PROCEEDINGS OF SCIENCE

ONLINE ICRC 2021
THE ASTROPARTICLE PHYSICS CONFERENCE
Berlin | Germany
37th International Cosmic Ray Conference
12–23 July 2021

Cosmic rays modulation in heliosphere models on GPU

Michal Solanik,[a,*] **Pavol Bobík**[b] **and Ján Genči**[a]

[a] *Technical university of Košice, Dept. of Computers and Informatics, Letná 9, Košice, Slovak Republic*

[b] *Institute of Experimental Physics SAV Kosice, Department of Cosmic Physics, Watsonova 47, Košice, Slovak Republic*

E-mail: michal.solanik@tuke.sk, bobik@saske.sk, jan.genci@tuke.sk

Parker's transport equation stochastic solution for simulation cosmic rays distribution in the heliosphere is demanding on computing resources. Simulations can last days, weeks, or even months with certain input parameters. We implemented 1D Forward-in-time and Backward-in-time models for GPU with successful acceleration ranged from 7x to 86x. This acceleration was gained with not a negligible reduction of accuracy, especially with changing the entire simulation from double-precision float-point format to floating-point format. This led to a certain deviation that we called pulsations that showed in results with input time step less than 2.0 s. In this paper, we discuss the parallelization process on GPU. We also discuss the comparison of our solution with Dunzlaff et al. and the overall accuracy of results gained from GPU implementation of 1D Forward-in-time and Backward-in-time models.

37th International Cosmic Ray Conference (ICRC 2021)
July 12th – 23rd, 2021
Online – Berlin, Germany

*Presenter

PoS(ICRC2021)1320

1. Introduction

Parker's transport equation stochastic differential equation (SDE) solution [1] [2] can be demanding on resources. This is especially true where we want to use simulations with low-time steps.

SDE simulations are independent of each other. Data-parallel architectures address this kind of problem. Instead of SIMD[1], GPU [3] uses a SIMT[2] execution model. In the case of the forward-in-time model, we need at least a hundred billion of simulations for a 1% statistical error of whole spectrum integral till 100 GeV, depending on the combination of input parameters [4]. Usually, we execute 100 billions simulations on the GPU with, an execution time in the order of hours.

Dunzlaff et. al. [5] accelerated their Parker's transport equation SDE solution using a GPU[3] as a computing unit. Compared to the CPU solution, execution time was decreased by 10-60 times depending on the input parameters.

Using s GPU as the computing unit is not that common in the cosmic rays modulation field. Most of the available papers aim at using GPU implementation not the entire process of parallelization except Dunzlaff et al..

In this paper, we will use 1D forward-in-time and backward-in-time SDE methods with the momentum p model from [2] to describe GPU implementation.

2. Implementation of cosmic ray modulations in heliosphere models on a GPU

Simulating cosmic ray distribution in heliosphere meets the requirements [6] given by Owens et al.. In the case of forward-in-time models, there are usually billions of quasi particle trajectory simulations.

We decided to split the CPU implementation of the F-p and B-p model, as defined in figure 1. Our approach is similar to that used by Dunzlaff et al. in [5]. However, instead of defining the injection point on the CPU, we define it and calculate the additional parameters on a GPU.

Dunzlaff et al. [5] did not mention the overall occupancy of a GPU. On Pascal architecture, we have been able to reach 75% usage of GPU. On the Turing and Ampere architecture, the amount of shared memory on them can be raised via GPU's API. This leads to the ability to reach 100% usage of GPU.

Each simulation of particles in the case of the B-p model has one result. Dunzlaff et al. [5] do not describe memory optimization or consumption. However, simulation in the case of F-p model can have zero or more results (registration particle inside the heliosphere at the selected position). During the development of the 1D F-p implementation for a GPU, we allocated memory for 200 results for every simulation. The relation between the number of simulations executed on GPU and memory turned out to be a bottleneck. We were unable to increase the number of blocks because of this relation. Consequently, we implemented a global counter, which was increased by the atomic add operation. Without the existing relation, we were able to increase the number of blocks from 8192 to 32768, where we reached maximal efficiency.

[1] Single Instruction Multiple Data
[2] Single Instruction Multiple Threads
[3] Graphics processing unit

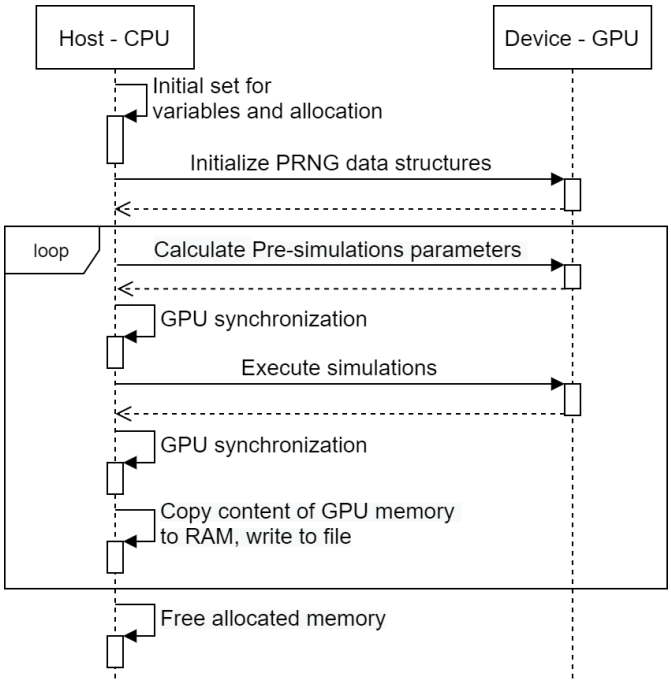


Figure 1: Redesign of a simulation on a GPU

Test	Size	Time of transfer	Execution time
2.1 millions with unified memory	9.375GB	5.95s	51.15s
4.2 millions with unified memory	18.75GB	10.68s	92.31s
4.2 millions with manual managed memory	18.75GB	6.13s	64.4s

Table 1: Proposed input parameters for accuracy tests

Authors of [5] also considered thread divergence as a problem. They decided to implement a step mechanism to minimize differences between the execution time of threads in the warp. Particles were divided into partitions. Particles in partitions were marked with a flag that indicated if the simulation is finished. Simulations of particles on GPU were executed and after each iteration, particles in partitions are sorted by this flag and only new simulations or not finished simulations were executed on the GPU. Best acceleration was achieved between 60-170 number of partitions.

From our perspective, we decided to take a different approach than Dunzlaff et al.. Our approach was aimed at minimizing divergence in the context of the entire GPU rather than warp. During the development of F-p model, we increased the number of blocks several times. As we mentioned earlier, the most radical increase was from 8192 to 32768. This resulted in 1.1-times acceleration against solution with lower number of blocks.

In the earliest part of the development, we decided to use unified memory. Unified memory [3] is a memory that is allocated and copied automatically between GPU memory and RAM. However, this approach soon became a bottleneck. Therefore, we decided to use memory profiling for implementation of the F-p model. As we can see in table 1, the time to transfer the memory between GPU's memory and RAM is significant with such a large number of simulations. We

Test no.	$K_0[cm^2/s]$	V [km/s]	dt [s]
I.	5 x 10^{22}	400	5.0
II.	1 x 10^{23}	300	5.0
III.	1 x 10^{23}	700	5.0
IV.	1 x 10^{22}	300	5.0
V.	1 x 10^{22}	700	5.0

Table 2: Proposed input parameters for accuracy tests

tested a run of 4.2 millions of simulations without data filtration and writing to disk. The execution time was 55.32s. Memory profiling does not take into account allocation time in unified memory. Consequently, we removed unified memory and replaced it with manual allocation and copying data from GPU memory to RAM. We were able to decrease execution time by 35.61%, and we reduced the transfer time between RAM and GPU memory by 54.13%.

3. Accuracy of GPU implementation

To measure accuracy, we choose to compare the results from GPU implementation with the CPU version of F-p, B-p methods divided by results from the Crank-Nicolson method that is described in [7]. The same approach was used and verified in [2] [8].

For testing purposes, we decided to use the following input parameters defined in table 2. Test I. adopts a similar approach to that used by Yamada in [9]. Test II-V represent input parameters border values in the heliosphere model. We decided to use solar wind speed based on the measurements that were analyzed in [10] and [11].

The overall accuracy of the 1D F-p model is acceptable. In the majority of evaluated cases, the maximal deviation was 10% for energies greater than 1 GeV. For each test case except test I, we simulated 500 billion quasiparticle trajectories. For the test I. statistic was increased to 1 trillion simulations. Figure 2a shows the ratio between the GPU implementation of the F-p model and the spectrum from CN. An accuracy over 1 GeV is acceptable, with a maximum deviation of 10%. Under 1 GeV we can observe deviations up to 24% at 0.3 GeV. To verify of our GPU implementation of the F-p model, we compared multiple versions of GPU implementations with different accuracy against a CPU implementation of the F-p model in double precision. The energy spectra for single and double precision are nearly identical with maximal deviation of 13% at low energy. This can be explained by insufficient statistics of the CPU spectra.

In the case of tests II and III, as shown in the figure 3a and 3b, we can see very similar shape as a wave with a deviation of 5% at the peak. Under 10 GeV, the maximal deviation is 12% in the case of test III and 4% in case of test II.

We were able to compare the ratio between both implementations of the B-p model and CN model. The maximum deviation between the ratio of GPU and CPU implementation of B-p method was at the level of 1%. Comparing the B-p to the CN model, the deviation reached 1% in the case of test II and III. In test IV and V, a deviation over 1 GeV is less than 5% compared to the CN model. However, under 1 GeV we observe a very similar shape, reaching a maximum deviation of 11% in the case of test IV and 23% in the case of test V.

System	CPU	RAM	GPU
CPU System	4 x AMD Opteron(TM) Processor 6274	32GB	-
GPU System	Intel(R) Core(TM) i5-7500 CPU	16GB	Geforce GTX1080TI 11GB

Table 3: Configuration of refence systems

Figure 5a shows the deviation that was found during the testing of 1D F-p model GPU implementation. Pulsations were present on the entire spectrum evaluated by simulation with a time step dt < 2.0. Pulsations were not present in spectra from simulations with time steps longer than 2.0s. We suggest several hypotheses for the primary cause of the pulsation error:

- Use of the –use-fast-math parameter during compilation
- Incorrect distribution of random numbers
- Incorrect distribution of injected energies
- Differences between single and double precision.

The fast math parameter [3] is used to accelerate math operation but it also decreases accuracy in certain mathematical operations. We tried to simulate 50 billion particles without the –use-fast-math parameter. From the output spectrum that is shown in the figure 5b, we can assume that this compilation parameter is not the cause of the pulsations. The only difference is visible between 0-2 GeV, where pulsations are more clearly visible.

As mentioned in section 2, we used the XORWOW pseudo-random number generator. Saito et al. [12] confirmed that the XORWOW pseudo-random number generator failed in a few tests of the BigCrush test framework. Dunzlaff et al. [5] measured differences between simulation using XORWOW and MTGP32[4] pseudo-random number generators. They did not find any issues that can affect simulation but they were not measured in potential edge cases. After using the MTGP32 pseudo-random generator, the output spectrum did not change when compared against the spectrum in figure 5a.

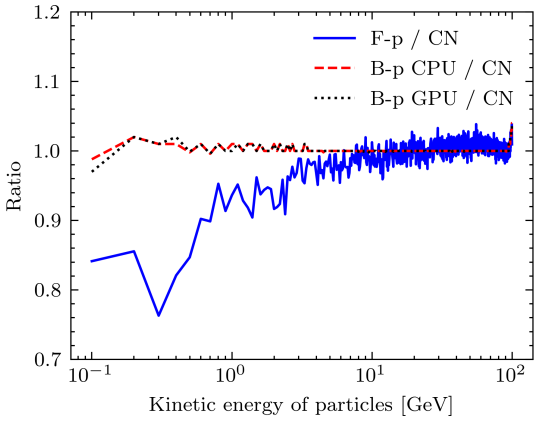
The most plausible hypothesis is that the pulsations are caused by changing precision in the GPU implementation. Pulsations are not present in the output spectrum from CPU implementation with identical input parameters.

4. GPU implementation acceleration testing

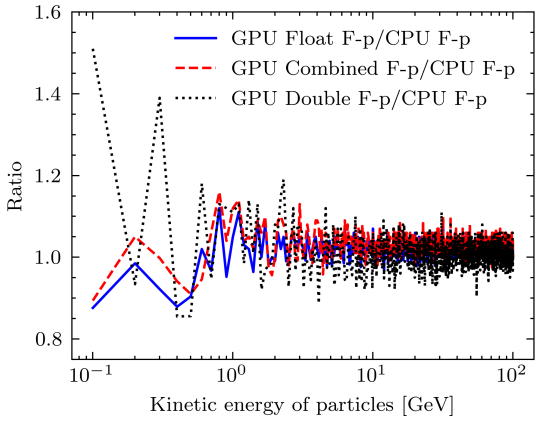
Redesigning and reimplementing heliospheric models on a GPU would be useless if we had not accelerated the GPU solution against the CPU solution on the reference system. The configuration of the reference GPU and CPU system is defined in table 3.

Dunzlaff et al. [5] implemented a GPU solution that was accelerated 10-60 times against an OpenMP solution on a CPU. This range is valid for backward-in-time GPU implementation because they solely focused on backward-in-time. Our implementation of the B-p model was accelerated from 86.87 to 183.47 depending on input parameters, similarly to Dunzlaff et al.. The differences

[4]Variant Mersenne Twister pseudo-random generator suitable for GPU

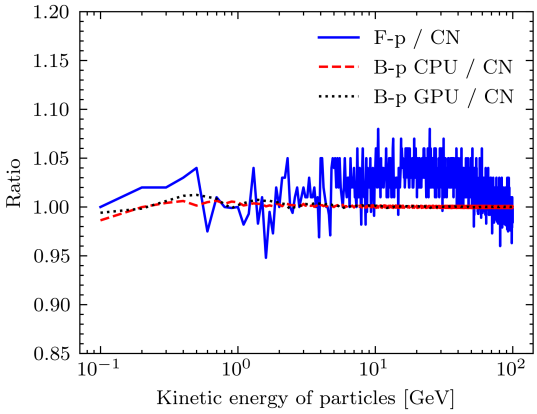


(a) Ratio of energy spectra for V = 400 km/s, $K_0 = 5x10^{22} cm^2/s$ and time step dt = 5.0s

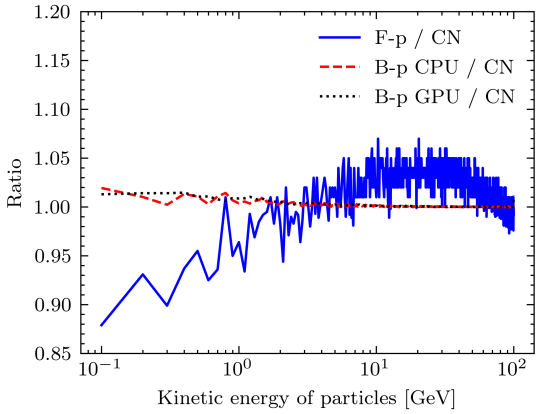


(b) Ratio of energy spectra between GPU and CPU implementations

Figure 2: Ratio of energy spectra for F-p and B-p model for test I.



(a) Ratio of energy spectra for V = 300 km/s, $K_0 = 1x10^{23} cm^2/s$ and time step dt = 5.0s



(b) Ratio of energy spectra for V = 700 km/s, $K_0 = 1x10^{23} cm^2/s$ and time step dt = 5.0s

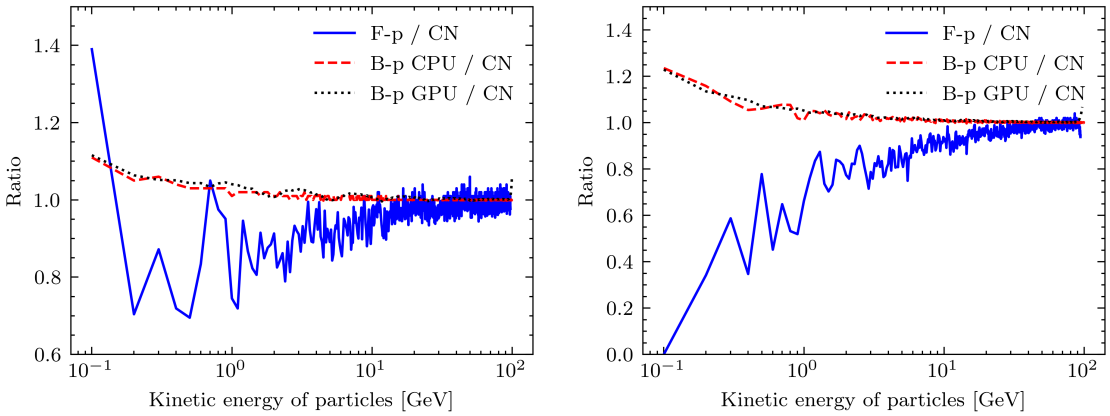
Figure 3: Ratio of energy spectra for tests II and III.

between both accelerations are caused by a greater time gap between the used reference CPU and GPU that we used.

In the case of the forward-in-time model, we did not find any paper on the topic of acceleration. Therefore, we decided to test acceleration with input parameters from test I. Our GPU implementation of 1D F-p was accelerated 7.71-times compared to CPU implementation on a reference multiprocessor system.

5. Conclusion

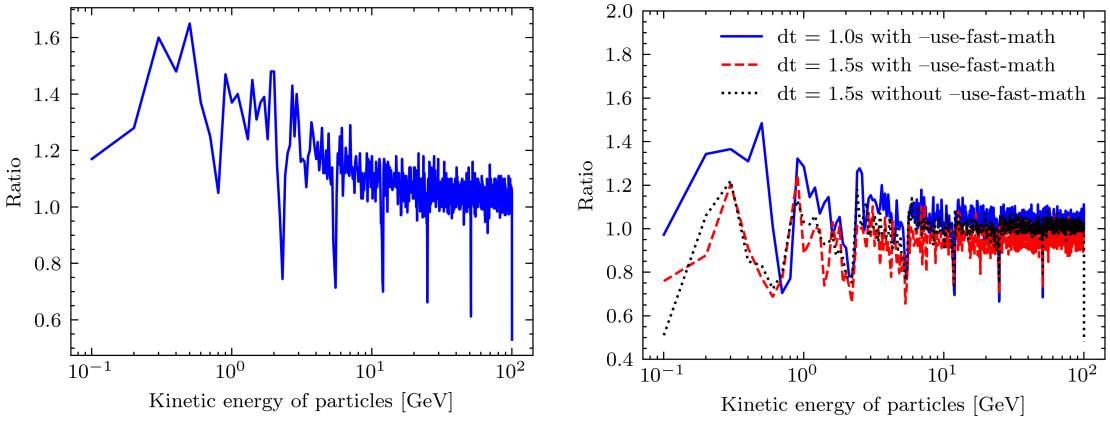
Acceleration on a GPU can provide much needed computing power, especially for backward-in-time implementation. The accuracy of the tested GPU implementation of the B-p model is nearly identical with a CPU implementation.



(a) Ratio of energy spectra for V = 300 km/s, $K_0 = 1x10^{22} cm^2/s$ and time step dt = 5.0s

(b) Ratio of energy spectra for V = 700 km/s, $K_0 = 1x10^{22} cm^2/s$ and time step dt = 5.0s

Figure 4: Ratio of energy spectra for tests IV and V.



(a) Ratio between F-p and CN spectra with dt = 0.5s

(b) Ratio between F-p and B-p spectra with dt = 1.5s

Figure 5: Ratio between F-p spectra with pulsation and B-p spectra

GPU implementation of the forward-in-time model proved useful, even with lesser acceleration than the backward-in-time model. Issues with accuracy with time steps less than 2.0 s can be considered as an edge case because it is not common to use such a low time step, which we have assumed that from fact that many authors have not used time steps lower than 5.0 s in [2] [5] [13] [14].

The overall effect of acceleration on a single GPU can be multiplied by implementing support for multiple GPU on a single computer or distributed system that can use a GPU as a computing unit across multiple computers.

6. Acknowledgment

The PECS project TUKE Space Forum is acknowledged. PB acknowledges the Slovak VEGA grant agency, project 2/0077/20, for their support.

References

[1] Strauss, R. Du Toit, and Frederic Effenberger. "A hitch-hiker's guide to stochastic differential equations." Space Science Reviews 212.1 (2017): 151-192.

[2] Bobik, P., et al. (2016), On the forward-backward-in-time approach for Monte Carlo solution of Parker's transport equation: One-dimensional case, J. Geophys. Res. Space Physics, 121, doi:10.1002/2015JA022237.

[3] "CUDA Toolkit Documentation", Docs.nvidia.com, 2021. [Online]. Available: https://docs.nvidia.com/cuda/. [Accessed: 14- Apr- 2021].

[4] Mykhailenko V., Bobik P., Statistical error for cosmic rays modulation evaluation by 1D and 2D models, 37th International Cosmic Ray Conference, Berlin 2021

[5] Dunzlaff, P., R. D. Strauss, and M. S. Potgieter. "Solving Parker's transport equation with stochastic differential equations on GPUs." Computer Physics Communications 192 (2015): 156-165.

[6] Owens, John D., et al. "GPU computing." Proceedings of the IEEE 96.5 (2008): 879-899.

[7] Batalha, L. (2012), Solar Modulation effects on cosmic rays (Modelization with force field approximation, 1D, 2D numerical approaches and characterization with AMS-02 proton fluxes), MS thesis, Instituto Superior Tecnico, Universidade Tecnico de Lisboa, Portugal.

[8] Kolesnyk, Yuriy L., et al. "An analytically iterative method for solving problems of cosmic-ray modulation." Monthly Notices of the Royal Astronomical Society 470.1 (2017): 1073-1085.

[9] Yamada, Yoshihiko, Shohei Yanagita, and Tatsuo Yoshida. "A stochastic view of the solar modulation phenomena of cosmic rays." Geophysical research letters 25.13 (1998): 2353-2356.

[10] Tokumaru, Munetoshi, Masayoshi Kojima, and Ken'ichi Fujiki. "Long-term evolution in the global distribution of solar wind speed and density fluctuations during 1997–2009." Journal of Geophysical Research: Space Physics 117.A6 (2012).

[11] McComas, D. J., et al. "Ulysses observations of very different heliospheric structure during the declining phase of solar activity cycle 23." Geophysical research letters 33.9 (2006).

[12] Saito, Mutsuo, and Makoto Matsumoto. "A deviation of CURAND: standard pseudorandom number generator in CUDA for GPGPU." Proceedings of 10th International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. 2012.

[13] Wawrzynczak, Anna, Renata Modzelewska, and Agnieszka Gil. "Algorithms for Forward and Backward Solution of the Fokker-Planck Equation in the Heliospheric Transport of Cosmic Rays." International Conference on Parallel Processing and Applied Mathematics. Springer, Cham, 2017.

[14] Kappl, Rolf. "SOLARPROP: Charge-sign dependent solar modulation for everyone." Computer Physics Communications 207 (2016): 386-399.