NVIDIA DOCA DNS Filter

Reference Application Guide

MLNX-15-060486 _v1.2 | January 2022

Table of Contents

Chapter 1. Introduction

Domain name system (DNS) translates domain names to IP addresses so browsers can load internet resources. Each device connected to the internet has a unique IP address which other machines use to find the device.

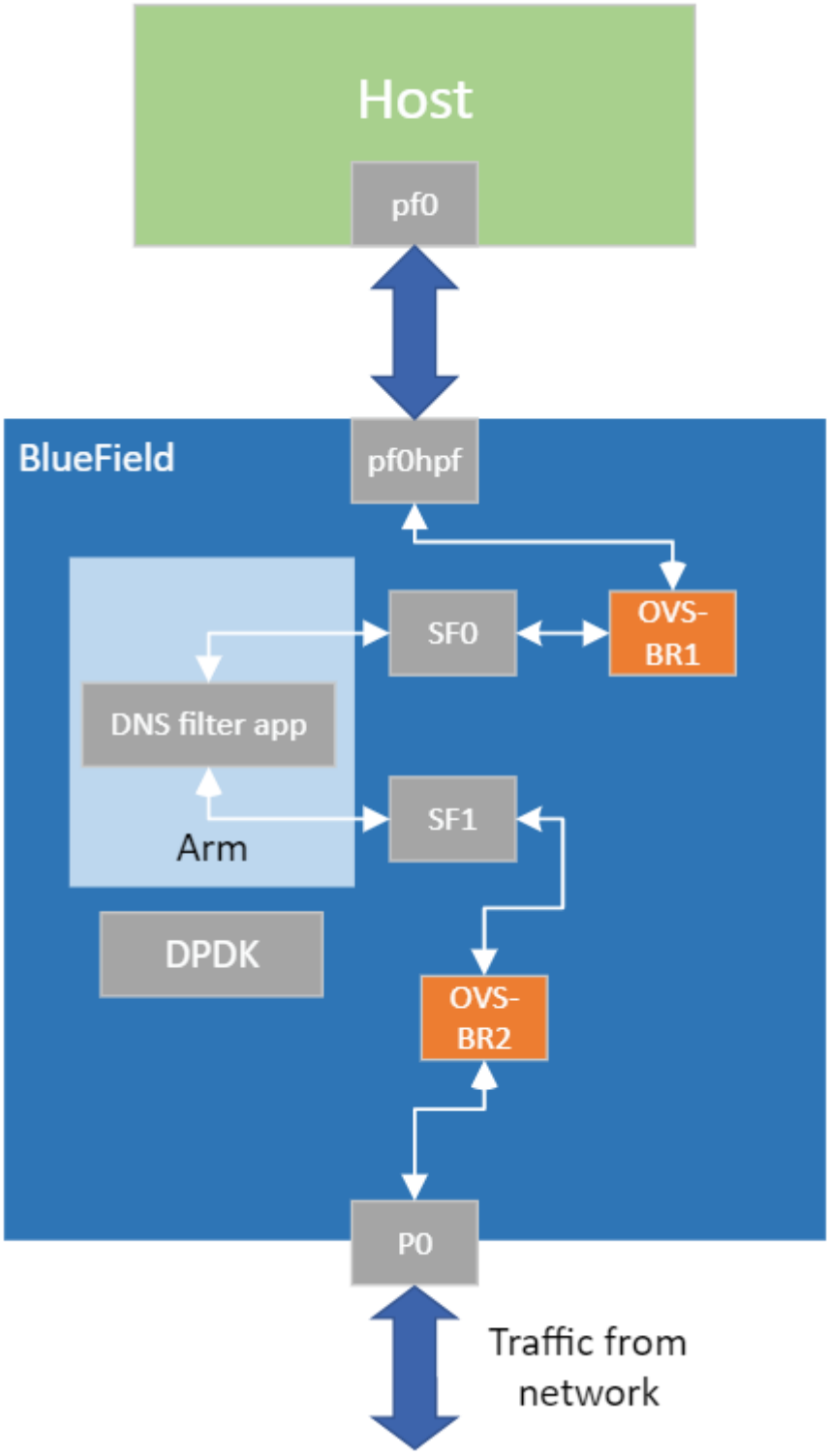
The DNS process includes several steps:

1. Once a user tries to log into a website using a browser, the user's device creates a DNS query and sends it to a DNS resolver.
2. The DNS resolver queries the DNS domain to get an IP address by searching its cache or sending the request to another DNS server.
3. Once a match is found, the DNS resolver returns the correct IP matching the DNS domain.
4. The user can log into the required website using the correct IP.

DNS filter is used to offload DNS requests from the host to the BlueField DPU Arm which allows reducing CPU overhead as Arm allows further DNS processing to be done (e.g., allowlisting, logging, filtering, etc).

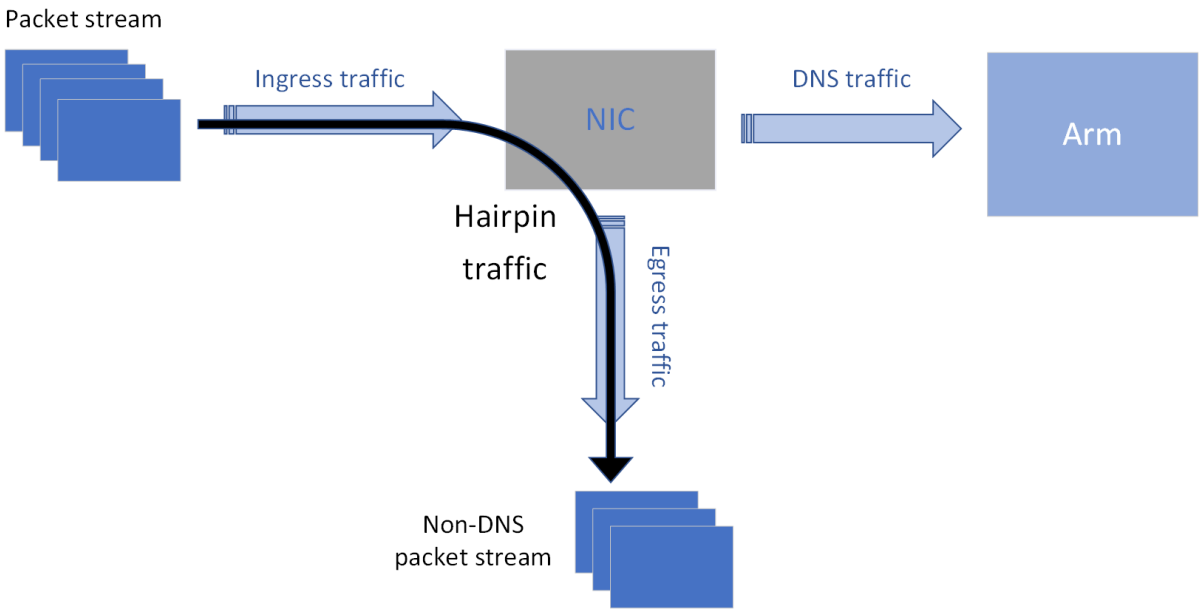
Chapter 2. System Design

The DNS filter application is designed to run as a "bump-on-the-wire" on the BlueField-2 DPU instance. The DPU intercepts the traffic coming (ingress traffic) from the wire and either passes it to the Arm or forwards it to the egress port using hairpin. The decision is made by traffic classification.


Host
pf0
BlueField
pf0hpf
SF0
OVS-
BR1
DNS filter app
SF1
Arm
DPDK
OVS-
BR2
P0
Traffic from
network

Chapter 3. Application Architecture

The DNS filter runs on top of DOCA FLOW to classify DNS requests.



1. Ingress packet types are identified using pipes which encapsulate flow rule matching patterns and actions.
2. The DNS filter application builds two pipes for each port (DNS pipe and hairpin pipe). Every pipe includes exactly one entry.
3. The DNS pipe matches only DNS traffic and FORWARDS it to the Arm. The hairpin pipe matches every packet (no misses). The DNS pipe serves as a root pipe and the hairpin pipe serves as a FORWARDING miss component to the DNS pipe. Therefore, every received packet is checked first against the DNS pipe, and if there is a match then it is forwarded to the Arm. Otherwise (miss case), it is forwarded to the hairpin pipe and then is matched.

Chapter 4. Configuration Flow

1. Parse application argument.
   ```
   arg_parser_init();
   ```
   a). Initialize arg parser resources.
   b). Register DOCA general flags.
   ```
   arg_parser_start();
   ```
   c). Parsing DPDK flags and calling `rte_eal_init()` function.
2. DPDK initialization.
   ```
   dpdk_init();
   ```
   a). Initialize DPDK ports, including mempool allocation.
   b). Initialize hairpin queues if needed.
   c). Binds hairpin queues of each port to its peer port.
3. DNS filter initialization.
   ```
   dpdk_dns_filter_init();
   ```
   a). DOCA flow and DOCA flow port initialization.
   b). Creates hairpin pipe for both ports. This pipe includes one entry that matches every type of packet (no misses) and forwards it to the egress port through hairpin.
   c). Creates DNS pipe, that serves as a root pipe, for both ports. The built pipe has one entry for matching DNS traffic and forwarding it to Arm. In addition, the hairpin pipe serves for a FORWARDING if the DNS entry does not match (i.e., for each non-DNS packet, packets are hairpined).
4. Processing packets.
   ```
   process_packets();
   ```
   a). All received packets on Arm are DNS packets, while non-DNS packets are forwarded to the egress port using hairpin allowing DNS packets to be filtered.
5. DNS filter destroy.
   ```
   dns_filter_destroy();
   ```
   a). Frees all allocated resources.

Chapter 5. Running Application on BlueField

1. Please refer to the DOCA Installation Guide for details on how to install BlueField related software.
2. The DNS filter example binary is located under `/opt/mellanox/doca/examples/dns_filter/bin/doca_dns_filter`. To re-build the application:

   a). Run:

   ```
   cd /opt/mellanox/doca/examples/dns_filter/src
   meson /tmp/build
   ninja -C /tmp/build
   ```

   `doca_dns_filter` will be created under `/tmp/build`.

   b). The build process depends on the `PKG_CONFIG_PATH` environment variable to locate the DPDK libraries. If the variable was accidently corrupted, and the build fails, run the following command:

   - For Ubuntu:

     ```
     export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/opt/mellanox/dpdk/lib/aarch64-linux-gnu/pkgconfig
     ```

   - For CentOS:

     ```
     export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/opt/mellanox/dpdk/lib64/pkgconfig
     ```

3. Pre-run setup.

   The DNS filter example is based on DPDK libraries. Therefore, the user is required to provide DPDK flags, and allocate huge pages. Run:

   ```
   echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
   ```

4. To run the application:

   ```
   Usage: doca_dns_filter [DPDK Flags] -- [DOCA Flags]
   DOCA Flags:
     -h, --help                    Print a help synopsis
     -l, --log-level               Set the log level for the app <CRITICAL=0,
    DEBUG=4>
   ```

   For example:

   ```
   /opt/mellanox/doca/examples/dns_filter/bin/doca_dns_filter -a
    auxiliary:mlx5_core.sf.4 -a auxiliary:mlx5_core.sf.5 -- -l 3
   ```

   Or using a JSON file:

   ```
   doca_dns_filter --json [json_file]
   ```

   For example:

```
/opt/mellanox/doca/examples/dns_filter/bin/doca_dns_filter --json /root/
dns_filter_params.json
```

Note: Sub-Functions must be enabled according to Scalable Function Setup Guide.

Note: The flags `-a auxiliary:mlx5_core.sf.4 -a auxiliary:mlx5_core.sf.5` are necessary for proper usage of the application. Modifying these flags results in unexpected behavior as only two ports are supported. The SF numbers are is arbitrary and configurable.

For additional information on available flags for DPDK, use `-h` before the `--` separator:

```
/opt/mellanox/doca/examples/dns_filter/bin/doca_dns_filter -h
```

For information on available flags for the application, use `-h` after the `--` separator:

```
/opt/mellanox/doca/examples/dns_filter/bin/doca_dns_filter -- -h
```

Chapter 6. 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
DPDK flags	a	devices	Add a PCIe device into the list of devices to probe.	`"devices": [ { "device": "sf", "id": "4", "sft": true }, { "device": "sf", "id": "5", "sft": true }, ]`
General flags	l	log-level	Sets the log level for the application: ▶ CRITICAL=0 ▶ ERROR=1 ▶ WARNING=2 ▶ INFO=3 ▶ DEBUG=4	`"log-level": 4`
	h	help	Print a help synopsis	N/A

Chapter 7. Running Application on Host

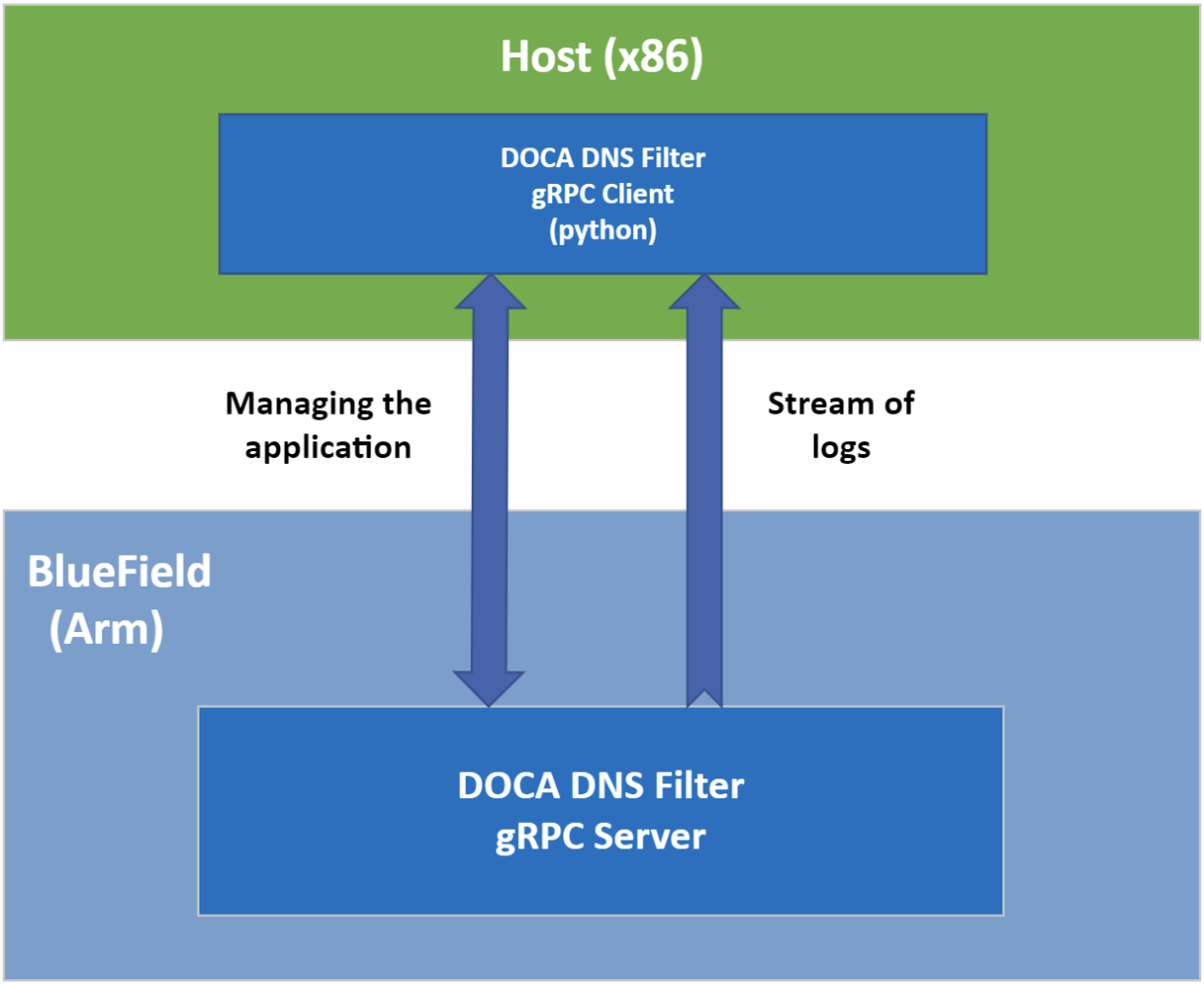
x86 CLI example:

```
/opt/mellanox/doca/examples/dns_filter/sbin/doca_dns_filter -a 03:00.3 -a 03:00.4 -c
 0xff -- -l 4
```

Refer to section "Running DOCA Application on Host" in NVIDIA DOCA Virtual Functions User Guide.

Chapter 8. Managing gRPC-Enabled Application from Host

For instructions on running the gRPC application server on the BlueField, refer to NVIDIA DOCA gRPC Infrastructure User Guide.



To run the Python client of the gRPC-enabled application:

```
./doca_dns_filter_gRPC_client.py -d/--debug <server address[:server port]>
```

For example:

```
/opt/mellanox/doca/examples/dns_filter/bin/grpc/client/
doca_dns_filter_gRPC_client.py 192.168.104.2
```

Chapter 9. References

- /opt/mellanox/doca/examples/dns_filter/src/dns_filter.c
- /opt/mellanox/doca/examples/dns_filter/src/grpc/dns_filter.proto

Chapter 10. Running Application on NVIDIA Converged Accelerator

This section details the steps necessary for running the DNS filter application on NVIDIA converged accelerator.

The DNS-filter application running on the converged accelerator has the same logic as described in previous sections of this page. However, instead of processing the DNS packets in the Arm, the packets are copied to the GPU memory for further processing. To make use of the GPU's capabilities, several steps must be taken.

1. Refer to the DOCA Installation Guide for instructions on installing NVIDIA driver for CUDA and a CUDA-repo on your setup.
2. Create sub-functions and configure the OVS according to Scalable Function Setup Guide.
3. The meson version must at at least be 0.59.1 in order for meson to identify the CUDA compiler. To set up the right meson version:

   a). Download the `*.tar` file of the right meson version (0.59.01) from this link.

   b). Copy the `*.tar` file onto the BlueField.

   c). Untar the file:

   ```
   tar -xzf meson-0.59.1.tar.gz
   ```

   This untars it to the current directory. After untarring the file, a Python script called `meson.py` is extracted under `<path-to-current directory>/meson-0.59.1`. Use this Python script instead of the installed meson version in the system. Going forward, this section refers to "path to the current directory" just as "path" when using meson.

10.1. Compiling and Running Application

Since there is no pre-compiled DNS filter application binary provided that uses the GPU support, you must compile it and run it. All the sources needed for building, compiling, and running the application with GPU support are found under `/opt/mellanox/doca/examples/dns_filter/src`.

1. Set `gpu_support` to `true` in the application's `meson_options.txt` file found at `/opt/mellanox/doca/examples/dns_filter/src/meson_options.txt`.

2. Setup CUDA path:

```
export CPATH=/usr/local/cuda/targets/sbsa-linux/include:$CPATH
export LD_LIBRARY_PATH=/usr/local/cuda/targets/sbsa-linux/lib:$LD_LIBRARY_PATH
export PATH=/usr/local/cuda/bin:/usr/local/cuda-11.4/bin:$PATH
```

3. To build the application, run:

```
<path>/meson-0.59.1/meson.py /tmp/build
ninja -C /tmp/build
```

`doca_dns_filter` is created under `/tmp/build`.

Note: If CUDA-11.4 is installed, the version of CUDA in the meson file must be modified to match it (11.4).

4. The DNS filter example is a DPDK application. Therefore, the user is required to provide DPDK flags and allocate huge pages. Run:

```
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

5. To run the application, follow the steps in Running Application on BlueField.

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 nor any of its direct or indirect subsidiaries and affiliates (collectively: "NVIDIA") make no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assume 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.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

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, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of Mellanox Technologies Ltd. and/or NVIDIA Corporation in the U.S. and in other countries. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world¬wide basis. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

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