SoftiWARP Performance with Chelsio 100GbE

NVMe-oF Bandwidth, IOPS and Latency Performance

Executive Summary

SoftiWARP is an open source software implementation of the iWARP protocol suite. Due to close integration with the Linux kernel socket layer, SoftiWARP allows for efficient data transfer operations and since the implementation conforms to the iWARP protocol specification, it is wire compatible with any peer network adapter (RNIC) implementing iWARP in hardware. SoftiWarp was contributed by IBM Zurich Research and very recently became part of the Linux operating system.

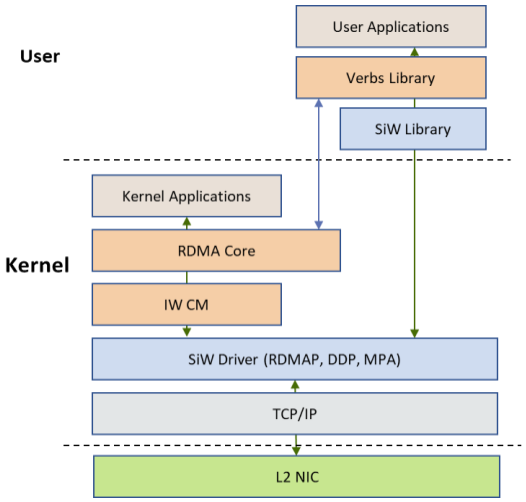

Figure 1 – SoftiWARP

This paper presents the 100G NVMe-oF performance results with iWARP RDMA Offload Target and SoftiWARP Initiator. Chelsio's T6 with SoftiWARP delivers line-rate throughput and more than 2.1 Million IOPS at 4K I/O size. In addition, with only 12 µs delta latency between remote and local storage, Chelsio's solution proves to be the best-in-breed in providing the next generation, scalable storage network over standard and cost-effective Ethernet infrastructure with an efficient processing path.

Why SoftiWARP?

Chelsio adapters supporting iWARP RDMA hardware Offload completely bypass the host software stack processing thus eliminating any inefficiencies due to software processing. They now additionally support software iWARP solution which offers the below advantages:

- Provides a simple path for transition of RDMA applications to the cloud platform.
- Is useful for Client/Initiator side applications like iSER, NVMe-oF, NFSoRDMA, LustreoRDMA etc. to connect to hardware offloaded versions on the target side.
- Supports the ability to work with any legacy switch infrastructure, enabling a decoupled server and switch upgrade cycle.

Test Results

The following graph presents NVMe-oF random READ and random WRITE performance (IOPS and throughput) results of a typical deployment running Chelsio T6 hardware offload on the Target side and SoftiWARP using Chelsio T6 Unified Wire in L2 NIC mode on the Initiator side with Null Block devices. The results were collected using the **fio** tool with I/O sizes varying from 4 to 256 Kbytes.

Copyright 2019. Chelsio Communications Inc. All rights reserved.

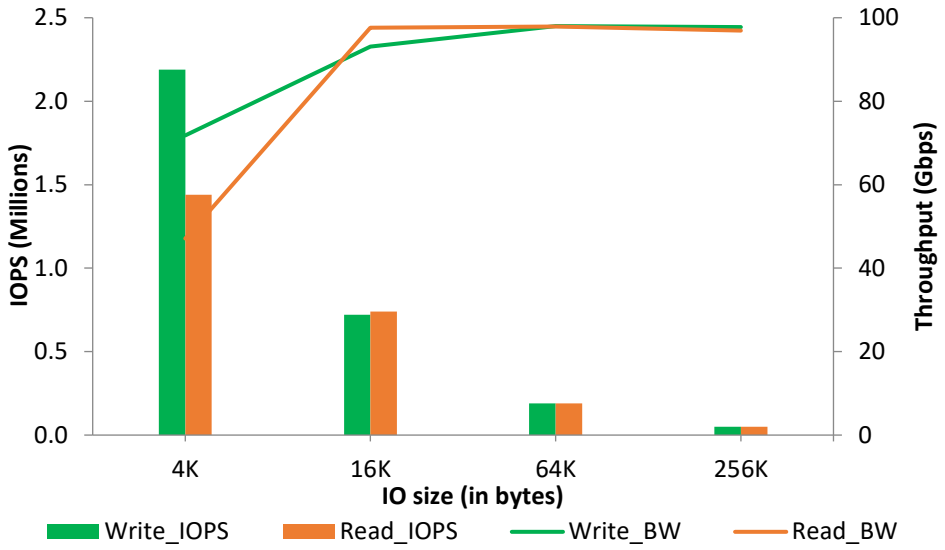

Figure 2 – NVMe-oF IOPS and Throughput vs. I/O size

While we continue to tune the performance, we see 2 x Initiators running SoftiWARP over Chelsio T6 adapters can deliver 100G line-rate throughput.

The following table shows the latency at 4K I/O size with 1 and 2 connections using NVMe SSDs.

# of Connections	1	2
	Latency (µSec)	
Local Read	90.16	90.59
Remote Read	106.19	109.09
Delta Local v. Remote Read	**16**	**19**
Local Write	10.64	10.27
Remote Write	22.76	23.07
Delta Local v. Remote Write	**12**	**13**

Table 1 - Latency vs. #. connection

Remote access adds only 12 µs and 16 µs latency to the native storage access speed. Measured at 4K I/O for WRITE and READ operations respectively.

Test Setup

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

Copyright 2019. Chelsio Communications Inc. All rights reserved.

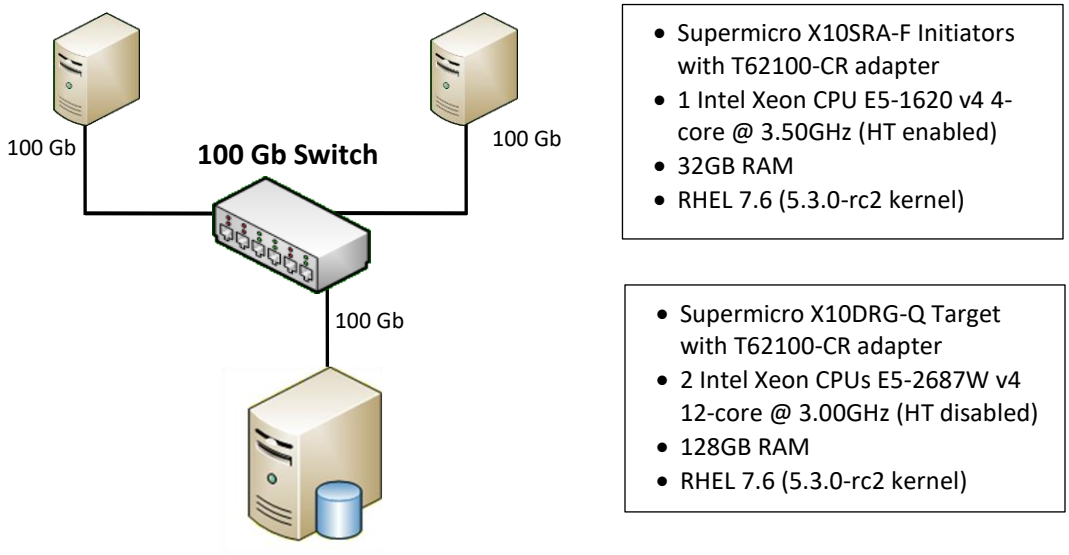

Figure 3 – Test setup

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.

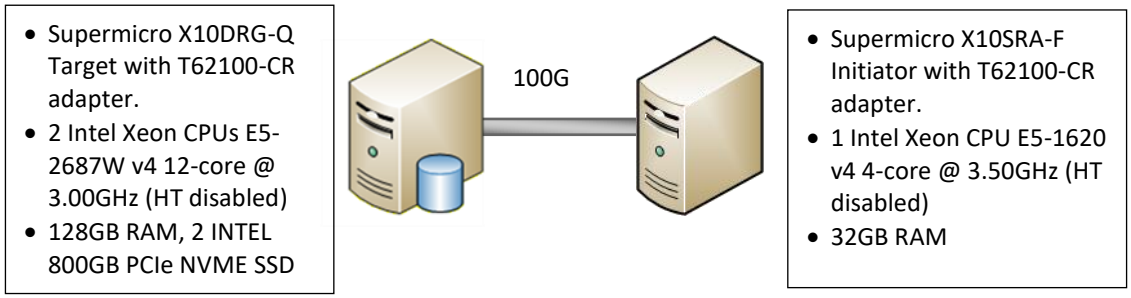

Figure 4 – Latency Test setup

Storage configuration

The Target server is enabled with T6 iWARP RDMA Offload and configured with 16 NVMe-oF targets, using NULL BLOCK devices, each 1GB in size. Each SoftiWARP Initiator connects to 8 targets using 2 connections per target.

For latency test, 2 targets are configured, each with 1 Intel NVMe SSD. The Initiator connects to the target devices.

Setup Configuration

Target/Initiator Configuration

i. Disable virtualization, c-state technology, VT-d, Intel I/O AT and SR-IOV in system BIOS.

ii. Compile and install the latest 5.3 kernel from kernel.org, https://git.kernel.org/torvalds/t/linux-5.3-rc2.tar.gz.

Copyright 2019. Chelsio Communications Inc. All rights reserved.

Make sure the below kernel config parameters are enabled:

```
CONFIG_RDMA_SIW=m
CONFIG_NVME_RDMA=m
CONFIG_NVME_CORE=m
CONFIG_NVME_FABRICS=m
CONFIG_NVME_TARGET=m
CONFIG_NVME_TARGET_RDMA=m
CONFIG_NVME_MULTIPATH=y
```

iii. Reboot the machine into the newly installed kernel.
iv. 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
```

v. For latency test, add *idle=poll* to the kernel command line.
vi. Stop the iwpmd service.

```
[root@host~]# systemctl disable iwpmd.service
[root@host~]# systemctl stop iwpmd.service
```

Target Configuration

i. Download the latest Chelsio Unified Wire for Linux package from https://service.chelsio.com/, extract it and copy the NVMe-oF performance config file.

```
[root@host~]# tar zxfv ChelsioUwire-x.x.x.x.tar.gz
[root@host~]# cp ChelsioUwire-
x.x.x.x/src/network/firmware/nvme_perf_config/t6-config.txt
/lib/firmware/cxgb4/.
```

ii. Load the Chelsio iWARP RDMA Offload driver and bring up the interface with an MTU 9000.

```
[root@host~]# modprobe iw_cxgb4 peer2peer=0 crc_enabled=0
[root@host~]# ifconfig ens2f4 10.1.1.149/24 mtu 9000 up
```

NOTE: MPA CRC was disabled to prevent the SoftiWARP Initiator machines doing this CPU intensive tasks.

iii. Load the NVMe drivers.

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

BW/IOPs test:

i. Clone the below repo to get the 'nvmetcli' utility. Follow the README inside for installing it.

```
[root@host~]# git clone git://git.infradead.org/users/hch/nvmetcli.git
```

Copyright 2019. Chelsio Communications Inc. All rights reserved.

ii. Create 16 Null Block devices, each of 1GB size.

```
[root@host~]# modprobe null_blk nr_devices=16 gb=1 use_per_node_hctx=Y
```

iii. Configure 16 targets using the below script.

```
IPPORT="4420"        # 4420 is the reserved NVME/Fabrics RDMA port
IPADDR="10.1.1.149"  # the ipaddress of your target rdma interface
NAME="nvme-nullb"    # Use "nvme-ssd" while configuring SSDs
DEV="/dev/nullb"     # Use " /dev/nvme" while configuring SSDs

for i in `seq 0 15`; do      # For latency test, use `seq 0 1`
mkdir /sys/kernel/config/nvmet/subsystems/${NAME}${i}
mkdir -p /sys/kernel/config/nvmet/subsystems/${NAME}${i}/namespaces/1
echo -n ${DEV}${i}
>/sys/kernel/config/nvmet/subsystems/${NAME}${i}/namespaces/1/device_path
# Use "echo -n ${DEV}${i}n1" while configuring SSDs
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 "rdma" > /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 15`; do
 ln -s /sys/kernel/config/nvmet/subsystems/${NAME}${i}
/sys/kernel/config/nvmet/ports/1/subsystems/${NAME}${i}
done
```

iv. Run the below to check the target configuration.

```
  [root@host~]# nvmetcli/nvmetcli ls
```

Initiator Configuration

i. Install the rdma-core by following the below procedure to get the SoftiWARP user space library (libsiw).

Note: 'pandoc' library is needed for generating rdma-core rpms. It can be downloaded from https://github.com/jgm/pandoc/releases and can be installed as shown below:

```
[root@host~]# tar xvzf pandoc-X.X-linux.tar.gz --strip-components 1 -C
/usr/local/

[root@host~]# mkdir -p rpmbuild/SOURCES
[root@host~]# cd rpmbuild/SOURCES/
[root@host~]# git clone https://github.com/linux-rdma/rdma-core rdma-core
[root@host~]# grep "Version:" rdma-core/redhat/rdma-core.spec Version: 26.0
```

Rename the rdma-core by appending the version number as shown below.

```
[root@host~]# mv rdma-core rdma-core-26.0
[root@host~]# tar -cf rdma-core-26.0.tgz ./rdma-core-26.0
[root@host~]# cd rdma-core-26.0
```

Copyright 2019. Chelsio Communications Inc. All rights reserved.

Build rdma-core RPMs (All rdma-core rpms including libibverbs, librdmacm etc will be generated under 'rpmbuild/RPMS/' directory):

```
[root@host~]# rpmbuild -bb ./redhat/rdma-core.spec
[root@host~]# cd /root/rpmbuild/RPMS/x86_64
```

Uninstall all the previously installed rdma-core, librdmacm and libibverbs rpms on the system and install the newly built RPMs using 'rpm -ivh <rpm-name>' command.

ii. Clone and compile the 'nvme-cli' repo to get the 'nvme' command.

```
[root@host~]# git clone https://github.com/linux-nvme/nvme-cli
[root@host~]# cd nvme-cli
[root@host~]# make
[root@host~]# ./nvme --version
nvme version 1.8.1.172.g1235
```

iii. Clone the iproute2 package and install the 'rdma' tool.

a. Git link: https://git.kernel.org/pub/scm/network/iproute2/iproute2.git , branch: master
b. Pre-requisites: Install libmnl and libmnl-devel rpms. To find out more dependencies, run './configure' and install the missing packages.

iv. Load the necessary drivers along with 'siw' and bring up the interface.

```
[root@host~]# modprobe cxgb4
[root@host~]# modprobe siw
[root@host~]# rmmod iw_cxgb4
[root@host~]# rdma link add siw-ens1f4 type siw netdev ens1f4
[root@host~]# ifconfig ens1f4 10.1.1.150/24 mtu 9000 up
```

v. Load the NVMe drivers.

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

vi. Discover the target.

```
[root@host~]# nvme discover -t rdma -a <target_ip> -s 4420 -q default
```

BW/IOPs test:

i. Initiator1 connects to 8 Targets.

```
[root@host1~]# for i in `seq 0 7`; do nvme connect -i 2 -t rdma -a <target_ip>
-s 4420 -n nvme-nullb${i} -q default; done
```

ii. Initiator2 connects to 8 Targets.

```
[root@host2~]# for i in `seq 8 15`; do nvme connect -i 2 -t rdma -a
<target_ip> -s 4420 -n nvme-nullb${i} -q default; done
```

Copyright 2019. Chelsio Communications Inc. All rights reserved.

iii. *fio* tool is run on both initiators at the same time.

```
[root@host~]# fio --rw=randwrite/randread --ioengine=libaio --norandommap --
name=random  --size=400m --invalidate=1 --exitall --fsync_on_close=1 --
direct=1 --runtime=30 --time_based --group_reporting --filename= <device list>
--iodepth=64 --numjobs=16 --bs=<value> --unit_base=1 --kb_base=1000 --
ramp_time=2
```

iv. To disconnect the targets.

```
[root@host~]# nvme disconnect-all
```

Latency test:

i. Single Initiator connects to the target.

```
[root@host~]# for i in `seq 0 $((N))`; do /root/nvme-cli/nvme connect -t rdma
-a 10.1.1.149 -n nvme-ssd${i} -i 1; done
```

where N specifies the number of target devices.

ii. *fio* tool is run on the initiator.

```
[root@host~]# fio --rw=randwrite/randread --ioengine=libaio --name=random  --
size=400m --invalidate=1 --direct=1 --runtime=30 --time_based --
fsync_on_close=1 --group_reporting --filename= <device list> --iodepth=1 --
numjobs=1 --bs=4K
```

iii. After collecting latency for single target device, initiator was disconnected and connected to two target devices for latency collection.

Conclusion

This paper showcases the remote storage access performance capabilities of Chelsio T6 100G NVMe-oF with SoftiWARP initiator and hardware iWARP offload on target. It also proves the SoftiWARP solution is compatible with Chelsio RNICs which already has iWARP implemented in the hardware. The complete solution is in-boxed in latest Linux kernel and is readily available to configure and use. Combining both can result in significant acquisition with operational savings and can achieve adequate performance for applications (like iSER initiator, NVMe-oF initiator, NFSRDMA client and others). Using iWARP RDMA enables the NVMe storage devices to be shared, pooled and managed more effectively across a low latency, high performance network.

References

https://arxiv.org/pdf/1703.07626.pdf

Copyright 2019. Chelsio Communications Inc. All rights reserved.