# Obtaining Faithful/Reproducible Measurements on Modern CPUs

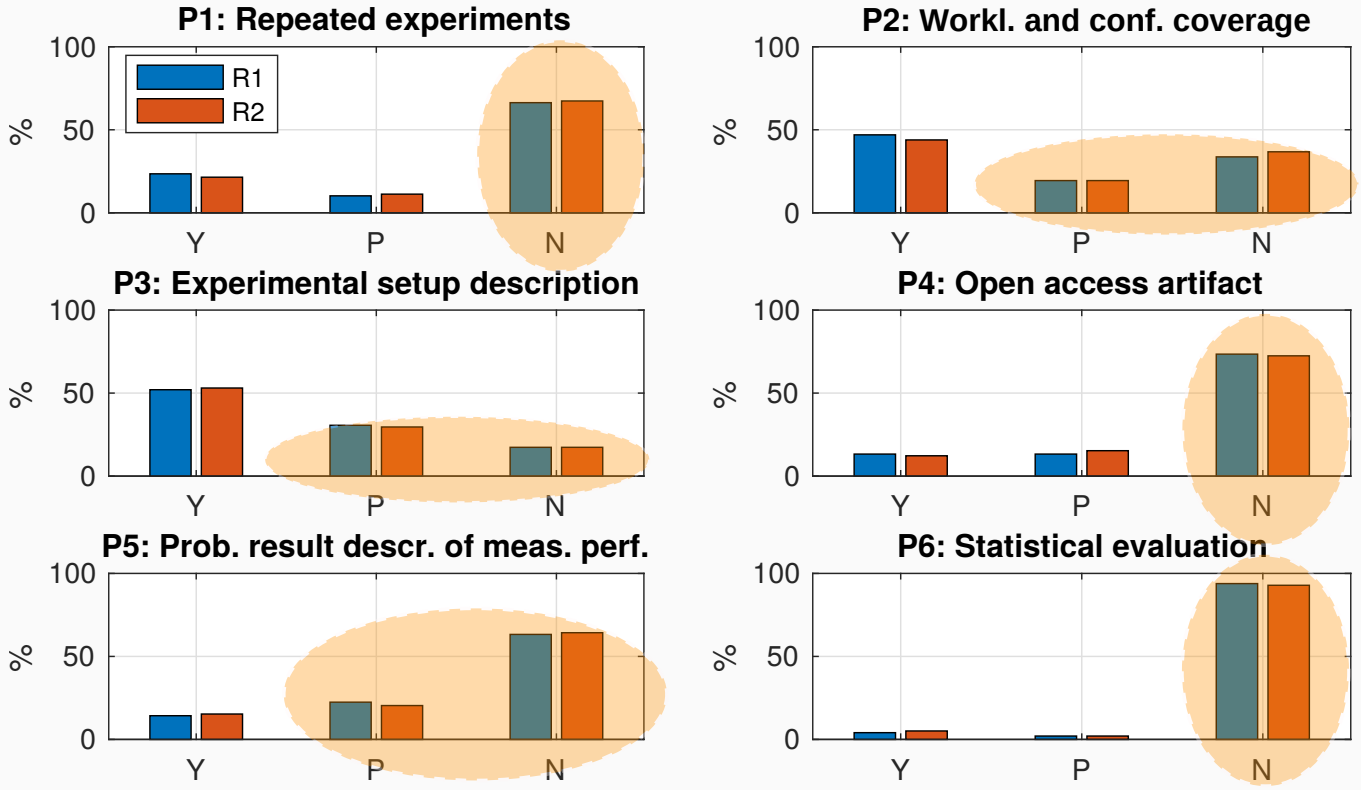
Tom Cornebize and Arnaud Legrand
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP


Nantes, Journées du GDR RSD, January 2020

# Toward Reproducible Computer Science Experiments ?

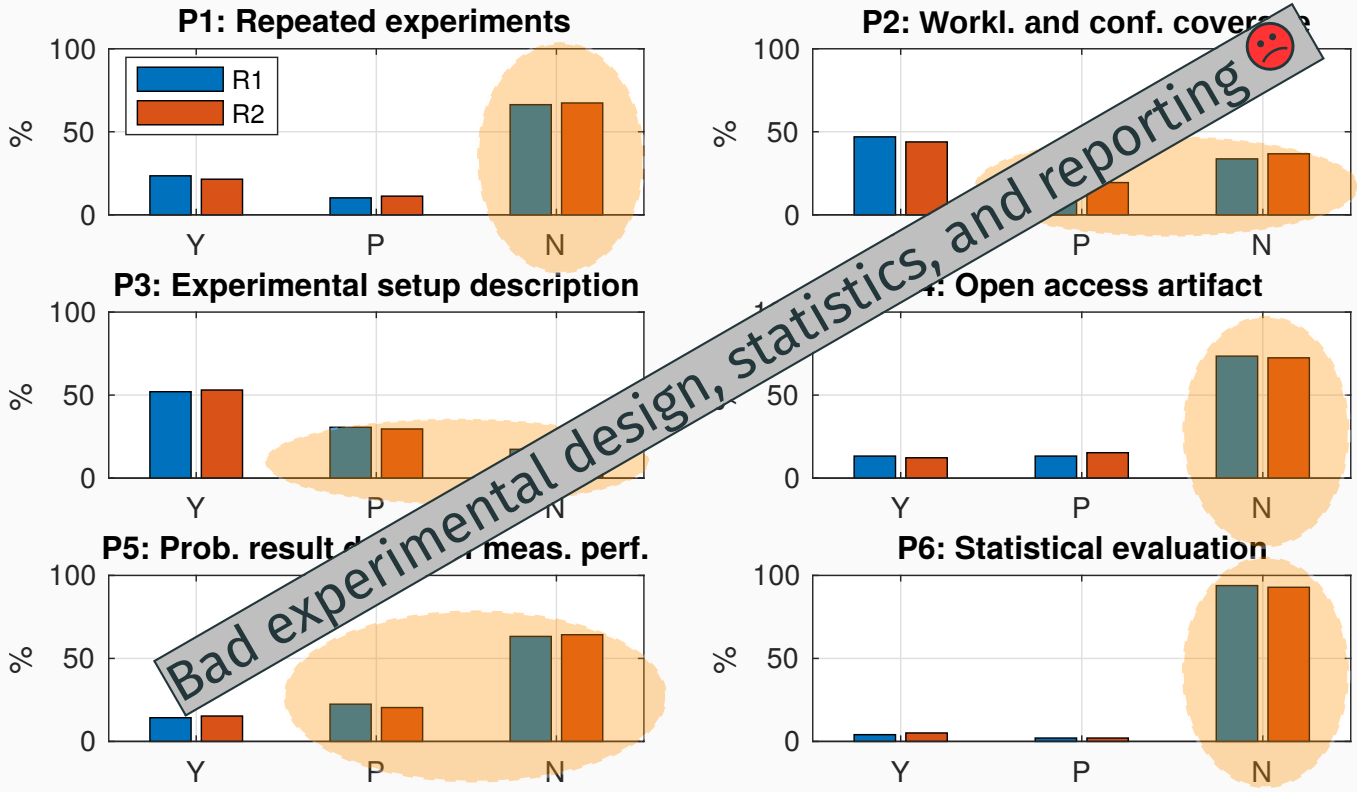
# Key Concerns for our Community (Room for Improvement)

*How are cloud performance currently obtained and reported?*, *Methodological Principles for Reproducible Performance Evaluation in Cloud Computing*, IEEE Trans. on Soft. Eng., July 2019



# Key Concerns for our Community (Room for Improvement)

*How are cloud performance currently obtained and reported?*, *Methodological Principles for Reproducible Performance Evaluation in Cloud Computing*, IEEE Trans. on Soft. Eng., July 2019




Key DoE principles:

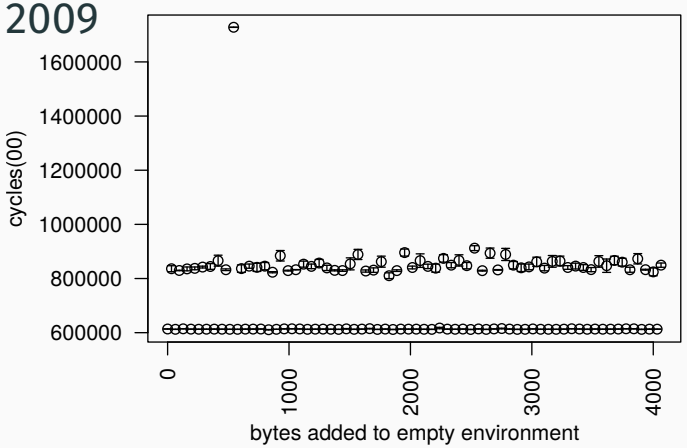
1. Replicate to increase reliability.
2. Randomize to reduce bias ⇝ Evaluate statistical confidence.

# Measuring performance is difficult

*Producing wrong data without doing anything obviously wrong!*

Mytkowicz et al. in ACM SIGPLAN Not. 44(3), March 2009

*changing the size of environment variables can trigger performance degradation as high as 300%; simply changing the link order of object files can cause performance to decrease by as much as 57%.*

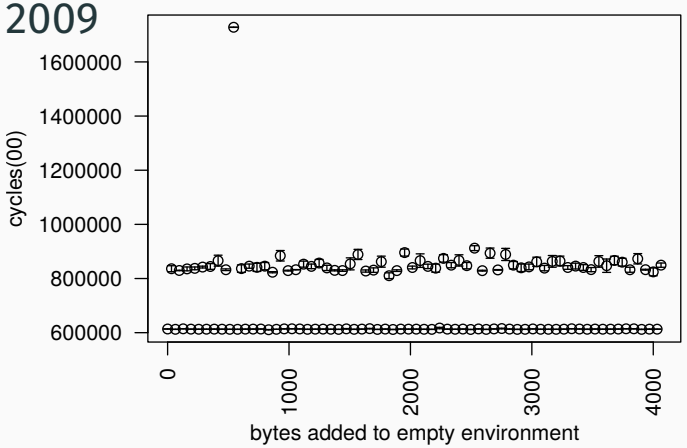


# Measuring performance is difficult

*Producing wrong data without doing anything obviously wrong!*

Mytkowicz et al. in ACM SIGPLAN Not. 44(3), March 2009

*changing the size of environment variables can trigger performance degradation as high as 300%; simply changing the link order of object files can cause performance to decrease by as much as 57%.*




**Taming the Influence of Memory Layout.** *Stabilizer: Statistically Sound Performance Evaluation*, C. Curtsinger and E. Berger in ASPLOS 2013

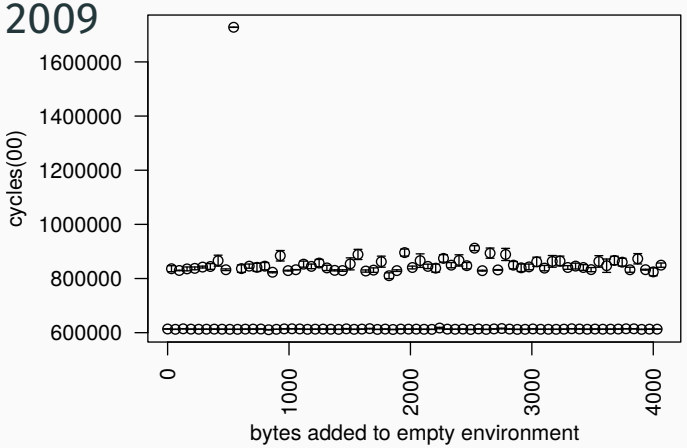
> *Stabilizer forces executions to sample the space of memory configurations by repeatedly rerandomizing layouts of code, stack, and heap objects at run-time. [..] Re-randomization ensures that layout effects follow a Gaussian distribution, enabling the use of statistical tests like ANOVA.*

# Measuring performance is difficult

*Producing wrong data without doing anything obviously wrong!*

**Mytkowicz et al. in ACM SIGPLAN Not. 44(3), March 2009**

*changing the size of environment variables can trigger performance degradation as high as 300%; simply changing the link order of object files can cause performance to decrease by as much as 57%.*




**Taming the Influence of Memory Layout.** *Stabilizer: Statistically Sound Performance Evaluation*, **C. Curtsinger and E. Berger in ASPLOS 2013**

> *Stabilizer forces executions to sample the space of memory configurations by repeatedly rerandomizing layouts of code, stack, and heap objects at run-time. [..] Re-randomization ensures that layout effects follow a Gaussian distribution, enabling the use of statistical tests like ANOVA.*

Randomization helps fighting bias incured by:

- specific configurations $AA\ldots A \rightarrow A_1A_2\ldots A_n$ (~~pseudo-replication~~)
- temporary perturbations $AA\ldots A\,BB\ldots B \rightarrow ABBAAAB\ldots$

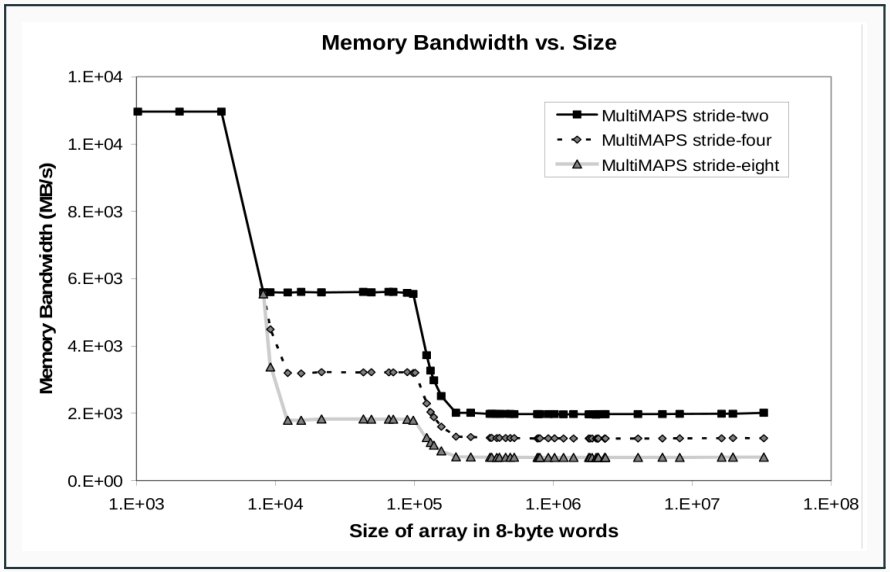
# Cache effects!

# Impact of Working Set Size on Effective Bandwdith

- Cache hierarchy (L1, L2, L3, RAM) with different bandwidth
- LRU, pre-fetching for linear access
- An array fits in a cache level $\Rightarrow$ operate at the corresponding bandwidth
- Stride access decrease bandwidth

# Impact of Working Set Size on Effective Bandwdith

- Cache hierarchy (L1, L2, L3, RAM) with different bandwidth
- LRU, pre-fetching for linear access
- An array fits in a cache level $\Rightarrow$ operate at the corresponding bandwidth
- Stride access decrease bandwidth



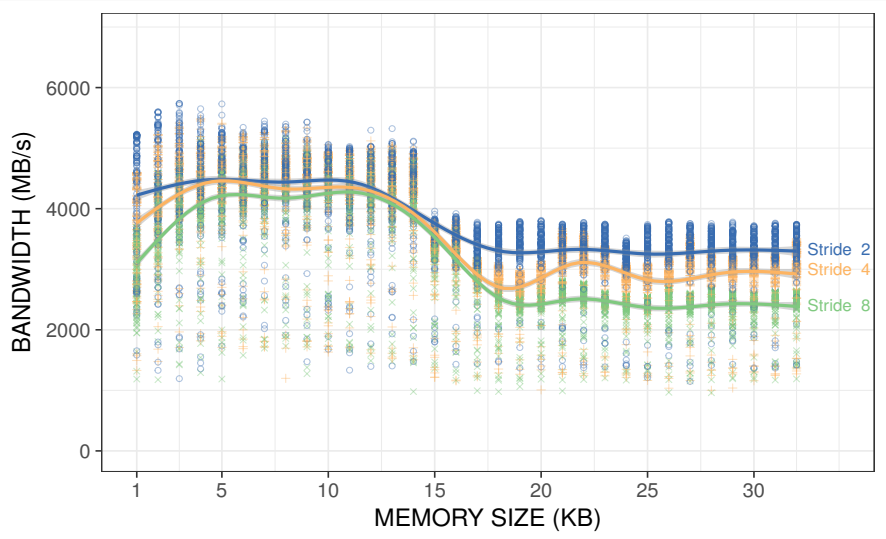

MultiMAPS on an Opteron

*Genetic Algorithms Approach to Modeling the Performance of Memory-bound Computations*, Tikir et. al. in SC'07

```
MultiMAPS(size, stride, nloops) {
  allocate buffer[size];
  timer_start();
  for rep in (1..nloops)
    for i in (0..size/stride)
      access buffer[stride*i];
  timer_stop();
  bandwidth = nb_access /
              elapsed_time;
  deallocate buffer;
}
```

# Impact of Working Set Size on Effective Bandwdith

- Cache hierarchy (L1, L2, L3, RAM) with different bandwidth
- LRU, pre-fetching for linear access
- An array fits in a cache level ⇒ operate at the corresponding bandwidth
- Stride access decrease bandwidth

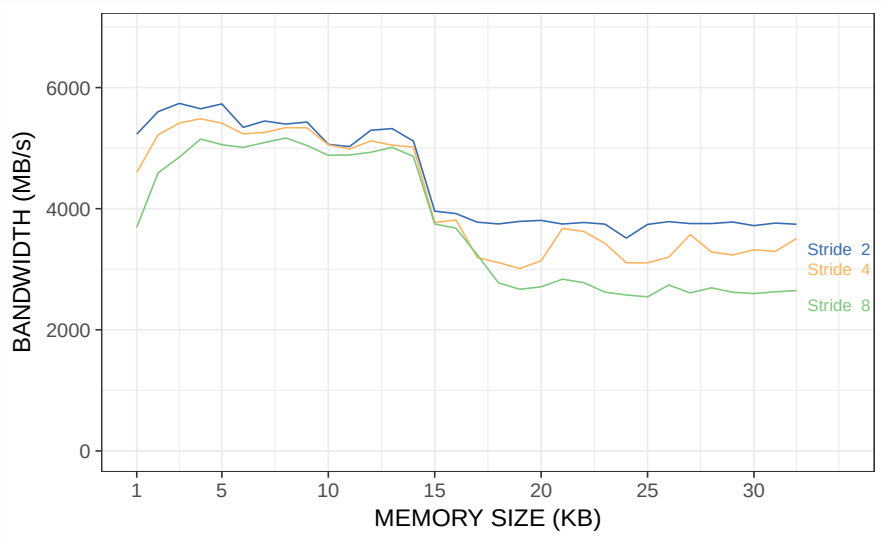



Our first attempt on a Pentium 4...

```
MultiMAPS(size, stride, nloops) {
  allocate buffer[size];
  timer_start();
  for rep in (1..nloops)
    for i in (0..size/stride)
       access buffer[stride*i];
  timer_stop();
  bandwidth = nb_access /
              elapsed_time;
  deallocate buffer;
}
```

# Impact of Working Set Size on Effective Bandwdith

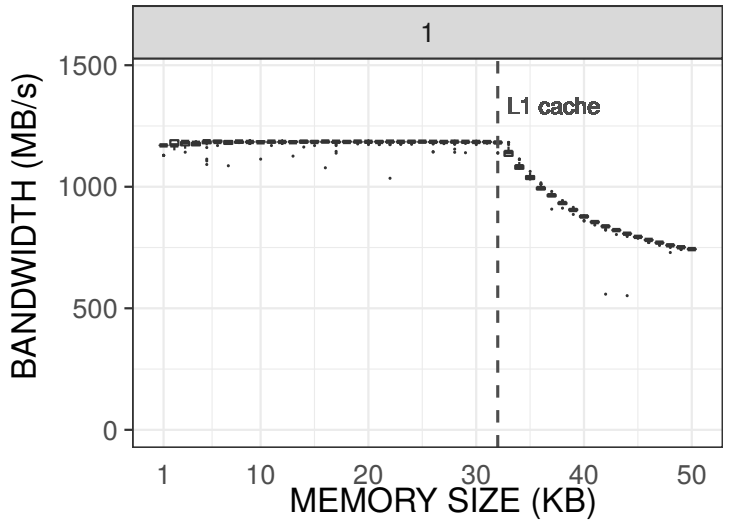
- Cache hierarchy (L1, L2, L3, RAM) with different bandwidth
- LRU, pre-fetching for linear access
- An array fits in a cache level ⇒ operate at the corresponding bandwidth
- Stride access decrease bandwidth




Our first attempt on a Pentium 4...

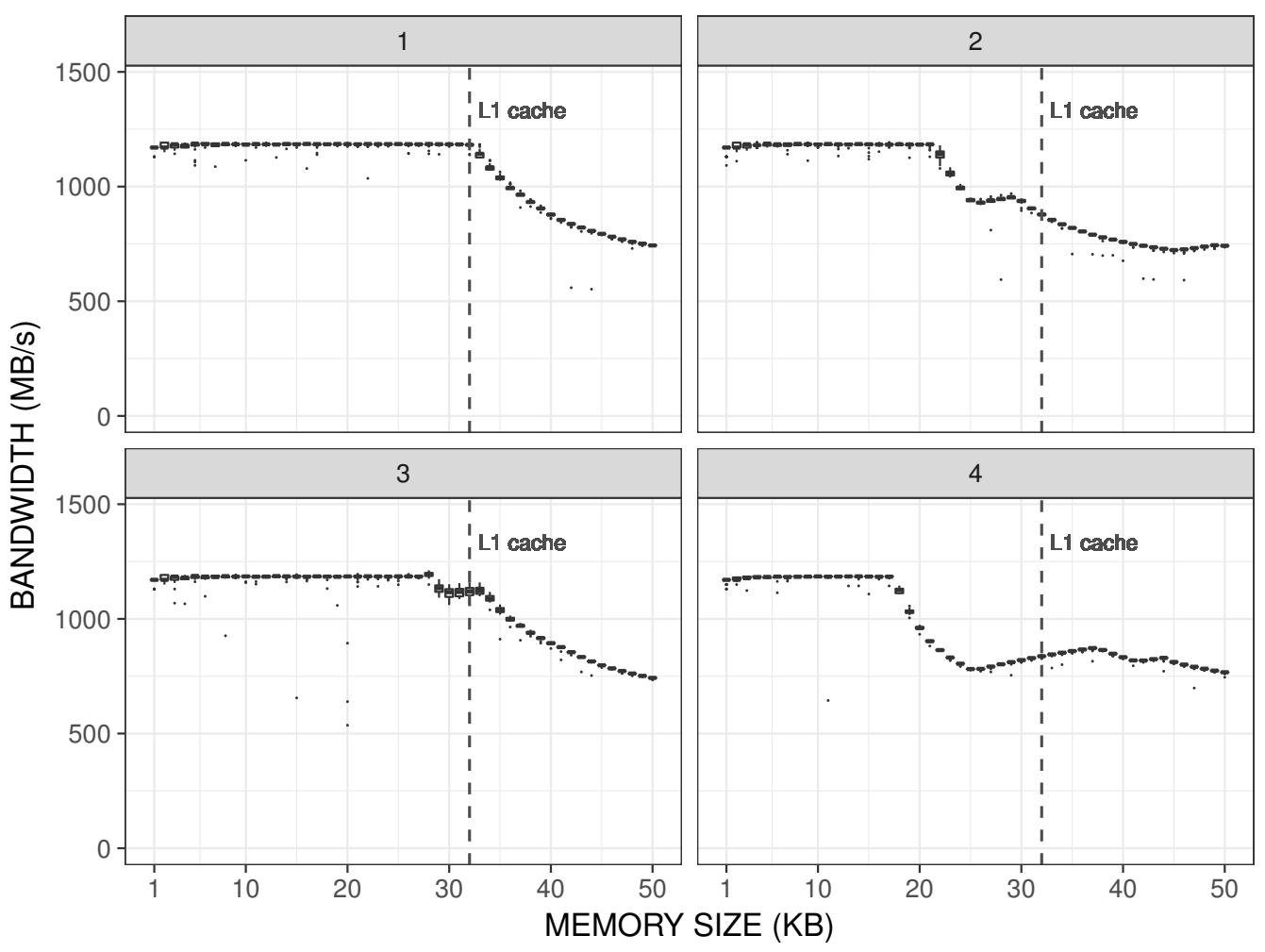
```
MultiMAPS(size, stride, nloops) {
  allocate buffer[size];
  timer_start();
  for rep in (1..nloops)
    for i in (0..size/stride)
       access buffer[stride*i];
  timer_stop();
  bandwidth = nb_access /
              elapsed_time;
  deallocate buffer;
}
```

# Impact of Architecture



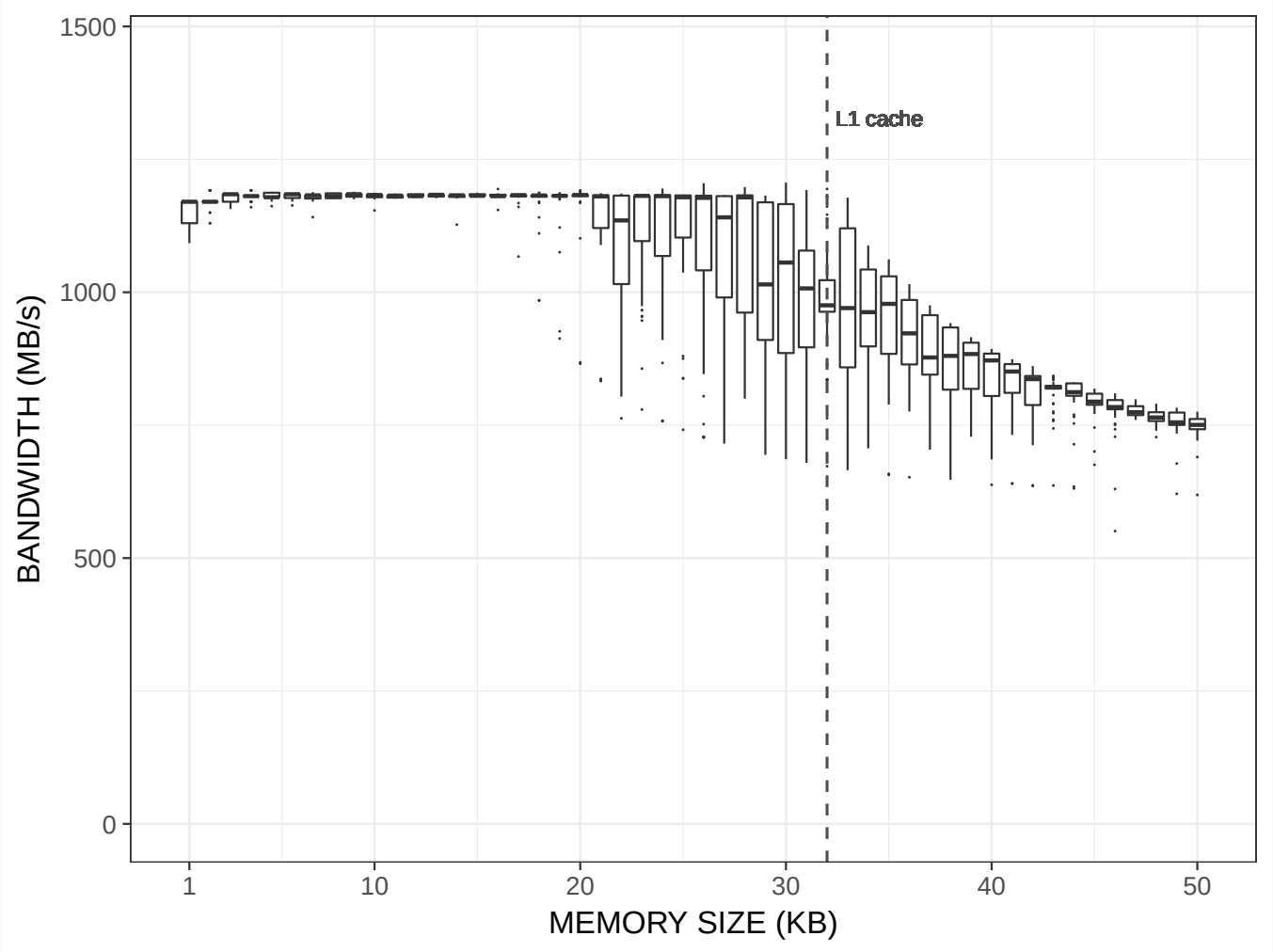

Courtesy of L. Stanisic

# Impact of Architecture




Courtesy of L. Stanisic

# Impact of Architecture (the ARM associativity issue)

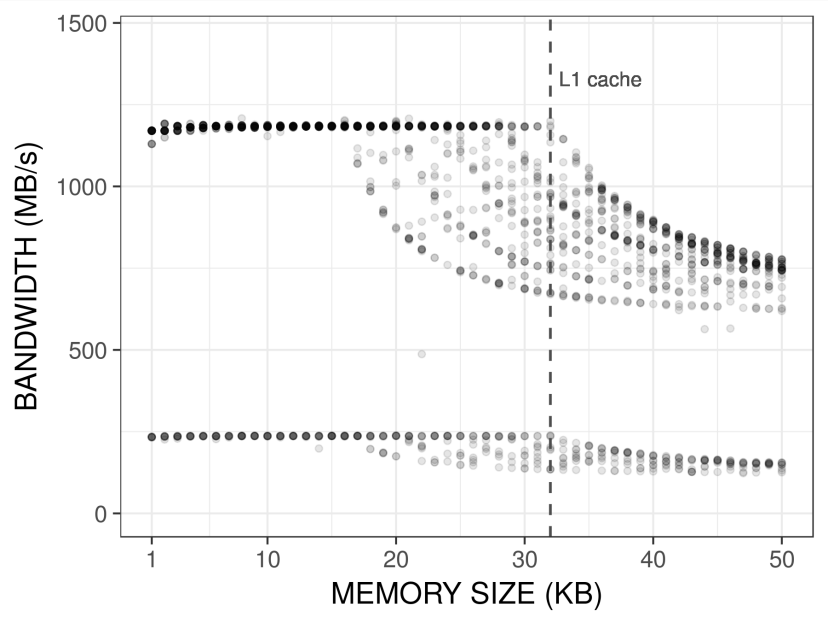


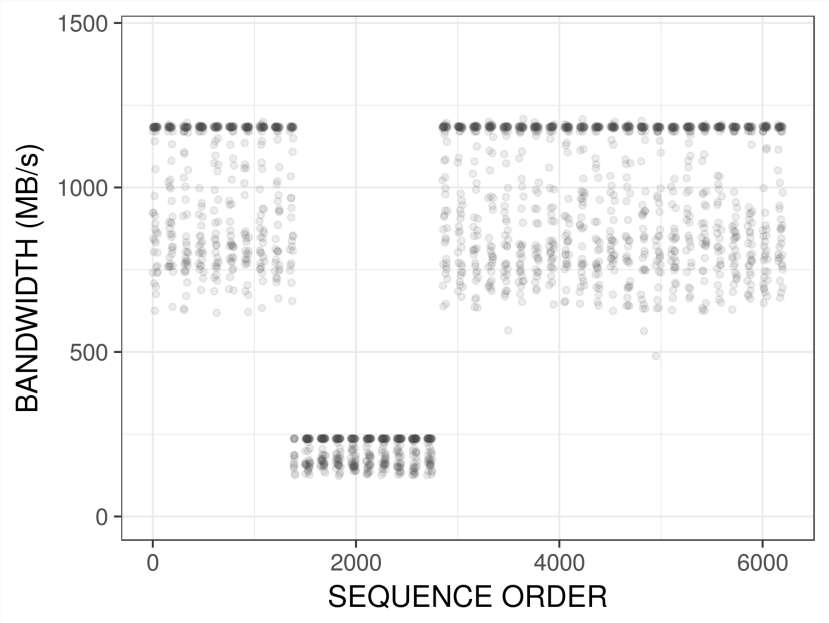

Randomize physical address start!

Courtesy of L. Stanisic

# Impact of Operating System Scheduler

- Activating real-time kernel scheduler

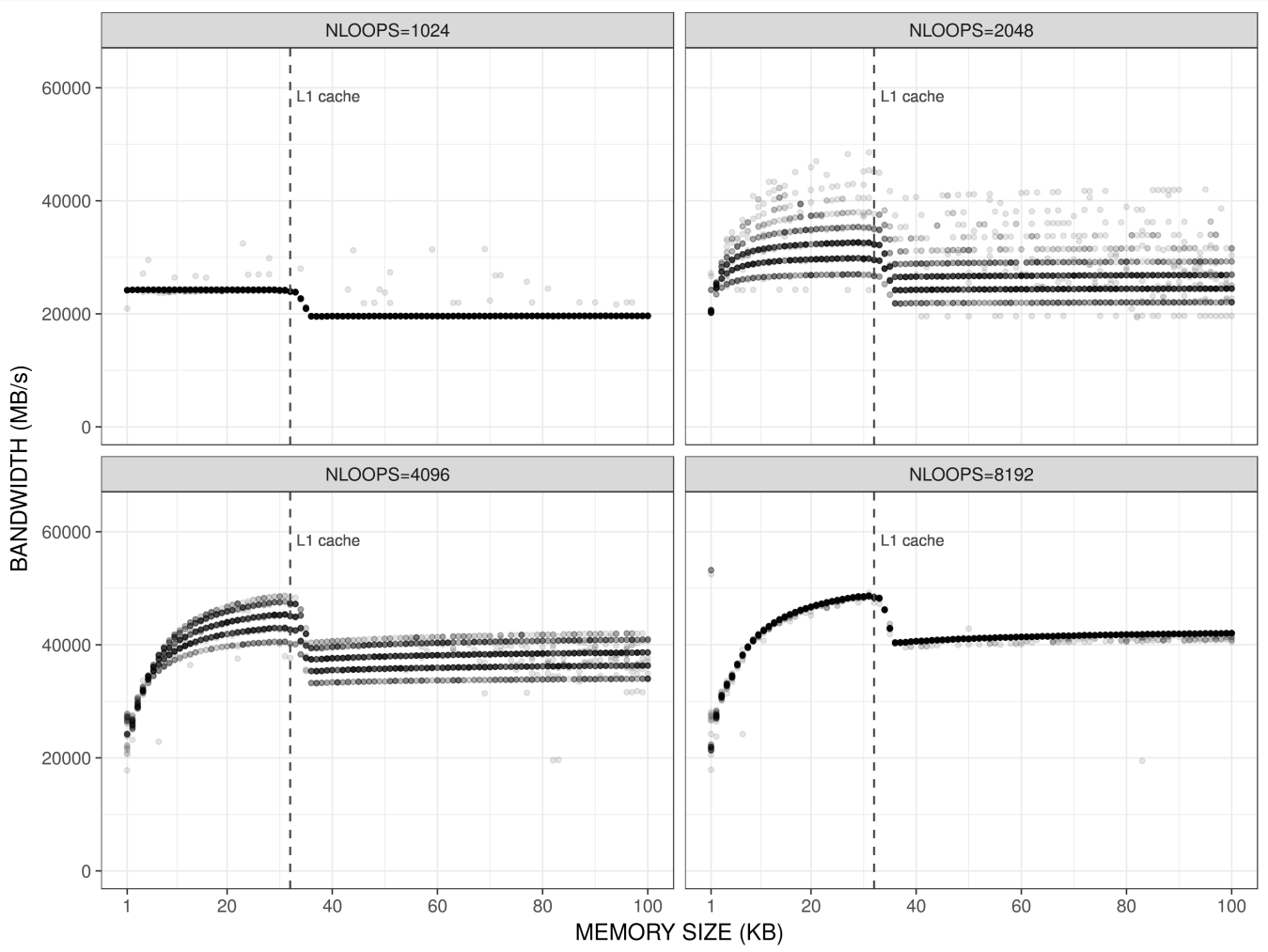



Courtesy of L. Stanisic

# Impact of Repetitions

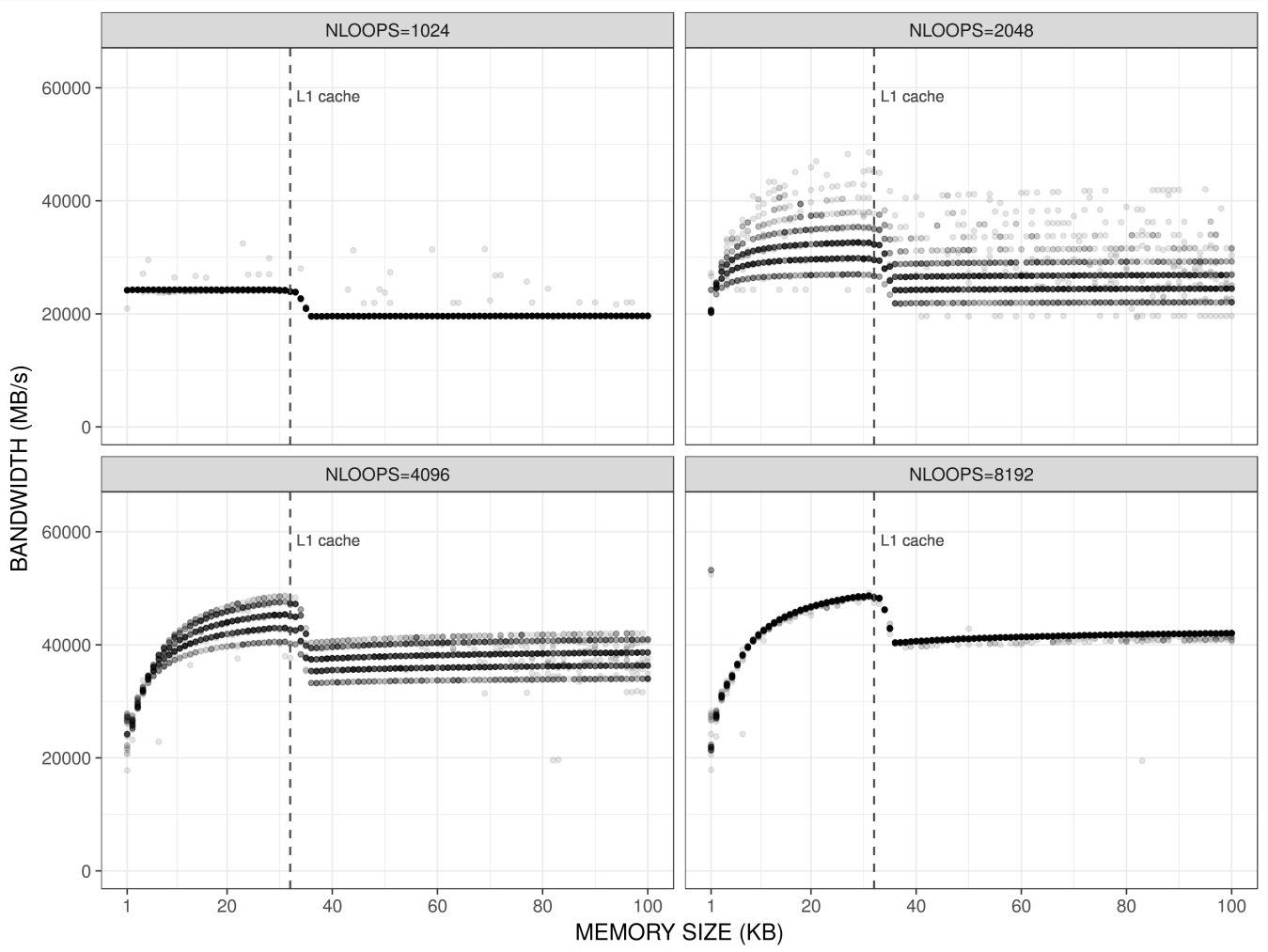
- Remember `nloops` ?




Courtesy of L. Stanisic

# Impact of ~~Repetitions~~ DVFS

- Remember `nloops` ?

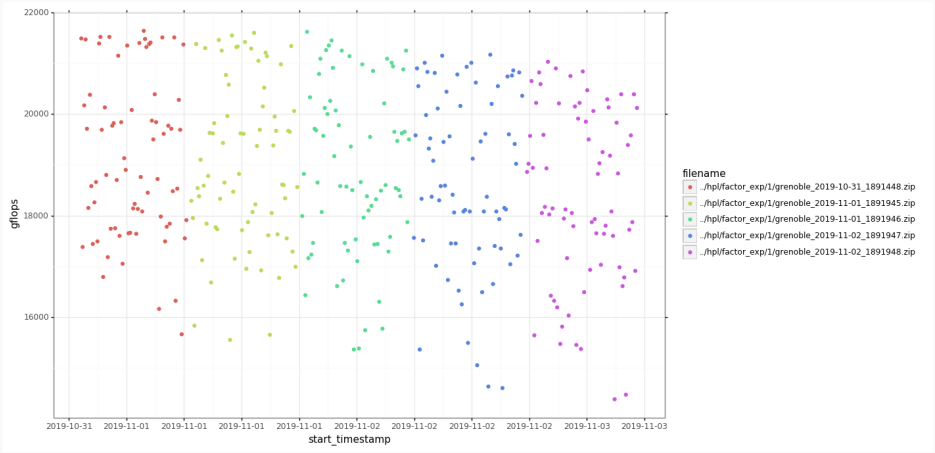



Courtesy of L. Stanisic

# CPU PERFORMANCE

# Avoiding "Temporary" Perturbations (Randomizing a Factorial Design)

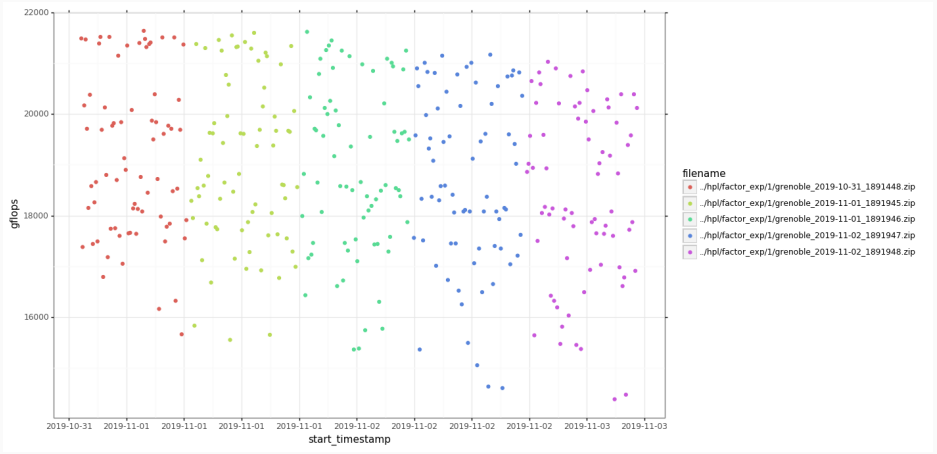
- HPL performance (70 config., 5 repetitions)
- Time scale = 3 days

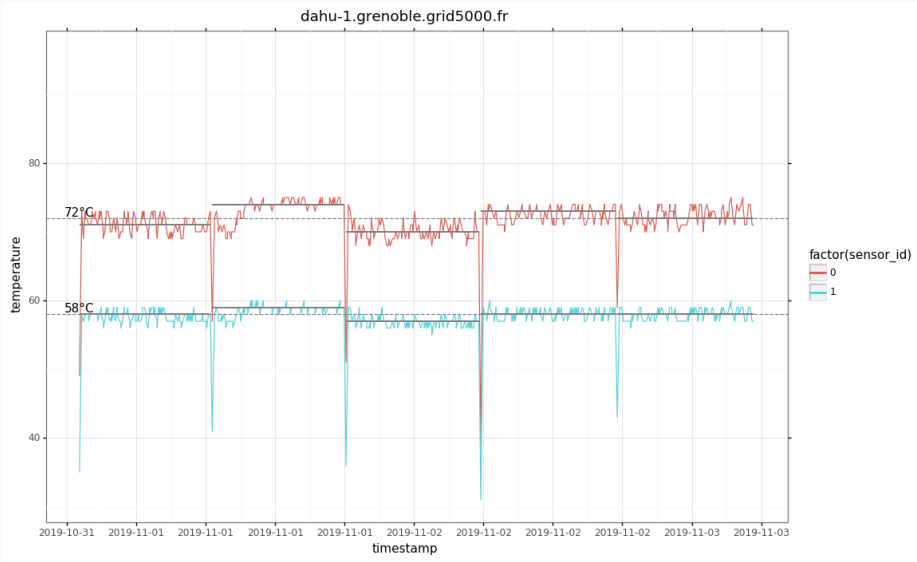


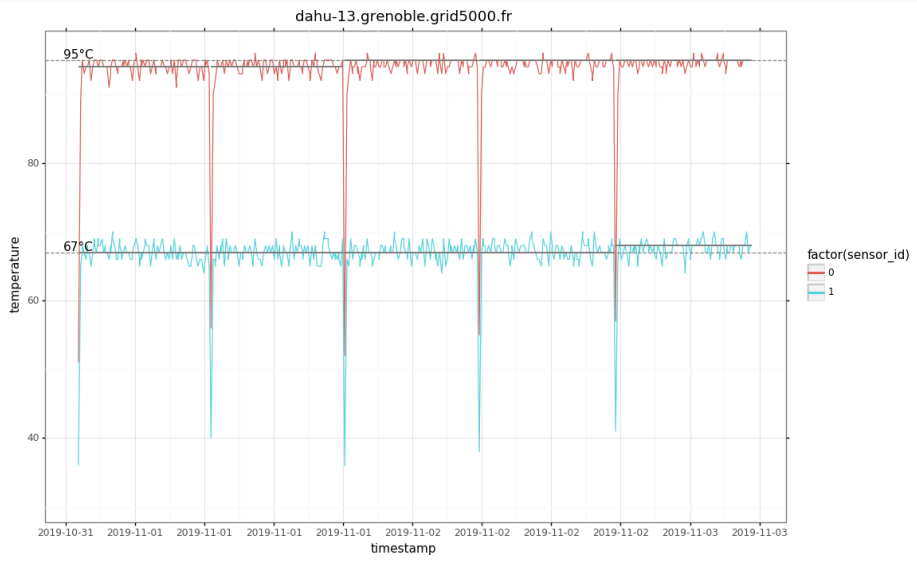

Courtesy of T. Cornebize

# Avoiding "Temporary" Perturbations (Randomizing a Factorial Design)

- HPL performance (70 config., 5 repetitions)
- Time scale = 3 days



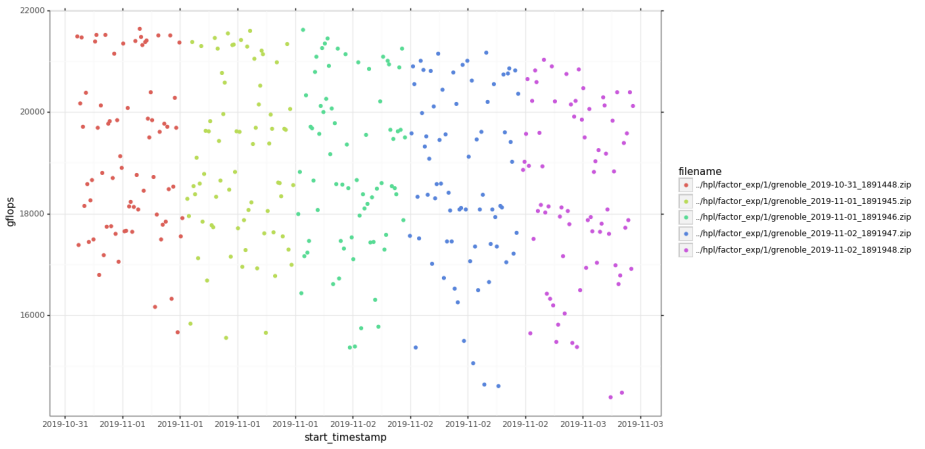



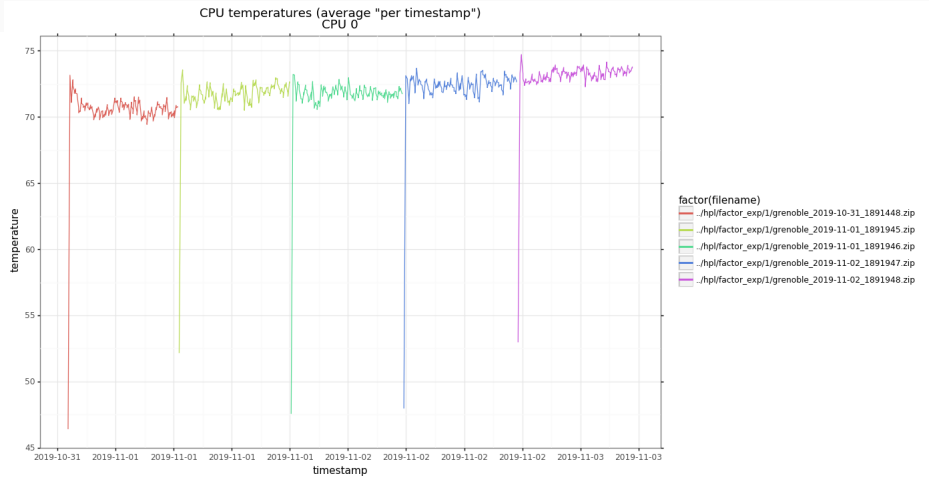



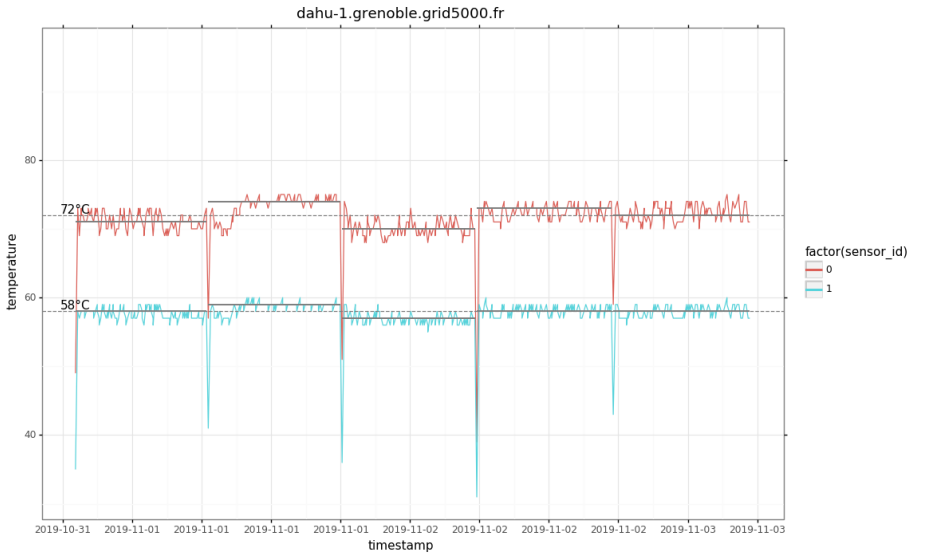

Courtesy of T. Cornebize

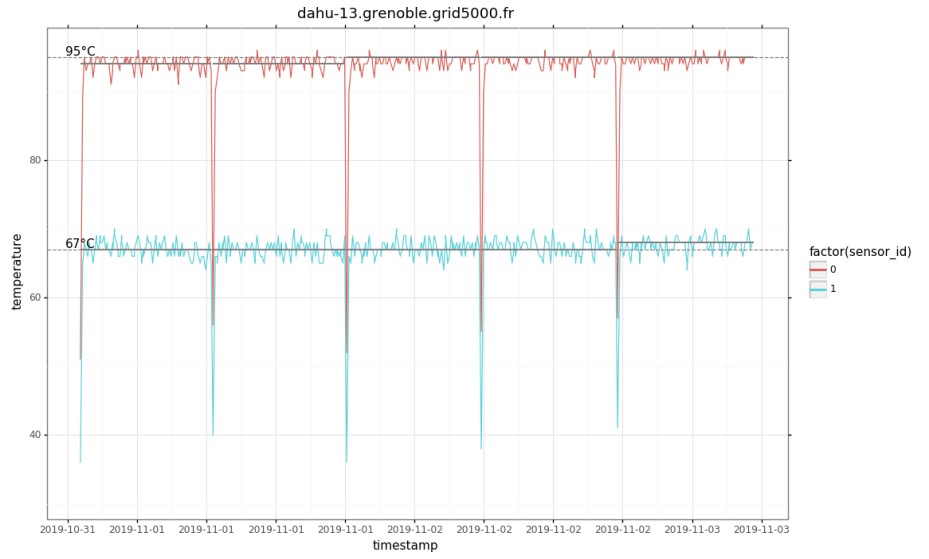
# Avoiding "Temporary" Perturbations (Randomizing a Factorial Design)

- HPL performance (70 config., 5 repetitions)
- Time scale = 3 days



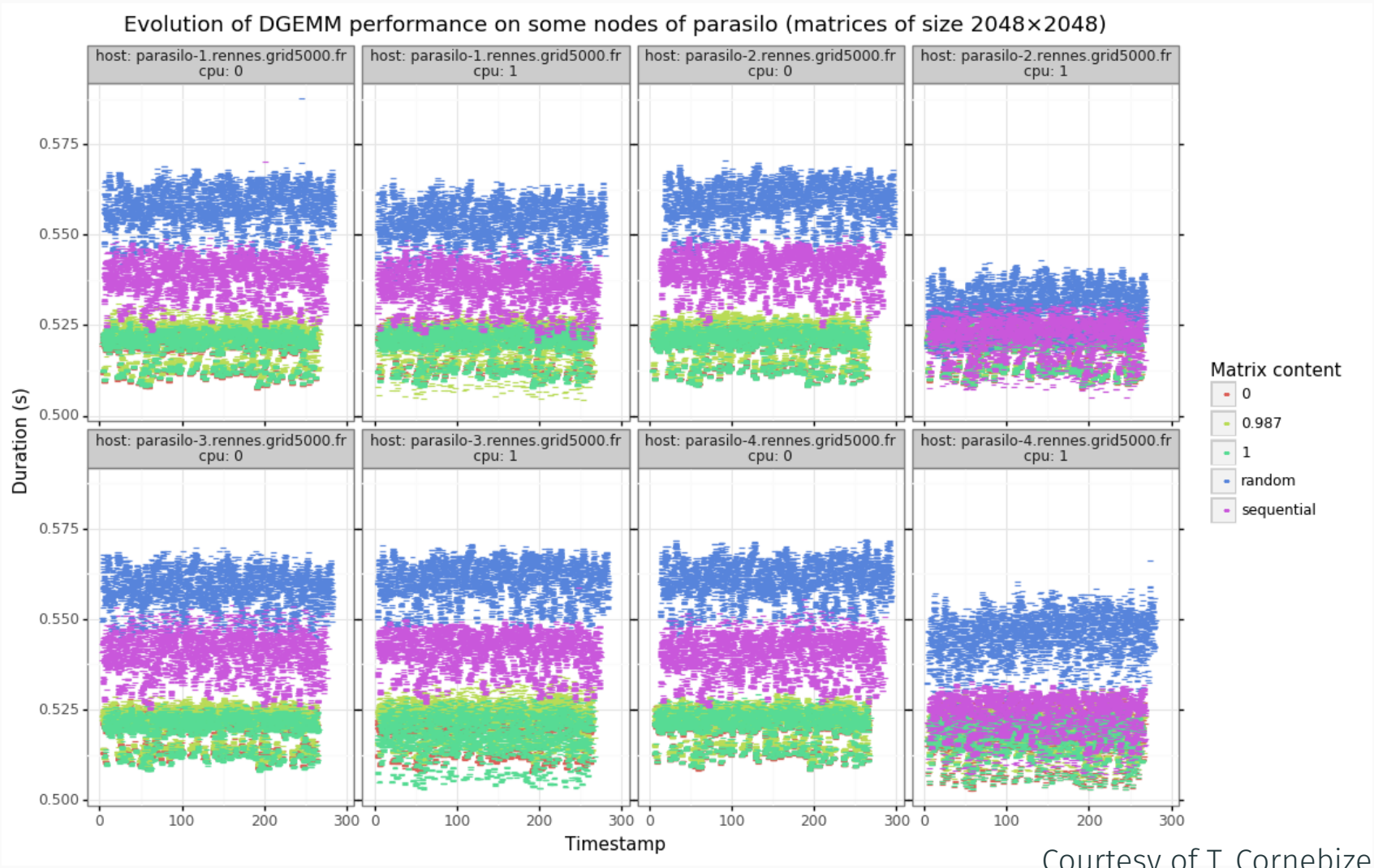

Courtesy of T. Cornebize

# On the Importance of Content Initialization

- $C = A \times A$, $2048 \times 2048$ matrix
- $A$ initialized with $\boxed{0}$ $\boxed{1}$ $\boxed{0.987}$ $\boxed{1, 2, 3, \ldots}$ $\boxed{\textit{random}}$?
- Time scale = 5 minutes

# On the Importance of Content Initialization

- $C = A \times A$, $2048 \times 2048$ matrix
- Time scale = 5 minutes
- $A$ initialized with $\boxed{0}$ $\boxed{1}$ $\boxed{0.987}$ $\boxed{1, 2, 3, \ldots}$ $\boxed{\textit{random}}$ ?




Courtesy of T. Cornebize

# On the Importance of Content Initialization

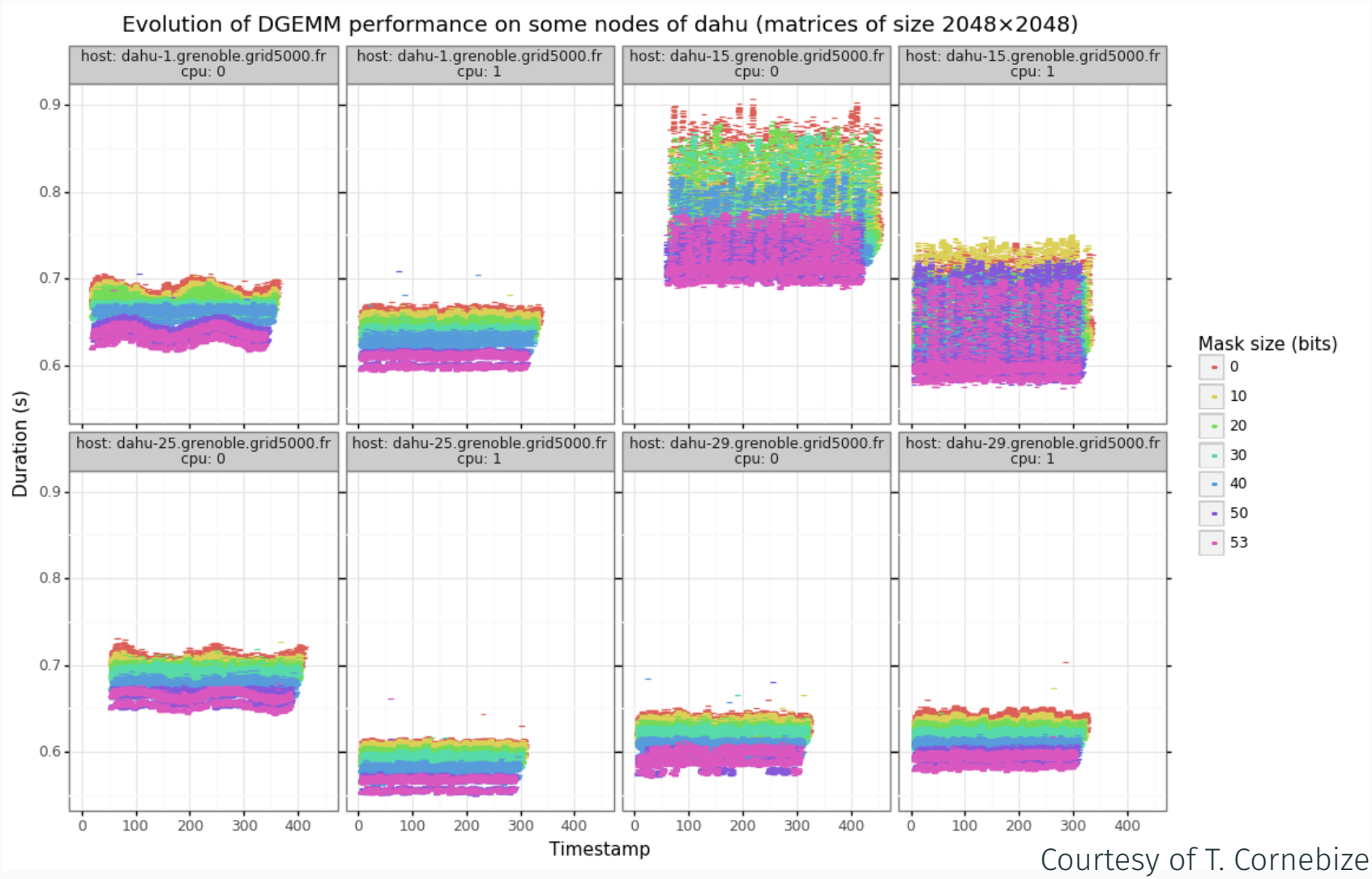
- $C = A \times A$, $2048 \times 2048$ matrix
- Time scale = 5 minutes
- $A$ initialized with $\boxed{0}$ $\boxed{1}$ $\boxed{0.987}$ $\boxed{1, 2, 3, \ldots}$ $\boxed{\textit{random}}$ ?


Bit-flips ⇒ NRJ Consumption ⇒ T°↑ + TDP ⇒ Frequency ⇒ Performance ?!?

Courtesy of T. Cornebize

# On the Importance of Content Initialization

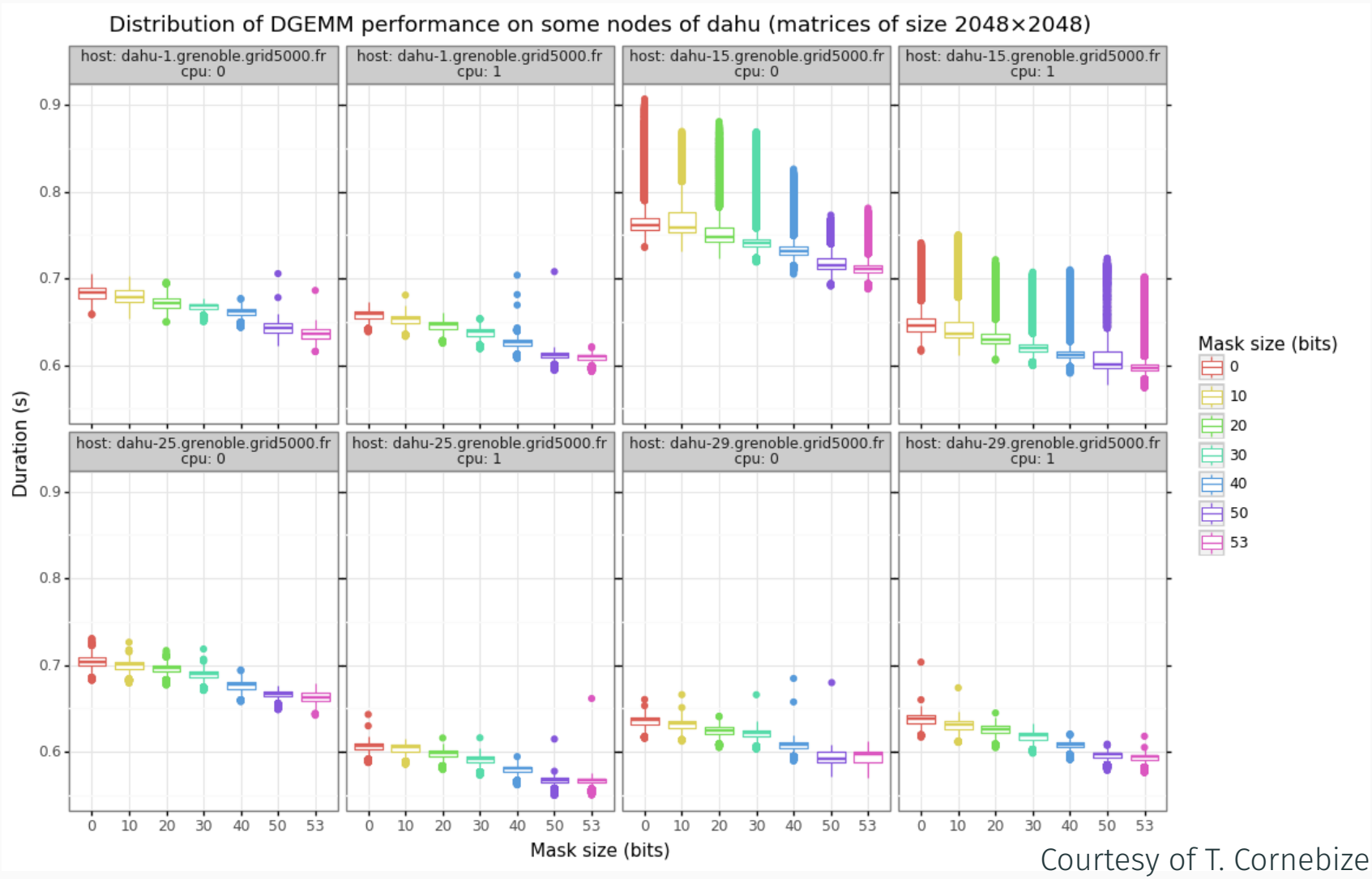
- $C = A \times A$, $2048 \times 2048$ matrix
- Time scale = 5 minutes
- $A$ initialized with $\boxed{0}$ $\boxed{1}$ $\boxed{0.987}$ $\boxed{1, 2, 3, \ldots}$ $\boxed{\textit{random}}$?




Courtesy of T. Cornebize

# On the Importance of Content Initialization

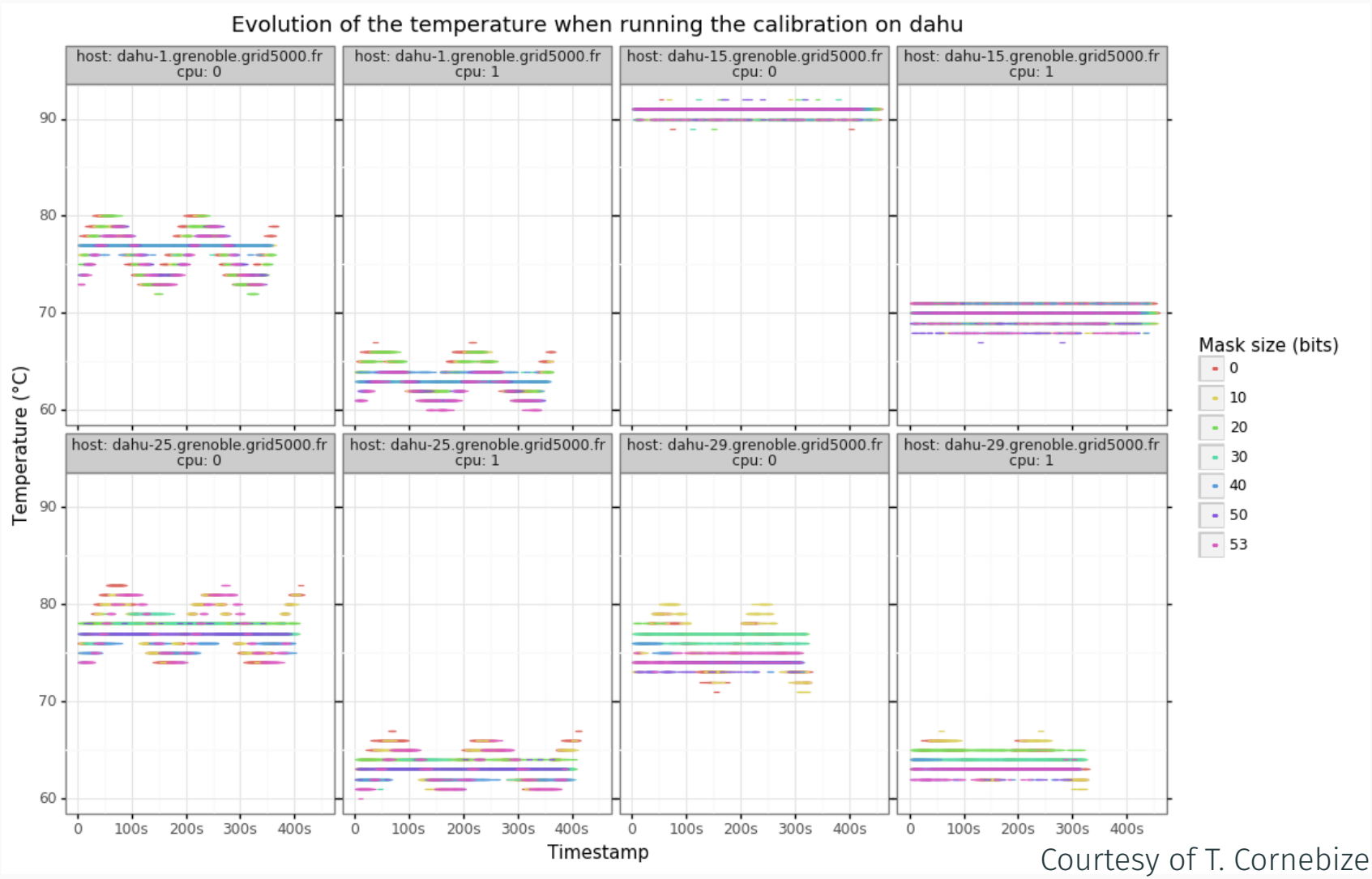
- $C = A \times A$, $2048 \times 2048$ matrix
- Time scale = 5 minutes
- $A$ initialized with [0] [1] [0.987] [$1, 2, 3, \ldots$] [*random*] ?




Courtesy of T. Cornebize

# On the Importance of Content Initialization

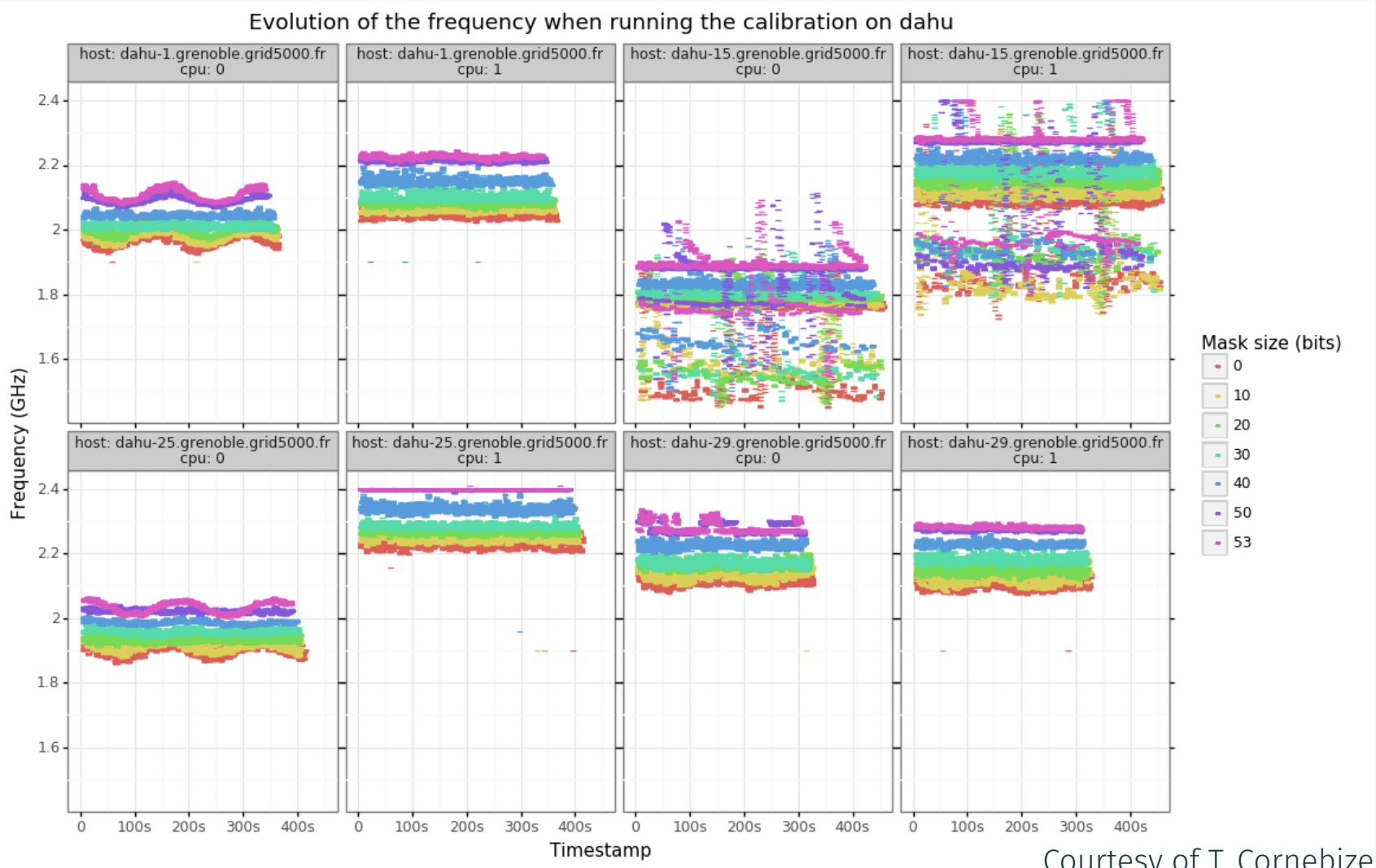
- $C = A \times A$, $2048 \times 2048$ matrix
- Time scale = 5 minutes
- $A$ initialized with 0 | 1 | 0.987 | $1, 2, 3, \ldots$ | *random* ?




Courtesy of T. Cornebize

# On the Importance of Content Initialization

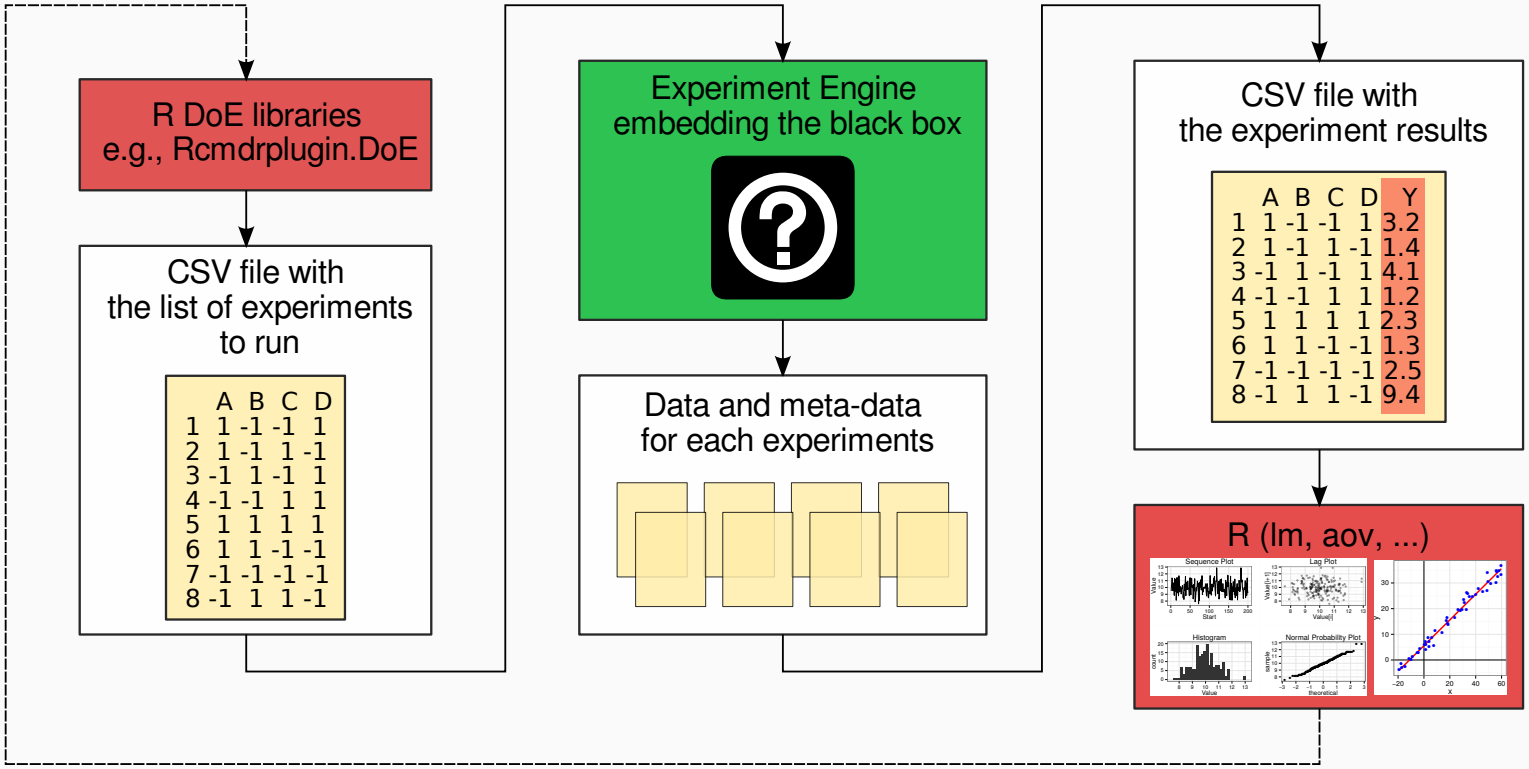
- $C = A \times A$, $2048 \times 2048$ matrix
- Time scale = 5 minutes
- $A$ initialized with [0] [1] [0.987] [$1, 2, 3, \ldots$] [*random*] ?




Courtesy of T. Cornebize

# Conclusion

# Experimental Methodology: Noticing the Unexpected




1. A **separation of concerns**
   - Transparent Measurement Procedure and Analysis Procedure
2. **Randomized and Designed Experiments** allowing to both:
   - Check the model and Instanciate it
3. Careful **recording of all experimental parameters** (before and during XPs)

# Reproducible Research = Rigor and Transparency

To err is human.

**Good research requires time and resources**

1. **Train yourself and your students**: RR, statistics, experiments
   - Beware of checklists and norms
   - Understand what's at stake
2. **Change the norm:** make publication practices evolve
3. **Incentive**: consider RR/open science when hiring/promoting

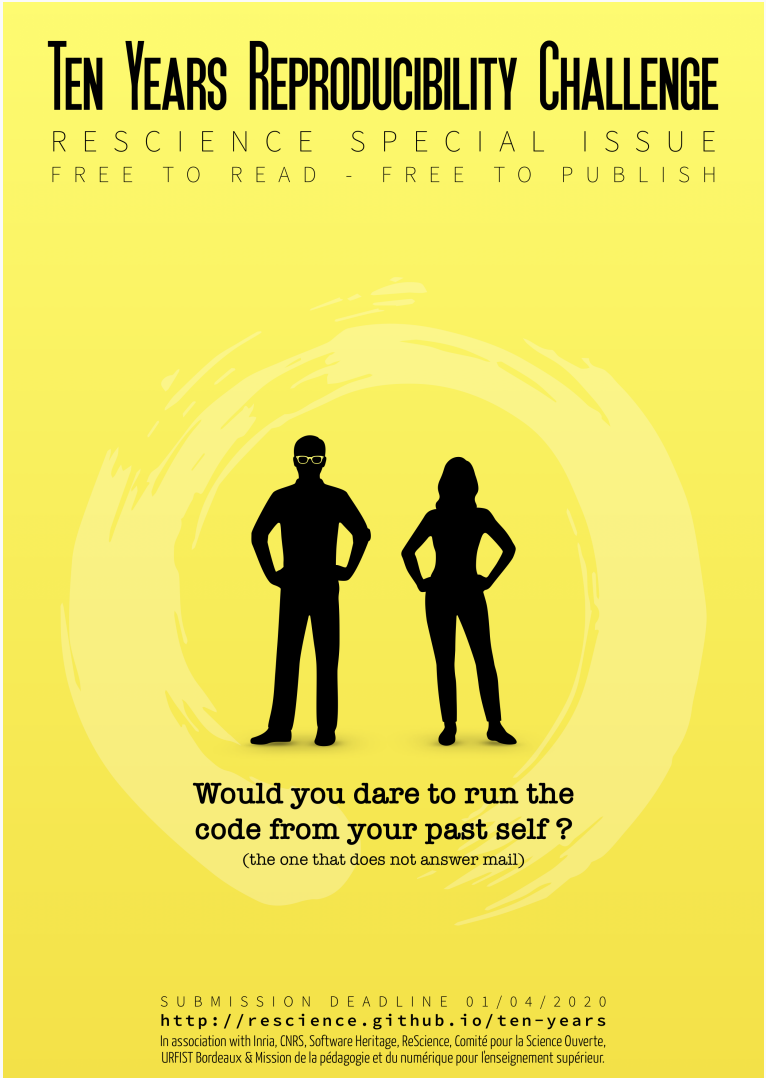
# Reproducible Research = Rigor and Transparency

To err is human.

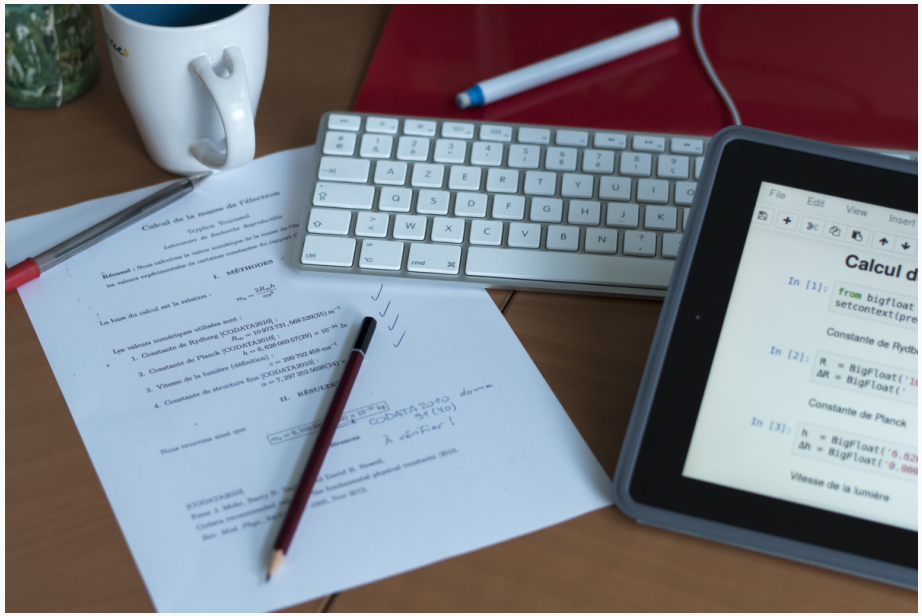
**Good research requires time and resources**

1. **Train yourself and your students**: RR, statistics, experiments
   - Beware of checklists and norms
   - Understand what's at stake
2. **Change the norm:** make publication practices evolve
3. **Incentive**: consider RR/open science when hiring/promoting
4. **Prepare the Future:** Toward literate experimentation?
   - Reuse, reuse, reuse!
   - Shared and controled testbeds (e.g., Grid'5000/SILECS)
   - How to share Experiments ?

# Some Advertising




http://rescience.github.io/ten-years/



3rd Edition: ≈ Feb. 2020

A new MOOC: "Advanced RR"

- Software environment control (Docker)
- Scientific workflow (snakemake)
- Managing data (HDF5, archiving)

October 2020 ?