



Wprowadzenie do obliczeń naukowych na Komputerach Dużej Mocy

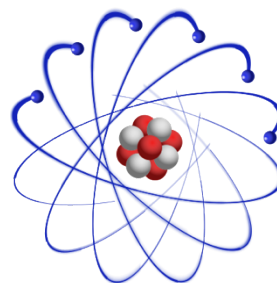
Klemens Noga

ACK Cyfronet AGH

Wydział Chemii Uniwersytetu Jagiellońskiego, Kraków 1 XII 2017

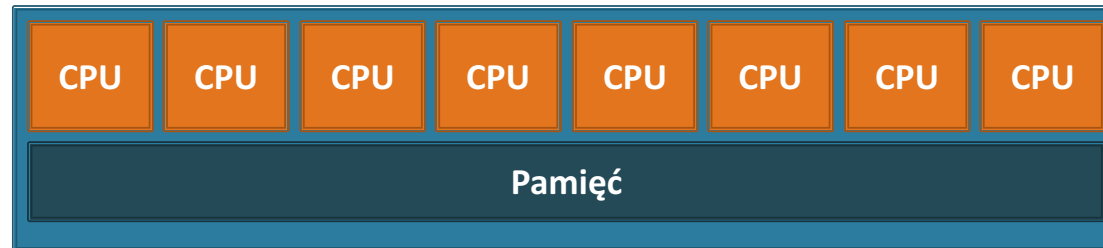
- Klastry obliczeniowe w ACK Cyfronet AGH
 - sposoby dostępu
 - praca z tekstowym interfejsem użytkownika
 - dostępne zasoby obliczeniowe i dyskowe
 - zarządzanie środowiskiem aplikacji obliczeniowych
- Przeprowadzenie obliczeń
 - systemy kolejkowe
 - skrypty obliczeniowe
 - obliczenia sekwencyjne i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

- Wszystkie klastry obliczeniowe w PLGrid wykorzystują Linux
 - Scientific Linux 6 na Zeusie
 - CentOS 7 na Prometheusie

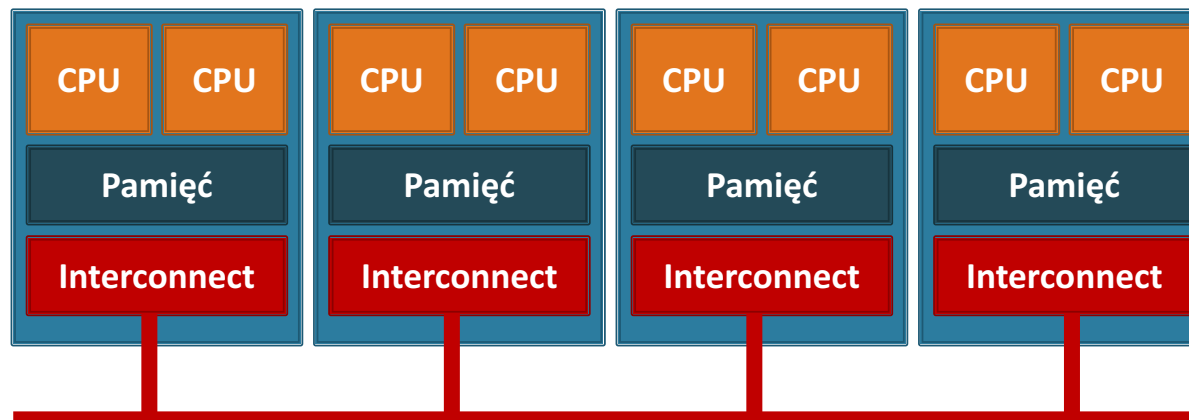


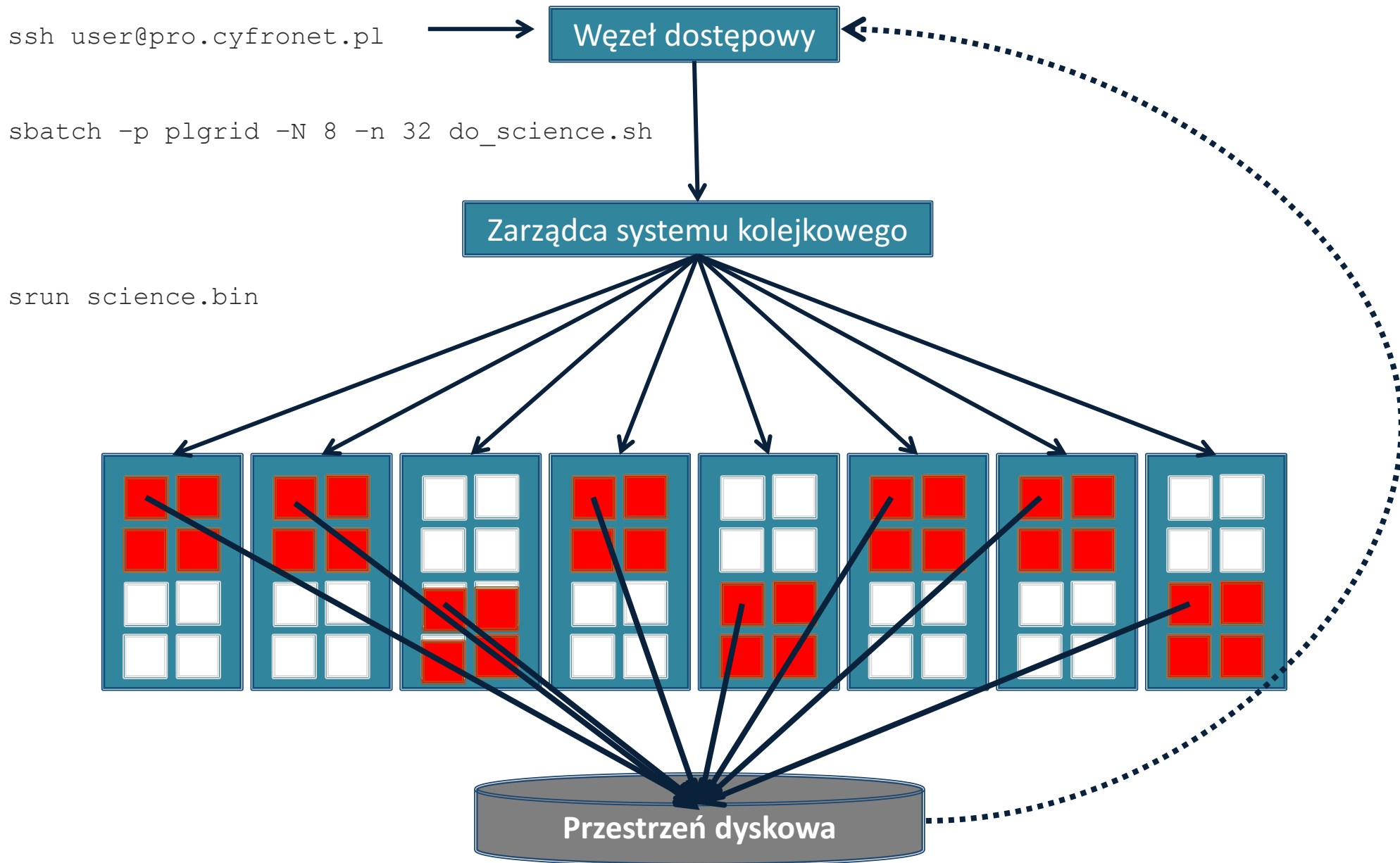
- Klastry obliczeniowe zawierają
 - węzeł dostępowy (user interface (UI))
 - węzły obliczeniowe
- Przeprowadzanie obliczeń na węźle dostępowym jest zabronione
- Sprawiedliwym dostępem do zasobów zajmuje się system kolejkowy
 - SLURM na Zeusie i Prometheusie

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 otwarciu 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
- Autouzupełnianianie
 - 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

- 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), `-` (odjęcie praw), `=` (ustawienie na podane prawa)
 - prawa: `r` (`read`), `w` (`write`), `x` (`excecute`)
- 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`

- 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 rekursywnie 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`

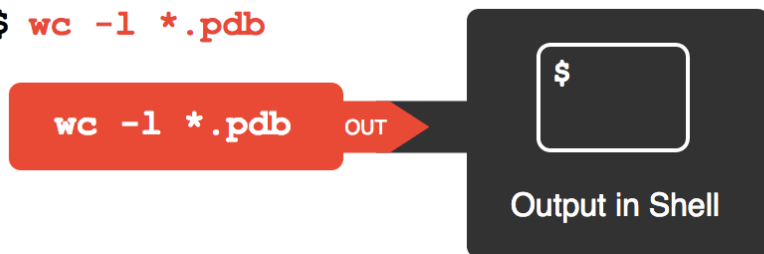
- Do wypisywania plików służy polecenie `cat`

- `$ cat new_file.txt other_new_file.txt`

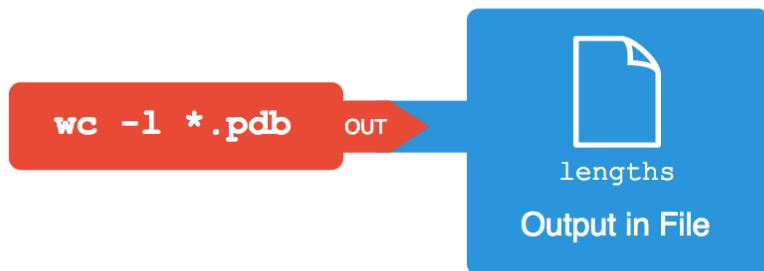
- Znak * oznacza dowolny symbol(e) w nazwie pliku (tzw. wildcard)
- Każdy proces w systemie posiada swoją tablicę deskryptorów plików (posiada trzy standardowe strumienie) do komunikacji:
 - standardowy strumień wejścia (0, stdin)
 - standardowy strumień wyjścia (1, stdout)
 - standardowy strumień błędów (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

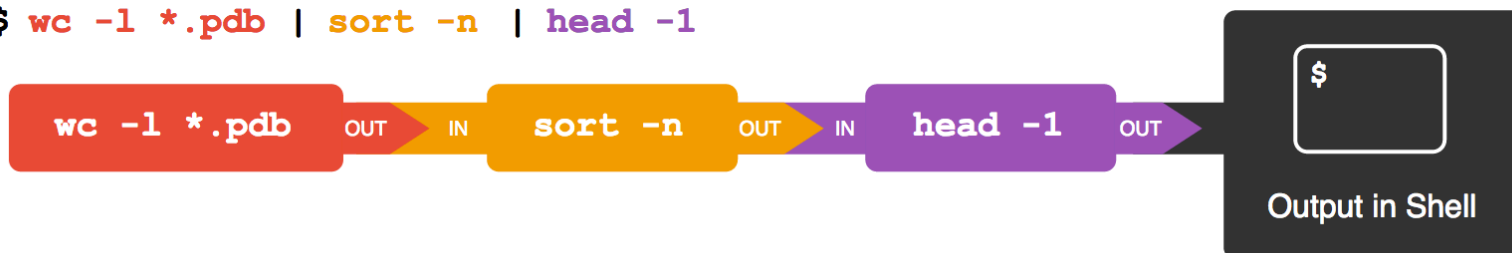
```
$ wc -l *.pdb
```



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



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



- 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 |
| InfiniBand 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 |
| InfiniBand interconnect | QDR | QDR | QDR |
| dotatkowe | | – | 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 skonfliktowanych 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/<gałęź>/<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/g16.A.03`
 - `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 granice 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 | SLURM |
| ICM | Hydra | SLURM |
| | Topola | SLURM |
| PSNC | Eagle | SLURM |
| | Inula | PBS |
| TASK | 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 zadanie 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 informację 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 | dla testów |
| plgrid-short | 1:00:00 | dla krótkich zadań (tzw. back fill) |
| plgrid | 3-00:00:00 | domyślna |
| plgrid-large | 1-00:00:00 | zadania wykorzystujące co najmniej 18 węzłów |
| 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_partycji>` – wylistowanie informacji jedynie dla wskazanej partycji
 - domyślny czas zadania w wszystkich partycjach `plgrid*` jest ustawiony na 15 minut
- * - partycje dostępne za żądanie (w trakcie negocjacji grantów obliczeniowych lub 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 liniach, 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 wykorzystywanym granicie 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 zadania
- `-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 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`

- polecenie `ssh_slurm`

- `ssh_slurm <jobID> <nodeID> [command]`

- 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 tę 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 zawieszony, a nie można ich usunąć poleceniem `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

- `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

- 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 zapisywane są dane wysyłane przez standardowe strumienie wyjść (*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`)

- **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 obliczeniach

- 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, g16.A.03
- 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
 - dla G09 należy podawać ilość wykorzystywanych rdzeni w pliku wejściowym (`%NProcShared=`)
 - G16 sam ustawia odpowiednio liczbę rdzeni obliczeniowych
- 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 funkcyjałów 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, 2016.107, 2017.101, 2017.103, 2017.106
- 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, 7.2
- 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, 2015.1.23
- 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/mopac

- Wersje dostępne na klastrze Prometheus
 - 2016
- Użycie
 - `module add plgrid/apps/mopac/<version>`
 - `mopac.sh input`
- Uwagi
 - Program do obliczeń empirycznych, metody PM6, PM7

plgrid/apps/schrodinger

- Wersje dostępne na klastrze Prometheus
 - 2015-3, 2016-1, 2016-4
- Użycie
 - `module add plgrid/apps/schroedinger/<version>`
 - `jaguar run [options] input.inp`
- Uwagi
 - Wysokiej jakości startowa struktura elektronowa dla układów z metalami przejściowymi
 - efekty solwatacyjne 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
 - 1.91, 1.93
- 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



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

helpdesk@plgrid.pl

+48 12 632 33 55 wew. 312