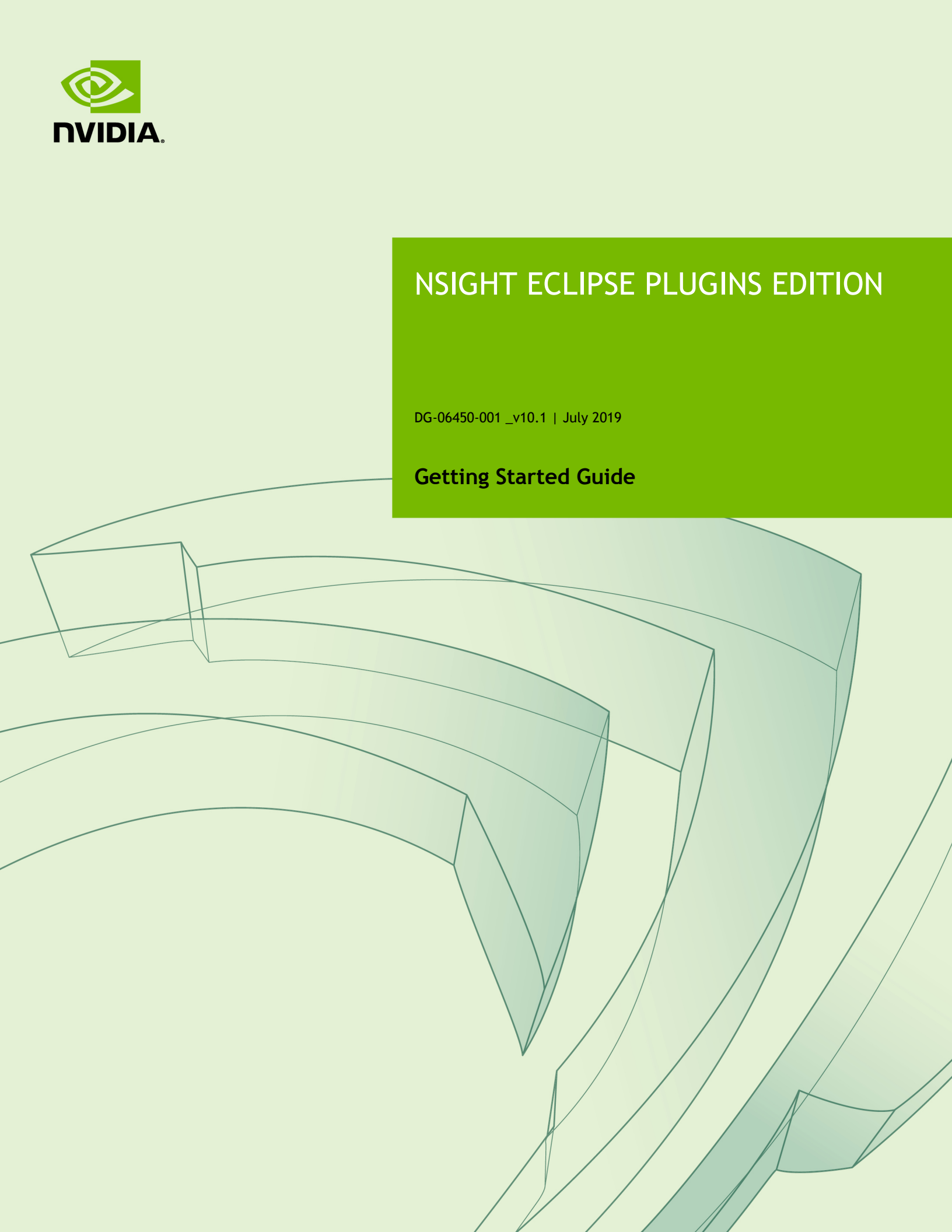


NSIGHT ECLIPSE PLUGINS EDITION

DG-06450-001 _v10.1 | July 2019

Getting Started Guide

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1. INTRODUCTION

This guide introduces Nsight Eclipse Plugins Edition and provides instructions necessary to start using this tool. Nsight Eclipse is based on Eclipse CDT project. For a detailed description of Eclipse CDT features consult the integrated help "C/C++ Development User Guide" available from inside Nsight (through Help->Help Contents menu).

1.1. About Nsight Eclipse Plugins Edition

NVIDIA® Nsight™ Eclipse Edition is a unified CPU plus GPU integrated development environment (IDE) for developing CUDA® applications on Linux and Mac OS X for the x86, POWER and ARM platforms. It is designed to help developers on all stages of the software development process. Nsight Eclipse Plugins can be installed on vanilla Eclipse using the standard Help->Install New Software.. Menu. The principal features are as follows:

- Edit, build, debug and profile CUDA-C applications
- CUDA aware source code editor – syntax highlighting, code completion and inline help
- Graphical user interface for debugging heterogeneous applications
- Profiler integration – Launch visual profiler as an external application with the CUDA application built in this IDE to easily identify performance bottlenecks

For more information about Eclipse Platform, visit http://eclipse.org

Chapter 2.
USING NSIGHT ECLIPSE EDITION

2.1. Installing Nsight Eclipse Edition

Nsight Eclipse Plugins archive is part of the CUDA Toolkit. Nsight Eclipse Plugins archive can be installed using the Help -> Install New Software... Menu on Eclipse

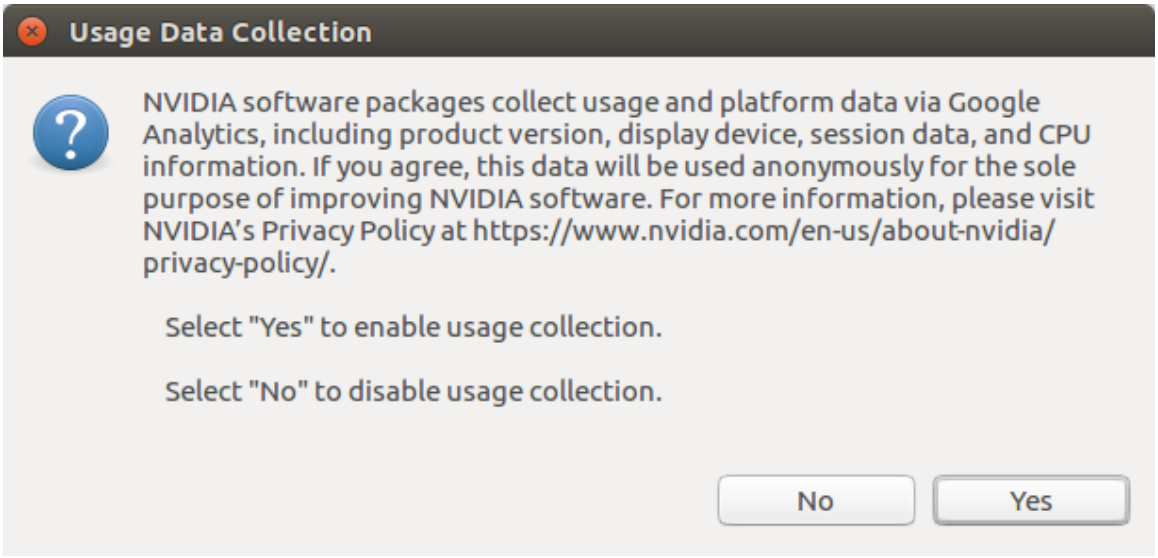
2.1.1. Installing CUDA Toolkit

To install CUDA Toolkit:

1. Visit the NVIDIA CUDA Zone download page:
   http://www.nvidia.com/object/cuda_get.html
2. Select appropriate operating system. Nsight Eclipse Edition is available in Mac OS X and Linux toolkit packages.
3. Download and install the CUDA Driver.
4. Download and install the CUDA Toolkit.
5. Follow instructions to configure CUDA Driver and Toolkit on your system.

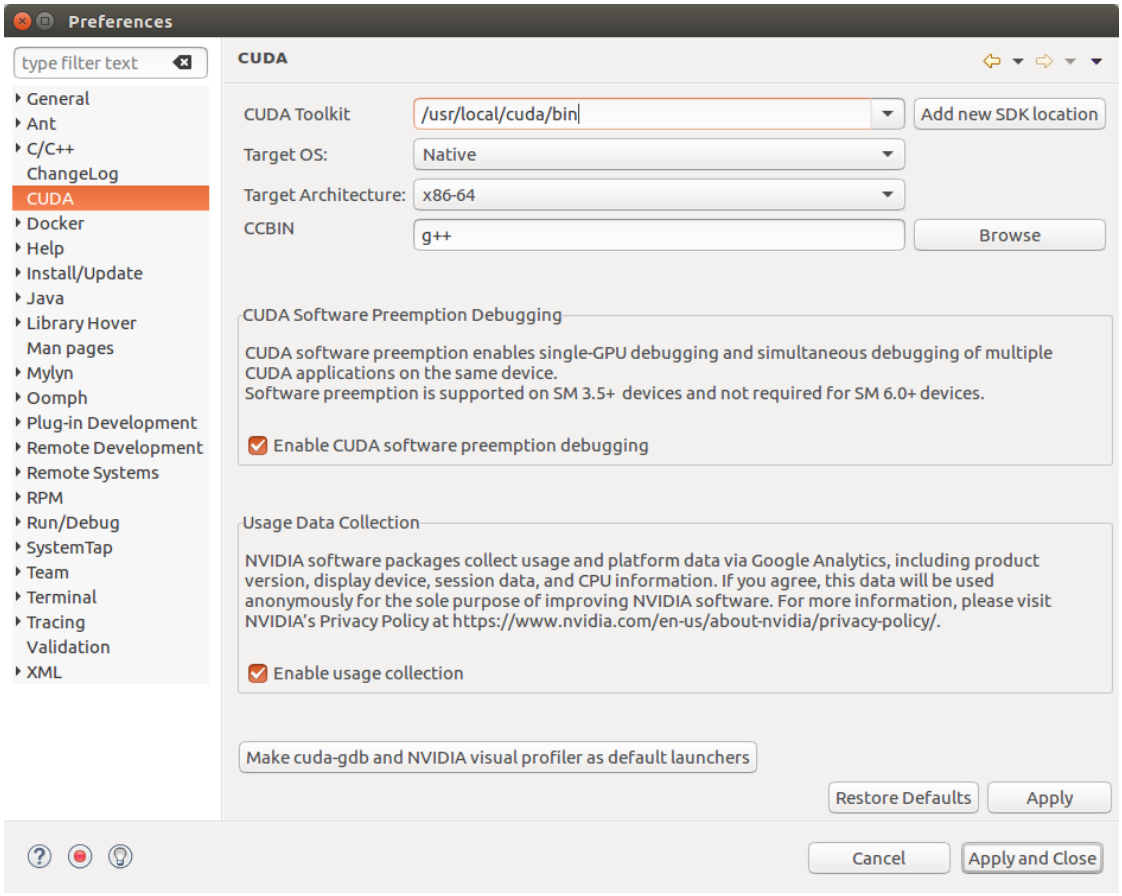
2.1.2. Configure CUDA Toolkit Path

When Eclipse is first launched with Nsight Eclipse plugins in the new workspace, NVIDIA usage data collection dialog will be displayed as below. Click Yes to enable usage collection. This can be disabled later from the CUDA preference page.

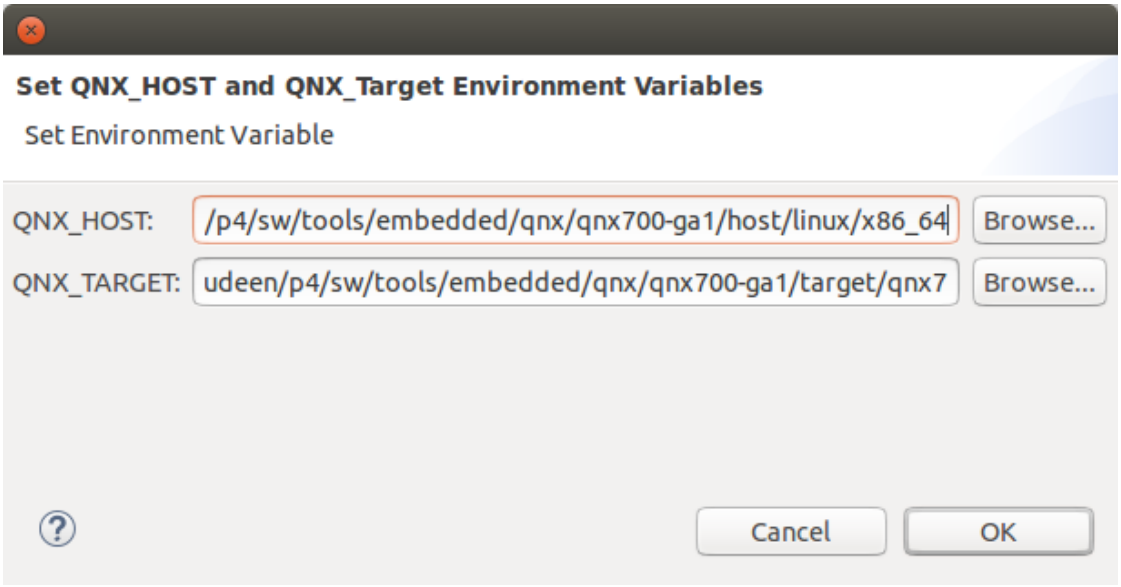


To get started, CUDA Toolkit path must be configured in Eclipse with Nsight Plugins:

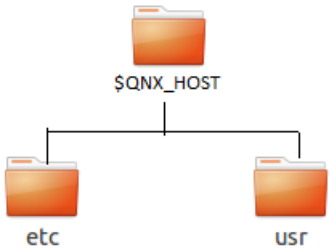
1. Open the Preferences page, Window > Preferences.
2. Go to CUDA toolkit section.
3. Select the CUDA toolkit path to be used by Nsight. CUDA tookits that are installed in the default location will automatically appear.



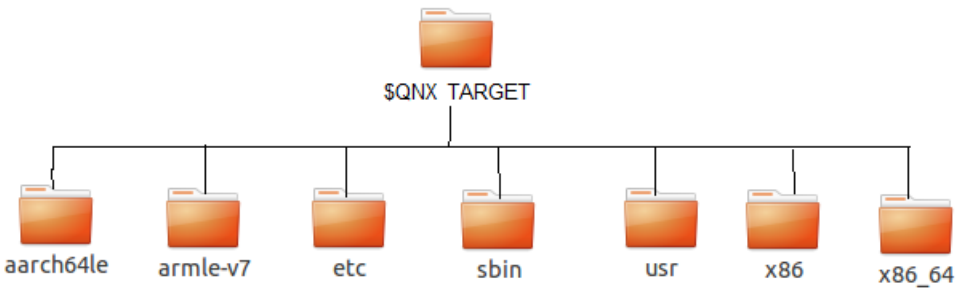
4. CUDA toolkit path can be also specified in the project properties page in order to use different toolkit for a project.
5. Enable usage data collection if you wish to send usage data to NVIDIA.
6. Click on the button to set cuda-gdb and Visual Profiler as the default launchers.
7. **For QNX:** When QNX is selected as Target OS, a dialog will be displayed to set the QNX_HOST and QNX_TARGET environment variables if they were not already set.



QNX_HOST environment variable identifies the directory that holds the host-related components:



QNX_TARGET environment variable identifies the directory that holds the target-related components:



2.2. Nsight Eclipse Main Window

On the first run Eclipse will ask to pick a workspace location. The workspace is a folder where Nsight will store its settings, local files history and caches. An empty folder should be selected to avoid overwriting existing files.

The main Nsight window will open after the workspace location is selected. The main window is divided into the following areas:

- *Editor* - displays source files that are opened for editing.
- *Project Explorer* - displays project files
- *Outline* - displays structure of the source file in the current editor.
- *Problems* - displays errors and warnings detected by static code analysis in IDE or by a compiler during the build.
- *Console* - displays make output during the build or output from the running application.

2.3. Creating a New Project

1. From the main menu, open the new project wizard - **File** > **New...** > **C/C++ Project**
2. Specify the project name and project files location.
3. Specify the project type like executable project.
4. Specify the CUDA toolchain from the list of toolchains.
5. Specify the project configurations on the next wizard page.
6. Complete the wizard.
   The project will be shown in the **Project Explorer** view and source editor will be opened.
7. Build the project by clicking on the hammer button on the main toolbar.

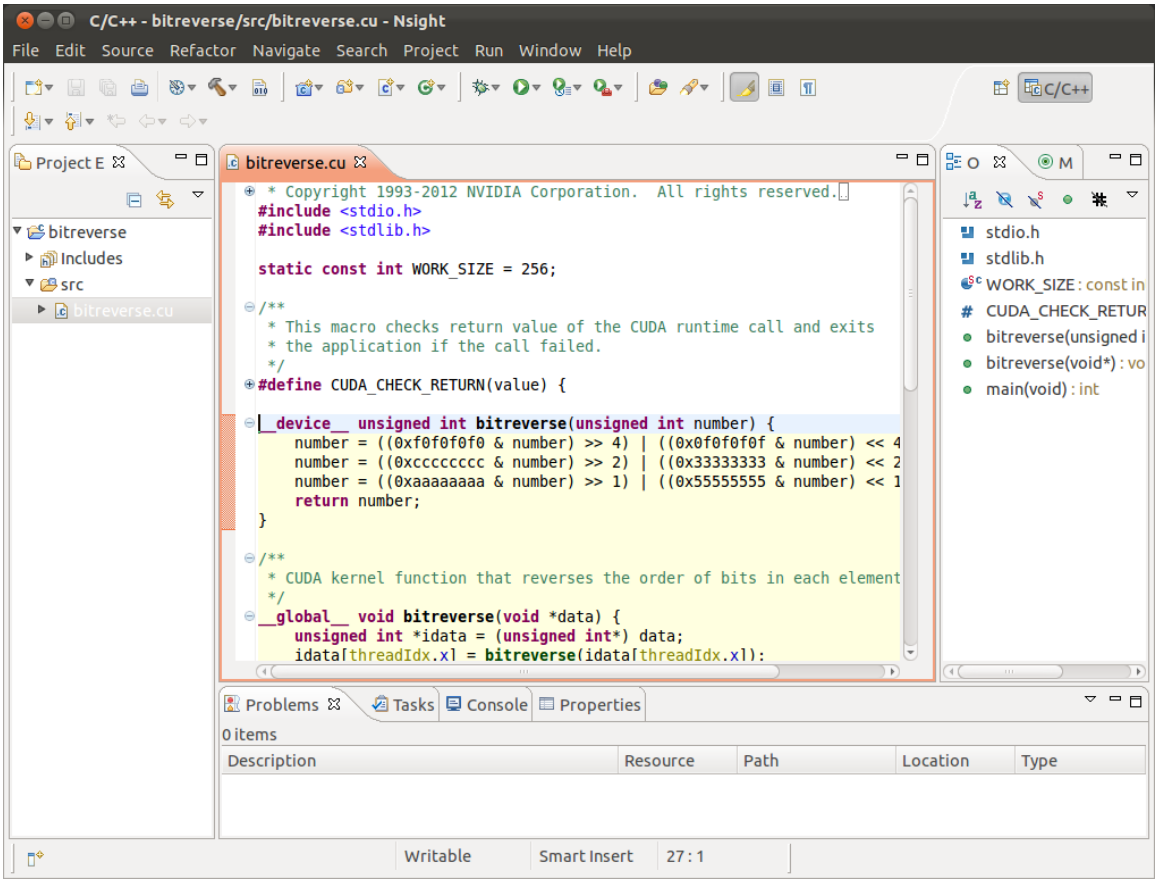


Figure 1 Nsight main window after creating a new project

2.4. Importing CUDA Samples

The CUDA samples are an optional component of the CUDA Toolkit installation. Nsight provides a mechanism to import these samples and work with them easily:

Samples that use the CUDA driver API (suffixed with "Drv") are not supported by Nsight.

1. From the main menu, open the new project wizard - **File** > **New...** > **C/C++ Project**
2. Specify the project name and project files location.
3. Select **Import CUDA Sample** under **Executable** in the **Project type** tree.
4. Select CUDA toolchain from the Toolchains option. location.
5. On the next wizard page select project sample you want to import. Also select the target CPU architecture. Press **Next...**
6. Specify the project parameters on the next wizard page.
7. Complete the wizard.

The project will be shown in the **Project Explorer** view and source editor will be opened.

8. Build the project by clicking on the hammer button on the main toolbar.

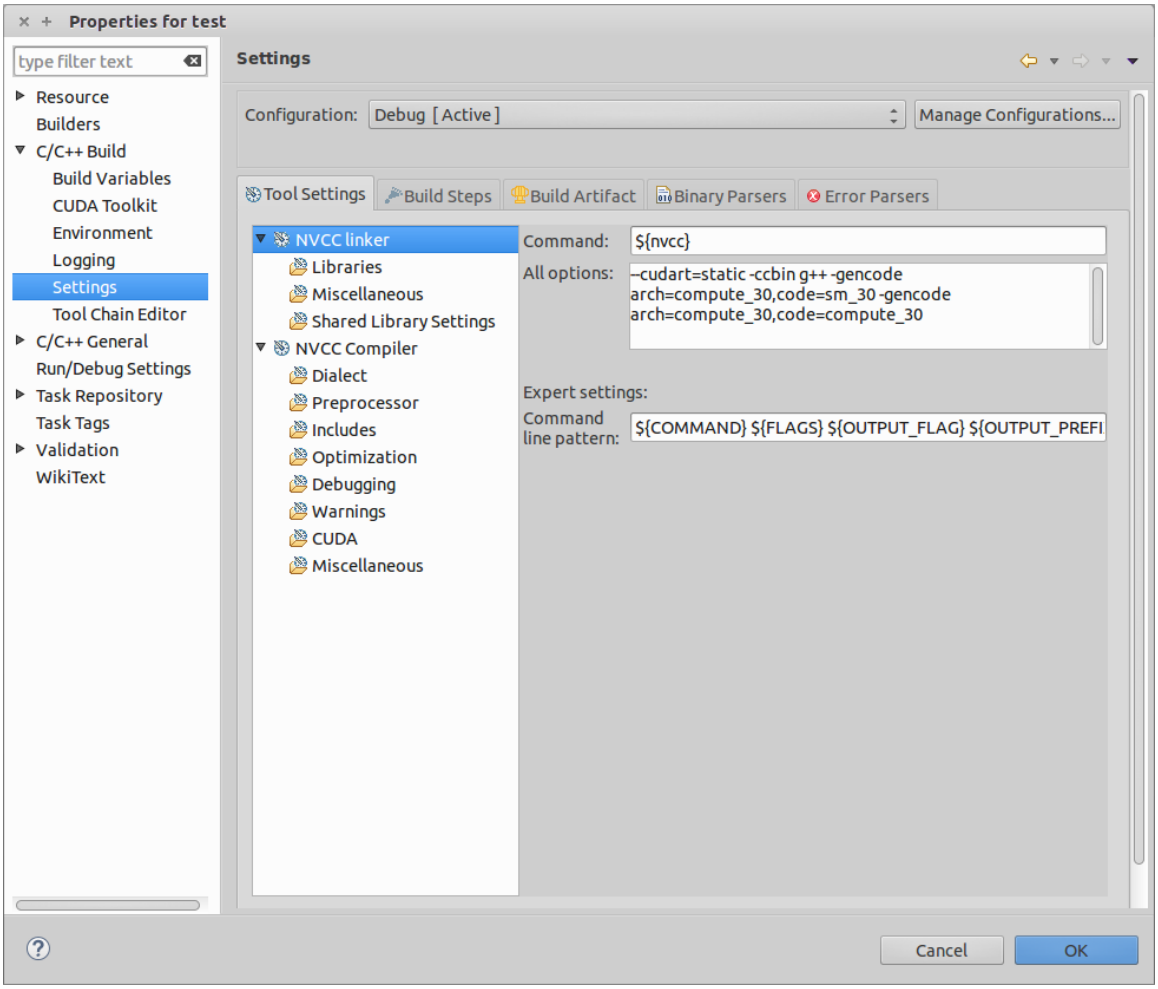
2.4.1. cuHook Sample

cuHook sample builds both the library and the executable. cuHook sample should be imported as the "makefile" project using the following steps.

1. From the main menu, open the new project wizard - **File** > **New...** > **C/C++ Project**
2. Select project type "Makefile project" and choose "Empty Project"
3. Specify the project name and project files location.
4. Complete the wizard.
   The project will be shown in the **Project Explorer** view.
5. Right click on the project - **Import...** > **General** > **File System**
6. On the next wizard page, select the location of cuHook sample(Samples/7_CUDALibraries/cuHook)
7. Select all the source files and makefile and Finish the wizard
8. Build the project by clicking on the hammer button on the main toolbar.
9. To run the sample, from the main menu - **Run** > **Run Configurations...** > Select the executable > Go to Environment tab > **New...** > enter Name=LD_PRELOAD, Value=./libcuhook.so.1 > **Run** will execute the sample

2.5. Configure Build Settings

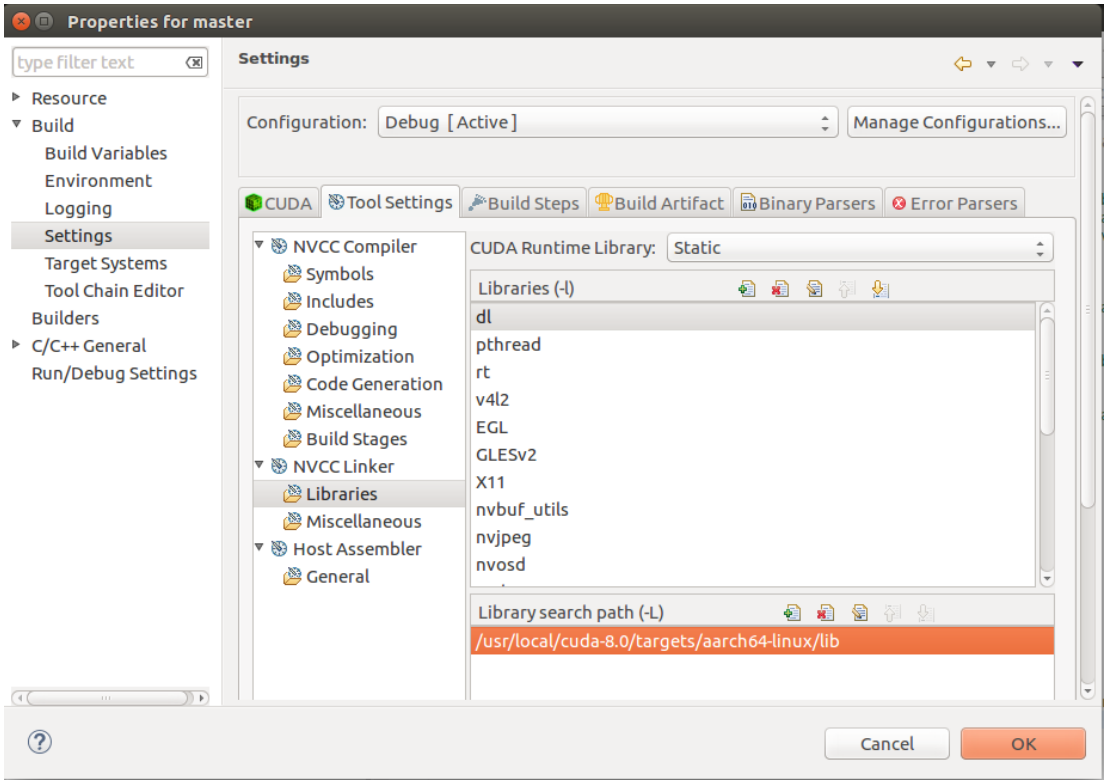
To define build settings: In the C/C++ Projects view, right-click your project, and select Properties. Select C/C++ Build, Settings from the list.



The following are the categories of Nvcc linker settings that can be configured for the selected project.

All options field in the main page is not editable and it's the collection of options set in the child categories.

- *Libraries* - Configure library search path(-L) and to include linker libraries(-l). When you are cross compiling for different target os, the library search path should point to the appropriate location where the target os libraries are present.



- *Miscellaneous* - Set additional linker options and option to link with OpenGL libraries.
- *Shared Library Settings* - Set option to build a shared library.

The following are the categories of Nvcc Compiler settings that can be configured for the selected project.

All options field in the main page is not editable and it's the collection of options set in the child categories.

- *Dialect* - Select the language standard and dialect options.
- *Preprocessor* - Add the defined and undefined symbols for the preprocessor.
- *Includes* - Set include paths and include files for the compiler.
- *Optimization* - Set the compiler optimization level.
- *Debugging* - Set the options to generate debug information.
- *Warnings* - Set inhibit all warning messages.
- *CUDA* - Generate code for different real architectures with the PTX for the same vitrual architectures.

2.6. Debugging CUDA Applications

Nsight must be running and at least one project must exist.

1. In the **Project Explorer** view, select project you want to debug. Make sure the project executable is compiled and no error markers are shown on the project.
2. Right click on the project and go to **Debug As > NVIDIA CUDA GDB Debugger** menu.
3. You will be offered to switch perspective when you run debugger for the first time. Click "Yes".

   Perspective is a window layout preset specifically designed for a particular task.
4. Application will suspend in the *main* function. At this point there is no GPU code running.
5. Add a breakpoint in the device code. Resume the application.

Debugger will break when application reaches the breakpoint. You can now explore your CUDA device state, step through your GPU code or resume the application.

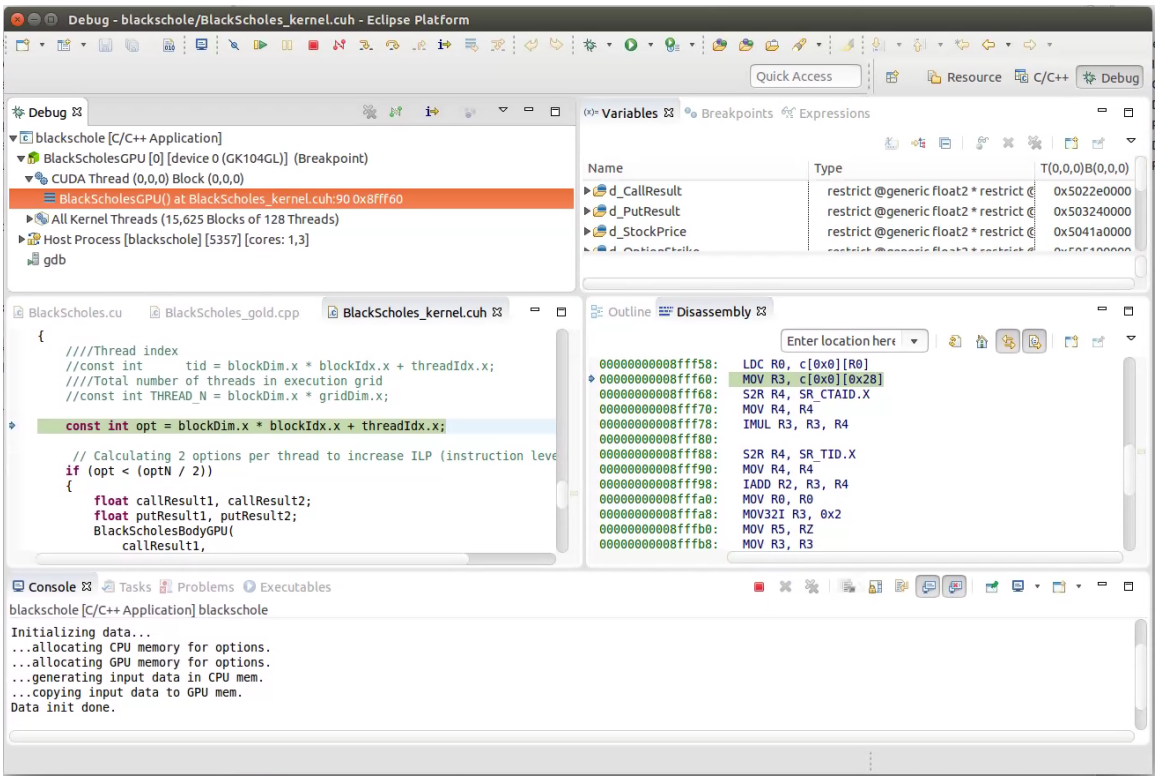


Figure 2 Debugging CUDA application

Additional debugger options can be set in the debug configuration dialog through Run > Debug Configurations .. menu..

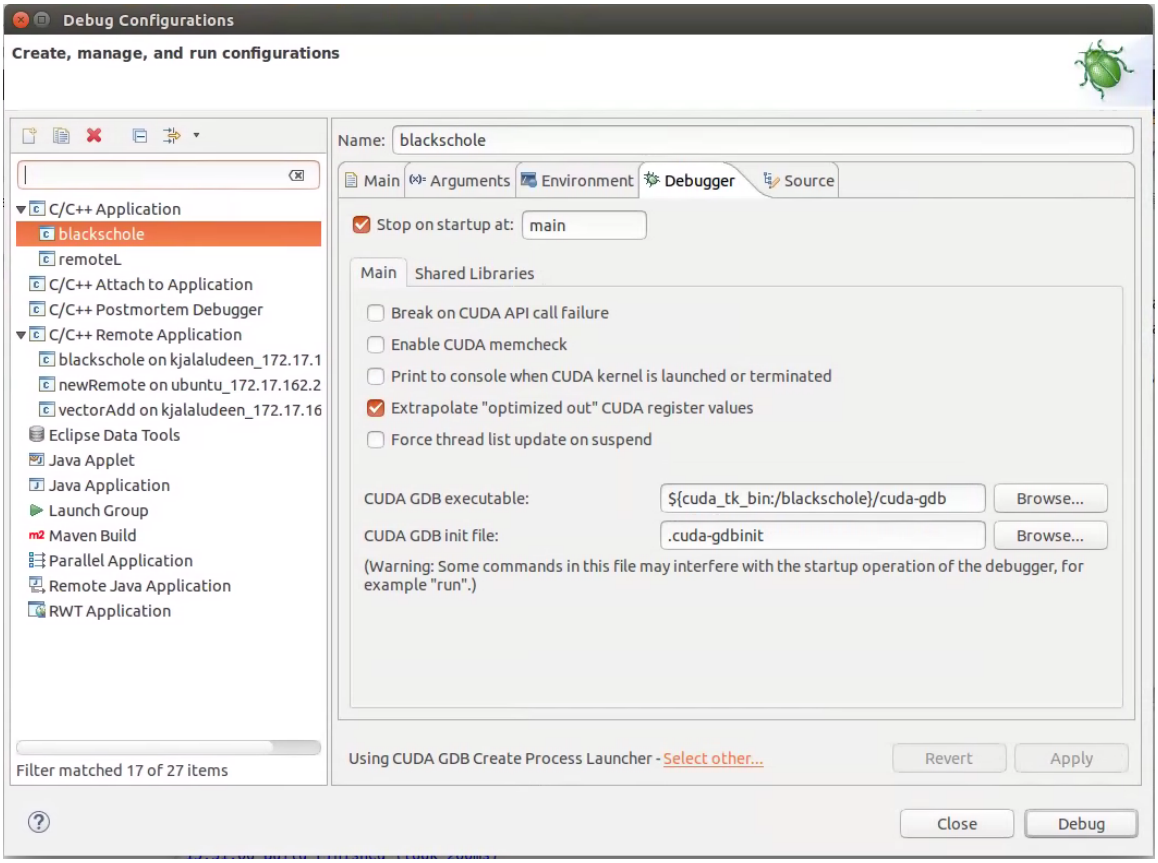
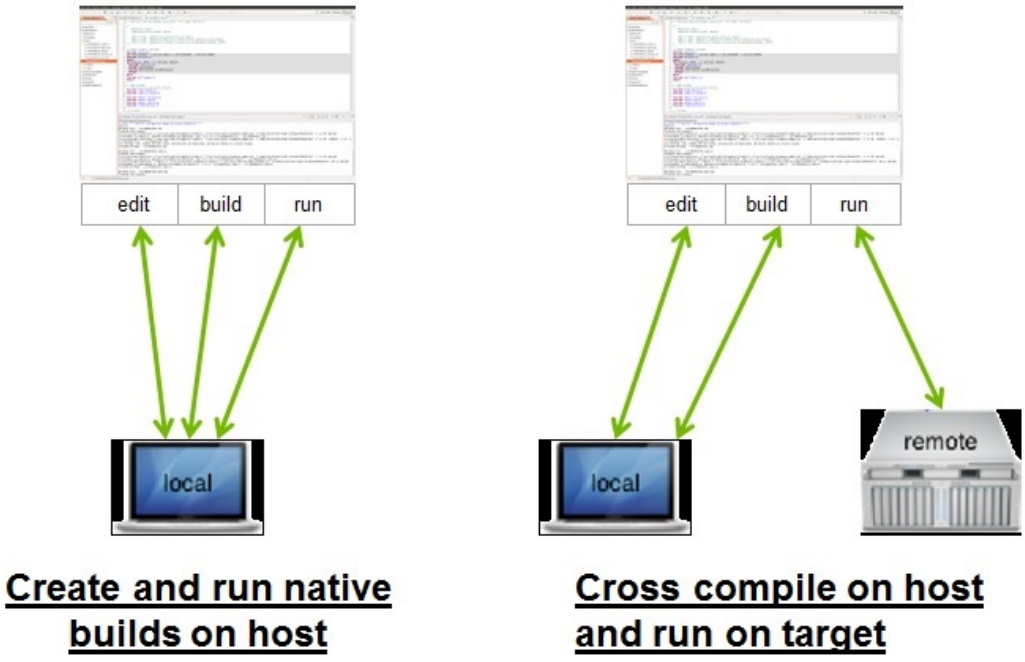


Figure 3 Debugging CUDA application

2.7. Remote development of CUDA Applications

Nsight Eclipse Edition also supports remote development of CUDA application starting with CUDA Toolkit 6.0. The picture below shows how Nsight Eclipse Edition can be used for local as well as remote development:

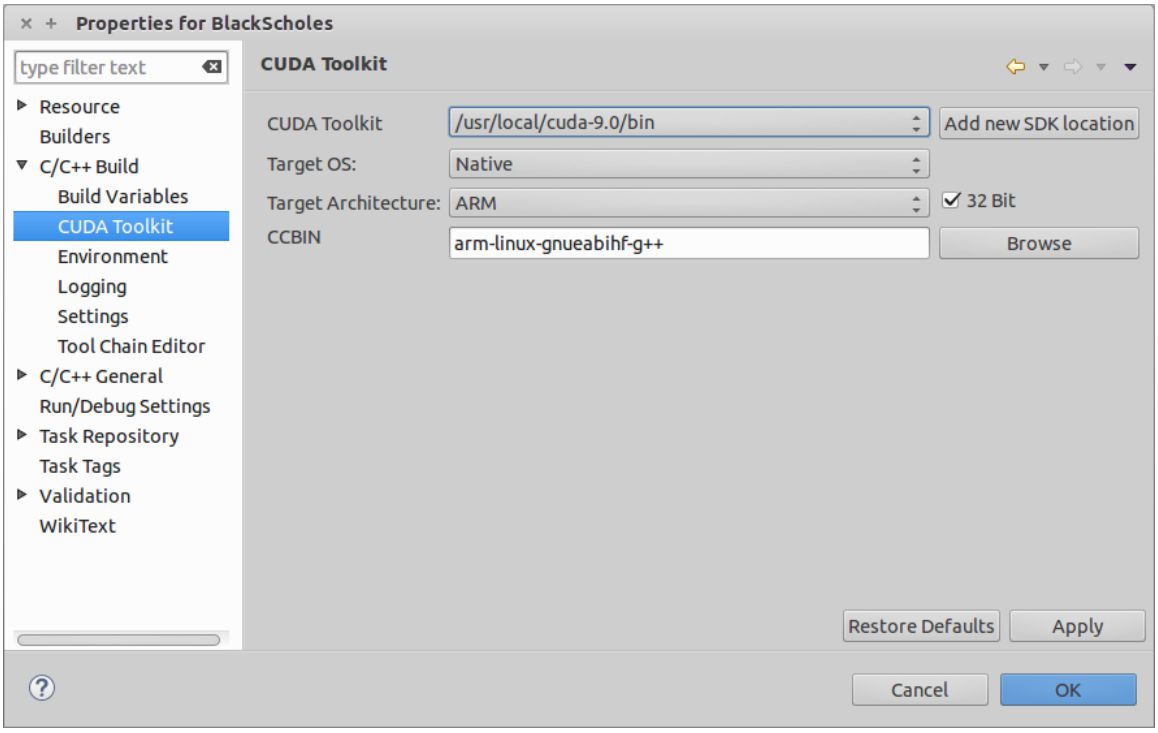


For remote development you do not need any NVIDIA GPU on your host system. The remote target system can be a Linux x86 or POWER system with an NVIDIA GPU or an Tegra-based ARM system. Nsight IDE and UI tools can only be hosted on x86 and POWER systems.

Nsight Eclipse Plugins supports the cross compilation mode for remote devices.

In the **cross compilation mode** the project resides on the host system and the cross compilation is also done on the host system. The cross compilation mode is only supported on an Ubuntu x86 host system.

To cross compile select the target cross compile architecture in CPU architecture drop down in the project properties page:

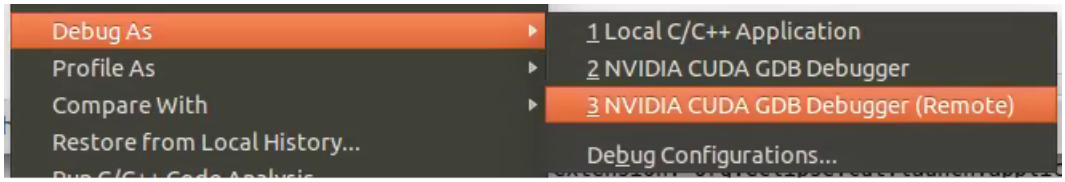


2.8. Debugging Remote CUDA Applications

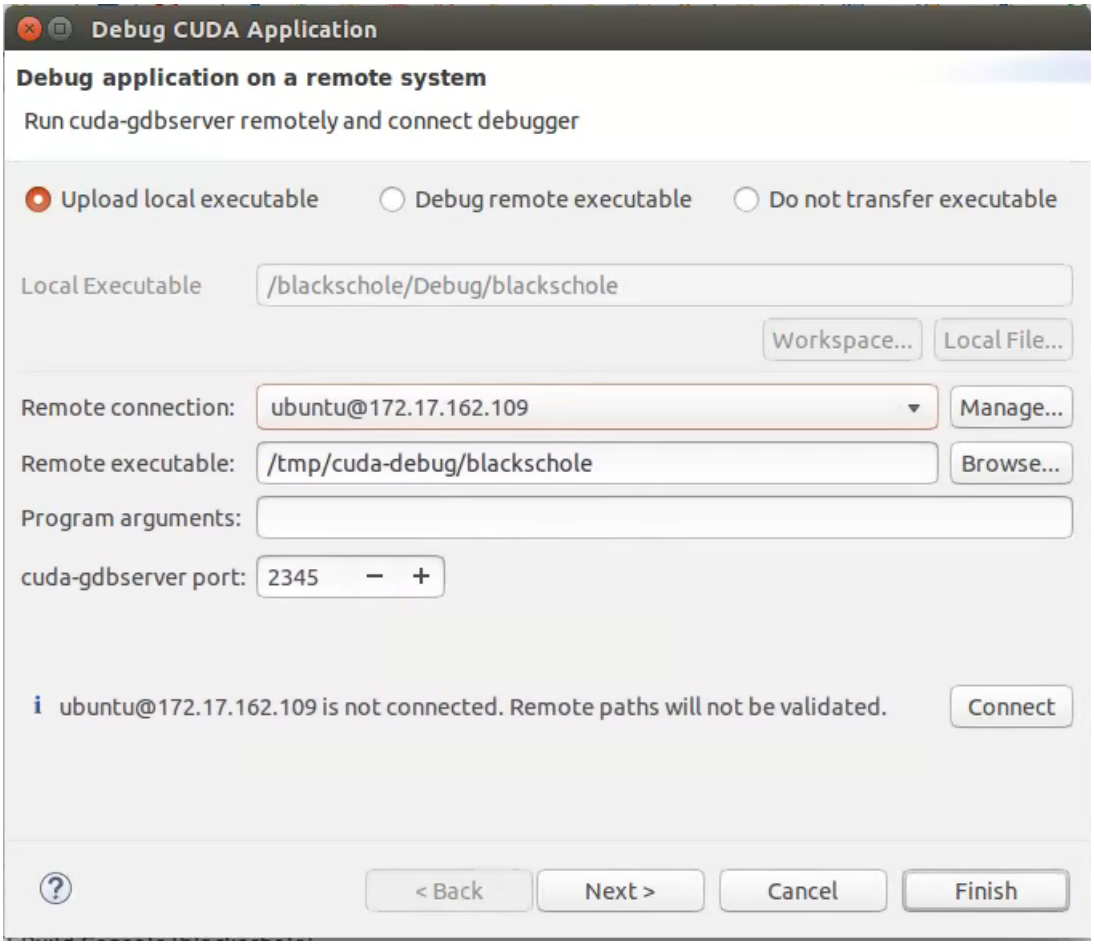
Remote debugging is available starting with CUDA Toolkit 5.5. A dedicated GPU is not required to use Nsight remote debugging UI. A dedicated GPU is still required on the debug target. Only Linux targets are supported. Debug host and target may run different operating systems or have different CPU architectures. The remote machine must be accessible via SSH and CUDA Toolkit must be installed on both machines.

If there is a firewall between the host and the target, it must be set up to let RSP messages through, or SSH port-forwarding must be used.

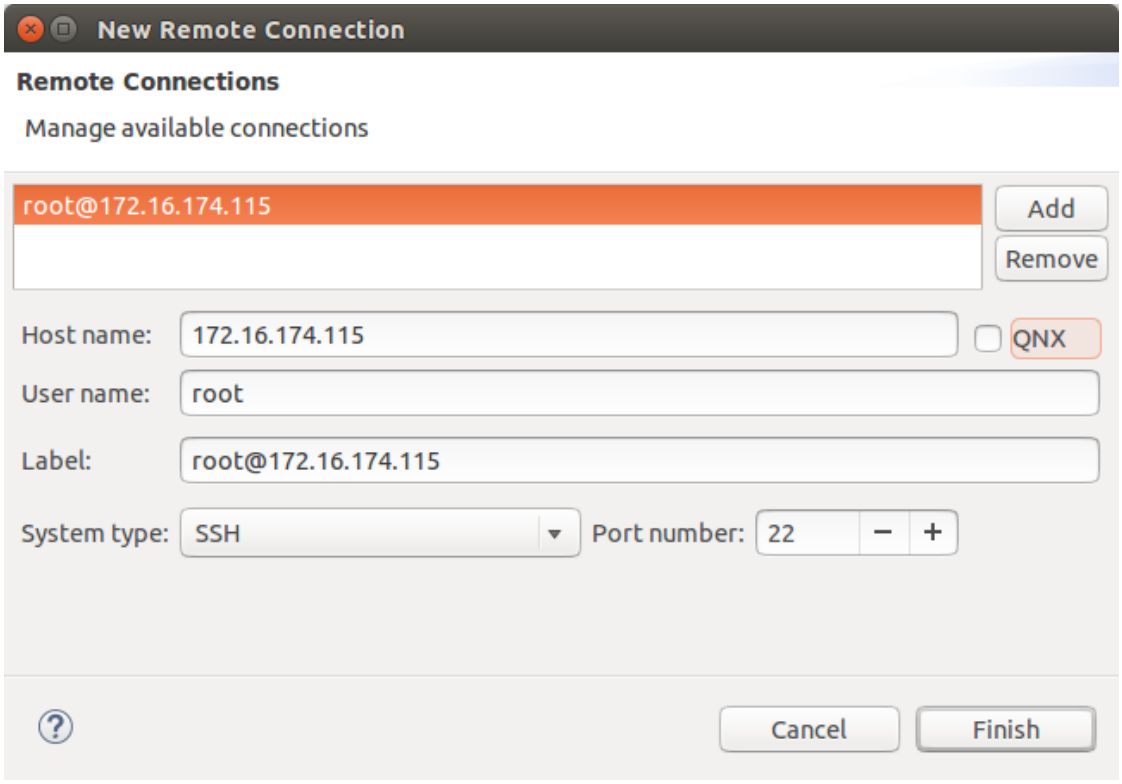
1. Select the project and right click then go to **Debug As...>NVIDIA CUDA GDB Debugger(Remote)** menu item.



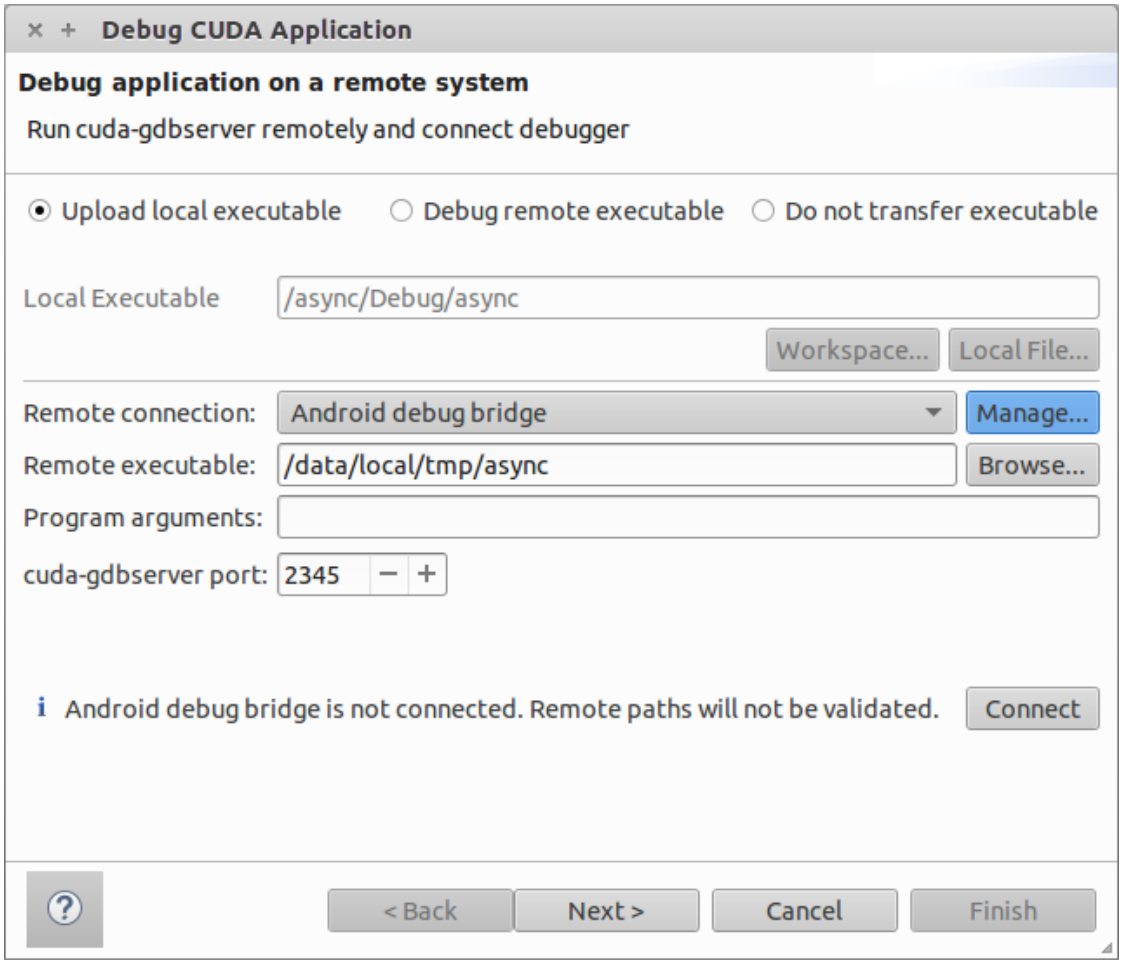
2. Type the full path to a local executable or select one using the **Local file...** button.



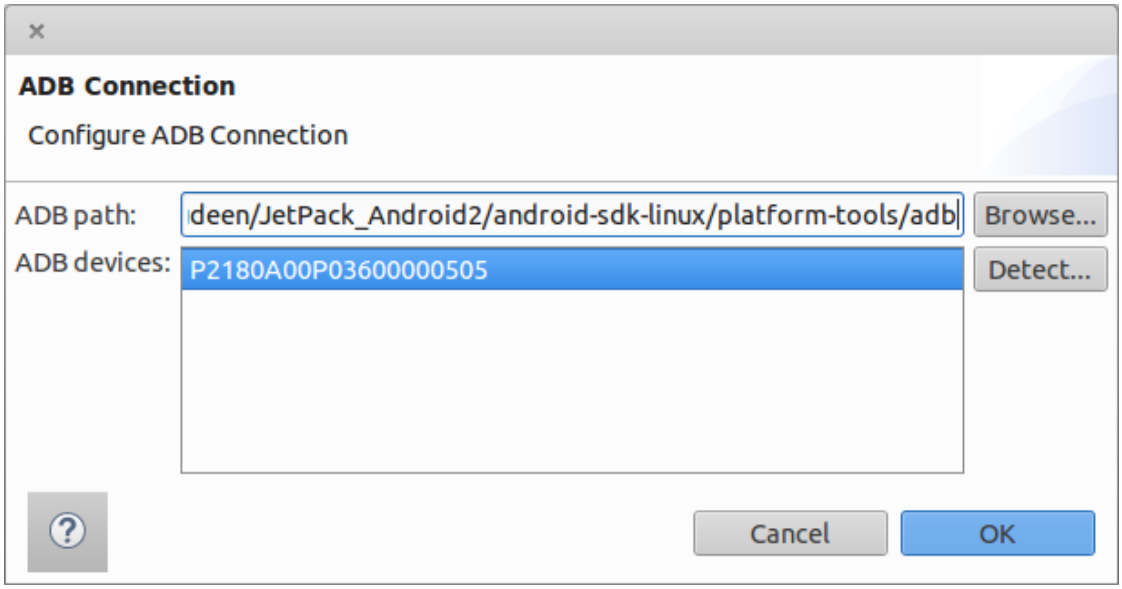
3. Select a remote connection from a drop-down list or press the **Add connection...** button to create a new one.
4. If you are creating a new remote connection, enter the host name(or IP address) as well as the user name. Select the SSH as system type. Also select the QNX check box for QNX targets and then press **Finish**.



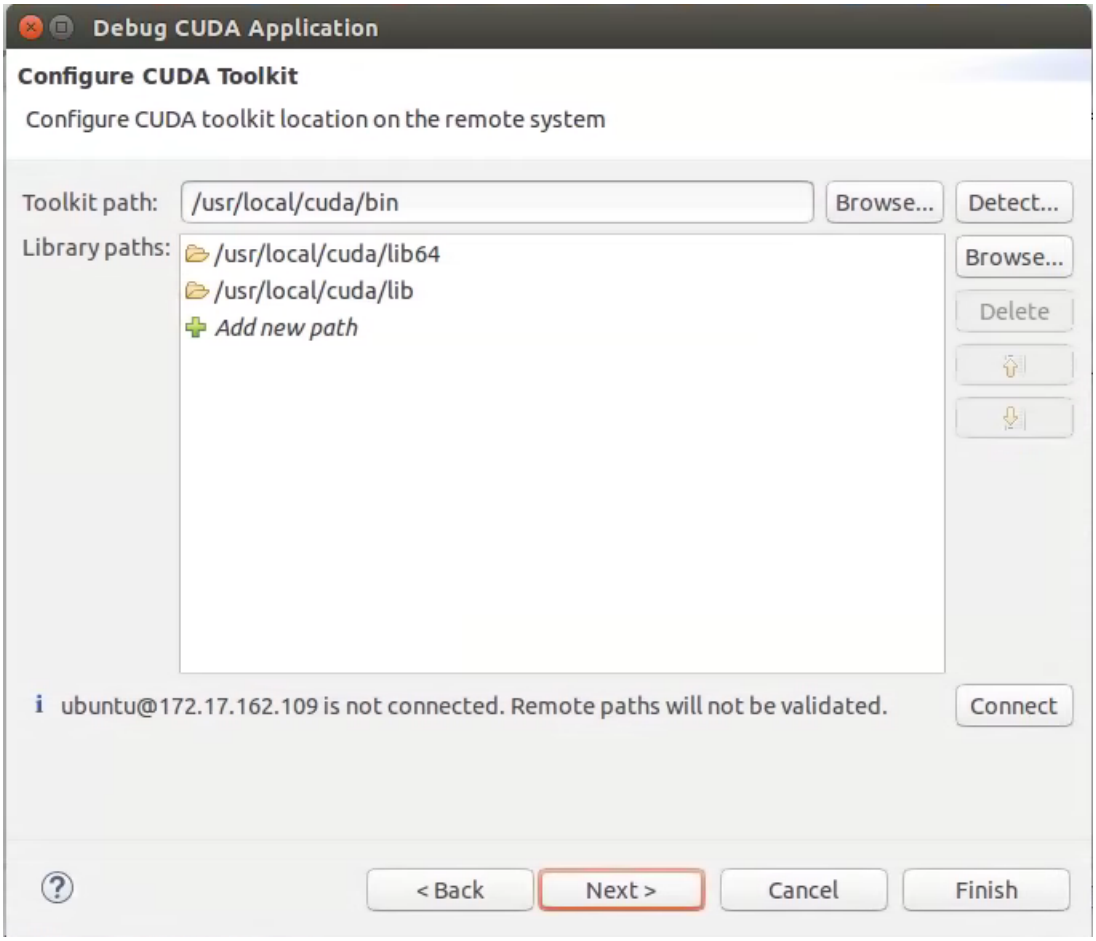
5. **For Android devices:** To configure the remote connection using Android debug bridge, select the **Android debug bridge** from the Remote Connection drop-down list, Android device must be connected to the host system using USB port.



Press **Manage** button, and enter or select the path to adb utility. You need to install Android SDK platform tools to use Android debug bridge. press **Detect** button to find the android device available through ADB.



6. Optional: Press **Connect** to verify the selected remote connection.
7. Press the **Next** button.
8. Type the full path to cuda-gdbserver on the remote system or select one using the **Browse...** button.



9. Click on "Add new path" or on the **Browse...** button to specify the path to the shared libraries the remote application depends on.
10. Click on the **Finish** button to finish the new debug configuration wizard and start debugging the application.
11. You will be offered to switch perspective when you run the debugger for the first time. Click **Yes**.

    Perspective is a window layout preset specifically designed for a particular task.

The debugger will stop at the application main routine. You can now set breakpoints, or resume the application.

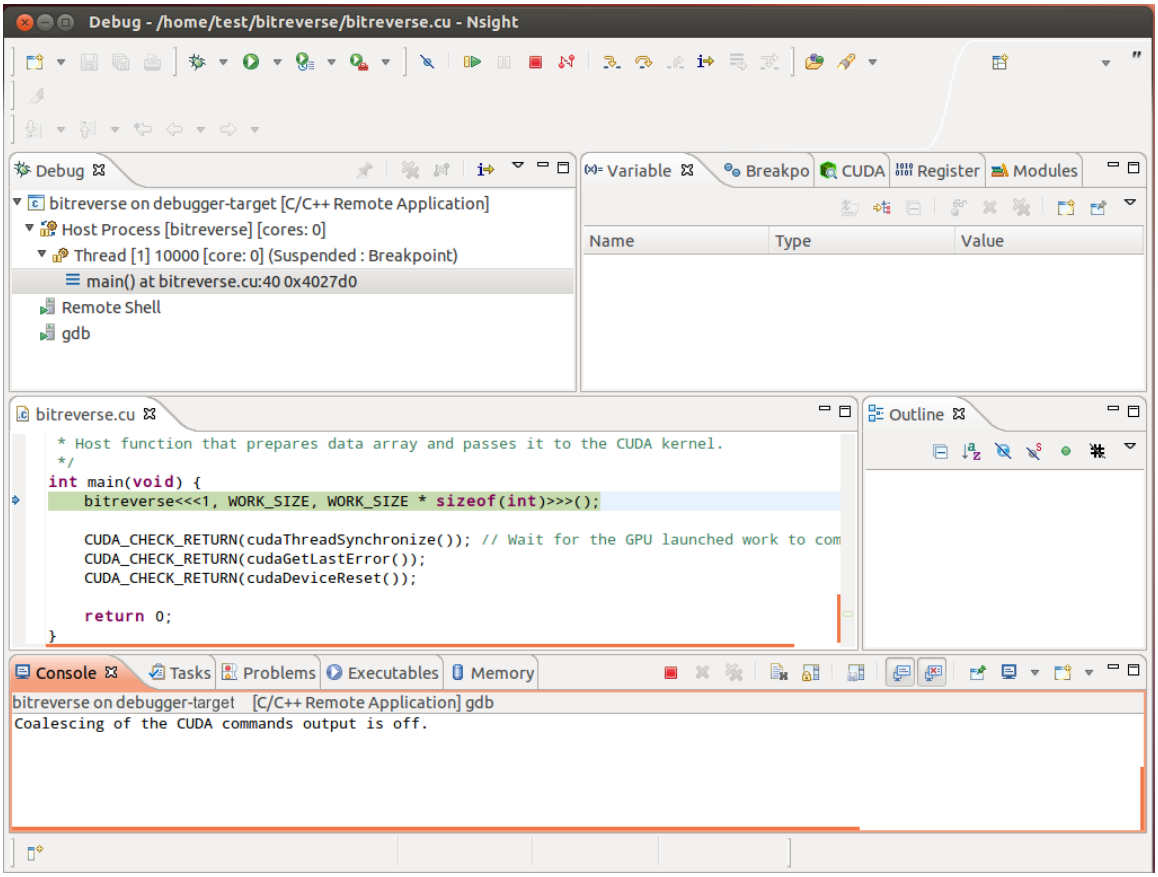


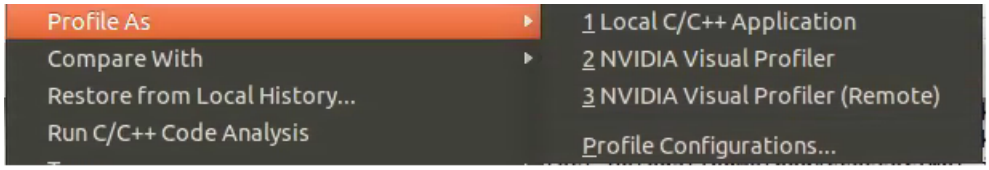
Figure 4 Debugging remote CUDA application

2.9. Profiling CUDA applications

Nsight must be running and at least one project must exist. Profiler cannot be used when debugging session is in progress.

Nsight Eclipse Edition profiling features are based on the NVIDIA Visual Profiler (*nvvp*) code. Nsight Eclipse Plugins Edition will launch the Visual Profiler as an external tool with the executable and other information from the selected project.

1. In the **Project Explorer** view, select project you want to profile. Make sure the project executable is compiled and no error markers are shown on the project.
2. Select the project and right click and go to **Profile As>NVIDIA Visual Profiler** menu.



Nsight Eclipse will launch the Visual Profiler to specify extra profiler options with the executable information already passed from the selected project.

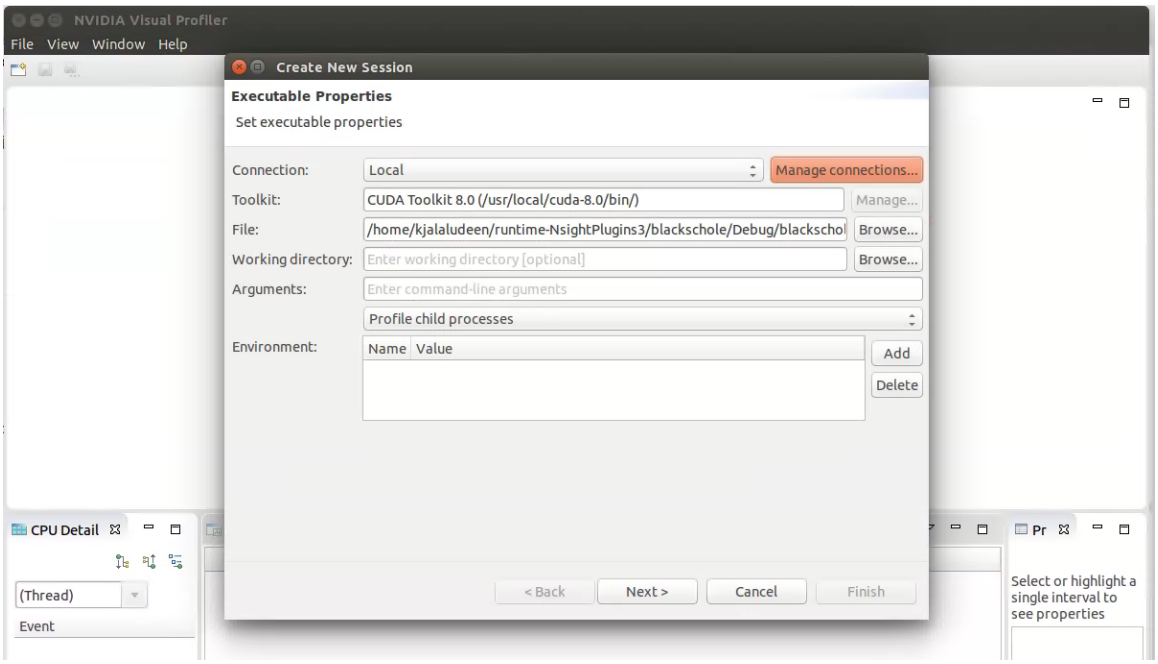
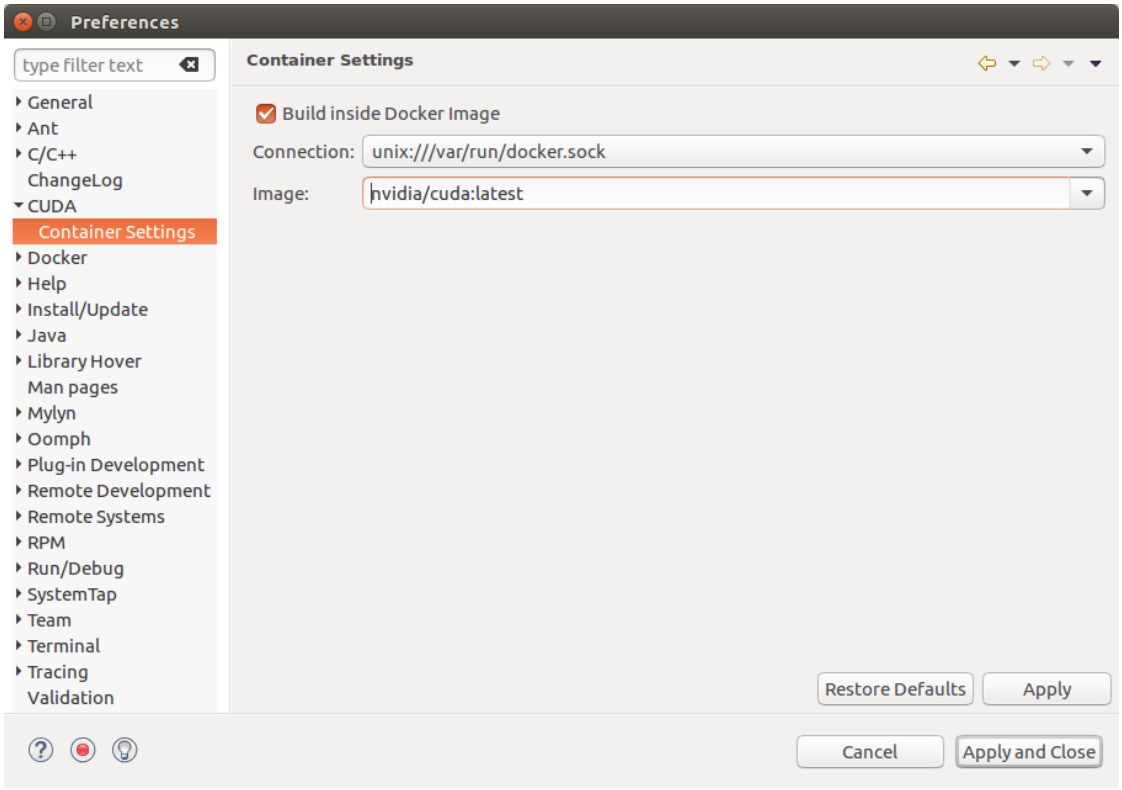


Figure 5 Profiling CUDA Application

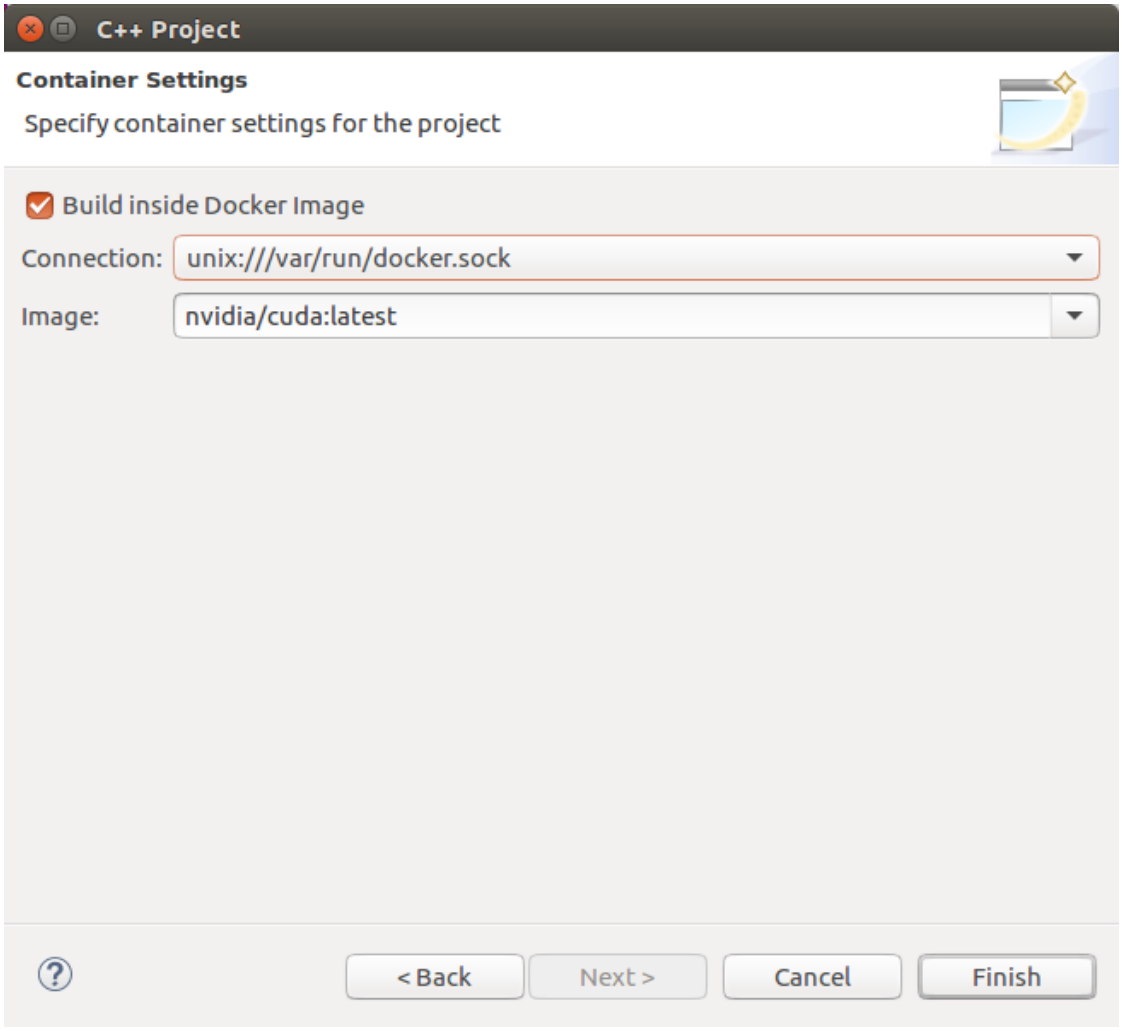
2.10. Build Projects in a Docker Container

You can build and debug C/C++ and CUDA projects in a Docker container using Nsight Eclipse Edition. To get started, you need to first pull and install the Docker image that encapsulates the CUDA toolkit and cross platform tool chains. You can get the Docker images from NVIDIA GPU Cloud.Then you can use Nsight Eclipse Edition to build CUDA projects in a Docker container.

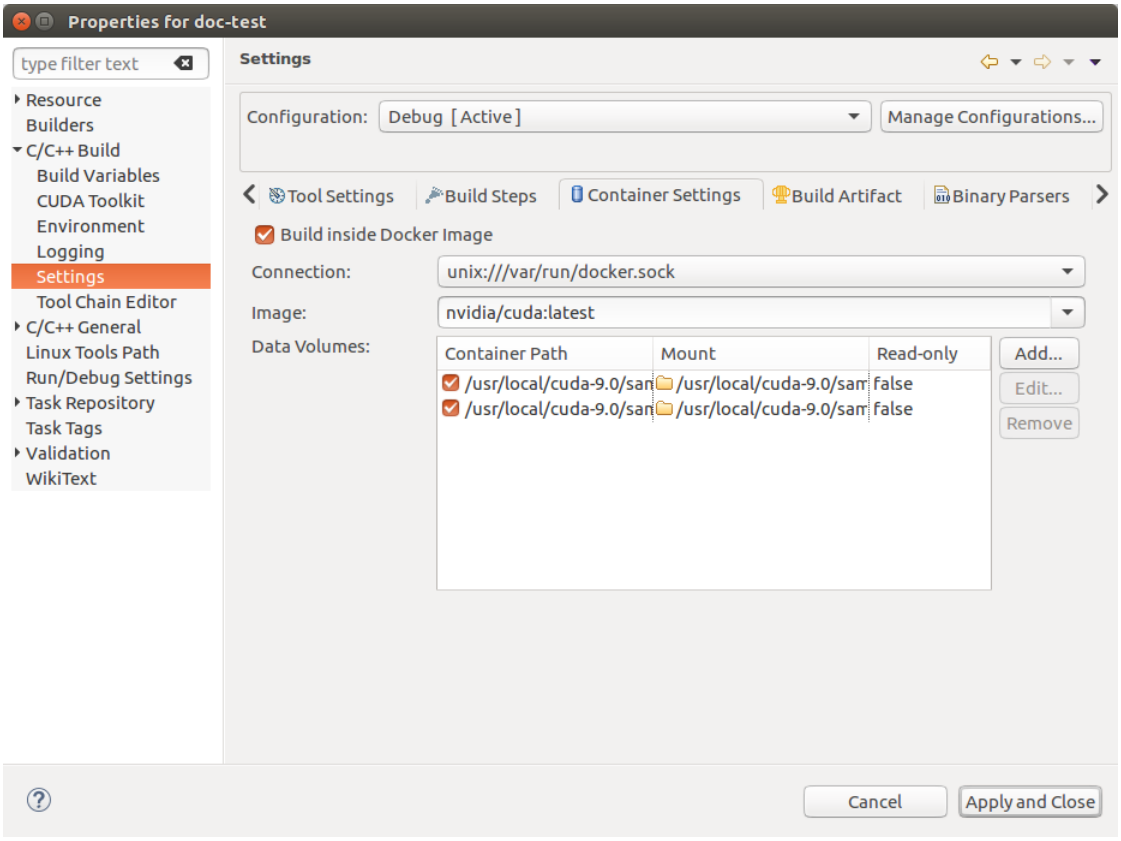
1. Open Nsight Eclipse Edition and configure the container settings.
2. Open the Preferences page, Window > Preferences and go to: **CUDA > Container Settings**



3. Select the option if you want to build the projects inside the Docker container. Make sure the CUDA toolkit path that is specified in the CUDA preferences is the path of the CUDA toolkit inside a Docker container.
4. Select the Connection and the Image dropdown will display all the Docker images that are currently installed. Choose the docker image that you want to use to build/debug the projects. The preferences that are set here will be automatically displayed in the project setup wizard. You can choose to modify the container settings for the individual projects from the project setup wizard.
5. To create a project, From the main menu, open the new project wizard: File > New... > CUDA C/C++ Project
6. Specify the project name and project files location. And select the CUDA toolchain from the list of toolchains.
7. In the last page of project setup wizard, the container options will be displayed. The default container settings from the preference page will be displayed here. You can choose to modify the settings for this project in this Container settings page.



8. Complete the project setup wizard. The project will be created and shown in the Project Explorer view.
9. The project source directories will be automatically mounted to the docker container.
10. If you need to mount any other directories that contains the include files/libraries and etc to the docker container, you can mount those directories from the project property page.
11. Right click on the project and go to Properties. Select C/C++ Build > Settings > Container Settigns Tab. Additional directories can be mounted from this property page.

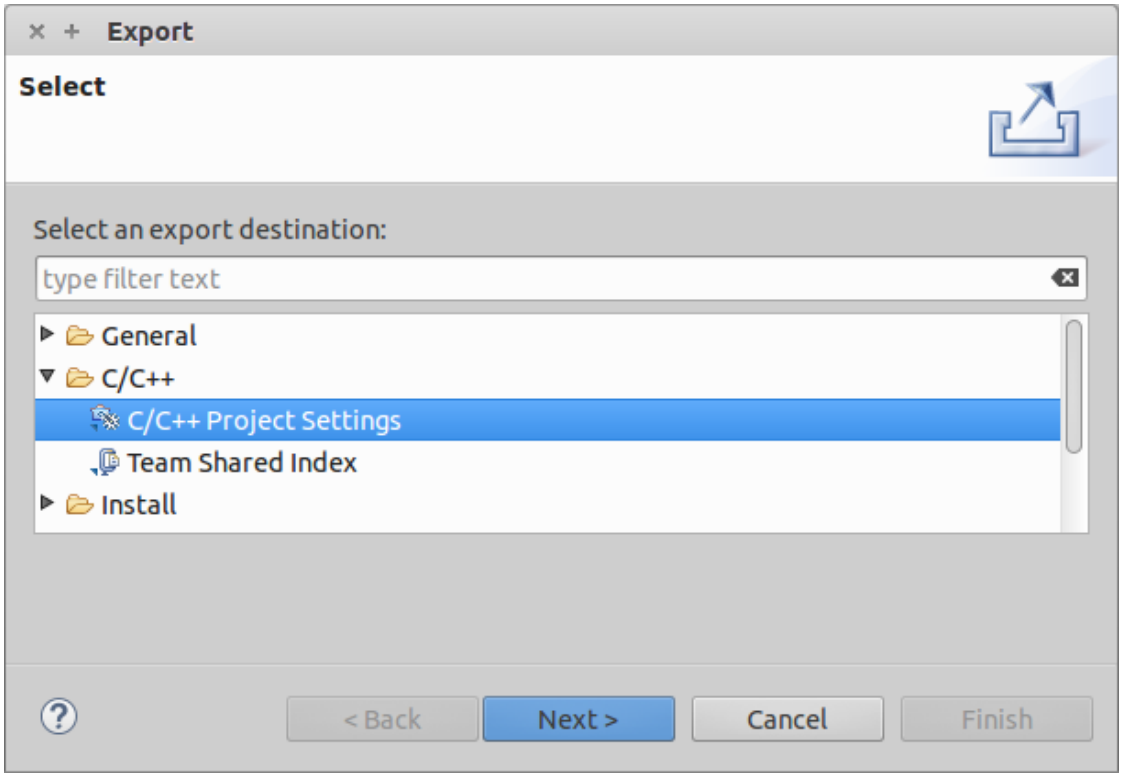


12 Build the project by hitting the hammer button on the main toolbar.
The project is now built in the chosen Docker container the executable will be available on the host.

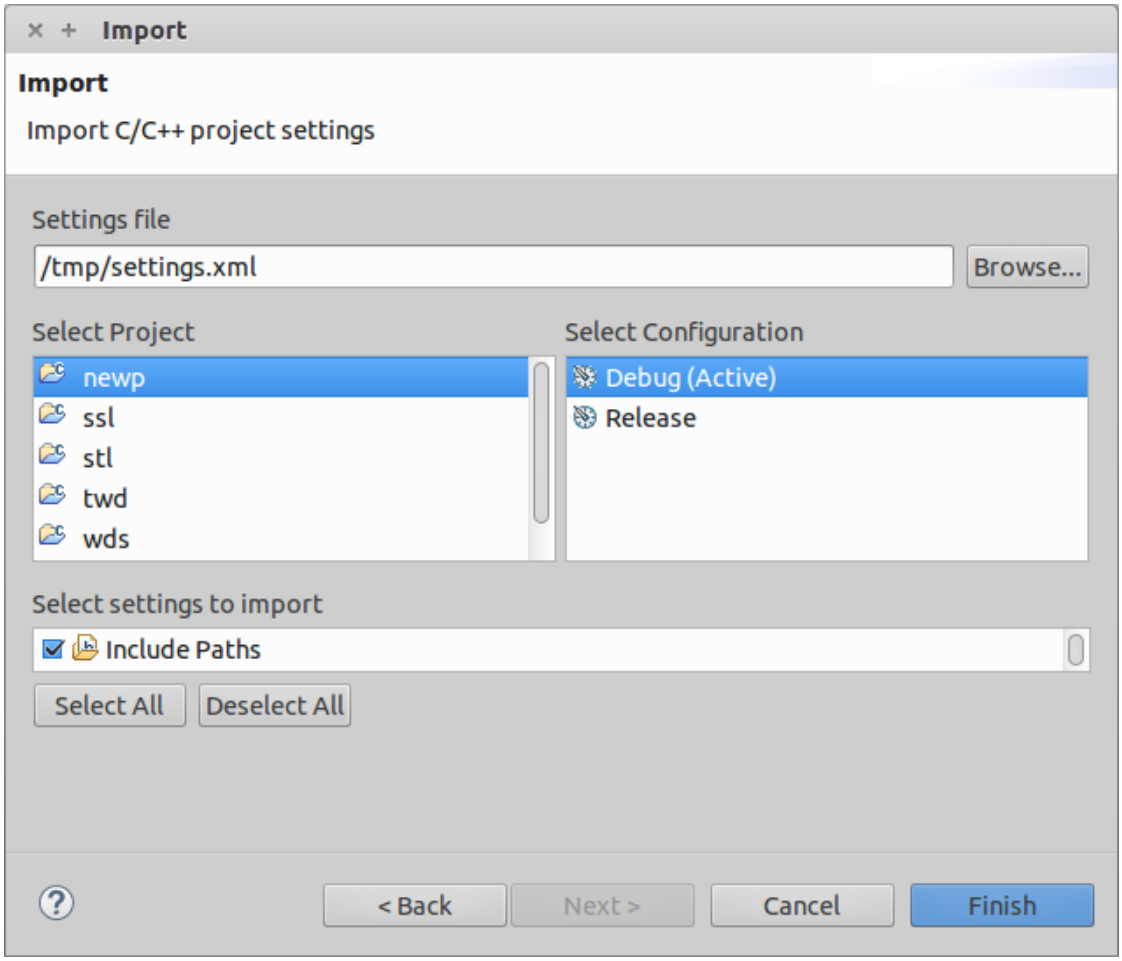
2.11. Importing Nsight Eclipse Projects

The projects that are created with Nsight Eclipse Edition can be imported into the Eclipse workbench with Nsight Eclipse plugins.

1. Open Nsight Eclipse edition and select the project that needs to be exported.
2. Right click on the Nsight Eclipse project and go to - **Export** > **C/C++** > **C/C++ Project Settings** > **Next** menu.



3. Select the project and settings to export.
4. Specify the "Export to file" location.
5. Settings will be stored in the given XML file.
6. Go to Eclipse workbench where the project settings needs to be imported.
7. Create a C/C++ Project from the main menu **File** > **New** > **C/C++ Project**
8. Specify the project name and choose Empty project type with CUDA toolchains.
9. Right click on the project to import the source files. **Import** > **General** > **File System** >(From directory) or copy the source files from the existing project.
10. Import the project settings like include paths and symbols using the following right click menu **Import** > **C/C++** > **C/C++ Project Settings** >Next...
11. Select the location of the project settigns file and select the project and configuration on the next wizard page.



12. Complete the wizard.
    The project settings will be imported from the file exported from Nsight Eclipse Edition.
13. Build the project by clicking on the hammer button on the main toolbar.

2.12. More Information

More information about the Eclipse CDT features and other topics is available in the Help contents. To access Help contents select *Help->Help Contents* from the Nsight main menu.

More information about CUDA, CUDA Toolkit and other tools is available on CUDA web page at http://developer.nvidia.com/cuda