



Wprowadzenie do obliczeń naukowych na Komputerach Dużej Mocy

Klemens Noga

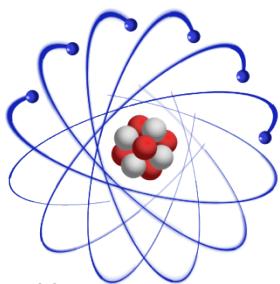
ACK Cyfronet AGH

Wydział Chemiczny Uniwersytetu Jagiellońskiego, Kraków 2 XII 2016

- Klastry obliczeniowe w ACK Cyfronet AGH
 - sposoby dostępu
 - praca z tekstowym interfejsem użytkownika
 - dostępne zasoby obliczeniowe i dyskowe
 - zarządzanie środowiskiem aplikacji obliczeniowych
- Przeprowadzenie obliczeń
 - systemy kolejkowe
 - skrypty obliczeniowe
 - obliczenia sekwencyjne i równoległe
 - najlepsze praktyki
- Dokumentacja i pomoc dla użytkowników

- Wszystkie klastry obliczeniowe w PLGrid wykorzystują Linux

- Scientific Linux 6 na Zeusie
 - CentOS 7 na Prometheusie



- Klastry obliczeniowe zawierają

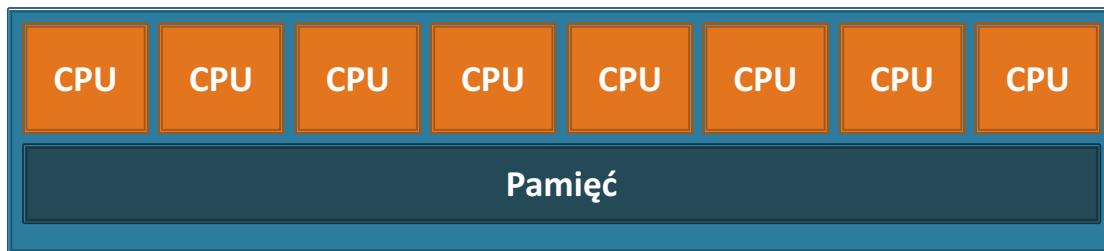
- węzeł dostępowy (user interface (UI))
 - węzły obliczeniowe

- Przeprowadzanie obliczeń na węźle dostępowym jest zabronione

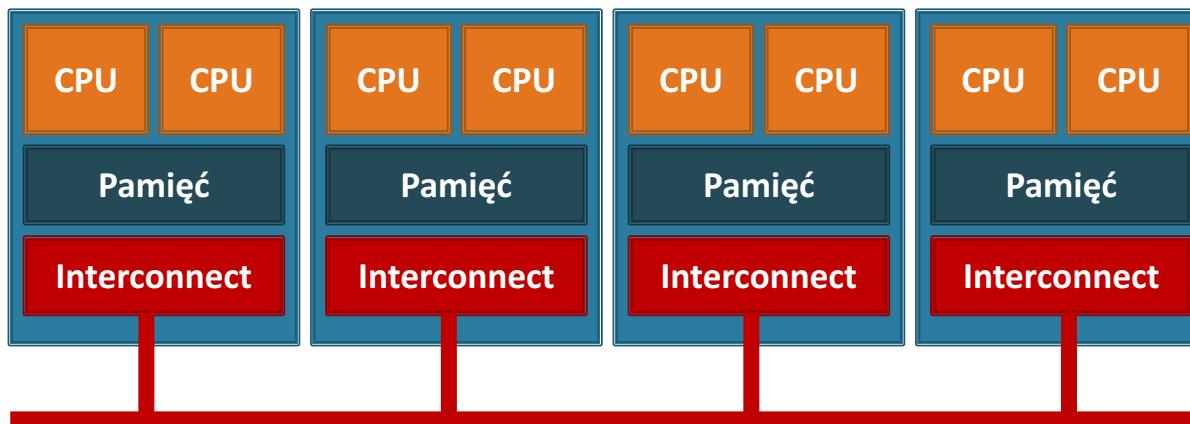
- Sprawiedliwym dostępem do zasobów zajmuje się system kolejkowy

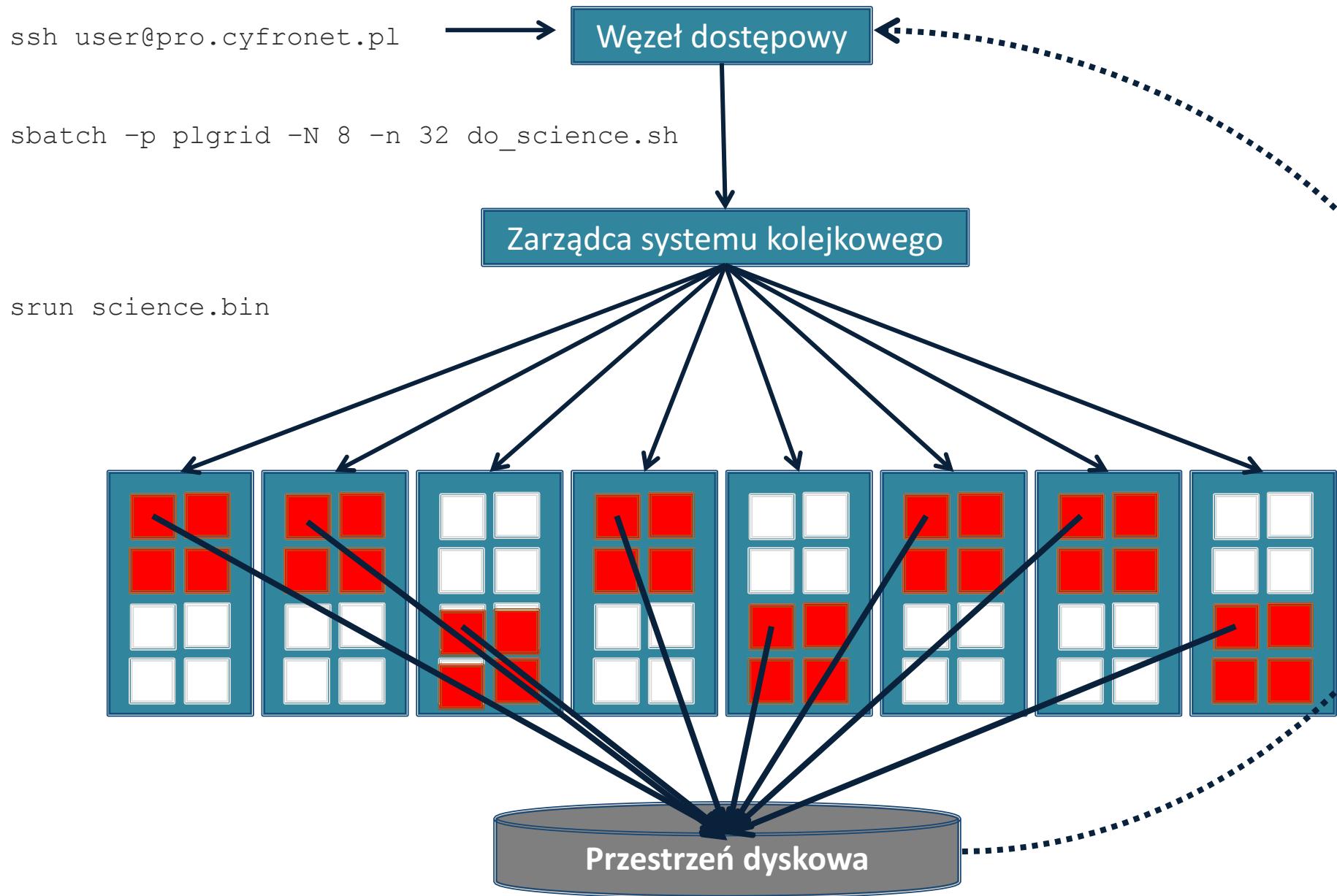
- PBS/Torque/Moab on Zeus
 - SLURM on Prometheus

SMP



Klaster





- Użytkownik loguje się na węzeł dostępowy (user interface (UI)) wykorzystując protokół SSH
 - nazwy węzłów dostępowych:
 - login@zeus.cyfronet.pl
 - login@pro.cyfronet.pl
 - programy do łączenia się po SSH
 - Linux oraz MacOS zawarte w systemie operacyjnym
 - polecenie **ssh** w terminalu
 - 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>
 - kopiowanie plików i katalogów
 - Linux oraz MacOS zawarte w systemie operacyjnym
 - polecenie **scp** w terminalu
 - Windows
 - WinSCP - <http://winscp.net/>

- Część systemu operacyjnego zajmującego się przechowywaniem danych nazywamy systemem plików (file system). Jego zadaniem jest organizacja danych w pliki oraz katalogi
- Ciekawostka: w Linuksie wszystko jest plikiem (katalogi, urządzenia, powłoka)
- By dowiedzieć się gdzie jesteśmy w systemie plików wykonajmy polecenie
 - `pwd`
- Polecenie to wypisuje aktualny katalog roboczy (`pwd = print working directory`)
- Domyślnie powłoka po otworzeniu znajduje się w katalogu domowym użytkownika (`~/`, `$HOME`)
 - Linux/MacOS: `/home/nelle`, `/people/nelle`, `/Users/nelle`
 - Windows: `C:\Documents and Settings\nelle`, `C:\Users\nelle`
- Cały system plików zaczyna się od korzenia (tzw. root directory) `/`

- Wyświetlanie zawartości katalogów
 - `ls [opcje] [argumenty]`
- Używając flag/opcji możemy modyfikować zachowanie programów
 - `ls -F`
- Przydatne opcje polecenia `ls`
 - `-a` – wyświetla wszystkie pliki, w tym ukryte
 - `-t` – sortuje wyświetlanie według czasu modyfikacji
 - `-r` – odwraca kolejność sortowania
 - `-l` – wyświetla dodatkowe informacje w tzw. długim formacie
- Używając argumentów możemy wylistować zawartość innego katalogu
 - `ls -F slurm`
- Każda komenda w systemie linux ma stronę pomocy dostępną przez polecenie `man`
 - `man ls`

- Listowanie zawartości katalogów
 - używając relatywnych ścieżek
 - ls -F slurm
 - używając absolutnych ścieżek
 - ls -F /net/people/tutorial/tutorial20/slurm
- Zmienianie katalogów – cd (change directory)
 - cd slurm
- Zmienianie katalogów
 - używając relatywnych ścieżek
 - cd tutorial20/slurm
 - cd ..
 - używając absolutnych ścieżek
 - cd /net/people/tutorial/tutorial20/slurm

- Przydatne skróty do katalogów
 - . / - bieżący katalog
 - . . / - katalog nadrzędny
 - ~ / - katalog domowy
 - / - katalog „root”, „korzeń” systemu plików
- Autouzupełnianie
 - po wpisaniu fragmentu nazwy polecenia, pliku lub katalogu naciskając klawisz Tab powłoka
 - uzupełnia nazwę gdy jest unikalna
 - podaje możliwe uzupełnienia nazwy
- Przydatne skróty klawiszowe
 - Ctrl + c przerywa działanie komendy/programu
 - Ctrl + r przeszukuje historie wydanych komend
 - ↑ oraz ↓ przechodzenie po historii użytych komend
 - Ctrl + l czyści terminal

Tworzenie nowych plików i katalogów

- Do utworzenia nowego katalogu w systemie plików służy polecenie `mkdir`
 - `mkdir new_directory`
 - `mkdir /Users/nelle/new_directory`
 - `mkdir -p /Users/nelle/new_directory/another_directory`
- Do tworzenia nowych plików tekstowych można użyć edytora tekstu, np. `nano`
 - `nano new_file.txt`
- Rozszerzenia plików – czy są potrzebne?
 - `type file` – identyfikuje typ pliku
- Do kopiowania plików i katalogów służy polecenie `cp`
 - `cp first_file backup_file`
 - `cp -r first_directory backup_directory`
- Do przenoszenia plików i katalogów służy polecenie `mv`
 - `mv first_file second_file`
 - `mv first_directory second_directory`

- Do usuwania plików katalogu w służy polecenie `rm`
 - `rm file`
- Do usuwania pustych katalogów służy polecenie `rmdir`
 - `rmdir empty_directory`
- Do usuwania rekursywnego służy flaga `-r` polecenia `rm`
 - `rm -r first_directory`
 - `rm -ri first_directory`
 - `rm -rf first_directory`
- Usuwanie plików i katalogów w powłoce jest **nieodwracalne!**

- Do przeglądania plików można użyć
 - `cat file` – wyświetla zawartość pliku na ekranie (na standardowym wyjściu)
 - `more file` – wyświetla zawartość pliku z opcją przewijania
 - `less file` – wyświetla zawartość pliku z opcją przewijania, również wstecz
- Dla komend `more/less`
 - `/wzorzec` – wyszukanie wzorca w pliku
 - `q` – wyjście do terminala
- Do edycji plików służą edytory: `nano`, `vim`, `emacs`
 - `nano`, podstawowe komendy
 - `Ctrl + x` – wyjście do terminala
 - `Ctrl + c` – zapis zmian
- Linux/Unix (`LF`) oraz Windows (`CR+LF`) różnie kończą linie w plikach tekstowych
- Konwersja
 - `dos2unix plik`
 - `unix2dos plik`

- Ze względu na dostęp do plików/katalogów użytkownicy podzieleni są na
 - właściciel (user)
 - grupa, do której należy właściciel (group)
 - pozostali użytkownicy (others)
- Prawa dostępu do pliku/katalogu
 - odczyt (read, r, 4)
 - zapis (write, w, 2)
 - prawo do wykonania (execute, x, 1)
- Komenda chmod zmienia prawa dostępu
 - składnia
 - komu: u (user), g (group), o (others), a (all)
 - operator: + (dodanie praw), - (odejście praw), = (ustawienie na podane prawa)
 - prawa: r (read), w (write), x (execute)
- Uruchomienie programu
 - ./program.exe

- Polecenie tar (tape archive) służy do pracy z archiwami plikowymi
 - tworzenie archiwum
 - tar cvf plik.tar katalogi_do_spakowania
 - tar czvf plik.tar.gz katalogi_do_spakowania
 - rozpakowywanie archiwum
 - tar xvf plik.archiwum

Kopiowanie plików na maszyny zdalne



- Polecenie scp (secure copy) służy kopiowania plików pomiędzy maszynami zdalnymi
 - przesyłanie plików na maszynę zdalną
 - `scp plik user@server:katalog/`
 - przesyłanie plików z maszyny zdalnej
 - `scp user@server:katalog/plik .`
 - przydatne flagi
 - `-C` – włącza kompresje
 - `-r` – kopiuje rekursively podkatalogi
 - przesyłanie katalogu na maszynę zdalną
 - `scp -r katalog user@server:katalog/`
 - przesyłanie katalogów z maszyny zdalnej
 - `scp -r user@server:katalog .`

- Do odczytania ilości znaków/linii w pliku służy polecenie `wc`

```
■ $ wc -l plik.txt
```

- Do sortowania w pliku służy polecenie `sort`

```
■ $ sort new_file.txt
```

- Do wyświetlania wyłącznie początku/końca pliku służą polecenia `head/tail`

```
■ $ head new_file.txt
```

```
■ $ tail new_file.txt
```

- Do wypisywania tekstu służy polecenie `echo`

```
■ $ echo I am who am I
```

- Do wypisywania plików służy polecenie `cat`

```
■ $ cat new_file.txt other_new_file.txt
```

- Znak * oznacza dowolny symbol(e) w nazwie pliku (tzw. wildcard)
- Każdy proces w systemie posiada swoją tablicę deskryptorów plików (posiada trzy standardowe strumienie) do komunikacji:
 - standardowy strumień wejścia (0, stdin)
 - standardowy strumień wyjścia (1, stdout)
 - standardowy strumień błędów (2, stderr)
- Do przekierowania strumienia stdout z programu służy symbol > lub >>
 - wc -l *.pdb > lines.txt
 - echo hello >> hello.txt
- Do przekierowania strumienia stderr z programu służy symbol 2> lub 2>>
 - wc -l *.pdb 2> lines.txt
 - echo hello 2>> hello.txt
- Do przekierowania strumienia stdin do programu służy symbol <
 - wc -l < test.pdb

- Potoki (pipes, |) pozwalają łatwo połączyć pracę kilku programów w jeden strumień

```
$ wc -l *.pdb
```

```
wc -l *.pdb
```

OUT



```
$ wc -l *.pdb > lengths
```

```
wc -l *.pdb
```

OUT



```
$ wc -l *.pdb | sort -n | head -1
```

```
wc -l *.pdb
```

OUT

IN

```
sort -n
```

OUT

IN

```
head -1
```

OUT



- Pętle pozwalają na wykonanie pewnych zadań na grupie obiektów automatycznie
- Schemat pętli w powłoce bash wygląda następująco:

```
for obiekt in grupa
do
    przetwarzaj $obiekt
done
```

Dzięki pętlom możliwa jest znaczna automatyzacja działań (nie trzeba każdego obiektu z grupy przetwarzać osobno).

- Skrypty pozwalają na zapisanie kilku komend wykonywanych sekwencyjnie przez powłokę po uruchomieniu danego skryptu
- Łącząc skrypty, pętle i potoki można wykonywać automatycznie nieskończoną ilość operacji automatycznie na każdym obiekcie z danej grupy
- Czasami lepiej poświęcić chwilę na napisanie dobrego skryptu, który zautomatyzuje pracę, niż wykonywać całość ręcznie
- Do skryptów można przekazać dodatkowe parametry i odczytać je poprzez zmienne \$1, \$2, ..., \$@
- Skrypty uruchamia się poprzez podanie ich jako argument powłoki bash
 - bash scriptname
 - nadanie praw do wykonywania plikowi skryptu
- Dzięki skryptom można całkowicie zautomatyzować swoją pracę!

- Do przeszukiwania w plikach wzorca służy polecenie grep

- \$ grep wzorzec plik

- Do wyszukiwania plików/katalogów służy polecenie find

- \$ find directory -name filename

- Oba polecenia można łączyć

- \$ grep wzorzec `find directory -name filename`
 - \$ grep wzorzec \$(find directory -name filename)

- Prometheus składa się z węzłów dostępowych, serwisowych oraz obliczeniowych
 - węzły obliczeniowe (2 232 węzłów z 2 procesorami Intel Xeon E5-2680v3)
 - 72 węzły z GPPGU (2 karty nVidia Tesla K40XL)

Własność	Prometheus
częstotliwość CPU	2.50 GHz
RAM	128 GB
rdzenie na węzeł	24
InifiniBand interconnect	FDR 56 Gb/s

- Zeus składa się z węzła dostępowego (UI) oraz grup węzłów o różnych parametrach
 - tradycyjne węzły (1198 węzłów, w tym 136 węzłów z dużą ilością RAM)
 - GPGPU – węzły zawierające procesory graficzne GPGPU (44 węzły, 208 kart GPGPU)

Własność	Zeus	Zeus BigMem	Zeus GPGPU
częstotliwość CPU	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
rdzenie na węzeł	8, 12	12, 64	12
InifiniBand interconnect	QDR	QDR	QDR
dodatkowe		–	karty GPGPU

- Składowanie danych (nie należy używać do obliczeń gdy duży I/O)
 - \$HOME – katalog domowy użytkownika
 - quota 40 GB
 - \$PLG_GROUPS_STORAGE – dodatkowa przestrzeń uzyskiwana w ramach grantów PLGrid
- Przestrzeń na dane tymczasowe (tzw. scratch)
 - \$SCRATCH – rozproszony system plików Lustre
 - dostępny z każdego z węzłów klastra
- Komenda do sprawdzania zajętości: pro-fs

- Składowanie danych (nie należy używać do obliczeń gdy duży I/O)
 - \$HOME – katalog domowy użytkownika
 - quota 7 GB
 - codzienny backup
 - \$STORAGE – długotrwałe przechowywanie plików
 - quota 100 GB
 - \$PLG_GROUPS_STORAGE – dodatkowa przestrzeń uzyskiwana w ramach grantów PLGrid
- Przestrzeń na dane tymczasowe (tzw. scratch)
 - \$TMPDIR – katalog lokalny na węźle obliczeniowym
 - dostępny jedynie z węzła, do którego jest podłączony
 - dostępny jedynie w trakcie trwania obliczeń
 - \$SCRATCH – rozproszony system plików Lustre
 - dostępny z każdego z węzłów klastra
- Komenda do sprawdzania zajętości: zeus-fs

- Aplikacje często wymagają specyficznego środowiska uruchomieniowego (m.in. zmiennych środowiskowych, dostępu do bibliotek) oraz wiedzy jak instalować je najbardziej efektywnie
- Narzędzie Lmod/Modules umożliwia proste ustawianie środowiska uruchomieniowego dla programów na każdym z klastrów dostępnych w PLGrid
- Zalety
 - łatwe ustawianie środowiska uruchomieniowego programów
 - przenoszenie skryptów obliczeniowych między maszynami
 - możliwość łatwego uruchamiania różnych wersji programów (często o skonfliktywnych środowiskach uruchomieniowych)
 - transparentne dla użytkownika uruchamianie wersji zoptymalizowanych na konkretny typ węzła obliczeniowego
- Wady
 - dodatkowa komenda do zapamiętania

- Załadowanie środowiska uruchomieniowego pakietu obliczeniowego

- `module add <module-name>` (np. `module add plgrid/apps/gaussian`)
 - `module load <module-name>` (np. `module load plgrid/apps/matlab`)

- Usunięcie modułu

- `module rm < module-name >` (np. `module rm plgrid/apps/gaussian`)
 - `module unload <module-name>` (np. `module unload plgrid/apps/matlab`)

- Listowanie wszystkich dostępnych modułów

- `module avail`
 - `module avail plgrid/tools` (tylko z gałęzi tools)
 - `module avail plgrid/apps/gaussian` (wszystkie dostępne wersje Gaussian)
 - `module spider gaussian` (wszystkie dostępne wersje Gaussian)

- Listowanie załadowanych modułów

- `module list`

- Usuwanie wszystkich załadowanych modułów

- `module purge`

- Każdy pakiet obliczeniowy zainstalowany na infrastrukturze PLGrid ma swój moduł
 - plgrid/<gałąź>/<software-name>/<wersja>
- Rodzaje gałęzi
 - apps – dla większości pakietów obliczeniowych
 - libs – dla bibliotek
 - tools – dla aplikacji narzędziowych i pomocniczych
- Przykłady:
 - plgrid/tools/intel/16.0.1
 - plgrid/apps/gaussian/g09.E.01
 - plgrid/tools/python/3.4.2
 - plgrid/apps/abaqus/2016

<https://apps.plgrid.pl/>

- System kolejkowy
 - zarządza zleconymi zadaniami obliczeniowymi
 - zarządza zasobami klastra
 - przydziela zasoby zadaniom obliczeniowym
 - dba o sprawiedliwą dystrybucję zasobów
- Zadania obliczeniowe są umieszczane w kolejkach i uruchamiane zgodnie z ich priorytetem i dostępnymi zasobami obliczeniowymi
- Priorytet zadania zależy m.in. od:
 - wielkości zasobów przyznanych w grancie obliczeniowym
 - ilości zażądanych przez zadanie zasobów i ich dostępności
 - szczególnie istotny jest maksymalny czas trwania obliczeń
 - zasobów klastra zużywanych aktualnie przez danego użytkownika

- Klastry obliczeniowe dostępne w PLGrid używają kilku odmian systemów kolejkowych

- 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>)

Centrum komputerowe	Klaster	System kolejkowy
ACC Cyfronet AGH	Prometheus	SLURM
	Zeus	Torque/Moab
ICM	Hydra	SLURM
	Topola	SLURM
	Eagle	SLURM
PSNC	Inula	PBS
	Tryton	SLURM
	Galera PLus	PBS/Maui
WCSS	Bem	PBS Pro

- Użytkownik komunikuje się z systemem kolejkowym SLURM za pomocą komend
 - `sbatch` – zlecenie nowego zadania wsadowego
 - `squeue` – wyświetlenie informacji o zadaniach w systemie
 - `scancel` – usuwa zdanie z systemu
 - `sinfo/scontrol` – wyświetlenie bardziej szczegółowym informacji o kolejkach, zadaniach i węzłach obliczeniowych
 - `srun` – uruchamia zadanie interaktywne lub pozadanie w zadaniu wsadowym
- Każde zadanie umieszczone w systemie uzyskuje unikalny identyfikator (`jobID`)

- Polecenie `sbatch` umieszcza nowe zadanie w kolejce
- Wszystkie parametry opisujące zadanie mogą być umieszczone w skrypcie obliczeniowym odpisującym zadanie lub podane jako argumenty polecenia `sbatch`
 - `sbatch [argumenty] script.slurm`
- Przykładowy skrypt

```
#!/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
```

- Polecenia `squeue` oraz `pro-jobs` wyświetlają zadania, które znajdują się w systemie
- Stany zadania
 - PD – zakolejkowane
 - R – uruchomione
- Dodatkowe pomocne flagi
 - `squeue --user $USER` – informacja o zadaniach użytkownika `$USER`
 - `squeue --start` – informacja estymowanym czasie rozpoczęcia zadania
 - `pro-jobs -j <jobID>` – informacja o wymienionych zadaniach o `jobID`
 - `pro-jobs -q/-r` – informacja tylko zadaniach o zakolejkowanych/uruchomionych
 - `pro-jobs -h` – ekran pomocy
- Dodatkowo `scontrol` oraz `sinfo` i `smap` dają dodatkową informacje o stanie klastra
 - `scontrol show job <jobID>` – informacja o zadaniu `<jobID>`
 - `scontrol show node <nodes_list>` – informacja o węzłach

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	węzły z GPGPU*

- System kolejkowym SLURM kolejki nazywa partycjami
 - `scontrol show partitions <nazwa_partycji>` – szczegółowe informacje o partycji
 - `sinfo` – wylistowanie informacji o stanach węzłów dostępnych w partycjach
 - `sinfo -p <nazwa_parycj_i>` – wylistowanie informacji jedynie dla wskazanej partycji
 - domyślny czas zadania w wszystkich partycjach `plgrid*` jest ustawiony na 15 minut
- * - partycje dostępne za żądanie (poprzez Helpdesk PLGrid)

```
#!/bin/env bash
#SBATCH -A tutorial

# 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
```

- Zasoby wymagane przez zadanie mogą być przekazane systemowi kolejkowemu poprzez opcje SLURM. Mogą one być
 - podane jako argumenty polecenia sbatch (sbatch [argumenty] script.slurm)
 - dołączone do skryptu obliczeniowego w pierwszych linijkach, które powinny zaczynać się od dyrektywy #SBATCH

- Polecenie sbatch posiada wiele opcji, które dodatkowo opisują zadanie obliczeniowe
 - -p <partition>, --partition=<partition> nazwa kolejki/partycji
 - -J <jobname>, --job-name=<jobname> nazwa zadania
 - -a, --array=<indexes> zadanie macierzowe
 - --mail-user=<user's e-mail> notyfikacje mailowe
 - --mail-type=<type> informacje kiedy notyfikacje mają być wysyłane: na początku (BEGIN), końcu(END) lub przy błędzie wykonania(FAIL)
 - -A <grantID>, --account= <grantID> informacje o wykorzystywany grancie obliczeniowym (gdy pominięte będzie wybrany domyślny)
- Gdy opcja -p jest pominięta zadanie zostanie włożone do partycji domyślnej (plgrid na klastrze Prometheus)

■ Zgłaszanie wymagań zadania

- `-t, --time=<time>` maksymalny czas wykonania zadania
- `-N, --nodes=<nodes>` ilość węzłów obliczeniowych dla zadani
- `-n, --ntasks=<number>` ilość podzadań uruchamianych w ramach zadania
- `--ntasks-per-node=<ntasks>` ilość podzadań na węzeł obliczeniowy
- `--cpus-per-task=<cores>` ilość rdzeni na jedno podzadanie (gdy wykorzystujemy wątki np. w OpenMP)
- `--mem=<MB/GB>` ilość pamięci potrzebnej na węźle obliczeniowym
- `--mem-per-cpu=<MB/GB>` ilość pamięci potrzebnej na rdzeń obliczeniowy

■ Formaty

- czasu: "min", "min:sec", "hours:min:sec", "days-hours", "days-hours:min" and "days-hours:min:sec"
- pamięci: B, kB (=1024b), MB (=1024kB), GB (=1,024MB)

```
#SBATCH -J adf.ethanol
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<user's e-mail>
#SBATCH --time=10:00
#SBATCH --mem=24000
#SBATCH -p plgrid
#SBATCH -A tutorial

module add plgrid/apps/adf

cd $SLURM_SUBMIT_DIR

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

- W SLURM zadania są wkładane do partycji a nie kolejek
 - argument `-p <nazwa_partycji>` lub `--partition <nazwa_partycji>`
 - partycje dla użytkowników PLGrid: **plgrid***

```
#SBATCH -J adf.ethanol
#SBATCH -N 2
#SBATCH --ntasks-per-node=24
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<user's e-mail>
#SBATCH --time=10:00
#SBATCH --mem=24000
#SBATCH -p plgrid
#SBATCH -A tutorial

module add plgrid/apps/adf

cd ${SLURM_SUBMIT_DIR}

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

```
#SBATCH -J gaussian.parallel
#SBATCH -N 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
```

- SLURM udostępnia zmienne środowiskowe ułatwiające obliczenia

Zmienna	Opis
SLURM_JOB_ID	identyfikator zadania (jobID)
SLURM_SUBMIT_DIR	katalog, z którego zadanie zostało zlecone
SLURM_NTASKS	ilość podzadań zleconych w zadaniu
SLURM_JOB_NODELIST	lista węzłów przydzielonych zadaniu
SLURM_CPUS_PER_TASK	ilość rdzeni obliczeniowych na podzadanie
TMPDIR	katalog na dane tymczasowe zadania
SCRATCH	główny katalog użytkownika na systemie Lustre na pliki tymczasowe

- Dodatkowo, gdy moduł tools/scratch jest załadowany
 - SCRATCHDIR – katalog utworzony dla zadania na systemie Lustre na pliki tymczasowe

- Praca interaktywna na klastrze powinna być wykonywana na węzłach obliczeniowych wykorzystując zadania interaktywne

```
■ srun -p plgrid -A <grant_id> -n 1 --pty /bin/bash
```

- Węzeł dostępowy nie powinien być wykorzystywany do obliczeń

- By podłączyć terminal do działającego zadania można wykorzystać polecenie `srun`

```
■ srun -N1 -n1 --jobid=<jobID> --pty /bin/bash
```

```
■ srun -N1 -n1 --jobid=<jobID> -w <nodeID> --pty /bin/bash
```

- Zadania macierzowe umożliwiają zakolejkowanie wielu zadań jednym wykonaniem polecenia `sbatch` wykorzystując przełącznik `-a, --array=<indexes>`
 - `sbatch -a n-m,k,l script.slurm` (np. `sbatch -a 0-9` lub `sbatch -a 2,4,7`)
- Wszystkie zadania w mają tą samą wartość zmiennych `SLURM_SUBMIT_DIR` oraz `SLURM_ARRAY_JOB_ID`, ale są identyfikowane przez zmienną `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` – pokazuje wszystkie zadania należące do macierzy podczas listowania zadań zakolejkowanych w systemie kolejkowym

- Do ustalania zależności między zadaniami służy przełącznik --dependency= <dependency_list> komend sbatch/srun
- Możliwe zależności
 - after:job_id[:jobid...] – zadanie rozpocznie się dopiero po rozpoczęciu zadań job_id
 - afterany:job_id[:jobid...] – zadanie rozpocznie się dopiero po zakończeniu zadań job_id
 - afternotok:job_id[:jobid...] – zadanie rozpocznie się dopiero po zakończeniu zadań job_id z błędem
 - afterok:job_id[:jobid...] – zadanie rozpocznie się dopiero po poprawnym zakończeniu zadań job_id
 - expand:job_id – zadanie zostanie uruchomione jako rozszerzenie zadania job_id
 - singleton – zadanie zostanie uruchomione po zakończeniu wszystkich innych zadań użytkownika

- Karty GPGPU są widoczne w systemie kolejkowym SLURM jako tzw. „generic resources” (GRES) z identyfikatorem gpu
- By sprawdzić gdzie są dostępne karty GPGPU

```
sinfo -o '%P || %N || %G'
```

- By zażądać kart graficznych należy dodać opcje --gres=gpu[:count] do polecenia sbatch/srun

```
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]
```

- Karty GPGPU są dostępne jedynie w partycji plgrid-gpu

- scancel służy do usunięcia niechcianych zadań z systemu kolejkowego
 - scancel <JobID>
- Zadania, które są zawieszone, a nie można ich usunąć polecienniem scancel należy zgłaszać poprzez Helpdesk PLGrid
 - <https://helpdesk.plgrid.pl>
 - helpdesk@plgrid.pl

- pro-jobs i pro-jobs-history mogą służyć do sprawdzania wydajności zadań
 - zużycie cykli CPU
 - maksymalne zużycie pamięci
- pro-jobs – dla zadań zakolejkowanych i uruchomionych
- pro-jobs-history – historyczne dane dla zadań zakończonych
- przykłady użycia pro-jobs*
 - pro-jobs -j <jobID> – informacje o zadaniu o jobID
 - pro-jobs -h – ekran pomocy

- Skrypt obliczeniowy SLURM zawsze rozpoczyna pracę w katalogu, z którego został umieszczony w systemie kolejkowym. Nazwę tego katalogu zawiera również zmienna `SLURM_SUBMIT_DIR`
- Zadania wsadowe posiadają plik(i), w których zapisywanie są dane wysyłane przez standardowe strumienie wyjścia (`stdout` oraz `stderr`). Standardowo jego nazwa to `slurm-<JobID>.out`
 - plik ten jest tworzony w katalogu `SLURM_SUBMIT_DIR`
- Gdy skrypt wsadowy przekazuje duże ilości danych do strumieni wyjścia użytkownik powinien przekazać te strumienie do pliku(-ów)
 - dla standardowego strumienia wyjścia (`stdout`): `command > file.out`
 - dla standardowego strumienia wyjścia błędów (`stderr`): `command 2> file.err`
 - by zapisać oba strumienie do jednego pliku: `command &> file.log`

- Przy zlecaniu zadań obliczeniowych użytkownik zawsze powinien
 - specyfikować maksymalny czas wykonania zadania (parametr `t/time`; wartość domyślna to 15 min)
 - specyfikować maksymalną ilość pamięci potrzebnej zadaniu przez parametry `mem` lub `mem-per-cpu` (wartość domyślna to `mem-per-cpu = 4GB`)
 - w przypadku zadań równoległych wykorzystywać całe węzły obliczeniowe, jeśli to możliwe
 - używać tzw. checkpointów do zapisu danych częściowych
 - gdy zadanie przetwarza dużą ilość danych pamiętać o przechowywaniu plików wejściowych i wyjściowych w `$SCRATCH`
 - starać się grupować zapisy i odczyty z plików by były duże (1MB+)
 - tworzyć większe pliki zamiast dużej ilości małych
 - w miarę możliwości wykorzystywać HDF (Hierarchical Data Format)
 - **czyścić** katalogi z plikami tymczasowymi po obliczeniach
 - ustawiać środowisko obliczeniowe oprogramowania poprzez polecenie `module` w skrypcie wsadowym
 - nie ładować modułów w skryptach uruchamianych podczas logowania na maszynę (np. `.bashrc`)

- Zawsze kompilować na **węźle obliczeniowym** wykorzystując
 - zadanie wsadowe komendą sbatch
 - zadanie interaktywne komendą srun --pty bash
 - wykorzystywać moduły (także dla bibliotek)
 - Intel® MKL Link Line Advisor: <https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/>
 - gdy oprogramowanie ma potencjał dla innych użytkowników prosić administratorów o instalację globalną
 - wykorzystywać IntelMPI (plgrid/tools/impi) i OpenMPI (plgrid/tools/openmpi) zbudowane przez administratorów
 - sprawdzać przypinanie procesów i wątków (i.e. zmienne środowiskowe KMP_AFFINITY)
- Flagi kompilacji na procesorach Haswell
 - GCC: -march=native, -fopenmp
 - Intel: -xCORE-AVX2 (or -xHost), -qopenmp, -fma-/no-fma
 - PGI: -tp haswell, -mp, -fast, -Mipa=fast, inline, -i8
- Gdy pojawią się problemy należy je zgłosić przez PLGrid Helpdesk
 - <https://helpdesk.plgrid.pl/>
 - helpdesk@plgrid.pl

plgrid/apps/gaussian

- Wersje dostępne na klastrze Prometheus
 - g09.C.01, g09.D.01 , g09.E.01
- Użycie
 - module add plgrid/apps/gaussian/<version>
 - g09 input.gjf
- Uwagi
 - duża ilość zaimplementowanych schematów obliczeń (w tym wiele funkcjonałów DFT)
 - rozpowszechnione użycie w środowisku naukowym
 - prosta składnia plików wejściowych
 - należy podawać ilość wykorzystywanych rdzeni w pliku wejściowym (%NProcShared=)
- Ograniczenia użycia
 - tylko tryb SMP (do 24 rdzeni @Prometheus, bo obliczenia na jednym węźle)

plgrid/apps/qchem

- Wersje dostępne na klastrze Prometheus
 - 4.3, 4.4.1
- Użycie
 - module add plgrid/apps/qchem/<version>
 - qchem -nt <number_of_cores> input.inp output.out
- Uwagi
 - duża ilość zaimplementowanych schematów obliczeń (w tym wiele funkcjonałów DFT)
 - Constrained DFT
 - prosta składnia plików wejściowych
- Ograniczenia użycia
 - tylko tryb SMP (do 24 rdzeni @Prometheus, bo obliczenia na jednym węźle)
 - tryb wielowęzłowy MPI w przygotowaniu

plgrid/apps/adf

- Wersje dostępne na klastrze Prometheus
 - 2013.01, 2014.07, 2014.10, 2016.102
- Użycie
 - module add plgrid/apps/adf/<version>
 - adf < input.in >& output.log
- Uwagi
 - używa baz STO
 - szeroko stosowany w badaniach związków metaloogranicznych
 - Zeroth Order Regular Approximation (ZORA)
 - prosta składnia plików wejściowych
 - moduł COSMO-RS
 - dobre narzędzia do analizy wyników (NBO, NOCV, ETS)
 - ADFView – GUI do wizualizacji wyników

plgrid/apps/turbomole

- Wersje dostępne na klastrze Prometheus
 - 7.0, 7.0.1, 7.1
- Użycie
 - module add plgrid/apps/turbomole/<version>
 - przygotowanie obliczeń przez skrypty pomocnicze (i.e. x2t, define, cosmoprep)
 - poszczególne obliczenia uruchamiane skryptami (i.e. dscf, ridft, ricc2, jobex, aofroce, egrad, riper)
- Uwagi
 - szybkie obliczenia i dobre skalowanie z wielkością układu
 - obliczenia DFT dla układów periodycznych
 - dobre zrównoleglenie

plgrid/apps/molpro

- Wersje dostępne na klastrze Prometheus
 - 2012.1.24, 2015.1.4, 2015.1.11
- Użycie
 - module add plgrid/apps/molpro/<version>
 - molpro [options] input.inp
 - --single-helper-server dla zadań na wielu węzłach i wymaganej dużej pamięci
- Uwagi
 - Obliczenia stanów wzbudzonych i.a metodami MCSCF/CASSCF, CASPT2, MRCI, or FCI,
 - dobre zrównoleglenie metod post-HF, w tym MP2-F12, CCSD(T)-F12

plgrid/apps/schrodinger

- Wersje dostępne na klastrze Prometheus
 - 2015-3, 2016-1, 2016-3 (in preparation)
- Użycie
 - module add plgrid/apps/schroedinger/<version>
 - jaguar run [options] input.inp
- Uwagi
 - Wysokiej jakości startowa struktura elektronowa dla układów z metalami przejściowymi
 - efekty solwacyjne z wykorzystaniem metody pola reakcyjnego (SCRF)
 - Maestro – zaawansowane GUI
 - automatyzacja zadań poprzez skrypty Pythona
 - inne programy dostępne w ramach pakietu SMDDs i.a. Desmond, Glide, Qsite, QSAR
- Ograniczenia użycia
 - tylko tryb SMP (do 24 rdzeni @Prometheus, bo obliczenia na jednym węźle)

plgrid/apps/terachem

- Wersje dostępne na klastrze Prometheus
 - 1.9
- Użycie
 - module add plgrid/apps/terachem/<version>
 - \$TERACHEMRUN input.inp > output.log
- Uwagi
 - szybkie obliczenia wykorzystujące GPGPUs
 - HF, DFT, TD-DFT oraz symulacje MD
- Ograniczenia użycia
 - tylko na węzłach z GPGPUs (opcja systemu kolejkowego --gres=gpu [:count])

plgrid/apps/niedoida

- Wersje dostępne na klastrze Prometheus
 - 0.5.3 (0.6 w przygotowaniu)
- Użycie
 - module add plgrid/apps/niedoida
 - niedoida --no-cores=XX input.inp
- Uwagi
 - HF, DFT (with DF and RI), TD-DFT oraz Dressed TD-DFT
 - znaczące usprawnienia w wersji 0.6
 - HF, DFT obliczenia na GPGPUs

plgrid/apps/cp2k

- Wersje dostępne na klastrze Prometheus
 - 2.6.1 (3.0.x w przygotowaniu)
- Użycie
 - module add plgrid/apps/niedoida
 - \$CP2K_RUN -i geopt.inp
- Uwagi
 - FF, HF, DFT and MP2 dla systemów periodycznych
 - symulacje MD

System kolejkowy PBS - polecenia

- Użytkownik komunikuje się z systemem kolejkowym za pomocą komend PBS
 - `qsub` – umieszcza zadanie w kolejce
 - `qstat` – wyświetla status zadań
 - `qdel` – usuwa zadanie z kolejki
 - `qalter` – zmiana parametrów zakolejkowanego zadania
- Każde zadanie w systemie kolejkowym ma swój własny, unikalny identyfikator zadania tzw. `jobID`

- Do umieszczenia zadania w kolejce systemu kolejkowego służy komenda qsub
- Komendy opisujące wykonywane zadanie mogą być zebrane w tzw. skrypt uruchomieniowy i przekazywane systemowi kolejowemu polecением
 - qsub skrypt
- Przykładowy skrypt

```
#!/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
```

Do sprawdzania statusów zadań w systemie kolejkowym służą komendy qstat lub zeus-jobs

Statusy zadań:

- zakolejkowane Q
- działające R

Dodatkowe przydatne filtry

- qstat -u \$USER – informacja o zadaniach użytkownika \$USER
- qstat -n <jobID> – informacja o węzłach przydzielonych zadaniu
- qstat -q – ogólna sytuacja na klastrze

- zeus-jobs -e- lub zeus-jobs -e+ – sortowanie po wydajności
- zeus-jobs -w – zadania o niskiej wydajności
- zeus-jobs -f (<jobID>) – szczegółowe informacje o zadaniach
- zeus-jobs -h – wyświetlenie pomocy

Dostępne kolejki

Nazwa	max czas	Uwagi
l_test	0:15:00	do testów
l_prio	1:00:00	
l_short	3:00:00	domyślna
l_long	336:00:00	
l_exclusive	336:00:00	zadania zajmujące całe węzły
l_interactive	72:00:00	zadania interaktywne
plgrid-testing	1:00:00	do testów
plgrid	72:00:00	
plgrid-long	168:00:00	
l_infinite	2160:00:00	*
l_bigmem	336:00:00	węzły z dużą ilością pamięci*
gpgpu	336:00:00	węzły z kartami GPGPU*

* dostęp na żądanie

qstat -Q -f <nazwa-kolejki> – szczegółowe parametry kolejki

qstat <nazwa kolejki> – wyświetlenie zadań w konkretnej kolejce

```
#!/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
```

- Opcje PBS pozwalają dostarczyć systemowi kolejowemu informacje o zasobach, które zadanie zmierza zużyć. Sposób wywołania:
 - w linii poleceń polecenia qsub [opcje PBS]
 - w początkowych linijkach skryptu uruchomieniowego poprzedzone dyrektywą #PBS
- Opcje wyspecyfikowane w linii poleceń nadpisują opcje podane w skrypcie

```
#!/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
```

- Skrypt zadania zawsze rozpoczynany jest w katalogu \$HOME na węźle obliczeniowym (WN). Przejście do katalogu, z którego wysłano skrypt jest możliwe dzięki zmiennej PBS_O_WORKDIR

- SLURM udostępnia zmienne środowiskowe ułatwiające obliczenia

Zmienna	Opis
PBS_JOBID	identyfikator zadania (jobID)
PBS_O_WORKDIR	ściezka, z której wysłano zadanie do PBS
PBS_NP	ilość rdzeni obliczeniowych dostępnych dla zadania
TMPDIR	katalog na dane tymczasowe zadania
SCRATCH	główny katalog użytkownika na systemie Lustre na pliki tymczasowe

- Dodatkowo, gdy moduł tools/scratch jest załadowany
 - SCRATCHDIR – katalog utworzony dla zadania na systemie Lustre na pliki tymczasowe

- Polecenie qsub posiada wiele opcji, które dodatkowo opisują zadanie obliczeniowe
 - -q kolejka definiuje kolejkę
 - -N nazwa ustawia nazwę zadania
 - -I zgłasza, że zadanie będzie interaktywne
 - -X włącza przenoszenie trybu graficznego przez protokół X11
 - -l zasób informuje o ilości zasobów potrzebnych zadaniu
 - -t n-m,k,l uruchamia zadania tablicowe o numerach od n do m oraz k i l
 - -M <user's e-mail> notyfikacje mailowe
 - -m b e a informacje kiedy notyfikacje mają być wysyłane: na początku (b), końcu (e) lub przy błędzie wykonania (a)
 - -A grantID informacje o wykorzystywanym grancie obliczeniowym (gdy pominięte będzie wybrany domyślny)
- Gdy opcja -q jest pominięta zadanie zostanie włożone do kolejki domyślnej (l_short na klastrze Zeus)

```
#!/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
```

- Zgłaszanie wymagań zadania (poprzez opcje -l)
 - walltime – maksymalny czas wykonania zadani
 - nodes=x:ppn=y – ilość węzłów oraz rdzeni obliczeniowych na węzeł
 - mem – ilość pamięci dostępnej dla zadania
 - pmem – ilość pamięci dostępnej na poszczególny rdzeń obliczeniowy
- Parametry powinny być przekazywane w formie “parametr=wartość” i oddzielone przecinkami
 - np. qsub -l walltime=10:00:00,nodes=1:ppn=12,mem=12gb
- Formaty
 - czas hhh:mm:ss
 - pamięć b, kb (=1024b), mb (=1024kb), gb (=1,024mb)
 - węzły obliczeniowe nodes=ilosc-wezlow:ppn=liczba-rdzeni-na-wazeł:wlaściwości (np. nodes=2:ppn=12 – dwa węzły po dwanaście rdzeni)

```
#!/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
```

- komenda qdel komis służy do usuwania zadań z systemu kolejkowego
 - `qdel <JobID>`
- Zadania, które są zawieszone, a nie można ich usunąć poleceniem scancel należy zgłaszać poprzez Helpdesk PLGrid
 - <https://helpdesk.plgrid.pl>
 - helpdesk@plgrid.pl

- Do zmian parametrów zadań w systemie kolejkowym służy komenda qalter

```
qalter <jobID> [zmieniane_parametry]
```

- Przykładowe zastosowania

```
qalter <jobID> -l nodes=x:ppn=y
```

```
qalter <jobID> -l walltime=hhh:mm:ss
```

```
qalter <jobID> -N nowa_nazwa_zadania
```

- qalter nie może zmieniać kolejki zadania oraz modyfikować parametrów uruchomionego zadania

- Praca interaktywna na klastrze powinna być wykonywana na węzłach obliczeniowych wykorzystując zadania interaktywne
 - `qsub -I`
 - `qsub -I -X` by przesyłać tryb graficzny protokołem X11
- Kolejka `l_interactive` jest dedykowana do pracy iteraktywnej
- W przypadku użycia trybu graficznego należy pamiętać
 - o zalogowaniu się na klaster z użyciem przekierowania wyświetlania X11 (np. `ssh -Y -C login@serwer`; w PuTTY "Enable X11 Forwarding")
 - włączeniu serwera X11 na maszynie, z której następuje logowanie
- Nie wolno wykonywać żadnych obciążających operacji na węźle dostępowym (UI) klastra

- Zadania macierzowe umożliwiają zakolejkowanie wielu zadań jednym wykonaniem polecenia qsub
 - qsub -t n-m,k,l script.pbs (np. qsub -t 0-9 lub qsub -t 2,4,7)
- Wszystkie zadania w mają tą samą wartość zmiennej PBS_O_WORKDIR, ale są identyfikowane przez zmienną \$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 – pokazuje wszystkie zadania należące do macierzy podczas listowania zadań zakolejkowanych w systemie kolejkowym

- zeus-jobs i zeus-jobs-history mogą służyć do sprawdzania wydajności zadań
 - zużycie cykli CPU
 - maksymalne zużycie pamięci
- zeus-jobs – dla zadań zakolejkowanych i uruchomionych
- zeus-jobs-history – historyczne dane dla zadań zakończonych
- Przykłady użycia zeus-jobs*
 - zeus-jobs -e- lub zeus-jobs -e+ - sortowanie po efektywności zadań
 - zeus-jobs -w – zadania o niskiej wydajności
 - zeus-jobs -f (<jobID>) – szczegółowe informacje o zadaniach <jobID>
 - zeus-jobs -h – ekran pomocy

- Zadanie wsadowe PBS jest uruchamianie w katalog domowym użytkownika \$HOME na węźle obliczeniowym. Dostęp do katalogu, z którego skrypt został umieszczony w systemie kolejkowym, zapewnia zmienna PBS_O_WORKDIR
- Dla każdego zadania tworzone są automatycznie pliki zawierające
 - standardowe wyjście nazwa .o<JobID>
 - standardowe wyjście błędów nazwa .e<JobID>
 - Powyższe pliki nie powinny być bardzo duże (nie więcej niż kilka MB) i są dostępne dopiero po zakończeniu zadania
- Gdy polecenia w skrypcie przekierowują dużą ilość danych na standardowe wyjścia należy wyspecyfikować jawnie przekierowanie do pliku
 - standardowe wyjście komenda > plik.out
 - standardowe wyjście błędów komenda 2> plik.err
 - oba strumienie komenda &> plik.log

- Przy specyfikacji każdego zadania należy zawsze
 - specyfikować maksymalny czas wykonania `walltime`
 - unikać stosowania kolejki `l_infinite`
 - specyfikować ilość pamięci wykorzystywanej `mem` (lub `pmem`)
 - tworzyć pliki zapisujące kroki obliczeń umożliwiające restart (tzw. `checkpoints`)
 - w przypadku zadań zrównoleglonych zajmować całe węzły obliczeniowe, zalecana kolejka `l_exclusive`
 - gdy zadanie przekierowuje dużą ilość danych na standardowe wyjścia należy wyspecyfikować jawnie przekierowania do pliku
 - środowisko obliczeniowe aplikacji i bibliotek ładować komendą `module`
 - nie ładować modułów w plikach startowych powłok (np. `.bashrc`)

- **nie wykorzystywać do obliczeń \$HOME i \$STORAGE gdy obliczeniowa o dużym I/O**
- używać katalogów na pliki tymczasowe tzw. scratch
 - lokalne dyski scratch (zmienna \$TMPDIR)
 - dostęp do danych tylko z jednego węzła
 - duża liczba operacji odczytu/zapisu małych porcji danych (<<1MB na zapis/odczyt)
 - nieduże pliki (do ~5 GB na węzeł)
 - rozproszony zasób scratch (Lustre; \$SCRATCH oraz \$SCRATCHDIR)
 - dostęp do danych z wielu węzłów
 - duże pliki tymczasowe (10+GB)
 - duże jednorazowe odczyty/zapisy (1+MB)
 - potrzebny podgląd w trakcie wykonywania
- **czyścić katalogi z plikami tymczasowymi po obliczeni**



Rejestracja: <https://portal.plgrid.pl>

helpdesk@plgrid.pl

+48 12 632 33 55 wew. 312