VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

INTRODUCTION TO HIGH PERFORMANCE COMPUTING

PERFORMANCE ANALYSIS BASICS

Radim Vavřík

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research, Development and Education

MINISTRY OF EDUCATION, YOUTH AND SPORTS

OUTLINE

Performance analysis and optimisation

- Motivation
- Hardware aspects
- Development process
- Best-practices

Performance tools and methodology

- Performance metrics
- CPU/GPU tools
- Live examples

POP CoE

Cray-1 supercomputer, source: wikipedia.org

TECHNICAL NOTES

- All presented tools/examples can be accessed and reproduced at IT4I clusters **anytime**
- Please, setup your preferred GUI access:
  1. **VNC** - server on a Karolina login node + client on laptop
     - How to? https://docs.it4i.cz/general/accessing-the-clusters/graphical-user-interface/vnc/
     - Recommended client https://www.realvnc.com/en/connect/download/viewer/
  2. **OOD** - Open OnDemand GUI via web browser, **IT4I VPN required**
     - How to? https://docs.it4i.cz/general/accessing-the-clusters/graphical-user-interface/ood/
     - Connection link https://ood-karolina.it4i.cz/
  3. **X11** - Log in via terminal with X-Window system enabled
     - How to? https://docs.it4i.cz/general/accessing-the-clusters/graphical-user-interface/x-window-system/
     - Usually worse UX for GUI apps due to network latency
- Most of the presented tools provide a **remote profiling**, e.g., generate output remotely from CLI while analysis can be done locally in GUI - not covered today

PERFORMANCE ANALYSIS

Who has any experience with a performance analysis tool?

- What was the tool?

Objectives today?

- Not to become an expert analyst
- Not to reach an incredible performance improvement of example codes

- Rather to get idea about the domain and introduce some tools

EFFICIENT USE OF HPC

What does it mean?

- To get the most performance out of your hardware
- The process is called **Performance Optimisation**

Why should I care about performance?

- Industry – achieve goals faster and **cheaper**
- Academia – do **more science**
  - The trend in grant competition (resource allocation) is to prove performance, scalability, etc.

KEY INGREDIENTS

Know your application

- What does it compute? (domain, methods, algorithms)
- How is it parallelized? (programming models)
- What final performance is expected? (HW limits)

Know your hardware

- What are the target machines and how many? (laptop, workstation, cluster)
- Machine-specific optimisations?

Know your tools

- Strengths and weaknesses of each tool? (easy-to-use vs detailed information)
- Learn how to use them (examples with problems/patterns)

Know your process

- Constant learning

HARDWARE ASPECTS OF PERFORMANCE

Filesystem

- I/O operations

Network

- internode communication

Memory subsystem

- NUMA effect

CPU cores

- thread/process affinity, pinning, caches

Vector registers

- vectorization, vector instructions

Accelerators

- GPU/MIC utilization, host-device data transfers

GET READY

Connect to login node via GUI

- VNC / OOD / X11

Submit an interactive job

```
salloc --account=DD-23-116 --reservation=dd-23-116_2024-06-05T09:00:00_2024-06-05T12:30:00_5_qgpu --gpus 1
```

BASIC TOOLS

Useful to get familiar with the machine

| `lscpu`

| `cat /proc/cpuinfo`

- processor : 71 -> 72 logical processors per node
- cpu cores : 18 -> 18 physical cores per socket
- siblings : 36 -> 36 logical processors per socket
- -> 2 hyperthreads per core
- -> 2 sockets per node

| `cat /proc/meminfo`

- MemTotal: 196510848 kB -> 187 GB

| `ml impi`

| `cpuinfo` # Intel MPI utility

BASIC TOOLS

Use HTOP tool for interactive jobs

| htop -d 5 # delay 0.5s

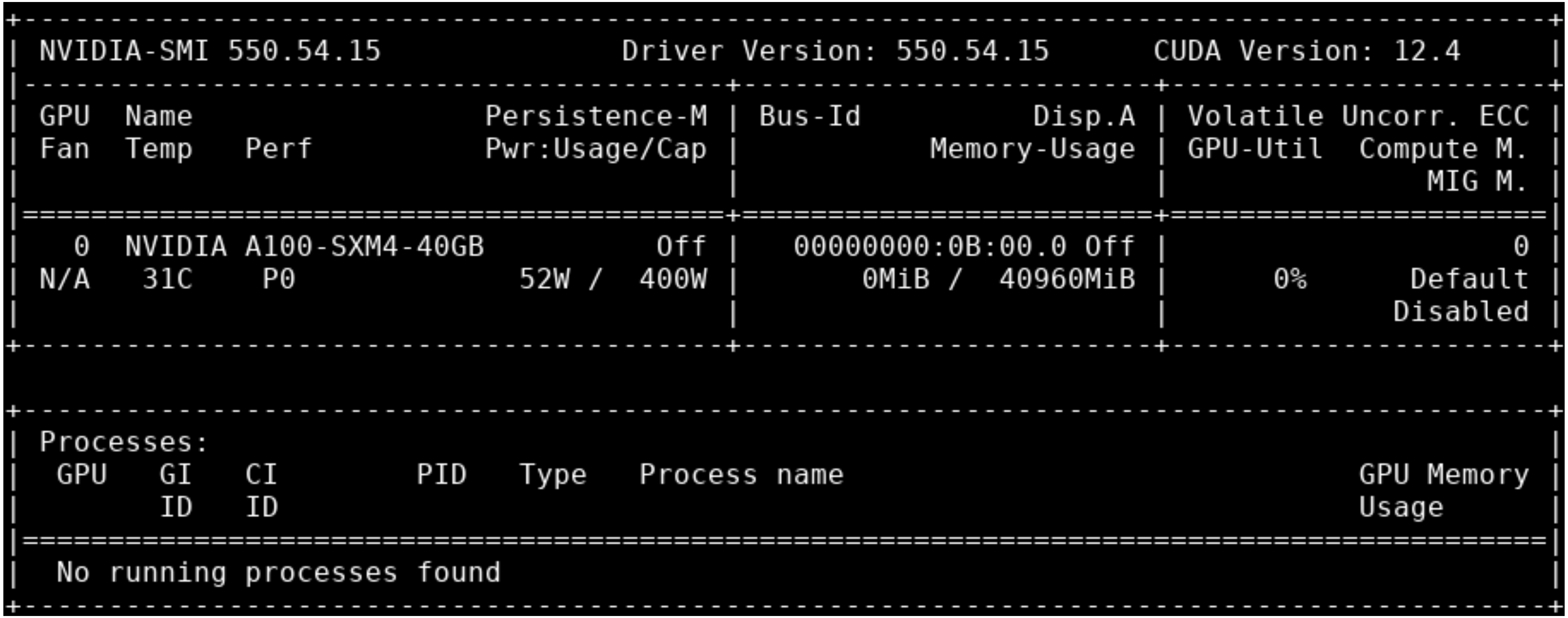
- Configurable (e.g. core id, threads, process tree)

```
  1  [||||||||||||100.0%]   10 [||||||||||||100.0%]   19 [||||||||||||100.0%]   28 [||||||||||||100.0%]
  2  [||||||||||||100.0%]   11 [||||||||||||100.0%]   20 [||||||||||||100.0%]   29 [||||||||||||100.0%]
  3  [||||||||||||100.0%]   12 [||||||||||||100.0%]   21 [||||||||||||100.0%]   30 [||||||||||||100.0%]
  4  [||||||||||||100.0%]   13 [||||||||||||100.0%]   22 [||||||||||||100.0%]   31 [||||||||||||100.0%]
  5  [||||||||||||100.0%]   14 [||||||||||||100.0%]   23 [||||||||||||100.0%]   32 [||||||||||||100.0%]
  6  [||||||||||||100.0%]   15 [||||||||||||100.0%]   24 [||||||||||||100.0%]   33 [||||||||||||100.0%]
  7  [||||||||||||100.0%]   16 [||||||||||||100.0%]   25 [||||||||||||100.0%]   34 [||||||||||||100.0%]
  8  [||||||||||||100.0%]   17 [||||||||||||100.0%]   26 [||||||||||||100.0%]   35 [||||||||||||100.0%]
  9  [||||||||||||100.0%]   18 [||||||||||||100.0%]   27 [||||||||||||100.0%]   36 [||||||||||||100.0%]
  Mem[|||||                        13.8G/187G]   Tasks: 79, 346 thr; 36 running
  Swp[                                 0K/0K]   Load average: 23.62 6.93 3.32
                                                 Uptime: 15 days, 12:06:32

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
11171 vav0038    35  15  296M 90236  7232 R 99.5  0.0  1:02.72 ../examples/wave_c 100
11203 vav0038    35  15  298M 90240  7244 R 99.5  0.0  1:03.07 ../examples/wave_c 100
11212 vav0038    35  15  322M 92280  7324 R 99.5  0.0  1:03.04 ../examples/wave_c 100
11162 vav0038    35  15  300M 90220  7272 R 99.5  0.0  1:03.10 ../examples/wave_c 100
11188 vav0038    35  15  323M 90236  7328 R 99.5  0.0  1:03.05 ../examples/wave_c 100
11207 vav0038    35  15  311M 92272  7296 R 99.5  0.0  1:03.04 ../examples/wave_c 100
11164 vav0038    35  15  326M 90232  7340 R 99.5  0.0  1:03.09 ../examples/wave_c 100
11195 vav0038    35  15  298M 90232  7232 R 99.5  0.0  1:03.09 ../examples/wave_c 100
11158 vav0038    35  15  319M 92284  7304 R 99.5  0.0  1:03.07 ../examples/wave_c 100
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

BASIC TOOLS

Similar tool for NVIDIA GPUs

| watch -n 1 nvidia-smi



```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA A100-SXM4-40GB          Off |   00000000:0B:00.0 Off |                    0 |
| N/A   31C    P0             52W /  400W |       0MiB /  40960MiB |      0%      Default |
|                                         |                        |             Disabled |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

PERFORMANCE-AWARE DEVELOPMENT PROCESS

1. Develop correct functionality (testing helps)
2. Identify bottlenecks (performance limiters) using performance tools
3. Optimise bottlenecks until satisfied
    1. Build a hypothesis (ask a question)
    2. Explain the behavior (answer the question)
    3. Change the code (double-check correct functionality)
    4. Verify optimisations using profiling tools
4. Repeat until job done

BEST PRACTICES

- Do not optimise your code prematurely!
- Focus on main computational time-consuming phases (hotspots), omit preprocessing/postprocessing phases if applicable
- The 80/20 rule:
  - Programs typically spend 80% of their time in 20% of the code
  - Programmers typically spend 20% of their effort to get 80% of the total speedup possible for the application
- Keep track of your optimisation progress over time
- Always use compute nodes for profiling (**not login nodes - shared**)
- **Use SW libraries!**

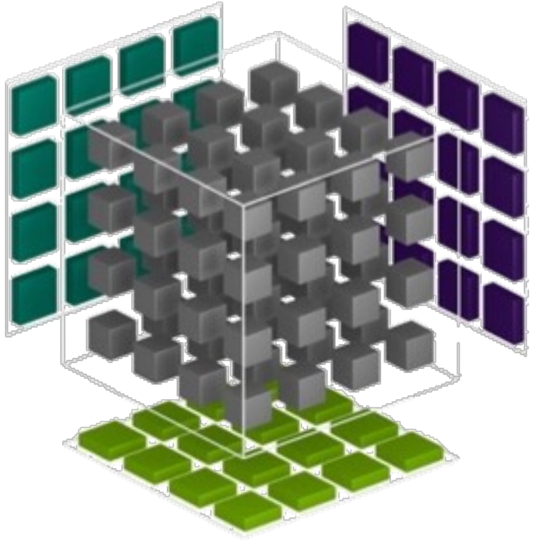
SOFTWARE LIBRARIES

General-purpose math libraries

- BLAS (MKL, OpenBLAS, ATLAS, cuBLAS, ...)
- LAPACK (MKL, OpenBLAS, ATLAS, cuSolver, ...)
- FFT (MKL, cuFFT, ...)
- ...

Domain-specific libraries

- Chemistry, Bio, Geo, Physics, CAE, Big data, ML/DL



HW-specific libraries

- GPU/MIC, Intel/AMD/IBM

Optimized implementation

- Usually much better performance than a custom code
- Do NOT reinvent a wheel!
- (But avoid overkill)

PERFORMANCE METRICS

Execution time (time, time.h, ...)

- real 0m10.245s (elapsed real time)
- user 0m19.890s (user CPU time using OMP_NUM_THREADS=2)
- sys 0m0.285s (system CPU time)

Processor speed (flop/s) and Memory throughput (GB/s)

- Calculated operations per time (e.g. c = a + b + c -> 2 operations)
- Transferred bytes per time (e.g. c = a + b + c -> 3 RD + 1 WR * 8 bytes)

Speedup and Efficiency

- $S_P = T_1 / T_P$
- $E_P = S_P / P$

Scalability

- Strong/weak scaling

Others: portability, programming ability, etc.

PEAK PERFORMANCE EXAMPLE

- The theoretical HW limits, e.g. AMD EPYC 7H12 (Rome)

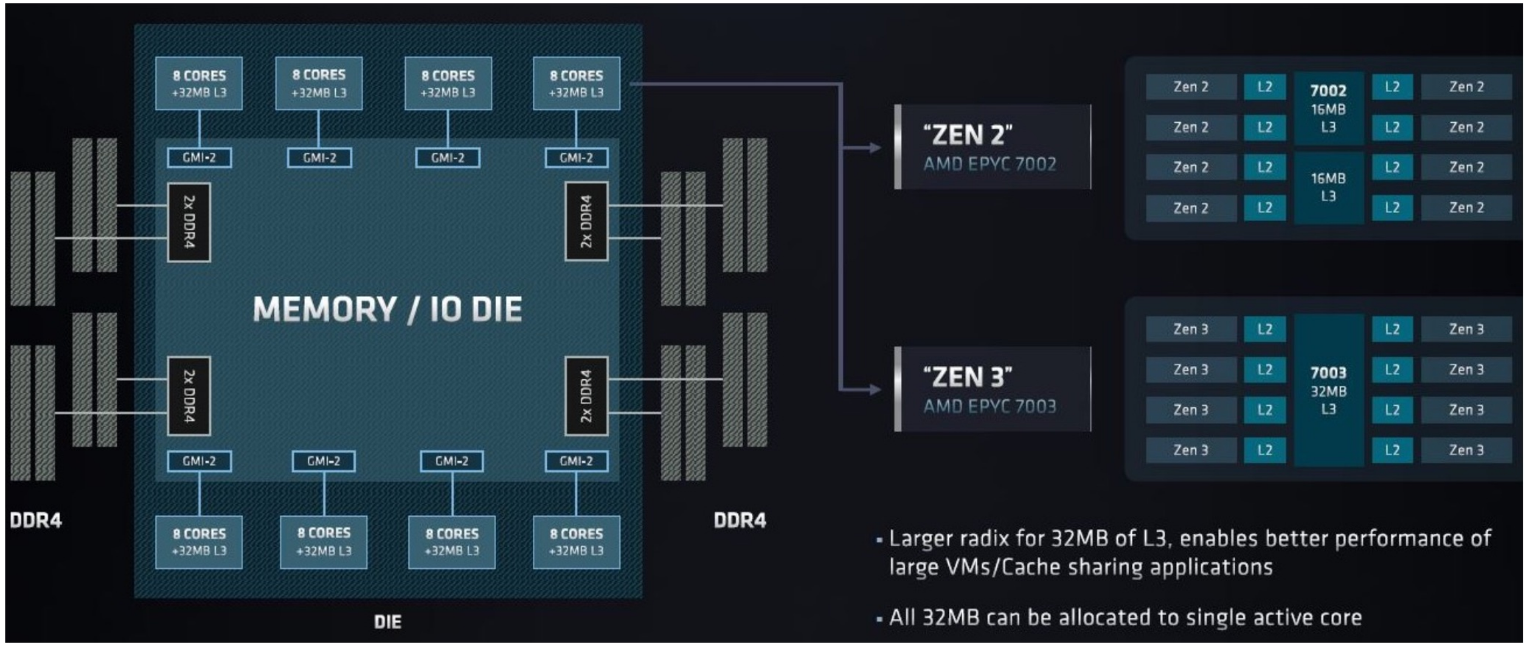
Processor speed:

Number of compute nodes (Karolina-size machine)	720
Number of sockets (CPUs) per node	2
Frequency	2.6 GHz
Number of cores per socket	64
FMA instructions (**a** * **b** + **c**)	2
FMA units per core	2
SIMD (AVX2 256b) = 4x double precision	4
	3 833 856 Gflop/s
	3.8 Pflop/s
	(2.6 Tflop/s per socket)

PEAK PERFORMANCE EXAMPLE

Memory bandwidth:

▪ Number of compute nodes (Karolina-size machine)	720
▪ Number of sockets (CPUs) per node	2
▪ # channels per socket	8
▪ DDR4 bus width	8 B
▪ Frequency	3200 MT/s



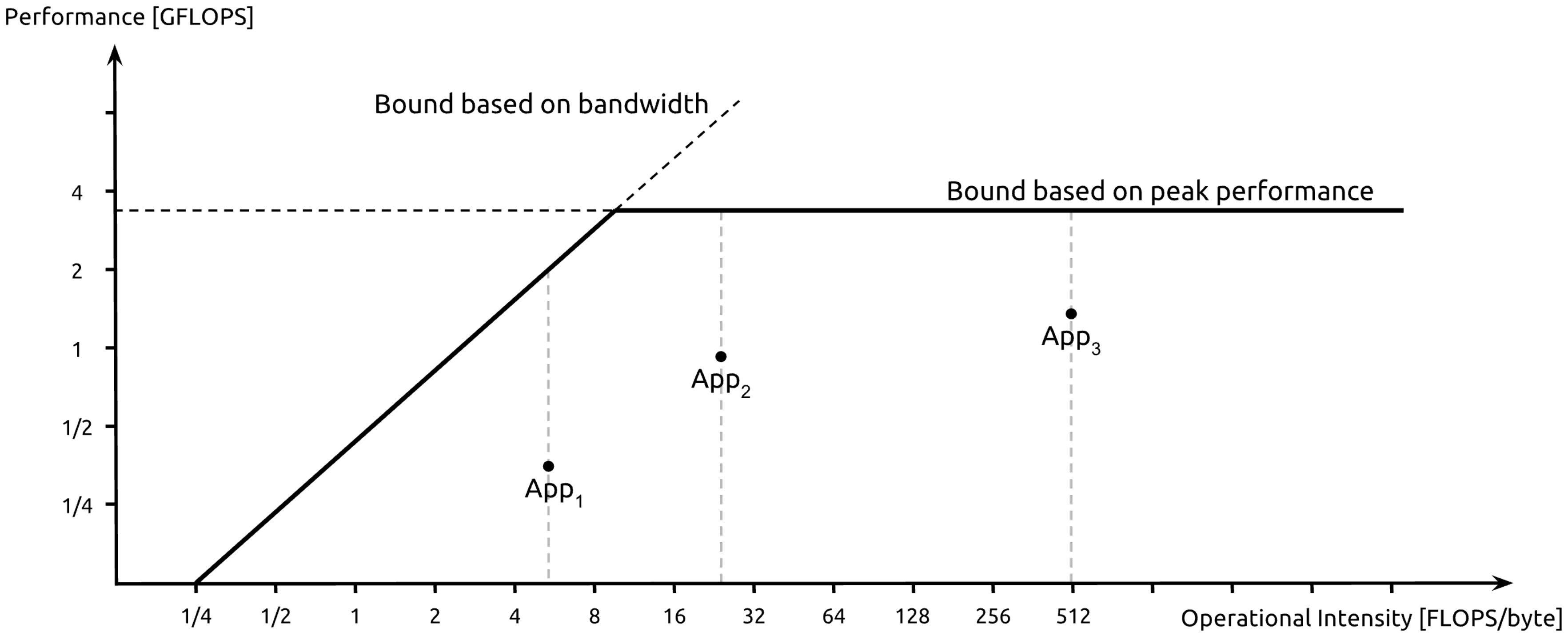
294 912 000 MB/s

294 TB/s

(204 GB/s per socket)

ROOFLINE MODEL

- Shows the performance of an algorithm (application) with respect to the HW limits of the architecture
- Identify if an algorithm is **compute bound** or **memory bound**
- Based on **Operational intensity** - a ratio of FLOPS (arithmetic operations) performed with required amount of data (operands)

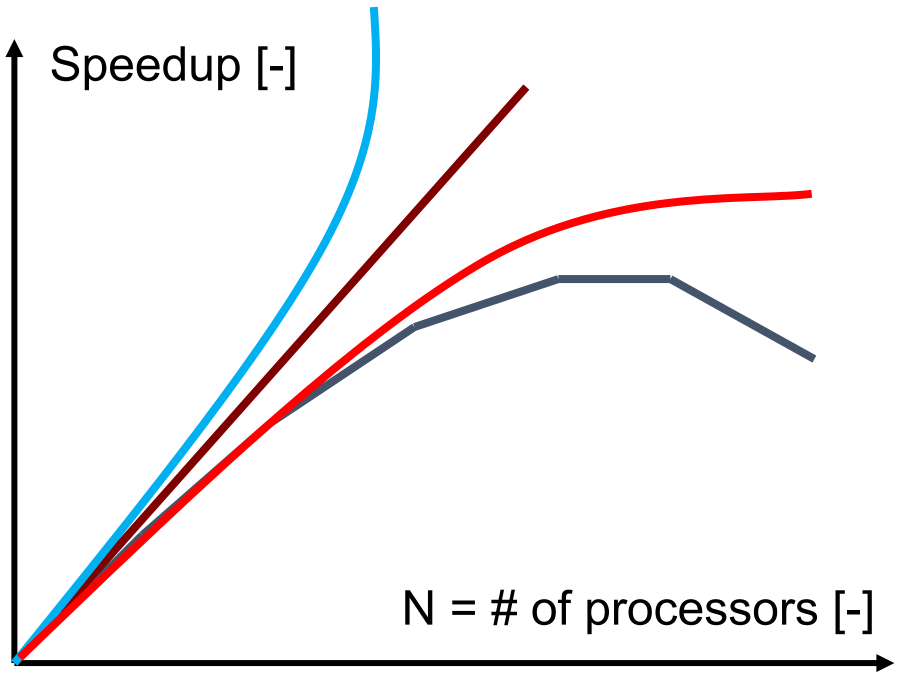


SPEEDUP

- **Speedup** – a ratio of a serial execution time to a parallel execution time

speedup on N processors - $S_N = \frac{T_1}{T_N}$
- execution time on 1 processor (T_1)
- execution time on N processors (T_N)

- Linear speedup $S_N = N$
- Sub-linear speedup $S_N < N$
  - Communication
  - Load imbalance
  - Decomposition overhead
- Super-linear speedup $S_N > N$
  - Cache
  - Algorithm



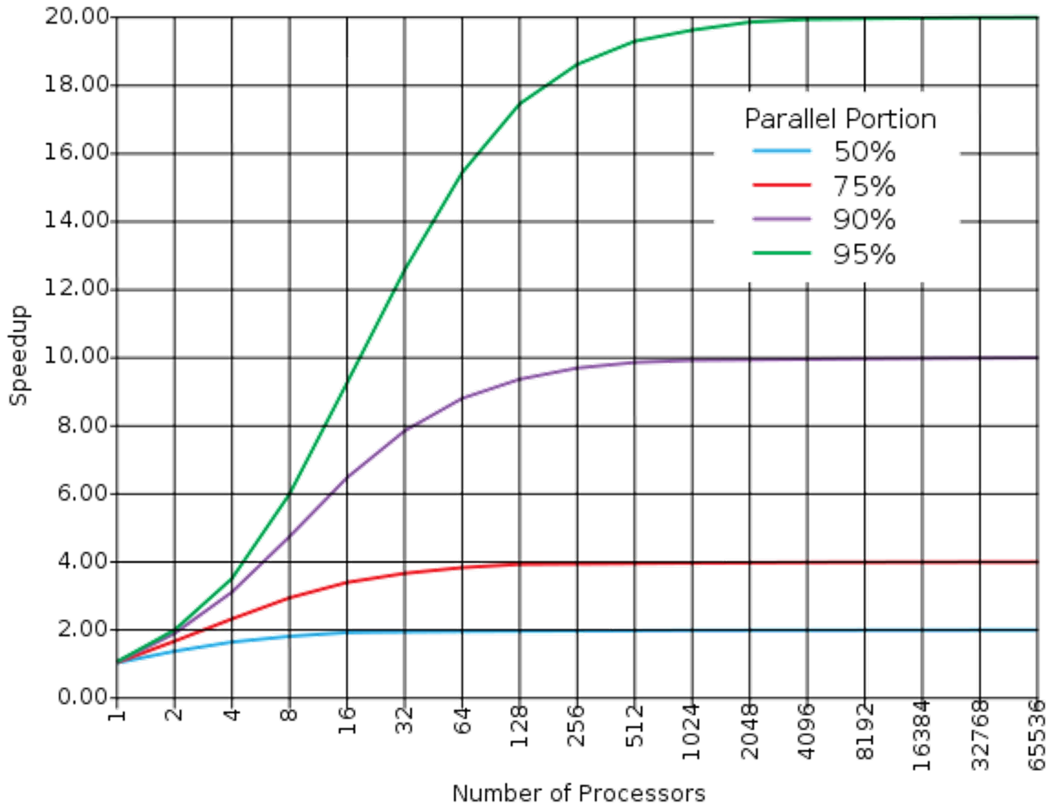
SCALABILITY

- **Scalability** - the ability to maintain performance gain when system and problem size increase

- **Amdahl's law** – maximum achievable speedup is limited by the serial portion of the code

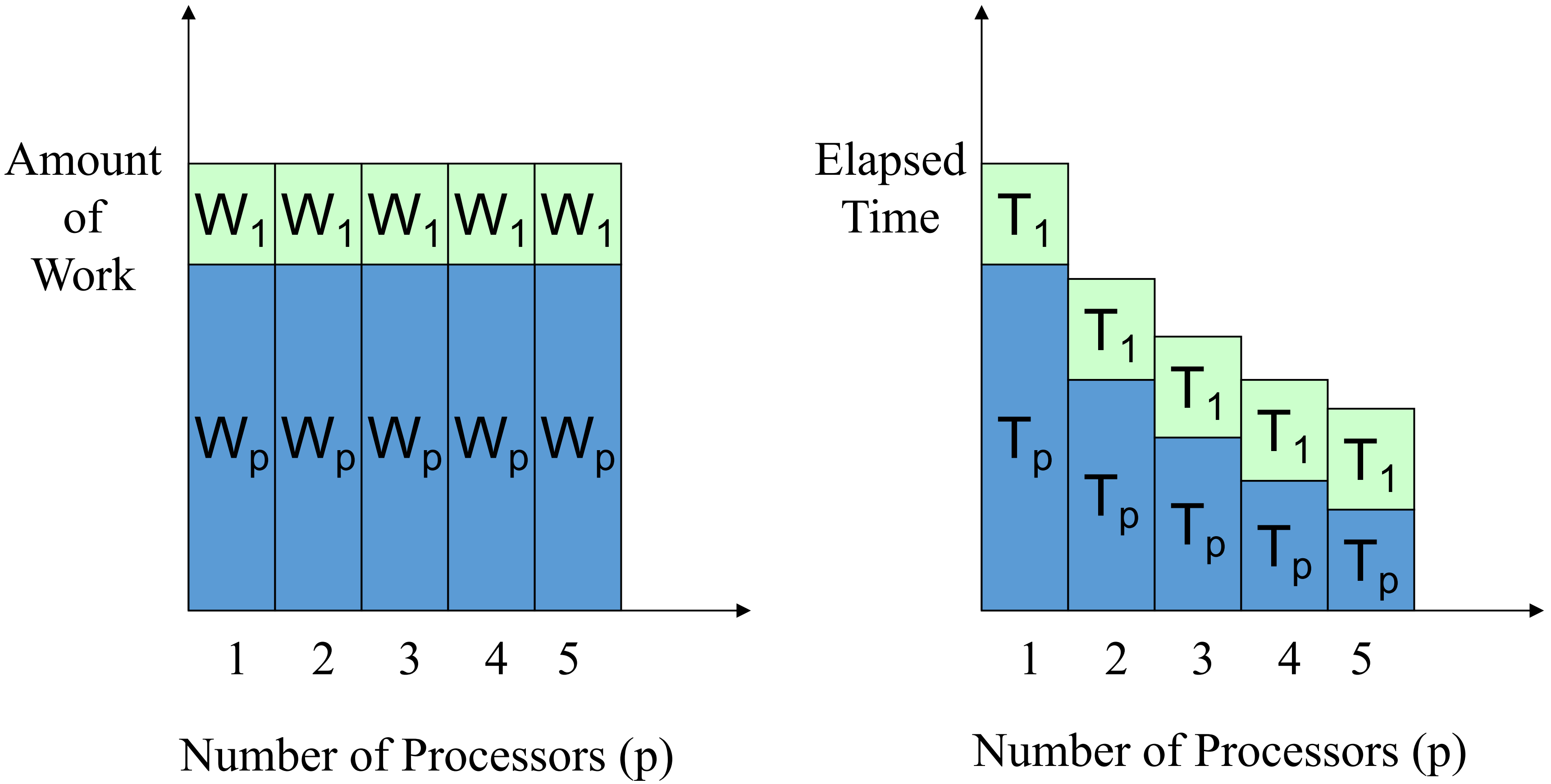
X% (parallel)	Y% (ser.)

$$S_{MAX} = \frac{1}{\frac{Y}{100}}$$



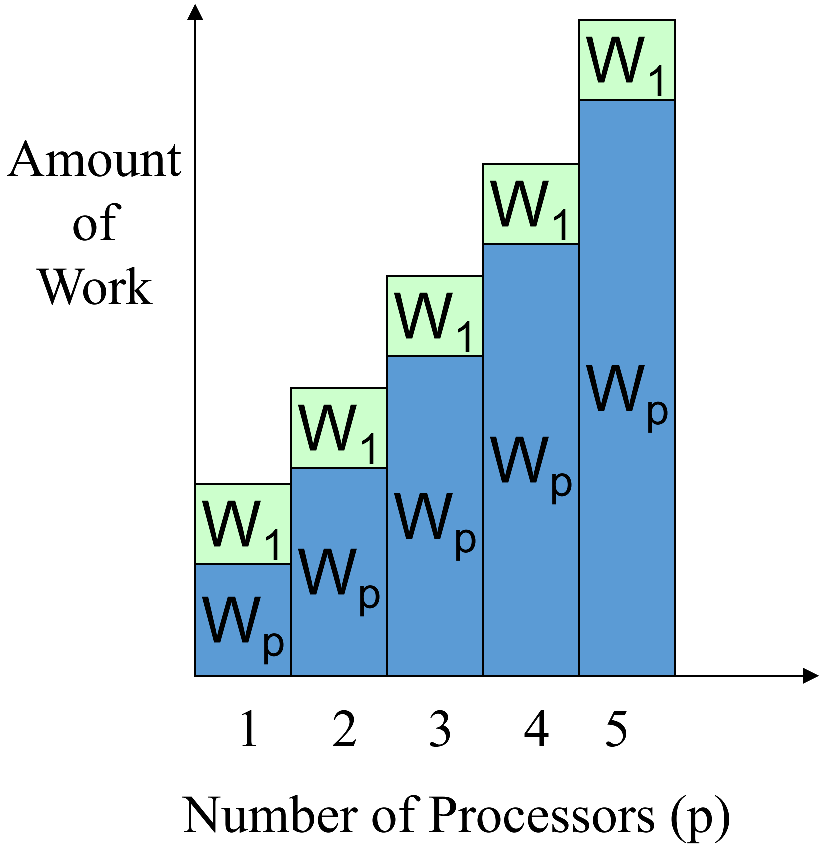
SCALABILITY

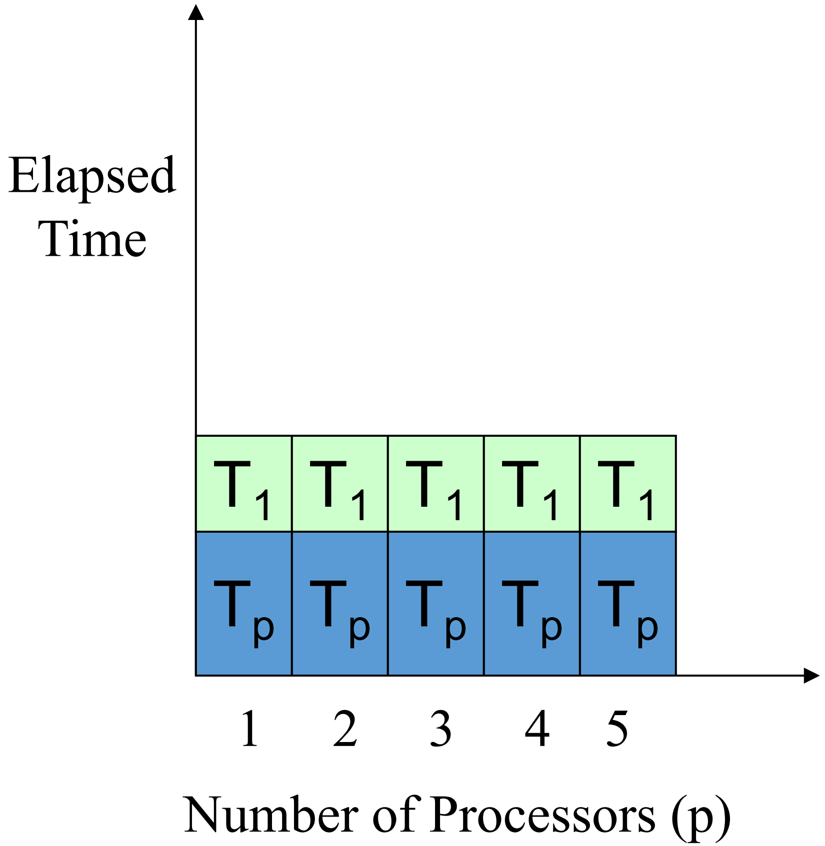
- **Strong scaling** - how the processing time varies with the number of processors for **a fixed total problem size**



SCALABILITY

- **Weak scaling** - how the processing time varies with the number of processors for **a fixed problem size per processing unit**





CLASSIFICATION OF PERFORMANCE TOOLS

- There are many tools that can be classified by the implemented approach

Data collecting mechanism

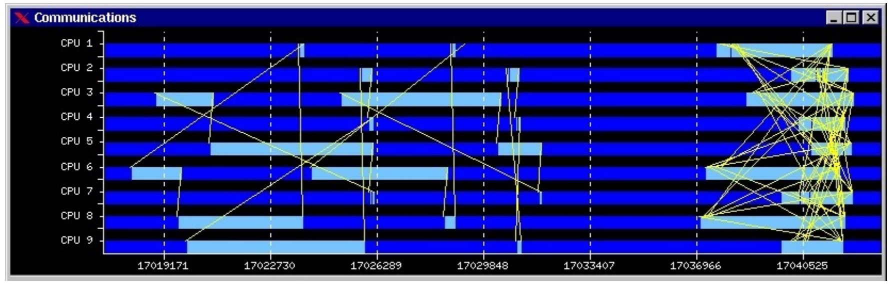
- **Sampling** - automatically collect data per time unit
- **Instrumentation** - manually/automatically add instructions to the source code to collect data - intrusive

Form of data presentation

- **Reports** - general overview of the whole application
- **Profiling** - accumulated characteristics of metrics
- **Tracing** - details about selected events - intrusive

Analysis of the collected data

- **Online** - during the execution - rare
- **Post mortem** - after the execution



Example of a trace, source: tools.bsc.es

Modeling - simulate state, ideal network, HW failure, etc.

PERFORMANCE TOOLS - CPU

- Single-node/parallel, architecture, language, programming model, focus (instrumentation, correctness checking, etc.)

Proprietary tools – licenses usually available on clusters

- Linaro (~~ARM~~ (~~Allinea~~)) Performance Report
- Linaro (~~ARM~~ (~~Allinea~~)) MAP
- Intel Application Performance Snapshot
- Intel Vtune
- AMD μProf
- Vampir

Open-source tools (VI-HPS)

- BSC tools (Extrae/Paraver)
- JSC tools (Score-P/Scalasca/Cube)
- MAQAO
- https://www.vi-hps.org/tools/tools.html (guide)

GPU PROFILING – TOOLS

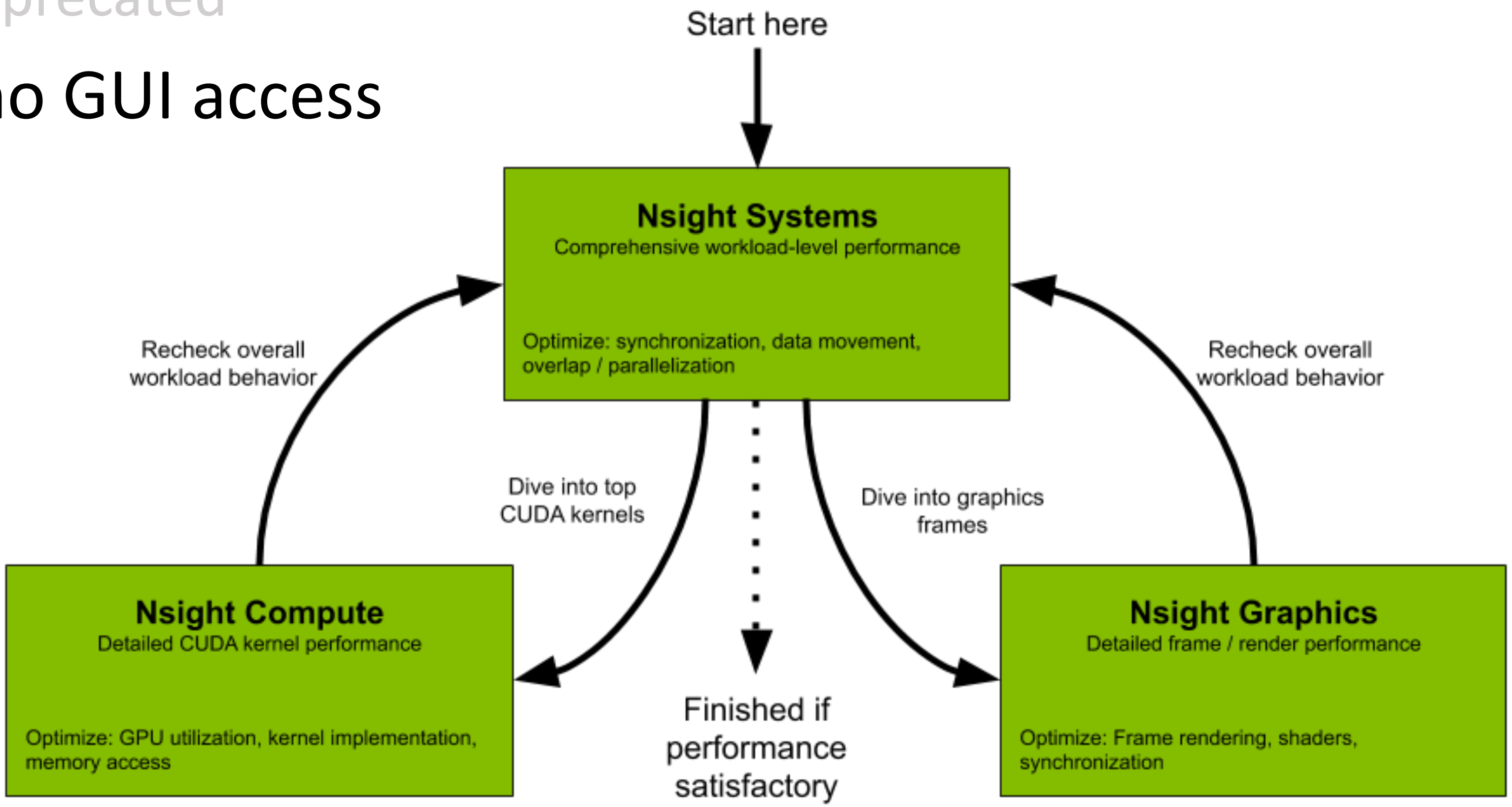
GUI tools

- NVIDIA Nsight Systems – **system-level** profiling
- NVIDIA Nsight Compute – **CUDA kernel-level** profiling
- NVIDIA Visual Profiler - deprecated

Command-line tools - for no GUI access

(e.g. in batch jobs)

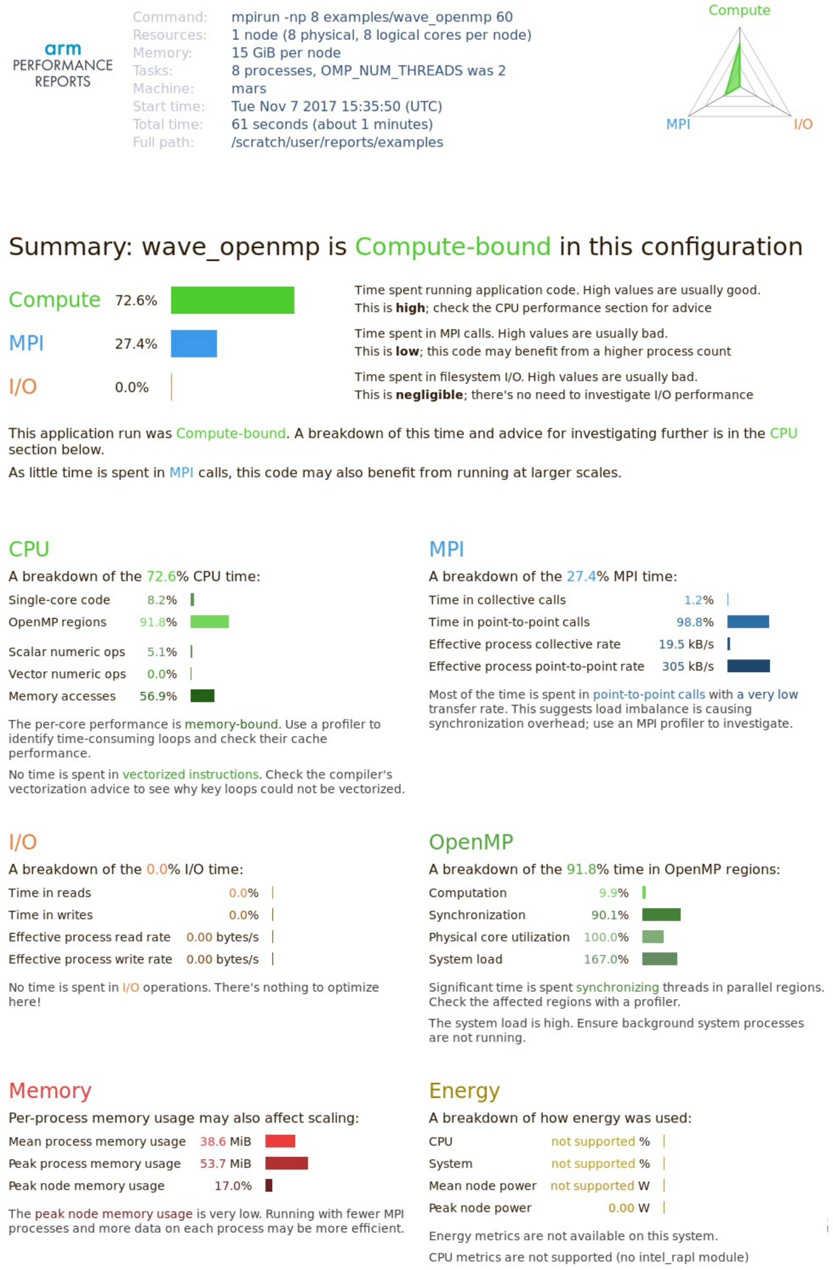
- NVIDIA nsys
- NVIDIA ncu
- AMD ROC-profiler
  - analogous to nsys
  - Chrome for visualization
- NVIDIA nvprof
  - deprecated



Nsight tools, source: nvidia.com

ARM PERFORMANCE REPORTS

- Global high-level overview of the application
- No source code or recompilation required
- Run: **perf-report** srun -n <#procs> <app>
- Auto-generated text and HTML output
- Report summary (Compute, MPI, Input/Output)
- CPU, MPI, I/O, OpenMP, Memory, Energy, Accelerator breakdown sections
- Advanced configuration through command line flags possible



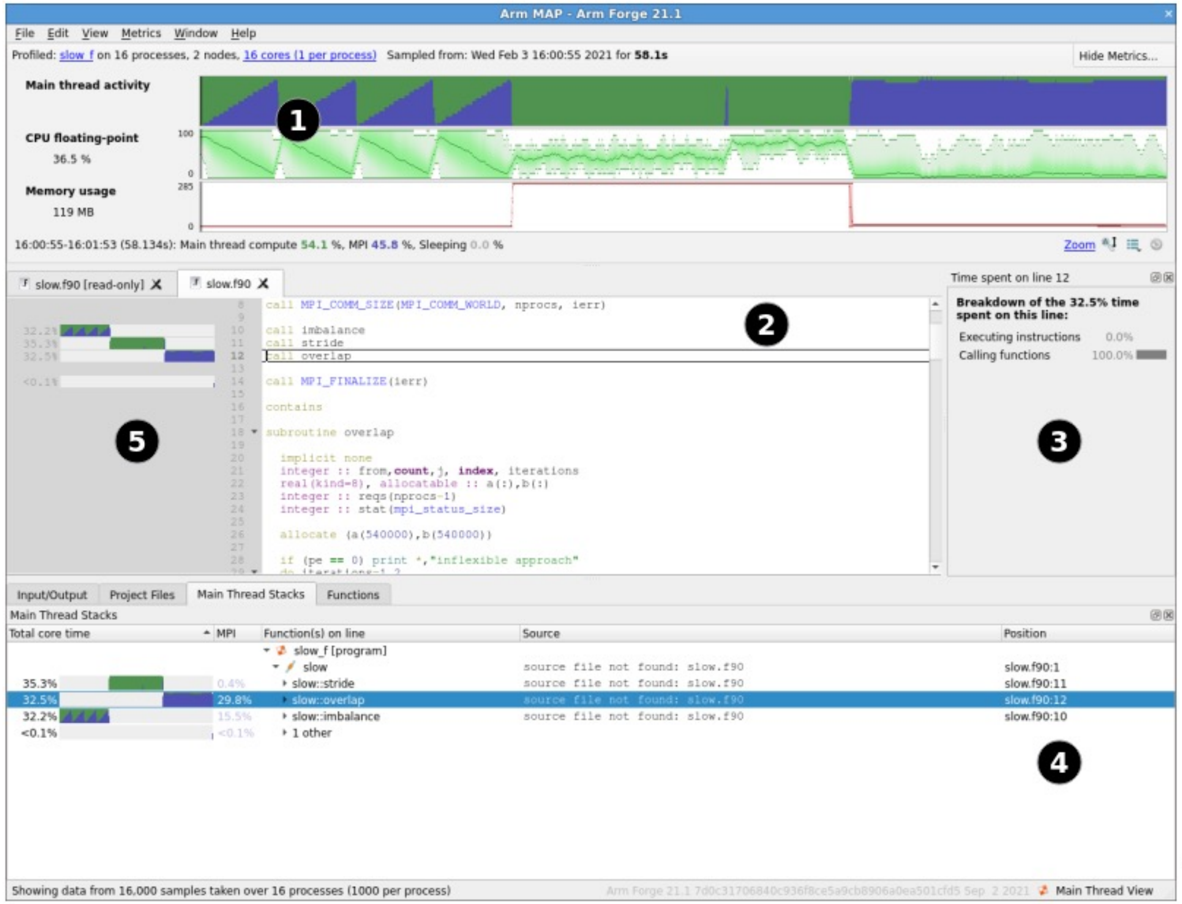
ARM PERFORMANCE REPORTS - EXAMPLE

```
ml Forge/23.1.2 OpenMPI/4.1.6-GCC-12.2.0-CUDA-12.4.0
ml show Forge
cp -r /apps/all/Forge/23.1.2/examples ~/forge_examples
cd ~/forge_examples
make

srun -n 16 ./wave_c 10

mkdir perf_reports && cd perf_reports
perf-report srun -n 16 ../wave_c 10
firefox wave_c_16p_1n_YYYY-MM-DD_hh-mm.html &          # on login node
OMP_NUM_THREADS=8 perf-report srun -n 2 -c 8 ../wave_openmp 10
firefox wave_openmp_2p_1n_8t_YYYY-MM-DD_hh-mm.html &
```

ARM MAP

- Low overhead sampling profiler for localisation of bottlenecks
- No recompilation required, only debugging symbols are useful (-g)

1. Metrics view (CPU, MPI, I/O, memory, vectorization)
2. Source code viewer
3. Selected lines view
4. Output, files, callpaths
5. Sparkline charts

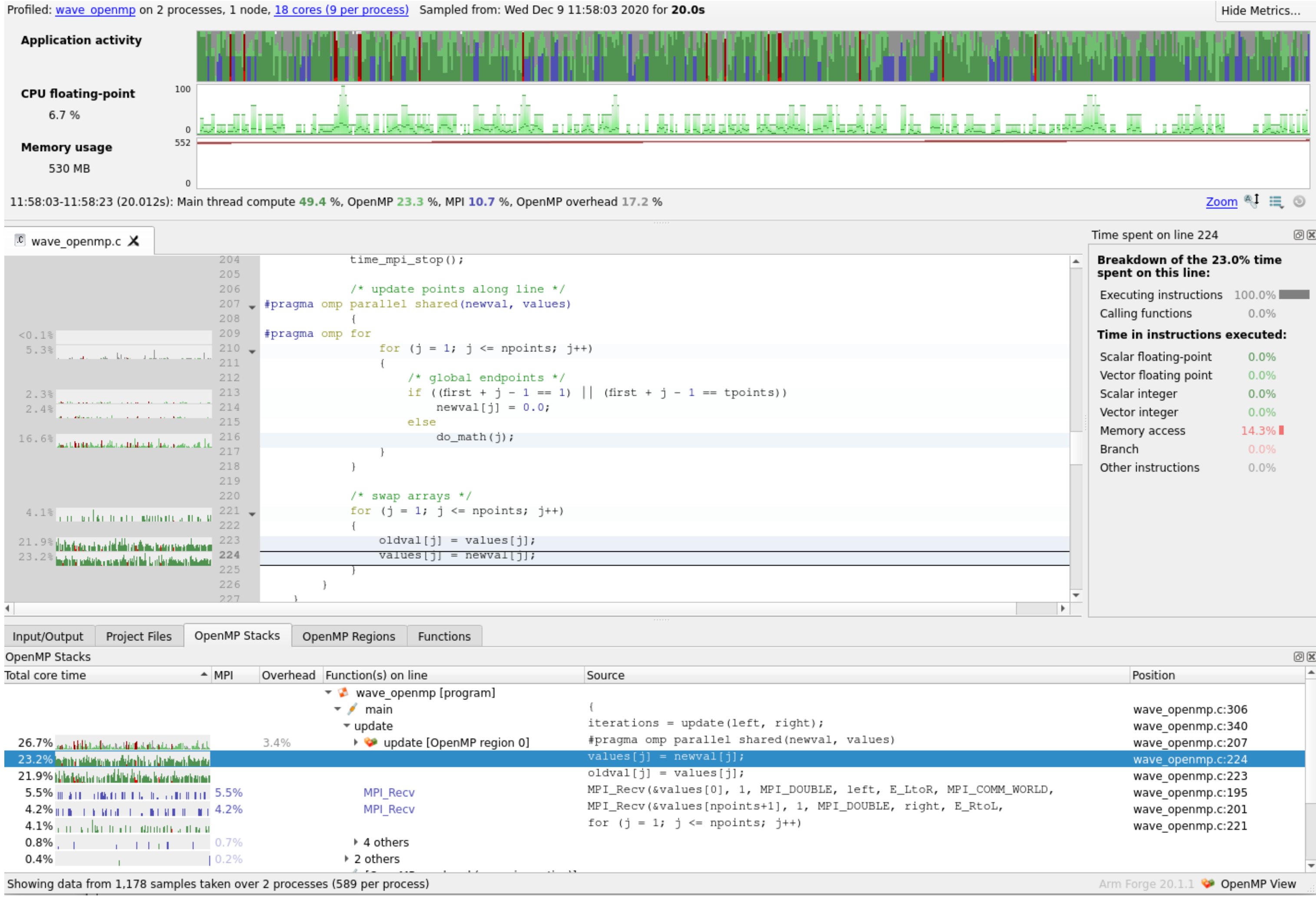



```
map
map srun -n <#procs> <app> [args]
map --profile srun -n <#procs> …
map <profile.map>
perf-report <profile.map>
```

ARM MAP

- All charts are timelines
  - Horizontal axis time
- Vertical axis are processes
- Useful code is green
- MPI is blue
- breakout recalculated

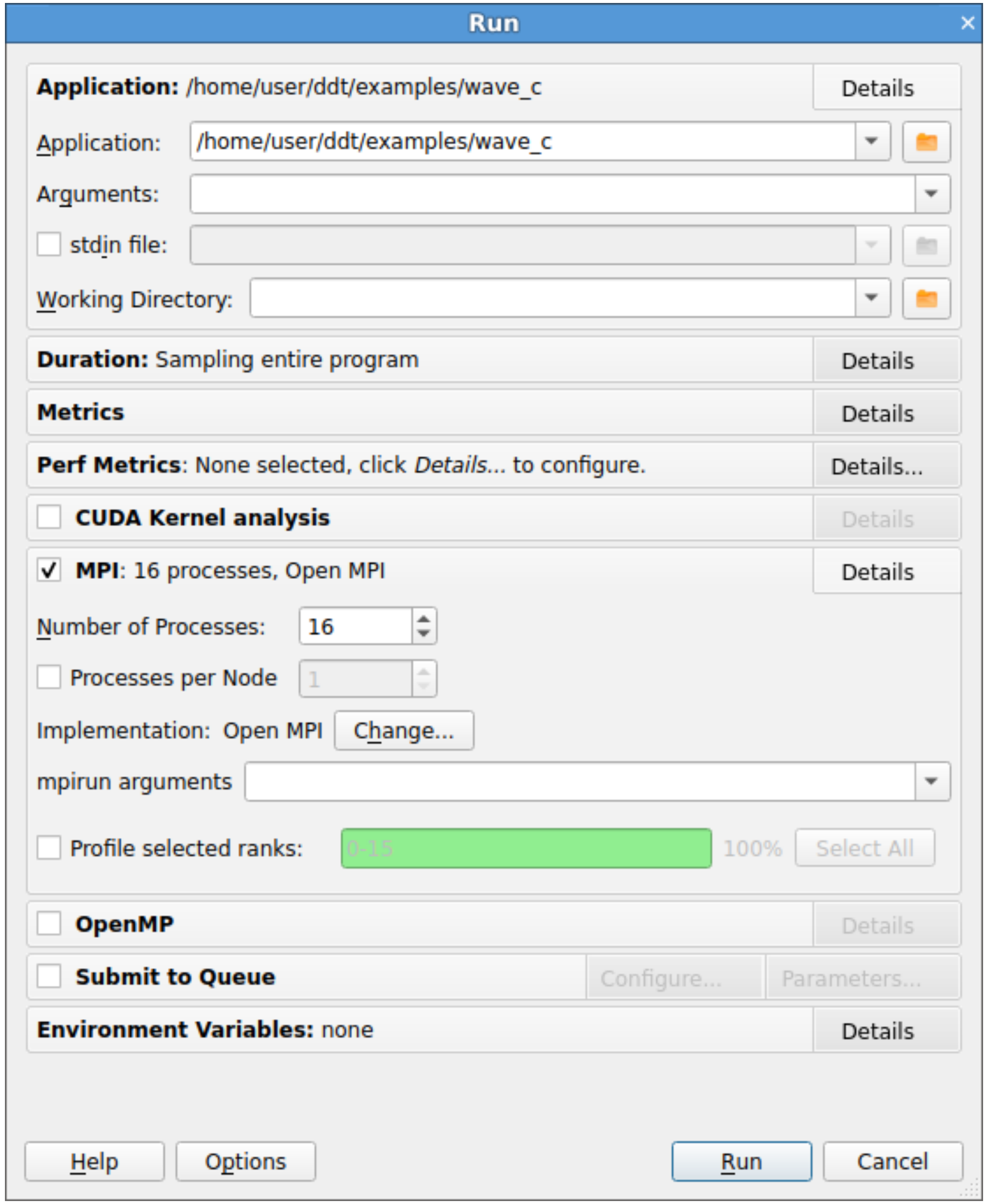
  when zooming
- Multiple presets available
  - CPU
  - MPI
  - I/O
  - memory
  - …



ARM MAP - EXAMPLE

- `ml Forge/23.1.2 OpenMPI/4.1.6-GCC-12.2.0-CUDA-12.4.0`
- `mkdir ~/forge_examples/map && cd ~/forge_examples/map`
- `OMP_NUM_THREADS=8 map srun -n 2 -c 8 ../wave_openmp 10`

- Optionally limit duration
- Optionally adapt metrics
- Click Run
- Use the User guide!

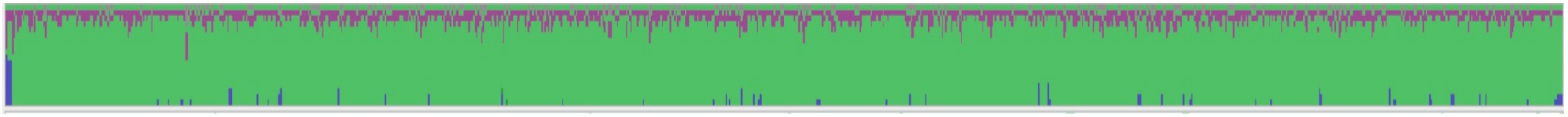


ARM MAP - EXAMPLE

- A large section of blue means all the processes in MPI calls - try to reduce these. Triangular shape indicates load imbalance.

- A large section of dark green means all the processes in single-threaded computations - try to avoid.

- A large sections of light green - OpenMP regions being effectively used across all processes simultaneously.



NVIDIA NSIGHT SYSTEMS

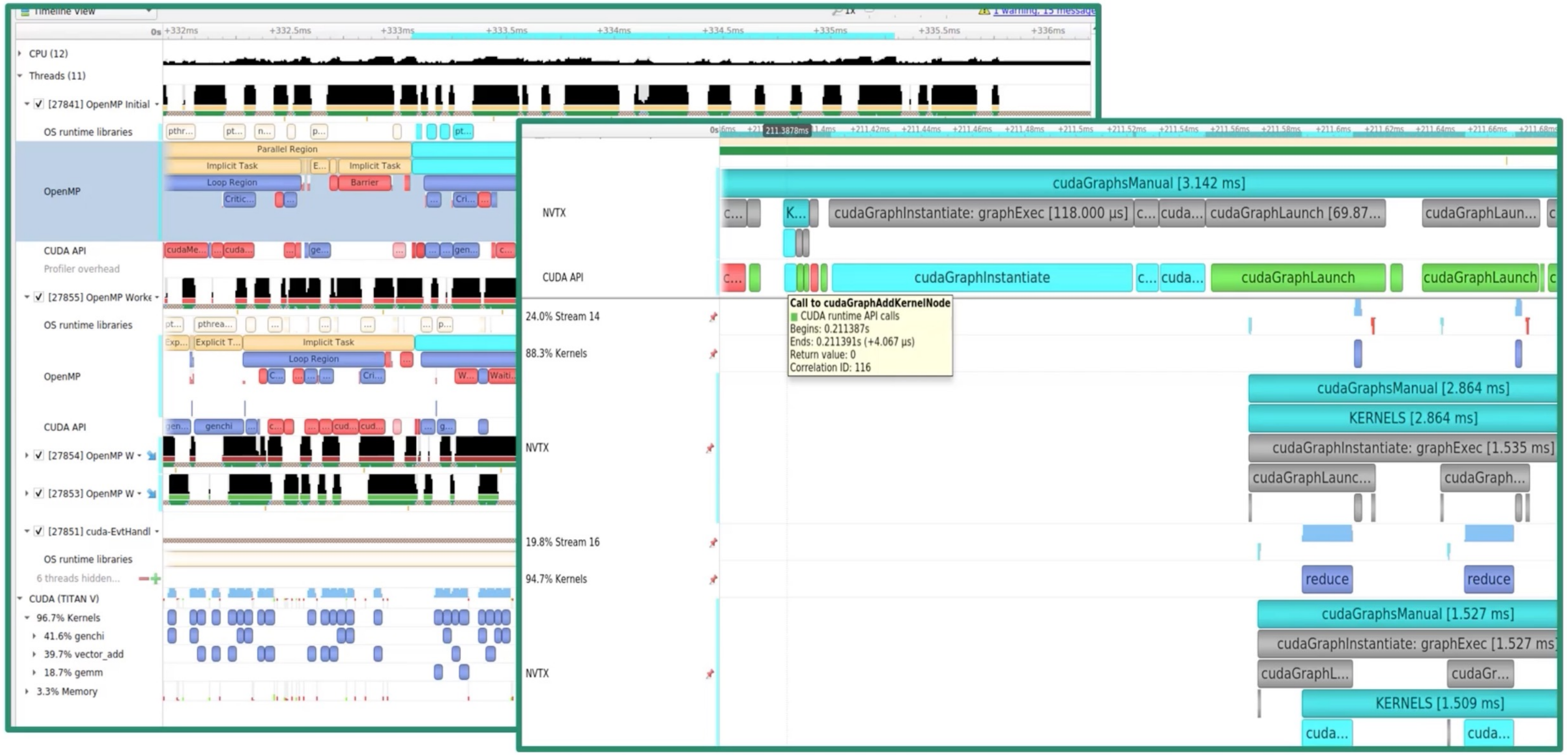
Scalable system-wide performance analysis tool

- Low-overhead multi-node, multi-GPU profiling
- Visualize millions of events on a very fast GUI timeline
- Assess on timeline to narrow down frames/areas of the app to focus
- Locate optimization opportunities, CPU/GPU bottlenecks
- or gaps of unused CPU and GPU time - idle
- Balance your workload across multiple CPUs and GPUs
- Expert system GPU utilization analysis
- Detailed information, documentation, free download
  https://developer.nvidia.com/nsight-systems

NVIDIA NSIGHT SYSTEMS

Multi-level information

- CPU cores utilization
- MPI calls
- OpenMP, Pthreas
- OS runtime calls
- NVTX
- CUDA API calls
- HtD / DtH data transfers
- CUDA kernels / streams
- OpenACC
- CUDA libraries (cuBLAS, …), GPU HW metrics, UCX, NIC, …



NVIDIA NSIGHT SYSTEMS

PROFILING WITH NSIGHT SYSTEMS

GUI profiling and analysis

```
ml CUDA/12.4.0 Qt5
```

```
nsys-ui # On the allocated GPU compute node
```

- File -> New Project
- Select target for profiling... -> acnXX.karolina.it4i.cz (allocated GPU node)
- Enter binary (**absolute path** with arguments if necessary)
- Select tracing modules (CPU, OS, CUDA, GPU, NVTX,...)
- Start

Cmd line profiling + GUI analysis

```
nsys profile -t cuda,osrt --stats=true --gpu-metrics-device=0
-o profile_name ./program
```

```
nsys-ui # On login node
```

- File -> Open -> Select profile_name.nsys-rep

NVIDIA NSIGHT SYSTEMS - EXAMPLE

```
git clone https://code.it4i.cz/training/intro2hpc.git
```

```
ml CUDA/12.4.0
```

```
cd hpcintro24/intro2hpc/5_gpu_accelerators/handson
```

```
vim vector_add.solution.cu # int count = 123456789;
```

```
nvcc -g -O2 -gencode arch=compute_80,code=sm_80 vector_add.solution.cu -o vector_add
```

Barbora sm_70
Karolina sm_80

```
./vector_add
```

- Perform profiling of vector_add example:

```
nsys profile -t cuda,osrt --stats=true --gpu-metrics-device=0 -o vector_add ./vector_add
```

- An extra CUDA examples:

```
git clone https://github.com/NVIDIA/cuda-samples.git
```

POP COE

- An EuroHPC **Centre of Excellence** (CoE)
  - On **Performance Optimisation and Productivity**
  - Promoting **best practices in parallel programming**
- Providing **FREE Services** for EU **academic AND industrial codes in all domains!**
  - **Performance Assessment**: initial analysis to identify performance issues and recommend approaches to address them
  - **Proof-of-concept**: explore the potential benefit of proposed optimisations by applying them to selected regions of the applications
  - **Correctness-check**: evaluate the correctness of hybrid MPI + OpenMP applications
  - **Energy-efficiency study**: investigate improvements of energy consumption or efficiency
  - **Advisory study**: ongoing consultancy for customers that choose to implement proposed optimisations on their own

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

HLRS

RWTH AACHEN UNIVERSITY

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

IT4I

inesc id lisboa

Teratec

USEFUL LINKS

VI-HPS – Association of institutions developing tools and providing training

- Overview of the tools with a description: https://www.vi-hps.org/cms/upload/material/general/ToolsGuide.pdf

Nvidia tools for GPUs: Nsight Systems and Nsight Compute

Intel performance tools: VTune and Advisor

Database of code patterns and best practices developed in POP: co-design

Docs + further reading:

- https://docs.linaroforge.com/23.1.2/html/forge/performance_reports/index.html
- https://docs.linaroforge.com/23.1.2/html/forge/map/index.html
- https://software.intel.com/content/www/us/en/develop/articles/intel-advisor-roofline.html

Radim Vavřík
radim.vavrik@vsb.cz

IT4Innovations National Supercomputing Center
VSB – Technical University of Ostrava
Studentská 6231/1B
708 00 Ostrava-Poruba, Czech Republic
www.it4i.cz

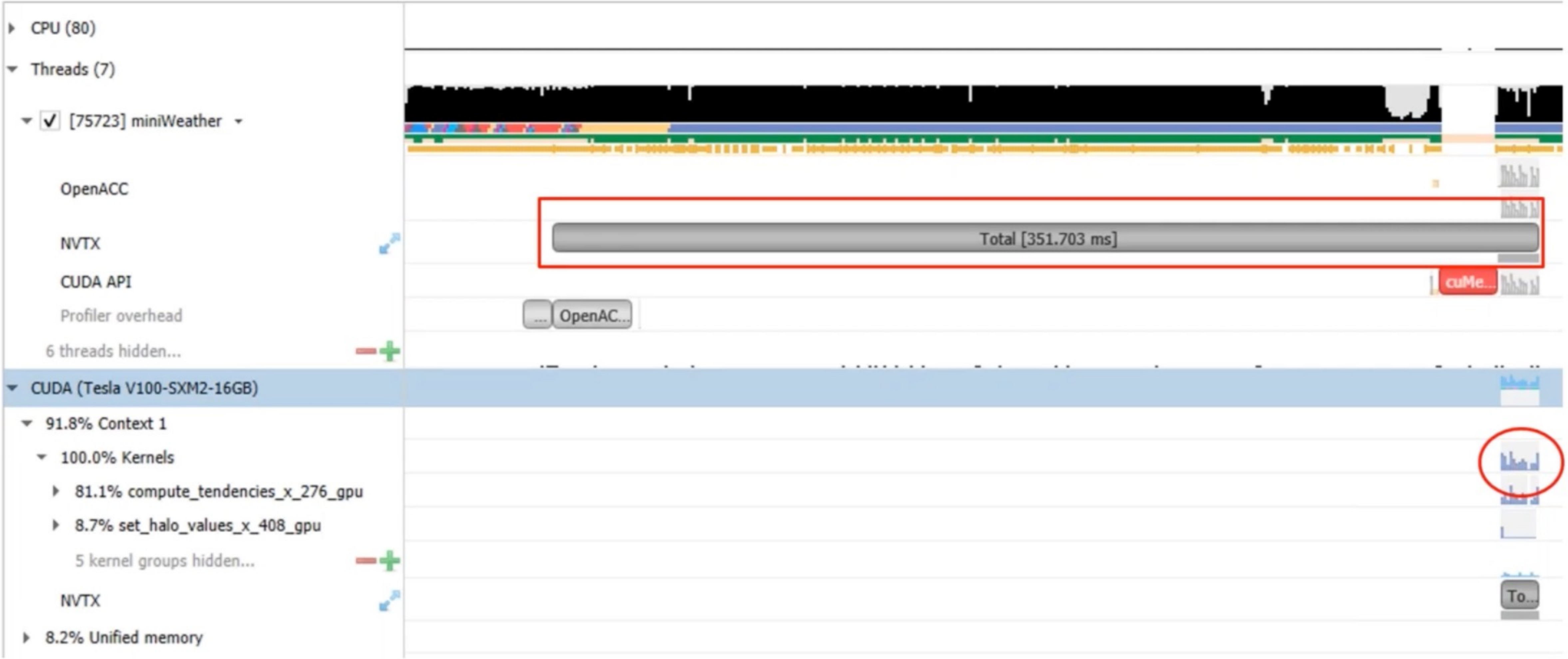
VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research, Development and Education

MINISTRY OF EDUCATION, YOUTH AND SPORTS

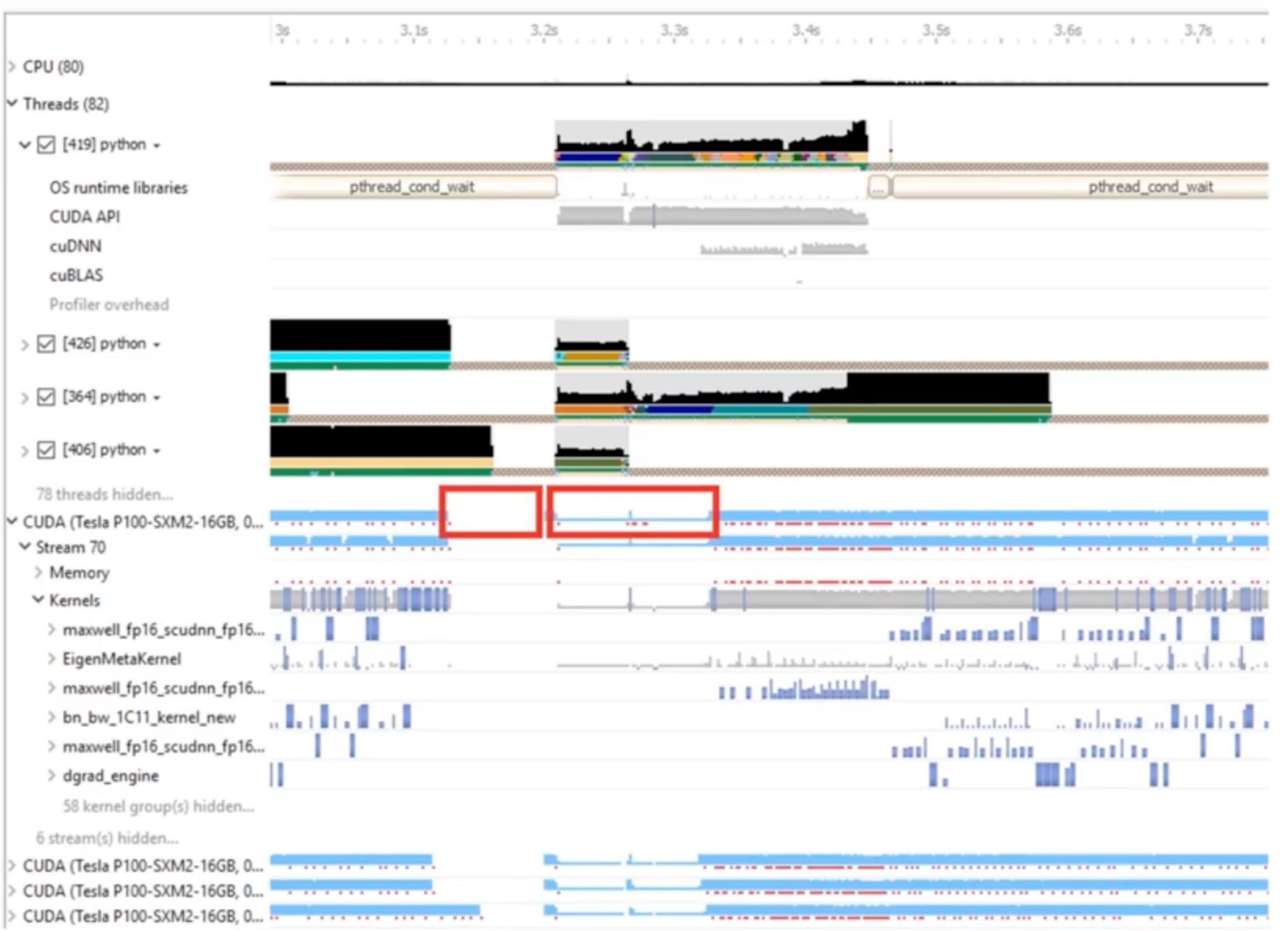
ANALYSIS WITH NSIGHT SYSTEMS

Only small portion of application accelerated (for real-world apps)



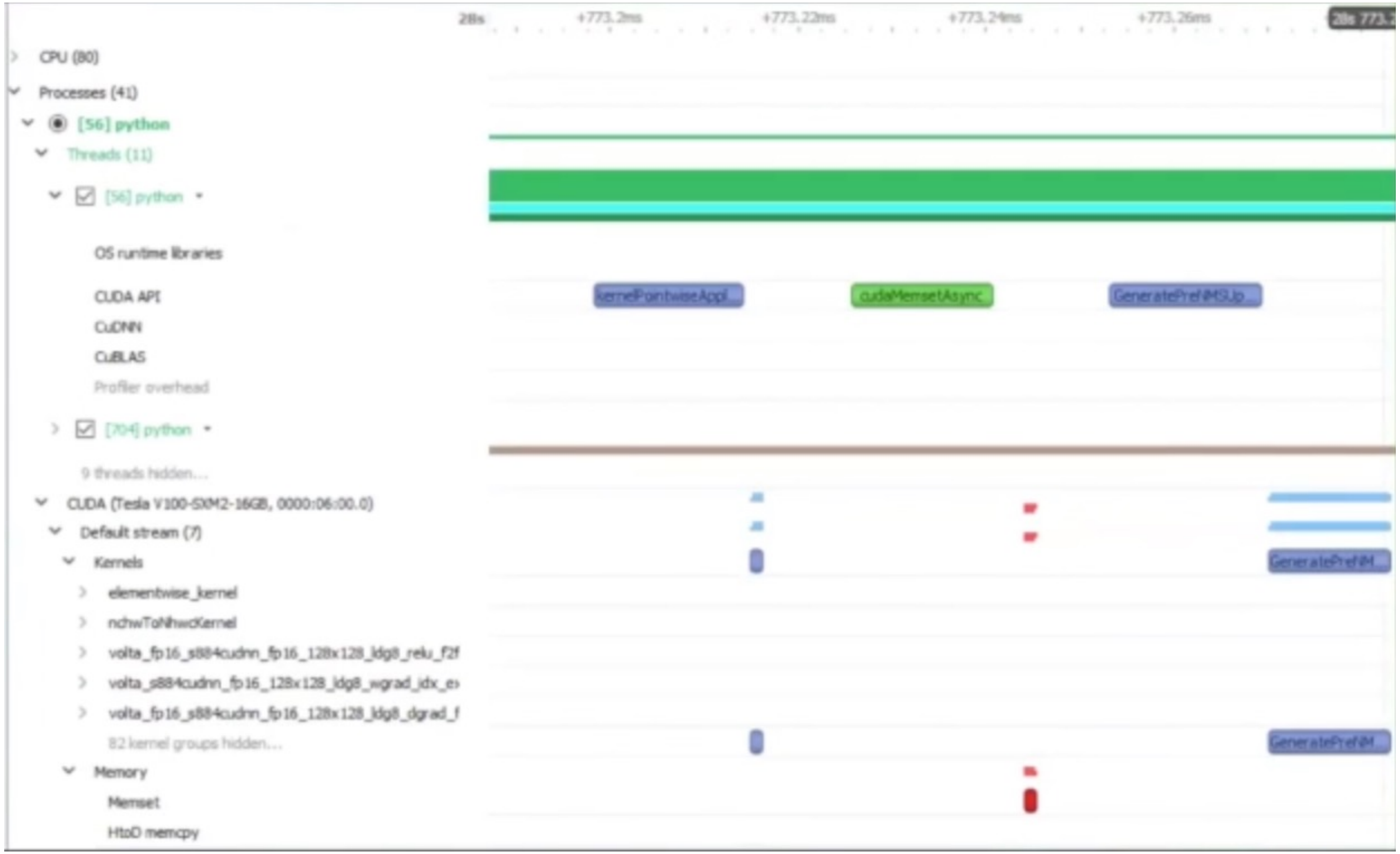
ANALYSIS WITH NSIGHT SYSTEMS

GPU idle/low utilization of detailed zoom (because of Pthread creation)



ANALYSIS WITH NSIGHT SYSTEMS

Fusion opportunities: CPU launch cost + small GPU work size -> GPU idle



ANALYSIS WITH NSIGHT SYSTEMS

cudaMemcpyAsync behaving synchronously – DtH pageable memory -> Mitigate with pinned memory

ANALYSIS WITH NSIGHT SYSTEMS

GPU idle caused by stream synchronization