100G JBOF using Chelsio Offloads

Using FADU Delta SSDs & Chelsio T6 Adapter

Executive Summary

FADU Technology is a company developing advanced flash storage technology to meet the explosively increasing data storage demands placed on hyperscale, enterprise and cloud data centers. Their innovative SSD solutions are based on industry-standard specifications, designed with FADU's proprietary Flash Memory Controller architecture, and compatibility with multiple industry NAND suppliers. FADU's storage designs address all aspects of Flash-based storage - very low power, ultra-high performance, rich feature sets, solid reliability, and superior QOS. They are currently delivering PCIe Gen3 (Bravo) and Gen4 (Delta) E1.S and U.2 SSDs.

Chelsio Terminator 6 (T6) 1/10/25/40/50/100GbE Unified Wire adapters offering Storage Protocol and TCP/IP offload (*TOE*) capability, enable enterprise storage systems that are purpose-built to deliver optimized NVMe/TCP and iSCSI performance for various application workloads in mission-critical virtualized and private cloud environments. Designed for industry-leading performance, efficiency, and with the unique ability to offload multiple storage protocols (iSCSI, iSER, NVMe/iWARP, NVMe/TCP, FCoE, S2D) using a single ASIC and firmware, Chelsio adapters unburden communication responsibilities and processing overhead from host servers and storage systems resulting in a dramatic increase in application performance with a minimum of CPU cycles.

The combination of FADU SSDs with Chelsio's industry-leading Unified Wire adapter solution delivers compelling performance, power, and total cost of ownership (TCO) advantages. This enables innovative topologies and networked computing models to address the most demanding processing needs. This paper presents the storage performance results of FADU Delta SSDs accessed remotely using the below Chelsio T6 100GbE transports:

- Kernel NVMe/TCP
- Kernel NVMe/TCP Offload
- SPDK NVMe/TCP Offload
- Kernel NVMe/iWARP Offload
- SPDK NVMe/iWARP Offload
- iSCSI Offload

Test Results

The following graph presents throughput and %CPU/Gbps results of various Target (using FADU Delta SSDs) modes with regular Kernel mode NIC hosts. The results are collected using fio tool with I/O size varying from 8 to 256 KBytes with an access pattern of random READs.

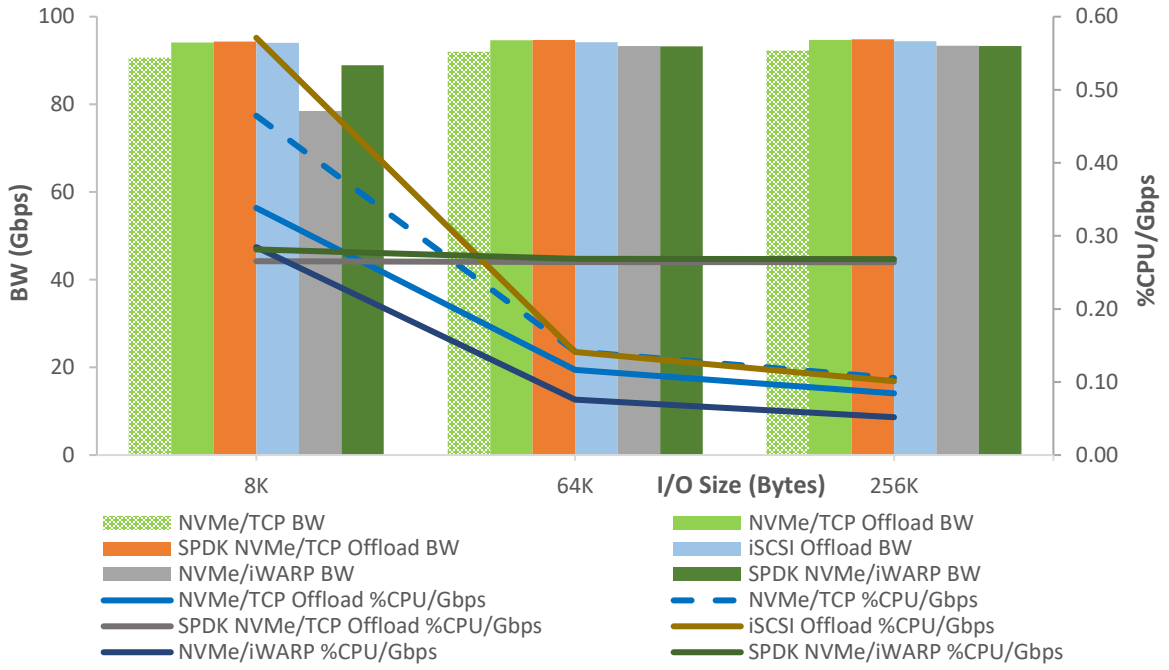


Figure 1 – Throughput and %CPU/Gbps vs. I/O size

The above graph shows how the T6 solution delivers line-rate READ throughput for all the target modes. T6 Kernel NVMe Offloads consume significantly less server CPU compared to the Kernel NVMe/TCP (no-offload).

The following graph presents 4K random IOPs of various Target (using FADU Delta SSDs) modes with regular Kernel mode NIC hosts.

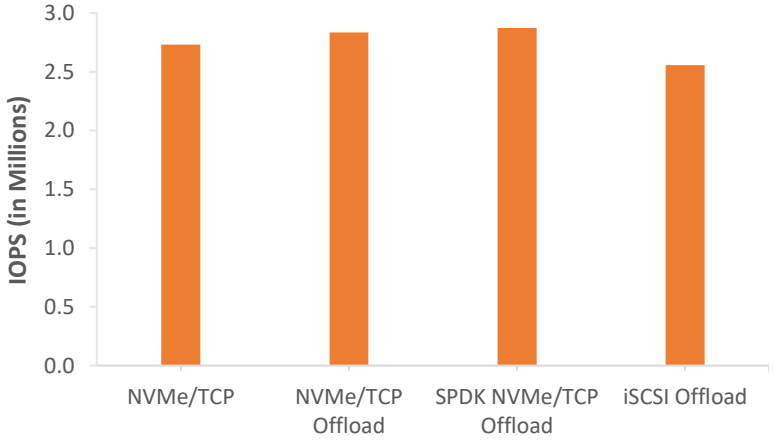


Figure 2 – 4K READ IOPs vs. I/O size

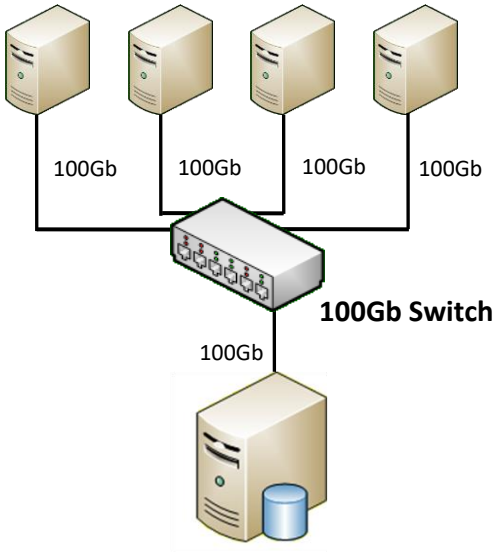
The T6 NVMe Offload enabled solution delivers 2.9M IOPs at 4K IO Size.

The following table presents the 4K Random I/O latency difference between the FADU Delta SSD local to the server and the latency observed when it is accessed remotely using Kernel NIC hosts.

4K Latency		Read			Write		
S.No	Target	Local	Remote	Delta	Local	Remote	Delta
1	NVMe/TCP	20.62	45.1	24.48	10.44	35.2	24.76
2	NVMe/TCP Offload	20.62	40.8	20.18	10.44	32.8	22.36
3	SPDK NVMe/TCP Offload	20.62	36.59	15.97	10.44	28.8	18.36
4	NVMe/iWARP	20.62	33.55	12.93	10.44	24.82	14.38
5	SPDK NVMe/iWARP	20.62	32.2	11.58	10.44	23.05	12.61
6	iSCSI Offload	20.62	45	24.38	10.44	35.83	25.39

T6 Offload (with SPDK) provides the least delta latency between remote and local storage access. This demonstrates the local like performance of remote distributed storage using T6 Offload enabled and SPDK solution.

Test Setup



- Supermicro X10DRH hosts with T62100-CR adapter
- 2 Intel Xeon CPUs E5-2650 v3 10-core @ 2.30 GHz (HT disabled)
- 128 GB RAM
- RHEL 8.5

- Supermicro X12DPi-N6 Target with T62100-CR adapter
- 2 Intel Xeon Silver CPUs 4310 12-core @ 2.10 GHz (HT disabled)
- 128 GB RAM
- RHEL 8.5
- 2 FADU Delta U.2 3.84 TB SSDs

Figure 3 – BW and IOPs Test Setup vs. I/O size

The Bandwidth/IOPs test setup consists of a target machine connected to 4 host machines through a 100GbE switch using a single port on each system.

The Latency test setup consists of a target machine connected to a single host machine back-to-back using a single port on each system.

For both the tests, standard MTU of 1500B is used on the ports under test. The latest Chelsio Unified Wire driver for Linux is installed on all machines.

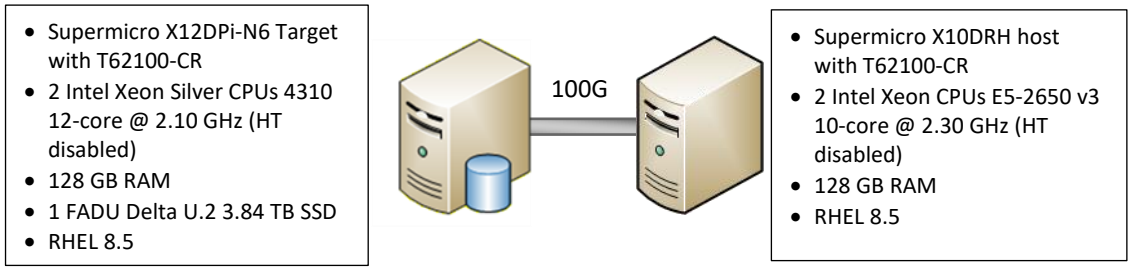


Figure 4 – Latency Test Setup

Storage configuration

For Bandwidth/IOPs test, 2 FADU Delta U.2 3.84 TB SSDs are configured on the target server. 2 hosts connect to each SSD using 20 connections each.

For latency test, the target is configured with 1 FADU Delta U.2 3.84 TB SSD. One host connects to the target using one connection.

Set-up Configuration

General Configuration

Execute the below steps on target and all host machines.

i. Disable virtualization, c-state technology, VT-d, Intel I/O AT, SR-IOV in system BIOS.
ii. Compile and install the latest Chelsio Unified Wire package and reboot the machine.

```
[root@host~]# cd ChelsioUwire-x.x.x.x
[root@host~]# make install
[root@host~]# reboot
```

iii. Add the below parameters to grub kernel command line.
BW/IOPs test: intel_idle.max_cstate=0 processor.max_cstate=0 intel_pstate=disable
Latency test: idle=poll

iv. Set cpupower governor to performance.

```
[root@host~]# cpupower frequency-set --governor performance
```

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 NIC driver (*cxgb4*) and bring up interface with an IPv4 address.

```
[root@host~]# modprobe cxgb4
[root@host~]# ifconfig ethX <IPv4 address> up
```

Kernel NVMe-oF Target Configuration

i. Load the Chelsio NVMe/TCP Offload drivers (not required for non-offload NVMe/TCP).

```
[root@host~]# modprobe t4_tom
```

Note: For NVMe/iWARP, load *iw_cxgb4* instead of *t4_tom*.

ii. Apply the below cop policy for NVMe/TCP Offload.

```
[root@host~]# cat /root/cop_policy
all => offload !nagle !ddp !coalesce
[root@host~]# cop -d -o file /root/cop_policy
[root@host~]# cxgbtool ethX policy file
```

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

```
[root@host~]# t4_perftune.sh -s -n -Q ofld -c 0-23
```

Set the CPU affinity for Latency test to use a single CPU (0 in this case).

```
[root@host~]# t4_perftune.sh -s -n -Q ofld -c 0
```

Note: For NVMe/TCP, please use *nic* instead of *ofld*. For NVMe/iWARP, please use *rdma* instead of *ofld*.

iv. Load the NVMe drivers.

```
[root@host~]# modprobe nvmet
[root@host~]# modprobe nvmet-tcp
[root@host~]# modprobe nvmet-rdma
```

v. Configure the target using the below script.

```
#!/bin/bash
nvmetcli clear > /dev/null 2>$1
mount -t configfs none /sys/kernel/config > /dev/null 2>$1

IPPORT="4420"     # 4420 is the reserved NVME/Fabrics port
IPADDR="0.0.0.0"
NAME="nvme-ssd"
DEV="/dev/nvme"

for i in `seq 0 1`; do
mkdir /sys/kernel/config/nvmet/subsystems/${NAME}${i}
mkdir -p /sys/kernel/config/nvmet/subsystems/${NAME}${i}/namespaces/1
echo -n "${DEV}${i}n1"
>/sys/kernel/config/nvmet/subsystems/${NAME}${i}/namespaces/1/device_path
echo 1 > /sys/kernel/config/nvmet/subsystems/${NAME}${i}/attr_allow_any_host
echo 1 > /sys/kernel/config/nvmet/subsystems/${NAME}${i}/namespaces/1/enable
done

mkdir /sys/kernel/config/nvmet/ports/1
# echo 8192 > /sys/kernel/config/nvmet/ports/1/param_inline_data_size
echo "ipv4" > /sys/kernel/config/nvmet/ports/1/addr_adrfam
echo "tcp" > /sys/kernel/config/nvmet/ports/1/addr_trtype
echo $IPPORT > /sys/kernel/config/nvmet/ports/1/addr_trsvcid
```

```
echo $IPADDR > /sys/kernel/config/nvmet/ports/1/addr_traddr

for i in `seq 0 1`; do
                ln -s /sys/kernel/config/nvmet/subsystems/${NAME}${i}
/sys/kernel/config/nvmet/ports/1/subsystems/${NAME}${i}
done
```

Note: For NVMe/iWARP, please use *param_inline_data_size* and use *rdma* instead of *tcp*.

SPDK NVMe-oF Target Configuration

i. Load the Chelsio SPDK NVMe/TCP Offload driver.

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

Note: For NVMe/iWARP, please use *iw_cxgb4* instead of *chtcp*.

ii. Configure Huge Pages.

```
[root@host~]# echo 8192 > /proc/sys/vm/nr_hugepages
[root@host~]# echo 8192 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
[root@host~]# cd ChelsioUwire-x.x.x.x/build/src/chspdk/user/spdk/
[root@host~]# CLEAR_HUGE=yes HUGENODE='nodes_hp[1]=8192,nodes_hp[0]=8192'
scripts/setup.sh config
```

iii. Start the target.

```
[root@host~]# cd ChelsioUwire-x.x.x.x/build/src/chspdk/user/spdk/
[root@host~]# ./build/bin/nvmf_tgt -m 3F000
```

iv. Configure the target with the SSDs.

```
SPDK_PATH=$'ChelsioUwire-x.x.x.x/build/src/chspdk/user/spdk/'
$SPDK_PATH/scripts/rpc.py nvmf_create_transport -t TCP
$SPDK_PATH/scripts/rpc.py bdev_nvme_attach_controller -b NVMe0 -t PCIe -a 98:00.0
$SPDK_PATH/scripts/rpc.py bdev_nvme_attach_controller -b NVMe1 -t PCIe -a e3:00.0

for i in `seq 0 1`
do
        $SPDK_PATH/scripts/rpc.py nvmf_create_subsystem nqn.2016-
06.io.spdk:cnode$i -a -s SPDK0000000000000$i -d SPDK_Controller$i
        $SPDK_PATH/scripts/rpc.py nvmf_subsystem_add_ns nqn.2016-
06.io.spdk:cnode$i NVMe${i}n1
        $SPDK_PATH/scripts/rpc.py nvmf_subsystem_add_listener nqn.2016-
06.io.spdk:cnode$i -t tcp -a 10.2.2.136 -s 4420
sleep 0.1
done
```

Note: For NVMe/iWARP, please use *RDMA* instead of *TCP* and *rdma* instead of *tcp*.

Kernel NVMe-oF Host Configuration

Kernel NVMe/TCP (no offload) is used on the hosts to connect to Kernel/SPDK NVMe/TCP Targets configured in above sections. Kernel NVMe/iWARP offload is used on the hosts to connect to Kernel/SPDK NVMe/iWARP offload Targets configured in above sections.

i. Load the NVMe drivers.

```
[root@host~]# modprobe nvme
[root@host~]# modprobe nvme-tcp
[root@host~]# modprobe nvme-rdma
```

ii. Enable Adaptive Rx for the Chelsio Interface.

```
[root@host~]# ethtool -C ethX adaptive-rx on
```

iii. Set the CPU affinity for BW/IOPs test (to use *local_cpulist* of interface).

```
[root@host~]# t4_perftune.sh -s -n -Q nic -c 10-19
```

Set the CPU affinity for Latency test to use a single CPU (0 in this case).

```
[root@host~]# t4_perftune.sh -s -n -Q nic -c 0
```

Note: For NVMe/iWARP, please use *rdma* instead of *nic*.

BW/IOPs test:

i. Connect the 4 hosts to the targets.

```
[root@host1~]# nvme connect -t tcp -s 4420 -a 10.2.2.136 -n nvme-ssd0
[root@host2~]# nvme connect -t tcp -s 4420 -a 10.2.2.136 -n nvme-ssd0
[root@host3~]# nvme connect -t tcp -s 4420 -a 10.2.2.136 -n nvme-ssd1
[root@host4~]# nvme connect -t tcp -s 4420 -a 10.2.2.136 -n nvme-ssd1

nqn.2016-06.io.spdk:cnode0 and nqn.2016-06.io.spdk:cnode1 should be used
while connecting to the SPDK NVMe/TCP Offload Target.
```

Note: For NVMe/iWARP, please use *rdma* instead of *tcp*.

ii. Run *fio* tool on all 4 hosts at the same time.

```
[root@host~]# fio --rw=randwrite/randread --ioengine=libaio --name=random --
norandommap --group_reporting --exitall --fsync_on_close=1 --invalidate=1 --
direct=1 --runtime=60 --time_based --filename= <device list> --iodepth=16 --
numjobs=20 --bs=<value> --unit_base=1 -kb_base=1000 --ramp_time=2
```

Latency test:

i. Connect single host to the target.

```
[root@host~]# nvme connect -t tcp -a 10.2.2.136 -n nvme-ssd0 -i 1

nqn.2016-06.io.spdk:cnode0 should be used while connecting to the SPDK
NVMe/TCP Offload Target.
```

Note: For NVMe/iWARP, please use *rdma* instead of *tcp*.

ii. Run *fio* tool on the host.

```
[root@host~]# fio --rw=randwrite/randread --ioengine=libaio --name=random --
invalidate=1 --direct=1 --runtime=60 --time_based --fsync_on_close=1 --
group_reporting --filename= <device list> --iodepth=1 --numjobs=1 --bs=4K --
unit_base=1 -kb_base=1000 --ramp_time=2
```

iSCSI Offload Target Configuration

i. Load the Chelsio LIO PDU Offload Target driver.

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

ii. Set the CPU affinity for BW/IOPs test (to use *local_cpulist* of interface).

```
[root@host~]# t4_perftune.sh -s -n -Q iSCSIT -c 12-23
```

Set the CPU affinity for Latency test to use a single CPU (0 in this case).

```
[root@host~]# t4_perftune.sh -s -n -Q iSCSIT -c 0
```

iii. Create 10 namespaces on each NVMe SSDs.

```
[root@host~]# nvme delete-ns /dev/nvme0 -n 1
[root@host~]# nvme delete-ns /dev/nvme1 -n 1
[root@host~]# for i in `seq 1 10` ; do nvme create-ns /dev/nvme0 --
nsze=58605568 --ncap=58605568 --flbas=0 -dps=0 ; done
[root@host~]# for i in `seq 1 10` ; do nvme create-ns /dev/nvme1 --
nsze=58605568 --ncap=58605568 --flbas=0 -dps=0 ; done
[root@host~]# for i in `seq 1 10`; do nvme attach-ns /dev/nvme0 --namespace-
id=${i} -controllers=0x1 ; done
[root@host~]# for i in `seq 1 10`; do nvme attach-ns /dev/nvme1 --namespace-
id=${i} -controllers=0x1 ; done
```

iv. Configure the target using the below script.

```
i=1
for j in `seq 1 10` ; do
        /usr/local/bin/targetcli /backstores/block/ create name=nvme0n${j}
dev=/dev/nvme0n${j} readonly=false
        /usr/local/bin/targetcli /iscsi create iqn.2022-11.org.linux-
iscsi.target${i}
        /usr/local/bin/targetcli /iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpg1/ set attribute authentication=0
demo_mode_write_protect=0 generate_node_acls=1 cache_dynamic_acls=1
        /usr/local/bin/targetcli /iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpg1/luns create lun=0
storage_object=/backstores/block/nvme0n${j}
        /usr/local/bin/targetcli /iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpg1/portals/ delete ip_address=0.0.0.0 ip_port=3260
        /usr/local/bin/targetcli /iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpg1/portals create ip_address=11.11.11.10 ip_port=3260
        echo 1 > /sys/kernel/config/target/iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpgt_1/np/11.11.11.10\:3260/cxgbit
        let i=i+1
done

for j in `seq 1 10` ; do
        /usr/local/bin/targetcli /backstores/block/ create name=nvme1n${j}
dev=/dev/nvme1n${j} readonly=false
        /usr/local/bin/targetcli /iscsi create iqn.2022-11.org.linux-
iscsi.target${i}
        /usr/local/bin/targetcli /iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpg1/ set attribute authentication=0
demo_mode_write_protect=0 generate_node_acls=1 cache_dynamic_acls=1
```

```
        /usr/local/bin/targetcli /iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpg1/luns create lun=0
storage_object=/backstores/block/nvme1n${j}
        /usr/local/bin/targetcli /iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpg1/portals/ delete ip_address=0.0.0.0 ip_port=3260
        /usr/local/bin/targetcli /iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpg1/portals create ip_address=11.11.11.10 ip_port=3260
        echo 1 > /sys/kernel/config/target/iscsi/iqn.2022-11.org.linux-
iscsi.target$i/tpgt_1/np/11.11.11.10\:3260/cxgbit
        let i=i+1
done
```

iSCSI Initiator Configuration

i. Enable Adaptive Rx for the Chelsio Interface.

```
[root@host~]# ethtool -C ethX adaptive-rx on
```

ii. Set the CPU affinity for BW/IOPs test (to use *local_cpulist* of interface).

```
[root@host~]# t4_perftune.sh -s -n -Q nic -c 10-19
```

Set the CPU affinity for Latency test to use a single CPU (0 in this case).

```
[root@host~]# t4_perftune.sh -s -n -Q nic -c 0
```

BW/IOPs test:

i. Connect all the 4 initiators to the targets using the below command.

```
[root@host1~]# for i in `seq 1 20` ; do iscsiadm -m node -T iqn.2022-
11.org.linux-iscsi.target${i} -p 10.10.10.10 -l ; done
```

ii. Run *fio* tool on all 4 initiators at the same time.

```
[root@host~]# fio --rw=randwrite/randread --ioengine=libaio --name=random --
norandommap --group_reporting --exitall --fsync_on_close=1 --invalidate=1 --
direct=1 --runtime=60 --time_based --filename= <device list> --iodepth=64 --
numjobs=20 --bs=<value> --unit_base=1 -kb_base=1000 --ramp_time=2
```

Latency test:

i. Connect single initiator to the target.

```
[root@host~]# iscsiadm -m node -T iqn.2022-11.org.linux-iscsi.target1 -p
10.10.10.10 -l
```

ii. Run *fio* tool on the host.

```
[root@host~]# fio --rw=randwrite/randread --ioengine=libaio --name=random --
invalidate=1 --direct=1 --runtime=60 --time_based --fsync_on_close=1 --
group_reporting --filename= <device list> --iodepth=1 --numjobs=1 --bs=4K --
unit_base=1 -kb_base=1000 --ramp_time=2
```

Conclusion

This paper showcases the performance capabilities of Chelsio T6 100G Offload-enabled adapters with FADU Delta SSDs. With concurrent support for NVMe/TCP, NVMe/iWARP and iSCSI, users can create and maintain a true Converged Fabric-based server cluster for software-defined storage and other applications. The test result proof points in this paper show that T6 Offload:

- Delivers line-rate 94 Gbps READ throughput.
- Reaches 2.9 Million IOPs at 4K I/O size.
- Provides local-like access to remote storage.
- Provides significant CPU savings compared to NVMe/TCP.

The Chelsio T6 enables the FADU SSDs to be shared, pooled, and managed more effectively across a low latency, high-performance, scalable, standard Ethernet network, with CPU server savings – delivering a highly cost-effective solution.

Related Links

FADU Delta Gen4 SSD
100G Kernel and User Space NVMe/TCP Using Chelsio Offload
"No-compromise" NVMe/TCP Deployment using Server Storage I/O Offload
100G iSCSI Performance for AMD EPYC
Demartek Evaluation: Chelsio Terminator 6 (T6) Unified Wire Adapter iSCSI Offload
Windows iSCSI Performance at 100Gbps