MPI4PY

- The Problem
- Warning
- Step-by-step guide
  - Make a directory for all your virtualenv projects
  - Decide on your compiler/MPI and know where you are
  - Run virtualenv and install mpi4py
  - Installing for another combination
  - Using mpi4py
  - Launching MPI jobs
  - Notes for Deneb
    - MPI4PY with Intel Infiniband/OmniPath
    - Different architectures and the GPU nodes

This article explains how to install mpi4py in your home directory on the cluster in such a way that it is possible to use different clusters and/or compilers and MPI libraries.

The Problem

mpi4py needs to be built for a specific combination of the following:

- Compiler
- MPI flavour
- Interconnect
- Python version

This means that if installed with `pip install --user mpi4py` it will be tied to the exact combination of the aforementioned requirements used at install time.

The solution to this is to use the Python virtualenv package which allows multiple python environments to co-exists.

Warning

⚠️ Warning

mpi4py doesn’t work anymore with the latest gcc, mvapich, python combination:

1) gcc/6.4.0  2) mvapich2/2.3rc2  3) python/3.6.5

Step-by-step guide

Make a directory for all your virtualenv projects

```bash
$ cd
$ mkdir virtualenv
```

Decide on your compiler/MPI and know where you are

For each combination we need to create a directory and so we choose the following naming convention:

```
SYS_TYPE _ compiler _ MPI
```

e.g.
$ cd virtualenv
$ mkdir ${SYS_TYPE}_gcc_mvapich
$ ls
x86_E5v2_IntelIB_gcc_mvapich

Note that for Intel there is only one MPI available so ${SYS_TYPE}_intel will suffice as a name

$SYS_TYPE identifies the hardware type and looks something like x86_E5v2_IntelIB

Run virtualenv and install mpi4py

First check that the correct modules have been loaded!

$ module load gcc mvapich2 python
$ module list
Currently Loaded Modules:
1) gcc/5.4.0 2) mvapich2/2.2 3) python/3.6.1

Now tell virtualenv to use the appropriate directory and python version (2 or 3)

$ virtualenv -p python3 --system-site-packages x86_E5v2_IntelIB_gcc_mvapich
Running virtualenv with interpreter /ssoft/spack/cornalin/v1/opt/spack/linux-rhel7-x86_E5v2_IntelIB/gcc-5.4.0/python-3.6.1-ycwnkqkorgcaoh3eb7vcwg5nc6i2312/bin/python3
Using base prefix /ssoft/spack/cornalin/v1/opt/spack/linux-rhel7-x86_E5v2_IntelIB/gcc-5.4.0/python-3.6.1-ycwnkqkorgcaoh3eb7vcwg5nc6i2312'
New python executable in /home/user/virtualenv/x86_E5v2_IntelIB_gcc_mvapich/bin/python
Also creating executable in /home/user/virtualenv/x86_E5v2_IntelIB_gcc_mvapich/bin/python
Installing setuptools, pip, wheel...done.

And change to the newly created virtual environment

$ source x86_E5v2_IntelIB_gcc_mvapich/bin/activate
(x86_E5v2_IntelIB_gcc_mvapich) [user@cluster ]$

Note that the prompt has changed as a reminder!

Now we can install mpi4py using the "--no-cache-dir" option to make sure that it always get rebuilt correctly

(x86_E5v2_IntelIB_gcc_mvapich) [user@cluster]$$ pip install --no-cache-dir mpi4py
Collecting mpi4py
Using cached mpi4py-2.0.0.tar.gz
Building wheels for collected packages: mpi4py
Running setup.py bdist_wheel for mpi4py ... done
Stored in directory: /home/user/.cache/pip/wheels/e4/54/93/c216e777e447e0261d961c6480f5cb0d8284e7aa1920df8fc0
Successfully built mpi4py
Installing collected packages: mpi4py
Successfully installed mpi4py-2.0.0

To leave the virtual environment simply type "deactivate"
Installing for another combination

We can repeat the above process for as many permutations as we want:

$ mkdir ${SYS_TYPE}_intel
$ module purge
$ module load intel intel-mpi python
$ module list
Currently Loaded Modules:
1) intel/17.0.2 2) intel-mpi/2017.2.174 3) python/3.6.1

$ virtualenv -p python3 x86_E5v2_IntelIB_intel
Using base prefix '/ssoft/spack/cornalin/v1/opt/spack/linux-rhel7-x86_E5v2_IntelIB/intel-17.0.2/python-3.6.1-cl7lfy7srj45sz4owdeaeagy55jofz5q'
New python executable in /home/user/virtualenv/x86_E5v2_IntelIB_intel/bin/python3
Also creating executable in /home/user/virtualenv/x86_E5v2_IntelIB_intel/bin/python
Installing setuptools, pip, wheel...done.
$ source x86_E5v2_IntelIB_intel/bin/activate
(x86_E5v2_IntelIB_intel) [user@cluster ]$ pip install --no-cache-dir mpi4py
Collecting mpi4py
Installing collected packages: mpi4py
Successfully installed mpi4py-2.0.0

Using mpi4py

When you want to use mpi4py you now need to load the appropriate virtual environment as well as loading the corresponding modules:

$ module load intel intel-mpi python
$ source virtualenv/x86_E5v2_IntelIB_intel/bin/activate
(x86_E5v2_IntelIB_intel) [eroche@deneb2 ~]$ python
[GCC Intel(R) C++ gcc 4.8.5 mode] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from mpi4py import MPI
>>> comm = MPI.COMM_WORLD
>>> print("Hello! I'm rank %d from %d running in total..." % (comm.rank, comm.size))
Hello! I'm rank 0 from 1 running in total...

Note how virtualenv knows which python is associated with the environment so simply typing "python" is sufficient.

The same applies for batch scripts - just source the virtualenv after loading your modules.

Launching MPI jobs
As with traditional MPI jobs you need to use `srun` to correctly launch the jobs:

```
srun python mympicode.py
```

Failure to use `srun` will result in only one rank being launched.

**Notes for Deneb**

**MPI4PY with Intel Infiniband/Omnipath**

By default MPI4PY is not fully compatible with the Interconnect on the Deneb cluster due to the use of some MPI 3.0 calls which are not supported. The usual symptom is that communications between ranks will block or fail. Please see the following page for the details: [https://software.intel.com/en-us/articles/python-mpi4py-on-intel-true-scale-and-omni-path-clusters](https://software.intel.com/en-us/articles/python-mpi4py-on-intel-true-scale-and-omni-path-clusters)

The solution is to set the following variable just after importing MPI4PY:

```
mpi4py.rc.recv_mprobe = False
```

A more explicit example is:

```
import mpi4py
mpi4py.rc.recv_mprobe = False
from mpi4py import MPI

comm = MPI.COMM_WORLD
```

When using Intel MPI (module load intel-mpi) it is also possible to work around the issue by changing the fabric protocol via "`export I_MPI_FABRICS=shm:ofa`". This is not possible for MVAPICH2 nor OpenMPI.

**Different architectures and the GPU nodes**

Deneb is a heterogeneous cluster with the following `$SYS_TYPE`

- `x86_E5v2_IntelIB`
- `x86_E5v3_IntelIB`
- `x86_E5v2_Mellanox_GPU`

The first two are cross compatible for mpi4py but if using the GPU nodes you should change to the appropriate SYS_TYPE before configuring mpi4py.
$ smodules -s x86_E5v2_Mellanox_GPU -v
[INFO] S+L release: stable
[INFO] S+L systype: x86_E5v2_Mellanox_GPU
[INFO] S+L engaged!
$ echo $SYS_TYPE
x86_E5v2_Mellanox_GPU