

Noise-Resilient Performance Modeling of HPC Applications



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Felix Wolf, Technical University of Darmstadt
Scalable Tools Workshop 2022



Acknowledgement



TU Darmstadt

- Alexander Geiß
- Gustavo de Moraes
- Marcus Ritter
- Johannes Wehrstein
- Felix Wolf
- [...]



ETH Zurich

- Alexandru Calotoiu
- Marcin Copik
- Torsten Hoefler
- [...]

FZ Jülich

- Nour Daoud
- Bernd Mohr
- [...]

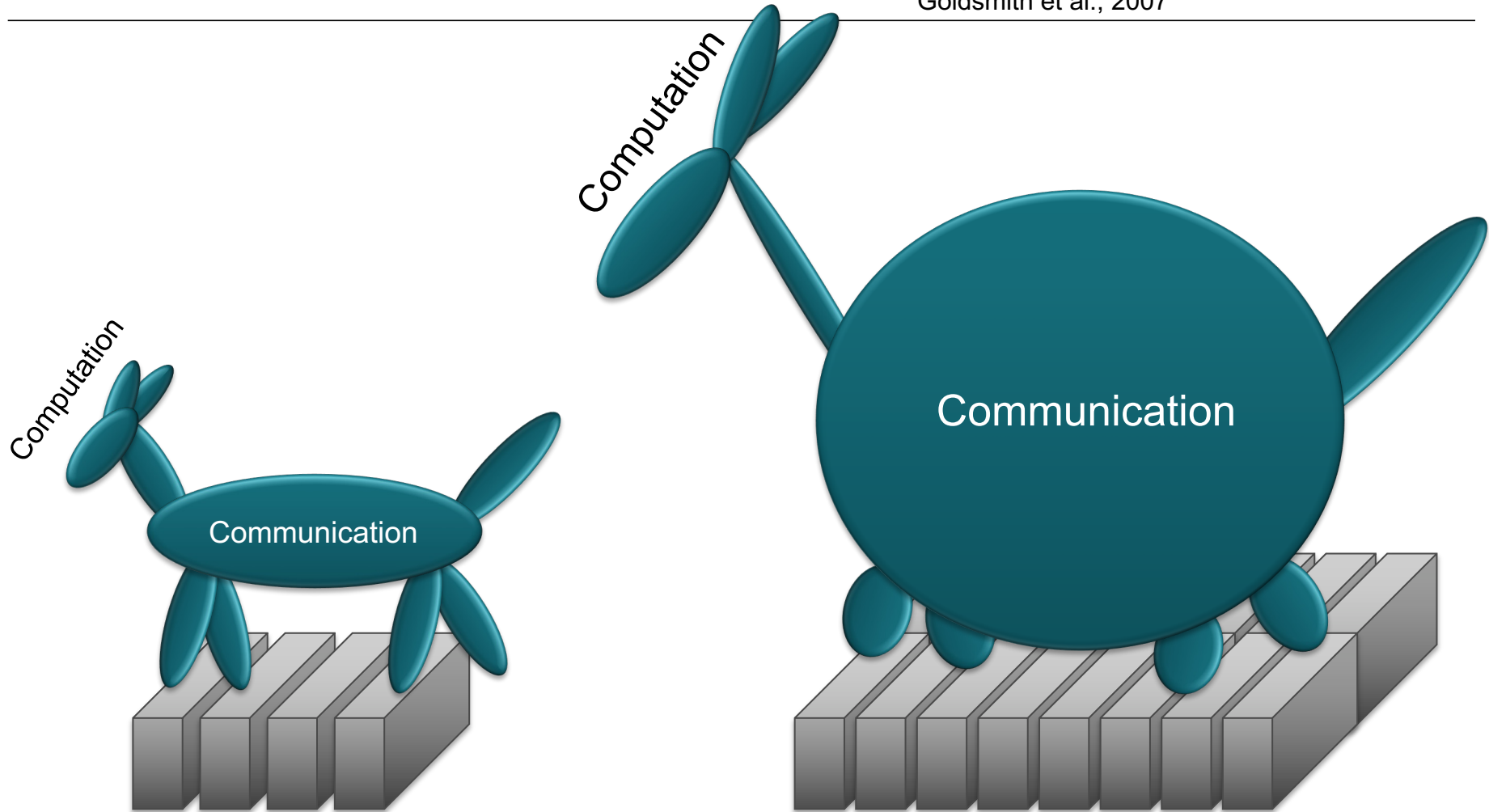
KIT

- Larissa Schmidt
- [...]



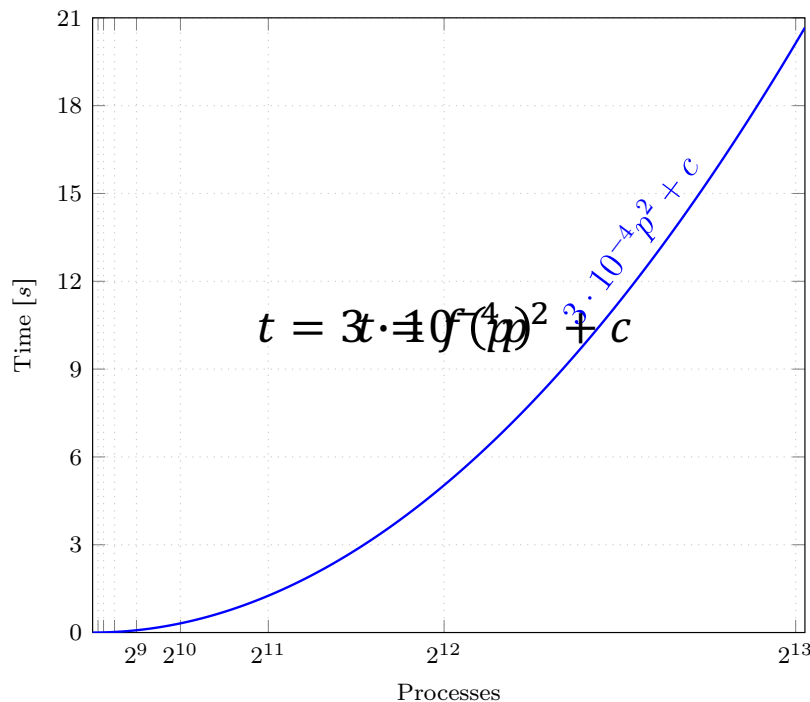
Scaling your code can harbor *performance surprises* ...

*Goldsmith et al., 2007

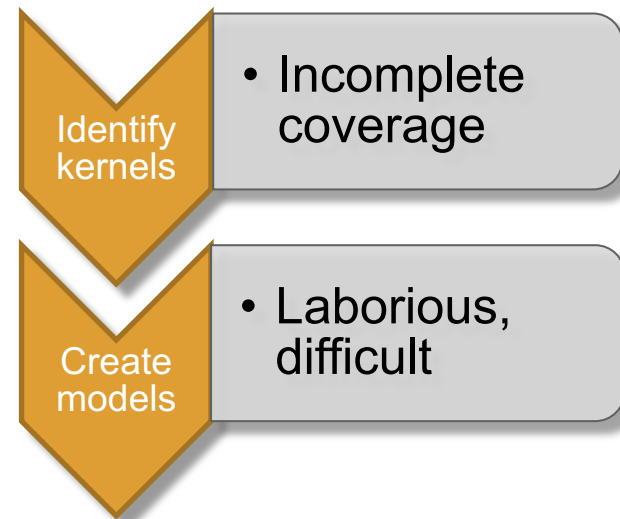


Performance model

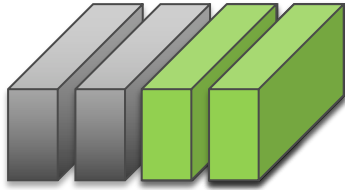
Formula that expresses a relevant performance metric as a function of one or more execution parameters



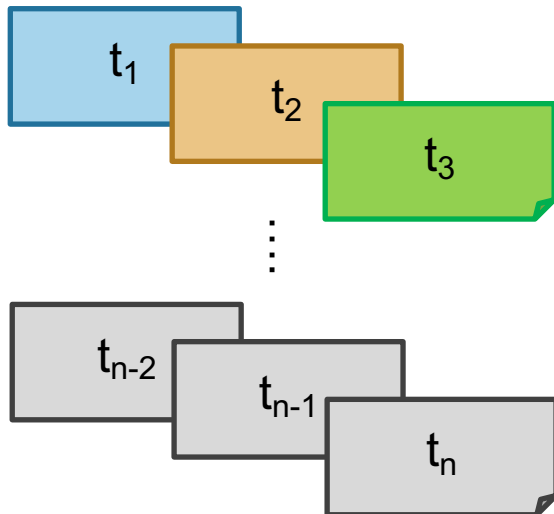
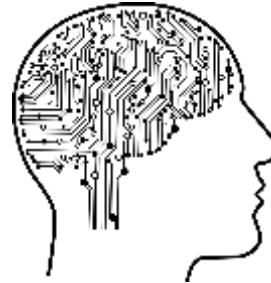
Analytical (i.e., manual) creation
challenging for entire programs



Empirical performance modeling



Performance measurements
with different execution
parameters $\mathbf{x}_1, \dots, \mathbf{x}_n$



Machine
learning

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

Alternative metrics:
FLOPs, data volume...

Challenges

Applications



Run-to-run variation / noise



System



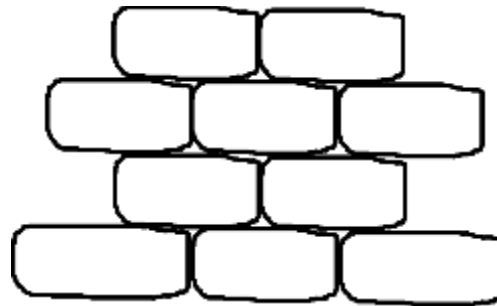
Cost of the required experiments

How to deal with noisy data


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



Time



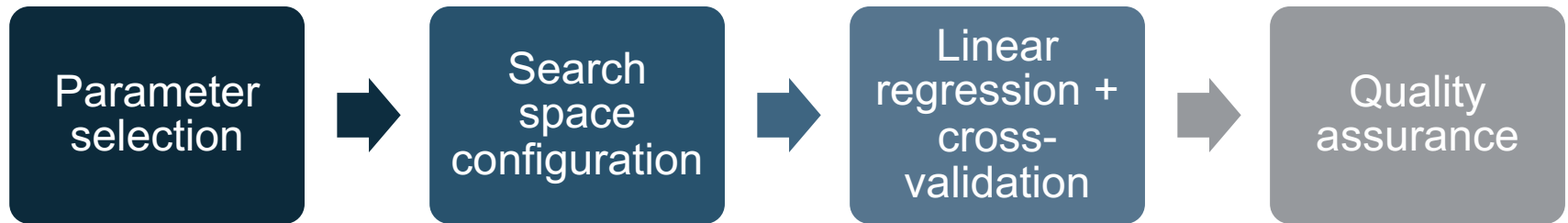
Effort

- 
- Computation
 - Memory access
 - Communication
 - I/O

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]

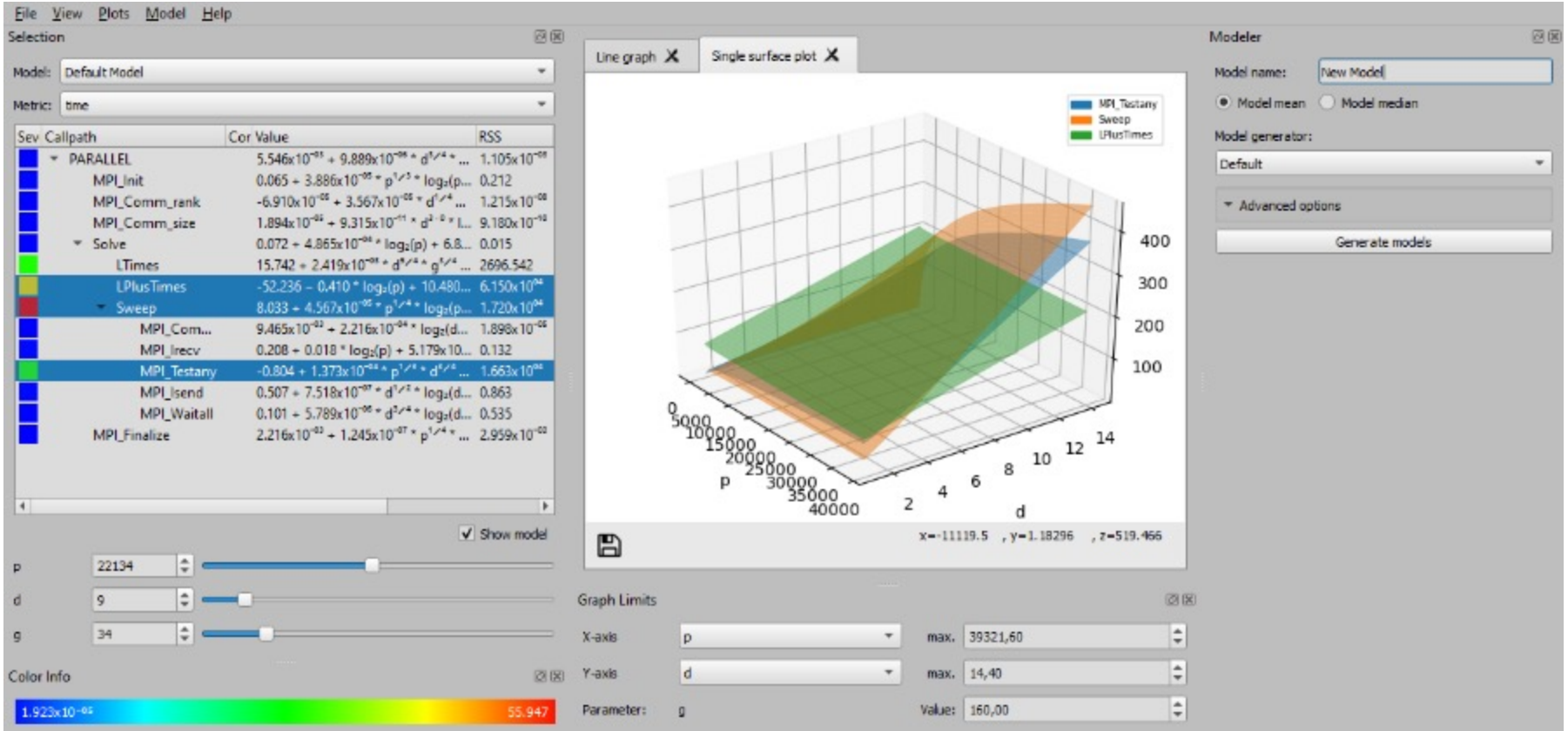


$$f(x_1, \dots, x_m) = \sum_{k=1}^n c_k \prod_{l=1}^m x_l^{i_{kl}} \cdot \log_2^{j_{kl}}(x_l)$$

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

Heuristics to
reduce
search space

Extra-P 4.0



The screenshot displays the Extra-P 4.0 software interface. On the left, a table lists various metrics and their values. The 'Solve' section is expanded, showing 'LPlusTimes' and 'Sweep' as the primary metrics. The 'Sweep' section is further expanded, showing 'MPI_Testany' as the selected metric. Below the table, there are sliders for parameters p (22134), d (9), and g (34). A color bar at the bottom left indicates the range of values from 1.923x10^-04 to 55.947.

Sev	Callpath	Cor Value	RSS
▾	PARALLEL	$5.546 \times 10^{-01} + 9.889 \times 10^{-06} \cdot d^{1/4} \dots$	1.105×10^{-00}
▾	MPI_Init	$0.065 + 3.886 \times 10^{-05} \cdot p^{1/2} \cdot \log_2(p) \dots$	0.212
▾	MPI_Comm_rank	$-6.910 \times 10^{-05} + 3.567 \times 10^{-05} \cdot d^{1/4} \dots$	1.215×10^{-00}
▾	MPI_Comm_size	$1.894 \times 10^{-02} + 9.315 \times 10^{-11} \cdot d^2 \cdot p \cdot l \dots$	9.180×10^{-12}
▾	Solve	$0.072 + 4.865 \times 10^{-04} \cdot \log_2(p) + 6.8 \dots$	0.015
▾	LTimes	$15.742 + 2.419 \times 10^{-04} \cdot d^{1/4} \cdot g^{1/4} \dots$	2696.542
▾	LPlusTimes	$-52.236 - 0.410 \cdot \log_2(p) + 10.480 \dots$	6.150×10^{04}
▾	Sweep	$8.033 + 4.567 \times 10^{-05} \cdot p^{1/4} \cdot \log_2(p) \dots$	1.720×10^{04}
▾	MPI_Comm...	$9.465 \times 10^{-02} + 2.216 \times 10^{-04} \cdot \log_2(d) \dots$	1.898×10^{-04}
▾	MPI_irecv	$0.208 + 0.018 \cdot \log_2(p) + 5.179 \times 10 \dots$	0.132
▾	MPI_Testany	$-0.804 + 1.373 \times 10^{-04} \cdot p^{1/4} \cdot d^{1/4} \dots$	1.663×10^{04}
▾	MPI_Isend	$0.507 + 7.518 \times 10^{-07} \cdot d^{1/2} \cdot \log_2(d) \dots$	0.863
▾	MPI_Waitall	$0.101 + 5.789 \times 10^{-05} \cdot d^{1/4} \cdot \log_2(d) \dots$	0.535
▾	MPI_Finalize	$2.216 \times 10^{-02} + 1.245 \times 10^{-07} \cdot p^{1/4} \dots$	2.959×10^{-02}

The central 3D surface plot shows the relationship between parameters p, d, and g. The x-axis is p (0 to 40000), the y-axis is d (0 to 14), and the z-axis is g (0 to 400). Three surfaces are plotted: MPI_Testany (blue), Sweep (orange), and LPlusTimes (green). The plot is titled 'Single surface plot' and includes a legend. Below the plot, the coordinates are given as x=-11119.5, y=1.18296, z=519.466.

On the right, the 'Modeler' panel shows the 'Model name' as 'New Model', with options for 'Model mean' (selected) and 'Model median'. The 'Model generator' is set to 'Default'. A 'Generate models' button is visible.

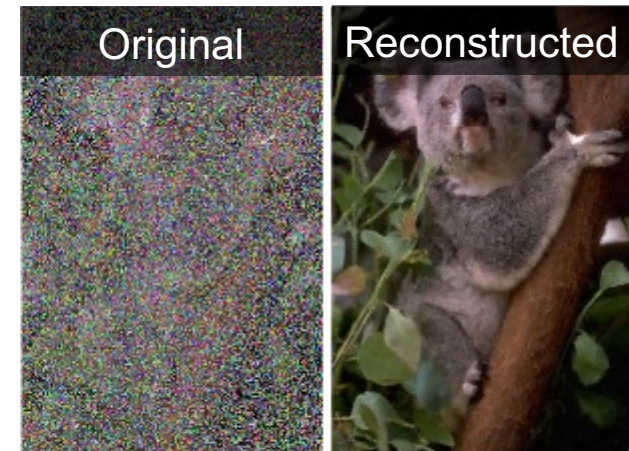
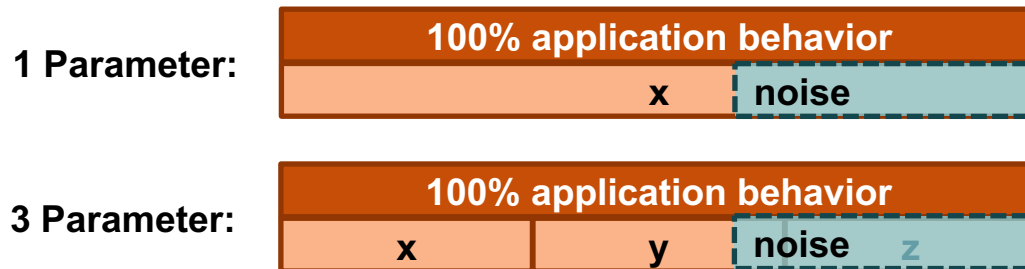
At the bottom, the 'Graph Limits' panel shows the X-axis as p (max. 39321,60), the Y-axis as d (max. 14,40), and the Parameter as g (Value: 160,00).

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

Noise-resilient performance modeling

[Ritter et al., IPDPS'21]

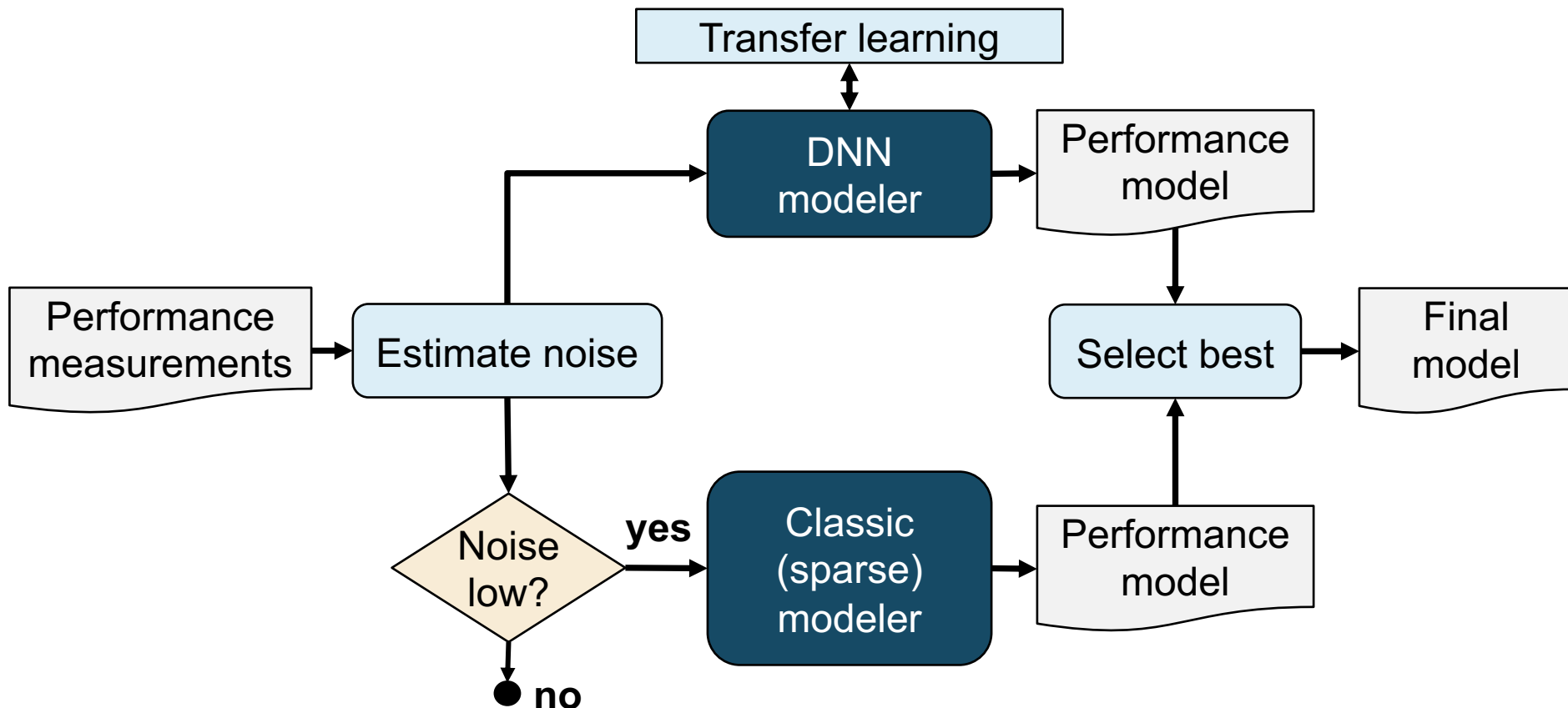
- Performance measurements frequently affected by noise
- Regression struggles with increased amounts of noise – especially w/ more parameters
- Neural networks are resilient to noise – **when noise is part of their training**



Adapted from: <https://developer.nvidia.com/blog/ai-can-now-fix-your-grainy-photos-by-only-looking-at-grainy-photos/>

Noise-resilient adaptive modeling

DNNs often better at guessing models in the presence of noise



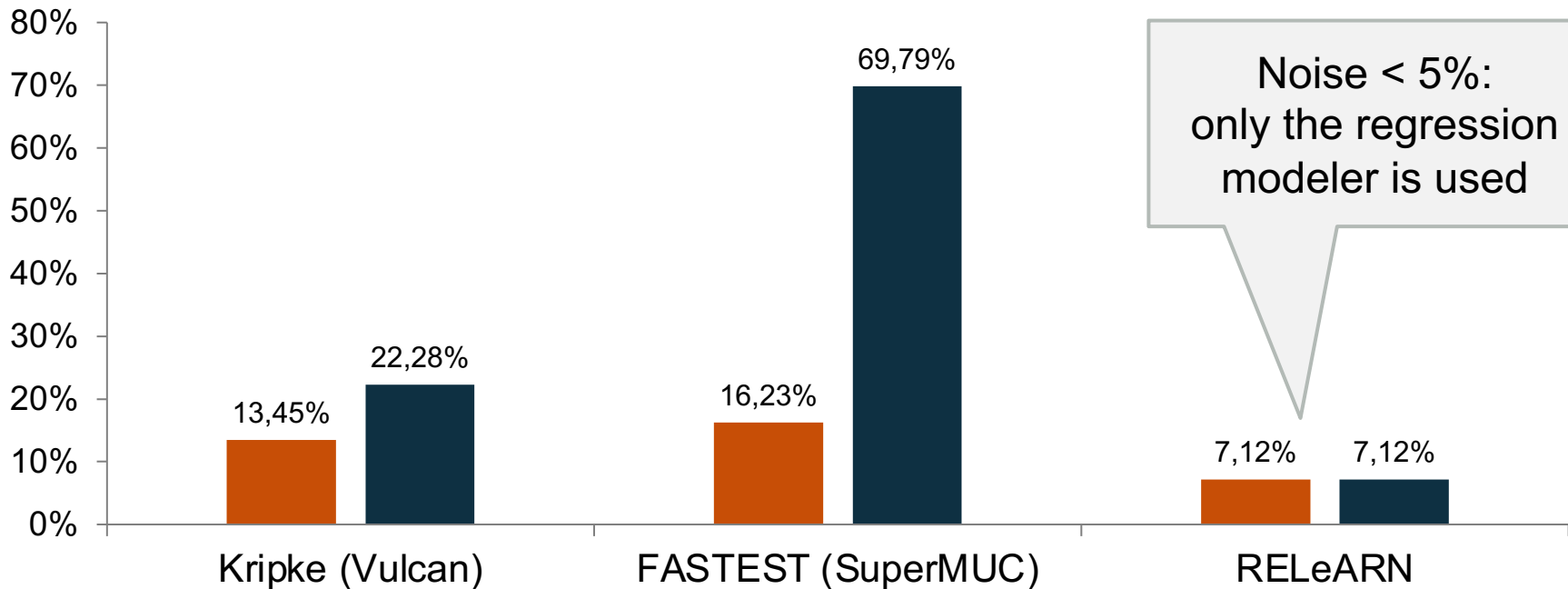
Noise-resilient performance modeling

Case studies – Results

Median relative error

(at unseen point, one tick in each dimension)

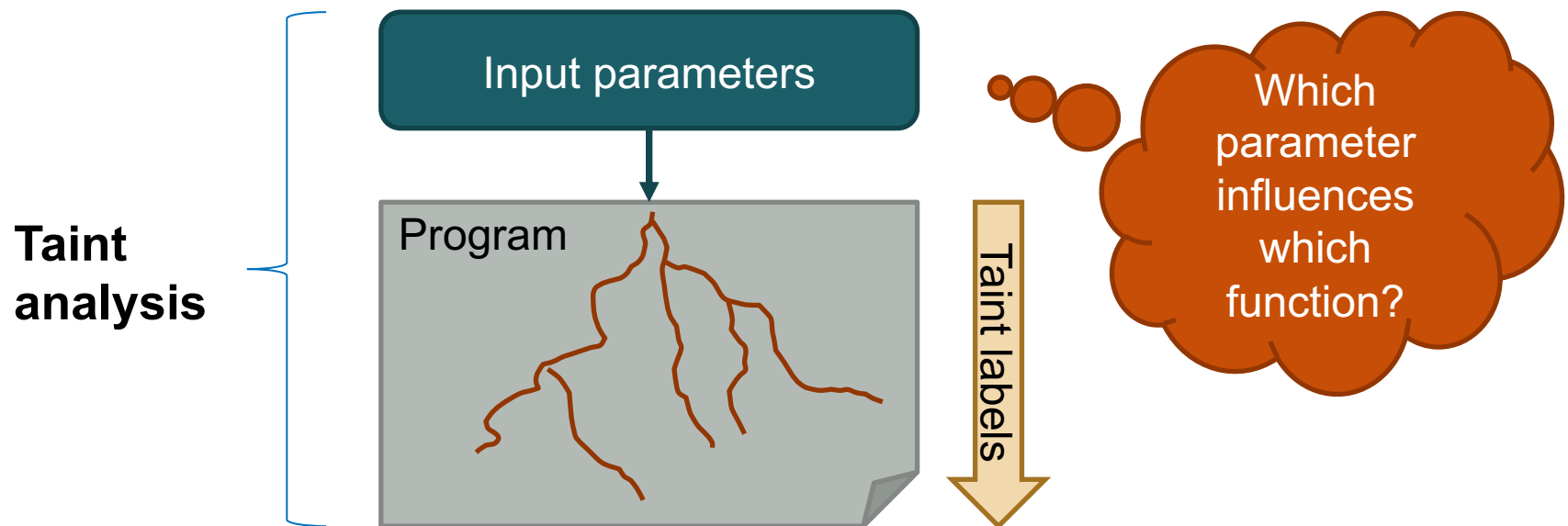
■ Adaptive ■ Regression



Parameter selection

[Copik et al, PPoPP'21]

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

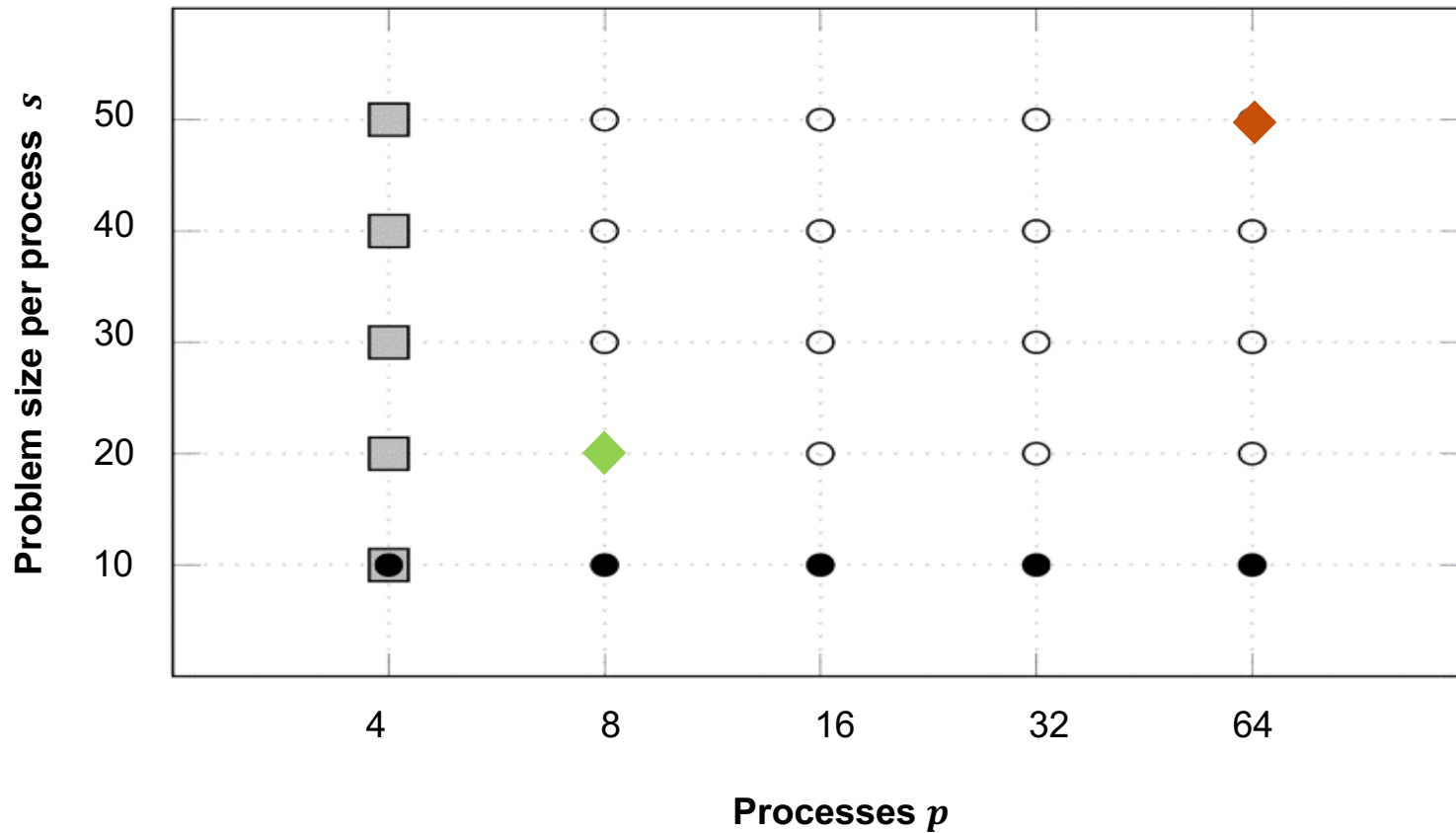


Case study – LULESH & MILC

Influence of program parameters

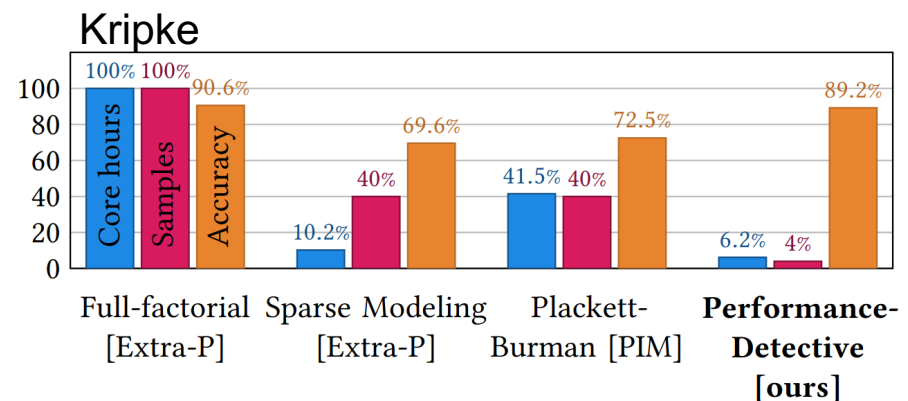
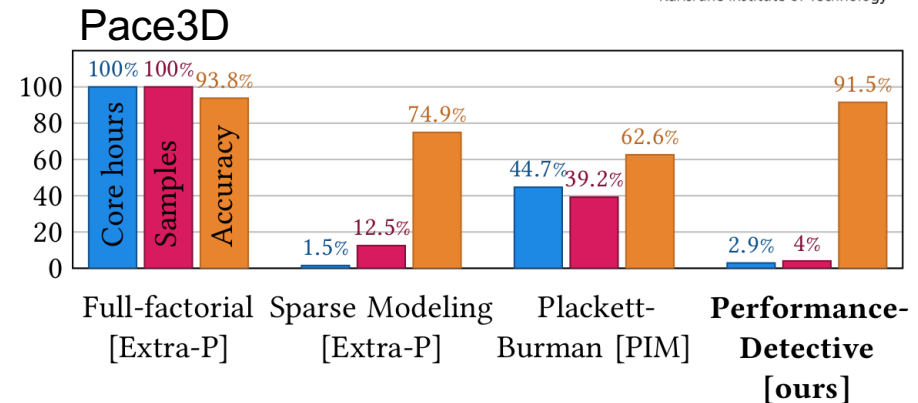
LULESH	Total	p	size	regions	iters	balance	cost		p, size
Functions	349	2	40	15	1	1	2		40
Loops	275	2	78	29	1	1	2		78
MILC	Total	p	size	trajecs	warms steps	nrest. niter	mass, beta nfl.	u0	p, size
Functions	621	54	53	12	9	6	1	4	56
Loops	874	187	161	39	31	15	1	7	196

How many data points do we really need?



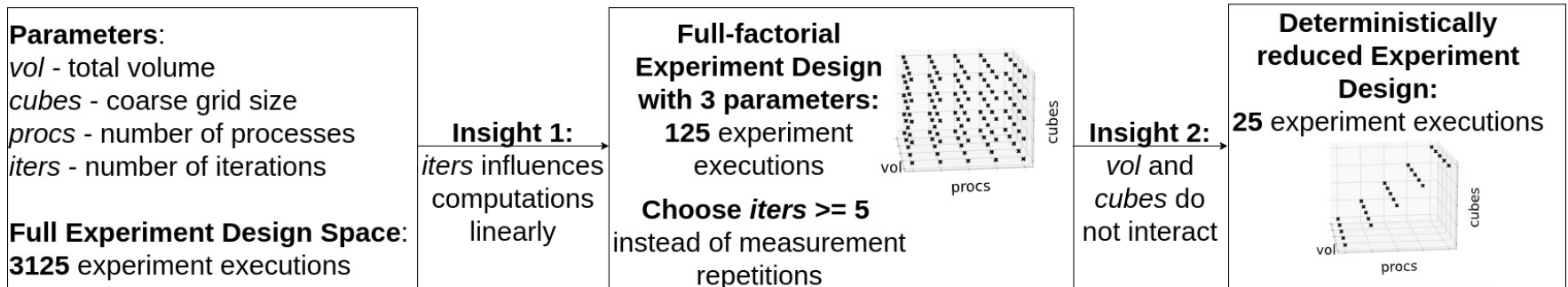
Performance Detective – Automatic deduction of cheap and accurate performance models [Schmidt et al, ICS 2022]

- **Problem** – Current heuristic sampling strategy too expensive
- **Contribution** – Use PerfTaint to deduce minimum set of experiments
- Case studies show same model accuracy at reduced cost



Performance Detective (2)

- Parameters that influence computation linearly
 - Instead of repeating measurements, set parameter influencing the computation linearly to ≥ 5
- Strike out configurations aimed at finding interactions between parameters that do not interact



Refine prior based on noise-resilient metrics

[de Morais et al., work in progress]



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Single parameter

$$f_{\text{bb}}(x) = \sum_{k=1}^{n_1} c_k \cdot p^{i_k} \cdot \log_2^{j_k}(x)$$

(basic block based model)



Search space to (i, j)

$$f(x) = \sum_{k=1}^{n_2} c_k \cdot p^{i_k} \cdot \log_2^{j_k}(x)$$

(time based model)

Multiple parameters

$$f_{\text{bb}}(x_1, \dots, x_m) = \sum_{k=1}^{n_3} c_k \prod_{l=1}^{m_3} x_l^{i_{kl}} \cdot \log_2^{j_{kl}}(x_l)$$

(basic block based model)



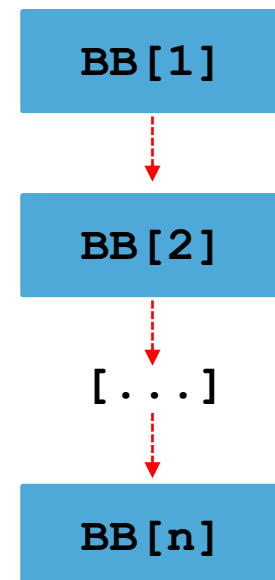
Search space to (i, j)

$$f(x_1, \dots, x_m) = \sum_{k=1}^{n_4} c_k \prod_{l=1}^{m_4} x_l^{i_{kl}} \cdot \log_2^{j_{kl}}(x_l)$$

(time based model)

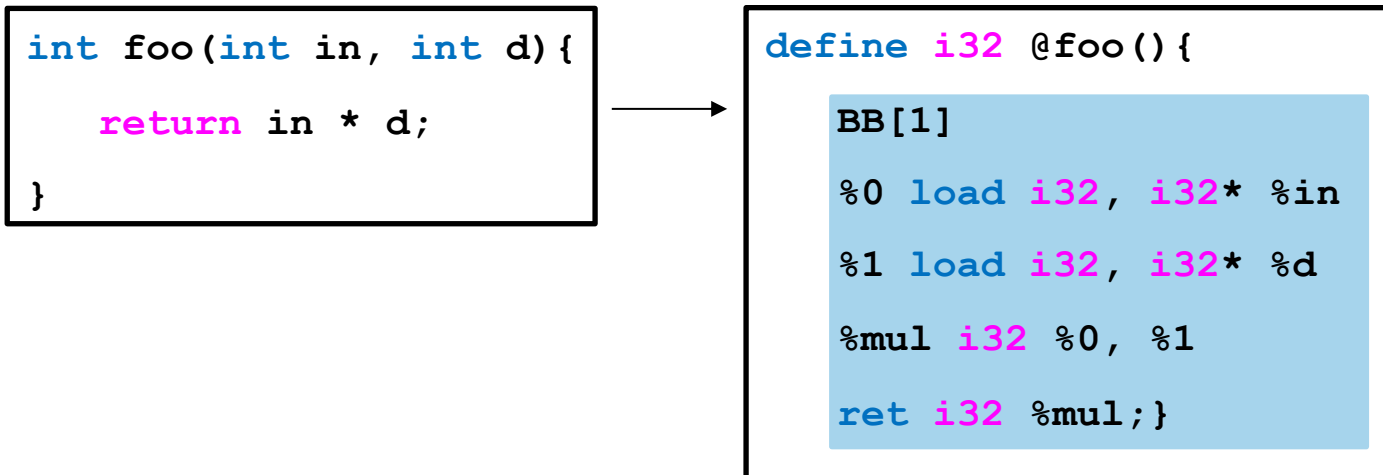
Basic block

- Code sequence with no branches (except input and output) where all the instructions are executed sequentially
- Roughly constant execution time modulo noise
 - Good unit of effort
 - Number of basic blocks executed is reproducible



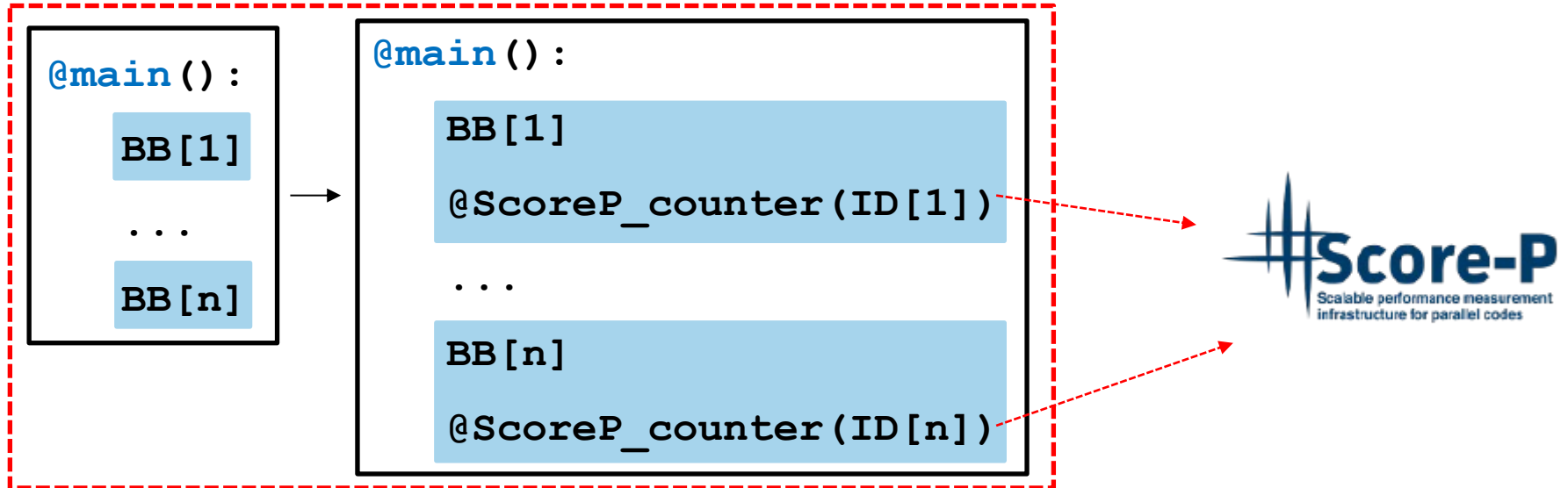
- Converts C/C++ source code to Intermediate Representation (IR)
- Allows automatic instrumentation, analyses, and optimizations of source codes on the IR level

C/C++ code to IR representation:



Basic block instrumentation w/ Score-P

IR instrumentation



- Static analysis: shows the number of functions, basic blocks and instructions present in the code
- Dynamic analysis: runs the code and counts the basic blocks as they are executed

Selected papers

Topic	Bibliography
Foundation (single model paramter)	Alexandru Calotoiu, Torsten Hoefler, Marius Poke, Felix Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes. SC13 .
Foundation (multipler model paramters)	Alexandru Calotoiu, David Beckingsale, Christopher W. Earl, Torsten Hoefler, Ian Karlin, Martin Schulz, Felix Wolf: Fast Multi-Parameter Performance Modeling. IEEE Cluster 2016 .
Noise resilience	Marcus Ritter, Alexander Geiß, Johannes Wehrstein, Alexandru Calotoiu, Thorsten Reimann, Torsten Hoefler, Felix Wolf: Noise-Resilient Empirical Performance Modeling with Deep Neural Networks. IPDPS 2021 .
Taint-based performance modeling	Marcin Copik, Alexandru Calotoiu, Tobias Grosser, Nicolas Wicki, Felix Wolf, Torsten Hoefler: Extracting Clean Performance Models from Tainted Programs. PPoPP 2021 .
Performance detective	Larissa Schmid, Marcin Copik, Alexandru Calotoiu, Dominik Werle, Andreas Reiter, Michael Selzer, Anne Koziolk, Torsten Hoefler: Performance-Detective: Automatic Deduction of Cheap and Accurate Performance Models. ICS 2022 .

Thank you!



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Q&A