Python Workflows on ManeFrame II (M2)

Robert Kalescky
HPC Applications Scientist

March 18, 2021

Research and Data Sciences Services
Office of Information Technology
Center for Research Computing
Southern Methodist University

Outline

Research Support

Research and Data Science Services

- Provide research computing support, consultations, and collaborations
- Data Science - Dr. Eric Godat
- High-Performance Computing - Dr. Robert Kalescky
- Custom Devices (IOT, wearables, etc.) - Guillermo Vasquez

Center for Research Computing (CRC)

- Maintains our primary shared resource for research computing, ManeFrame II (M2), in collaboration with OIT
- Provides research computing tools, support, and training to all faculty, staff, and students using research computing resources
- `www.smu.edu/crc` has documentation and news
- help@smu.edu or rkalescky@smu.edu for help
- Request an account at `www.smu.edu/crc`

Spring 2021 CRC HPC Workshop Series

Date	Workshop
March 4	ManeFrame II (M2) Introduction
March 11	Intel oneAPI HPC Toolkit Overview
March 18	Python Workflows on ManeFrame II (M2)
March 25	NVIDIA HPC SDK Overview
April 1	ManeFrame II (M2) Introduction
April 8	R Workflows on ManeFrame II (M2)
April 15	Introduction to OpenMP and OpenACC
April 22	Introduction to MPI
April 29	Introduction to Standard C++ Parallel Algorithms

Table 1: Workshops will be held each Thursday via Zoom from 12:00 to 2:00 PM. Register to receive Zoom coordinates. Sessions will be recorded and posted along with session materials. Register!

ManeFrame II (M2)

ManeFrame II (M2) Node Types

Type	Quantity	Cores	Memory [GB]	Additional Resources
Standard-Memory	176	36	256	
Medium-Memory-1	35	36	768	
Medium-Memory-2	4	24	768	3 TB SSD local scratch
High-Memory-1	5	36	1,536	
High-Memory-2	6	40	1,536	3 TB SSD local scratch
GPGPU-1	36	36	256	NVIDIA P100 GPU has 3,584 CUDA cores and 16 GB CoWoS
MIC-1	36	64	384	16 GB of high bandwidth (400 GB/s) stacked memory
VDI	5	36	256	NVIDIA Quadro M5000 GPU
v100x8	3	36	768	8 NVIDIA V100 GPUs with 5,120 CUDA cores and 32 GB CoWoS
Faculty Partner Nodes	3			Various research specific NVIDIA GPU configurations
ManeFrame II	354	11,276	120 TB	2.8 PB storage and InfiniBand network

ManeFrame II (M2) HPC OnDemand Web Portal

ManeFrame II (M2) HPC OnDemand Web Portal

- Provides an integrated single access point for HPC resources on the ManeFrame II (M2) supercomputer
- Accessing the Portal:
    - Access to the HPC portal requires an existing M2 account
    - Go to `hpc.smu.edu`
    - Sign in using your SMU ID and SMU password
- HPC Portal Documentation
- HPC Portal Walkthrough Video

Slurm

Serial Python Script: π Monte Carlo

```python
import random, sys

def monte_carlo_pi(points):
    return 4 * sum(1 for _ in range(points) if random.random()**2 + random.random()**2 < 1) / float(points)

if __name__ == "__main__":
    print(monte_carlo_pi(int(sys.argv[1])))
```

Listing 1: Serial algorithm to estimate the value of Pi.

Parallel Python Script: π Monte Carlo

```python
import random, sys
import multiprocessing as mp

def check_point(points):
    return sum(1 for _ in range(points) if random.random()**2 + random.random()**2 < 1)

def parallel_monte_carlo_pi(points, cores):
    points_per_core = int(points / cores)
    n = [points_per_core] * cores
    n[0] = points_per_core + (points - (points_per_core * cores))
    pool = mp.Pool(processes = cores)
    results = pool.map(check_point, n)
    return 4 * sum(results) / float(points)

if __name__ == "__main__":
    print(parallel_monte_carlo_pi(int(sys.argv[1]), int(sys.argv[2])))
```

Listing 2: Parallel algorithm to estimate the value of Pi.

Interactive Sessions

```
module load python
srun -p htc --mem=6G --pty $SHELL
```

Listing 3: Using `srun` to log into a compute node to run commands interactively.

Using `srun`

```
srun -p htc --mem=6G  python pi_monte_carlo.py 1000
```

Listing 4: Using `srun` to run commands directly on a compute node.

Using sbatch --wrap

```
sbatch -p htc --mem=6G  --wrap "sleep 30; time python pi_monte_carlo.py 1000"
```

Listing 5: Using `sbatch --wrap` wrap a commands in an `sbatch` script that is then submitted to the queue can run non-interactively.

Using sbatch

```
#!/bin/bash
#SBATCH -J python
#SBATCH -o python_%j.out
#SBATCH -p htc
#SBATCH --mem=6G

module purge
module load python

time python pi_monte_carlo.py 1000
```

Listing 6: Using sbatch run serial computations via an sbatch script.

Using sbatch

```bash
#!/bin/bash
#SBATCH -J pi
#SBATCH -o pi_%j.out
#SBATCH -p development
#SBATCH -N 1
#SBATCH --cpus-per-task=2
#SBATCH --mem=6G

module purge
module load python

time python pi_monte_carlo_shared.py 10000000 ${SLURM_CPUS_PER_TASK}
```

Listing 7: Using sbatch run parallel computations via an sbatch script.

Using sbatch --array

```bash
#!/bin/bash
#SBATCH -J pi_array
#SBATCH -o pi_array_%a-%A.out
#SBATCH --array=1-4%2
#SBATCH -p development
#SBATCH --mem=6G

module purge
module load python

time python pi_monte_carlo.py $((100**${SLURM_ARRAY_TASK_ID}))
```

Listing 8: Using `sbatch --array` run parallel jobs via a single `sbatch` script.

Python Environments

Managing Python Installations

- Python environments allow users to use specific versions of Python with the packages of their choice
- The packages can be installed via `conda` or `pip`, depending on the type of environment being used

Anaconda Environments

- Anaconda environments are managed using `conda`, which is available via Anaconda installations such as `python/2` and `python/3` on ManeFrame II.

Anaconda Environments Example

```
module purge
module load python/3
conda create -y -n jupyter_37 -c conda-forge jupyterlab python=3.7
source activate ~/.conda/envs/jupyter_37
which python3
python3 --version
deactivate
```

Listing 9: A specific version of Python is installed along with the JupyterLab package and its dependencies. The new environment is then loaded and then unloaded.

Python Virtual Environments

- Python 3 has native support for managing environments, which can be used with any Python 3 installation on ManeFrame II.

Python Virtual Environments Example

```
module purge
module load spack gcc-9.2
source <(spack module tcl loads --dependencies python@3.7%gcc@9.2)
python3 -m venv ~/.venv/jupyter_37
source ~/.venv/jupyter_37/bin/activate
pip3 install --upgrade pip
pip3 install --upgrade jupyterlab
which python3
python3 --version
deactivate
```

Listing 10: A specific installation of Python, in this case from Spack, is used and then JupyterLab and its dependencies are installed. The new environment is then loaded and then unloaded.

Containers

Containers Overview

- Containers offer the ability to run fully customized software stacks, e.g. based on different Linux distributions and versions
- Containers are not virtual machines, where an entire hardware platform is virtualized, rather containers share a common kernel and access to physical hardware resources
- Docker is a popular container platform in development and server environments
- Singularity is a popular container platform in HPC environments
    - Docker is not directly support on ManeFrame II
    - Singularity can consume and run Docker containers

Singularity Build Script Example

```
Bootstrap: docker                  # Base container source
From: ubuntu:18.04                 # Base container version

%post                              # Changes to container
export DEBIAN_FRONTEND=noninteractive
apt-get update
apt-get -y install\
 python3-pip\
 python3-numpy\
 python3-pandas
pip3 install\
 jupyterlab

%runscript                         # Default container command
python3
```

Listing 11: Example Singularity build file that uses Ubuntu 18.04 and Python3 with package installation via `apt` and `pip`.

Building Singularity Container Images

```
#!/usr/bin/env sh

module purge
module load singularity/3.5.3

srun -p development -x k001 -c 1 --mem=1G\
 --pty singularity build\
 --fakeroot python3.sif python3.singularity
```

Listing 12: Steps to build a Singularity container image. Note that building Singularity container images from build scripts on M2 requires permission. Request permission via help@smu.edu.

Docker Build Script Example

```
FROM ubuntu:18.04

ENV DEBIAN_FRONTEND noninteractive

RUN apt-get update &&\
 apt-get -y install\
 python3-pip\
 python3-numpy\
 python3-pandas

RUN pip3 install\
 jupyterlab

ENTRYPOINT ["python3"]
```

Listing 13: Example Dockerfile that uses Ubuntu 18.04 and Python3 with package installation via `apt` and `pip`.

Building and Converting Docker Container Images

```
#!/usr/bin/env sh

docker build -t python3:18.04 -f python3.dockerfile .
docker save -o python3_18_04.tar python3:18.04
scp python3_18_04.tar m2.smu.edu:~/workshops/examples/
```

Listing 14: Building the Docker container off of M2, exporting the container image, and uploading to M2.

```
#!/usr/bin/env sh

module purge
module load singularity/3.5.3.lua

srun -p development -x k001 -c 1 --mem=1G --pty singularity build python3_18_04.sif docker-archive://python3_18_04.tar
```

Listing 15: Converting the uploaded Docker container image to a Singularity container image.

Help?

Need help or have questions?

rkalescky@smu.edu