

# **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

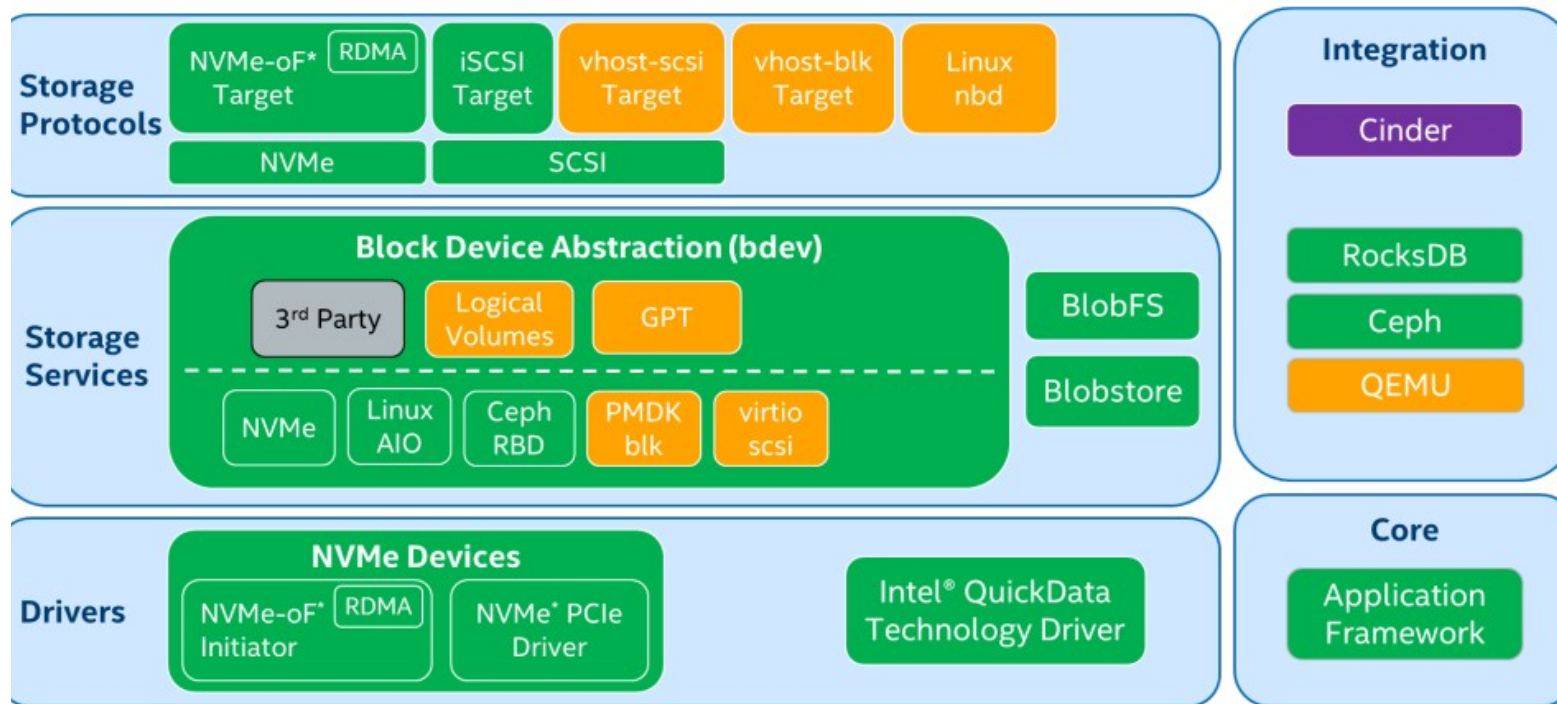
~/pynvme/~/examples/nvme/identify>

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.qpair_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

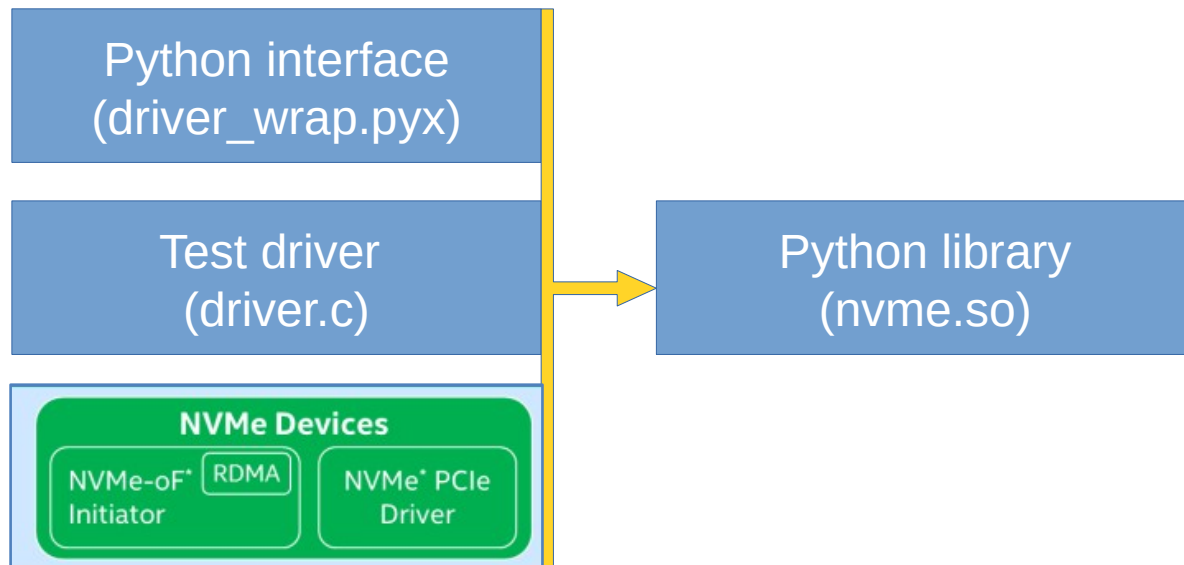
# Pynvme is based on SPDK



# Pynvme Architecture

Build python library with **Cython**:

- setup.py
- driver.c
- driver.h
- cdriver.pxd
- driver\_wrap.pyx
- Makefile

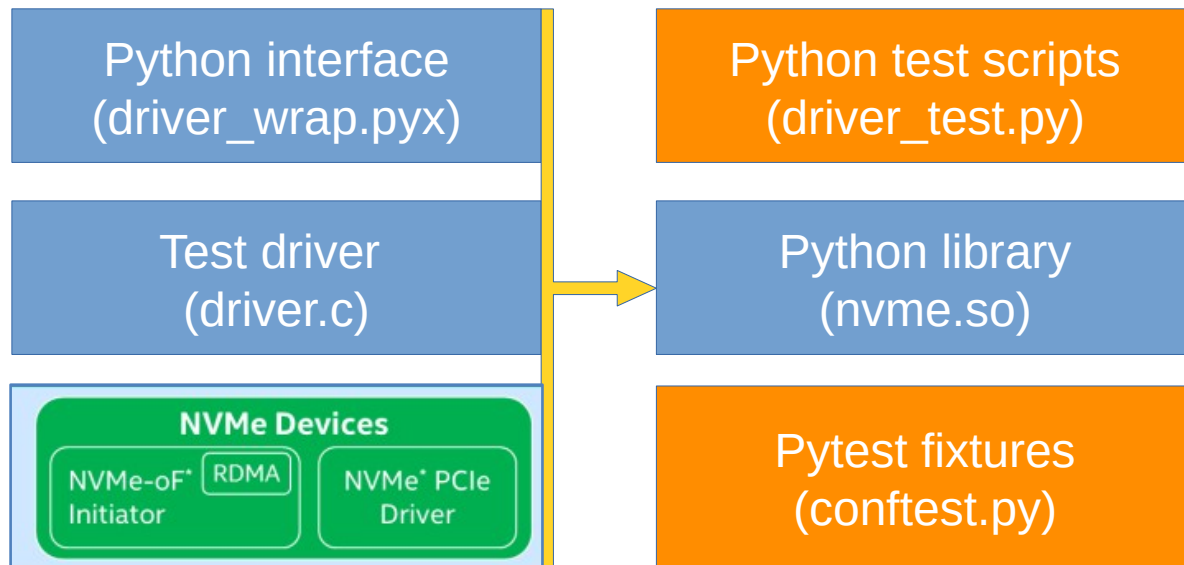


# Pynvme Architecture

Organize test cases in

**pytest:**

- mvme.so
- pytest.ini
- conftest.py
- driver\_test.py



# Why Python?

✓ Many beautiful mature libraries

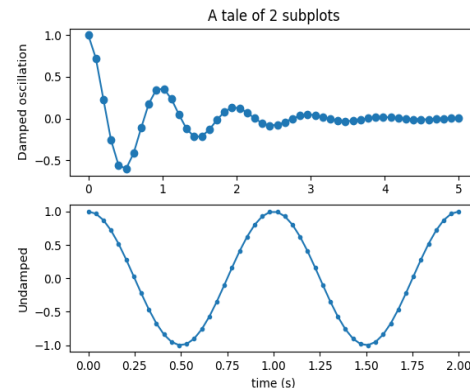
- pytest
- logging
- multiprocessing
- numpy
- matplotlib
- pydoc
- os, io, time, pytemperature, statistics, yaml, json, struct, ...

✓ 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		
<span>🟡 pending</span>	#50853930 by  <span>latest</span>	🔑 master → 8b57310a  CI: without-isal to avoid writin...	<span>⏸</span>		<span>✖</span>
<span>🟢 running</span>	#50846622 by  <span>latest</span>	🔑 master → 8b57310a  CI: without-isal to avoid writin...	<span>⏸</span>	🕒 00:00:01 📅 3 hours ago	<span>✖</span>
<span>🟢 running</span>	#50842389 by  <span>latest</span>	🔑 master → 8b57310a  CI: without-isal to avoid writin...	<span>🌙</span>	🕒 00:00:01 📅 3 hours ago	<span>✖</span>
<span>🟢 passed</span>	#50835305 by  <span>latest</span>	🔑 master → 8b57310a  CI: without-isal to avoid writin...	<span>✅</span>	🕒 00:22:26 📅 16 minutes ago	
<span>🟢 passed</span>	#50828571 by  <span>latest</span>	🔑 master → 8b57310a  CI: without-isal to avoid writin...	<span>✅</span>	📅 39 minutes ago	
<span>🟢 passed</span>	#50822509 by  <span>latest</span>	🔑 master → 8b57310a  CI: without-isal to avoid writin...	<span>✅</span>	🕒 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"
- `pytest test_mod.py` # run tests in a module
- `pytest testing/` # run tests in a directory
- `pytest test_mod.py::test_func` # run a specific test case
- `pytest test_mod.py -s` # run tests without log capturing
- <https://media.readthedocs.org/pdf/pytest/latest/pytest.pdf>

# 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

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

```
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
55k 249: 0 UU- ~/pynvme/driver_test.py 15% -master Python [test
In: test_write_identify_and_verify_with_callback()
```

# 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

# 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>

# 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-0.1.2.vsix*
- Add DUT pci address to .vscode/settings.json
  - get the BDF address with *lspci*
- *make setup; code .* # launch the vscode

TEST

PYTHON

test\_read\_after\_reset

test\_cmd\_cb\_features

test\_buffer\_token\_single\_process

test\_buffer\_token\_multi\_processes

test\_buffer\_token\_single\_small\_pr...

test\_buffer\_token\_single\_large\_pr...

test\_command\_supported\_and\_e...

test\_reset\_admin\_io\_mixed

test\_reap\_without\_command

test\_reentry\_waitdone\_io\_qpair

test\_ioworker\_test\_end

test\_admin\_generic\_cmd

test\_io\_generic\_cmd

test\_ioworker\_vscod...showcase

test\_ioworker\_stress

test\_ioworker\_longtime

test\_ioworker\_longtime\_deep

test\_qpair\_different\_size

test\_format\_basic

test\_firmware\_download

test\_dst\_short

test\_different\_io\_size\_and\_count

PYNVME QPAIRS

QPair 0:

QPair 1:

QPair 2:

QPair 3:

QPair 4:

driver\_test.py

1 1555925613.406268: [cmd: Read]

2 0x00050002, 0x00000001, 0x00000000, 0x00000000

3 0x00000000, 0x00000000, 0xeb600000, 0x00000000

4 0x00000000, 0x00000000, 0x000000ba0, 0x00000000

5 0x00000000, 0x00000000, 0x00000000, 0x00000000

6 0x00000000, 0x00000000, 0x00000000, 0x00000000

7 0x00000000, 0x00000000, 0x00000000, 0x00000000

8 0x00000000, 0x00000000, 0x00000000, 0x00000000

9 0x00000000, 0x00000000, 0x00000000, 0x00000000

10 0x00000000, 0x00000000, 0x00000000, 0x00000000

11 0x00000000, 0x00000000, 0x00000000, 0x00000000

12 0x00000007, 0x00000000, 0x00000000, 0x00000000

13 not completed ...

14

15 1555925613.395702: [cmd: Read]

16 0x00060002, 0x00000001, 0x00000000, 0x00000000

17 0x00000000, 0x00000000, 0xeb60a000, 0x00000000

18 0x00000000, 0x00000000, 0x000000b98, 0x00000000

19 0x00000000, 0x00000000, 0x00000000, 0x00000000

20 0x00000000, 0x00000000, 0x000000bb8, 0x00000000

21 0x00000000, 0x00000000, 0x00000000, 0x00000000

22 0x00000000, 0x00000000, 0x00000000, 0x00000000

23 0x00000000, 0x00000000, 0x00000000, 0x00000000

24 0x00000000, 0x00000000, 0x00000000, 0x00000000

25 0x00000000, 0x00000000, 0x000000bb8, 0x00000000

26 0x00000007, 0x00000000, 0x00000000, 0x00000000

27 not completed ...

28

29 1555925613.385694: [cmd: Read]

30 0x00070002, 0x00000001, 0x00000000, 0x00000000

31 0x00000000, 0x00000000, 0xeb60c000, 0x00000000

32 0x00000000, 0x00000000, 0x000000b90, 0x00000000

33 0x00000007, 0x00000000, 0x00000000, 0x00000000

34 not completed ...

35

36 1555925613.385691: [cmd: Read]

37 0x00000002, 0x00000001, 0x00000000, 0x00000000

38 0x00000000, 0x00000000, 0x00000000, 0x00000000

39 0x00000000, 0x00000000, 0x000000bb0, 0x00000000

40 0x00000007, 0x00000000, 0x00000000, 0x00000000

41 not completed ...

42

43 1555925613.375693: [cmd: Read]

44 0x00080002, 0x00000001, 0x00000000, 0x00000000

45 0x00000000, 0x00000000, 0xeb60e000, 0x00000000

46 0x00000000, 0x00000000, 0x000000b88, 0x00000000

47 0x00000007, 0x00000000, 0x00000000, 0x00000000

48 not completed ...

49

50 1555925613.375691: [cmd: Read]

51 0x00000002, 0x00000001, 0x00000000, 0x00000000

52 0x00000000, 0x00000000, 0x00000000, 0x00000000

53 0x00000000, 0x00000000, 0x000000ba8, 0x00000000

54 0x00000007, 0x00000000, 0x00000000, 0x00000000

driver\_test.py

1820 def test\_io\_generic\_cmd(nvme0n1, nvme0):

1821 q = d.Qpair(nvme0n1, 8)

1822 # invalid command

1823 with pytest.warns(UserWarning, match="ERROR status: nvme0n1.send\_cmd(0xff, q, nsid=1).waitdone()")

1824 nvme0n1.send\_cmd(0xff, q, nsid=1).waitdone()

1825 # invalid nsid

1826 with pytest.warns(UserWarning, match="ERROR status: nvme0n1.send\_cmd(0x0, q).waitdone()")

1827 nvme0n1.send\_cmd(0x0, q).waitdone()

1828 # flush command

1829 nvme0n1.send\_cmd(0x0, q, nsid=1).waitdone()

1830

1831

Run Test | Debug Test

1832 def test\_ioworker\_vscod...showcase(nvme0n1):

1833 with nvme0n1.ioworker(io\_size=8, lba\_align=8, lba\_r

1834 qdepth=16, read\_percentage=10

1835 iops=100, time=10), \

1836 nvme0n1.ioworker(io\_size=8, lba\_align=8, lba\_r

1837 qdepth=16, read\_percentage=10

1838 iops=1000, time=10), \

1839 nvme0n1.ioworker(io\_size=8, lba\_align=8, lba\_r

1840 qdepth=16, read\_percentage=10

1841 time=10), \

1842 nvme0n1.ioworker(io\_size=8, lba\_align=8, lba\_r

1843 qdepth=16, read\_percentage=0,

1844 iops=10, time=10):

1845 pass

1846

1847

Run Test | Debug Test

1848 def test\_ioworker\_stress(nvme0n1):

1849 for i in range(2000):

1850 with nvme0n1.ioworker(io\_size=8, lba\_align=8, l

1851 qdepth=16, read\_percentag

1852 logging.info(i)

1853

1854

1855 @pytest.mark.parametrize("repeat", range(500))

Run Test (Multiple) | Debug Test (Multiple)

1856 def test\_ioworker\_stress\_multiple(nvme0n1, repeat):

1857 l = []

1858 for i in range(15):

1859 a = nvme0n1.ioworker(io\_size=8, lba\_align=8,

1860 lba\_random=True, qdepth=51

1861 read\_percentage=100, time=

1862

1863 l.append(a)

1864

1865 for a in l:

1866 r = a.close()

1867

1868

1869

Run Test | Debug Test

1868 def test\_ioworker\_longtime(nvme0n1, verify):

1869 l = []

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

driver\_test.py::test\_ioworker\_vscod...showcase

live log teardown

[2019-04-22 17:32:58.194] INFO pciaddr(19): running tests on DUT 01:00:0

PASSED

[100%]

live log teardown

[2019-04-22 17:33:12.695] INFO script(33): test duration: 12.254 sec

/home/cranechu/pynvme/.vscode/pytest.sudo.sh: line 1: 3819 Segmentation fault sudo /usr/bin/python3 -B -m pytest \$@

Error: TypeError: Cannot read property 'testsuite' of null

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

driver\_test.py::test\_ioworker\_vscod...showcase

live log setup

[2019-04-22 17:33:27.018] INFO pciaddr(19): running tests on DUT 01:00:0

There was an error in running the tests.

Source: Python (Extension)

View Output

master\*

Python 3.7.3 64-bit

Running Tests |

Ln 1, Col 1

Spaces: 4

Plain Text



# Welcome to use and contribute

The screenshot shows the GitHub repository page for `cranechu / pynvme`. At the top, there are buttons for `Unwatch` (4), `Unstar` (10), and `Fork` (4). Below this is a navigation bar with tabs for `Code`, `Issues` (0), `Pull requests` (0), `Projects` (0), `Wiki`, `Insights`, and `Settings`. The repository description is "test NVMe devices in Python" with an `Edit` button. Below the description are tags for `nvme`, `ssd`, `driver`, `spdk`, `test`, `python`, `linux`, and `storage`, along with a `Manage topics` link. A summary bar shows `168` commits, `1` branch, `5` releases, `1` contributor, and the `BSD-3-Clause` license. Below this is a section for the `master` branch with a `New pull request` button and buttons for `Create new file`, `Upload files`, `Find File`, and `Clone or download`. The commit history table shows the following entries:

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

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