IONIC Manual for FreeBSD

Introduction

This document provides prerequisites and instructions for building and testing the Pensando IONIC device driver on FreeBSD

Pre-requisites

Compiling the Pensando IONIC FreeBSD driver requires having FreeBSD source code including kernel source code.

Clone the FreeBSD github repo to checkout FreeBSD Head (12 branch):

```
$ git clone http://github.com/freebsd/freebsd /usr/src
```

Checkout the 11.2 source branch:

```
$ cd /usr/src
$ git checkout  remotes/origin/releng/11.2 -b releng11.2
```

Configure the FreeBSD Kernel

The IONIC driver requires certain options from the running kernel. If the running kernel does not support **OFED** and **COMPAT_LINUXKPI** options, then the kernel will need to be rebuilt. Add following lines in sys/amd64/conf/GENERIC, anywhere in the file:

```
options    OFED
options    COMPAT_LINUXKPI
```

- **COMPAT_LINUXKPI** is required for the sonic/storage accelerator driver and ionic_rdma.
- **OFED** is required for RDMA

Create file /etc/src.conf and add the following line:

```
WITH_OFED='yes'
```

Build the FreeBSD Kernel

```
$ make -j 8  buildworld buildkernel installworld installkernel
```

When complete, reboot and type ‘uname -a’ to verify.

Ethernet

Building ionic

There are two ways to get Pensando driver source code.

- Checkout Pensaod/sw or copy platform/driver/freebsd directory to the required host and go to platform/drivers/freebsd/usr/src
- Copy driver-freebsd.tar.gz package from build to host

```
$ env OS_DIR=/usr/src ./build.sh
```

Loading ionic

Before installing Pensando drivers, make sure all the devices are visible on PCI bus. Verify as below:

```
# pciconf -l |grep 1dd8
pcib9@pci0:94:0:0:      class=0x060400 card=0x40011dd8
chip=0x10001dd8 rev=0x00 hdr=0x01
pcib10@pci0:95:0:0:     class=0x060400 card=0x40011dd8
chip=0x10011dd8 rev=0x00 hdr=0x01
pcib11@pci0:95:1:0:     class=0x060400 card=0x40011dd8
chip=0x10011dd8 rev=0x00 hdr=0x01
pcib12@pci0:95:2:0:     class=0x060400 card=0x40011dd8
chip=0x10011dd8 rev=0x00 hdr=0x01
ion0@pci0:96:0:0:       class=0x020000 card=0x40011dd8
chip=0x10021dd8 rev=0x00 hdr=0x00 << Network
ion1@pci0:97:0:0:       class=0x020000 card=0x40011dd8
chip=0x10021dd8 rev=0x00 hdr=0x00  << Network
none135@pci0:98:0:0:    class=0xff0000 card=0x40011dd8
chip=0x10071dd8 rev=0x00 hdr=0x00 << Storage accelerator
```

If system doesn't list all the above devices, add the following line in **/boot/loader.conf** and reboot:

```
     hw.pci.enable_ari="0"
```

Once the driver is built, you can load the **ionic** NIC/Ethernet driver:

```
# kldload sys/modules/ionic/ionic.ko
```

To load the **ionic** RDMA driver:

```
# kldload sys/modules/ionic_rdma/ionic_rdma.ko
```

Configure Network interface

Once the module is loaded and if ARI is disabled, you should see 3 network interfaces

```
ionic0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
```

```
options=e507bb<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,JUMBO_MTU,VLAN_H
WCSUM,TSO4,TSO6,LRO,VLAN_HWFILTER,VLAN_HWTSO,RXCSUM_IPV6,TXCSUM_IPV6>
      ether 00:ae:cd:00:01:3a
      hwaddr 00:ae:cd:00:01:3a
      nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
      media: Ethernet autoselect (100GBase-CR4 <full-duplex>)
      status: active
ionic1: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500

options=e507bb<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,JUMBO_MTU,VLAN_H
WCSUM,TSO4,TSO6,LRO,VLAN_HWFILTER,VLAN_HWTSO,RXCSUM_IPV6,TXCSUM_IPV6>
      ether 00:ae:cd:00:01:3b
      hwaddr 00:ae:cd:00:01:3b
      nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
      media: Ethernet autoselect (100GBase-CR4 <full-duplex>)
      status: active
ionic2: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500

options=e507bb<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,JUMBO_MTU,VLAN_H
WCSUM,TSO4,TSO6,LRO,VLAN_HWFILTER,VLAN_HWTSO,RXCSUM_IPV6,TXCSUM_IPV6>
      ether 00:ae:cd:00:01:3c
      hwaddr 00:ae:cd:00:01:3c
      nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
      media: Ethernet autoselect (1000Base-KX <full-duplex>)
      status: active
```

NB: First two ports, **ionic0** and **ionic1** are 100G data ports. **ionic2** is mgmt interface to NIC and is used as the Management port by ***penctl***.

Collecting statistics

- **Statistics are available through “sysctl dev.ionic.0” and “sysctl dev.ionic1” for the respective ports, providing detailed statistics. Ex:**

```
# sysctl dev.ionic.0
...
dev.ionic.0.txq15.dma_map_error: 0
dev.ionic.0.txq15.num_descs: 16384
dev.ionic.0.txq15.comp_index: 9521
dev.ionic.0.txq15.tail: 9521
dev.ionic.0.txq15.head: 9522
```

```
dev.ionic.0.txq14.tso_max_sg: 0
dev.ionic.0.txq14.tso_max_size: 0
dev.ionic.0.txq14.tso_ipv6: 0
dev.ionic.0.txq14.tso_ipv4: 0
dev.ionic.0.txq14.no_csum_offload: 0
dev.ionic.0.txq14.csum_offload: 224802440
dev.ionic.0.txq14.bytes: 15288359411
dev.ionic.0.txq14.pkts: 224802440
dev.ionic.0.txq14.bad_ethtype: 0
dev.ionic.0.txq14.linearize_err: 0
dev.ionic.0.txq14.linearize: 0
dev.ionic.0.txq14.no_descs: 0
dev.ionic.0.txq14.pkts_retry: 0
dev.ionic.0.txq14.tx_clean: 284816431
dev.ionic.0.txq14.dma_map_error: 0
dev.ionic.0.txq14.num_descs: 16384
```

- **To focus on a particular queue, (e.g. receive queue 0 stats), run:**

```
# sysctl dev.ionic.0.rxq0
dev.ionic.0.rxq0.rss_unknown: 0
dev.ionic.0.rxq0.rss_udp_ip6_ex: 0
dev.ionic.0.rxq0.rss_tcp_ip6_ex: 0
dev.ionic.0.rxq0.rss_ip6_ex: 0
dev.ionic.0.rxq0.rss_udp_ip6: 0
dev.ionic.0.rxq0.rss_tcp_ip6: 0
dev.ionic.0.rxq0.rss_ip6: 0
dev.ionic.0.rxq0.rss_udp_ip4: 0
dev.ionic.0.rxq0.rss_tcp_ip4: 313362
dev.ionic.0.rxq0.rss_ip4: 0
dev.ionic.0.rxq0.lro_bad_csum: 0
dev.ionic.0.rxq0.lro_flushed: 240745
dev.ionic.0.rxq0.lro_queued: 312625
dev.ionic.0.rxq0.mbuf_free: 0
dev.ionic.0.rxq0.mbuf_alloc: 329633
dev.ionic.0.rxq0.isr_count: 185068
dev.ionic.0.rxq0.csum_l4_bad: 0
dev.ionic.0.rxq0.csum_l4_ok: 313362
dev.ionic.0.rxq0.csum_ip_bad: 0
dev.ionic.0.rxq0.csum_ip_ok: 313362
dev.ionic.0.rxq0.bytes: 20691314
dev.ionic.0.rxq0.pkts: 313376
```

```
dev.ionic.0.rxq0.comp_err: 0
dev.ionic.0.rxq0.alloc_error: 0
dev.ionic.0.rxq0.dma_setup_error: 0
dev.ionic.0.rxq0.num_descs: 16384
dev.ionic.0.rxq0.comp_index: 2080
dev.ionic.0.rxq0.tail: 2080
dev.ionic.0.rxq0.head: 1953
```

- **To focus on transmit queue 10:**

```
# sysctl dev.ionic.0.txq10
dev.ionic.0.txq10.tso_max_sg: 0
dev.ionic.0.txq10.tso_max_size: 0
dev.ionic.0.txq10.tso_ipv6: 0
dev.ionic.0.txq10.tso_ipv4: 0
dev.ionic.0.txq10.no_csum_offload: 0
dev.ionic.0.txq10.csum_offload: 0
dev.ionic.0.txq10.bytes: 0
dev.ionic.0.txq10.pkts: 0
dev.ionic.0.txq10.bad_ethtype: 0
dev.ionic.0.txq10.linearize_err: 0
dev.ionic.0.txq10.linearize: 0
dev.ionic.0.txq10.no_descs: 0
dev.ionic.0.txq10.pkts_retry: 0
dev.ionic.0.txq10.tx_clean: 14
dev.ionic.0.txq10.dma_map_error: 0
dev.ionic.0.txq10.num_descs: 16384
dev.ionic.0.txq10.comp_index: 0
dev.ionic.0.txq10.tail: 0
dev.ionic.0.txq10.head: 0
```

Change MTU size

- Change the MTU size through “ifconfig”. Ex:

```
# ifconfig ionic0 mtu 1500
```

Enable/disable checksum, TSO and LRO through “ifconfig”

Ex:

```
# ifconfig ionic0 -rxcsum — Disable Rx checksum

# ifconfig ionic0 rxcsum — Reenable Rx checksum

# ifconfig ionic0 -txcsum — Disable Tx checksum

# ifconfig ionic0 txcsum — Reeable Tx checksum

# ifconfig ionic0 -tso — Disable TSO

# ifconfig ionic0 tso — Re-enable TSO

# ifconfig ionic0 -lro — Disable LRO

# ifconfig ionic0 lro — Reenable LRO
```

Changing number of queues

Changing number of queues is done through “kenv” and requires reloading the ionic driver. Ex:

```
root # kenv hw.ionic.max_queues=8
hw.ionic.max_queues="8"
root# kldunload ionic
root # kldload ionic.ko
root # sysctl hw.ionic
hw.ionic.max_sg: 0
hw.ionic.rx_coalesce_usecs: 64
hw.ionic.tx_coalesce_usecs: 64
hw.ionic.rx_process_limit: 128
hw.ionic.tx_clean_threshold: 128
hw.ionic.rx_fill_threshold: 128
hw.ionic.rx_stride: 32
hw.ionic.rx_descs: 16384
hw.ionic.tx_descs: 16384
hw.ionic.adminq_descs: 16
hw.ionic.enable_msix: 1
hw.ionic.max_queues: 8 << Number of queues is 8 now.
```

Changing Ring size

To change the ring size:

- set hw.ionic.tx_descs for Transmit side descriptors
  ```
  # kenv hw.ionic.tx_descs=16384
  ```
- set hw.ionic.rx_descs for Receive side descriptors
  ```
  # kenv hw.ionic.rx_descs=16384
  ```
- Verify by running:

  ```
  # sysctl hw.ionic
  hw.ionic.max_sg: 0
  hw.ionic.rx_coalesce_usecs: 64
  hw.ionic.tx_coalesce_usecs: 64
  hw.ionic.rx_process_limit: 128
  hw.ionic.tx_clean_threshold: 128
  hw.ionic.rx_fill_threshold: 128
  hw.ionic.rx_stride: 32
  hw.ionic.rx_descs: 16384
  hw.ionic.tx_descs: 16384
  hw.ionic.adminq_descs: 16
  hw.ionic.enable_msix: 1
  hw.ionic.max_queues: 16
  ```

- Reload ionic driver

  ```
  # kldunload ionic
  # kldload ionic.ko
  ```

For user space, there is a corresponding libionic.so which is built by our build script. For freebsd, applications should be built referencing this library as a dependency. If not, this can be loaded by LD_PRELOAD, or by starting user space programs with the run_rdma.sh helper script that is provided. For example: ./run_rdma.sh ibv_devinfo

Linux Module Parameters

These are the module params and descriptions, from `modinfo ionic_rdma.ko`:

```
parm:           dbgfs:Enable debugfs for this driver. (bool)
```

```
parm:           aq_depth:Min depth for admin queues. (ushort)
parm:           eq_depth:Min depth for event queues. (ushort)
parm:           isr_budget:Max events to poll per round in isr context. (ushort)
parm:           work_budget:Max events to poll per round in work context. (ushort)
parm:           max_pd:Max number of PDs. (int)
parm:           max_gid:Max number of GIDs. (int)
```

The metaparameter dyndbg and the file /sys/kernel/debug/dynamic_debug/control can be used to control dynamic debugging to dmesg.

For user space, libionic_rdmav19.so is built with, and can be installed with rdma-core according to the build and install instructions of rdma-core. We provide a copy of rdma-core with our driver added. The version v19 comes from rdma-core, and changes with the rdma-core release cycle.

Rdma device information:

```
root#  run_rdma.sh ibv_devinfo
```

Using `rdmactl.py` for debugging

Counters and detailed internal state of rdma resources on the device can be inspected with the debugging program `rdmactly.py`. The `rdmactl.py` tool is developed primarily for internal use, for debugging. The counters will be made available via the `penctl` user interface in a subsequent release.

```
usage: rdmactl.py [-h] [--dmesg] [--dmesg_file DMESG_FILE] [--DEVNAME
DEVNAME]
                  [--offline] [--host HOST]
                  [--sqcb0 qid | --sqcb1 qid | --sqcb2 qid | --sqcb3
qid | --rqcb0 qid | --rqcb1 qid | --rqcb2 qid | --rqcb3 qid | --cqcb
qid | --eqcb qid | --aqcb0 qid | --aqcb1 qid | --req_tx_stats qid |
--req_rx_stats qid | --resp_tx_stats qid | --resp_rx_stats qid |
--kt_entry key_id | --pt_entry pt_offset | --lif_stats | --q_stats
qid | --q_state qid | --rsq qid | --rrq qid | --aq_debug_enable qid |
--aq_debug_disable qid]
```

example: `rdmactl.py --DEVNAME ionic0 --lif_stats`

optional arguments:

```
  -h, --help            show this help message and exit
  --dmesg               parse dmesg and parse rdma adminq wqes/cqes
  --dmesg_file DMESG_FILE
```

```
                        parse dmesg from file and parse rdma adminq
wqes/cqes
  --DEVNAME DEVNAME     prints info for given rdma device
  --offline             prints rdma per queue state through the
console when
                        the host is not accessible
  --host HOST           name of the host where Naples is present
  --sqcb0 qid           prints sqcb0 state given qid
  --sqcb1 qid           prints sqcb1 state given qid
  --sqcb2 qid           prints sqcb2 state given qid
  --sqcb3 qid           prints sqcb3 state given qid
  --rqcb0 qid           prints rqcb0 state given qid
  --rqcb1 qid           prints rqcb1 state given qid
  --rqcb2 qid           prints rqcb2 state given qid
  --rqcb3 qid           prints rqcb3 state given qid
  --cqcb qid            prints cqcb state given qid
  --eqcb qid            prints eqcb state given qid
  --aqcb0 qid           prints aqcb0 state given qid
  --aqcb1 qid           prints aqcb1 state given qid
  --req_tx_stats qid    prints req_tx stats given qid
  --req_rx_stats qid    prints req_rx stats given qid
  --resp_tx_stats qid   prints resp_tx stats given qid
  --resp_rx_stats qid   prints resp_rx stats given qid
  --kt_entry key_id     prints key table entry given key id
  --pt_entry pt_offset  print page table entry given pt offset
  --lif_stats           prints rdma LIF statistics
  --q_stats qid         prints rdma per queue statistics
  --q_state qid         prints rdma per queue state
  --rsq qid             prints rdma rsq entries
  --rrq qid             prints rdma rrq entries
  --aq_debug_enable qid
                        Enable AQ captrace
  --aq_debug_disable qid
                        Disable AQ captrace
```

Link pause

“Link Pause” need to be enabled end to end (including switches). By default link pause is enabled on Naples. Users must ensure Link Pause is enabled on any intermediate switches.

A sample “Link Pause” configuration on N3K is provide below for reference

Sample Nexus 3K Link pause configuration (Nexus 3232)

```
 policy-map type network-qos pause-no-drop
    class type network-qos c-nq-default
      match qos-group 0
      mtu 9216
      pause pfc-cos 0
system qos
  service-policy type network-qos pause-no-drop
```

Link pause configuration on the interfaces:

```
int ethx/y
  flowcontrol receive on
  flowcontrol send on
  mtu 9216
```