GWDG
AG-C
Hendrik Nolte

Tutorial 1 / April 21st, 2023
Practical Course High-Performance Computing / SoSe 2023
75 Minutes Total

Learning Objectives

The learning objectives in the tutorial are

- Construct and *Pickle Python* objects

- Describe and measure the performance impact "Pickling" has on MPI communication

- Apply the differing MPI-routines based on the first letter correctly

Tools

- mpi4py

- Anaconda

Contents

Working with Anaconda 1: Tutorial (15 min)

Anaconda can be used to create your own, dedicated python environments. *Anaconda* is already installed on the *SCC* and can be immediately used by you.

Steps

1. Load the *anaconda3* module

2. Create a new environment called *pchpc*

3. Activate the new environment

4. Install necessary modules with pip, try to install *numpy* or *pandas*

5. Submit a simple python script with *sbatch*, which only imports *numpy* and *pandas*. You can print a *Done!* afterwards to have some output on yout *stdout*

Hints

- In your run/batch script, you also need to load the *anaconda3* module and activate your environment

- You can load a module with:

```
1   module load anaconda3
```

- to be able to install python modules with *pip* use:

```
1   # note that you could use a different name for your environment than 'pip_test'
2   conda create -n pip_test pip
```

- You can list your different conda environments with:

```
1   conda env list
```

- You can remove a conda environments with:

```
1   conda env remove -n pip_test
```

- You can activate a conda environment with:

```
1   conda activate pip_test
```

- If you get an error activating your environment run:

```
1   conda init bash
2   source ~/.bashrc
3   conda activate pip_test
```

Sending a Custom Object 2: Tutorial (30 min)

In the lecture we have seen two different groups of communication based on the performed "pickling", remember, it depended on the capitalized first character (send vs. Send). In this tutorial we want to write a short ping-pong benchmark to measure the effects "pickling" have on the overall communication performance.

Steps

1. Load the module *openmpi* `$ module load openmpi`

2. Install *Python 3.7* using:
   `$ conda install python=3.7`

3. Install *mpi4py* into your *conda* environment
   `$ pip install mpi4py`

4. Create a "rather complicated" class which has an internal state, e.g. some variables are set to a certain value

5. Define a function which modifies the state (variable) of an object of that class

6. Define a function to print the state of an object of that class

7. The process with $rank == 0$ should create one instance

8. Send the object to process with $rank == 1$

9. Print the state of the object on the process $rank == 1$.

10. Change the state of the object in process $rank == 1$ and send it to process $rank == 0$

11. Print the state of the object in process $rank == 0$

Hints

- If you get an error installing *mpi4py* with *pip* you are likely having a wrong *Python* version.

- The difference between *conda istall* and *pip install* is that *conda* pulls prebuilt packages, whereas *pip* is building a package locally

- You can built your custom class with something like:

```python
class MyClass:

  def __init__(self):
    self.my_str = 'Initialized'

  def print_state(self):
    print(self.my_str)

  def change_state(self, new_str):
    self.my_str = new_str
```

- You ca initialize your MPI with:

```python
from mpi4py import MPI

comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()
```

- You can check the rank of a process with

```python
if rank == 0:
  # do stuff
```

- You can send and receive messages with:

```python
if rank == 0:
  data = {'a': 1, 'b': 2, 'c':'test string'}
  comm.send(data,dest=1,tag=11)
elif rank == 1:
  data = comm.recv(source=0,tag=11)
  print(data)
```

Benchmarking the "Pickling" Impact 3: Tutorial (30 min)

Steps

1. Initialize a *numpy* array with 1000000 elements

2. Send this array from process 0 to process 1 using the lower case mpi function

3. Measure the time this communication needs. Feel free to average over multiple runs

4. Repeat the previous steps with the upper case MPI functions

5. Describe and explain the difference

Hints

- You can initialize a numpy array with

```
1    data = numpy.arange(1000000, dtype=numpy.float)
```

- You can measure the time a code snippet needs with:

```
1    import time
2
3    start = time.time()
4    print("hello")
5    end = time.time()
6    print(end - start)
```

Further Reading

- https://docs.python.org/3/tutorial/classes.html

- https://mpi4py.readthedocs.io/en/stable/index.html