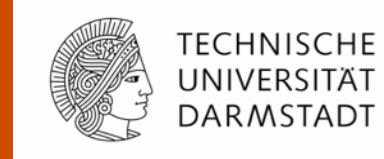


# The price performance of performance models



Felix Wolf, Technical University of Darmstadt

IEEE Cluster Conference 2020, Kobe, Japan



# Acknowledgement

## TU Darmstadt

- Yannick Berens
- Alexandru Calotoiu
- Alexander Geiß
- Alexander Graf
- Daniel Lorenz
- Benedikt Naumann
- Thorsten Reimann
- Sebastian Rinke
- Marcus Ritter
- Sergei Shudler

## ETH Zurich

- Alexandru Calotoiu
- Marcin Copik
- Tobias Grosser
- Torsten Hoefler
- Nicolas Wicki

## LLNL

- David Beckingsale
- Christopher Earl
- Ian Karlin
- Martin Schulz

## FZ Jülich

- Alexandre Strube



Bundesministerium  
für Bildung  
und Forschung

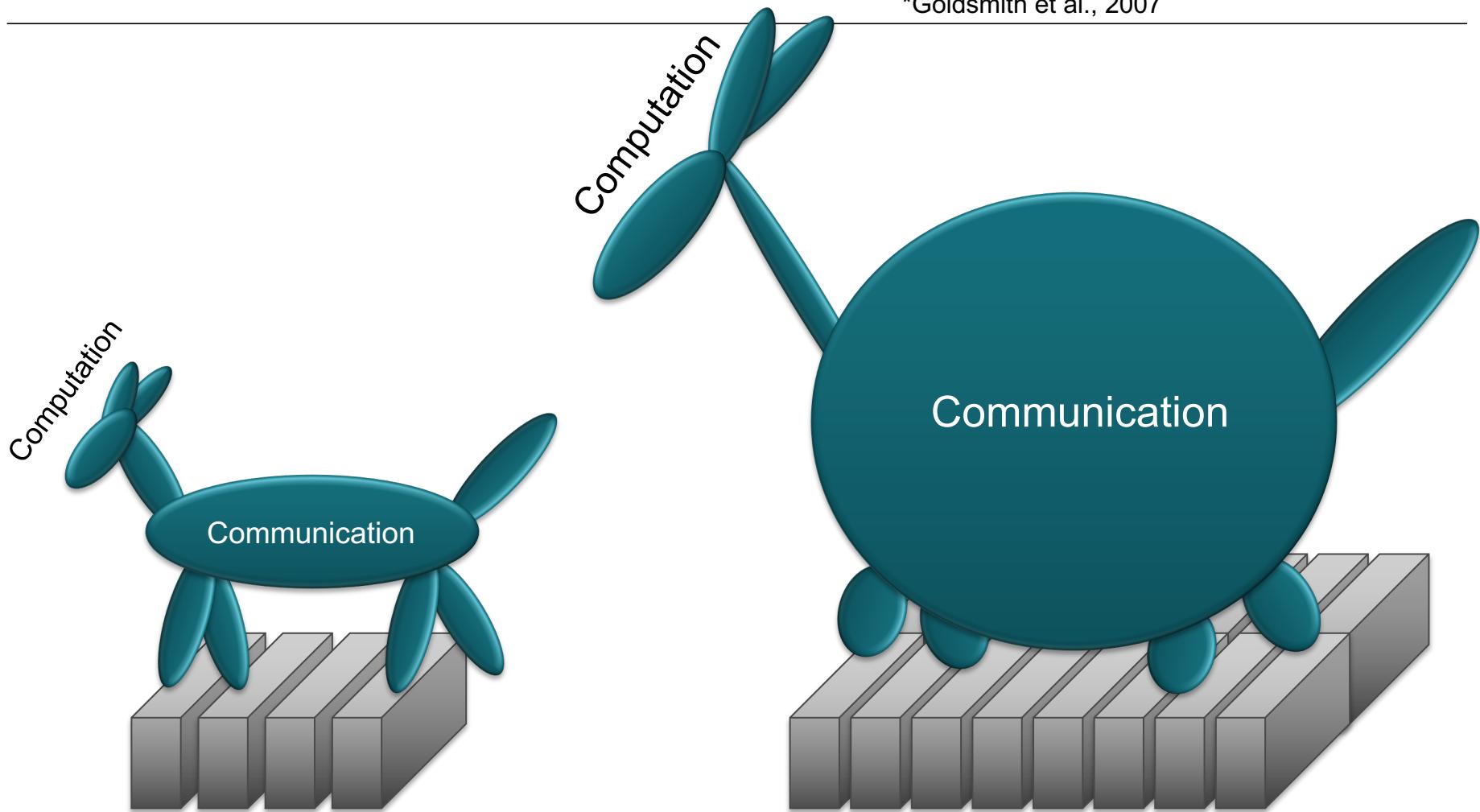


# Scaling your code can harbor *performance surprises*\*...



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

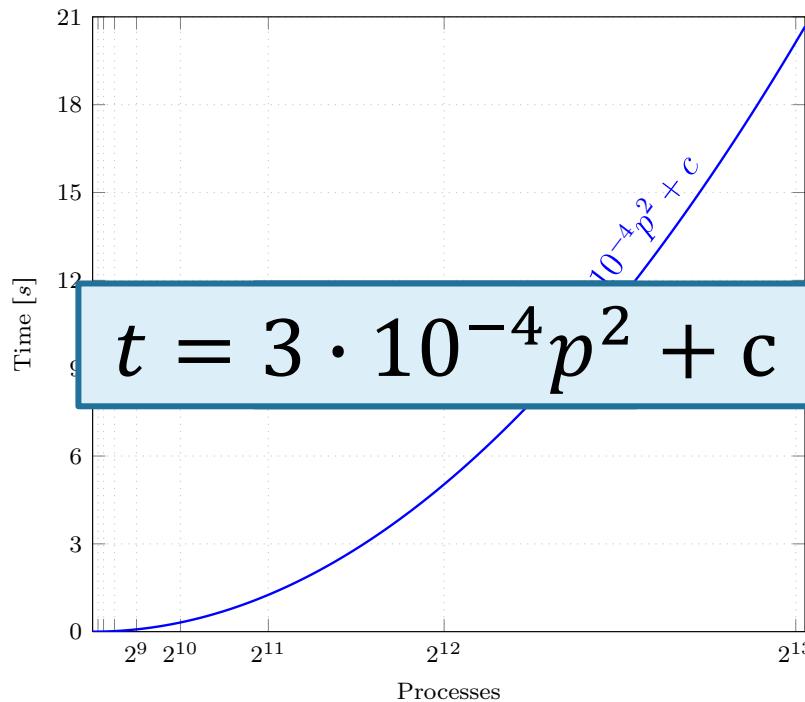
\*Goldsmith et al., 2007



# Performance model



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

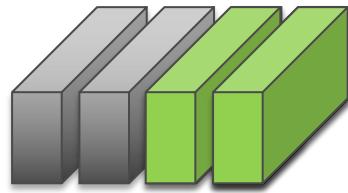


Analytical (i.e., manual) creation challenging for entire programs

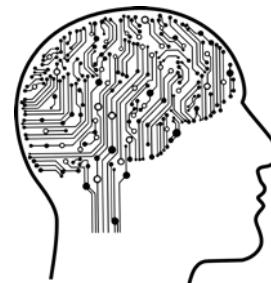
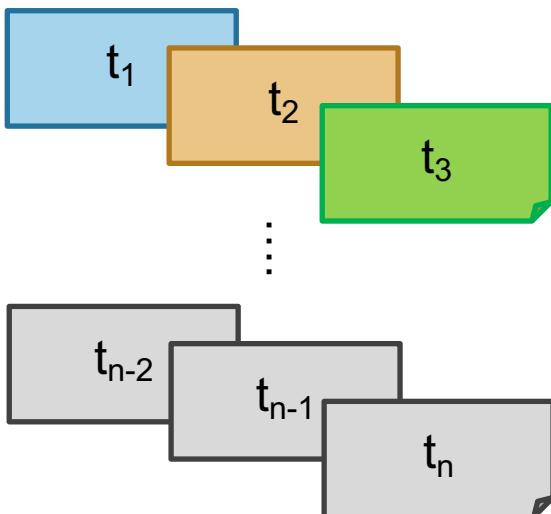


- Incomplete coverage
- Laborious, difficult

# Empirical performance modeling



Performance measurements  
with different execution  
parameters  $x_1, \dots, x_n$



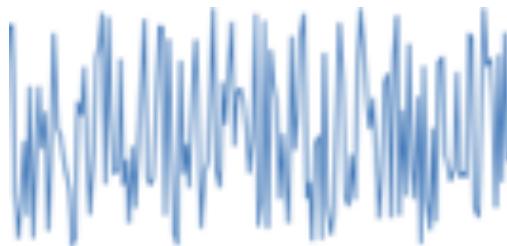
$$t = f(x_1, \dots, x_n)$$

Alternative metrics:  
FLOPs, data volume...

# Challenges



Applications



Run-to-run variation / noise



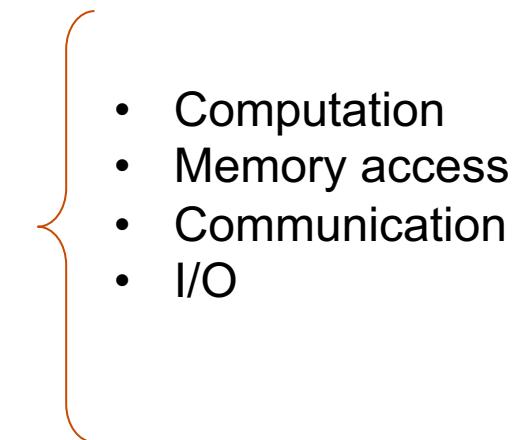
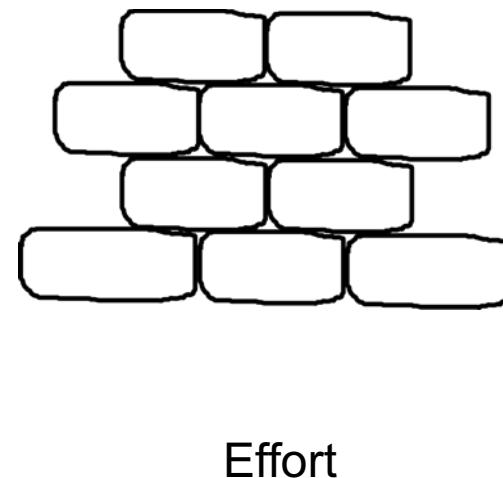
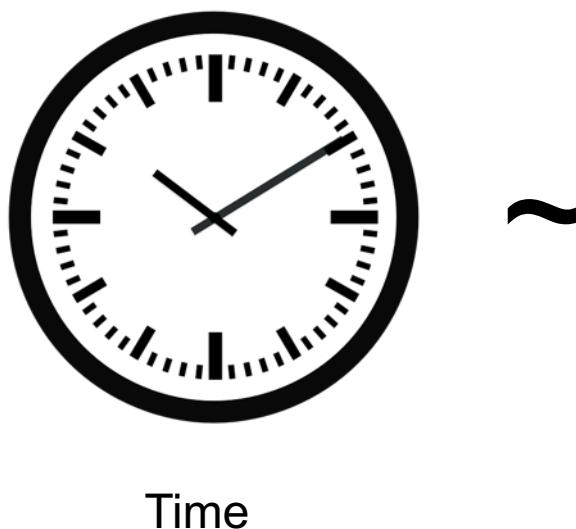
System



Cost of the required experiments

# How to deal with noisy data

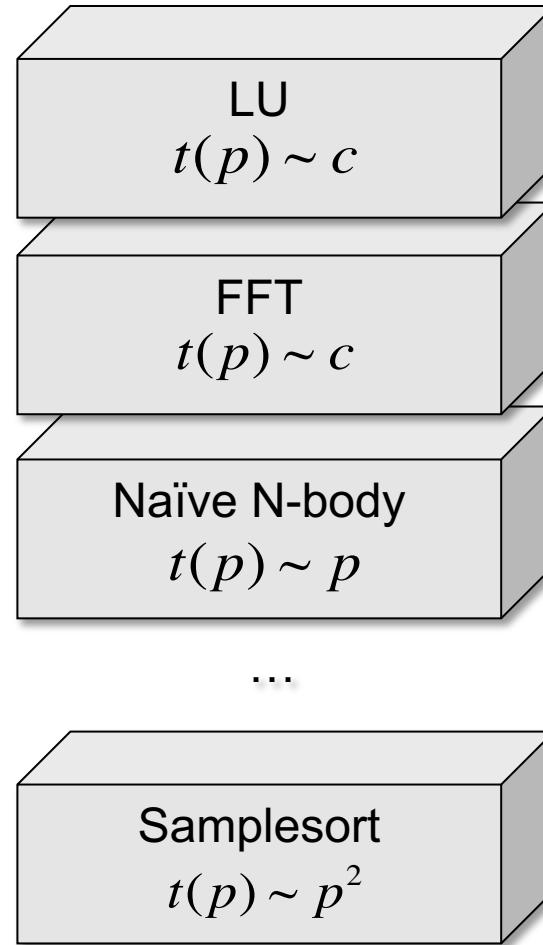
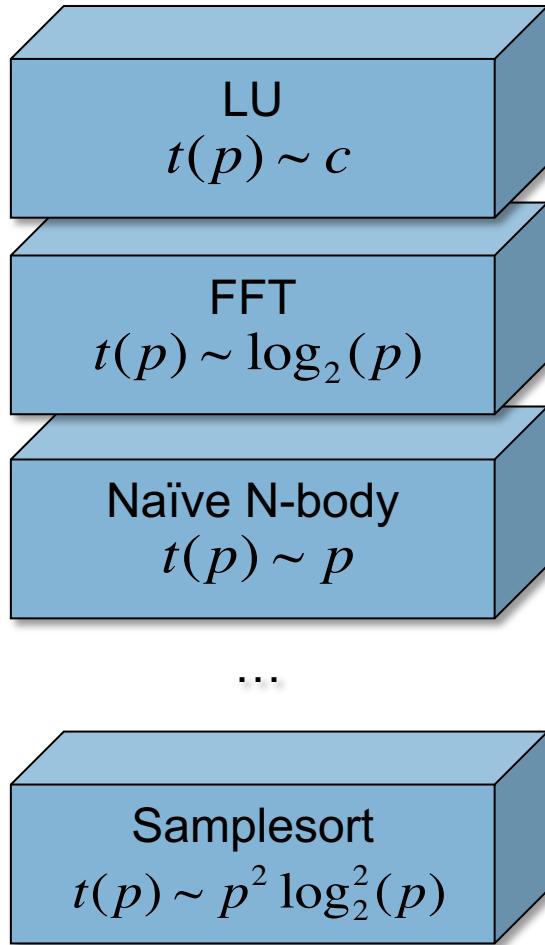
- Introduce **prior** into learning process
  - Assumption about the probability distribution generating the data



# Typical algorithmic complexities in HPC



Computation

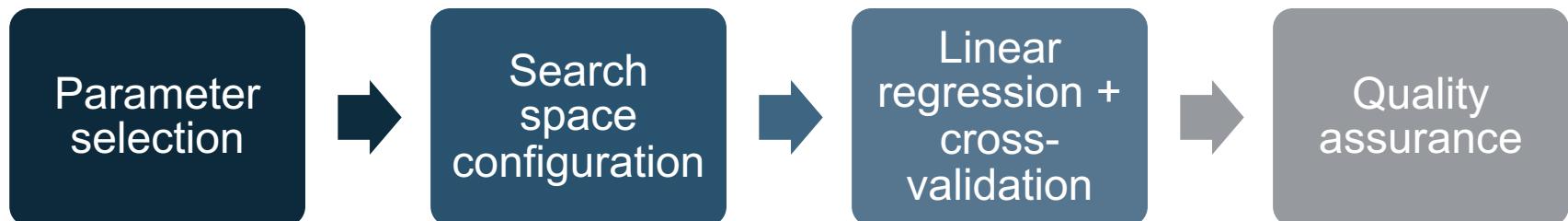


Communication

# Performance model normal form (PMNF)

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

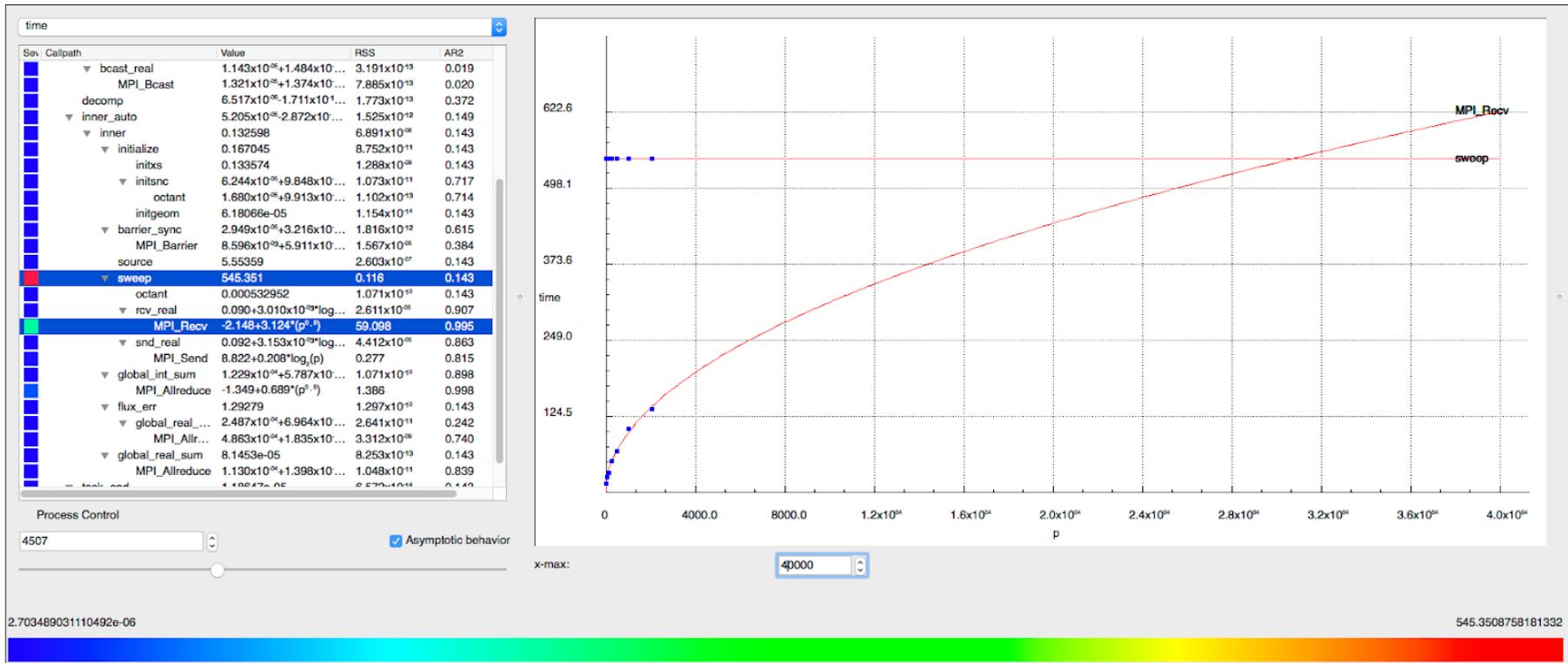
Single parameter  
[Calotoiu et al., SC13]



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

Multiple parameters [Calotoiu et al., Cluster'16]

Heuristics to  
reduce  
search space



## New BSD license

<http://www.scalasca.org/software/extra-p/download.html>

# MPI implementations

[Shudler et al., IEEE TPDS 2019]



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Platform	Juqueen	Juropa	Piz Daint	
Allreduce [s]		<b>Expectation: <math>O(\log p)</math></b>		
Model	$O(\log p)$	$O(p^{0.5})$	$O(p^{0.67} \log p)$	
R <sup>2</sup>	0.87	0.99	0.99	
Match	✓	~	✗!	
Comm_dup [B]		<b>Expectation: <math>O(1)</math></b>		
Model	2.2e5	256	$3770 + 18p$	
R <sup>2</sup>	1	1	0.99	
Match	✓	✓	✗	

# Kripke - example w/ multiple parameters



## SweepSolver

Main computation kernel

Expectation – Performance depends on  
**problem size**

$$t \sim d \cdot g$$

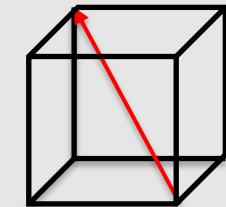
Actual model:

$$t = 5 + d \cdot g + 0.005 \cdot \sqrt[3]{p} \cdot d \cdot g$$

Kernels must wait on  
each other

## MPI\_Testany

Main **communication**  
kernel: 3D wave-front  
communication pattern



Expectation – Performance depends on  
**cubic root of process count**

$$t \sim \sqrt[3]{p}$$

Actual model:

$$t = 7 + \sqrt[3]{p} + 0.005 \cdot \sqrt[3]{p} \cdot d \cdot g$$

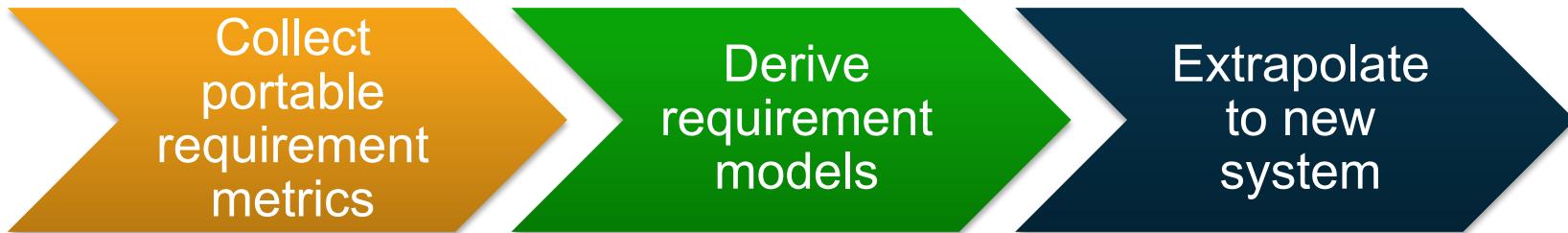
**Smaller compounded effect discovered**

\*Coefficients have been rounded for convenience

# Lightweight requirements engineering for (exascale) co-design



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



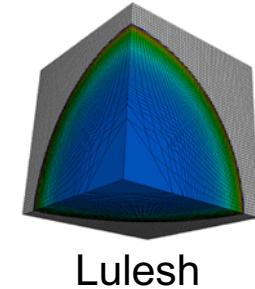
Resource	Metric (per process)
Memory footprint	# Bytes used (resident memory size)
Computation	# Floating-point operations (#FLOP)
Network communication	# Bytes sent / received
Memory access	# Loads / stores; stack distance

Counters often more noise resilient than time



# Application demands for different resources scale differently

#Bytes used	$10^5 \cdot n \log n$
#FLOP	$10^5 \cdot n \log n \cdot p^{0.25} \log p$
#Bytes sent & received	$10^3 \cdot n \cdot p^{0.25} \log p$
#Loads & stores	$10^5 \cdot n \log n \cdot \log p$
Stack distance	Constant



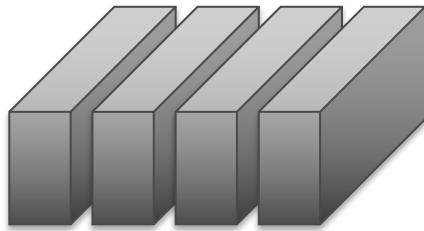
Models are per process

$p$  – Number of processes

$n$  – Problem size per process

Calculate relative changes of resource demand by scaling  $p$  and  $n$

- $n$  is a function of the memory size
- $p$  is a function of the number of cores / sockets



Given a budget and a set of applications, how can we best invest in upgrades for a given hardware system?

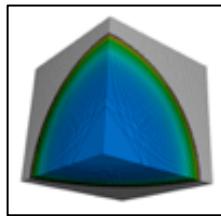
## Examples

- Double the racks
- Double the sockets
- Double the memory

[Calotoiu et al., Cluster'18]

Three upgrades – summary

Best option



LULESH

Worst option

	Apps	Kripke	LULESH	MILC	Relearn	icoFoam	Baseline
Ratios							
<b>System Upgrade A: Double the racks</b>							
Problem size per process	1	1	1	1	1	0.5	1
Overall problem size	2	2	2	2	1	1	2
Computation	1	1.2	1.2	1	1	0.5	1
Communication	1	1.2	1.2	1	1	0.7	1
Memory accesses	2	1.2	1.2	2.8	2	0.7	1
<b>System Upgrade B: Double the sockets</b>							
Problem size per process	0.5	0.5	0.5	0.5	0.3	0.3	0.5
Overall problem size	1	1	1	1	0.5	0.6	1
Computation	0.5	0.6	0.6	0.5	0.3	0.2	0.5
Communication	0.5	0.6	0.6	0.5	0.3	0.3	0.5
Memory accesses	0.5	1	1	1.4	1	0.5	0.5
<b>System Upgrade C: Double the memory</b>							
Problem size per process	2	1.4	1.4	2	4	1.4	2
Overall problem size	2	1.4	1.4	2	4	1.4	2
Computation	2	1.4	1.4	2	4	1.7	2
Communication	2	1.4	1.4	2	4	1.4	2
Memory accesses	2	1.4	1.4	2	4	1.4	2

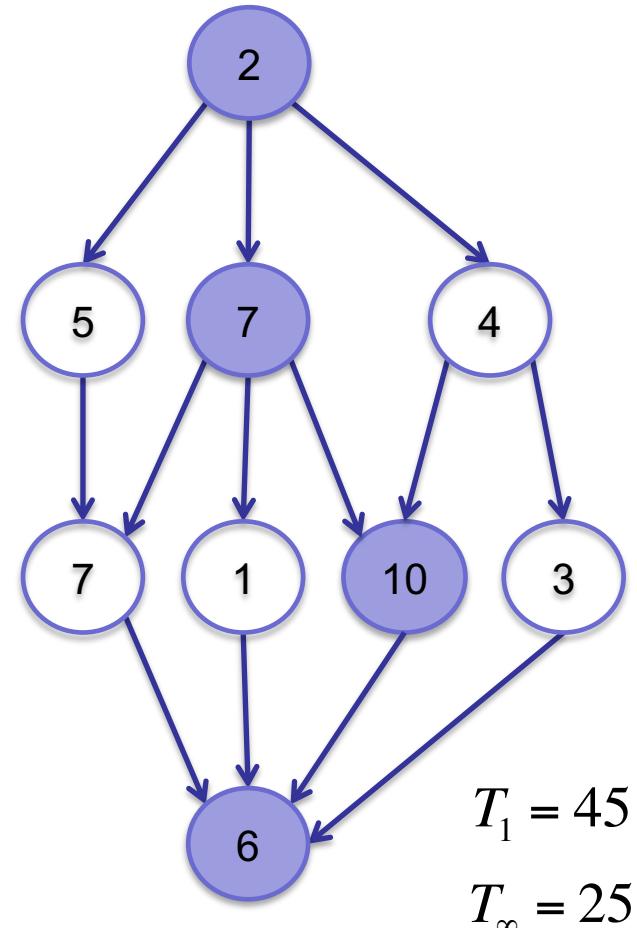
# Task-graph modeling

[Shudler et al., PPoPP'17]



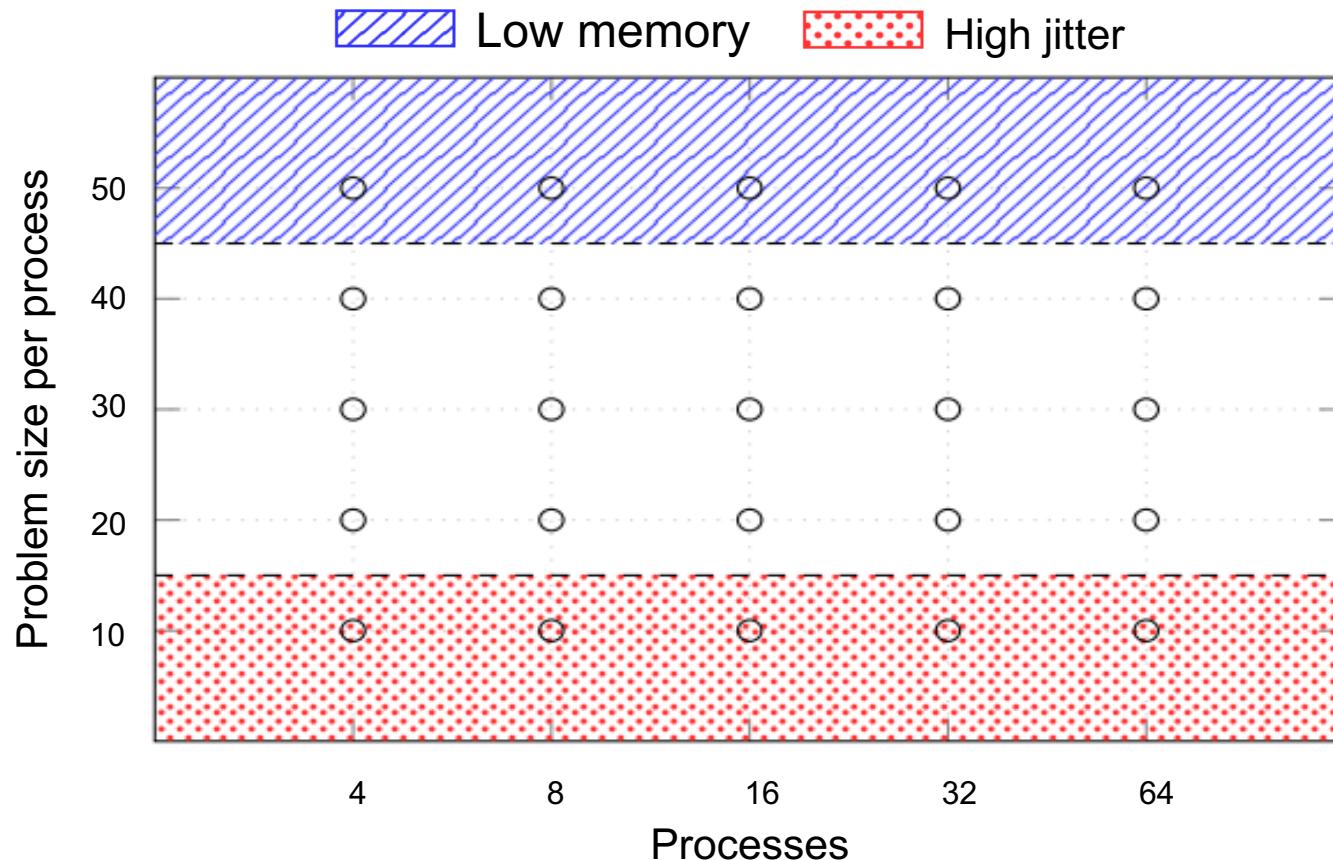
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Nodes – tasks, edges – dependencies
- $p, n$  – processing elements, input size
- $T_1(n)$  – all the task times (*work*)
- $T_\infty(n)$  – longest path (*depth*)
- $\pi(n) = \frac{T_1(n)}{T_\infty(n)}$  – average parallelism



# Experiments can be expensive

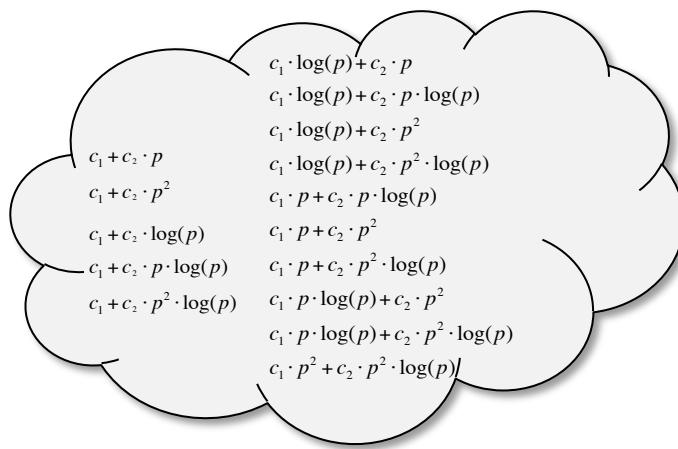
Need  $5^{(m+1)}$  experiments,  $m = \#$ parameters



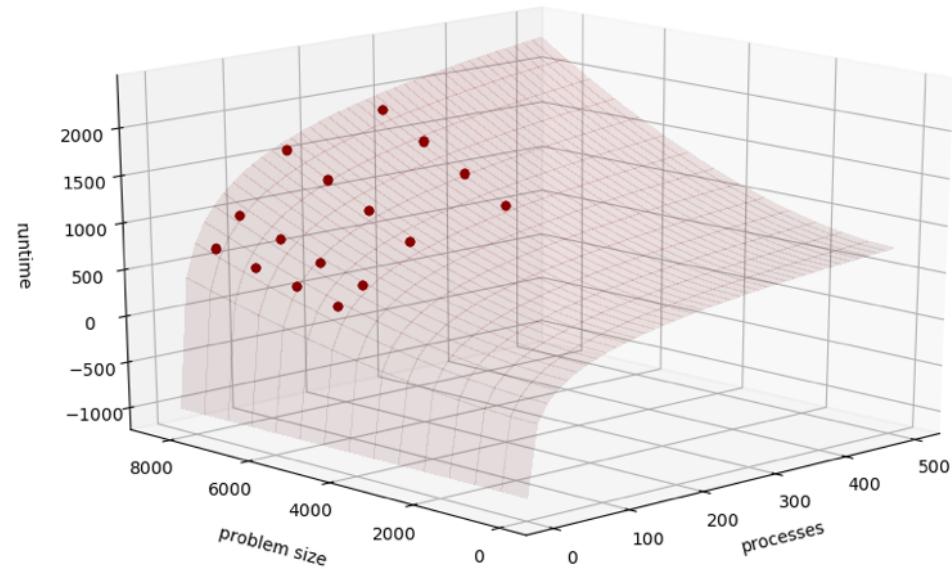
# Multi-parameter modeling in Extra-P

Find best single-parameter model

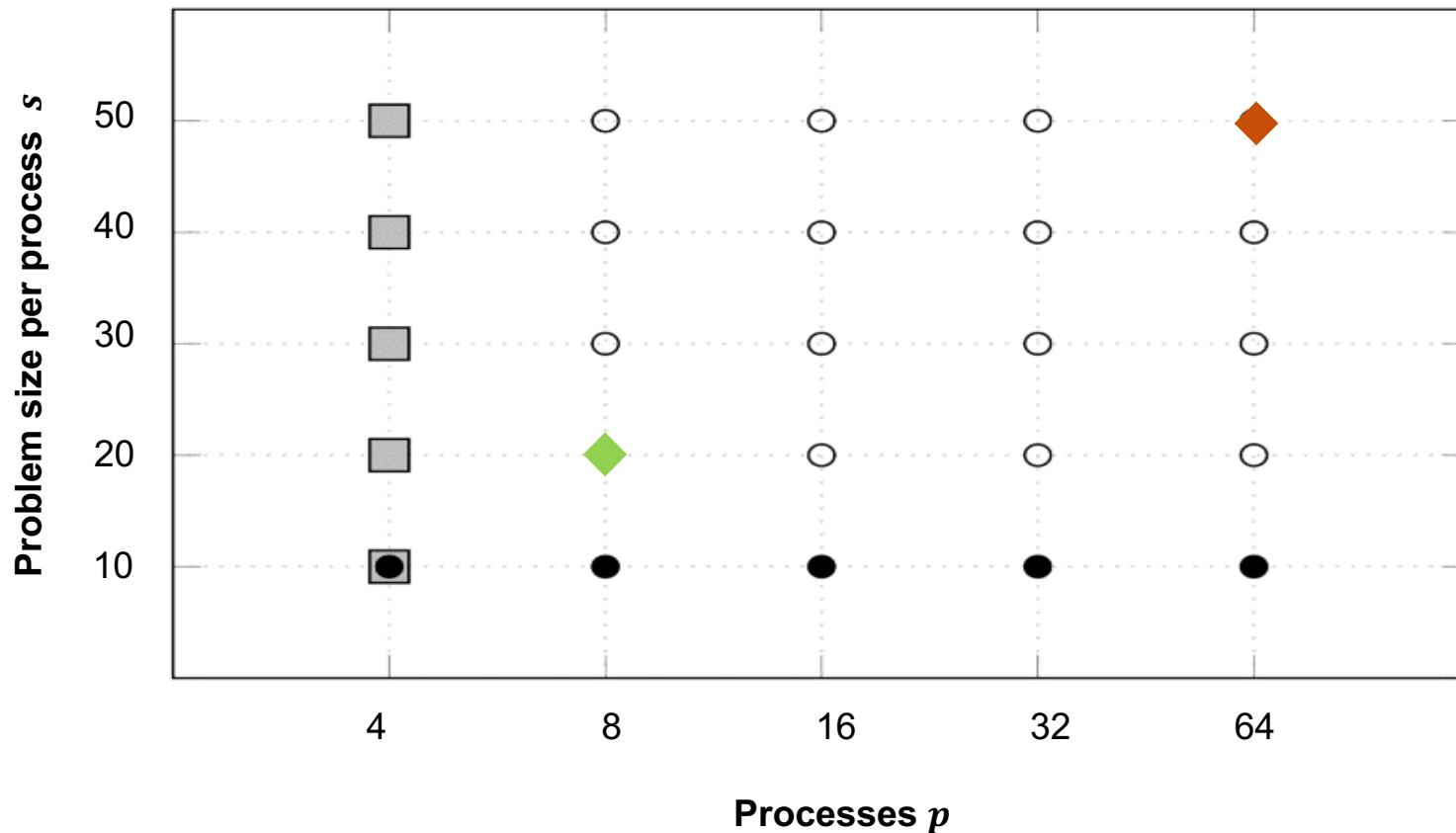
Combine them in the most plausible way (+, \*, none)



Generation of candidate models and selection of best fit



# How many data points do we really need?

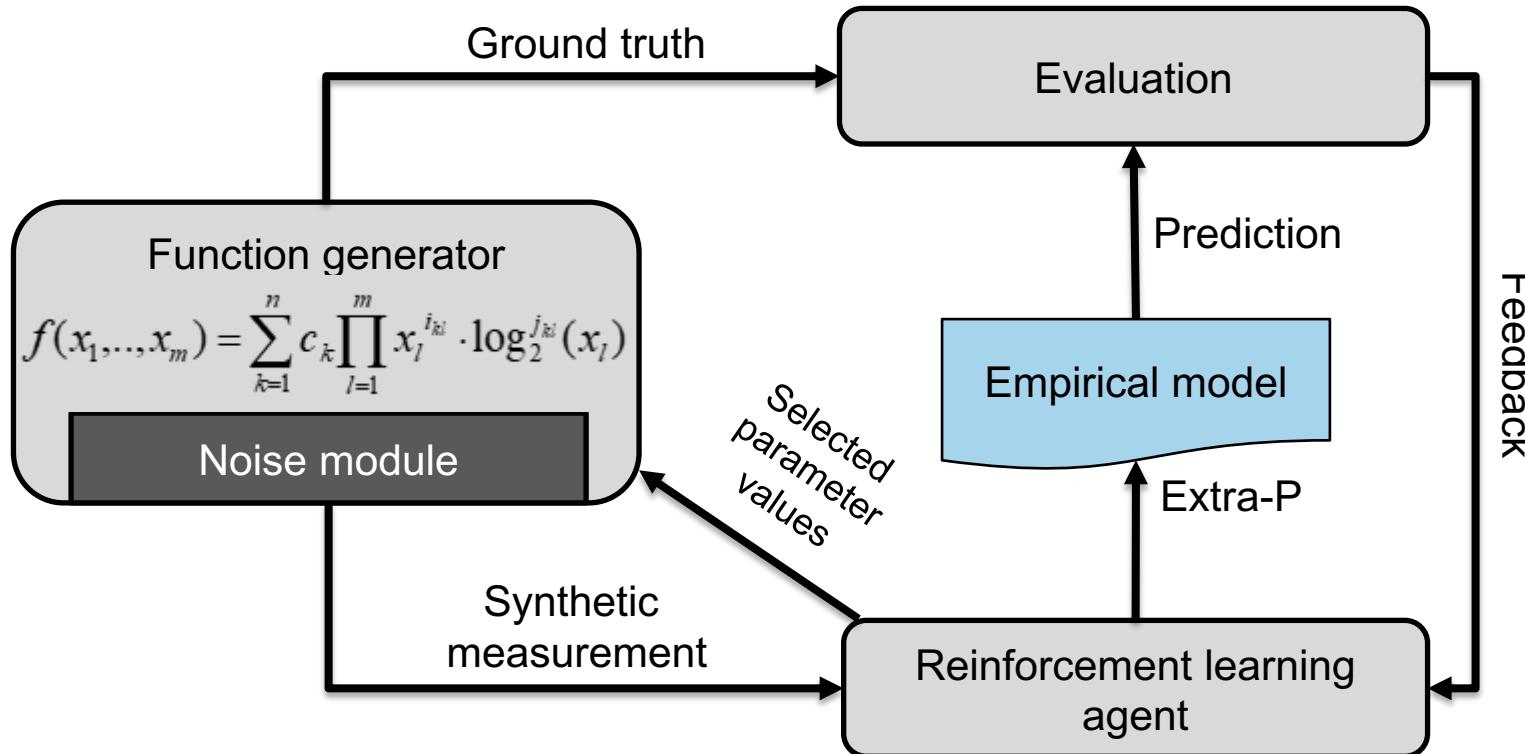


# Learning cost-effective sampling strategies

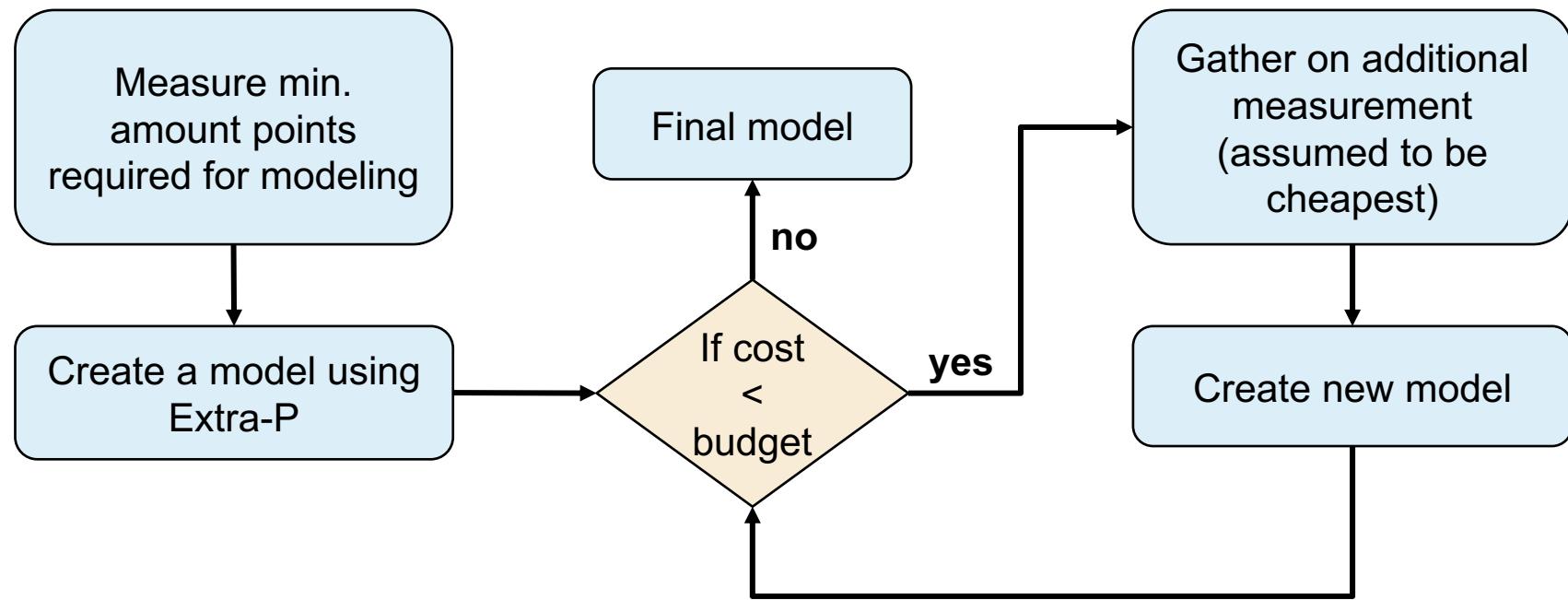
[Ritter et al., IPDPS'20]



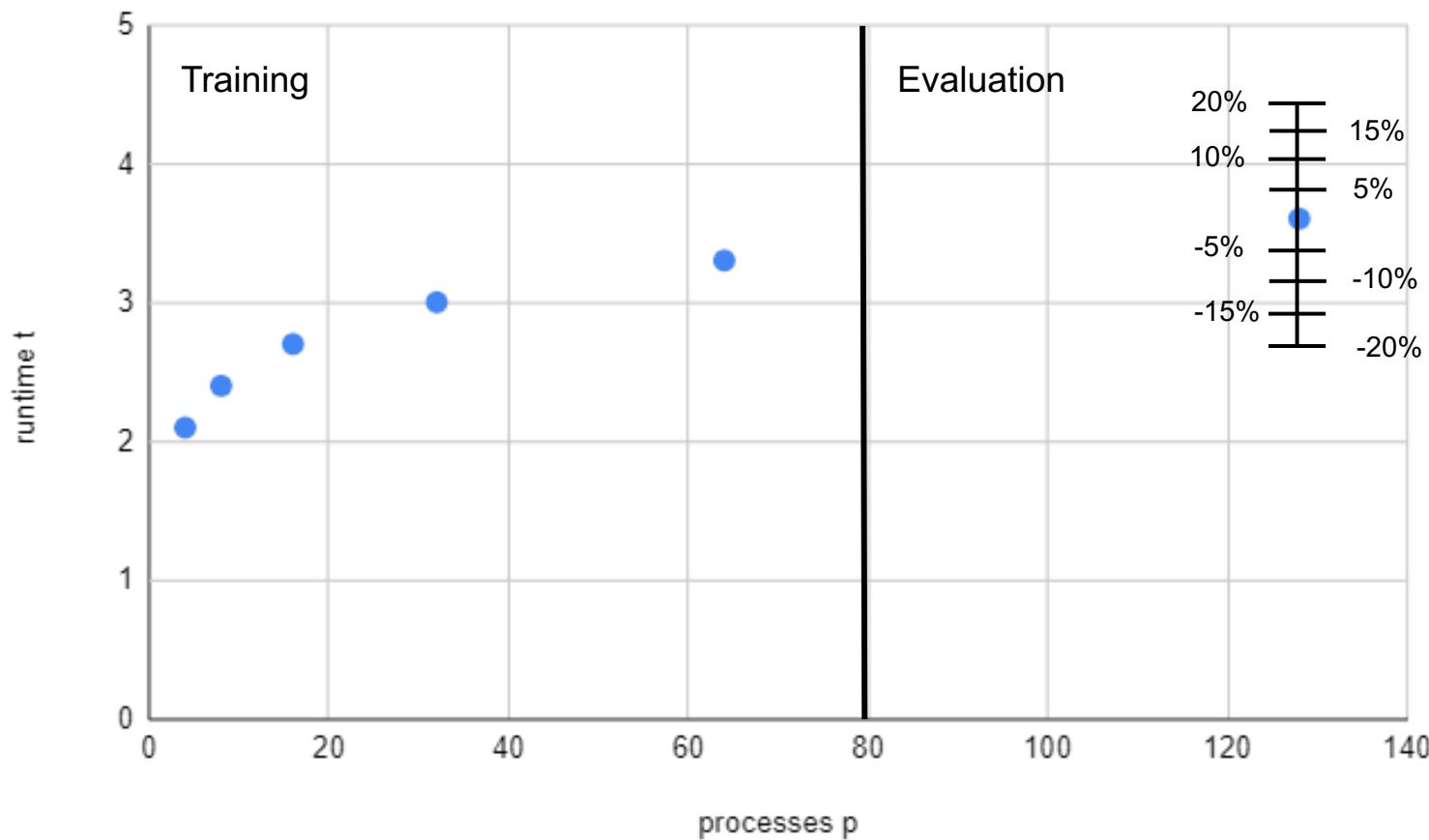
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



# Heuristic parameter-value selection strategy



# Synthetic data evaluation

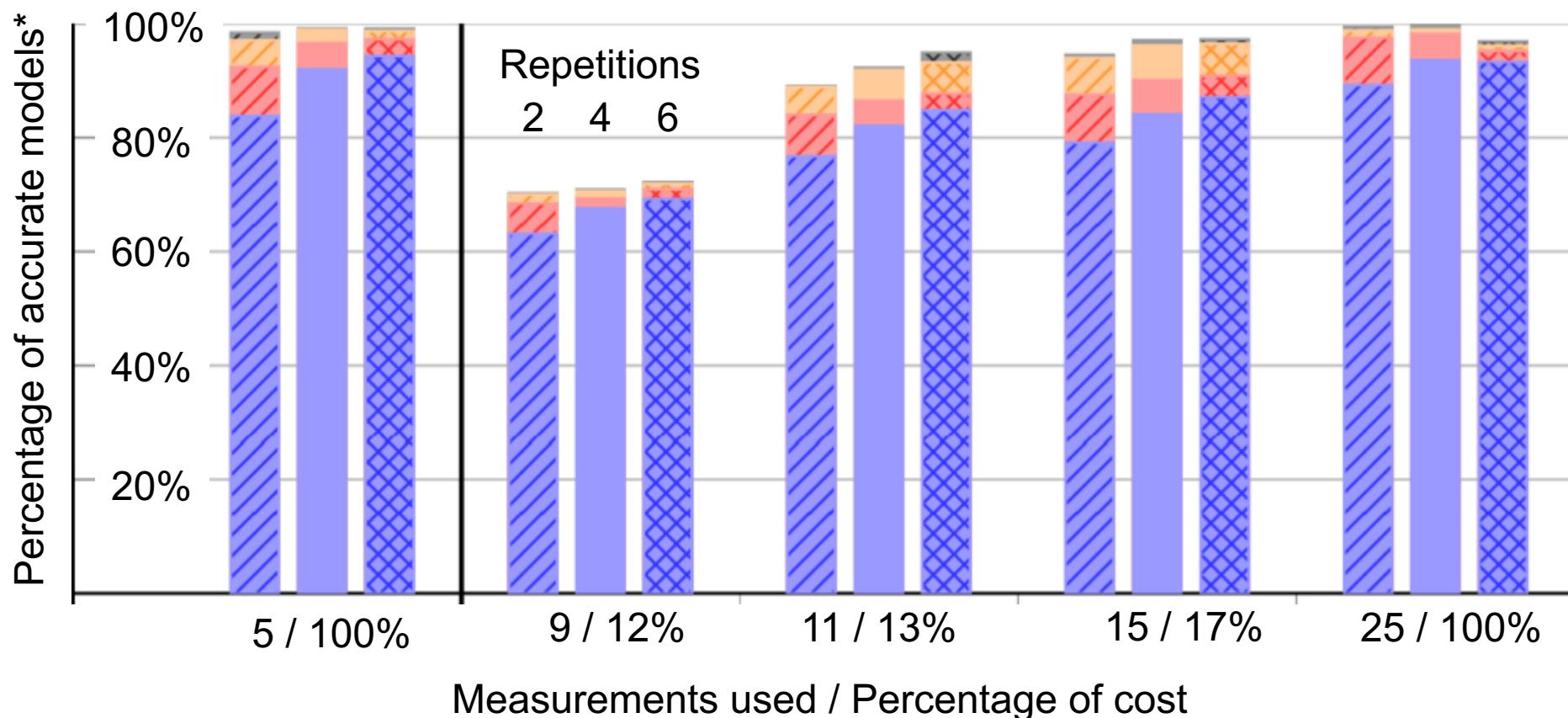


# Synthetic evaluation results



1 parameter, 5% noise

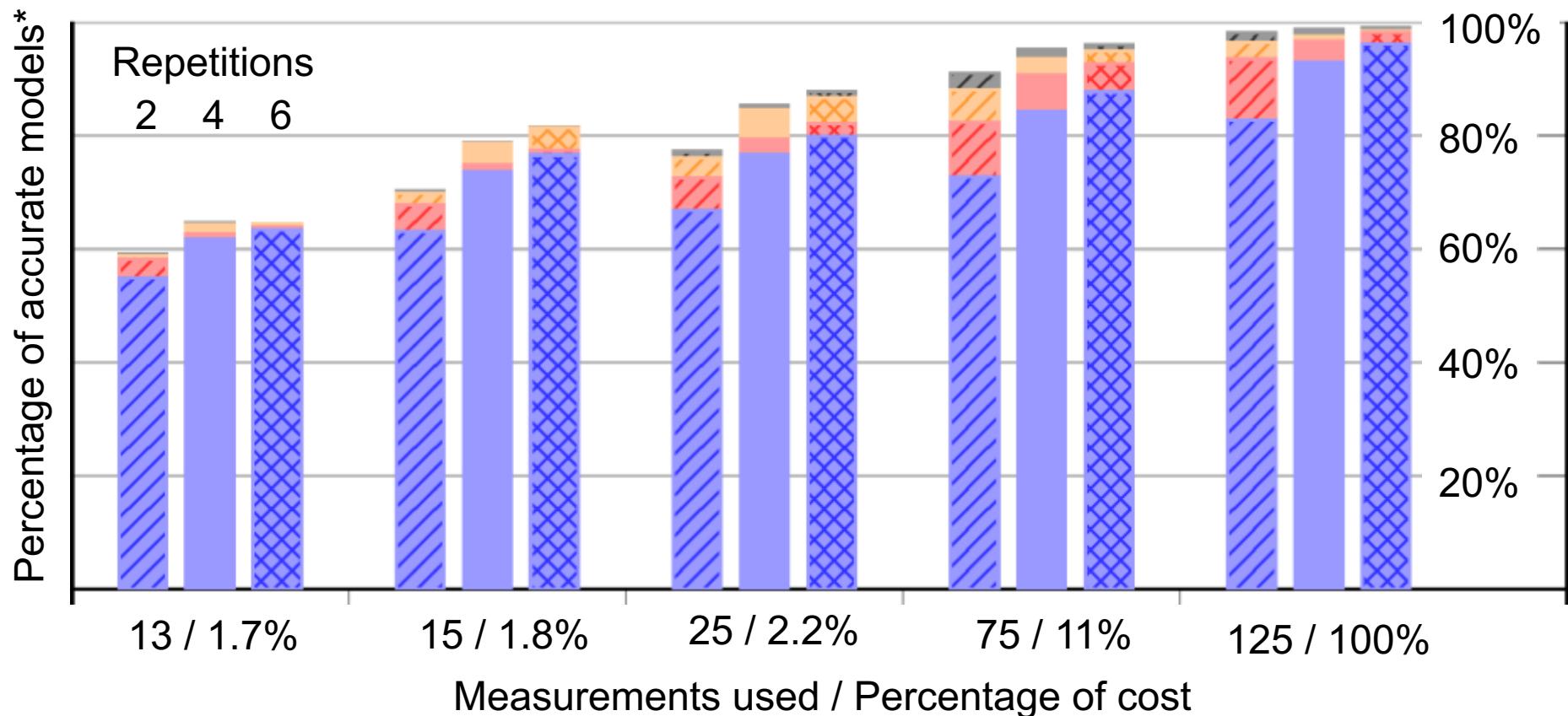
2 parameters, 5% noise



# Synthetic evaluation results



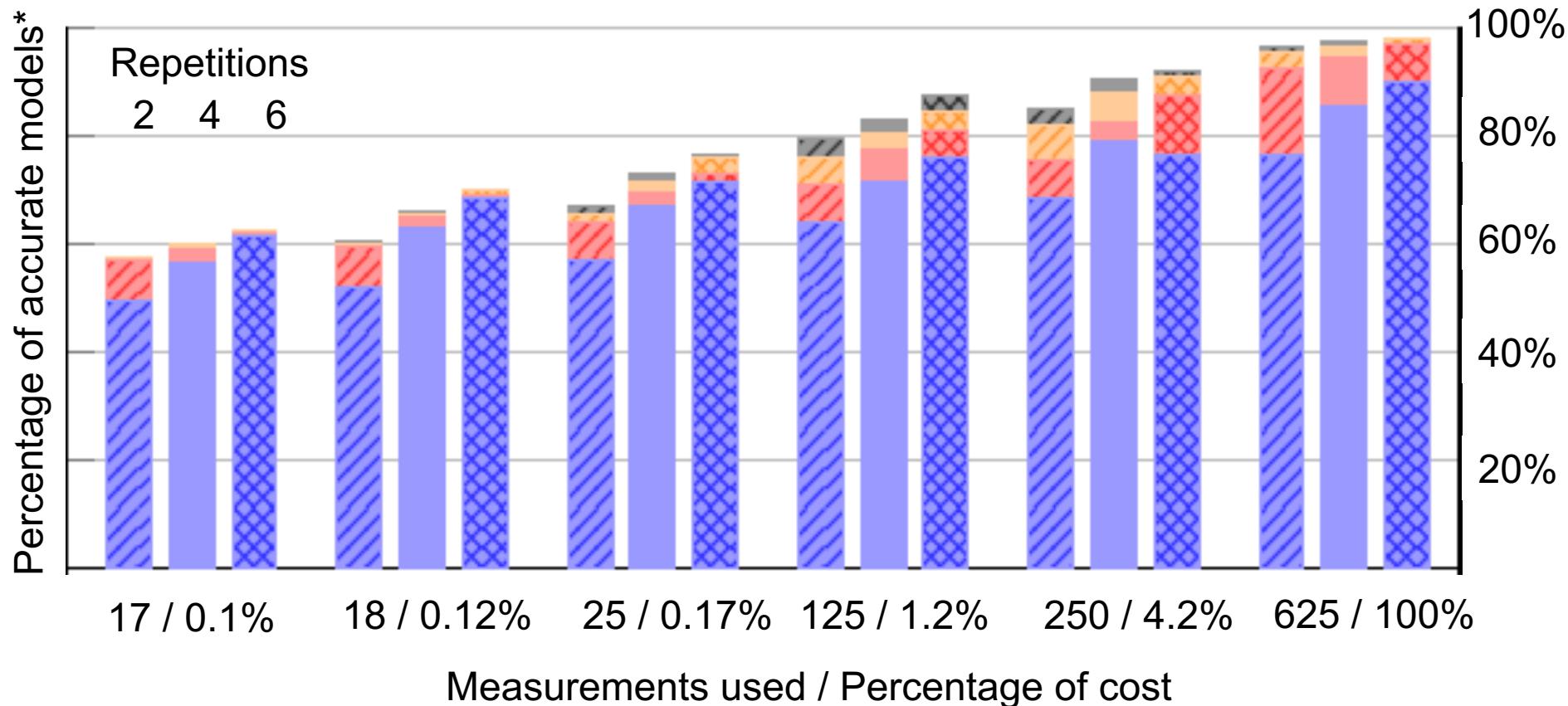
3 parameters, 5% noise



# Synthetic evaluation results



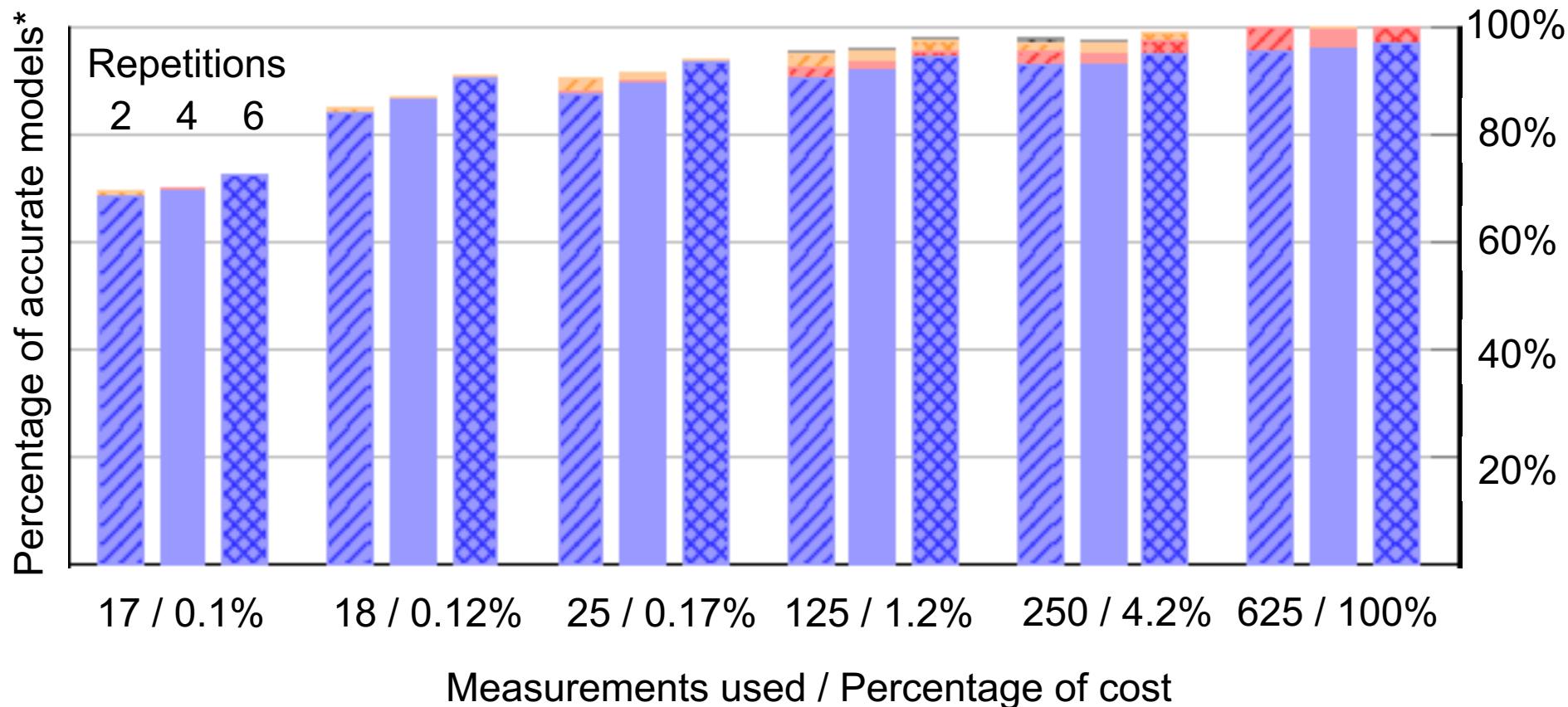
4 parameters, 5% noise



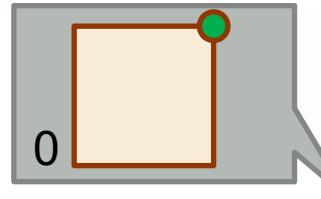
# Synthetic evaluation results



4 parameters, 1% noise



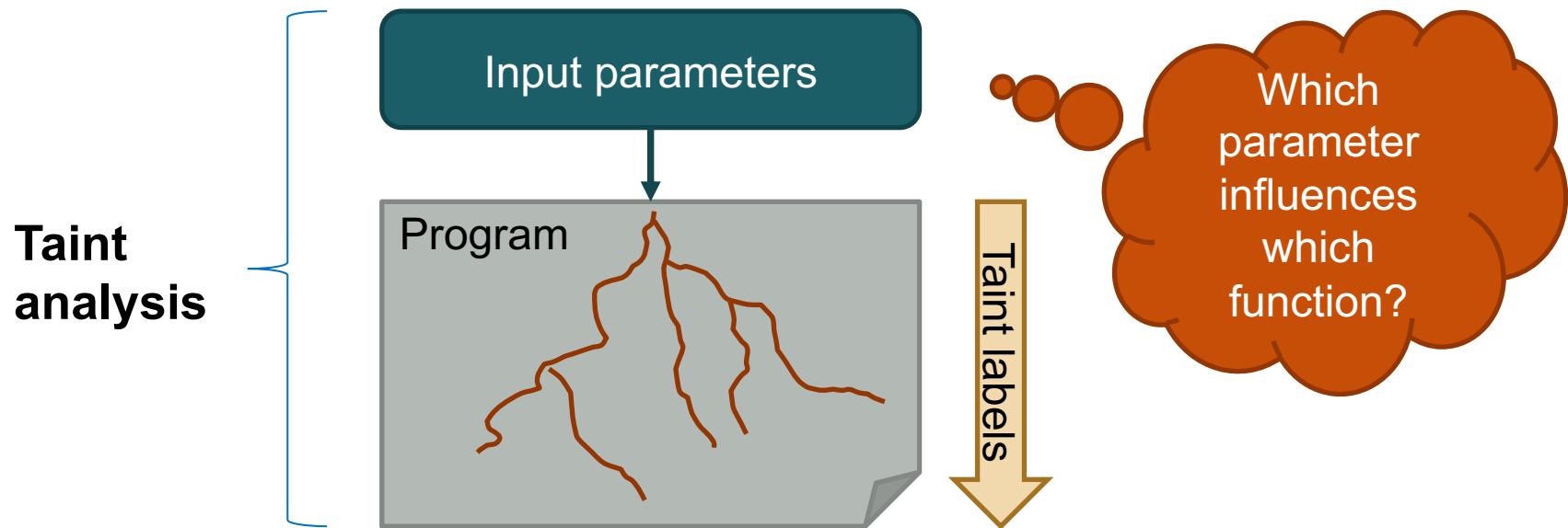
# Case studies



Application	#Parameters	Extra points	Cost savings [%]	Prediction error [%]
FASTEEST	2	0	70	2
Kripke	3	3	99	39
Relearn	2	0	85	11

# Parameter selection

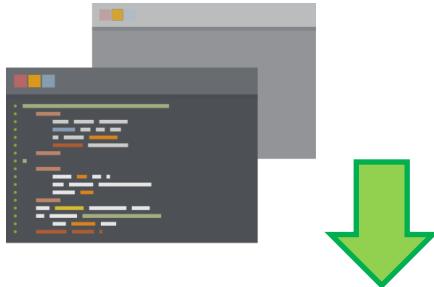
- The more parameters the more experiments
- Modeling parameters without performance impact is harmful



# PerfTaint – Taint-based performance modeling



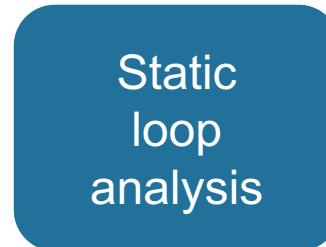
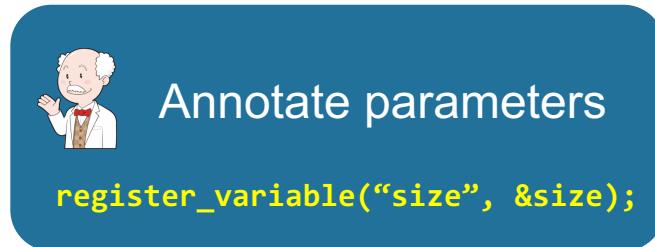
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



LLVM

Bleeding Edge Compiler Technology

DataFlowSanitizer  
+ control-flow taint propagation



- Parameter effects & dependencies
- Constant functions

[Copik et al., submitted,  
Code: spcl/perf-taint@ GitHub]

# PerfTaint - White-box performance modeling

	<b>Black box (before)</b>	<b>White box (now)</b>
Parameter identification	Manual	Taint coverage
Experiment design	Vary all parameters blindly	Exploit knowledge of parameter influence and dependencies
Instrumentation	All functions	Only functions with parameter influence
Model generation	All functions	Only functions with parameter influence

# Case study – LULESH & MILC

## Influence of program parameters



LULESH	Total	p	size	regions	iters	balance	cost		p, size
Functions	349	2	40	15	1	1		2	40
Loops	275	2	78	29	1	1		2	78
MILC	Total	p	size	trajecs	warms steps	nrest. niter	mass, beta nfl.	u0	p, size
Functions	621	54	53	12	9	6	1	4	56
Loops	874	187	161	39	31	15	1	7	196

# PerfTaint – Taint-based performance modeling

Overhead

- 50% less overhead  
(rel. to Score-P default filter)

Quality

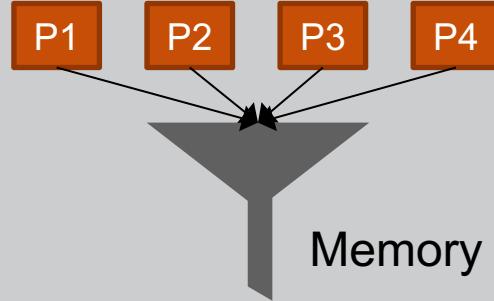
- Constant functions
- Perturbation



$$2.4 \times 10^{-8} p^{0.25} s^3$$

Validity

Hardware contention

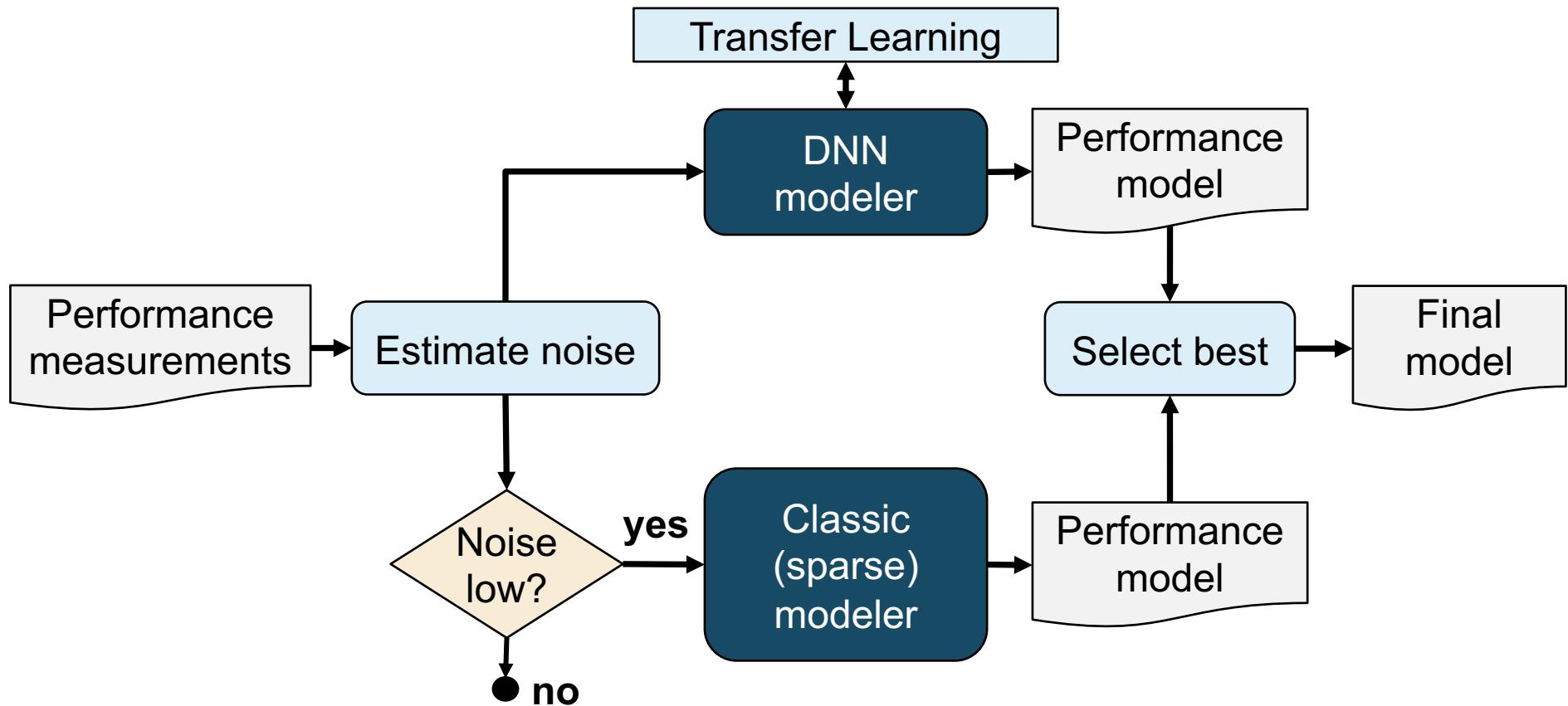


Segmented behavior

```
int foo(int a) {  
    if (a < 4)  
        kernel_linear(a);  
    else  
        kernel_log(a);  
}
```

# Noise-resilient adaptive modeling

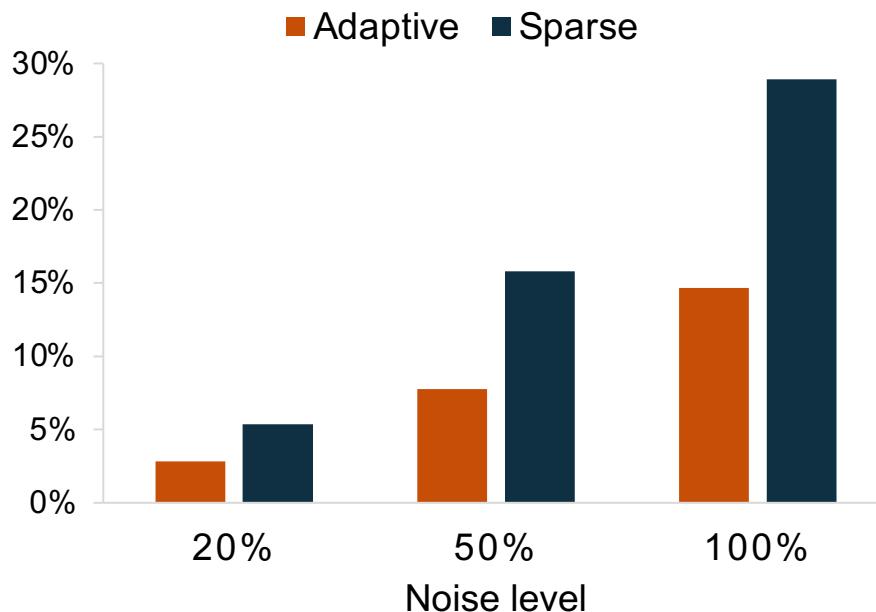
DNNs often better at guessing models in the presence of noise



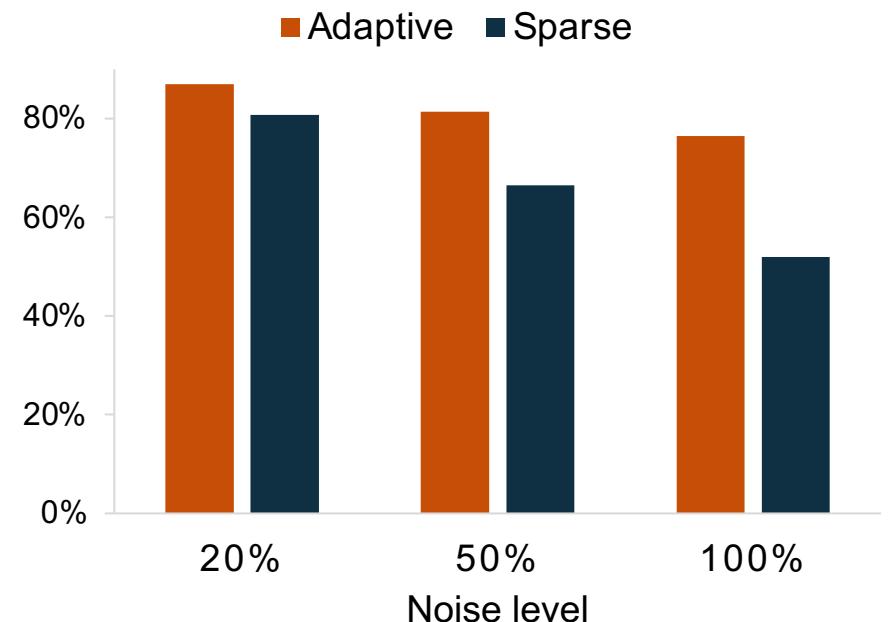
# Noise-resilient adaptive modeling

## Synthetic evaluation

**Relative error**  
(at unseen point, two ticks in each dimension)



**Lead exponents within  
1/3 of ground truth**

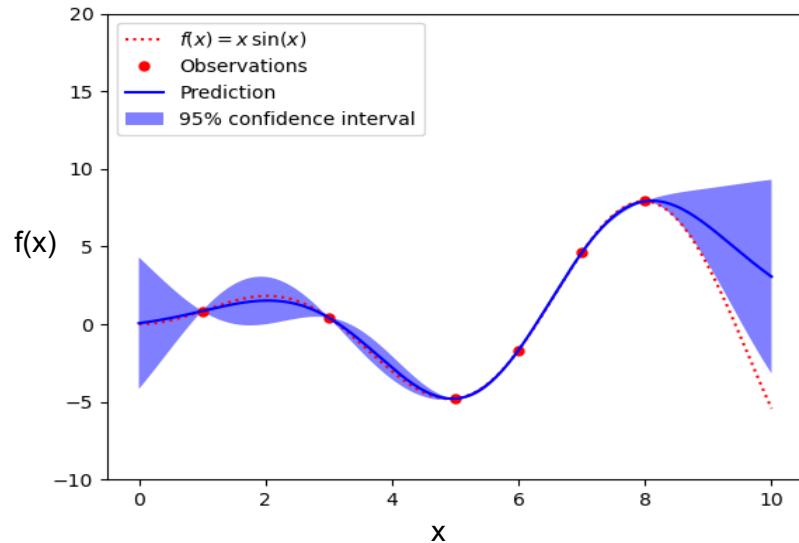
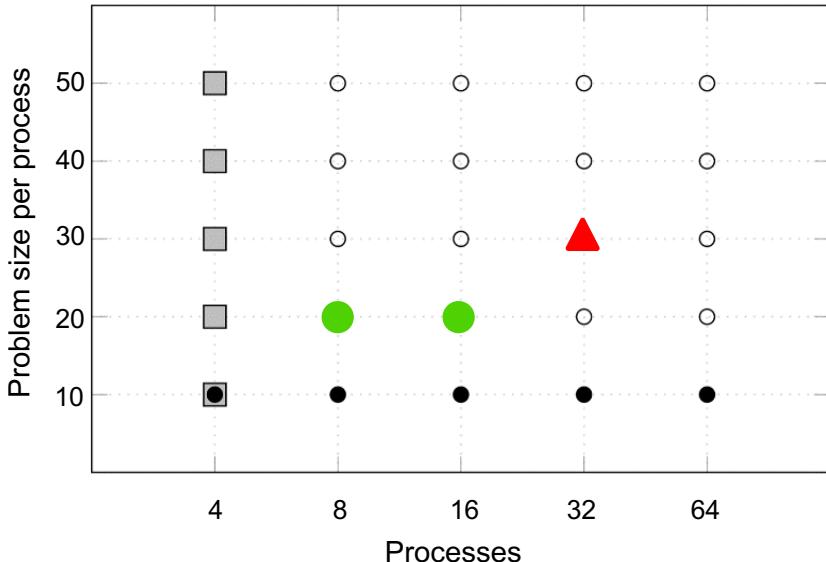


2 parameters

# Gaussian processes



**Goal:** better tradeoff between accuracy and cost for specific models



Source: [https://scikit-learn.org/stable/\\_images/sphx\\_glr\\_plot\\_gpr\\_noisy\\_targets\\_001.png](https://scikit-learn.org/stable/_images/sphx_glr_plot_gpr_noisy_targets_001.png)  
Buitinck et al.: API design for machine learning software: experiences from the scikit-learn project, 2013.

# New version of Extra-P in Q4 2020

- Includes the new sparse modeler
- Available as a Python package
- No interfaces or external dependencies
- Support for Windows and Linux (Ubuntu)
- Easy installation via pip
- BSD 3-Clause License



# Selected papers

Topic	Bibliography
Foundation (single model parameter)	Alexandru Calotoiu, Torsten Hoefer, Marius Poke, Felix Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes. <b>SC13</b> .
MPI case study	Sergei Shudler, Yannick Berens, Alexandru Calotoiu, Torsten Hoefer, Alexandre Strube, Felix Wolf: Engineering Algorithms for Scalability through Continuous Validation of Performance Expectations. <b>IEEE TPDS</b> , 30(8):1768–1785, 2019.
Multiple model parameters	Alexandru Calotoiu, David Beckingsale, Christopher W. Earl, Torsten Hoefer, Ian Karlin, Martin Schulz, Felix Wolf: Fast Multi-Parameter Performance Modeling. <b>IEEE Cluster 2016</b> .
Co-design	Alexandru Calotoiu, Alexander Graf, Torsten Hoefer, Daniel Lorenz, Sebastian Rinke, Felix Wolf: Lightweight Requirements Engineering for Exascale Co-design. <b>IEEE Cluster 2018</b> .
Task-graph modeling	Sergei Shudler, Alexandru Calotoiu, Torsten Hoefer, Felix Wolf: Isoefficiency in Practice: Configuring and Understanding the Performance of Task-based Applications. <b>PPoPP 2017</b> .
Learning cost-effective sampling strategies	Marcus Ritter, Alexandru Calotoiu, Sebastian Rinke, Thorsten Reimann, Torsten Hoefer, Felix Wolf: Learning Cost-Effective Sampling Strategies for Empirical Performance Modeling. <b>IPDPS 2020</b> .
Taint-based performance modeling	Marcin Copik, Alexandru Calotoiu, Tobias Grosser, Nicolas Wicki, Felix Wolf, Torsten Hoefer: Extracting Clean Performance Models from Tainted Programs. <b>Submitted</b> .

---

# Thank you!

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

---

# Q&A