Network Balancing Act Documentation

Release 0.1

Joongi Kim

Oct 10, 2018

Getting Started Guide

Network Balancing Act, or **NBA**, is a packet processing framework based on Linux and commodity Intel x86 hardware.

Its goal is to offer both programmability and high performance by exploiting modern commodity hardware such as multi-queue-enabled 10 GbE network cards, multi-socket/multi-core CPUs, and computation accelerators like GPUs. The programming interface resembles the Click modular router, which express packet processing functions as composable C++ classes (called *elements*). Click has been a de-facto standard framework for software-based packet processors for years, but suffered from low performance. In contrast, on Intel Sandy Bridge (E5-2670) dual-socket servers with eight 10 GbE (Intel X520-DA2) network cards, NBA saturates the hardware I/O capacity with light-weight workloads: 80 Gbps IPv4 forwarding throughput for large packets and 62 Gbps for minimum-sized packets.

Note that NBA covers per-packet processing only—it does not support flow-level processing yet.

CHAPTER 1

System Requirements

1.1 Supported Hardware

NBA can run on most Intel servers with supported NICs, but we recommend Sandy Bridge or newer generations of Intel CPUs since integrated memory controller and integrated PCIe controller is a important performance booster.

For GPU acceleration, the current version only supports NVIDIA CUDA GPUs, either desktop class (GeForce) or server class (Tesla) models.

1.2 BIOS Settings

- If available, turn off Intel EIST technology in BIOS.
- You need to fix the clock speed of individual CPU cores on Haswell systems (Xeon E5-XXXXv3) for accurate timing control and performance measurements.

CHAPTER 2

Building NBA

2.1 Supported Platforms

Currently NBA is only tested on Linux x86_64 3.10 or newer kernels, and the Ubuntu 14.04/16.04 LTS distribution.

2.2 Step-by-step Guide

2.2.1 Installing dependencies

Ensure that you have a C/C++ compiler (e.g., g++ 4.8 or newer). The compiler must support the C++11 standard.

Check out the latest DPDK source tree:

```
$ git clone git://dpdk.org/dpdk
$ cd dpdk
$ $EDITOR configs/common_linuxapp
$ make install T=x86_64-native-linuxapp-gcc
```

See details on what you need to edit on `configs/common_linuxapp` at *DPDK configurations for network cards*.

Note: You need to install the kernel header/source packages first.

Install library dependencies on your system:

```
$ sudo apt-get install libev-dev libssl-dev libpapi-dev
```

Install Python 3.5 on your system. You may use the system package manager such as `apt-get`. In that case, ensure that you also have development package as well:

```
$ sudo apt-get install python3.5 libpython3.5-dev
```

Then install our Python dependencies:

```
$ pip3 install --user snakemake
```

Note: We recommend using a separate Python environment contained inside the user directory. See pyenv for more details.

2.2.2 Compilation

Clone the project source code:

```
$ git clone https://github.com/anlab-kaist/NBA nba
```

Install our 3rd-party libraries, the Click configuration parser:

```
$ cd nba
$ git submodule init && git submodule update
```

It will be *automatically built* along with NBA when you first build NBA.

Set the environment variable as follows:

```
$ export NBA_DPDK_PATH=/home/userid/dpdk/x86_64-native-linuxapp-gcc
$ export USE_CUDA=0  # for testing CPU-only version without CUDA installation
```

Finally, run:

```
$ snakemake -j
```

If all is well, the executable is located in `bin/main`.

2.3 DPDK Configs for Network Cards

2.3.1 Intel X520 Series (82599 chipset)

You just need to bind the PCI addresses of network cards to igb_uio using `tools/dpdk_nic_bind.py` script provided by DPDK.

Attention: When using ixgbe driver with *vectorized* PMD enabled, you should fix the IO batch size to be 32, whereas you may change the computation batch size as you want.

In our experiements, the IO batch size 32 and the computation batch size 64 performs best. We have already set those as the default values. (see `config/default.py`)

2.3.2 Mellanox ConnectX Series

You need to install the OFED toolchain provided by Mellanox because DPDK's mlx4 poll-mode driver uses Mellanox's kernel Infiniband driver to control the hardware and perform DMA. We recommend to use version 3.0 or later, as these new versions have much better performance and includes firmware updates.

To use mlx4_pmd on DPDK, turn on it inside DPDK's configuration:

```
CONFIG_RTE_LIBRTE_MLX4_PMD=y
```

To increase throughputs, set the following in the same config:

```
CONFIG_RTE_LIBRTE_MLX4_SGE_WR_N=1
```

For maximum throughputs, turn off the followings:

- blueflame[1]: `sudo ethtool --set-priv-flags ethXX blueflame off`
- rx/tx auto-negotiation for flow control: `sudo ethtool -A ethXX rx off tx off`

Note that above settings must be done in packet generators as well.

Warning: We recommend to turn off blueflame when loading the mlx4_core kernel module as module parameters, instead of using ethtool afterwards.

You do not need to explicitly bind the PCI addresses of Mellanox cards to igb_uio because mlx4_pmd automatically detects them using the kernel driver.

To use mlx4 in NBA, set the following environment variable and rebuild:

```
$ export NBA_PMD=mlx4
$ snakemake clean && snakemake -j
```

2.4 Optional Installations

2.4.1 NVIDIA CUDA

If you want to use GPU acceleration, install NVIDIA CUDA 7.0 or newer. We recommend to download the latest version of `.bin` package from the NVIDIA website instead of using system packages.

Note: A small daemon is required to "pin" GPU's interrupts to specific cores. See details in our gist.

Make CUDA binaries accessible from your shell:

```
$ echo 'export PATH="$PATH:/usr/local/cuda/bin"' >> ~/.profile
$ sudo sh -c 'echo /usr/local/cuda/lib64 > /etc/ld.so.conf.d/cuda.conf'
$ sudo ldconfig
```

To use CUDA in NBA, do:

```
$ export USE_CUDA=1
$ snakemake clean && snakemake -j
```

[1] "blueflame" is a Mellanox-specific feature that uses PCI BAR for tranferring descriptors of small packets instead of using DMA on RX/TX rings. It is known to have lower latency, but causes throughput degradation with NBA.

2.4.2 Intel Xeon Phi

If you want to use Xeon Phi acceleration, install the latest Intel MPSS (many-core platform software stack) by visiting the official website.

2.4.3 CPU statistics

To run experiment scripts, install `sysstat` package (or any package that offers `mpstat` command).

2.5 Customizing Your Build

Our build script offers a few configurable parameters as environment variables:

- `NBA_DPDK_PATH`: specifies the path to Intel DPDK (required)
- `NBA_RANDOM_PORT_ACCESS`: randomizes the RX queue scanning order for each worker thread (default: `false`)
- `NBA_OPENSSL_PATH`: specifies the path of OpenSSL library (default: `/usr`)
- `DEBUG`: build without compiler optimization (default: 0)
- `USE_CUDA`: activates NVIDIA CUDA support (default: 1)
- `USE_KNAPP`: activates Knapp-based Intel Xeon Phi support (default: 0)
- `USE_PHI`: activates OpenCL-based Intel Xeon Phi support (default: 0, not implemented)
- `USE_NVPROF`: activates nvprof API calls to track GPU-related timings (default: 0)
- `USE_OPENSSL_EVP`: determines whether to use EVP API for OpenSSL that enables AES-NI support (default: 1)
- `NBA_NO_HUGE`: determines whether to use huge-pages (default: 1)
- `NBA_PMD`: determines what poll-mode driver to use (default: `ixgbe`)

Note: 1 means true and 0 means false for boolean options.

CHAPTER 3

Running NBA

3.1 Standalone Execution

Execute `bin/main` with DPDK EAL arguments and NBA arguments. For example,

```
$ sudo bin/main -cffff -n4 -- configs/rss.py configs/ipv4-router.click
```

For details about DPDK EAL arguments, see DPDK's documentation.

3.2 Scripted Execution

CHAPTER 4

Updating Documentation

Install the latest Python (3.5+ recommended) and Sphinx utility.

```
$ pip3 install --user Sphinx sphinx_rtd_theme
```

Then you may now compile the documentation as follows:

```
$ cd docs
$ make html
# open _build/index.html in your web browser
```

Committing and pushing the updates to the NBA repository will automatically trigger the updates in the online documentation.

CHAPTER 5

System Configuration

CHAPTER 6

Pipeline Configuration

CHAPTER 7

Writing Elements

CHAPTER 8

Writing Offloadable Elements

CHAPTER 9

Intrinsic Elements

- FromInput
- ToOutput
- Discard
- Queue
- RandomWeightedBranch
- Classifier
- PacketSizeClassifier
- None

CHAPTER 10

Ethernet Elements

- L2Forward
- DropBroadcasts

CHAPTER 11

IP Elements

- CheckIPHeader
- DecIPTTL
- IPLookup
- IPv4 datablocks

CHAPTER 12

IPv6 Elements

- CheckIP6Header
- DecIP6HLIM
- LookupIP6Route
- IPv6 datablocks

CHAPTER 13

IPsec Elements

- IPsecESPEncap
- IPsecAES
- IPsecAuthHMACSHA1
- IPsec datablocks

CHAPTER 14

Load Balancers

- CPUOnly
- GPUOnly
- LoadBalanceThruput
- LoadBalanceAdaptiveMeasure

CHAPTER 15

Writing Your Load Balancer

CHAPTER 16

Adding a New Compute Device

CHAPTER 17

Indices and tables

- genindex
- modindex
- search

Index