Reproductibilité et Répétabilité, Peut-on les négliger en Calcul à Haute Performance?

HILL David – ISIMA/LIMOS UMR CNRS 6158

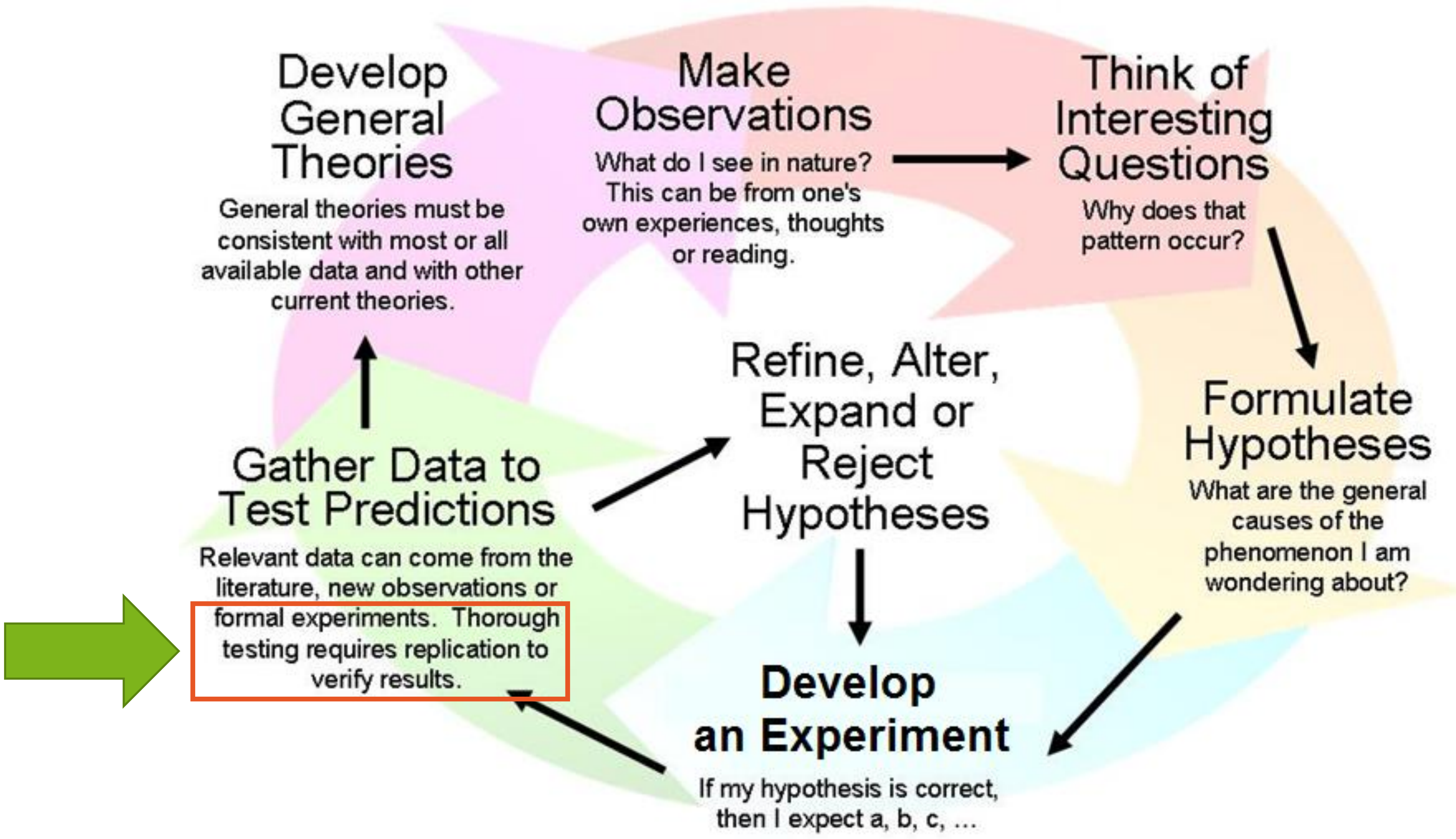
JCAD 2019 – TOULOUSE

1st Let’s Go Back To Philosophy Of Science

Just because we (computer scientists) give our own definitions, which adds confusion - It’s getting cold… isn’t it?

THE SCIENTIFIC METHOD

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



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

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…

REPRODUCIBILITY & CORROBORATION

- Many of us know the important work of Karl Popper (philosopher of sciences) in modeling and simulation. Karl Popper is generally regarded as one of the greatest philosophers of science of the 20th century.

- The criterion of **reproducibility** is one of the conditions on which Popper distinguishes between the scientific or **pseudo-scientific** character of a study.

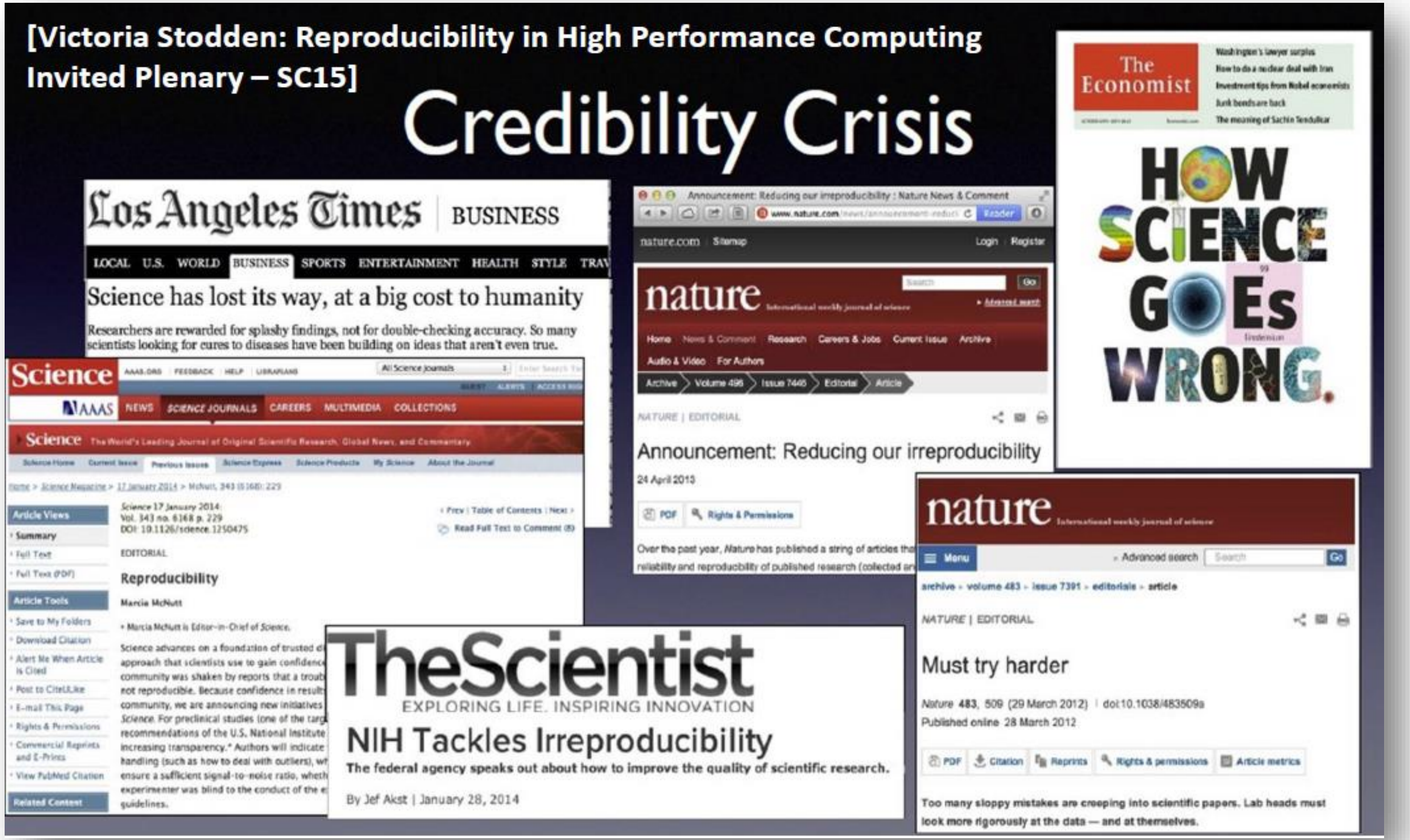
- Scientific conclusions **can only be drawn from a well observed and described "event", which has appeared several times**, observed by different people and/or studies.

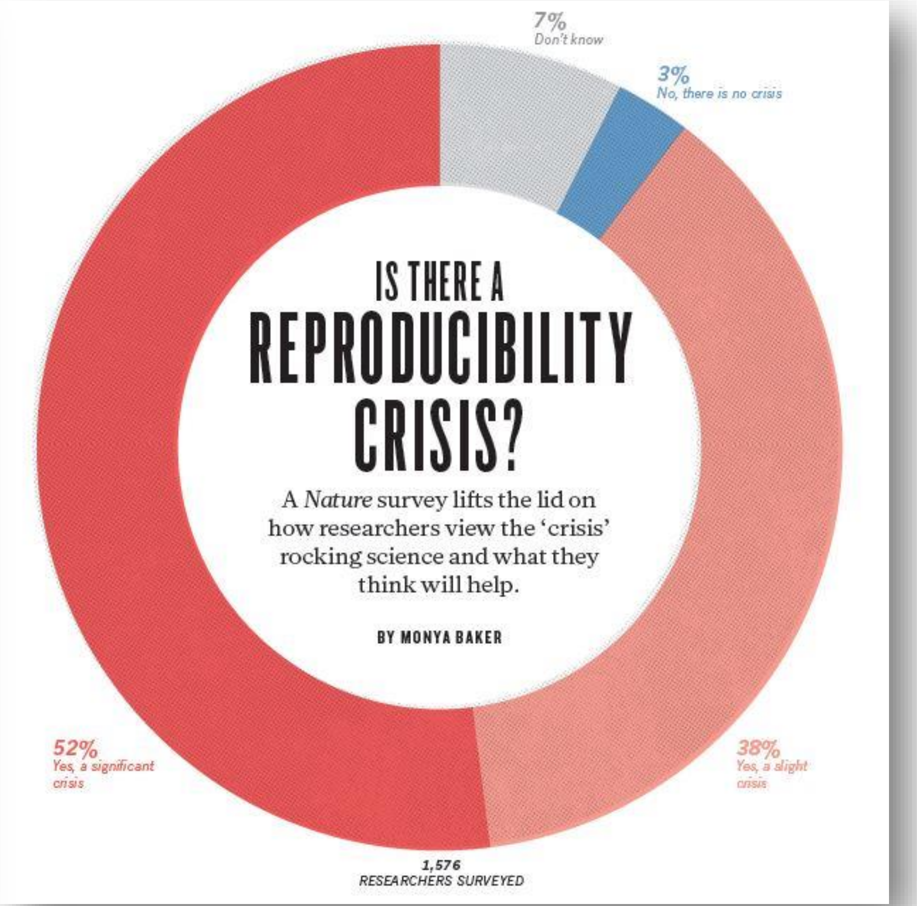
- Science moves forward by **corroboration** when researchers verify/reproduc each other's data. This criterion **eliminates random effects** that distort the results as well as **errors** in judgment or **manipulations** by scientists.

Distinguish between Reproducibility & Repeatability

- There is a growing alarm of results that have been published but **that cannot be reproduced**. **This means waste of time pursuing false leads…**
- A study **of top scientific research** in UK (REF) showed that only 11% of medical studies where reproducible. (First page of “The Guardian”).
- **Reproducibility** (need changes) means observing the same trend, getting the same scientific conclusion (with different infrastructures, methods, experiments…)
- **Repeatability** means you have the same execution trace and the same results (up to bitwise identical results)

Many Domains are Impacted
‘Credibility & Reproducibility Crisis…’





Reproducible Science is Good

But Repeated Computer Science Is Also Needed to DEBUG!

WHY DO WE (ALSO) NEED REPEATABILITY ?

- If you don't have repeatability, **how do you debug** ?
  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** » of computer experiments.
  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…).

- In the context of a Biological or Physical experiment, **repeatability** measures the variation in measurements taken by a single instrument or person under the same conditions, while **reproducibility** measures whether an entire study or experiment can be reproduced in its entirety – by the same research team or by another team.

Floating point...

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

Hardware

- 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)
- ...

Here are some technical reasons for HPC numerical repeatability failures

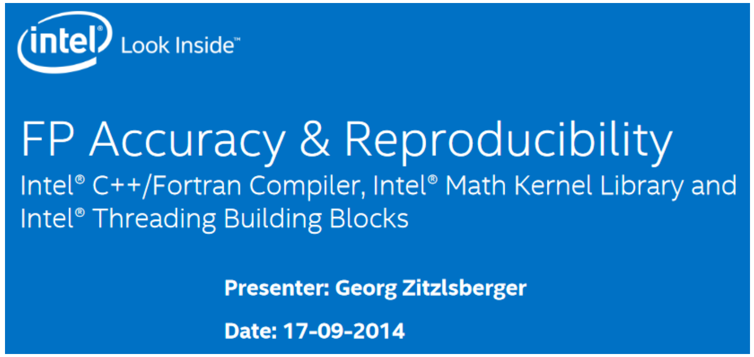
...

In addition to possible individual errors and misconducts...

Zoom in some technical Reasons like « Out of Order Execution » of floating point instructions

See Intel – 2014 https://www.mpcdf.mpg.de/services/computing/software/languages-1/FP_accuracy_reproducibility.pdf

- **Out-of-order execution** is also known as **dynamic execution**.
  Most **modern high-performance microprocessors** optimize the execution of instructions based on the availability of input data to avoid delays.
- **The original order of instructions is now not always respected !**
- The microprocessor avoids having parts of its internal computing units being idle by processing the next instructions which are able to run immediately and “independently”.
- It is the equivalent of the software dynamic recompilation which enables improving instruction scheduling.
- It may impact floating point operations
  floating point arithmetic is not associative (for + & *)
  ex: **a + (b + c) != (a + b) + c**



$$(10^{-3} + 1) - 1 \sim 0$$
$$10^{-3} + (1 - 1) = 10^{-3}$$

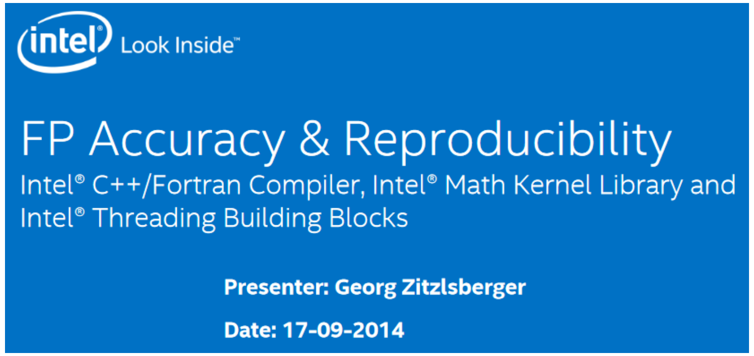
```
>>> (pow(10,-3)+1)-1
0.00099999999999998899
>>> pow(10,-3)+(1-1)
0.001
>>>
```

ANOTHER EXAMPLE OF MICROPROCESSOR DESIGN ERRORS AND MISS-BEHAVIORS > HYPER-THREADING, MELTDOWN, SPECTRE,…

[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 ...

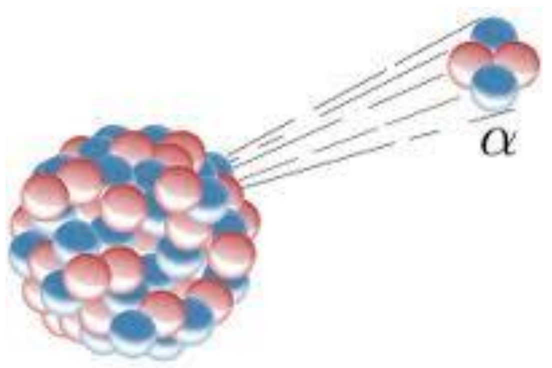
```
Users of systems with Intel Skylake processors may have two choices:

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** ...

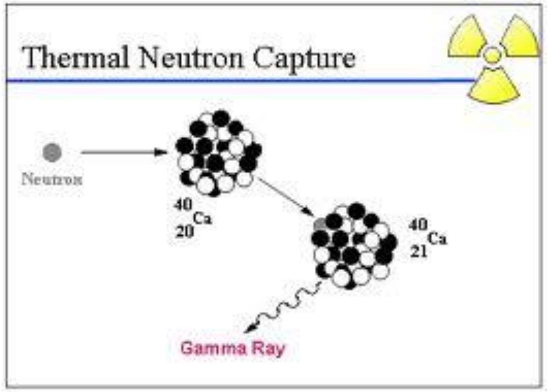


RELIABILITY & HPC… SILENT ERRORS & SOFT ERRORS…

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

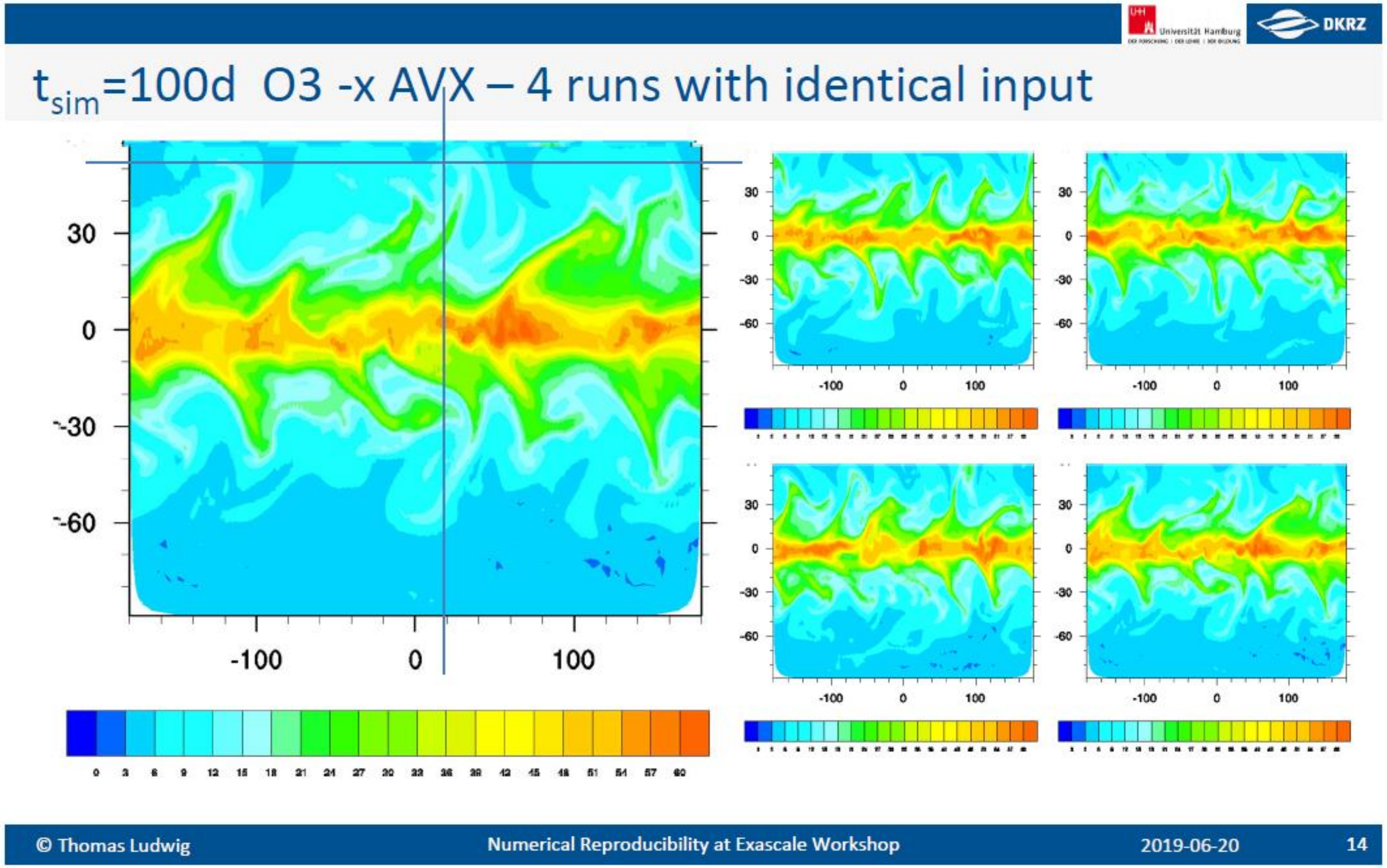


2. They are at the origin of ECC…to avoids bits flips in memory cells
    - There is also a **rising of soft errors in arithmetic units !!!**
    - **The more we size down the more this problem increases.**
    - Chip manufacturers spend money and silicon space to avoid this kind of errors:
        -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



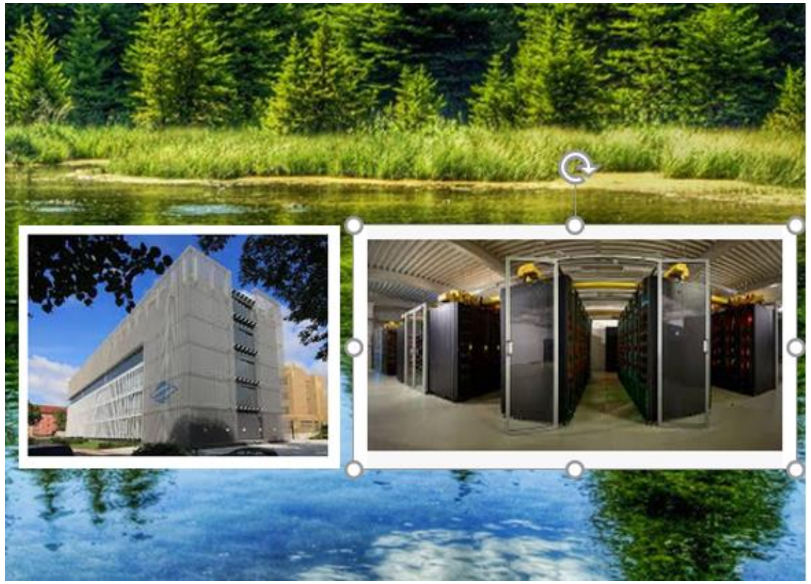
3. Soft errors are difficult to detect **and almost impossible to reproduce**
   Using spare time of Supercomputers to check ? Use of Fault injection framework…

Run to run repeatability errors



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

See also the work of Francois Thomas – Optimization of weather applications on Power and x86 architectures (Toulouse CERFACS)



We Don't Have Easy Solutions – But Tools Are coming...

Workflow Tracking & Resarch Environment :
Sumatra, CoRR (NIST), CDE, Kepler, Chameleon, Galaxy, Tavera, Pegasus, Jupyter notebook, GenePattern,...

Dissemination Platforms:
IPOL, ResearchCompendia.org, Madagascar, MLOSS.org, CoRR (NIST), RunMyCode.org, nanoHUB.org, thedatahub.org, Open Science Framework, Scientific Open Data,...

Embedded publishing :
Sweave, knitR, ReScience, SHARE, Verifiable Computational Research, SOLE, Collage Authoring Environment.

Evolutions of containers like **Singularity** for HPC
Efficient binary containers (ready for ARM processors...)

Towards A Method for Repeatable Parallel Stochastic Simulations

Parallel Stochastic Simulations… Various requirements…

Most Parallel Monte Carlo Simulations are often easy to parallelize.

- Particularly when they fit with the **independent bag-of-work** paradigm.
- Such stochastic simulations can easily tolerate a loss of jobs, if hopefully enough jobs finish for the final statistics…
- **Requirements**:
  - Fine Generator, Fine Parallelization technique and "independent" Parallel random streams.
  - Random statuses should be small and fast to checkpoint at Exascale (Original MT – 6Kb status – MRG32K3a 6 integers)
- Should fit with **different distributed computing platforms / HPC nodes**
  - Using regular processors
  - Using hardware accelerators : GP-GPUs, Intel IGP/GPU Xe, Old Phi, (and FPGAs ?)

Even If We Have No Dependencies Between Elementary Computing, Repeatability Of Parallel Simulation Is Not Granted

A system being of collection of interacting "objects" (dictionary definition)
– a simulation will make all those objects evolve during the simulation time
with a precise modeling goal.

- **To obtain repeatability – think parallel when you design your sequential code** :
  Assign an « independent » pseudo-random stream and initialization status **for each stochastic object of the simulation**.
- An object could also encapsulate a random variate used at some points of the simulation. Every random variate could also have their own random stream with the same approach.
- **This O.O. approach, applied to stochastic objects, is the key to have a reference sequential program that we will be able to compare to a parallel version.**

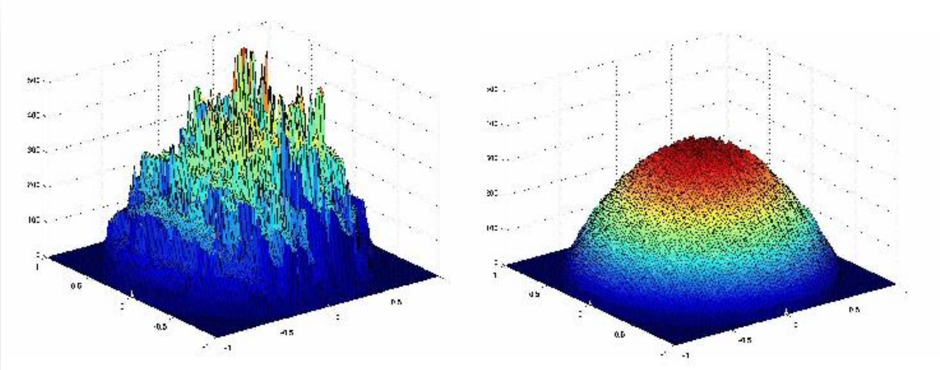
[Hill 1996] : HILL D., "Object-oriented Analysis and Simulation", Addison-Wesley, 1996, 291 p.

BASICS FOR **REPEATABLE** STOCHASTIC SIMULATIONS WITH PARALLEL RANDOM NUMBER GENERATORS

Before proposing a method, we need to be aware of some elements:

1. Check with some top PRNGs used with different execution context (hardware, operating systems, compilers… (Use exactly the same inputs, Execute on various environments, When possible, compare our outputs with author's outputs (from publications or given files)

2. Have a short list of top generators.



3. Be aware that the initialization of generators can matter (keep a huge amount of fine statuses if needed).

4. Be aware of the major parallelization techniques for the current top generators

DAO V.T., MAIGNE L., BRETON V., NGUYEN H.Q., HILL D., "Numerical Reproducibility, Portability And Performance Of Modern Pseudo Random Number Generators : Preliminary study for parallel stochastic simulations using hybrid Xeon Phi computing processors", European Simulation And Modelling Conference, Oct. 22-24, 2014, Porto, Portugal, pp. 80-87.

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 Phylox 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.

See the following reference for advices including hardware accelerators.

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.

A Method For Repeatable parallel stochastic simulations

Remember that a stochastic program is « deterministic » if we use (initialize and parallelize) correctly the pseudo-random number.

1. **An object oriented approach has to be chosen for every stochastic objects which has its own random stream.**

2. Select **a modern and statistically sound generators** according to the most stringent testing battery (TestU01);

3. Select **a fine parallelization technique adapted to the selected generator**,

4. The simulation must first be designed as a sequential program which would emulate parallelism: this sequential execution – **with compiler flags set on 'repeatability'** – will be the reference to compare parallel and sequential execution at small scales on the same node.

5. Externalize, sort or give IDs to the results for reduction in order to keep the execution order or use compensated algorithms

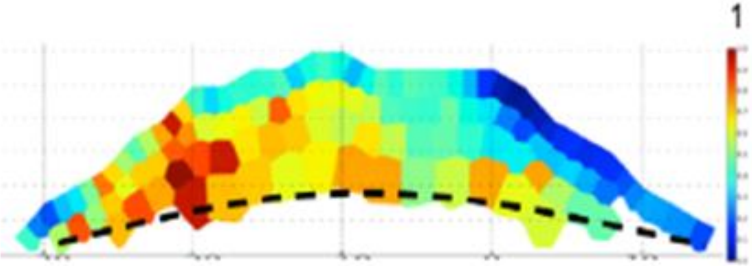
[Hill 2015] : Hill D., "Parallel Random Numbers, Simulation and reproducibility". IEEE/AIP - Computing in Science and Engineering, vol. 17, no 4, 2015, pp. 66-71.
[Hill et al 2017] : Hill D., Dao V.T., Mazel C., Breton V., « Répétabilité et reproductibilité numérique - Constats, conseils et bonnes pratiques pour le cas des simulations stochastiques parallèles et distribuées ». TSI, Technique et Sciences Informatiques, Vol. 36 n° 3-4/2017, pp. 243-272

Sample Test Application: Parallel Monte Carlo Simulation of Muonic Tomography

TOMUVOL PROJECT

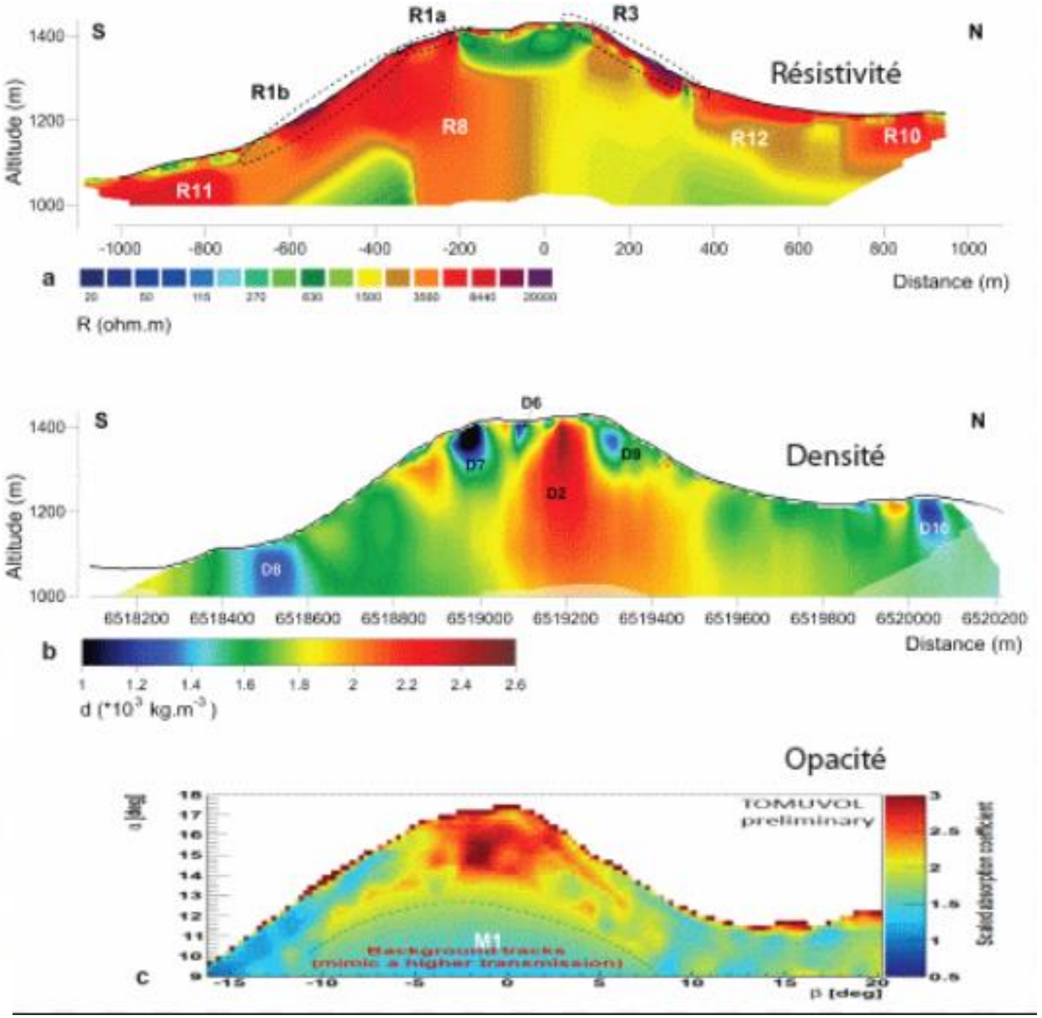
http://wwwobs.univ-bpclermont.fr/tomuvol/presentation.php

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)



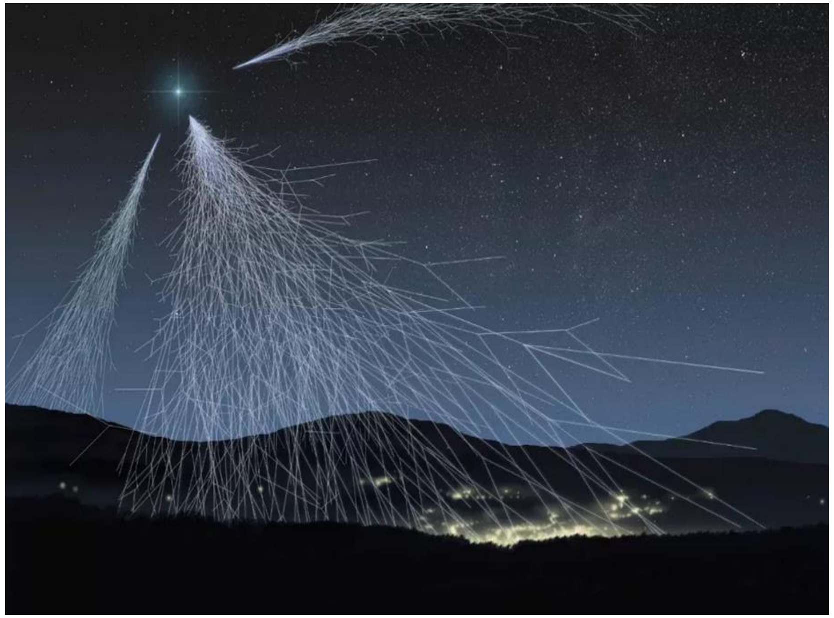
Density contrast

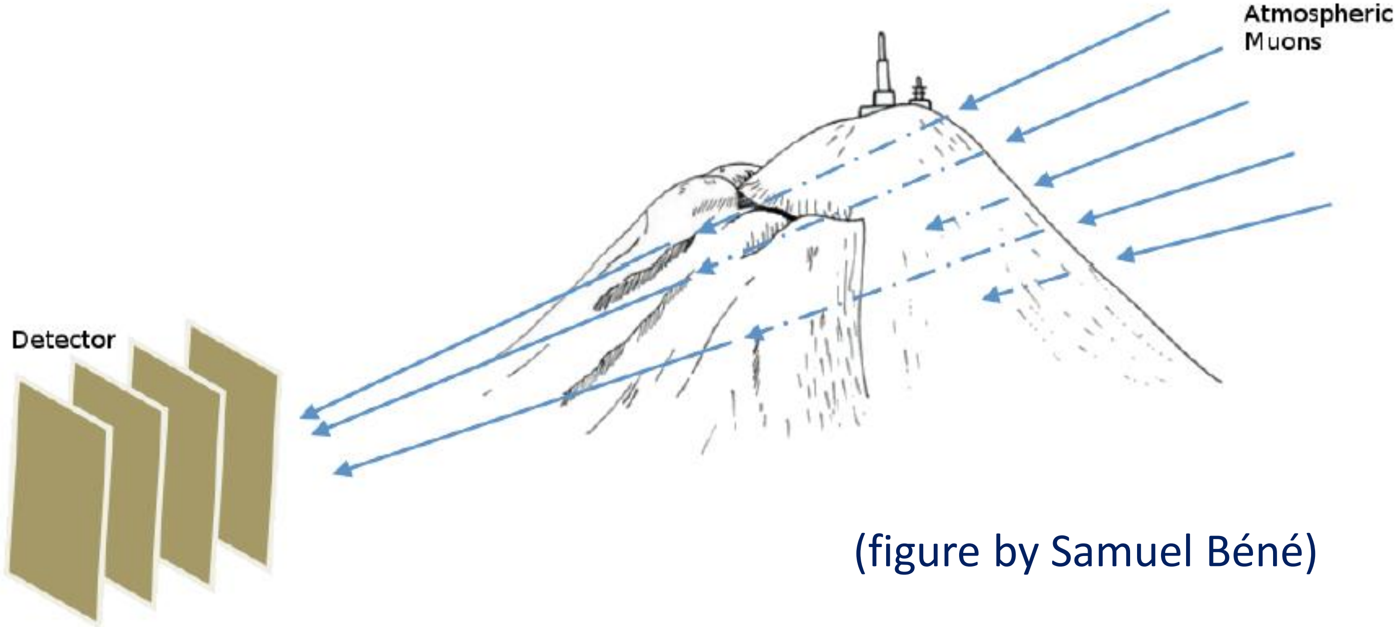
14 days of data taking, 0.66 m^2 x 1 m



Principle of Muonic Tomography

Atmospheric muons go through matter. Depending on their energy and of the matter they traverse it is possible to reconstruct the inner image of a large edifice with multiple sensors





(figure by Samuel Béné)

The muon is an elementary particle similar to the electron, with a negative charge and a spin of 1/2, but with a much greater mass. It is classified as a lepton. The muon is not believed to have any sub-structure—that is, it is not thought to be composed of any simpler particles (as is the case of other leptons).

TARGET NODES WITH REGULAR XEON & INTEL XEON PHI
XEON PHI – STILL ON TOP CEA MACHINE LIKE JOLIOT – CURIE > 9 PF

Parallel stochastic simulation of muonic tomography – Aim finish computing **in less than 24h**

- Parallel programming model using p-threads https://github.com/HeisSpiter/HPCsim
- Each Muon is a stochastic object
- Multiple streams using MRG32k3a
- A billion threads handled by a single node

1st we did a round of sequential optimization
with the code given by our physicists colleagues
16X on a single CPU core – then // 25X with 32 phys.c.
400X on a node – Seq. computing time on a single node : 3 months

- Compiling flags set to maximum reproducibility – Sequential results obtained after 5 weeks – 3 months for a single Phi core (results below are with all CPU/Phi cores).

Performance of a billion event simulation when parallelized on 1 Phi, 1 CPU, 2 CPUs

	Intel Xeon Phi 7120P	Intel Xeon E5-2650v2	2x Intel Xeon E5-2650v2
Time	48 h 49 min	36 h 32 min	18 h 17 min
Speedup	1	1.34	2.67

SCHWEITZER, P., MAZEL, C., FEHR, F., CÂRLOGANU, C., HILL D., “Proper parallel Monte Carlo for computed tomography of volcanoes”, Proceedings of the 2013 International Conference on High Performance Computing & Simulation, ACM/IEEE/IFIP, Helsinki July 1st-5th, 2013, pp. 519-526.

Reproducibility Between Phi & regular Xeon First Attempts

- First try with simple compilations of simulation to study the validity of the results
  Intel C compiler with the "-O2 -g -Wall -Wextra" - (no -fast-math no aggressive -O3)
- For Xeon Phi, we added the "-mmic" option. (no -fast-math no aggressive -O3).
- We evaluate the deviation in the results when the compilation is left free (limited to 1000 muons events – muon reaching the detector). Very important differences in final muon energy have been noticed (up to 0.18 GeV). We also noticed important differences for the final position (up to 0.3 m).
- If the initial energy of the particle is between 5 GeV and 10 TeV, its final energy is between 0.15 GeV and 5 TeV (or even zero, if it does not even reach the detector). A difference of 0.18 GeV is therefore not acceptable.
- The detector has plans whose size is one meter by one meter. An inaccuracy of 0.3 m on the end position means a 30% inaccuracy on one dimension of the plane!
- Worse, the detector has a spatial detection of about 1 cm. An inaccuracy of the order of 30 cm (i.e., 30 times more!) shows a clear failure of the reproducibility of the simulation.

More Careful Attention To Compiler Flags

After different tries with Intel Compiler flags we retained the following:

- ✓ `"-fp-model precise -fp-model source -fimf-precision=high -no-fma"` for the compilation on the Xeon Phi – (no `-fast-math` no aggressive `-O3`)
- ✓ `"-fp-model precise -fp-model source -fimf-precision=high"` for the compilation on the Xeon CPU – (again no `-fast-math` no aggressive `–O3`)

With this set of flags, the results on the two architectures are reproducible (the same order).

Both of them have the same sign and the same exponent (even if some exceptions would be theoretically possible, they would be very rare and haven't been observed).

The only bits that can differ between these results are the least significant bits of the significand.

For a given exponent e, and a result r1 = m × 2e, the closest value greater than r1 is r2 = (m + εd) × 2e, where εd is the value of the least significant bit of the significand: $\varepsilon d = 2^{-52} \approx 2.22\ 10^{-16}$.

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.**

- 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.

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.**
- **Comparison possible with sequential results !!! – (scale of a node - with a 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…

QUESTIONS ?