU for VMWare hypervisors, and Intel™ GVT-G recently added to Linux 4.10 kernel. However, these are not yet supported by KVM, the most popular open source

hypervisor deployed in the large majority of the cloud resource centres part of the EGI Federated Cloud.

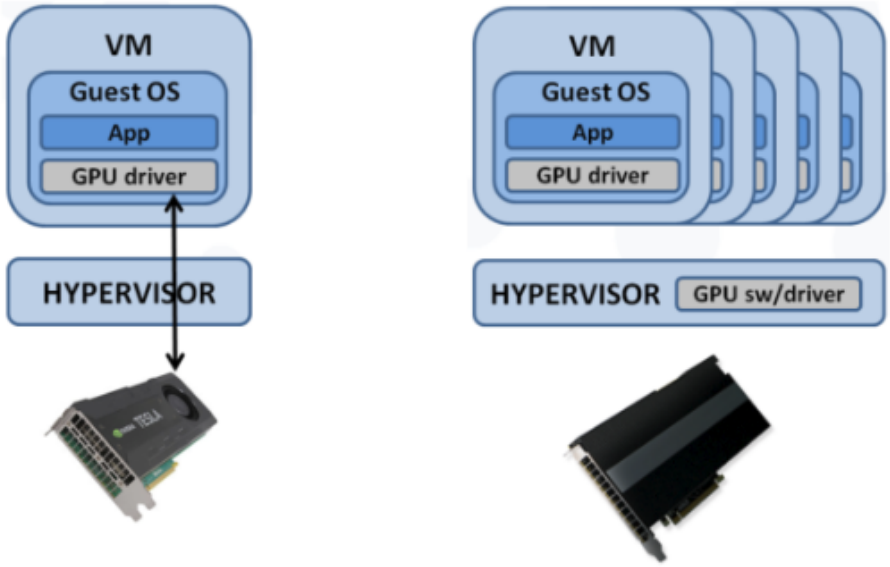

Figure 8: PCI pass-through virtualisation of GPUs (left) versus sharing virtual GPU technology (right).

3.1 Accelerated computing with OpenStack

As mentioned above, the technology for sharing virtual GPUs is not supported by KVM for NVIDIA™ Tesla cards, therefore PCI pass-through has been used. OpenStack has built-in support for PCI pass-through from Grizzly release. That includes the following changes in the OpenStack Nova:

- Implementation of support for PCI pass-through mechanism on compute nodes for configuring and creating VMs with PCI pass-through enabled. The PCI devices for pass-through are defined in the configuration file of OpenStack Nova as whitelist "pci_pass-through_whitelist = { "vendor_id": "8086", "product_id": "10fb" }", its alias will be assigned to VM flavour as properties.
- Implementation of scheduler supporting PCI pass-through to correctly allocate VMs with PCI pass-through enabled to compute nodes with available resources. With the support of scheduler, VMs with PCI pass-through can be mixed with normal VMs on the same OpenStack deployment.

The first cloud computing testbed supporting GPU cards was integrated into EGI at IISAS as the "IISAS-GPUCloud" site, which comprises four IBM dx360 M4 servers each with 2x Intel™ Xeon™ E5-2650v2 (16 CPU cores), 64GB RAM, 1TB storage and 2x NVIDIA™ Tesla™ K20m, running Ubuntu 14.04 LTS.

The initial setup demonstrated that KVM PCI pass-through was not stable, compute nodes crashed randomly. The VMs had direct access to PCI devices and every misconfiguration in VMs could send NMI (non-maskable interrupt) signal to the hardware that can be propagated to the host machine and caused the system crash. The workaround solution is to change BIOS setting for isolating NMI of the pass-through device. However, a more consistent solution where

the host operating system can retain control over hardware is critical for overall stability of the system. A viable alternative is to use LXD [16] container hypervisor as replacement for KVM hypervisor, as discussed in the next subsection 3.3.

3.2 Accelerated computing with OpenNebula

Since version 4.14 the OpenNebula cloud computing framework supports PCI pass-through virtualisation including GPU accelerators. By default, PCI devices can be specified in OS templates using vendor, device and class PCI codes. This approach requires the system administrator to define special templates for every combination of virtual appliance images and GPUs provided by worker nodes. It gets complicated if the site provides more than one type of accelerator and worker nodes have more than one accelerator installed. Also, the integration of GPU accelerators in the OpenStack cloud computing framework follows an approach where GPU accelerators are requested using resource template, therefore for compatibility reasons the rOCCI server for OpenNebula had to be modified by its developers to allow specification of GPU device PCI codes inside resource templates. To test the integration of the OpenNebula framework in the EGI Federated Cloud, an experimental configuration was set up at CESNET-Metacloud site in May 2016, and a new IISAS-Nebula cloud site was set up and put into production in January 2017.

3.3 Accelerated computing with LXD containers

In the recent years, container technologies have been emerging as a viable alternative of full virtualisation for cloud computing. From the point of view of applications running in the cloud, the full containers like LXD have very similar behaviours like VMs: users can create and log into containers, install/configure software, running applications and terminate the containers when they are no longer needed. In comparison with full VMs managed by hypervisors like KVM, containers have much faster start, lower overheads, and easier access to hardware resources via device mapping. The main limitation of container technologies is that the containers must share the same OS kernel as the hosting machine and users cannot easily change kernel configurations (e.g. loading kernel modules). For use cases with accelerated computing, where VMs are mostly used for computational tasks, such limitations are not relevant.

There are several implementation of Nova driver for OpenStack. However, most of them are still immature at the time of testing (2016). An experimental cloud site has been set up at IISAS to enable GPU support with LXD hypervisor with OpenStack through nova-lxd driver. Unfortunately, the support for configuring GPU device mapping in Nova configuration files of the tested version of nova-lxd was not yet implemented. Thus, the devices had to be hard-coded in the code for testing. The performance and stability of the LXD containers were outperforming the VMs in KVM and practically reached native performance, that is very promising. However, some critical features, mainly the supports for block devices, were still missing, that prevented so far use of the experimental site in production.

3.4 Accelerated computing with Docker containers

Docker is another widely adopted container technology. Unlike LXD which is full OS-level container technology and behave more like hypervisor, Docker containers are application containers with the aim to run single application process inside. It wraps up application software in a complete file system that contains everything it needs to run: code, runtime, system tools, system libraries, therefore the application in Docker is highly portable and independent from the hosting environment.

Docker containers with GPU support can be easily executed in cloud sites supporting GPUs. Users can create a VM with GPU-enabled flavour and image, and run docker with proper mapping to access GPUs via the `--device` option, as for example:

```
$ docker run -it --device=/dev/nvidia0:/dev/nvidia0 \
  --device=/dev/nvidiactl:/dev/nvidiactl \
  --device=/dev/nvidia-uvm:/dev/nvidia-uvm IMAGE /bin/bash
```

3.5 Integration with EGI federated cloud

Like HTC Accelerated platform, the integration with EGI Federated Cloud will require additional support for GPU in Information and Accounting systems.

The conceptual model for a generic Cloud Provider Site does not fit correctly into the GLUE 2.0 schema. The concepts of virtual machines, images and flavours are brand new from the logical point of view. Extending the objects of a classical Computing Service is not enough, it is necessary to redesign the entire schema and create a Cloud Compute Service.

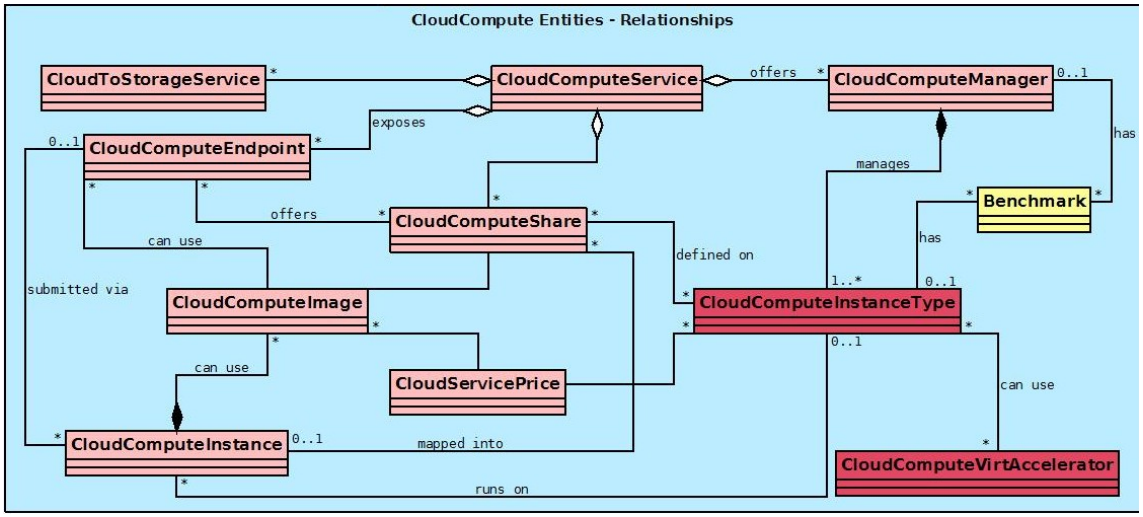

Figure 9: UML diagram of GLUE 2.1 draft Cloud Computing Entities. In light red the classes introduced with GLUE 2.1, in dark red the ones also involved in the description of the accelerators.

The conceptual model of the Cloud Compute Service is being defined in GLUE 2.1 draft specification, as depicted in Figure 9. There's a strong similarity between classes in a Computing Service and a Cloud Compute service. The Cloud Compute Instance Type class describes the hardware environment, or flavour, of the VM, like the Execution Environment

does within a Computing Service. The Cloud Compute Manager, or hypervisor, corresponds to the Computing Manager, or batch system manager, in the classical GLUE 2.0 schema.

In order to keep the similarity even for modelling "virtual accelerator devices" a new entity, the Cloud Compute Virtual Accelerator, has been defined. It describes a set of homogeneous virtual accelerator devices in terms of number of GPUs, amount of memory and hardware details, like vendor and model. A many-to-many association ties the Cloud Compute Instance Type to the Cloud Compute Virtual Accelerator. Since with PCI pass-through virtualisation only one VM at a time can be attached to one or more physical accelerator devices, it does not make sense to have the concept of accelerator slot. Within the Cloud Compute Service an accelerator slot corresponds to the entire device.

GPU accounting is also easier in cloud environment. Considering that the cloud frameworks currently return wall clock time only to the APEL based accounting system, if the wall clock for how long a GPU was attached to a VM is available then the GPU reporting would be in line with cloud CPU time, i.e. wall clock time only. The APEL team has therefore been involved to define an extended usage record and new views to display GPU usage in the EGI Accounting Portal.

To facilitate the testing and development of accelerated applications in EGI, a dedicated Virtual Organization "acc-comp.egi.eu" has been established. A dedicated VO allows to create a separate set of virtual appliance images in the EGI Application Database [17] with pre-installed drivers and libraries (e.g CUDA, OpenCL) and allows to limit the list of sites to the ones that provide relevant hardware. Furthermore, the separated VO allows users to share and exchange their knowledge and experiences related to accelerated computing easier than general-purpose VOs like fedcloud.egi.eu. More information can be found in [18].

4. Applications

As pointed out in section 1, two EGI scientific communities organized respectively in the MoBrain (Structural Biology) and LifeWatch (Biodiversity and Ecosystems) Competence Centres were the early adopters of the EGI Accelerated Platform, setting the requirements and providing a number of use cases exploiting the GPU hardware to enhance the performance of their calculations. These applications are described in the next subsections.

4.1 Molecular dynamics

The Structural Biology user community participating to the MoBrain Competence Centre provided several applications designed in highly parallel manner to profit of GPU architecture. Some of them, i.e. DisVis, implementing visualisation and quantification of the accessible interaction space of distance restrained binary bio-molecular complexes, and PowerFit, implementing automatic rigid body fitting of bio-molecular structures in Cryo-Electron Microscopy densities, have been already described in a recent publication [19] and will not reported here. Molecular Dynamics (MD) simulations with GROMACS and NAMD packages exploiting GPU resources were carried out by the CESNET team of MoBrain and by the MolDynGrid Virtual Laboratory [20] team, from the National Academy of Sciences of Ukraine, who have a long-standing collaboration with EGI.

MD simulations with the PMEMD tool of the AMBER package (ambermd.org) were carried out by the CIRMMP team and will be described in more detail here. CIRMMP benchmarked two applications of MD simulations that are often performed with the AMBER package:

1. Restrained MD (rMD), i.e. including experimental data that define the conformational space available to the molecule.
2. Unrestrained, also called free, MD simulations.

The main application of rMD is the optimization of 3D structural models generated by techniques such as distance geometry or molecular modelling. Optimized models can be deposited to the Protein Data Bank (PDB) [21]. Instead, free MD simulations have an extensive range of biological applications. Besides their specific research purpose, the two types of simulations differ in the input data required. For free MDs only the atomic coordinates of the initial 3D structural model of the biological system of interest are needed. For rMD simulations, the experimental data are required as an additional input, in a suitable format. rMD simulations are enabled by the AMPS-NMR portal which was developed in the context of the WeNMR project [22, 23]. For the rMD calculation a standardized simulation protocol consisting of a constant-time simulation, where the solvated protein is initially heated, then allowed to move for a predefined number of steps and then cooled again to 0 K was applied to a small protein with publicly available experimental data (PDB entry 2KT0, DOI: 10.2210/pdb2kt0/pdb).

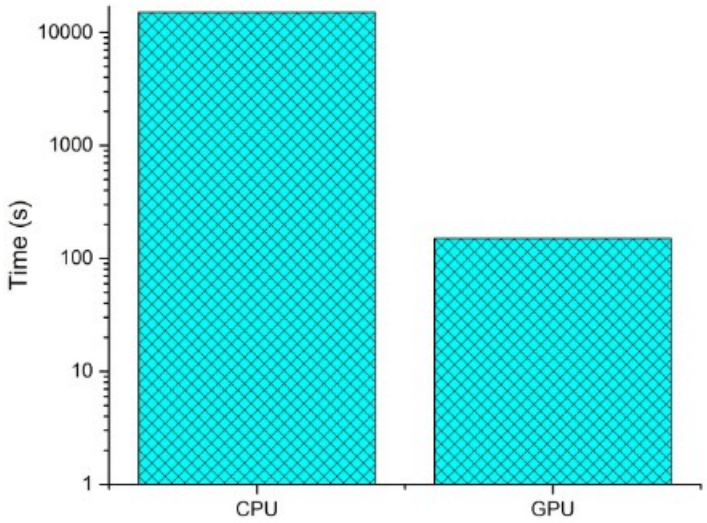

Figure 10: Calculation times for rMD simulations for PDB entry 2KT0 on a single-core CPU (AMD™ Opteron™ 6366-HE) vs one GPU card (NVIDIA™ Tesla™ K20). Note the logarithmic scale of the y axis.

The acceleration factor under this real case scenario, as shown in Figure 10, is about 100. Note that rMD simulations do not scale well with the number of cores because of the way the restraint potential is coded in the software. For the AMBER code, the CPU server does not affect the GPU performance. The agreement between the refined structural models and the experimental restraint data in the two simulations, as measured by the residual deviations of the back-calculated data with respect to the input, was essentially identical. This proves that the use of GPUs does not introduce numerical errors due to single precision. The GPU-enabled service

is already available to users via the AMPS-NMR portal. A user commented "It was very fast. About 40 minutes for 20 structures ! ". Typical run times in the traditional CPU-based ITC grid were of the order of 10 hours.

To benchmark unrestrained (free) MD simulations we used a more challenging macromolecular system than the system used for rMD, owing to its large total mass. The system of choice was the M homopolymer ferritin from bullfrog (PDB entry 4DAS, DOI: 10.2210/pdb4das/pdb). This system is composed by 24 identical protein chains, forming a single macromolecular adduct, thus with an overall protein mass of 499,008 Da (with respect to 10,051 of 2KT0). After solvation, this corresponds to ca. 180,000 atoms in the system vs 1,440, so about a factor 125 in size. The physiological role of ferritin is to store iron ions and release them when needed by the cell. Molecular systems of the size of ferritin are not amenable to routine structural studies by NMR and therefore their dynamic properties can be explored only by MD simulations. In practice, the current setup constituted the basis for an extensive investigation of the molecular mechanism of iron release, which will be published elsewhere. The acceleration afforded by the GPU in our benchmark is depicted in Figure 11.

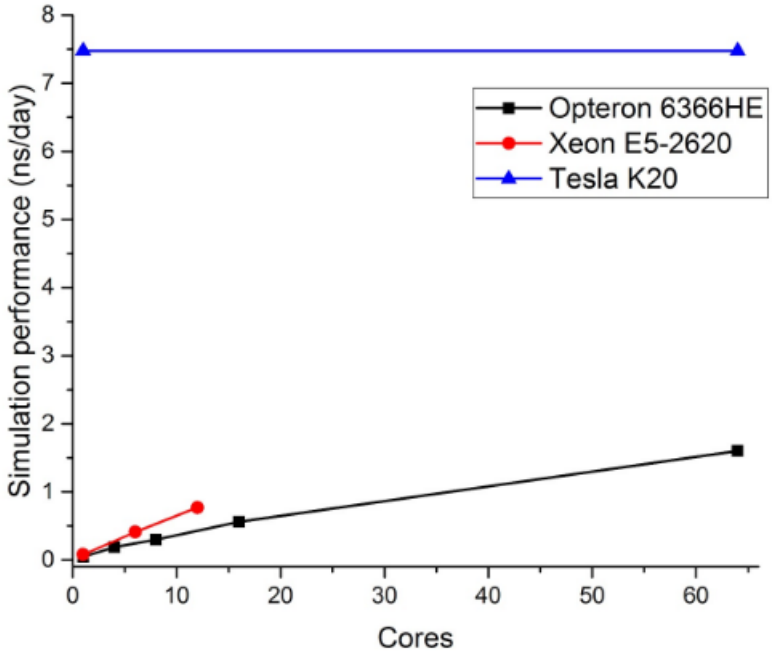

Figure 11: Extended comparison of the performance achieved using single/multi-core CPUs (Intel™ Xeon™ CPU E5-2620 and AMD™ Opteron™ 6366-HE) vs one GPU card (NVIDIA™ Tesla™ K20). The graph reports the nanoseconds of simulation that can be computed in one day as a function of the number of cores used on each CPU type.

It appears that the correlation with the number of cores is actually less than optimal on either CPU type. For the Intel™ CPU, the simulation performance (ns/day of simulation) as a function of the number of cores used is 1:5.1:9.6, against a ratio of 1:6:12 cores; for the AMD™ CPU the corresponding data are 1:11.2:32 against a ratio of 1:16:64 cores (Table 1, compare the first and third columns). Thus, using a 64-core system constituted by four AMD™ Opteron™ 6366-HE CPUs (1.8GHz, 16C, cache L2 16MB/L3 16MB, 32x4GB RDIMM LV dual rank memory) on a single blade provides about 50% of the expected increase in simulation length

that can be computed per day with respect to a single core. The acceleration provided by a single GPU card with respect to the full 64-core system is 4.7 (Table 1, last column).

Number of cores	Simulation performance in ns/day	Ratio vs. single-core	GPU acceleration
AMD™ Opteron™			
1	0.05	1.0	150.00x
4	0.18	3.6	41.70x
8	0.30	6.0	25.00x
16	0.56	11.2	13.40x
64	1.60	32.0	4.69x
Intel™ Xeon™ CPU E5-2620			
1	0.08	1.0	93.70x
6	0.41	5.1	18.30x
12	0.77	9.6	9.74x
GPU card			
1	7.50	-	-

Table 1: Benchmark results for AMBER, on unrestrained MD simulations for the ferritin system. The table reports the performance achieved using only single/multi-core CPUs (Intel™ Xeon™ CPU E5-2620 and AMD™ Opteron™ 6366-HE) vs the same cores plus one GPU card (NVIDIA™ Tesla™ K20m), measured by the nanoseconds of simulation that can be computed in one day for the various hardware configurations. The table also reports the scale factor resulting from the use of multiple cores vs. a single core as well as acceleration provided by the GPU card.

With the above configuration we could thus compute multiple 100-ns simulations (reproducing different experimental conditions) of human ferritin, in less than two weeks each. With a traditional system, each calculation would have taken about two months. The energy consumed for one ns of simulation was also measured with the iDRAC monitoring utility for the 64-core AMD™ Opteron™ system and the Intel™ Xeon™ system accelerated with the GPU. The difference between the active power and the idle power (generally known as “dynamic power”) of both systems was calculated. The dynamic power was considered because the two systems had different base configurations, like power supplies and CPUs. It turned out that the electric cost of one ns of simulation with the GPU is only the 8% of the cost when using the full 64-core AMD™ Opteron™ system.

4.2 Biodiversity

In March 2017 the European Commission granted the legal status of European Research Infrastructure Consortium (ERIC) to LifeWatch: the e-Science and Technology European Infrastructure for Biodiversity and Ecosystem Research [24]. It aims to advance biodiversity and ecosystem research and to provide major contributions to address big environmental challenges such as climate change by providing access through a pan-European distributed e-infrastructure to large sets of data, services and tools that enable the creation of virtual laboratories and decision-support applications. Within the EGI-Engage project, the LifeWatch community, organized as Competence Centre, explored the integration and deployment of assisted pattern recognition tools on EGI specific resources, including servers with GPUs on the cloud. After a state-of-the-art research on computer vision and deep learning tools, the Caffe framework was selected to build a prototype of classification service on flower species through image recognition. Caffe is a deep learning framework using Convolutional Neural Networks and can be deployed on the cloud. It provides two services: "train" and "classify". The first one is a compute intensive service allowing to train a new model. Its performance can be enhanced using GPUs supporting CUDA. The speed-up using GPU versus CPU is a factor 9 using e.g. AlexNet model, while with the most recent NVIDIA™ Kepler or Maxwell GPU architectures the use of CuDNN v3 library allows to achieve a factor 32. The second service allows to classify user provided images according to the trained model. It is a I/O bounded task that can be run efficiently without GPUs. Starting from the Caffe framework, the LifeWatch team developed a demonstrator service to help users to train a web based image classifier. It implements a two steps process: at first, the Neural Network is trained with the image datasets and tags provided by the user; then, the trained Neural Network is connected with the web service offering a API to be used with the provided web interface and android app. The user can e.g. upload a flower image from his smart-phone through the demonstrator android app, and the returned result will be a ranked list of the potential Latin names of the flower with their accuracy. The service has been tested and validated on the EGI Federated Cloud site of IISAS hosting the GPU servers, by using several datasets, including images from Portuguese flora and Real Jardin Botanico of Madrid, as described in detail in [25].

5. Conclusions and future work

The CREAM GPU-enabled prototype was tested at four sites hosting the following LRMS: TORQUE, LSF, HTCondor, Slurm, and SGE. Three new JDL attributes were defined: GPUNumber, GPUModel, MICNumber. At three sites the prototype is run in "production": QMUL and ARNES (with Slurm LRMS) and CIRMMP (with TORQUE/Maui LRMS/scheduler). A number of new classes and attributes describing accelerators were proposed and included in the GLUE2.1 draft latest version, after discussion with the OGF Working Group. A major release of CREAM is almost ready with GPU/MIC support for the LRMS listed above, with the GLUE2.1 draft prototype implemented as information system. Future official approval of GLUE 2.1 would occur after the specification is revised based on prototype lessons learned. The major release of CREAM will support CentOS 7 operating system, in order to be included in the EGI UMD-4 distribution.

For what concerns the Cloud Accelerated platform, three cloud sites hosting NVIDIA™ GPUs were put in production using PCI pass-through virtualisation with KVM hypervisor, and integrated in the EGI Federated Cloud. An experimental cloud site was set up at IISAS to enable GPU support with LXD hypervisor with OpenStack. LXD is a full container solution supported by Linux that is expected to provide better performance and stability than KVM (faster start-up time, better integration with OS), especially in terms of GPU support (simpler site setup, more stable than KVM PCI pass-through).

Acknowledgments

This work is supported by European Union's Horizon 2020 Framework Programme e-Infrastructure grants (EGI-Engage, Grant No. 654142; INDIGO-DataCloud, Grant No. 653549; West-Life Grant No. 675858). Andrea Sartirana (GRIF/LLR), Daniel Traynor (QMUL), Barbara Krasovec (ARNES) and Stefano Dal Pra (INFN-CNAF) are acknowledged for having supported the deployment of the GPU/MIC enabled CREAM CE prototype on their resource centres. Boris Parak (CESNET) is acknowledged for having implemented PCI pass-through support for GPUs on CESNET-Metacloud OpenNebula site of the EGI Federated Cloud.

References

[1] John Walsh, *Accelerated computing on computational grid infrastructures*, [thesis], Trinity College (Dublin, Ireland). School of Computer Science & Statistics, 2016, pp 187

[2] S. Andreozzi et al., *Towards GLUE 2: evolution of the computing element information model, Journal of Physics: Conference Series* **119** (2008) 062009

[3] P. Andreetto et al., *Status and Developments of the CREAM Computing Element Service, Journal of Physics: Conference Series* **331** (2011) 062024

[4] A.Casajus et al., *DIRAC Pilot Framework and the DIRAC Workload Management System, Journal of Physics: Conference Series* **219** (2010) 062049

[5] F.Giacomini et al., *The gLite Workload Management System, Journal of Physics: Conference Series* **119** (2008) 062007

[6] *CREAM JDL guide*, https://wiki.italiangrid.it/twiki/bin/view/CREAM/JdlGuide

[7] M.Mezzadri et al., *Job submission and control on a generic batch system: the BLAH experience, Journal of Physics: Conference Series* **331** (2011) 062039

[8] *Maui scheduler*, http://www.adaptivecomputing.com/products/open-source/maui/

[9] *GPU/MIC support in HTCondor*, https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=HowToManageGpus

[10] *GPU/MIC support in LSF*, https://www.ibm.com/support/knowledgecenter/it/SSETD4_9.1.2/lsf_admin/define_gpu_mic_resources.html

[11] *GPU/MIC support in Slurm*, https://slurm.schedmd.com/gres.html

[12] D. Salomoni et al., I*NDIGO-Datacloud: foundations and architectural description of a Platform as a Service oriented to scientific computing*, arXiv:1603.09536v3 [cs.SE]

[13] S. Andreozzi, S. Burke, L. Field, G. Galang, B. Konya, M. Litmaath, P. Millar, J. P. Navarro, *GLUE Specification Version 2.0, OGF Document Series*, GFD.147, 2009

[14] L Field and M W Schulz, *Grid deployment experiences: The path to a production quality ldap based grid information system*, Proc. *Int. Conf. on Computing in High Energy and Nuclear Physics* (2004)

[15] Ming Jiang et al., *An APEL Tool Based CPU Usage Accounting Infrastructure for Large Scale Computing Grids, Data Driven e-Science*, Springer New York, 2011, pp. 175–186

[16] *S*apan Gupta, Deepanshu Gera, *A Comparison of LXD, Docker and Virtual Machine, International Journal of Scientific & Engineering Research,* Volume 7, Issue 9, September-2016

[17] *EGI AppDB*, https://appdb.egi.eu/

[18] *Accelerated computing Virtual Organization*, https://wiki.egi.eu/wiki/Accelerated_computing_VO

[19] G.C.P. van Zundert et al., *The DisVis and PowerFit Web Servers: Explorative and Integrative Modeling of Biomolecular Complexes, Journal of Molecular Biology*, 2017, **429** (3) pp. 399-407

[20] *MolDynGrid Virtual Laboratory*, http://moldyngrid.org

[21] Berman H, Henrick K, Nakamura H, Markley JL, *The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data, Nucleic Acids Res.* 2007 Jan, **35**(Database issue):D301-3

[22] Wassenaar et al., *WeNMR: Structural Biology on the Grid, J. Grid. Comp.,* 2012, **10**:743-767

[23] Bertini I, Case DA, Ferella L, Giachetti A, Rosato A, *A Grid-enabled web portal for NMR structure refinement with AMBER, Bioinformatics* 2011 Sep 1, **27**(17):2384-90

[24] A. Basset, W. Los, *Biodiversity e-Science: LifeWatch, the European infrastructure on biodiversity and ecosystem research*, *Plant Biosystems - An International Journal Dealing with all Aspects of Plant Biology*, 2012, **146**:4, 780-782

[25] Eduardo Lostal, Francisco Sanz, Jesus Marco, *Integration of assisted pattern recognition tools*, https://documents.egi.eu/document/2647