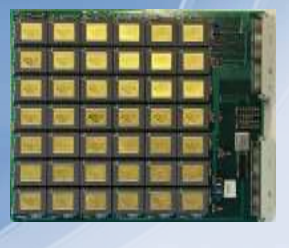
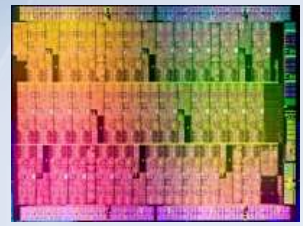
Distribution of Random Streams For Parallel Stochastic Simulation


David Hill
UCA (former Blaise Pascal University)
UMR 6158

Simulations Deterministic vs. Stochastic

- **Model** : representation of a system
- **Simulation** : Evolution of this model during time
- **Deterministic** : the output of the model is precisely determined by its structure and its parameter values
- **Stochastic modelling** : means that we use a random source (often deterministic if for Science)



In various domains we still need to increase reliability…

http://www.youtube.com/watch?v=bzD4tIvPHwE

In the 1990s Mulhouse FRANCE

First fully electronic automated commercial plane, Automatic landing test with no pilot, no electro-mechanic controls or cables

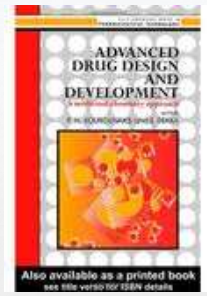
Crash during auto landing !

- Analytical and deterministic models are very fast and have to preferred when possible

BUT : they can be imprecise in some conditions

- Monte Carlo (MC) simulations helps increasing model precisions with spatial constraints but they can be very slow
- In many industries stochastic models are more widely used for risk assessment and to take into account rare and or random events
- Quantitative risk analysis can be improved with Monte Carlo simulations

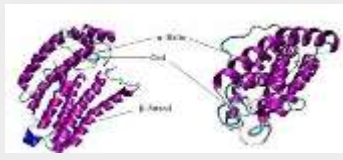
Pharmaceutical Industry & Drug discovery



Ex: problems with the Mercator drug in France (and with more than 200 other drugs)

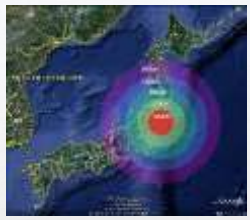
Example of Improvements :

- Multi-scale stochastic drug release model
- Stochastic model for the origin and treatment of tumors containing drug-resistant cells
- Comparison of stochastic models to to predict the influence of drug distribution, enzyme heterogeneity…
- …



With Fukushima, many realized that very improbable does not mean : Impossible

Improving our skills is crucial : vulnerability assessment




- **Stochastic** models of normal **plant** operation
- **System reliability Monte Carlo models**
- **Simulation of ground motion using stochastic method**
- **Non-stationary stochastic model of earthquake motions…**
- **Shock wave modeling…**

5

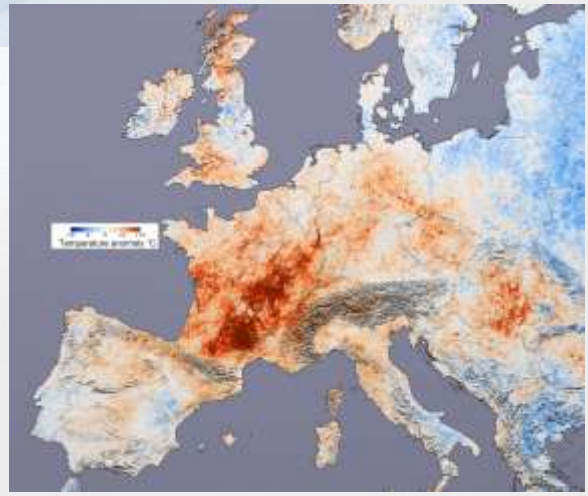
Recent World Health Organization report dealing with Mobile phone industry

- Improvement of spatial models

- Estimation of how a mobile phone electromagnetic radiation penetrates your brain depending on your age. (GSM frequency 900 Mhz)
  On the right – Specific absorption rate in W/kg at different depth) – Check the DAS of your future mobile phone…

Credit : Dr Bernard Asselain,
Head of Cancer Biostatistic group - Curie Institute France

Stochastic models for climate too…



Europe

Summer 2003

Temperature Anomaly

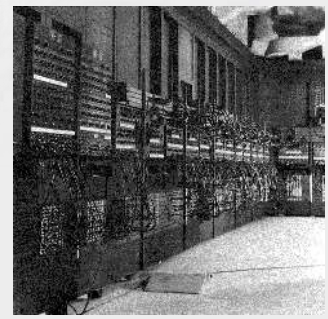
Dark red = 10°C Above average

Majda, A. J., I. Timofeyev, E. V. Eijnden, 2001: A mathematical framework for stochastic climate models. *Commun. Pure Appl. Math.*, ***54, 891-974.***
DelSole T., 2004: Stochastic models of quasigeostrophic turbulence. *Surveys in Geophys,* ***25 (2), 107-149.***

Summer 2003 – Where are we ?

In Lorraine : North of France… Summer 2003…

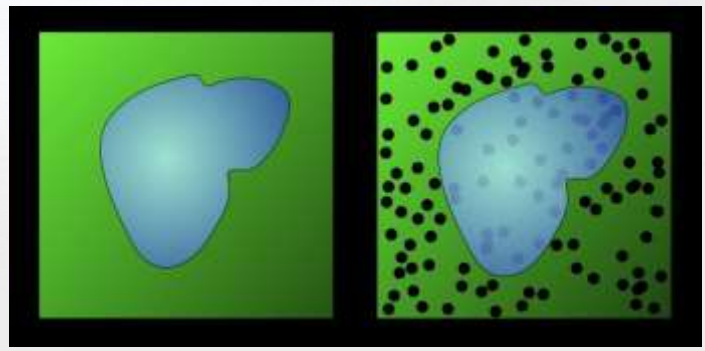
WWII– Code Name Monte Carlo



A computing method for initially for complex multiple integrals – used in nuclear physics…
Now used is many civil domains.

10

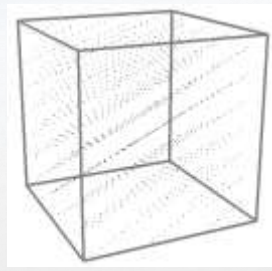
Do you play darts ?



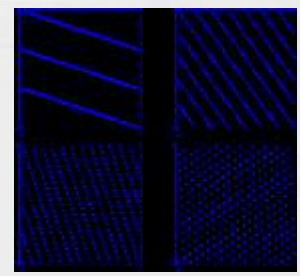
Surface estimation – Uniform random shooting
Slow and rough approximation

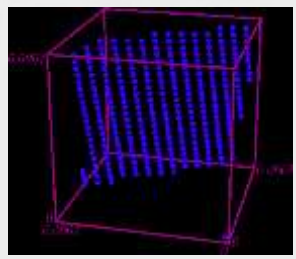
But : Works fine even with a large number of dimensions.

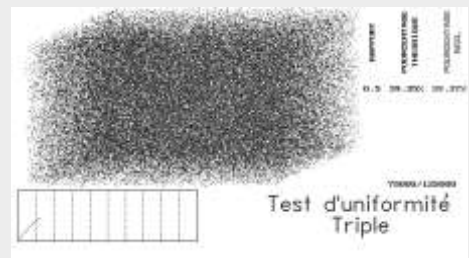
So we need a uniform generator…?
Are std. gen. really well equi-distributed ?



Spectral tests



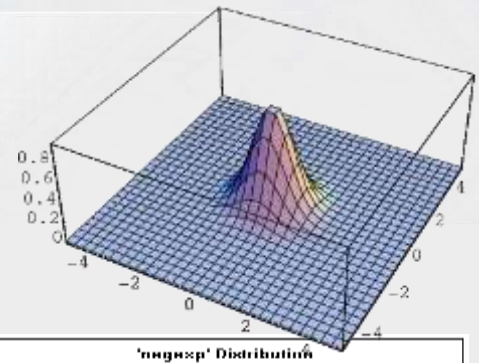


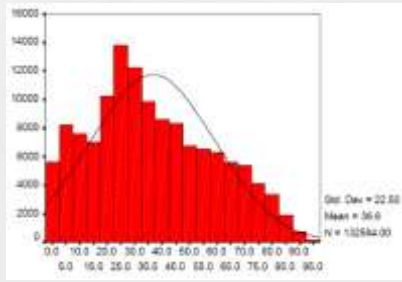


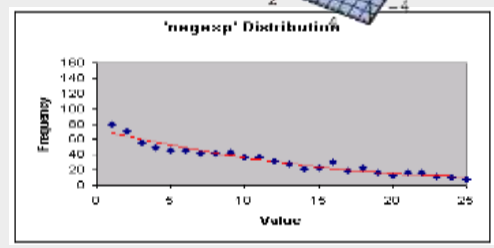
With uniformity we can do anything…

« *God does not play dice...* » A. Einstein

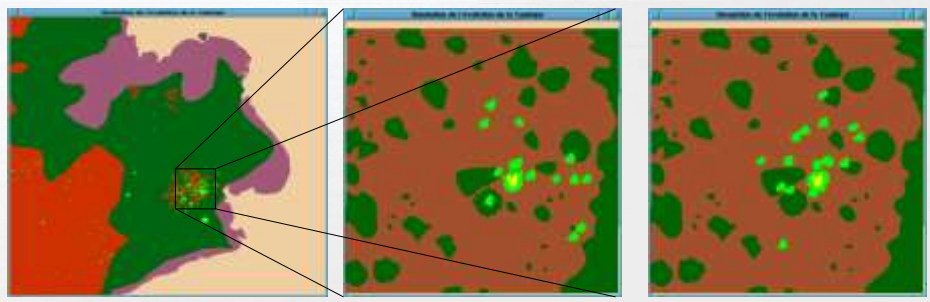
- Anamorphosis
  (inverting a distribution law)
  $x = F^{-1}(rn)$
- Rejection Method
- Reproducing Histograms

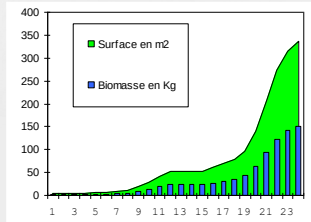


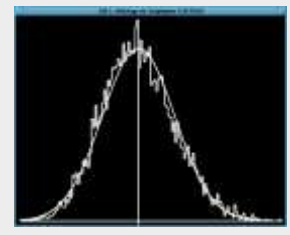




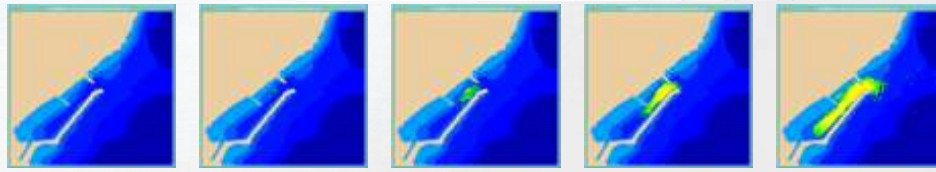
Statistical studies of 2D results with spatial constraints – replications…

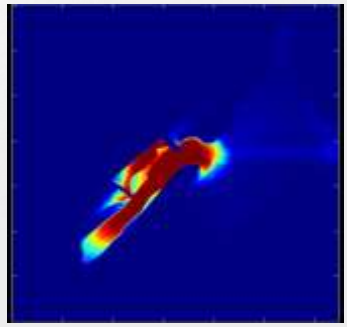






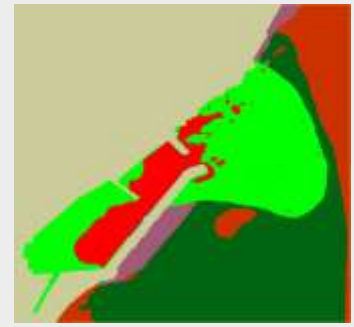
Spectral analysis of stochastic simulations with 2 dimensions spatial constraints (*Caulerpa taxifolia)*





< Result of a spectral analysis

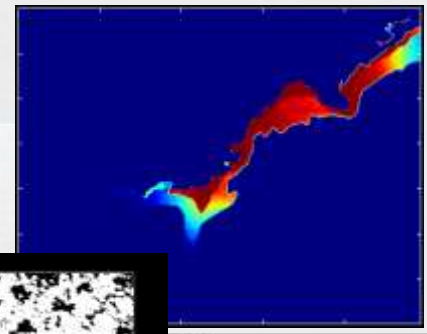
Comparative study >

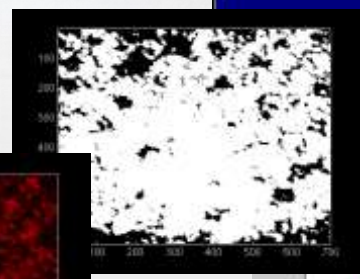


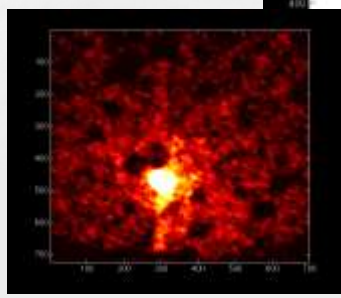
15

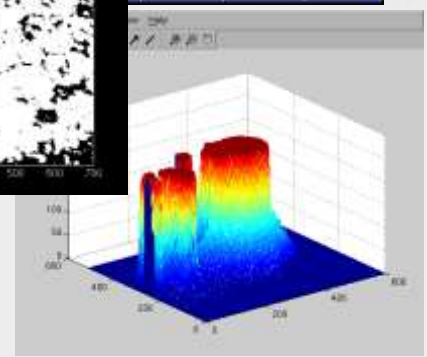
Spectral analysis

Among the various validation methods, the spatial spectral analysis presents a great interest in studying spatial correlations of stochastic simulation results.



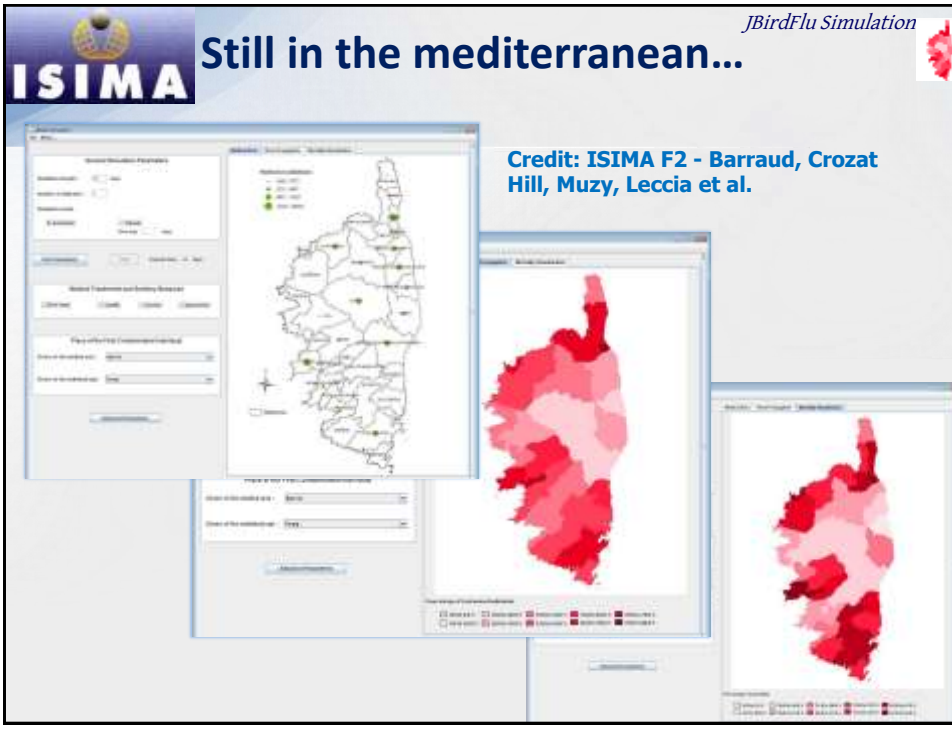






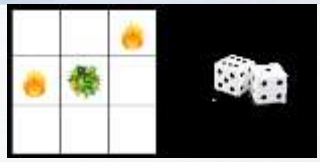
Still in the mediterranean…

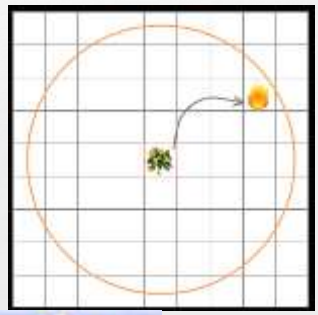
JBirdFlu Simulation

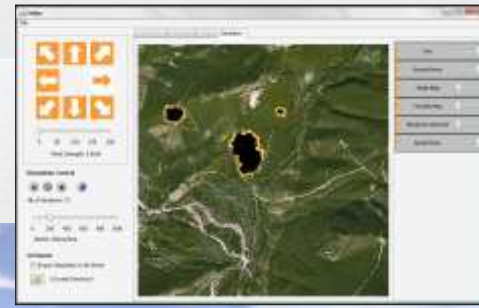


Credit: ISIMA F2 - Barraud, Crozat Hill, Muzy, Leccia et al.

Fire forests and fire brands simulation

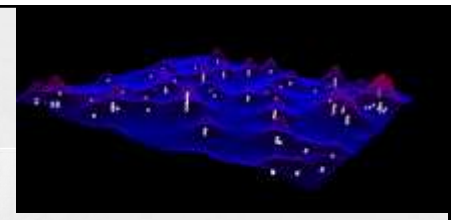






Fire Simulation

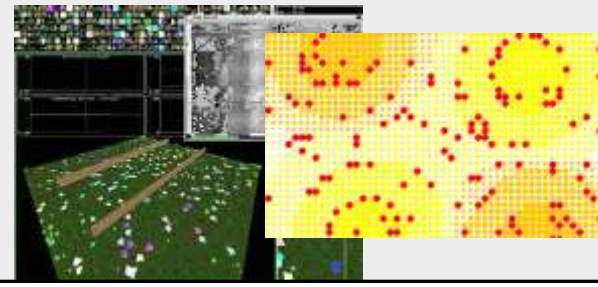
Towards artificial societies



- Robert Axtell is a researcher in Social Science at the Mason Center for Social Complexity (CSC) proposes model to study human interactions with multi-agents systems.
- « Interactions could be social, financial or political. It's very hard to render those models mathematically – not easy to summarize the functionality or the performance of the simulation groups in numbers or graphs. Often what we'll try and do is depict the entire market as it emerges."

Robert Axtell
Brookings Institute

Polyworld, heatbugs, SugarScape…



To summarize : Monte Carlo and stochastic simulations...

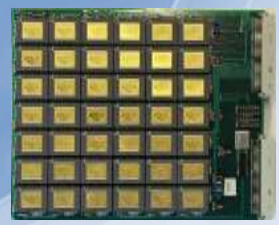
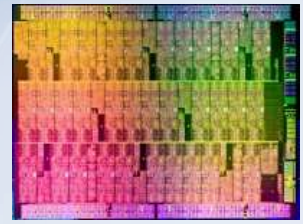
- Use a set a statistical sampling experiments which make a smart use of randomness ("stochos" in Greek led to "stochastic"
- Based on a random sampling
- The convergence speed is slow :
  - Approximatively $O(N^{-1/2})$

How can we accelerate Monte Carlo methods ?

- Variance reduction techniques (maths & stat)
- "Quasi-Monte Carlo" Methods
- **Parallel Computing**

20

Distribution of Random Streams For Parallel Stochastic Simulation

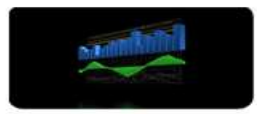




Part II


Bio-Informatics and Life Sciences

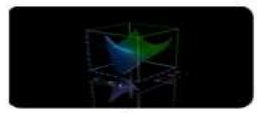

Computational Chemistry


Computational Electromagnetics and Electrodynamics

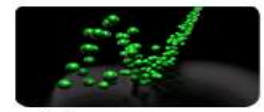

Computational Finance

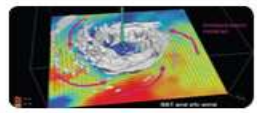

Computational Fluid Dynamics


Imaging and Computer Vision


MATLAB Acceleration


Medical Imaging


Molecular Dynamics

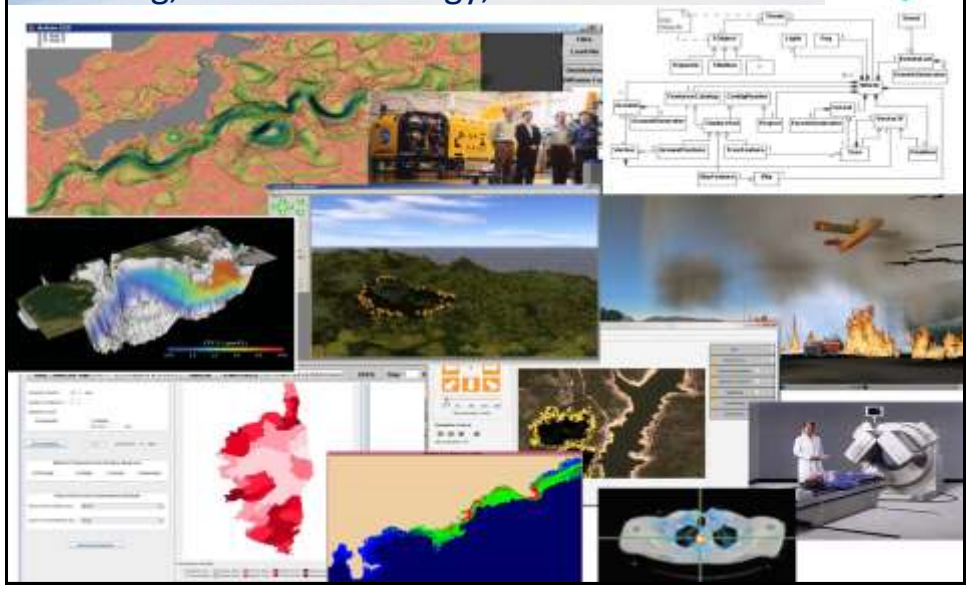

Weather, Atmospheric, Ocean Modeling, and Space Sciences

In many research domains, we often try to improve our knowledge with slower stochastic models.
But most of the time they are easy to distribute on // computers ☺ !

However, be careful when designing Parallel Stochastic Simulations

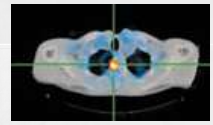
- Deterministic approaches to distributed simulation are well studied (Chandy, Misra, Jefferson, Fujimoto,…)
- **New scientific barriers to break are sometimes far behind our best deterministic models**
- **The assessment of parallel stochastic simulations is tough and this domain is less studied**

Experience from past distributed simulations In Life Science, Environmental, Ecological Modeling, Molecular Biology, Nuclear Medicine…

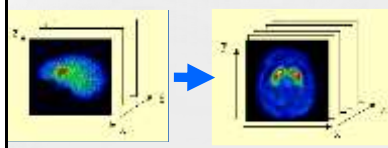


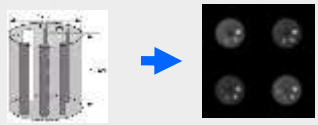
Ex: safer use of radioactive products…
An application to Nuclear Medicine
Stochastic vs. deterministic

Credit : El Bitar, Buvat, Breton, Reuillon and Hill
INSERM, LPC, LIMOS - CNRS



3D Monte-Carlo Simulations are used to improve the quality of SPECT & PET Tomography images - less than 10% of errors in small tumors detection compared to analytical models (much faster but with 100% of errors for small tumors below 1 cm)





Approx :
3 years of CPU in 2006
3000 MC simulations

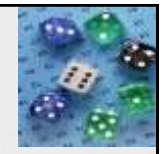
The computing of image reconstructions lasted 3 days with 600 EGI nodes Grid and two local clusters.

25

What can we easily distribute
in modeling & simulation ?

- **Replications (MRIP)** → multiple independent simulations
- Scenario analysis → multiple independent simulations
- Sensitivity analysis → multiple independent simulations
- Monte Carlo analysis → multiple independent simulations
- Optimization :
  - Genetic Algorithm → multiple independent simulations
  - Simplex Algorithm → multiple dependent simulations !
- **Experimental Design** → multiple independent sensitivity analysis for different parameters
- Robustness Optimization
  = Optimization over sensitivity analysis
  → multiple independent sensitivity analysis for different parameters
- PDEs & continuous simulations → multiple dependent simulations !

Basics in stochastic modeling



- Even a unique « random » source in your code makes your model « **stochastic** »
- More than 50% of « supercomputing » applications use random sources :

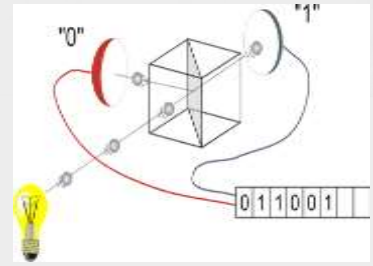
  Software sources :
  - Pseudo-random
  - Quasi-random (can improve performances in some limited cases)

  Hardware sources :
  - Some recent quantum devices are promising
  - Not currentlty used in HPC
  - Main drawback - not reproducible
- Compared to analytic models their implementation are **VERY** slow : **parallelism is often mandatory**

27

Some new perspectives using Quantum Random numbers

- Quantis is a physical random number generator exploiting an elementary quantum optics process.
- Photons - light particles – are sent one by one onto a **semi-transparent mirror** and detected.
- The exclusive events (reflection – transmission) are associated to "0" - "1" bit values.



- The operation of Quantis Hardware device is continuously monitored to ensure immediate detection of a failure and disabling of the random bit stream.

28

Hardware caracteristics

- USB - High bit rate of 4Mbits/sec
- PCI cards – Very high bit rate : 16Mbits/sec

Expensive if you have to buy one device per HPC node
Could be interesting for cryptography.

29

Still some questions…

- **The main problem – For large simulations**
  How can we reproduce :
  - **sequences ?**
  - **Experiments ?**
- Cost of multiple devices in massive parallel architecture
- Correlation between devices ?
- Splitting and archival (massive storage ?!) ?
  - Limited to "small" sequences
    and
  - Limited use - Memory mapping / Hardware dependant
- Bias correction & questionable inner quality of tests

30

Most stochastic simulations use PRNGs (Pseudo Random Number Generators)

Quick survey of the main PRNG types: **Green: recommended**

- LCG (Linear Congruential Generator)
  $x_i = (a*x_{i-1} + c) \bmod m$
- LCGPM (Linear Congruential Generator with Prime Modulus – could be Mersenne or Sophie Germain primes)
- **MRG** (Multiple Recursive Generator)
  $x_i = (a_1*x_{i-1} + a_2*x_{i-2} + \ldots + a_k*x_{i-k} + c) \bmod m$ – with $k>1$
  (Ex: **MRG32k3a & MRG32kp** – of l'Ecuyer and Panneton)
- LFG (Lagged Fibonacci Generator)
  $x_i = x_{i-p} ? x_{i-q}$
- **MLFG** (Multiple Lagged Fibonacci Generator) – Michael Mascagni MLFG 6331_64
- L & GFSR (Generalised FeedBack Shift Register…) Mod 2
  Mersenne Twisters (MT19237, SFMT, MTGP) – **WELLs**

Generation of parallel Pseudo Random Numbers Streams

- **Well-known sequential techniques exist for generating PRNs, in a deterministic fashion:**
  - **MRG32k3a & MRG32kp,**
  - **MT & Well families of generators**
  - **MLFG_6331_64** (not always portable)
- **The number sequences are largely indistinguishable from true random sequences**
- **The deterministic nature is necessary to have the reproducibility of the large scale simulated experiments**
- **Efficiency is needed, but we have to preserve**
  - **randomness**
  - **reproducibility**

So why should we prefer Pseudo-Random Numbers ?

- True random numbers are rarely used in computing because:
  - they are difficult to generate reliably
  - the lack of reproducibility for large scale simulations would make the validation of programs that use them extremely difficult if not impossible
- Computers use pseudo-random numbers:
  - finite sequences fastly generated by a deterministic process
  - but indistinguishable, by some set of statistical tests, from a random sequence.
- Quasi-random numbers are also considered for some kinds of applications (integral computing)

Application Areas 1/2

- Simulation: When we simulate natural phenomena, random numbers are required to make things realistic. For example: where & when do people come into an airport, job submissions, phone calls, etc...
- Sampling: It is often impractical to examine all possible cases, but a random sample will provide insight into what constitutes a "typical" behaviour.
- Computer Programming: Random values make good source of data for testing the effectiveness of computer algorithms.

Application Areas 2/2

- Numerical Analysis
- Decision Making
- Aesthetics: A little bit of randomness makes computer generated graphics and music seem more lively.
- Reproduce "reality" or solve complex problems with a "Monte Carlo method" :
  - MC Method tends to become a general term used to describe any algorithm that employs random numbers and sampling to compute a result.

Parallelism in Stochastic & Monte Carlo (MC) Applications

MC simulations

- are computationally intensive
- most of the time applications appear to be naturally parallel
- Appropriate for the independent *bag-of-work* paradigm
- Fits the distributed computing paradigm

But remember that we want rigorous simulations !

Here is the main requirement for parallel stochastic simulations :

Independence of the parallel random number streams
used by the different logical processors
(CPUs, physical or logical cores, threads…)

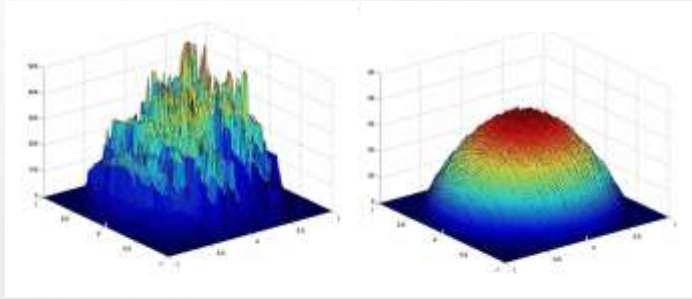
36

« Random number generators, particularly for parallel computers, should not be trusted. »

In [Traore & Hill 2001] **By Paul Coddington**

- It is necessary that RNs can be generated in parallel (ie **each process may have an autonomous access to a sub-sequence** issued from a common global sequence
- **If such an autonomy is not guaranteed, the potential parallelism of the application is affected** (if for instance processes access to a central RNG, even if also run in //)
- In addition, as stated by Paul Coddington: *"It is strongly recommended that all simulations be done with two or more different generators, and the result compared to check whether the random number generator is introducing a bias"*
- The **main problem is to find partitioning techniques** which preserves the good properties required to guarantee not only the efficiency of the simulation but mainly **the credibility of the results.**

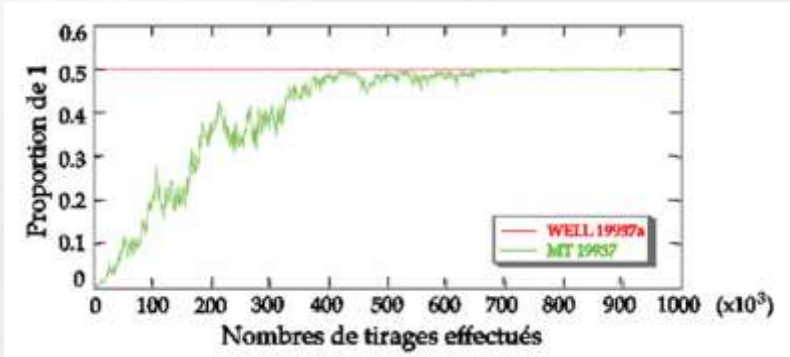
Let's deal with the real impact of the generator quality…



Two results of the same simulation (sequential) – PDE Harmonic solution computed with Brownian movements.
On the left the image is obtained with Linux rand
On the right – same simulation with a Mastumoto Mersenne Twister (1997 version) – right solution ellipsoid with a circular section.

There is no perfect Generator… (1)
First Mersenne Twister (1997) : a known default…

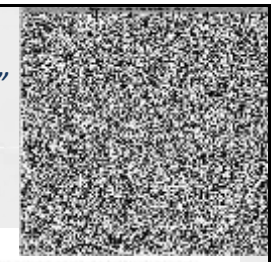
Between 1997 and 2002 : very long recovery of **zero-excess initial state** for MT19237 (700 000 draws…)

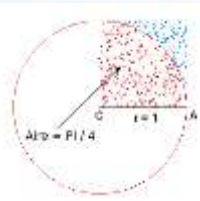


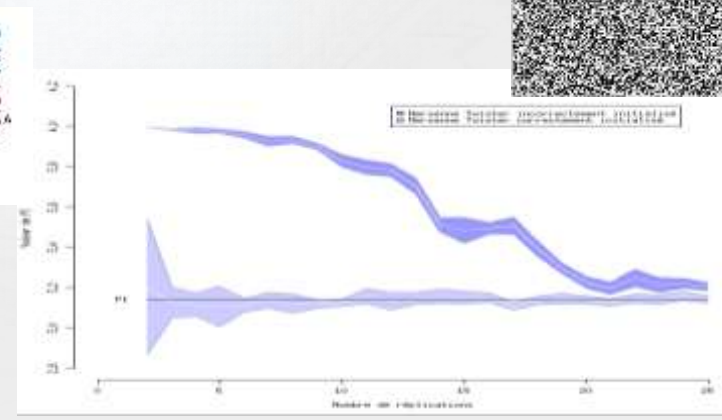
The 2002 version is correcting this problem with a faster recovery in case of zero-excess initial state.

There is no perfect Generator… (2)

We are here dealing with "mere cooking recipes" for making digits (1951 Von Neumann)







If it is not correctly initialized, even a very good generator can lead to biased results. Here, we obtain 3.25 for π after 25 replications each drawing 50 000 points (100000 pseudo-random numbers). π is not even in the confidence interval (Darker Blue) after the 25 replicates… Lighter blue : a correct simulation.

Even some not so old and well distributed software, were showing strong weaknesses… were used for distributed simulations

- NS 2 – Famous Network Simulation Software
  - Over 50% of ACM and IEEE network simulation papers from 2000-2004 cite the use of *ns-2*
  - 8000 download / month
  - Was based on a 1969 PRNG with 2^{31} period and used for // simulation.
- CLHEP – Particle Physics Library still has many fairly obsolete generators – kept for reproducibility of legacy applications (used in GEANT 4 for nuclear & particle physics)
- GATE : nuclear medicine software, based by default on James random PRNG (1997) – using CLHEP and based on GEANT 4
  - They changed their default generator to Mersenne Twister

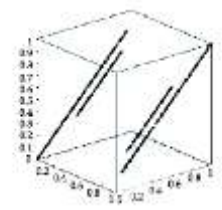
41

"not so old" simulation software still widely used…

- Even after improvements a famous **Open Source Network simulator – developed and financed by DARPA** (Defense Advanced Research Project Agency) was still showing great weaknesses

- It proposed an integer seed for its parallel PRNG initialization

BUT: simple tests with 1,2 & 3 were showing strong linear correlations.



42

(Shown by Entacher et al., 2002)

NS 2 again with a "very good – generator" from L'Ecuyer…
But a very bad seeding API…

- Due to bad documentation and re-use of old scripts many people still use the old API functions to explicitly set seeds.
- Unfortunately, this corrupts the correct functioning of the new generator (**MRG32k3a**) and can lead to correlated simulation results.
- This might affect the ns-2 simulation results
  - how many currently published results ?

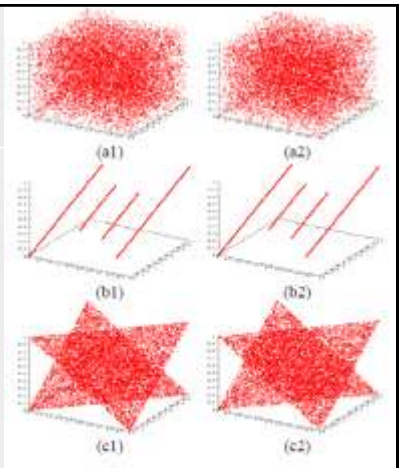


Figure 1. Correlation between three random variables for 10,000 values drawn. Left: MRG32k3a, right: old Park/Miller RNG. (a1) new (correct) seeding method, (a2) Seedset 1 (known "good" seeds), (b1/2) Seedset 2, (c1/2) Seedset 4.

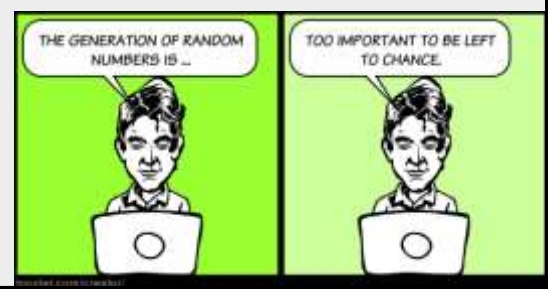
By Martina Umlauft & Peter Reichl

"Don't Trust Parallel Monte Carlo !!! "

Hellekalek PADS 1998

- There are many examples were bad parallelization of random streams lead to erroneous results
- In 2003 Entacher and Hechenleitner, showed the kind of pitfalls obtained when using parallel streams in OMNET++ simulations

But let's also talk about **OUR errors…**



Impact of bad parallelization of random streams on *small tumor detection*

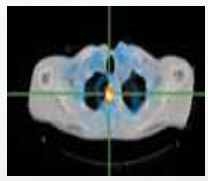


Image that should be obtained…

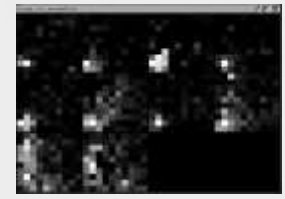
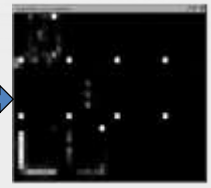
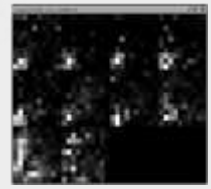


Image we obtained involuntary correlations and overlapping between parallel random streams





Correct simulated reconstruction

Image reconstructed with correlations between parallel random streams

45

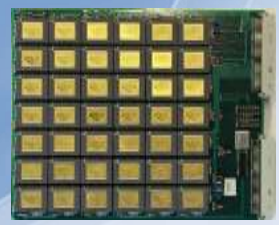
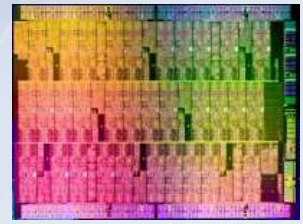
Manual scripting

So… what are the requirements for Parallel RNGs ?

- They have to be **easy to split** (partitioning the numbers into many independent sub-sequences that are allocated to different Logical Processors or Cores, without the need of communication or synchronization;
- Each sub-sequence has to be a good sequential RNG; it must possess good "random" qualities according to the current most stringent tests (TestU01)
- There should be **no correlation between the sub-sequences on different LPs** (crossed-correlation)
- No "mathematical proof" can really help…you have to run tests or to use "tested sequences".

46

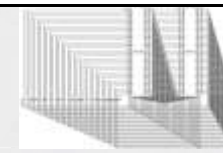
Distribution of Random Streams For Parallel Stochastic Simulation


Part III

Some top pseudo-random number generators and their partitioning techniques

Let's start with some history of the current "best" generators



Matsumoto & Nishimura - Mersenne Twister 1997

- Large linear-feedback shift register
- The "Seed" is a register 19937 bits long, stored in a 624 array of 32 bits values
- Period: $2^{19937} - 1$ (Mersenne Prime #)
- Claimed to be free of long-term correlations and equi-distributed in 623 dimensions
- Not cryptosecure

48

(2004) WELL…
Panneton, L'Ecuyer & Matsumoto

- In fact the huge-period generators proposed so far are not quite optimal when assessed via their equi-distribution properties.
- **WELL corresponds to new generators, with better equi-distribution and "bit-mixing" properties** for equivalent period length and speed.
- Approximately half of the coefficients of the characteristic polynomial of these generators are nonzero.
- The state of this new kind of generators evolves in a more "random" way than for the original Mersenne twister.
- This can reduce the impact of persistent dependencies among successive output values, which can be observed in certain parts of the period of gigantic generators such as the MTs.

49

MT has been updated : 2002, 2004, 2006, 2009…

- SFMT is introduced in 2006 and MTGP in 2009
- Improves the long recovery of zero-excess initial state (enhancements since 2002)
- SFMT (Saito & Matsumoto – 2006)
  - SIMD-oriented Fast Mersenne Twister (SFMT) using SSE instruction set of modern processors
  - SFMT is roughly twice faster than the original Mersenne Twister,
  - Has a better equi-distibution property,
  - And a quicker recovery from zero-excess initial state.
- MTGP… (presented later) for GP-GPUs

50

SFMT with April 2009 updates

- Period ranging from $2^{607}-1$ to $2^{216091}-1$
- SFMT is faster than MT, in most platforms.
- Improvement in the dimensions of equi-distribution
- SFMT19937 is less portable but can be compiled in three possible platforms:
  - Standard C without SIMD instructions
  - CPUs with Intel's SSE2 instructions (Streaming SIMD Extension 2)
  - CPUs with PowerPC's AltiVec instructions

See : http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html

51

Some requirements for massive parallelism in Stochastic Applications

- Fits with the Independent bag-of-work paradigm
- As we said : ensure as much as possible the "independence" of underlying random number streams
  - Use partitioning techniques
  - Use dedicated libraries

The techniques presented hereafter will fit with **different distributed computing platforms**

Quick survey of random streams parallelization (1) **Using the same generator**

- The **Central Server** (CS) technique
- The **Leap Frog** (LF) technique. Means partitioning a sequence $\{x_i, i=0, 1, \ldots\}$ into 'n' sub-sequences, the j^{th} sub-sequence is $\{x_{kn+j-1}, k=0, 1, \ldots\}$ - like a deck of cards dealt to card players.
- The **Sequence Splitting** (SS) – or blocking or regular/fixed spacing technique. Means partitioning a sequence $\{x_i, i=0, 1, \ldots,\}$ into 'n' sub-sequences, the j^{th} sub-sequence is $\{x_{k+(j-1)m}, k=0, \ldots, m1\}$ where m is the length of each sub-sequence
  - Jump Ahead technique (can be used for both Leap Frog or Sequence splitting)
- The **Indexed Sequences** (IS) - or random spacing. Means that the generator is initialized with 'n' different seeds/statuses
- The **Cycle Division** or **Jump ahead** approach. Analytical computing of the generator state in advance after a huge number of cycles (generations)

53

Quick survey of random streams parallelization (2) **Using different generators**: parameterization

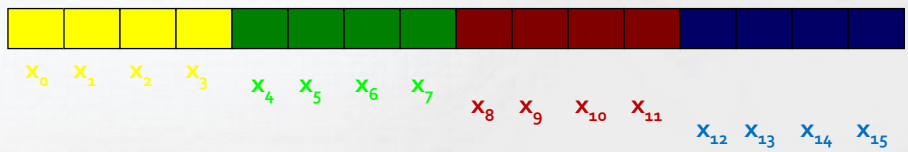
- The same type of generator is used with different parameters for each processor meaning that we produce different generators
- In the case of linear congruential generators (LCG), this can rapidly lead to poor results even when the parameters are very carefully checked. (Ex: Mascagni and Chi proposed that the modulus be Mersenne or Sophie Germain prime numbers)
- Explicit Inversive Congruential generator (EICG) with prime modulus has some very compelling properties for parallelizing via parameterizing. A recent paper describes an implementation of parallel random number sequences by varying a set of different parameters instead of splitting a single random sequence (Chi and Cao 2010).
- **In 2000 Matsumoto et al proposed a dynamic creation technique**

54

Central server approach

- **Not truly parallel**.
- A central server (CS), running a "good" sequential RNG and provides pseudorandom numbers on demand – feeding other processors (first demand, first served)
- The two major drawbacks for this approach are the following:
  - **a simulation using this technique will not be reproducible** (if the number of nodes / CPUs / cores is varying)
  - **the central server can become a bottleneck** when we have a large number of processors
- Suitable for serious games with limited parallelism or some decision aid software but not for scientific applications.

Sequence splitting technique



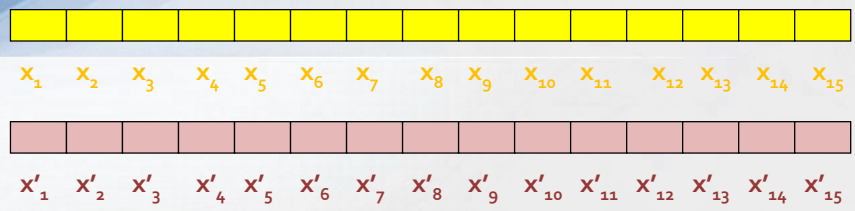
- Split a sequence into non overlapping contiguous blocks.
- Hechenleitner showed that in the OMNeT++, **the spacing between sequences was set to 1 million drawings led to biased results** (due to inter-sequence correlations) for processes using more random numbers.
- Even if overlapping can be easily avoided, long range correlations in the initial RNG can lead to small range correlations between the potential substreams.

Leap Frog Technique

- Partition of random stream like a deck of cards dealt to card players.
- Given the period p of the global sequence, the period of each stream is p/N.
- Like with the splitting technique, the long range correlations in the initial RNG can lead to small range correlations between the potential substreams, particularly if we have a large number of processors.
- Wu and Huang 2006 showed that **depending on the interval used** (ie the number of processors) **cross correlations can be observed**.
- A case where the quality of the original RNG is seriously affected by the LF technique was shown by Hellekalek.
- When this technique is used without cycle division (explained in a forward slide), the performances are divided by the number of LPs and we find again the bottleneck problem exposed in the Central Server technique.

57

Independent sequences technique



- The Indexed Sequences (IS) method builds a partition of N streams by initializing the same generator with N random statuses.
- In the case of old LCGs it was named random seeding.
- For modern generators with a more complex status, the random statuses are generated with another RNG and this technique is interesting when generators have a huge period.
- The risk is of course to get a bad initialization. Recent history has showed that even some of the best RNG algorithm could fail when badly initialized.
- In 2008, Reuillon proposed 1 million statuses for the first Mersenne Twister (MT) with its $2^{19937} - 1$ period, he used a RNG with cryptographic qualities to propose independent and well balanced bit statuses.

Cycle division (1/2)

- This technique enables the analytical computing of the generator state in advance after a huge number of cycles (generations) and corresponds to a jump ahead in the random stream.
- Interesting with generators that have very large periods.
- Until recently, MT and the WELL generators, based on linear recurrences modulo 2, did not have an efficient companion software providing multiple disjoint streams and substreams with cycle division techniques.
- Matsumoto and L'Ecuyer teams joined to develop a viable jump-ahead algorithm taking some milliseconds on that time processors (2008).
- More efficient algorithms exist, like the MRG32k3a generator from L'Ecuyer which takes only a few microseconds for its jump ahead (but can be 2 times slower than MT).

Independent streams …and substreams (2/2)

- With **cycle division** a random stream can be controlled by having it jump to fixed checkpoints.
- Interesting if you may want to return to a previous state of a simulation.
- With substreams one can jump back and forth among multiple substreams.
- **Initially available for two generator types** supporting multiple independent streams and substreams:
  - The Combined Multiple Recursive MRG32k3a – 2^{124}
  - the Multiplicative Lagged Fibonacci MLFG6331_64 generators (2^{127})
- Now possible with MTs & WELLs generators (a bit less efficient)

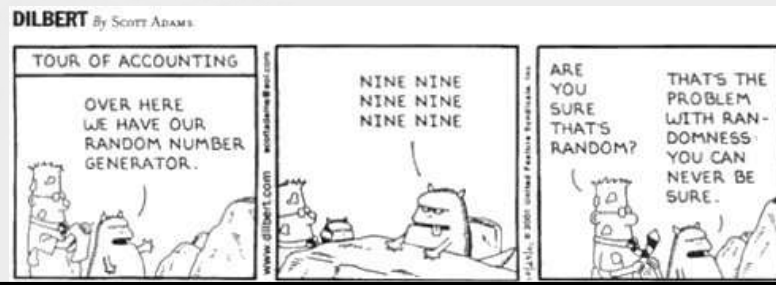
(2) Different generators - detail : DC dynamic creation of generators

- Each generator has a different ID going for instance from 1 to the number of generators needed.
- The ID of a generator is encoded in the characteristic polynomial of the MT RNG – then each generator is assumed highly independent from the other ones
- **PNRGs based on linear recurrences with characteristic polynomials relatively prime to each other are supposed to be mutually independent.**
- As it is stated one quickly understands that there is **no mathematical proof for such independence…** since we have the algorithm of the generator – we have in fact the proof of a dependence
- Currently it has been found much safer than some other parameterization approaches.

61

"Random number generators should not be chosen at random ! " (D. Knuth)

- Even if this is not the case with D.C. Matsumoto et al. warn us since 2000: Dynamic creator "**is not tested well yet, so it may contain many bugs. We are not responsible for any damage caused by these codes**."
- We have to test the produced generators

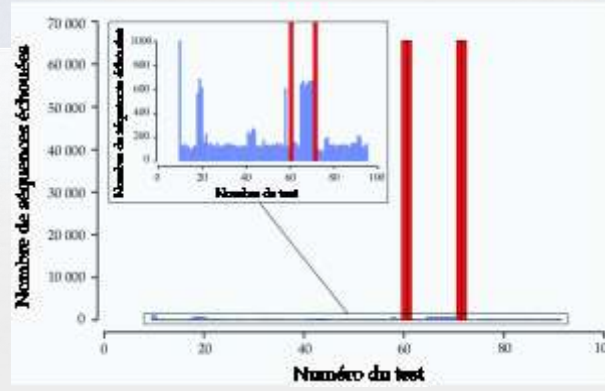


Production and test of MT generators

Credit : Reuillon - Hill

- Generation of 65536 generators
- Each generated MT has a period of $2^{512}-1$
- Test U01 – « crush » battery with 96 tests
- Identification and archival of more than 63000 good « statuses »
- Presented at the 2008 EGEE Grid user forum
- 59 years of computing
- Executed in 13 days

Results



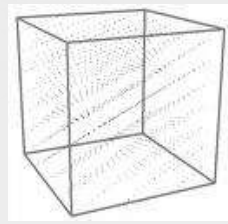
- Identification of weak MTs failing some tests
- Archival of more than 63000 good « statuses »
- MT generators do not claim to be cryptosecure and as expected each generated MT failed to the 4 cryptographic tests

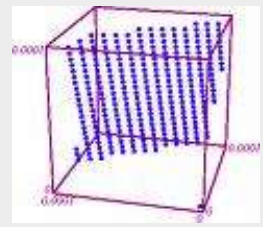
64

Credit : Reuillon - Hill

Testing Parallel Random Number Generators (1)

- A good parallel random number generator must be a good sequential generator.
- Sequential tests check for correlations within a stream, while parallel tests check for correlations between different streams.





Testing Parallel Random Number Generators (2)

Batteries of tests for sequentially generated "random" numbers

- Some tests initially proposed by Knuth
- DieHard (G. Marsaglia) – 1990s
- NIST – STS (mainly for cryptography…)
- Tests form L'Ecuyer, Matsumoto, Kurita…
- DieHarder… (R.G. Brown)
- The most complete test battery is currently **TestU01**
  - L'Ecuyer et al. 2002-2006
  - Small crush, crush and big crush with more than a hundred statistical tests

66

Testing Parallel Random Number Generators (3)

- Exponential sums
- Parallel spectral test
- Interleaved tests
- Fourier transform test
- Blocking test
- Ismay parallel tests
- …

Fourier Transform // Test

- Fill a two dimensional array with random numbers. Each row of the array is filled with random numbers from a different stream.
- Calculate the Fourier coefficients and compare with the expected values.
- This test is repeated several times and checks if there are particular coefficients that are repeatedly "bad".

Blocking // Test

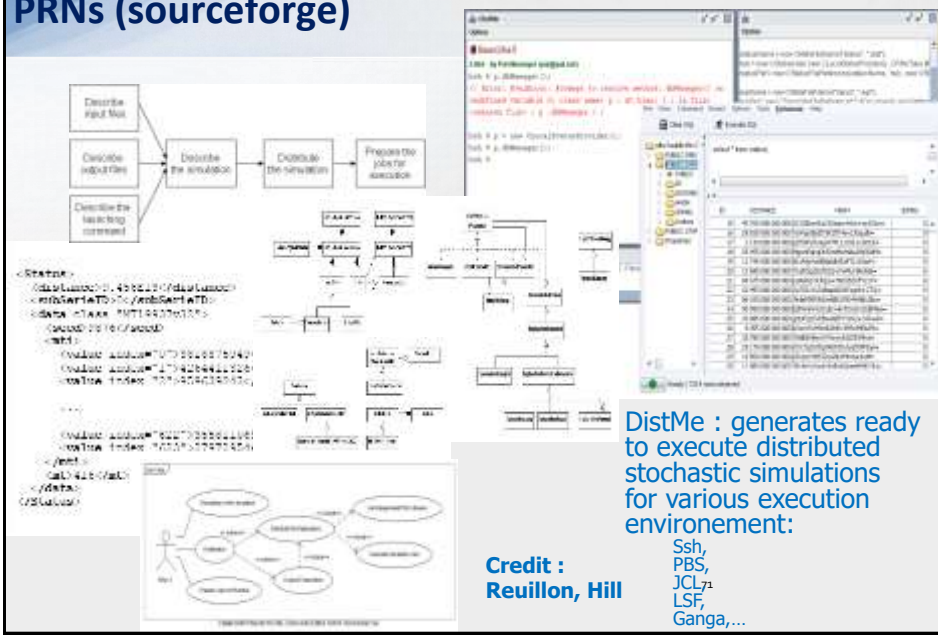
- Use the fact that the sum of independent variables asymptotically approaches the normal distribution to test for the independence of random number streams.
- Add random numbers from several stream and form a sum.
- Generate several such sums and check if their distribution is normal.

Survey of Libraries & software for PRNG and stoch. sim. paralellization

- SPRNG (Scalable Parallel Random Number Generation) library – (C/C++) Mascagni et al.
- Parallel streams and substreams in C/C++, Fortran, R, Java and Matlab (R-streams (L'Ecuyer et al. 2002-2005)
- Evolution and maintenance mainly in Java for SSJ (Stochastic Simulation in Java)
  - A very good library from L'Ecuyer et al.
- JAPARA – A Java Parallel Random Number Library for High-Performance Computing – Coddington et al (2004)
- Grid-Computing Infrastructure for Monte Carlo Applications (GCIMCA) - (Mascagni et al 2003)
- DistMe / DistRNG – Reuillon et al. > **OpenMole** (2010 +)
- Dynamic creation of MT generators (for CPU & GPU)

70

Since 2008 « DistMe » for rigorous distribution of PRNs (sourceforge)



DistMe : generates ready to execute distributed stochastic simulations for various execution environement:
Ssh,
PBS,
JCL,
LSF,
Ganga,…

Credit : Reuillon, Hill

Example of EGEE GRID Data Challenges : Applications to emerging or neglected diseases - LPC – PCSV – Blaise Pascal University

Malaria

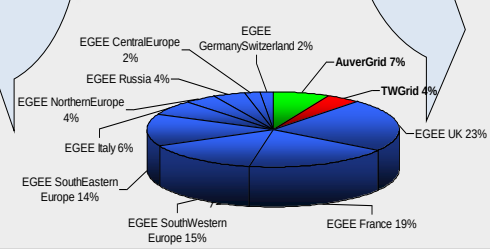
Docking input preparation

Approx. 80 years of computing Years achived in 1 month

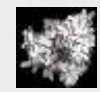
Docking output analysis

Coordination: Fraunhofer Institute

Credit N. Jacq Ph.D. thesis
Collaboration and deployment
LPC Clermont-Ferrand
V. Breton – IDG Director



Bird Flu



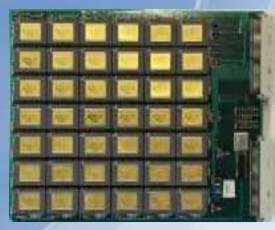
Docking input preparation

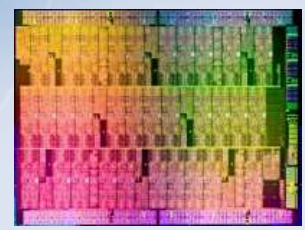
Approx. 100 years of computing years achieved in 1 month

Docking output analysis

Coordination: Sinica Academy Taiwan

Distribution of Random Streams For Parallel Stochastic Simulation



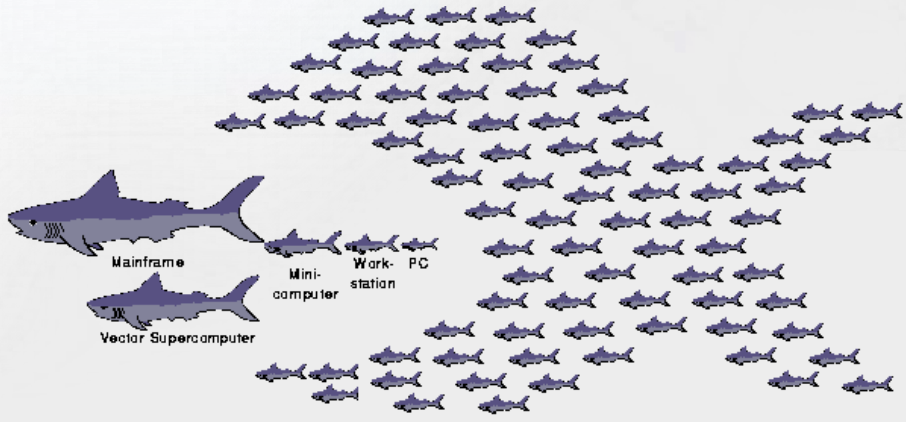


Part IV

Random streams for Hardware Accelerators - The case of GP-GPUs

Where are we going in distributed computing ?

Going « manycore » at least for the next 5 years…



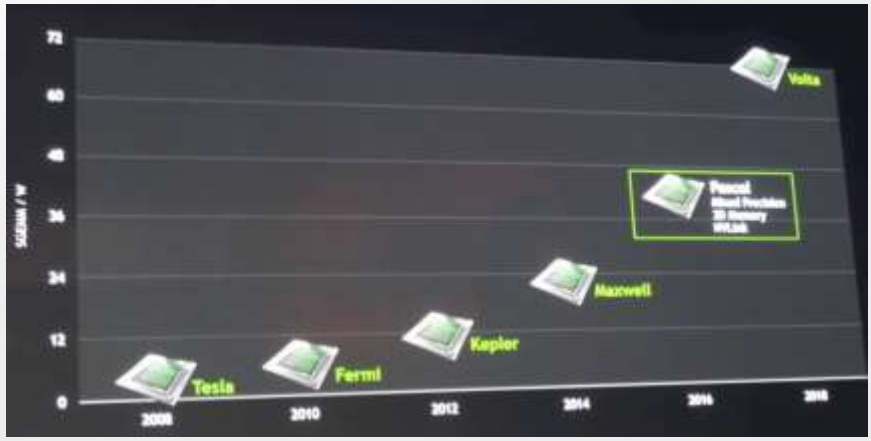
The current computing « Food Chain »…

At least going hybrid since the introduction of “personal super-computers” in 2008

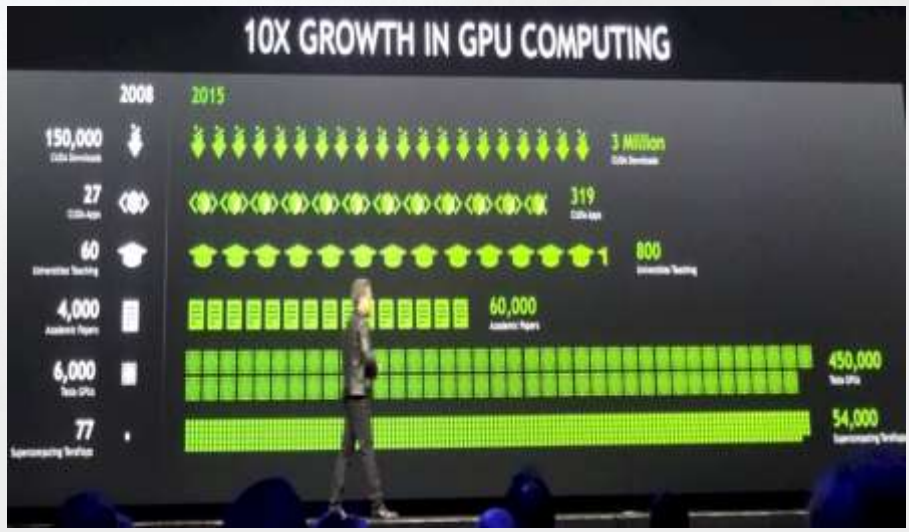
…Tesla in 2008, introduction of the Fermi architecture in 2009, Kepler in 2011, Maxwell in 2014, Pascal coming soon…

NVIDIA GP-GPU in Giga Flops per Watt

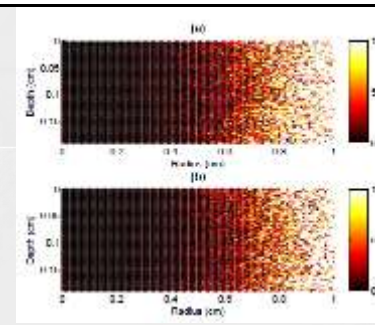
CUDA Roadmap



NVIDIA GP-GPU Key performance index



Designing a PRNG for GPU enabled platform

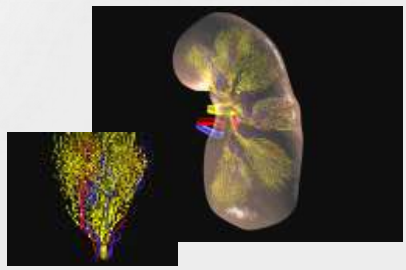


- In 2006 Sussman et al. list the limitations of GPU dedicated PRNGs due to the past weaknesses of the hardware. Limited output per thread or untruthful operations were part of the restrictions that made these PRNGs feeble for HPC apps.
- A few years ago a common solution was to generate RN on CPU before transferring them on the graphics processor. This solution has to face the well-known bottleneck of data transfer between the CPU & GPU.
- In 2008 and 2009 Langdon implemented the (old) Park and Miller algorithm on GPUs with a speedup of 4ox (Tesla / compared modern Intel processor)
- **However this generator has many known flows, though it was still in use until recently in some well distributed networking simulation software . See (Entacher and Hechenleitner 2003).**

Application example in Integrative biology

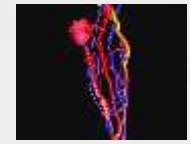
Multiscale organ morphogenesis

- Self organization
- Multi-domain knowledge:
  - anatomy, physiology,…
- Agent based simulation
- Simulation of 3D cellular spaces with GP-GPUs

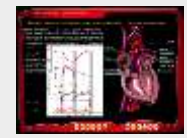


© IBC, EMCO3

Credit: Siregar, Julen, Caux, Hill



Multi-scale modeling Kidney, Heart, and … the whole body



Companies are also proposing their solutions

Modern Random Numbers Generation for GPUs
Mersenne Twister for Graphic Processors

- Proposed by Mutso Saito (main author of SFMT) in collaboration with Matsumoto - 3 November 2009
- Latest source code online last Monday (June 6th 2011)
- Implemented on CPU & GPU
- Algorithm of the MT family adapted for GPU with generation in device memory
- Preprint of the paper found online at the beginning of June 2010
- Specific version of Dynamic creator Latest version – 3rd of March 2011 (v0.3)

http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MTGP/index.html

81

MTGP - continued

- Huge period $2^{110\,503}$ (maximum)
- Each generator uses shared memory on graphic processors as its internal state space, and uses constant memory as a look-up table for a linear transformation.
- Output formats of the generator are 32-bit integers and single precision floating point numbers obeying the IEEE 754 format.
- A 64-bit integer version and double precision floating point version are also available.
- The author also proposes a parameter generator for these generator MTGP DC
- The parameter generator provides us with independent streams of pseudorandom numbers, which can be generated in parallel. (Adapted from Dynamic Creator)

82

MTGP Dynamic Creator


A VARIANT OF MERSENNE TWISTER SUITABLE FOR GRAPHIC PROCESSORS

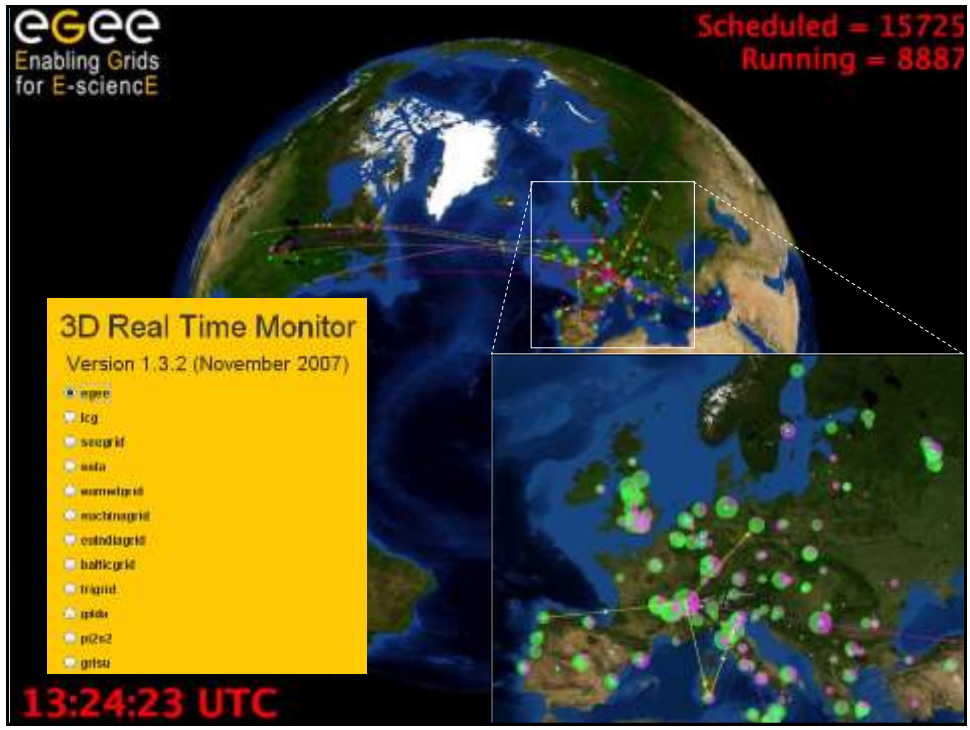
- MTGPDC can create parameter sets of MTGPs which have period of 2^{3217},2^{4423},2^{11213}, 2^{23209} or 2^{44497}.
- MTGPDC can create 2^{32} parameter sets for each period. (better than the original DC for MT 2^{16})
- MTGP DC requires some libraries :
  - Victor Shoup's NTL: A Library for doing Number Theory,
  - C++ TR1 Libraries (with g++ ver. 4.0 or later),
  - Boost C++ Libraries
  - Open SSL

83

« Quick » test MTGP DC generators

Credit :
Passerat-PalmBach – Mazel – Hill

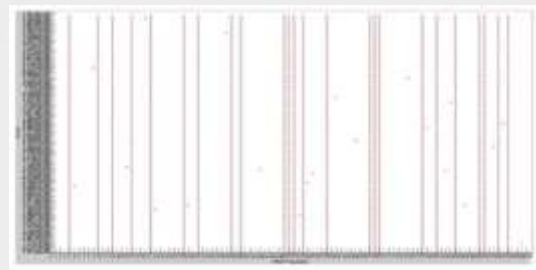
- Production of 10 000 generators in 24h (on a Nehalem processors - 5520)
- Selection of a 'small' period to have a 'small' internal state to store in GPU shared memory (larger memory imprint than MT) - 2^{3207}
- Use of Test U01 'big Crush' without the 4 cryptographic tests - 70, 71, 80 & 81
- 80 000 CPU hours (EGEE/EGI grid & local clusters)



Additional testing to check the impact of different initializations

Credit :
Passerat-PalmBach – Mazel - Hill

- Testing of the first 100 generators with 100 different random initializations
- Most generators considered ok on the first round are still ok with different initializations – a few are showing small weaknesses
- Generators considered as failing 'Big Crush' in the first test confirm this status with different initializations.

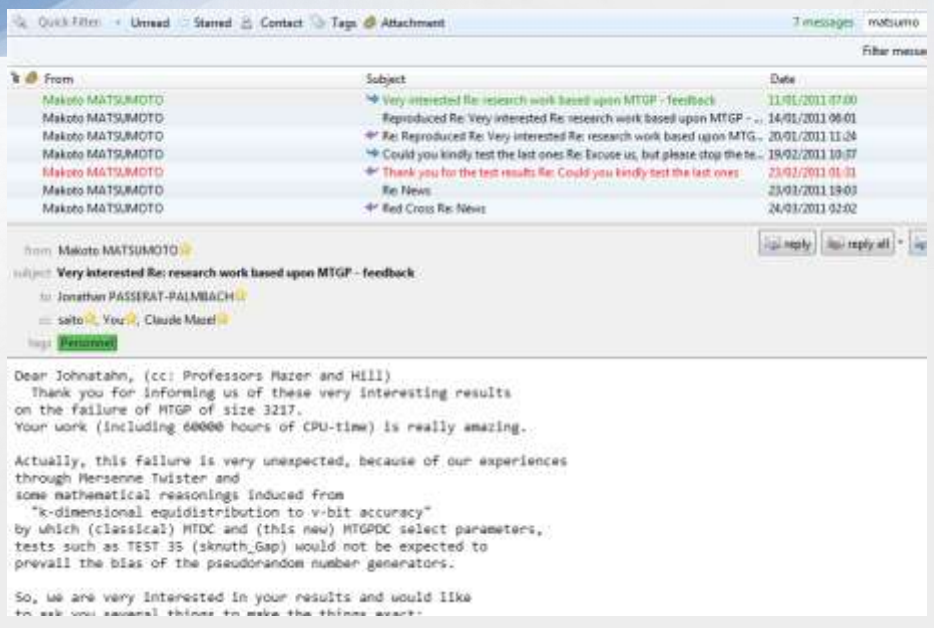


86

Preliminary results obtained
… with the preliminary versions of MTGP DC

- 33% generators failed the BigCrush tests
  - Tests are considered as failed or suspect
  - The previous chart gave proportions
  - 2 tests have systematic problems (N° 35 et 100)
    #35 is a Knuth Gap test with a specific bin size
  - 10% fail with test 100
  - 16% fail with test 35
  - Among the « failed » generators: 50% fail with test 35 and 25% with test 100.
- 66% are successful – less than what we were accustomed to with the MT family

A quick reaction from the authors…



The current version is thoroughly verified

- As soon as they were contacted Pr. Matsumoto & Dr. Saito corrected the problem (end of 2010) we warmly thank them.
- "*It was a small bug affecting only the least significant 5 bits, but it made the dimension of equidistribution of the LSBs smaller than they should be.*"
- Another recent bug has been fixed in the 64 bit version of MTGP DC mtgp64dc – It was a serious bug, and the 0.3 64 bits version should not be used.
  - All tempering parameters searched by mtgp64dc are zeros.
  - This bug does not exist in the earlier versions, and it is fixed in the last available version 0.3.1 above. (2011/6/13)

Last version, free of known bugs March, 9th 2011

Download MTGP

This code is released on March 26th in 2010, so it may contain some bugs. Any feedback is welcome (send an email to Mutsuo Saito, saito "**at sigs**" math.sci.hiroshima-u.ac.jp and m-mat "**at sigs**" math.sci.hiroshima-u.ac.jp)

version	archive	changes
0.3.1	MTGPDC-src-0.3.1.zip MTGPDC-src-0.3.1.tar.gz	Fix the bug in mtgp64dc mentioned below. (2011/3/9)
0.3	MTGPDC-src-0.3.zip MTGPDC-src-0.3.tar.gz	A small bug in search_tempar_lsqs is fixed. The bug effects only on the least significant 5 bits, but it makes the dimension of equidistribution of the LSBs smaller than they should be. In addition, we modified the algorithm of searching tempering parameters, so that the algorithm tempers the dimensions of equidistribution of LSBs too, in addition to those of MSBs. This is to address a report on some statistical tests of these LSBs due to Jonathan Passerat-Palmbach, David R.C. Hill and Claude Mazel. We thank them. (2011/3/4) The 64 bit version, mtgp64dc in this version has a serious bug, and should not be used. BUG:All tempering parameters searched by mtgp64dc are zeros. This bug exists only in this version 0.3. It does not exist in the earlier versions, and it is fixed in the version 0.3.1 above. (2011/3/9)

- More to come on Friday at 9:30 "Pseudo-random number generation on GP-GPU" by J. Passerat et al.

After « Intel inside » - « GPU inside » …
DEISA: Grid of Supercomputers

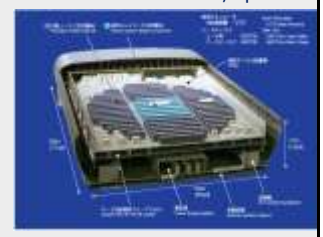
TOP 10 Systems - 11/2010

1	Tianhe-1A - NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C
2	Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz
3	Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU
4	TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows
5	Hopper - Cray XE6 12-core 2.1 GHz
6	Tera-100 - Bull bullx super-node S6010/S6030
7	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband
8	Kraken XT5 - Cray XT5-HE Opteron 6-core 2.6 GHz
9	JUGENE - Blue Gene/P Solution
10	Cielo - Cray XE6 8-core 2.4 GHz

Jülich Reseach Center
Germany

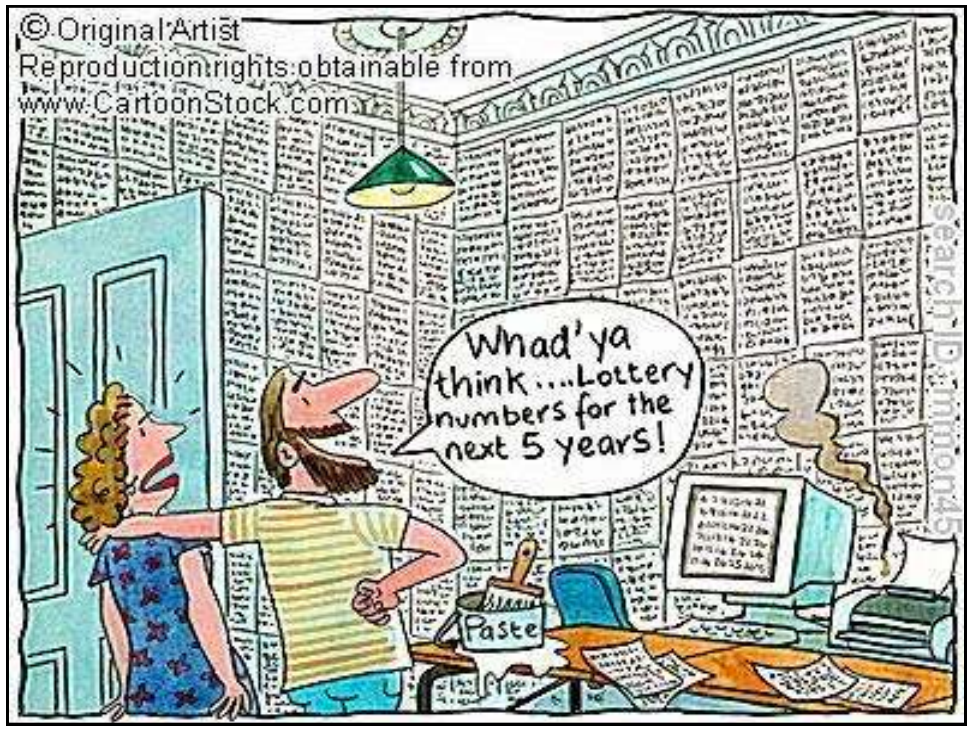
Mare nostrum
"Chapel Supercomputer"
Polytechnical University
in Barcelona, Spain

The Atomic Energy Commissariat
(Saclay – Near Paris)



Earth Simulator Center Japan

91



Conclusion

- Complex systems and stochastic models now consume more than 50% of our "supercomputing" cycles…
- A wrong/bad distribution of pseudo-random numbers to parallel processes can seriously affect your results.
- Be sure to use the best sequential generators and test the stochastic variability of your application by changing your generator
  - Ex: going from MTs, MRG32k3a, MLFG6331_64 or WELLs for instance.
- Parallel Stochastic applications should rely on a sound distribution of pseudo random numbers
  - Use existing libraries or code with reliable distribution techniques.
- When testing (TestU01 or your app.) be aware that you test a tuple
  - (generator<P>, initialization status, application, parallelization method)

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

95

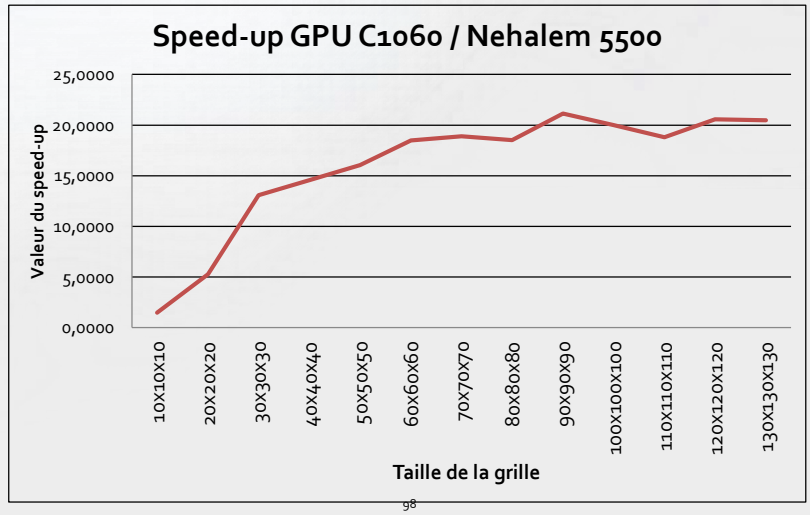
Special Thanks :
- M. Matsumoto
- M. Mascagni
- P. L'Ecuyer
- T. Priol
- M. Saito
- P. Lecuyer
- F. Panneton
-M.K. Traore
-V. Breton
-N. Jacq
-G. Michaels
-M. Mascagni
-R. Reuillon
-P. Peyret
-T. Komatsu
-S. Aubourg
-I. Buvat
-Z. ElBitar
-J. Passerat-Palmbach
- R. Buyya
- A. Duran
-X&Y Unknown
-« Google Image ☺ »
-…

Results on this example 1/2

- Speed-up croissant avec la taille de la grille utilisée jusqu'à ≈500 000 cellules (80^3).
- Selon le processeur comparé (>80^3 cellules) :
  - x40-60 par rapport au Q9300 Xeon.
  - x14-20 par rapport au Nehalem 5500.
- Impact de la copie significatif pour les petits cubes (10^3).
- Aucun gain par rapport à un CPU Nehalem pour ces petites tailles de cubes.

97

Résultats 2/2



98

solution d'une EDP harmonique approchée de manière probabiliste (la solution est calculée à partir de l'espérance de mouvements browniens). La solution est un ellipsoïde à section circulaire (la condition initiale est que la fonction vaille 0 sur le cercle unité). Dans le premier cas, par un rand(), dans le second (beaucoup plus juste comme tu peux le voir) par un Mersenne Twister