

pynvme: test NVMe devices in Python

<https://github.com/cranechu/pynvme>

pynvme

The `pynvme` is a python extension module.

Users can operate NVMe SSD intuitively in Python scripts. It is designed for NVMe SSD `testing` with performance considered.

Integrated with third-party tools, vscode and pytest, pynvme provides a convenient and professional solution to test NVMe devices.



First Example

```
cranechu@localhost: ~/pynvme/spdk/examples/nvme/identify
Host Read Commands: 203987588
Host Write Commands: 25207644
Controller Busy Time: 1542 minutes
Power Cycles: 45
Power On Hours: 42 hours
Unsafe Shutdowns: 19
Unrecoverable Media Errors: 28
Lifetime Error Log Entries: 1563
Warning Temperature Time: 0 minutes
Critical Temperature Time: 0 minutes
Temperature Sensor 1: 318 Kelvin (45 Celsius)

Number of Queues
=====
Number of I/O Submission Queues: 31
Number of I/O Completion Queues: 31

Namespace ID:1
Deallocate: Supported
Deallocated/Unwritten Error: Not Supported
Deallocated Read Value: Unknown
Deallocate in Write Zeroes: Not Supported
Deallocated Guard Field: 0xFFFF
Flush: Supported
Reservation: Not Supported
Namespace Sharing Capabilities: Private
Size (in LBAs): 500118192 (476M)
Capacity (in LBAs): 500118192 (476M)
Utilization (in LBAs): 500118192 (476M)
EUI64: 00080d04001b4316
Thin Provisioning: Not Supported
Per-NS Atomic Units: No
NGUID/EUI64 Never Reused: No
Number of LBA Formats: 2
Current LBA Format: LBA Format #01
LBA Format #00: Data Size: 4096 Metadata Size: 0
LBA Format #01: Data Size: 512 Metadata Size: 0

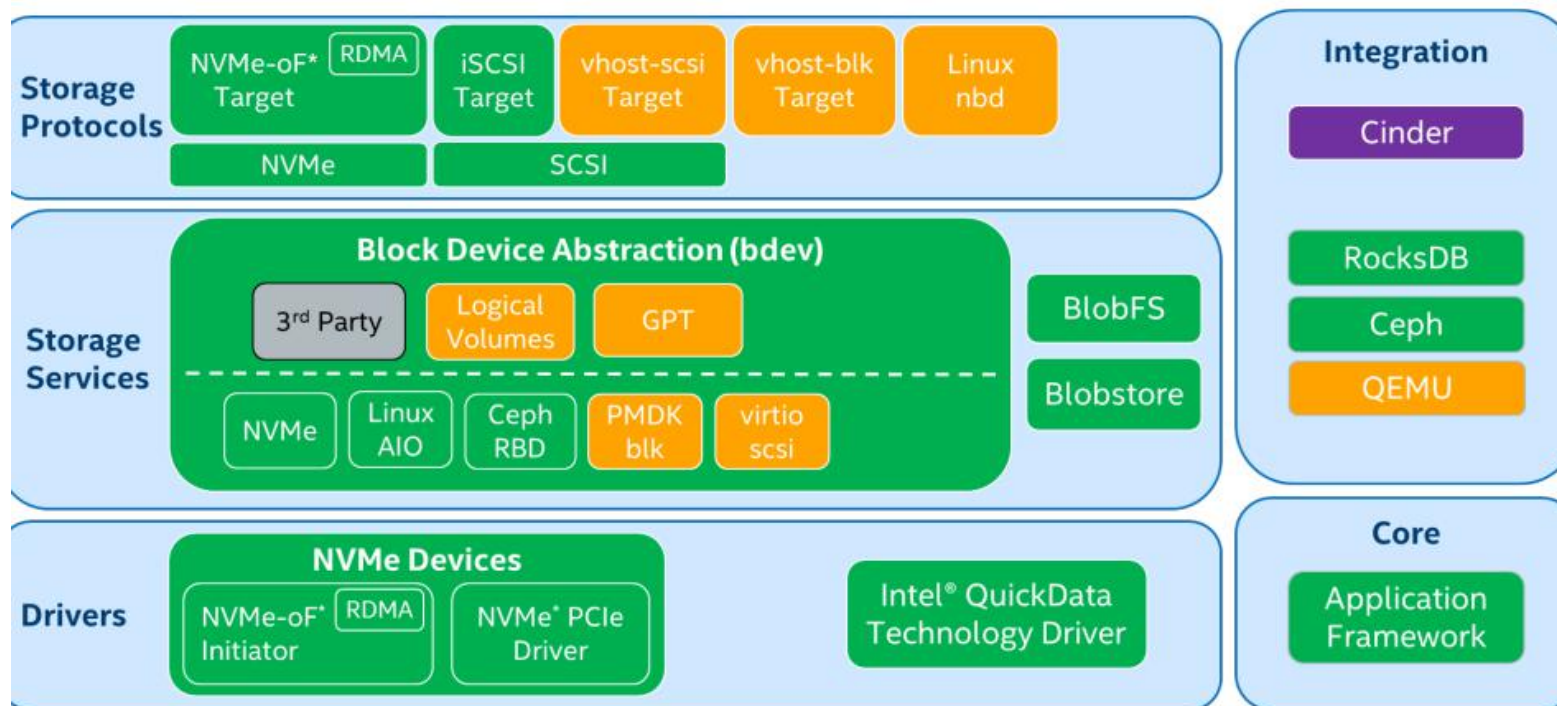
~/pynvme/~/examples/nvme/identify>

cranechu@localhost: ~/pynvme
~/pynvme> sudo python3
Python 3.7.2 (default, Jan 16 2019, 19:49:22)
[GCC 8.2.1 20181215 (Red Hat 8.2.1-6)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import nvme
Starting SPDK v19.01-pre / DDPK 18.08.0 initialization...
[ DDPK EAL parameters: pynvme_driver -c 0x2 -m 5892 --base-virtaddr=0x20
00000000 --file-prefix=spdk0 --proc-type=auto ]
EAL: Detected 4 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Auto-detected process type: PRIMARY
EAL: Multi-process socket /var/run/dpdk/spdk0/mp_socket
EAL: Probing VFIO support...
EAL: no supported IOMMU extensions found!
EAL: VFIO support could not be initialized
>>> nvme0 = nvme.Controller(b'01:00.0')
EAL: PCI device 0000:01:00.0 on NUMA socket 0
EAL: probe driver: 1179:113 spdk_nvme
nvme.pcie.c: 992:nvme.pcie.construct: *INFO*: max_completions_cap
32 num_trackers = 96
driver.c: 449:attach_cb: *INFO*: attached device 0000:01:00.0: KBG30ZMS2
, 1 namespaces, pid 3001 ADDA0102
>>> nvme0n1 = nvme.Namespace(nvme0)
driver.c: 76:memzone_reserve_shared_memory: *INFO*: create token table,
size: 2000472768
>>> eui64 = nvme0n1.id_data(127, 120)
>>> eui64
1604155579155941376
>>> eui64.to_bytes(8, byte_order='little')
... ).hex()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: to_bytes() missing required argument 'byteorder' (pos 2)
>>> eui64.to_bytes(8, byteorder='little').hex()
'00080d04001b4316'
>>> # the EUI64 got by pynvme is just the same as the spdk example. Good
...
>>> exit()
```

SPDK

pynvme

Pynvme is based on SPDK

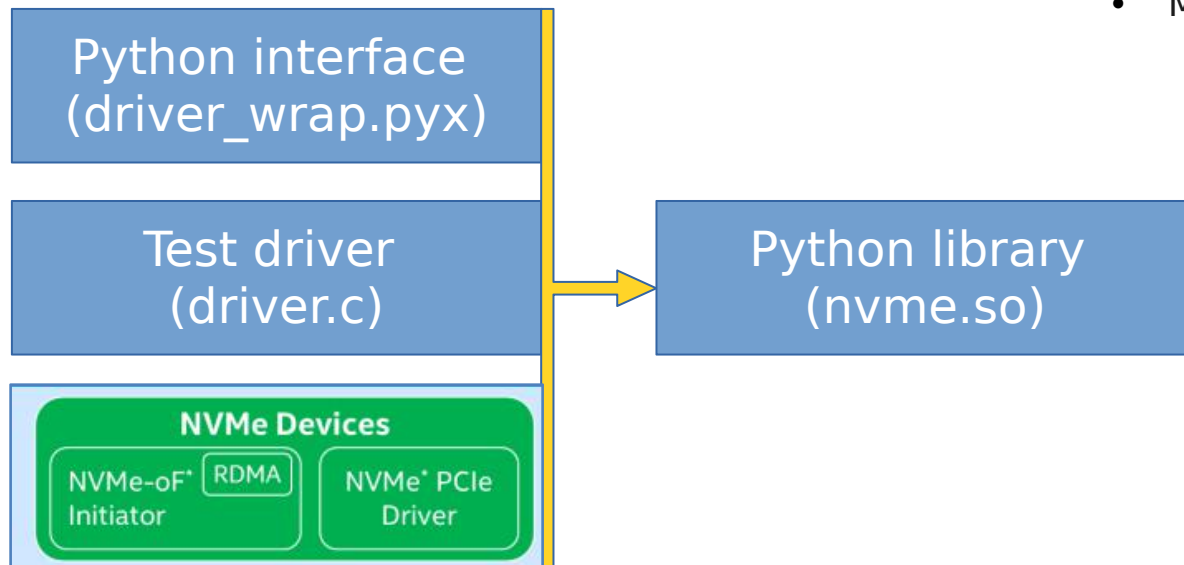


Pynvme Architecture

Build python library with

Cython:

- setup.py
- driver.c
- driver.h
- cdriver.pxd
- driver_wrap.pyx
- Makefile

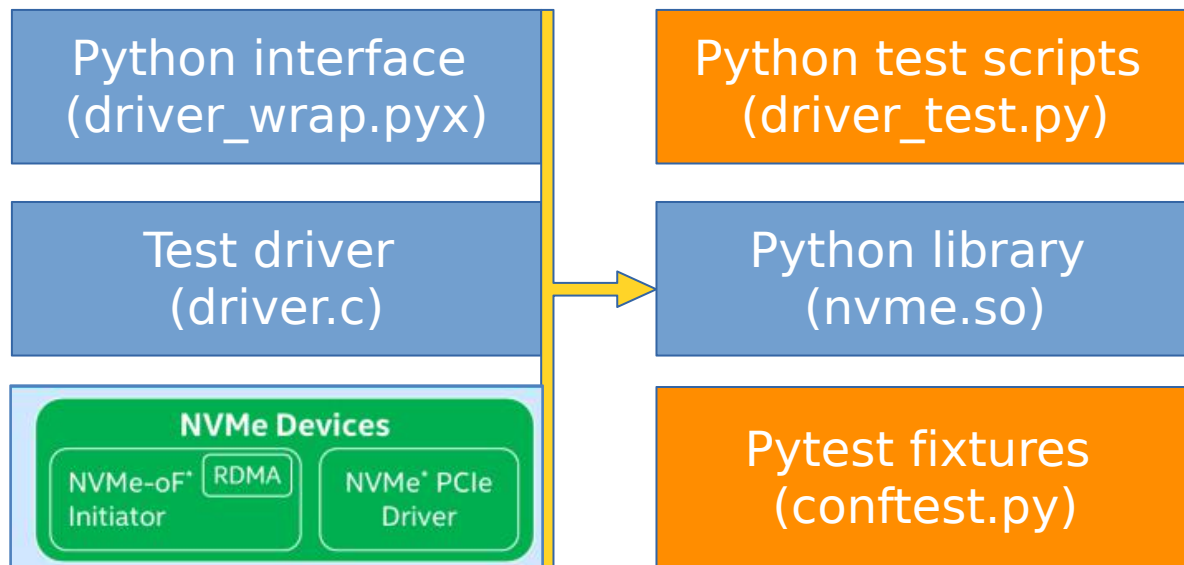


Pynvme Architecture

Organize test cases in

pytest:

- mvme.so
- pytest.ini
- conftest.py
- driver_test.py



Why Python?



✓ Many beautiful mature libraries

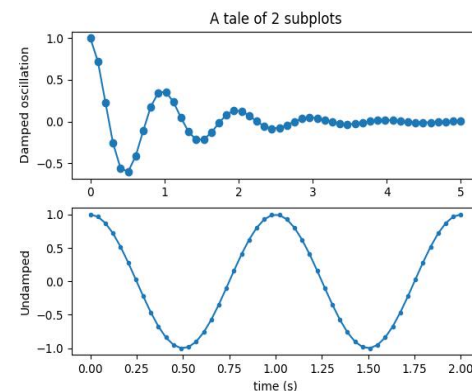
- pytest
- logging
- multiprocessing
- numpy
- matplotlib
- pydoc
- os, io, time, pytemperature, statistics, yaml, json, struct, ...

✓ Friendly IDEs for developing, debugging, and testing

- VSCode, emacs, Pycharm

✓ CI: develop firmware softly













pipeline passed



Pynvme CI Status

Crane Chu > pynvme > Pipelines

All 160 Pending 1 Running 2 Finished 157 Branches Tags Run Pipeline Clear Runner Caches CI Lint

Status	Pipeline	Commit	Stages		
🟡 pending	#50853930 by  latest	🔑 master → 8b57310a  CI: without-isal to avoid writin...	⏸		✖
🟢 running	#50846622 by  latest	🔑 master → 8b57310a  CI: without-isal to avoid writin...	⏸	🕒 00:00:01 📅 3 hours ago	✖
🟢 running	#50842389 by  latest	🔑 master → 8b57310a  CI: without-isal to avoid writin...	🌙	🕒 00:00:01 📅 3 hours ago	✖
🟢 passed	#50835305 by  latest	🔑 master → 8b57310a  CI: without-isal to avoid writin...	✅	🕒 00:22:26 📅 16 minutes ago	
🟢 passed	#50828571 by  latest	🔑 master → 8b57310a  CI: without-isal to avoid writin...	✅	📅 39 minutes ago	
🟢 passed	#50822509 by  latest	🔑 master → 8b57310a  CI: without-isal to avoid writin...	✅	🕒 00:23:02 📅 1 hour ago	

Config and Compile

1. `git clone https://github.com/cranechu/pynvme`
 2. `cd pynvme`
 3. `./install.sh`
 4. now, ready to run test scripts with `pytest`
- Fedora29 is recommended.
 - find test script examples in `driver_test.py`.
 - **find more** => <https://github.com/cranechu/pynvme>

Pytest Execution



- `'''The pytest framework makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries.'''`
- `'''pytest fixtures offer dramatic improvements over the classic xUnit style of setup/teardown functions'''`
- use “make test” to start pytest sessions
 - make test
 - make test TESTS=scripts
 - make test TESTS=scripts/demo_test.py
 - make test TESTS=scripts/utility_test.py::test_download_firmware
- find test logs in test.log

Test Script Files

import pytest

import nvme

test functions

test file name

```
emacs
34 # -*- coding: utf-8 -*-
35
36
37 import os
38 import time
39 import pytest
40 import logging
41 import warnings
42
43 import nvme as d
44
45 def test_create_device(nvme0, nvme0n1):
46     assert nvme0 is not None
47
48 def test_create_device_invalid():
49     with pytest.raises(d.NvmeEnumerateError):
50         nvme1 = d.Controller(b"00:00.0")
51
55k 35: 0 UU-x-~/pynvme/driver_test.py 3% -master Python
```

pytest collects test files and cases before execution

An Example

pytest cases
are started
with test_

create qpair and
buffer for write
commands

write data,
and then read
in the callback

```
emacs
248 def test_write_identify_and_verify_with_callback(nvme0, nvme0n1):
249     id_buf = d.Buffer(4096)
250     nvme0.identify(id_buf).waitdone()
251
252     q = d.Qpair(nvme0, 20)
253     n = nvme0n1
254     read_buf = d.Buffer(4096, "read buffer")
255
256     def read_cb(cdw0, status):
257         assert id_buf[:40] == read_buf[:40]
258
259     def write_cb(cdw0, status):
260         n.read(q, read_buf, 5, 8, cb=read_cb)
261
262     n.write(q, id_buf, 5, 8, cb=write_cb).waitdone(2)
263
264     id_buf[0] += 1
265     n.write(q, id_buf, 5, 8, cb=write_cb).waitdone(2)
266     id_buf[9] = (id_buf[9] >> 1)
267     n.write(q, id_buf, 5, 8, cb=write_cb).waitdone(2)
268
269 55k 249: 0 UU- ~/pynvme/driver_test.py 15% -master Python [test
In: test_write_identify_and_verify_with_callback()
```

pytest fixtures

fill buffer with
identify data

callback functions
are called when
cmds are completed

the status of the
callback function
also includes the
Phase Tag bit

Fixtures of Pynvme

- create/delete test objects. in conftest.py:
 - nvme0
 - nvme0n1
 - pcie
 - ...
- parametrize of tests
 - @pytest.mark.parametrize("qcount", [1, 2, 4, 8, 15])
 - @pytest.mark.parametrize("repeat", range(10))
- test control
 - @pytest.mark.skip("nvme over tcp")
- doc: <https://docs.pytest.org/en/latest/fixture.html>

IOWorker

```
emacs
1066 @pytest.mark.parametrize("qcount", [1, 2, 4, 8, 15])
1067 def test_ioworker_iops_multiple_queue(nvme0n1, qcount):
1068     l = []
1069     io_total = 0
1070     for i in range(qcount):
1071         a = nvme0n1.ioworker(io_size=8, lba_align=8,
1072                             region_start=0, region_end=256*1024*8, # 1GB space
1073                             lba_random=False, qdepth=16,
1074                             read_percentage=100, time=10).start()
1075         l.append(a)
1076
1077     for a in l:
1078         r = a.close()
1079         io_total += (r.io_count_read+r.io_count_write)
1080
1081     logging.info("Q %d IOPS: %dK" % (qcount, io_total/10000))
1082
55k 1068: 0 UU- ~/pynvme/driver_test.py 60% -master Python [test_ioworker_iop
In: test_ioworker_iops_multiple_queue()
```

define IO patterns
in ioworker's
parameter list

send IO in a
separated process

wait ioworkers till
finish, and collect
result data

Visual Studio Code



- VSCode is [the most popular IDE](#).
 - root user is not recommended by vscode, so users need to run sudo without a password: *sudo visudo*
- Pynvme also provides an extension to monitor device status and cmdlog in every qpair. To install the extension:
 - *code --install-extension pynvme-console-1.0.0.vsix*
- Add DUT pci address to .vscode/settings.json
 - get the BDF address with *lspci*
- *make setup; code .* # launch the vscode

test items

qpairs

```
1 import time
2 import pytest
3 import logging
4 import nvme as d
5
6
7 def test_trim_basic(nvme0, nvme0n1, verify):
8     GB = 1024*1024*1024
9     all_zeor_databuf = d.Buffer(512)
10    trimbuf = d.Buffer(4096)
11    q = d.Qpair(nvme0, 32)
12
13    # DUT info
14    logging.info("model number: %s" % nvme0.id_data(63, 24, str))
15    logging.info("firmware revision: %s" % nvme0.id_data(71, 64, str))
16
17    # write
18    logging.info("write data in 10G ~ 20G")
19    io_size = 128*1024//512
20    start_lba = 10*GB//512
21    lba_count = 10*GB//512
22    nvme0n1.ioworker(io_size = io_size,
23                    lba_align = io_size,
24                    lba_random = False,
25                    read_percentage = 0,
26                    lba_start = start_lba,
27                    io_count = lba_count//io_size,
28                    qdepth = 128).start().close()
29
30    # verify data after write, data should be modified
31    with pytest.warns(UserWarning, match="ERROR status: 02/85"):
32        nvme0n1.compare(q, all_zeor_databuf, start_lba).waitdone()
33
34    # get the empty trim time
35    trimbuf.set_dsm_range(0, 0, 0)
36    trim_cmd = nvme0n1.dsm(q, trimbuf, 1).waitdone() # first call is longer, due to
37    start_time = time.time()
38    trim_cmd = nvme0n1.dsm(q, trimbuf, 1).waitdone()
39    empty_trim_time = time.time()-start_time
40
41    # the trim time on device-side only
42    logging.info("trim the 10G data from LBA 0x%x" % start_lba)
43    trimbuf.set_dsm_range(0, start_lba, lba_count)
44    start_time = time.time()
45    trim_cmd = nvme0n1.dsm(q, trimbuf, 1).waitdone()
46    trim_time = time.time()-start_time-empty_trim_time
47    logging.info("trim bandwidth: %0.2fGB/s" % (10/trim_time))
48
49    # verify after trim
50    nvme0n1.compare(q, all_zeor_databuf, start_lba).waitdone()
```

test scripts

test log

```
===== test session starts =====
platform linux -- Python 3.7.3, pytest-4.4.0, py-1.8.0, pluggy-0.9.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /home/cranechu/pynvme, inifile: pytest.ini
plugins: cov-2.6.1
collecting ... collected 1 item

scripts/test_trim_basic.py::test_trim_basic
----- live log setup -----
[2019-04-29 12:37:06.237] INFO pciaddr(19): running tests on DUT 01:00.0
----- live log call -----
[2019-04-29 12:37:10.777] INFO test_trim_basic(14): model number: SM961 NVMe SAMSUNG 1024GB
[2019-04-29 12:37:10.778] INFO test_trim_basic(15): firmware revision: CXA75D0Q
[2019-04-29 12:37:10.779] INFO test_trim_basic(18): write data in 10G ~ 20G
[2019-04-29 12:37:16.761] INFO test_trim_basic(42): trim the 10G data from LBA 0x1400000
[2019-04-29 12:37:16.763] INFO test_trim_basic(47): trim bandwidth: 16219.27GB/s
PASSED [100%]
----- live log teardown -----
[2019-04-29 12:37:16.766] INFO script(33): test duration: 5.992 sec

----- generated xml file: /tmp/tmp-5551D69216jfy7tm.xml -----
===== 1 passed in 10.65 seconds =====
===== test session starts =====
platform linux -- Python 3.7.3, pytest-4.4.0, py-1.8.0, pluggy-0.9.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /home/cranechu/pynvme, inifile: pytest.ini
plugins: cov-2.6.1
collecting ... collected 1 item

scripts/test_trim_basic.py::test_trim_basic
----- live log setup -----
[2019-04-29 12:37:45.654] INFO pciaddr(19): running tests on DUT 01:00.0
----- live log call -----
```


demo_test.py x

```
1 import time
2 import pytest
3 import logging
4 import nvme as d
5 from pytemperature import k2c
6
7 Run Test | Debug Test
8 def test_ioworker_with_temperature(nvme0, nvme0n1):
9     smart_log = d.Buffer(512, "smart log")
10     with nvme0n1.ioworker(io_size=8, lba_align=16, lba_random=True,
11                           qdepth=16, read_percentage=0, time=30):
12         for i in range(40):
13             nvme0.getlogpage(0x02, smart_log, 512).waitdone()
14             ktemp = smart_log.data(2, 1)
15             ctemp = k2c(ktemp)
16             logging.info("temperature: %.0.2f degreeC" % ctemp)
17             time.sleep(1)
```

test scripts

CMDLOG Q0 x

```
1 1556546728.807225: [cmd: Get Log Page]
2 0x005f0002, 0xffffffff, 0x00000000, 0x00000000
3 0x00000000, 0x00000000, 0x7439a000, 0x00000001
4 0x00000000, 0x00000000, 0x007f0002, 0x00000000
5 0x00000000, 0x00000000, 0x00000000, 0x00000000
6 1556546728.809293: [cpl: SUCCESS]
7 0x00000000, 0x00000000, 0x00000023, 0x0001005f
8
9 1556546727.804234: [cmd: Get Log Page]
10 0x005f0002, 0xffffffff, 0x00000000, 0x00000000
11 0x00000000, 0x00000000, 0x7439a000, 0x00000001
12 0x00000000, 0x00000000, 0x007f0002, 0x00000000
13 0x00000000, 0x00000000, 0x00000000, 0x00000000
14 1556546727.806340: [cpl: SUCCESS]
15 0x00000000, 0x00000000, 0x00000022, 0x0001005f
16
17 1556546726.801221: [cmd: Get Log Page]
18 0x005f0002, 0xffffffff, 0x00000000, 0x00000000
```

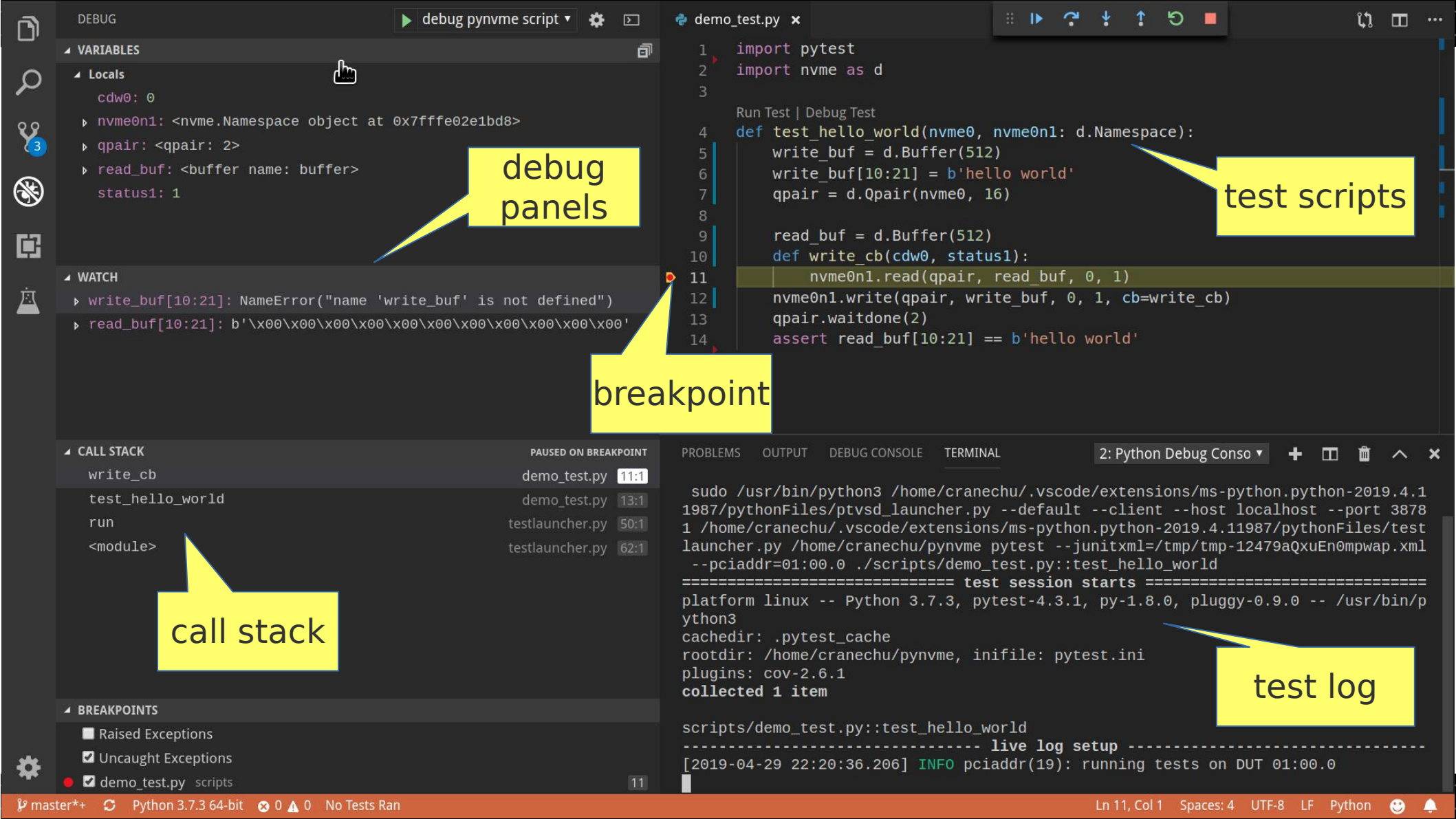
test log

cmdlog

PROBLEMS OUTPUT ...

Python Test Log

```
[2019-04-29 22:05:11.721] INFO test_ioworker_with_temperature(15):
temperature: 43.85 degreeC
[2019-04-29 22:05:12.728] INFO test_ioworker_with_temperature(15):
temperature: 44.85 degreeC
[2019-04-29 22:05:13.733] INFO test_ioworker_with_temperature(15):
temperature: 44.85 degreeC
[2019-04-29 22:05:14.739] INFO test_ioworker_with_temperature(15):
temperature: 44.85 degreeC
[2019-04-29 22:05:15.756] INFO test_ioworker_with_temperature(15):
temperature: 44.85 degreeC
[2019-04-29 22:05:16.760] INFO test_ioworker_with_temperature(15):
temperature: 45.85 degreeC
[2019-04-29 22:05:17.764] INFO test_ioworker_with_temperature(15):
temperature: 45.85 degreeC
[2019-04-29 22:05:18.775] INFO test_ioworker_with_temperature(15):
temperature: 45.85 degreeC
[2019-04-29 22:05:19.779] INFO test_ioworker_with_temperature(15):
temperature: 45.85 degreeC
[2019-04-29 22:05:20.782] INFO test_ioworker_with_temperature(15):
temperature: 45.85 degreeC
[2019-04-29 22:05:21.786] INFO test_ioworker_with_temperature(15):
temperature: 46.85 degreeC
[2019-04-29 22:05:22.789] INFO test_ioworker_with_temperature(15):
temperature: 46.85 degreeC
[2019-04-29 22:05:23.792] INFO test_ioworker_with_temperature(15):
temperature: 46.85 degreeC
[2019-04-29 22:05:24.795] INFO test_ioworker_with_temperature(15):
temperature: 46.85 degreeC
[2019-04-29 22:05:25.800] INFO test_ioworker_with_temperature(15):
temperature: 46.85 degreeC
[2019-04-29 22:05:26.803] INFO test_ioworker_with_temperature(15):
temperature: 46.85 degreeC
[2019-04-29 22:05:27.806] INFO test_ioworker_with_temperature(15):
temperature: 47.85 degreeC
[2019-04-29 22:05:28.809] INFO test_ioworker_with_temperature(15):
temperature: 47.85 degreeC
```



debug
panels

test scripts

breakpoint

call stack

test log

pynvme is OPEN

cranechu / pynvme

Unwatch 4 Unstar 10 Fork 4

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

test NVMe devices in Python

nvme ssd driver spdk test python linux storage Manage topics

168 commits 1 branch 5 releases 1 contributor BSD-3-Clause

Branch: master New pull request

Create new file Upload files Find File Clone or download

cranechu driver: ioworker timeout 30s	Latest commit a7c69ef 11 hours ago
.vscode	update spdk for outstanding count 4 days ago
doc	doc: update readme a month ago
scripts	vscode: debug pytest script as sudo 27 days ago
snippets/python-mode	build: pre-release pypi 3 months ago
spdk @ e2305df	add pynvme branch in dpdk 18 hours ago
.gitignore	doc: upload a ppt a month ago
.gitlab-ci.yml	test: no memory leakage, no reboot after test 2 days ago
.gitmodules	dpdk: 17.11 a day ago

<https://github.com/cranechu/pynvme>