Update on the Performance-Modeling Tool Extra-P

TECHNISCHE UNIVERSITÄT DARMSTADT

Felix Wolf, TU Darmstadt

Acknowledgement

- David Beckingsale
- Alexandru Calotoiu
- Christopher W. Earl
- Torsten Hoefler
- Kashif Ilyas
- Ian Karlin
- Daniel Lorenz
- Patrick Reisert
- Martin Schulz
- Sergei Shudler
- Andreas Vogel

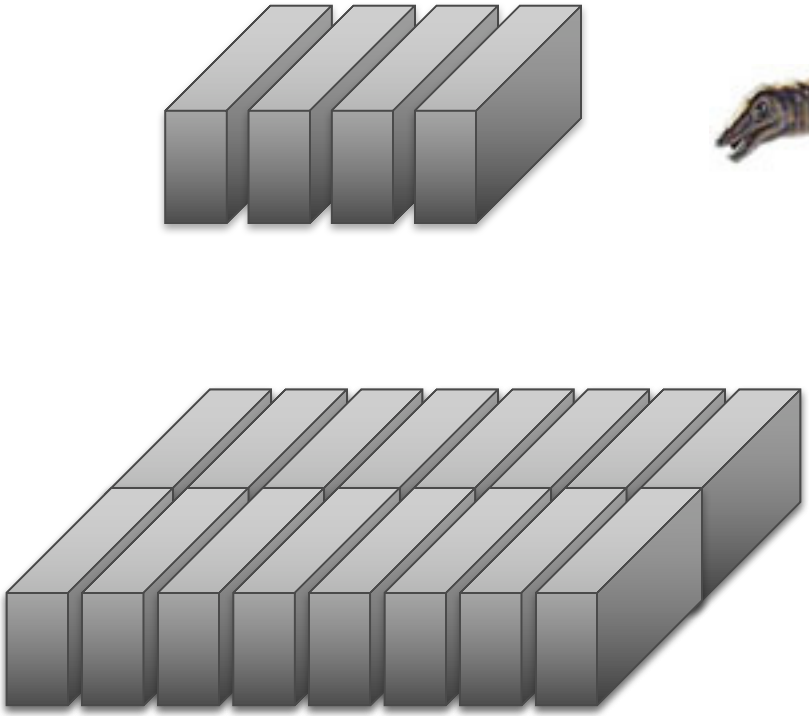
GEFÖRDERT VOM

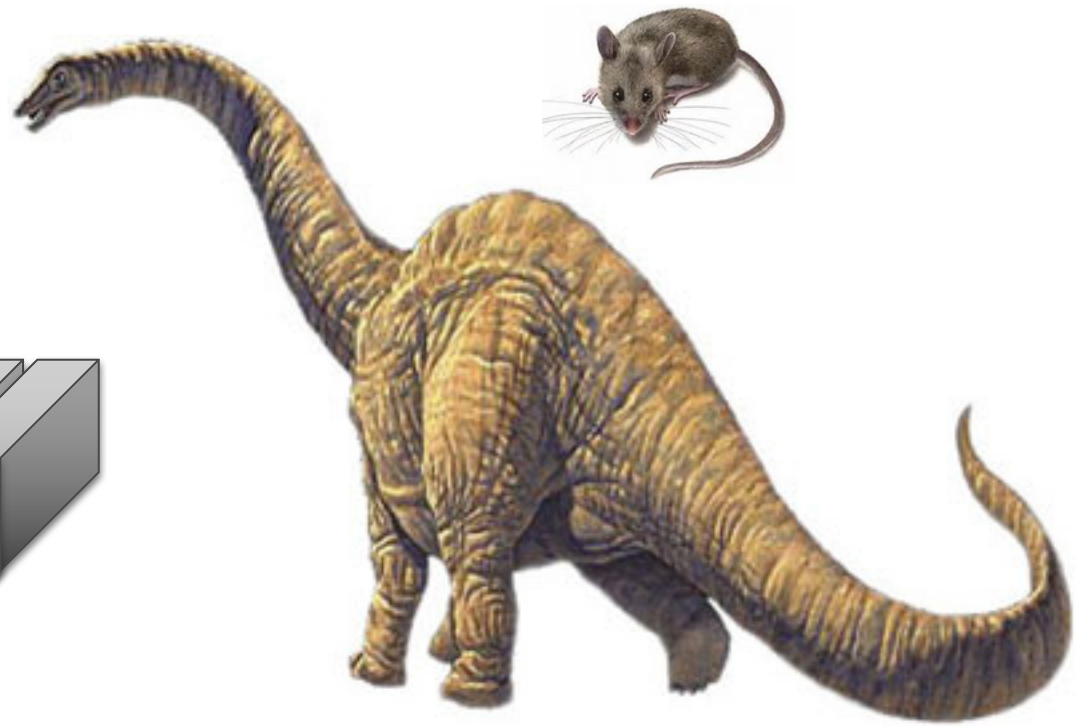
Bundesministerium für Bildung und Forschung

Latent scalability bugs

System size

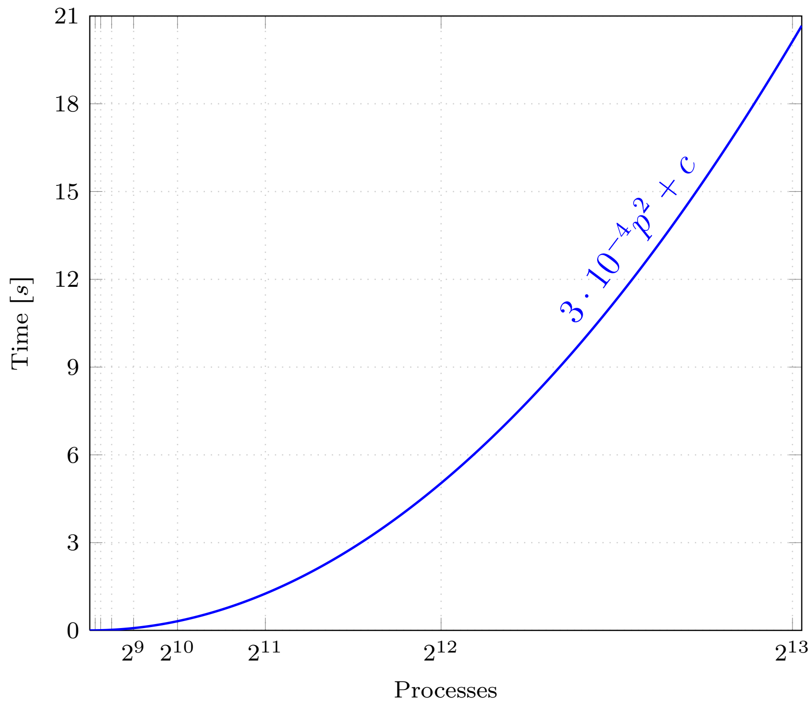
Wall time





Motivation

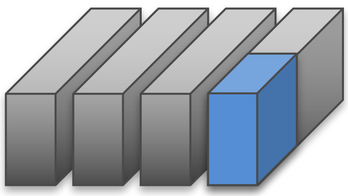
Performance model = formula that expresses relevant performance metrics as a function of one or more execution parameters



Manual creation challenging

- Identify kernels
  - Incomplete coverage
- Create models
  - Laborious, difficult

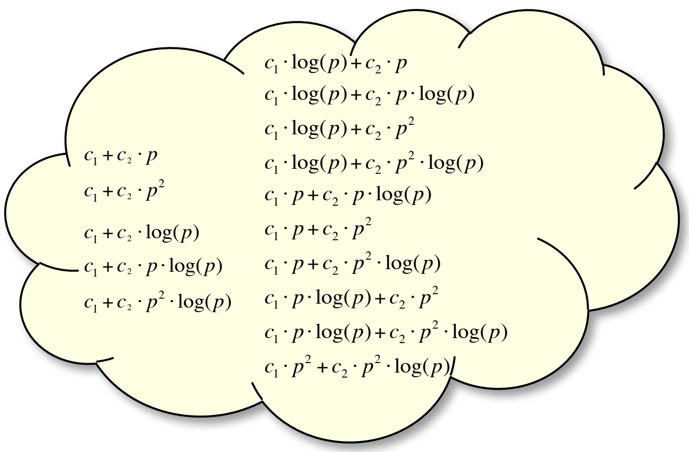
Automatic empirical performance modeling



Small-scale measurements

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

Performance model normal form (PMNF)

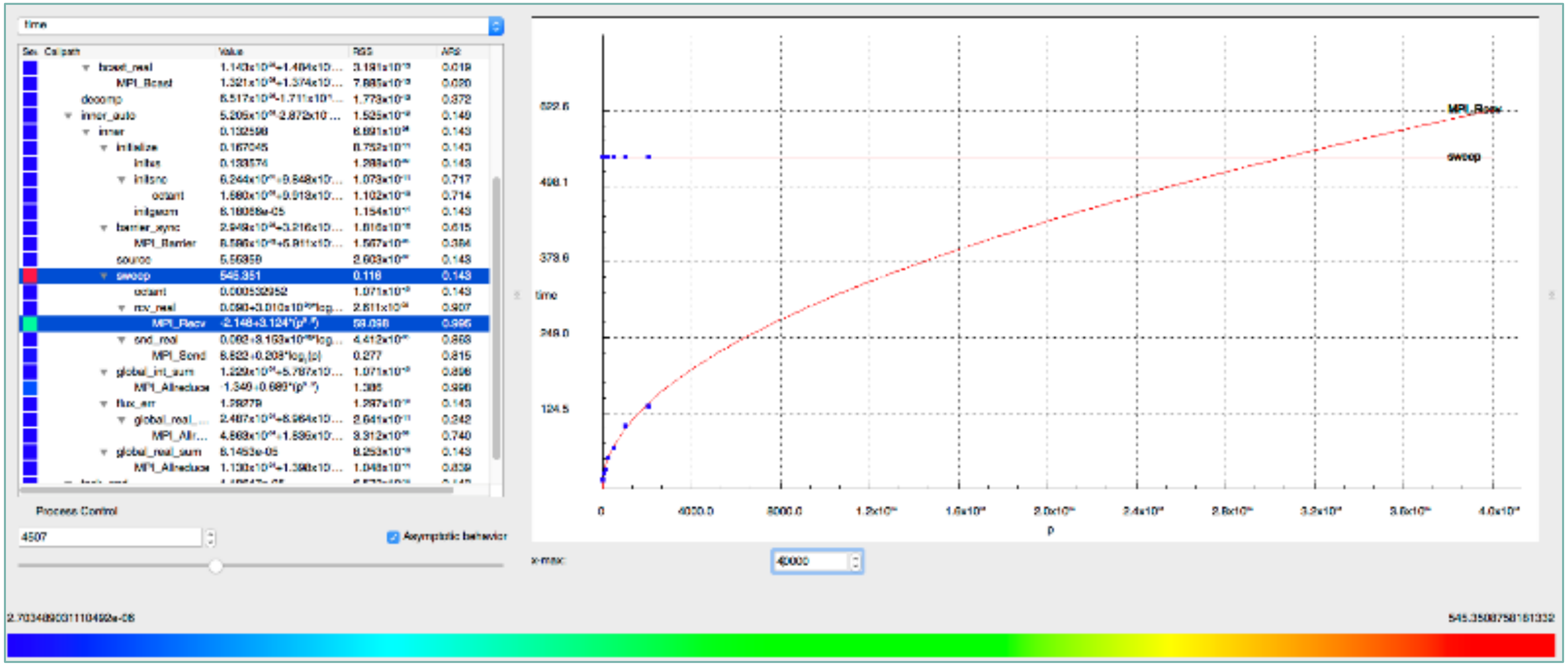


Generation of candidate models
and selection of best fit

Kernel [2 of 40]	Model [s] t = f(p)
sweep → MPI_Recv	$4.03\sqrt{p}$
sweep	582.19

Extra-P 3.0

TECHNISCHE UNIVERSITÄT DARMSTADT



- GUI improvements, better stability, additional features
- Tutorials available through VI-HPS and upon request

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

Recent developments

1. Performance models with multiple parameters
2. Automatic configuration of the search space
3. Segmented models
4. Iso-efficiency modeling
5. Lightweight requirements engineering for co-design

Models with more than one parameter

TECHNISCHE UNIVERSITÄT DARMSTADT

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

$n = 3$

$m = 3$

$I = \left\{\frac{0}{4}, \frac{1}{4}, \ldots, \frac{12}{4}\right\}$

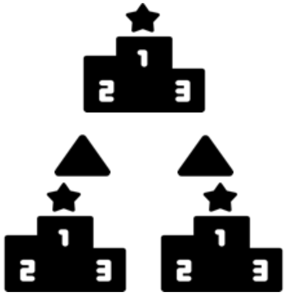
$J = \{0,1,2\}$

Search space explosion

- Total number of hypotheses to search: 34.786,300,841,019
- Too slow for any practical purpose

Search space reduction through heuristics

- **Hierarchical search** – Assumes the best multi-parameter model is created out of the combination of the best single parameter hypothesis for each parameter



- **Modified golden section search** – Speeds up the single parameter search by ordering the hypothesis space and then using a variant of binary search to find the model in logarithmic time rather than linear time

Calotoiu et al.

IEEE Cluster 2016
The Institute of Electrical and Electronics Engineers Cluster 2016
2016.09.13 – 15
Palais de Chine Hotel / Taipei, Taiwan

Search space reduction

- Assuming 300.000 hypotheses searched per second*
- 3-parameter models

$$n = 3$$

$$m = 3$$

$$I = \left\{ \frac{0}{4}, \frac{1}{4}, \ldots, \frac{12}{4} \right\}$$

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

Search space reduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Assuming 300.000 hypotheses searched per second*
- 3-parameter models

*This is optimistic

$$n = 3$$

$$m = 3$$

$$I = \left\{ \frac{0}{4}, \frac{1}{4}, \ldots, \frac{12}{4} \right\}$$

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

Exhaustive search

34.786.300.841.019 hypotheses searched

~1 model / 3.5 years

Search space reduction

- Assuming 300.000 hypotheses searched per second*
- 3-parameter models

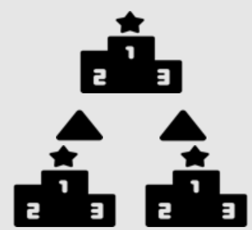
*This is optimistic

$$n = 3$$
$$m = 3$$
$$I = \left\{ \frac{0}{4}, \frac{1}{4}, \ldots, \frac{12}{4} \right\}$$
$$J = \{0, 1, 2\}$$

Exhaustive search

34.786.300.841.019 hypotheses searched

~1 model / 3.5 years



27.929 hypotheses searched

~11 models / second

Search space reduction

- Assuming 300.000 hypotheses searched per second*
- 3-parameter models

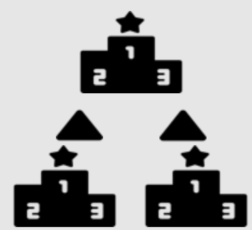
*This is optimistic

$$n = 3$$
$$m = 3$$
$$I = \left\{ \frac{0}{4}, \frac{1}{4}, \ldots, \frac{12}{4} \right\}$$
$$J = \{0, 1, 2\}$$

Exhaustive search

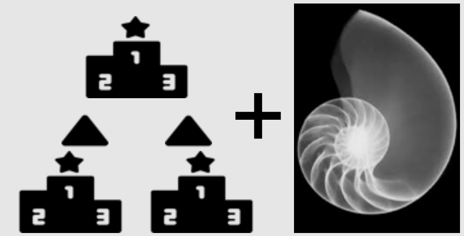
34.786.300.841.019 hypotheses searched

~1 model / 3.5 years



27.929 hypotheses searched

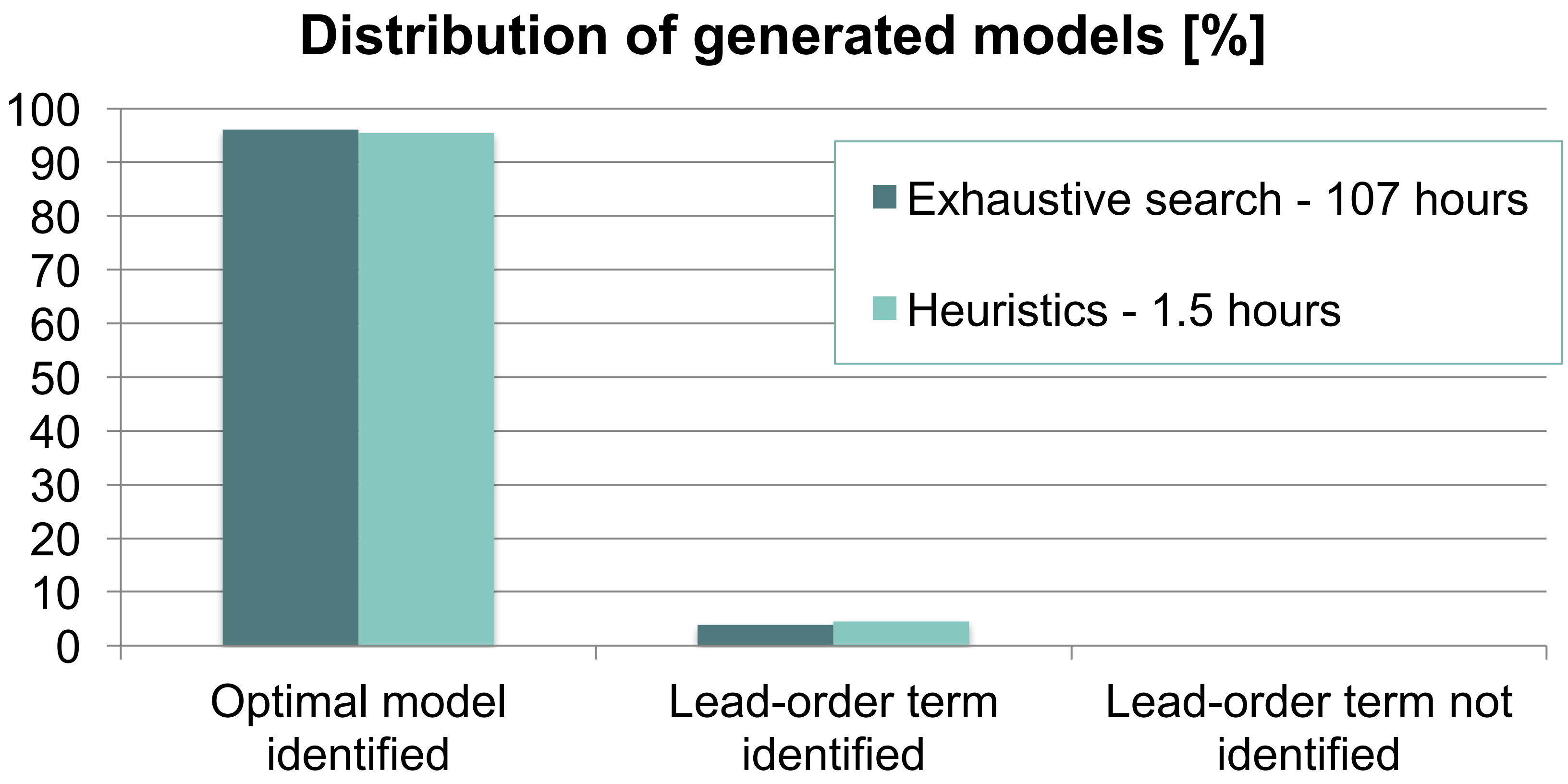
~11 models / second



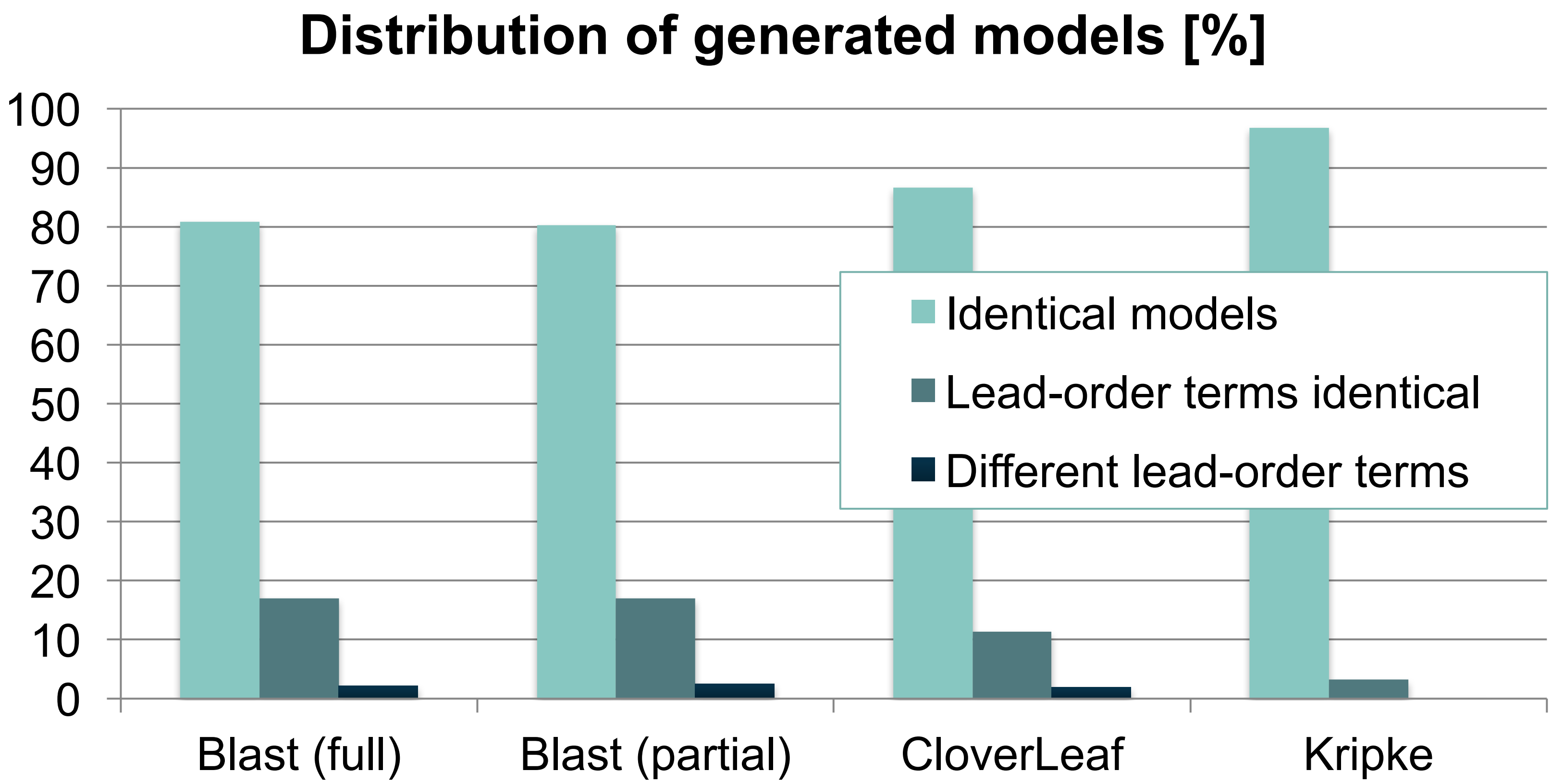
590 hypotheses searched

~508 models / second

Evaluation with synthetic data (100,000 models with two parameters)

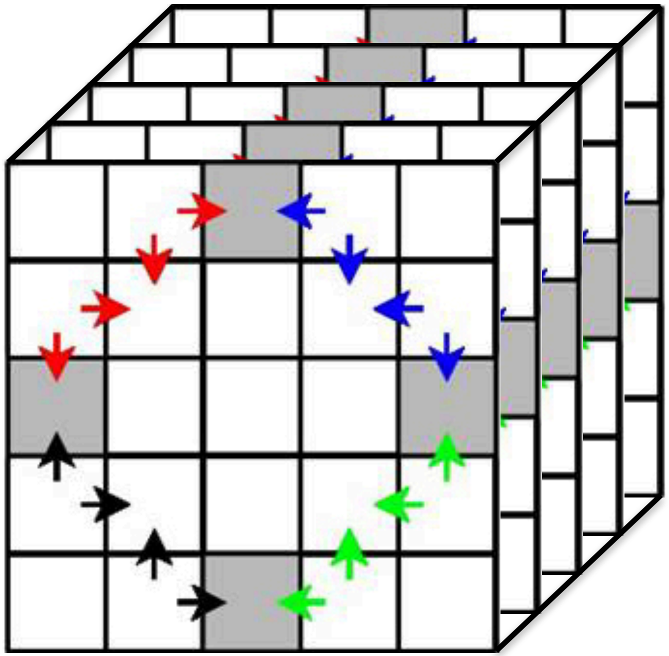


Evaluation with application data



Case study – Kripke

- Neutron transport proxy code
- Three parameters considered
  - Process count – p
  - Number of directions – d
  - Number of groups – g



Expected behavior

SweepSolver

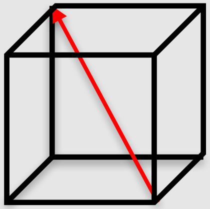
Main **computation** kernel

Expectation – Performance depends on **problem size**

$$t \sim d \cdot g$$

MPI_Testany

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



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

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

Expected behavior

TECHNISCHE UNIVERSITÄT DARMSTADT

SweepSolver

Main **computation** kernel

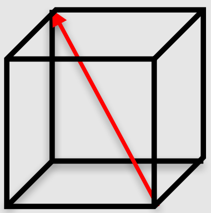
Expectation – Performance depends on **problem size**

$$t \sim d \cdot g$$

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:

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

Actual model:

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

Smaller compounded effect discovered

How to find good PMNF parameters?

Option **(1)** : Rely on *default parameters*

→ But what if they don‘t fit the problem?

Option **(2):** Try those parameters that you *expect* to fit

→ Requires prior expertise!

Also, what if your expectation is wrong?

Option **(3)**: Try *very large sets I, J*

→ Requires more resources (especially bad for multiple parameters)!

Option **(4)**: Let Extra-P *automatically refine* the search space based on previous results.

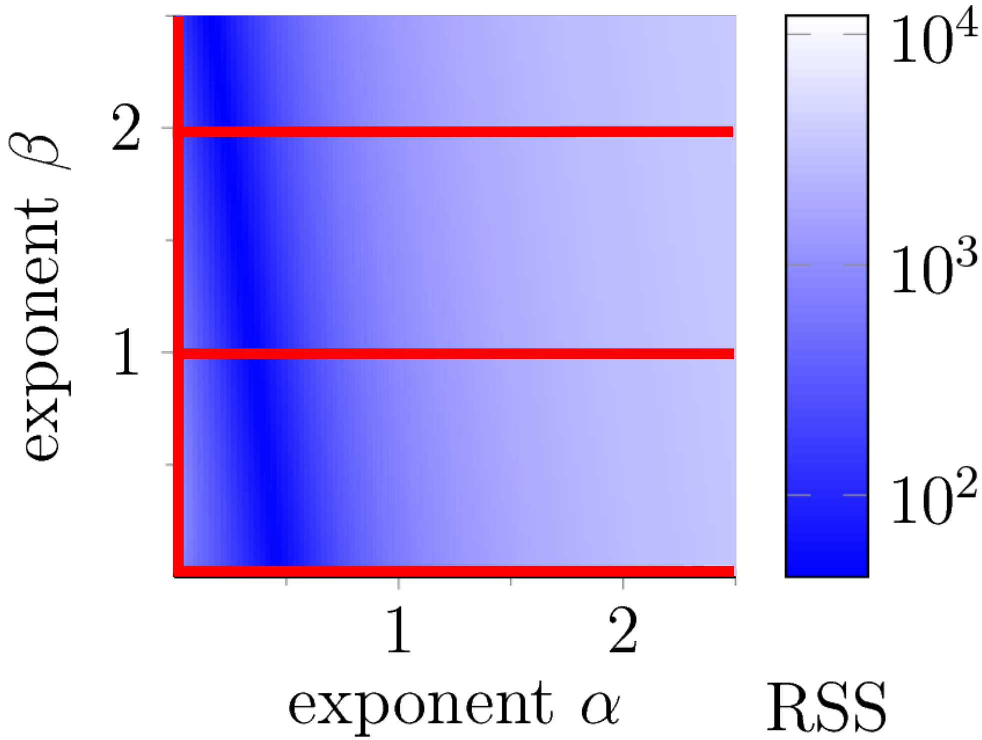
Simplified PMNF

- Use only constant and "lead order" term

$$f(p) = c_0 + c_1 \cdot p^{\alpha} \cdot \log_2^{\beta} p$$

- Want to find values for c_0, c_1, α, and β, such that model error is minimized
  - c_0 and c_1 are determined by regression
  - What about α and β?

Simplified PMNF $f(p) = c_0 + c_1 \cdot p^{\alpha} \cdot \log_2^{\beta} p$



We define four slices:

- β = 0, α = ?
- β = 1, α = ?
- β = 2, α = ?
- α = 0, β = ?

Goal:

Unimodal error distribution along each slice

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

Data from previous case studies

- Sweep3D
- MILC
- UG4
- MPI collective operations
- BLAST
- Kripke

- 5–9 points available
- Last data point (largest p) not used for modeling, but to evaluate prediction accuracy

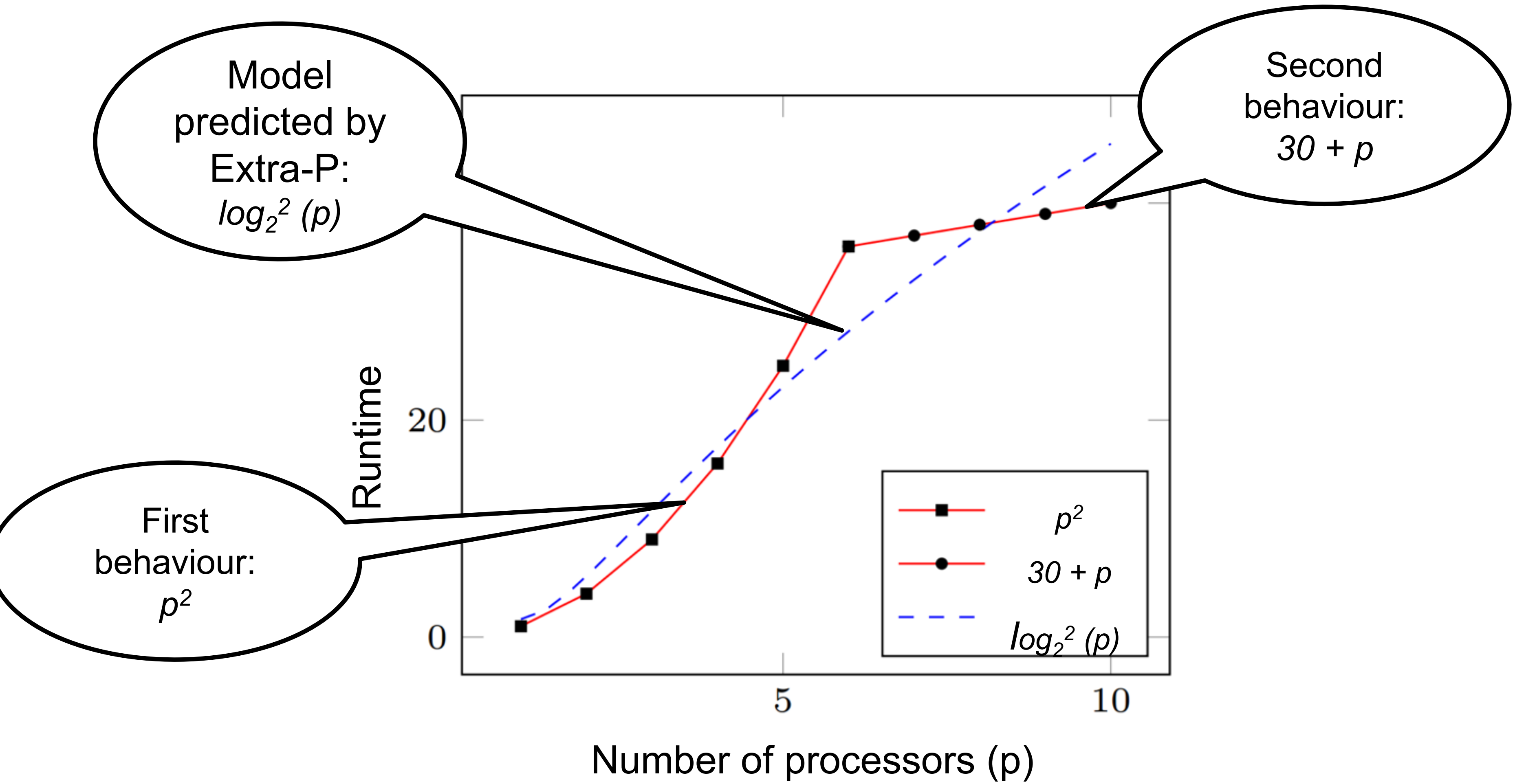
Results

- 4453 models
- 49% remain unchanged
- 39% get better
- 12% get worse
- Mean relative prediction down from 45.7% to 13.0%
- Improvements in every individual case study

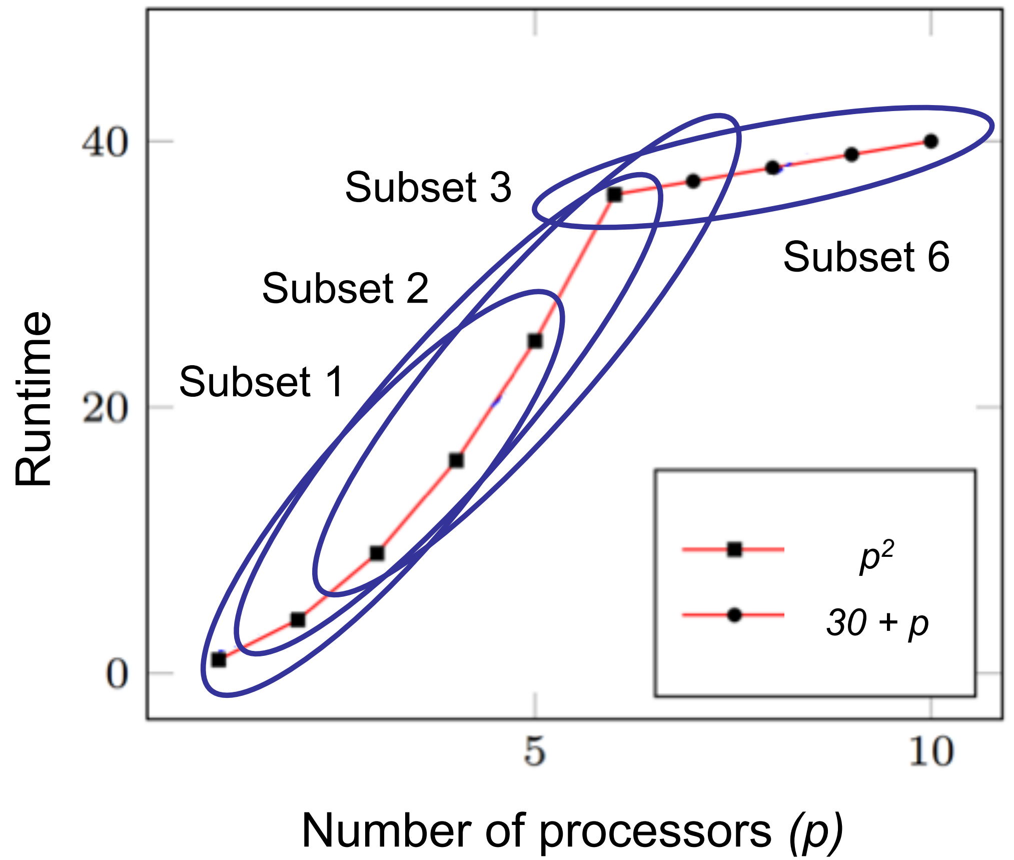
Reisert et al.

Euro-Par 2017

Segmented behavior



Divide data into subsets



Model each subset and compute nRSS

TECHNISCHE UNIVERSITÄT DARMSTADT

Normalized RSS

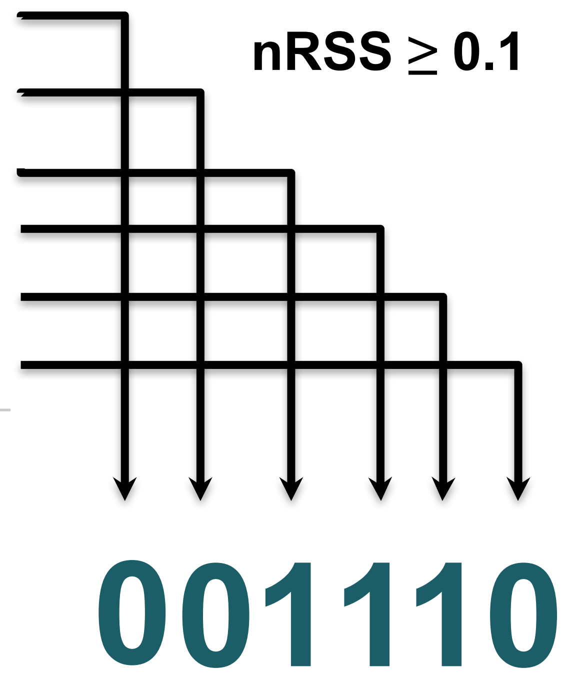
High nRSS values

Subset	Model	nRSS
$s_1 = \{1, 4, 9, 16, 25\}$	p^2	≈ 0
$s_2 = \{4, 9, 16, 25, 36\}$	p^2	≈ 0
$s_3 = \{9, 16, 25, 36, 37\}$	$-49.41 + 33.45 \cdot \sqrt{p}$	$\boxed{0.18}$
$s_4 = \{16, 25, 36, 37, 38\}$	$-28.53 + 23.17 \cdot log_2(p)$	$\boxed{0.19}$
$s_5 = \{25, 36, 37, 38, 39\}$	$-6.19 + 1$[illegible]	$\boxed{0.16}$
$s_6 = \{36, 37, 38, 39, 40\}$	$30 + p$[illegible]	≈ 0

Heterogeneous subsets

Identify change point

Subset	nRSS
$s_1 = \{1, 4, 9, 16, 25\}$	≈ 0
$s_2 = \{4, 9, 16, 25, 36\}$	≈ 0
$s_3 = \{9, 16, 25, 36, 37\}$	0.18
$s_4 = \{16, 25, 36, 37, 38\}$	0.19
$s_5 = \{25, 36, 37, 38, 39\}$	0.16
$s_6 = \{36, 37, 38, 39, 40\}$	≈ 0



Identify change point

TECHNISCHE UNIVERSITÄT DARMSTADT

Valid Patterns

....00000111110000...

....0000011110000...

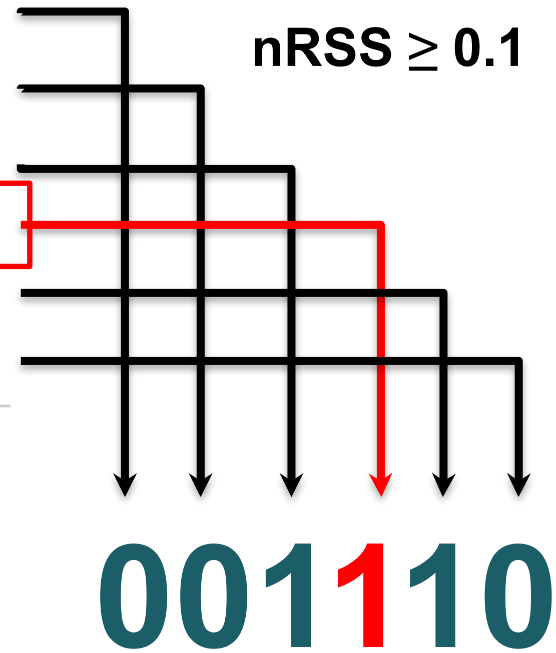
Just Noise

....01000110010...

Identifying the change point

Subset	nRSS
$s_1 = \{1, 4, 9, 16, 25\}$	≈ 0
$s_2 = \{4, 9, 16, 25, 36\}$	≈ 0
$s_3 = \{9, 16, 25, 36, 37\}$	0.18
$s_4 = \{16, 25, 36, 37, 38\}$	0.19
$s_5 = \{25, 36, 37, 38, 39\}$	0.16
$s_6 = \{36, 37, 38, 39, 40\}$	≈ 0

Change Point



HOMME

- Dynamic core of Community Atmosphere Model (CAM)
- Run for p $\in$ {600; 1,176; …; 54,150}
- 25 out of 664 kernels found segmented
- Change point found between 15,000 and 16,224
- Example: *laplace_sphere_wk*

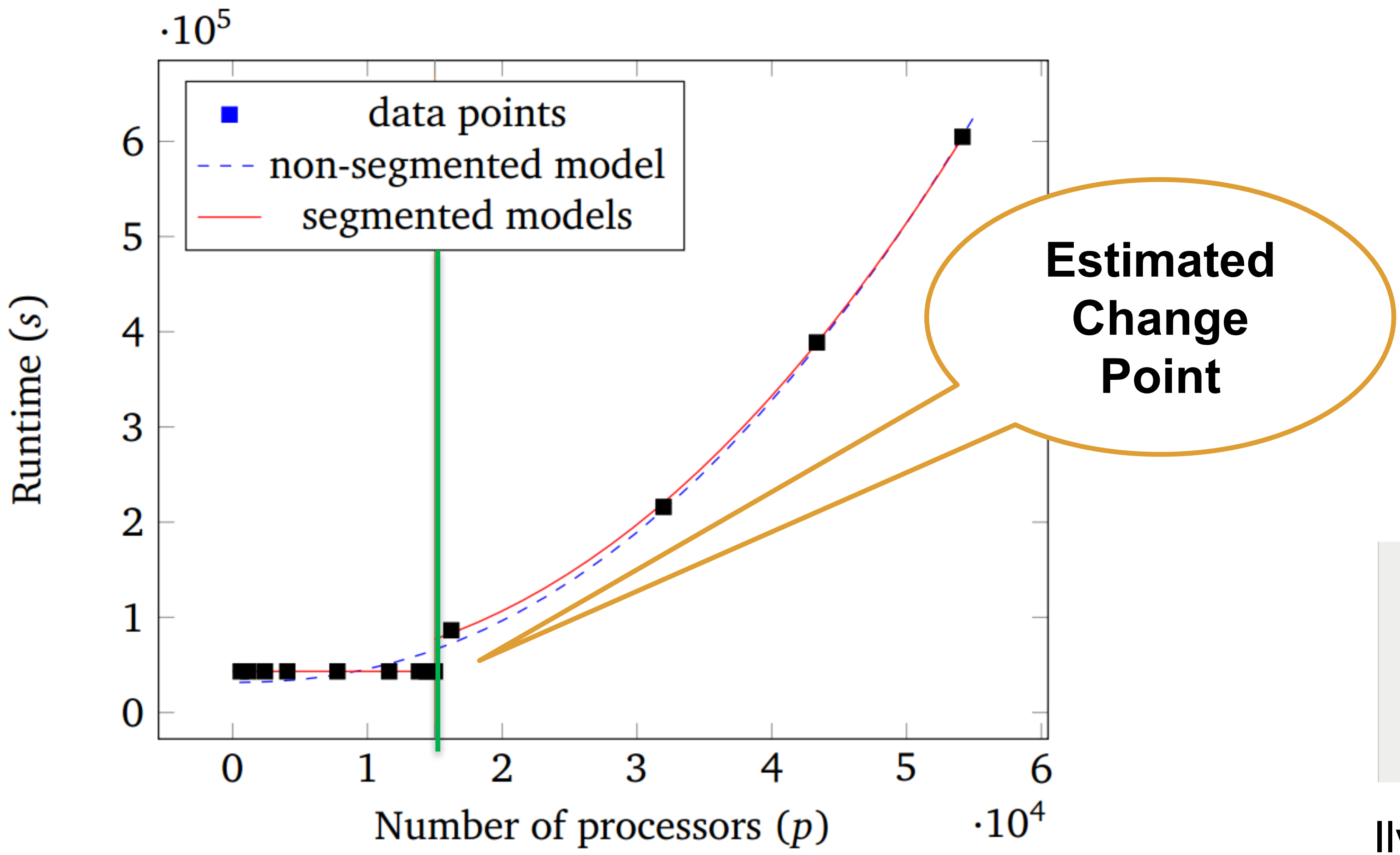
Non-segmented model:

$$f(p) = 27.7 + 2.23 \cdot 10^{-7} \cdot p^2$$

Segmented model:

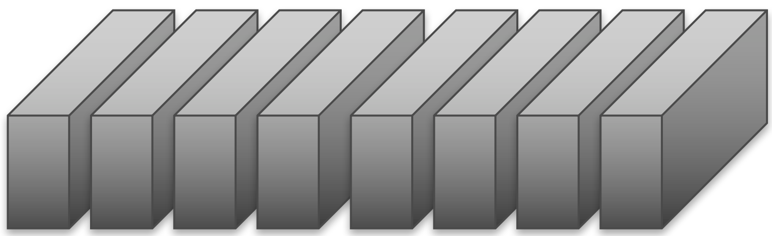
$$f_{seg}(p) = \begin{cases} 49.36 & p \leq 15,000 \\ 20.8 + 2.3 \cdot 10^{-7} \cdot p^2 & p \geq 16,224 \end{cases}$$

HOMME



Ilyas et al.

System upgrade

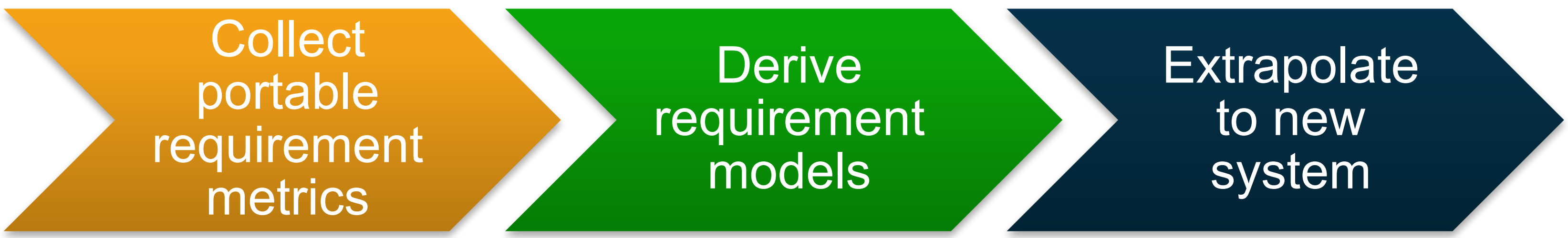


Given a budget and a set of applications, how can we best invest in upgrades for a given hardware system?

Examples

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

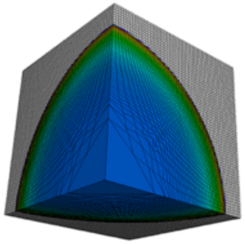
Lightweight requirements engineering for (exascale) co-design



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

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


Lulesh

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

Response of workload to system upgrades

TECHNISCHE UNIVERSITÄT DARMSTADT

Ratios \ Apps.	Kripke	LULESH	MILC	Relearn	icoFoam	**Baseline**
System upgrade A: Double the racks						
Problem size per process	1	1	1	1	0.5	**1**
Overall problem size	2	2	2	2	1	**2**
Computation	1	1.2	1	1	0.5	**1**
Communication	1	1.2	1	1	0.7	**1**
Memory access	2	1.2	2.8	2	0.7	**1**
System upgrade B: Double the sockets						
Problem size per process	0.5	0.5	0.5	0.3	0.3	**0.5**
Overall problem size	1	1	1	0.6	0.6	**1**
Computation	0.5	0.6	0.5	0.3	0.2	**0.5**
Communication	0.5	0.6	0.5	0.3	0.3	**0.5**
Memory access	0.5	1	1.4	1	0.5	**0.5**
System upgrade C: Double the memory						
Problem size per process	2	1.4	2	2.8	1.4	**2**
Overall problem size	2	1.4	2	2.8	1.4	**2**
Computation	2	1.4	2	2.8	1.7	**2**
Communication	2	1.4	2	2.8	1.4	**2**
Memory access	2	1.4	2	2.8	1.4	**2**

Calotoiu et al.

Best option for Lulesh

Worst option for Lulesh

Publications

[1] Alexandru Calotoiu, Alexander Graf, Torsten Hoefler, Daniel Lorenz, Felix Wolf: Lightweight Requirements Engineering for Exascale Co-design. In Proc. of the 2018 IEEE International Conference on Cluster Computing (CLUSTER), Belfast, UK (accepted)

[2] Sebastian Rinke, Markus Butz-Ostendorf, Marc-André Hermanns, Mikaël Naveau, Felix Wolf: A Scalable Algorithm for Simulating the Structural Plasticity of the Brain. Journal of Parallel and Distributed Computing, 2018.

[3] Patrick Reisert, Alexandru Calotoiu, Sergei Shudler, Felix Wolf: Following the Blind Seer – Creating Better Performance Models Using Less Information. In Proc. of the 23rd Euro-Par Conference, Santiago de Compostela, Spain

[4] Kashif Ilyas, Alexandru Calotoiu, Felix Wolf: Off-Road Performance Modeling – How to Deal with Segmented Data. In Proc. of the 23rd Euro-Par Conference, Santiago de Compostela, Spain

[5] Sergei Shudler, Alexandru Calotoiu, Torsten Hoefler, Felix Wolf: Isoefficiency in Practice: Configuring and Understanding the Performance of Task-based Applications. In Proc. of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), Austin, TX, USA

[6] Alexandru Calotoiu, David Beckingsale, Christopher W. Earl, Torsten Hoefler, Ian Karlin, Martin Schulz, Felix Wolf: Fast Multi-Parameter Performance Modeling. In Proc. of the 2016 IEEE International Conference on Cluster Computing (CLUSTER), Taipei, Taiwan

[7] Felix Wolf, Christian Bischof, Alexandru Calotoiu, Torsten Hoefler, Christian Iwainsky, Grzegorz Kwasniewski, Bernd Mohr, Sergei Shudler, Alexandre Strube, Andreas Vogel, Gabriel Wittum: *Software for Exascale Computing - SPPEXA 2013-2015*, Chapter *Automatic Performance Modeling of HPC Applications*. Springer, pages 445–465, 2016.

[8] Andreas Vogel, Alexandru Calotoiu, Arne Nägel, Sebastian Reiter, Alexandre Strube, Gabriel Wittum, Felix Wolf: Software for Exascale Computing - SPPEXA 2013-2015, chapter Automated Performance Modeling of the UG4 Simulation Framework.

[9] Christian Iwainsky, Sergei Shudler, Alexandru Calotoiu, Alexandre Strube, Michael Knobloch, Christian Bischof, Felix Wolf: How Many Threads will be too Many? On the Scalability of OpenMP Implementations. In Proc. of the 21st Euro-Par Conference, Vienna, Austria

[10] Andreas Vogel, Alexandru Calotoiu, Alexandre Strube, Sebastian Reiter, Arne Nägel, Felix Wolf, Gabriel Wittum: 10,000 Performance Models per Minute - Scalability of the UG4 Simulation Framework. In Proc. of the 21st Euro-Par Conference, Vienna, Austria

[11] Sergei Shudler, Alexandru Calotoiu, Torsten Hoefler, Alexandre Strube, Felix Wolf: Exascaling Your Library: Will Your Implementation Meet Your Expectations?. In Proc. of the International Conference on Supercomputing (ICS), Newport Beach, CA, USA

[12] Alexandru Calotoiu, Torsten Hoefler, Felix Wolf: Mass-producing Insightful Performance Models. In *Workshop on Modeling & Simulation of Systems and Applications, University of Washington*, Seattle, Washington, USA

[13] Felix Wolf, Christian Bischof, Torsten Hoefler, Bernd Mohr, Gabriel Wittum, Alexandru Calotoiu, Christian Iwainsky, Alexandre Strube, Andreas Vogel: Catwalk: A Quick Development Path for Performance Models. In Euro-Par 2014: Parallel Processing Workshops

[14] Alexandru Calotoiu, Torsten Hoefler, Marius Poke, Felix Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes. In Proc. of the ACM/IEEE Conference on Supercomputing (SC13), Denver, CO, USA

7th Workshop on Extreme Scale Programming Tools (ESPT'18)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Performance tools
- Debugging and correctness tools
- Program development tool chains (incl. IDEs)
- Performance engineering
- Tool technologies for extreme-scale challenges (e.g., scalability, resilience, power)
- Tool support for accelerated architectures
- Tools for networks and I/O
- Tool infrastructures and environments
- Application developer experiences

SC18 Dallas, TX | hpc inspires.

Author stipends !

SPPEXA