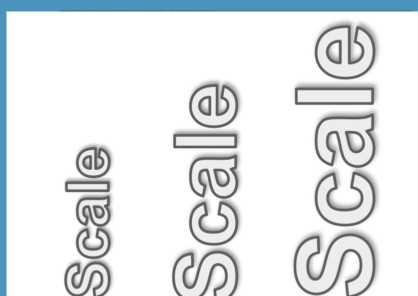
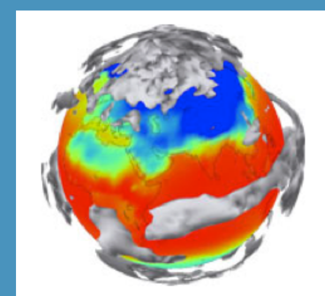
German Research School
for Simulation Sciences

Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes



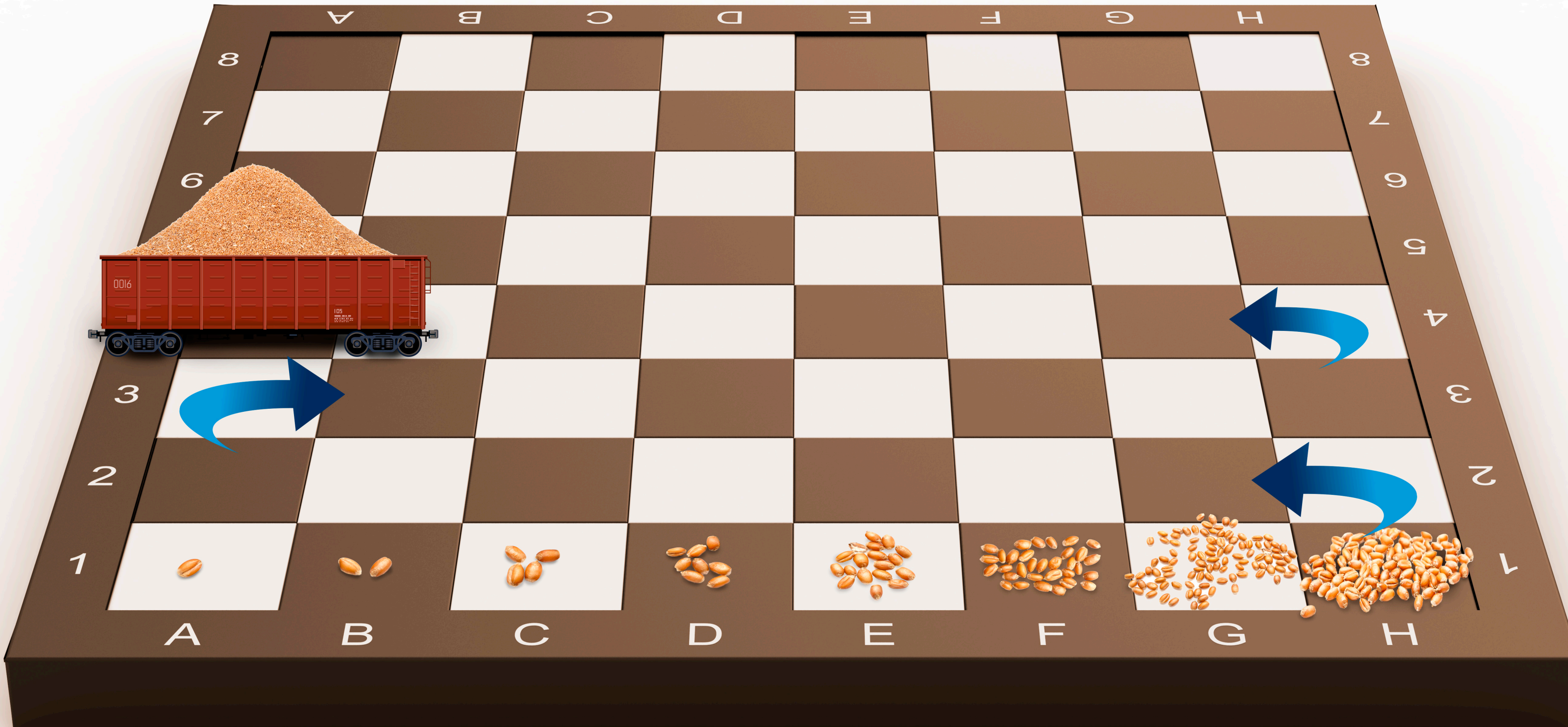


A. Calotoiu[1], T. Hoefler[2], M. Poke[1], F. Wolf[1]

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

July 15, 2013

JÜLICH FORSCHUNGSZENTRUM
RWTH AACHEN UNIVERSITY


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

Analytical performance 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

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-time models

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)

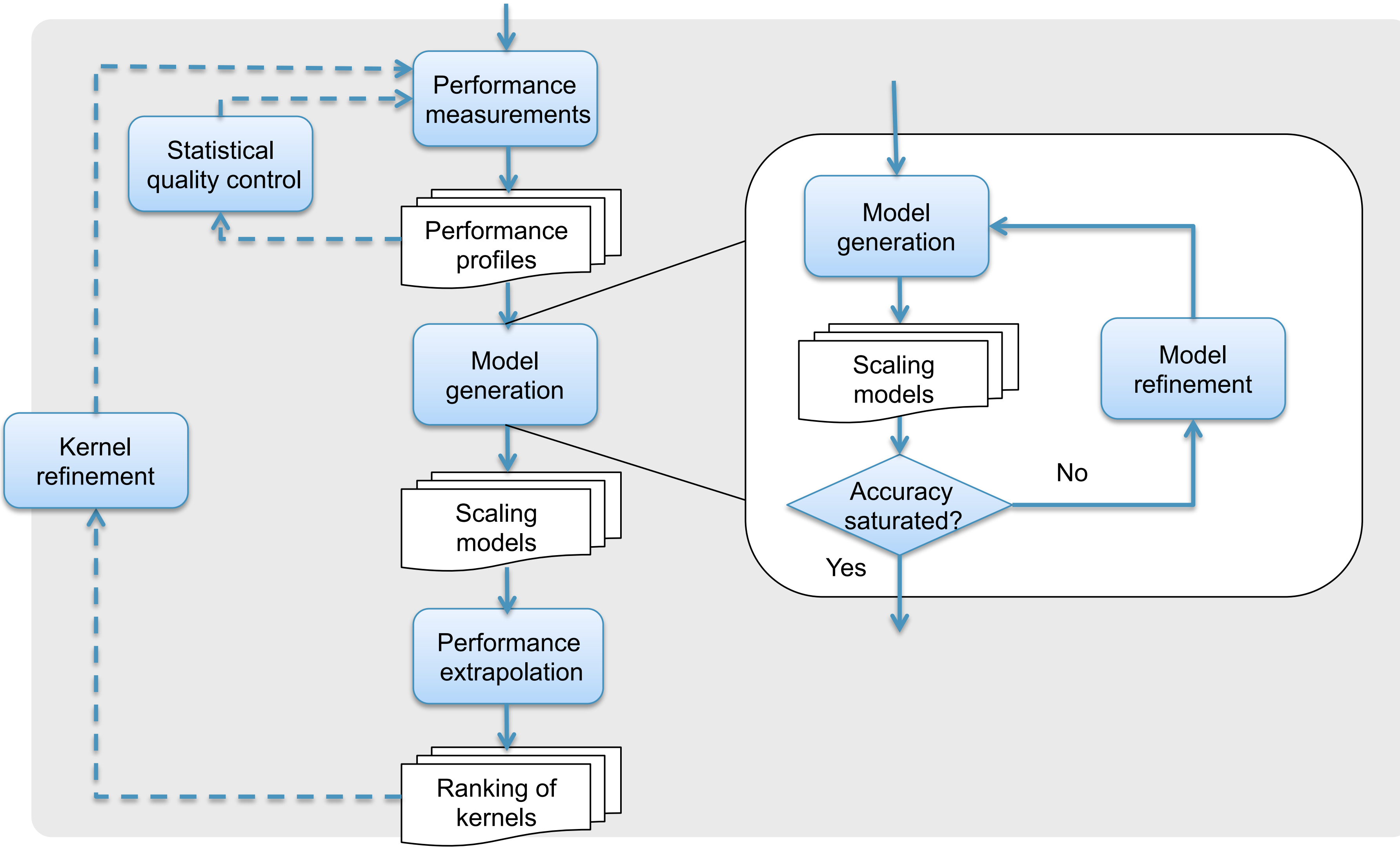
Output

- 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

Workflow


Performance measurements
Statistical quality control
Performance profiles
Model generation
Scaling models
Performance extrapolation
Ranking of kernels
Kernel refinement
Model generation
Scaling models
Accuracy saturated?
No
Yes
Model refinement

Model generation

Performance Model Normal Form (PMNF)

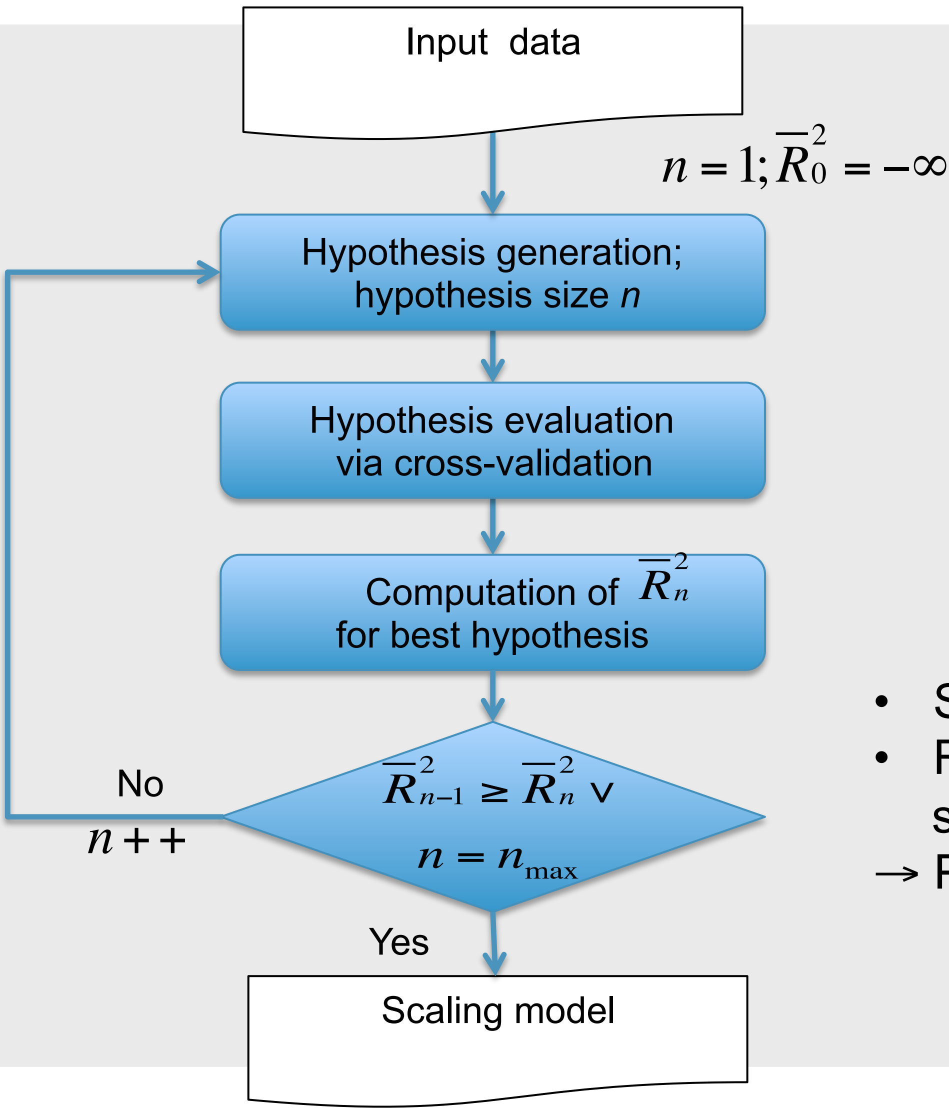
$$f(p) = \sum_{k=1}^{n} c_k \cdot p^{i_k} \cdot \log_2^{j_k}(p)$$

- 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**

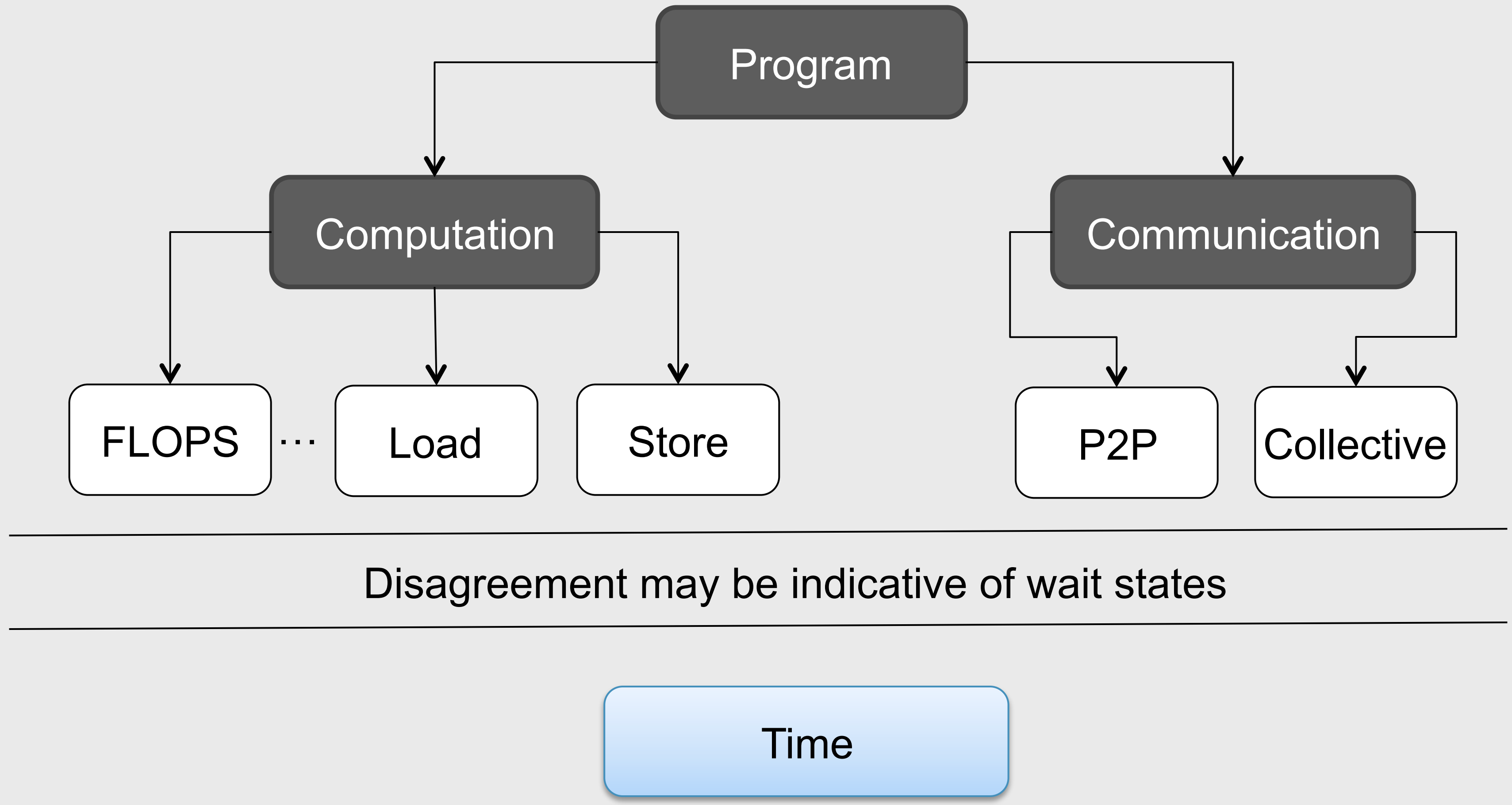
Model refinement



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

→ Protection against over-fitting

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

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

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

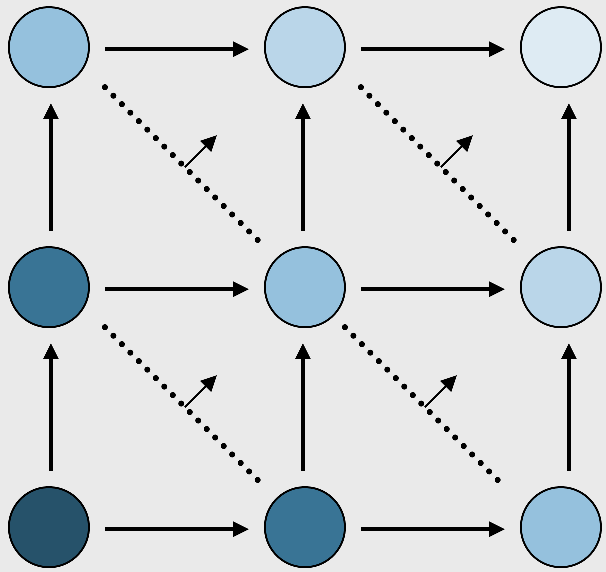
$$n = 5$$

German Research School for Simulation Sciences

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.

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

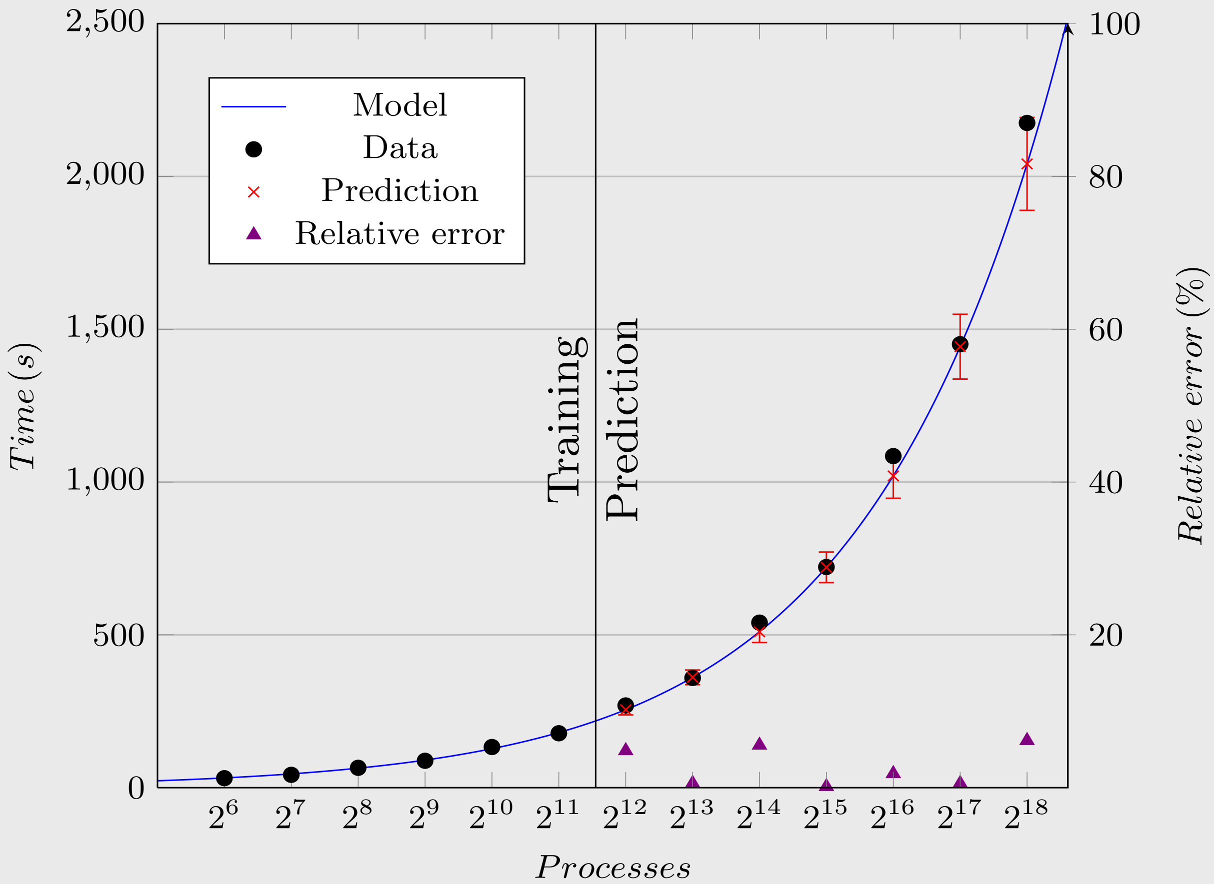
$$t^{comm} = c \cdot \sqrt{p}$$

Sweep3D (2)

Kernel	Runtime[%] p_t=262k	Increase $\frac{t(p=262k)}{t(p=64)}$	Model [s] $t = f(p)$ $p_i \leq 8k$	Predictive error [%] p_t=262k
sweep → MPI_Recv	65.4	16.5	$4.0\sqrt{p}$	5.1
sweep	20.9	0.2	582.2	0.01
global_int_sum → MPI_Allreduce	12.9	18.7	$1.1\sqrt{p} + 0.03\sqrt{p}\bullet\log(p)$	13.6
sweep → MPI_Send	0.4	0.2	$11.5 + 0.1\sqrt{p}\bullet\log(p)$	15.4
source	0.3	0.04	$6.7 + 9.1\bullet10^{-5}\log(p)$	0.01

#bytes = const.
#msg = const.

Sweep3D (3)



Model
Data
Prediction
Relative error
Training
Prediction
Time (s)
Relative error (%)
Processes
0
500
1,000
1,500
2,000
2,500
20
40
60
80
100
2^{6} 2^{7} 2^{8} 2^{9} 2^{10} 2^{11} 2^{12} 2^{13} 2^{14} 2^{15} 2^{16} 2^{17} 2^{18}

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

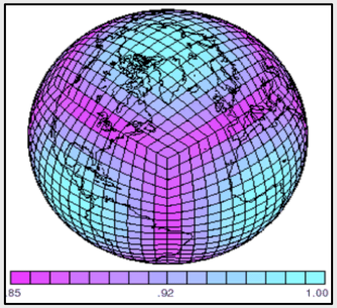
Kernel	Model [s] $t=f(p)$ $p_i \leq 16k$	Predictive Error [%] p_t=64k
compute_gen_staple_field	0.02	0.4
g_vecdoublesum → MPI_Allreduce	$6.3 \cdot 10^{-6} \cdot \log^2 p$	0.01
mult_adj_su3_fieldlink_lathwec	0.004	0.04

German Research School for Simulation Sciences

HOMME

Core of the Community Atmospheric Model (CAM)

- Spectral element dynamical core on a cubed sphere grid

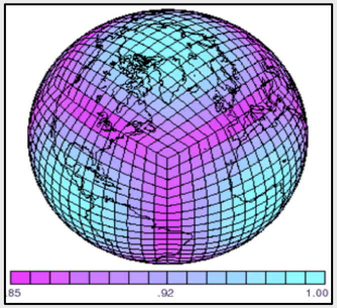


Kernel	Model [s] $t = f(p)$ $p_i \leq 15k$	Predictive error [%] $p_t = 130k$
Box_rearrange->MPI_Reduce	$0.03 + 2.5 \cdot 10^{-6} p^{3/2} + 1.2 \cdot 10^{-12} p^3$	57.0
Vlaplace_sphere_vk	49.5	99.3
…		
Compute_and_apply_rhs	48.7	1.7

HOMME

German Research School
for Simulation Sciences

Core of the Community Atmospheric Model (CAM)
- Spectral element dynamical core
  on a cubed sphere grid



Kernel	Model [s] $t = f(p)$ $p_i \leq 43k$	Predictive error [%] $p_t = 130k$
Box_rearrange->MPI_Reduce	$3.6 \cdot 10^{-6} p^{3/2} + 7.2 \cdot 10^{-13} p^3$	30.3
Vlaplace_sphere_vk	$24.4 + 2.3 \cdot 10^{-7} p^2$	4.3
…		
Compute_and_apply_rhs	49.1	0.8

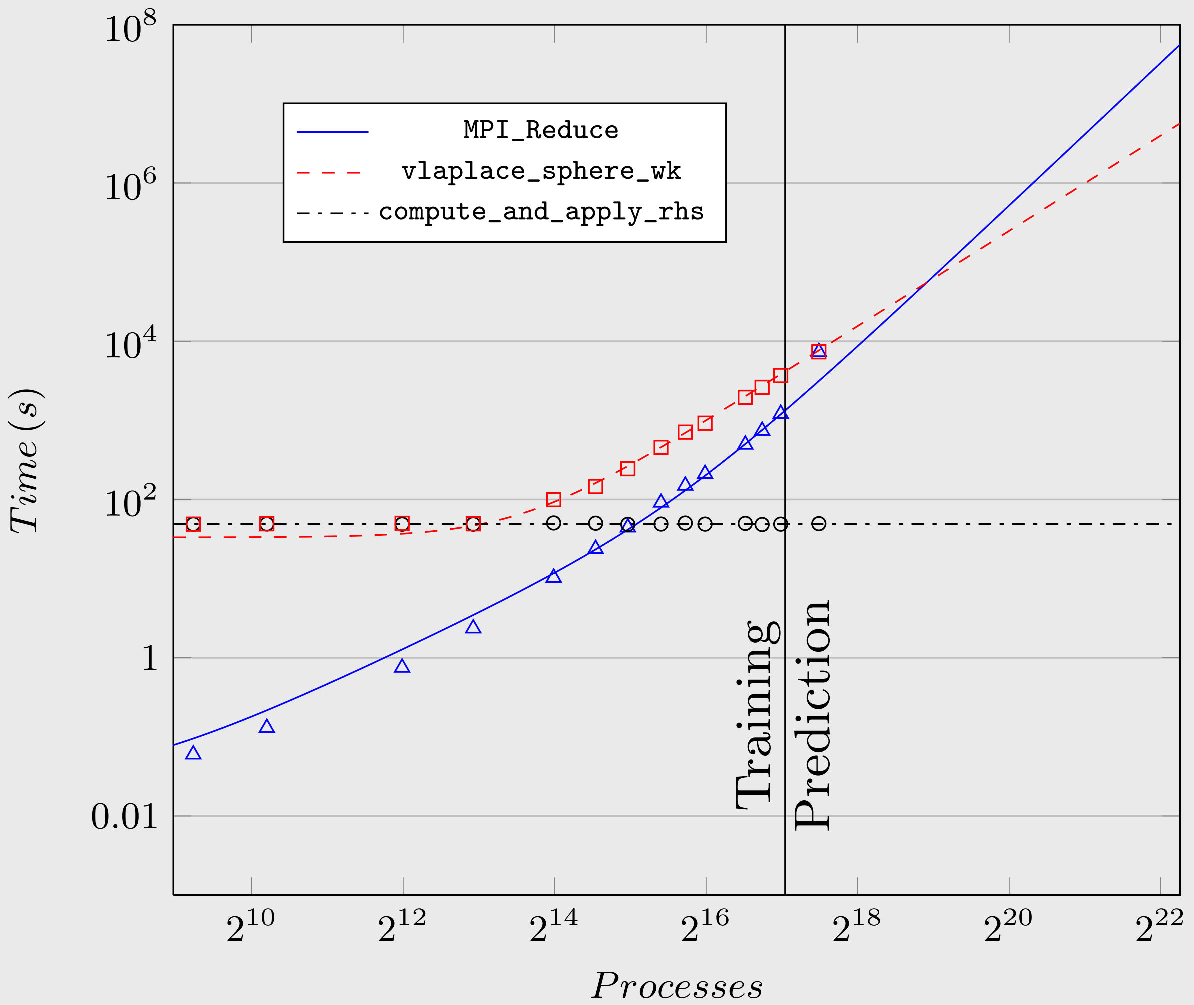
HOME (2)

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%

HOME (3)



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

Conclusion

Automated performance modeling is feasible

Generated models accurate enough to identify scalability bugs or show their absence with high probability

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 more model parameters)
- Integration into Scalasca

German Research School
for Simulation Sciences

Acknowledgement

DFG

SPPEXA

FNSNF