DPDK Based iperf for Layer 4 DPDK Applications / Load Balancers

M JAY
PLATFORM APPLICATION ENGINEER
Agenda

• The Benefit of DPDK-iperf
• Why do you need it?
• ANS – Accelerated Network Stack
• DPDK-iperf – without a switch in between the server and client
• DPDK-iperf – with a switch in between the server and client
• Call To Action

https://github.com/ansyun/dpdk-iperf
L2fwd, L3fwd - Stateless

- We have all measured DPDK l2fwd, l3fwd traffic
  - Hardware Traffic Generator
  - Software Traffic Generator
    - PktGen
    - Trex
    - Ostinato

- When we go up the stack – stateful generator needed
  - Question- Which stateful Software Traffic Generators you have used?

So, which stateless tool have you used?
Kernel space tools – Bottleneck in the tool

- Your DPDK based Layer 4 appliances
  - Layer 4 Load Balancer - DPVS
  - What other Layer 4 appliances you have built?

- What s/w generators to use?

- Kernel space generators
  - Most popular tools
    - Iperf
    - netperf
  - But, bottleneck in the tool for measuring user space appliances

So, which stateless tool do you recommend?
DPDK-ANS Architecture

ANS Application
- nginx
- http Server
- iperf
- redis
- https server

Linux Application
- OSPF/BGP
- BFD/VRRP/DHCP

ANS (Fast Path)
- Socket
- TCP/UDP
- IP/Routing
- ICMP/ACL
- IP Reassembly
- Ethernet/ARP

Linux Kernel (Slow Path)
- TCP/UDP
- IP/Routing/ICMP
- Ethernet/ARP

DPDK
- rte_eal
- rte_malloc
- rte_ring
- rte_timer
- rte_ether
- net driver
- rte_mempool
- rte_mbuf

Linux Driver
- ixgbe/i40e
- kni

SYNC
KNI
Dpdk-iperf

- **Step 1:**
  - Clone and Compile DPDK
  - Git checkout stable Release
  - Configure DPDK
  - Install DPDK
Accelerated Network Stack (ANS)

- Step 2:
  - Set Env variables
  - Git clone ANS
Install_deps.sh - accelerated Network Stack

- Step 3:
- Set env for ANS
- Install_deps.sh
- Look for march related message

```
[root@localhost dpdk-iperf]# #******************************************************************************
[root@localhost dpdk-iperf]# # Export env for ANS (Advanced Network Stack)
[root@localhost dpdk-iperf]# #******************************************************************************
[root@localhost dpdk-iperf]# cd dpdk-ans/
[root@localhost dpdk-ans]# p d
/home/user/dpdk-iperf/dpdk-ans
[root@localhost dpdk-ans]# export RTE_ANS=$PWD
[root@localhost dpdk-ans]#
[root@localhost dpdk-ans]# echo $RTE_ANS
/home/user/dpdk-iperf/dpdk-ans

[root@localhost dpdk-ans]# ls
ans  doc  install_deps.sh  librte_anscli  README.md
cli_examples  librte_ans  librte_anssock  test

[root@localhost dpdk-ans]# #******************************************************************************
[root@localhost dpdk-ans]# # /install_deps.sh
[root@localhost dpdk-ans]# #******************************************************************************

Start to generate librte_ans.a/librte_anssock.a/librte_anscli.a for -mnative.
Your native march is core-avx2.
Your native march is none of westmere/sandybridge/ivybridge/haswell/broadwell/knl.
You may generate librte_ans.a/librte_anssock.a based on your -mnative by manual, or ask help from ANS develop team.

westmere: Intel Westmere CPU with 64-bit extensions, MMX, SSE, SSE2, SSE3, SSSE3, SSE4.2, POPCNT, AES and PCLMUL instruction set support.
sandybridge: Intel Sandy Bridge CPU with 64-bit extensions, MMX, SSE, SSE2, SSE3, SSSE3, SSSE4, POPCNT, AVX, AES and PCLMUL instruction set support.
ivybridge: Intel Ivy Bridge CPU with 64-bit extensions, MMX, SSE, SSE2, SSE3, SSSE3, SSSE4, POPCNT, AVX2, AES, PCLMUL, FSGSBASE, RDRND and F16C instruction set support.
broadwell: Intel Broadwell CPU with 64-bit extensions, MOVBE, MMX, SSE, SSE2, SSE3, SSSE3, SSSE4, POPCNT, AVX, AVX2, AES, PCLMUL, FSGSBASE, RDRND, FMA, BMI, BMI2, F16C, RDSEED, ADX and PREFETCHW instruction set support.
knl: Intel Knights Landing CPU with 64-bit extensions, MOVBE, MMX, SSE, SSE2, SSE3, SSSE3, SSSE4, SSSE4.2, POPCNT, AVX, AVX2, AES, PCLMUL, FSGSBASE, RDRND and F16C instruction set support.
haswell: Intel Haswell CPU with 64-bit extensions, MOVBE, MMX, SSE, SSE2, SSE3, SSSE3,
DPDK-ANS – three libs, cli, ans executable

- Step 4: Build DPDK-ANS
- Lookout for 3 files
  - Librte_ans.a
  - Librte_anscli.a
  - Librte_anssock.a
You should have installed binutils

- march – machine arch
- DPDK Prog. Manual
  - Second half sections
- Any suggestions?
- Brute force copy? – all 3 .a files
- Successful make !!

```
[root@localhost dpdk-ans]# cd ans
[root@localhost ans]# ls
ans_kni.c  ans_main.c  ans_param.c  build
ans_kni.h  ans_main.h  ans_param.h  Makefile
[root@localhost ans]# make
LD ans
INSTALL-APP ans
INSTALL-MAP ans.map
[root@localhost ans]# make clean
[root@localhost ans]# make
CC ans_param.o
CC ans_main.o
CC ans_kni.o
LD ans
INSTALL-APP ans
```
Get Started To Run Accelerated Network Stack

- Step 5
- Check HugePage
- Bind Port of Interest
- Run Accelerated Network Stack

```
[root@localhost ans]# #***************************************************************************
[root@localhost ans]# # 1) Check huge page
[root@localhost ans]# # 2) Bind the port of interest to igb_uio
[root@localhost ans]# #***************************************************************************
[root@localhost ans]# cat /proc/meminfo | grep Huge
AnonHugePages: 616448 kB
HugePages_Total: 1024
HugePages_Free: 1024
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB
```

```
[root@localhost usertools]# [root@localhost usertools]# modprobe uio
[root@localhost kmod]# insmod igb_uio.ko
[root@localhost kmod]# pwd
/home/user/dpdk-iperf/dpdk/x86_64-native-linuxapp-gcc/kmod

[root@localhost usertools]# ./dpdk-devbind.py --unbind 0000:17:00.1
```

```
# ./dpdk-devbind.py --bind=igb_uio 0000:17:00.1
```

```
# sudo ./build/ans -c 0x2 -n 1 --p 0x1 --config="(0,0,1)"
```
Open Second Terminal & Run ANS CLI

- Step 6 – Leave first terminal as it is.
- Now open New terminal to start CLI. Why start CLI?
- Client and Server will have same IP address
- Need to change one of that. Hence CLI.

Note:
* Use “ip addr show” to see IP address of server / client
  * To change the IP Address – to keep addresses unique
  * do “ip add del 10.0.0.2/24 dev veth0”
  * Followed by “ip addr add 10.0.0.2/24 dev veth0”
  * Verify with “ip addr show”
Open Third Terminal for dpdk-iperf

- Step 7:
  - Git clone dpdk-iperf
  - Make dpdk-iperf
  - Start Server in one machine
  - Start client in another machine

```
[root@localhost dpdk-iperf]# ***********************
[root@localhost dpdk-iperf]# # make dpdk-iperf
[root@localhost dpdk-iperf]# ***********************
[root@localhost dpdk-iperf]# # make dpdk-iperf
[root@localhost dpdk-iperf]# ***********************
[root@localhost dpdk-iperf]# gcc -D HAVE_DPDK_ANS -c -o src/iperf_client_api_dpdk.o
[root@localhost dpdk-iperf]# gcc -D HAVE_DPDK_ANS -c -o src/iperf_client_api_dpdk.o
```

```
[root@localhost dpdk-iperf]# ./dpdk_iperf3 -s --bind 10.0.0.2
EAL: Detected 16 cores(s)
EAL: Detected 1 NUMA nodes
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket_14033_29278183519b
EAL: Probing VFIO support...
USERB: LCORE[1] anssock any lcore id 0xffffffff
USERB: LCORE[1] anssock app id: 14033
USERB: LCORE[1] anssock app name: dpdk_iperf3
USERB: LCORE[1] anssock app lcoreId: 1
USERB: LCORE[1] mp ops number 4, mp ops index: 0
USERB: LCORE[1] setsockopt: not support optname 2

Server listening on 5201
```

```
dpdk_iperf3 # start dpdk_iperf client
```

```
adler: 400 ops number 4, mp ops index: 0
encl:getfl: Bad file descriptor
iperf: error: unable to receive control message: Resource temporarily unavailable
```
Directly connection – No Switch – in between Server & Client

- Works Out Of The Box
- Default Server and Client will have same address
- Change IP address for one of them
- So they can be in same subnet each with its own unique IP address

- Delete the default IP address
- Add new IP address *in same subnet and mask*
Bad File Descriptor – How to take care of it?

1) Make sure client & server are at different IP addr -
   • Through ANS CLI

2) Works without switch in between.

Suggestions To Make it work with Switch
   • Enable VLAN?ans_main.c
     ➢ How?
     ➢ In ans_main.c

```c
.rxmode =
{
   .mq_mode = ETH_MQ_RX_RSS,
   .max_rx_pkt_len = ETHER_MAX_LEN,
   .split_hdr_size = 0,
   .offloads = DEV_RX_OFFLOAD_CHECKSUM | DEV_RX_OFFLOAD_VLAN_STRIP,
},
```
Dpdk-iperf output
• Unidirectional - 5.76 Gbytes transfer
• 4.90 Gbits/sec bandwidth
Acknowledgements

- [https://github.com/ansyun/dpk-dpk-iperf](https://github.com/ansyun/dpk-dpk-iperf)
- Excellent Support. Thanks to anssupport [anssupport@163.com](mailto:anssupport@163.com)
  - Even on 3 days weekend – everyday got reply from support
ANS library, ANS CLI and DPDK-IPerf
Backup
When You Have Switch in between Server & Client

- Enable VLAN
- Change code in ans_main.c

```c
.rxmode =
{
  .mq_mode = ETH_MQ_RX_RSS,
  .max_rx_pkt_len = ETHER_MAX_LEN,
  .split_hdr_size = 0,
  .offloads = DEV_RX_OFFLOAD_CHECKSUM | DEV_RX_OFFLOAD_VLAN_STRIP,
},
```
Contact
M Jay
Muthurajan.Jayakumar@intel.com