DPDK

DATA PLANE DEVELOPMENT KIT

Crypto Device Drivers

Release 2.2.0

January 16, 2016

CONTENTS

CHAPTER

ONE

AESN-NI MULTI BUFFER CRYTPO POLL MODE DRIVER

The AESNI MB PMD (**librte_pmd_aesni_mb**) provides poll mode crypto driver support for utilizing Intel multi buffer library, see the white paper Fast Multi-buffer IPsec Implementations on Intel® Architecture Processors.

The AES-NI MB PMD has current only been tested on Fedora 21 64-bit with gcc.

1.1 Features

AESNI MB PMD has support for:

Cipher algorithms:

- RTE_CRYPTO_SYM_CIPHER_AES128_CBC
- RTE_CRYPTO_SYM_CIPHER_AES256_CBC
- RTE_CRYPTO_SYM_CIPHER_AES512_CBC

Hash algorithms:

- RTE_CRYPTO_SYM_HASH_SHA1_HMAC
- RTE_CRYPTO_SYM_HASH_SHA256_HMAC
- RTE_CRYPTO_SYM_HASH_SHA512_HMAC

1.2 Limitations

- Chained mbufs are not supported.
- Hash only is not supported.
- Cipher only is not supported.
- Only in-place is currently supported (destination address is the same as source address).
- Only supports session-oriented API implementation (session-less APIs are not supported).
- Not performance tuned.

1.3 Installation

To build DPDK with the AESNI_MB_PMD the user is required to download the mult- buffer library from here and compile it on their user system before building DPDK. When building the multi-buffer library it is necessary to have YASM package installed and also requires the overriding of YASM path when building, as a path is hard coded in the Makefile of the release package.

```
make YASM=/usr/bin/yasm
```

The environmental variable AESNI_MULTI_BUFFER_LIB_PATH must be exported with the path where you extracted and built the multi buffer library and finally set CONFIG_RTE_LIBRTE_PMD_AESNI_MB=y in config/common_linuxapp.

CHAPTER

TWO

QUICK ASSIST CRYPTO POLL MODE DRIVER

The QAT PMD provides poll mode crypto driver support for **Intel QuickAssist Technology DH895xxC** hardware accelerator.

The QAT PMD has been tested on Fedora 21 64-bit with gcc and on the 4.3 kernel.org Linux kernel.

2.1 Features

The QAT PMD has support for:

Cipher algorithms:

- `RTE_CRYPTO_SYM_CIPHER_AES128_CBC`
- `RTE_CRYPTO_SYM_CIPHER_AES192_CBC`
- `RTE_CRYPTO_SYM_CIPHER_AES256_CBC`

Hash algorithms:

- `RTE_CRYPTO_AUTH_SHA1_HMAC`
- `RTE_CRYPTO_AUTH_SHA256_HMAC`
- `RTE_CRYPTO_AUTH_SHA512_HMAC`
- `RTE_CRYPTO_AUTH_AES_XCBC_MAC`

2.2 Limitations

- Chained mbufs are not supported.
- Hash only is not supported.
- Cipher only is not supported.
- Only in-place is currently supported (destination address is the same as source address).
- Only supports the session-oriented API implementation (session-less APIs are not supported).
- Not performance tuned.

2.3 Installation

To use the DPDK QAT PMD an SRIOV-enabled QAT kernel driver is required. The VF devices exposed by this driver will be used by QAT PMD.

If you are running on kernel 4.3 or greater, see instructions for *Installation using kernel.org driver* below. If you are on a kernel earlier than 4.3, see *Installation using 01.org QAT driver*.

2.4 Installation using 01.org QAT driver

Download the latest QuickAssist Technology Driver from 01.org Consult the *Getting Started Guide* at the same URL for further information.

The steps below assume you are:

- Building on a platform with one `DH895xCC` device.
- Using package `qatmux.l.2.3.0-34.tgz`.
- On Fedora21 kernel `3.17.4-301.fc21.x86_64`.

In the BIOS ensure that SRIOV is enabled and VT-d is disabled.

Uninstall any existing QAT driver, for example by running:

- `./installer.sh uninstall` in the directory where originally installed.
- or `rmmod qat_dh895xcc; rmmod intel_qat`.

Build and install the SRIOV-enabled QAT driver:

```
mkdir /QAT
cd /QAT
# copy qatmux.l.2.3.0-34.tgz to this location
tar zxof qatmux.l.2.3.0-34.tgz

export ICP_WITHOUT_IOMMU=1
./installer.sh install QAT1.6 host
```

You can use `cat /proc/icp_dh895xcc_dev0/version` to confirm the driver is correctly installed. You can use `lspci -d:443` to confirm the bdf of the 32 VF devices are available per `DH895xCC` device.

To complete the installation - follow instructions in *Binding the available VFs to the DPDK UIO driver*.

Note: If using a later kernel and the build fails with an error relating to `strict_stroul` not being available apply the following patch:

```
/QAT/QAT1.6/quickassist/utilities/downloader/Target_CoreLibs/uclo/include/linux/uclo_platform.h
+ #if LINUX_VERSION_CODE >= KERNEL_VERSION(3,18,5)
+ #define STR_TO_64(str, base, num, endPtr) {endPtr=NULL; if (kstrtoul((str), (base), (num))) p
+ #else
#if LINUX_VERSION_CODE >= KERNEL_VERSION(2,6,38)
#define STR_TO_64(str, base, num, endPtr) {endPtr=NULL; if (strict_strtoull((str), (base), (num
#else
#if LINUX_VERSION_CODE >= KERNEL_VERSION(2,6,25)
#define STR_TO_64(str, base, num, endPtr) {endPtr=NULL; strict_strtoll((str), (base), (num));}
#else
#define STR_TO_64(str, base, num, endPtr)                                       \
     do {                                                                  \
```

```
            if (str[0] == '-')                                              \
            {                                                               \
                 *(num) = -(simple_strtoull((str+1), &(endPtr), (base)));  \
            }else {                                                         \
                 *(num) = simple_strtoull((str), &(endPtr), (base));        \
            }                                                               \
       } while(0)
+ #endif
#endif
#endif
```

If the build fails due to missing header files you may need to do following:

- `sudo yum install zlib-devel`
- `sudo yum install openssl-devel`

If the build or install fails due to mismatching kernel sources you may need to do the following:

- `sudo yum install kernel-headers-`uname -r`
- `sudo yum install kernel-src-`uname -r`
- `sudo yum install kernel-devel-`uname -r`

2.5 Installation using kernel.org driver

Assuming you are running on at least a 4.3 kernel, you can use the stock kernel.org QAT driver to start the QAT hardware.

The steps below assume you are:

- Running DPDK on a platform with one `DH895xCC` device.
- On a kernel at least version 4.3.

In BIOS ensure that SRIOV is enabled and VT-d is disabled.

Ensure the QAT driver is loaded on your system, by executing:

```
lsmod | grep qat
```

You should see the following output:

```
qat_dh895xcc            5626  0
intel_qat              82336  1 qat_dh895xcc
```

Next, you need to expose the VFs using the sysfs file system.

First find the bdf of the DH895xCC device:

```
lspci -d : 435
```

You should see output similar to:

```
03:00.0 Co-processor: Intel Corporation Coleto Creek PCIe Endpoint
```

Using the sysfs, enable the VFs:

```
echo 32 > /sys/bus/pci/drivers/dh895xcc/0000\:03\:00.0/sriov_numvfs
```

If you get an error, it's likely you're using a QAT kernel driver earlier than kernel 4.3.

To verify that the VFs are available for use - use `lspci -d:443` to confirm the bdf of the 32 VF devices are available per `DH895xCC` device.

To complete the installation - follow instructions in *Binding the available VFs to the DPDK UIO driver*.

2.6 Binding the available VFs to the DPDK UIO driver

The unbind command below assumes `bdfs` of `03:01.00-03:04.07`, if yours are different adjust the unbind command below:

```
cd $RTE_SDK
modprobe uio
insmod ./build/kmod/igb_uio.ko

for device in $(seq 1 4); do \
    for fn in $(seq 0 7); do \
        echo -n 0000:03:0${device}.${fn} > \
        /sys/bus/pci/devices/0000\:03\:0${device}.${fn}/driver/unbind; \
    done; \
done

echo "8086 0443" > /sys/bus/pci/drivers/igb_uio/new_id
```

You can use `lspci -vvd:443` to confirm that all devices are now in use by igb_uio kernel driver.