



# NSIGHT COMPUTE

v2023.3.1 | October 2023

**User Manual**



# TABLE OF CONTENTS

<b>Chapter 1. Introduction.....</b>	<b>1</b>
1.1. Overview.....	1
<b>Chapter 2. Quickstart.....</b>	<b>2</b>
2.1. Interactive Profile Activity.....	3
2.2. Non-Interactive Profile Activity.....	7
2.3. System Trace Activity.....	9
2.4. Navigate the Report.....	12
<b>Chapter 3. Connection Dialog.....</b>	<b>15</b>
3.1. Remote Connections.....	17
3.2. Interactive Profile Activity.....	21
3.3. Profile Activity.....	22
3.4. Reset.....	23
<b>Chapter 4. Main Menu and Toolbar.....</b>	<b>24</b>
4.1. Main Menu.....	24
4.2. Main Toolbar.....	27
4.3. Status Banners.....	27
<b>Chapter 5. Tool Windows.....</b>	<b>28</b>
5.1. API Statistics.....	28
5.2. API Stream.....	29
5.3. Baselines.....	30
5.4. Metric Details.....	31
5.5. NVTX.....	32
5.6. Resources.....	33
5.6.1. Memory Allocations.....	33
5.6.2. Graphviz DOT and SVG exports.....	34
5.7. Metric Selection.....	34
<b>Chapter 6. Profiler Report.....</b>	<b>37</b>
6.1. Header.....	37
6.2. Report Pages.....	38
6.2.1. Session Page.....	39
6.2.2. Summary Page.....	39
6.2.3. Details Page.....	41
6.2.4. Source Page.....	45
6.2.4.1. Navigation.....	46
6.2.4.2. Metrics.....	47
6.2.4.3. Profiles.....	55
6.2.4.4. Limitations.....	56
6.2.5. Comments Page.....	57
6.2.6. Call Stack / NVTX Page.....	57
6.2.7. Raw Page.....	57

6.3. Metrics and Units.....	58
<b>Chapter 7. Baselines.....</b>	<b>59</b>
<b>Chapter 8. Standalone Source Viewer.....</b>	<b>61</b>
<b>Chapter 9. Source Comparison.....</b>	<b>63</b>
<b>Chapter 10. Occupancy Calculator.....</b>	<b>65</b>
10.1. Tables.....	66
10.2. Graphs.....	67
10.3. GPU Data.....	68
<b>Chapter 11. Acceleration Structure Viewer.....</b>	<b>69</b>
11.1. Navigation.....	71
11.2. Filtering and Highlighting.....	72
11.3. Rendering Options.....	74
11.4. Exporting.....	74
<b>Chapter 12. Options.....</b>	<b>76</b>
12.1. Profile.....	77
12.2. Environment.....	79
12.3. Connection.....	79
Target Connection Properties.....	80
Host Connection Properties.....	80
12.4. Source Lookup.....	80
12.5. Send Feedback.....	81
<b>Chapter 13. Projects.....</b>	<b>82</b>
13.1. Project Dialogs.....	82
13.2. Project Explorer.....	83
<b>Chapter 14. Visual Profiler Transition Guide.....</b>	<b>84</b>
14.1. Trace.....	84
14.2. Sessions.....	84
14.3. Timeline.....	85
14.4. Analysis.....	85
14.5. Command Line Arguments.....	86
<b>Chapter 15. Visual Studio Integration Guide.....</b>	<b>87</b>
15.1. Visual Studio Integration Overview.....	87

## LIST OF TABLES

Table 1 Remote File Type Support .....	24
Table 2 OptiX Traversable Graph Node Types .....	34
Table 3 Roofline Chart Zoom and Pan Controls .....	44
Table 4 Acceleration Structure Hierarchical Columns .....	70
Table 5 Acceleration Structure Analysis Tools .....	70
Table 6 Acceleration Structure Key Bindings .....	71
Table 7 NVIDIA Nsight Compute Profile Options .....	77
Table 8 NVIDIA Nsight Compute Environment Options .....	79
Table 9 NVIDIA Nsight Compute Target Connection Properties .....	80
Table 10 NVIDIA Nsight Compute Host Connection Options .....	80
Table 11 NVIDIA Nsight Compute Source Lookup Options .....	80
Table 12 NVIDIA Nsight Compute Send Feedback Options .....	81

# Chapter 1. INTRODUCTION

For users migrating from Visual Profiler to NVIDIA Nsight Compute, please see the [Visual Profiler Transition Guide](#) for comparison of features and workflows.

## 1.1. Overview

This document is a user guide to the next-generation NVIDIA Nsight Compute profiling tools. NVIDIA Nsight Compute is an interactive kernel profiler for CUDA applications. It provides detailed performance metrics and API debugging via a user interface and command line tool. In addition, its baseline feature allows users to compare results within the tool. NVIDIA Nsight Compute provides a customizable and data-driven user interface and metric collection and can be extended with analysis scripts for post-processing results.

### Important Features

- ▶ Interactive kernel profiler and API debugger
- ▶ Graphical profile report
- ▶ Result comparison across one or multiple reports within the tool
- ▶ Fast Data Collection
- ▶ UI and Command Line interface
- ▶ Fully customizable reports and analysis rules

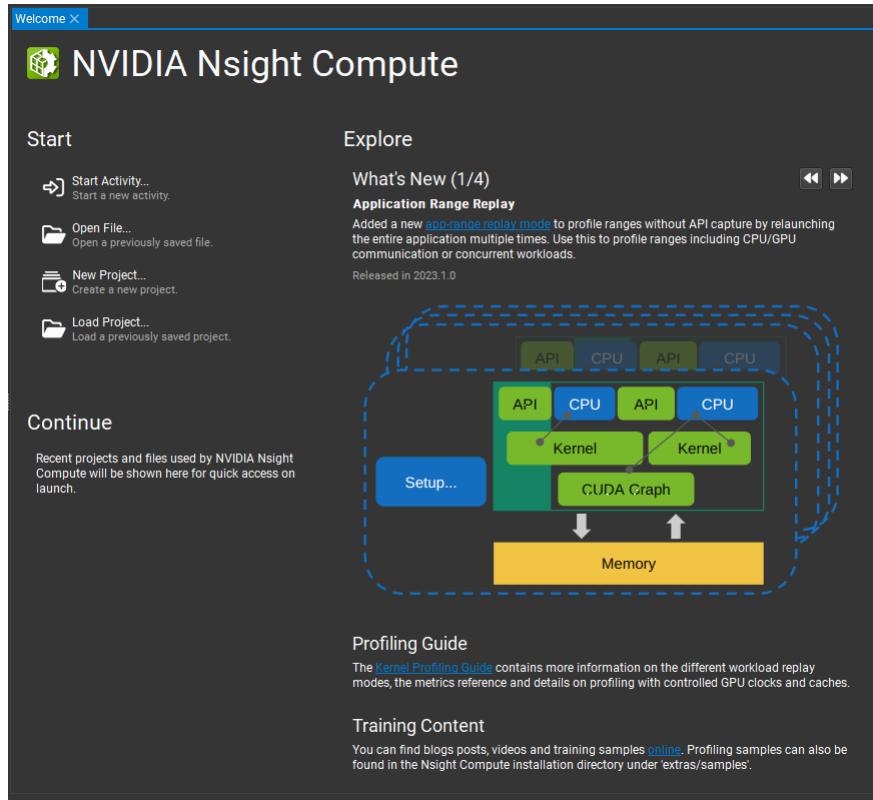
# Chapter 2.

# QUICKSTART

The following sections provide brief step-by-step guides of how to setup and run NVIDIA Nsight Compute to collect profile information. All directories are relative to the base directory of NVIDIA Nsight Compute, unless specified otherwise.

The UI executable is called ncu-ui. A shortcut with this name is located in the base directory of the NVIDIA Nsight Compute installation. The actual executable is located in the folder `host\windows-desktop-win7-x64` on Windows or `host\linux-desktop-glibc_2_11_3-x64` on Linux. By default, when installing from a Linux `.run` file, NVIDIA Nsight Compute is located in `/usr/local/cuda-<cuda-version>/nsight-compute-<version>`. When installing from a `.deb` or `.rpm` package, it is located in `/opt/nvidia/nsight-compute/<version>` to be consistent with [Nsight Systems](#). In Windows, the default path is `C:\Program Files\NVIDIA Corporation\Nsight Compute <version>`.

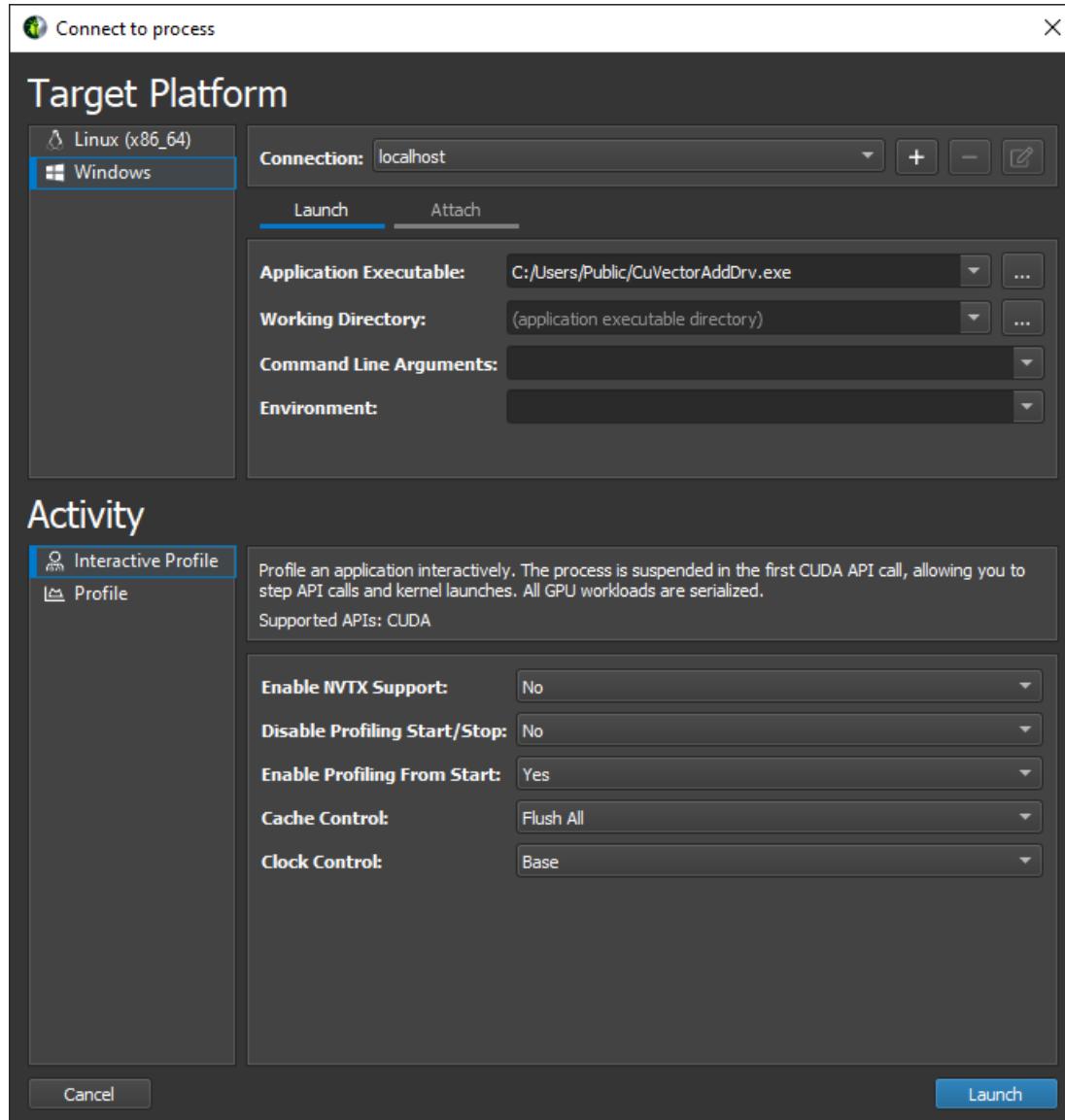
After starting NVIDIA Nsight Compute, by default the *Welcome Page* is opened. The *Start* section allows the user to start a new activity, open an existing report, create a new project or load an existing project. The *Continue* section provides links to recently opened reports and projects. The *Explore* section provides information about what is new in the latest release, as well as links to additional training. See [Environment](#) on how to change the start-up action.



## 2.1. Interactive Profile Activity

### 1. Launch the target application from NVIDIA Nsight Compute

When starting NVIDIA Nsight Compute, the *Welcome Page* will appear. Click on *Quick Launch* to open the *Connection* dialog. If the *Connection* dialog doesn't appear, you can open it using the *Connect* button from the main toolbar, as long as you are not currently connected. Select your target platform on the left-hand side and your connection target (machine) from the *Connection* drop down. If you have your local target platform selected, **localhost** will become available as a connection. Use the + button to add a new connection target. Then, continue by filling in the details in the *Launch* tab. In the *Activity* panel, select the Interactive Profile activity to initiate a session that allows controlling the execution of the target application and selecting the kernels of interest interactively. Press *Launch* to start the session.

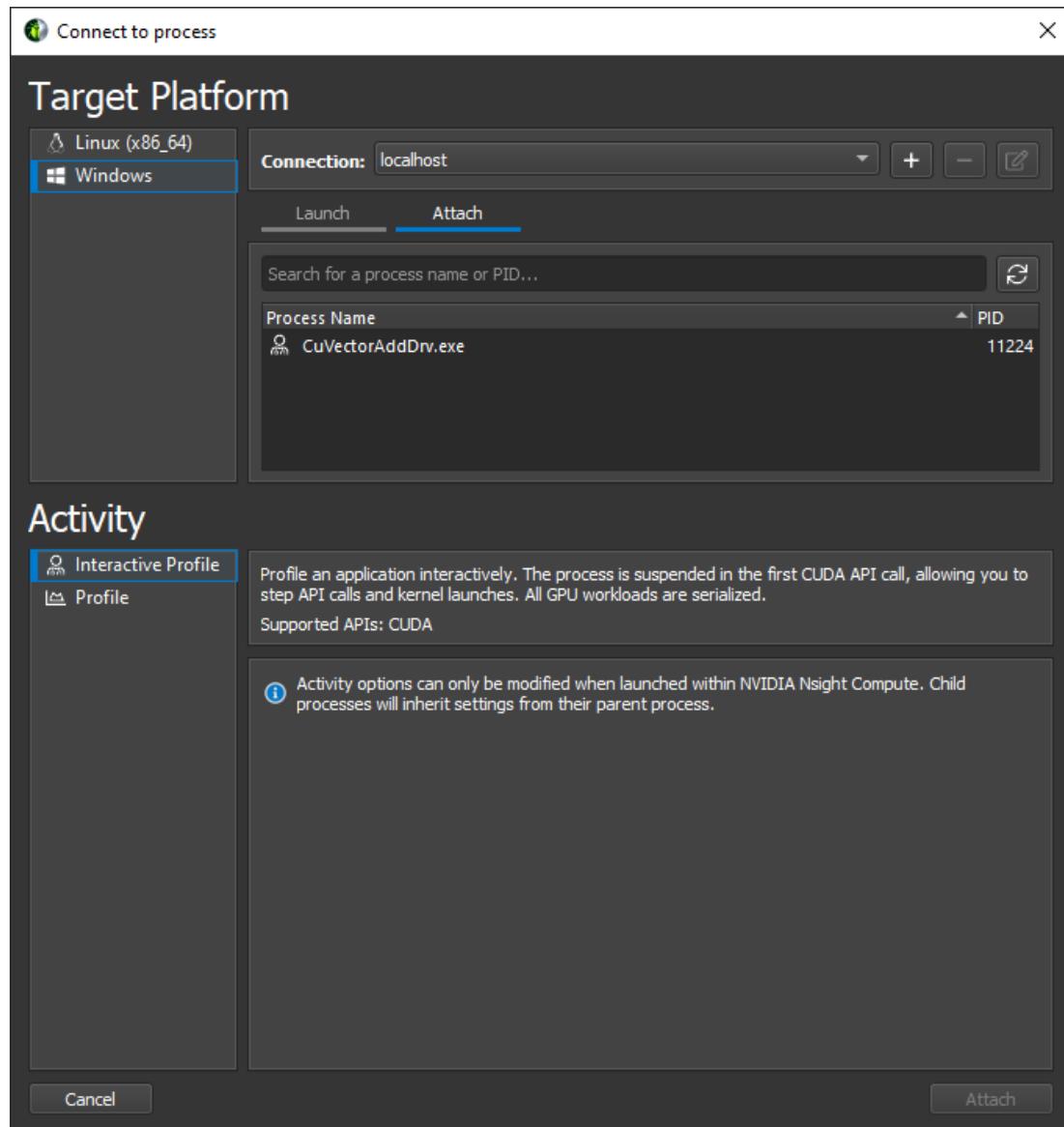


## 2. Launch the target application with tools instrumentation from the command line

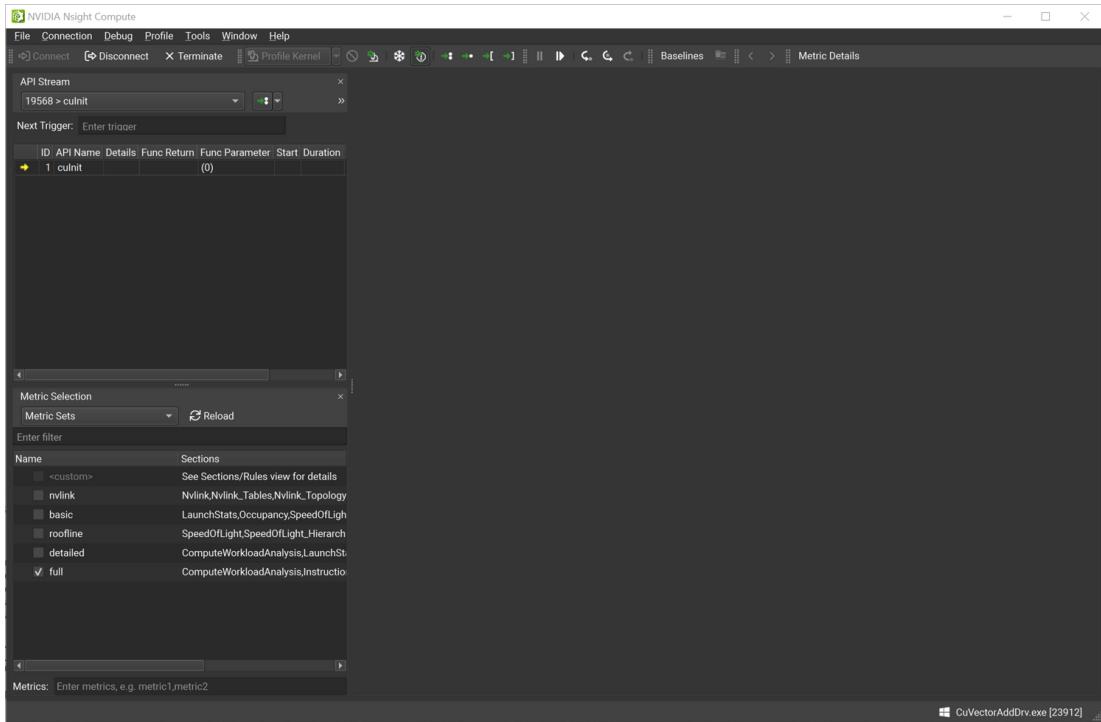
The ncu can act as a simple wrapper that forces the target application to load the necessary libraries for tools instrumentation. The parameter `--mode=launch` specifies that the target application should be launched and suspended before the first instrumented API call. That way the application waits until we connect with the UI.

```
$ ncu --mode=launch CuVectorAddDrv.exe
```

## 3. Launch NVIDIA Nsight Compute and connect to target application



Select the target machine at the top of the dialog to connect and update the list of attachable applications. By default, *localhost* is pre-selected if the target matches your current local platform. Select the *Attach* tab and the target application of interest and press *Attach*. Once connected, the layout of NVIDIA Nsight Compute changes into stepping mode that allows you to control the execution of any calls into the instrumented API. When connected, the *API Stream* window indicates that the target application waits before the very first API call.



#### 4. Control application execution

Use the *API Stream* window to step into the instrumented API. The dropdown at the top allows switching between different CPU threads of the application. *Step In* (F11), *Step Over* (F10), and *Step Out* (Shift + F11) are available from the *Debug* menu or the corresponding toolbar buttons. While stepping, function return values and function parameters are captured.

ID	API Name	Details	Func Return	Func Parameter
1	cuLaunchKernel		CUDA_SUCCESS()	(0x32aef81c100, 196, 1, 1, 256, 1, 1, 0, 0x0, 0x32aef8ff88(0x2810e7a3b00), 0x0)
2	cuDeviceGetCount		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00))
3	cuDeviceGet		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0)
4	cuEventCreate		CUDA_SUCCESS()	("Quadro P1000", 256, 0x0)
5	cuEventSetAttribute		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0x2000000000000000, 0x2000000000000000, 0x0)
6	cuEventDestroy		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0x0)
7	cuCtxCreate_v2		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0x0)
8	cuCtxGetCurrent		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00))
9	cuModuleLoadDataEx		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0x7f7fed1f1afe, 0x2000000000000000, "VecAdd_kernel")
10	cuModuleGetFunction		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0x7f7fed1f1afe, 0x2000000000000000)
11	cuMemAlloc_v2		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0x0)
12	cuMemAlloc_v2		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 200000)
13	cuMemAlloc_v2		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 200000)
14	cuMemcpyAsync_v2		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0x32aef932d00, 200000)
15	cuMemcpyAsync_v2		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 0x32aef932d00, 200000)
16	cuCtxsetCurrent		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00))
17	cuLaunchKernel		CUDA_SUCCESS()	(0x32aef8ff88(0x2810e7a3b00), 196, 1, 1, 256, 1, 1, 0, 0x0, 0x32aef8ff88(0x2810e7a3b00), 0x0)
18	VecAdd_kernel		VecAdd_ker	

Use *Resume* (F5) and *Pause* to allow the program to run freely. Freeze control is available to define the behavior of threads currently not in focus, i.e. selected in the thread drop down. By default, the *API Stream* stops on any API call that returns an error code. This can be toggled in the *Debug* menu by *Break On API Error*.

#### 5. Isolate a kernel launch

To quickly isolate a kernel launch for profiling, use the *Run to Next Kernel* button in the toolbar of the *API Stream* window to jump to the next kernel launch. The execution will stop before the kernel launch is executed.

API Stream			
10336 > VecAdd_kernel		All	Export to CSV
ID	API Name	Details	Func.Return
1	cudaInit		CUDA_SUCCESS()
2	cuDeviceGetCount		CUDA_SUCCESS()
3	cuDeviceGet		CUDA_SUCCESS()
4	cuDeviceGetName	"Quadro P1000", 256, 0x00	CUDA_SUCCESS()
5	cuDeviceGetAttribute	cudaDeviceAttributeComputeCapability_Major(75), 0x00	CUDA_SUCCESS()
6	cuDeviceGetAttribute	cudaDeviceAttributeComputeCapability_Minor(76), 0x00	CUDA_SUCCESS()
7	cuCtxCreate_v2		CUDA_SUCCESS()
8	cuCtxGetCurrent		CUDA_SUCCESS()
9	cuModuleLoadFromBin		CUDA_SUCCESS()
10	cuModuleGetFunction		CUDA_SUCCESS()
11	cuMemAlloc_v2		CUDA_SUCCESS()
12	cuMemAlloc_v2		CUDA_SUCCESS()
13	cuMemAlloc_v2		CUDA_SUCCESS()
14	cuMemAlloc_v2		CUDA_SUCCESS()
15	cuMemAlloc_v2		CUDA_SUCCESS()
16	cuCtxGetCurrent		CUDA_SUCCESS()
17	cuLaunchKernel	(0x2010e7a000, 196, 1, 1, 256, 1, 1, 0, 0x0, 0x352a4ff888(0x2010e7a000), 0x0)	VecAdd_Kern

## 6. Profile a kernel launch

Once the execution of the target application is suspended at a kernel launch, additional actions become available in the UI. These actions are either available from the menu or from the toolbar. Please note that the actions are disabled, if the API stream is not at a qualifying state (not at a kernel launch or launching on an unsupported GPU). To profile, press *Profile Kernel* and wait until the result is shown in the [Profiler Report](#). Profiling progress is reported in the lower right corner status bar.

Instead of manually selecting *Profile*, it is also possible to enable *Auto Profile* from the *Profile* menu. If enabled, each kernel matching the current kernel filter (if any) will be profiled using the current section configuration. This is especially useful if an application is to be profiled unattended, or the number of kernel launches to be profiled is very large. Sections can be enabled or disabled using the [Metric Selection](#) tool window.

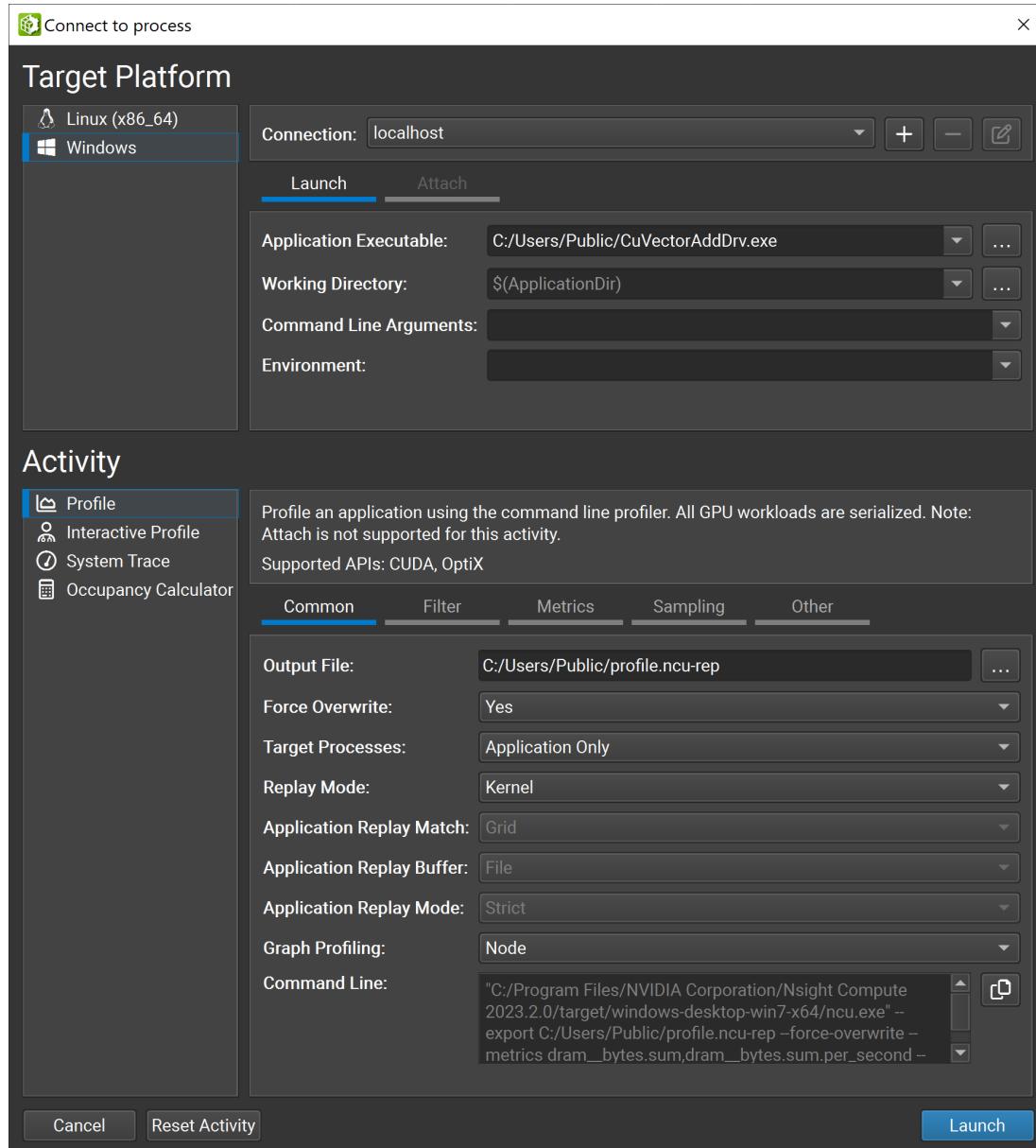
*Profile Series* allows to configure the collection of a set of profile results at once. Each result in the set is profiled with varying parameters. Series are useful to investigate the behavior of a kernel across a large set of parameters without the need to recompile and rerun the application many times.

For a detailed description of the options available in this activity, see [Interactive Profile Activity](#).

## 2.2. Non-Interactive Profile Activity

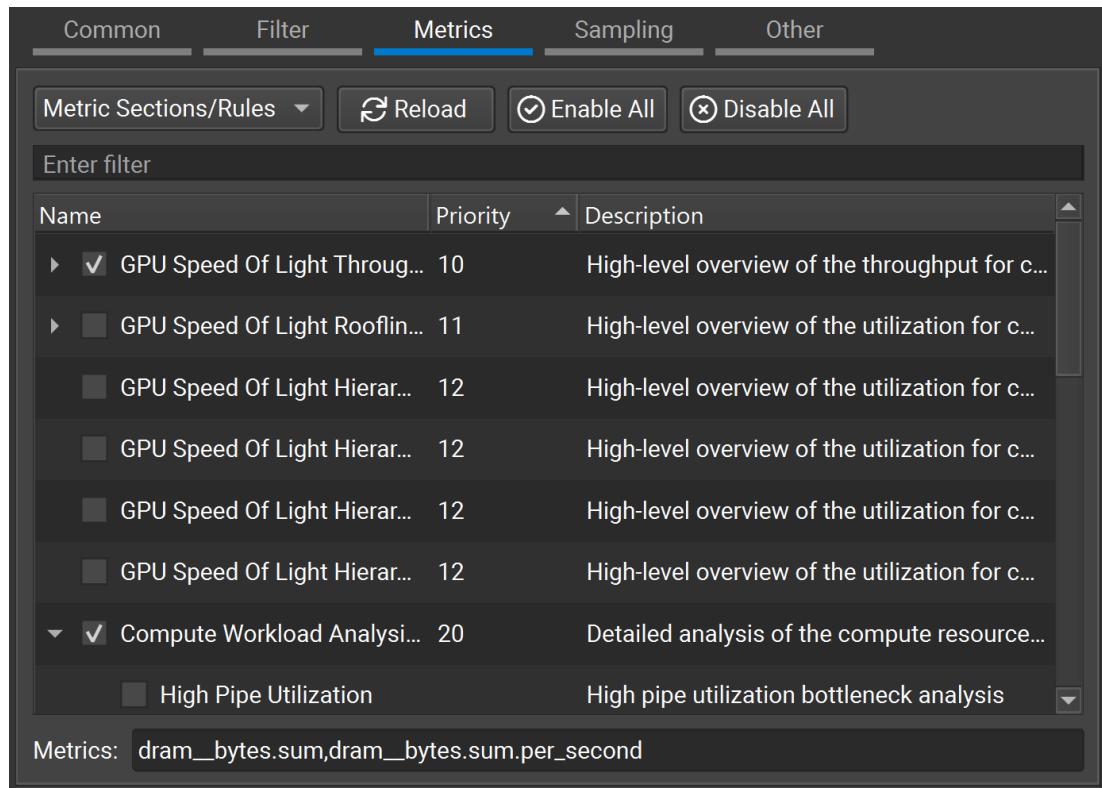
### 1. Launch the target application from NVIDIA Nsight Compute

When starting NVIDIA Nsight Compute, the *Welcome Page* will appear. Click on *Quick Launch* to open the *Connection* dialog. If the *Connection* dialog doesn't appear, you can open it using the *Connect* button from the main toolbar, as long as you are not currently connected. Select your target platform on the left-hand side and your localhost from the *Connection* drop down. Then, fill in the launch details. In the *Activity* panel, select the *Profile* activity to initiate a session that pre-configures the profile session and launches the command line profiler to collect the data. Provide the *Output File* name to enable starting the session with the *Launch* button.



## 2. Additional Launch Options

For more details on these options, see [Command Line Options](#). The options are grouped into tabs: The *Filter* tab exposes the options to specify which kernels should be profiled. Options include the kernel regex filter, the number of launches to skip, and the total number of launches to profile. The *Sections* tab allows you to select which sections should be collected for each kernel launch. Hover over a section to see its description as a tool-tip. To change the sections that are enabled by default, use the [Metric Selection](#) tool window. The *Sampling* tab allows you to configure sampling options for each kernel launch. The *Other* tab includes the option to collect NVTX information or custom metrics via the `--metrics` option.

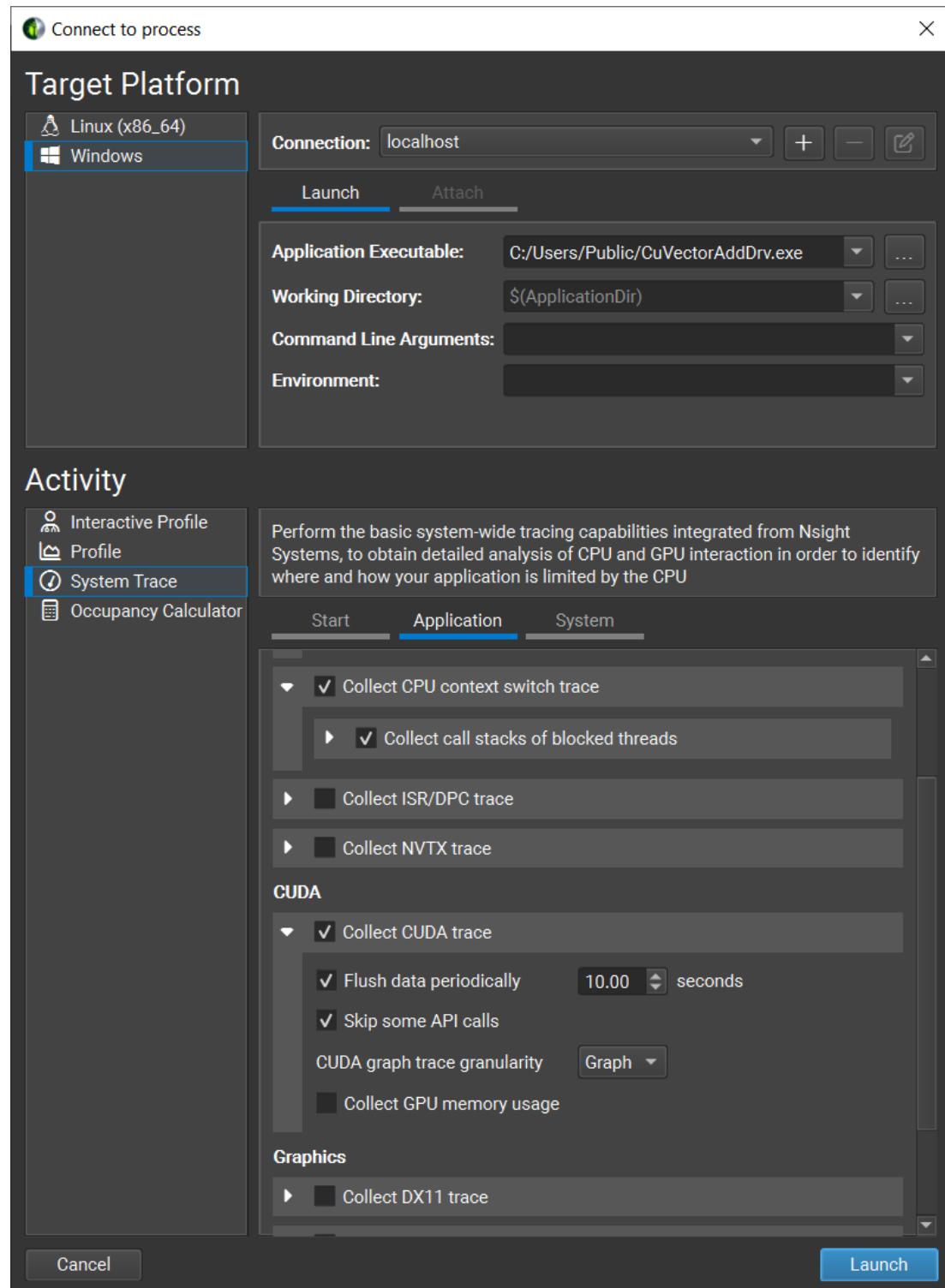


For a detailed description of the options available in this activity, see [Profile Activity](#).

## 2.3. System Trace Activity

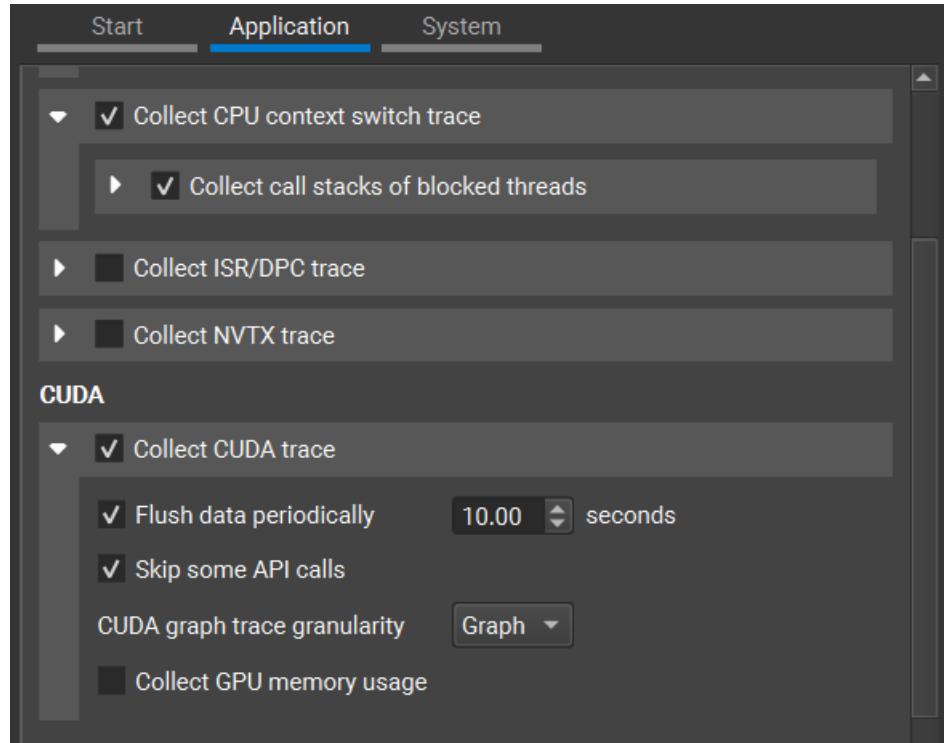
### 1. Launch the target application from NVIDIA Nsight Compute

When starting NVIDIA Nsight Compute, the *Welcome Page* will appear. Click on *Quick Launch* to open the *Connection* dialog. If the *Connection* dialog doesn't appear, you can open it using the *Connect* button from the main toolbar, as long as you are not currently connected. Select your local target platform on the left-hand side and your localhost from the *Connection* drop down. Then, fill in the launch details. In the *Activity* panel, select the *System Trace* activity to initiate a session with pre-configured settings. Press *Launch* to start the session.

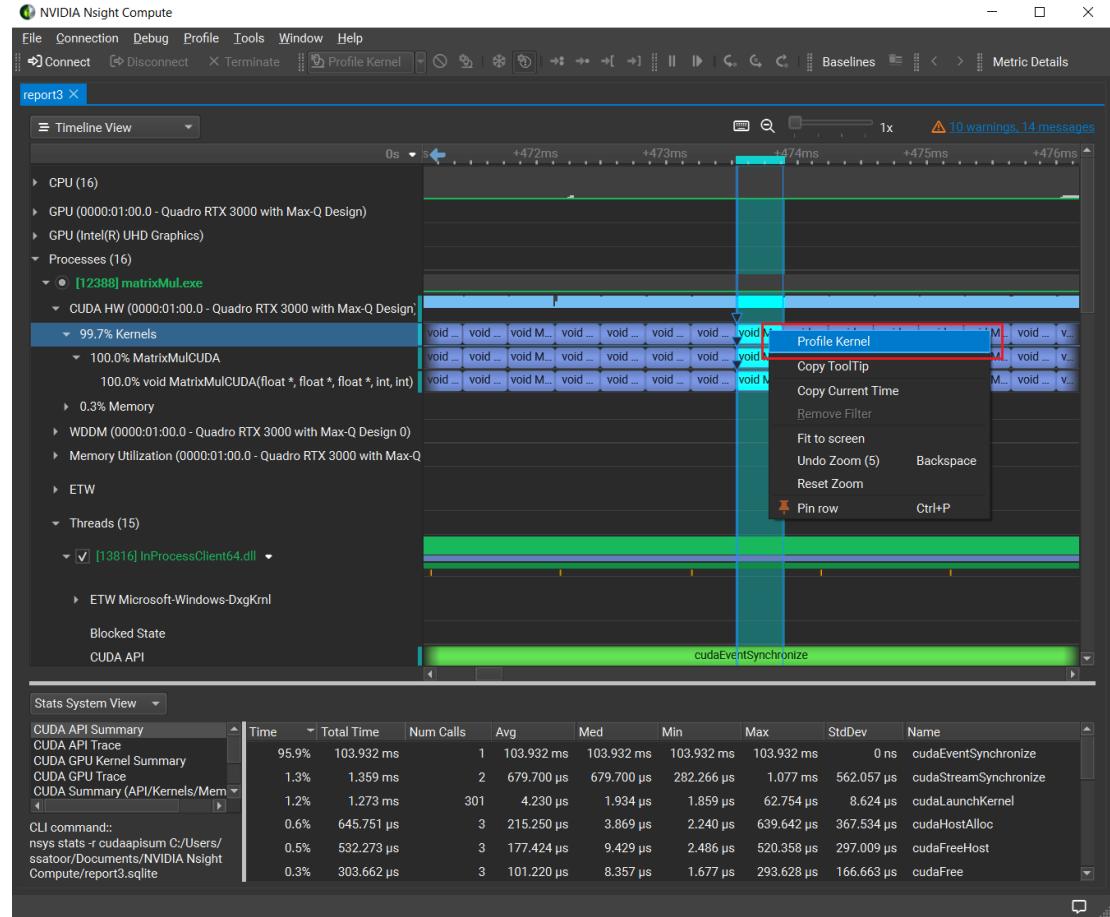


## 2. Additional Launch Options

For more details on these options, see [System-Wide Profiling Options](#).



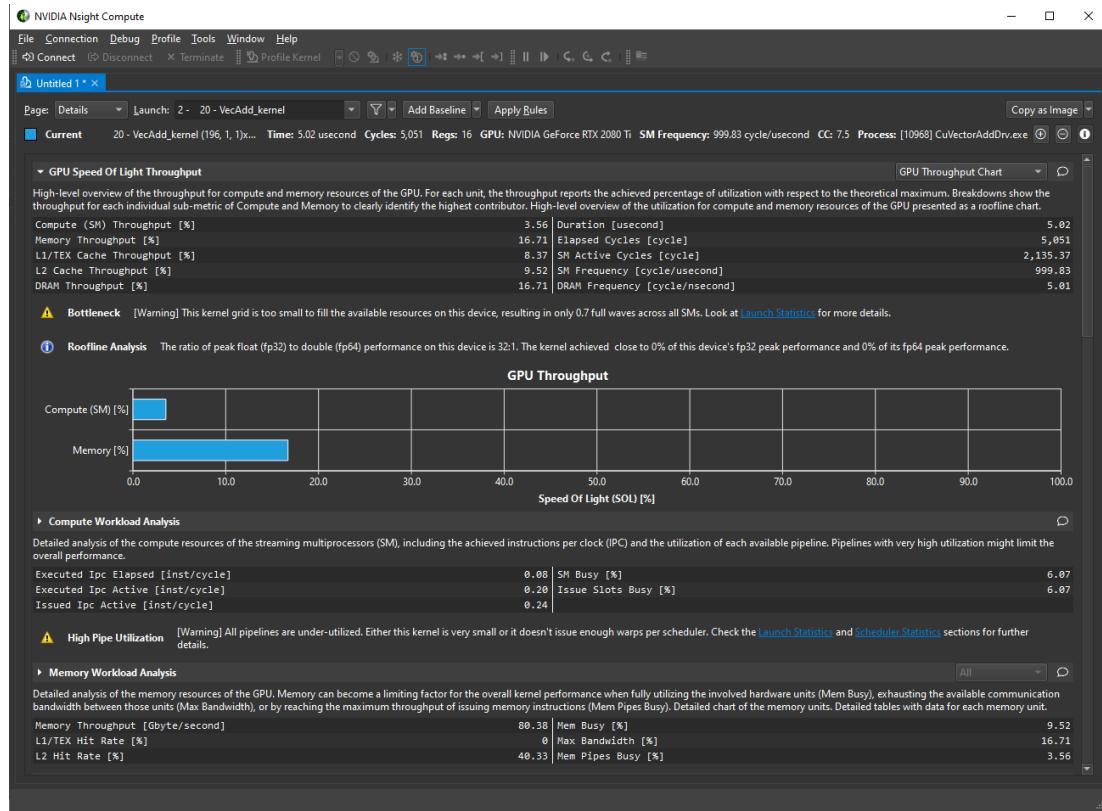
3. Once the session is completed, the [Nsight Systems](#) report is opened in a new document. By default, the timeline view is shown. It provides detailed information of the activity of the CPU and GPUs and helps understanding the overall behavior and performance of application. Once a CUDA kernel is identified to be on the critical path and not meeting the performance expectations, right click on the kernel launch on timeline and select *Profile Kernel* from the context menu. A new [Connection Dialog](#) opens up that is already preconfigured to profile the selected kernel launch. Proceed with optimizing the selected kernel using [Non-Interactive Profile Activity](#)



## 2.4. Navigate the Report

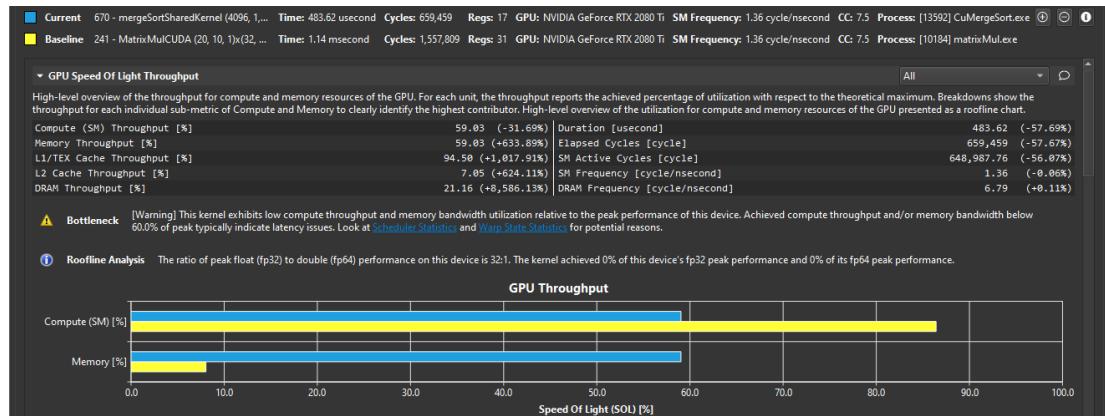
### 1. Navigate the report

The profile report comes up by default on the *Details* page. You can switch between different **Report Pages** of the report with the dropdown labeled **Page** on the top-left of the report. You can also use ***Ctrl + Shift + N*** and ***Ctrl + Shift + P*** shortcut keys or corresponding toolbar button to navigate next and previous pages, respectively. A report can contain any number of results from kernel launches. The **Result** dropdown allows switching between the different results in a report.



## 2. Difffing multiple results

On the *Details* page, press the button *Add Baseline* in order for the current result to become the baseline all other results from this report and any other report opened in the same instance of NVIDIA Nsight Compute get compared to. If a baseline is set, every element on the Details page shows two values: The current value of the result in focus and the corresponding value of the baseline or the percentage of change from the corresponding baseline value.



Use the *Clear Baselines* entry from the dropdown button, the Profile menu or the corresponding toolbar button to remove all baselines. For more information see [Baselines](#).

## 3. Executing rules

On the *Details* page some sections may provide rules. Press the *Apply* button to execute an individual rule. The *Apply Rules* button on the top executes all available rules for the current result in focus. Rules can be user-defined too. For more information see the [Customization Guide](#).

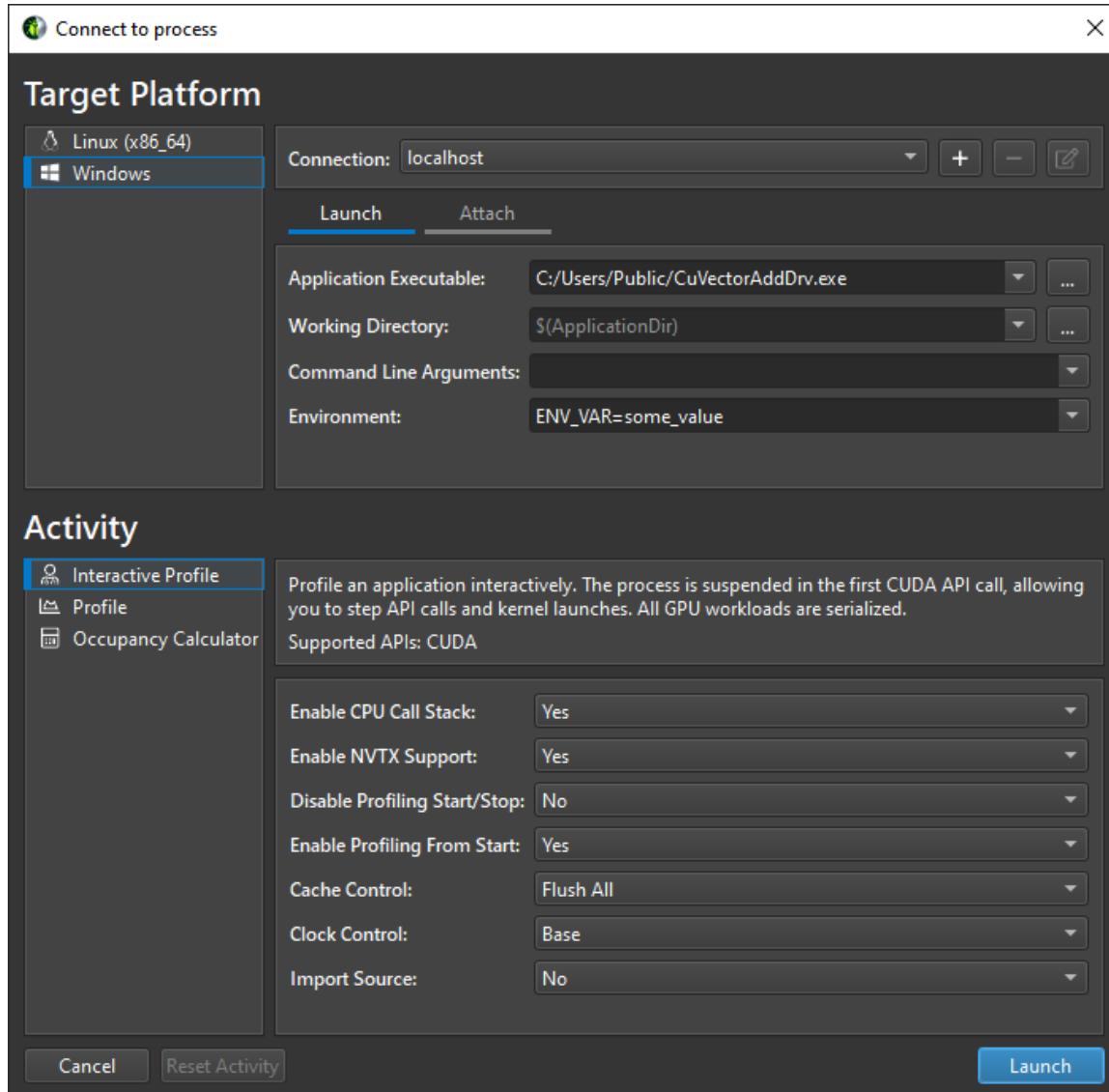
 **Bottleneck** [Warning] This kernel exhibits low compute throughput and memory bandwidth utilization relative to the peak performance of this device. Achieved compute throughput and/or memory bandwidth below 60.0% of peak typically indicate latency issues. Look at [Scheduler Statistics](#) and [Varp State Statistics](#) for potential reasons.

 **Roofline Analysis** The ratio of peak float (fp32) to double (fp64) performance on this device is 32:1. The kernel achieved 0% of this device's fp32 peak performance and 0% of its fp64 peak performance.

# Chapter 3.

# CONNECTION DIALOG

Use the *Connection Dialog* to launch and attach to applications on your local and remote platforms. Start by selecting the *Target Platform* for profiling. By default (and if supported) your local platform will be selected. Select the platform on which you would like to start the target application or connect to a running process.



When using a remote platform, you will be asked to select or create a *Connection* in the top drop down. To create a new connection, select + and enter your connection details. When using the local platform, *localhost* will be selected as the default and no further connection settings are required. You can still create or select a remote connection, if profiling will be on a remote system of the same platform.

Depending on your target platform, select either *Launch* or *Remote Launch* to launch an application for profiling on the target. Note that *Remote Launch* will only be available if supported on the target platform.

Fill in the following launch details for the application:

- ▶ **Application Executable:** Specifies the root application to launch. Note that this may not be the final application that you wish to profile. It can be a script or launcher that creates other processes.
- ▶ **Working Directory:** The directory in which the application will be launched.

- ▶ **Command Line Arguments:** Specify the arguments to pass to the application executable.
- ▶ **Environment:** The environment variables to set for the launched application.

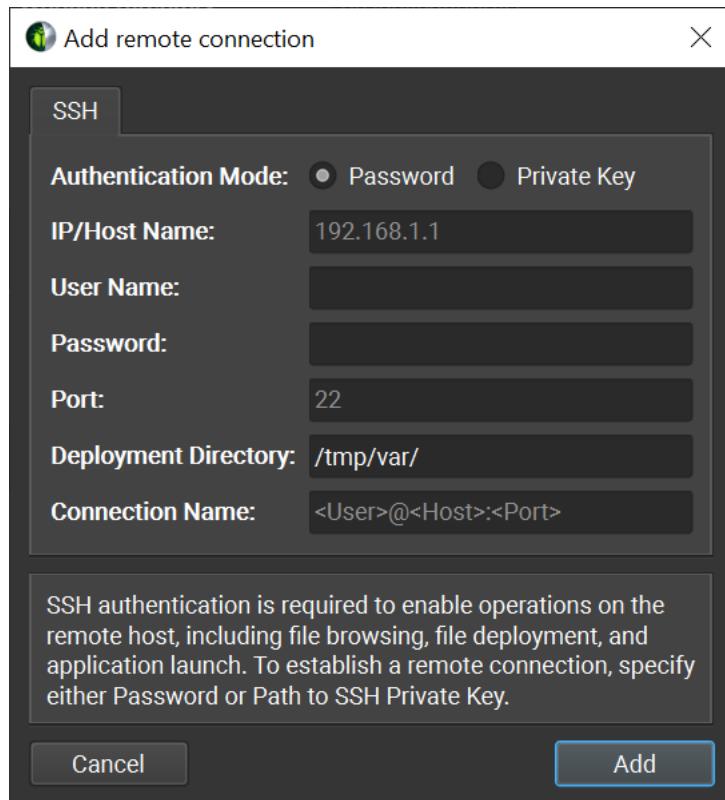
Select *Attach* to attach the profiler to an application already running on the target platform. This application must have been started using another NVIDIA Nsight Compute CLI instance. The list will show all application processes running on the target system which can be attached. Select the refresh button to re-create this list.

Finally, select the *Activity* to be run on the target for the launched or attached application. Note that not all activities are necessarily compatible with all targets and connection options. Currently, the following activities exist:

- ▶ Interactive Profile Activity
- ▶ Profile Activity
- ▶ System Trace Activity
- ▶ Occupancy Calculator

## 3.1. Remote Connections

Remote devices that support SSH can also be configured as a target in the *Connection Dialog*. To configure a remote device, ensure an SSH-capable *Target Platform* is selected, then press the + button. The following configuration dialog will be presented.

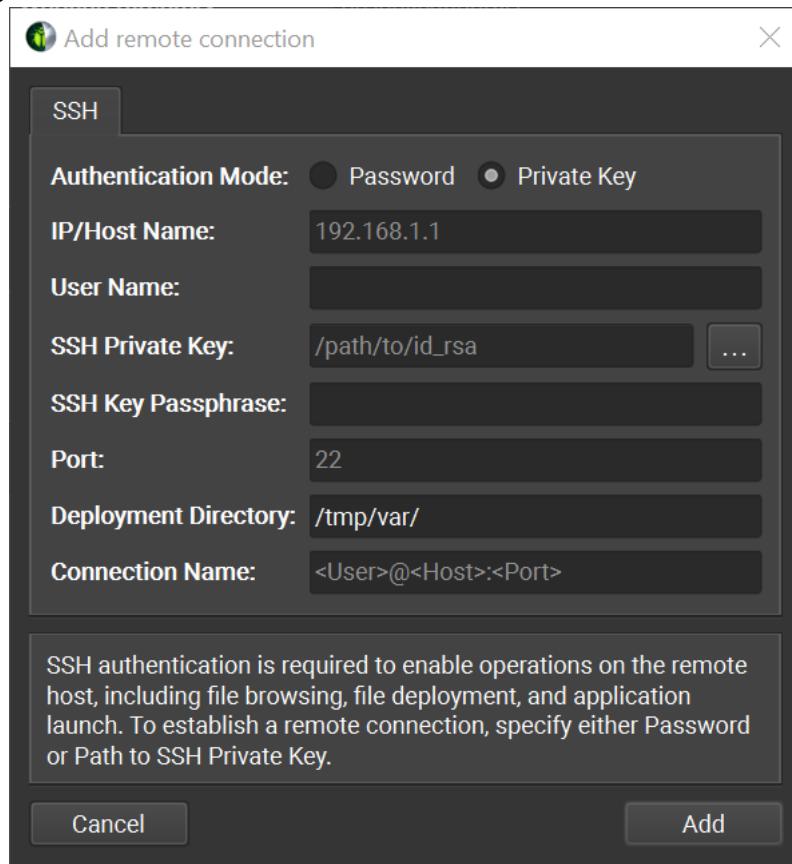


NVIDIA Nsight Compute supports both password and private key authentication methods. In this dialog, select the authentication method and enter the following information:

► **Password**

- **IP/Host Name:** The IP address or host name of the target device.
- **User Name:** The user name to be used for the SSH connection.
- **Password:** The user password to be used for the SSH connection.
- **Port:** The port to be used for the SSH connection. (The default value is 22)
- **Deployment Directory:** The directory to use on the target device to deploy supporting files. The specified user must have write permissions to this location.
- **Connection Name:** The name of the remote connection that will show up in the *Connection Dialog*. If not set, it will default to <User>@<Host>:<Port>.

► **Private Key**



- **IP/Host Name:** The IP address or host name of the target device.
- **User Name:** The user name to be used for the SSH connection.
- **SSH Private Key:** The private key that is used to authenticate to SSH server.
- **SSH Key Passphrase:** The passphrase for your private key.
- **Port:** The port to be used for the SSH connection. (The default value is 22)
- **Deployment Directory:** The directory to use on the target device to deploy supporting files. The specified user must have write permissions to this location.

- ▶ **Connection Name:** The name of the remote connection that will show up in the *Connection Dialog*. If not set, it will default to <User>@<Host>:<Port>.

In addition to keyfiles specified by path and plain password authentication, NVIDIA Nsight Compute supports keyboard-interactive authentication, standard keyfile path searching and SSH agents.

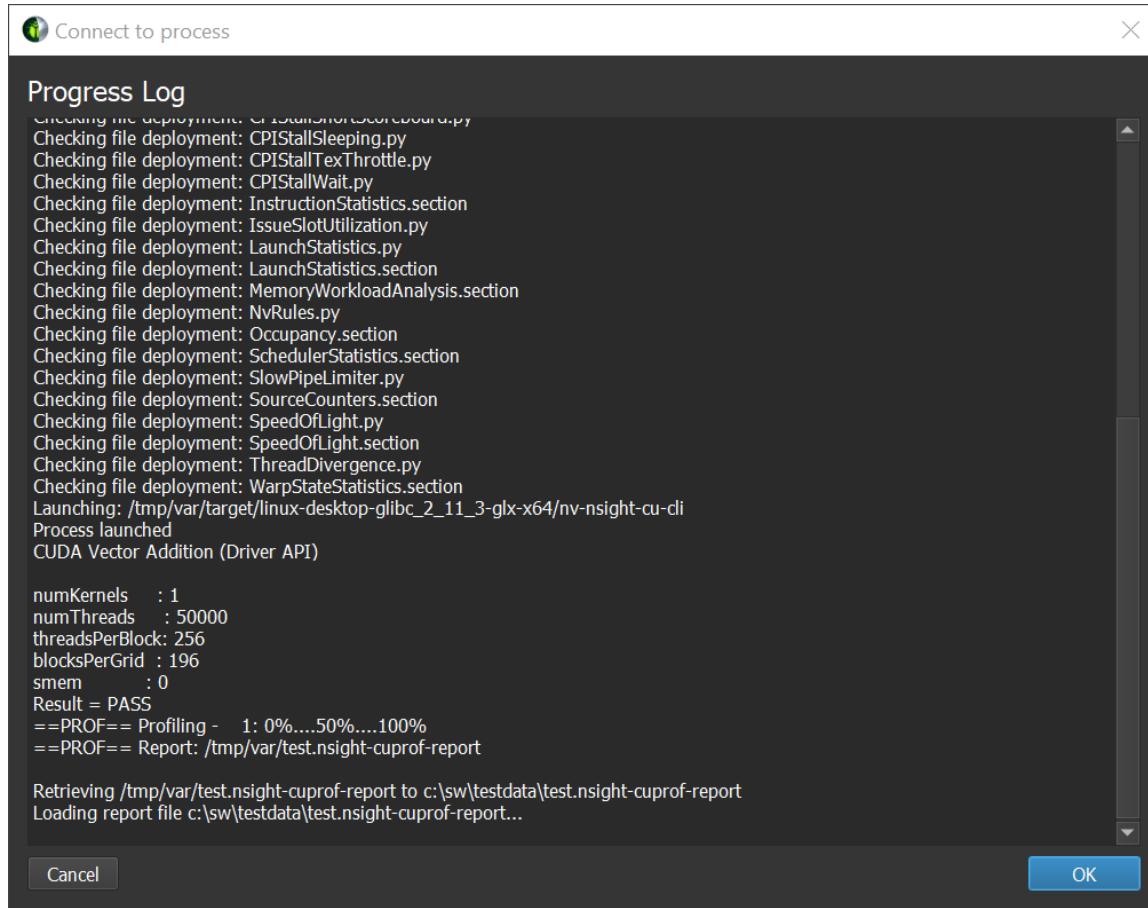
When all information is entered, click the *Add* button to make use of this new connection.

When a remote connection is selected in the *Connection Dialog*, the *Application Executable* file browser will browse the remote file system using the configured SSH connection, allowing the user to select the target application on the remote device.

When an activity is launched on a remote device, the following steps are taken:

1. The command line profiler and supporting files are copied into the *Deployment Directory* on the the remote device. (Only files that do not exist or are out of date are copied.)
2. Communication channels are opened to prepare for the traffic between the UI and the *Application Executable*.
  - ▶ For *Interactive Profile* activities, a *SOCKS proxy* is started on the host machine.
  - ▶ For *Non-Interactive Profile* activities, a remote forwarding channel is opened on the target machine to tunnel profiling information back to the host.
3. The *Application Executable* is executed on the remote device.
  - ▶ For *Interactive Profile* activities, a connection is established to the remote application and the profiling session begins.
  - ▶ For *Non-Interactive Profile* activities, the remote application is executed under the command line profiler and the specified report file is generated.
4. For non-interactive profiling activities, the generated report file is copied back to the host, and opened.

The progress of each of these steps is presented in the *Progress Log*.



Note that once either activity type has been launched remotely, the tools necessary for further profiling sessions can be found in the *Deployment Directory* on the remote device.

On Linux and Mac host platforms, NVIDIA Nsight Compute supports SSH remote profiling on target machines which are not directly addressable from the machine the UI is running on through the **ProxyJump** and **ProxyCommand** SSH options.

These options can be used to specify intermediate hosts to connect to or actual commands to run to obtain a socket connected to the SSH server on the target host and can be added to your SSH configuration file.

Note that for both options, NVIDIA Nsight Compute runs external commands and does not implement any mechanism to authenticate to the intermediate hosts using the credentials entered in the [Connection Dialog](#). These credentials will only be used to authenticate to the final target in the chain of machines.

When using the **ProxyJump** option NVIDIA Nsight Compute uses the *OpenSSH client* to establish the connection to the intermediate hosts. This means that in order to use **ProxyJump** or **ProxyCommand**, a version of OpenSSH supporting these options must be installed on the host machine.

A common way to authenticate to the intermediate hosts in this case is to use a *SSH agent* and have it hold the private keys used for authentication.

Since the *OpenSSH SSH client* is used, you can also use the *SSH askpass* mechanism to handle these authentications in an interactive manner.

It might happen on slow networks that connections used for remote profiling through SSH time out. If this is the case, the **ConnectTimeout** option can be used to set the desired timeout value.

A known limitation of the remote profiling through SSH is that problems may arise if NVIDIA Nsight Compute tries to do remote profiling through *SSH* by connecting to the same machine it is running on. In this case, the workaround is to do local profiling through **localhost**.

For more information about available options for the *OpenSSH client* and the ecosystem of tools it can be used with for authentication refer to the official [manual pages](#).

## 3.2. Interactive Profile Activity

The *Interactive Profile* activity allows you to initiate a session that controls the execution of the target application, similar to a debugger. You can step API calls and workloads (CUDA kernels), pause and resume, and interactively select the kernels of interest and which metrics to collect.

This activity does currently not support profiling or attaching to child processes.

- ▶ **Enable CPU Call Stack**

Collect the CPU-sided Call Stack at the location of each profiled kernel launch.

- ▶ **Enable NVTX Support**

Collect NVTX information provided by the application or its libraries. Required to support stepping to specific NVTX contexts.

- ▶ **Disable Profiling Start/Stop**

Ignore calls to **cu(da) ProfilerStart** or **cu(da) ProfilerStop** made by the application.

- ▶ **Enable Profiling From Start**

Enables profiling from the application start. Disabling this is useful if the application calls **cu(da) ProfilerStart** and kernels before the first call to this API should not be profiled. Note that disabling this does not prevent you from manually profiling kernels.

- ▶ **Cache Control**

Control the behavior of the GPU caches during profiling. Allowed values: For *Flush All*, all GPU caches are flushed before each kernel replay iteration during profiling. While metric values in the execution environment of the application might be slightly different without invalidating the caches, this mode offers the most reproducible metric results across the replay passes and also across multiple runs of the target application.

For *Flush None*, no GPU caches are flushed during profiling. This can improve performance and better replicates the application behavior if only a single kernel replay pass is necessary for metric collection. However, some metric results will

vary depending on prior GPU work, and between replay iterations. This can lead to inconsistent and out-of-bounds metric values.

- ▶ **Clock Control**

Control the behavior of the GPU clocks during profiling. Allowed values: For *Base*, GPC and memory clocks are locked to their respective base frequency during profiling. This has no impact on thermal throttling. For *None*, no GPC or memory frequencies are changed during profiling.

- ▶ **Import Source**

Enables permanently importing available source files into the report. Missing source files are searched in [Source Lookup](#) folders. Source information must be embedded in the executable, e.g. via the `-lineinfo` compiler option. Imported files are used in the *CUDA-C* view on the [Source Page](#).

- ▶ **Graph Profiling**

Set if CUDA graphs should be stepped and profiled as individual *Nodes* or as complete *Graphs*. See the [Kernel Profiling Guide](#) for more information on this mode.

### 3.3. Profile Activity

The *Profile* activity provides a traditional, pre-configurable profiler. After configuring which kernels to profile, which metrics to collect, etc, the application is run under the profiler without interactive control. The activity completes once the application terminates. For applications that normally do not terminate on their own, e.g. interactive user interfaces, you can cancel the activity once all expected kernels are profiled.

This activity does not support attaching to processes previously launched via NVIDIA Nsight Compute. These processes will be shown grayed out in the *Attach* tab.

- ▶ **Output File**

Path to report file where the collected profile should be stored. If not present, the report extension `.ncu-rep` is added automatically. The placeholder `%i` is supported for the filename component. It is replaced by a sequentially increasing number to create a unique filename. This maps to the `--export` command line option.

- ▶ **Force Overwrite**

If set, existing report file are overwritten. This maps to the `--force-overwrite` command line option.

- ▶ **Target Processes**

Select the processes you want to profile. In mode *Application Only*, only the root application process is profiled. In mode *all*, the root application process and all its child processes are profiled. This maps to the `--target-processes` command line option.

- ▶ **Replay Mode**

Select the method for replaying kernel launches multiple times. In mode *Kernel*, individual kernel launches are replayed transparently during the single execution of the target application. In mode *Application*, the entire target application is

relaunched multiple times. In each iteration, additional data for the target kernel launches is collected. Application replay requires the program execution to be deterministic. This maps to the `--replay-mode` command line option. See the [Kernel Profiling Guide](#) for more details on the replay modes.

- ▶ **Graph Profiling**

Set if CUDA graphs should be profiled as individual *Nodes* or as complete *Graphs*.

- ▶ **Additional Options**

All remaining options map to their command line profiler equivalents. See the [Command Line Options](#) for details.

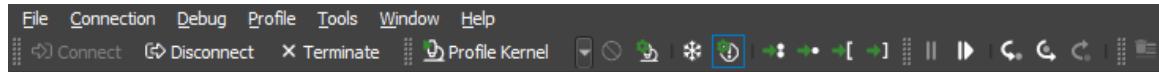
## 3.4. Reset

Entries in the connection dialog are saved as part of the current [project](#). When working in a custom project, simply close the project to reset the dialog.

When not working in a custom project, entries are stored as part of the *default project*. You can delete all information from the default project by closing NVIDIA Nsight Compute and then [deleting the project file from disk](#).

# Chapter 4. MAIN MENU AND TOOLBAR

Information on the main menu and toolbar.



## 4.1. Main Menu

- ▶ **File**
  - ▶ **New Project** Create new profiling [Projects](#) with the [New Project Dialog](#).
  - ▶ **Open Project** Open an existing profiling project.
  - ▶ **Recent Projects** Open an existing profiling project from the list of recently used projects.
  - ▶ **Save Project** Save the current profiling project.
  - ▶ **Save Project As** Save the current profiling project with a new filename.
  - ▶ **Close Project** Close the current profiling project.
  - ▶ **New File** Create a new file.
  - ▶ **Open File** Open an existing file.
  - ▶ **Open Remote File**

Download an existing file from a remote host and open it locally. The opened file will only exist in memory and will not be written to the local machine's disk unless the user explicitly saves it. For more information concerning the selection of a remote host to download the file from, see the section about [Remote Connections](#).

Only a subset of file types that are supported locally can be opened from a remote target. The following table lists file types that can be opened remotely.

Table 1 Remote File Type Support

Extensions	Description	Supported
ncu-report	Nsight Compute Profiler Report	Yes

Extensions	Description	Supported
ncu-occ	Occupancy Calculator File	Yes
ncu-bvh	OptiX AS Viewer File	Yes (except on MacOSX)
section	Section Description	No
cubin	Cubin File	No
cuh,h,hpp	Header File	No
c,cpp,cu	Source File	No
txt	Text file	No
nsight-cuprof-report	Nsight Compute Profiler Report (legacy)	Yes

- ▶ **Save** Save the current file
- ▶ **Save As** Save a copy of the current file with a different name or type or in a different location.
- ▶ **Save All Files** Save all open files.
- ▶ **Close** Close the current file.
- ▶ **Close All Files** Close all open files.
- ▶ **Recent Files** Open an existing file from the list of recently used files.
- ▶ **Exit** Exit Nsight Compute.
- ▶ Connection
  - ▶ **Connect** Open the [Connection Dialog](#) to launch or attach to a target application. Disabled when already connected.
  - ▶ **Disconnect** Disconnect from the current target application, allows the application to continue normally and potentially re-attach.
  - ▶ **Terminate** Disconnect from and terminate the current target application immediately.
- ▶ Debug
  - ▶ **Pause** Pause the target application at the next intercepted API call or launch.
  - ▶ **Resume** Resume the target application.
  - ▶ **Step In** Step into the current API call or launch to the next nested call, if any, or the subsequent API call, otherwise.
  - ▶ **Step Over** Step over the current API call or launch and suspend at the next, non-nested API call or launch.
  - ▶ **Step Out** Step out of the current nested API call or launch to the next, non-parent API call or launch one level above.
  - ▶ **Freeze API**

When disabled, all CPU threads are enabled and continue to run during stepping or resume, and all threads stop as soon as at least one thread arrives at the next API call or launch. This also means that during stepping or resume the currently selected thread might change as the old selected thread makes no forward progress and the API Stream automatically switches to the thread with a new API call or launch. When enabled, only the currently selected CPU thread is enabled. All other threads are disabled and blocked.

Stepping now completes if the current thread arrives at the next API call or launch. The selected thread never changes. However, if the selected thread does not call any further API calls or waits at a barrier for another thread to make progress, stepping may not complete and hang indefinitely. In this case, pause, select another thread, and continue stepping until the original thread is unblocked. In this mode, only the selected thread will ever make forward progress.

- ▶ **Break On API Error** When enabled, during resume or stepping, execution is suspended as soon as an API call returns an error code.
- ▶ **Run to Next Kernel** See [API Stream](#) tool window.
- ▶ **Run to Next API Call** See [API Stream](#) tool window.
- ▶ **Run to Next Range Start** See [API Stream](#) tool window.
- ▶ **Run to Next Range End** See [API Stream](#) tool window.
- ▶ **API Statistics** Opens the [API Statistics](#) tool window
- ▶ **API Stream** Opens the [API Stream](#) tool window
- ▶ **Resources** Opens the [Resources](#) tool window
- ▶ **NVTX** Opens the [NVTX](#) tool window
- ▶ Profile
  - ▶ **Profile Kernel** When suspended at a kernel launch, select the profile using the current configuration.
  - ▶ **Profile Series** When suspended at a kernel launch, open the Profile Series configuration dialog to setup and collect a series of profile results.
  - ▶ **Auto Profile** Enable or disable auto profiling. If enabled, each kernel matching the current kernel filter (if any) will be profiled using the current section configuration.
  - ▶ **Baselines** Opens the [Baselines](#) tool window.
  - ▶ **Clear Baselines** Clear all current baselines.
  - ▶ **Import Source** Permanently import resolved source files into the report. Existing content may be overwritten.
  - ▶ **Section/Rules Info** Opens the [Metric Selection](#) tool window.
- ▶ Tools
  - ▶ **Project Explorer** Opens the [Project Explorer](#) tool window.
  - ▶ **Output Messages** Opens the Output Messages tool window.
  - ▶ **Options** Opens the [Options](#) dialog.
- ▶ Window
  - ▶ **Save Window Layout** Allows you to specify a name for the current layout. The layouts are saved to a Layouts folder in the documents directory as named ".nvlayout" files.
  - ▶ **Apply Window Layout** Once you have saved a layout, you can restore them by using the "Apply Window Layout" menu entry. Simply select the entry from sub-menu you want to apply.
  - ▶ **Manage Window Layout** Allows you to delete or rename old layouts.
  - ▶ **Restore Default Layout** Restore views to their original size and position.
  - ▶ **Show Welcome Page** Opens the [Welcome Page](#).

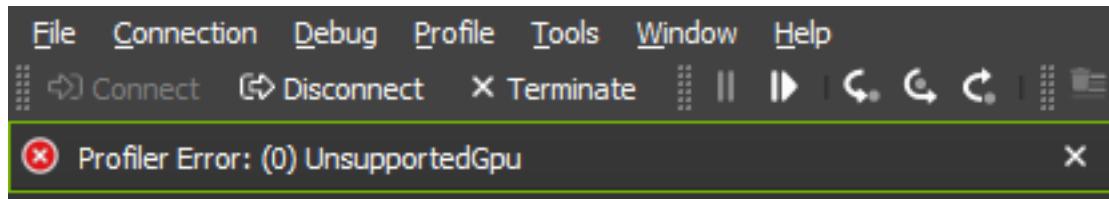
- ▶ Help
  - ▶ Documentation Opens the latest documentation for NVIDIA Nsight Compute online.
  - ▶ Documentation (local) Opens the local HTML documentation for NVIDIA Nsight Compute that has shipped with the tool.
  - ▶ Check For Updates Checks online if a newer version of NVIDIA Nsight Compute is available for download.
  - ▶ Reset Application Data Reset all NVIDIA Nsight Compute configuration data saved on disk, including option settings, default paths, recent project references etc. This will not delete saved reports.
  - ▶ Send Feedback Opens a dialog that allows you to send bug reports and suggestions for features. Optionally, the feedback includes basic system information, screenshots, or additional files (such as profile reports).
  - ▶ About Opens the About dialog with information about the version of NVIDIA Nsight Compute.

## 4.2. Main Toolbar

The main toolbar shows commonly used operations from the main menu. See [Main Menu](#) for their description.

## 4.3. Status Banners

Status banners are used to display important messages, such as profiler errors. The message can be dismissed by clicking the 'X' button. The number of banners shown at the same time is limited and old messages can get dismissed automatically if new ones appear. Use the *Output Messages* window to see the complete message history.

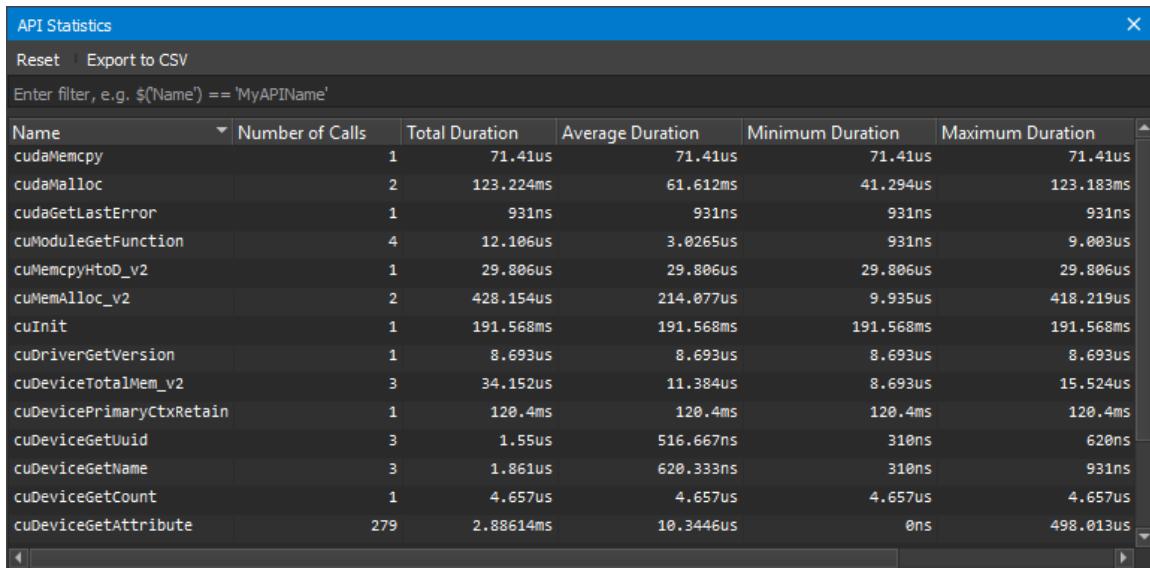


# Chapter 5.

# TOOL WINDOWS

## 5.1. API Statistics

The *API Statistics* window is available when NVIDIA Nsight Compute is connected to a target application. It opens by default as soon as the connection is established. It can be re-opened using *Debug > API Statistics* from the main menu.



Name	Number of Calls	Total Duration	Average Duration	Minimum Duration	Maximum Duration
cudaMemcpy	1	71.41us	71.41us	71.41us	71.41us
cudaMalloc	2	123.224ms	61.612ms	41.294us	123.183ms
cudaGetLastError	1	931ns	931ns	931ns	931ns
cuModuleGetFunction	4	12.106us	3.0265us	931ns	9.003us
cuMemcpyHtoD_v2	1	29.806us	29.806us	29.806us	29.806us
cuMemAlloc_v2	2	428.154us	214.077us	9.935us	418.219us
cuInit	1	191.568ms	191.568ms	191.568ms	191.568ms
cuDriverGetVersion	1	8.693us	8.693us	8.693us	8.693us
cuDeviceTotalMem_v2	3	34.152us	11.384us	8.693us	15.524us
cuDevicePrimaryCtxRetain	1	120.4ms	120.4ms	120.4ms	120.4ms
cuDeviceGetUuid	3	1.55us	516.667ns	310ns	620ns
cuDeviceGetName	3	1.861us	620.333ns	310ns	931ns
cuDeviceGetCount	1	4.657us	4.657us	4.657us	4.657us
cuDeviceGetAttribute	279	2.88614ms	10.3446us	0ns	498.013us

Whenever the target application is suspended, it shows a summary of tracked API calls with some statistical information, such as the number of calls, their total, average, minimum and maximum duration. Note that this view cannot be used as a replacement for [Nsight Systems](#) when trying to optimize CPU performance of your application.

The *Reset* button deletes all statistics collected to the current point and starts a new collection. Use the *Export to CSV* button to export the current statistics to a CSV file.

## 5.2. API Stream

The *API Stream* window is available when NVIDIA Nsight Compute is connected to a target application. It opens by default as soon as the connection is established. It can be re-opened using *Debug > API Stream* from the main menu.

API Stream				
	ID	API Name	Details	Func Return
:	114	cudaMalloc		cudaSuccess(0)
:	115	cuMemAlloc_v2		CUDA_SUCCESS(0)
:	116	cudaMemcpy		cudaSuccess(0)
:	117	cuMemcpyHtoD_V2		CUDA_SUCCESS(0)
:	118	cudaGetLastError		cudaSuccess(0)
:	119	cudaLaunchKernel		cudaSuccess(0)
:	120	cuLaunchKernel		CUDA_SUCCESS(0)
:	121	acosh_main	acosh_main	
:	122	cudaDeviceSynchronize		cudaSuccess(0)
:	123	cuCtxSynchronize		CUDA_SUCCESS(0)
:	124	cudaGetLastError		cudaSuccess(0)
:	125	cudaMemcpy		cudaSuccess(0)
:	126	cuMemcpyDtoH_V2		CUDA_SUCCESS(0)
:	127	cudaFree		cudaSuccess(0)
:	128	cuMemFree_v2		CUDA_SUCCESS(0)
:	129	cudaFree		cudaSuccess(0)
:	130	cuMemFree_v2		CUDA_SUCCESS(0)
:	131	cudaDeviceReset		cudaSuccess(0)
:	132	cuCtxGetCurrent		CUDA_SUCCESS(0)
:	133	cuCtxGetApiVersion		CUDA_SUCCESS(0)
:	134	cuDevicePrimaryCtxRe...		CUDA_SUCCESS(0)
→	135	cudaDeviceReset		
:	136	cuCtxGetCurrent		CUDA_SUCCESS(0)
!	137	cuCtxGetApiVersion		CUDA_ERROR_INVALID_CONTEXT(201)

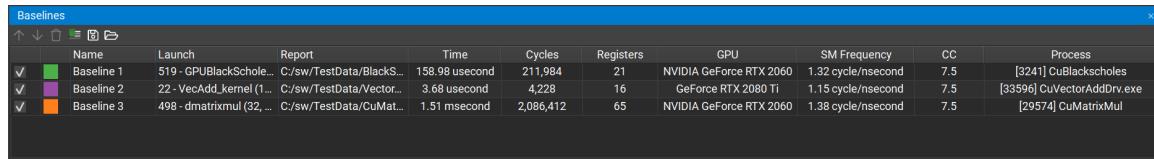
Whenever the target application is suspended, the window shows the history of API calls and traced kernel launches. The currently suspended API call or kernel launch (activity) is marked with a yellow arrow. If the suspension is at a subcall, the parent call is marked with a green arrow. The API call or kernel is suspended before being executed.

For each activity, further information is shown such as the kernel name or the function parameters (*Func Parameters*) and return value (*Func Return*). Note that the function return value will only become available once you step out or over the API call.

Use the *Current Thread* dropdown to switch between the active threads. The dropdown shows the thread ID followed by the current API name. One of several options can be chosen in the trigger dropdown, which are executed by the adjacent >> button. *Run to Next Kernel* resumes execution until the next kernel launch is found in any enabled thread. *Run to Next API Call* resumes execution until the next API call matching *Next Trigger* is found in any enabled thread. *Run to Next Range Start* resumes execution until the next start of an active profiler range is found. Profiler ranges are defined by using the **cu(da) ProfilerStart/Stop** API calls. *Run to Next Range Stop* resumes execution until the next stop of an active profiler range is found. The *API Level* dropdown changes which API levels are shown in the stream. The *Export to CSV* button exports the currently visible stream to a CSV file.

## 5.3. Baselines

The *Baselines* tool window can be opened by clicking the *Baselines* entry in the *Profile* menu. It provides a centralized place from which to manage configured baselines. (Refer to [Baselines](#), for information on how to create baselines from profile results.)



Name	Launch	Report	Time	Cycles	Registers	GPU	SM Frequency	CC	Process
Baseline 1	519 - GPUBlackSchole...	C:/sw/TestData/BlackS...	158.98 usecond	211,984	21	NVIDIA GeForce RTX 2060	1.32 cycle/nsecond	7.5	[3241] CuBlackScholes
Baseline 2	22 - VecAdd_kernel (1...	C:/sw/TestData/Vector...	3.68 usecond	4,228	16	GeForce RTX 2080 Ti	1.15 cycle/nsecond	7.5	[33596] CuVectorAddDrv.exe
Baseline 3	498 - dmatrixmul (32, ...	C:/sw/TestData/CuMat...	1.51 msecound	2,086,412	65	NVIDIA GeForce RTX 2060	1.38 cycle/nsecond	7.5	[29574] CuMatrixMul

The baseline visibility can be controlled by clicking on the check box in a table row. When the check box is checked, the baseline will be visible in the summary header as well as all graphs in all sections. When unchecked the baseline will be hidden and will not contribute to metric difference calculations.

The baseline color can be changed by double-clicking on the color swatch in the table row. The color dialog which is opened provides the ability to choose an arbitrary color as well as offers a palette of predefined colors associated with the stock baseline color rotation.

The baseline name can be changed by double-clicking on the *Name* column in the table row. The name must not be empty and must be less than the *Maximum Baseline Name Length* as specified in the options dialog.

The z-order of a selected baseline can be changed by clicking the *Move Baseline Up* and *Move Baseline Down* buttons in the tool bar. When a baseline is moved up or down its new position will be reflected in the report header as well as in each graph. Currently, only one baseline may be moved at a time.

The selected baselines may be removed by clicking on the *Clear Selected Baselines* button in the tool bar. All baselines can be removed at once by clicking on the *Clear All Baselines* button, from either the global tool bar or the tool window tool bar.

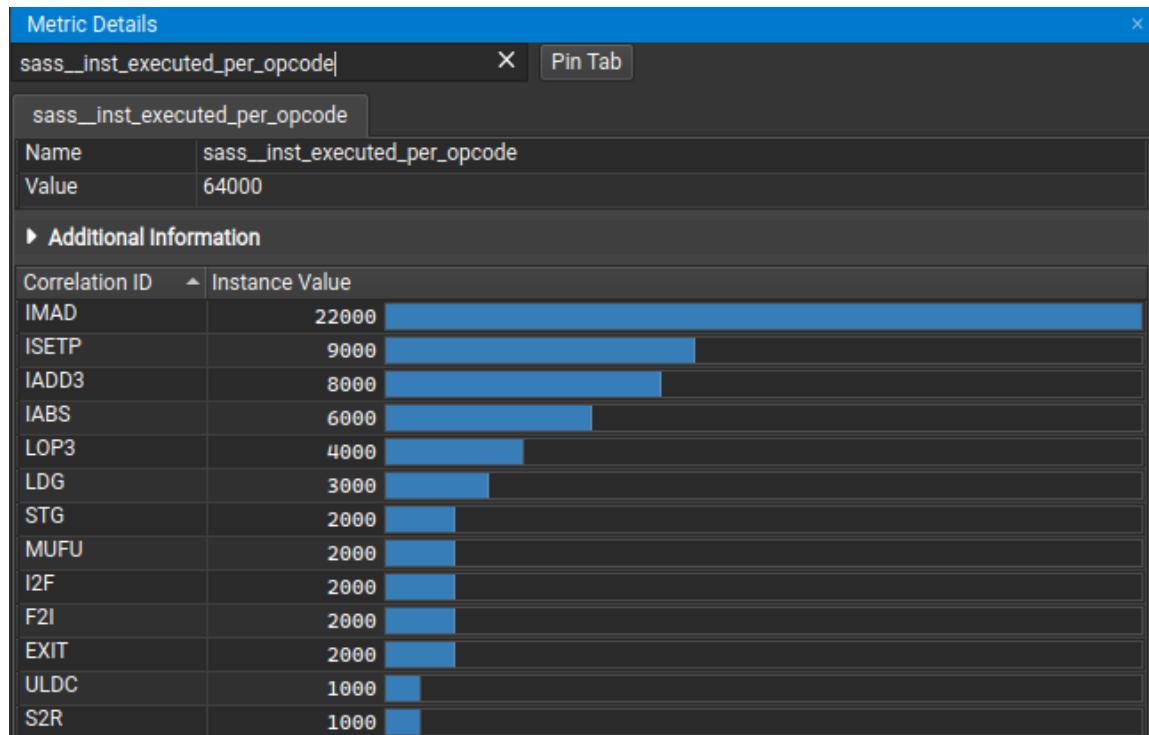
The configured baselines can be saved to a file by clicking on the *Save Baselines* button in the tool bar. By default baseline files use the **.ncu-bln** extension. Baseline files can be opened locally and/or shared with other users.

Baseline information can be loaded by clicking on the *Load Baselines* button in the tool bar. When a baseline file is loaded, currently configured baselines will be replaced. A dialog will be presented to the user to confirm this operation when necessary.

Differences between the current result and the baselines can be visualized with graphical bars for metrics in Details page section headers. Use the *Difference Bars* drop down to select the visualization mode. Bars are extending from left to right and have a fixed maximum.

## 5.4. Metric Details

The *Metric Details* tool window can be opened using the *Metric Details* entry in the *Profile* menu or the respective tool bar button. When a report and the tool window are open, a metric can be selected in the report to display additional information in the tool window. It also contains a search bar to look up metrics in the focused report.



Report metrics can be selected in the [Details Page](#) or the [Raw Page](#). The window will show basic information (name, unit and raw value of the metric) as well as additional information, such as its extended description.

The search bar can be used to open metrics in the focused report. It shows available matches as you type. The entered string must match from the start of the metric name.

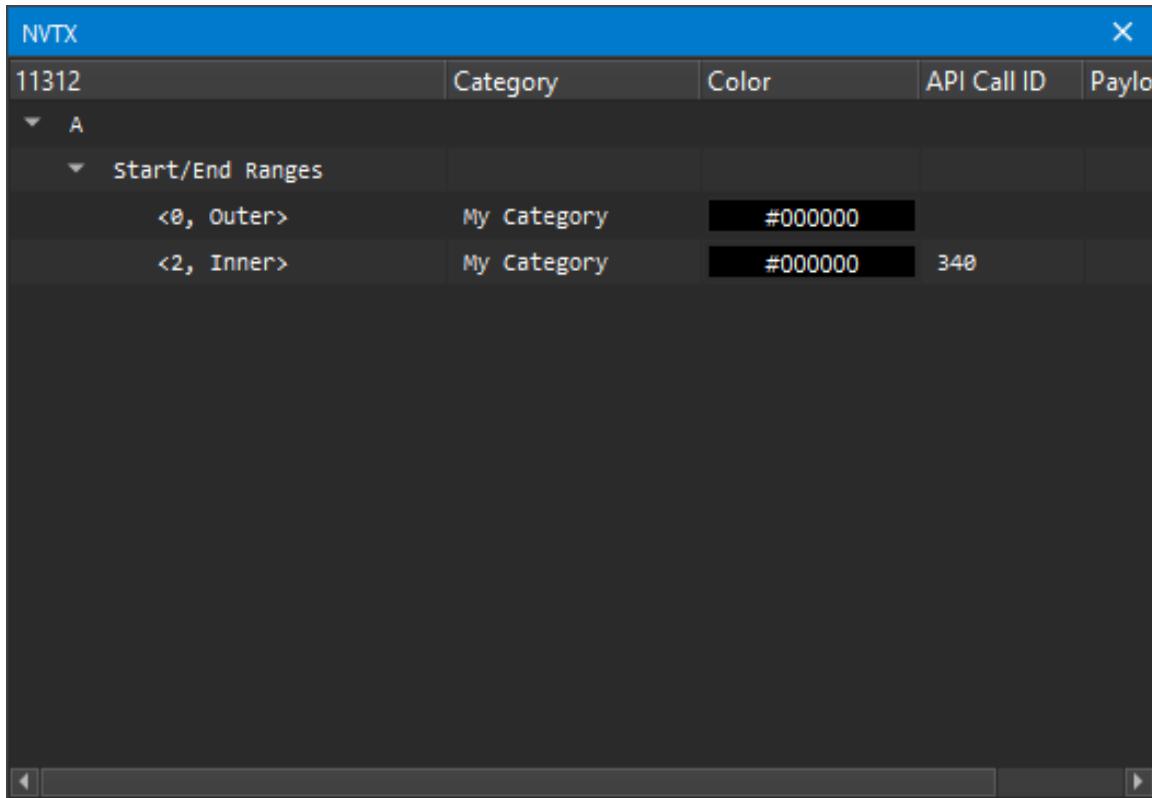
By default, selecting or searching for a new metric updates the current *Default Tab*. You can click the *Pin Tab* button to create a copy of the default tab, unless the same metric is already pinned. This makes it possible to save multiple tabs and quickly switch between them to compare values.

Some metrics contain [Instance Values](#). When available, they are listed in the tool window. Instance values can have a *Correlation ID* that allows correlating the individual value with its associated entity, e.g. a function address or instruction name.

For metrics collected with [PM sampling](#), the correlation ID is the GPU timestamp in nanoseconds. It is shown as an absolute value and relative to the first timestamp for this metric.

## 5.5. NVTX

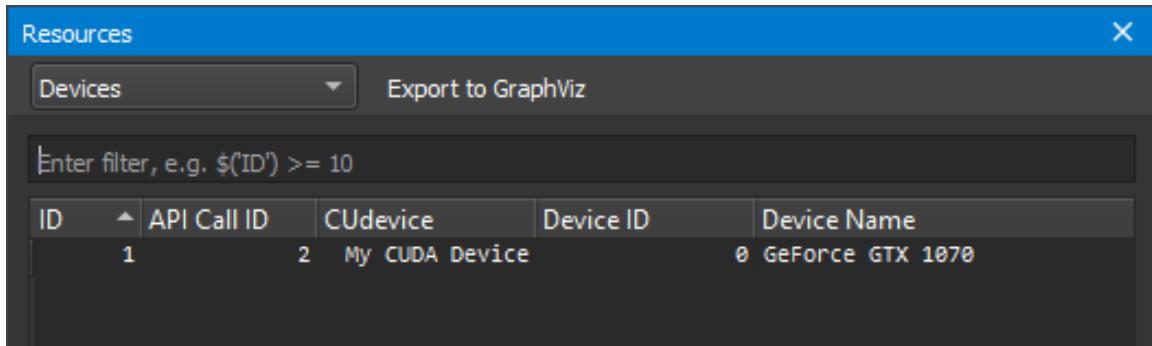
The NVTX window is available when NVIDIA Nsight Compute is connected to a target application. If closed, it can be re-opened using *Debug > NVTX* from the main menu. Whenever the target application is suspended, the window shows the state of all active NVTX domains and ranges in the currently selected thread. Note that NVTX information is only tracked if the launching command line profiler instance was started with `--nvtx` or NVTX was enabled in the NVIDIA Nsight Compute launch dialog.



The screenshot shows the NVTX tool window with the title bar "NVTX". The main area is a table with columns: ID, Category, Color, API Call ID, and Payload. There are two rows under a category "A" which is expanded. The first row is for a range named "<0, Outer>" with "My Category" and "#000000" color. The second row is for a range named "<2, Inner>" with "My Category" and "#000000" color, and a payload value of 340.

ID	Category	Color	API Call ID	Payload
11312				
A				
Start/End Ranges				
<0, Outer>	My Category	#000000		
<2, Inner>	My Category	#000000	340	

Use the *Current Thread* dropdown in the [API Stream](#) window to change the currently selected thread. NVIDIA Nsight Compute supports NVTX named resources, such as threads, CUDA devices, CUDA contexts, etc. If a resource is named using NVTX, the appropriate UI elements will be updated.

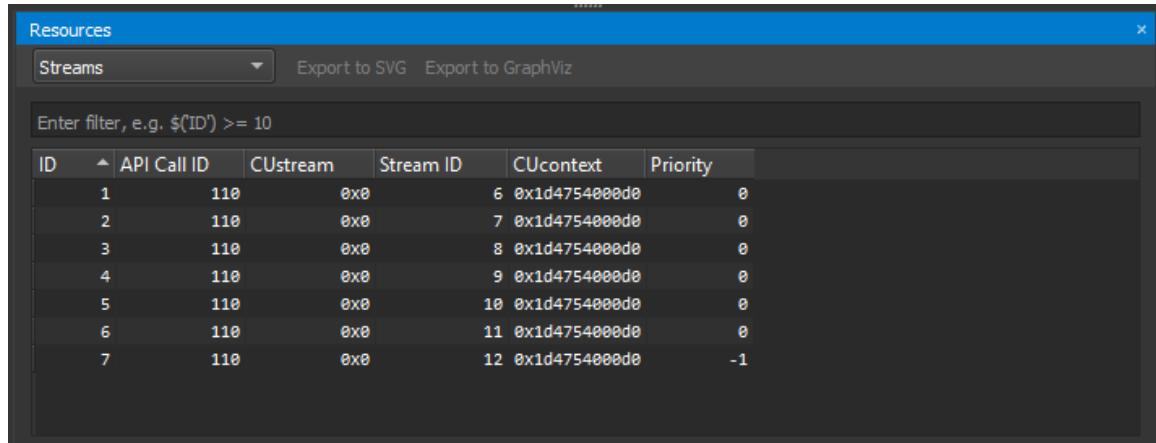


The screenshot shows the Resources tool window with the title bar "Resources". It has a dropdown menu set to "Devices" and a button "Export to GraphViz". Below is a search bar with placeholder text "Enter filter, e.g. \${'ID'} >= 10". A table lists one device entry:

ID	API Call ID	CUdevice	Device ID	Device Name
1	2	My CUDA Device	0	GeForce GTX 1070

## 5.6. Resources

The *Resources* window is available when NVIDIA Nsight Compute is connected to a target application. It shows information about the currently known resources, such as CUDA devices, CUDA streams or kernels. The window is updated every time the target application is suspended. If closed, it can be re-opened using *Debug > Resources* from the main menu.



The screenshot shows the 'Resources' window with the 'Streams' tab selected. At the top, there is a dropdown menu set to 'Streams', and two export options: 'Export to SVG' and 'Export to GraphViz'. Below the header is a search bar with the placeholder 'Enter filter, e.g. \${ID} >= 10'. The main area is a table with the following columns: ID, API Call ID, CUstream, Stream ID, CUcontext, and Priority. The table contains 7 rows of data:

ID	API Call ID	CUstream	Stream ID	CUcontext	Priority
1	110	0x0	6	0x1d4754000d0	0
2	110	0x0	7	0x1d4754000d0	0
3	110	0x0	8	0x1d4754000d0	0
4	110	0x0	9	0x1d4754000d0	0
5	110	0x0	10	0x1d4754000d0	0
6	110	0x0	11	0x1d4754000d0	0
7	110	0x0	12	0x1d4754000d0	-1

Using the dropdown on the top, different views can be selected, where each view is specific to one kind of resource (context, stream, kernel, ...). The *Filter* edit allows you to create filter expressions using the column headers of the currently selected resource.

The resource table shows all information for each resource instance. Each instance has a unique ID, the *API Call ID* when this resource was created, its handle, associated handles, and further parameters. When a resource is destroyed, it is removed from its table.

### 5.6.1. Memory Allocations

When using the asynchronous malloc/free APIs, the resource view for *Memory Allocation* will also include the memory objects created in this manner. These memory objects have a non-zero memory pool handle. The *Mode* column will indicate which code path was taken during the allocation of the corresponding object. The modes are:

- ▶ **REUSE\_STREAM\_SUBPOOL:** The memory object was allocated in memory that was previously freed. The memory was backed by the memory pool set as current for the stream on which the allocation was made.
- ▶ **USE\_EXISTING\_POOL\_MEMORY:** The memory object was allocated in memory that was previously freed. The memory is backed by the default memory pool of the stream on which the allocation was made.
- ▶ **REUSE\_EVENT\_DEPENDENCIES:** The memory object was allocated in memory that was previously freed in another stream of the same context. A stream ordering dependency of the allocating stream on the free action existed. Cuda events and null stream interactions can create the required stream ordered dependencies.

- ▶ REUSE\_OPPORTUNISTIC: The memory object was allocated in memory that was previously freed in another stream of the same context. However, no dependency between the free and allocation existed. This mode requires that the free be already committed at the time the allocation is requested. Changes in execution behavior might result in different modes for multiple runs of the application.
- ▶ REUSE\_INTERNAL\_DEPENDENCIES: The memory object was allocated in memory that was previously freed in another stream of the same context. New internal stream dependencies may have been added in order to establish the stream ordering required to reuse a piece of memory previously released.
- ▶ REQUEST\_NEW\_ALLOCATION: New memory had to be allocated for this memory object as no viable reusable pool memory was found. The allocation performance is comparable to using the non-asynchronous malloc/free APIs.

## 5.6.2. Graphviz DOT and SVG exports

Some of the shown *Resources* can also be exported to *GraphViz DOT* or *SVG* files using the **Export to GraphViz** or **Export to SVG** buttons.

When exporting *OptiX traversable handles*, the traversable graph node types will be encoded using shapes and colors as described in the following table.

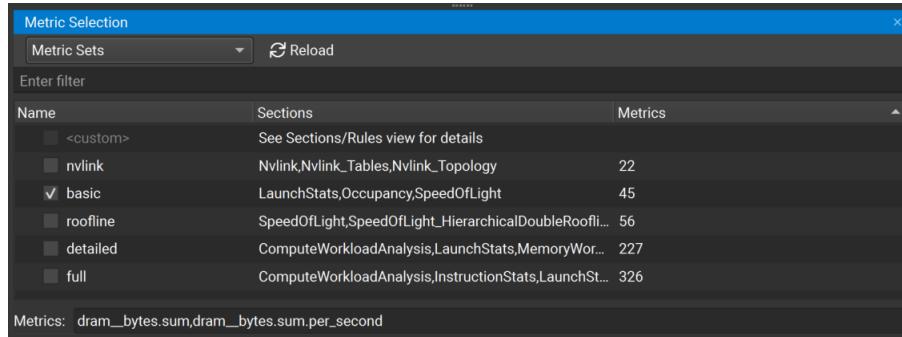
**Table 2** OptiX Traversable Graph Node Types

Node Type	Shape	Color
IAS	Hexagon	#8DD3C7
Triangle GAS	Box	#FFFFB3
AABB GAS	Box	#FCCDE5
Curve GAS	Box	#CCEBC5
Sphere GAS	Box	#BEBADA
Static Transform	Diamond	#FB8072
SRT Transform	Diamond	#FDB462
Matrix Motion Transform	Diamond	#80B1D3
Error	Paralellogram	#D9D9D9

## 5.7. Metric Selection

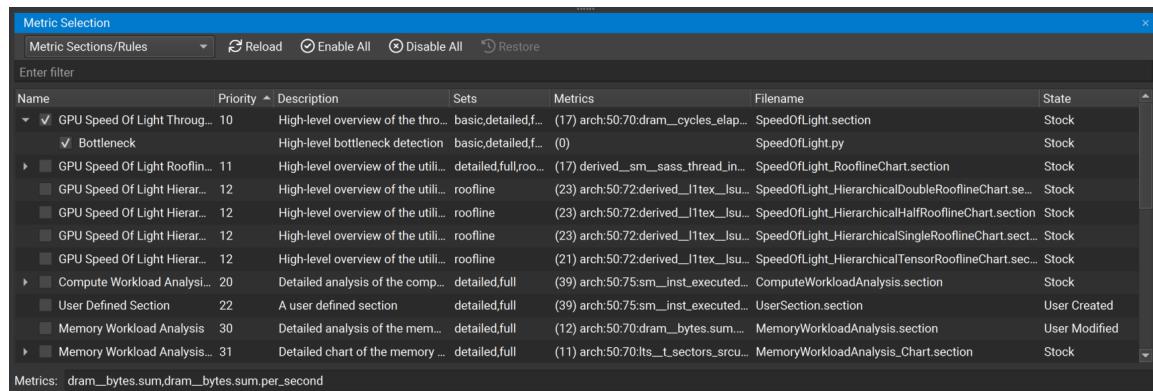
The *Metric Selection* window can be opened from the main menu using *Profile > Metric Selection*. It tracks all metric sets, sections and rules currently loaded in NVIDIA Nsight Compute, independent from a specific connection or report. The directory to load those files from can be configured in the **Profile** options dialog. It is used to inspect available sets, sections and rules, as well as to configure which should be collected, and which rules should be applied. You can also specify a comma separated list of individual metrics, that should be collected. These follow the rules of the NVIDIA Nsight Compute CLI's **--metrics** option. The window has two views, which can be selected using the dropdown in its header.

The **Metric Sets** view shows all available metric sets. Each set is associated with a number of metrics sections. You can choose a set appropriate to the level of detail for which you want to collect performance metrics. Sets which collect more detailed information normally incur higher runtime overhead during profiling.



When enabling a set in this view, the associated metric sections are enabled in the *Metric Sections/Rules* view. When disabling a set in this view, the associated sections in the *Metric Sections/Rules* view are disabled. If no set is enabled, or if sections are manually enabled/disabled in the *Metric Sections/Rules* view, the <custom> entry is marked active to represent that no section set is currently enabled. Note that the *basic* set is enabled by default.

Whenever a kernel is profiled manually, or when auto-profiling is enabled, only sections enabled in the **Metric Sections/Rules** view and individual metrics specified in input box are collected. Similarly, whenever rules are applied, only rules enabled in this view are active.



The enabled states of sections and rules are persisted across NVIDIA Nsight Compute launches. The *Reload* button reloads all sections and rules from disk again. If a new section or rule is found, it will be enabled if possible. If any errors occur while loading a rule, they will be listed in an extra entry with a warning icon and a description of the error.

Use the *Enable All* and *Disable All* checkboxes to enable or disable all sections and rules at once. The Filter text box can be used to filter what is currently shown in the view. It does not alter activation of any entry.

The table shows sections and rules with their activation status, their relationship and further parameters, such as associated metrics or the original file on disk. Rules

associated with a section are shown as children of their section entry. Rules independent of any section are shown under an additional *Independent Rules* entry.

Double-clicking an entry in the table's *Filename* column opens this file as a document. It can be edited and saved directly in NVIDIA Nsight Compute. After editing the file, *Reload* must be selected to apply those changes.

When a section or rule file is modified, the entry in the *State* column will show *User Modified* to reflect that it has been modified from its default state. When a *User Modified* row is selected, the *Restore* button will be enabled. Clicking the *Restore* button will restore the entry to its default state and automatically *Reload* the sections and rules.

Similarly, when a stock section or rule file is removed from the configured *Sections Directory* (specified in the [Profile](#) options dialog), the *State* column will show *User Deleted*. *User Deleted* files can also be restored using the *Restore* button.

Section and rule files that are created by the user (and not shipped with NVIDIA Nsight Compute) will show up as *User Created* in the *state column*.

See the [Sections and Rules](#) for the list of default sections for NVIDIA Nsight Compute.

# Chapter 6. PROFILER REPORT

The profiler report contains all the information collected during profiling for each kernel launch. In the user interface, it consists of a header with general information, as well as controls to switch between report pages or individual collected launches.

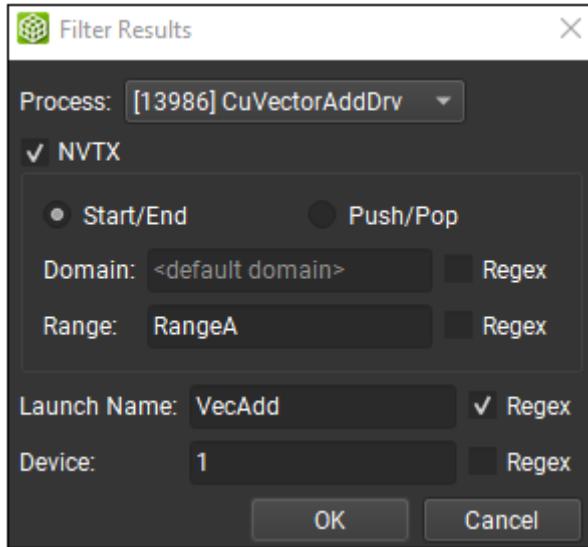
## 6.1. Header

The *Page* dropdown can be used to switch between the available report pages, which are explained in detail in the [next section](#).



The *Launch* dropdown can be used to switch between all collected kernel launches. The information displayed in each page commonly represents the selected launch instance. On some pages (e.g. *Raw*), information for all launches is shown and the selected instance is highlighted. You can type in this dropdown to quickly filter and find a kernel launch.

The *Apply Filters* button opens the filter dialog. You can use more than one filter to narrow down your results. On the filter dialog, enter your filter parameters and press OK button. The *Launch* dropdown, [Summary Page](#) table, and [Raw Page](#) table will be filtered accordingly. Select the arrow dropdown to access the *Clear Filters* button, which removes all filters.



The *Add Baseline* button promotes the current result in focus to become the baseline of all other results from this report and any other report opened in the same instance of NVIDIA Nsight Compute. Select the arrow dropdown to access the *Clear Baselines* button, which removes all currently active baselines.

The *Apply Rules* button applies all rules available for this report. If rules had been applied previously, those results will be replaced. By default, rules are applied immediately once the kernel launch has been profiled. This can be changed in the options under *Tools > Options > Profile > Report UI > Apply Applicable Rules Automatically*.

A button on the right-hand side offers multiple operations that may be performed on the page. Available operations include:

- ▶ **Copy as Image** - Copies the contents of the page to the clipboard as an image.
- ▶ **Save as Image** - Saves the contents of the page to a file as an image.
- ▶ **Save as PDF** - Saves the contents of the page to a file as a PDF.
- ▶ **Export to CSV** - Exports the contents of page to CSV format.
- ▶ **Reset to Default** - Resets the page to a default state by removing any persisted settings.

Note that not all functions are available on all pages.

Information about the selected kernel is shown as *Current*. [+] and [-] buttons can be used to show or hide the section body content. The visibility of the output of the rules can be toggled with the *r* button. The info toggle button *i* changes the section description's visibility.

## 6.2. Report Pages

Use the *Page* dropdown in the header to switch between the report pages.

By default, when opening a report with a single profile result, the [Details Page](#) is shown. When opening a report with multiple results, the [Summary Page](#) is selected instead. You can change the default report page in the [Profile](#) options.

## 6.2.1. Session Page

This *Session* page contains basic information about the report and the machine, as well as device attributes of all devices for which launches were profiled. When switching between launch instances, the respective device attributes are highlighted.

## 6.2.2. Summary Page

The *Summary* page shows a table of all collected results in the report, as well as a list of the most important rule outputs (*Prioritized Rules*) which are ordered by the estimated speedup that could potentially be obtained by following their guidance. *Prioritized Rules* are shown by default and can be toggled with the [R] button on the upper right of the page.

The screenshot shows the NVIDIA Nsight Compute interface with the 'Summary' page selected. At the top, there's a navigation bar with 'File', 'Connection', 'Debug', 'Profile', 'Tools', 'Window', and 'Help'. Below the navigation bar, there are buttons for 'Connect', 'Disconnect', 'Terminate', 'Profile Kernel', and various metrics like 'Time', 'Cycles', 'Regs', 'GPU', 'SM Frequency', 'CC', and 'Process'. A 'Baselines' section is also present. The main area displays a table titled 'Summary' with columns: ID, Estimated Speedup, Function Name, Demangled Name, Duration, Runtime Improvement, Compute Throughput, Memory Throughput, # Registers, and Grid Size. The table lists 12 rows of data, each corresponding to a different kernel or function. Below the table, a note states: 'The following performance optimization opportunities were discovered for this result. Follow the rule links to see more context on the Details page. Note: Speedup estimations are experimental and might overestimate optimization potential.' Underneath the table, three sections provide detailed analysis: 'Shared Store Bank Conflicts', 'Uncalesced Shared Accesses', and 'Shared Load Bank Conflicts', each with specific metrics and recommendations.

ID	Estimated Speedup	Function Name	Demangled Name	Duration	Runtime Improvement (4.46311e+06)	Compute Throughput	Memory Throughput	# Registers	Grid Size
0	42.38	mergeSortSharedK...	void mergeSortSh...	1.67	0.71	84.56	84.56	17	409
1	93.13	generateSampleRa...	void generateSamp...	0.08	0.08	3.99	41.45	18	6
2	68.35	mergeRanksAndInd...	mergeRanksAndInd...	0.01	0.00	19.53	13.46	16	6
3	68.92	mergeRanksAndInd...	mergeRanksAndInd...	0.01	0.00	19.63	13.53	16	6
4	37.78	mergeElementaryin...	void mergeElement...	0.57	0.21	87.06	87.06	22	3276
5	93.05	generateSampleRa...	void generateSamp...	0.08	0.07	4.84	43.70	18	6
6	68.35	mergeRanksAndInd...	mergeRanksAndInd...	0.01	0.01	20.43	11.96	16	6
7	68.46	mergeRanksAndInd...	mergeRanksAndInd...	0.01	0.01	20.95	12.26	16	6
8	38.48	mergeElementaryin...	void mergeElement...	0.57	0.22	86.87	86.87	22	3276
9	95.26	generateSampleRa...	void generateSamp...	0.08	0.08	4.69	46.18	18	6
10	68.55	mergeRanksAndInd...	mergeRanksAndInd...	0.01	0.01	21.63	11.00	16	6
11	67.81	mergeRanksAndInd...	mergeRanksAndInd...	0.01	0.01	21.66	11.02	16	6
12	38.78	mergeElementaryin...	void mergeElement...	0.57	0.22	86.98	86.98	22	3276

The following performance optimization opportunities were discovered for this result. Follow the rule links to see more context on the Details page.  
Note: Speedup estimations are experimental and might overestimate optimization potential.

**Shared Store Bank Conflicts**  
Est. Speedup: 42.38%  
The memory access pattern for shared stores might not be optimal and causes on average a 2.0-way bank conflict across all 29873508 shared store requests. This results in 2873508 bank conflicts, which represent 49.91% of the overall 5757731 wavefronts for shared stores. Check the [Source Counter](#) section for uncoalesced shared stores.

**Uncalesced Shared Accesses**  
Est. Speedup: 37.66%  
This kernel has uncoalesced shared accesses resulting in a total of 7891571 excessive wavefronts (38% of the total 20867699 wavefronts). Check the L1 Wavefronts Shared Excessive table for the primary source locations. The [CUDA Best Practices Guide](#) has an example on optimizing shared memory accesses.

**Shared Load Bank Conflicts**  
Est. Speedup: 28.49%  
The memory access pattern for shared loads might not be optimal and causes on average a 1.5-way bank conflict across all 10092544 shared load requests. This results in 5103707 bank conflicts, which represent 33.55% of the overall 15212319 wavefronts for shared loads. Check the [Source Counter](#) section for uncoalesced shared loads.

The *Summary Table* gives you a quick comparison overview across all profiled workloads. It contains a number of important, pre-selected metrics which can be customized as explained below. Its columns can be sorted by clicking the column header. You can transpose the table with the *Transpose* button. Aggregate of all results per each counter metric is shown in the table header along with the column name. You

can change the aggregated values by selecting the desired results for multiple metrics simultaneously. When selecting any entry by single-click, a list of its *Prioritized Rules* will be shown below the table. Double-click any entry to make the result the currently active one and switch to the [Details Page](#) page to inspect its performance data.

ID	Estimated Speedup	Function Name	Demangled Name	Duration	Runtime Improvement (4.46311e+06)	Compute Throughput	Memory Throughput	# Registers	Grid Size
0	42.38	mergeSortSharedK...	void mergeSortSh...	1.67	0.71	84.56	84.56	17	495
32	39.27	mergeElementaryIn...	void mergeElement...	0.58	0.23	86.94	86.94	22	3276
36	39.28	mergeElementaryIn...	void mergeElement...	0.57	0.23	86.96	86.96	22	3276
28	39.28	mergeElementaryIn...	void mergeElement...	0.57	0.23	86.95	86.95	22	3276

You can configure the list of metrics included in this table in the [Profile](#) options dialog.

If a metric has multiple instance values, the number of instances is shown after its standard value. A metric with ten instance values could for example look like this:

**35.48 {10}**. In the [Profile](#) options dialog, you can select that all instance values should be shown individually. You can also inspect the instances values of a metric result in the [Metric Details](#) tool window.

In addition to metrics, you can also configure the table to include any of the following properties:

## Properties

property__api_call_id	ID of the API call associated with this profile result.
property__block_size	Block Size.
property__creation_time	Local collection time.
property__demangled_name	Kernel demangled name.
property__device_name	GPU device name.
property__estimated_speedup	Maximal relative speedup achievable for this profile result as estimated by the guided analysis rules.
property__function_name	Kernel function name or range name.
property__grid_dimensions	Grid Dimensions.
property__grid_offset	Grid Offset.
property__grid_size	Grid Size.
property__issues_detected	Number of issues detected by guided analysis rules for this profile result.
property__kernel_id	Kernel ID.
property__mangled_name	Kernel mangled name.
property__process_name	Process name.
property__runtime_improvement	Runtime improvement corresponding to the estimated speedup.

property__series_id	ID of the profile series.
property__series_parameters	Profile series parameters.
property__thread_id	CPU thread ID.

For [Range Replay](#) reports, a smaller set of columns is shown by default, as not all apply to such results.

For the currently selected metric result the *Prioritized Rules* show the most impactful rule results with respect to the estimated potential speedup. Clicking on any of the rule names on the left allows you to easily navigate to the containing section on the details page. With the downward-facing arrow on the right a table with the relevant *key performance indicators* can be toggled. This table contains the metrics which should be tracked when optimizing performance according to the rule guidance.

The following performance optimization opportunities were discovered for this result. Follow the rule links to see more context on the Details page.  
Note: Speedup estimations are experimental and might overestimate optimization potential.

Shared Store Bank Conflicts	Est. Speedup: 49.63%	This memory access pattern for shared stores might not be optimal and causes on average a 2.0 - way bank conflict across all 2883584 shared store requests. This results in 2862630 bank conflicts, which represent 49.83% of the overall 5744585 wavefronts for shared stores. Check the <a href="#">Source Counter</a> section for uncoalesced shared stores.
Uncoaesced Shared Accesses	Est. Speedup: 37.66%	This kernel has uncoalesced shared accesses resulting in a total of 7891571 excessive wavefronts (38% of the total 20867699 wavefronts). Check the L1 Wavefronts Shared Excessive table for the primary source locations. The <a href="#">CUDA Best Practices Guide</a> has an example on optimizing shared memory accesses.

The following table lists the metrics that are key performance indicators:

Metric Name	Value	Guidance
derived_memory_l1_wavefronts_shared_excessive	7.89157e+06	Reduce the number of excessive wavefronts in L1TEX

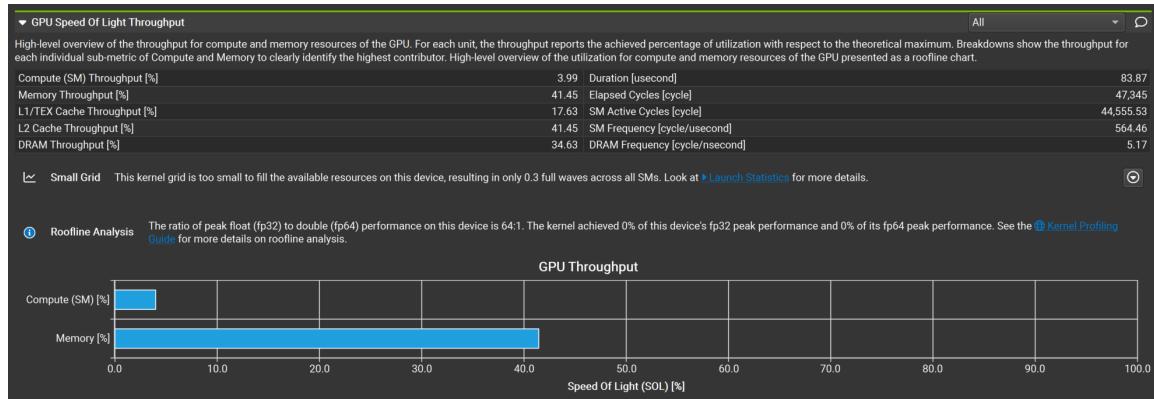
## 6.2.3. Details Page

### Overview

The *Details* page is the main page for all metric data collected during a kernel launch. The page is split into individual sections. Each section consists of a header table and an optional body that can be expanded. The sections are completely user defined and can be changed easily by updating their respective files. For more information on customizing sections, see the [Customization Guide](#). For a list of sections shipped with NVIDIA Nsight Compute, see [Sections and Rules](#).

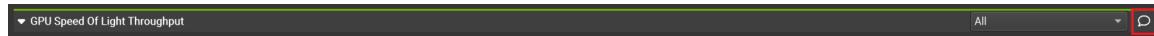
By default, once a new profile result is collected, all applicable rules are applied. Any rule results will be shown as *Recommendations* on this page. Most rule results will contain an optimization advice along with an estimate of the improvement that could be achieved when successfully implementing this advice. Other rule results will be purely informative or have a warning icon to indicate a problem that occurred during execution (e.g., an optional metric that could not be collected). Results with error icons typically indicate an error while applying the rule.

Estimates of potential improvement are shown below the rule result's name and exist in two types. *Global estimates* ("Est. Speedup") are an approximation of the decrease in workload runtime, whereas *local estimates* ("Est. Local Speedup") are an approximation of the increase in efficiency of the hardware utilization of the particular performance problem the rule addresses.

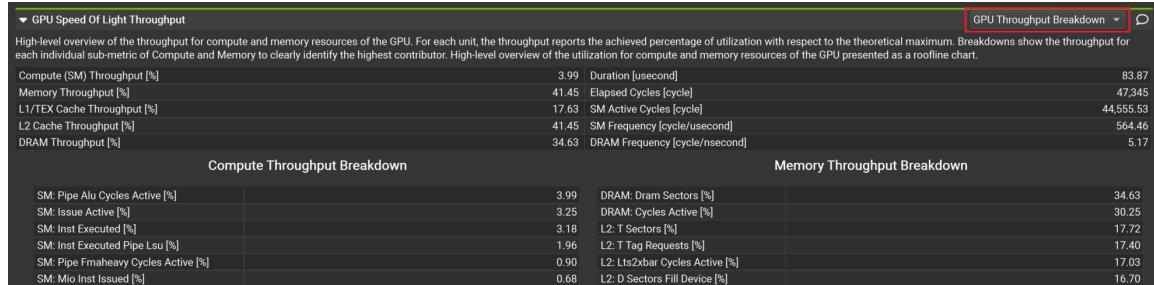


If a rule result references another report section, it will appear as a link in the recommendation. Select the link to scroll to the respective section. If the section was not collected in the same profile result, enable it in the [Metric Selection](#) tool window.

You can add or edit comments in each section of the *Details* view by clicking on the comment button (speech bubble). The comment icon will be highlighted in sections that contain a comment. Comments are persisted in the report and are summarized in the [Comments Page](#).



Besides their header, sections typically have one or more *bodies* with additional charts or tables. Click the triangle *Expander* icon in the top-left corner of each section to show or hide those. If a section has multiple bodies, a dropdown in their top-right corner allows you to switch between them.



## Memory

If enabled, the *Memory Workload Analysis* section contains a Memory chart that visualizes data transfers, cache hit rates, instructions and memory requests. More information on how to use and read this chart can be found in the [Kernel Profiling Guide](#).

## Occupancy

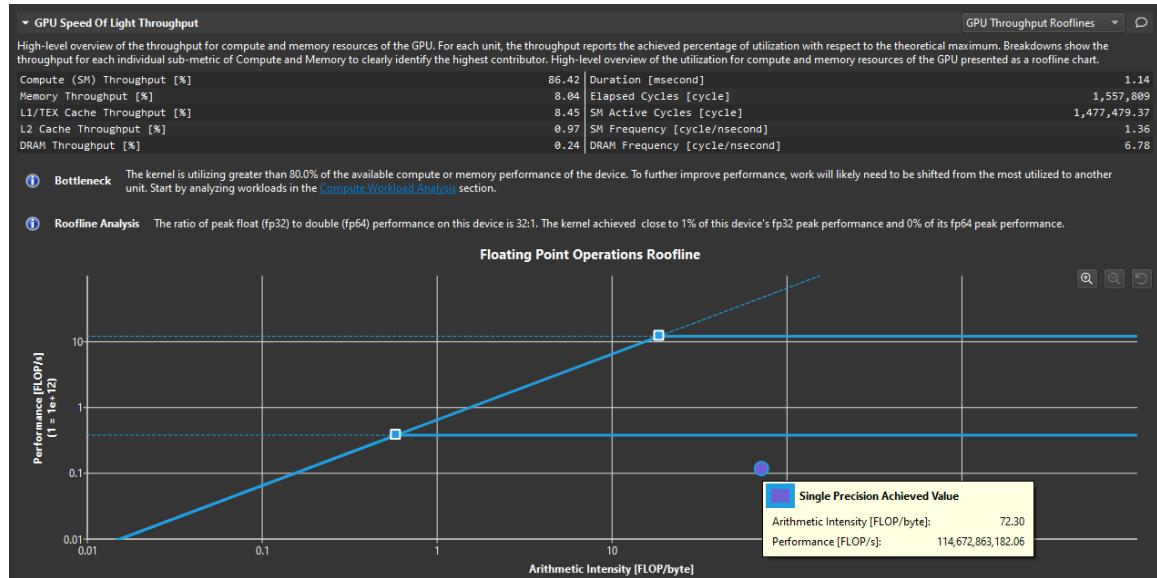
You can open the [Occupancy Calculator](#) by clicking on the calculator button in the report header or in the header of the *Occupancy Section*.

## Range Replay

Note that for **Range Replay** results some UI elements, analysis rules, metrics or section body items such as charts or tables might not be available, as they only apply to kernel launch-based results. The filters can be checked in the corresponding section files.

## Rooflines

If enabled, the *GPU Speed Of Light Roofline Chart* section contains a Roofline chart that is particularly helpful for visualizing kernel performance at a glance. (To enable roofline charts in the report, ensure that the section is enabled when profiling.) More information on how to use and read this chart can be found in [Roofline Charts](#). NVIDIA Nsight Compute ships with several different definitions for roofline charts, including hierarchical rooflines. These additional rooflines are defined in different section files. While not part of the *full* section set, a new section set called *roofline* was added to collect and show all rooflines in one report. The idea of hierarchical rooflines is that they define multiple ceilings that represent the limiters of a hardware hierarchy. For example, a hierarchical roofline focusing on the memory hierarchy could have ceilings for the throughputs of the L1 cache, L2 cache and device memory. If the achieved performance of a kernel is limited by one of the ceilings of a hierarchical roofline, it can indicate that the corresponding unit of the hierarchy is a potential bottleneck.



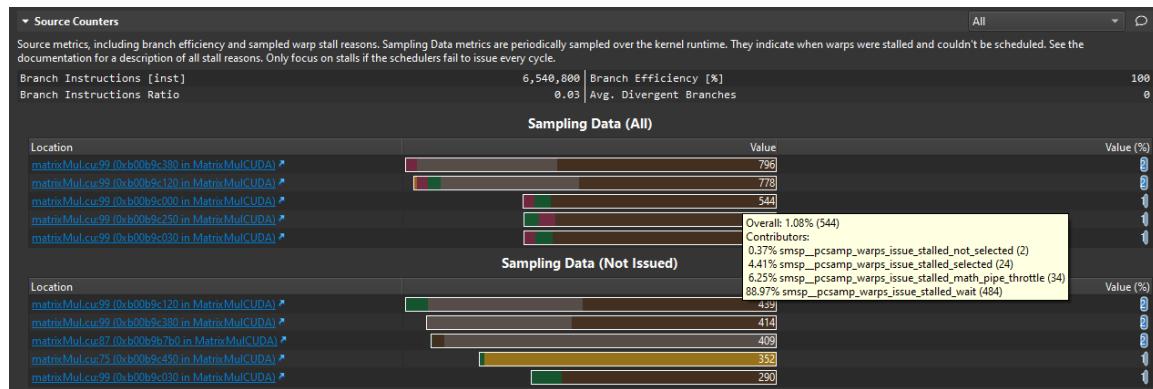
The roofline chart can be zoomed and panned for more effective data analysis, using the controls in the table below.

Table 3 Roofline Chart Zoom and Pan Controls

Zoom In	Zoom Out	Zoom Reset	Pan
<ul style="list-style-type: none"> <li>▶ Click the Zoom In button in the top right corner of the chart.</li> <li>▶ Click the left mouse button and drag to create a rectangle that bounds the area of interest.</li> <li>▶ Press the plus (+) key.</li> <li>▶ Use Ctrl + MouseWheel (Windows and Linux only)</li> </ul>	<ul style="list-style-type: none"> <li>▶ Click the Zoom Out button in the top right corner of the chart.</li> <li>▶ Click the right mouse button.</li> <li>▶ Press the plus (-) key.</li> <li>▶ Use Ctrl + MouseWheel (Windows and Linux only)</li> </ul>	<ul style="list-style-type: none"> <li>▶ Click the Zoom Reset button in the top right corner of the chart.</li> <li>▶ Press the Escape (Esc) key.</li> </ul>	<ul style="list-style-type: none"> <li>▶ Use Ctrl (Command on Mac) + LeftMouseButton to grab the chart, then move the mouse.</li> <li>▶ Use the cursor keys.</li> </ul>

## Source

Sections such as *Source Counters* can contain source hot spot tables. These tables indicate the N highest or lowest values of one or more metrics in your kernel source code. Select the location links to navigate directly to this location in the [Source Page](#). Hover the mouse over a value to see which metrics contribute to it.



## Timelines

When collecting metrics with [PM sampling](#), they can be viewed in a *timeline*. The timeline shows metrics selected in the respective section file or on the command line with their labels/names and their values over time.

Different metrics may be collected in different passes (replays) of the workload, as only a limited number of them can be sampled in the same pass. Context switch trace is used to filter the collected data to only include samples from the profiled contexts and to align it in the timeline.

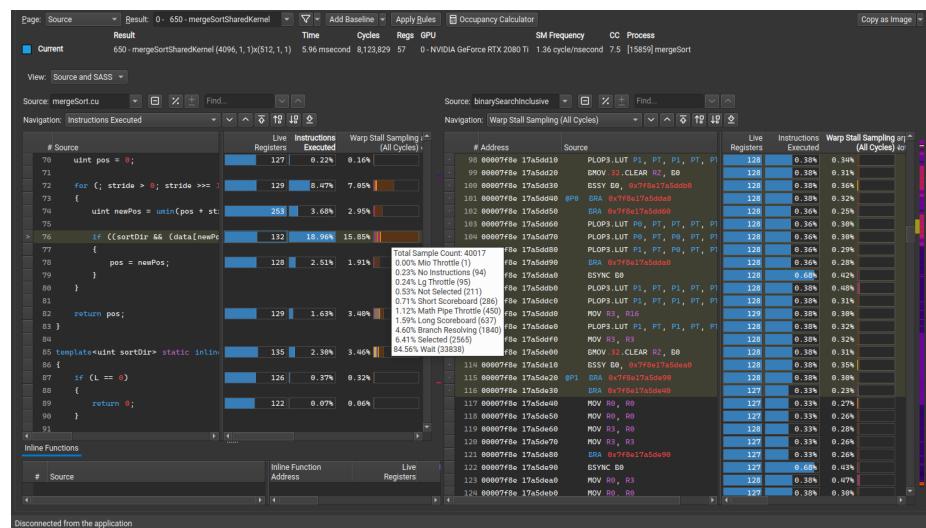
You can hover the mouse over a metric row label to see further information on the metrics in the row. Hovering over a sample on the timeline shows the metric values at that timestamp within the current row. With the [Metric Details](#) tool window open, click to select a value on the timeline and show the metric and all its raw timestamps (absolute and relative) correlated values in the tool window.

You can also use the [Metric Details](#) tool window to inspect profiler metrics generated during PM sampling. These provide information about the used sampling intervals, buffer sizes, dropped samples and other properties for each collection pass. A detailed list can be found in the [metrics reference](#).

The timeline has a context menu for further actions regarding copying content and zooming. In addition, the *Toggle Context Switch Filter* option can be used to enable or disable the filtering of the timeline data with [context switch](#) information, if it is available. When the context switch filter is enabled (the default), samples from each pass group are only shown for the active contexts. When the context switch filter is disabled, the raw collected sampling data is shown along with a separate row for each pass group's context switch trace. When the context menu option is not available, the report does not include context switch trace data.

## 6.2.4. Source Page

The *Source* page correlates assembly (SASS) with high-level code such as CUDA-C or PTX. In addition, it displays instruction-correlated metrics to help pinpoint performance problems in your code.



The page can be switched between different *Views* to focus on a specific source layer or see two layers side-by-side. This includes SASS, PTX and Source (CUDA-C, Fortran, Python, ...), as well as their combinations. Which options are available depends on the source information embedded into the executable.

The high-level Source (CUDA-C) view is available if the application was built with the `-lineinfo` or `--generate-line-info` nvcc flag to correlate SASS and source. When using separate linking at the ELF level, there is no PTX available in the ELF that would correspond to the final SASS. As such, NVIDIA Nsight Compute does not show any PTX even though it would be available statically in the executable and could be shown with `cuobjdump -all -lptx`. However, this is a pre-linked version of the PTX and cannot be reliably used for correlation.

The code in the different *Views* can also contain warnings, errors or just notifications that are displayed as *Source Markers* in the left header, as shown below. These can be generated from multiple systems, but as of now only NvRules are supported.

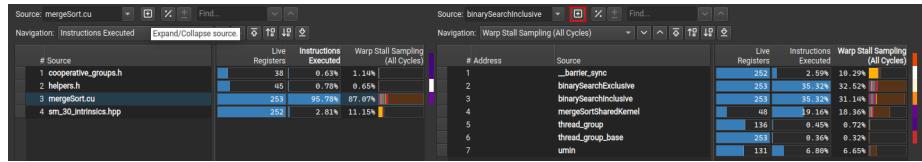
	9 00007f12 b327bb80	IMAD.WIDE R2, R10, R11, c[0x0][0x168]
⚠	10 00007f12 b327bb90	LDG.E.64 R4, [R2.64]
⚠	11 00007f12 b327bba0	LDG.E.64 R6, [R2.64+0x8]
⚠	12 00007f12 b327bbb0	LDG.E.64 R8, [R2.64+0x10]
	13 00007f12 b327bbc0	IMAD.WIDE R10, R10, R11, c[0x0][0x178]
	14 00007f12 b327bbd0	DADD R4, R4, c[0x0][0x170]
	15 00007f12 b327bbe0	DADD R6, R6, c[0x0][0x170]
⚠	16 00007f12 b327bbf0	STG.E.64 [R10.64], R4
	17 00007f12 b327bc00	DADD R8, R8, c[0x0][0x170]
⚠	18 00007f12 b327bc10	STG.E.64 [R10.64+0x8], R6
⚠	19 00007f12 b327bc20	STG.E.64 [R10.64+0x10], R8
	20 00007f12 b327bc30	EXIT

### 6.2.4.1. Navigation

The *View* dropdown can be used to select different code (correlation) options: SASS, PTX and Source (CUDA-C, Fortran, Python, ...).

In side-by-side views, when selecting a line in the left-hand- or right-hand-side, any correlated lines in the opposite view are highlighted. However, when the *Show Single File For Multi-File Sources* option is set to *Yes*, the target file or source object must already be selected in the respective view for those correlated lines to be shown.

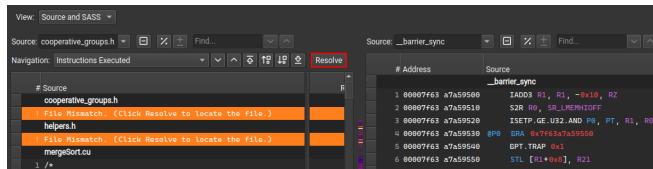
The *Source* drop down allows you to switch between the files or functions that provide the content in the view. When a different source entry is selected, the view scrolls to the start of this file or function. If a view contains multiple source files or functions, [+] and [-] buttons are shown. These can be used to expand or collapse the view, thereby showing or hiding the file or function content except for its header. If collapsed, all *metrics* are shown aggregated to provide a quick overview.



You can use the *Find* (source code) line edit to search the *Source* column of each view. Enter the text to search and use the associated buttons to find the next or previous occurrence in this column. While the line edit is selected, you can also use the *Enter* or *Shift+Enter* keys to search for the next or previous occurrence, respectively.

The SASS view is filtered to only show functions that were executed in the launch. You can toggle the **Show Only Executed Functions** option to change this, but performance of this page may be negatively affected for large binaries. It is possible that some SASS instructions are shown as *N/A*. Those instructions are not currently exposed publicly.

Only filenames are shown in the view, together with a *File Not Found* error, if the source files cannot be found in their original location. This can occur, for example, if the report was moved to a different system. Select a filename and click the *Resolve* button above to specify where this source can be found on the local filesystem. However, the view always shows the source files if the **import source** option was selected during profiling, and the files were available at that time. If a file is found in its original or any source lookup location, but its attributes don't match, a *File Mismatch* error is shown. See the **Source Lookup** options for changing file lookup behavior.

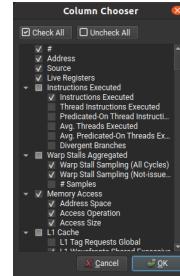


If the report was collected using remote profiling, and automatic resolution of remote files is enabled in the **Profile** options, NVIDIA Nsight Compute will attempt to load the source from the remote target. If the connection credentials are not yet available in the current NVIDIA Nsight Compute instance, they are prompted in a dialog. Loading from a remote target is currently only available for Linux x86\_64 targets and Linux and Windows hosts.

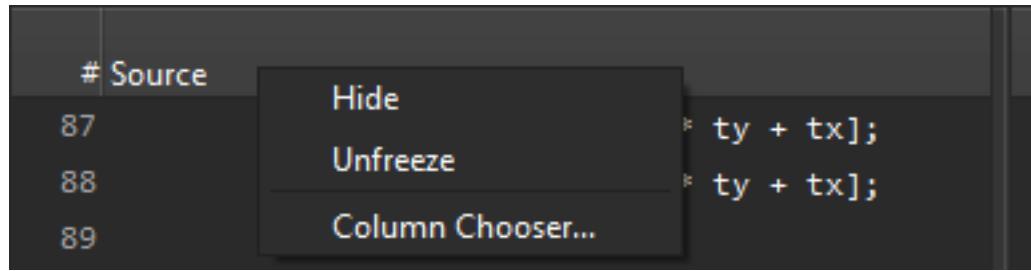
### 6.2.4.2. Metrics

#### Metrics Correlation

The page is most useful when inspecting performance information and metrics correlated with your code. Metrics are shown in columns, which can be enabled or disabled using the *Column Chooser* accessible using the column header right click menu.



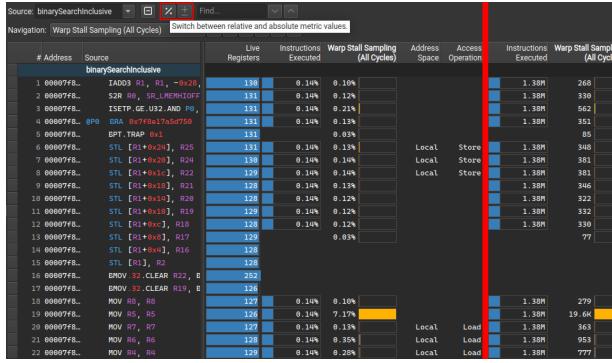
To not move out of view when scrolling horizontally, columns can be fixed. By default, the *Source* column is fixed to the left, enabling easy inspection of all metrics correlated to a source line. To change fixing of columns, right click the column header and select *Freeze* or *Unfreeze*, respectively.



The heatmap on the right-hand side of each view can be used to quickly identify locations with high metric values of the currently selected metric in the dropdown. The heatmap uses a black-body radiation color scale where black denotes the lowest mapped value and white the highest, respectively. The current scale is shown when clicking and holding the heatmap with the right mouse button.



By default, applicable metrics are shown as percentage values relative to their sum across the launch. A bar is filling from left to right to indicate the value at a specific source location relative to this metric's maximum within the launch. The [%] and [+/-] buttons can be used to switch the display from relative to absolute and from abbreviated absolute to full-precision absolute, respectively. For relative values and bars, the [circle/pie] button can be used to switch the display between relative to global (launch) and relative to local (function/file) scope. This button is disabled when the view is collapsed, as percentages are always relative to the global launch scope in this case.



## Pre-Defined Source Metrics

- ▶ **Live Registers**

Number of registers that need to be kept valid by the compiler. A high value indicates that many registers are required at this code location, potentially increasing the register pressure and the maximum number of register required by the kernel.

The total number of registers reported as `launch_registers_per_thread` may be significantly higher than the maximum live registers. The compiler may need to allocate specific registers that can creates holes in the allocation, thereby affecting `launch_registers_per_thread`, even if the maximum live registers is smaller. This may happen due to ABI restrictions, or restrictions enforced by particular hardware instructions. The compiler may not have a complete picture of which registers may be used in either callee or caller and has to obey ABI conventions, thereby allocating different registers even if some register could have theoretically been re-used.

- ▶ **Warp Stall Sampling (All Samples)<sup>1</sup>**

The number of samples from the [Statistical Sampler](#) at this program location.

- ▶ **Warp Stall Sampling (Not-issued Samples)<sup>2</sup>**

The number of samples from the [Statistical Sampler](#) at this program location on cycles the warp scheduler issued no instructions. Note that (*Not Issued*) samples may be taken on a different profiling pass than (*All*) samples mentioned above, so their values do not strictly correlate.

This metric is only available on devices with compute capability 7.0 or higher.

- ▶ **Instructions Executed**

Number of times the source (instruction) was executed per individual warp, independent of the number of participating threads within each warp.

- ▶ **Thread Instructions Executed**

<sup>1</sup> This metric was previously called *Sampling Data (All)*.

<sup>2</sup> This metric was previously called *Sampling Data (Not Issued)*.

Number of times the source (instruction) was executed by any thread, regardless of predicate presence or evaluation.

- ▶ **Predicated-On Thread Instructions Executed**

Number of times the source (instruction) was executed by any active, predicated-on thread. For instructions that are executed unconditionally (i.e. without predicate), this is the number of active threads in the warp, multiplied with the respective *Instructions Executed* value.

- ▶ **Avg. Threads Executed**

Average number of thread-level executed instructions per warp, regardless of their predicate.

- ▶ **Avg. Predicated-On Threads Executed**

Average number of predicated-on thread-level executed instructions per warp.

- ▶ **Divergent Branches**

Number of divergent branch targets, including fallthrough. Incremented only when there are two or more active threads with divergent targets. Divergent branches can lead to warp stalls due to resolving the branch or instruction cache misses.

- ▶ **Information on Memory Operations**

Label	Name	Description
Address Space	memory_type	The accessed address space (global/local/shared).
Access Operation	memory_access_type	The type of memory access (e.g. load or store).
Access Size	memory_access_size_type	The size of the memory access, in bits.
L1 Tag Requests Global	memory_l1_tag_requests_global	Number of L1 tag requests generated by global memory instructions.
L1 Conflicts Shared N-Way	derived_memory_l1_conflicts_shared_n-way	Averaged N-way conflict in L1 per shared memory instruction. A 1-way access has no conflicts and resolves in a single pass. Note: This

		is a derived metric which can not be collected directly.
L1 Wavefronts Shared Excessive	derived_memory_l1_wavefronts_shared_excessive	Excessive number of wavefronts in L1 from shared memory instructions, because not all not predicated-off threads performed the operation. Note: This is a derived metric which can not be collected directly.
L1 Wavefronts Shared	memory_l1_wavefronts_shared	Number of wavefronts in L1 from shared memory instructions.
L1 Wavefronts Shared Ideal	memory_l1_wavefronts_shaded	Ideal number of wavefronts in L1 from shared memory instructions, assuming each not predicated-off thread performed the operation.
L2 Theoretical Sectors Global Excessive	derived_memory_l2_theoretical_global_excessive	Excessive global excessive number of sectors requested in L2 from global memory instructions, because not all not predicated-off threads performed the operation. Note: This is a derived metric which can not be collected directly.
L2 Theoretical Sectors Global	memory_l2_theoretical_global	Ideal global number of sectors requested in L2 from global memory instructions.

L2 Theoretical Sectors Global Ideal	memory_l2_theoretical_sectors_global_ideal	<i>Ideal</i> number of sectors requested in L2 from global memory instructions, assuming each not predicated-off thread performed the operation.
L2 Theoretical Sectors Local	memory_l2_theoretical_sectors_local_ideal	<i>Ideal</i> number of sectors requested in L2 from local memory instructions.

All *L1/L2 Sectors/Wavefronts/Requests* metrics give the number of achieved (actually required), ideal, and excessive (achieved - ideal) sectors/wavefronts/requests.

*Ideal* metrics indicate the number that would be needed, given each not predicated-off thread performed the operation of given width. *Excessive* metrics indicate the required surplus over the ideal case. Reducing divergence between threads can reduce the excess amount and result in less work for the respective HW units.

Several of the above metrics on memory operations were renamed in version 2021.2 as follows:

Old name	New name
memory_l2_sectors_global	memory_l2_theoretical_sectors_global
memory_l2_sectors_global_ideal	memory_l2_theoretical_sectors_global_ideal
memory_l2_sectors_local	memory_l2_theoretical_sectors_local
memory_l1_sectors_global	memory_l1_tag_requests_global
memory_l1_sectors_shared	memory_l1_wavefronts_shared
memory_l1_sectors_shared_ideal	memory_l1_wavefronts_shared_ideal

- ▶ **L2 Explicit Evict Policy Metrics**

Starting with the NVIDIA Ampere architecture the eviction policy of the L2 cache can be tuned to match the kernel's access pattern. The eviction policy can be either set implicitly for a memory window (for more details see [CUaccessProperty](#)) or set explicitly per executed memory instruction. If set explicitly, the desired eviction behavior for the cases of an L2 cache hit or miss are passed as input to the instruction. For more details refer to CUDA's [Cache Eviction Priority Hints](#).

Label	Name	Description
-------	------	-------------

L2 Explicit Evict Policies	smsp_inst_executed_memdep_explicitated_type	This table lists the type of configured explicit eviction policies. As the policies can be set dynamically at runtime, this list includes all policies that were part of any executed instruction.
L2 Explicit Hit Policy Evict First	smsp_inst_executed_memdep_evict_first	The number of times a memory instruction was executed by any warp which had the <b>evict_first</b> policy set in case the access leads to a cache hit in L2. Data cached with this policy will be first in the eviction priority order and will likely be evicted when cache eviction is required. This policy is suitable for streaming data.
L2 Explicit Hit Policy Evict Last	smsp_inst_executed_memdep_evict_last	The number of times a memory instruction was executed by any warp which had the <b>evict_last</b> policy set in case the access leads to a cache hit in L2. Data cached with this policy will be last in the eviction priority order and will likely be evicted only after other data with <b>evict_normal</b> or <b>evict_first</b> eviction policy is already evicted. This policy is suitable for streaming data.

		for data that should remain persistent in cache.
L2 Explicit Hit Policy Evict Normal	smsp_inst_executed_memd	<b>Number of times</b> prop_evict_normal memory instruction was executed by any warp which had the <b>evict_normal</b> (default) policy set in case the access leads to a cache hit in L2.
L2 Explicit Hit Policy Evict Normal Demote	smsp_inst_executed_memd	<b>Number of times</b> prop_evict_normal_de memory instruction was executed by any warp which had the <b>evict_normal_demote</b> policy set in case the access leads to a cache hit in L2.
L2 Explicit Miss Policy Evict First	smsp_inst_executed_memd	<b>Number of times</b> prop_evict_first memory instruction was executed by any warp which had the <b>evict_first</b> policy set in case the access leads to a cache miss in L2. Data cached with this policy will be first in the eviction priority order and will likely be evicted cache eviction is required. This policy is suitable for streaming data.
L2 Explicit Miss Policy Evict Normal	smsp_inst_executed_memd	<b>Number of times</b> prop_evict_normal memory instruction was executed by any warp which had the <b>evict_normal</b> (default) policy set in

		case the access leads to a cache miss in L2.
--	--	--

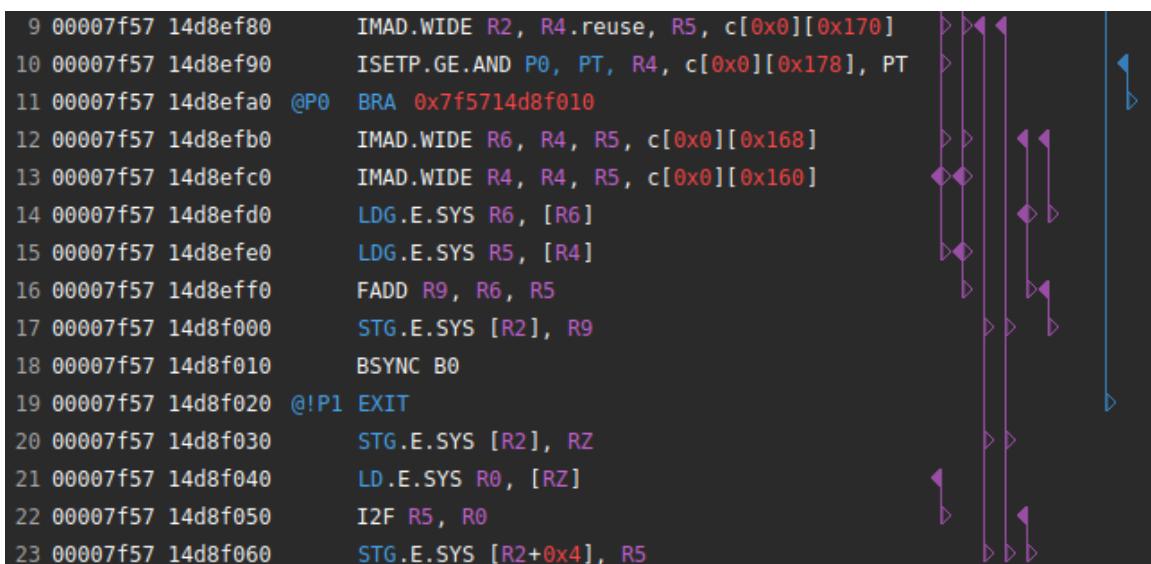
- ▶ **Individual Warp Stall Sampling Metrics**

All *stall\_\** metrics show the information combined in *Warp Stall Sampling* individually. See [Statistical Sampler](#) for their descriptions.

- ▶ See the [Customization Guide](#) on how to add additional metrics for this view and the [Metrics Reference](#) for further information on available metrics.

### Register Dependencies

Dependencies between registers are displayed in the SASS view. When a register is read, all the potential addresses where it could have been written are found. The links between these lines are drawn in the view. All dependencies for registers, predicates, uniform registers and uniform predicates are shown in their respective columns.



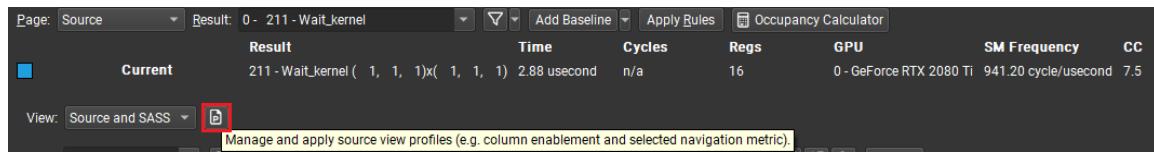
The picture above shows some dependencies for a simple CUDA kernel. On the first row, which is line 9 of the SASS code, we can see *writes* on registers R2 and R3, represented by *filled triangles pointing to the left*. These registers are then *read* on lines 17, 20 and 23, and this is represented by *regular triangles pointing to the right*. There are also some lines where both types of triangles are on the same line, which means that a read and a write occurred for the same register.

Dependencies across source files and functions are not tracked.

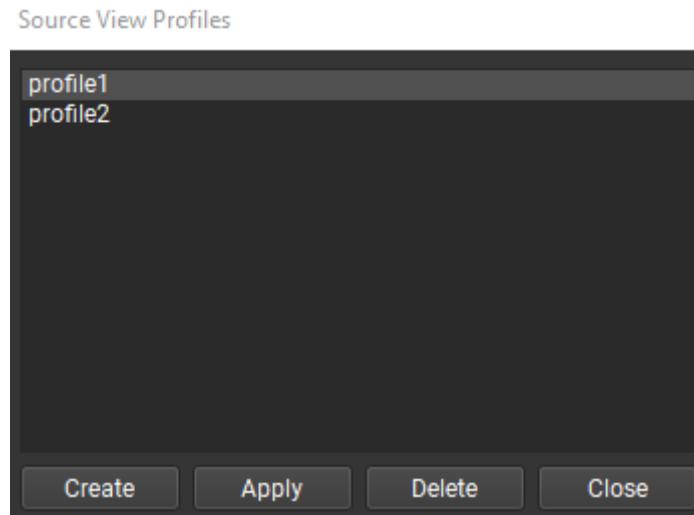
The Register Dependencies Tracking feature is enabled by default, but can be disabled completely in *Tools > Options > Profile > Report Source Page > Enable Register Dependencies*.

### 6.2.4.3. Profiles

The icon next to the *View* dropdown can be used to manage *Source View Profiles*.



This button opens a dialog that shows you the list of saved source view profiles. Such profiles can be created using the *Create* button in the dialog. Profiles let you store the column properties of all views in the report to a file. Such properties include column visibility, freeze state, width, order and the selected navigation metric. A saved profile can be applied to any opened report using the *Apply* button. This updates the column properties mentioned above from the selected profile in all views.



Profiles are useful for configuring views to your preferences, or for a certain use case. Start by choosing metric columns from the *Column Chooser*. Next, configure other properties like freezing column, changing width or order and setting a heatmap metric in the *Navigation* dropdown before creating the profile. Once a profile is created, you can always use this profile on any opened report to hide all non-required columns or to restore your configured properties. Simply select the profile from the source view profiles dialog and click the *Apply* button.

Note that the column properties are stored separately for each *View* in the profile and when applied, only those views will be updated which are present in the selected profile. You will not see the metric columns that are not available in your report even if those were configured to be visible in the source profile you have applied.

#### 6.2.4.4. Limitations

##### Range Replay

When using Range Replay mode, instruction-level source metrics are not available.

## Graph Profiling

When profiling complete CUDA graphs, instruction-level source metrics are not available.

## Cmdlsts

When profiling (OptiX) cmdlists, instruction-level source metrics are not available.

## 6.2.5. Comments Page

The *Comments* page aggregates all section comments in a single view and allows the user to edit those comments on any launch instance or section, as well as on the overall report. Comments are persisted with the report. If a section comment is added, the comment icon of the respective section in the [Details Page](#) will be highlighted.

## 6.2.6. Call Stack / NVTX Page

The *CPU Call Stack* section of this report page shows the CPU call stack for the executing CPU thread at the time the kernel was launched. For this information to show up in the profiler report, the option to collect CPU call stacks had to be enabled in the [Connection Dialog](#) or using the corresponding NVIDIA Nsight Compute CLI command line parameter.

CPU Call Stack:					
#	Module	PC	Function	Line	File
5	CuVectorAddDrv.exe	0x7ff7106d3236	CudaDevice::Launch	288	D:\Src\Sw\
4	CuVectorAddDrv.exe	0x7ff7106d6b7f	main	818	D:\Src\Sw\
3	CuVectorAddDrv.exe	0x7ff7106dceb0	_scrt_common_main_seh	288	d:\agent\_\v
2	KERNEL32.DLL	0x7fff1afb7034	BaseThreadInitThunk		
1	ntdll.dll	0x7fff1bde2651	RtlUserThreadStart		

The *NVTX State* section of this report page shows the NVTX context when the kernel was launched. All thread-specific information is with respect to the thread of the kernel's launch API call. Note that NVTX information is only collected if the profiler is started with NVTX support enabled, either in the [Connection Dialog](#) or using the NVIDIA Nsight Compute CLI command line parameter.

7967	Category	Color	API Call ID	Payload
▼ <default domain>				
▼ Push/Pop Stack				
0: starting_cuda			105	
1: starting_kernel_execution			122	

## 6.2.7. Raw Page

The *Raw* page shows a list of all collected metrics with their units per profiled kernel launch. It can be exported, for example, to CSV format for further analysis. The page

features a filter edit to quickly find specific metrics. You can transpose the table of kernels and metrics by using the *Transpose* button.

If a metric has multiple instance values, the number of instances is shown after the standard value. This metric for example has ten instance values: **35.48 {10}**. You can select in the [Profile](#) options dialog that all instance values should be shown individually or inspect the metric result in the [Metric Details](#) tool window.

## 6.3. Metrics and Units

Numeric metric values are shown in various places in the report, including the header and tables and charts on most pages. NVIDIA Nsight Compute supports various ways to display those metrics and their values.

When available and applicable to the UI component, metrics are shown along with their unit. This is to make it apparent if a metric represents cycles, threads, bytes/s, and so on. The unit will normally be shown in rectangular brackets, e.g. **Metric Name [bytes] 128**.

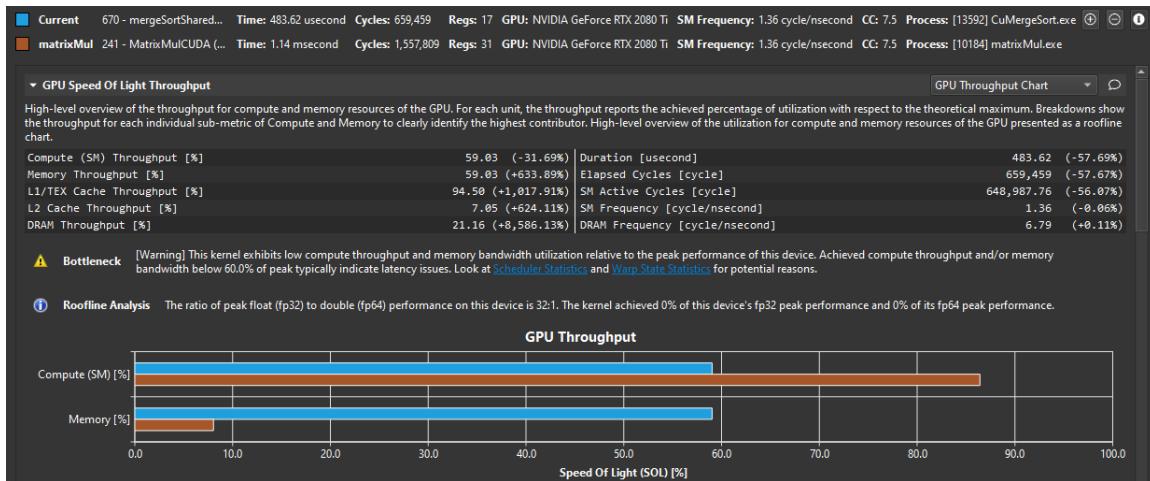
By default, units are scaled automatically so that metric values are shown with a reasonable order of magnitude. Units are scaled using their SI-factors, i.e. byte-based units are scaled using a factor of 1000 and the prefixes K, M, G, etc. Time-based units are also scaled using a factor of 1000, with the prefixes n, u and m. This scaling can be disabled in the [Profile](#) options.

Metrics which could not be collected are shown as **n/a** and assigned a warning icon. If the metric floating point value is out of the regular range (i.e. **nan** (Not a number) or **inf** (infinite)), they are also assigned a warning icon. The exception are metrics for which these values are expected and which are allow-listed internally.

Numeric metric counter values for selected results are aggregated and shown in the column header of the *Raw* and *Summary* page table. If no result is selected, aggregated values of all results are shown. To see the aggregation for distinct metrics, select the desired cells in the table. This will show the aggregate of all selected values in the bottom-right status bar, e.g. **Sum: 1000**. This aggregation can be disabled in the [Profile](#) options.

# Chapter 7. BASELINES

NVIDIA Nsight Compute supports diffing collected results across one or multiple reports using Baselines. Each result in any report can be promoted to a baseline. This causes metric values from all results in all reports to show the difference to the baseline. If multiple baselines are selected simultaneously, metric values are compared to the average across all current baselines. Baselines are not stored with a report and are only available as long as the same NVIDIA Nsight Compute instance is open, unless they are saved to a **ncu-bln** file from the [Baselines tool window](#).



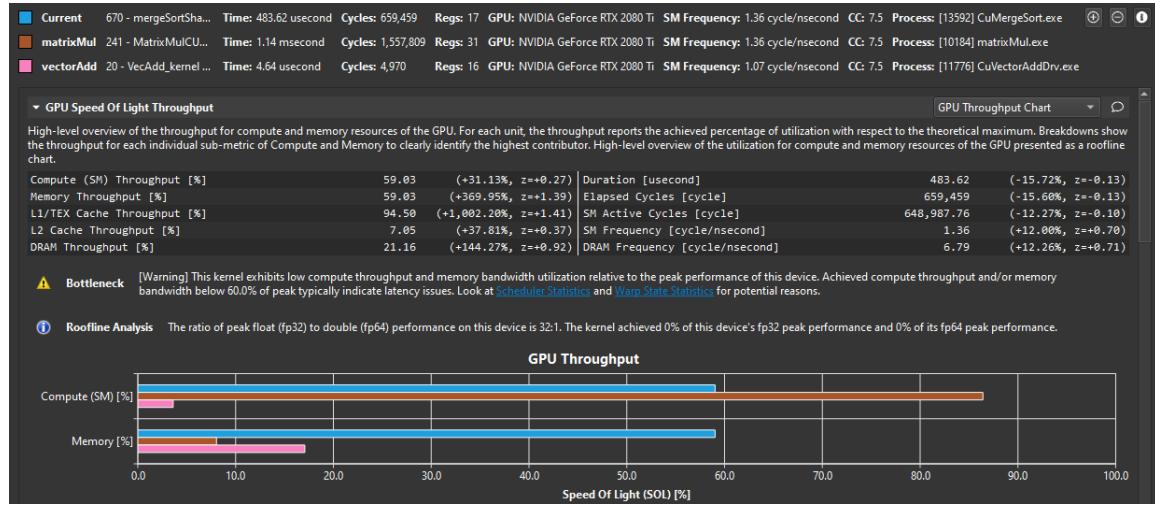
Select *Add Baseline* to promote the current result in focus to become a baseline. If a baseline is set, most metrics on the [Details Page](#), [Raw Page](#) and [Summary Page](#) show two values: the current value of the result in focus, and the corresponding value of the baseline or the percentage of change from the corresponding baseline value. (Note that an infinite percentage gain, *inf%*, may be displayed when the baseline value for the metric is zero, while the focus value is not.)

If multiple baselines are selected, each metric will show the following notation:

```
<focus value> (<difference to baselines average [%]>, z=<standard score>@<number of values>)
```

The standard score is the difference between the current value and the average across all baselines, normalized by the standard deviation. If the number of metric values

contributing to the standard score equals the number of results (current and all baselines), the @<number of values> notation is omitted.



Double-clicking on a baseline name allows the user to edit the displayed name. Edits are committed by pressing **Enter/Return** or upon loss of focus, and abandoned by pressing **Esc**. Hovering over the baseline color icon allows the user to remove this specific baseline from the list.

Use the *Clear Baselines* entry from the dropdown button, the **Profile** menu, or the corresponding toolbar button to remove all baselines.

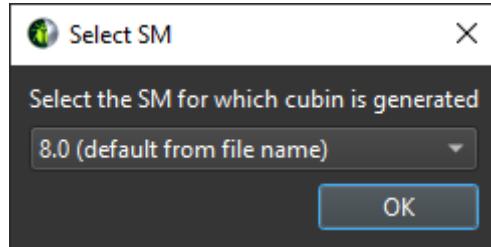
Baseline changes can also be made in the **Baselines tool window**.

# Chapter 8.

# STANDALONE SOURCE VIEWER

NVIDIA Nsight Compute includes a standalone source viewer for *cubin* files. This view is identical to the [Source Page](#), except that it won't include any performance metrics.

Cubin files can be opened from the *File > Open* main menu command. The SM Selection dialog will be shown before opening the standalone source view. If available, the SM version present in the file name is pre-selected. For example, if your file name is `mergeSort.sm_80.cubin` then SM 8.0 will be pre-selected in the dialog. Choose the appropriate SM version from the drop down menu if it's not included in the file name.



Click Ok button to open [Standalone Source Viewer](#).

The screenshot displays the NVIDIA Nsight Compute Standalone Source Viewer interface. It features two main panes: "Source mergeSort.cu" and "Source binarySearchInclusive".

**Source mergeSort.cu:**

```

45     return iDivUp(dividend, SAMPLE_STRIDE);
46 }
47
48 #define W (sizeof(uint) * 8)
49 static inline __device__ uint nextPowerOfTwo(uint x)
50 {
51     /*
52     --x;
53     x |= x >> 1;
54     x |= x >> 2;
55     x |= x >> 4;
56     x |= x >> 8;
57     x |= x >> 16;
58     return ++x;
59 */
60     return 1U << (W - __clz(x - 1));
61 }
62
63 template<uint sortDir> static inline __device__ uint binarySearchInclusive(uint \n
64 {
65     if (L == 0)
66     {
67         return 0;
68     }
69
70     uint pos = 0;
71
72     for (; stride > 0; stride >>= 1)
73     {
    
```

**Source binarySearchInclusive:**

```

15 00000000 0001cd60    STL [R1], R2
16 00000000 0001cd70    BMov.32,CLEAR R22, B7
17 00000000 0001cd80    BMov.32,CLEAR R19, B6
18 00000000 0001cd90    MOV R8, R8
19 00000000 0001ca00    MOV R5, R5
20 00000000 0001cb00    MOV R7, R7
21 00000000 0001cc00    MOV R6, R6
22 00000000 0001cd00    MOV R4, R4
23 00000000 0001ce00    MOV R4, R4
24 00000000 0001cf00    MOV R6, R6
25 00000000 0001ce00    MOV R7, R7
26 00000000 0001ce10    MOV R5, R5
27 00000000 0001ce20    MOV R8, R8
28 00000000 0001ce30    MOV R17, R4
29 00000000 0001ce40    MOV R24, R6
30 00000000 0001ce50    MOV R25, R7
31 00000000 0001ce60    MOV R18, R5
32 00000000 0001ce70    MOV R8, R8
33 00000000 0001ce80    ISETP_EQ,AND P0, PT, R18, R2, PT
34 00000000 0001ce90    PLOP3,LUT P0, PT, P0, PT, PT, 0x8, 0x0
35 00000000 0001ceA0    BSSY B7, 0x1d4c0
36 00000000 0001ceB0    @P0 BRA 0x1cf30
37 00000000 0001ceC0    BRA 0x1ced0
38 00000000 0001cd00    MOV R8, RZ
39 00000000 0001ce00    MOV R8, R0
40 00000000 0001ceF0    MOV R8, R8
41 00000000 0001cf00    MOV R8, R0
42 00000000 0001cf10    BRA 0x1d4b0
43 00000000 0001cf20    BRA 0x1d460
    
```

# Chapter 9.

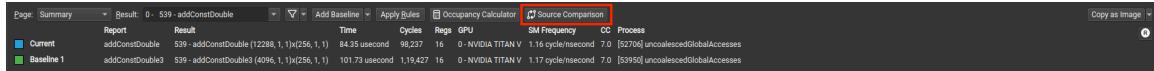
## SOURCE COMPARISON

Source comparison provides a way to see the source files of two profile results side by side. It enables to quickly identify source differences and understand changes in metric values.

To compare two results side by side add one result as a baseline, navigate to the other result, and then click the *Source Comparison* button located in the report header.

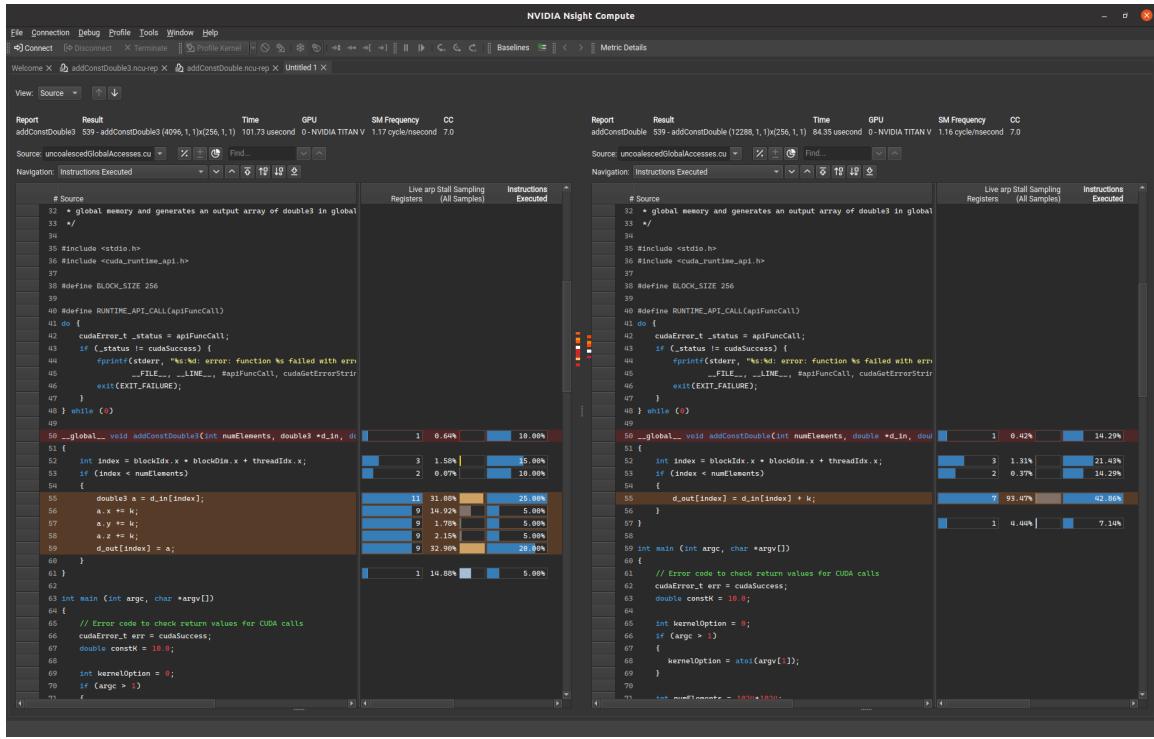
For example, if you want to compare kernel XYZ from report R1 with kernel XYZ from report R2, first open report R1, add the profile result for kernel XYZ as baseline, open report R2, choose kernel XYZ, and then click the Source Comparison button.

Source comparison will be shown only with first added baseline result.



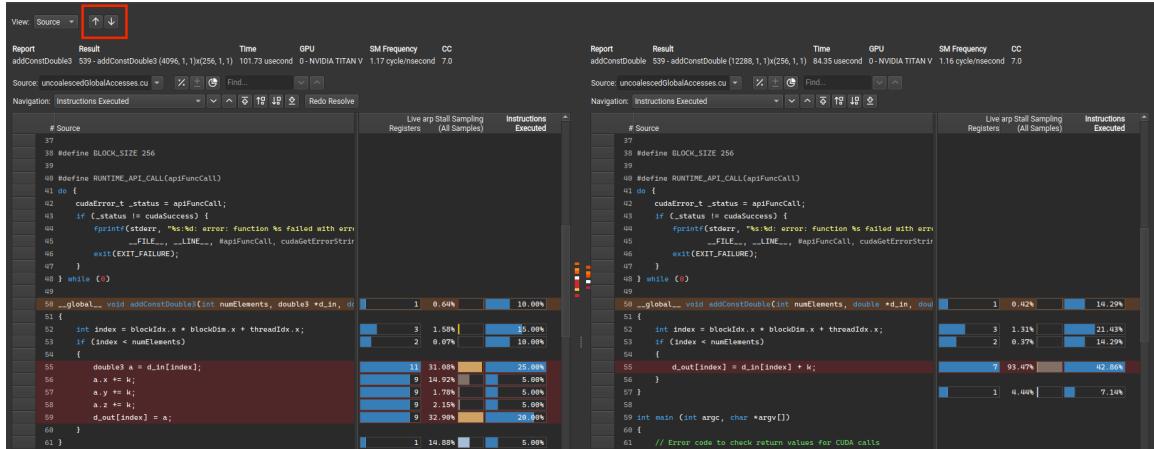
The screenshot shows the Nsight Compute interface with the 'Source Comparison' button highlighted in red. The table below lists two results: 'Current' and 'Baseline 1'. The 'Current' row shows the kernel 'addConstDouble' with a result of 539, running on an NVIDIA TITAN V GPU at 1.16 cycles/nsec. The 'Baseline 1' row shows the kernel 'addConstDouble3' with a result of 539, running on an NVIDIA TITAN V GPU at 1.17 cycles/nsec.

Report	Result	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	addConstDouble	539 - addConstDouble(12288, 1)(256, 1, 1)	84.35 usecond	98,237	16	0 - NVIDIA TITAN V	1.16 cycle/nsec	7.0 [52705] uncoalescedGlobalAccesses
Baseline 1	addConstDouble3	539 - addConstDouble3(4096, 1)(256, 1, 1)	101.79 usecond	1,19,427	16	0 - NVIDIA TITAN V	1.17 cycle/nsec	7.0 [53950] uncoalescedGlobalAccesses



Currently only high-level Source (CUDA-C) view is supported for comparison.

Navigation to the previous or next difference is supported using the navigation buttons or the keyboard shortcuts *Ctrl + 1* and *Ctrl + 2*.



# Chapter 10.

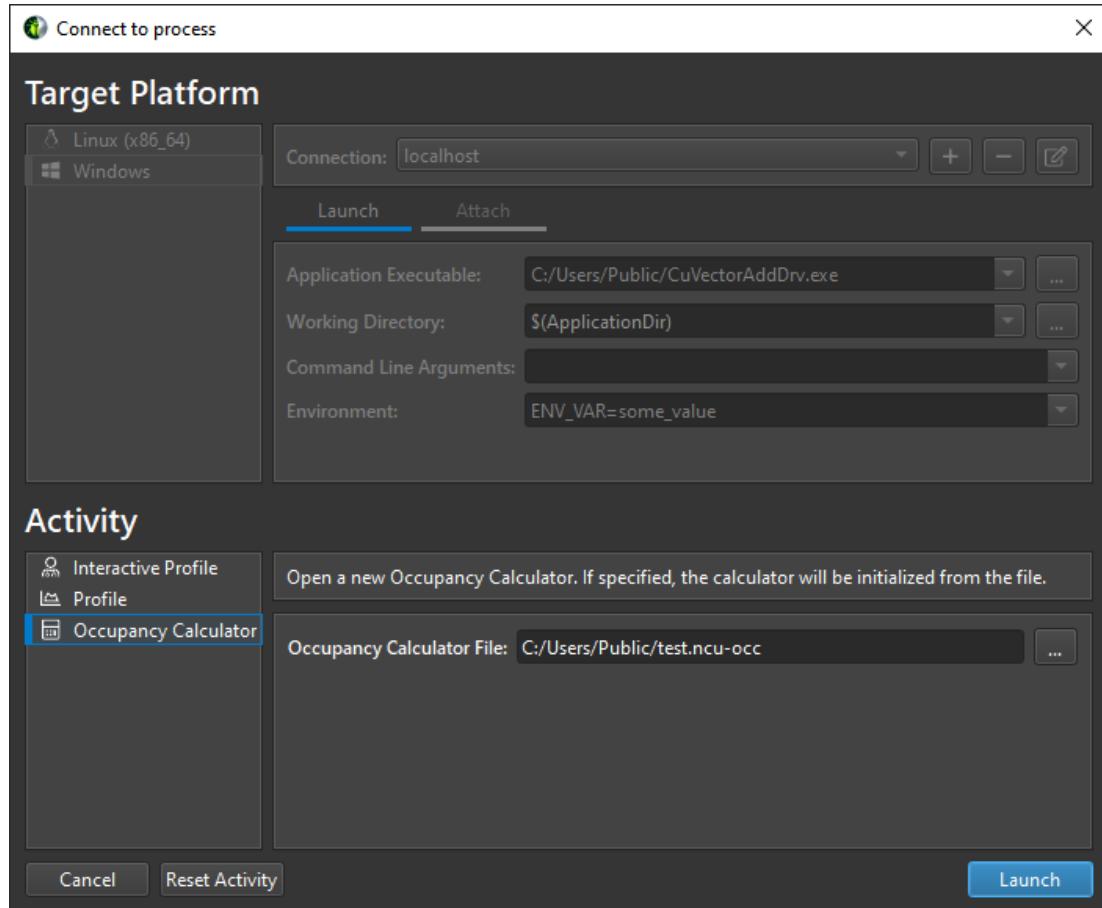
# OCCUPANCY CALCULATOR

NVIDIA Nsight Compute provides an *Occupancy Calculator* that allows you to compute the multiprocessor occupancy of a GPU for a given CUDA kernel. It offers feature parity to the CUDA Occupancy Calculator [spreadsheet](#).

The Occupancy Calculator can be opened directly from a profile report or as a new activity. The occupancy calculator data can be saved to a file using *File > Save*. By default, the file uses the `.ncu-occ` extension. The occupancy calculator file can be opened using *File > Open File*

## 1. Launching from the *Connection Dialog*

Select the Occupancy Calculator activity from the connection dialog. You can optionally specify an occupancy calculator data file, which is used to initialize the calculator with the data from the saved file. Click the *Launch* button to open the Occupancy Calculator.



## 2. Launching from the *Profiler Report*

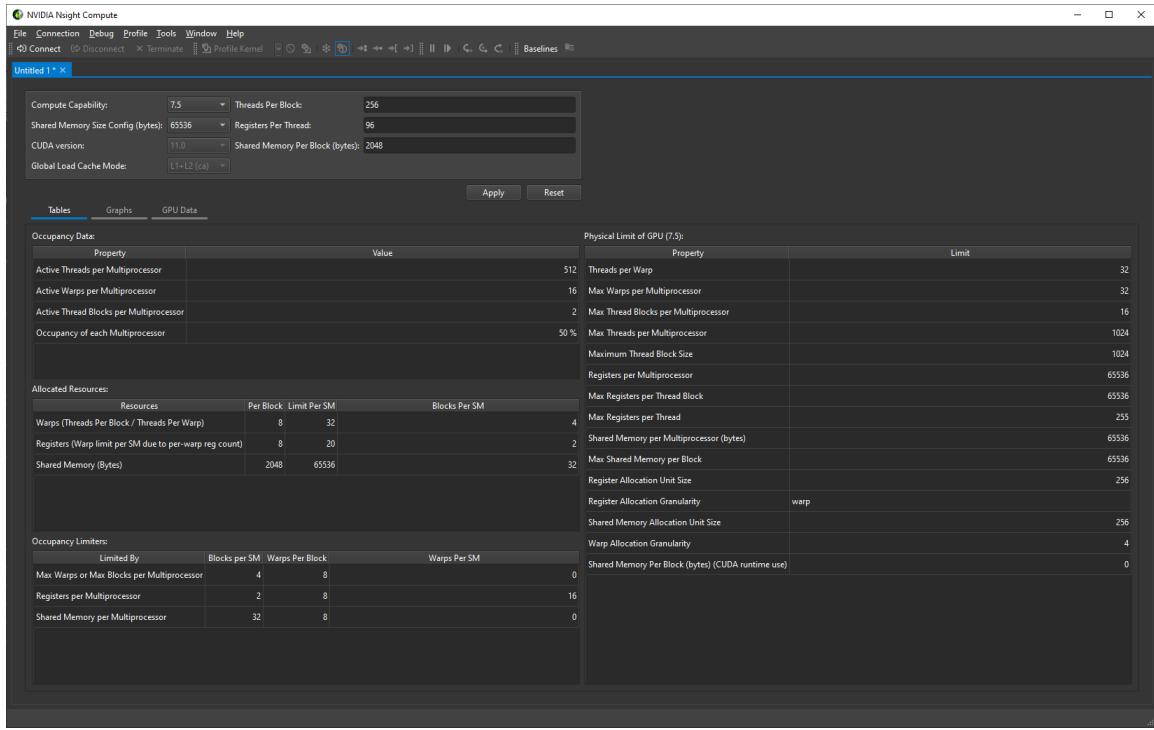
The Occupancy Calculator can be opened from the *Profiler Report* using the calculator button located in the report header or in the header of the *Occupancy* section on the *Detail Page*.

Theoretical Occupancy [%]	50	Block Limit Registers [block]	84
Theoretical Active Warps per SM [warp]	32	Block Limit Shared Mem [block]	32
Achieved Occupancy [%]	1.56	Block Limit Warps [block]	64
Achieved Active Warps Per SM [warp]	1.00	Block Limit SM [block]	32

The user interface consists of an input section as well as tables and graphs that display information about GPU occupancy. To use the calculator, change the input values in the input section, click the *Apply* button and examine the tables and graphs.

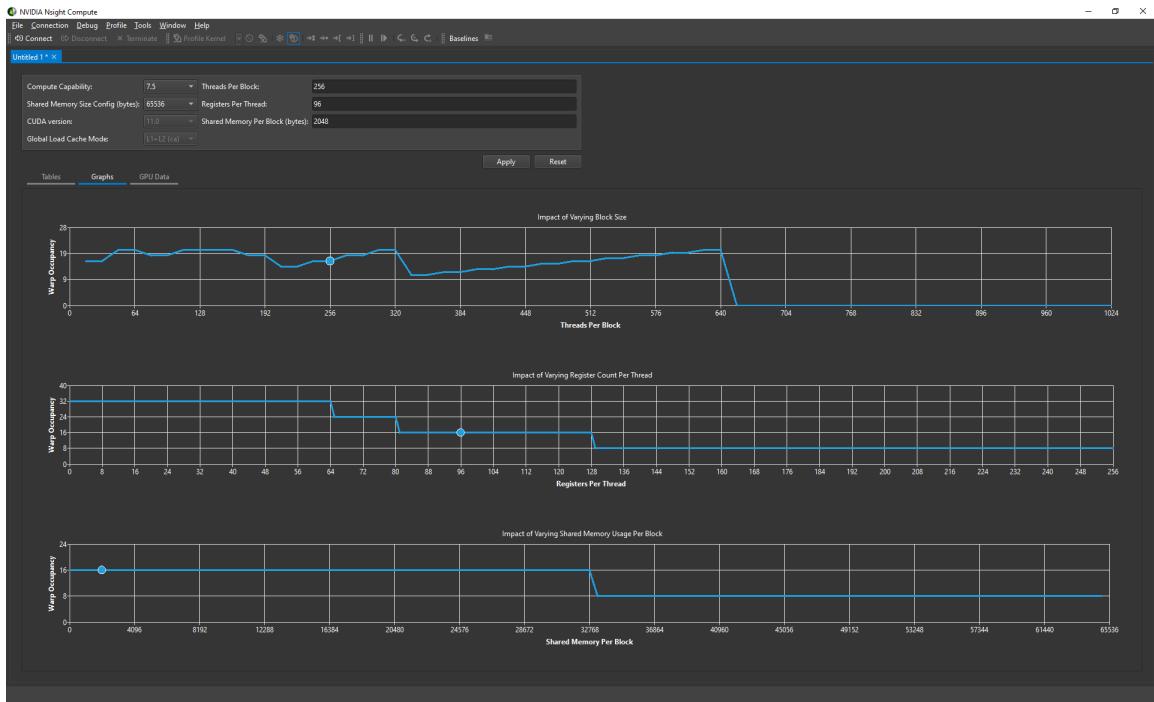
## 10.1. Tables

The tables show the occupancy, as well as the number of active threads, warps, and thread blocks per multiprocessor, and the maximum number of active blocks on the GPU.



## 10.2. Graphs

The graphs show the occupancy for your chosen block size as a blue circle, and for all other possible block sizes as a line graph.



## 10.3. GPU Data

The *GPU Data* shows the properties of all supported devices.

The screenshot shows the NVIDIA Nsight Compute application window. The title bar reads "NVIDIA Nsight Compute". The menu bar includes File, Connection, Debug, Profile, Tools, Window, Help, and a toolbar with various icons. A top panel displays configuration settings: Compute Capability (7.5), Threads Per Block (256), Shared Memory Size Config (bytes) (65536), Registers Per Thread (96), CUDA version (11.0), Shared Memory Per Block (bytes) (2048), and Global Load Cache Mode (L1+L2 (ca)). Below this is a "Tables" tab selected, showing a detailed table of GPU properties for various architectures from sm\_32 to sm\_86. The table includes columns for Compute Capability, SM Version, Threads / Warp, Warps / Multiprocessor, Threads / Multiprocessor, Thread Blocks / Multiprocessor, Shared Memory / Multiprocessor, Max Shared Memory / Block, Register File Size / Multiprocessor, Max Registers / Block, Register Allocation Unit Size, and Registers. The data shows that for Compute Capability 7.5 (sm\_75), there are 32 warps per multiprocessor, 64 threads per warp, 1024 threads per block, 16 thread blocks per multiprocessor, 65536 shared memory bytes per block, and 65536 registers per block. The table continues for other architectures like sm\_32, sm\_35, sm\_37, sm\_50, sm\_52, sm\_53, sm\_60, sm\_61, sm\_62, sm\_70, sm\_75, sm\_80, and sm\_86.

Compute Capability	SM Version	Threads / Warp	Warps / Multiprocessor	Threads / Multiprocessor	Thread Blocks / Multiprocessor	Shared Memory / Multiprocessor	Max Shared Memory / Block	Register File Size / Multiprocessor	Max Registers / Block	Register Allocation Unit Size	Registers
3.2	sm_32	32	64	2048	16	49152	49152	65536	65536	256	warp
3.5	sm_35	32	64	2048	16	49152	49152	65536	65536	256	warp
3.7	sm_37	32	64	2048	16	114688	49152	131072	65536	256	warp
5.0	sm_50	32	64	2048	32	65536	49152	65536	65536	256	warp
5.2	sm_52	32	64	2048	32	98304	49152	65536	32768	256	warp
5.3	sm_53	32	64	2048	32	65536	49152	65536	32768	256	warp
6.0	sm_60	32	64	2048	32	65536	49152	65536	65536	256	warp
6.1	sm_61	32	64	2048	32	98304	49152	65536	65536	256	warp
6.2	sm_62	32	64	2048	32	65536	49152	65536	65536	256	warp
7.0	sm_70	32	64	2048	32	98304	98304	65536	65536	256	warp
7.5	sm_75	32	32	1024	16	65536	65536	65536	65536	256	warp
8.0	sm_80	32	64	2048	32	167936	167936	65536	65536	256	warp
8.6	sm_86	32	48	1536	16	102400	102400	65536	65536	256	warp

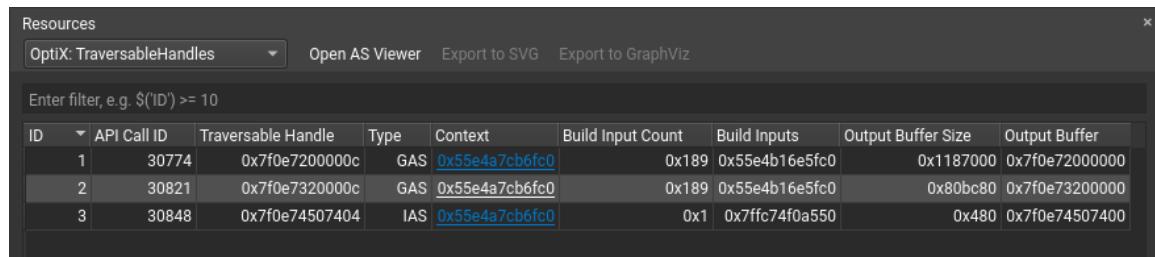
# Chapter 11.

## ACCELERATION STRUCTURE VIEWER

The *Acceleration Structure Viewer* allows inspection of acceleration structures built using the OptiX API. In modern ray tracing APIs like OptiX, *acceleration structures* are data structures describing the rendered scene's geometries that will be intersected when performing ray tracing operations. More information concerning acceleration structures can be found in the [OptiX programming guide](#).

It is the responsibility of the user to set these up and pass them to the OptiX API which translates them to internal data structures that perform well on modern GPUs. The description created by the user can be very error-prone and it is sometimes hard to understand why the rendered result is not as expected. The *Acceleration Structure Viewer* is a component allowing OptiX users to inspect the acceleration structures they build before launching a ray tracing pipeline.

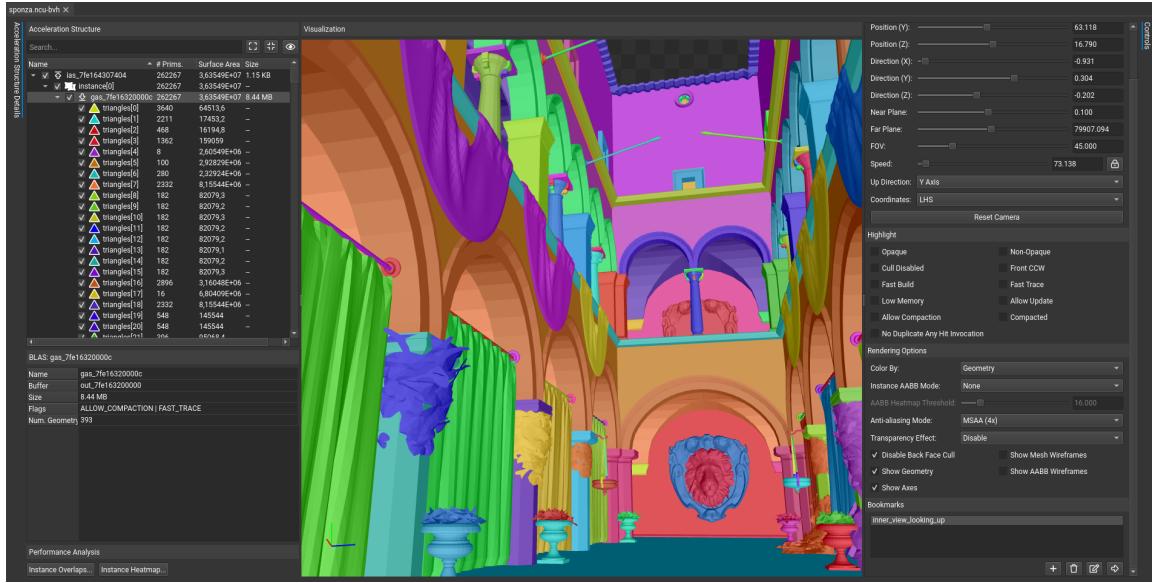
The *Acceleration Structure Viewer* is opened through a button in the [Resources](#) window. The button will only be available when the currently viewed resource is *OptiX: TraversableHandles*. It opens the currently selected handle.



The screenshot shows the 'Resources' window with the tab 'OptiX: TraversableHandles' selected. There are three buttons at the top: 'Open AS Viewer', 'Export to SVG', and 'Export to GraphViz'. Below is a search bar with placeholder text 'Enter filter, e.g. \${ID} >= 10'. A table lists three traversable handles:

ID	API Call ID	Traversable Handle	Type	Context	Build Input Count	Build Inputs	Output Buffer Size	Output Buffer
1	30774	0x7f0e7200000c	GAS	0x55e4a7cb6fc0	0x189	0x55e4b16e5fc0	0x1187000	0x7f0e72000000
2	30821	0x7f0e7320000c	GAS	0x55e4a7cb6fc0	0x189	0x55e4b16e5fc0	0x80bc80	0x7f0e73200000
3	30848	0x7f0e74507404	IAS	0x55e4a7cb6fc0	0x1	0x7ff74f0a550	0x480	0x7f0e74507400

The viewer is multi-paned: it shows a hierarchical view of the acceleration structure on the left, a graphical view of the acceleration structure in the middle, and controls and options on the right. In the hierarchical tree view on the left of the viewer the *instance acceleration structures (IAS)*, *geometry acceleration structures (GAS)*, child instances and child geometries are shown. In addition to this, some general properties for each of them is shown such as their primitive count, surface area and size on the device.



In the hierarchical view on the left of the *Acceleration Structure Viewer*, the following information is displayed where applicable.

**Table 4 Acceleration Structure Hierarchical Columns**

Column	Description
Name	An identifier for each row in the hierarchy. Click on the check box next to the name to show or hide the selected geometry or hierarchy. Double-click on this entry to jump to the item in the rendering view.
# Prims	The number of primitives that make up this acceleration structure.
Surface Area	A calculation of the total surface area for the AABB that bounds the particular entry.
Size	The size of the output buffer on the device holding this <i>acceleration structure</i> .

Performance analysis tools are accessible in the bottom left corner on the main view. These tools help identify potential performance problems that are outlined in the [RTX Ray Tracing Best Practices Guide](#). These analysis tools aim to give a broad picture of acceleration structures that may exhibit sub-optimal performance. To find the most optimal solution, profiling and experimentation is recommended but these tools may paint a better picture as to why one structure performs poorly compared to another.

**Table 5 Acceleration Structure Analysis Tools**

Action	Description
Instance Overlaps	Identifies instance AABBs that overlap with other instances in 3D. Consider merging GAses when instance world-space AABBs overlap significantly to potentially increase performance.
Instance Heatmap	This allows you to set the threshold used by the AABB heatmap rendered in the visualizer.

## 11.1. Navigation

The *Acceleration Structure Viewer* supports multiple navigation modes. The navigation mode can be changed using the combo box in the camera controls pane, to the right of the rendering pane. The keyboard and mouse bindings for each mode are as follows:

**Table 6 Acceleration Structure Key Bindings**

Binding	Fly Camera	Dolly Camera	Orbit Camera
WASD/ Arrow Keys	Move forward, backward, left, right	Move forward, backward, left, right	Track (Move up, down, left, right)
E/Q	Move up/down	Move up/down	n/a
Z/C	Increase/decrease field of view	Increase/decrease field of view	Increase/decrease field of view
Shift/ Ctrl	Move faster/slower	Move faster/slower	Move faster/slower
Mousewheel	Zoom in/out	Zoom in/out	Zoom in/out
LMB + Drag	Rotate in place	Rotate left/right, move forward/backward	Rotate around the geometry
RMB + Drag	Zoom in/out	Rotate in place	Zoom in/out
MMB + Drag	Track (Move up, down, left, right)	Track (Move up, down, left, right)	Track (Move up, down, left, right)
Alt	Temporarily switch to Orbit Camera	Temporarily switch to Orbit Camera	n/a
F/ Double Click	Focus on the selected geometry	Focus on the selected geometry	Focus on the selected geometry

Based on the coordinate system of the input geometry, you may need to change the **Up Direction** setting to Z-Axis or the **Coordinates** setting to RHS. To reset the camera to its original location, click **Reset Camera**.

There are also a selection of Camera Controls for fast and precise navigation. To save a position, use the bookmarks controls. Each node within the acceleration structure hierarchy can also be double-clicked to quickly navigate to that location.



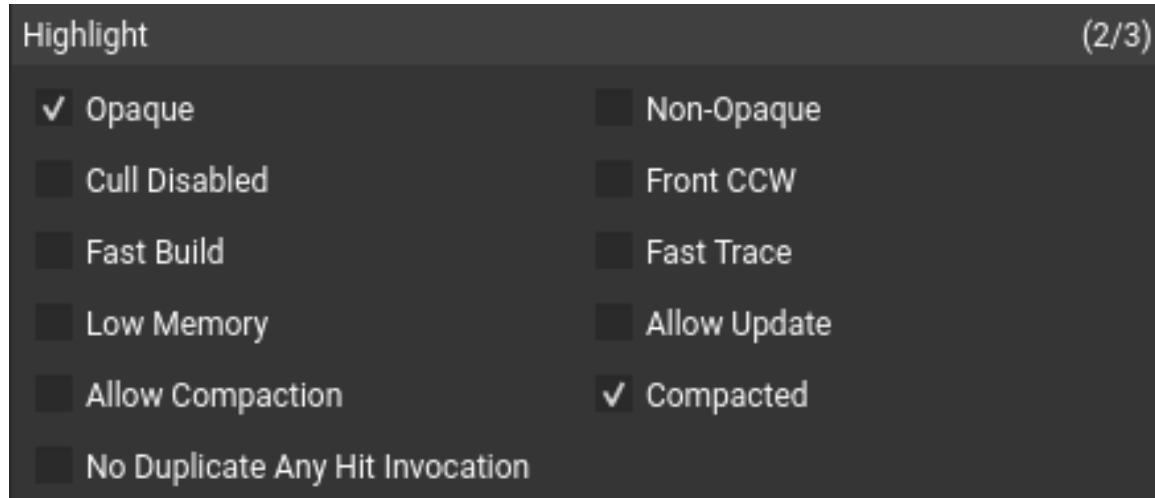
## 11.2. Filtering and Highlighting

The acceleration structure view supports acceleration structure filtering as well as highlighting of data matching particular characteristics. The checkboxes next to each geometry allow users to toggle the rendering of each traversable.

Geometry instances can also be selected by clicking on them in the main graphical view. Additionally, right clicking in the main graphical view gives options to hide or show all geometry, hide the selected geometry, or hide all but the selected geometry.

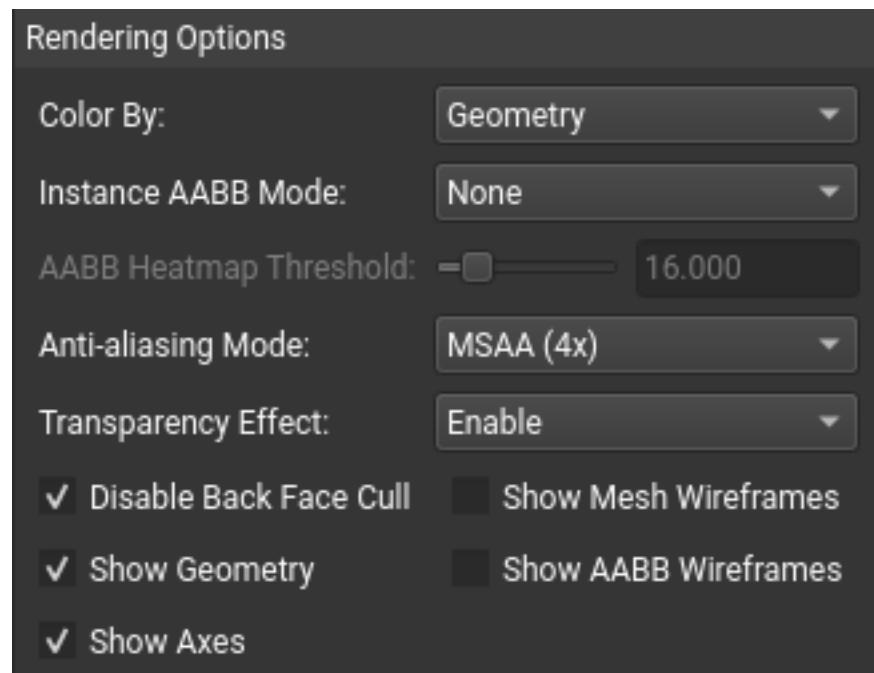
Name	# Prims.	Surface Area	Size
ias_7f024a507404	262267	3,63549e+07	1.15 KB
instance[0]	262267	3,63549e+07	-
gas_7f024920000c	262267	3,63549e+07	8.44 MB
triangles[0]	3640	64513,6	-
triangles[1]	2211	17453,2	-
triangles[2]	468	16194,8	-
triangles[3]	1362	159059	-
triangles[4]	8	2,60549e+06	-
triangles[5]	100	2,92829e+06	-
triangles[6]	280	2,32924e+06	-
triangles[7]	2332	8,15544e+06	-
triangles[8]	182	82079,3	-
triangles[9]	182	82079,2	-
triangles[10]	182	82079,3	-
triangles[11]	182	82079,2	-
triangles[12]	182	82079,2	-
triangles[13]	182	82079,1	-
triangles[14]	182	82079,2	-
triangles[15]	182	82079,3	-
triangles[16]	2896	3,16048e+06	-

Beyond filtering, the view also supports highlight-based identification of geometry specified with particular flags. Checking each highlight option will identify those resources matching that flag, colorizing for easy identification. Clicking an entry in this section will dim all geometry that does **not** meet the filter criteria allowing items that match the filter to standout. Selecting multiple filters requires the passing geometry to meet all selected filters (e.g., AND logic). Additionally, the heading text will be updated to reflect the number of items that meet this filter criteria.



## 11.3. Rendering Options

Under the highlight controls, additional rendering options are available. These include methods to control the geometry colors and the ability to toggle the drawing of wireframes for meshes and AABBs.



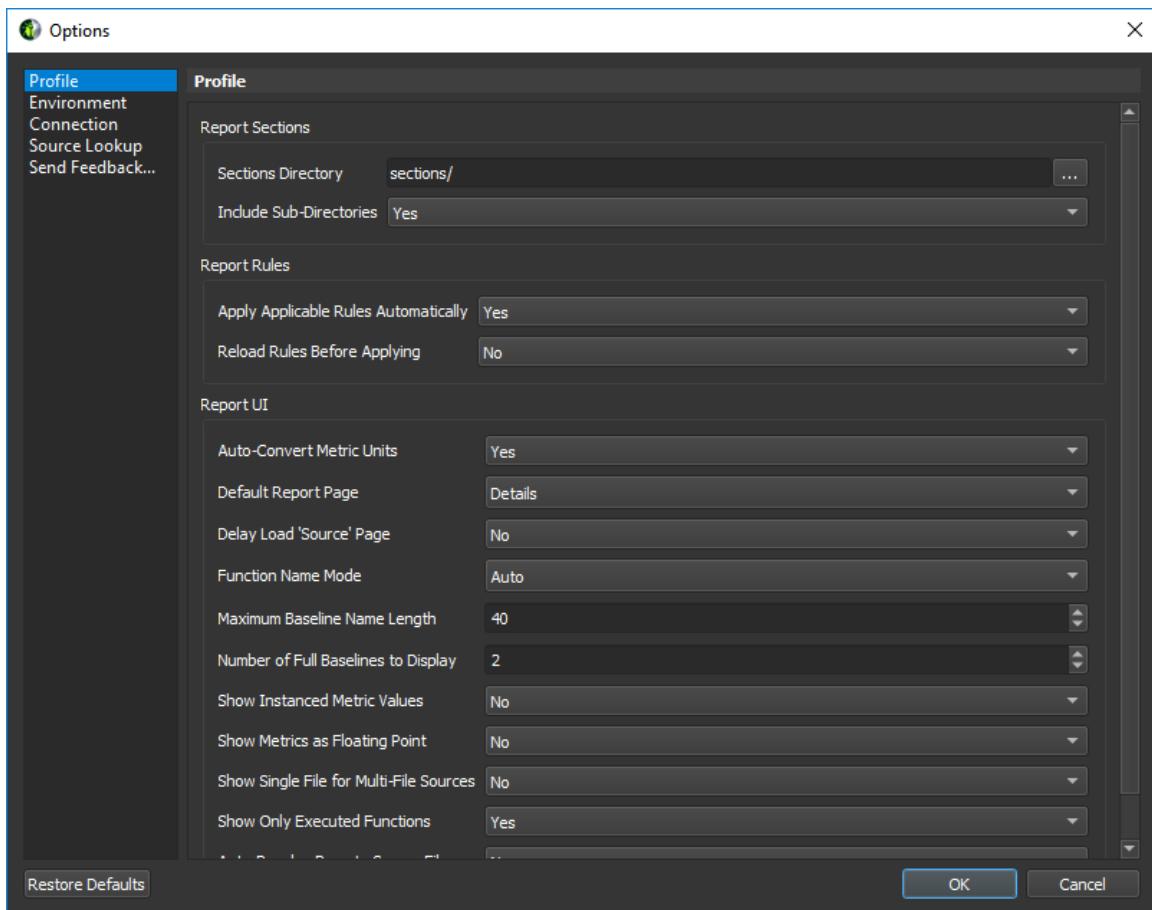
## 11.4. Exporting

The data displayed in the acceleration structure viewer document can be saved to file. Exporting an *Acceleration Structure Viewer* document allows for persisting the data you have collected beyond the immediate analysis session. This capability is particularly

valuable for comparing different revisions of your geometry or sharing with others. Bookmarks are persisted as well.

# Chapter 12. OPTIONS

NVIDIA Nsight Compute options can be accessed via the main menu under *Tools > Options*. All options are persisted on disk and available the next time NVIDIA Nsight Compute is launched. When an option is changed from its default setting, its label will become bold. You can use the *Restore Defaults* button to restore all options to their default values.



## 12.1. Profile

Table 7 NVIDIA Nsight Compute Profile Options

Name	Description	Values
Sections Directory	Directory from which to import section files and rules. Relative paths are with respect to the NVIDIA Nsight Compute installation directory.	
Include Sub-Directories	Recursively include section files and rules from sub-directories.	Yes (Default)/No
Apply Applicable Rules Automatically	Automatically apply active and applicable rules.	Yes (Default)/No
Reload Rules Before Applying	Force a rule reload before applying the rule to ensure changes in the rule script are recognized.	Yes/No (Default)
Default Report Page	The report page to show when a report is generated or opened. <i>Auto</i> lets the tool decide the best page to show when opening a report.	<ul style="list-style-type: none"> <li>▶ Session</li> <li>▶ Summary</li> <li>▶ Details</li> <li>▶ Source</li> <li>▶ Comments</li> <li>▶ Call Stack/NVTX</li> <li>▶ Raw</li> <li>▶ Auto (default)</li> </ul>
Function Name Mode	Determines how function/kernel names are shown.	<ul style="list-style-type: none"> <li>▶ Auto (default): each component uses its preferred mode</li> <li>▶ Demangled: kernel names are shown demangled with all parameters</li> <li>▶ Function: kernel names are shown with their demangled function name without parameters</li> <li>▶ Mangled: kernel names are shown with their mangled name, if applicable</li> </ul>
NVTX Rename Mode	Determines how NVTX information is used for renaming. Range replay results are always renamed when possible.	<ul style="list-style-type: none"> <li>▶ None: no renaming</li> <li>▶ Kernel: kernel names are renamed using the most recent enclosing push/pop range</li> <li>▶ Resources (default): resources like CPU threads</li> </ul>

Name	Description	Values
		or CUDA contexts and streams are renamed ► All: Kernel and Resources
Maximum Baseline Name Length	The maximum length of baseline names.	1..N (Default: 40)
Number of Full Baselines to Display	Number of baselines to display in the report header with all details in addition to the current result.	0..N (Default: 2)
Auto-Convert Metric Units	Auto-adjust displayed metric units and values (e.g. Bytes to KBytes).	Yes (Default)/No
Show Instanced Metric Values	Show the individual values of instanced metrics in tables.	Yes/No (Default)
Show Metrics As Floating Point	Show all numeric metrics as floating-point numbers.	Yes/No (Default)
Show Knowledge Base Information	Show information from the knowledge base in (metric) tooltips to explain terminology. Note: Nsight Compute needs to be restarted for this option to take effect.	Yes (Default)/No
Metrics/Properties	List of metrics and properties to show on the summary page. Comma-separated list of metric entries. Each entry has the format {Label:MetricName}.	
Delay Load 'Source' Page	Delays loading the content of the report page until the page becomes visible. Avoids processing costs and memory overhead until the report page is opened.	Yes/No (Default)
Show Single File For Multi-File Sources	Shows a single file in each Source page view, even for multi-file sources.	Yes/No (Default)
Show Only Executed Functions	Shows only executed functions in the source page views. Disabling this can impact performance.	Yes (Default)/No
Auto-Resolve Remote Source Files	Automatically try to resolve remote source files on the source page (e.g. via SSH) if the connection is still registered.	Yes/No (Default)
Enable Register Dependencies	Track dependencies between SASS registers/predicates and display them in the SASS view.	Yes (Default)/No
Kernel Analysis Size Threshold (KB)	Enable SASS flow graph analysis for functions below this threshold. SASS analysis is required for Live Register	-1..N (Default: 1024)

Name	Description	Values
	and Register Dependency information. Set to -1 to enable analysis for all functions.	
Enable ELF Verification	Enable ELF (cubin) verification to run every time before SASS analysis. This should only be enabled when working with applications compiled before CUDA 11.0 or when encountering source page issues.	Yes/No (Default)
API Call History	Number of recent API calls shown in API Stream View.	1..N (Default: 100)

## 12.2. Environment

Table 8 NVIDIA Nsight Compute Environment Options

Name	Description	Values
Color Theme	The currently selected UI color theme.	<ul style="list-style-type: none"> <li>▶ Dark (Default)</li> <li>▶ Light</li> </ul>
Mixed DPI Scaling	Disable Mixed DPI Scaling if unwanted artifacts are detected when using monitors with different DPs.	<ul style="list-style-type: none"> <li>▶ Auto (Default)</li> <li>▶ Off</li> </ul>
Default Document Folder	Directory where documents unassociated with a project will be saved.	
At Startup	What to do when NVIDIA Nsight Compute is launched.	<ul style="list-style-type: none"> <li>▶ Show welcome page (Default)</li> <li>▶ Show quick launch dialog</li> <li>▶ Load last project</li> <li>▶ Show empty environment</li> </ul>
Show version update notifications	Show notifications when a new version of this product is available.	<ul style="list-style-type: none"> <li>▶ Yes (Default)</li> <li>▶ No</li> </ul>

## 12.3. Connection

Connection properties are grouped into *Target Connection Options* and *Host Connection Properties*.

## Target Connection Properties

The *Target Connection Properties* determine how the host connects to the target application during an *Interactive Profile Activity*. This connection is used to transfer profile information to the host during the profile session.

**Table 9 NVIDIA Nsight Compute Target Connection Properties**

Name	Description	Values
Base Port	Base port used to establish a connection from the host to the target application during an <i>Interactive Profile</i> activity (both local and remote).	1-65535 (Default: 49152)
Maximum Ports	Maximum number of ports to try (starting from <i>Base Port</i> ) when attempting to connect to the target application.	2-65534 (Default: 64)

## Host Connection Properties

The *Host Connection Properties* determine how the command line profiler will connect to the host application during a *Profile Activity*. This connection is used to transfer profile information to the host during the profile session.

**Table 10 NVIDIA Nsight Compute Host Connection Options**

Name	Description	Values
Base Port	Base port used to establish a connection from the command line profiler to the host application during a <i>Profile</i> activity (both local and remote).	1-65535 (Default: 50152)
Maximum Ports	Maximum number of ports to try (starting from <i>Base Port</i> ) when attempting to connect to the host application.	1-100 (Default: 10)

## 12.4. Source Lookup

**Table 11 NVIDIA Nsight Compute Source Lookup Options**

Name	Description	Values
Program Source Locations	Set program source search paths. These paths are used to resolve CUDA-C source files on the Source page if the respective file	

Name	Description	Values
	cannot be found in its original location. Files which cannot be found are marked with a <i>File Not Found</i> error. See the <i>Ignore File Properties</i> option for files that are found but don't match.	
Ignore File Properties	Ignore file properties (e.g. timestamp, size) for source resolution. If this is disabled, all file properties like modification timestamp and file size are checked against the information stored by the compiler in the application during compilation. If a file with the same name exists on a source lookup path, but not all properties match, it won't be used for resolution (and a <i>File Mismatch</i> error will be shown).	Yes/No (Default)

## 12.5. Send Feedback

Table 12 NVIDIA Nsight Compute Send Feedback Options

Name	Description	Values
Collect Usage and Platform Data	Choose whether or not you wish to allow NVIDIA Nsight Compute to collect usage and platform data.	<ul style="list-style-type: none"> <li>▶ Yes</li> <li>▶ No (Default)</li> </ul>

# Chapter 13. PROJECTS

NVIDIA Nsight Compute uses *Project Files* to group and organize profiling reports. At any given time, only one project can be open in NVIDIA Nsight Compute. Collected reports are automatically assigned to the current project. Reports stored on disk can be assigned to a project at any time. In addition to profiling reports, related files such as notes or source code can be associated with the project for future reference.

Note that only references to reports or other files are saved in the project file. Those references can become invalid, for example when associated files are deleted, removed or not available on the current system, in case the project file was moved itself.

NVIDIA Nsight Compute uses the **ncu-proj** file extension for project files.

When no custom project is current, a *default project* is used to store e.g. the current [Connection Dialog](#) entries. To remove all information from the default project, you must close NVIDIA Nsight Compute and then delete the file from disk.

- ▶ On Windows, the file is located at `<USER>\AppData\Local\NVIDIA Corporation\NVIDIA Nsight Compute\`
- ▶ On Linux, the file is located at `<USER>/.local/share/NVIDIA Corporation/NVIDIA Nsight Compute/`
- ▶ On MacOSX, the file is located at `<USER>/Library/Application Support/NVIDIA Corporation/NVIDIA Nsight Compute/`

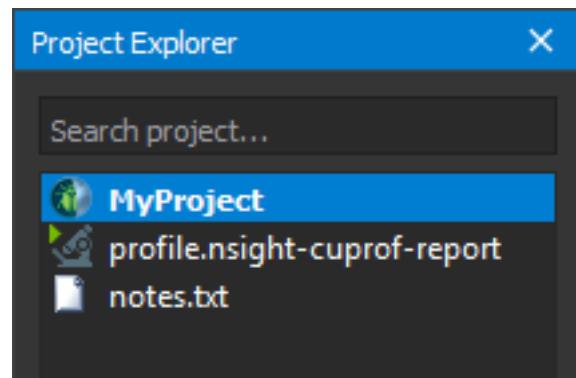
## 13.1. Project Dialogs

- ▶ **New Project**

Creates a new project. The project must be given a name, which will also be used for the project file. You can select the location where the project file should be saved on disk. Select whether a new directory with the project name should be created in that location.

## 13.2. Project Explorer

The *Project Explorer* window allows you to inspect and manage the current project. It shows the project name as well as all *Items* (profile reports and other files) associated with it. Right-click on any entry to see further actions, such as adding, removing or grouping items. Type in the *Search project* toolbar at the top to filter the currently shown entries.



# Chapter 14.

# VISUAL PROFILER TRANSITION GUIDE

This guide provides tips for moving from Visual Profiler to NVIDIA Nsight Compute. NVIDIA Nsight Compute tries to provide as much parity as possible with Visual Profiler's kernel profiling features, but some functionality is now covered by different tools.

## 14.1. Trace

NVIDIA Nsight Compute does not support tracing GPU or API activities on an accurate timeline. This functionality is covered by [NVIDIA Nsight Systems](#). In the [Interactive Profile Activity](#), the [API Stream](#) tool window provides a stream of recent API calls on each thread. However, since all tracked API calls are serialized by default, it does not collect accurate timestamps.

## 14.2. Sessions

Instead of sessions, NVIDIA Nsight Compute uses [Projects](#) to launch and gather connection details and collected reports.

- ▶ **Executable and Import Sessions**

Use the [Project Explorer](#) or the [Main Menu](#) to create a new project. Reports collected from the command line, i.e. using NVIDIA Nsight Compute CLI, can be opened directly using the main menu. In addition, you can use the Project Explorer to associate existing reports as well as any other artifacts such as executables, notes, etc., with the project. Note that those associations are only references; in other words, moving or deleting the project file on disk will not update its artifacts.

`nvprof` or command-line profiler output files, as well as Visual Profiler sessions, cannot be imported into NVIDIA Nsight Compute.

## 14.3. Timeline

Since trace analysis is now covered by Nsight Systems, NVIDIA Nsight Compute does not provide views of the application timeline. The [API Stream](#) tool window does show a per-thread stream of the last captured CUDA API calls. However, those are serialized and do not maintain runtime concurrency or provide accurate timing information.

## 14.4. Analysis

### ► Guided Analysis

All trace-based analysis is now covered by [NVIDIA Nsight Systems](#). This means that NVIDIA Nsight Compute does not include analysis regarding concurrent CUDA streams or (for example) UVM events. For per-kernel analysis, NVIDIA Nsight Compute provides recommendations based on collected performance data on the [Details Page](#). These rules currently require you to collect the required metrics via their sections up front, and do not support partial on-demand profiling.

To use the rule-based recommendations, enable the respective rules in the [Metric Selection](#). Before profiling, enable *Apply Rules* in the [Profile Options](#), or click the *Apply Rules* button in the report afterward.

### ► Unguided Analysis

All trace-based analysis is now covered by Nsight Systems. For per-kernel analysis, Python-based rules provide analysis and recommendations. See *Guided Analysis* above for more details.

### ► PC Sampling View

Source-correlated PC sampling information can now be viewed in the [Source Page](#). Aggregated warp states are shown on the [Details Page](#) in the *Warp State Statistics* section.

### ► Memory Statistics

Memory Statistics are located on the [Details Page](#). Enable the *Memory Workload Analysis* sections to collect the respective information.

### ► NVLink View

NVLink topology diagram and NVLink property table are located on the [Details Page](#). Enable the *NVLink Topology* and *NVLink Table* sections to collect the respective information.

Refer to the [Known Issues](#) section for the limitations related to NVLink.

### ► Source-Disassembly View

Source correlated with PTX and SASS disassembly is shown on the [Source Page](#). Which information is available depends on your application's compilation/JIT flags.

### ► GPU Details View

NVIDIA Nsight Compute does not automatically collect data for each executed kernel, and it does not collect any data for device-side memory copies. Summary information for all profiled kernel launches is shown on the [Summary Page](#). Comprehensive information on all collected metrics for all profiled kernel launches is shown on the [Raw Page](#).

- ▶ **CPU Details View**

CPU callstack sampling is now covered by [NVIDIA Nsight Systems](#).

- ▶ **OpenACC Details View**

OpenACC performance analysis with NVIDIA Nsight Compute is available to limited extent. OpenACC parallel regions are not explicitly recognized, but CUDA kernels generated by the OpenACC compiler can be profiled as regular CUDA kernels. See the [NVIDIA Nsight Systems](#) release notes to check its latest support status.

- ▶ **OpenMP Details View**

OpenMP performance analysis is not supported by NVIDIA Nsight Compute. See the [NVIDIA Nsight Systems](#) release notes to check its latest support status.

- ▶ **Properties View**

NVIDIA Nsight Compute does not collect CUDA API and GPU activities and their properties. Performance data for profiled kernel launches is reported (for example) on the [Details Page](#).

- ▶ **Console View**

NVIDIA Nsight Compute does not currently collect stdout/stderr application output.

- ▶ **Settings View**

Application launch settings are specified in the [Connection Dialog](#). For reports collected from the UI, launch settings can be inspected on the [Session Page](#) after profiling.

- ▶ **CPU Source View**

Source for CPU-only APIs is not available. Source for profiled GPU kernel launches is shown on the [Source Page](#).

## 14.5. Command Line Arguments

Please execute ncu-ui with the **-h** parameter within a shell window to see the currently supported command line arguments for the NVIDIA Nsight Compute UI.

To open a collected profile report with ncu-ui, simply pass the path to the report file as a parameter to the shell command.

# Chapter 15.

# VISUAL STUDIO INTEGRATION GUIDE

This guide provides information on using NVIDIA Nsight Compute within Microsoft Visual Studio, using the [NVIDIA Nsight Integration](#) Visual Studio extension, allowing for a seamless development workflow.

## 15.1. Visual Studio Integration Overview

NVIDIA Nsight Integration is a Visual Studio extension that allows you to access the power of NVIDIA Nsight Compute from within Visual Studio.

When NVIDIA Nsight Compute is installed along with NVIDIA Nsight Integration, NVIDIA Nsight Compute activities will appear under the NVIDIA 'Nsight' menu in the Visual Studio menu bar. These activities launch NVIDIA Nsight Compute with the current project settings and executable.

For more information about using NVIDIA Nsight Compute from within Visual Studio, please visit

- ▶ [NVIDIA Nsight Integration Overview](#)
- ▶ [NVIDIA Nsight Integration User Guide](#)

## **Notice**

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.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## **Trademarks**

NVIDIA and the NVIDIA logo are trademarks 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**

© 2018-2023 NVIDIA Corporation and affiliates. All rights reserved.

This product includes software developed by the Syncro Soft SRL (<http://www.sync.ro/>).