



# NSIGHT COMPUTE COMMAND LINE INTERFACE

v2021.2.1 | July 2021

**User Manual**



# TABLE OF CONTENTS

|  |           |
|--|-----------|
| <b>Chapter 1. Introduction.....</b>            | <b>1</b>  |
| <b>Chapter 2. Quickstart.....</b>              | <b>2</b>  |
| <b>Chapter 3. Usage.....</b>                   | <b>5</b>  |
| 3.1. Modes.....                                | 5         |
| 3.2. Multi-Process Support.....                | 5         |
| 3.3. Output Pages.....                         | 6         |
| 3.4. Profile Import.....                       | 6         |
| 3.5. Metrics and Units.....                    | 7         |
| 3.6. NVTX Filtering.....                       | 7         |
| <b>Chapter 4. Command Line Options.....</b>    | <b>10</b> |
| 4.1. General.....                              | 10        |
| 4.2. Launch.....                               | 11        |
| 4.3. Attach.....                               | 11        |
| 4.4. Profile.....                              | 12        |
| 4.5. Sampling.....                             | 20        |
| 4.6. File.....                                 | 20        |
| 4.7. Console Output.....                       | 21        |
| 4.8. Response File.....                        | 23        |
| 4.9. File Macros.....                          | 23        |
| <b>Chapter 5. Nvprof Transition Guide.....</b> | <b>24</b> |
| 5.1. Trace.....                                | 24        |
| 5.2. Metric Collection.....                    | 24        |
| 5.3. Metric Comparison.....                    | 26        |
| 5.4. Event Comparison.....                     | 30        |
| 5.5. Filtering.....                            | 32        |
| 5.6. Output.....                               | 33        |
| 5.7. Launch and Attach.....                    | 33        |

## LIST OF TABLES

|          |   |    |
|----------|---|----|
| Table 1  | General Command Line Options .....  | 10 |
| Table 2  | Launch Command Line Options .....   | 11 |
| Table 3  | Attach Command Line Options .....   | 11 |
| Table 4  | Profile Command Line Options .....  | 12 |
| Table 5  | Sampling Command Line Options .....   | 20 |
| Table 6  | File Command Line Options .....   | 20 |
| Table 7  | Console Output Command Line Options .....   | 21 |
| Table 8  | Macro Expansions .....  | 23 |
| Table 9  | Metrics Mapping Table from CUPTI to PerfWorks .....   | 26 |
| Table 10 | Events Mapping Table from CUPTI Events to PerfWorks Metrics for Compute<br>Capability >= 7.0..... | 30 |



# Chapter 1.

## INTRODUCTION

NVIDIA Nsight Compute CLI (ncu) provides a non-interactive way to profile applications from the command line. It can print the results directly on the command line or store them in a report file. It can also be used to simply launch the target application (see [General](#) for details) and later attach with NVIDIA Nsight Compute or another ncu instance.

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

# Chapter 2.

## QUICKSTART

### 1. Launch the target application with the command line profiler

The command line profiler launches the target application, instruments the target API, and collects profile results for the specified kernels. The CLI executable is called `ncu`. 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 `target\windows-desktop-win7-x64` on Windows or `target/linux-desktop-glibc_2_11_3-x64` on Linux. By default, NVIDIA Nsight Compute is installed in `/usr/local/cuda-<cuda-version>/NsightCompute-<version>` on Linux and in `C:\Program Files\NVIDIA Corporation\Nsight Compute <version>` on Windows.

To collect the default set of data for all kernel launches in the target application, launch:

```
$ ncu -o profile CuVectorAddMulti.exe
```

The application runs in instrumented mode and for each kernel launch, a profile result is created. The results are written by default to `profile.nsisht-cuprof`. Each output from the compute profiler starts with `==PROF==`. The other lines are output from the application itself. For each profiled kernel, the name of the kernel function and the progress of data collection is shown. To collect all requested profile information, it may be required to replay the kernels multiple times. The total number of replay passes per kernel is shown after profiling has completed.

```
[Vector addition of 1144477 elements]
==PROF== Connected to process 5268
Copy input data from the host memory to the CUDA device
CUDA kernel launch A with 4471 blocks of 256 threads
==PROF== Profiling "vectorAdd_A" - 1: 0%....50%....100% - 46 passes
CUDA kernel launch B with 4471 blocks of 256 threads
==PROF== Profiling "vectorAdd_B" - 2: 0%....50%....100% - 46 passes
Copy output data from the CUDA device to the host memory
Done
==PROF== Disconnected from process 5268
==PROF== Report: profile.ncu-rep
```

### 2. Customizing data collection

Options are available to specify for which kernels data should be collected. **-c** limits the number of kernel launches collected. **-s** skips the given number of kernels before data collection starts. **-k** allows you to filter the kernels by a regex match of their names. **--kernel-id** allows you to filter kernels by context, stream, name and invocation, similar to `nvprof`.

To limit what should be collected for each kernel launch, specify the exact \*.section (files) by their identifier using **--section**. Each section file defines a set of metrics to be collected, grouped logically to solve a specific performance question. By default, the sections associated with the default set are collected. Use **--list-sets** to see the list of currently available sets. Use **--list-sections** to see the list of currently available sections. The default search directory and location of pre-defined section files is also called **sections/**. See the [Profiling Guide](#) for more details.

Alternatively, you can collect a set of individual metrics using **--metrics**. The available metrics can be queried using **--query-metrics**. For an explanation of the naming conventions and structuring of metrics used on Volta and newer architectures, see [Perfworks Metrics API](#).

On Volta and newer GPUs, most metrics are named using a base name and various suffixes, e.g. `sm__throughput.avg.pct_of_peak_sustained_elapsed`. The base name is `sm__throughput` and the suffix is `avg.pct_of_peak_sustained_elapsed`. This is because most metrics follow the same structure and have the same set of suffixes. You need to pass the full name to NVIDIA Nsight Compute when selecting a metric for profiling. Use **--query-metrics-mode suffix --metrics <metrics list>** to see the full names for the chosen metrics.

### 3. Changing command line output

By default, a temporary file is used to store profiling results, and data is printed to the command line. To permanently store the profiler report, use **-o** to specify the output filename.

Besides storing results in a report file, the command line profiler can print results using different pages. Those pages correspond to the respective pages in the UI's report. By default, the [Details page](#) is printed, if no explicit output file is specified. To select a different page or print in addition to storing in an explicit file, use the **--page=<Page>** command. Currently, the following pages are supported: **details**, **raw**.

Use **--csv** to make any output comma separated and easier to process further. See [Console Output](#) for further options, e.g. summary views.

### 4. Open the report in the UI

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. In the UI window, close the *Connection* dialog and open the report file through *File > Open*, by dragging the report file into NVIDIA Nsight Compute.

You can also specify the report file as a command line parameter to the executable, i.e. as `ncu-ui <MyReport.ncu-rep>`. Alternatively, when using NVIDIA Nsight

Compute CLI on a platform with host support, `--open-in-ui` can be used directly with `ncu` to open a collected report in the user interface.

The report opens in a new document window. For more information about the report, see the [Profiler Report](#) for collecting profile information through NVIDIA Nsight Compute.



# Chapter 3.

## USAGE

### 3.1. Modes

Modes change the fundamental behavior of the command line profiler. Depending on which mode is chosen, different [Command Line Options](#) become available. For example, [Launch](#) is invalid if the *Attach* mode is selected.

- ▶ **Launch-and-attach:** The target application is launched on the local system with the tool's injection libraries. Depending on which profiling options are chosen, selected kernels in the application are profiled and the results printed to the console or stored in a report file. The tool exits once the target application finishes or crashes, and once all results are processed.

This is the default, and the only mode that supports profiling of child processes on selected platforms.

- ▶ **Launch:** The target application is launched on the local system with the tool's injection libraries. As soon as the first intercepted API call is reached (commonly `cuInit()`), all application threads are suspended. The application now expects a tool to attach for profiling. You can attach using NVIDIA Nsight Compute or using the command line profiler's *Attach* mode.
- ▶ **Attach:** The tool tries to connect to a target application previously launched using NVIDIA Nsight Compute or using the command line profiler's *Launch* mode. The tool can attach to a target on the local system or using a remote connection.

### 3.2. Multi-Process Support

NVIDIA Nsight Compute CLI supports profiling multi-process applications on the following platforms: x86\_64 Windows, x86\_64 Linux, DRIVE OS Linux, DRIVE OS QNX, PowerPC. See the [Launch](#) options on how to enable this feature.

On x86\_64 Windows, NVIDIA Nsight Compute CLI supports profiling 64-bit processes launched from 32-bit applications by default. On x86\_64 Linux, launching from 32-bit applications requires you to enable the **support-32bit** option, and the required 32-bit

libraries must be installed on your system. On DRIVE OS Linux, DRIVE OS QNX and PowerPC, tracking of 32-bit applications is not supported. *Profiling* of 32-bit processes is not supported on any platform.

### 3.3. Output Pages

The command line profiler supports printing results to the console using various pages. Each page has an equivalent in NVIDIA Nsight Compute's *Profiler Report*. In the command line profiler, they are slightly adapted to fit console output. To select a page, use the `--page` option. By default, the details page is used. Note that if `--page` is not used but `--export` is, no results will be printed to the console.

- **Details:** This page represents NVIDIA Nsight Compute's *Details* page. For every profiled kernel launch, each collected is printed as section as a three-column table, followed by any rule results applied to this section. Rule results not associated with any section are printed after the kernel's sections.

The first section table column shows the metric name. If the metric was given a label in the section, it is used instead. The second column shows the metric unit, if available. The third column shows the unit value. Both metric unit and value are automatically adjusted to the most fitting order of magnitude. By default, only metrics defined in section headers are shown. This can be changed by passing the `--details-all` option on the command line.

Some metrics will show multiple values, separated by ";", e.g. `memory_l2_transactions_global Kbytes 240; 240; 240; 240; 240`. Those are instanced metrics, which have one value per represented instance. An instance can be a streaming multiprocessor, an assembly source line, etc.

- **Raw:** This page represents NVIDIA Nsight Compute's *Raw* page. For every profiled kernel launch, each collected metric is printed as a three-column table. Besides metrics from sections, this includes automatically collected metrics such as device attributes and kernel launch information.

The first column shows the metric name. The second and third columns show the metric unit and value, respectively. Both metric unit and value are automatically adjusted to the most fitting order of magnitude. No unresolved regex;, group;, or breakdown: metrics are included.

### 3.4. Profile Import

Using the `--import` option, saved reports can be imported into the command line profiler. When using this flag, most other options are not available, except for certain result filtering options. They are marked as such in the [Profile options](#) table.

## 3.5. Metrics and Units

When available and applicable, metrics are shown along with their unit. This is to make it apparent if a metric represents cycles, threads, bytes/s, and so on.

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 changed using a command line option, see [Console Output](#) options for details.

## 3.6. NVTX Filtering

**--nvtx-include** <configuration> **--nvtx-exclude** <configuration>

These options are used to profile only those kernels which satisfy the conditions mentioned in the configuration. Through these options, you can choose which kernel falls into a specific range or collection of ranges.

You can use both options multiple times, mentioning all the **--nvtx-include** configurations followed by all **--nvtx-exclude** configurations. NVTX filtering requires **--nvtx** option.

NVTX ranges are of two types: NvtxRangeStart/End and NvtxRangePush/Pop. The configuration syntax for both the types are briefly described below.

### ► Start-End Ranges

| Quantifier | Description  | Example                                      |
|------------|--|--|
| ,          | Delimiter between range names                                    | Range A,Range B<br>Range B,Range A,Range C   |
| @          | Specify domain name. If not mentioned, assuming <default domain> | Domain A@Range A<br>Domain B@Range B,Range Z |

```
ncu --nvtx --nvtx-include "Domain A@Range A" CuNvtx.exe
```

The kernels wrapped inside 'Range A' of 'Domain A' in the application are profiled.

```
ncu --nvtx --nvtx-include "Range A,Range B" CuNvtx.exe
```

The kernels wrapped inside both ranges, 'Range A' and 'Range B' of '<default domain>' in the application are profiled.

```
ncu --nvtx --nvtx-include "Range A" --nvtx-include "Range B" CuNvtx.exe
```

The kernels wrapped inside ranges, 'Range A' or 'Range B' of '<default domain>' in the application are profiled.

```
ncu --nvtx --nvtx-exclude "Range A" CuNvtx.exe
```

All the kernels in the application are profiled except those which are wrapped inside 'Range A' of '<default domain>'.

```
ncu --nvtx --nvtx-include "Range B"--nvtx-exclude "Range A" CuNvtx.exe
```

The kernels wrapped inside only 'Range B' and not 'Range A' of '<default domain>' in the application are profiled.

### ► Push-Pop Ranges

| Quantifier | Description  | Example  |
|------------|--|--|
| /          | Delimiter between range names                                    | Range A/Range B<br>Range A/*/Range B<br>Range A/ |
| [          | Range is at the bottom of the stack                              | [Range A<br>[Range A/+ /Range Z                  |
| ]          | Range is at the top of the stack                                 | Range Z]<br>Range C/* /Range Z]                  |
| +          | Only one range between the two other ranges                      | Range B/+ /Range D                               |
| *          | Zero or more range(s) between the two other ranges               | Range B/* /Range Z                               |
| @          | Specify domain name. If not mentioned, assuming <default domain> | Domain A@Range A<br>Domain B@Range A/* /Range Z] |

```
ncu --nvtx --nvtx-include "Domain A@Range A/" CuNvtx.exe
```

The kernels wrapped inside 'Range A' of 'Domain A' in the application are profiled.

```
ncu --nvtx --nvtx-include "[Range A" CuNvtx.exe
```

The kernels wrapped inside 'Range A' of '<default domain>' where 'Range A' is at the bottom of the stack in the application are profiled.

```
ncu --nvtx --nvtx-include "Range A/*/Range B" CuNvtx.exe
```

The kernels wrapped inside 'Range A' and 'Range B' of '<default domain>' with zero or many ranges between them in the application are profiled.

```
ncu --nvtx --nvtx-exclude "Range A/*/Range B" CuNvtx.exe
```

All the kernels in the application are profiled except those which are wrapped inside 'Range A' and 'Range B' of '<default domain>' with zero or many ranges between them.

```
ncu --nvtx --nvtx-include "Range A/" --nvtx-exclude "Range B]" CuNvtx.exe
```

The kernels wrapped inside only 'Range A' of '<default domain>' but not inside 'Range B' at the top of the stack in the application are profiled.

### ► Additional Information

```
--nvtx-include DomainA@RangeA,DomainB@RangeB //Not a valid config
```

In a single NVTX configuration, multiple ranges with regard to a single domain can be specified. Mentioning ranges from different domains inside a single NVTX config is not supported.

```
--nvtx-include "Range A\[i\]"
```

Quantifiers '@' ';' '[' ']' '/' '\*' '+' can be used in range names using prefix '\\'. The kernels wrapped inside 'Range A[i]' of '<default domain>' in the application are profiled.

```
--nvtx-include "Range A" //Start/End configuration
--nvtx-inlcude "Range A/" //Push/Pop configuration
--nvtx-inlcude "Range A]" //Push/Pop configuration
```

Including/Excluding only single range for Push/Pop configuration without specifying stack frame position '[' or ']', use '/' quantifier at the end.

```
--nvtx-include "Range A/*/RangeB"
```

The order in which you mention Push/Pop configurations is important. In the above example, 'Range A' should be below 'Range B' in the stack of ranges so that the kernel is profiled.

NVTX filtering honors `cudaProfilerStart()` and `cudaProfilerStop()`. There is no support for ranges with no name.

# Chapter 4.

## COMMAND LINE OPTIONS

For long command line options, passing a unique initial substring can be sufficient.

### 4.1. General

Table 1 General Command Line Options

| Option          | Description  | Default           |
|-----------------|--|-------------------|
| h,help          | Show help message  |                   |
| v,version       | Show version information   |                   |
| mode            | Select the mode of interaction with the target application <ul style="list-style-type: none"><li>▶ <b>launch-and-attach:</b> Launch the target application and immediately attach for profiling.</li><li>▶ <b>launch:</b> Launch the target application and suspend in the first intercepted API call, wait for tool to attach.</li><li>▶ <b>attach:</b> Attach to a previously launched application to which no other tool is attached.</li></ul> | launch-and-attach |
| p,port          | Base port used for connecting to target applications   | 49152             |
| max-connections | Maximum number of ports for connecting to target applications  | 64                |

## 4.2. Launch

Table 2 Launch Command Line Options

| Option            | Description   | Default          |
|-------------------|---|------------------|
| check-exit-code   | Check the application exit code and print an error if it is different than 0. If set, <code>--replay-mode</code> application will stop after the first pass if the exit code is not 0.  | yes              |
| injection-path-64 | Override the default path for the injection libraries. The injection libraries are used by the tools to intercept relevant APIs (like CUDA or NVTX).  |                  |
| nvtx              | Enable NVTX support for tools.  |                  |
| target-processes  | Select the processes you want to profile. Available modes are: <ul style="list-style-type: none"> <li>▶ <b>application-only</b> Profile only the root application process.</li> <li>▶ <b>all</b> Profile the application and all its child processes.</li> </ul> This option is only available for Linux and Windows targets. | application-only |
| support-32bit     | Support profiling processes launched from 32-bit applications. This option is only available on x86_64 Linux. On Windows, tracking 32-bit applications is enabled by default.   | no               |

## 4.3. Attach

Table 3 Attach Command Line Options

| Option   | Description   | Default   |
|----------|---|-----------|
| hostname | Set the hostname or IP address for connecting to the machine on which the target application is running. When attaching to a local target application, use 127.0.0.1. | 127.0.0.1 |

## 4.4. Profile

Table 4 Profile Command Line Options

| Option    | Description  | Default/Examples  |
|-----------|--|---|
| devices   | List the GPU devices to enable profiling on, separated by comma. <sup>1</sup>  | All devices<br><br><b>Examples</b><br><br><code>--devices 0,2</code>  |
| kernel-id | <p>Set the identifier to use for matching kernels. If the kernel does not match the identifier, it will be ignored for profiling.</p> <p>The identifier must be of the following format:<br/> <code>context-id:stream-id:<br/> [name-operator:]kernel-name:invocation-nr</code></p> <ul style="list-style-type: none"> <li>▶ <b>context-id</b> is the CUDA context ID or regular expression to match the NVTX name.</li> <li>▶ <b>stream-id</b> is the CUDA stream ID or regular expression to match the NVTX name.</li> <li>▶ <b>name-operator</b> is an optional operator to <i>kernel-name</i>. Currently, only <i>regex</i> is the only supported operator.</li> <li>▶ <b>kernel-name</b> is the expression to match the kernel name. By default, this is a full, literal match to what is specified by <code>--kernel-name-base</code>. When specifying the optional <i>regex</i> name operator, this is a partial regular expression match to what is specified by <code>--kernel-name-base</code>.</li> <li>▶ <b>invocation-nr</b> is the N'th invocation of this kernel function. Multiple invocations can also be specified using regular expressions.</li> </ul> | <b>Examples</b><br><code>--kernel-id ::foo:2</code> For kernel "foo", match the second invocation.<br><code>--kernel-id :::". *5 3"</code><br>For all kernels, match the third invocation, and all for which the invocation number ends in "5".<br><code>--kernel-id ::regex: ^.*foo\$</code> : Match all kernels ending in "foo".<br><code>--kernel-id ::regex: ^(?!foo)</code> : Match all kernels except those starting with "foo". Note that depending on your OS and shell, you might need to quote the expression, e.g. using single quotes in Linux <i>bash</i> : <code>--kernel-id ::regex: '^(?!foo)'</code><br><code>--kernel-id 1:2::7</code> Match all seventh kernel invocations on context 1, stream 2. |

<sup>1</sup> This filtering option is available when using `--import`.



| Option            | Description  | Default/Examples   |
|-------------------|--|--|
|                   | <p>If the context/stream ID is a positive number, it will be strictly matched against the CUDA context/stream ID. Otherwise it will be treated as a regular expression and matched against the context/stream name specified using the NVTX library.</p> <p>1</p>  |  |
| kernel-regex      | <p>Set the regular expression to use when matching kernel names. If the kernel name does not match the expression, it will be ignored for profiling. On shells that recognize regular expression symbols as special characters (e.g. Linux bash), the expression needs to be escaped with quotes, e.g. <code>--kernel-regex ". *Foo"</code>. <b>This option is deprecated and replaced by <code>--kernel-name</code>.</b> 1</p>  | <p><b>Examples</b></p> <p><code>-k foo</code> Match all kernels that include the string "foo", e.g. "foo" and "fooBar".</p> <p><code>-k "^foo\$" </code> Match all kernels named exactly "foo".</p> <p><code>-k "foo bar" </code> Match all kernels including the strings "foo" or "bar", e.g. "foo", "foobar", "_bar2".</p>       |
| k, kernel-name    | <p>Set the expression to use when matching kernel names.</p> <ul style="list-style-type: none"> <li>▶ <b>&lt;kernel name&gt;</b> Set the kernel name for an exact match.</li> <li>▶ <b>regex:&lt;expression&gt;</b> Set the regex to use for matching the kernel name. On shells that recognize regular expression symbols as special characters (e.g. Linux bash), the expression needs to be escaped with quotes, e.g. <code>--kernel-name regex: ". *Foo"</code>.</li> </ul> <p>If the kernel name or the provided expression do not match, it will be ignored for profiling. 1</p> | <p><b>Examples</b></p> <p><code>-k foo</code> Match all kernels named exactly "foo".</p> <p><code>-k regex:foo</code> Match all kernels that include the string "foo", e.g. "foo" and "fooBar".</p> <p><code>-k regex:"foo bar" </code> Match all kernels including the strings "foo" or "bar", e.g. "foo", "foobar", "_bar2".</p> |
| kernel-regex-base | <p>Set the basis for <code>--kernel-name</code>, and <code>--kernel-id</code> kernel-name. Options are:</p> <ul style="list-style-type: none"> <li>▶ <b>function:</b> Function name without parameters, templates etc. e.g. <code>dmatrixmul</code></li> <li>▶ <b>demangled:</b> Demangled function name, including parameters, templates, etc. e.g. <code>dmatrixmul(float*,int,int)</code></li> </ul>  | function   |

| Option                   | Description   | Default/Examples |
|--------------------------|---|------------------|
|                          | <ul style="list-style-type: none"> <li>► <b>mangled:</b> Mangled function name. e.g. <code>_Z10dmatrixmulPfiiS_iiS_</code></li> </ul> <p>This option is deprecated and replaced by <code>--kernel-name-base</code>. 1</p>   |                  |
| kernel-name-base         | <p>Set the basis for <code>--kernel-name</code>, and <code>--kernel-id</code> kernel-name. Options are:</p> <ul style="list-style-type: none"> <li>► <b>function:</b> Function name without parameters, templates etc. e.g. <code>dmatrixmul</code></li> <li>► <b>demangled:</b> Demangled function name, including parameters, templates, etc. e.g. <code>dmatrixmul(float*,int,int)</code></li> <li>► <b>mangled:</b> Mangled function name. e.g. <code>_Z10dmatrixmulPfiiS_iiS_</code></li> </ul> <p>1</p> | function         |
| c,launch-count           | Limit the number of profiled kernel launches. The count is only incremented for launches that match the kernel filters. 1   |                  |
| s,launch-skip            | Set the number of kernel launches to skip before starting to profile kernels. The number takes into account only launches that match the kernel filters. 1  | 0                |
| launch-skip-before-match | Set the number of kernel launches to skip before starting to profile. The count is incremented for all launches, regardless of the kernel filters. 1  | 0                |
| kill                     | <p>Terminate the target application when the requested <code>--launch-count</code> was profiled. Allowed values:</p> <ul style="list-style-type: none"> <li>► on/off</li> <li>► yes/no</li> </ul>   | no               |
| replay-mode              | <p>Mechanism used for replaying a kernel launch multiple times to collect all requested profiling data:</p> <ul style="list-style-type: none"> <li>► <b>kernel:</b> Replay individual kernel launches "transparently" during</li> </ul>   | kernel           |

| Option            | Description  | Default/Examples |
|-------------------|--|------------------|
|                   | <p>the execution of the application.</p> <ul style="list-style-type: none"> <li>▶ <b>application:</b> Relaunch the entire application multiple times. Requires deterministic program execution.</li> </ul>   |                  |
| app-replay-buffer | <p>Application replay buffer location.</p> <ul style="list-style-type: none"> <li>▶ <b>file:</b> Replay pass data is buffered in a temporary file. The report is created after profiling completed. This mode is more scalable, as the amount of required memory does not scale with the number of profiled kernels.</li> <li>▶ <b>memory:</b> Replay pass data is buffered in memory, and the report is created while profiling. This mode can result in better performance if the filesystem is slow, but the amount of required memory scales with the number of profiled kernels.</li> </ul>   | file             |
| app-replay-match  | <p>Application replay kernel matching strategy. For all options, kernels are matched on a per-process and per-device (GPU) basis. Below options are used to configure the applied strategy in more detail.</p> <ul style="list-style-type: none"> <li>▶ <b>name:</b> Kernels are matched in the following order: 1. (mangled) name, 2. order of execution</li> <li>▶ <b>grid:</b> Kernels are matched in the following order: 1. (mangled) name, 2. CUDA grid/block size, 3. order of execution</li> <li>▶ <b>all:</b> Kernels are matched in the following order: 1. (mangled) name, 2. CUDA grid/block size, 3. CUDA context ID, 4. CUDA stream ID, 5. order of execution</li> </ul> | all              |
| list-sets         | List all section sets found in the searched section folders and exit. For each set, the associated sections are shown, as well as  |                  |

| Option                   | Description   | Default/Examples   |
|--------------------------|---|--|
|                          | the estimated number of metrics collected as part of this set. This number can be used as an estimate of the relative profiling overhead per kernel launch of this set.   |  |
| set                      | Identifier of section set to collect. If not specified, the default set is collected. The full set of sections can be collected with <code>--set full</code> .  | If no <code>--set</code> option is given, the default set is collected. If not specified and <code>--section</code> or <code>--metrics</code> are used, no sets are collected. Use <code>--list-sets</code> to see which set is the default. |
| list-sections            | List all sections found in the searched section folders and exit.   |  |
| section                  | <p>Add a section identifier to collect in one of the following ways:</p> <ul style="list-style-type: none"> <li>▶ <code>&lt;section identifier&gt;</code><br/>Set the section identifier for an exact match.</li> <li>▶ <code>regex:&lt;expression&gt;</code><br/>Regular expression allows matching full section identifier. For example, <code>.*Stats</code>, matches all sections ending with 'Stats'. On shells that recognize regular expression symbols as special characters (e.g. Linux bash), the expression needs to be escaped with quotes, e.g. <code>--section "regex:.*Stats"</code>.</li> </ul> <p>This option is ignored when used with <code>--import</code> and <code>--page raw. 1</code></p> | If no <code>--section</code> options are given, the sections associated with the default set are collected. If no sets are found, all sections are collected.  |
| section-folder           | Add a non-recursive search path for .section files. Section files in this folder will be made available to the <code>--section</code> option.   | If no <code>--section-folder</code> options are given, the <code>sections</code> folder is added by default.   |
| section-folder-recursive | Add a recursive search path for .section files. Section files in this folder and all folders below will be made available to the <code>--section</code> option.   | If no <code>--section-folder</code> options are given, the <code>sections</code> folder is added by default.   |
| list-rules               | List all rules found in the searched section folders and exit.  |  |
| apply-rules              | Apply active and applicable rules to each profiling result. Use   | yes  |

| Option             | Description  | Default/Examples   |
|--------------------|--|--|
|                    | <p><code>--rule</code> to limit which rules to apply. Allowed values:</p> <ul style="list-style-type: none"> <li>▶ on/off</li> <li>▶ yes/no</li> </ul>   |  |
| rule               | Add a rule identifier to apply. Implies <code>--apply-rules yes</code> .   | If no <code>--rule</code> options are given, all applicable rules in the <code>sections</code> folder are applied. |
| import-source      | <p>If available from <code>-lineinfo</code>, correlated CUDA source files are permanently imported into the report. Allowed values:</p> <ul style="list-style-type: none"> <li>▶ on/off</li> <li>▶ yes/no</li> </ul>   | no   |
| list-metrics       | List all metrics collected from active sections. If the list of active sections is restricted using the <code>--section</code> option, only metrics from those sections will be listed.  |  |
| query-metrics      | Query available metrics for the devices on system. Use <code>--devices</code> and <code>--chips</code> to filter which devices to query. Note that by default, listed metric names need to be appended a valid suffix in order for them to become valid metrics. See <code>--query-metrics-mode</code> for how to get the list of valid suffixes, or check the <a href="#">Kernel Profiling Guide</a> .  |  |
| query-metrics-mode | <p>Set the mode for querying metrics. Implies <code>--query-metrics</code>. Available modes:</p> <ul style="list-style-type: none"> <li>▶ <b>base</b>: Only the base names of the metrics.</li> <li>▶ <b>suffix</b>: Suffix names for the base metrics. This gives the list of all metrics derived from the base metrics. Use <code>--metrics</code> to specify the base metrics to query.</li> <li>▶ <b>all</b>: Full names for all metrics. This gives the list of all base metrics and their suffix metrics.</li> </ul> | base   |
| metrics            | Specify all metrics to be profiled, separated by comma. If no <code>--section</code> options are given, only the temporary section   |  |

| Option                      | Description  | Default/Examples                                    |
|-----------------------------|--|---|
|                             | <p>containing all metrics listed using this option is collected. If <code>--section</code> options are given in addition to <code>--metrics</code>, all metrics from those sections and from <code>--metrics</code> are collected.</p> <p>Names passed to this option support the following prefixes:<br/> <code>regex:&lt;expression&gt;</code> expands to all metrics that partially match the expression. Enclose the regular expression in <code>^...\$</code> to force a full match.<br/> <code>group:&lt;name&gt;</code> lists all metrics of the metric group with that name. See section files for valid group names.<br/> <code>breakdown:&lt;metric&gt;</code> expands to the input metrics of the high-level throughput metric. If the specified metric does not support a breakdown, no metrics are added.</p> <p>If a metric requires a suffix to be valid, and neither <code>regex:</code> nor <code>group:</code> are used, this option automatically expands the name to all available first-level sub-metrics.</p> <p>When importing a report <code>:group</code> and <code>:breakdown</code> are not supported.</p> <p>1</p> |   |
| list-chips                  | List all supported chips that can be used with <code>--chips</code> .  |   |
| chips                       | Specify the chips for querying metrics, separated by comma.  | <b>Examples</b><br><code>--chips gv100,tu102</code> |
| profile-from-start          | Set if application should be profiled from its start. Allowed values: <ul style="list-style-type: none"> <li>▶ on/off</li> <li>▶ yes/no</li> </ul>   | yes   |
| disable-profiler-start-stop | Disable profiler start/stop. When enabled, <code>cu(da)ProfilerStart/Stop</code> API calls are ignored.  |   |
| quiet                       | Suppress all profiling output.   |   |
| cache-control               | Control the behavior of the GPU caches during profiling. Allowed values:   | all   |

| Option        | Description   | Default/Examples |
|---------------|---|------------------|
|               | <ul style="list-style-type: none"> <li>▶ <b>all:</b> 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.</li> <li>▶ <b>none:</b> 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.</li> </ul> |                  |
| clock-control | <p>Control the behavior of the GPU clocks during profiling. Allowed values:</p> <ul style="list-style-type: none"> <li>▶ <b>base:</b> GPC and memory clocks are locked to their respective base frequency during profiling. This has no impact on thermal throttling. Note that actual clocks might still vary, depending on the level of driver support for this feature. As an alternative, use <code>nvidia-smi</code> to lock the clocks externally and set this option to <b>none</b>.</li> <li>▶ <b>none:</b> No GPC or memory frequencies are changed during profiling.</li> </ul>   | base             |
| nvtx-include  | Adds an include statement to the <a href="#">NVTX filter</a> , which allows selecting kernels to profile based on NVTX ranges. <a href="#">1</a>  |                  |
| nvtx-exclude  | Adds an exclude statement to the <a href="#">NVTX filter</a> , which allows   |                  |

| Option | Description  | Default/Examples |
|--------|--|------------------|
|        | selecting kernels to profile based on NVTX ranges. <a href="#">1</a> |                  |

## 4.5. Sampling

Table 5 Sampling Command Line Options

| Option               | Description  | Default      |
|----------------------|--|--------------|
| sampling-interval    | Set the sampling period in the range of [0..31]. The actual frequency is $2^{(5 + \text{value})}$ cycles. If set to 'auto', the profiler tries to automatically determine a high sampling frequency without skipping samples or overflowing the output buffer. | auto         |
| sampling-max-passes  | Set maximum number of passes used for sampling (see the <a href="#">Kernel Profiling Guide</a> for more details on profiling overhead).  | 5            |
| sampling-buffer-size | Set the size of the device-sided allocation for samples in bytes.  | 32*1024*1024 |

## 4.6. File

Table 6 File Command Line Options

| Option   | Description  | Default  |
|----------|--|--|
| log-file | Send all tool output to the specified file, or one of the standard channels. The file will be overwritten. If the file doesn't exist, a new one will be created. "stdout" as the whole file name indicates standard output channel (stdout). "stderr" as the whole file name indicates standard error channel (stderr)." | If <code>--log-file</code> is not set, profile results will be printed on the console.   |
| o,export | Set the output file for writing the profile report. If not set, a temporary file will be used which is removed afterwards. The specified name supports macro expansion. See <a href="#">File Macros</a> for more details.  | If <code>--export</code> is set and no <code>--page</code> option is given, no profile results will be printed on the console. |



| Option                 | Description  | Default   |
|------------------------|--|---|
| f,force-overwrite      | Force overwriting all output files.  | By default, the profiler won't overwrite existing output files and show an error instead. |
| i,import               | Set the input file for reading the profile results.  |   |
| open-in-ui             | Open report in UI instead of showing result on terminal. (Only available on host platforms)  |   |
| section-folder-restore | Restores stock files to the default section folder or the folder specified by an accompanying <code>--section-folder</code> option. If the operation will overwrite modified files then the <code>--force-overwrite</code> option is required. |   |

## 4.7. Console Output

Table 7 Console Output Command Line Options

| Option      | Description   | Default  |
|-------------|---|--|
| csv         | Use comma-separated values as console output. Implies <code>--print-units</code> base by default.   |  |
| page        | <p>Select the report page to print console output for. Available pages are:</p> <ul style="list-style-type: none"> <li>► <b>details</b> Show results grouped as sections, include rule results. Some metrics that are collected by default (e.g. device attributes) are omitted if not specified explicitly in any section or using <code>--metrics</code>.</li> <li>► <b>raw</b> Show all collected metrics by kernel launch.</li> </ul> | <b>details</b> . If no <code>--page</code> option is given and <code>--export</code> is set, no results are printed to the console output. |
| details-all | Include all section metrics on details page.  |  |
| print-units | <p>Select the mode for scaling of metric units. Available modes are:</p> <ul style="list-style-type: none"> <li>► <b>auto</b> Show all metrics automatically scaled to the most fitting order of magnitude.</li> </ul>  | auto   |

| Option                              | Description   | Default   |
|-------------------------------------|---|-----------|
|                                     | <ul style="list-style-type: none"> <li>► <b>base</b> Show all metrics in their base unit.</li> </ul> Replaces deprecated option <code>--units</code> .  |           |
| <code>print-fp</code>               | Show all numeric metrics in the console output as floating point numbers. Replaces deprecated option <code>--fp</code> .  | false     |
| <code>print-kernel-base</code>      | Set the basis for kernel name output. See <code>--kernel-regex-base</code> for options. Replaces deprecated option <code>--kernel-base</code> .   | demangled |
| <code>print-metric-instances</code> | Set output mode for metrics with instance values: <ul style="list-style-type: none"> <li>► <b>none</b> Only show GPU aggregate value.</li> <li>► <b>values</b> Show GPU aggregate followed by all instance values.</li> </ul>   | none      |
| <code>print-nvtx-rename</code>      | Select how NVTX should be used for renaming: <ul style="list-style-type: none"> <li>► <b>none</b> Don't use NVTX for renaming.</li> <li>► <b>kernel</b> Rename kernels with the most recent enclosing NVTX push/pop range.</li> </ul>   | none      |
| <code>print-rule-details</code>     | Print additional details for rule results, such as the triggering metrics. Currently has no effect in CSV mode.   | false     |
| <code>print-summary</code>          | Select the summary output mode. Available modes are: <ul style="list-style-type: none"> <li>► <b>none</b> No summary.</li> <li>► <b>per-gpu</b> Summary for each GPU.</li> <li>► <b>per-kernel</b> Summary for each kernel type.</li> <li>► <b>per-nvtx</b> Summary for each NVTX context.</li> </ul> Replaces deprecated option <code>--summary</code> . | none      |

## 4.8. Response File

Response files can be specified by adding **@FileName** to the command line. The file name must immediately follow the @ character. The content of each response file is inserted in place of the corresponding response file option.

## 4.9. File Macros

The file name specified with option **-o** or **--export** supports the following macro expansions. Occurrences of these macros in the report file name are replaced by the corresponding character sequence. If not specified otherwise, the macros cannot be used as part of the file path.

Table 8 Macro Expansions

| Macro        | Description   |
|--------------|---|
| %h           | Expands to the host name of the machine on which the command line profiler is running.  |
| %q{ENV_NAME} | Expands to the content of the variable with the given name <b>ENV_NAME</b> from the environment of the command line profiler.   |
| %p           | Expands to the process ID of the command line profiler.   |
| %i           | Expands to the lowest unused positive integer number that guarantees the resulting file name is not yet used. This macro can only be used once in the output file name. |
| %%           | Expands to a single % character in the output file name. This macro can be used in the file path and the file name.   |

# Chapter 5.

## NVPROF TRANSITION GUIDE

This guide provides tips for moving from nvprof to NVIDIA Nsight Compute CLI. NVIDIA Nsight Compute CLI tries to provide as much feature and usage parity as possible with nvprof, but some features are now covered by different tools and some command line options have changed their name or meaning.

### 5.1. Trace

- ▶ **GPU and API trace**

NVIDIA Nsight Compute CLI does not support any form of tracing GPU or API activities. This functionality is covered by [NVIDIA Nsight Systems](#).

### 5.2. Metric Collection

- ▶ **Finding available metrics**

For nvprof, you can use **--query-metrics** to see the list of metrics available for the current devices on your machine. You can also use **--devices** to filter which local devices to query. For NVIDIA Nsight Compute CLI, this functionality is the same. However, in addition, you can combine **--query-metrics** with **--chip [chipname]** to query the available metrics for any chip, not only the ones in your present CUDA devices.

Note that metric names have changed between nvprof and NVIDIA Nsight Compute CLI and metric names also differ between chips after (and including) GV100 and those before. See [Metric Comparison](#) for a comparison of nvprof and NVIDIA Nsight Compute metric names.

On Volta and newer GPUs, most metrics are named using a base name and various suffixes, e.g. *sm\_\_throughput.avg.pct\_of\_peak\_sustained\_elapsed*. The base name is *sm\_\_throughput* and the suffix is *avg.pct\_of\_peak\_sustained\_elapsed*. This is because most metrics follow the same structure and have the same set of suffixes. You need to pass the full name to NVIDIA Nsight Compute when selecting a metric for profiling.

To reduce the number of metrics shown for Volta and newer GPUs when using **--query-metrics**, by default only the base names are shown. Use **--query-metrics-mode suffix --metrics <metrics list>** to see the full names for the chosen metrics. Use **--query-metrics-mode all** to see all metrics with their full name directly.

► **Selecting which metrics to collect**

In both nvprof and NVIDIA Nsight Compute CLI, you can specify a comma-separated list of metric names to the **--metrics** option. While nvprof would allow you to collect either a list or all metrics, in NVIDIA Nsight Compute CLI you can use regular expressions to select a more fine-granular subset of all available metrics. For example, you can use **--metrics "regex:.\*"** to collect all metrics, or **--metrics "regex:smsp\_\_cycles\_elapsed"** to collect all "smsp\_\_cycles\_elapsed" metrics.

► **Selecting which events to collect**

You cannot collect any events in NVIDIA Nsight Compute CLI.

► **Selecting which section to collect**

In nvprof, you can either collect individual metrics or events, or a pre-configured set (all, analysis-metrics). NVIDIA Nsight Compute CLI adds the concept of a *section*. A section is a file that describes which metrics to collect for which GPU architecture, or architecture range. Furthermore, it defines how those metrics will be shown in both the command line output or the user interface. This includes structuring in tables, charts, histograms etc.

NVIDIA Nsight Compute CLI comes with a set of pre-defined sections, located in the **sections** directory. You can inspect, modify or extend those, as well as add new ones, e.g. to easily collect recurring metric sets. Each section specifies a unique *section identifier*, and there must not be two sections with the same identifier in the search path.

By default, the sections associated with the default section set are collected. You can select one or more individual sections using the **--section [section identifier]** option one or more times. If no **--section** option is given, but **--metrics** is used, no sections will be collected.

► **Selecting which section set to collect**

In nvprof, you can either collect individual metrics or events, or a pre-configured set (all, analysis-metrics). NVIDIA Nsight Compute CLI adds the concept of *section sets*. A section set defines a group of sections to collect together, in order to achieve different profiling overheads, depending on the required analysis level of detail.

If no other options are selected, the default section set is collected. You can select one or more sets using the **--set [set identifier]** option one or more times. If no **--set** option is given, but **--section** or **--metrics** is used, no sets will be collected.

## 5.3. Metric Comparison

NVIDIA Nsight Compute uses two groups of metrics, depending on which GPU architecture is profiled. For nvprof metrics, the following table lists the equivalent metrics in NVIDIA Nsight Compute, if available. For a detailed explanation of the structuring of PerfWorks metrics, see [Metrics Structure](#).

Metrics starting with *sm\_\_* are collected per-SM. Metrics starting with *smsp\_\_* are collected per-SM subpartition. However, all corresponding nvprof events are collected per-SM, only. Check the [Metrics Guide](#) for more details on these terms.

**Table 9 Metrics Mapping Table from CUPTI to PerfWorks**

| nvprof Metric                   | PerfWorks Metric or Formula (>= SM 7.0)   |
|---------------------------------|---|
| achieved_occupancy              | sm__warps_active.avg.pct_of_peak_sustained_active   |
| atomic_transactions             | l1tex__t_set_accesses_pipe_lsu_mem_global_op_atom.sum + l1tex__t_set_accesses_pipe_lsu_mem_global_op_red.sum  |
| atomic_transactions_per_request | (l1tex__t_sectors_pipe_lsu_mem_global_op_atom.sum + l1tex__t_sectors_pipe_lsu_mem_global_op_red.sum) / (l1tex__t_requests_pipe_lsu_mem_global_op_atom.sum + l1tex__t_requests_pipe_lsu_mem_global_op_red.sum) |
| branch_efficiency               | smsp__sass_average_branch_targets_threads_uniform.pct   |
| cf_executed                     | smsp__inst_executed_pipe_cbu.sum + smsp__inst_executed_pipe_adu.sum   |
| cf_fu_utilization               | n/a   |
| cf_issued                       | n/a   |
| double_precision_fu_utilization | smsp__inst_executed_pipe_fp64.avg.pct_of_peak_sustained_active  |
| dram_read_bytes                 | dram__bytes_read.sum  |
| dram_read_throughput            | dram__bytes_read.sum.per_second   |
| dram_read_transactions          | dram__sectors_read.sum  |
| dram_utilization                | dram__throughput.avg.pct_of_peak_sustained_elapsed  |
| dram_write_bytes                | dram__bytes_write.sum   |
| dram_write_throughput           | dram__bytes_write.sum.per_second  |
| dram_write_transactions         | dram__sectors_write.sum   |
| eligible_warps_per_cycle        | smsp__warps_eligible.sum.per_cycle_active   |
| flop_count_dp                   | smsp__sass_thread_inst_executed_op_dadd_pred_on.sum + smsp__sass_thread_inst_executed_op_dmul_pred_on.sum + smsp__sass_thread_inst_executed_op_dfma_pred_on.sum * 2   |
| flop_count_dp_add               | smsp__sass_thread_inst_executed_op_dadd_pred_on.sum   |
| flop_count_dp_fma               | smsp__sass_thread_inst_executed_op_dfma_pred_on.sum   |
| flop_count_dp_mul               | smsp__sass_thread_inst_executed_op_dmul_pred_on.sum   |
| flop_count_hp                   | smsp__sass_thread_inst_executed_op_hadd_pred_on.sum + smsp__sass_thread_inst_executed_op_hmul_pred_on.sum + smsp__sass_thread_inst_executed_op_hfma_pred_on.sum * 2   |
| flop_count_hp_add               | smsp__sass_thread_inst_executed_op_hadd_pred_on.sum   |
| flop_count_hp_fma               | smsp__sass_thread_inst_executed_op_hfma_pred_on.sum   |
| flop_count_hp_mul               | smsp__sass_thread_inst_executed_op_hmul_pred_on.sum   |
| flop_count_sp                   | smsp__sass_thread_inst_executed_op_fadd_pred_on.sum + smsp__sass_thread_inst_executed_op_fmul_pred_on.sum + smsp__sass_thread_inst_executed_op_ffma_pred_on.sum * 2   |
| flop_count_sp_add               | smsp__sass_thread_inst_executed_op_fadd_pred_on.sum   |
| flop_count_sp_fma               | smsp__sass_thread_inst_executed_op_ffma_pred_on.sum   |
| flop_count_sp_mul               | smsp__sass_thread_inst_executed_op_fmul_pred_on.sum   |
| flop_count_sp_special           | n/a   |
| flop_dp_efficiency              | smsp__sass_thread_inst_executed_ops_dadd_dmul_dfma_pred_on.avg.pct_of_peak_sustained_elapsed  |

| nvprof Metric                    | PerfWorks Metric or Formula (>= SM 7.0)   |
|----------------------------------|---|
| flop_hp_efficiency               | smsp__sass_thread_inst_executed_ops_hadd_hmul_hfma_pred_on.avg.pct_of_peak_sustained_elapsed  |
| flop_sp_efficiency               | smsp__sass_thread_inst_executed_ops_fadd_fmuls_ffma_pred_on.avg.pct_of_peak_sustained_elapsed   |
| gld_efficiency                   | smsp__sass_average_data_bytes_per_sector_mem_global_op_ld.pct   |
| gld_requested_throughput         | n/a   |
| gld_throughput                   | l1tex__t_bytes_pipe_lsu_mem_global_op_ld.sum.per_second   |
| gld_transactions                 | l1tex__t_sectors_pipe_lsu_mem_global_op_ld.sum  |
| gld_transactions_per_request     | l1tex__average_t_sectors_per_request_pipe_lsu_mem_global_op_ld.ratio  |
| global_atomic_requests           | l1tex__t_requests_pipe_lsu_mem_global_op_atom.sum   |
| global_hit_rate                  | (l1tex__t_sectors_pipe_lsu_mem_global_op_ld_lookup_hit.sum + l1tex__t_sectors_pipe_lsu_mem_global_op_st_lookup_hit.sum + l1tex__t_sectors_pipe_lsu_mem_global_op_red_lookup_hit.sum + l1tex__t_sectors_pipe_lsu_mem_global_op_atom_lookup_hit.sum) / (l1tex__t_sectors_pipe_lsu_mem_global_op_ld.sum + l1tex__t_sectors_pipe_lsu_mem_global_op_st.sum + l1tex__t_sectors_pipe_lsu_mem_global_op_red.sum + l1tex__t_sectors_pipe_lsu_mem_global_op_atom.sum) |
| global_load_requests             | l1tex__t_requests_pipe_lsu_mem_global_op_ld.sum   |
| global_reduction_requests        | l1tex__t_requests_pipe_lsu_mem_global_op_red.sum  |
| global_store_requests            | l1tex__t_requests_pipe_lsu_mem_global_op_st.sum   |
| gst_efficiency                   | smsp__sass_average_data_bytes_per_sector_mem_global_op_st.pct   |
| gst_requested_throughput         | n/a   |
| gst_throughput                   | l1tex__t_bytes_pipe_lsu_mem_global_op_st.sum.per_second   |
| gst_transactions                 | l1tex__t_sectors_pipe_lsu_mem_global_op_st.sum  |
| gst_transactions_per_request     | l1tex__average_t_sectors_per_request_pipe_lsu_mem_global_op_st.ratio  |
| half_precision_fu_utilization    | smsp__inst_executed_pipe_fp16.avg.pct_of_peak_sustained_active  |
| inst_bit_convert                 | smsp__sass_thread_inst_executed_op_conversion_pred_on.sum   |
| inst_compute_ld_st               | smsp__sass_thread_inst_executed_op_memory_pred_on.sum   |
| inst_control                     | smsp__sass_thread_inst_executed_op_control_pred_on.sum  |
| inst_executed                    | smsp__inst_executed.sum   |
| inst_executed_global_atomics     | smsp__sass_inst_executed_op_global_atom.sum   |
| inst_executed_global_loads       | smsp__inst_executed_op_global_ld.sum  |
| inst_executed_global_reductions  | smsp__inst_executed_op_global_red.sum   |
| inst_executed_global_stores      | smsp__inst_executed_op_global_st.sum  |
| inst_executed_local_loads        | smsp__inst_executed_op_local_ld.sum   |
| inst_executed_local_stores       | smsp__inst_executed_op_local_st.sum   |
| inst_executed_shared_atomics     | smsp__inst_executed_op_shared_atom.sum + smsp__inst_executed_op_shared_atom_dot_alu.sum + smsp__inst_executed_op_shared_atom_dot_cas.sum  |
| inst_executed_shared_loads       | smsp__inst_executed_op_shared_ld.sum  |
| inst_executed_shared_stores      | smsp__inst_executed_op_shared_st.sum  |
| inst_executed_surface_atomics    | smsp__inst_executed_op_surface_atom.sum   |
| inst_executed_surface_loads      | smsp__inst_executed_op_surface_ld.sum + smsp__inst_executed_op_shared_atom_dot_alu.sum + smsp__inst_executed_op_shared_atom_dot_cas.sum   |
| inst_executed_surface_reductions | smsp__inst_executed_op_surface_red.sum  |
| inst_executed_surface_stores     | smsp__inst_executed_op_surface_st.sum   |
| inst_executed_tex_ops            | smsp__inst_executed_op_texture.sum  |
| inst_fp_16                       | smsp__sass_thread_inst_executed_op_fp16_pred_on.sum   |
| inst_fp_32                       | smsp__sass_thread_inst_executed_op_fp32_pred_on.sum   |
| inst_fp_64                       | smsp__sass_thread_inst_executed_op_fp64_pred_on.sum   |
| inst_integer                     | smsp__sass_thread_inst_executed_op_integer_pred_on.sum  |
| inst_inter_thread_communication  | smsp__sass_thread_inst_executed_op_inter_thread_communication_pred_on.sum   |
| inst_issued                      | smsp__inst_issued.sum   |
| inst_misc                        | smsp__sass_thread_inst_executed_op_misc_pred_on.sum   |

| nvprof Metric                        | PerfWorks Metric or Formula (>= SM 7.0)  |
|--------------------------------------|--|
| inst_per_warp                        | smsp__average_inst_executed_per_warp.ratio   |
| inst_replay_overhead                 | n/a  |
| ipc                                  | smsp__inst_executed.avg.per_cycle_active   |
| issue_slot_utilization               | smsp__issue_active.avg.pct_of_peak_sustained_active  |
| issue_slots                          | smsp__inst_issued.sum  |
| issued_ipc                           | smsp__inst_issued.avg.per_cycle_active   |
| l1_sm_lg_utilization                 | l1tex__lsu_writeback_active.avg.pct_of_peak_sustained_active   |
| l2_atomic_throughput                 | $2 * (lts\_t\_sectors\_op\_atom.sum.per\_second + lts\_t\_sectors\_op\_red.sum.per\_second)$   |
| l2_atomic_transactions               | $2 * (lts\_t\_sectors\_op\_atom.sum + lts\_t\_sectors\_op\_red.sum)$   |
| l2_global_atomic_store_bytes         | lts__t_bytes_equiv_l1sectormiss_pipe_lsu_mem_global_op_atom.sum  |
| l2_global_load_bytes                 | lts__t_bytes_equiv_l1sectormiss_pipe_lsu_mem_global_op_ld.sum  |
| l2_local_global_store_bytes          | lts__t_bytes_equiv_l1sectormiss_pipe_lsu_mem_local_op_st.sum + lts__t_bytes_equiv_l1sectormiss_pipe_lsu_mem_global_op_st.sum         |
| l2_local_load_bytes                  | lts__t_bytes_equiv_l1sectormiss_pipe_lsu_mem_local_op_ld.sum   |
| l2_read_throughput                   | $lts\_t\_sectors\_op\_read.sum.per\_second + lts\_t\_sectors\_op\_atom.sum.per\_second + lts\_t\_sectors\_op\_red.sum.per\_second^2$ |
| l2_read_transactions                 | $lts\_t\_sectors\_op\_read.sum + lts\_t\_sectors\_op\_atom.sum + lts\_t\_sectors\_op\_red.sum^2$                                     |
| l2_surface_load_bytes                | lts__t_bytes_equiv_l1sectormiss_pipe_tex_mem_surface_op_ld.sum   |
| l2_surface_store_bytes               | lts__t_bytes_equiv_l1sectormiss_pipe_tex_mem_surface_op_st.sum   |
| l2_tex_hit_rate                      | lts__t_sector_hit_rate.pct   |
| l2_tex_read_hit_rate                 | lts__t_sector_op_read_hit_rate.pct   |
| l2_tex_read_throughput               | lts__t_sectors_srcunit_tex_op_read.sum.per_second  |
| l2_tex_read_transactions             | lts__t_sectors_srcunit_tex_op_read.sum   |
| l2_tex_write_hit_rate                | lts__t_sector_op_write_hit_rate.pct  |
| l2_tex_write_throughput              | lts__t_sectors_srcunit_tex_op_write.sum.per_second   |
| l2_tex_write_transactions            | lts__t_sectors_srcunit_tex_op_write.sum  |
| l2_utilization                       | lts__t_sectors.avg.pct_of_peak_sustained_elapsed   |
| l2_write_throughput                  | $lts\_t\_sectors\_op\_write.sum.per\_second + lts\_t\_sectors\_op\_atom.sum.per\_second + lts\_t\_sectors\_op\_red.sum.per\_second$  |
| l2_write_transactions                | $lts\_t\_sectors\_op\_write.sum + lts\_t\_sectors\_op\_atom.sum + lts\_t\_sectors\_op\_red.sum$                                      |
| ldst_executed                        | n/a  |
| ldst_fu_utilization                  | smsp__inst_executed_pipe_lsu.avg.pct_of_peak_sustained_active  |
| ldst_issued                          | n/a  |
| local_hit_rate                       | n/a  |
| local_load_requests                  | l1tex__t_requests_pipe_lsu_mem_local_op_ld.sum   |
| local_load_throughput                | l1tex__t_bytes_pipe_lsu_mem_local_op_ld.sum.per_second   |
| local_load_transactions              | l1tex__t_sectors_pipe_lsu_mem_local_op_ld.sum  |
| local_load_transactions_per_request  | l1tex__average_t_sectors_per_request_pipe_lsu_mem_local_op_ld.ratio  |
| local_memory_overhead                | n/a  |
| local_store_requests                 | l1tex__t_requests_pipe_lsu_mem_local_op_st.sum   |
| local_store_throughput               | l1tex__t_sectors_pipe_lsu_mem_local_op_st.sum.per_second   |
| local_store_transactions             | l1tex__t_sectors_pipe_lsu_mem_local_op_st.sum  |
| local_store_transactions_per_request | l1tex__average_t_sectors_per_request_pipe_lsu_mem_local_op_st.ratio  |
| nvlink_data_receive_efficiency       | n/a  |
| nvlink_data_transmission_efficiency  | n/a  |
| nvlink_overhead_data_received        | $(nvlrx\_bytes\_data\_protocol.sum / nvlrx\_bytes.sum) * 100$  |
| nvlink_overhead_data_transmitted     | $(nvltx\_bytes\_data\_protocol.sum / nvltx\_bytes.sum) * 100$  |

<sup>2</sup> Sector reads from reductions are added here only for compatibility to the current definition of the metric in nvprof. Reductions do not cause data to be communicated from L2 back to L1.



| nvprof Metric                         | PerfWorks Metric or Formula (>= SM 7.0)   |
|---------------------------------------|---|
| nvlink_receive_throughput             | nvlrx__bytes.sum.per_second   |
| nvlink_total_data_received            | nvlrx__bytes.sum  |
| nvlink_total_data_transmitted         | nvltx__bytes.sum  |
| nvlink_total_nratom_data_transmitted  | n/a   |
| nvlink_total_ratom_data_transmitted   | n/a   |
| nvlink_total_response_data_received   | n/a   |
| nvlink_total_write_data_transmitted   | n/a   |
| nvlink_transmit_throughput            | nvltx__bytes.sum.per_second   |
| nvlink_user_data_received             | nvlrx__bytes_data_user.sum  |
| nvlink_user_data_transmitted          | nvltx__bytes_data_user.sum  |
| nvlink_user_nratom_data_transmitted   | n/a   |
| nvlink_user_ratom_data_transmitted    | n/a   |
| nvlink_user_response_data_received    | n/a   |
| nvlink_user_write_data_transmitted    | n/a   |
| pcie_total_data_received              | pcie__read_bytes.sum  |
| pcie_total_data_transmitted           | pcie__write_bytes.sum   |
| shared_efficiency                     | smsp__sass_average_data_bytes_per_wavefront_mem_shared.pct  |
| shared_load_throughput                | l1tex__data_pipe_lsu_wavefronts_mem_shared_op_ld.sum.per_second   |
| shared_load_transactions              | l1tex__data_pipe_lsu_wavefronts_mem_shared_op_ld.sum  |
| shared_load_transactions_per_request  | n/a   |
| shared_store_throughput               | l1tex__data_pipe_lsu_wavefronts_mem_shared_op_st.sum.per_second   |
| shared_store_transactions             | l1tex__data_pipe_lsu_wavefronts_mem_shared_op_st.sum  |
| shared_store_transactions_per_request | n/a   |
| shared_utilization                    | l1tex__data_pipe_lsu_wavefronts_mem_shared.avg.pct_of_peak_sustained_elapsed  |
| single_precision_fu_utilization       | smsp__pipe_fma_cycles_active.avg.pct_of_peak_sustained_active   |
| sm_efficiency                         | smsp__cycles_active.avg.pct_of_peak_sustained_elapsed   |
| sm_tex_utilization                    | l1tex__texin_sm2tex_req_cycles_active.avg.pct_of_peak_sustained_elapsed   |
| special_fu_utilization                | smsp__inst_executed_pipe_xu.avg.pct_of_peak_sustained_active  |
| stall_constant_memory_dependency      | smsp__warp_issue_stalled_imc_miss_per_warp_active.pct   |
| stall_exec_dependency                 | smsp__warp_issue_stalled_short_scoreboard_per_warp_active.pct + smsp__warp_issue_stalled_wait_per_warp_active.pct           |
| stall_inst_fetch                      | smsp__warp_issue_stalled_no_instruction_per_warp_active.pct   |
| stall_memory_dependency               | smsp__warp_issue_stalled_long_scoreboard_per_warp_active.pct  |
| stall_memory_throttle                 | smsp__warp_issue_stalled_drain_per_warp_active.pct + smsp__warp_issue_stalled_lg_throttle_per_warp_active.pct               |
| stall_not_selected                    | smsp__warp_issue_stalled_not_selected_per_warp_active.pct   |
| stall_other                           | smsp__warp_issue_stalled_dispatch_stall_per_warp_active.pct + smsp__warp_issue_stalled_misc_per_warp_active.pct             |
| stall_pipe_busy                       | smsp__warp_issue_stalled_math_pipe_throttle_per_warp_active.pct + smsp__warp_issue_stalled_mio_throttle_per_warp_active.pct |
| stall_sleeping                        | smsp__warp_issue_stalled_sleeping_per_warp_active.pct   |
| stall_sync                            | smsp__warp_issue_stalled_barrier_per_warp_active.pct + smsp__warp_issue_stalled_membar_per_warp_active.pct                  |
| stall_texture                         | smsp__warp_issue_stalled_tex_throttle_per_warp_active.pct   |
| surface_atomic_requests               | l1tex__t_requests_pipe_tex_mem_surface_op_atom.sum  |
| surface_load_requests                 | l1tex__t_requests_pipe_tex_mem_surface_op_ld.sum  |
| surface_reduction_requests            | l1tex__t_requests_pipe_tex_mem_surface_op_red.sum   |
| surface_store_requests                | l1tex__t_requests_pipe_tex_mem_surface_op_st.sum  |
| sysmem_read_bytes                     | lts__t_sectors_aperture_sysmem_op_read * 32   |
| sysmem_read_throughput                | lts__t_sectors_aperture_sysmem_op_read.sum.per_second   |
| sysmem_read_transactions              | lts__t_sectors_aperture_sysmem_op_read.sum  |

| nvprof Metric                     | PerfWorks Metric or Formula (>= SM 7.0)  |
|-----------------------------------|--|
| sysmem_read_utilization           | n/a  |
| sysmem_utilization                | n/a  |
| sysmem_write_bytes                | lts__t_sectors_aperture_sysmem_op_write * 32   |
| sysmem_write_throughput           | lts__t_sectors_aperture_sysmem_op_write.sum.per_second   |
| sysmem_write_transactions         | lts__t_sectors_aperture_sysmem_op_write.sum  |
| sysmem_write_utilization          | n/a  |
| tensor_precision_fu_utilization   | sm__pipe_tensor_op_hmma_cycles_active.avg.pct_of_peak_sustained_active   |
| tensor_precision_int_utilization  | sm__pipe_tensor_op_imma_cycles_active.avg.pct_of_peak_sustained_active (SM 7.2+)   |
| tex_cache_hit_rate                | l1tex__t_sector_hit_rate.pct   |
| tex_cache_throughput              | n/a  |
| tex_cache_transactions            | l1tex__lsu_writeback_active.avg.pct_of_peak_sustained_active +<br>l1tex__tex_writeback_active.avg.pct_of_peak_sustained_active |
| tex_fu_utilization                | smsp__inst_executed_pipe_tex.avg.pct_of_peak_sustained_active  |
| tex_sm_tex_utilization            | l1tex__f_tex2sm_cycles_active.avg.pct_of_peak_sustained_elapsed  |
| tex_sm_utilization                | sm__mio2rf_writeback_active.avg.pct_of_peak_sustained_elapsed  |
| tex_utilization                   | n/a  |
| texture_load_requests             | l1tex__t_requests_pipe_tex_mem_texture.sum   |
| warp_execution_efficiency         | smsp__thread_inst_executed_per_inst_executed.ratio   |
| warp_nonpred_execution_efficiency | smsp__thread_inst_executed_per_inst_executed.pct   |

## 5.4. Event Comparison

For nvprof events, the following table lists the equivalent metrics in NVIDIA Nsight Compute, if available. For a detailed explanation of the structuring of PerfWorks metrics, see [Metrics Structure](#).

Metrics starting with *sm\_\_* are collected per-SM. Metrics starting with *smsp\_\_* are collected per-SM subpartition. However, all corresponding nvprof events are collected per-SM, only. Check the [Metrics Guide](#) for more details on these terms.

**Table 10 Events Mapping Table from CUPTI Events to PerfWorks Metrics for Compute Capability >= 7.0**

| nvprof Event           | PerfWorks Metric or Formula (>= SM 7.0)         |
|------------------------|---|
| active_cycles          | sm__cycles_active.sum                           |
| active_cycles_pm       | sm__cycles_active.sum                           |
| active_cycles_sys      | sys__cycles_active.sum                          |
| active_warps           | sm__warps_active.sum                            |
| active_warps_pm        | sm__warps_active.sum                            |
| atom_count             | smsp__inst_executed_op_generic_atom_dot_alu.sum |
| elapsed_cycles_pm      | sm__cycles_elapsed.sum                          |
| elapsed_cycles_sm      | sm__cycles_elapsed.sum                          |
| elapsed_cycles_sys     | sys__cycles_elapsed.sum                         |
| fb_subp0_read_sectors  | dram__sectors_read.sum                          |
| fb_subp1_read_sectors  | dram__sectors_read.sum                          |
| fb_subp0_write_sectors | dram__sectors_write.sum                         |
| fb_subp1_write_sectors | dram__sectors_write.sum                         |

| nvprof Event                            | PerfWorks Metric or Formula (>= SM 7.0)  |
|---|--|
| global_atom_cas                         | smsp__inst_executed_op_generic_atom_dot_cas.sum                                      |
| gred_count                              | smsp__inst_executed_op_global_red.sum  |
| inst_executed                           | sm__inst_executed.sum  |
| inst_executed_fma_pipe_s0               | smsp__inst_executed_pipe_fma.sum   |
| inst_executed_fma_pipe_s1               | smsp__inst_executed_pipe_fma.sum   |
| inst_executed_fma_pipe_s2               | smsp__inst_executed_pipe_fma.sum   |
| inst_executed_fma_pipe_s3               | smsp__inst_executed_pipe_fma.sum   |
| inst_executed_fp16_pipe_s0              | smsp__inst_executed_pipe_fp16.sum  |
| inst_executed_fp16_pipe_s1              | smsp__inst_executed_pipe_fp16.sum  |
| inst_executed_fp16_pipe_s2              | smsp__inst_executed_pipe_fp16.sum  |
| inst_executed_fp16_pipe_s3              | smsp__inst_executed_pipe_fp16.sum  |
| inst_executed_fp64_pipe_s0              | smsp__inst_executed_pipe_fp64.sum  |
| inst_executed_fp64_pipe_s1              | smsp__inst_executed_pipe_fp64.sum  |
| inst_executed_fp64_pipe_s2              | smsp__inst_executed_pipe_fp64.sum  |
| inst_executed_fp64_pipe_s3              | smsp__inst_executed_pipe_fp64.sum  |
| inst_issued1                            | sm__inst_issued.sum  |
| l2_subp0_read_sector_misses             | lts__t_sectors_op_read_lookup_miss.sum   |
| l2_subp1_read_sector_misses             | lts__t_sectors_op_read_lookup_miss.sum   |
| l2_subp0_read_sysmem_sector_queries     | lts__t_sectors_aperture_sysmem_op_read.sum   |
| l2_subp1_read_sysmem_sector_queries     | lts__t_sectors_aperture_sysmem_op_read.sum   |
| l2_subp0_read_tex_hit_sectors           | lts__t_sectors_srcunit_tex_op_read_lookup_hit.sum                                    |
| l2_subp1_read_tex_hit_sectors           | lts__t_sectors_srcunit_tex_op_read_lookup_hit.sum                                    |
| l2_subp0_read_tex_sector_queries        | lts__t_sectors_srcunit_tex_op_read.sum   |
| l2_subp1_read_tex_sector_queries        | lts__t_sectors_srcunit_tex_op_read.sum   |
| l2_subp0_total_read_sector_queries      | lts__t_sectors_op_read.sum + lts__t_sectors_op_atom.sum + lts__t_sectors_op_red.sum  |
| l2_subp1_total_read_sector_queries      | lts__t_sectors_op_read.sum + lts__t_sectors_op_atom.sum + lts__t_sectors_op_red.sum  |
| l2_subp0_total_write_sector_queries     | lts__t_sectors_op_write.sum + lts__t_sectors_op_atom.sum + lts__t_sectors_op_red.sum |
| l2_subp1_total_write_sector_queries     | lts__t_sectors_op_write.sum + lts__t_sectors_op_atom.sum + lts__t_sectors_op_red.sum |
| l2_subp0_write_sector_misses            | lts__t_sectors_op_write_lookup_miss.sum  |
| l2_subp1_write_sector_misses            | lts__t_sectors_op_write_lookup_miss.sum  |
| l2_subp0_write_sysmem_sector_queries    | lts__t_sectors_aperture_sysmem_op_write.sum  |
| l2_subp1_write_sysmem_sector_queries    | lts__t_sectors_aperture_sysmem_op_write.sum  |
| l2_subp0_write_tex_hit_sectors          | lts__t_sectors_srcunit_tex_op_write_lookup_hit.sum                                   |
| l2_subp1_write_tex_hit_sectors          | lts__t_sectors_srcunit_tex_op_write_lookup_hit.sum                                   |
| l2_subp0_write_tex_sector_queries       | lts__t_sectors_srcunit_tex_op_write.sum  |
| l2_subp1_write_tex_sector_queries       | lts__t_sectors_srcunit_tex_op_write.sum  |
| not_predicated_off_thread_inst_executed | smsp__thread_inst_executed_pred_on.sum   |
| pcie_rx_active_pulse                    | n/a  |
| pcie_tx_active_pulse                    | n/a  |
| prof_trigger_00                         | n/a  |
| prof_trigger_01                         | n/a  |
| prof_trigger_02                         | n/a  |
| prof_trigger_03                         | n/a  |
| prof_trigger_04                         | n/a  |
| prof_trigger_05                         | n/a  |
| prof_trigger_06                         | n/a  |

| nvprof Event                 | PerfWorks Metric or Formula (>= SM 7.0)                  |
|------------------------------|--|
| prof_trigger_07              | n/a  |
| inst_issued0                 | smsp__issue_inst0.sum                                    |
| sm_cta_launched              | sm__ctas_launched.sum                                    |
| shared_load                  | smsp__inst_executed_op_shared_ld.sum                     |
| shared_store                 | smsp__inst_executed_op_shared_st.sum                     |
| generic_load                 | smsp__inst_executed_op_generic_ld.sum                    |
| generic_store                | smsp__inst_executed_op_generic_st.sum                    |
| global_load                  | smsp__inst_executed_op_global_ld.sum                     |
| global_store                 | smsp__inst_executed_op_global_st.sum                     |
| local_load                   | smsp__inst_executed_op_local_ld.sum                      |
| local_store                  | smsp__inst_executed_op_local_st.sum                      |
| shared_atom                  | smsp__inst_executed_op_shared_atom.sum                   |
| shared_atom_cas              | smsp__inst_executed_shared_atom_dot_cas.sum              |
| shared_ld_bank_conflict      | l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum |
| shared_st_bank_conflict      | l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum |
| shared_ld_transactions       | l1tex__data_pipe_lsu_wavefronts_mem_shared_op_ld.sum     |
| shared_st_transactions       | l1tex__data_pipe_lsu_wavefronts_mem_shared_op_st.sum     |
| tensor_pipe_active_cycles_s0 | smsp__pipe_tensor_cycles_active.sum                      |
| tensor_pipe_active_cycles_s1 | smsp__pipe_tensor_cycles_active.sum                      |
| tensor_pipe_active_cycles_s2 | smsp__pipe_tensor_cycles_active.sum                      |
| tensor_pipe_active_cycles_s3 | smsp__pipe_tensor_cycles_active.sum                      |
| thread_inst_executed         | smsp__thread_inst_executed.sum                           |
| warps_launched               | smsp__warps_launched.sum                                 |

## 5.5. Filtering

### ► Filtering by kernel name

Both nvprof and NVIDIA Nsight Compute CLI support filtering which kernels' data should be collected. In nvprof, the option is **--kernels** and applies to following metric collection options. In NVIDIA Nsight Compute CLI, the option is named **--kernel-regex** and applies to the complete application execution. In other words, NVIDIA Nsight Compute CLI does not currently support collecting different metrics for different kernels, unless they execute on different GPU architectures.

### ► Filtering by kernel ID

Nvprof allows users to specify which kernels to profile using a kernel ID description, using the same **--kernels** option. In NVIDIA Nsight Compute CLI, the syntax for this kernel ID is identical, but the option is named **--kernel-id**.

### ► Filtering by device

Both nvprof and NVIDIA Nsight Compute CLI use **--devices** to filter the devices which to profile. In contrast to nvprof, in NVIDIA Nsight Compute CLI the option applies globally, not only to following options.

## 5.6. Output

### ▶ API trace and summary

NVIDIA Nsight Compute CLI does not support any form of API-usage related output. No API data is captured during profiling.

### ▶ Dependency analysis

NVIDIA Nsight Compute CLI does not support any dependency analysis. No API data is captured during profiling.

### ▶ GPU trace

NVIDIA Nsight Compute CLI does not support any GPU trace output. Due to kernel replay during profiling, kernel executions are serialized, and start and end timestamps do not necessarily match those during application execution. In addition, no records for memory activities are recorded.

### ▶ Print summary

While nvprof has several command line options to specify which summary information to print, NVIDIA Nsight Compute CLI uses further arguments to the **--summary** options. Profiling data can be summarized **per-gpu**, **per-kernel** or **per-nvtx** context.

### ▶ Kernel name demangling

Nvprof allows users to decide between name demangling on or off using the **--demangling** options. NVIDIA Nsight Compute CLI currently always demangles kernel names in the output. In addition, the option **--kernel-regex-base** can be used to decide which name format should be used when matching kernel names during filtering.

### ▶ Pages

Nvprof has no concept of output pages, all data is shown as a list or summarized. NVIDIA Nsight Compute CLI uses *pages* to define how data should be structured and printed. Those correspond to the report pages used in the GUI variant. The option **--page** can be used to select which page to show, and **details** is selected by default. All pages also support printing in CSV format for easier post-processing, using the **--csv** option.

## 5.7. Launch and Attach

### ▶ Launching a process for profiling

In nvprof, the application to profile is passed to the tool as a command line argument. The application must be a local executable. Alternatively, you can choose to use the tool in a *daemon mode* and profile all applicable processes on the local machine (nvprof option **--profile-all-processes**). In nvprof, the decision to profile the complete process tree or only the root process is done via the **--**

**profile-child-processes** flag. In NVIDIA Nsight Compute CLI, the **--target-processes** option is used for this.

NVIDIA Nsight Compute CLI has several modes to determine which application to collect data for. By default, the executable passed via the command line to the tool is started, connected to, and profiled. This mode is called **launch-and-attach**.

- ▶ **Launching a process for attach**

In contrast to nvprof, you can choose to only launch a local executable. In this mode (**--mode launch**), the process is started, connected to, but then suspended at the first CUDA API call. Subsequently, there is a third mode (**--mode attach**) to attach to any process launched using the aforementioned mode. In this case, all profiling and output options would be passed to the attaching instance of NVIDIA Nsight Compute CLI.

- ▶ **Remote profiling**

Finally, using **launch** and **attach**, you can connect to a launched process on a remote machine, which could even run a different operating system than the local host. Use **--hostname** to select which remote host to connect to.

## **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 of 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**

© 2018-2021 NVIDIA Corporation. All rights reserved.

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