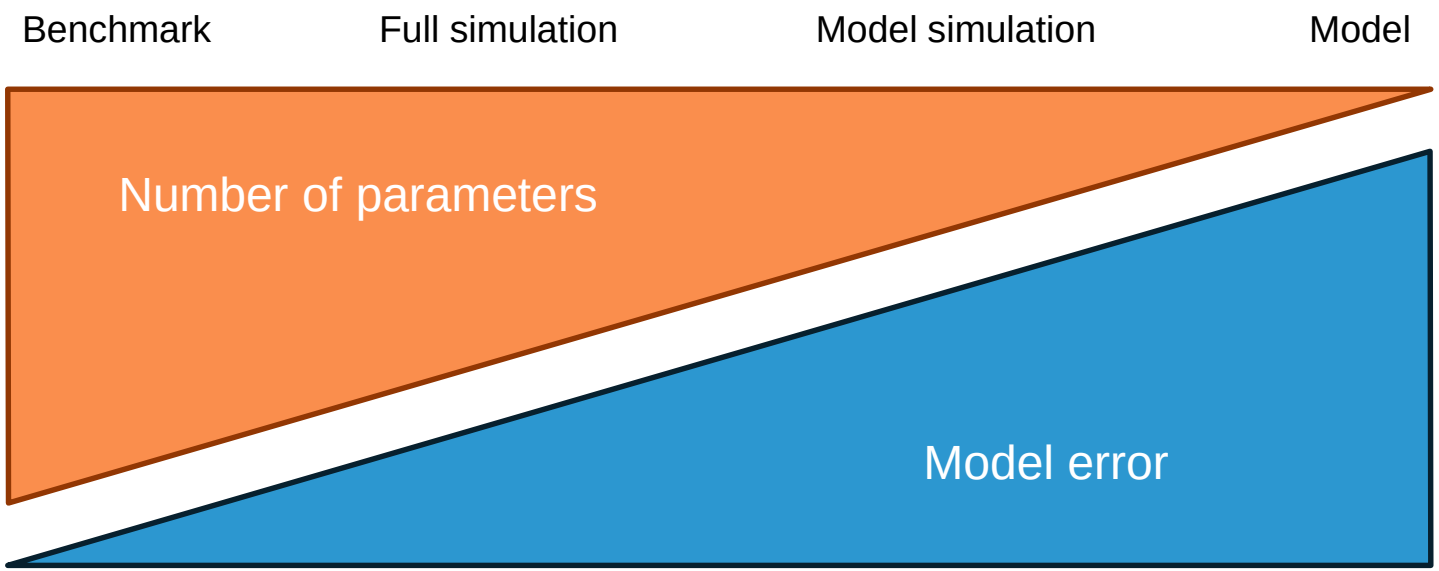
Extra-P - Performance Modeling at Scale

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

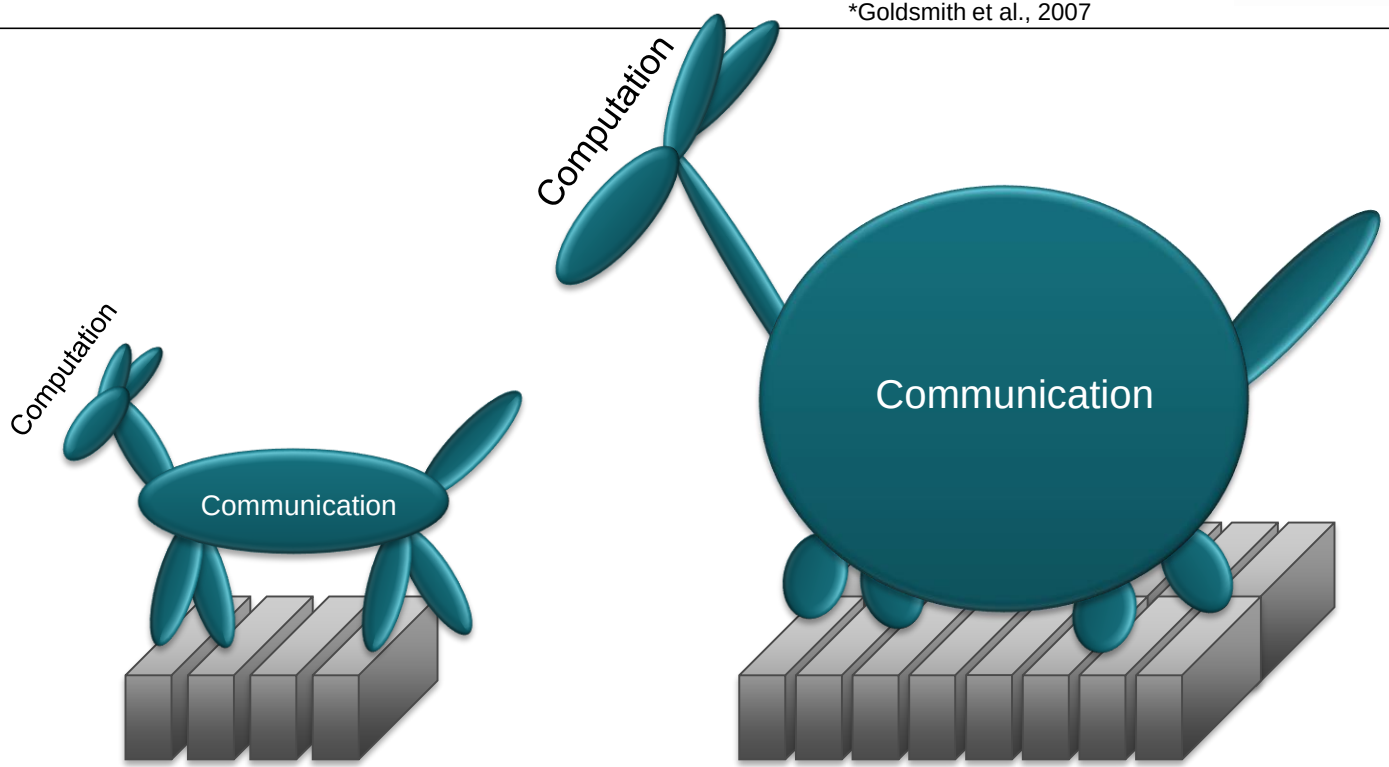
[1] Technical University of Darmstadt , [2] ETH Zürich

Spectrum of performance analysis methods



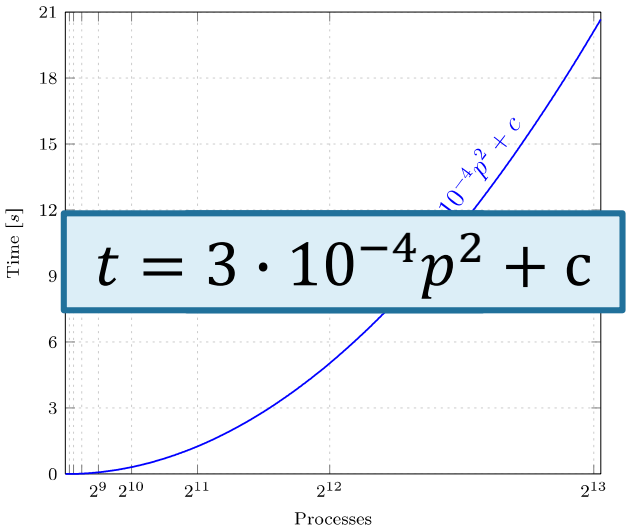
Scaling your code can harbor *performance surprises**…

*Goldsmith et al., 2007



Performance model

Formula that expresses relevant performance metric as a function of one or more execution parameters



$$t = 3 \cdot 10^{-4} p^2 + c$$

Analytical (i.e., manual) creation challenging for entire programs

Identify kernels
- Incomplete coverage

Create models
- Laborious, difficult

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

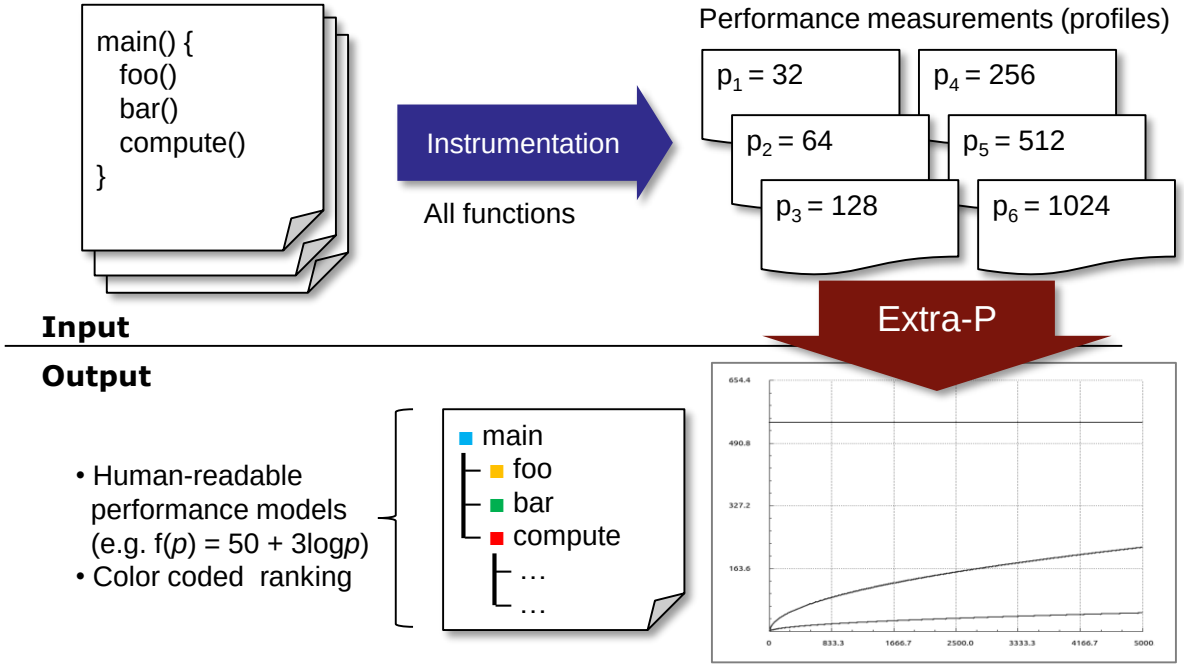
Disadvantages:

- 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

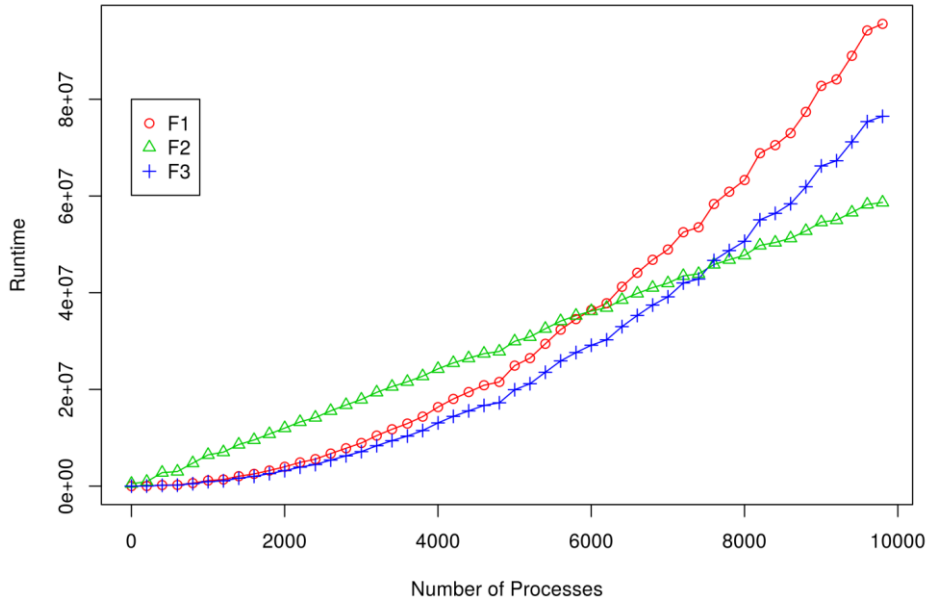
Automatic empirical performance modeling with Extra-P



Source: A. Calotoiu, D. Beckingsale, C. W. Earl, T. Hoefler, I. Karlin, M. Schulz, F. Wolf: Fast Multi-Parameter Performance Modeling. In Proc. of the 2016 IEEE International Conference on Cluster Computing (CLUSTER), Taipei, Taiwan, September 2016

Primary focus on scaling trend

Common performance analysis chart in a paper

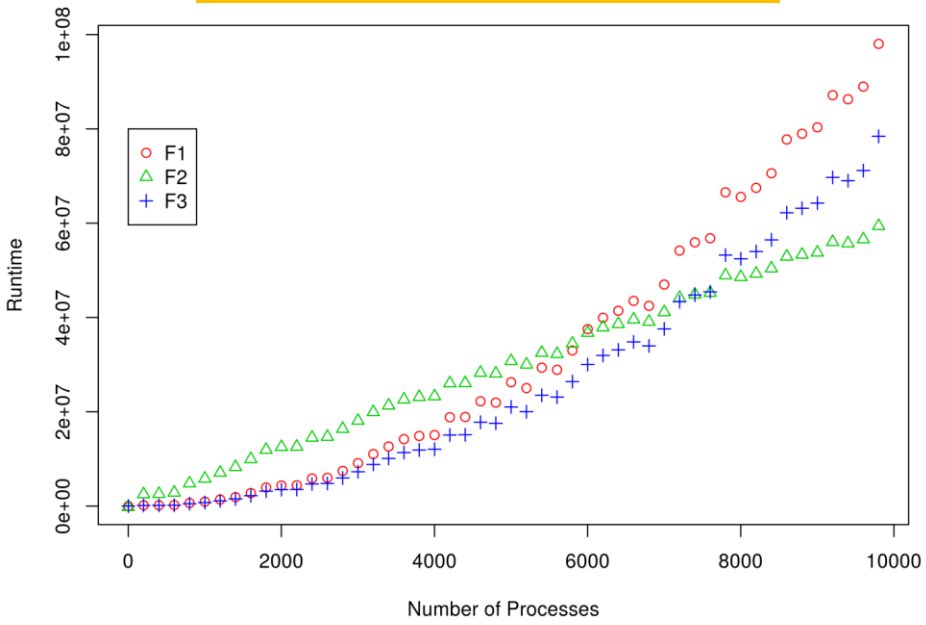


Ranking

1. F_2
2. F_1
3. F_3

Primary focus on scaling trend

Actual measurement in laboratory conditions

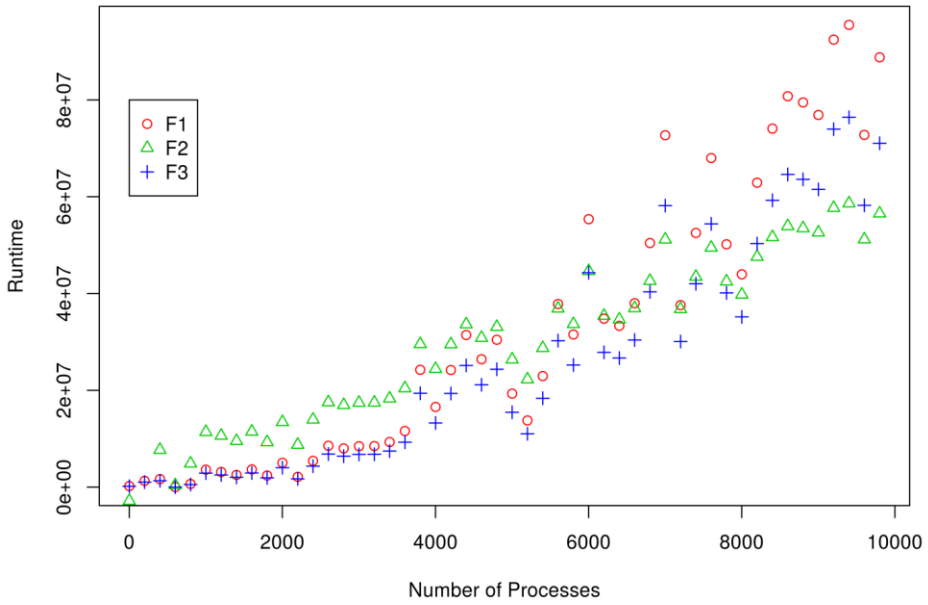


Ranking

1. F_2
2. F_1
3. F_3

Primary focus on scaling trend

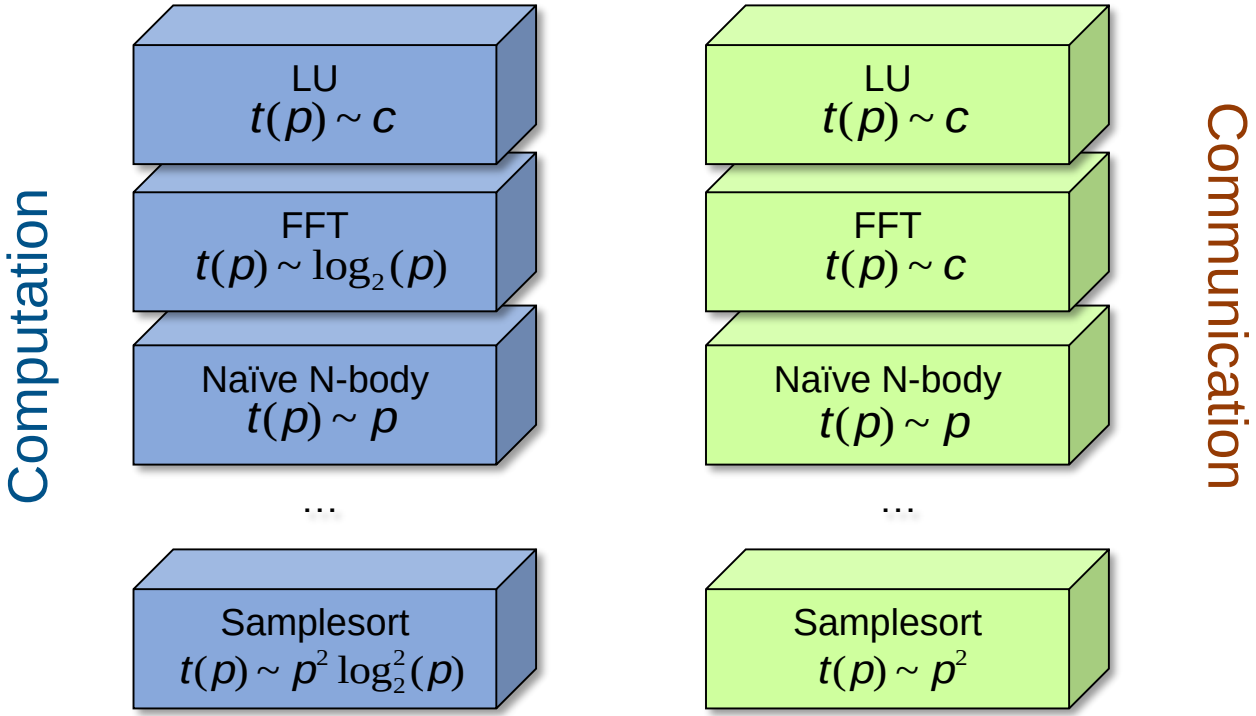
Production Reality



Ranking

1. F_2
2. F_1
3. F_3

Model building blocks



Performance model normal form

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

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

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

c_1	$c_1 \cdot \log(p)$
$c_1 \cdot p$	$c_1 \cdot p \cdot \log(p)$
$c_1 \cdot p^2$	$c_1 \cdot p^2 \cdot \log(p)$

Performance model normal form

$$f(p) =$$

$$n = 1$$

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

$$J = \{0, 1\}$$

$$c_1 + c_2 \times p$$

$$c_1 + c_2 \times p^2$$

$$c_1 + c_2 \times \log(p)$$

$$c_1 + c_2 \times p \times \log(p)$$

$$c_1 + c_2 \times p^2 \times \log(p)$$

$$c_1 \times \log(p) + c_2 \times p$$

$$c_1 \times \log(p) + c_2 \times p \times \log(p)$$

$$c_1 \times \log(p) + c_2 \times p^2$$

$$c_1 \times \log(p) + c_2 \times p^2 \times \log(p)$$

$$c_1 \times p + c_2 \times p \times \log(p)$$

$$c_1 \times p + c_2 \times p^2$$

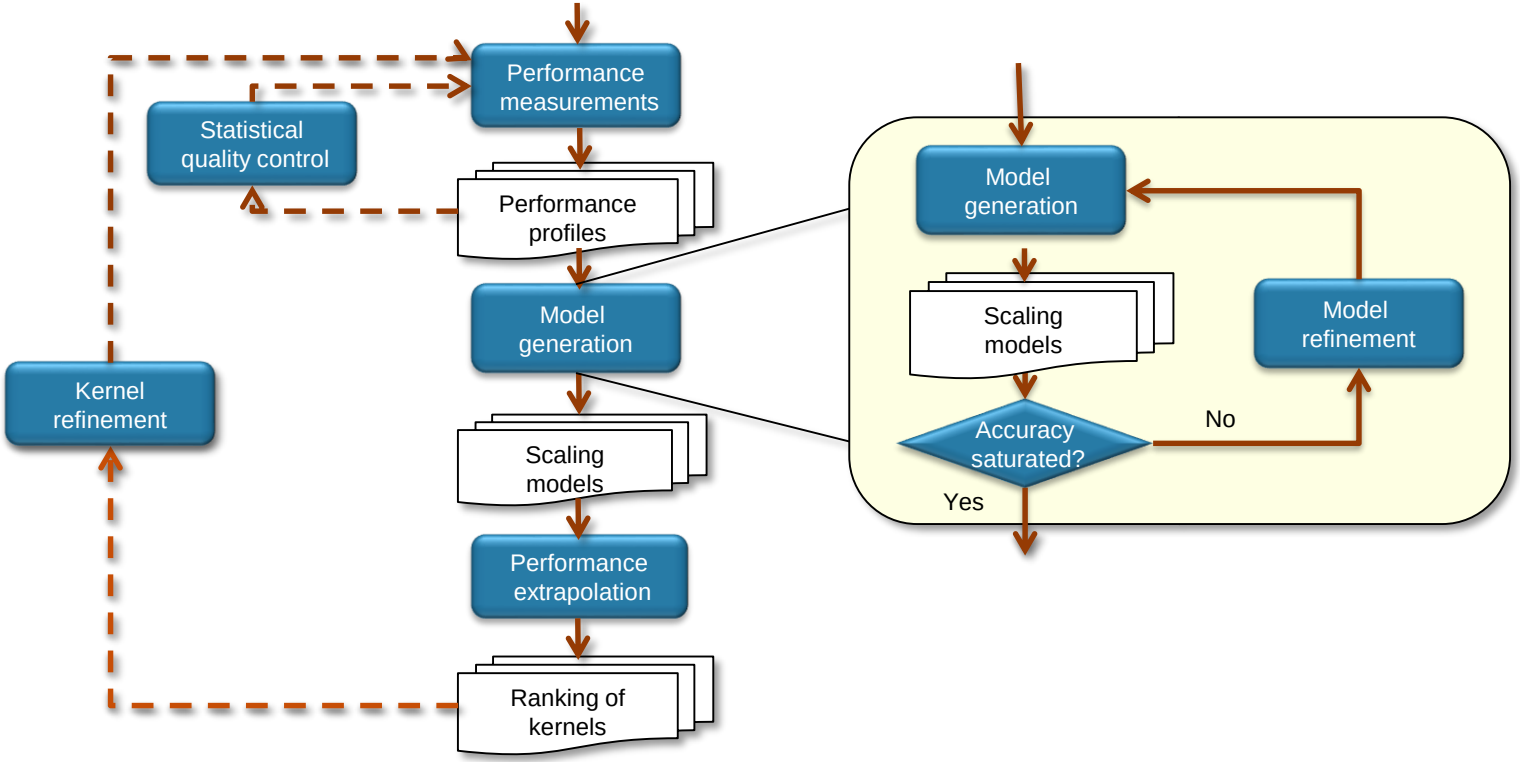
$$c_1 \times p + c_2 \times p^2 \times \log(p)$$

$$c_1 \times p \times \log(p) + c_2 \times p^2$$

$$c_1 \times p \times \log(p) + c_2 \times p^2 \times \log(p)$$

$$c_1 \times p^2 + c_2 \times p^2 \times \log(p)$$

Workflow

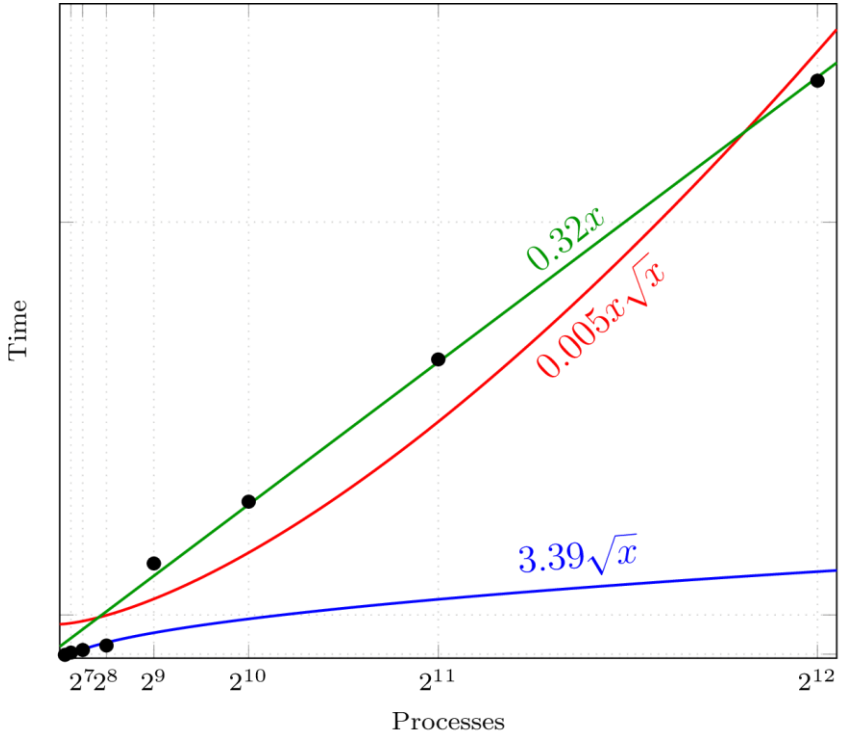

Performance measurements
Statistical quality control
Performance profiles
Model generation
Kernel refinement
Scaling models
Performance extrapolation
Ranking of kernels
Model generation
Scaling models
Model refinement
Accuracy saturated?
No
Yes

Assumptions & limitations

- Only one scaling behavior for all the measurements; no jumps
- Some MPI collective operations switch their algorithm – results in bad models

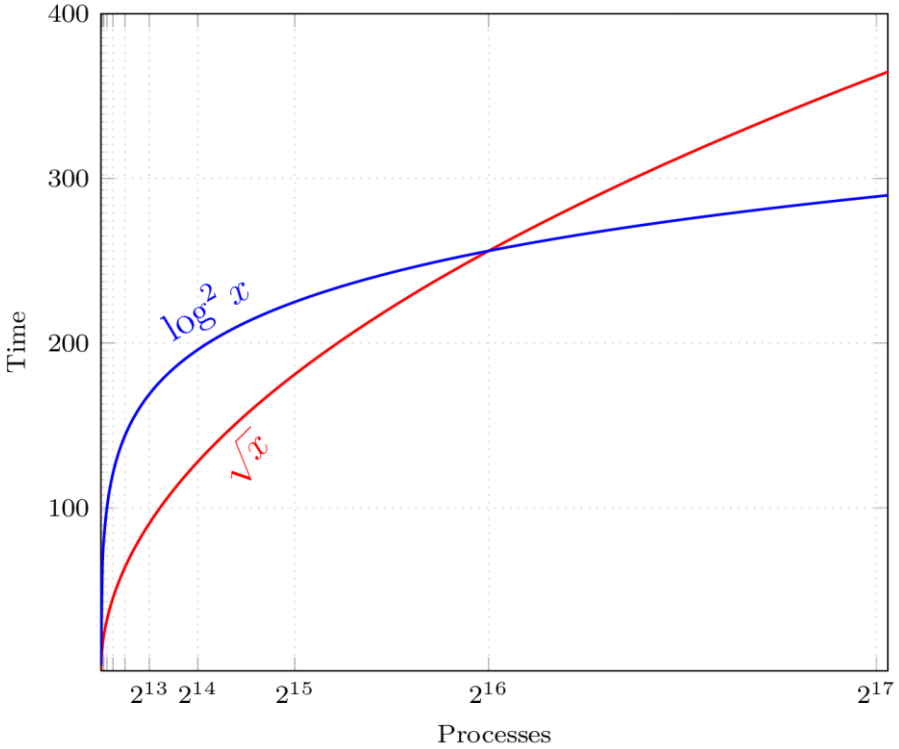
Example: red model tries to model measurements of different algorithms

- First 4 points – one function
- Last 4 points – another function (linear)
- Adj. R2 = 0.95085 (!)

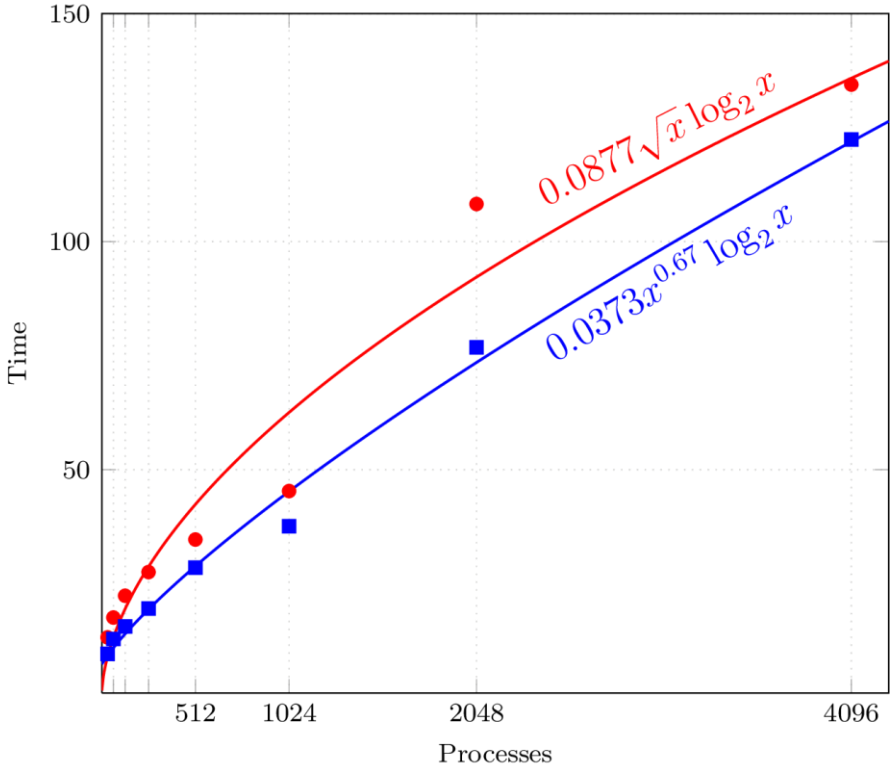


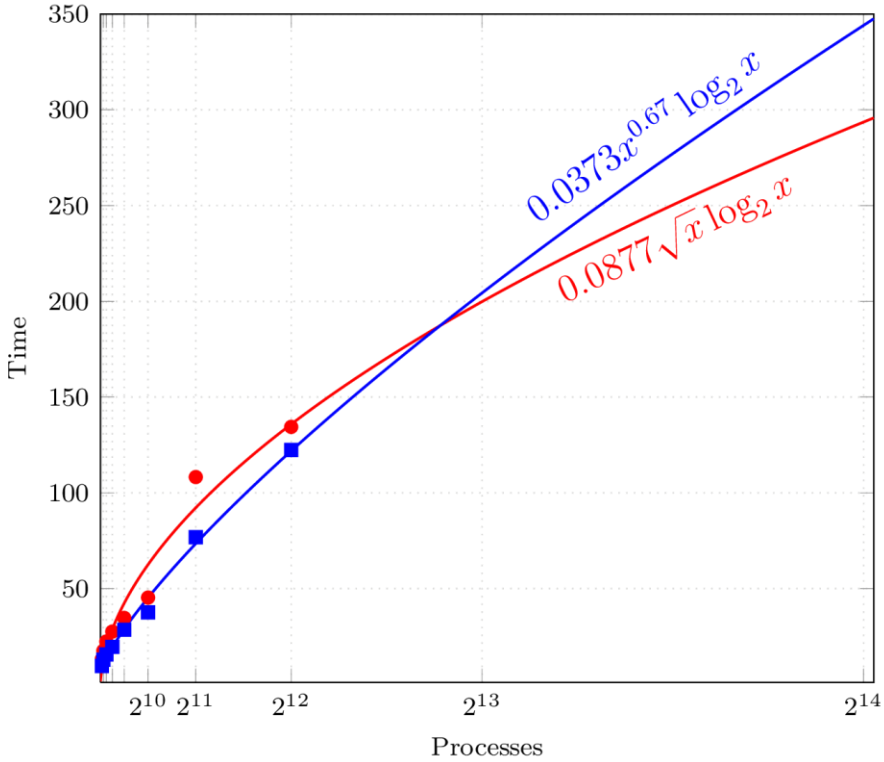
Changing growth trends

- Ranking according to growth rate difficult:
- $\log^2(p)$ vs. $\sqrt{p}$



Changing growth trends (2)





Ranking of kernels

- Kernels are ranked according the leading-order terms in the models

- Leading-order term → big-O notation

- For example: $O(x)$ comes before $O(x^2)$

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 required

Performance measurements (profiles)

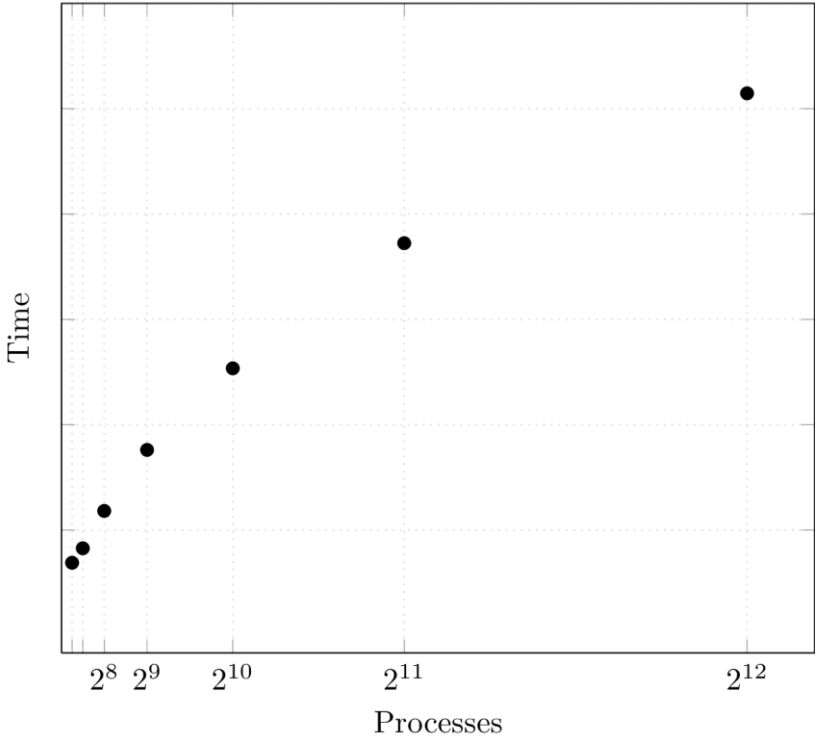
$p_1 = 256$

$p_2 = 512$

$p_3 = 1024$

$p_4 = 2048$

$p_5 = 4096$

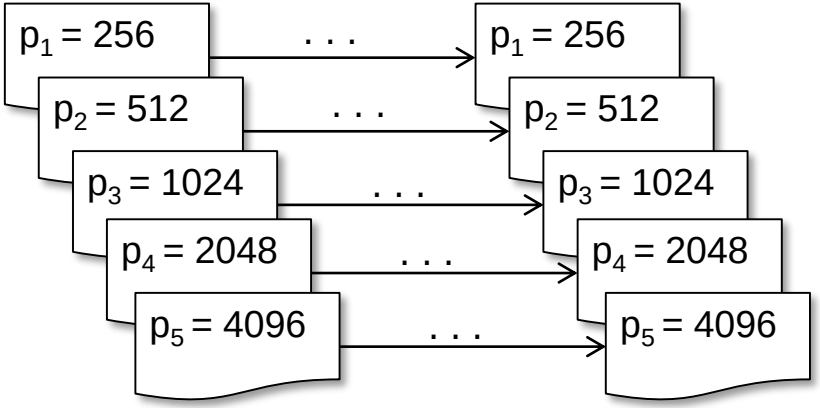


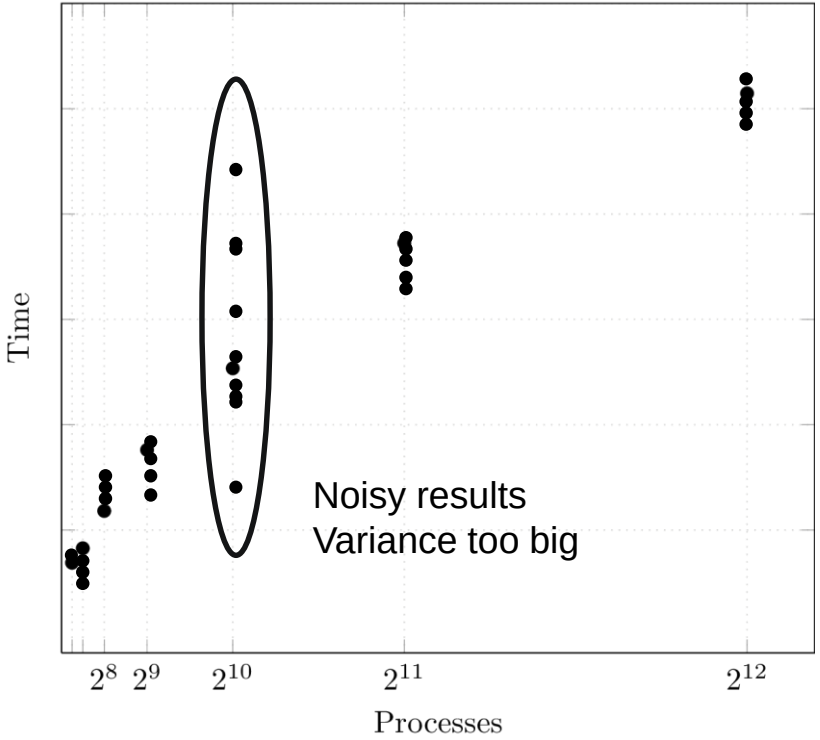
Performance measurements (3)

- At least 5 different measurements required

Each measurement repeated multiple times

Performance measurements (profiles)



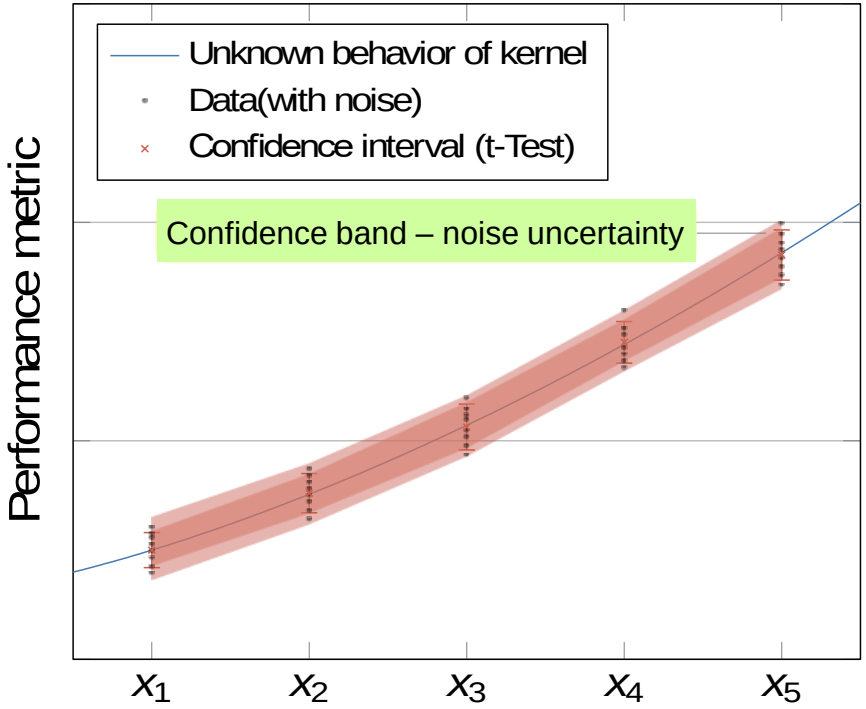


Statistical quality control

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

  CI = f(mean, stddev)

- To improve CI - increase repetitions, include different configurations



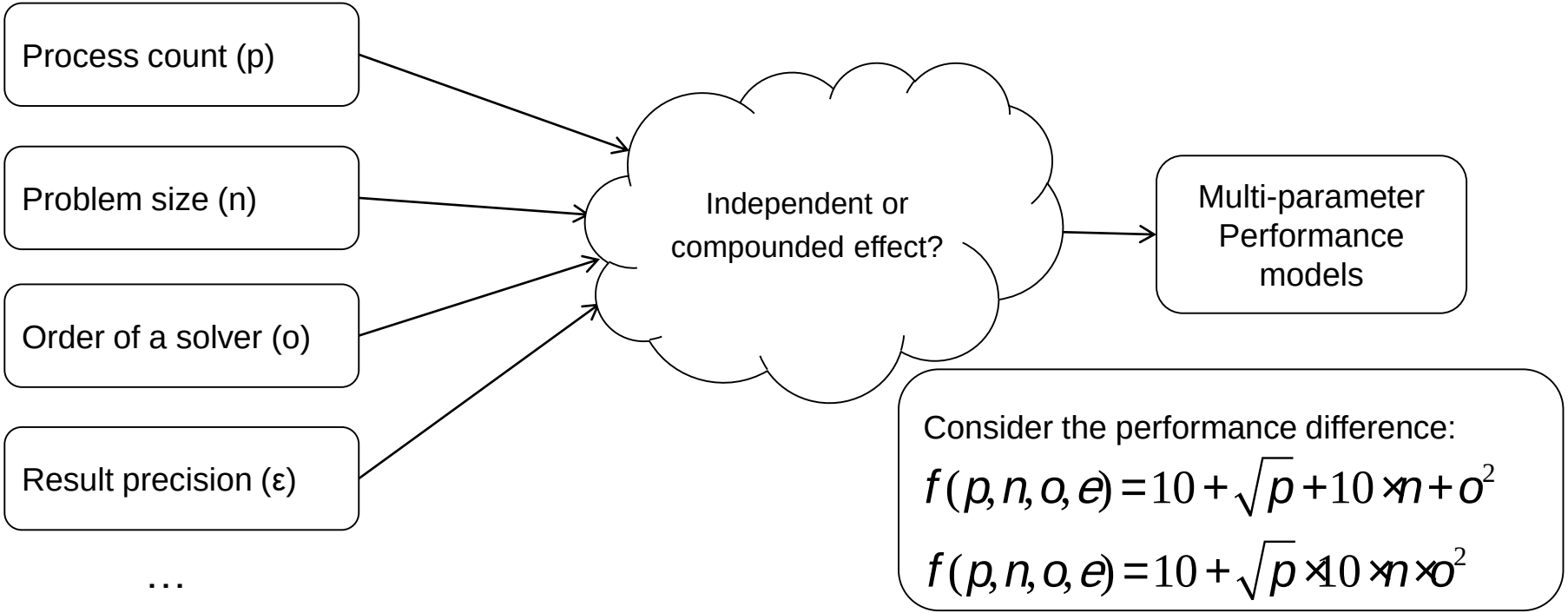
How to select the best model? - adjusted R^2

- 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. $R^2 > 0.95$

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

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

Extra-P 3.0: Fast multi-parameter performance modeling



Extra-P 3.0: Fast multi-parameter performance modeling

Expanded performance model normal form

$$f(p) = \sum_{k=1}^{n} c_k \prod_{l=1}^{m} p_l^{i_{kl}} \times \log_2^{j_{kl}}(p_l)$$

$n \in \mathbb{N}$
$m \in \mathbb{N}$
$i_{kl} \in I$
$j_{kl} \in J$
$I, J \subset \mathbb{Q}$

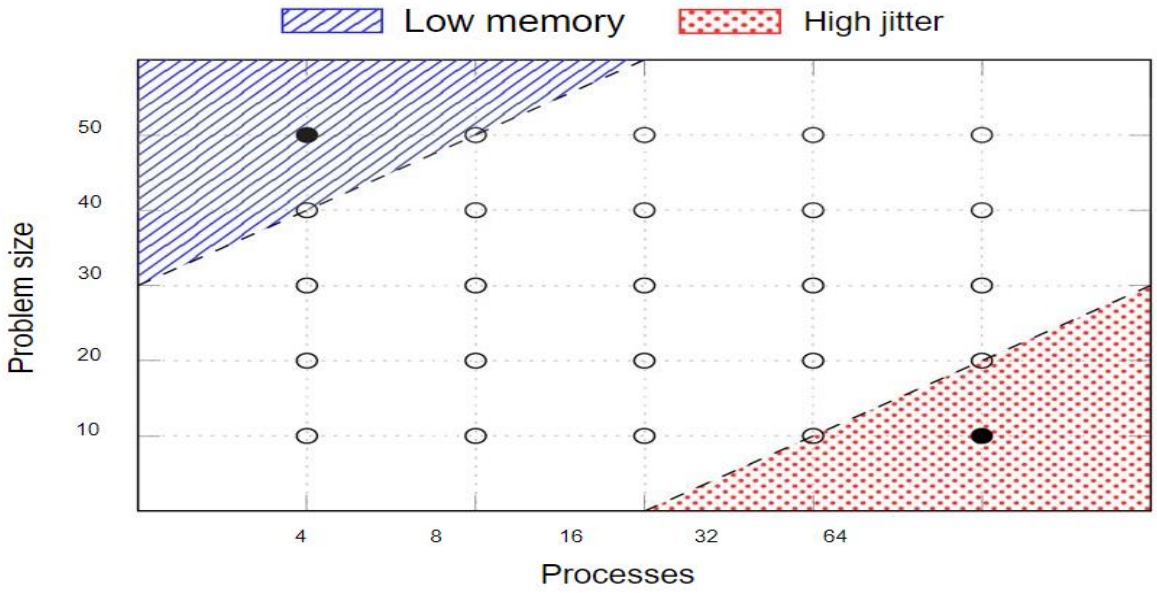
$n = 2$
$m = 2$
$I = \left\{ \frac{0}{4}, \frac{1}{4}, \ldots, \frac{12}{4} \right\}$
$J = \{0, 1, 2\}$

Model candidates

- Constant c_1
- Single parameter $c_1 + c_2 \times p_1$
- Multiple parameters ...
  - Additive $c_1 + c_2 \times p_1 + c_3 \times p_2$
  - Multiplicative $c_1 + c_2 \times p_1 \times p_2$
  - Complex $c_1 + c_2 \times p_1 + c_3 \times p_1^2 \times p_2 \times \log_2(p_2)$

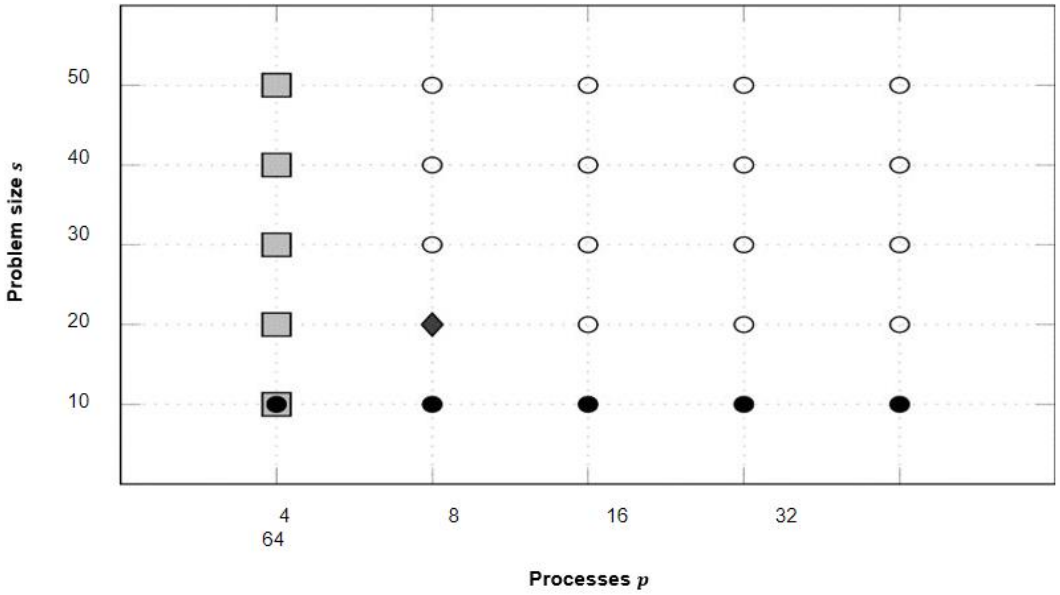
Extra-P 4.0: Sparse modeling

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



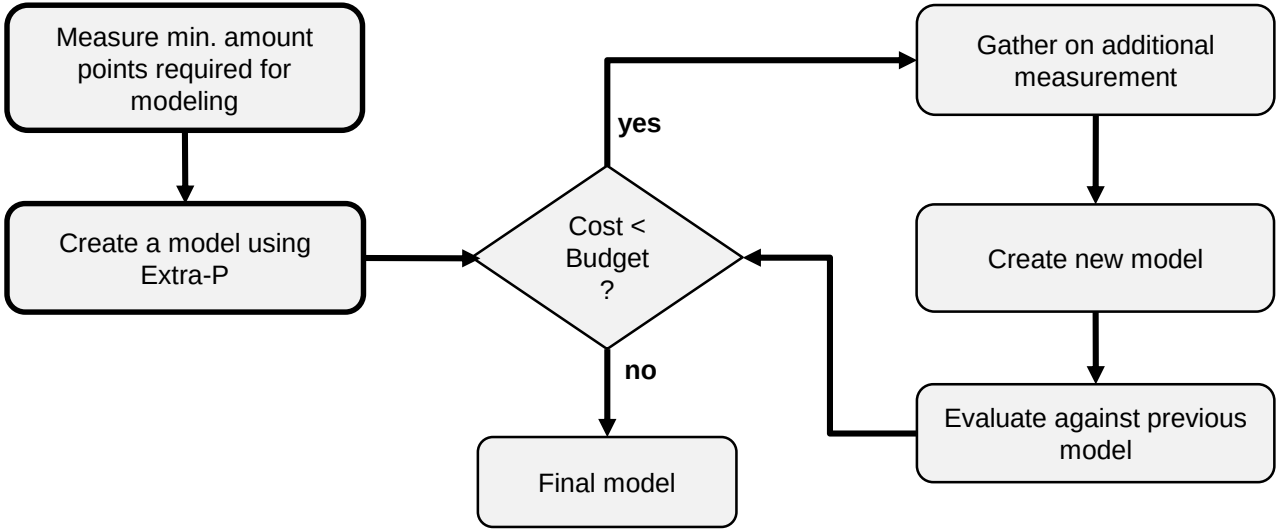
Extra-P 4.0: Sparse modeling

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



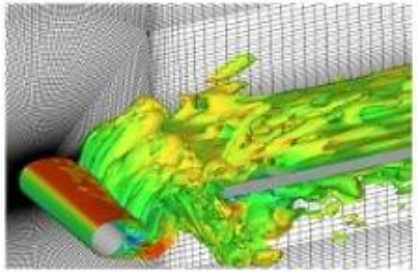
Extra-P 4.0: Sparse modeling

- Experiment configuration strategy using our heuristic guideline

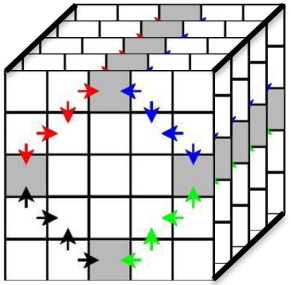


Extra-P 4.0: Sparse modeling

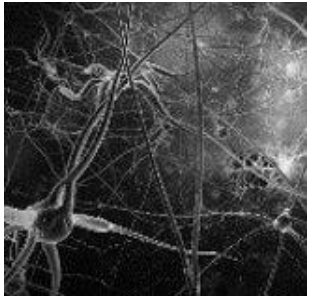
- Using sparse modeling we can reduce the average modeling cost by ~85% (on synthetic data)
- We can retain ~92% of the model accuracy (on synthetic data)
- Allows a more flexible experiment design
- **FASTEST**: 70% decrease in cost, ~2% prediction error
- **Kripke**: 99% decrease in cost, ~39% prediction error
- **Relearn**: 85% decrease in cost, ~11% prediction error



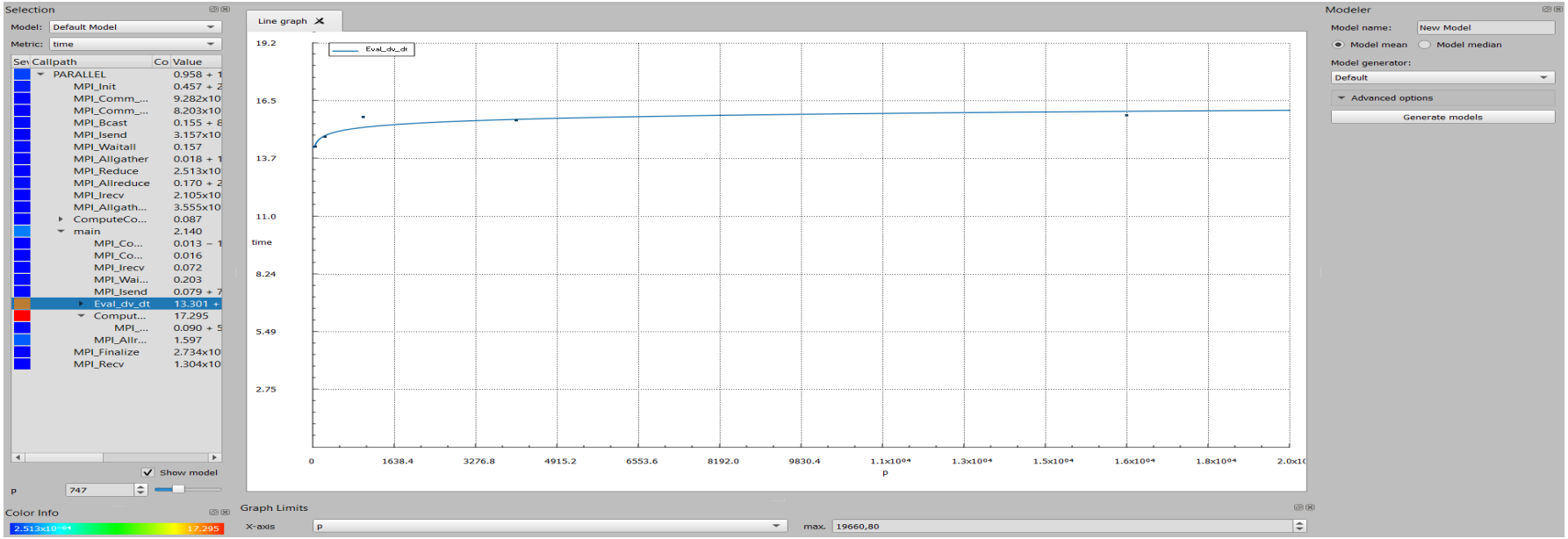
FASTEST



Kripke



RELeARN



Using Extra-P 4.0

Extra-P requirements

- Python 3.7 or higher
- numpy
- pycubexr
- marshmallow
- tqdm
- PySide2 (for GUI)
- matplotlib (for GUI)
- pyobjc-framework-Cocoa (only for GUI on macOS)

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

- Install Extra-P either on your own computer or on the cluster
- Example input data provided at: https://gigamove.rwth-aachen.de/en/download/4e812f0fe2e5a2134aa2b2af398f2d3a
- We will show you how to use Extra-P using the example data sets or your own
- Usage of the command line tool
- How to utilize the created performance models to obtain further insights

Extra-P in the tuning workshop

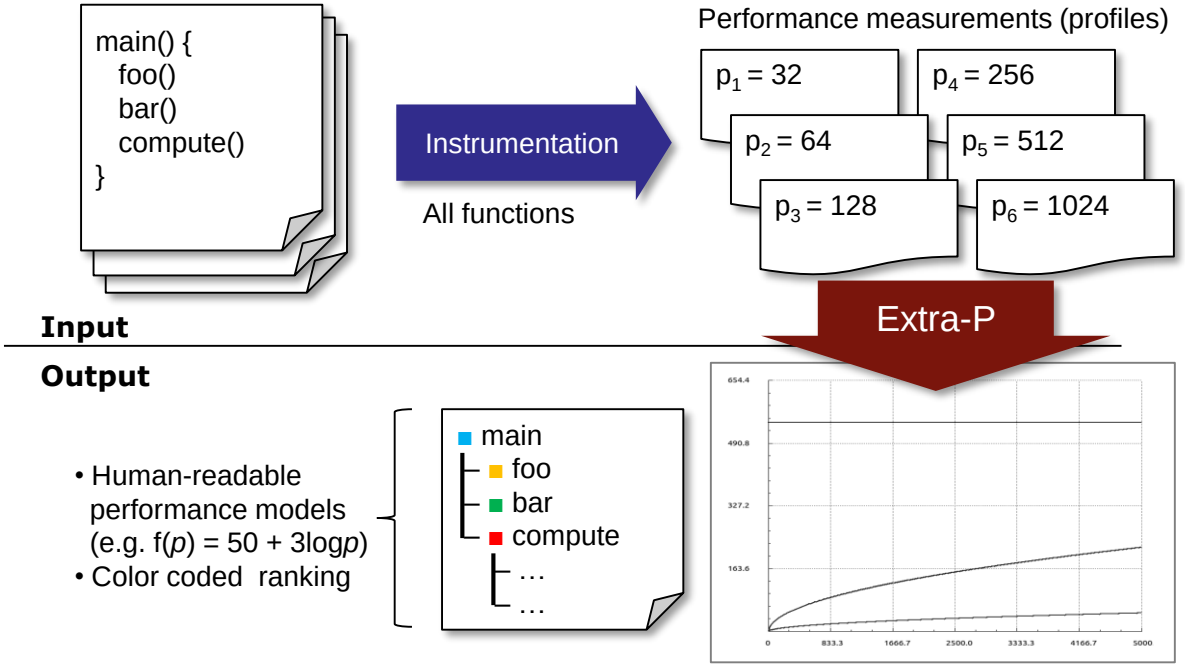
- Available at: https://github.com/extra-p/extrap

- 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

Modeling Sets of Cube Experiments

Automatic empirical performance modeling with Extra-P

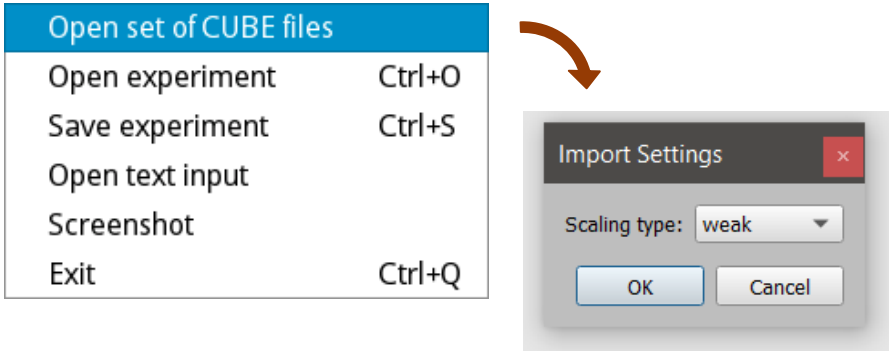


Extra-P Cube input description

- Modeling tool expects Cube files in the following format:
- <DIR>/<PREFIX><X><POSTFIX>.r<{1,..,REPS}>/<FILENAME>
- DIR, PREFIX, X, POSTFIX, REPS and FILENAME must all be defined
  - X – value of varied parameter e.g. number of processes
  - REPS – number of repeated experiments with same parameter value

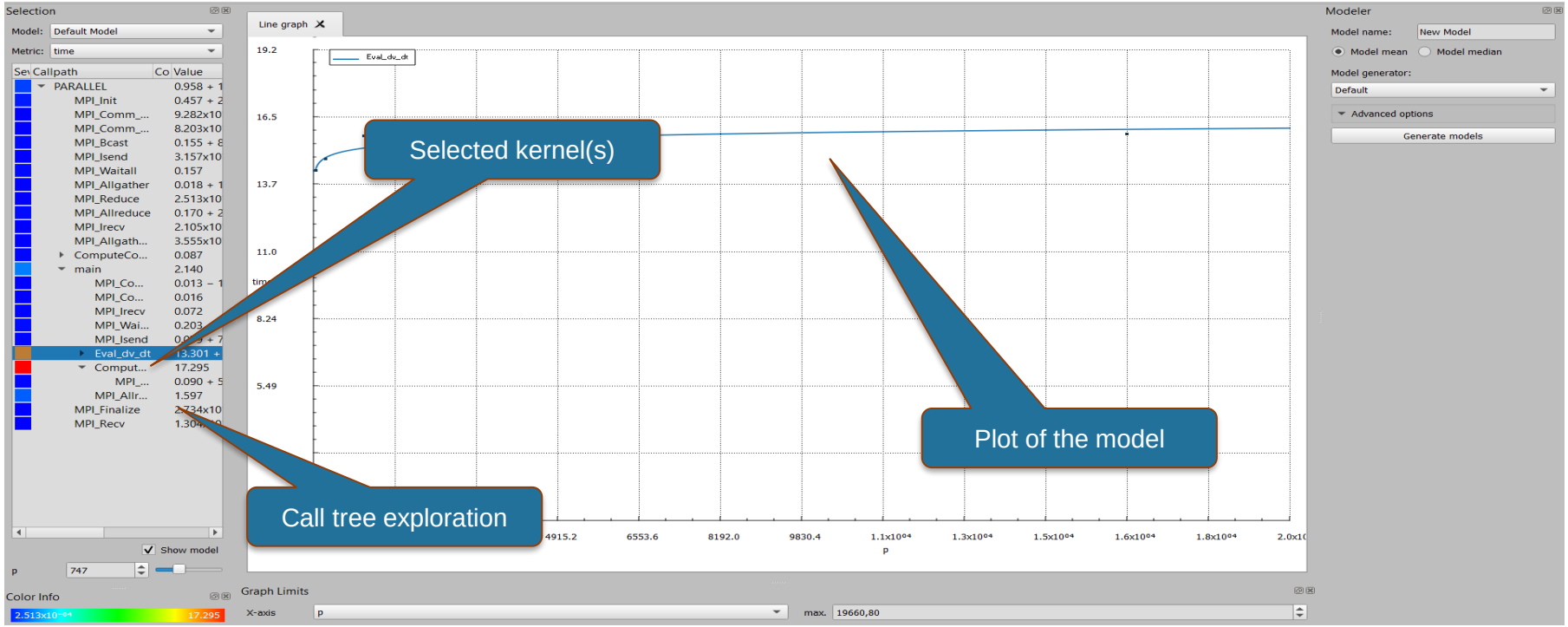
Extra-P Cube input description

<DIR>/<PREFIX><X><POSTFIX>.r<{1,..,REPS}>/<FILENAME>



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

Model: Default Model

Metric: time

Sev	Callpath	Comm	Value	RSS	Adj. R^2	SMAPE	RE
	PARALLEL		$0.958 + 1.837\times10^{-05} * p * \log_2(p)$	0.092	0.990	3.629	0
	MPI_Init		$0.457 + 2.393\times10^{-10} * p^{9/4}$	4.871×10^{-04}	0.998	1.036	0
	MPI_Comm_size		$9.282\times10^{-04} + 1.232\times10^{-09} * p * \log_2(p)$	8.406×10^{-10}	0.981	1.370	0
	MPI_Comm_rank		$8.203\times10^{-04} + 1.796\times10^{-10} * p * \log_2(p)$	1.241×10^{-11}	0.987	0.194	0
	MPI_Bcast		$0.155 + 8.580\times10^{-11} * p^{7/3}$	6.011×10^{-04}	0.997	4.769	0
	MPI_Isend		$3.157\times10^{-03} + 3.410\times10^{-05} * \log_2(p)$	1.641×10^{-08}	0.568	1.707	0
	MPI_Waitall		0.157	1.788×10^{-04}	1	2.934	0
	MPI_Allgather		$0.018 + 1.218\times10^{-09} * p^{7/4}$	9.760×10^{-06}	0.980	7.293	0
	MPI_Reduce		2.513×10^{-04}	1.119×10^{-08}	1	13.626	0
	MPI_Allreduce		$0.170 + 2.478\times10^{-04} * p^{1/3} * \log_2(p)$	7.463×10^{-04}	0.790	6.737	0
	MPI_Irecv		$2.105\times10^{-03} + 5.098\times10^{-05} * \log_2(p)$	2.552×10^{-08}	0.693	3.059	0
	MPI_Allgatherv		$3.555\times10^{-04} + 2.418\times10^{-07} * p^{5/4}$	2.974×10^{-06}	0.997	8.098	0
	▸ ComputeCornerForces		0.087	1.425×10^{-06}	1	0.557	0
	▾ main		2.140	4.007×10^{-03}	1	1.264	0
	MPI_Comm_rank		$0.013 - 1.971\times10^{-05} * \log_2(p)$	9.469×10^{-09}	0.319	0.350	0
	MPI_Comm_size		0.016	1.771×10^{-08}	1	0.341	0
	MPI_Irecv		0.072	9.185×10^{-06}	1	1.582	0
	MPI_Waitall		0.203	0.019	1	22.235	0
	MPI_Isend		$0.079 + 7.508\times10^{-04} * \log_2(p)$	5.459×10^{-06}	0.695	1.291	0
	▸ Eval_dv_dt		$13.301 + 0.190 * \log_2(p)$	0.803	0.392	2.763	0
	▾ ComputeCornerForces		17.295	0.038	1	0.340	0
	MPI_Reduce		$0.090 + 5.990\times10^{-09} * p^{5/4} * \log_2(p)$	4.877×10^{-05}	0.684	3.088	0
	[illegible]		[illegible]	0.051	1	4.829	0
	MPI_Finalize		2.734×10^{-04}	5.817×10^{-08}	1	28.617	0
	MPI_Recv		$1.304\times10^{-04} + 4.542\times10^{-07} * p$	5.099×10^{-06}	0.829	48.915	0

Show model

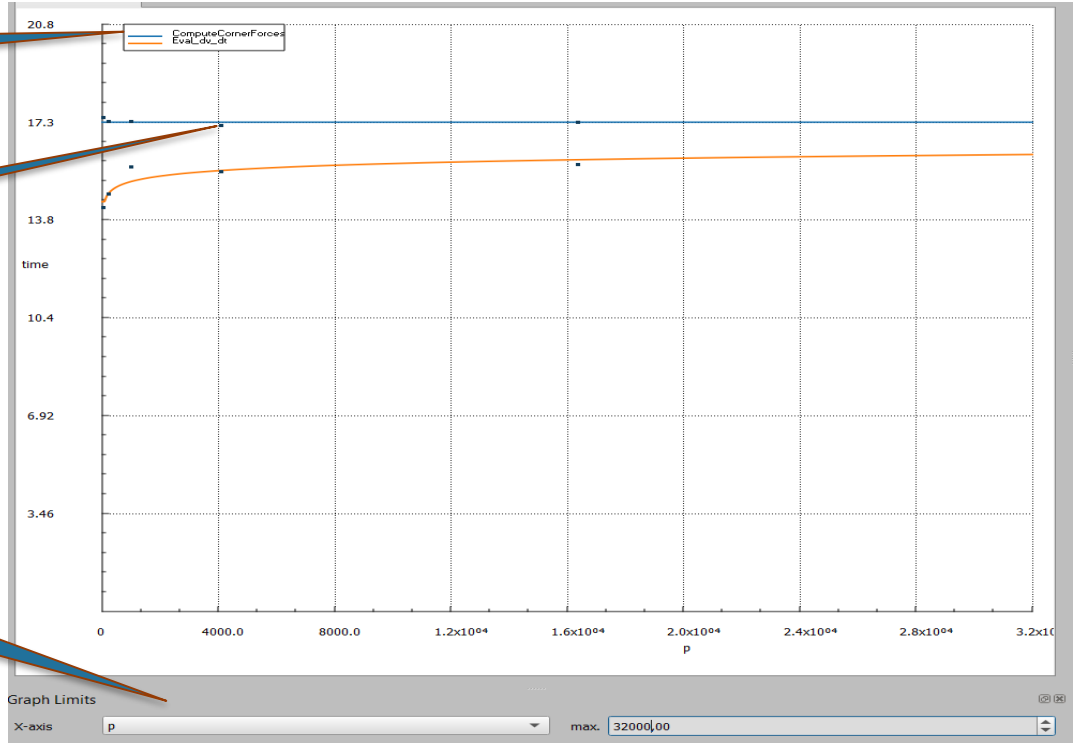
p 747

Extra-P model view

Models selected in the Call path view

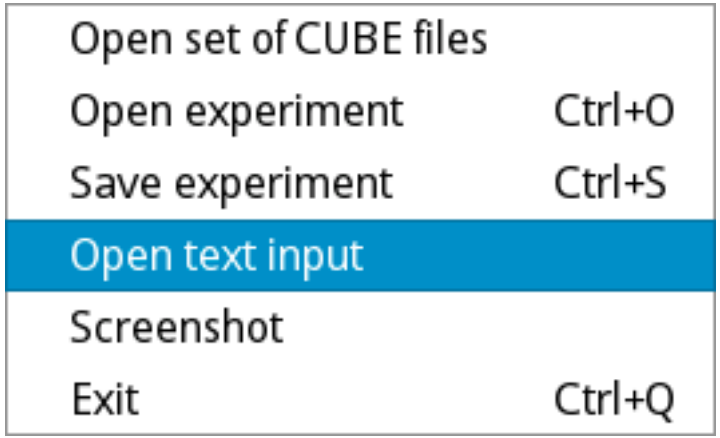
Measurement values

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



Modeling Measurements from a Text File

Choose input file



Select input file in the GUI

Extra-P input in text form

- Useful when no CUBE files are available or when a small data set must be modeled

```
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
```

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
```

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 points

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

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

Utilizing 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 /input.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)

Feedback

- 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*

Questions


WHAT?
WHY?
WHERE?
WHEN?
WHO?
HOW?
WHICH?
WHOSE?
WHERE? WHO?
HOW? WHO?
WHO? WHERE? WHAT? HOW?
WHAT? HOW? WHY?
HOW?