PARALLELIZATION OF MONTE CARLO SIMULATIONS AND SUBMISSION TO A GRID ENVIRONMENT

Maigne L. [(1)], Reuillon R. [(1)], Breton V. [(1)], Lazaro D. [(1)], Legré Y. [(1+5)], Donnarieix D. [(1+2)], Calvat P. [(3)], Hill D. [(4)]

(1) Laboratoire de Physique Corpusculaire, 24 avenue des Landais, 63177 Aubière Cedex, FRANCE.

(2) Unité de physique médicale, département de radiothérapie-curiethérapie du Centre anticancereux Jean Perrin, 63000 Clermont-Ferrand, FRANCE.

(3) Centre de Calcul de l'IN2P3/CNRS, 27, bd du 11 novembre 1918, 69622 Villeurbanne Cedex, FRANCE.

(4) ISIMA/LIMOS UMR CNRS 6158, Computer Science & Modeling Laboratory, Blaise Pascal University, BP. 125, 63173 Aubiere Cedex FRANCE.

(5) University of Auvergne, 24 avenue des Landais, 63177 Aubière Cedex, France.

Corresponding author :

Maigne Lydia
Laboratoire de Physique Corpusculaire, 24 avenue des Landais, 63177 Aubière cedex, FRANCE.
Tel : 04 73 40 78 49
Fax : 04 73 26 45 98
e-mail : maigne@clermont.in2p3.fr

ABSTRACT:

Monte Carlo simulations are increasingly used in medical physics. In scintigraphic imaging these simulations are used to model imaging systems and to develop and assess tomographic reconstruction algorithms and correction methods for improved image quantization. In radiotherapy-brachytherapy the goal is to evaluate accurately the dosimetry in complex phantoms and at interfaces of tissue, where analytic calculations have shown some limits. But, the main drawback of Monte Carlo simulations is their high computing time. The aim of our research is to reduce the computing time by parallelizing a simulation on geographically distributed processors. The method is based on the parallelization of the Random Number Generator (RNG) used in Monte Carlo simulations. The long serial of number used by the sequential simulation is split. Once the partitioning is done, a software application allows the user to generate automatically the files describing each simulation part. Finally, another software executes them on the DataGrid testbed using an API. All these steps have been made transparent for the user by providing a web page asking the user for all the parameters necessary to launch the simulation and retrieve results. Different tests have been done in order to show first, the reliability of the physical results obtained by concatenation of parallelized output data and secondly the time gained for jobs execution.

KEYWORDS: Monte Carlo simulations, Grid, pseudorandom numbers generator, parallelization, computing time.

Introduction:

Monte Carlo simulations are widely used in medical physics [1, 2, 3]. The physics of ionizing particles, used in medical treatments (in particularly photons and electrons interactions), is well known today. However, it is impossible to produce an analytic expression to describe the transport of particles through matter. Due to the multitude and the complexity of its interactions in matter, Monte Carlo simulations are needed to solve accurately this kind of problem.

The principle of a Monte Carlo simulation in physics, is to simulate the radiation transport knowing the probability distributions governing each interaction of particles in materials. Different possible trajectories or histories of a particle could be produced. Then, simulations store physical quantities of interest for a large number of histories to provide information on required quantities. That process involves the use of random numbers responsible for the name Monte Carlo given to this technique which was introduced during the development of the atomic energy in the post World War II era.

In the case of radiotherapy-brachytherapy problems, the goal is to calculate accurately the dose distribution for a patient. Most of the commercial systems, named TPS (Treatment Planning Systems), use an analytic calculation to determine these dose distributions and so, errors near inhomogeneities in the patient can reach 10 to 20%. Such codes are very fast (execution time below one minute to give the dose distribution of a treatment), thus allowing their usage in medical centres.

Even if the cost of computing resources is continually decreasing and makes easier consequent calculations, Monte Carlo simulations can't currently be used for clinical treatments planning for which the computing time remains too high on a single machine. So, there is a real interest for parallel and distributed Monte Carlo simulations in order to provide very accurate studies in medical physics.
A Random Number Generator (RNG) used in Monte Carlo calculations is generally made of a recursive deterministic algorithm to produce a sequence of pseudo random numbers. Because particles ranges in matter are independent of each other, parallel simulations are produced by partitioning the random sequence into independent streams. Then, we present a method to display the simulations on geographically distributed processors using a grid environment. The final outcome of the application is to provide a web portal to launch simulations in parallel.

1. Method

1.1 Applications of the Monte Carlo method in medical physics

The Monte Carlo technique can be used in a numerous of medical physics applications, we will describe here some of the main applications.

1.1.a Brachytherapy calculations

Brachytherapy is the use of encapsulated radioactive sources to treat cancer. Radioactive sources are used to deposit therapeutic doses near tumours while preserving surrounding healthy tissues. Monte Carlo is used to optimize the dose calculations around a brachytherapy source. The type of sources used (^{192}Ir, ^{32}P, ^{106}Ru sources) involves that the dose drops off rapidly with distance from the source and makes difficult an accurate knowledge of the deposit dose using experimental measurements or analytic calculations. Furthermore, Monte Carlo calculations are used to compute model-dependant parameters such as the variation of distribution dose with angle around a source (e.g the anisotropy functions).

1.1.b Modelling radiotherapy beams

Radiotherapy involves directing a beam of megavoltage x rays or electrons (occasionally protons) at a very complex object, the human body. Currently, Monte Carlo simulation techniques are the most accurate method for dose calculation in radiotherapy, in particular when radiation is transported from one medium to another. All the codes of practice (analytical codes) for absolute dose determination in radiotherapy beams now use Monte Carlo calculations to generate the stopping power ratio $s_{\text{water, air}}$ for both photons and electrons beams. Thus, everyday radiotherapy practice all over the world benefited directly from the Monte Carlo method [1, 2, 3]. But, radiotherapy treatments are becoming more complex, often requiring the dose to be calculated in three dimensions and sometimes involving the application of non-coplanar beams; Intensity Modulation Radiotherapy (IMRT) is the one of the most technologically advanced treatment methods available in external beam radiation therapy.

IMRT allows very precise external beam radiotherapy treatments. Rather than having a single large radiation beam pass through the body, with IMRT the radiation is effectively broken up into thousands of tiny pencil-thin radiation beams each of a different intensity. With millimeter accuracy, these beams enter the body from many angles and intersect on the cancer. This results in a high dosage to the tumor and a lower dose to the surrounding healthy tissues. IMRT can allow us to treat tumors to a higher dose, retreat cancers which have previously been irradiated, and safely treat tumors which are located very close to delicate organs like the eye, spinal cord, or rectum.

Such very small photon-beam fields do not exhibit charged-particle equilibrium (CPE) on the central axis [4] due to the range of secondary electrons exceeding the dimensions of the beam cross section. So, when a narrow photon beam crosses a broad low-density inhomogeneity: due to electron transport away from the central axis, which is uncompensated by transport towards the axis, the dose in the air region falls and there is then a re-buildup in the water beyond. Monte Carlo always yields the absorbed dose in whatever the medium is, whereas all the current analytical methods effectively yield equivalent water dose [5].
The ability of TPS to accurately calculate dose under a minimal range and other irradiation conditions requires evaluation. TPS use approximations in the beam model and the dose calculation (e.g, the exclusion of electron transport) to speed up the computation. This may introduce significant uncertainties in the dose distributions in a patient, especially in the presence of heterogeneities such as the air-tissue, lung-tissue and tissue-bone interfaces.
The requirements for the spatial accuracy of the correlation between dose and volumes of interest are constantly increasing. In fact, errors in the dose calculation should be kept below 2%.

1.1.c In nuclear medicine

Monte Carlo methods are extensively used in Nuclear Medicine to tackle a variety of problems that are difficult to study by an experimental or analytical approach. Such simulations should help optimizing the geometry and components of the imaging device, test and assess imaging and processing strategies.

Monte Carlo simulations can be used in internal dosimetry concerning radioactive sources incorporated to the human body. This discipline, using calculation and/or measurements, contributes to the determination of the deposited energy in a living organism due to the accidental or medical absorption of a radioactive substance. It's the custom for metabolic radiotherapy to consider cellular and tissue levels for the determination of the dosimetry.
The necessity to have realistic dosimetric data is all the more important since the metabolic radiotherapy field applies to the use of new radioactive molecules [6].
In this way, radiopharmaceutics administered to patients generally use beta emitters radionuclides. This type of radioactivity, implying a good local efficiency, imposes to know accurately the spatial distribution of the molecule because energy deposit gradients due to directly ionizing charged particles are very high. If the absorbed dose estimations don't take into account this aspect, results become erroneous [7]. Monte Carlo methods are a good alternative to this problematic. They allow the user to calculate some relevant data associated with local variations of radiopharmaceutics fixation, in order to precisely know the absorbed dose [8].

1.2 The Random Number Generator (RNG) in Monte Carlo simulations

The computing time of a Monte Carlo simulation depends of different parameters: the number of particles generated during a simulation, the medium where particles interactions occur. According to the type of material filling the medium and the type of particles generated, the number of physical interactions can vary. The number of calls to the RNG is consequently dependant on these parameters [9].

Even if the computing time of a simulation depends also on the construction of the geometry, physical processes used, etc.., the parallelization of such parameters appears to be tricky and won't be tackled in this study.

Each Monte Carlo simulation uses a sequence of random numbers to reproduce the probability of the physical interactions in matter. The more numerous the interactions in a medium are, the longer the sequence of random numbers generated for the simulation is. A simple way to reduce the execution time of a Monte Carlo simulation used in physical experiments, is to sub-divide a long or a very long simulation into little ones by indexing to each simulation a sub-sequence of random numbers obtained by partitioning a long sequence of random numbers. Sub-sequences have to be independent and as we explained before this method is valid only because the particles emitted in simulations are not dependant on each other.

1.2.a Principle

Physicists need very good random number generators for Monte Carlo calculations. Random numbers generators can be classified according to the three types of random numbers [9, 10]:

- Truly random numbers are unpredictable, they are not generated by a determinist process, and must be produced by a random physical process such as radioactive decay.
- Pseudorandom numbers are produced in the computer by a simple numerical algorithm, and are therefore not truly random.
- Quasirandom numbers are introduced to improve the accuracy of Monte Carlo integration; these numbers are not independent and thus cannot be used generally.

The main properties for a good pseudorandom number generator are:

- A good uniform distribution, e.g good randomness.
- A long period: this is the period after which a pseudorandom number generator begins to generate the same sequence of numbers over again.

Typical generators used now in Monte Carlo simulations generate pseudorandom numbers and respect the two main properties mentioned. However, a long period isn't sufficient to guarantee a good distribution. Some pseudorandom generators are based on a single integer "seed", which means that the period is limited to the number of different states that can be represented in one computer word. So, for a 32 bit computer a simple generator can have a maximum period of 2^{30}. Because it's easy to achieve this maximum, it is no longer sufficient for many present day problems. That is why the majority of generators used now, combine two or more simple generators to attain a longer period and a better distribution, this combination is usually called a shuffling of generators.

In our study, we use a Very Long Period (VLP) shuffled generator; the general technique in order to run this type of generator is:

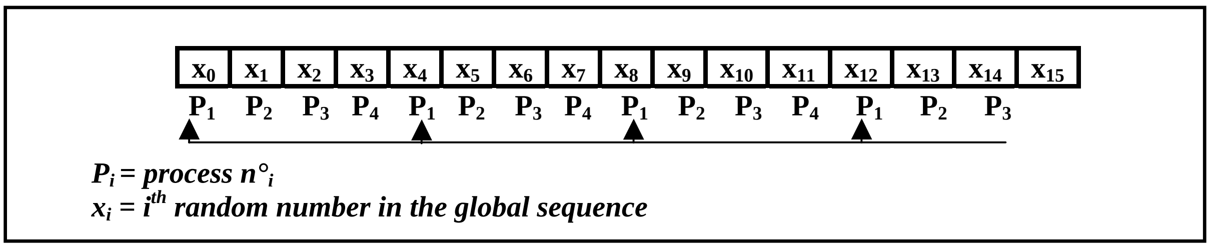
- The set up of an internal table, containing a large number of seeds (typically between 10 and a few hundreds), and the values of a few indices (typically two) pointing to seeds in the table, are also initialized.
- The generation of pseudorandom number by combining only those seeds corresponding to the current values of the indices.
- The update of the seeds just used and the pseudorandom generation of new indices to point to other seeds.

So, the sequences of random numbers generated are defined by a state (e.g seed and table) that contain all what is needed to initialize a recurrence formula which enables the generation of the random sequence.

1.2.b The parallelization of RNGs

An obvious way to get parallel random numbers streams is to partition a sequence of a given generator into suitable independent sub-sequences. This can be done in three major ways [10, 11]:

- **The Leap Frog (LF)** method (e.g., Fig1) allocates in turn the random numbers of a sequence to a partition of streams like a deck of cards dealt to card players (as illustrated in figure 1). Thus, for a partitioning of a global sequence $\{x_i, i = 0,1,2,\ldots\}$ into N parallel streams, the jth stream is $\{x_{kN+j-1}, k = 0,1,2,\ldots\}$. Given the period p of the global sequence, the period of each stream can reach p/N.



x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}
P_1 P_2 P_3 P_4 P_1 P_2 P_3 P_4 P_1 P_2 P_3 P_4 P_1 P_2 P_3
P_i = process n°$_i$
x_i = i^{th} random number in the global sequence

Fig.1. The Leap Frog method (with 4 parallel processes)

- **The Sequence Splitting (SS)** method (e.g., Fig2) splits a sequence into non overlapping contiguous blocks, as shown in figure 2. Thus, for a partitioning of a sequence $\{x_i, i= 0, 1, 2, \ldots\}$ into N streams, the jth stream is $\{x_{k+(j-1)m}, k = 0, \ldots, m-1\}$, where m is the length of each stream. One major difficulty is the determination of a good value for 'm', which must be chosen so that each stream is long enough to achieve the stochastic simulation performed by the corresponding processes.

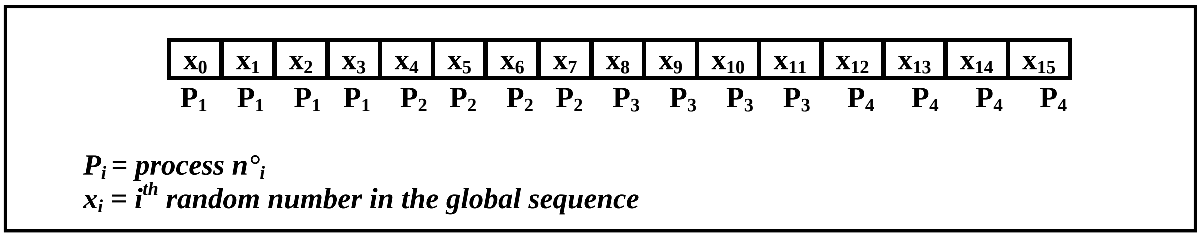

x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}
P_1 P_1 P_1 P_1 P_2 P_2 P_2 P_2 P_3 P_3 P_3 P_3 P_4 P_4 P_4 P_4
P_i = process n°$_i$
x_i = i^{th} random number in the global sequence

Fig.2. The Sequence Splitting method (with 4 parallel processes)

- **The Independent Sequences (IS)** method (e.g., Fig3) builds a partition of N streams by initializing the same generator with N different seeds. Figure 3 gives it an illustration. Such a technique can lead to overlapping streams, since a random number generated in one stream can match the seed used for another stream. Some generators exhibit several sub-cycles that can be encapsulated into other streams. Such generators seem to be well adapted to the IS method, since the seeds to use for parallel streams can be the first terms of the sub-cycles.

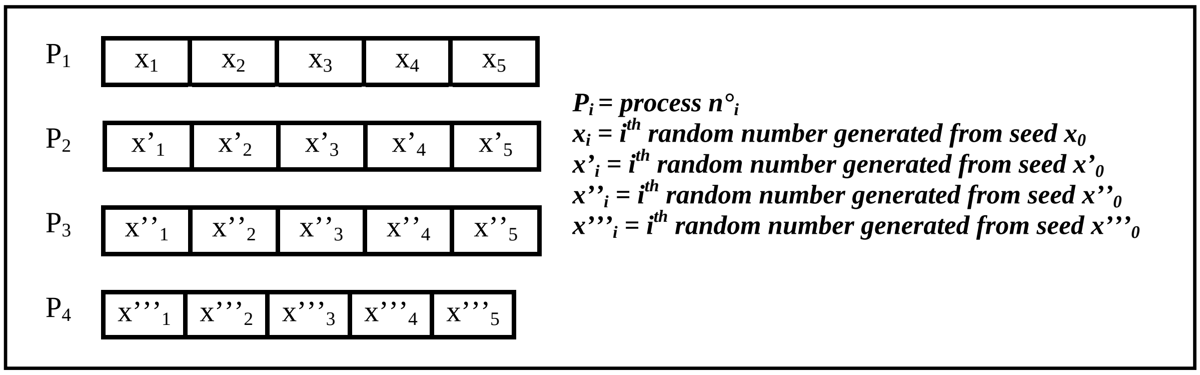

P_1 x_1 x_2 x_3 x_4 x_5
P_2 x'_1 x'_2 x'_3 x'_4 x'_5
P_3 x''_1 x''_2 x''_3 x''_4 x''_5
P_4 x'''_1 x'''_2 x'''_3 x'''_4 x'''_5
P_i = process n°$_i$
x_i = i^{th} random number generated from seed x_0
x'_i = i^{th} random number generated from seed x'_0
x''_i = i^{th} random number generated from seed x''_0
x'''_i = i^{th} random number generated from seed x'''_0

Fig.3. The Independent Sequences method (with 4 parallel processes)

We have chosen the Sequence Splitting parallelization method which is of better commodity for our study. Our simulation is based on the pseudorandom number generator developed following the algorithm of Fred James [12]. This algorithm implements the original universal random number generator as proposed by Marsaglia & Zaman in report FSU-SCRI-87-50 and coded in FORTRAN77 by Fred James. This generator is used in the CERN library routine and known as the RANMAR generator, which is also part of the MATHLIB HEP library [13]. The algorithm combines a Fibonacci sequence and an arithmetic sequence. This random number generator has been widely tested [14]; its period reaches 2^{144} and it allows the creation of 900 million different sub-sequences with each subsequence having a length of approximately 10^{30}.

1.2 Application of the parallel method

The random number generator of F. James is implemented in the HEPJamesRandom module as part of the module HEPRandom of CLHEP (available in C++) [15] which is directly employed by our Monte Carlo simulation code. The HEPRandom module consists of C++ classes implementing different random "engines" and different random "distributions". A distribution associated to an engine constitute a random "generator". The method consists in splitting a long simulation into a number of independent little ones that will use non-overlapping random number sequences generated with HEPJamesRandom.

In order to do that, the method saveStatus( ) of the HEPRandom module is employed to save in a file the current status of a random engine. Then, another method is used to launch the simulation with the file containing the status saved before. This allows the user to avoid saving the totality of the random number sequences, which would generate too large files up to hundred of Tera-octets, on the contrary the status of each sequence is very limited in size (around a few hundreds of octets).

Two possibilities are provided by our application to generate status of random numbers sequences:

- The first way consists in adapting the length of the random number sequences with the size of the corresponding simulations. This is done by estimating the consumption in random numbers for a particular simulation to parallelize. Figure 4 shows for different Monte Carlo simulations the consumption in random numbers per particle for a variable number of particles generated increasing hundred to twenty thousand particles. We can see that the random numbers per particle is stabilised for twenty thousand particles emitted.

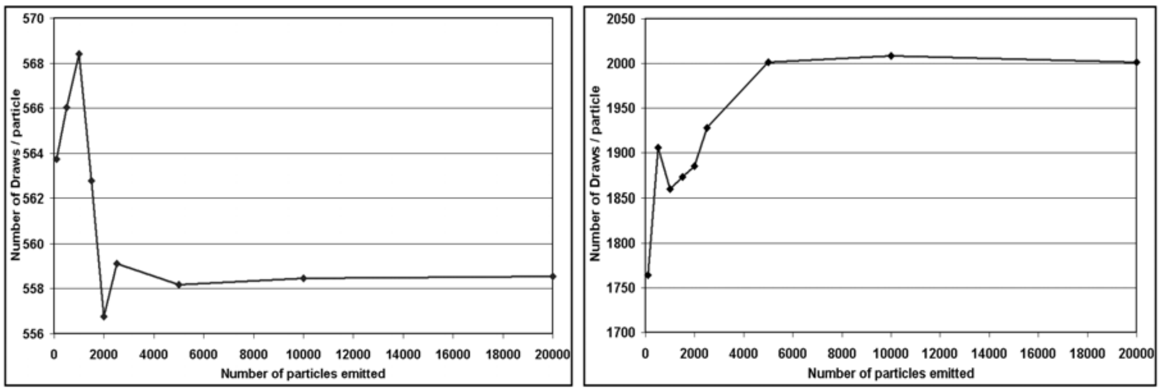


Fig.4. Consumption in random numbers per particle for different Monte Carlo simulations

- The other way consists in pre-generating hundreds of status of the random number generator with periods long enough to be sure that all the smaller Monte Carlo simulations will be able to run with. For instance, a list of two hundreds files corresponding to different status of the random number generator is generated. This method is relatively long, it takes approximately twenty days on a PIV of 1,6 GHz to generate two hundreds status of 3.10^{10} non-overlapping random numbers, but the advantage is that it has to be done only once.

In these two cases, the file corresponding to the all the status saved is loaded at the beginning of the run of each simulation. All the other parameters describing a simulation are generated automatically by the software application JobConstructor written in C++ language in a repository: the environment scripts, the macros describing the entire Monte Carlo simulation and the files required to run it on the grid. All these files will be used to describe a job. A job is a command to be run on a remote resource. For this job to run, the remote server must have the European Datagrid (EDG) software installed [16].

1.3 Sending the simulations on a grid environment

The EDG project spans three years, from 2001 to 2003. The DataGrid consists of physical resources (computers, disks and networks) and “middleware” software that ensures the access and the coordinated use of such resources. The middleware component of the project coordinates the development of the necessary software modules leveraging, where possible, existing and long tested open standard solutions such as the Globus Toolkit [17]. Once the partitioning of the RNG is done, the jobs are submitted on a grid environment using an Application Program Interface (API) written with the Java language and based on a relational database. Using EDG commands, this API allows the launching of a list of jobs on any resources where the simulation software is installed on. The computational resources are organized into a number of sites, each site provides services which are represented into different “machine types”. The machines which are used in our application are:

- Computing Elements (CE), which are the gatekeeper nodes;
- Behind them are the Worker Nodes (WNs) which are typically managed by a local batch system (BQS, PBS, …), where the user computations run.
- Storage Elements (SE) provide uniform access to large storage spaces where the output files of the simulation are stored at the end of the computation on the worker nodes.

The Java API allows the user to select geographical computing sites (CE and SE), belonging to the DataGrid computing environment, where he wants his jobs to be executed. Then, the job submission system works as a large batch system with commands to submit a job, check its status and retrieve any output. The key to the job submission and resource matching process is the job description file. This file describes the necessary inputs in order to run the simulation, the generated output and the resource requirements using the Job Description Language (JDL).

A typical JDL file has in inputs, a script describing the environment of the application, a macro describing the simulation parameters, the file where the status has been saved and the different macros necessary to the launching of the simulation. In output, the name of the output file and as requirement the CPU time. The variable CPU time fixes the limit above which the job is automatically killed by the batch system. These files are automatically generated and sequentially numbered by the software application JobConstructor which takes in input only the Monte Carlo simulation file to parallelize.

The splitting of the Monte Carlo simulations by saving the states of pseudorandom numbers generator (e.g JobConstructor application), the launching of a list of jobs on the grid environment and the retrieving of data outputs (e.g Java API) are grouped together and available for the user on a web portal. It can allow a user to parallelize a simulation without difficulty.

2. Results

All the calculations were distributed between the site of CC-IN2P3 worker nodes in Lyon and the site of Laboratoire de Physique Corpusculaire (LPC) in Clermont-Ferrand, on the DataGrid testbed. CC-IN2P3 provides 200 bi-processors mixed in PIII 750 MHz, 1GHz and PIV at 1,4 GHz ; and the LPC provides 4 PIII mono-processors at 933 MHz.

The distribution of jobs is managed by a batch manager developed at CC-IN2P3 named BQS (Batch Queuing System) and by PBS (Personal Batch System). This management takes into account the CPU time specified by the user for his jobs. This CPU time takes into account the time during which the job waits in the batch queue and the time during which the job is executed on a processor. The job is automatically killed if the CPU time is exceeded. We have to note that the shorter time the job needs for execution is, the shorter the queuing time is.

Monte Carlo simulations were performed using a new generic Monte Carlo simulation platform based on GEANT 4 (version 4.5.2.p01) and named GATE [18, 19, 20, 21, 22]. GATE (Geant4 Application for Tomographic Emission) provides new functionalities for nuclear imaging applications, among which movement and time management, it can also be used for radiotherapy-brachytherapy applications. On top of GEANT4, GATE includes specific modules that have been developed to meet specific requirements encountered in SPECT (Single Photon Emission Tomography) and PET (Positron Emission Tomography), this platform can also be used in radiotherapy-brachytherapy applications. To facilitate the description of simulation parameters such as complex detectors, phantom geometry modeling, etc..., a user-friendly mechanism based on scripts has been developed. GATE has been installed at the CC-IN2P3 site.

2.1 Comparison of the physical results

The first question to consider is whether local and parallel simulations give identical physical results. We have chosen to compare some physical data of interest for different medical physics problems. For all the problems, we have taken care to generate a sufficient number of particles in order to avoid statistical fluctuations in physical results. Three types of medical simulations were performed:

- In nuclear medicine: The simulation describes a gamma camera Philips Marconi. The two heads of the camera have been simulated, for each one a parallel collimator, a NaITl crystal, a photomultiplier and the leakage have been built. The radioactive source of 20 MBq in activity is contained in a capillary tube and emits monoenergetic photons of 140 keV (e.g ^{99m}Tc). The physical data output studied is the energy spectrum of the radioactive source of ^{99m}Tc.

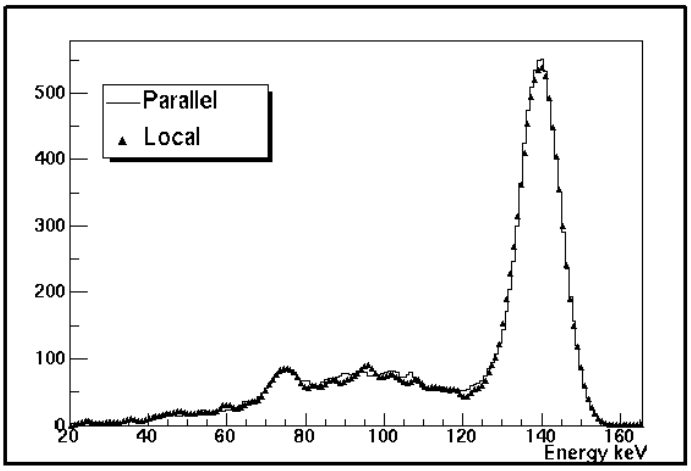


Fig.6. Energy spectrum of a capillary source of 20 MBq of ^{99m}Tc

- In brachytherapy (e.g., Fig7): The simulation describes an ocular brachytherapy treatment with ^{106}Ru/^{106}Rh ophthalmic applicators. Particles emitted followed an electron emission spectrum of mean energy 1,5 MeV. The physical data output studied is the relative deposit dose along the central axis of the eye simulated by a water sphere of 24 mm in diameter.

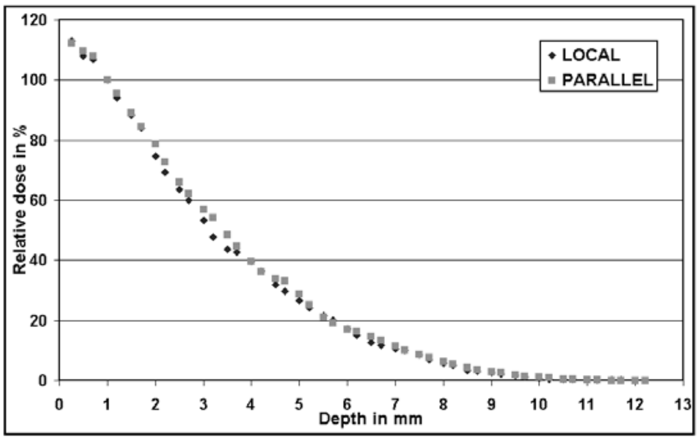


Fig.7. Comparison of relative deposit doses along the central axis of the eye using a local and a parallel computation

Results obtained for these three medical physics simulations show the good agreement between local and parallel computation. The sequence splitting method seems to be very well appropriate for parallelizing Monte Carlo simulations using the GATE platform.

2.2 Tests of computing time

With the enhanced Java API, the user is able to evaluate different computing time:

- The duration of jobs submission: the submission is realized using threads, the number of threads can be modified.
- The total computing time: launching, execution on Worker Nodes and retrieval of jobs on the user repository.

When a job is finished, it is automatically downloaded in a repository specified by the user even if all the jobs are not finished.

2.2.a Influence of the number of threads on the computing time

A *thread* is a thread of execution in a program. The Java API allows an application to have multiple threads of execution running with a pseudo-parallelism. The number of threads can be changed by the user. Figure 8 shows an example of the influence of 1, 2, 4 and 10 threads on the computing time of ten jobs launched in parallel. We can see that the more the threads are important, the lower the computing time is, but we have to note that the decrease of launching time is not proportional with the number of threads.

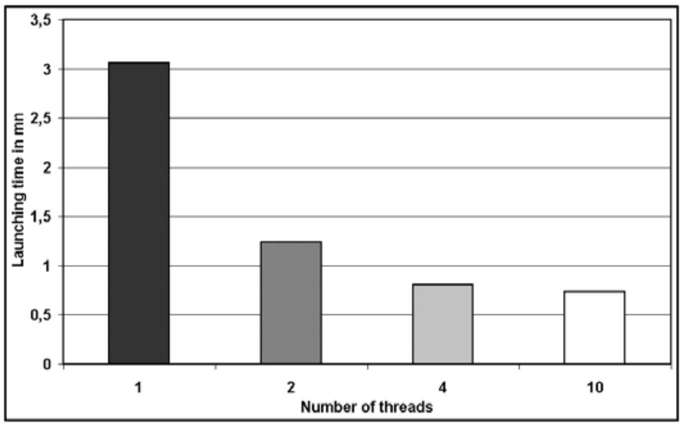


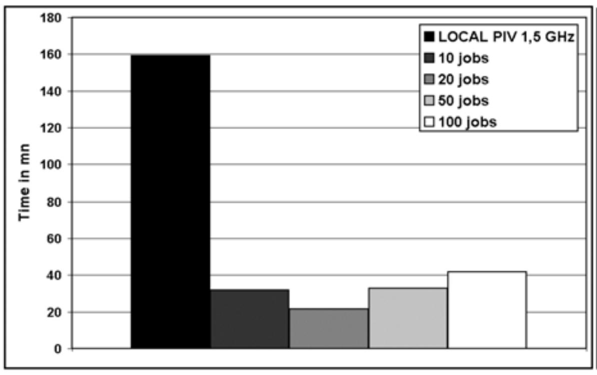
Fig.8. Influence of the number of threads in a parallel launchings

2.2.b Influence of the partitioning on computing time.

The goal is to show the advantage for the Monte Carlo simulations to partition the calculation on multiple processors. The different jobs have been launched using ten threads on CCIN2P3 in Lyon, the tests weren't repeated in duration and are representative of the processors charge at a given date.
Figure 9 illustrates the computing time in minutes of a Monte Carlo simulation running on a single processor PIV of 1,5 GHz locally and the same simulation splitting in 10, 20, 50 and 100 jobs on multiple processors. In this case, the lower computing time is obtained for 20 jobs running in parallel.
Such example shows that the computing time is not proportional with the number of jobs running in parallel. This is in part due to three parameters:

- the launching time of the jobs,
- the building of the geometry at the beginning of each simulation which do not depends of the random generation of the particles,
- the BQS managing of the jobs: even if the splitting of the simulation is very high the number of jobs waiting in the BQS depends of the CPU charge of processors.

Figure 10 shows the influence of launching time compared to the total duration of the computation. Even if the launching time is longer for a number of jobs increasing, we can see that its responsibility in the computing time is not significant.
In this study, we reach a factor 8 in the decrease of the computing time of the simulation between a computation on a processor PIV of 1,5GHz and a parallel launching by sending jobs at CCIN2P3.



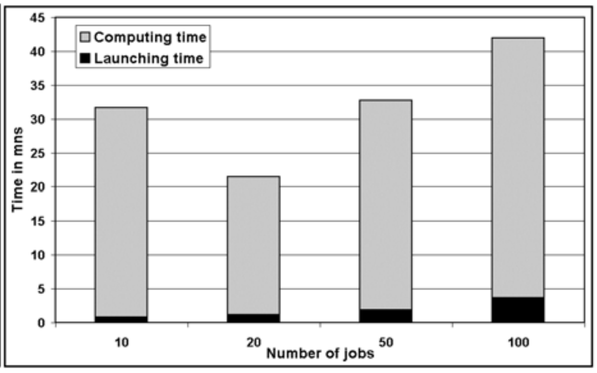


Fig.9. Comparison of computing time between a local and a parallel submission

Fig.10. Influence of the launching time in parallel launchings

Conclusion and future prospects

There have been some remarkable developments in the last few years which have taken the concept of using Monte Carlo for clinical treatment planning out of the research lab and into commercial implementations. This will become an imperative issue in order to have a sufficient precision in cancer treatments. However, the computing time using Monte Carlo calculations must stay comparable to what it is currently with analytical calculations.
It can be seen that the parallelization of Monte Carlo simulations by splitting pseudorandom numbers generator sequences in a multitude of streams is a simple, rigorous and generic (e.g it can be applied for each Monte Carlo codes used in physics with a pseudorandom number generator) method.
The comparison between a local and a parallel computation of some physical results of interest shows that the output data stay unchanged and therefore validate the parallelization method used.
The gain in computing time obtained by splitting the simulations is encouraging. Even if a parallel launching using a consequent number of threads speeds up the computing time, it is not responsible of the time consuming. The build of the geometry in a Monte Carlo simulation and especially the CPU charge of processors are the limiting factor in reducing time consuming.
In the future, in order to evaluate the CPU charge of processors, the tests of computing time have to be repeated in the meantime of one month.
Up to now, 200 status of the pseudorandom numbers generator have been generated and allow the user to parallelize a simulation in 200 jobs. With the CPU resources increasing in future grid projects, we can suppose to reach until a factor 200 in order to reduce time consuming.
Then, the development of a convivial tool to split, launch and retrieve Monte Carlo simulations on a grid environment using a web portal could answer easily to the parallelization of Monte Carlo simulations in physics.

Acknowledgements

Authors wish to thank the technical help of CC-IN2P3 and informatics service of LPC. We are also indebted to the technical coordination of the GATE collaboration and to all the technical help obtained by the coordinators of the DataGrid project.

References

1. Mackie TR Applications of the Monte Carlo method in radiotherapy. In: Kase KR, Bjärngard BE and Attix FH (eds) *The dosimetry of ionizing radiation*, Vol.III. (San Diego: Academic Press) pp. 541-620 1990.

2. Rogers DWO, Bielajew AF Monte Carlo techniques of electron and photon transport for radiation dosimetry. In Kase KR, Bjängard BE and Attix FH (eds) *The dosimetry of ionizing radiation*, Vol. III. (San Diego: Academic Press) pp. 427-539 1990.

3. Andreo P Monte Carlo techniques in medical radiation physics. Phys. Med. Biol. **36** 861-920 1991.

4. Solberg TD, Holly FE, De Salles AAF et al Implications of tissue heterogeneity for radiosurgery in head and neck tumors. Int. J. Radiat. Oncol. Biol. Phys. **32** 235-239 1995.

5. Siebers JV, Keall PJ, Nahum AE and Mohan R Converting absorbed dose to medium to absorbed dose to water for Monte Carlo based photon beam dose calculations. Phys. Med. Biol. **45** 983-995 2000.

6. Chinn PC, Leonard JE, Hanna N, Anderson DR. Preclinical evaluation of 90Y labelled anti CD20 monoclonal antibody for treatment of non Hodgkin's lymphoma. Int J Oncol. **15** 1017-1025 1999.

7. Wiseman GA, White CA, Stabin M, Dunn WL, Erwin W, Dahlbom M, Raubitschek A, Karvelis K, Schultheiss T, Witzig TE, Belanger R, Spies S, Silverman DHS, Berlfein JR, Ding E, Grillo-Lopez AJ. Phase I/II 90Y Zevalin (90 yttrium ibritumomab tiuxetan, IDEC-Y2B8) radioimmunotherapy dosimetry results in relapsed or refractory non-Hodgkin's lymphoma. Eur J Nucl Med **27** 766-77 2000.

8. Roberson PL, Heidorn DB, Besslerand ML, Haken RKT, Buchsbaum DJ. Three dimensional reconstruction of monoclonal antibody uptake in tumour and calculation of beta dose rate nonuniformity. Cancer Suppl **73** 912-8 1994.

9. Gentle J.E., Random Number Generation and Monte Carlo Methods, 2nd Edition, Statistics & Computing collec, Springer, 2003, 381 p.

10. James, A Review of Pseudorandom Number Generators, Computer Phys. Comm. **60** (1990) 329-344.

11. Traore M., Hill D., "The use of random number generation for stochastic distributed simulation: application to ecological modeling". 13th European Simulation Symposium, Marseille, October 18th-20th, 2001, pp. 555-559

12. G. Marsaglia and A. Zaman, Toward a Universal Random Number Generator, Florida State University FSU-SCRI-87-50 (1987).

13. CLHEP Reference Guide
(http://wwwinfo.cern.ch/asd/lhc++/clhep/manual/RefGuide/index.html)

14. Noell Karl-L, Weber H. Faxhhochschule Wiesbaden, W. Germany, Department of Computer Science.
http://astronomy.swin.edu.au/~pbourke/analysis/random/randomlib.c

15. CLHEP User Manual
(http://wwwinfo.cern.ch/asd/lhc++/clhep/manual/UserGuide/index.html).
(http://wwwinfo.cern.ch/asd/geant/geant4_public/Random.html).

16. Segal B., "Grid computing : the European data project", IEEE Nuclear Science Symposium and Medical Imaging Conference, Lyon, 15-20 October 2000.

17. Ian Foster and Carl Kesselman. The Globus project: a status report, Future Generation Computer Systems, Volume 15, Issues 5-6, October 1999, Pages 607-621

18. Strul D., Santin G., Lazaro D., Breton V., Morel C., "GATE (Geant4 Application for Tomographic Emission): a PET/SPECT general purpose simulation platform", to appear in Nucl. Phys. B., 2003.

19. Santin G., Strul D., Lazaro D., Simon L., Krieguer M., Vieira Martins M., Breton V., Morel C., “GATE, a Geant4-based simulation platform for PET and SPECT integrating movement and time managment”, IEEE Trans. Nucl. Sci., 50 (2003) 1516-1521.

20. Assié K., Breton V., Buvat I., Comtat C., Jan S., Krieguer M., Lazaro D., Morel C., Rey M., Santin G., Simon L., Staelens S., Strul D., Vieira JM., Van de Walle R., “Monte Carlo simulation in PET and SPECT instrumentation using GATE”, to appear in Nucl. Instr. And Methods, 2003.

21. Lazaro D., Buvat I., Loudos G., Strul D., Santin G., Giokaris N., Donnarieix D., Maigne L., Spanoudaki V., Styliaris S., Staelens S. and Breton V., “Monte Carlo simulation of a CsI(Tl) gamma camera dedicated to small animal imaging using GATE”, submitted to Phys. Med. Biol., 2003.

22. Staelens S., Strul D., Santin G., Koole M., Vandenberghe S., D’Asseler Y., Lemahieu I. and Van de Wall R., “Monte Carlo simulations of a scintillation camera using GATE: validation and application modelling”, submitted to Phys. Med. Biol., 2002.