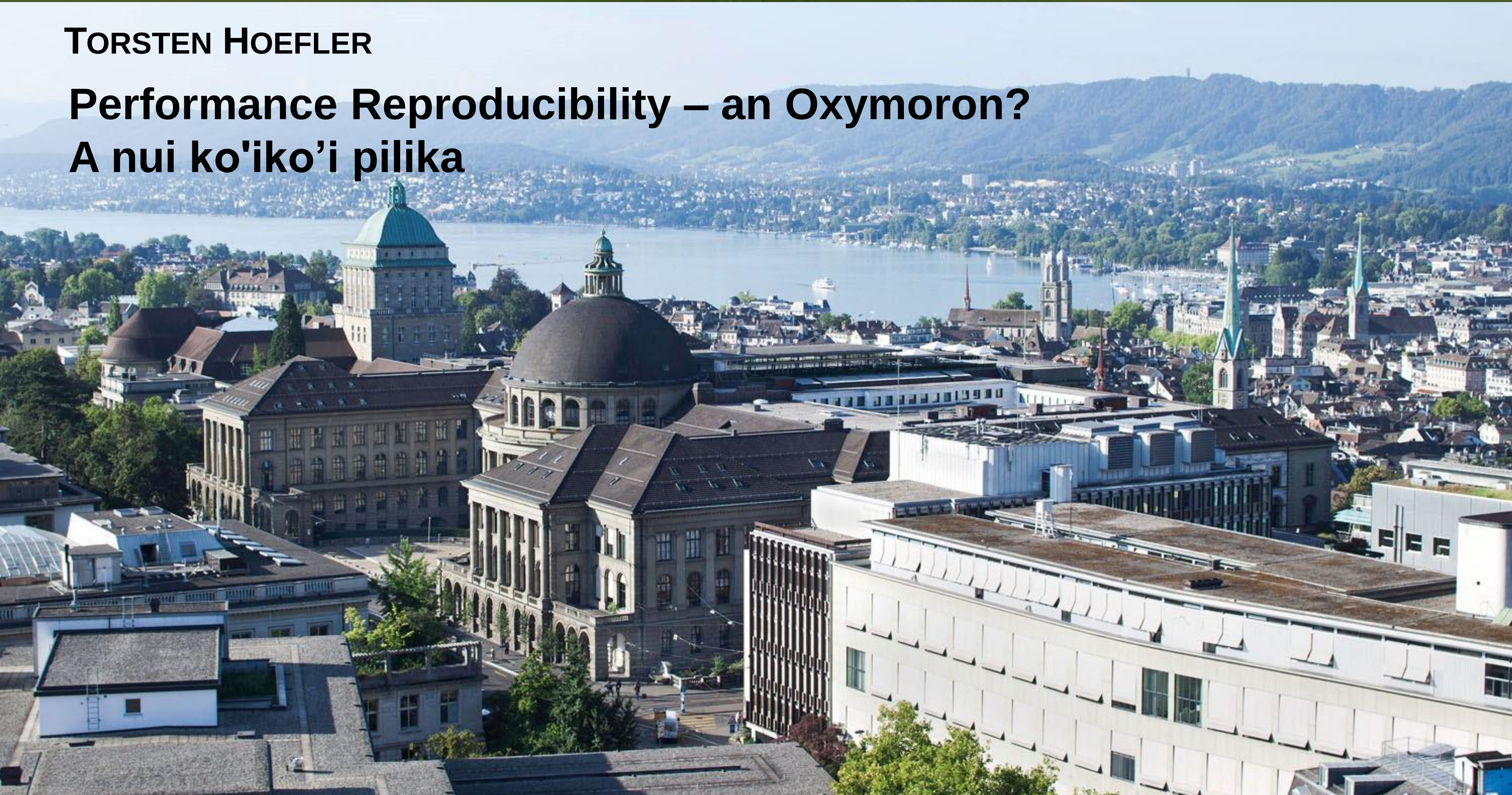
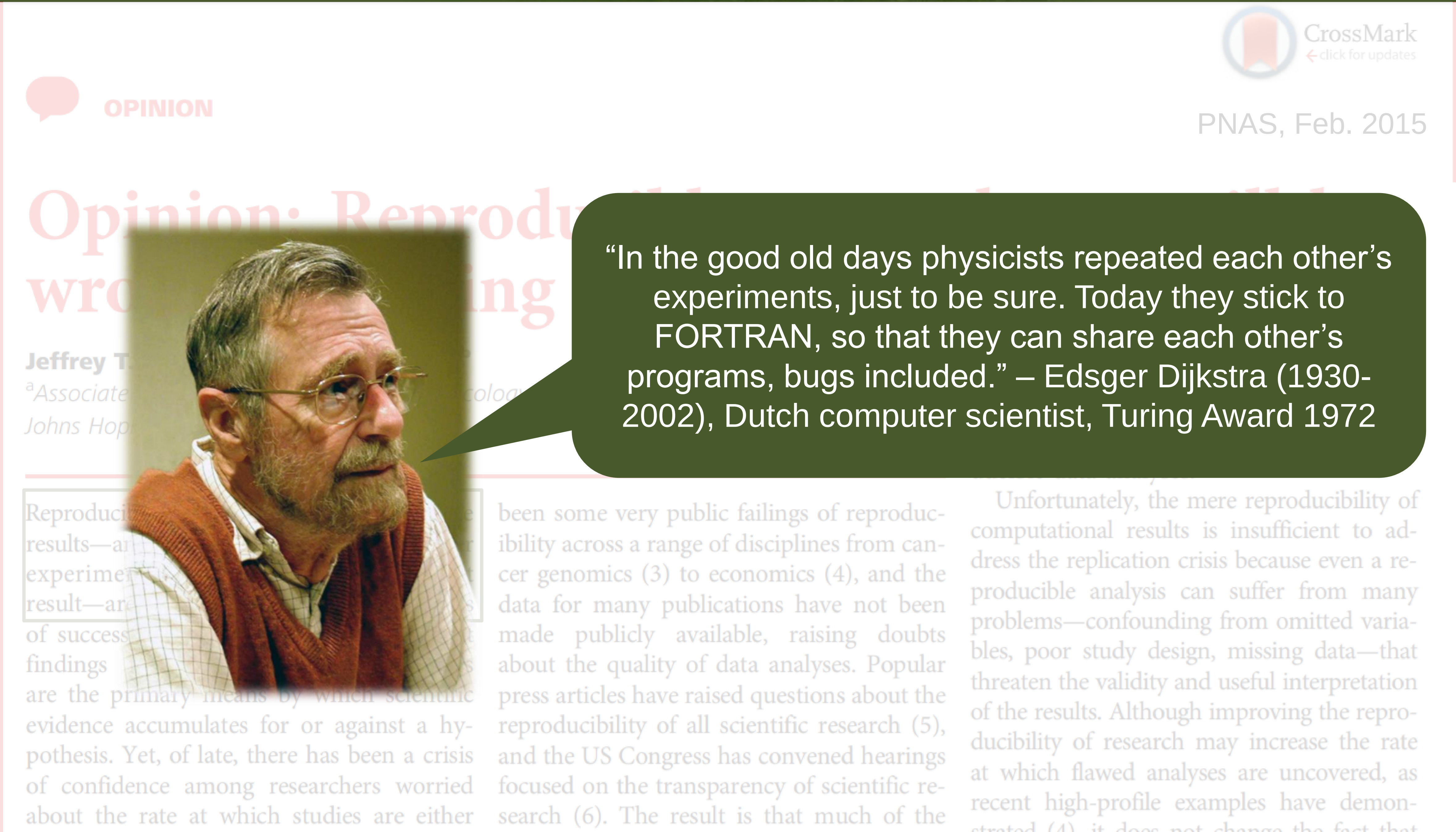


TORSTEN HOEFLER

Performance Reproducibility – an Oxymoron?
A nui ko'iko’i pilika

PNAS, Feb. 2015



OPINION

CrossMark
click for updates

OPINION

Opinion: Reprodu
wro ing

Jeffrey T
[a]Associate colog
Johns Hop

Reproduci
results—ar
experime
result—ar
of success
findings
are the primary means by which scientific
evidence accumulates for or against a hy-
pothesis. Yet, of late, there has been a crisis
of confidence among researchers worried
about the rate at which studies are either

been some very public failings of reproduc-
ibility across a range of disciplines from can-
cer genomics (3) to economics (4), and the
data for many publications have not been
made publicly available, raising doubts
about the quality of data analyses. Popular
press articles have raised questions about the
reproducibility of all scientific research (5),
and the US Congress has convened hearings
focused on the transparency of scientific re-
search (6). The result is that much of the

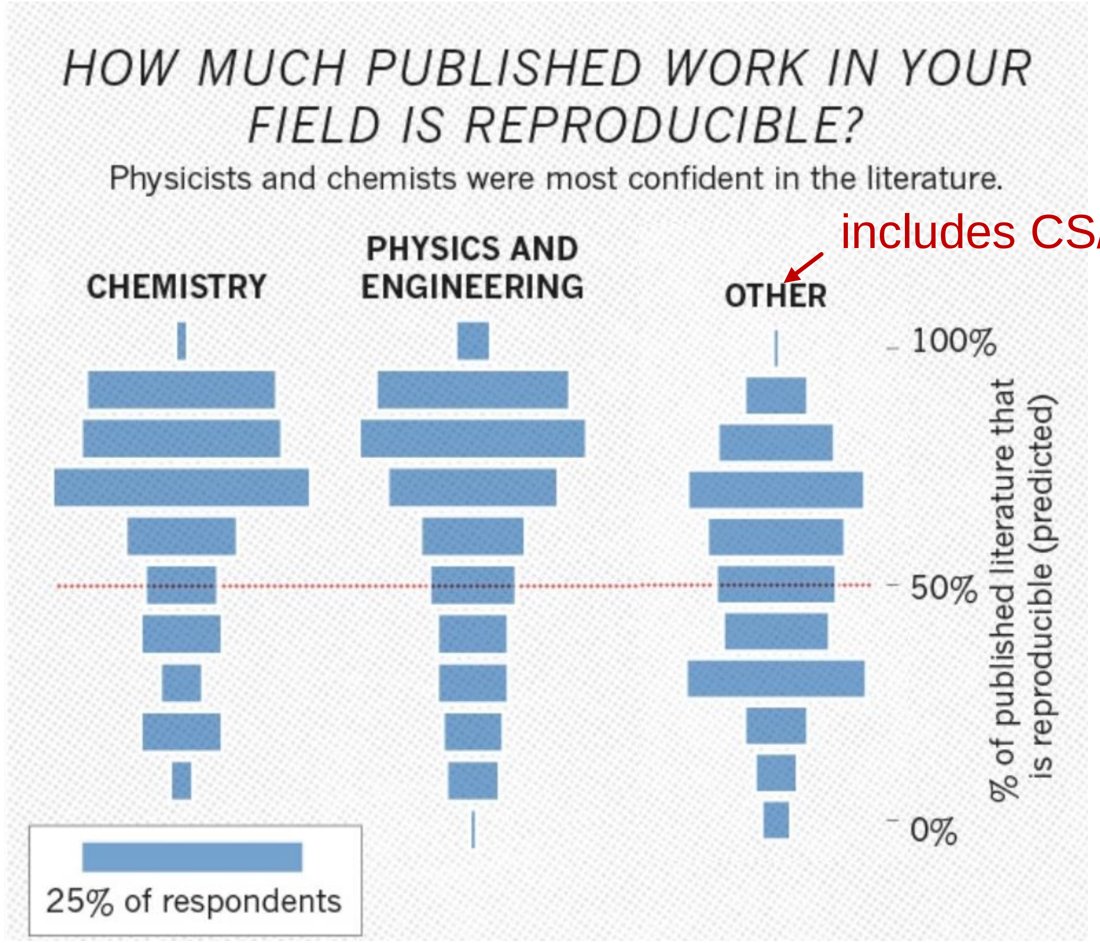
Unfortunately, the mere reproducibility of
computational results is insufficient to ad-
dress the replication crisis because even a re-
producible analysis can suffer from many
problems—confounding from omitted varia-
bles, poor study design, missing data—that
threaten the validity and useful interpretation
of the results. Although improving the repro-
ducibility of research may increase the rate
at which flawed analyses are uncovered, as
recent high-profile examples have demon-

"In the good old days physicists repeated each other's experiments, just to be sure. Today they stick to FORTRAN, so that they can share each other's programs, bugs included." – Edsger Dijkstra (1930-2002), Dutch computer scientist, Turing Award 1972

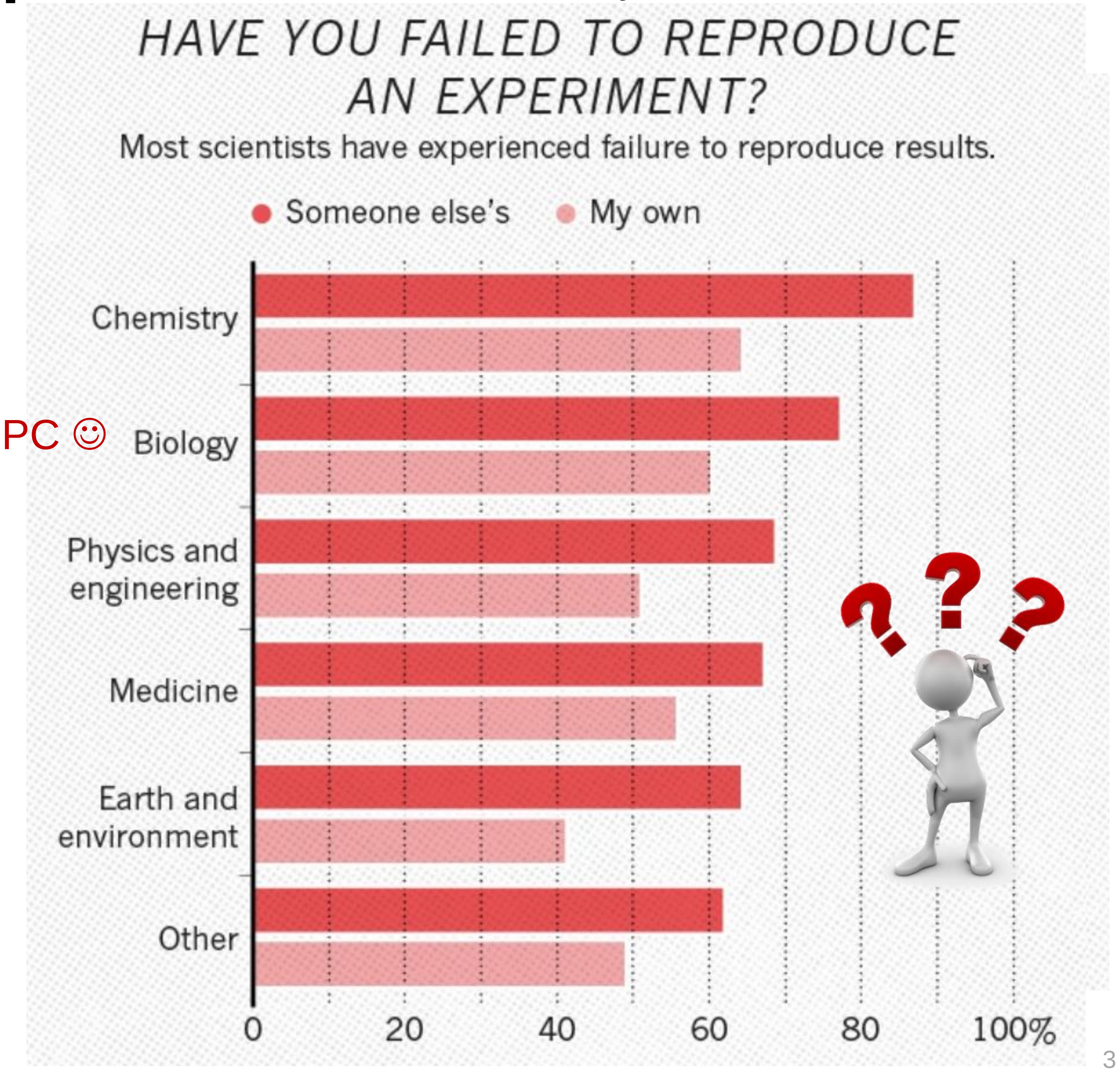
Reproducibility and replicability?

Nature, May 2016

- **Reproducibility – get the exact results**
- **Replicability – repeat the effect/insight**



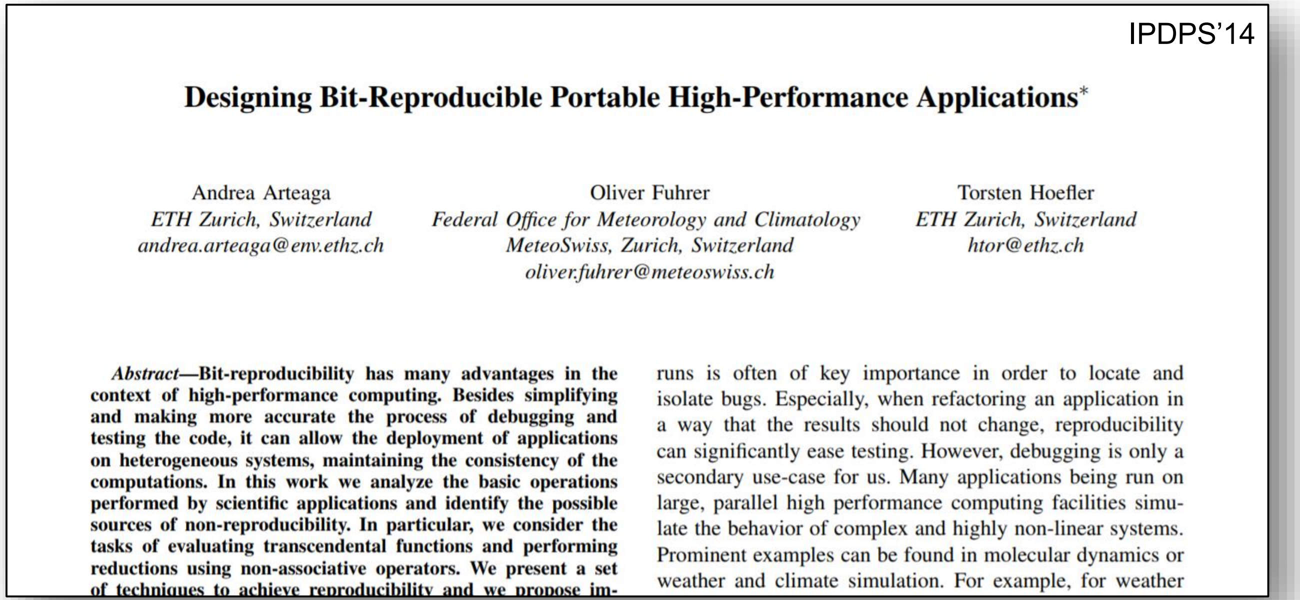
includes CS/HPC ☺



Functional reproducibility is relatively simple – release the code!

Single-threaded, if you don't care much about performance

Gets a bit more complex when you share parallel codes (IEEE 754 is not associative)



IPDPS'14

Designing Bit-Reproducible Portable High-Performance Applications*

Andrea Arteaga
ETH Zurich, Switzerland
andrea.arteaga@env.ethz.ch

Oliver Fuhrer
Federal Office for Meteorology and Climatology
MeteoSwiss, Zurich, Switzerland
oliver.fuhrer@meteoswiss.ch

Torsten Hoefler
ETH Zurich, Switzerland
htor@ethz.ch

***Abstract*—Bit-reproducibility has many advantages in the context of high-performance computing. Besides simplifying and making more accurate the process of debugging and testing the code, it can allow the deployment of applications on heterogeneous systems, maintaining the consistency of the computations. In this work we analyze the basic operations performed by scientific applications and identify the possible sources of non-reproducibility. In particular, we consider the tasks of evaluating transcendental functions and performing reductions using non-associative operators. We present a set of techniques to achieve reproducibility and we propose im-**

runs is often of key importance in order to locate and isolate bugs. Especially, when refactoring an application in a way that the results should not change, reproducibility can significantly ease testing. However, debugging is only a secondary use-case for us. Many applications being run on large, parallel high performance computing facilities simulate the behavior of complex and highly non-linear systems. Prominent examples can be found in molecular dynamics or weather and climate simulation. For example, for weather

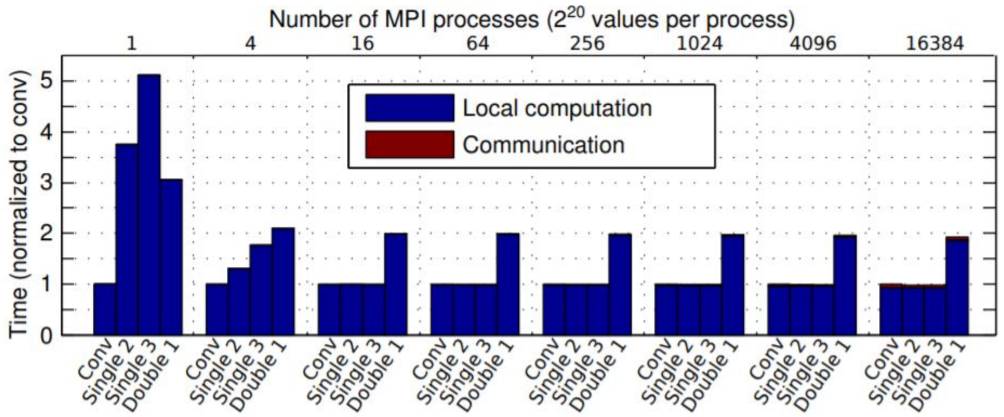
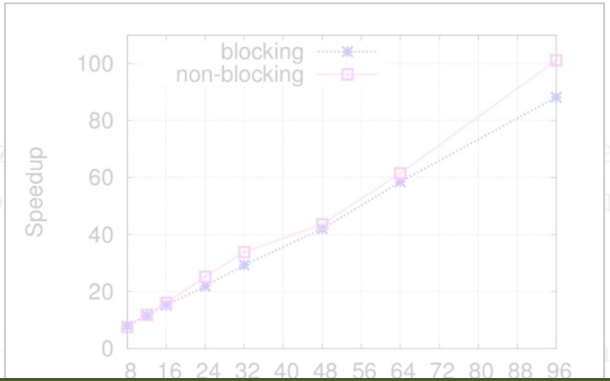


Figure 8. Performance comparison of conventional reduction performed with MKL (*Conv*), single-sweep reduction with two levels (*Single2*), with three levels (*Single3*) and double-sweep reduction with 1 level (*Double 1*) on varying number of processes, each owning 2^{20} double-precision values,

But what if performance is your science result?

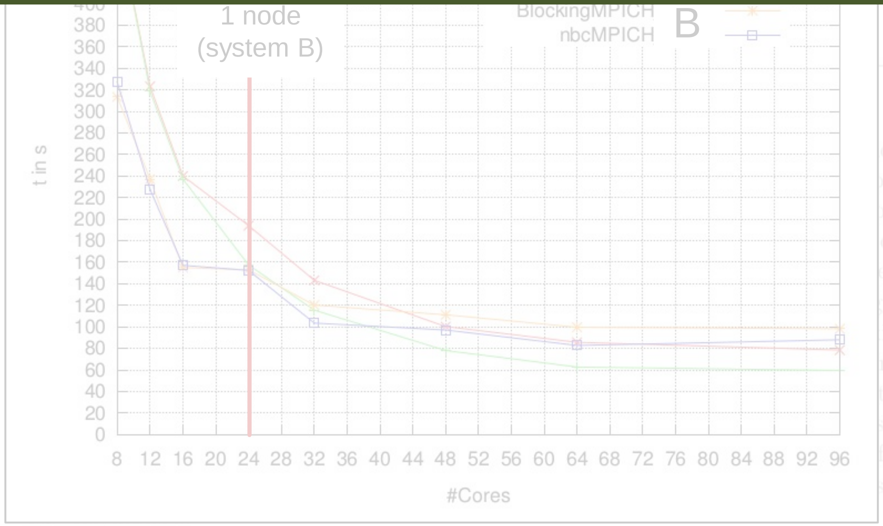
(2006)



Deadline in a bit more than 24 hours, no extensions ☺

- **Original findings:**
  - If carefully tuned, NBC speed up a 3D solver
    Full code published
  - 800^3 domain – 4 GB (distributed) array

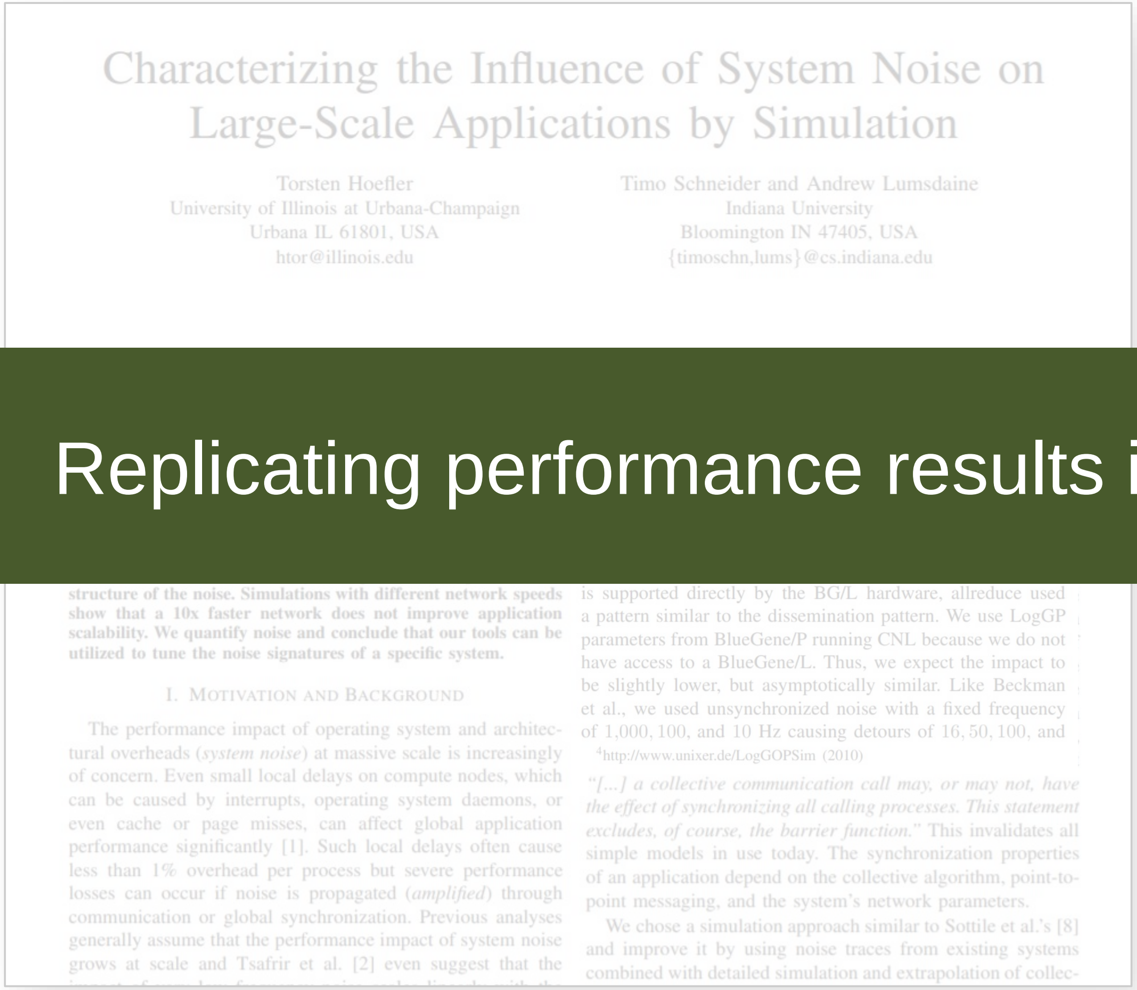
Reproducing performance results is hard! Is it even possible?



- **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)."

My own replication result



Characterizing the Influence of System Noise on Large-Scale Applications by Simulation

Torsten Hoefler
University of Illinois at Urbana-Champaign
Urbana IL 61801, USA
htor@illinois.edu

Timo Schneider and Andrew Lumsdaine
Indiana University
Bloomington IN 47405, USA
{timoschn,lums}@cs.indiana.edu

structure of the noise. Simulations with different network speeds show that a 10x faster network does not improve application scalability. We quantify noise and conclude that our tools can be utilized to tune the noise signatures of a specific system.

I. Motivation and Background

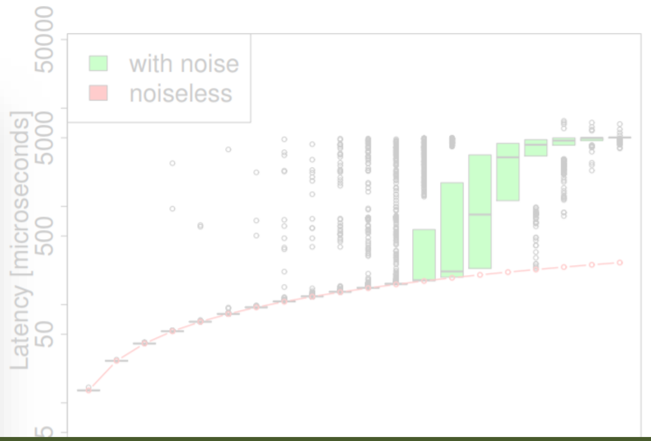
The performance impact of operating system and architectural overheads (*system noise*) at massive scale is increasingly of concern. Even small local delays on compute nodes, which can be caused by interrupts, operating system daemons, or even cache or page misses, can affect global application performance significantly [1]. Such local delays often cause less than 1% overhead per process but severe performance losses can occur if noise is propagated (*amplified*) through communication or global synchronization. Previous analyses generally assume that the performance impact of system noise grows at scale and Tsafrir et al. [2] even suggest that the

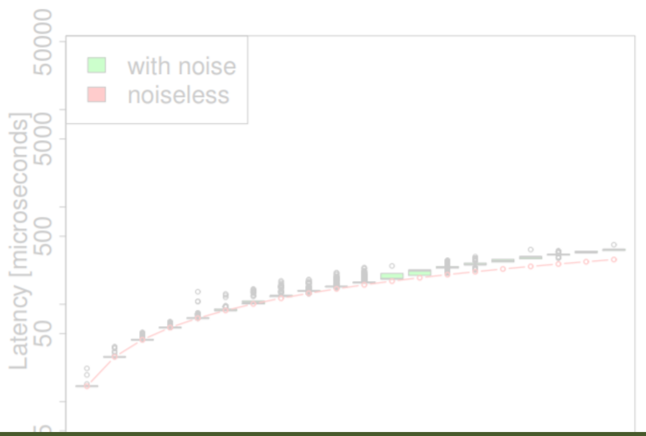
is supported directly by the BG/L hardware, allreduce used a pattern similar to the dissemination pattern. We use LogGP parameters from BlueGene/P running CNL because we do not have access to a BlueGene/L. Thus, we expect the impact to be slightly lower, but asymptotically similar. Like Beckman et al., we used unsynchronized noise with a fixed frequency of 1,000, 100, and 10 Hz causing detours of 16, 50, 100, and

[4]http://www.unixer.de/LogGOPSim (2010)

"[...] a collective communication call may, or may not, have the effect of synchronizing all calling processes. This statement excludes, of course, the barrier function." This invalidates all simple models in use today. The synchronization properties of an application depend on the collective algorithm, point-to-point messaging, and the system's network parameters.

We chose a simulation approach similar to Sottile et al.'s [8] and improve it by using noise traces from existing systems combined with detailed simulation and extrapolation of collec-





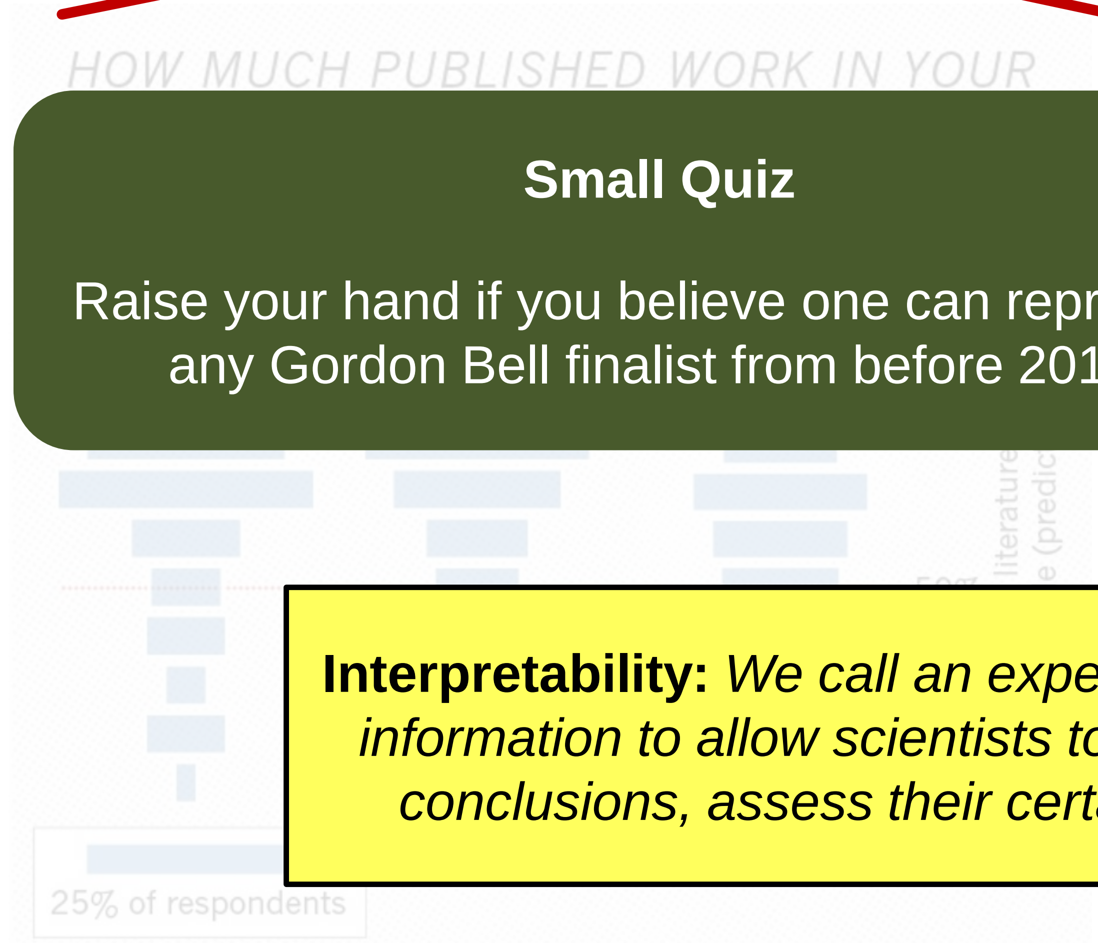
Replicating performance results is possible but rare! Make it the default?

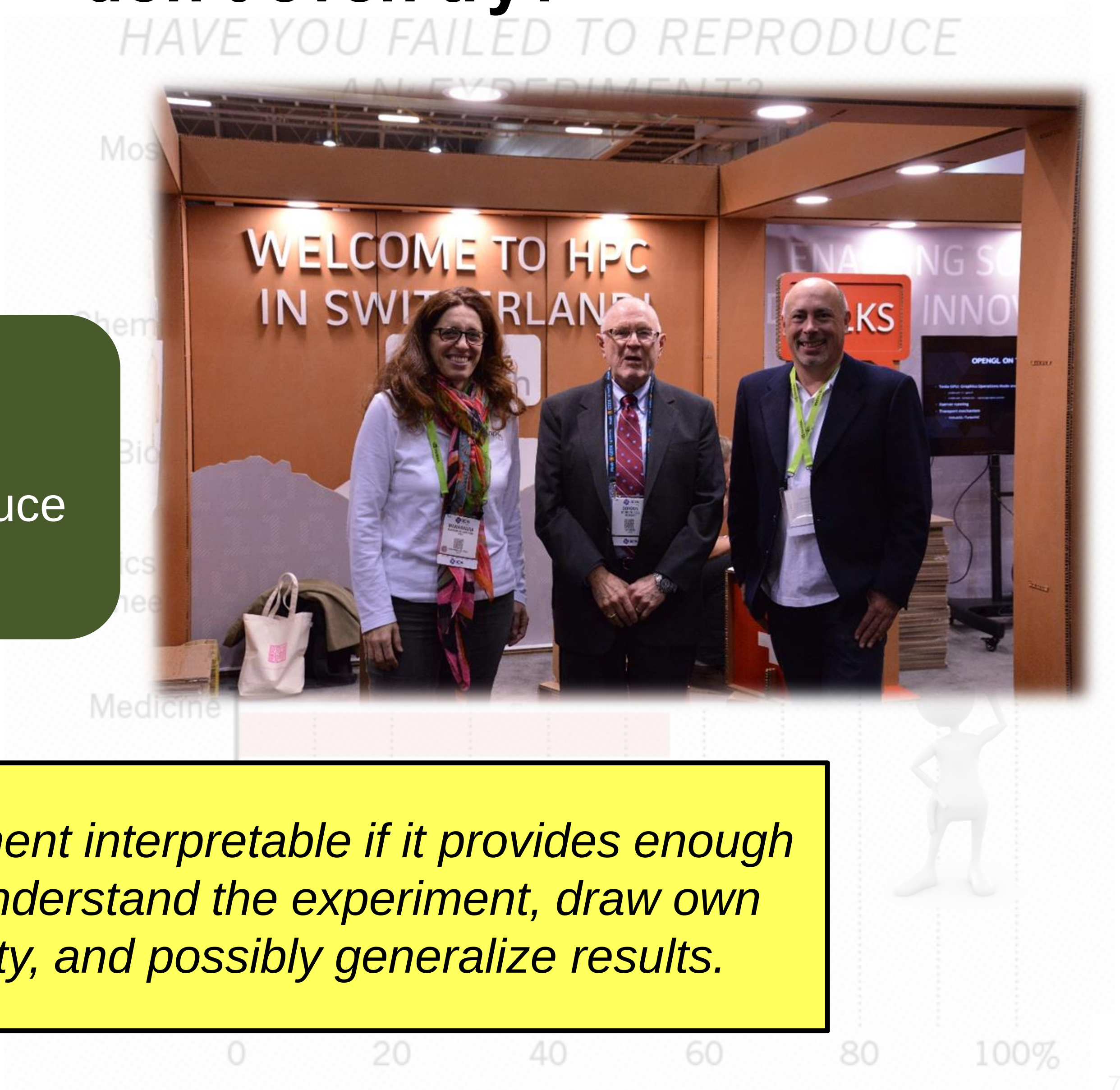
results from Ferreira, Bridges, Brightwell
as well as Beckman et al. both two years earlier on different machines

SC10
BEST PAPER
Characterizing the Influence of System Noise on Large-Scale Applications by Simulation
Torsten Hoefler, Timo Schneider and Andrew Lumsdaine

HPC Performance reproducibility – don't even try?

- ~~**Reproducibility – get the exact results**~~
- ~~**Replicability – repeat the effect/insight**~~





Small Quiz

Raise your hand if you believe one can reproduce any Gordon Bell finalist from before 2013!

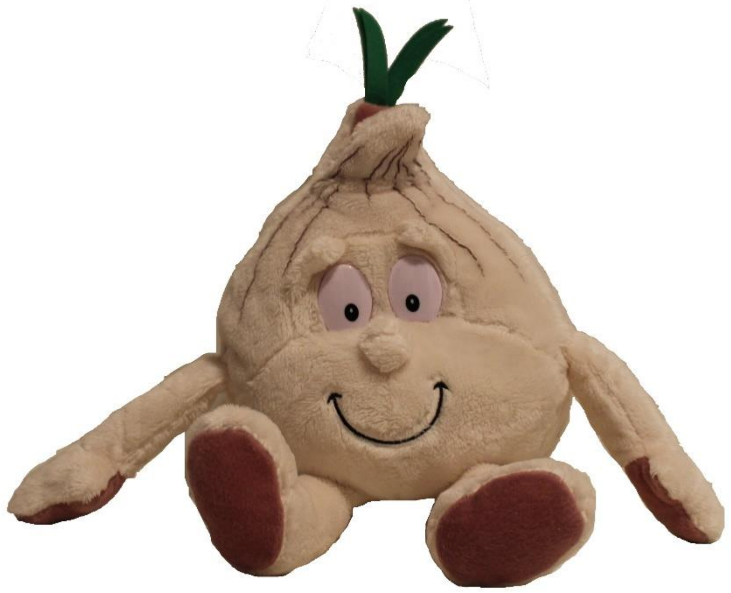
Interpretability: *We call an experiment interpretable if it provides enough information to allow scientists to understand the experiment, draw own conclusions, assess their certainty, and possibly generalize results.*

How does Garth measure and report performance?

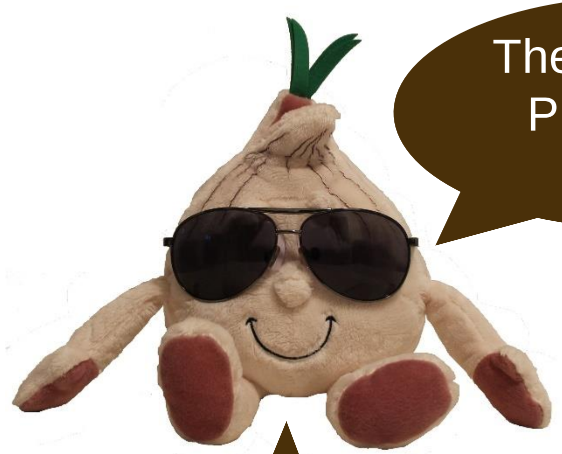
- **We are interested in High Performance Computing**
  - We (want to) see it as a science – reproducing experiments is a major pillar of the scientific method

- **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?"
  - …

- **I asked: "How are they answered in the field today?"**
  - "Experience"
  - "Gut feeling"
  - "Clever observation"
  - …

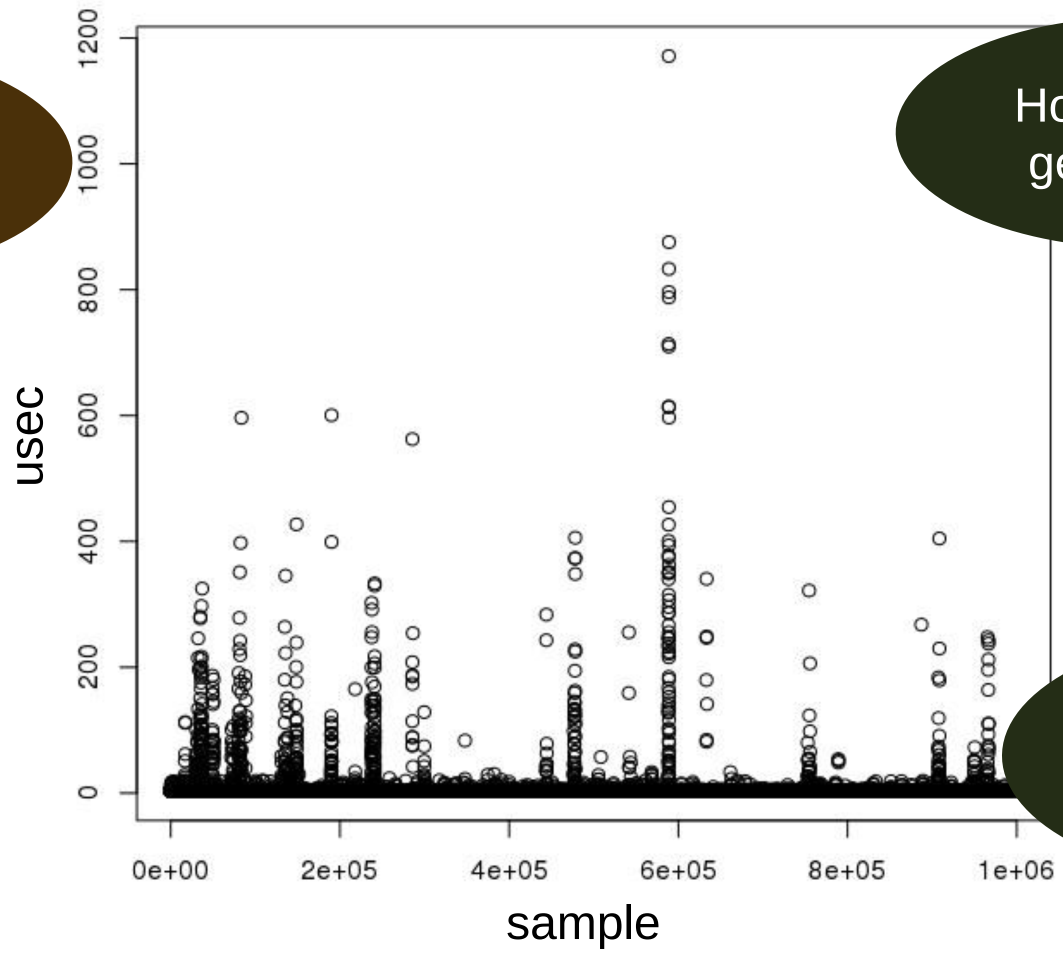


The simplest networking question: ping pong latency!



The latency of Piz Dora is 1.77us!

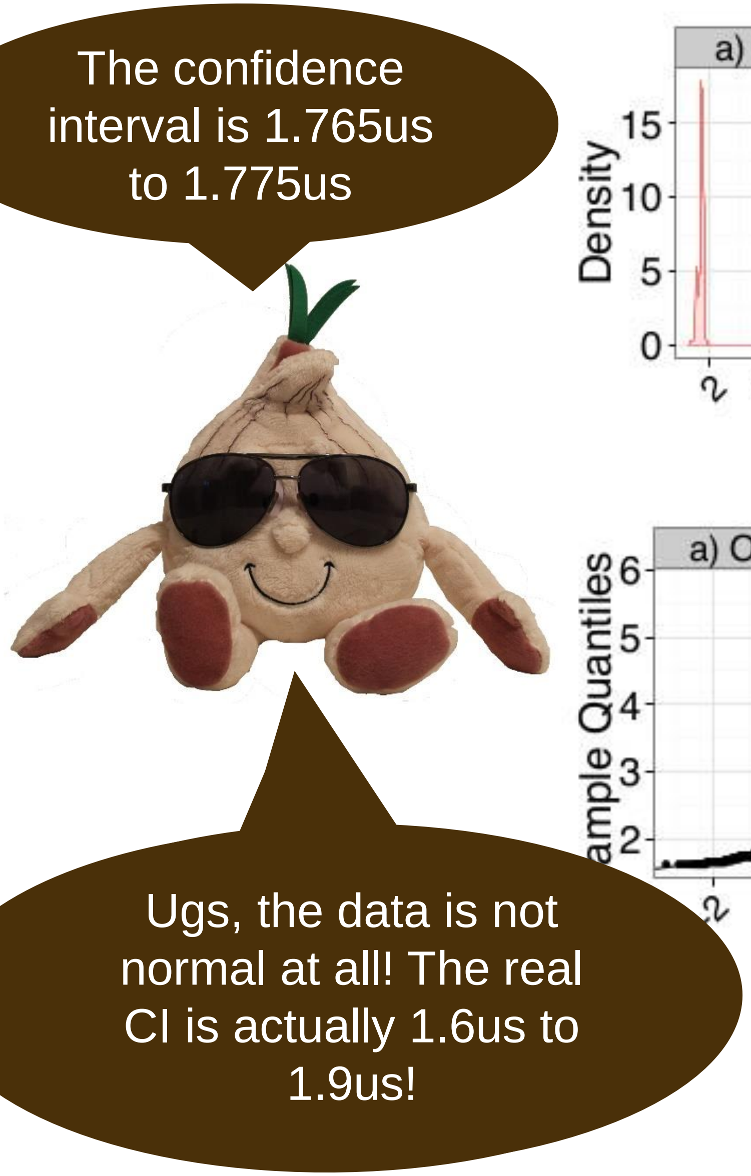
I averaged 10^6 tests, it must be right!

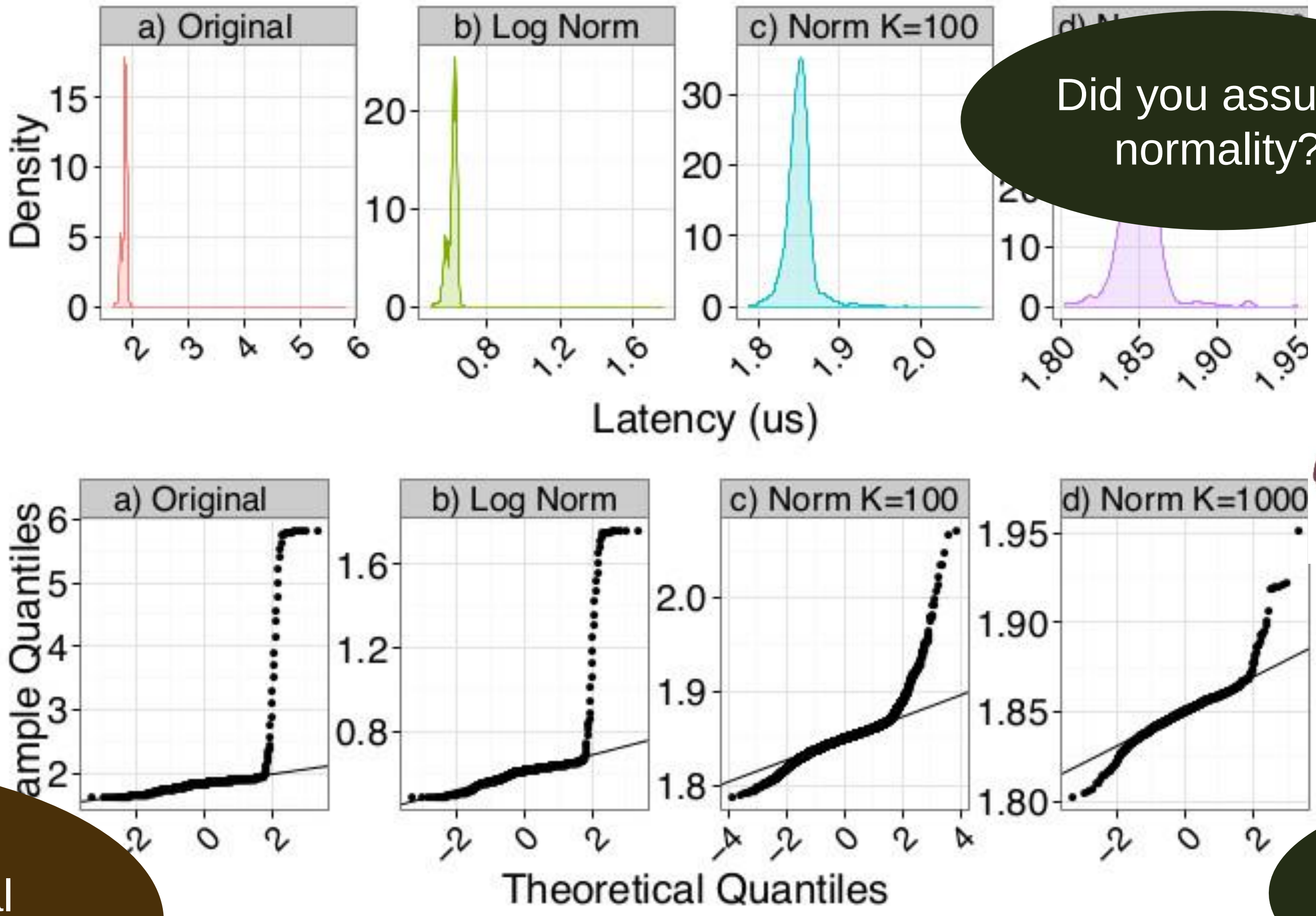


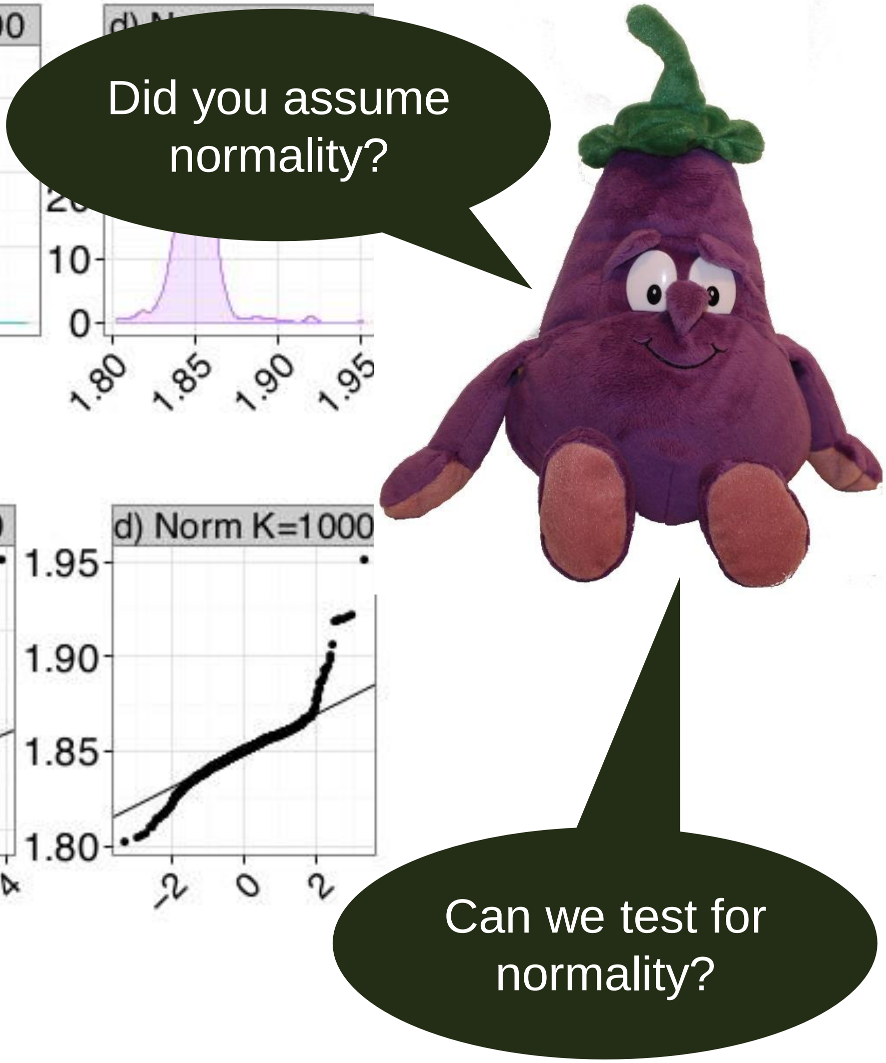
How did you get to this?

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

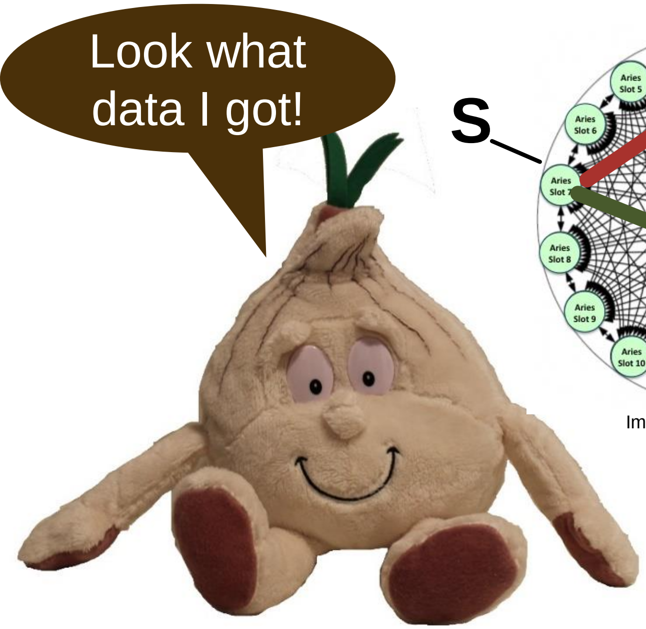
Thou shalt not trust your average textbook!


The confidence interval is 1.765us to 1.775us
Ugs, the data is not normal at all! The real CI is actually 1.6us to 1.9us!





Thou shalt not trust your system!



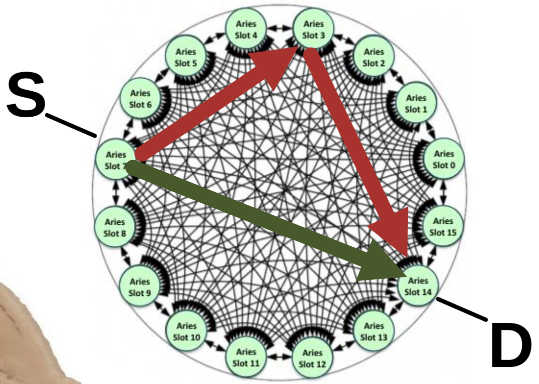
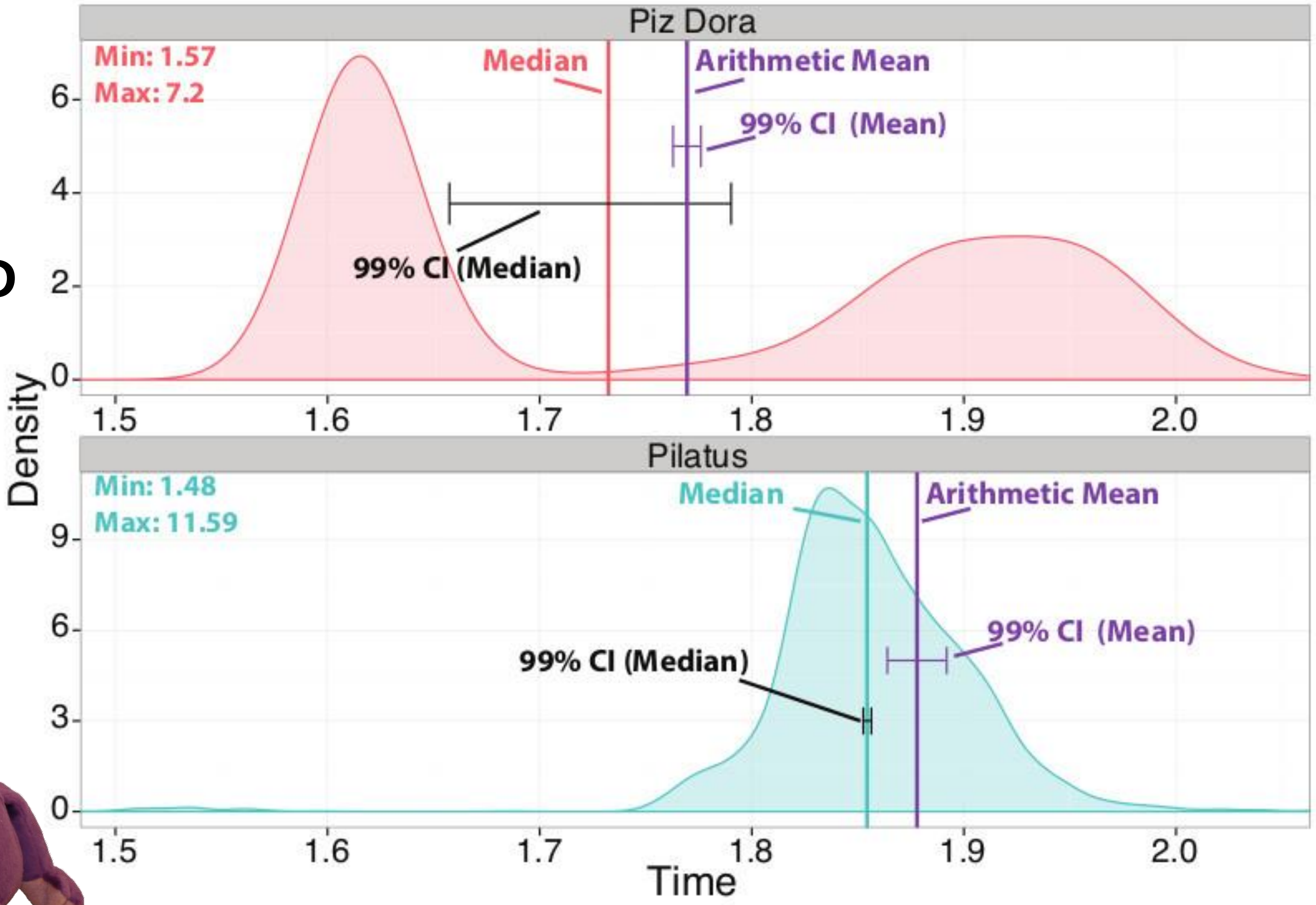
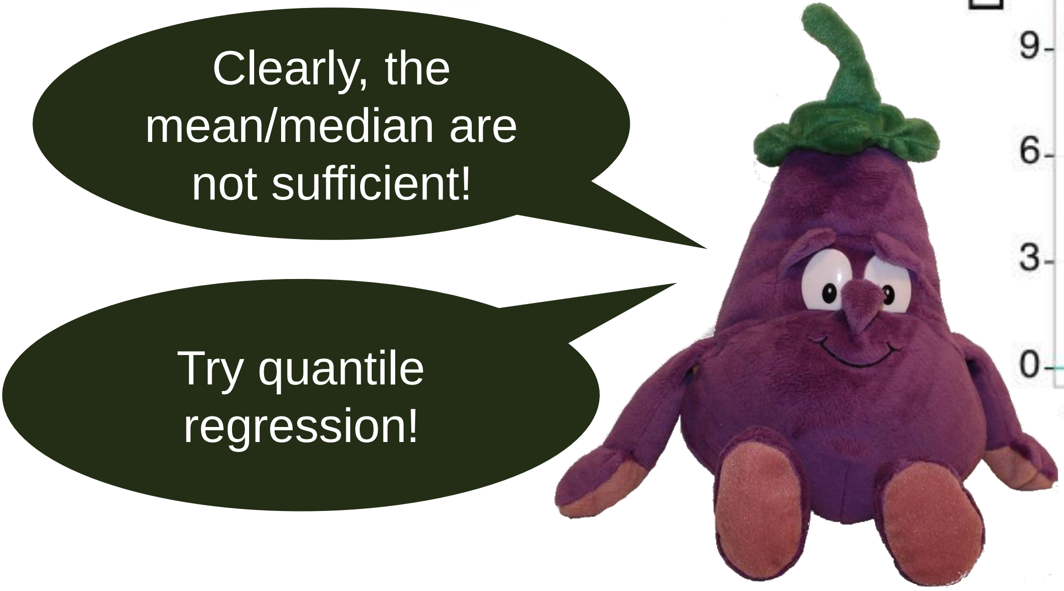
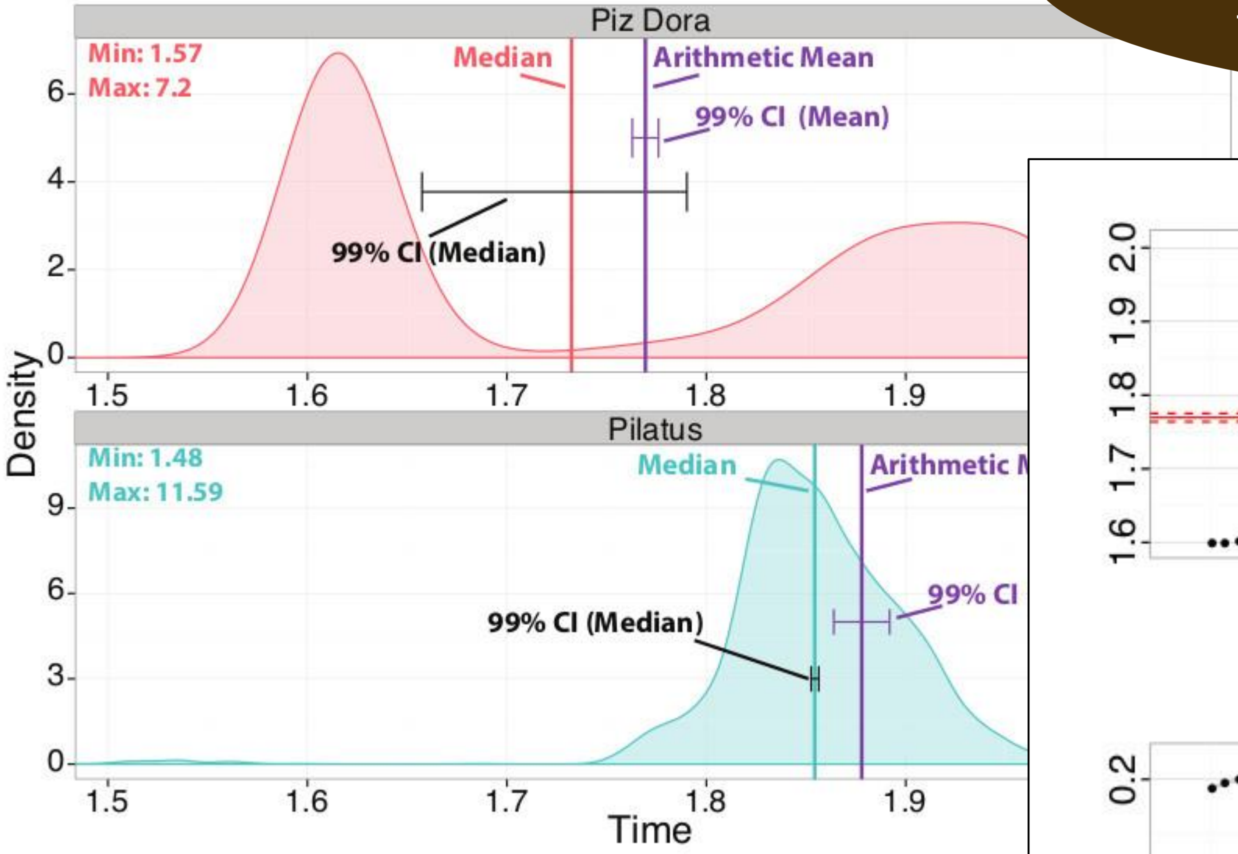


Image credit: nersc.gov

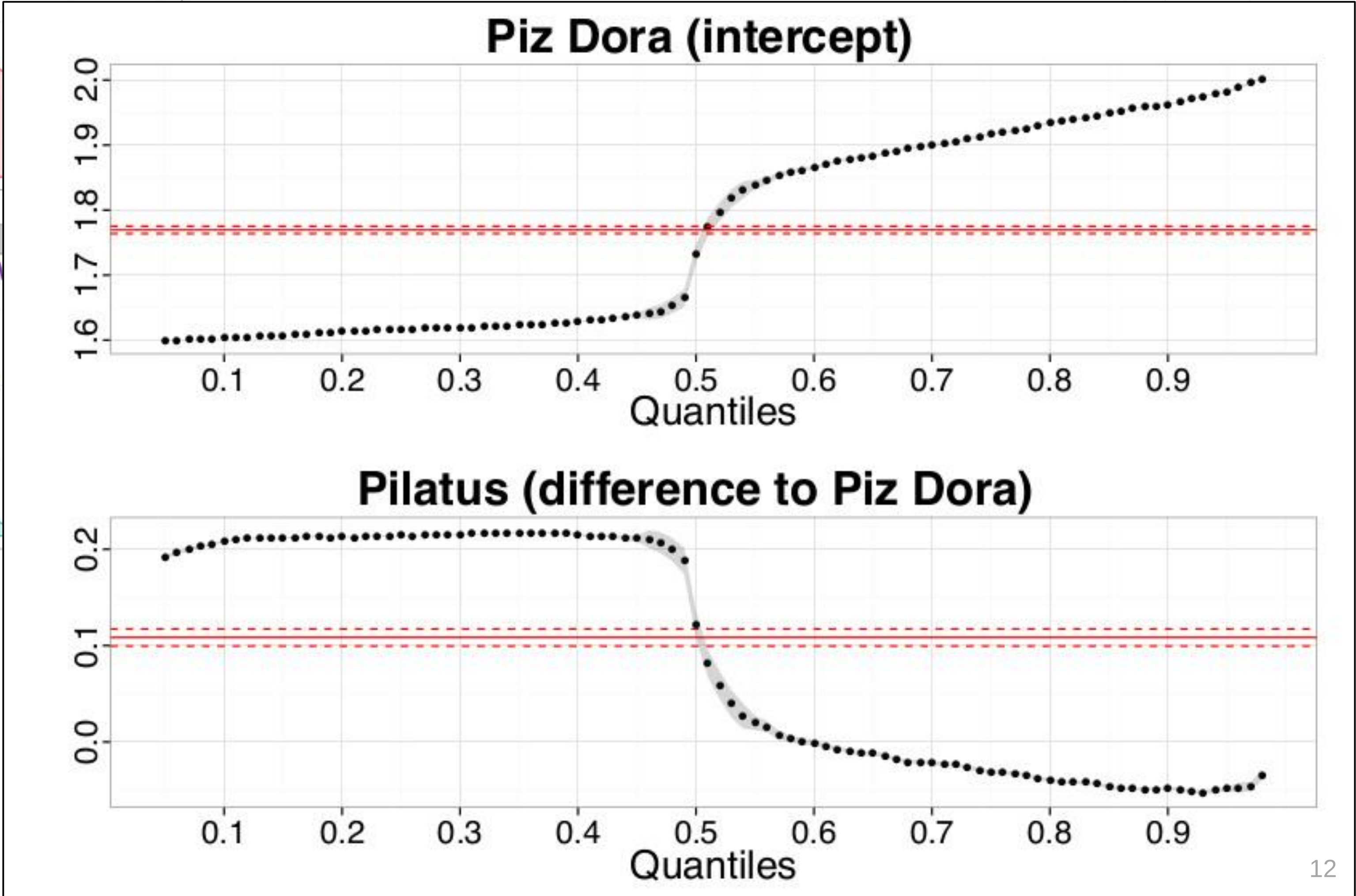




Quantile regression

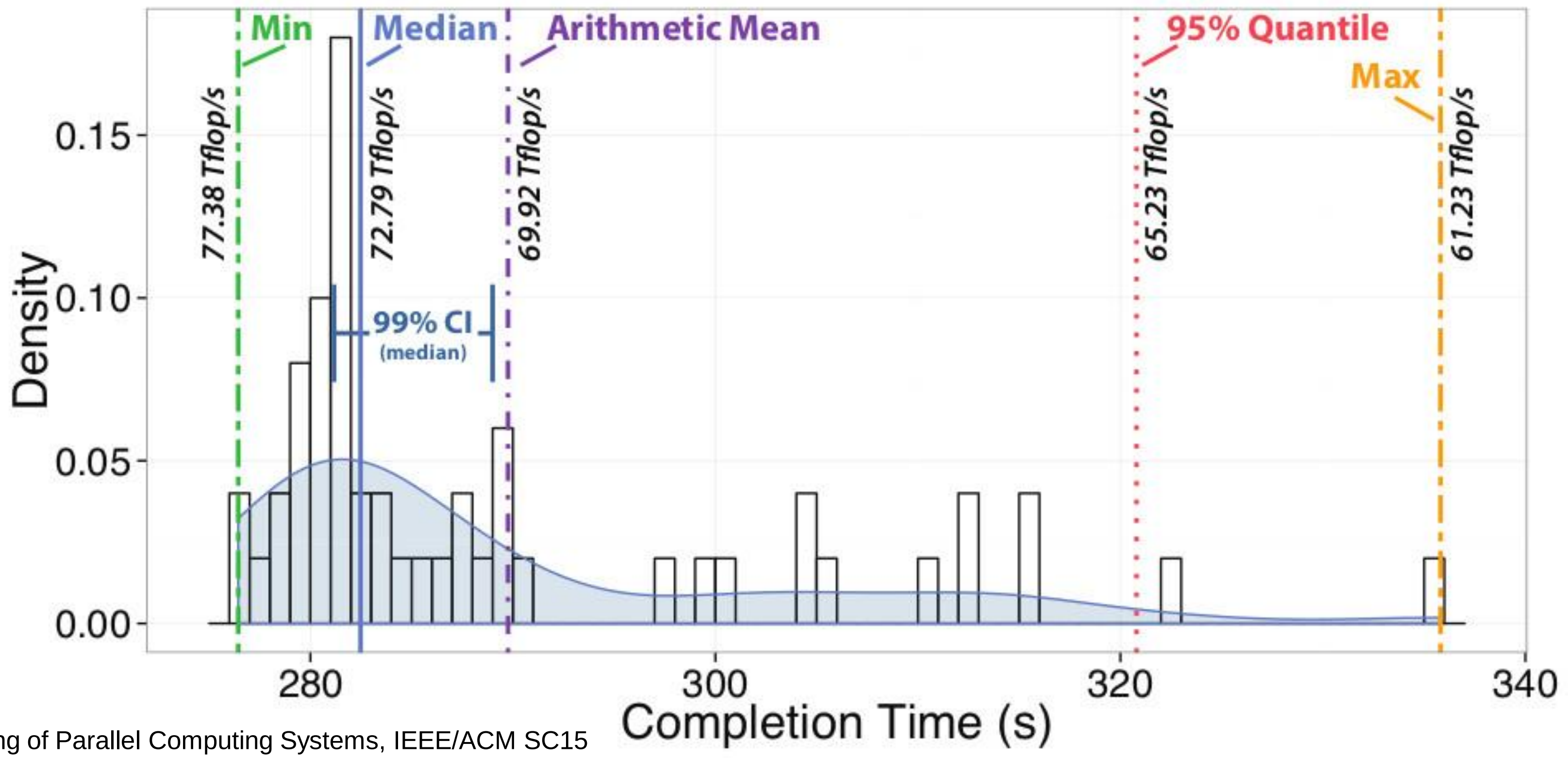


Wow, so Pilatus is better for (worst-case) latency-critical workloads even though Dora is expected to be faster

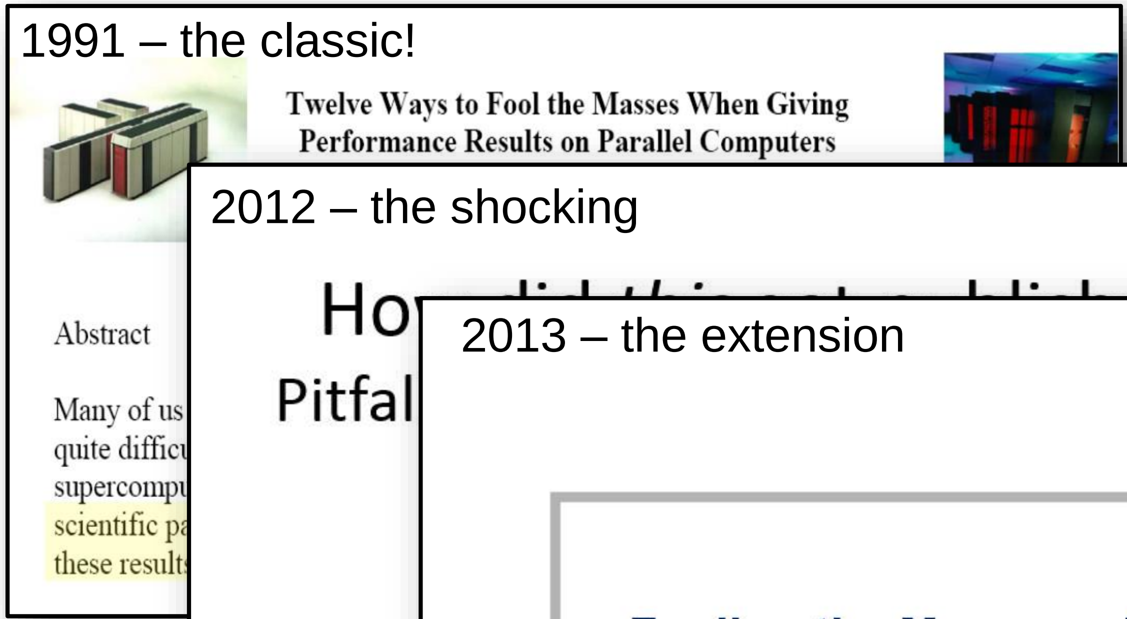


A note on good scientific HPC practice ☺ - HPL

- **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



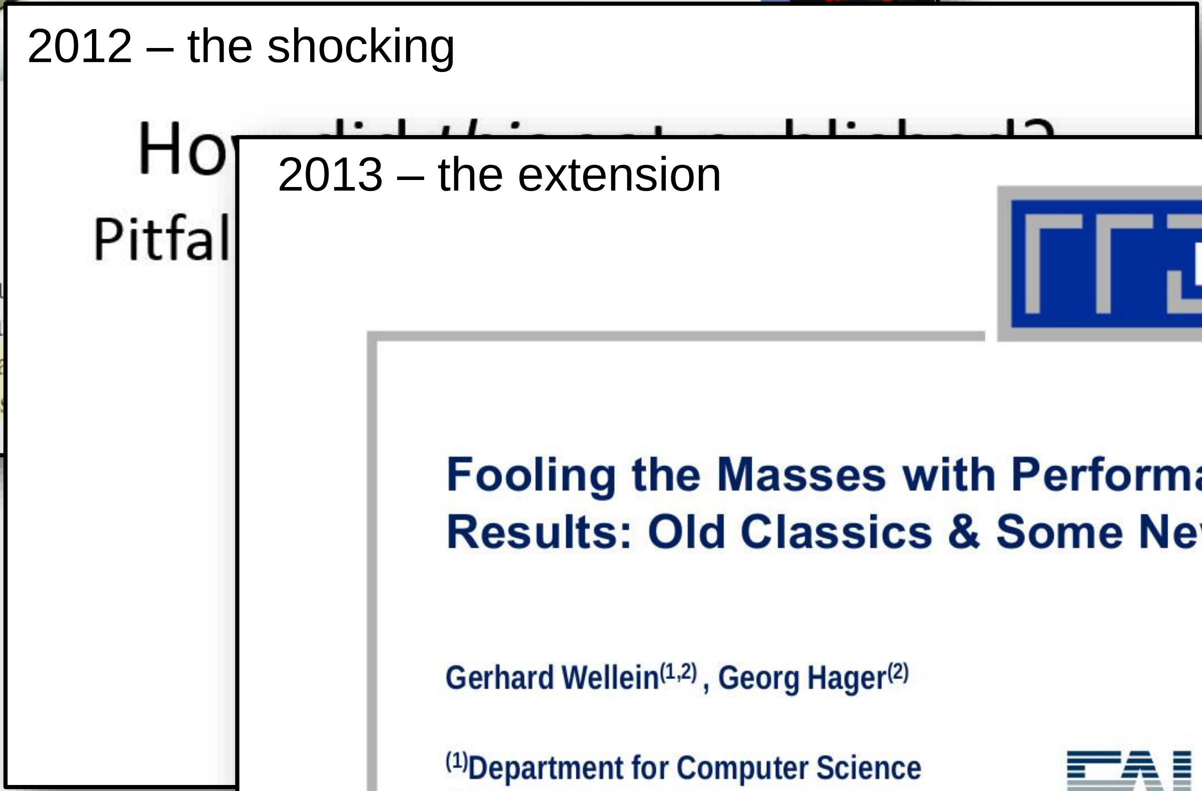
But it's ok, HPC people can laugh about ourselves!


1991 – the classic!

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

Abstract

Many of us
quite difficu
supercompu
scientific pa
these results


2012 – the shocking

How did *this* get published?

Pitfal

2013 – the extension

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

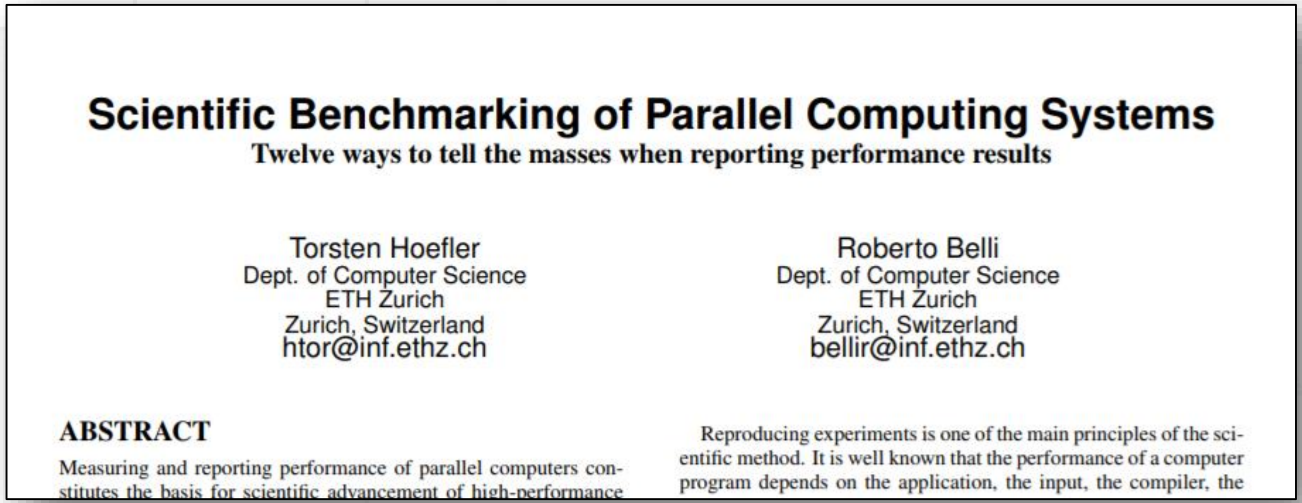
Gerhard Wellein[1,2] , Georg Hager[2]

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

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

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

- **Attempt to emphasize interpretability of performance experiments**
  - Teach some basic statistics
- **The set of 12 rules is not complete**
  - And probably never will be
  - Intended to serve as a solid start
  - Call to the community to extend it



Scientific Benchmarking of Parallel Computing Systems
Twelve ways to tell the masses when reporting performance results

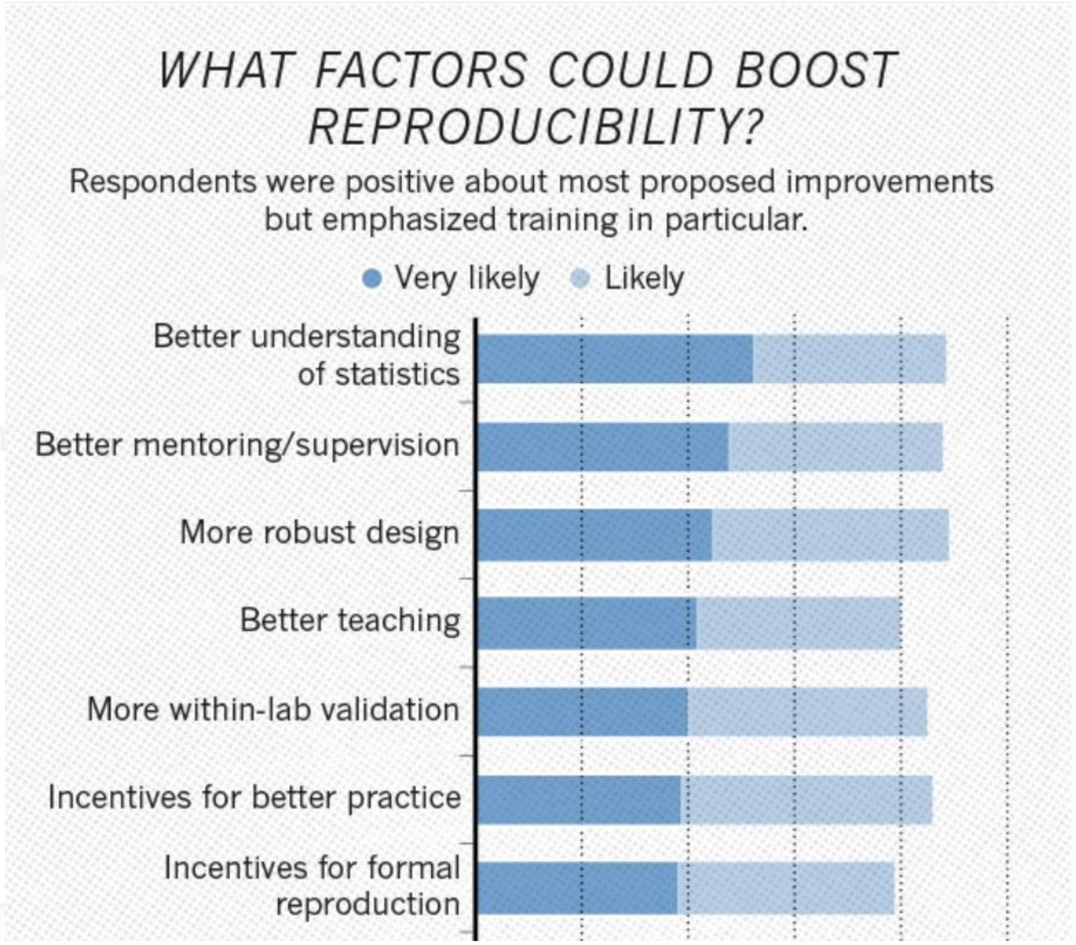
Torsten Hoefler
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
htor@inf.ethz.ch

Roberto Belli
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
bellir@inf.ethz.ch

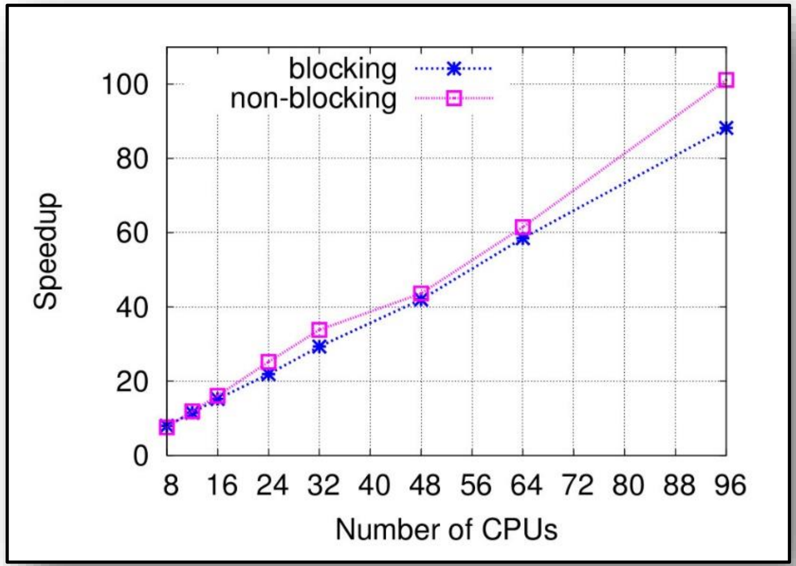
ABSTRACT
Measuring and reporting performance of parallel computers constitutes the basis for scientific advancement of high-performance

Reproducing experiments is one of the main principles of the scientific method. It is well known that the performance of a computer program depends on the application, the input, the compiler, the

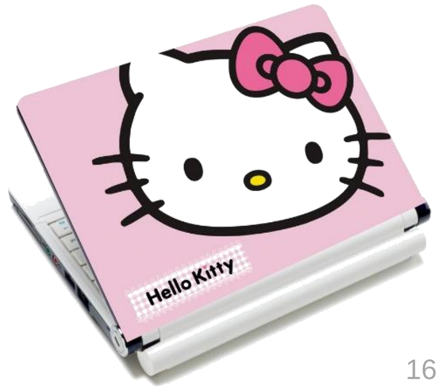
Nature, 2016



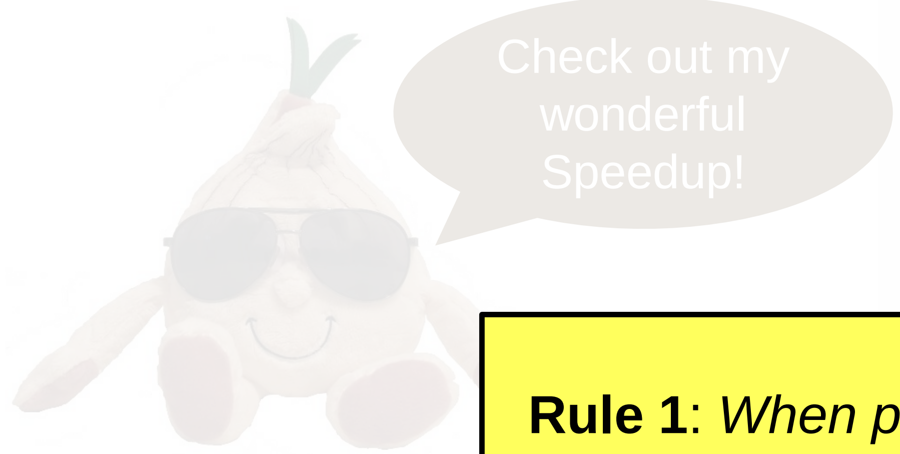
The most common issue: speedup plots

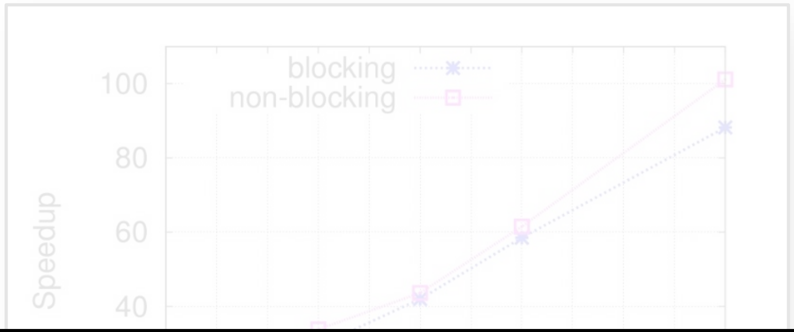


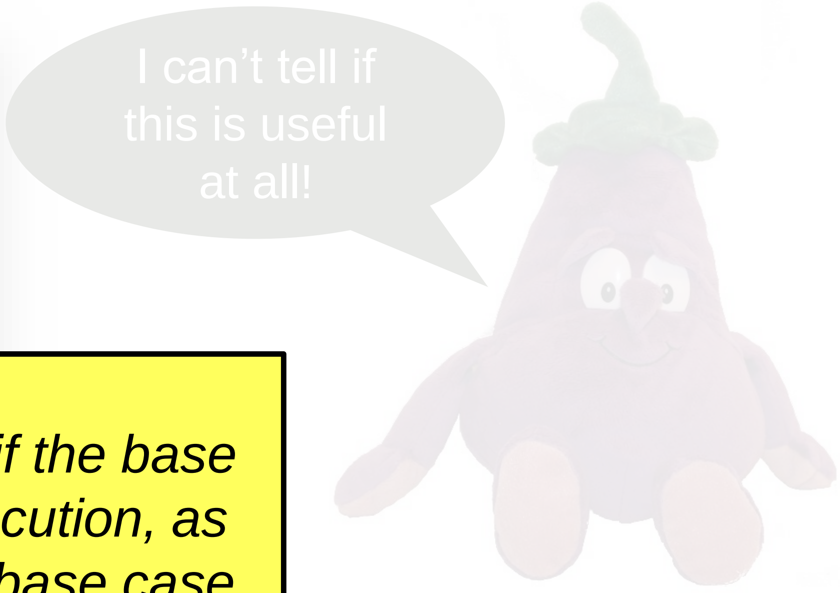
- **Most common and oldest-known issue**
  - First seen 1988 – also included in Bailey's 12 ways
  - 39/120 checked papers reported speedups
    15 (38%) did not specify the base-performance ☹
  - Recently rediscovered in the "big data" universe
    A. Rowstron et al.: Nobody ever got fired for using Hadoop on a cluster, HotCDP 2012
    F. McSherry et al.: Scalability! but at what cost?, HotOS 2015



The most common issue: speedup plots







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

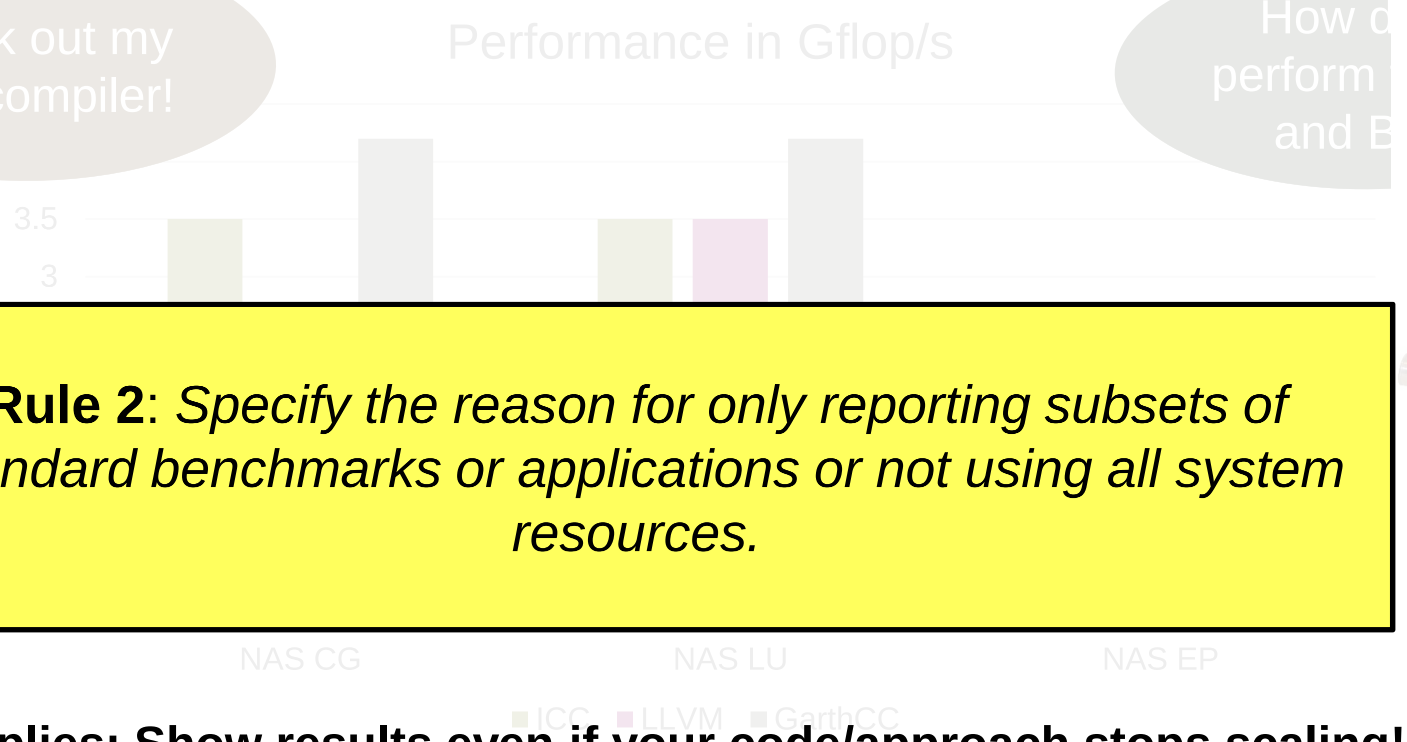
    A. Rowstron et al.: Nobody ever got fired for using Hadoop on a cluster, HotCDP 2012

    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

ck out my
compiler!



How d
perform
and B

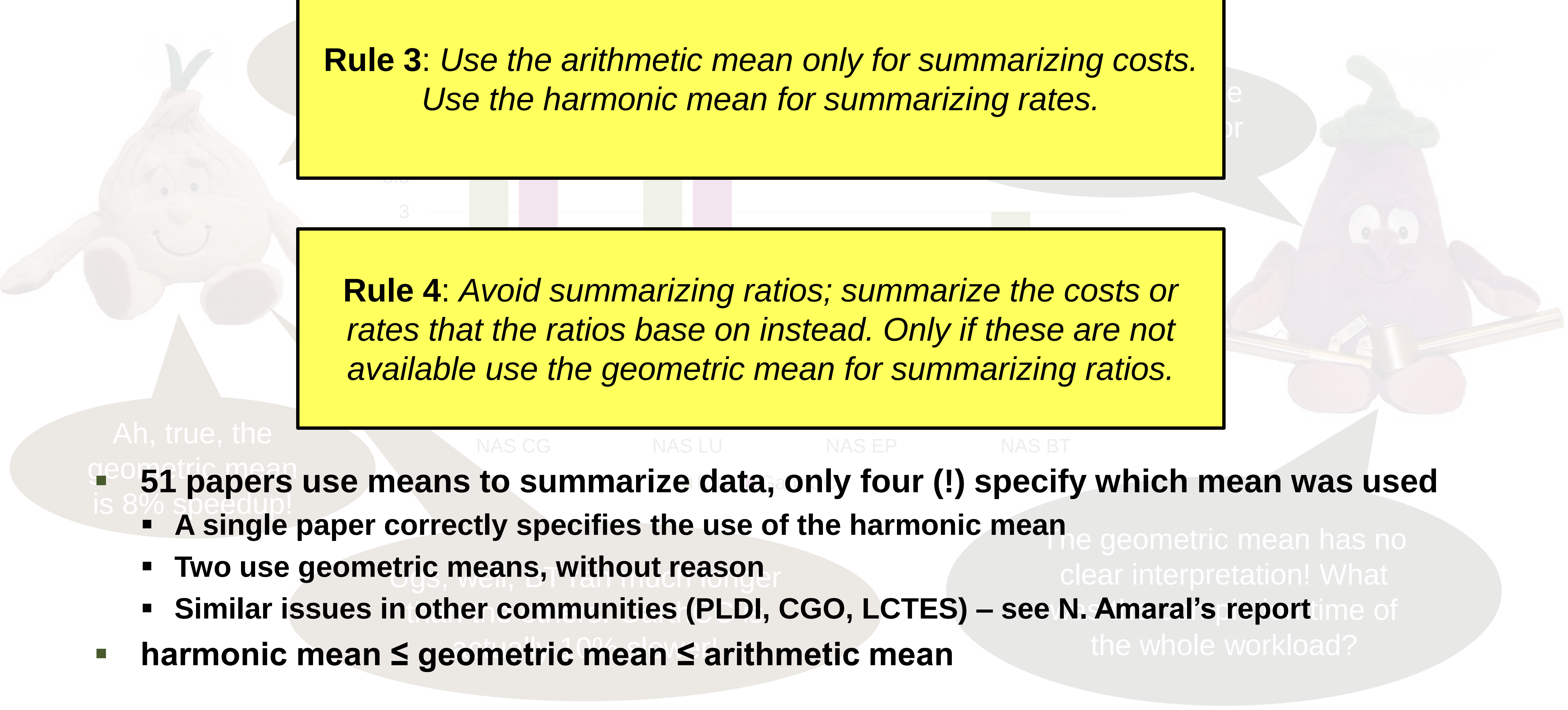
Well, GarthCC segfaulted for FT and was 20% slower for BT.

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!**

spcl.inf.ethz.ch
@spcl_eth

The mean parts of means – or how to summarize data

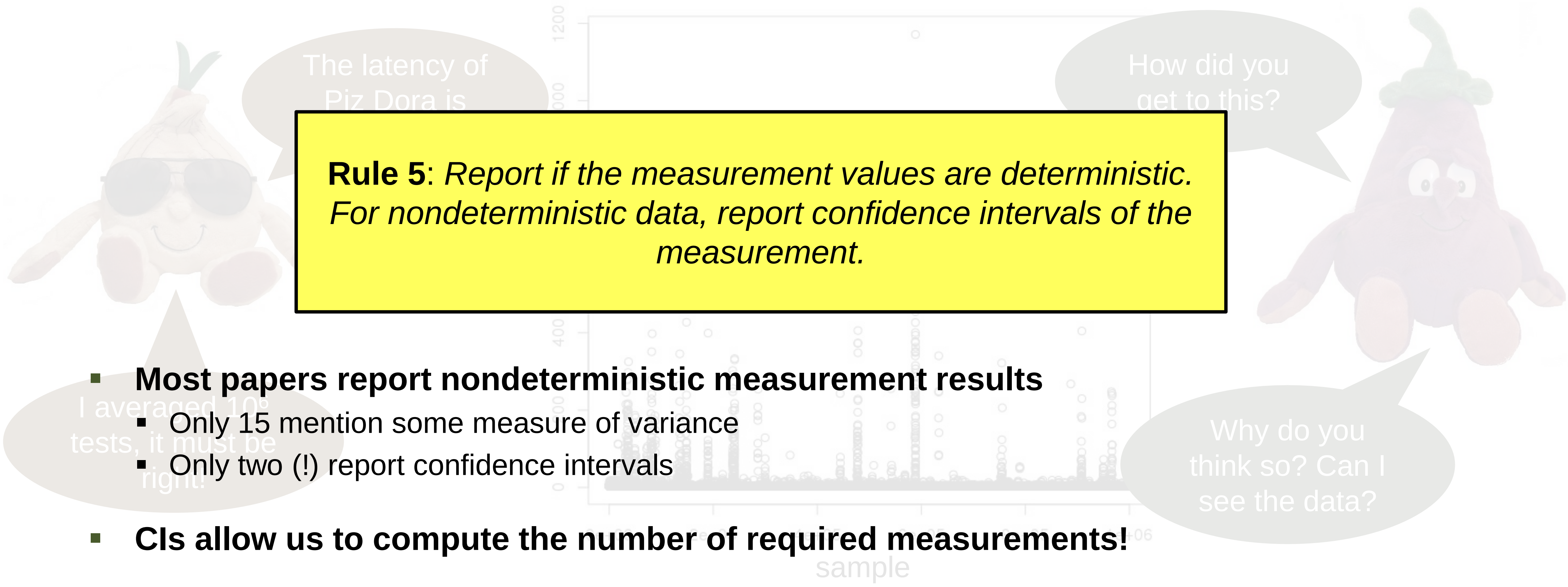

3
NAS CG
NAS LU
NAS EP
NAS BT
Ah, true, the geometric mean is 8% speedup!
Ugs, well, BT ran much longer than the others. SumBC is actually 10% slower!
The geometric mean has no clear interpretation! What was the completion time of the whole workload?

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?
I averaged 10⁶ tests, it must be right!
Why do you think so? Can I see the data?
sample

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

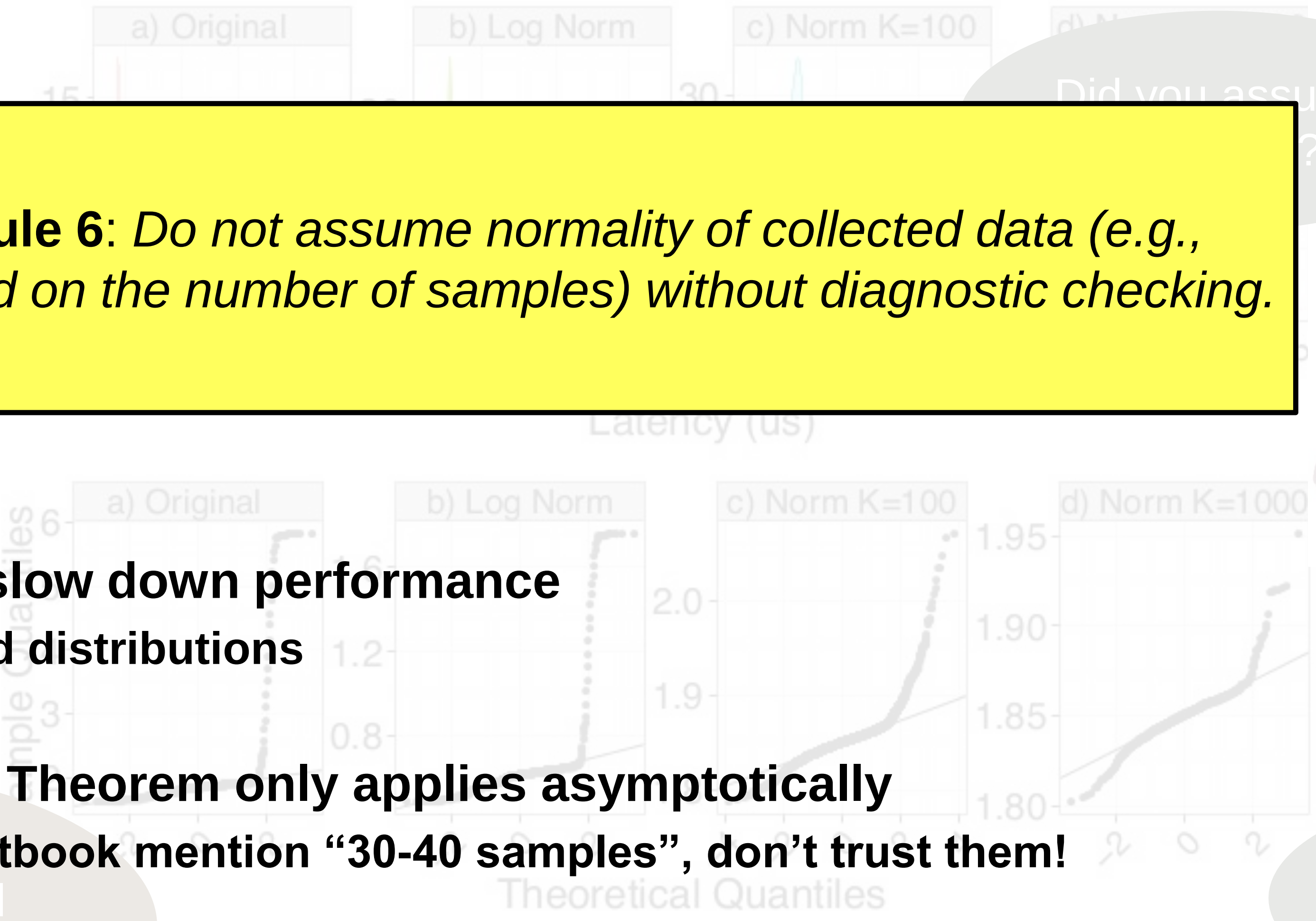
- **Most papers report nondeterministic measurement results**
  - Only 15 mention some measure of variance
  - Only two (!) report confidence intervals
- **CIs allow us to compute the number of required measurements!**
- **Can be very simple, e.g., single sentence in evaluation:**

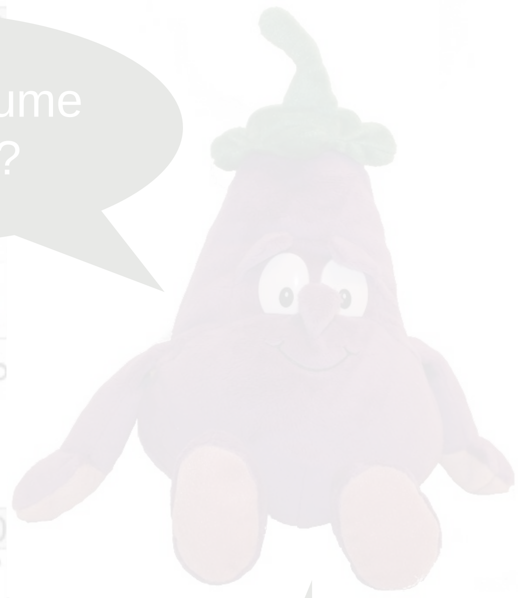
  "We collected measurements until the 99% confidence interval was within 5% of our reported means."

Dealing with variation

The confidence interval is 1.765us to 1.775us

Did you assume ?





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**

normal at all! The real CI is actually 1.6us to 1.9us!

Can we test for normality?

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

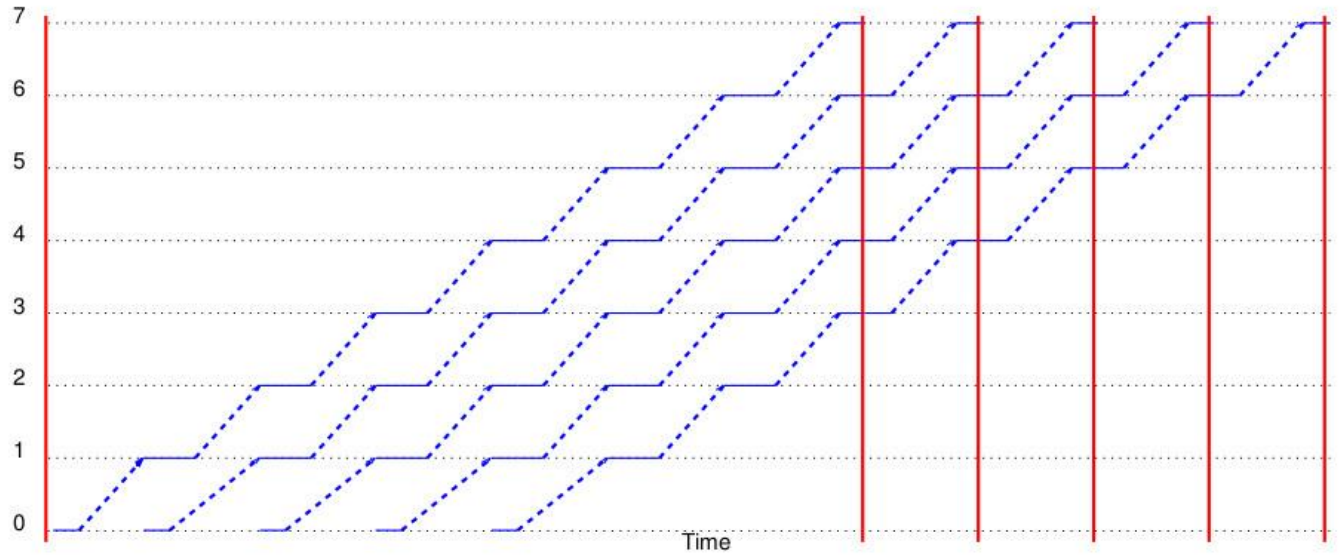
    Even 3-4 measurement can provide very tight CI (assuming normality)

    Can also exploit repetitive nature of many applications

Time in parallel systems

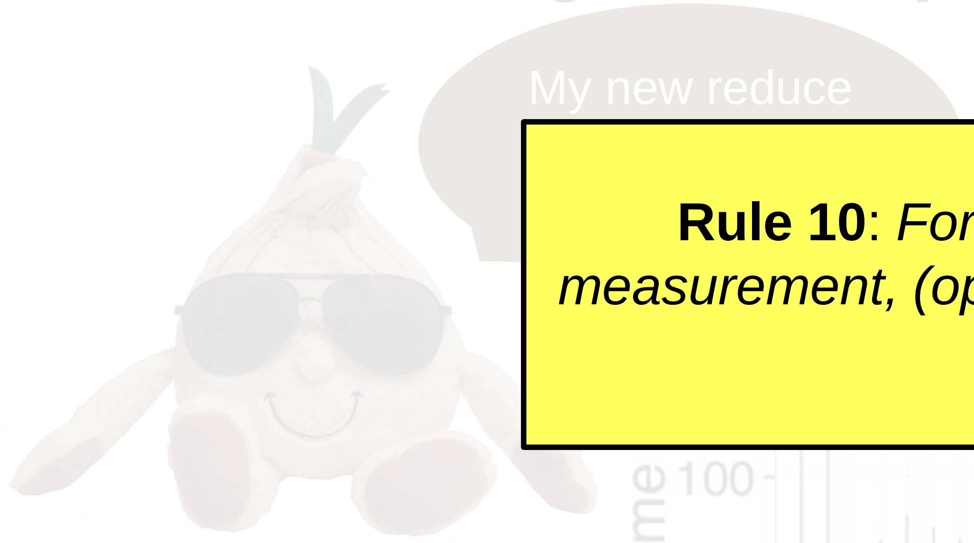
My simple broadcast takes only one latency!
But I measured it so it must be true!

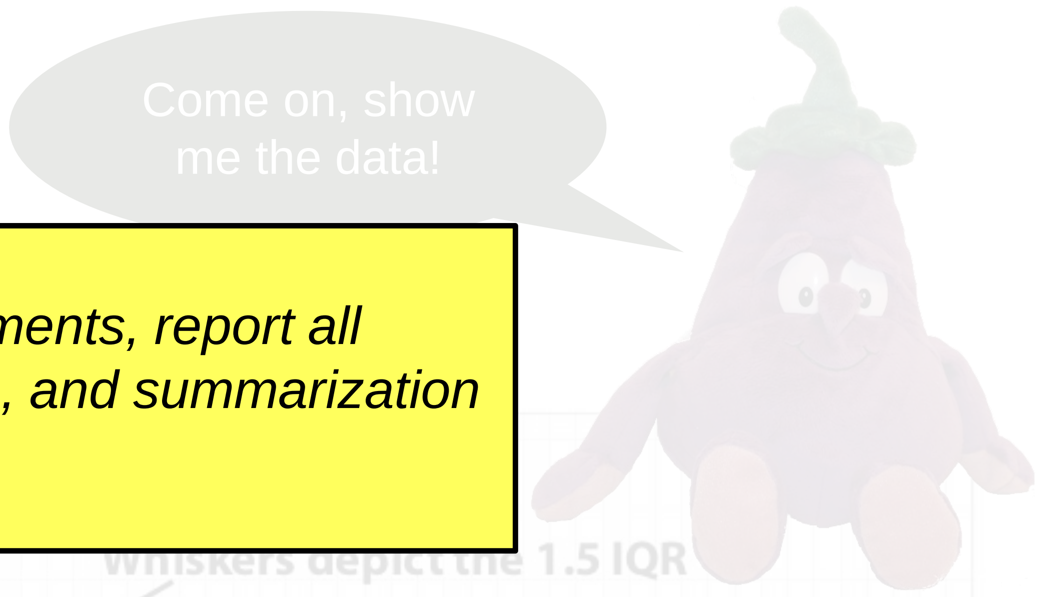
```
t = -MPI_Wtime();
for(i=0; i<1000; i++) {
  MPI_Bcast(…);
}
t += MPI_Wtime();
t /= 1000;
```

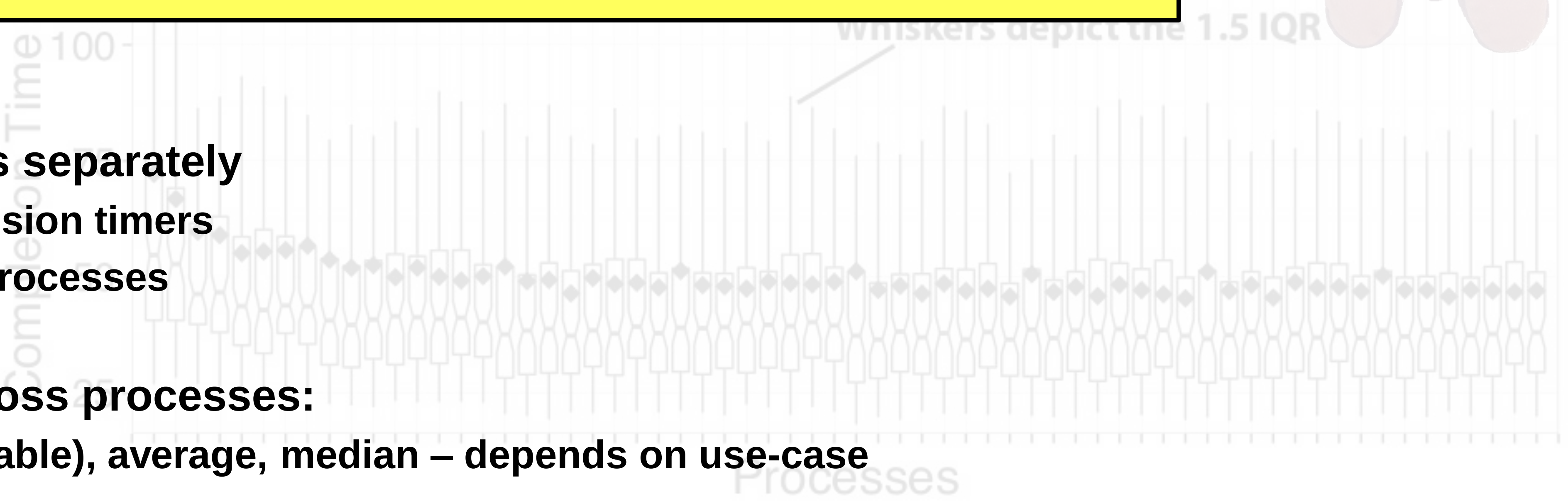


…

Summarizing times in parallel systems!





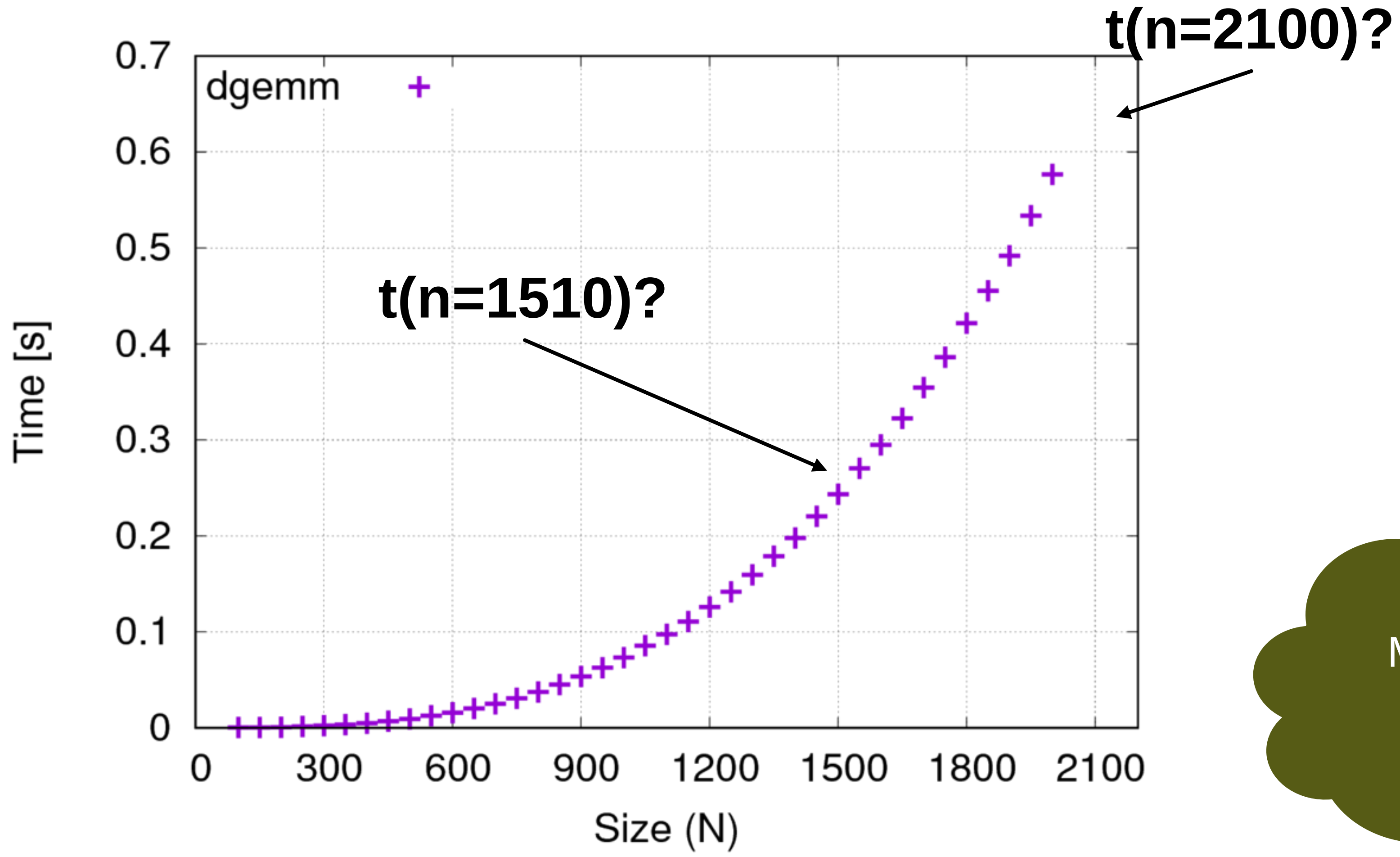


Rule 10: *For parallel time measurements, report all measurement, (optional) synchronization, and summarization techniques.*

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

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

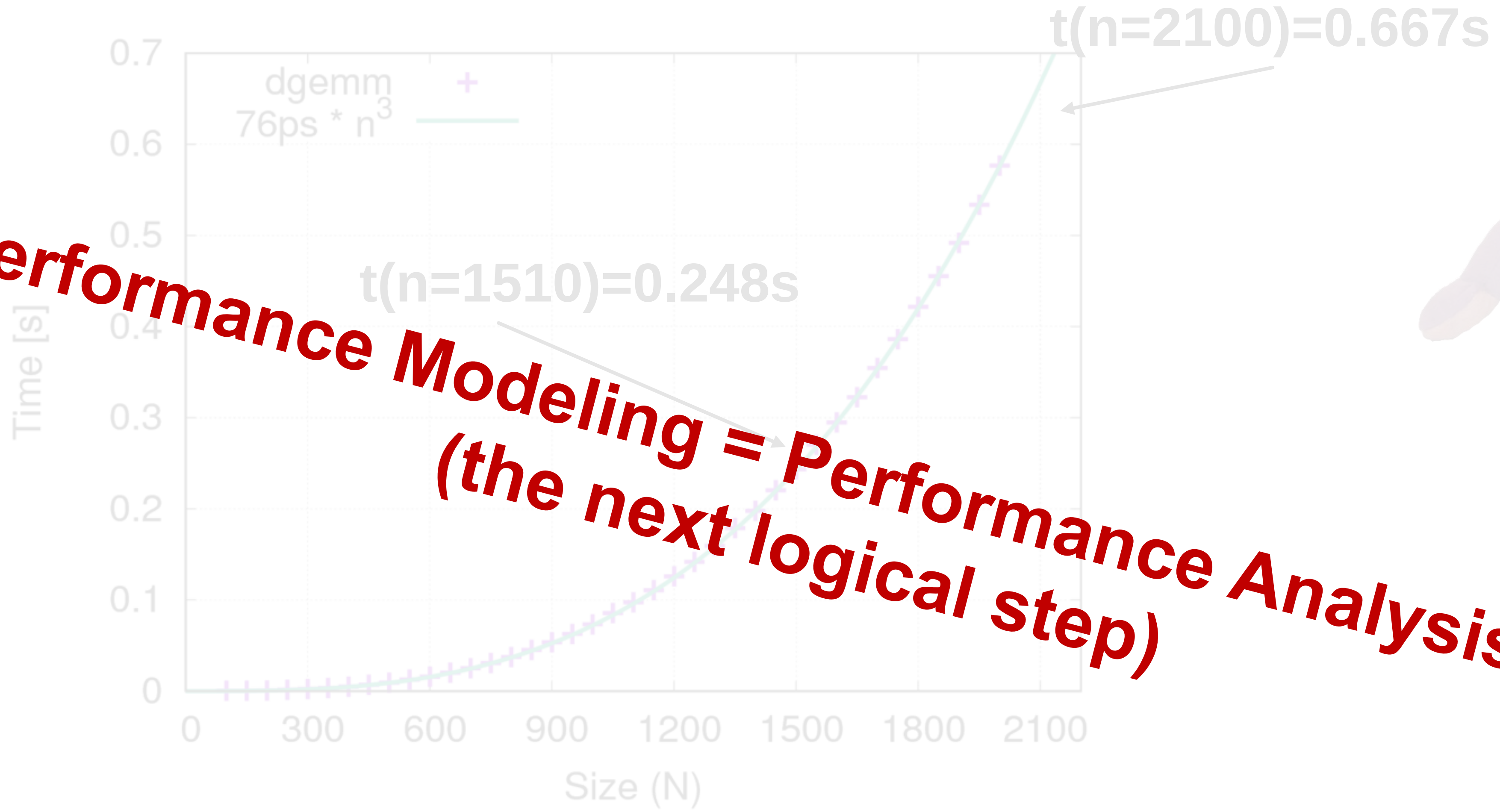
We have the (statistically sound) data, now what?



Matrix Multiply
$t(n) = a*n^3$

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

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

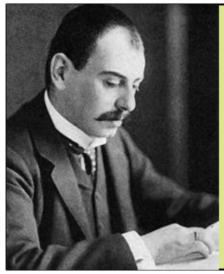


Performance Modeling = Performance Analysis v 2.0
(the next logical step)

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

Conclusions and call for action

- **HPC performance is not reproducible**
- **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]



My inner mathematician to the HPC crowd:

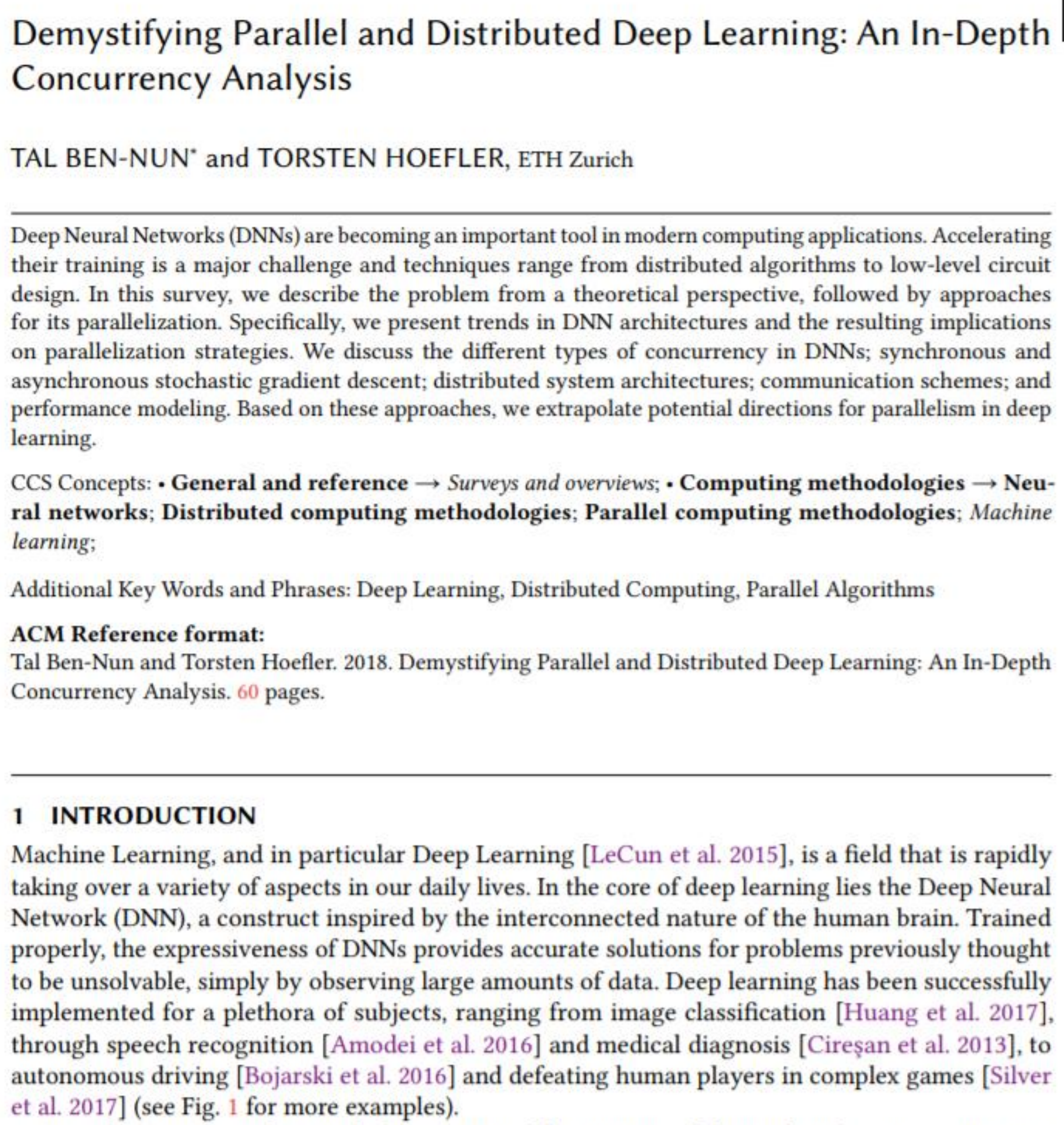
Landau really thought about this hard ☺

O(100) = O(10) = O(10^5) = O(0.5) = O(1)

Mahalo Keli'i!

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

and a shameless plug!



1

Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis

TAL BEN-NUN* and TORSTEN HOEFLER, ETH Zurich

Deep Neural Networks (DNNs) are becoming an important tool in modern computing applications. Accelerating their training is a major challenge and techniques range from distributed algorithms to low-level circuit design. In this survey, we describe the problem from a theoretical perspective, followed by approaches for its parallelization. Specifically, we present trends in DNN architectures and the resulting implications on parallelization strategies. We discuss the different types of concurrency in DNNs; synchronous and asynchronous stochastic gradient descent; distributed system architectures; communication schemes; and performance modeling. Based on these approaches, we extrapolate potential directions for parallelism in deep learning.

CCS Concepts: • **General and reference** → *Surveys and overviews*; • **Computing methodologies** → **Neural networks**; **Distributed computing methodologies**; **Parallel computing methodologies**; *Machine learning*;

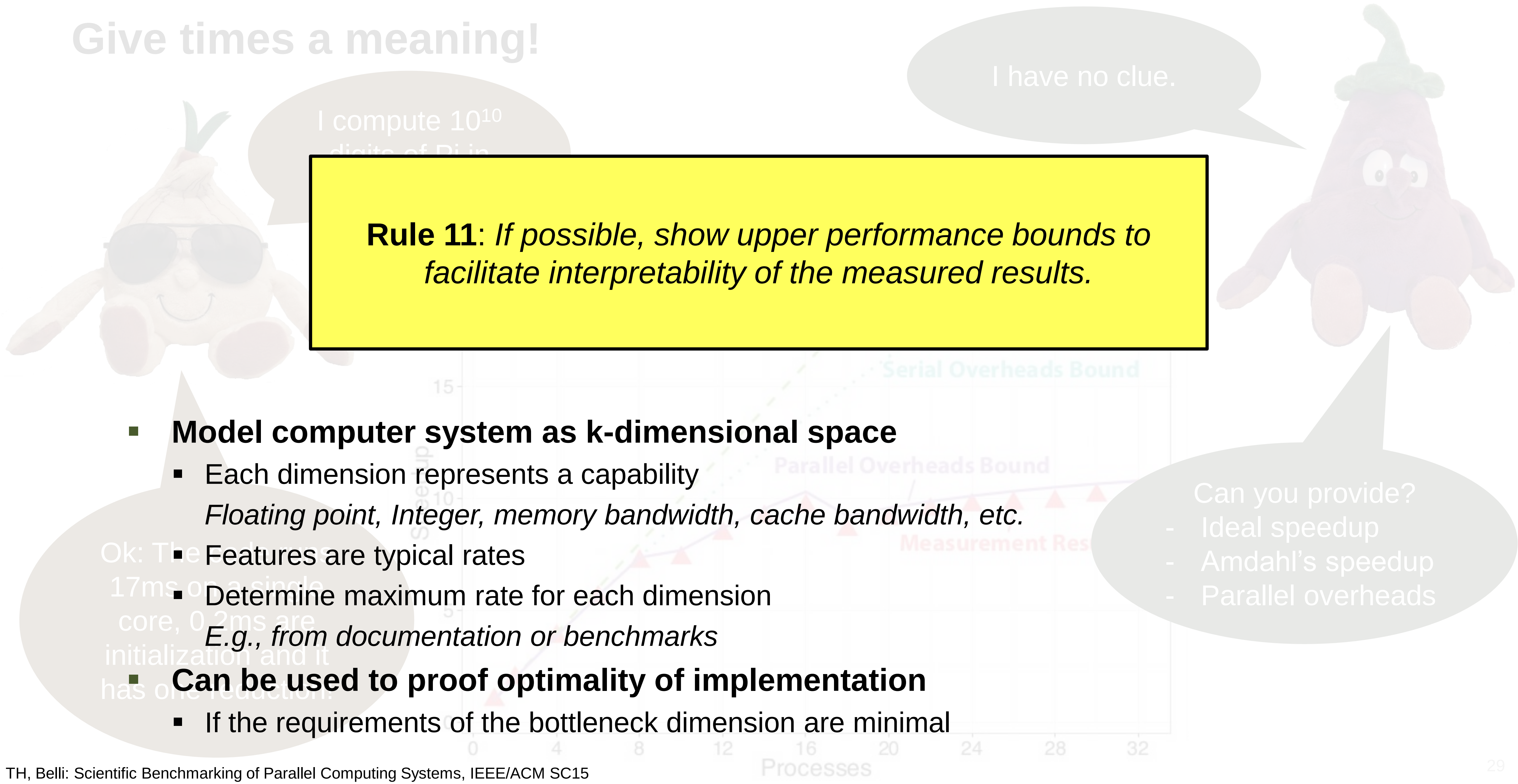
Additional Key Words and Phrases: Deep Learning, Distributed Computing, Parallel Algorithms

ACM Reference format:
Tal Ben-Nun and Torsten Hoefler. 2018. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. 60 pages.

1 INTRODUCTION

Machine Learning, and in particular Deep Learning [LeCun et al. 2015], is a field that is rapidly taking over a variety of aspects in our daily lives. In the core of deep learning lies the Deep Neural Network (DNN), a construct inspired by the interconnected nature of the human brain. Trained properly, the expressiveness of DNNs provides accurate solutions for problems previously thought to be unsolvable, simply by observing large amounts of data. Deep learning has been successfully implemented for a plethora of subjects, ranging from image classification [Huang et al. 2017], through speech recognition [Amodei et al. 2016] and medical diagnosis [Cireşan et al. 2013], to autonomous driving [Bojarski et al. 2016] and defeating human players in complex games [Silver et al. 2017] (see Fig. 1 for more examples).

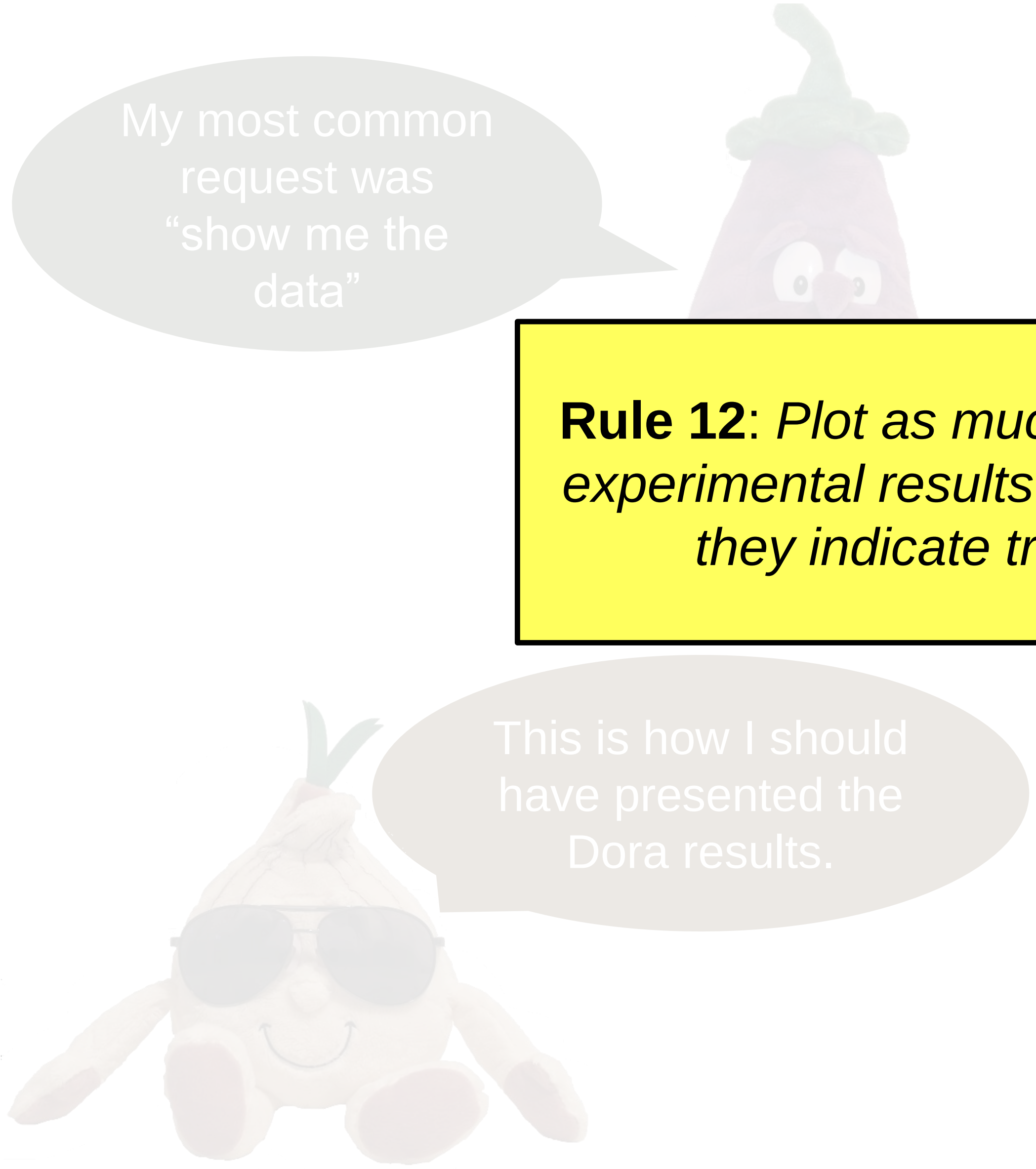
Give times a meaning!

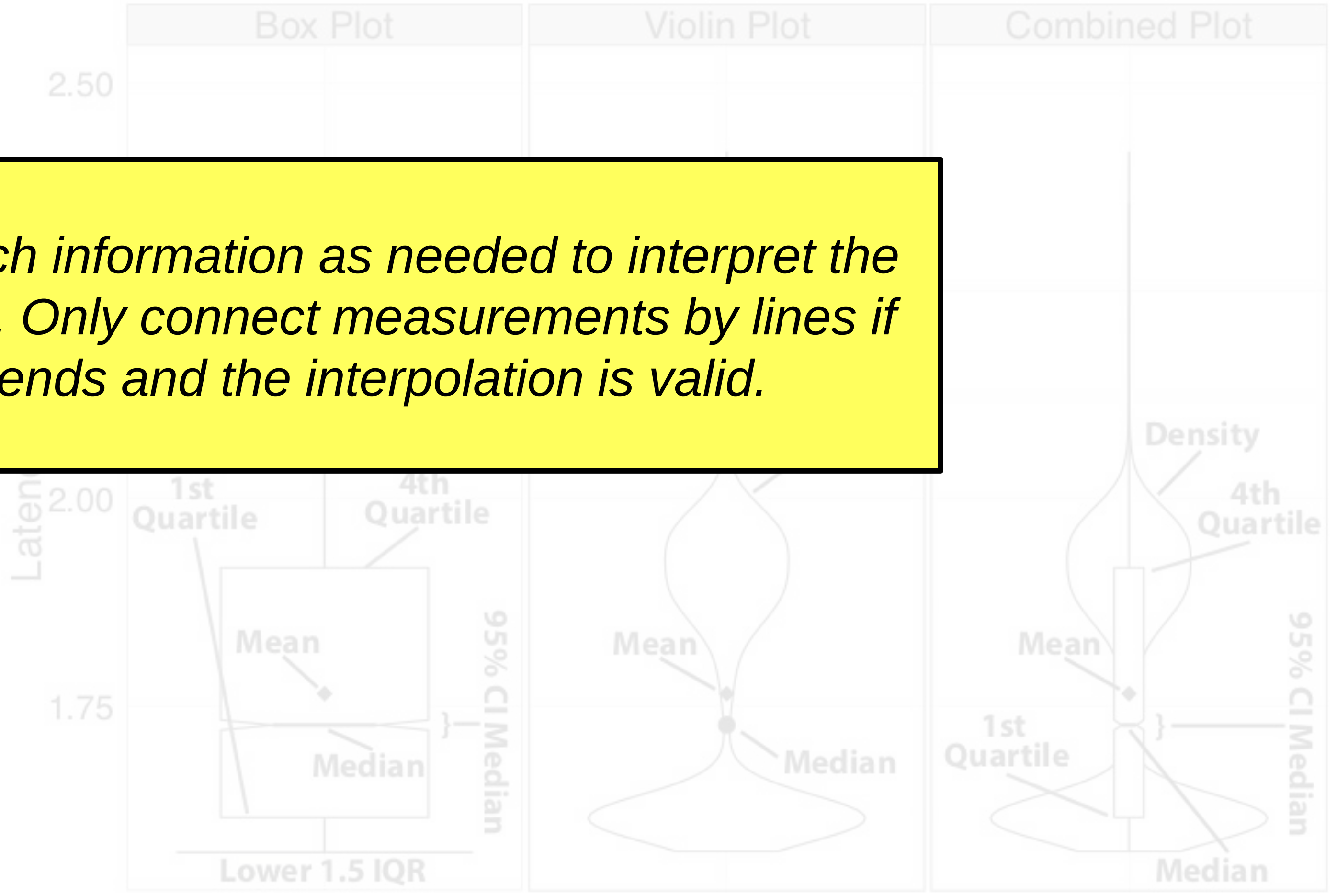


Rule 11: *If possible, show upper performance bounds to facilitate interpretability of the measured results.*

- **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

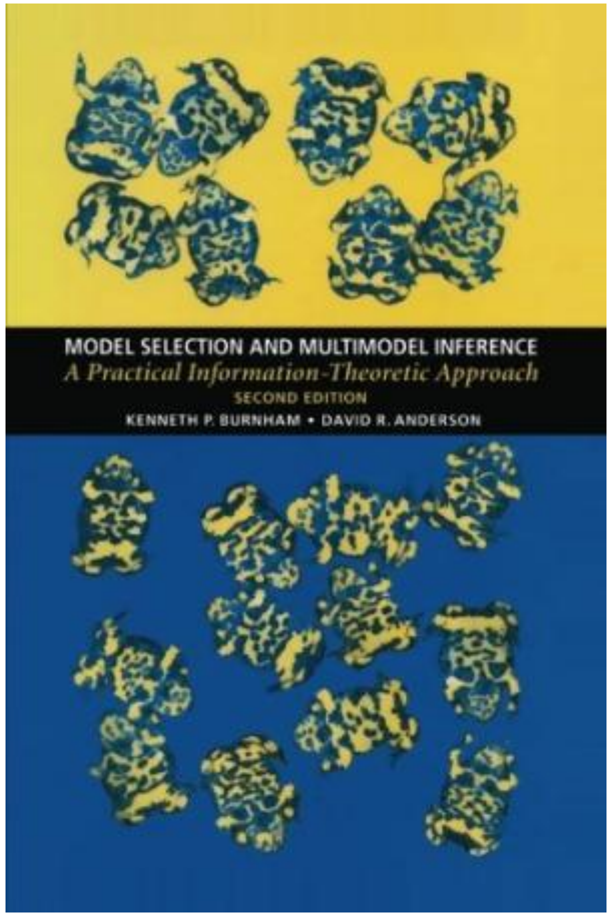
Plot as much information as possible!

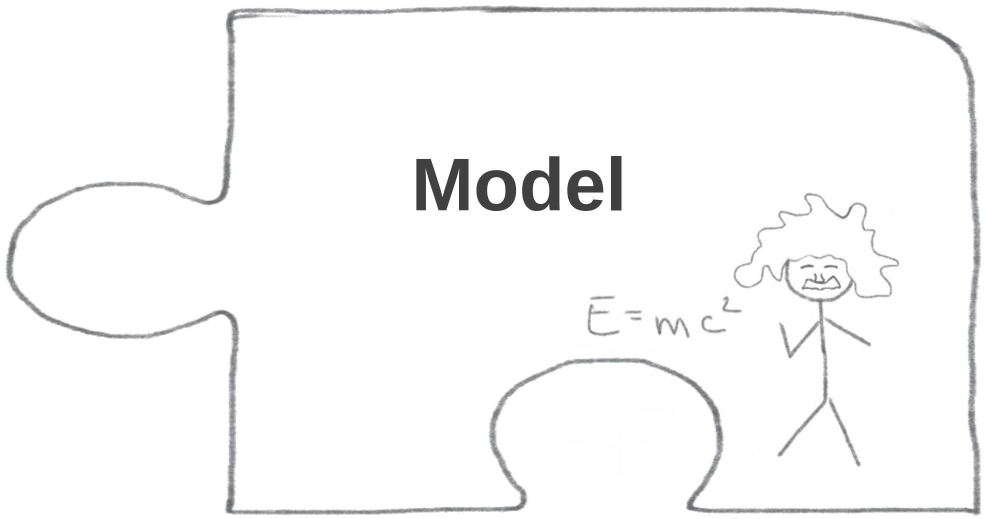

My most common request was "show me the data"
This is how I should have presented the Dora results.



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.*

Part II: Model


MODEL SELECTION AND MULTIMODEL INFERENCE
A Practical Information-Theoretic Approach
SECOND EDITION
KENNETH P. BURNHAM • DAVID R. ANDERSON



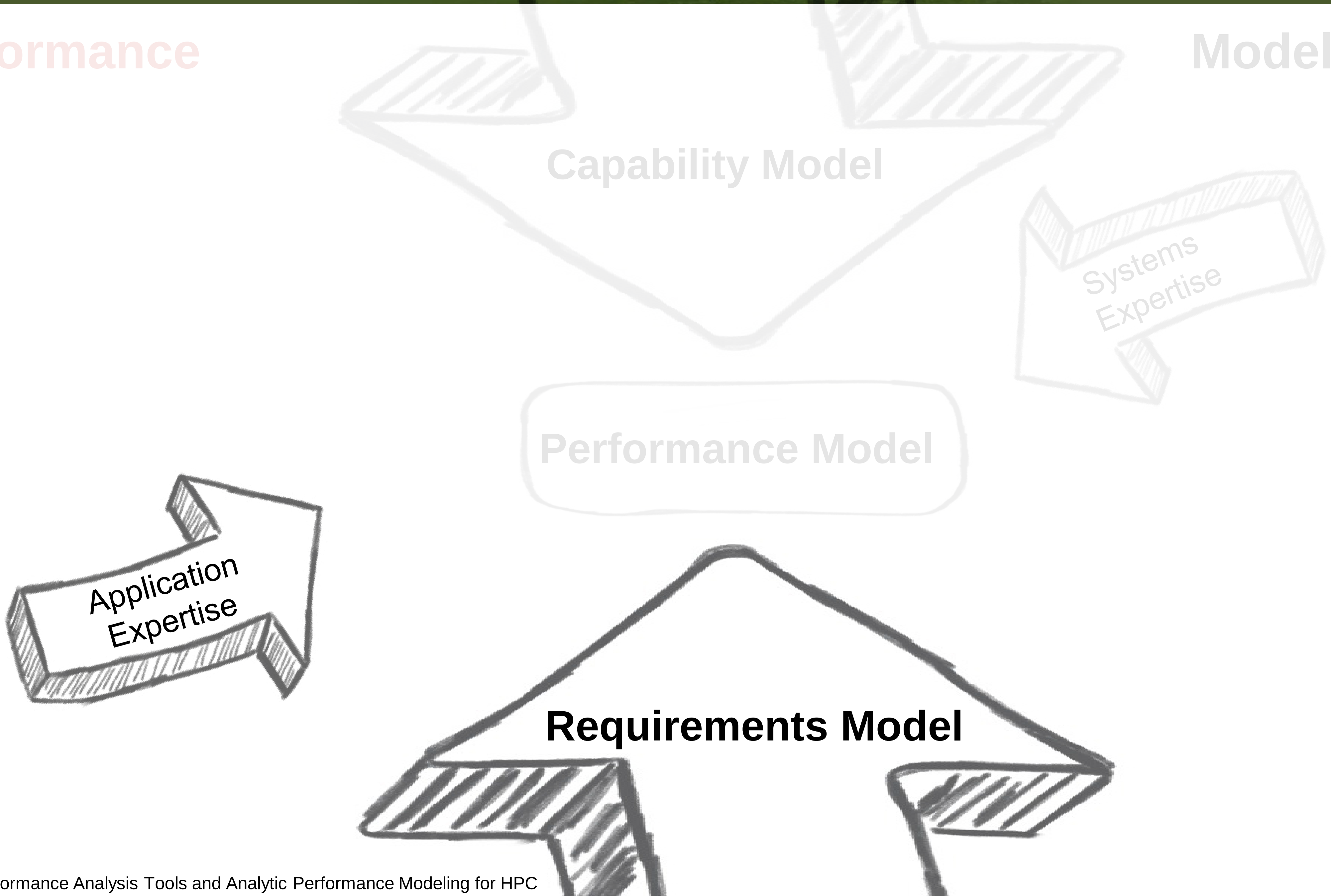
Burnham, Anderson: *"A model is a simplification or approximation of reality and hence will not reflect all of reality. ... Box noted that "all models are wrong, but some are useful." While a model can never be "truth," a model might be ranked from very useful, to useful, to somewhat useful to, finally,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

Cited by 33599

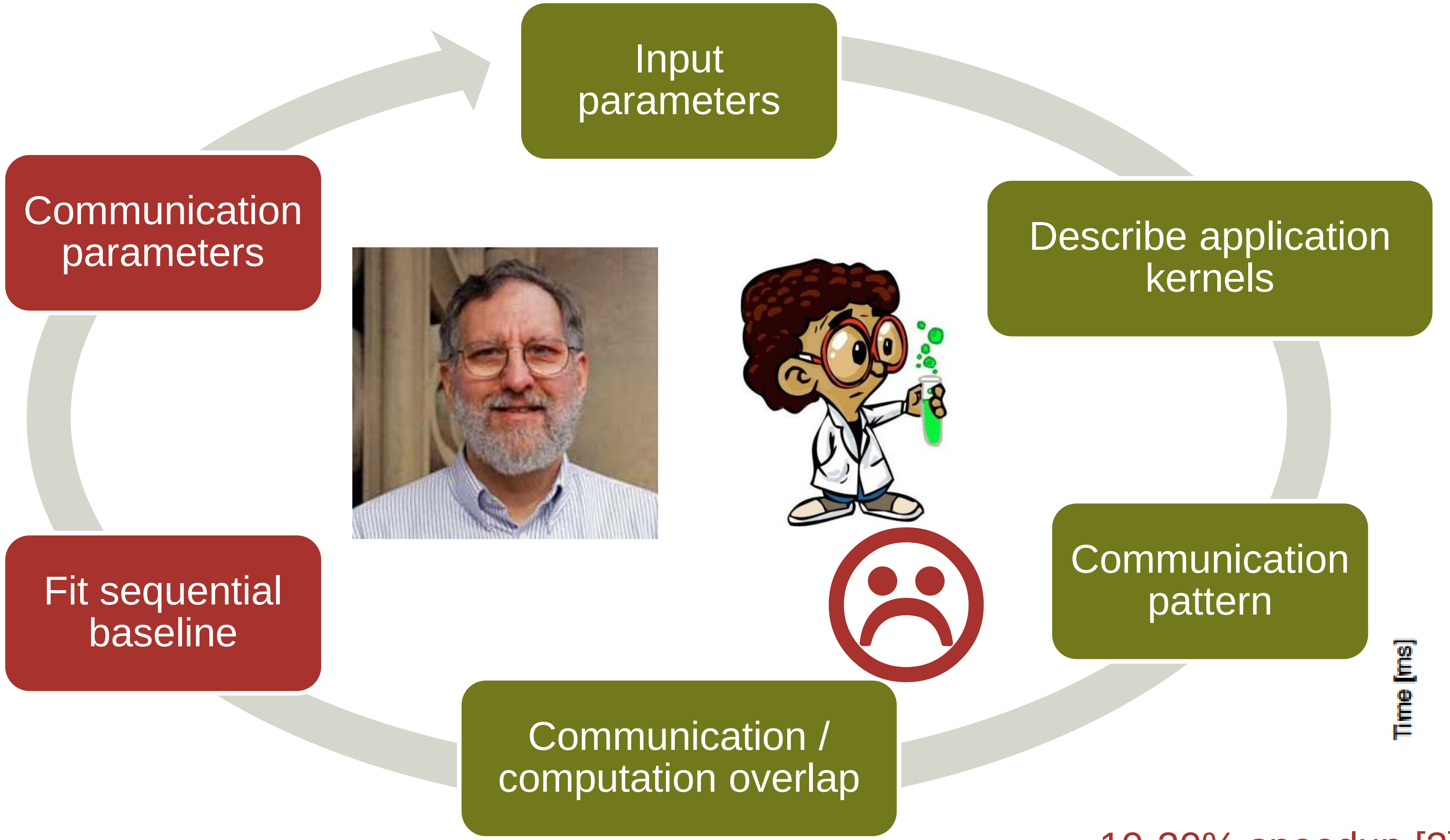
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016

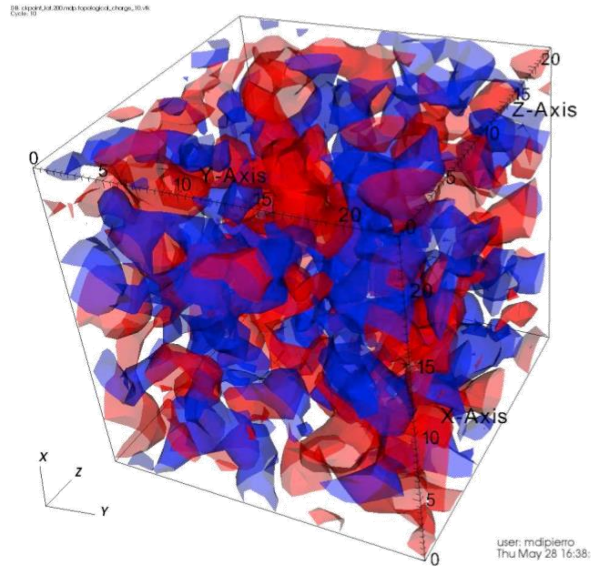
This is generally true for all kinds of modeling.
We focus on **performance modeling** in the following!

Performance Modeling

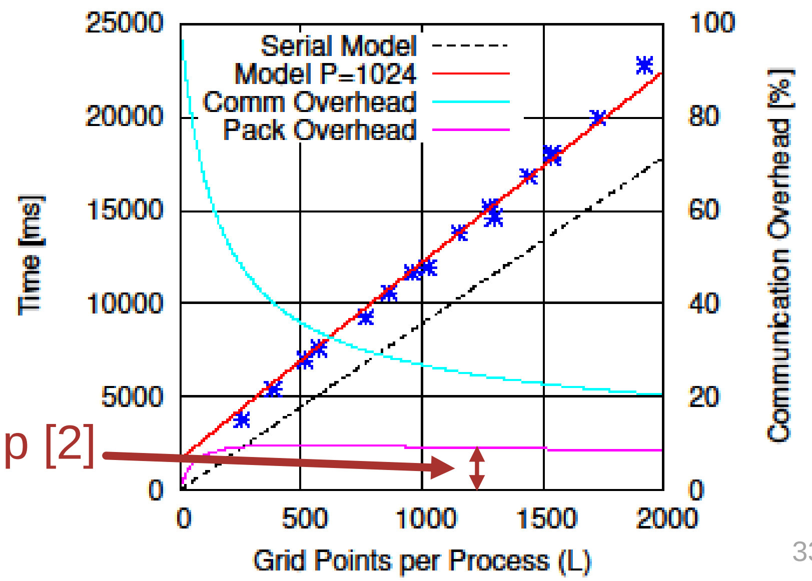


Requirements modeling I: Six-step performance modeling





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

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

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

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

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**

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

$n \in \mathbb{N}$

$i_k \in I$

$j_k \in J$

$I, J \subset \mathbb{Q}$

$n = 2$

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

$J = \{0, 1\}$

$c_1 + c_2 \cdot p$

$c_1 + c_2 \cdot p^2$

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

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

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

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

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

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

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

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

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

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

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

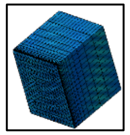
$c_1 \cdot p \cdot \log(p) + c_2 \cdot p^2 \cdot \log(p)$

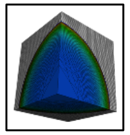
$c_1 \cdot p^2 + c_2 \cdot p^2 \cdot \log(p)$

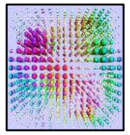
[1]: A. Calotoiu, T. Hoefler, M. Poke, F. Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes, IEEE/ACM SC13

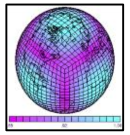
Tool support: Extra-P for automated best-fit modeling [1]

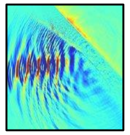
TECHNISCHE UNIVERSITÄT DARMSTADT

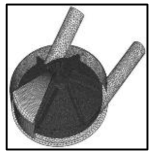

Sweep3d

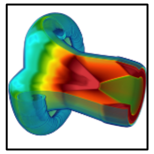

Lulesh


Milc

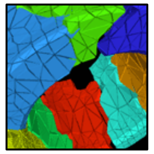

HOMME

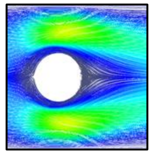

JUSPIC


XNS


BLAST

NEST


UG4


MP2C

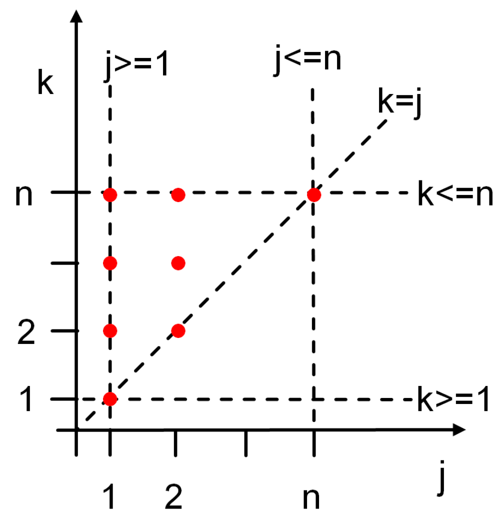
[1] Download Extra-P at: http://www.scalasca.org/software/extra-p/download.html

[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]

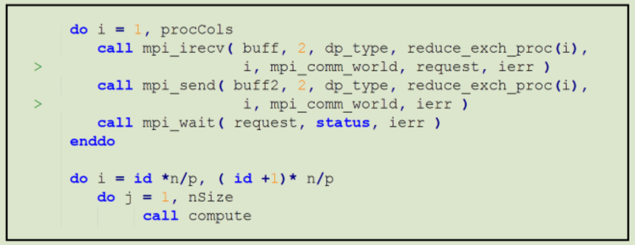
- **Extra-P selects model based on best fit to the data**
  - What if the data is not sufficient or too noisy?
- **Back to first principles**
  - The source code describes all possible executions
  - Describing all possibilities is too expensive, focus on counting loop iterations symbolically

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



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

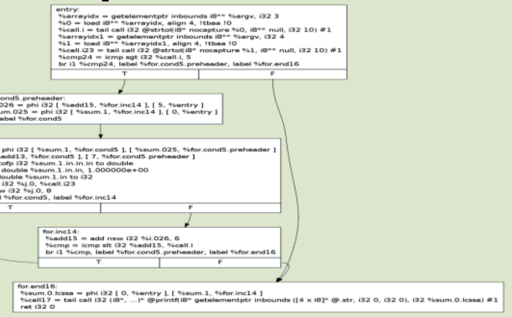
Parallel program



```
    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



```
entry:
 %arrayidx = getelementptr inbounds i8** %argv, i32 3
 %0 = load i8** %arrayidx, align 4, !tbaa !0
 %call.i = tail call i32 @strtol(i8* nocapture %0, i8** null, i32 10) #1
 %arrayidx1 = getelementptr inbounds i8** %argv, i32 4
 %1 = load i8** %arrayidx1, align 4, !tbaa !0
 %call.i23 = tail call i32 @strtol(i8* nocapture %1, i8** null, i32 10) #1
 %cmp24 = icmp sgt i32 %call.i, 5
 br i1 %cmp24, label %for.cond5.preheader, label %for.end16

for.cond5.preheader:
 %i.026 = phi i32 [ %add15, %for.inc14 ], [ 5, %entry ]
 %sum.025 = phi i32 [ %sum.1, %for.inc14 ], [ 0, %entry ]
 br label %for.cond5

for.cond5:
 %sum.1.in.in.in = phi i32 [ %sum.1, %for.cond5 ], [ %sum.025, %for.cond5.preheader ]
 %j.0 = phi i32 [ %add13, %for.cond5 ], [ 7, %for.cond5.preheader ]
 %sum.1.in.in = sitofp i32 %sum.1.in.in.in to double
 %sum.1.in = fadd double %sum.1.in.in, 1.000000e+00
 %sum.1 = fptosi double %sum.1.in to i32
 %cmp6 = icmp slt i32 %j.0, %call.i23
 %add13 = add nsw i32 %j.0, 8
 br i1 %cmp6, label %for.cond5, label %for.inc14

for.inc14:
 %add15 = add nsw i32 %i.026, 6
 %cmp = icmp slt i32 %add15, %call.i
 br i1 %cmp, label %for.cond5.preheader, label %for.end16

for.end16:
 %sum.0.lcssa = phi i32 [ 0, %entry ], [ %sum.1, %for.inc14 ]
 %call17 = tail call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @.str, i32 0, i32 0), i32 %sum.0.lcssa) #1
 ret i32 0
```

Number of iterations

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

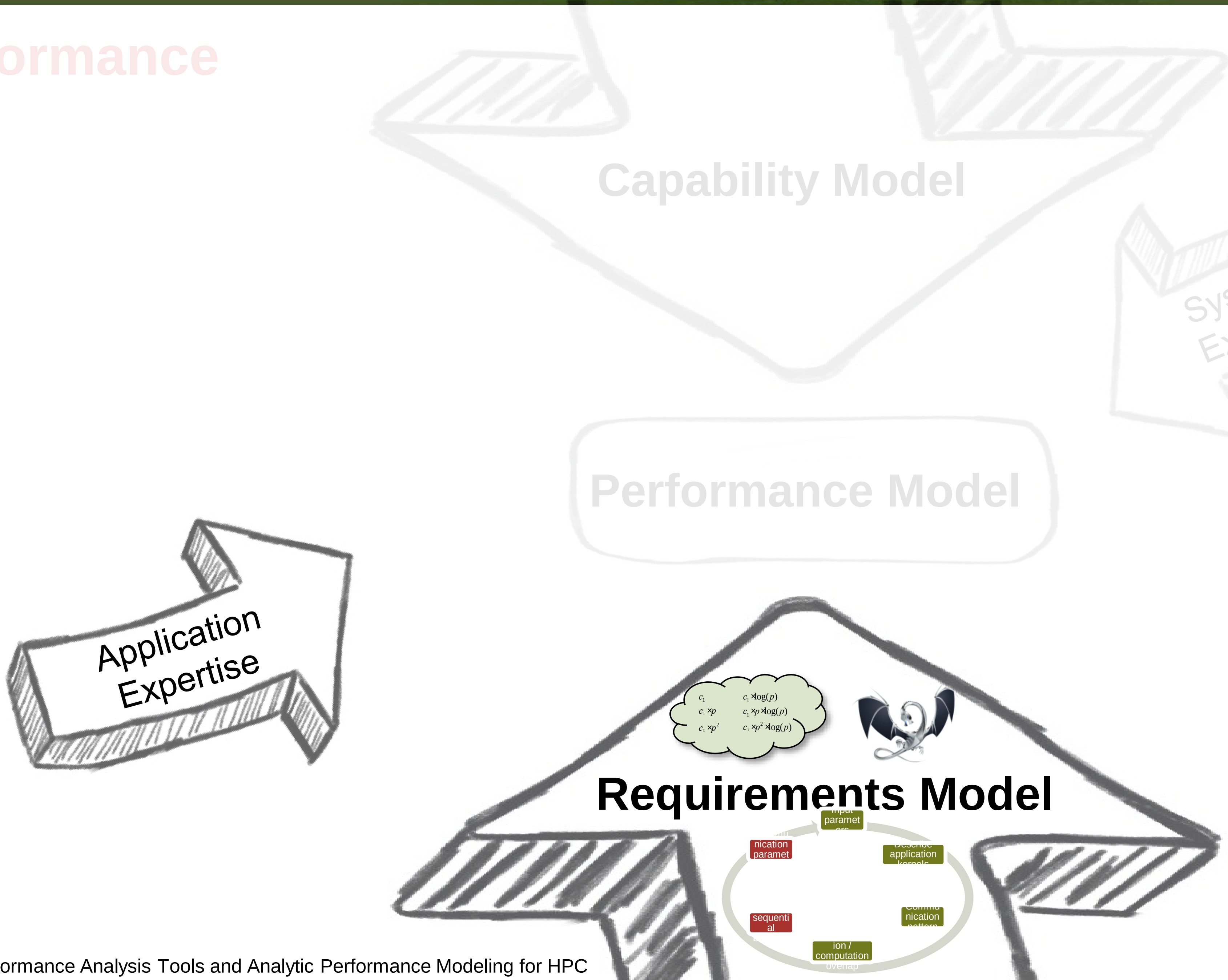
Requirements Models

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

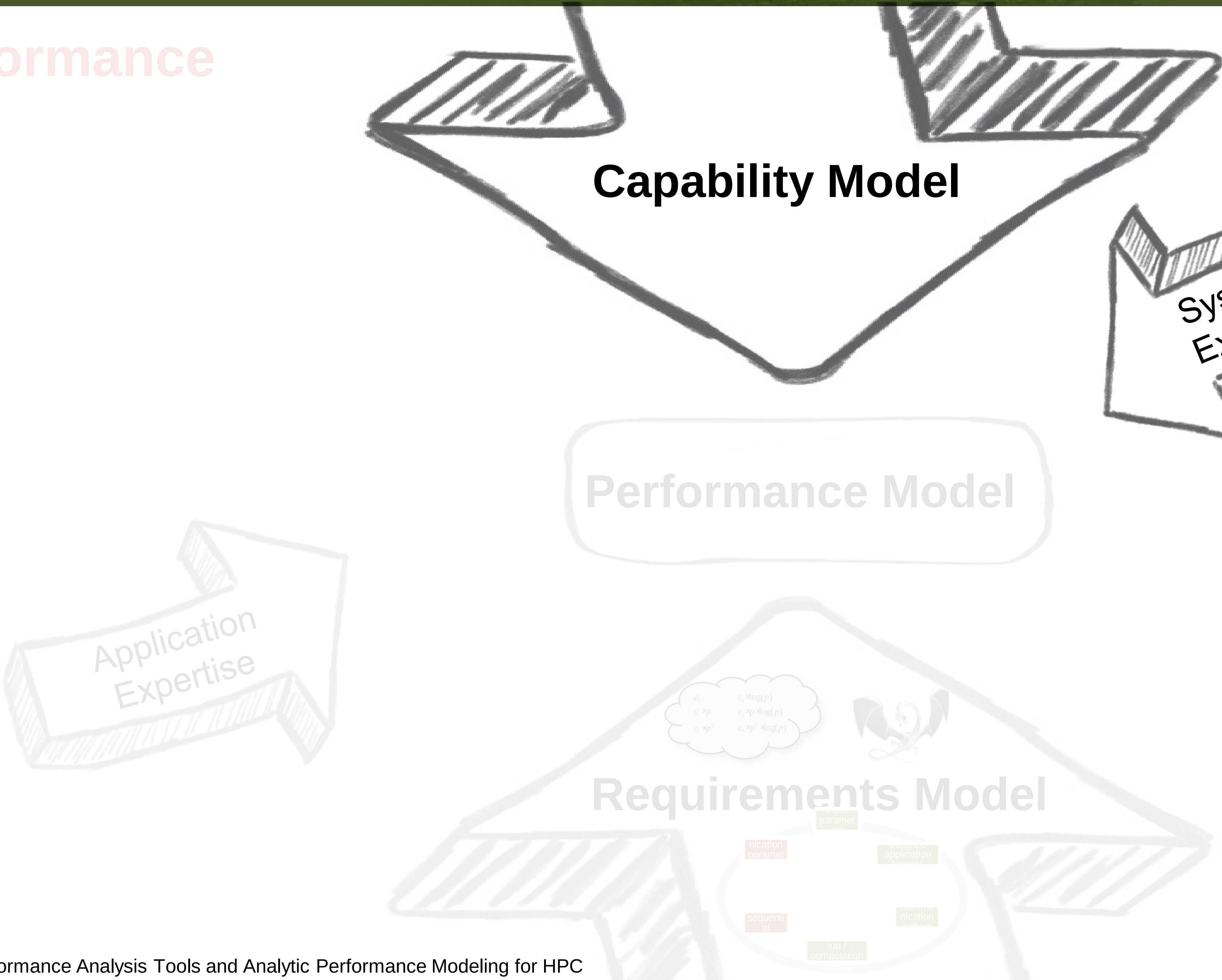
$$D = N\big|_{p\to\infty}$$

[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

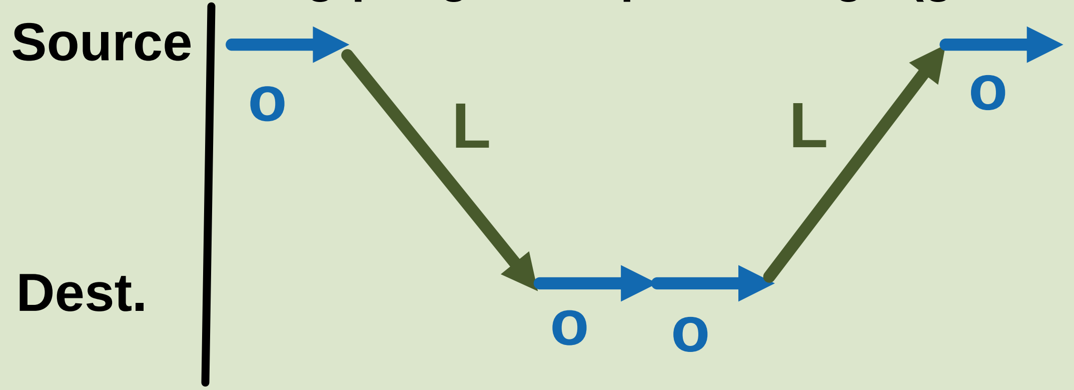
LogP

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

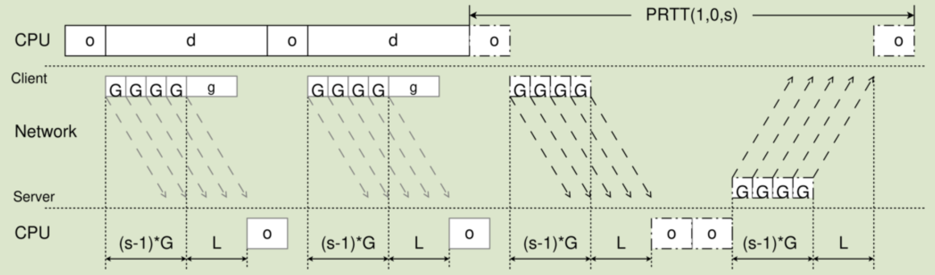
David E. Culler, Richard M. Karp, David Patterson, Abhijit Sahay, Eunice E. Santos, Klaus Erik Schauser, Ramesh Subramonian, and Thorsten von Eicken

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



Finding LogGOPS parameters

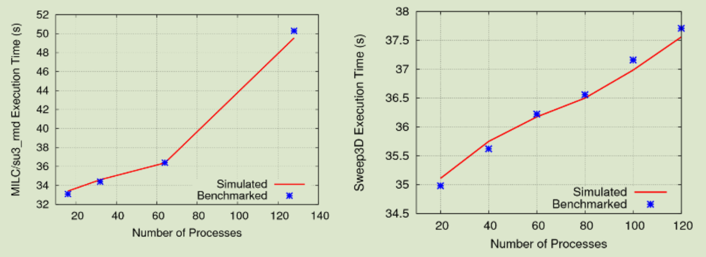
Netgauge [2], model from first principles, fit to data using special kernels



Large scale LogGOPS Simulation

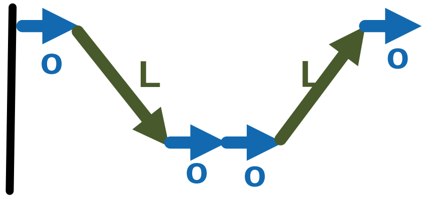
LogGOPSim [1], simulates LogGOPS with 10 million MPI ranks

<5% error

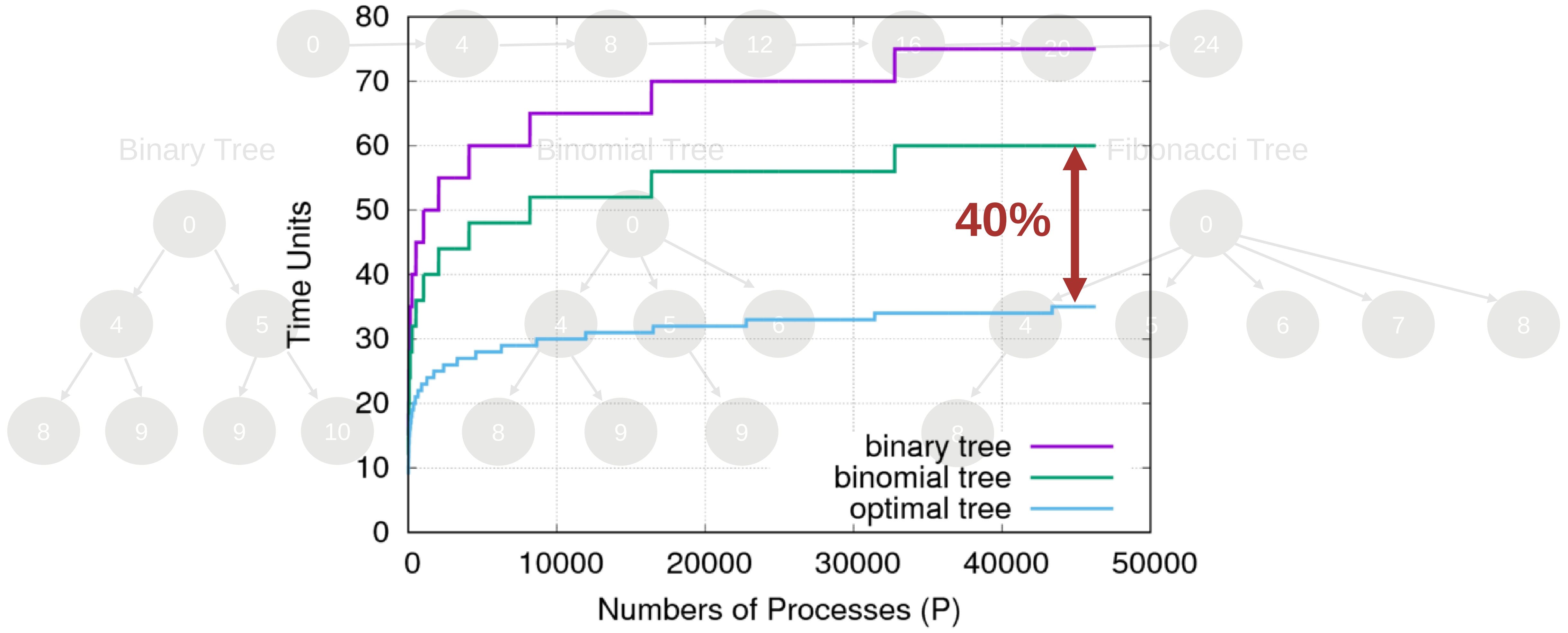


[1]: TH, T. Schneider and A. Lumsdaine: LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model, LSAP 2010, https://spcl.inf.ethz.ch/Research/Performance/LogGOPSim/
[2]: TH, T. Mehlan, A. Lumsdaine and W. Rehm: Netgauge: A Network Performance Measurement Framework, HPCC 2007, https://spcl.inf.ethz.ch/Research/Performance/Netgauge/

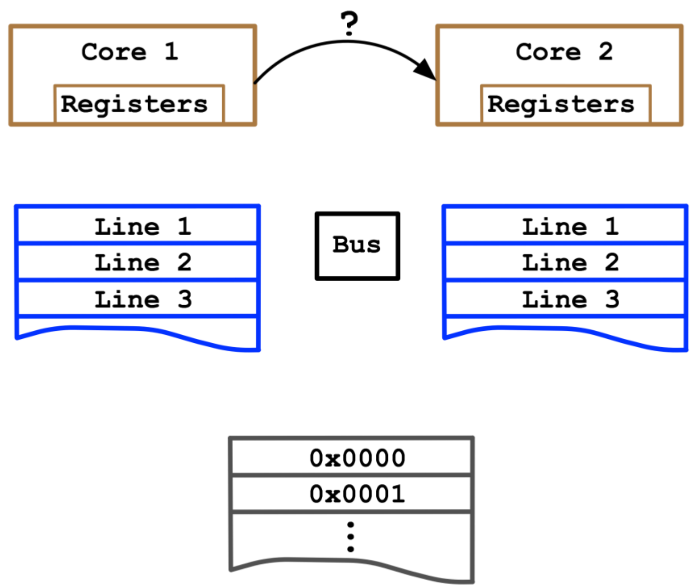
2) Design optimal algorithms – small broadcast in LogP



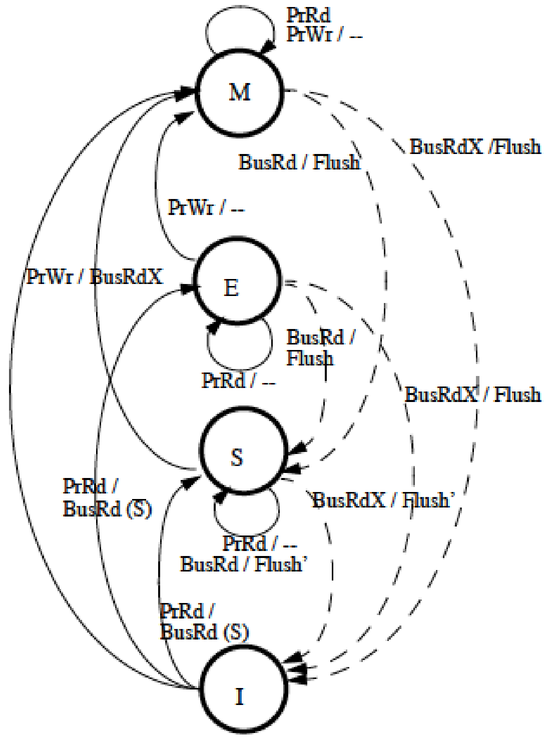
L=2, o=1, P=7



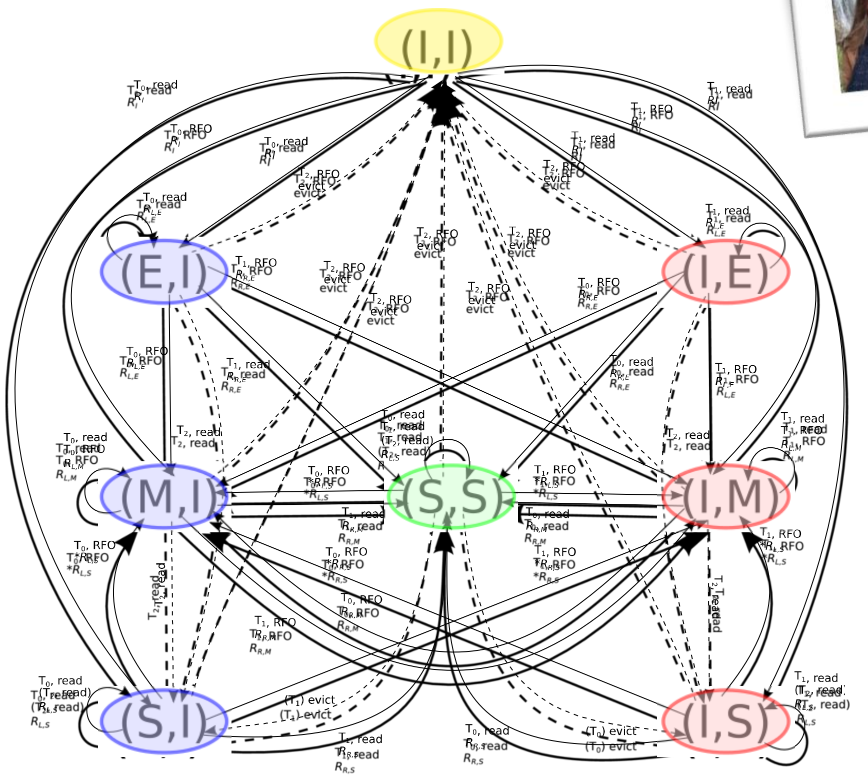
Capability models for cache-to-cache communication

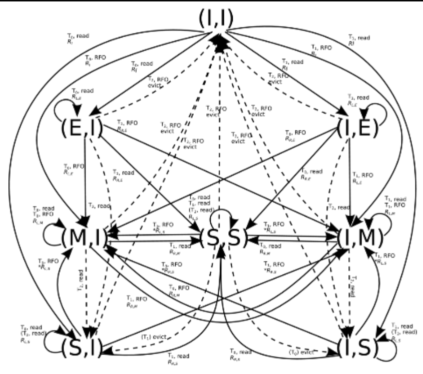


X



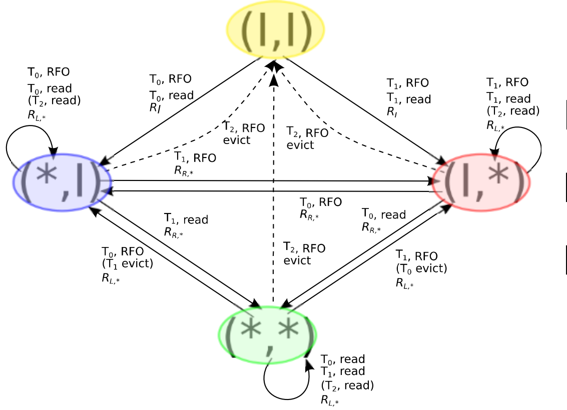
=





|

=



Invalid read $R_I \approx 135$ ns

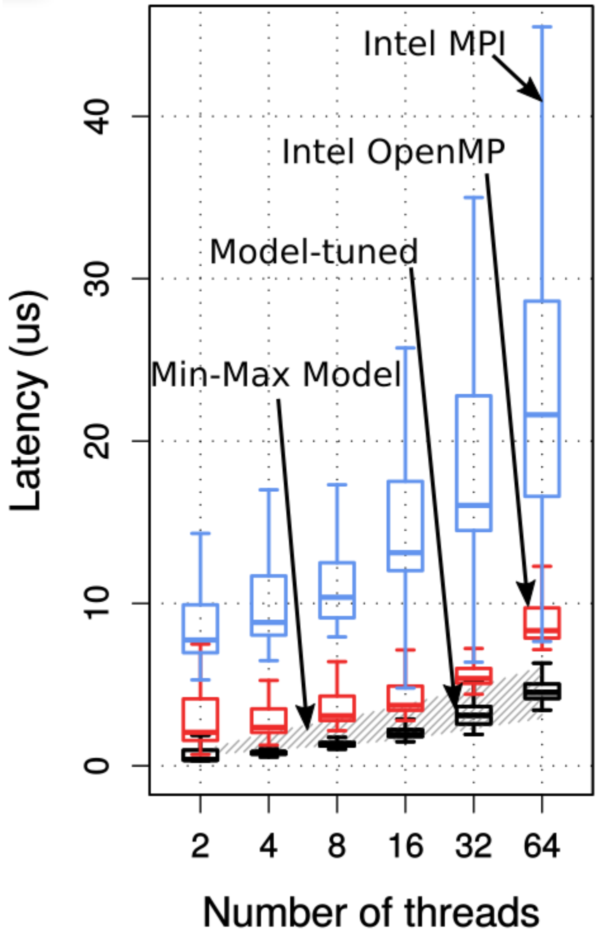
Local read: $R_L = 3.8$ ns

Remote read $R_R \approx 115$ ns

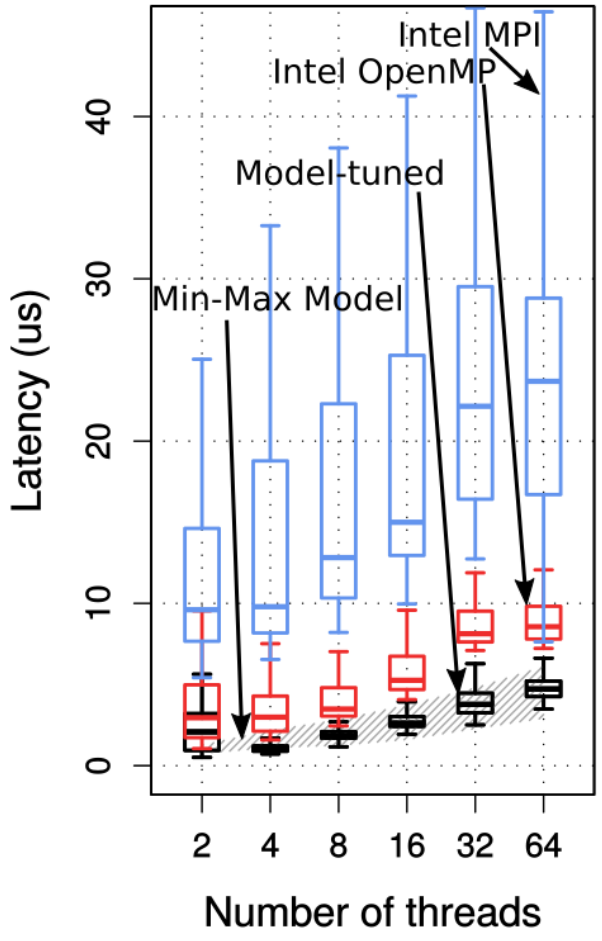
S. Ramos, TH: *"Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL", IEEE IPDPS'17*

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

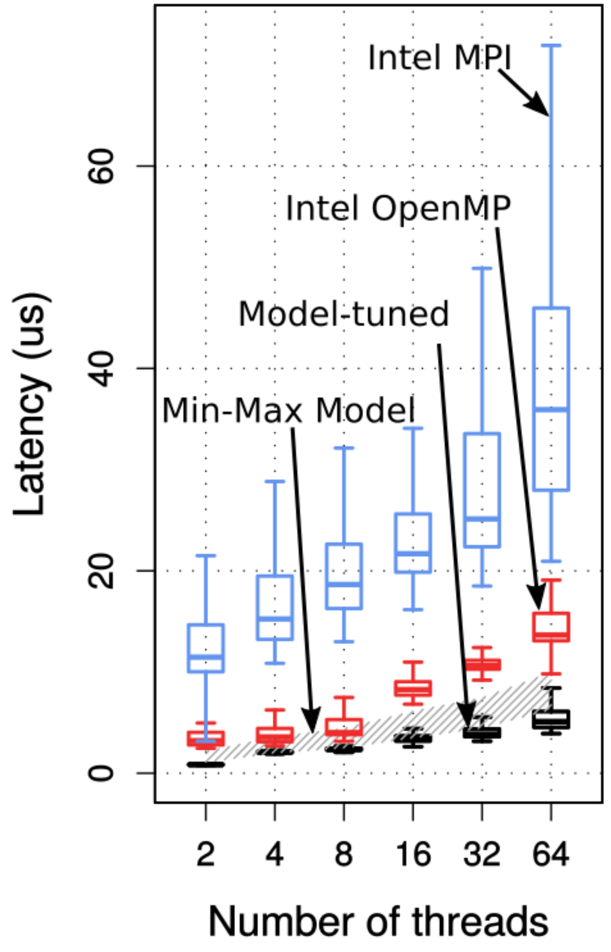


(a) Filling Tiles.

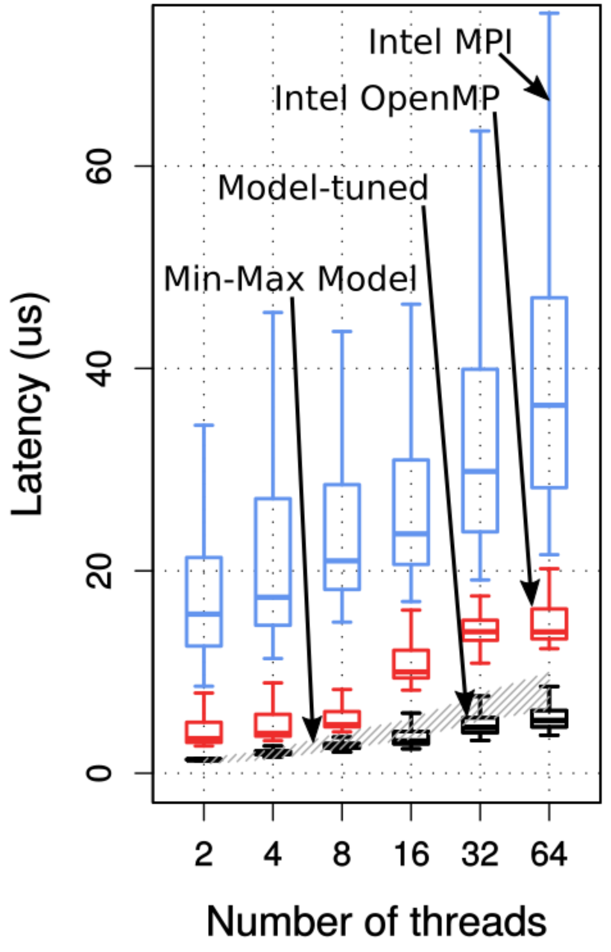


(b) Scatter.

Barrier (7x faster than OpenMP)



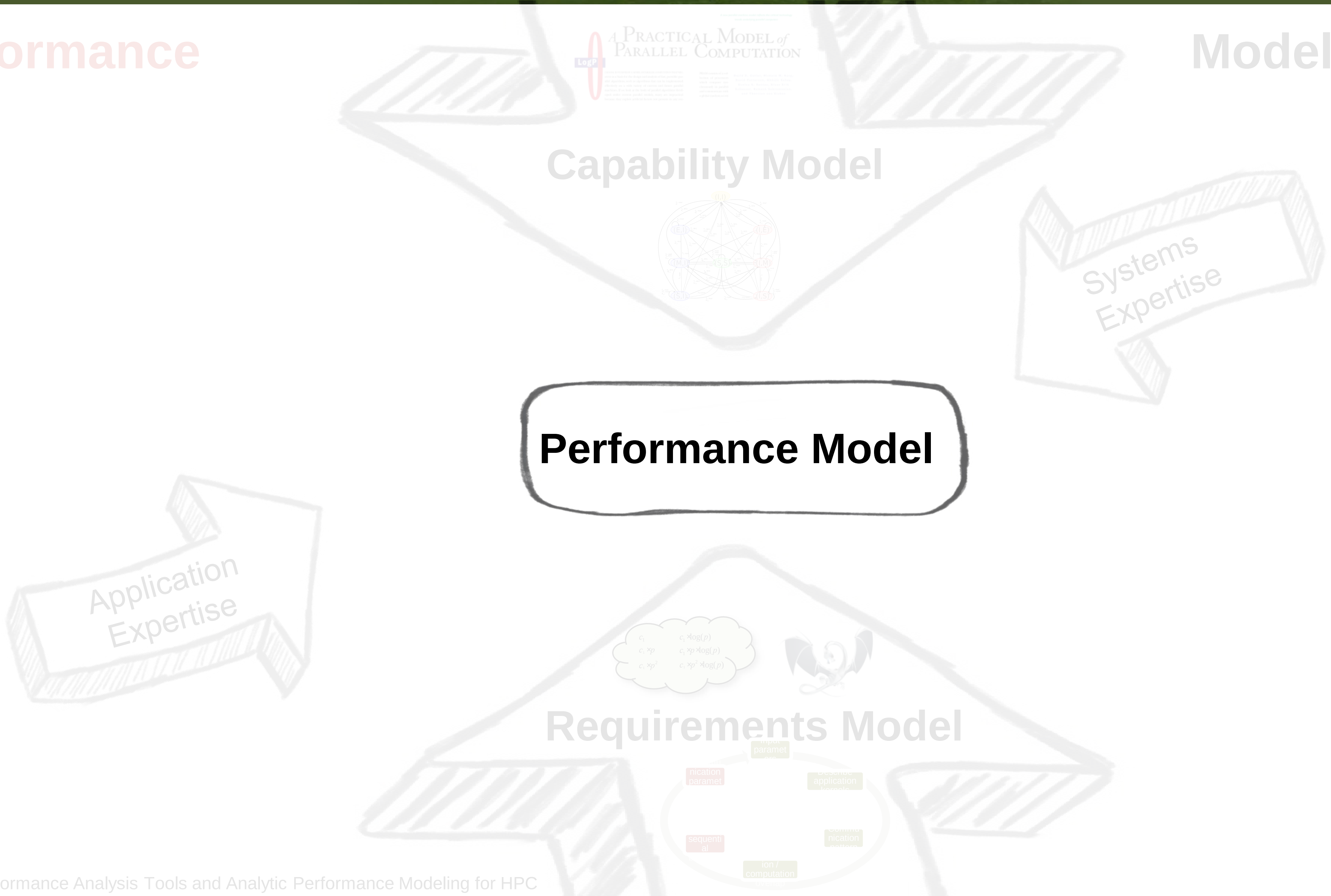
(a) Filling Tiles.



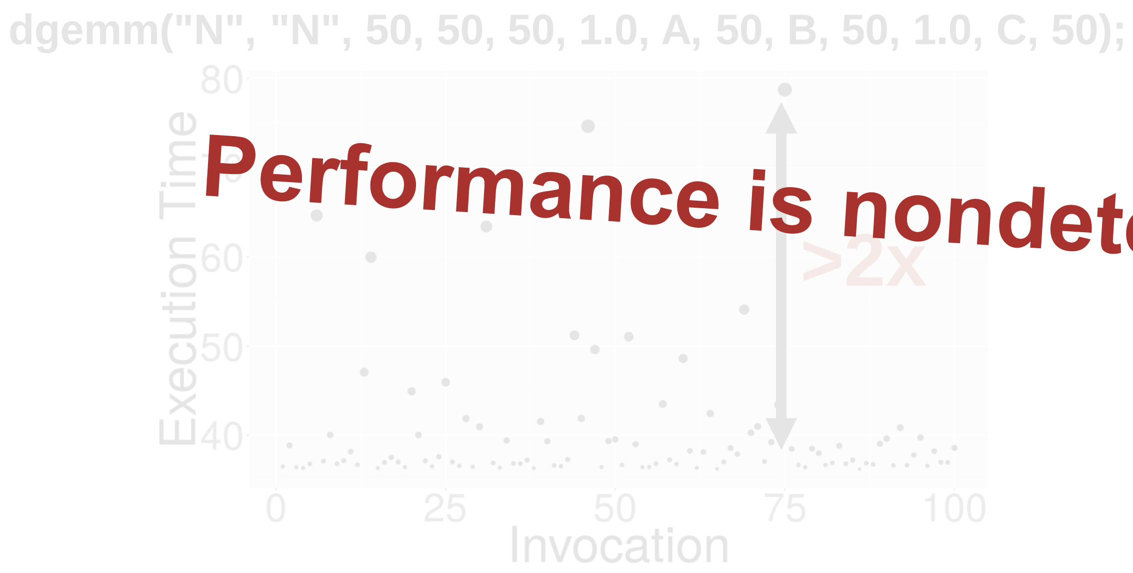
(b) Scatter.

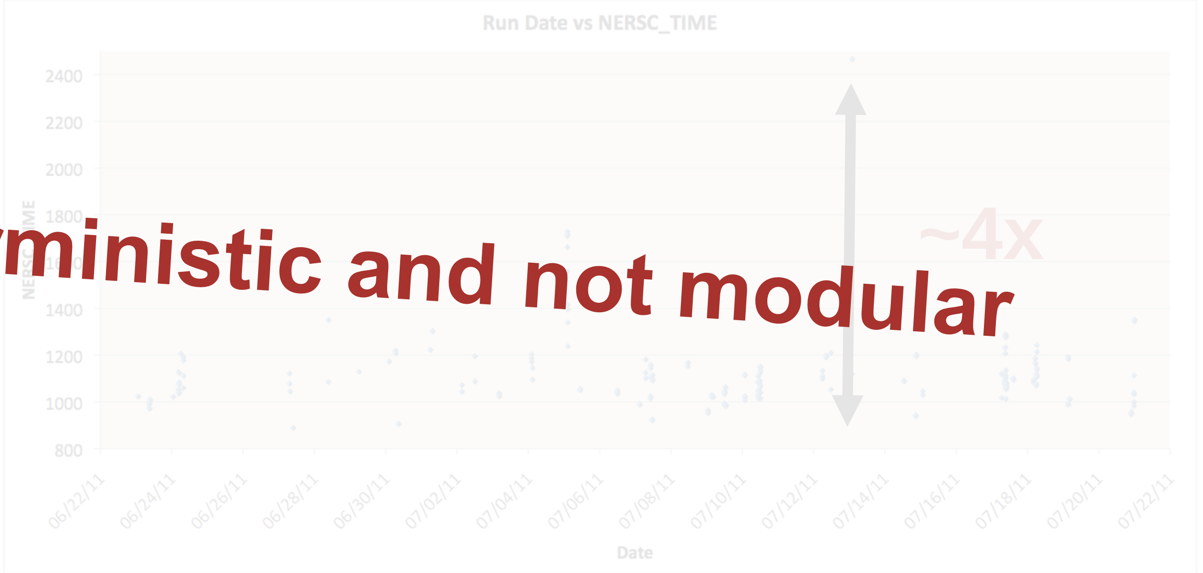
Reduce (5x faster then OpenMP)

Performance Modeling

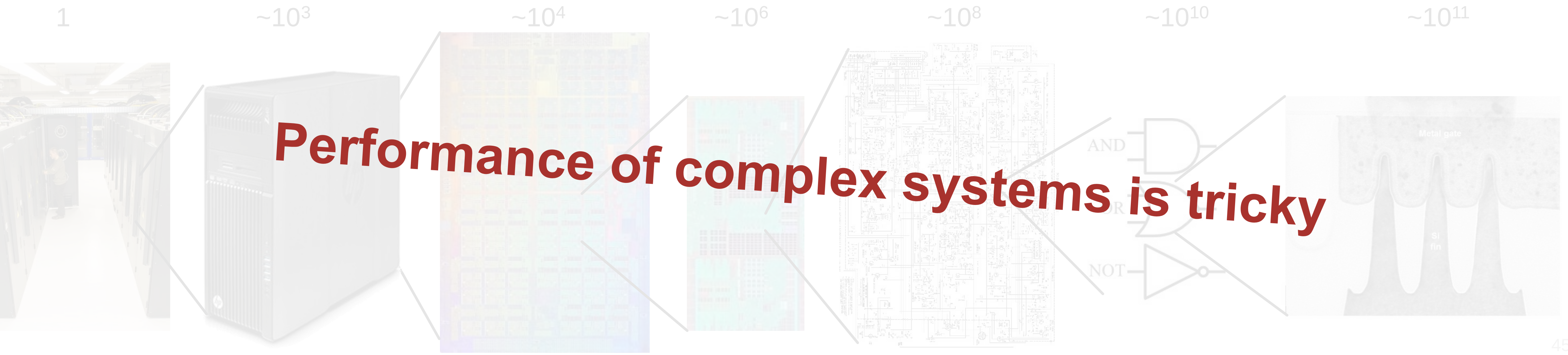


High Performance Computing



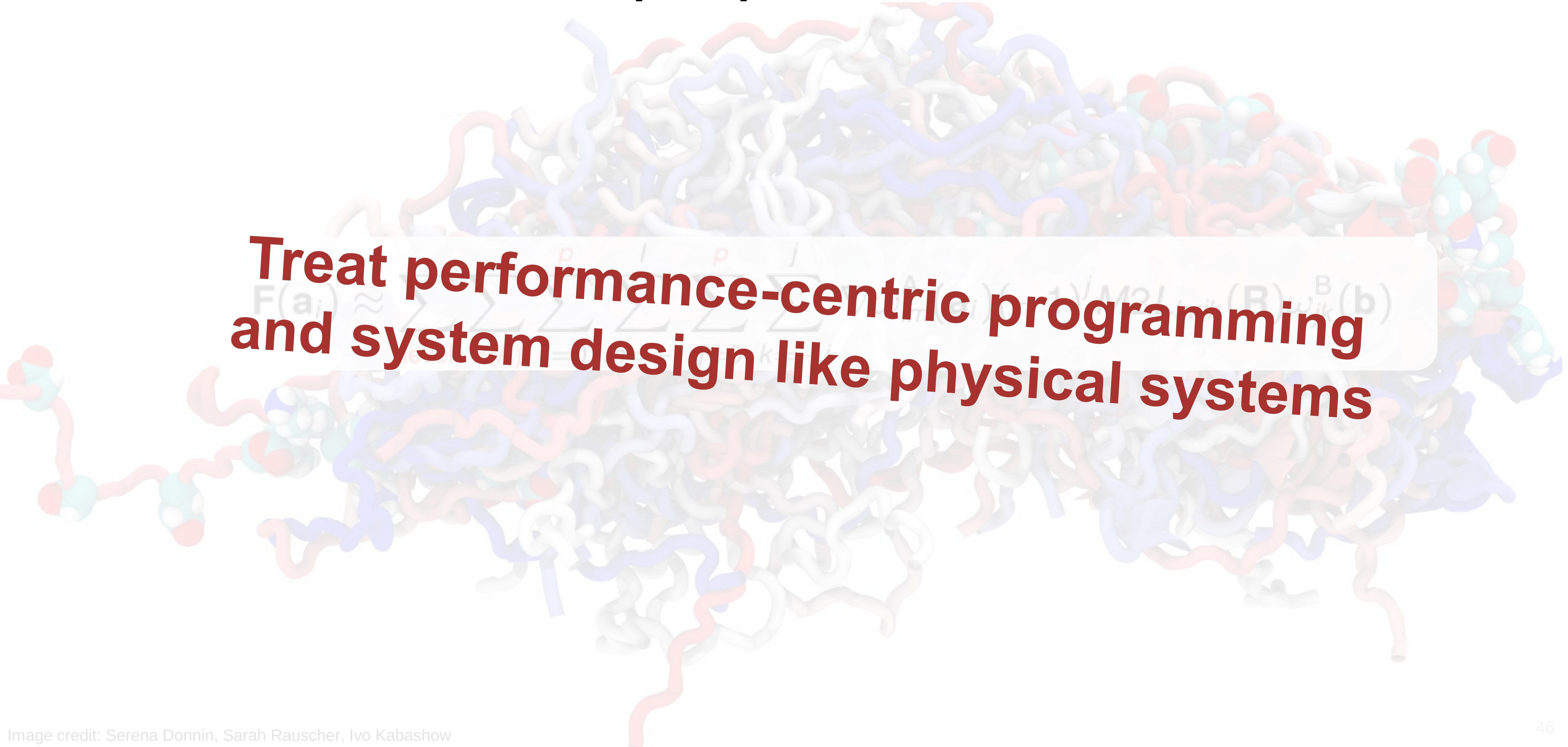


Performance is nondeterministic and not modular



Performance of complex systems is tricky

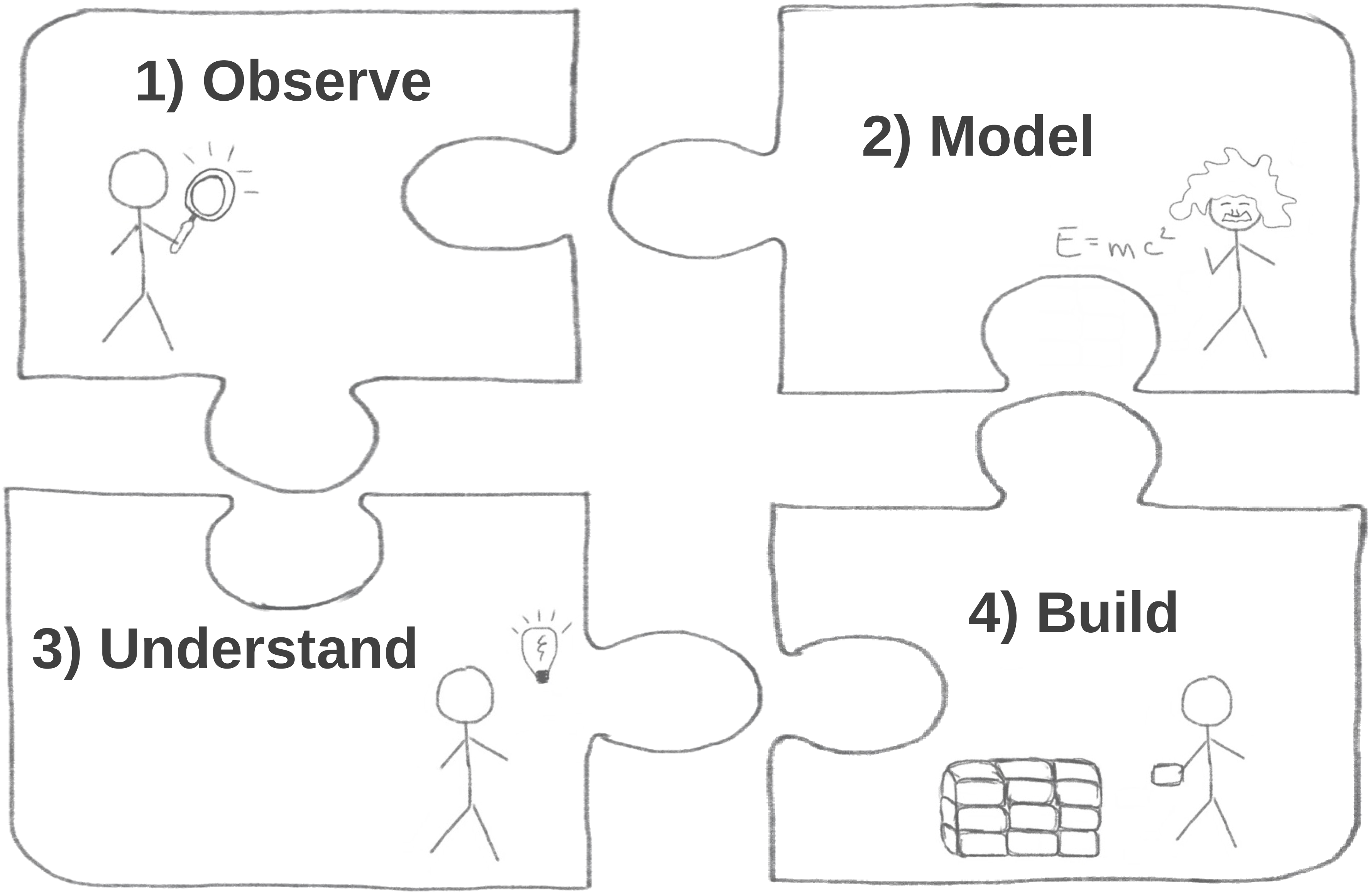
HPC is used to solve complex problems!



Treat performance-centric programming and system design like physical systems

Image credit: Serena Donnin, Sarah Rauscher, Ivo Kabashow

Scientific Performance Engineering



Part I: Observe

Measure systems

Experimental design

Collect data

Examine documentation

Gather statistics

Document process

Factorial design

Disclaimer(s)

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

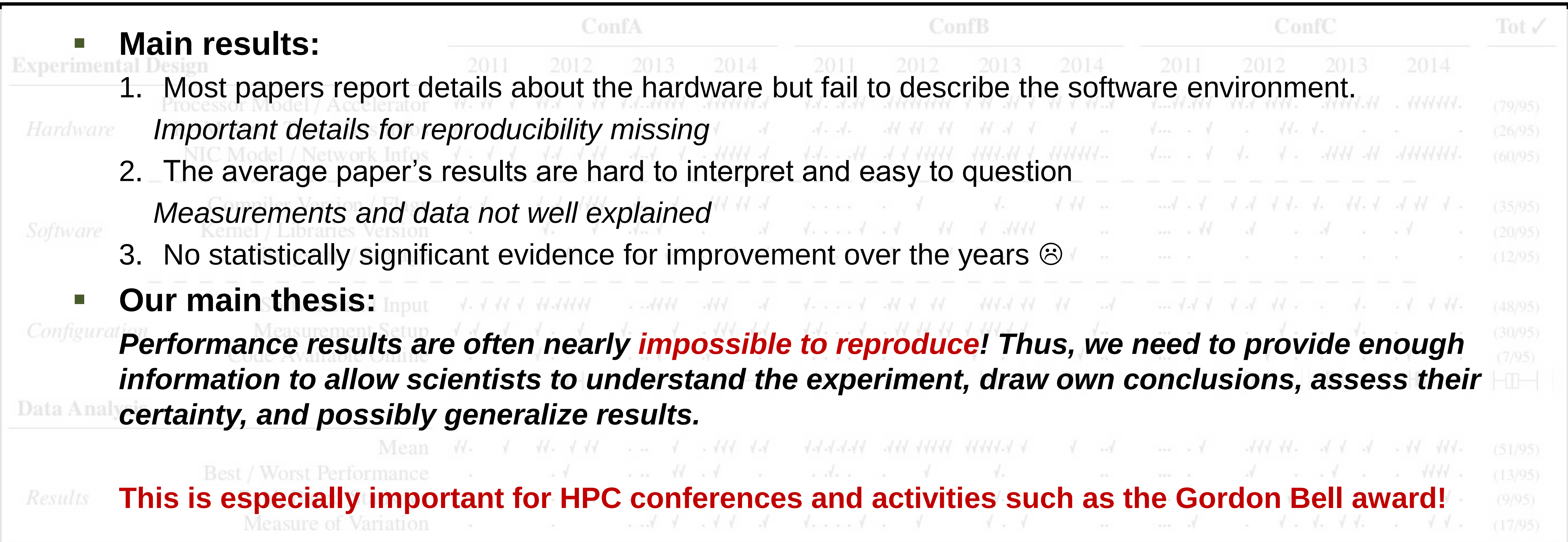
    “generalizable insights as gained from experiences with particular HPC machines/operations/applications/benchmarks, overall analysis of the status quo of a particular metric of the entire field or historical reviews of the progress of the field.”
  - Don’t expect novel insights

    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!**

spcl.inf.ethz.ch
@spcl_eth

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!