TECHNISCHE
UNIVERSITÄT
DARMSTADT

Parallel
Programming

Validating the Performance of GPU-Offloading with Differential Performance Models

Motivation

- When offloading computation to the GPU we expect
  - Things to get faster
  - More energy efficient

- Are these expectations for a specific code fulfilled?
- What about individual application parts?
- What happens when scaling up?

TECHNISCHE UNIVERSITÄT DARMSTADT

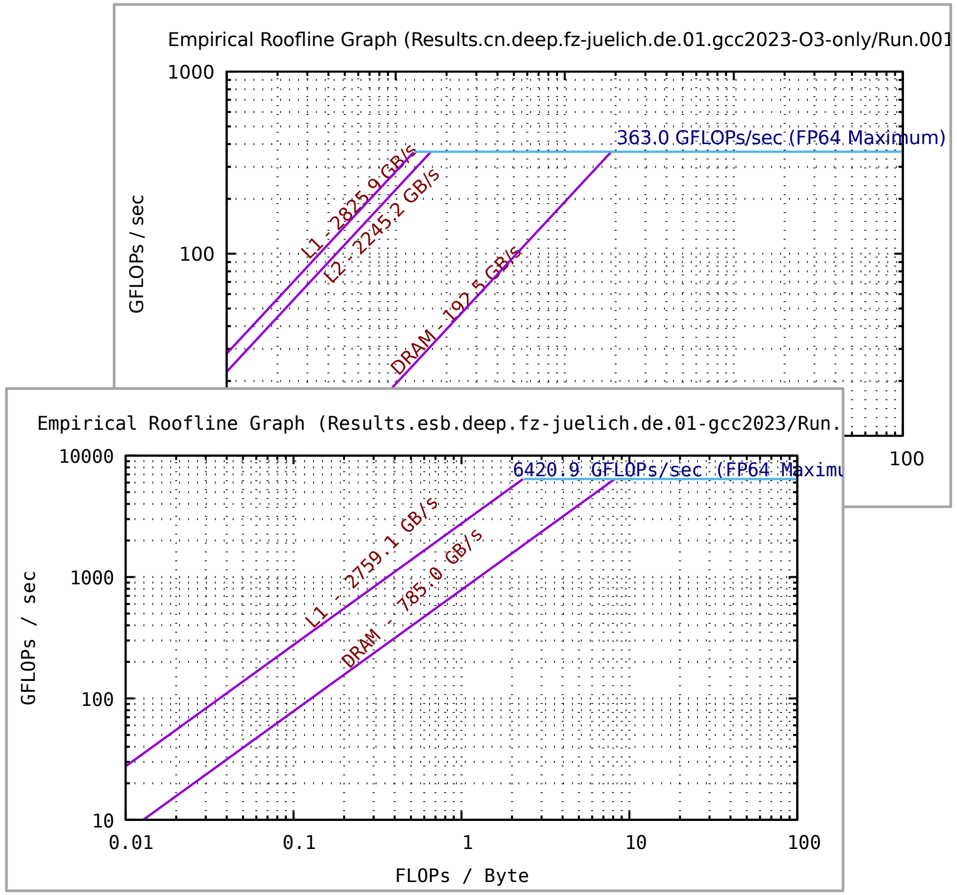
Parallel Programming

Workflow

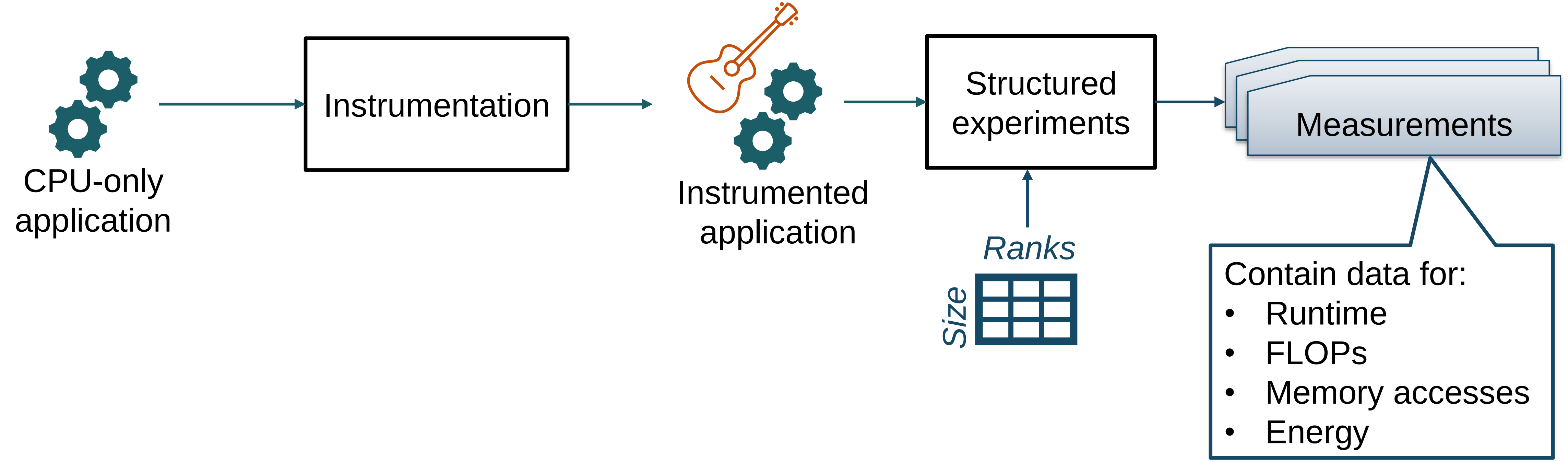
1. Obtain hardware capabilities
2. Collect baseline for the CPU-only version
3. Benchmark the GPU port
4. Generate performance models and explore them

1. Obtaining Hardware Capabilities

- Once per system/hardware
- Determining hardware capabilities
  - We use roofline models
    - Easy to create & understand
  - Generated with empirical roofline toolkit



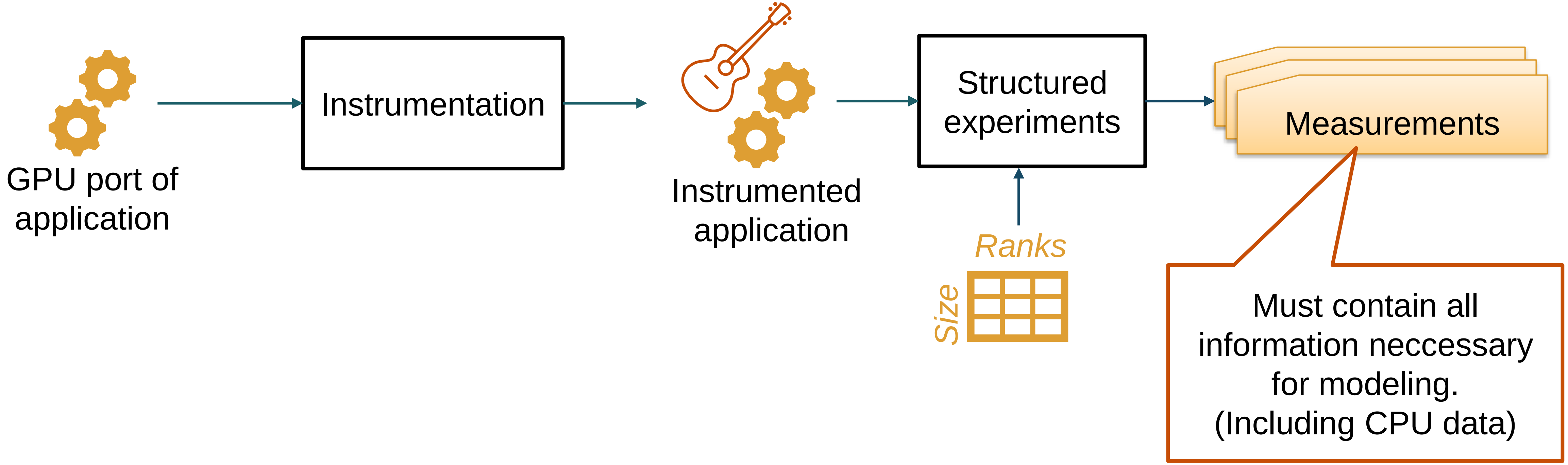
2. Baseline Measurements



Contain data for:

- Runtime
- FLOPs
- Memory accesses
- Energy

3. Benchmark the GPU port



GPU port of application

Instrumentation

Instrumented application

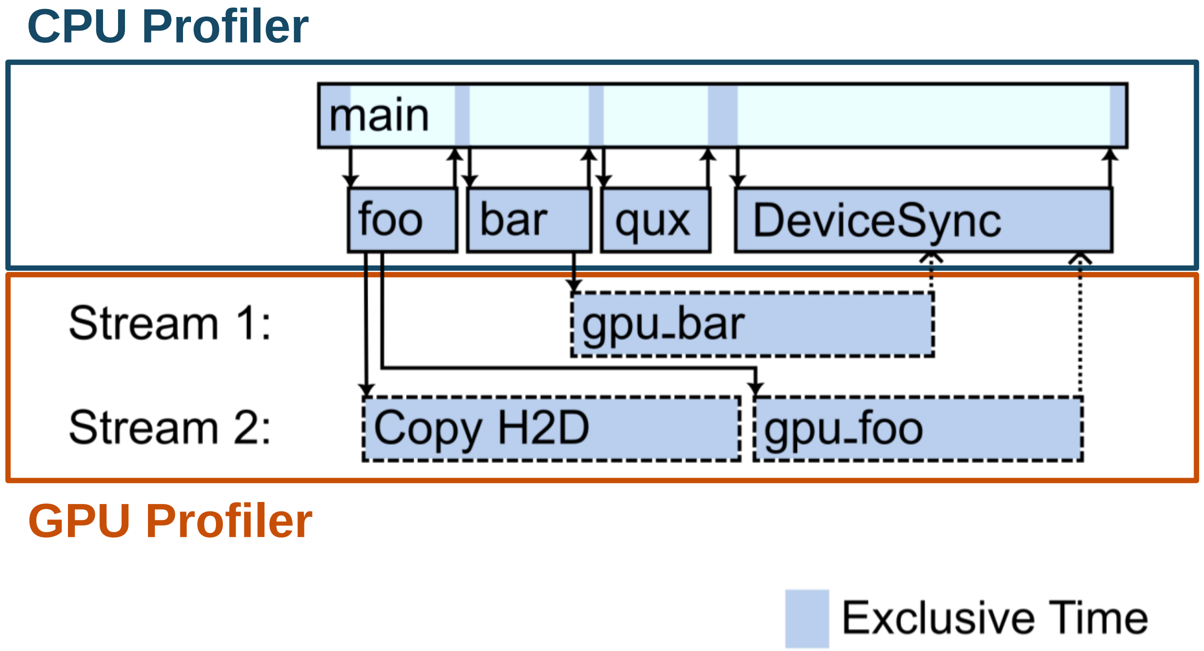
Structured experiments

Measurements

Must contain all information neccessary for modeling. (Including CPU data)

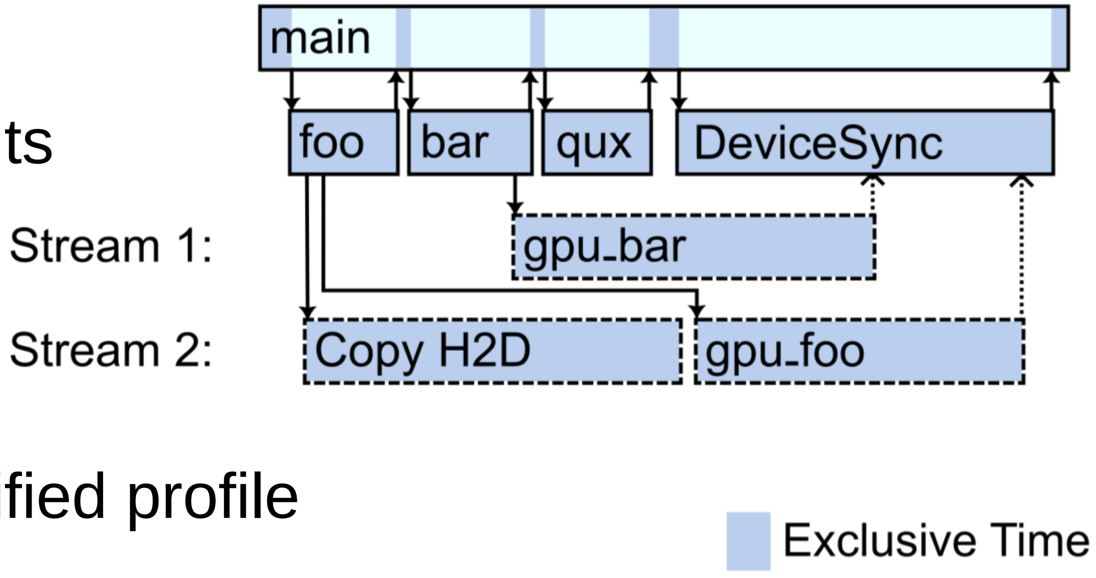
Joint CPU–GPU-Profiling

```
cudaStream_t stream1, stream2;
void foo(double* a_gpu, ...) { ...
  cudaMemcpyAsync(a_gpu, ..., H2D, stream2);
  gpu_foo<<<..., stream2>>>(a_gpu, ...); ...
}

void bar(...) { ...
  gpu_bar<<<..., stream1>>>(...); ...
}

int main() { ...
  foo(a_gpu, ...); ...
  bar(...); ...
  qux(...); ...
  cudaDeviceSynchronize(); ...
}
```



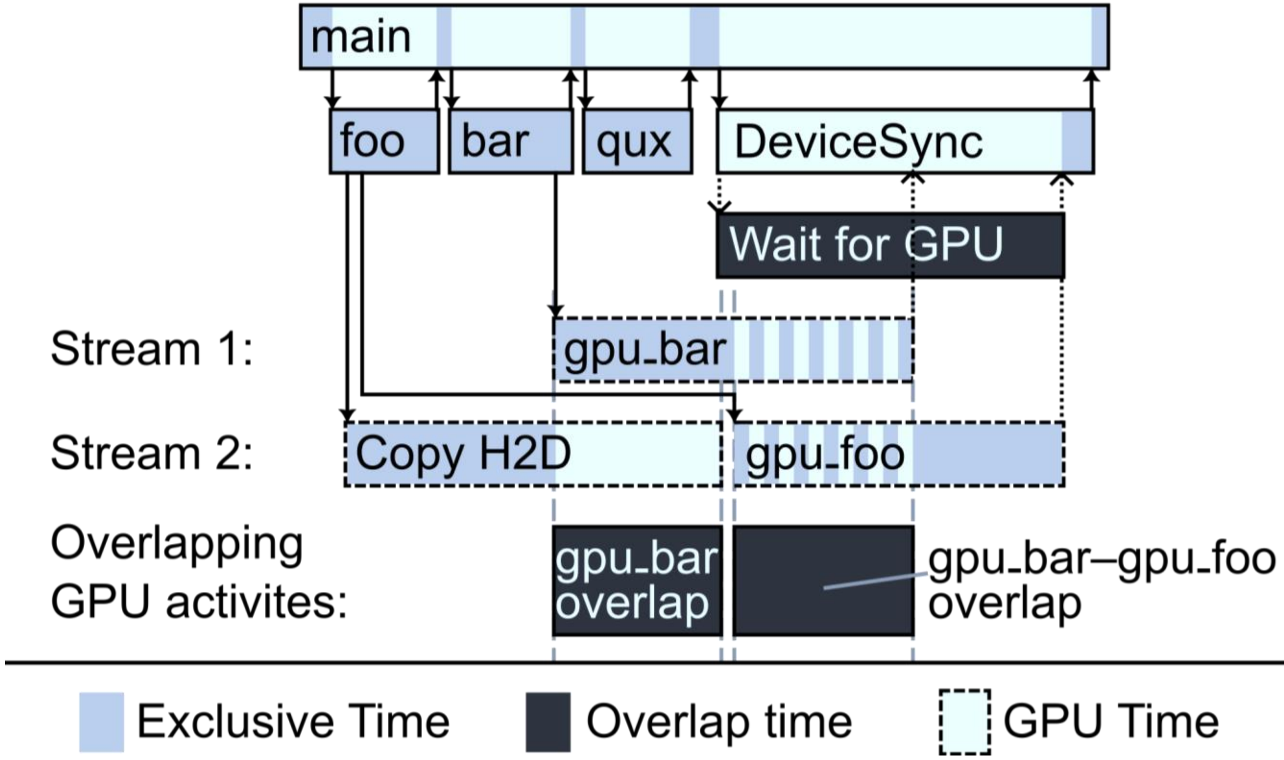
Joint CPU–GPU-Profiling II

- We collect timestamps on start and end of each function and kernel
  - Using compiler instrumentation on CPU
  - CUPTI Callback API for GPU
- We profile the CPU and trace the GPU events
  - We store only limited trace data
  - Calling context is tracked
  - During post-processing we convert to a unified profile

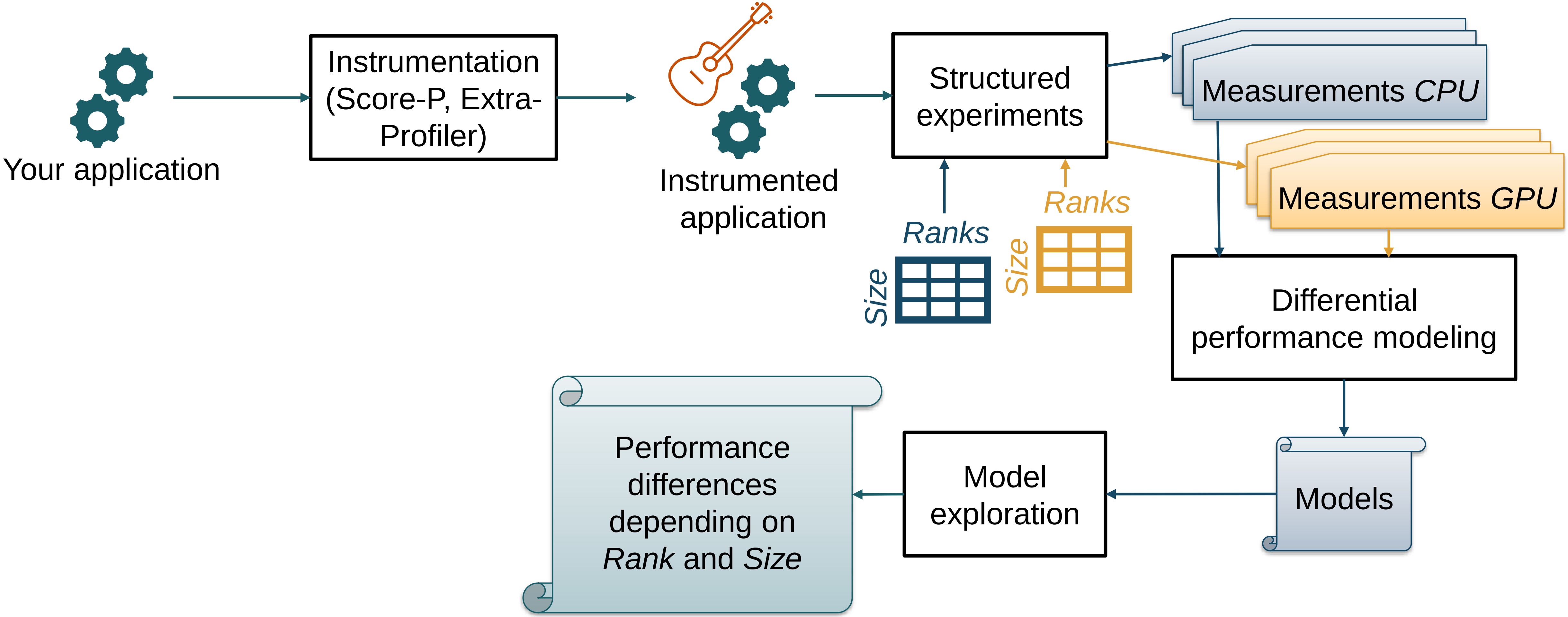


Joint CPU–GPU-Profiling III

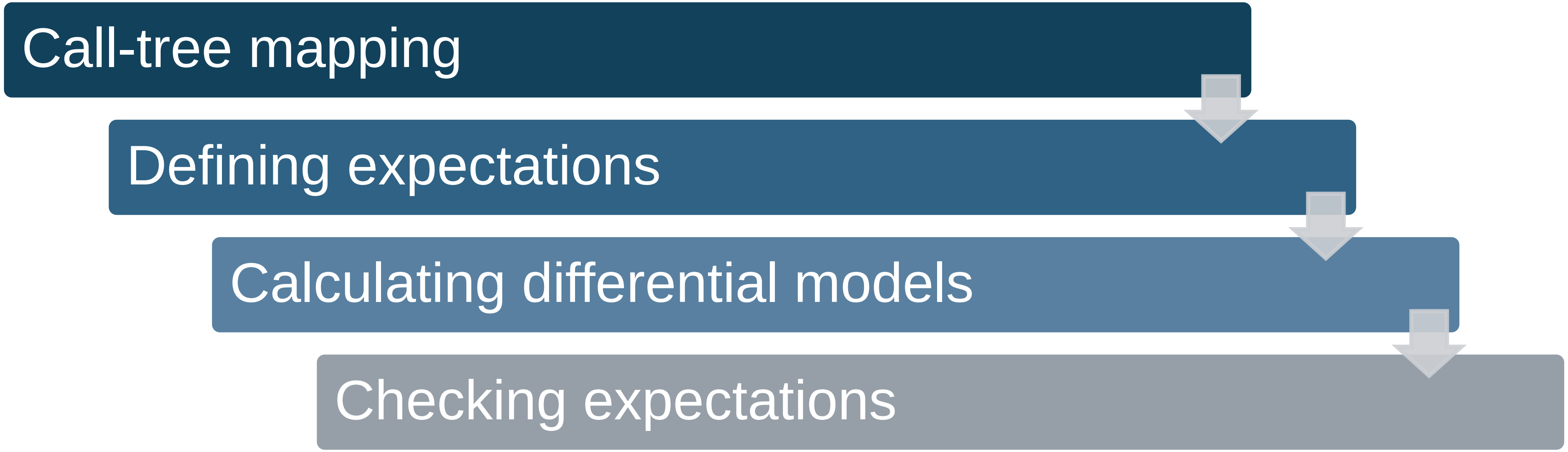
- This allows recording of
  - Synchronization of GPU actions with CPU functions
  - Overlap of concurrent GPU activities
- Small result files with data necessary for modeling
- Full call path up to kernel



4. Differential Performance Modeling



4. Differential performance modeling



Call-tree mapping

Defining expectations

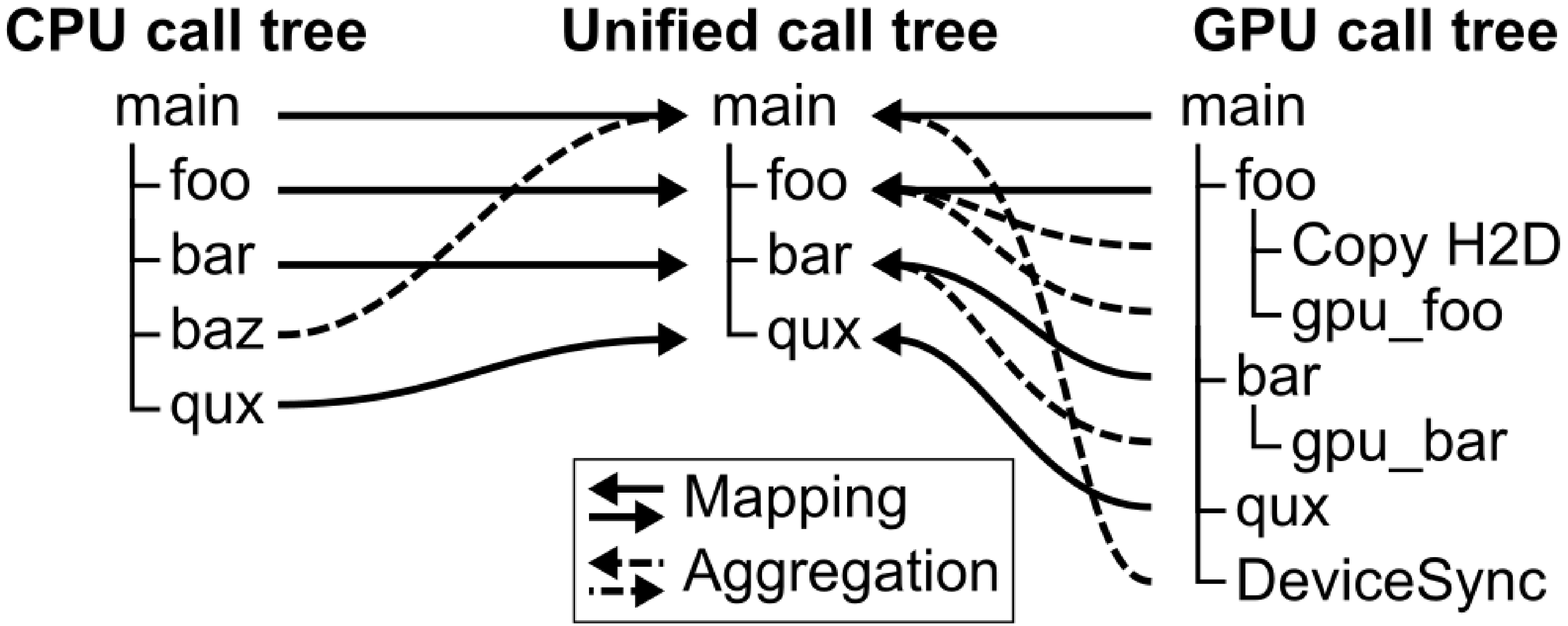
Calculating differential models

Checking expectations

Call-Tree Mapping

- Prefix matching with aggregation

Remember we are using exclusive values

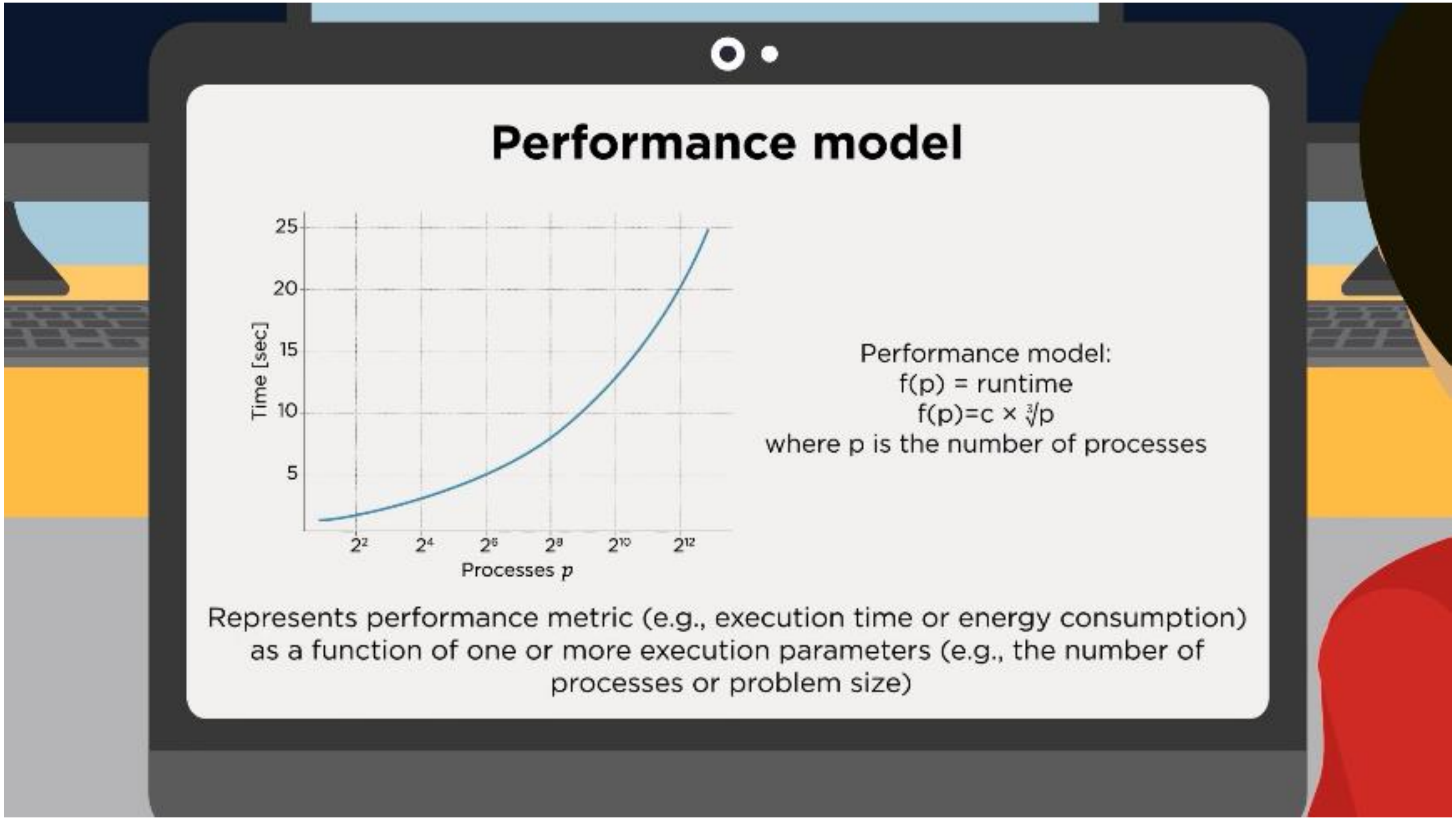


Defining Expectations

Faster than CPU-only implementation	• Runtime of GPU port is lower than runtime of CPU-only version
Uses less energy	• Energy usage of GPU port is lower than runtime of CPU-only version
Uses the hardware well	• Achieves same or higher hardware efficiency

Refresher

Performance Modeling with Extra-P



Performance model

Performance model:
$f(p)$ = runtime
$f(p)=c \times \sqrt[3]{p}$
where p is the number of processes

Represents performance metric (e.g., execution time or energy consumption) as a function of one or more execution parameters (e.g., the number of processes or problem size)

Watch Extra-P overview video

https://www.youtube.com/watch?v=Cv2YRCMWqBM

Refresher

Automatic Performance Modeling

TECHNISCHE UNIVERSITÄT DARMSTADT

Parallel Programming

```
main() {
    foo()
    bar()
    compute()
}
```

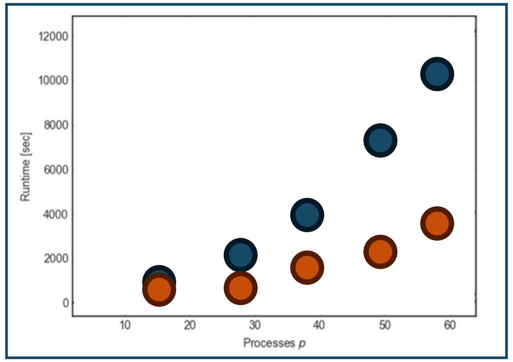
Input

Instrumentation

- All functions

Performance measurements

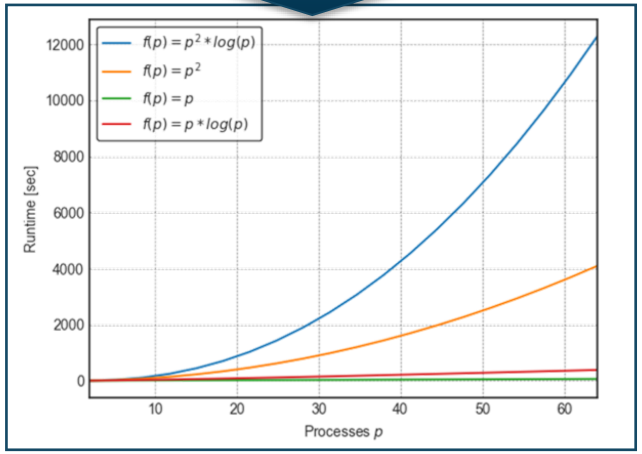
M_i M_j



Extra-P

Output

Human-readable performance models of all functions (e.g., $t(p) = c_1 \cdot \log(p) + c_2$)



Performance Model Normal Form

$$f(x) = \sum_{k=1}^{n} c_k \cdot x^{i_k} \cdot log_2^{j_k}(x)$$

$n \in \mathbb{N}$
$i_k \in I$
$j_k \in J$
$I, J \subset \mathbb{Q}$

$n = 1$
$I = \{0, 1, 2\}$
$J = \{0, 1\}$

c_1	$c_1 \cdot \log x$
$c_1 \cdot x$	$c_1 \cdot x \cdot \log x$
$c_1 \cdot x^2$	$c_1 \cdot x^2 \cdot \log x$

Calculating Differential Models

- All models f_c^{GPU} and f_c^{CPU} are mathematical expressions
  - We can calculate with them

Differential models express the difference

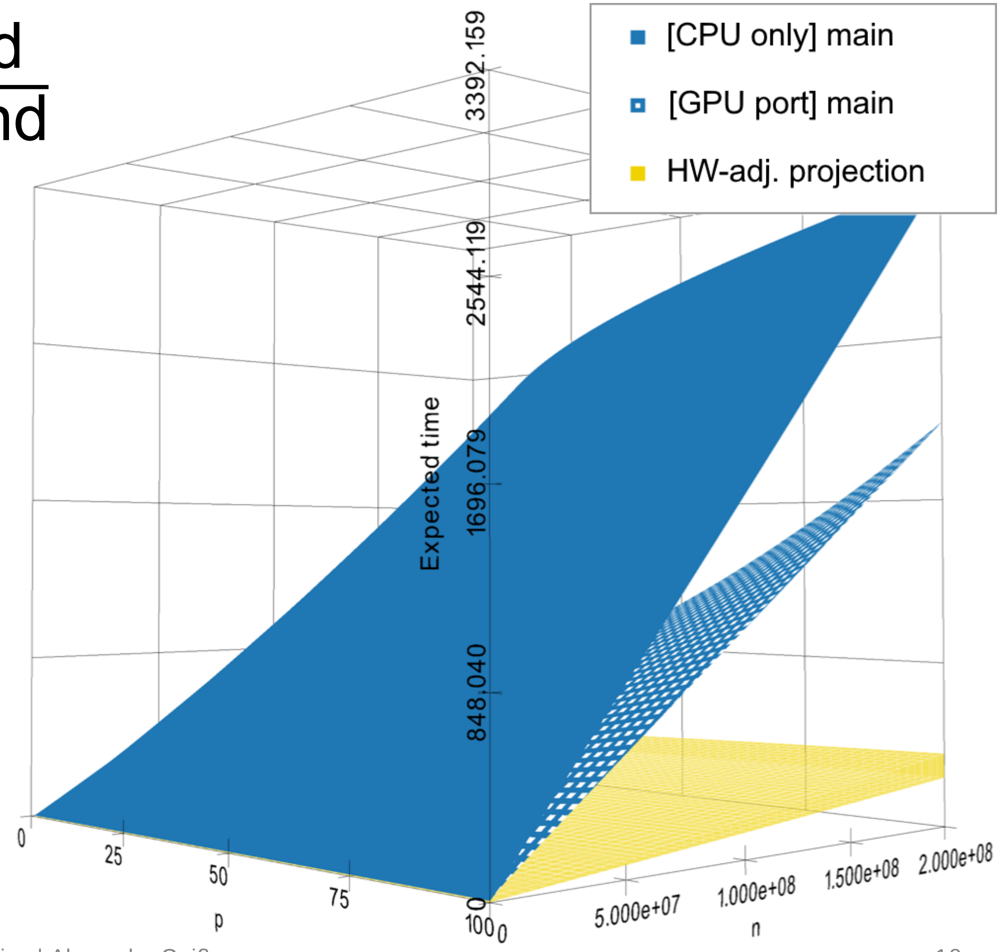
- $\Delta_c(p_1, \dots) = f_c^{\mathrm{GPU}}(p_1, \dots) - f_c^{\mathrm{CPU}}(p_1, \dots)$
  - For a specific call tree entry c

Hardware Efficiency

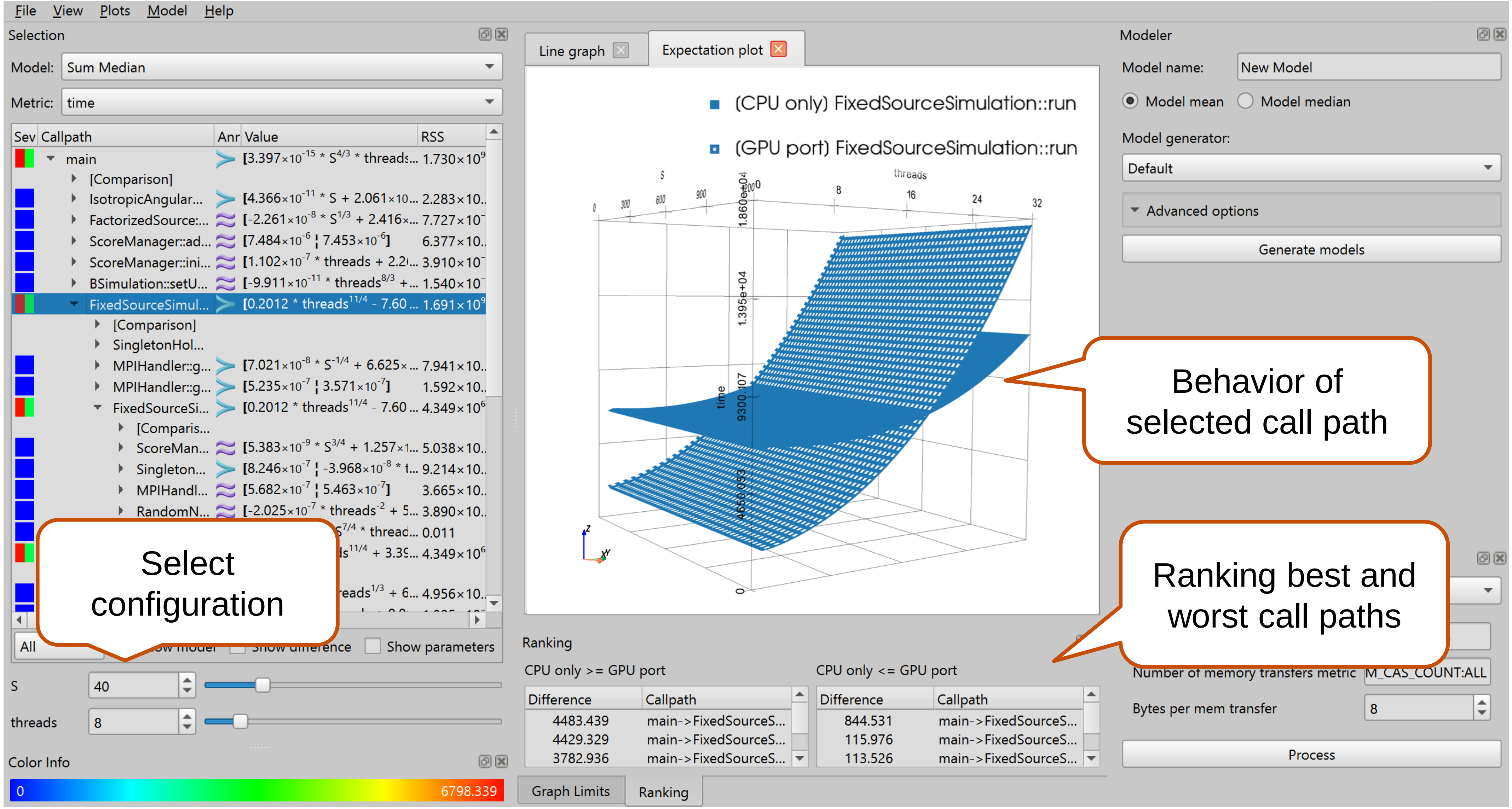
$$\text{Hardware efficiency} = \frac{\text{achieved FLOPs per second}}{\text{achievable FLOPs per second}}$$

- Modeled with Extra-P
  - We build models for FLOPs, time, and memory accesses from measurements
    - All models are mathematical expressions

- We present this as a hardware adjusted runtime model to the user



Interactive Exploration



Model: Sum Median

Metric: time

Sev	Callpath	Anr	Value	RSS
🟥🟩	main	≻	[3.397×10^{-15} * S$^{4/3}$ * threads...	1.730×10^{9}
	[Comparison]			
🟦	IsotropicAngular...	≻	[4.366×10^{-11} * S + 2.061×10...	2.283×10...
🟦	FactorizedSource:...	≈	[-2.261×10^{-8} * S$^{1/3}$ + 2.416×...	7.727×10
🟦	ScoreManager::ad...	≈	[7.484×10^{-6} ¦ 7.453×10^{-6}]	6.377×10...
🟦	ScoreManager::ini...	≈	[1.102×10^{-7} * threads + 2.2...	3.910×10
🟦	BSimulation::setU...	≈	[-9.911×10^{-11} * threads$^{8/3}$ +...	1.540×10
🟥🟩	FixedSourceSimul...	≻	[0.2012 * threads$^{11/4}$ - 7.60...	1.691×10^{9}
	[Comparison]			
	SingletonHol...			
🟦	MPIHandler::g...	≻	[7.021×10^{-8} * S$^{-1/4}$ + 6.625×...	7.941×10...
🟦	MPIHandler::g...	≻	[5.235×10^{-7} ¦ 3.571×10^{-7}]	1.592×10...
🟥🟩	FixedSourceSi...	≻	[0.2012 * threads$^{11/4}$ - 7.60...	4.349×10^{6}
	[Comparis...			
🟦	ScoreMan...	≈	[5.383×10^{-9} * S$^{3/4}$ + 1.257×1...	5.038×10...
🟦	Singleton...	≻	[8.246×10^{-7} ¦ -3.968×10^{-8} * t...	9.214×10...
🟦	MPIHandl...	≈	[5.682×10^{-7} ¦ 5.463×10^{-7}]	3.665×10...
🟦	RandomN...	≈	[-2.025×10^{-7} * threads^{-2} + 5...	3.890×10...
🟦			S$^{7/4}$ * thread...	0.011
🟥🟩			ds$^{11/4}$ + 3.39...	4.349×10^{6}
🟦			reads$^{1/3}$ + 6...	4.956×10...

All □ Show difference □ Show parameters

S 40

threads 8

■ (CPU only) FixedSourceSimulation::run

■ (GPU port) FixedSourceSimulation::run

Model name: New Model

● Model mean ○ Model median

Model generator: Default

Advanced options

Generate models

Number of memory transfers metric M_CAS_COUNT:ALL

Bytes per mem transfer 8

Process

Ranking

CPU only >= GPU port

Difference	Callpath
4483.439	main->FixedSourceS...
4429.329	main->FixedSourceS...
3782.936	main->FixedSourceS...

CPU only <= GPU port

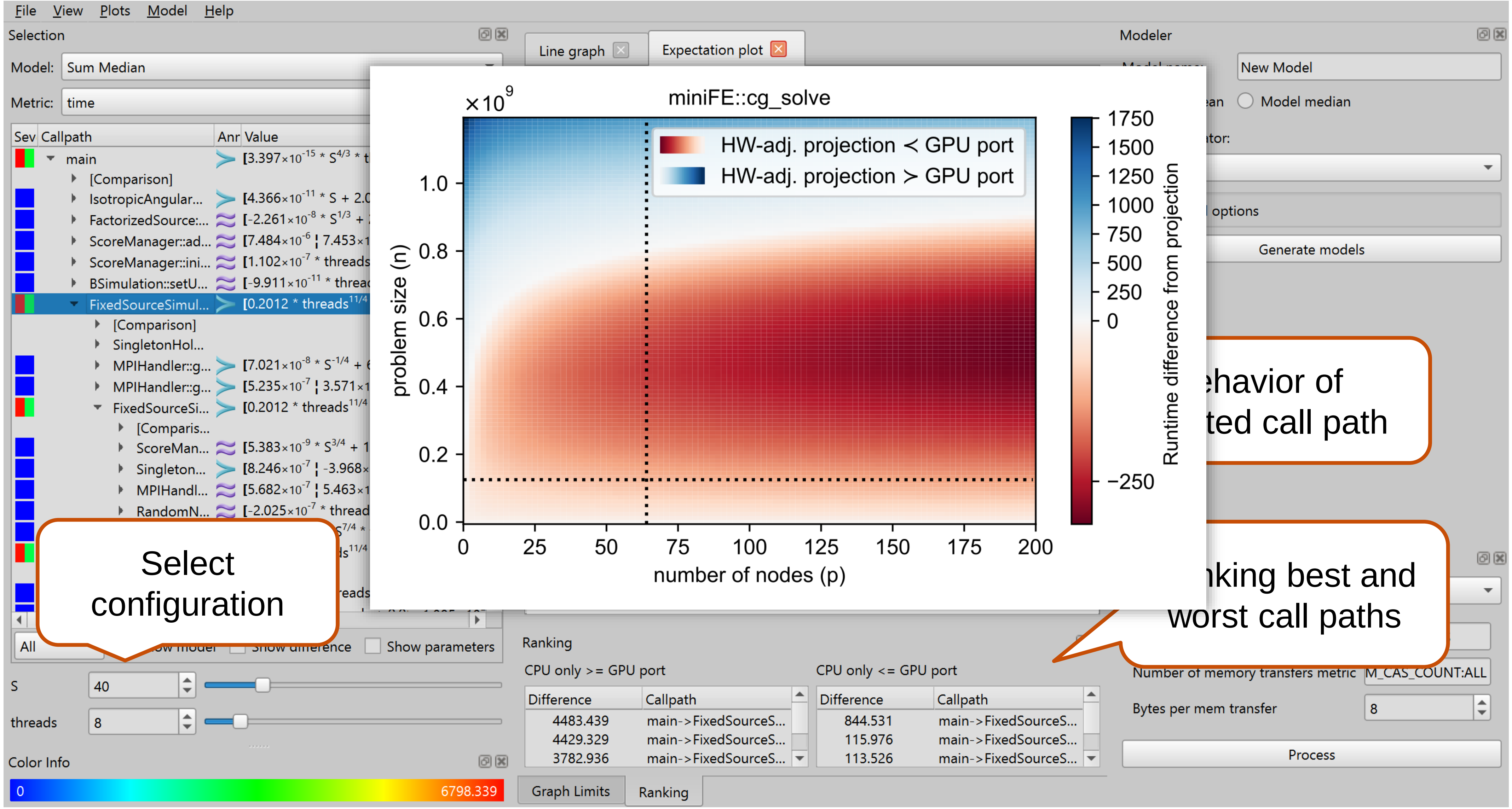
Difference	Callpath
844.531	main->FixedSourceS...
115.976	main->FixedSourceS...
113.526	main->FixedSourceS...

Behavior of selected call path

Select configuration

Ranking best and worst call paths

Interactive Exploration



Model: Sum Median

Metric: time

Sev Callpath Anr Value

- main [$3.397{\times}10^{-15}$ * $S^{4/3}$ * t
  - [Comparison]
  - IsotropicAngular... [$4.366{\times}10^{-11}$ * S + 2.0
  - FactorizedSource:... [$-2.261{\times}10^{-8}$ * $S^{1/3}$ +
  - ScoreManager::ad... [$7.484{\times}10^{-6}$ ¦ 7.453×1
  - ScoreManager::ini... [$1.102{\times}10^{-7}$ * threads
  - BSimulation::setU... [$-9.911{\times}10^{-11}$ * threac
  - FixedSourceSimul... [0.2012 * $\text{threads}^{11/4}$
    - [Comparison]
    - SingletonHol...
    - MPIHandler::g... [$7.021{\times}10^{-8}$ * $S^{-1/4}$ + 6
    - MPIHandler::g... [$5.235{\times}10^{-7}$ ¦ 3.571×1
    - FixedSourceSi... [0.2012 * $\text{threads}^{11/4}$
      - [Comparis...
      - ScoreMan... [$5.383{\times}10^{-9}$ * $S^{3/4}$ + 1
      - Singleton... [$8.246{\times}10^{-7}$ ¦ -3.968×
      - MPIHandl... [$5.682{\times}10^{-7}$ ¦ 5.463×1
      - RandomN... [$-2.025{\times}10^{-7}$ * thread

All ☐ Show difference ☐ Show parameters

S 40

threads 8

Model name: New Model

○ Model median

options

Number of memory transfers metric M_CAS_COUNT:ALL

Bytes per mem transfer 8

Ranking

CPU only >= GPU port

Difference	Callpath
4483.439	main->FixedSourceS...
4429.329	main->FixedSourceS...
3782.936	main->FixedSourceS...

CPU only <= GPU port

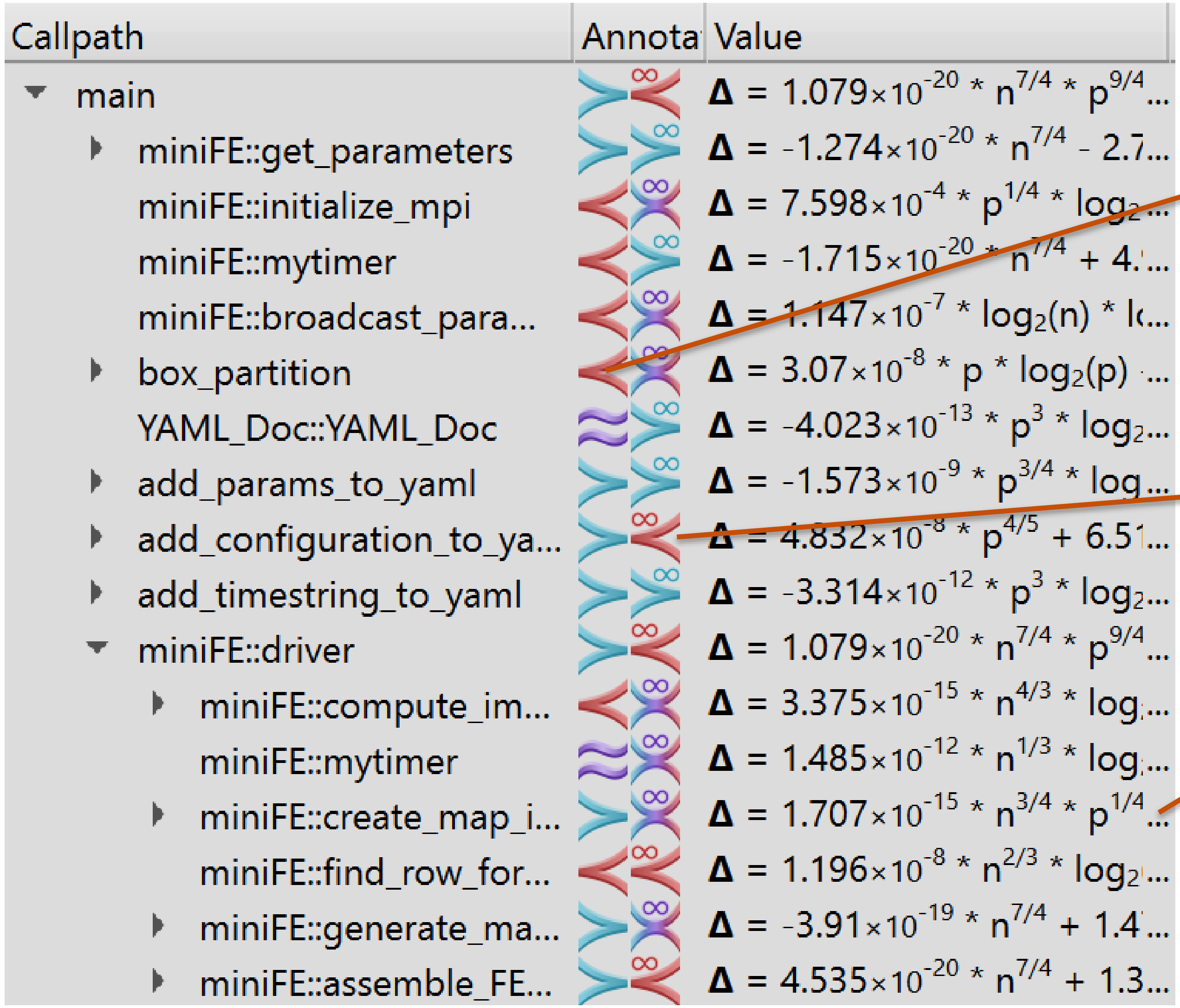
Difference	Callpath
844.531	main->FixedSourceS...
115.976	main->FixedSourceS...
113.526	main->FixedSourceS...

Select configuration

ehavior of ted call path

nking best and worst call paths

Interactive Exploration



Callpath	Annota	Value
▾ main	≻ ∞≺	$\Delta = 1.079\times10^{-20} * n^{7/4} * p^{9/4}$...
▸ miniFE::get_parameters	≻ ≻∞	$\Delta = -1.274\times10^{-20} * n^{7/4} - 2.7$...
miniFE::initialize_mpi	≺ ≻≺∞	$\Delta = 7.598\times10^{-4} * p^{1/4} * \log_2$...
miniFE::mytimer	≺ ≻∞	$\Delta = -1.715\times10^{-20} * n^{7/4} + 4.$...
miniFE::broadcast_para...	≺ ≻≺∞	$\Delta = 1.147\times10^{-7} * \log_2(n) * l$...
▸ box_partition	≺ ≻≺∞	$\Delta = 3.07\times10^{-8} * p * \log_2(p)$ ...
YAML_Doc::YAML_Doc	≈ ≻∞	$\Delta = -4.023\times10^{-13} * p^{3} * \log_2$...
▸ add_params_to_yaml	≻ ≻∞	$\Delta = -1.573\times10^{-9} * p^{3/4} * \log$...
▸ add_configuration_to_ya...	≻ ∞≺	$\Delta = 4.832\times10^{-8} * p^{4/5} + 6.51$...
▸ add_timestring_to_yaml	≻ ≻∞	$\Delta = -3.314\times10^{-12} * p^{3} * \log_2$...
▾ miniFE::driver	≻ ∞≺	$\Delta = 1.079\times10^{-20} * n^{7/4} * p^{9/4}$...
▸ miniFE::compute_im...	≺ ≻≺∞	$\Delta = 3.375\times10^{-15} * n^{4/3} * \log$...
miniFE::mytimer	≈ ≻≺∞	$\Delta = 1.485\times10^{-12} * n^{1/3} * \log$...
▸ miniFE::create_map_i...	≻ ≻≺∞	$\Delta = 1.707\times10^{-15} * n^{3/4} * p^{1/4}$...
miniFE::find_row_for...	≺ ∞≺	$\Delta = 1.196\times10^{-8} * n^{2/3} * \log_2$...
▸ miniFE::generate_ma...	≻ ≻≺∞	$\Delta = -3.91\times10^{-19} * n^{7/4} + 1.4$...
▸ miniFE::assemble_FE...	≻ ∞≺	$\Delta = 4.535\times10^{-20} * n^{7/4} + 1.3$...

Result: GPU port...

≺ ... is slower than expected

≻ ... is faster than expected

≈ ... meets expectation

Result: GPU port is asymptotically...

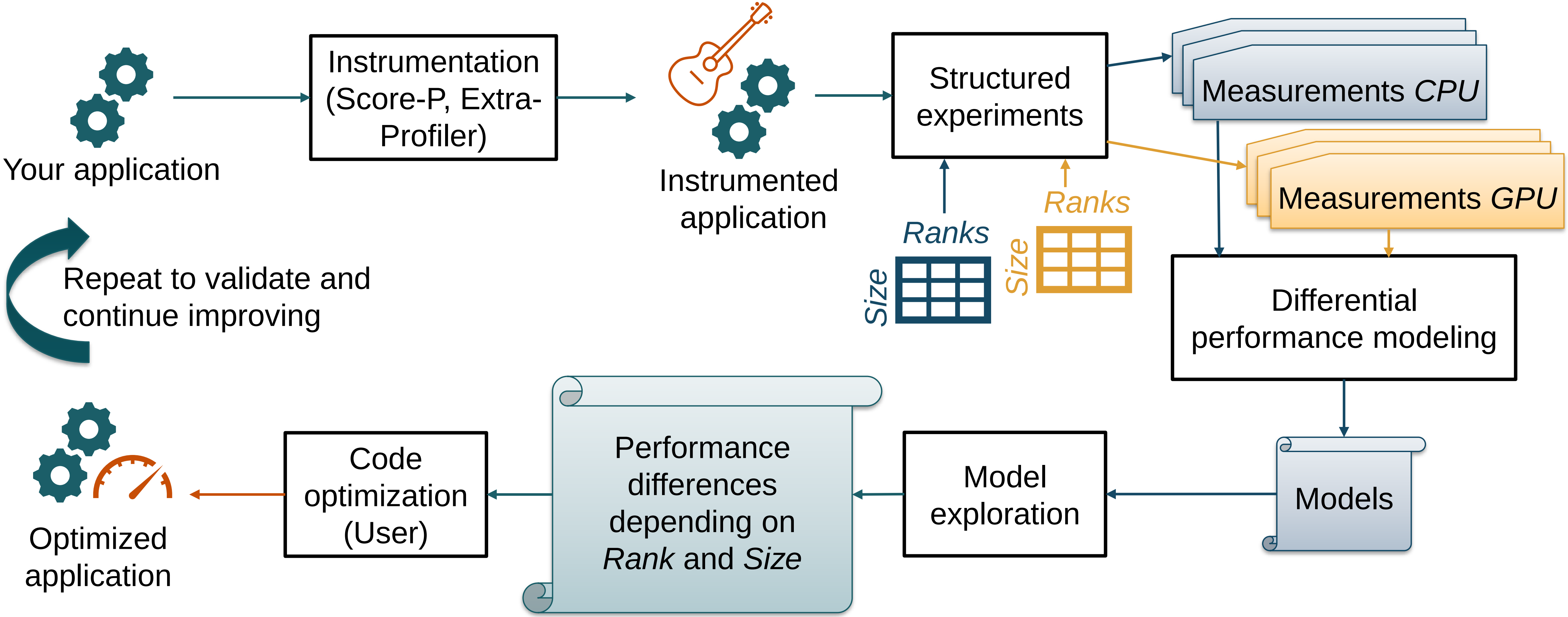
∞≺ ... slower than expected

≻∞ ... faster than expected

≻≺∞ ... faster and slower than expected depending on the parameter

Differential performance models

Summary



Your application
Instrumentation (Score-P, Extra-Profiler)
Instrumented application
Structured experiments
Ranks
Size
Ranks
Size
Measurements *CPU*
Measurements *GPU*
Differential performance modeling
Models
Model exploration
Performance differences depending on *Rank* and *Size*
Code optimization (User)
Optimized application
Repeat to validate and continue improving

Acknowledgement

We acknowledge the support of the European Commission and the German Federal Ministry of Education and Research (BMBF) under the EuroHPC programme DEEP-SEA (GA 955606, BMBF funding No. 16HPC015), which receives support from the European Union's Horizon 2020 programme and DE, FR, ES, GR, BE, SE, GB, CH. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project No. 449683531 (ExtraNoise).

EuroHPC
Joint Undertaking

SPONSORED BY THE
Federal Ministry
of Education
and Research

Funded by
DFG Deutsche
Forschungsgemeinschaft
German Research Foundation