**V P P**

**( Vector Packet Processing )**



Contents

[**I. Introduction** 4](#_Toc3966664)

[1.1 VPP architecture 4](#_Toc3966665)

[1.2 Huge Pages 4](#_Toc3966666)

[**II. Build VPP from source** 5](#_Toc3966667)

[2.1 download VPP source with https 5](#_Toc3966668)

[2.2 compile VPP 5](#_Toc3966669)

[**III. Run VPP** 5](#_Toc3966670)

[3.1 configure VPP 5](#_Toc3966671)

[3.2 run VPP 6](#_Toc3966672)

[3.3 configure vpp interface 6](#_Toc3966673)

[3.4 ping test 7](#_Toc3966674)

[3.5 build rpm package 8](#_Toc3966675)

[3.6 install VPP with rpm package 8](#_Toc3966676)

[3.7 running without GDB 9](#_Toc3966677)

[3.8 running with GDB 9](#_Toc3966678)

[**IV. VPP performance** 9](#_Toc3966679)

[4.1 iperf3 9](#_Toc3966680)

[**V. Appendix** 10](#_Toc3966681)

[5.1 references 10](#_Toc3966682)

[5.2 errors 10](#_Toc3966683)

[5.3 Install devtoolset-7 on arm(2019.03.01) 11](#_Toc3966684)

[5.4 compile VPP with mellanox NIC 11](#_Toc3966685)

[5.5 compile VPP stable/1901 12](#_Toc3966686)

[5.6 startup.conf parameters 12](#_Toc3966687)

***\* Release note \****

|  |  |  |
| --- | --- | --- |
| ***version*** | ***release date*** | ***content*** |
| *v1.0* | *2019-03-08* | *first edition* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***\* Release Copyleft©free \****

**I. Introduction**

The VPP platform is an extensible framework that provides out-of-the-box production quality switch/router functionality. It is the open source version of Cisco’s Vector Packet Processing(VPP) technology: a high performance, packet-processing stack that can run on commodity CPUs.

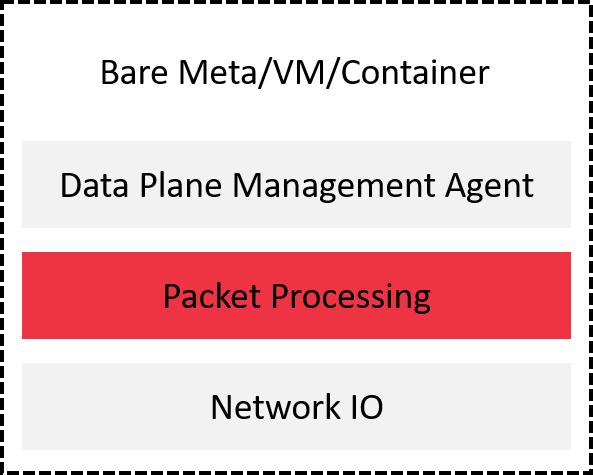


Figure VPP in network stack overview

1.1 VPP architecture

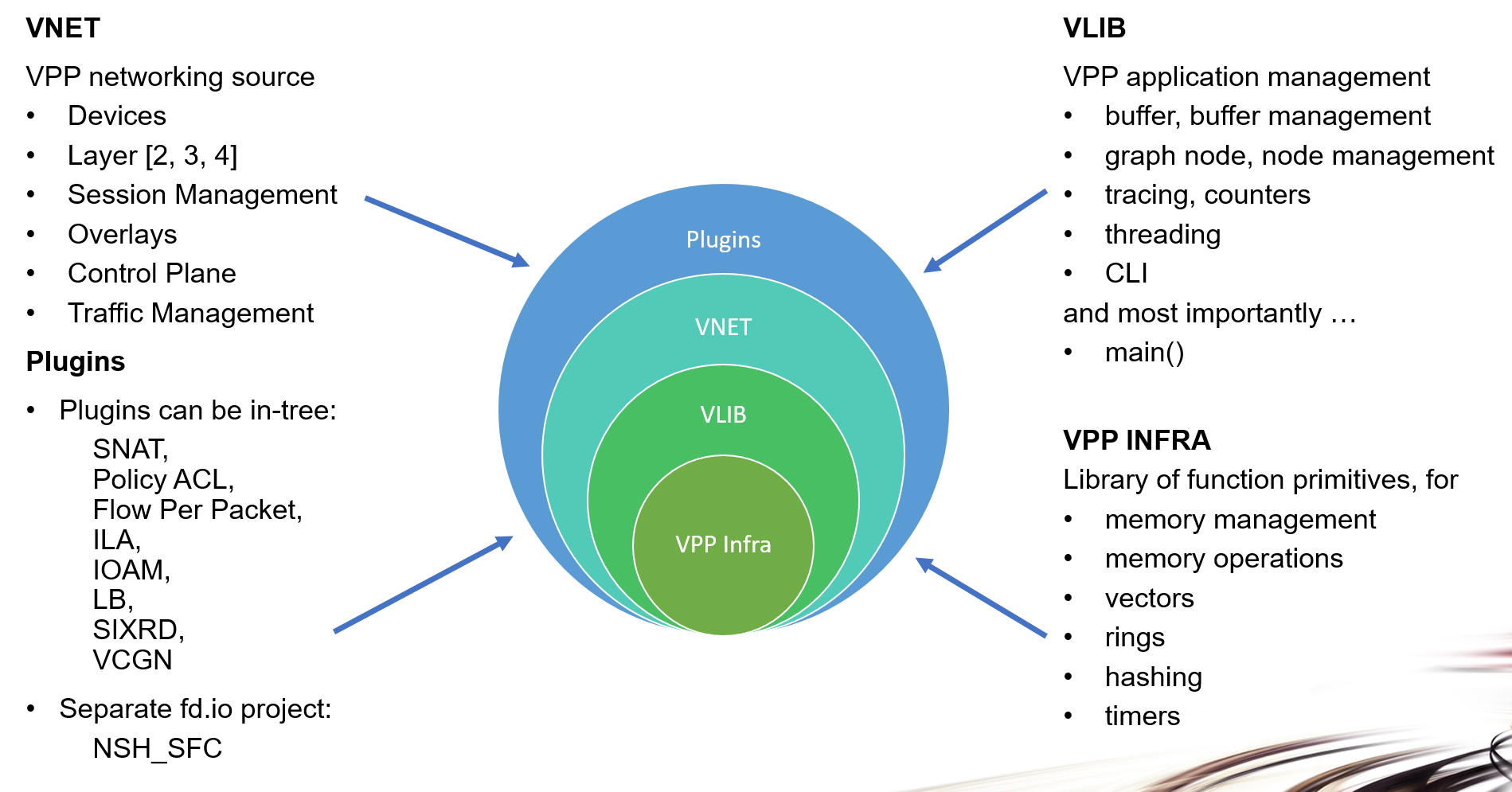


Figure VPP layers

1.2 Huge Pages

During VPP installation. VPP will overwrite the existing hugepage settings. By default, VPP sets the number of hugepages on a system to 1024 2M hugepages. This is the number of hugepages on the system, not just used by VPP. the settings are included in /etc/sysctl.d/80-vpp.conf, [reference here](https://fdio-vpp.readthedocs.io/en/latest/gettingstarted/users/configuring/hugepages.html) to know more about huge pages. you can update your configuration depending on how the system is being used.

|  |  |
| --- | --- |
|  | If VPP is being run in a Virtual Machine, the VM must have hugepage backing. If not, the install will fail, but the failure may go unnoticed. When the system reboot again. ‘vm.nr\_hugepages’ will be reapplied and will fail, and the VM will abort kernel boot, locking up the VM. |

**II. Build VPP from source**

2.1 download VPP source with https

|  |
| --- |
| # git clone https://gerrit.fd.io/r/vpp |

2.2 compile VPP

|  |
| --- |
| # make install-dep  # make wipe  # make wipe-release  //reference how to [install devtoolset-7 on arm](#install_devtoolset7_on_arm)  # scl --list  devtoolset-7  # scl enable devtoolset-7 bash  # gcc -v  …  gcc version 7.3.1 20180303 (Red Hat 7.3.1-5) (GCC)  …  # make build-release  // Waiting for compilation to complete and there is no error occurred. |

**III. Run VPP**

3.1 configure VPP

|  |
| --- |
| # cd $VPP\_ROOT/src/vpp/conf  # mkdir -p /etc/vpp  # mkdir -p /var/log/vpp  # groupadd vpp  # cp startup.conf /etc/vpp/  # cp 80-vpp.conf /etc/sysctl.d/  # lspci | grep Ethernet  …  0004:01:00.0 Ethernet controller: Mellanox Technologies MT27700 Family [ConnectX-4]  …  # cat /etc/vpp/startup.conf  …  unix {  nodaemon  log /var/log/vpp/vpp.log  full-coredump  cli-listen /run/vpp/cli.sock  gid vpp  }  api-trace {  on  }  api-segment {  gid vpp  }  socksvr {  default  }  cpu {  }  dpdk {  dev 0004:01:00.0 {  num-rx-queues 4  }  }  … |

3.2 run VPP

|  |
| --- |
| # cd $VPP\_ROOT/build-root/build-vpp-native/vpp  # T=`pwd`  # rm -f /dev/shm/db /dev/shm/global\_vm /dev/shm/vpe-api || return  # modprobe uio\_pci\_generic || return  # $T/bin/vpp -c /etc/vpp/startup.conf |

3.3 configure vpp interface

|  |
| --- |
| # cd $T/bin  # ./vppctl show interface  Name Idx State MTU (L3/IP4/IP6/MPLS) Counter Count  TenGigabitEthernet1/0/0 1 down 9000/0/0/0  local0 0 down 0/0/0/0  # ./vppctl set interface ip address TenGigabitEthernet1/0/0 192.168.2.51/24  # ./vppctl show interface address  TenGigabitEthernet1/0/0 (dn):  L3 192.168.2.51/24  local0 (dn):  # ./vppctl set interface state TenGigabitEthernet1/0/0 up  # ./vppctl show interface  Name Idx State MTU (L3/IP4/IP6/MPLS) Counter Count  TenGigabitEthernet1/0/0 1 up 9000/0/0/0 rx packets 370  rx bytes 54398  tx packets 5  tx bytes 482  drops 367  ip4 49  ip6 82  local0 0 down 0/0/0/0 |

3.4 ping test

|  |
| --- |
| // configure a VPP server  Name Idx State MTU (L3/IP4/IP6/MPLS) Counter Count  FiftySixGigabitEthernet1/0/0 1 down 9000/0/0/0  local0 0 down 0/0/0/0  # ./vppctl set interface ip address FiftySixGigabitEthernet1/0/0 192.168.4.51/24  # ./vppctl show interface address  FiftySixGigabitEthernet1/0/0 (dn):  L3 192.168.4.51/24  local0 (dn):  # ./vppctl set interface state FiftySixGigabitEthernet1/0/0 up  # ./vppctl show interface  Name Idx State MTU (L3/IP4/IP6/MPLS) Counter Count  FiftySixGigabitEthernet1/0/0 1 up 9000/0/0/0 rx packets 3  rx bytes 180  drops 3  local0 0 down 0/0/0/0  //configure for another VPP  # ./vppctl show interface  Name Idx State MTU (L3/IP4/IP6/MPLS) Counter Count  FiftySixGigabitEthernet1/0/0 1 down 9000/0/0/0  local0 0 down 0/0/0/0  # ./vppctl set interface ip address FiftySixGigabitEthernet1/0/0 192.168.4.48/24  # ./vppctl show interface address  FiftySixGigabitEthernet1/0/0 (dn):  L3 192.168.4.48/24  local0 (dn):  # ./vppctl set interface state FiftySixGigabitEthernet1/0/0 up  # ./vppctl show interface  Name Idx State MTU (L3/IP4/IP6/MPLS) Counter Count  FiftySixGigabitEthernet1/0/0 1 up 9000/0/0/0 rx packets 2  rx bytes 120  drops 2  local0 0 down 0/0/0/0  //ping each other  # ./vppctl ping 192.168.4.51  116 bytes from 192.168.4.51: icmp\_seq=1 ttl=64 time=.0360 ms  …  Statistics: 5 sent, 5 received, 0% packet loss  # ./vppctl ping 192.168.4.48  116 bytes from 192.168.4.48: icmp\_seq=1 ttl=64 time=.0471 ms  …  Statistics: 5 sent, 5 received, 0% packet loss |

3.5 build rpm package

|  |
| --- |
| # make pkg-rpm  // the rpm packages will be created in the directory $VPP\_ROOT/build-root/ |

3.6 install VPP with rpm package

|  |
| --- |
| # rpm -ivh $VPP\_ROOT/build-root/\*.rpm  # systemctl start vpp  # systemctl status vpp |

3.7 running without GDB

|  |
| --- |
| # cd $VPP\_ROOT  // run the release image  # make run-release  //run the debug ima  # make run |

3.8 running with GDB

|  |
| --- |
| # cd $VPP\_ROOT  // run the release image  # make debug-release  (gdb) run -c /etc/vpp/startup.conf  // run the debug image  # make debug  (gdb) run -c /etc/vpp/startup.conf |

**IV. VPP performance**

4.1 iperf3

|  |
| --- |
| // this needs to configure for both vpp in the file startup.conf  session {  evt\_qs\_memfd\_seg  }  socksvr {  socket-name /tmp/vpp-api.sock  }  // add new configuration file vcl.conf  # vim /etc/vpp/vcl.conf  …  vcl {  rx-fifo-size 4000000  tx-fifo-size 4000000  app-scope-local  app-scope-global  api-socket-name /tmp/vpp-api.sock  }  …  # VCL\_CFG=/etc/vpp/vcl.conf  # LDP\_PATH=$VPP\_ROOT/build-root/build-vpp-native/vpp/lib/libvcl\_ldpreload.so  //start server  # LD\_PRELOAD=$LDP\_PATH VCL\_CONFIG=$VCL\_CFG taskset --cpu-list 16 iperf3 -4 -s  -----------------------------------------------------------  Server listening on 5201  //start client  # LD\_PRELOAD=$LDP\_PATH VCL\_CONFIG=$VCL\_CFG taskset --cpu-list 15 iperf3 -c 192.168.4.48  Connecting to host 192.168.4.48, port 5201  [ 33] local 192.168.4.51 port 7579 connected to 192.168.4.48 port 5201  [ ID] Interval Transfer Bandwidth Retr Cwnd  [ 33] 0.00-1.00 sec 1.01 GBytes 8.64 Gbits/sec 0 0.00 Bytes  …  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bandwidth Retr  [ 33] 0.00-10.00 sec 10.1 GBytes 8.63 Gbits/sec 0 sender  [ 33] 0.00-10.00 sec 10.0 GBytes 8.63 Gbits/sec receiver  iperf Done.  vl\_client\_disconnect:330: queue drain: 573  vl\_client\_disconnect:330: queue drain: 587 |

**V. Appendix**

5.1 references

[1] [VPP/HostStack](https://wiki.fd.io/view/VPP/HostStack)

[2] [How to Build VPP FD.IO 18.07/18.10 With Mellanox DPDK PMD on RHEL 7.5 Without Using OFED](https://community.mellanox.com/s/article/How-to-Build-VPP-FD-IO-18-07-18-10)

[3] [VPP/Pulling, Building, Running, Hacking and Pushing VPP Code](https://wiki.fd.io/view/VPP/Pulling,_Building,_Running,_Hacking_and_Pushing_VPP_Code)

[4] [Huge Pages](https://fdio-vpp.readthedocs.io/en/latest/gettingstarted/users/configuring/hugepages.html)

5.2 errors

|  |
| --- |
| **E:**  net\_mlx5: cannot load glue library: librte\_pmd\_mlx5\_glue.so.18.11.0: cannot open shared object file: No such file or directory  net\_mlx5: cannot initialize PMD due to missing run-time dependency on rdma-core libraries (libibverbs, libmlx5)  **S:**  # make dpdk-install-dev DPDK\_MLX5\_PMD=y DPDK\_MLX5\_PMD\_DLOPEN\_DEPS=y  # cp /opt/vpp/external/x86\_64/lib/librte\_pmd\_mlx5\_glue.so\* /usr/lib64/  https://community.mellanox.com/s/article/How-to-Build-VPP-FD-IO-18-07-18-10  **E:**  build-vpp-native/vpp/bin/vpp[22584]: register\_node:477: process stack: Invalid argument (errno 22)  **S:**  vim $VPP\_ROOT/ src/vlib/node.h  …  619 #define PAGE\_SIZE\_MULTIPLE 0x10000  …  **E:**  load\_one\_plugin:145: /home/cpu/VPP/vpp/build-root/build-vpp-native/vpp/lib/vpp\_plugins/nsh\_plugin.so: undefined symbol: nsh\_adj\_incomplete  load\_one\_plugin:146: Failed to load plugin 'nsh\_plugin.so'  **S:** |

5.3 Install devtoolset-7 on arm(2019.03.01)

|  |
| --- |
| # yum -y install yum-utils  # yum install centos-release-scl-rh  # yum-config-manager --enable centos-sclo-rh-testing  // comment centos-sclo-rh repo  # cat /etc/yum.repos.d/CentOS-SCLo-scl-rh.repo  …  # [centos-sclo-rh]  # name=CentOS-7 - SCLo rh  # baseurl=http://mirror.centos.org/centos/7/sclo/$basearch/rh/  # gpgcheck=1  # enabled=1  # gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-SCLo  …  # yum install devtoolset-7 |

5.4 compile VPP with mellanox NIC

|  |
| --- |
| # wget http://content.mellanox.com/ofed/MLNX\_OFED-4.5-1.0.1.0/MLNX\_OFED\_LINUX-4.5-1.0.1.0-rhel7.5alternate-aarch64.tgz  # tar -zxvf MLNX\_OFED\_LINUX-4.5-1.0.1.0-rhel7.5alternate-aarch64.tgz  # cd MLNX\_OFED\_LINUX-4.5-1.0.1.0-rhel7.5alternate-aarch64  # ./mlnxofedinstall --dpdk --upstream-libs --add-kernel-support  # cd $VPP\_ROOT  # make dpdk-install-dev DPDK\_MLX5\_PMD=y  # make build-release  # make pkg-rpm vpp\_uses\_dpdk\_mlx5\_pmd=yes  note: kernel-devel package can find [here](http://swlab004/centos7/); |

5.5 compile VPP stable/1901

|  |
| --- |
| # git clone -b stable/1901 https://gerrit.fd.io/r/vpp  # mv vpp vpp1901  # cd vpp1901  # make install-dep  # make wipe  # make wipe-release  # scl enable devtoolset-7 bash  # gcc -v  …  gcc version 7.3.1 20180303 (Red Hat 7.3.1-5) (GCC)  …  //ensure there is no VPP package installed  // for mellanox NIC  # make dpdk-install-dev DPDK\_MLX5\_PMD=y  # make build-release  # make pkg-rpm vpp\_uses\_dpdk\_mlx5\_pmd=yes  // for another NIC  # make dpdk-install-dev  # make build-release  # make pkg-rpm |

5.6 startup.conf parameters

[reference here.](https://fdio-vpp.readthedocs.io/en/latest/gettingstarted/users/configuring/startup.html#unix)

* unix

|  |
| --- |
| unix {  interactive  nodaemon  log <filename> # default /var/log/vpp/vpp.log  exec|startup-config <filename> # both the two keywords are alias for the same function  gid number|name  full-coredump # dump all memory-mapped address region instead of just text + data + bss  coredump=size unlimited |<n> G|<n> M| <n> K| <n>  cli-listen <ipaddress:port>|<socket-path>  cli-line-mode  cli-prompt <string>  cli-history-limit <n>  cli-no-banner  cli-no-pager  cli-pager-buffer-limit <n>  runtime-dir <dir>  poll-sleep-usec <n>  pidfile <filename>  } |

* dpdk

|  |
| --- |
| dpdk {  dev <pci-dev>  dev <pci-dev> {  num-rx-queues <n> # default value is 1  num-tx-queues <n> # equal to number of worker threads or 1 if no workers treads.  num-rx-desc <n> # default is 1024  num-tx-desc <n> # default is 1024  workers  vlan-strip-offload on|off  hqos  rss  }  vdev <eak0command> # provide a DPDK EAL command to specify bonded Ethernet interface  num-mbufs <n> # default is 16384  no-pci  no-hugetlb  kni <n> # kernel NIC interface  uio\_driver uio\_pci\_generi|igb\_uio|vfio-pci|auto # default is auto  socket-mem <n> #default is 64  enable-tcp-udp-checksum  no-multi-seg  no-tx-checksum-offload  decimal-interface-names  log-level emergency|alert|critical|error|warning|notice|info|debug # default is notice  dev default {…}  } |

* cpu: the main thread and worker thread(s) can be pinned to cpu core(s) automatically or manually.

|  |
| --- |
| cpu {  # automatic pinning:  workers <n>  io <n>  main-thread-io  skip-cores <n>  # manual pinning:  main-core <n>  coremask-workers <hex-mask>  corelist-workers <list>  coremask-io <hex-mask>  corelist-io <list>  coremask-hqos-threads <hex-mask>  corelist-hqos-threads <list>  # other  use-pthreads  thread-prefix <prefix>  scheduler-policy rr|fifo|batch|idle|other  scheduler-priority <n>  <thread-name> <count> #threads include hqos-threads and workers.  } |