ilifu Online Training

Robin Hall

User Training Workshop – Advanced Training

21 September 2021

What is a program?

- Set of discrete instructions
- Carried out sequentially

What is a program?

Print average grade of a class

```
1. total = 0
2. for grade in grades:

     total = total + grade

3. average = total / number_of_grades
4. print average
```

Parallelism

- Executing portions of program simultaneously
- Possible when we have many CPUs (cores/processors)
- Capacity dependent on structure of both hardware AND software
- Requires overall control/coordination mechanism

Parallelism on the cluster

- A cluster includes many connected nodes
- Each node has RAM and multiple CPUs
- Work of job is partitioned into smaller jobs
- Sometimes with a partition of the data

Parallelism on the cluster

Partition grades into 2:

```
1. total = 0
2. for grade in 1/2 grades:

    total = total + grade

3. average1 = total /
number_of_grades
```

```
1. total = 0
2. for grade in 1/2 grades:

    total = total + grade

3. average2 = total /
number_of_grades
```

Combine results:

```
average = (average1 + average2) / number_of_partitions
```

Parallelism on the cluster

Can be achieved on a single machine / node

- Distributes work over many CPUs
- Typically implemented using OpenMP

Or over multiple machines / nodes

- Distributes work over many tasks, over 1+ nodes
- Each given amount of memory to use
- Generally requires a cluster
- Typically implemented using OpenMPI
- Requires a message passing interface (MPI) wrapper

  mpirun, aprun, srun (SLURM), mpicasa (CASA 5)

Managed on ilifu by SLURM

Parallelism on the cluster

Implementing a normal job in SLURM

- Will only use 1 CPU, 1 task, and 1 node
- Default for many processes

Implementing an OpenMP job in SLURM

- Need to use >1 CPU, while nodes & tasks must be 1 (unless also using MPI)

  cpus-per-task

  May need to export OMP_NUM_THREADS

Implementing an MPI job in SLURM

- Need to use >1 task, while nodes and CPUs can be 1

  nodes, ntasks-per-node, cpus-per-task

  Need to wrap singularity in MPI call

Cannot exceed 32 CPUs (or tasks) per node

Serial jobs

If code is serial, i.e. doesn't use OpenMP or MPI
Increasing CPUs or nodes will not decrease execution time

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
```

Multi-threaded jobs

Using multiple CPUs within a node with OpenMP
Where N is an optimal number of CPUs

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=N
```

Note: The maximum number of CPUs per node (32) will not always give the most speedup

Multi-threaded jobs

Can also specify tasks per node
This method will create tasks that can communicate via MPI

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=N
#SBATCH --cpus-per-task=1
```

Multi-node jobs

Using multiple nodes with MPI

```
#SBATCH --nodes=N
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
```

Note: Not ideal because internode communication is slower than intranode communication

Multi-threaded & multi-node jobs

Using multiple nodes with MPI as well as multiple cores within node with OpenMP

```
#SBATCH --nodes=N
#SBATCH --ntasks-per-node=M
#SBATCH --cpus-per-task=1
```

MPI Wrapper

When submitting jobs that use MPI:

```
#!/bin/bash

#SBATCH --nodes=N
#SBATCH --ntasks-per-node=M
#SBATCH --cpus-per-task=1
#SBATCH --mem=10GB

/path/to/mpirun singularity exec /path/to/container.simg python my_script.py
```

Summary

- Parallel processing is all about speed up
- Can be labour intensive
- Need to understand your software and it’s requirements