Parallel Programming with Python: Outline

Schedule

09:00 am - 09:15 am: Introduction
09:15 am - 10:30 am: Efficient Serial Programming using NumPy
10:30 am - 10:45 am: Break
10:45 am - 12:15 pm: Parallel Programming with IPyParallel
12:15 pm – 01:15 pm: Lunch
01:15 pm – 02:30 pm: Advanced Topics: Python Environment Management; MPI4PY
02:30 pm – 02:45 pm: Break
02:45 pm – 04:00 pm: Parallelize Your Code! (Hackathon/Consulting time)

Session 1: Serial Programming with Numpy (75 min)

Concepts:

- Timing
- Arrays vs. lists
- Numpy array attributes
- Loops vs. array operations
- In-place vs. out-of-place operations
- Array access patterns (inner vs. outer index)
- Useful numpy functions (max,min, dot products, etc.)
- Map & Reduce functions

Examples:

- timing.py (shows how to use the time module)
- initialization.py (demonstrates various ways of initializing numpy arrays)
- arrays_vs_lists.py (times examples carried out using array syntax vs lists)
- inplace.py (illustrates timing difference between in-place and out-of-place operations)
- noloops.py (example to show that explicit loops can really slow things down - use array syntax instead)
- Mapit.py : simple illustration of map

Exercises:

1. Convert a code written in list/loop notation into one written in array syntax
2. Rewrite the code provided using map/reduce functionality

Session 2: Parallel Programming with IPyParallel (90 min)

Concepts:

- Engine/Hub paradigm
- Variable assignment

- Pushing class definitions
- Parallel function definitions
- Syncing imports
- Map/Reduce in Parallel

Examples:

- Hello world (hello.py)
- Variables and assignment (assignment.py)
- Scatter/Gather (scatter_gather.py)
- Parallel function execution (parallel_functions.py)
- Computing Pi
- Using classes (push.py)

Exercises:

1. Modify assignment.py so that var1 receives the value n, where n = (Engine ID % 3).
2. Modify scatter_gather_ex.py as follows:
    a. Scatter list a to all even processors, assigning its values to the variable mylist
    b. Scatter list b to all odd processors, assigning its values to mylist
    c. Gather from all **processors** to create the list [0,1,0,1,2,3,4,9,....]
3. Modify parallel_functions.py as follows:
    a. Have squared return the tuple (x^2, process ID). You will need to import os within squared.
    b. Revise the formatting of the loop output to read:
    47^2 is 2209 according to process PID: 192159
4. Modify misc/trapezoid.py to carry out the calculation in parallel for each value of n used in the serial mode. Report the timing information for both serial and parallel calculations as we did with compute_pi.py
5. Modify misc/collatz_length.py to carry out the calculation in parallel. Examine the compute times in parallel and serial for different maximum values of n.

<u>Session 3:</u> Parallel Programming with MPI4PY and Using Virtualenv (75 mins)

Concepts:

- Distributed memory paradigm
- Point-to-point send/receive (pickling and numpy variants)
- Global operations (barrier & reduction; sum)
- Load-balancing

- Virtual environments

Examples:

- hello1.py (basic hello world)
- hello2.py (hello world with I/O done by rank 0)
- barrier.py (hello world with barrier example)
- token_pass.py (token passing)
- trapezoid.py (trapezoidal integration -- discuss and modify)
- Virtual environment setup walkthrough...

Exercises:

1. Modify barrier.py so that processes print “in order”
2. *(MAYBE)* Modify token_pass.py so that odd-numbered ranks only exchange a token with odd-numbered ranks and similarly so for even-numbered ranks
3. Complete the load-balancing in trapezoid.py

Miscellaneous Notes

Notes on getting ipython going:
salloc -N1 -n24 -t60 --qos=debug
ssh into headnode
module load intel
module load impi
module load python/3.5.1

ipcluster start -n 12 (takes a second)
Can either ssh into headnode a second time or else wait on the message:
[IPClusterStart] Engines appear to have started successfully
And then background the process (ctrl+z; bg enter)
Use top, “u username”; will see processes