Mass-producing insightful performance models of parallel applications

Felix Wolf, TU Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Acknowledgement

- Alexandru Calotoiu (TU Darmstadt)
- Torsten Höfler (ETH Zurich)
- Sergei Shudler (TU Darmstadt)
- Alexandre Strube (Jülich Supercomputing Centre)
- Andreas Vogel (GU Frankfurt)

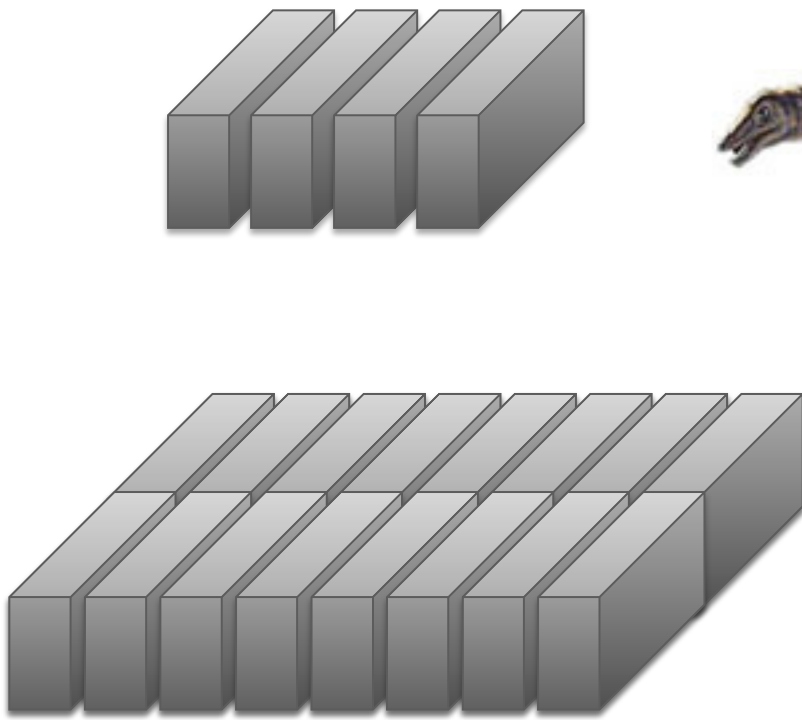
- Marius Poke (RWTH Aachen)
- Paul Wiedeking (RWTH Aachen)

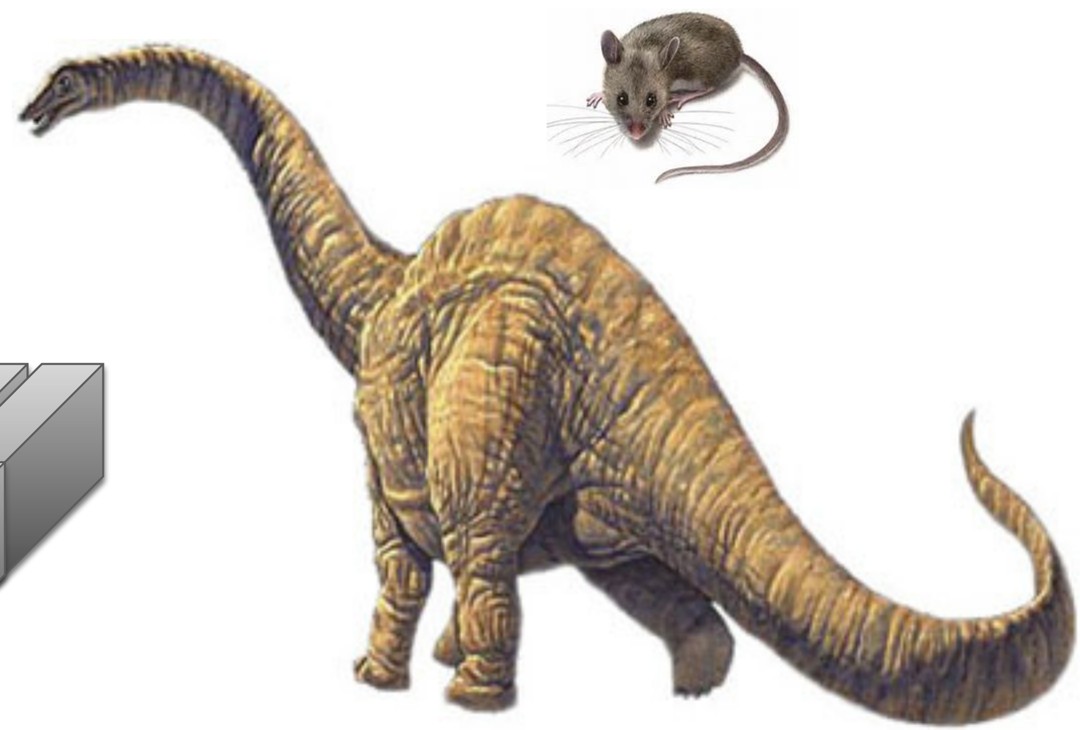
DFG SPPEXA FNSNF

Latent scalability bugs

System size

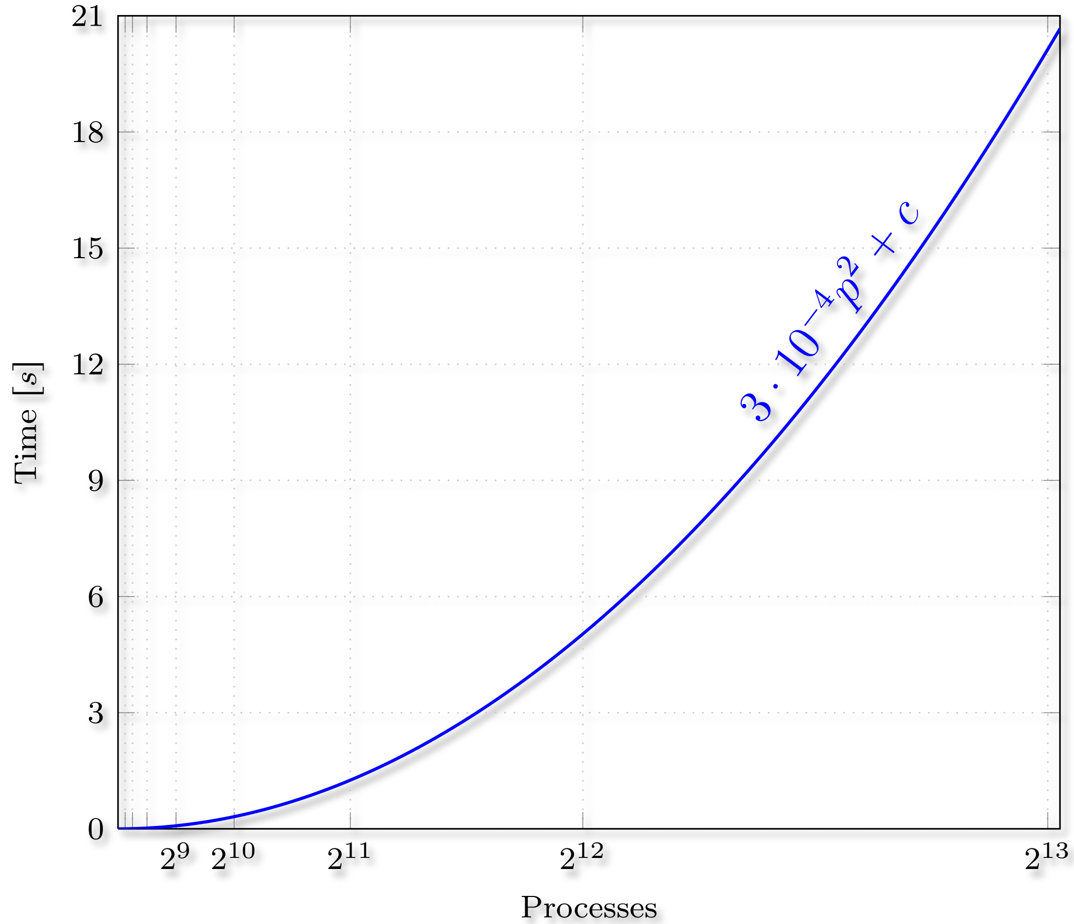
Execution time





Scalability model

TECHNISCHE UNIVERSITÄT DARMSTADT



Analytical scalability modeling

Identify kernels

- Parts of the program that dominate its performance at larger scales
- Identified via small-scale tests and intuition

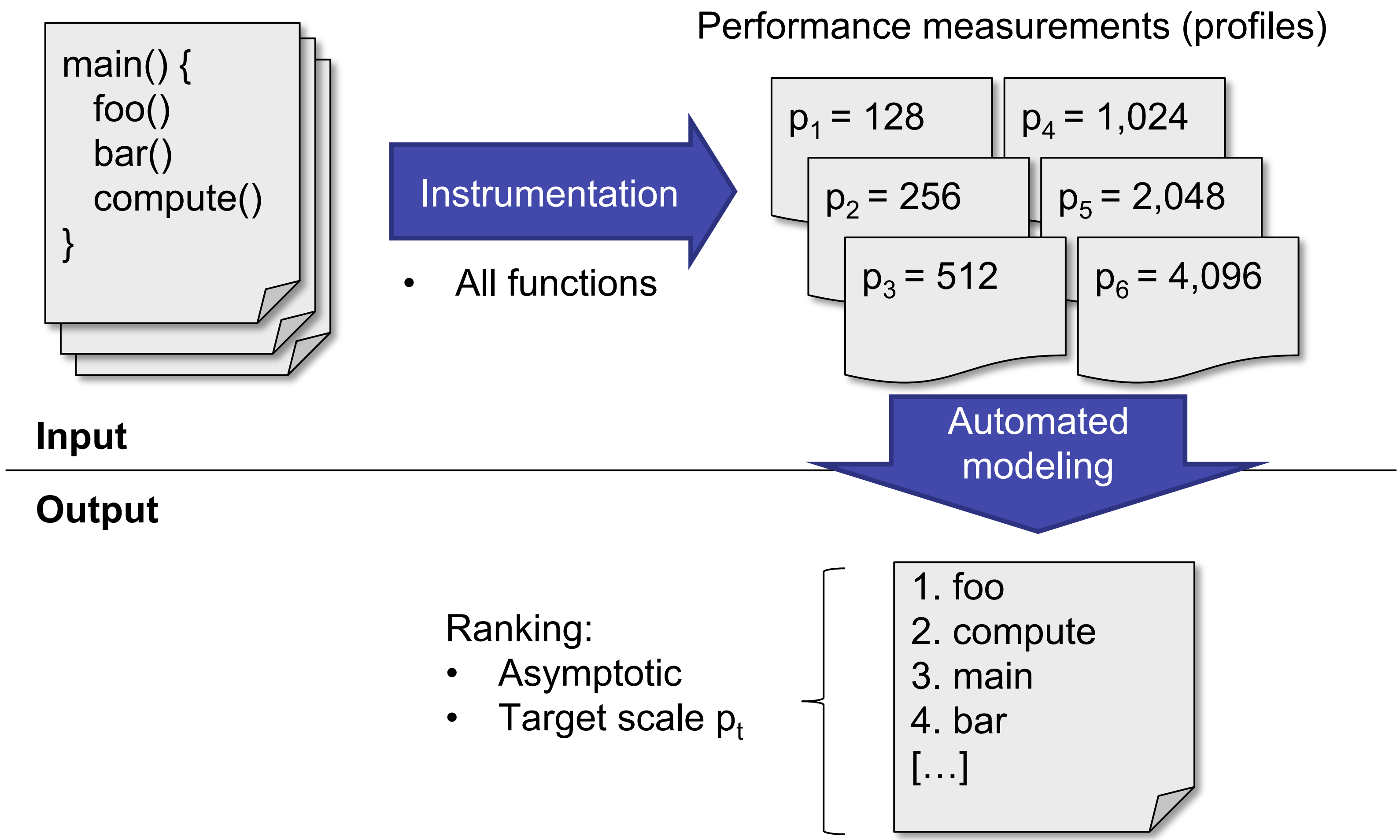
Create models

- Laborious process
- Still confined to a small community of skilled experts

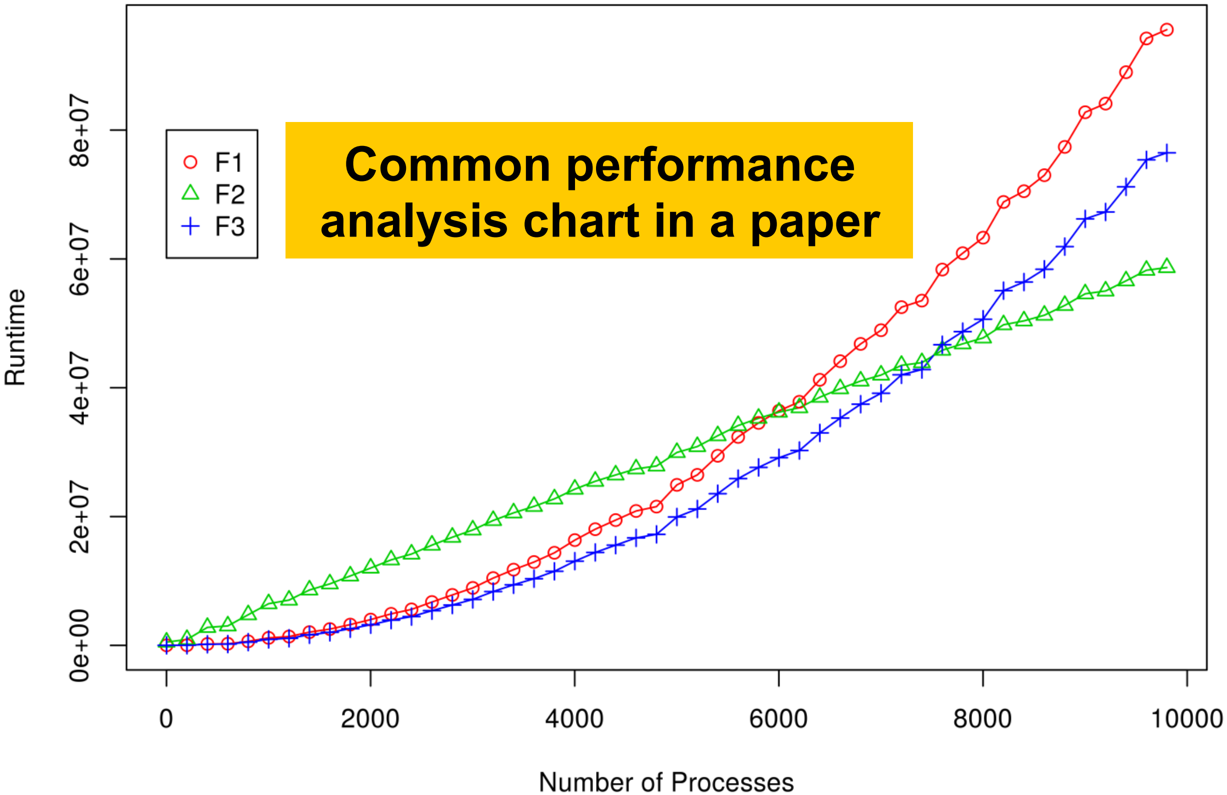
Disadvantages

- Time consuming
- Danger of overlooking unscalable code

Automated empirical modeling (2)



Primary focus on scaling trend

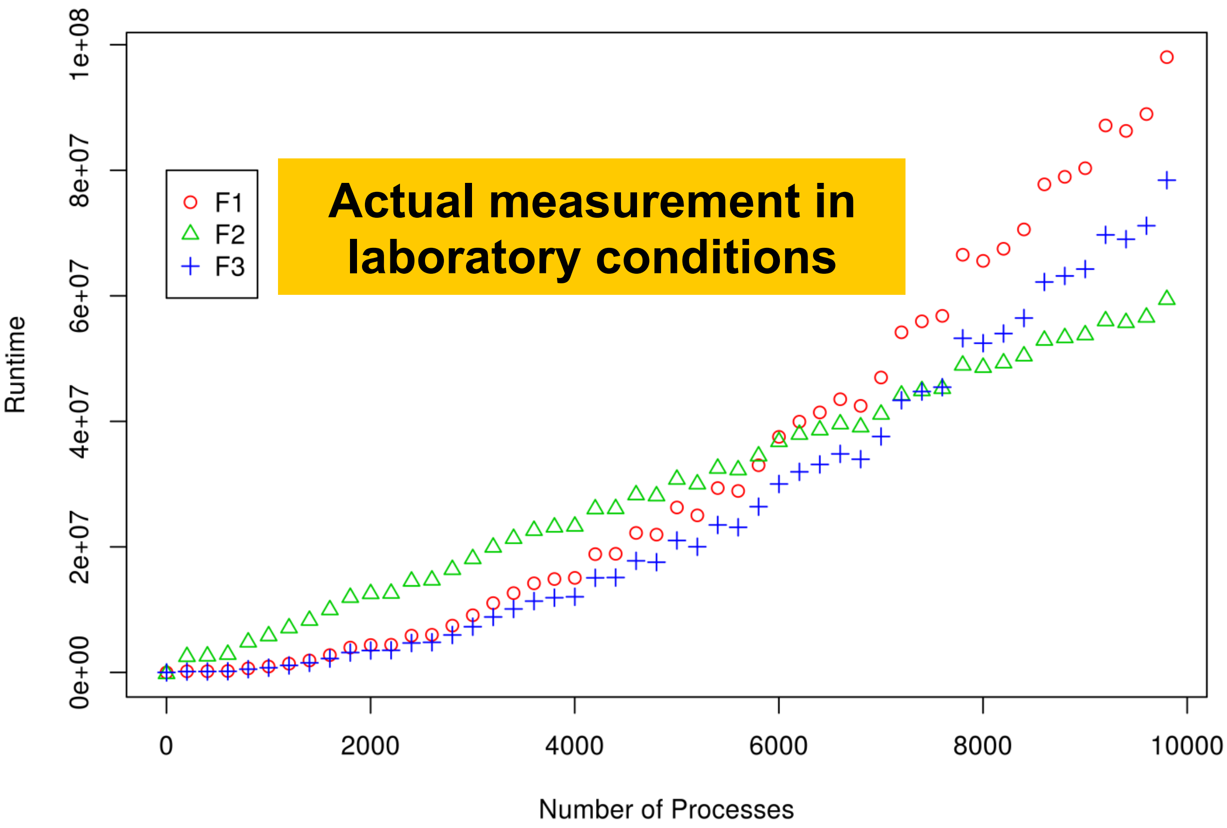


Our ranking

1. F_1
2. F_3
3. F_2

Primary focus on scaling trend

TECHNISCHE UNIVERSITÄT DARMSTADT

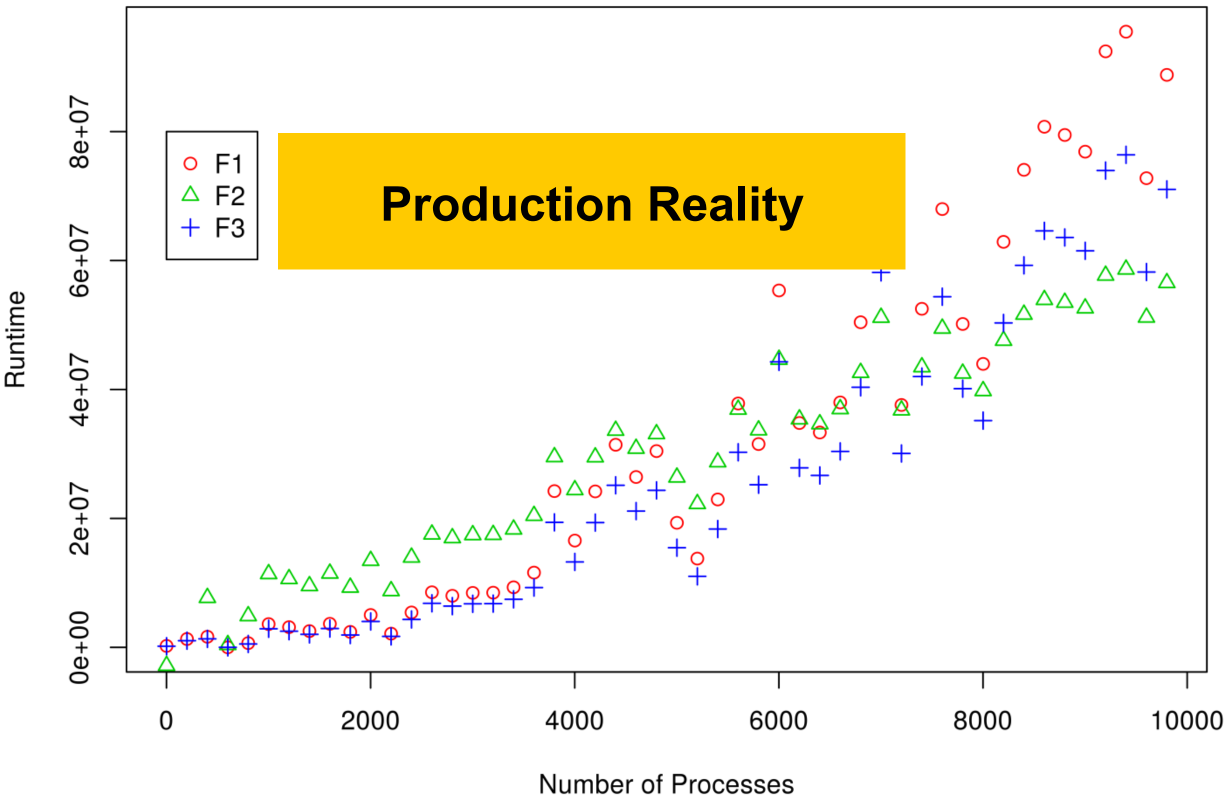


Our ranking

1. F_1
2. F_3
3. F_2

Primary focus on scaling trend

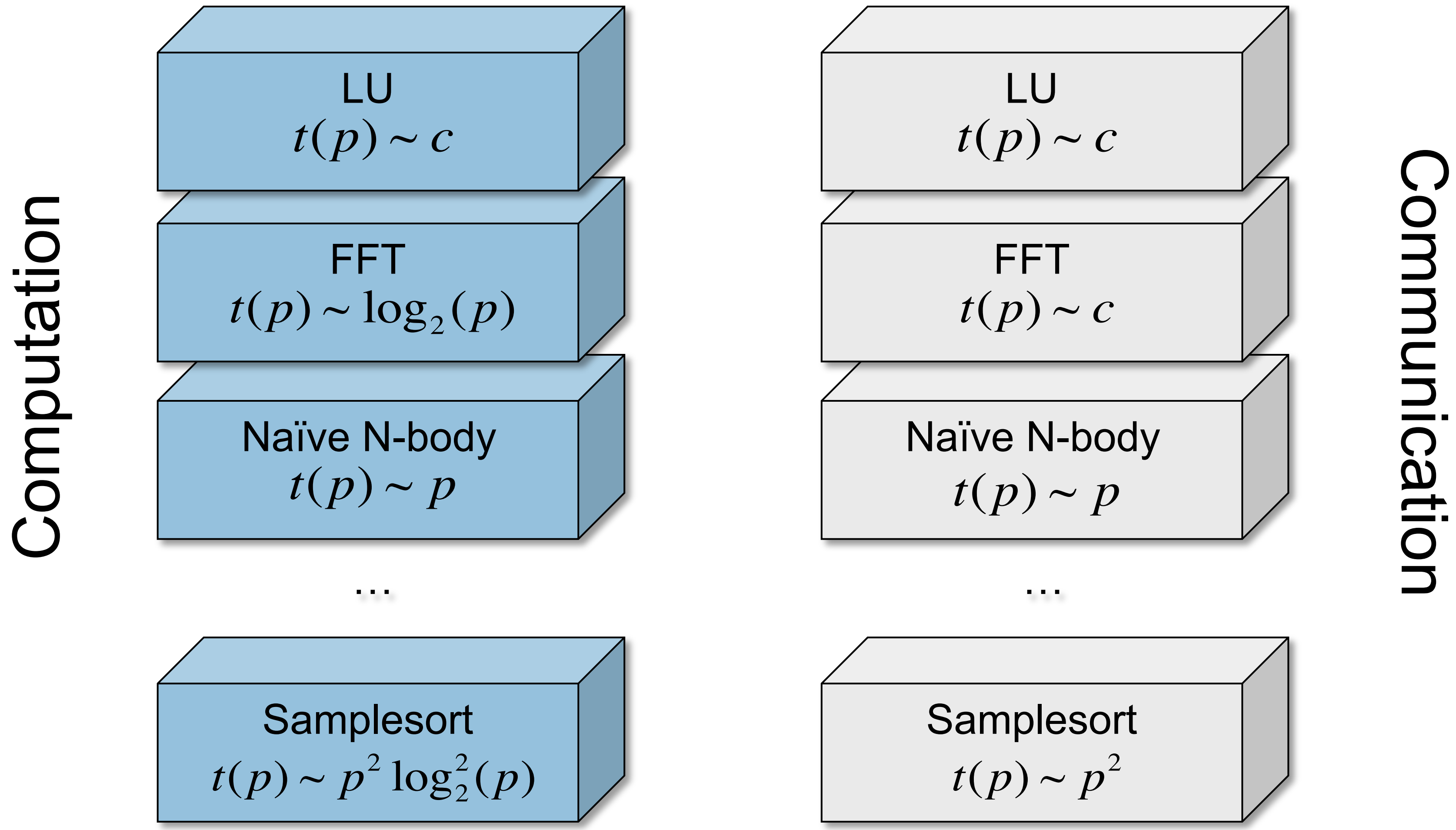
TECHNISCHE UNIVERSITÄT DARMSTADT



Our ranking

1. F_1
2. F_3
3. F_2

Model building blocks



Performance model normal form

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

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

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

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

$n \in \mathbb{N}$

$n = 2$

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

$J = \{0,1\}$

$c_1 + c_2 \cdot p$

$c_1 + c_2 \cdot p^2$

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

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

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

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

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

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

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

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

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

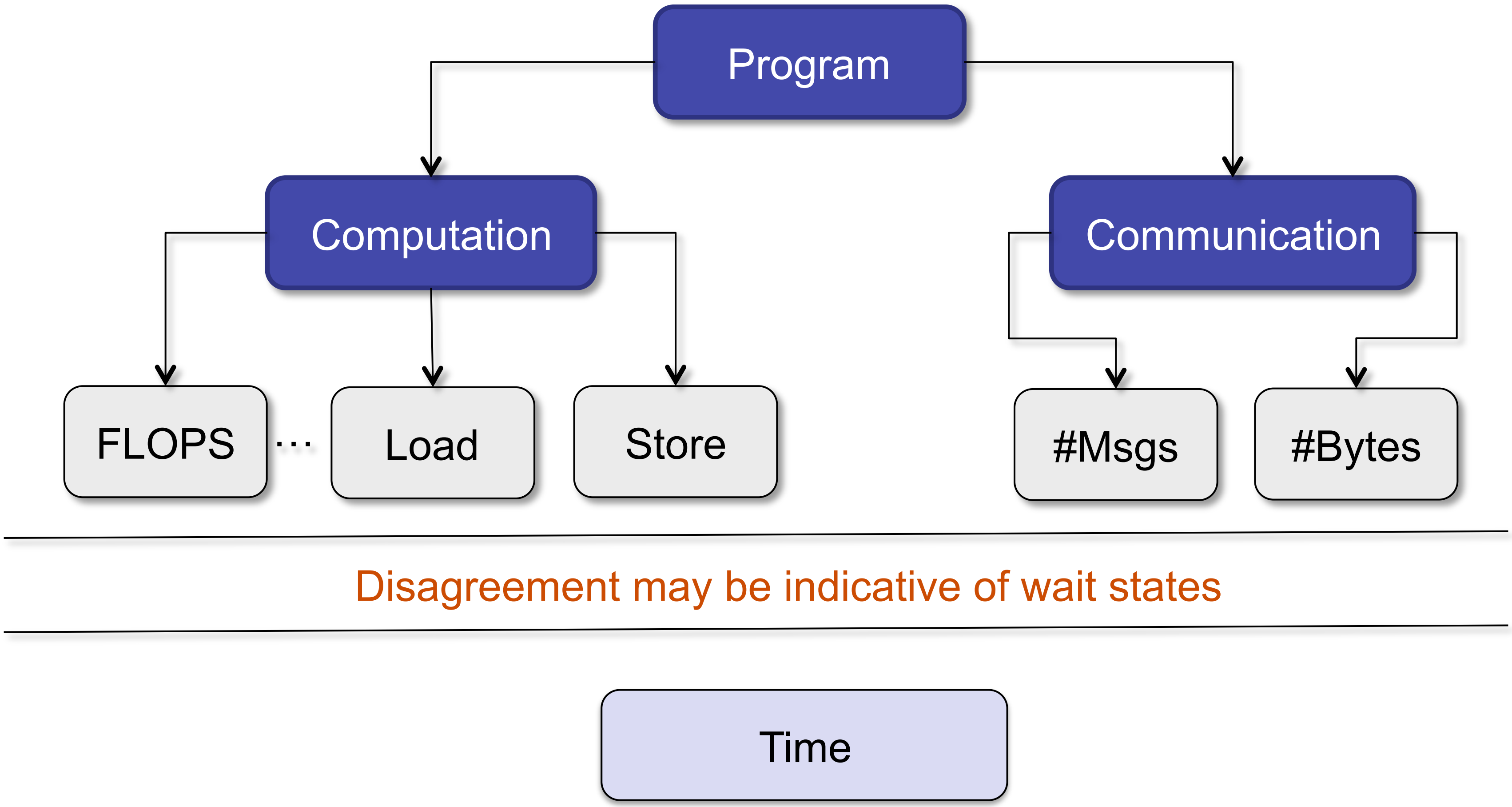
$c_1 \cdot p + c_2 \cdot p^2 \cdot \log(p)$

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

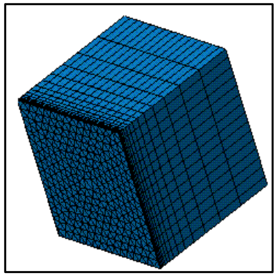
$c_1 \cdot p \cdot \log(p) + c_2 \cdot p^2 \cdot \log(p)$

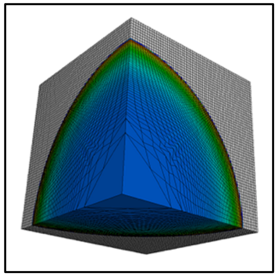
$c_1 \cdot p^2 + c_2 \cdot p^2 \cdot \log(p)$

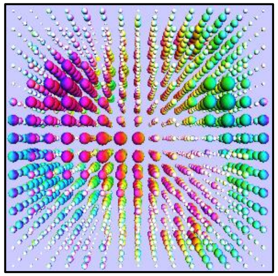
Modeling operations vs. time

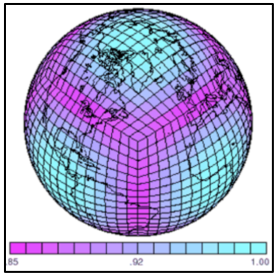


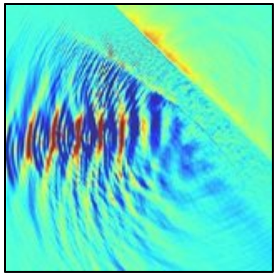
Case studies

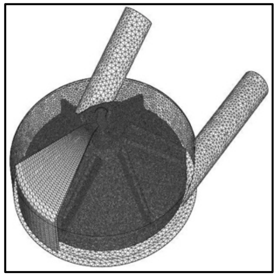

Sweep3d


Lulesh

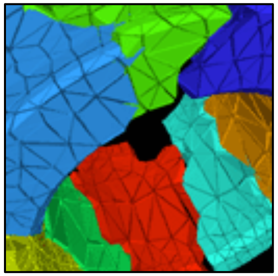

Milc

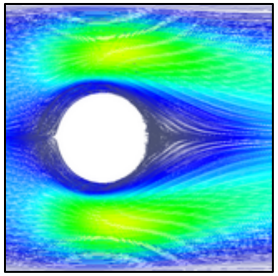

HOMME

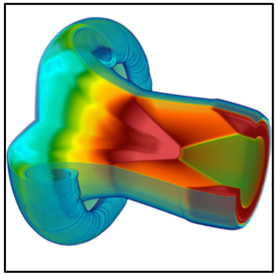

JUSPIC


XNS


NEST


UG4


MP2C


BLAST

MPI

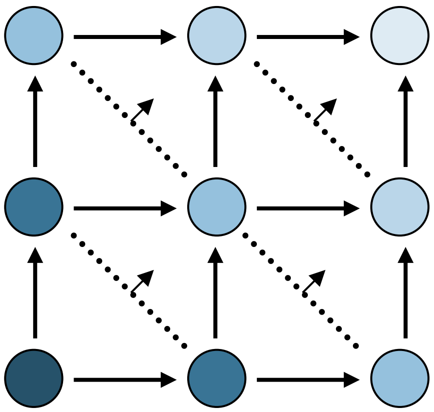
OpenMP

Sweep3D - Neutron transport simulation

- LogGP model for communication developed by Hoisie et al.

$$t^{comm} = [2(p_x + p_y - 2) + 4(n_{sweep} - 1)] \cdot t_{msg}$$

$$t^{comm} = c \cdot \sqrt{p}$$



Kernel [2 of 40]	Model [s] t = f(p)	Predictive error [%] p_t=262k
sweep → MPI_Recv	$4.03\sqrt{p}$	5.10
sweep	582.19	.01

#bytes = const.
#msg = const.

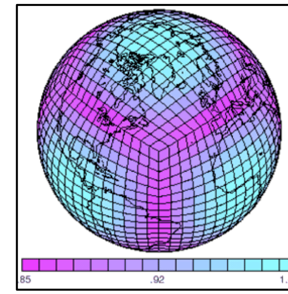
$p_i \leq 8\text{k}$

HOMME – Climate

TECHNISCHE UNIVERSITÄT DARMSTADT

Core of the Community Atmospheric Model (CAM)

- Spectral element dynamical core on a cubed sphere grid



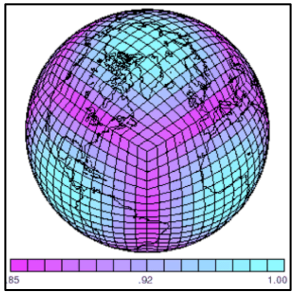
Kernel [3 of 194]	Model [s] $t = f(p)$	Predictive error [%] $p_t = 130k$
box_rearrange → MPI_Reduce	$0.026 + 2.53 \cdot 10^{-6} p \cdot \sqrt{p} + 1.24 \cdot 10^{-12} p^3$	57.02
vlaplace_sphere_vk	49.53	99.32
compute_and_apply_rhs	48.68	1.65

$$p_i \leq 15\text{k}$$

HOMME – Climate

Core of the Community Atmospheric Model (CAM)

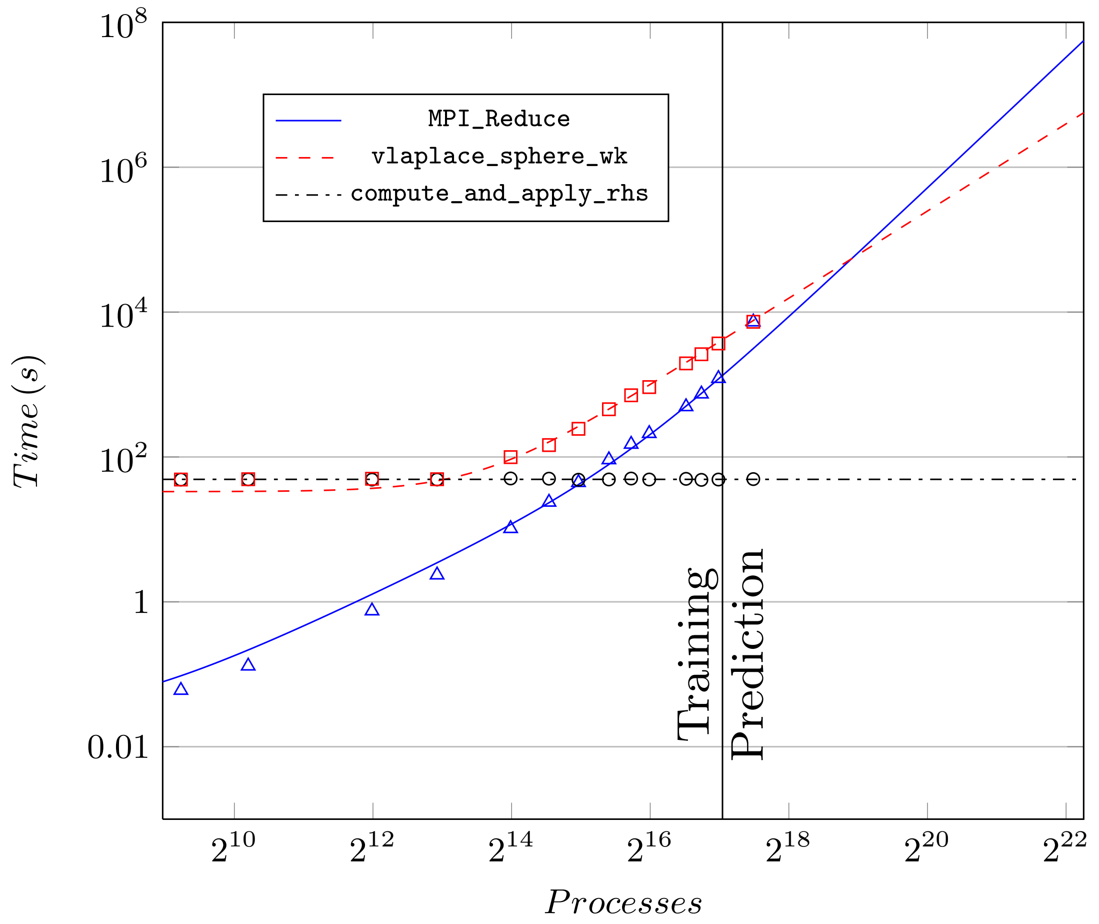
- Spectral element dynamical core on a cubed sphere grid



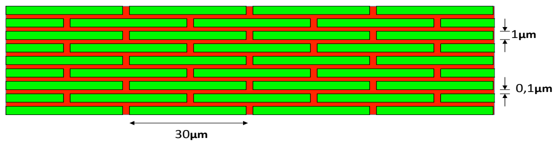
Kernel [3 of 194]	Model [s] $t = f(p)$	Predictive error [%] $p_t = 130k$
box_rearrange → MPI_Reduce	$3.63 \cdot 10^{-6} p \cdot \sqrt{p} + 7.21 \cdot 10^{-13} p^3$	30.34
vlaplace_sphere_vk	$24.44 + 2.26 \cdot 10^{-7} p^2$	4.28
compute_and_apply_rhs	49.09	0.83

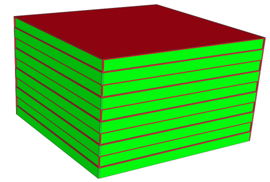
$p_i \leq 43\text{k}$

HOMME – Climate (2)



UG4





TECHNISCHE UNIVERSITÄT DARMSTADT

- Numerical framework for grid-based solution of partial differential equations (~500,000 lines of C++ code, 2,000 kernels)
  - Application: drug diffusion through the human skin
- In general, all kernels scale well
  - Multigrid solver kernel (MGM) scales logarithmically
  - Number of iterations needed by the unpreconditioned conjugate gradient (CG) method depends on the mesh size
    - Increases by factor of two with each refinement
    - Will therefore suffer from iteration count increase in weak scaling

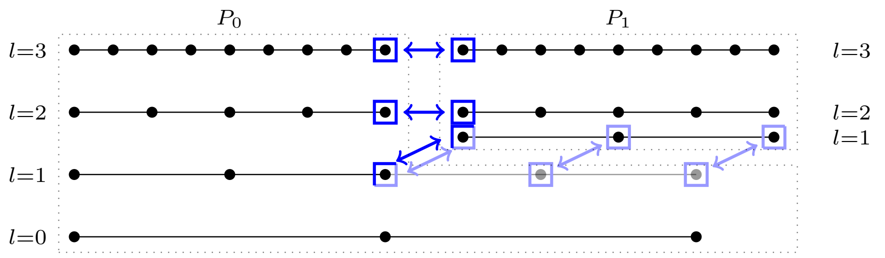
Kernel	Model (time [s])
CG	$0.227 + 0.31 * p^{0.5}$
MGM	$0.219 + 0.0006 * \log^2(p)$

Issue with MPI communicator group creation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Create MPI communicator groups for each level of multigrid hierarchy

- Exclude processes that do not own a grid part on that level



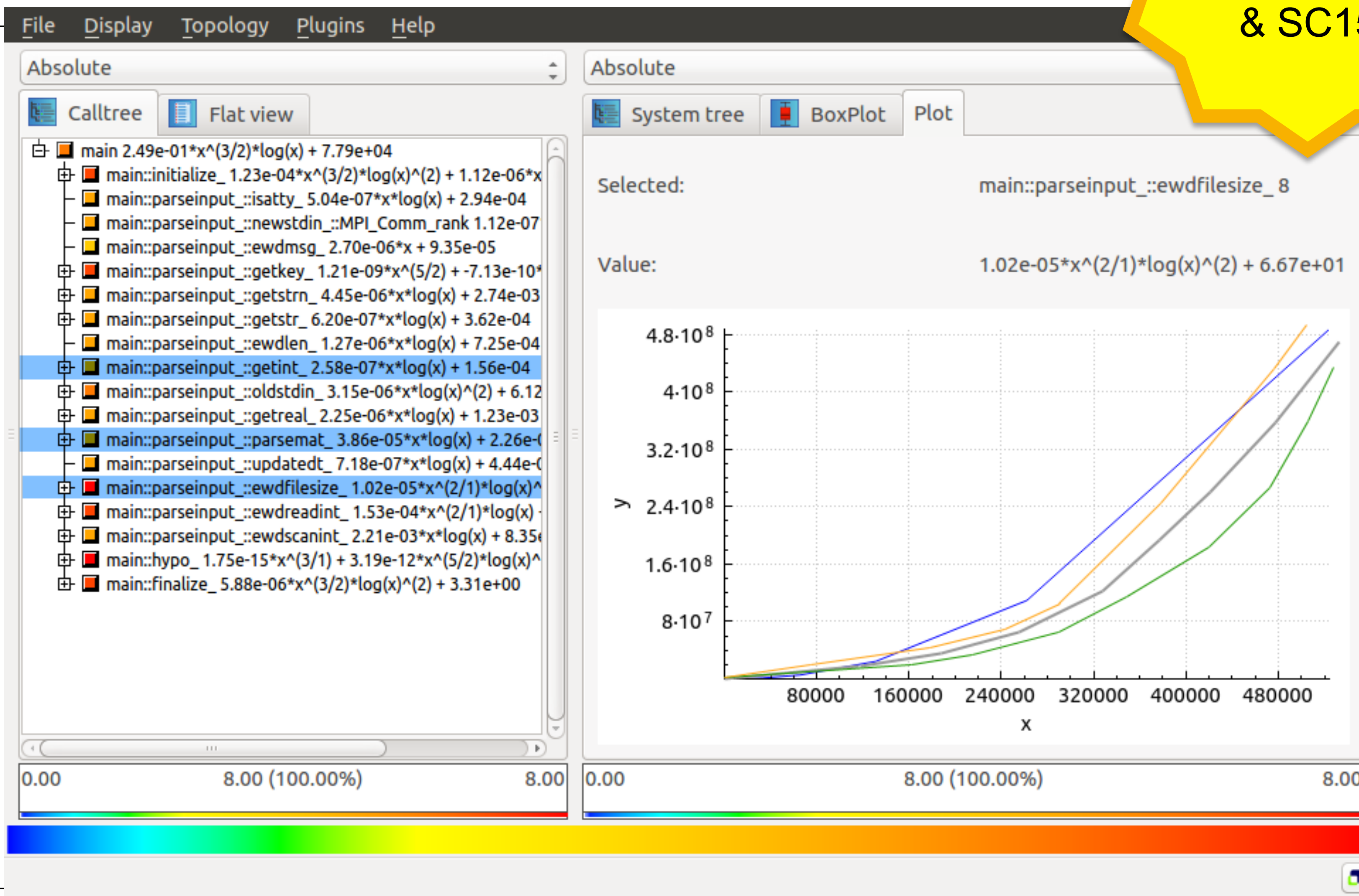
- *Before*: Membership info communicated using MPI_Allreduce with array of length p - non-scalable *p * O*(*MPI_Allreduce*) complexity

- *Now*: MPI_Allreduce replaced by MPI_Comm_split - enhanced algorithms of which are known to have $O(\log^2 p)$ complexity

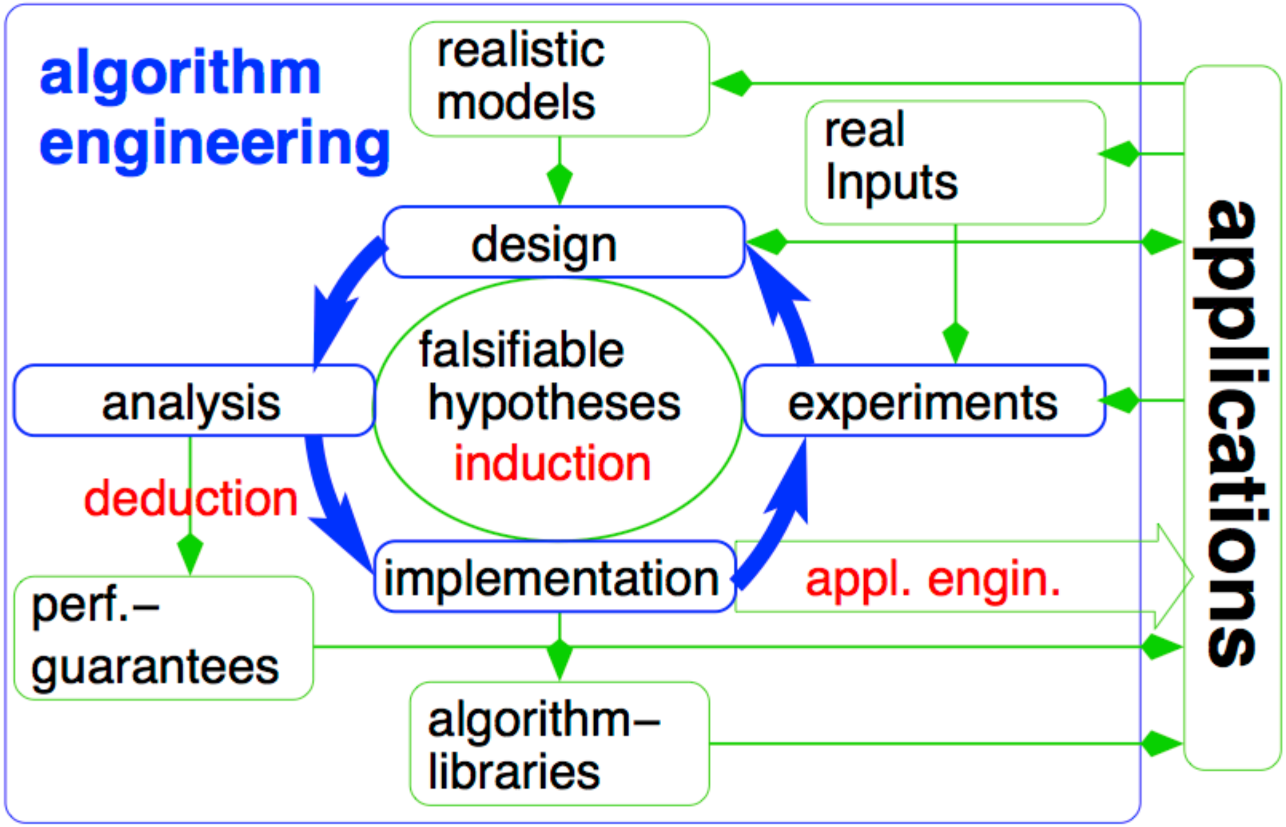
(C. Siebert, F. Wolf: Parallel sorting with minimal data. Recent Advances in the Message Passing Interface, 2011)

scalasca

Tutorials at EuroMPI'15 & SC15



Algorithm engineering



Courtesy of Peter Sanders, KIT

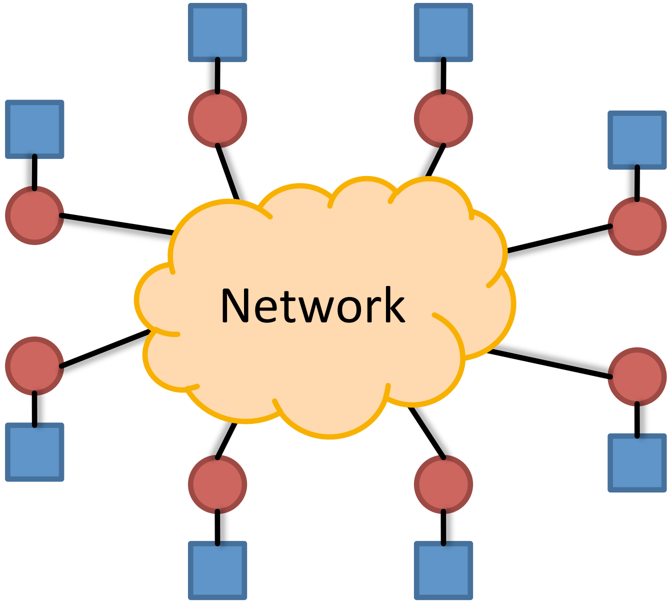
How to validate scalability in practice?

TECHNISCHE UNIVERSITÄT DARMSTADT

Program	Small text book example	Real application
Expectation	Verifiable analytical expression	Asymptotic complexity
	#FLOPS = $n^2(2n - 1)$	#FLOPS = $O(n^{2.8074})$

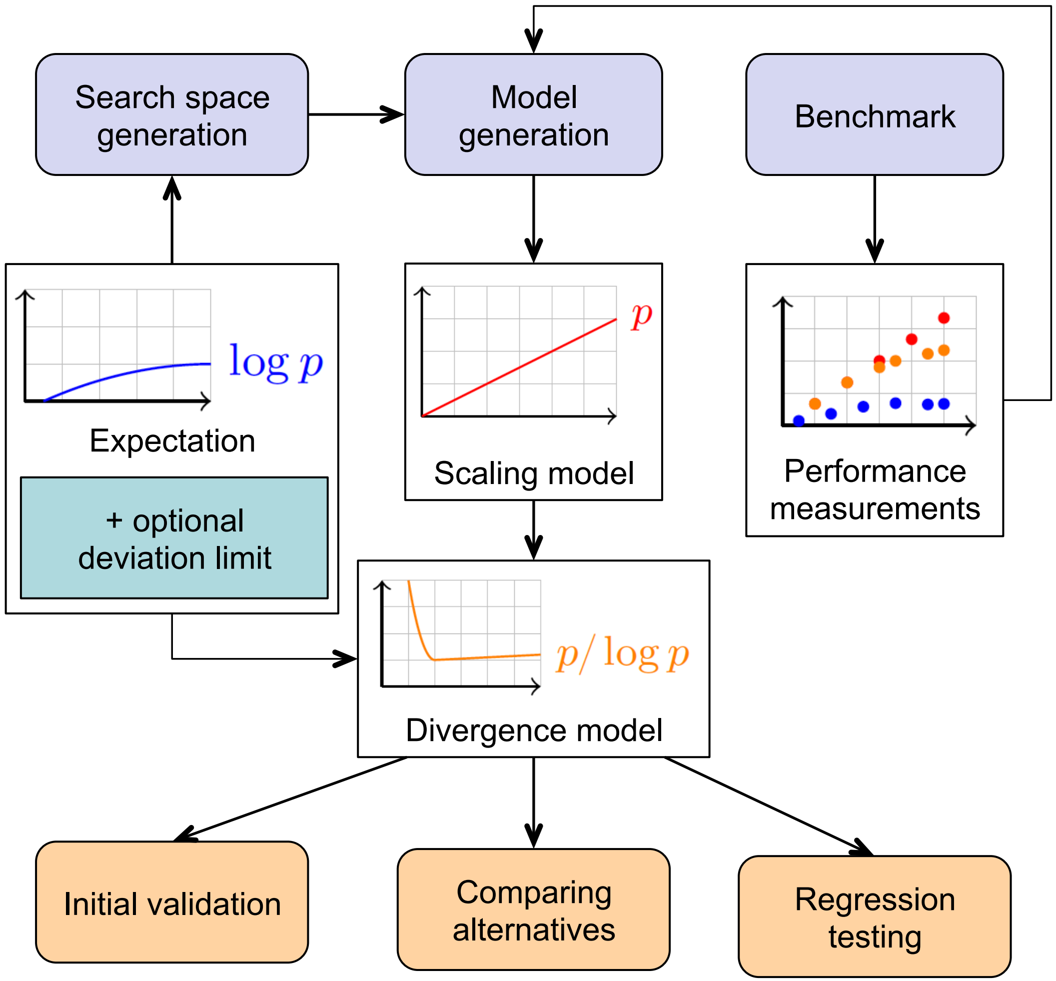
HPC libraries

- Focus on algorithms rather than applications
- Theoretical expectations more common
- Reuse factor makes scalability even more important



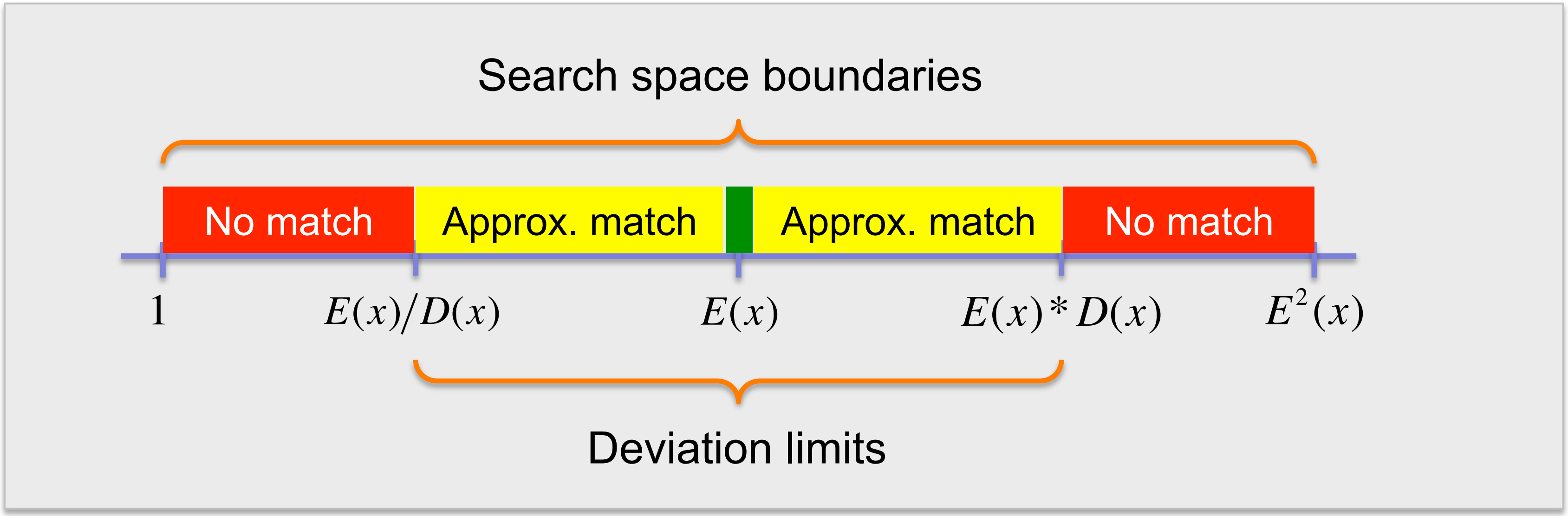
Example:
MPI communication library

Scalability evaluation framework



Customized search space

- Constructed around expectation
- Supports wider range of model functions than original PMNF



MPI

Platform	Juqueen	Juropa	Piz Daint
Allreduce [s]		**Expectation: *O* (log *p*)**	
Model	$O(\log p)$	$O(p^{0.5})$	$O(p^{0.67} \log p)$
R^2	0.87	0.99	0.99
Divergence	0	$O(p^{0.5}/\log p)$	$O(p^{0.67})$
Match	✓	~	✗ !
Comm_dup [B]		**Expectation: *O* (1)**	
Model	2.2e5	256	$3770 + 18p$
R^2	1	1	0.99
Divergence	$O(1)$	$O(1)$	$O(p)$
Match	✓	✓	✗

Platform	Daint
Barrier [s]	(log *p*)
Model	og *p*)
R^2	
Divergence	)
Match	
Bcast [s]	on: *O* (*p*)
Model	)
R^2	)
Divergence	)
Match	
Reduce [s]	on: *O* (*p*)
Model	)
R^2	)
Divergence	)
Match	

MAFIA

TECHNISCHE UNIVERSITÄT DARMSTADT

Sub-space clustering code used in data-mining

- Cluster dimensionality ***k*** is the model parameter
- Result: observed behavior matched the expectations

	gen	dedup	pcount	unjoin
Expectation	$O\ (k^3 2^k)$	$O\ (k^4 2^k)$	$O\ (k 2^k)$	$O\ (k^3 2^k)$
Model	$O\ (k^4 2^k)$	$O\ (k^4 2^k)$	$O\ (k 2^k)$	$O\ (k^2 2^k)$
Divergence	$O\ (k)$	$O\ (1)$	$O\ (1)$	$O\ (1/k)$
Match	~	✓	✓	~

Mass-producing performance models

TECHNISCHE UNIVERSITÄT DARMSTADT

- Is feasible
- Offers insight
- Requires low effort
- Improves code coverage

A. Vogel, A. Calotoiu, A. Strube, S. Reiter, A. Nägel, F. Wolf, G. Wittum: 10,000 performance models per minute - scalability of the UG4 simulation framework. In Proc. of the Euro-Par Conference, Vienna, Austria, August 2015

Euro-Par 2015

S. Shudler, A. Calotoiu, T. Hoefler, A. Strube, F. Wolf: Exascaling Your Library: Will Your Implementation Meet Your Expectations?. In *Proc. of the International Conference on Supercomputing (ICS), Newport*

ICS 2015

A. Calotoiu, T. Hoefler, M. Poke, F. 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, pages 1-12, ACM, November 2013.

Thank you!

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Deutsche
Forschungsgemeinschaft
DFG