Accelerated Computing support in EGI-Engage

Marco.Verlato (at pd.infn.it)

Istituto Nazionale di Fisica Nucleare (INFN)
Italy

www.egi.eu

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

Introduction/1

- Task part of WP4 (JRA2): Platforms for the Data Commons
  - Expand the EGI federated Cloud platform with new IaaS capabilities
  - Prototype an open data platform
  - **Provide a new accelerated computing platform**
  - Integrate existing commercial and public IaaS Cloud deployments and e-Infrastructures with the current EGI production infrastructure
- Duration: 1st March 2015 – 31st May 2016 (15 Months)
- Partners:
  - INFN – CREAM developers at Padua and Milan divisions
  - IISAS – Institute of Informatics, Slovak Academy of Sciences
  - CIRMMP – Scientific partner of MoBrain CC

Introduction/2

- Goal:
  - implement the support in the information system, to expose the correct information about the accelerated computing technologies available – both software and hardware – at site level, developing a common extension of the information system structure, based on OGF GLUE standard…
  - extend the HTC and cloud middleware support for co-processors, where needed, in order to provide a transparent and uniform way to allocate these resources together with CPU cores efficiently to the users
- Ideally divided in two subtasks:
  - Accelerated computing in the Grid
  - Accelerated computing in the Cloud
- Requirements from user communities collected at last EGI Conference in May 2015: see http://bit.ly/Lisbon-GPU-Session

Synergy with MoBrain CC

- A Competence Center to Serve Translational Research from Molecule to Brain
- Dedicated task for "GPU portals for biomolecular simulations"
  - To deploy the AMBER and/or GROMACS packages on GPGPU test beds, develop standardized protocols optimized for GPGPUs, and build web portals for their use
  - Develop GPGPU-enabled web portals for exhaustive search in cryo-EM density. This result links to the cryoEM task 2 of MoBrain
- Requirements
  - GPGPU resources for development and testing (CIRMMP, CESNET, BCBR)
  - Discoverable GPGPU resources for Grid (or Cloud) submission (for putting the portals into production
  - GPGPU Cloud solution, if used, should allow for transparent and automated submission
  - Software and compiler support on sites providing GPGPU resources (CUDA, openCL)

Accelerated computing on the Grid

- Accelerators:
  - GPGPU (General-Purpose computing on Graphical Processing Units)
    - NVIDIA GPU/Tesla/GRID, AMD Radeon/FirePro, Intel HD Graphics,...
  - Intel Many Integrated Core Architecture
    - Xeon Phi Coprocessor
  - Specialized PCIe cards with accelerators:
    - DSP (Digital Signal Processors)
    - FPGA (Field Programmable Gate Array)
- Work plan (until May 2016)
  - Indentifying the relevant GPGPU-related parameters supported by the different LRMS, and abstract them to significant JDL attributes
  - Implementing the needed changes in CREAM-core and BLAH components
  - Writing the info-providers according to GLUE 2.1
  - Testing and certification of the prototype
  - Releasing a CREAM-CE update with full GPGPU support (at least for Torque LRMS)
- Progresses recorded in https://wiki.egi.eu/wiki/GPGPU-CREAM

Work done so far: job submission/1

- Started from previous analysis and work of EGI Virtual Team (2012) and GPGPU Working Group (2013-2014)
- CIRMMP/MoBrain use-case and test-bed
  - AMBER and GROMACS applications with CUDA 5.5
  - 3 nodes (2x Intel Xeon E5-2620v2) with 2 NVIDIA Tesla K20m GPUs per node at CIRMMP
  - Torque 4.2.10 (source compiled with NVML libs) + Maui 3.3.1 (patched)
  - EMI3 CREAM-CE
- Tested local AMBER job submission with the different GPGPU supported options, e.g. with Torque/pbs_sched:

```
$ qsub -l nodes=1:gpus=2:default job.sh
$ qsub -l nodes=1:gpus=2:exclusive_process job.sh
```

- …and with Torque/Maui:

```
$ qsub -l nodes=1 -W x='GRES:gpu@2' job.sh
```

Work done so far: job submission/2

- GPUMode can have the following values for CUDA contexts:
  - 0/default – accept simultaneous CUDA contexts.
  - 1/exclusive_thread - Single context allowed, from a single thread.
  - 2/prohibited - No CUDA contexts allowed.
  - 3/exclusive_process - Single context allowed, multiple threads OK. Most common setting.
- New JDL attributes "GPUNumber" e "GPUMode" defined and implemented in BLAH and CREAM parser
- First prototype deployed at CIRMMP and remote submission tested:

```
$ glite-ce-job-submit -o jobid.txt -d -a -r cegpu.cerm.unifi.it:8443/cream-pbs-batch test.jdl
$ cat test.jdl
[
executable = "test_gpu.sh";
inputSandbox = { "test_gpu.sh" };
stdoutput = "out.out";
outputsandboxbasedesturi = "gsiftp://localhost";
stderror = "err.err";
outputsandbox = { "out.out","err.err","min.out","heat.out" };
GPUNumber=2;
GPUMode="exclusive_process";
]
```

EGI-ENGAGE

Work done so far: infosys/1

- GLUE2.1 draft analysed as a base for writing GPU-aware Torque infoprovider
  - **ExecutionEnvironment**
    - Is usually defined statically by YAIM during the deployment/configuration of the service
    - Info can be obtained e.g. from pbsnodes:

```
GLUE2ExecutionEnvironmentPhysicalGPUs
GLUE2ExecutionEnvironmentLogicalGPUs
GLUE2ExecutionEnvironmentGPUVendor
GLUE2ExecutionEnvironmentGPUModel
GLUE2ExecutionEnvironmentGPUVersion
GLUE2ExecutionEnvironmentGPUClockSpeed
```

```
gpus = 2
gpu_status =
gpu[1]=gpu_id=0000:42:00.0;gpu_pci_device_id=271061214;gpu_pci_location_id=0000:42:00.0;gpu_produ
ct_name=Tesla K20m;gpu_display=Enabled;gpu_memory_total=5119 MB;gpu_memory_used=11
MB;gpu_mode=Default;gpu_state=Unallocated;gpu_utilization=0%;gpu_memory_utilization=0%;gpu_ecc_
mode=Disabled;gpu_temperature=18
C,gpu[0]=gpu_id=0000:04:00.0;gpu_pci_device_id=271061214;gpu_pci_location_id=0000:04:00.0;gpu_pro
duct_name=Tesla K20m;gpu_display=Enabled;gpu_memory_total=5119 MB;gpu_memory_used=13
MB;gpu_mode=Default;gpu_state=Unallocated;gpu_utilization=0%;gpu_memory_utilization=0%;gpu_ecc_
mode=Disabled;gpu_temperature=16 C,driver_ver=319.37,timestamp=Thu Sep 17 10:18:07 2015
```

EGI-ENGAGE

Work done so far: infosys/2

– **ComputingManager**

- In the current draft GPU-related attributes declared only for cloud:

  `GLUE2CloudComputeManagerTotalGPU`

  `GLUE2CloudComputeManagerGPUVirtualizationType`

- Should GPUs be considered also in the context of a classical Computing Service? e.g. the attributes below are missing:

  `GLUE2ComputingManagerTotalGPU`

  `GLUE2ComputingManagerGPUVirtualizationType`

- There are no attributes specifying the number of used and/or free GPUs in the ComputingManager.

- It's not easy to calculate the number of free GPUs, especially when the GPUs can be shared among different threads or processes.
  In case of NVIDIA GPUs, it's worth investigating the information accessible with NVML.

Discussion thread at https://issues.infn.it/jira/browse/CREAM-177

Work done so far: accounting

- CREAM Accounting sensors, mainly relying on LRMS logs, were in the past developed by the APEL team (only for SLURM there was direct involvement of the INFN team)
- APEL team has been involved in the GPGPU accounting discussion
- Job accounting records of Torque and LSF8, LSF9 do not contain GPGPU usage info ☹
- NVML allows to enable per-process accounting of GPGPU usage using Linux PID, but not LRMS integration yet
- Discussion ongoing with APEL team

Current activity/1

- Started the implementation of the CREAM prototype for LSF
- RPs contacted so far and potentially interested:
  - STFC/Emerald (LSF8 based GPU cluster)
  - INFN-CNAF (LSF9 based GPU cluster)
  - IFCA (SGE based GPU cluster)
  - ARNES (SLURM based GPU cluster)
  - Queen Mary (SGE based cluster with OpenCL compatible AMD GPU)
- We are looking for NGI/RP providing HTCondor based GPU clusters
- Intel MIC clusters?

EGI-ENGAGE

Current activity/2

- Interested RPs can join the testbed providing us:
  - One or more testing application examples that we can execute on their GPU cluster and the commands/options used for their submission
  - Remote access to a server that can submit the above application sample jobs to their GPU cluster
  - Eventually, admin rights to allow us to deploy and test the CREAM prototype directly on top of their GPU cluster LRMS (as was done at CIRMMP)

EGI-ENGAGE

Next steps

- Implementing a GPGPU-enabled CREAM prototype for the other LRMSes
- Finalizing GLUE2.1 and writing the info-providers
  - INFN CREAM team produces info-providers for Torque and SLURM only
  - For LSF relies on CERN
  - For SGE relies on CESGA
  - For HTCondor relies on LAL
- GPGPU accounting
- Release testing and certification

EGI-ENGAGE

GPGPUs on the EGI FedCloud/1

- Work carried out by Viet Tran, Jan Astalos and Miroslav Dobrucky at IISAS
- Set up a testbed with:
  - IBM dx360 M4 server with two NVIDIA Tesla K20 accelerators.
  - Ubuntu 14.04.2 LTS with KVM/QEMU, PCI passthrough virtualization of GPU cards.
- Performance tests with NAMD molecular dynamics simulation (CUDA version), STMV test example
- Deployed Openstack/Kilo with GPU-enable flavors:
  - gpu1cpu6 (1GPU + 6 CPU cores)
  - gpu2cpu12 (2GPU +16 CPU cores)

EGI-ENGAGE

GPGPUs on the EGI FedCloud/2

- Fully integrated into EGI FedCloud in October 2015
  - Supported VOs: fedcloud.egi.eu, ops, dteam, moldyngrid, enmr.eu, vo.lifewatch.eu
  - Used sofar by moldyngrid, enmr.eu and vo.lifewatch.eu VOs

- A demonstration with Moldyngrid applications (a Virtual Laboratory for collaborative research in computational biology and bioinformatics part of Ukrainian Academic Grid Infrastructure) planned at EGI CF in Bari next week: http://bit.ly/Bari-GPU-Session

- More info, and instructions on how to create your own GPGPU server in cloud available at https://wiki.egi.eu/wiki/GPGPU-FedCloud

Thank you for your attention.

Questions?

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