Extra-P:
Insightful Automatic Performance Modeling

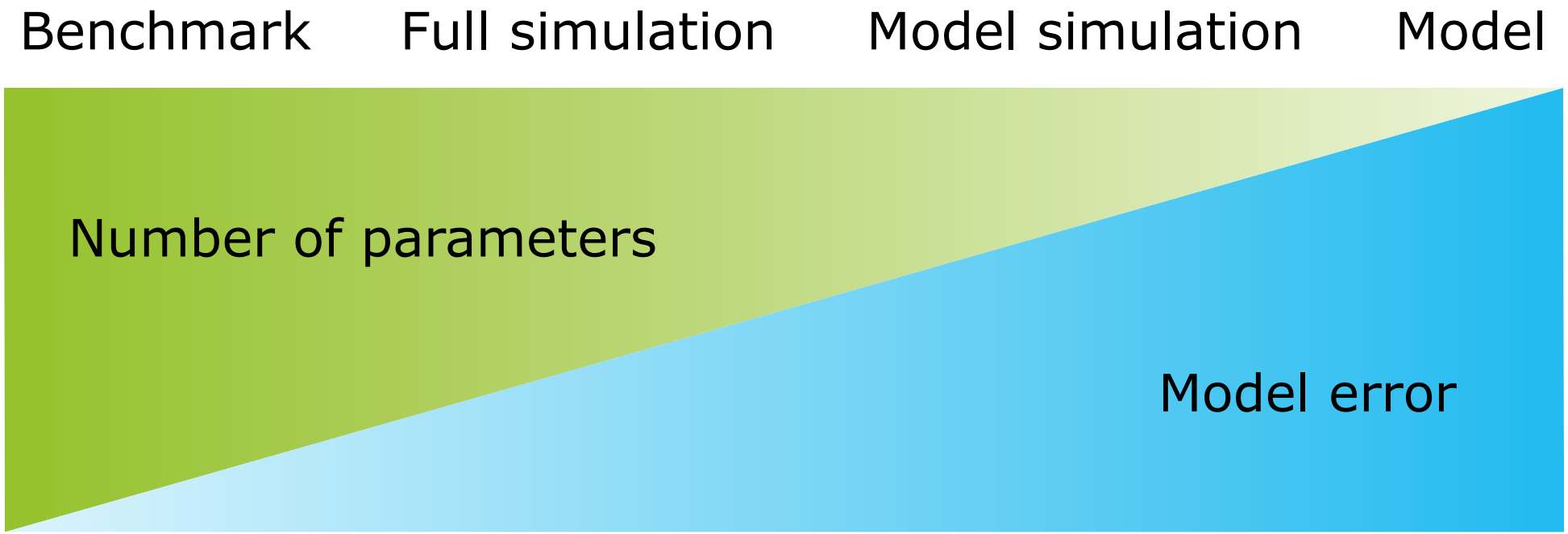
TECHNISCHE UNIVERSITÄT DARMSTADT

Alexander Geiß[1], Marcus Ritter[1], Benedikt Naumann[1],
Alexandru Calotoiu[2], Torsten Hoefler[2], and Felix Wolf[1]

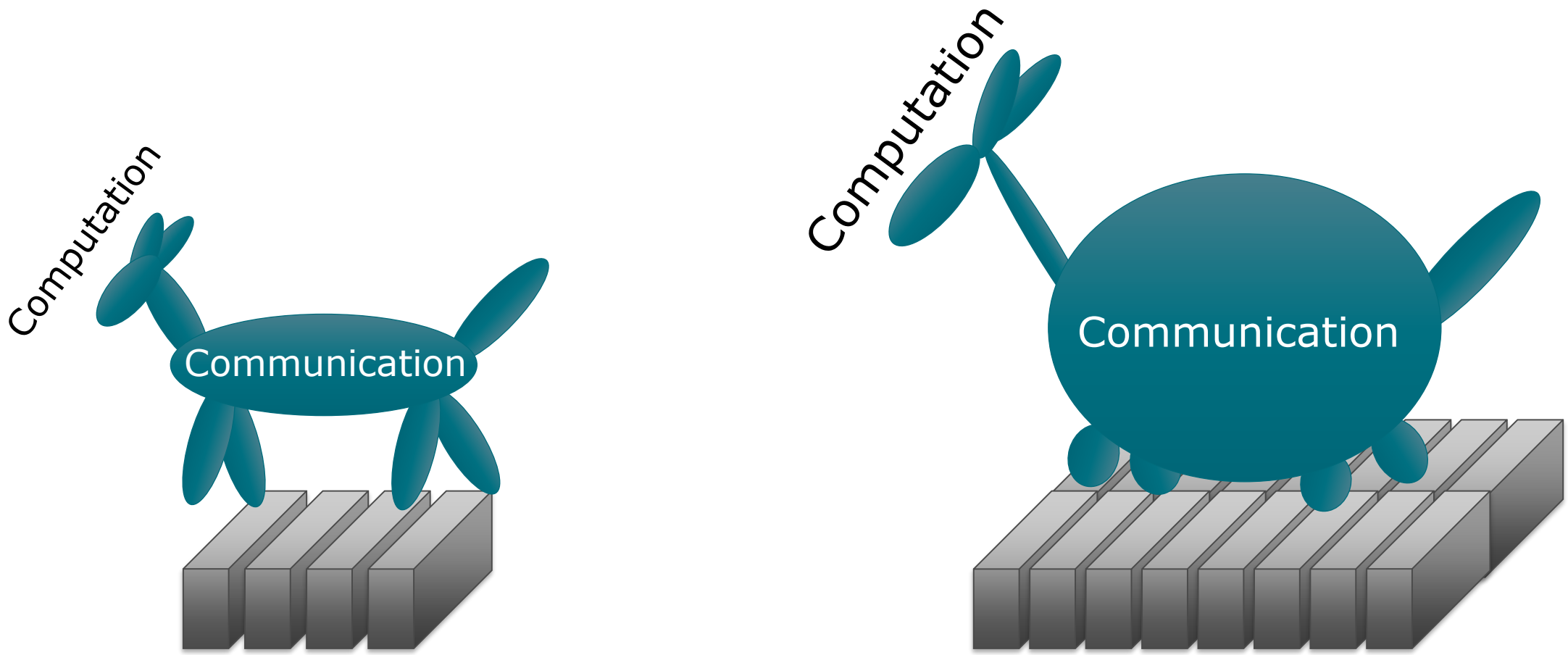
ETH zürich

[1] TU Darmstadt , [2] ETH Zürich

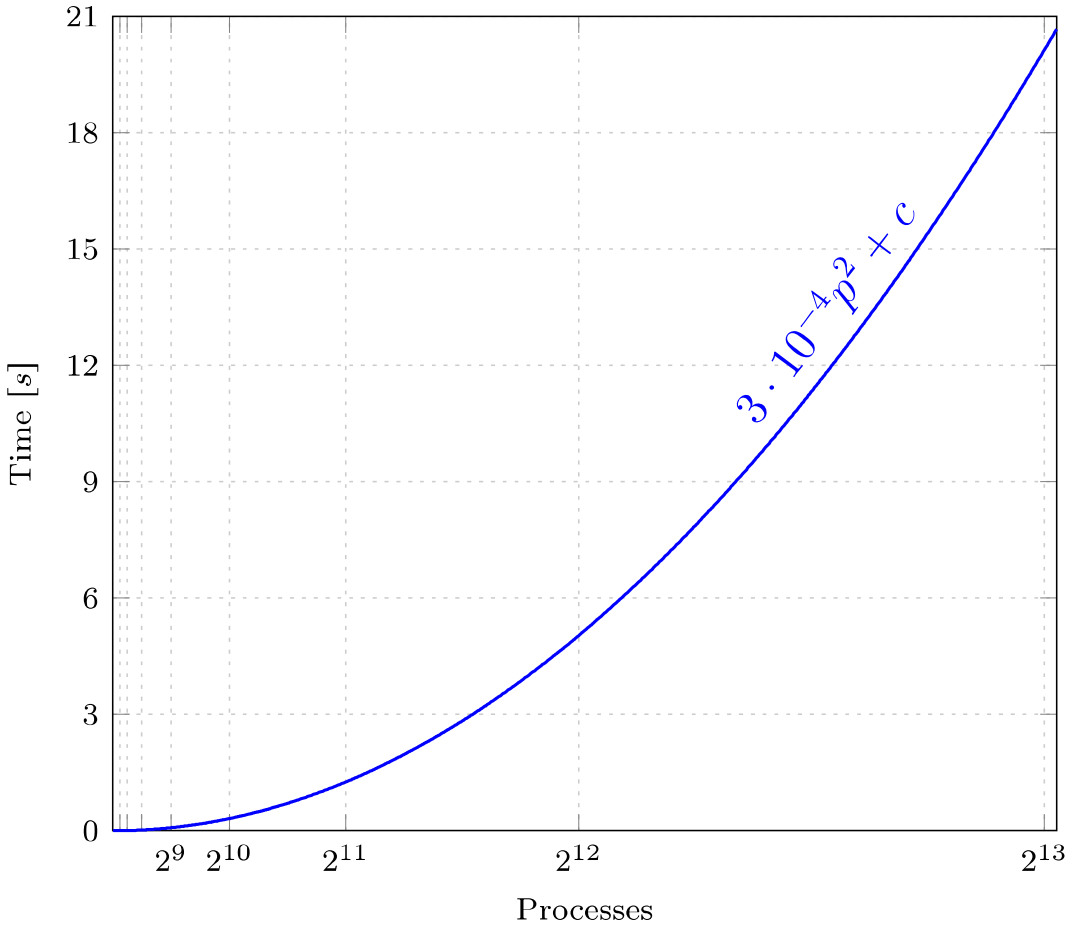
Spectrum of performance analysis methods



Motivation - latent scalability bugs



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

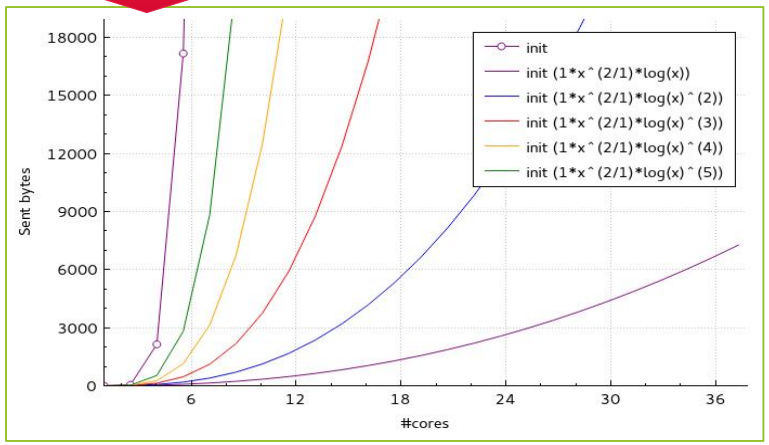
Instrumentation

- All functions

Performance measurements

Extra-P

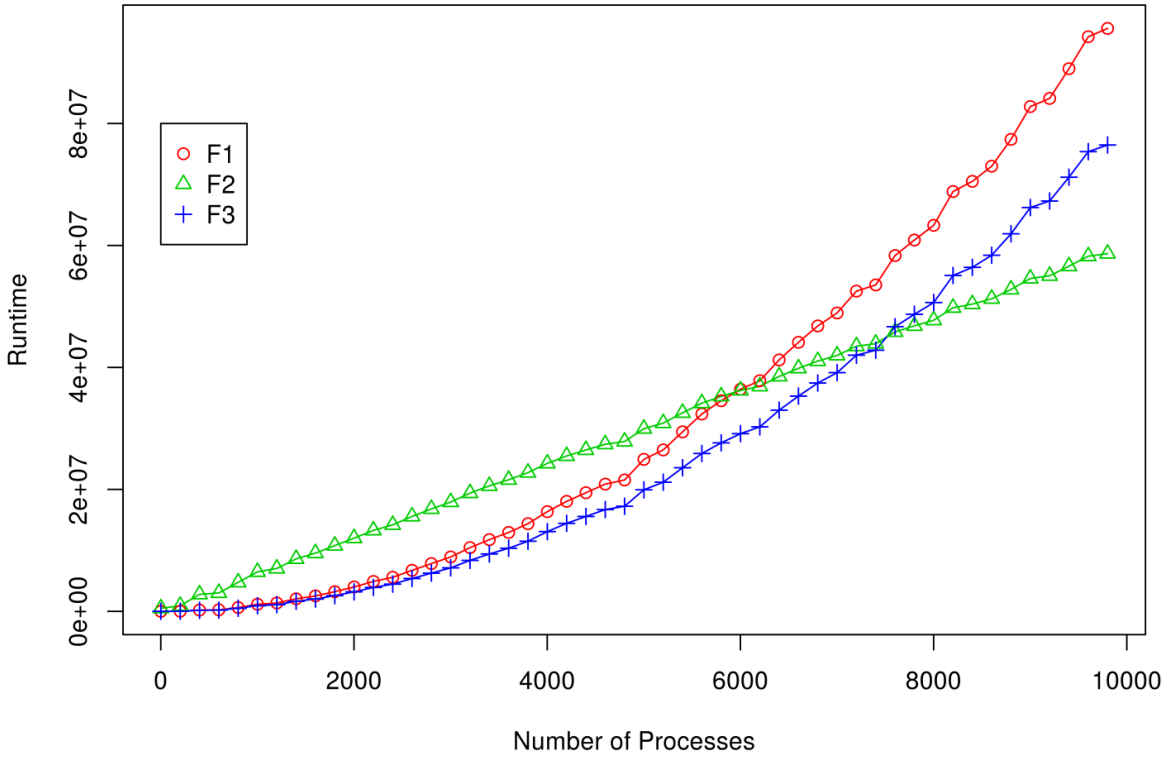
Output



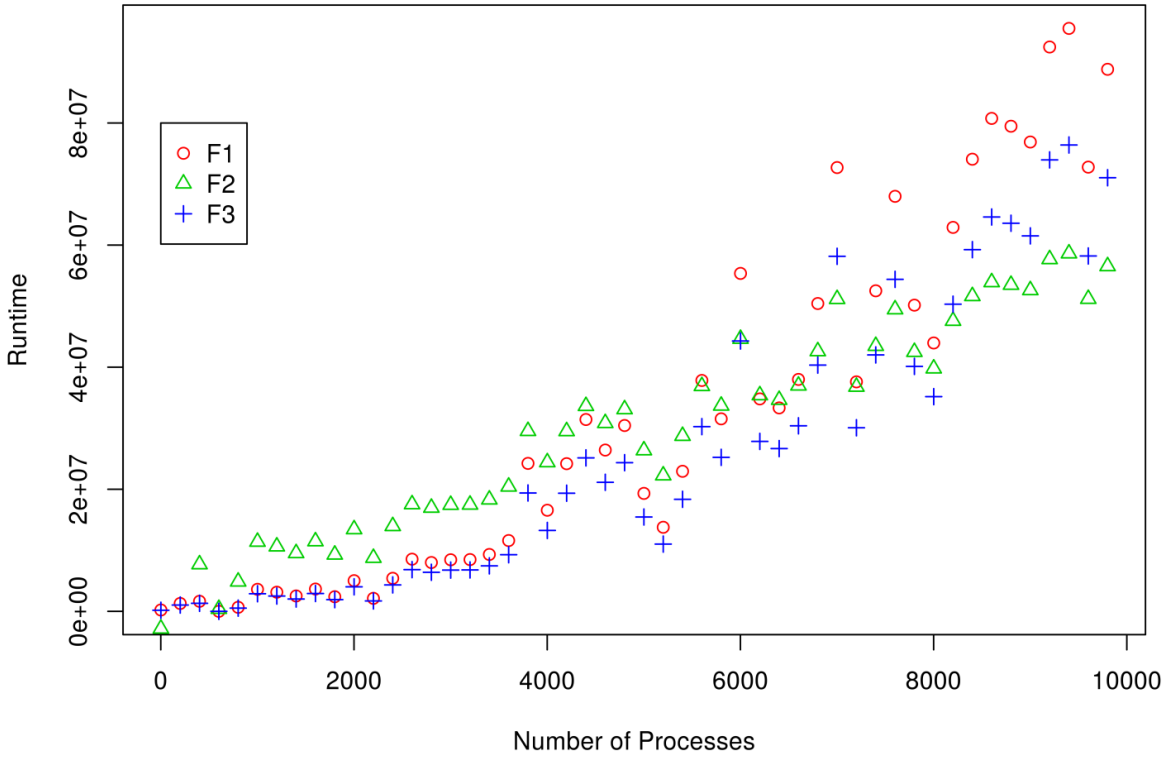
Human-readable performance models of all functions (e.g., $t(p) = c_1 \cdot \log(p) + c_2$)

Primary focus on scaling trend

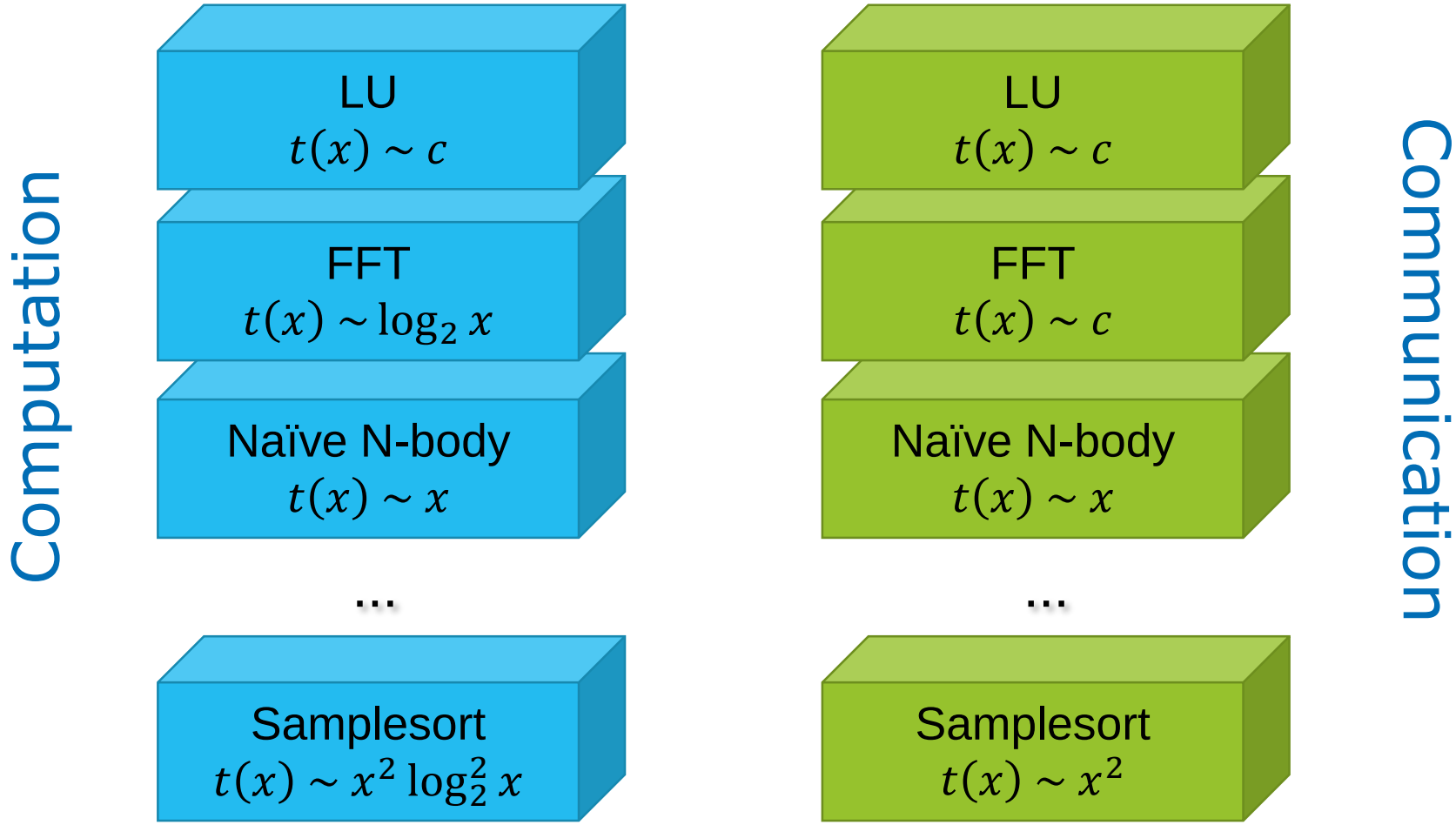
Common performance analysis chart in a paper



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

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

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

Performance model normal form

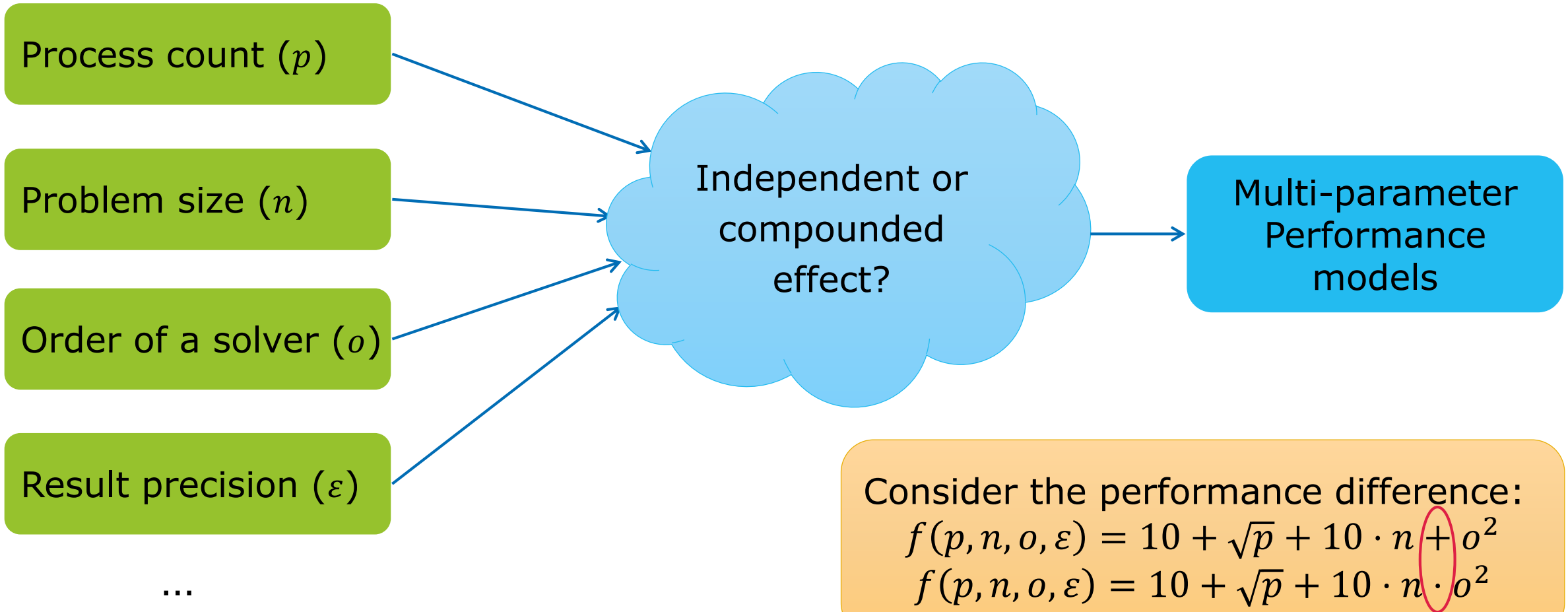
$$\in \mathbb{N}$$

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

$$c_1 \cdot \log x + c_2 \cdot x$$
$$c_1 \cdot \log x + c_2 \cdot x \cdot \log x$$
$$c_1 \cdot \log x + c_2 \cdot x^2$$
$$c_1 \cdot \log x + c_2 \cdot x^2 \cdot \log x$$
$$c_1 \cdot x + c_2 \cdot x \cdot \log x$$
$$c_1 \cdot x + c_2 \cdot x^2$$
$$c_1 \cdot x + c_2 \cdot x^2 \cdot \log x$$
$$c_1 \cdot x \cdot \log x + c_2 \cdot x^2$$
$$c_1 \cdot x \cdot \log x + c_2 \cdot x^2 \cdot \log x$$
$$c_1 \cdot x^2 + c_2 \cdot x^2 \cdot \log x$$

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

Fast multi-parameter performance modeling



Fast multi-parameter performance modeling

- Expanded performance model normal form

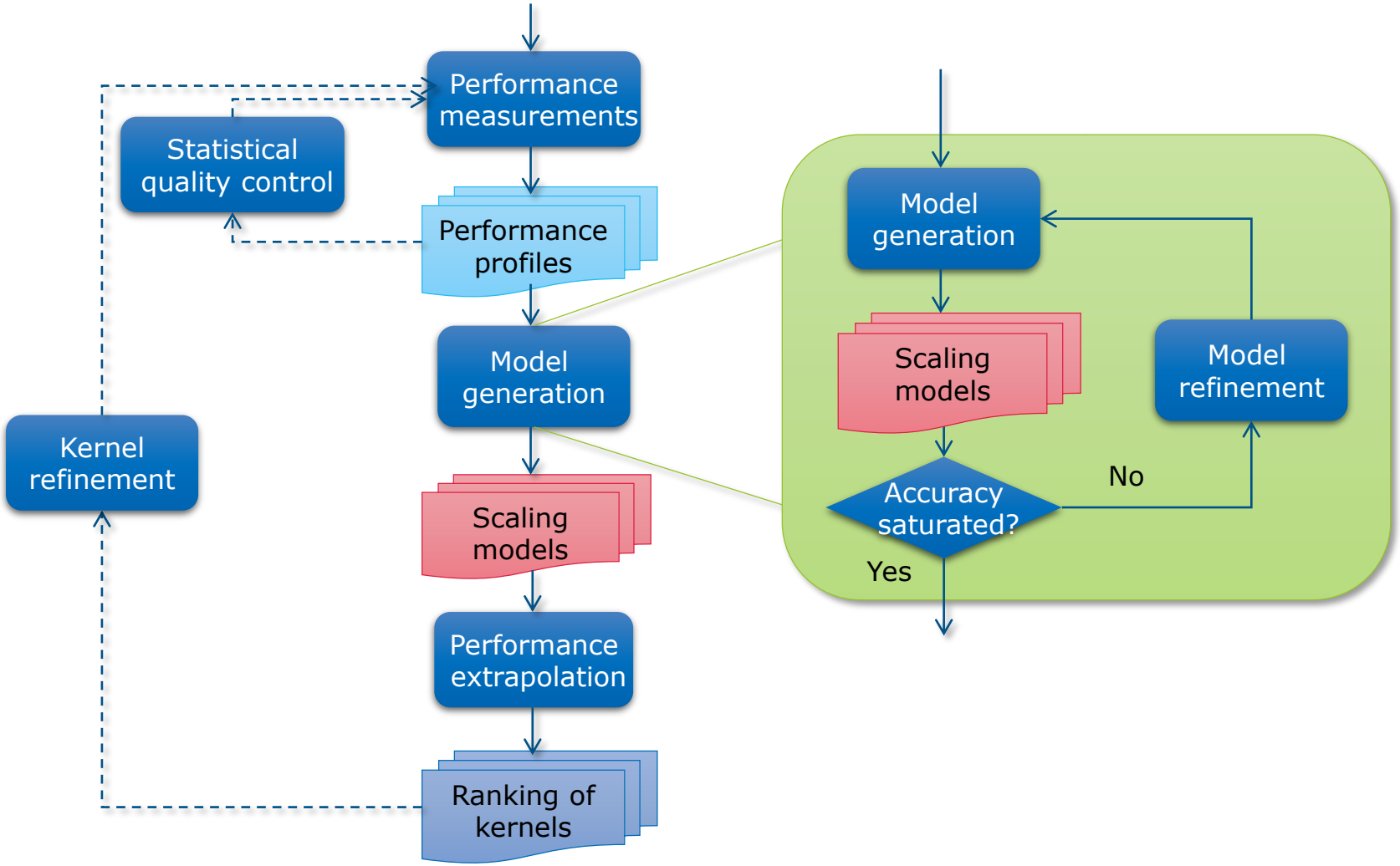
$$f(x_1, \ldots, x_m) = \sum_{k=1}^{n} c_k \prod_{l=1}^{m} x_l^{i_{kl}} \cdot log_2^{j_{kl}}(x_l)$$

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

Model candidates

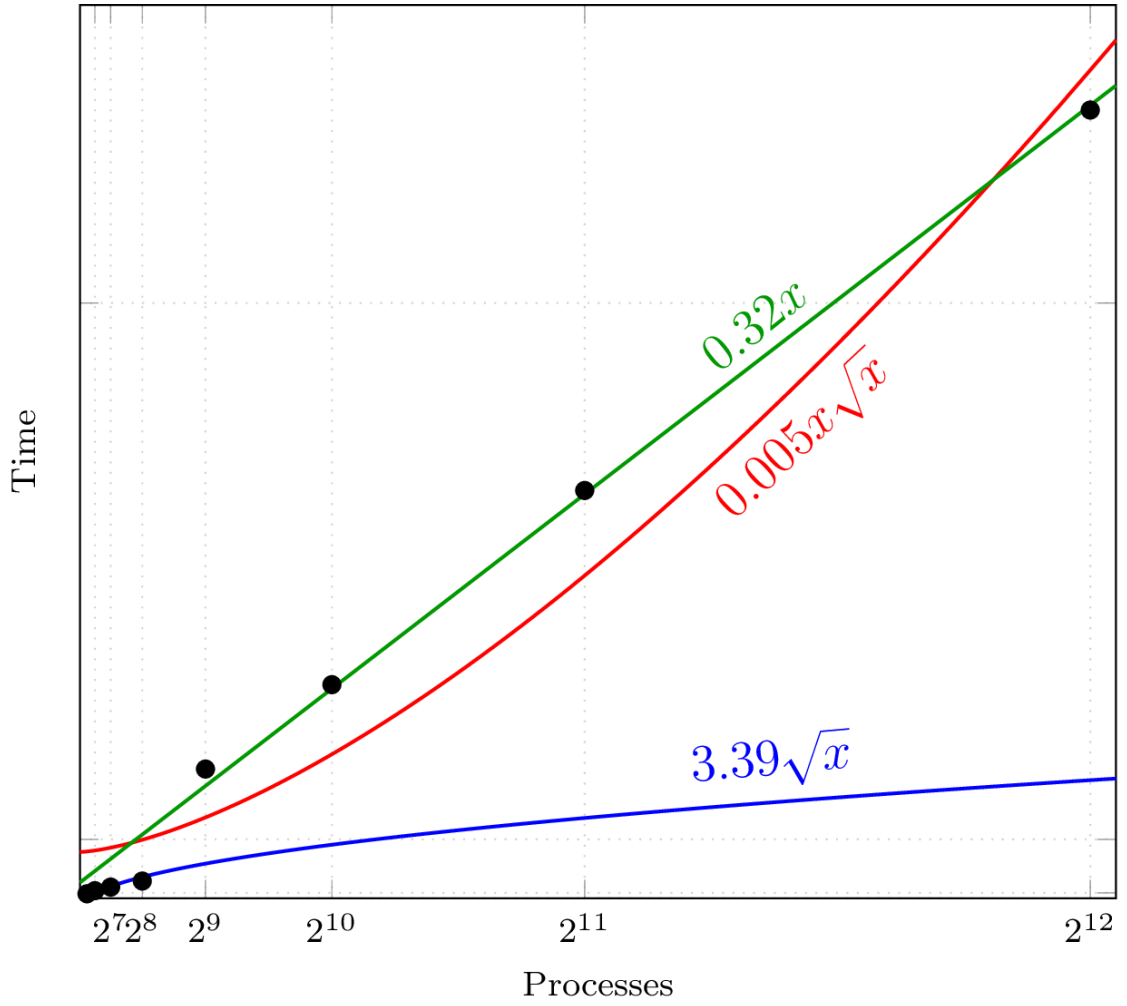
- Constant c_1
- Single parameter $c_1 + c_2 \cdot x_1$
- Multiple parameters …
  - Additive $c_1 + c_2 \cdot x_1 + c_3 \cdot x_2$
  - Multiplicative $c_1 + c_2 \cdot x_1 \cdot x_2$
  - Complex $c_1 + c_2 \cdot x_1 \cdot x_2 + c_3 \cdot \log x_2 \cdot x_2^3$

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

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 (!)



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)

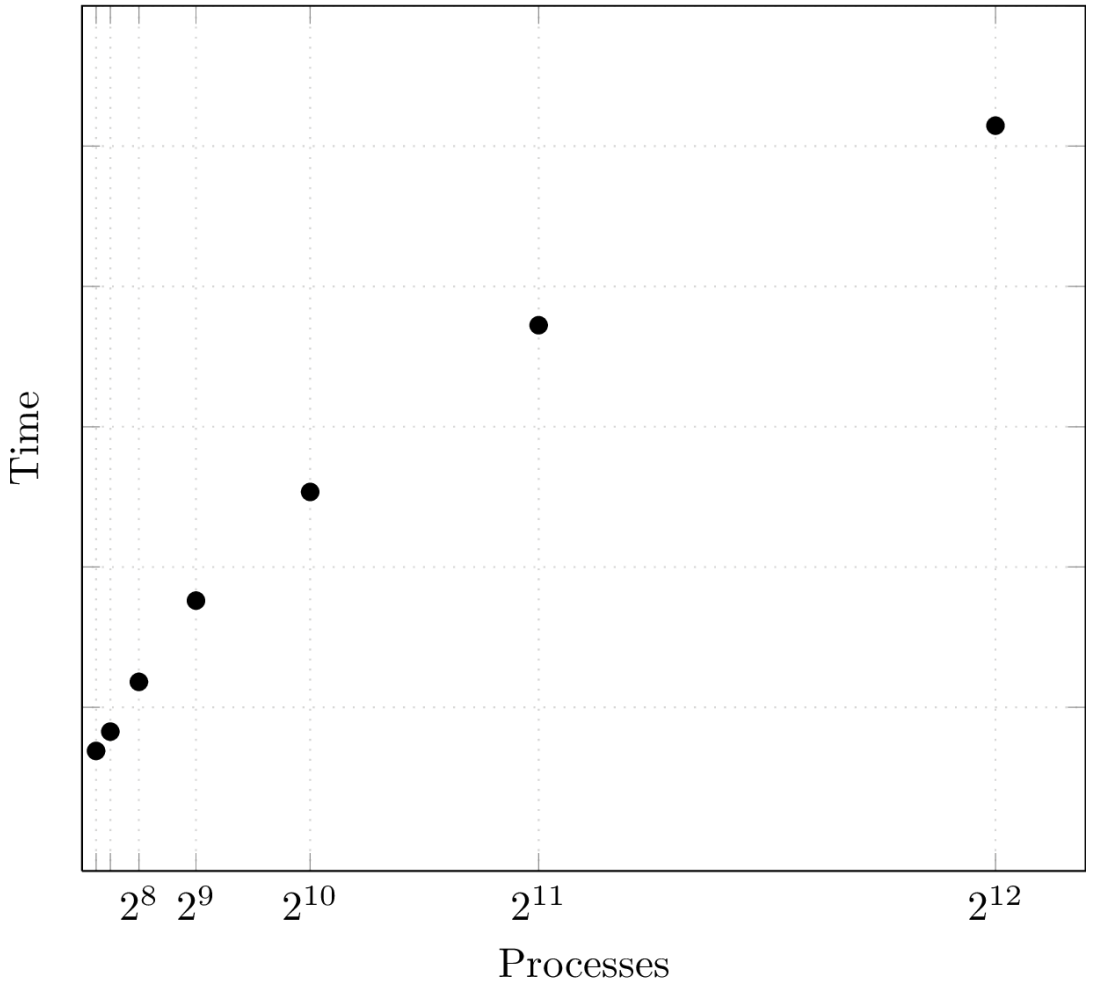
$p_1 = 256$

$p_2 = 512$

$p_3 = 1024$

$p_4 = 2048$

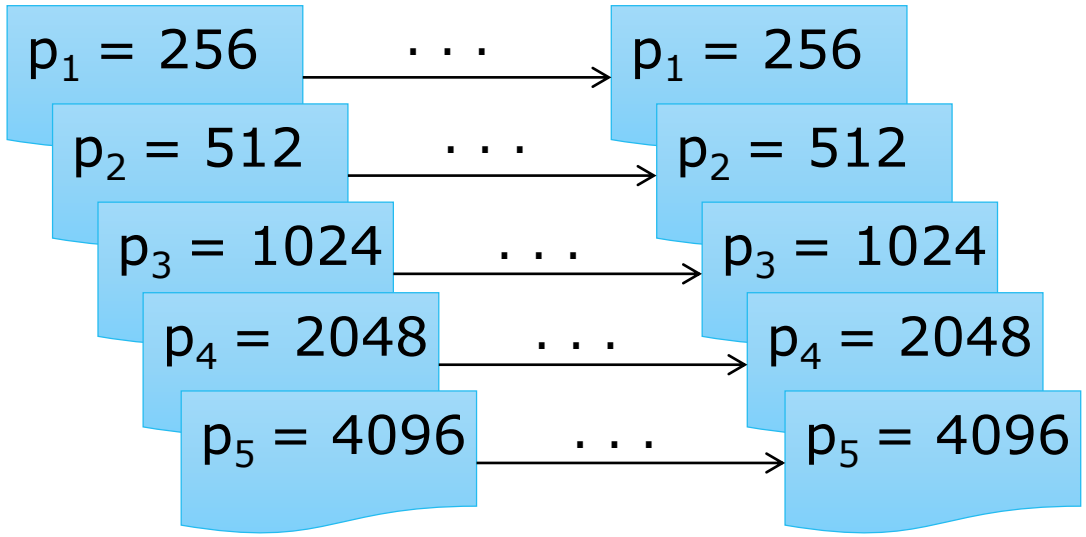
$p_5 = 4096$

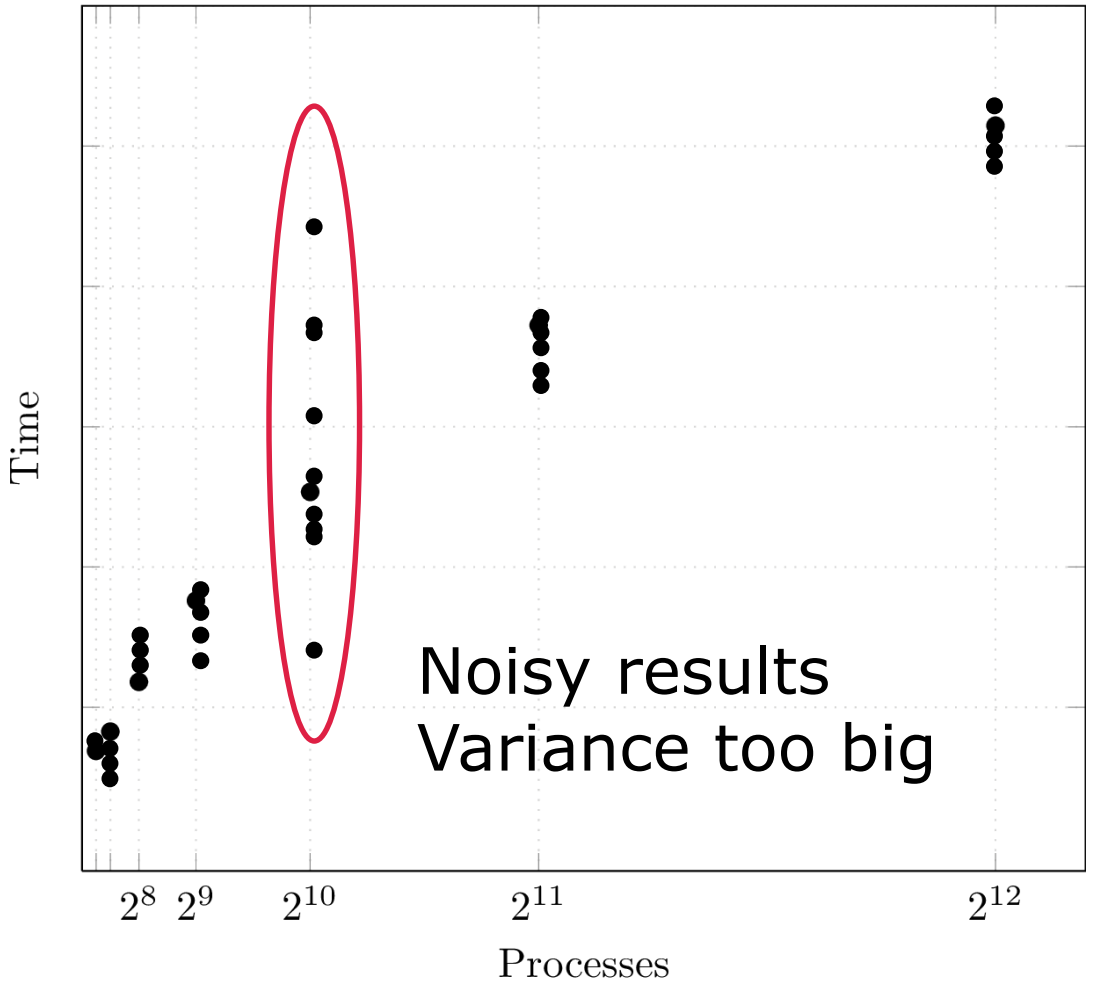


Performance measurements (3)

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

Performance measurements (profiles)



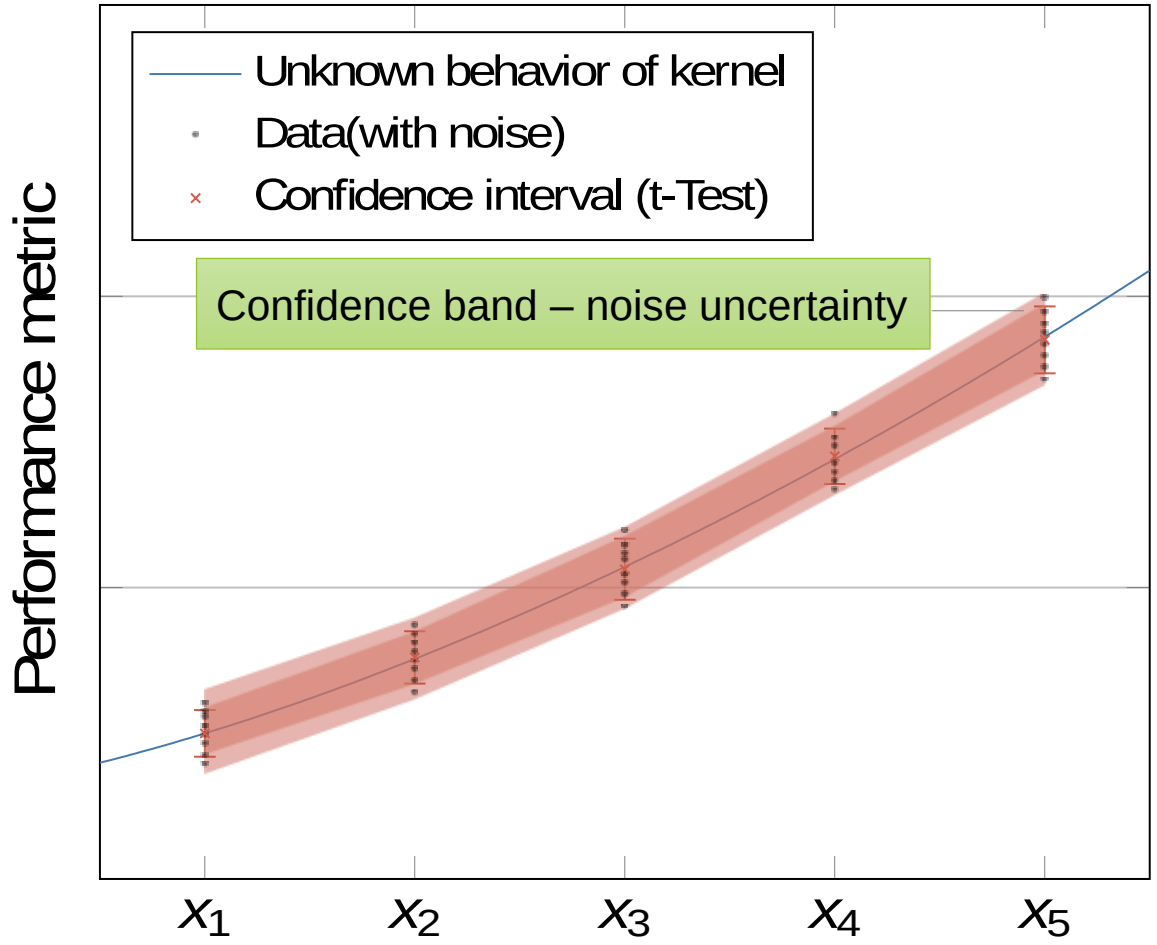


Statistical quality control

- If the confidence interval is too wide, the fit will not be optimal, or overfitting might occur

$$\mathrm{CI} = f(\mathrm{mean}, \mathrm{stddev})$$

- To improve CI – increase repetitions, include different configurations



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

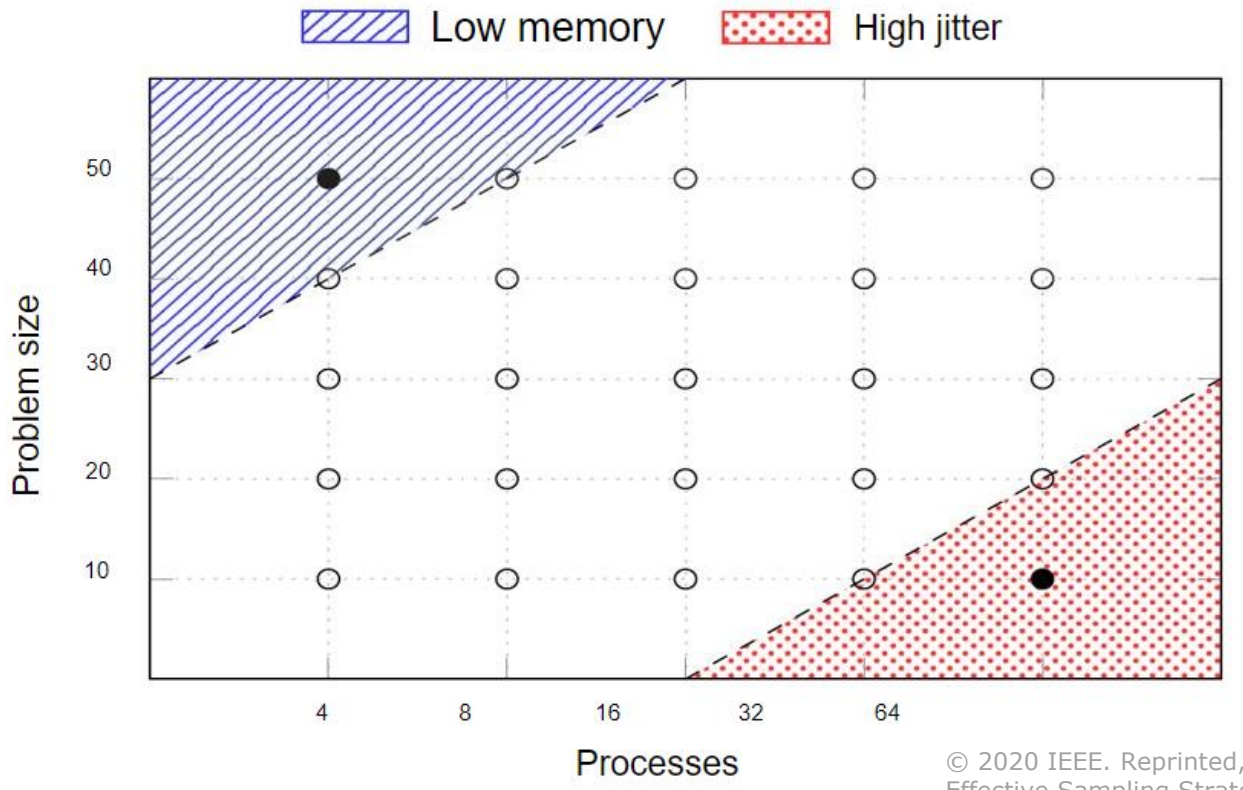
- Rule of thumb: adj. $R^2 > 0.95$

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

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

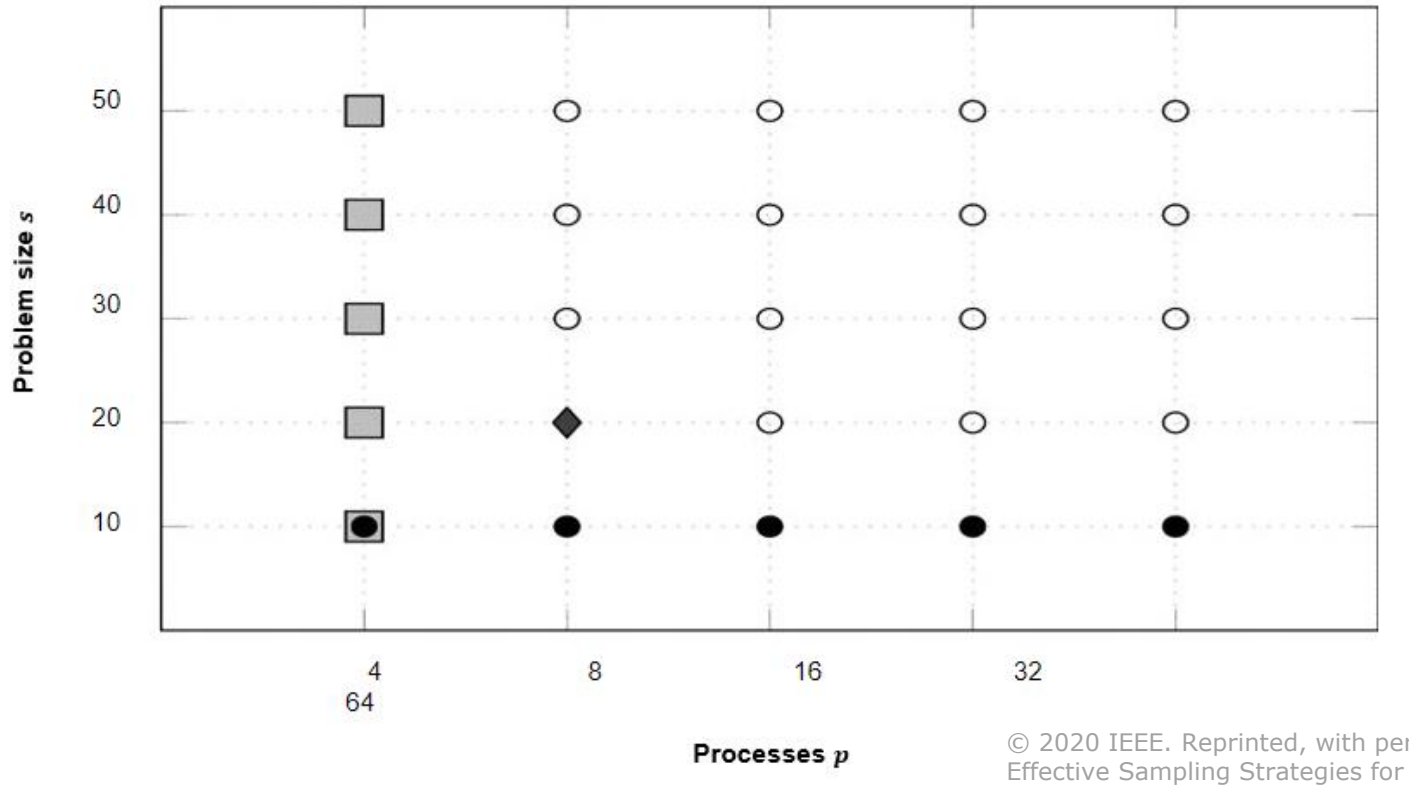
Sparse Modeling

- Experiments can be expensive
- So far we needed 5^{m+1} experiments, m=number of parameters



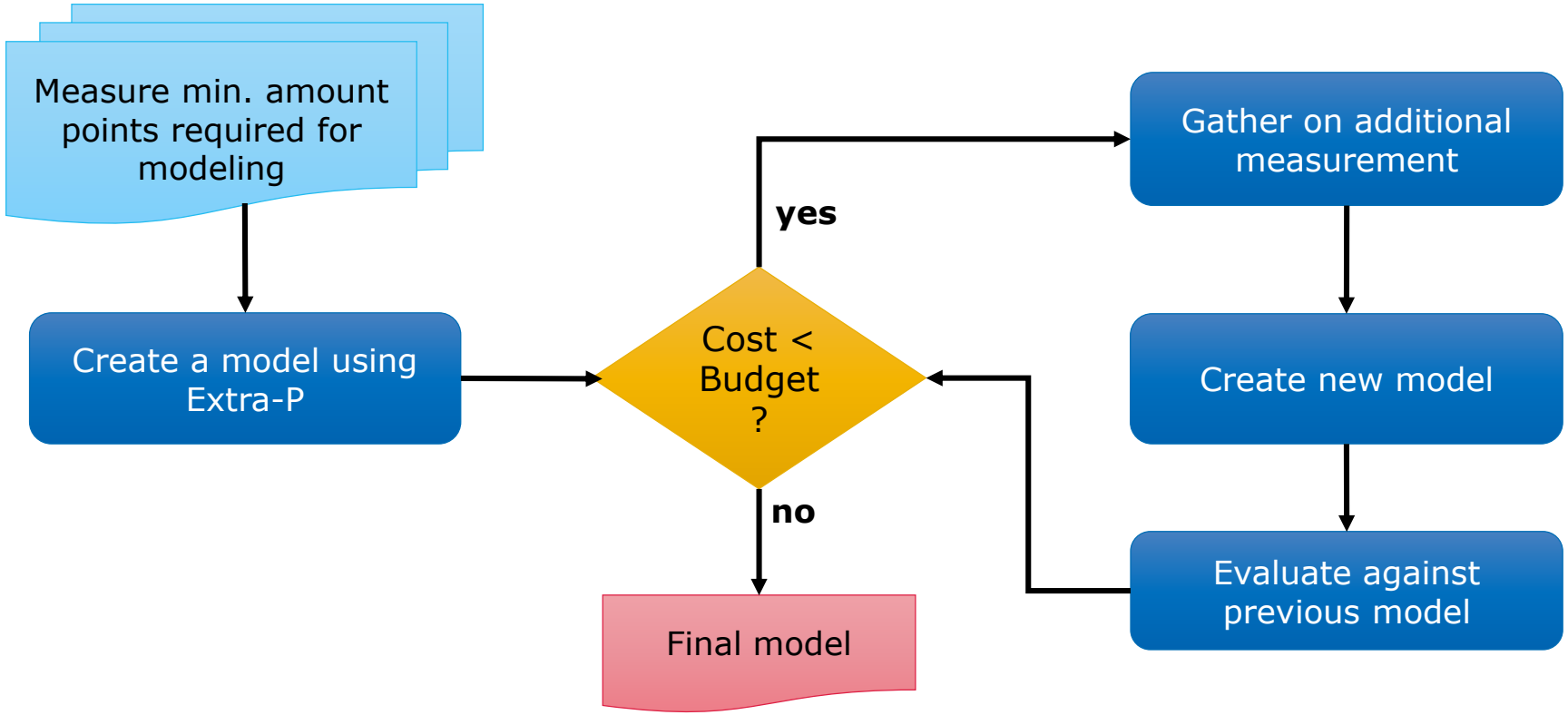
Extra-P 4.0: Sparse Modeling

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



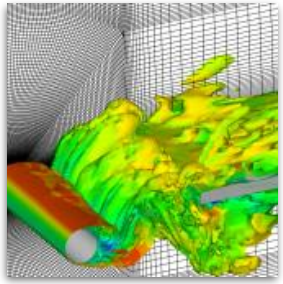
Extra-P 4.0: Sparse Modeling

- Recommended experiment configuration strategy using our heuristic guideline



Extra-P 4.0: Sparse Modeling

- Using sparse modeling we can reduce the average modeling cost by ~85% (on synthetic data)
- We can retain ~92% of the model accuracy (on synthetic data)
- Allows a more flexible experiment design



FASTEST
- 70% decrease in cost
- ~2% prediction error

Image by Institute for Numerical Methods in Mechanical Engineering, TU Darmstadt

Kripke
- 99% decrease in cost
- ~39% prediction error

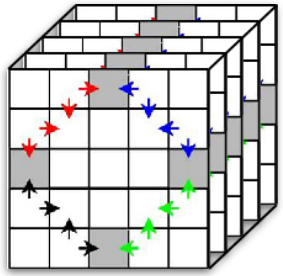


Image by Lawrence Livermore National Laboratory (CC BY-NC-SA 4.0)

Relearn
- 85% decrease in cost
- ~11% prediction error

Using Extra-P

Installing Extra-P

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

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

Automatic performance modeling with Extra-P

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

Instrumentation & Measurement

Score-P

Performance measurements (profiles)

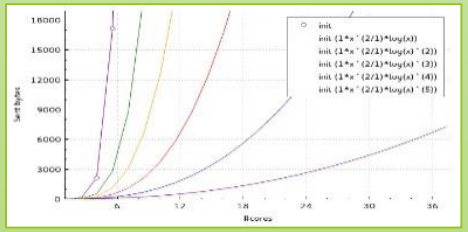
- Myapp.weak.p128.r1/profile.cubex
- Myapp.weak.p256.r1/profile.cubex
- Myapp.weak.p512.r1/profile.cubex
- Myapp.weak.p1024.r1/profile.cubex
- Myapp.weak.p2048.r1/profile.cubex

Input

Extra-P

Output

Results (visual)



Results (textual)

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

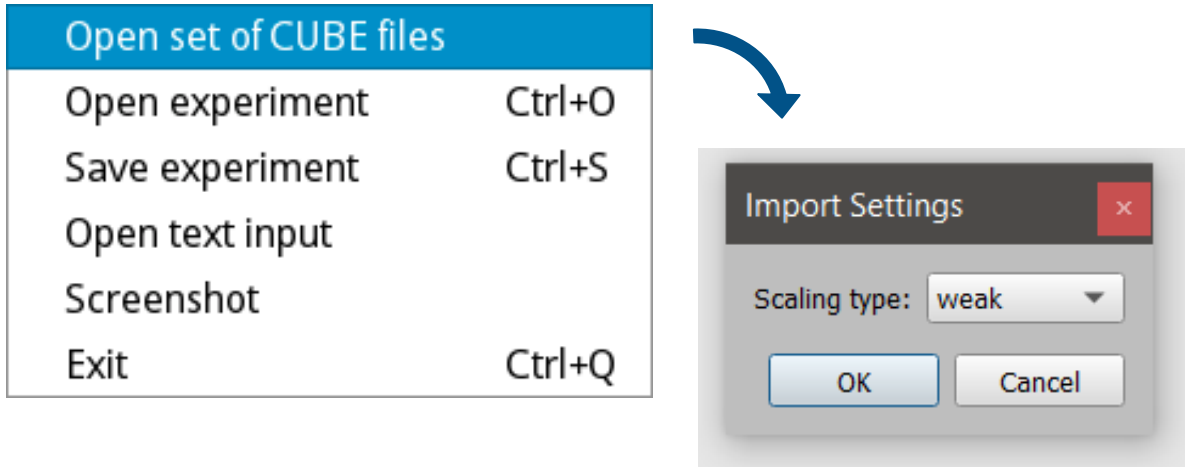
Modeling sets of Cube experiments

Extra-P Cube input description

Modeling tool expects Cube files in the following format:

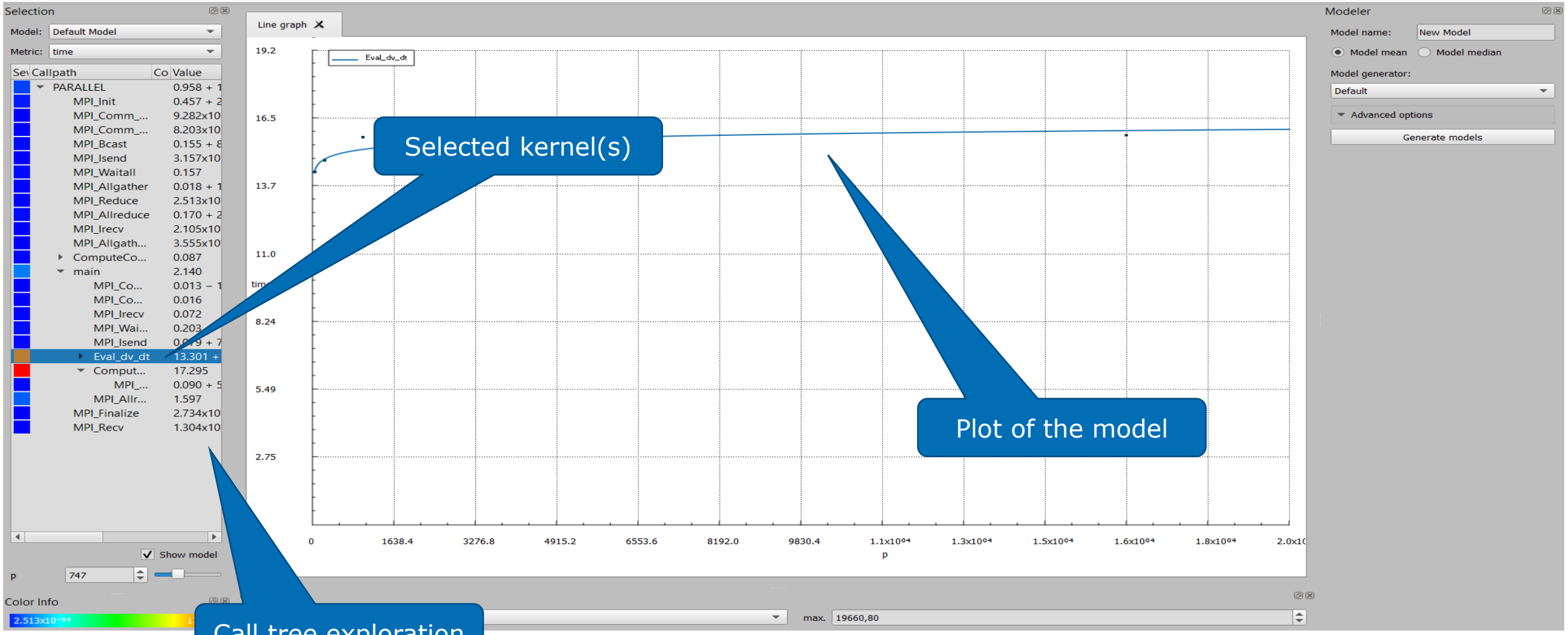
- `<DIR>/<PREFIX>.<PARAMETERS>.r<REPETITION>/<FILENAME>.cubex`
  - `DIR`, `PREFIX`, `FILENAME` – are just names, no meaning for Extra-P
  - `REPETITION` – number of the repetition of the experiments with same parameter values
- `<PARAMETERS>:=<PARAM1><VALUE1>.⋯.<PARAM`n`><VALUE`n`>`
  - List of parameter-value-pairs separated by "."
  - `PARAM` – varied parameter e.g. number of processes
  - `VALUE` – value of the varied parameter

Extra-P Cube input description



Visualization with Extra-P

Extra-P user interface



Extra-P call tree view

Metric selection

Model selection

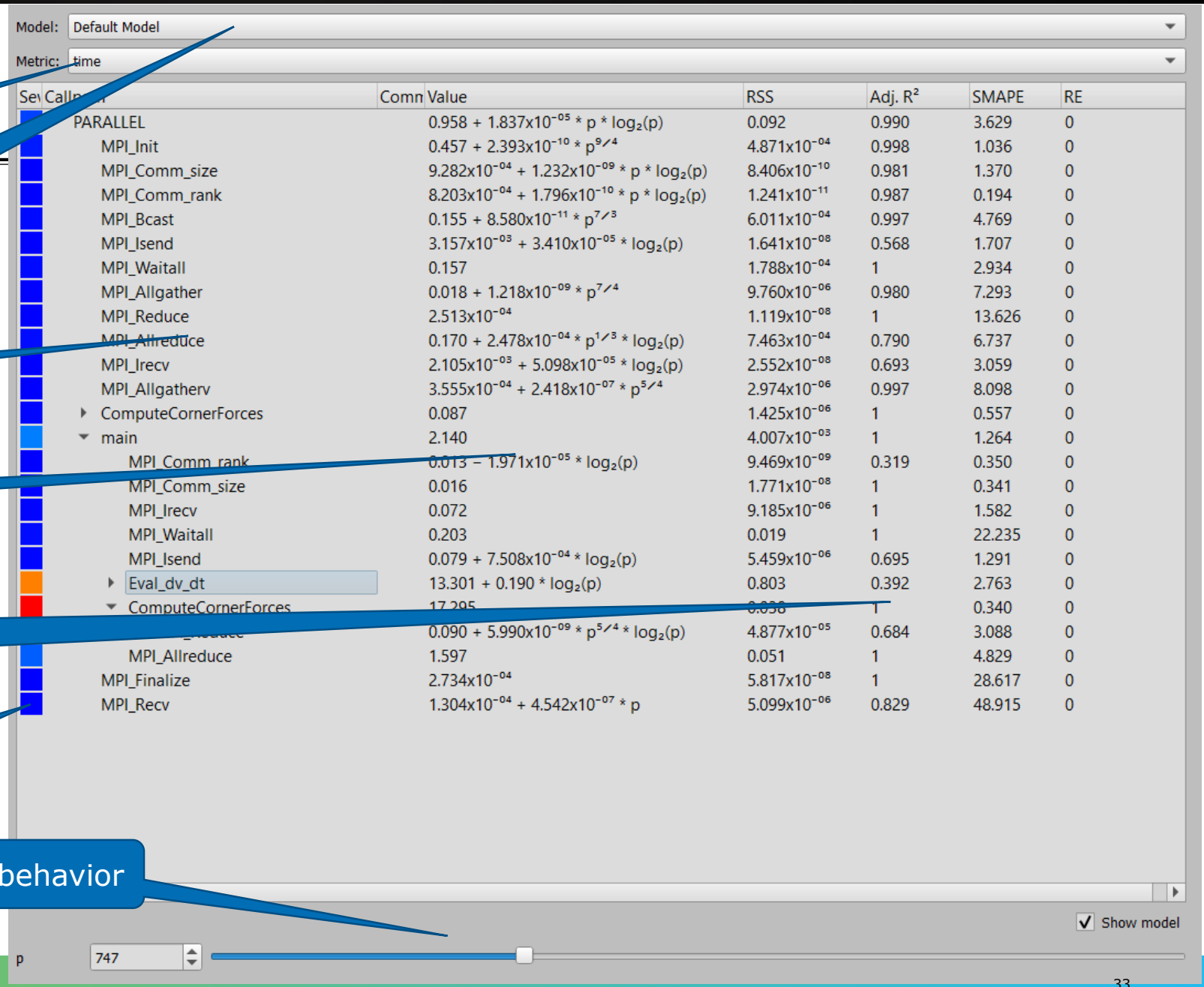
Call tree exploration

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

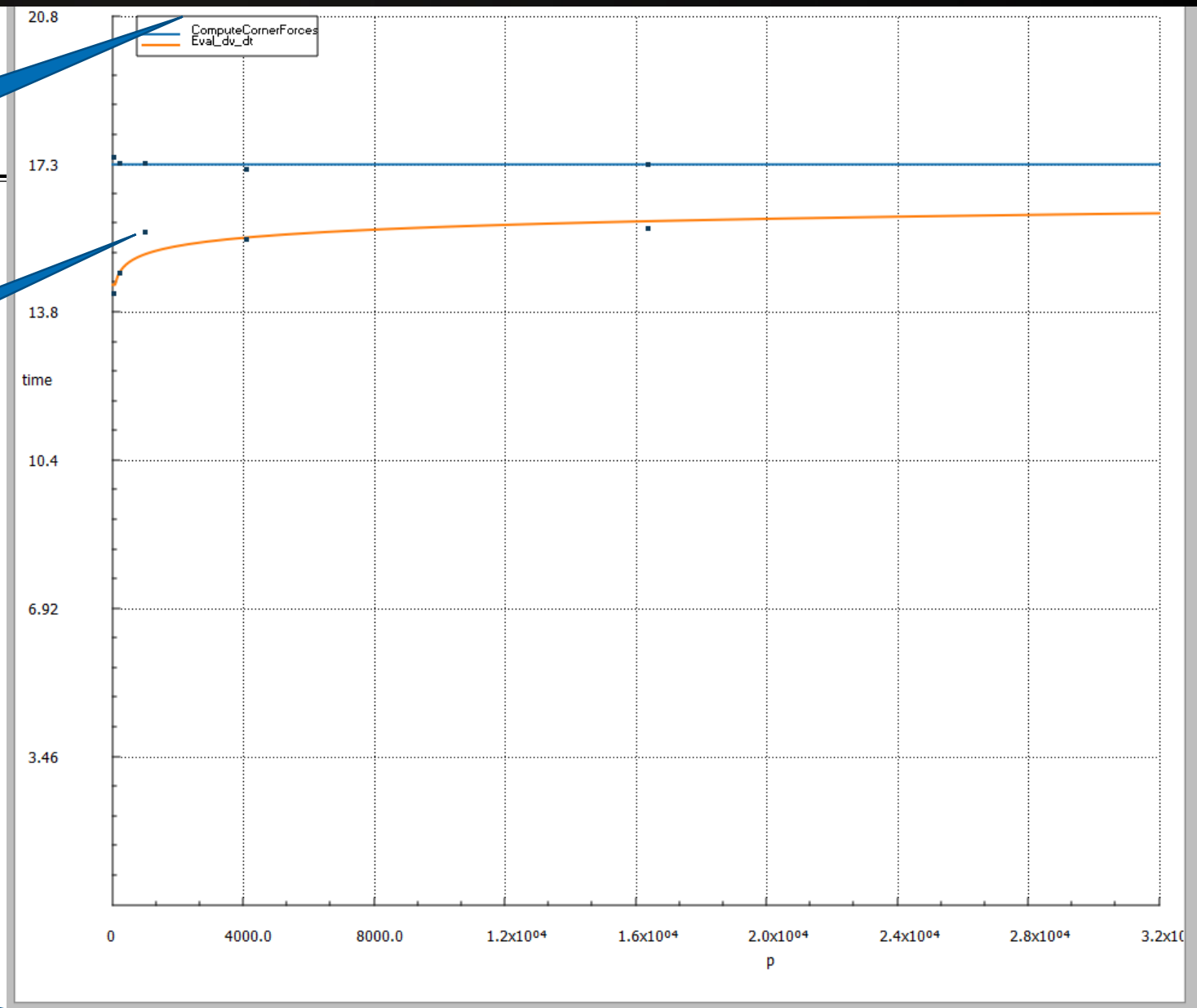


Extra-P model view

Models selected in the Call path view

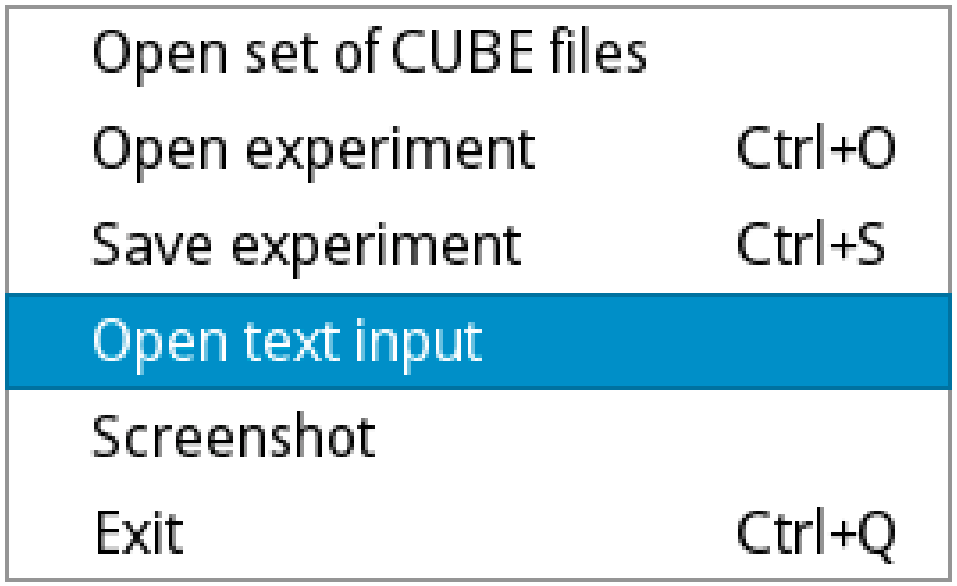
Measurement values

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



Modeling measurements from a text file

Choose input file



Select input file in the GUI

Extra-P input in text form

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

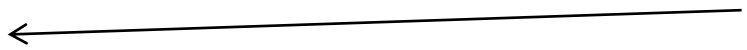
```
PARAMETER p
POINTS 1000 2000 4000 8000 16000
METRIC metric1
REGION region1
DATA 1 1 1 1 1
DATA 4 4 4 3.99 4.01
DATA 16 15.999 16.01 16.01 15.99
DATA 64 64 64 64.01 63.99
DATA 256.01 255.99 256 256
```

Parameter name

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

Extra-P input in text form

```
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
```



Measurement points

Use at least 5, preferably 6,
but in general the more the better

Extra-P input in text form

```
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
```

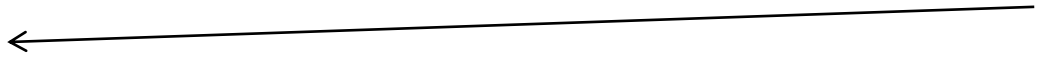
Metric name

Region name

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

Extra-P input in text form

```
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
```



Data points

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

Using the command line tool

Extra-P command line tool

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

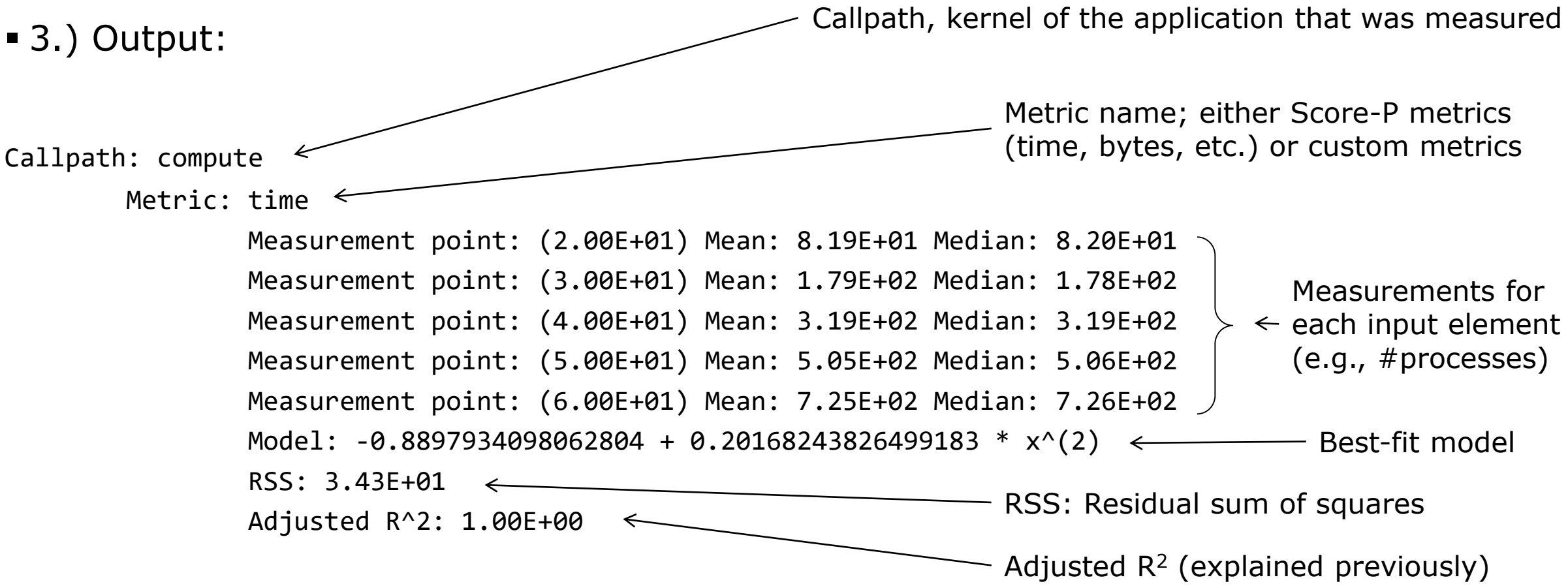
- 1.) Run: `extrap`

- Command Format: `extrap` *`OPTIONS`* `(--cube | --text | --talpas | --json | --extra-p-3)` *`FILEPATH`*

- 2.) Select input type: `extrap --text /lrz/sys/courses/vihps/material/extrap_data/input.txt`

Extra-P command line tool

- 3.) Output:



Hands-on exercises

Extra-P exercises

- Run: `extrap`
- Example data: tests/data
- Open the examples in the GUI
- Use the command line tool
- Open the text based, JSON input example
- Produce textual output and inspect it

- 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*