Hands On Session with Nsight Systems and Nsight Compute

By: Brett Neuman & Daniel Howard, Consulting Services Group, CISL & NCAR

bneuman@ucar.edu (mailto:bneuman@ucar.edu) & dhoward@ucar.edu (mailto:dhoward@ucar.edu)

Date: June 16th, 2022

In this notebook we explore profiling of the mini-app MiniWeather to present profiling techniques and code examples. We will cover:

- Overview of Profiling and Performance Sampling Tools
    - Typical development workflows with profiling tools
- NSight Compute for Individual GPU Kernel Performance Analysis
    - How to generate ncu reports and command line parameters
    - Overview of GPU kernel profiling data and source code timing heatmaps
    - External resources for interpreting ncu reports data

Head to the NCAR JupyterHub portal (https://jupyterhub.hpc.ucar.edu/stable) and **start a JupyterHub session on Casper login** (or batch nodes using 1 CPU, no GPUs) and open the notebook in `10_HandsOnNsight/nsys/10_HandsOnNsight_nsys.ipynb`. Be sure to clone (if needed) and update/pull the NCAR GPU_workshop directory.

```
# Use the JupyterHub GitHub GUI on the left panel or the below shell commands
git clone git@github.com:NCAR/GPU_workshop.git
git pull
```

Workshop Etiquette

- Please mute yourself and turn off video during the session.
- Questions may be submitted in the chat and will be answered when appropriate. You may also raise your hand, unmute, and ask questions during Q&A at the end of the presentation.
- By participating, you are agreeing to UCAR's Code of Conduct (https://www.ucar.edu/who-we-are/ethics-integrity/codes-conduct/participants)
- Recordings & other material will be archived & shared publicly.
- Feel free to follow up with the GPU workshop team via Slack or submit support requests to support.ucar.edu (https://support.ucar.edu)
    - Office Hours: Asynchronous support via Slack (https://ncargpuusers.slack.com) or schedule a time with an organizer

Notebook Setup

Set the `PROJECT` code to a currently active project, ie `UCIS0004` for the GPU workshop, and `QUEUE` to the appropriate routing queue depending on if during a live workshop session (`gpuworkshop`), during weekday 8am to 5:30pm MT (`gpudev`), or all other times (`casper`). Due to limited shared GPU resources, please use `GPU_TYPE=gp100` during the workshop. Otherwise, set `GPU_TYPE=v100` (required for `gpudev`) for independent work. See [Casper queue documentation (https://arc.ucar.edu/knowledge_base/72581396#StartingCasperjobswithPBS-Concurrentresourcelimits)](https://arc.ucar.edu/knowledge_base/72581396#StartingCasperjobswithPBS-Concurrentresourcelimits) for more info.

In [ ]:
```
export PROJECT=UCIS0004
export QUEUE=gpudev
export GPU_TYPE=gp100

module load nvhpc/22.5 openmpi &> /dev/null
export PNETCDF_INC=/glade/u/apps/dav/opt/pnetcdf/1.12.3/openmpi/4.1.4/nvhpc/22.
5/include
export PNETCDF_LIB=/glade/u/apps/dav/opt/pnetcdf/1.12.3/openmpi/4.1.4/nvhpc/22.
5/lib
```

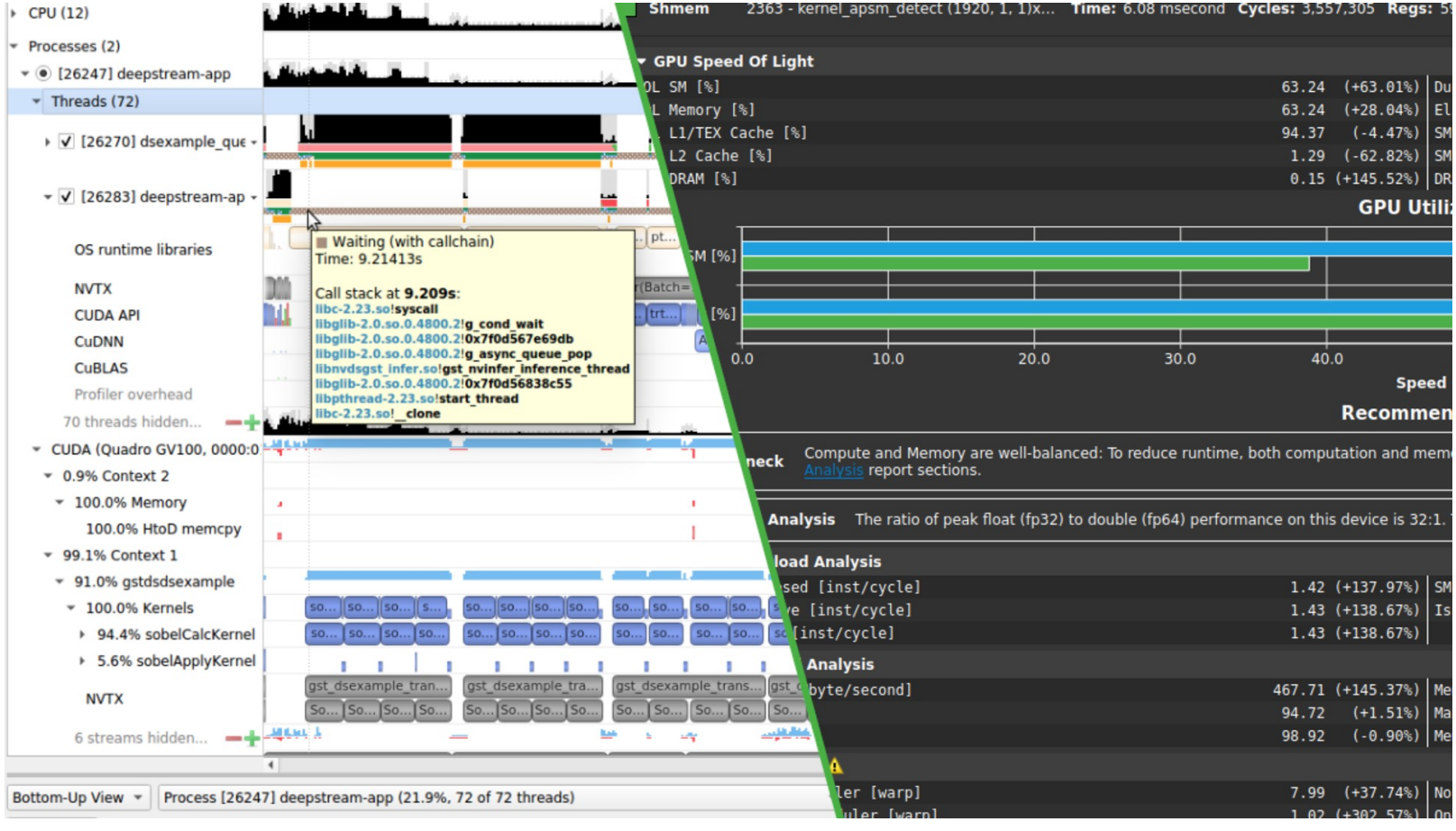
What is a Profiler?

Profilers are tools that **samples** and measure performance characteristics of an executable across its runtime. This information is intended to aid program optimization and performance engineering.

Profiler software that are supported at NCAR include **Arm Map**, **Nsight Systems**, and **Nsight Compute**. All of these tools are able to analyze GPU code. Other profilers you may be aware of include TAU, Intel VTune Advisor, HPC Toolkit, and Vampir.

Today, we will focus on the NVIDIA Nsight profiling tools and usage techniques of these tools.

- **Nsight Systems** - Provides a high level runtime and trace analysis of the program runtime via a measured timeline of various metrics and GPU kernels across a program.
- **Nsight Compute** - Provides an in depth level assessment of individual GPU kernel performance and how various GPU resources are utilized across many different metrics.



Nsight Systems (left) shows a timeline of code runtime.

Nsight Compute (right) records and presents extensive performance statistics for individual kernels.

Profiling Documentation Resources

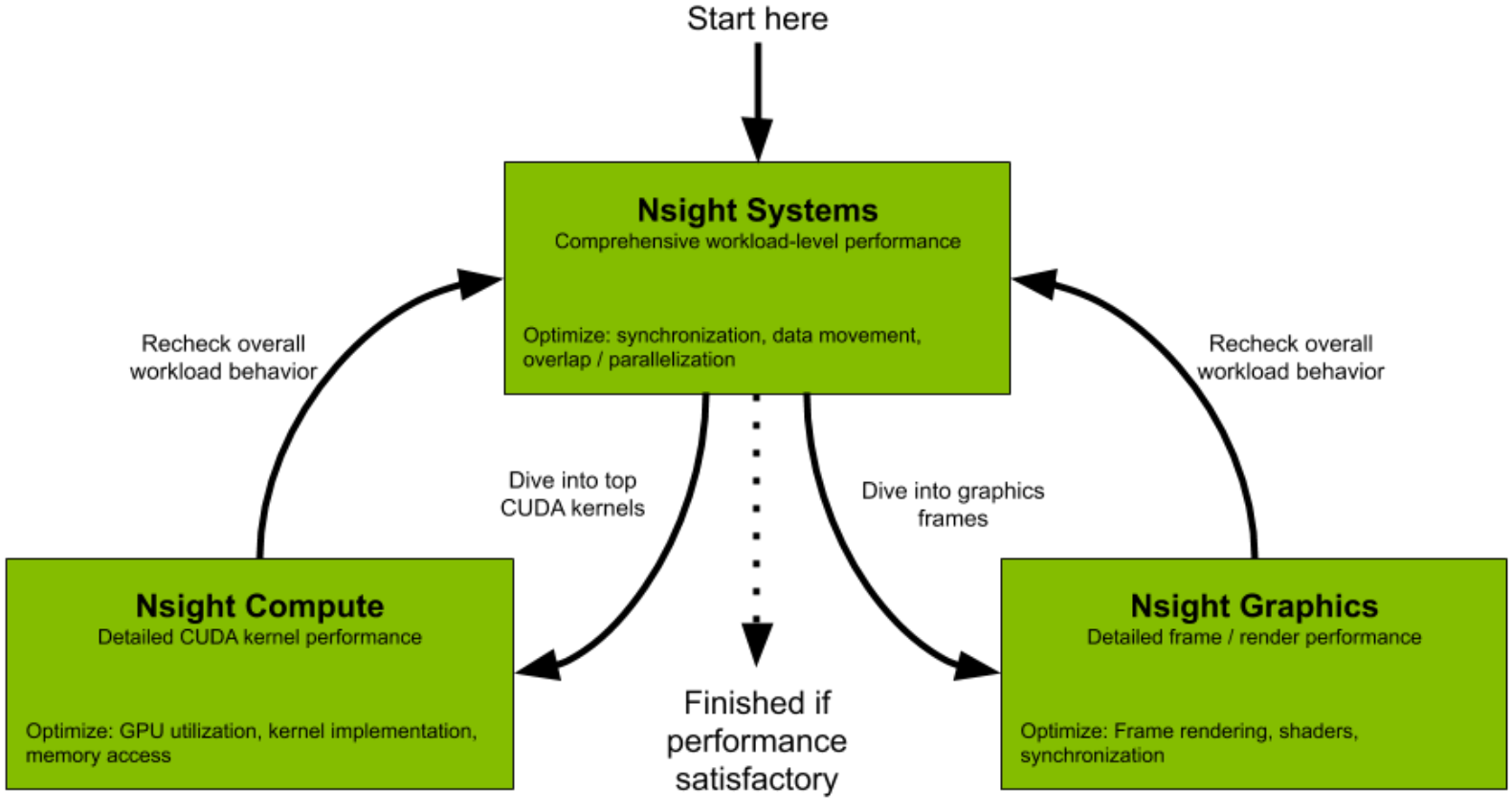
NVIDIA provides extensive documentation for each of these profilers. We will go over basic usage of these tools but to learn more and get the most out of Nsight, consult the below resources:

- Nsight Systems Main Documentation (https://docs.nvidia.com/nsight-systems)
- Nsight Compute Main Documentation (https://docs.nvidia.com/nsight-compute/)
- Nsight Compute Profiling Guide (https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html)
- Nsight Compute Training Resources (https://docs.nvidia.com/nsight-compute/Training/index.html) - Forum, Videos, and Blog Posts curated by NVIDIA

An excellent interactive step-by-step tutorial given by Max Katz (NVIDIA) using Nsight Compute to optimize an OpenACC kernel in the BerkeleyGW many-body perturbation theory software can be found at this Gitlab repository (https://gitlab.com/NERSC/roofline-on-nvidia-gpus). A recorded video on this material is here (https://www.youtube.com/watch?v=fsC3QeZHM1U).

Additionally, the CLI help pages via the `-h` flag for each profiler is a useful quick reference point. Run the below cells to view them.

Profiling Workflow



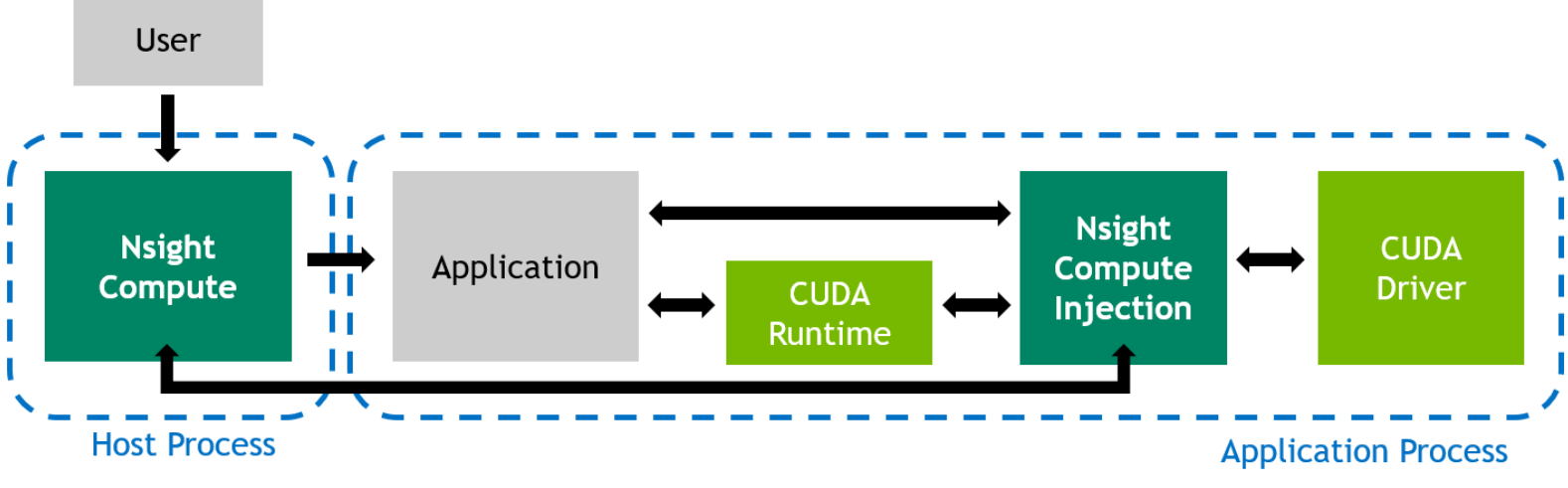
When assessing performance of software, first profile the overall program with Nsight Systems. Then, expensive kernels can be identified and profiled using Nsight Compute.

Iteratively analyze and modify code to optimize performance, up to the amount of effort is worthwhile.

Nsight Compute

After getting a sense of the overall performance of your program with Nsight Systems, use Nsight Compute to dive deeper into the performance of individual GPU kernels.

- **CUDA kernel profiler** (or CUDA kernels generated by OpenACC/OpenMP/Kokkos code)
- Curates **performance statistics** into targeted metrics sections
- Able to **select amount of data to collect** and how it's presented
    - More detailed analysis has greater **overhead with profiler usage**
- Fully featured **Command Line** and **User Friendly GUI** interfaces
- Regularly updated and customizable Python based rules for **guided analysis** and post-processing



Preparing Code for Nsight Compute

When preparing code for Nsight Compute, an important compile option to add is `-gpu=lineinfo`. **DON'T USE `-pg`, `-g`, or `-G` flags**. The lineinfo flag allows the Source/SASS analysis section of Nsight Compute correlate performance information with specific lines of CUDA and/or OpenACC/OpenMP code.

Use the below cell to compile and re-compile MiniWeather after code changes are made. You may also modify the runtime parameters, grid size, and simulation time to investigate how different problem sizes impact performance. Review the generated GPU kernel specifications from the `-Minfo=acc` output.

In [ ]:
```
export OPENACC_FLAGS="-acc -gpu=cc60,cc70,lineinfo"

mpif90 -I${PNETCDF_INC} -Mextend -O0 -DNO_INFORM -c miniWeather_mpi_openacc.F90
-o miniWeather_mpi_openacc.F90.o \
-D_NX=9192 -D_NZ=4096 -D_SIM_TIME=0.1 -D_OUT_FREQ=2.0 -D_DATA_SPEC=DATA_SPEC_TH
ERMAL ${OPENACC_FLAGS} -Minfo=acc

mpif90 -Mextend -O3 miniWeather_mpi_openacc.F90.o -o openacc -L${PNETCDF_LIB} -
lpnetcdf ${OPENACC_FLAGS}
rm -f miniWeather_mpi_openacc.F90.o
```

Notably, only a short simulation time (enough to cover a few timesteps) is required for us to effectively analyze and optimize model performance.

Nsight Compute CLI Options

- `-o <report-name>` - Writes output to a `*.ncu-rep` file to analyze via GUI
    - Without `-o`, analysis is summarized in stdout.
- `-f` - Force overwrite of output files
- `-c` or `--launch-count` - Specifies the number of kernel launches to profile. Otherwise, all launched kernels are profiled
- `-s` or `--launch-skip` - Skips a specified number of kernel launches. Useful for letting the GPU "warm-up"
- `--set <arg>` - Sets the amount of data collected and kernel metrics measured, i.e. `detailed`, `full`, or others given from `--list-sets` flag
    - More data collected requires more redundant runs of GPU kernels and increases profiler overhead
- `-k` or `--kernel-name` - Specifies the exact name (see `nsys`) of kernels to be profiled
    - Use `-k regex:<expression>` to filter kernels by a regex expression
- `--nvtx` - Enables support for NVTX ranges
- `--nvtx-include arg` - Filters profiled kernels based on NVTX ranges
- `--import-source on` - Imports CUDA/source code directly into the report.

Generate Nsight Compute Report

Start with the final version of miniWeather_mpi_openacc.F90 (miniWeather_mpi_openacc.F90). As we analyze performance, use the generated report to inform code optimizations to experiment with.

First, use the submit script ncu_bash.sh (ncu_bash.sh) to run Nsight Compute on MiniWeather by running command `ncu <ncu options> <exec> <exec arguments>`. Useful `ncu` options are listed above but also may be reviewed via `ncu -h`.

The first profile run of MiniWeather will profile all kernels using `--sets full` in order to make a baseline (requires redundantly running kernels 73-74 times). When changing code, modify the report filename when you re-run the below cell to help you keep track of reports between different code versions.

In [ ]:
```
qsub -q $QUEUE -l gpu_type=$GPU_TYPE -A $PROJECT -v NCU_REPORT="MW_DivToMult" n
cu_bash.sh
```

`SHIFT + right click` MW_baseline.ncu-rep (MW_baseline.ncu-rep) in order to save the Nsight Compute report to your personal machine (or download the file from the left pane explorer). Use your local Nsight Compute client to open the file. Alternatively, after setting `module load nvhpc`, you can run `ncu-ui <report-name>` over a terminal X session or VNC/FastX session on Casper.

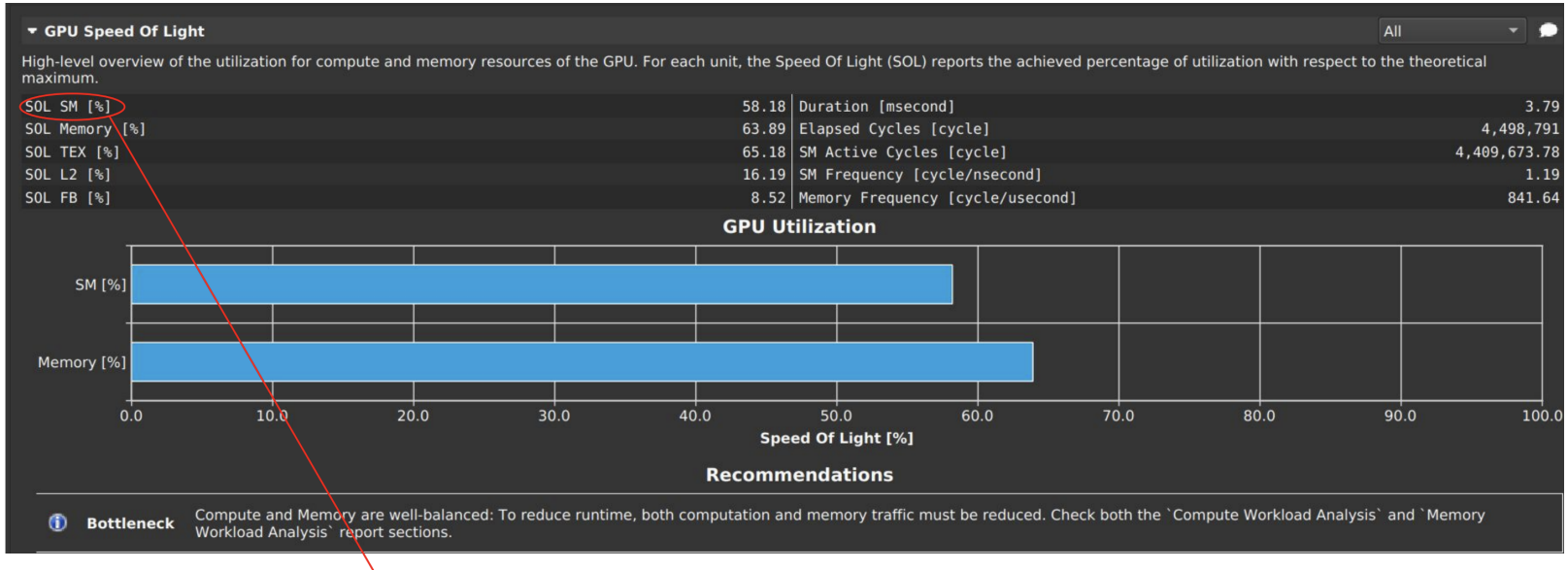
Analysis of Nsight Compute Profiles

Depending on the option chosen for `--set` and number of metrics measured, the kernel profiling report will contain a selection of different sections for review covering performance metrics of each kernel profiled.

When using the GUI, **guided analysis** as alerted via exclamation point warning signs will suggest specific issues the profiler identifies and tries to suggest solutions. These are automatically triggered Python rules written by Nsight Compute maintainers and experts, which can be further customized or added to. If you need help interpretting this information, hover your mouse over a piece of information and an informative text box will appear to explain.

Below, we review a few important sections.

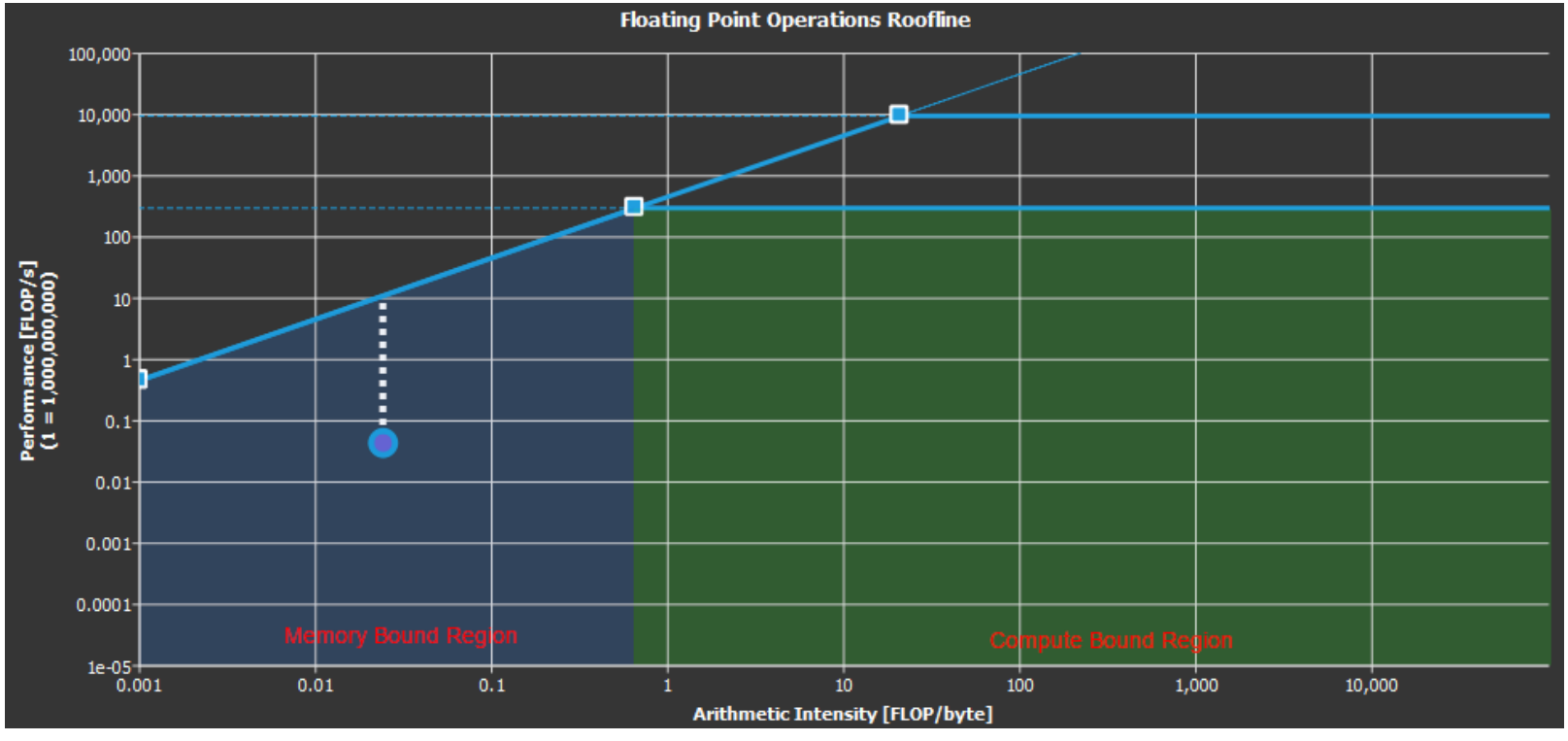
Nsight Compute - GPU Speed of Light



Mouse over each to see the associated metric

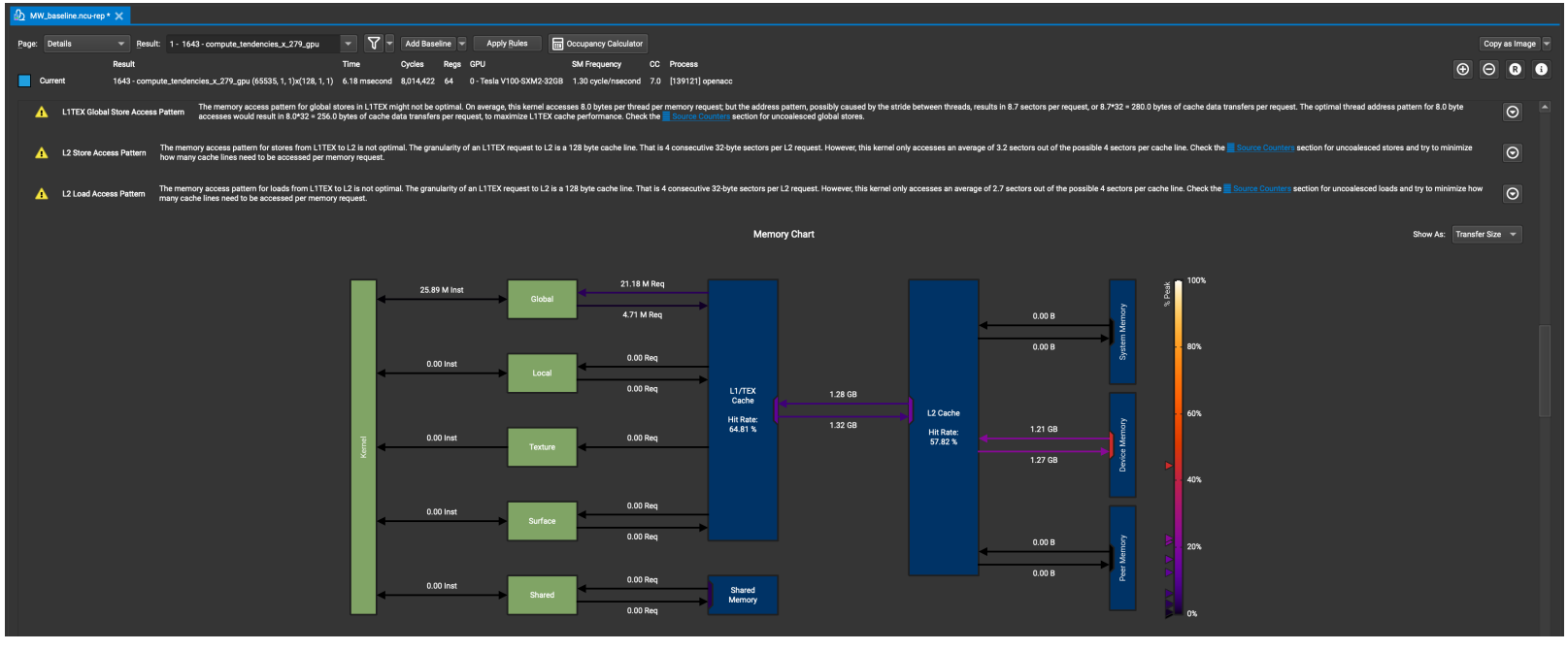
The GPU Speed of Light section highlights to what percentage is this kernel using the full capability of the GPU, both in terms of Streaming Multiprocesser (SM) occupancy and Memory Throughput.

Nsight Compute - Roofline Analysis



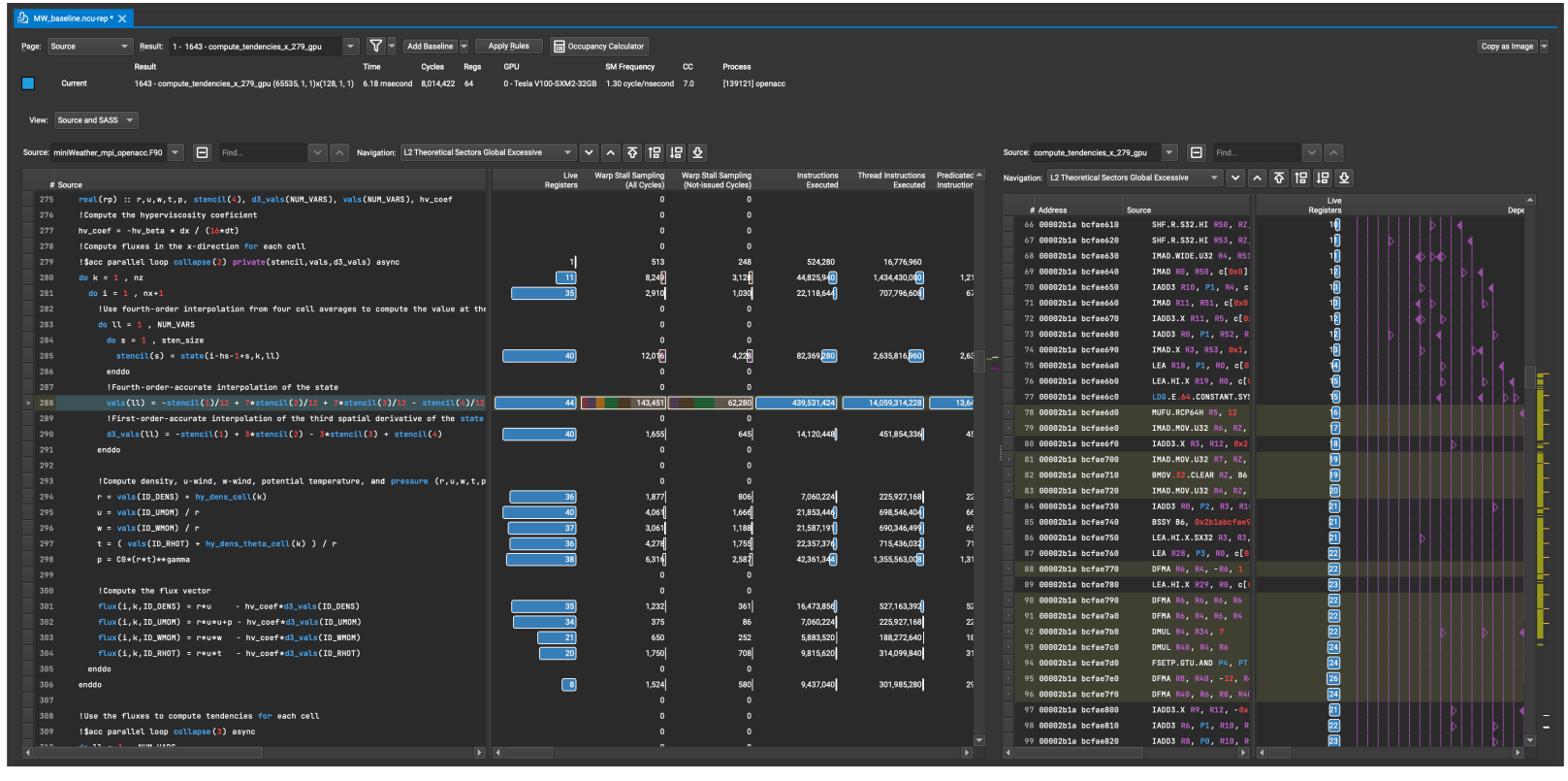
With at least `--set detailed`, a roofline plot is generated, used to determine if the kernel is **compute bound** or **memory bound**. Memory bound kernels can perhaps benefit by assigning more compute operations per thread if possible. Compute bound kernels will likely require further analysis for optimization, typically by checking for warp stalls or coallesced memory issues.

Nsight Compute - Memory Workload Analysis



This section provides a detailed analysis of the memory resources of the GPU. In this case, Nsight Compute identifies that there is an imbalance of data movement between the L1 and L2 caches due to uncoalesced memory. To improve this, memory access patterns need to be re-designed within the source code and OpenACC kernel.

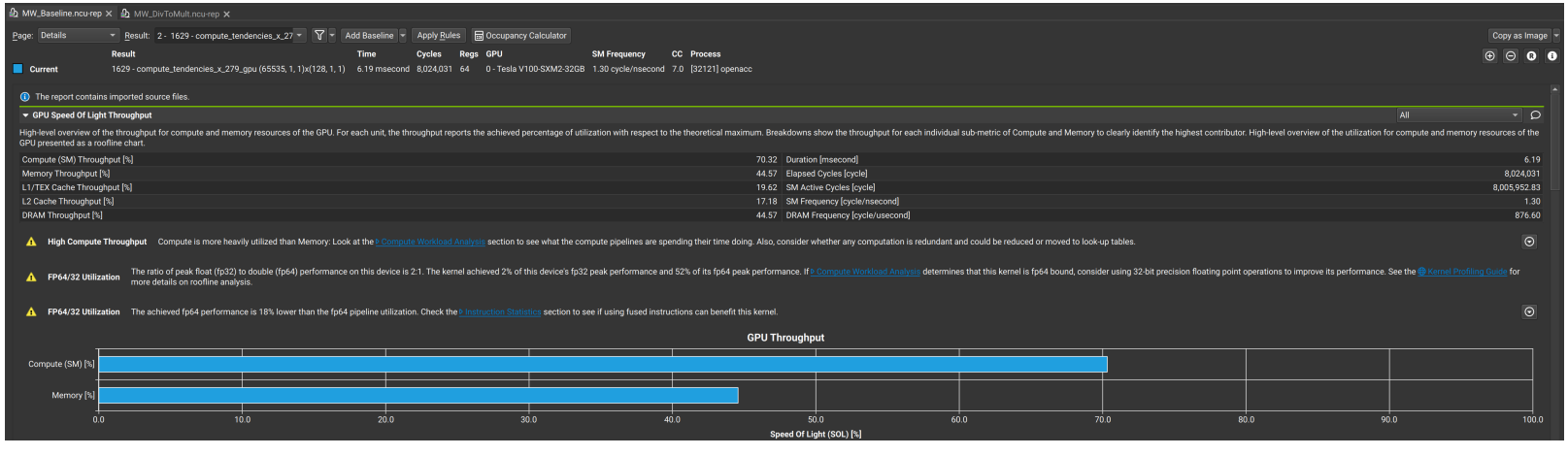
Nsight Compute - Source/SASS and Instruction Hotspots



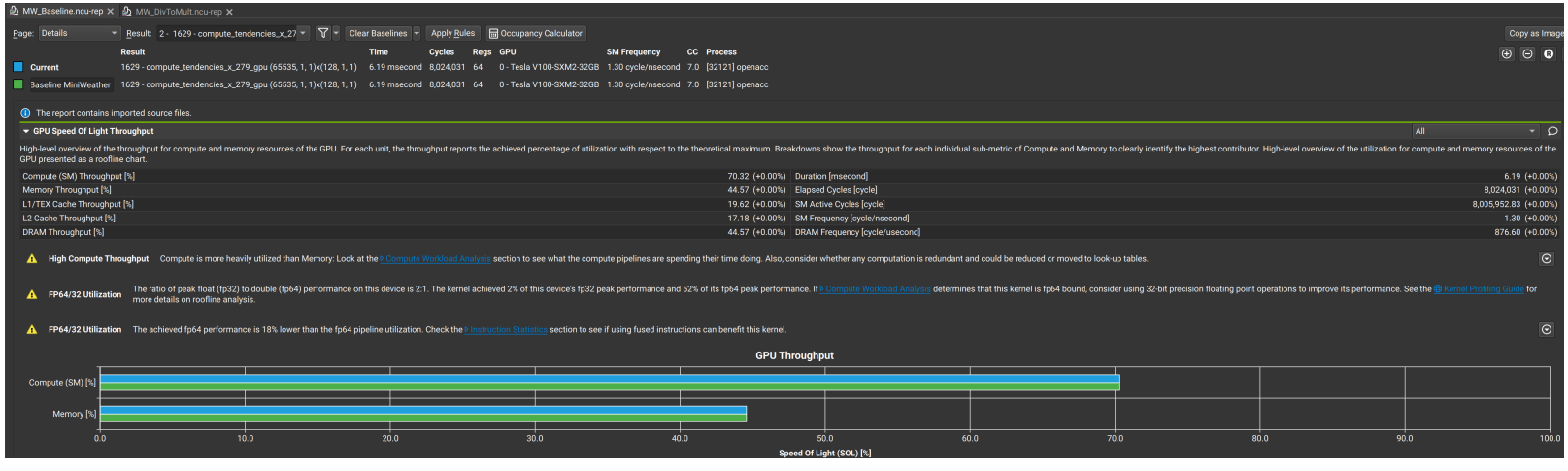
Navigated to via the **Source Counters** section, a heatmap of resource usage and other metrics can be correlated to specfic lines of code within the source files. This can more easily identify which specific areas of your program are causing poor performance.

Nsight Compute - Add a Baseline

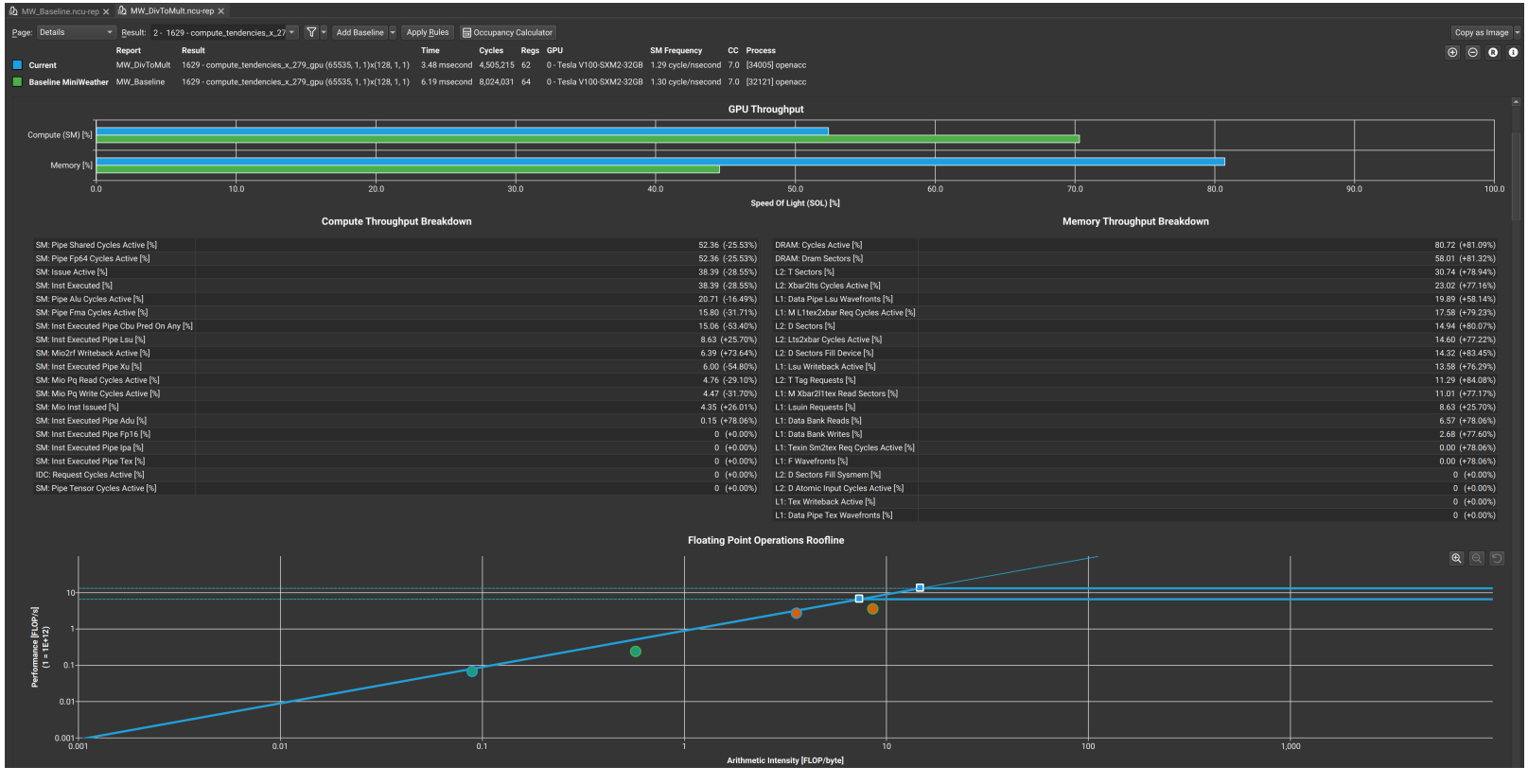
Whenever profiling a program or specific kernel, it is vitally important to record and **set a baseline** to reference performance changes against. In Nsight Compute, set a baseline by clicking **Add Baseline** near the top of the main window within the Nsight Compute GUI. Note, you can add multiple "baselines" from multiple reports.



Rename a baseline by clicking the **Baseline #** text label.



Now, **open the new profile report** or switch to the other tab referencing this report. The baseline performance metrics will now be displayed and compared to the new current report's performance metrics.

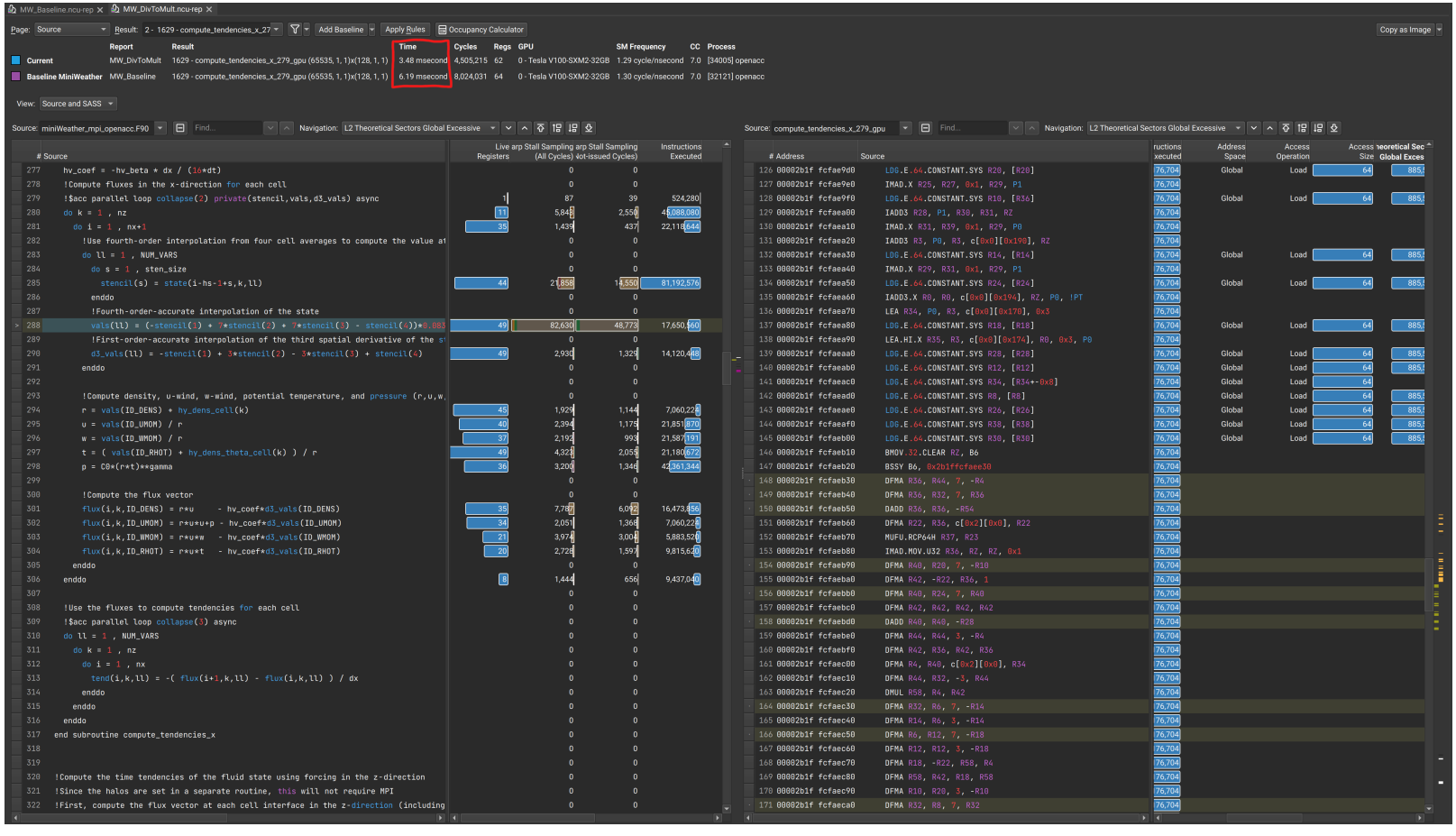


Experiment with a Proposed Optimization - Replace Divide with Multiply

Noting the **hotspot at line 288**, we can assess if there's a way to re-formulate this line to either reduce redundant operations or refactor the overall algorithm. The metrics provided may be able to provide a hint towards why this line is a bottleneck for MiniWeather.

In this case, there are a significant number of warp stalls (see here (https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html#statistical-sampler) for descriptions of types of warp stalls) as well as a much higher number of instructions executed compared to other lines in this kernel. Looking at this line, we see multiple divisions by 12 that could be simplified. Additionally, division is typically more expensive than multiplication within IEEE computational arithmetic.

Thus, let's try changing this line to `vals(ll) = (-stencil(1) + 7*stencil(2) + 7*stencil(3) - stencil(4))*0.08333333333333333`. The report that analyses this change is MW_DivToMult.ncu-rep (MW_DivToMult.ncu-rep).


MW_Baseline.ncu-rep
MW_DivToMult.ncu-rep
Page: Source
Result: 2 - 1629 - compute_tendencies_x_27
Add Baseline
Apply Rules
Occupancy Calculator
Copy as Image
Report Result Time Cycles Regs GPU SM Frequency CC Process
Current MW_DivToMult 1629 - compute_tendencies_x_279_gpu (65535, 1, 1)x(128, 1, 1) 3.48 msecond 4,505,215 62 0 - Tesla V100-SXM2-32GB 1.29 cycle/nsecond 7.0 [34005] openacc
Baseline MiniWeather MW_Baseline 1629 - compute_tendencies_x_279_gpu (65535, 1, 1)x(128, 1, 1) 6.19 msecond 8,024,031 64 0 - Tesla V100-SXM2-32GB 1.30 cycle/nsecond 7.0 [32121] openacc
View: Source and SASS
Source: miniWeather_mpi_openacc.F90
Find...
Navigation: L2 Theoretical Sectors Global Excessive
Source: compute_tendencies_x_279_gpu
Live arp Stall Sampling (All Cycles)
arp Stall Sampling (Not-issued Cycles)
Instructions Executed
Source
Address
Source
Address Space
Access Operation
Global Excess

EXERCISE - Adjust MiniWeather Problem Size and Other Optimizations

Adjust MiniWeather's problem size using the values `nx=128,512,1024,2048,4096,9192` with `nz=nx/2`. Try more problem sizes if interested. Generate `ncu` reports for each of these problem sizes.

Then, open up all the reports and add each one as a named baseline for that problem size. Compare performance between problem sizes.

1. **Describe the performance for small problem sizes? What is the SM utilization and memory throughput for small problems?**
2. **Is there an optimal problem size?**
3. **Do performance or other metrics stop changing after a certain order of magnitude for the problem size?**
4. Experiment with and attempt other optimizations/code changes to improve MiniWeather's performance. **What other ways or styles of refactoring might you try to improve performance?**

Resources

- Nsight Systems Main Documentation (https://docs.nvidia.com/nsight-systems)
- Nsight Compute Main Documentation (https://docs.nvidia.com/nsight-compute/)
- Nsight Compute Profiling Guide (https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html)
- Nsight Compute Training Resources (https://docs.nvidia.com/nsight-compute/Training/index.html) - Forum, Videos, and Blog Posts curated by NVIDIA
- Introduction to Kernel Performance Analysis with NVIDIA Nsight Compute, Max Katz (NVIDIA invited to Argonne/NERSC)
    - GitLab repo (https://gitlab.com/NERSC/roofline-on-nvidia-gpus) and video (https://www.youtube.com/watch?v=fsC3QeZHM1U)

Return to Nsight Systems Profiler Tool

Nsight Systems Profiler (../nsys/10_HandsOnNsight_nsys.ipynb)