Journal of Physics: Conference Series **396** (2012) 022046 doi:10.1088/1742-6596/396/2/022046

IceCubes GPGPU's cluster for extensive MC production

Martin Merck, Dmitry Chirkin, Juan Carlos Díaz Vélez, Heath Skarlupka

University of Wisconsin Madison, Madison, WI 53703 USA

E-Mail: dima@icecube.wisc.edu, juancarlos.diazvelez@icecube.wisc.edu, heath.skarlupka@icecube.wisc.edu

Abstract. GPGPU computing offers extraordinary increases in pure processing power for parallelizable applications. In IceCube we use GPUs for ray-tracing of cherenkov photons in the Antarctic ice as part of detector simulation. We report on how we implemented the mixed simulation production chain to include the processing on the GPGPU cluster for the IceCube Monte-Carlo production. We also present ideas to include GPGPU accelerated reconstructions into the IceCube data processing.

1. Introduction

The IceCube Neutrino Observatory is a cubic kilometer neutrino detector located at the South Pole. Buried one kilometer below the Antarctic ice, the detector consists of over 5,000 digital optical modules (DOMs) spread over 86 strings producing over one terabyte of raw data every day. Processing of raw data and analysis by physicists happens on our local 1,200 core cluster. This leaves very little local CPU time for simulation of the detector, so it is important that simulation be done on GRID resources at universities in the IceCube collaboration around the world. Simulation of the detector is necessary to help test new reconstruction algorithms and to give an overall better understanding of the detector. The Icecube simulation chain recreates the events necessary to trigger an event in the detector in a virtual environment. First the simulation chain generates photons and then propagates them through the detector array. If enough DOMs in a proper ordering of time and space are fired during the propagation of photons, an event is triggered. Propagating millions of photons through the detector can be time consuming and unrealistic with the limited number of CPU cores and the large amount of simulation work that needs to be done. Photonics tables are used to do lookups on photon propagation, which greatly reduces the time needed to propagate photons, but limits the amount of work that can be processed by the GRID due to photonics tables requiring 25GB of storage per job. Solutions to this problem are discussed below, including the usage of a new 48 Nvidia Tesla M2070 GPU cluster.

2. Introduction to GPGPU computing

GPUs introduce hundreds of extra computing cores per card to a HPC cluster, versus adding only a few additional cores per CPU. Problems that can be made massively parallel with little dependence on other events being calculated are best run on the hundreds of extra cores in a GPU. Coding in a massively parallel environment for GPUs can present its own challenges especially for those used to coding in a CPU environment. One of the biggest challenges is keeping up with the GPUs and constantly feeding them new calculations. If new work handed to the GPU isn't timed properly,

software can quickly become CPU bound and the GPU will be underutilized. Propagating millions of photons through Icecube is an excellent example of a massively parallel problem that lends itself to GPU computing.

3. Icecube simulation chain

One of the most CPU-intensive aspects of the IceCube Monte Carlo simulation involves the in-ice propagation of photons emitted by charged leptons via Cherenkov radiation. This is represented in Figure 1. as the blue edge labeled "hitmaker". In order to speed up our simulations we rely on sampling from large probability density function (PDF) tables that describe photon amplitude and arrival times according to location of both receptor and emitter as well as arrival direction of emitter particle, arrival direction of photon at receptor, and wavelength of photon. Generation of these tables involves running many Monte Carlo simulations a priori but once generated, can speed simulation production significantly. The current software implementation of this technique is called "photonics" and includes both the Monte Carlo code for generating PDF tables and the code to read and interpolates between table bins. Photonics has made it possible to generate IceCube Monte Carlo simulations almost in real time by taking advantage of grid computing in the U.S. and Europe. There are, however, some drawbacks to this approach.

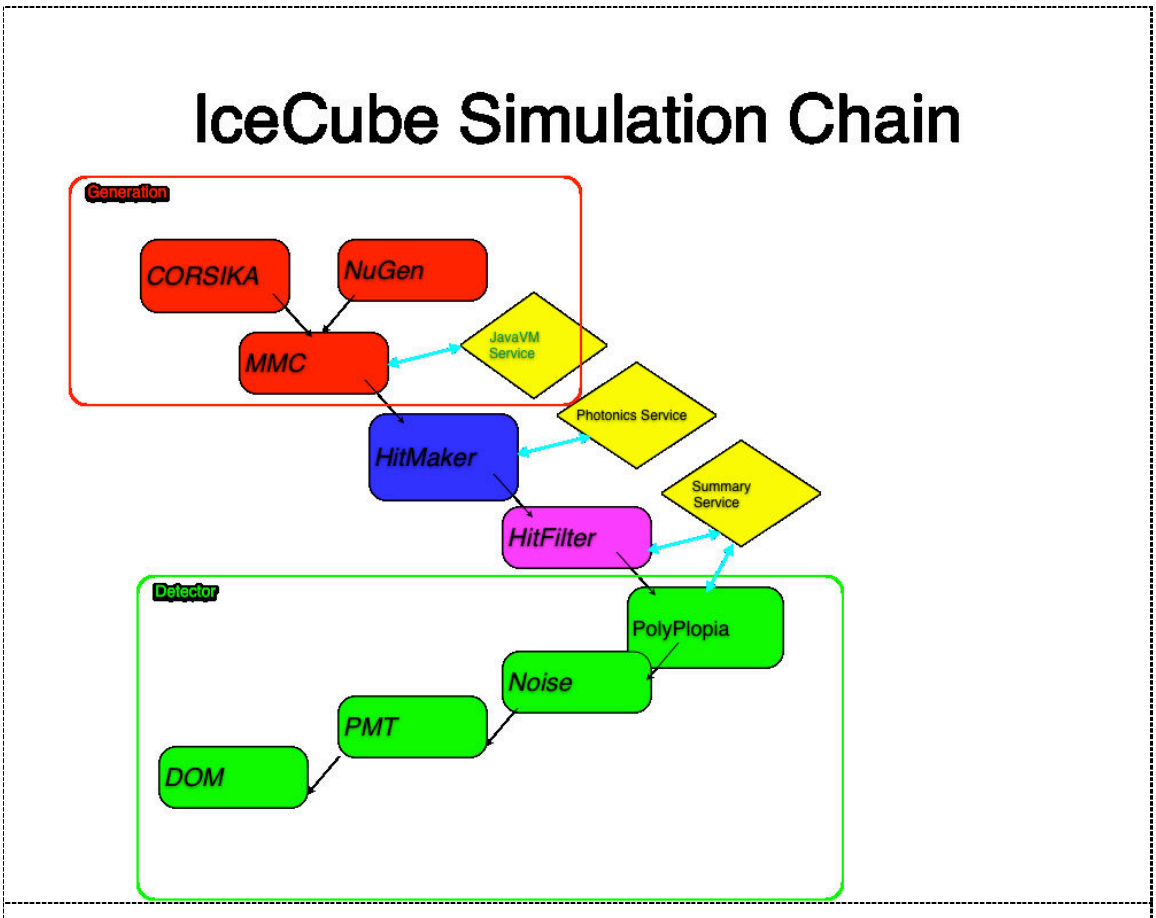

Figure 1. Graph representation of the IceCube simulation chain

1. Typical tables are too large to load in memory at once on an average Grid computing node.
2. Interpolation between points in tables is difficult and results in large errors unless tables are finely binned.
3. Binning must be kept sufficiently coarse in order to fit in the locally allocated storage for compute nodes.
4. The size of even coarsely binned tables is too large to be transferred over the network with each job. As a result we are restricted to run our Monte Carlo on machines that have preinstalled tables.

In order to address the first point, we have devised an approach of splitting tables by incident angle bins and making several passes to the data, once for each bin. The photons generated at each pass are then added to the data structure for further processing down the simulation chain. This approach allows us to increase the granularity of the binning somewhat but we are still restricted by the third point. One solution to address point no. 4 has been to sort events according to incident angle and split the photonics processing into separate task which only have to transfer a small section of the PDF tables. In order to manage and coordinate the scheduling of such tasks, we have incorporated the use of Condor DAGMan (Directed Acyclic Graph Manager) into our simulation production management software. DAGMan manages job ordering and dependencies such as input and output files between within a Condor system [1].

The introduction of GPU-based PPC (photon propagation code) has not only sped up simulations dramatically but has eliminated some of the errors introduced by interpolation and binning effects introduced by the Photonics service. PPC actually runs faster than the Photonics service for all but the highest energy events that produce many photons in the ice volume around the detector. At the moment we are limited as to the number of GPU hardware resources compared to standard CPU cores available on the Grid. In addition, this once dominating part of the simulation chain has now become negligible compared to other parts of the simulation as far as wall time is concerned. If we run the full simulation chain on one of our GPU equipped computing nodes, most of the time is spent on CPU-based computations while the GPU is idle.

The solution to this is obvious and calls for a splitting of tasks so that the CPU-intensive parts of the simulation run on standard hardware while the PPC simulation is executed on GPU-equipped nodes. The DAG-based simulation makes it possible to optimize utilization of both CPU and GPU resources by the use of different topologies as shown in Figure 2, Figure 3, and Figure 4. The optimal DAG topology might differ from simulation to simulation. For low energy simulations, typical GPU tasks are fast and thus call for many CPU-bound generator tasks to feed into a single PPC task. Depending on how CPU intensive the subsequent detector simulation and event reconstruction tasks are, these can be fanned out to multiple nodes as shown in Figure 3.

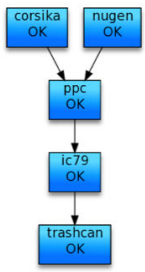

Figure 2. Neutrino-Cosmic Ray coincident simulation

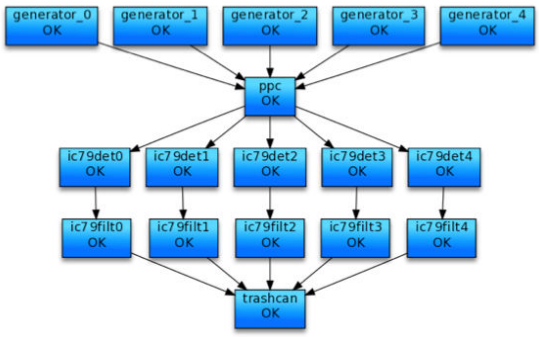

Figure 3. Fan-in, fan-out Simulation + Reconstruction

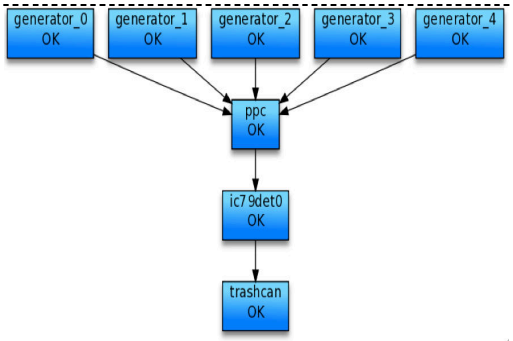

Figure 4. Fan-in simulation

4. Photon Propagation Code (PPC)

The photon propagation code (PPC) [2] was initially written to study the feasibility of direct photon propagation for simulation of events in IceCube. The simple nature of photon propagation physics allowed us to focus on the code optimization, to make sure the simulation ran as fast as possible. The simulation was written in C++, then re-written entirely in Assembly for the 32-bit i686 architecture with SSE vector optimizations. The Assembly version of the program used the SSE instructions for photon rotation and locating the optical sensor closest to the photon segment, while the calculation of the scattering angle was performed in one go using only the registers of the FPU stack.

A project called i3mcml [3] demonstrated that significant acceleration of the photon propagation is possible by using the graphics processing units (GPUs). We confirmed this with a version of PPC that employs the NVIDIA GPUs (graphics processing units) via the CUDA programming interface [4]. Recently a new version was written that additionally uses OpenCL [5], supporting both NVIDIA and AMD GPUs, and also multiCPU environments. The relative performance of these different implementations (for simulating both in-situ light sources, or flashers, and Cerenkov light from muons) is compared in Table 1. We have studied the simulation with these versions of PPC and i3mcml and were able to demonstrate excellent agreement between them.

Table 1. Speedup factor of different implementations of PPC compared to the C++ version. The GPU used in this comparison was either of the two in the NVIDIA GTX 295 video card.

	C++	Assembly	GTX 295 GPU
Flasher	1.00	1.25	147
Muon	1.00	1.27	157

The reason for the substantial acceleration of PPC on GPUs is the highly parallel nature of the simulation of the photon propagation. All of the simulated photons go through the same simulation steps (see Figure 5): photon propagation between the scattering points, calculation of the scattering angle and new direction, and evaluation of whether the current photon segment intersects any of the optical sensors of the detector array. The GPUs are designed to perform the same computational operation in parallel across multiple threads. Each thread works on its own photon for as long as the photon exists. When the photon is absorbed or hits the detector the thread receives the new photon from a pool of photons for as long as that pool is not empty. Although a single thread runs slower than a typical modern computer CPU core, running thousands of them in parallel results in the much faster processing of photons from the same pool on the GPU.

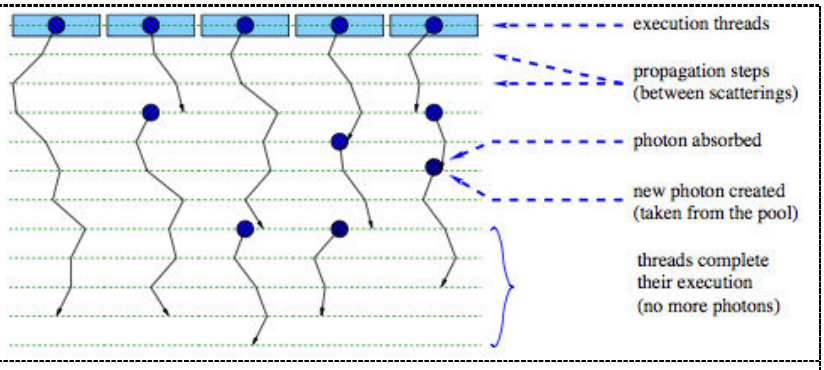

Figure 5. Parallel nature of the photon propagation simulation: tracking of photons entails the computationally identical steps: propagation to the next scatter, calculation of the new direction after scatter, and evaluation of inter- section points of the photon track segment with the detector array. These same steps are computed simultaneously for thousands of photons.

5. Simulation with photon propagation code

The direct photon simulation with PPC is typically used in the two scenarios shown in Figure 6. In the first the in-situ light sources of the detector are simulated for calibrating the detector and the properties of the surrounding ice. It is possible to very quickly re-simulate the detector response to a variety of ice scattering and absorption coefficients finely tabulated in depth bins. This allows for these coefficients to be fit directly, by finding the combination that is a best simultaneous fit to all of the in-situ light source calibration data [6]. For the 10 meter depth bins, 200 coefficients are fitted (with scattering and absorption defined in 100 layers spanning 1 km of depth of the detector), with nearly a million possible ice parameter configurations tested in less than a week on a single GPU-enabled computer. This method is intractable with the photonics-based simulation, as each new parameter set would require generation of the new set of photonics tables, each generation taking on the order of a week of computing time of a ∼100-CPU cluster.

In the second scenario the Cerenkov photons created by the passing muons and cascades are simulated as part of the larger simulation of the detector response to atmospheric and other fluxes of muons and neutrinos. The simulation is able to account for some effects that are difficult to implement with the photonics-based simulation, because their simulation would lead to additional degrees of freedom, thus increasing the size of the parametrization effort and tables many-fold. One of these is the tilt of the ice layers, i.e., dependence of the ice parameters not only on the depth, but also on the xy surface coordinates (shown in Figure 7).

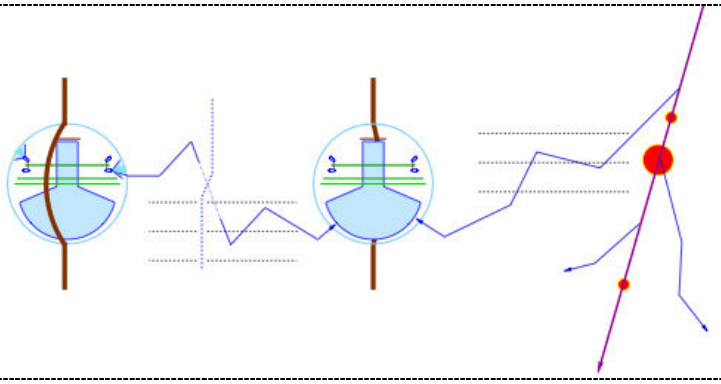

Figure 6. Typical simulation scenarios: photons emitted by the detector are tracked as part of the calibration procedure (left). Cerenkov photons emitted by a passing muon and cascades along its track are tracked to simulate the typical IceCube events (right).

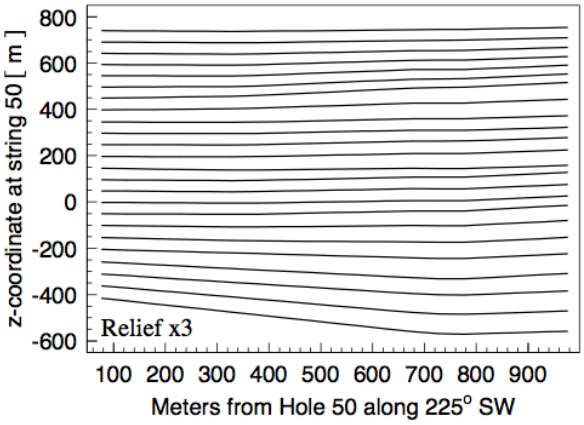

Figure 7. Extension of ice layers along the average gradient direction (taken from [6]). The y-axis shows the layer shift (relief) from its position at the location of a reference string at the distance shown on the x-axis from this string along the average gradient direction (225 degrees SW). The relief shown is amplified by a factor of 3 for visual clarity of the ice layer tilt.

Effects that are treated precisely with PPC (and only approximately with photonics) include the simulation of the longitudinal profile of light generation by cascades and the angular distribution of the Cerenkov photons around the emitting muons or cascades.

Other effects implemented recently only into PPC include the direct simulation of the somewhat different ice properties in the column of ice refrozen around the detector strings after they have been deployed; and the slight azimuthal dependence of the scattering function.

6. Concurrent execution and runtime optimization

PPC keeps track of several execution time counters that help judge the performance of the GPU code. One counter operates on the CPU side, by calculating the time elapsed waiting for the GPU code to return. While waiting for the GPU code the CPU can run other parts of the simulation (e.g., trigger simulation), but to maximize the use of the GPUs these parts should finish quicker than the GPU part.

The typical utilization of the GPUs achieved in our tests is > 90%. The implementation of the concurrent execution of the program on both CPU and GPU sides is facilitated by the fact that the GPU side works on its own memory buffer, which is exchanged with the main computer memory buffer only at the beginning or at the end of the execution on the GPU side. So, while the CPU processes photon detector hits created by the previous run on the GPU, the GPU is running through the photons previously prepared by the CPU (see Figure 8).

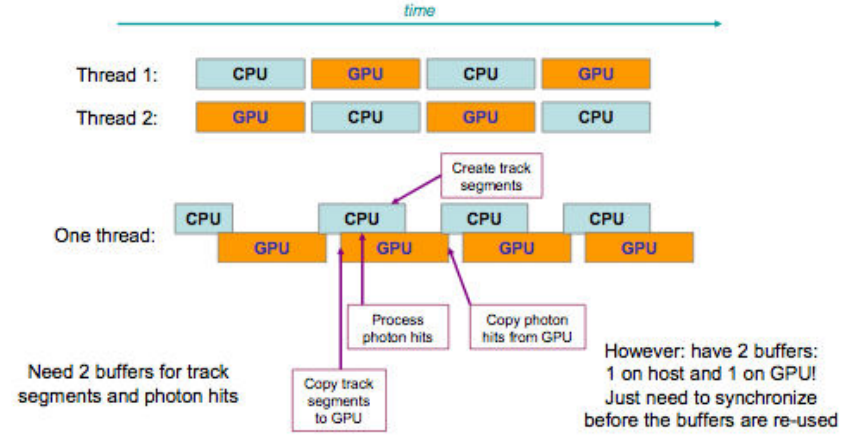

Figure 8. Concurrent execution on CPU and GPU sides. Two possible solutions are shown: the first (top) requires running at least two ppc threads was eventually replaced with the solution (bottom) that is enabled in a single thread.

The other two execution time counters calculate the minimum and maximum time spent by different threads running on the GPU. If these are close to each other, all execution units of the GPU have been equally loaded. The difference we observe is typically on the order of ~ 0.5%.

7. Conclusions

With the addition of PPC to the IceCube simulation chain and the purchase of additional GPU resources, simulation production is able to harness GRID resources more effectively. New ideas such as changes to the ice model can be added into the simulation chain quickly without the lead time necessary to create new photonics tables and propagate them throughout IceCube's distributed computing resources. These enhancements have also freed up space on the local cluster to allow for more analysis and freed up money for the purchase of more GPU resources. Detailed analysis of PPC's usage of the GPU hardware is currently being done. This will help determine what specifications are important when purchasing new GPU resources and could help reduce costs and increase the number of GPUs in the cluster, if expensive components are shown to be unused. As software increases the abstraction of GPUs physicists will be able to move more of their analysis to faster GPU resources. This will ultimately lead to faster results and the ability to make more changes quickly in a fast paced physics environment.

8. Acknowledgments

Funding by the National Science Foundation provides support for the IceCube Project including research done in GPGPU computing clusters for extensive MC production. I would like to thank those working in the IceCube Collaboration for their help and support. This research in GPU computing could not have been done without their extensive knowledge and expertise.

References

[1] Condor DAGMan http://research.cs.wisc.edu/condor/dagman/

[2] D. Chirkin, photon propagation code: http://icecube.wisc.edu/~dima/work/WISC/ppc.

[3] T. Abuzayyad, i3mcml: http://wiki.icecube.wisc.edu/index.php/i3mcml.

[4] NVIDIA CUDA: http://www.nvidia.com/cuda.

[5] OpenCL: http://www.khronos.org/opencl/.

[6] D. Chirkin, et al., Study of south pole ice transparency with icecube flashers, Contribution to the 32st International Cosmic Ray Conference, Beijing, China, 11-18 August 2011, session HI.2.3, contribution 0333.