TORSTEN HOEFLER

Performance Reproducibility in HPC and Deep Learning

Numerical Reproducibility at Exascale Workshop (NRE2019), ISC'19, Frankfurt, Germany

WITH CONTRIBUTIONS FROM ROBERTO BELLI, TAL BEN-NUN, DAN ALISTARH, YOSUKE OYAMA, CEDRIC RENGGLI, AND OTHERS AT SPCL AND IST AUSTRIA

Disclaimer(s)

- **This is an experience talk (paper 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!**

OPINION

PNAS, Feb. 2015

“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

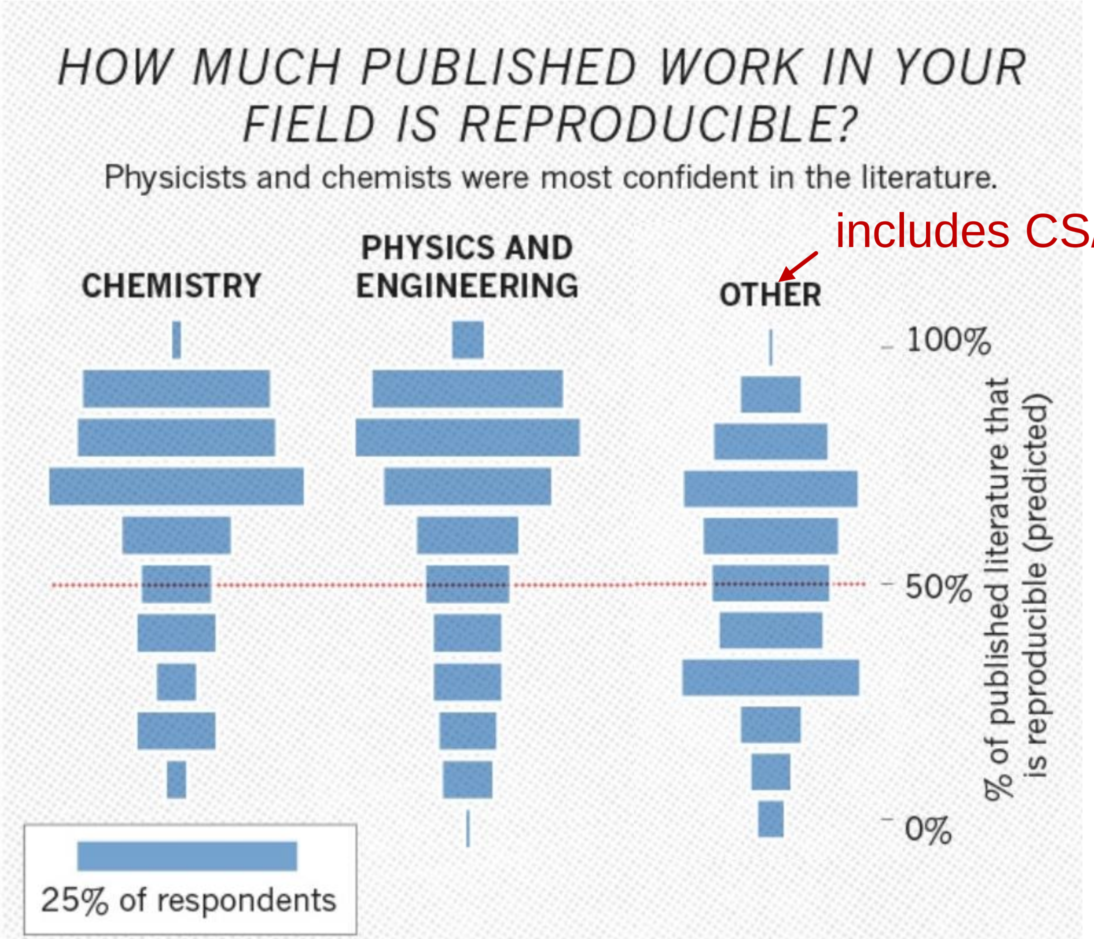
Reproduci... results—ar... experime... result—ar... of success... findings are the primary means by which scientific evidence accumulates for or against a hypothesis. Yet, of late, there has been a crisis of confidence among researchers worried about the rate at which studies are either

been some very public failings of reproducibility across a range of disciplines from cancer 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 research (6). The result is that much of the

Unfortunately, the mere reproducibility of computational results is insufficient to address the replication crisis because even a reproducible analysis can suffer from many problems—confounding from omitted variables, poor study design, missing data—that threaten the validity and useful interpretation of the results. Although improving the reproducibility of research may increase the rate at which flawed analyses are uncovered, as recent high-profile examples have demonstrated (4), it does not change the fact that

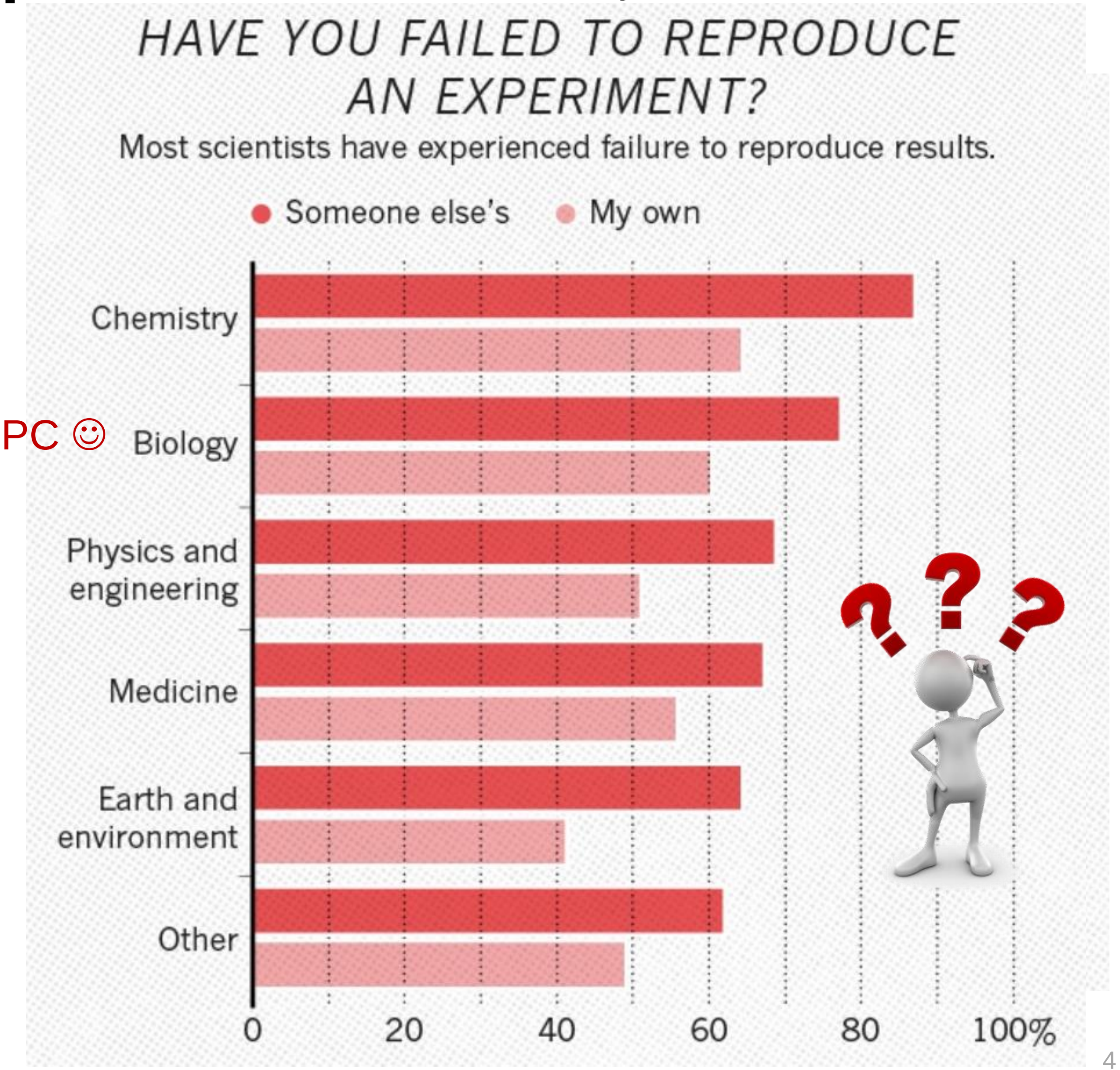
Reproducibility and replicability?

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



includes CS/HPC ☺

Nature, May 2016



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

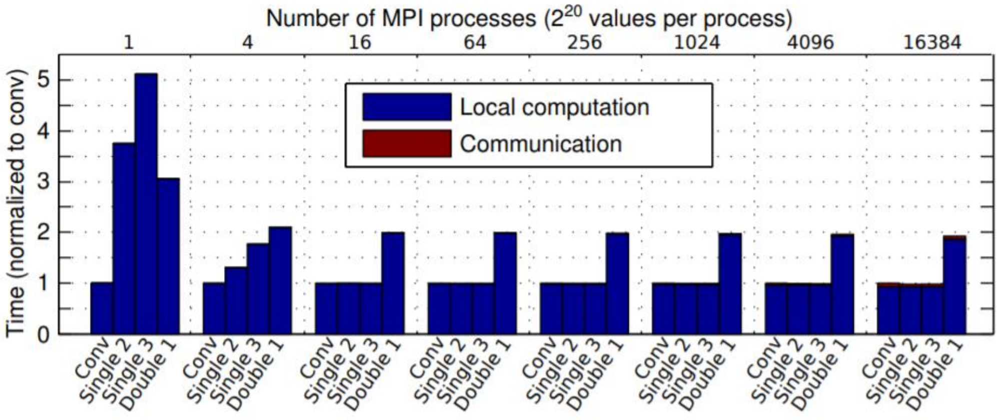


Figure 8. Performance comparison of conventional reduction performed with MKL (*Conv*), single-sweep reduction with two levels (*Single2*), with three levels (*Single3*) and double-sweep reduction with 1 level (*Double 1*) on varying number of processes, each owning 2^{20} double-precision values,

But what if performance is your science result?

(2006)

- **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

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!

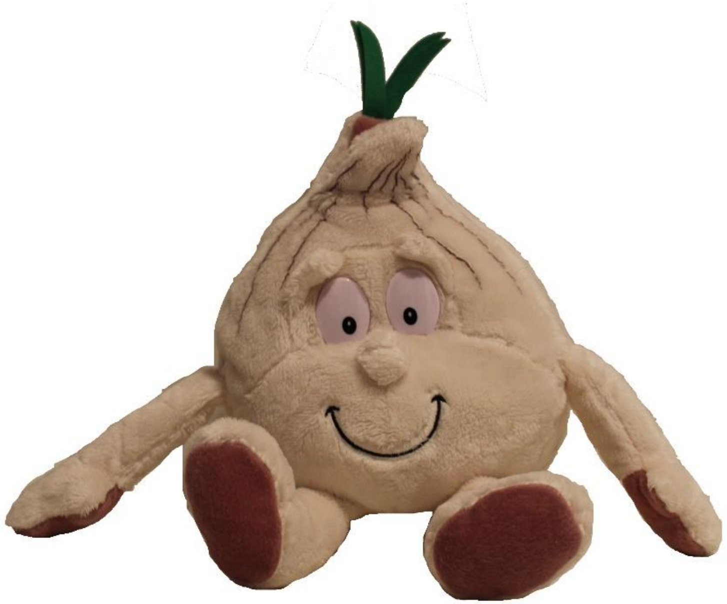
Interpretability: *We call an experiment interpretable if it provides enough information to allow scientists to understand the experiment, draw own conclusions, assess their certainty, and possibly generalize results.*

How does Garth measure and report performance?

- **We are 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 ☺



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)

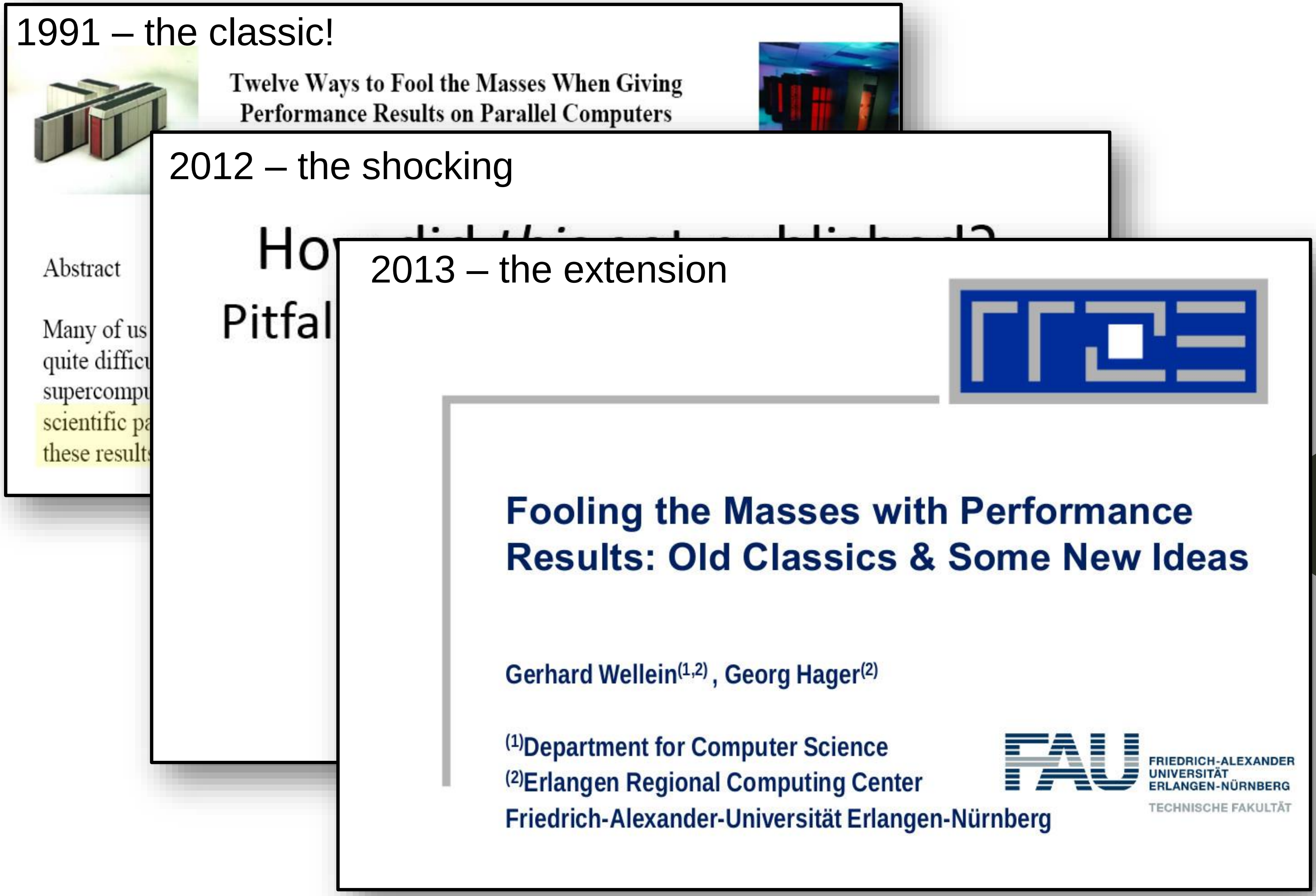
- **Main results:**
  1. Most papers report details about the hardware but fail to describe the software environment.
     Important details for reproducibility missing
  2. The average paper's results are hard to interpret and easy to question
     Measurements and data not well explained
  3. No statistically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

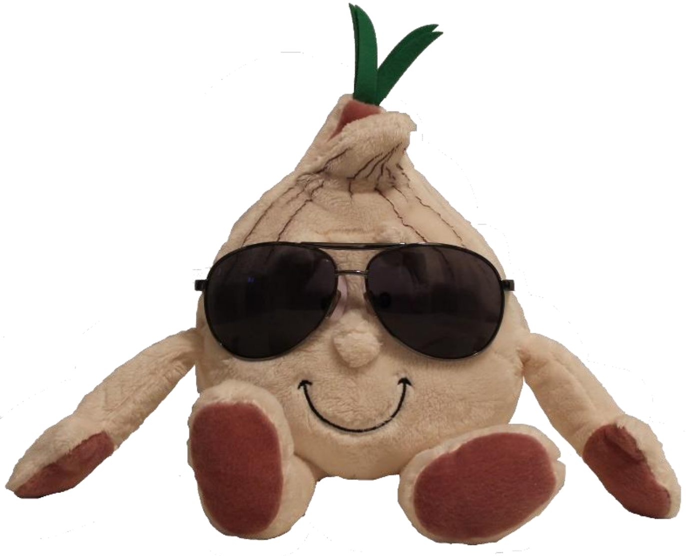
- **Our main thesis:**

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?



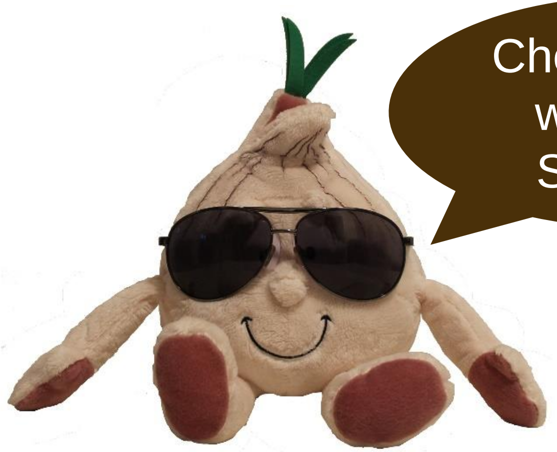


Our constructive approach: provide a set of (12) rules

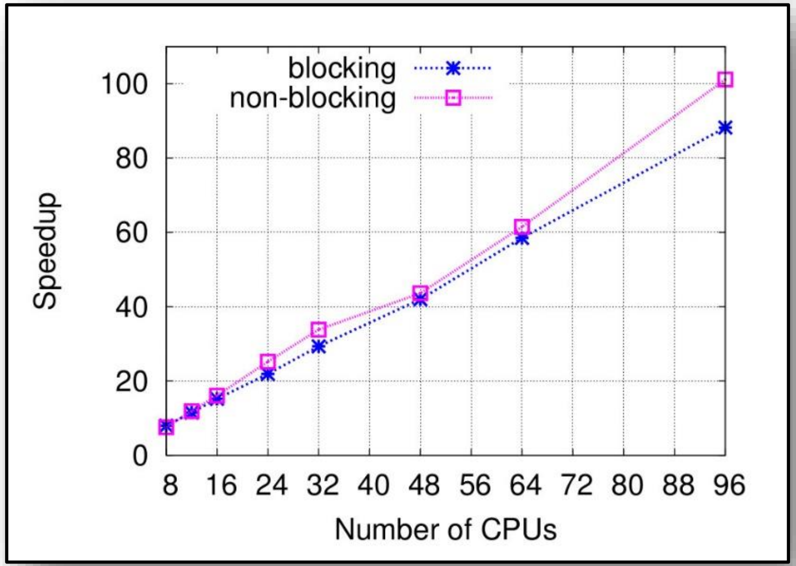
- **Attempt to emphasize interpretability of performance experiments**
- **The set is not complete**
  - And probably never will be
  - Intended to serve as a solid start
  - Call to the community to extend it

- **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



Check out my wonderful Speedup!



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

- **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.**

The simplest networking question: ping pong latency!

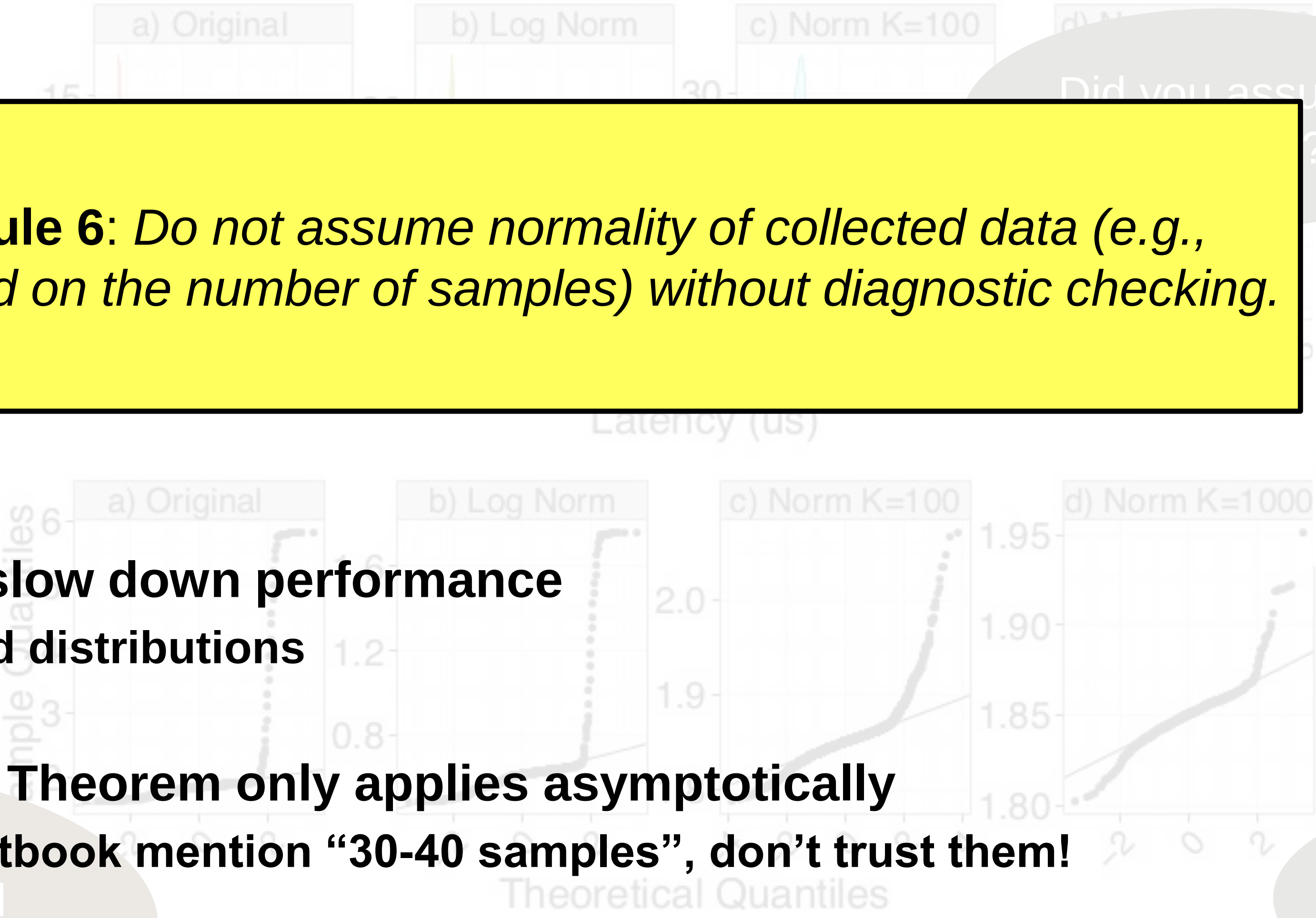
Rule 5: *Report if the measurement values are deterministic. For nondeterministic data, report confidence intervals of the measurement.*

- **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."

Thou shalt not trust your average textbook!

The confidence interval is 1.765us to 1.775us



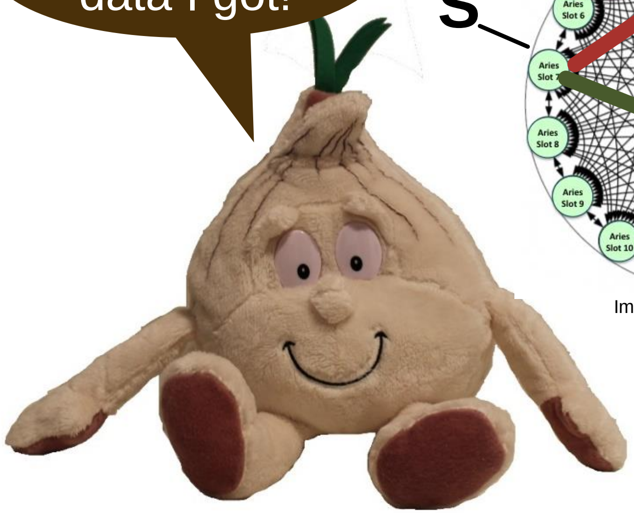
Rule 6: *Do not assume normality of collected data (e.g., based on the number of samples) without diagnostic checking.*

Can we test for normality?

- **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**

Thou shalt not trust your system!

Look what data I got!



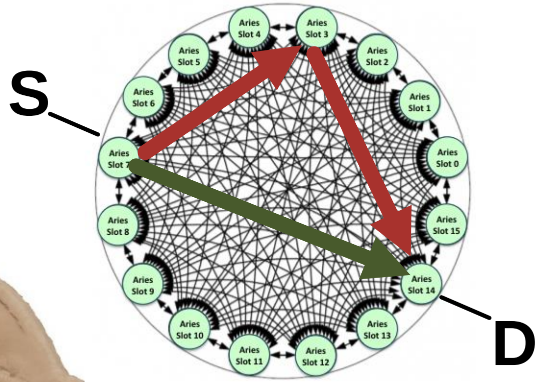
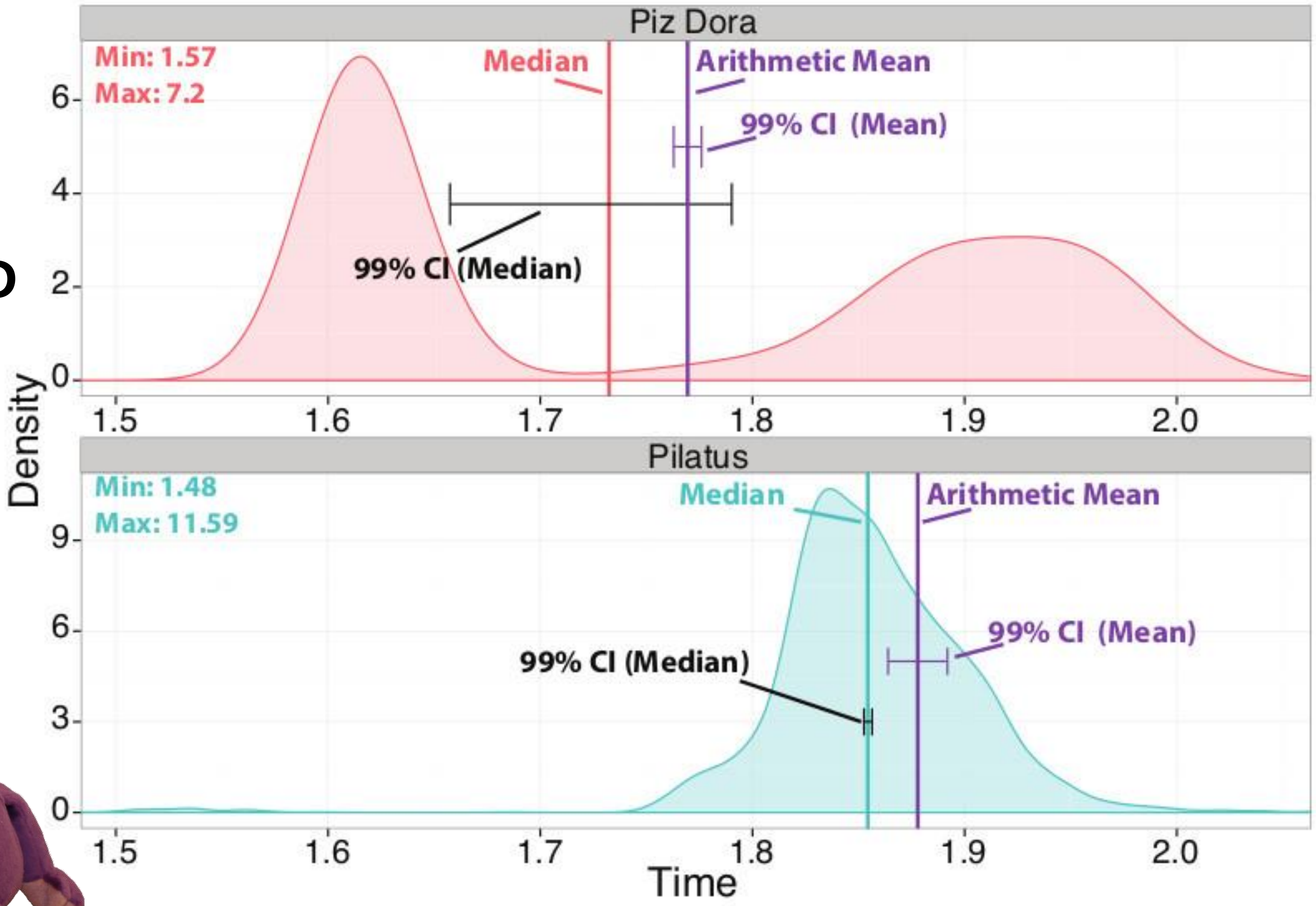
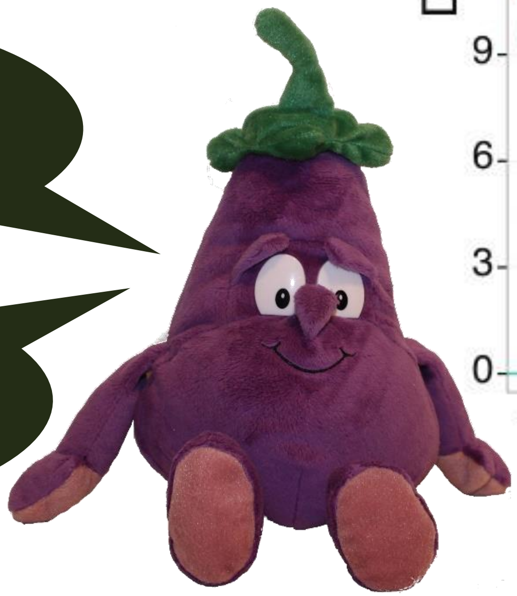


Image credit: nersc.gov



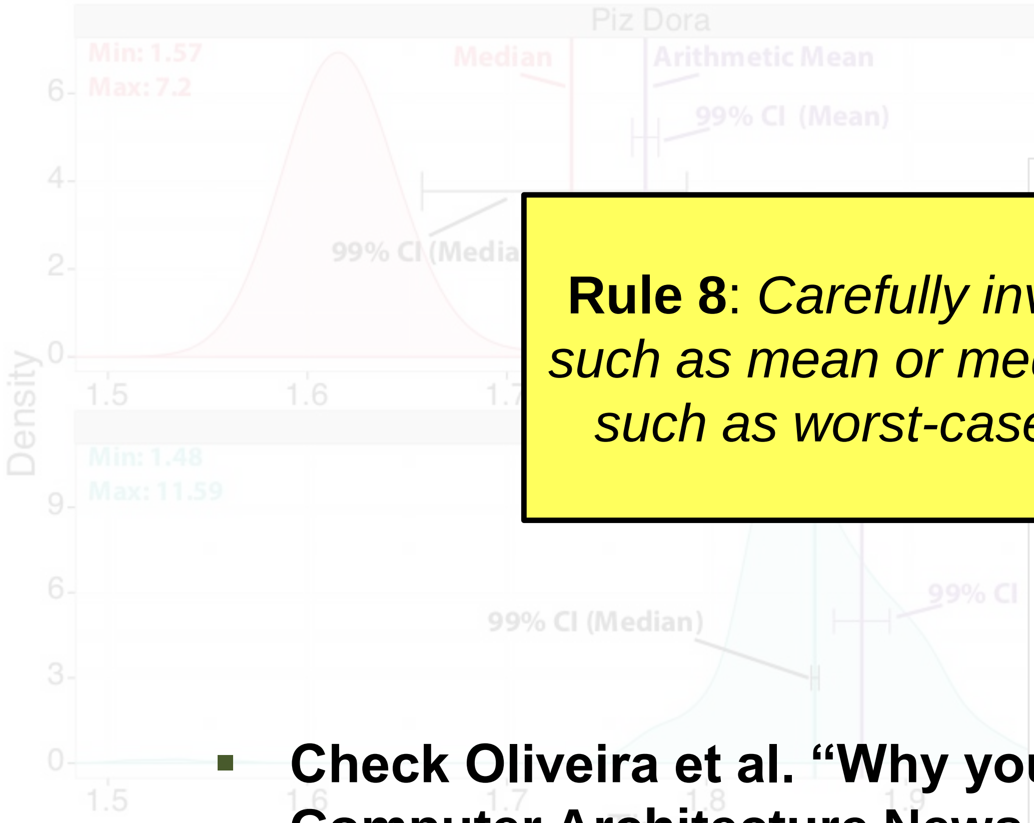
Clearly, the mean/median are not sufficient!

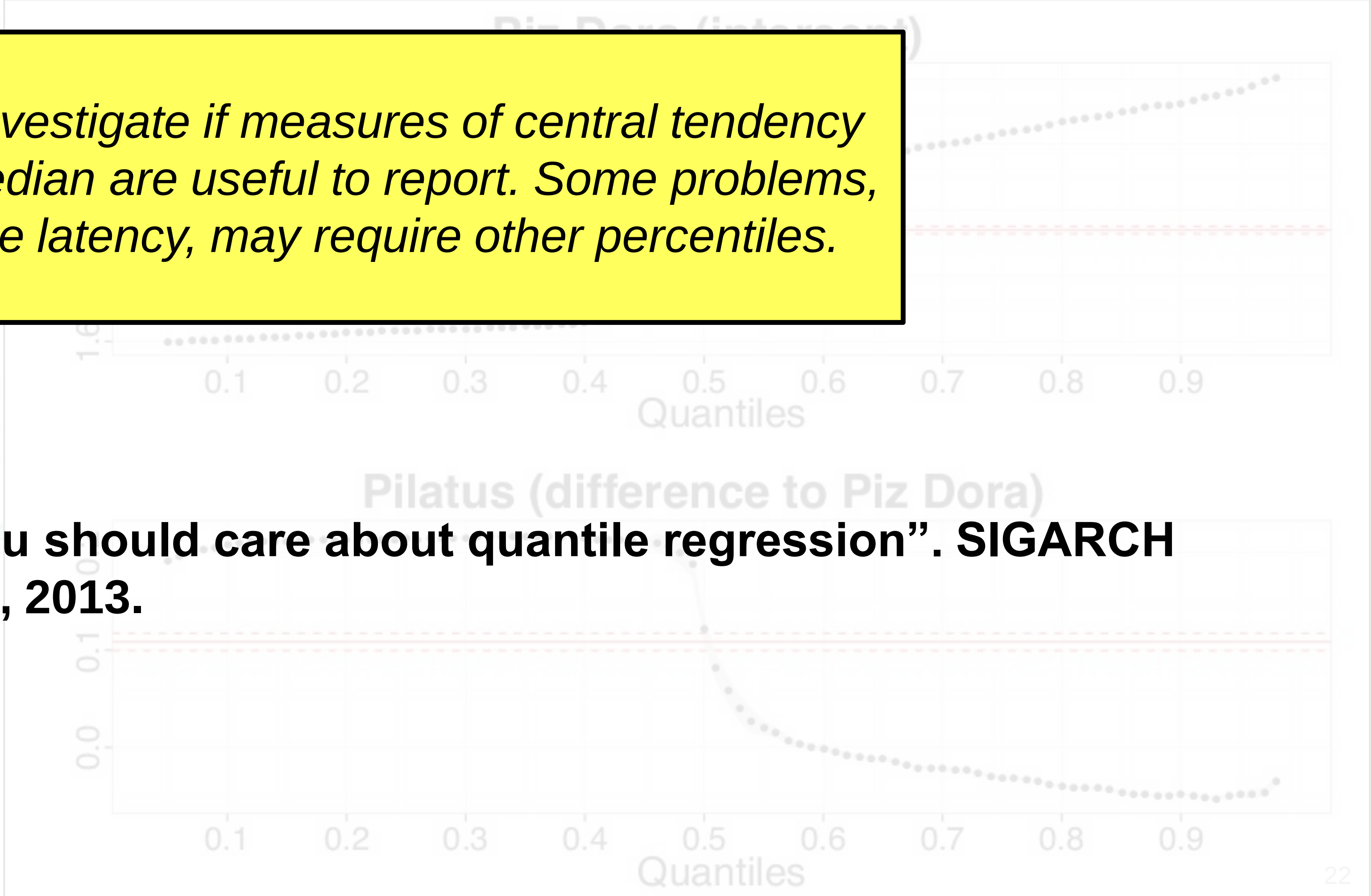
Try quantile regression!



Quantile Regression

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



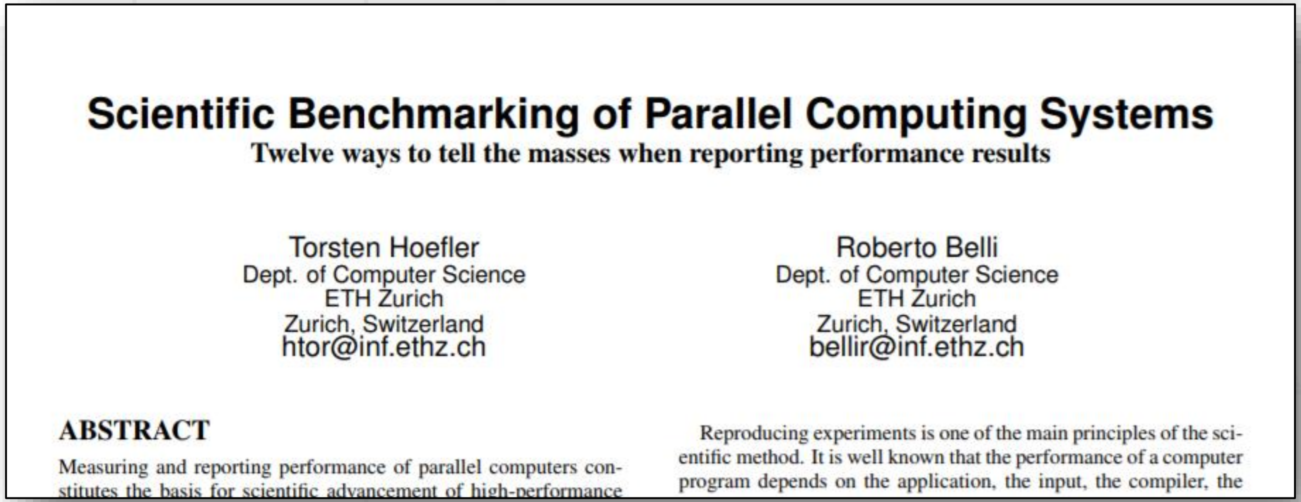


Rule 8: *Carefully investigate if measures of central tendency such as mean or median are useful to report. Some problems, such as worst-case latency, may require other percentiles.*

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

Wrapping up the 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

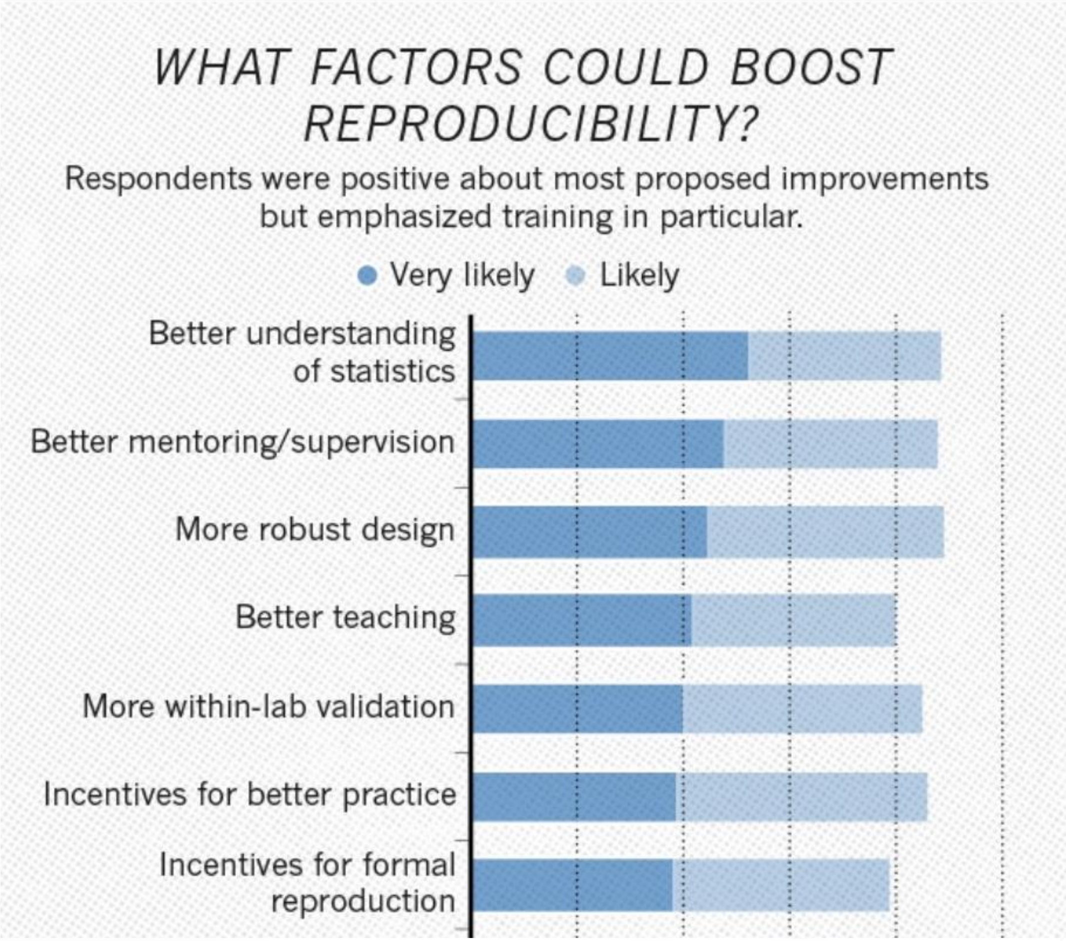
Roberto Belli
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
bellir@inf.ethz.ch

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

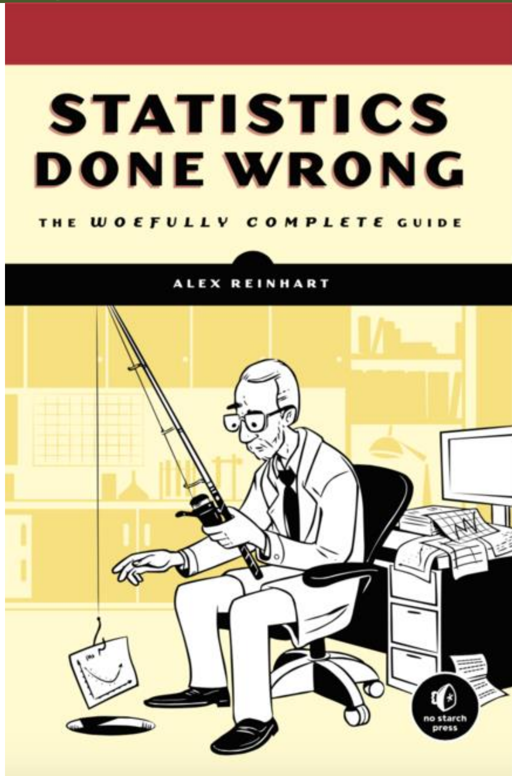


NOV
8
2018

Twelve ways to fool the masses when reporting performance of deep learning workloads

blog Uncategorized

"Statistical performance" vs. "hardware performance"



- **Tradeoffs between those two**
  - Very foreign for HPC people – we always operated in double precision
    Mostly out of fear of rounding issues

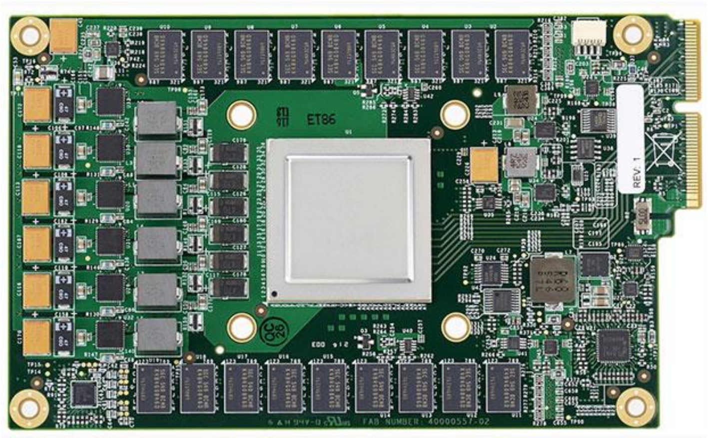
- **Deep learning shows how little accuracy one can get away with**
  - Well, examples are drawn randomly from some distribution we don't know …
  - Usually, noise is quite high …
  - So the computation doesn't need to be higher precision than that noise
    Pretty obvious! In fact, it's similar in scientific computing but in tighter bounds and not as well known

- **But we HPC folks like flop/s! Or maybe now just ops or even aiops? Whatever, fast compute!**
  - A humorous guide to **floptimization**
  - Twelve rules to help present your (not so great?) results in a much better light

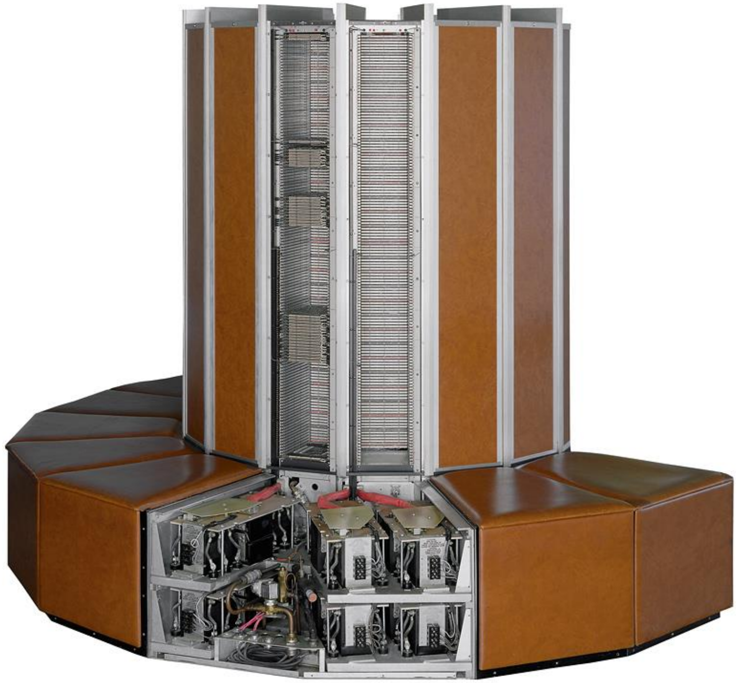
4) Compare outdated hardware with special-purpose hardware!

- **Tesla K20 in 2018!?**

  Even though the older machines would win the beauty contest!

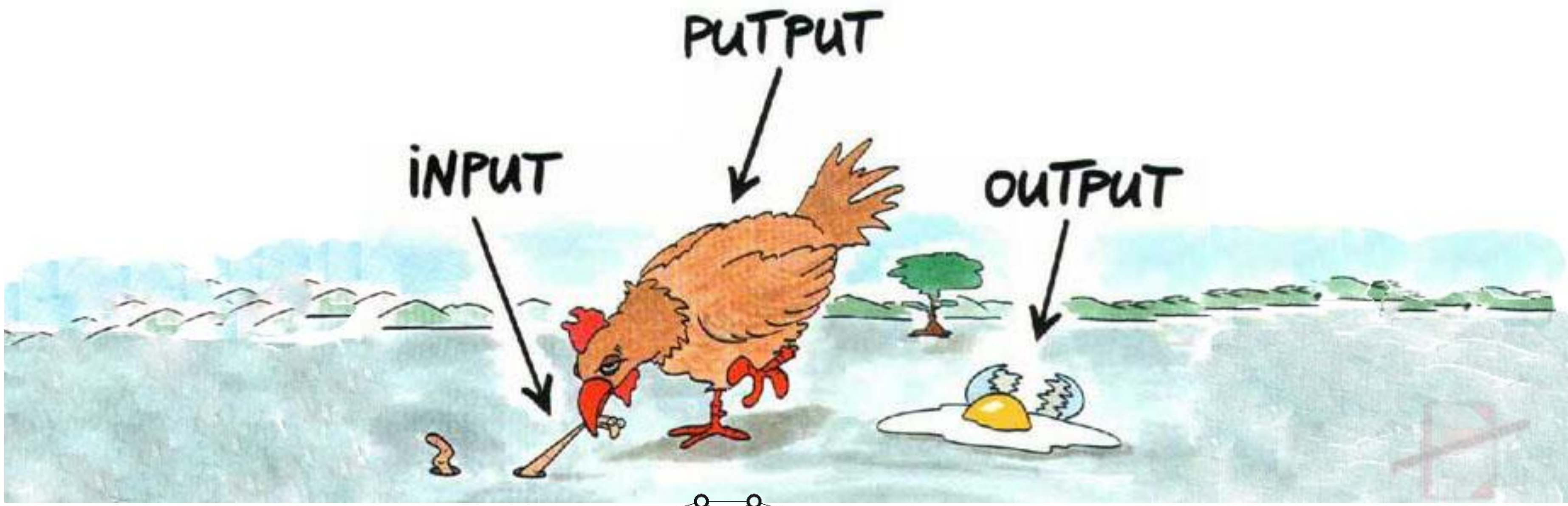


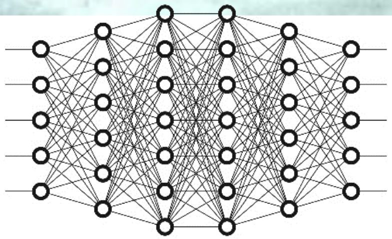
vs.



6) Do not consider I/O!

- **Reading the data? Nah, make sure it's staged in memory when the benchmark starts!**





9) Train on (unreasonably) large inputs!

- **The pinnacle of floptimization! Very hard to catch!**

  But Dr. Catlock Holmes below can catch it.

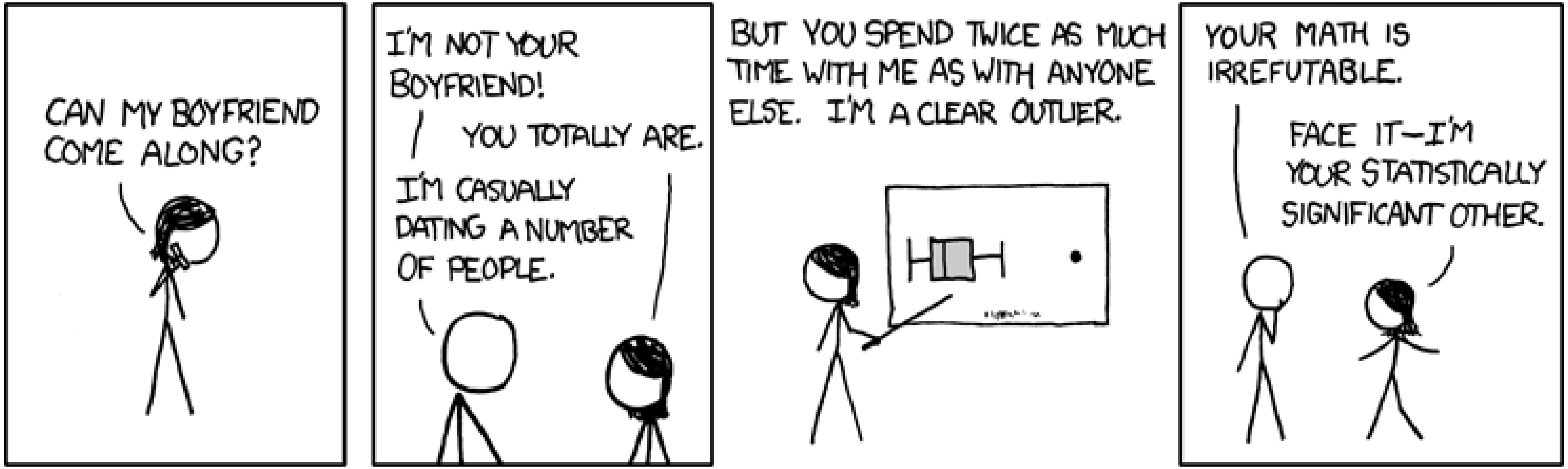
Low-resolution cat (244x244 – 1 Gflop/example)

VS.

High-resolution cat (8kx8x – 1 Tflop/example)

12) Select carefully how to compare to the state of the art!

- **Compare either time to solution or accuracy if both together don't look strong!**
  There used to be conventions but let's redefine them.



Reproducing and Benchmarking Deep Learning

- ## End result – generalization

Benchmark	Focus			Metrics									Criteria			Customizability			DL Workloads						Remarks
	Perf	**Con**	**Acc**	**Tim**	**Cos**	**Ene**	**Util**	**Mem**	**Tput**	**Brk**	**Sca**	**Com**	**TTA**	**FTA**	**Lat**	**Clo**	**Ope**	**Inf**	**Ops**	**Img**	**Obj**	**Spe**	**Txt**	**RL**	
DeepBench [39]	👍	👎	👎	👍	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👍	👍	👎	👍	👍	👎	👎	👎	👎	👎	Ops: Conv., GEMM, RNN, Allreduce
TBD [47]	👍	👎	👎	👎	👎	👎	👍	👍	👍	👎	👎	👎	👎	👎	👍	👍	👎	👎	👎	👍	👍	👍	👍	👍	+GANs
Fathom [2]	👍	👎	👎	👍	👎	👎	👎	👎	👍	👍	👍	👎	👎	👎	👍	👍	👎	👎	👎	👍	👎	👍	👍	👍	+Auto-encoders
DAWNBench [9]	👍	👍	👎	👍	👍	👎	👎	👎	👎	👎	👎	👎	👍	👎	👍	👎	👍	👎	👎	👍	👎	👎	👍	👎	
Kaggle [21]	👎	👎	👍	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👍	👎	👎	👍	👎	👎	👍	👍	👍	👍	👍	Varying workloads
ImageNet [13]	👎	👎	👍	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👍	👎	👎	👍	👎	👎	👍	👍	👎	👎	👎	
MLPerf [30]	👍	👍	👍	👍	👍	👍	👎	👎	👎	👎	👎	👎	👍	👍	👍	👍	👍	👎	👎	👍	👍	👍	👍	👍	
Deep500	👍	👍	👍	👍	👍	(👍)	👍	(👍)	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	(👍)	(👍)	(👍)	(👍)	

TABLE II: **An overview of available DL benchmarks**, focusing on the offered functionalities. **Perf**: Performance, **Con**: Convergence, **Acc**: Accuracy, **Tim**: Time, **Cos**: Cost, **Ene**: Energy, **Util**: Utilization, **Mem**: Memory Footprint, **Tput**: Throughput (Samples per Second), **Brk**: Timing Breakdown, **Sca**: Strong Scaling, **Com**: Communication and Load Balancing, **TTA**: Time to Accuracy, **FTA**: Final Test Accuracy, **Lat**: Latency (Inference), **Clo**: Closed (Fixed) Model Contests, **Ope**: Open Model Contests, **Inf**: Fixed Infrastructure for Benchmarking, **Ops**: Operator Benchmarks, **Img**: Image Processing, **Obj**: Object Detection and Localization, **Spe**: Speech Recognition, **Txt**: Text Processing and Machine Translation, **RL**: Reinforcement Learning Problems, 👍: A given benchmark does offer the feature. (👍): Planned benchmark feature. 👎: A given benchmark does not offer the feature.

- ## Sample throughput

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

Existing Deep Learning Frameworks

System	Operators		Networks				Training			Dist. Training			
	Sta	Cus	Def	Eag	Com	Tra	Dat	Opt	Cus	PS	Dec	Asy	Cus
(L) cuDNN	👍	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎
(L) MKL-DNN	👍	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎
(F) TensorFlow [1]	👍	👍	👍	👍	(👍)	(👍)	👍	UR	👍	👍	(👍)	👍	👎
(F) Caffe, Caffe2† [21]	👍	(👍)	👍	👎	👎	👎	👎	UR	👎	👎	👍	👎	👎
(F) [Py]Torch† [10, 35]	👍	👍	👎	👍	👎	👎	👎	👍	👍	👎	👍	👎	👎
(F) MXNet [6]	👍	(👍)	👍	👎	👎	👎	👍	UR	(👍)	👍	👎	👍	👎
(F) CNTK [48]	👍	(👍)	👍	👎	👎	👎	👍	UR	(👍)	👎	👍	👎	👎
(F) Theano [4]	👍	👍	👍	👍	👍	👍	👎	(👍)	👎	👎	👎	👎	👎
(F) Chainer[MN] [44]	👍	👍	👎	👍	👎	👎	👍	(👍)	👍	👎	👍	👎	👎
(F) Darknet [38]	👍	👎	👍	👎	👎	👎	👎	(👍)	👎	👎	👎	👎	👎
(F) DL4j [43]	👍	👍	👍	👎	👎	👎	👍	UR	(👍)	👍	👎	👎	👎
(F) DSSTNE	👍	👎	👍	👎	👎	👎	👎	UR	👎	👎	👎	👎	👎
(F) PaddlePaddle	👍	👍	👍	👎	👎	(👍)	(👍)	UR	(👍)	👍	👎	👍	(👍)
(F) TVM [7]	👍	👍	👍	👎	👍	👍	👎	👎	👎	👎	👎	👎	👎
(E) Keras [8]	👍	👎	👎	👎	👎	👎	👎	UR	(👍)	👎	👎	👎	👎
(E) Horovod [42]	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👍	👎	👎
(E) TensorLayer [14]	👍	👎	👎	👎	👎	👎	👎	UR	👎	👎	(👍)	👎	👎
(E) Lasagne	(👍)	👍	👎	👎	👎	👎	👎	UR	(👍)	👎	👎	👎	👎
(E) TFLearn [11]	(👍)	👎	👎	👎	👎	👎	👍	👎	👎	👎	👎	👎	👎

- Customizing operators relies on framework
- Network representation
- Dataset representation
- Training algorithm
- Distributed training (e.g., asynchronous SGD)

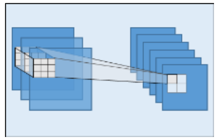
T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

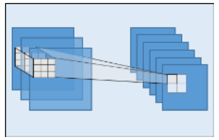
Deep500

- **Deep learning meta-framework: a framework for frameworks to reside in**

Level 0

```
forward()
gradient()
```





CustomOp

Operators

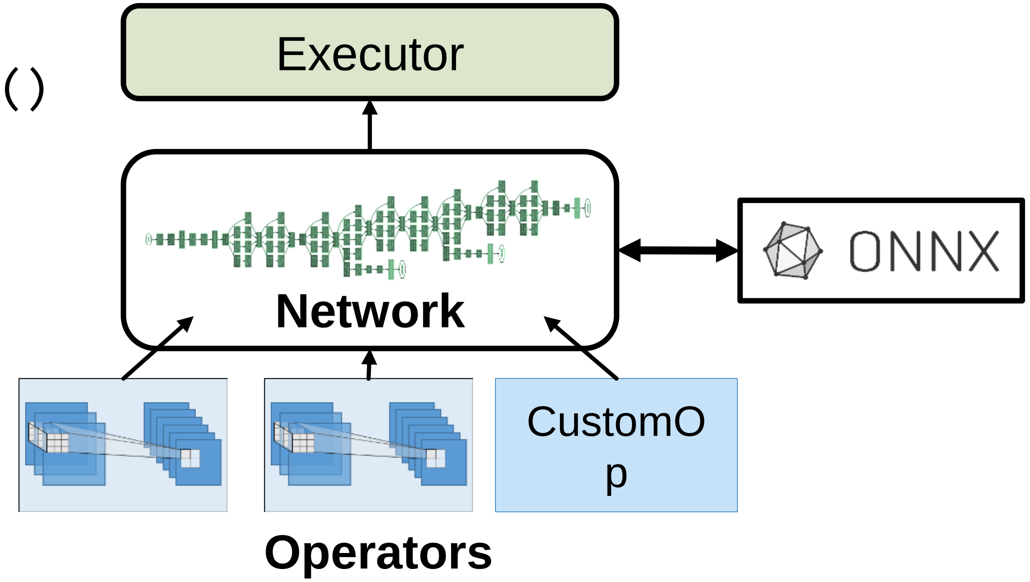
Deep500

- Deep learning **meta-framework**: a framework for frameworks to reside in

Level 0

Level 1

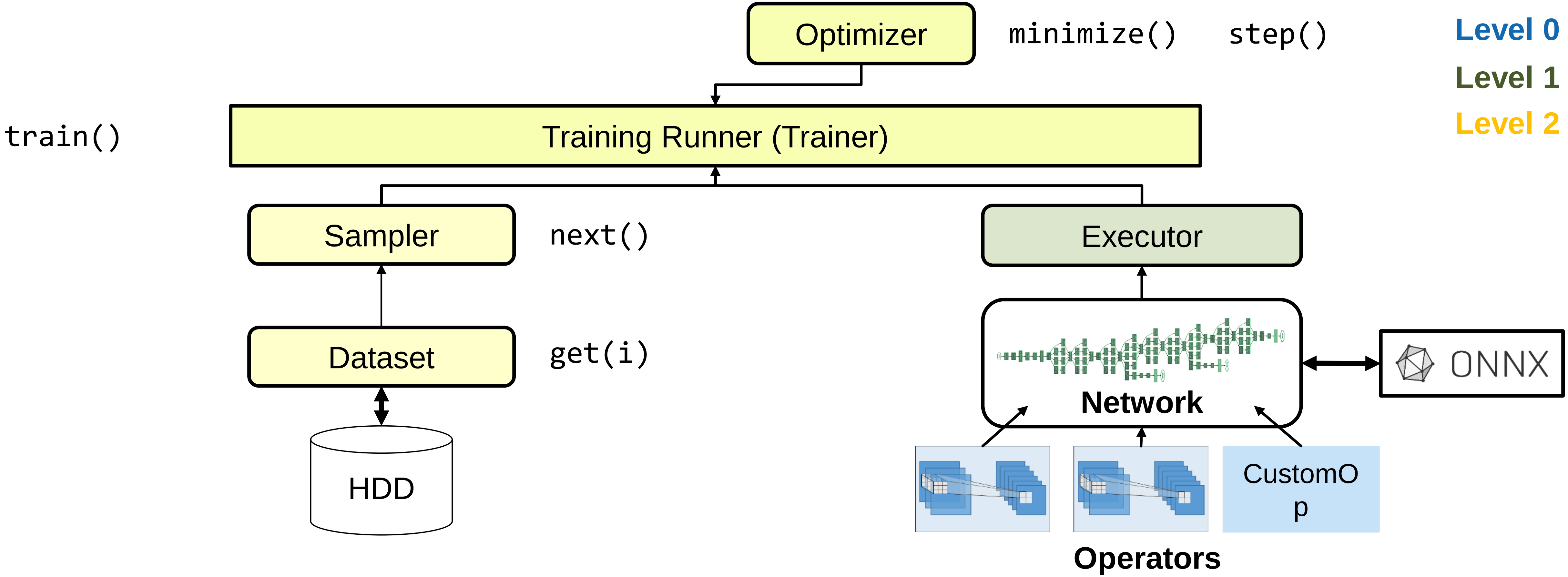
```
inference()
inference_and_backprop()

      add_node()
      add_edge()
      remove_...
```



T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

Deep500

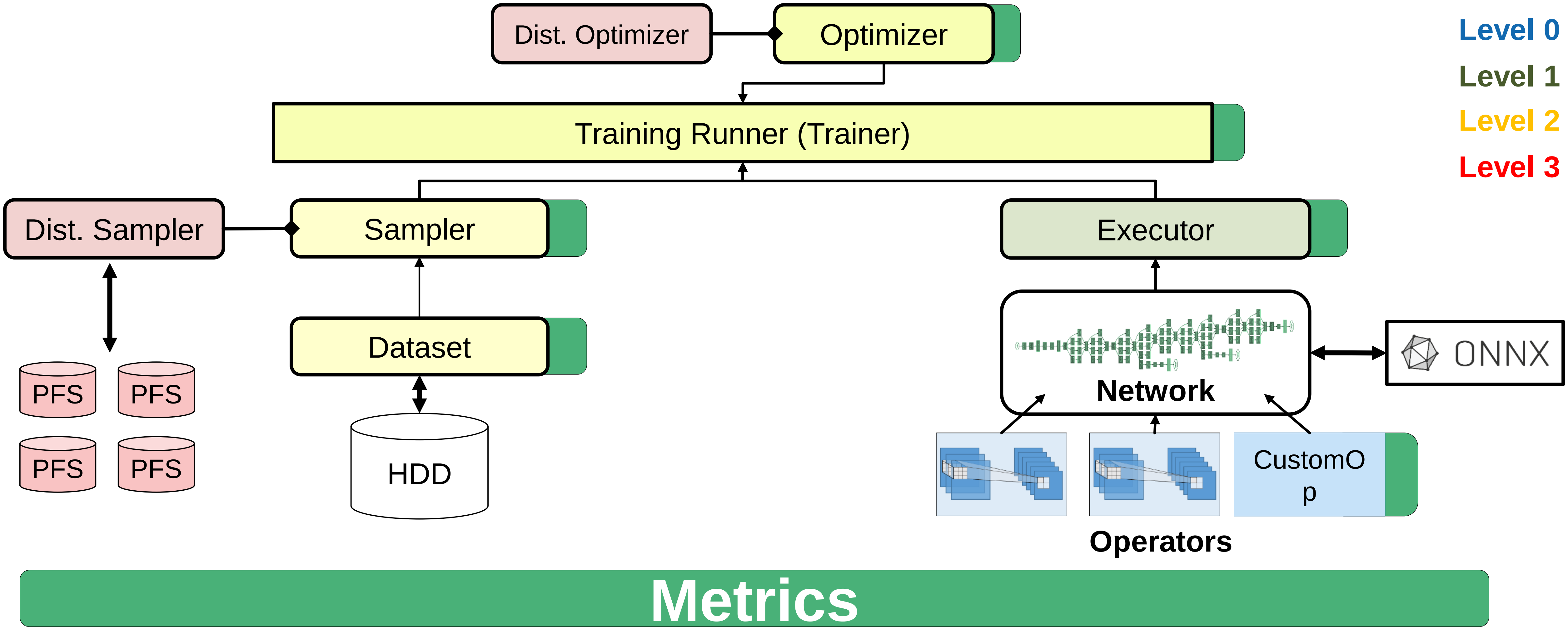
- **Deep learning meta-framework: a framework for frameworks to reside in**



T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

Deep500

- **Deep learning meta-framework: a framework for frameworks to reside in**



T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

For Benchmarking: Recipes

Fixed definitions + mutable definitions +
acceptable metric set = Recipe

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

For Benchmarking: Recipes

Fixed definitions + mutable definitions + acceptable metric set = Recipe

```python
""" A recipe for running the CIFAR-10 dataset with ResNet-44 and a momentum
    optimizer, with metrics for final test accuracy. """

import deep500 as d5
from recipes.recipe import run_recipe

# Using PyTorch as the framework
import deep500.frameworks.pytorch as d5fw


# Fixed Components
FIXED = {
    'model': 'resnet',
    'model_kwargs': dict(depth=44),
    'dataset': 'cifar10',
    'train_sampler': d5.ShuffleSampler,
    'epochs': 1
}

# Mutable Components
MUTABLE = {
    'batch_size': 64,
    'executor': d5fw.from_model,
    'executor_kwargs': dict(device=d5.GPUDevice()),
    'optimizer': d5fw.MomentumOptimizer,
    'optimizer_args': (0.1, 0.9),
}

# Acceptable Metrics
METRICS = [
    (d5.TestAccuracy(), 93.0)
]


if __name__ == '__main__':
    run_recipe(FIXED, MUTABLE, METRICS) or exit(1)
```

https://github.com/deep500/deep500/blob/master/recipes/cifar10_resnet44.py

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

For Customizing: New Operator

```python
class IPowOp(CustomPythonOp):
    def __init__(self, power):
        super(IPowOp, self).__init__()
        self.power = power
        assert int(power) == power # integral

    def forward(self, inputs):
        return inputs[0] ** self.power

    def backward(self, grads, fwd_inputs, fwd_outputs):
        return (grads[0] * self.power *
            (fwd_inputs[0] ** (self.power - 1)))
```

Python

```cpp
template<typename T>
class ipowop : public deep500::CustomOperator {
protected:
    int m_len;
public:
    ipowop(int len) : m_len(len) {}
    virtual ~ipowop() {}

    void forward(const T *input, T *output) {
        #pragma omp parallel for
        for (int i = 0; i < m_len; ++i)
            output[i] = std::pow(input[i], DPOWER);
    }

    void backward(const T *nextop_grad,
                  const T *fwd_input_tensor,
                  const T *fwd_output_tensor,
                  T *input_tensor_grad) {
        #pragma omp parallel for
        for (int i = 0; i < m_len; ++i) {
            input_tensor_grad[i] = nextop_grad[i] * DPOWER *
                std::pow(fwd_input_tensor[i], DPOWER - 1);
        }
    }
};
```

C++

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

For Customizing: Distributed Optimization

```python
class ConsistentNeighbors(DistributedOptimizer):
    # Follows communication scheme from https://arxiv.org/pdf/1705.09056.pdf

    def step(self, inputs):
        self.base_optimizer.new_input()
        for param in self.network.get_params():
            self.base_optimizer.prepare_param(param)
        output = self.executor.inference_and_backprop(inputs, self.base_optimizer.loss)
        gradients = self.network.gradient(self.base_optimizer.loss)
        for param_name, grad_name in gradients:
            param, grad = self.network.fetch_tensors([param_name, grad_name])
            grad = self.communication.reduce_from_neighbors(grad) / 3
            param = self.base_optimizer.update_rule(grad, param, param_name)
            self.network.feed_tensor(param_name, param)
        return output
```

https://github.com/deep500/deep500/blob/master/deep500/frameworks/reference/distributed_optimizers.py#L68

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

Conclusions and call for action

DEEP500

- **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
- **See the 12 rules and 12 ways as a start**
  - Much is implemented in LibSciBench [1]
- **Deep500 [2] aims to enable reproducibility in deep learning – across frameworks**
  - Call to action to community to:
    Define more recipies (datasets, networks, tasks)
    Improve implementations/techniques
    Implement reprodicibly
    New (aggregate) metrics?

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/
[2]: https://www.deep500.org/