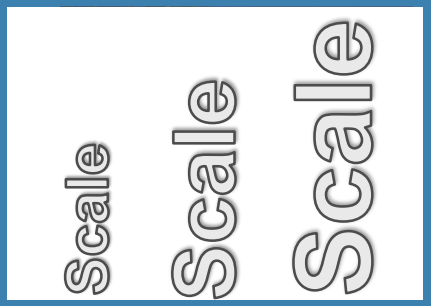
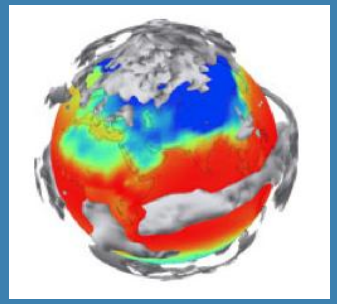
German Research School for Simulation Sciences

Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes



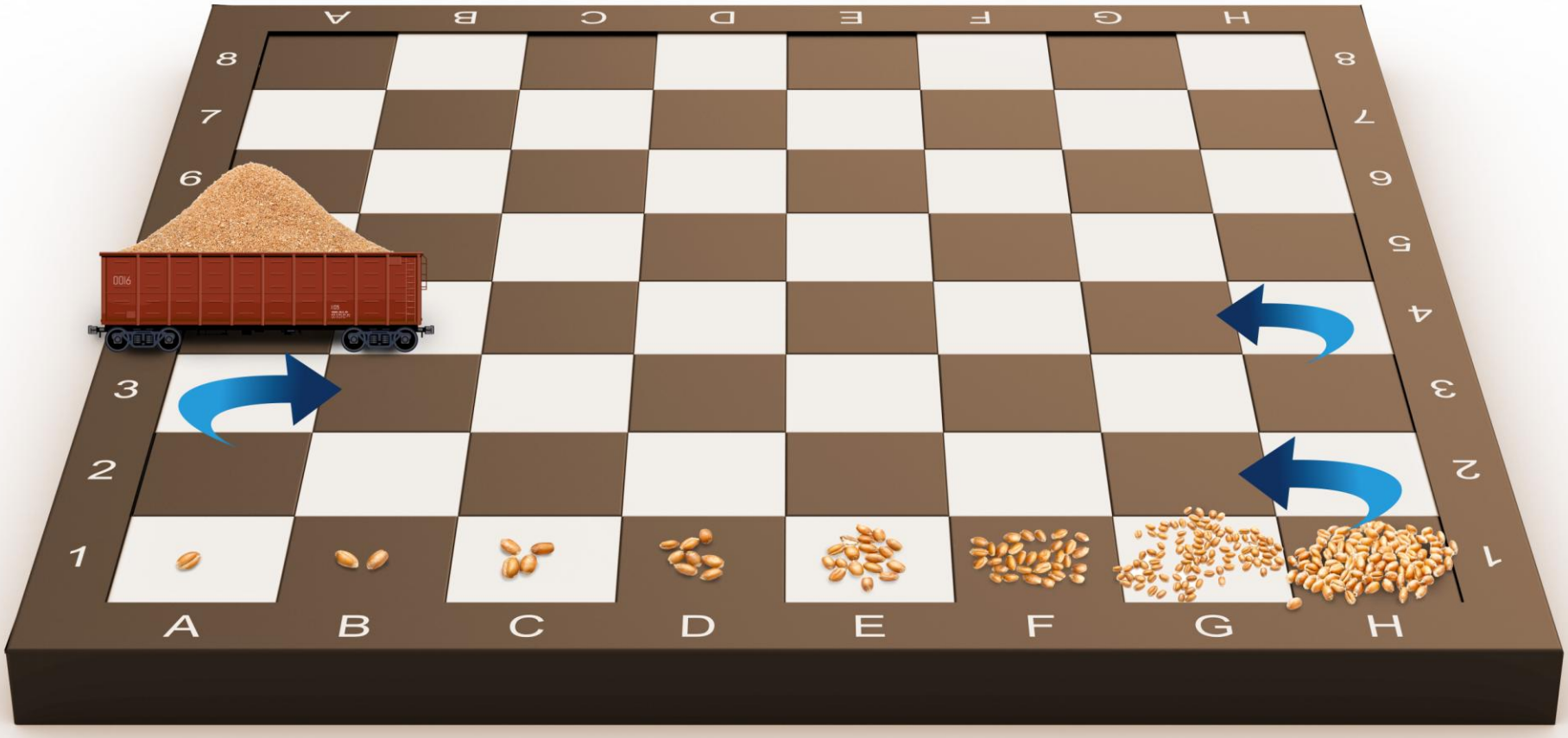
A. Calotoiu[1], T. Hoefler[2], M. Poke[1], F. Wolf[1]

1) German Research School for Simulation Sciences
2) ETH Zurich

September 13, 2013

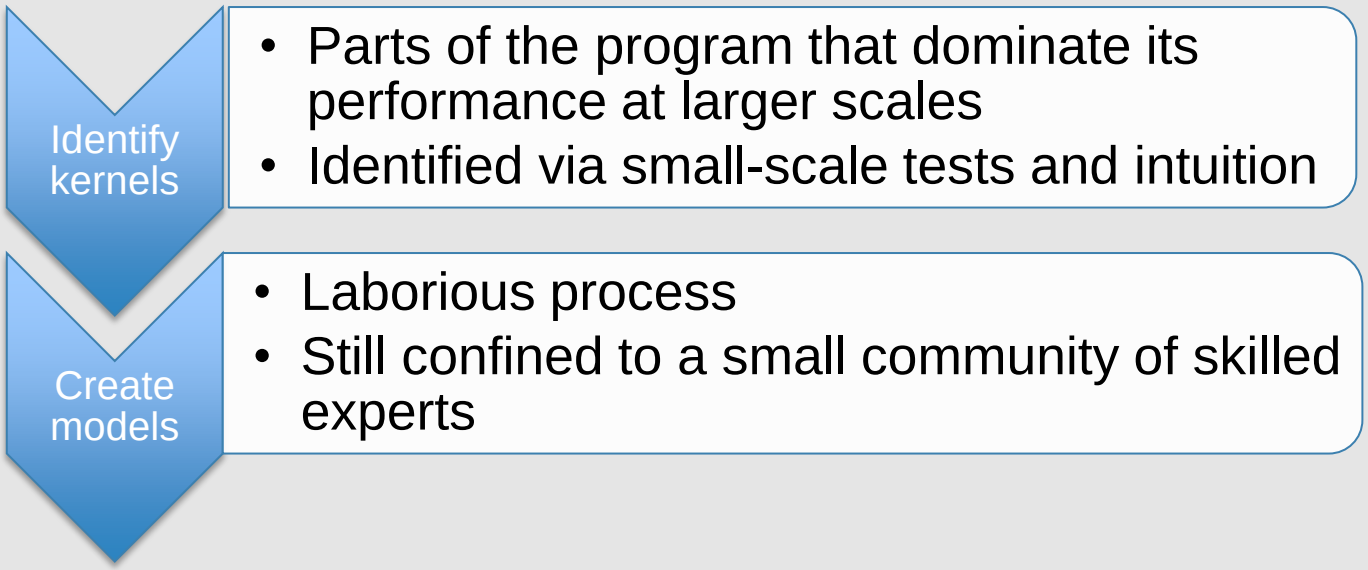
JÜLICH FORSCHUNGSZENTRUM

RWTH AACHEN UNIVERSITY


A B C D E F G H
8 7 6 5 4 3 2 1

German Research School for Simulation Sciences

Analytical performance modeling



Disadvantages

- Time consuming
- Danger of overlooking unscalable code

German Research School
for Simulation Sciences

Our approach

Generate an empirical model for each part of the program automatically

- Run a manageable number of small-scale performance experiments
- Launch our tool
- Compare extrapolated performance to expectations

Key ideas

- Exploit that space of function classes underlying such model is small enough to be searched by a computer program
- Abandon model accuracy as the primary success metric and rather focus on the binary notion of scalability bugs
- Create requirements models alongside execution models

German Research School for Simulation Sciences

SPPEXA project Catwalk

- German Research School for Simulation Sciences, Laboratory for Parallel Programming (Prof. Dr. Felix Wolf)
- Technische Universität Darmstadt, Institute for Scientific Computing (Prof. Dr. Christian Bischof)

- Swiss Federal Institute of Technology Zurich, Institute of Computer Systems, (Prof. Dr. Torsten Hoefler)

- Forschungszentrum Jülich, Jülich Supercomputing Centre (Dr.-Ing. Bernd Mohr)

- Goethe University Frankfurt, Goethe Center for Scientific Computing (Prof. Dr. Gabriel Wittum)

DFG FNSNF

German Research School
for Simulation Sciences

Outline

Overview
Detailed approach
Evaluation
Conclusion

German Research School for Simulation Sciences

Scalability bug detector

Input
- Set of performance measurements (profiles) on different processor counts $\{p_1, \ldots, p_{max}\}$ w/ weak scaling
- Individual measurement broken down by program region (call path)

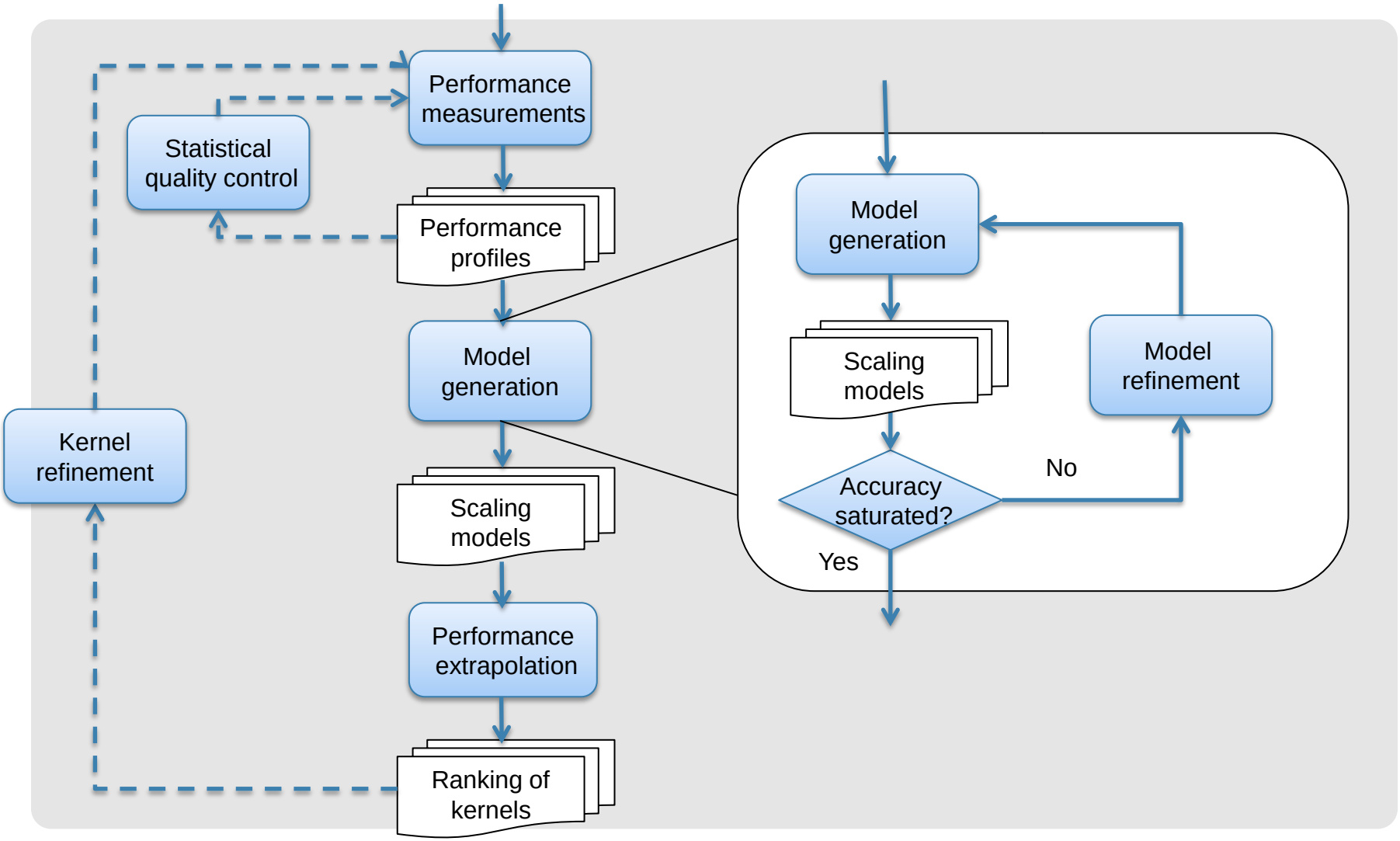
Output
- List of program regions (kernels) ranked by their predicted execution time at target scale $p_t > p_{max}$
- Or ranked by growth function ($p_t \rightarrow \infty$)

- Not 100% accurate but good enough to draw attention to right kernels
- False negatives when phenomenon at scale is not captured in data
- False positives possible but unlikely

- Can also model parameters other than p

German Research School
for Simulation Sciences

Workflow



German Research School for Simulation Sciences

Model generation

Performance Model Normal Form (PMNF)

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

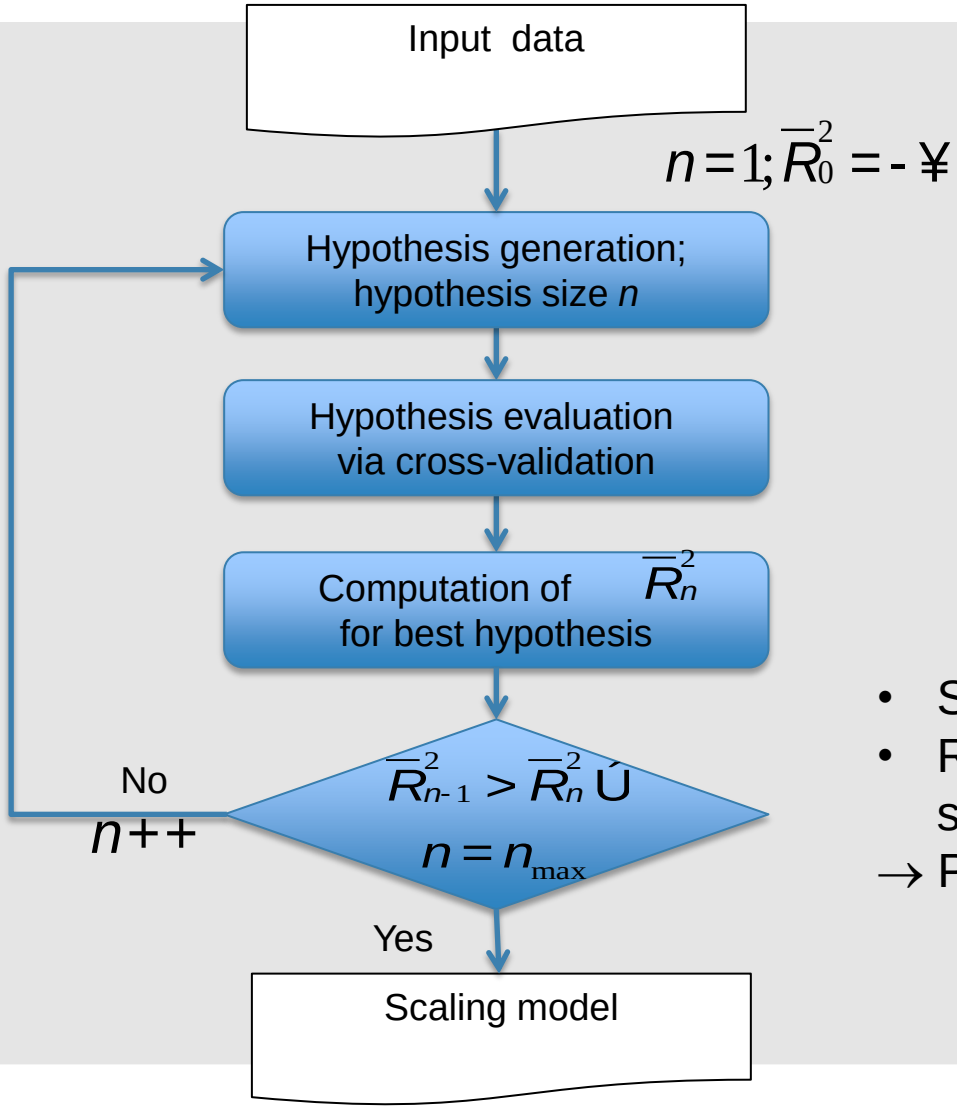
- Not exhaustive but works in most practical scenarios
- An assignment of n, i_k and j_k is called model hypothesis
- i_k and j_k are chosen from sets $I,J \subset \mathbf{Q}$
- n, $|I|$, $|J|$ don't have to be arbitrarily large to achieve good fit

Instead of deriving model through reasoning, make reasonable choices for n, I, J and try all assignment options one by one

- Select winner through **cross-validation**

German Research School for Simulation Sciences

Model refinement

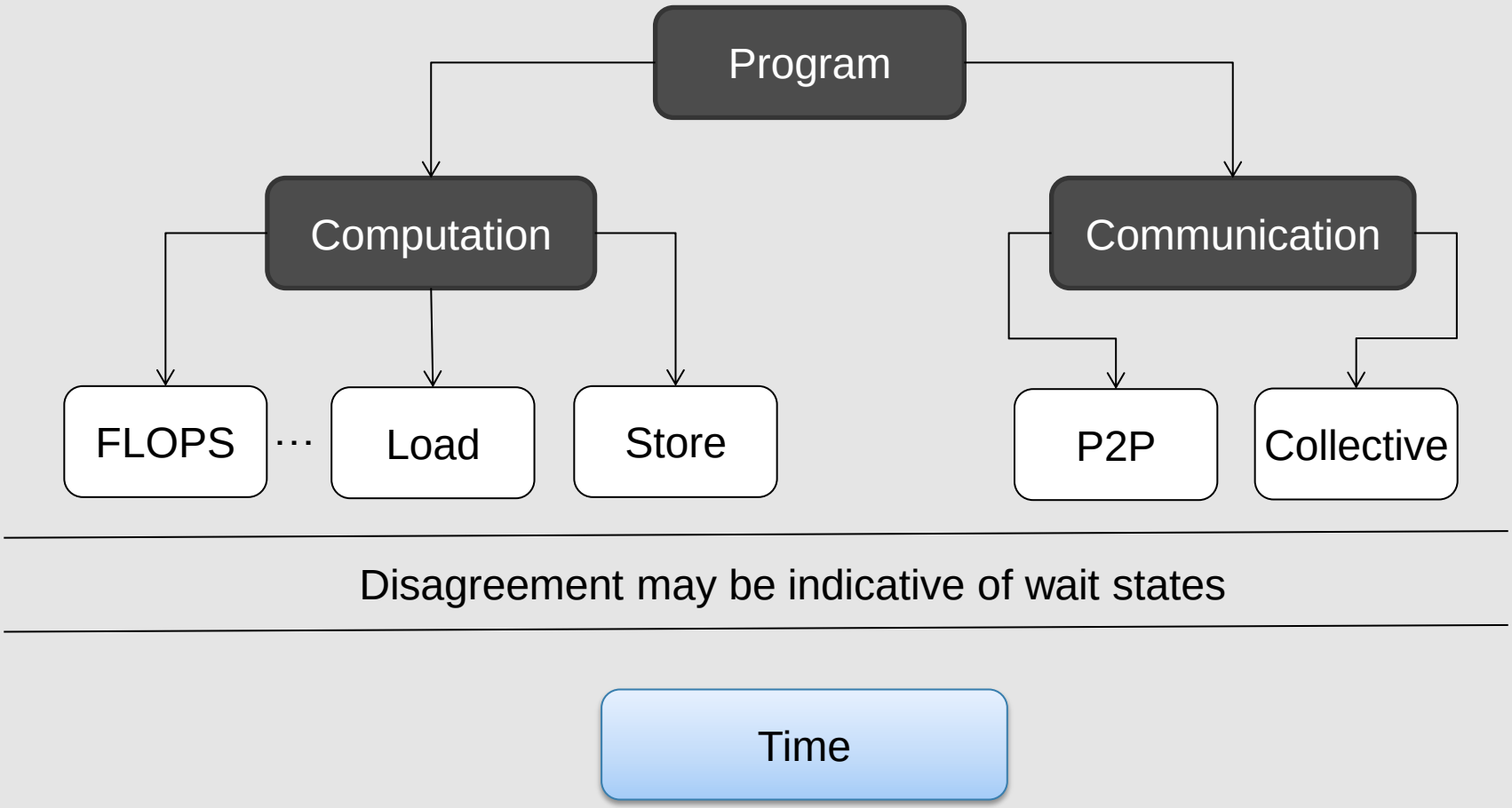


- Start with coarse approximation
- Refine to the point of statistical shrinkage

→ Protection against over-fitting

German Research School for Simulation Sciences

Requirements modeling



German Research School
for Simulation Sciences

Evaluation

We demonstrate that our tool

- identifies a scalability issue in a code that is known to have one
- does not identify a scalability issue in a code that is known to have none
- identifies two scalability issues in a code that was thought to have only one

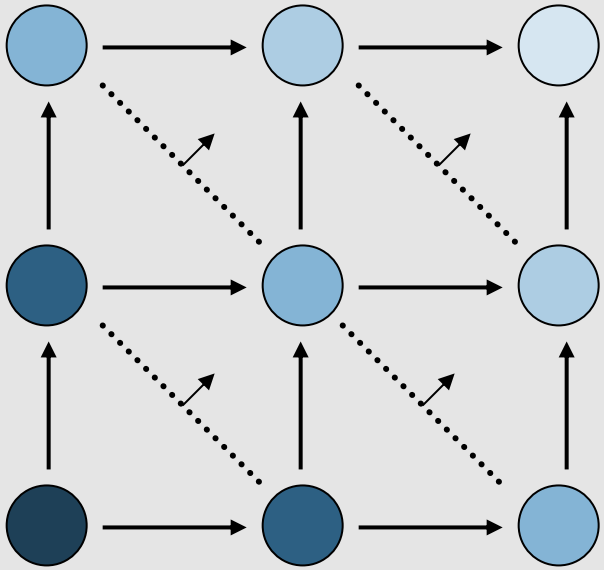
Test platform:
IBM Blue Gene/Q
Juqueen in Jülich

$$I = \{\tfrac{0}{2}, \tfrac{1}{2}, \tfrac{2}{2}, \tfrac{3}{2}, \tfrac{4}{2}, \tfrac{5}{2}, \tfrac{6}{2}\}$$
$$J = \{0,1,2\}$$
$$n = 5$$

Sweep3D

Solves neutron transport problem

- 3D domain mapped onto 2D process grid
- Parallelism achieved through pipelined wave-front process



LogGP model for communication developed by Hoisie et al.

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

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

German Research School for Simulation Sciences

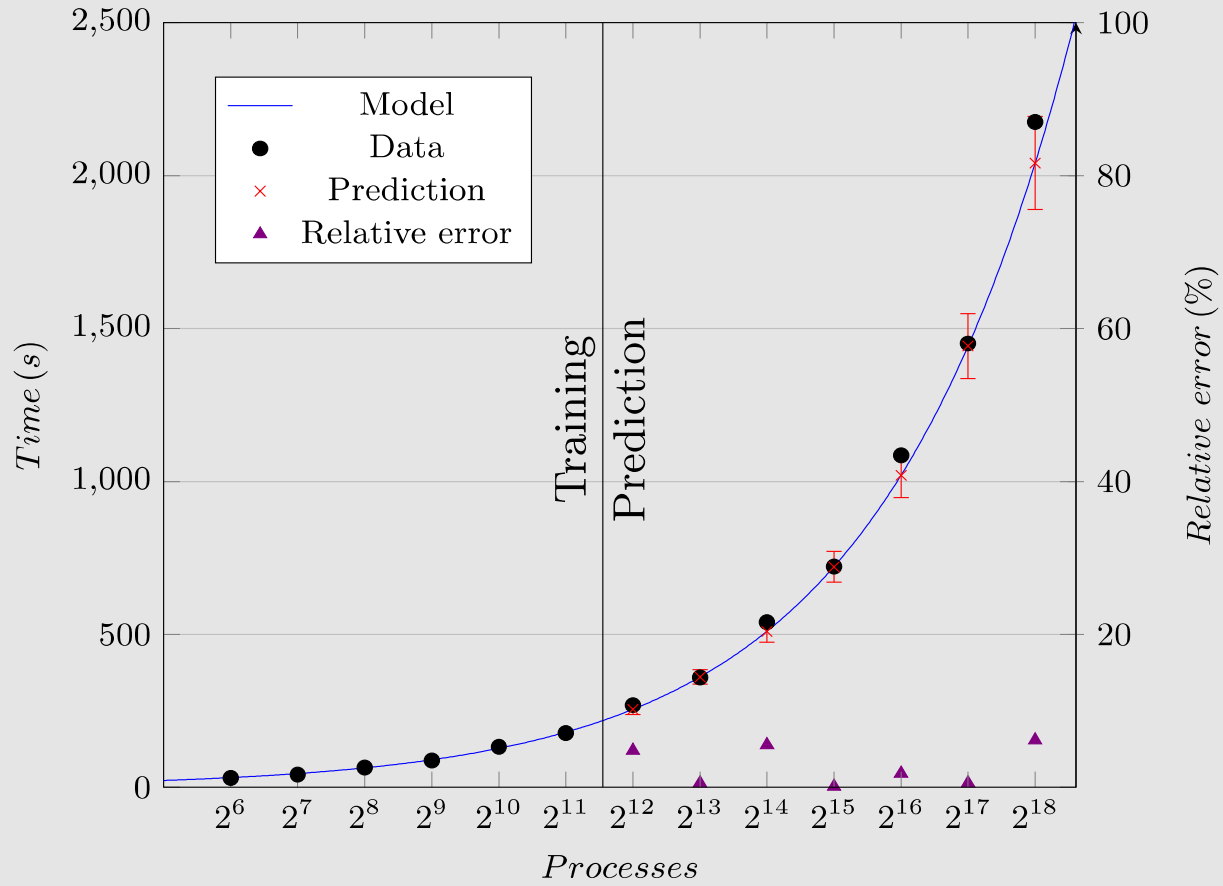
Sweep3D (2)

Kernel	Runtime[%] p_t=262k	Increase $\frac{t(p=262k)}{t(p=64)}$	Model [s] t = f(p)	Predictive error [%] p_t=262k
sweep->MPI_Recv	65.35	16.54	$4.03\sqrt{p}$	5.10
sweep	20.87	0.23	582.19	0.01
global_int_sum-> MPI_Allreduce	12.89	18.68	$1.06\sqrt{p}+ 0.03\sqrt{p}*\log(p)$	13.60
sweep->MPI_Send	0.40	0.23	$11.49+0.09\sqrt{p}*\log(p)$	15.40
source	0.25	0.04	$6.86+9.13*10^{-5}\log(p)$	0.01

$p_i \leq 8k$

German Research School
for Simulation Sciences

Sweep3D (3)



MILC

MILC/su3_rmd – code from MILC suite of QCD codes with performance model manually created by Hoefler et al.

- Time per process should remain constant except for a rather small logarithmic term caused by global convergence checks

Kernel	Model [s] t=f(p)	Predictive Error [%] p_t=64k
compute_gen_staple_field	$2.40*10^{-2}$	0.43
g_vecdoublesum>MPI_Allreduce	$6.30*10^{-6}*\log^2 p$	0.01
mult_adj_su3_fieldlink_lathwec	$3.80*10^{-3}$	0.04

$p_i \leq 16k$

German Research School for Simulation Sciences

MILC – Varying grid points per process

Kernel	Flops Model[s] flops=f(V)	$\overline{R^2}$ [*10^{-3}]	Visits Model visits=f(V)	$\overline{R^2}$ [*10^{-3}]	Flops/Visit Model	$\overline{R^2}$ [*10^{-3}]
load_lnglinks	$5.64*10^4*V$	0.030	$2.31*10^3$	0.000	24.42*V	0.030
load_fatlinks_cpu	$1.95*10^6*V$	0.210	$7.14*10^4$	0.000	27.36*V	0.210
ks_congrad	$1.16*10^8+ 3.24*10^5*V^{5/4}$	0.292	$5.11*10^4+ 1.38*10^4*V^{1/4}$	4.000	15.94*V	0.143
imp_gauge_force_cpu	$1.65*10^6*V$	0.015	$7.40*10^4$	0.000	22.28*V	0.015
eo_fermion_force_two_terms_site	$4.02*10^6*V$	0.002	$1.27*10^5$	0.000	31.61*V	0.002

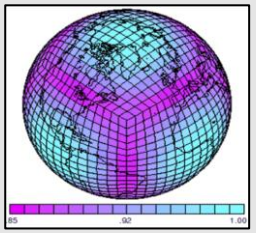
Test platform: Juropa in Jülich (Intel Nehalem cluster)
p = 32= constant

German Research School for Simulation Sciences

HOMME

Core of the Community Atmospheric Model (CAM)

- Spectral element dynamical core on a cubed sphere grid



Kernel	$p_i \leq 15k$ Model [s] t = f(p)	$p_i \leq 15k$ Predictive error [%] $p_t = 130k$	$p_i \leq 43k$ Model [s] t = f(p)	$p_i \leq 43k$ Predictive error [%] $p_t = 130k$
Box_rearrange->MPI_Reduce	$0.03+2.53*10^{-6}p*\sqrt{p}+1.24*10^{-12}p^3$	57.02	$3.63*10^{-6}p*\sqrt{p}+7.21*10^{-13}p^3$	30.34
Vlaplace_sphere_vk	49.53	99.32	$24.44+2.26*10^{-7}p^2$	4.28
…				
Compute_and_apply_rhs	48.68	1.65	49.09	0.83

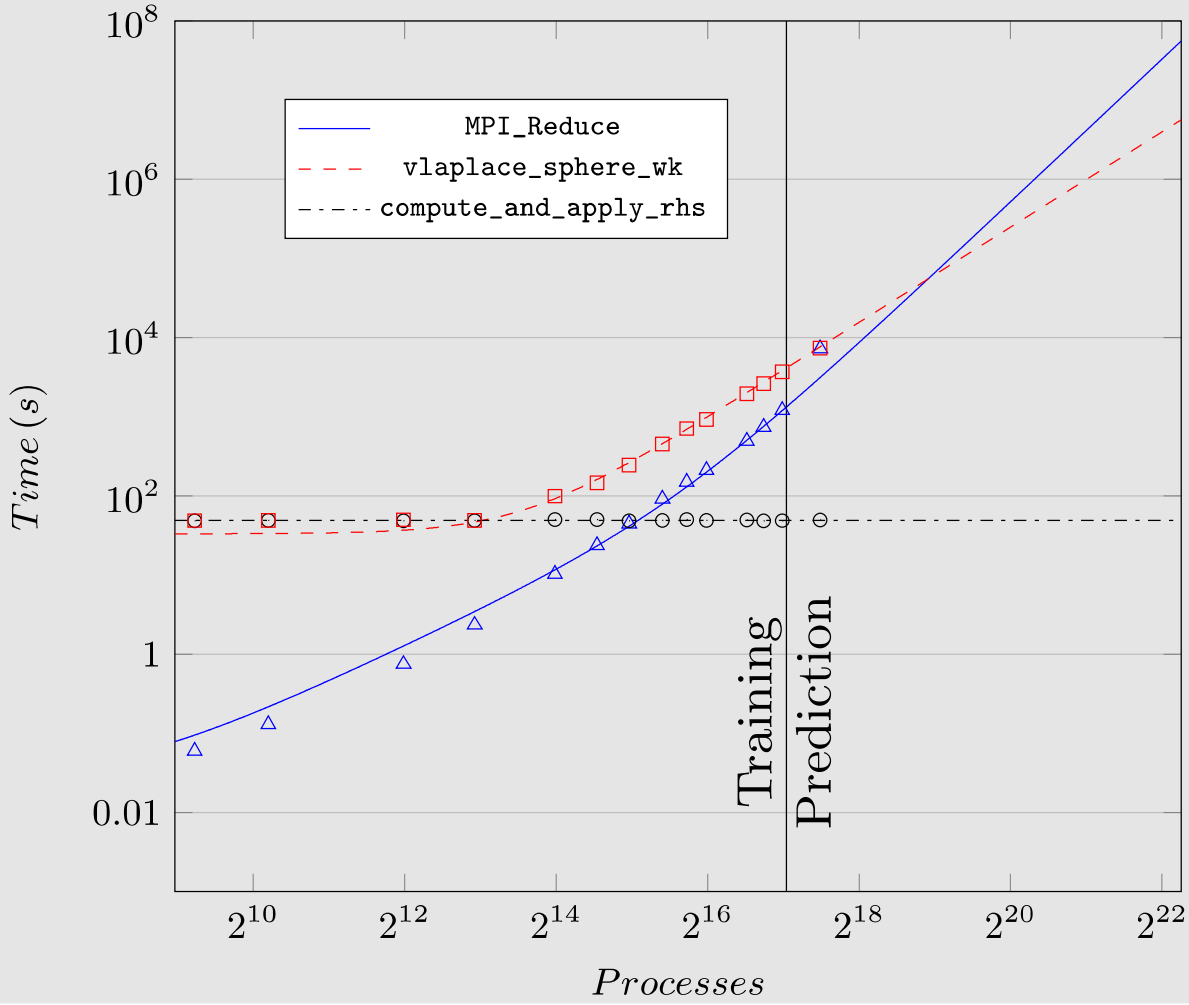
HOMME (2)

Two issues

Number of iterations inside a subroutine grows with p^2
- Ceiling for up to and including 15k
- Developers were aware of this issue and had developed work-around

Growth of time spent in reduce function grows with p^3
- Previously unknown
- Function invoked during initialization to funnel data to dedicated I/O processes
- Execution time at 183k ~ 2h, predictive error ~40%

HOMME (3)


MPI_Reduce
vlaplace_sphere_wk
compute_and_apply_rhs
Training
Prediction
Time (s)
Processes
10^8
10^6
10^4
10^2
1
0.01
2^10
2^12
2^14
2^16
2^18
2^20
2^22

German Research School for Simulation Sciences

Conclusion

Automated performance modeling is feasible

Generated models accurate enough to identify scalability bugs and in good agreement with hand-crafted models

Advantages of mass production also performance models

- Approximate models are acceptable as long as the effort to create them is low and they do not mislead the user
- Code coverage is as important as model accuracy

Future work

- Study influence of further hardware parameters
- More efficient traversal of search space (allows increase of modeling parameters)
- Integration into Scalasca