PROCEEDINGS OF SCIENCE

Performance optimization of air shower simulations with CORSIKA

Dominik Baack* **for the CORSIKA 8 collaboration** †
Technische Universität Dortmund, Germany
E-mail: dominik.baack@tu-dortmund.de

With the steady increase in accuracy, size and complexity of astroparticle physics experiments the need for an extensive amount of high precision Monte Carlo simulations is rapidly growing. Contrary to the increasing demand is the demise of "Moore's law" which leads to situations where the system structure of high-performance computing is fundamentally changing and large amounts of money are invested in new infrastructure. In the field of astroparticle physics CORSIKA 7 is currently the most commonly used simulation program, therefore the presented work is focused on this software. All methods provided are also currently transferred to the new CORSIKA 8 framework which will replace CORSIKA 7 in the near future. Due to various constraints on hardware, geometry and physics no experiment is able to observe the full air shower particle cascade developing in the atmosphere. The removal of the non-visible phase space of the cascade at an early stage of the simulation has immense potential to reduce the expense of calculations without changing the results of the simulation for the experiment. Fast machine learning models allow the identification and removal of particles from those regions to speed up the simulations. This works for example for neutrinos by orders of magnitude. First results are shown to demonstrate this technique.

Furthermore, when showers are simulated with the IACT configuration around 75% of the time is spent on the Cherenkov photon creation and propagation. We also show results from parallelizing this part of the simulation on GPUs and CPUs with OpenCL.

36th International Cosmic Ray Conference -ICRC2019-
July 24th - August 1st, 2019
Madison, WI, U.S.A.

*Speaker.

†for collaboration list see PoS(ICRC2019)1177 or https://gitlab.ikp.kit.edu/AirShowerPhysics/corsika/wikis/ICRC2019-author-list

PoS(ICRC2019)181

1. Introduction

With the increasing quality, size and complexity of modern astroparticle experiments, like CTA, it is possible to observe cosmic rays in even higher detail than today's experiments. To enable the precise analysis of the observed information it is necessary to increase the amount and quality of simulated data in a similar fashion. This requires new methods to improve physical correctness, which typically slows down the simulation. In addition, a massive investment in computing time is required to reach the demanded number of simulations.

With even bigger experiments like IceCube-Gen2 [1] or SKA [2] on the horizon the computing power needed to generate those simulations become the limiting factor for extensive analysis. Even today the analysis of rarely occurring physical events is significantly hampered due to the difficulty to generate those in simulations.
The only solution to reach the demanded simulation statistics, besides tailoring the simulation which can introduce bias or difficult to detect errors, is to speed up the simulation overall. On the other hand, the gained runtime can be invested in higher quality simulation methods.
The focus is placed on the widely used Monte Carlo Simulation CORSIKA for atmospheric particle showers. Here, most methods presented in this work were tested in the current release CORSIKA 7 [3] but they will be transferred to the complete rewritten CORSIKA 8 [4]. This new version of CORSIKA is written in modern C++ and allows an even broader approach for performance improvements compared to the old and complex F77 code.

The main optimizations explained in section 2 and 3 can be separated into two different approaches. The first is a reduction of the number of calculations needed do to machine-learned experiment specific cuts. The second approach is a more efficient parallelization of the workload generated during the simulation.

2. Efficent removal of unobservable particles

The core idea behind this optimization is to reduce the necessary amount of calculation during the simulation by removing particles that do not contribute to the observable phase space of the experiment. For Cherenkov telescopes, like the CTA telescope, the observed Cherenkov light propagated to the ground is displayed in image 1 with preliminary telescope positions and sizes for the north observatory. It is clearly evident that a large amount of Cherenkov photons are generated which dos not hit the detector and contribute to the measurement at all. When looking at individual particle tracks on cascade level, image 2, you see similar results. Only a fraction of the simulated particles emits experimental observable light. With increasing zenith angle, energy and distance from the shower core, the effect is even more prominent. This effect is not limited to Cherenkov radiation but is identifiable for other experiments as well where the size of the observable area is much smaller than the shower area. Beside the mentioned geometrical constraints, physical constraints can be found as well.

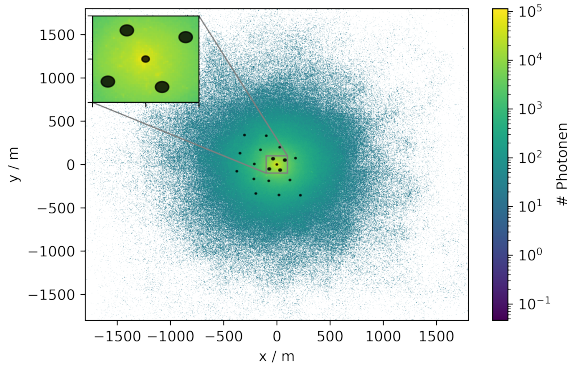

Figure 1: Cherenkov Photon emission of a single 1 TeV gamma-ray shower compared to the size of CTA.

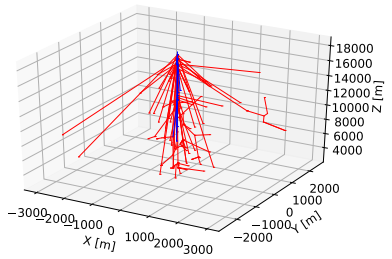

Figure 2: Displayed are non electromagnetic particles of a 1 TeV proton cascade. The blue lines are particles that possibly contribute Cherenkov photons to the observation of a single 10m telescope positioned at the origin.

2.1 Dynamic Stack

The main working memory inside of CORSIKA 7, used to store intermediate particles, is called stack. Particles created during interactions will be stored there and iteratively read in the "last in - first out" order. The dynamic stack [5, 6], originally developed for CORSIKA 7, is a replacement for this relatively simple storage to create easy access and modifiable interface to the processed particles. In addition, the calculation order of the cascade can be changed by prioritizing certain particles. The decorator design pattern implemented with the dynstack interface with several pre-implemented features can be used to tailor the simulation to specific detectors.

2.2 "Small" size optimizations

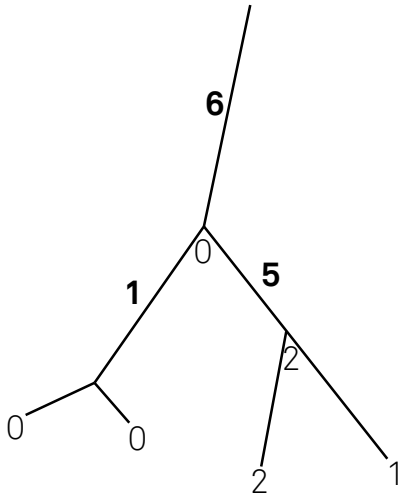

Figure 3: Reconstruction of the contribution parameter in a cascade. Highlighted is the accumulated factor, normal the factor of this vertices.

The main optimization potential for experiments smaller then the shower diameter is the removal of not observed particles displayed in image 2. To identify those particles several cascades will be simulated and completely logged. In case of Cherenkov experiments, this includes Cherenkov photons hitting a sphere around the telescope. The widely used IACT [7] extension for CORSIKA 7 can be used to detect those photons. For even greater performance improvement, the experiment specific acceptance must be applied to the stored data and marked accordingly.

With this experiment specific data, it is possible to reconstruct the cascade and annotate each particle with its contribution to the experiment. The most important factor here is to add the contribution of every child particle to its parent particle displayed in image 3. This guarantees that important particles further down in the cascade will not be eliminated by accidentally removing invisible parent particles.

Applying basic machine learning methods, e.g. decision tree, on the cascade data shows that even a single parameter allow the reduction of nearly 40% of the particles without significant loss of observable information. The strongest separation parameter for small Cherenkov experiments is the angle between the momentum and the direction to the experiment center. This is displayed in figure 4. When more complex methods, like random forests, are used a bigger fraction of the shower can safely be removed. The disadvantage is the slightly longer execution time for each particle. Depending on the experiment a reasonable compromise must be found.

2.3 "Physics" optimization

For some experiments or special analysis, only specific showers are of interest. Those can have rare physical requirements, like a very high energetic muon hitting the detector. If the necessary conditions can be roughly evaluated during runtime it is possible to stop the cascade simulation that does not meet those. With the sorting modules integrated into dynstack [5, 6] for CORSIKA the program can be set up to prioritize those particles that still can create the wanted effects and delay all other particles until the criteria are met. The additional memory required of this method does grow with energy, but with far less than 1 kB per particle, even the intermediate level of PeV shower are easily stored on current infrastructure.

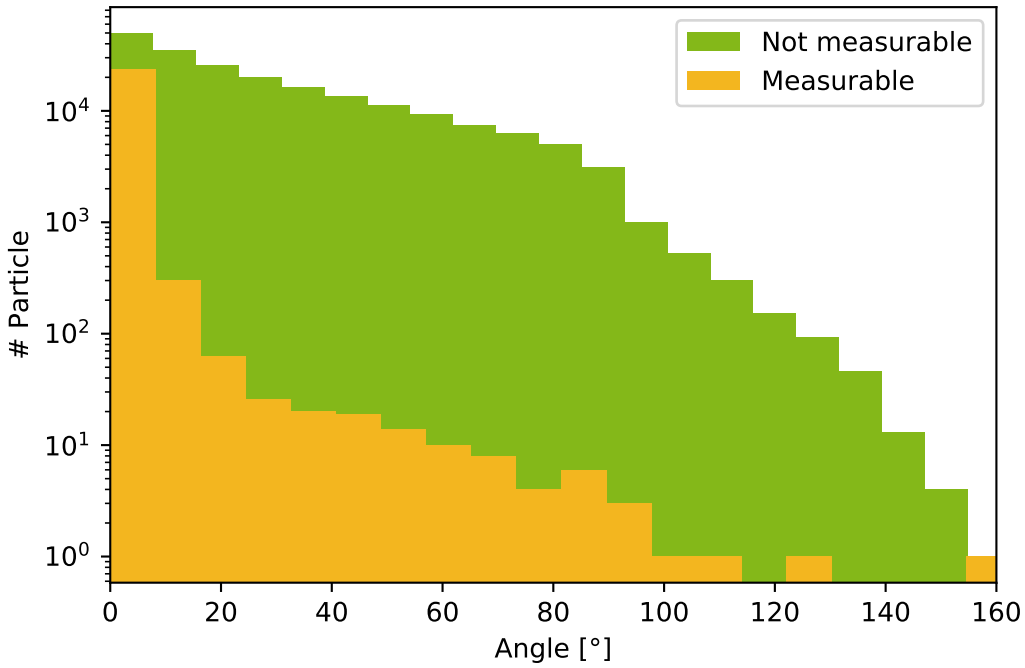

Figure 4: Differentiation between non-/observable particles based on its flight angle to the telescope direction.

3. Acceleration of CORSIKA via parallel execution

Besides reducing the calculation time through workload reduction (see chapter 2) it is possible to distribute the workload efficient over multiple computing cores. For modern CPUs the performance improvements per generation of individual cores is declining as seen in figure 5. To utilize future hardware to its fullest potential it becomes necessary to execute several simulation instances in parallel.

The current method of parallel execution that is used for several different software frameworks,

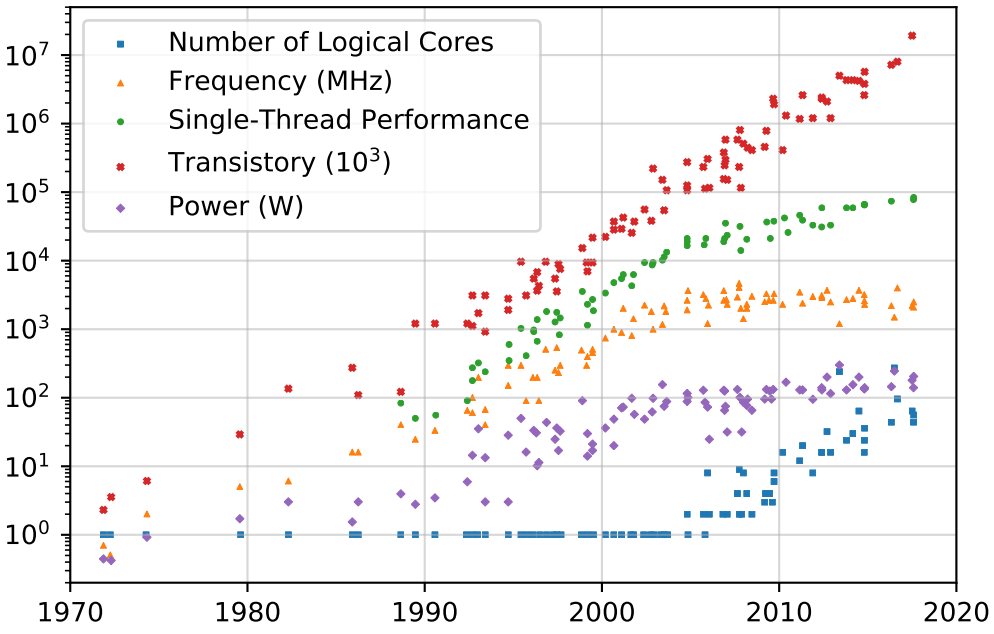

Figure 5: Performance developement of CPU's in recent years [8].

including CORSIKA 7, is through multiple executions of the same program. For CORSIKA 7 this method of parallelization computes its results without sharing basic information, this includes the OpenMPI [9] parallelization which only shares top-level data. For the "old" CORSIKA this approach worked for several years, because of its relative low memory footprint and a sufficient core to memory (RAM) ratio in most computing clusters.

For the new generation of CORSIKA several new functions and optimizations are added to the base program which, when chosen, require a bigger memory footprint than ever before. In addition, the number of cores is rapidly increasing while memory is not. This effect is backed by several additional technical innovations like hyperthreading and alternative CPU architectures like ARM. Both technics show promising results for the future. In addition, separate acceleration hardware such as graphic cards allow improvements over a factor of 10^2 for selected parts such as Cherenkov emission.

3.1 CPU Parallelization

Most of the memory resource needed in CORSIKA are static tables with interpolation values or other constant data. The excessive memory use from multiple processes can, therefore, be avoided if these tables are shared between parallel running calculations. This can be enabled by using multiple threads which allow an easy approach to resource sharing. The stack described in the last chapter 2.1 can be used as a synchronization layer between multiple independently running threads.

The interaction models used in CORSIKA 8 are currently still mostly written in Fortran with data exchange over common blocks. Those need to be modified, for example via an automatic python script, to call thread-safe memory routines. An alternative method is the use of OpenMP pragmas in Fortran which only introduce slight modifications.

3.2 GPU Parallelization

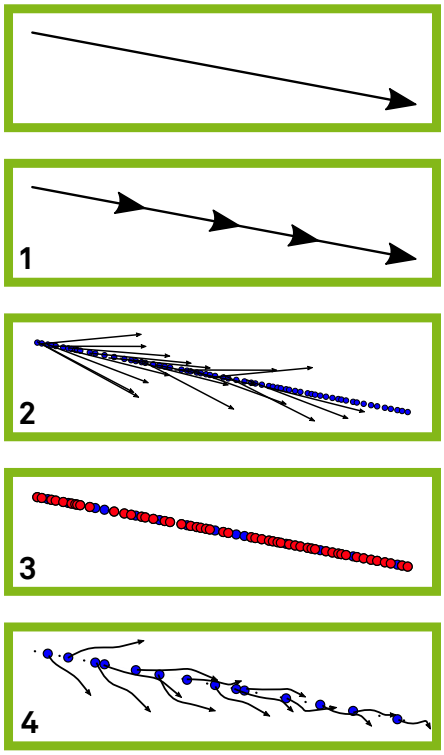

Figure 6: Overview of calculation order of Cherenkov photons on the GPU.

Parallelization on GPUs is only feasible for ultra-high parallel computing tasks which rely mostly on arithmetic calculations. In the case of CORSIKA, this is mostly the handling of radio and Cherenkov emission from cascade particles. In case of Cherenkov light $\sim 10^6\,\mathrm{TeV}^{-1}$ (settings for the MAGIC-Telescope [10]) independent photons must be handled and propagated to the experiment. To minimize the workload on the CPU and transfer time to the GPU, the best way is to transfer individual particle tracks to the GPU and not pre-generated photons. Further, it is recommended to avoid idle timeframes on the GPU by collecting particle tracks of several parallel running (section 3.1) shower simulations on a single GPU. In image 6 an outline over the current processing order on the GPU is given. The platform and hardware independent programming language OpenCL [11] is used to run the Cherenkov code on a wide variety of systems, including CPU, GPU (AMD ™& NVidia ®) and FPGAs.

The method of distributing computation task in a parallel setup is essential to gain the highest performance. The steps taken here are shown in Figure 6:

1. (1x work-group/block per track) The track is split into several smaller straight tracks
2. (1x work-group/block per track) Random numbers are drawn and photons along the track segments are generated including a direction, and afterward stored in GPU memory. Atmospheric absorption is applied during the generation.

3. (1x work-item/kernel per photon) The photons are projected onto the ground for a very fast first check. If it is not possible to hit the detector the particle is discarded.

4. (1x work-item/kernel per photon) Surviving photons are propagated correctly to ground-level. Interpolation tables inside the texture memory provide fast access to atmospheric data.

4. Conclusion

For the most efficient production of cosmic ray simulations with CORSIKA a broad spectrum of different optimization techniques is available. With the specialization of the general simulation software to specific experiments or physical problems, it is possible to reduce overall calculation time significantly. In addition, the use of heterogeneous computing infrastructure with the upcoming parallelization options can result in more cost-effective simulation.

Acknowledgments

Part of this work is supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 4 "Providing Information by Resource-Constrained Analysis", project C3.

References

[1] M. Aartsen, M. Ackermann, J. Adams, J. Aguilar, M. Ahlers, M. Ahrens, D. Altmann, T. Anderson, G. Anton, C. Arguelles, et al., *arXiv preprint arXiv:1412.5106* (2014).

[2] P. Dewdney, P. Hall, R. Schillizzi, and J. Lazio, *Proceedings of the Institute of Electrical and Electronics Engineers IEEE* **97** (2009) 1482–1496.

[3] D. Heck, G. Schatz, J. Knapp, T. Thouw, and J. Capdevielle, *Corsika: a monte carlo code to simulate extensive air showers*, tech. rep., 1998.

[4] M. Reininghaus and R. Ulrich, *Corsika 8–towards a modern framework for the simulation of extensive air showers*, in *EPJ Web of Conferences*, vol. 210, p. 02011, EDP Sciences, 2019.

[5] D. Baack, *Data Reduction for CORSIKA*, Tech. Rep. 2, E5b, Faculty Physic, TU Dortmund, 06, 2016.

[6] D. Baack, "Cor-PlusPlus."

[7] K. Bernlöhr, *Astroparticle Physics* **30** (2008) 149–158.

[8] K. Rupp, "42 years of microprocessor Trend Data." License: Creative Commons 4.

[9] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, *Open MPI: Goals, concept, and design of a next generation MPI implementation*, in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, (Budapest, Hungary), pp. 97–104, September, 2004.

[10] J. Aleksić, S. Ansoldi, L. Antonelli, P. Antoranz, A. Babic, P. Bangale, M. Barceló, J. Barrio, J. B. González, W. Bednarek, et al., *Astroparticle Physics* **72** (2016) 76–94.

[11] J. E. Stone, D. Gohara, and G. Shi, *Computing in science & engineering* **12** (2010) 66.