Application of Monte Carlo method in Grid Computing and Allied Fields

Manas kumar Yogi, Vijaya kranthi chinthala

***Abstract*— This paper presents the strategy involved in Monte Carlo Method and how it impacts the functionality of many dependable application domains like Stock Markets, Nuclear reactors, Gaming Industry, Biological fields. The huge benefits outweigh the small number of challenges posed while simulation with this method. Monte Carlo methods are generally useful for tools to model empirical studies where complex input nature demands precision handling.**

***Index Terms*—Monte Carlo, Random ,Generator, simulation, Grid.**

I. Introduction

The Monte Carlo method is intensively used in various areas of scientific research. From computational physics to fluid dynamics, it has seen exponential growth with the introduction of computationally powerful computers. Truly, the Monte Carlo method is of great interest for solving systems with unknown analytical solutions. In the real world, more often than not, straight forward analytical solutions are not readily available. So, empirical modeling and numerical simulations are much helpful to better understand the physical problems involved. While in the past such modeling and numerical simulations were not very accurate, ongoing research and advances in computational power have led to more and more complex and high quality models to better estimate the physical problems. Despite that the computational power has grown exponentially over the years, this power is not able to keep up with the ever increasing demands of the improved model developed by researchers.

A. Basics of the Monte Carlo Method

The Monte Carlo method was first applied in the computation of the number π. To derive the numerical value of π, one possibility is to calculate numerically the area of a circle A_{circle} of radius r, and then to find the value of the number π using the relation $A_{circle} = \pi r^2$. To calculate the area of the circle, we start by filling up a square, of width 2r to contain the circle, with N points distributed randomly. At each point, we take note of its position with respect to the circle. If the point falls within the circle, we group it under the set C. The number of points in C is denoted by N_C. Likewise, we group all the points falling outside the circle under the set B. An approximation of the area of the circle can be calculated as the product between the ratio of the number of points in C to the total number of points in B ∪ C and the area of the square. The relation is the following:

$$A_{circle} \approx \left(\frac{N_c}{N}\right) A_{square}. \qquad (1)$$

This approximation tends towards the exact value of the area of the circle as the number of points *N* tends to infinity, i.e.,

$$A_{circle} = \lim_{N\to\infty} \left(\frac{N_c}{N}\right) A_{square} = \pi r^2. \qquad (2)$$

The value of π is then obtained with the relation $\pi = A_{circle}/r^2$.

II. Mechanism for Applying Monte Carlo Method in Grid Computing

A. Random Number Generator

The Monte Carlo method uses the stochastic landing of points in the square to find the area of the circle. This area tends towards the exact value of the area of the circle as the number of points tends towards infinity. However, in order for this method to work, the landing, or in other words the trajectories, of points has to be truly randomized. This is easily seen in the extreme case where the landing of points on the diagram is deterministic and all points land on a single point. In this case, the area cannot be calculated even if the number of points or trajectories of points tends towards infinity. However, the series of random numbers generated with software is often only truly random for a relatively small amount of generated random numbers, after which, the generated numbers loop back and the generated numbers can be predicted in a deterministic way. Furthermore, as Monte Carlo computations become more and more complex, the amount of required random numbers becomes greater and greater. Thus, standard random number generators with a maximum generation limit of random numbers can no longer fulfill the needs of such computations. In order to reduce this problem, parallel infrastructures in grid systems are used. This paper shows how to generate a sequence of random numbers that stays truly random even when the number of generated numbers is very large.

B. Sequential Random Number Generator

First, a vector S_0 containing the first p random numbers as its vector components is generated:
$S_0 = (x_{-p}, x_{-p+1}, \ldots, x_{-p+p})^T$. This is done by using standard random generators built-in with software such as Mat lab, Scilab, C++, Fortran, etc. The vector S_0 is also call the seed vector because the rest of the sequence of random numbers is generated on the basis of this vector. Secondly, we apply an additive random matrix to the seed vector in order to obtain

the next vector $(x_{-p+1}, x_{-p+2}, \ldots, x_{-p+p+1})^T$, where the last coefficient, i.e., the random number x_{-p+p+1}, has been generated. The additive random matrix is repeatedly applied to obtain sequentially all the random numbers of the expected sequence.

The additive matrix is a $p \times p$ matrix
with integer coefficients Mij having the following properties:
$Mij = 1$ if $j = i + 1$
$Mij = 1$ if $i = p$ and $j = 1$
$Mij = 1$ if $i = p$ and $j = q$
$Mij = 0$ otherwise
where q is a parameter given by the user.

C. *Parallel Random Number Generator*

The sequential method of generating random numbers can be parallelized and applied in grid networks by breaking the sequence to be generated into different segments where each segment can be generated by a different node located in the grid. For example, in order to generate 10^{11} random numbers, we can divide this sequence of 10^{11} numbers into 10^4 segments of 10^7 numbers. Each node of the grid is responsible for generating 10^7 random numbers based on a different seed vector for each node. These vectors are usually called the node seed vectors. We now state the full parallel algorithm.

1. *Generation of the node seed vectors:*

For the first node of the grid, numbered node zero, the node seed vector is obtained by the generation of random numbers using standard random number generator. This is the seed vector $S_0 = (x_{-p}, x_{-p+1}, \ldots, x_{-p+p})^T$ mentioned earlier.

For the second node, numbered node one, the node seed vector is obtained in two steps. First, the seed vector S_0 generated by the node number zero, is copied and stored locally. The additive matrix M is
then raised to the power of $1{*}10^7$ to obtain the matrix M_1. Secondly, the matrix M_1 is applied to the seed vector S_0 to generate the node seed vector $S_1 = (x_{10^7-p}, x_{10^7-p+1}, \ldots, x_{10^7}$)T . This is also called the node seed vector for node one.

For the other nodes, numbered node n, the matrix M_n is obtained by raising the matrix M to the power of $n{*}\ 10^7$ and the node seed vector for node n denoted S_n is generated by applying M_n to S_0.

2. *Generation of the sequence of random numbers:*

The matrix M is then applied repeatedly to each node seed vector to generate the random numbers in the sequence.

In this way, the node seed vector for each node is generated in parallel independently. Furthermore each node performs in parallel the generation of its own set of random numbers.

This brings us to a practical aspect of the implementation. As M is a $p \times p$ matrix, raising this matrix to the power of $n{*}10^7$ proves to be very costly in terms of computational time for large values of n and/or p. As the matrix M is highly sparse, the algorithms for such matrix computation can be optimized to reduce the computational time to construct the matrix $M_n = M_n{*}10^7$.

Nonetheless, if the matrix computation of M_n is optimized, the parallel algorithm described above proves to be more strong and stable for large random number generation.

D. *Parallel Computation of Trajectories*

In the previous section, we discussed the use of parallel characteristics of grid infrastructure to generate random numbers. Once the sequence of random numbers is obtained, the stochastic aspect of the trajectories is achieved and the simulation of the Monte Carlo trajectories can begin. As the simulation of each trajectory is independent of the others, they can also be parallelized using the grid infrastructure. In this case, the implementation is simpler than that of the parallel random generator as the number of operations to be performed is the same on each node. For instance, if each Monte Carlo trajectory requires 10^3 random numbers, each node can simulate 10^4 trajectories by using a total of 10^7 random numbers from the sequence of generated random numbers.

III. APPLICATION OF MONTE CARLO METHOD

A. *Application to Options Pricing in Computational Finance*

The Monte Carlo method can be used in computational finance, specifically in the pricing of derivatives products such as options, futures and forward contracts. In computational finance, using the Monte Carlo method requires the assumption that the movement of stock prices follows a stochastic diffusion process. This allows us to simulate a trajectory of the evolution of the stock prices upon time (hours, days, year, etc.). This simulation is then performed for N trajectories in order to get an approximate distribution of the stock prices at a certain expiry date T. This approximate distribution tends towards the normal distribution as N tends towards infinity.

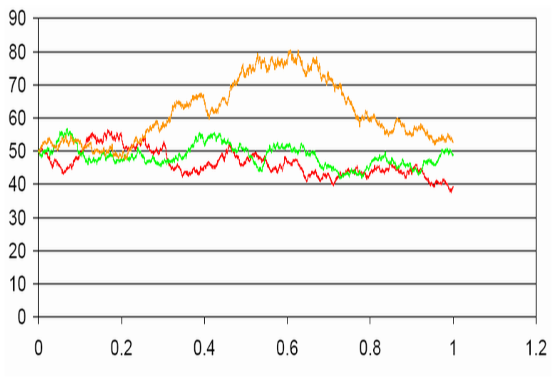

Figure 1. Simulation of stock prices.

Illustration of three trajectories for an expiry date of one year. In Fig. 1, Consider starting with a stock price S_0 equal to \$50 and we simulate the evolution of the stock price within an expiry period of one year. Each line in the diagram constitutes an independent trajectory in the Monte Carlo simulation. Each trajectory is evaluated based on the sequence of the random numbers generated. Using the stock price values obtained by all the trajectories at the expiry date T gives the expected value $E(X)$ and the possible stock price at the expiry date T. This could be done under the assumption that the distribution of the stock price at the expiry date T is a normal distribution.

With the expected value of the stock prices at the expiry date, we are able to calculate the price of the option at the initial time t_0 based on risk-free evaluation of the market and the possible price of the option at the expiry date .The fundamental notions of risk-neutral evaluation, stochastic processes, log normal distribution of stock prices are presented below.

1. Risk Neutral Evaluation:

We assume that the evaluation of the options is done in a risk-free world . In this world, the returns by investing the money in stocks are the same as that obtained by placing the money in a bank as the investors will not gain extra profits by taking extra risks in the stock market . In other words, there is no difference in the money returns between the banks and the stock market if it is a risk-free world. Thus, the gains in stocks after a certain time _t can be obtained by using the interest rates . For example, if an investor has an amount of cash $S0$ at time $t = 0$, the investor is sure to have $S0e^{r\Delta t}$ after time _t without any investment risks involved. The quantity r denotes the interest rate of the bank. We neglect the inflation rate here. In the case of a European call option, the owner exercises the option at time $t = T$ only if the price of the stock is higher than that of the strike price K and will not exercise it if the stock price is lower than that of the strike price K. Thus, the value of the option at time T will be $\max(ST - K, 0)$ where K is the price of the stock determined at time $t = 0$ and ST is the price of the stock at time $t = T$. Based on this price at time $t = T$, we can then calculate the price of the option at time $t = 0$ by discounting the option value at time $t = T$ with the risk neutral interest rate. We obtain $C0 = CT\ e^{-r(T-0)}$ where C_0 and C_T are the prices of the option at time $t = 0$ and time $t = T$ respectively.

We are now left with the evaluation of ST. If we know ST, we are able to determine CT and the evaluation of the price of option at time $t = 0$ would be mandatory. Now how can we evaluate the stock price at time $t = T$? The evolution of stock prices over time is nondeterministic. Such random behavior can be modeled by a stochastic process, which is mentioned below.

2. Stochastic Process:

The most widely used model for the evolution of stock prices over time is the following:
$dS = \mu Sdt + \sigma Sdz$ where μ is the expected rate of return and σ is the volatility of the stock prices. The variable z is assumed to be a standard Brownian process where $dz = \varepsilon\sqrt{dt}$ and ε follows a standardized normal distribution, $N(0, 1)$. The expected rate of return and the volatility are assumed to be constant in the time step dt. In the risk neutral world, the interest rate is taken as the expected rate of return. Thus, based on the price at time $t = 0$, $S0$, we are able to simulate a price $S0 + dS$ at time $t = 0+dt$ using the random process described above. This procedure is repeated until $t = T$ where we obtain a simulated price ST at time $t = T$. Such a simulation is a trajectory in the Monte Carlo method.

3. Evaluation of Stock Prices and Monte Carlo:

Based on the simulation procedure presented in the previous section, we are able to simulate many trajectories, which gives us a distribution of the stock price, ST at time $t = T$. Based on this distribution of ST, we can then calculate the expected stock price at time $t = T$ by taking the arithmetic mean of this distribution data. This is taken as the stock price, ST, at time $t = T$. However, for this method to be valid many trajectories have to be simulated and this is where grid computations become useful. As each trajectory is independent of each other, the total number of trajectories to be simulated can be carried out on different nodes in the grid. For example, if 108 trajectories are required for the distribution to be valid, it will take too long for the simulations to complete on a single computer. However, if these trajectories are distributed among 104 computational nodes on a grid, each node needs to compute only 104 trajectories of which the computational time is comparably less.

B. Monte Carlo Method And Grid Computing

Grid computing is consequently suitable for the Monte Carlo method due to the independence of the trajectories. To understand why this property is fundamental for implementation in the grid, it is important to understand that the architecture of grid networks is intrinsically parallel. The nodes, i.e., the computers of the grid are most of the time located in different countries. All these features make grid systems well suited for distributed computations on the different nodes with little communication among these nodes.

During the evaluation of the trajectories of the Monte Carlo method, there is almost no communication involved between the nodes running in parallel. Indeed, if communications were required between the nodes, the communication overhead might be too substantial to make grids a better alternative to standard computation .Furthermore, for the Monte Carlo method to work, it is advisable that a large value of the number of trajectories be evaluated. A typical value is of the scale of 10^8 or more. If these trajectories are allocated to 10^4 different nodes in the grid, the communication overhead generated between these nodes will be significant and it is likely that this will become the limiting factor for the overall computational speed.

Regarding the simulation of the trajectory, the time scale is divided into discrete time steps. These steps are quite small in order to ensure the accuracy of the prices simulated. As the time step decreases, the number of discrete points per simulation path increases. A typical value of the number of discrete points per simulation path is 10^3. Thus, large quantities of points have to be taken into account in order to implement the Monte Carlo method. Moreover, the trajectory values at two discrete points are computed through the stochastic diffusion process using one random number issued from the sequence generated earlier. This means that very many random numbers and simulations have to be carried out in order for the Monte Carlo method to be accurate. This, in turn, makes the utilization of computational grids interesting as the simulation time can be significantly reduced by using the parallel structure of grids.

C. Application To Nuclear Reactors In Computational Mechanics

In computational mechanics, the Monte Carlo method can be applied to many areas, such as the radiation of radioactive particles. Such radiation involves a highly random process.

Due to the randomness of this phenomenon, this method can be used to simulate the effect of radiation on the resistivity of the material. In the following sections, we illustrate how the Monte Carlo method can be used to simulate such a phenomenon.

1. Nuclear Reactor-Related Criticality Calculations:

In a nuclear reactor, the structure containing the reactor is often subjected to an intense amount of radiation due to nuclear fission reactions. These radioactive particles impact the surrounding mechanical structure and some areas may receive most of the radioactive impact. These areas increase the vulnerability of the structure, and thus these areas have to be reinforced in order to ensure the integrity of the surrounding structure. The simulation of the radioactive particle trajectories allows for the discovery of the points of weakness that could exist due to the deformation over time or a poor design of the structure. The radiation particles generated during the nuclear process can be considered to be random. Furthermore, the trajectories of the radioactive particles after they are produced are also highly random due to the presence of air particles that interact with this radioactive particle. Due to this phenomenon, Monte Carlo methods can be used to simulate the overall reaction and in particular the impact on the surrounding structure. Based on the properties of the surrounding environment, the radioactive particles emitted would follow a nondeterministic trajectory. Monte Carlo can be used to simulate different trajectories caused of the radioactivity in the nuclear reactor core. The inherent random property of the trajectories is provided by the random number generator discussed. Similarly, the total number of trajectories to be simulated is divided among various nodes to reduce the overall time needed. However, this example differs from the previous as it has a three dimensional environment, as compared to a one dimensional problem, which has an exercise time *T* as the boundary. Furthermore, we have also to consider the structural properties of the structure exposed to radiation. These mechanical properties have to be inculcated into the three dimensional boundary when implementing the Monte Carlo method. This is unlike the previous example where the boundary is just a vertical line. The definition of the structural properties for modeling in the grid is also beyond the scope of this book. However, we have seen here that, by running the Monte Carlo simulations, we will be able to have a distribution of the areas where the impact of the radioactive particles is the most intense. This allows us to identify potential weakness in the structure and help to prevent the compromise of the structure surrounding the nuclear reactor core.

D. Application In The Field Of Engineering

Monte Carlo methods are widely used in engineering for sensitivity analysis and quantitative probabilistic analysis in process design. The need arises from the interactive, co-linear and non-linear behavior of typical process simulations. For example,

1. In microelectronics engineering, Monte Carlo methods are applied to analyze correlated and uncorrelated variations in analog and digital integrated circuits.

2.In geostatistics and geometallurgy, Monte Carlo methods underpin the design of mineral processing flowsheets and contribute to quantitative risk analysis.

3. In wind energy yield analysis, the predicted energy output of a wind farm during its lifetime is calculated giving different levels of uncertainty (P90, P50, etc.)

4. Impacts of pollution are simulated and diesel compared with petrol.

5. In autonomous robotics, Monte Carlo localization can determine the position of a robot. It is often applied to stochastic filters such as the Kalman filter or Particle filter that forms the heart of the SLAM (Simultaneous Localization and Mapping) algorithm.

6. In telecommunications, when planning a wireless network, design must be proved to work for a wide variety of scenarios that depend mainly on the number of users, their locations and the services they want to use. Monte Carlo methods are generally used to generate these users and their states. The network performance is then evaluated and, if results are not satisfactory, the network design goes through an optimization process.

7. In reliability engineering, one can use Monte Carlo simulation to generate mean time between failures and mean time to repair for components.

E. Artificial Intelligence For Games

Monte Carlo methods have been developed into a technique called Monte-Carlo tree search that is useful for searching for the best move in a game. Possible moves are organized in a search tree and a large number of random simulations are used to estimate the long-term potential of each move. A black box simulator represents the opponent's moves. The Monte Carlo Tree Search (MCTS) method has four steps:

Starting at root node of the tree, select optimal child nodes until a leaf node is reached.

1. Expand the leaf node and choose one of its children.
2. Play a simulated game starting with that node.
3. Use the results of that simulated game to update the node and its ancestors.

The net effect, over the course of many simulated games, is that the value of a node representing a move will go up or down, hopefully corresponding to whether or not that node represents a good move. Monte Carlo Tree Search has been used successfully to play games such as go, tantrix, battleship, Havannah, and Arimaa.

F. Computational Biology

Monte Carlo methods are used in computational biology, such for as Bayesian inference in phylogeny. Biological systems such as proteins membranes, images of cancer, are being studied by means of computer simulations. The systems can be studied in the coarse-grained or ab initio frameworks depending on the desired accuracy. Computer simulations allow us to monitor the local environment of a particular molecule to see if some chemical reaction is happening for instance. We can also conduct thought experiments when the physical experiments are not feasible, for instance breaking bonds, introducing impurities at specific sites, changing the local/global structure, or introducing external fields.

IV. CONCLUSION

Monte Carlo simulation provides a number of advantages over deterministic, or "single-point estimate" analysis. Its results show not only what could happen, but how likely each outcome is. Because of the data a Monte Carlo simulation generates, it's easy to create graphs of different outcomes and their chances of occurrence. This is important for communicating findings to other stakeholders. With just a few cases, deterministic analysis makes it difficult to see which variables impact the outcome the most. In Monte Carlo simulation, it's easy to see which inputs had the biggest effect on bottom-line results. In deterministic models, it's very difficult to model different combinations of values for different inputs to see the effects of truly different scenarios. Using Monte Carlo simulation, analysts can see exactly which inputs had which values together when certain outcomes occurred. This is invaluable for pursuing further analysis. In Monte Carlo simulation, it's possible to model interdependent relationships between input variables. It's important for accuracy to represent how, in reality, when some factors goes up, others go up or down accordingly. Due to all these advantages we can conclude that Monte Carlo strategy will have a considerable effect in development of future technology.

REFERENCES

[1] Srinivas Aluru. "Lagged Fibonacci random number generators for distributed memory parallel computers". *Journal of Parallel and Distributed Computing*, 45:1–12, 1997.

[2] Russel E. Caflisch and Suneal Chaudhary. "Monte Carlo methods for american options". In *WSC'04: Proceedings of the 36th Conference on Winter Simulation*, pages 1656–1660, Washington, D.C., 2004. Winter Simulation Conference.

[3] John H. Halton. "Sequential Monte Carlo techniques for solution of linear systems". *Journal of Scientific Computing*, 9(2):213–257, 1994.

[4] Spassimir H. Paskov. "Computing high dimensional integrals with applications to finance", Technical report CUCS-023-94, Columbia University, Department of Computer Science, New York, NY 10027, USA,1994.

[5] Bernard Lapeyre and Emmanuel Temam. "Competitive Monte Carlo methods for pricing of asian options". *Journal of Computational Finance*, 5(1):39–57, 2001.

[6] Fischer Black and Myron Scholes. "The pricing of options and coporate liabilities", *The Journal of Political Economy*, 81(3):637–654, June 1973.

[7] John C. Hull. *Options, futures and other derivatives*. Prentice Hall finance series. Prentice Hall, Upper Saddle River, New Jersey 07458,5th edition, 2002.

Mr.Manas Kumar Yogi received his MTECH Degree in Computer Science Engineering from Malla Reddy College of Engineering and Technology affiliated to JNTU, Hyderabad in 2012.Currently Working in Ellenki Engineering College, Siddipet as Sr.Assistant Professor. He is having an overall experience of 7 years including Industry exposure. His area of research interest includes wireless computer networks, Artificial Intelligence, Software Engineering.

Miss.vijayakranthi chinthala pursuing MTECH (2012-2014) in Computer Science Engineering from Indur Institute of Engineering and Technology, siddipet, affiliated to JNTU, Hyderabad, Her Area of interest include wireless networks, computer networks, software engineering.