

OBTAINING FAITHFUL/REPRODUCIBLE MEASUREMENTS ON MODERN CPUS

Tom Cornebize and Arnaud Legrand

Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP

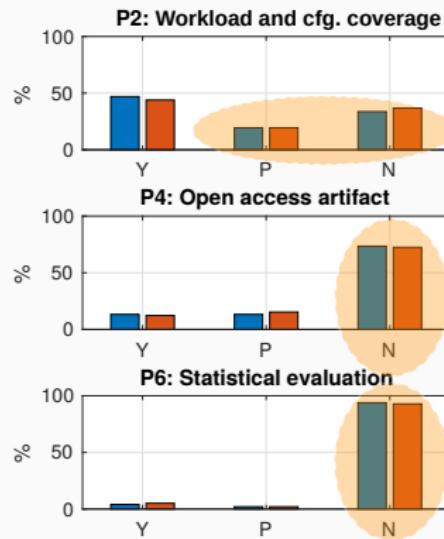
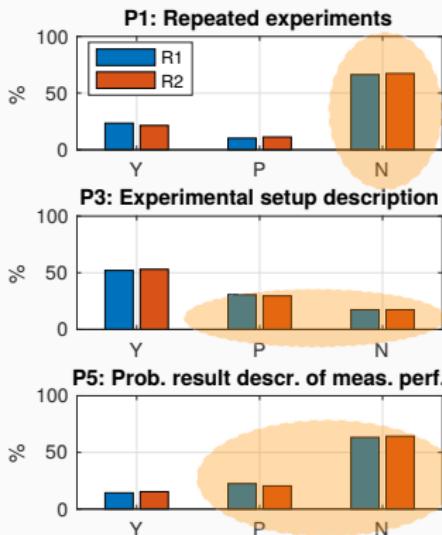
Nantes, Journées du GDR RSD, January 2020



TOWARDS REPRODUCIBLE COMPUTER SCIENCE RESEARCH ?

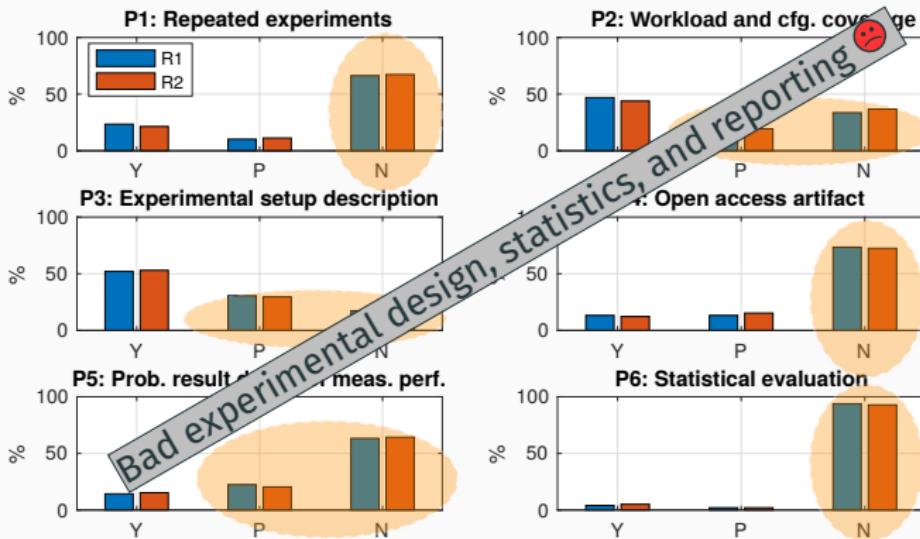
KEY CONCERN FOR OUR COMMUNITY (ROOM FOR IMPROVEMENT)

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 CONCERN 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:

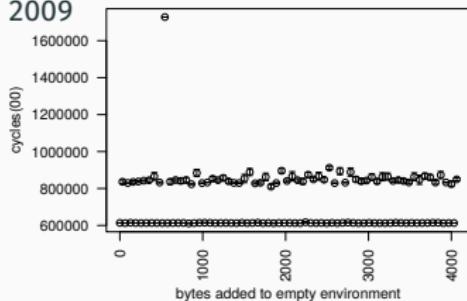
1. Replicate to increase reliability.
2. Randomize to reduce bias ↪ Evaluate statistical confidence.

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%.

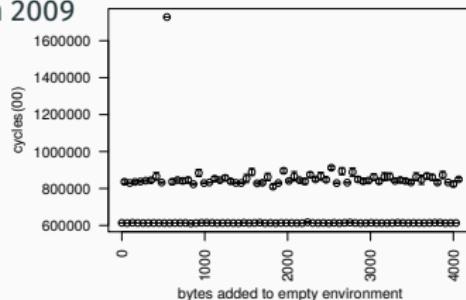


MEASURING PERFORMANCE IS DIFFICULT

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

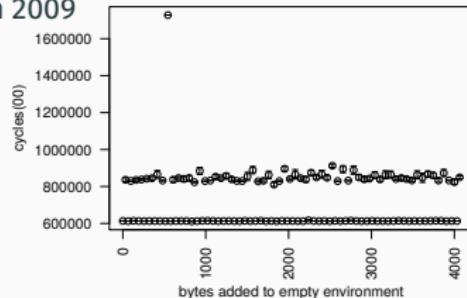
STABILIZER forces executions to sample the space of memory configurations by *repeatedly rerandomizing* layouts of code, stack, and heap objects at run-time. [...] Re-randomization ensures that layout effects *follow a Gaussian distribution*, enabling the use of statistical tests like ANOVA.

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 incurred by:

1. specific configurations $AA \dots A \rightarrow A_1 A_2 \dots A_n$ (*pseudo-replication*)
2. temporary perturbations $AA \dots A BB \dots B \rightarrow ABBAAB \dots$

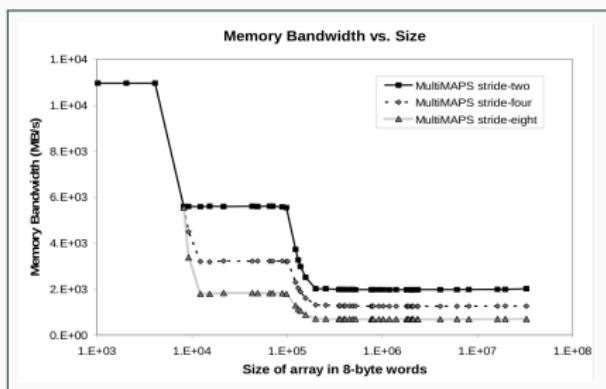
CACHE EFFECTS!

IMPACT OF WORKING SET SIZE ON EFFECTIVE BANDWIDTH

- 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 BANDWIDTH

- 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



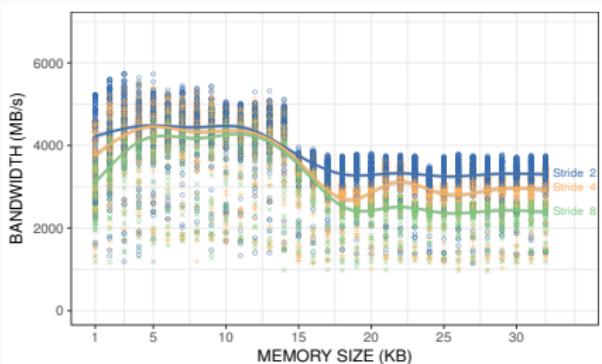
MultiMAPS on an Opteron

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 BANDWIDTH

- 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

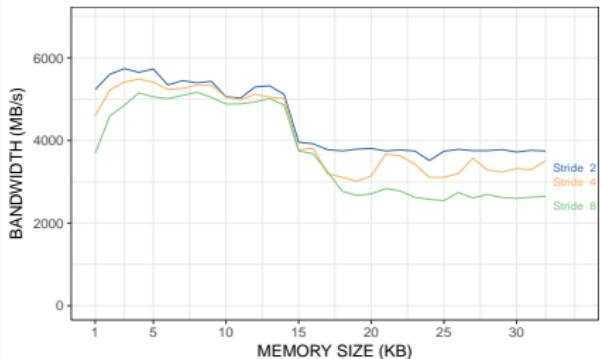


Our first attempt on a Pentium 4...

```
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 BANDWIDTH

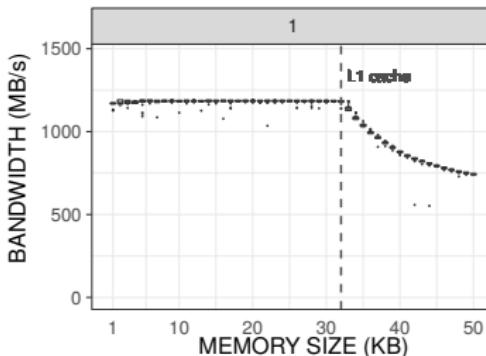
- Cache hierarchy (L1, L2, L3, RAM) with different bandwidth
- LRU, pre-fetching for linear access
- An array fits in a cache level \Rightarrow operate at the corresponding bandwidth
- Stride access decrease bandwidth



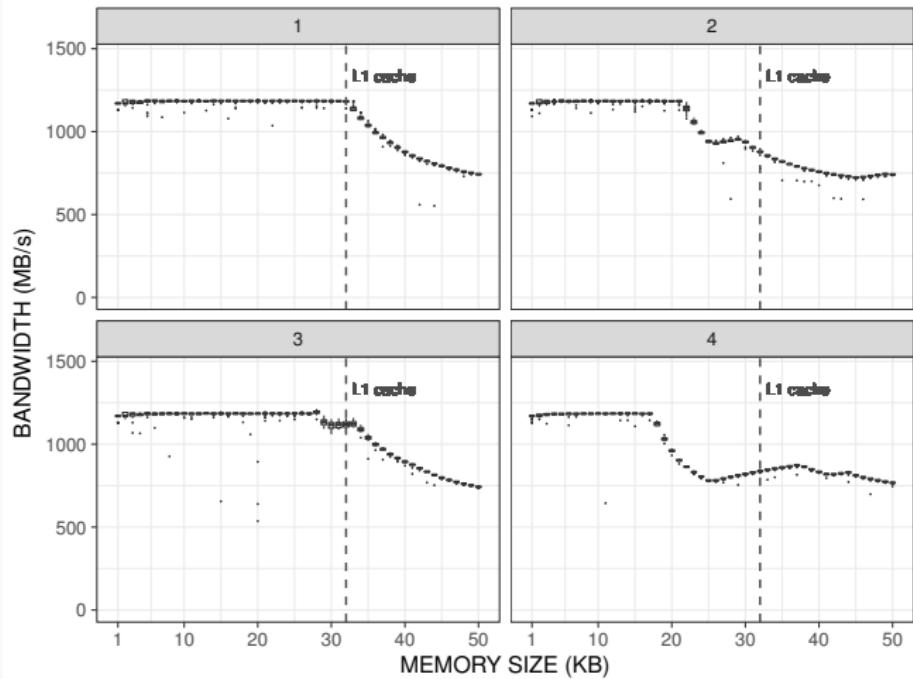
Our first attempt on a Pentium 4...

```
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;  
}
```

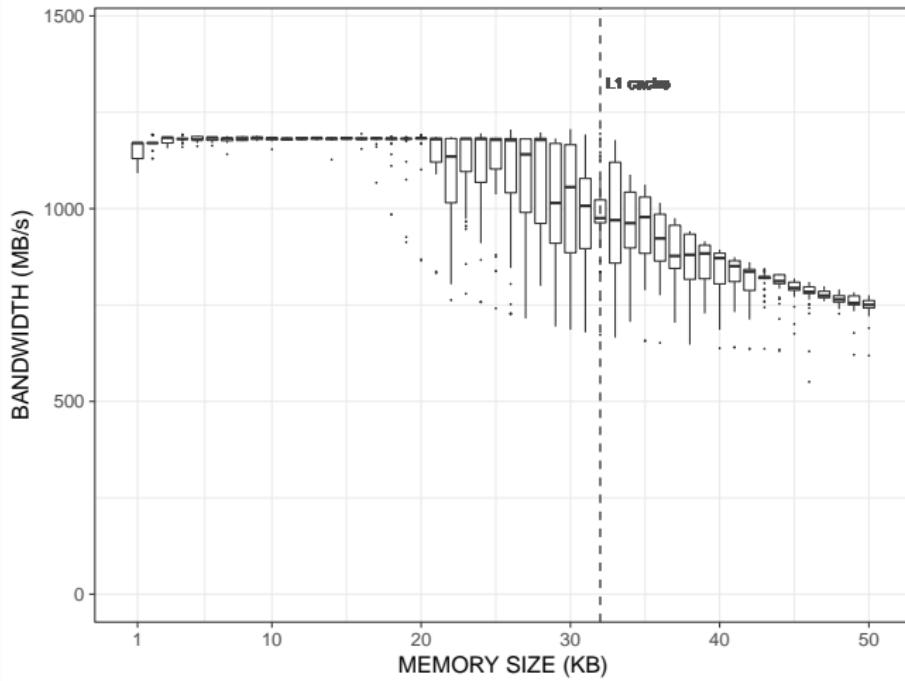
IMPACT OF ARCHITECTURE



IMPACT OF ARCHITECTURE



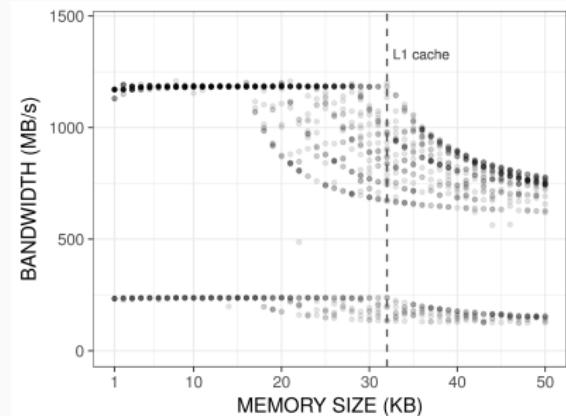
IMPACT OF ARCHITECTURE (THE ARM ASSOCIATIVITY ISSUE)



Randomize physical address start!

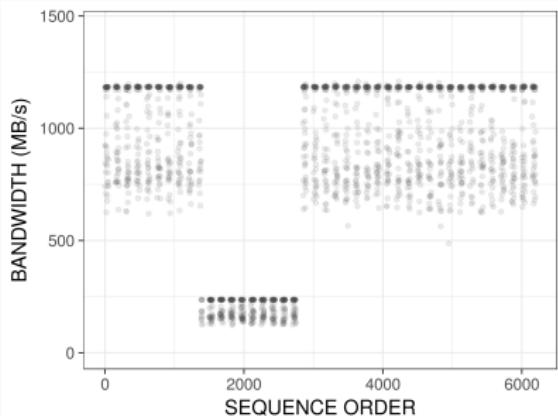
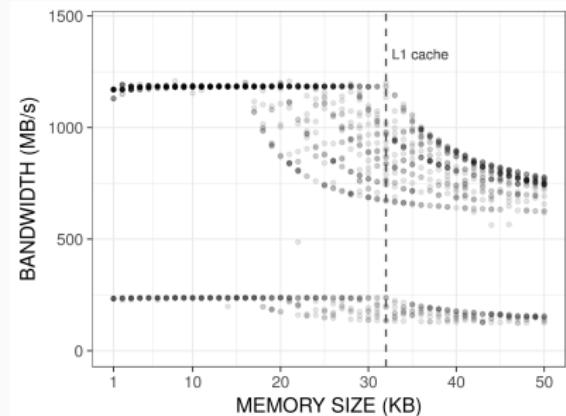
IMPACT OF OPERATING SYSTEM SCHEDULER

- Activating real-time kernel scheduler



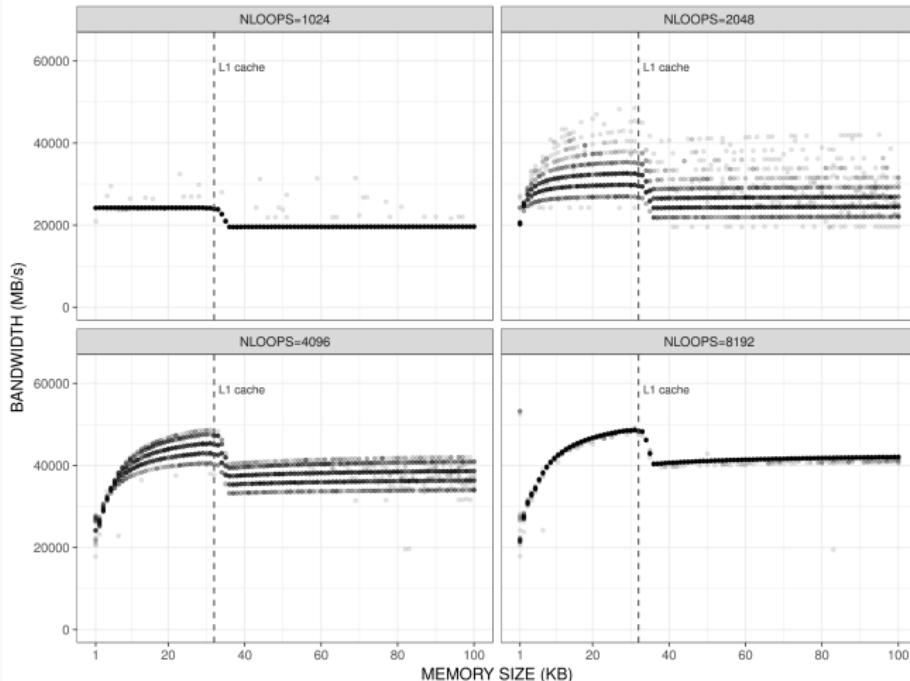
IMPACT OF OPERATING SYSTEM SCHEDULER

- Activating real-time kernel scheduler



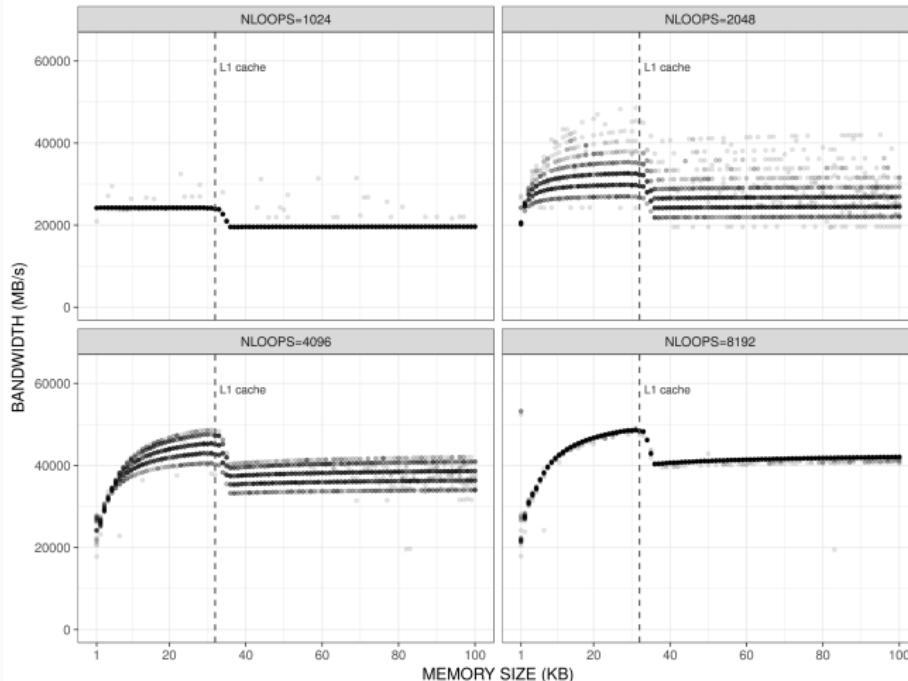
IMPACT OF REPETITIONS

- Remember `nloops` ?



IMPACT OF REPETITIONS DVFS

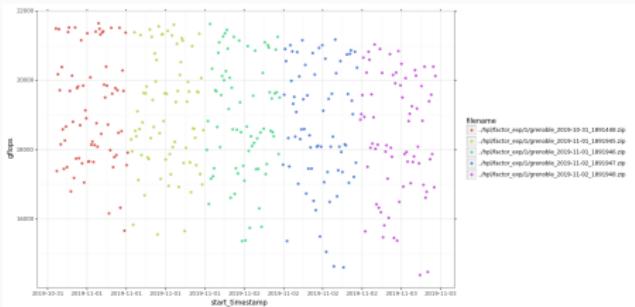
- Remember nloops ?



CPU PERFORMANCE

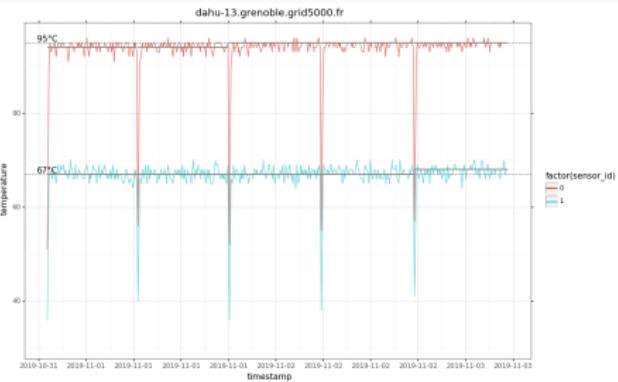
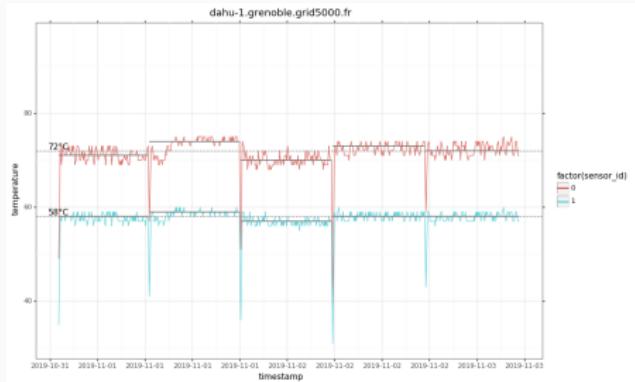
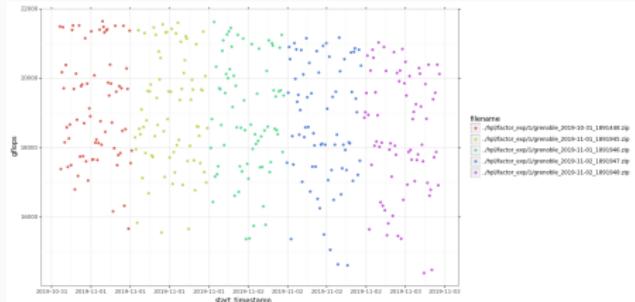
AVOIDING "TEMPORARY" PERTURBATIONS (RANDOMIZING A FACTORIAL DESIGN)

- HPL performance (32 nodes, 70 cfg., 5 repetitions) • Time scale = 3 days



AVOIDING "TEMPORARY" PERTURBATIONS (RANDOMIZING A FACTORIAL DESIGN)

- HPL performance (32 nodes, 70 cfg., 5 repetitions) • Time scale = 3 days



AVOIDING "TEMPORARY" PERTURBATIONS (RANDOMIZING A FACTORIAL DESIGN)

- HPL performance (32 nodes, 70 cfg., 5 repetitions) • Time scale = 3 days



Experiments:

- `ApplFactor_explgridnode_2019-10-31_180144.zip`
- `ApplFactor_explgridnode_2019-11-01_180145.zip`
- `ApplFactor_explgridnode_2019-11-01_180146.zip`
- `ApplFactor_explgridnode_2019-11-01_180147.zip`
- `ApplFactor_explgridnode_2019-11-02_180148.zip`

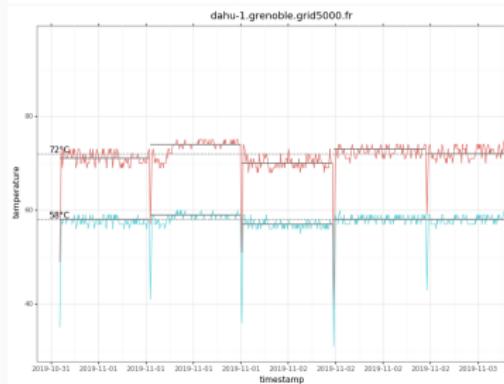


CPU temperatures (average "per timestamp")

CPU 0

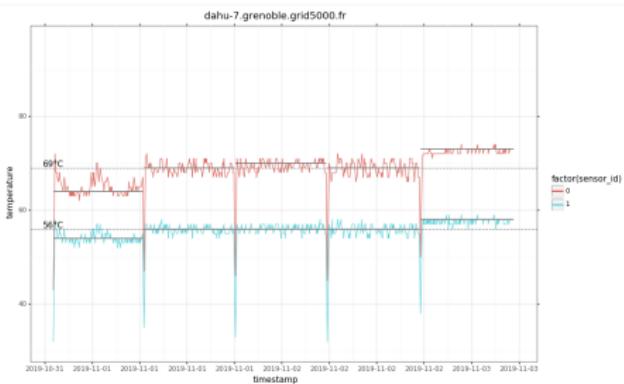
Factor/Experiment:

- `ApplFactor_explgridnode_2019-10-31_180144.zip`
- `ApplFactor_explgridnode_2019-11-01_180145.zip`
- `ApplFactor_explgridnode_2019-11-01_180146.zip`
- `ApplFactor_explgridnode_2019-11-01_180147.zip`
- `ApplFactor_explgridnode_2019-11-02_180148.zip`



factor(sensor_id)

0
1



factor(sensor_id)

0
1

Courtesy of T. Cornebize

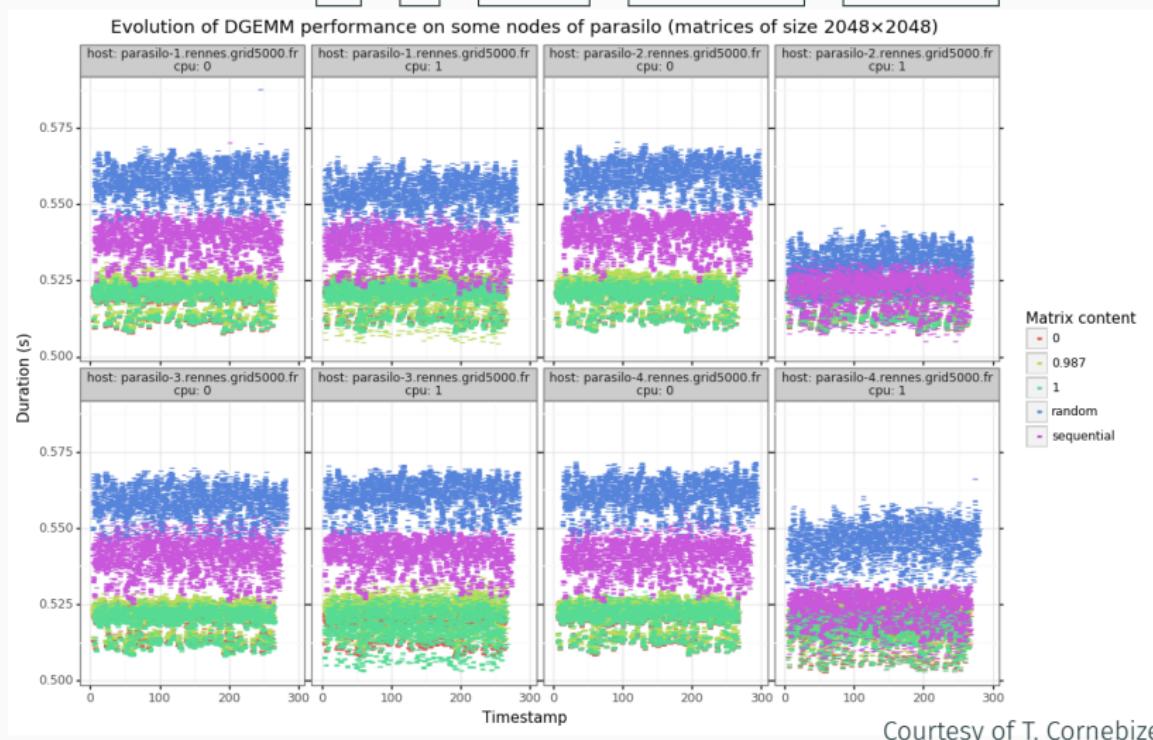
ON THE IMPORTANCE OF CONTENT INITIALIZATION

- $C = A \times A$ (2048×2048), independant
- Time scale = 5 minutes
- A initialized with

ON THE IMPORTANCE OF CONTENT INITIALIZATION

- $C = A \times A$ (2048×2048), independant
- A initialized with

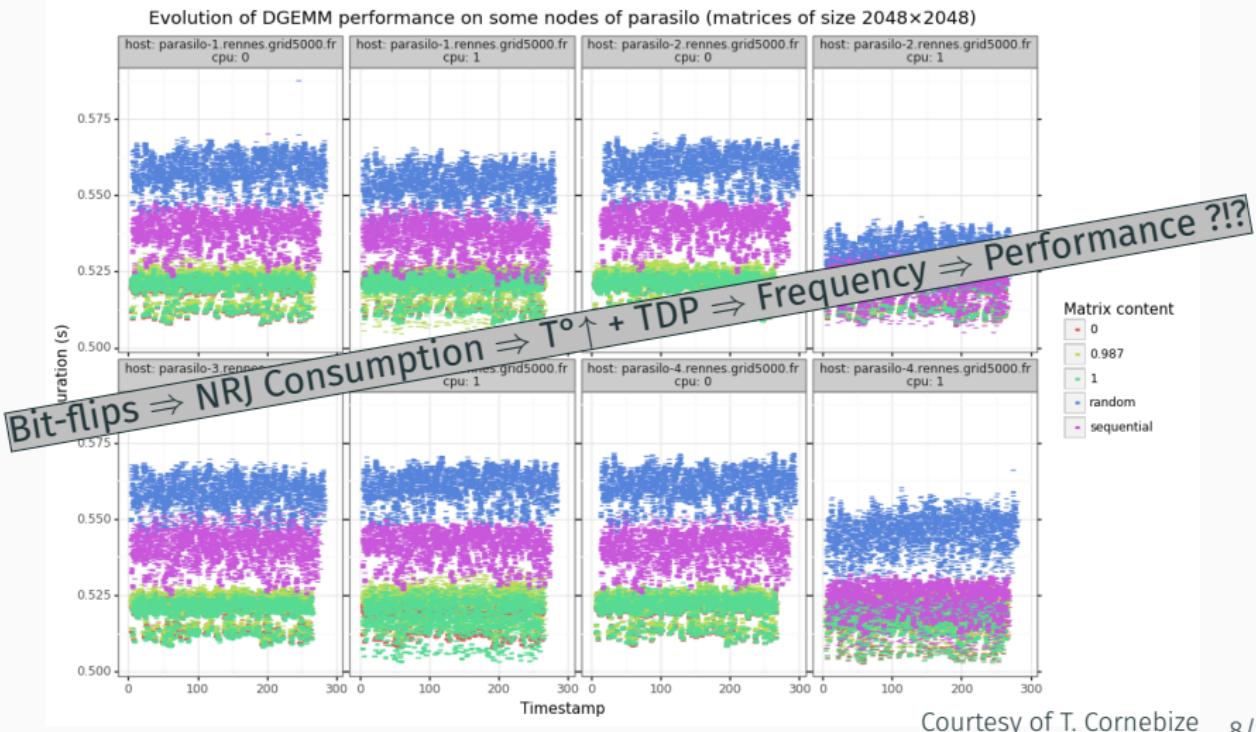
0	1	0.987	1, 2, 3, ...	random ?
---	---	-------	--------------	----------
- Time scale = 5 minutes



ON THE IMPORTANCE OF CONTENT INITIALIZATION

- $C = A \times A$ (2048×2048), independant
- A initialized with

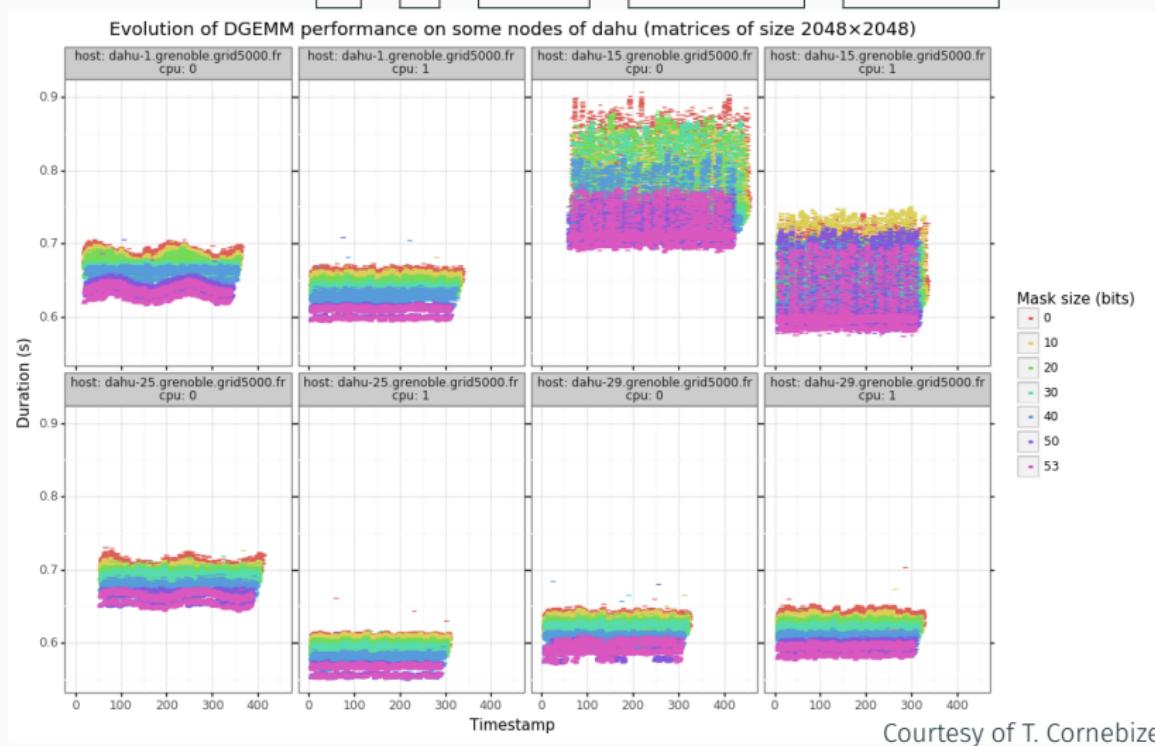
0	1	0.987	1, 2, 3, ...	random ?
---	---	-------	--------------	----------
- Time scale = 5 minutes



ON THE IMPORTANCE OF CONTENT INITIALIZATION

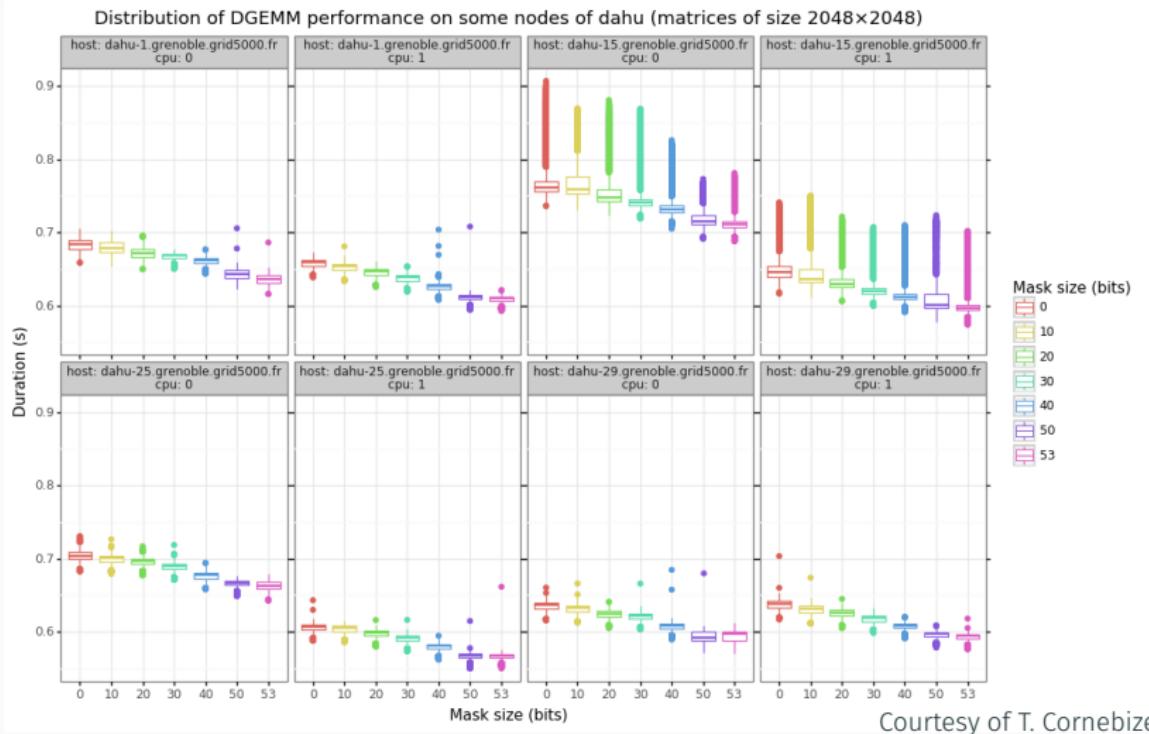
- $C = A \times A$ (2048×2048), independant
- A initialized with

0	1	0.987	1, 2, 3, ...	random ?
---	---	-------	--------------	----------
- Time scale = 5 minutes



ON THE IMPORTANCE OF CONTENT INITIALIZATION

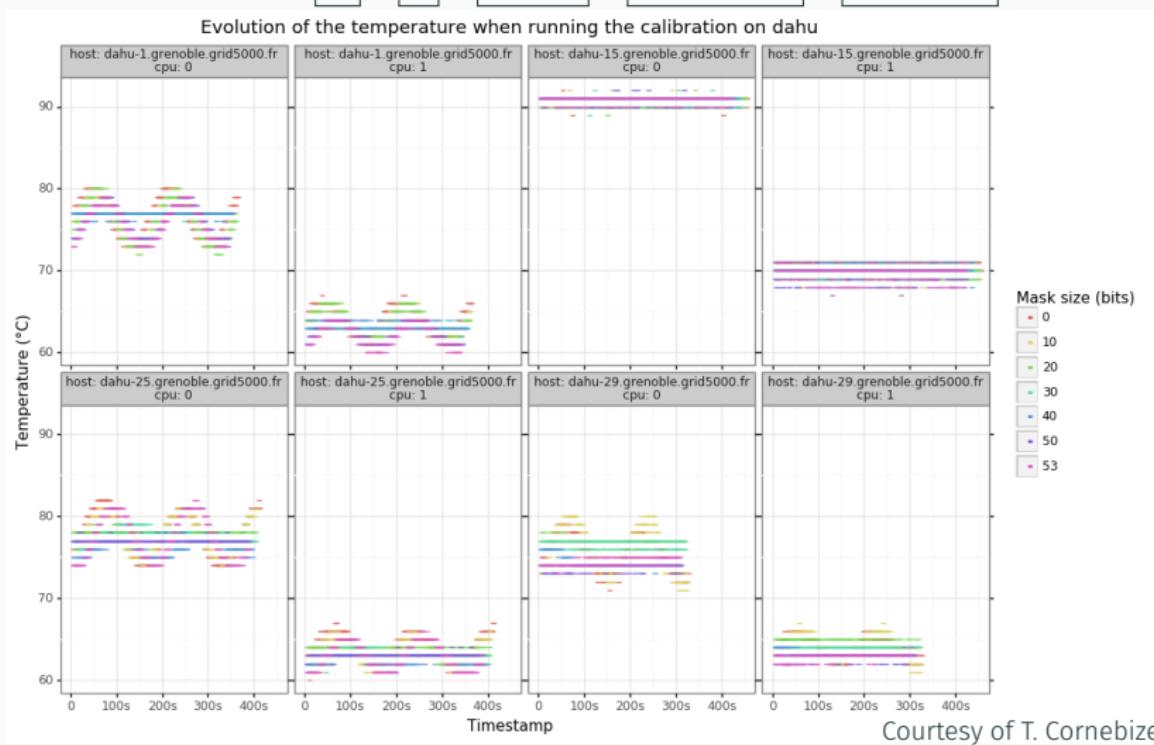
- $C = A \times A$ (2048×2048), independant • Time scale = 5 minutes
 - A initialized with 0 1 0.987 1, 2, 3, ... random ?



ON THE IMPORTANCE OF CONTENT INITIALIZATION

- $C = A \times A$ (2048×2048), independant
- A initialized with

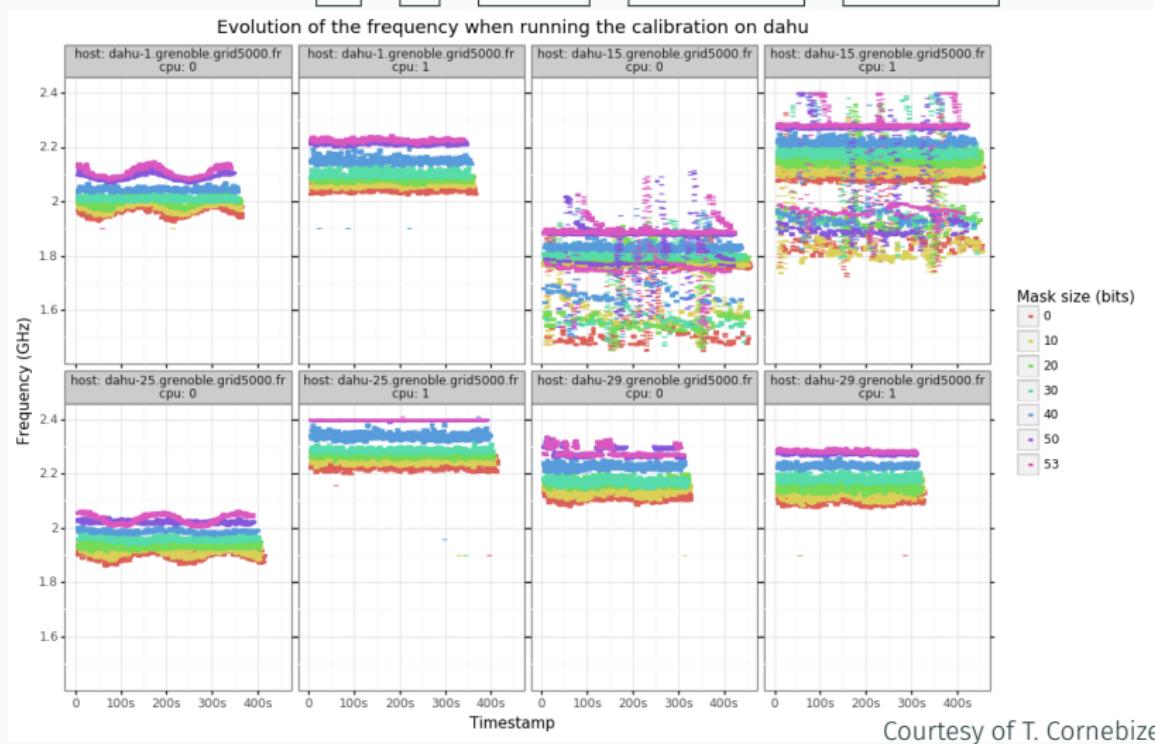
0	1	0.987	1, 2, 3, ...	random ?
---	---	-------	--------------	----------
- Time scale = 5 minutes



ON THE IMPORTANCE OF CONTENT INITIALIZATION

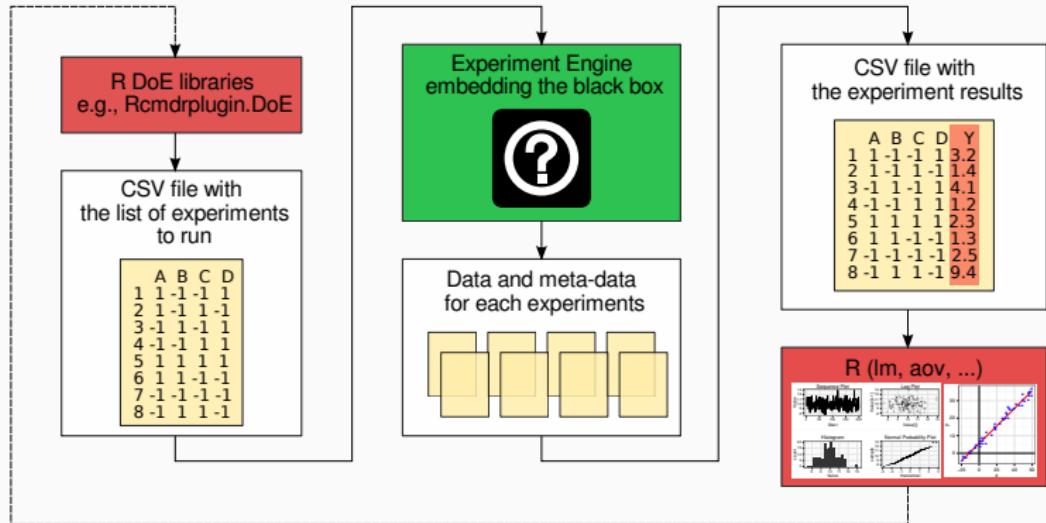
- $C = A \times A$ (2048×2048), independant
- A initialized with

0	1	0.987	1, 2, 3, ...	random ?
---	---	-------	--------------	----------
- Time scale = 5 minutes



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 Instantiate 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
 - Require data, code, environment, XP protocol, ...
3. **Incentive:** consider RR/open science when hiring/promoting

REPRODUCIBLE RESEARCH = RIGOR AND TRANSPARENCY

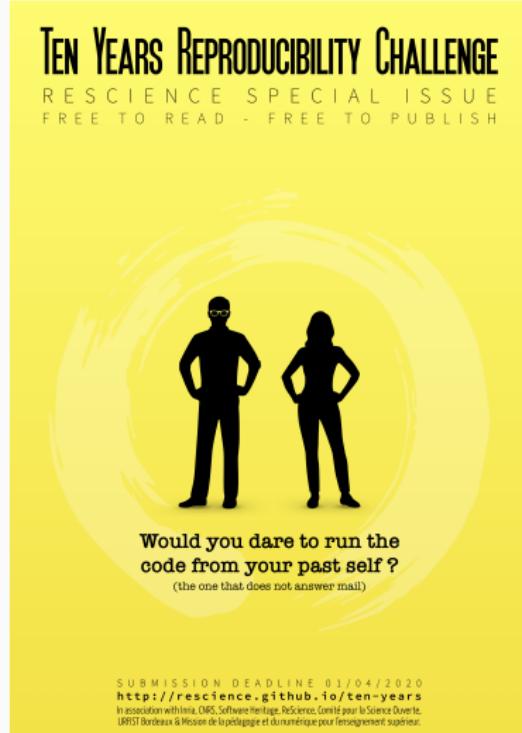
To err is human.

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
 - Require data, code, environment, XP protocol, ...
3. **Incentive:** consider RR/open science when hiring/promoting
4. **Prepare the Future:** How to share Experiments?
 - Reuse, reuse, reuse!
 - Shared and controled testbeds (e.g., Grid'5000/SILECS)
 - Toward **literate experimentation?**



SOME ADVERTISING



TEN YEARS REPRODUCIBILITY CHALLENGE
RESCIENCE SPECIAL ISSUE
FREE TO READ - FREE TO PUBLISH

Would you dare to run the code from your past self?
(the one that does not answer mail)

SUBMISSION DEADLINE 01/04/2020
<http://rescience.github.io/ten-years>
In association with Inria, CNRS, Software Heritage, Rescience, Comité pour la Science Ouverte, DFRST Bordeaux & Mission de la pédagogie et du numérique pour l'enseignement supérieur.

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



3rd Edition: ≈ Feb. 2020

A new MOOC: "Advanced R"

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

October 2020 ?