



Mmap Advise

Table of contents

Introduction

Prerequisites

Architecture

Cache Invalidate Operation

Environment

Objects

Device and Device Representor

Memory Buffers

Configuration Phase

Configurations

Mandatory Configurations

Device Support

Buffer Support

Execution Phase

Tasks

Cache Invalidate Task

Events

State Machine

Idle

Starting

Running

Stopping

Alternative Datapath Options

Samples

Cache Invalidate Sample

Introduction

DOCA Mmap Advise is used to give advanced memory-related instructions to NVIDIA® BlueField® DPUs in order to improve system or application performance.

Note

To use DOCA Mmap Advise with BlueField, the device must be configured to work in DPU mode as described in [BlueField Modes of Operation](#).

The operations in the instructions are meant to influence the performance of the application, but not its semantics. The operations allow an application to inform the NIC how it expects it to use some mapped memory areas, so the BlueField's hardware can choose appropriate optimization techniques.

Prerequisites

DOCA Mmap Advise is a context and follows the architecture of a DOCA Core Context, it is recommended to read the following sections of the DOCA Core page before proceeding:

- [DOCA Core Execution Model](#)
- [DOCA Core Device](#)
- [DOCA Core Memory Subsystem](#)

Architecture

DOCA Mmap Advise is a DOCA Context as defined by DOCA Core. See [DOCA Core Context](#) for more information.

DOCA Mmap Advise currently supports the following list of advised operations:

- Cache Invalidate Operation

Cache Invalidate Operation

When data is processed by BlueField's cores it may be temporarily stored in the cores' system-level cache (i.e., L3 cache). When a cache line is occupied and new data must be written to it, the cache management sub-system evicts the existing data, usually based on LRU policy, by performing a write-back operation to store this data in the main (DDR) memory. When this data is not required to be stored in the BlueField's memory (e.g., it is host data and is no longer needed after it is copied to the host's memory), the cache's write-back operation wastes memory bandwidth that reduces overall system performance, which is undesirable. The simplest to avoid this write-back operation is to mark the appropriate cache lines as "invalid". This enables their immediate reuse, without additional operations.

The cache invalidate operation facilitates invalidating a set of cache lines.

Environment

Applications based on DOCA Mmap Advise can run on the BlueField target.

Objects

Device and Device Representor

The MMAP Advise context requires a DOCA Device to operate. The device is used to access memory and perform the copy operation. See [DOCA Core Device Discovery](#).

Info

For the same DPU, it does not matter which device is used (i.e., PF, VF, SF) as all these devices utilize the same hardware components.

Note

The device must stay valid for as long as the MMAP Advise instance is not destroyed.

Memory Buffers

The cache invalidate task requires one DOCA Buffer containing the address space to invalidate depending on the allocation pattern of the buffers (refer to the table in section "[Inventory Types](#)"). To find what kind of memory is supported, refer to the table in section "[Buffer Support](#)".

Buffers must not be modified or read during the cache invalidate operation.

Configuration Phase

To start using the context, users must go through a configuration phase as described in [DOCA Core Context Configuration Phase](#).

This section describes how to configure and start the context, to allow execution of tasks and retrieval of events.

Configurations

The context can be configured to match the application's use case.

To find if a configuration is supported, or what the min/max value for it is, refer to section "[Device Support](#)".

Mandatory Configurations

These configurations are mandatory and must be set by the application before attempting to start the context:

- At least one task/event type must be configured. See configuration of tasks and/or events in sections "[Tasks](#)" and "[Events](#)" respectively for information.
- A device with appropriate support must be provided upon creation

Device Support

DOCA Mmap Advise requires a device to operate. To pick a device, refer to [DOCA Core Device Discovery](#).

As device capabilities may change (see [DOCA Core Device Support](#)), it is recommended to select your device using the following method:

- `doca_mmap_advise_cap_task_cache_invalidate_is_supported`

Some devices expose different capabilities as follows:

- Maximum cache invalidate buffer size may differ.

Buffer Support

Tasks support buffers with the following features:

Buffer Type	Buffer
Local mmap buffer	Yes
MMAP from PCIe export buffer	No
MMAP from RDMA export buffer	No
Linked list buffer	No

Execution Phase

This section describes execution on the CPU using [DOCA Core Progress Engine](#).

Tasks

DOCA Mmap Advise exposes asynchronous tasks that leverage DPU hardware according to the DOCA Core architecture. See [DOCA Core Task](#) for information.

Cache Invalidate Task

The cache invalidate task facilitates invalidating a set of cache lines, preventing them from being written back to the RAM (thus increasing performance).

Task Configuration

Description	API to Set the Configuration	API to Query Support
Enable the task	<code>doca_mmap_advise_task_invalidate_cache_set_conf</code>	<code>doca_mmap_advise_cap_task_cache_invalidate_is_supported</code>
Number of tasks	<code>doca_mmap_advise_task_invalidate_cache_set_conf</code>	-
Maximal buffer size	-	<code>doca_mmap_advise_task_cache_invalidate_get_max_buf_size</code>
Maximal buffer list size	-	-

Task Input

Common input as described in [DOCA Core Task](#).

Name	Description
buffer	Buffer that points to the memory to be invalidated

Task Output

Common output as described in [DOCA Core Task](#).

Task Completion Success

After the task is completed successfully:

- The cache is invalidated

Task Completion Failure

If the task fails midway:

- The context may enter stopping state, if a fatal error occurs
- The cache is not invalidated

Task Limitations

- The operation is not atomic
- Once the task has been submitted, the buffer should not be read/written to
- Other limitations are described in [DOCA Core Task](#)

Events

DOCA Mmap Advise exposes asynchronous events to notify on changes that happen unexpectedly, according to [DOCA Core architecture](#).

The only events DOCA Mmap Advise exposes are common events as described in [DOCA Core Event](#).

State Machine

DOCA Mmap Advise context follows the context state machine as described in [DOCA Core Context State Machine](#).

The following section describes how to move states and what is allowed in each state.

Idle

In this state it is expected that the application:

- Destroys the context
- Starts the context

Allowed operations:

- Configuring the context according to section "[Configurations](#)"
- Starting the context

It is possible to reach this state as follows:

Previous State	Transition Action
None	Create the context
Running	Call stop after making sure all tasks have been freed
Stopping	Call progress until all tasks are completed and freed

Starting

This state cannot be reached.

Running

In this state, it is expected that the application:

- Allocates and submits tasks
- Calls progress to complete tasks and/or receive events

Allowed operations:

- Allocating a previously configured task
- Submitting a task
- Calling stop

It is possible to reach this state as follows:

Previous State	Transition Action
Idle	Call start after configuration

Stopping

In this state it is expected that the application:

- Calls progress to complete all in-flight tasks (tasks complete with failure)
- Frees any completed tasks

Allowed operations:

- Call progress

It is possible to reach this state as follows:

Previous State	Transition Action
Running	Call progress and fatal error occurs
Running	Call stop without freeing all tasks

Alternative Datapath Options

DOCA Mmap Advise only supports datapath on the CPU. See section "[Execution Phase](#)".

Samples

Cache Invalidate Sample

The sample illustrates how to invalidate the cache for a memory range after copying it using DOCA DMA.

The sample logic includes:

1. Locating DOCA device.
2. Initializing needed DOCA core structures.
3. Populating DOCA memory map with two relevant buffers.
4. Allocating element in DOCA buffer inventory for each buffer.
5. Initializing DOCA DMA memory copy task object.
6. Initializing DOCA Mmap Advise cache invalidate task object
7. Submitting DMA task.
8. Polling for completion:
 1. Handling DMA task completion – submitting the cache invalidate task in the DMA task completion callback body.
 2. Handling cache invalidate task completion.
9. Polling for completion.
10. Destroying DMA, DOCA MMAP Advise, and DOCA Core objects.

Reference:

- `/opt/mellanox/doca/samples/doca_common/cache_invalidate/cache_inv`
- `/opt/mellanox/doca/samples/doca_common/cache_invalidate/cache_inv`
- `/opt/mellanox/doca/samples/doca_common/cache_invalidate/meson.bui`

Notice
This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality. NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.
NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and

conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document. NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk. NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs. No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product. **Trademarks** NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© Copyright 2024, NVIDIA. PDF Generated on 01/02/2025