Benchmark Guide for mTCP with DSA

1. Hardware Environmental Preparation

1. Two machines directly connected via 100G network card.
2. One of the two must be a SPR machine, and this SPR machine kernel(v5.13) support DSA
    1. To compile the kernel, you can refer to this document: "**kernel_build_and_iax_enable_v1.0**.pdf" (The steps for compiling the kernel are universal)
    2. Read the **README**.md in the **dsa_userlib** directory to install the necessary libraries.

       Tips:

        1. when you clone : https://github.com/intel/idxd-config.git , you should switch the version, do not use the latest version: you can

           git checkout 6bd68e68;

        2. Modify config file : /etc/ld.so.conf , add a new line : "./", and then run command "ldconfig", Configure the current directory./ to the library search path .
        3. Then follow the readme to continue.
3. SPR machine BIOS Setting

BIOS Setting
EDKII Menu > Socket Configuration > Uncore Configuration > Uncore Dfx Configuration: Cache entries for non-atomics = 120
EDKII Menu > Socket Configuration > Uncore Configuration > Uncore Dfx Configuration: Cache entries for atomics = 8
EDKII Menu > Socket Configuration > Uncore Configuration > Uncore Dfx Configuration: CTAG entry avail mask = 255
EDKII Menu-> "Socket Configuration" -> "IIO Configuration" -> "Intel VT for directed IO (VT -d)" → Intel VT for directed IO → Enable
EDKII Menu-> "Socket Configuration" -> "IIO Configuration" -> "PCI ENQCMD/ENQCMDS" → Enable**

2. Software Environmental Preparation

Configure demo machine(spr machine run demo)

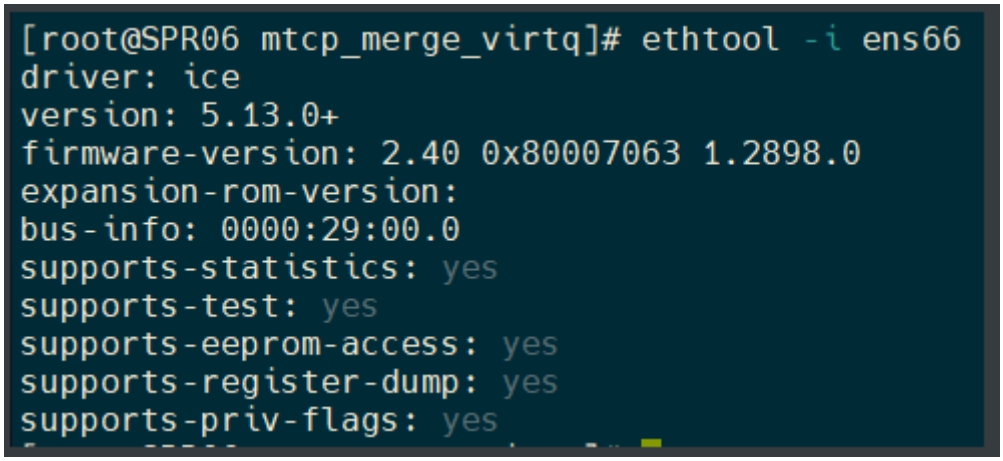
1. Record Information

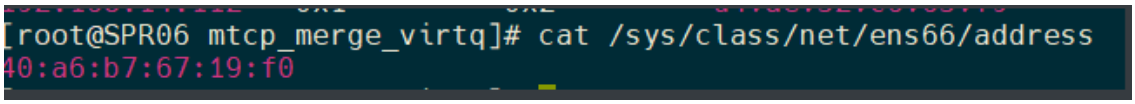
View and record the information of the directly connected network card

Suppose, you are directly connected as follows:

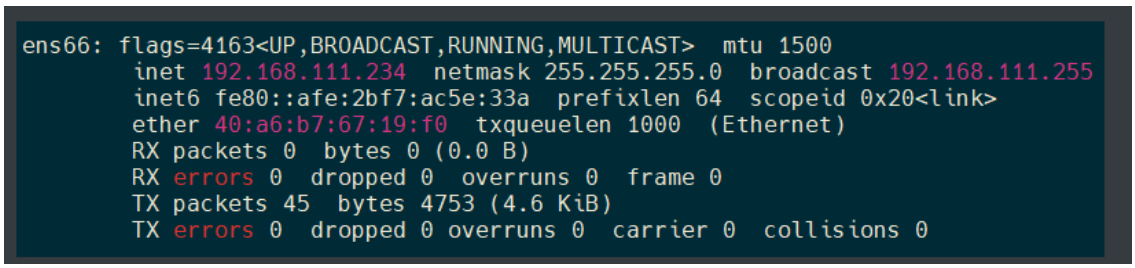
192.168.1.1（pktgen machine） NIC: ens5f1 <------NIC direct connection------> **192.168.1.2 （demo machine） NIC: ens66(0000:29:00.0)**

You should record these information, for example:





or enter "ifconfig":



NIC: ens66

ens66 bus-info:0000:29:00.0

ens66 mac address: 40:A6:B7:67:19:F0

then, uninstall an existing drive

```
ifconfig ens66 down
```

2. Download Source Code

```
# repo: https://github.com/intel-collab/libraries.performance.accelerators.data-streaming.dsa-share-memory-framework
# Source Code Main Directory Structure
    dsa_userlib # DSA library
    ├─config_dsa # config file is used to dsa
    |─include # dsa library headfile
    ├─src  # dsa library source code
    ├─test # dsa test program, you can read dsa readme.md to learn how to run it
    mtcp_merge_virtq # mtcp source folder
    ├─apps
    |   ├─example # mtcp app example folder
    |   | ├─ epping.c # mtcp app demo
    |   | ├─ epping.conf # mtcp app demo config file
    ├─mtcp # mtcp lib source code
    ├─dpdk # dpdk lib souce code
    ...

git clone git@github.com:intel-collab/libraries.performance.accelerators.data-streaming.dsa-share-memory-framework.git
# Get into source code path
cd /dsa-accelerated-mtcp/mtcp_merge_virtq
# Run prepare shell (only need run once)
./compile_prepare.sh
```

Bind NIC with igb_uio Driver (Need to reconfigure after reboot)

```
# https://doc.dpdk.org/dts/gsg/usr_guide/igb_uio.html
1. Igb_uio driver can build by dpdk
2. Bind nic with igb_uio driver. This network card is directly connected to another machine.

TIPS:
1. If you need to use a lasted version of dpdk to compile IGB_ UIO driver, please refer to:  https://doc.dpdk.org/dts/gsg/usr_guide/igb_uio.html
2. You can also use default dpdk(Not the latest version) follow the steps below (after unzip /dsa-accelerated-mtcp/mtcp_merge_virtq/dpdk/x86_64-native-linuxapp-gcc.tar.gz):
cd /dsa-accelerated-mtcp/mtcp_merge_virtq/
export RTE_SDK=`echo $PWD`/dpdk
export RTE_TARGET=x86_64-native-linuxapp-gcc
./setup_mtcp_dpdk_env.sh $RTE_SDK
1. build x86_64-native-linuxapp-gcc (for this dpdk version , enter 38 to build it,
If there is an error in this step, you can download a separate dpdk kmod repo and compile it)
```

```
[root@SPR06 mtcp_merge_virtq]# ./setup_mtcp_dpdk_env.sh $RTE_SDK
```

```
[36] x86_64-native-freebsd-gcc
[37] x86_64-native-linuxapp-clang
[38] x86_64-native-linuxapp-gcc
[39] x86_64-native-linuxapp-icc
[40] x86_64-native-linux-clang
[41] x86_64-native-linux-gcc
[42] x86_64-native-linux-icc
[43] x86_x32-native-linuxapp-gcc
[44] x86_x32-native-linux-gcc

------------------------------------------------------------
 Step 2: Setup linux environment
------------------------------------------------------------
[45] Insert IGB UIO module
[46] Insert VFIO module
[47] Insert KNI module
[48] Setup hugepage mappings for non-NUMA systems
[49] Setup hugepage mappings for NUMA systems
[50] Display current Ethernet/Baseband/Crypto device settings
[51] Bind Ethernet/Baseband/Crypto device to IGB UIO module
[52] Bind Ethernet/Baseband/Crypto device to VFIO module
[53] Setup VFIO permissions

------------------------------------------------------------
 Step 3: Run test application for linux environment
------------------------------------------------------------
[54] Run test application ($RTE_TARGET/app/test)
[55] Run testpmd application in interactive mode ($RTE_TARGET/app/testpmd)

------------------------------------------------------------
 Step 4: Other tools
------------------------------------------------------------
[56] List hugepage info from /proc/meminfo

------------------------------------------------------------
 Step 5: Uninstall and system cleanup
------------------------------------------------------------
[57] Unbind devices from IGB UIO or VFIO driver
[58] Remove IGB UIO module
[59] Remove VFIO module
[60] Remove KNI module
[61] Remove hugepage mappings

[62] Exit Script

Option: 38
```

```
export RTE_SDK=/root/dpdk-stable-18.11.11
```

```
== Build drivers/raw/dpaa2_qdma
== Build drivers/raw/ifpga_rawdev
== Build app
== Build app/test-pmd
== Build app/proc-info
== Build app/pdump
== Build app/test-bbdev
== Build app/test-crypto-perf
== Build app/test-eventdev
Build complete [x86_64-native-linuxapp-gcc]
Installation cannot run with T defined and DESTDIR undefined
------------------------------------------------------------------------
 RTE_TARGET exported as x86_64-native-linuxapp-gcc
------------------------------------------------------------------------

Press enter to continue ...
```

2. Insert IGB UIO module (for this dpdk version , enter 45 to build it)

```
------------------------------------------------------------
 Step 2: Setup linux environment
------------------------------------------------------------
[45] Insert IGB UIO module
[46] Insert VFIO module
[47] Insert KNI module
[48] Setup hugepage mappings for non-NUMA systems
[49] Setup hugepage mappings for NUMA systems
[50] Display current Ethernet/Baseband/Crypto device settings
[51] Bind Ethernet/Baseband/Crypto device to IGB UIO module
[52] Bind Ethernet/Baseband/Crypto device to VFIO module
[53] Setup VFIO permissions

------------------------------------------------------------
 Step 3: Run test application for linux environment
------------------------------------------------------------
[54] Run test application ($RTE_TARGET/app/test)
[55] Run testpmd application in interactive mode ($RTE_TARGET/app/testpmd)

------------------------------------------------------------
 Step 4: Other tools
------------------------------------------------------------
[56] List hugepage info from /proc/meminfo

------------------------------------------------------------
 Step 5: Uninstall and system cleanup
------------------------------------------------------------
[57] Unbind devices from IGB UIO or VFIO driver
[58] Remove IGB UIO module
[59] Remove VFIO module
[60] Remove KNI module
[61] Remove hugepage mappings

[62] Exit Script

Option: 45
```

```
Option: 45

Unloading any existing DPDK UIO module
Loading DPDK UIO module

Press enter to continue ...
```

3. Bind Ethernet/Baseband/Crypto device to IGB UIO module (for this dpdk version , enter 51 to build it, Please enter the bus information of the network card that directly connected to the test machine, such as : 0000:29:00.0)

```
Option: 51


Network devices using kernel driver
===================================
0000:16:00.0 'Ethernet Controller X710 for 10GBASE-T 15ff' if=ens1f0 drv=i40e unused=igb_uio *Active*
0000:16:00.1 'Ethernet Controller X710 for 10GBASE-T 15ff' if=ens1f1 drv=i40e unused=igb_uio
0000:29:00.0 'Ethernet Controller E810-C for QSFP 1592' if=ens66 drv=ice unused=igb_uio

No 'Baseband' devices detected
==============================

Other Crypto devices
====================
0000:6b:00.0 'Device 4940' unused=igb_uio
0000:6d:00.0 'Device 2710' unused=igb_uio
0000:70:00.0 'Device 4940' unused=igb_uio
0000:72:00.0 'Device 2710' unused=igb_uio
0000:75:00.0 'Device 4940' unused=igb_uio
0000:77:00.0 'Device 2710' unused=igb_uio
0000:7a:00.0 'Device 4940' unused=igb_uio
0000:7c:00.0 'Device 2710' unused=igb_uio
0000:e8:00.0 'Device 4940' unused=igb_uio
0000:ea:00.0 'Device 2710' unused=igb_uio
0000:ed:00.0 'Device 4940' unused=igb_uio
0000:ef:00.0 'Device 2710' unused=igb_uio
0000:f2:00.0 'Device 4940' unused=igb_uio
0000:f4:00.0 'Device 2710' unused=igb_uio
0000:f7:00.0 'Device 4940' unused=igb_uio
0000:f9:00.0 'Device 2710' unused=igb_uio

No 'Eventdev' devices detected
==============================

No 'Mempool' devices detected
=============================

No 'Compress' devices detected
==============================

No 'Misc (rawdev)' devices detected
===================================

Enter PCI address of device to bind to IGB UIO driver: 0000:29:00.0
```

```
4.Setup hugepage mappings for NUMA systems (for this dpdk version , enter 49 to
build it, Set 4G size for each node)
```

```
Option: 49

Removing currently reserved hugepages
Unmounting /mnt/huge and removing directory

  Input the number of 1048576kB hugepages for each node
  Example: to have 128MB of hugepages available per node in a 2MB huge page system,
  enter '64' to reserve 64 * 2MB pages on each node
Number of pages for node0: 4
Number of pages for node1: 4
Reserving hugepages
Creating /mnt/huge and mounting as hugetlbfs
```

Modify The Configuration File

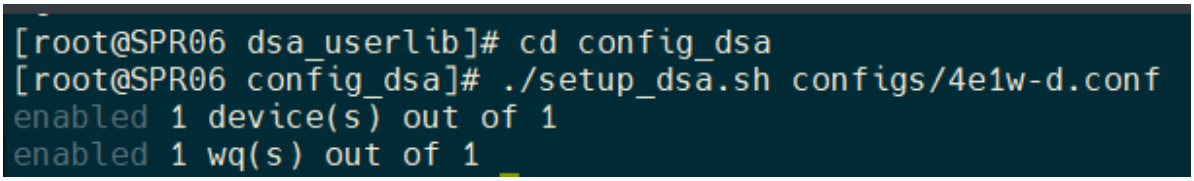
```
Enter the path where the epping program is locate:
cd /dsa-accelerated-mtcp/mtcp_merge_virtq/apps/example/
vim epping.conf
    # modify port(line 51) to your NIC PCIE number, for example:
    port = 0000:29:00.0 (your NIC PCIE number)
    # moidfy stat_print(line 105), for example:
    stat_print = 0000:29:00.0
```

Configure DSA (Reconfiguration is required when restart)

You can read README in ：

```
/dsa-accelerated-mtcp/dsa_userlib/README.md
```

```
# Configure dsa
# Make sure you have installed accel-config library and tools before doing this step
cd config_dsa
./setup_dsa.sh configs/4e1w-d.conf
```



Build Test Program "Epping"

```
# make test programe: epping
# epping :  /mtcp_merge_virtq//app/examples/epping
cd /dsa-accelerated-mtcp/mtcp_merge_virtq/
export RTE_SDK=`echo $PWD`/dpdk
export RTE_TARGET=x86_64-native-linuxapp-gcc
make -j32
```

Configure pktgen machine(machine run pktgen to send packets)

Pktgen-DPDK/INSTALL.md at dev · pktgen/Pktgen-DPDK (github.com)

1. compile DPDK:
    1. Install DPDK

        ```
        git clone https://dpdk.org/git/dpdk
        sudo rm -fr /usr/local/lib/x86_64-linux-gnu # DPDK changed a number of lib names and need to clean up
        cd dpdk
        meson build
        ninja -C build
        sudo ninja -C build install
        sudo ldconfig  # make sure ld.so is pointing new DPDK libraries
        ```

    2. Compile Pktgen

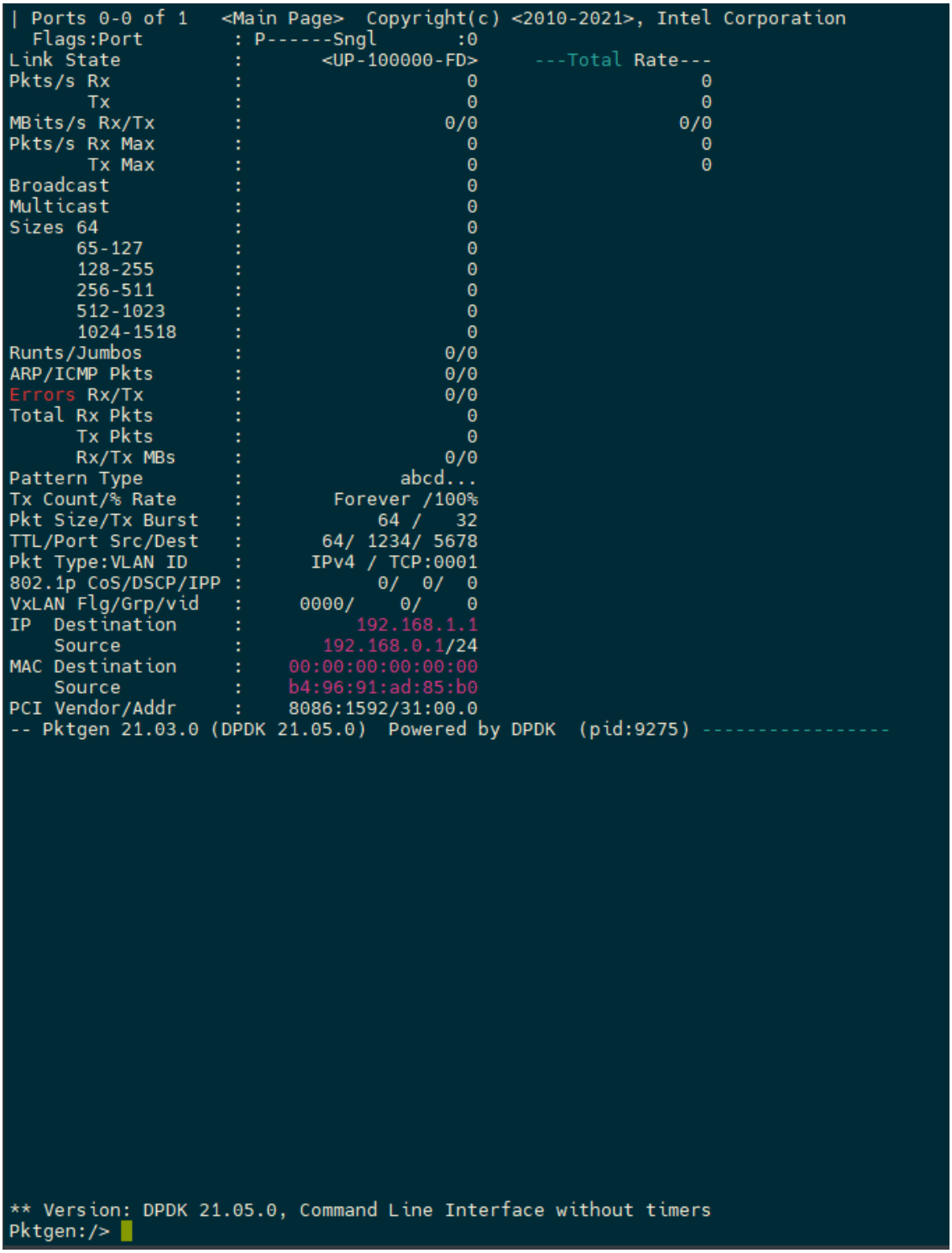
```
git clone http://dpdk.org/git/apps/pktgen-dpdk
cd pktgen-dpdk
make
or
make build    # Same as 'make'
or
make rebuild  # Rebuild Pktgen, which removes the Builddir then builds
it again via meson/ninja
```

Run Demo

pktgen machine send packets ---->----->------>------>---->---->---->---->----> demo spr machine receive packets

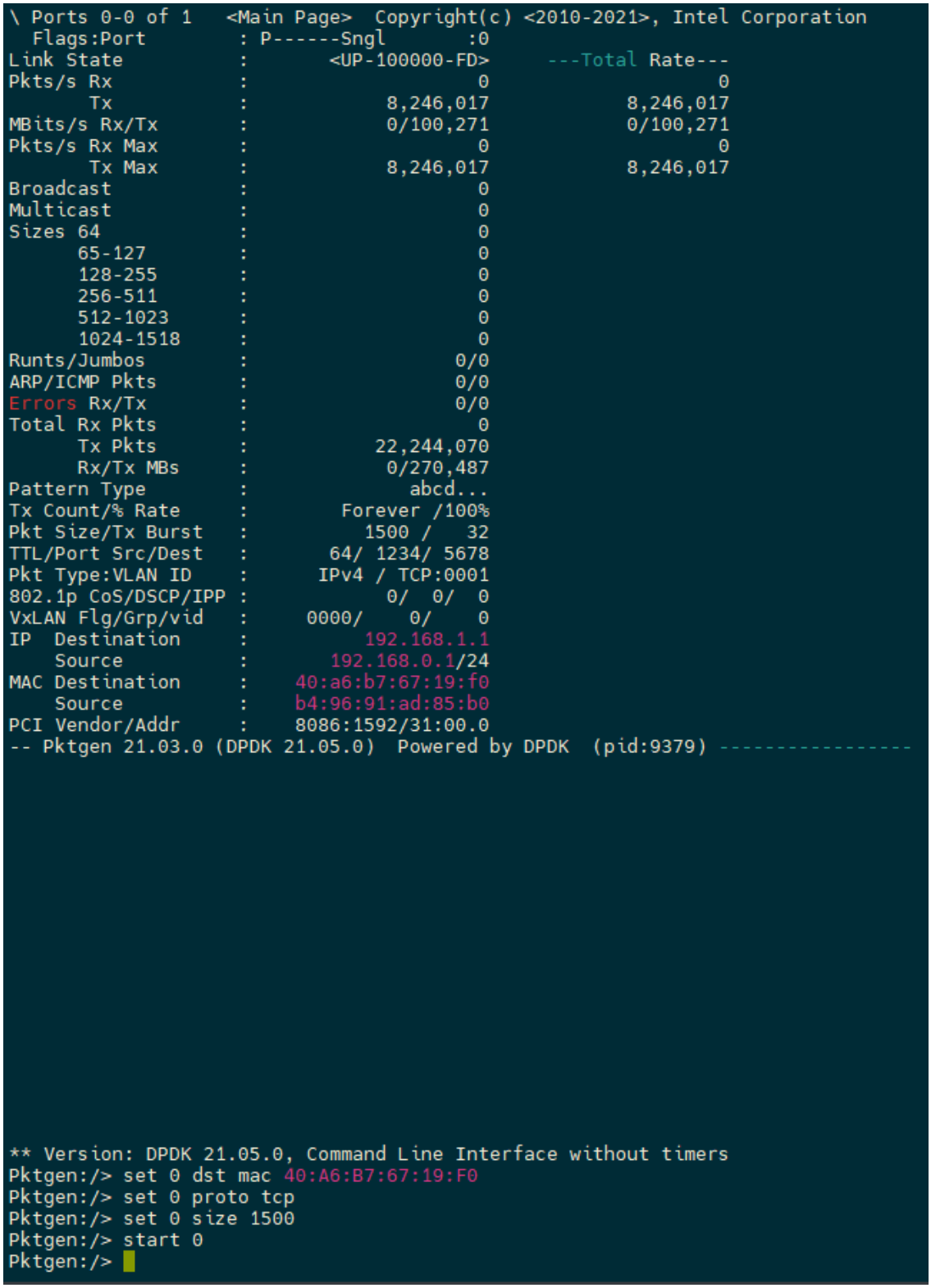
1. pktgen machine run "pktgen-dpdk" to send packets:

```
cd /pktgen-dpdk
# start pktgen
./usr/local/bin/pktgen -c 0x3 -n 2 -- -P -m "1.0" # if the prompt "Pktgen
got a Segment Fault", try again, which may be a bug of pktgen, Until the
following interface appears
```



Enter the following command to send packets:

```
# this mac address is SPR NIC mac address, which had previously recorded
set 0 dst mac 40:A6:B7:67:19:F0
set 0 proto tcp
set 0 size 9000
start 0 rate 0.01
# if you want pause, you can enter:
stop 0
```



2. demo machine:

```
# start test produce
cd /dsa-accelerated-mtcp/mtcp_merge_virtq/apps/example
./epping -f epping.conf
```

Then observe whether pktgen and Epping have data flow.

```
--------------------------------------------------------------------------------
Loading ARP table from : config/arp.conf
ARP Table:
IP addr: 192.168.18.124, dst_hwaddr: B4:96:91:AD:85:B1
--------------------------------------------------------------------------------
Initializing port 0... done:
[dpdk_load_module: 686] Failed to get flow control info!
[dpdk_load_module: 693] Failed to set flow control info!: errno: -95

Checking link statusdone
Port 0 Link Up - speed 100000 Mbps - full-duplex
***********************************DSA_INIT*****************************
[ info] alloc wq 0 shared 0 size 128 addr 0x7ffa2ab80000 batch sz 0x400 xfer sz 0x80000000
***********************************CORE INIT DSA_DONE*****************************
CPU 0: initialization finished.
dsa check count : 0
[mtcp_create_context:1448] CPU 0 is now the master thread.
dsa check count : 10000000
dsa check count : 20000000
dsa check count : 30000000
dsa check count : 40000000
dsa check count : 50000000
dsa check count : 60000000
TCP recved PPS:139.91
write  cnt 521402
failed cnt 513000
dsa check count : 70000000
cpu memcpy len :1048576 cost 162826 cycles
dsa check count : 80000000
write  cnt 561574
failed cnt 552655
TCP recved PPS:138.21
cpu memcpy len :1048576 cost 149142 cycles
dsa check count : 90000000
write  cnt 561153
failed cnt 552240
```

4. Start Test

You must start epping first. and then start pktgen.

Sample Run

```
# Use  memmove transfer size of 1m with 128 descriptors
./epping -f epping.conf -l 1m  -n 128
```

The different **-n** parameter is mainly to not enter the cache when test specific transfer size(**-l** parameter).

benchmark with CPU

```
cd /dsa-accelerated-mtcp/mtcp_merge_virtq/apps/example
# 4K
./epping -f epping.conf -l 4k  -n 32768
# 8K
./epping -f epping.conf -l 8k  -n 16384
# 16K
./epping -f epping.conf -l 16k  -n 8192
# 32K
./epping -f epping.conf -l 32k  -n 4096
# 64K
./epping -f epping.conf -l 64k  -n 2048
# 128K
./epping -f epping.conf -l 128k -n 1024
# 256K
./epping -f epping.conf -l 256k -n 512
```

```
# 512K
./epping -f epping.conf -l 512k -n 256
# 1M
./epping -f epping.conf -l 1m  -n 128
```

benchmark with DSA

```
# The performance of different parameters is different. The following is a
combination of parameters with better performance
cd /dsa-accelerated-mtcp/mtcp_merge_virtq/apps/example
# 4K
./epping -f epping.conf -l 4k  -n 32768 -d
# 8K
./epping -f epping.conf -l 8k  -n 16384 -d
# 16K
./epping -f epping.conf -l 16k  -n 8192 -d
# 32K
./epping -f epping.conf -l 32k  -n 4096 -d
# 64K
./epping -f epping.conf -l 64k  -n 2048 -d
# 128K
./epping -f epping.conf -l 128k -n 1024 -d
# 256K
./epping -f epping.conf -l 256k  -n 512 -d
# 512K
./epping -f epping.conf -l 512k  -n 256 -d
# 1M
./epping -f epping.conf -l 1m  -n 128 -d
```

5.Measurement

PPS Measurement

```
# You can observe PPS in PKTGEN
```

```
/ Ports 0-0 of 1   <Main Page>  Copyright(c) <2010-2021>, Intel Corporation
  Flags:Port        : P------Sngl       :0
Link State          :       <UP-100000-FD>     ---Total Rate---
Pkts/s Rx           :              571,324             571,324
       Tx           :                  160                 160
MBits/s Rx/Tx       :           37,563/11           37,563/11
Pkts/s Rx Max       :              583,994             583,994
       Tx Max       :                  160                 160
Broadcast           :                    0
Multicast           :                    0
Sizes 64            :                    0
      65-127        :                    0
      128-255       :                    0
      256-511       :                    0
      512-1023      :                    0
      1024-1518     :                    0
Runts/Jumbos        :         0/62,138,033
ARP/ICMP Pkts       :                  0/0
Errors Rx/Tx        :                  0/0
Total Rx Pkts       :           61,567,071
      Tx Pkts       :               10,688
      Rx/Tx MBs     :        4,048,712/771
Pattern Type        :               abcd...
Tx Count/% Rate     :       Forever /0.01%
Pkt Size/Tx Burst   :          9000 /   32
TTL/Port Src/Dest   :       64/ 1234/ 5678
Pkt Type:VLAN ID    :      IPv4 / TCP:0001
802.1p CoS/DSCP/IPP :            0/  0/  0
VxLAN Flg/Grp/vid   :     0000/    0/    0
IP  Destination     :          192.168.1.1
    Source          :       192.168.0.1/24
MAC Destination     :    40:a6:b7:67:19:f0
    Source          :    b4:96:91:ad:85:b0
PCI Vendor/Addr     :    8086:1592/31:00.0
-- Pktgen 21.03.0 (DPDK 21.05.0)  Powered by DPDK  (pid:21903) ----------------
Pktgen:/> stop 0
Pktgen:/> start 0
Pktgen:/> stop 0
Pktgen:/> start 0
```

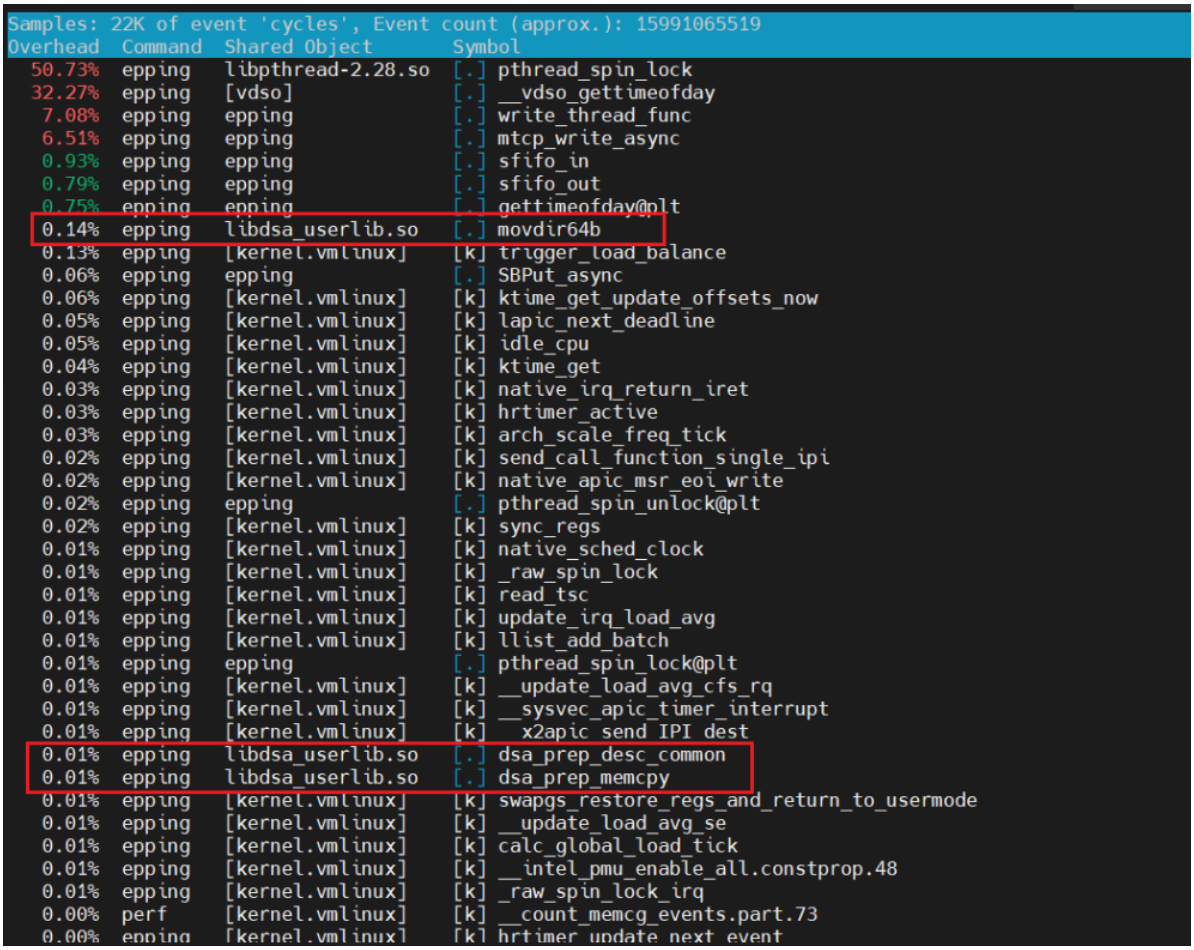
CPU Usage

```
# Copy the CPU occupied by the actual operation without DSA
perf record -C 7
perf report -i perf.data
```

```
Samples: 15K of event 'cycles', Event count (approx.): 9954117485
Overhead  Command  Shared Object       Symbol
  47.87%  epping   libpthread-2.28.so  [.] pthread_spin_lock
  37.58%  epping   libc-2.28.so        [.] __memmove_avx_unaligned_erms
   3.41%  epping   epping              [.] sfifo_out
   3.34%  epping   epping              [.] mtcp_write
   3.11%  epping   epping              [.] sfifo_in
   3.07%  epping   [vdso]              [.] __vdso_gettimeofday
   0.69%  epping   epping              [.] write_thread_func
   0.31%  epping   epping              [.] SBPut
   0.10%  epping   libpthread-2.28.so  [.] __errno_location
   0.08%  epping   epping              [.] pthread_spin_lock@plt
   0.08%  epping   epping              [.] gettimeofday@plt
   0.07%  epping   epping              [.] pthread_spin_unlock@plt
   0.05%  epping   libpthread-2.28.so  [.] pthread_spin_unlock
   0.03%  epping   epping              [.] __errno_location@plt
   0.03%  epping   [kernel.vmlinux]    [k] sync_regs
   0.03%  epping   [kernel.vmlinux]    [k] hrtimer_active
   0.02%  epping   [kernel.vmlinux]    [k] perf_event_task_tick
   0.02%  epping   [kernel.vmlinux]    [k] __intel_pmu_enable_all.constprop.48
   0.01%  epping   [kernel.vmlinux]    [k] ktime_get
   0.01%  epping   [kernel.vmlinux]    [k] __sysvec_apic_timer_interrupt
   0.01%  epping   [kernel.vmlinux]    [k] lapic_next_deadline
   0.01%  perf     perf                [.] cmd_record
   0.01%  epping   [kernel.vmlinux]    [k] rb_next
   0.01%  epping   [kernel.vmlinux]    [k] trigger_load_balance
   0.01%  epping   [kernel.vmlinux]    [k] scheduler_tick
   0.01%  epping   [kernel.vmlinux]    [k] __update_load_avg_cfs_rq
   0.01%  epping   [kernel.vmlinux]    [k] __acct_update_integrals
   0.01%  epping   [kernel.vmlinux]    [k] account_user_time
   0.01%  epping   [kernel.vmlinux]    [k] ktime_get_update_offsets_now
   0.01%  epping   [kernel.vmlinux]    [k] calc_global_load_tick
   0.00%  perf     [kernel.vmlinux]    [k] _perf_ioctl
   0.00%  perf     [kernel.vmlinux]    [k] native_apic_msr_write
   0.00%  perf     [kernel.vmlinux]    [k] smp_call_function_single
```

CPU Usage with DSA

```
The CPU occupied by actually initiating the DSA copy (sum of three values)
```



Bandwidth Measurement

1. Peak value measurement

```
peakValue =  sizeof ( Total packets sent) / latency
```

2. Perf measurement

```
perf stat -e
dsa0/event=0x1,event_category=0x1/,dsa0/event=0x2,event_category=0x1/ -a -I 1000
```

The final result is the counts value * 32 = 195405108 * 32