Blue Waters: python profiling webinar

bwpy module

module load bwpy # for Blue Waters, or use the default python3 on your local system

Function profiling

```
The python cProfile builtin function profiler works with the bwpy module.
To use it with your script, it can be as simple as:

$ aprun -n 1 python -m cProfile <myscript.py>

Optionally add the -o <bw.profile> and your bw.profile may be analyzed
with any of the
python profile viewers you may install in a virtualenv or via the python
pstats module:

$ python
>>> import pstats
>>> p = pstats.Stats('bw.profile')
>>> p.sort_stats('cumulative').print_stats(5)
```

function profiling is built-in

Most python installations will have cProfile included. Its usage is highly recommended before you transition to "production" python code.

A new ssh will soon be deployed that can leverage Globus Authentication and any of your linked identities. For Xsede, we map <Xsede_portal_identity> to <local_account> for each resource provider in Xsede. Because the python code will be production and deployed at a variety of sites, we want to verify that performance is reasonable and look for any hot spots in the code. Functional profiling revealed the code to be performant with no noticeable hotspots save for the data query of the REST interface in front of the Xsede Central Database. This was expected.

```
>>> p.sort_stats('cumulative').print_stats(20)
Thu Oct  4 11:30:17 2018    myprofile

         209324 function calls (204291 primitive calls) in 0.735 seconds

   Ordered by: cumulative time
   List reduced from 2357 to 20 due to restriction <20>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    274/1    0.012    0.000    0.736    0.736 {built-in method builtins.exec}
        1    0.000    0.000    0.736    0.736 xdcdb.py:4(<module>)
        1    0.003    0.003    0.481    0.481 xdcdb.py:76(main)
        2    0.000    0.000    0.441    0.220 /Users/galen/anaconda3/lib/python3.6/site-packages/requests/api.py:61(get)
        2    0.000    0.000    0.441    0.220 /Users/galen/anaconda3/lib/python3.6/site-packages/requests/api.py:16(request)
        2    0.000    0.000    0.436    0.218 /Users/galen/anaconda3/lib/python3.6/site-packages/requests/sessions.py:441(request)
        2    0.000    0.000    0.425    0.213 /Users/galen/anaconda3/lib/python3.6/site-packages/requests/sessions.py:589(send)
        2    0.000    0.000    0.397    0.198 /Users/galen/anaconda3/lib/python3.6/site-packages/requests/adapters.py:388(send)
        2    0.000    0.000    0.395    0.198 /Users/galen/anaconda3/lib/python3.6/site-packages/urllib3/connectionpool.py:447(urlopen)
        2    0.000    0.000    0.395    0.197 /Users/galen/anaconda3/lib/python3.6/site-packages/urllib3/connectionpool.py:322(_make_request)
        1    0.009    0.009    0.344    0.344 xdcdb.py:43(gen_mapfile)
      409    0.001    0.000    0.328    0.001 /Users/galen/anaconda3/lib/python3.6/socket.py:572(readinto)
      409    0.001    0.000    0.327    0.001 /Users/galen/anaconda3/lib/python3.6/site-packages/urllib3/contrib/pyopenssl.py:278(recv_into)
      409    0.002    0.000    0.326    0.001 /Users/galen/anaconda3/lib/python3.6/site-packages/OpenSSL/SSL.py:1787(recv_into)
      409    0.323    0.001    0.323    0.001 {built-in method _openssl.SSL_read}
      175    0.000    0.000    0.307    0.002 {method 'readline' of '_io.BufferedReader' objects}
        2    0.000    0.000    0.306    0.153 /Users/galen/anaconda3/lib/python3.6/http/client.py:1287(getresponse)
        2    0.000    0.000    0.306    0.153 /Users/galen/anaconda3/lib/python3.6/http/client.py:290(begin)
        2    0.000    0.000    0.305    0.152 /Users/galen/anaconda3/lib/python3.6/http/client.py:257(_read_status)
    284/5    0.002    0.000    0.257    0.051 <frozen importlib._bootstrap>:966(_find_and_load)


<pstats.Stats object at 0x1016ff828>
>>>
```

Line level profiling

```
You may setup the line_profiler in a virtualenv as shown and use it to
profile functions you decorate with @profile .

arnoldg@h2ologin2:~> module load bwpy
arnoldg@h2ologin2:~> virtualenv line_profiler
Using base prefix '/mnt/bwpy/single/usr'
New python executable in /mnt/a/u/staff/arnoldg/line_profiler/bin/python3.5
Also creating executable in /mnt/a/u/staff/arnoldg/line_profiler/bin/python
Installing setuptools, pip, wheel...done.
arnoldg@h2ologin2:~> source line_profiler/bin/activate
(line_profiler) arnoldg@h2ologin2:~> pip install line_profiler
Collecting line_profiler
  Downloading https://files.pythonhosted.org/packages/14/fc
/ecf4e238bb601ff829068e5a72cd1bd67b0ee0ae379db172eb6a0779c6b6
/line_profiler-2.1.2.tar.gz (83kB)
    100% |################################| 92kB 3.9MB/s
...
(line_profiler) arnoldg@h2ologin2:~> kernprof
Usage: kernprof [-s setupfile] [-o output_file_path] scriptfile [arg] ...

# switching into a batch job here...

(line_profiler) arnoldg@h2ologin2:~> grep --after-context=5 @profile
ep_task_errors_bw.py
@profile
def my_endpoint_manager_task_list(tclient, endpoint):
    """
    Get tasks from an endpoint, then look through them for error events.
    Also mark as SRC, DEST, or SRC_DEST as the case may be.
    """

(line_profiler) arnoldg@h2ologin2:~> aprun -n 1 kernprof -l
ep_task_errors_bw.py
...

(line_profiler) arnoldg@h2ologin2:~> python -m line_profiler
ep_task_errors_bw.py.lprof
```

This is a production code we run on the backend to monitor our Globus Online endpoints for some transfer error conditions. The code was developed for the ncsa#Nearline endpoint. When it was ported to the ncsa#BlueWaters endpoint, cycle time for the python script became much longer (because that endpoint is typically involved in more transfers). We used both functional and line-level profiling to diagnose the issue. Line-level profiling pointed out a single line consuming most of the time. Some research into the Globus API for the call on that line led us to a better query formulation which greatly minimized the data returned from GO. +1 for python profiling.

```
+ Timer unit: 1e-06 s
+
+ Total time: 451.817 s
+ File: ep_task_errors_bw.py
+ Function: my_endpoint_manager_task_list at line 98
+
+ Line #      Hits         Time  Per Hit   % Time  Line Contents
+ ==============================================================
+     98                                           @profile
+     99                                           def
 my_endpoint_manager_task_list(tclient, endpoint):
+    100                                               """
+    101                                               Get tasks from an endpoint, then
 look through them for error events.
+    102                                               Also mark as SRC, DEST, or SRC_DEST
 as the case may be.
+    103                                               """
+    104         2        181.0     90.5      0.0      source_total_files = 0
+    105         2          6.0      3.0      0.0      dest_total_files = 0
+    106         2         17.0      8.5      0.0      source_total_bps = 0
+    107         2          6.0      3.0      0.0      dest_total_bps = 0
+    108         2          4.0      2.0      0.0      source_total_tasks = 0
+    109         2          4.0      2.0      0.0      dest_total_tasks = 0
+    110
+    111    179511  434769564.0   2422.0     96.2      for task in
 tclient.endpoint_manager_task_list(filter_endpoint=endpoint, num_results=None):
+    112    179509     787031.0      4.4      0.2          if task["status"] == "ACTIVE":
+    113       191       1040.0      5.4      0.0              if
```

```
(line_profiler) arnoldg@h2ologin2:~> python -m line_profiler ep_task_errors_bw.py.lprof

Total time: 7.17466 s
File: ep_task_errors_bw.py
Function: my_endpoint_manager_task_list at line 103

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
   103                                           @profile
   104                                           def my_endpoint_manager_task_list(tclient, endpoint):
   105                                               """
   106                                               Get tasks from an endpoint, then look through them for error events.
   107                                               Also mark as SRC, DEST, or SRC_DEST as the case may be.
   108                                               """
   109         3         20.0      6.7      0.0      source_total_files = 0
   110         3         13.0      4.3      0.0      dest_total_files = 0
   111         3          9.0      3.0      0.0      source_total_bps = 0
   112         3          8.0      2.7      0.0      dest_total_bps = 0
   113         3          8.0      2.7      0.0      source_total_tasks = 0
   114         3          8.0      2.7      0.0      dest_total_tasks = 0
   115
   116         3         17.0      5.7      0.0      for task in tclient.endpoint_manager_task_list(filter_endpoint=endpoint,
   117         3          9.0      3.0      0.0                                                     filter_status="ACTIVE",
   118        44    1631294.0  37074.9     22.7                                                     num_results=None):
   119        41        431.0     10.5      0.0          if task["destination_endpoint_id"] == endpoint:
```

References:

https://bluewaters.ncsa.illinois.edu/Python-profiling

https://github.com/ncsa/endpoint_task_errors

https://docs.globus.org/api/