N3000 vRouter-DPDK

Building process proposal

Requirements and Goals

- Build vRouter code with Intel N3000 related code enabled

  The N3000 may be enabled on compilation time to compile and link the N3000-related code with the vRouter-DPDK binary. When the code is compiled in, it then can be enabled on vRouter start by providing specific command line arguments.

- Push custom images with specific names to docker registry

  The images may then be used during deployment.

- Make sure incoming changes unrelated to the N3000 won’t break N3000 support

  We should enforce all future changes to not break the N3000-related code. The TF CI system should at least build all changes with N3000 code enabled. Some N3000-related tests would also be great, but it requires using baremetal servers and N3000 cards in the CI.

- Use custom, patched uio_pci_generic driver

Custom driver justification

- vfio-pci cannot be used

  vfio-pci driver cannot currently be used because due to lack of ACS all devices are put into the same iommu group which causes vRouter to throw an error.

- Usable uio_pci_generic not available in RHEL 8.2 kernel

  uio_pci_generic driver can be used for the time being, but it requires specific kernel patches which are not available in RHEL 8.2 kernel. RHEL 8.4 contains all necessary patches, but it requires RHOS 16.2 which is not yet available.

- Patched version is required

  RHEL 8.2 with backported kernel patches can be used together with RHOS 16.1 (for now), however, we must provide an image that could be used by Juniper (the image should be available via TF docker hub).

Proposed solution

- Create custom Jenkins job for N3000

  It will allow checking every commit with N3000-related code and building and pushing N3000 images to docker registry.

- Build custom driver

  The source code of uio_pci_generic should be downloaded, the patch should be applied, and it should be build. Changes in container-builder and/or tf-dev-env may be required.

- Load custom driver on vRouter startup

  The driver may be stored inside one of the images, e.g. kernel-init. It then may be loaded during vRouter initialization, when kernel-init initializes all other vRouter-related things

Thank you

codilime®
CREATING VALUE

2100 Geng Road, Suite 210
Palo Alto, CA 94303
United States of America
+1 650 285 2458

Krancowa 5
02-493, Warsaw
Poland
+48 22 389 51 00

al. Grunwaldzka 472B (OBC)
80-309 Gdansk
Poland
+48 575 700 785

contact@codilime.com