DOCA Tools

Table of contents

This is an overview of the set of tools provided by DOCA and their purpose.

Introduction

DOCA tools are a set of executables/scripts that are needed to produce inputs to some of the DOCA libraries and applications.

All tools are installed with DOCA, as part of the doca-tools package, and can either be directly accessed from the terminal or can be found at `/opt/mellanox/doca/tools`. Refer to NVIDIA DOCA Installation Guide for Linux for more information.

Info

For questions, comments, and feedback, please contact us at DOCA-Feedback@exchange.nvidia.com.

Tools

Comm Channel Admin Tool

CLI name: `doca_comm_channel_admin_tool`

The Comm Channel Admin Tool is used to monitor Comm Channel services and connections on both BlueField and the host.

DPA EU Management Tool

CLI name: `dpaeumgmt`

The DPA execution unit management tool allows users to manage the DPA's EUs which are the basic resource of the DPA. The tool enables the resource control of EUs to optimize the usage of computation resources of the DPA. Using this tool, users may query, create, and destroy EU partitions and groups , thus ensuring proper EU allocation between devices.

DPACC Compiler

CLI name: `dpacc`

DPACC is a high-level compiler for the DPA processor. It compiles code targeted for the DPA processor into an executable and generates a DPA program.

The DPA program is a host library with interfaces encapsulating the DPA executable. This DPA program can be linked with the host application to generate a host executable where the DPA code is invoked through the FlexIO runtime API.

FlexIO Build

CLI name: `build_flexio_device.sh`

The FlexIO Build tool is used to build and compile FlexIO device code into a static library.

It is designed to generate a host library that encapsulating DPA execution. This tool relies on DPACC.

PCC Counter

CLI name: `pcc_counters.sh`

The PCC Counter tool is used to print PCC-related hardware counters. The output counters help debug the PCC user algorithm embedded in the DOCA PCC application.

Socket Relay

CLI name: `doca_socket_relay`

DOCA Socket Relay allows Unix Domain Socket (AF_UNIX family) server applications to be offloaded to Bluefield while communication between the two sides is proxied by DOCA Comm Channel.

NVIDIA DOCA Comm Channel Admin Tool

This document describes the Comm Channel Admin Tool, used to monitor Comm Channel services, connections, etc.

Introduction

Comm Channel Admin Tool is used to monitor Comm Channel services and connections on both the DPU and the host. For more information on the DOCA Comm Channel library, refer to the DOCA Comm Channel – Deprecated.

Prerequisites

NVIDIA® BlueField®-2 firmware version 24.35.1012 or higher.

Description

On the DPU, Comm Channel Admin Tool can show the user which services are up alongside additional information about them:

- `service_name` – the name of the service
- `transport_type` – UD or DC
- `allowed_vhca_id` – which `vhca_id` is allowed to connect to this service
- `service_qpn_or_dct` –
    - UD transport type – indicates QP number of the service
    - DC transport type – indicates DC number of the service
- `num_connected_client` – indicates the number of current connected clients

- `max_num_connected_client` – indicates the maximum number of connected clients

Furthermore, users may query specific services according to the service name to get information about the connected clients. For every connected client the tool shows:

- `connection_id` – unique identifier for the connection
- `qpn` – QP number (or DCT) used for the connection

On the host, the tool can show the user the name of the active services on the DPU. Furthermore, it can show information about the active connections between the host and the DPU:

- `service_name` – the name of the service
- `transport_type` – UD or DC
- `my_qpn_or_dci` –
    - UD transport type – indicates QPN of the connection initiator
    - DC transport type – the field indicates the DCI of the connection initiator
- `connection_type` – `CONNECT_BY_SERVICE_ID` or `CONNECT_BY_VHCA_ID`
- `dst_vhca_id` – `vhca_id` of the target side of the connection. Valid only when the connection type is `CONNECT_BY_VHCA_ID`.

Execution

To execute the DOCA Comm Channel Admin Tool, run:

```
/opt/mellanox/doca/tools/doca_comm_channel_admin_tool
```

Afterwards, users get an interactive CI that awaits a command:

```
Comm-Channel-Admin-Tool >>
```

The commands and their flags can be obtained by writing help:

```
Comm-Channel-Admin-Tool >> help
Comm-Channel-Admin-Tool CLI Commands:
     service, s          To be used on the DPU
         --all, -a                              Show all
services
         --service-name, -sn [service_name]      Show specific
service and its connected clients
     connection, c       To be used on the host
         --all, -a                              Show all
connections
         --service-name, -sn [service_name]      Show specific
connection according to service name

     discovery, d        To be used on the host
         --all, -a                              Show all active
services
         --service-name, -sn [service_name]      Show specific
active service according to service name

     quit            Exit tool
```

On the DPU, to see all services, users may use the `service` (or `s`) command with the flag `--all` (or `-a`). For example:

```
Comm-Channel-Admin-Tool >> service --all
2022-09-28 09:54:28,410 - Comm-Channel-Admin-Tool - INFO - On
/dev/mst/mt41686_pciconf0:
```

```
Active services:
+------------------------+-------------------+-------------------+-
---------------------+-------------------------+------------------
----------+
| service_name           | transport_type    |   allowed_vhca_id |
service_qpn_or_dct |   num_connected_client |
max_num_connected_client |
+========================+===================+===================+==
| secure_channel_server | DC                 |                 0 |
4548 |                      1 |                         512 |
+------------------------+-------------------+-------------------+-
---------------------+-------------------------+------------------
----------+
2022-09-28 09:54:28,886 - Comm-Channel-Admin-Tool - INFO - On
/dev/mst/mt41686_pciconf0.1:
No active services
```

To query a specific service and see its connected clients, users may use the `--serivce-name` (or `-sn`) flag while providing the `service_name`. For example:

```

Comm-Channel-Admin-Tool >> service --service-name
secure_channel_server
2022-09-28 09:56:16,335 - Comm-Channel-Admin-Tool - INFO - On
/dev/mst/mt41686_pciconf0:
+------------------------+-------------------+-------------------+-
---------------------+-------------------------+------------------
----------+
| service_name           | transport_type    |   allowed_vhca_id |
service_qpn_or_dct |   num_connected_client |
max_num_connected_client |
+========================+===================+===================+==
| secure_channel_server | DC                 |                 0 |
4548 |                      1 |                         512 |
+-----------------------+-------------------+-------------------+-
---------------------+-------------------------+------------------
----------+

Connected clients:
+-----------------+------------------+
|  connection_id  | transport_type   |
+=================+==================+
|               0 |             4547 |
+-----------------+------------------+
2022-09-28 09:56:16,809 - Comm-Channel-Admin-Tool - INFO - On
/dev/mst/mt41686_pciconf0.1:
No active service by name secure_channel_server
```

On the host, to see all active services, users may use the `discovery` (or `d`) command with `--all` flag (or `-a`). For example:

```
Comm-Channel-Admin-Tool >> discovery --all
2022-09-28 12:58:42,201 - Comm-Channel-Admin-Tool - INFO - On
/dev/mst/mt41686_pciconf0:
Active services:
+-----------------------+
| service_name          |
+=======================+
| secure_channel_server |
+-----------------------+
2022-09-28 12:58:42,632 - Comm-Channel-Admin-Tool - INFO - On
/dev/mst/mt41686_pciconf0.1:
No active services
```

Users may also filter to show only specific services according to the `service_name` by using the `--serivce-name` (or `-sn`) flag while providing the `service_name`.

On the host, to see all active connections, users may use the `connection` command (or `c`) with `--all` flag (or `-a`). For example:

```
Comm-Channel-Admin-Tool >> connection --all
2022-09-28 13:01:54,420 - Comm-Channel-Admin-Tool - INFO - On
/dev/mst/mt41686_pciconf0:
Active connections:
+-----------------------+------------------+-----------------+---
--------------------+---------------+
| service_name          | transport_type   |   my_qpn_or_dci |
connection_type       |   dst_vhca_id |
+=======================+==================+=================+=====
| secure_channel_server | DC               |              71 |
CONNECT_BY_SERVICE_ID |             0 |
+-----------------------+------------------+-----------------+---
--------------------+---------------+
2022-09-28 13:01:54,860 - Comm-Channel-Admin-Tool - INFO - On
/dev/mst/mt41686_pciconf0.1:
No active Connections
```

Users may also filter to show only specific connections according to `service_name` by using the `--serivce-name` (or `-sn`) flag while providing the `service_name`.

Note

If a new service or connection is added, there is no need to restart the tool. After each command, the tool always shows the latest services and connections.

NVIDIA DOCA PCC Counter Tool

This document provides instruction on the usage of the PCC Counter tool.

Introduction

The PCC Counter tool is used to print PCC-related hardware counters. The output counters help debug the PCC user algorithm embedded in the DOCA PCC application.

Prerequisites

DOCA 2.2.0 and higher.

Description

If NVIDIA® BlueField®-3 is operating in DPU mode, the script must be executed on the Arm side. If BlueField-3 is operating in NIC mode, the script must be executed on the host side.

Info

Refer to NVIDIA BlueField DPU Modes of Operation for more information on the DPU's modes of operation.

The following performance counters are supported for PCC:

- `MAD_RTT_PERF_CONT_REQ` – the number of RTT requests received in total
- `MAD_RTT_PERF_CONT_RES` – the number of RTT responses received in total
- `SX_EVENT_WRED_DROP` – the number of TX events dropped due to the CC event queue being full

- `SX_RTT_EVENT_WRED_DROP` – the number of "TX event with RTT request sent indication" dropped due to the CC event queue being full
- `ACK_EVENT_WRED_DROP` – the number of Ack events dropped due to the CC event queue being full
- `NACK_EVENT_WRED_DROP` – the number of Nack events dropped due to the CC event queue being full
- `CNP_EVENT_WRED_DROP` – the number of CNP events dropped due to the CC event queue being full
- `RTT_EVENT_WRED_DROP` – the number of RTT events dropped due to the CC event queue being full
- `HANDLED_SXW_EVENTS` – the number of handled CC events related to SXW
- `HANDLED_RXT_EVENTS` – the number of handled CC events related to RXT
- `DROP_RTT_PORT0_REQ` – the number of RTT requests dropped in total from port 0
- `DROP_RTT_PORT1_REQ` – the number of RTT requests dropped in total from port 1
- `DROP_RTT_PORT0_RES` – the number of RTT responses dropped in total from port 0
- `DROP_RTT_PORT1_RES` – the number of RTT responses dropped in total from port 1
- `RTT_GEN_PORT0_REQ` – the number of RTT requests sent in total from port 0
- `RTT_GEN_PORT1_REQ` – the number of RTT requests sent in total from port 1
- `RTT_GEN_PORT0_RES` – the number of RTT responses sent in total from port 0
- `RTT_GEN_PORT1_RES` – the number of RTT responses sent in total from port 1
- `PCC_CNP_COUNT` – the number of CNP received in total, regardless of whether it is handled or ignored

Execution

To use the PCC Counter:

1. Initialize all supported hardware counters. Run:

```
sudo ./pcc_counters.sh set /dev/mst/mt41692_pciconf0
```

> **Info**
>
> Counters are zeroed after each `set` command.

2. Query all supported hardware counters. Run:

```
sudo ./pcc_counters.sh query /dev/mst/mt41692_pciconf0
```

> **Info**
>
> The output counters are counted from the time the `set` command is executed to the time when the `query` command is issued.

Example output:

```
sudo ./pcc_counters.sh query /dev/mst/mt41692_pciconf0
-----------------PCC Counters-----------------
```

```
Counter: MAD_RTT_PERF_CONT_REQ  Value: 000000000028b85b
Counter: MAD_RTT_PERF_CONT_RES  Value: 000000000028b85a
Counter: SX_EVENT_WRED_DROP     Value: 0000000000000000
Counter: SX_RTT_EVENT_WRED_DROP Value: 0000000000000000
Counter: ACK_EVENT_WRED_DROP    Value: 0000000000ccdf4f
Counter: NACK_EVENT_WRED_DROP   Value: 0000000000000000
Counter: CNP_EVENT_WRED_DROP    Value: 0000000000000000
Counter: RTT_EVENT_WRED_DROP    Value: 0000000000000000
Counter: HANDLED_SXW_EVENTS     Value: 000000000932543a
Counter: HANDLED_RXT_EVENTS     Value: 000000000028b85c
Counter: DROP_RTT_PORT0_REQ     Value: 0000000000000000
Counter: DROP_RTT_PORT1_REQ     Value: 0000000000000000
Counter: DROP_RTT_PORT0_RES     Value: 0000000000000000
Counter: DROP_RTT_PORT1_RES     Value: 0000000000000000
Counter: RTT_GEN_PORT0_REQ      Value: 0000000000000000
Counter: RTT_GEN_PORT1_REQ      Value: 000000000028b85c
Counter: RTT_GEN_PORT0_RES      Value: 0000000000000000
Counter: RTT_GEN_PORT1_RES      Value: 000000000028b85d
Counter: PCC_CNP_COUNT          Value: 0000000000000000
```

NVIDIA DOCA Socket Relay

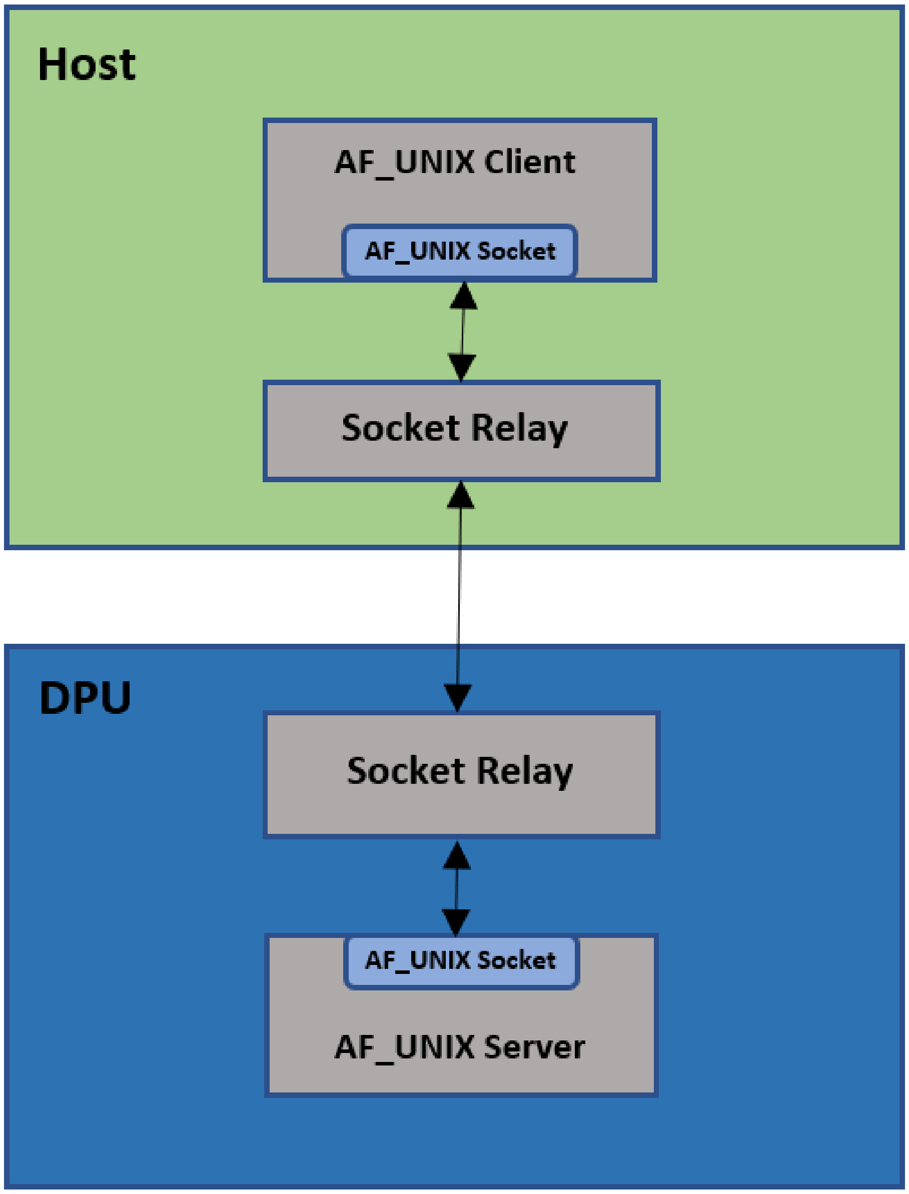
This document describes DOCA Socket Relay architecture, usage, etc.

Introduction

DOCA Socket Relay allows Unix Domain Socket (AF_UNIX family) server applications to be offloaded to the DPU while communication between the two sides is proxied by DOCA Comm Channel.

Socket relay only supports SOCK_STREAM communication with a limit of 512 AF_UNIX application clients.

The tool is coupled to the client AF_UNIX server application. That is, a socket relay instance should be initiated per AF_UNIX server application.



Socket relay is transparent to the application except for the following TCP flows:

- Connection termination must be done by the host side application only
- Once a FIN packet (shutdown system call has been made) is sent by the host side application, data cannot be transferred between the DPU and the host, and the connection must be closed.

The following details the communication flow between the client and server:

- The AF_UNIX client application connects to the socket relay AF_UNIX server in the same way as in the original flow
- The AF_UNIX client application sends SOCK_STREAM packets

- The socket relay (host) AF_UNIX server receives the client application packets, and the Comm Channel client sends them on the channel
- The socket relay (DPU) Comm Channel server receives the client application packets and the AF_UNIX client sends them to the user's AF_UNIX server application

Prerequisites

Windows 10 build 17063 is the minimal Windows version to run DOCA Socket Relay on a Windows host.

Dependencies

NVIDIA® BlueField®-2 firmware version 24.35.1012 or higher.

Execution

To execute DOCA Socket Relay:

```
Usage: doca_socket_relay [DOCA Flags] [Program Flags]

DOCA Flags:
  -h, --help                    Print a help synopsis
  -v, --version                 Print program version information
  -l, --log-level               Set the (numeric) log level for the
program <10=DISABLE, 20=CRITICAL, 30=ERROR, 40=WARNING, 50=INFO,
60=DEBUG, 70=TRACE>
  --sdk-log-level               Set the SDK (numeric) log level for
the program <10=DISABLE, 20=CRITICAL, 30=ERROR, 40=WARNING,
50=INFO, 60=DEBUG, 70=TRACE>
  -j, --json <path>             Parse all command flags from an
input json file

Program Flags:
  -s, --socket                  Unix domain socket path, host side
will bind to and DPU connect to
  -n, --cc-name                 Comm Channel service name
  -p, --pci-addr             DOCA Comm Channel device PCI address
  -r, --rep-pci              DOCA Comm Channel device representor
PCI address (needed only on DPU)
```

For example (DPU side):

```
doca_socket_relay -s /tmp/sr_server.socket -n cc_channel -p
03:00.0 -r b1:00.0
```

To run `doca_socket_relay` using a JSON file:

```
doca_socket_relay --json [json_file]
```

For example:

```
doca_socket_relay --json /tmp/doca_socket_relay.json
```

Arg Parser DOCA Flags

Refer to the DOCA Arg Parser for more information.

Flag Type	Short Flag	Long Flag/ JSON Key	Description	JSON Content
General flags	`h`	`help`	Prints a help synopsis	N/A
	`v`	`version`	Prints program version information	N/A

Flag Type	Short Flag	Long Flag/ JSON Key	Description	JSON Content
	`l`	`log-level`	Set the log level for the application: • DISABLE=10 • CRITICAL=20 • ERROR=30 • WARNING=40 • INFO=50 • DEBUG=60 • TRACE=70 (requires compilation with `TRACE` log level support)	`"log-level" : 60`
	N/A	`sdk-log-level`	SDK log events are currently unsupported for this tool	N/A
	`j`	`json`	Parse all command flags from an input JSON file	N/A
Program flags	`s`	`socket`	AF_UNIX (`SOCK_STREAM`) path. On the host, this is the path of the socket relay AF_UNIX server for the client's application to connect to. On the DPU, this is the path of the client AF_UNIX server application. **Note** This flag is mandatory.	`"socket" : "/tmp/uds-server.socket"`
	`n`	`cc-name`	Comm Channel service name **Note**	`"cc-name" :`

Flag Type	Short Flag	Long Flag/ JSON Key	Description	JSON Content
			This flag is mandatory.	sr_channel
	p	pci-addr	DOCA Comm Channel device PCIe address **Note** This flag is mandatory.	"pci-addr": b1:00.1
	r	rep-pci	DOCA Comm Channel device representor PCIe address **Note** This flag is available and mandatory only on the DPU.	"rep-pci": b1:02.2

NVIDIA DOCA Ngague

NOTE THAT THIS CONFLUENCE PAGE IS NOT READY YET!

Contents:

This document provides instructions on the usage of the ngague tool.

Introduction

ngauge is a tool for probing NIC HW counters, and storing them in an HDF5 format , together with the relevant metadata, for later processing. In addition, the progress and measurement summary are displayed graphically, on a CLI.

Supported hardware are BlueField-3, ConnectX-7, and above.

Prerequisites

NVIDIA® BlueField®-3, ConnectX®-7, and above with firmware version xx.43.1000 or higher, and fwctl driver.

Info

To install the fwctl driver (for host only, for DPU it's already installed), search for a package with "fwctl" and install the package you find.

On deb-based distros, use `apt-cache search fwctl` and for RPM-based distros use `dnf search fwctl`

NOTE: On Ubuntu **20.04** the `fwctl` driver is not loaded automatically, and one needs to `modprobe mlx5_fwctl` after every reboot.

Description

All the configurations are done in the input YAML file.

Start by copying a sample configuration from `/usr/share/doc/ngauge/examples/settings.`

The device to run on should be configured as the PCI address (*e.g.* 0000:03:00.0):

```
device: "0000:03:00.0"
```

The data output path is configured like so (path and prefix to the output file - both are **mandatory**):

```
output:
  path: /path/to/output/directory
  prefix: "ngauge_data_"
```

In the example above, the output will be saved like so: `/path/to/output/directory/ngauge_data_<DATE>_<TIME>.h5`. The explicit output name will be printed after each run.

Run parameters (the most useful of them is the sampling period!) are configured like so:

```
params:
  mode: repetitive  # [repetitive, single]
  period_us: 1e2
```

In the example above, "1e2" means 100 µs. Numbers in decimal or scientific notation are accepted.

The counters to measure are configured like so. The only mandatory configuration for a counter is the Data ID. All the other configurations are optional.

```
counters:
  - id: 0x1020000100000000
    desc: RX bytes port 0
    unit: RX port
    accumulating: false
    normalizer: time  # Normalizer, if present, must be either 'time'
or a number.
```

You can find all supported performance counters in this link: Supported Data IDs

Tip

You may want to install **doca-telemetry-utils** - a tool which can generate counter IDs to be used to configure ngauge. Do it like so: `sudo apt-get install doca-telemetry-utils` or `sudo dnf install doca-telemetry-utils`.

Then run `doca_telemetry_utils -h` for help, and `doca_telemetry_utils get-counters` to get the list of available counters.

Parsing output

A sample plugin, named *simple-plot*, will be installed in `/usr/share/doc/ngauge/examples/plugins`.

This plugin is a basic demonstration of how you can open the output HDF5 file with the data in it and plot it. Besides plotting, many types of analyses can be done on these data. The sample plugin is just a rudimentary demonstration.

```
Usage: /usr/share/doc/ngauge/examples/plugins/simple_plot.py
<ngauge output .h5 file> <counter ID> [<counter ID> ...]
```

Tip

If you only want to plot the results of the very last run, and your output directory is `/tmp` (the default) then you can use this expression to always represent the last results, instead of copy-pasting the file name every time): `"$(ls -1 /tmp/ngauge_data_*.h5 | tail -n1)"`.

Info

The sample plugins are just examples, and should not be considered integral parts of the ngauge tool. therefore you may need to install the dependencies to tun them, such as NumPy, H5py, Matplotlib, plotext, and others separately.

Execution

To run ngague:

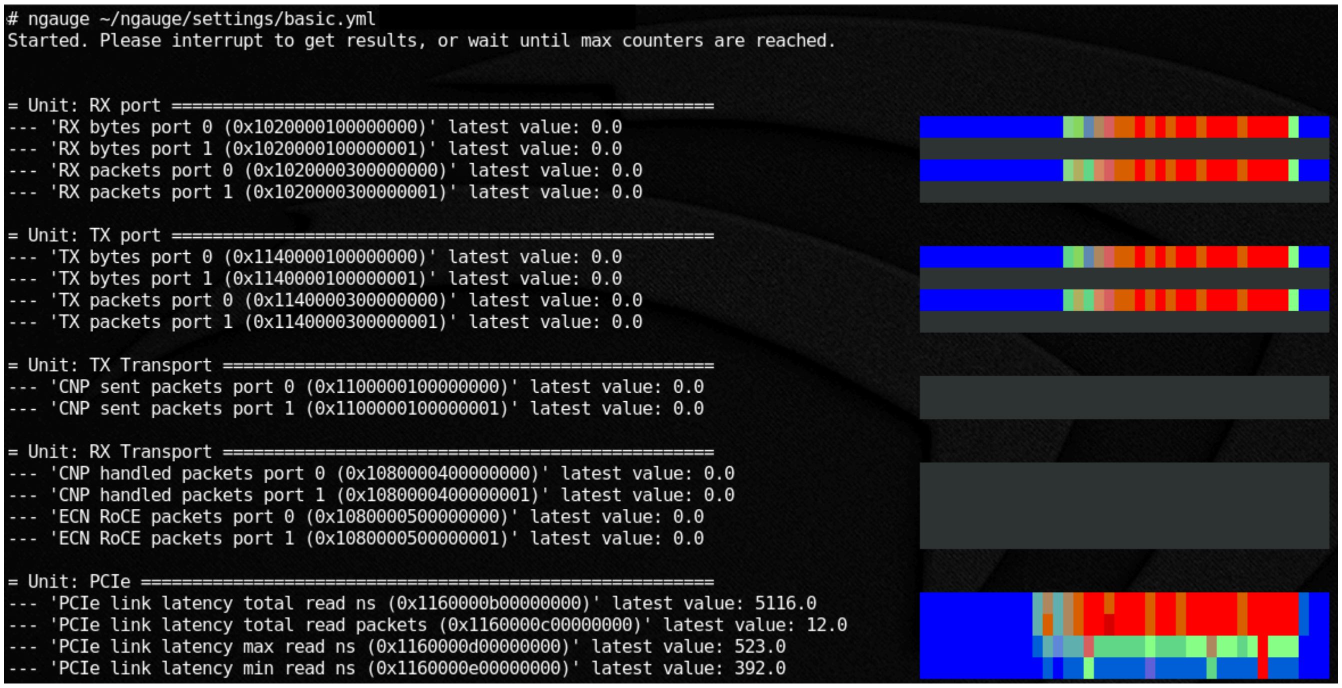
```
Usage: ngague <configuration YAML file>
```

The output will be saved in an HDF5 file (.h5) in the path that you have specified in the configuration YAML.

During the run you will see progress bars for each counter, just as in the image below. The colors symbolize the following:

- Blue - low values (relatively to the other values of the same counter).
- Red - high values (relatively to the other values of the same counter).
- Any color between blue and red - intermediate values.

- Solid gray bars mean that the values of this counter did not change at all during the run.



```
# ngauge ~/ngauge/settings/basic.yml
Started. Please interrupt to get results, or wait until max counters are reached.

= Unit: RX port =====================================================
--- 'RX bytes port 0 (0x1020000100000000)' latest value: 0.0
--- 'RX bytes port 1 (0x1020000100000001)' latest value: 0.0
--- 'RX packets port 0 (0x1020000300000000)' latest value: 0.0
--- 'RX packets port 1 (0x1020000300000001)' latest value: 0.0

= Unit: TX port =====================================================
--- 'TX bytes port 0 (0x1140000100000000)' latest value: 0.0
--- 'TX bytes port 1 (0x1140000100000001)' latest value: 0.0
--- 'TX packets port 0 (0x1140000300000000)' latest value: 0.0
--- 'TX packets port 1 (0x1140000300000001)' latest value: 0.0

= Unit: TX Transport ================================================
--- 'CNP sent packets port 0 (0x1100000100000000)' latest value: 0.0
--- 'CNP sent packets port 1 (0x1100000100000001)' latest value: 0.0

= Unit: RX Transport ================================================
--- 'CNP handled packets port 0 (0x1080000400000000)' latest value: 0.0
--- 'CNP handled packets port 1 (0x1080000400000001)' latest value: 0.0
--- 'ECN RoCE packets port 0 (0x1080000500000000)' latest value: 0.0
--- 'ECN RoCE packets port 1 (0x1080000500000001)' latest value: 0.0

= Unit: PCIe ========================================================
--- 'PCIe link latency total read ns (0x1160000b00000000)' latest value: 5116.0
--- 'PCIe link latency total read packets (0x1160000c00000000)' latest value: 12.0
--- 'PCIe link latency max read ns (0x1160000d00000000)' latest value: 523.0
--- 'PCIe link latency min read ns (0x1160000e00000000)' latest value: 392.0
```