System kolejkowy `slurm`

Prelegent: Jakub Gałecki, `jgalecki@icm.edu.pl`

ICM, 04.06.2024

Sylwetka prowadzącego:

- Członek Zespołu Oprogramowania i Wsparcia Użytkowników ICM
- Autor biblioteki do skalowalnego rozwiązywania RRC (symulacji)
- Jeden z maintainerów stosu aplikacji ICM

Materiały:

- `https://slurm.schedmd.com/`
- `https://kdm.icm.edu.pl/Tutorials/HPC-intro/slurm_intro/`

Problem

Zakupiliśmy i uruchomiliśmy 1000 maszyn wyposażonych w procesory serwerowe, etc.

Jak możemy udostępnić je do użytkowania?

Przykłady systemów kolejkowych

- **slurm**
- PBS
- TORQUE
- Grid Engine
- LSF

Simple Linux Utility for Resource Management

`https://slurm.schedmd.com/`

- Przydziela zasoby obliczeniowe (sprawiedliwie)
- Pozwala na prowadzenie księgowości CPUh (i innych)
- Skalowalny
- Zarządza kolejką zadań
- Integracja z MPI

Komendy slurm

- **squeue** – stan kolejki
- sinfo – informacje o stanie zasobów
- salloc – alokacja zasobów na zadanie
- **srun** – uruchomienie zadania
- **sbatch** – dodanie zadania do kolejki
- scancel – anulowanie zadania
- scontrol*
- sacct*
- sreport*

Więcej na:

`https://kdm.icm.edu.pl/Tutorials/HPC-intro/slurm_intro/`

Terminologia slurm

- Node (węzeł) – jednostka obliczeniowa (maszyna), posiadająca CPU i pamięć
- Partition (partycja) – zbiór węzłów o konkretnych własnościach, bądź wydzielony do osobnej grupy
- Job (zadanie) – zlecona praca, np. skrypt
- Step (krok) – podzbiór zadania, często zadanie składa się tylko z jednego kroku
- Task (...) – jeden z procesów biorących udział w kroku/zadaniu (typowo rząd MPI)

Zmienne środowiskowe slurm

Slurm dodaje do środowiska uruchamianych zadań zmienne, zawierające istotne informacje

Przykładowo:

- `SLURM_CPUS_ON_NODE`
- `SLURM_JOB_ID`
- `SLURM_NNODES`
- `SLURMD_NODENAME`

srun

Komendą `srun` uruchamiamy dany skrypt/program

`srun` jest *blokujący*

`srun` jest uruchamiany w ramach zaalokowanych zasobów (`salloc`, `sbatch`), jeżeli takowe istnieją

`srun` należy używać *zamiast* `mpirun`/`mpiexec`

Sesja interaktywna

```
srun [...] --pty bash -l
```

Efektywnie logowanie ssh na węzeł (z ewentualnym ograniczeniem zasobów)

Przydatne do niewielkich eksperymentów

sbatch

Zleca wykonanie skryptu/programu do kolejki

`sbatch` działa *asynchronicznie*

Typowy workflow zlecanego skryptu:

1. Ładujemy potrzebne moduły
2. Inicjalizujemy środowisko
3. Uruchamiamy skalowalny program komendą `srun`
4. Finalizujemy zadanie

Najczęściej wykorzystywane opcje

- `-A/--account`
- `-p/--partition`
- `-n/--ntasks`
- `-N/--nodes`
- `--ntasks-per-X`
- `--time`
- `--mem`
- `-J/--job-name`
- `-o/--output`
- `--dependency`

Opcje możemy także zaszyć w skryptach zlecanych `sbatch`, wymieniając je w komentarzach na początku skryptu:

```
#SBATCH --opcja wartość
```

Job array

W przypadku, kiedy chcemy zlecić dużą liczbę zadań, różniących się jedynie parametrami, slurm daje nam do dyspozycji tablicę(?) zadań – job array.

Zlecamy jedynie komendą `sbatch`, przy pomocy opcji `--array`, np.:

- `--array=0-31`
- `--array=1,3,5,7`
- `--array=1-7:2`

Poszczególne elementy tablicy zadań będą mogły sprawdzić swój indeks przy pomocy zmiennej środowiskowej `SLURM_ARRAY_TASK_ID`

Więcej na: `https://slurm.schedmd.com/job_array.html`

Zadania

Hello world – slurm

Zapoznaj się z obecnym stanem dostępnych partycji. Jakie zadania znajdują się obecnie w kolejce?

Zaloguj się na węzeł obliczeniowy w sesji interaktywnej, wykorzystując jeden rdzeń. Wydrukuj nazwę węzła, do którego uzyskałeś/aś dostęp.

Sprawdzanie przyznanych zasobów

Zbadaj przydzielone zasoby, zlecając do kolejki program:

```
#include <mpi.h>
#include <omp.h>
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char *argv[]) {
  MPI_Init(&argc, &argv);
  const auto node_id = getenv("SLURM_NODEID");
  const auto node_str = node_id ? node_id : "not launched via slurm";
  int comm_sz, rank;
  MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  const int num_threads = omp_get_max_threads();
  cout << string("Rank ") + to_string(rank) + " of " + to_string(comm_sz) +
          ", threads available: " + to_string(num_threads) +
          ", slurm node ID: " + node_str + "\n";
  MPI_Finalize();
}
```

Kompilacja (po załadowaniu modułu MPI):

```
$ mpic++ -fopenmp kod.cpp -o program.x
```

Job array

Uruchom 5 zadań, każde z których:

1. Oblicza trajektorię wzdłuż systemu Lorentza
   - od punktu $[1, 1, 1]$ + mała, losowa perturbacja
   - w interwale czasowym $[0, 10]$
   - wykorzystuje do obliczeń Pythona
2. Zapisuje wynik do pliku

Następnie w trybie interaktywnym wczytaj wszystkie wyniki i narysuj je na jednym wykresie (innymi kolorami). Zapisz wykres do pliku i obejrzyj wyniki – chaotyczne trajektorie wzdłuż atraktora Lorentza.

Do napisania potrzebnego kodu w Pythonie wykorzystaj ChatGPT.