Why SSD developers need pynvme?

and why pynvme needs SPDK?

https://github.com/cranechu/pynvme

SSD Testing

SAMSUNG	Kingston TECHNOLOGY	intel	WD	Lenovo	TOSHIBA Leading Innovation >>>	HIKVISION 海康威视	PLEXTOR True as Promised.
hp	SanDisk 闪迪	GLOWAY	COLORFUL 七彩虹	威刚（ADATA）	Asgard	tigo 金泰克	crucial by micron
TECLAST 台电	KingBank 金百达	影驰	acer	Netac 朗科	Pioneer	Lexar 雷克沙	KingSpec
建興 LITEON	EAGET 忆捷	ThinkPad	GIGABYTE	紫光	美商海盗船	TJ	亿储（Billion reserv...

SSD Testing

back Data Center SSDs

Intel® SSD D3-S4610 Series	Intel® SSD DC S3100 Series	Intel® SSD D5-P4326 Series
Intel® SSD DC S4600 Series	Intel® Optane™ SSD DC P4801X Series	Intel® SSD D5-P4320 Series
Intel® SSD D3-S4510 Series	Intel® Optane™ SSD DC P4800X Series	Intel® SSD DC P4101 Series
Intel® SSD DC S4500 Series	Intel® SSD DC P4618 Series	Intel® SSD DC P3700 Series
Intel® SSD DC S3710 Series	Intel® SSD DC P4610 Series	Intel® SSD DC P3608 Series
Intel® SSD DC S3700 Series	Intel® SSD DC P4608 Series	Intel® SSD DC P3600 Series
Intel® SSD DC S3610 Series	Intel® SSD DC P4600 Series	Intel® SSD DC P3520 Series
Intel® SSD DC S3520 Series	Intel® SSD DC P4511 Series	Intel® SSD DC P3500 Series
Intel® SSD DC S3510 Series	Intel® SSD DC P4510 Series	Intel® SSD DC P3100 Series
Intel® SSD DC S3500 Series	Intel® SSD DC P4501 Series	Intel® Optane™ SSD DC D4800X Series
Intel® SSD DC S3320 Series	Intel® SSD DC P4500 Series	Intel® SSD DC D3700 Series
Intel® SSD DC S3110 Series	Intel® SSD D5-P4420 Series	Intel® SSD DC D3600 Series

SSD Testing

MARVELL

SandForce

MICROCHIP

SiliconMotion

STARBLAZE

YEESTOR

NETINT

SAGE MICROELECTRONIQUE

INNOGRIT

NGD systems

FADU

PHISON Knows What You Need
Phison Electronics Corp.

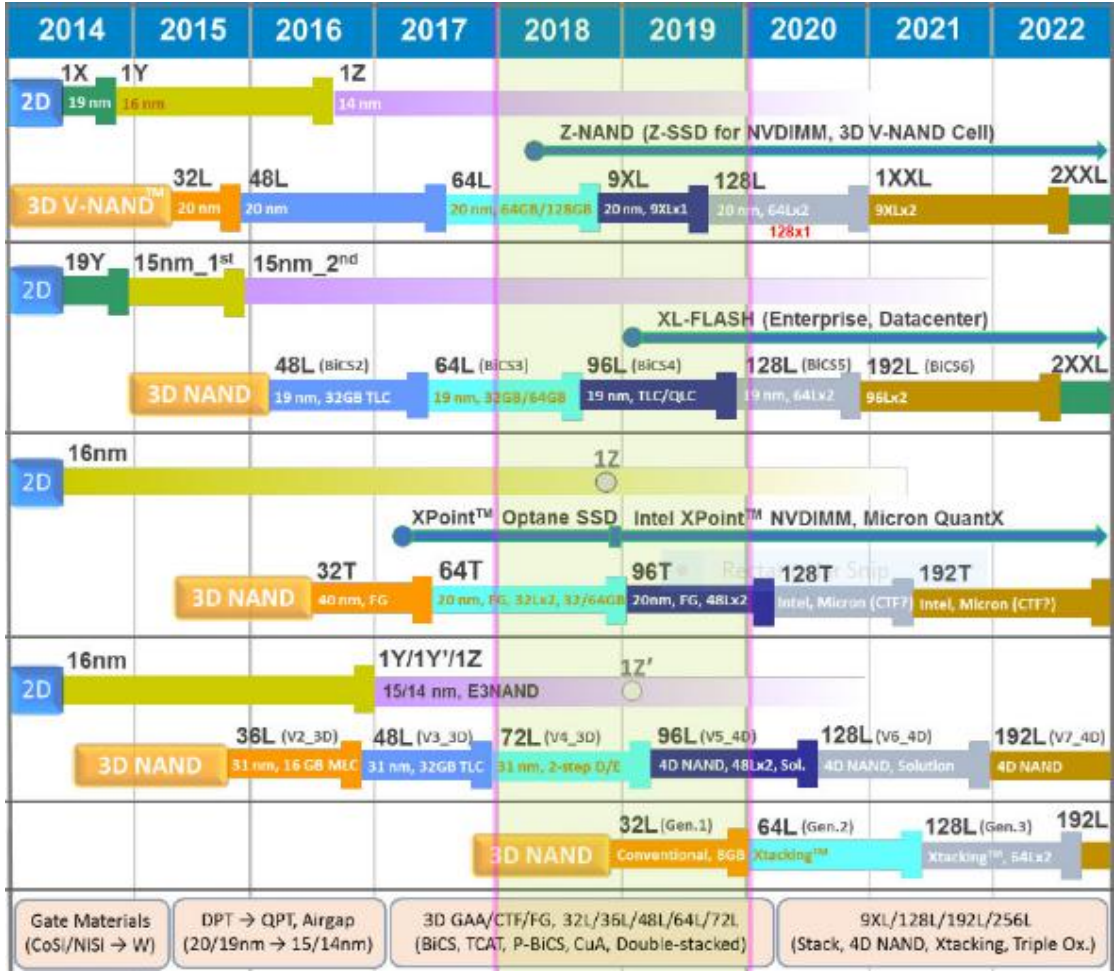
MaXio

CNEXLABS

SSD Testing

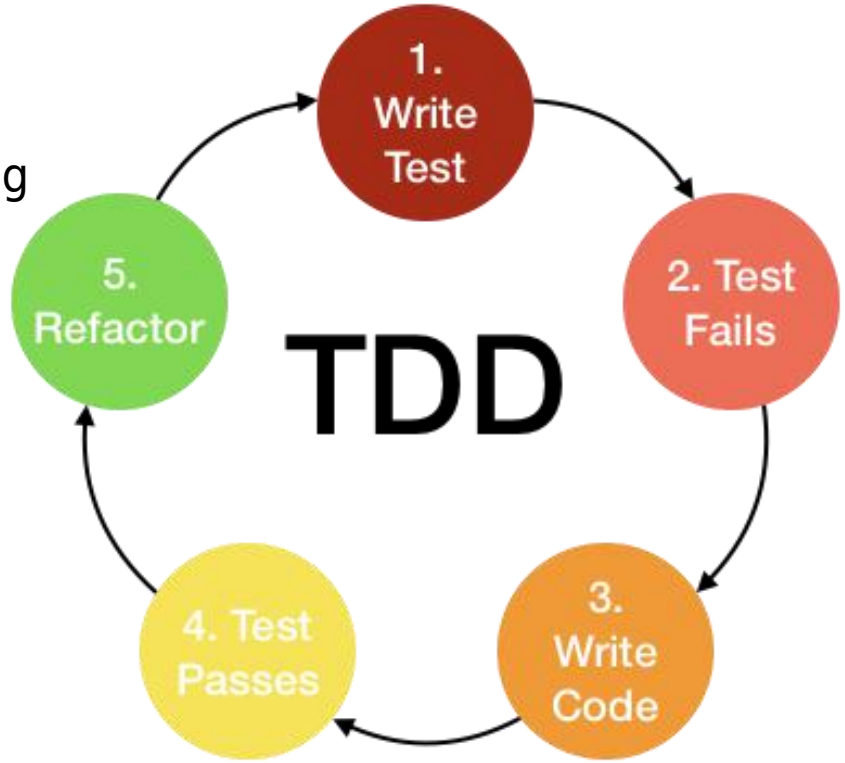
NVMe spec:

- 1.0e (Jan. 2013)
- 1.1b (July 2014)
- 1.2 (Nov. 2014)
  - 1.2a (Oct. 2015)
  - 1.2b (June 2016)
  - 1.2.1 (June 2016)
- 1.3 (May 2017)
  - 1.3a (Oct. 2017)
  - 1.3b (May 2018)
  - 1.3c (May 2018)
  - 1.3d (March 2019)
- 1.4 (June 2019)



Agile and TDD

- Challenge
  - NAND is changing
  - Applications and specifications are changing
  - Diversity on NAND and controllers
- SSD development should be ...
  - fast iteration
  - customer-oriented
  - open to change
- Waterfall v.s. Agile
  - Test-driven development (TDD)



Test Driver

- Embedded devices provide very limited resources

- We need A test-dedicated NVMe driver in host platforms:
  - exports device's features to host
  - exports device's flaws to host
  - exports device's performance to host
  - friendly to test script development
  - friendly to firmware debug
  - friendly to CI

Existed Tools

	tnvme	DM	fio	*Marks
feature	√	√	×	×
performance	×	√	√	√
scripts	√	×	√	×
debug	×	√	×	×
CI	√	×	√	×
driver	dnvme	OFA	Linux	Windows

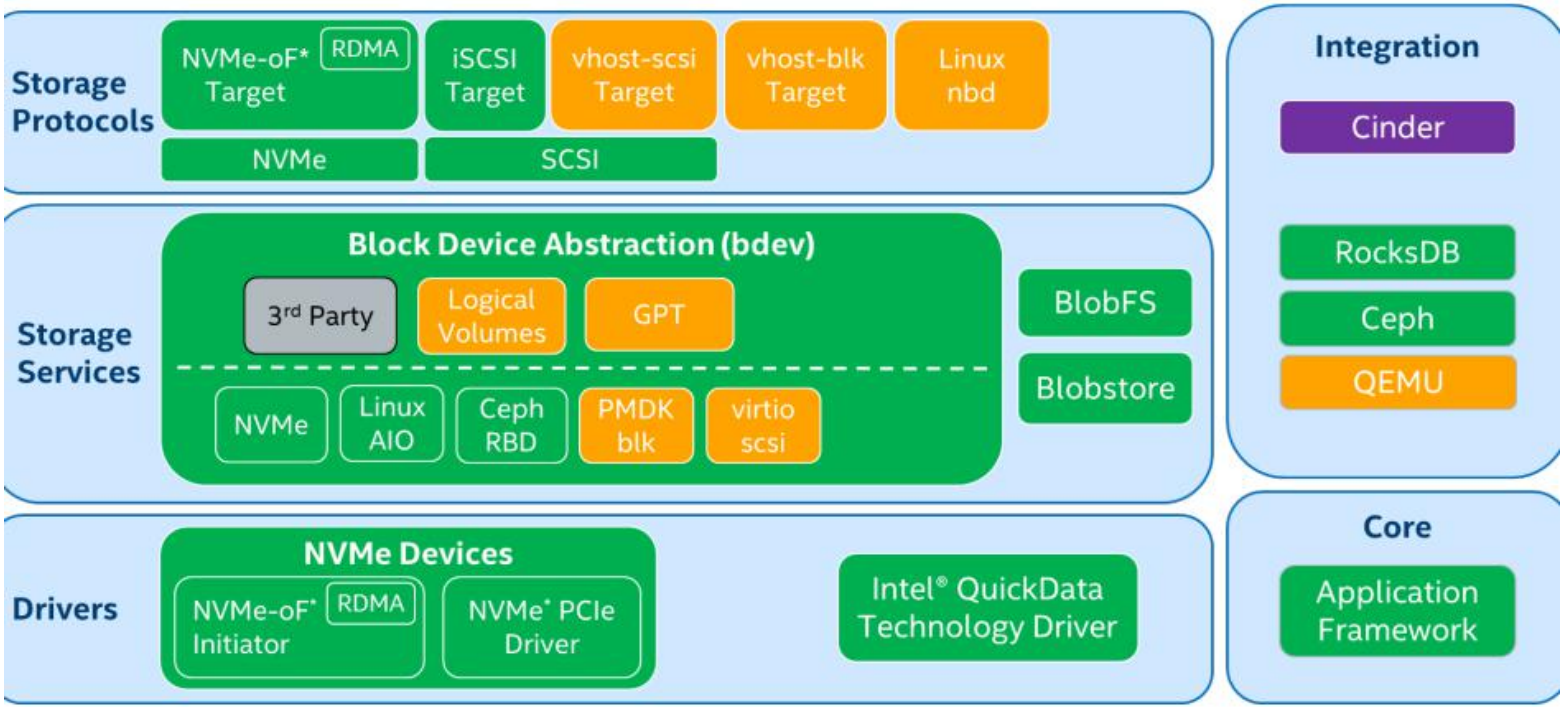
pynvme

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.

Why SPDK?

- ✓ user space:
  - easy for debugging
  - maintainness
- ✓ well modularized:
  - jsonrpc: for the vscode plugin
  - memzone: share memory between processers
  - crc32
- ✓ best performance:
  - test efficiency
  - stress test
- ✓ open and active
  - SSD, NVMe, NAND are all keeping changing!

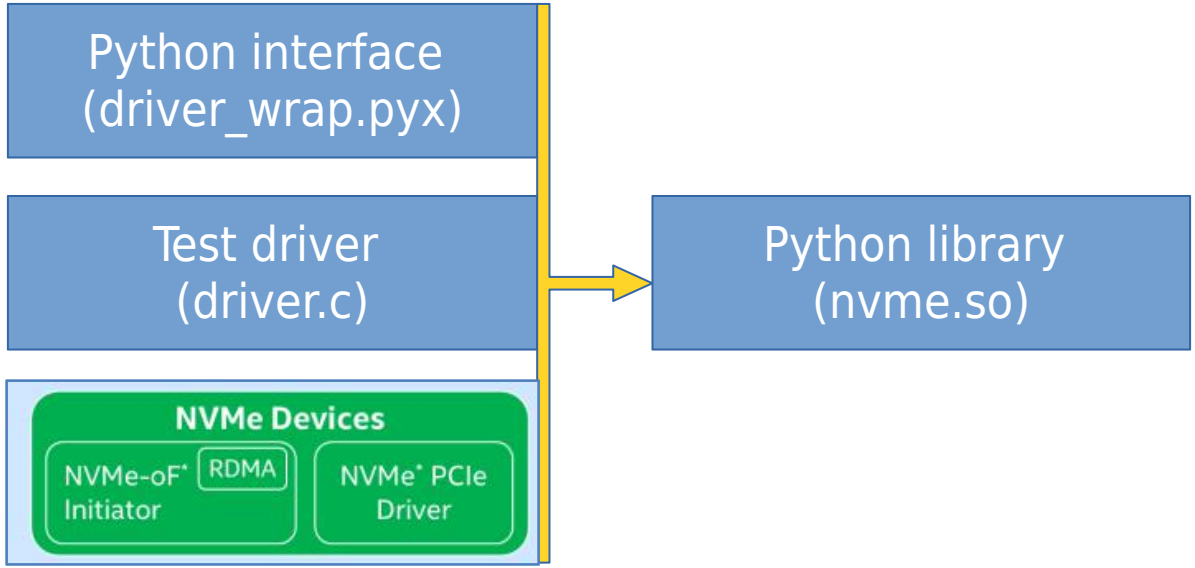
Pynvme is based on SPDK/DPDK



Pynvme Architecture

Build python library with Cython:

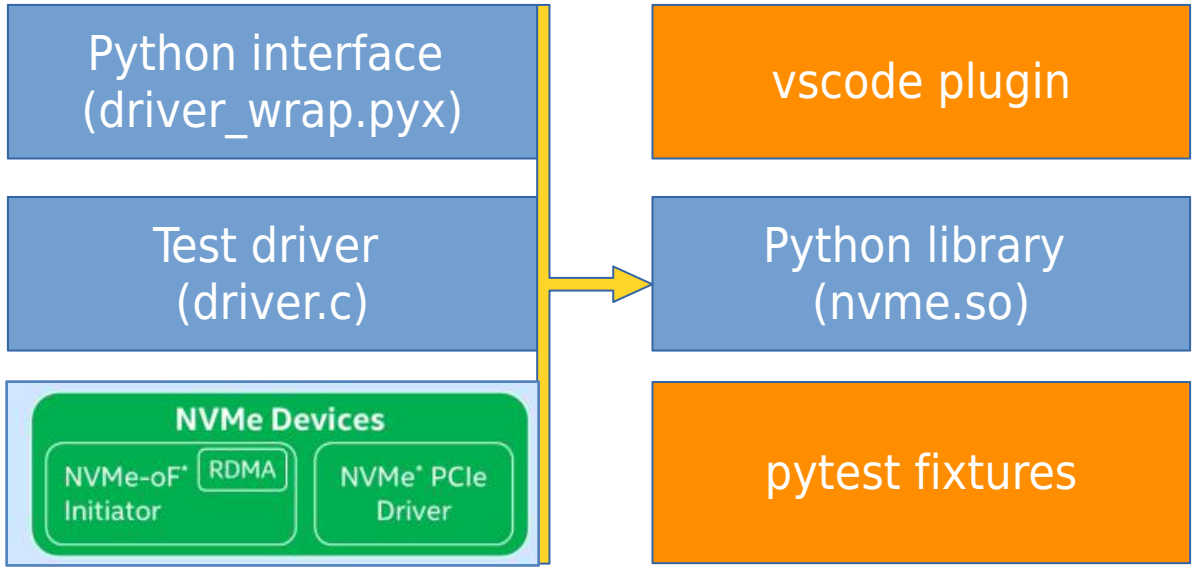
- setup.py
- driver.c
- driver.h
- cdriver.pxd
- driver_wrap.pyx
- Makefile



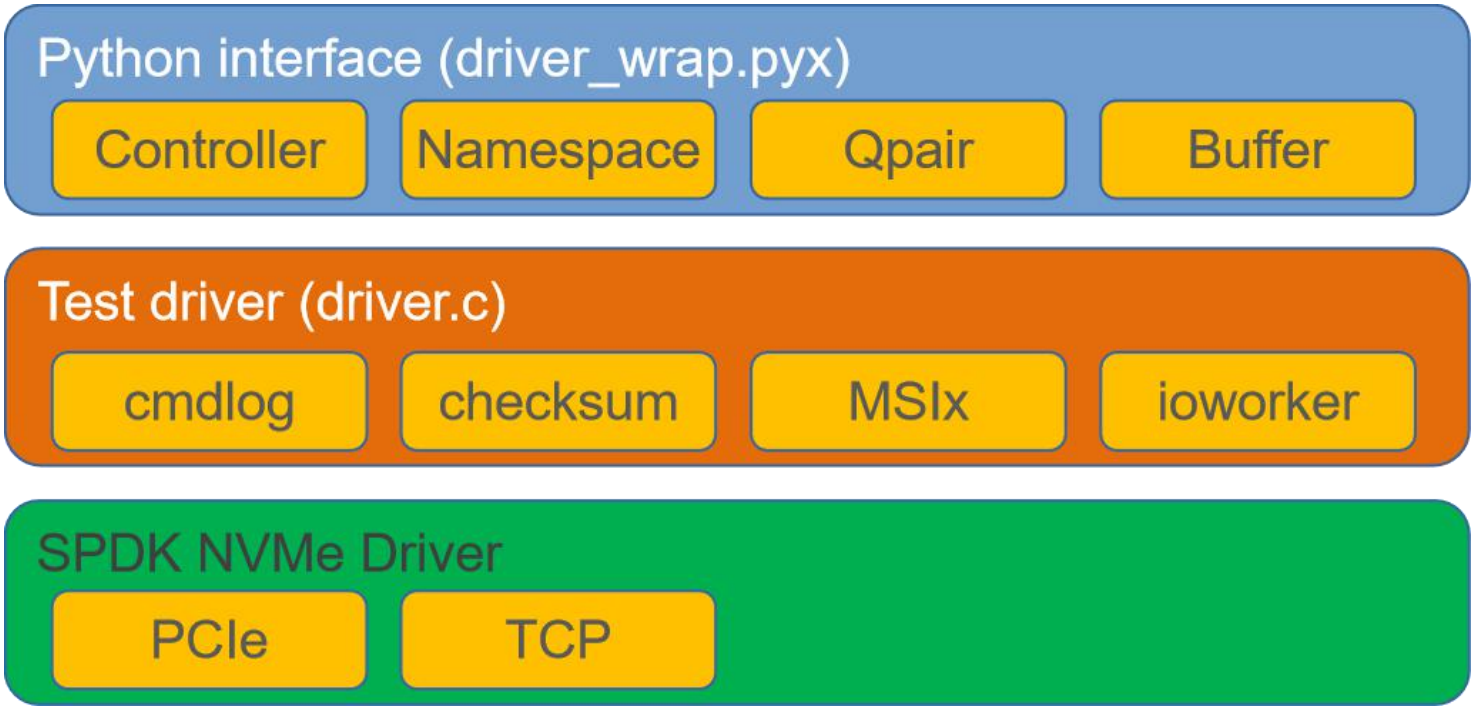
Pynvme Architecture

Organize test cases in pytest:

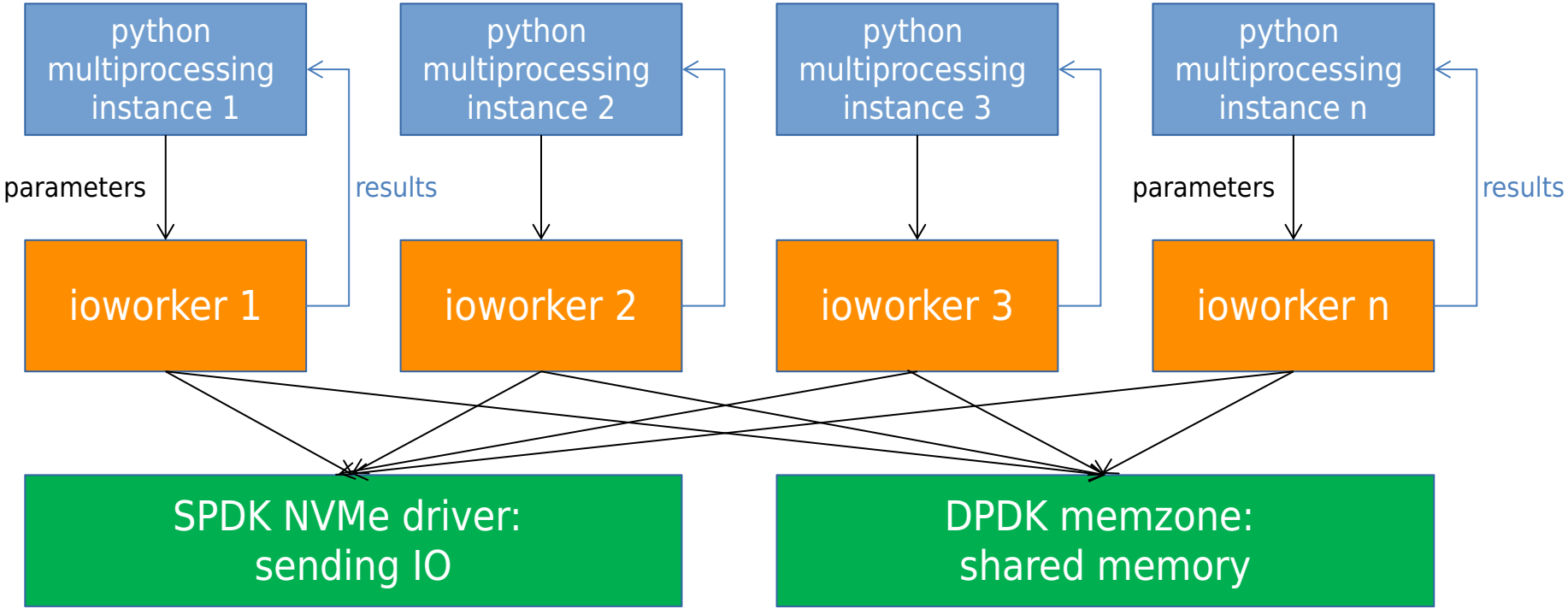
- mvme.so
- pytest.ini
- conftest.py
- driver_test.py



Pynvme Architecture



IOWorker


python multiprocessing instance 1
python multiprocessing instance 2
python multiprocessing instance 3
python multiprocessing instance n
parameters
results
parameters
results
ioworker 1
ioworker 2
ioworker 3
ioworker n
SPDK NVMe driver: sending IO
DPDK memzone: shared memory

Why Python?

- ✔ Many beautiful and mature libraries
  - ✔ Cython
  - ✔ pytest
  - ✔ logging
  - ✔ multiprocessing
  - ✔ pydoc, os, io, time, pytemperature, statistics, yaml, json, struct, matplotlib, ...
- ✔ Friendly to test script development
  - VSCode, Emacs, Pycharm, ...
- ✔ Friendly to CI: develop firmware softly

  pipeline passed
  - Introducing software methodologies, processes and tools to firmware.

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

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 providers an extension to monitor device status and cmdlog of every qpair, via jsonrpc. To install the extension:
  - *code --install-extension pynvme-console-1.x.x.vsix*

- Add DUT pci address to .vscode/settings.json
  - get the BDF address with *lspci*

- *make setup; code .* # launch the vscode

DEBUG

debug pynvme script

VARIABLES

Locals

```
cdw0: 0
nvme0n1: <nvme.Namespace object at 0x7fffe02e1bd8>
qpair: <qpair: 2>
read_buf: <buffer name: buffer>
status1: 1
```

debug panels

WATCH

```
write_buf[10:21]: NameError("name 'write_buf' is not defined")
read_buf[10:21]: b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

breakpoint

demo_test.py

```
import pytest
import nvme as d

Run Test | Debug Test
def test_hello_world(nvme0, nvme0n1: d.Namespace):
    write_buf = d.Buffer(512)
    write_buf[10:21] = b'hello world'
    qpair = d.Qpair(nvme0, 16)

    read_buf = d.Buffer(512)
    def write_cb(cdw0, status1):
        nvme0n1.read(qpair, read_buf, 0, 1)
    nvme0n1.write(qpair, write_buf, 0, 1, cb=write_cb)
    qpair.waitdone(2)
    assert read_buf[10:21] == b'hello world'
```

test scripts

CALL STACK

PAUSED ON BREAKPOINT

write_cb	demo_test.py	11:1
test_hello_world	demo_test.py	13:1
run	testlauncher.py	50:1
<module>	testlauncher.py	62:1

call stack

BREAKPOINTS

- [ ] Raised Exceptions
- [x] Uncaught Exceptions
- [x] demo_test.py scripts 11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: Python Debug Conso

```
 sudo /usr/bin/python3 /home/cranechu/.vscode/extensions/ms-python.python-2019.4.1
1987/pythonFiles/ptvsd_launcher.py --default --client --host localhost --port 3878
1 /home/cranechu/.vscode/extensions/ms-python.python-2019.4.11987/pythonFiles/test
launcher.py /home/cranechu/pynvme pytest --junitxml=/tmp/tmp-12479aQxuEn0mpwap.xml
 --pciaddr=01:00.0 ./scripts/demo_test.py::test_hello_world
============================= test session starts ==============================
platform linux -- Python 3.7.3, pytest-4.3.1, py-1.8.0, pluggy-0.9.0 -- /usr/bin/p
ython3
cachedir: .pytest_cache
rootdir: /home/cranechu/pynvme, inifile: pytest.ini
plugins: cov-2.6.1
collected 1 item

scripts/demo_test.py::test_hello_world
------------------------------- live log setup --------------------------------
[2019-04-29 22:20:36.206] INFO pciaddr(19): running tests on DUT 01:00.0
```

test log

master*+ Python 3.7.3 64-bit 0 0 No Tests Ran Ln 11, Col 1 Spaces: 4 UTF-8 LF Python

TEST

PYTHON

- driver_test.py
- scripts
  - test_trim_basic.py
    - test_trim_basic

test items

PYNVME QPAIRS

QPair 00:

QPair 01:

QPair 02: ...

qpairs

test_trim_basic.py

```python
import time
import pytest
import logging
import nvme as d


Run Test | Debug Test
def test_trim_basic(nvme0, nvme0n1, verify):
    GB = 1024*1024*1024
    all_zeor_databuf = d.Buffer(512)
    trimbuf = d.Buffer(4096)
    q = d.Qpair(nvme0, 32)

    # DUT info
    logging.info("model number: %s" % nvme0.id_data(63, 24, str))
    logging.info("firmware revision: %s" % nvme0.id_data(71, 64, str))

    # write
    logging.info("write data in 10G ~ 20G")
    io_size = 128*1024//512
    start_lba = 10*GB//512
    lba_count = 10*GB//512
    nvme0n1.ioworker(io_size = io_size,
                     lba_align = io_size,
                     lba_random = False,
                     read_percentage = 0,
                     lba_start = start_lba,
                     io_count = lba_count//io_size,
                     qdepth = 128).start().close()

    # verify data after write, data should be modified
    with pytest.warns(UserWarning, match="ERROR status: 02/85"):
        nvme0n1.compare(q, all_zeor_databuf, start_lba).waitdone()

    # get the empty trim time
    trimbuf.set_dsm_range(0, 0, 0)
    trim_cmd = nvme0n1.dsm(q, trimbuf, 1).waitdone() # first call is longer, due to
    start_time = time.time()
    trim_cmd = nvme0n1.dsm(q, trimbuf, 1).waitdone()
    empty_trim_time = time.time()-start_time

    # the trim time on device-side only
    logging.info("trim the 10G data from LBA 0x%lx" % start_lba)
    trimbuf.set_dsm_range(0, start_lba, lba_count)
    start_time = time.time()
    trim_cmd = nvme0n1.dsm(q, trimbuf, 1).waitdone()
    trim_time = time.time()-start_time-empty_trim_time
    logging.info("trim bandwidth: %0.2fGB/s" % (10/trim_time))

    # verify after trim
    nvme0n1.compare(q, all_zeor_databuf, start_lba).waitdone()
```

test scripts

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL — Python 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-5551D6921Gjfy7tm.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 ---------------------------------
```

test log

trim_script* Python 3.7.3 64-bit 0 0 Running Tests / Ln 50, Col 63 Spaces: 4 UTF-8 LF Python

demo_test.py

```python
import time
import pytest
import logging
import nvme as d
from pytemperature import k2c

Run Test | Debug Test
def test_ioworker_with_temperature(nvme0, nvme0n1):
    smart_log = d.Buffer(512, "smart log")
    with nvme0n1.ioworker(io_size=8, lba_align=16, lba_random=True,
                          qdepth=16, read_percentage=0, time=30):
        for i in range(40):
            nvme0.getlogpage(0x02, smart_log, 512).waitdone()
            ktemp = smart_log.data(2, 1)
            ctemp = k2c(ktemp)
            logging.info("temperature: %0.2f degreeC" % ctemp)
            time.sleep(1)
```

test scripts

define IO patterns in ioworker's parameter list, and run in a separated process

monitor temperature at the same time

CMDLOG Q0

```
1556546728.807225: [cmd: Get Log Page]
0x005f0002, 0xffffffff, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x7439a000, 0x00000001
0x00000000, 0x00000000, 0x007f0002, 0x00000000
0x00000000, 0x00000000, 0x00000000, 0x00000000
1556546728.809293: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x00000023, 0x0001005f

1556546727.804234: [cmd: Get Log Page]
0x005f0002, 0xffffffff, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x7439a000, 0x00000001
0x00000000, 0x00000000, 0x007f0002, 0x00000000
0x00000000, 0x00000000, 0x00000000, 0x00000000
1556546727.806340: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x00000022, 0x0001005f

1556546726.801221: [cmd: Get Log Page]
```

test log

cmdlog

PROBLEMS OUTPUT Python Test Log

```
temperature: 44.85 degreeC
[2019-04-29 22:05:15.756] INFO
temperature: 44.85 degreeC
[2019-04-29 22:05:16.760] INFO
temperature: 45.85 degreeC
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
```

master*+ Python 3.7.3 64-bit 0 0 Running Tests \ Ln 1, Col 1 Spaces: 4 Plain Text

pynvme goes to 1.x

cranechu / pynvme

Watch 8 | Unstar 32 | Fork 15

Code | Issues 11 | Pull requests 0 | Projects 2 | Wiki | Security | Insights | Settings

Branch: master | pynvme / README.md | Find file | Copy path

cranechu change homepage video 6648db0 on Jun 3

1 contributor

1228 lines (902 sloc) | 39.1 KB | Raw | Blame | History

pynvme

test NVMe devices in Python. [https://github.com/cranechu/pynvme]

pipeline passed | license BSD-3-Clause | release v1.0

Thanks!

pynvme

Q & A

Live Demo