Auro Tripathy

31 Followers About Follow

Use puDB to Debug Python Multiprocessing Code

Auro Tripathy Feb 24, 2019 · 2 min read

puDB is a lightweight, keyboard-friendly visual debugger for Python without the encumbrances of a full-blown Integrated Development Environment (IDE). pdb is its more well-known no-frills cousin. I easily get lost in pdb's line-oriented view of the code. My go-to Python debugger is puDB.

Recently, I wanted to debug a multiprocessing Reinforcement Learning algorithm called Asynchronous Advantage Actor-Critic (A3C). Because it asynchronously trains agents, it can launch tens of processes, and each process (agent) contributes to the globally shared model.

How would you debug such a massively parallel algorithm? That's the topic I'll cover here with a didactic example to drive home the steps.

The code below defines a `worker` function that is spawned off into two processes.

```
"""
Using pudb in debugging multiprocessing
https://documen.tician.de/pudb/starting.html#remote-debugging
"""
```

```
import multiprocessing as mp
import time
from pudb.remote import set_trace

def worker(worker_id):
    """ Simple worker process"""
    i = 0
    while i < 10:
        if worker_id == 1:  # debug process with id 1
            set_trace(term_size=(80, 24))
        time.sleep(1)  # represents some work
        print('In Process {}, i:{}'.format(worker_id, i))
        i = i + 1

if __name__ == '__main__':
    processes = []
    for p_id in range(2): # 2 worker processes
        p = mp.Process(target=worker, args=(p_id,))
        p.start()
        processes.append(p)

    for p in processes:
        p.join()
```

If you need to debug the `worker`, then simply insert the line below along with your other module imports. Notice the use of `pudb.remote` indicating the use of a remote terminal. For single process debugging, use

```
from pudb.remote import set_trace
```

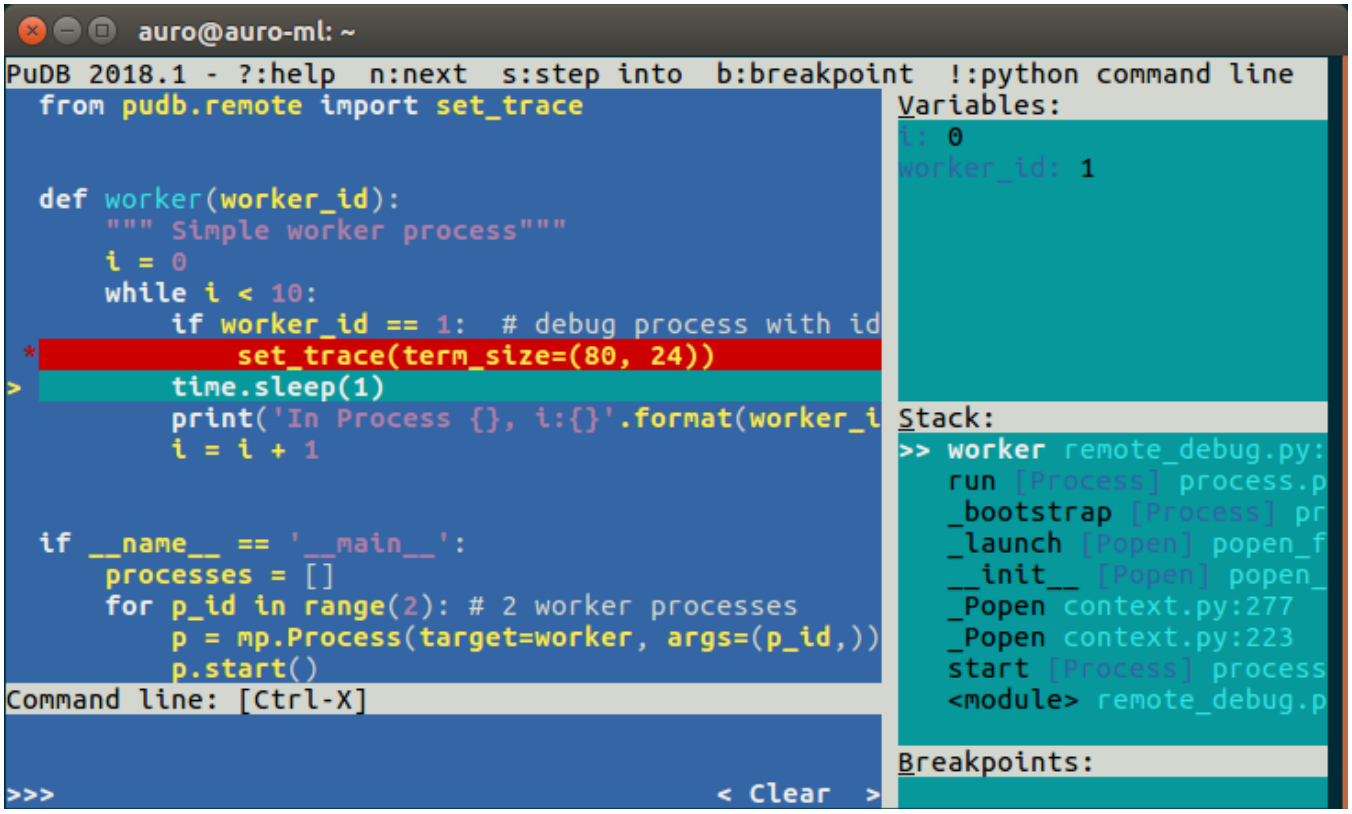
Now insert `set_trace(term_size=(80, 24))` where you would like to break. If you want to stop in all processes, then that's all you need to do. Recall that you may have spawned off tens of processes and you want to break in just one of them. You can easily do that with some conditional logic show below.

```
if worker_id == 1:  # debug process with id 1
     set_trace(term_size=(80, 24))
```

Once the condition is met in the process you want to break in, puDB will look for a free port and wait for a telnet connection:

```
pudb:6899: Please telnet into 127.0.0.1 6899.
pudb:6899: Waiting for client...
```

Go to another terminal, type, `telnet 127.0.0.1 6899` and expect to see the console taken over by the visual puDB debugger.



Note, you've stopped in the process you specified (in this example, process with id 1). The rest of the processes will continue to run unfettered.

That's it! Hoping you see value in visual debugging with puDB.

Debugging