Efektywne wykorzystanie Komputerów Dużej Mocy w chemii obliczeniowej

Klemens Noga

ACK Cyfronet AGH

Wydział Chemii Uniwersytetu Jagiellońskiego, Kraków, 30 X 2015

Efficient usage of HPC clusters in computational chemistry

Klemens Noga

ACC Cyfronet AGH

Jagiellonian University, Chemistry Department, Kraków, 30 X 2015

Agenda

- **Access to HPC clusters**
- **Performing calculations**
  - Zeus cluster
    - queuing systems
    - best practices
  - Prometheus
- **Documentation and user support**

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@login01.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/

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)
    - vSMP – big virtual machines (each consist several ordinary work nodes)
    - GPGPU – nodes with GPGPU (44 nodes, 208 GPGPU cards)

Property	Zeus	Zeus BigMem	Zeus vSMP	Zeus GPGPU
CPU frequency	2.26-2.67 GHz	2.67; 2.30 GHz	2.67 GHz	2.93; 2.40 GHz
RAM	16, 24 GB	96, 256 GB	up to 6 TB	72, 96 GB
cores per node	8, 12	12, 64	up to 768	12
InafiniBand interconnect	available, QDR	available, QDR	–	available, QDR
additional		–	RAMDisk	GPGPU cards

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 GPPGU (2x nVidia Tesla K40)

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

Zeus – file systems

- Storage of data – access through NFS (quite slow, should not be used for heavy I/O calculations)
    - `$HOME` – user's home directory
        - quota 7 GB
        - daily backup
    - `$STORAGE` – for long lasting storage of data
        - quota 100 GB
    - `$PLG_GROUPS_STORAGE` – additional storage gained through PLGrid grants system
- Temporary scratch file systems
    - `$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
    - `$SCRATCH` – distributed scratch Lustre file system
        - accessible from all nodes of cluster (including UI)
- To check qouta use `zeus-fs`

Prometheus – file systems

- Storage of data – access through NFS (quite slow, should not be used for heavy I/O calculations)
    - `$STORAGE` – for storage of data
        - quota 40 GB
    - `$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)
- To check qouta use `pro-fs`

Software

- Scientific software usually needs specific runtime environment (i.e. additional libraries) and sometimes technical knowledge is needed to install them efficiently

- Modules package is solution 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

- 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 tools` (only from `tools` branch)
  - `module avail plgrid/apps/gaussian` (all available Gaussian versions)
- Listing of loaded modules
  - `module list`
- Clearing all loaded modules
  - `module purge`

Software

- 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

- Examples:
  - `plgrid/tools/intel/14.0`
  - `plgrid/apps/gaussian/g09.D.01`
  - `plgrid/tools/python/2.7.5`
  - `plgrid/apps/terachem/1.5`

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
    - PBS:
        - Torque (http://www.adaptivecomputing.com/products/open-source/torque/)
        - PBS Pro (http://www.pbsworks.com/product.aspx?id=1)
    - SLURM (http://slurm.schedmd.com)

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

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.m >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*
vsmp	-	vSMP nodes*

`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 [opcje PBS]`
  - 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 dead, zombie 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

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

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

SLURM - example ADF job

```
#SBATCH -J adf.ethanol
#SBATCH -N 1
#SBATCH --ntasks-per-node 1
#SBATCH --mail-type=ALL
#SBATCH --mail-user=k.noga@cyfronet.pl
#SBATCH --time=10:00
#SBATCH --mem 24000
#SBATCH -p plgrid

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 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 -e-` or `zeus-jobs -e+` - jobs sorted according to efficiency
  - `pro-jobs -w` – lists jobs only with low efficiency
  - `pro-jobs -f (<jobID>)` – detailed information about jobs
  - `pro-jobs -h` – help screen
- In addition `scontrol` and `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
  - `sinfo` – lists all available nodes

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

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

Rejestracja: https://portal.plgrid.pl

helpdesk@plgrid.pl

+48 12 632 33 55 wew. 312