BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Introduction to Python in HPC

Omar Padron
National Center for Supercomputing Applications
University of Illinois at Urbana Champaign
opadron@illinois.edu

NCSA

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

CRAY

Outline

- What is Python?
  - Why Python for HPC?
- Python on Blue Waters
  - How to load it
  - Available Modules
- Known Limitations
- Where to Get More information

What is Python?

- Modern programming language with a succinct and intuitive syntax
- Has grown over the years to offer an extensive software ecosystem
- Interprets JIT-compiled byte-code, but can take full advantage of AOT-compiled code
  - using python libraries with AOT code (numpy, scipy)
  - interfacing seamlessly with AOT code (CPython API)
  - generating native modules with/from AOT code (cython)

Why Python for HPC?

- When you want to maximize *productivity* (not necessarily performance)
- Mature language with large user base
- Huge collection of freely available software libraries
  - High Performance Computing
    - Engineering, Optimization, Differential Equations
    - Scientific Datasets, Analysis, Visualization
  - General-purpose computing
    - web apps, GUIs, databases, and tons more
- Python combines the best of both JIT and AOT code.
  - Write performance critical loops and kernels in C/FORTRAN
  - Write high level logic and “boiler plate” in Python

Python on Blue Waters

- Blue Waters Python Software Stack (bw-python)
  - Over 60 python modules supporting HPC applications
  - Implements the Scipy Stack Specification (http://www.scipy.org/stackspec.html)
  - Provides extensions for parallel programming (mpi4py, pycuda) and scientific data IO (h5py, pycuda)
  - Numerical routines linked from ACML
  - Built specifically for running jobs on the compute nodes

```
module swap PrgEnv-cray PrgEnv-gnu
module load bw-python
python
```

```
Python 2.7.5 (default, Jun  2 2014, 02:41:21)
[GCC 4.8.2 20131016 (Cray Inc.)] on Blue Waters
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Portal Documentation: https://bluewaters.ncsa.illinois.edu/python

Python on Blue Waters – Available Modules

- Numpy
  - Fast numerical array datatype
  - Array operations
  - Core foundation of numerical Python libraries

```
>>> b = np.ones(4) + 1
>>> a - b
array([-1.,  0.,  1.,  2.])

>>> a * b
array([ 2.,  4.,  6.,  8.])

>>> j = np.arange(5)
>>> 2**(j + 1) - j
array([ 2,  3,  6, 13, 28])
```

- http://www.numpy.org/

Python on Blue Waters – Available Modules

- Scipy
  - High-level scientific computing routines for numpy arrays
    - Linear algebra, FFT, optimizations
    - Statistics, interpolation, integration
  - Used together with numpy as an alternative to MATLAB

```
>>> from scipy import linalg
>>> arr = np.array([[1, 2],
...                 [3, 4]])
>>> linalg.det(arr)
-2.0

>>> arr = np.array([[3, 2],
...                 [6, 4]])
>>> linalg.det(arr)
0.0
```

- http://www.scipy.org

Python on Blue Waters – Available Modules

- Cython
  - Generates native Python modules in C
  - Create new AOT code or wrap existing code for Python
  - Limited OpenMP support

```
cdef extern from "math.h":
    double cos(double arg)

def cos_func(arg):
    return cos(arg)
```

```
>>> from cos_module import cos_func as cos
>>> cos(1.0)
0.5403023058681398

>>> cos(0.0)
1.0

>>> cos(3.14159265359)
-1.0
```

- http://cython.org

Python on Blue Waters – Available Modules

- mpi4py
  - Python wrapper around MPI for parallel applications
  - Provide generic Python object serialization as well as fast transfers based on numpy

```
from mpi4py import MPI
import numpy

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

# automatic MPI datatype discovery
if rank == 0:
    data = numpy.arange(100)
    comm.Send(data, dest=1, tag=13)
elif rank == 1:
    data = numpy.empty(100)
    comm.Recv(data, source=0, tag=13)
```

- http://mpi4py.scipy.org/

Python on Blue Waters – Available Modules

- pycuda
  - Python wrapper around CUDA for GPU-enabled applications
  - Kernel call and dispatch capabilities integrated with numpy

```
import pycuda.driver as drv
import pycuda.tools import pycuda.autoinit
import numpy import numpy.linalg as la
from pycuda.compiler import SourceModule

mod = SourceModule("""
__global__ void multiply_them(float *dest,
                              float *a,
                              float *b) {
    const int i = threadIdx.x;
    dest[i] = a[i] * b[i];
}
"""
)

multiply_them = mod.get_function("multiply_them")
```

- http://mathema.tician.de/software/pycuda/

Python on Blue Waters – Available Modules

- pycuda
  - Python wrapper around CUDA for GPU-enabled applications
  - Kernel call and dispatch capabilities integrated with numpy

```
a = numpy.random.randn(400).astype(numpy.float32)
b = numpy.random.randn(400).astype(numpy.float32)
dest = numpy.zeros_like(a)

multiply_them(
    drv.Out(dest), drv.In(a), drv.In(b),
    block=(400,1,1)
)

print dest-a*b
```

- http://mathema.tician.de/software/pycuda/

Python on Blue Waters – Available Modules

- Plenty more
  - ipython – interactive interpreter
  - f2py – wrap FORTRAN code
  - sympy – symbolic algebra
  - scikit.learn – machine learning and data mining algorithms
  - visualization with matplotlib, VTK, visit, paraview

Known Limitations

- Global Interpreter Lock
  - Serializes Python object access
  - Must work around using multiprocessing, or compiled code
- Parallel Import Problem
  - Many processes hitting the same files on the file system
    - Python modules and shared libraries
- Need to run MPI applications with aprun
  - Even if only using a single rank

Where to Get More Information

- https://www.python.org/ - Python website
- https://pypi.python.org/pypi - over 40 *thousand* available packages
- https://scipy-lectures.github.io/ - tutorials and notes for scientists

- https://bluewaters.ncsa.illinois.edu/python - our portal documentation

- And Your Blue Waters SEAS Team
  - Let us know what other questions we can answer for you!

Thank you!

Omar Padron
opadron@illinois.edu