DEEP Projects

Optimisation Cycles for Modular Supercomputing and Energy Efficiency

Alexander Geiß, Technical University of Darmstadt
Mathieu Stoffel, Eviden

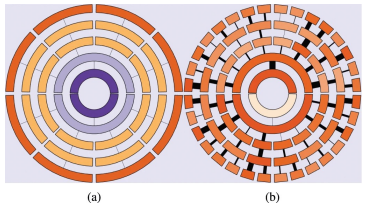
16.01.2024

The Research leading to these results has received funding from the European Commission's FP7, H2020, and EuroHPC Programmes, under Grant Agreements n° 287530, 610476, 754304, and 955606

A lot of tools in DEEP-SEA

DEEP Projects

Score-P
Scalable performance measurement infrastructure for parallel codes



Mitos

scalasca

MUSA

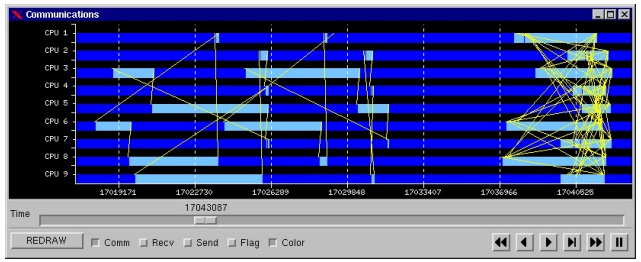
Zsim+DRAMsim3

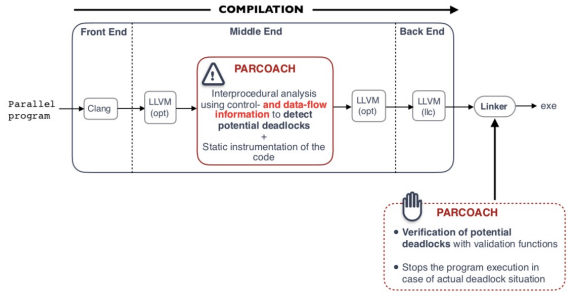
Extrae

Sys-sage

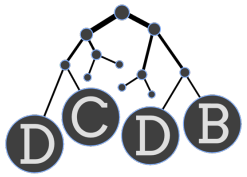
DaCe

Extra-P





HybroGen



Bull Dynamic Performance Optimizer

And more…

User perspective

I want to optimise an application.

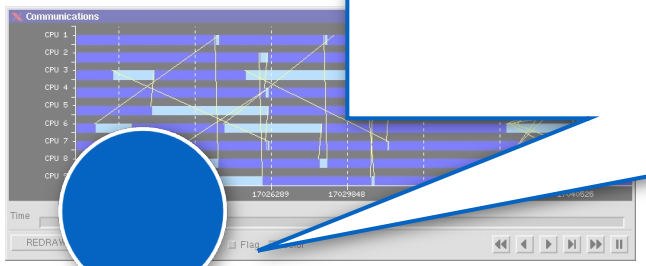
Which tools do I need?

How do they work together?

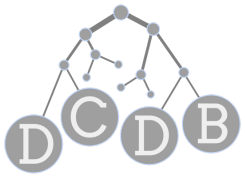
Score-P
Scalable performance measurement infrastructure for parallel codes

Zsim+D

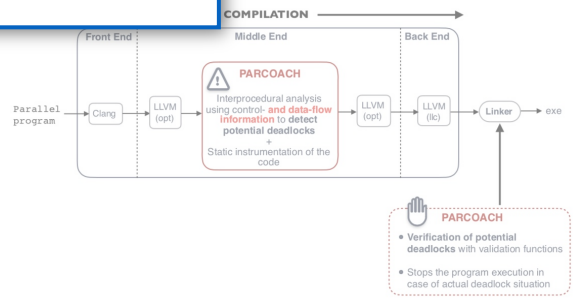
Sys-sage



And more…



Bull Dynamic Performance Optimizer

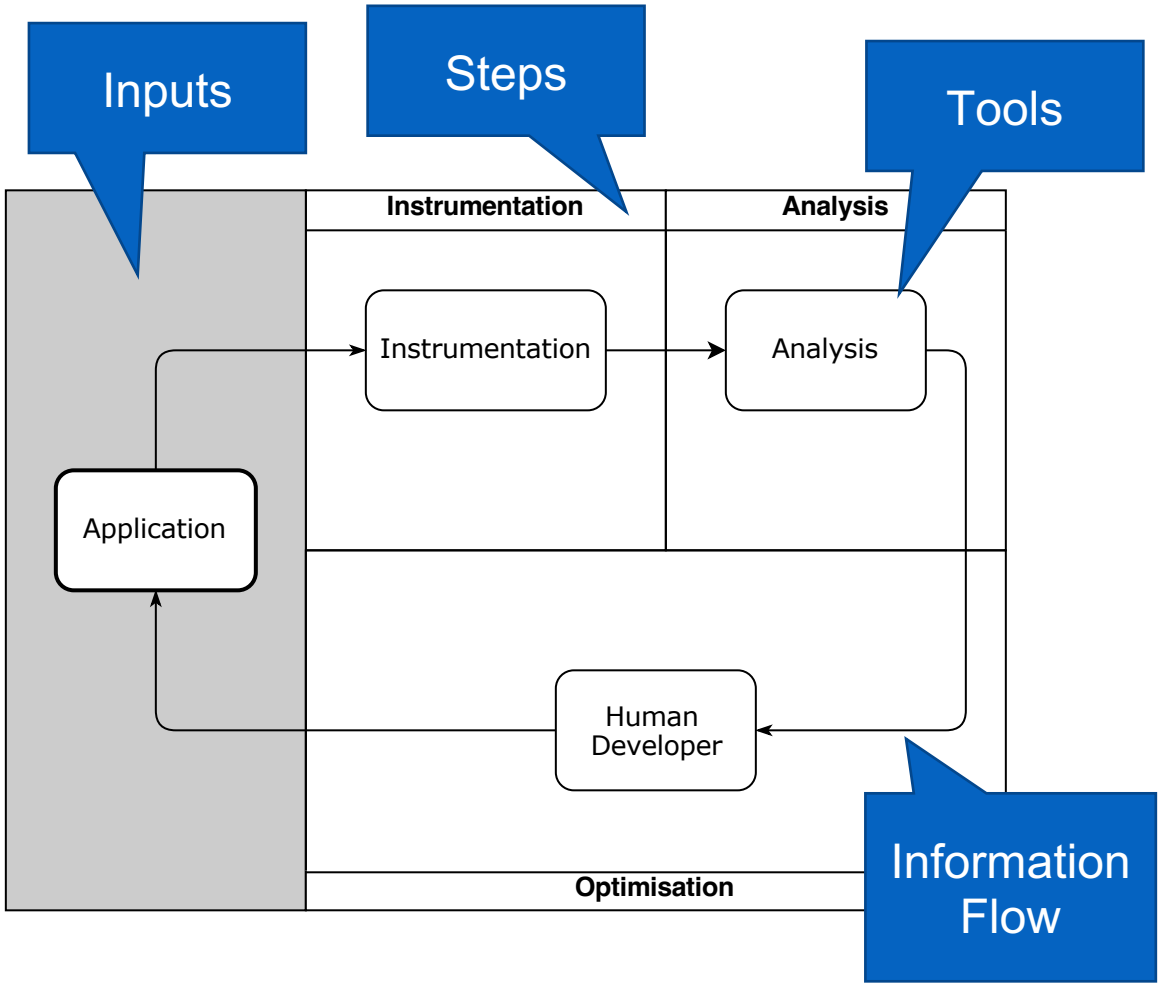


Optimisation Cycles

… represent the typical workflow of an application developer implementing and optimising an HPC-code

Optimisation Cycles

- Identify the software components that have to play together
  - Optimally support the developer
  - Efficiently utilise the resources of an MSA system
- Define the relevant interfaces between the connected components
  - Describe the information flow



Optimising for Modular Supercomputing

Alexander Geiß, Technical University of Darmstadt

DEEP-SEA

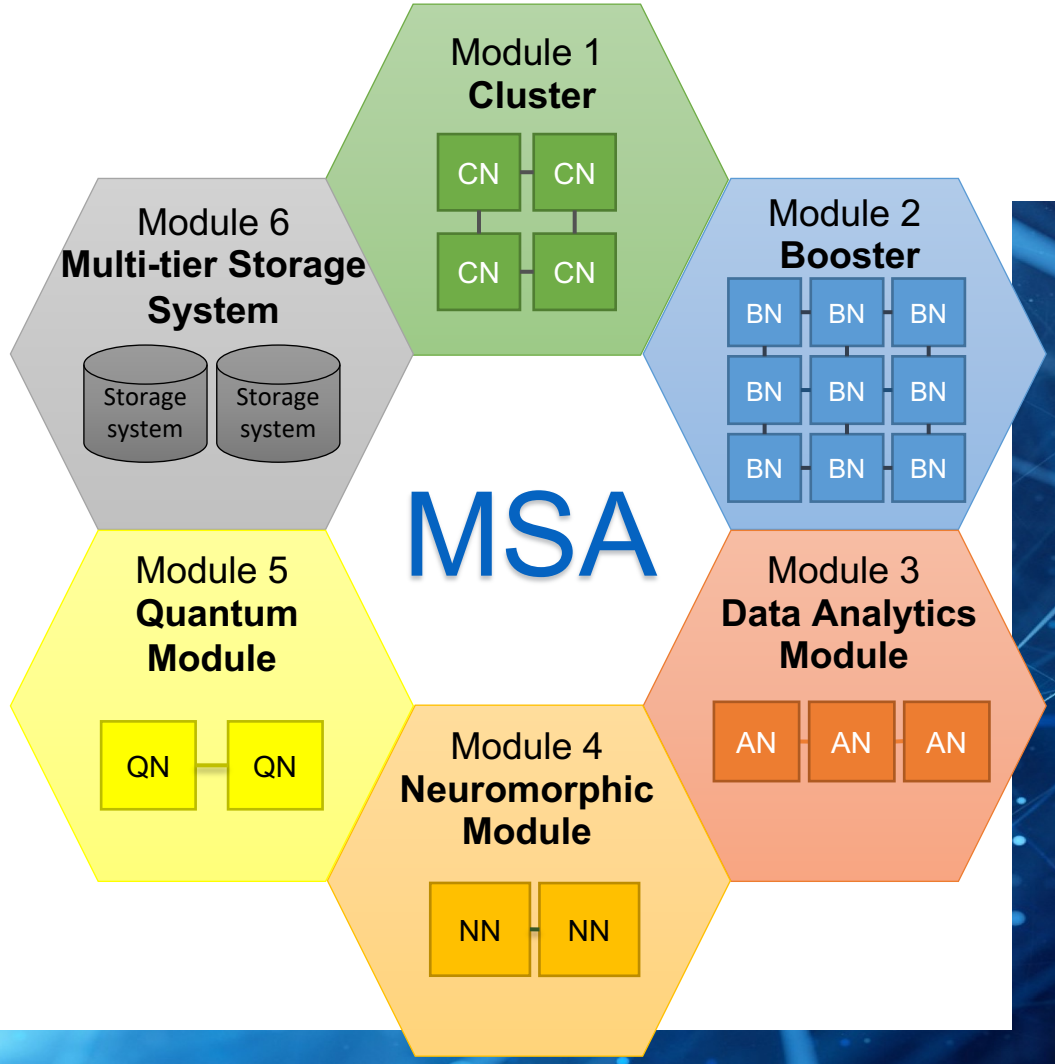
DEEP Projects

DEEP Software for Exascale Architectures

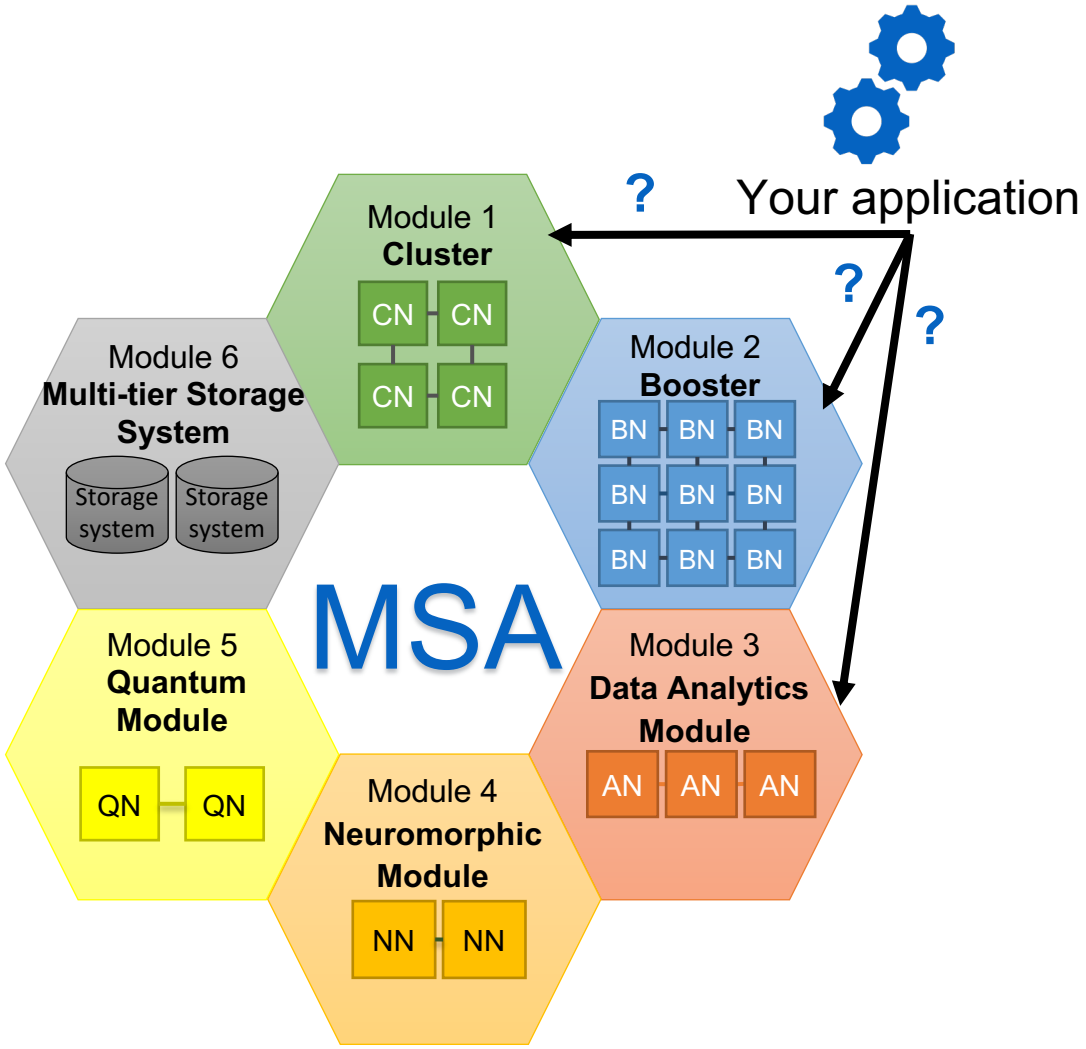
- Better manage and program compute and memory heterogeneity
- Targets easier programming for Modular Supercomputers
- Continuation of the DEEP projects series

Modular Supercomputing

- Cost-efficient scaling
- Effective resource-sharing
- Composability of heterogeneous resources

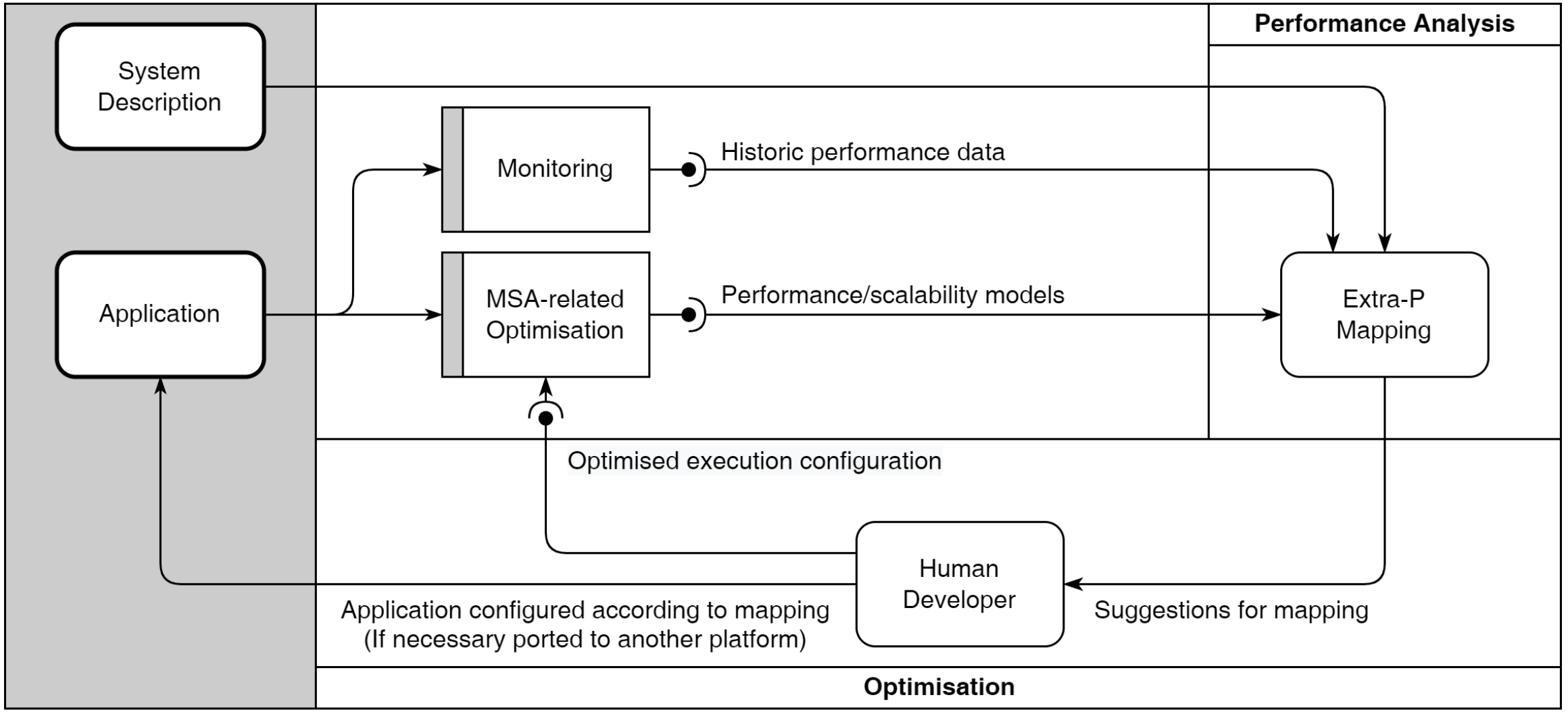


Which module(s) should run your application?

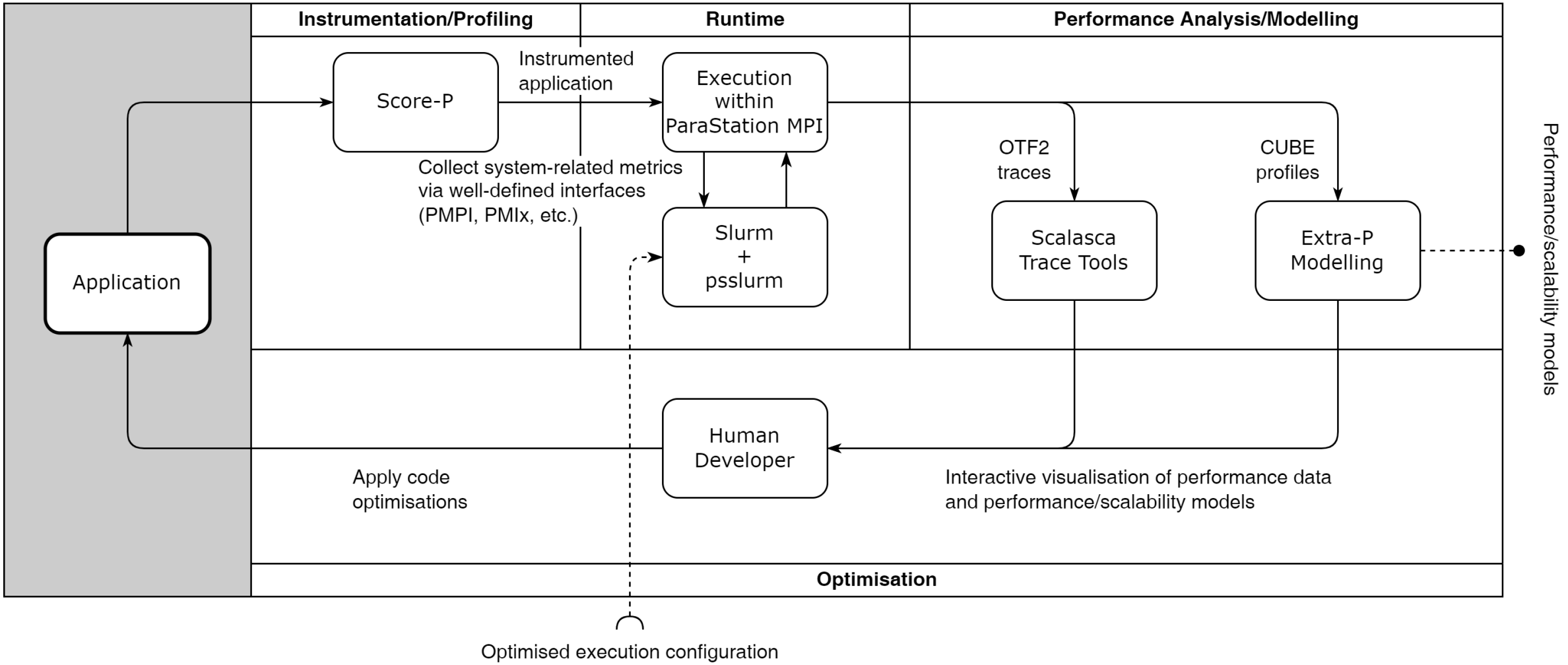


- You probably have a good idea
  - Guided by intuition and expectations
    - *e.g., the Booster is expected to be faster than the Cluster*
- Are these expectations fulfilled?
  - Hard to answer when parameters change
    - *e.g., problem size, number of nodes*
- The same issue exists on individual CPU-GPU nodes

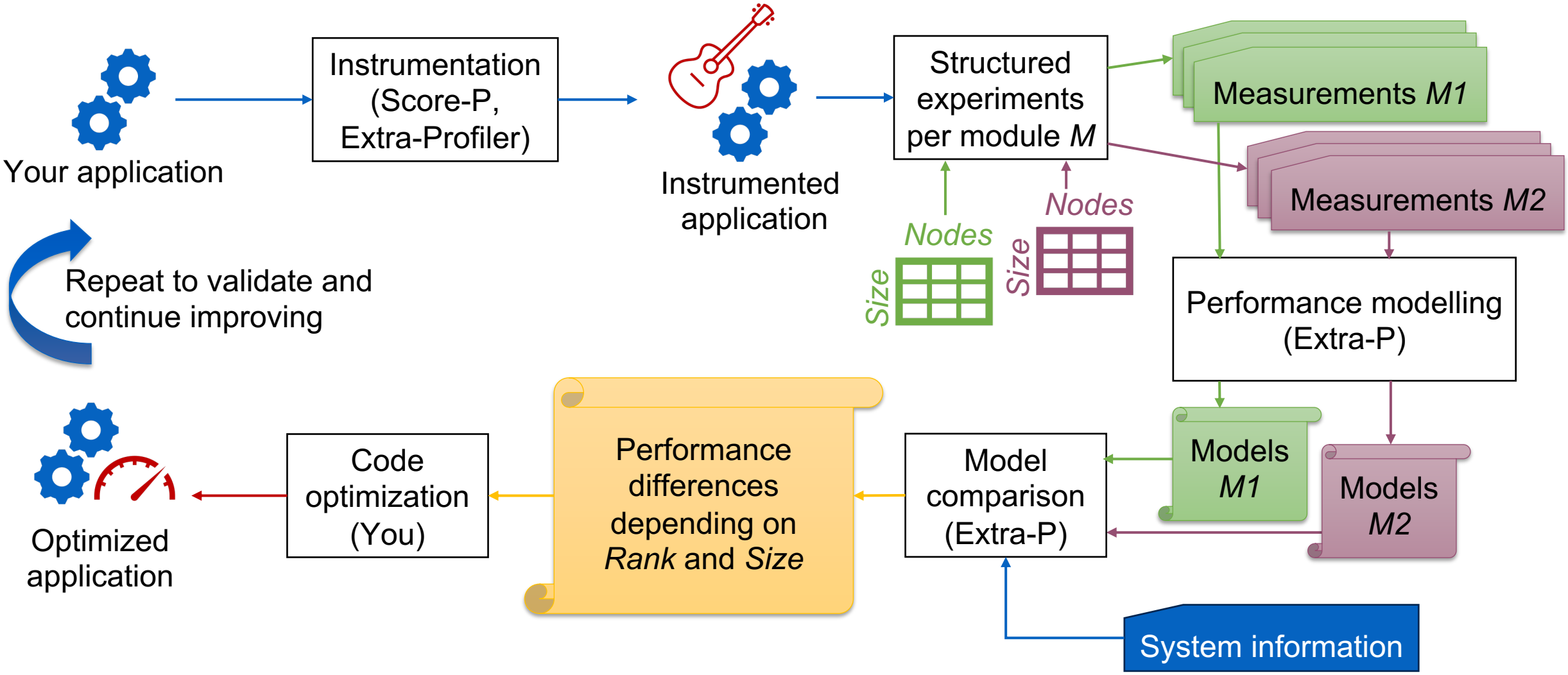
Application Mapping Optimisation Cycle



MSA-related Optimisation Cycle



Application Mapping OC (simplified)



Instrumentation and Measurement

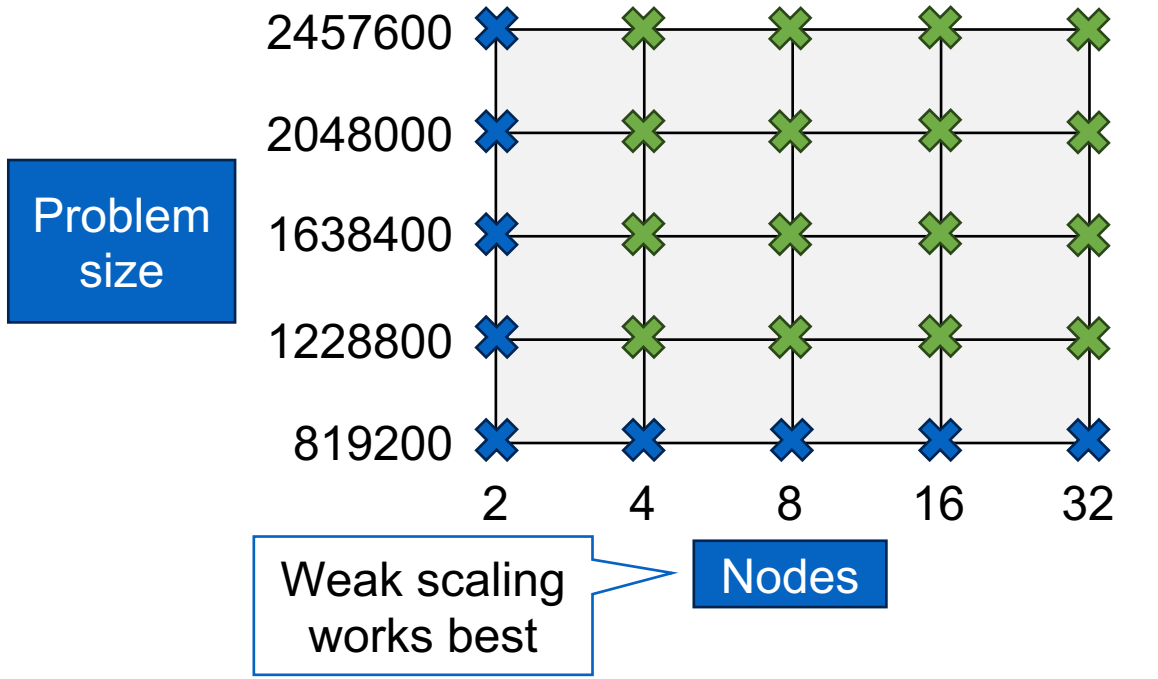
Cluster Nodes

- Use Score-P
  - Repeat measurement 5 times
  - Measure floating point operation count and DRAM accesses once

Booster Nodes

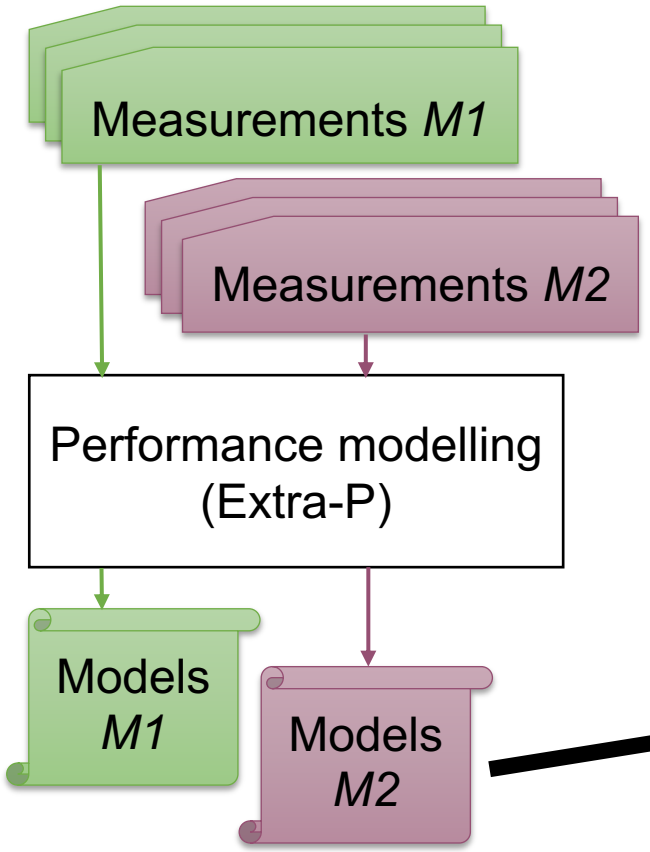
- Use the Extra-Prof CPU-GPU profiler
  - Repeat measurement 5 times

Measurement grid

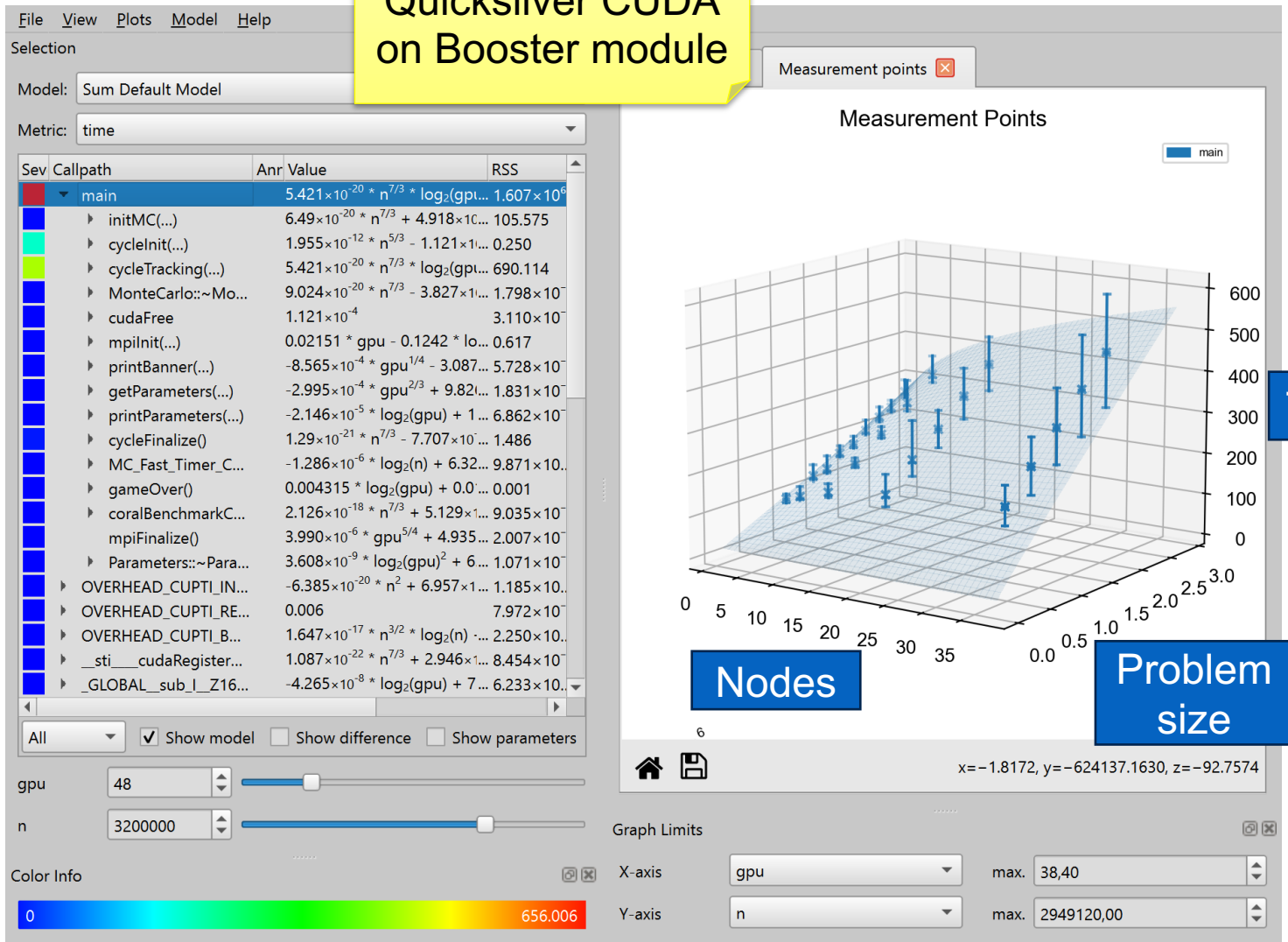


Example: Quicksilver mini-app
Solves a simplified dynamic monte-carlo particle transport problem

Performance Modelling: Extra-P



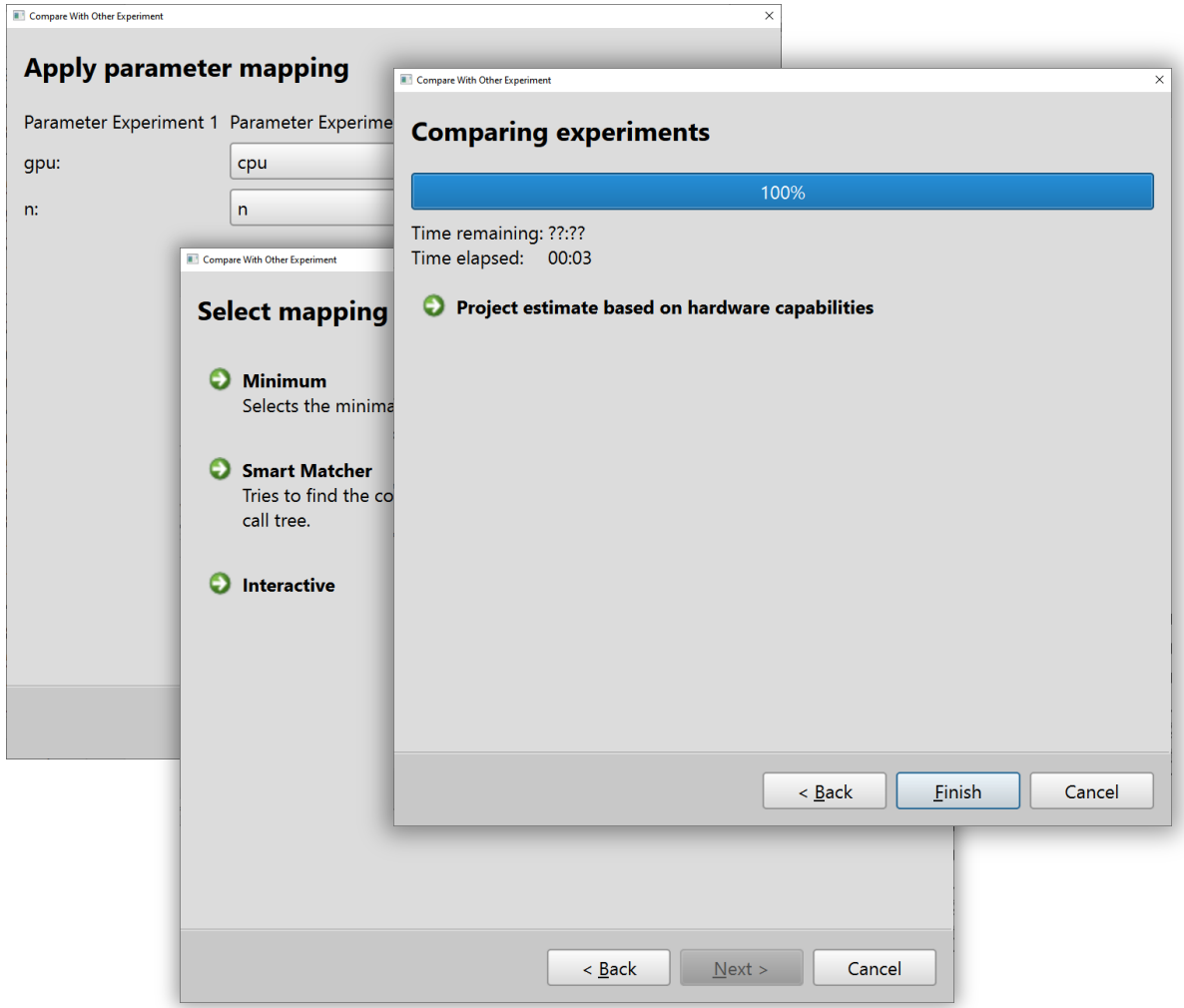
Quicksilver CUDA on Booster module

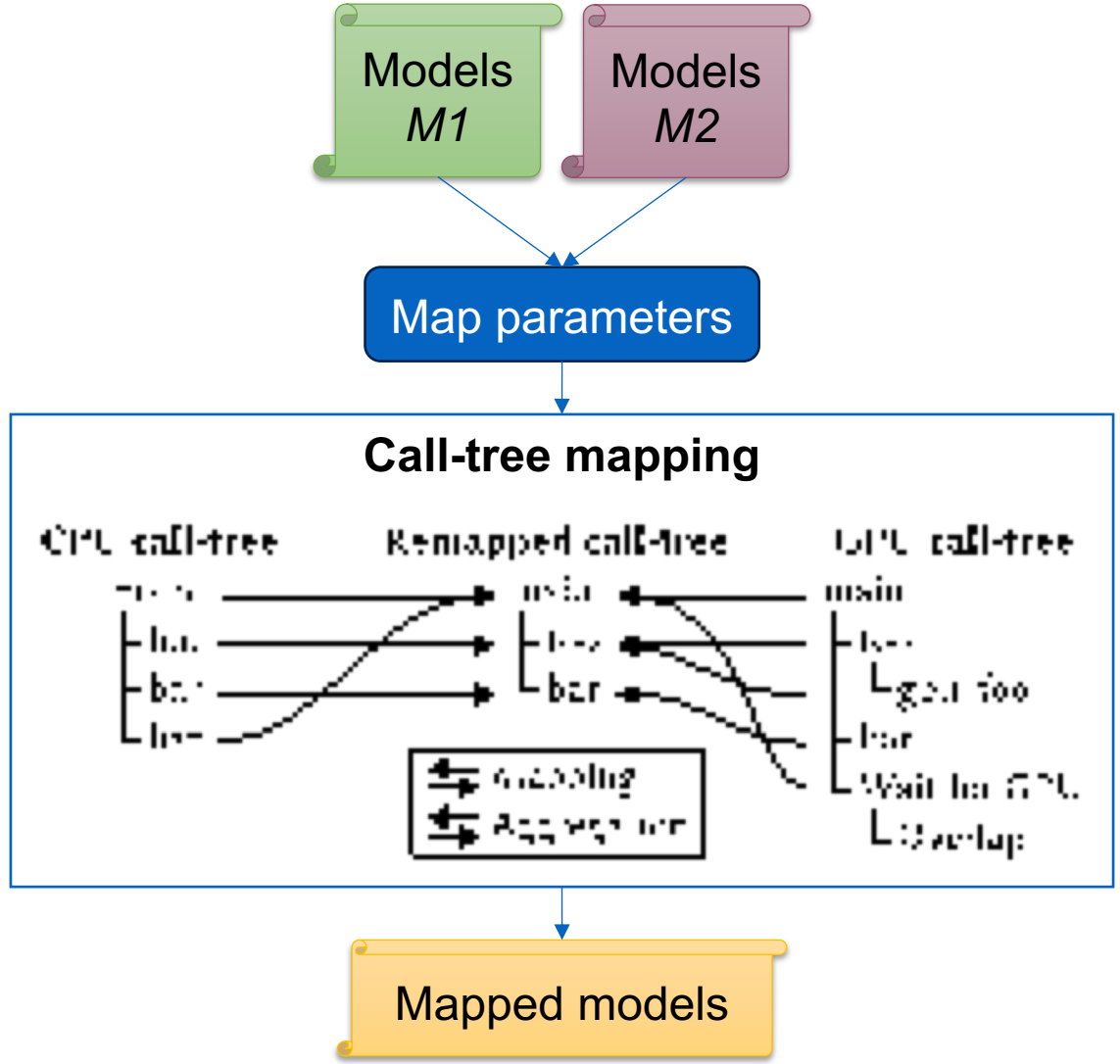


More about performance modelling with Extra-P:

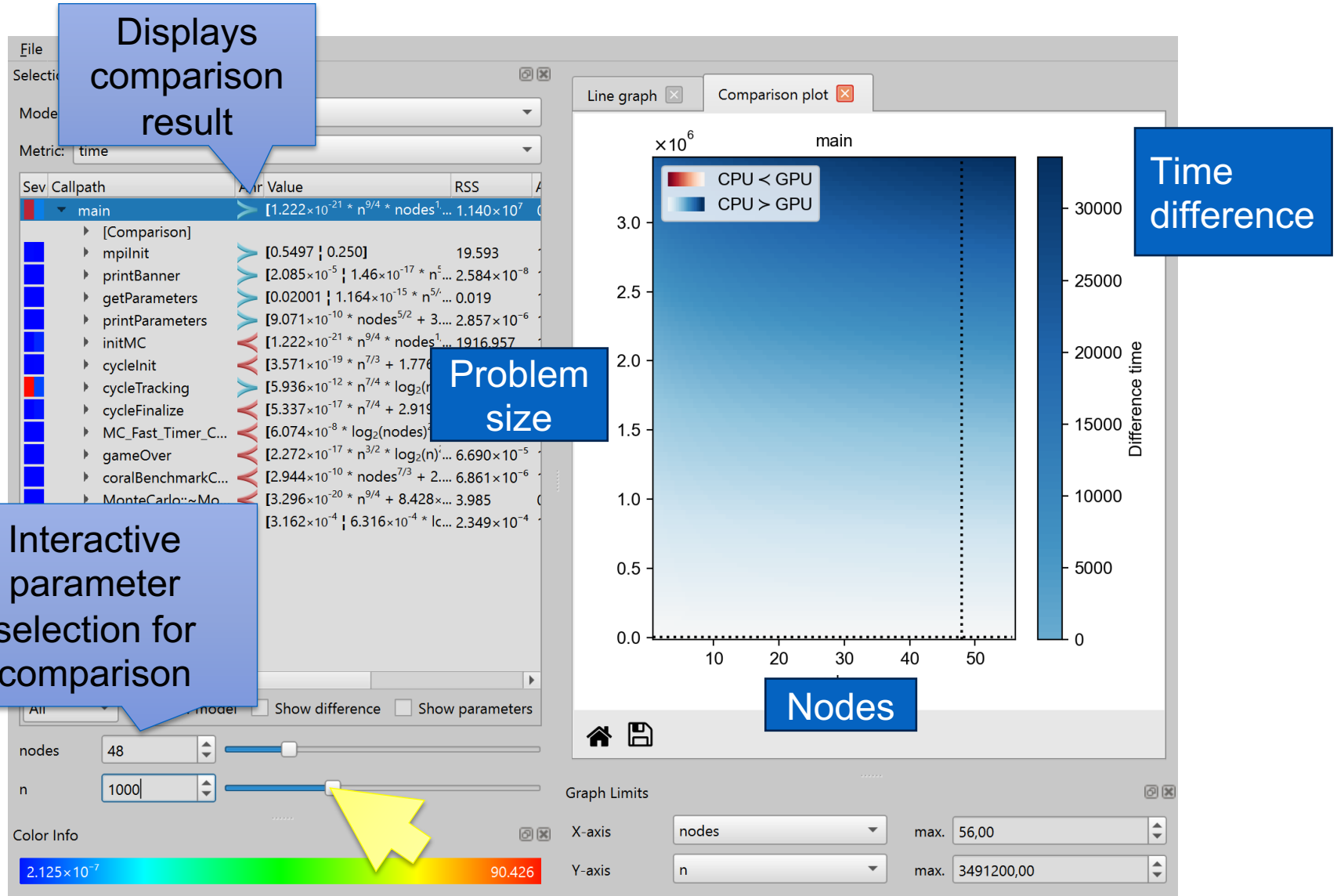
- https://youtu.be/Cv2YRCMWqBM
- https://github.com/extra-p/extrap

Comparison

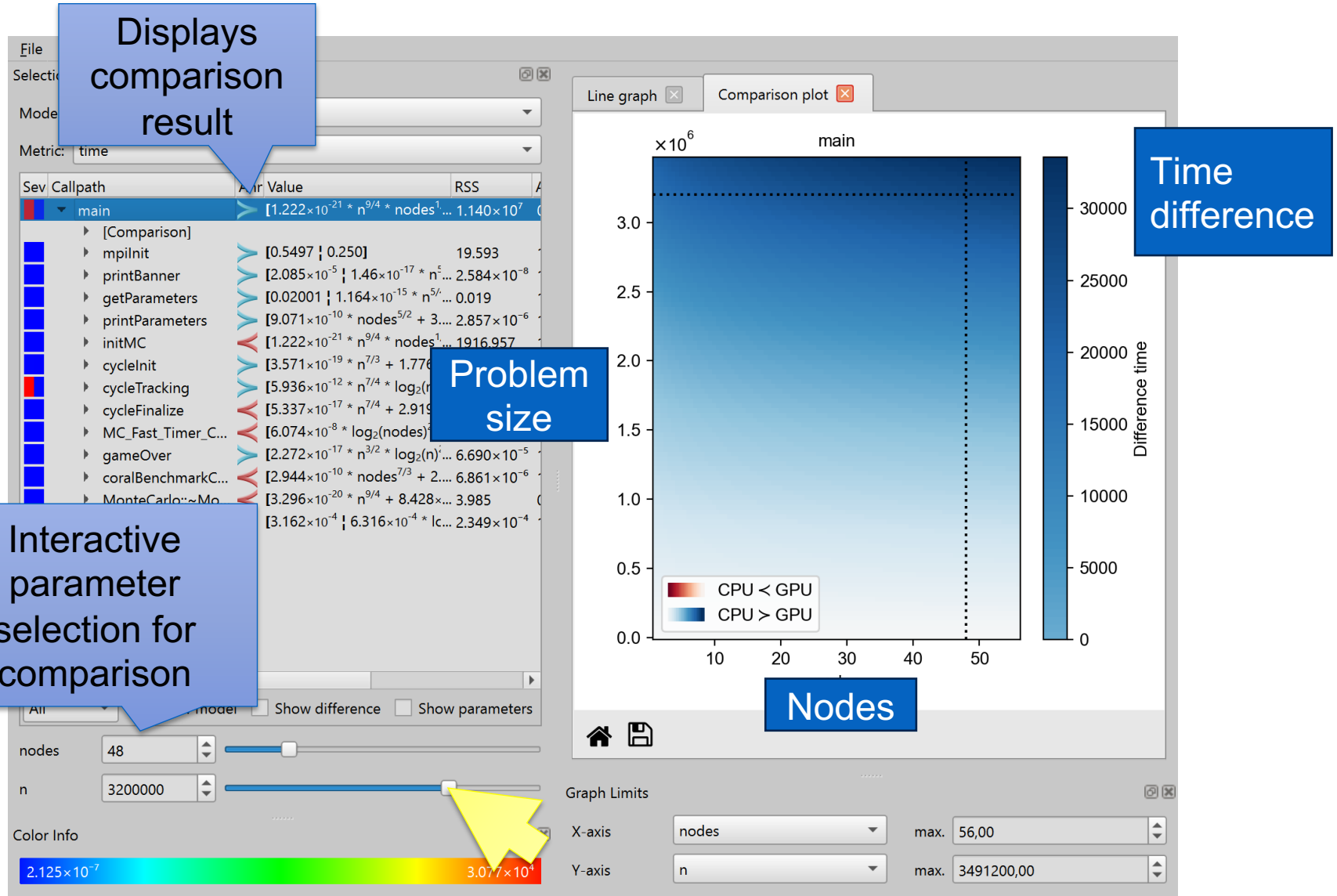

Compare With Other Experiment
Apply parameter mapping
Parameter Experiment 1
Parameter Experime
gpu:
cpu
n:
n
Compare With Other Experiment
Select mapping
Minimum
Selects the minima
Smart Matcher
Tries to find the co
call tree.
Interactive
< Back
Next >
Cancel
Compare With Other Experiment
Comparing experiments
100%
Time remaining: ??:??
Time elapsed: 00:03
Project estimate based on hardware capabilities
< Back
Finish
Cancel


Models
M1
Models
M2
Map parameters
Call-tree mapping
Mapped models

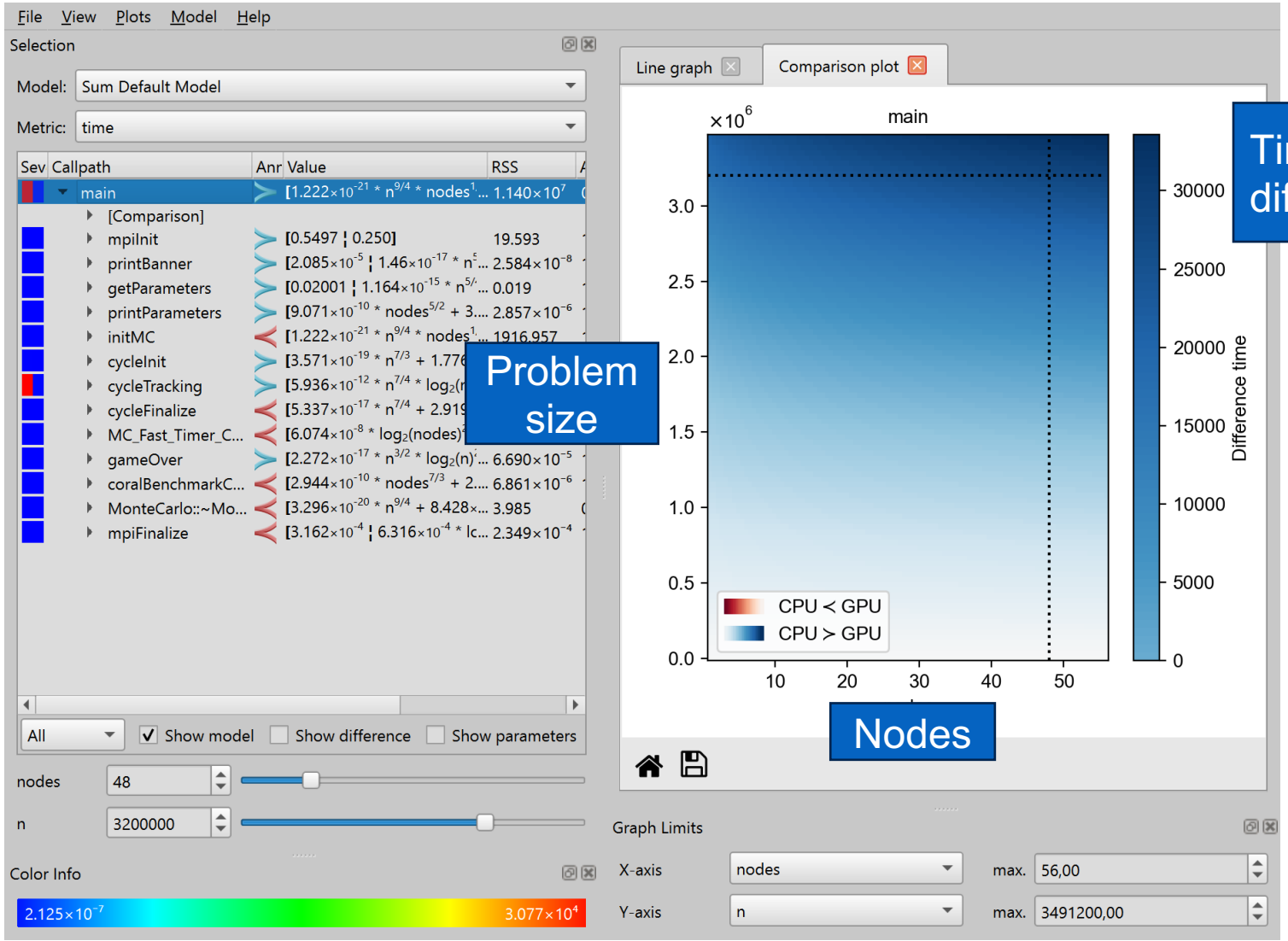
Interactive Comparison and Exploration



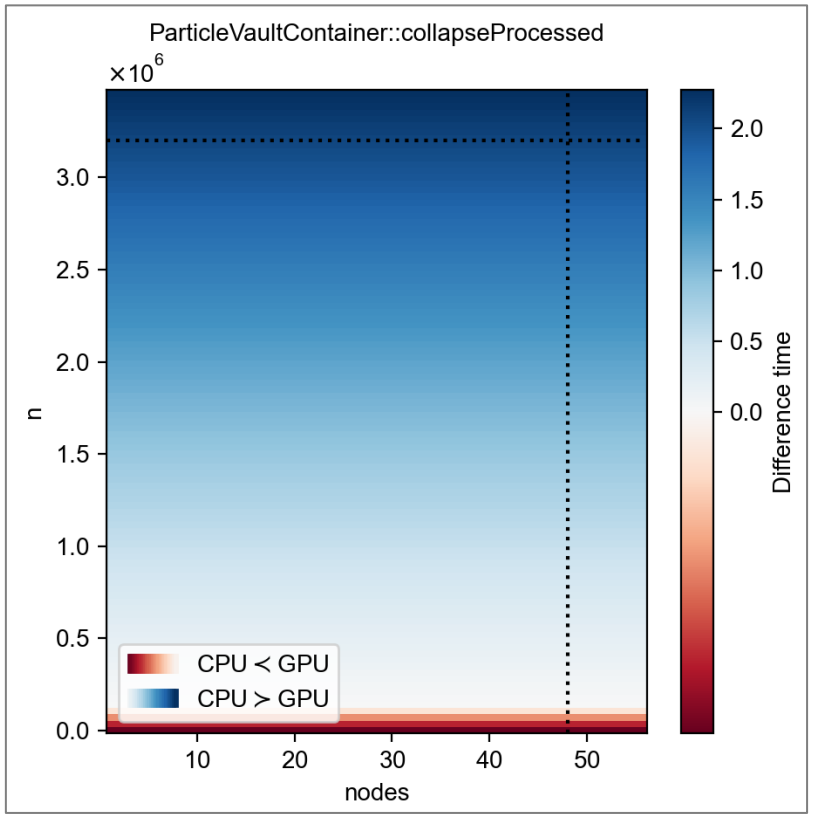
Interactive Comparison and Exploration



Interactive Comparison and Exploration

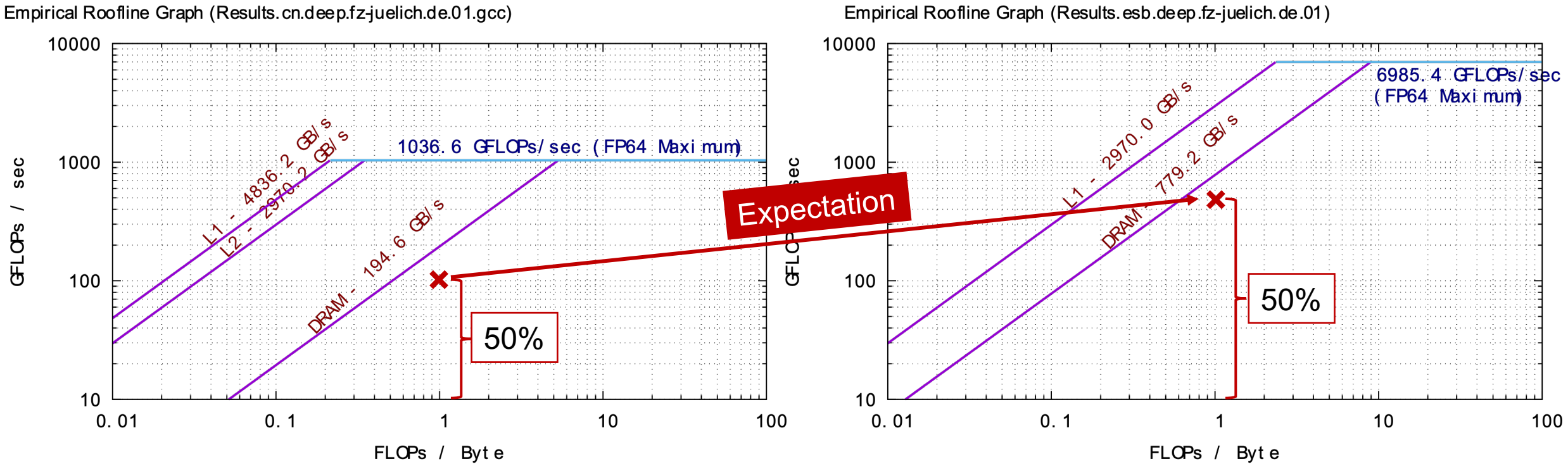


Time difference



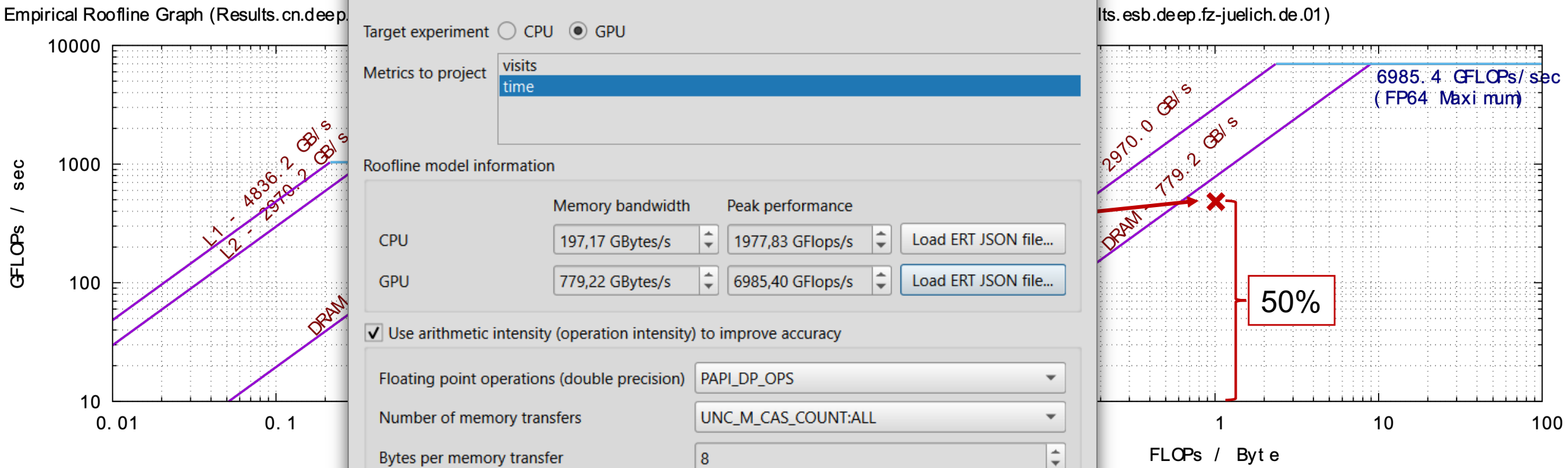
Determining the Expectation

- We formulate the performance expectation in terms of the roofline model

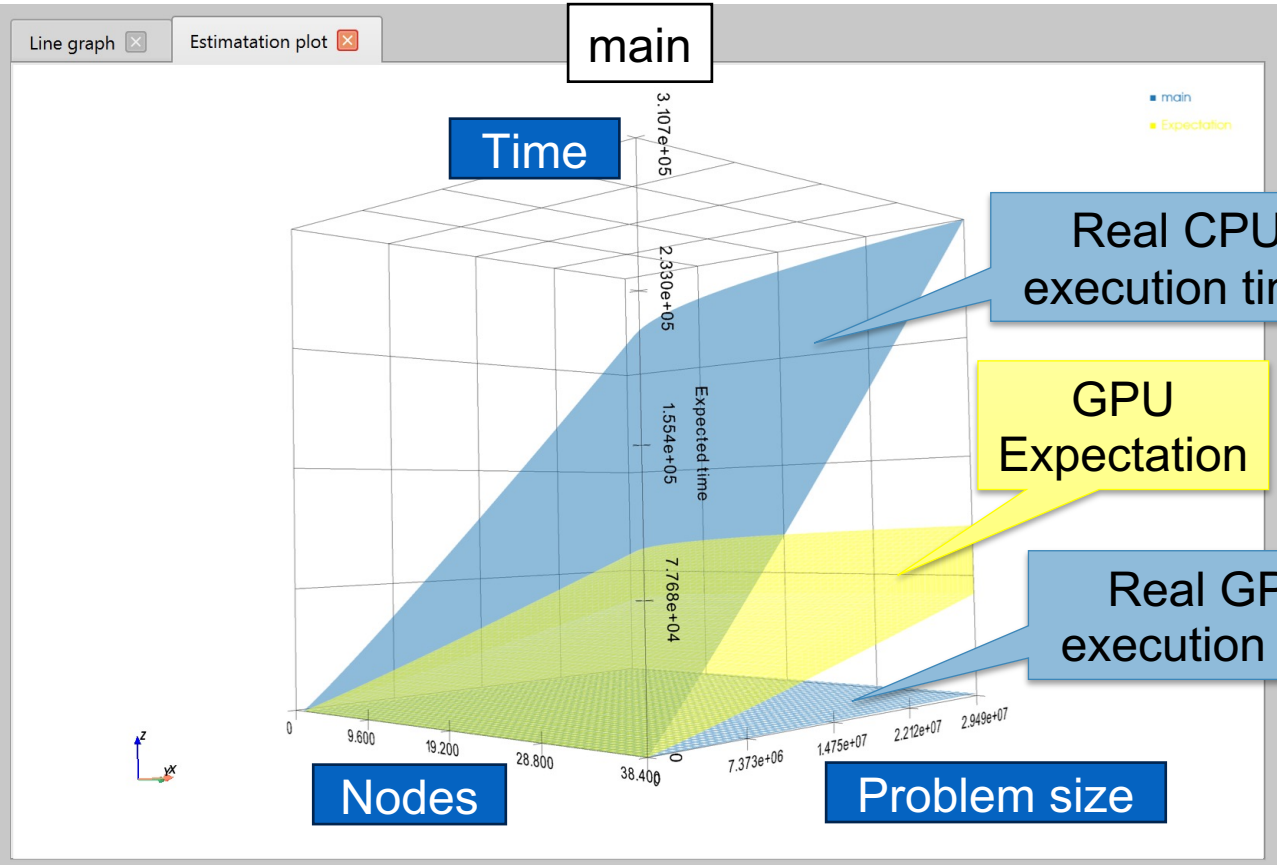


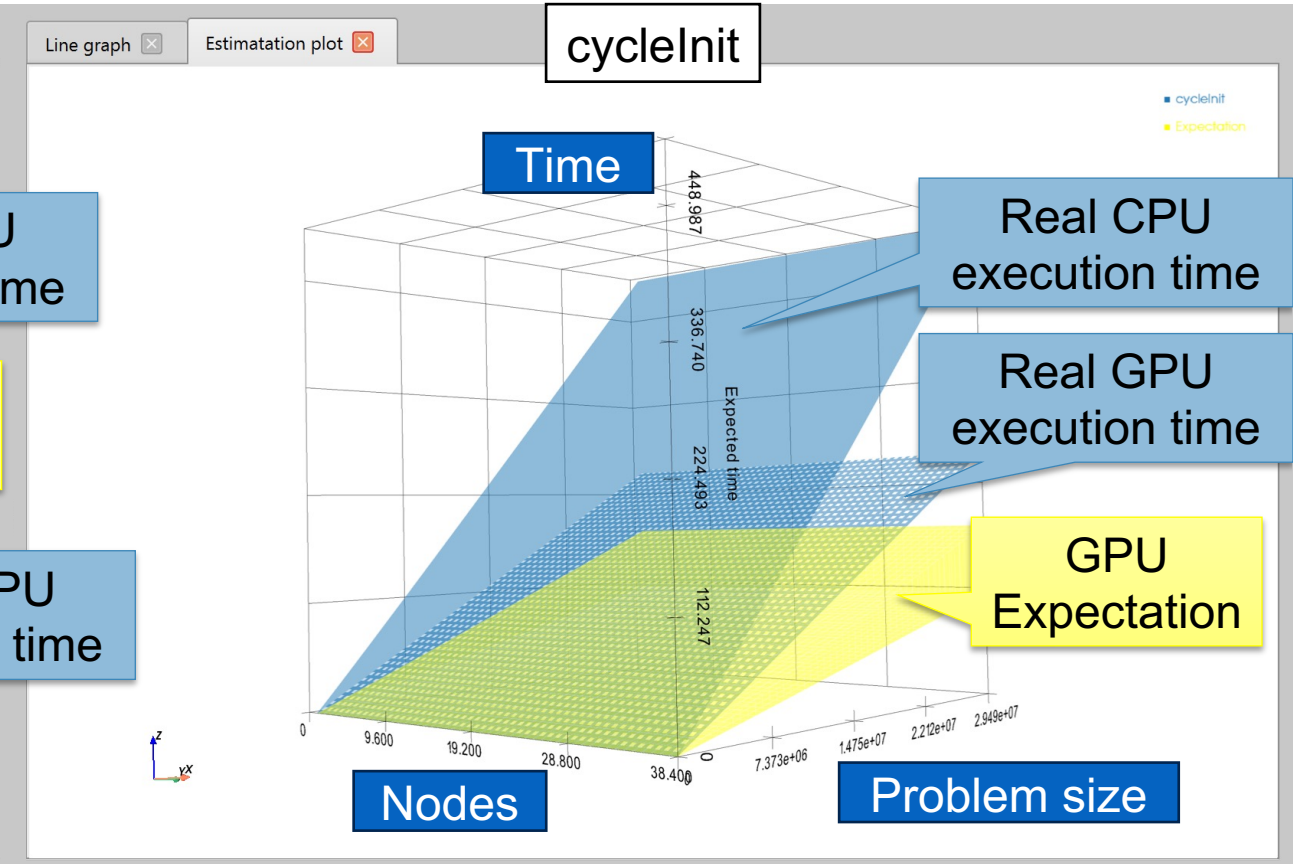
Determining the Expectation

- We formulate … of the roofline model

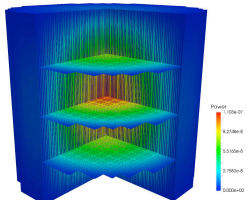


Comparison Against Roofline Expectation



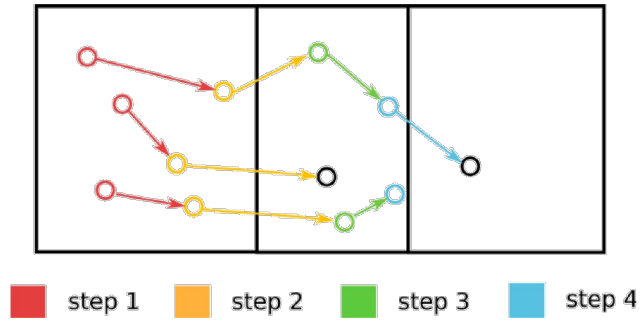


Use Case: PATMOS

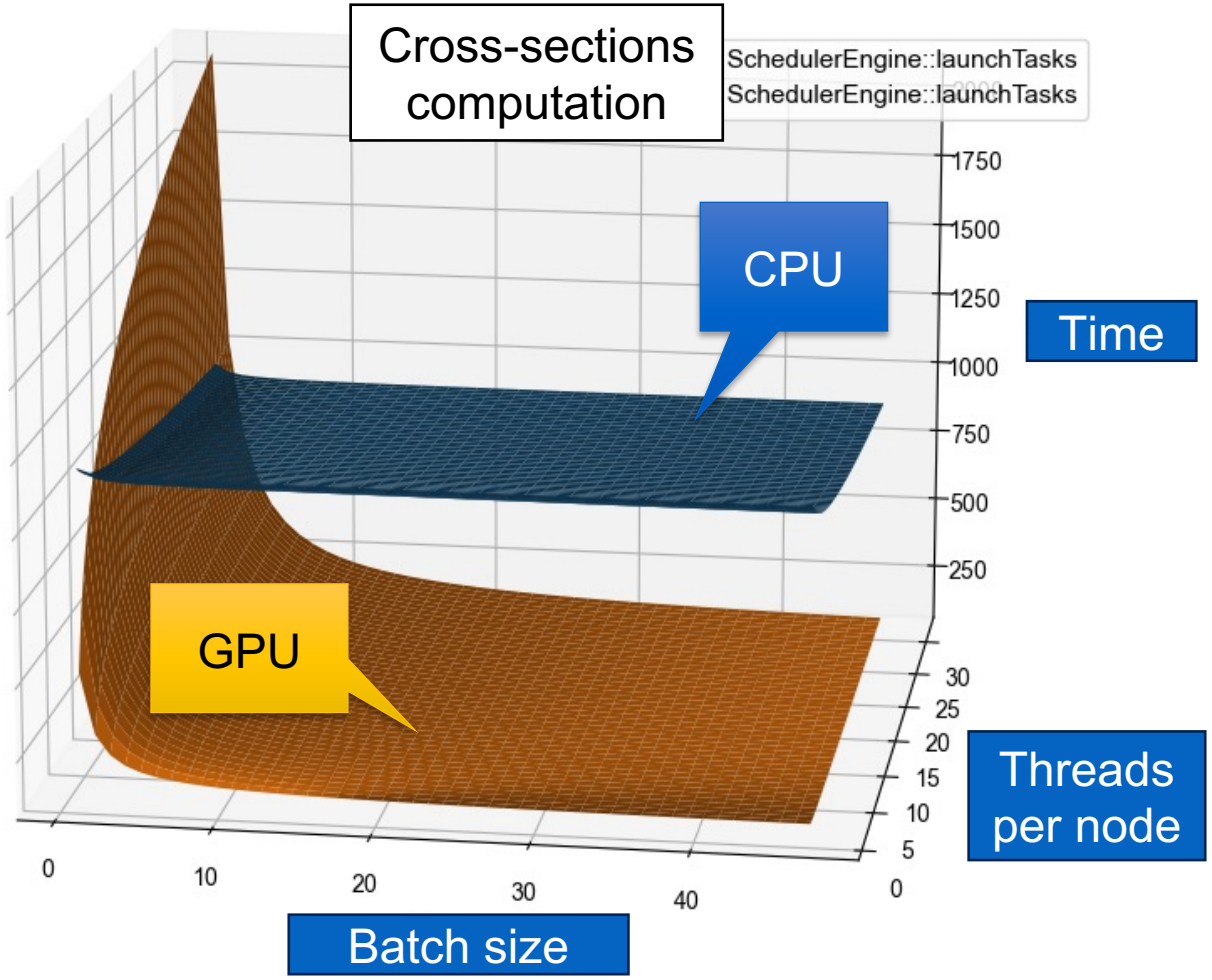


Solves the neutron transport equations to simulate evolution of physical quantities for complex systems

- Cross-sections computation represents 60% to 90% of total runtime
  - Porting cross section computation to GPU
  - Offload batch-size particles at a time



- **What is a good batch-size?**



Discussion

Optimizing for Energy Efficiency

Mathieu Stoffel, Eviden

The topic of energy-efficiency in HPC

- The most important metric still is performance
- Hardware tends to be more energy-efficient
- Kernels/applications not using the full computational power of the system have room for improvement of their energy-efficiency
- Resource management is still far from optimal regarding energy-efficiency

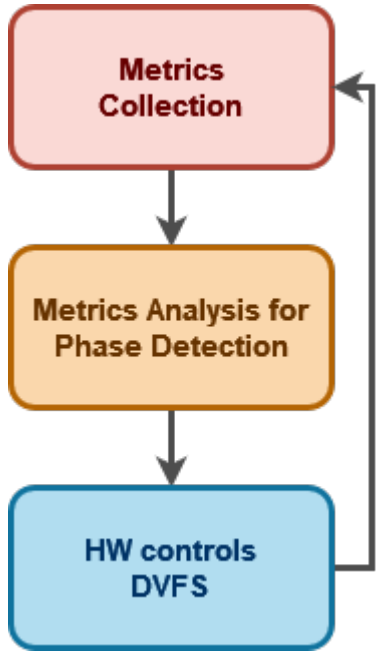
➔ Opportunities for optimizing the energy-efficiency of HPC applications at runtime

What is Bull Dynamic Power Optimizer (BDPO)?

- A lightweight tool to be run in parallel of an HPC application
- It aims at improving the energy-efficiency associated with the execution of an HPC application
- It is agnostic of the target HPC application
- It features two compatible modes: profiling and optimizing
- Its standard lifecycle is fully integrated with Slurm (and, hence, ParaStation)

How does BDPO work?

- One standalone instance of BDPO per compute node – one main control loop
- Monitoring of HW CPU-centric performance metrics through PMC (retired Instructions Per reference Cycle – IPC)
- Detection of phases with low computation intensity
- Enforcement of Dynamic Voltage Frequency Scaling (DVFS) accordingly to the phase detection step

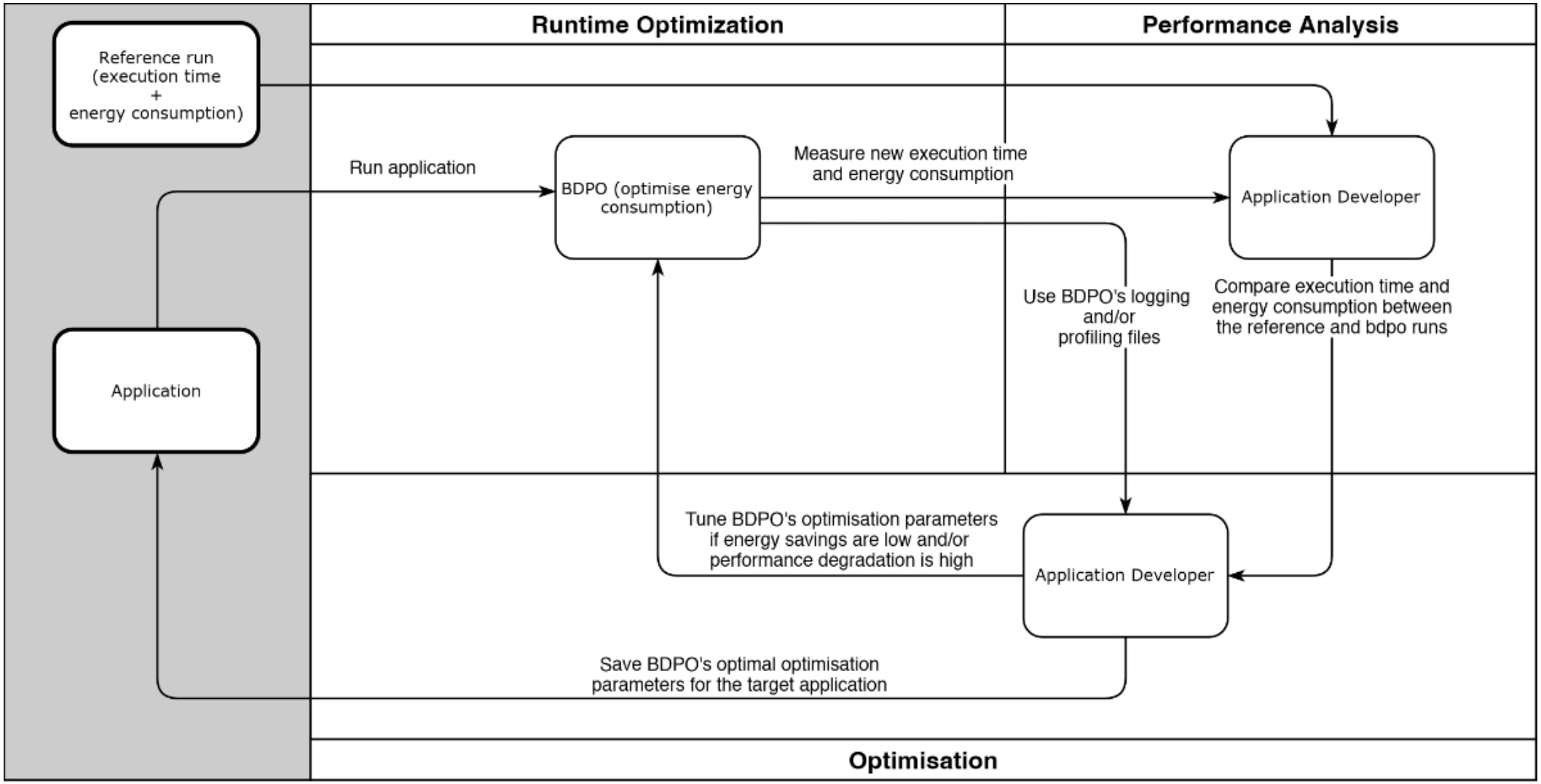


How does BDPO work?

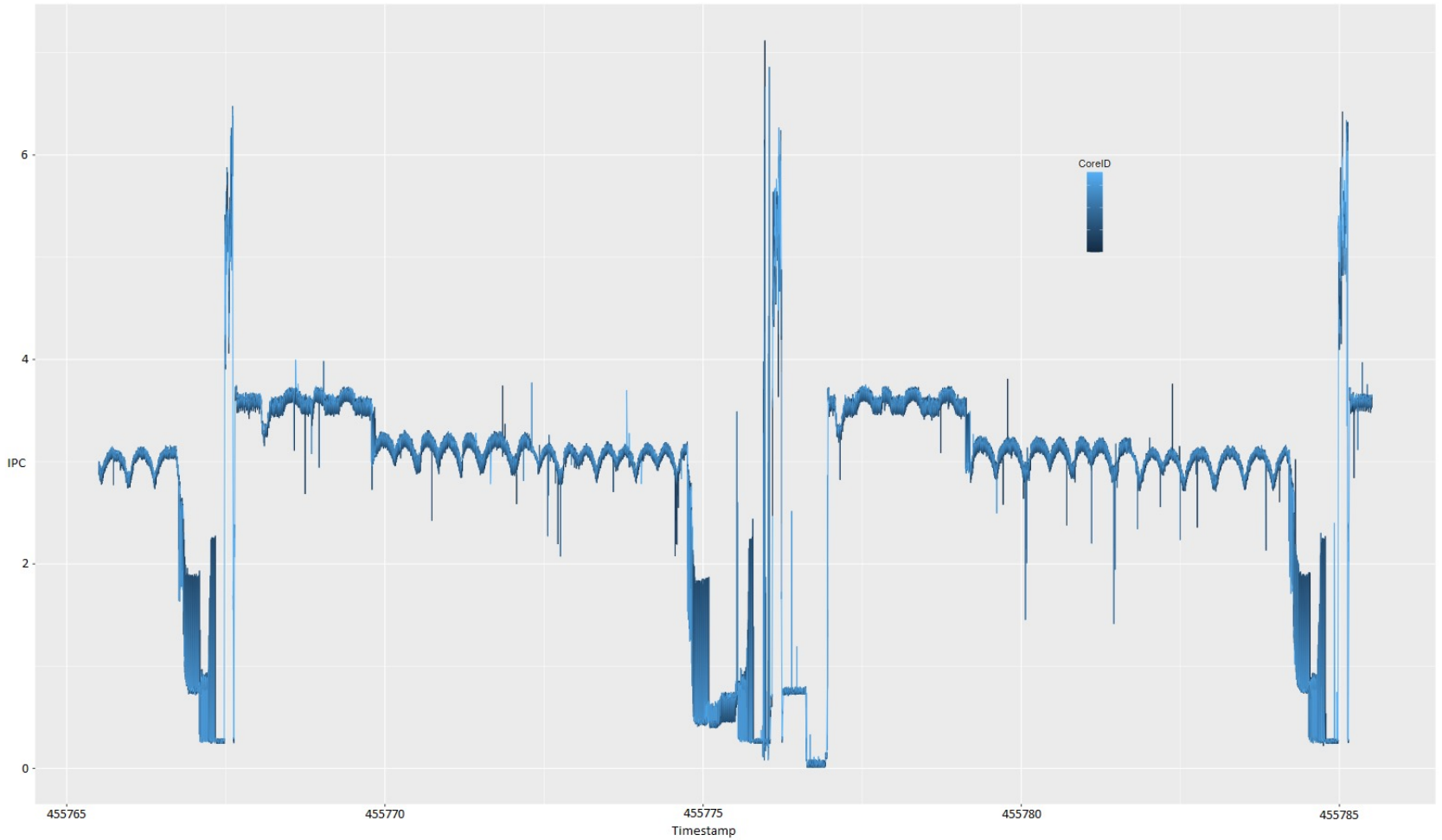
```
↳ user@frontend-hpc $> srun --bdpo=yes -N 180 --ntasks-per-node 96 ./myapp
↳ user@frontend-hpc $> sbatch --bdpo=yes ./script.sbatch
```

```
#!/bin/bash

#SBATCH --job-name=$NAME
#SBATCH --time=1:10:00
#SBATCH --exclusive
#SBATCH --partition=$PARTITION
#SBATCH --nodes=$NODES
#SBATCH --ntasks=$NTASKS
#SBATCH --ntasks-per-node=$NTASKS_PER_NODE
#SBATCH --cpus-per-task=$CPUS_PER_TASK
#SBATCH --bdpo=yes

# Export DPO configuration:
export BDPO_SAMPLING_MS=100
export BDPO_PFM_INSTRUCTIONS_PER_CYCLE_PROFILING=off
export BDPO_PFM_INSTRUCTIONS_PER_CYCLE_TRIGGERING=on
export BDPO_PFM_INSTRUCTIONS_PER_CYCLE_TRIGGERING_THRESHOLD=2.0
export BDPO_PFM_INSTRUCTIONS_PER_CYCLE_THRESHOLD_CROSSING_UP_ACTION=set_cpufreq:3201000
export BDPO_PFM_INSTRUCTIONS_PER_CYCLE_THRESHOLD_CROSSING_DOWN_ACTION=set_cpufreq:2400000
export BDPO_COMM_OPTIMIZATION_ENABLED=off
export BDPO_ENTER_MPI_FREQUENCY_ACTION=set_cpufreq:min
export BDPO_EXIT_MPI_FREQUENCY_ACTION=set_cpufreq:resume
export BDPO_OPTIMIZE_MPI_CLOCK_MODULATION=off
export BDPO_ENTER_MPI_CLOCK_MODULATION_ACTION=set_clock_modulation:87.5
export BDPO_EXIT_MPI_CLOCK_MODULATION_ACTION=set_clock_modulation:max

```

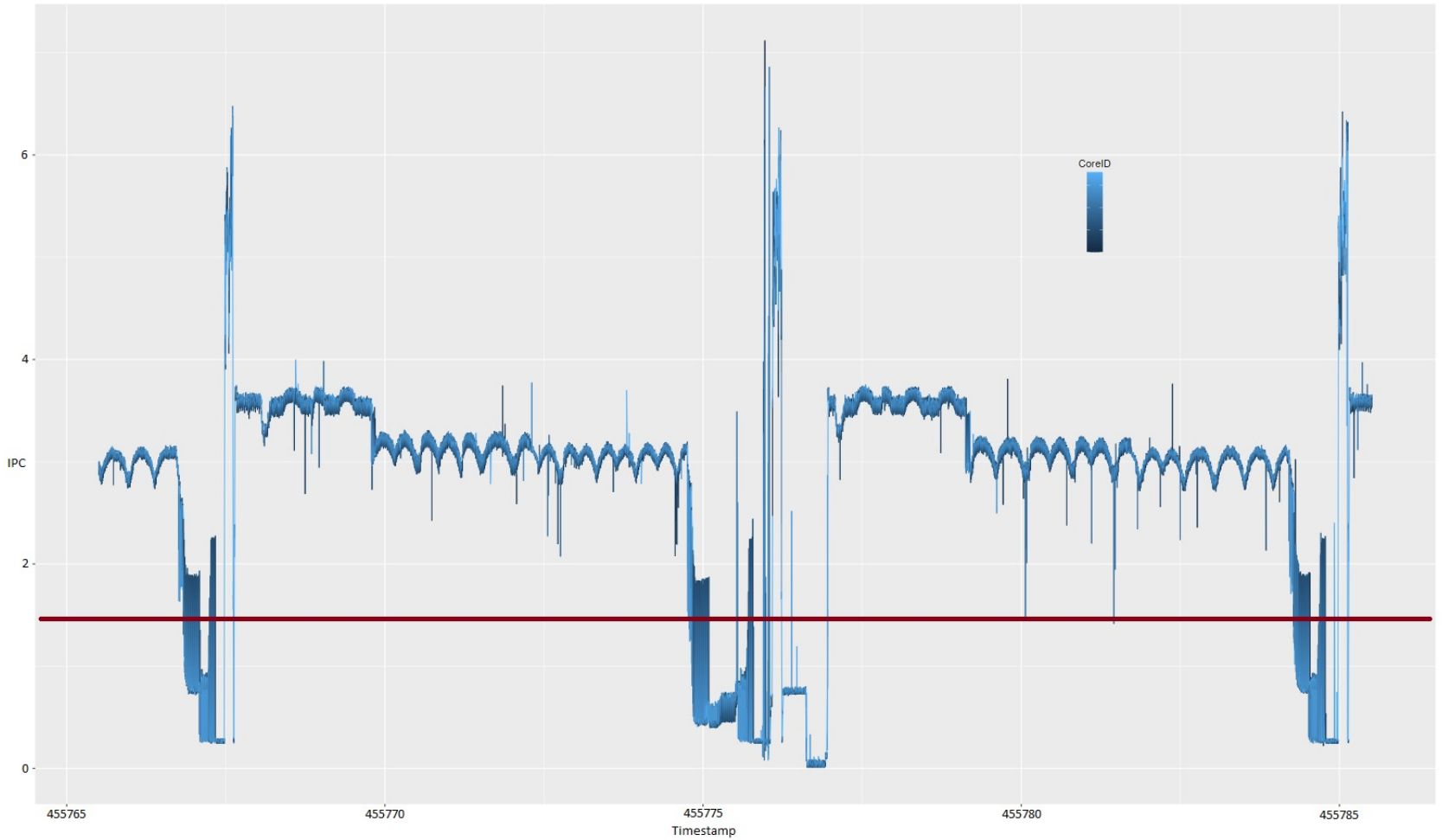
Optimization cycle of BDPO



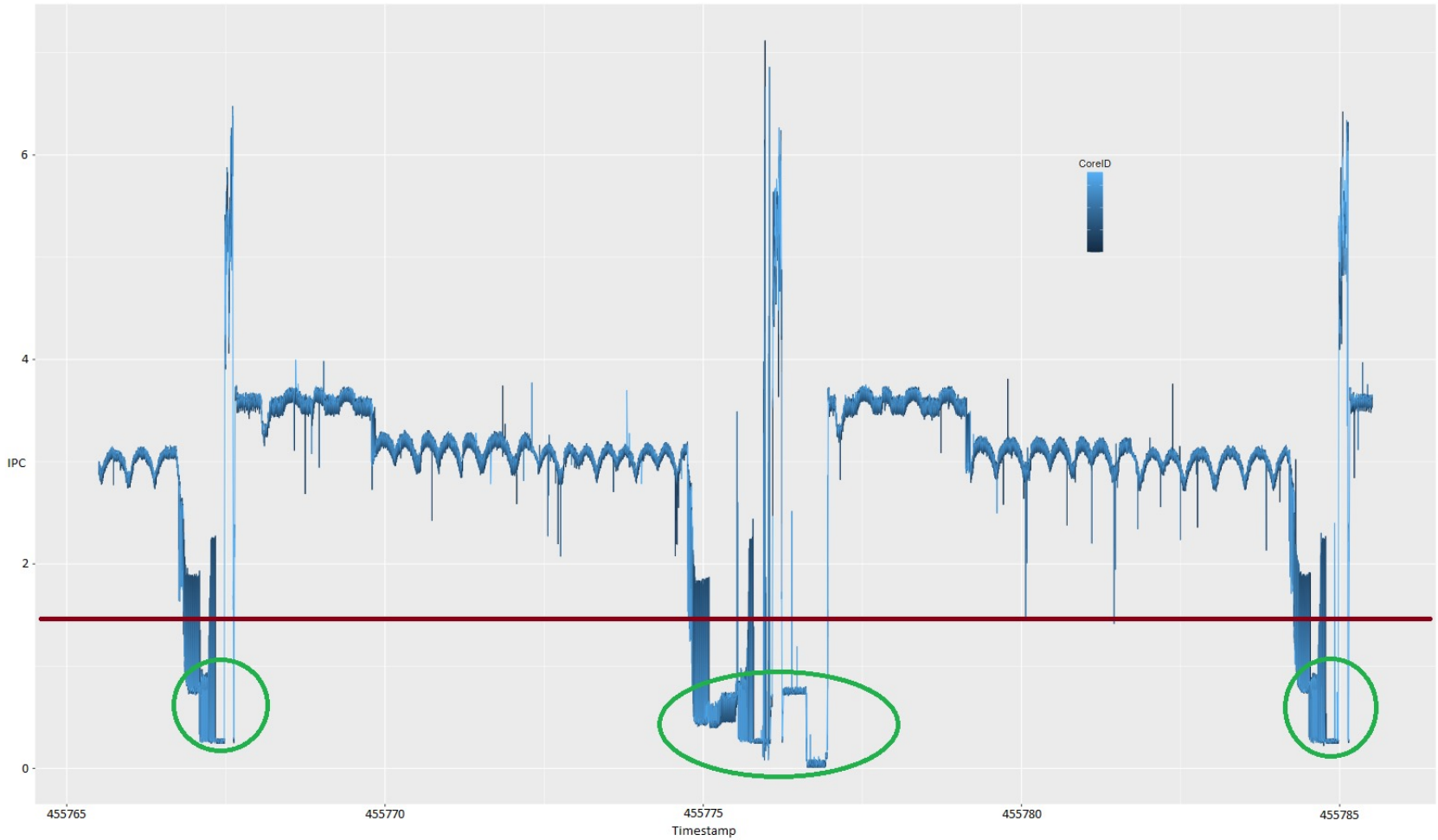
Example of retroaction in BDPO OC



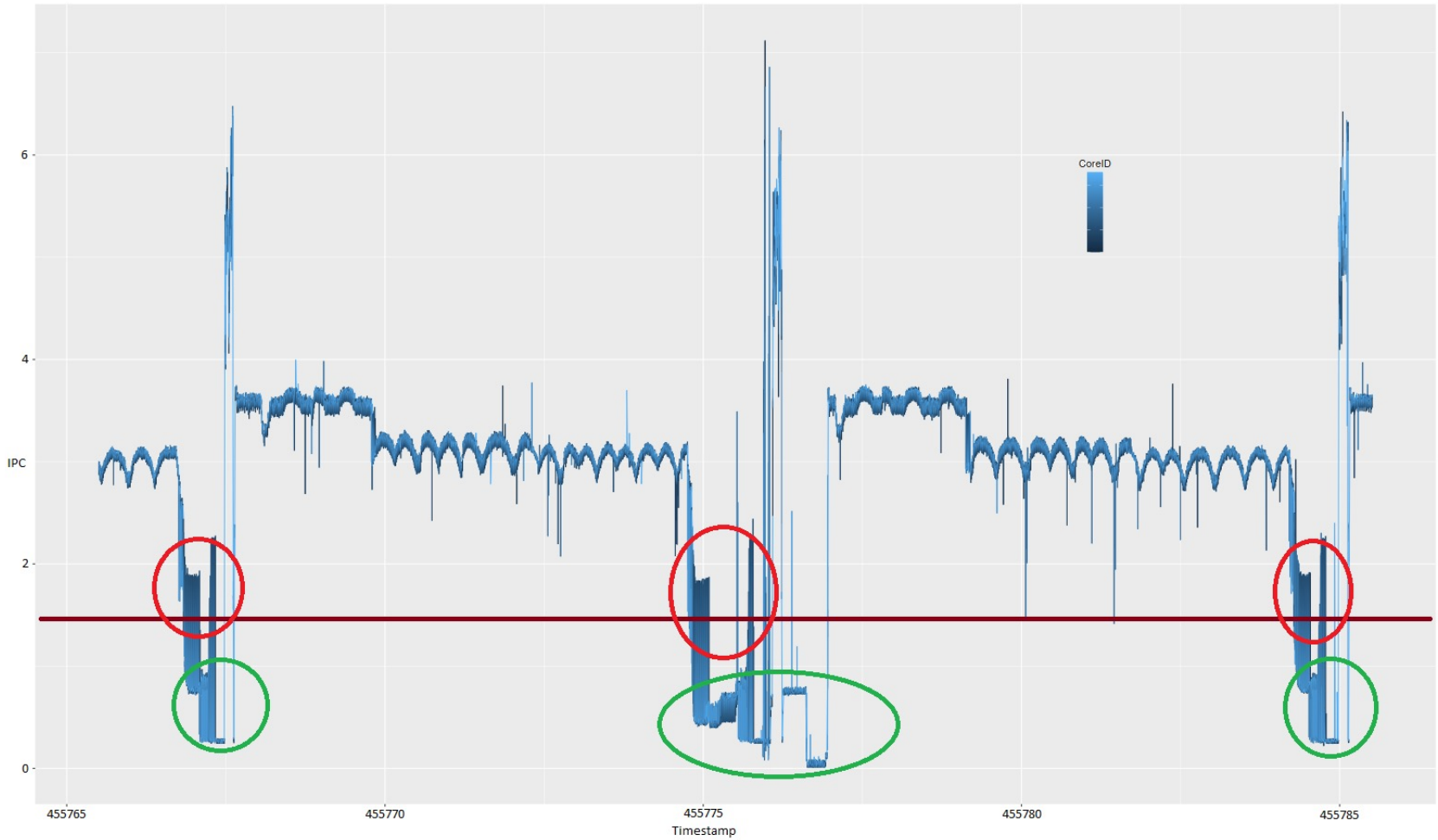
Example of retroaction in BDPO OC



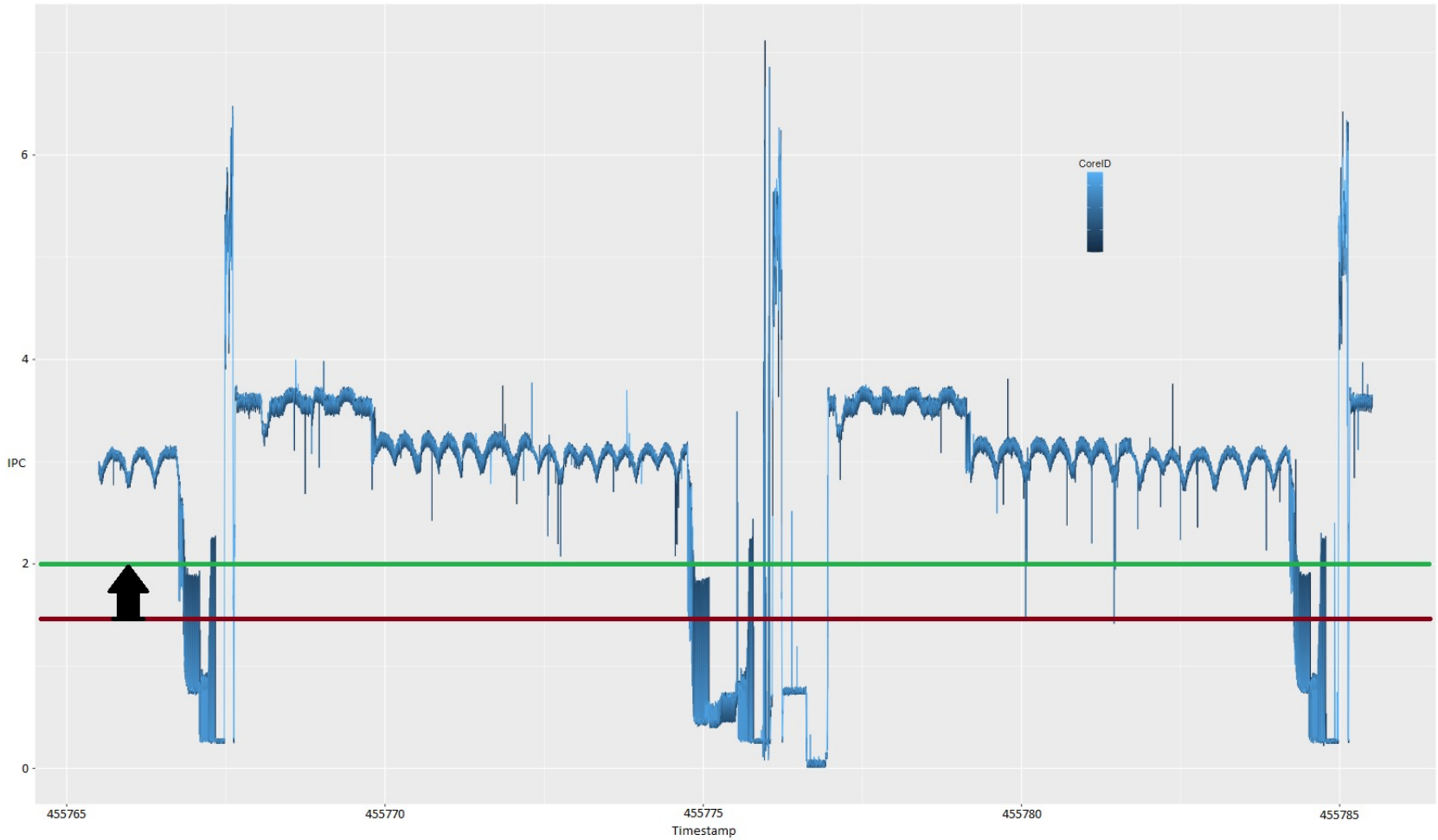
Example of retroaction in BDPO OC



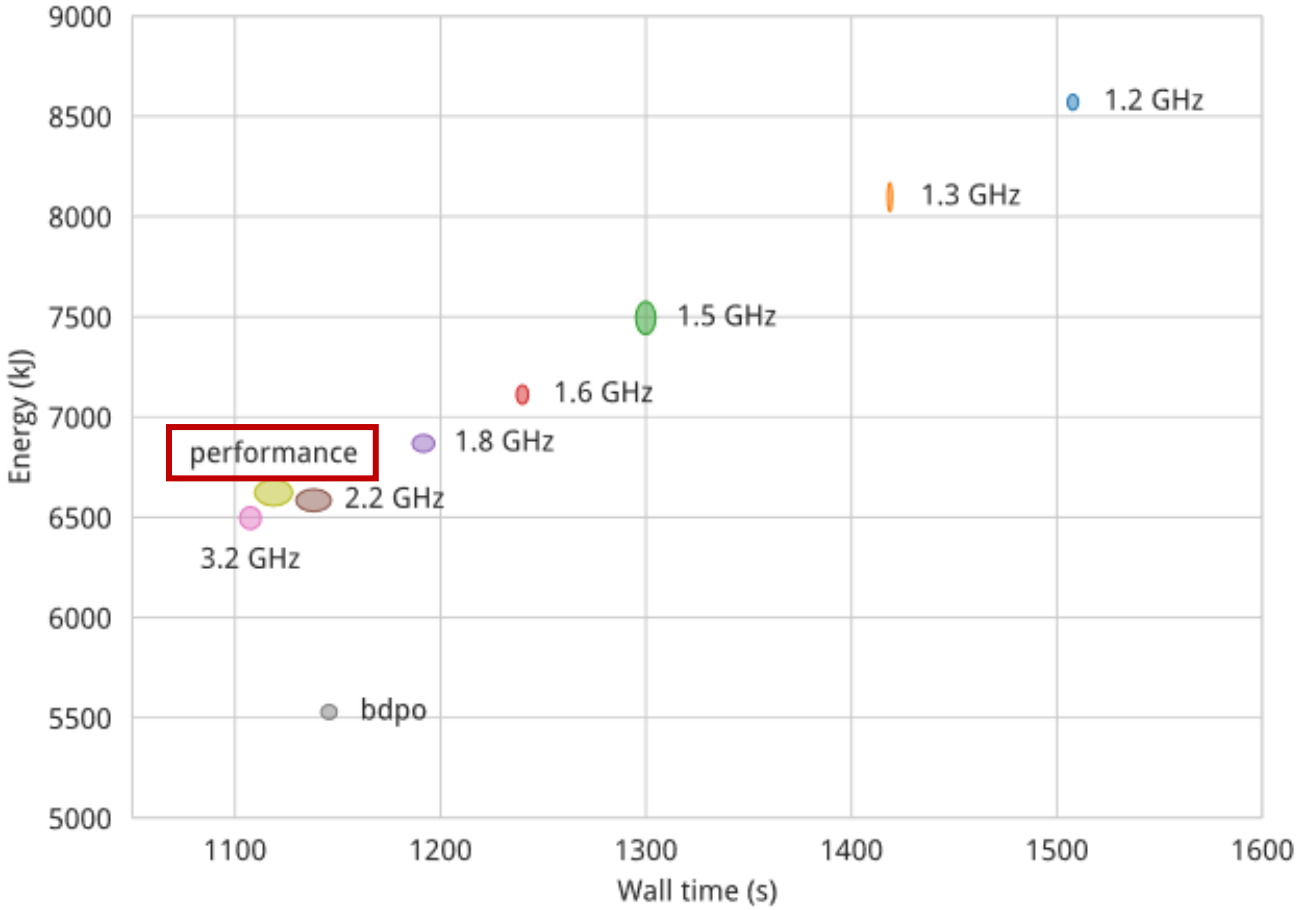
Example of retroaction in BDPO OC



Example of retroaction in BDPO OC



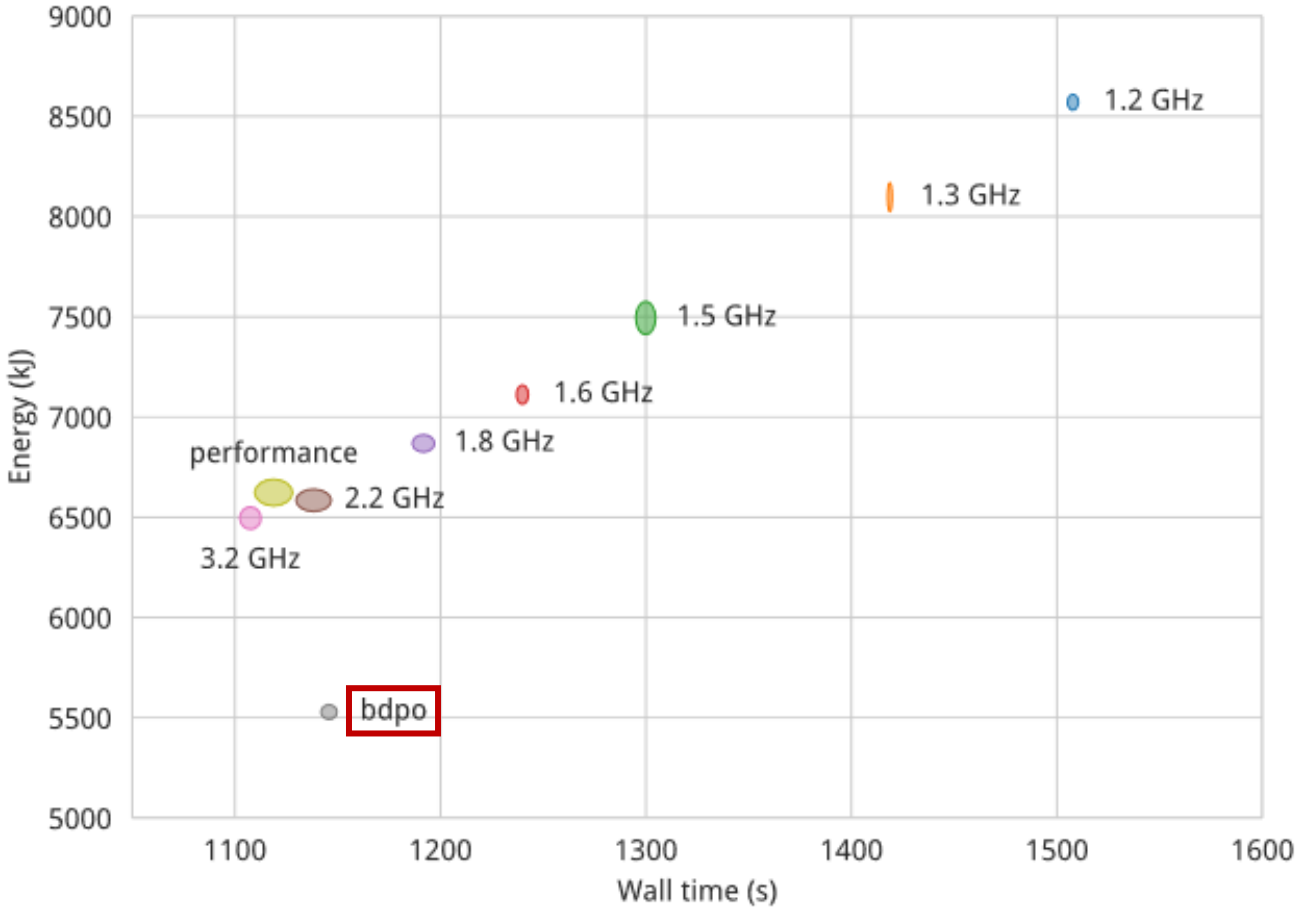
Results with ecTrans (part of the IFS) on DEEP



- Executions on 16 compute nodes of the CN module of DEEP
- Each point stems from statistics on 10 executions
- "performance" represents executions with the performance governor and acts as the reference point

Hatfield S. et. al. Mapping a coupled Earth-System simulator onto the Modular Supercomputing Architecture, 20th ECMWF Workshop on High Performance Computing in Meteorology

Results with ecTrans (part of the IFS) on DEEP



- BDPO:
  - 16% decrease of the energy consumption
  - 3% increase of the execution time

  ➔ better energy-efficiency

- No fixed CPU frequency outperforms BDPO

Hatfield S. et. al. Mapping a coupled Earth-System simulator onto the Modular Supercomputing Architecture, 20th ECMWF Workshop on High Performance Computing in Meteorology

TSMP & BDPO: a few words about co-conception

- TSMP is constituted of three applications: COSMO (modelisation of the atmosphere), ParFlow (modelisation of the ocean), and CLM (coupler between COSMO and ParFlow)

- CLM is a good candidate for BDPO OC. However, the three applications are started in the same Slurm job, on shared nodes, through the multiprog option

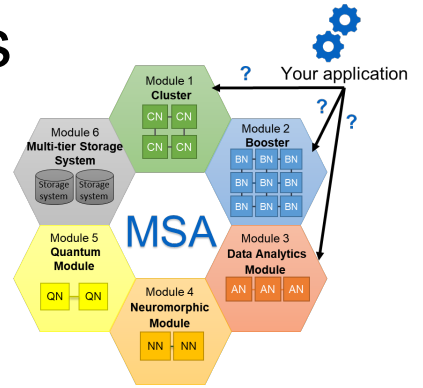
- Co-conception: support of the multiprog option in BDPO, and design of a TSMP job where applications do not share nodes (evaluation phase in progress)

Discussion

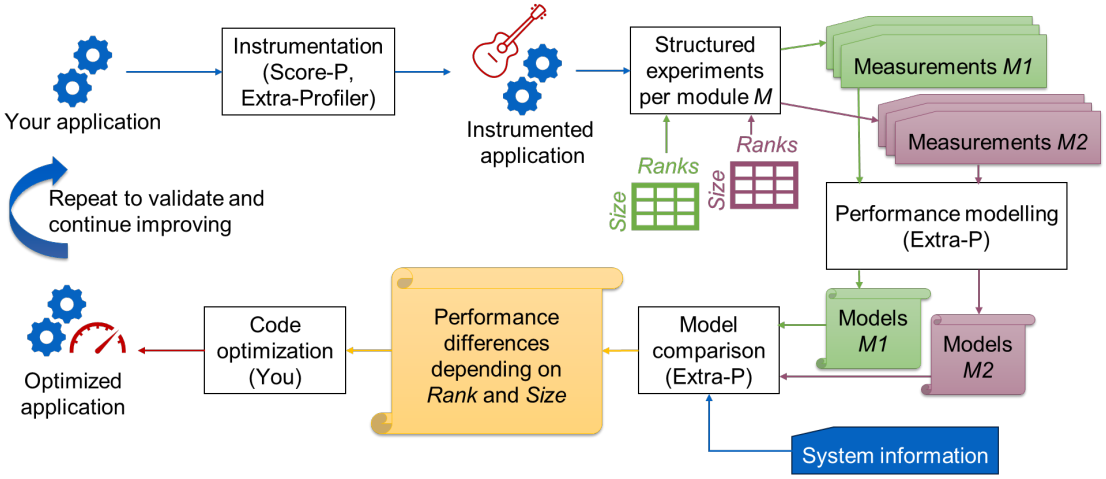
Summary

The Application Mapping OC

- Mapping applications onto modules can cause surprises
  - Also, when offloading on heterogeneous nodes
  - Performance might differ from expectations



- The Application Mapping OC helps you to check if expectations are fulfilled
  - Integrated set of tools
  - Systematic approach
  - Flexible, when needed



Two takeaways about BDPO OC

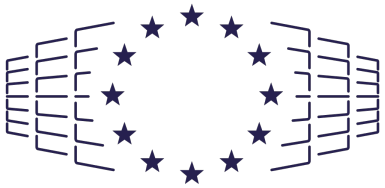
- There are opportunities for optimization of the energy-efficiency of the executions of HPC applications at runtime:
  - ➔ Very good results with ecTrans on 16 nodes:
    - 16% decrease of the energy-to-solution
    - 3% increase of the time-to-solution

- Usability is important:
  - A lot of partners tried BDPO OC on their applications: ecTrans (IFS), TSMP, PATMOS, xPic, GROMACS, …
  - When asked why: BDPO works out-of-the-box and is easy to use

www.deep-projects.eu

@DEEPprojects

@deep-projects



EuroHPC Joint Undertaking

SPONSORED BY THE Federal Ministry of Education and Research

Swedish Research Council

The DEEP Projects have received funding from the European Commission's FP7, H2020, and EuroHPC Programmes, under Grant Agreements n° 287530, 610476, 754304, and 955606.

The EuroHPC Joint Undertaking (JU) receives support from the European Union's Horizon 2020 research and innovation programme and Germany, France, Spain, Greece, Belgium, Sweden, United Kingdom, Switzerland.