Parallélisme et reproductibilité de modèles stochastiques

Donner plusieurs chances au hasard ?

David HILL

CURRENT ACM DEFINITIONS

https://www.acm.org/ publications/policies/artifact-review-badging

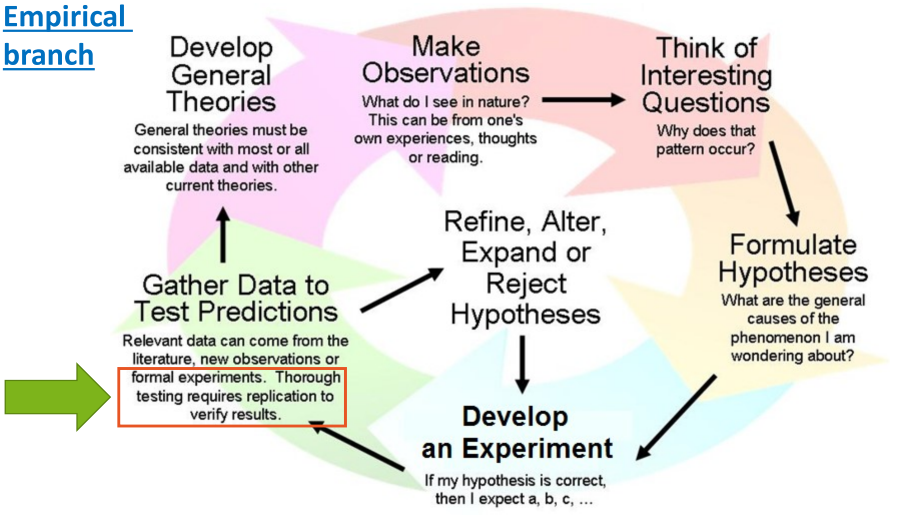
1. **Repeatability** : For the current ACM definition, repeatability implies that a computer scientist can repeat calculation and should find each time the same result **with a stated precision** (the same team and the same experimental setup) Stated precision = 0 > bitwise identical results
2. **Reproducibility** : The latest ACM definition for reproducibility, says that a person or a group of researchers independent from the initial author (or group of authors) is able to obtain, with a stated precision, the same result with thanks to the initial author's artifact (a different team trying to obtain the same results using the same experimental setup).
3. **Replicability** : This term replicability is considered as rather new for many computer scientists (and dictionaries…). It means that a new team should obtain the same result, with a stated precision, using artifacts which they develop completely independently (a different team working with a different experimental setup). Strong corroboration – best choice for Epistemology.

The notion of **"same results" remains vague**. With a stated precision it fits with the requirements for measurements which inspired ACM, **but it fails to meet the debugging requirements, essential for software development (bitwise identical results are needed).**

THE SCIENTIFIC METHOD

https://i0.wp.com/peegel.info/wp-content/uploads/2017/10/scientific_method.png

Empirical branch



Traditionally we have 2 main branches of the scientific method:
1 – Deductive branch Mathematics and formal logic
2 – Empirical branch Statistical analysis of controlled experiments.

There is a hope for a 3rd & 4th branches
3 – Large Scale Simulation
4 – Data intensive & data driven computer Science

But we do not meet the standards of Branch 1 & 2…

WHY DO WE HAVE PROBLEMS WITH COMPUTERS?

Here are some Technical Reasons for HPC numerical repeatability FAILURES

…

In addition to possible Individual Errors and Misconducts…

Floating point…

- Round off errors
- Order of floating point operations (dynamic execution / out of order)
- …

Hardware (failures of hardware change)

- Number of processors, Networking Interconnect, devices and latency
- Difference between architectures (regular processors, vs accelerators,…) – Hybrid computing.
- Processor implementation or design bugs
- Silent/soft errors
- …

Software

- Operating systems, compilers,
- Libraries, dependencies and software stack versions
- Parallelization techniques
- Virtual machines and containers (rare in HPC > bare metal)
- …

Example of Hardware Error a Few Years Ago and miss-behaviors > Hyper-Threading Bugs, Meltdown,…

[WARNING] Intel Skylake/Kaby Lake processors: broken hyper ...
https://lists.debian.org/debian-devel/2017/06/msg00308.html ▾ Traduire cette page
25 juin 2017 - TL;DR: unfixed **Skylake** and **Kaby Lake** processors could, in some situations, dangerously misbehave when **hyper-threading** is enabled. Disable **hyper-threading** immediately in BIOS/UEFI to work around the **problem**. Read this advisory for instructions about an Intel-provided fix. SO, WHAT IS THIS ALL ...

```
Users of systems with Intel Skylake processors may have two choices:

1. If your processor model (listed in /proc/cpuinfo) is 78 or 94, and
   the stepping is 3, install the non-free "intel-microcode" package
   with base version 3.20170511.1, and reboot the system.  THIS IS
   THE RECOMMENDED SOLUTION FOR THESE SYSTEMS, AS IT FIXES OTHER
   PROCESSOR ISSUES AS WELL.
```

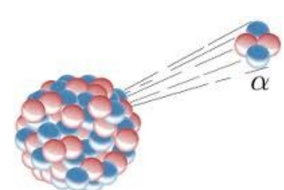
Skylake and Kaby Lake CPUs have broken hyper-threading - Fudzilla
https://www.fudzilla.com/.../43964-skylake-and-kaby-lake-cpus-ha... ▾ Traduire cette page
26 juin 2017 - During April and May, Intel started updating processor documentation with a new errata note and it turned out that the reason was that **Skylake** and **Kaby Lake** silicon has a microcode **bug** it did not want any one to find out about. The errata is described in detail on the Debian mailing list, and affects **Skylake** ...

Why do we (ALSO) need REPEATABILITY ?

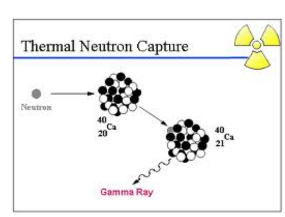
- If you don't have repeatability, **how do you debug ?**
  And how do we repeat/reproduce the events observed in simulations ?
  (confirmation of Higgs discovery, etc…)
- In Digital Computer Science we are used to **deterministic computing** and we expect « **repeatability** » - it was "granted" for many years.
  Computer debugging and program setup is based on repeatability!
- **Even when we use pseudo-random numbers** for stochastic models, **we are running deterministic experiments since pseudo-random number generators have been carefully designed to be repeatable** (though some computer scientist often use the "reproducible" term…).

RELIABILITY & HPC AT SCALE… MORE FREQUENT SILENT ERRORS (A.K.A. SOFT ERRORS…)

1. Change the system state by 'external forces'
   - Alpha particles
   - Cosmic rays (High Energy Particles from space)
   - Thermal neutrons
   - Variation in voltage, temperature, etc.



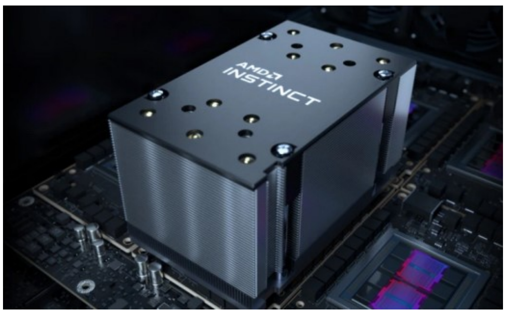
2. They are at the origin of ECC…to avoids bits flips in memory cells
   - There is also a **rising of soft errors in arithmetic units !!!**
   - **The more we size down the more this problem increases.**
   - Chip manufacturers spend money and silicon space to avoid this kind of errors:
     - Samsung, GlobalFoundries, and IBM introduced the world's first **5nm** chip with GAAFET transistors, GAA (gate-all-around) stacked nano-sheet transistors.



3. Soft errors are difficult to detect **and almost impossible to reproduce**
   Using spare time of Supercomputers to check ? Use of Fault injection framework…

8

2002 – 2022: A SHORT IMAGE OF RECENT PROGRESS



AMD Instinct™ MI250X

Compute Units: 220

Peak Single Precision Matrix (FP32) Performance:
95.7 TFLOPs

Peak Double Precision (FP64) Performance:
47.9 TFLOPs

Peak bfloat16: 383 TFLOPs

NEC Earth Simulator 1
32 TF in 2002,
40 TF in 2005 (DP).
Ranked #1
for 5 Top 500 contests.
Full building in 2002

2022 a single accelerator board : 47.9 TF in DP
Impressive "size down"

9

REMEMBER ECC (ERROR CORRECTION CODE)
THIS TECHNIQUE SOLVES INEVITABLE BIT ERRORS IN RAM

1. How often do silent errors impact our RAM ?
   - Bit errors occur about **once a week** in 4GB RAM due to the background radiation.
   - From 2% to 15% of these errors lead to faulty calculations, system crashes or unpredictable behavior.
   - On a computer without ECC there's one serious incident in the computer system every year, at the lowest estimate.
   - For our classical compute servers with 512 GB : **128 errors per week**
   - For our top server with 3 TB : **768 errors per week**
   - Our old SG UV Brain 12 TB server : **3072 errors / week (18+ / hour)**
   - **On Super Computers, the MTBF is below one day.**
2. What can we check and correct :
   - An ECC-capable memory controller can generally **detect and correct errors of a single bit per word** (the unit of the memory bus),
   - It can detect (but not correct) errors of **two bits per word**.
3. For scientific applications **avoid 'playing' with GPUs or accelerators without ECC**

NVIDIA. Control Panel

Change ECC State

The Change ECC State page lets you:

- Change the Error Correction Code (ECC) state for GPUs.
- View GPU memory details.

Set the ECC state for your GPUs

This table shows the ECC state for each GPU that supports ECC.

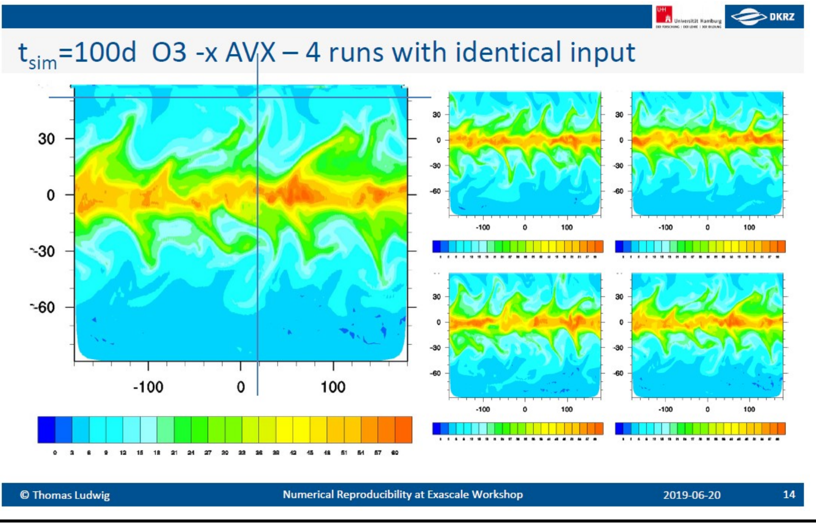
10

FRONTIER SUPERCOMPUTER SUFFERING 'DAILY HARDWARE FAILURES' NOT ONLY DURING TESTING

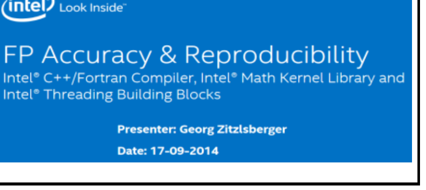
- Oak Ridge National Laboratory's (ORNL) exascale Frontier supercomputer is seeing daily hardware failures during its testing phase.
- "Being exascale ain't easy", ORNL's Justin Whitt say teething troubles are normal.
- It's mostly issues of scale & coupling with AMD accelerator when faced to the breadth of applications. The issues we're encountering mostly relate to running very, very large jobs using the entire system … and getting all the hardware to work in concert to do that,"
- **A day-long run without a system failure "would be outstanding**". "Our goal is still hours" but longer than Frontier's current failure rate, adding that "we're not super far off our goal.

https://www.datacenterdynamics.com/en/news/frontier-supercomputer-suffering-daily-hardware-failures-during-testing/

EXAMPLE OF RUN TO RUN REPEATABILITY ERRORS (SOFTWARE ISSUES)



From Prof. Dr. T. Ludwig – DKRZ Director - ISC Supercomputing Frankfurt – June 2019



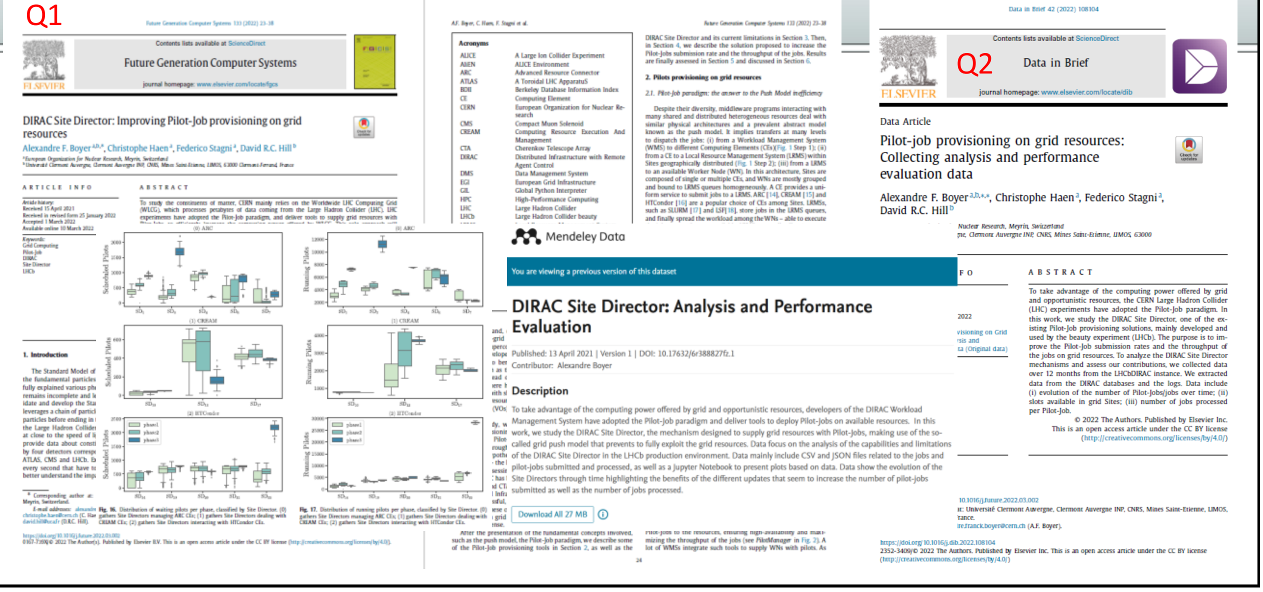
DUMMY CHEK OF YOUR COMPUTING CLUSTER
(DON'T DO THIS –WE HAVE ECC !)

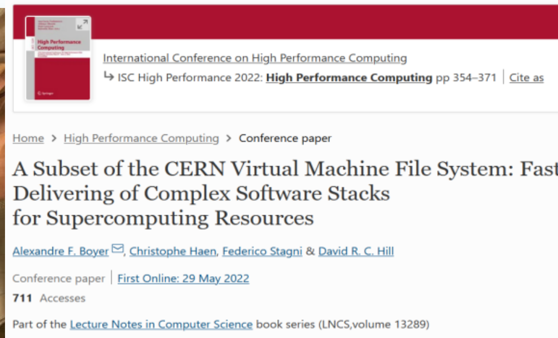
```
#pragma omp parallel ...

void funnyCosmicRayDetector()
{
   // static const used to emphasize that
   // this variable should keep its value
   static const int shouldNotChange = TRUE;

   while(shouldNotChange)
   {
      // if (!shouldNotChange) for nerds
      if (shouldNotChange == FALSE) exit(COSMIC_RAY_DETECTED);
   }
}
```

For babies ?!

13

WHAT CAN WE DO WHEN THE HPC HARDWARE IS SPECIFIC ?




International Conference on High Performance Computing
↳ ISC High Performance 2022: **High Performance Computing** pp 354–371 | Cite as

Home > High Performance Computing > Conference paper

A Subset of the CERN Virtual Machine File System: Fast Delivering of Complex Software Stacks for Supercomputing Resources

Alexandre F. Boyer, Christophe Haen, Federico Stagni & David R. C. Hill

Conference paper | First Online: 29 May 2022

711 Accesses

Part of the Lecture Notes in Computer Science book series (LNCS,volume 13289)

Reduce complexity : Generic utility to help any community install complex software dependencies on supercomputers without external connectivity.
(multi-million computing facilities 5 to 12 MW)

Experimentation on "Mare Nostrum", a Top 500 supercomputer (#5 – 2006 – Now MN5 314 PF)
We provide steps to encapsulate the minimum required files and provide a lightweight, easily updated subset of essential dependencies (12.4 GB instead of 5.2 TB for the entire LHCb repository) to facilitate portability and reproducibility) – Size Optimization : 99,8 % !!!

Part II Applications Reproducibility Issues

(A) A Use Case Quantum Dissipative Dynamics (QDD) Reproducibility and Performance

	Gfortran	ifort	ifx
	Average time (in s.) gfortran	**Average time (in s.) ifort**	**Average time (in s.) ifx**
About 13% more efficient	**969,48**	1296,91	1597,03

FFTW	MKL	
Average time (in s.) FFTW	**Average time (in s.) MKL**	
1373,22	**1195,72**	**About 13% more efficient**

DINH P.M., VINCENDON M., COPPENS F., SURAUD E., REINHARD P.G., "Quantum Dissipative Dynamics (QDD): A real-time real-space approach to far-off-equilibrium dynamics in finite electron systems". Computer Physics Communications, 2022, vol. 270, p. 108155.

Simulation of electron dynamics under the influence of external electromagnetic fields


Computer Physics Communications
Volume 270, January 2022, 108155

Quantum Dissipative Dynamics (QDD): A real-time real-space approach to far-off-equilibrium dynamics in finite electron systems ☆, ☆☆

P.M. Dinh, M. Vincendon, F. Coppens, E. Suraud, P.-G. Reinhard

Using the default settings provided, we obtain superior performance of the **gfortran** compiler (vs ifort and ifx) as well as the **MKL** library for Fourier transform (vs FFTW)

17

Reproducibility and Performance (2/5): (1) A Library Found Non-Repeatable

Repeatability issues:
FFTW
The Fastest Fast Fourrier Transform of the WEST!
A well known library

*File1 : 0.25529373*10⁻⁸ 0.83551334*10⁻⁹ 0.83553563*10⁻⁹*

*File2 : 0.25529376*10⁻⁸ 0.83551386*10⁻⁹ 0.83547498*10⁻⁹*

Output files
./gfortran-FFTW-OMP-DYN-Debug/repli1/pdip.Na2-egs
and ./gfortran-FFTW-OMP-DYN-Debug/repli2/pdip.Na2-egs

are different for the same execution conditions (RUN to RUN)

The differences are relatively minor, 4th to 7th decimal places on values from 10^{-8} to 10^{-9}.
Can be **(1) problematic for debugging** and **(2) since the code use non linear equations it might contain chaotic regimes** which can depend on small variations.

FFTW documentation "*If you use FFTW_MEASURE or FFTW_PATIENT mode, then the algorithm FFTW employs is not deterministic: it depends on runtime performance measurements. This will cause the results to vary slightly from run to run.* (https://www.fftw.org/faq/section3.html#nondeterministic)."

Making FFTW repeatable **see an average performance drop of about 17%**

18

(2) Software Dev. Nightmare - Different results between normal execution and debug mode – Again How Do You Debug ?!

1 Option: decrease the default level of optimization (gfortran et ifort)

Table 7. compiler flags that are based on the compiler optimization options listed in Table 5. The first compiler flag is the strictest one (which limits compiler optimizations most significantly), while every other compiler flag is derived from the first one through changing only one compiler optimization option.

No.	Compiler flag
1	-ffloat-store -fno-unsafe-math-optimizations -fno-associative-math -fno-reciprocal-math -fno-finite-math-only -fno-rounding-math -fno-cx-limited-range
2	-fno-unsafe-math-optimizations -fno-associative-math -fno-reciprocal-math -fno-finite-math-only -fno-rounding-math -fno-cx-limited-range
3	-ffloat-store -funsafe-math-optimizations -fno-associative-math -fno-reciprocal-math -fno-finite-math-only -fno-rounding-math -fno-cx-limited-range
4	-ffloat-store -fno-unsafe-math-optimizations -fassociative-math -fno-reciprocal-math -fno-finite-math-only -fno-rounding-math -fno-cx-limited-range
5	-ffloat-store -fno-unsafe-math-optimizations -fno-associative-math -freciprocal-math -fno-finite-math-only -fno-rounding-math -fno-cx-limited-range
6	-ffloat-store -fno-unsafe-math-optimizations -fno-associative-math -fno-reciprocal-math -ffinite-math-only -fno-rounding-math -fno-cx-limited-range
7	-ffloat-store -fno-unsafe-math-optimizations -fno-associative-math -fno-reciprocal-math -fno-finite-math-only -frounding-math -fno-cx-limited-range
8	-ffloat-store -fno-unsafe-math-optimizations -fno-associative-math -fno-reciprocal-math -fno-finite-math-only -fno-rounding-math -fcx-limited-range

Optimized performances (non repetable) →

	FFTW Average time (s)	MKL Average time (s)
Optimized	1373,22	1195,72
New	2242,23	1841,84

New performances
1.54 to 1.63 times slower

Source : LI, R., LIU, L., YANG, G., et al. **Bitwise identical compiling setup: prospective for reproducibility and reliability of Earth system modeling**. Geoscientific Model Development, 2016, vol. 9, no 2, p. 731-748.

Table 6. Intel compiler flags that are based on the compiler optimization options given in Table 3. The first compiler flag is the strictest one (which limits compiler optimizations most significantly), while every other compiler flag is derived from the first one through changing only one compiler optimization option.

No.	Compiler flag
1	-fp-model strict -fp-speculation=strict -mp1 -no-vec -no-simd
2	-fp-model precise -fp-speculation=strict -mp1 -no-vec -no-simd
3	-fp-model fast -fp-speculation=strict -mp1 -no-vec -no-simd
4	-fp-model source -fp-speculation=strict -mp1 -no-vec -no-simd
5	-fp-model strict -fp-speculation=safe -mp1 -no-vec -no-simd
6	-fp-model strict -fp-speculation=fast -mp1 -no-vec -no-simd
7	-fp-model strict -fp-speculation=strict -no-vec -no-simd
8	-fp-model strict -fp-speculation=strict -mp1 -vec -no-simd
9	-fp-model strict -fp-speculation=strict -mp1 -no-vec -simd

Comparison with the times related to the basic optimized options provided in the QDD package:

	gfortran Average time (s.)	ifort Average time (s.)	Ifx Average time (s.)
Optimized	969,48	1296,91	1597,03
New	2105,79	1950,59	2069,72

1.29 to 2.17 times slower

Colleagues obtained bitwise repeatable results between the CPU and GPU versions of QDD ! (J. Héraud)

CPU and GPU versions of QDD are both compiled with nvfortran

- Degrade the optimization of the executable code by changing the compilation options.
- We enforce compliance with the IEEE754 standard for floating-point operations.
- Disable hardware optimization to ensure strict use of physical resources (nofma, -O0 = no extended register use, no vectorization !!!)
- Decide the way to perform a sum on the CPU so that it respects the same order of calculations (on both CPU and GPU.
- Implement a math library to use the algorithms common in this program for both the GPU and the CPU.
  - We implemented the EXP, ERF, SIN, COS, DIVISION (for complex numbers), SQRT, and CBRT functions.
- Implementation of an FFT (Cooley-Tukey algorithm) common to both GPU and CPU.
- To enable reproducible code, a preprocessor variable is added at compile time.

Price to pay: the repeatable versions on GPU are 20 times slower…

(B) PARALLEL STOCHASTIC SIMULATIONS CAN BE REPRODUCIBLE

Most Parallel Monte Carlo Simulations are often easy to parallelize.

- Particularly when they fit with the **independent bag-of-work** paradigm.
- San easily tolerate a loss of jobs if enough jobs finish for the final statistics…
- **Requirements**:

  (1) A fine **repeatable** Generator, (2) a fine Parallelization technique and (3) "independent" Parallel random streams for all stochastic objects (4) Then it is possible to compare Sequential and Parallel results at small scales
- Should fit with **different distributed computing platforms / HPC nodes**
  - Using regular processors
  - Using accelerators : GP-GPUs, Intel IGP/GPU Xe, (Old X.Phi - FPGAs ?)

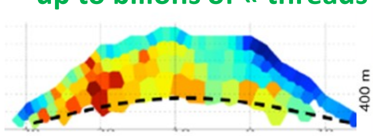
HILL D. PASSERAT-PALMBACH J. MAZEL C., TRAORE, M.K., "Distribution of Random Streams for Simulation Practitioners", Concurrency and Computation: Practice and Experience, June 2013, Vol. 25, Issue 10, pp. 1427-1442.
Hill D., "Parallel Random Numbers, Simulation and reproducibility". IEEE/AIP - Computing in Science and Engineering, vol. 17, no 4, 2015, pp. 66-71.

(B.1) - TOMUVOL PROJECT

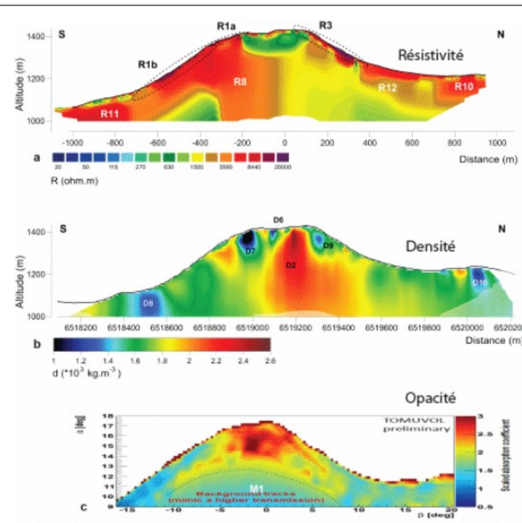
http://wwwobs.univ-bpclermont.fr/tomuvol/presentation.php

LMV (Laboratoire Magmas et Volcans) and LPC (Laboratoire de Physique Corpusculaire) made a joint venture with computer scientists for this TOMUVOL project (TOmographie MUonique des VOLcans) – with with C. Carloganu and supervision of P. Schweitzer thesis in our LIMOS CNRS Laboratory

Reproducible parallel stochastic simulations - up to billons of « threads »



Density contrast
14 days of data taking, 0.66 m² x 1 m



Bitwise Reproducibility
Study on 2 different Architectures (x86 vs k1Om)

As announced by Intel we cannot expect bit for bit reproducibility when working with such different architectures - in our case (x86 & k1Om).

- However with the best compiler flags, **we observed bit for bit repeatability in single precision** but **not in double precision where we have little differences (reproducibility).**
- The relative difference between processors (E5 vs Phi) in double precision were analyzed and are shown here >

Relative CPU-Phi differences between the results and number of altered bits

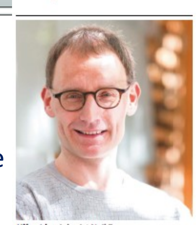
Difference ↓ \ Result →	Position X	Position Z	Direction X	Direction Y	Direction Z
0 bit: bit for bit reproducibility	4922	4934	4896	4975	4913
1 bit: 1.11E-16 ≤ Δ < 2.22E-16	25	21	14	5	18
2 bits: 2.22E-16 ≤ Δ < 4.44E-16	21	18	52	4	31
3 bits: 4.44E-16 ≤ Δ < 8.88E-16	15	12	23	6	12
4 bits: 8.88E-16 ≤ Δ < 1.78E-15	10	7	5	4	10
≥ 5 bits: 1.78E-15 ≤ Δ < 2.25E-11	7	8	10	6	16

Run-to-Run Reproducibility of Floating-Point Calculations for Applications on Intel® Xeon Phi™ Coprocessors (and Intel® Xeon® Processors) – by Martin Cordel - https://software.intel.com/en-us/articles/run-to-run-reproducibility-of-floating-point-calculations-for-applications-on-intel-xeon

See also P. Schweitzer thesis & paper : SCHWEITZER P., CIPIÈRE S., DUFAURE A., PAYNO H., PERROT Y., HILL D. and MAIGNE L., "Performance evaluation of multi-threaded Geant4 simulations using an Intel Xeon Phi cluster", Scientific Programming, Article ID 980752, 10 pages, 2015. doi:10.1155/2015/980752.

(B.2) COVID 19
Annoying Reproducibility Problems...

	Toutes	Depuis 2017
Citations	56768	33528
indice h	96	76
indice i10	259	198



UK epidemiologist Neil Ferguson.

- Mid-March 2020: Neil Ferguson and his teams at Imperial college release projections estimating that the UK could face up to 500,000 COVID-19 death
- May 2020 : the American magazine "National Review" declares that the model used by Neil Ferguson is inaccurate and intended to be used for pandemic influenza rather than coronavirus (estimated using the classical equation-based approach based on the SEIR model).
https://www.nationalreview.com/corner/professor-lockdown-modeler-resigns-in-disgrace/
- Ferguson refuses to publish his original code.
Other scientists claimed they could not verify his results.
After a six-week delay, he released a heavily revised code.
- Experts deem the code (written in C++) to be "completely unreliable".
On Github, a collective of engineers started a petition to ask all articles that relied on this code to be removed. https://github.com/mrc-ide/covid-sim
- In addition, **scientists from the University of Edinburgh claimed that it was impossible to reproduce the same results from the same data using the model**
- Another case: two research papers against chloroquine retracted in Top journals

Covid-19 et The Lancet : le co-auteur a demandé la rétractation répond à Sciences


Par Nicolas Gutierrez C. le 05.06.2020 à 19h59

Le 4 juin 2020, trois des auteurs de l'article publié le 22 mai dans le journal médical The Lancet ont demandé la rétractation de leur étude, qui associait la chloroquine a une surmortalité chez les patients atteints de Covid-19. Réactions à cet épilogue inattendu.

Hydroxychloroquine or chloroquine with or without a macrolide for treatment of COVID-19: a multinational registry analysis

Some top PRNGs (Pseudo Random Number Generators) for REPEATABLE Parallel Stochastic Simulations

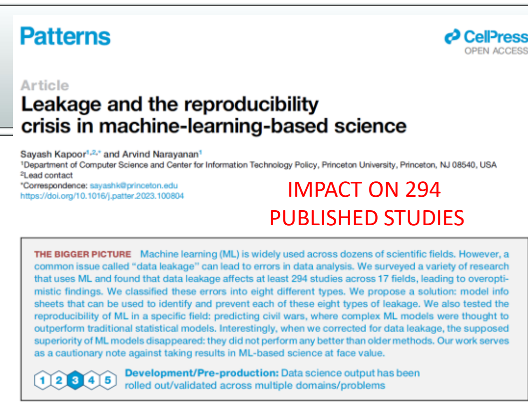
Green PRNGs are said 'crush' resistant (TestU01 software) and can be recommended:

- MRG (Multiple Recursive Generator) – **slow but top API for reproducing parallel simulations**
  $x_i = (a_1 * x_{i-1} + a_2 * x_{i-2} + \ldots + a_k * x_{i-k} + c) \bmod m$ – with k>1
  Ex: **MRG32k3a & MRG32kp** – by L'Ecuyer and Panneton
- MLFG (Multiple Lagged Fibonacci Generator) – Non linear
  by Michael Mascagni MLFG 6331_64
- **Mersenne Twisters** – by Matsumoto, Nishimura, Saito (MT, SFMT, MTGP, TinyMT…)
- **WELLs generators by** – Panneton, L'Ecuyer and Matsumoto
- **1,2,3… Parallel Philox and Threefry –** by Salmon et al. presented at SC'11 with crypto background and a parameterization technique. In his master's thesis, Liang Li (Prof. Mascagni's student couldn't reproduce these tests. We had the same problem with Philox4x32-10.

PCG and Xoshiro modern generators are said to be very fast and 'Crush' resistant and this is not always the case in 1/3 of the streams we have tested (and MT is faster un double precision).

HILL D. PASSERAT-PALMBACH J. MAZEL C., TRAORE, M.K., "Distribution of Random Streams for Simulation Practitioners", Concurrency and Computation: Practice and Experience, June 2013, Vol. 25, Issue 10, pp. 1427-1442. 25

(C) Machine Learning and Reproducibility Issues


Patterns
CellPress OPEN ACCESS

Article
Leakage and the reproducibility crisis in machine-learning-based science

Sayash Kapoor[1,2,*] and Arvind Narayanan[1]
[1]Department of Computer Science and Center for Information Technology Policy, Princeton University, Princeton, NJ 08540, USA
[2]Lead contact
*Correspondence: sayashk@princeton.edu
https://doi.org/10.1016/j.patter.2023.100804

IMPACT ON 294 PUBLISHED STUDIES

THE BIGGER PICTURE Machine learning (ML) is widely used across dozens of scientific fields. However, a common issue called "data leakage" can lead to errors in data analysis. We surveyed a variety of research that uses ML and found that data leakage affects at least 294 studies across 17 fields, leading to overoptimistic findings. We classified these errors into eight different types. We propose a solution: model info sheets that can be used to identify and prevent each of these eight types of leakage. We also tested the reproducibility of ML in a specific field: predicting civil wars, where complex ML models were thought to outperform traditional statistical models. Interestingly, when we corrected for data leakage, the supposed superiority of ML models disappeared: they did not perform any better than older methods. Our work serves as a cautionary note against taking results in ML-based science at face value.

1 2 3 4 5 **Development/Pre-production:** Data science output has been rolled out/validated across multiple domains/problems

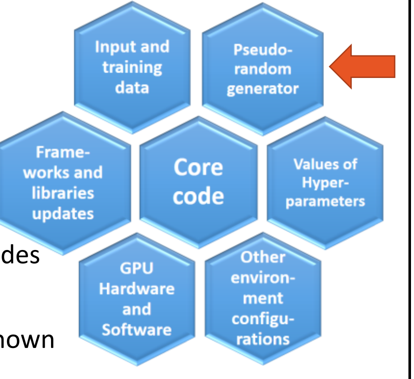
Many factors were found:

- Henderson et al. (2018) and Gundersen et al. (2022) give for instance overviews of reproducibility problems in deep reinforcement learning.
- Kapoor and Narayanan recently discussed the reproducibility crisis in machine learning-based science (2023) because of data leakage.
- Pham et al. (2020) and Zhuang et al. (2022) cover sources of variability in deep learning methods :
  - Initialization of pseudorandom number generators (PRNGs) is one of them.
  - Multi-threading is another, but there are several others.
- Only setting what is commonly named 'seeds' and thread parameters will not be enough to make the result of a neural network deterministic.

Deep Embedded Clustering Test Case For A Medical App Discovery of 3 Different Pseudorandom Number Generators

- Seems strange to test this (all should be initialized). Here we see that the numpy generator is not really used.

Mersenne Twister	PCG	Philox	Results
Not initialized	Not initialized	Not initialized	Non repeatable results
Not initialized	Not initialized	Initialized	Non repeatable results
Not initialized	Initialized	Not initialized	Non repeatable results
Not initialized	Initialized	Initialized	Non repeatable results
Initialized	Not initialized	Not initialized	Non repeatable results
Initialized	Not initialized	Initialized	**Repeatable results**
Initialized	Initialized	Not initialized	Non repeatable results
Initialized	Initialized	Initialized	**Repeatable results**



Sources of Non-Repeatability of Run-To-Run Machine Learning Experiments

- PCG (the default Numpy generator) is recognized as weak for parallel codes in the numpy documention.
- An extension has been proposed: PCG64DXSM but Vigna has recently shown that both are weak and they are slower than other older known PRNGs (URL: https://pcg.di.unimi.it/pcg.php).
- Advice: **Be careful, the seeds are not the states of modern generators**.

GPUs ? Still a Real Challenge For Repeatability !

PyTorch

Reproducibility over multiGPUs is impossible until randomness of threads is controled,

sbelharbi Soufiane Belharbi

Did anyone succeed to reproduce their code when using multiGPUs?
If yes, could you share how you did it? (general idea)

My code is totally reproducible when using one single GPU (independently of the number of workers >= 0); however, it loses its reproducibility when using multiple GPUs.
Randomness on samples (such as transformations) is controlled (fixed using seeds for each sample).
I use `torch.nn.DataParallel` to wrap the model for multiGPUs.

Who to blame for this non-reproducibility in the case of multiGPUs? atomic operations (I hope not)?

Pytorch reproducibility note 27 .
I use Pytorch 1.0.0, Python 3.7.0.

I use standard guide to fix the seeds of the modules:

CUDA-induced randomness
Difference up to 5%
For some Apps it could be significant

arXiv > cs > arXiv:2410.02806

Computer Science > Computer Vision and Pattern Recognition

[Submitted on 19 Sep 2024]

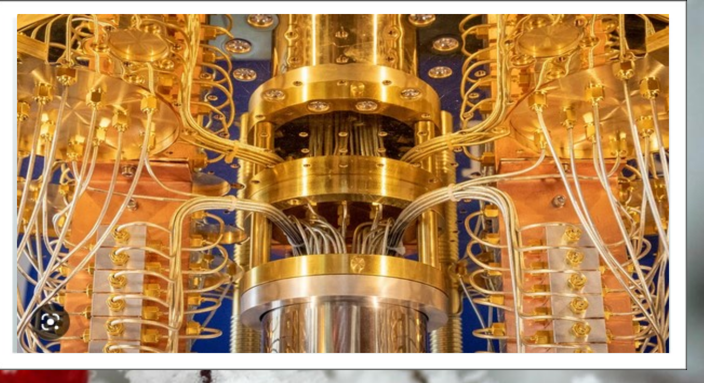
Investigating the Impact of Randomness on Reproducibility in Computer Vision: A Study on Applications in Civil Engineering and Medicine

Bahadır Eryılmaz, Osman Alperen Koraş, Jörg Schlötterer, Christin Seifert

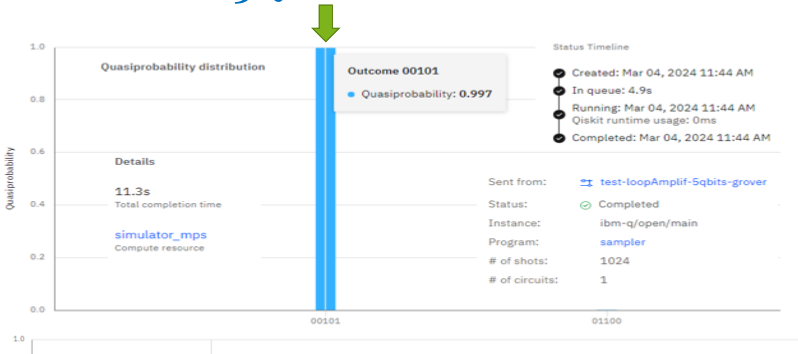
Reproducibility is essential for scientific research. However, in computer vision, achieving consistent results is challenging due to various factors. One influential, yet often unrecognized, factor is CUDA-induced randomness. Despite CUDA's advantages for accelerating algorithm execution on GPUs, if not controlled, its behavior across multiple executions remains non-deterministic. While reproducibility issues in ML being researched, the implications of CUDA-induced randomness in application are yet to be understood. Our investigation focuses on this randomness across one standard benchmark dataset and two real-world datasets in an isolated environment. Our results show that CUDA-induced randomness can account for differences up to 4.77% in performance scores. We find that managing this variability for reproducibility may entail increased runtime or reduce performance, but that disadvantages are not as significant as reported in previous studies.

28

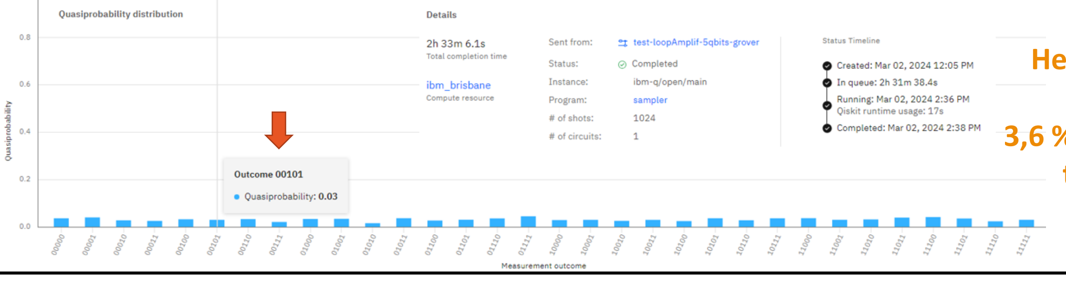
(Q) Towards Reproducible Quantum Computing



Test with the 5 qubits Grover Algorithm with 4 iterations



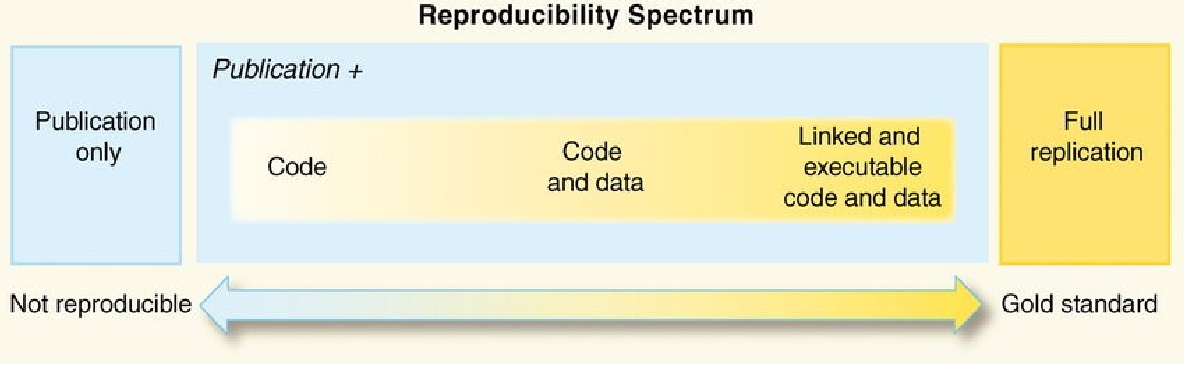
Good and reproducible results on different simulators



Not so good results on real quantum machines …
Here « Kyoto » 127 qubits
3,6 % on January the 1st 2024

Improving Reproducibility Spectrum – Training ! (Really) Wonderful MOOCs Are Available

A must do for my PhD student to improve the quality of our work (let's stop working like we did, not being "aware…")
Level 1: https://www.fun-mooc.fr/en/courses/reproducible-research-methodological-principles-transparent-scie/
Level 2: https://www.fun-mooc.fr/fr/cours/reproducible-research-ii-practices-and-tools-for-managing-comput/
Level 2Bis: coming !

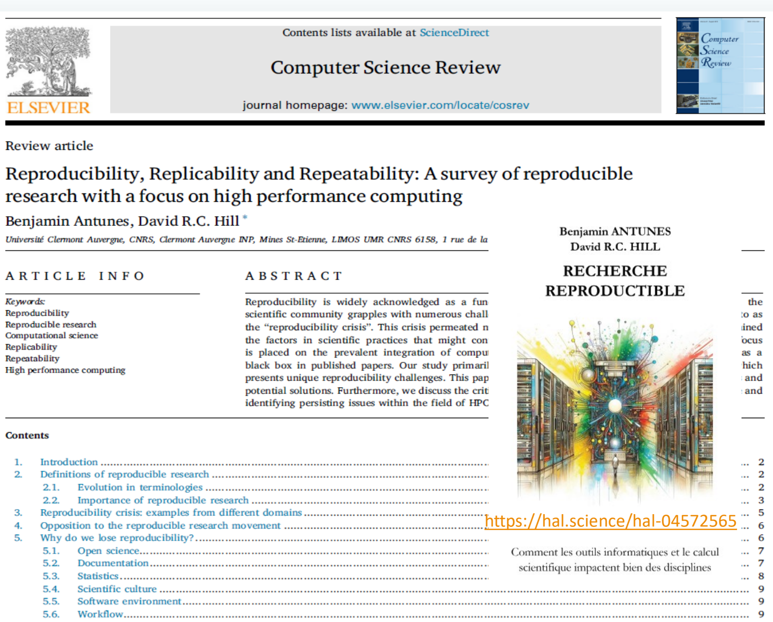


Source : https://www.researchgate.net/figure/Reproducibility-Spectrum_fig2_325910795

Conclusion
HPC can be a BIG amplifier of errors…

- **Huge Numerical differences when we do not pay attention to repeatability & compiler flags**
- **Repeatability achieved for identical execution platforms… even portability ! (sometimes)**
- **Comparison possible between parallel and sequential results !!!**
  – (work at the scale of a node - with the given method)
- **Numerical Reproducibility is possible (not repeatability) for Parallel Stochastic applications with independent computing on different architectures.**
- **Can be resilient to silent errors on supercomputers (use statistics – 'N out of M').**
- Key elements of a method have been presented to produce numerically reproducible results for parallel stochastic simulations **comparable with a sequential implementation** (at the scale of a parallel node before large scaling on bigger systems)
- Numerical replication is important for scientists to verify and setup codes in many sensitive areas, finance, climate, nuclear safety, medicine…

More details…


Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

ELSEVIER

Review article

Reproducibility, Replicability and Repeatability: A survey of reproducible research with a focus on high performance computing

Benjamin Antunes, David R.C. Hill *

Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Mines St-Etienne, LIMOS UMR CNRS 6158, 1 rue de la

ARTICLE INFO

Keywords:
Reproducibility
Reproducible research
Computational science
Replicability
Repeatability
High performance computing

ABSTRACT

Reproducibility is widely acknowledged as a fun… the scientific community grapples with numerous chall… to as the "reproducibility crisis". This crisis permeated n… ined the factors in scientific practices that might con… ocus is placed on the prevalent integration of compu… as a black box in published papers. Our study primaril… hich presents unique reproducibility challenges. This pap… and potential solutions. Furthermore, we discuss the crit… and identifying persisting issues within the field of HPC

Contents

1.	Introduction	2
2.	Definitions of reproducible research	2
2.1.	Evolution in terminologies	2
2.2.	Importance of reproducible research	3
3.	Reproducibility crisis: examples from different domains	5
4.	Opposition to the reproducible research movement	6
5.	Why do we lose reproducibility?	6
5.1.	Open science	7
5.2.	Documentation	7
5.3.	Statistics	8
5.4.	Scientific culture	9
5.5.	Software environment	9
5.6.	Workflow	9
5.7.	Software engineering	10

https://hal.science/hal-04572565

A recent Survey in 2024 Computer Science Review (IF 13.3 – Q1)

https://doi.org/10.1016/j.cosrev.2024.100655

PDF Available Free with HAL
Printed version on demand
Mail @ david.hill@uca.fr

ANTUNES B., HILL D., "Reproducibility, Replicability and Repeatability: A survey of reproducible research with a focus on high performance computing", Computer Science Review, Volume 53, 100655, 2024, 28 p.