100G SPDK NVMe over Fabrics

T6 iWARP RDMA Bandwidth, IOPS and Latency Performance

Executive Summary

Chelsio's T6 iWARP RDMA provides a low latency, high throughput, plug-and-play Ethernet solution for connecting high performance NVMe SSDs over a scalable, congestion controlled and traffic managed fabric, with no special configuration needed.

SPDK, designed to extract maximum performance by moving all the necessary drivers to user space, polling hardware for completions instead of relying on interrupts and avoiding all locks in the I/O path provides the benefits of high and scalable performance and low latency for user mode storage applications. SPDK together with the NVMe over Fabrics specification extends the benefits of NVMe to large fabrics which are beyond the reach and scalability of PCIe. NVMe enables deployments with hundreds or thousands of SSDs using a network interconnect, such as RDMA over Ethernet.

This paper presents the performance results of the Chelsio T6 SPDK NVMe-oF over 100GbE iWARP fabric solution. Chelsio's T6 adapter delivers line-rate 99 Gbps throughput and 2.5 Million IOPS (at the 4K I/O size). In addition, with only 4.7µs delta latency between remote and local storage, Chelsio's solution proves to be best-in-breed providing the next generation, scalable storage network over standard and cost effective ethernet infrastructure with an efficient processing path.

Test Results

The following graph presents SPDK NVMe-oF READ, WRITE IOPS and throughput results of the Chelsio iWARP solution using RAM device. The results are collected using the **fio** tool with I/O size varying from 4 to 256 KBytes with an access pattern of random READs and WRITEs.

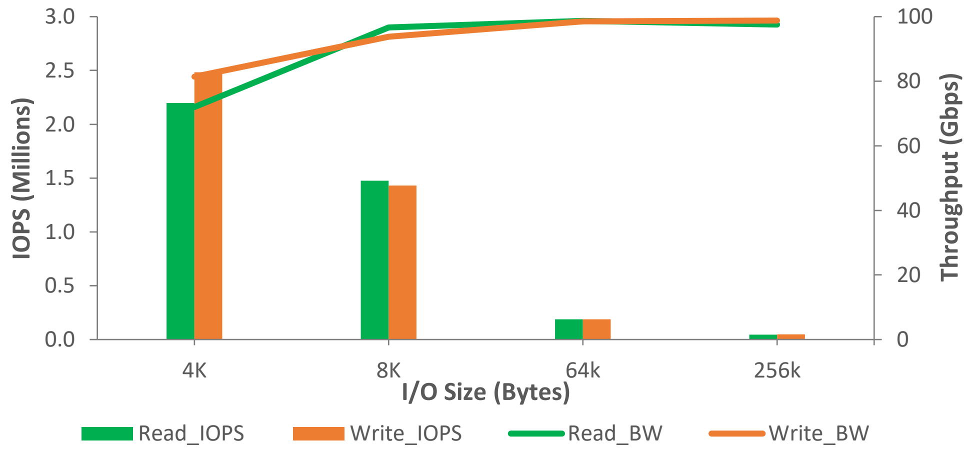


Figure 1 – SPDK NVMe-oF IOPS and Throughput vs. I/O size

As evident from the graphs above, the T6 solution delivers line-rate 99 Gbps throughput for READ and WRITE. WRITE IOPs reaches 2.5 Million and READ IOP reaches 2.2 Million at the 4K I/O size.

The below graph and table present the latency numbers at 4K I/O size varying the number of connections.

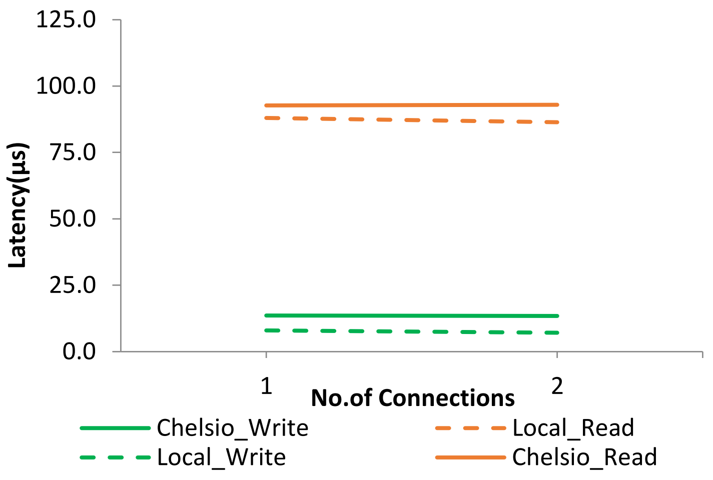
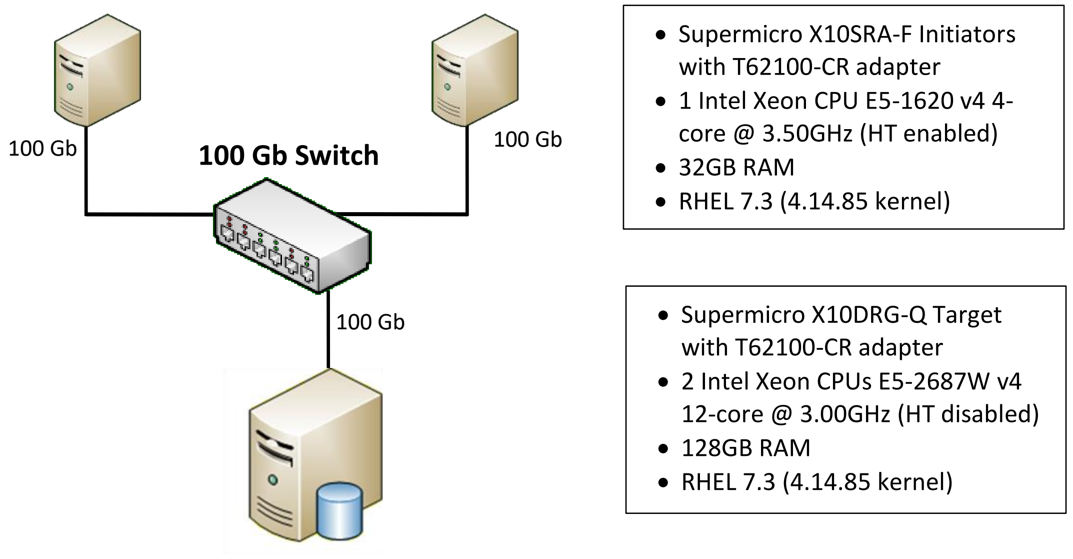


Figure 2 - Latency vs. #. connections

	READ			WRITE		
	Local	**Remote**	**Delta**	**Local**	**Remote**	**Delta**
1 connection	88.00	92.72	4.72	7.99	13.60	5.62

Observe that remote versus local NVMe device access adds only 4.7 and 5.6 µs latency for READ and WRITE operations.

The Demonstration



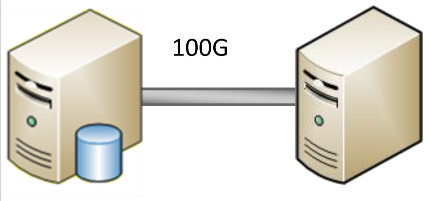
- Supermicro X10SRA-F Initiators with T62100-CR adapter
- 1 Intel Xeon CPU E5-1620 v4 4-core @ 3.50GHz (HT enabled)
- 32GB RAM
- RHEL 7.3 (4.14.85 kernel)

- Supermicro X10DRG-Q Target with T62100-CR adapter
- 2 Intel Xeon CPUs E5-2687W v4 12-core @ 3.00GHz (HT disabled)
- 128GB RAM
- RHEL 7.3 (4.14.85 kernel)

Figure 3 – Bandwidth/IOPs Test setup

The BW/IOPS setup consists of an NVMe target machine connected to two initiator machines through a 100GbE switch using a single port on each system. An MTU of 9000B is used on the ports under test. The latest Chelsio Unified Wire driver is installed on each machine.

- Supermicro X10DRG-Q Target with T62100-CR
- 2 Intel Xeon CPUs E5-2687W v4 12-core @ 3.00GHz (HT disabled)
- 128GB RAM
- 1 INTEL 1.6 TB PCIe NVME SSD



- Supermicro X10SRA-F Initiator with T62100-CR
- 1 Intel Xeon CPU E5-1620 v4 4-core @ 3.50GHz (HT disabled)
- 32GB RAM

Figure 4 – Latency Test setup

The Latency test setup consists of an NVMe target machine connected to a single initiator back to back using single port on each system. MTU of 9000B is used.

Storage configuration

For the BW/IOPS test, the SPDK NVMe-oF Target is configured with 4 LUNs (RAM device), each 512 MB in size. Each initiator connects to two target LUNs using four connections per target.

For the latency test, the target is configured with 1 Intel 1.6TB NVMe SSD. One Initiator connects to the target using one connection.

Setup Configuration

Target/Initiator Configuration

i. In the System BIOS: Disable virtualization, c-state technology, VT-d, Intel I/O AT, Hyperthreading and SR-IOV.

ii. Compile and install 4.14 kernel from Chelsio Unified Wire v3.10.0.0 package and reboot the machine into the newly installed kernel.

```
[root@host~]# make kernel_install
[root@host~]# reboot
```

iii. Install NVMe tools and drivers using:

```
[root@host~]# make CONF=NVME_PERFORMANCE install
```

iv. For latency test, add *idle=poll* to the kernel boot command line.

v. Set the below tuned-adm profile for BW/IOPS test:

```
[root@host~]# tuned-adm profile network-throughput
```

Set the below tuned-adm profile for Latency test:

```
[root@host~]# tuned-adm profile network-latency
```

vi. Load the Chelsio iWARP RDMA driver.

```
[root@host~]# modprobe iw_cxgb4
```

vii. Configure the Chelsio interfaces with IPv4 addresses, MTU 9000 and bring-up.

```
Target:
[root@host~]# ifconfig ens2f4 10.1.1.149/24 up mtu 9000
[root@host~]# ifconfig ens2f4:1 10.11.11.149/24 up mtu 9000

Initiator1:
[root@host~]# ifconfig ens1f4 10.1.1.41/24 up mtu 9000

Initiator2:
[root@host~]# ifconfig ens1f4 10.11.11.42/24 up mtu 9000
```

viii. Set the CPU affinity for BW/IOPs test.

```
[root@host~]# t4_perftune.sh -n -Q rdma
```

ix. Clone the SPDK, configure and compile with RDMA.

```
Note: Ensure CUnit and libuuid packages are installed.

[root@host~]# git clone https://github.com/spdk/spdk
[root@host~]# cd spdk
[root@host~]# git submodule update -init
 Change the below in CONFIG file.
               CONFIG_FIO_PLUGIN?=y
               FIO_SOURCE_DIR?=<path_to_FIO_source>
               CONFIG_RDMA?=y
[root@host~]# make clean ; ./configure --with-rdma; make; make install
[root@host~]# scripts/setup.sh
```

x. Create huge_pages.

```
[root@host~]# mkdir /root/huge_1GB
[root@host~]# echo 4 > /sys/kernel/mm/hugepages/hugepages-
1048576kB/nr_hugepages
[root@host~]# echo 20 > /sys/kernel/mm/hugepages/hugepages-
2048kB/nr_hugepages
[root@host~]# vim /etc/fstab
nodev      /dev/hugepages          hugetlbfs pagesize=2MB  0 0
nodev      /root/huge_1GB          hugetlbfs pagesize=1GB  0 0
[root@host~]# mount -a
```

Target Configuration

BW/IOPs test:

i. Configure four SPDK NVMe-oF targets using ram-device (malloc), each 512 MB in size.

```
[root@host~]# spdk/app/nvmf_tgt/nvmf_tgt -m 0xFFF -c spdk/etc/spdk/nvmf.conf
[root@host~]# cat spdk/etc/spdk/nvmf.conf
[Malloc]
  NumberOfLuns 4
  LunSizeInMB  512
```

```
[Nvmf]
InCapsuleDataSize 8192

[Subsystem1]
NQN nqn.2016-06.io.spdk:cnode1
Listen RDMA 10.1.1.149:4420
AllowAnyHost Yes
Host nqn.2016-06.io.spdk:init
SN SPDK00000000000001
Namespace Malloc0 1

[Subsystem2]
NQN nqn.2016-06.io.spdk:cnode2
Listen RDMA 10.11.11.149:4421
AllowAnyHost Yes
Host nqn.2016-06.io.spdk:init
SN SPDK00000000000002
Namespace Malloc1 1

[Subsystem3]
NQN nqn.2016-06.io.spdk:cnode3
Listen RDMA 10.1.1.149:4420
AllowAnyHost Yes
Host nqn.2016-06.io.spdk:init
SN SPDK00000000000003
Namespace Malloc2 1

[Subsystem4]
NQN nqn.2016-06.io.spdk:cnode4
Listen RDMA 10.11.11.149:4421
AllowAnyHost Yes
Host nqn.2016-06.io.spdk:init
SN SPDK00000000000004
Namespace Malloc3 1
```

Latency test:

Configure SPDK NVMe target using Intel PCIe NVMe SSD.

```
     [root@host~]# spdk/app/nvmf_tgt/nvmf_tgt -m 0x8
     [root@host~]# sh create_spdk_nvme_rdma_targets.sh 1

     [root@host~]# cat create_spdk_nvme_rdma_targets.sh
     #!/bin/bash
     RPC_PATH=/root/nvme-cli/spdk/scripts
     python $RPC_PATH/rpc.py nvmf_create_transport -t RDMA -c 8192
     for i in `seq 1 $1`
     do
             python $RPC_PATH/rpc.py construct_nvme_bdev -b nvme$i -t pcie -a
0000:0((i + 2)):00.0 ## NVMe SSD PCIe address(in this case, it is 0000:03:00.0 &
0000:04:00.0)
             python $RPC_PATH/rpc.py nvmf_subsystem_create nqn.2016-
06.io.spdk:cnode$i -a -s SPDK0000000000000$i
             python $RPC_PATH/rpc.py nvmf_subsystem_add_ns nqn.2016-
06.io.spdk:cnode$i nvme${i}n1
             python $RPC_PATH/rpc.py nvmf_subsystem_add_listener nqn.2016-
06.io.spdk:cnode$i -t rdma -a 10.1.1.149 -s 4420 -f ipv4
     sleep 0.1
     done
```

Initiator Configuration

BW/IOPs test:

i. Run the fio test on Initiator1 and Initiator2 at the same time.

```
Initiator1:
[root@host1~]# LD_PRELOAD=spdk/examples/nvme/fio_plugin/fio_plugin fio --
rw=randread/randwrite --name=random --norandommap=1 --ioengine=spdk --thread=1
--size=400m --group_reporting --exitall --fsync_on_close=1 --invalidate=1 --
direct=1 --filename='trtype=RDMA adrfam=IPv4 traddr=10.1.1.149 trsvcid=4420
ns=1' --time_based --runtime=20 --iodepth=64 --numjobs=4 --unit_base=1 --
bs=<value>--kb_base=1000 --ramp_time=3

Initiator2:
[root@host2~]# LD_PRELOAD=spdk/examples/nvme/fio_plugin/fio_plugin fio --
rw=randread/randwrite --name=random --norandommap=1 --ioengine=spdk --thread=1
--size=400m --group_reporting --exitall --fsync_on_close=1 --invalidate=1 --
direct=1 --filename='trtype=RDMA adrfam=IPv4 traddr=10.11.11.149 trsvcid=4421
ns=1' --time_based --runtime=20 --iodepth=64 --numjobs=4 --unit_base=1 --
bs=<value>--kb_base=1000 --ramp_time=3
```

Latency test:

i. Single Initiator is used to connect to SPDK target NVMe SSD device/s:

```
Initiator1:
[root@host1~]# LD_PRELOAD=spdk/examples/nvme/fio_plugin/fio_plugin fio --rw=
randread/randwrite --name=random --norandommap=1 --ioengine=spdk --thread=1 --
size=400m --group_reporting --exitall --fsync_on_close=1 --invalidate=1 --
direct=1 --filename='trtype=rdma adrfam=IPv4 traddr=10.1.1.149 trsvcid=4420
ns=1' --time_based --runtime=20 --iodepth=1 --numjobs=1 --unit_base=1 --
bs=<value> --kb_base=1000 --ramp_time=3
```

Conclusion

This paper showcases the remote storage access performance capabilities of Chelsio T6 SPDK NVMe-oF over 100GbE iWARP fabric solution. Using iWARP RDMA enables the NVMe storage devices to be shared, pooled and managed more effectively across a low latency, high performance network. The results show that Chelsio's iWARP RDMA:

- delivers line-rate 99 Gbps throughput for both READ and WRITE.
- reaches 2.5 Million WRITE IOPS at an I/O size of 4K.
- Remote versus local NVMe device access adds only 4.7 and 5.6 µs latency for READ and WRITE.

Related Links

NVMe-oF with iWARP and NVMe/TCP
NVMe over Fabrics Performance JBOF
NVMe over Fabrics Performance for AMD EPYC
High Performance NVMe-oF with T6 100G iWARP RDMA
NVMe Over Fabrics Performance for Qualcomm ARM
NVMe over Fabrics iWARP Performance