CW 4

URL	https://erau.instructure.com/courses/178285/assignments/3591720
Files & media	ma553_cw4.pdf classwork4_files.zip
Course	MA 453 - High Performance Scientific Computing
Due	@10/14/2024 12:00 AM
Status	In progress
Type	Classwork
Points	100

Assignment Outline

In this hands-on exercises, you will learn how to use multiprocessing module and OpenMP functions in Cython to parallelize the Python codes for a single shared memory computer.

1. Download the file `classwork4_files.zip` from your Canvas course page. Save it in the appropriate work directory (`~/MA453/Classwork/cw4`) and unzip it. The code hello `parallel.py` spawns 10 different processes, each waits for 1 second and prints its system assigned PID. Run the code as:

   ```
   $ time python hello_ parallel.py
   ```

   Did you notice performance gain from this code? The order of execution for parallel processes is unpredictable. It depends on how OS schedules the tasks for parallel execution. Observe this behavior by running the code multiple times.

2. Copy the code hello `parallel.py` to `square.py` and modify it to calculate the square of all numbers $\{1, 2, \ldots, 10\}$ using a separate Process to calculate the square of each number and print out the result.

   ```
   from multiprocessing import Process

   def squared(x):
       print (' {0}x{0} = {1} computed from PID {2}'. format (x,x*x,os.getpid()))

   # create a list of parallel processes
   procs = [Process (target=squared, args=[x]) for x in range(1,11)]

   # start all processes
   for p in procs:
       p.start ()

   # wait for all tasks to finish
   for p in procs :
       p.join ()
   ```

   How is the order of execution for parallel processes?

3. The code `particle_simulator4.py` implements four different versions of `evolve()` function: Pure Python, NumPy, Cython and OpenMP.

   a. You already worked on the pure python and numpy in the Classwork 3.

   b. Here you will explore Cython and OpenMP versions. Why aren't we getting the best out of Cython function? Give a look to the file `cevolve. pyx`.
   The
   `cython` module `cevolve` uses the untyped version.

   c. Comment the lines 7-15 and uncomment the lines 18-34, rebuild the module and time the `benchmark()` again.

   ```
   def c_evolve(r_i, ang_speed_i, dt, nsteps):
           v_i = np. empty_like (r_i)
           for i in range (nsteps):
                   norm_i = np.sqrt((r_i ** 2).sum(axis=1))
           v_i=r_i[:, [1, 0]]
           v_i[:, 0] *=- 1
           v_i /= norm_i [:, np.newaxis]
           d_i =dt * ang_speed_i[:,np.newaxis] * v_i
           r_i += d_i
   ```

   and run

   ```
   $ python setup.py build_ext --inplace
   $ python-m timeit-s "from particle_simulator4 import benchmark" "benchmark(100, 'cython')"
   ```

then

```
def c_evolve (double [:, :] r_i, double[:] ang_speed_i,double dt, int nsteps):
    cdef int i, j, nparticles = r_i.shape[0]
    cdef double norm, x, y, vx, vy, dx, dy, ang_speed
    for i in range (nsteps):
        for j in range (nparticles):
            y=r_i[j, 0]
            y=r_i[j, 1]
            ang_speed = ang_speed_i[j]
            norm = sqrt (x ** 2 + y ** 2)
            vx=(-y)/norm
            vy = x/norm
            dx = dt * ang_speed * vx
            dy = dt * ang_speed * vy
            r_i[j, 0] += dx
            r_i[j, 1] += dy
```

4. Let us profile the Cython module `cevolve.pyx` with the annotated view option.

   ```
   cython -a cevolve.pyx
   ```

   Then, open the file `cevolve.html` and check which lines do have more interpreter-related calls.

   ```
   $ firefox cevolve.html
   ```

   The white lines corresponds translated C code, you can click these lines to see the code.

5. In the line `v_y=x/norm`, Cython checks that computed norm is not zero, otherwise it raises a `ZeroDivisorError`, and in the line `r_i[j,0]`, Cython checks if the indexes are within the bounds of the array. Add the following two lines before the function `c_evolve()` in the file `cevolve.pyx` and measure the timing of `benchmark(1000,'cython')`.

   ```
   @cython.boundscheck(False)
   @cython.cdivision(True)
   def c_evolve(double [:, :] r_i, double[:] w_i, double dt, int nsteps):
           ...
   ```

   and the BASH commands

   ```
    > python setup.py build_ext --inplace
    > python -m timeit -s "from particle_simulator4 import benchmark" "benchmark(100, 'cython')"
   ```

6. Now, check the OpenMP version of the `c_evolve()` function `c_evolve_openmp()` in the file `cevolve.pyx` did you see the difference in line 44?

   ```
   for j in prange(nparticles, nogil=True)
   ```

   Now, measure the timing of `benchmark(1000, 'openmp')`. Which has the best performance?

7. Record all the timings from steps (2), (3), (5) and (6) in a file `timing.txt` and send it to me using the mail command:

   ```
   mail -s "ma453:cw3" 453 < timings.txt
   ```

- [x] ~~Run the code~~ `hello_parallel.py` ~~multiple times to observe the unpredictable order of execution for parallel processes~~
- [x] ~~Modify~~ `hello_parallel.py` ~~to create~~ `square.py` ~~and calculate squares of numbers using separate processes~~
- [x] ~~Explore Cython and OpenMP versions in~~ `particle_simulator4.py`
- [x] ~~Modify~~ `cevolve.pyx` ~~by commenting lines 7-15 and uncommenting lines 18-34, then rebuild and time the~~ `benchmark()`
- [x] ~~Profile the Cython module~~ `cevolve.pyx` ~~with the annotated view option~~

- [x] ~~Add `boundscheck` and `cdivision` decorators to `c_evolve()` in `cevolve.pyx` and measure timing~~
- [x] ~~Check the OpenMP version of `c_evolve()` function and measure timing of `benchmark(1000, 'openmp')`~~
- [x] ~~Record all timings from steps 2, 3, 5, and 6 in `timing.txt` file~~
- [x] ~~Send `timing.txt` file via email using the provided mail command~~

Notes

- @October 14, 2024 8:33 PM I keep getting errors whenever I run `hello_parallel.py` and I am not sure why. This is the code straight from the professor; I did not edit it.
- I cannot run this on PowerShell—for now, I can run it other on MobaXterm, or Ubuntu (both are Unix environments, so all the codes are directly transferrable)
  - Also, in Ubuntu, I have to run the code with `python3` ; there are no aliases for `py` and `python` like there is in MobaXterm as well as PowerShell
- Finally, this command works in PowerShell

```
python -m timeit -s "from particle_simulator4 import benchmark" "benchmark(100, 'cython')"
```

- For some reason, the normal way to “cythonize” a script in `setup.py` was not working, I had to change it to

```
from Cython.Build import cythonize
from setuptools import setup, Extension

# Define the extension module
ext_modules = cythonize(Extension('cevolve', sources=['cevolve.pyx']))

# Setup script
setup(ext_modules=ext_modules)
```

Outputs

Step 1

```
kconfeiteiro@KCBeast:/mnt/c/Users/kconf/Downloads/venv_courses/Coursecodes/Fall 2024/ma553/classwork/cw4$ t
Spawned a new process from PID 120
Hello World, from process 0 with PID 121
Hello World, from process 1 with PID 122
Hello World, from process 2 with PID 123
Hello World, from process 3 with PID 124
Hello World, from process 4 with PID 125
Hello World, from process 5 with PID 126
Hello World, from process 6 with PID 127
Hello World, from process 7 with PID 128
Hello World, from process 8 with PID 129
Hello World, from process 9 with PID 130

real    0m1.073s
user    0m0.033s
sys     0m0.018s
```

Step 2

```
kconfeiteiro@KCBeast:/mnt/c/Users/kconf/Downloads/venv_courses/Coursecodes/Fall 2024/ma553/classwork/cw4$ t
 1x1 = 1 computed from PID 227
```

```
 2x2 = 4 computed from PID 228
 3x3 = 9 computed from PID 229
 4x4 = 16 computed from PID 230
 5x5 = 25 computed from PID 231
 6x6 = 36 computed from PID 232
 7x7 = 49 computed from PID 233
 8x8 = 64 computed from PID 234
 9x9 = 81 computed from PID 235
 10x10 = 100 computed from PID 236

real    0m0.053s
user    0m0.017s
sys     0m0.019s
```

Step 3

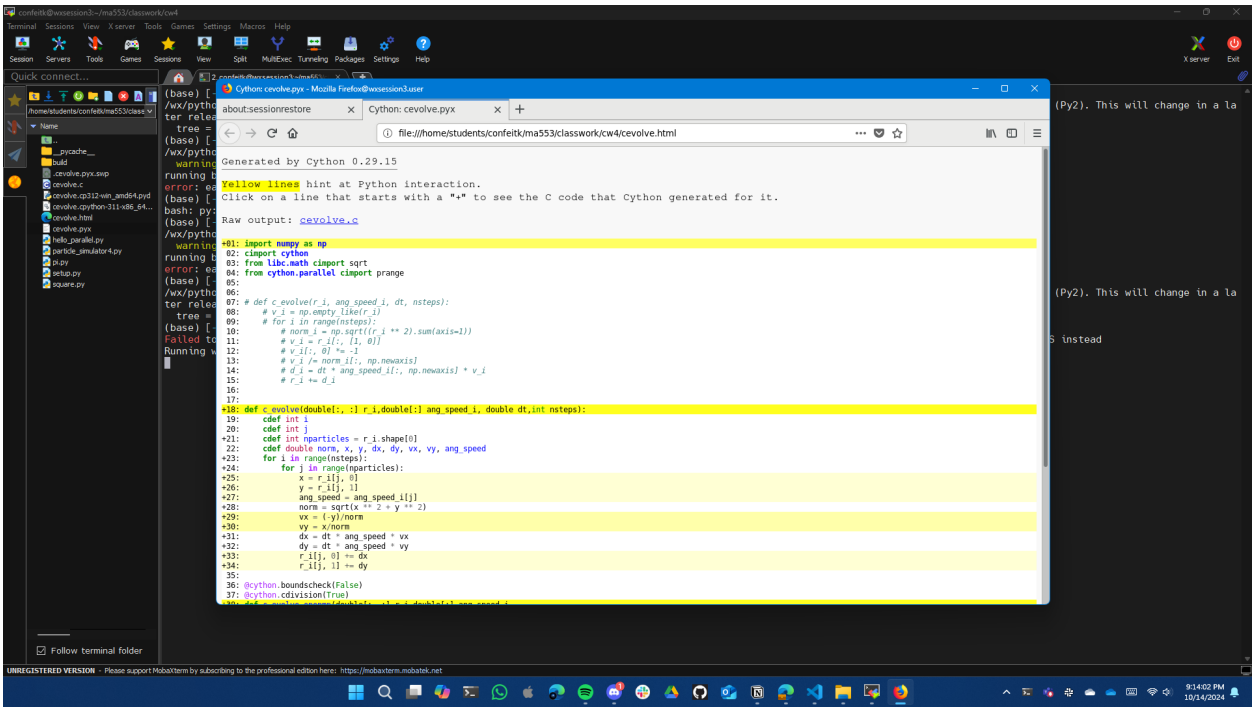
Part (1)

```
PS C:\Users\kconf\Downloads\venv_courses\Coursecodes\Fall 2024\ma553\classwork\cw4> python -m timeit -s "fr
5 loops, best of 5: 87.9 msec per loop
```

Part (2)

```
PS C:\Users\kconf\Downloads\venv_courses\Coursecodes\Fall 2024\ma553\classwork\cw4> python -m timeit -s "fr
10 loops, best of 5: 32.3 msec per loop
```

Step 4



Step 5

```
(base) [-bash confeitk@wxsession3 ~/ma553/classwork/cw4 ]$ python3 setup.py build_ext --inplace
running build_ext
```

```
building 'cevolve' extension
creating build
creating build/temp.linux-x86_64-3.7
gcc -pthread -B /wx/python/anaconda3/compiler_compat -Wl,--sysroot=/ -Wsign-compare -DNDEBUG -g -fwrapv -O3
creating build/lib.linux-x86_64-3.7
gcc -pthread -shared -B /wx/python/anaconda3/compiler_compat -L/wx/python/anaconda3/lib -Wl,-rpath=/wx/pyth
copying build/lib.linux-x86_64-3.7/cevolve.cpython-37m-x86_64-linux-gnu.so ->
(base) [-bash confeitk@wxsession3 ~/ma553/classwork/cw4 ]$ python3 -m timeit -s "from particle_simulator4 i
10 loops, best of 5: 23.6 msec per loop
```

Step 6

```
(base) [-bash confeitk@wxsession3 ~/ma553/classwork/cw4 ]$ python3 -m timeit -s "from particle_simulator4 i
10 loops, best of 5: 31 msec per loop
```

Step 7