PLGrid Workshop - Computational chemistry calculations in PLGrid Infrastructure

Klemens Noga

ACC Cyfronet AGH

Current Trends in Theoretical Chemistry VII, Kraków 8 IX 2016

Agenda

- **PLGrid Infrastructure**
- **Prometheus & Zeus clusters HPC clusters at ACC Cyfronet AGH**
  - available resources
  - access to clusters/data transfer
- **Performing calculations**
    - software environment management using Modules/Lmod framework
    - batch scripts
      - sequential and parallel runs
    - best practices
- **Documentation and users support**

PLGrid - computational infrastructure for science

Computing resources

- 3,6+ PTFLOPS
- 90 000+ cores

Scientific Software

- 500+ applications, tools, libraries
- apps.plgrid.pl

Storage

- 20+ PB
- fast scratch
- distributed access

Tools for collaboration

- project tracking (JIRA)
- version control (Git)
- teleconferencing (Adobe Connect)

Computational Cloud

- PaaS based on OpenStack

PLGrid - computational infrastructure for science

- The PLGrid Infrastructure is available free of charge for Polish researchers and all those engaged in scientific activities in Poland
- On-line registration through PLGrid Users' Portal - portal.plgrid.pl
- User verification based on Polish Science Database - www.nauka-polska.pl

On PLGrid Users Portal user can

- apply for access to tools and services
- monitor utilization of resources
- manage their computational grants and grid certificates

Access to all PLGrid resources through **one account** and **one passphrase** (or grid certificate)

PLGrid Consortum

Academic Computer Centre CYFRONET AGH, Kraków (Coordinator)

Tricity Academic Computer Centre, Gdańsk

Poznan Supercomputing and Networking Center, Poznań

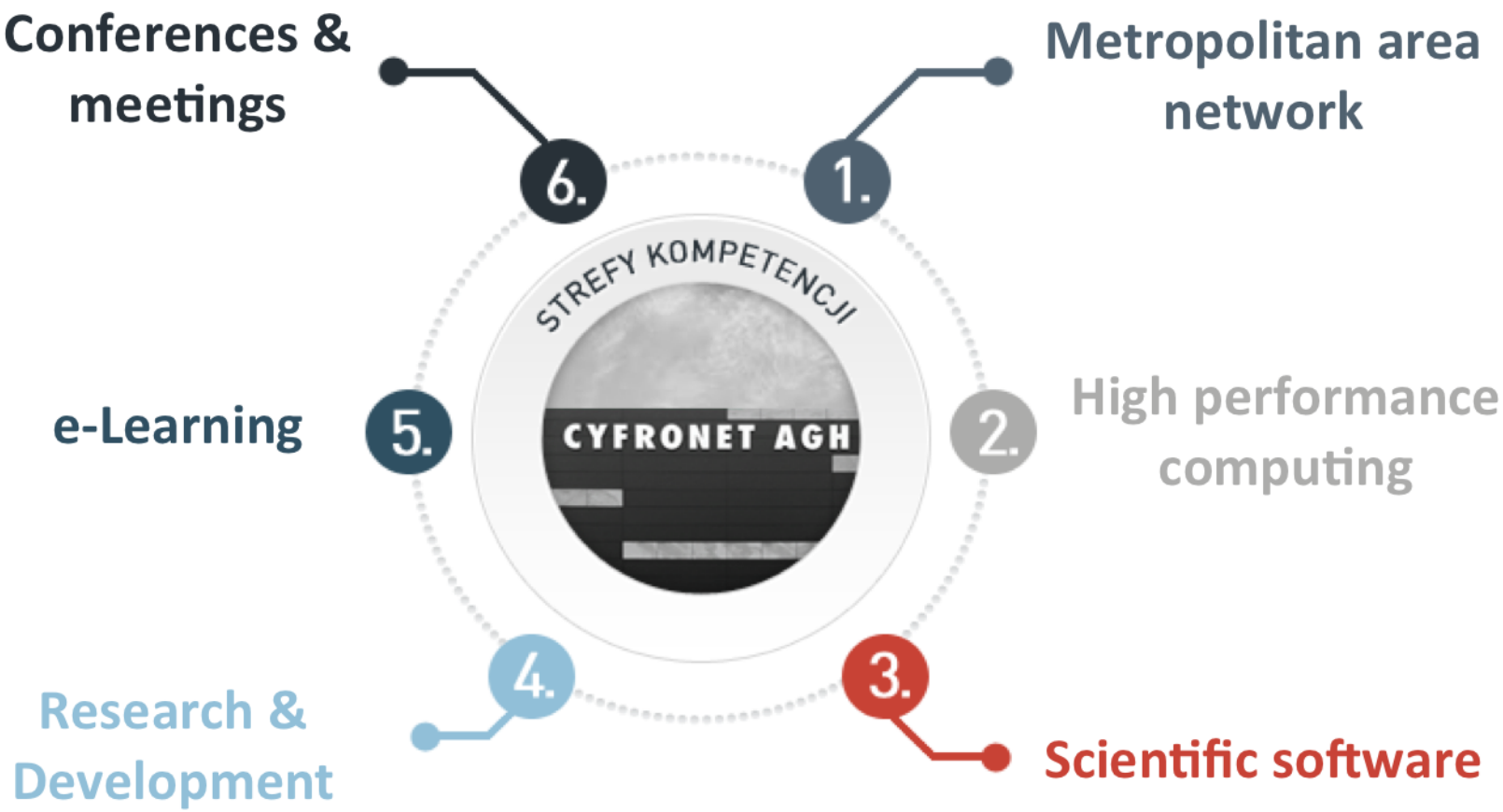
Interdisciplinary Centre for Mathematical and Computational Modelling, Warszawa

Wroclaw Centre for Networking and Supercomputing, Wrocław

ACC Cyfronet AGH

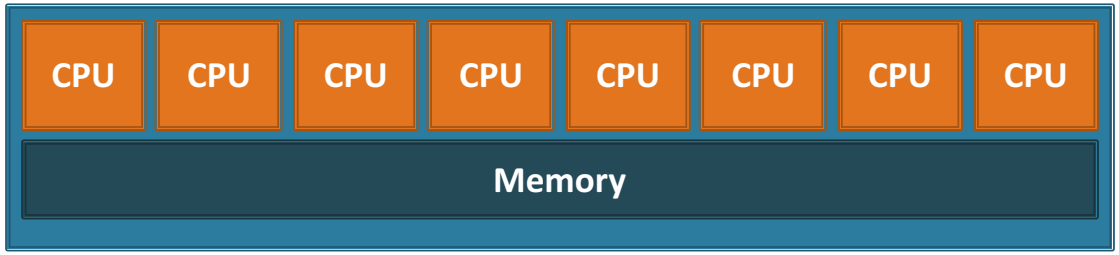
- **Academic Computer Centre Cyfronet AGH**

ACC Cyfronet AGH – main activity areas

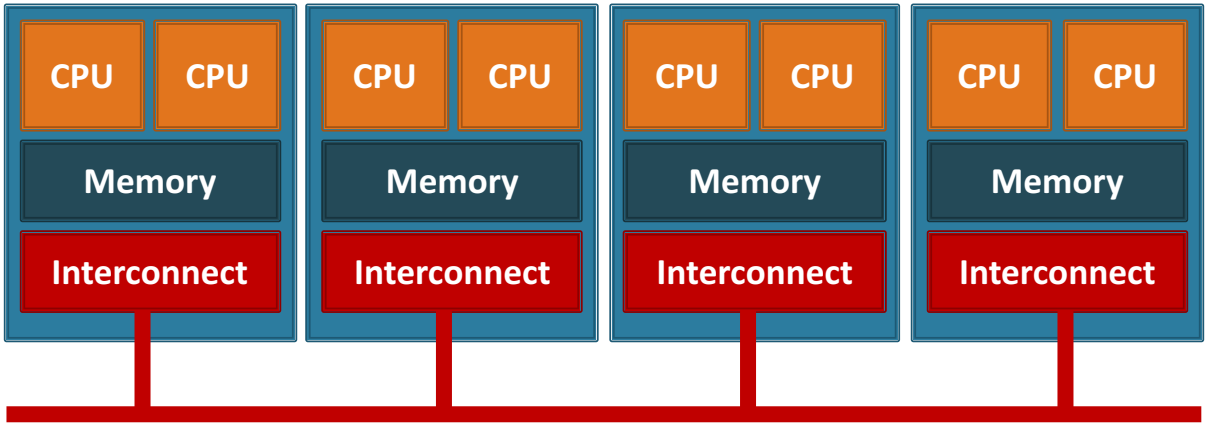


High performance computing

SMP



Cluster

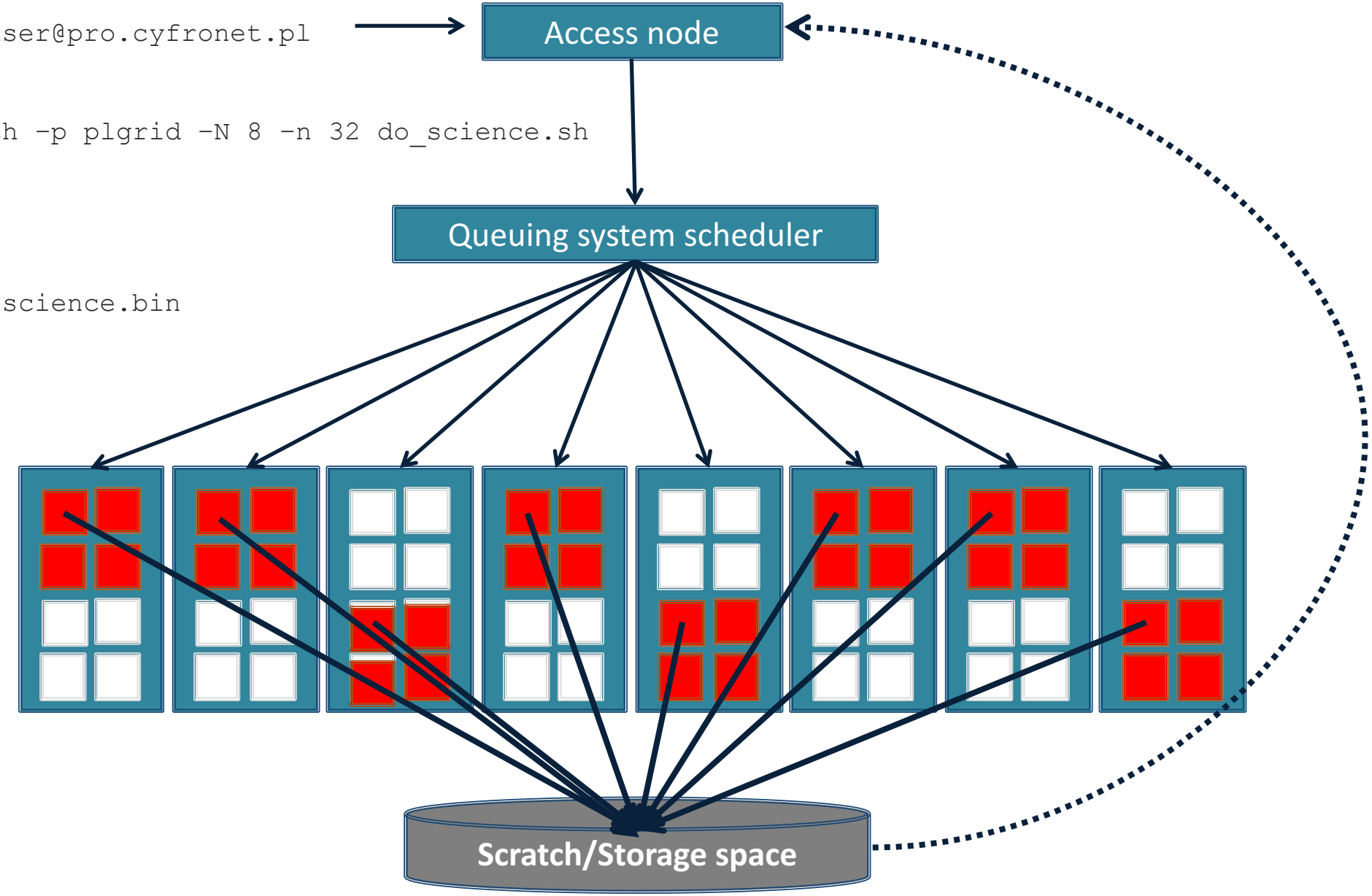


High performance computing

```
ssh user@pro.cyfronet.pl
```

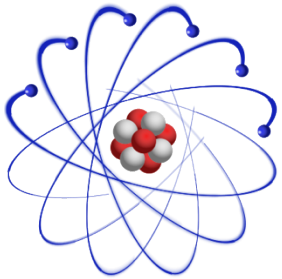
```
sbatch -p plgrid -N 8 -n 32 do_science.sh
```

```
srun science.bin
```



Access to computing clusters

- All PLGrid HPC clusters use Linux as OS
  - Scientific Linux 6 on Zeus
  - CentOS 7 on Prometheus



- HPC clusters contain
  - user interface (UI) node(s)
  - computing nodes (a.k.a worker nodes)
- User interface **must not be used** for computing
- Fair share between users tasks and computations provided by queuing system
  - PBS/Torque/Moab on Zeus
  - SLURM on Prometheus

Access to computing clusters

- User log on user interface (UI) node using SSH protocol
  - UI names:
    - **login@zeus.cyfronet.pl**
    - **login@prometheus.cyfronet.pl (login@pro.cyfronet.pl)**
  - SSH clients
    - on Linux and MacOS included in OS
      - **`ssh`** command in terminal
    - on Windows
      - PuTTY - http://www.chiark.greenend.org.uk/~sgtatham/putty/
      - KiTTY - http://www.9bis.net/kitty/
      - Babun - http://babun.github.io/faq.html
      - MobaXterm - http://mobaxterm.mobatek.net
  - copying files and directories
    - on Linux and MacOS included in OS
      - **`scp`** command in terminal
    - on Windows
      - WinSCP - http://winscp.net/

Access to computing clusters during turorial

- Access credentials
  - accounts on local tutorial computers in the room:
    - Linux:
      - user: **cttc_XX (where XX=01-15)**
      - password: **cttc7_plgrid**
    - Windows:
      - user: **Student**
      - **passwordless**
  - tutorial's accounts @Prometheus cluster:
    - user: **tutorialXX (where XX=01-32)**
    - password: on leaflet
    - host (UI): login@prometheus.cyfronet.pl (login@pro.cyfronet.pl)

Prometheus HPC cluster

- Prometheus consist user interface nodes (UI), service nodes and worker nodes
  - worker nodes (2 232 nodes, each witch 2x Intel Xeon E5-2680v3 processors)
    - 72 nodes with additional GPPGU (2x nVidia Tesla K40XL)

Property	Prometheus
CPU frequency	2.50 GHz
RAM	128 GB
cores per node	24
InifiniBand interconnect	available, FDR 56 Gb/s

Zeus – hybrid HPC cluster

- Zeus consist user interface node (UI) and several groups of nodes with different configurations
  - normal worker nodes (1198 nodes including 136 with vast RAM amount)
  - GPGPU – nodes with GPGPU (44 nodes, 208 GPGPU cards)

Property	Zeus	Zeus BigMem	Zeus GPGPU
CPU frequency	2.26-2.67 GHz	2.67; 2.30 GHz	2.93; 2.40 GHz
RAM	16, 24 GB	96, 256 GB	72, 96 GB
cores per node	8, 12	12, 64	12
InifiniBand interconnect	available, QDR	available, QDR	available, QDR
additional		–	GPGPU cards

Prometheus – file systems

- Storage of data – NFS (quite slow, should not be used for heavy I/O calculations)
  - `$HOME` – user's home directory
    - quota 40 GB
  - `$PLG_GROUPS_STORAGE` – additional storage gained through PLGrid grants system
- Temporary scratch file systems
  - `$SCRATCH` – distributed scratch Lustre file system
    - accessible from all nodes of cluster (including UI)
    - `$TMPDIR` and `$SCRATCHDIR` – unique subdirectories on `$SCRATCH` created for the job at it's start
- To check quota use `pro-fs`

Zeus – file systems

- Storage of data – NFS (quite slow, should not be used for heavy I/O calculations)
  - `$HOME` – user's home directory
    - quota 7 GB
    - daily backup
  - `$PLG_GROUPS_STORAGE` – additional storage gained through PLGrid grants system
  - `$STORAGE` – for long lasting storage of data
    - quota 100 GB
- Temporary scratch file systems
  - `$SCRATCH` – distributed scratch Lustre file system
    - accessible from all nodes of cluster (including UI)
  - `$TMPDIR` – local file system located on worker node
    - accessible only from node to which it is attached
    - accessible only during the run of computation task
- To check qouta use `zeus-fs`

Software

- Scientific software usually needs specific runtime environment (i.e. additional libraries) and sometimes technical knowledge is needed to install them efficiently
- Modules and Lmod packages are solutions for loading runtime environments on every cluster in PLGrid infrastructure
- Advantages
  - simplicity of preparing software to run efficiently
  - computation scripts could be transferable between HPC clusters
  - possibility of concurrent runs of different versions of software
  - on hybrid HPC systems transparent switching to most efficient version of software
- Drawbacks
  - additional command to remember .-)

Software – modules

- Load environment for scientific package
  - `module add <module-name>` (i.e. `module add plgrid/apps/gaussian`)
  - `module load <module-name>` (i.e. `module load plgrid/apps/matlab`)
- Remove module
  - `module rm < module-name >` (i.e. `module rm plgrid/apps/gaussian`)
  - `module unload < module-name>` (i.e. `module unload plgrid/apps/matlab`)
- Listing of all available modules
  - `module avail`
  - `module avail plgrid/tools` (only from `tools` branch)
  - `module avail plgrid/apps/gaussian` (all available Gaussian versions)
  - `module spider gaussian` (all available Gaussian versions)
- Listing of loaded modules
  - `module list`
- Clearing all loaded modules
  - `module purge`
- Saving collection of modules for later use, restoring it and listing saved collections
  - `module save [collection]` and `module restore [collection]`
  - `module savelist` and `module describe [collection]`

Software – modules

- Each software package installed in PLGrid infrastructure has it's own module
  - `plgrid/<branch>/<software-name>/<version>`
- Branch kinds
  - `apps` – for most of scientific packages
  - `libs` – for software libraries
  - `tools` – for toolkits and helper packages
- User's own modules
  - `module use path` – adds `path` with additional modules
- Examples:
  - `plgrid/tools/intel/16.0.1`
  - `plgrid/apps/gaussian/g09.E.01`
  - `plgrid/tools/python/3.4.2`

https://apps.plgrid.pl/

Queuing system

- Queuing system
  - manage all computational task on cluster
  - monitor available resources
  - acts as matchmaker between needs of jobs and resources
  - empowers fair share between different users

- All computational tasks are run as **jobs** queued in **queues** and run according to their priority and available resources.
- Priority of job depends on
  - amount of resources obtained by user in computational grant
  - amount of resources requested by job
    - **maximum wall time of computation** is most essential resource
  - amount of other resources concurrently used by job's owner

Queuing systems in PLGrid

- HPC clusters available in PLGrid use several kinds of queuing systems
  - SLURM (http://slurm.schedmd.com)
  - PBS:
    - Torque (http://www.adaptivecomputing.com/products/open-source/torque/)
    - PBS Pro (http://www.pbsworks.com/product.aspx?id=1)

HPC Centre	Cluster	Queuing system
ACC Cyfronet AGH	Prometheus	SLURM
	Zeus	Torque/Moab
ICM	Hydra	SLURM
	Topola	SLURM
PSNC	Eagle	SLURM
	Inula	PBS
TASK	Tryton	SLURM
	Galera Plus	PBS/Maui
WCSS	Bem	PBS Pro

SLURM queuing system – commands

- User interact with SLURM queuing system using commands
  - `sbatch` – to submit new job to queue
  - `squeue` – gives information about jobs running in queuing system
  - `scancel` – deletes jobs from queue
  - `sinfo`/`scontrol` – gives detailed information about queue, job or node
  - `smap` – gives graphical information about state of HPC cluster
  - `srun` – runs interactive job or step in batch job

- Each job has got **unique job identifier** (jobID)

SLURM queuing system – job submission

- Command `sbatch` submits new job in queue
- All parameters describing job's requirements could be included in batch script and given to queuing system using command
  - `sbatch [options] script.slurm`
- Example script

```
#!/bin/env bash

# Commands that will be run after start of the job
echo "Computation started on work node: "; hostname

module add plgrid/apps/matlab

matlab -nodisplay <matlab.in >matlab.out
```

SLURM queuing system – job monitoring

- Commands `squeue` and `pro-jobs` give view of jobs scheduled in queuing system
- Jobs States
  - `PD` – queued
  - `R` – running

- Additional helpful flags
  - `squeue --user $USER` – information about `$USER`'s jobs
  - `pro-jobs -j <jobID>` – information about specified jobs
  - `pro-jobs -N` – additional information about information about exec nodes
  - `pro-jobs -q/-r` – information about queued (pending)/running jobs only
  - `pro-jobs -h` – help screen
- In addition `scontrol`, `sinfo` and `smap` give information about status of cluster
  - `scontrol show job <jobID>` – information about `<jobID>` job
  - `scontrol show node <nodes_list>` – information about nodes

Available partitions

Partitions	max time	Information
plgrid-testing	1:00:00	
plgrid	3-00:00:00	
plgrid-long	7-00:00:00	*
plgrid-gpu	3-00:00:00	nodes with GPGPU*

- `scontrol show partitions <patition_name>` – detailed information about partition
- `sinfo` – lists all available nodes in all partitions
  - `sinfo -p <partition_name>` – lists information only about partition
- default time in all `plgrid*` partitions is set to 15 minutes

* - partitions available after request

SLURM – example ADF job

```
#!/bin/env bash

# Commands that will be run after start of the job
echo "Computation started on work node: "; hostname

module add plgrid/apps/adf

adf < ethanol.in > ethanol.log
```

- SLURM options provide information about job requirements to queuing system. They could be
  - given in command line `sbatch [SLURM options]`
  - included in first lines of batch script with `#SBATCH` at start of line

SLURM – `sbatch` command options

- `sbatch` command uses various options to provide queuing system with additional info about the job
  - `-p <partition>, --partition=<partition>` defines partition
  - `-J <jobname>, --job-name=<jobname>` give name to job
  - `-a, --array=<indexes>` submit a job array
  - `--mail-user=<user's e-mail>` setting email for notifications
  - `--mail-type=<type>` information when notifications should be send: at beginning (`BEGIN`), end (`END`) or execution error (`FAIL`)
  - `-A <grantID>, --account= <grantID>` information about computational grant (if omitted job use default)

- When option `-p` is omitted job is queued into default partition (on Prometheus `plgrid`)

SLURM – job requirements specification

- There are several recourses available for job
  - `-t, --time=<time>` total maximal execution wall time of job
  - `-N, --nodes=<nodes>` amount of nodes allocated to job
  - `-n, --ntasks=<number>` amount of tasks allocated to job
  - `--ntasks-per-node=<ntasks>` amount of tasks invoked on each node
  - `--cpus-per-task=<cores>` amount of cores per each task (i.e. when using threads in OpenMP)
  - `--mem=<MB>` amount of memory per node requested by job
  - `--mem-per-cpu=<MB>` amount of memory per core requested by job

- Parameter formats
  - time format: `"min", "min:sec", "hours:min:sec", "days-hours", "days-hours:min"` and `"days-hours:min:sec"`
  - memory: `MB (=1024kB), GB (=1,024MB)`

SLURM – example ADF job

```
#SBATCH -J adf.ethanol
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<user's e-mail>
#SBATCH --time=10:00
#SBATCH --mem=24GB
#SBATCH -p plgrid
#SBATCH -A tutorial

module add plgrid/apps/adf

cd $SLURM_SUBMIT_DIR

adf < ethanol.in > ethanol.log
```

- In SLURM job is sent to partition not to queue
  - flag `-p <partition_name>` or `--partition <partition_name>`
  - partition for PLGrid users: **`plgrid*`**

SLURM – example parallel ADF job

```
#SBATCH -J adf.ethanol.parallel
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<user's e-mail>
#SBATCH --time=10:00
#SBATCH --mem=24GB
#SBATCH -p plgrid
#SBATCH -A tutorial

module add plgrid/apps/adf

cd $SLURM_SUBMIT_DIR

adf < ethanol.in > ethanol.log
```

- ADF is parallelized using MPI
  - use `--ntasks-per-node=<cores_per_node` and `--nodes=<no_of_nodes>` for request of resources
  - variable `SLURM_NTASKS` holds information about number of tasks to be run

SLURM – example parallel Gaussian job

```
#SBATCH -J gaussian.parallel
#SBATCH --nodes=1
#SBATCH --cpus-per-task=24
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<user's e-mail>
#SBATCH --time=10:00
#SBATCH --mem=24000
#SBATCH -p plgrid

module add plgrid/apps/gaussian

cd $SLURM_SUBMIT_DIR

g09 input.gjf
```

- Gaussian on PLGrid clusters uses only shared memory mode
  - use `--cpus-per-task=<cores_per_job>` and `--nodes=1` for request of resources
  - variable `SLURM_CPUS_PER_TASK` holds information about number CPUs allocated to each task

SLURM – environmental variables

- SLURM adds environmental variables which could ease performing computation

Variable	Description
SLURM_JOB_ID	job identifier (jobID)
SLURM_SUBMIT_DIR	dir, from which batch script was submitted to queuing system
SLURM_NTASKS	total number of tasks (i.e. MPI processes) in the current job
SLURM_NTASKS_PER_NODE	number of tasks to be run on one node
SLURM_NODELIST	list of nodes allocated to the job
SLURM_CPUS_PER_TASK	number of cores requested per task
TMPDIR, SCRATCHDIR	scratch file temporary directories for job
SCRATCH	$USER's root scratch directory on distributed Lustre file system
SCRATCHDIR	unique directory for the job on $SCRATCH

- Environment variables can be used to control distribution of job
  - MPI jobs: SLURM_NTASKS to run MPI processes (using srun) variable
  - OpenMP jobs: SLURM_CPUS_PER_TASK to run proper number of threads
  - hybrid MPI/OpenMP jobs: combine SLURM_NTASKS to run MPI processes and SLURM_CPUS_PER_TASK to expand threads

SLURM – interactive jobs

- Interactive work on cluster should be done using interactive jobs trough `srun` command
  - `srun -p plgrid -A <grant_id> -n 1 --pty /bin/bash`

- User interface **must not be used** for computing

- To attach terminal to running batch job
- `srun -N1 -n1 --jobid=<jobID> --pty /bin/bash`
- `srun -N1 -n1 --jobid=<jobID> -w <nodeID> --pty /bin/bash`
- `sattach <jobid.stepid>`

- Prometheus helper script `ssh_slurm`
  - `ssh_slurm <jobid> <dest_host> [command]`

SLURM – array jobs

- Multiple jobs can be executed with identical parameters within one `sbatch` run as array jobs when `-a, --array=<indexes>` option used
  - `sbatch -a n-m,k,l script.slurm` (np. `sbatch -a 0-9` or `sbatch -a 2,4,7`)
- All jobs within array have same value of `SLURM_SUBMIT_DIR` and `SLURM_ARRAY_JOB_ID` variables, but have additional unique identifier `SLURM_ARRAY_TASK_ID`

```
#!/bin/env bash
#SBATCH -a 0-4,9
#SBATCH --time=5:00
OUTPUTDIR=$SLURM_SUBMIT_DIR/$SLURM_ARRAY_JOB_ID
mkdir -p $OUTPUTDIR
cd $TMPDIR
hostname > task.$SLURM_ARRAY_TASK_ID

mv task.$SLURM_ARRAY_TASK_ID $OUTPUTDIR
```

- `squeue -a` – shows all jobs in array queued in system

SLURM – job's dependencies

- Dependencies between jobs can be added through `--dependency=<dependency_list>` option

- Possible dependencies
  - `after:job_id[:jobid...]` – job can begin execution after the specified jobs have begun execution
  - `afterany:job_id[:jobid...]` – job can begin execution after the specified jobs have terminated
  - `afternotok:job_id[:jobid...]` – job can begin execution after the specified jobs have terminated in some failed state
  - `afterok:job_id[:jobid...]` – job can begin execution after the specified jobs have successfully executed
  - `expand:job_id` – resources allocated to this job should be used to expand the specified job
  - `singleton` – job can begin execution after any previously launched jobs sharing the same job name and user have terminated

SLURM – jobs with GPGPUs

- GPGPUs are shown in SLURM queuing system as generic resources (GRES) with `gpu` identifier.
- To check where GPGPUs are available
  - `sinfo -o '%P || %N || %G'`
- To request GPGPUs for a job `--gres=gpu[:count]` has to be added to sbatch/srun command
  - `srun -p plgrid-gpu -N 2 --ntasks-per-node=24 -n 48 -A <grant_id> --gres=gpu[:count] --pty /bin/bash -l`
  - `#SBATCH --gres=gpu[:count]`
- GPGPUs are available only in `plgrid-gpu` partition

SLURM `scancel` – deleting jobs

- `scancel` command is used to delete unwanted jobs from queuing system
  - `scancel <JobID>`

- Information about jobs which cannot be deleted using `scancel` should be sent to system administrators through
  - Helpdesk PLGrid PL
    - https://helpdesk.plgrid.pl
    - helpdesk@plgrid.pl
  - directly to system administrators prometheus@cyfronet.pl

Job monitoring with `pro-jobs*`

- `pro-jobs` and `pro-jobs-history` could be used to monitor efficiency of jobs
  - memory usage
  - CPU usage

- `pro-jobs` – running and queued jobs
- `pro-jobs-history` – historical data of completed jobs

- `pro-jobs*` usage
  - `pro-jobs -N` – additional information about nodes of job(s)
  - `pro-jobs -v` – more detailed information about job(s)
  - `pro-jobs -j (<jobID>)` – information only about job(s)
  - `pro-jobs -h` – help screen

Best practices

- SLURM job batch script is always started in directory from which it was submitted to queuing system. Access to that directory is also possible with `SLURM_SUBMIT_DIR`

- All batch jobs have got file in which data from standard outputs (both standard output stream *stdout* and standard error stream *stderr*) is stored named `slurm-<JobID>.out`
  - those file should not be big (less than several MBs) and are stored in `SLURM_SUBMIT_DIR`
  - `-o, --output=<file>` and `-e, --error=<file>` - options to redirect *stdout* and *stderr*

- When commands in SLURM script print big amount of data into output streams user should redirect that data to file(s)
  - for standard output stream (*stdout*): `command > file.out`
  - for standard error stream (*stderr*): `command 2> file.err`
  - for both streams to one file: `command &> file.log`

- `$HOME` and `$PLG_GROUPS_STORAGE` **must not be used** for heavy I/O computations

Best practices

- During batch job submission user should always
  - specify maximal time of job execution (parameter `t/time`)
  - specify maximal RAM amount needed by job through `mem` (or `mem-per-cpu`)
  - enable checkpoints
  - for parallel computations use all cores on nodes when possible
  - when big amount of data is used in computation always use `$SCRATCH` for files
  - when big amount of data is going to be passed to standard output streams redirect it to files and use `$SCRATCH`
  - load runtime environment of software via `module` command in batch script
  - do not load software modules in scripts loaded at user's login (i.e. `.bashrc`)

Own software compilation

- Always compile on **computing node** using
  - batch job with `sbatch`
  - interactive job with `srun --pty bash`
  - use modules (also for libraries)
    - Intel® MKL Link Line Advisor: https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/
    - when software could be used by various users consider global compilation by admins
  - use IntelMPI (`plgrid/tools/impi`) and OpenMPI (`plgrid/tools/openmpi`) modules build by admins
  - check threads and processes bindings (i.e. `KMP_AFFINITY` environment variable)

- Common compilation flags on Haswell CPUs
  - GCC: `-march=native`, `-fopenmp`
  - Intel: `-xCORE-AVX2` (or `-xHost`), `-qopenmp`, `-fma-/-no-fma`
  - PGI: `-tp haswell`, `-mp`, `-fast`, `-Mipa=fast,inline`, `-i8`

- When in trouble contact admins trough PLGrid Helpdesk
  - https://helpdesk.plgrid.pl/
  - helpdesk@plgrid.pl

Gaussian

plgrid/apps/gaussian

- Versions available at Prometheus cluster
  - `g09.C.01,g09.D.01,g09.E.01`

- Usage
  - `module add plgrid/apps/gaussian/<version>`
  - `g09 input.gjf`

- Remarks
  - big amount of implemented methods (and DFT functionals)
  - wide spread use in scientific community
  - easy to learn input syntax

- Usage restrictions
  - only SMP mode (up to 24 computing cores@Prometheus)

Q-Chem

plgrid/apps/qchem

- Versions available at Prometheus cluster
  - 4.3, 4.4.1

- Usage
  - module add plgrid/apps/qchem/<version>
  - qchem -nt <number_of_cores> input.inp output.out

- Remarks
  - big amount of implemented methods (and DFT functionals)
    - Constrained DFT
  - easy to learn input syntax

- Usage restrictions
  - only SMP mode (up to 24 computing cores@Prometheus)
    - multi-node MPI version in preparation

ADF

plgrid/apps/adf

- Versions available at Prometheus cluster
  - `2013.01,2014.07,2014.10,2016.102`

- Usage
  - `module add plgrid/apps/adf/<version>`
  - `adf < input.in >& output.log`

- Remarks
  - uses STO bases
  - wide spread use for metaloorganic compounds
    - Zeroth Order Regular Approximation (ZORA)
  - easy to learn input syntax
  - included COSMO-RS module
  - big amount of tools for results analysis (NBO, NOCV, ETS)
  - ADFView for visualisations included

Turbomole

`plgrid/apps/turbomole`

- Versions available at Prometheus cluster
  - `7.0, 7.0.1, 7.1`

- Usage
  - `module add plgrid/apps/turbomole/<version>`
  - preparation of directory with input using scripts (i.e. `x2t`, `define`, `cosmoprep`)
  - run using various scripts for various methods (i.e. `dscf`, `ridft`, `ricc2`, `jobex`, `aofroce`, `egrad`, `riper`)

- Remarks
  - fast and good scaling with system size
  - DFT calculations for periodic systems
  - good parallelization

Molpro

plgrid/apps/molpro

- Versions available at Prometheus cluster
  - `2012.1.24`, `2015.1.4`, `2015.1.11` (in preparation)

- Usage
  - `module add plgrid/apps/molpro/<version>`
  - `molpro [options] input.inp`
  - `--single-helper-server` for jobs on multiple nodes and big memory requirements

- Remarks
  - Electronically excited states using i.a MCSCF/CASSCF, CASPT2, MRCI, or FCI methods
  - Parallel explicitly correlated methods MP2-F12, CCSD(T)-F12,
  - good parallelization of “high-level” methods

Jaguar (and small molecule drug discovery suite)

`plgrid/apps/schrodinger`

- Versions available at Prometheus cluster
  - `2015-3`, `2016-1`, `2016-3` (in preparation)

- Usage
  - `module add plgrid/apps/schroedinger/<version>`
  - `jaguar run [options] input.inp`

- Remarks
  - High-quality initial guess for transition metals
  - solvent effects by applying a self-consistent reaction field (SCRF)
  - Maestro – sophisticated GUI
  - Automated workflows through Python scripts
  - other programs included in SMDDS i.a. Desmond, Glide, Qsite, QSAR

- Usage restrictions
  - only SMP mode (up to 24 computing cores@Prometheus)

Terachem

plgrid/apps/terachem

- Versions available at Zeus cluster
  - `1.45`, `1.5`,
  - Version `1.9` in preparation on Prometheus cluster

- Usage
  - `module add plgrid/apps/terachem/<version>`
  - `$TERACHEMRUN input.inp > output.log`

- Remarks
  - fast calculations on GPGPUs
  - HF, DFT, TD-DFT (in `1.9`) and MD runs

- Usage restrictions
  - only on nodes with GPGPUs (option `--gres=gpu[:count]`)

Niedoida

`plgrid/apps/niedoida`

- Versions available at Prometheus cluster
  - `0.5.3` (`0.6` in final development)

- Usage
  - `module add plgrid/apps/niedoida`
  - `niedoida --no-cores=XX input.inp`

- Remarks
  - HF, DFT (with DF and RI), TD-DFT and Dressed TD-DFT
  - major improvements in version 0.6
    - HF, DFT computations on GPGPUs

CP2K

`plgrid/apps/cp2k`

- Versions available at Prometheus cluster
  - `2.6.1` (`3.0.x` in preparation)

- Usage
  - `module add plgrid/apps/niedoida`
  - `$CP2K_RUN -i geopt.inp`

- Remarks
  - FF, HF, DFT and MP2 for periodic systems
  - MD runs

PBS queuing system - commands

- User interact with PBS queuing system using commands
  - `qsub` – to submit new job to queue
  - `qstat` – gives information about jobs running in queuing system
  - `qdel` – deletes jobs from queue
  - `qalter` – modifies queued job

- Each job has got **unique job identifier** (jobID)

PBS queuing system – job submission

- Command `qsub` submits new job in queue
- All parameters describing job's requirements could be included in batch script and given to queuing system using command
  - `qsub script.pbs`
- Example script

```
#!/bin/env bash

# Commands that will be run after start of the job
echo "Computation started on work node: "; hostname

module add plgrid/apps/matlab

matlab -nodisplay <matlab.in >matlab.out
```

PBS queuing system – job monitoring

- Commands `qstat` and `zeus-jobs` give view of jobs scheduled in queuing system
- Jobs States
  - `Q` – queued
  - `R` – running
- Additional helpful flags
  - `qstat -u $USER` – information about `$USER`'s jobs
  - `qstat -n <jobID>` – information about nodes allocated for job `<jobID>`
  - `qstat -q` – view of general state of system
  - `zeus-jobs -e-` or `zeus-jobs -e+` - jobs sorted according to efficiency
  - `zeus-jobs -w` – lists jobs only with low efficiency
  - `zeus-jobs -f (<jobID>)` – detailed information about jobs
  - `zeus-jobs -h` – help screen

Available queues

Queue	max time	Information
l_test	0:15:00	for tests
l_prio	1:00:00	
l_short	3:00:00	default
l_long	336:00:00	
l_exclusive	336:00:00	jobs allocating whole nodes
l_interactive	72:00:00	interactive jobs
plgrid-testing	1:00:00	
plgrid	72:00:00	
plgrid-long	168:00:00	
l_infinite	2160:00:00	*
l_bigmem	336:00:00	nodes with big RAM*
gpgpu	336:00:00	nodes with GPGPU*

`qstat -Q -f <queue-name>` – detailed information about queue

`qstat <queue-name>` – lists jobs in specified queue

* - queues available after request

PBS - example Gaussian job

```
#!/bin/env bash

# Commands that will be run after start of the job
echo "Computation started on work node: "; hostname

module add plgrid/apps/gaussian

g09 h2o.gjf
```

- PBS options provide information about job requirements to queuing system. They could be
  - given in command line `qsub [PBS's options]`
  - included in first lines of batch script with `#PBS` at start of line

PBS - example Gaussian job

```
#!/bin/env bash

# Commands that will be run after start of the job
echo "Computation started on work node: "; hostname

module add plgrid/apps/gaussian

echo "Current working directory:"; pwd
# Changing directory to one from which PBS script was stated
(where inputs should be stored)
cd $PBS_O_WORKDIR
echo "Current working directory:"; pwd

g09 h2o.gjf
```

- Batch script is always executed in user's `$HOME` directory on worker node, to change directory to script input director environment variable `$PBS_O_WORKDIR` could be used

PBS – environmental variables

- PBS adds environmental variables which could ease performing computation

Variable	Description
`PBS_JOBID`	job identifier (jobID)
`PBS_O_WORKDIR`	dir, from which batch script was submitted
`PBS_NP`	amount of computing cores requested by job
`TMPDIR`	local scratch file directory temporary dir for job
`SCRATCH`	scratch directory on distributed Lustre file system

- Additionally, when module `tools/scratch` used
  - `SCRATCHDIR` – distributed scratch file directory temporary dir for job

PBS – qsub command options

- qsub command uses various options to provide queuing system with additional info about the job
  - -q queue defines queue
  - -N name give name to job
  - -I informs that job is going to be not batch but interactive
  - -X enables X11 forwarding
  - -l gives information about requirements requested by job
  - -t n-m,k,l starts array job
  - -M <user's e-mail> email for notifications
  - -m bea information when notifications should be send: at beginning(b), end (e) or execution error (a)
  - -A grantID information about computational grant (if omitted job use default)

- When option -q is omitted job is putted into default queue

PBS – job requirements specification

```
#!/bin/env bash
##### Max amount of RAM requested by job
#PBS -l mem=1gb
##### Wall time requested by job
#PBS -l walltime=10:00
##### Queue name
#PBS -q l_prio
##### Name of job in queuing system
#PBS -N g09.ethanol

# Set environment for default Gaussian default version
module add plgrid/apps/gaussian
# Scratch directory for job
echo "Temporary files stored in" $GAUSS_SCRDIR
# Changing directory to one from which PBS script was stated
cd $PBS_O_WORKDIR
# Commands to start computations
g09 ethanol.gjf
# Deleting temporary files
rm -rf $GAUSS_SCRDIR
```

PBS – job requirements specification

- There are several recourses available for job (through option `-l`)
  - `walltime -` maximal execution wall time
  - `nodes=x:ppn=y` - amount of nodes and cores per node
  - `mem` – amount of memory requested by job
  - `pmem` – amount of memory per core requested by job

- Parameter values should be given in "parameter=value" notation, coma separated
  - i.e. `qsub -l walltime=10:00:00,nodes=1:ppn=12,mem=12gb`

- Parameter formats
  - time `hhh:mm:ss`
  - memory `b, kb (=1024b), mb (=1024kb), gb (=1,024mb)`
  - worker nodes `nodes=amount-of-nodes:ppn=cores-per-node:properties-of-node` (np. `nodes=2:ppn=12` – 2 nodes, 12 cores each)

PBS – job requirements specification

```
#!/bin/env bash
##### Max amount of RAM requested by job
#PBS -l mem=1gb
##### Amount of nodes=x:cores=y requested by job
#PBS -l nodes=1:ppn=12
##### Wall time requested by job
#PBS -l walltime=10:00
##### Queue name
#PBS -q l_prio
##### Name of job in queuing system
#PBS -N g09.ethanol

# Set environment for default Gaussian default version
module add plgrid/apps/gaussian
# Scratch directory for job
echo "Temporary files stored in" $GAUSS_SCRDIR
# Changing directory to one from which PBS script was stated
cd $PBS_O_WORKDIR
# Commands to start computations
g09 ethanol.gjf
# Deleting temporary files
rm -rf $GAUSS_SCRDIR
```

PBS – interactive jobs

- Interactive work on cluster should be done using interactive jobs
  - `qsub -I`
  - `qsub -I -X` when X11 forwarding is necessary
- Queue `l_interactive` is dedicated for interactive work
- When GUI is necessary user should remember about
  - login on cluster using X11 forwarding
  - launching X11 server on client side
- User interface **must not be used** for computing

PBS – deleting jobs `qdel`

- `qdel` command is used to delete unwanted jobs from queuing system
  - `qdel <JobID>`

- Information about jobs which cannot be deleted using qdel should be sended to system administrators through
  - Helpdesk PL-Grid PL
    - https://helpdesk.plgrid.pl
    - helpdesk@plgrid.pl
  - directly to system administrators zeus@cyfronet.pl

Job monitoring with `zeus-jobs*`

- `zeus-jobs` and `zeus-jobs-history` could be used to monitor efficiency of jobs
  - memory usage
  - CPU usage

- `zeus-jobs` – running and queued jobs
- `zeus-jobs-history` – historical data of completed jobs

- `zeus-jobs*` usage
  - `zeus-jobs -e-` or `zeus-jobs -e+` - jobs sorted according to efficiency
  - `zeus-jobs -w` – lists jobs only with low efficiency
  - `zeus-jobs -f (<jobID>)` – detailed information about jobs
  - `zeus-jobs -h` – help screen

PBS – array jobs

- Array jobs enable queuing several jobs using one `qsub` command
  - `qsub -t n-m,k,l script.pbs` (ie. `qsub -t 0-9` or `qsub -t 2,4,7`)
- All jobs within array have same `PBS_O_WORKDIR`, there are identified by additional variable `$PBS_ARRAYID`

```
#!/bin/env bash
#PBS -t 0-4,9
#PBS -l walltime=5:00
OUTPUTDIR=$PBS_O_WORKDIR/${PBS_JOBID%%\[*}
mkdir -p $OUTPUTDIR
cd $TMPDIR
hostname > job.$PBS_ARRAYID

mv job.$PBS_ARRAYID $OUTPUTDIR
```

- `qstat -t` – expands job arrays while listing jobs in queuing system

Best practices

- PBS job script is always started in user's `$HOME` directory on WN. Access to directory from which script was submitted via `PBS_O_WORKDIR`

- All batch jobs have got files in which data from standard outputs is stored
  - standard output stream (*stdout*): `name-of-script.o<JobID>`
  - standard error stream (*stderr*): `name-of-script.e<JobID>`
  - Those files should not be big (less than several MBs) and are accessible only after finishing the job

- When commands in PBS script print big amount of data into output streams user should redirect that data to file(s)
  - for standard output stream (*stdout*): `command > file.out`
  - for standard error stream (*stderr*): `command 2> file.err`
  - for both streams to one file: `command &> file.log`

Best practices

- During batch job submission user should always
  - specify maximal time of job exectution (parameter `walltime`)
  - do not use queue `l_infinite` if not necessary
  - specify maximal RAM amount needed by job through `mem` (or `pmem`)
  - enabling checkpoints
  - for parallel computations use all cores on nodes when possible via `l_exclusive`
  - when big amount of data is going to be passed to standard output streams redirect it to files
  - load runtime environment of software via `module` command in batch script
  - do not load software modules in scripts loaded at user's login (i.e. `.bashrc`)

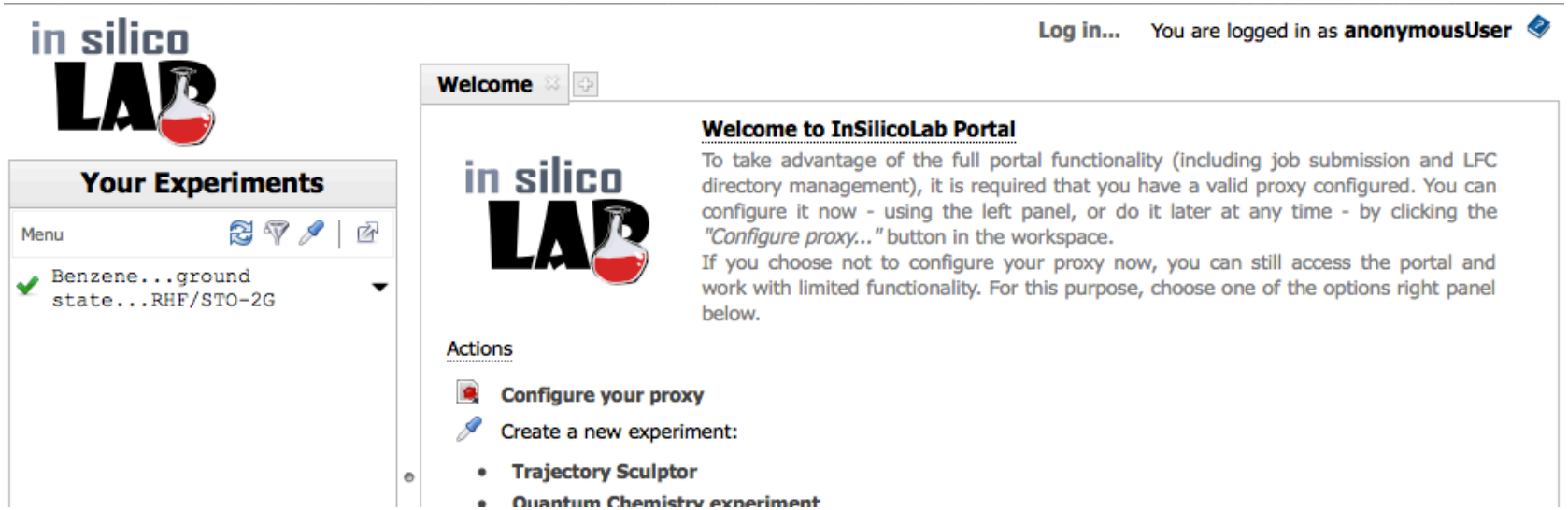
Best practices

- During batch job preparation user
  - **must not use** `$HOME` and `$STORAGE` for heavy I/O computations
  - should always use scratch file systems
    - local scratch disk attached to WN (accessible through `$TMPDIR`)
      - access to data only from WN to which disk is attached
      - big amount of very small in size I/O operations (<<1MB per read/write)
      - rather small files (up to 5 GB per core)
    - distributed scratch Lustre file system (`$SCRATCH` and `$SCRATCHDIR`)
      - necessity of access to scratch data from multiple WNs
      - big or huge temporary files (10+ GB)
      - big I/O operations (1+ MB)
      - easy access to temporary files during computation from UI node
    - **clean up** temporary files and directories after computations

InSilicoLab

http://insilicolab.grid.cyfronet.pl

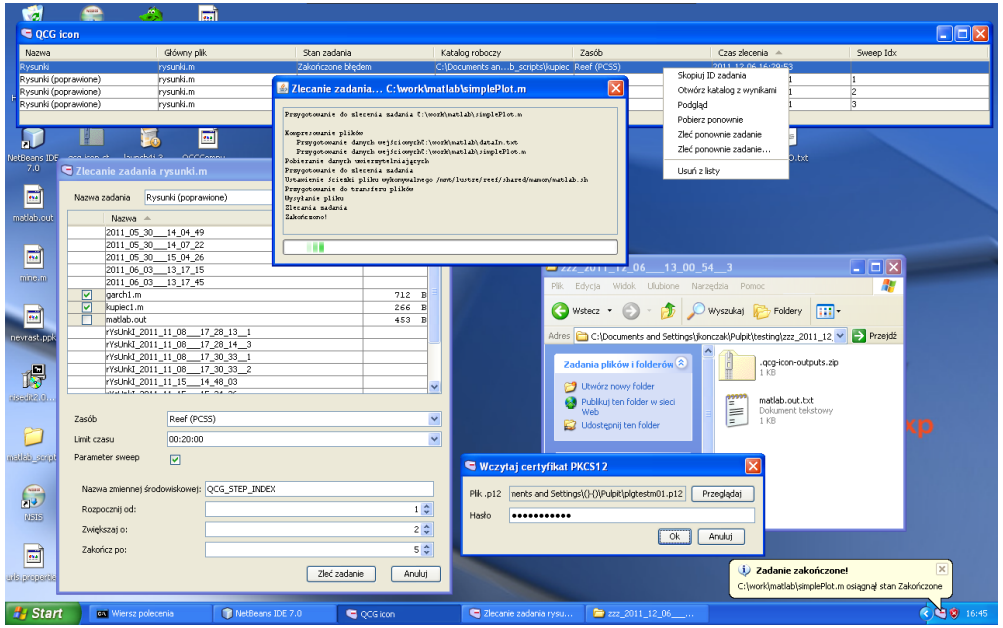
- Complete workspace in web browser for a scientists who wishes to perfor chemistry calculations
- Assist with
  - preparing input to various scientific applications
  - performing computations on grid infrastructure
  - controlling complex and recurrent jobs
  - collecting output files
  - analysis of obtained results (also from many calculations at once)



QCG-Icon

http://www.qoscosgrid.org/trac/qcg-icon

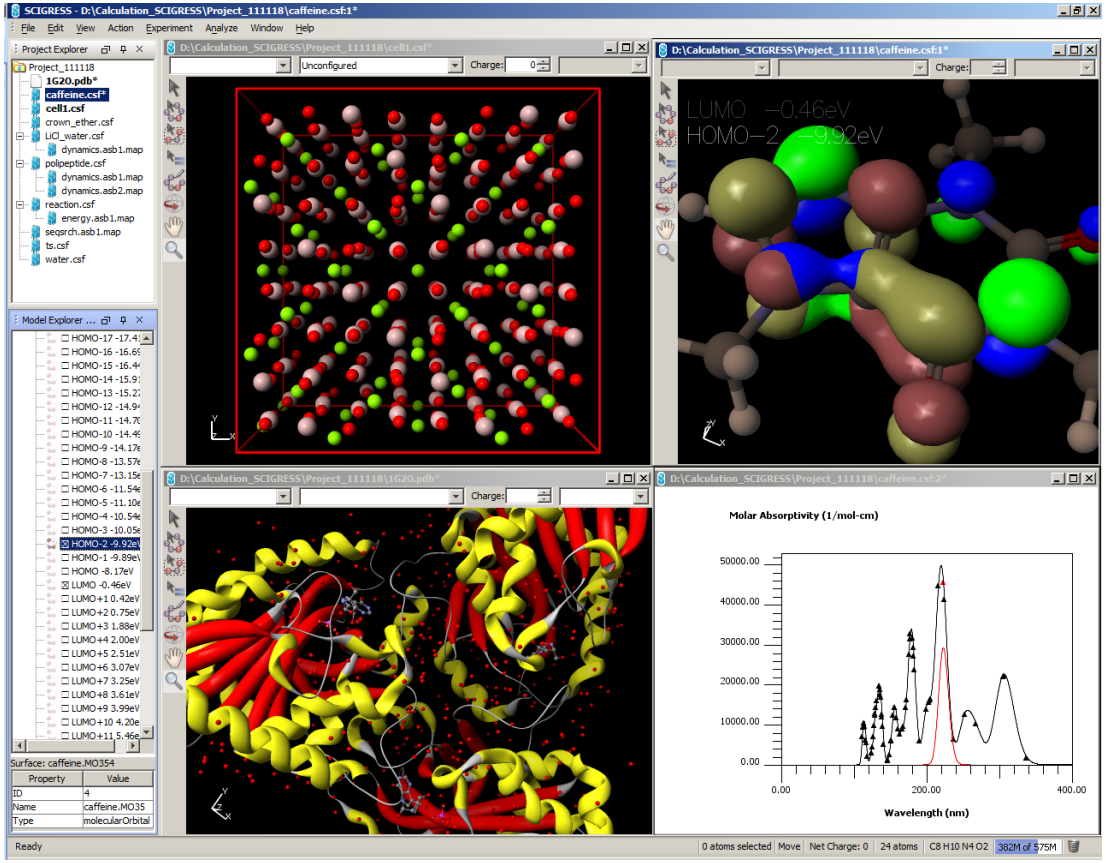
- Lightweight desktop client application (Windows, MacOSX, Linux)
  - runs quantum chemistry calculations from desktop to grids
    - preview of partial results
    - templates for multiple QC packages including: Gaussian, GAMESS, Molpro, Dalton, Turbomole, ADF, CRYSTAL09
    - other software could be run trough custom scripts
  - Integration with GaussView

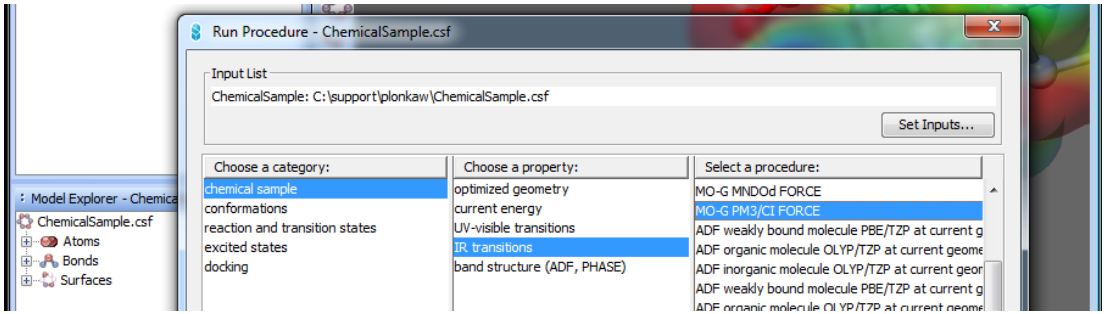


SCIGRESS

- Molecular modeling suite for Windows, Mac and Linux
- Intuitive GUI, predefined computational procedures
- Theory levels from MM to DFT
- Interface to ADF, GAMESS, GAUSSIAN, MOPAC 2012
- Quantum docking
- Batch processing
- Remote calculations (PLGrid, Amazon Cloud, others)

- More information:
  - **http://www.scigress.com**
  - **chemistry@fqs.pl**





Rejestracja: https://portal.plgrid.pl

helpdesk@plgrid.pl

+48 12 632 33 55 wew. 312