MPI4Py (MPI for Python)

OpenMPI interface to Python

Introduction

MPI for Python provides bindings of the Message Passing Interface (MPI) standard for the Python programming language, allowing any Python program to exploit multiple processors.

This package is constructed on top of the MPI-1/2 specifications and provides an object oriented interface which closely follows MPI-2 C++ bindings. It supports point-to-point (sends, receives) and collective (broadcasts, scatters, gathers) communications of any picklable Python object, as well as optimized communications of Python object exposing the single-segment buffer interface (NumPy arrays, builtin bytes/string/array objects).

On Anselm MPI4Py is available in standard Python modules.

Modules

MPI4Py is build for OpenMPI. Before you start with MPI4Py you need to load Python and OpenMPI modules.

```
$ module load python
$ module load openmpi
```

Execution

You need to import MPI to your python program. Include the following line to the python script:

```
from mpi4py import MPI
```

The MPI4Py enabled python programs execute as any other OpenMPI code.The simpliest way is to run

```
$ mpiexec python <script>.py
```

For example

```
$ mpiexec python hello_world.py
```

Examples

Hello world!

```
from mpi4py import MPI

comm = MPI.COMM_WORLD

print "Hello! I'm rank %d from %d running in total..." % (comm.rank, comm.size)

comm.Barrier()   # wait for everybody to synchronize
```

Collective Communication with NumPy arrays

```
from mpi4py import MPI
from __future__ import division
import numpy as np

comm = MPI.COMM_WORLD

print("-"*78)
print(" Running on %d cores" % comm.size)
print("-"*78)

comm.Barrier()

# Prepare a vector of N=5 elements to be broadcasted...
N = 5
if comm.rank == 0:
    A = np.arange(N, dtype=np.float64)    # rank 0 has proper data
else:
   A = np.empty(N, dtype=np.float64)    # all other just an empty array

# Broadcast A from rank 0 to everybody
comm.Bcast( [A, MPI.DOUBLE] )

# Everybody should now have the same...
print "[%02d] %s" % (comm.rank, A)
```

Execute the above code as:

```
$ qsub -q qexp -l select=4:ncpus=16:mpiprocs=16:ompthreads=1 -I

$ module load python openmpi

$ mpiexec -bycore -bind-to-core python hello_world.py
```

In this example, we run MPI4Py enabled code on 4 nodes, 16 cores per node (total of 64 processes), each python process is bound to a different core. More examples and documentation can be found on MPI for Python webpage.