NVIDIA Topology-Aware GPU Selection

User Guide

DU-09998-001_v1.1 | April 2023

Table of Contents

Chapter 1. Introduction

Many graphics processing units (GPU)-accelerated high performance computing (HPC) applications that use Message Passing Interface (MPI) spend a substantial portion of their runtime in non-uniform GPU-to-GPU communications. The substantial time spent in GPU-to-GPU communication can prevent users from maximizing performance from their existing hardware.

To ensure that GPU-to-GPU communication in these applications is efficient, you need to make informed decisions when assigning GPUs to MPI processes. The assigning of GPUs to processes depends on the following factors:

- System GPU topology

  This topology shows how different GPUs are linked and the communication channel they use to connect. Different communication channels exist in GPU clusters and multi-GPU servers, which results in some GPU pairs using faster communication links.
- Application GPU profiling

  This topology considers the total volume of communication between different GPUs in the system. It also shows the application's communication pattern and that some GPU pairs can have a higher communication volume.

NVIDIA Topology-Aware GPU Selection (NVTAGS) is a toolset for HPC applications that uses MPI to enable faster solve times for applications with a high GPU communication time and a communication pattern that does not fit the underlying system GPU topology. NVTAGS can also benefit applications that do not efficiently perform their CPU and NIC (or HCA) affinity setting. Tests suggest that Deep Learning workloads on the framework, such as TensorFlow and HPC applications, can benefit from efficient affinity settings that are provided by NVTAGS.

NVTAGS automates the following processes:

- Profiling application GPU communication by using a PMPI-based profiler.
- Extracting the system GPU communication topology by leveraging NVDIA's System Management Interface (`nvidia-smi`) or by benchmarking the underlying GPU links to determine their strength.
- Finding an efficient way to assign GPUs to processes to minimize GPU communication congestion.

  This reduces the overall GPU-to-GPU communication time of HPC applications that run on a multi-GPU system.

- Setting the CPU and NIC (or HCA) affinity so that processes use CPU cores/sockets and network interfaces in their affinity.

Here is the two-step process that NVTAGS follows to identify and apply efficient GPU assignments:

1. NVTAGS Tune

   In this step, NVTAGS does the following:

   - Gathers application and system profiling data to understand how GPU-to-GPU communication is performed for a target application.
   - Leverages this profiling information to identify and recommend a GPU assignment solution that better suits your application on the target system, so this assignment solution can be used in subsequent runs.
   - Stores profiling and mapping results locally in the NVTAGS cache for use in subsequent runs.

     NVTAGS allows you to change the default cache directory path. See Changing the NVTAGS Cache Path for more information.
2. NVTAGS Run

   In this step, NVTAGS explores system affinity and applies the suggested GPU assignment from the tuning step and initiates your application `run` command. Based on how GPUs are selected in your application, NVTAGS also automatically sets the proper CPU and NIC (or HCA) affinity.

   > Note: Both NVTAGS Tune and Run steps are lightweight and impose a negligible overhead for most of the MPI applications.

1.1. Systems that NVTAGS Benefits

NVTAGS leverages the GPU communication pattern of an application and the GPU topology of a system to generate efficient GPU assignments for an application that runs on the system.

Systems with asymmetric GPU topologies, where some GPU pairs share stronger communication links than other pairs, will benefit the most by using NVTAGS. Examples of these systems include most GPU clusters, NVIDIA DGX-1™, and PCIe servers that use different GPU communication channels to connect GPUs.

Systems with symmetric GPU topologies, where all GPU pairs use the same communication links with equal capacity, will not benefit from custom GPU assignment because shuffling GPUs do not guide processes to use GPU pairs with stronger communication links. Examples of these systems include NVIDIA DGX-2™ and NVIDIA DGX™ A100. Systems with symmetric topologies do not benefit from the custom GPU assignments provided by NVTAGS because all GPU assignments on these systems are equally optimal. However, these systems might still benefit from efficient GPU selection when used as a part of a cluster where network communication links are not as strong as intranode communication channels. All of the previously mentioned systems can be assessed by NVTAGS and will benefit from efficient affinity settings.

1.2. NVTAGS Affinity Settings

Today, many modern systems are equipped with multiple GPUs, multiple NICs, and multi-socket/multi-core CPUs. Using NICs and CPU cores that are within the affinity of the selected GPUs plays a crucial role in applications that run on these systems.

However, the default affinity setting with the current schedulers cannot always meet these requirements. It can also be difficult for users to determine and set efficient custom affinity settings, because these settings can be system and scheduler dependent.

To remove the burden of efficient affinity settings from users, NVTAGS provides automatic system affinity assessment and settings. In this process, NVTAGS determines your system's NIC and CPU topology, and based on selected GPUs, decides how to apply the CPU and NIC affinity settings. To apply affinity settings, NVTAGS considers the system and scheduler requirements. When running with Slurm, NVTAGS exploits `numactl` for CPU binding. If `numactl` is not installed, NVTAGS falls back to using `taskset`.

NVTAGS is scheduler aware, so when performing affinity setting process, NVTAGS uses the following:

- `SLURM_HOSTID` when you run your application with Slurm.
- The MPI hostfile, when you launch your application with `mpirun`.

Chapter 2. Getting Started

This section provides information about the requirements to install NVTAGS, the installation instructions, and how to use NVTAGS.

2.1. Prerequisites

This section provides information about installing and using NVTAGS.

Ensure that you have completed the following prerequisites:

- Have a Linux operating system.
- Have an x86 system architecture.
- Installed a working NVIDIA Graphics Driver with CUDA 11 support (version 450 or later).

  To download the latest NVDIA Graphics Driver for your system, go to Download Drivers.
- Have a GCC compiler with GLIBCXX_3.4.21 support.

  You can use GCC 7.4 and later.
- Have at least 3 GPUs installed on your machine(s).
- Verified that your application uses one GPU per MPI process.
- Have a CUDA-aware Open MPI to run your application.

  MPI Fortran is not supported.
- Verified that you are using a CUDA-aware OpenMPI Version that is supported by NVTAGS.

  NVTAGS currently supports OpenMPI versions 3.0, 3.1, 4.0, and 4.1.

2.2. Installing NVTAGS

Complete these steps to install NVTAGS.

About this task

Before you install NVTAGS, read the Prerequisites.

Procedure

1. Download the latest NVTAGS release from the NVTAGS releases page.
2. To extract the NVTAGS archive, run the following command.

```
tar -xzvf nvtags-1.1.0.tar.gz
```

3. Copy the NVTAGS directory into the default NVTAGS path on your machine.

```
cp -r nvtags-1.1.0 /opt/nvtags
```

4. Update `PATH` to make the NVTAGS binaries and libraries (for example, `libmpi_prof_x.y.so` or `libmpi_prof.so`) are discoverable.

```
export NVTAGS_PATH=/MY/PATH/TO/nvtags/bin
export PATH=$NVTAGS_PATH:$PATH
```

5. Optional: Although the NVTAGS binaries and scripts that are bundled in the NVTAGS release archive are executable, depending on your system, you might need to update your permissions.

```
chmod +x ${NVTAGS_PATH}/*
```

Chapter 3. Using NVTAGS

This section provides additional information about the two modes, NVTAGS Tune and NVTAGS Run.

NVTAGS supports running bare-metal or containerized applications with `singularity` or `enroot`/`pyxis` and includes Slurm integration support for `srun`.

> Note: To display generic help messages and NVTAGS usage information, run `nvtags --help`.

3.1. NVTAGS Tune Mode

In the NVTAGS Tune mode, the application and system profiling data are used to recommend an efficient GPU assignment.

The Tune mode requires application profiling data to evaluate the efficiency of default GPU assignments and search for a better GPU assignment by using mapping algorithms. After the tuning is complete, subsequent application runs can be used with NVTAGS in the Run mode.

In the NVTAGS Tune mode, application profiling data is used to extract the GPU communication pattern of the application. NVTAGS uses an MPI profiler library that dynamically links to your MPI application and intercepts MPI calls to build a GPU communication pattern. After the profiling is complete, NVTAGS looks for a better GPU assignment solution by using the application and system GPU topology information. The profiling results and recommended GPU assignments are cached in the local NVTAGS cache that defaults to `./.nvtags/.cache`. You can change the default NVTAGS default cache directory to a custom path. See Changing the NVTAGS Cache Path for more information. If the specified directory does not exist, NVTAGS will create it before storing the cached data.

Although NVTAGS can provide an efficient GPU assignment by using the default settings, NVTAGS might provide a better GPU assignment by using non-default settings. This process can be achieved by changing the default profiling and mapping settings with input arguments. See About the NVTAGS CLI for more information.

By default, NVTAGS uses pre-defined link values to represent the detected GPU linknames on your system. However, these values might vary from system to system and might not accurately represent your system's GPU link strength. To provide a more precise view of the communication channels that are available in your system, NVTAGS allows dynamic profiling by benchmarking the available GPU linknames on your system to determine their strength values. See Tuning with Dynamic Profiling for more information.

When the tuning step is complete, if a better GPU assignment is found, a message like the following is printed:

> Note: A list of GPU IDs is stored in the local NVTAGS cache.

```
Better mapping found!
Max Congestion_improvement: 10.00%
Avg Congestion_improvement: 17.27%
0,1,3,2,7,6,4,5
```

GPU IDs are only stored when the congestion improvement is greater than the NVTAGS threshold value. The default threshold value is 5%.

If no better GPU assignment is found, NVTAGS outputs the following message:

```
No Better mapping found!
```

3.1.1. Launching Your Application with NVTAGS Tune

There are a few ways in which you can launch your application in the NVTAGS Tune mode.

Bare Metal With mpirun

To run NVTAGS in the Tune with Profiling mode, prepend your application `run` command with `nvtags tune` and specify the runner binary that you want to use to launch your application:

```
nvtags tune --runner-bin mpirun --num-procs <number of processes> --app-run-cmd
 '<application run cmd>' [options]
```

Bare Metal With srun

> Tip: By default, NVTAGS will launch your application with `srun`.

To run NVTAGS in the Tune mode for a multi-GPU workload that was launched with `srun`, prepend your application `run` command with `nvtags tune`:

```
nvtags tune --runner-bin srun --num-procs <number of processes> ---app-run-cmd
 '<application run cmd>' [options]
```

Container With srun

Running NVTAGS with `srun` requires prepending your `application` command with `nvtags tune` and specifying the container type (`pyxis` or `singularity`) that you want to use:

> Note: Ensure that you tell NVTAGS where to find the container image you would like to use.

```
nvtags tune --runner-bin srun --num-procs <number of processes> --container-type
 <singularity or pyxis> --container-img <path to container image> --app-run-cmd
 '<application run cmd>' [options]
```

For Slurm runs, by default, NVTAGS uses `--mpi=pmi2`. If you have a different PMI support, you need to manually set this option by using the `--runner-params` option.

To display the help message for NVTAGS tune options and usage information, run `nvtags tune --help`.

3.2. NVTAGS Run Mode

Here is some information about the NVTAGS Run mode.

In the Run mode, NVTAGS applies the recommended efficient GPU assignment from the tuning process by setting `CUDA_VISIBLE_DEVICES` and executing your application run command. NVTAGS can also pin the CPU and the NIC (or HCA) based on their affinity information and the GPU assignment. For multi-node workloads, NVTAGS selects nodes that use `hostfile` or `SLURM_HOSTID` based on your selected runner binary.

> Note: The Run mode automatically sets the CPU and NIC (or HCA) affinity.

1. Explicitly pass the number of MPI process to NVTAGS by using `--num-proc`.

   ```
   nvtags run --app-run-cmd 'application run cmd' --num-procs N [options]
   ```

2. Explicitly pass the `application run` command to NVTAGS by using `--app-run-cmd`.

   NVTAGS constructs an MPI `run` command by using the provided application `run` command, where the number of processes with the new GPU, the CPU affinity, and the NIC (or HCA) affinity settings are applied.

   Here is an example of how to apply NVTAGS to the `mpirun -np 8 app args` run command:

   ```
   nvtags run --runner-bin mpirun --app-run-cmd 'app args' --num-procs 8 [options]
   ```

To display help messages for the NVTAGS `run` options and usage information, run **`nvtags run --help`**.

3.3. About the NVTAGS CLI

This section provides additional information about the two NVTAGS modes, Tune and Run.

3.3.1. System Profiling Options

Here is some information about the options that are used to modify NVTAGS system profiling parameters.

The system profiling options are `-m, --manual`.

By default, NVTAGS assigns predefined values to system GPU communication channels, which are calculated by using the channels' bandwidth and latency. Table 1 shows the list of GPU links that are recognized by `nvidia-smi` and their corresponding default values. To better represent the strength of the communication links on your system, you can modify these values by setting the environment variable that NVTAGS associates with the link. The environment variable name that is used by NVTAGS is constructed by adding `NVTAGS_PROF_` to the name of the link. For example, `NVTAGS_PROF_SYS` is used to change the SYS default link value, and `NVTAGS_PROF_NV1` is used to change the NV1 default link value.

3.3.2. Supported Link Names

This table provides a list of the supported link names and their default values.

Table 1. List of Supported Link Names in NVTAGS and their Default Value and Description

Link Name	Link Description	Link Value
SYS	Connection traversing the PCIe and SMP interconnect between NUMA nodes (Inter-socket)	10
NODE	Connection traversing the PCIe and interconnect between Host Bridges in a NUMA node	19
PHB	Connection traversing PCIe and a PCIe Host Bridge	18
PXB	Connection traversing multiple PCIe bridges without traversing the Host Bridge	20
PIX	Connection traversing a maximum of one PCIe bridge	20

Link Name	Link Description	Link Value
NV1	Connection traversing a bonded set of 1 NVLinks	25
NV2	Connection traversing a bonded set of 2 NVLinks	25
NV3	Connection traversing a bonded set of 3 NVLinks	25
NV4	Connection traversing a bonded set of 4 NVLinks	25
NV5	Connection traversing a bonded set of 5 NVLinks	25
NV6	Connection traversing a bonded set of 6 NVLinks	25
NV12	Connection traversing a bonded set of 6 NVLinks	25
NET	Connection traversing a inter-node link (for example, InfiniBand, Ethernet, and so on).	8

After you set your new values to the link names by using their environment variables, use the `--manual` argument so their values can be applied by NVTAGS.

Although NV1 to NV12 have different bandwidth capacities, experiments on various systems and MPI applications shows that using the same value for all NVLinks leads to better mapping results. The mapping algorithm uses this value to select NVLinks, over non-NVLinks instead of selecting NVLinks with different bonded sets over each other. If it does not apply to your application and/or system, you can use a manual assignment to change the default link values. For multi-node runs, NVTAGS assumes that network connections will provide worse performance than all intra-node links. If this assumption is not true for your cluster, you can improve the link value for your network connections use `NVTAGS_PROF_NET` and run NVTAGS with the `--manual` flag to manually set the `NET` value. On DGX servers with FDR socket interconnect, setting NET to 20 typically provides good results.

3.3.3. Application Profiling Options

This section provides information about the options that are used to modify the application profiling parameters.

`-d, --disable-normalized`

By default, NVTAGS normalizes raw application GPU communication pattern values, represented in bytes, because some mapping algorithms work better when normalized values are used. To disable this feature, and use raw communication pattern values, pass `--disable-normalize` (or `-d`) to the NVTAGS Tune command.

`-e, --enable-symmetric`

This option allows you to make application profiling values symmetric. By default, application communication patterns are not symmetric, but sometimes mapping algorithms can find a better solution if a symmetric profiling value is used.

`-z, --normalized-value <value>`

The default normalization value is 100, which results in scaling raw GPU communication data that ranges between 0 and 100. You can change this default normalization value by using the `--normalized-value (or -z)` argument with the new value.

`-u, --use-full-prof`

Only relevant for multi-node runs. By default, NVTAGS assumes that all nodes share the same GPU communication topology and performs a quick system topology extraction only on a node with `rank0`. If GPU communication topology varies from node to node in your cluster, you can enable profiling each node in your multi-node workload with `--use-full-prof`.

> Note: This command might add some performance overhead to the NVTAGS tune step.

3.3.4. Tuning with Binding

To tune with binding, run `-y, --with-bind`.

By default, NVTAGS tunes applications as is and applies bindings as a part of the NVTAGS run step. However, you can opt to allow NVTAGS to perform automatic affinity settings for your application while tuning and looking for better resource selection. This process can potentially speed up the tune step.

> Note: To use NVTAGS tuning with binding, your application should not use any explicit binding mechanism, because this mechanism will create a conflict with NVTAGS automatic bindings.

3.3.5. Tuning with Dynamic Profiling

To tune with dynamic profiling, run `-f, --dynamic-prof`.

NVTAGS supports dynamic profiling to achieve more accurate profiling that is tuned for the application and system that you are using. In some cases, when dynamic profiling is run with NVTAGS, it can result in substantially better performance because the process provides a more complete picture of your application and system to NVTAGS.

Dynamic profiling imposes an extra overhead, usually between 10 to 20 seconds, and is disabled by default. When dynamic profiling enabled, NVTAGS performs GPU communication performance analyses on all detected links by using adapted OSU Latency and Bandwidth benchmarks. This way, NVTAGS uses GPU communication performance values to represent the strength of the GPU communications channels of the underlying system.

Without dynamic profiling, predefined values will be used to represent detected communication channels. Dynamic profiling also tracks the GPU message range that is used in application MPI calls and exploits this range to determine the GPU link strength for this message range.

3.3.6. Mapping Options

These mapping group options can be used to modify mapping parameters.

`-i, --improvement-threshold`

NVTAGS uses a congestion metric to compare new GPU assignment candidates against your application's default assignment. Only GPU assignments that can improve the default assignment congestion by more than the threshold value are stored. By default, this threshold value is set to 5%, but it can be changed by using the `--improvement-threshold` (or `-i`) argument with the new threshold value. The new value must be between 0 and 100.

`-a, --map-alg <map alg name>`

Here are the options for the *<map alg name>* variable:

- **`greedy`**
- **`rb`**
- **`all`**

Currently, NVTAGS supports the Greedy (**`greedy`**), Recursive-bipartitioning (**`rb`**), and All (**`all`**) mapping algorithms. The `All` mapping algorithm is the default mapping, which is a combination of the `Greedy` and `RB` algorithms. You can change the `All` mapping algorithm to the `RB` or the `Greedy` algorithm by using `--map-alg` (or `-m`) and the mapping name.

`-o, --opt-time time in ms`

By default, NVTAGS spends 1000 ms (1 second) to evaluate and optimize different mapping solutions. If an efficient GPU assignment solution exists, the solution is found during this period. To change this value, use the `--opt-time` (or `-o`) argument with the new optimization period.

3.3.7. CLI Options for the Run Mode

Here is some information about the CLI options that you can use to run NVTAGS in Run mode.

> Tip: To use the NVTAGS Run mode, run `nvtags run`.

For example, to run the `mpirun -np 8 app all other args` command in this run mode, run the following command:

```
nvtags run --runner-bin mpirun --app-run-cmd 'app all other args' --num-procs 8
```

The NVTAGS `run` script does the following:

- Reads the new potential GPU assignment from the NVTAGS cache file or, if the default cache is modified, from a different path.

  See Changing the NVTAGS Cache Path for more information.
- Before starting the `application run` command, sets `CUDA_VISIBLE_DEVICES`.
- Extracts the system affinity information and applies the CPU and NIC (or HCA) affinity.

By default, the NVTAGS `run` script uses 1 thread per process and binds the process to `core`. You can change these default values by using the following NVTAGS `run` options:

`-b, --bind-to <binding target>`
: Specifies the binding target which can be `core` or `socket`. The default binding target is `core`.

`-t, --num-threads <number of threads per process>`
: Provides the number of threads that can be used with each process. If a value is not provided, the value defaults to `OMP_NUM_THREADS`, and if `OMP_NUM_THREADS` is not defined, the value defaults to 1. The provided value will not override `OMP_NUM_THREADS`, so if both `--num-threads` and `OMP_NUM_THREADS` are defined, their values must match.

3.3.7.1. Examples

Here are some examples of running NVTAGS in Run mode with binding.

Example: Use NVTAGS Run with Binding to the Socket

Here is an example where the Run mode is used with binding to `socket` with Slurm:

```
nvtags run --runner-bin srun --app-run-cmd '<app all other args>' --num-procs 8 --
bind-to socket
```

Example: Use NVTAGS Run with Binding to Core using Four Threads per Process

Here is an example where the Run mode is used to bind 4 threads per process to `core` with `mpirun`:

```
nvtags run --runner-bin mpirun --app-run-cmd '<app all other args>' --num-procs 8 --
bind-to core --num-threads 4
```

> Important: NVTAGS will not run when the requested number of threads exceeds the number of available physical CPU cores on the system.

If NVTAGS cannot find a better mapping in the tuning step, running NVTAGS in Run mode will only set the CPU and NIC (or HCA) affinity.

3.4. Generic CLI Options

Here is a list of generic options that can be used with the NVTAGS binary in the Tune and Run modes.

`-k, --num-nodes`

Provides the number of nodes that will be used when running your application. NVTAGS assumes an equal number of processes per node. The number of processes, which is provided via `--num-procs`, must be divisible by the number of nodes.

`-h, --help`

Prints a help message that includes a generic description of how to use NVTAGS and its options. You can use `nvtags tune --help` and `nvtags run --help` to get tune-specific and run-specific help message, respectively.

`-l, --log-level`

Enables debug logs that are, by default, disabled. To enable the logs, use the `--log-level` DEBUG option.

`-v, --version`

Prints the current NVTAGS version.

3.5. Application Launch Options

These options modify how NVTAGS launches your application in the Tune and Run modes.

`-c, --app-run-cmd <application run command>`

Application run command to run with your specified runner. This option should include your program executable and the runtime arguments that need to be passed to each new process.

`-w, --runner-bin <mpirun | mpiexec | srun>`

Binary that you can use to launch your application, and the valid options include `mpirun`, `mpiexec`, and `srun`. The provided binary and the defined `--runner-params` will be used to launch your application.

NVTAGS constructs an MPI or Slurm `run` command from the input arguments that you provide. However, by default, NVTAGS does not include additional MPI or Slurm `run` parameters, such as `--allow-run-as-root`, to the constructed run command. To allow NVTAGS to pass the selected runner additional run parameters, use the `--runner-params` option.

`-p, --runner-params <runner parameters>`

Provides the parameters for the selected runner. For example, if you use the mpirun runner, `'--allow-run-as-root'` can be passed as a parameter to the runner.

3.6. Container Options

NVTAGS supports running bare-metal and containerized applications with Singularity or Pyxis.

`-q|--container-type <singularity or pyxis>`

Currently, NVTAGS supports Singularity (`singularity`) and enroot/pyxis (`pyxis`) container systems. By default, NVTAGS does not assume any container type, and if NVTAGS is used with containers, the container type must be explicitly provided.

`-r, --container-params <Container run parameters>`

Additional container `run` parameters to pass when launching your container. By default, NVTAGS sets `-run --nv` only for singularity containers. By setting this flag, you can append additional singularity parameters to default values.

`-s, --container-img <Container image>`

Specifies the path to the container image that you want to launch. This option is required for all container runs with NVTAGS.

3.7. Changing the NVTAGS Cache Path

NVTAGS uses a cache directory to store and read cached files.

The NVTAGS default cache directory is `./.nvtags/.cache`. By default, NVTAGS stores the profiling and mapping results in the NVTAGS cache directory. If the cached directory does not exist, NVTAGS creates it at the beginning of the tuning process.

The default NVTAGS cache directory is relative to the current working directory, so if you run NVTAGS in Tune and Run modes from different directories, on different machines, or nodes, you need to update the NVTAGS cache path. The NVTAGS default cache directory path can change by setting the NVTAGS cache directory `NVTAGS_CACHE_DIR` environment variable to the path that you specify.

3.8. Using NVTAGS with Different MPI Versions

NVTAGS supports applications that are built with OpenMPI. Generally, NVTAGS should use the same MPI version as the application.

`-x, --mpi-version <MPI version number>`

Selects the NVTAGS MPI version, and NVTAGS currently supports OpenMPI versions 3.0, 3.1, 4.0, and 4.1.

If specified, NVTAGS will look for the library and the binary files that were built for this specific version (X.Y) and will exit if it cannot locate the library and the files. If the MPI version is not specified, NVTAGS will default to using version 4.0.

Chapter 4. NVTAGS Examples

This section contains information and sample code to help you understand NVTAGS.

4.1. Examples: NVTAGS Tune Mode

The following examples show a variety of tuning options.

Tune

Example 1: Tune `app2` with `dataset2` by using the default tuning options with a normalization value of `50`:

```
nvtags tune --runner-bin mpirun --num-procs 8 --app-run-cmd 'app2 dataset2' --
normalized-value 50
```

Tune with Custom Profiling Options

Example 2: Tune `app3` with `dataset3` by using custom manual system profiling link values. In this example, 4 DGX-1 servers are used with the `NET`, `SYS`, `NV1`, `NV2` link names, and you need to manually assign `1`, `2`, and `4`, and `4` to these names:

```
export NVTAGS_PROF_NET=1
export NVTAGS_PROF_SYS=2
export NVTAGS_PROF_NV1=4
export NVTAGS_PROF_NV2=4
nvtags tune --runner-bin srun --num-procs 32 --app-run-cmd  'app3 dataset3' --manual
```

Tune with Custom Mapping Options

Example 3: Tune `app4` with `dataset4` by using symmetric, no normalization, application profiling with an improvement threshold value of `2.5`%:

```
nvtags tune --runner-bin srun --num-procs 16 --app-run-cmd  'app4 dataset4 app4
 dataset4' --disable-normalized --enable-symmetric --improvement-threshold 2.5
```

Tune with the Custom Cache Path

Example 4: Tune `app` with `dataset` and store the cached data in the `/home/my_cache/` directory:

```
export NVTAGS_CACHE_DIR='/home/my_cache'
nvtags tune --runner-bin mpirun --num-procs 8 --app-run-cmd 'app dataset'
```

If a better GPU mapping is found, it will be stored in the `/home/my_cache/` file instead of the default `./.nvtags/.cache` file.

Tune a Container Image with Custom Runner Parameters

Example 5: Tune a pyxis container from `nvcr.io#path/name:tag` with the `app5` binary and `dataset5` on 32 processes and four nodes.

```
nvtags tune --runner-params "--mpi=pmix" --num-procs 32 --num-nodes 4 --container-
type pyxis --container-img nvcr.io#path/name:tag --app-run-cmd "app5 dataset5"
```

Tune with Dynamic Profiling

Example 6: Tune `app6` with `dataset6` by using the default tuning options and dynamic profiling:

```
nvtags tune --runner-bin mpirun --num-procs 8 --app-run-cmd 'app6 dataset6' --
dynamic-prof
```

Tune the Application by using MPI Version 3.1

Example 7: Tune `app7` with `dataset7` that is using MPI version 3.1:

```
nvtags tune --runner-bin mpirun --num-procs 8 --app-run-cmd 'app7 dataset7' --mpi-
version 3.1
```

4.2. Examples: NVTAGS Run Mode

Here are some examples that show the Run mode with binding.

In this mode, `CUDA_VISIBLE_DEVICES` is set by using the cached mapping file, and the default path for this file is `./nvtags/.cache`. The mapping file content and the extracted system affinity information are used to complete the CPU and NIC affinity settings.

Example 8: Run `app4` with `dataset4` (tuned in *Example 3* in Examples: NVTAGS Tune Mode) by using the default setting. This setting binds the CPUs to core and uses 1 thread per CPU core.

> Note: This example uses the default cache path and mapping name. If you previously set the caching environment variable, ensure that you undo this setting.

```
nvtags run --num-procs 16 --runner-bin srun --app-run-cmd 'app4 dataset4'
```

Example 9: Run `app4` with `dataset4` (tuned in *Example 3* in Examples: NVTAGS Tune Mode) using socket for binding:

```
nvtags run --num-procs 16 --runner-bin srun --app-run-cmd 'app4 dataset4' --bind-to
 socket
```

Example 10: Run `app4` with `dataset4` (tuned in *Example 3* in Examples: NVTAGS Tune Mode) using core binding and 4 threads per process:

```
nvtags run --num-procs 16 --runner-bin srun --app-run-cmd 'app4 dataset4' --bind-to
 core --num-threads 4
```

In this mode, when a better GPU assignment is found from previous step(s) in the `./nvtags/.cache` file, `CUDA_VISIBLE_DEVICES` is set before starting the application command.

Example 11: Run NVTAGS in `run` mode on the following `run` command:

```
mpirun --allow-run-as-root --tag-output -np 8 app5 dataset5
```

```
nvtags run --runner-bin mpirun --app-run-cmd 'app5 dataset5' --num-procs 8 --runner-
params' --allow-run-as-root --tag-output'
```

> Note: The `mpirun` parameters here are `--tag-output` and `--allow-run-as-root`.

Example 12: Run NVTAGS in run mode for 512 processes and 64 nodes using a local `pyxis` container, `./pyxis_img.sqsh`, and include the following container `run` parameters:

```
--container-name=my_container_name --container-mounts=$(pwd):/host_pwd
```

```
nvtags run --runner-bin srun --num-procs 512 --num-nodes 64 --container-type
 pyxis --container-img ./pyxis_img.sqsh --container-params "--container-
name=my_container_name --container-mounts=$(pwd):/host_pwd" --app-run-cmd "app
 dataset"
```

4.3. End-to-End Usage Examples

This section includes an example to complete NVTAGS tuning and running with the Jacobi kernel by using `mpirun`.

4.3.1. NVTAGS with mpirun on Bare Metal

Here are the steps to run NVTAGS with mpirun on bare metal.

Here is the standard Jacobi run command for this example:

```
mpirun -np 8 ./jacobi -t 4 2
```

NVTAGS Tune

1. Run the tuning step to profile your application and system topology:

   ```
   nvtags tune -runner-bin mpirun -app-run-cmd './jacobi -t 4 2' --num-procs 8
   ```

2. Review the logs that indicate by how much communication congestion will improve with the NVTAGS-recommended GPU assignment:

   ```
   NVTAGS: 2020-06-16 08:36:08 info : Better mapping found!
   NVTAGS: 2020-06-16 08:36:08 info : Max Congestion improvement: 0.00%
   NVTAGS: 2020-06-16 08:36:08 info : Avg Congestion improvement: 11.54%
   NVTAGS: 2020-06-16 08:36:08 info : mapping result is stored in "./.nvtags/.cache/
   map.txt"!
   ```

NVTAGS Run

To launch your application with the improved GPU assignment that was recommended by NVTAGS:

```
nvtags run --runner-bin mpirun -app-run-cmd './jacobi -t 4 2' --num-procs 8
```

4.3.2. NVTAGS with srun and a Singularity Container

This section includes an example to complete NVTAGS tuning and running with a LAMMPS singularity container that uses `srun`.

Here is the standard LAMMPS `run` command for this example.

```
srun --mpi=pmi2 -n 16 -N 2 singularity run --nv -B $(pwd):/host_pwd/ ./lammps.simg /
bin/bash -c '/host_pwd/script.sh'
```

where `script.sh` has the following content:

```
#!/bin/bash

cd /host_pwd/;

lmp -k on g 16 -sf kk -pk kokkos -var x 16 -var y 8 -var z 8 -var iterations 300 -in
 in.eam.all.inp
```

NVTAGS Tune

Run the tuning step to profile your application and system topology.

```
nvtags tune --num-procs 16 --runner-params --num-nodes 2 --container-type
 singularity --container-img './lammps.simg' --container-params '-B $(pwd):/
host_pwd' --app-run-cmd '/host_pwd/script.sh'
```

NVTAGS Run

To launch your application with potentially improved GPU assignment and efficient CPU and HCA (or NIC) affinity setting use.

```
nvtags run --num-procs 16 --runner-params --num-nodes 2 --container-type singularity
 --container-img './lammps.simg' --container-params '-B $(pwd):/host_pwd' --app-run-
cmd '/host_pwd/script.sh'
```

Chapter 5. Licensing

This section includes the license for NVTAGS and some third-party licenses.

5.1. NVTAGS License

Here is the license for NVTAGS.

```
Apache License
 Version 2.0, January 2004
 http://www.apache.org/licenses/

 TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

 1. Definitions.

 "License" shall mean the terms and conditions for use, reproduction,
 and distribution as defined by Sections 1 through 9 of this document.

 "Licensor" shall mean the copyright owner or entity authorized by
 the copyright owner that is granting the License.

 "Legal Entity" shall mean the union of the acting entity and all
 other entities that control, are controlled by, or are under common
 control with that entity. For the purposes of this definition,
 "control" means (i) the power, direct or indirect, to cause the
 direction or management of such entity, whether by contract or
 otherwise, or (ii) ownership of fifty percent (50%) or more of the
 outstanding shares, or (iii) beneficial ownership of such entity.

 "You" (or "Your") shall mean an individual or Legal Entity
 exercising permissions granted by this License.

 "Source" form shall mean the preferred form for making modifications,
 including but not limited to software source code, documentation
 source, and configuration files.

 "Object" form shall mean any form resulting from mechanical
 transformation or translation of a Source form, including but
 not limited to compiled object code, generated documentation,
 and conversions to other media types.

 "Work" shall mean the work of authorship, whether in Source or
 Object form, made available under the License, as indicated by a
 copyright notice that is included in or attached to the work
 (an example is provided in the Appendix below).

 "Derivative Works" shall mean any work, whether in Source or Object
 form, that is based on (or derived from) the Work and for which the
 editorial revisions, annotations, elaborations, or other modifications
 represent, as a whole, an original work of authorship. For the purposes
```

```
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
```

```
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS
```

```
APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

5.2. FPMPI License

Here is the third-party license for FPMPI.

```
COPYRIGHT NOTIFICATION
(C) COPYRIGHT 2000 UNIVERSITY OF CHICAGO
This program discloses material protectable under copyright laws of the United
 States. Permission to copy and modify this software and its documentation is
 hereby granted, provided that this notice is retained thereon and on all copies
 or modifications. The University of Chicago makes no representations as to the
 suitability and operability of this software for any purpose. It is provided "as
 is" without express or implied warranty. Permission is hereby granted to use,
 reproduce, prepare derivative works, and to redistribute to others, so long as this
 original copyright notice is retained.
This software was authored by:
William D. Gropp: (630) 252-4318
Mathematics and Computer Science Division
Argonne National Laboratory,
Argonne IL 60439 FAX: (630) 252-5986
Any questions or comments on the software may be directed to fpmpi@mcs.anl.gov.
Argonne National Laboratory with facilities in the states of Illinois and Idaho, is
 owned by The United States Government, and operated by the University of Chicago
 under provision of a contract with the Department of Energy.
DISCLAIMER
THIS PROGRAM WAS PREPARED AS AN ACCOUNT OF WORK SPONSORED BY AN AGENCY OF THE UNITED
 STATES GOVERNMENT. NEITHER THE UNITED STATES GOVERNMENT NOR ANY AGENCY THEREOF,
 NOR THE UNIVERSITY OF CHICAGO, NOR ANY OF THEIR EMPLOYEES OR OFFICERS, MAKES ANY
 WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR
 THE ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT,
 OR PROCESS DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY
 OWNED RIGHTS. REFERENCE HEREIN TO ANY SPECIFIC COMMERCIAL PRODUCT, PROCESS, OR
 SERVICE BY TRADE NAME, TRADEMARK, MANUFACTURER, OR OTHERWISE, DOES NOT NECESSARILY
 CONSTITUTE OR IMPLY ITS ENDORSEMENT, RECOMMENDATION, OR FAVORING BY THE UNITED
 STATES GOVERNMENT OR ANY AGENCY THEREOF. THE VIEW AND OPINIONS OF AUTHORS EXPRESSED
 HEREIN DO NOT NECESSARILY STATE OR REFLECT THOSE OF THE UNITED STATES GOVERNMENT OR
 ANY AGENCY THEREOF.
```

5.3. LipTopoMap License

Here is the license for LipTopoMap.

5.4. Metis License

Here is the license for Metis.

```
 Works of, publicly display, publicly perform, sublicense, and distribute the Work
 and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License,
 each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-
charge, royalty-free, irrevocable (except as stated in this section) patent license
 to make, have made, use, offer to sell, sell, import, and otherwise transfer
 the Work, where such license applies only to those patent claims licensable
 by such Contributor that are necessarily infringed by their Contribution(s)
 alone or by combination of their Contribution(s) with the Work to which such
 Contribution(s) was submitted. If You institute patent litigation against any
 entity (including a cross-claim or counterclaim in a lawsuit) alleging that
 the Work or a Contribution incorporated within the Work constitutes direct or
 contributory patent infringement, then any patent licenses granted to You under
 this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative
 Works thereof in any medium, with or without modifications, and in Source or Object
 form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this
 License; and
You must cause any modified files to carry prominent notices stating that You
 changed the files; and
You must retain, in the Source form of any Derivative Works that You distribute, all
 copyright, patent, trademark, and attribution notices from the Source form of the
 Work, excluding those notices that do not pertain to any part of the Derivative
 Works; and
If the Work includes a "NOTICE" text file as part of its distribution, then
 any Derivative Works that You distribute must include a readable copy of the
 attribution notices contained within such NOTICE file, excluding those notices
 that do not pertain to any part of the Derivative Works, in at least one of the
 following places: within a NOTICE text file distributed as part of the Derivative
 Works; within the Source form or documentation, if provided along with the
 Derivative Works; or, within a display generated by the Derivative Works, if and
 wherever such third-party notices normally appear. The contents of the NOTICE file
 are for informational purposes only and do not modify the License. You may add Your
 own attribution notices within Derivative Works that You distribute, alongside
 or as an addendum to the NOTICE text from the Work, provided that such additional
 attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide
 additional or different license terms and conditions for use, reproduction, or
 distribution of Your modifications, or for any such Derivative Works as a whole,
 provided Your use, reproduction, and distribution of the Work otherwise complies
 with the conditions stated in this License.
5. Submission of Contributions. Unless You explicitly state otherwise, any
 Contribution intentionally submitted for inclusion in the Work by You to the
 Licensor shall be under the terms and conditions of this License, without any
 additional terms or conditions. Notwithstanding the above, nothing herein shall
 supersede or modify the terms of any separate license agreement you may have
 executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names,
 trademarks, service marks, or product names of the Licensor, except as required for
 reasonable and customary use in describing the origin of the Work and reproducing
 the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to
 in writing, Licensor provides the Work (and each Contributor provides its
 Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 KIND, either express or implied, including, without limitation, any warranties or
 conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR
 PURPOSE. You are solely responsible for determining the appropriateness of using
 or redistributing the Work and assume any risks associated with Your exercise of
 permissions under this License.
```

```
8. Limitation of Liability. In no event and under no legal theory, whether in tort
 (including negligence), contract, or otherwise, unless required by applicable law
 (such as deliberate and grossly negligent acts) or agreed to in writing, shall any
 Contributor be liable to You for damages, including any direct, indirect, special,
 incidental, or consequential damages of any character arising as a result of this
 License or out of the use or inability to use the Work (including but not limited
 to damages for loss of goodwill, work stoppage, computer failure or malfunction, or
 any and all other commercial damages or losses), even if such Contributor has been
 advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or
 Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance
 of support, warranty, indemnity, or other liability obligations and/or rights
 consistent with this License. However, in accepting such obligations, You may
 act only on Your own behalf and on Your sole responsibility, not on behalf of
 any other Contributor, and only if You agree to indemnify, defend, and hold each
 Contributor harmless for any liability incurred by, or claims asserted against,
 such Contributor by reason of your accepting any such warranty or additional
 liability.

END OF TERMS AND CONDITIONS
```

5.5. SPDLOG License

Here is the SPDLOG license.

```
The MIT License (MIT)

Copyright (c) 2016 Gabi Melman.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

5.6. OMB License

Here is the OSU Micro Benchmark (OMB) license.

```
                         COPYRIGHT

Copyright (c) 2001-2021, The Ohio State University. All rights
reserved.

The OMB (OSU Micro Benchmarks) software package is developed by the team
members of The Ohio State University's Network-Based Computing Laboratory
(NBCL), headed by Professor Dhabaleswar K. (DK) Panda.

Contact:
```

```
Prof. Dhabaleswar K. (DK) Panda
Dept. of Computer Science and Engineering
The Ohio State University
2015 Neil Avenue
Columbus, OH - 43210-1277
Tel: (614)-292-5199; Fax: (614)-292-2911
E-mail:panda@cse.ohio-state.edu

This program is available under BSD licensing.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

(1) Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

(2) Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

(3) Neither the name of The Ohio State University nor the names of
their contributors may be used to endorse or promote products derived
from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

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

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