NVIDIA DOCA DMA Copy

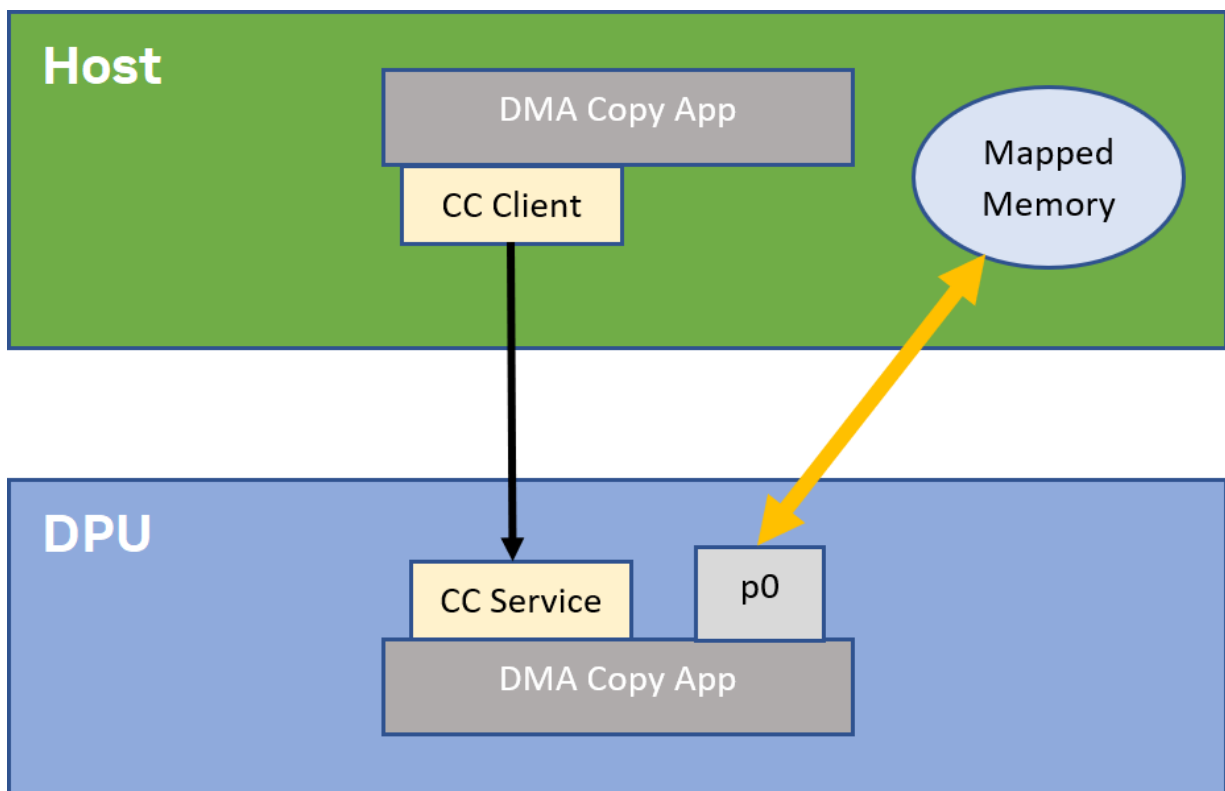Application Guide

Table of Contents

Chapter 1. Introduction

DOCA DMA (direct memory access) Copy application transfers files (data path) up to 1MB in size between the DPU and the x86 host using the DOCA DMA library which provides an API to copy data between DOCA buffers using hardware acceleration, supporting both local and remote memory.

DOCA DMA allows complex memory copy operations to be easily executed in an optimized, hardware-accelerated manner.
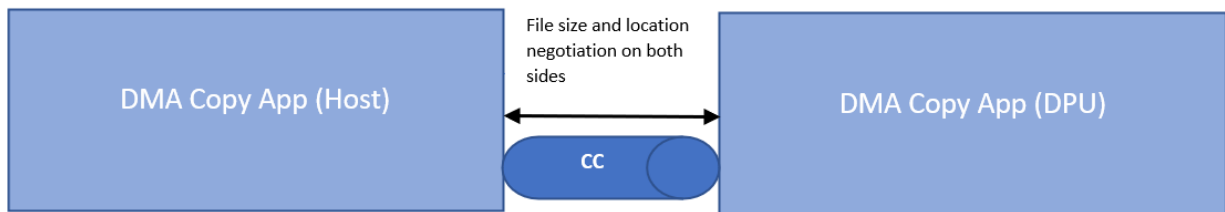
Chapter 2.   System Design

DOCA DMA Copy is designed to run on both the instances of the BlueField-2 DPU and x86 host. The DPU application must be the first to spawn as it opens the Comm Channel (CC) service between the two sides on which all the necessary DMA library configuration files (control path) are transferred.
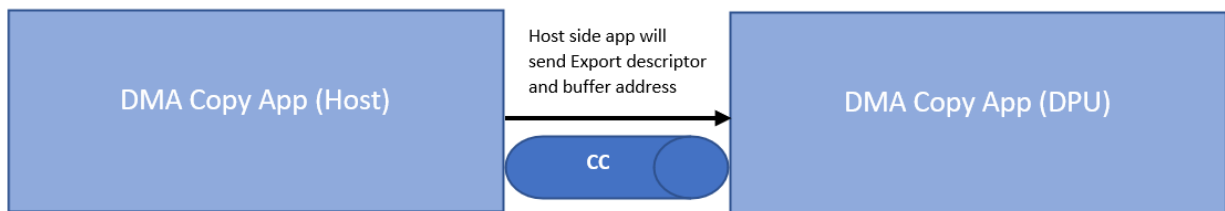
Chapter 3. Application Architecture

DOCA DMA Copy runs on top of DOCA DMA to read/write directly from the host's memory without any user/kernel space context switches, allowing for a fast memory copy.
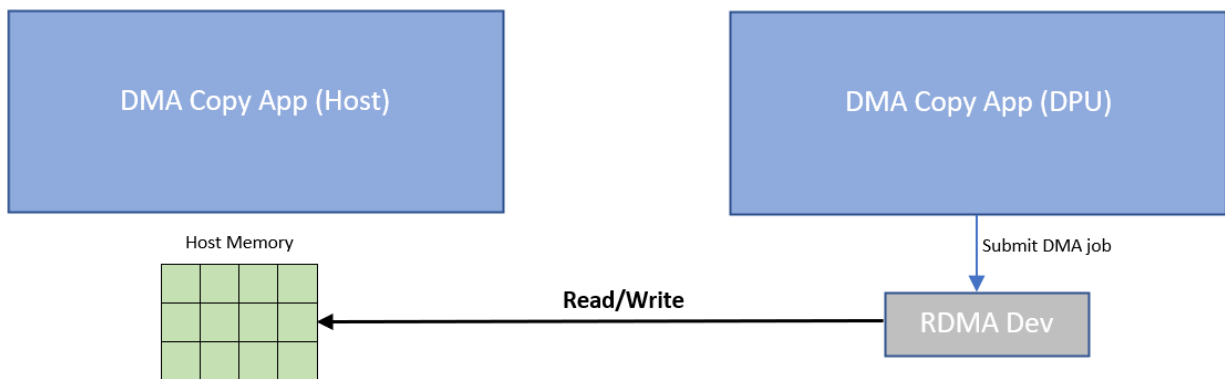
First stage

DMA Copy App (Host)	File size and location negotiation on both sides	DMA Copy App (DPU)
	CC	

Second stage

DMA Copy App (Host)	Host side app will send Export descriptor and buffer address	DMA Copy App (DPU)
	CC	

Third stage

DMA Copy App (Host)		DMA Copy App (DPU)

Host Memory

Read/Write

Submit DMA job

RDMA Dev

1. The two sides initiate a short negotiation in which the file size and location are determined.
2. Host side creates the export descriptor with `doca_mmap_export()` and sends it with the local buffer address and length on the Comm Channel to the DPU side application.

3. DPU side application uses the received export descriptor to create a remote memory map locally with `doca_mmap_create_from_export()` and the host buffer information to create a remote DOCA buffer. From this point on, the DPU side application has all the needed memory information and the DMA copy will take place.

Chapter 4. DOCA Libraries

This application leverages following DOCA libraries:

▶ DOCA Comm Channel

▶ DOCA DMA Programming Guide

Chapter 5. Configuration Flow

1. Parse application argument.

   a). Initialize arg parser resources and register DOCA general parameters.
   ```
   doca_argp_init();
   ```
   b). Register application parameters.
   ```
   register_dma_copy_params();
   ```
   c). Parse app flags.
   ```
   doca_argp_start();
   ```
2. Initialize DOCA App Shield lib context.
   ```
   init_cc();
   ```
   a). Create Comm Channel endpoint.

   b). Parse user PCIe address for Comm Channel device.

   c). Open Comm Channel DOCA device.

   d). Parse user PCIe address for Comm Channel device representor (on DPU side).

   e). Open Comm Channel DOCA device representor (on DPU side).

   f). Set Comm Channel endpoint properties.
3. Open the DOCA hardware device from which the copy would be made.
   ```
   open_dma_device();
   ```
   a). Parse the PCIe address provided by the user.

   b). Create a list of all available DOCA devices.

   c). Find the appropriate DOCA device according to specific properties.

   d). Open the device.
4. Create all required DOCA core objects.
   ```
   create_core_objects();
   ```
5. Initiate DOCA core objects.
   ```
   init_core_objects();
   ```
6. Start host/DPU DMA copy.

   a). Host-side application:
   ```
   host_start_dma_copy();
   ```
   i.   Start negotiation with the DPU side application for the location and size of the file.

   ii. Allocate memory for the DMA buffer.

   iii. Export the memory map and send the output (export descriptor) to the DPU side application.

  iv. Send the host local buffer memory address and length on the Comm Channel to the DPU side application.

  v. Wait for the DPU to notify that DMA Copy has ended.

  vi. Close all memory objects.

  vii. Clean resources.

b). DPU-side application:

```
dpu_start_dma_copy();
```

  i. Start negotiation with the host side application for file location and size.

  ii. Allocate memory for the DMA buffer.

  iii. Receive the export descriptor on the Comm Channel.

  iv. Create the DOCA memory map for the remote buffer on the host.

  v. Receive the host buffer information on the Comm Channel.

  vi. Create two DOCA buffers, one for the remote (host) buffer and one for the local buffer.

  vii. Submit the DMA job into the work queue.

  viii. Send a host message to notify that DMA copy ended.

  ix. Clean resources.

7. Destroy Comm Channel.

```
destroy_cc();
```

8. Destroy DOCA core objects.

```
destroy_core_objects();
```

9. Arg parser destroy.

```
doca_argp_destroy();
```

Chapter 6. Dependencies

The minimum required firmware version is 24.32.1010.

Chapter 7. Running the Application

1. Refer to the following documents:

   ▶ NVIDIA DOCA Installation Guide for Linux for details on how to install BlueField-related software.

   ▶ NVIDIA DOCA Troubleshooting Guide for any issue you may encounter with the installation, compilation, or execution of DOCA applications.

   ▶ NVIDIA DOCA Applications Overview for additional compilation instructions and development tips of DOCA applications.

2. The DMA Copy example binary is located under `/opt/mellanox/doca/applications/dma_copy/bin/doca_dma_copy`. To build all the applications together, run:

```
cd /opt/mellanox/doca/applications/
meson build
ninja -C build
```

3. To build only the DMA Copy application:

   a). Edit the following flags in `/opt/mellanox/doca/applications/meson_option.txt`:

   ▶ Set `enable_all_applications` to `false`

   ▶ Set `enable_dma_copy` to `true`

   b). Run the commands in step 2.

   > 📑 **Note:** `doca_dma_copy` will be created under `./build/dma_copy/src/`.

   Application usage:

```
Usage: doca_dma_copy [DPDK Flags] -- [DOCA Flags] [Program Flags]

DOCA Flags:
  -h, --help                        Print a help synopsis
  -v, --version                     Print program version information
  -l, --log-level                   Set the log level for the program
 <CRITICAL=20, ERROR=30, WARNING=40, INFO=50, DEBUG=60>

Program Flags:
  -f, --file                        Full path to file to be copied/created after
 a successful DMA copy
  -d, --dev-pci                     Comm Channel DOCA device PCI address
  -r, --rep-pci                     Comm Channel DOCA device representor PCI
 address
```

4. Running the application on BlueField, CLI example:

```
/opt/mellanox/doca/applications/dma_copy/bin/doca_dma_copy -d 03:00.1 -r b1:00.1
 -f /tmp/file_to_read.txt
```

5. Running the application on the host, CLI example:

```
/opt/mellanox/doca/applications/dma_copy/bin/doca_dma_copy -d b1:00.1 -f /tmp/
file_to_create.txt
```

> 📝 **Note:** Refer to section "Running DOCA Application on Host" in NVIDIA DOCA Virtual Functions User Guide.

6. To run `doca_dma_copy` using a JSON file:

```
doca_dma_copy --json [json_file]
```

For example:

```
cd /opt/mellanox/doca/applications/dma_copy/bin
./doca_dma_copy --json /root/dma_copy_params.json
```

Chapter 8.   Arg Parser DOCA Flags

Refer to [NVIDIA DOCA Arg Parser User Guide](#) for more information.

Flag Type	Short Flag	Long Flag/JSON Key	Description	JSON Content
General flags	`l`	`log-level`	Set the log level for the application: ▶ CRITICAL=20 ▶ ERROR=30 ▶ WARNING=40 ▶ INFO=50 ▶ DEBUG=60	`"log-level": 60`
	`v`	`version`	Print program version information	N/A
	`h`	`help`	Print a help synopsis	N/A
Program flags	`f`	`file`	Full path to file to be copied/created after a successful copy 💬 **Note:** This is a mandatory flag.	`"file": "/tmp/sample.txt"`
	`p`	`dev-pci`	Comm Channel DOCA device PCIe address 💬 **Note:** This is a	`"dev-pci": "b1:00.0"`

Flag Type	Short Flag	Long Flag/JSON Key	Description	JSON Content
			mandatory flag.	
	r	rep-pci	Comm Channel DOCA device representor PCIe address **Note:** This is a mandatory flag only on the DPU.	`"rep-pci": "b1:02.0"`

Chapter 9. References

- ► `/opt/mellanox/doca/applications/dma_copy/src/dma_copy.c`
- ► `/opt/mellanox/doca/applications/dma_copy/src/dma_copy_core.c`
- ► `/opt/mellanox/doca/applications/dma_copy/src/dma_copy_core.h`

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com