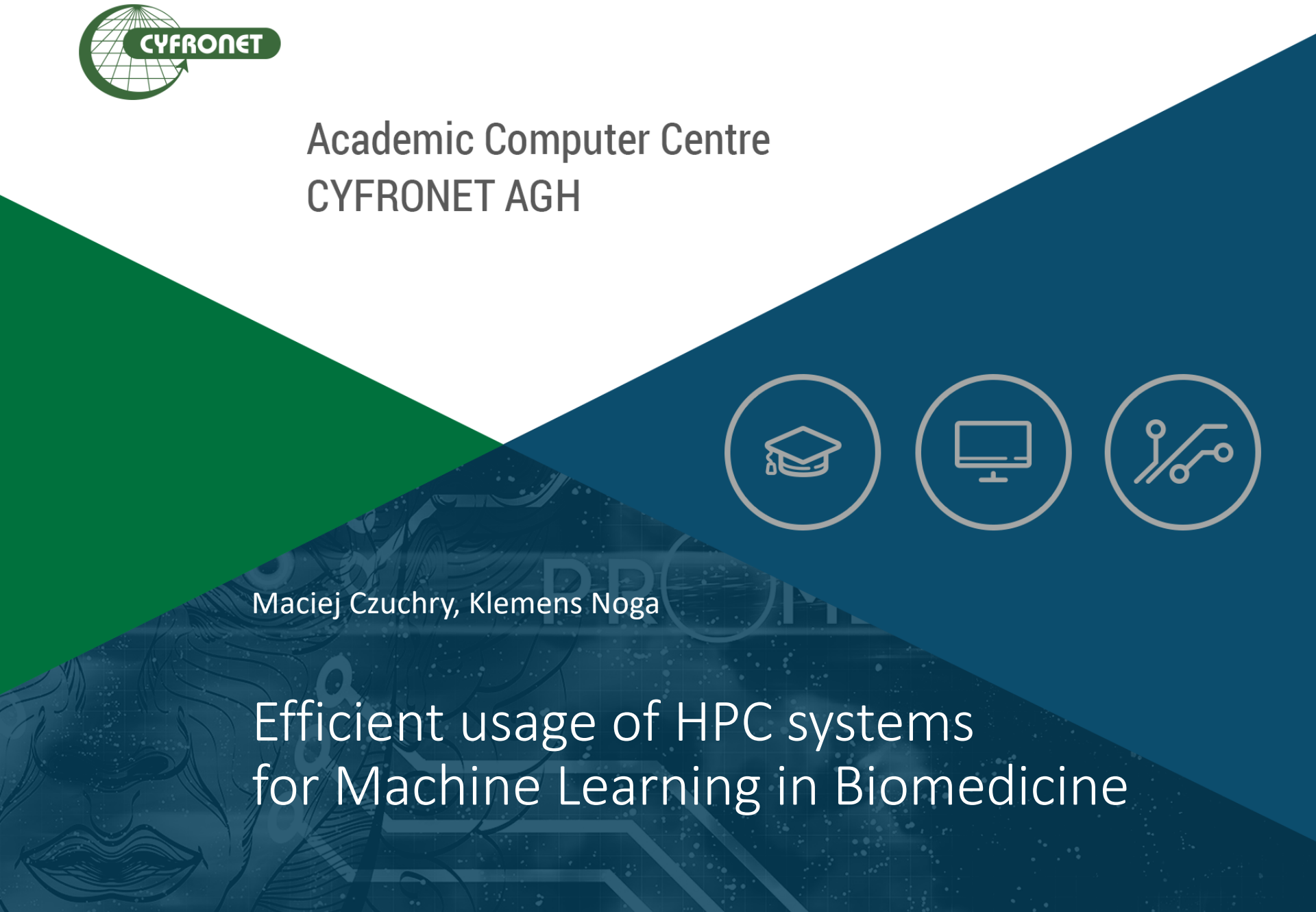


Academic Computer Centre
CYFRONET AGH

Maciej Czuchry, Klemens Noga

Efficient usage of HPC systems for Machine Learning in Biomedicine

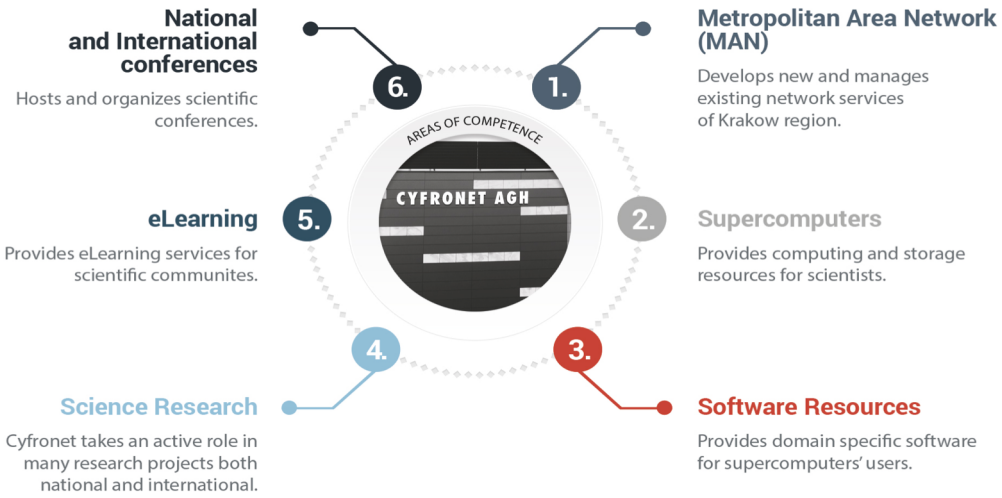
Agenda

- **Prometheus & Zeus clusters at ACC Cyfronet AGH**
  - available resources
  - access to clusters/data transfer
- **Performing calculations**
    - batch scripts
      - sequential and parallel runs
      - software environment management using Modules/Lmod
    - best practices
- **Documentation and users support**
- **Questions and exercises**
- **Zeus & Prometheus as a part of PLGrid Infrastructure**
- **Access to PRACE resources**

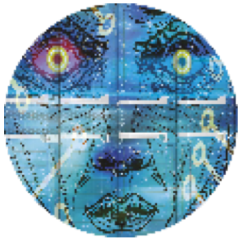

CYFRONET

Academic Computer Centre Cyfronet AGH

- The biggest Polish Academic **Computer Centre**
  - **45+ years of experience** in IT provision
  - Centre of excellence in **HPC, Grid and Cloud Computing**
  - Home for **Prometheus** and **Zeus supercomputers**
  - **LUMI** consortium partner (EuroHPC pre-exascale supercomputer)
- Legal status: an **autonomous** within AGH University of Science and Technology
- Staff: >150 , ca. 60 in R&D
- Leader of **PLGrid**: Polish Grid and Cloud Infrastructure for Science
- NGI Coordination in **EGI e-Infrastructure**



Prometheus



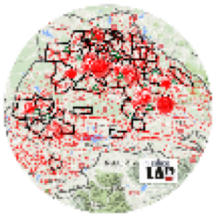
- 2.4 PFLOPS
- 53 568 cores
- 1st HPC system in Poland (174th on Top 500, 38th in 2015)

Zeus

- 374 TFLOPS
- 25 468 cores
- 1st HPC system in Poland (from 2009 to 2015, highest rank on Top500 – 81st in 2011)

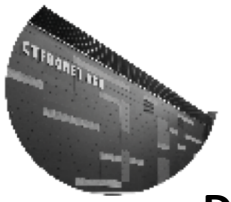
Storage

- 60+ PB
- hierarchical data management



Computing portals and frameworks

- OneData
- PLG-Data
- DataNet
- Rimrock
- InSilicoLab



Research & Development

- distributed computing environments
- computing acceleration
- machine learning
- software development & optimization

Computational Cloud

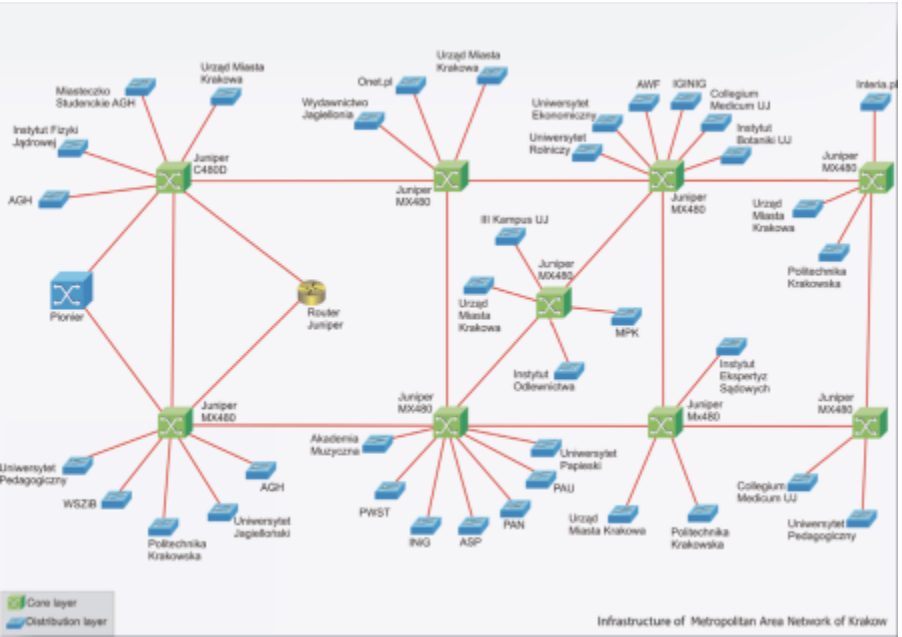
- based on OpenStack

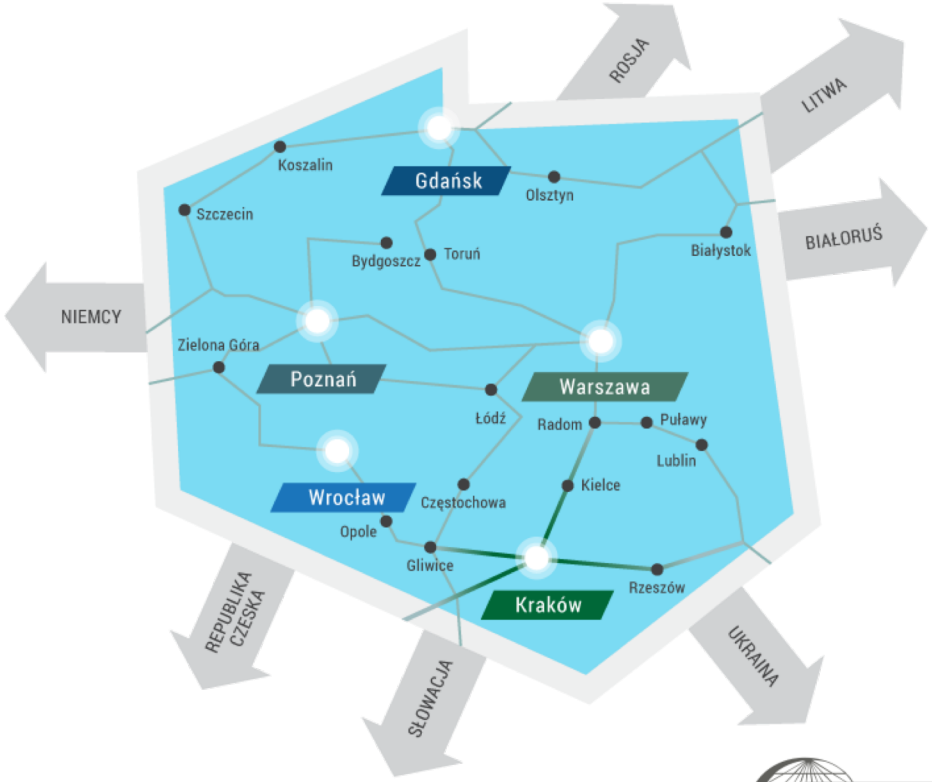
Data Centres

- 3 independent data centres
- dedicated backbone links

Network backbone lines

- 4 main links to achieve maximum reliability
- Each link with 7x10Gbps capacity
- Additional 2x100Gbps dedicated links
- Direct connection with GEANT scientific network





The fastest supercomputer in Poland:

- Installed in Q2 2015 (upgraded in Q4 2015)
- Centos 7 + SLURM
- HP Apollo 8000 – direct liquid cooling – PUE 1.06
  - 20 racks (4 CDU, 16 compute)
- 2232 nodes, 53 568 CPU cores (Haswell, Xeon E5-2680v3 12C 2.5GHz), 279 TB RAM
  - 2160 regular nodes (2 CPUs, 128 GB RAM)
  - 72 nodes with GPGPUs (2x NVIDIA Tesla K40 XL)
  - 4 islands
- 2.4 PFLOPS total performance (Rpeak)
- <850 kW power (including cooling)
- **TOP500**: current 174th position, highest: 38th (XI 2015)

- Supercomputers from Poland
  - 131 – Prometheus (ACC Cyfronet AGH) – 2.4 Pflops (PLGrid)
  - 414 – Eagle/Orzeł (PSNC) – 1.37 PFLOPS (PLGrid)
  - 419 – Tryton (CI TASK) – 1.41 PFLOPS (PLGrid)
  - 478 – Okeanos (ICM) – 1.08 PFLOPS

TOP500 The List.

CYFRONET

- Supercomputers from Poland
  - 174 – Prometheus (ACC Cyfronet AGH) – 2.4 Pflops (PLGrid)

TOP500 The List.

CYFRONET

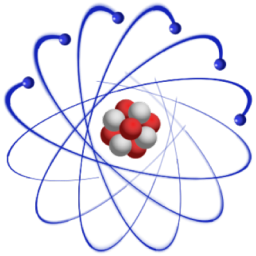
Prometheus HPC cluster

- 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)
    - 3 big memory nodes (2x Intel Xeon Gold 6128, 12 x 3.4 GHz, 768 or 1536 GB)
    - 4 nodes with additional GPPGU (8x nVidia Tesla V100)

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

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
  - SLURM on Prometheus & Zeus

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 tutorial

- Access credentials
  - tutorial's accounts @Prometheus cluster:
    - user: **tutorialXX (where XX=01-40)**
    - password: on leaflet
    - host (UI): login@prometheus.cyfronet.pl (login@pro.cyfronet.pl)

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`

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/r`)
  - `module load <module-name>` (i.e. `module load plgrid/apps/matlab`)
- Remove module
  - `module rm <module-name>` (i.e. `module rm plgrid/apps/r`)
  - `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/r` (all available R versions in `plgrid/apps`)
  - `module spider python` (all available Python versions)
  - `module spider "/r/"` (all available R versions, regexp search)
- Listing of loaded modules
  - `module list`

Software – modules

- Clearing all loaded modules
  - `module purge`
- Saving collection of modules for later use, restoring it and listing saved collections
  - `module save [collection]`
  - `module restore [collection]`
  - `module savelist`
  - `module describe [collection]`
  - `ml` is shorthand for `module` command
    - `ml = module list`
    - `ml <module-name> = module load <module-name>`
    - `ml -<module-name> = module unload <module-name>`
    - `ml av <string> = module avail <string>`
  - Getting help
    - `module help`
    - `ml -h`

Software – modules

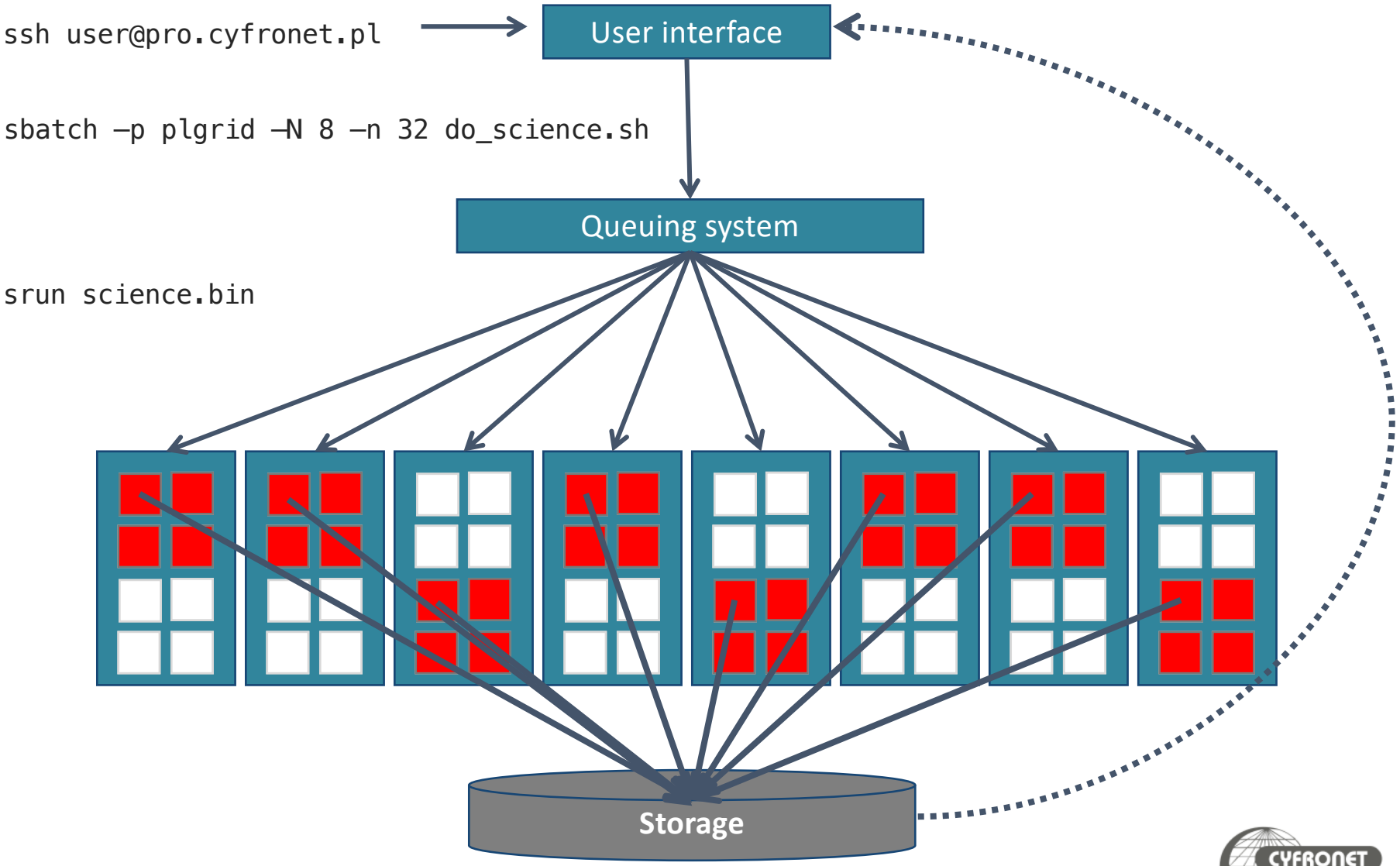
- Each software package installed in PLGrid infrastructure has it's own module
  - `plgrid/<branch>/<software-name>/<version>`
- Branch kinds
  - `apps` – for most of scientific packages
  - `libs` – for software libraries
  - `tools` – for toolkits and helper packages
- User's own modules
  - `module use path` – adds `path` with additional modules
- Examples:
  - `plgrid/tools/intel/19.0.5`
  - `plgrid/apps/r/3.6.0`
  - `plgrid/tools/python/3.6.5`

https://apps.plgrid.pl/

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)


ssh user@pro.cyfronet.pl
User interface
sbatch –p plgrid –N 8 –n 32 do_science.sh
Queuing system
srun science.bin
Storage

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 Pro (http://pbspro.org)

HPC Centre	Cluster	Queuing system
ACC Cyfronet AGH	Prometheus	SLURM
	Zeus	SLURM
ICM	Topola	SLURM
PSNC	Eagle	SLURM
TASK	Tryton	SLURM
WCSS	Bem	PBS Pro

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
  - `CF` – configuring (resources for job are being prepared)
- 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

CYFRONET

Available partitions

Partitions	max time	Information
plgrid-testing	1:00:00	for test runs (small number of jobs)
plgrid-short	1:00:00	
plgrid	3-00:00:00	
plgrid-now	12:00:00	interactive runs, max one job on one node
plgrid-long	7-00:00:00	*
plgrid-gpu	3-00:00:00	nodes with GPGPU*
plgrid-gpu	3-00:00:00	nodes with V100 GPGPU*
plgrid-bigmem	3-00:00:00	big mem nodes*

- In SLURM queues are called partitions
- `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 Python job


```
#!/bin/env bash

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

module add plgrid/tools/python

./python-script.py > python.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=<ntasks>` amount of tasks invoked in whole 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 serial job


```
#SBATCH --job-name=serial.job
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=10:00
#SBATCH --mem=24000
#SBATCH --partition=plgrid
#SBATCH --account=tutorial

module add plgrid/tools/intel

icc -xHost hello.c -o hello.x

./hello.x
```

- 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 job


```
#SBATCH --job-name=parallel-srun
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=1GB
#SBATCH --partition=plgrid
#SBATCH --account=tutorial

module add plgrid/tools/intel

icc -xHost hello.c -o hello.x

srun ./hello.x
```

- `srun` inside batch job executes command `./hello.x` on allocated resources according to `requested --ntask` or `--nodes*--ntasks-per-node` flags
  - variable SLURM_NTASKS holds information about number of tasks to be run
- each `srun` could request more than one core
  - `srun –nodes=x --ntasks=y --cpus-per-task=z …`

SLURM – example parallel OpenMP job


```
#SBATCH --job-name=parallel-openmp
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=24
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=2GB
#SBATCH --partition=plgrid
#SBATCH --account=tutorial

module add plgrid/tools/intel

icc -xHost -qopenmp hello.c -o hello.x

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

./hello.x
```

- When use OpenMP
  - 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 – example parallel MPI job


```
#SBATCH --job-name=distributed-mpi
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=1GB
#SBATCH --partition=plgrid
#SBATCH --account=tutorial

module add plgrid/tools/impi

mpiicc -xHost hello.c -o hello.x

mpiexec -np $SLURM_NTASKS ./hello.x
```

- When software 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 hybrid MPI/OpenMP job


```
#SBATCH --job-name=mpi-openmp
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=6
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=2GB
#SBATCH --partition=plgrid
#SBATCH --account=tutorial

module add plgrid/tools/impi

mpiicc -xHost -qopenmp hello.c -o hello.x

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

mpiexec -np $SLURM_NTASKS ./hello.x
```

- When hybrid MPI/OpenMP
  - use `--cpus-per-task=<cores_per_job>` and `$SLURM_CPUS_PER_TASK` for distribution of threads
  - use `--ntasks-per-node=<cores_per_node>` for request of MPI processes
  -

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
- High priority queue `plgrid-now` for interactive work
  - one job on one node up to 12:00:00
- 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]`

Python

`plgrid/tools/python or plgrid/tools/python-intel`

- Versions available at Prometheus cluster
  - GNU:
    - `2.7.14, 3.6.5`
    - `plgrid/tools/python`
  - Intel:
    - `2.7.12, 2.7.13, 2.7.14, 3.5.2, 3.5.3, 3.6.2, 3.6.5`
    - `plgrid/tools/python-intel`
  - Special modules for `tensorflow, scipy, numpy, mpi4py`

- Usage
  - `module add plgrid/tools/python/<version>`
  - `module add plgrid/tools/python-intel/<version>`

- Remarks
  - Build with MKL numerical libraries, support for GPGPU computing
  - `python-intel` use `conda` package manager

- Usage restrictions
  - only SMP mode (up to 24 computing cores@Prometheus)
    - multi-node MPI only with libs such as py4mpi , dask i.e.

Jupyer notebook @Prometheus

- On HPC clusters computations are executed on worker nodes
  - usually no GUI
  - usually behind firewall
- Jupyter notebook/ Jupyterlab needs GUI in web browser

- Therefore there is a need to port tunnelling
  - First submit job which create jupyter notebook and tunnel from worker node to login node
  - Establish tunnel from your computer to login node of cluster
  - Open notebook in your favourite browser on your computer

Jupyer notebook @Prometheus

- SLURM job create
  - jupyter notebook
  - tunnel from worker node to login node

```
#!/bin/bash
#SBATCH --partition plgrid-testing

## get tunneling info
XDG_RUNTIME_DIR=""
ipnport=$(shuf -i8000-9999 -n1)
ipnip=$(hostname -i)
user=$USER

module load plgrid/tools/python/3.6.5

## start an ipcluster instance and launch jupyter server
jupyter-notebook --no-browser --port=$ipnport --ip=$ipnip pyton-notebook.slurm
```

Jupyer notebook @Prometheus

- User has to create second tunnel form user's computer to login node of Prometheus (pro.cyfronet.pl)
  - `ssh -N -L <local-port>:<worker-node-ip>:<remote-port> plgusername@pro.cyfronet.pl`
  - info about tunnel details `<local-port>`, `<worker-node-ip>`, `<remote-port>` are in log file of SLURM job
  - `plgusername` – user's logname
- After establishing both tunnels jupyter notebook is ready to start
  - open webpage `localhost:<local-port>` in browser on your local clomputer
  - remember about token, which is listed in log file of SLURM job

plgrid/apps/R

- Versions available at Prometheus cluster
  - `3.2.1, 3.4.3, 3.4.4, 3.5.1, 3.6.0`
- Usage
  - `module add plgrid/apps/r/<version>`
  - `R`
- Remarks
  - big amount libraries installed
    - could be listed from R with function `installed.packages()`
- Usage restrictions
  - only SMP mode (up to 24 computing cores@Prometheus)
    - multi-node MPI only with libs such as Rmpi

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` (number of job in array)

```
#!/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

CYFRONET

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
  - `pro-jobs-history -d <period>` jobs completed in last <period> days

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`)

Computational grants

- Obtained through PLGrid Portal - https://bazaar.plgrid.pl/
  - distinct grants for GPGPU

- Commands
  - `plg-show-grants (pro-show-grants)`
  - `plg-show-grant-details <account> (pro-show-grant-details <account>)`
  - `plg-show-default-grant (pro-show-default-grant)`

- Accounting portal - https://accounting.plgrid.pl/

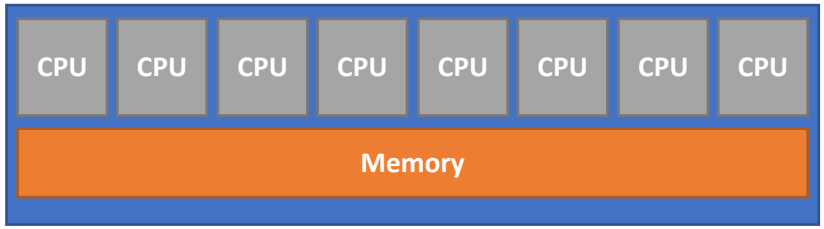
Additional „filesystems”

- MEMFS
  - `-C memfs`
  - `$MEMFS`
  - use memory as filesystem (120GB max)
    - Accessible only within node
  - available during JOB and lost after it finishes

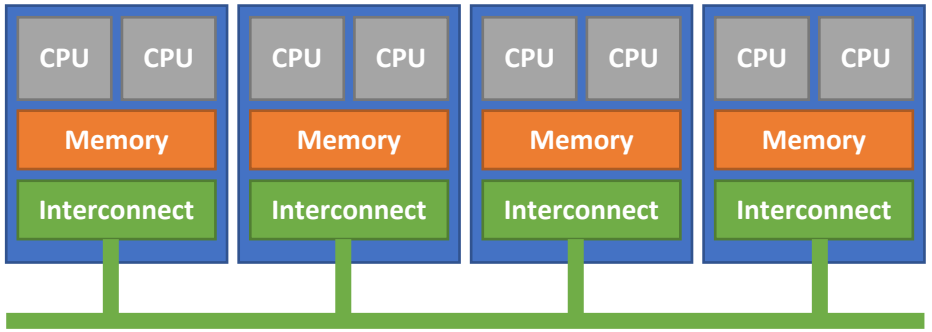
- LOCALFS
  - `-C localfs`
  - `$SCRATCH_LOCAL`
  - use file as filesystem (512GB per node)
  - Each node has its own file! (not a shared filesystem)
    - Accessible only within node
  - Available during JOB and lost after it is finished

- OpenMP
  - API for writing shared-memory software
  - Shared memory in threads
  - Requires support in compiler (-qopenmp, -fopenmp)
- MPI
  - Message Passing Interface (send, receive, broadcast)
  - Every process has its own isolated memory space
  - Can use more than one machine via interconnect (eth, openid)

SMP



Cluster



Python – package manager (pip)

- In Python you can easily manage your python modules via `pip` (`pip3`)
  - `pip list`
  - `pip search numpy`
  - `pip install numpy`
  - `pip install –U numpy`
  - `pip install –u numpy==1.10.1`
- The problem is when you need more than one environment
- Solution is virtualenv
  - `virtualenv --system-site-packages my_new_env`
  - `source my_new_env/bin/activate`
  - `pip install –U` whatever you want
  - `deactivate`
- One directory for each environment – clean project management

Creating own module

- Create directory for modules, eg. `$PLG_GROUPS_STORAGE/your-team-name/modules`
- Create there subdirs for modules, eg. `app/name`
- Create Lua file for lmod, eg. `1.0.lua`

```
local pkgName     = myModuleName()
local fullVersion = myModuleVersion()
whatis("Name: "..pkgName)
whatis("Version "..fullVersion)
whatis("Description: Abaqus")
local APPDIR = '/net/software/local/abaqus/2017'
depends_on('plgrid/tools/intel/18.0.0')
prepend_path('PATH', APPDIR .. '/bin')
prepend_path(,LD_LIBRARY_PATH', APPDIR .. '/lib')
```

- Set path for new module: `module use $PLG_GROUPS_STORAGE/your-team-name/modules`
- Load module with: `module load app/name/1.0`

- Pro-viz is a new service for users of Prometheus that allows running GUI mode of software using: TurboVNC https://www.turbovnc.org.
- To run TurboVNC you have to install Java JRE x86.
- At first step user need to run pro-viz on the cluster. To use it you need to load software module of pro-viz:

```
module load tools/pro-viz
```

Pro-Viz command syntax


```
      pro-viz ...
        start [-n CORES | -N NODES | -p PARTITION | -t TIME | -A
ACCOUNT | -r RESERVATION | -g GPUS | -C constraints | -m EMAIL-ADDRESS
] - start a new batch session
        interactive [ -p PARTITION | -t TIME | -A ACCOUNT | -r
RESERVATION | -g GPUS | -C constraints  ] - start a new interactive
session
        list - list all sessions
        attach JOBID - attach session to a working job with JOBID
        password JOBID - generate access token for session JOBID
        stop JOBID - terminate session JOBID
        killall - terminate all sessions
        help - duh
```

Pro-Viz command syntax

In this tutorial will be presented running one job on cluster Prometheus with 1 full working node, 24CPU. To do this you need to run commands:

- `module load tools/pro-viz`
- `pro-viz start -N 1 -n 24 -p plgrid -A tutorial -t 03:00:00`
- `pro-viz password JOBID`

NAMD

plgrid/apps/namd

- Versions available at Prometheus cluster:
  - `2.9, 2.10, 2.11, 2.12`
- Usage
  - `module add plgrid/apps/namd`
  - `namdrun $NAMD_SCRIPT`

- Remarks
  - Obliczenia przy wykorzystaniu kart GPGPU
  - symulacje MD

- Przykładowe obliczenia
  - 390k atomów, układ periodyczny, 1000 kroków symulacji, 15 wezłów obliczenowych, 360 rdzeni, 3 doby

Gromacs

plgrid/apps/gromacs

- Versions available at Prometheus cluster:
  - `4.6.x, 5.0.x, 5.1.x, 2016, 2018`
- Usage
  - `module add plgrid/apps/gromacs`
  - `$MDRUN –s <input.tpr> –deffnm <input> –nb cpu –pin on`
- Przykładowe użycie
  - 55k atomów, białko w wodzie, 248 węzłów, 5952 rdzenie, 30 ns symualcji czas: 30 godzin – średnia wydajność 31 ns/dobę.

PLGrid Infrastructure

Computing

- 5+ PTFLOPS
- 130 000+ cores

Scientific software

- 750+ apps, tools, libraries
- apps.plgrid.pl

PL GRID

Storage

- 70+ PB
- archives
- backups
- distributed access
- fast scratch filesystems

Team work utilities

- project management (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)

PRACE | members

Hosting Members

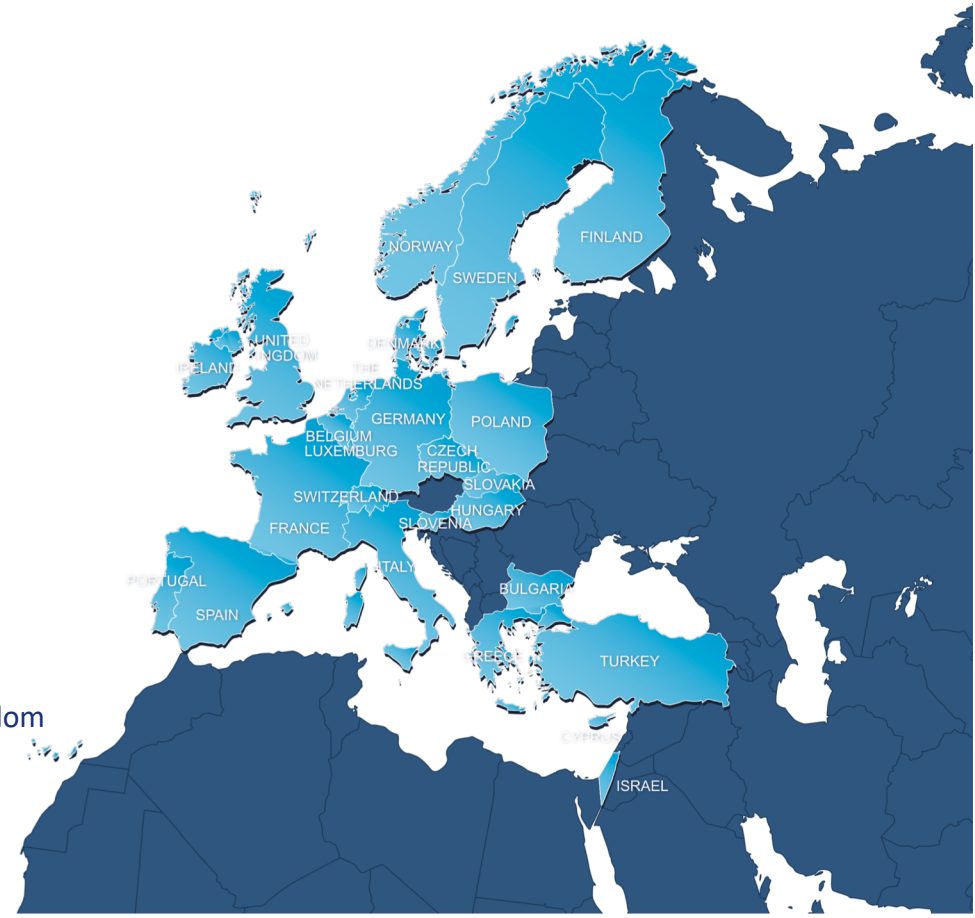
- France
- Germany
- Italy
- Spain
- Switzerland

General Partners (PRACE 2)

- Belgium
- Bulgaria
- Cyprus
- Czech Republic
- Denmark
- Finland
- Greece
- Hungary
- Ireland
- Israel
- Luxembourg
- Netherlands
- Norway
- Poland
- Portugal
- Slovakia
- Slovenia
- Sweden
- Turkey
- United Kingdom

Observers

- Croatia
- Romania



PRACE | what we do

- Open access to world-class HPC systems to EU scientists and researchers
- Variety of architectures to support the different scientific communities
- High standards in computational science and engineering
- Peer Review at European level to foster scientific excellence
- Robust and persistent funding scheme for HPC supported by national governments and European Commission (EC)
- Support the development of intellectual property rights (IPR) in Europe by working with industry and public services
- Collaborate with European HPC industrial users and suppliers

CYFRONET

PRACE | Tier-0 Systems

MareNostrum: IBM
BSC, Barcelona, Spain
#29 Top 500

Piz Daint: Cray XC50
CSCS, Lugano, Switzerland
#6 Top 500

SuperMUC-NG: Lenovo ThinkSystem
GAUSS @ LRZ, Garching, Germany
#9 Top 500

Joliot Curie: BULL Sequana X1000
GENCI/CEA, Bruyères-le-Châtel, France
#47 Top 500

MARCONI: Lenovo
CINECA, Bologna, Italy
#21 Top 500

JUWELS: BULL Sequana X1000
GAUSS @ FZJ, Jülich, Germany
#30 Top 500

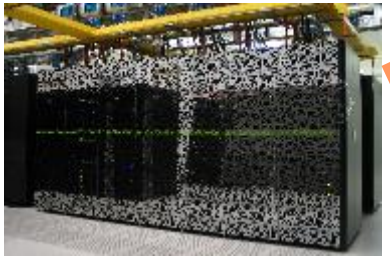
PRACE | Tier-1 Systems

ARCHER: Cray XC30
EPCC, Edinburgh, UK
#252 Top 500

Prometheus: HPE Apollo 8000
ACC Cyfronet AGH-UST, Krakow, Poland
#174 Top 500

Beskow: Cray XC40
KTH, Stockholm, Sweden
#151 Top 500

Salomon: SGI ICE X
IT4I, Ostrava, Czech Republic
#282 Top 500



Cartesius: Bull Bullx B720/B710
SURFSara, Amsterdam, The Netherlands
#455 Top 500

New

Puhti: BullSequana X400
CSC, Espoo, Finland

PRACE | project access

- Free-of-charge required to publish results at the end of the award period
- Preparatory Access (2 to 6 months)
- Project Access (12, 24 or 36 months)
- SHAPE Programme (2 to 6 months)
- Distributed European Computing Initiative (Tier-1 12 months)

Criterion: Scientific Excellence Assessed by an improved review process

www.prace-ri.eu/call-announcements/

CYFRONET

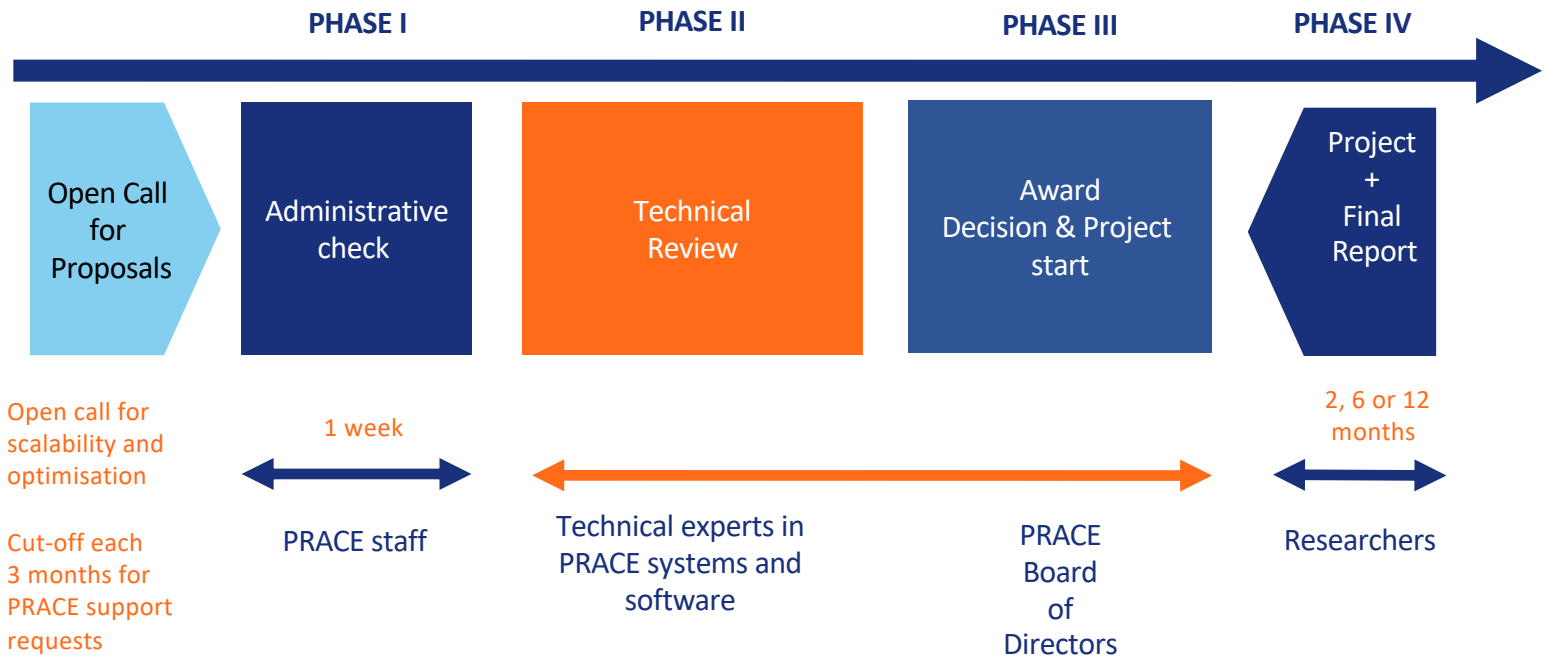
PRACE | project access

http://www.prace-ri.eu/prace-project-access/

PRACE | project access

- 20th Call for Proposals for Project Access
  - Opening of the call: 20 September 2019
  - Closing of the call: 29 October 2019, 10:00 CET
  - Allocation period for awarded proposals: April 2020 – March 2021
  - Type of Access: Project Access and Multi-Year Project Access
- Applications for Project Access must use codes that have been previously tested and
  - demonstrate high scalability and optimisation to multi-core architectures
  - demonstrate a requirement for ensemble simulations that need a very large amount of CPU/GPU

PRACE | preparatory access



http://www.prace-ri.eu/prace-preparatory-access/

PRACE | Distributed European Computing Initiative

- 15th Call for Proposals for DECI (Tier-1)
  - Opening of the call: 15 January 2019
  - Closing of the call: 28 February 2019, 17:00 UTC
  - Allocation period for awarded proposals: June 2019 – May 2020
  - Type of Access: DECI (Tier-1)
- Applications for DECI:
  - projects requiring access to Tier-1 resources that are not currently available in PI's own country or for international collaborations
  - individual projects limited to around 5 million machine hours (2.5 million machine hours in average)

PRACE | Training and Outreach activities

provide a sustained, high-quality training and education service for the European HPC community

- 6 PRACE Advanced Training Centres (PATCs) and 4 Training Centres (PTCs)
- PRACE training events: Seasonal Schools, International HPC Summer School, On-demand training events
- Summer of HPC (programme for undergraduate and postgraduate students)
- PRACE Training and Events portal
- CodeVault, Massive Open Online Courses (MOOCs)

Training topics

Different levels of training

- Basic, intermediate, advance

High performance computing

- Parallel programming
- Accelerators
- Performance optimization

Domain-specific topics

- Simulation software
- Visualization
- Data intensive computing

PRACE | Training and Events Portal

- www.training.prace-ri.eu
- Single hub for the PRACE training events, training material and tutorials
- PATC Programme 2019-2020
  - 79 courses, 215 training days
  - New courses on forward-looking topics
  - New hardware and programming paradigms
  - Data science
  - Collaboration with CoEs on several courses

Ministry of Science and Higher Education
Republic of Poland

"Prace realizowane przy wsparciu Ministerstwa Nauki i Szkolnictwa Wyższego, decyzja nr DIR/WK/2016/18"

CYFRONET

CYFRONET

AGH