EGI federated platforms supporting accelerated computing

Marco Verlato
INFN-Padova

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

EGI-Engage is co-funded by the Horizon 2020 Framework Programme
of the European Union under grant number 654142

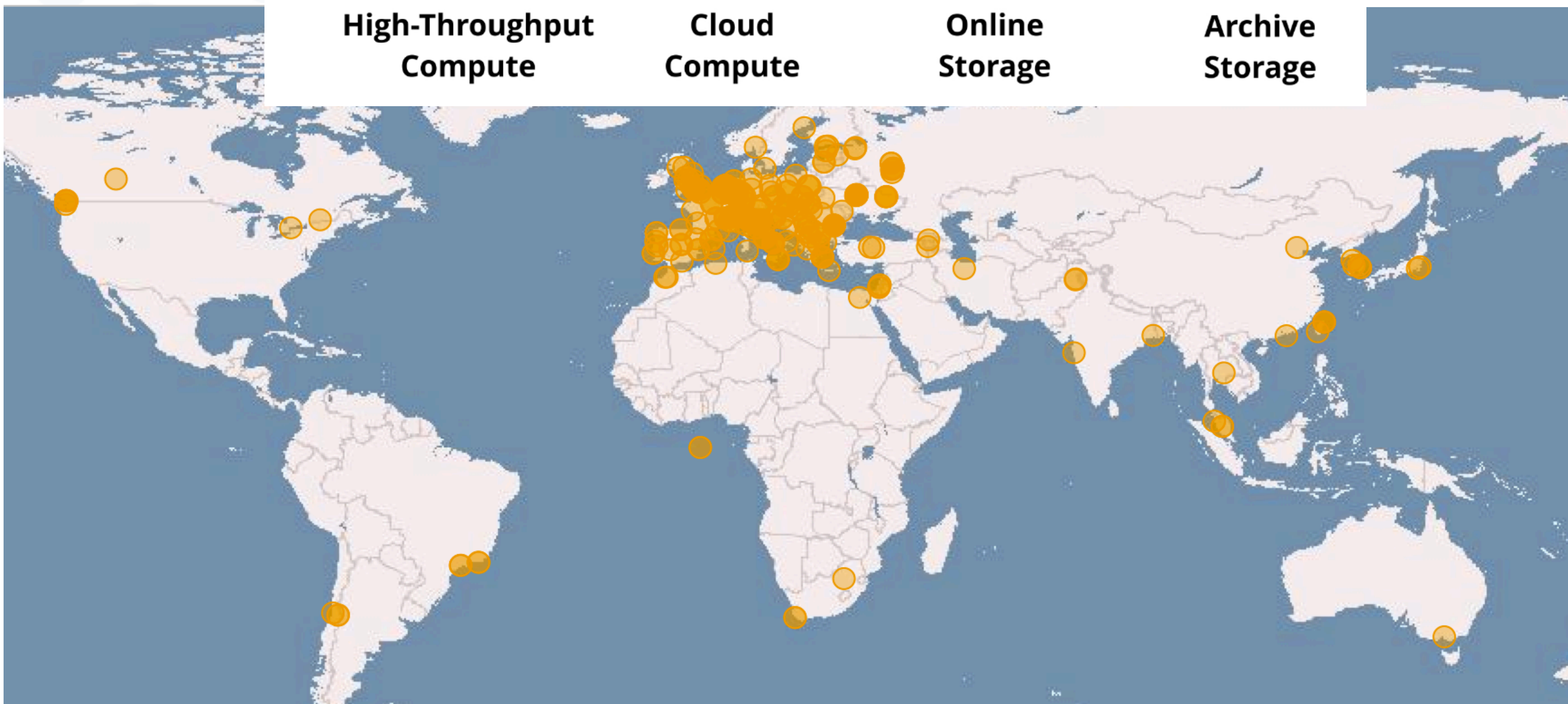
Layout

- Introduction
- HTC accelerated platform
- Cloud accelerated platform
- Applications use-cases

EGI federated infrastructure, Nov. 2016

The federation in numbers

730,000 cores	6,600 cores	285 PB	280 PB
High-Throughput Compute	Cloud Compute	Online Storage	Archive Storage



Introduction/1

- EGI infrastructure supported through H2020 project EGI-Engage, from March 2015 until August 2017 → new EU projects are in preparation
  - Dedicated task for "**Providing a new accelerated computing platform**"
- Accelerated computing:
  - **GPGPU** (General-Purpose computing on Graphical Processing Units)
    - NVIDIA GPU/Tesla/GRID, AMD Radeon/FirePro, Intel HD Graphics,...
  - Intel Many Integrated Core (**MIC**) Architecture
    - Xeon Phi Coprocessor
  - Specialized **PCIe cards** with accelerators
    - DSP (Digital Signal Processors)
    - FPGA (Field Programmable Gate Array)

Introduction/2

- Main goals:
  - To implement the support in the information system
    - both software and hardware info at site level must be published/discoverable
    - **OGF GLUE standard** based information system structure must be extended
  - To extend the HTC and Cloud middleware support for co-processors
    - to provide a transparent and uniform way to allocate these resources together with CPU cores efficiently to the users
- Requirements and use-cases from user communities were collected at various EGI events:
  - EGI Conference 2015: http://bit.ly/Lisbon-GPU-Session
  - EGI Community Forum 2015: http://bit.ly/Bari-GPU-Session
  - EGI Conference 2016: http://bit.ly/Amsterdam-GPU-Session

EGI-ENGAGE

Introduction/3

- Activity driven by the user communities
- Grouped in EGI-Engage as Competence Centers:
  - **LifeWatch**: to capture and address the requirements of Biodiversity and Ecosystems research communities
    - Deploy GPU based e-Infrastructure services supporting data management, processing and modelling for Ecological Observatories
      - **IC-DLT**: Image Classification Deep Learning Tool
  - **MoBrain**: to Serve Translational Research from Molecule to Brain
    - Deploy portals for biomolecular simulations leveraging GPU resources
      - **AMBER** and **GROMACS** Molecular Dynamics packages
      - **PowerFit**: exhaustive search in Cryo-EM density
      - **DisVis**: visualisation and quantification of the accessible interaction space of distance restrained binary biomolecular complexes, determined for example by using CXMS technique
    - Linked with several older and new EU projects involving the Bio-NMR community

e-nmr → we-nmr → MoBrain West-Life

2007 2010 2013 2015 2018

Introduction/4

- Some requirements from applications:
  - Need of GPU resources for development and testing
  - One job per GPU (AMBER)
  - CPUs must be powerful to match the GPU
    - CPU is still doing some work (e.g. bonded interactions)
  - Discoverable within the e-infrastructure (e.g. JDL requirement)
    - Preferably containing GPU type (GTX vs K-series, AMD vs NVIDIA)
    - AMD GPUs not supported by MD code (yet)
    - Double-precision only supported by Tesla cards
  - GPU Cloud solution, if used, should allow for transparent and automated submission
  - Software and compiler support on sites providing GPU resources (CUDA, OpenCL)

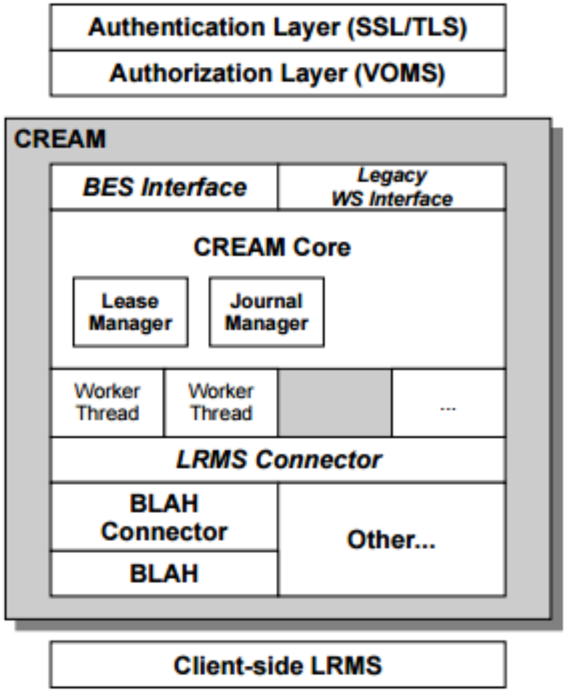
HTC Accelerated Platform

Paolo Andreetto (INFN-PD)
David Rebatto (INFN-MI)
Marco Verlato (INFN-PD)
Lisa Zangrando (INFN-PD)
Antonio Rosato (CIRMMP)
Andrea Giachetti (CIRMMP)

EGI-Engage is co-funded by the Horizon 2020 Framework Programme
of the European Union under grant number 654142

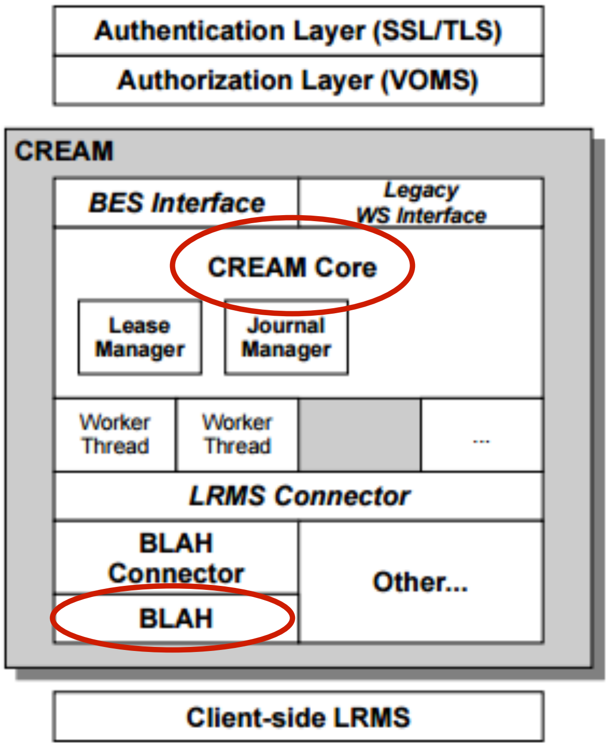
CREAM-CE

- Starting from previous work of EGI Virtual Team (2012) and GPGPU Working Group (2013-2014)
- **CREAM-CE** is the most popular grid interface (Computing Element) to a number of LRMSes (Torque, LSF, Slurm, SGE, HTCondor) since many years in EGI
- Most recent versions of these LRMSes do support natively GPUs (and MIC cards), i.e. servers hosting these cards can be selected by specifying LRMS directives
- CREAM must be enabled to publish this information and support these directives



Work plan

- Indentifying the relevant GPU/MIC related parameters supported by the different LRMSes, and abstract them to significant JDL attributes
- Implementing the needed changes in CREAM Core and and BLAH components
- Extending the GLUE 2.1 schema draft with accelerator information
- Writing the info-providers according to extended GLUE 2.1 draft specifications
- Testing and certification of the prototype
- Releasing a CREAM update with full GPU/MIC support



Implementing job submission/1

- **Testbed setup at CIRMMP**
  - 3 nodes 2x Intel Xeon E5-2620v2
  - 2 NVIDIA Tesla K20m GPUs per node
  - Torque 4.2.10 (source compiled with NVML libs) + Maui 3.3.1
  - AMBER application installed with CUDA

- **First step**:
  - Starting by testing local job submission with the different GPGPU supported options, e.g. with Torque/pbs_sched:

```
$ qsub -l nodes=1:gpus=1 job.sh
$ qsub -l nodes=1:gpus=1 job.sh
```

  - …and with Torque/Maui:

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

Implementing job submission/2

- **Second step**:
  - defining the new JDL attribute **GPUNumber**
  - implementing it in CREAM Core and BLAH components
  - the first GPGPU-enabled CREAM prototype working on top of the CIRMMP Torque/Maui cluster was implemented in December 2015
- **Third step**:
  - Looking at GPU and MIC supported options for the HTCondor, LSF, Slurm and SGE
  - Two additional JDL useful attributes identified and implemented:
    - **GPUModel**: for selecting the servers with a given model of GPU card
      - e.g. GPUModel="teslaK80"
    - **MICNumber**: for selecting the servers with the given number of MIC cards

Implementing job submission/3

- A **CREAM/HTCondor** prototype supporting both GPUs and MIC cards was successfully implemented and tested at **GRIF/LLR** data centre in March 2016 (thanks to A. Sartirana)

- A **CREAM/SGE** prototype supporting GPUs was successfully implemented and tested at **Queen Mary** data centre in April 2016 (thanks to D. Traynor)

- A **CREAM/Slurm** prototype supporting GPUs was successfully implemented and tested at **ARNES** data centre in April 2016 (thanks to B. Krasovec)

- A **CREAM/LSF** prototype supporting GPUs was successfully implemented and tested at **INFN-CNAF** data centre in July 2016 (thanks to S. Dal Pra)

- A **CREAM/Slurm** prototype supporting GPUs was successfully implemented and tested at **Queen Mary** data centre in August 2016 (thanks again to D. Traynor)
  - With Slurm Version 16.05 which supports the GPUModel specification

Example of submission to Slurm CE

- User job JDL:

```
[
  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=2;
  GPUModel="teslaK80";
]
```

- Definitions in Slurm gres.conf and slurm.conf configuration files:

```
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…
```

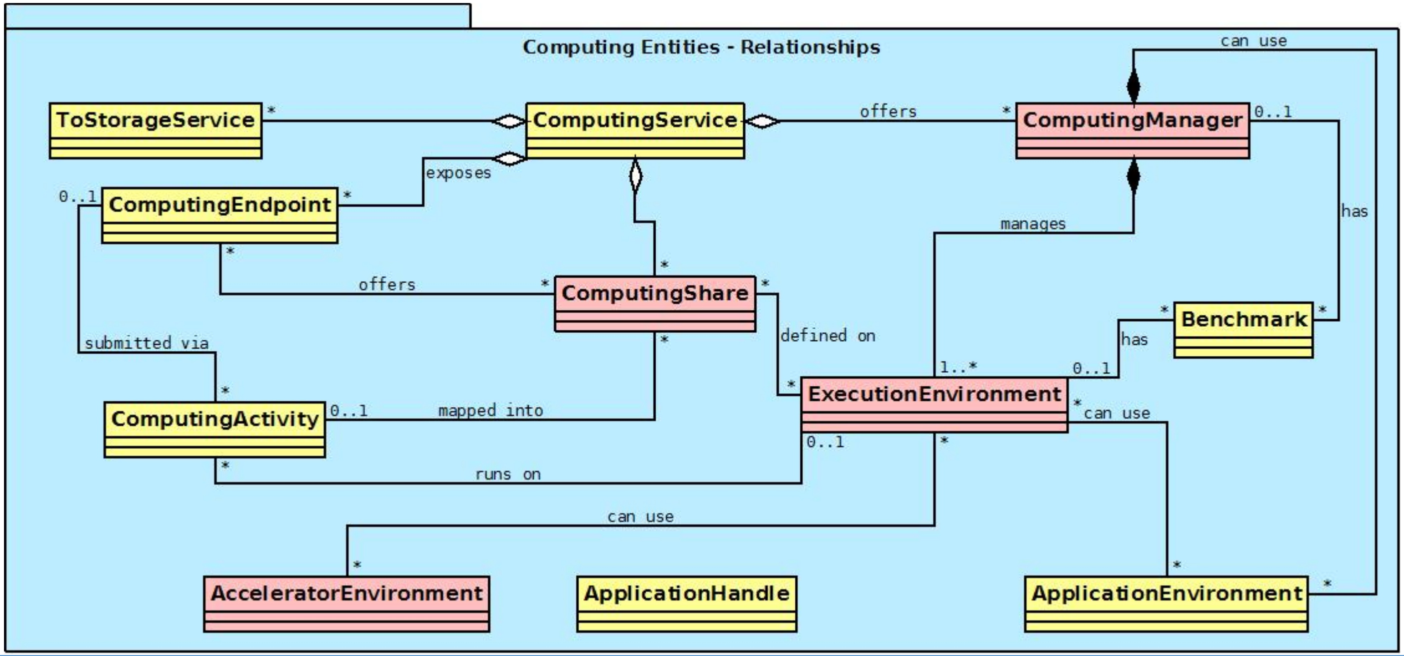
- On the worker node:

```
$ lspci | grep NVIDIA
0a:00.0 3D controller: NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
0b:00.0 3D controller: NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
86:00.0 3D controller: NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
87:00.0 3D controller: NVIDIA Corporation GK210GL [Tesla K80] (rev a1)

$ echo $CUDA_VISIBLE_DEVICES
0,1
```

Info system: GLUE2.1 Draft

- **ExecutionEnvironment** class: represents a set of homogeneous WNs
  - Is usually defined statically during the deployment of the service
  - These WNs however can host different types/models of accelerators
- **AcceleratorEnvironment** class: represents a set of homogeneous accelerator devices
  - Can be associated to one or more Execution Environments
- **New attributes:**
  - PhysicalAccelerators
  - Vendor
  - Type
  - Model
  - Memory
  - ClockSpeed
- **Driver info are in the Application Environment**



Info system: static info

- Example of GLUE2.1 static info publication:

```
$ ldapsearch -x -LLL -h cegpu.cerm.unifi.it -p 2170 -b o=glue (objectClass=GLUE2AcceleratorEnvironment)
GLUE2AcceleratorEnvironmentMemory: 5120
GLUE2AcceleratorEnvironmentID: tesla.cegpu.cerm.unifi.it
GLUE2AcceleratorEnvironmentModel: Tesla K20m
objectClass: GLUE2Entity
objectClass: GLUE2AcceleratorEnvironment
GLUE2EntityCreationTime: 2015-05-04T16:31:18Z
GLUE2AcceleratorEnvironmentExecutionEnvironmentForeignKey: cegpu.cerm.unifi.it
GLUE2AcceleratorEnvironmentVendor: NVIDIA
GLUE2AcceleratorEnvironmentPhysicalAccelerators: 2
GLUE2AcceleratorEnvironmentType: GPU
GLUE2EntityName: tesla.cegpu.cerm.unifi.it
GLUE2AcceleratorEnvironmentLogicalAccelerators: 2
GLUE2AcceleratorEnvironmentClockSpeed: 706
```

```
$ ldapsearch -x -LLL -h cegpu.cerm.unifi.it -p 2170 -b o=glue
(&(objectClass=GLUE2ExecutionEnvironment)
(GLUE2EntityName=cegpu.cerm.unifi.it))
GLUE2ExecutionEnvironmentCPUModel: Xeon
[…]
GLUE2ExecutionEnvironmentAcceleratorEnvironmentForeignKey:
tesla.cegpu.cerm.unifi.it
GLUE2ExecutionEnvironmentApplicationEnvironmentForeignKey: nvidia-driver
```

```
$ ldapsearch -x -LLL -h cegpu.cerm.unifi.it -p 2170 -b o=glue
(&(objectClass=GLUE2ApplicationEnvironment)(GLUE2EntityName=nvidia-driver))
GLUE2ApplicationEnvironmentAppName: nvidia-driver
GLUE2ApplicationEnvironmentDescription: NVidia driver for CUDA
GLUE2ApplicationEnvironmentExecutionEnvironmentForeignKey: cegpu.cerm.unifi.it
GLUE2ApplicationEnvironmentID: nvidia-driver
GLUE2ApplicationEnvironmentAppVersion: 352.93
GLUE2EntityCreationTime: 2015-05-04T16:31:18Z
GLUE2ApplicationEnvironmentComputingManagerForeignKey: cegpu.cerm.unifi.it_ComputingElement_Manager
GLUE2EntityName: nvidia-driver
```

Info system: dynamic info

- For dynamic info-providers, new attributes in GLUE2.1 draft for existing class were defined:
  - **ComputingManager** class (the LRMS)
    - TotalPhysicalAccelerators, TotalAcceleratorSlots, UsedAcceleratorSlots
  - **ComputingShare** class (the batch queue)
    - MaxAcceleratorSlotsPerJob, FreeAcceleratorSlots, UsedAcceleratorSlots

```
$ ldapsearch -x -h cegpu.cerm.unifi.it -p 2170 -b o=glue
objectClass=GLUE2ComputingShare
[…]
GLUE2EntityOtherInfo: CREAMCEId=cegpu.cerm.unifi.it:8443/cream-pbs-batch
GLUE2ComputingShareMaxAcceleratorSlotsPerJob: GPU:4
GLUE2ComputingShareUsedAcceleratorSlots: GPU:1
GLUE2ComputingShareFreeAcceleratorSlots: GPU:3
[…]
```

EGI-ENGAGE

Accounting

- CREAM Accounting sensors, mainly relying on LRMS logs, were in the past developed by the APEL team
- APEL team has been involved in the GPU accounting discussion
- Batch systems should report GPU usage attributable to the job in the batch logs. APEL would then parse the logs files to retrieve the data.
- Unfortunately job accounting records of Torque, LSF and other LRMSes do not contain GPU usage info ☹
- NVML allows to enable per-process accounting of GPU usage using Linux PID, but not LRMS integration yet, e.g.:

```
$ 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
```

EGI-ENGAGE

Summary

- The CREAM GPU-enabled prototype was tested at 5 sites
  - LRMSes: Torque, LSF, HTCondor, Slurm, and SGE LRMSes
  - 3 new JDL attributes defined: GPUNumber, GPUModel, MICNumber
- At 3 sites the prototype is run in “production”: QMUL and ARNES (Slurm) and CIRMMP (Torque/Maui)
- New classes and attributes describing accelerators proposed and included in GLUE2.1 draft after discussion with the OGF WG
- A **major release of CREAM** is almost ready
  - with GPU/MIC support for most LRMSes
  - with the GLUE2.1 draft prototype as information system
    - future official approval of GLUE 2.1 would occur after the specification is revised based on prototype lessons learned
  - on CentOS7, in order to be included in UMD-4 release

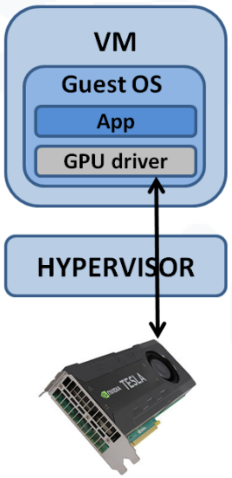
Cloud accelerated platform

Viet Tran (IISAS)
Jan Astalos (IISAS)
Miroslav Dobrucky (IISAS)

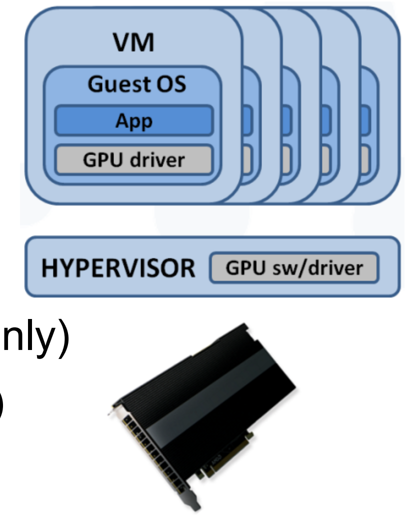
EGI-Engage is co-funded by the Horizon 2020 Framework Programme
of the European Union under grant number 654142

Accelerated computing in Clouds

- Virtualization technologies



  – KVM with passthrough is rather mature
    - But maximum 1 VM attached to 1 physical card
  – Virtualized GPU is in a early stage:
    - NVIDIA GRID vGPU (XenServer, VMWare hyperv. only)
    - SR-IOV based AMD MxGPU (VMWare hyperv. only)
    - Intel GVT-G recently added to Linux 4.10 kernel



- Cloud framework support
  – Openstack support for PCI passthrough
  – OpenNebula support for PCI passthrough from v4.14
- EGI Federated Cloud services support
  – Information system
  – Accounting

IISAS-GPUCloud site

- First test-bed set up at IISAS
- Hardware:
  - 4 x IBM dx360 M4 servers with 2x Intel Xeon E5-2650v2
  - 16 CPU cores, 64GB RAM, 1 TB storage on each WN
  - 2x NVIDIA Tesla K20m on each WN
- Software
  - Base OS: Ubuntu 14.04 LTS
  - KVM hypervisor with PCI passthrough virtualisation of GPU cards
  - OpenStack Kilo middleware
  - Newest Federated Cloud tools

Testing, optimization, troubleshooting

- Default setting is not suitable for production
  - Low performance
  - Random crashing
- Extensive testing, optimization and troubleshooting has been carried out behind the scenes:
  - Tuning BIOS setting (hardware dependent):
    VM can interact directly with hardware, e.g. sending NMI (Non-maskable interrupt) to BIOS caused system crashing. Setting BIOS to tolerate/immune to events from devices. Typical case: loading nouveau in VM cause system reboot
  - Disabling CPU hyperthreading
  - Setting correct CPU type in nova.conf:
    most safely cpu_mode = host-passthrough

EGI Federated Cloud integration

- Results
  - Fully working OpenStack based cloud site with GPGPUs
  - VMs reach native performance (around 2% differences)
  - Exact, repeatable crashing scenarios and workarounds
- **OpenStack/Kilo** site fully certified and integrated with EGI Federated Cloud in **October 2015**:
  - Pre-defined images with NVIDIA drivers and CUDA toolkit installed
  - GPU-enabled flavors: *gpu1cpu6* (1GPU + 6 CPU cores), *gpu2cpu12* (2GPU +12 CPU cores)
  - Supported VOs: fedcloud.egi.eu, ops, dteam, moldyngrid, enmr.eu, vo.lifewatch.eu, acc-comp.egi.eu

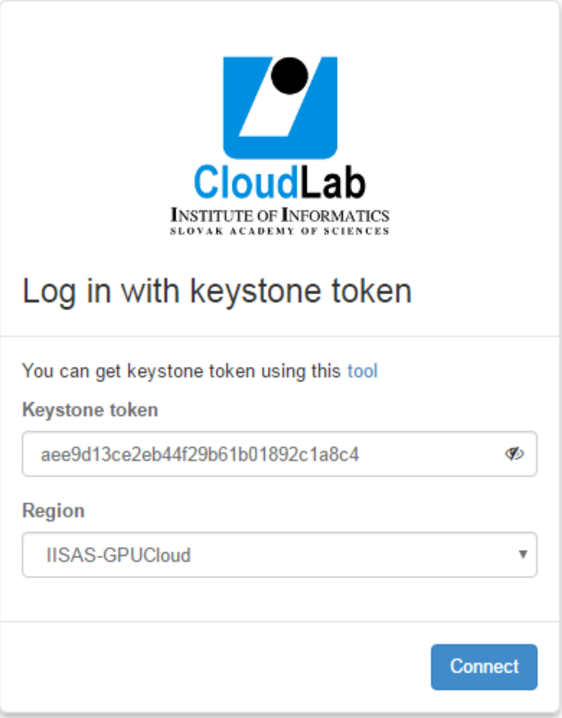
User and admin support

- User tutorial:
  - How to use GPUs on IISAS-GPUCloud
    - Access via rOCCI client
    - Access via OpenStack dashboard with token
  - How to create your own GPU server in cloud
- Site admin guide
  - How to enable GPU passthrough in OpenStack
- Additional tools
  - Automation via scripts:
    - NVIDIA + CUDA installer
    - Keystone-VOMS client for getting token
  - Keystone-voms module for Openstack Horizon
- All this in a wiki:

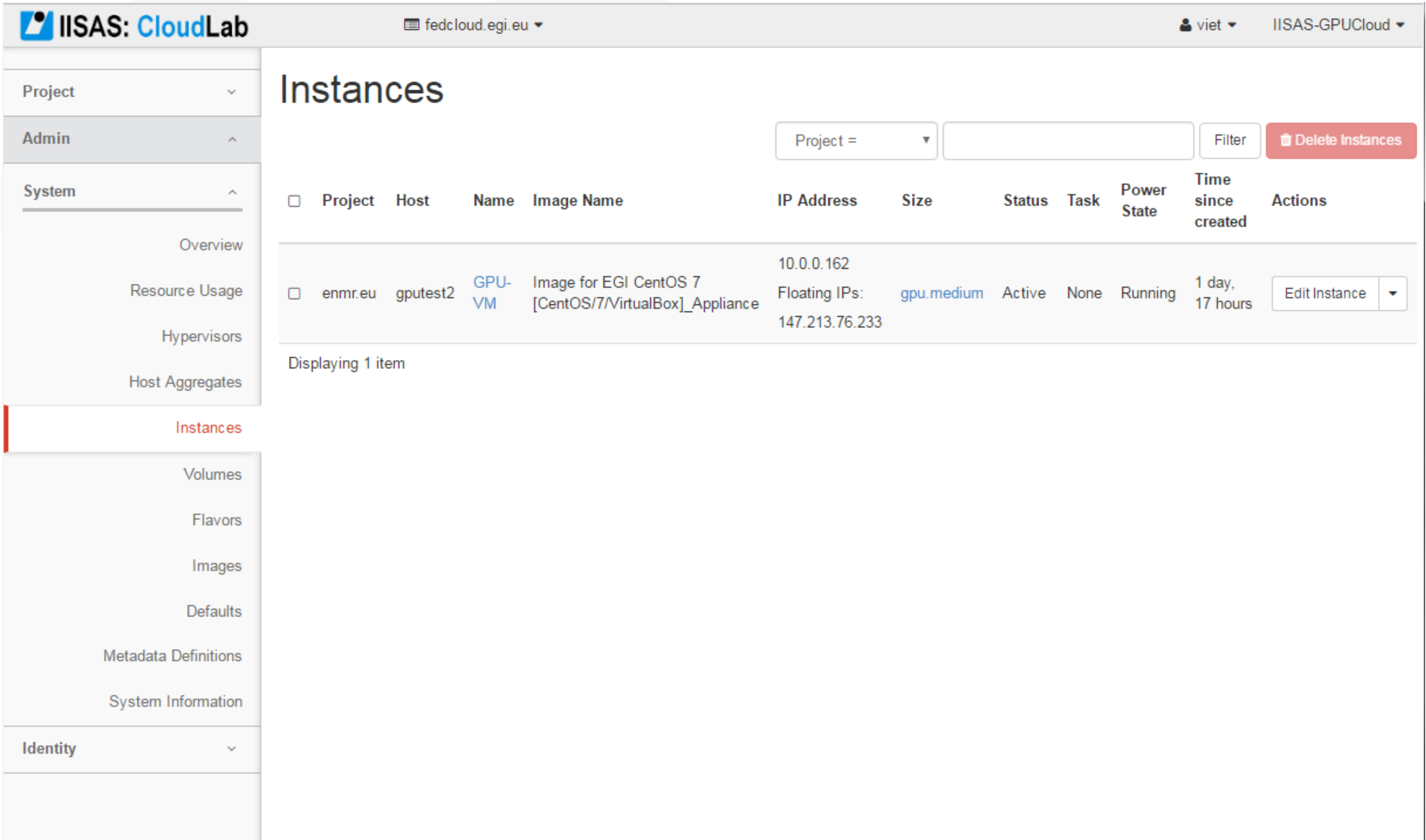
  https://wiki.egi.eu/wiki/GPGPU-FedCloud

Using IISAS-GPUCloud site

- Via rOCCI client
  - Simply choose GPU-enabled flavor (e.g. gpu2cpu12) as resource template
- Or via Openstack Horizon portal
  - Graphical interface
  - Adding support for EGI users to login via token (no username/ password)



IISAS-GPUCloud portal



EGI-ENGAGE

Docker support

- Dockers with GPUs can be executed at IISAS-GPUCloud site
  - Create a VM with GPU-enabled flavor and image
  - Run docker with proper mapping to access GPU

```
docker run --name=XXXXXX \
        --device=/dev/nvidia0:/dev/nvidia0 \
        --device=/dev/nvidia1:/dev/nvidia1 \
        --device=/dev/nvidiactl:/dev/nvidiactl \
        --device=/dev/nvidia-uvm:/dev/nvidia-uvm \
        …..
```

GPU support in OpenNebula

- A multi-purpose PCI passthrough capabilities were introduced in OpenNebula version 4.14
- CESNET-MetaCloud site upgraded in **May 2016** with 4 NVIDIA Tesla M2090 cards available (experimental set up, GPU properties in os_template)
- New IISAS-Nebula site with OpenNebula 5.0 set up and certified for production in EGI FedCloud in **January 2017**
  - rOCCI server upgraded for adding GPU properties to resource template
- Plans to provide templates and user guides for GPU-enabled virtual machines (as done for IISAS-GPUCloud)
- The long-term goal is to provide OCCI extensions to select these "additional" capabilities for virtual machines on a case-by-case basis (not just by using a pre-defined template)

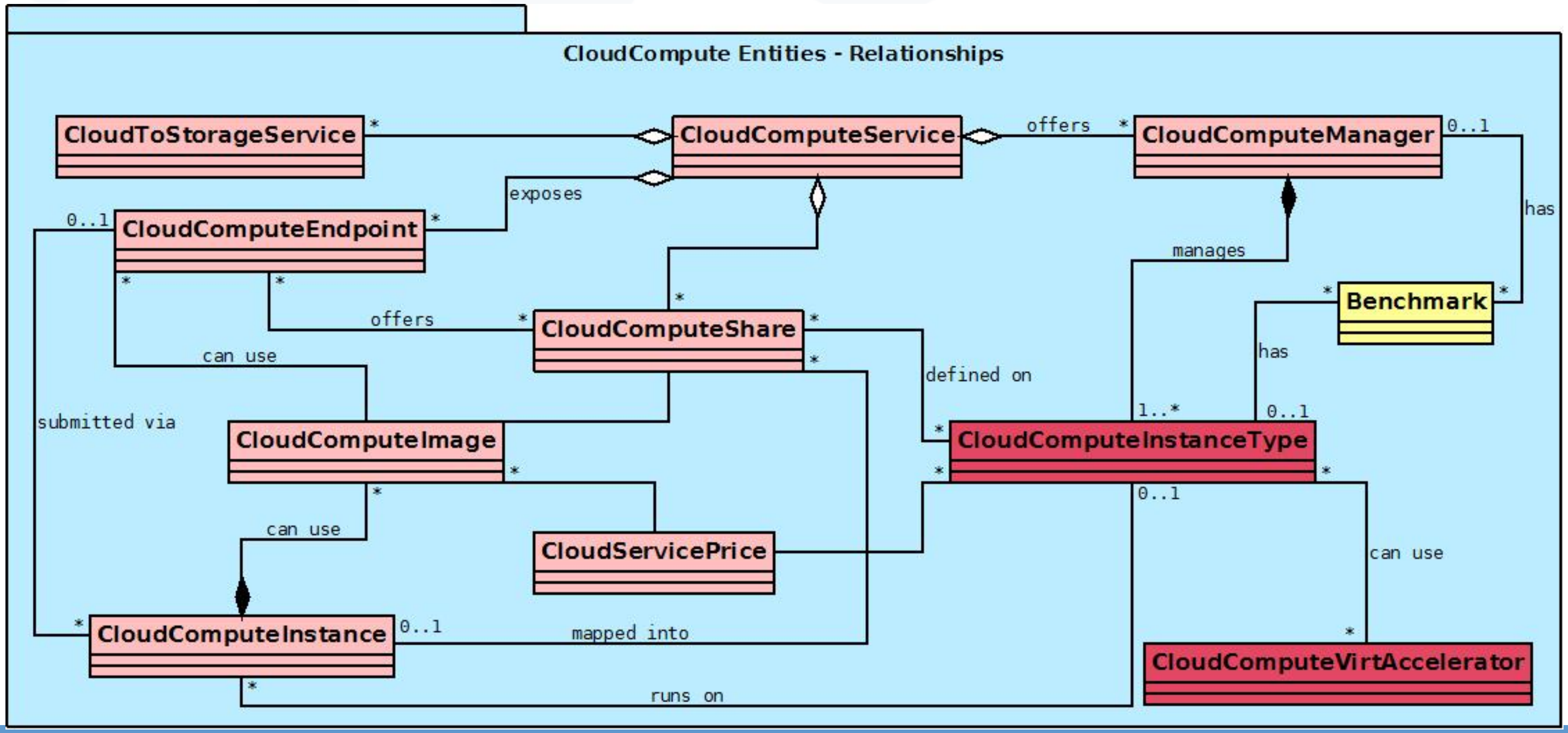
EGI-ENGAGE

GPU specific VO

- acc-comp.egi.eu VO has been established for testing and development with GPU:
  - VO image list with preinstalled GPU drivers and CUDA libraries are available via AppDB
  - Supported only at sites with GPU hardware
  - More info at https://wiki.egi.eu/wiki/Accelerated_computing_VO

Information System

- **Conceptual Model of the Cloud Computing Service is being defined in GLUE2.1 draft**
  - **CloudComputeInstanceType** class describes the hardware environment of the VM (i.e. the flavour)
  - **CloudComputingVirtualAccelerator** entity defined to describe a set of homogeneous virtual accelerator devices, who can be associated to one or more CloudComputeInstanceTypes



Accounting

- GPU accounting easier in cloud environment
  - 1 VM can be attached to 1 or more GPUs
  - Cloud systems currently return wallclock time only
  - 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 only
  - APEL team involved to define an extended usage record and new views to display GPU usage in the Accounting Portal

Next steps

- Experimental cloud site set up at IISAS to enable GPU support with LXC/LXD hypervisor with OpenStack
  - LXC/LXD is a full container solution supported by Linux
  - Expected to provide better performance and stability than KVM (must faster startup time, better integration with OS), especially in terms of GPU support (simpler site setup, more stable than KVM PCI passthrough)
- More info:
  - https://wiki.egi.eu/wiki/GPGPU-FedCloud
  - https://wiki.egi.eu/wiki/Accelerated_computing_VO
  - https://accelerated.ui.sav.sk/?page_id=21
  - https://horizon.ui.savba.sk/

Applications

Antonio Rosato (CIRMMP)
Andra Giachetti (CIRMMP)
Alexandre Bonvin (Univ. of Utrecht)
Zeynep Kurkcuoglu (Univ. of Utrecht)
Jörg Schaarschmidt (Univ. of Utrecht)
Mikael Trellet (Univ. of Utrecht)
Ales Krenek (CESNET)
Mario David (LIP)
Jesus Marco (IFCA-CSIC)
Fernando Aguilar (IFCA-CSIC)
Andrii Salnikov (KNU)
Oleksandr Savytskyi (IMBG of NASU)

EGI-Engage is co-funded by the Horizon 2020 Framework Programme
of the European Union under grant number 654142

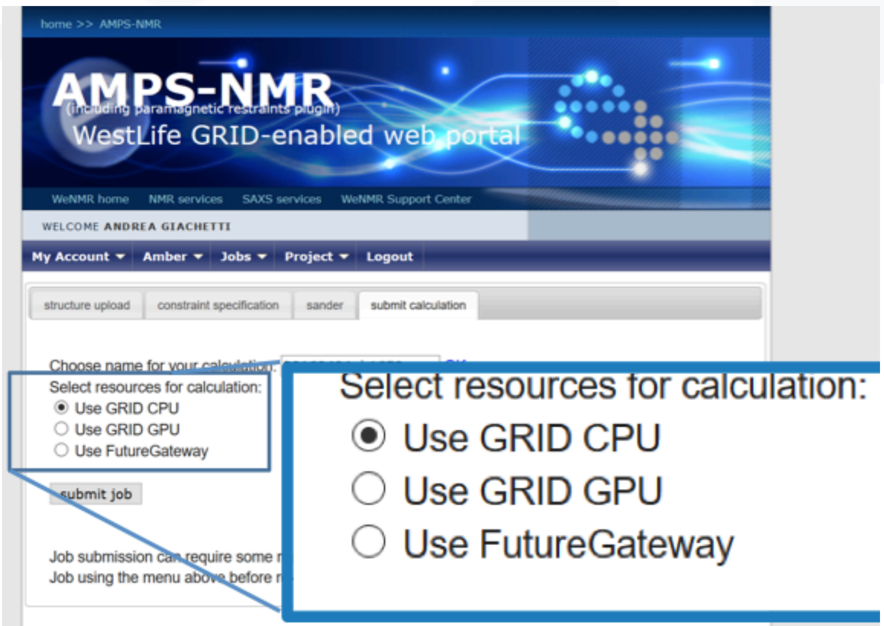
Molecular Dynamics

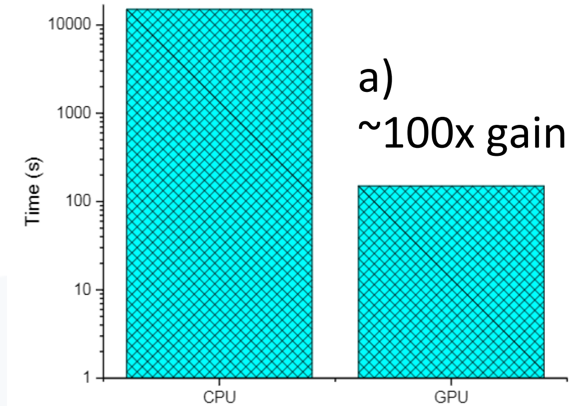
- MD simulations with **AMBER**
- MD simulations with **GROMACS**
- **MolDynGrid** Virtual Laboratory (National Academy of Sciences of Ukraine)
- **DisVis**: visualisation and quantification of the accessible interaction space of distance restrained binary biomolecular complexes
- **PowerFit**: automatic rigid body fitting of biomolecular structures in Cryo-Electron Microscopy densities

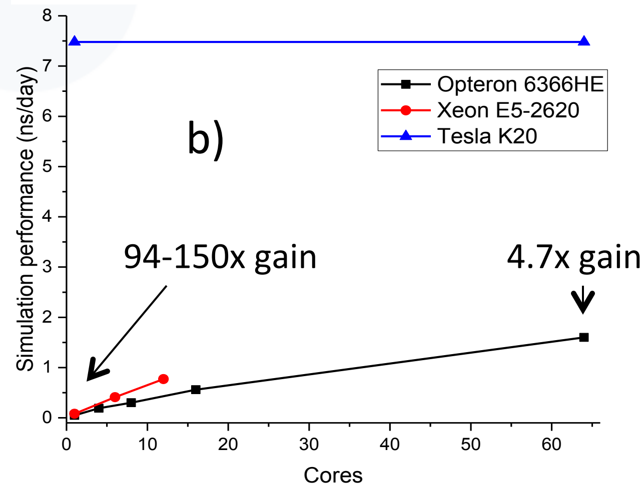
AMBER

a) Restrained (rMD) Energy Minimization on NMR Structures

b) Free MD simulations of ferritin



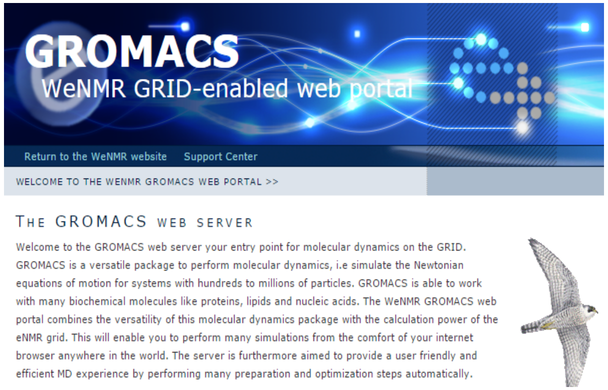


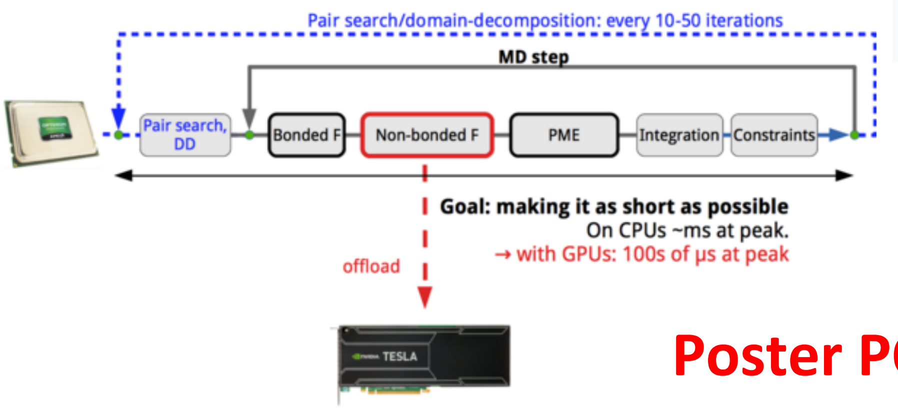


Talk at Biomedicine & Life Science II session today at 16:00

GROMACS

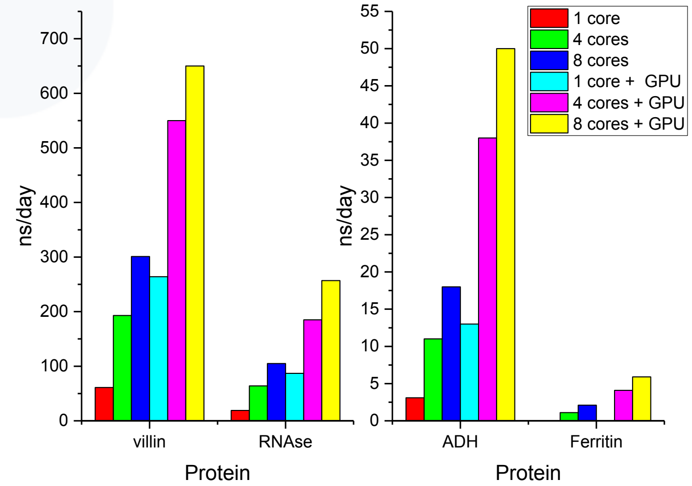
- GPU acceleration introduced in version v4.5
  – Grid portal runs it in multi-threading mode
  – No significant cloud overhead measured for GPU speedups





Poster PO-02

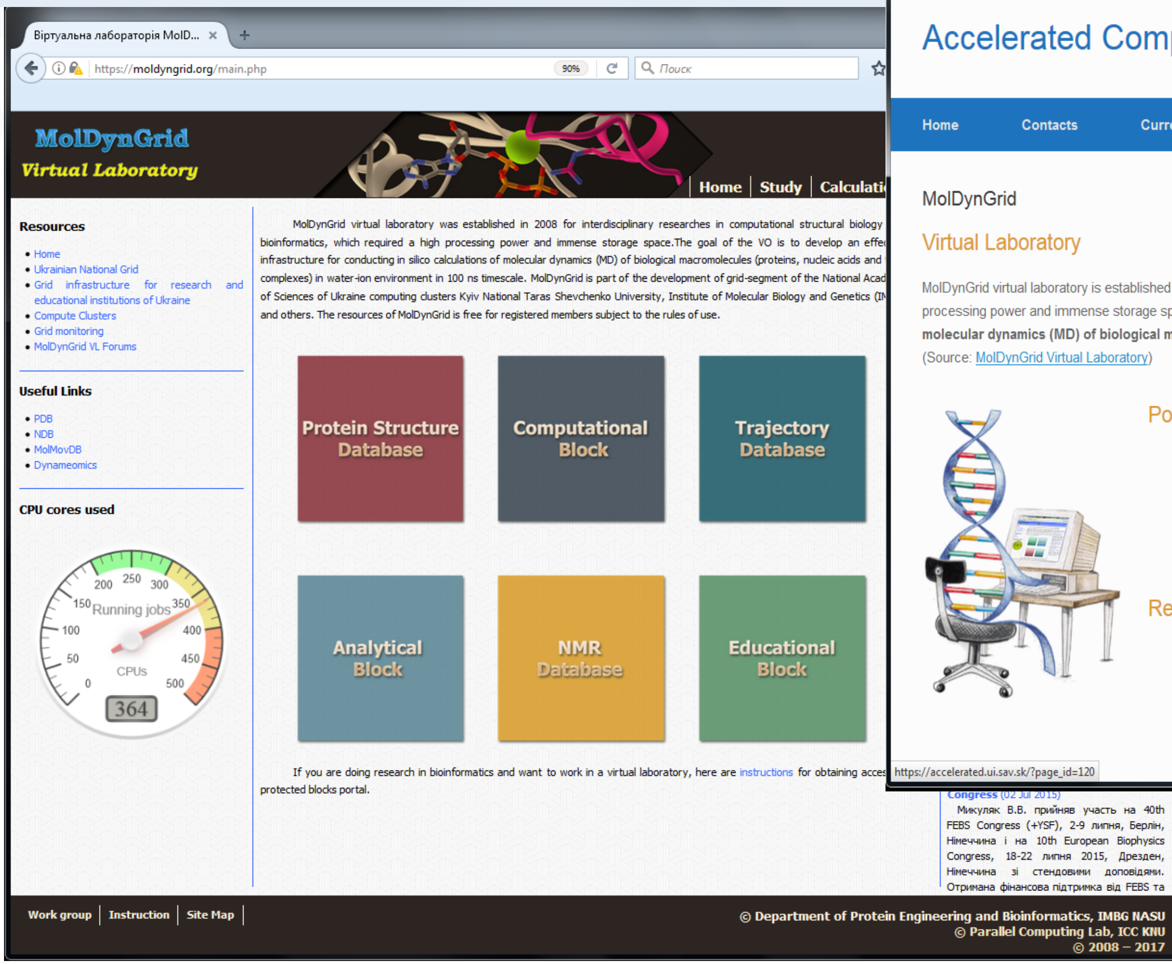
		Simulation performance in ns/day						GPU Acceleration		
Dataset	Protein size (aa)	1 core	4 cores	8 cores	1 core + GPU	4 cores + GPU	8 cores + GPU	1 core	4 cores	8 cores
Villin	35	61	193	301	264	550	650	4.3x	2.8x	2.2x
RNAse	126	19	64	105	87	185	257	4.6x	2.9x	2.4x
ADH	1,408	3.1	11	18	13	38	50	4.2x	3.5x	2.8x
Ferritin	4,200	-	1.1	2.1	-	4.1	5.9	-	3.7x	2.8x

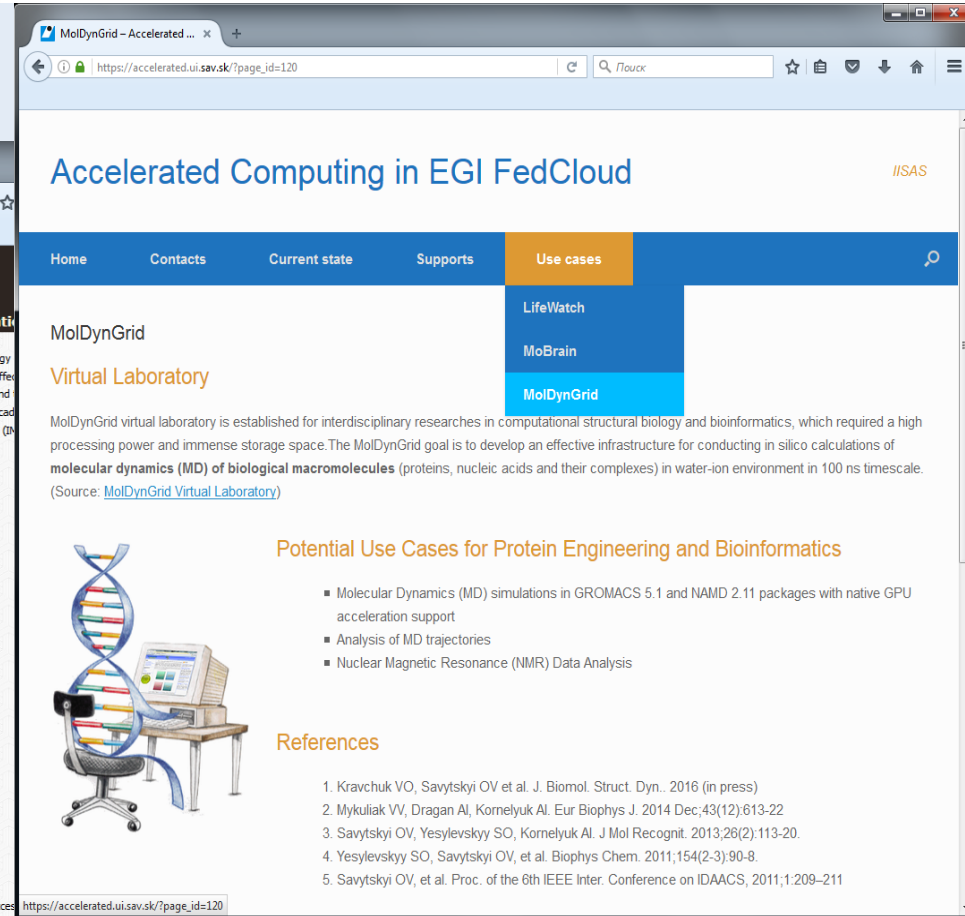


EGI-ENGAGE

MolDynGrid Virtual Laboratory

https://accelerated.ui.sav.sk/?page_id=120
http://moldyngrid.org

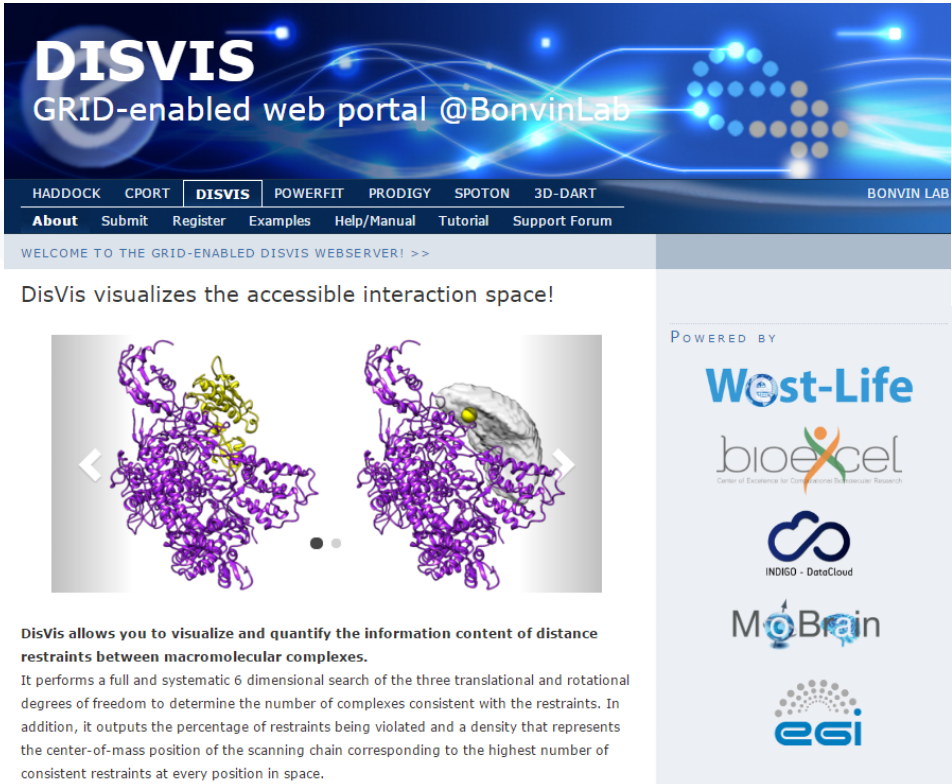




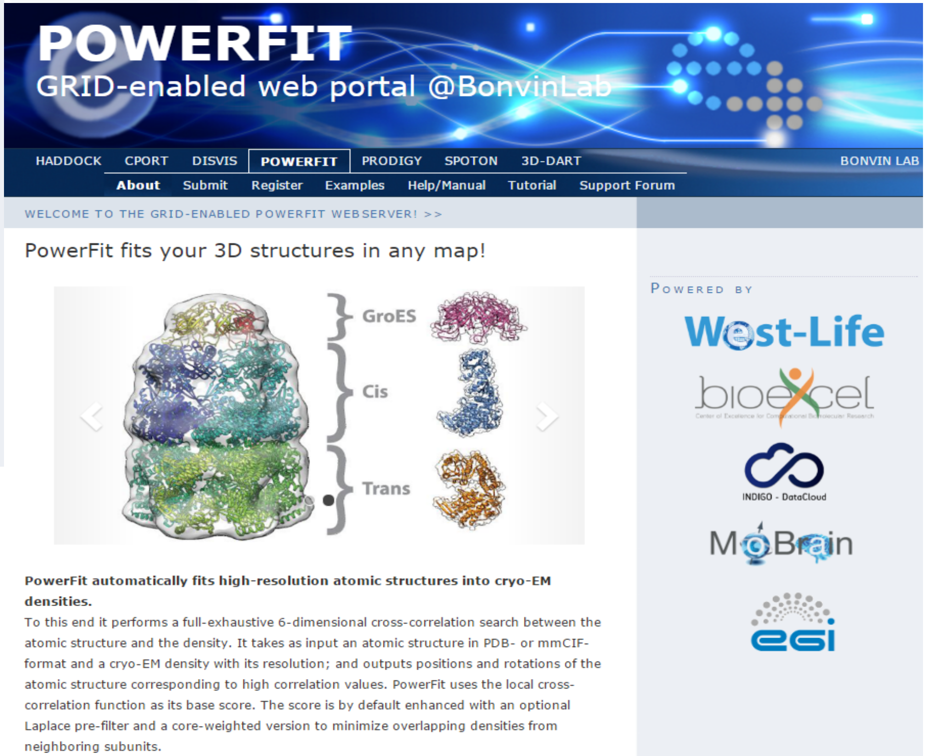
Andrii Salnikov (KNU)
Oleksandr Savytskyi (IMBG of NASU)

DisVis and PowerFit

EGI-ENGAGE



Case	Machine	TimeGPU (sec)	TimeCPU 1 core	CPU1/GPU	
B-K40	Baremetal	674	7928	11.8	
K-K40	KVM	671	7996	11.9	<- Cloud
B-K20	Baremetal	830	11839	14.3	
D- K20	Docker	837	11926	14.3	<- Grid

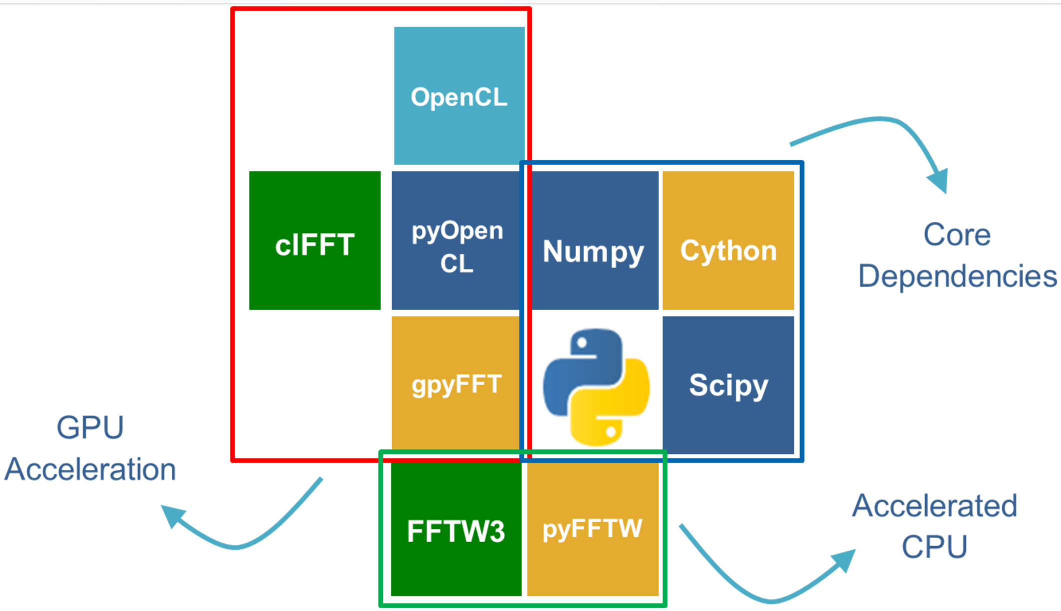


Talk at Biomedicine & Life Science I session today at 14:00

Full Tutorial given yesterday

DisVis and PowerFit on EGI platforms

Application requirements:



Docker engine not required on grid WNs: use udocker tool to run docker containers in user space (https://github.com/indigo-dc/udocker)

INDIGO - DataCloud

Solution for grid and cloud computing:

Docker containers built with proper libraries and OpenCL support:

indigodatacloudapps/**disvis**

indigodatacloudapps/**powerfit**

udocker
(Userspace Container Support)

disvis.sh job example

```
#!/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-$rnd disvis
/home/O14250.pdb /home/Q9UT97.pdb /home/restraints.dat -g -a 5.27 -vs 1
-d /home/out
./udocker.py rm disvis-$rnd
./udocker.py rmi indigodatacloudapps/disvis:nvdrv_$driver
cd $WDIR
tar zcvf res-gpu.tgz out/
```

Driver identification

Install udocker tool

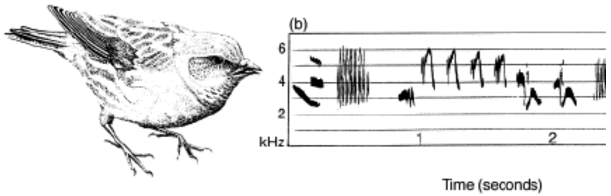
Pull DisVis image

Create the container

Run the container executing DisVis

Biodiversity

- LifeWatch is the European e-Science infrastructure for Biodiversity and Ecosystem Research (ESFRI)
- ANN & Pattern Recognition Tools can be applied in many cases:
  - Bird recognition (by sound)
  - Satellites data (land type, land use, water…)
  - Species classification



- Due to different features, like memory bandwidth or architecture, GPUs get much better performance in training ANN than CPUs
- They adopt Caffe: one of the most popular deep learning frameworks, implemented in pure C++/CUDA

  http://caffe.berkeleyvision.org

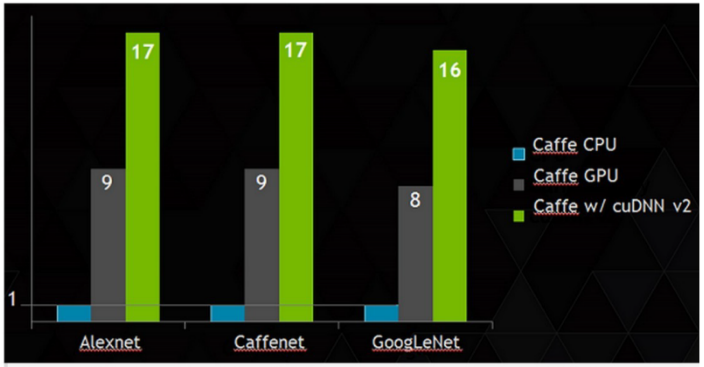
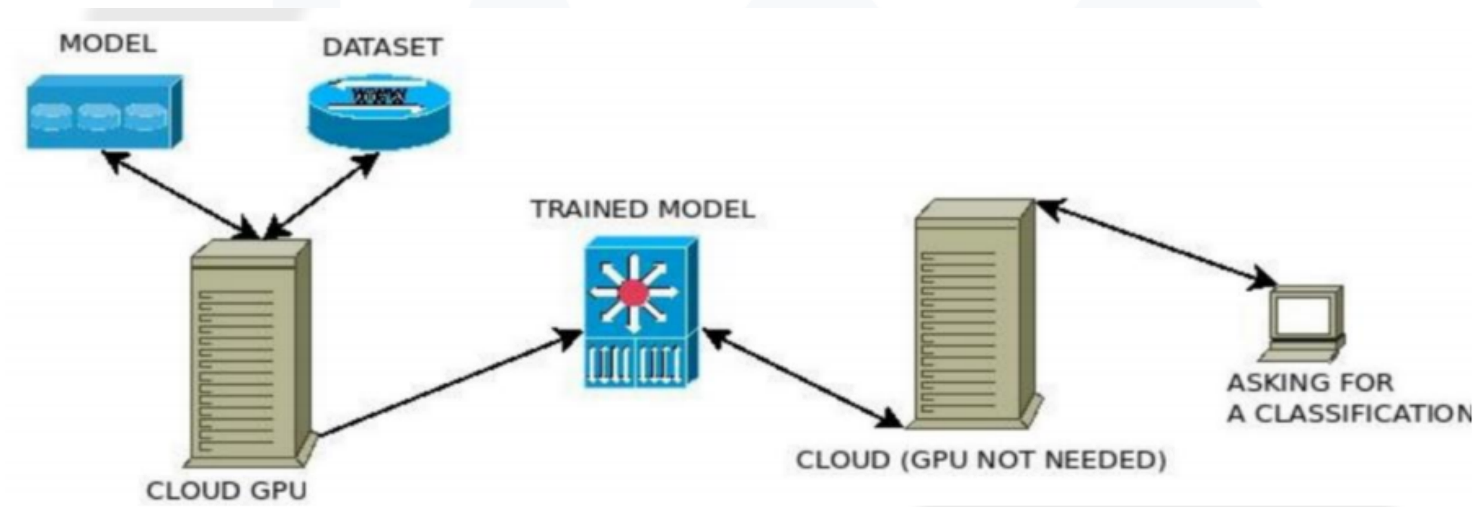
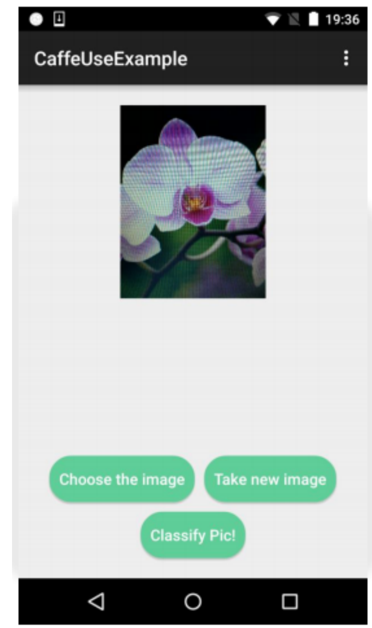


Figure 1: cuDNN performance comparison with CAFFE, using several well known networks. CPU is 16-core Intel Haswell E5-2698 2.3 GHz with 3.6 GHz Turbo. GPU is NVIDIA GeForce GTX TITAN X.

Flower recognition

- ANN on image recognition for photos taken with mobiles (see http://bit.ly/Bari-Lifewatch)
- Prototype based on Caffe framework trained with some flora images
- Deployed at IISAS-GPUCloud and at Seville cloud site with Tesla K20m GPUs





Thank you for your attention.

Questions?

Acknowledgments

Boris Parak (CESNET)
Barbara Krasovic (ARNES)
Alexandre Bonvin (Univ. of Utrecht)
Zeynep Kurkcuoglu (Univ. of Utrecht)
Jörg Schaarschmidt (Univ. of Utrecht)
Mikael Trellet (Univ. of Utrecht)
Ales Krenek (CESNET)
Mario David (LIP)
Jesus Marco (IFCA-CSIC)
Fernando Aguilar (IFCA-CSIC)
Stefano Dal Pra (INFN)
Daniel Traynor (QMUL)
Andrea Sartirana (CNRS-IN2P3)

EGI-Engage is co-funded by the Horizon 2020 Framework Programme
of the European Union under grant number 654142