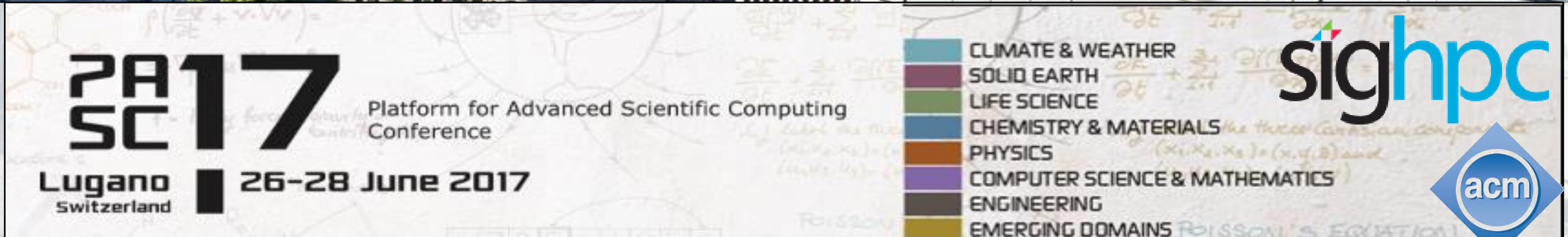
TORSTEN HOEFLER, ROBERTO BELLI

Scientific Benchmarking of Parallel Computing Systems

Twelve ways to tell the masses when reporting performance results

Performance Engineering for HPC Workshop at ICS'17, Frankfurt, Germany



Disclaimer(s)

- **This is an experience talk (published at SC 15 – State of the Practice)!**
  - Explained in SC15 FAQ:

    "generalizable insights as gained from experiences with particular HPC machines/operations/applications/benchmarks, overall analysis of the status quo of a particular metric of the entire field or historical reviews of the progress of the field."
  - Don't expect novel insights

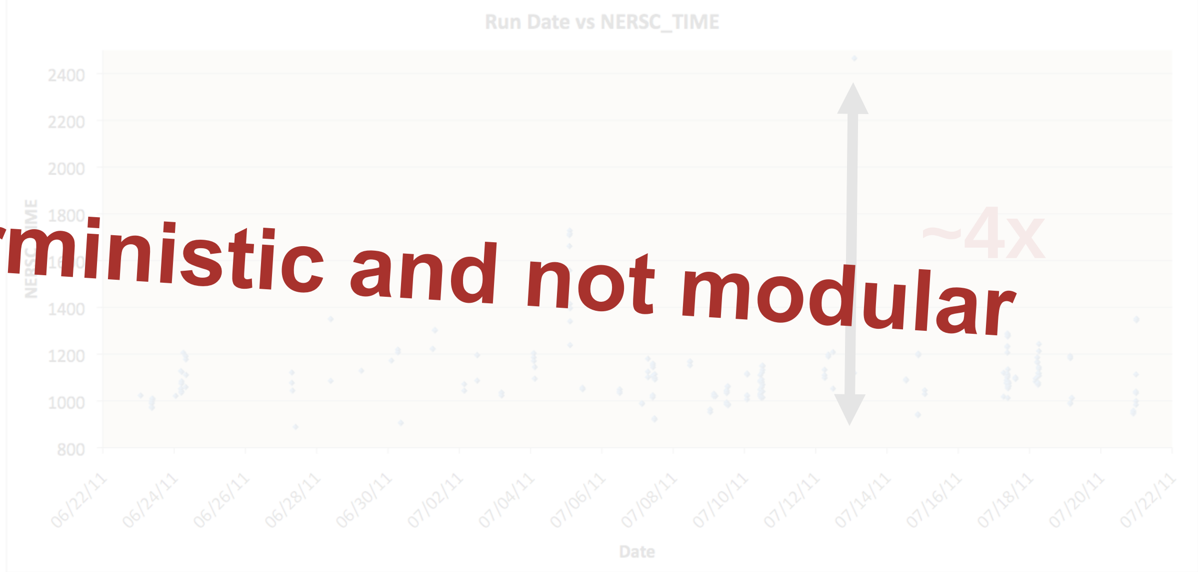
    Given the papers I read, much of what I say may be new for many

- **My musings shall not offend anybody**
  - Everything is (now) anonymized

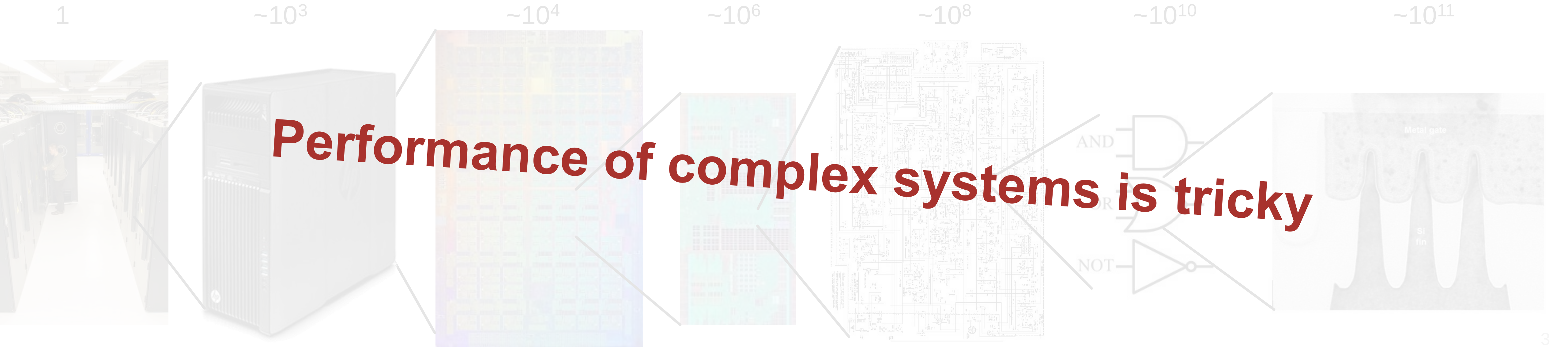
- **Criticism may be rhetorically exaggerated**
  - Watch for tropes!

- **This talk should be entertaining!**

High Performance Computing

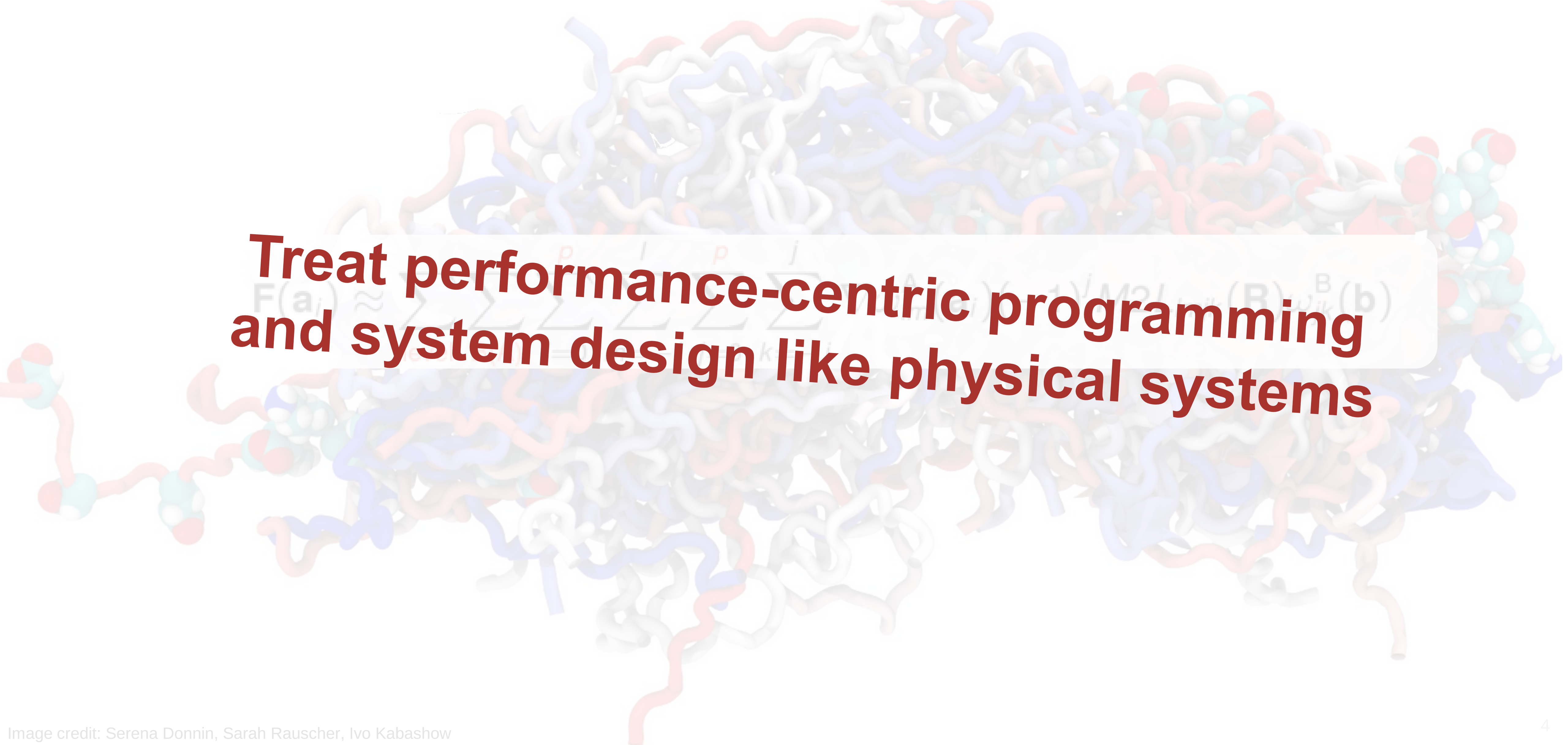


Performance is nondeterministic and not modular



Performance of complex systems is tricky

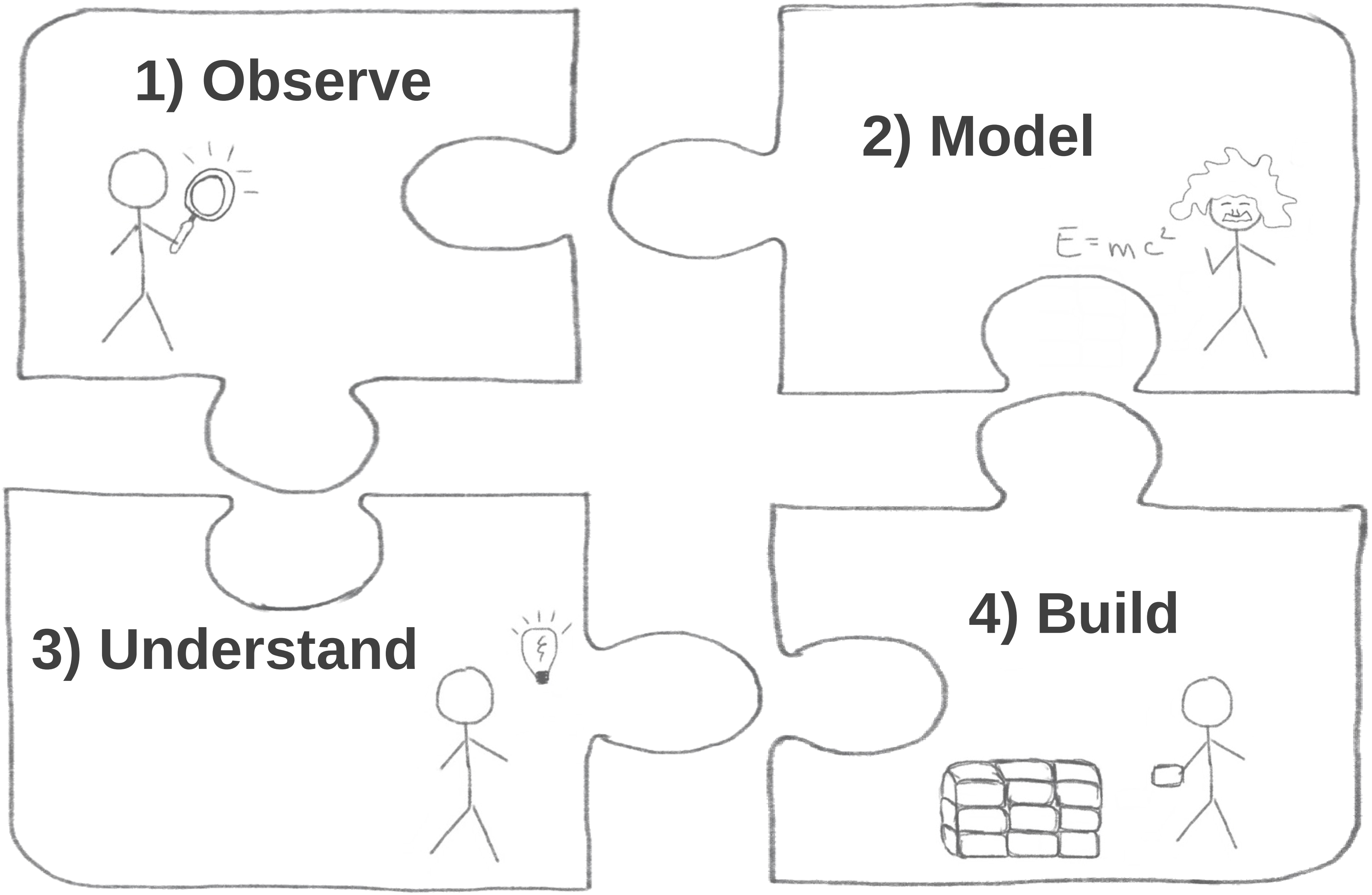
HPC is used to solve complex problems!



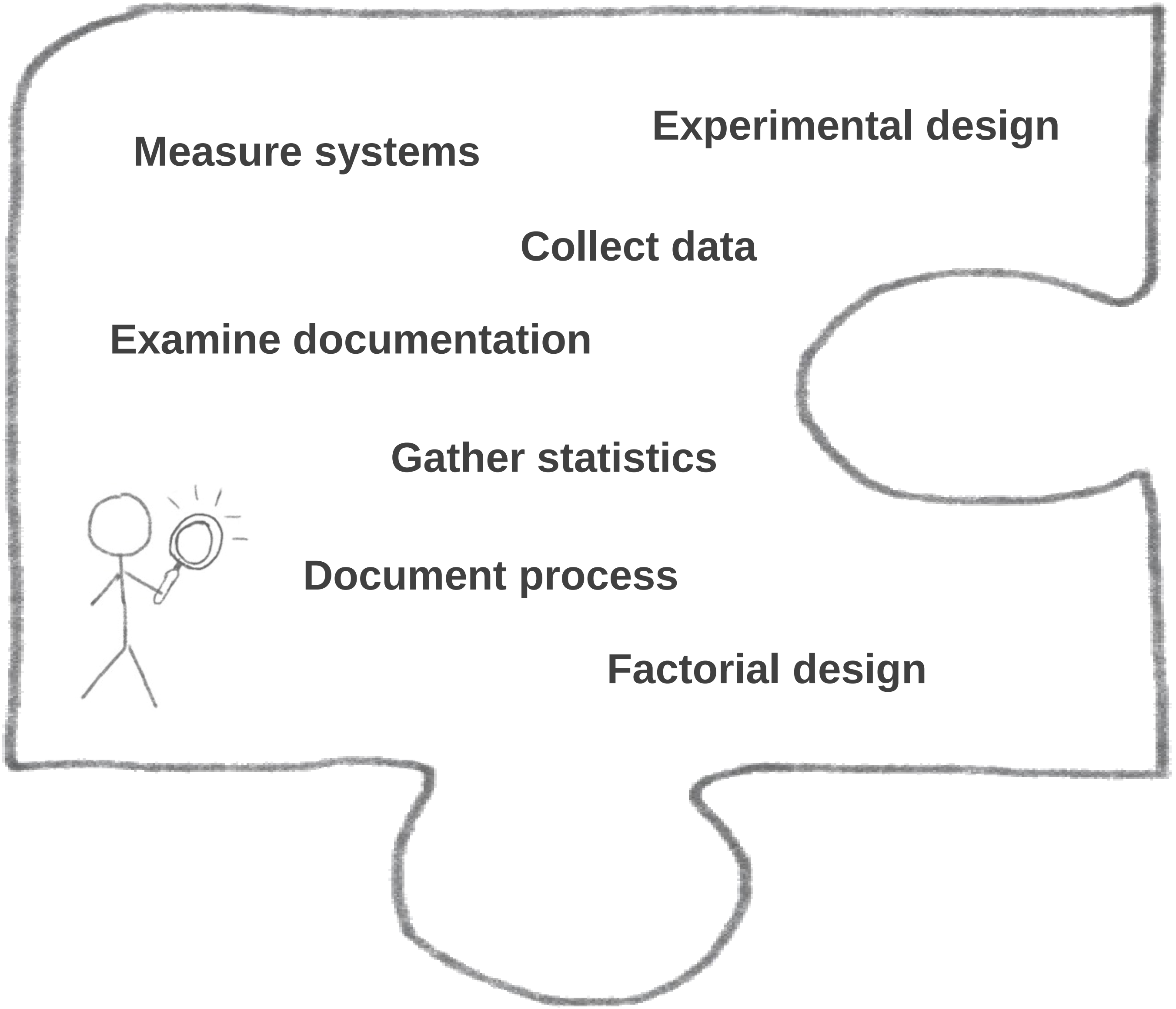
Treat performance-centric programming and system design like physical systems

Image credit: Serena Donnin, Sarah Rauscher, Ivo Kabashow

Scientific Performance Engineering



Part I: Observe

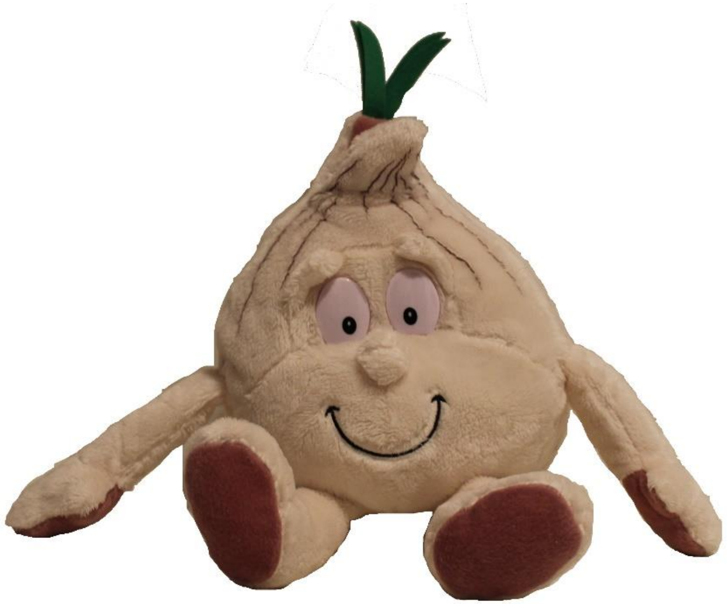


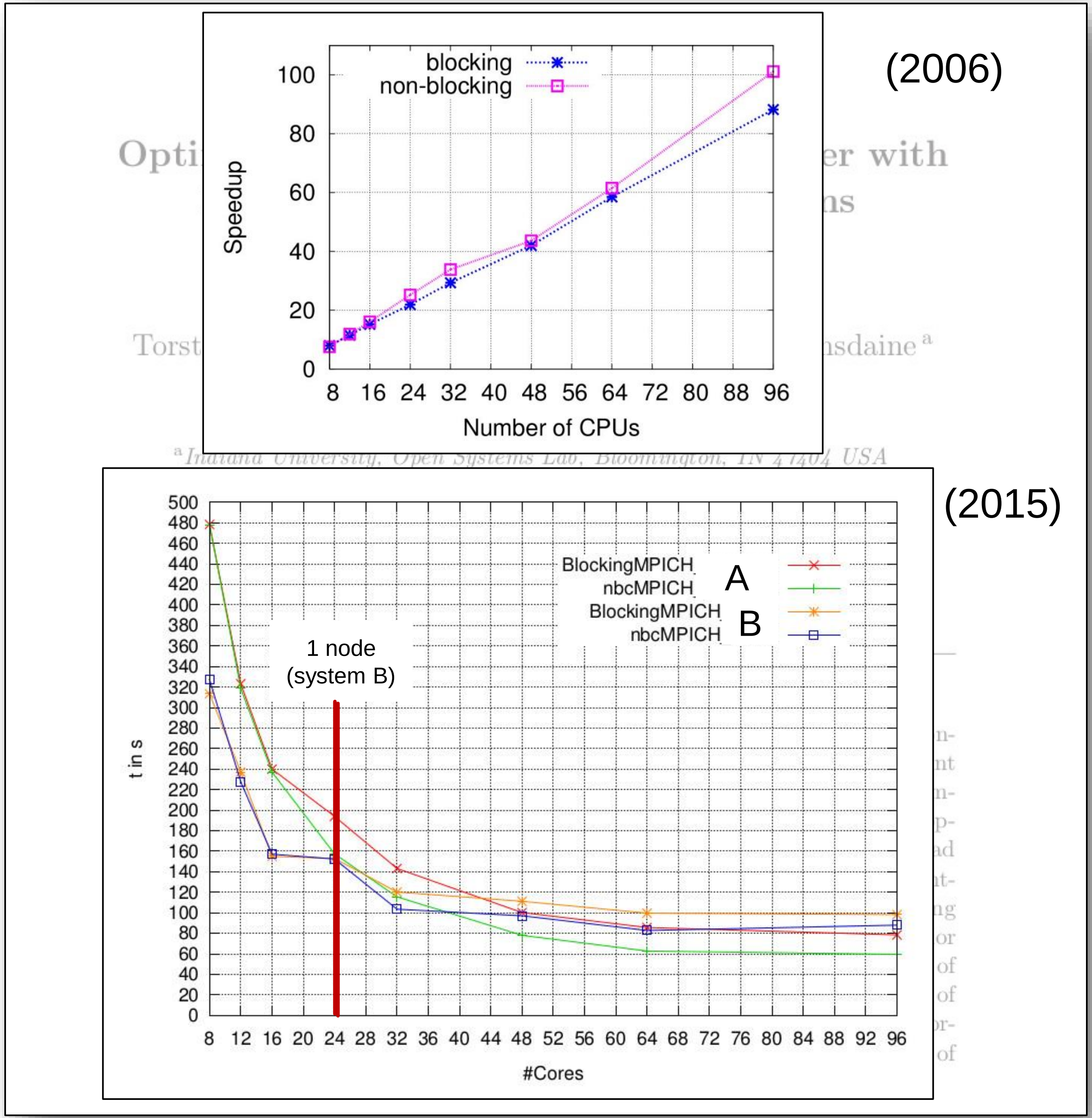
How does Garth measure and report performance?

- **We may be interested in High Performance Computing**
  - We (want to) see it as a science – reproducing experiments is a major pillar of the scientific method

- **When measuring performance, important questions are**
  - "How many iterations do I have to run per measurement?"
  - "How many measurements should I run?"
  - "Once I have all data, how do I summarize it into a single number?"
  - "How do I compare the performance of different systems?"
  - "How do I measure time in a parallel system?"
  - …

- **How are they answered in the field today?**
  - Let me start with a little anecdote … a reaction to this paper ☺





- **Original findings:**
  - If carefully tuned, NBC speed up a 3D solver

    Full code published
  - 800^3 domain – 4 GB (distributed) array

    1 process per node, 8-96 nodes

    Opteron 246 (old even in 2006, retired now)
  - Super-linear speedup for 96 nodes

    ~5% better than linear

- **9 years later: attempt to reproduce ☺!**

  System A: 28 quad-core nodes, Xeon E5520

  System B: 4 nodes, dual Opteron 6274

  "Neither the experiment in A nor the one in B could reproduce the results presented in the original paper, where the usage of the NBC library resulted in a performance gain for practically all node counts, reaching a superlinear speedup for 96 cores (explained as being due to cache effects in the inner part of the matrix vector product)."

State of the Practice in HPC

- **Stratified random sample of three top-conferences over four years**
  - HPDC, PPoPP, SC (years: 2011, 2012, 2013, 2014)
  - 10 random papers from each (10-50% of population)
  - 120 total papers, 20% (25) did not report performance (were excluded)

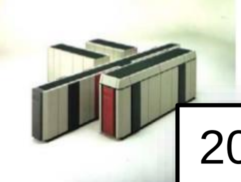
- **Main results:**
  1. Most papers report details about the hardware but fail to describe the software environment.
     Important details for reproducibility missing
  2. The average paper's results are hard to interpret and easy to question
     Measurements and data not well explained
  3. No statistically significant evidence for improvement over the years ☹

- **Our main thesis:**
  Performance results are often nearly impossible to reproduce! Thus, we need to provide enough information to allow scientists to understand the experiment, draw own conclusions, assess their certainty, and possibly generalize results.

This is especially important for HPC conferences and activities such as the Gordon Bell award!

Well, we all know this - but do we really know how to fix it?

1991 – the classic!



Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers

Abstract

Many of us
quite diffic
supercompu
scientific pa
these result

2012 – the shocking

How did *this* get published?

Pitfal

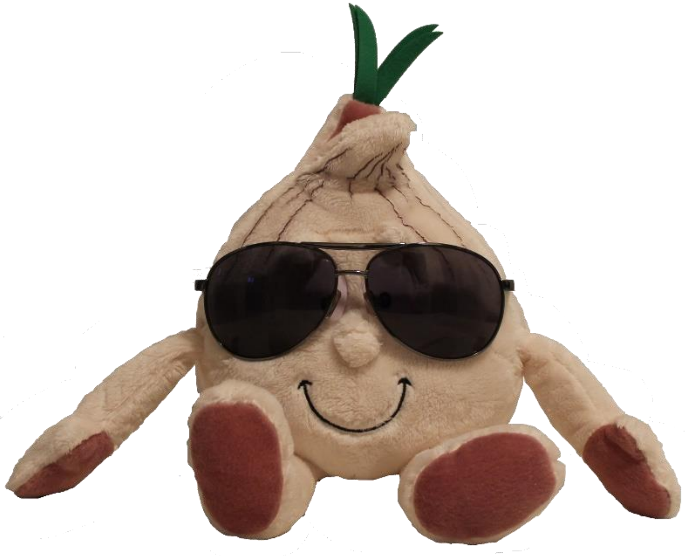
2013 – the extension

Fooling the Masses with Performance Results: Old Classics & Some New Ideas

Gerhard Wellein[1,2], Georg Hager[2]

[1]Department for Computer Science
[2]Erlangen Regional Computing Center
Friedrich-Alexander-Universität Erlangen-Nürnberg

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT



Yes, this is a garlic press!

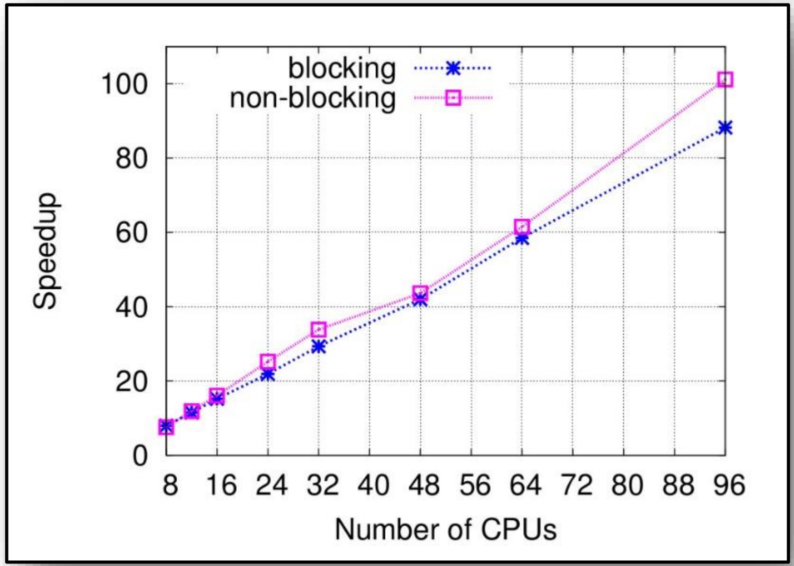
…

Our constructive approach: provide a set of (12) rules

- **Attempt to emphasize interpretability of performance experiments**
- **The set is not complete**
  - And probably never will be
  - Intended to serve as a solid start
  - Call to the community to extend it

- **I will illustrate the 12 rules now**
  - Using real-world examples
    All anonymized!
  - Garth and Eddie will represent the bad/good scientist

The most common issue: speedup plots



- **Most common and oldest-known issue**
  - First seen 1988 – also included in Bailey’s 12 ways
  - 39 papers reported speedups

    15 (38%) did not specify the base-performance ☹
  - Recently rediscovered in the “big data” universe

    A. Rowstron et al.: Nobody ever got fired for using Hadoop on a cluster, HotCDP 2012

    F. McSherry et al.: Scalability! but at what cost?, HotOS 2015

The most common issue: speedup plots

Check out my wonderful Speedup!

I can't tell if this is useful at all!

Rule 1: *When publishing parallel speedup, report if the base case is a single parallel process or best serial execution, as well as the absolute execution performance of the base case.*

- **Most common and oldest-known issue**
  - First seen 1988 – also included in Bailey's 12 ways
  - 39 papers reported speedups

    15 (38%) did not specify the base-performance ☹
  - Recently rediscovered in the "big data" universe

    A. Rowstron et al.: Nobody ever got fired for using Hadoop on a cluster, HotCDP 2012

    F. McSherry et al.: Scalability! but at what cost?, HotOS 2015

- **A simple generalization of this rule implies that one should never report ratios without absolute values.**

Garth's new compiler optimization

Performance in Gflop/s

Rule 2: *Specify the reason for only reporting subsets of standard benchmarks or applications or not using all system resources.*

- **This implies: Show results even if your code/approach stops scaling!**

The mean parts of means – or how to summarize data

Rule 3: *Use the arithmetic mean only for summarizing costs. Use the harmonic mean for summarizing rates.*

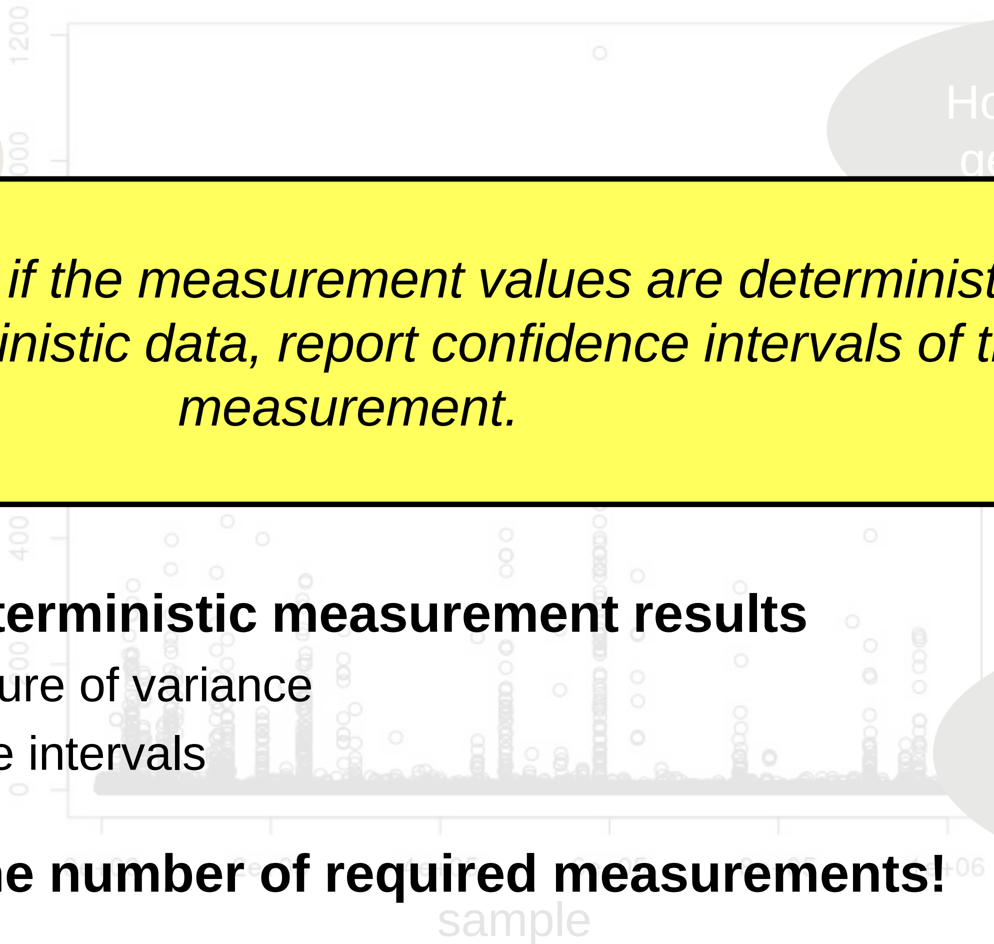
Rule 4: *Avoid summarizing ratios; summarize the costs or rates that the ratios base on instead. Only if these are not available use the geometric mean for summarizing ratios.*

- **51 papers use means to summarize data, only four (!) specify which mean was used**
  - **A single paper correctly specifies the use of the harmonic mean**
  - **Two use geometric means, without reason**
  - **Similar issues in other communities (PLDI, CGO, LCTES) – see N. Amaral's report**
- **harmonic mean ≤ geometric mean ≤ arithmetic mean**

Dealing with variation

The latency of Piz Dora is

How did you get to this?



Rule 5: *Report if the measurement values are deterministic. For nondeterministic data, report confidence intervals of the measurement.*

I averaged 10^6 tests, it must be right!

Why do you think so? Can I see the data?

- **Most papers report nondeterministic measurement results**
  - Only 15 mention some measure of variance
  - Only two (!) report confidence intervals
- **CIs allow us to compute the number of required measurements!**
- **Can be very simple, e.g., single sentence in evaluation:**

  “We collected measurements until the 99% confidence interval was within 5% of our reported means.”

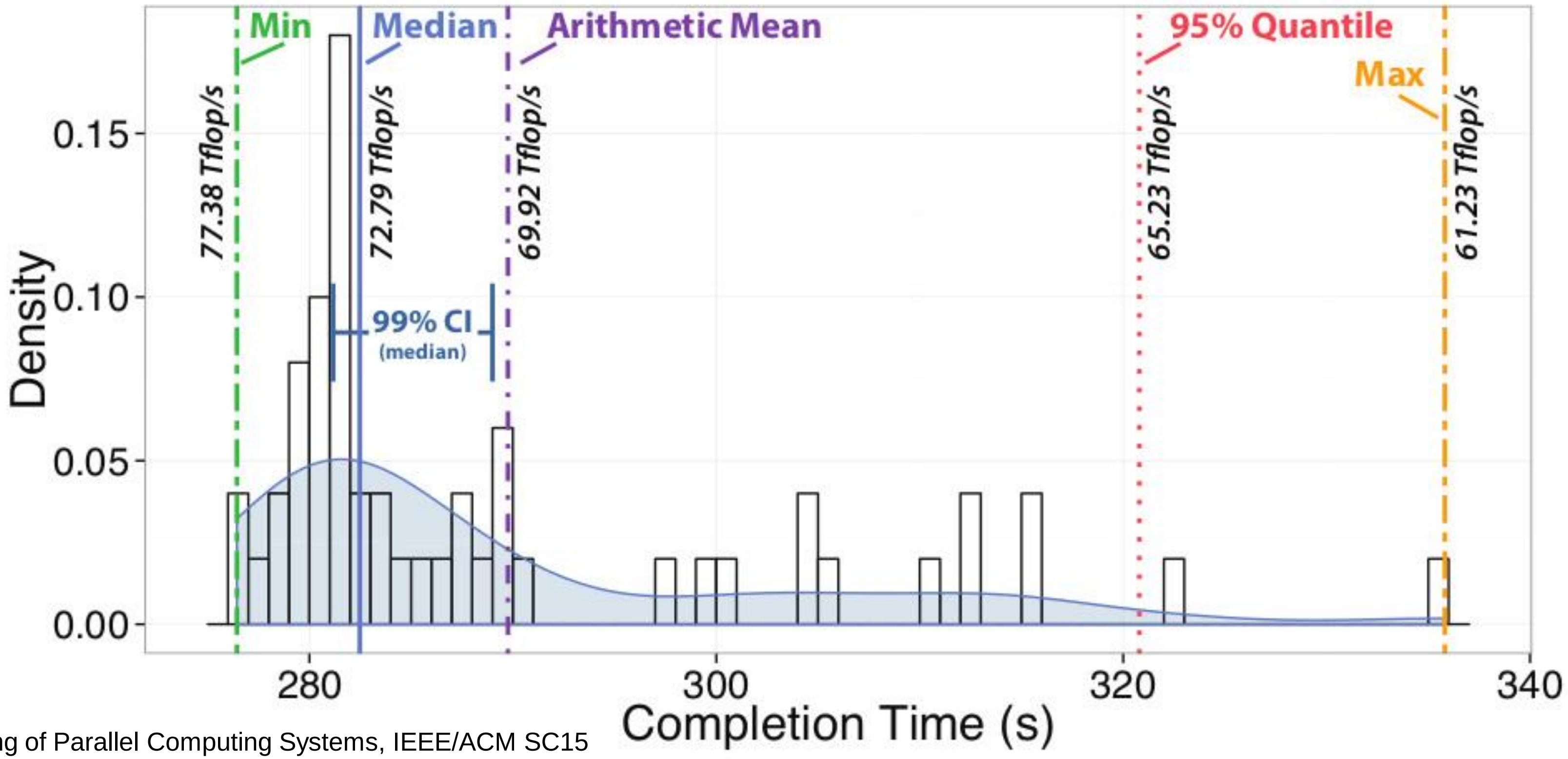
Dealing with variation

Rule 6: *Do not assume normality of collected data (e.g., based on the number of samples) without diagnostic checking.*

- **Most events will slow down performance**
  - **Heavy right-tailed distributions**
- **The Central Limit Theorem only applies asymptotically**
  - **Some papers/textbook mention “30-40 samples”, don’t trust them!**
- **Two papers used CIs around the mean without testing for normality**

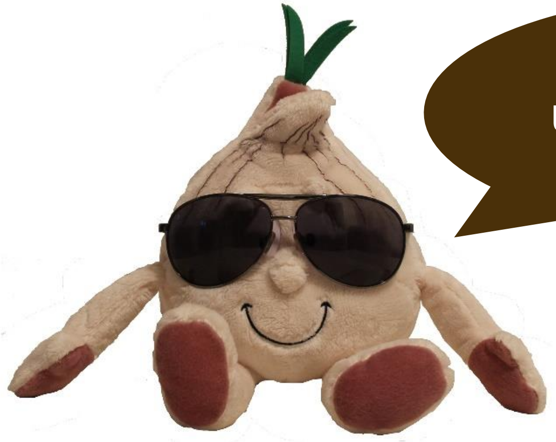
Dealing with non-normal data – nonparametric statistics

- **Rank-based measures (no assumption about distribution)**
  - Essentially always better than assuming normality
- **Example: median (50th percentile) vs. mean for HPL**
  - Rather stable statistic for expectation
  - Other percentiles (usually 25th and 75th) are also useful



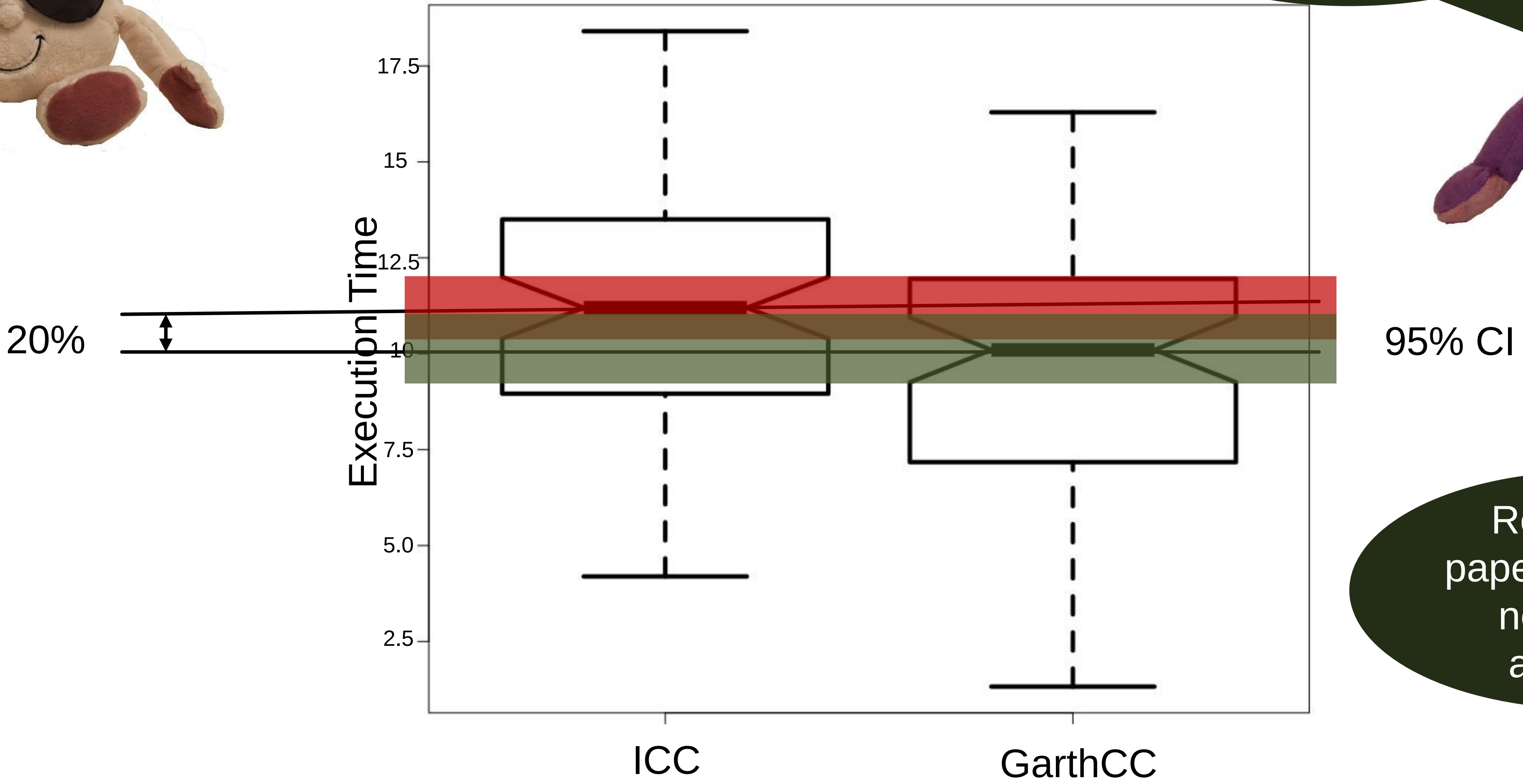
TH, Belli: Scientific Benchmarking of Parallel Computing Systems, IEEE/ACM SC15

Comparing nondeterministic measurements



I saw variance using GarthCC as well!

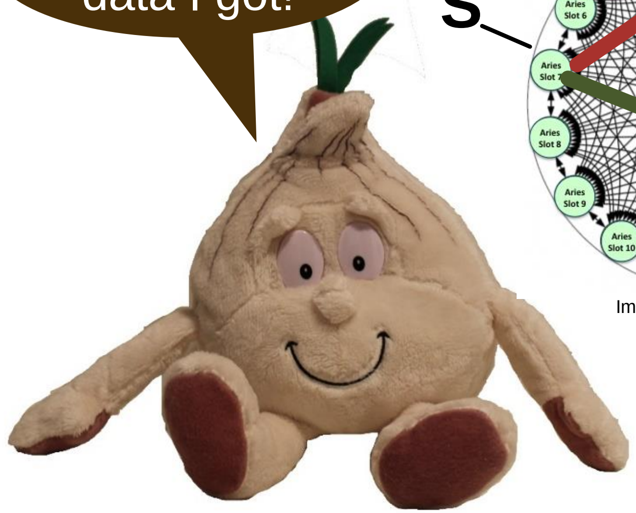
Show me the data!



Retract the paper! You have not shown anything!

What if the data looks weird!?

Look what data I got!



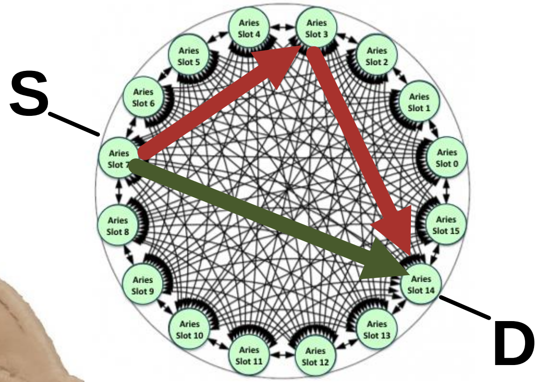
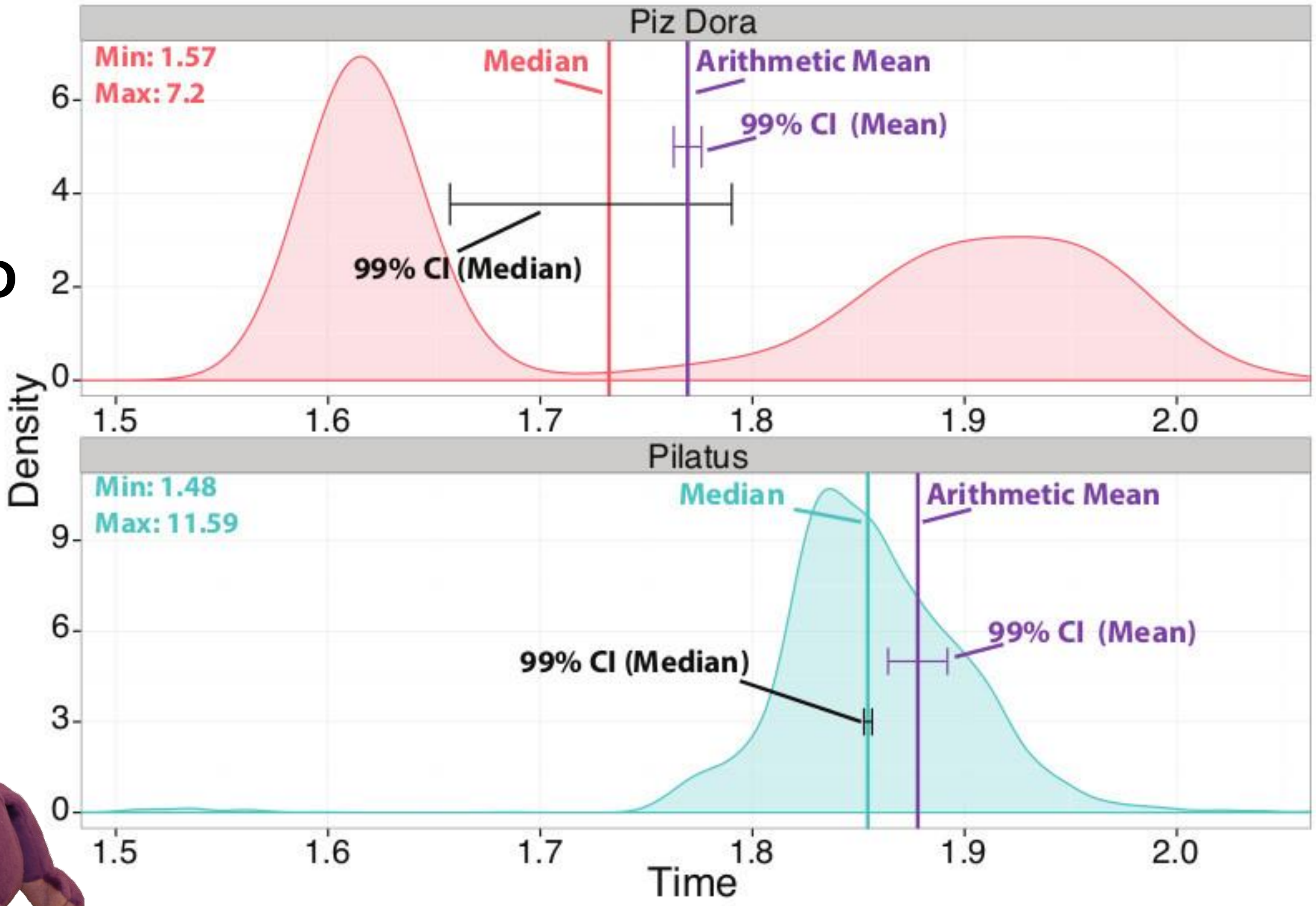
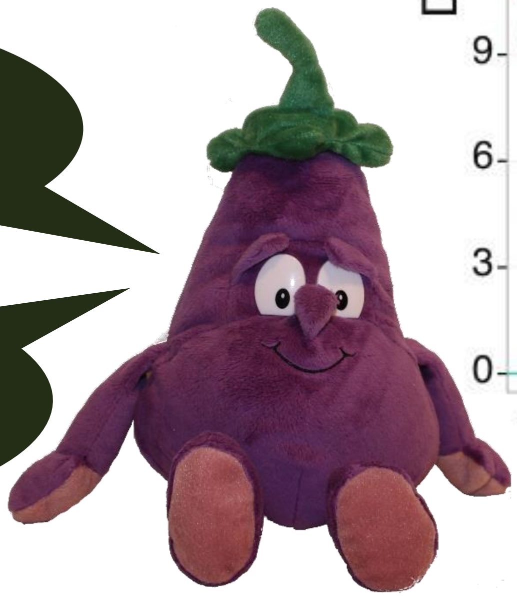


Image credit: nersc.gov



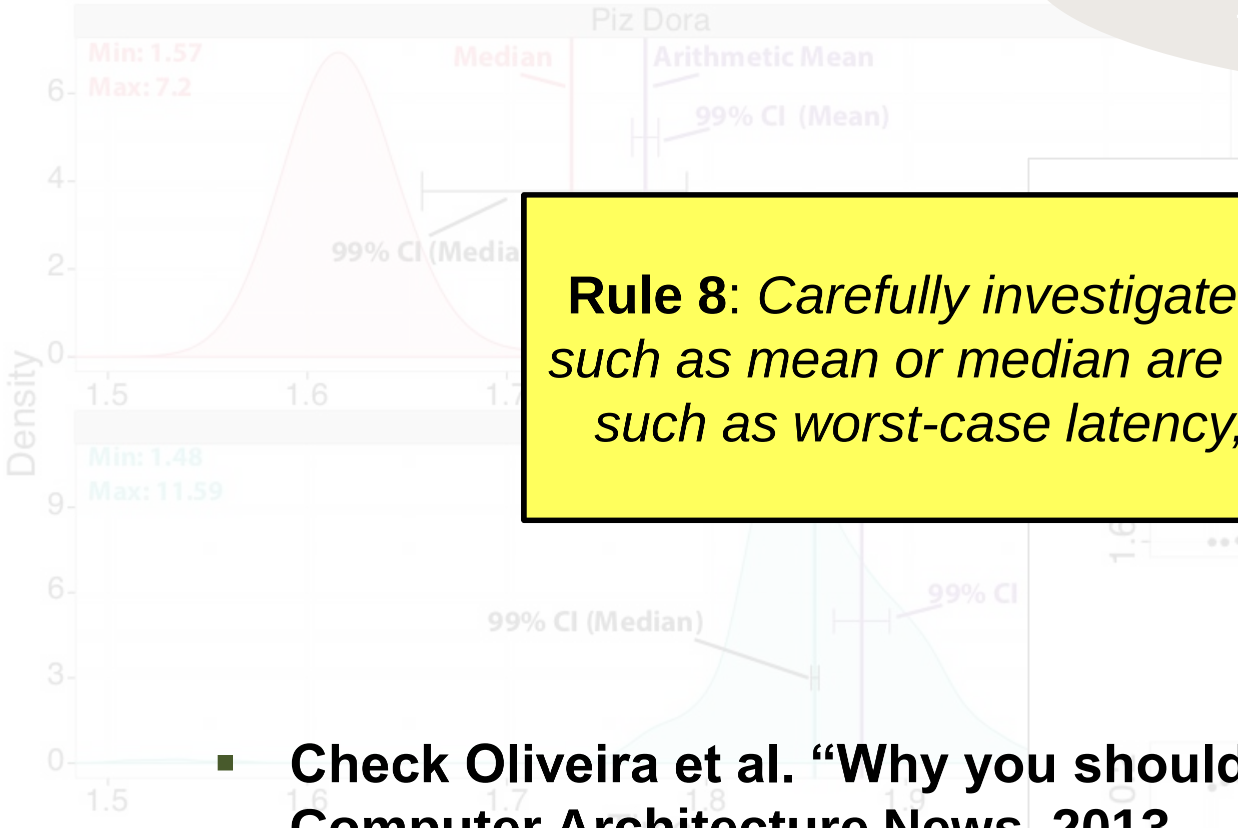
Clearly, the mean/median are not sufficient!

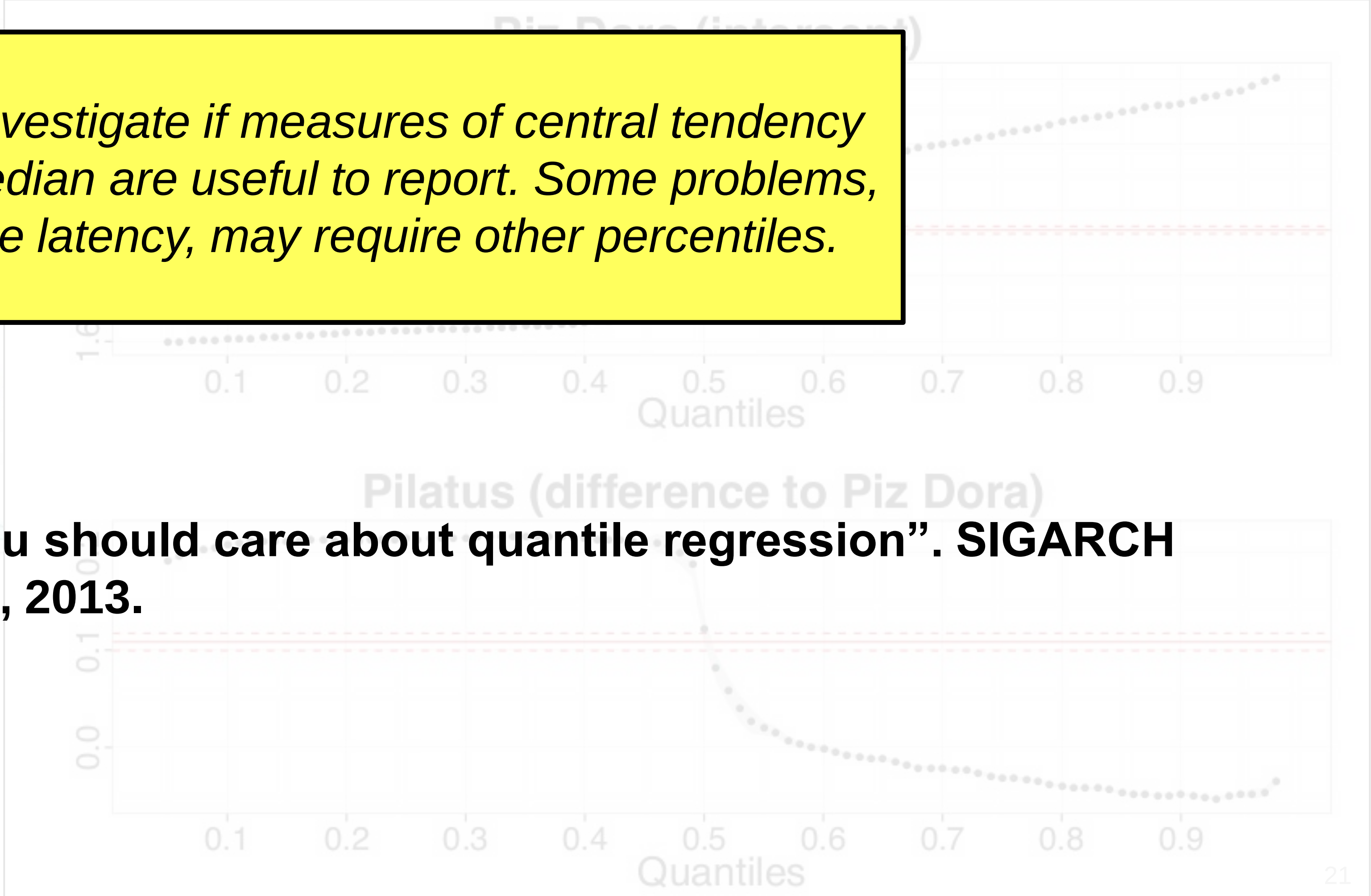
Try quantile regression!



Quantile Regression

Wow, so Pilatus is better for (worst-case) latency-critical workloads even though Dora is expected to be faster





Rule 8: *Carefully investigate if measures of central tendency such as mean or median are useful to report. Some problems, such as worst-case latency, may require other percentiles.*

- **Check Oliveira et al. "Why you should care about quantile regression". SIGARCH Computer Architecture News, 2013.**

How many measurements are needed?

- **Measurements can be expensive!**
  - Yet necessary to reach certain confidence

- **How to determine the minimal number of measurements?**
  - Measure until the confidence interval has a certain acceptable width
  - For example, measure until the 95% CI is within 5% of the mean/median
  - Can be computed analytically assuming normal data
  - Compute iteratively for nonparametric statistics

- **Often heard: "we cannot afford more than a single measurement"**
  - E.g., Gordon Bell runs
  - Well, then one cannot say anything about the variance

    Even 3-4 measurement can provide very tight CI (assuming normality)

    Can also exploit repetitive nature of many applications

Experimental design

Rule 9: *Document all varying factors and their levels as well as the complete experimental setup (e.g., software, hardware, techniques) to facilitate reproducibility and provide interpretability.*

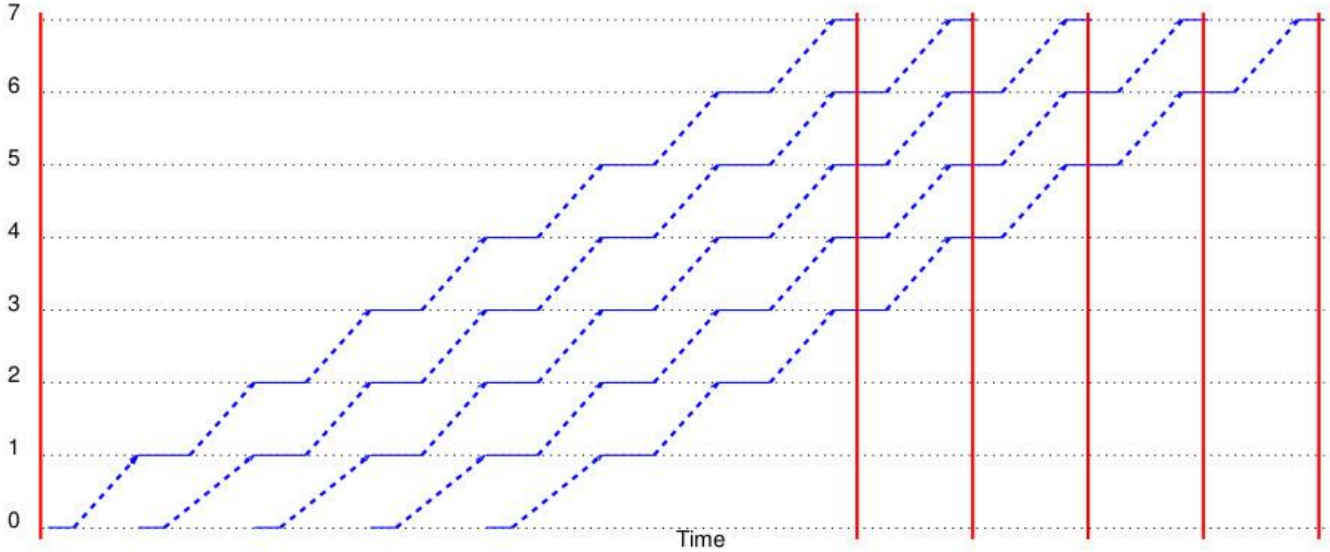
- **We recommend factorial design**
- **Consider parameters such as node allocation, process-to-node mapping, network or node contention**
  - If they cannot be controlled easily, use randomization and model them as random variable
- **This is hard in practice and not easy to capture in rules**

Time in parallel systems

My simple broadcast takes only one latency!

But I measured it so it must be true!

```
t = -MPI_Wtime();
for(i=0; i<1000; i++) {
  MPI_Bcast(…);
}
t += MPI_Wtime();
t /= 1000;
```

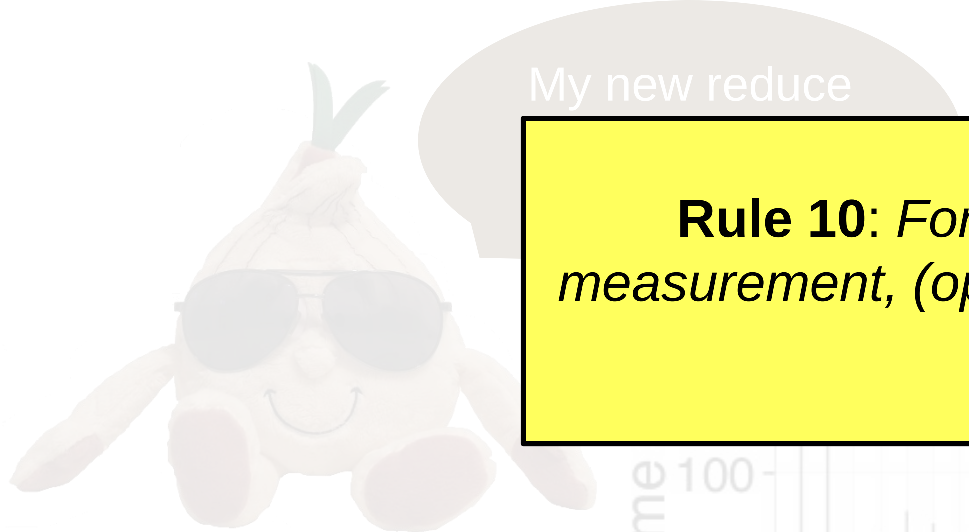


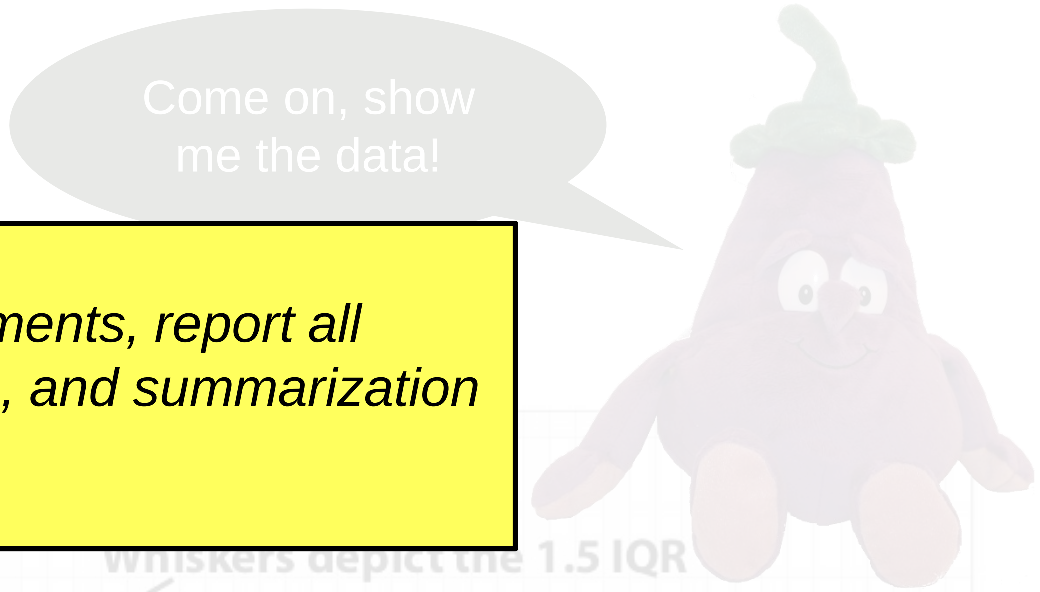
That's nonsense!

…

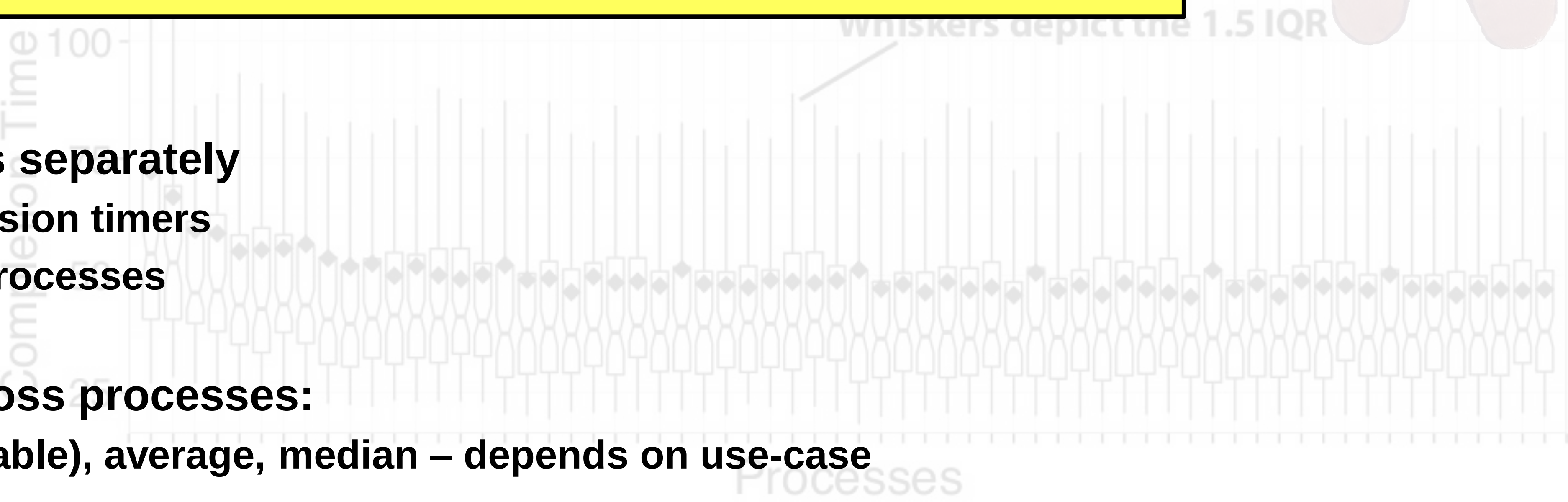
Measure each operation separately!

Summarizing times in parallel systems!





Rule 10: *For parallel time measurements, report all measurement, (optional) synchronization, and summarization techniques.*



- **Measure events separately**
  - **Use high-precision timers**
  - **Synchronize processes**

- **Summarize across processes:**
  - **Min/max (unstable), average, median – depends on use-case**

Give times a meaning!

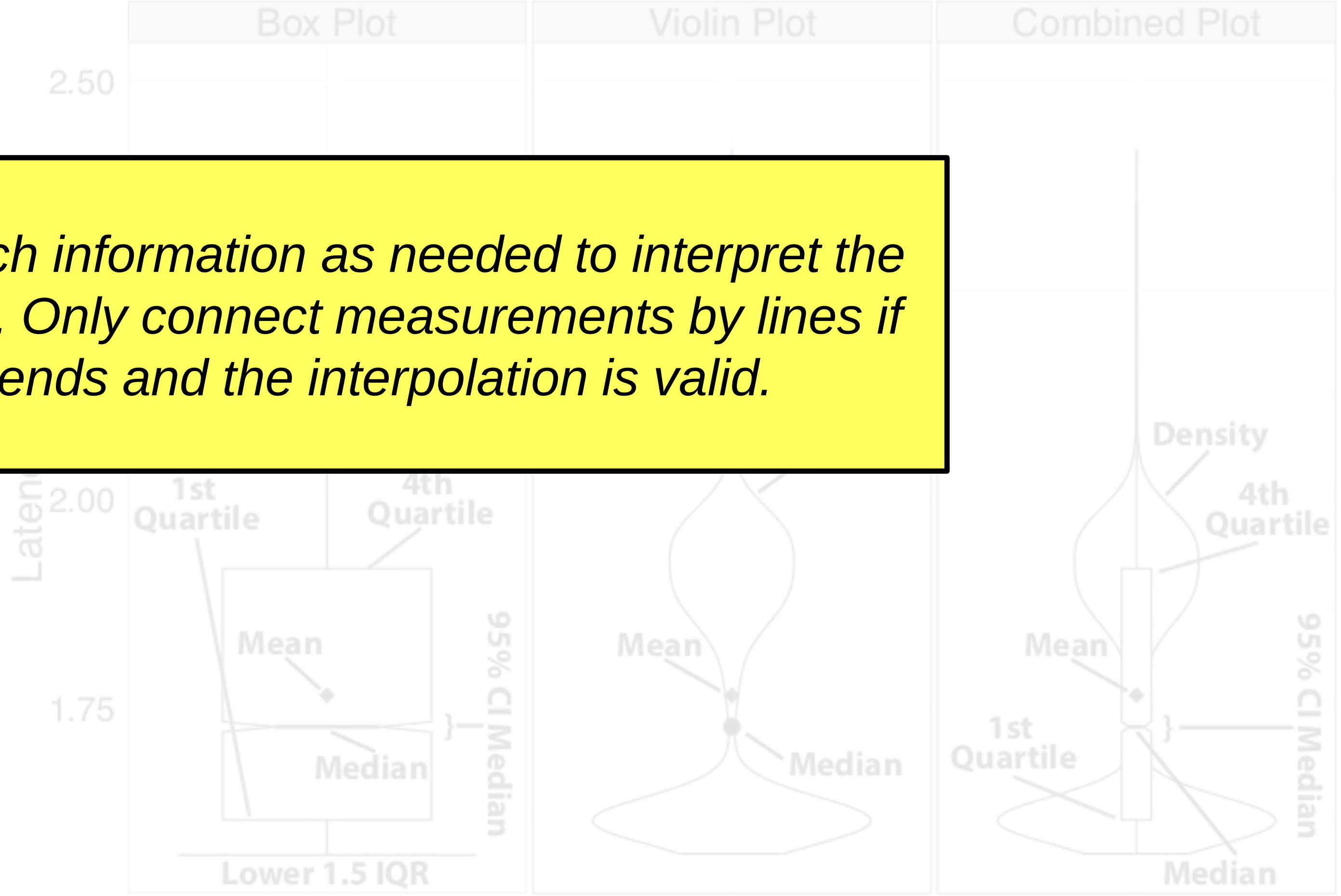
Rule 11: *If possible, show upper performance bounds to facilitate interpretability of the measured results.*

- **Model computer system as k-dimensional space**
  - Each dimension represents a capability
    Floating point, Integer, memory bandwidth, cache bandwidth, etc.
  - Features are typical rates
  - Determine maximum rate for each dimension
    E.g., from documentation or benchmarks
- **Can be used to proof optimality of implementation**
  - If the requirements of the bottleneck dimension are minimal

TH, Belli: Scientific Benchmarking of Parallel Computing Systems, IEEE/ACM SC15

Plot as much information as possible!

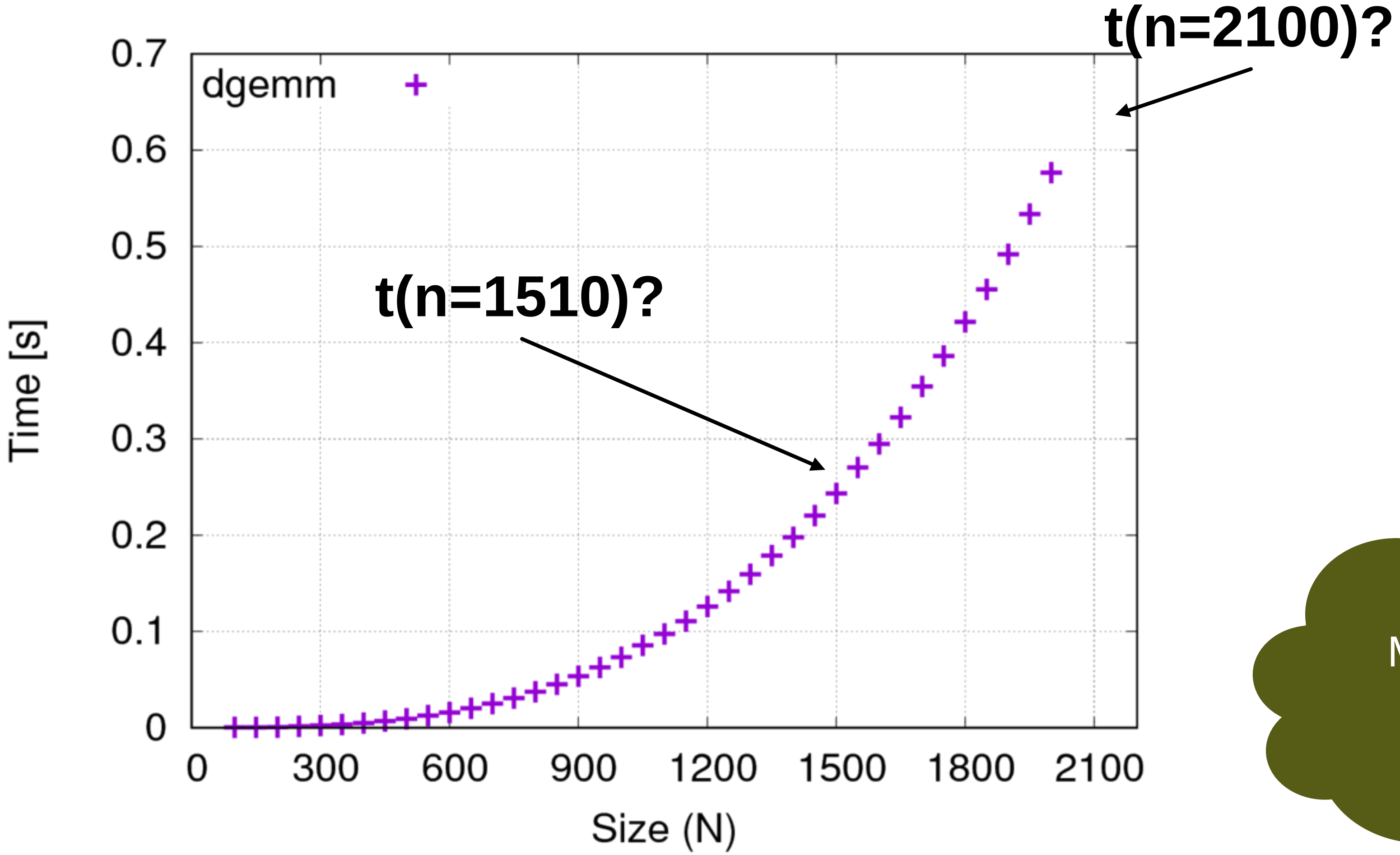
My most common request was “show me the data”

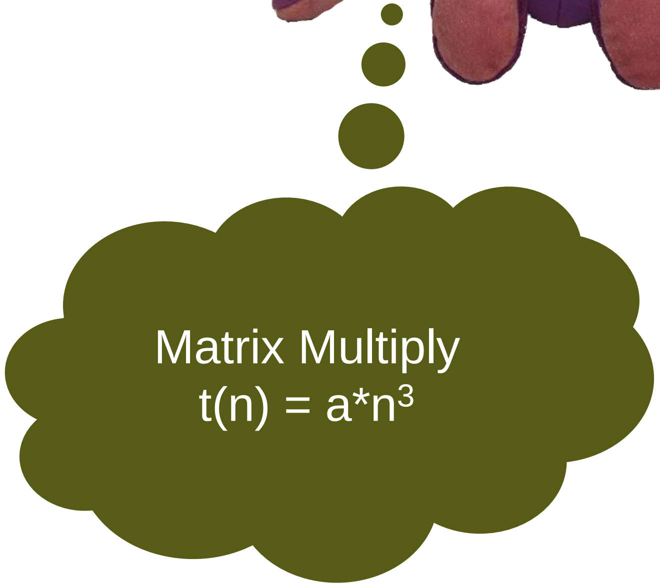


Rule 12: *Plot as much information as needed to interpret the experimental results. Only connect measurements by lines if they indicate trends and the interpolation is valid.*

This is how I should have presented the Dora results.

We have the (statistically sound) data, now what?



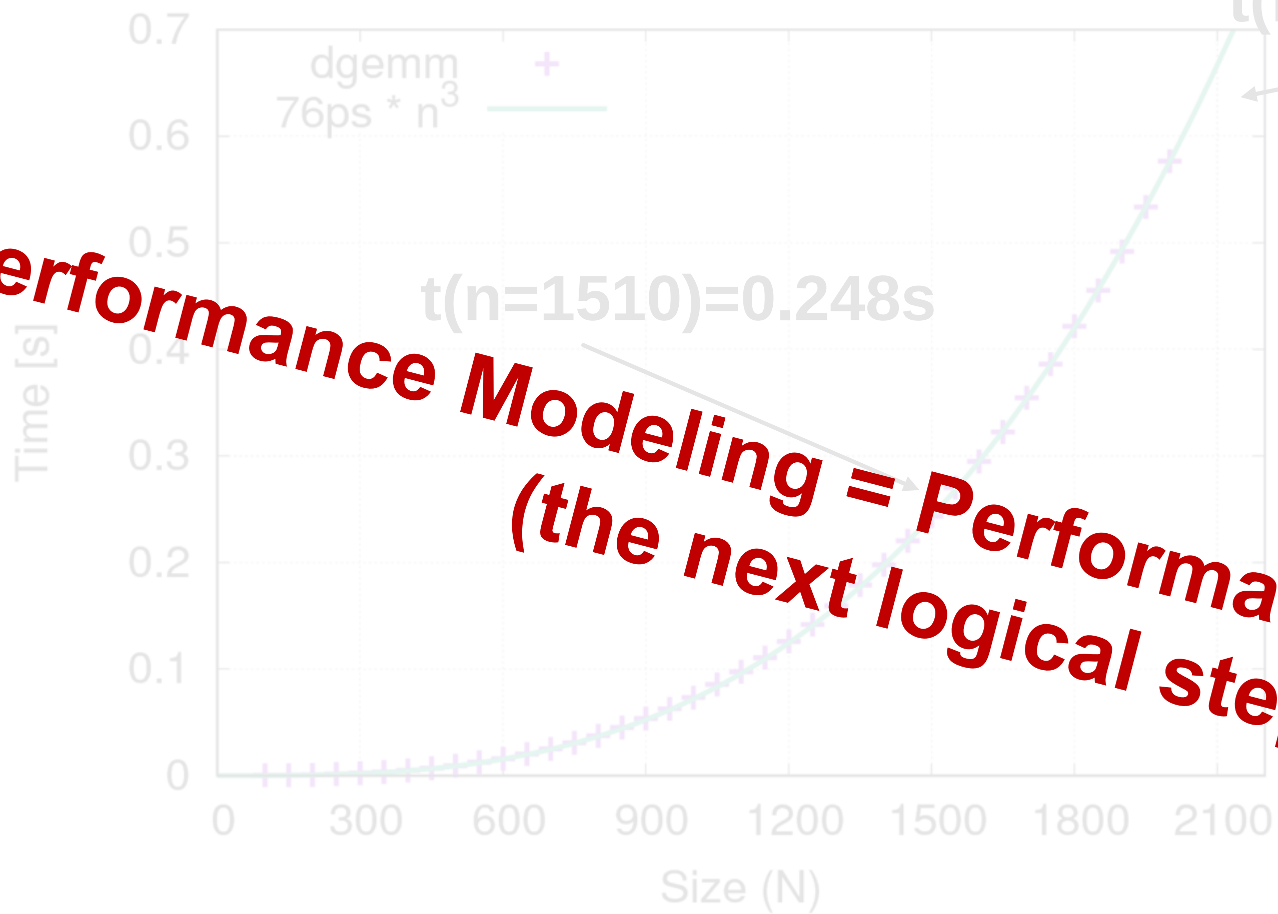


The 99% confidence interval is within 1% of the reported median.

TH, W. Gropp, M. Snir, W. Kramer: Performance Modeling for Systematic Performance Tuning, IEEE/ACM SC11

We have the (statistically sound) data, now what?

t(n=2100)=0.667s

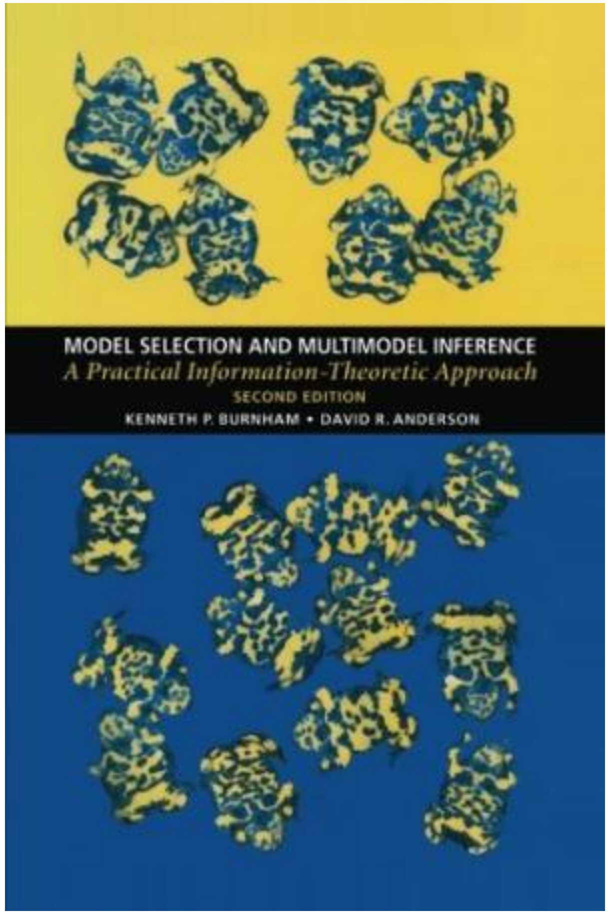


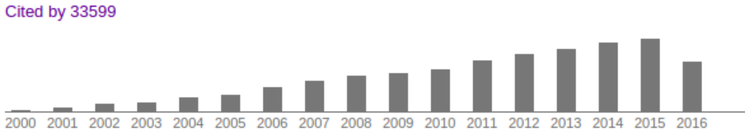
Performance Modeling = Performance Analysis v 2.0 (the next logical step)

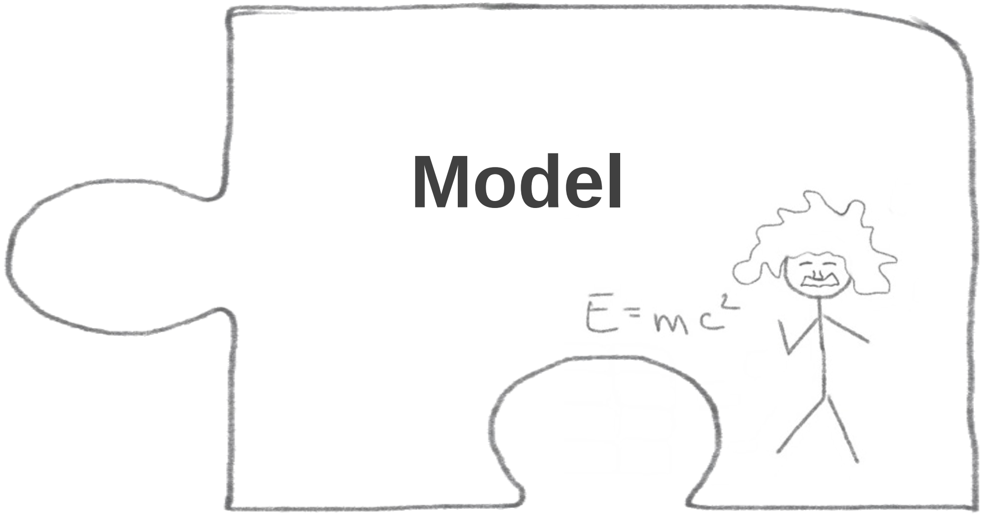
The 99% confidence interval is within 1% of the reported median.
The adjusted R^2 of the model fit is 0.99

TH, W. Gropp, M. Snir, W. Kramer: Performance Modeling for Systematic Performance Tuning, IEEE/ACM SC11

Part II: Model



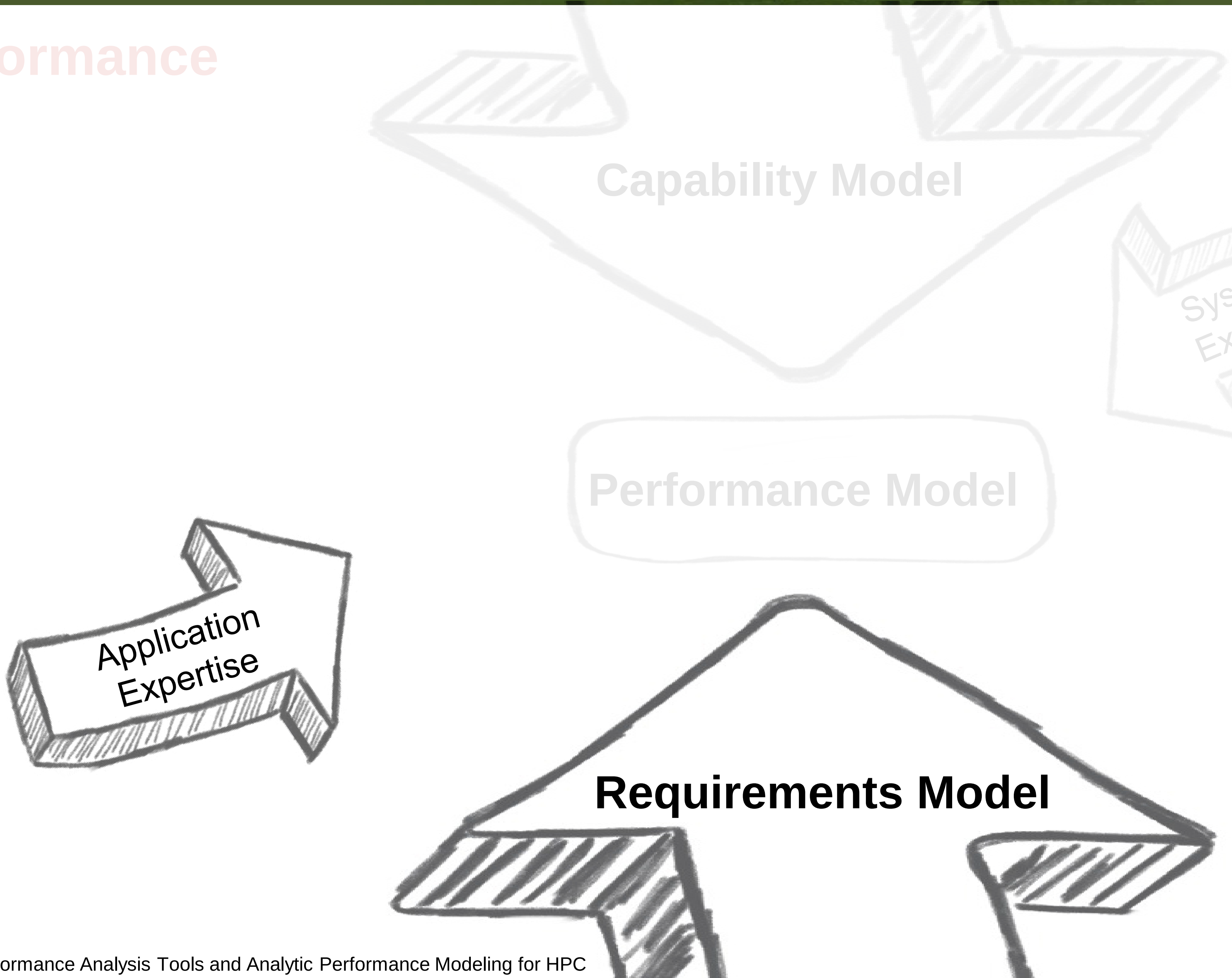




Burnham, Anderson: *"A model is a simplification or approximation of reality and hence will not reflect all of reality. ... Box noted that "all models are wrong, but some are useful." While a model can never be "truth," a model might be ranked from very useful, to useful, to somewhat useful to, finally, essentially useless."*

This is generally true for all kinds of modeling.
We focus on **performance modeling** in the following!

Performance Modeling



Requirements modeling I: Six-step performance modeling

Input parameters

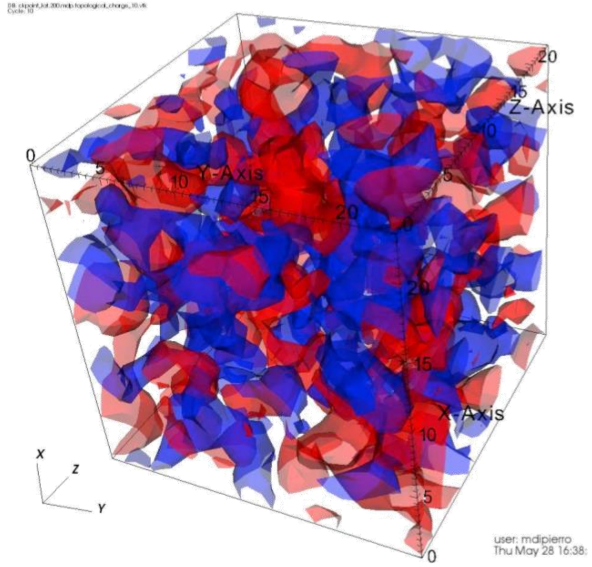
Describe application kernels

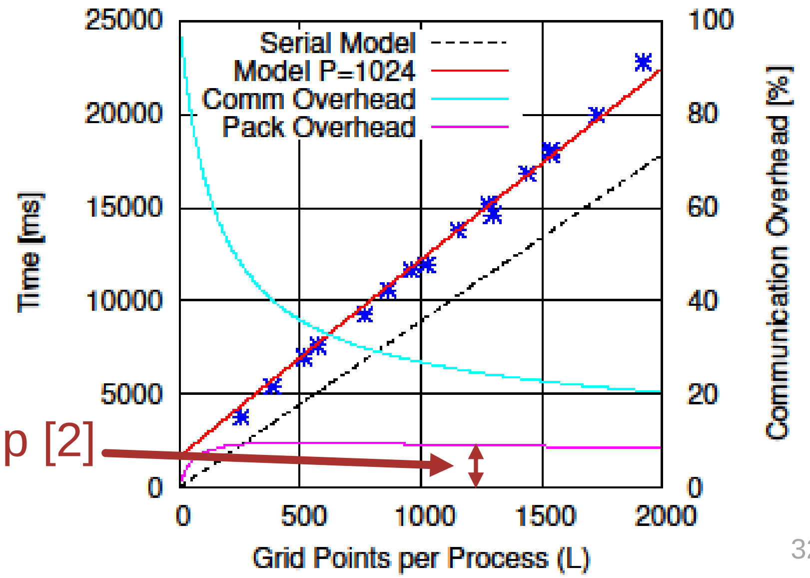
Communication pattern

Communication / computation overlap

Fit sequential baseline

Communication parameters





10-20% speedup [2]

[1] TH, W. Gropp, M. Snir and W. Kramer: Performance Modeling for Systematic Performance Tuning, SC11

[2] TH and S. Gottlieb: Parallel Zero-Copy Algorithms for Fast Fourier Transform and Conjugate Gradient using MPI Datatypes, EuroMPI'10

Requirements modeling II: Automated best-fit modeling

- **Manual kernel selection and hypothesis generation is time consuming (boring and tricky)**
- **Idea: Automatically select best (scalability) model from predefined search space**

number of terms

Number of processes

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

(model) constant

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

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

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

[1]: A. Calotoiu, TH, M. Poke, F. Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes, IEEE/ACM SC13

Requirements modeling II: Automated best-fit modeling

- **Manual kernel selection and hypothesis generation is time consuming (and boring)**
- **Idea: Automatically select best model from predefined space**

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

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

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

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

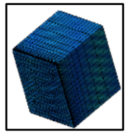
$$c_1 \cdot \log(p) + c_2 \cdot p$$
$$c_1 \cdot \log(p) + c_2 \cdot p \cdot \log(p)$$
$$c_1 \cdot \log(p) + c_2 \cdot p^2$$
$$c_1 \cdot \log(p) + c_2 \cdot p^2 \cdot \log(p)$$
$$c_1 \cdot p + c_2 \cdot p \cdot \log(p)$$
$$c_1 \cdot p + c_2 \cdot p^2$$
$$c_1 \cdot p + c_2 \cdot p^2 \cdot \log(p)$$
$$c_1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

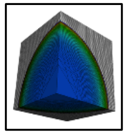
[1]: A. Calotoiu, T. Hoefler, M. Poke, F. Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes, IEEE/ACM SC13

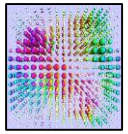
Tool support: Extra-P for automated best-fit modeling [1]

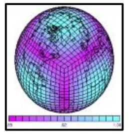
TECHNISCHE UNIVERSITÄT DARMSTADT

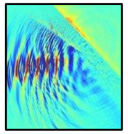


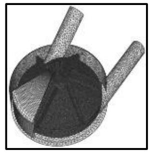

Sweep3d

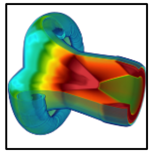

Lulesh


Milc

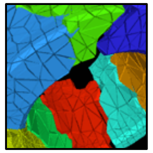

HOMME

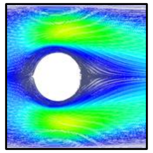

JUSPIC


XNS


BLAST


NEST


UG4


MP2C

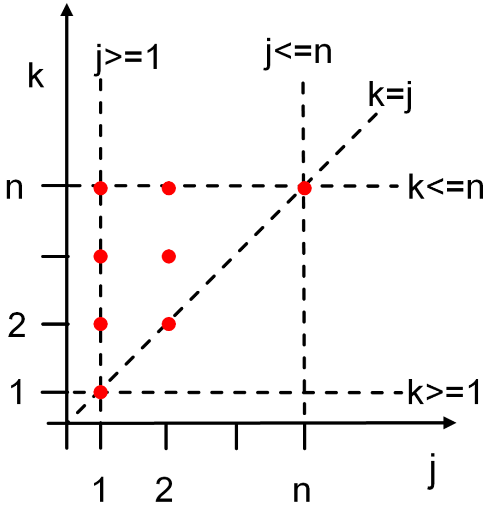
[1] Download Extra-P at: http://www.scalasca.org/software/extra-p/download.html

[2] A. Calotoiu, D. Beckingsale, C. W. Earl TH, I. Karlin, M. Schulz, F. Wolf: Fast Multi-Parameter Performance Modeling, IEEE Cluster 2016

Requirements modeling III: Source-code analysis [1]

- **Extra-P selects model based on best fit to the data**
  - What if the data is not sufficient or too noisy?
- **Back to first principles**
  - The source code describes all possible executions
  - Describing all possibilities is too expensive, focus on counting loop iterations symbolically

```
for (j = 1; j <= n; j = j*2)
  for (k = j; k <= n; k = k++)
    OperationInBody(j,k);
```



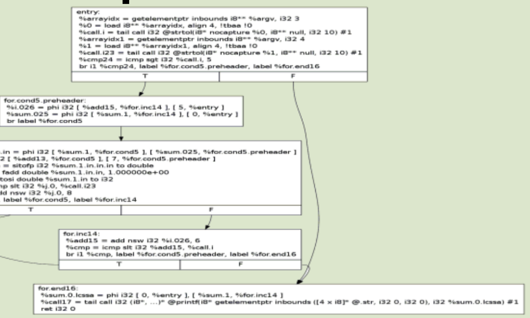
$$N = (n+1)\log_2 n - n + 2$$

Parallel program

```
do i = 1, procCols
   call mpi_irecv( buff, 2, dp_type, reduce_exch_proc(i),
                   i, mpi_comm_world, request, ierr )
   call mpi_send( buff2, 2, dp_type, reduce_exch_proc(i),
                   i, mpi_comm_world, ierr )
   call mpi_wait( request, status, ierr )
enddo

do i = id *n/p, ( id +1)* n/p
   do j = 1, nSize
      call compute
```

Loop extraction



Number of iterations

$$N = \sum_{i_1=0}^{n_1(x_{0,1})} \sum_{i_2=0}^{n_2(x_{0,2})} \cdots \sum_{i_{r-1}=0}^{n_{r-1}(x_{0,r-1})} n_r(x_{0,r}).$$

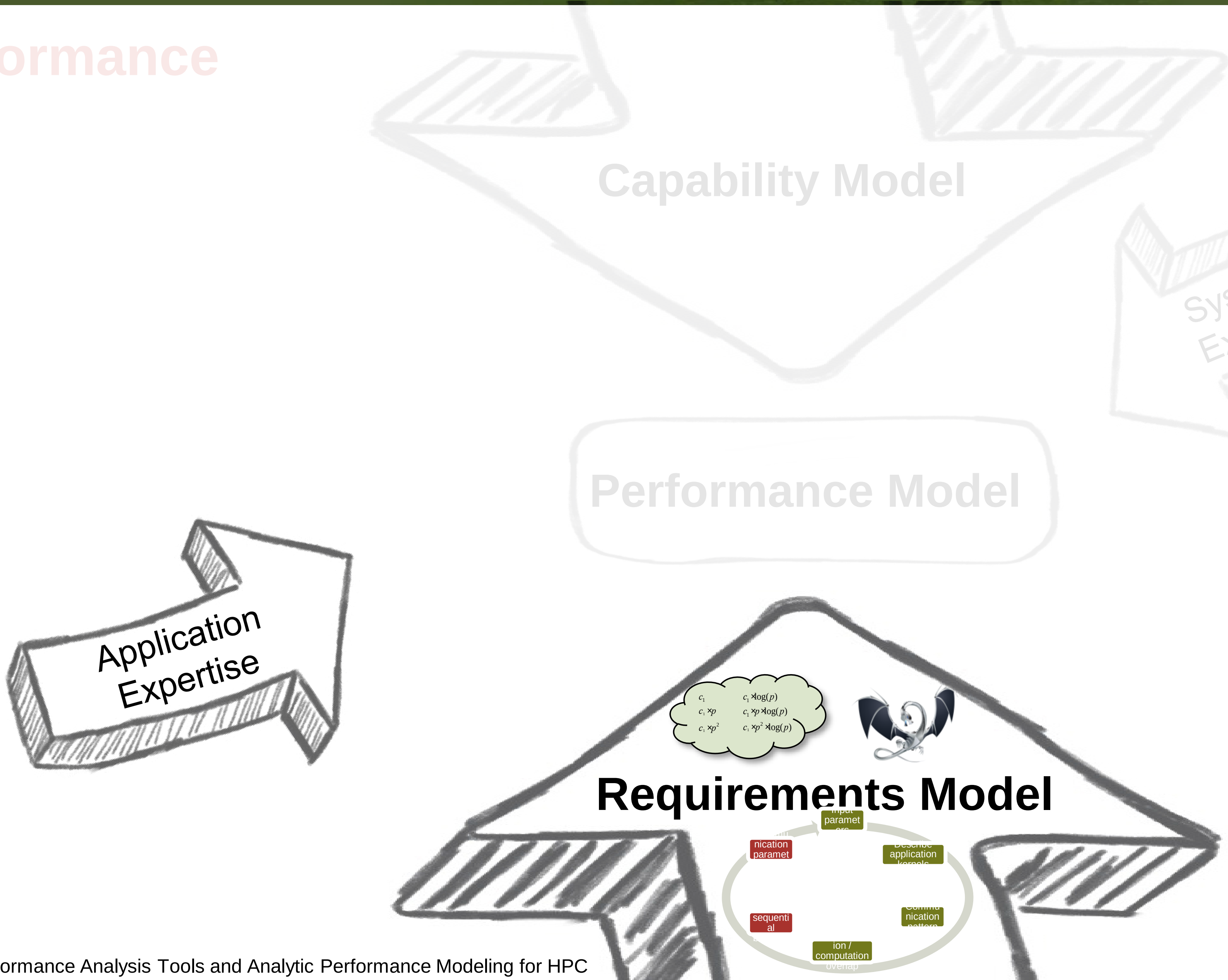
Requirements Models

$$W = N\big|_{p=1}$$

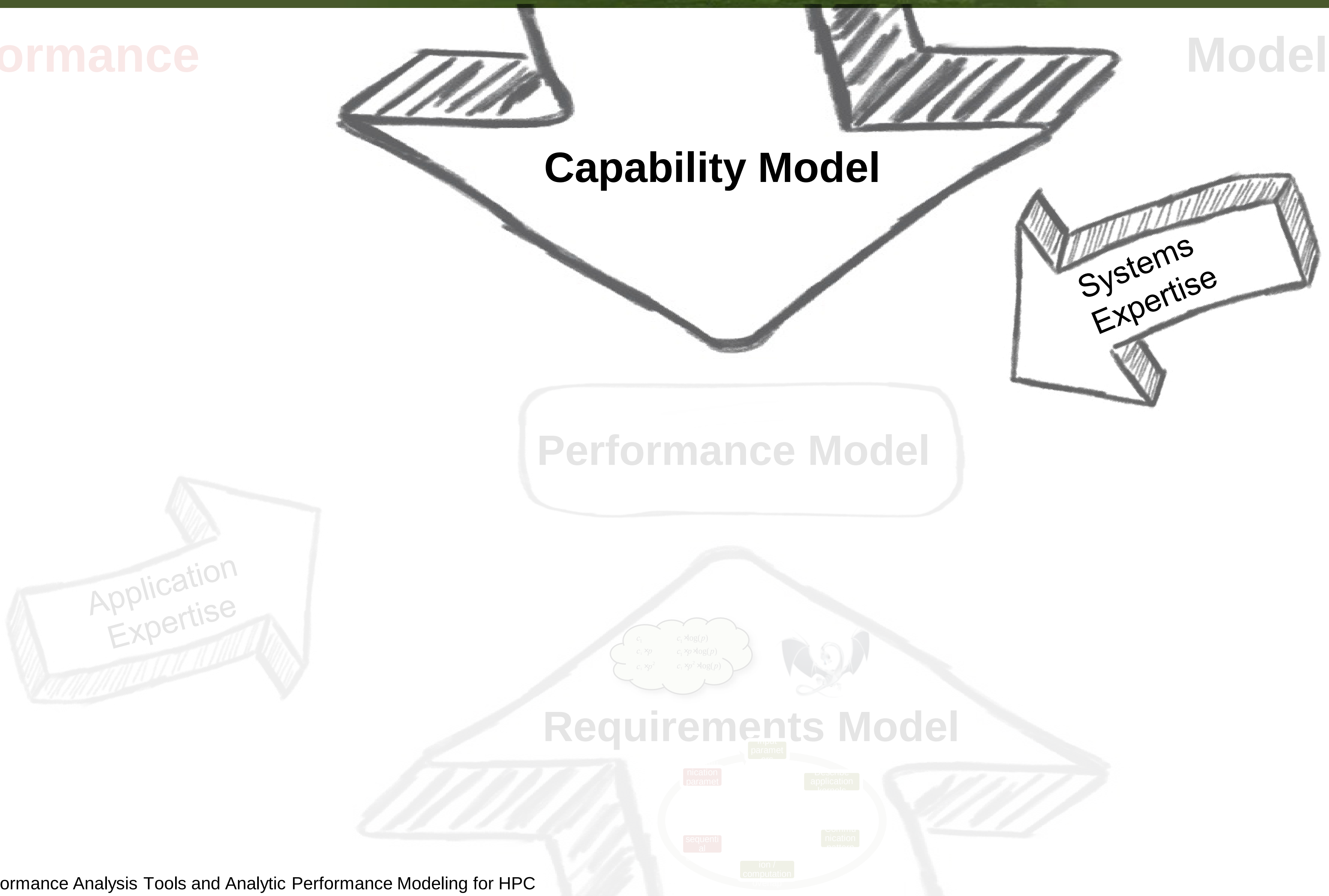
$$D = N\big|_{p\to\infty}$$

[1]: TH, G. Kwasniewski: Automatic Complexity Analysis of Explicitly Parallel Programs, ACM SPAA'14

Performance Modeling



Performance Modeling



Capability models for network communication

The LogP model family and the LogGOPS model [1]


A new parallel machine model reflects the critical technology trends underlying parallel computers

A PRACTICAL MODEL of PARALLEL COMPUTATION

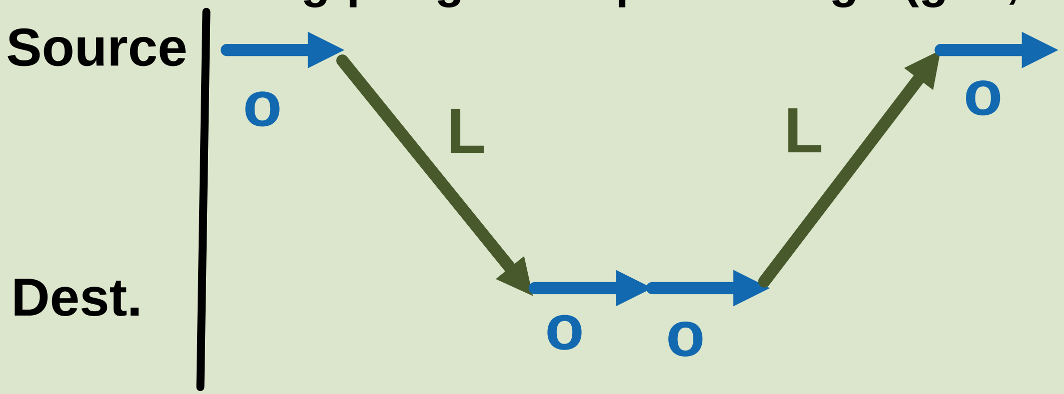
LogP

OUR GOAL IS TO DEVELOP A MODEL OF PARALLEL COMPUTATION THAT WILL serve as a basis for the design and analysis of fast, portable parallel algorithms, such as algorithms that can be implemented effectively on a wide variety of current and future parallel machines. If we look at the body of parallel algorithms developed under current parallel models, many are impractical because they exploit artificial factors not present in any rea-

PRAM consists of a collection of processors which compute synchronously in parallel and communicate with a global random access

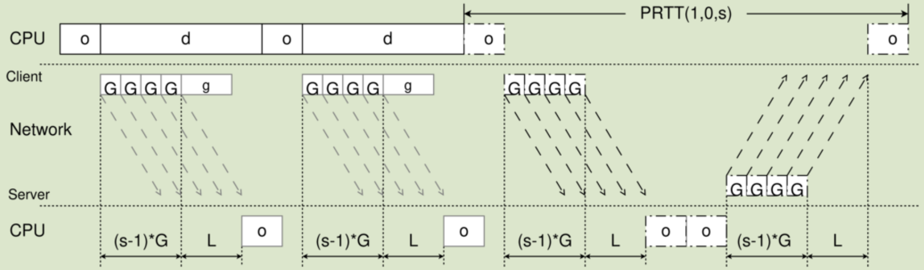
David E. Culler, Richard M. Karp, David Patterson, Abhijit Sahay, Eunice E. Santos, Klaus Erik Schauser, Ramesh Subramonian, and Thorsten von Eicken

Ping-pong in simplified LogP (g<o, P=2)



Finding LogGOPS parameters

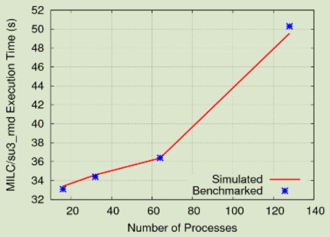
Netgauge [2], model from first principles, fit to data using special kernels

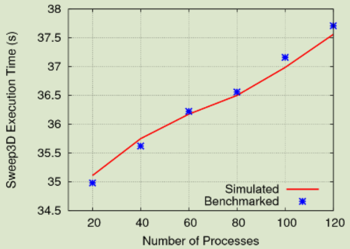


Large scale LogGOPS Simulation

LogGOPSim [1], simulates LogGOPS with 10 million MPI ranks

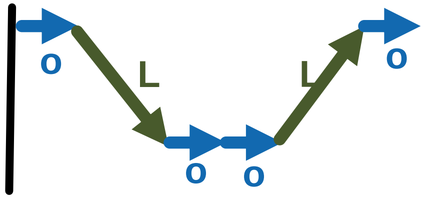
<5% error



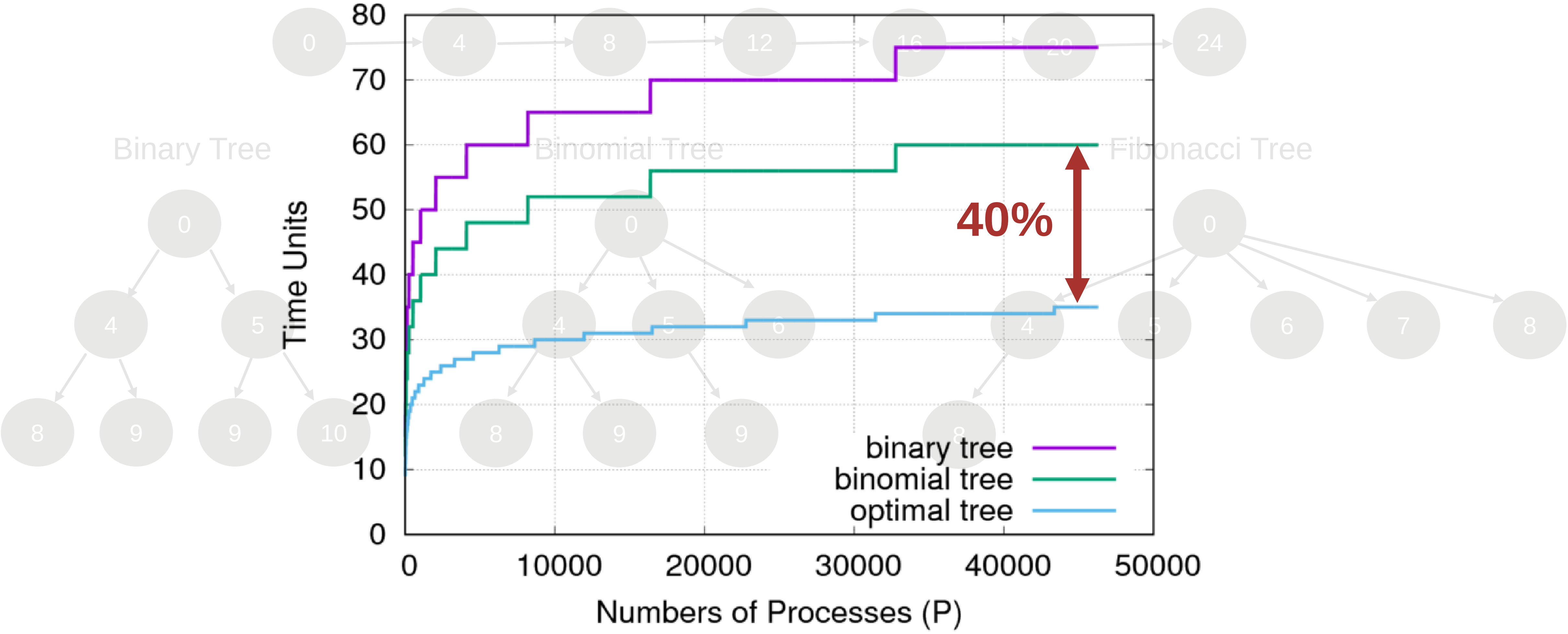


[1]: TH, T. Schneider and A. Lumsdaine: LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model, LSAP 2010, https://spcl.inf.ethz.ch/Research/Performance/LogGOPSim/
[2]: TH, T. Mehlan, A. Lumsdaine and W. Rehm: Netgauge: A Network Performance Measurement Framework, HPCC 2007, https://spcl.inf.ethz.ch/Research/Performance/Netgauge/

2) Design optimal algorithms – small broadcast in LogP

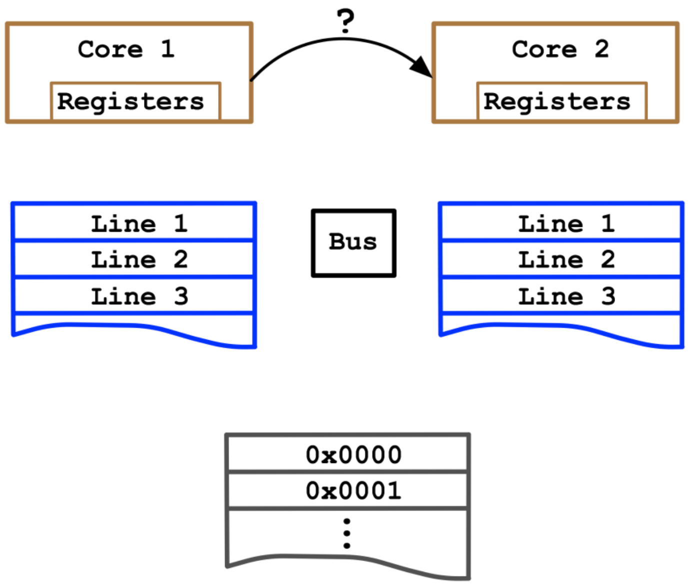


L=2, o=1, P=7

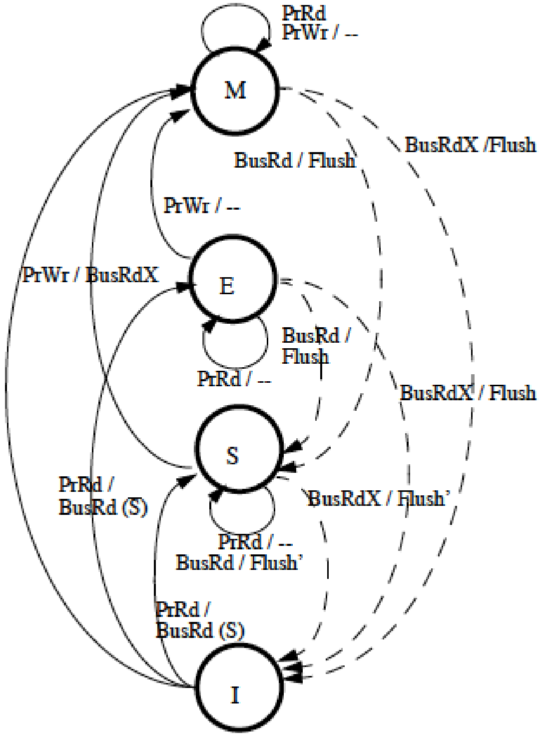


TH, D. Moor: Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations, JSFI 2015

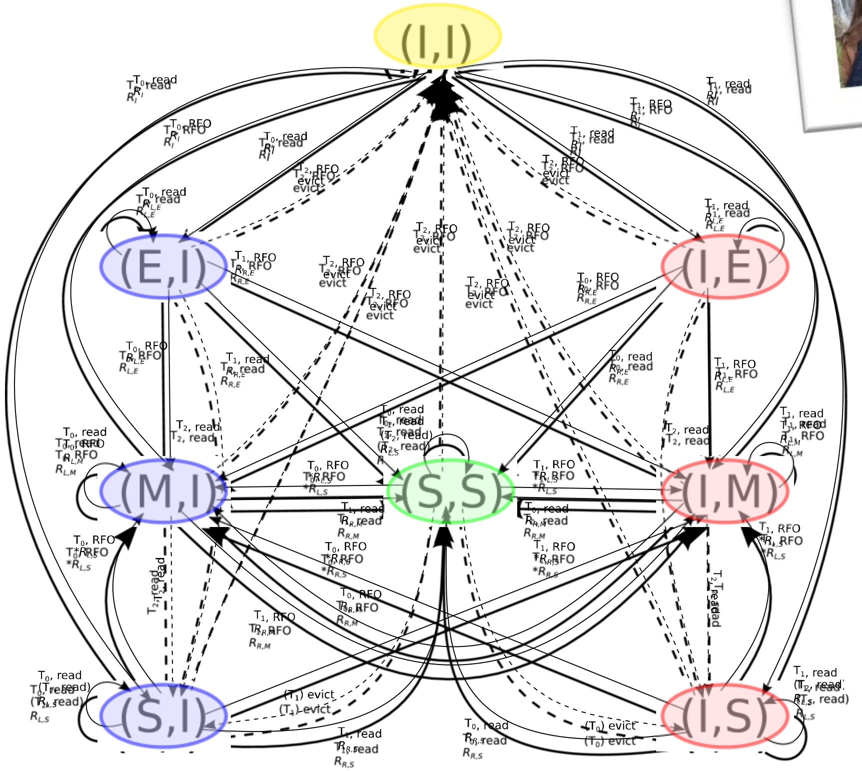
Capability models for cache-to-cache communication

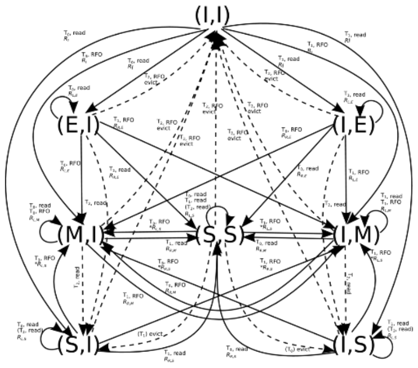


X



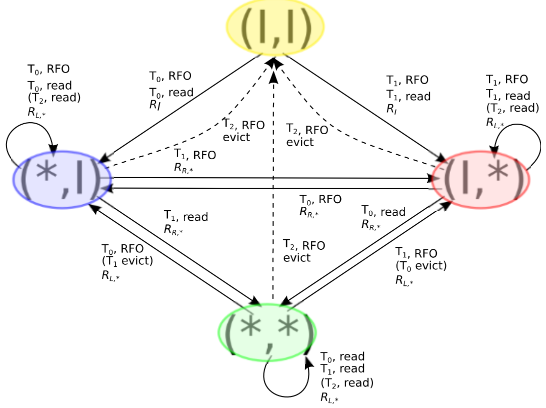
=





|

=



Invalid read $R_I \approx 135$ ns

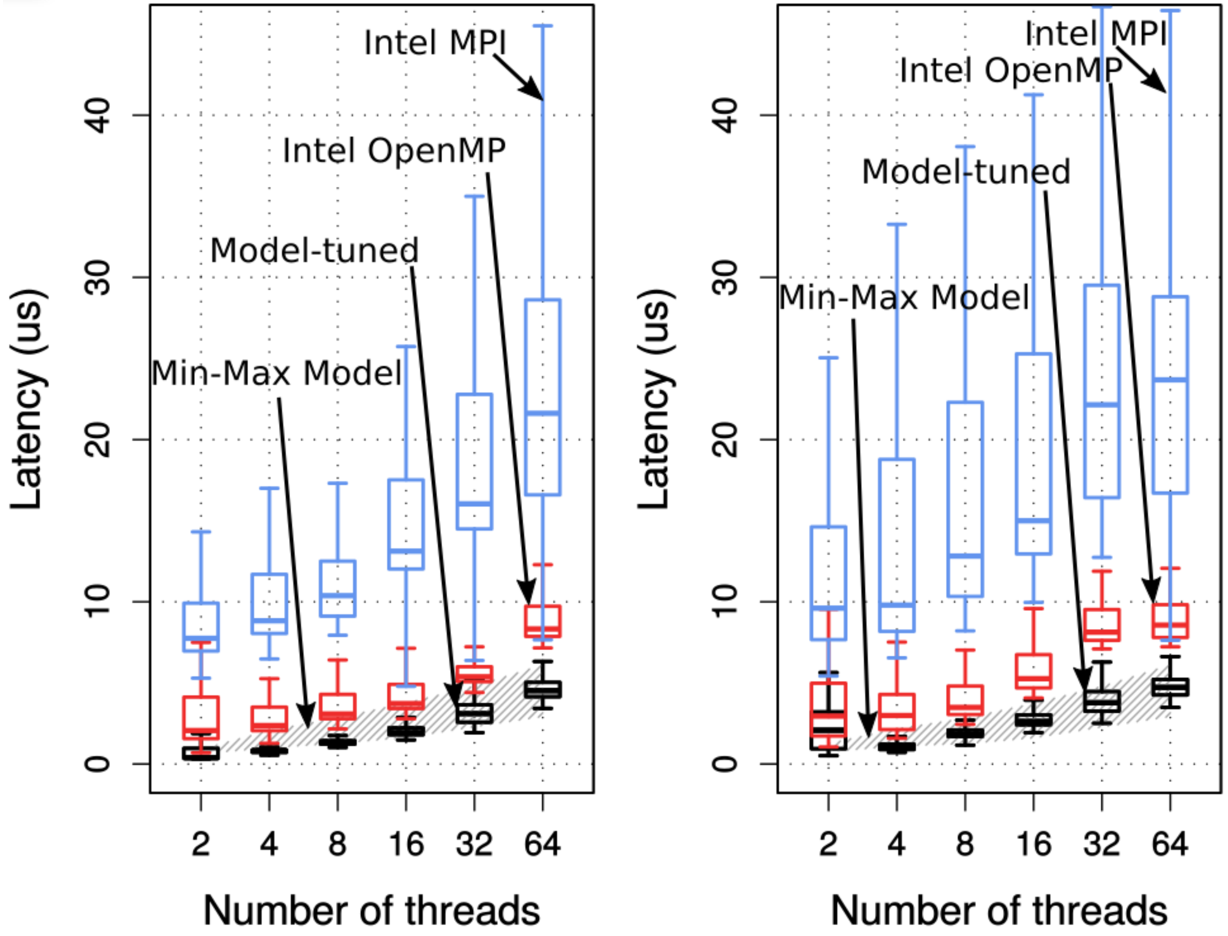
Local read: $R_L = 3.8$ ns

Remote read $R_R \approx 115$ ns

S. Ramos, TH: *"Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL", IEEE IPDPS'17*

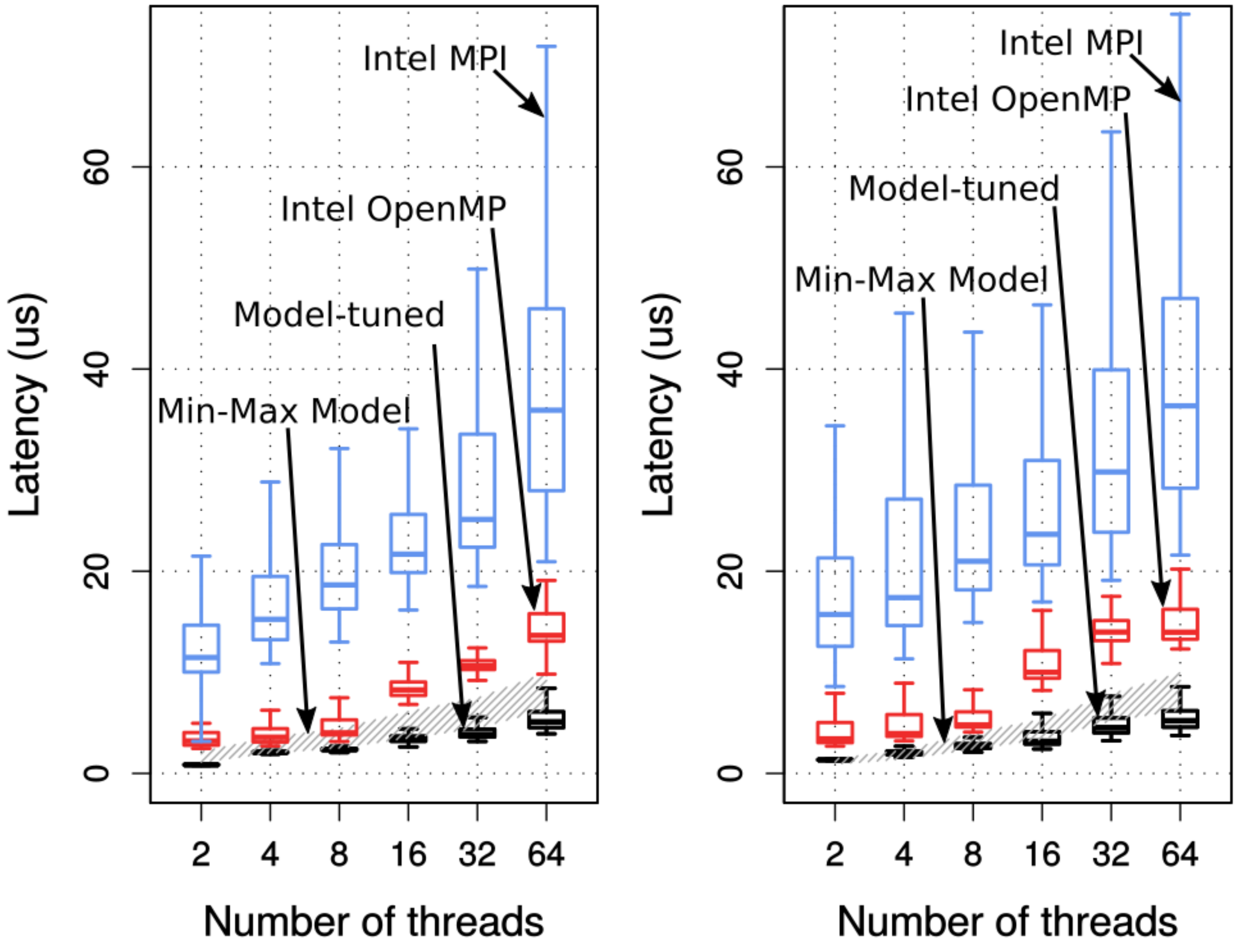
S. Ramos, TH: *"Modeling Communication in Cache-Coherent SMP Systems - A Case-Study with Xeon Phi ", ACM HPDC'13*

Model-tuned Barrier and Reduce vs. Intel’s OpenMP and MPI



(a) Filling Tiles. (b) Scatter.

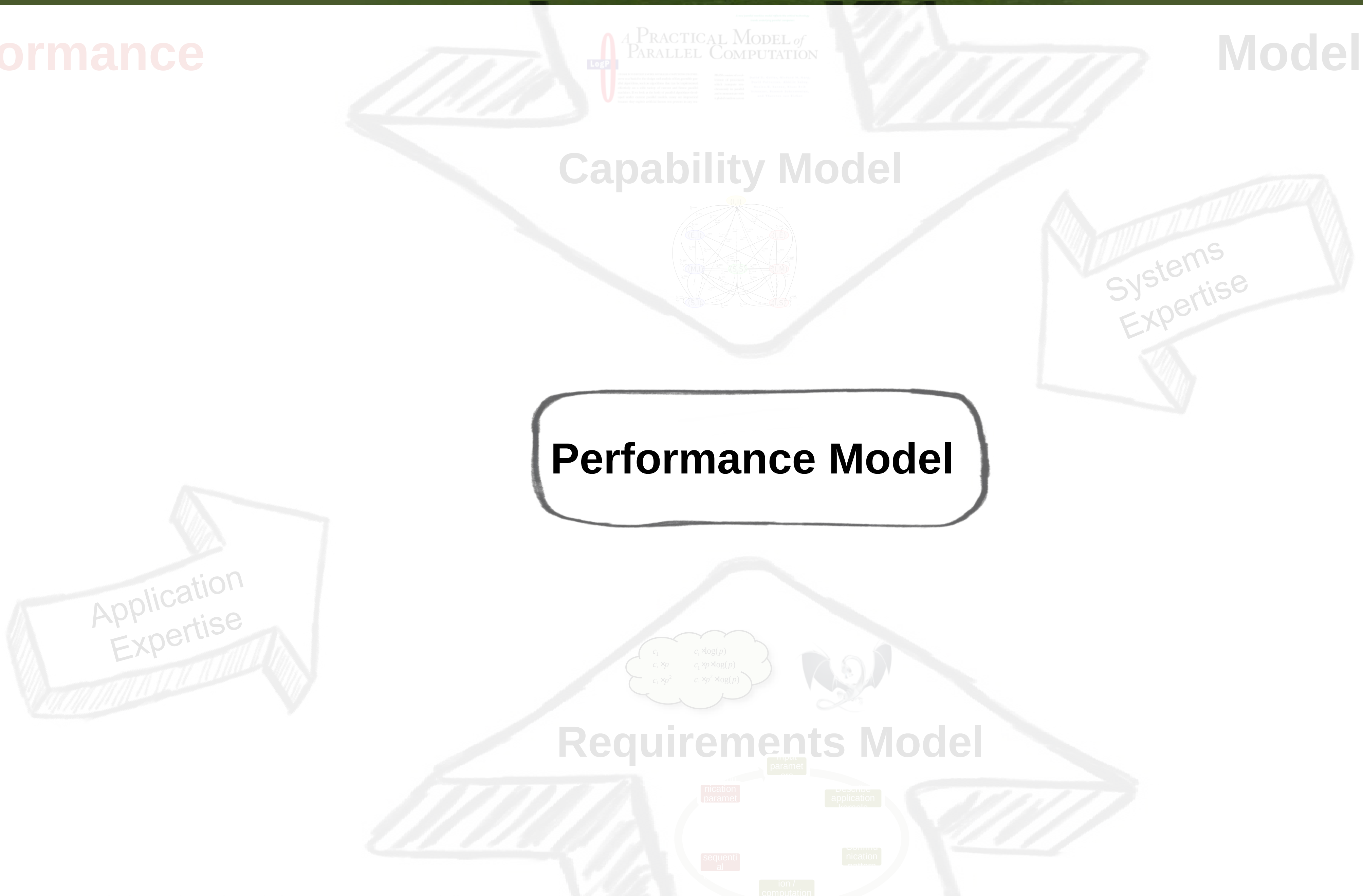
Barrier (7x faster than OpenMP)



(a) Filling Tiles. (b) Scatter.

Reduce (5x faster then OpenMP)

Performance Modeling



Conclusions and call for action

- **Performance may not be reproducible**
  - At least not for many (important) results
- **Interpretability fosters scientific progress**
  - Enables to build on results
  - Sounds statistics is the biggest gap today
- **We need to foster interpretability**
  - Do it ourselves (this is not easy)
  - Teach young students
  - Maybe even enforce in TPCs
- **See the 12 rules as a start**
  - Need to be extended (or concretized)
  - Much is implemented in LibSciBench [1]

No vegetables were harmed for creating these slides!

Acknowledgments

- **ETH's mathematics department (home of R)**
  - Hans Rudolf Künsch, Martin Maechler, and Robert Gantner
- **Comments on early drafts**
  - David H. Bailey, William T. Kramer, Matthias Hauswirth, Timothy Roscoe, Gustavo Alonso, Georg Hager, Jesper Träff, and Sascha Hunold
- **Help with HPL run**
  - Gilles Fourestier (CSCS) and Massimiliano Fatica (NVIDIA)

[1]: http://spcl.inf.ethz.ch/Research/Performance/LibLSB/