Profiling code on NVIDIA GPUs

CERN openlab Technical Workshop 2021

Ingvild Brevik Høgstøyl (NTNU-Trondheim, CERN)

Anne C. Elster (NTNU-Trondheim) and Maria Girone (CERN)

Goal and Overview

- Profiling CPU code
  - Using Valgrind, Callgrind and KCacheGrind

- Profiling GPU code using Nsight Systems
  - Identify overall system bottlenecks

- Profiling GPU code using Nsight Compute
  - Identify bottlenecks in kernels

Profiling CPU code

- Madgraph application

  https://github.com/madgraph5/madgraph4gpu
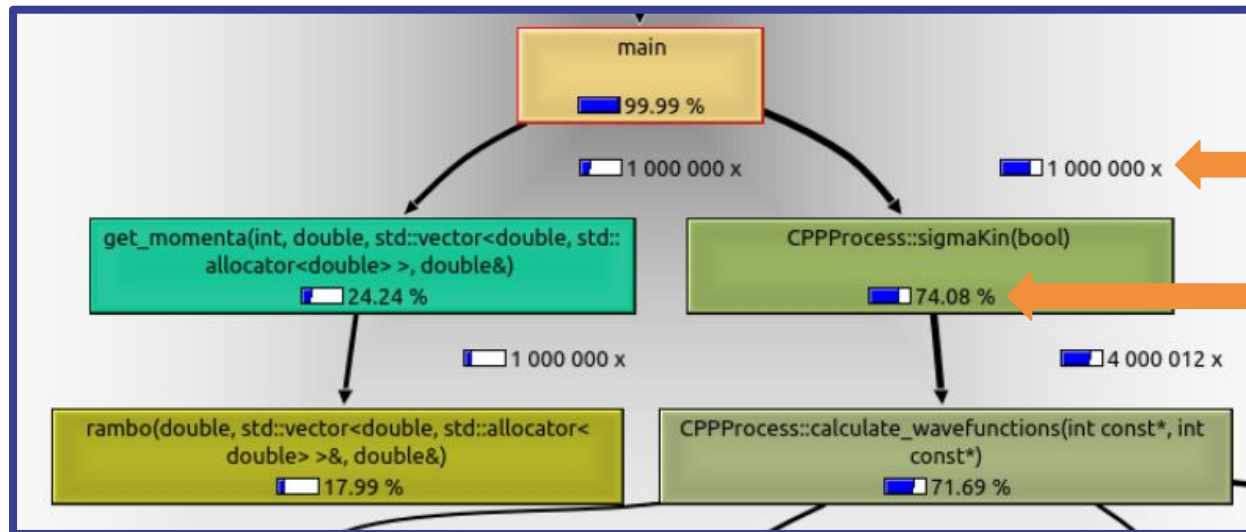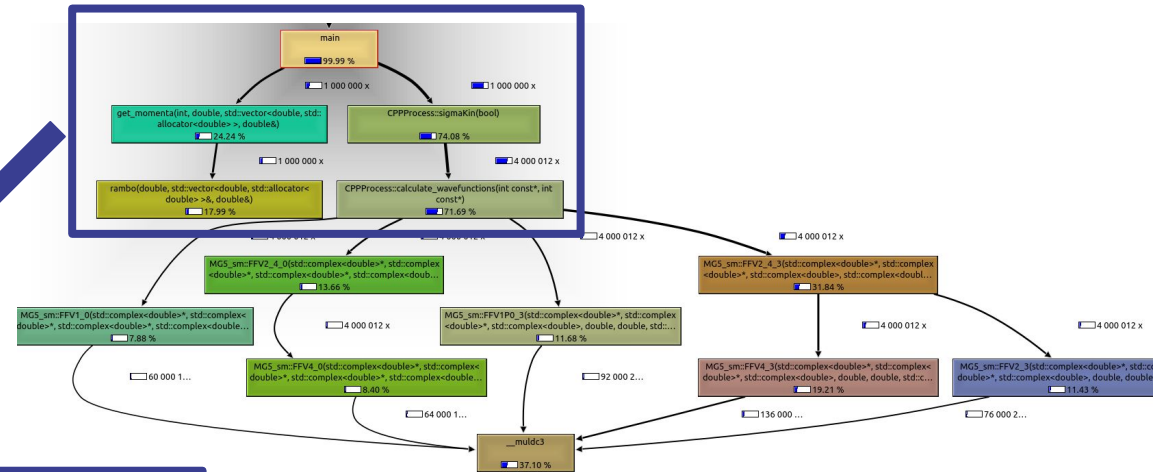
- Generate callgraph
  - Valgrind & Callgrind
- Hotspot visualization
  - KCacheGrind
  - Obtain overview

```
valgrind --tool=callgrind ./check.exe 1000000
```
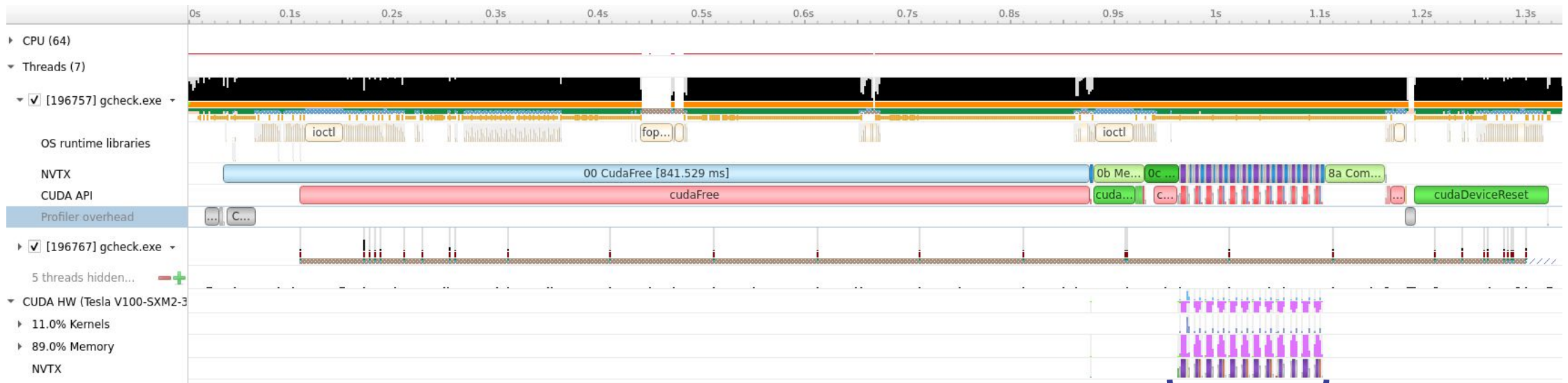
number of times function is called

percentage of time spent in this function and the functions it calls

3

Profiling GPU code: Nsight Systems

- Profiling:
  - Nsight Systems CLI
  - Overhead on profiling GPUs vs CPUs
- Visualization:
  - Nsight Systems GUI
  - Overall behaviour of the application

threads/block

blocks/grid   iterations

```
nsys profile ./gcheck.exe 16384 32 12
```


12 iterations

Profiling GPU code: Nsight Systems

- **Performance bottlenecks:**
  - Memory transfer
  - Kernel
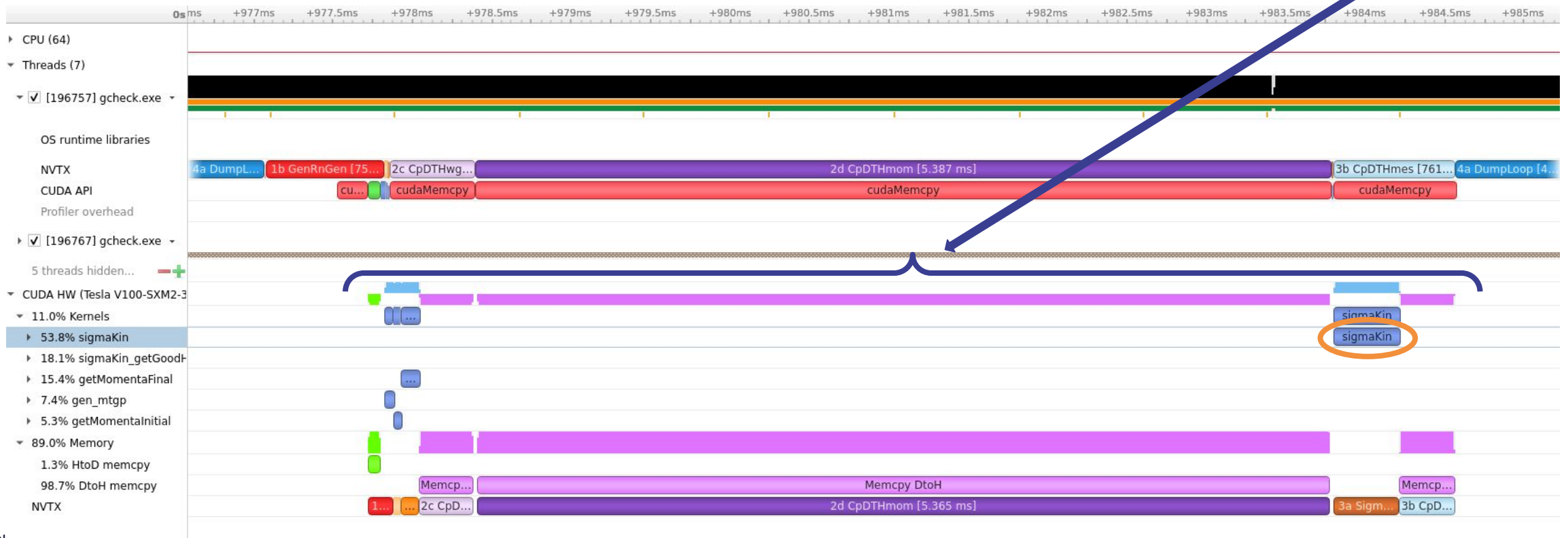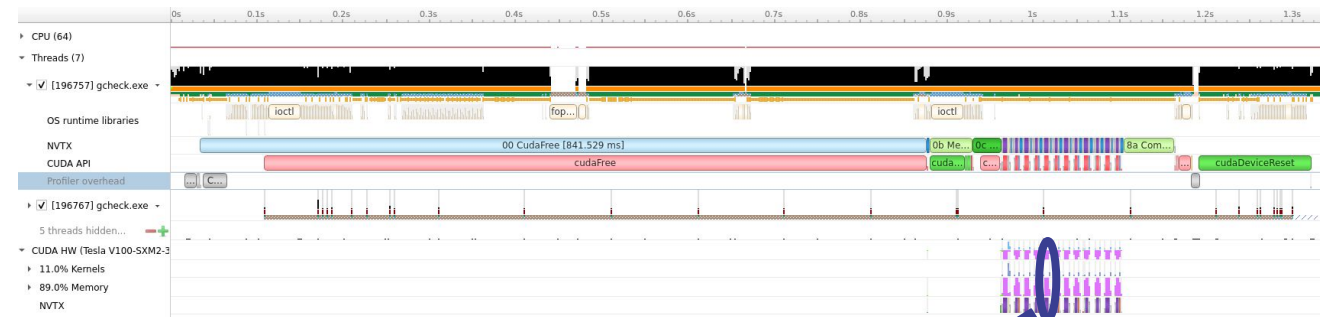
Profiling GPU code: Nsight Compute

- Profiling:
  - Nsight Compute CLI
  - Kernel profiling

- Visualization
  - Nsight Compute GUI
  - SigmaKin kernel

- Performance bottlenecks
  - SM vs Memory utilization
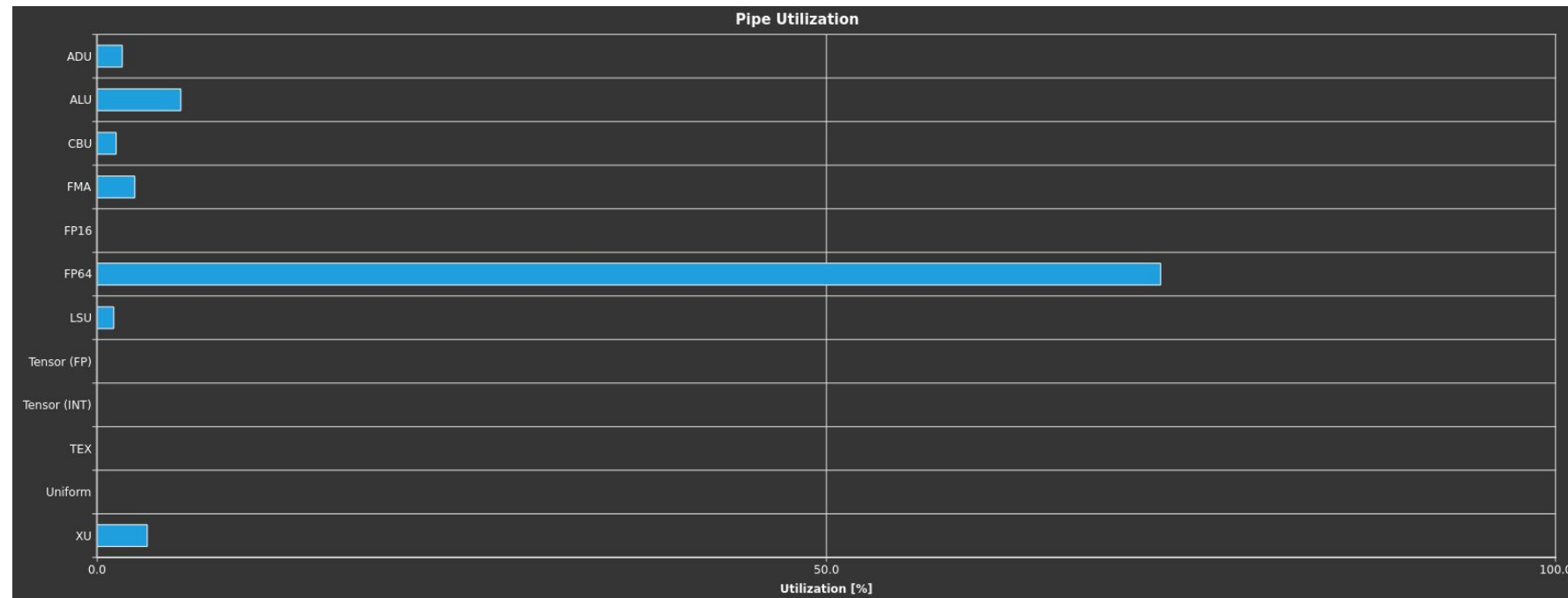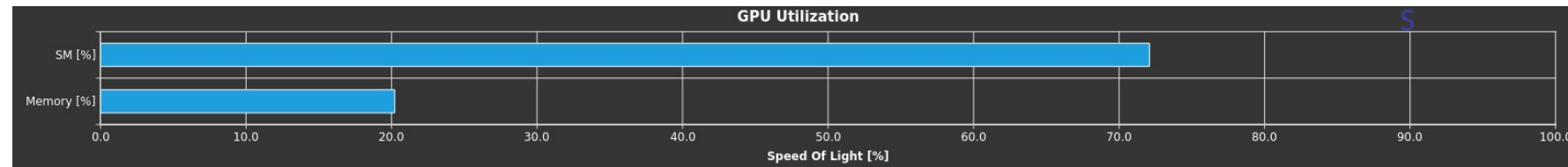 - Utilization of pipe

threads/block

```
nv-nsight-cu-cli -o results ./gcheck.exe 16384 32 1
```

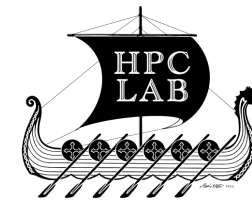create file with results

blocks/grid

iteration

Future work

- Ideas for GPU optimizations

  - Vary the number of threads/block

  - Change memory utilization

  - Reduce precision

- Use sampling information to identify hotspots

Thank you!

ingvild.brevik.hoegstoeyl@cern.ch