Extra-P:
Insightful Automatic Performance Modeling

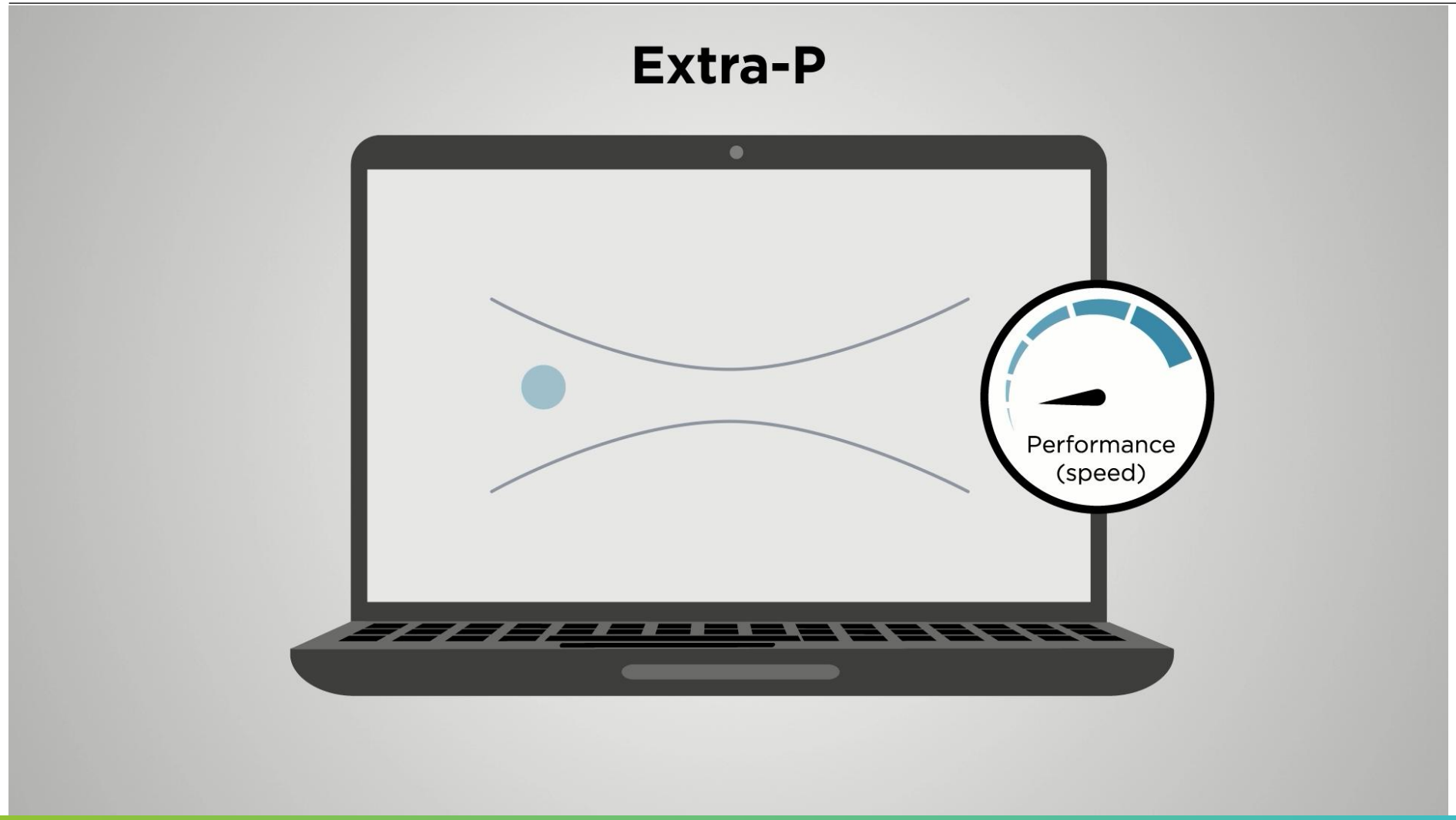
TECHNISCHE UNIVERSITÄT DARMSTADT

Alexander Geiß[1], Gustavo de Morais[1], Marcus Ritter[1],
Alexandru Calotoiu[2], Torsten Hoefler[2], and Felix Wolf[1]

ETH zürich

[1] TU Darmstadt , [2] ETH Zürich

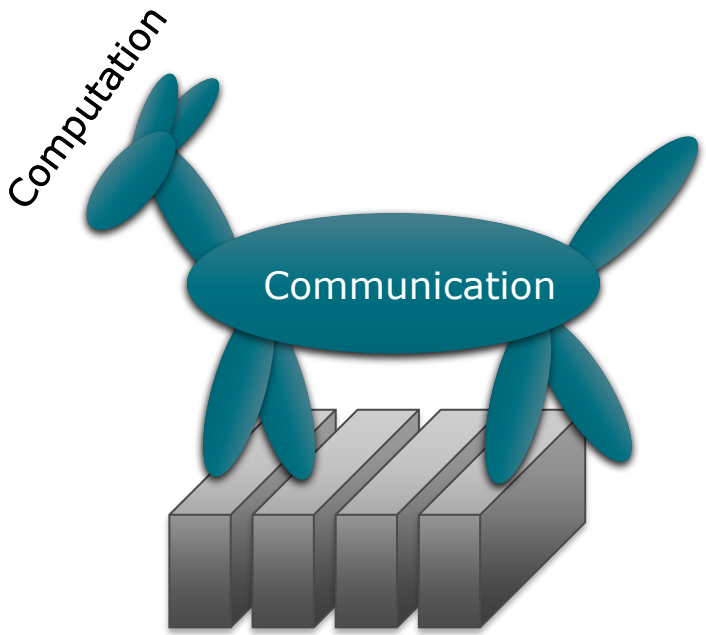
Introduction

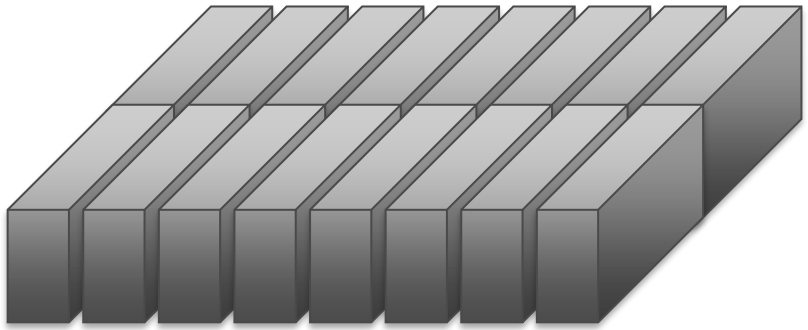


Watch Extra-P overview video

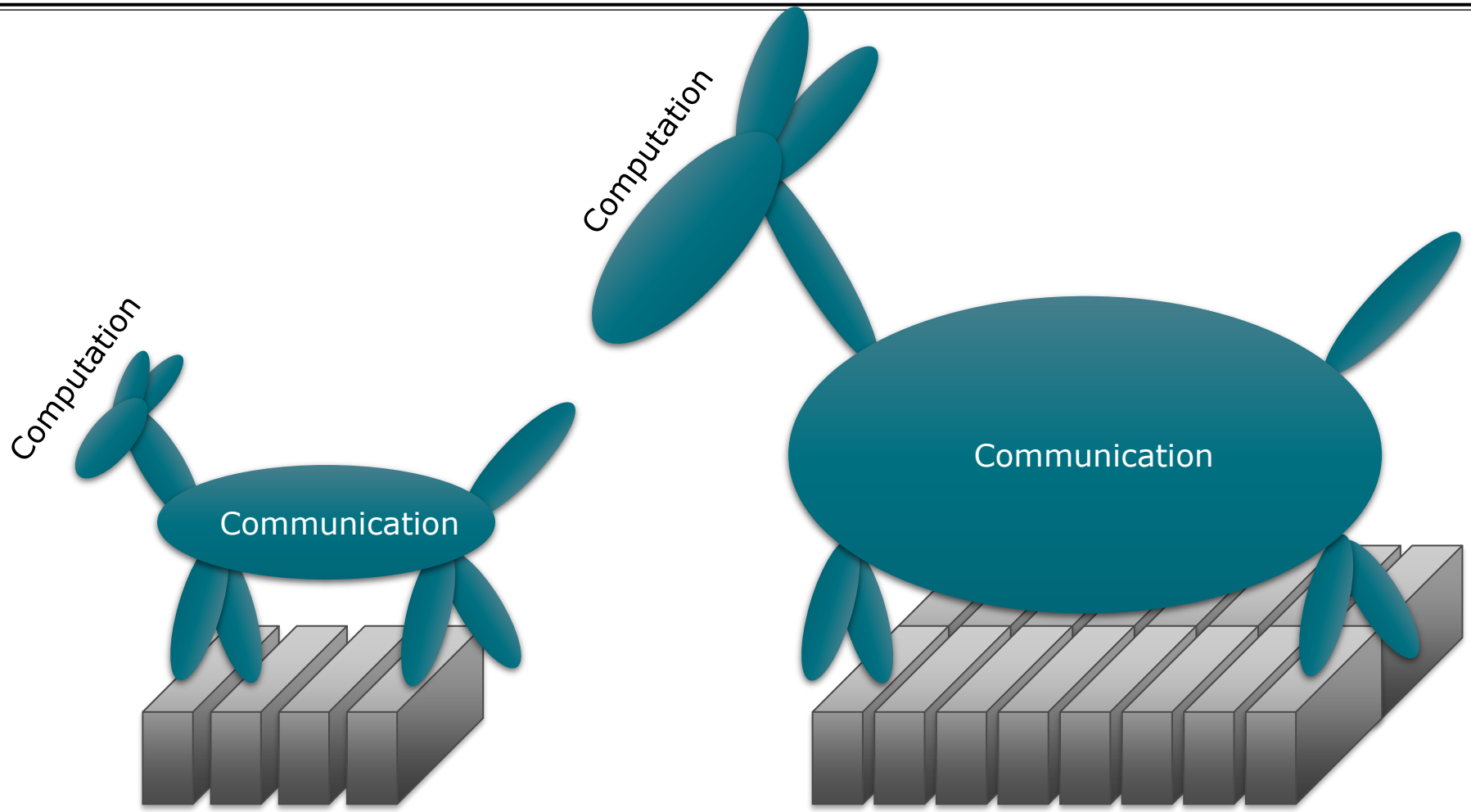
https://www.youtube.com/watch?v=Cv2YRCMWqBM

Motivation - latent scalability bugs



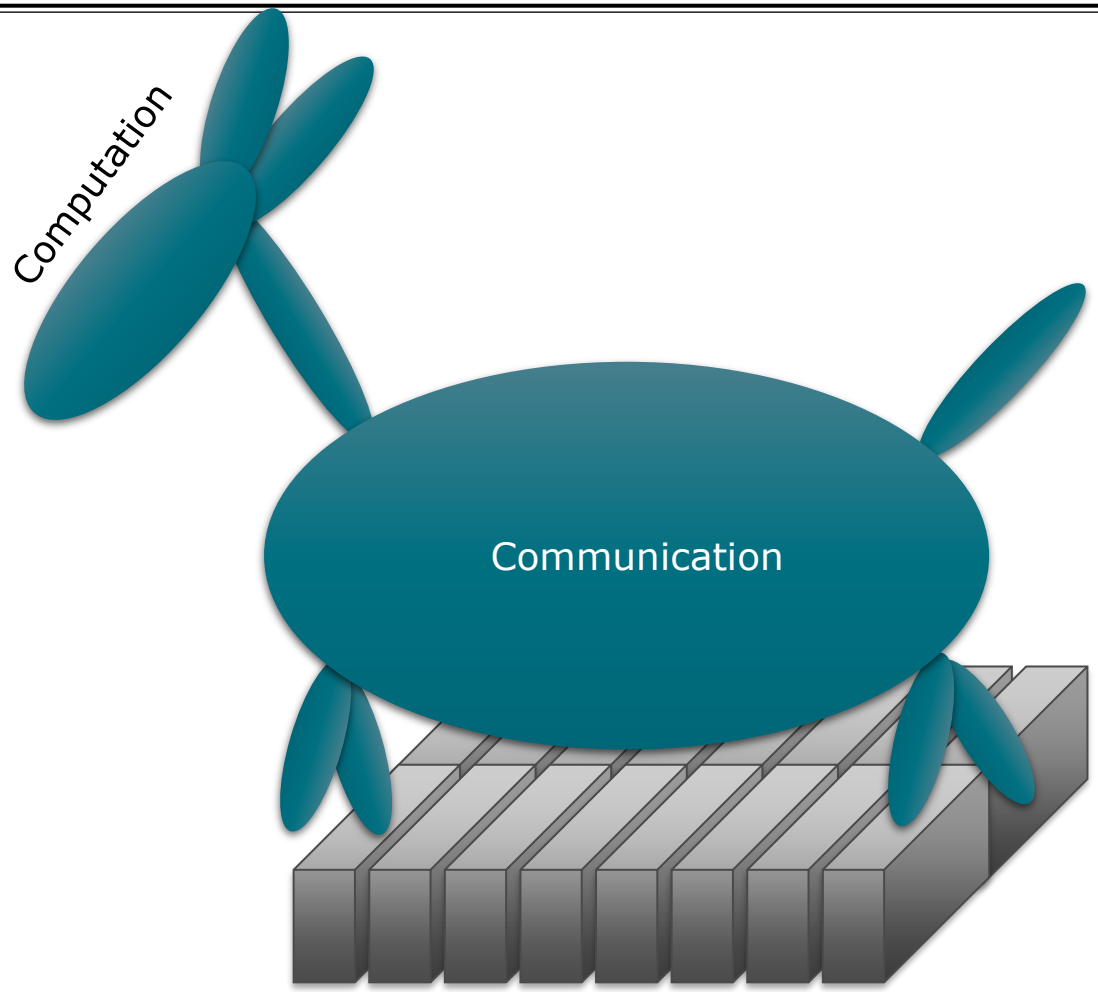


Scaling code to a bigger machine can unveil unpleasant surprises

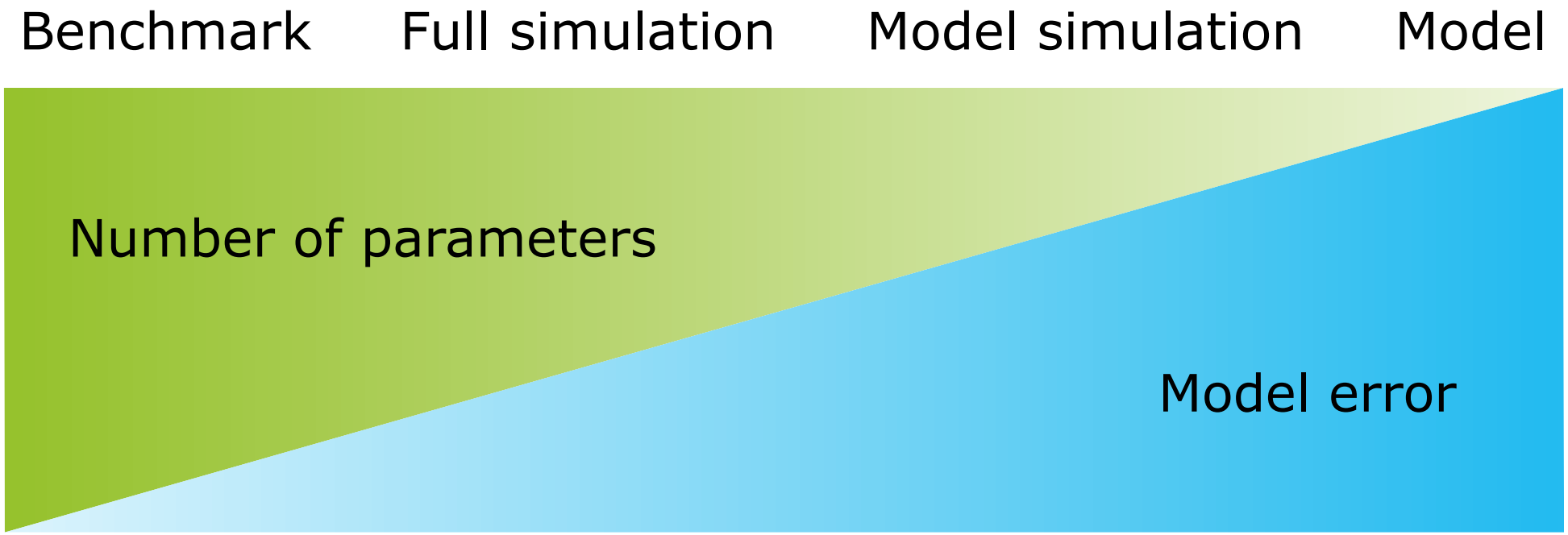


Scaling code to a bigger machine can unveil unpleasant surprises

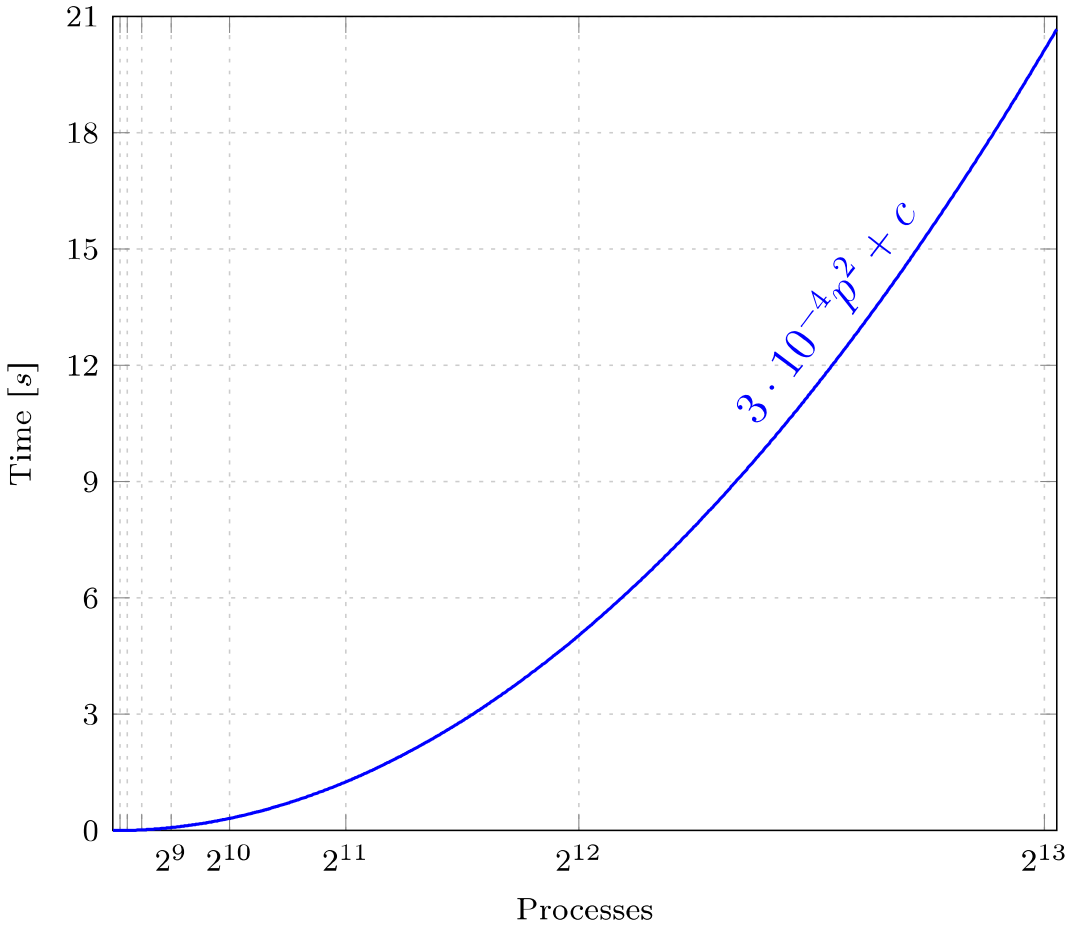
We need to find scaling issues before they occur



Spectrum of performance analysis methods



Scaling model



- Represents performance metric as a function of the number of processes
- Provides insight into the program behavior at scale

Analytical performance modeling

Identify kernels

- Parts of the program that dominate its performance at larger scales
- Identified via small-scale tests and intuition

Create models

- Laborious process
- Still confined to a small community of skilled experts

Disadvantages:

- Time consuming
- Danger of overlooking unscalable code

Hoisie et al.: *Performance and scalability analysis of teraflop-scale parallel architectures using multi-dimensional wavefront applications.* International Journal of High Performance Computing Applications,2000

Bauer et al.: *Analysis of the MILC Lattice QCD Application su3_rmd*. CCGrid, 2012

Automatic performance modeling

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

Input

Instrumentation

- All functions

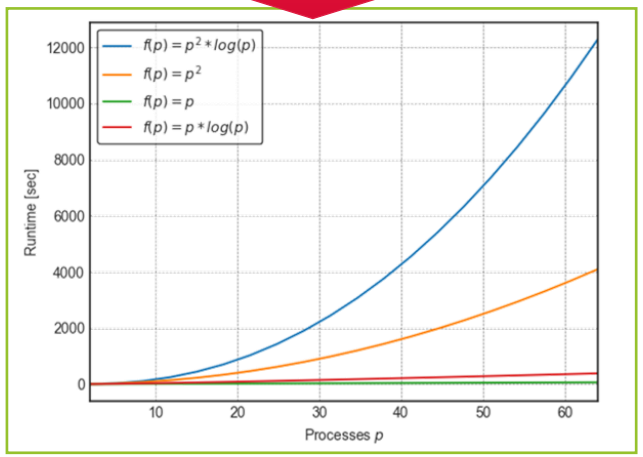
Performance measurements

M_i M_j

Extra-P

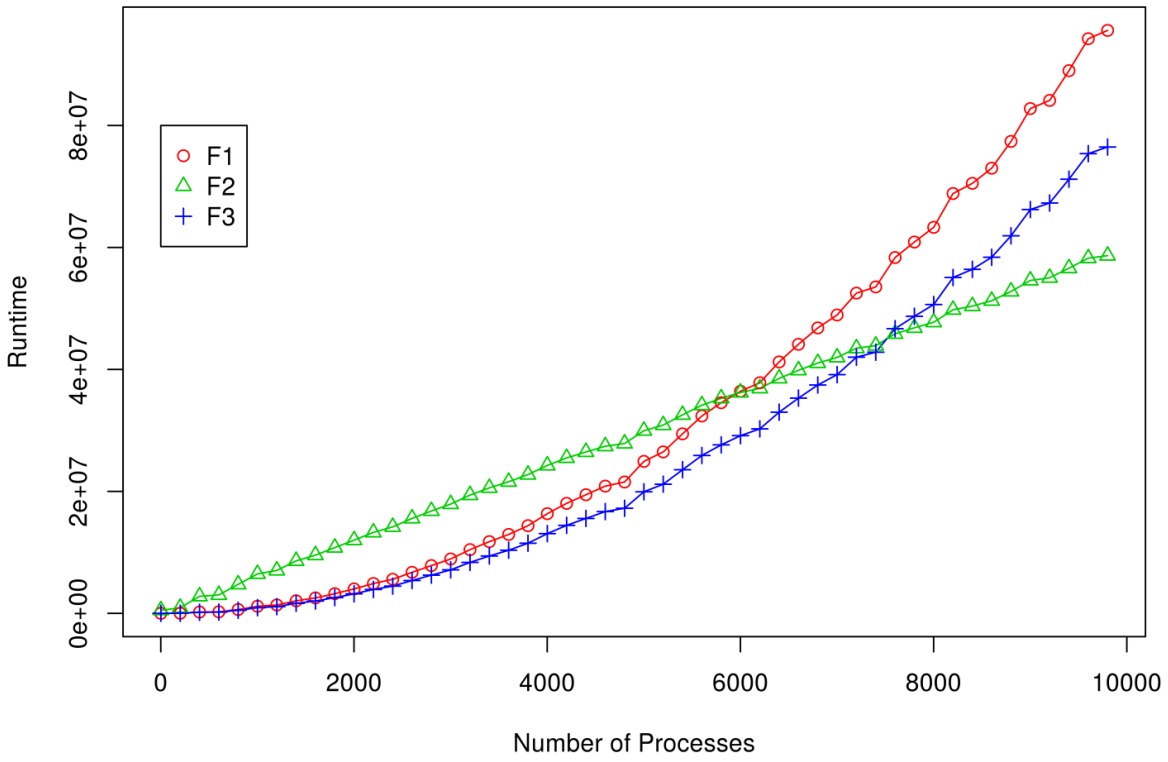
Output

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

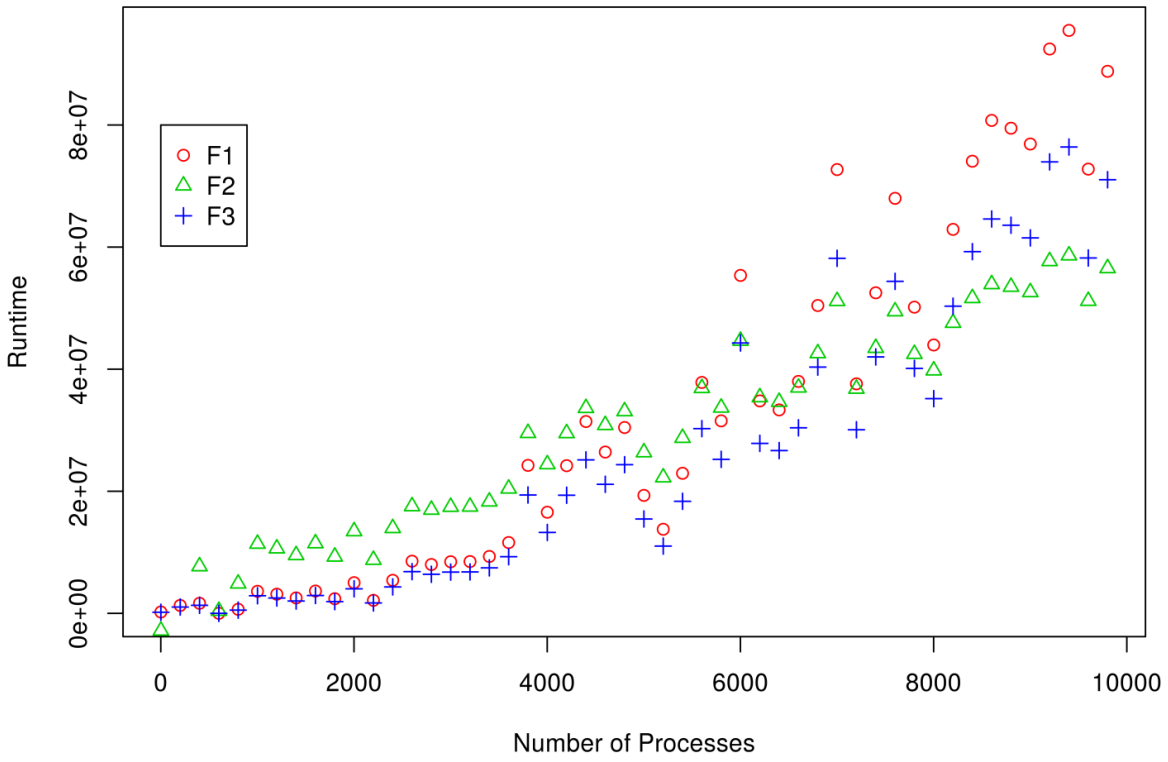


Primary focus on scaling trend

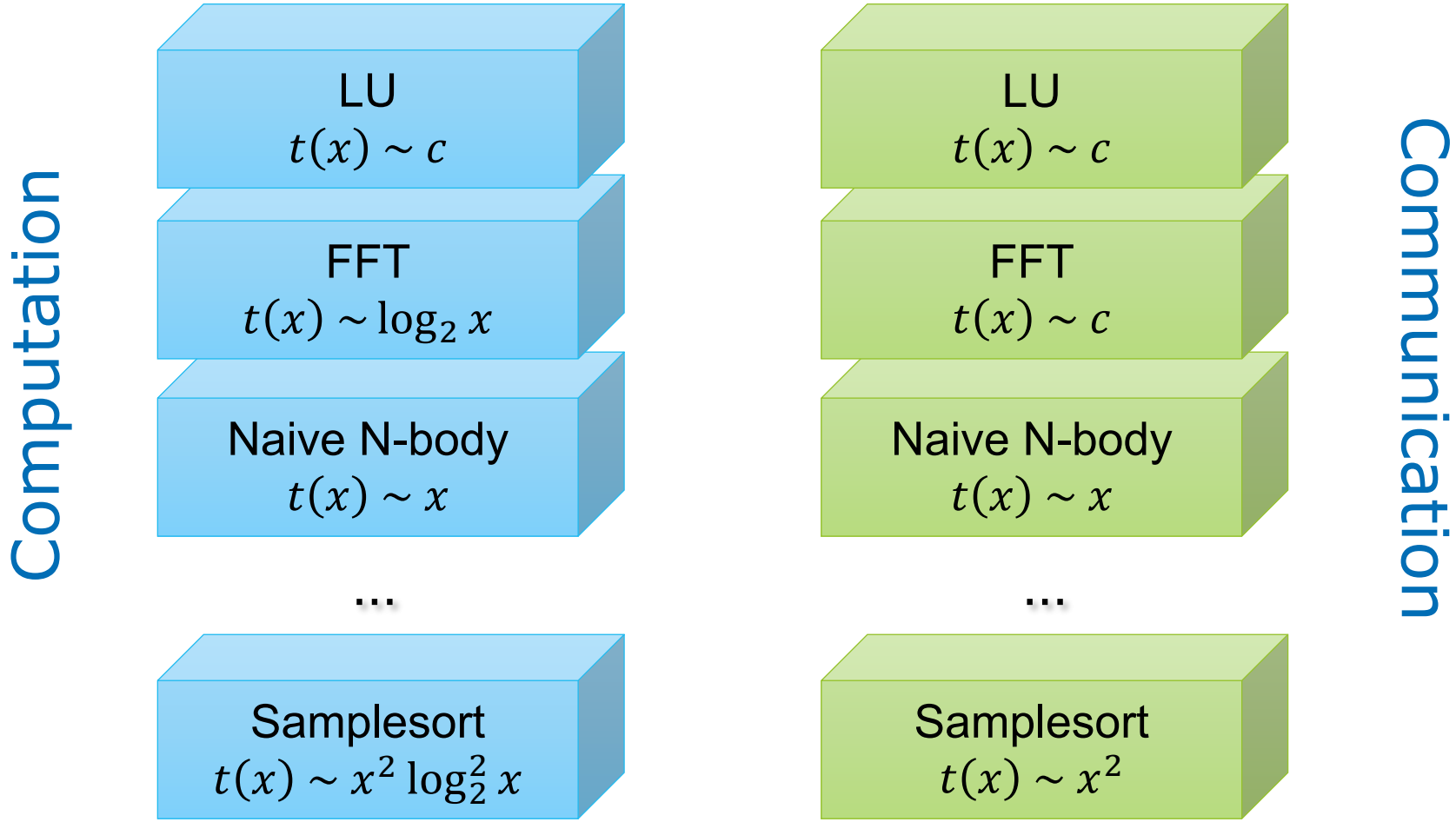
Common performance analysis chart in a paper



Production reality



Model building blocks



Performance model normal form

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

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

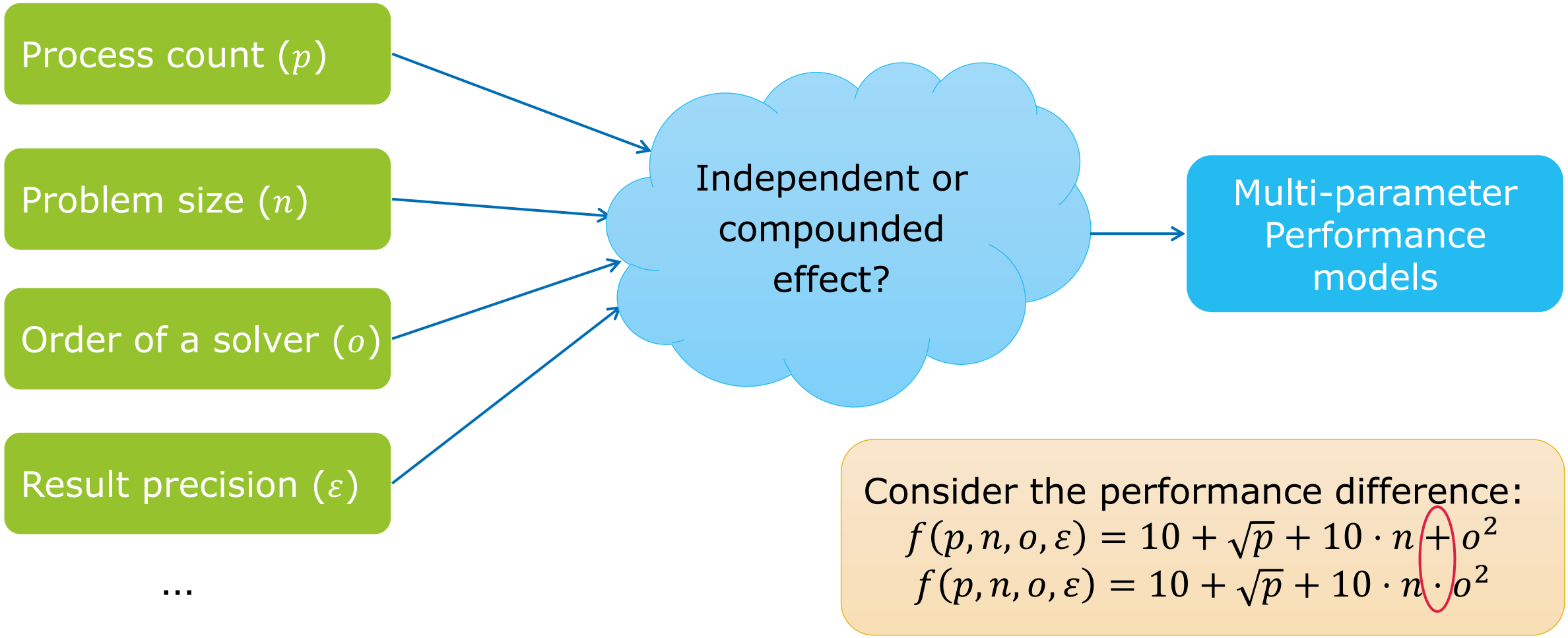
$$n = 1$$
$$I = \{0, 1, 2\}$$
$$J = \{0, 1\}$$

c_1 $\quad$ $c_1 \cdot \log x$

$c_1 \cdot x$ $\quad$ $c_1 \cdot x \cdot \log x$

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

Fast multi-parameter performance modeling



Fast multi-parameter performance modeling

- Expanded performance model normal form

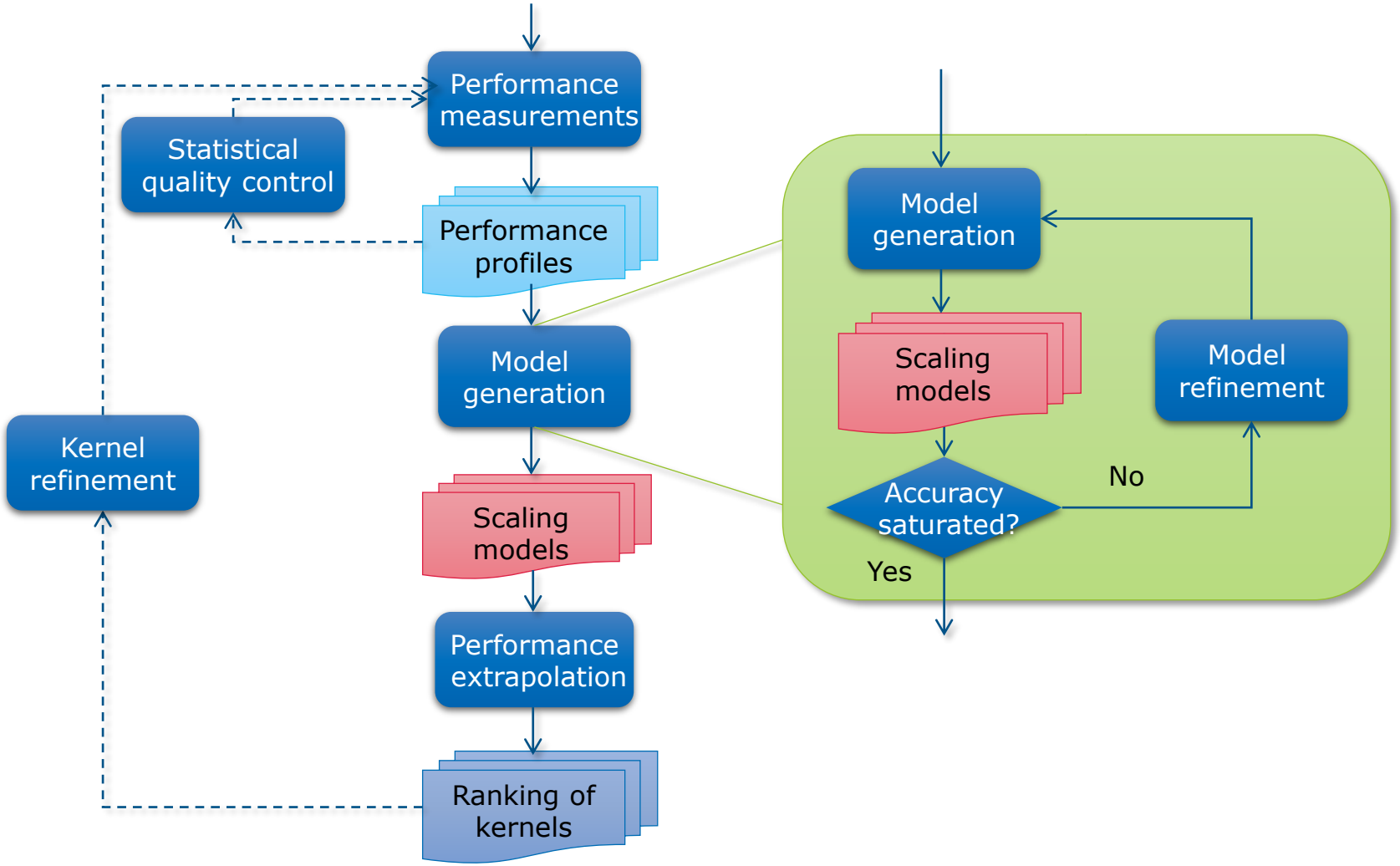
$$f(x_1, \ldots, x_m) = \sum_{k=1}^{n} c_k \prod_{l=1}^{m} x_l^{i_{kl}} \cdot log_2^{j_{kl}}(x_l)$$

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

Model candidates

- Constant c_1
- Single parameter $c_1 + c_2 \cdot x_1$
- Multiple parameters …
  - Additive $c_1 + c_2 \cdot x_1 + c_3 \cdot x_2$
  - Multiplicative $c_1 + c_2 \cdot x_1 \cdot x_2$
  - Complex $c_1 + c_2 \cdot x_1 \cdot x_2 + c_3 \cdot \log x_2 \cdot x_2^3$

Workflow

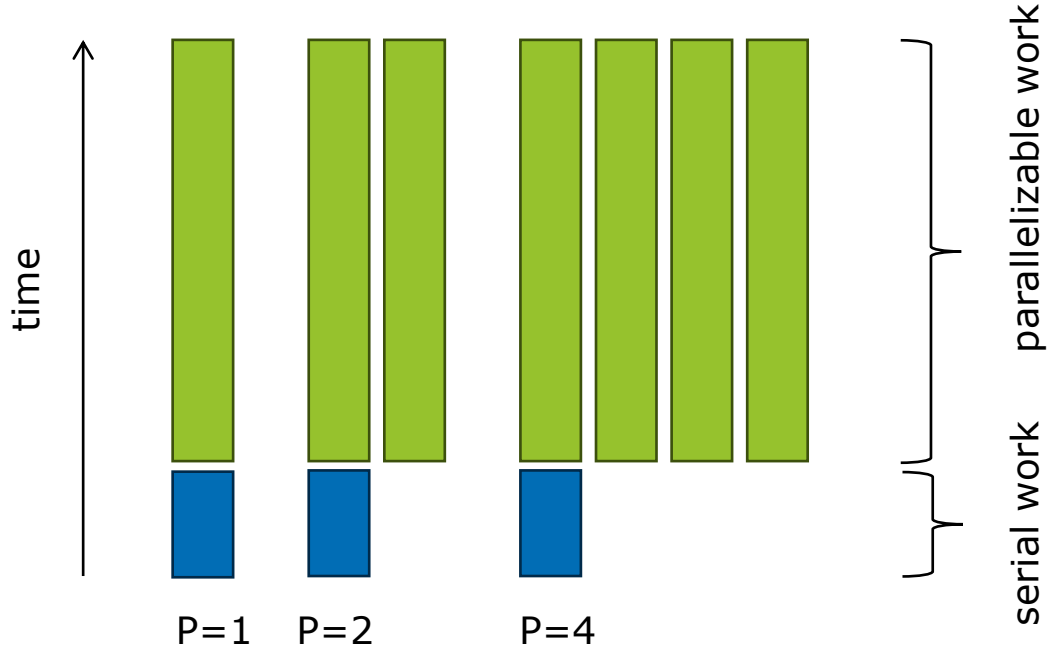

Performance measurements
Statistical quality control
Performance profiles
Model generation
Kernel refinement
Scaling models
Performance extrapolation
Ranking of kernels
Model generation
Scaling models
Model refinement
Accuracy saturated?
No
Yes

Scaling analysis: number of processes is increased

Preferred by Extra-P

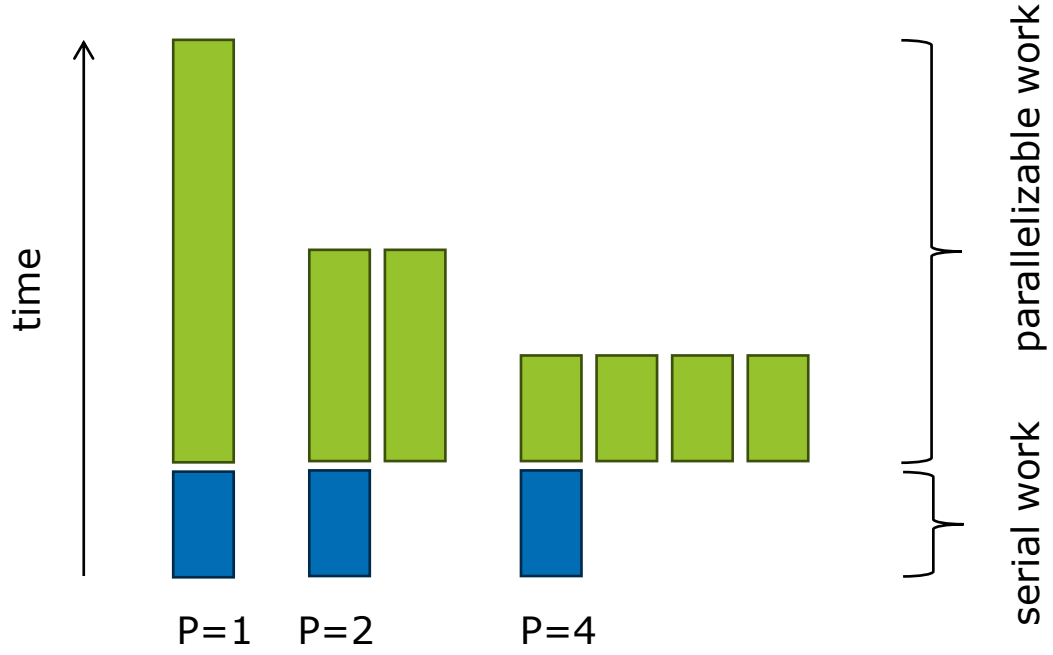
Weak scaling

- The problem size is increased alongside
- Law of Gustafson



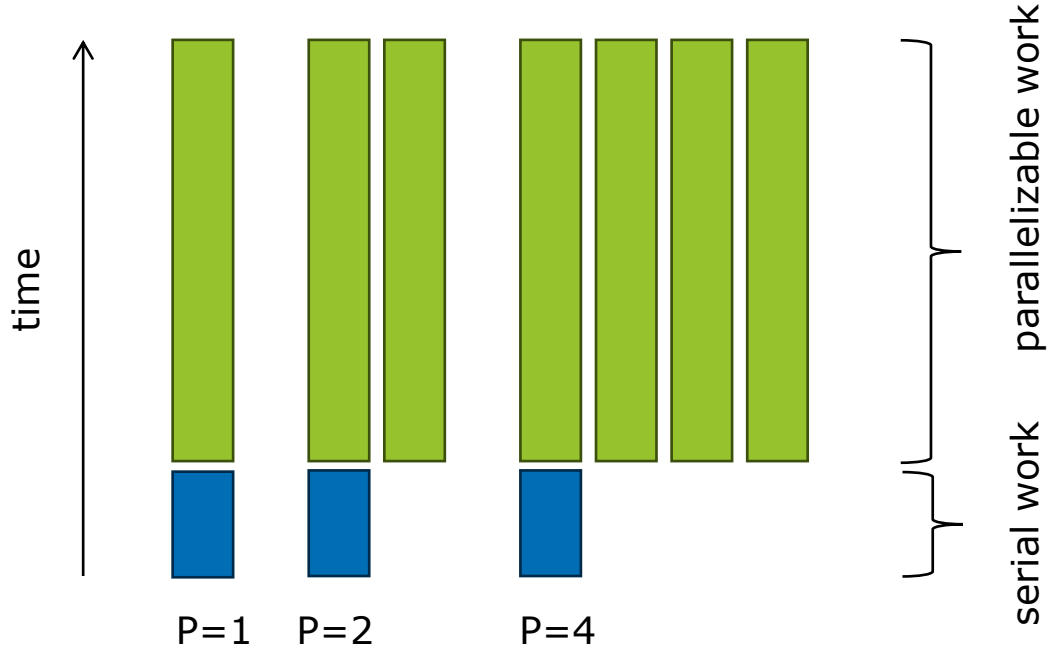
Strong scaling

- The problem size remains constant
- Amdahl's law

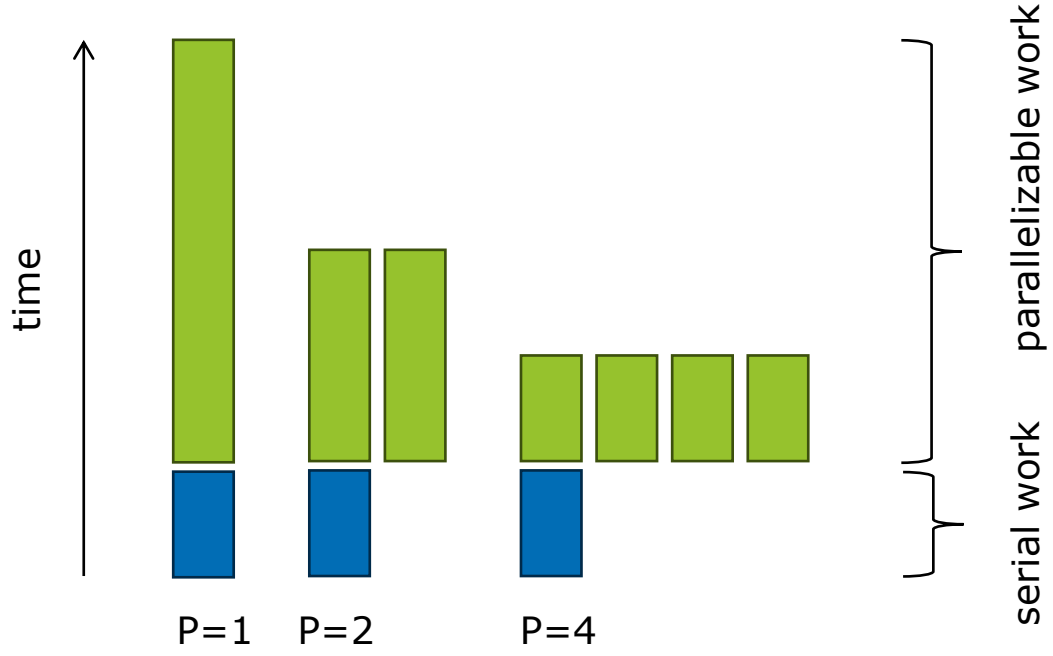


Scaling analysis with Extra-P

Weak scaling



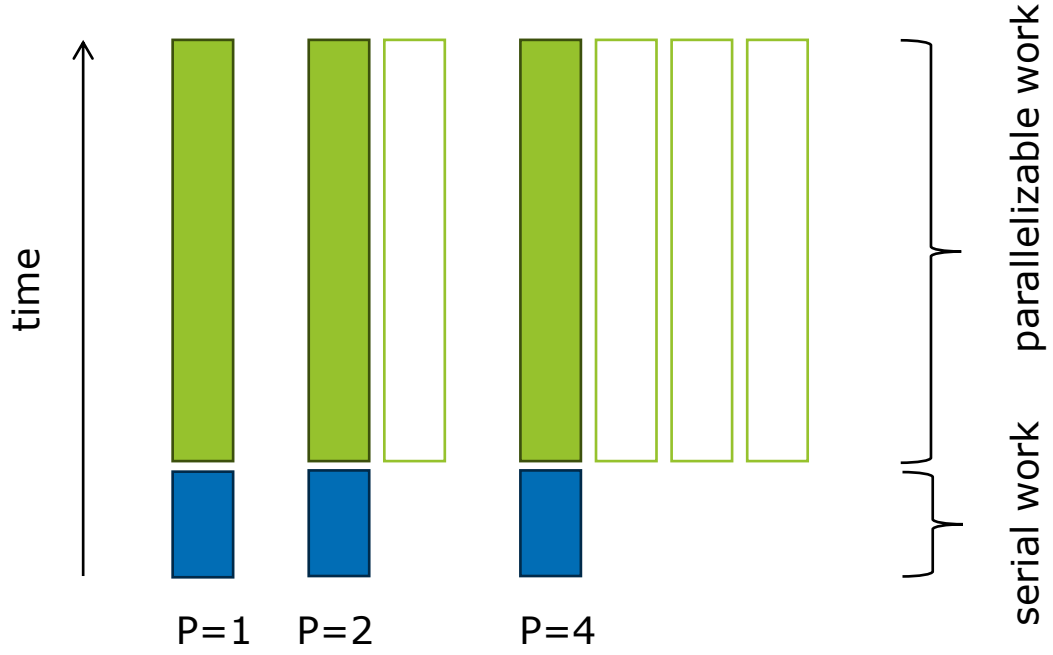
Strong scaling



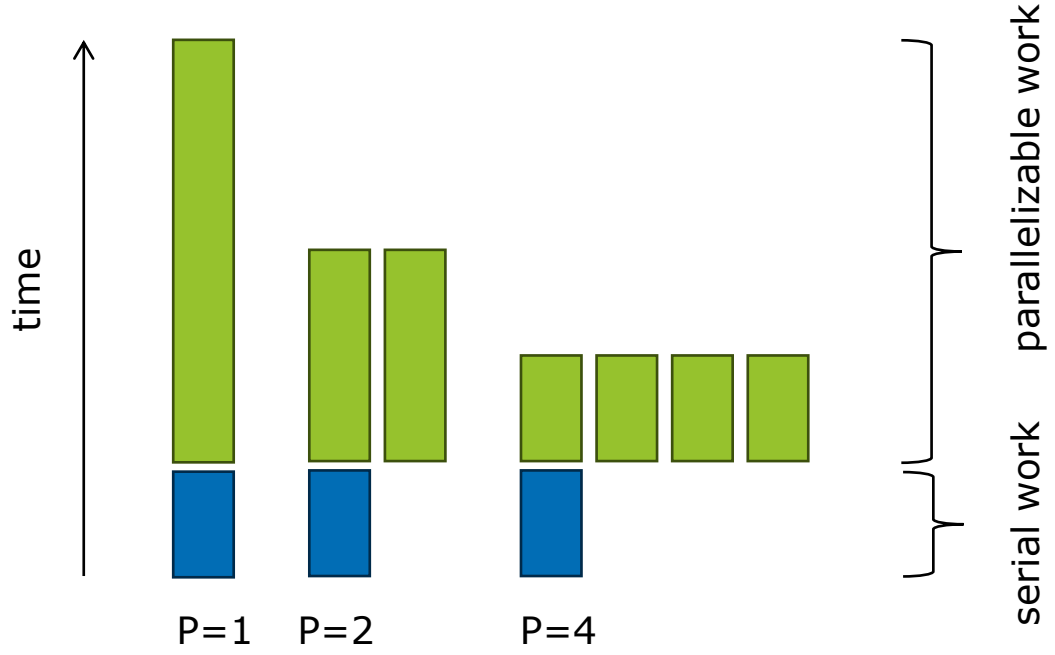
Scaling analysis with Extra-P

Weak scaling

- Extra-P models the runtime of one process



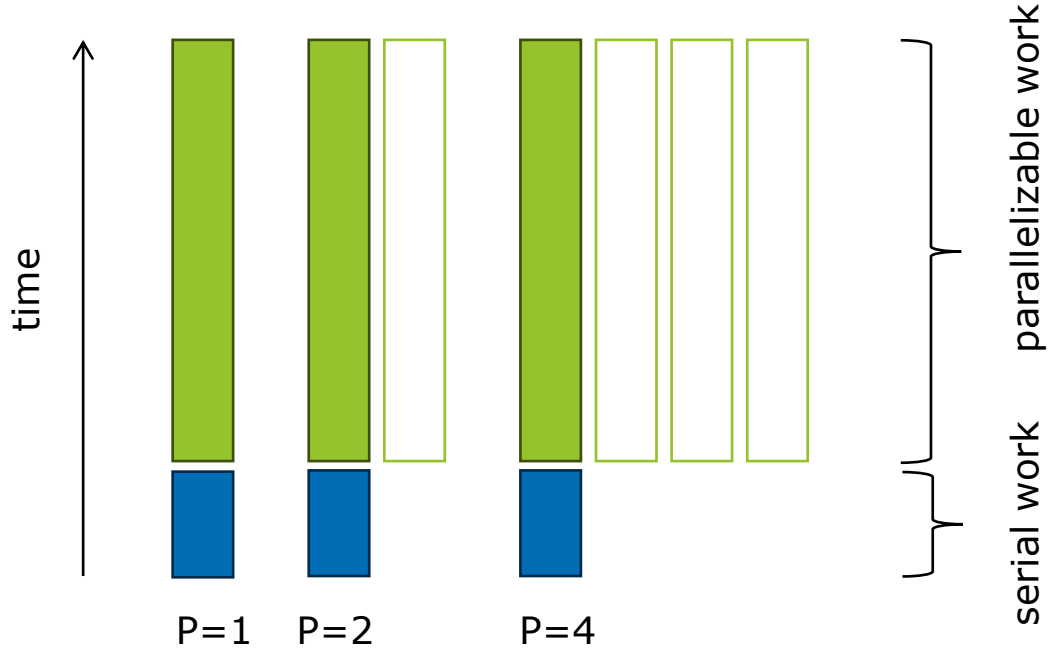
Strong scaling



Scaling analysis with Extra-P

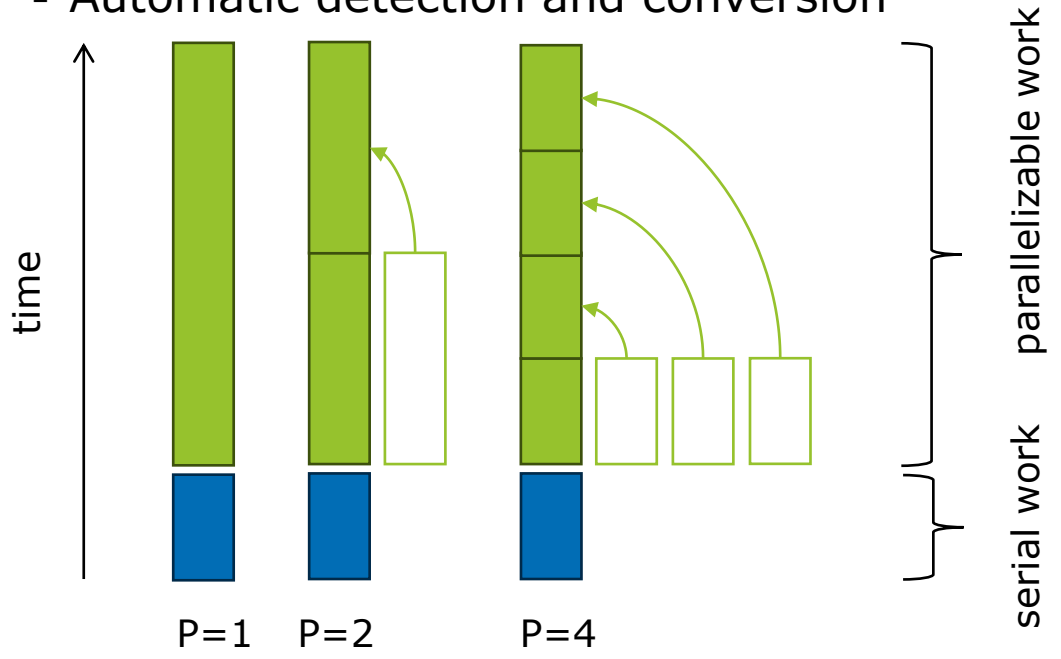
Weak scaling

- Extra-P models the runtime of one process



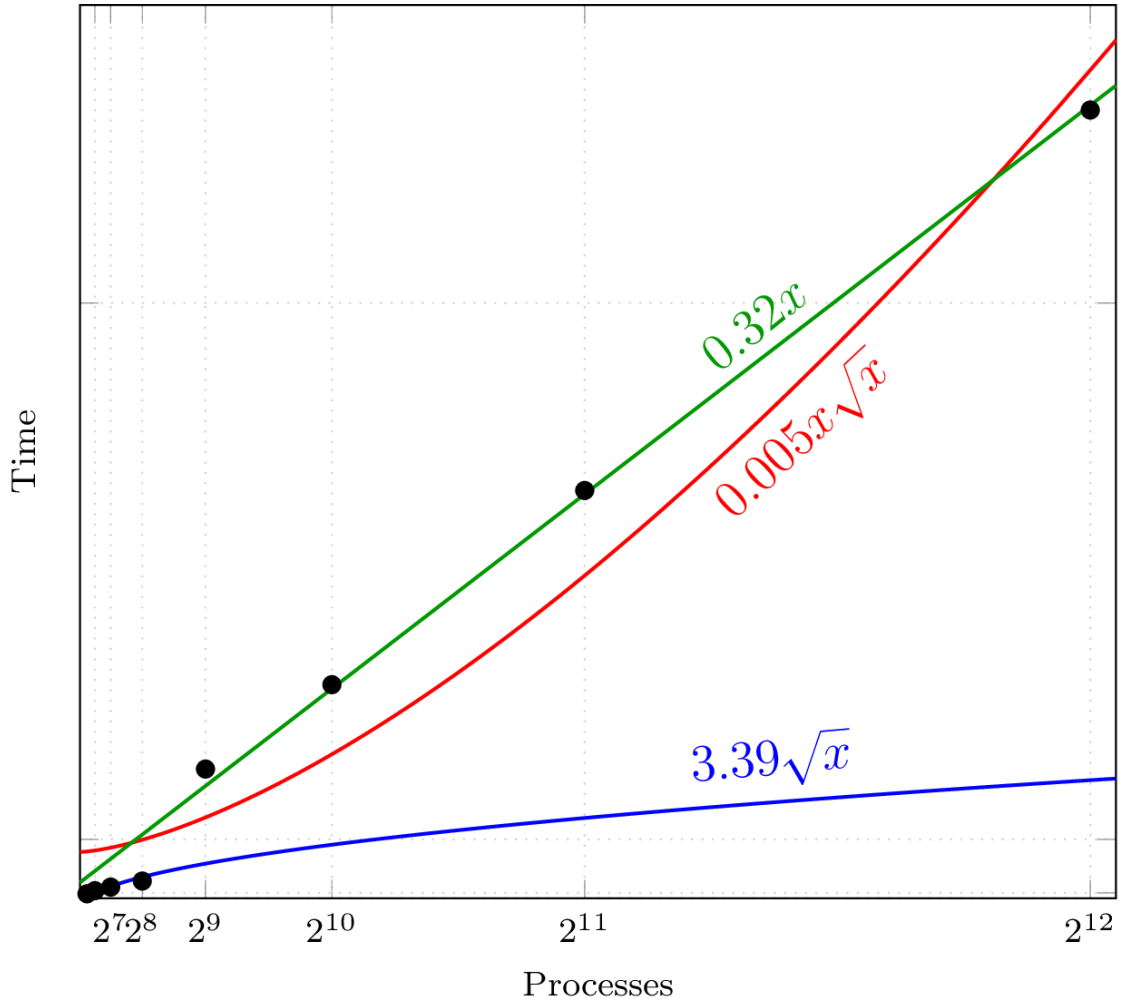
Strong scaling

- Extra-P models the resource consumption
  - Runtime of all processes combined
  - Equivalent to the number of core-hours
  - Automatic detection and conversion



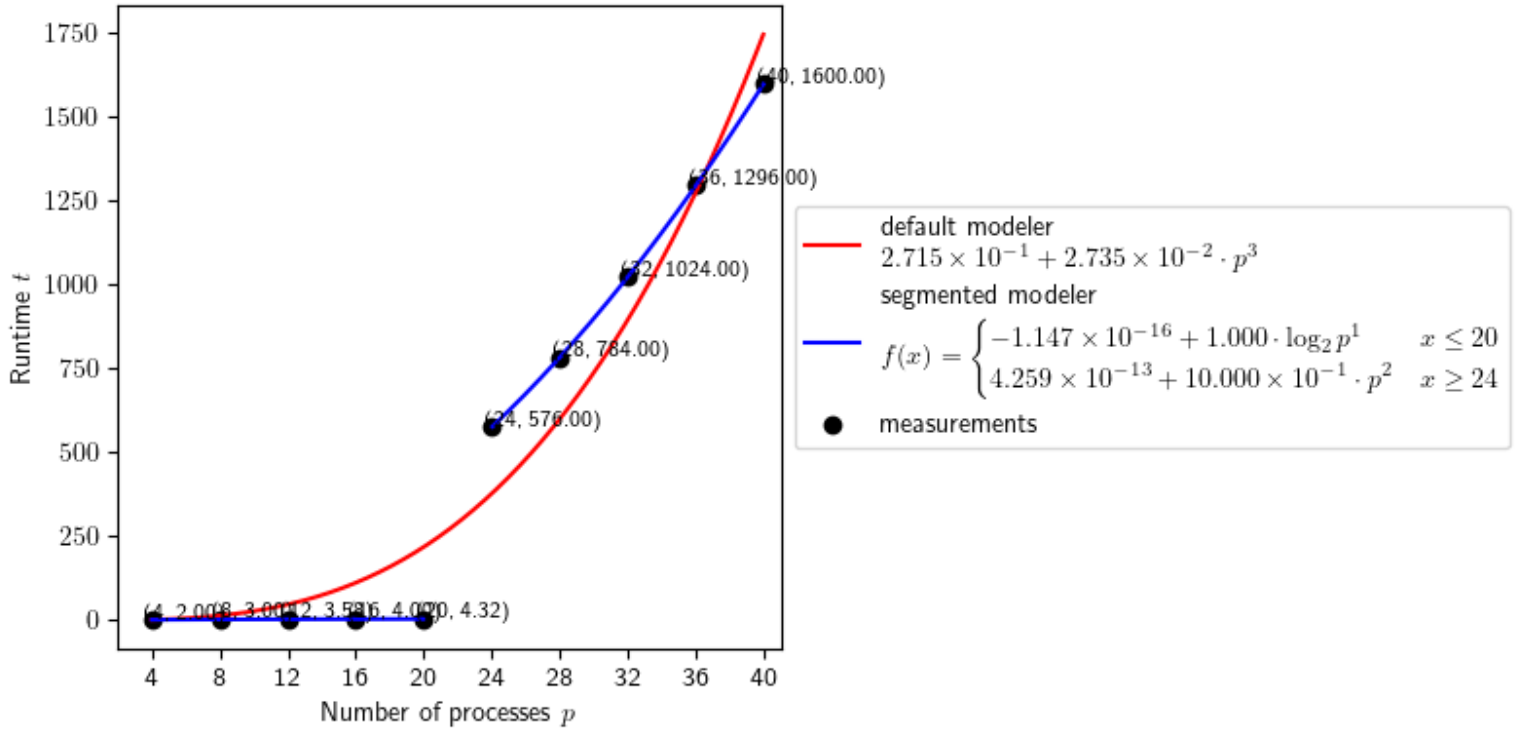
Assumptions & limitations

- Scaling behavior expressible with performance model normal form
- Only one scaling behavior for all the measurements; no jumps
- Some MPI collective operations switch their algorithm
  - results in bad models
- Example: red model tries to model measurements of different algorithms
  - First 4 points – one function
  - Last 4 points – another function (linear)
  - Adj. R2 = 0.95085 (!)



Segmented models

- We can detect and model segmented behavior
- When enough measurement points are present
- Segmented modeler must be manually selected
- Limited to two segments



Performance measurements

Different ways of collecting measurements

- Score-P (http://www.vi-hps.org/projects/score-p/)
- Other profiling tools, e.g. HPCToolkit
- Manual ad-hoc measurements

Performance measurements (2)

- At least 5 different measurements recommended

Performance measurements (profiles)

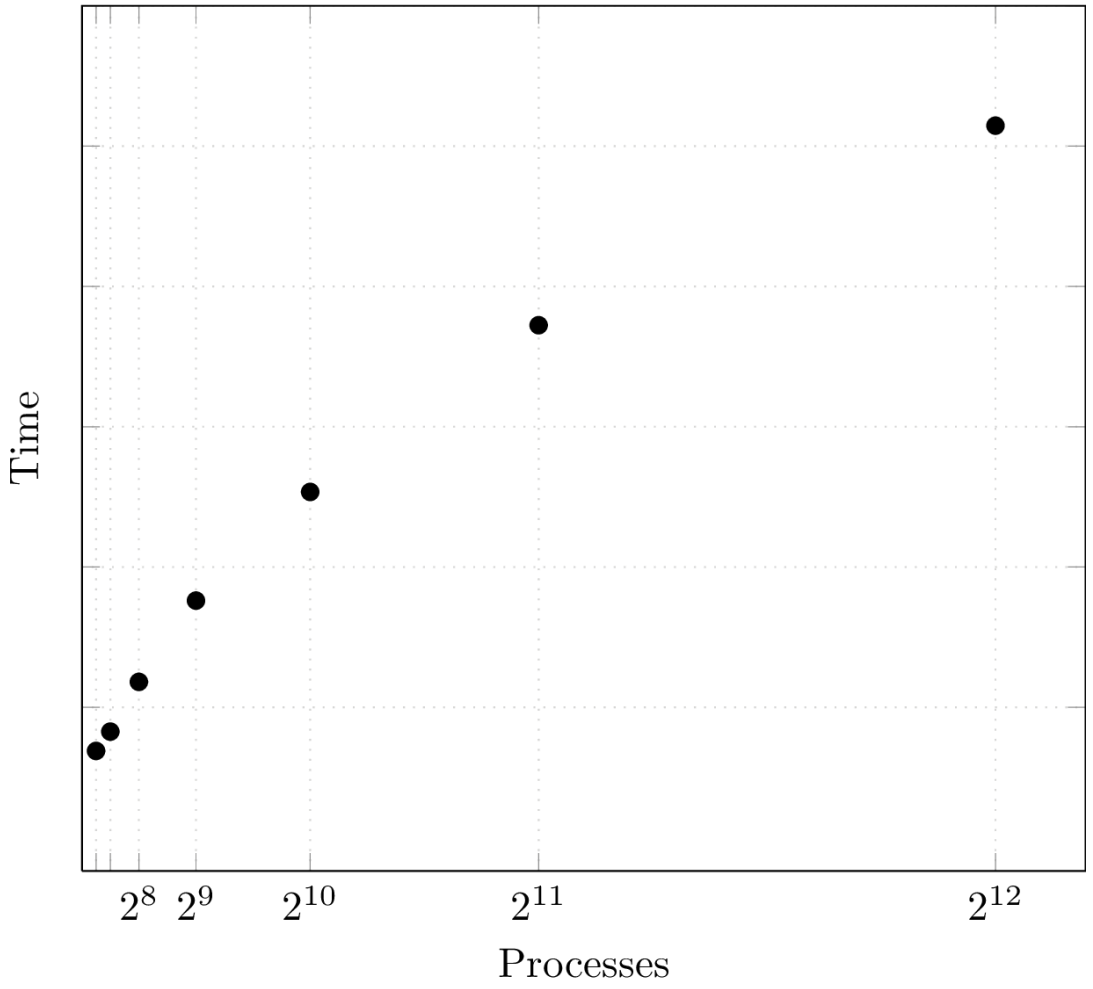
$p_1 = 256$

$p_2 = 512$

$p_3 = 1024$

$p_4 = 2048$

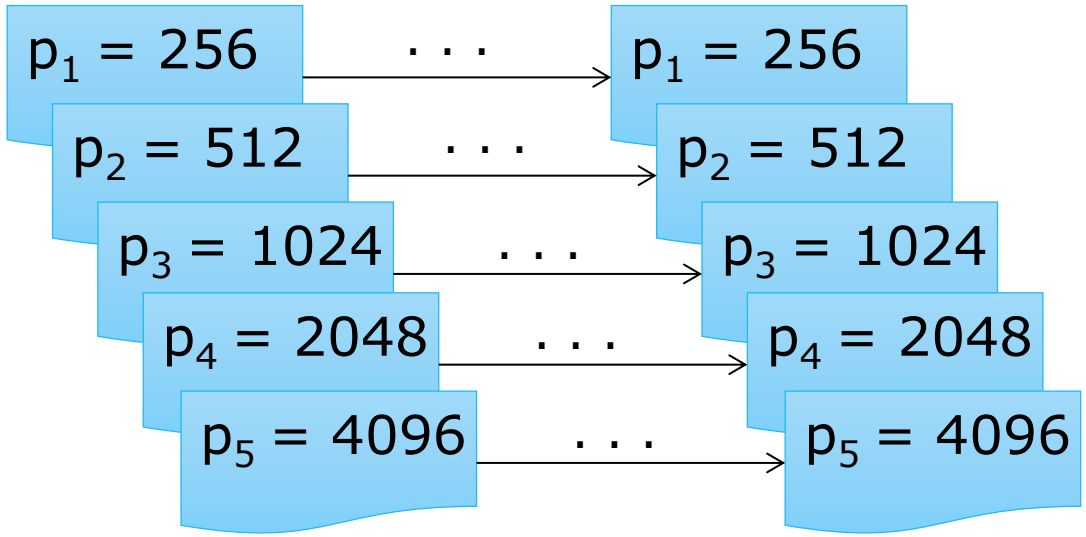
$p_5 = 4096$

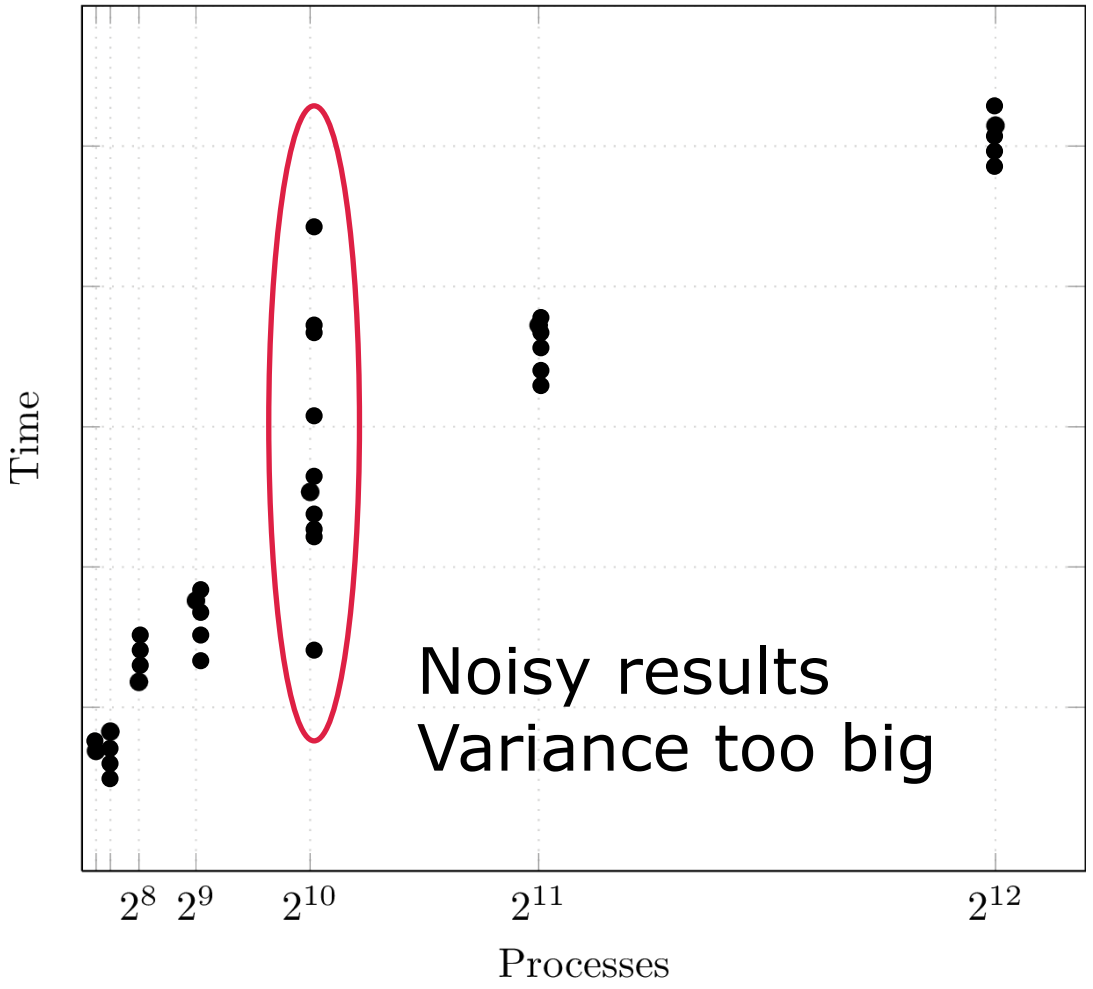


Performance measurements (3)

- At least 5 different measurements recommended
- Each measurement repeated multiple times

Performance measurements (profiles)





Adjusted R^2

- R^2 represents how well the determined function fits the M available measurements

- Adjusted R^2 adjusts for N, the number of terms used
  - Adj. R^2 decreases → more useless variables
  - Adj. R^2 increases → more useful variables

- Rule of thumb: adj. $R^2 > 0.95$

$$R^2 = 1 - \frac{\text{residualSumSquares}}{\text{totalSumSquares}}$$

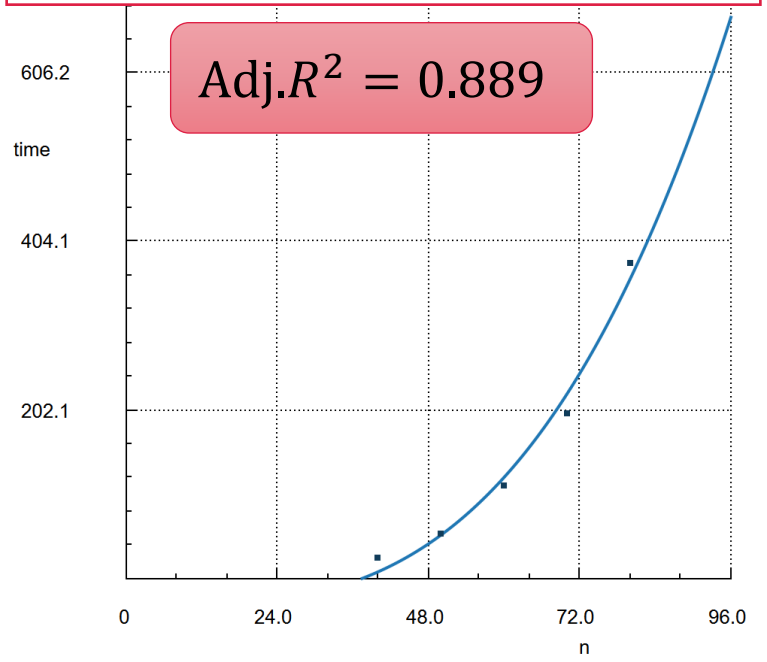
$$\overline{R^2} = 1 - (1 - R^2) \cdot \frac{M - 1}{M - N - 2}$$

Quadratic and cubic problems

- The whole problem size should be used as parameter
  - Not just the edge length

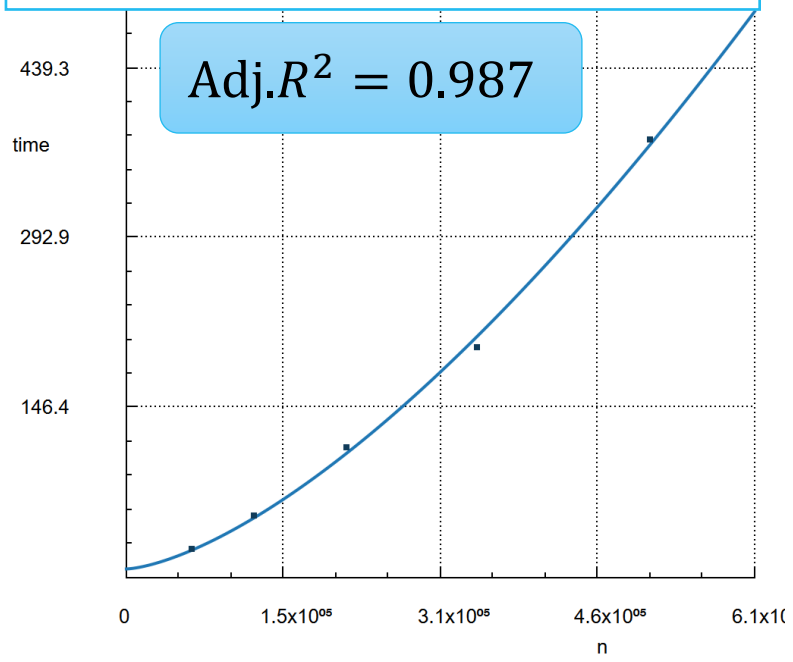
Edge length: n

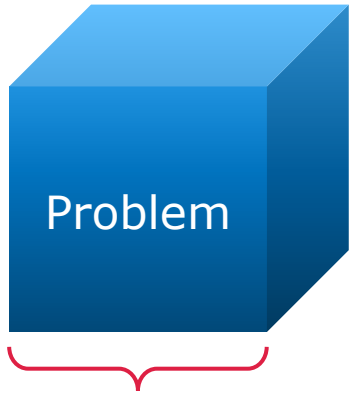
$-32.98 + 0.000121 * n^3 * \log2(n)$



Volume: n

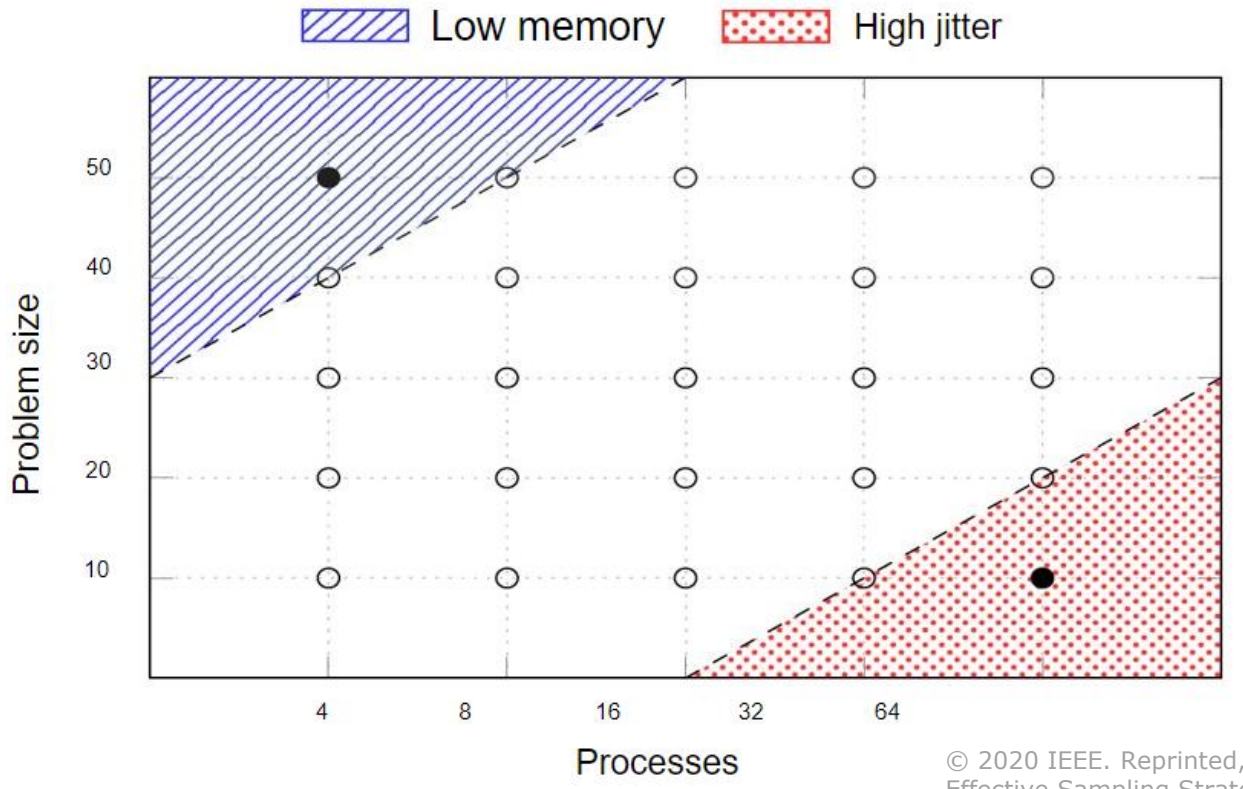
$7.53 + 9.98 \cdot 10^{-7} * n^{1.5}$





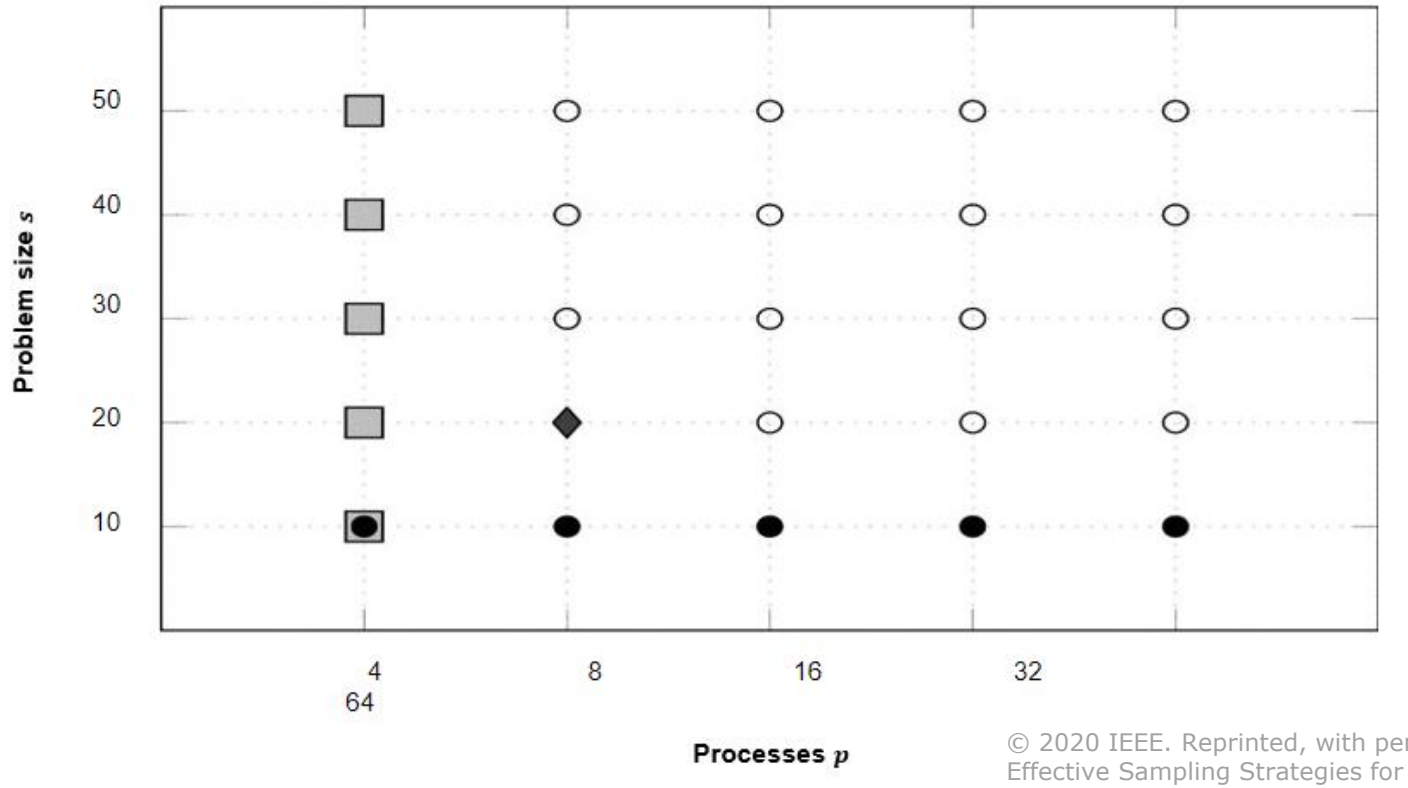
Sparse modeling

- Experiments can be expensive
- So far we needed 5×5^m experiments, m=number of parameters



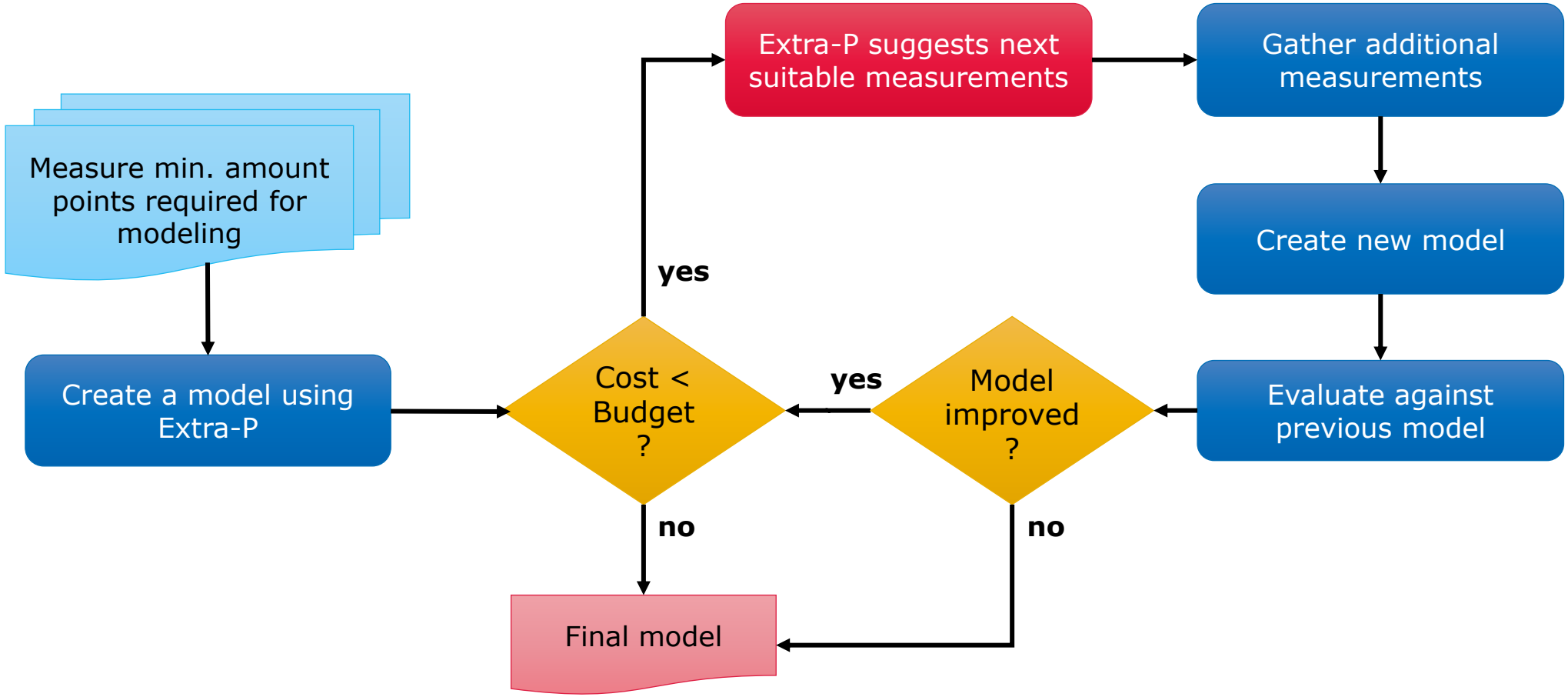
Sparse modeling

- Using our new sparse modeling approach we can model with less points!
- We only need $5 \times 5 \cdot m$ experiments to start, m=number of parameters



Sparse modeling with measurement point suggestions

- Recommends experiment configuration strategy using our heuristics



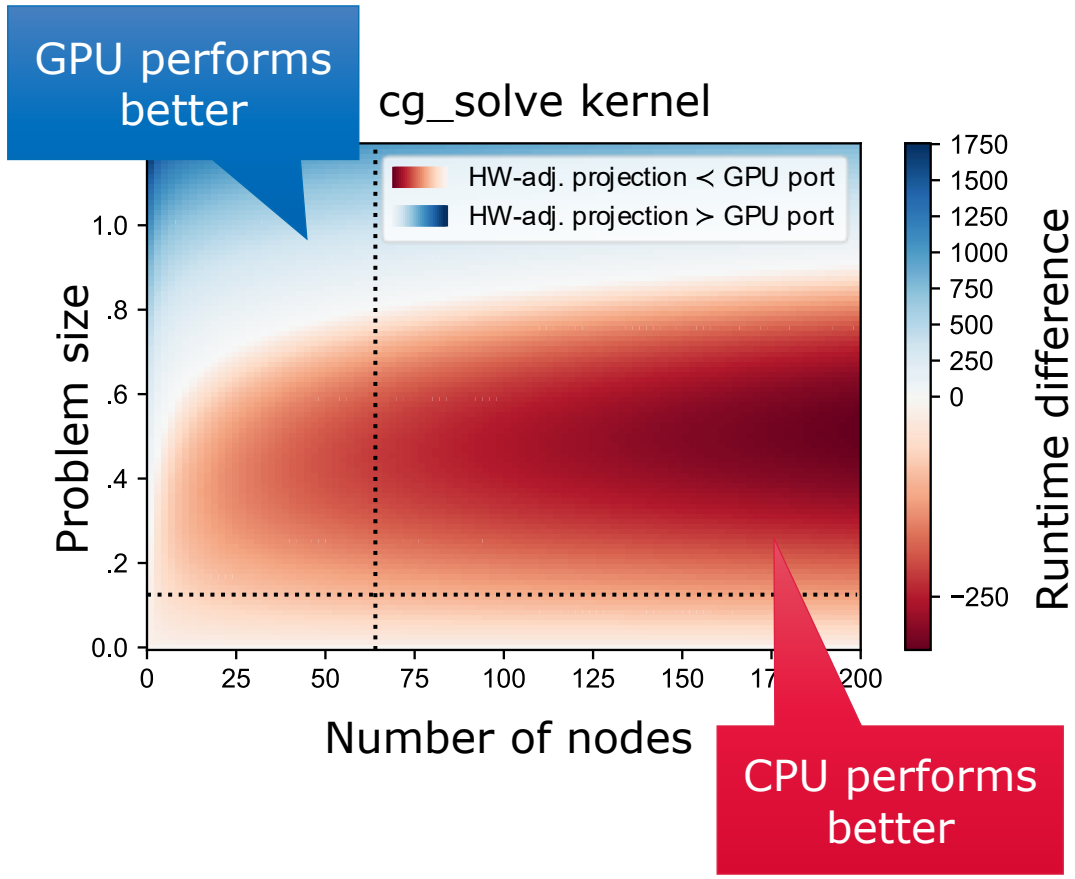
Does a GPU port meet scaling expectations?

Beta

- GPUs are a prime vehicle to improve performance and energy efficiency
- However, porting to GPUs is hard

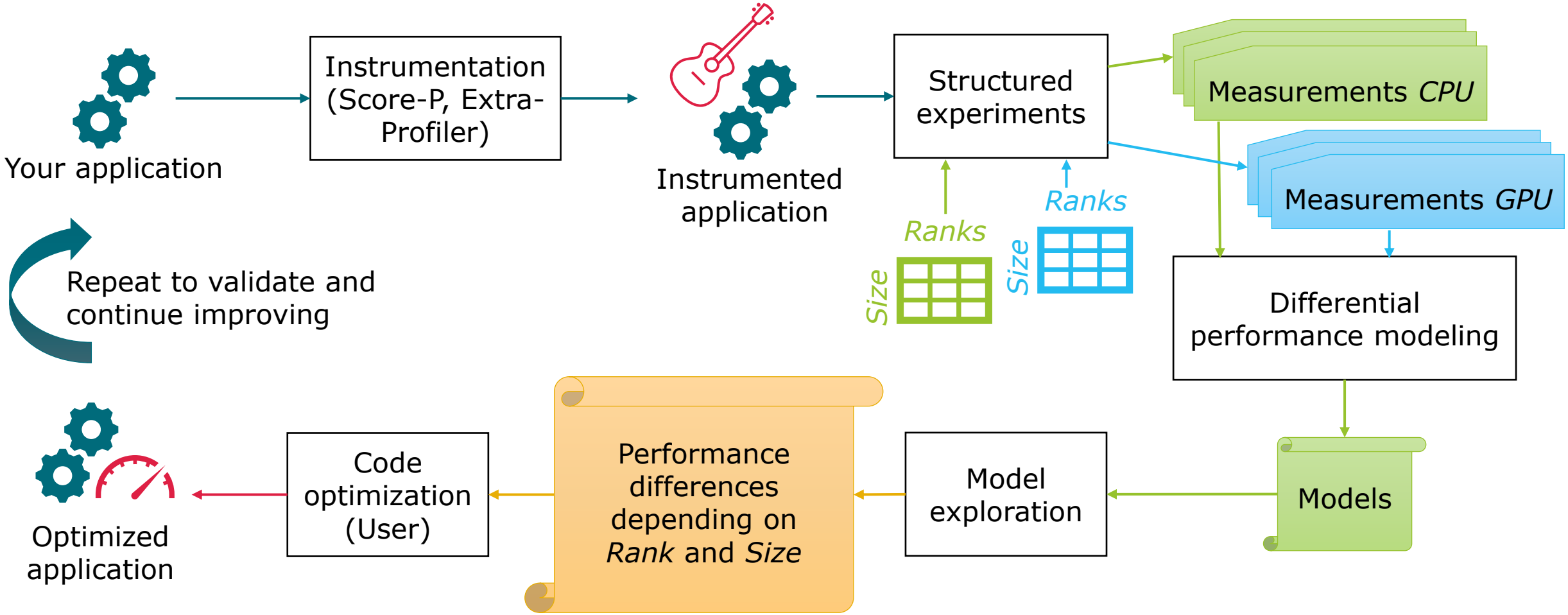
- Challenges of comparison
  - Re-structured source code
  - Asynchronous execution on GPU
  - Defining expectations

- Extra-P can help you with that!
  - Comparison assistant
  - Automatic call-tree mapping



Validating the performance of GPU ports

Beta



Using Extra-P

Installing Extra-P

- Easy to install via pip
- Just run: `python -m pip install extrap –upgrade --pre`
  - The `--upgrade` forces the installation of a new version
- All dependencies (packages) will be installed automatically

Use `--pre` to get the beta version

Extra-P in the tuning workshop

- Available at: https://github.com/extra-p/extrap

- When installed on the system simply run:
  - `extrap` – for the command line version
  - `extrap-gui` – for the graphical user interface version

- The GUI version is not intended to be used on the cluster

Already installed on COSMA, just do `module load extrap`

Automatic performance modeling with Extra-P

```
Myapp.cpp
main() {
    foo()
    bar()
    compute()
}
```

Instrumentation & Measurement

Score-P

Performance measurements (profiles)

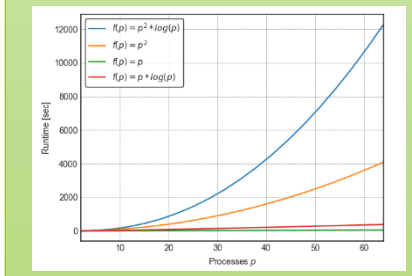
- Myapp.weak.p128.r1/profile.cubex
- Myapp.weak.p256.r1/profile.cubex
- Myapp.weak.p512.r1/profile.cubex
- Myapp.weak.p1024.r1/profile.cubex
- Myapp.weak.p2048.r1/profile.cubex

Input

Extra-P

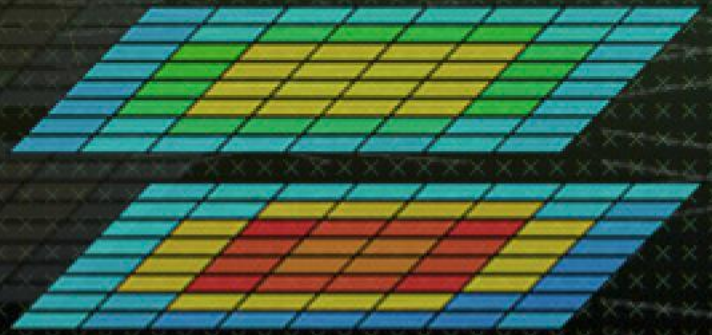
Output

Results (visual)



Results (textual)

```
Region 1: main
Model: (3) + (3.14 * x^( 2 ))
...
```



HANDS ON

Using the command line tool

HANDS ON

Extra-P command line tool

Already installed on COSMA, just do `module load extrap`

- Provides the same functionality, without visualization for use on cluster
- Usage guideline and command can be found at: https://github.com/extra-p/extrap

- 1.) Run: `extrap`
- Command Format: `extrap OPTIONS (--cube | --text | --talpas | --json | --extra-p-3) FILEPATH`
- 2.) Select input type: `extrap --text /cosma/apps/do009/shared/Extra-P/examples/one_parameter.txt`

Extra-P command line tool

- 3.) Output:

```
Callpath: compute
        Metric: time
                Measurement point: (2.00E+01) Mean: 8.19E+01 Median: 8.20E+01
                Measurement point: (3.00E+01) Mean: 1.79E+02 Median: 1.78E+02
                Measurement point: (4.00E+01) Mean: 3.19E+02 Median: 3.19E+02
                Measurement point: (5.00E+01) Mean: 5.05E+02 Median: 5.06E+02
                Measurement point: (6.00E+01) Mean: 7.25E+02 Median: 7.26E+02
                Model: -0.8897934098062804 + 0.20168243826499183 * x^(2)
                RSS: 3.43E+01
                Adjusted R^2: 1.00E+00
```

Callpath, kernel of the application that was measured

Metric name; either Score-P metrics (time, bytes, etc.) or custom metrics

Measurements for each input element (e.g., #processes)

Best-fit model

RSS: Residual sum of squares

Adjusted R^2 (explained previously)

Modeling sets of CUBE experiments

Extra-P Cube input description

Modeling tool expects CUBE files in the following format:

- <DIR>/<PREFIX>.<PARAMETERS>.r<REPETITION>/<FILENAME>.cubex
  - DIR, PREFIX, FILENAME – are just names, no meaning for Extra-P
  - REPETITION – number of the repetition of the experiments with same parameter values

- <PARAMETERS>:=<PARAM1><VALUE1>.….<PARAMn><VALUEn>
  - List of parameter-value-pairs separated by "."
  - PARAM – varied parameter e.g. number of processes
  - VALUE – value of the varied parameter

Extra-P Cube input description – example

- app.processes2.size8000.r1
- app.processes2.size8000.r2
- app.processes2.size8000.r3
- app.processes2.size8000.r4
- app.processes2.size16000.r1
- …
- app.processes2.size40000.r1
- app.processes4.size8000.r1
- …
- app.processes4.size40000.r4
- app.processes8.size8000.r1
- …
- app.processes8.size40000.r4
- app.processes16.size8000.r1
- …
- app.processes16.size40000.r4
- app.processes32.size8000.r1
- …
- app.processes32.size40000.r4

HANDS ON

Collecting a measurement series: sbatch script

```
#!/bin/bash
#SBATCH --job-name="BTMZ_B"          # Job name
##SBATCH --output=slurm-%A.out       # Job output file
#SBATCH --ntasks=2                   # Total number of MPI processes
#SBATCH --ntasks-per-node=4          # MPI processes per node
#SBATCH --cpus-per-task=4            # OMP threads per process
#SBATCH --time=00:05:00              # Maximum run time (hh:mm:ss)
#SBATCH --exclusive                  # No sharing of compute nodes
#SBATCH --partition=bluefield1       # DINE compute nodes
#SBATCH --account=do009
#SBATCH --array=1-5

# set up environment
module purge
module load gnu_comp openmpi
module load scorep

# measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_sum.p$SLURM_NTASKS.r$SLURM_ARRAY_TASK_ID
export SCOREP_FILTERING_FILE=../config/scorep.filt

# benchmark configuration
set -x
export OMP_NUM_THREADS=4
time -p mpiexec ./bt-mz_B.x
```

We want to vary the number of processes

Used here to control repetitions

Experiment directory name follows naming scheme

Copy file from:
`/cosma/apps/do009/shared/Extra-P/scorep-scaling.sbatch`

HANDS ON

Collecting a measurement series: start script

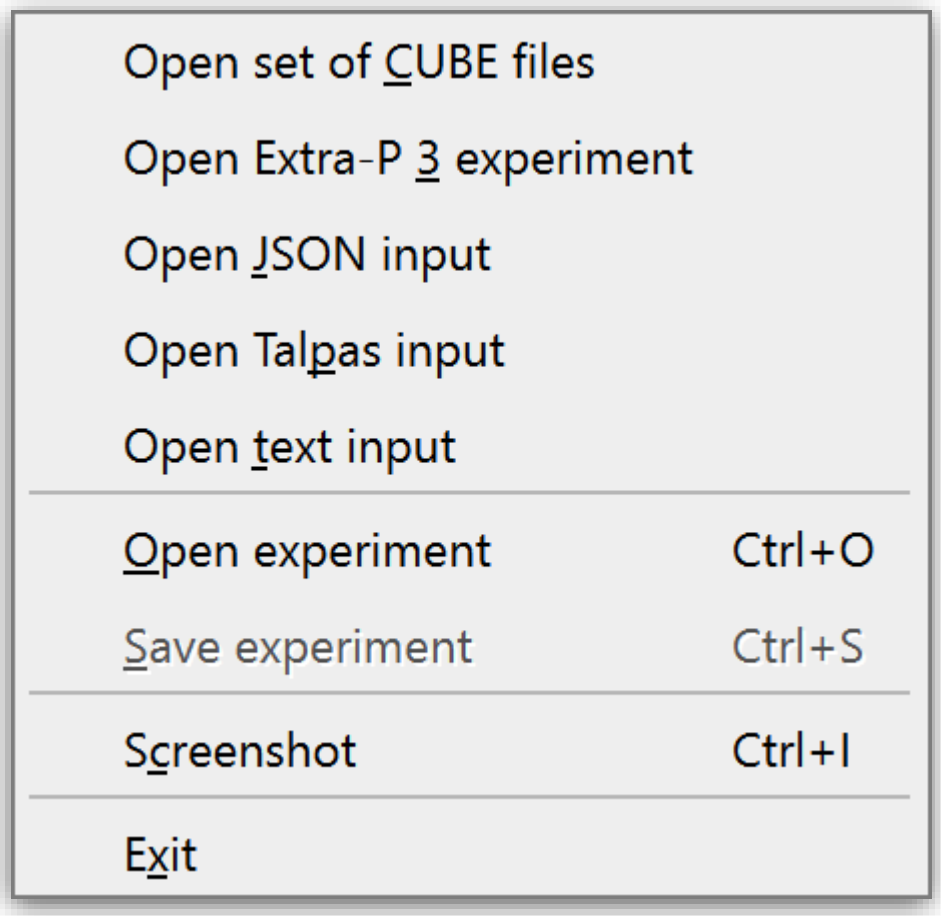
- We want to vary the number of processes

```
#!/bin/bash
for ntasks in 2 4 6 8 10; do
    sbatch --ntasks=$ntasks scorep-scaling.sbatch
done
```

Arguments take precedence over #SBATCH directives

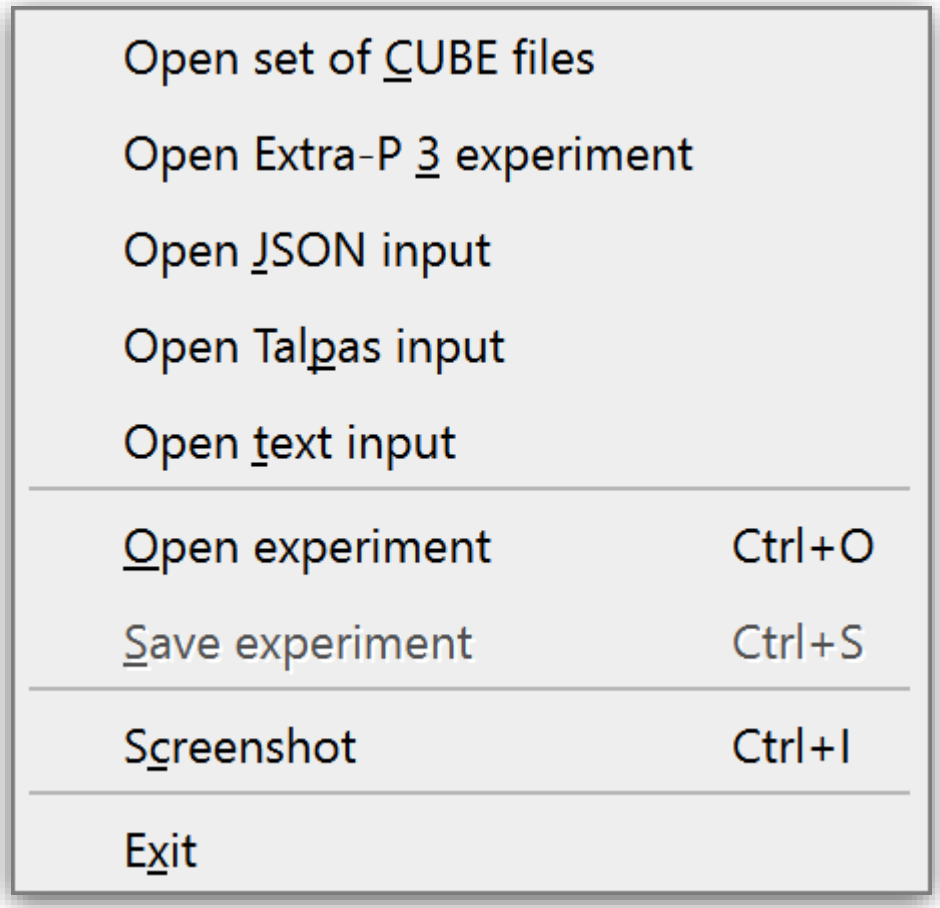
Copy file from:
`/cosma/apps/do009/shared/Extra-P/start-scaling.sh`

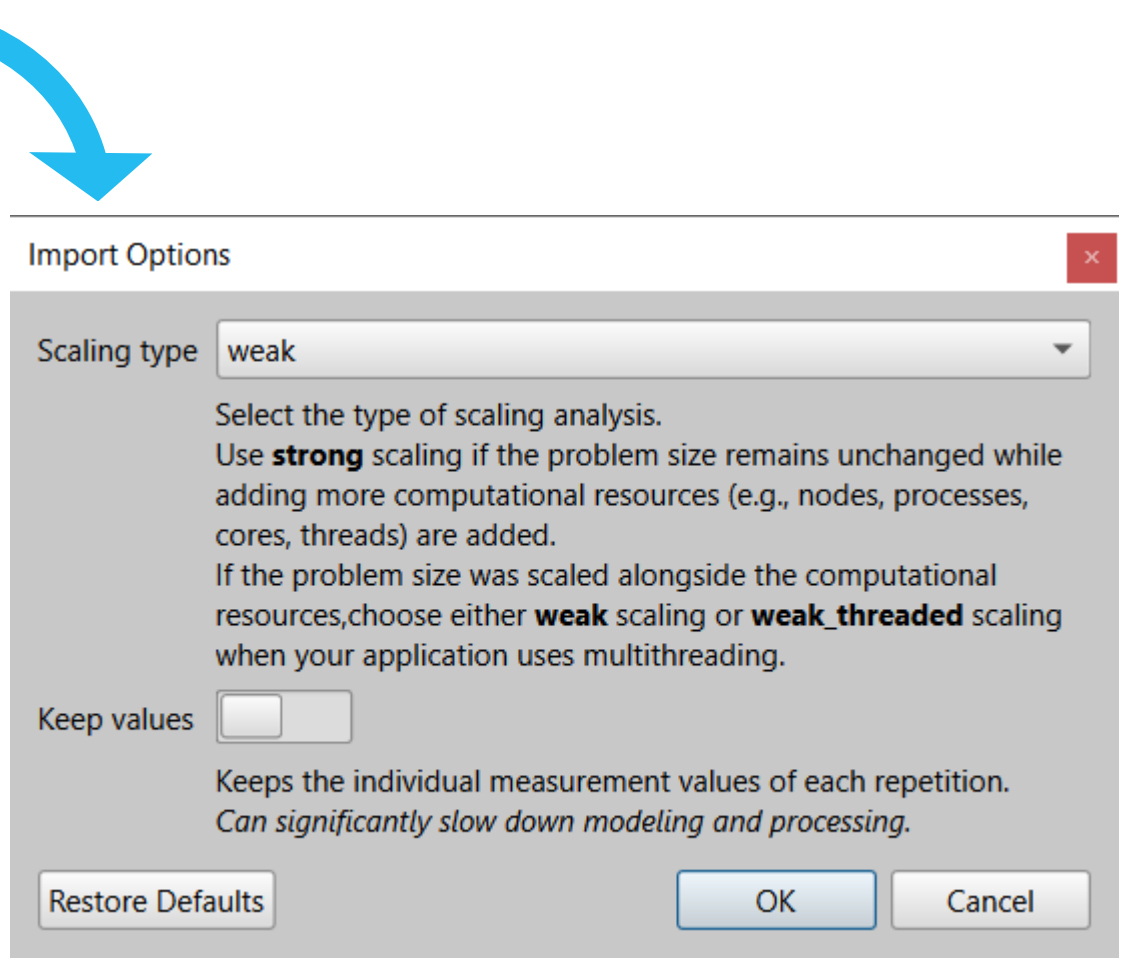
Choose input file



Select input file in the GUI

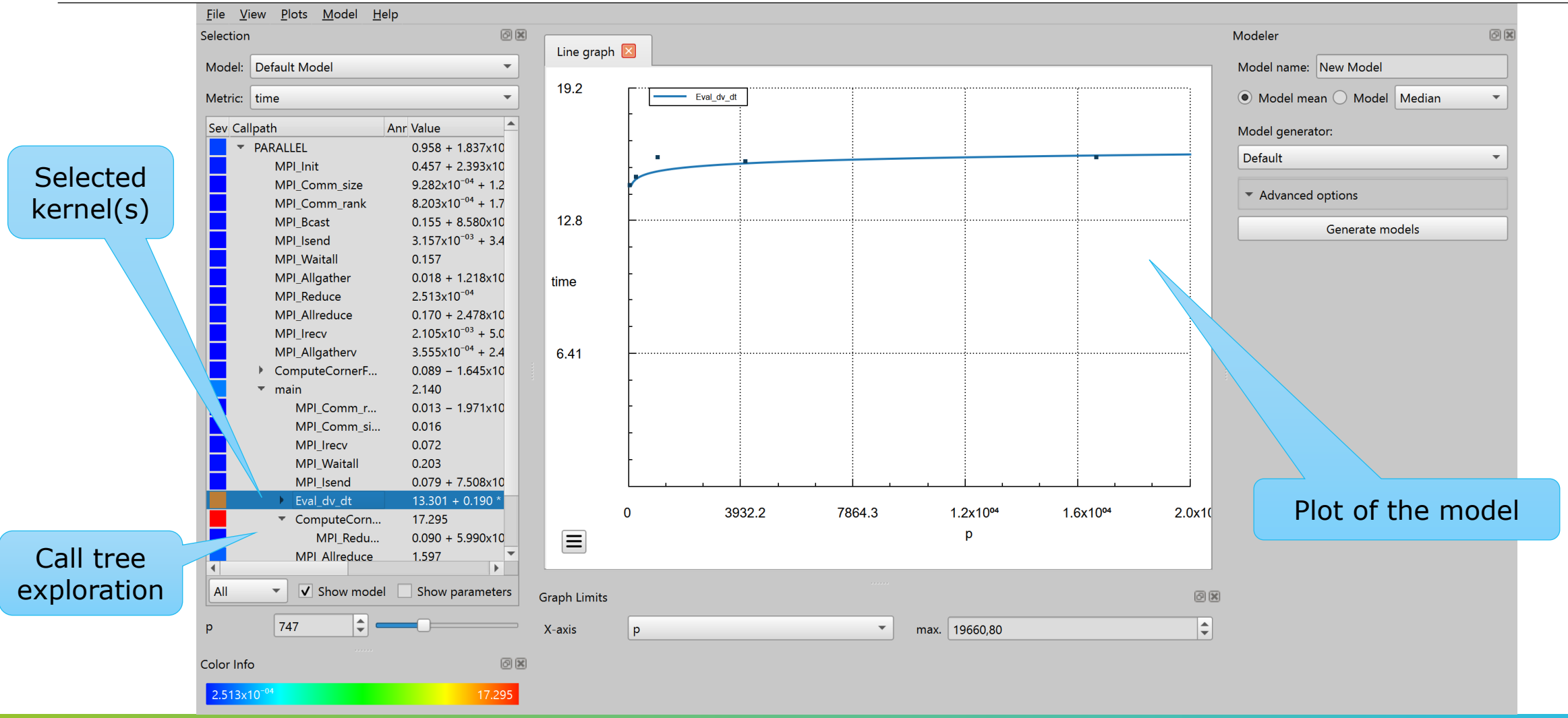
Extra-P Cube input





Visualization with Extra-P

Extra-P user interface



Selected kernel(s)

Call tree exploration

Plot of the model

Extra-P call tree view

Metric selection

Model selection

Call tree exploration

Model

Quality of fit metrics: Residual sum of squares and Adjusted R^2

Impact of each kernel on the metric at the selected process count compared to the other kernels

Asymptotic behavior

Model: Default Model

Metric: time

Sev	Call...	Anr	Value	RSS	Adj. R^2	SMAPE	RE
	PARALLEL		$0.958 + 1.837x10^{-05} * p * \log_2(p)$	0.092	0.990	2.903	0.029
	MPI_Init		$0.457 + 2.393x10^{-10} * p^{9/4}$	4.871x10...	0.998	0.829	$8.259x10^{-03}$
	MPI_Comm_size		$9.282x10^{-04} + 1.232x10^{-09} * p * \log_2(p)$	8.406x10...	0.981	1.096	0.011
	MPI_Comm_rank		$8.203x10^{-04} + 1.796x10^{-10} * p * \log_2(p)$	1.241x10...	0.987	0.155	$1.552x10^{-03}$
	MPI_Bcast		$0.155 + 8.580x10^{-11} * p^{7/3}$	6.011x10...	0.997	3.815	0.038
	MPI_Isend		$3.157x10^{-03} + 3.410x10^{-05} * \log_2(p)$	1.641x10...	0.568	1.365	0.014
	MPI_Waitall		0.157	1.788x10...	1	2.934	$3.950x10^{-03}$
	MPI_Allgather		$0.018 + 1.218x10^{-09} * p^{7/4}$	9.760x10...	0.980	5.835	0.058
	MPI_Reduce		$2.513x10^{-04}$	1.119x10...	1	13.626	0.261
	MPI_Allreduce		$0.170 + 2.478x10^{-04} * p^{1/3} * \log_2(p)$	7.463x10...	0.790	5.390	0.054
	MPI_Irecv		$2.105x10^{-03} + 5.098x10^{-05} * \log_2(p)$	2.552x10...	0.693	2.447	0.025
	MPI_Allgatherv		$3.555x10^{-04} + 2.418x10^{-07} * p^{5/4}$	2.974x10...	0.997	6.479	0.068
	ComputeCornerForces		$0.089 - 1.645x10^{-04} * \log_2(p)$	1.201x10...	-0.124	0.469	$4.685x10^{-03}$
	main		2.140	4.007x10...	1	1.264	$6.078x10^{-03}$
	MPI_Comm_rank		$0.013 - 1.971x10^{-05} * \log_2(p)$	9.469x10...	0.319	0.280	$2.799x10^{-03}$
	MPI_Comm_size		0.016	1.771x10...	1	0.341	$5.654x10^{-04}$
			0.072	9.185x10...	1	1.582	0.012
	MPI_Waitall		0.203	0.019	1	22.235	0.239
	MPI_Isend		$0.079 + 7.508x10^{-04} * \log_2(p)$	5.459x10...	0.695	1.033	0.010
	Eval_dv_dt		$13.301 + 0.190 * \log_2(p)$	0.803	0.392	2.211	0.022
	ComputeCornerForces		17.295	0.038	1	0.340	$7.466x10^{-04}$
	MPI_Reduce		$0.090 + 5.990x10^{-09} * p^{5/4} * \log_2(p)$	4.877x10...	0.684	2.470	0.025
	MPI_Allreduce		1.597		1	4.829	0.084
	MPI_Finalize		2.734x10		1	28.617	0.235
	MPI_Recv		1.304x10		0.829	39.132	0.529

All

Show model Show parameters

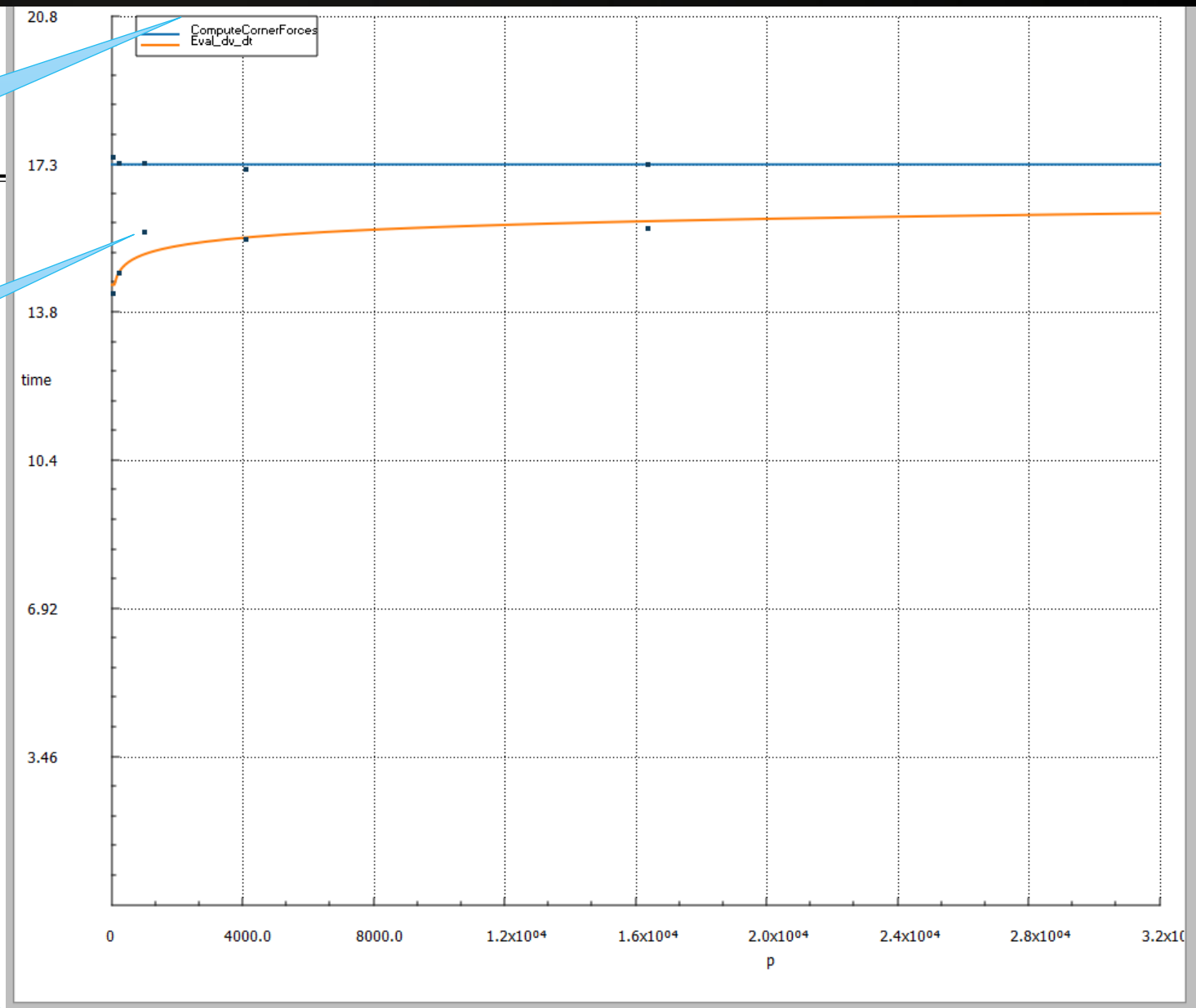
p 747

Extra-P model view

Models selected in the Call path view

Measurement values

X axis scale control for prediction of behavior at other process counts



Modeling measurements from a text file

Extra-P input in text form

- Useful when no CUBE files are available or when modeling a small data set

```
PARAMETER p
POINTS 1000 2000 4000 8000 16000
METRIC metric1
REGION region1
DATA 1 1 1 1 1
DATA 4 4 4 3.99 4.01
DATA 16 15.999 16.01 16.01 15.99
DATA 64 64 64 64.01 63.99
DATA 256.01 255.99 256 256
```

Parameter name
This name will be used in the GUI as well as in the textual output

Measurement points
Use at least 5, but in general the more the better

Metric name

Region name
Both used to determine the output Cube file hierarchical structure and identify separate data sets

Data points
Each row corresponds to a point; all values in a row are considered repeat measurements of the same experiment

Extra-P input as JSON lines

- Useful when you do not want to use CUBE files
  - Easy to generate with your own scripts
- Each line of the file is a JSON object
  - Describes one measurement value

```
{"params": {"<parameter1>": 0, "…": "…"}, "value": 0.0,
"callpath": "<callpath>", "metric": "<metric>" } ↲
```

One line

Example

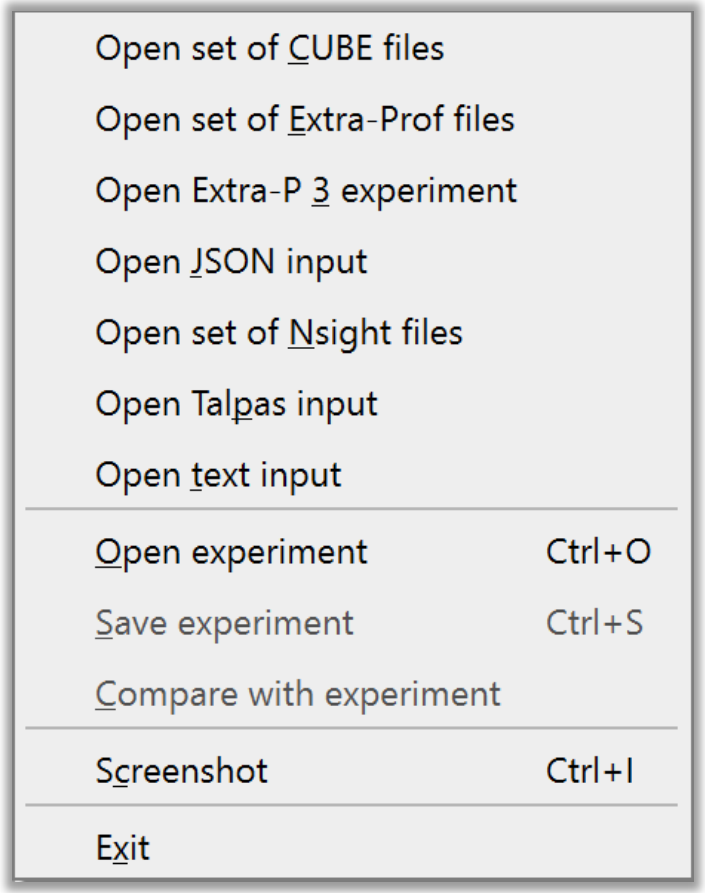
```
{"params":{"x":1,"y":1}, "metric":"metr", "callpath":"test", "value":2.03}
{"params":{"x":1,"y":2}, "metric":"metr", "callpath":"test", "value":3.02}
{"params":{"x":1,"y":3}, "metric":"metr", "callpath":"test", "value":4.01}
{"params":{"x":1,"y":4}, "metric":"metr", "callpath":"test", "value":5.02}
{"params":{"x":1,"y":5}, "metric":"metr", "callpath":"test", "value":6.01}
[…]
```

Comparing models between experiments

HANDS ON

Open/import first experiment

Beta



Import measurements

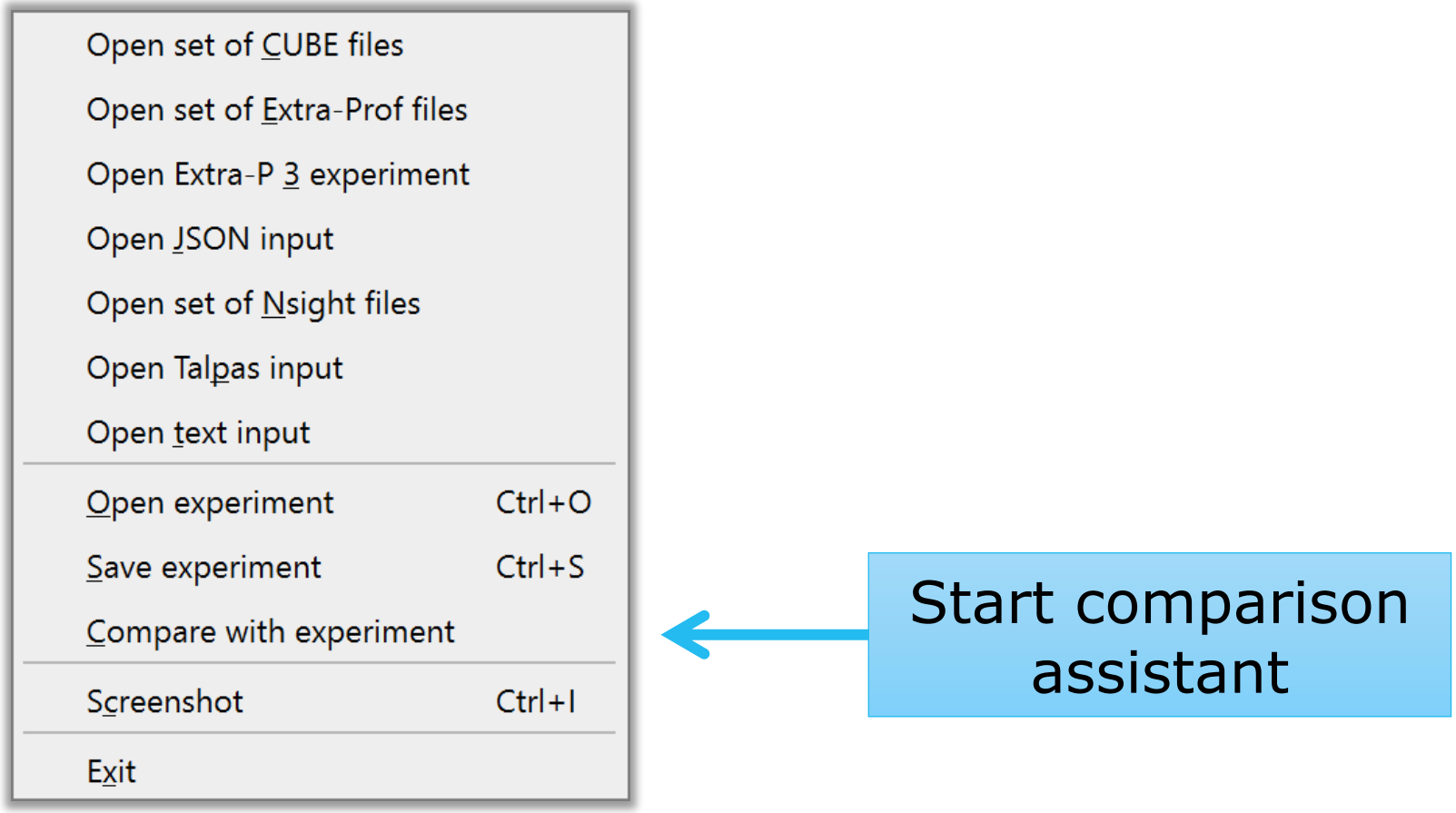
Open stored experiment

Example files are in:
`/cosma/apps/do009/shared/Extra-P/examples/CPU-GPU-comparison`

HANDS ON

Start comparison

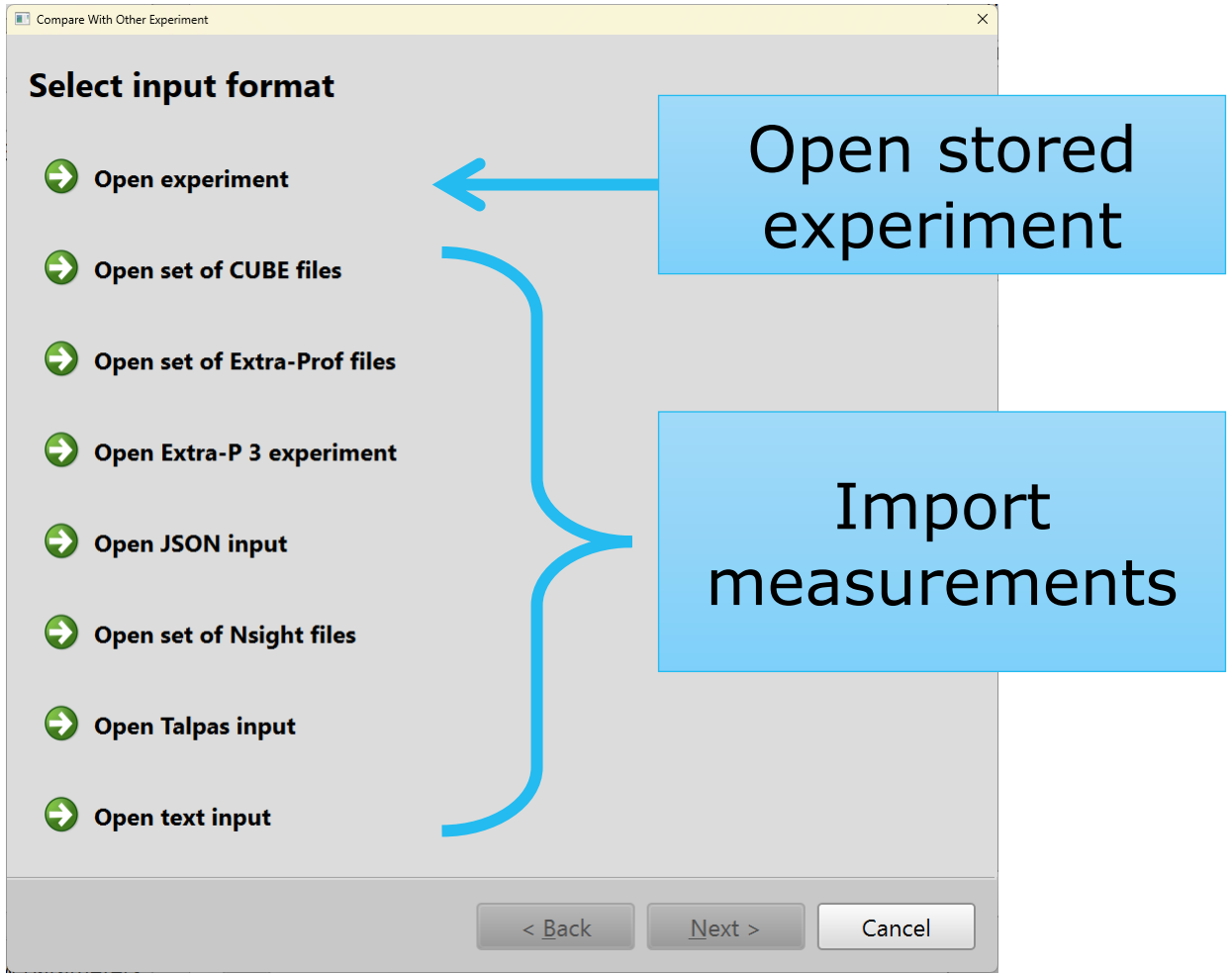
Beta



Start comparison assistant

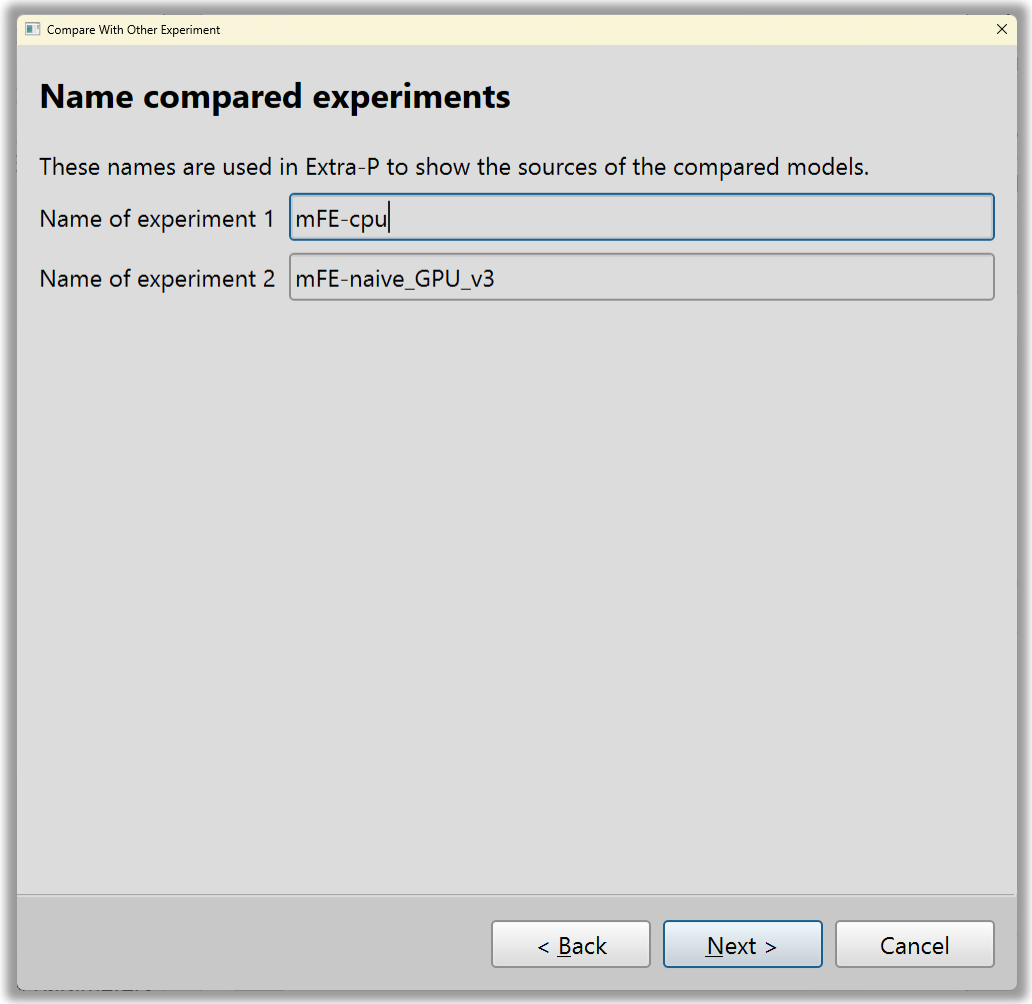
Open/import second experiment

Beta



Follow the comparison assistant

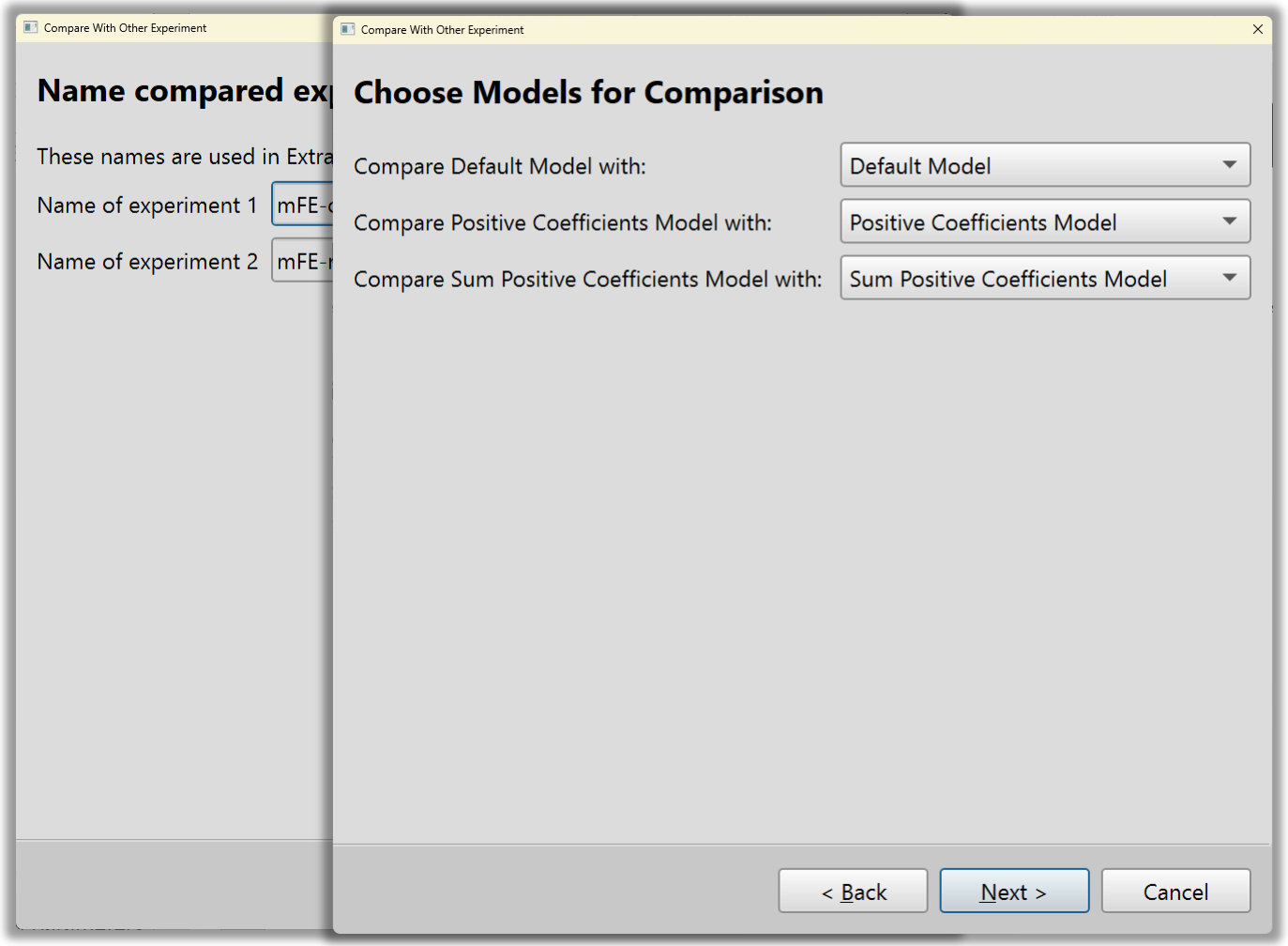
Beta



Tip: short names improve readability

Follow the comparison assistant

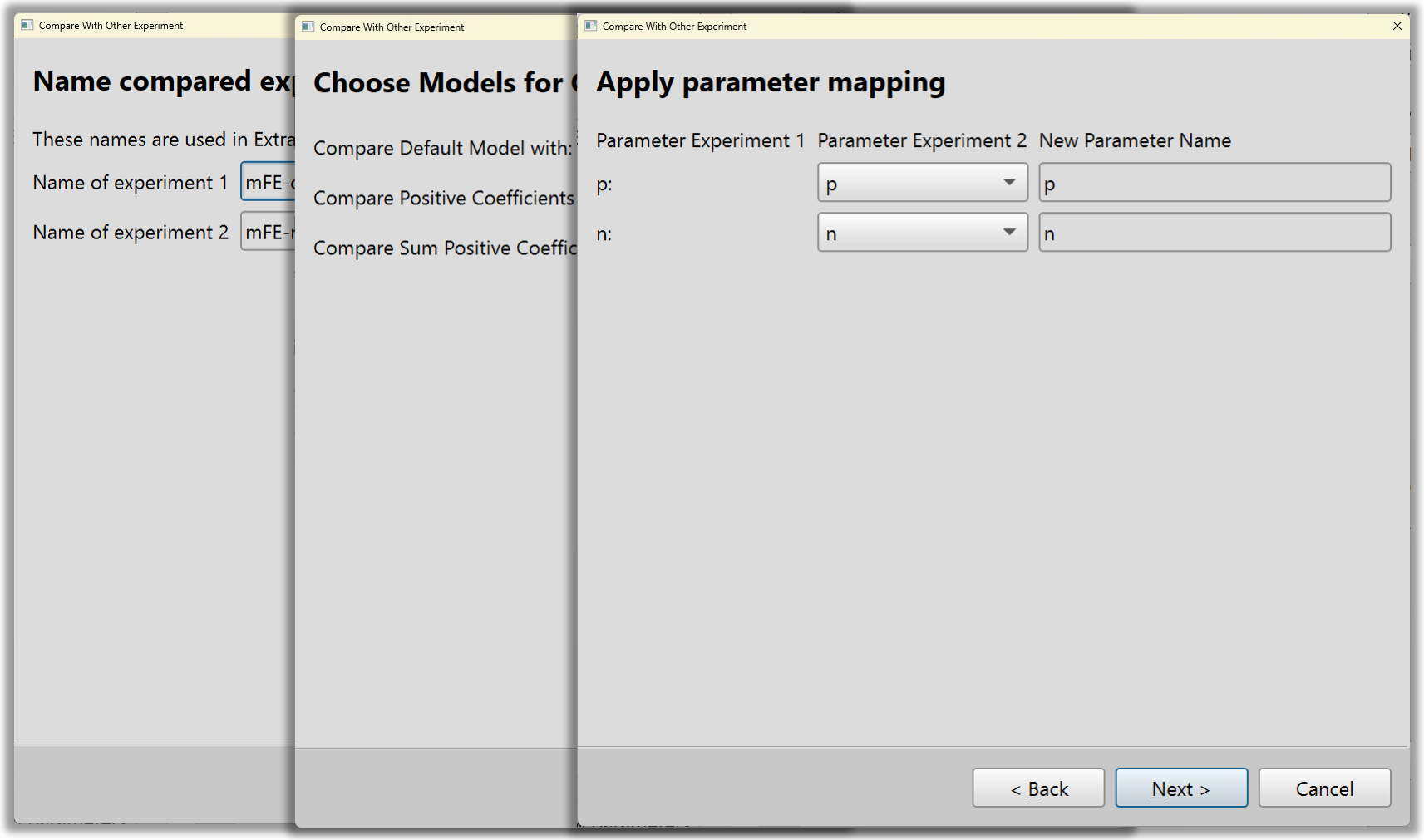
Beta



Model sets with identical name are matched automatically

Follow the comparison assistant

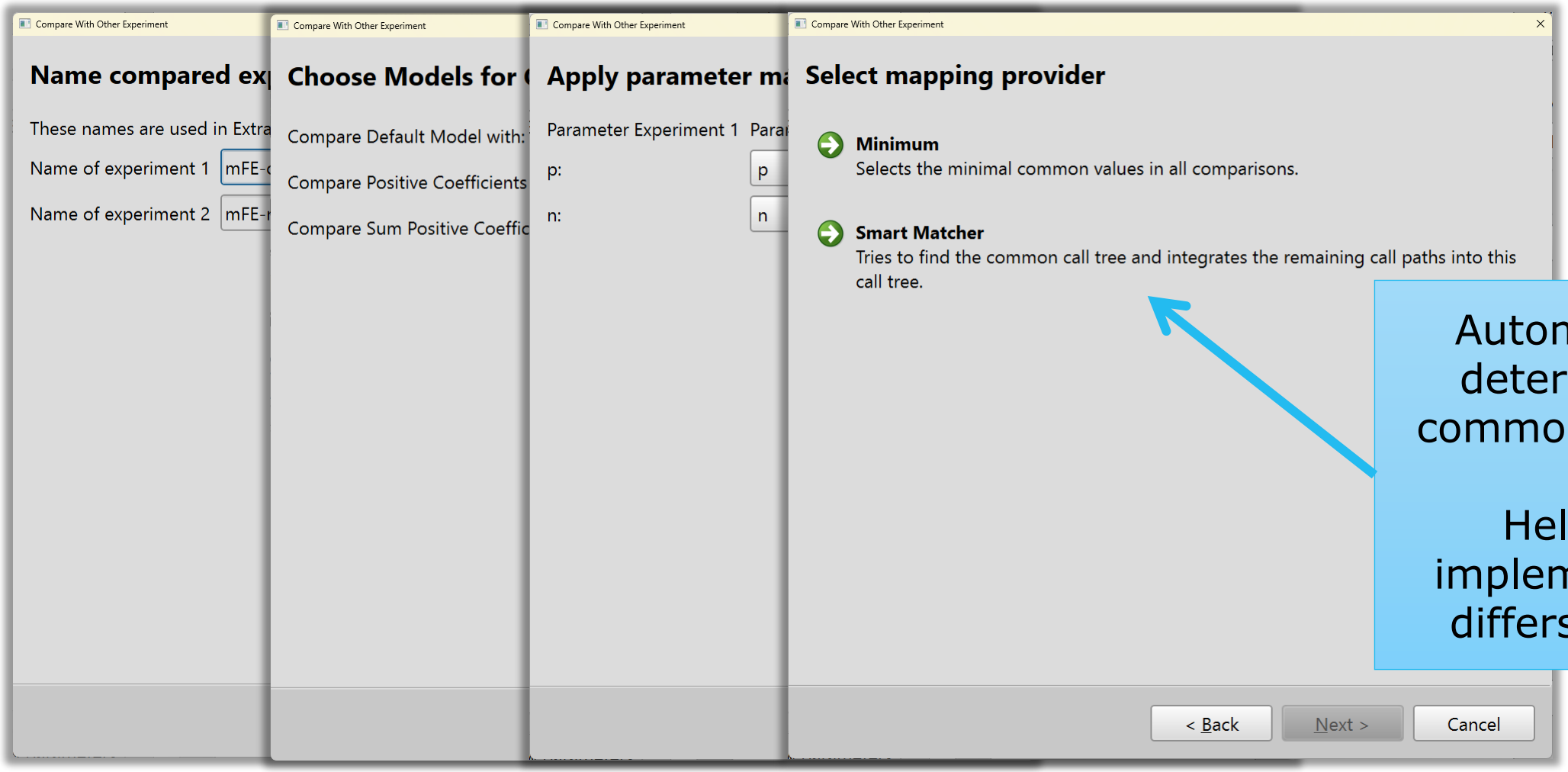
Beta



Tip: use identical names to avoid confusion

Follow the comparison assistant

Beta

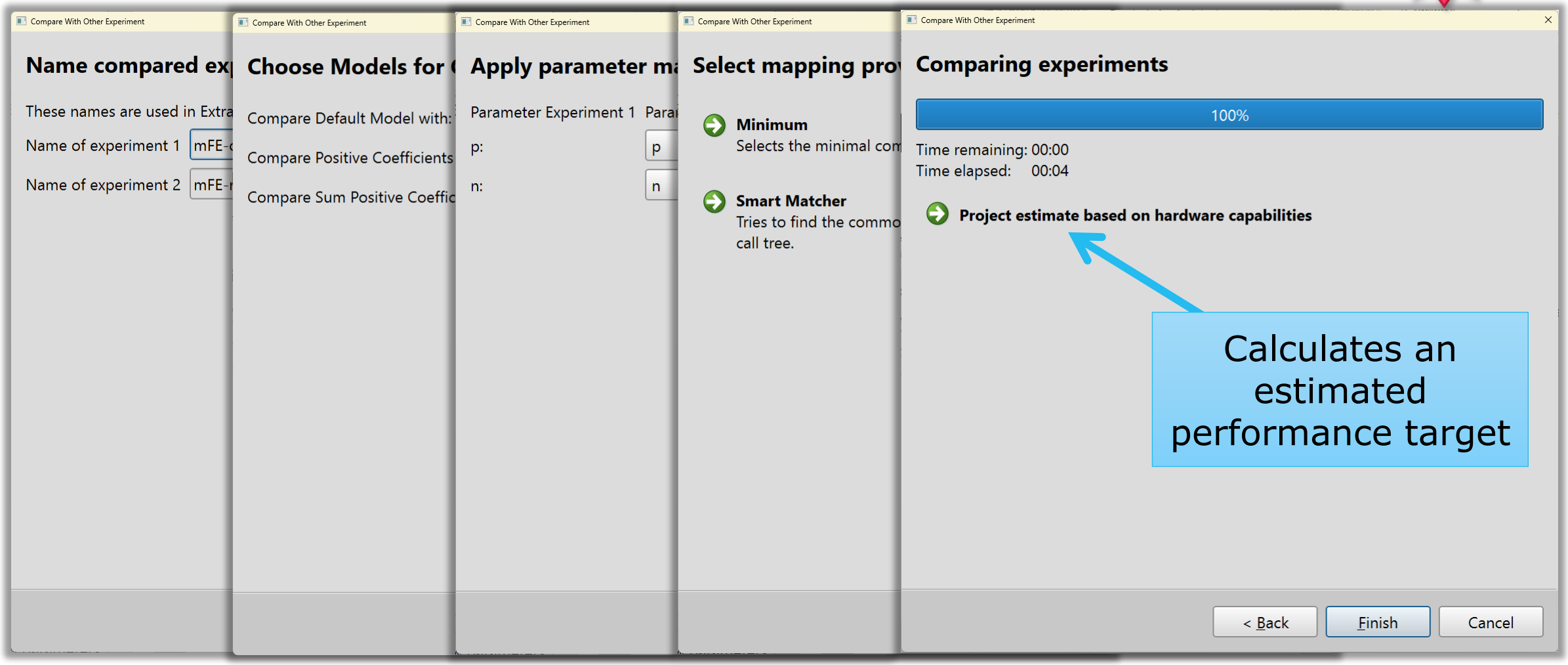


Automatically determines a common call tree

Helpful if implementation differs slightly

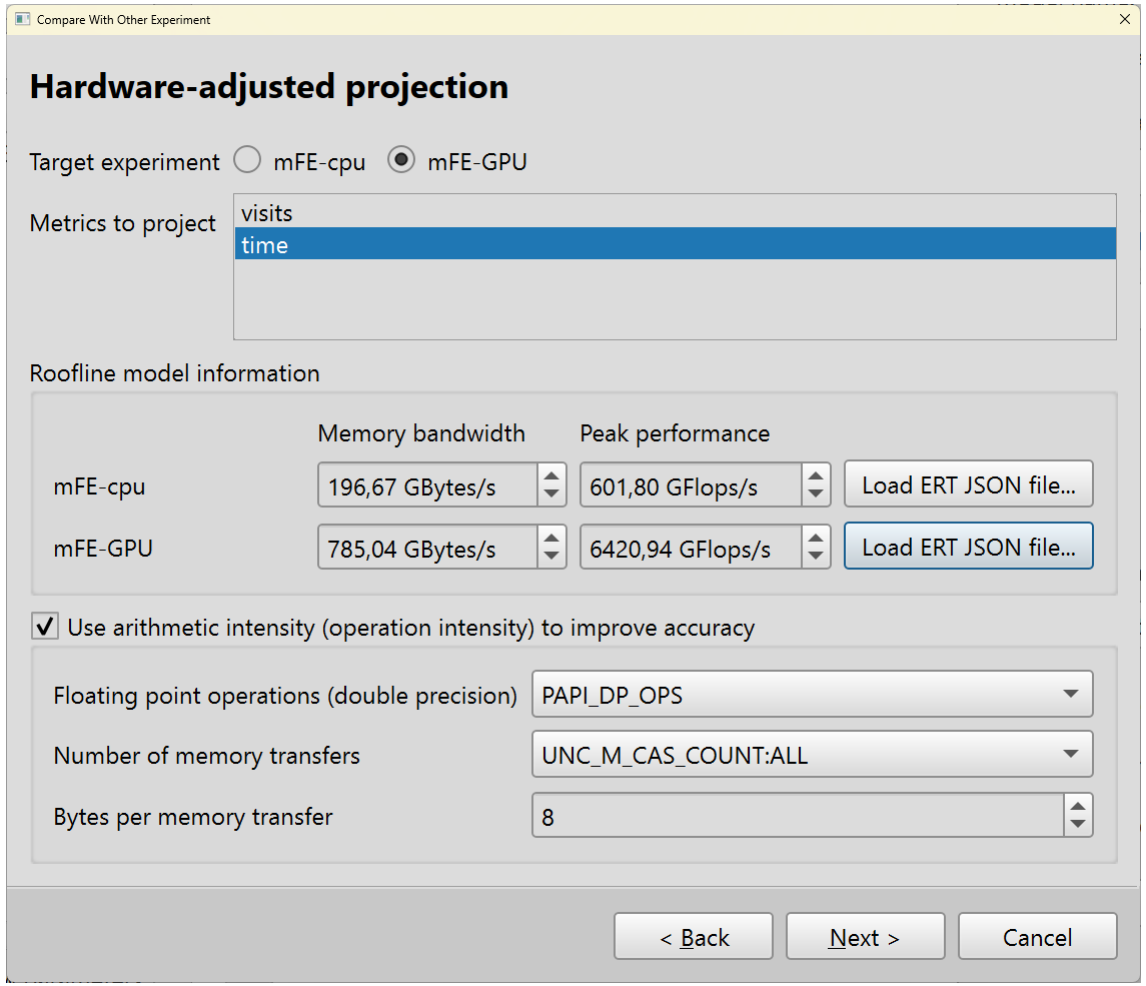
Follow the comparison assistant

Beta



Project expected improvement based on hardware capability

Beta



Uses roofline models to estimate improvement

The roofline model info can be determined using the empirical roofline toolkit

Interactive comparison of CPU and GPU variants

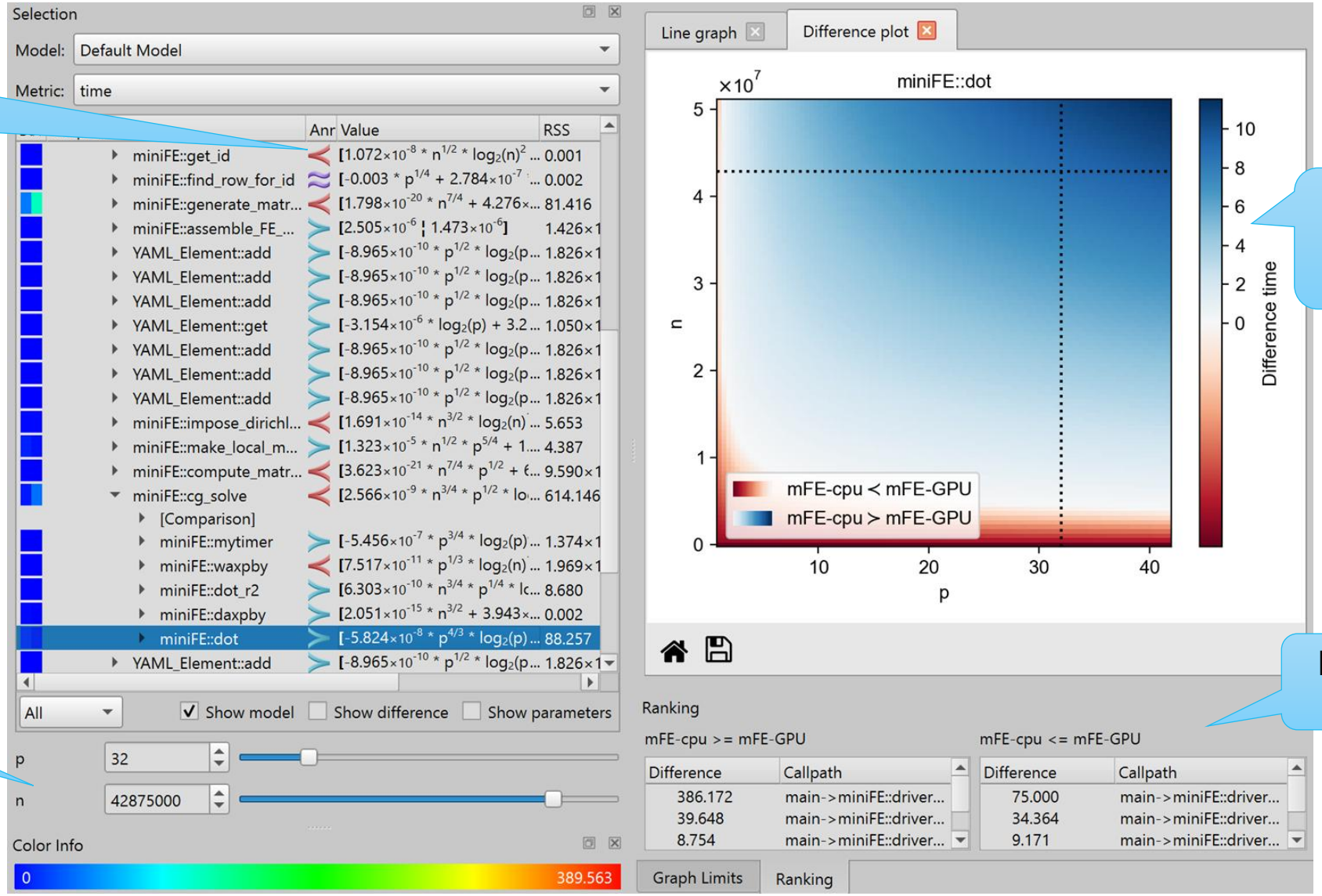
Beta

Comparison indicator

Select comparison point

Difference of behavior for selected call path

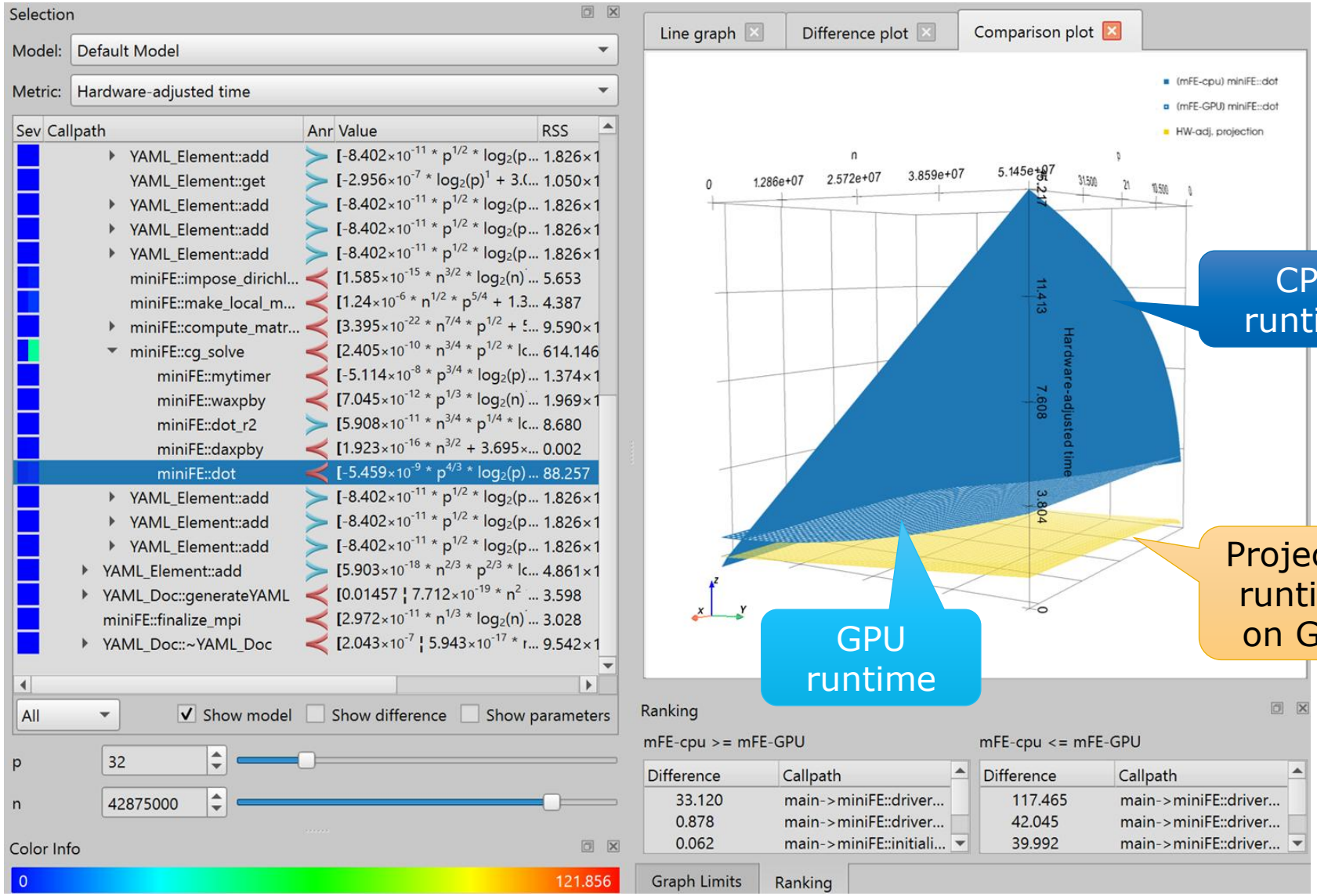
Ranking best and worst call paths



Interactive comparison of CPU and GPU variants with projected target

Projected runtime

Beta



- What additional features would you like to see?
- Did you find any bugs?

You can contact us via email: *extra-p-support@lists.parallel.informatik.tu-darmstadt.de*

Or on GitHub using the issues tool: *https://github.com/extra-p/extrap*