ETH zürich

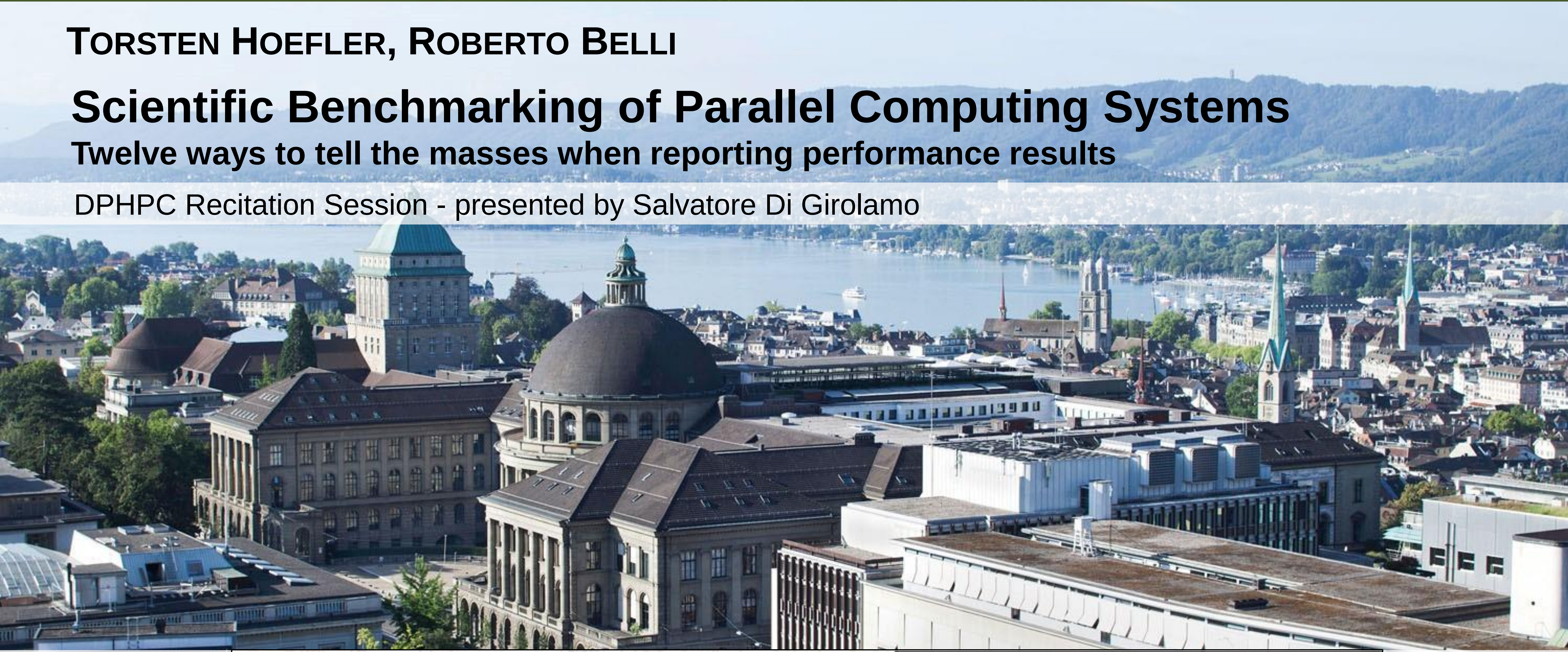
spcl.inf.ethz.ch
@spcl_eth

TORSTEN HOEFLER, ROBERTO BELLI

# Scientific Benchmarking of Parallel Computing Systems

## Twelve ways to tell the masses when reporting performance results

DPHPC Recitation Session - presented by Salvatore Di Girolamo



PASC17 Platform for Advanced Scientific Computing Conference

Lugano Switzerland | 26-28 June 2017

- CLIMATE & WEATHER
- SOLID EARTH
- LIFE SCIENCE
- CHEMISTRY & MATERIALS
- PHYSICS
- COMPUTER SCIENCE & MATHEMATICS
- ENGINEERING
- EMERGING DOMAINS

sighpc

acm

## 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!

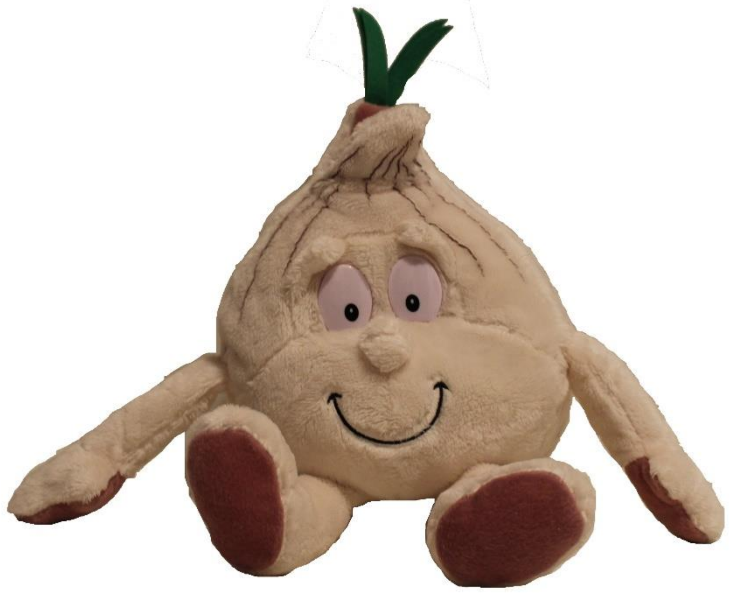
- **This talk should be entertaining!**

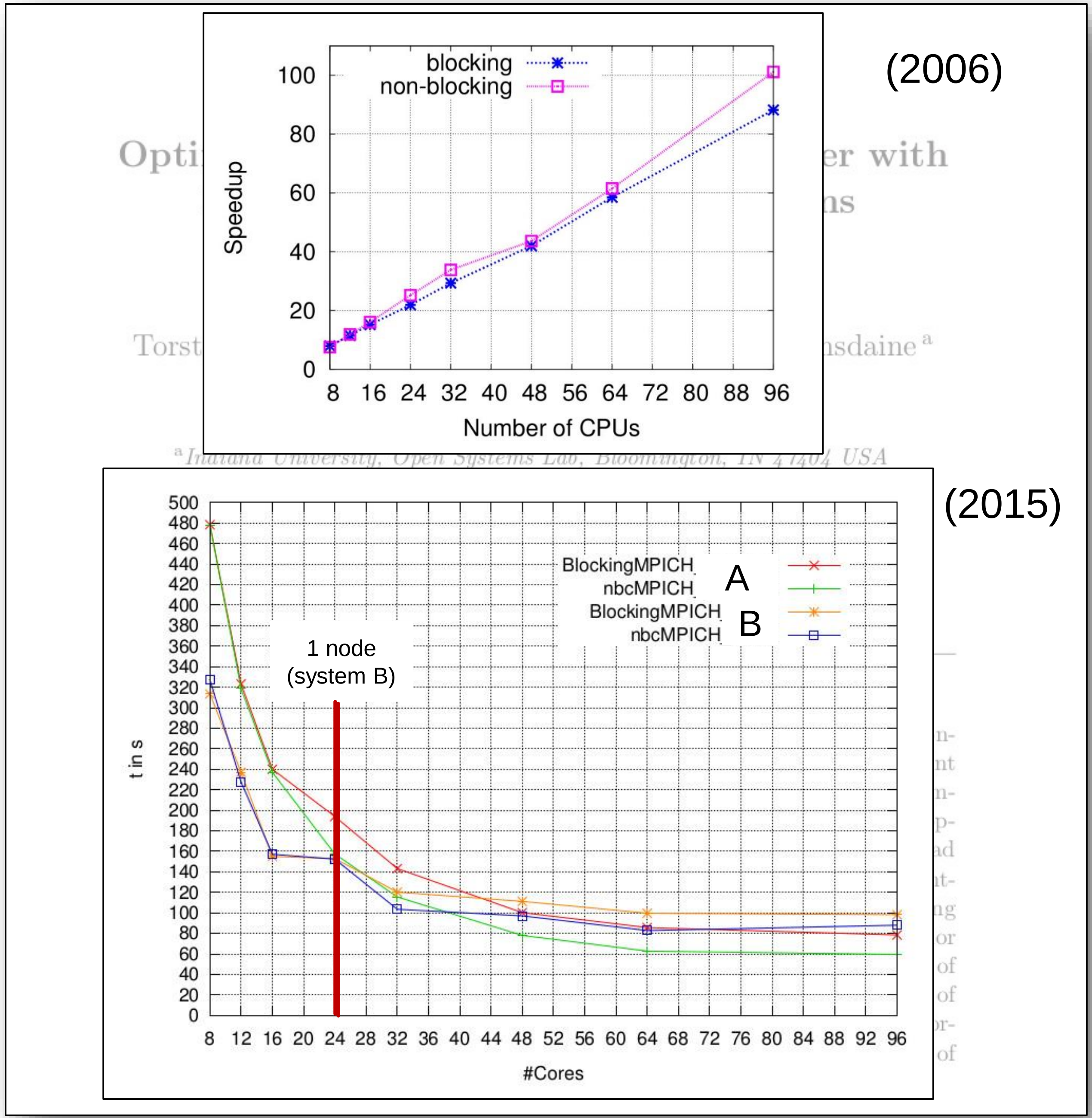
# How does Garth measure and report performance?

- **We are all 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 ☺





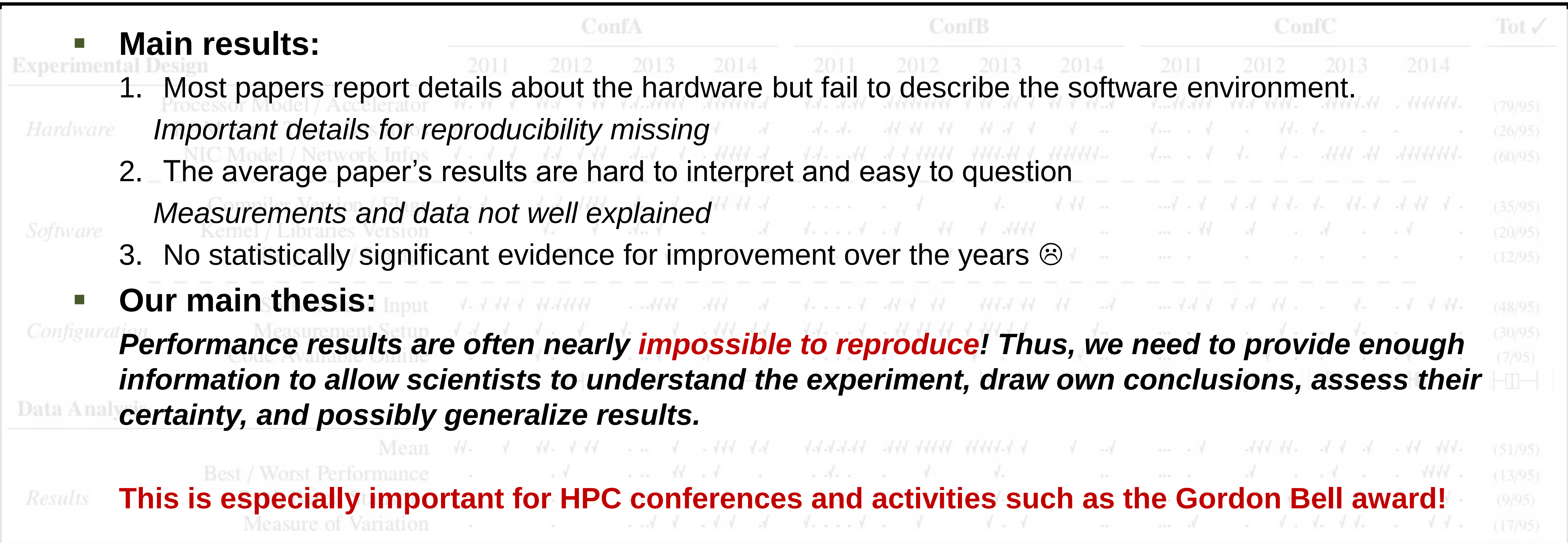
- **Original findings:**
  - If carefully tuned, NBC speeds up a 3D solver
    *Full code published*
  - $800^3$ domain – 4 GB (distributed) array
    *1 process per node, 8-96 nodes*
    *Opteron 246 (old even in 2006, retired now)*
  - Super-linear speedup for 96 nodes
    *~5% better than linear*

- **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)



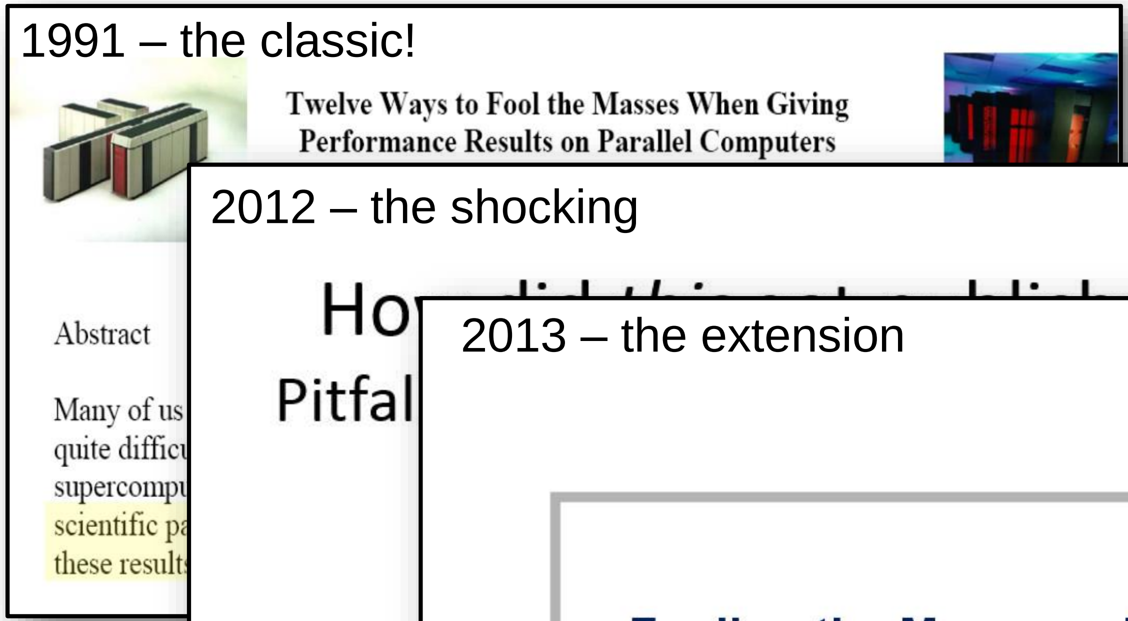
| Experimental Design | | ConfA 2011 | ConfA 2012 | ConfA 2013 | ConfA 2014 | ConfB 2011 | ConfB 2012 | ConfB 2013 | ConfB 2014 | ConfC 2011 | ConfC 2012 | ConfC 2013 | ConfC 2014 | Tot ✓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hardware | Processor Model / Accelerator | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (79/95) |
| | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (26/95) |
| | NIC Model / Network Infos | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (60/95) |
| Software | Compiler Version / Flags | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (35/95) |
| | Kernel / Libraries Version | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (20/95) |
| | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (12/95) |
| Configuration | [illegible] Input | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (48/95) |
| | Measurement Setup | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (30/95) |
| | Code Available Online | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (7/95) |
| Data Analysis | | | | | | | | | | | | | | |
| Results | Mean | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (51/95) |
| | Best / Worst Performance | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (13/95) |
| | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (9/95) |
| | Measure of Variation | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | (17/95) |

- **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?

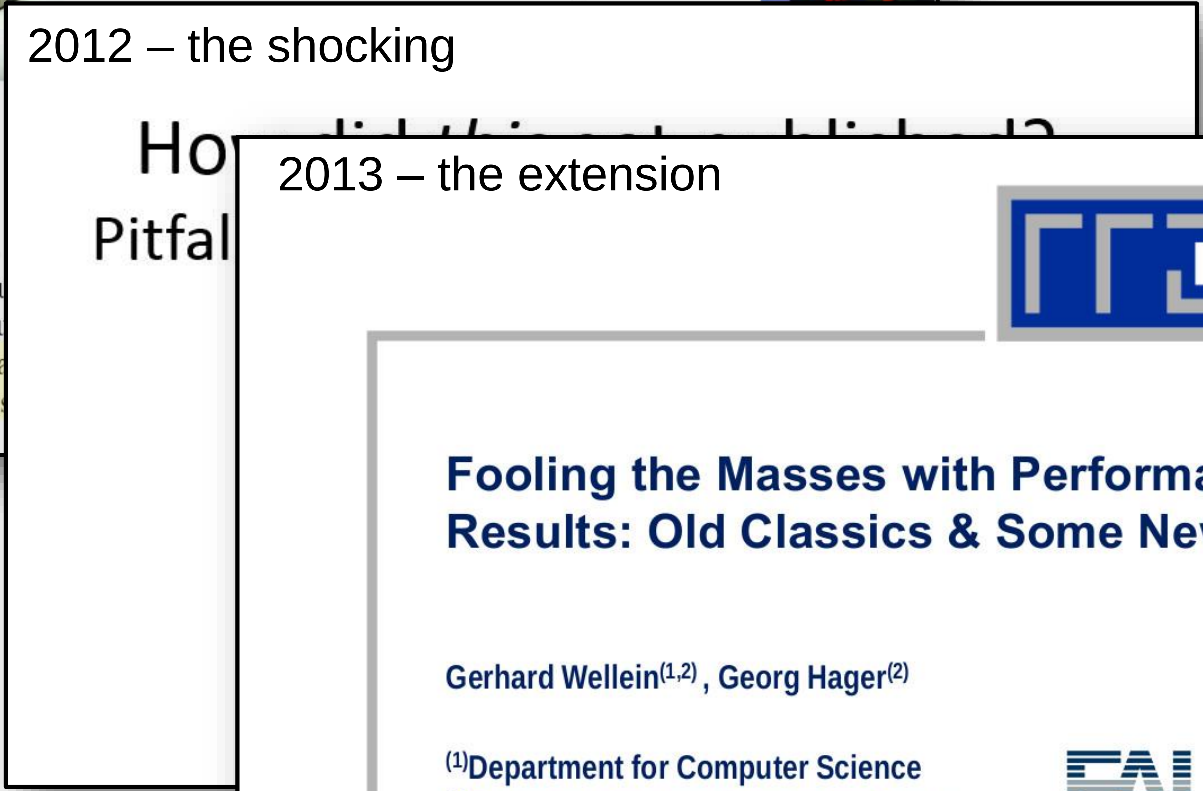


1991 – the classic!

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

Abstract

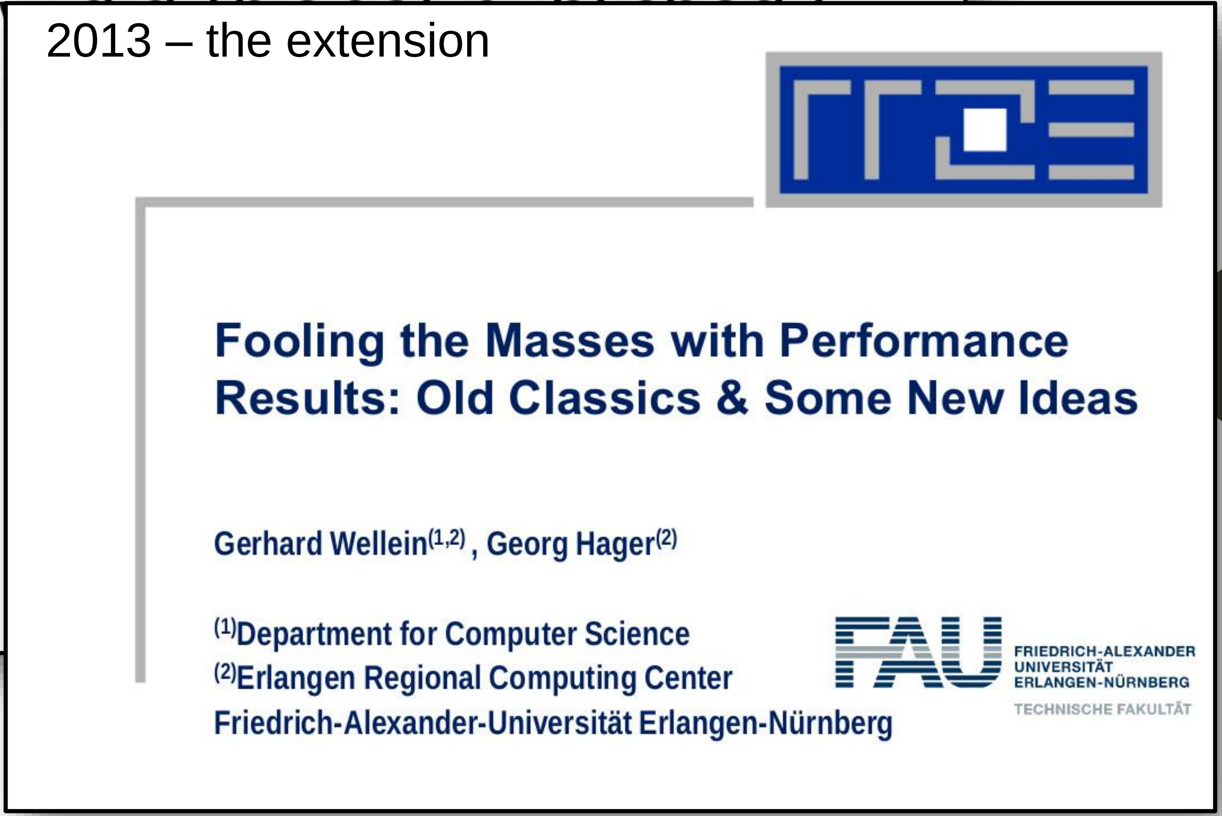
Many of us
quite diffic
supercompu
scientific p
these result



2012 – the shocking

How did this get published?

Pitfal



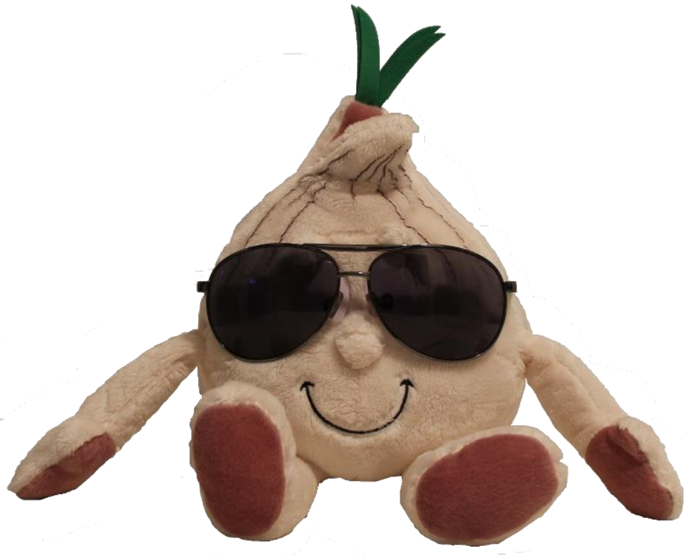
2013 – the extension

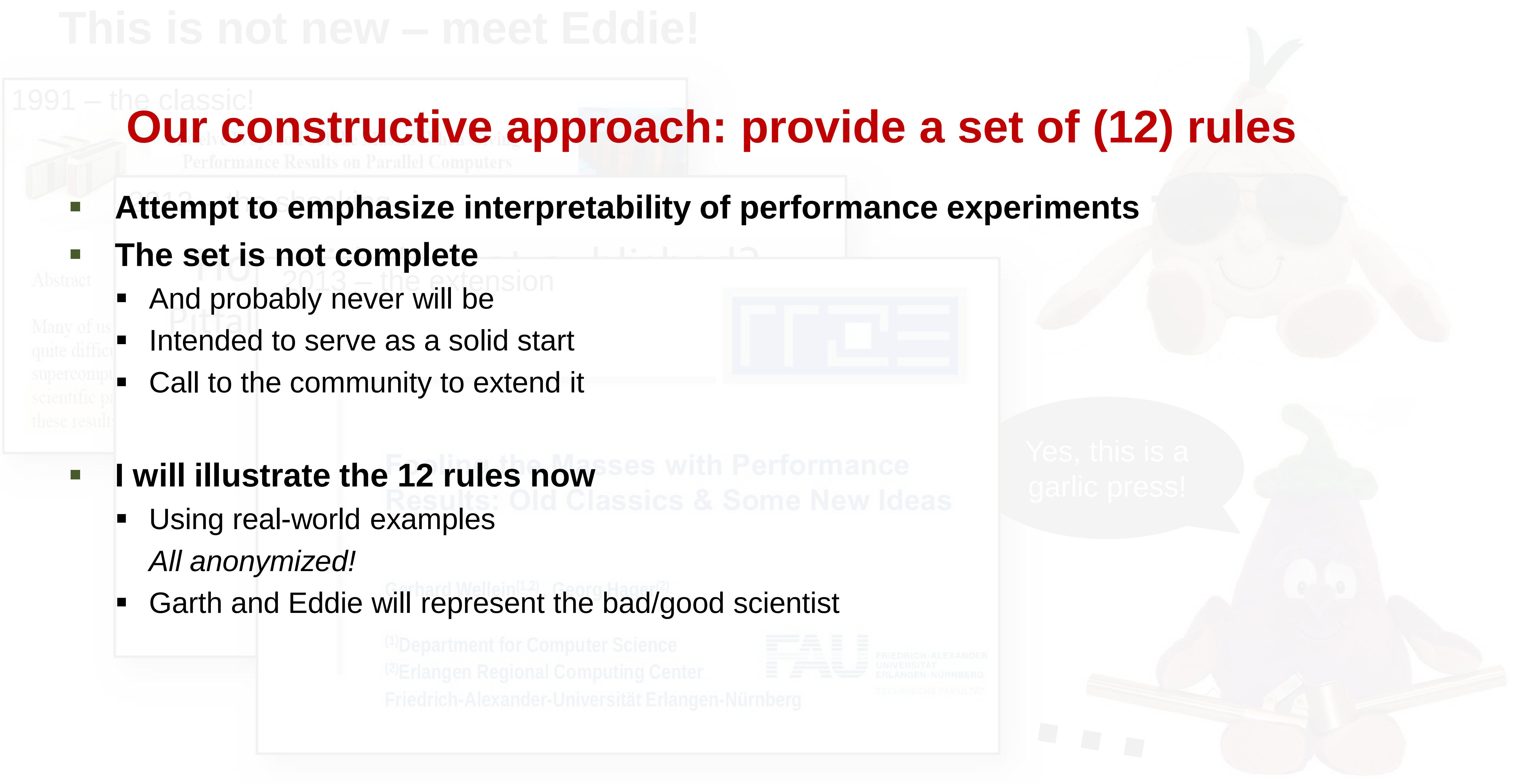
Fooling the Masses with Performance Results: Old Classics & Some New Ideas

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



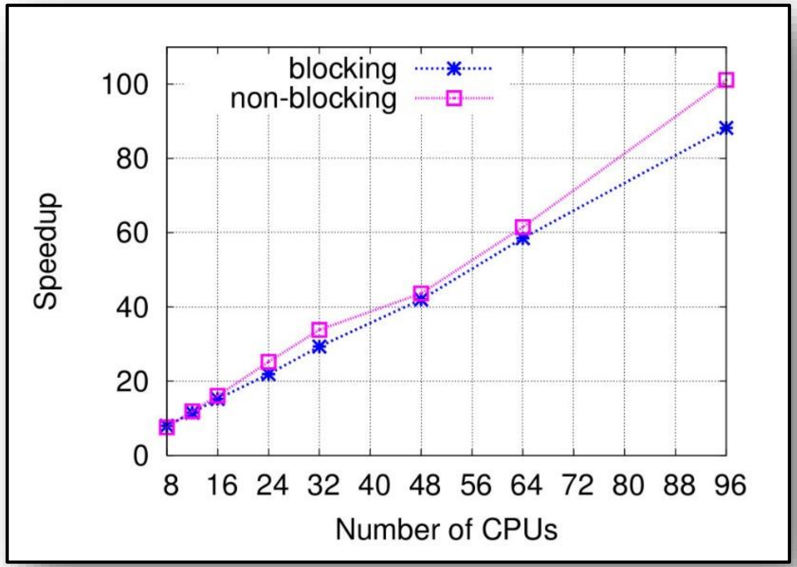

This is not new – meet Eddie!
1991 – the classic!
Performance Results on Parallel Computers
2013 – the extension
Fooling the Masses with Performance Results: Old Classics & Some New Ideas
Gerhard Wellein(1,2), Georg Hager(2)
(1)Department for Computer Science
(2)Erlangen Regional Computing Center
Friedrich-Alexander-Universität Erlangen-Nürnberg
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



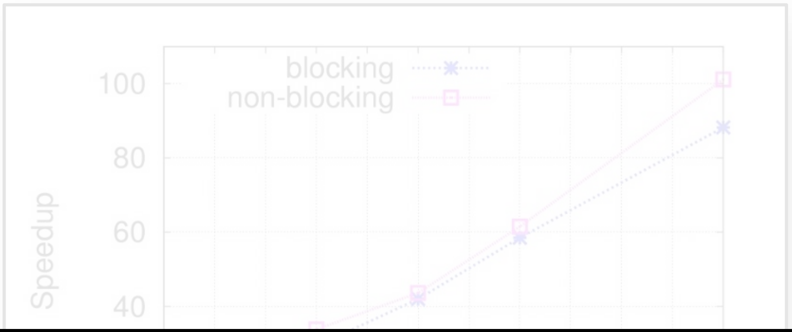
- **Most common and oldest-known issue**
  - First seen 1988 – also included in Bailey's 12 ways
  - 39 papers reported speedups

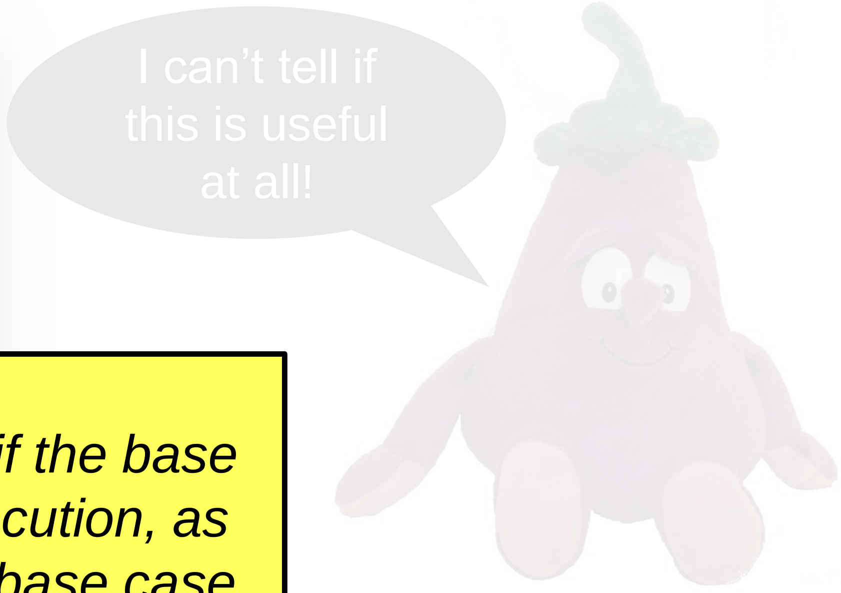
    *15 (38%) did not specify the base-performance ☹*
  - Recently rediscovered in the "big data" universe

    *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





**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

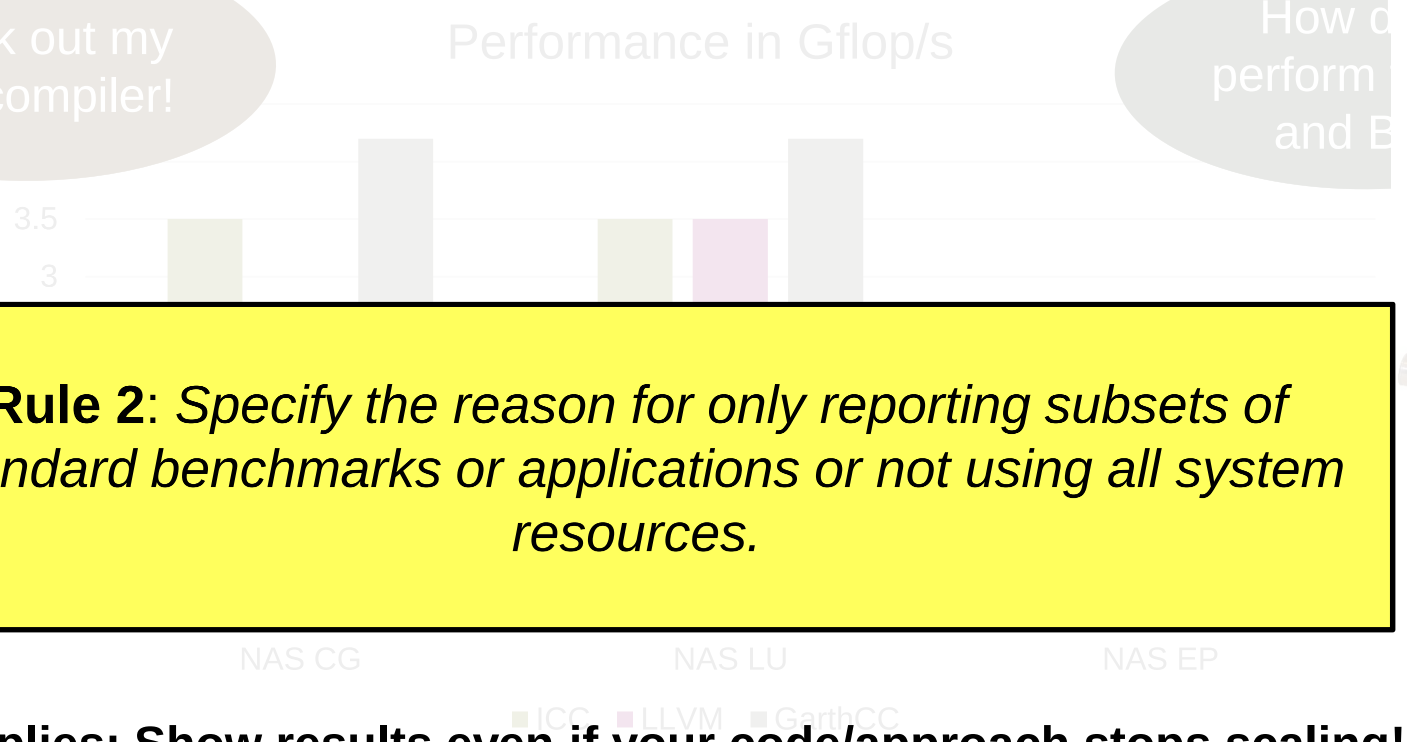
    *A. Rowstron et al.: Nobody ever got fired for using Hadoop on a cluster, HotCDP 2012*

    *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!



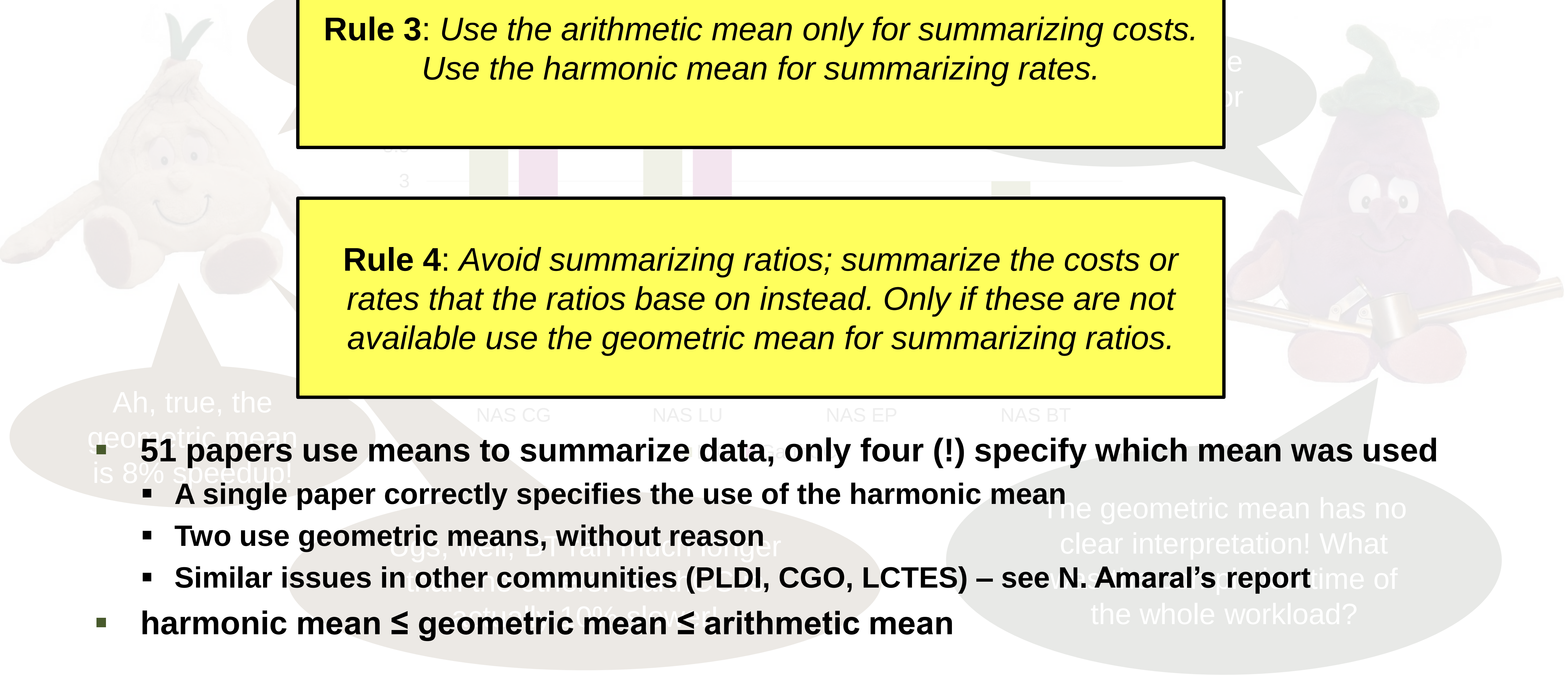
How d perform and B

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!**

# The mean parts of means – or how to summarize data

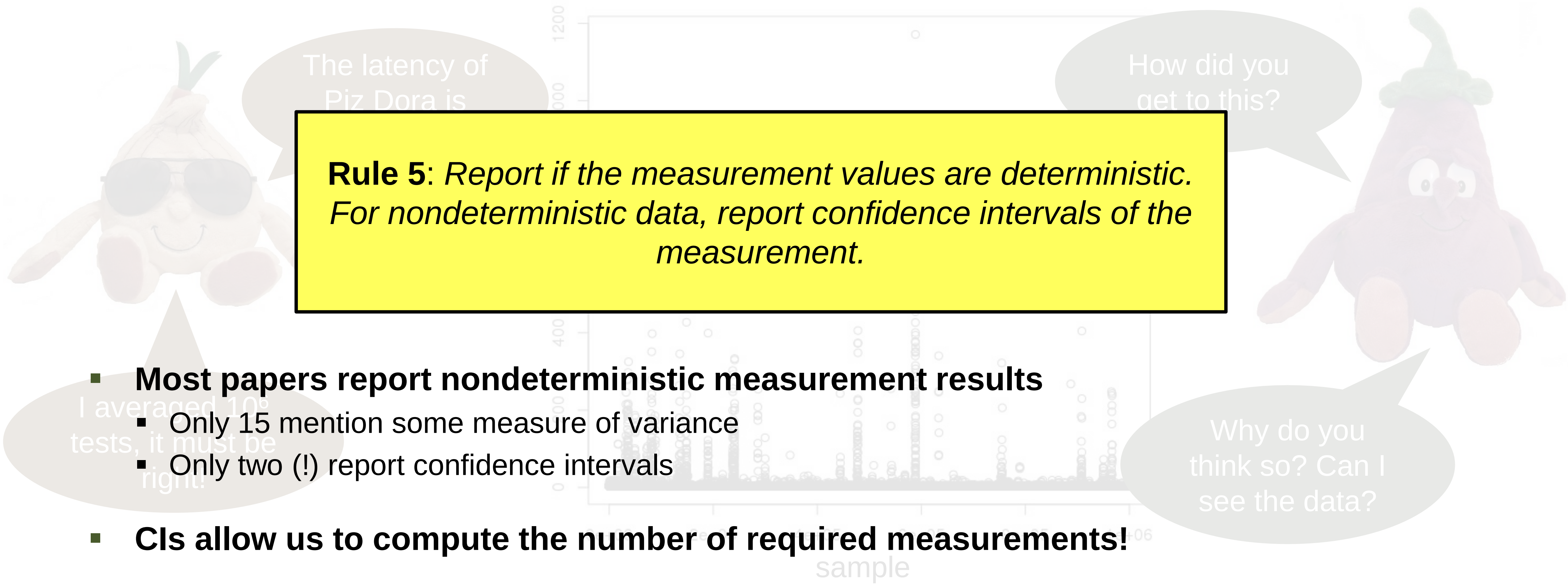

3
NAS CG
NAS LU
NAS EP
NAS BT
Ah, true, the geometric mean is 8% speedup!
The geometric mean has no clear interpretation! What was the completion time of the whole workload?

**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

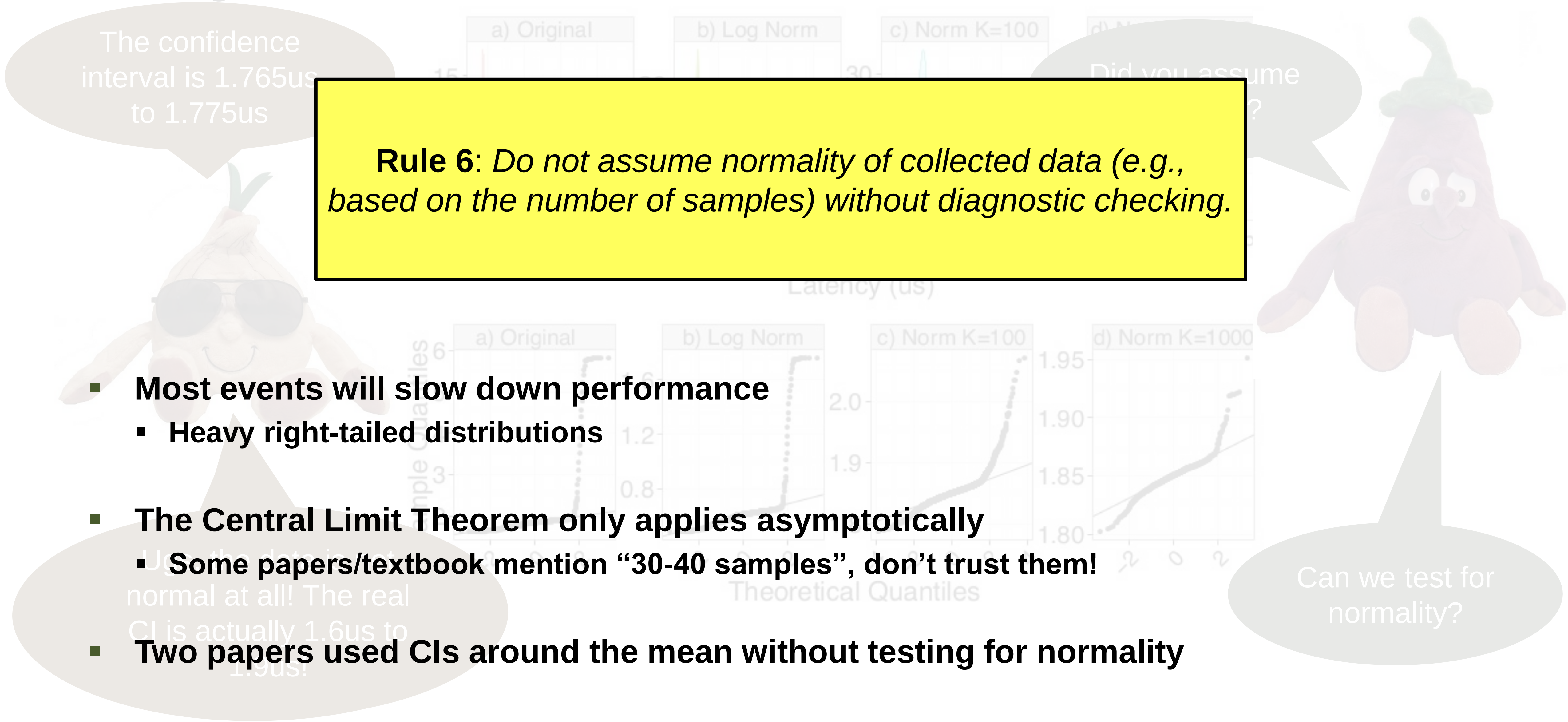


**Rule 5**: *Report if the measurement values are deterministic. For nondeterministic data, report confidence intervals of the measurement.*

- **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

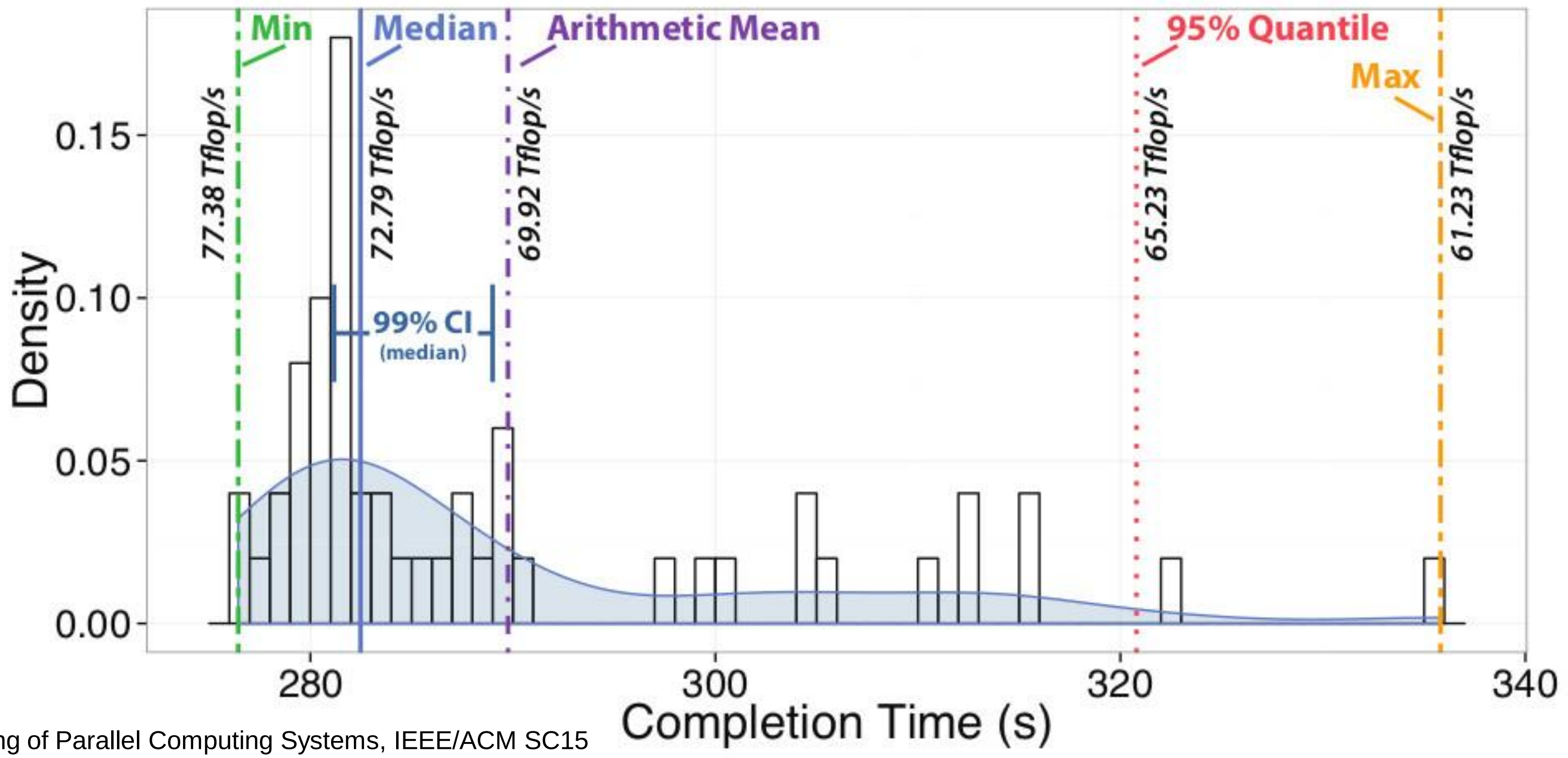


**Rule 6**: *Do not assume normality of collected data (e.g., based on the number of samples) without diagnostic checking.*

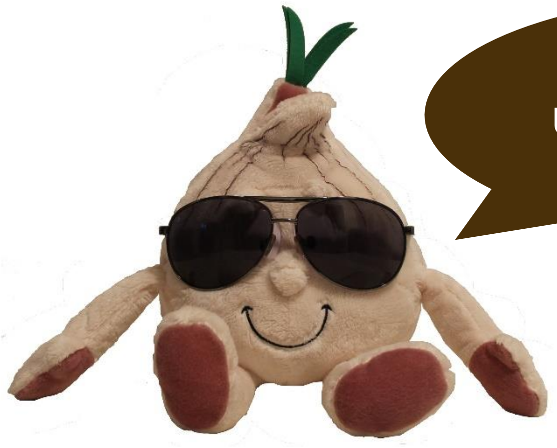
- **Most events will slow down performance**
  - **Heavy right-tailed distributions**
- **The Central Limit Theorem only applies asymptotically**
  - **Some papers/textbook mention “30-40 samples”, don’t trust them!**
- **Two papers used CIs around the mean without testing for normality**

## 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

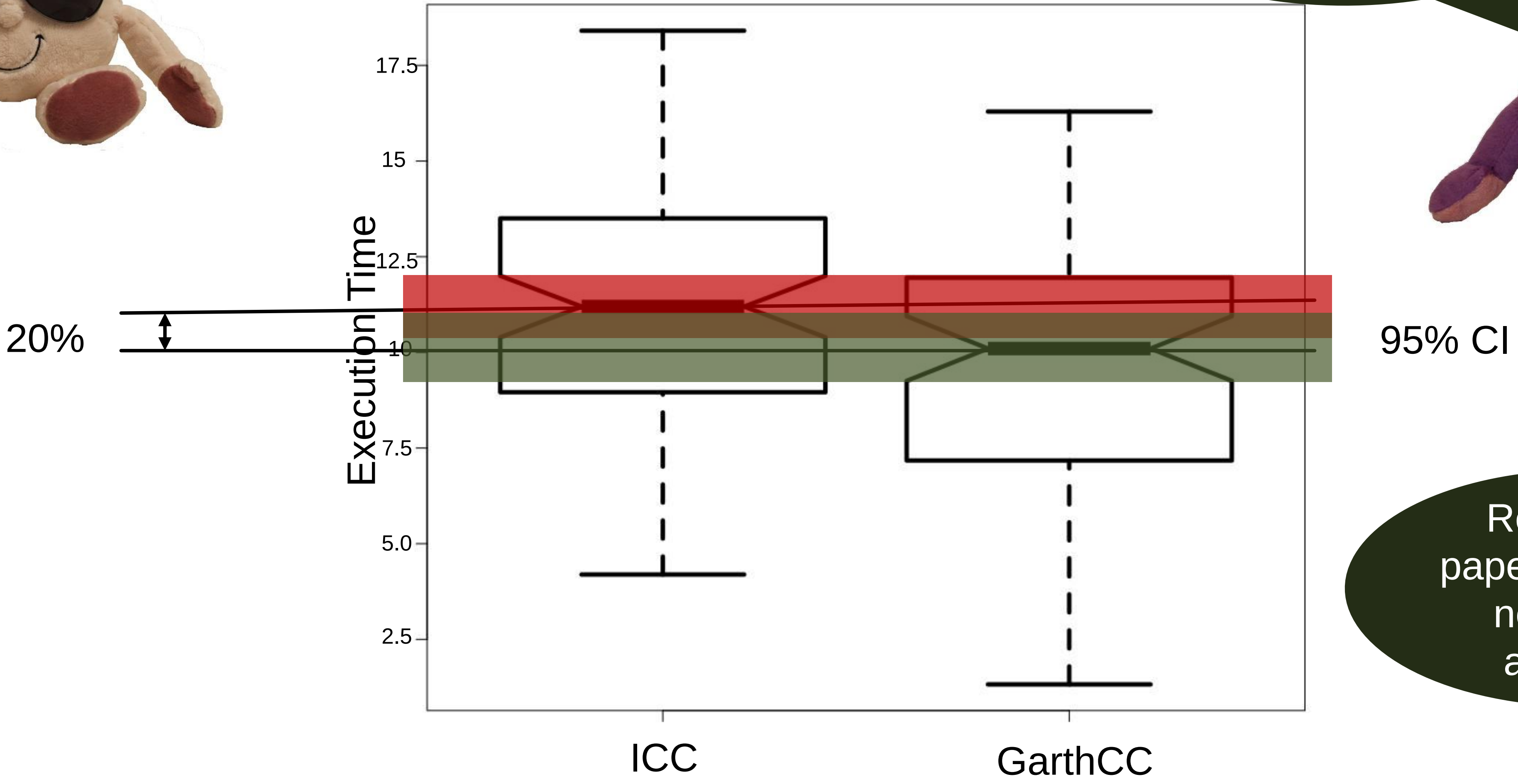


# Comparing nondeterministic measurements



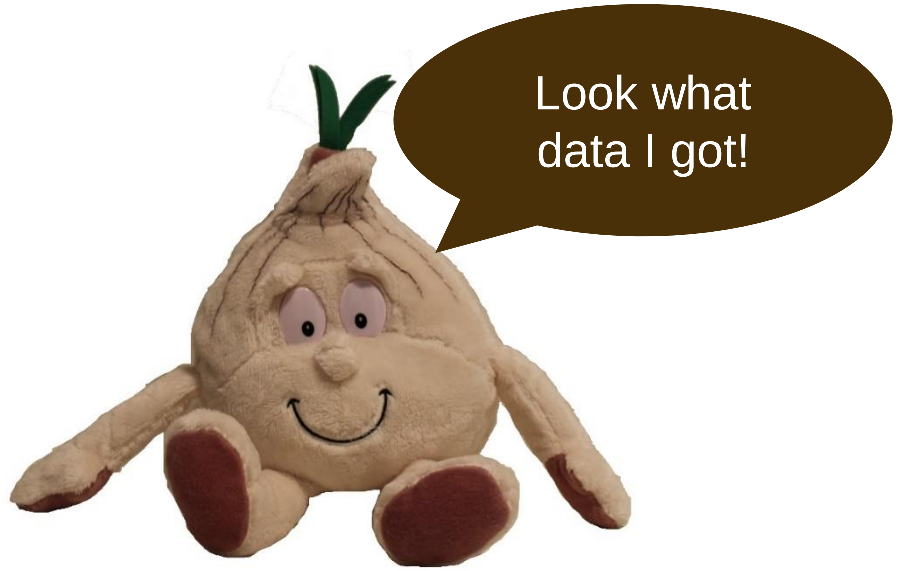
I saw variance using GarthCC as well!

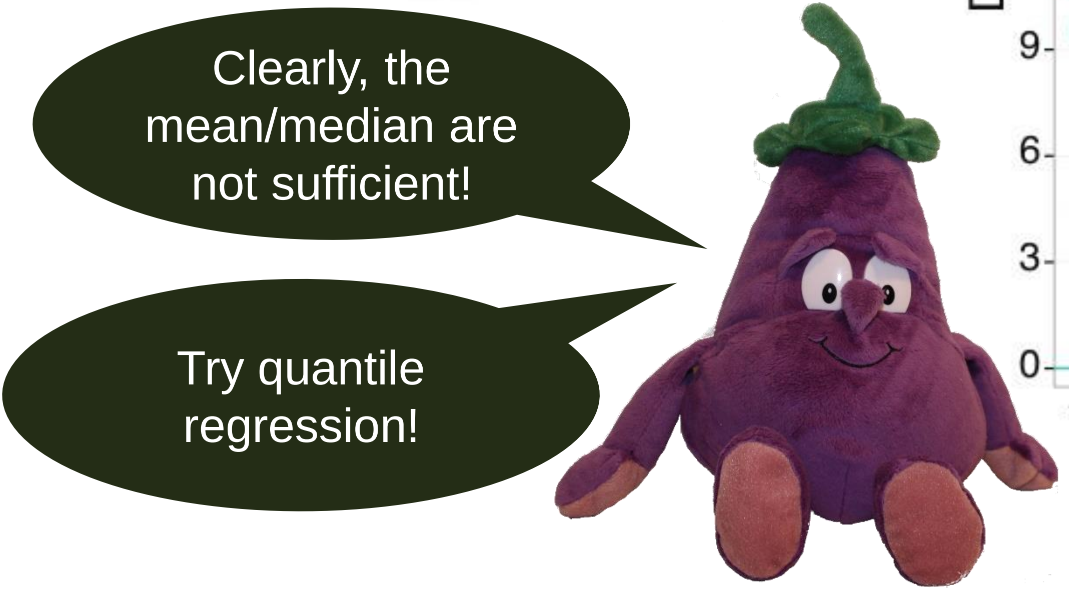
Show me the data!

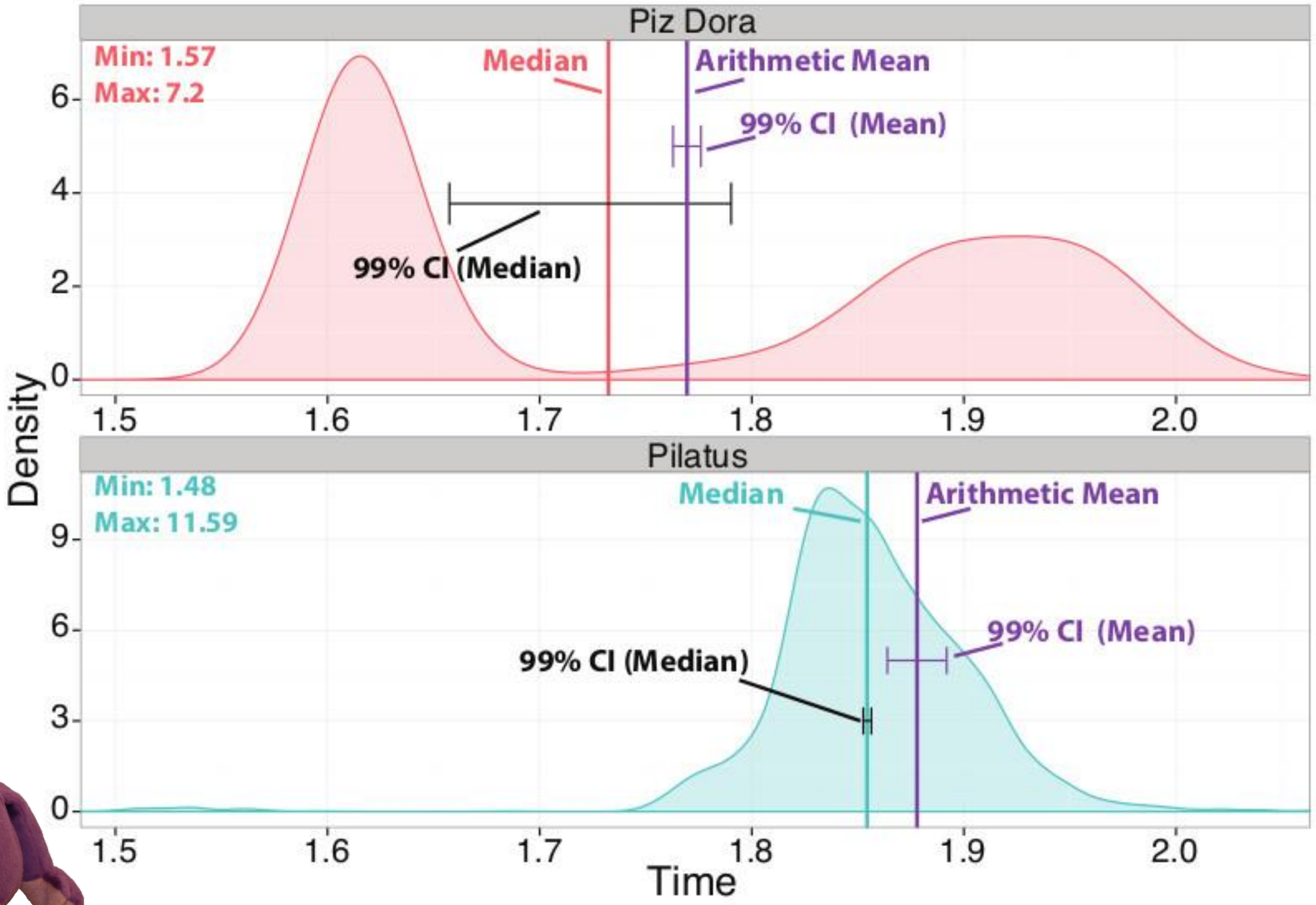


Retract the paper! You have not shown anything!

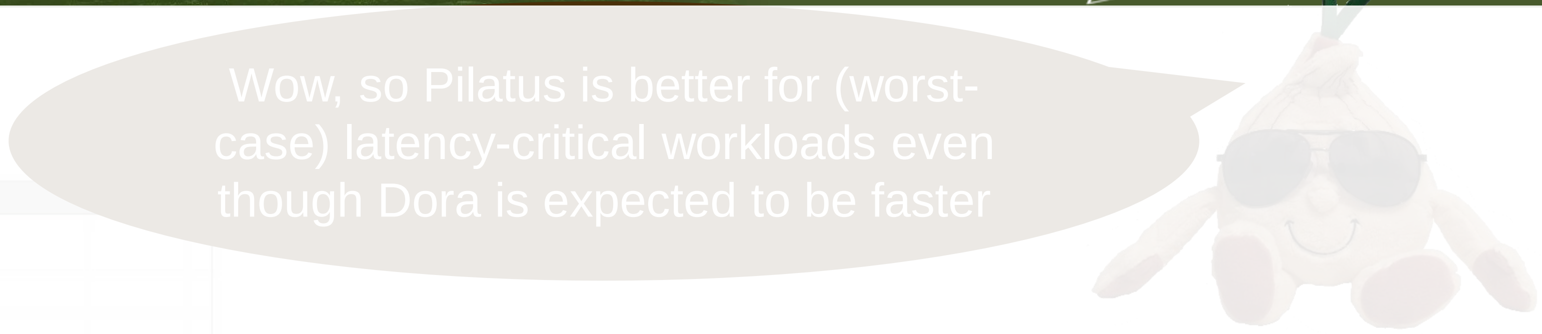
## What if the data looks weird!?

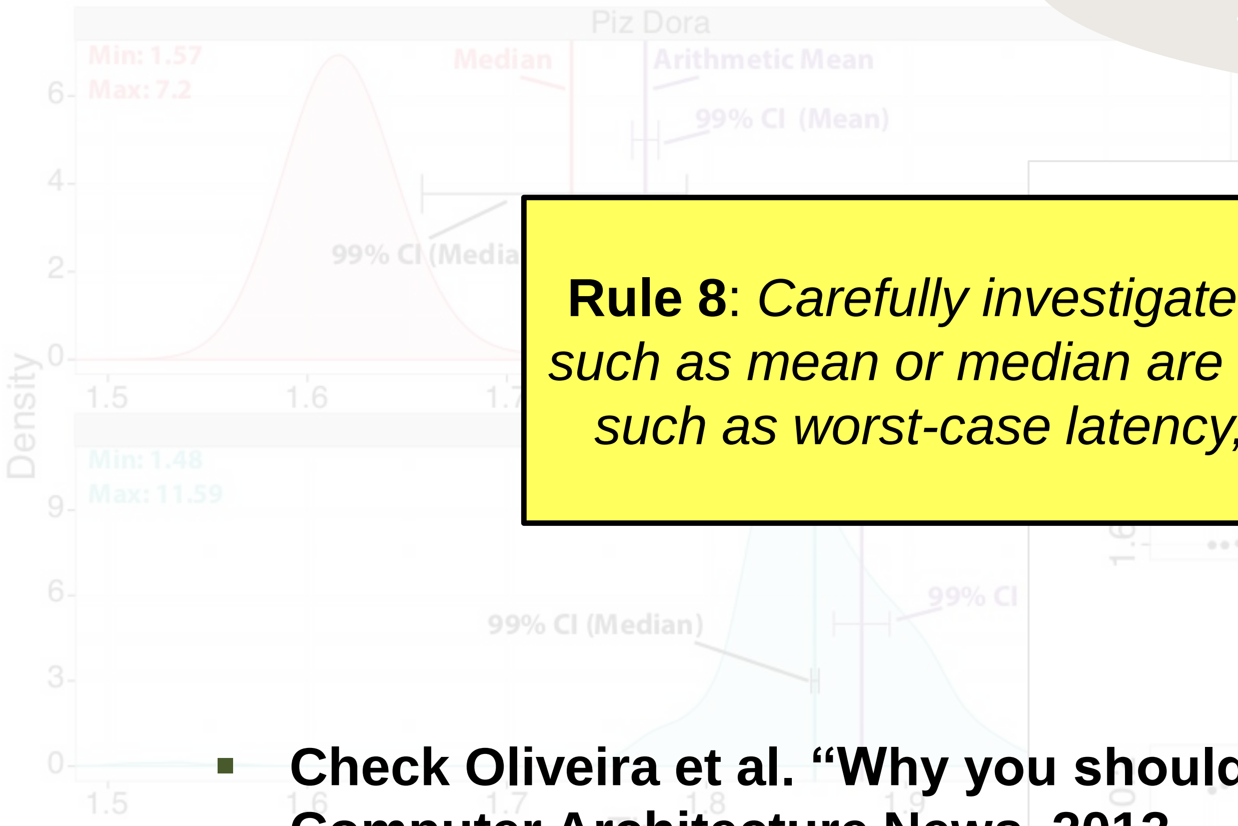


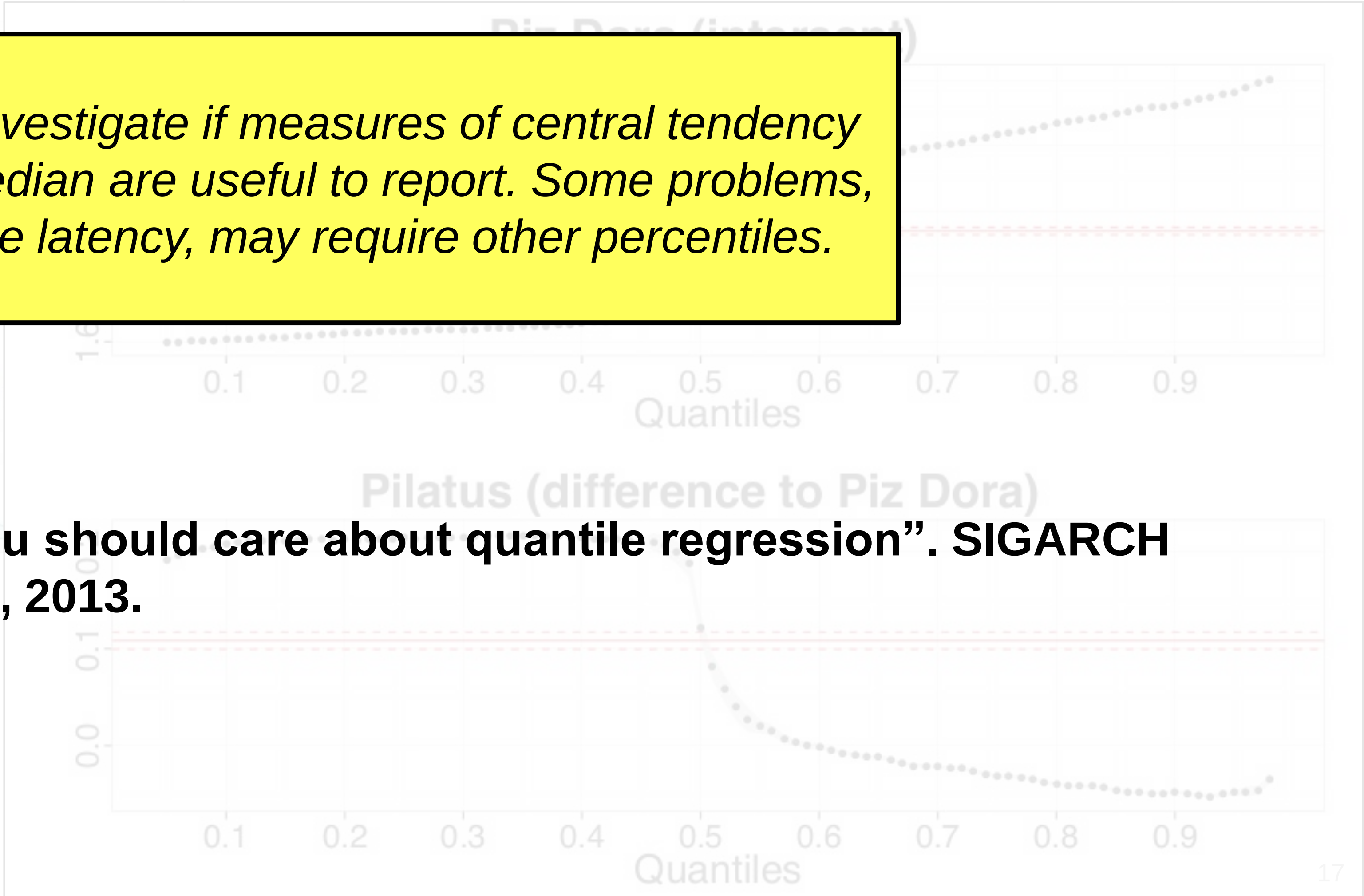




## Quantile Regression


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.**

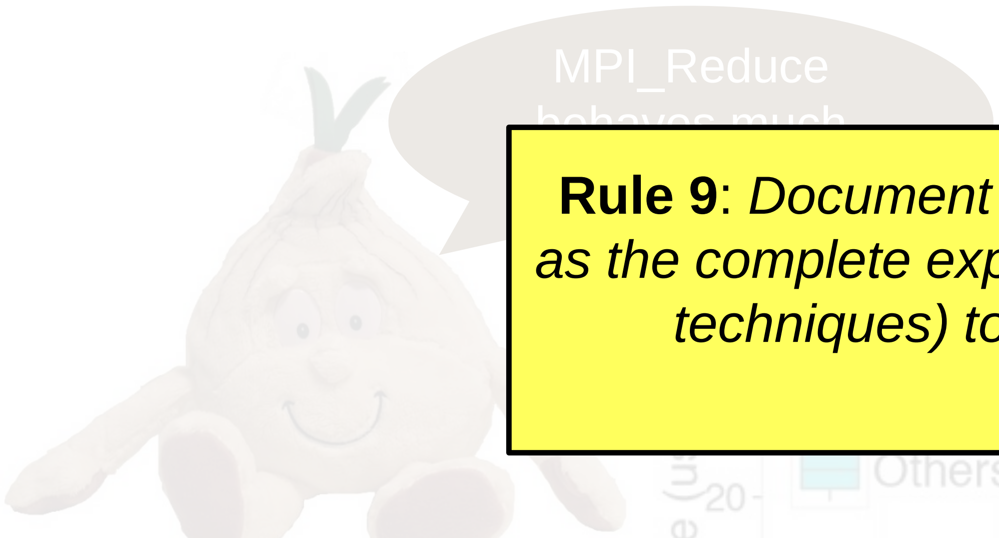
## How many measurements are needed?

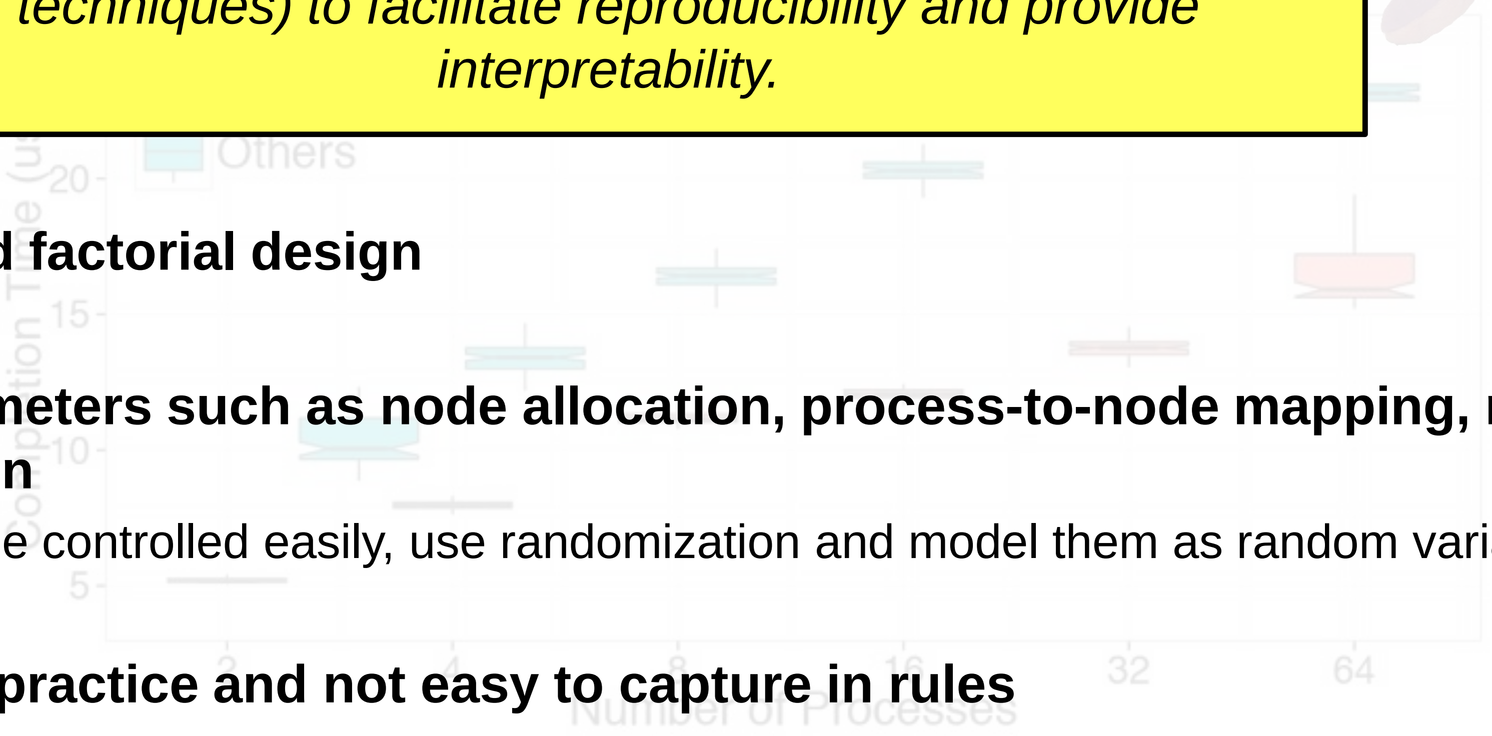
- **Measurements can be expensive!**
  - Yet necessary to reach certain confidence
- **How to determine the minimal number of measurements?**
  - Measure until the confidence interval has a certain acceptable width
  - For example, measure until the 95% CI is within 5% of the mean/median
  - Can be computed 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.*

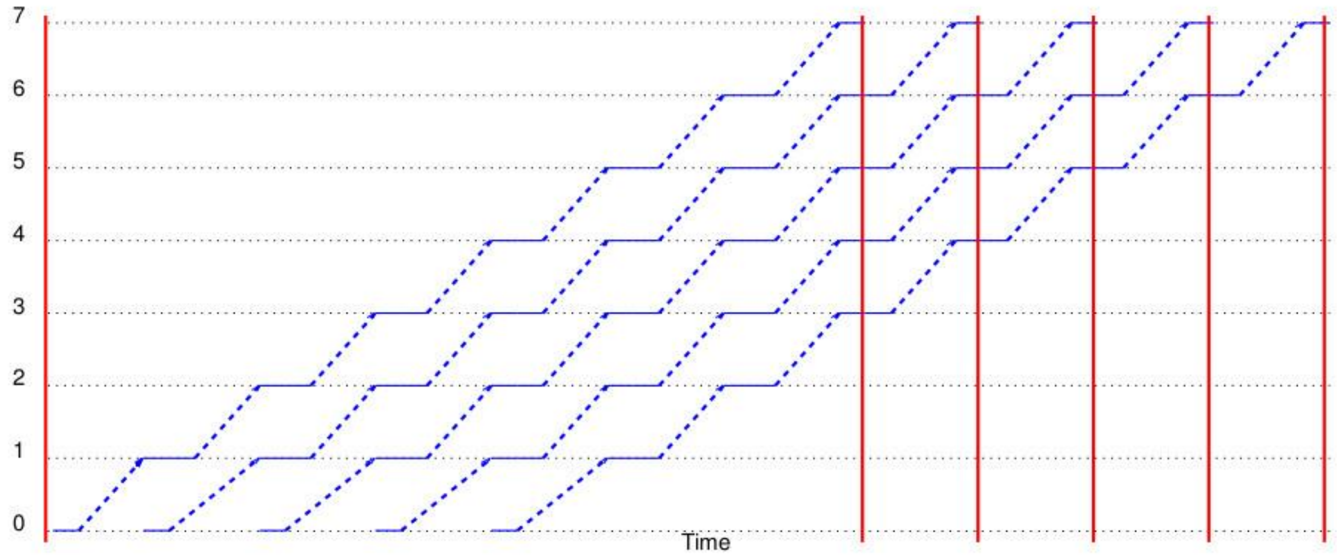
- **We recommend factorial design**
- **Consider parameters such as node allocation, process-to-node mapping, network or node contention**
  - If they cannot be controlled easily, use randomization and model them as random variable
- **This is hard in practice and not easy to capture in rules**

## 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;
```

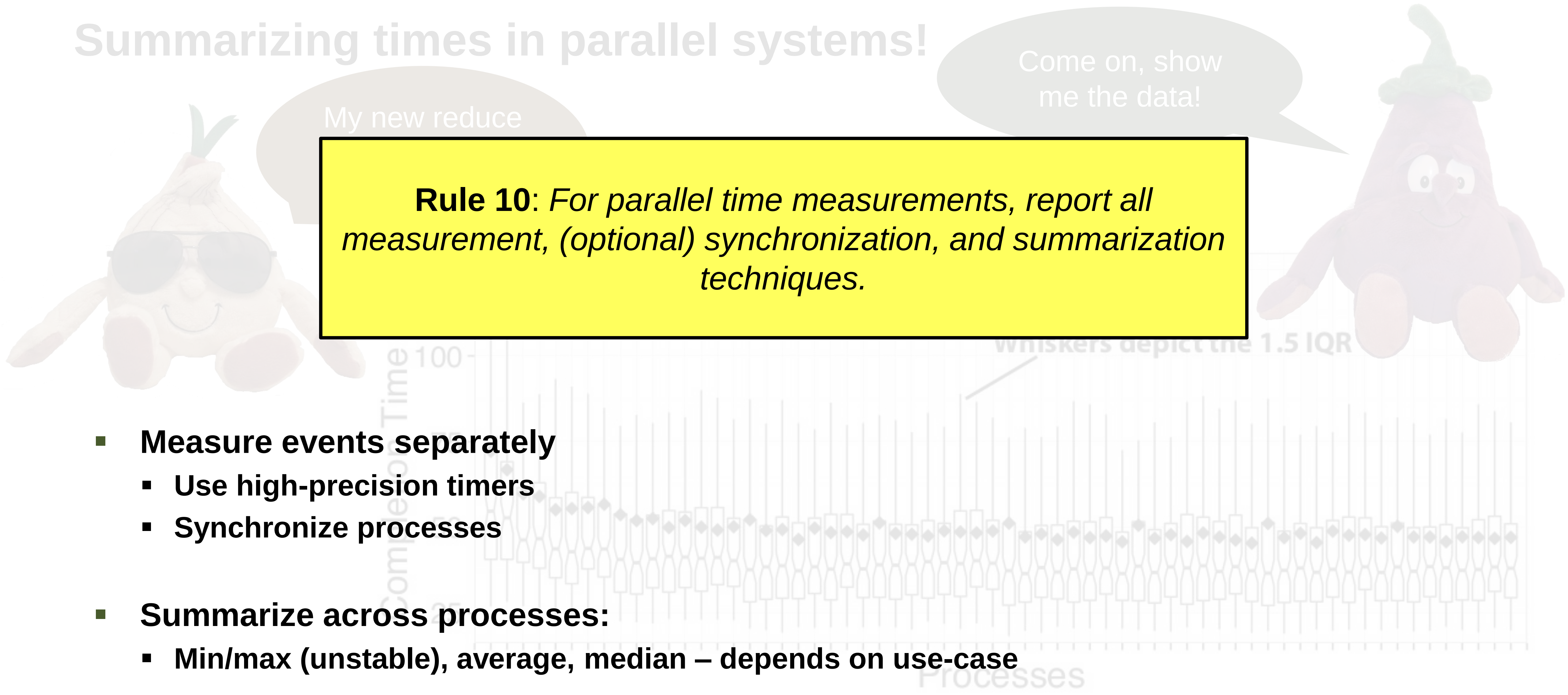


…

That's nonsense!

Measure each operation separately!

## Summarizing times in parallel systems!

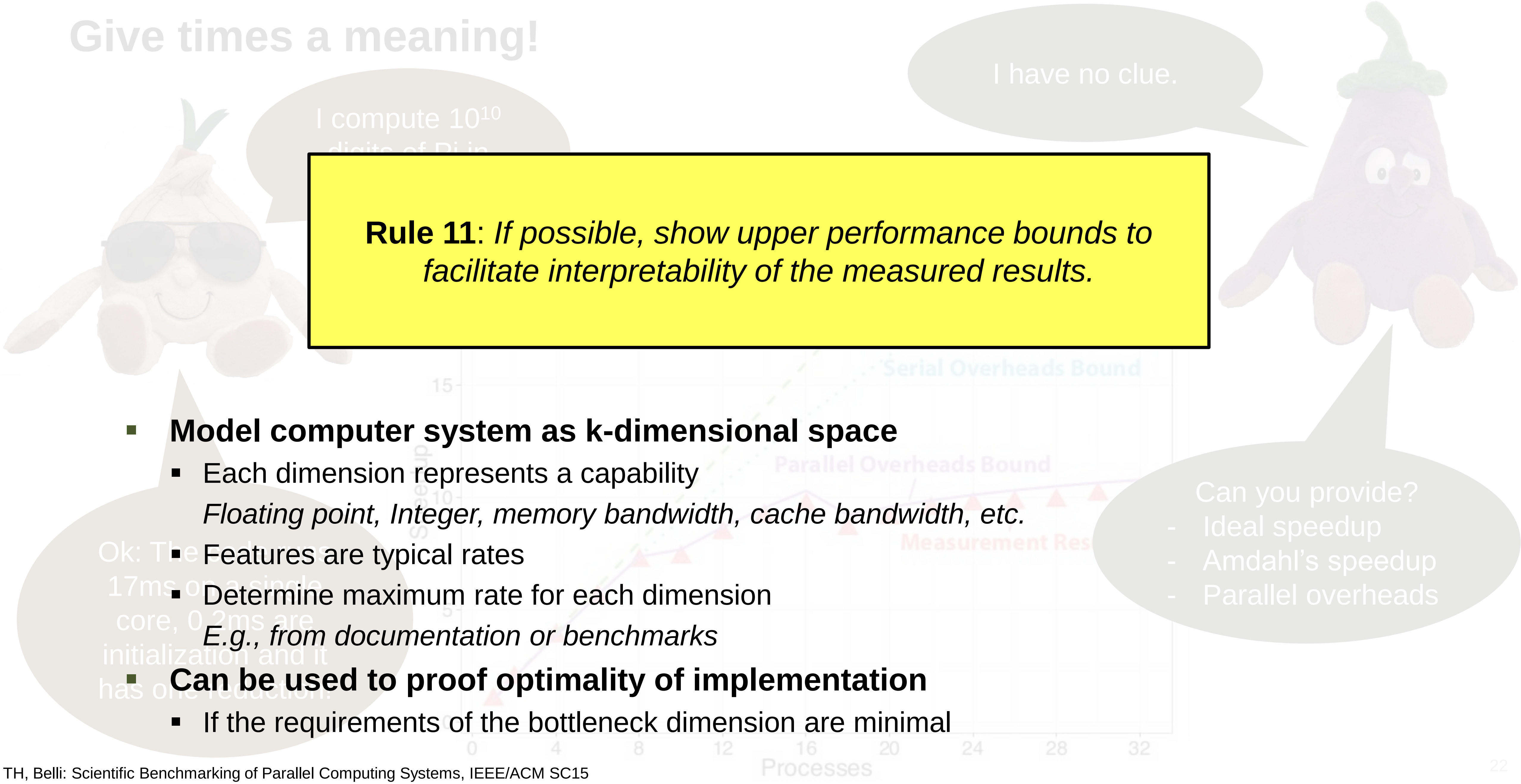


**Rule 10**: *For parallel time measurements, report all measurement, (optional) synchronization, and summarization techniques.*

- **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!

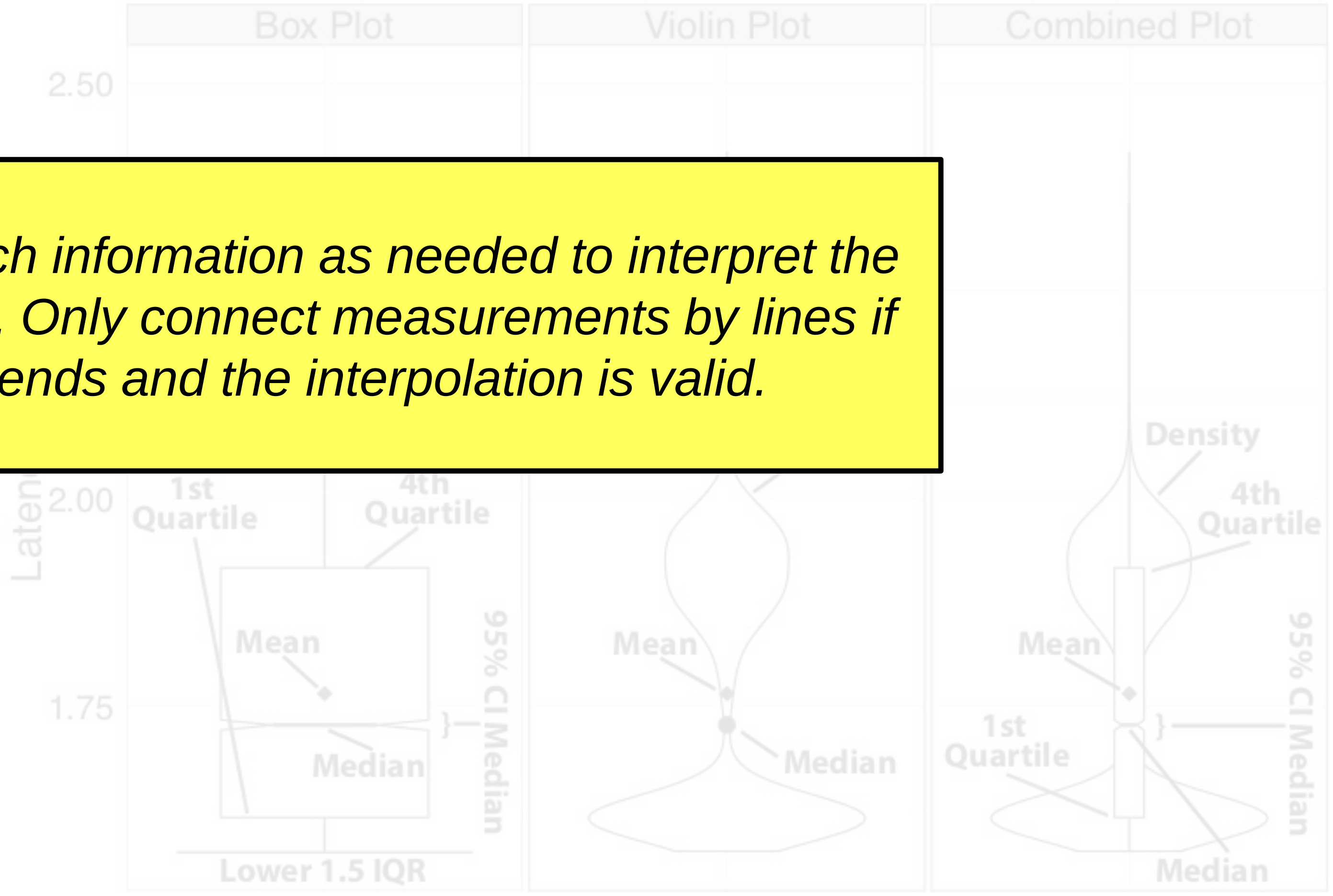


**Rule 11**: *If possible, show upper performance bounds to facilitate interpretability of the measured results.*

- **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

## 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.

## 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/.

# Event: LLVM Compiler and Code Generation Social

**When:** ***08.12.2016 19:00***

**Where:** ***ETH Zurich, CAB, E72***

**The LLVM Compiler and Code Generation Social is a meetup to discuss compilation and code generation topics, with a special focus on LLVM, clang, Polly, and related projects. If you are interested in generating code for a variety of architectures, (static) program analyses for real world C/C++/OpenCL/CUDA programs, building your own programming language, register allocation and instruction selection, software hardening techniques, have an idea for a great optimization, or want to target GPUs and FPGA, .... This event is for you!**

**Our primary focus are free discussions between interested people (+ beer and food). This is a great opportunity to get and discuss project ideas or to just learn about what people at ETH and around Zurich are doing.**

**Contact:** Tobias Grosser *(https://www.inf.ethz.ch/personal/tgrosser/)*