# GPU-Based Monte-Carlo Simulation for a Sea Ice Load Application

**Sara Ayubian**
Memorial University of Newfoundland
St. John's, NL, Canada
sa7818@mun.ca

**Shadi Alawneh**
C-CORE
St. John's, NL, Canada
shadi.alawneh@c-core.ca

**Jan Thijssen**
C-CORE
St. John's, NL, Canada
jan.thijssen@c-core.ca

## ABSTRACT

High Performance Computing (HPC) has recently been considerably improved, for instance General Purpose computation on Graphics Processing Units (GPGPU) has been developed to accelerate parallel computing by using hundreds of cores simultaneously. GPU computing with Compute Unified Device Architecture (CUDA) is a new approach to solve complex problems and transform the GPU into a massively parallel processor. The present study applies this new technology to a Monte-Carlo simulation for a sea ice load application. The goal of this study is to measure the performance of the GPU and Multi-GPU against the serial Central Processing Unit (CPU), parallel CPU (OpenMP), MATLAB and MATLAB (Parallel for) implementations. Results show a speedup of up to 89,000 times, and reduction in elapsed time from about 3 hours to approximately 0.1 second.

### Author Keywords

GPGPU ; CUDA; Monte-Carlo

## INTRODUCTION

The best methodology for making a significant impact on HPC is known as GPGPU. This method can be applied by using graphics processing units to crunch data and accelerate algorithms. GPUs have become a commercially attractive technology because these chips have the power to run projects that are computationally intensive. The powerful computational capabilities of GPUs stem from their vast available parallelism and result in a significant speedup compared to conventional CPUs.

The reason behind a lack of computational compatibility between the CPU and the GPU is that the GPU's architecture with large memory bandwidth is specialized for highly parallel and intensive computing. In other words, the architecture of GPUs is designed such that most of the transistors are devoted to data processing rather than data caching and flow control.

The present study demonstrates how to achieve HPC by utilizing GPU to implement a Monte-Carlo simulation for extreme level sea ice loads. The goal is to analyze this experiment in a CUDA environment and compare the performance results with regard to CPU implementations.

### Application of Monte-Carlo Simulation

Monte-Carlo simulations are a broad class of numerical algorithms for such problems where it is impractical or impossible to obtain analytical solutions, as it uses repeated sampling to determine the properties of some phenomena. Monte-Carlo simulations are ideally suited to GPU implementation and offer significant speedup over single CPU implementation [15].

The Monte-Carlo simulation, which is very robust and relatively simple to implement, has been used in this study in order to evaluate the annual maximum force between sea ice and a vertical-faced offshore structure. In other words, it uses probabilistic methods to simulate sea ice-structure interactions, rank the annual maximum loads, and plot them to lognormal scales. This method constructs probability distributions of ice environmental parameters. We have applied the GPGPU approach with CUDA programming to implement this interaction model between sea ice and a vertical structure [6].

## METHODOLOGY

This section describes the random nature of the environmental forces and how the sea ice load is calculated. There exist different types of parameters based on the sea ice characteristics and offshore structures. The present study demonstrates the interaction model between level sea ice and a vertical offshore structure. Ice thickness, floe diameter, and ice strength are defined as distributed parameters, and used to calculate the ice pressure for level sea ice in order to find the sea ice loads for each year. The width of the vertical structure is a fixed parameter.

### Sea Ice Characteristics

Sea ice is not a uniform sheet of ice, but is a complex surface and it has many characteristics that can be considered in each experiment [4]. In this experiment, the cumulative distribution function for ice thickness is defined as shown in Table 1.

In this study, floe diameter follows an exponential distribution. The mean and standard deviation for floe diameter are $\mu$ =300 (m) and Std=100 (m) with considering 10 meters and infinity as lower and upper bounds respectively. Ice strength is the third distributed parameter in this experiment. The probabilistic distribution for the sea ice strength is estimated by the

SummerSim-SCSC, 2016 July 24-27, Montreal, Quebec, Canada

| Ice Thickness | |
|---|---|
| **Bin (m)** | **Cumulative (%)** |
| 1.5 -2 | 10.2 |
| 2 -3 | 25.0 |
| 3 -4 | 56.1 |
| 4 -5 | 76.1 |
| 5 -7 | 89.1 |
| 7 -8 | 93.7 |
| 8 - 9 | 97.3 |
| 9 -10 | 98.5 |
| 10 -11 | 99.7 |
| 11 -13 | 99.9 |
| 13 -14 | 100.0 |

**Table 1**. Cumulative distribution function for ice thickness.

approach suggested in ISO 19906 [11]. In this study, the ice strength parameter " $C_R$" is defined by an uniform distribution with 1.8 and 2.8MPa as lower and upper bounds respectively.

### Interaction Model Approach

Ice interaction can create significant forces on offshore structures potentially causing damage the structure. Therefore, ice load analysis is an important aspect for designing a marine structure in icy waters, which is the application of this paper. The interaction between level ice and a vertical structure looks straightforward but actually is not [10]. Generally, when the ice interacts with the structure, the ice will fail and it may crush if the ice is thick enough [13]. More than one approach is possible to estimate the characteristics of ice loads, and there are arguments in favour of all of them. In this experiment, one of the common approaches is used as suggested in ISO 19906. Ice thickness, floe diameter and ice strength are distributed parameters which follow specific distributions as shown in Table 2. The floe encounter rate is defined as a gamma distribution with mean and standard deviation $\mu$ =50 and Std=30, considering 10 and 150 as lower and upper bounds respectively. The final results are based on ranking the annual maximum forces and determining the load associated with a specified probability.

The global average pressure p, as used for this experiment, is as follows:

$$p = C_R(\frac{h}{h^*})^n(\frac{W_s}{h})^m$$

Where

- p : The global average ice pressure (MPa).
- h : The ice thickness of the ice sheet (m).
- $W_s$ : The contact width (m).
- m : An empirical coefficient of -0.16.
- n : An empirical coefficient of -0.5+0.2h if $h < 1$ (m) and -0.3 if $h \geq 1$ (m).
- $C_R$ : Ice strength coefficient (MPa).
- $h^*$ : A coefficient is equal to 1 (m).

At the start of each year, the number of impacts are specified, and for each impact the relevant parameters and resulting impact loads are determined. The crushing force at each interaction between level ice and the vertical sided structure is determined as:

$$F = p \times A_c$$

Where

$$A_c = W_s \times h$$

### Monte-Carlo Simulation Framework

Many numerical problems in science, engineering, finance and statistics are modeled through Monte-Carlo simulations. The study of Monte-Carlo techniques require knowledge in a wide range of fields, for instance probabilistics to describe the random process, statistics to analyze the data, computational science to efficiently implement the algorithms, and mathematical programming to formulate and solve the problem of interest [14].

Monte-Carlo simulation is a very robust and relatively simple method to implement. However, using Monte-Carlo simulation requires a long execution time, though it is decreasing as computers become more powerful [20].

For each year and impact, we track and store annual maximum load with associated parameters as also shown in Figure 1.

This experiment is configured to facilitate the use of Monte-Carlo simulation to determine the design load using the following steps as shown in Figure 2:

- Specify an interaction model corresponding to level sea ice.
- Define probability distributions for the environmental inputs.
- Determine the maximum annual force on the structure.
- Perform this experiment for a sufficient number of years to achieve stable results.
- Rank the annual maximum forces.

Based on Figure 2, the calculations of the ice loads on the structure are independent for each year. Therefore, one thread can be assigned for each year to calculcate the ice load on the structure.

### Cuda

In November 2006, NVIDIA had an opportunity to bring GPUs into the mainstream by bringing a programming interface, which it dubbed CUDA. It was an attempt to make programming environment on GPUs more accessible to programmers. CUDA interface uses standard C language to implement an algorithm on GPU without having any knowledge about graphics programming using OpenGL, DirectX, and shading language. CUDA has produced great progress in the computer software industry by moving from serial to parallel programming [7]. It can take a simple model of data parallelism into a programming model without the need for

| Parameter | Symbol | Unit | Value or Distribution Type |
|---|---|---|---|
| Contact width | $W_s$ | m | 80 |
| Ice thickness | h | m | User-defined |
| Ice strength | $C_R$ | MPa | Uniform distribution |
| Floe diameter | | m | Exponential distribution |
| Floe encounter rate | | | Gamma distribution |

**Table 2**. Input parameters for Monte-Carlo simulation

graphics primitives. In other words, The CUDA environment makes the GPU look like another programmable device.

CUDA contains some libraries, which are not different than system libraries or user-built libraries. They can refer to a set of functions' definitions whose signatures are exposed through header files. The CUDA libraries are special in that all computation implemented in the library is accelerated using a GPU, as opposed to a CPU. There exist shared features and concepts in many CUDA libraries which can be called from a host application. This scenario demonstrated the application of the two most important libraries in CUDA [16].

As far as we know, the critical part of many scientific, and functional applications is random number generation. CUDA provides a library, CURAND, which can focus on the efficient generation of high quality pseudo-random and quasi-random numbers. CURAND is equipped by library on the host (CPU) side and a device (GPU) header file. These random numbers can be generated on the device or on the host CPU.

For device generation of random numbers, the actual work occurs on the device and the result would be stored in global memory on the device. The user can copy random numbers back to the host for further processing or call their own kernels to use the random numbers. However, for host CPU generation, all of the works are done on the host, and the random numbers would be stored in host memory [17].

The next library, Thrust, is a powerful library of parallel algorithms and data structures. It has a leverage to implement high performance application with minimal programming effort. Thrust provides a flexible, high-level interface for GPU programming that enhances developer productivity and the robustness of CUDA applications. Using Thrust, the programmer can write just a few lines of code to perform operations faster than the latest multi-core CPUs [12].

CUDA libraries and intensive GPU-accelerated applications are available from NVIDIA and the use of those libraries is a key area where you can obtain some serious productivity gains, as well as execution time and a significant improvement in speedup.

## PROCEDURE OF THE EXPERIMENT

The goal of this study is to find maximum annual force between sea ice and a vertical sided structure. To gain this result, we define fixed and distributed parameters in order to calculate the forces in each year. A fixed parameter will be the width of the vertical structure that has interaction with sea ice, and distributed parameters are ice thickness, floe diameter, and ice strength. Those distributed parameters except ice thickness follow specific probabilistic distributions.

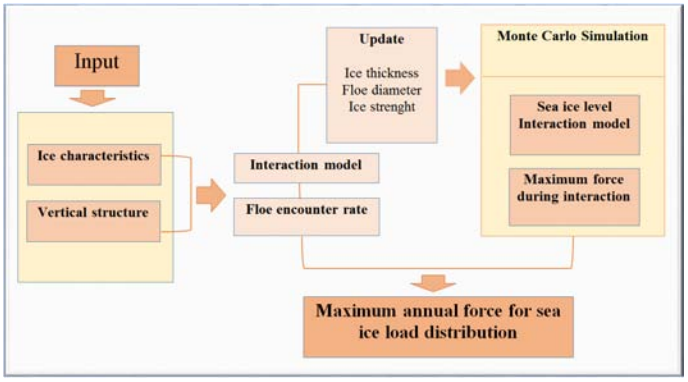


**Figure 1**. Probabilistic framework of load characteristic

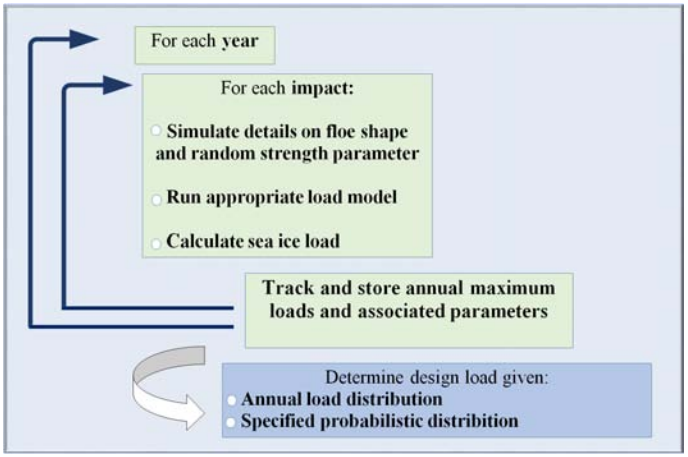


**Figure 2**. Monte-Carlo simulation framework

Therefore, we need to use CURAND libraries on CUDA environment to generate random numbers based on those probabilistic distributions and using Monte-Carlo simulation for interaction with sea ice. As it is mentioned in methodology, sea ice and an offshore structure can produce pressure with regards to an interaction, and then it is possible to calculate force in each year and rank the maximum force as an output.

The present scenario examined the interaction model in 5 cases which started from 10,000; 50,000; 100,000; 500,000; and 1,000,000 years. The simulation has been done by CUDA programming using GPUs and Multi-GPU to compare the performance of the GPU and Multi-GPU against the serial CPU, parallel CPU (OpenMP), MATLAB and MATLAB (Parfor) implementations.

## RESULTS

| Card type | Approximate speedup |
|---|---|
| Quadro K4100M | 3,000 |
| Tesla K80 | 12,000 |
| GeForce GTX TITAN Black | 14,000 |
| 4 Tesla K80 | 89,000 |

**Table 3**. Approximate speedup of different GPUs over MATLAB implementation using a signle CPU core

### Ice Loads

The methodology used in the present study results in a load-exceedance curve, which can be used to determine maximum loads at a desired annual exceedance probability level. One can think of the 100 year period load as having $10^{-2}$ annual probability of exceedance over the design of the structure. The goal of this study is to find the load on the structure associated with an annual probability of $10^{-2}$ as shown in Figure 3.

### Performance Results

The simulation of the sea ice interaction with a vertical structure is a good fit for the GPUs implementation. In order to have high performance results, GPUs require mapping and optimization based on their significantly different architectures from CPUs. GPUs have high parallel throughput and high memory bandwidth, which enable them to work with multiple processor configurations [7].

This study developed a model of the sea ice load to determine the characteristic loads using Monte-Carlo simulation. For each interaction, it simulates and runs the appropriate model to find the maximum ice load by generating random values from the input distributions.

This implementation of interaction model using CUDA programming on GPU and Multi-GPU has advantages over the serial CPU, parallel CPU (OpenMP), MATLAB and MATLAB (Parfor) implementations. Results tested on different types of GPUs such as GeForce GTX TITAN Black, Quadro K4100M, Tesla K80, and a Multi-GPU (4 Tesla K80). Those GPU implementations are used to calculate performance results over different CPU implementations. This work uses two Intel(R) Xeon(R) CPU E5-2620 @2.10GHz.

Figure 5, 6, 7 and 8 show an approximate speedup of up to 3,000; 12,000; 14,000 and 89,000 over MATLAB implementation using a single CPU core as shown in Table 3.

Figure 4 shows the elapsed time of CPU, parallel CPU (OpenMP), MATLAB, MATLAB (Parfor), and using three different GPU cards specified as GeForce GTX TITAN Black, Quadro K4100M, Tesla K80, and a Multi-GPU (4 Tesla K80). As we see in Figure 4 the elapsed time is reduced from about 3 hours to approximately 0.1 second. Therefore, the Multi-GPU (4 Tesla K80) is the fastest implementation in this scenario.

## RELATED WORK

Nowadays, improvements in the programmability and computing power of GPUs made an acceleration, for time consuming simulation. There are many works that had been done to accelerate the speedup of GPUs over CPUs implementations. Recently, a paper has been published to show the performance benefits of GPGPU for sea ice forecasting by using fast quadratic discriminant analysis [7].

The present study demonstrated a significant speedup of a standard code for Monte-Carlo simulation of sea ice load on a GPU by using CUDA programming over CPU implementations. Also, there is another article related to GPU-accelerated Monte-Carlo simulation of Brownian motors dynamics with CUDA that brought speedup of about 3,000 compared to typical CPU. The significant speedup that came after using different types of GPUs expands the range of problems solvable by using probabilistic simulations [18].

The Monte-Carlo simulations as a broad class of computational algorithms are used in many different areas [8].

The question is why we apply Monte-Carlo simulation for those different areas? It is apparent that we use Monte-Carlo simulation when we have some applications with uncertainty in inputs and for high dimensional problems with many degree of freedom. Therefore, it needs to discover the stages that we are supposed to use to improve performance in Monte-Carlo simulation [3].

A typical Monte-Carlo system consists of four stages such as random number generation, path generation, payoff function, and statistical aggregation. It is indisputable to use parallel constructs in order to design Monte-Carlo simulation in CUDA environment [2].

In order of using CURAND library in the present study, we used Thrust library which is the powerful abstraction tool and an efficient way of performing Monte-Carlo on GPUs. One of the popular example of using Thrust library on CUDA is the estimation of the value of the constant **PI** by using Monte-Carlo simulation on GPUs [1].

The direct simulation Monte-Carlo (DSMC) is a proven technique that take advantage of the computational performance of GPUs to simulate rarefied flows where real gas effects by internal relaxation and chemical reactions. This method achieved high performance which make that applicable by partially alleviating the main limitation of long computational run-times [9]

Another application of Monte-Carlo simulation by using CUDA programming on GPU is the evaluation of light-skin diffuse reflectance spectra for Multi-Layered Media. The speedup for this case is 71.19 and it varies across the wavelengths [21]

Ice engineers who work with different types of environmental inputs, estimate probabilistic distributions of sea ice parameters in order to simulate the interaction models with offshore structures. Also, they applied Monte-Carlo simulations on design of many models, based on the impact forces between sea ice and offshore structures. They did not have any implementations for sea ice load by using Monte-Carlo simulation on GPUs [19].

Sometimes, programmers who use GPUs to implement their algorithms, achieve speedups of order of magnitude versus efficient CPU implementations. GPUs developed as a dedicated chips to aid high performance computing in parallel. The reason for this advantage, is the bandwidth and computational horsepower of recent GPUs architectures. This development in GPUs computation have permitted the simulating of massive event set in timely and practical way. For instance, research has been done to demonstrate performance benefit of GPGPUs for simulating the complex mechanics of a ship operating in pack ice and it proved that GPUs have the potential to reduce the computational time significantly [6].

There is another study that has been done on the utility of graphics cards to perform massively parallel simulations of advanced Monte-Carlo simulation. This implementation worked with a set of stochastic simulation examples including population-based Markov chain Monte-Carlo simulations on GPU and it brought speedup from 35 to 500 over CPUs implementations [15].

The result indicated that GPUs and Multi-GPUs have high potential to facilitate the algorithm to access many-core computations, and motivate broad use of parallel simulation methods in order to reduce the computational time and offer significant speedup over CPU implementations.

## CONCLUSION

In this study, GPU performance benefits are discussed for a Monte-Carlo simulation of sea ice loads. Specific ice parameters were selected for demonstrative purposes, which allow the engineering designers to have a better understanding of those factors. The speedups attainable with different types of GPUs for sea ice load by using Monte-Carlo simulation is tremendous, and the elapsed time is reduced signifantly. It should be mentioned that while we have used CUDA to implement the parallel algorithm, the results are not specific to this method or to GPUs. There exist many-core processor markets with different devices and architectures, which take advantage of this improvement [15].

## FUTURE WORKS

The present study only focused on one scenario which was the interaction between level ice and a vertical structure, but there exist many other scenarios that worth to work in order to achieve development and optimization. For those who might be interested to find an experiment to work on, it is helpful to be familiar with different types of sea ice and explore an efficient way such as Monte-Carlo simulation to implement different interaction models.

In this experiment, memory limited us to work on more data. Then, finding a way to manage the memory would be helpful. Sometimes, the operating system on the co-processor allows us to allocate more memory space than is physical available. Typically, it is time consuming to start your code from beginning to optimize your implementation. Therefore, one of the solutions might be to use another processor that enables significant performance gains for highly parallel code, for instance the Intel Xeon Phi coprocessor might be suitable. Now, you can think “reuse” rather than “recode”. This type of co-processor optimized to be the right choice for highly parallel workloads [5].

Based on what researches have been done in the area of GPU computing and what we discussed in the related works section, there are many ways to improve high performance computing. For instance, using different types of GPUs brought significant speedup in some stochastic differential equations. Hopefully, with the help of this work, future studies which are related to Monte-Carlo simulation on GPUs would become much easier to work with and it will open a completely new chapter in the history of high performance computing [18].

## REFERENCES


1. Cuda-accelerated monte-carlo for hpc. http://www.nvidia.com/docs/IO/116711/sc11-montecarlo.pdf.
2. Monte carlo methods in cuda. http://www.thalesians.com.
3. Monte carlo simulation and its efficient implementation. http://www.nag.com/Market.
4. sea ice.retrieved from. http://www.eoearth.org/view/article/155931 (2012).
5. What is the intel xeon phi coprocessor? http://www.intel.com/content/www/us.
6. Alawneh, S., Dragt, R., Peters, D., Daley, C., and Bruneau, S. Hyper-real-time ice simulation and modeling using gpgpu. *Computers, IEEE Transactions on 64*, 12 (2015), 3475–3487.
7. Alawneh, S., Howell, C., and Richard, M. Fast quadratic discriminant analysis using gpgpu for sea ice forecasting. In *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, IEEE (2015), 1585–1590.
8. Cullinan, C., Wyant, C., Frattesi, T., and Huang, X. Computing performance benchmarks among cpu, gpu, and fpga.
9. Goldsworthy, M. A gpu–cuda based direct simulation monte carlo algorithm for real gas flows. *Computers & Fluids 94* (2014), 58–68.
10. Gürtner, A. Experimental and numerical investigations of ice-structure interaction.
11. Hauge, M., et al. Arctic offshore materials and platform winterisation. In *The Twenty-second International Offshore and Polar Engineering Conference*, International Society of Offshore and Polar Engineers (2012).
12. Hoberock, J., and Bell, N. Thrust: a parallel template library. version 1.3. 0, 2010.

13. Ibrahim, R., Chalhoub, N., and Falzarano, J. Interaction of ships and ocean structures with ice loads and stochastic ocean waves. *Applied Mechanics Reviews 60*, 5 (2007), 246–289.

14. Kroese, D. P., Brereton, T., Taimre, T., and Botev, Z. I. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics 6*, 6 (2014), 386–392.

15. Lee, A., Yau, C., Giles, M. B., Doucet, A., and Holmes, C. C. On the utility of graphics cards to perform massively parallel simulation of advanced monte carlo methods. *Journal of Computational and Graphical Statistics 19*, 4 (2010), 769–789.

16. Nvidia, C. Programming guide, 2008.

17. Nvidia, C. Curand library. *NVIDIA Corporation, Santa Clara, CA* (2010).

18. Spiechowicz, J., Kostur, M., and Machura, L. Gpu accelerated monte carlo simulation of brownian motors dynamics with cuda. *Computer Physics Communications 191* (2015), 140–149.

19. Thijssen, J., Fuglem, M., Muggeridge, K., Morrison, T., Spencer, P., et al. Update on probabilistic assessment of multi-year sea ice loads on vertical-faced structures. In *OTC Arctic Technology Conference*, Offshore Technology Conference (2015).

20. Veach, E. *Robust monte carlo methods for light transport simulation*. PhD thesis, Stanford University, 1997.

21. Yusoff, M. S., and Jaafar, M. Performance of cuda gpu in monte carlo simulation of light-skin diffuse reflectance spectra. In *Biomedical Engineering and Sciences (IECBES), 2012 IEEE EMBS Conference on*, IEEE (2012), 264–269.

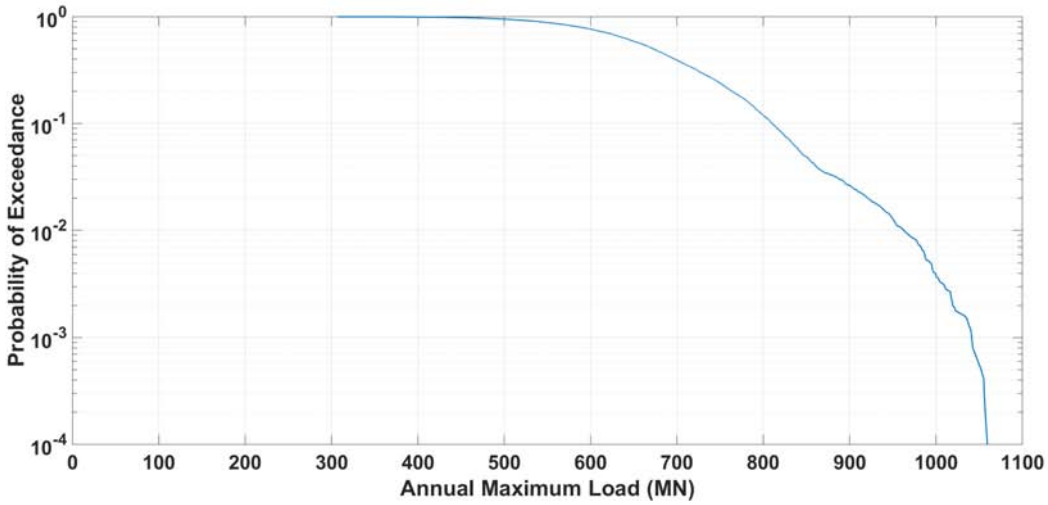


**Figure 3**. Ice load results from Monte-Carlo simulation

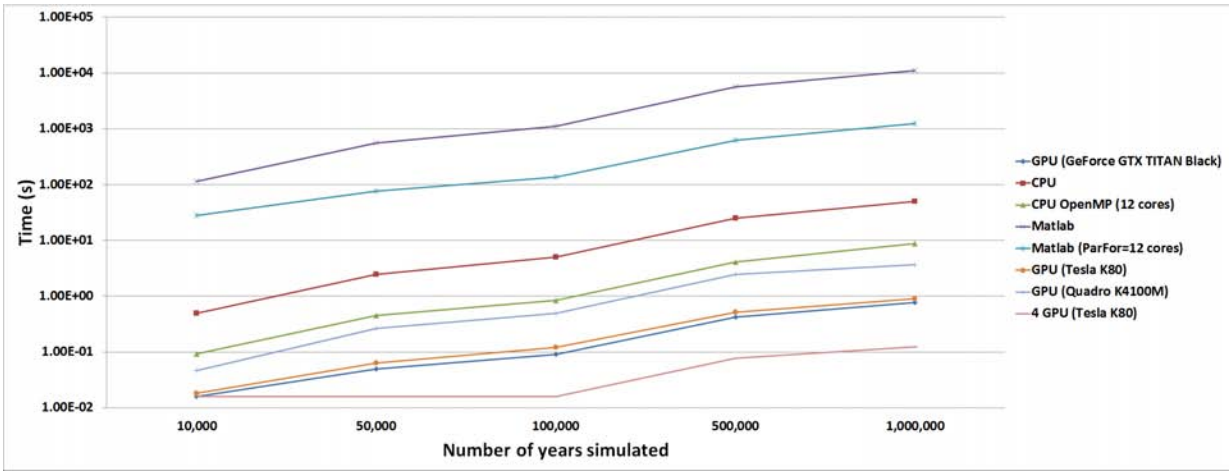


**Figure 4**. Elapsed time of the different Monte-Carlo implementations

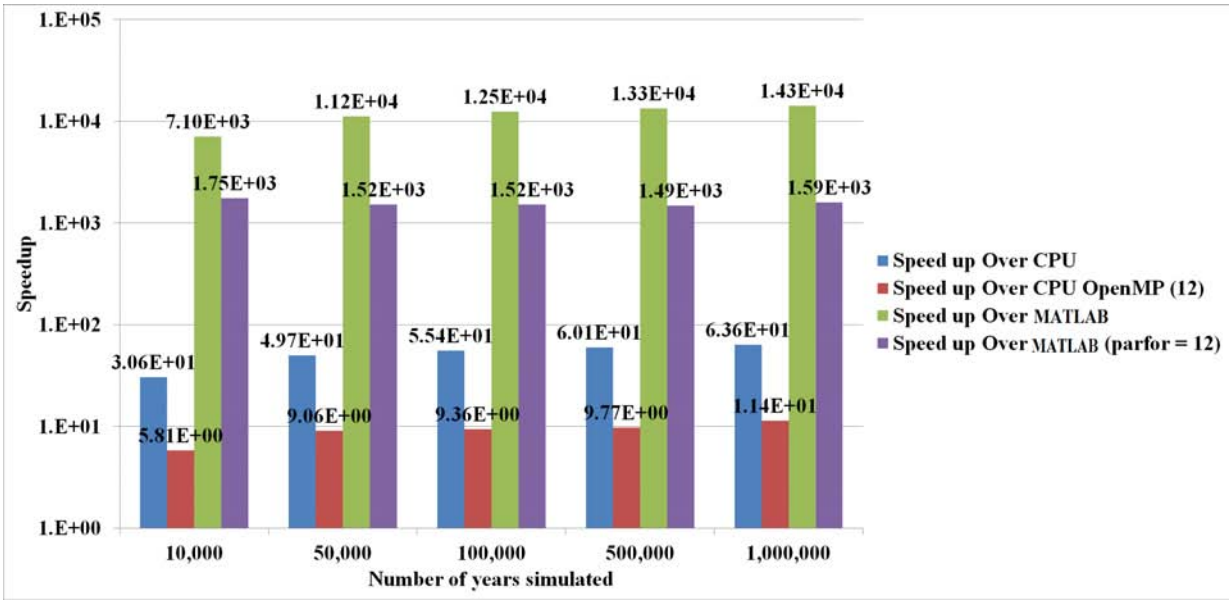


**Figure 5**. Speedup of the GPU (GeForce GTX TITAN Black) implementation

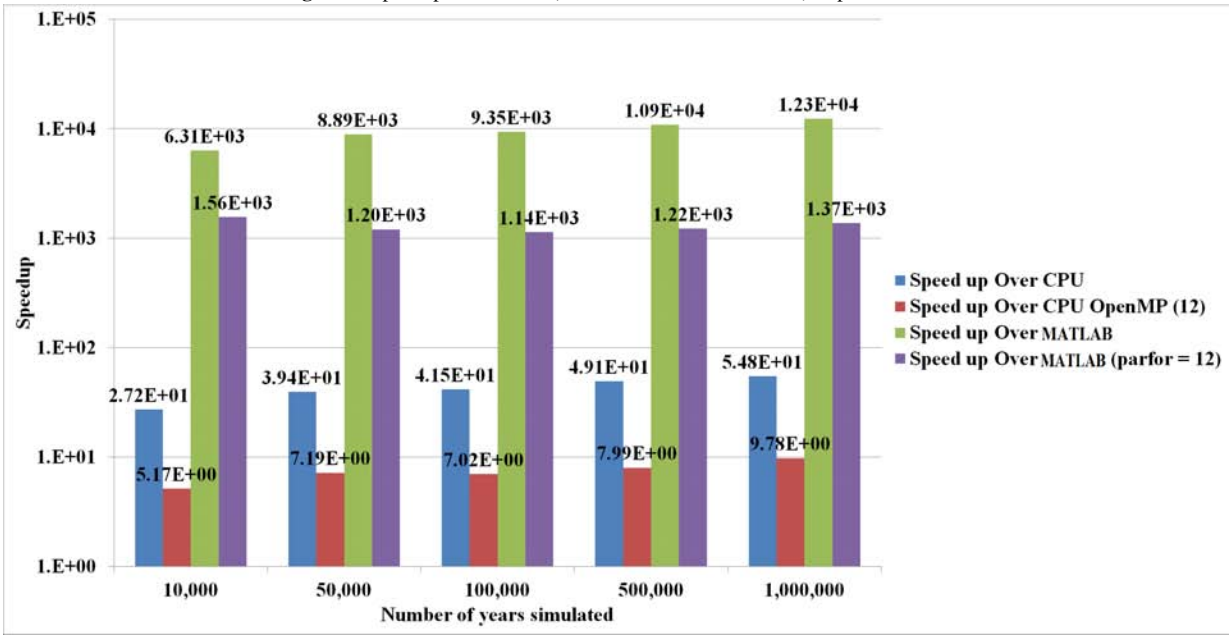


**Figure 6**. Speedup of the GPU (TeslaK80) implementation

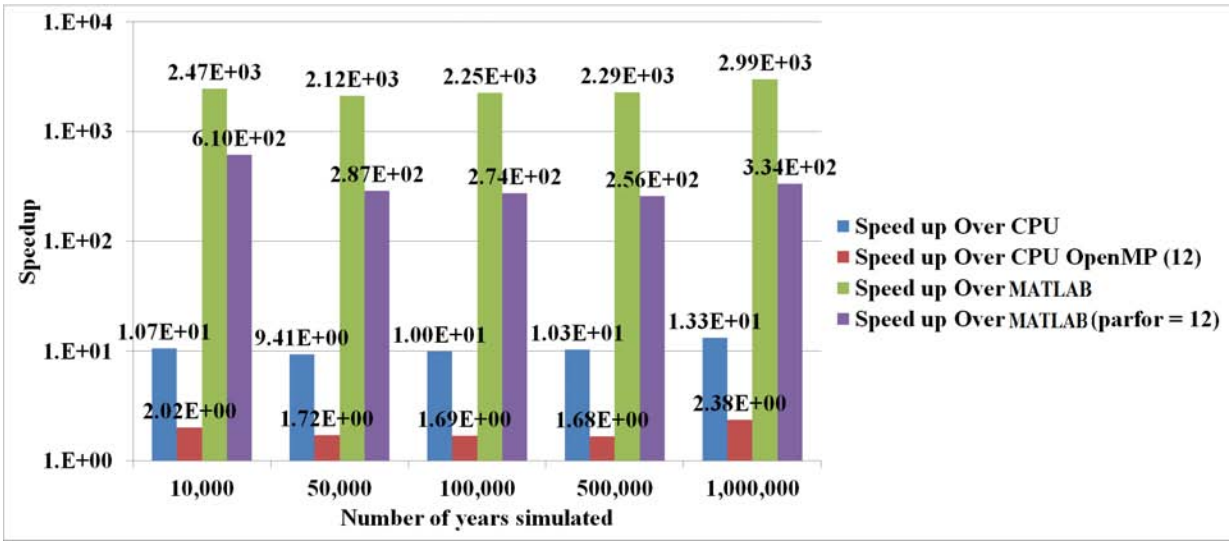


**Figure 7**. Speedup of the GPU (Quadro K4100M) implementation

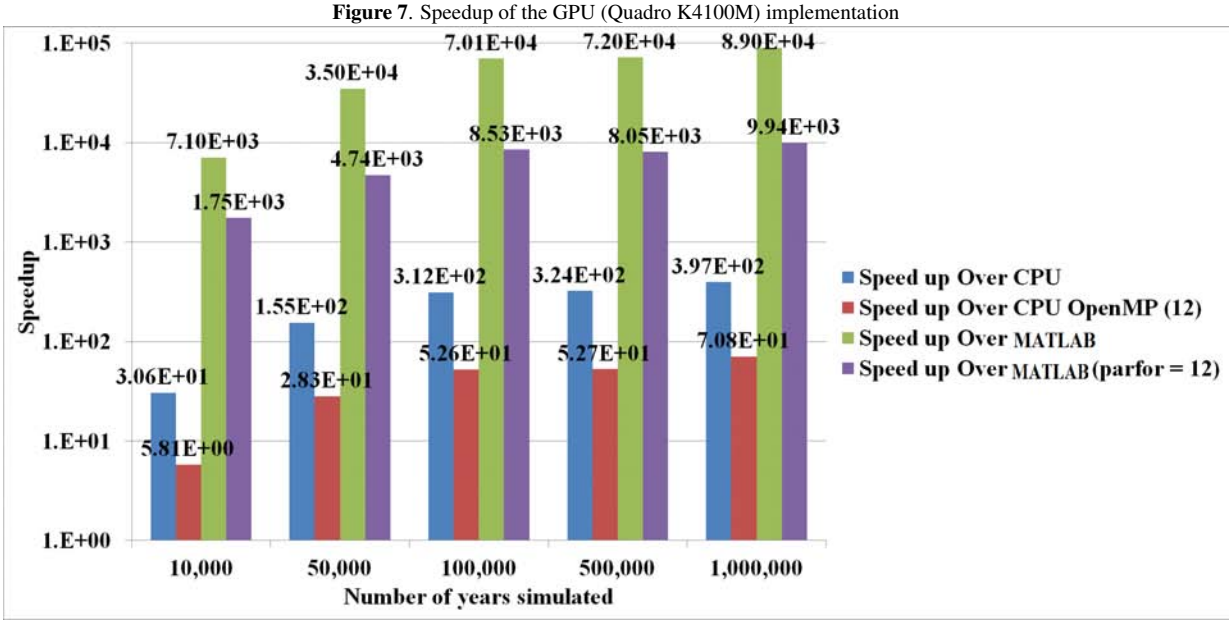


**Figure 8**. Speedup of the 4 GPUs (Tesla K80) implementation