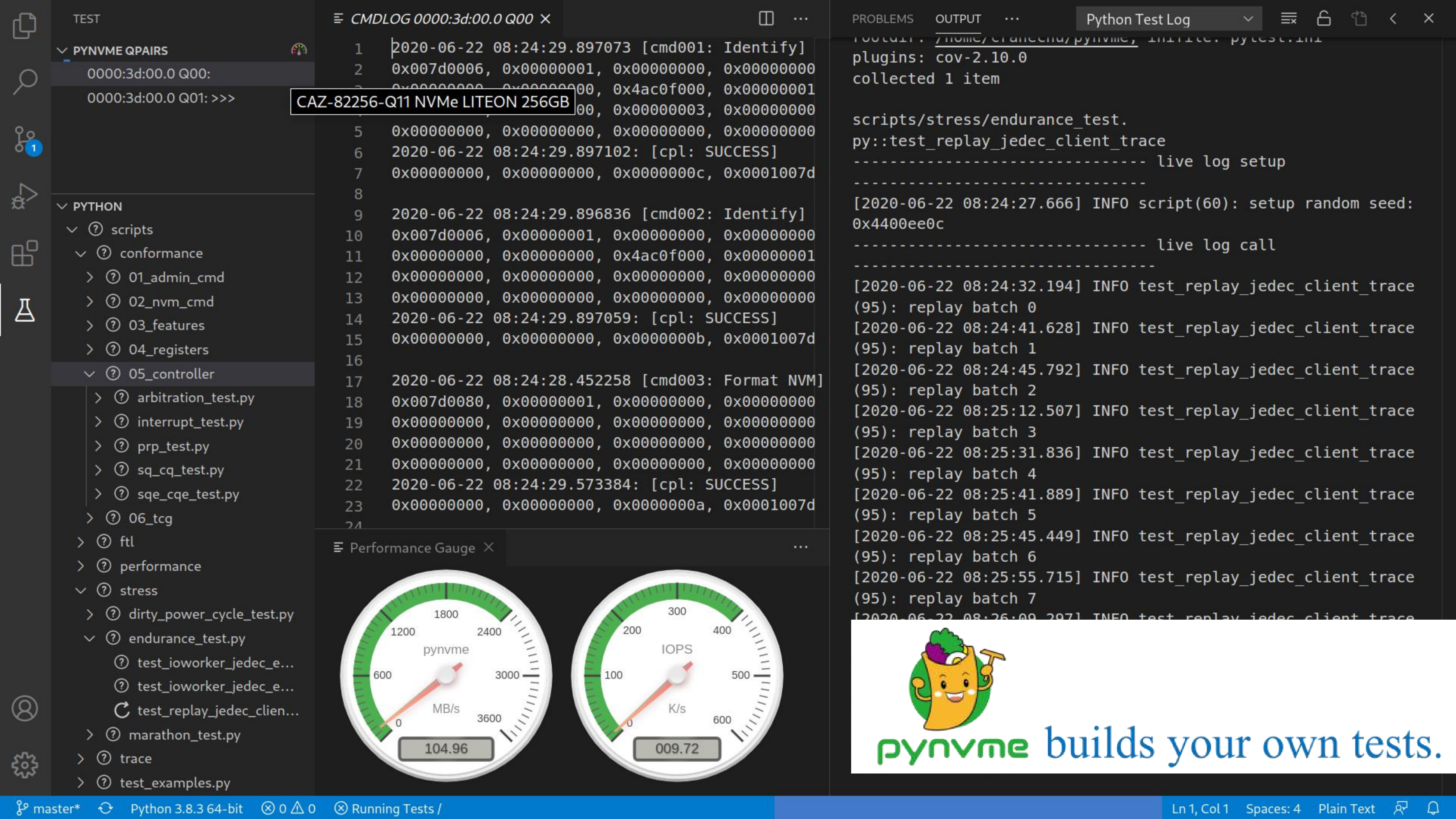


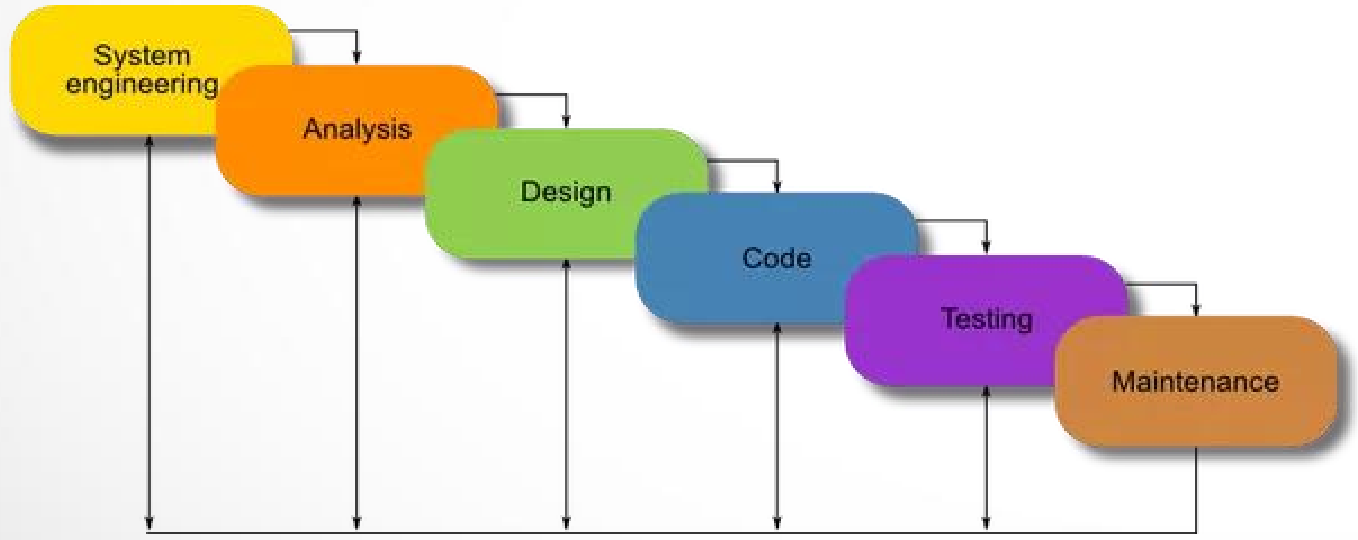
```
2020-06-22 08:24:29.897073 [cmd001: Identify]
0x007d0006, 0x00000001, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x4ac0f000, 0x00000001
0x00000000, 0x00000000, 0x00000003, 0x00000000
0x00000000, 0x00000000, 0x00000000, 0x00000000
2020-06-22 08:24:29.897102: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x0000000c, 0x0001007d

2020-06-22 08:24:29.896836 [cmd002: Identify]
0x007d0006, 0x00000001, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x4ac0f000, 0x00000001
0x00000000, 0x00000000, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x00000000, 0x00000000
2020-06-22 08:24:29.897059: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x0000000b, 0x0001007d

2020-06-22 08:24:28.452258 [cmd003: Format NVM]
0x007d0080, 0x00000001, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x00000000, 0x00000000
2020-06-22 08:24:29.573384: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x0000000a, 0x0001007d
```

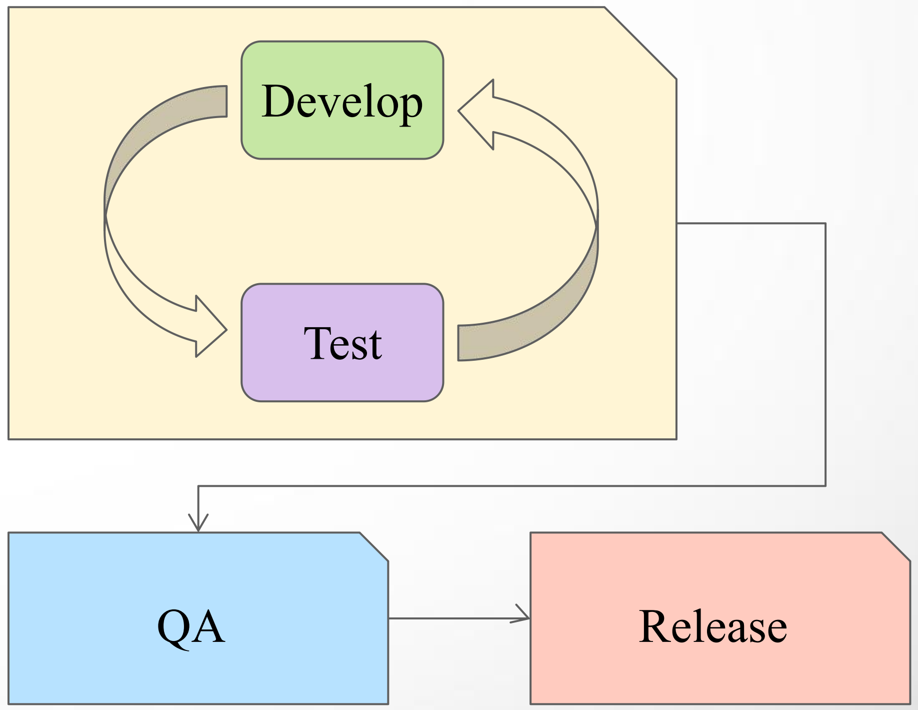
```
plugins: cov-2.10.0
collected 1 item

scripts/stress/endurance_test.
py::test_replay_jedec_client_trace
-------------------------------- live log setup
--------------------------------
[2020-06-22 08:24:27.666] INFO script(60): setup random seed:
0x4400ee0c
-------------------------------- live log call
---------------------------------
[2020-06-22 08:24:32.194] INFO test_replay_jedec_client_trace
(95): replay batch 0
[2020-06-22 08:24:41.628] INFO test_replay_jedec_client_trace
(95): replay batch 1
[2020-06-22 08:24:45.792] INFO test_replay_jedec_client_trace
(95): replay batch 2
[2020-06-22 08:25:12.507] INFO test_replay_jedec_client_trace
(95): replay batch 3
[2020-06-22 08:25:31.836] INFO test_replay_jedec_client_trace
(95): replay batch 4
[2020-06-22 08:25:41.889] INFO test_replay_jedec_client_trace
(95): replay batch 5
[2020-06-22 08:25:45.449] INFO test_replay_jedec_client_trace
(95): replay batch 6
[2020-06-22 08:25:55.715] INFO test_replay_jedec_client_trace
(95): replay batch 7
```

Test v.s. Quality Assurance

- Develop => QA
- Waterfall

- (Develop+Test) => QA
- Agile

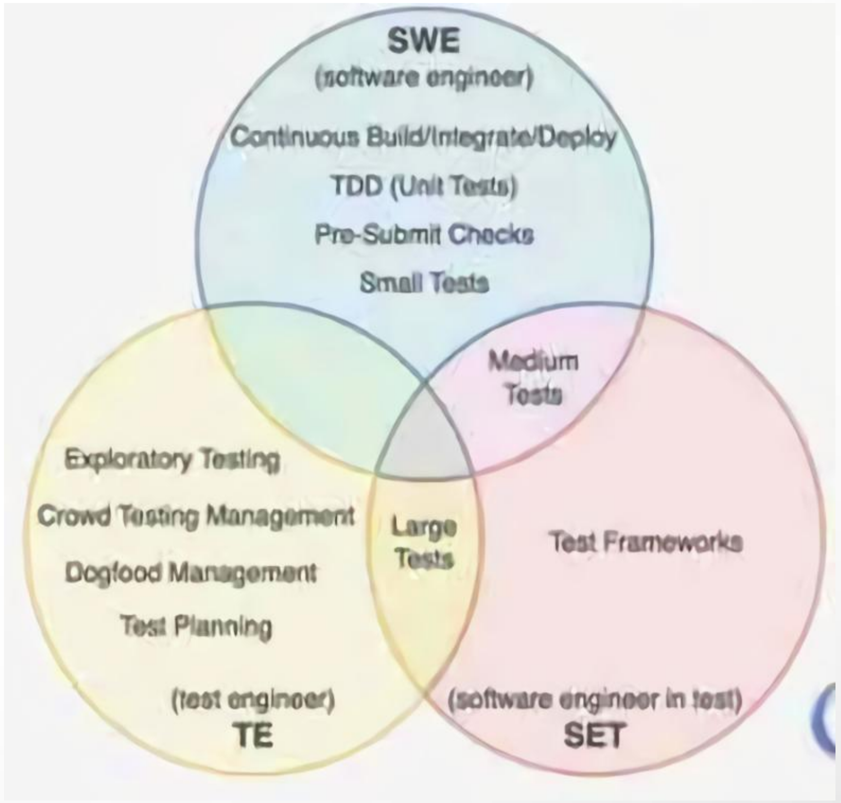




Experience of SSD Firmware Development

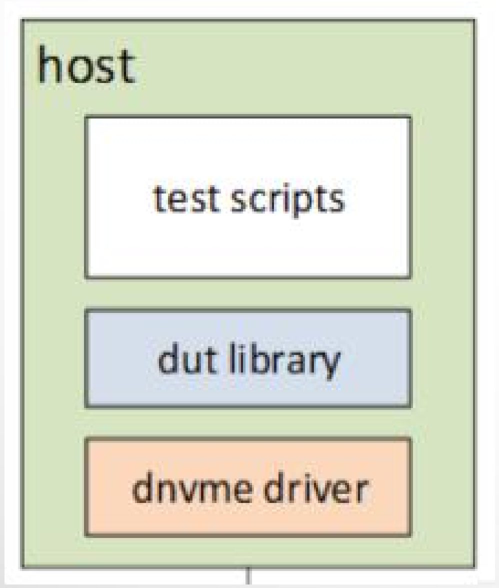
- Firmware Engineer & Test Engineer
  - FE develops firmware code
  - TE develops scripts to test firmware code
- test tool (DriveMaster):
  - difficult to develop and maintain
  - form a gap between FE and TE
  - source code is not open

- How Google Tests Software
  - SWE(FE), SET(TE), TE(QA)



Experience in a Startup: dnvme

- dnvme: @2015
  - wrap with python in userspace
  - functional tests
  - integrited with Jenkins
  - firmware engineers develop scripts
  - PASS IOL test on the first try



- challenges:
  - low performance:
    - IOPS, latency, consistency
    - test efficiency
    - stress tests
  - maintainness: kernel module
  - function coverage: PRP, ...
  - GPL License
- dnvme is good at NVMe test, but in-efficient on SSD test.

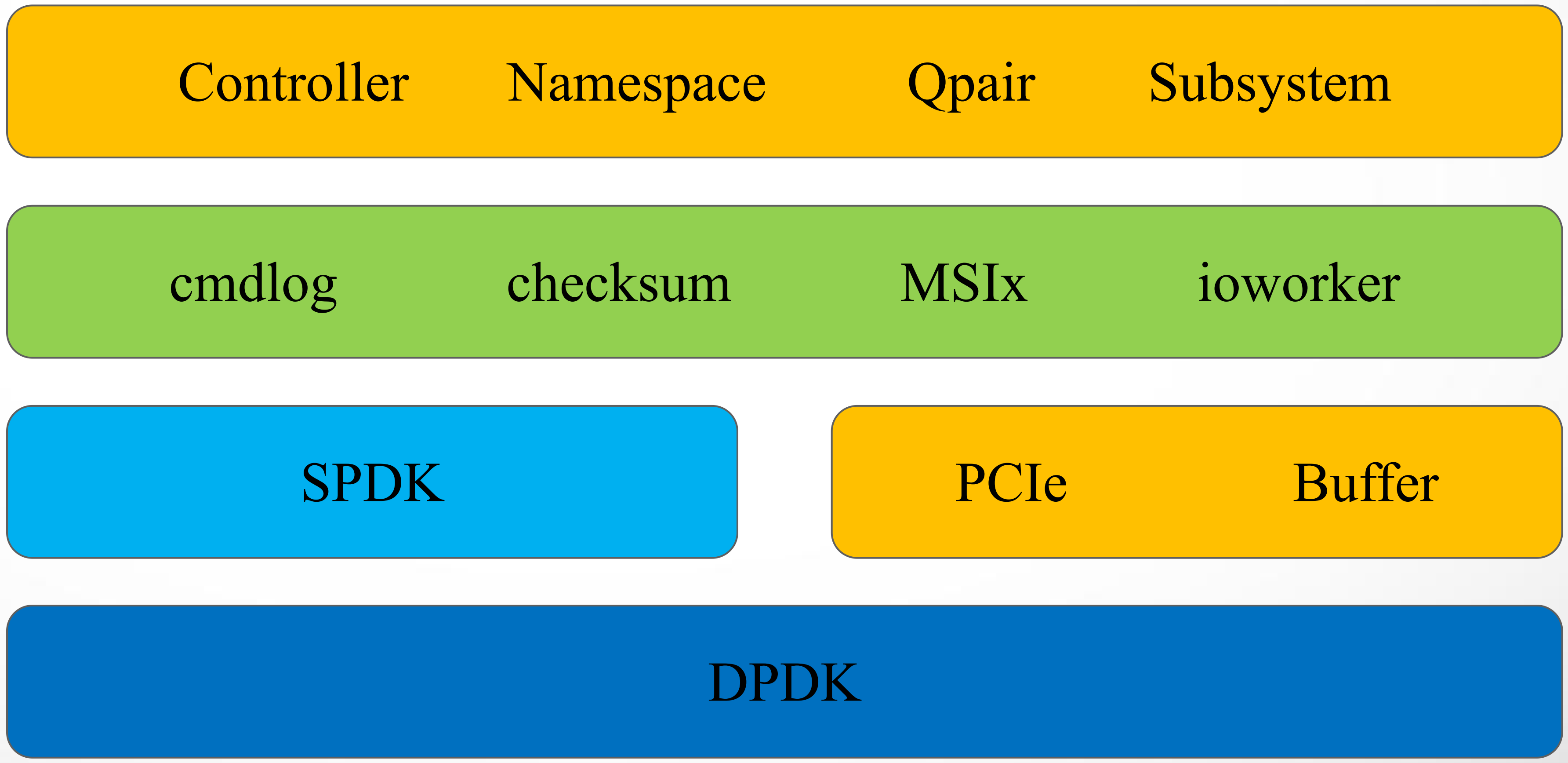
Born for NVMe: SPDK

- SPDK is open from 2016
  - super high performance: 10M IOPS
  - user space application
  - based on DPDK environment, which abstracts low-level resources
    - PCIe
    - memory

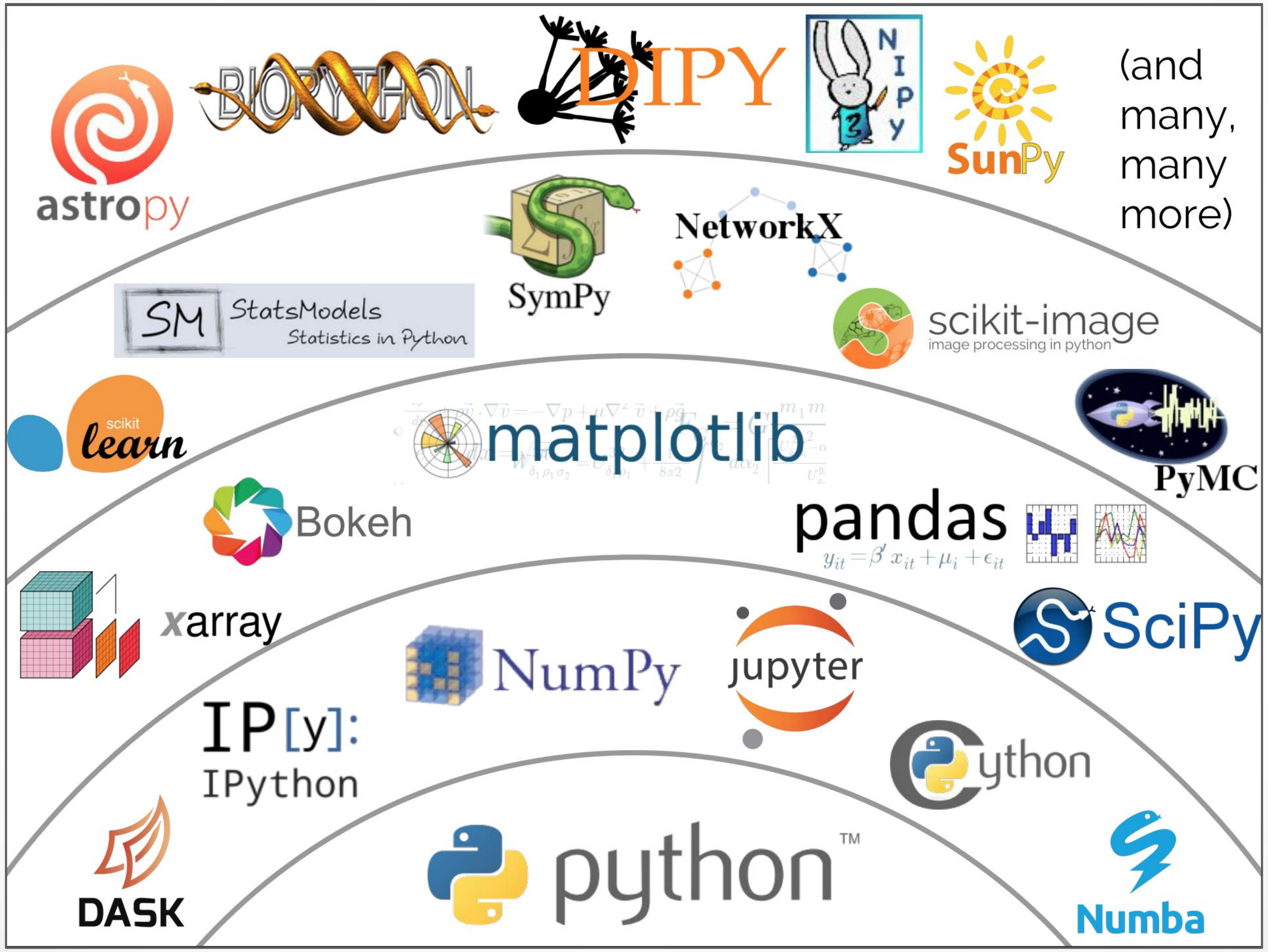
- meet challenges:
  - low performance:
    - IOPS, latency, consistency
    - test efficiency
    - stress tests
  - maintainness: kernel module
  - function coverage: PRP, ...
  - BSD License

pynvme: a software-defined SSD test framework

- pynvme abstracts test resources from low to high level, and provides Python API to access all these resources.

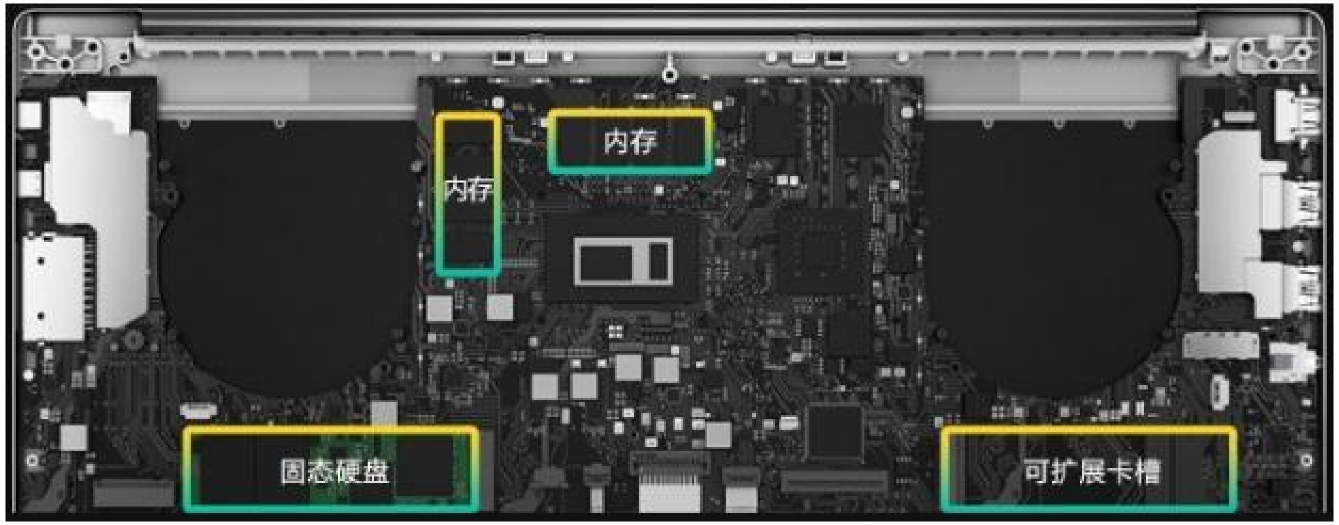


Open to the Ecosystem of Python



Flexible Hardware Configuration

- Single Node
  - laptop
  - workstation
- Mass Deploy
  - server



Test Scripts

```python
import time
import pytest
import logging

import nvme as d


# intuitive, spec, qpair, vscode, debug, cmdlog, assert
def test_hello_world(nvme0, nvme0n1, qpair):
    # prepare data buffer and IO queue
    read_buf = d.Buffer(512)
    write_buf = d.Buffer(512)
    write_buf[10:21] = b'hello world'

    # send write and read command
    def write_cb(cdw0, status1):  # command callback function
        nvme0n1.read(qpair, read_buf, 0, 1)
    nvme0n1.write(qpair, write_buf, 0, 1, cb=write_cb)

    # wait commands complete and verify data
    assert read_buf[10:21] != b'hello world'
    qpair.waitdone(2)
    assert read_buf[10:21] == b'hello world'
```

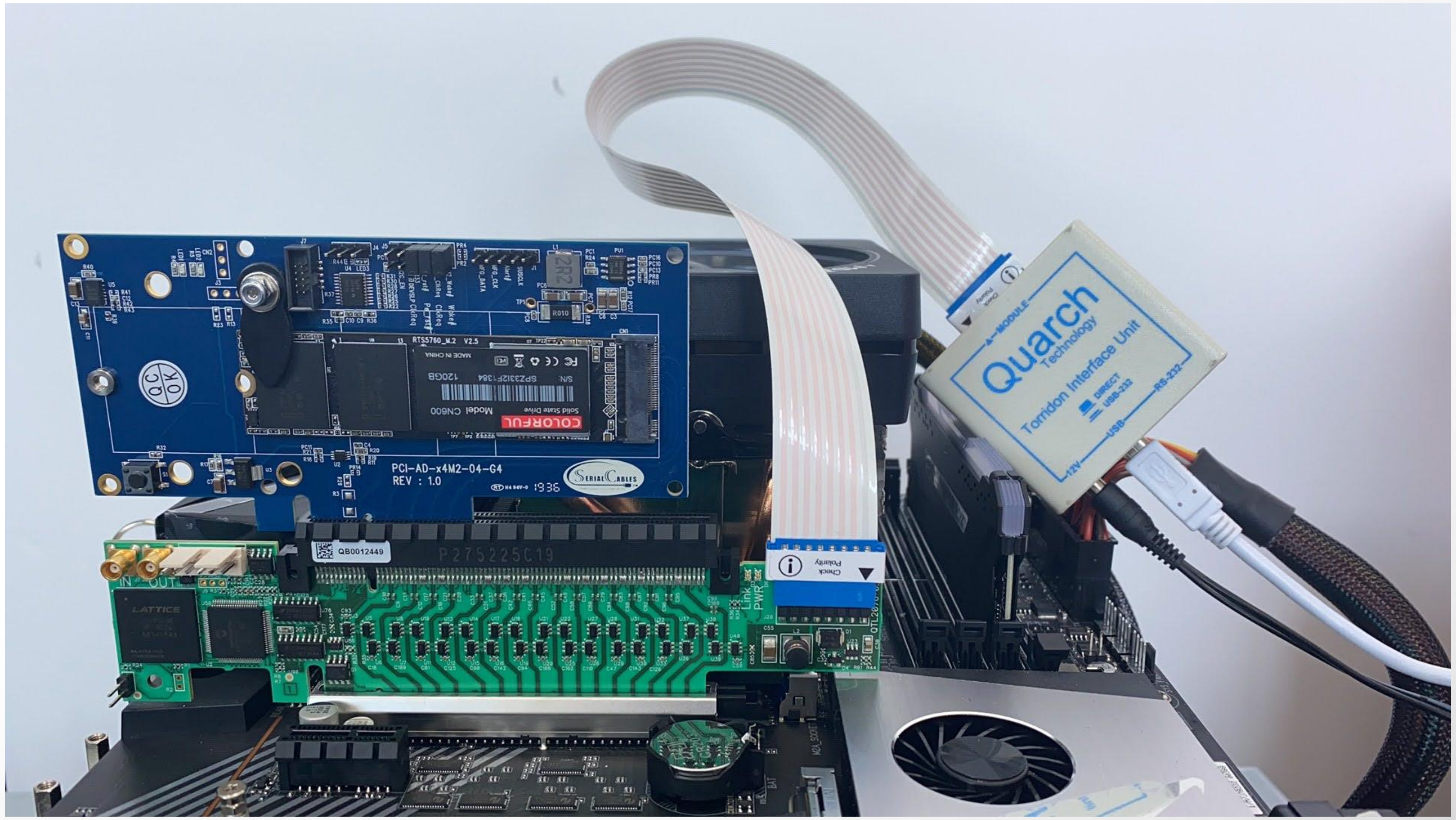
```python
def test_quarch_dirty_power_cycle_single(nvme0, nvme0n1, subsystem, buf, verify):
    # get the unsafe shutdown count before test
    nvme0.getlogpage(2, buf, 512).waitdone()
    orig_unsafe_count = buf.data(159, 144)
    logging.info("unsafe shutdowns: %d" % orig_unsafe_count)
    assert verify == True

    # 128K random write
    cmdlog_list = [None]*1000
    with nvme0n1.ioworker(io_size=256,
                          lba_random=True,
                          read_percentage=30,
                          region_end=256*1000*1000,
                          time=30,
                          qdepth=1024,
                          output_cmdlog_list=cmdlog_list):
        # sudden power loss before the ioworker end
        time.sleep(10)
        subsystem.poweroff()

    # power on and reset controller
    time.sleep(5)
    subsystem.poweron()
    time.sleep(0)
    nvme0.reset()
```

Test Scripts: dirty power cycle

- PCIe Card Module
- Torridon Interface Kit

- poweroff process
  - poweroff when ioworker is alive
  - remove device from system

- poweron process
  - poweron
  - remove kernel driver
  - rescan device
  - nvme initialization



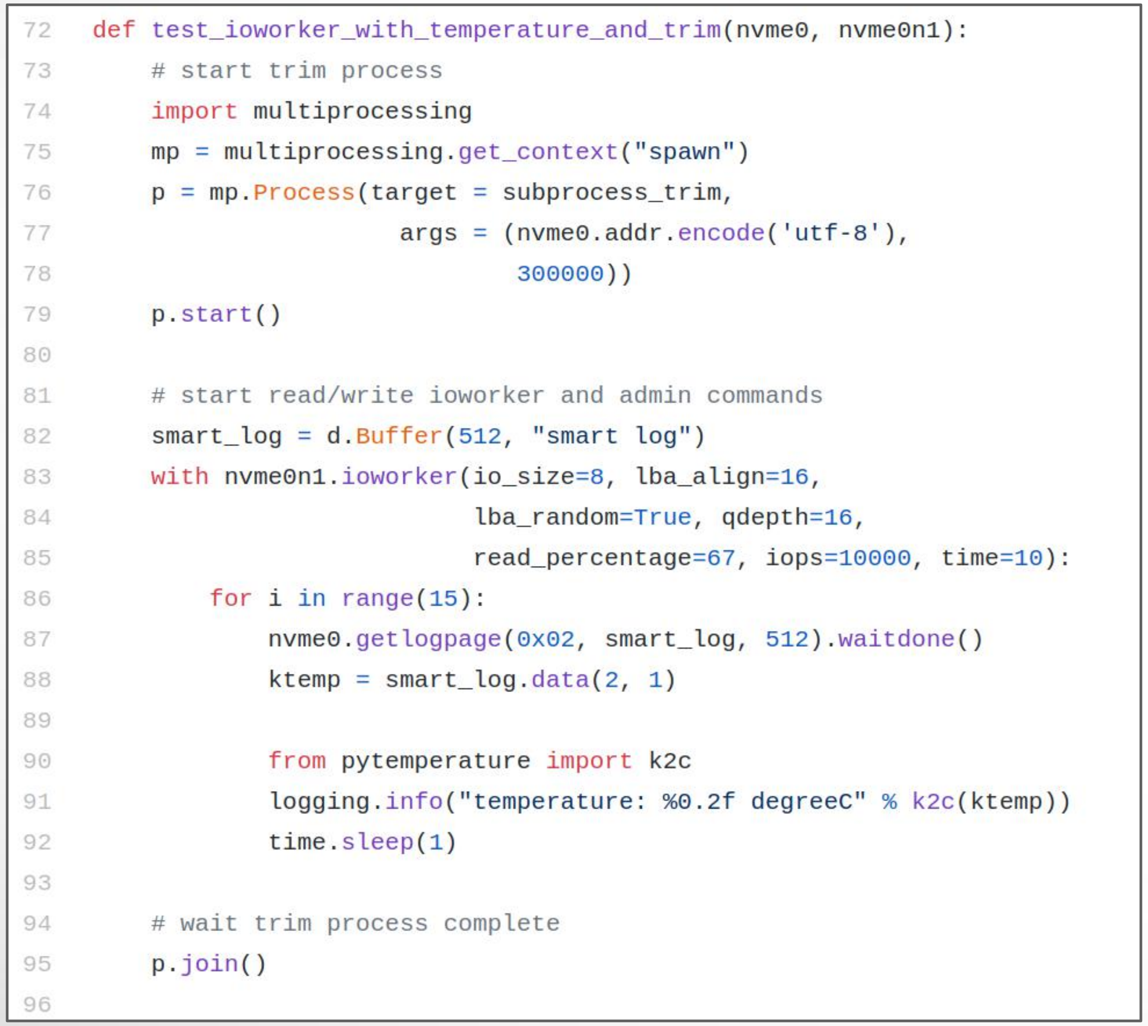
Test Scripts: 3 ways of sending IO

```python
# intuitive, spec, qpair, vscode, debug, cmdlog, assert
def test_hello_world(nvme0, nvme0n1, qpair):
    # prepare data buffer and IO queue
    read_buf = d.Buffer(512)
    write_buf = d.Buffer(512)
    write_buf[10:21] = b'hello world'

    # send write and read command
    def write_cb(cdw0, status1):  # command callback function
        nvme0n1.read(qpair, read_buf, 0, 1)
    nvme0n1.write(qpair, write_buf, 0, 1, cb=write_cb)
```

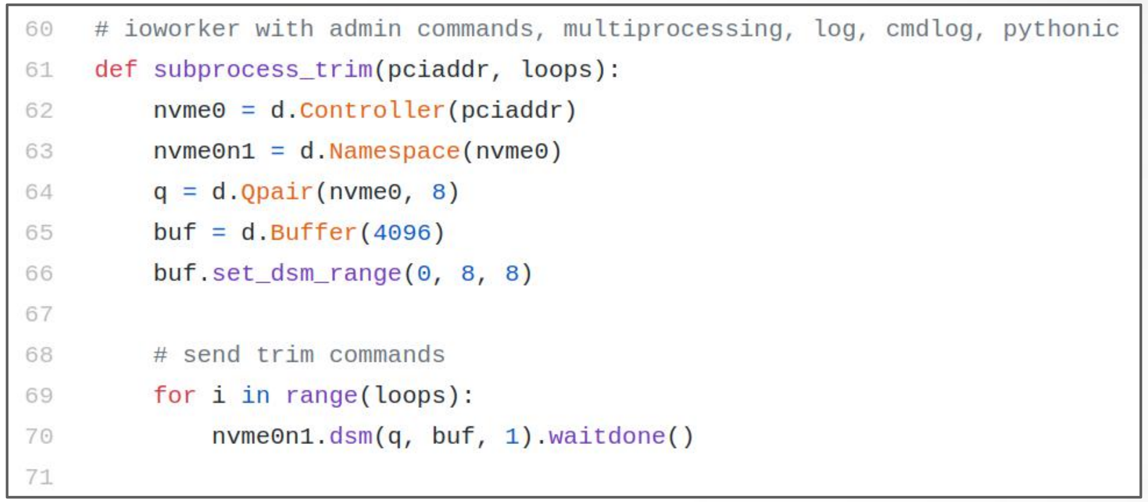
```python
def test_ioworker_simplified(nvme0n1):
    nvme0n1.ioworker(io_size=2, time=2).start().close()
```

```python
def test_send_single_cmd(nvme0):
    cq = IOCQ(nvme0, 1, 10, PRP())
    sq = IOSQ(nvme0, 1, 10, PRP(), cqid=1)

    # first cmd, invalid namespace
    sq[0] = [8] + [0]*15
    sq.tail = 1
    time.sleep(0.1)
    status = (cq[0][3]>>17)&0x7ff
    assert status == 0x000b

    sq.delete()
    cq.delete()
```

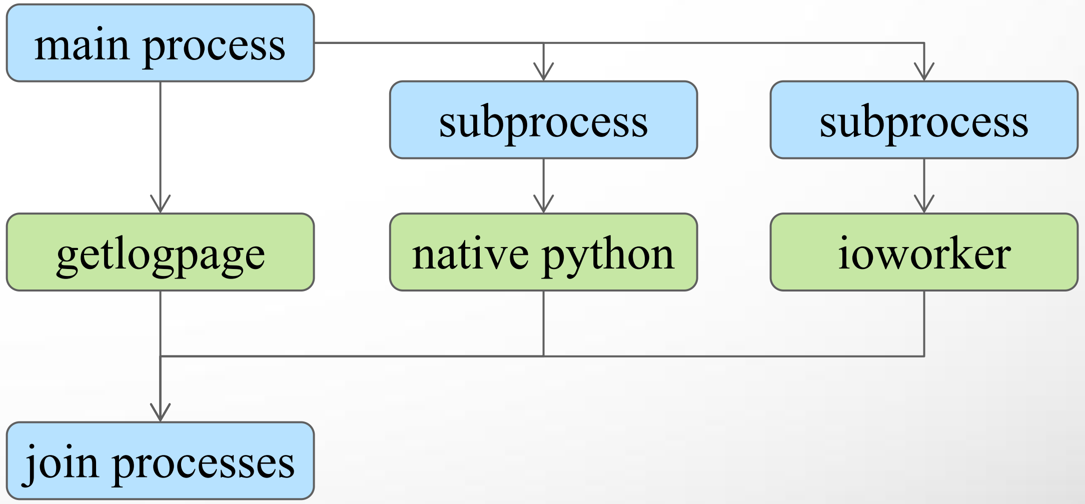
Test Scripts: multiprocessing



```python
def test_ioworker_with_temperature_and_trim(nvme0, nvme0n1):
    # start trim process
    import multiprocessing
    mp = multiprocessing.get_context("spawn")
    p = mp.Process(target = subprocess_trim,
                   args = (nvme0.addr.encode('utf-8'),
                           300000))
    p.start()

    # start read/write ioworker and admin commands
    smart_log = d.Buffer(512, "smart log")
    with nvme0n1.ioworker(io_size=8, lba_align=16,
                          lba_random=True, qdepth=16,
                          read_percentage=67, iops=10000, time=10):
        for i in range(15):
            nvme0.getlogpage(0x02, smart_log, 512).waitdone()
            ktemp = smart_log.data(2, 1)

            from pytemperature import k2c
            logging.info("temperature: %0.2f degreeC" % k2c(ktemp))
            time.sleep(1)

    # wait trim process complete
    p.join()

```



```python
# ioworker with admin commands, multiprocessing, log, cmdlog, pythonic
def subprocess_trim(pciaddr, loops):
    nvme0 = d.Controller(pciaddr)
    nvme0n1 = d.Namespace(nvme0)
    q = d.Qpair(nvme0, 8)
    buf = d.Buffer(4096)
    buf.set_dsm_range(0, 8, 8)

    # send trim commands
    for i in range(loops):
        nvme0n1.dsm(q, buf, 1).waitdone()

```



Test Scripts: NVMe initialization for WRR

```python
def nvme_init_wrr(nvme0):
    logging.info("user defined nvme init")

    nvme0[0x14] = 0
    while not (nvme0[0x1c]&0x1) == 0: pass

    # 3. set admin queue registers
    nvme0.init_adminq()

    # 4. set register cc
    if (nvme0.cap>>17) & 0x1:
        logging.info("set WRR arbitration")
        nvme0[0x14] = 0x00460800
    else:
        nvme0[0x14] = 0x00460000

    # 5. enable cc.en
    nvme0[0x14] = nvme0[0x14] | 1

    # 6. wait csts.rdy to 1
    while not (nvme0[0x1c]&0x1) == 1: pass

    # 7. identify controller
    nvme0.identify(Buffer(4096)).waitdone()

    # 8. create and identify all namespace
    nvme0.init_ns()

    # 9. set/get num of queues
    nvme0.setfeatures(0x7, cdw11=0x00ff00ff).waitdone()
    nvme0.getfeatures(0x7).waitdone()
```

```python
def test_ioworker_with_wrr(pcie):
    nvme0 = Controller(pcie, nvme_init_func=nvme_init_wrr)

    if (nvme0.cap>>17) & 0x1 == 0:
        pytest.skip("WRR is not supported")

    # 8:4:2
    assert nvme0[0x14] == 0x00460801
    nvme0.setfeatures(1, cdw11=0x07030103).waitdone()
    cdw0 = nvme0.getfeatures(1, cdw11=0x7313).waitdone()
    assert cdw0 == 0x07030103

    nvme0n1 = Namespace(nvme0, 1, 0x10000)
    nvme0n1.format(512)

    l = []
    for i in range(3):
        a = nvme0n1.ioworker(io_size=8,
                             read_percentage=100,
                             region_end=0x10000,
                             qprio=i+1,
                             time=10)
        l.append(a)
```

ioworker: Parameters

Input Parameters:

- lba_start, lba_step, lba_align
- lba_random
- region_start, region_end
- distribution

- io_size (int, range, list, dict)
- read_percentage, io_percentage
- time, io_count
- qdepth, qprio
- pvalue, ptype
- iops
- ...

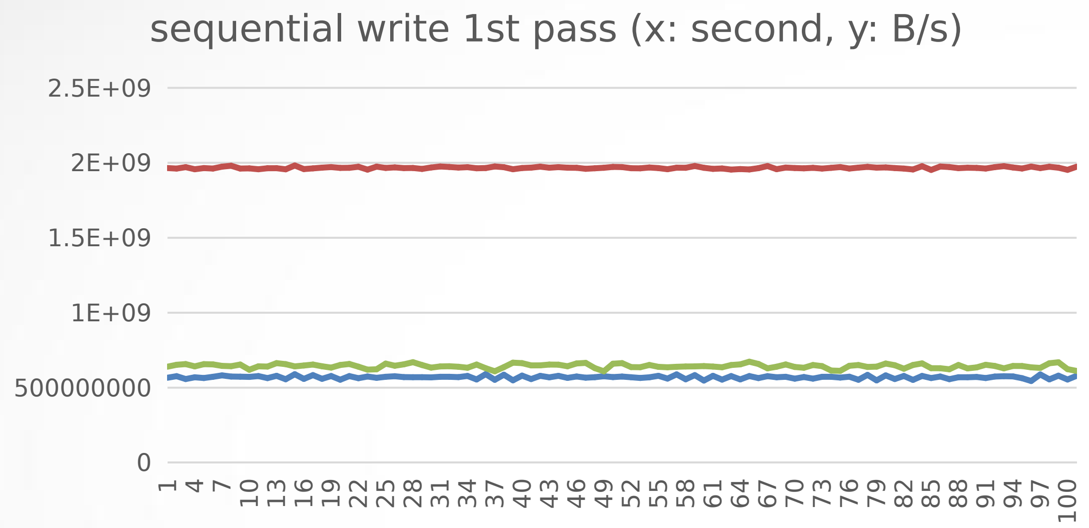
Output Parameters:

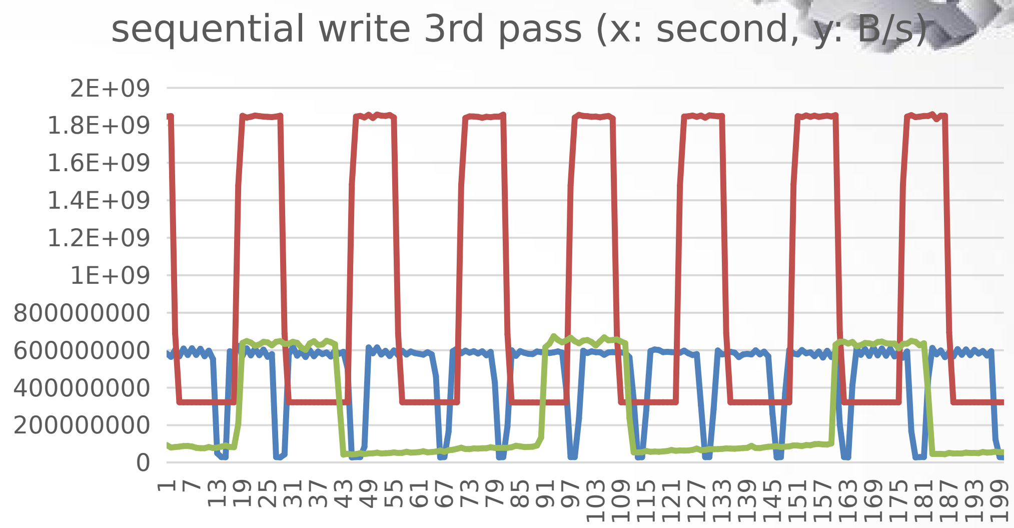
- io_count_read
- io_count_write
- mseconds
- latency_max_us
- latency_average_us
- error
- cpu_usage

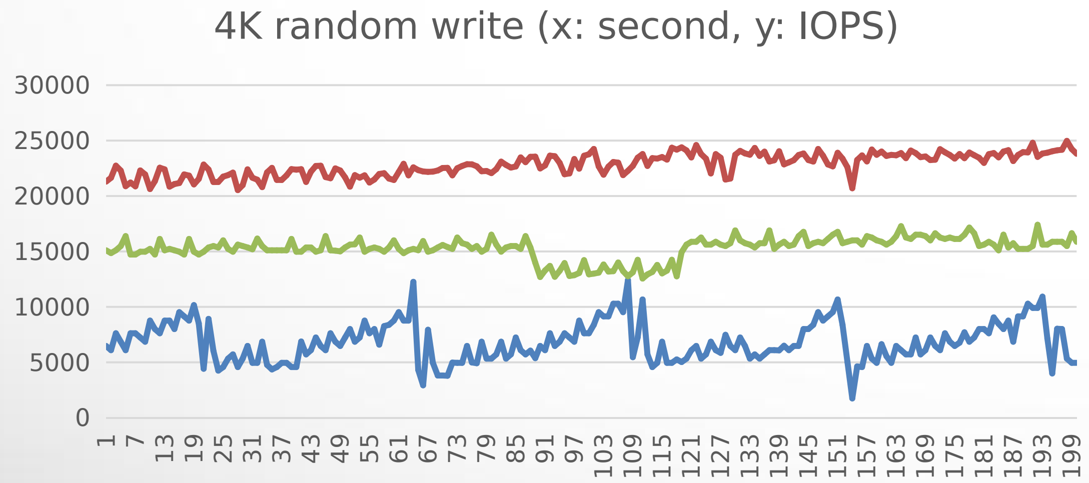
- output_io_per_second (optional)
- output_percentile_latency (optional)
- output_cmdlog_list (optional)
- ...

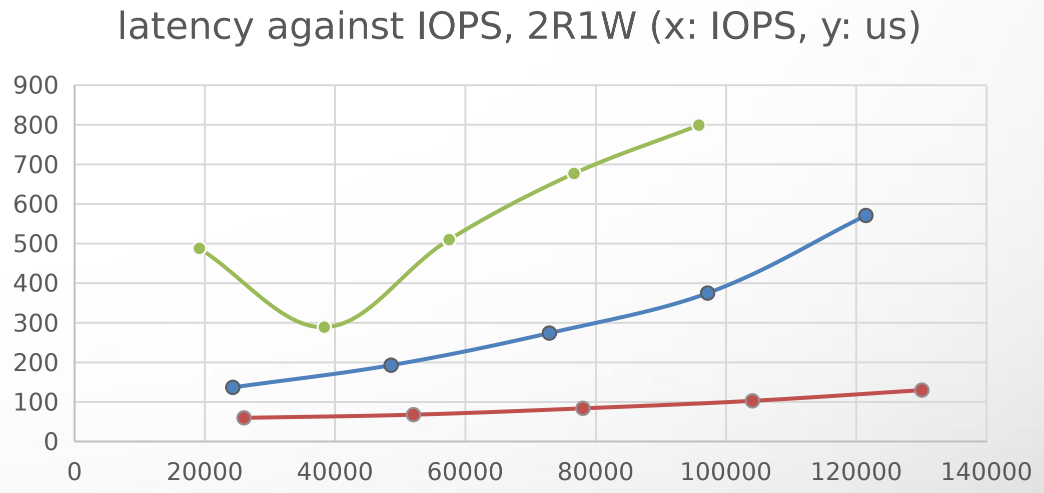
API document: https://pynvme.readthedocs.io/api.html

ioworker: Gallery









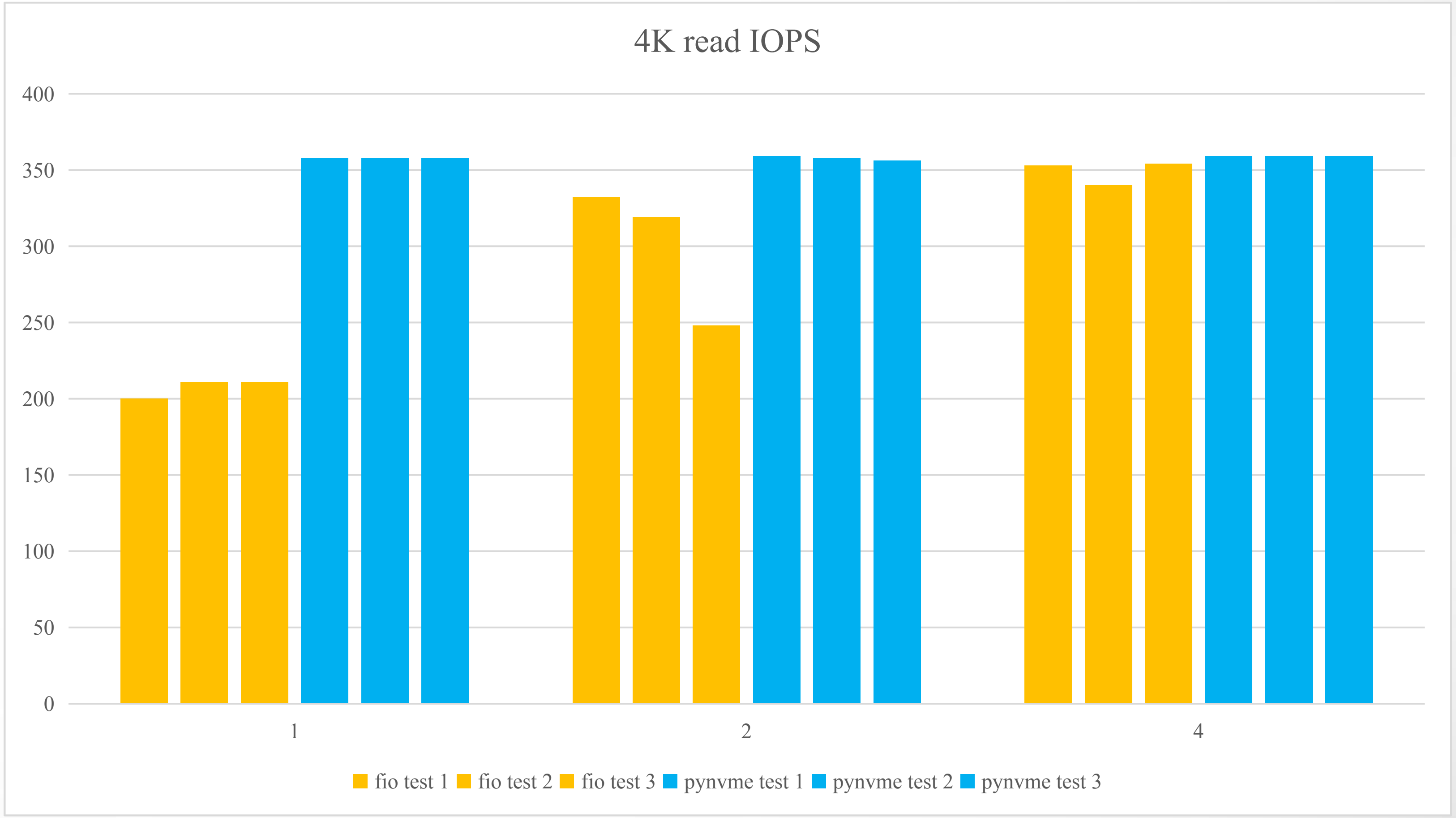
ioworker: Performance

fio (unit: K IOPS)

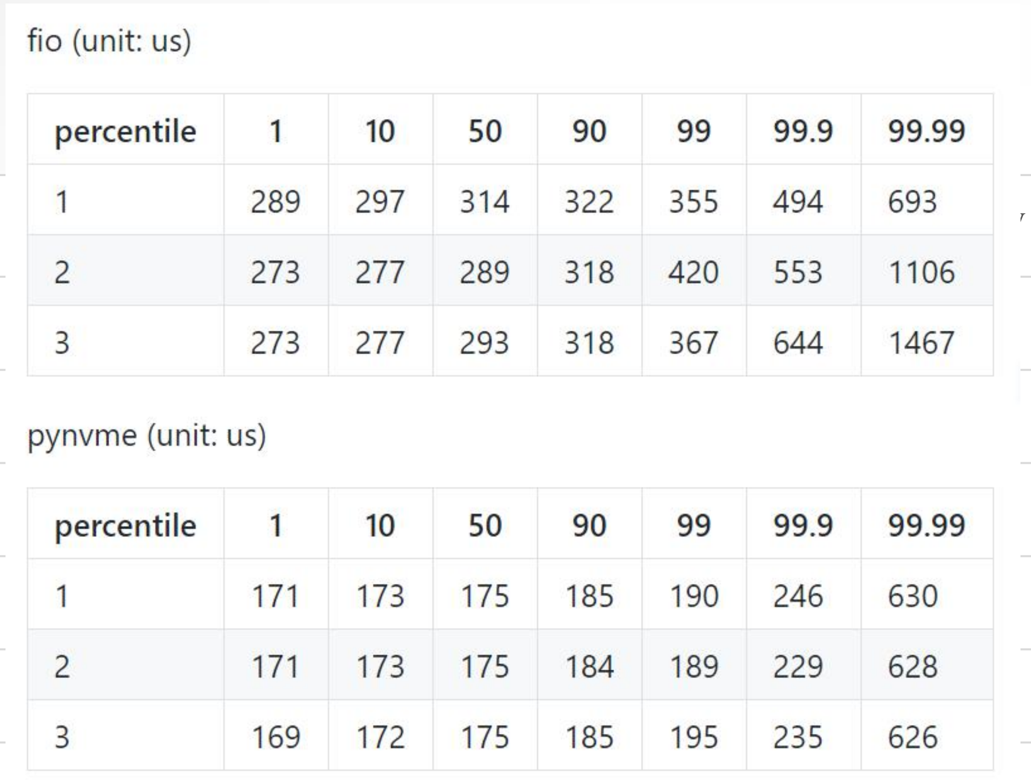
Q count	1	2	4
test 1	200	332	353
test 2	211	319	340
test 3	211	248	354

pynvme (unit: K IOPS)

Q count	1	2	4
test 1	358	359	359
test 2	358	358	359
test 3	358	356	359



ioworker: Latency

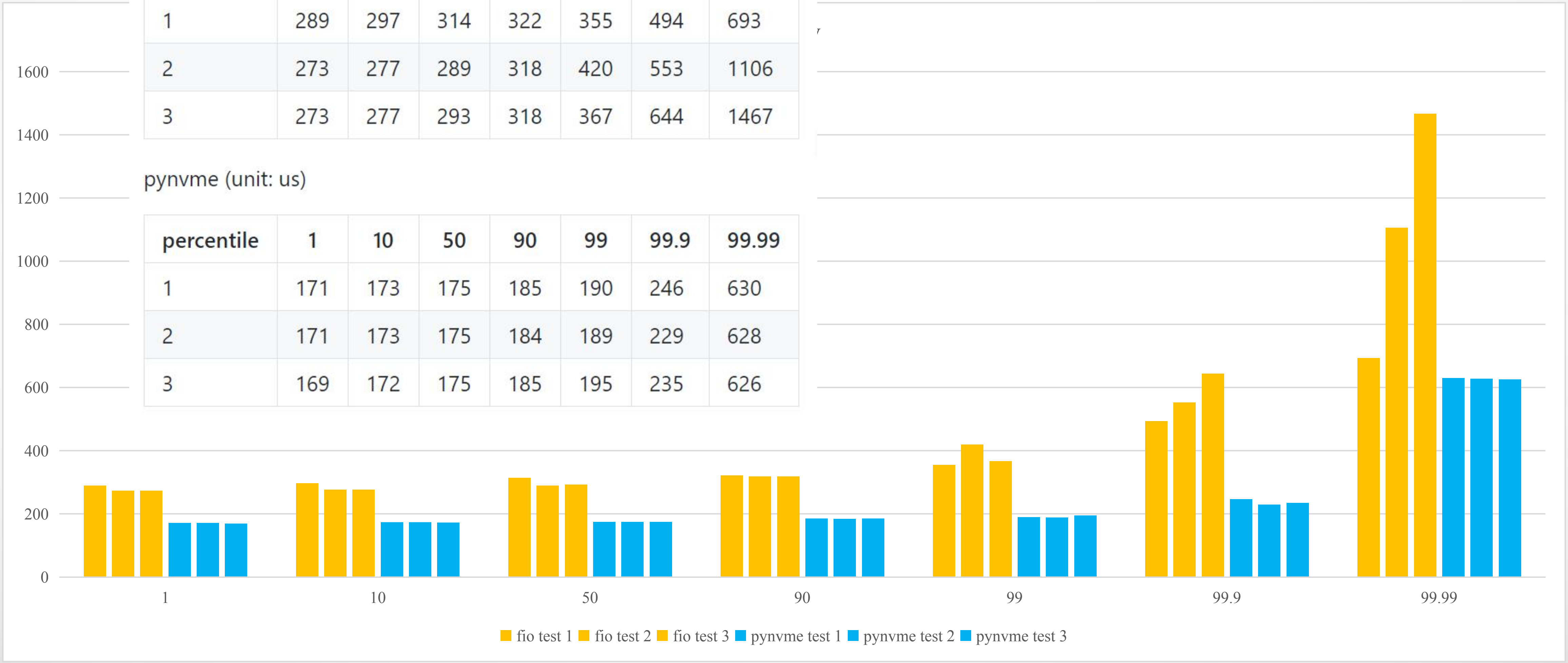


fio (unit: us)

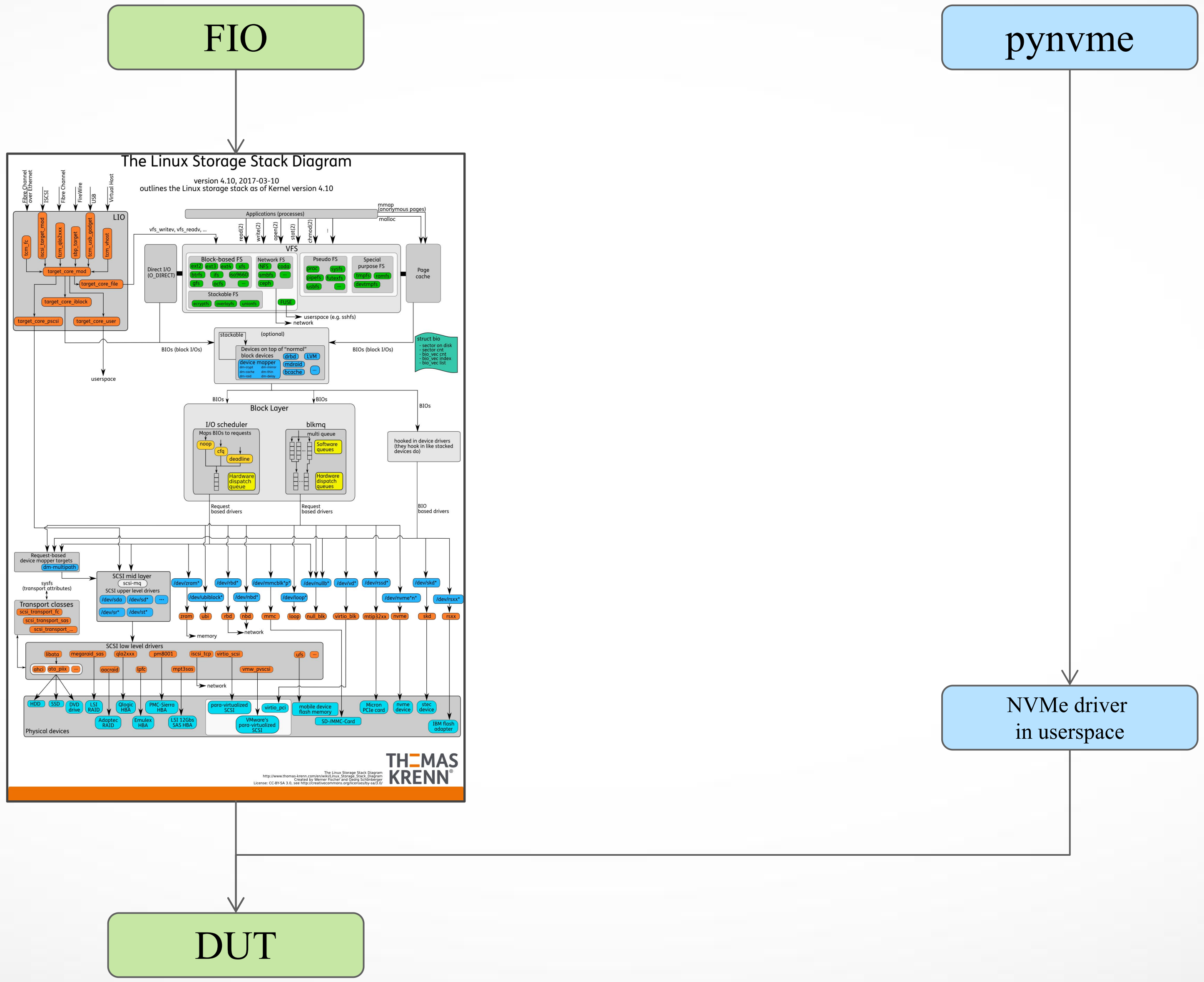
percentile	1	10	50	90	99	99.9	99.99
1	289	297	314	322	355	494	693
2	273	277	289	318	420	553	1106
3	273	277	293	318	367	644	1467

pynvme (unit: us)

percentile	1	10	50	90	99	99.9	99.99
1	171	173	175	185	190	246	630
2	171	173	175	184	189	229	628
3	169	172	175	185	195	235	626

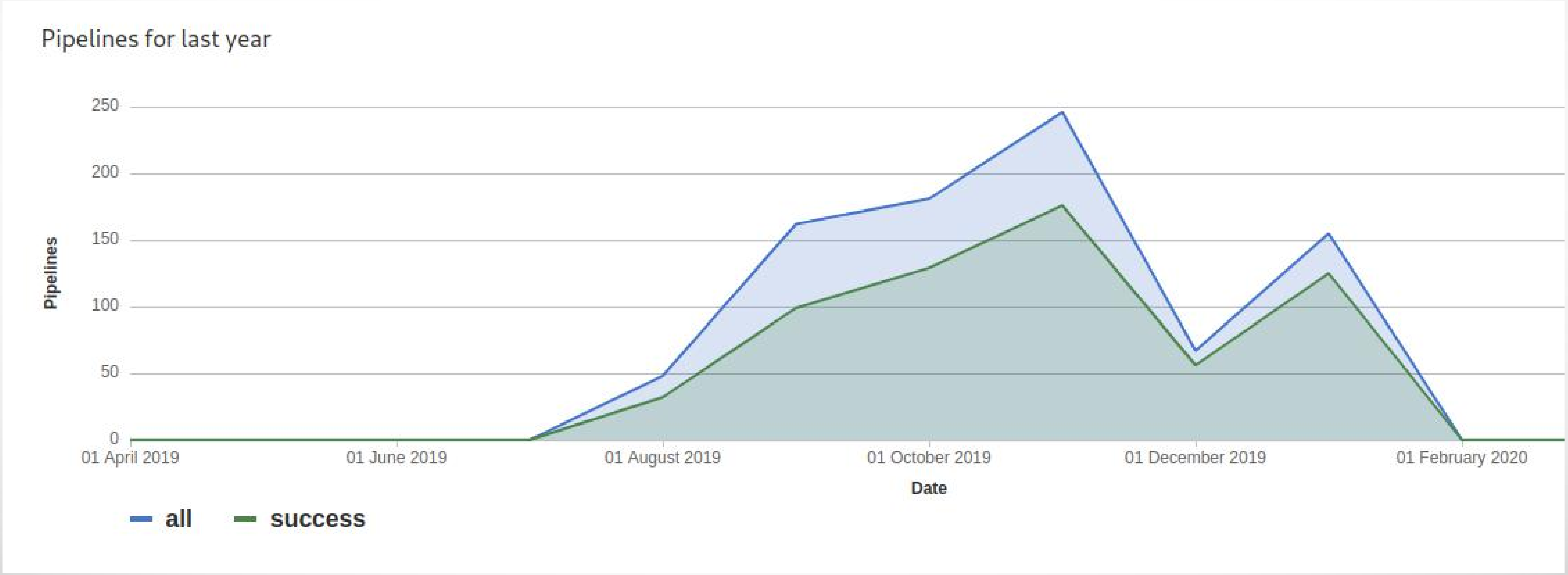


ioworker: Design



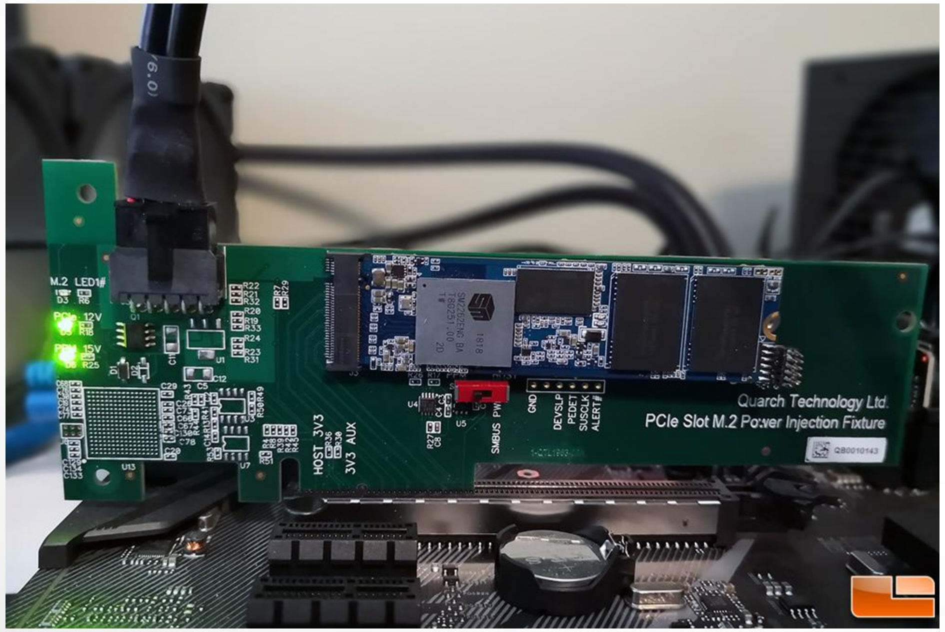
Quality: test of the test infrastructure

- CI in gitlab.com
- ~200 test cases

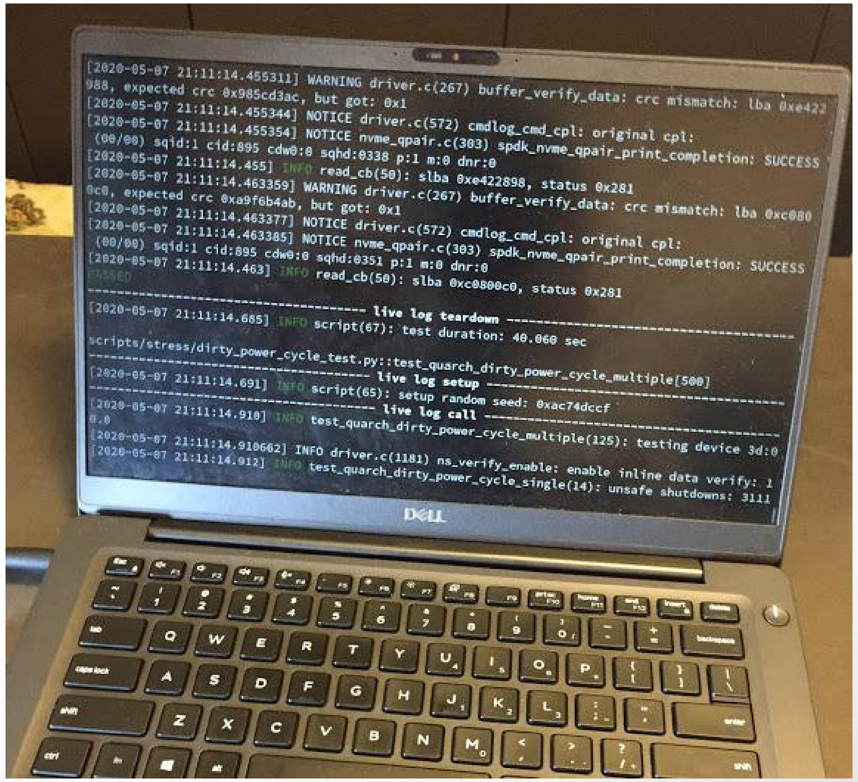


vPower to ON and OFF

- Usually we need a special hardware box to control the power of SSD

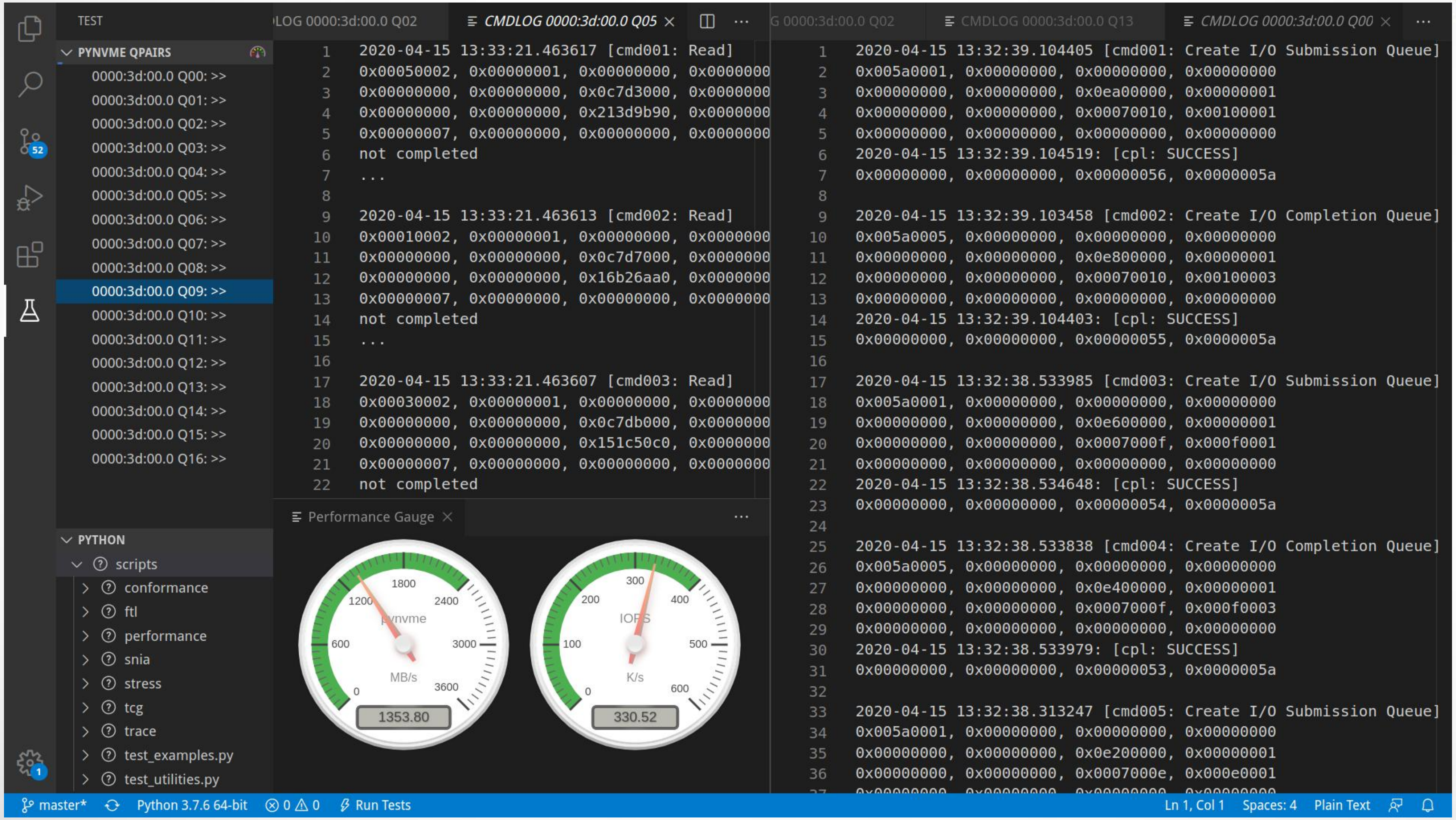


- vPower contorls the power of SSD:
  - power off: enter S3/sleep mode
  - power on: wake up by RTC



```
[2020-05-07 21:11:14.455311] WARNING driver.c(267) buffer_verify_data: crc mismatch: lba 0xe422
988, expected crc 0x985cd3ac, but got: 0x1
[2020-05-07 21:11:14.455344] NOTICE driver.c(572) cmdlog_cmd_cpl: original cpl:
 (00/00) sqid:1 cid:895 cdw0:0 sqhd:0338 p:1 m:0 dnr:0
[2020-05-07 21:11:14.455354] NOTICE nvme_qpair.c(303) spdk_nvme_qpair_print_completion: SUCCESS
[2020-05-07 21:11:14.455] INFO read_cb(50): slba 0xe422898, status 0x281
[2020-05-07 21:11:14.463359] WARNING driver.c(267) buffer_verify_data: crc mismatch: lba 0xc080
0c0, expected crc 0xa9f6b4ab, but got: 0x1
[2020-05-07 21:11:14.463377] NOTICE driver.c(572) cmdlog_cmd_cpl: original cpl:
[2020-05-07 21:11:14.463385] NOTICE nvme_qpair.c(303) spdk_nvme_qpair_print_completion: SUCCESS
 (00/00) sqid:1 cid:895 cdw0:0 sqhd:0351 p:1 m:0 dnr:0
[2020-05-07 21:11:14.463] INFO read_cb(50): slba 0xc0800c0, status 0x281
PASSED
------------------------------- live log teardown -------------------------------
[2020-05-07 21:11:14.685] INFO script(67): test duration: 40.060 sec

scripts/stress/dirty_power_cycle_test.py::test_quarch_dirty_power_cycle_multiple[500]
------------------------------- live log setup -------------------------------
[2020-05-07 21:11:14.691] INFO script(65): setup random seed: 0xac74dccf
------------------------------- live log call -------------------------------
[2020-05-07 21:11:14.910] INFO test_quarch_dirty_power_cycle_multiple(125): testing device 3d:0
0.0
[2020-05-07 21:11:14.910662] INFO driver.c(1181) ns_verify_enable: enable inline data verify: 1
[2020-05-07 21:11:14.912] INFO test_quarch_dirty_power_cycle_single(14): unsafe shutdowns: 3111
```

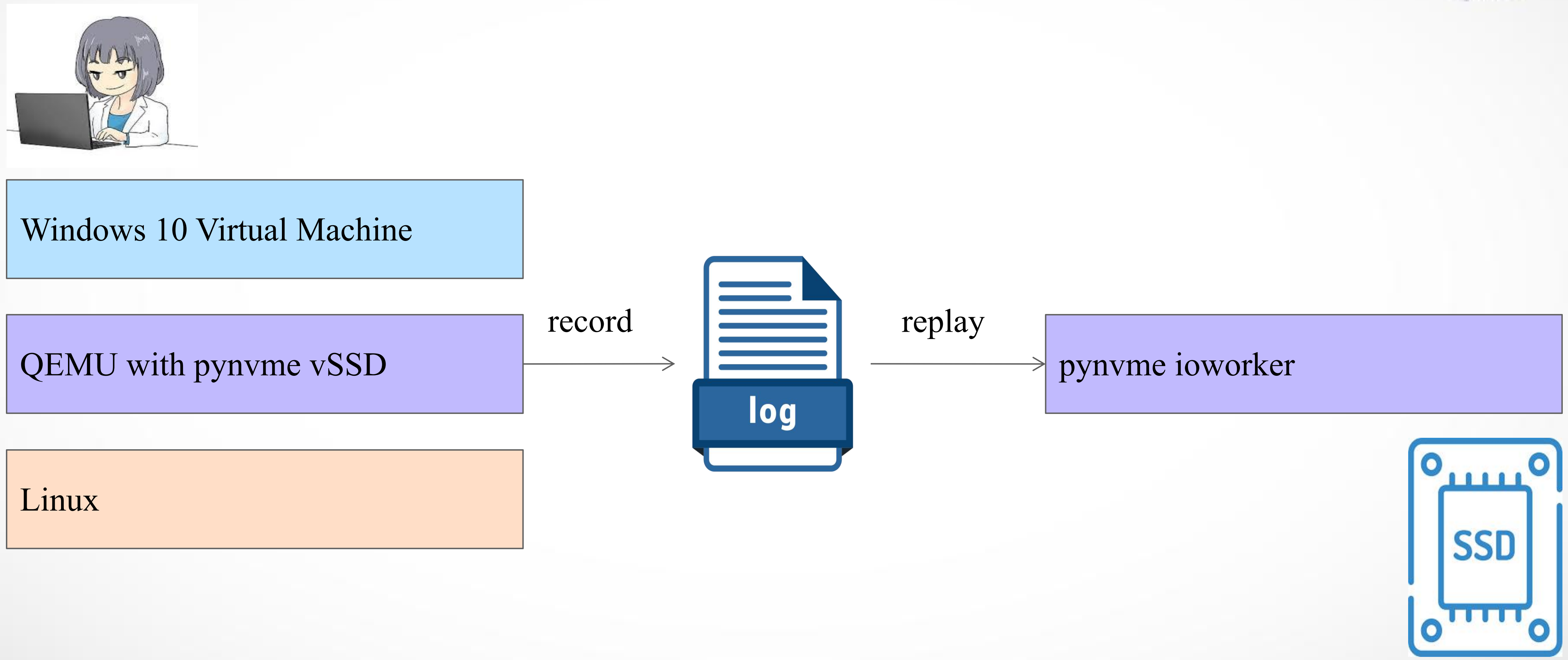
vAnalyzer



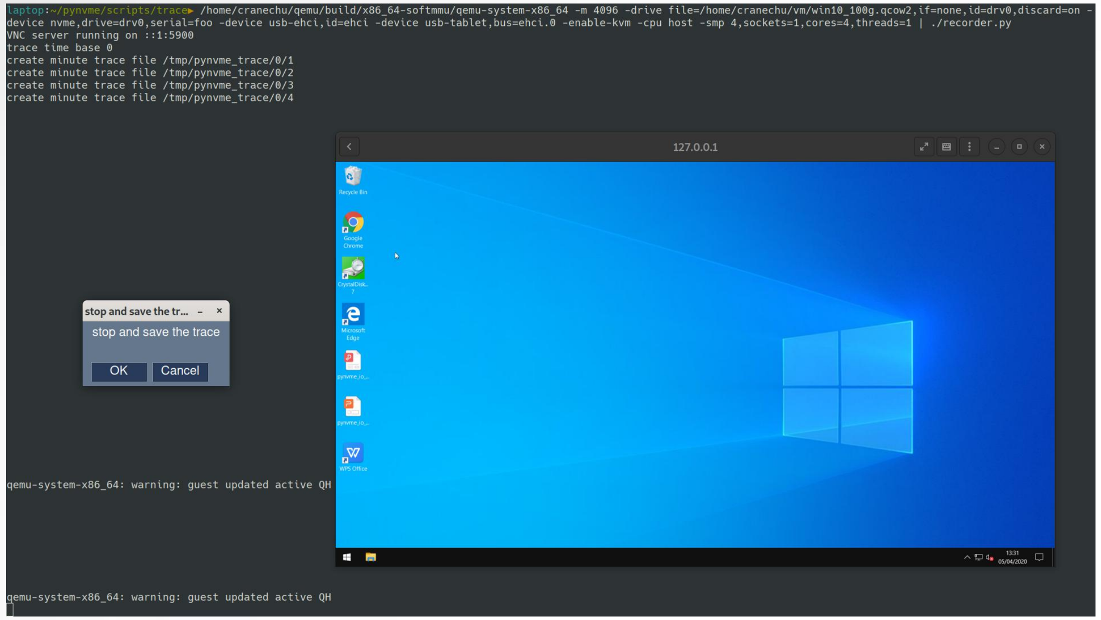
```
2020-04-15 13:33:21.463617 [cmd001: Read]
0x00050002, 0x00000001, 0x00000000, 0x0000000
0x00000000, 0x00000000, 0x0c7d3000, 0x0000000
0x00000000, 0x00000000, 0x213d9b90, 0x0000000
0x00000007, 0x00000000, 0x00000000, 0x0000000
not completed
...

2020-04-15 13:33:21.463613 [cmd002: Read]
0x00010002, 0x00000001, 0x00000000, 0x0000000
0x00000000, 0x00000000, 0x0c7d7000, 0x0000000
0x00000000, 0x00000000, 0x16b26aa0, 0x0000000
0x00000007, 0x00000000, 0x00000000, 0x0000000
not completed
...

2020-04-15 13:33:21.463607 [cmd003: Read]
0x00030002, 0x00000001, 0x00000000, 0x0000000
0x00000000, 0x00000000, 0x0c7db000, 0x0000000
0x00000000, 0x00000000, 0x151c50c0, 0x0000000
0x00000007, 0x00000000, 0x00000000, 0x0000000
not completed
```

```
2020-04-15 13:32:39.104405 [cmd001: Create I/O Submission Queue]
0x005a0001, 0x00000000, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x0ea00000, 0x00000001
0x00000000, 0x00000000, 0x00070010, 0x00100001
0x00000000, 0x00000000, 0x00000000, 0x00000000
2020-04-15 13:32:39.104519: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x00000056, 0x0000005a

2020-04-15 13:32:39.103458 [cmd002: Create I/O Completion Queue]
0x005a0005, 0x00000000, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x0e800000, 0x00000001
0x00000000, 0x00000000, 0x00070010, 0x00100003
0x00000000, 0x00000000, 0x00000000, 0x00000000
2020-04-15 13:32:39.104403: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x00000055, 0x0000005a

2020-04-15 13:32:38.533985 [cmd003: Create I/O Submission Queue]
0x005a0001, 0x00000000, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x0e600000, 0x00000001
0x00000000, 0x00000000, 0x0007000f, 0x000f0001
0x00000000, 0x00000000, 0x00000000, 0x00000000
2020-04-15 13:32:38.534648: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x00000054, 0x0000005a

2020-04-15 13:32:38.533838 [cmd004: Create I/O Completion Queue]
0x005a0005, 0x00000000, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x0e400000, 0x00000001
0x00000000, 0x00000000, 0x0007000f, 0x000f0003
0x00000000, 0x00000000, 0x00000000, 0x00000000
2020-04-15 13:32:38.533979: [cpl: SUCCESS]
0x00000000, 0x00000000, 0x00000053, 0x0000005a

2020-04-15 13:32:38.313247 [cmd005: Create I/O Submission Queue]
0x005a0001, 0x00000000, 0x00000000, 0x00000000
0x00000000, 0x00000000, 0x0e200000, 0x00000001
0x00000000, 0x00000000, 0x0007000e, 0x000e0001
0x00000000, 0x00000000, 0x00000000, 0x00000000
```

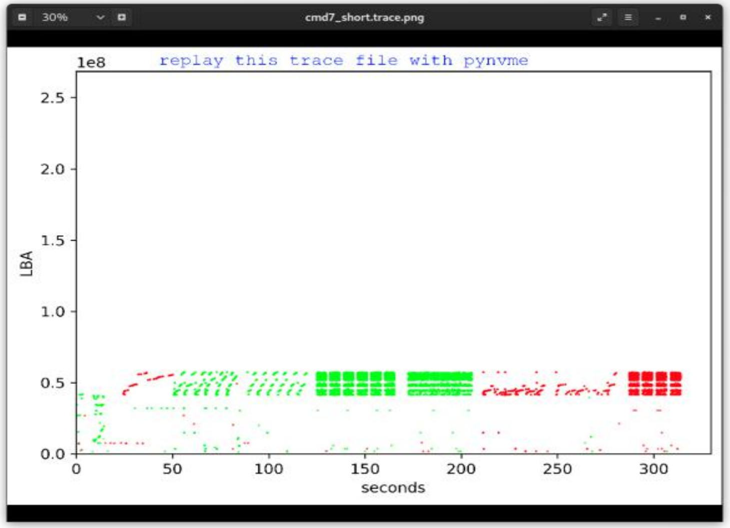
vTracer: IO Recorder and Replayer

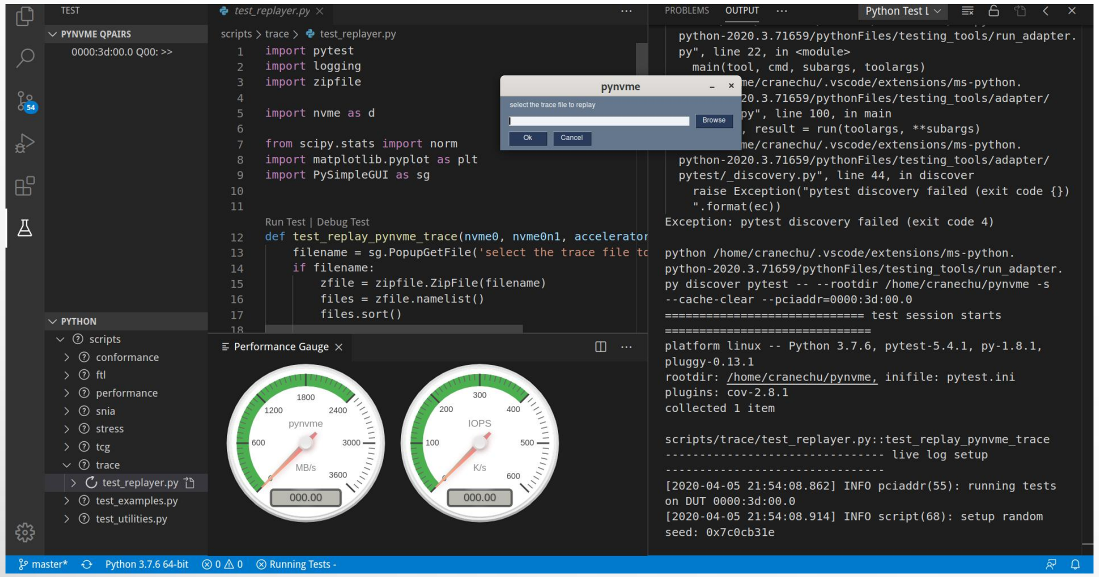


vTracer: Demo



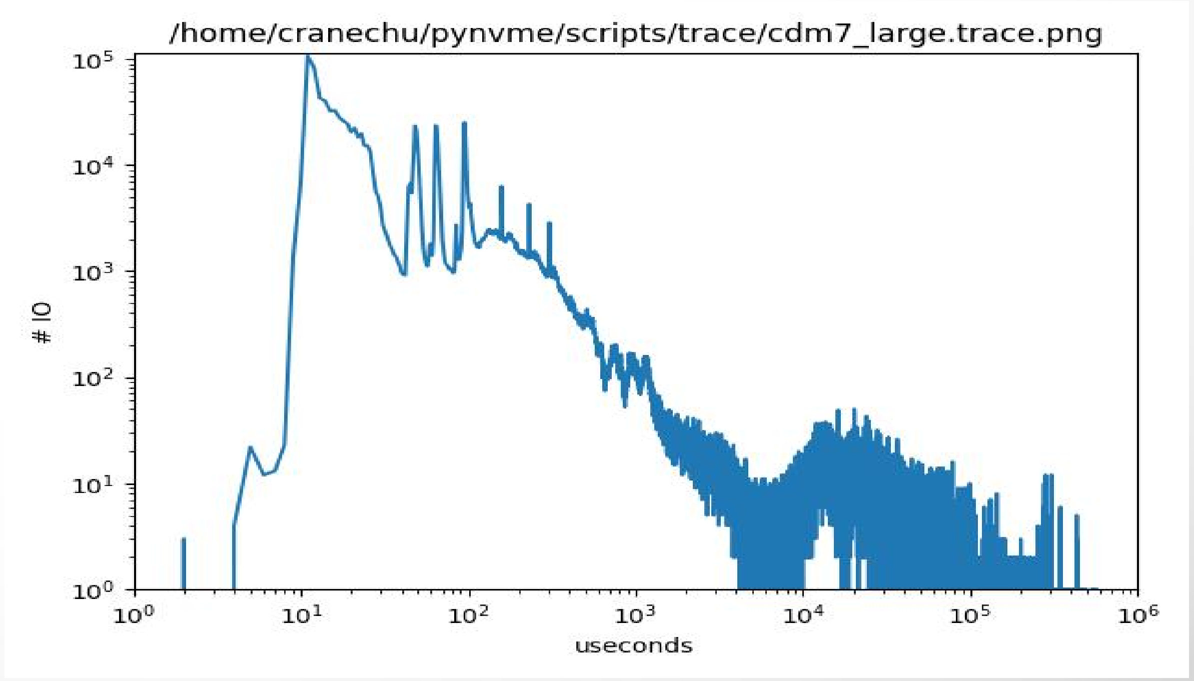
```
laptop:~/pynvme/scripts/trace▸ /home/cranechu/qemu/build/x86_64-softmmu/qemu-system-x86_64 -m 4096 -drive file=/home/cranechu/vm/win10_100g.qcow2,if=none,id=drv0,discard=on -
device nvme,drive=drv0,serial=foo -device usb-ehci,id=ehci -device usb-tablet,bus=ehci.0 -enable-kvm -cpu host -smp 4,sockets=1,cores=4,threads=1 | ./recorder.py
VNC server running on ::1:5900
trace time base 0
create minute trace file /tmp/pynvme_trace/0/1
create minute trace file /tmp/pynvme_trace/0/2
create minute trace file /tmp/pynvme_trace/0/3
create minute trace file /tmp/pynvme_trace/0/4

qemu-system-x86_64: warning: guest updated active QH

qemu-system-x86_64: warning: guest updated active QH
```



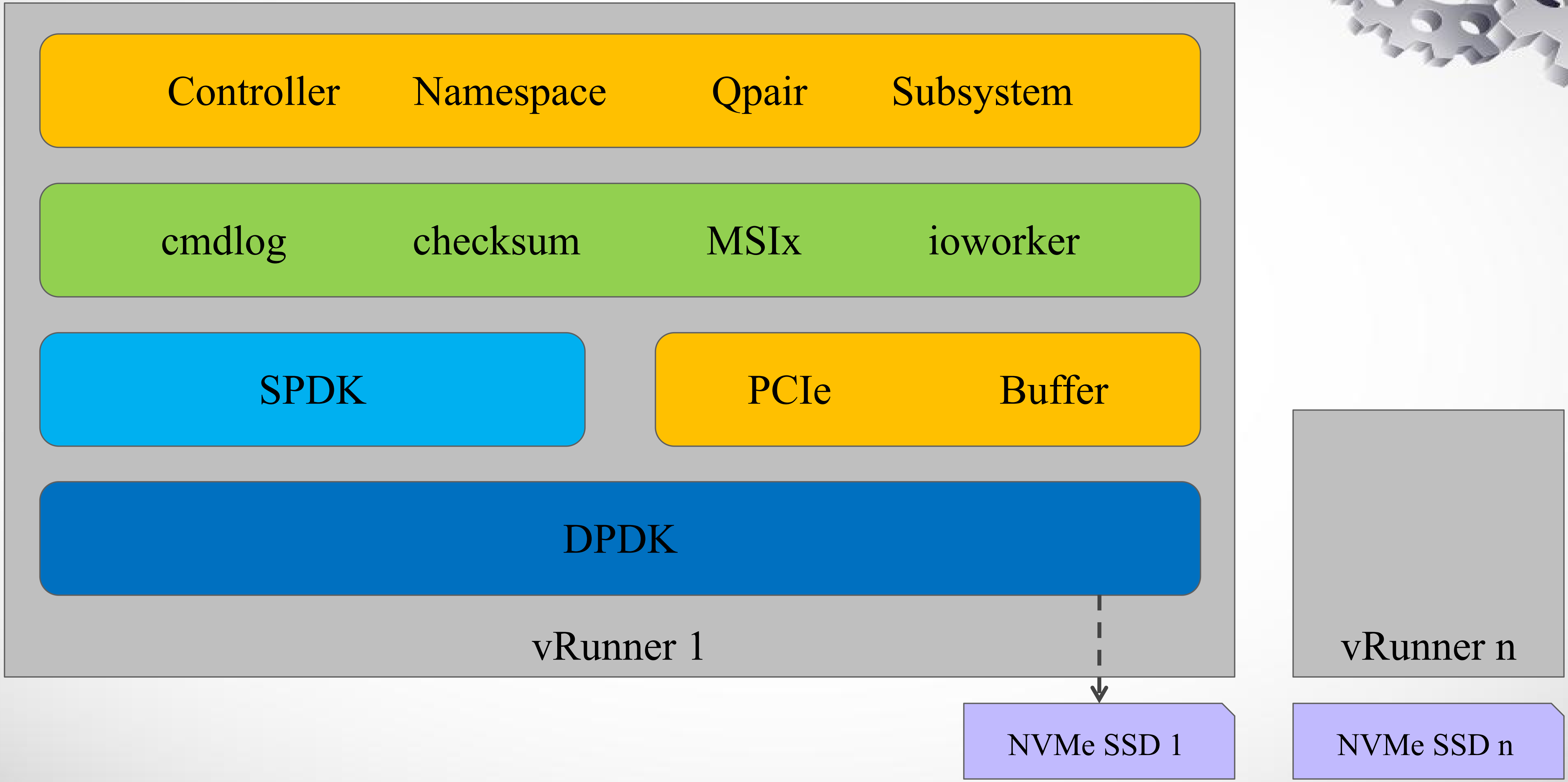


```python
import pytest
import logging
import zipfile

import nvme as d

from scipy.stats import norm
import matplotlib.pyplot as plt
import PySimpleGUI as sg


def test_replay_pynvme_trace(nvme0, nvme0n1, accelerator
    filename = sg.PopupGetFile('select the trace file to
    if filename:
        zfile = zipfile.ZipFile(filename)
        files = zfile.namelist()
        files.sort()
```

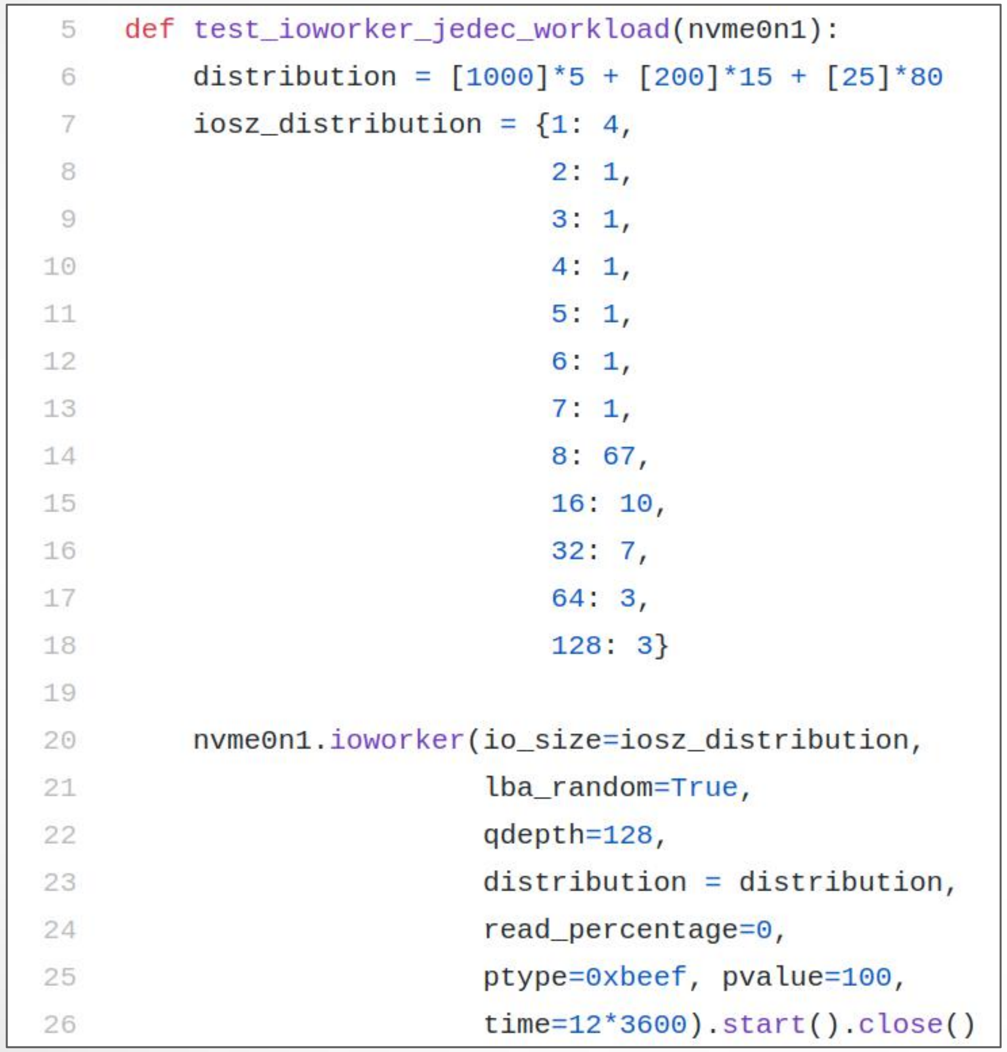
```
  python-2020.3.71659/pythonFiles/testing_tools/run_adapter.
  py", line 22, in <module>
    main(tool, cmd, subargs, toolargs)
                me/cranechu/.vscode/extensions/ms-python.
                20.3.71659/pythonFiles/testing_tools/adapter/
                py", line 100, in main
                , result = run(toolargs, **subargs)
                me/cranechu/.vscode/extensions/ms-python.
  python-2020.3.71659/pythonFiles/testing_tools/adapter/
  pytest/_discovery.py", line 44, in discover
    raise Exception("pytest discovery failed (exit code {})
    ".format(ec))
Exception: pytest discovery failed (exit code 4)

python /home/cranechu/.vscode/extensions/ms-python.
python-2020.3.71659/pythonFiles/testing_tools/run_adapter.
py discover pytest -- --rootdir /home/cranechu/pynvme -s
--cache-clear --pciaddr=0000:3d:00.0
============================ test session starts
==============================
platform linux -- Python 3.7.6, pytest-5.4.1, py-1.8.1,
pluggy-0.13.1
rootdir: /home/cranechu/pynvme, inifile: pytest.ini
plugins: cov-2.8.1
collected 1 item

scripts/trace/test_replayer.py::test_replay_pynvme_trace
-------------------------------- live log setup
--------------------------------
[2020-04-05 21:54:08.862] INFO pciaddr(55): running tests
on DUT 0000:3d:00.0
[2020-04-05 21:54:08.914] INFO script(68): setup random
seed: 0x7c0cb31e
```



vRunner: Mass Deploy



Services

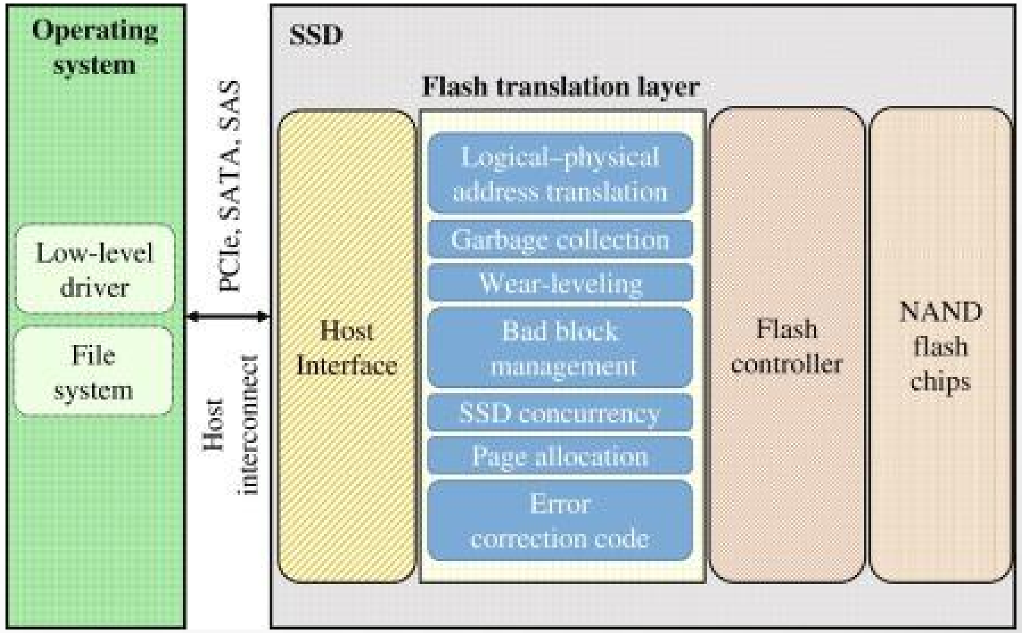


```python
def test_ioworker_jedec_workload(nvme0n1):
    distribution = [1000]*5 + [200]*15 + [25]*80
    iosz_distribution = {1: 4,
                         2: 1,
                         3: 1,
                         4: 1,
                         5: 1,
                         6: 1,
                         7: 1,
                         8: 67,
                         16: 10,
                         32: 7,
                         64: 3,
                         128: 3}

    nvme0n1.ioworker(io_size=iosz_distribution,
                     lba_random=True,
                     qdepth=128,
                     distribution = distribution,
                     read_percentage=0,
                     ptype=0xbeef, pvalue=100,
                     time=12*3600).start().close()
```

JEDEC®

SNIA®

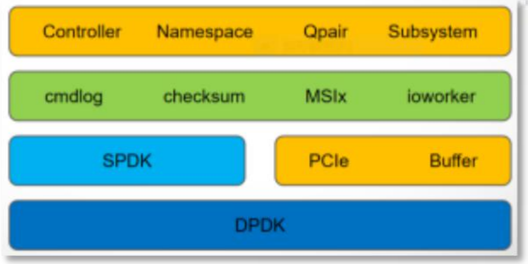
nvm EXPRESS



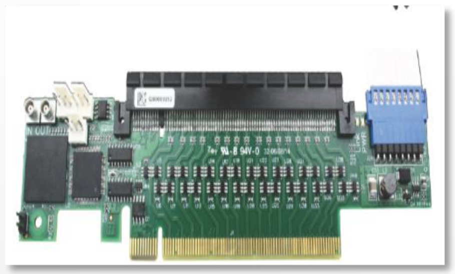
Values

Ecosystem

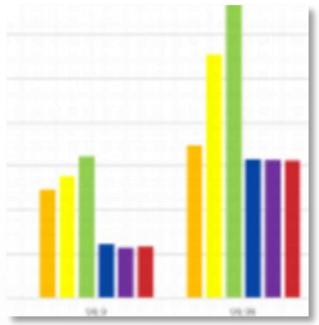
Scripts



Expendability



Hardware



Performance

Service

pynvme builds your own tests.

https://github.com/pynvme/pynvme

Thanks!