Krishna Manda[1], **Ayushya Pare**[1], Michael Commer[2], Martin Errenst[3], Matthias Hellmann[2], Marc-André Hermanns[4], Petra Mutzel[1]
[1]University of Bonn [2]University of Cologne [3]University of Wuppertal [4]RWTH Aachen University

HPC.nrw

Performance Made Visible | A Tool-based Exploration of HPC Applications

Motivation

- Performance analysis in HPC does not require deep HPC expertise.
- Visualizations can enable easy understanding of performance behavior and potential bottlenecks.
- Performance analysis is an important sous-chef in the HPC code development kitchen.

Find all run configurations and job script on our Github page!

Linaro Performance Reports

A sampling-based tool that provides a **high-level overview** of the application performance together with **recommendations of likely bottlenecks** and **next steps**. It serves as a **good entry point** for a subsequent comprehensive performance analysis.

CPU

A breakdown of the 91.8% (1145.8s) CPU time:

Single-core code	1.7%	20.0s
OpenMP regions	98.3%	1125.8s
Scalar numeric ops	11.4%	130.9s
Vector numeric ops	24.5%	280.3s
Memory accesses	64.1%	734.6s

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

MPI

A breakdown of the 8.2% (102.9s) MPI time:

Time in collective calls	51.6%	53.2s
Time in point-to-point calls	48.4%	49.8s
Effective process collective rate	59.5 bytes/s	
Effective process point-to-point rate	81.3 MB/s	

Most of the time is spent in collective calls with a very low transfer rate. This suggests load imbalance is causing synchronization overhead; use an MPI profiler to investigate.

The point-to-point transfer rate is low. This can be caused by inefficient message sizes, such as many small messages, or by imbalanced workloads causing processes to wait.

Memory

Per-process memory usage may also affect scaling:

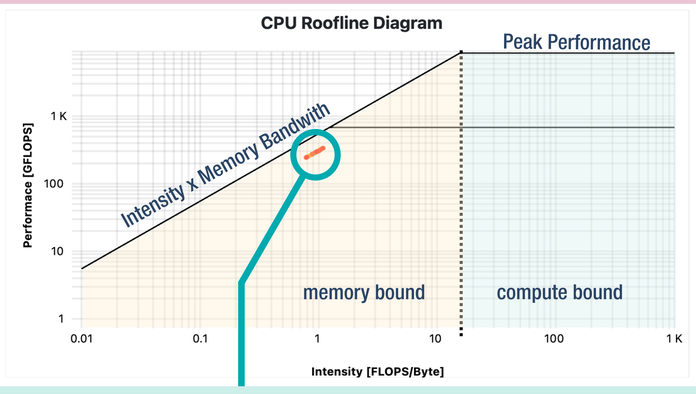
Mean process memory usage	2.50 GiB
Peak process memory usage	4.31 GiB
Peak node memory usage	51.0%

There is significant variation between peak and mean memory usage. This may be a sign of workload imbalance or a memory leak.

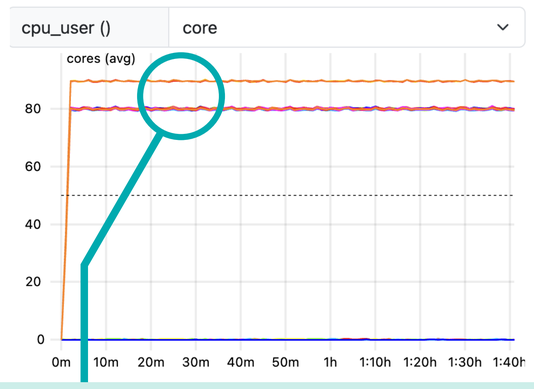
The peak node memory usage is modest. Running with fewer MPI processes and more data on each process may be more efficient.

Cluster Cockpit

A **job monitoring dashboard** that shows how your job performs over time. It provides a **basic overview** of performance values as an **entry point** for further investigation.



Functions and applications left of the knee are **memory-bound**. This means cores are not computing at their peak, but rather wait for data to be loaded.



A **ultilization plot** can unveil a **cpu-load imbalance**. This means not all cores perform on the same level.

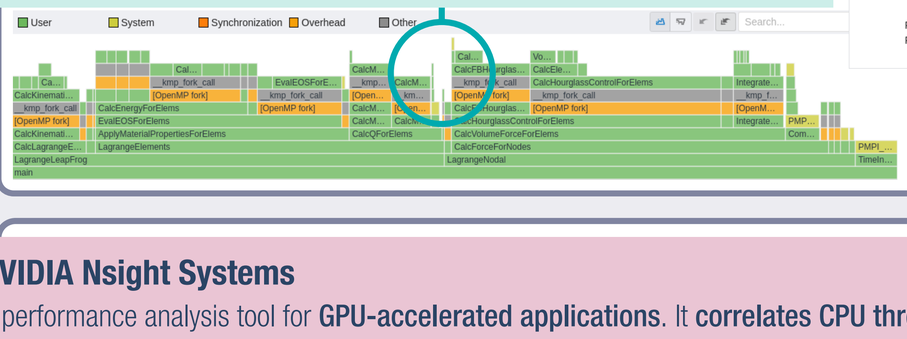
Intel VTune

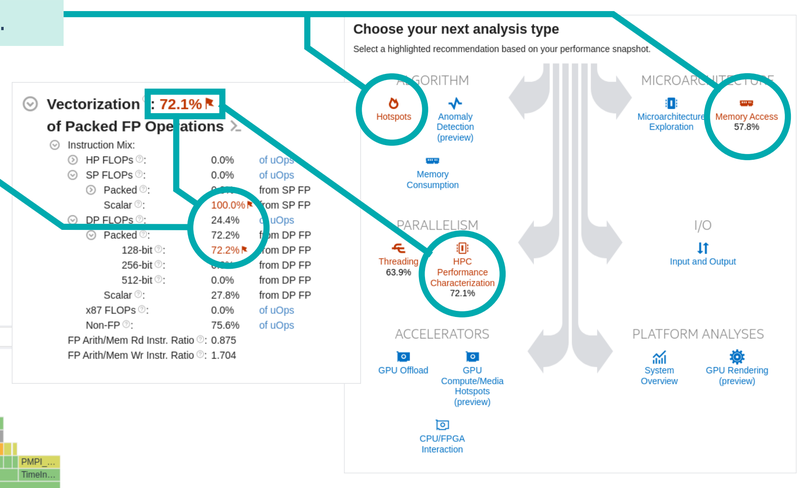
A sampling-based profiler to **characterize an application** by attributing performance measurements to currently active functions. Well-suited for an optimized **node utilization** in terms of CPU/accelerator and memory.

Performance snapshots **highlight potential issues** and recommend **next steps**.

Vectorization is a crucial factor for **achieving peak performance** on modern processors. In the shown example, next steps would involve the use of 256- or 512-bit packed instructions.

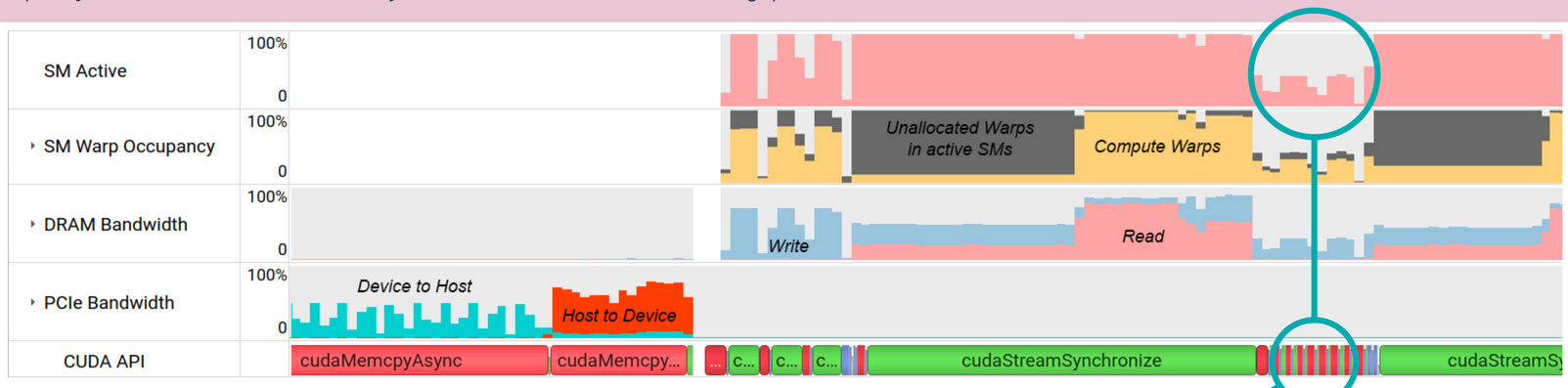
A **flamegraph** visualizes the function call hierarchy against the execution time. It allows for quick identification of **hotspots** and **deeply-nested function hierarchies**.





NVIDIA Nsight Systems

A performance analysis tool for **GPU-accelerated applications**. It **correlates CPU thread activity with GPU work** (kernels, memory transfers, and CUDA/NCCL calls), showing Streaming Multiprocessor (SM) utilization and communication throughput over time to **reveal bottlenecks**, such as low SM occupancy, transfer-bound execution, synchronization overhead, and idle gaps.

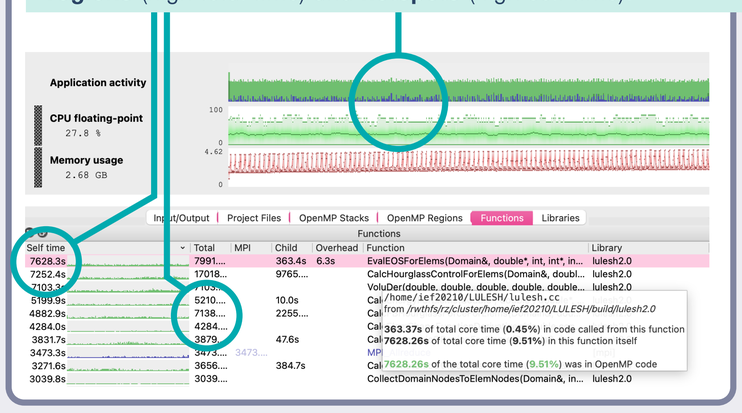


SM activity dips during consecutive ***cudaMalloc*** calls indicating potential for improvement by **pooling or using asynchronous allocations**.

Linaro MAP

A sampling-based tool with a **timeline overview** and **runtime profile** for different code regions.

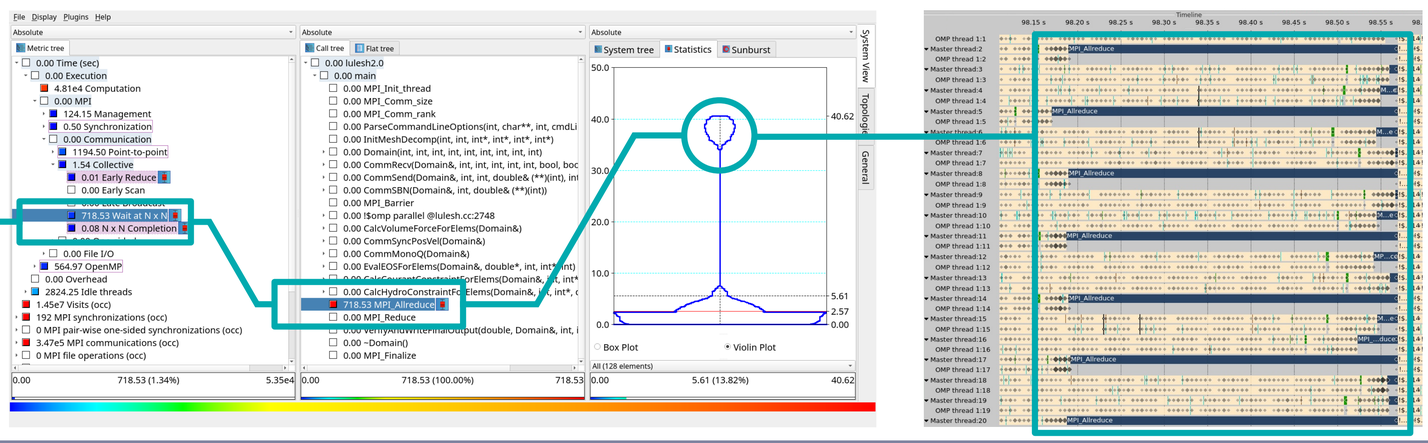
The timeline visualizes the **evolution of activites** during code execution. The function view let's you **identify important code regions** (high total time) and **hotspots** (high self time).



Score-P, Cube, Scalasca, Vampir

Tool **ecosystem** to take runtime summaries and trace measurements with a focus on **capturing interactions between threads and processes and their contexts**.

The enriched Cube profile written by Scalasca identifies **significant waiting time** that is distributed across a few processes. Zooming in on a corresponding trace segment in Vampir reveils the **load imbalance** and its context.



LIKWID

A low-level but easy-to-use tool to collect **performance counters**. Often used by monitoring systems.

Equal request and miss rate means almost **no requests are cached**. High miss ratio is characteristic for streaming workloads with **little temporal reuse**.

We provide free **consulting** and **training** to **scientists in NRW** to enable them to keep performance analysis a part of their software engineering process.

In cooperation with DIGITALE HOCHSCHULE NRW
Founded by: Ministry of Culture and Science of the State of North Rhine-Westphalia