Hewlett Packard Enterprise

# Python and Frameworks

LUMI Advanced Workshop

March 5–7, 2025

# Agenda

- Some notes relevant to using Python on the HPE Cray EX architecture
- Available Python environment
- Launching an application
- NetCDF from Python
- Future support for python profiling

 

# Not covered in this session

- Deep dives into mpi4py and numpy.
- Improving communication of frameworks like tensorflow or pytorch.

- Further information:
  ARCHER2 Python guide https://docs.archer2.ac.uk/user-guide/python
  LUMI Python: https://docs.lumi-supercomputer.eu/software/installing/python/
  LUMI PyTorch guide https://lumi-supercomputer.github.io/LUMI-EasyBuild-docs/p/PyTorch/

# Python Environment

- Do not use system python installations such as /usr/bin/python3 (v3.6.15)
- Load a cray-python module instead

```
$> module avail cray-python
   cray-python/3.9.12.1    cray-python/3.10.10    cray-python/3.11.7 (D)
   cray-python/3.9.13.1    cray-python/3.11.5
```

- Several packages are preinstalled :

```
$> pip list | grep -E "numpy|pandas|mpi4py”
   mpi4py              3.1.4
   numpy               1.24.4
   pandas              1.5.3
```

- To install more packages
  1. First set `PYTHONUSERBASE=$PWD/.local`
  2. Use `pip install --user <package>`
  3. Update the PYTHONPATH and PATH accordingly if needed

  - or use a virtual environment…

# Using a virtual environment based on cray-python

- For example, assume we want to take cray-python as a basis and add matplotlib

```
> module load cray-python
> python -m venv --system-site-packages craympl-venv
> source craympl-venv/bin/activate
(craympl-venv) > which pip
/tmp/craympl-venv/bin/pip
(craympl-venv) > pip install matplotlib
Collecting matplotlib
. . .
srun –n 8 python myapp.py
```

- The `--system-site-packages` option gives the venv access to the system site packages directory
- The resulting venv can now use the cray-python modules such as mpi4py

# Launch a Python script

```
$> srun -n 1 python script.py
```

- Use `srun` to launch a python script from within an interactive session or a batch script. (In particular large frameworks like tensorflow or pytorch. )
- Also if the script does not contain apparent multiprocessing/multithreading.
- Be careful with python multiprocessing, prefer to use psutil module to affinity inherited from `srun`
- (https://psutil.readthedocs.io/en/latest/index.html#psutil.Process.cpu_affinity)

```
$> srun -n 1 --cpu_bind=core python multi.py
Process psutil.Process(pid=66866, name='python', status='running', started='18:49:34') has affinity [0,… 255]
Process psutil.Process(pid=66866, name='python', status='running', started='18:49:34') has affinity [0,… 255]

$>  srun -n 1  -c 2 --cpu_bind=core python multi.py
Process psutil.Process(pid=67518, name='python', status='running', started='18:52:29') has affinity [0, 128]
Process psutil.Process(pid=67519, name='python', status='running', started='18:52:29') has affinity [0, 128]
```

- Use `psutil.Process().cpu_affinity()` to gather information.

# Launch an mpi4py Python script

```
$> srun -n 4 --cpu_bind=map_cpu:1,3,5,7  python hello.py

Process psutil.Process(pid=64648, name='python', status='running',
started='18:35:57') has affinity [1]
Hello world from node nid003404, rank 0 out of 4.

Process psutil.Process(pid=64650, name='python', status='running',
started='18:35:57') has affinity [3]
Hello world from node nid003404, rank 2 out of 4.

…
```

```python
import psutil
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
procname = MPI.Get_processor_name()

print(f"Process {psutil.Process()} has affinity \
{psutil.Process().cpu_affinity()}")

print("Hello world from node {}, rank {:d} out of \
{:d}.".format(procname, rank, size))
```

- Launch the mpi4py program with `srun`.
- mpi4py will use cray-mpich which in turn is configured for SLURM.
- If you see "… attempting to use MPI before initialization … " it could be related to mpi4py being built with GCC. Try `LD_PRELOAD=/opt/cray/pe/lib64/libmpi_gnu_123.so` or another GNU cray-mpich version.
- Affinity can be checked or set with `psutil.Process().cpu_affinity()`

# GPU-aware MPI in Python

- If the variable `MPICH_GPU_SUPPORT_ENABLED` is set, then MPI assumes that you link with the GTL library to enable the GPU support in MPI
  - Error message

```
MPIDI_CRAY_init: GPU_SUPPORT_ENABLED is requested, but GTL library is not linked
```

- To link the GTL library in python

Before the import of mpi4py

```python
from os import environ
if environ.get("MPICH_GPU_SUPPORT_ENABLED", False):
    from ctypes import CDLL, RTLD_GLOBAL
    CDLL(f"{environ.get('CRAY_MPICH_ROOTDIR')}/gtl/lib/libmpi_gtl_hsa.so", mode=RTLD_GLOBAL)

from mpi4py import MPI
```

# Netcdf with Python

```
$> pip install --user netcdf4
```

- Install the `netcdf4` package with pip.
- Look for instance at min/max/avg of all variables in a netcdf file.

```
$> module load cray-python
$> pip install --user recursive-diff
$> srun ncdiff --recursive --rtol 1e-6 --atol 1e-8
-b lhs rhs
```

- Netcdf files and directories can be compared in python recursively with the `ncdiff` utility.
- Pay attention to automatic installation of required package versions. A new numpy installation is not necessarily linked against cray-libsci

```python
from netCDF4 import Dataset
import numpy as np
import pandas as pd
import sys

'''
Loops over all variables present in the netcdf file
specified on the command line and
prints min/max/avg for every variable.
'''

filename = sys.argv[1]

tmp = Dataset(filename, "r", format="NETCDF4")

mins = []
maxs = []
avgs = []
for v in tmp.variables.keys():
            tmp2 = np.frombuffer(tmp[v][:])
            mins.append(tmp2.min())
            maxs.append(tmp2.max())
            avgs.append(tmp2.sum()/tmp2.size)

nc_df = pd.DataFrame(data={'min':mins, 'max':maxs,
'avg':avgs}, index=tmp.variables.keys())
print(nc_df)
```

# Perftools for Python - Sampling

```
$> module load perftools-preload
$> module load cray-python
$> srun -n 4 pat_run `which python` my_script.py
```

- Load the perftools-preload and the the cray-python modules.
- Use `pat_run`.

```
$> pat_report -v -o myrep <exp-dir>
$> pat_report -v -O ct+src -o myrep.ct <exp-dir>
```

- Generate call tree report in addition to default one.
- Python module must be loaded when `pat_report` is invocated.
- Python methods from the source code are prepended with `python.*`
- Check the `pat_run` man page for more details.

# Output: perftools for python using sampling (MPI application using mpi4py)

```
CrayPat/X:  Version 23.09.0 Revision 6034a9414 sles15.4_x86_64  08/01/23 18:55:19
Number of PEs (MPI ranks):    4
Numbers of PEs per Node:     4
Numbers of Threads per PE:    1
Number of Cores per Socket:  64
Execution start time:  Mon Jun  3 17:14:48 2025
System name and speed:  nid001067  2.456 GHz (nominal)
AMD   Milan              CPU  Family: 25  Model:  1  Stepping:  1

Current path to data file:
  /project/project_462000031/bracconi/perftools-python/python+224849-867507628s
```

General job information

```
Table 1:  Calltree View with Callsite Line Numbers

  Samp% |  Samp | Calltree  |      | PE=HIDE
 100.0% | 441.5 | Total
|-----------------------------------------------------------------------------
|  41.3% | 182.5 | read
|  22.1% |  97.5 | python.main:heat-p2p.py:line.125
||----------------------------------------------------------------------------
||  13.3% |  58.8 | python.iterate:heat-p2p.py:line.82
|||---------------------------------------------------------------------------
3||   5.6% |  24.8 | python.evolve:heat-p2p.py:line.51
||||--------------------------------------------------------------------------
4|||   4.4% |  19.2 | python.evolve:heat-p2p.py:line.51(exclusive)
4|||   1.2% |   5.5 | __memcpy_avx_unaligned_erms
||||==========================================================================
3||   4.2% |  18.5 | python.evolve:heat-p2p.py:line.53
3||   2.5% |  11.2 | python.evolve:heat-p2p.py:line.50
||||--------------------------------------------------------------------------
4|||   1.5% |   6.8 | __memcpy_avx_unaligned_erms
4|||   1.0% |   4.2 | python.evolve:heat-p2p.py:line.50(exclusive)
||||==========================================================================
3||   1.0% |   4.2 | python.evolve:heat-p2p.py:line.55
4||        |       | __memcpy_avx_unaligned_erms
|||===========================================================================
||   8.8% |  38.8 | python.iterate:heat-p2p.py:line.81
|||---------------------------------------------------------------------------
3||   6.1% |  26.8 | python.exchange:heat-p2p.py:line.73
4||   5.9% |  26.2 |  MPI_Sendrecv
3||   2.7% |  12.0 | python.exchange:heat-p2p.py:line.77
4||        |       | MPI_Sendrecv
||============================================================================
|  11.0% |  48.8 | dlopen
|   5.6% |  24.8 | python.main:heat-p2p.py:line.99
|        |       | MPI_Bcast
|   4.1% |  18.0 | python.read_style_directory:core.py:line.211
|   3.9% |  17.0 | python._rc_params_in_file:__init__.py:line.872
3        |       |  read
|   1.8% |   8.0 | python._check_versions:__init__.py:line.248
|   1.1% |   5.0 |  dlopen
...
```

Line number

File name

Function name

MPI calls (mpi4py)

Level in call tree

Python profiling summary
Notice python. in front of functions

# Perftools for Python - Tracing

- To trace callables, use the API's `PAT_trace` decorator:

```
import pat_api # **pat_run** ensures this module is found
@pat_api.PAT_trace
def user_function():
```

  - See https://cpe.ext.hpe.com/docs/24.03/performance-tools/man1/pat_api.html for more details on `pat_api`

- To trace entire modules and programming models:
  - PAT_RT_TRACE_PYTHON_GROUPS defines a list of predefined trace groups to trace (similar groups used in the `-g` option in `pat_build`)
  - PAT_RT_TRACE_PYTHON_MODULES defines a list of Python modules to trace

- The same procedure of the sampling, add `-w` and `–g` flags to `pat_run`:

```
$> srun -n 4 pat_run –w –g mpi `which python` my_script.py
```

# Output: perftools for python using tracing (MPI application using mpi4py)

```
CrayPat/X:  Version 23.09.0 Revision 6034a9414 sles15.4_x86_64  08/01/23 18:55:19
Number of PEs (MPI ranks):    4
Numbers of PEs per Node:     4
Numbers of Threads per PE:    1
Number of Cores per Socket:  64
Execution start time:  Mon Jun  3 17:14:48 2025
System name and speed:  nid001067  2.456 GHz (nominal)
AMD   Milan              CPU  Family: 25  Model:  1  Stepping:  1

Current path to data file:
  /project/project_462000031/bracconi/perftools-python/python+224849-567123549t
```

General job information

```
Table 1:  Calltree View with Callsite Line Numbers

  Time% |     Time | Calls | Calltree |        |      | PE=HIDE/
 100.0% | 8.057850 |    -- | Total
|-----------------------------------------------------------------------
|  64.2% | 5.170894 |   1.0 | main:python.c:line.14
|  18.9% | 1.519801 |  49.5 | dlopen
|   8.3% | 0.666561 |    -- | [R]__main__.write_field:_pat_base.py:line.82
|        |          |       | [R]__main__.exchange:_pat_base.py:line.82
3        |          |       |  [R]__main__.evolve:_pat_base.py:line.82
4        |          |       |   [R]__main__.iterate:_pat_base.py:line.82
5        |          |       |    [R]__main__.main:_pat_base.py:line.83
||||||-----------------------------------------------------------------
6|||||  4.1% | 0.327106 |    -- | python.main:heat-p2p-tracing.py:line.132
7|||||       |          |       | [R]__main__.write_field:_pat_base.py:line.82
8|||||       |          |       |  [R]__main__.exchange:_pat_base.py:line.82
9|||||       |          |       |   [R]__main__.evolve:_pat_base.py:line.82
10||||       |          |       |    [R]__main__.iterate:_pat_base.py:line.82
11||||       |          |       |     [R]__main__.main:_pat_base.py:line.83
12||||       |          |       |      python.iterate:heat-p2p-tracing.py:line.88
13||||       |          |       |       [R]__main__.write_field:_pat_base.py:line.82
14||||       |          |       |        [R]__main__.exchange:_pat_base.py:line.82
15||||       |          |       |         [R]__main__.evolve:_pat_base.py:line.82
16||||       |          |       |          [R]__main__.iterate:_pat_base.py:line.82
17||||       |          |       |           [R]__main__.main:_pat_base.py:line.83
||||||||||||||||-------------------------------------------------------
18|||||||||||||||  2.9% | 0.230167 |    -- | python.exchange:heat-p2p-tracing.py:line.79
19|||||||||||||||  2.9% | 0.230155 | 200.0 | MPI_Sendrecv
18|||||||||||||||  1.2% | 0.096939 |    -- | python.exchange:heat-p2p-tracing.py:line.83
19|||||||||||||||  1.2% | 0.096938 | 200.0 | MPI_Sendrecv
||||||||||||||||=======================================================
6|||||  2.9% | 0.231056 |    -- | python.main:heat-p2p-tracing.py:line.106
7|||||  2.9% | 0.231052 |   1.5 | MPI_Bcast
6|||||  1.1% | 0.086481 |    -- | python.main:heat-p2p-tracing.py:line.129
7|||||  1.1% | 0.086481 |   0.2 | [R]__main__.write_field
||||||=================================================================
|   7.5% | 0.607677 | 200.0 | [R]__main__.evolve
|   1.0% | 0.080417 |   1.0 | [R]__main__.main
```

Python profiling summary

Line number

Function name

File name

MPI calls (mpi4py)

Level in call tree

# New Python support available in the next PE versions

- Tracing profiling with pat_run
  - Tensorflow  Pytorch symbol profiling
  - Details at https://cpe.ext.hpe.com/docs/latest/performance-tools/man1/pat_python.html#pat-python
- Debugging via gdb4hpc
  - Details at https://cpe.ext.hpe.com/docs/latest/debugging-tools/gdb4hpc/man/help.html#python

 

# Python

- Low-level Python and Cython Bindings for HIP provided by HIP Python

➢ https://github.com/ROCm/hip-python
➢ https://rocm.docs.amd.com/projects/hip-python/en/latest/

There are many complex frameworks and libraries that implement GPU support
- CuPy (https://cupy.dev/) provides AMD support for computing with Python
- PyTorch

# Questions?