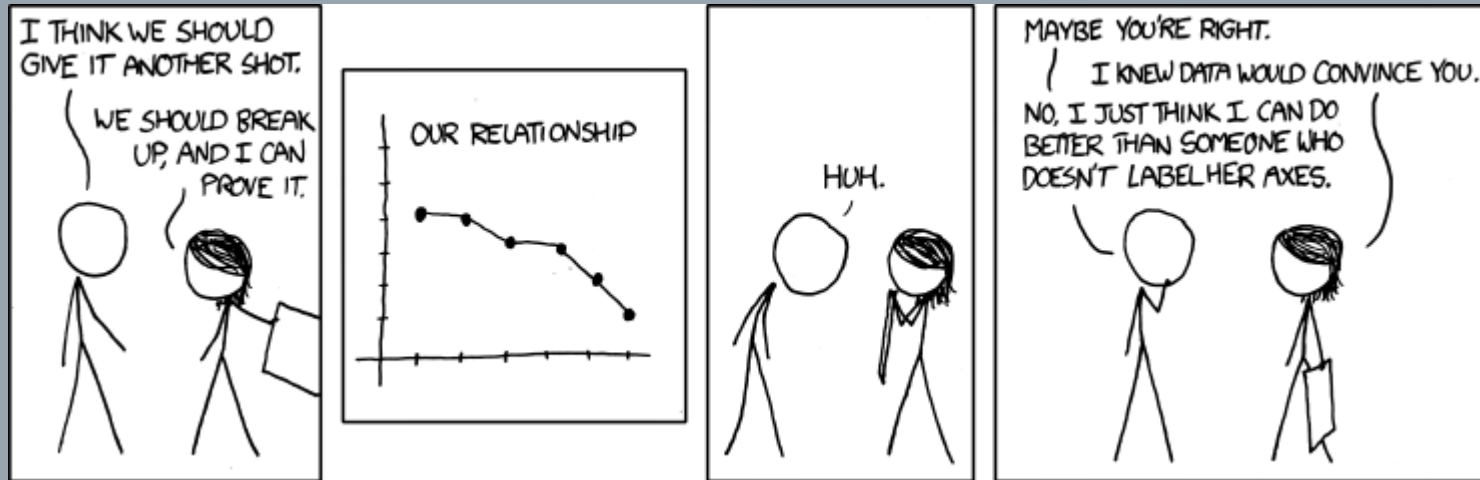


# Designing experiments and presenting data



<https://xkcd.com/833/>

# Overview

---

- Preliminaries
  - Questions to ask
  - Compile, run, document
- What and how to measure
  - Metrics
  - Digits
  - Variation
- How to present the data
  - Avoiding blatant mistakes
  - Exposing relevant effects

---

Data taking is just the beginning.

Science emerges from  
communication.

# Preliminaries



# Questions to ask

## 1. What is the **scientific question**?

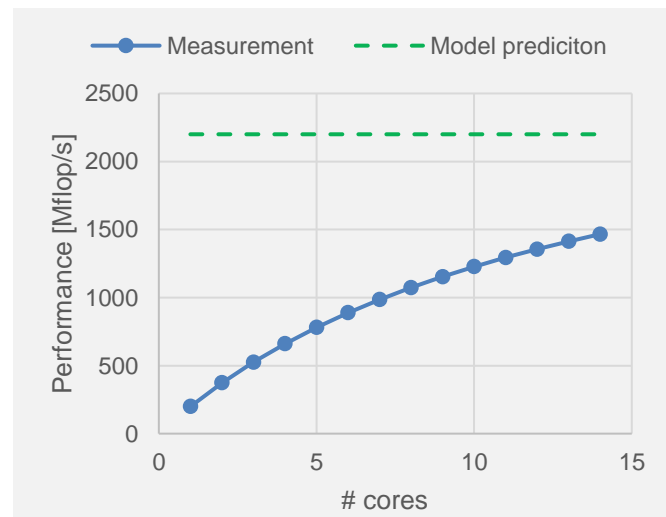
Example: “Does this parallel loop achieve the predicted performance limit on my CPU?”

## 2. What **measurements** do I need to discuss/answer it?

Example: Loop runtime for # of cores from 1 ... max

## 3. How do I **present the data** so my conclusions are **communicated** clearly?

Example: x-y plot of performance (work/time) vs. # cores, including comparison with expectation



# The most important question

---

4. Would the audience/readers, given the resources, be able to interpret and reproduce what I did?

Example: Specify

- System environment
- Compilers & compiler options
- Library versions
- Execution modalities (affinity, rank mapping, # of repetitions, frequency settings,...)

# Some useful general hints

- Load modules with full version (`intel/2022.1.0`) and document it
- Compile code with optimization flags suitable for system
  - Intel: `-O3 -xHost` is usually a good start
  - Document the flags
- Always run with pinning and controlled CPU frequency
  - Best use base frequency (`likwid-powermeter -i`)
  - `srun --cpu-freq=X-Y:performance` or `likwid-setFrequencies`
  - Document the settings
- Capture raw output in a logfile for later analysis
  - `$ ./run.sh 2>&1 | tee $(hostname -s).log`
- Perform manual runs AND check output before thinking about scripting

# What and how to measure





# What and how to measure

- What to measure?
  - ... depends on what you want to investigate
  - In HPC, this is often **runtime** (and other stuff based on it)

```
#include <time.h>


double timestamp(void) {
    struct timespec ts;
    clock_gettime(CLOCK_MONOTONIC, &ts);
    return (double)ts.tv_sec + (double)ts.tv_nsec * 1.e-9;
}

double resolution(void) {
    struct timespec ts;
    clock_getres(CLOCK_MONOTONIC, &ts);
    return (double)ts.tv_sec + (double)ts.tv_nsec * 1.e-9;
}
```

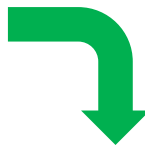
# Finite resolution/granularity

```
const int N=100;

auto s = timestamp();
for(int i=0; i<N; ++i)
    sum += a[i];
auto runtime = timestamp() - s;
auto perf = N/runtime;
```



- Runtime = raw metric
- Performance = derived metric
- Warmup iterations might be necessary
- Coarse granularity may also apply to other metrics (HW events)



```
int NITER = 1;
for(;;) {
    auto s = timestamp();
    for(int k=0; k<NITER; ++k) {
        for(int i=0; i<N; ++i)
            sum += a[i];
        if(sum < 0.) dummy(&s,a);
    }
    runtime = timestamp() - s;
    if(runtime >= 0.1) break;
    NITER *= 2;
}
auto perf = double(N)*NITER/runtime;
```

# Raw and derived metrics

## Raw metrics/events

- wall-clock time [s]
- clock cycles [cy]
- Instructions [ins]
- floating-point instructions
- (mispredicted) branches
- cache line transfers
- cache/TLB misses
- energy consumption [Joule]
- ...

## Derived metrics

- clock frequency [cy/s]
- IPC [ins/cy] or CPI [cy/ins]
- FP performance [flop/cy,flop/s]
- branch misprediction ratio
- data transfer bandwidth [byte/cy,byte/s]
- cache miss ratio
- power dissipation [W]
- ...

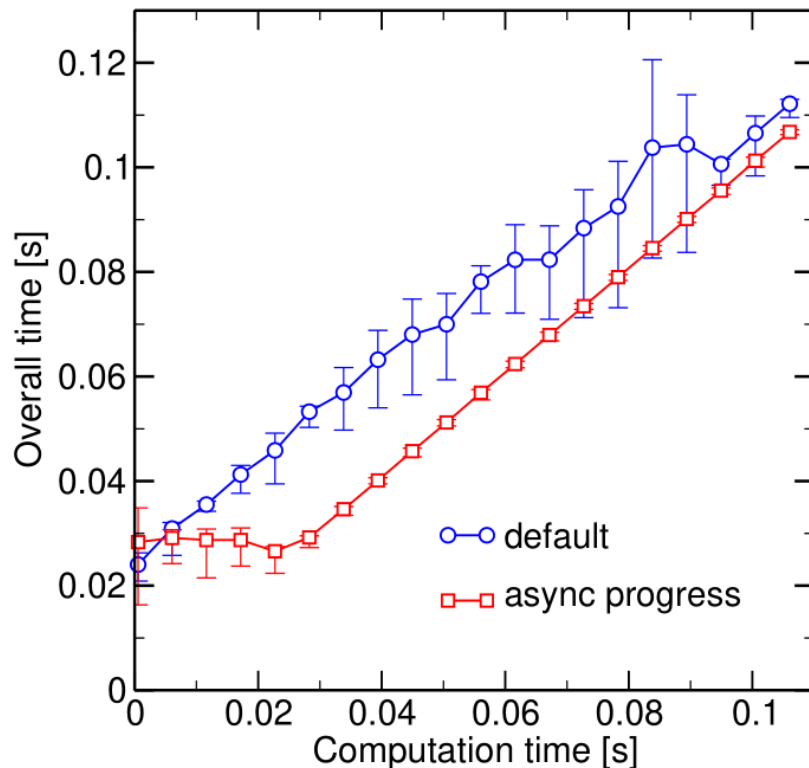
# Number of digits

- Most measurements are not accurate to many digits
  - 3 digits mostly suffice

# cores	Time (plain) [s]	Time (optimized) [s]	Speedup [%]
1	<del>12.0435766</del>	8.6198441	<del>39.71919283</del>
2	6.1179025	5.5901951	9.439874469
3	4.9041394	4.6966098	4.418710705
4	4.7002801	4.6912042	0.193466317

# Statistical variation (I)

- Although computers are supposed to be deterministic machines, **measurements often fluctuate**
- Communicate if fluctuations are **relevant** or not
  - If they are, **include statistical data** in your presentation!



# Statistical variation (II)

Many metrics in HPC are usually not normally distributed

→ Median may be more appropriate than average

→ multi-modal distributions can be represented by violin plots

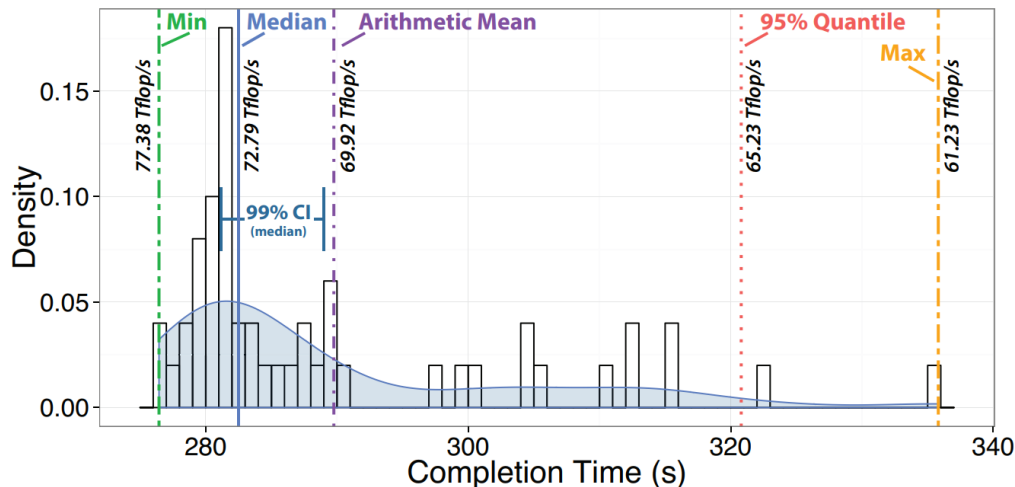
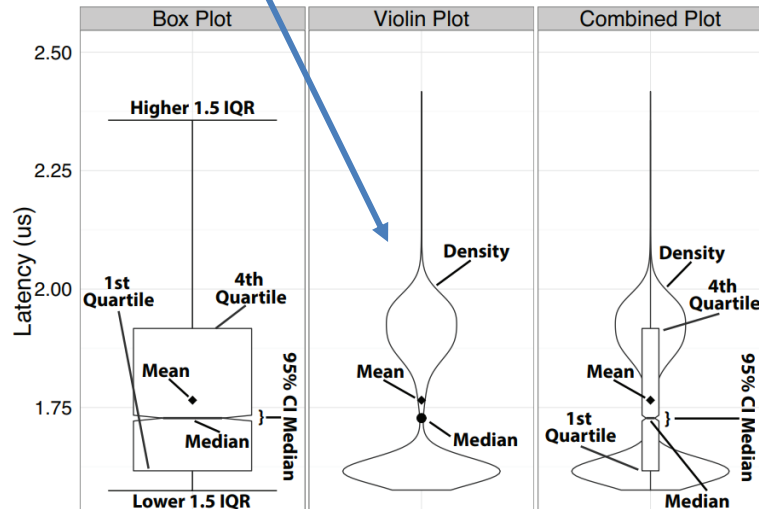


Figure 1: Distribution of completion times for 50 HPL runs.



Hoefler & Belli, DOI: [10.1145/2807591.2807644](https://doi.org/10.1145/2807591.2807644)

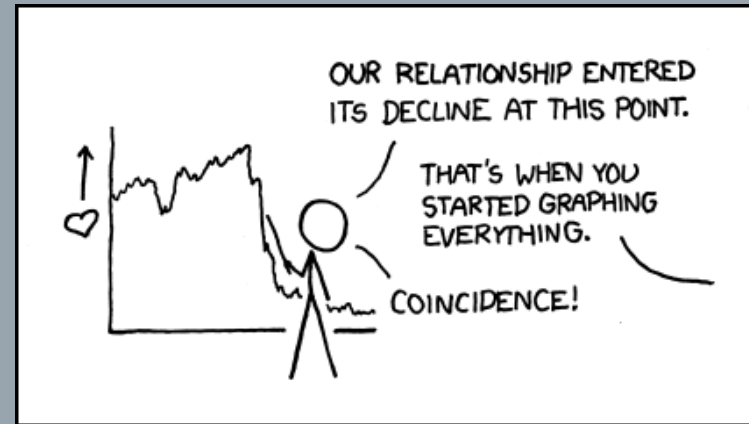
# Presenting data with graphs

See also:

Justin Zobel: *Writing for Computer Science* (Ch. 11)

<https://doi.org/10.1007/978-1-4471-6639-9>

<https://xkcd.com/523/>



# General guidelines

---

- X-Y plots showing measured data ...
  - have meaningful **labeling at the axes including units**,
  - provide a **meaningful caption** that allows to interpret what is seen,
  - have a title describing the **environment (system, settings)** or provide this info in the caption,
  - have a **legend** if more than one data set is shown,
  - have their **y-axis start at 0** unless there is a very good reason not to,
  - use **log scale with care**,
  - may use **straight lines** to connect data points to “**guide the eye**”



# Graphs: the good, the bad, and the ugly



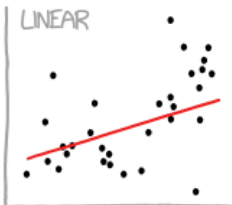
Figure 1: Scaling on Meggie

WTF??

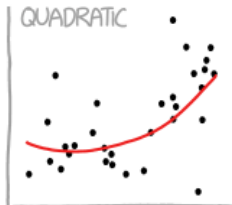
I'll have your head on a stick for such a presentation!

# Curve fitting

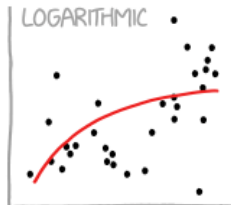
## CURVE-FITTING METHODS AND THE MESSAGES THEY SEND



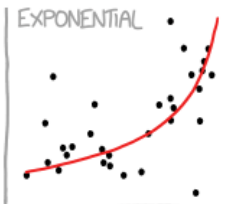
"HEY, I DID A REGRESSION."



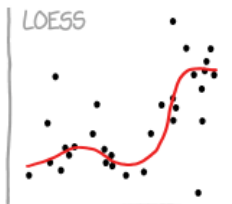
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."



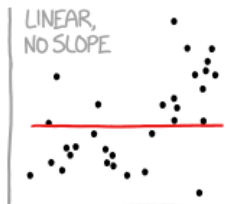
"LOOK, IT'S TAPERING OFF!"



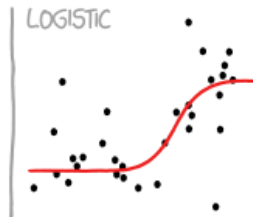
"LOOK, IT'S GROWING UNCONTROLLABLY!"



"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."



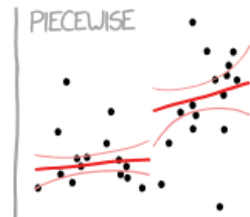
"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."



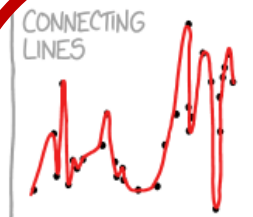
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."



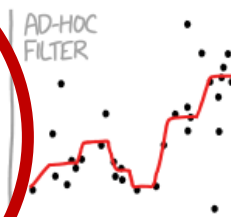
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."



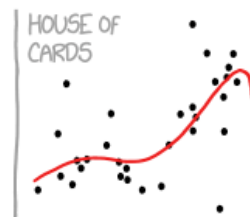
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."



"I CLICKED 'SMOOTH LINES' IN EXCEL."



"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"

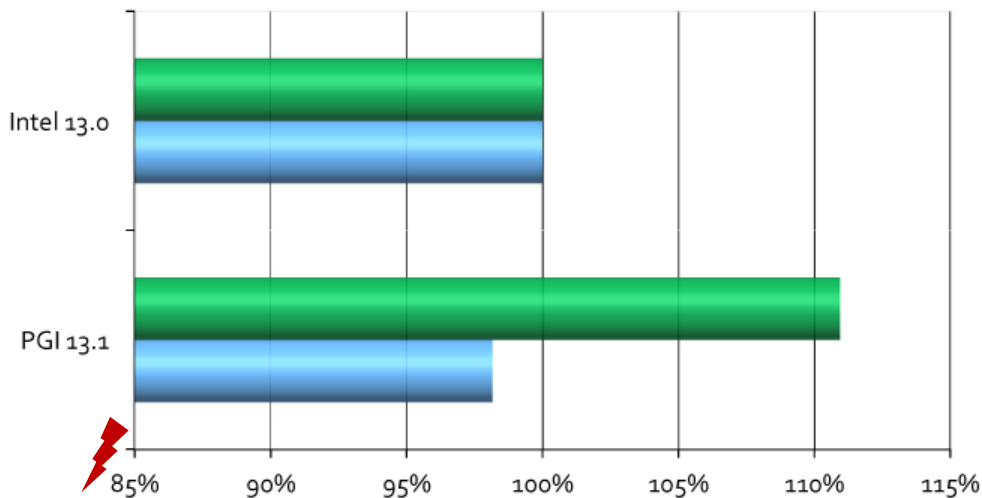


"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE- WAIT NO NO DON'T EXTEND IT AAAAAA!!"

<https://xkcd.com/2048>

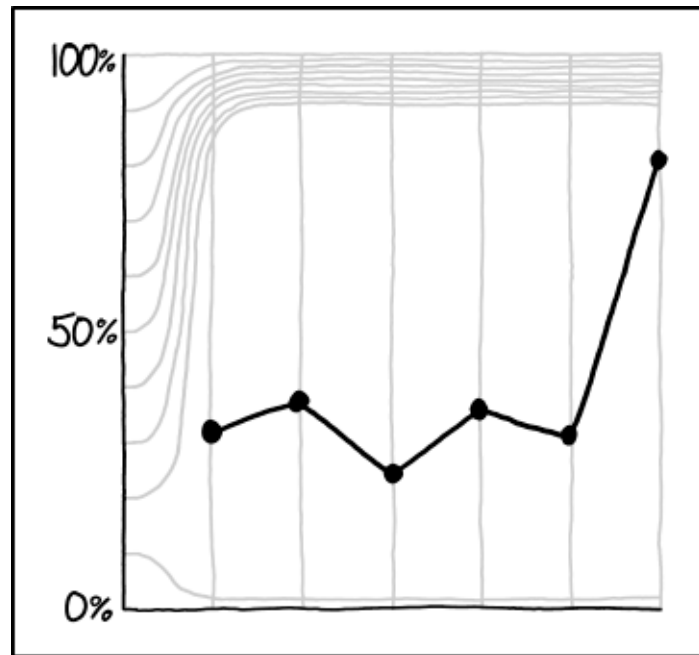
## SPEC OMP2012 Performance

- AMD Piledriver 2p/32 cores
- Intel Sandy Bridge 2p/16 cores without hyperthreading



SPECCompG\_base2012 relative performance as measured by The Portland Group during the weeks of January 28 and February 4, 2013. The number of OpenMP threads was set to match the number of cores on each system. SPECComp® is a registered trademark of the Standard Performance Evaluation Corporation (SPEC).

[http://www.pgroup.com/images/charts/spec\\_omp2012\\_chart\\_big.png](http://www.pgroup.com/images/charts/spec_omp2012_chart_big.png)



PEOPLE HAVE WISED UP TO THE "CAREFULLY CHOSEN Y-AXIS RANGE" TRICK, SO WE MISLEADING GRAPH MAKERS HAVE HAD TO GET CREATIVE.

<https://xkcd.com/2023/>

# Graphs: the good, the bad, and the ugly

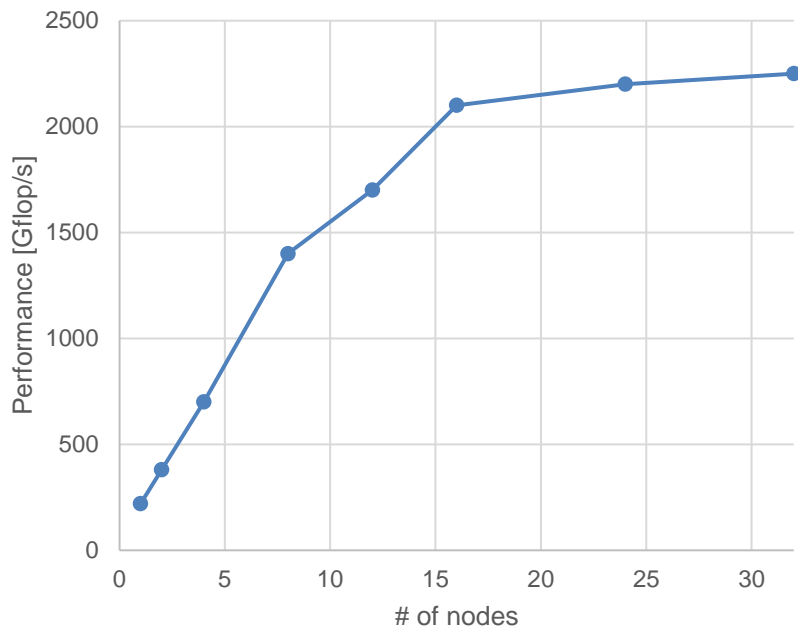


Figure 1: Performance scaling of the hybrid “CoolCFD” code on Meggie with problem set A

Provide the **necessary information** to allow the reader to **connect** the presented data to the **rest of your presentation/paper**

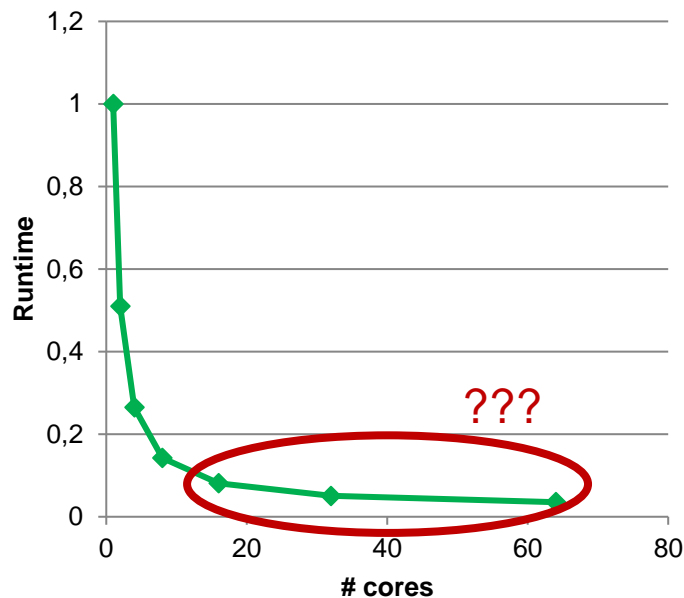
# Exposing relevant effects

Showing what you want to show

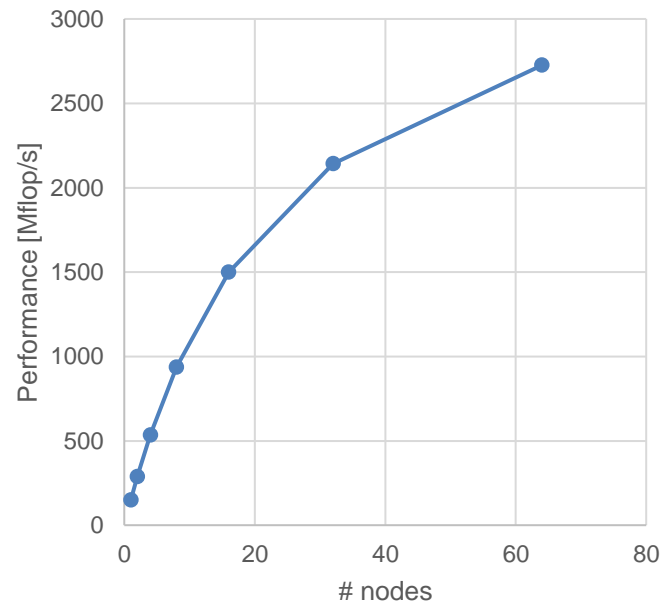
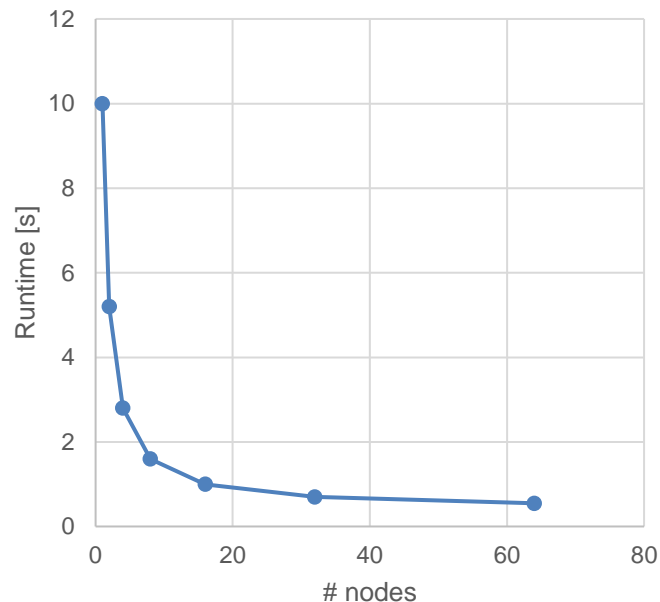


# Runtime or performance scaling?

- Ultimately, the user wants to know “How long will my problem take to solve?”
- Plotting runtime vs. resources answers this question
- However...
  - **Scaling behavior** hard to visualize
  - **Hard to generalize** to different problem size



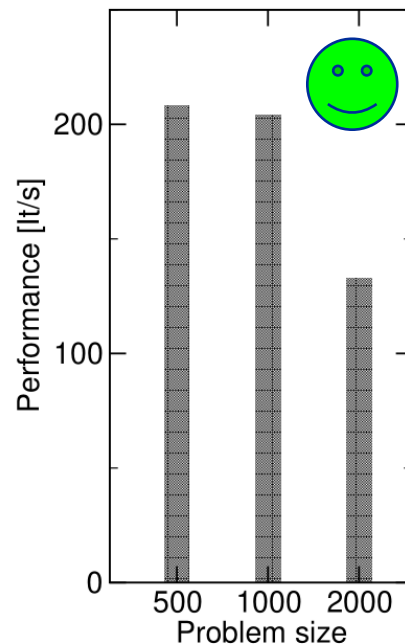
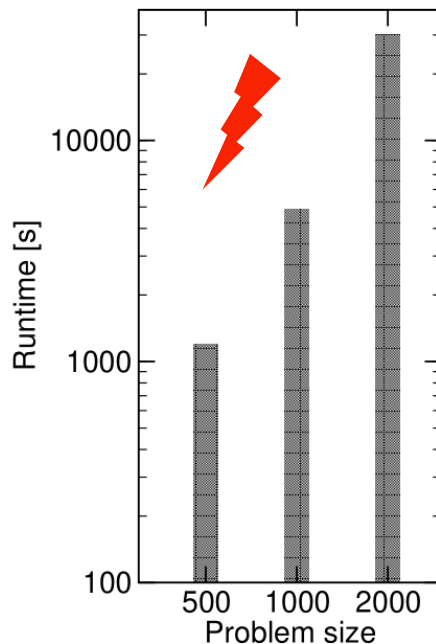
# Runtime or performance?



Performance is often the superior metric because it is normalized to a defined unit of work and scaling behavior is easier to read on a linear graph

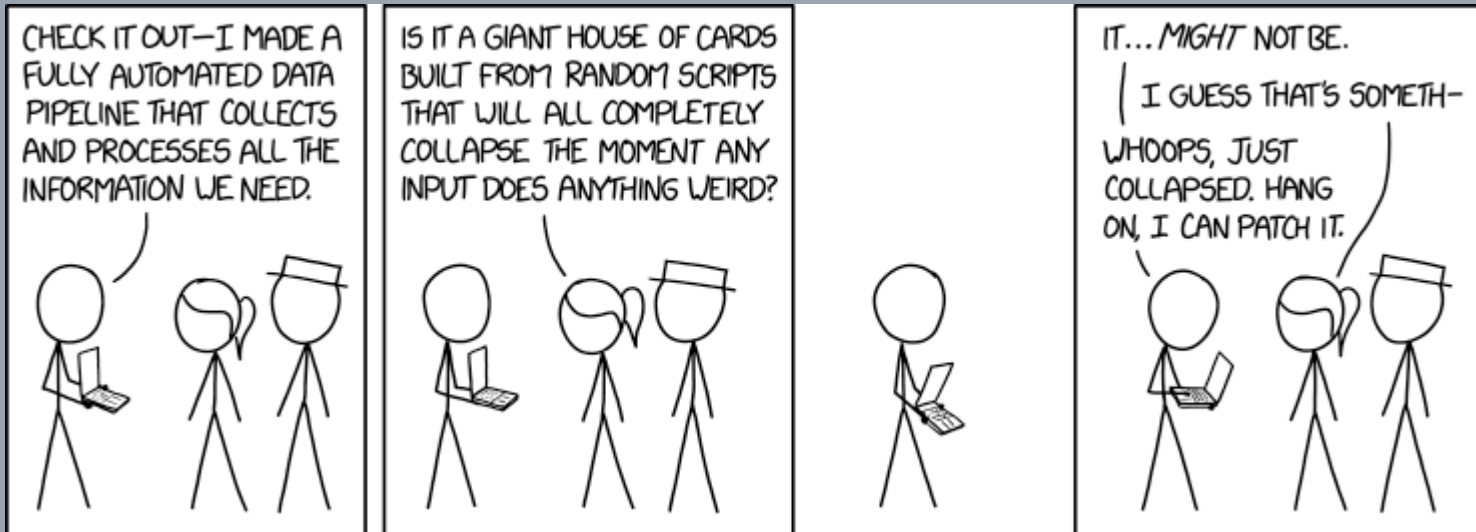
# Exposing the relevant effects

- Present data in a way that **exposes the interesting correlations** and ignores “trivial” dependencies
- Example: runtime or performance vs. problem size?
  - **Runtime** has a trivial dependence of “**larger problem takes longer**”
  - **Performance** vs. problem size shows clearly **a fundamental change** with larger problems
- This is **highly problem specific!**





# Thank you.



<https://xkcd.com/2054/>