Noise-Resilient Performance Modeling of HPC Applications

TECHNISCHE UNIVERSITÄT DARMSTADT

Felix Wolf, Technical University of Darmstadt

Scalable Tools Workshop 2022

Photo: Alex Becker / TU Darmstadt

Acknowledgement

TU Darmstadt

- Alexander Geiß
- Gustave de Morais
- Marcus Ritter
- Johannes Wehrstein
- Felix Wolf
- […]

ETH Zurich

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

FZ Jülich

- Nour Daoud
- Bernd Mohr
- […]

KIT

- Larissa Schmidt
- […]

Bundesministerium für Bildung und Forschung

EuroHPC Joint Undertaking

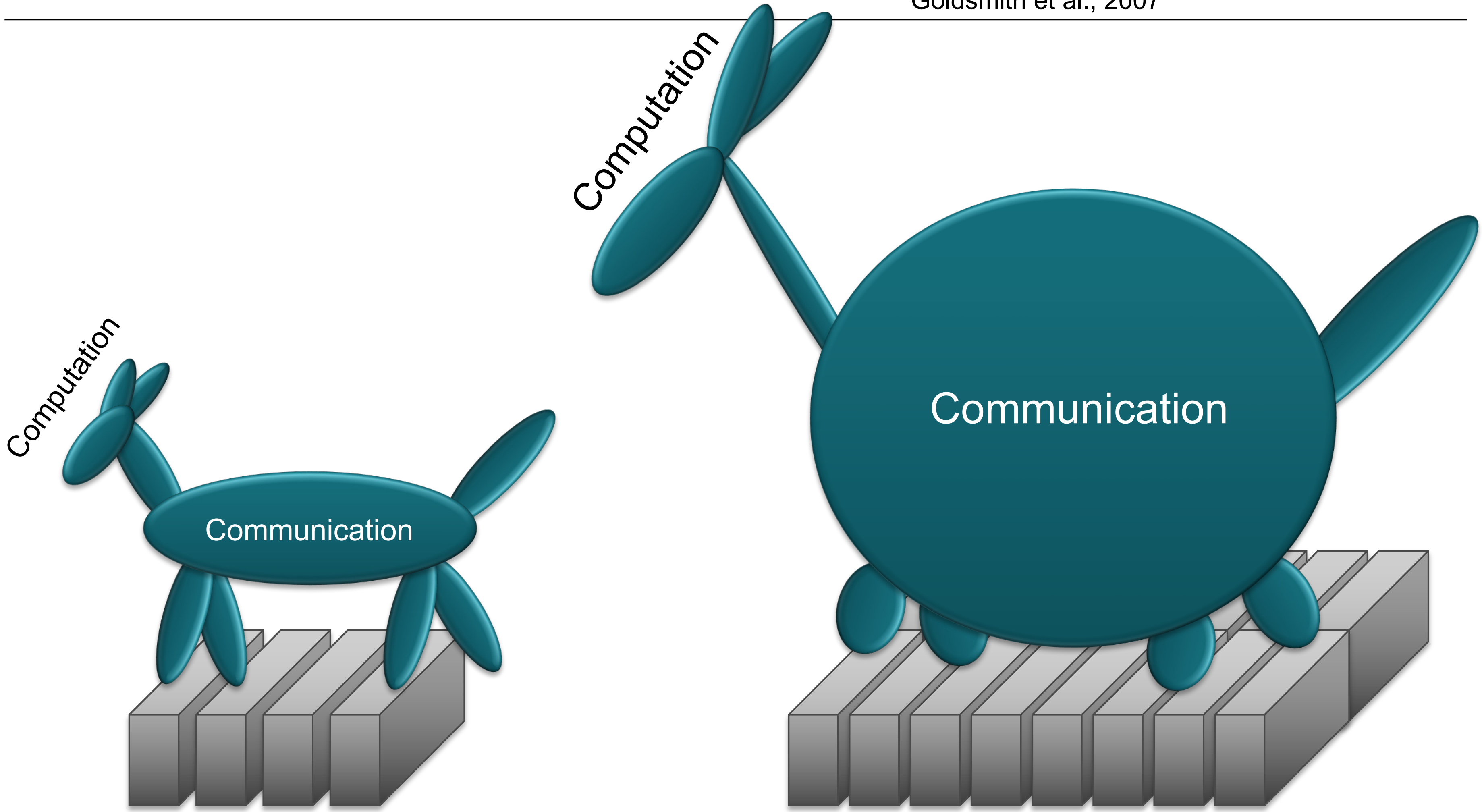
EXTRA NOISE

DFG

FNSNF

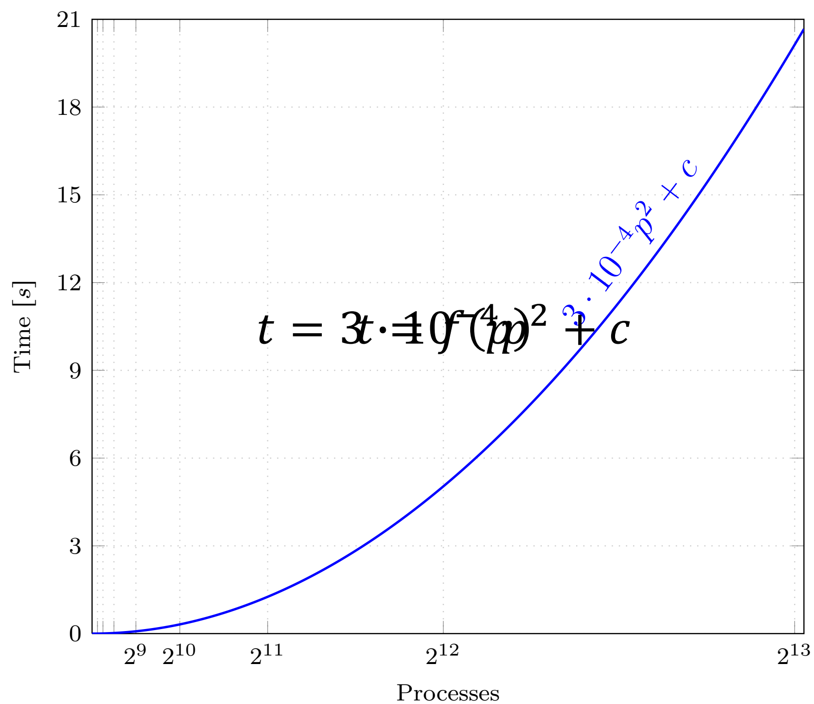
Scaling your code can harbor *performance surprises…**

*Goldsmith et al., 2007



Performance model

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

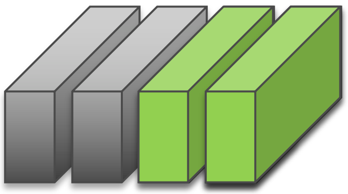


Analytical (i.e., manual) creation challenging for entire programs

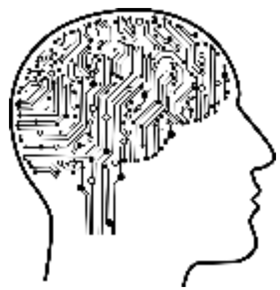
Identify kernels
- Incomplete coverage

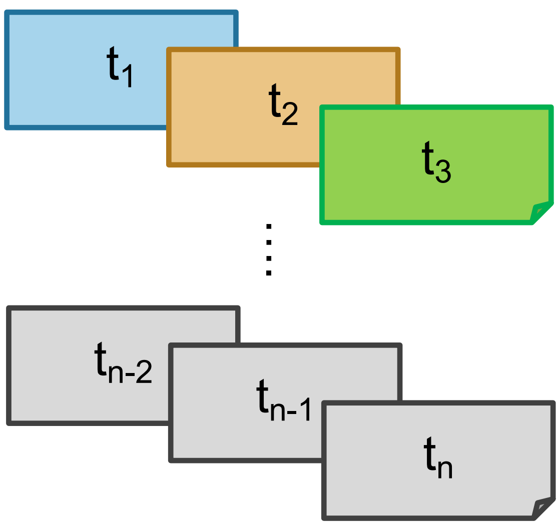
Create models
- Laborious, difficult

Empirical performance modeling



Performance measurements with different execution parameters $\mathbf{x_1,\ldots,x_n}$





Machine learning

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

Alternative metrics: FLOPs, data volume…

Challenges

Applications

Run-to-run variation / noise

System

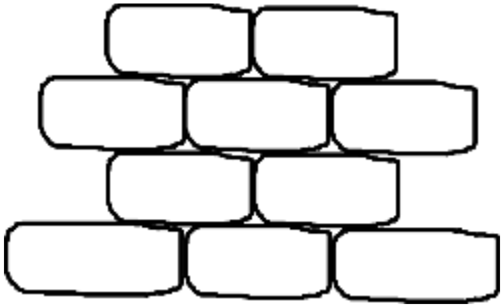
Cost of the required experiments

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]

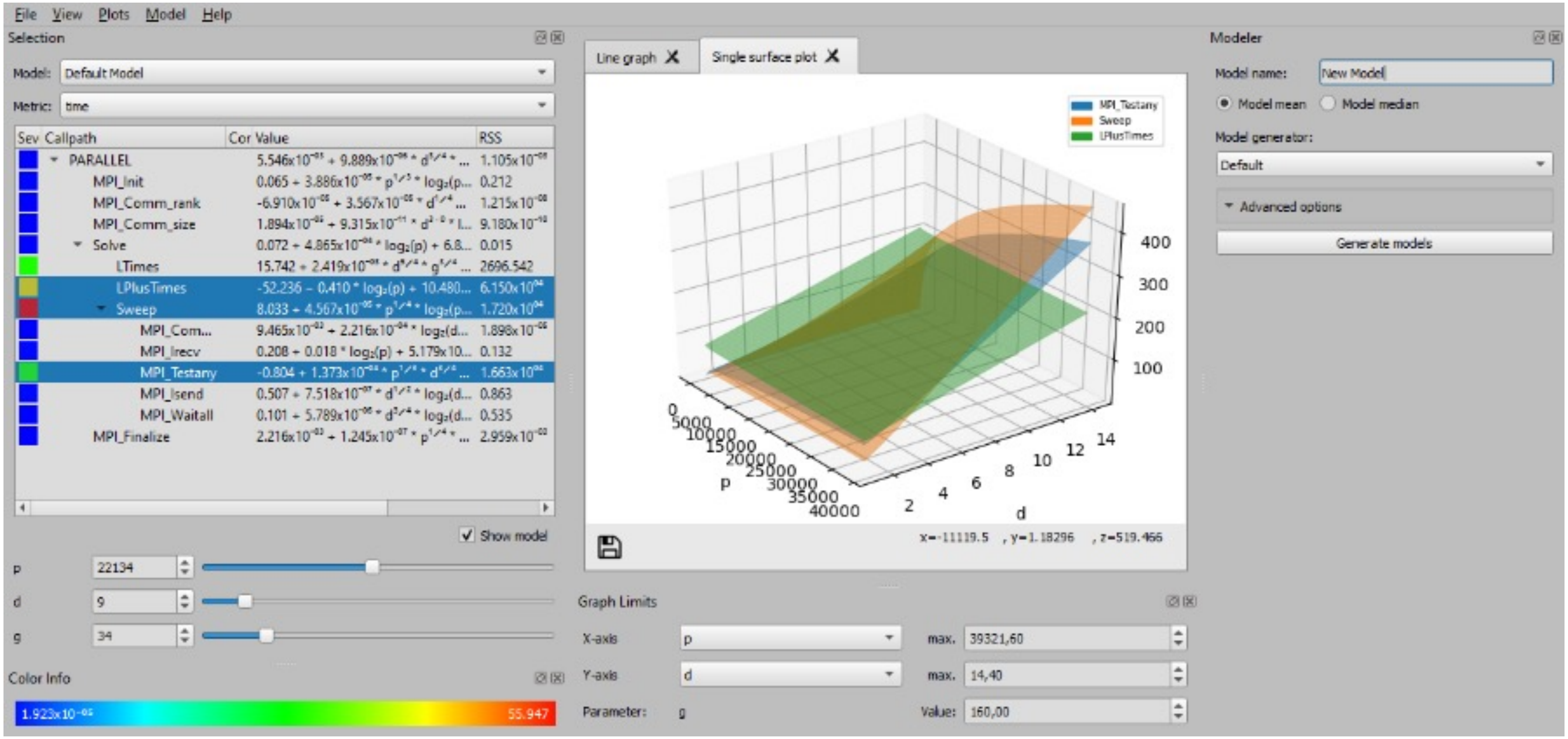
Parameter selection → Search space configuration → Linear regression + cross-validation → Quality assurance

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

Heuristics to reduce search space

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

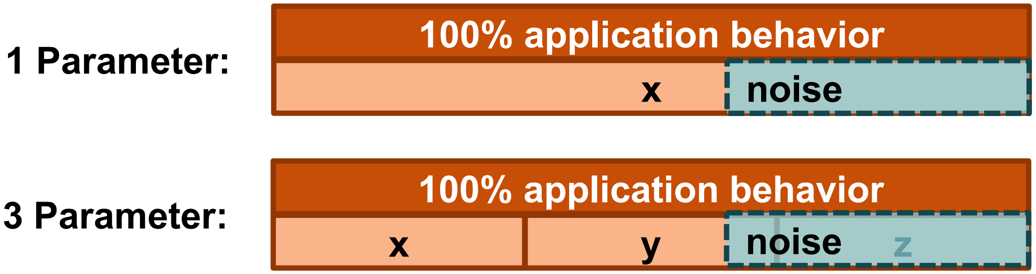
Extra-P 4.0

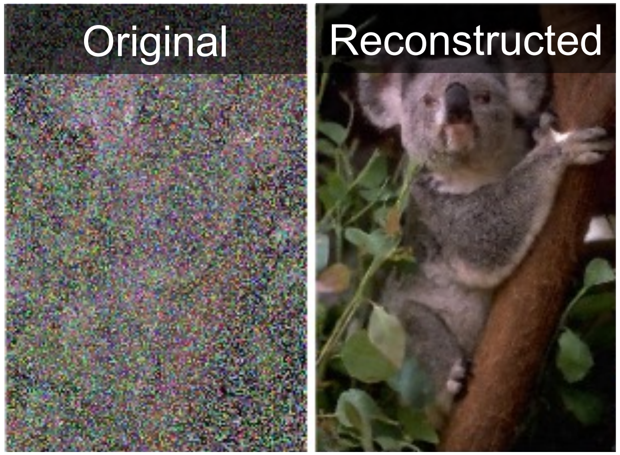


Available at: https://github.com/extra-p/extrap

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**

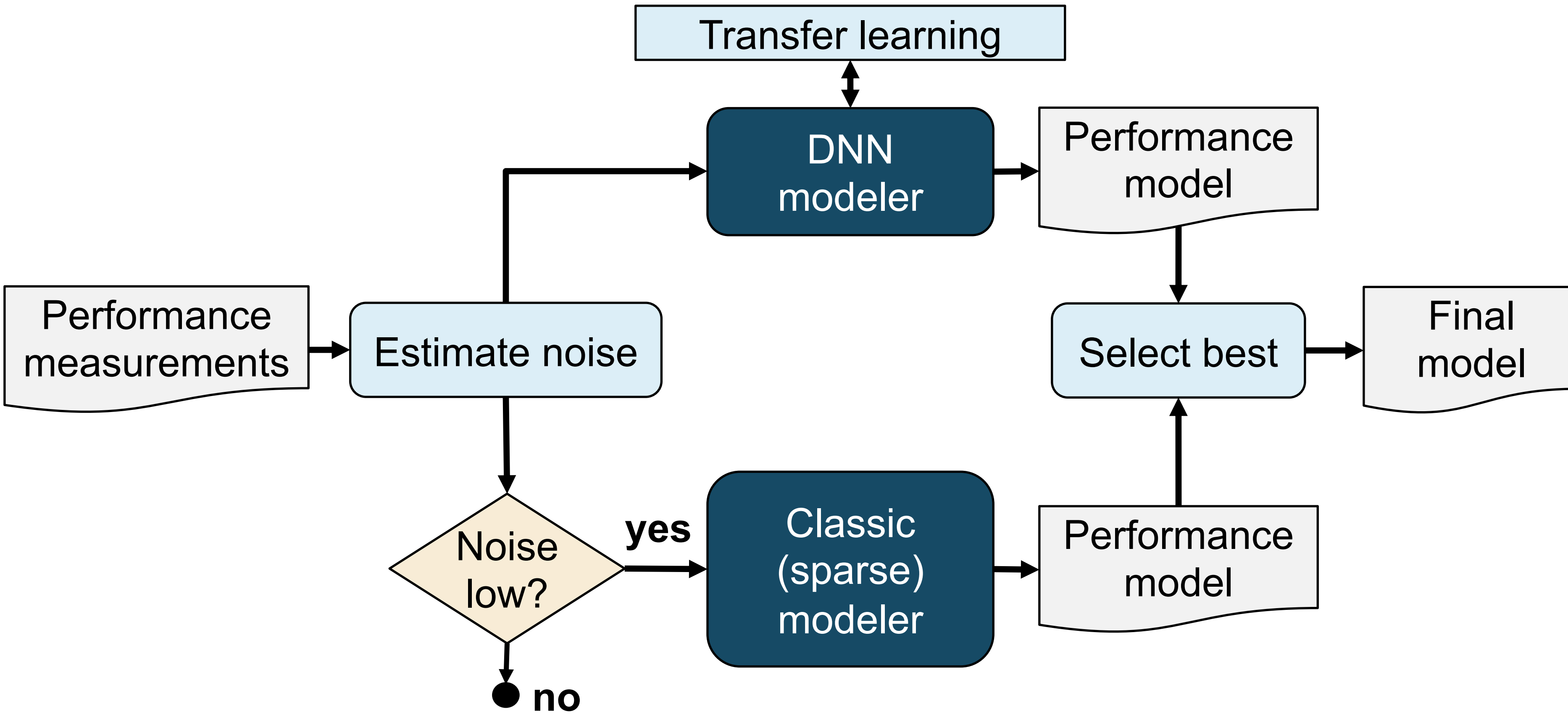




Adapted from: https://developer.nvidia.com/blog/ai-can-now-fix-your-grainy-photos-by-only-looking-at-grainy-photos/

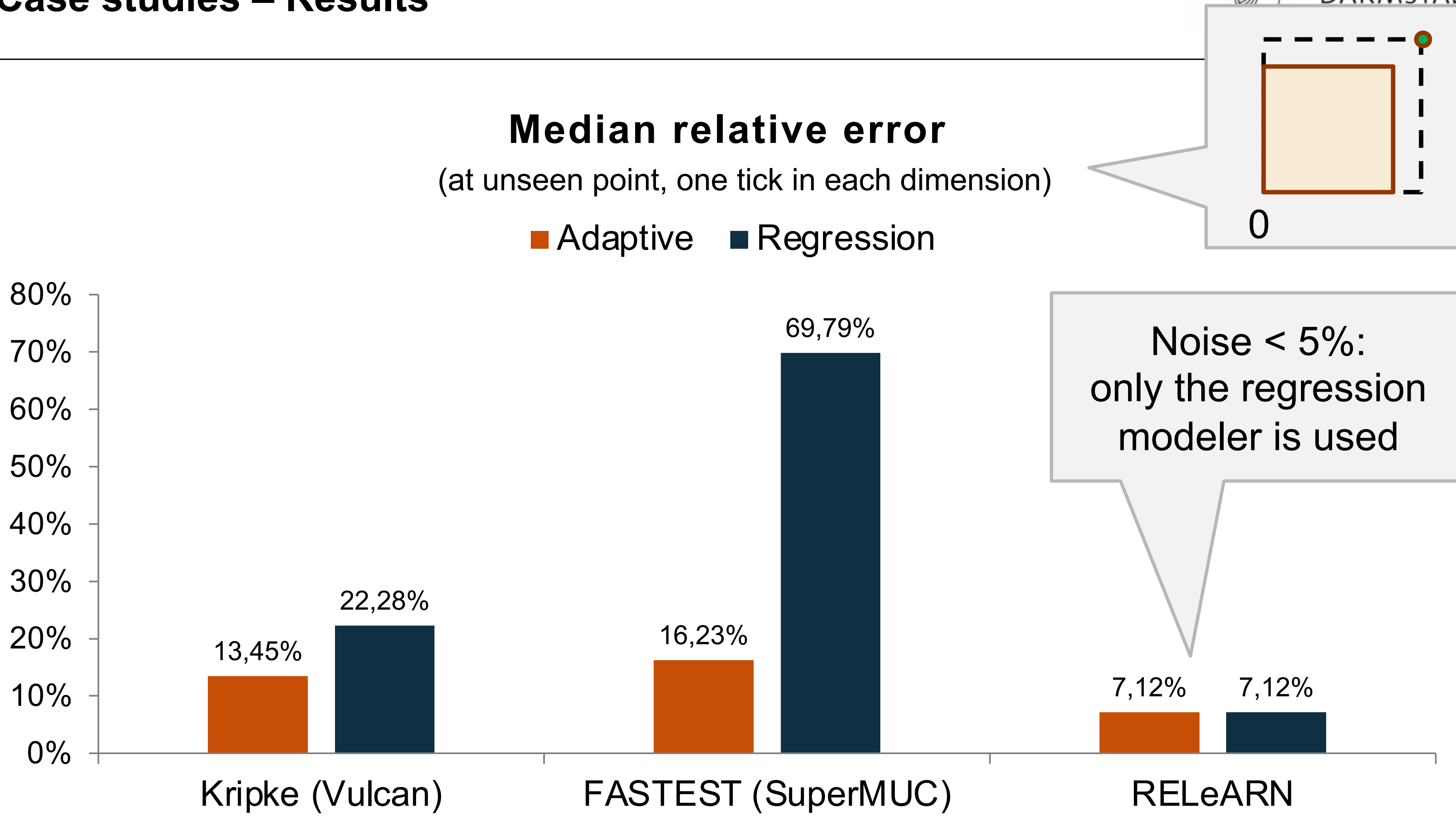
Noise-resilient adaptive modeling

DNNs often better at guessing models in the presence of noise



Noise-resilient performance modeling

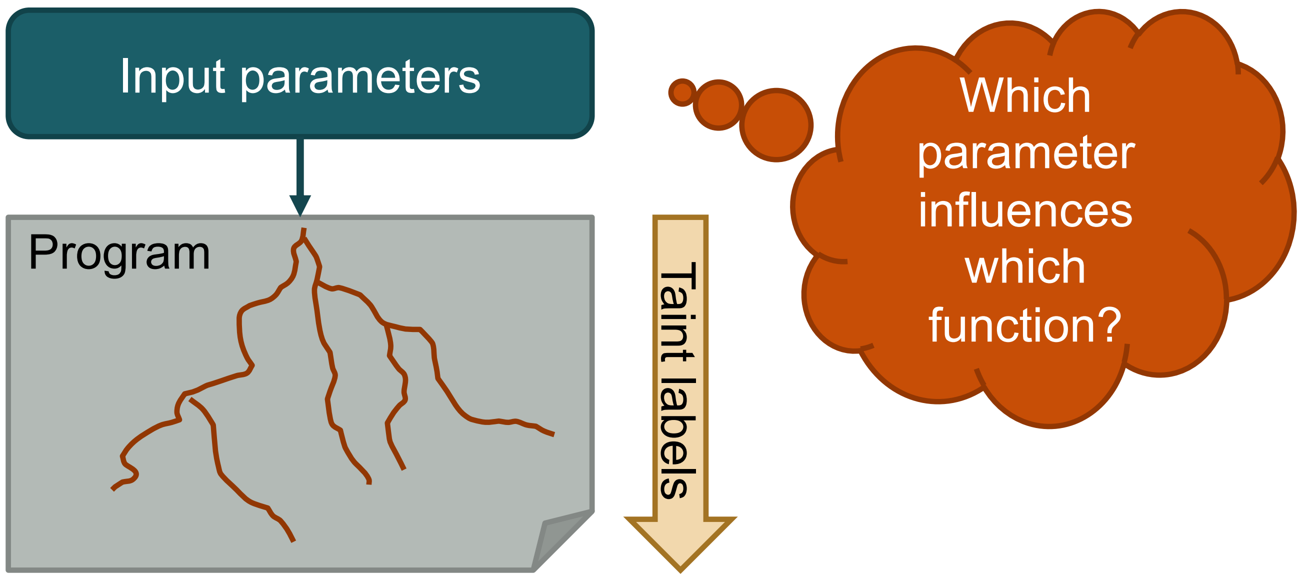
Case studies – Results



Parameter selection
[Copik et al, PPoPP'21]

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

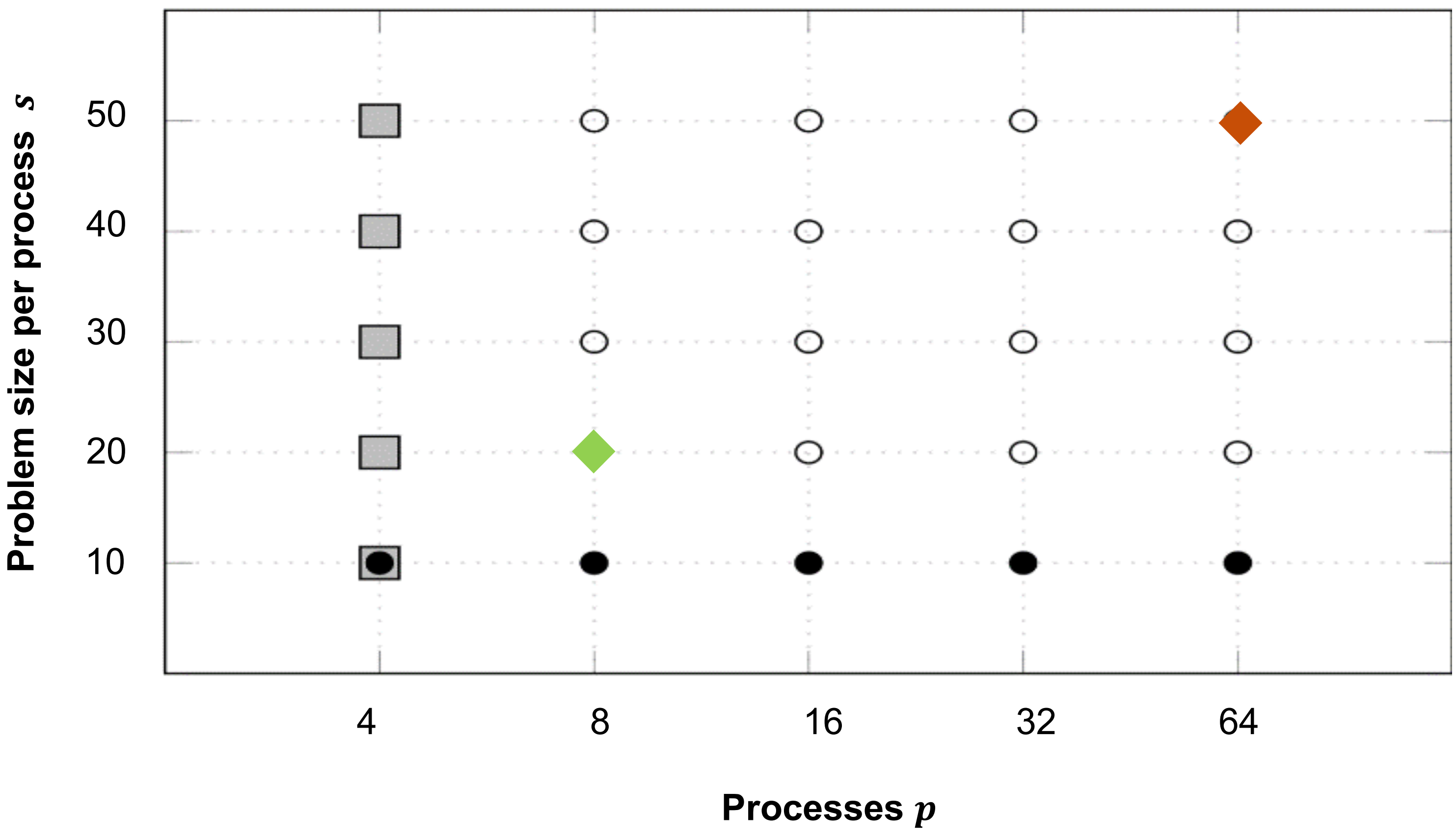
Taint analysis



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

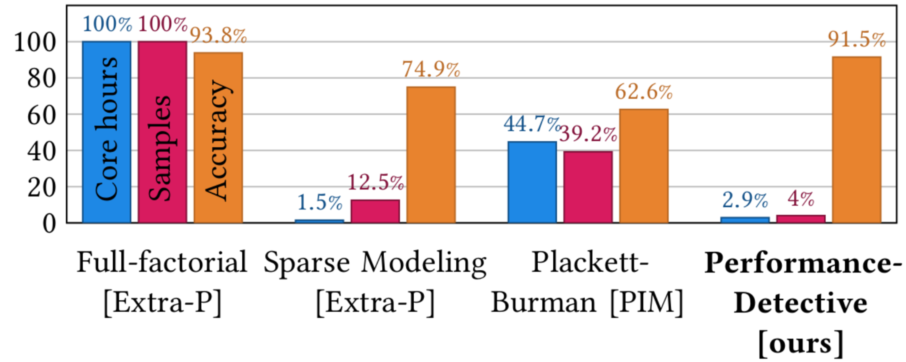
How many data points do we really need?



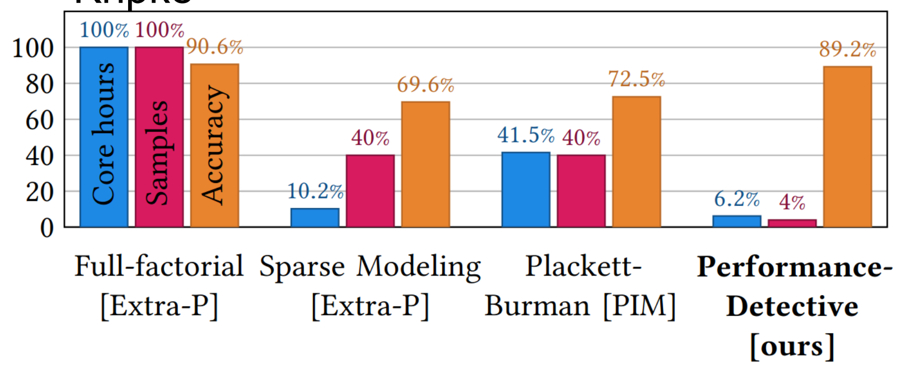
Performance Detective –
Automatic deduction of cheap and accurate performance models
[Schmidt et al, ICS 2022]

- **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

Pace3D



Kripke



Performance Detective (2)

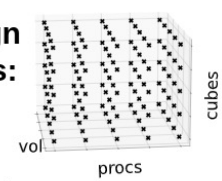
- Parameters that influence computation linearly
  - Instead of repeating measurements, set parameter influencing the computation linearly to >= 5
- Strike out configurations aimed at finding interactions between parameters that do not interact

Parameters:
vol - total volume
cubes - coarse grid size
procs - number of processes
iters - number of iterations

Full Experiment Design Space:
3125 experiment executions

→ **Insight 1:** *iters* influences computations linearly →

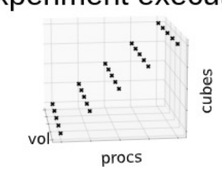
Full-factorial Experiment Design with 3 parameters:
125 experiment executions



Choose *iters* >= 5 instead of measurement repetitions

→ **Insight 2:** *vol* and *cubes* do not interact →

Deterministically reduced Experiment Design:
25 experiment executions



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)

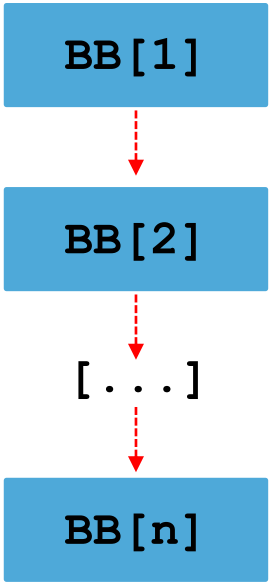
Search space to (i, j)

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

(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



LLVM

- 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:

```
int foo(int in, int d){
    return in * d;
}
```

→

```
define i32 @foo(){
    BB[1]
    %0 load i32, i32* %in
    %1 load i32, i32* %d
    %mul i32 %0, %1
    ret i32 %mul;}
```

Basic block instrumentation w/ Score-P

TECHNISCHE UNIVERSITÄT DARMSTADT

IR instrumentation

```
@main():
    BB[1]
    ...
    BB[n]
```

→

```
@main():
    BB[1]
    @ScoreP_counter(ID[1])
    ...
    BB[n]
    @ScoreP_counter(ID[n])
```

Score-P
Scalable performance measurement infrastructure for parallel codes

- 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 Koziolek, Torsten Hoefler: Performance-Detective: Automatic Deduction of Cheap and Accurate Performance Models. **ICS 2022**.

Thank you!

Q&A