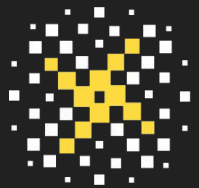
# from zero to ${0##*/}

an introduction to bash scripting and HPC

alberto sartori



SISSA
DATASCIENCE
Machine Learning for the Natural Sciences

feeling the power

# ssh settings

```
$ ls ~/.ssh
$ ssh-keygen -t rsa -C "your@email.com"
$ cat ~/.ssh/config
Host ulysses
    User your_username
    HostName frontend2.hpc.sissa.it

$ ssh-copy-id ulysses
$ ssh ulysses
```

# copying files

```
$ scp ulysses:~/remote/file .
$ scp local_file ulysses:~/some/where

$ rsync -av ulysses:~/remote/file .
$ rsync -av local_file ulysses:~/some/where

$ # use git
```

# know the cluster

```
$ df -h

$ sinfo -s

$ scontrol show partition

$ scontrol show partition regular2
```

# know the computing node(s)

```
$ salloc --nodes=1 --exclusive -p regular2 # acquire whole node

$ srun -n 1 lstopo --of png > computing_node.png

$ srun -n 1 cat /proc/cpuinfo

$ exit
```


Host: cn09-17

Indexes: physical

Date: Mon 25 Oct 2021 12:05:25 PM CEST

# environment modules (Lmod)

```
$ module avail

$ module avail cuda

$ module show cuda

$ module show cuda/11.0

$ module load git

$ module list

$ module purge
```

# environment modules (Lmod)

```
$ module load openmpi3

$ module spider openmpi3

$ module load intel/2021.2 openmpi3

$ mpicc --version

$ module load gnu8

$ mpicc --version
```

# how to exploit computing nodes

- interactive:
  - salloc + srun
  - srun --ntasks=64 --pty --partition=regular2 bash -i

- batch:
  - sbatch my_script.sh

```
#!/bin/bash
#SBATCH --job-name=ssss
#SBATCH --tasks=NNNN # total number of cpus
#SBATCH --tasks-per-node=MMMM # max tasks per node
#SBATCH --time=hh:mm:ss
#SBATCH --partition=ppppp
#SBATCH --hint=nomultithread
#SBATCH -e %x.err
#SBATCH -o %x.out

cd ${SLURM_SUBMIT_DIR}
module load gnu8 openmpi3/3.1.4
mpirun -n NNNN ./a.out
```

# strong scaling

choose total number of tosses N

increase the number of processes

compute the speedup

$$S(n) = \frac{\text{time of best sequential code}}{\text{time of parallel code}}$$

repeat for different N (e.g., N/2, N*2, N*4)

# weak scaling

keeping constant the workload of each process, change the number of processes (i.e., fix the ratio #tosses/#processes)

compute the efficiency

$$E(n) = \frac{\text{time of best sequential code}}{\text{time of parallel code}}$$

repeat for different ratios

- upload pi.c and mpi_pi.c to Ulysses (home or scratch)
- load gnu8 and openmpi3
- compile the two files
  - `gcc -O3 pi.c -o serial.x`
  - `mpicc -O3 mpi_pi.c -o parallel.x`

- compute strong and weak scaling
  - regular1 (1, 2, 4, 8, 16, 20, 40, 80 processes) x (10240, 102400000)
  - regular2 (1, 2, 4, 8, 16, 32, 64, 128 processes)x (10240, 102400000)

- during this morning you can use
  - `#SBATCH --partition=regular1 --reservation=mhpc_20211122`

the serial job can be run

```
./serial.x 10240
```

the parallel job

```
mpirun -n 128 ./parallel.x 10240
```

write a script to submit the jobs using the here document syntax

write a script to post-process the data, what can you infer?

send me the plots with your deductions