SPDK on LS2 PLATFORM

SPDK (Storage Performance Development Kit) is an optimized storage reference architecture. It is initiated and developed by Intel.

SPDK provides a set of tools and libraries for writing high performance, scalable, user-mode storage applications. It achieves high performance by moving all of the necessary drivers into userspace and operating in a polled mode, like DPDK.

Background

- Hard drive latency is dramatically dropping down: HDD(SAS/SATA) ~10ms → SSD (SATA) ~0.1ms → SSD (NVMe) ~0.075ms
- Bus width and command queue is increasing: SAS/SATA 6Gbps, 32 commands/queue → NVMe 24Gbps, 64k commands/queue
- Network bandwidth is increasing: 1Gbps → 10Gbps → 40Gbps → 100Gbps

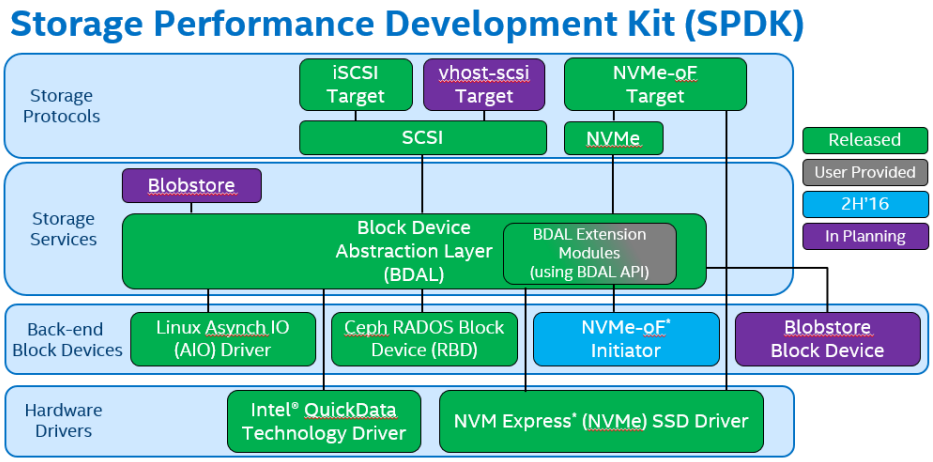
All these changes make software latency the major contributor to the whole latency stack in nowdays.

Architecture and subcomponents

Two key changes for SPDK to reduce latency caused by software stack:

- Poll mode driver: Submits the request for a read or write, and then goes off to do other work, checking back at some interval to see if the I/O has yet been completed. This avoids the latency and overhead of using interrupts and allows the application to improve I/O efficiency
- User space data process: Avoiding the kernel context switches and interrupts saves a significant amount of processing overhead, allowing more cycles to be spent doing the actual storing of the data.

Following is the software stack of SPDK (from Intel):



Subcomponents

NVMe Driver

lib/nvme

Provides direct, zero-copy data transfer to and from NVMe SSDs. It controls NVMe devices by directly mapping the PCI BAR into the local process and performing MMIO. I/O is submitted asynchronously via queue pairs.

NVMe over Fabrics Target

lib/nvmf

User space application that presents block devices over the network using RDMA. It requires an RDMA-capable NIC with its corresponding OFED software package installed to run.

iSCSI Target

lib/iscsi

Implementation of the established specification for block traffic over Ethernet. Current version uses the kernel TCP/IP stack by default.

Block Device Abstraction Layer

lib/bdev

This generic block device abstraction is the glue that connects the storage protocols to the various device drivers and block devices. Also provides flexible APIs for additional customer functionality (RAID, compression, dedup, and so on) in the block layer.

It defines:

- a driver module API for implementing bdev drivers
- an application API for enumerating and claiming SPDK block devices and performance operations
- bdev drivers for NVMe, malloc (ramdisk), Linux AIO and Ceph RBD

Blobstore

lib/blob

A persistent, power-fail safe block allocator designed to be used as the local storage system backing a higher level storage service, typically in lieu of a traditional filesystem.

This is a virtual device that VMs or databases could interact with**.**

BlobFS

lib/blobfs

Adds basic filesystem functionality like filenames on top of the blobstore.

vhost

lib/vhost

It extends SPDK to present virtio storage controllers to QEMU-based VMs and process I/O submitted to devices attached to those controllers

Event framework

lib/event

A framework for writing asynchronous, polled-mode, shared-nothing server applications.

The event framework is intended to be optional; most other SPDK components are designed to be integrated into an application without specifically depending on the SPDK event library. The framework defines several concepts - reactors, events, and pollers.

Hardware dependent parts

SPDK integrates DPDK as default running environment. So any hardware limitations/dependencies in DPDK also exists for SPDK. That implies the following area is what we will focus on enablment SPDK on AArch64 platform.

lib/ioat, /lib/copy

IOAT is a copy offload engine built into the Intel® Xeon® processor which used to greater utilization for small-size I/Os or NTB.

lib/nvmf

Depends on RDMA hardware.

lib/util/{crc*}

Depends on CPU's HW CRC features.

lib/env_dpdk

User space address range (47/48bits)

include/linux/virtio_pci.h

Page size of platform

Build and Test

General guides can be found here and here.

SPDK build/deployment is tested on LS2088.

Enviroment Setup

SW

- OS: Ubuntu 18.04.2 LTS
- SPDK: 43727fb7e5c@master branch
- DPDK: 18.11

HW

- LS2088 machine
- INTEL SSDPED1D280GA NVMe SSD card with firmware version of E2010325

Build

DPDK

```
# git clone git://dpdk.org/dpdk
# export RTE_TARGET=arm64-dpaa2-linuxapp-gcc
# export RTE_SDK=/code/dpdk
# make T=arm64-dpaa-linuxapp-gcc CONFIG_RTE_KNI_KMOD=n
CONFIG_RTE_LIBRTE_PPFE_PMD=n  CONFIG_RTE_EAL_IGB_UIO=n install -j 4
```

SPDK

```
# git clone https://github.com/spdk/spdk
# cd spdk
# sudo ./scripts/pkgdep.sh
#./configure –with-dpdk=/code/dpdk/arm64-dpaa2-linuxapp-gcc
# make -j8
```

Deploy

check NVMe status

```
# sudo lspci -vn | sed -n '/NVM Express/,/^$/p'
```

You should see lines like

```
root@localhost:~/spdk# sudo lspci -vn | sed -n '/NVM Express/,/^$/p'
0000:01:00.0 0108: 8086:2700 (prog-if 02 [NVM Express])
        Subsystem: 8086:3900
        Flags: bus master, fast devsel, latency 0, IRQ 365, NUMA node 0
        Memory at 3046010000 (64-bit, non-prefetchable) [size=16K]
        Expansion ROM at 3046000000 [disabled] [size=64K]
        Capabilities: [40] Power Management version 3
        Capabilities: [50] MSI-X: Enable- Count=32 Masked-
        Capabilities: [60] Express Endpoint, MSI 00
        Capabilities: [100] Advanced Error Reporting
        Capabilities: [150] Virtual Channel
        Capabilities: [180] Power Budgeting <?>
        Capabilities: [190] Alternative Routing-ID Interpretation (ARI)
        Capabilities: [270] Device Serial Number 55-cd-2e-41-4e-34-a9-2a
        Capabilities: [2a0] #19
        Kernel driver in use: uio_pci_generic
        Kernel modules: nvme
```

Deploy SPDK

UIO

```
# modprobe uio
# modprobe uio_pci_generic

# echo -n "8086 2700 8086 3900" > /sys/bus/pci/drivers/uio_pci_generic/new_id
# echo -n "0000:01:00.0" > /sys/bus/pci/drivers/nvme/unbind
echo -n "0000:01:00.0" > /sys/bus/pci/drivers/uio_pci_generic/bind
```

VFIO

```
# modprobe vfio-pci
# cd <SPDK_ROOT_DIR>
# ./scripts/setup.sh
```

Test

```
# sudo ./examples/nvme/identify/identify
```

This app should give you the detail disk info of attached NVMe storage.

```
# sudo ./examples/nvme/perf/perf -q 128 -s 4096 -w write -t 60 -c 0xFF -o 2048 -r 'trtype:PCIe
traddr:0000:01:00.0'
```

This will give SPDK performance data.
With prior described HW/SW settings, following data are achieved (performance in MBps):

	512B	2K	4K	8K
Rd	286	1082	1120	1461

Wr	117	458	1445	1137

Benchmark

FIO

Build FIO with SPDK

```
# git clone https://github.com/axboe/fio --branch fio-3.3
# cd fio
# make
```

Build SPDK with FIO plugin support

```
# cd spdk
# ./configure --with-fio=<path-to-fio-src> --enable-debug
# make DPDK_CONFIG=arm64-armv8a-linuxapp-gcc
```

Run FIO

```
# cd fio
# LD_PRELOAD=../spdk/examples/nvme/fio_plugin/fio_plugin ./fio --name=nvme --numjobs=1 --
filename="trtype=PCIe traddr=0000.01.00.0 ns=1" --bs=4K --iodepth=1 --
ioengine=../spdk/examples/nvme/fio_plugin/fio_plugin --direct=1 --sync=0 --norandommap --
group_reporting --size=10% --runtime=3 -rwmixwrite=30 --thread=1 --rw=r
```