

1 Profiling with NVIDIA Tools

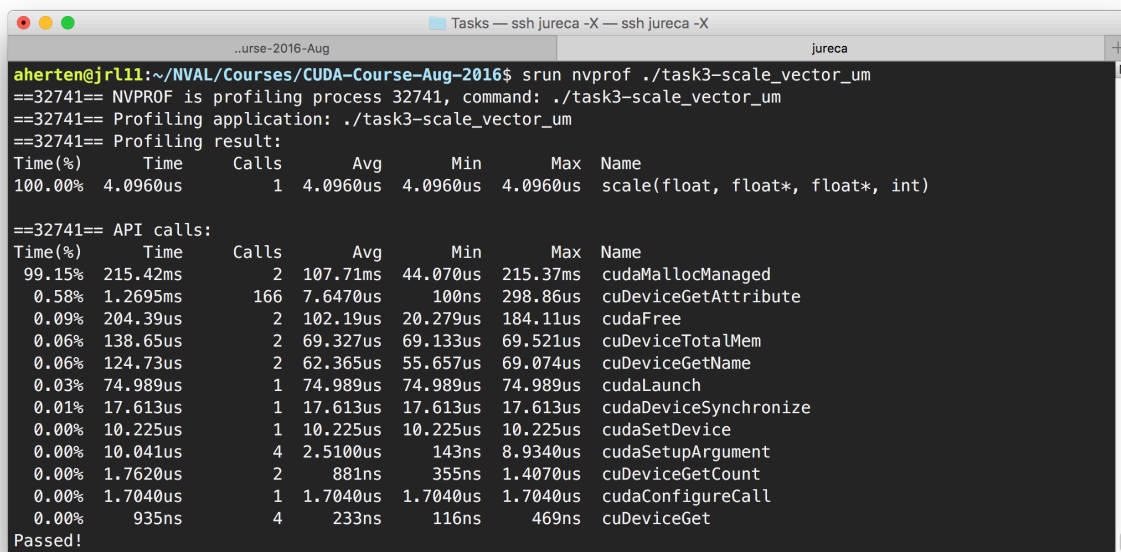
The CUDA Toolkit comes with two solutions for profiling an application: `nvprof`, which is a command line program, and the GUI application *NVIDIA Visual Profiler* (NVVP).

`nvprof` can be used in batch jobs or smaller interactive runs; NVVP can either import an `nvprof`-generated profile or run interactively through X forwarding.¹

On JURON, the CUDA Toolkit can be loaded by `module load nvidia/cuda/8.0`; on JURECA, load it by `module load CUDA`.

1.1 Command Line: `nvprof`

For a quick overview of the GPU-invocations of an application, prefix its run with `nvprof`: `nvprof ./APP`.² `nvprof` instruments the application at run-time.



```
aherten@jrl11:~/NVAL/Courses/CUDA-Course-Aug-2016$ srun nvprof ./task3-scale_vector_um
==32741== NVPROF is profiling process 32741, command: ./task3-scale_vector_um
==32741== Profiling application: ./task3-scale_vector_um
==32741== Profiling result:
Time(%)   Time      Calls      Avg      Min      Max      Name
100.00%   4.0960us    1   4.0960us  4.0960us  4.0960us  scale(float, float*, float*, int)

==32741== API calls:
Time(%)   Time      Calls      Avg      Min      Max      Name
99.15%   215.42ms    2   107.71ms  44.070us  215.37ms  cudaMallocManaged
0.58%    1.2695ms  166    7.6470us  100ns    298.86us  cuDeviceGetAttribute
0.09%    204.39us    2   102.19us  20.279us  184.11us  cudaFree
0.06%    138.65us    2    69.327us  69.133us  69.521us  cuDeviceTotalMem
0.06%    124.73us    2    62.365us  55.657us  69.074us  cuDeviceGetName
0.03%    74.989us    1    74.989us  74.989us  74.989us  cudaLaunch
0.01%    17.613us    1    17.613us  17.613us  17.613us  cudaDeviceSynchronize
0.00%    10.225us    1    10.225us  10.225us  10.225us  cudaSetDevice
0.00%    10.041us    4    2.5100us  143ns    8.9340us  cudaSetupArgument
0.00%    1.7620us    2      881ns    355ns    1.4070us  cuDeviceGetCount
0.00%    1.7040us    1    1.7040us  1.7040us  1.7040us  cudaConfigureCall
0.00%    935ns      4      233ns    116ns    469ns    cuDeviceGet
Passed!
```

Among the many options of `nvprof` (see `nvprof --help`) it can export a profile for further usage through NVVP: `nvprof --export-profile FILE ./APP` will export the profile to `FILE`.

To make use of NVVP performance experiments, certain metrics need to be measured by `nvprof`: `nvprof --analysis-metrics --export-profile FILE ./APP` will export the metrics to `FILE`.

Further options of potential interest:

- `--print-gpu-trace`: Show trace of function calls
- `--openacc-profiling on`: Profile OpenACC as well (`on` by default)
- `--cpu-profiling on`: Enable some CPU profiling
- `--csv --log-file FILE`: Generate CSV output and save to `FILE`; handy for plots or benchmarked analysis
- `--metrics M1`: Measure only metric `M1` which is one of the NVIDIA-provided metrics which can be listed via `--query-metrics`.

→ docs.nvidia.com/cuda/profiler-users-guide/

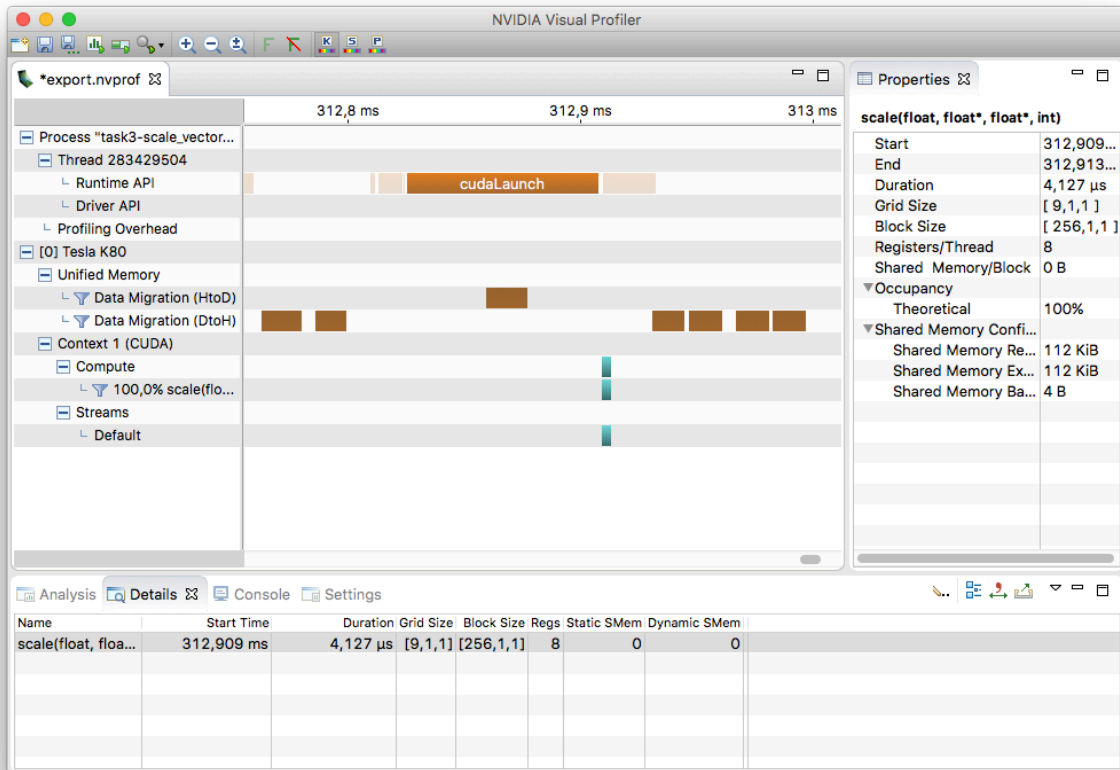
¹The CUDA Toolkit is freeware and can be installed on your local machine; even on a laptop without an NVIDIA GPU. This allows for downloading generated `nvprof` profiles and importing them locally or even for connecting to a remote server with NVVP.

²Since your application might be run via a batch system, the call to `nvprof` might need to be prefixed by an `srun`. Like in the screenshot.

1.2 GUI: NVIDIA Visual Profiler

While `nvprof` can be used to collect information or display it concisely on the command line, the Visual Profiler (NVVP) can be helpful to understand an application through a timeline view and by running performance analyses.

NVVP can be launched from the command line with `nvvp` or by installing the CUDA Toolkit on a local machine.



→ developer.nvidia.com/nvidia-visual-profiler