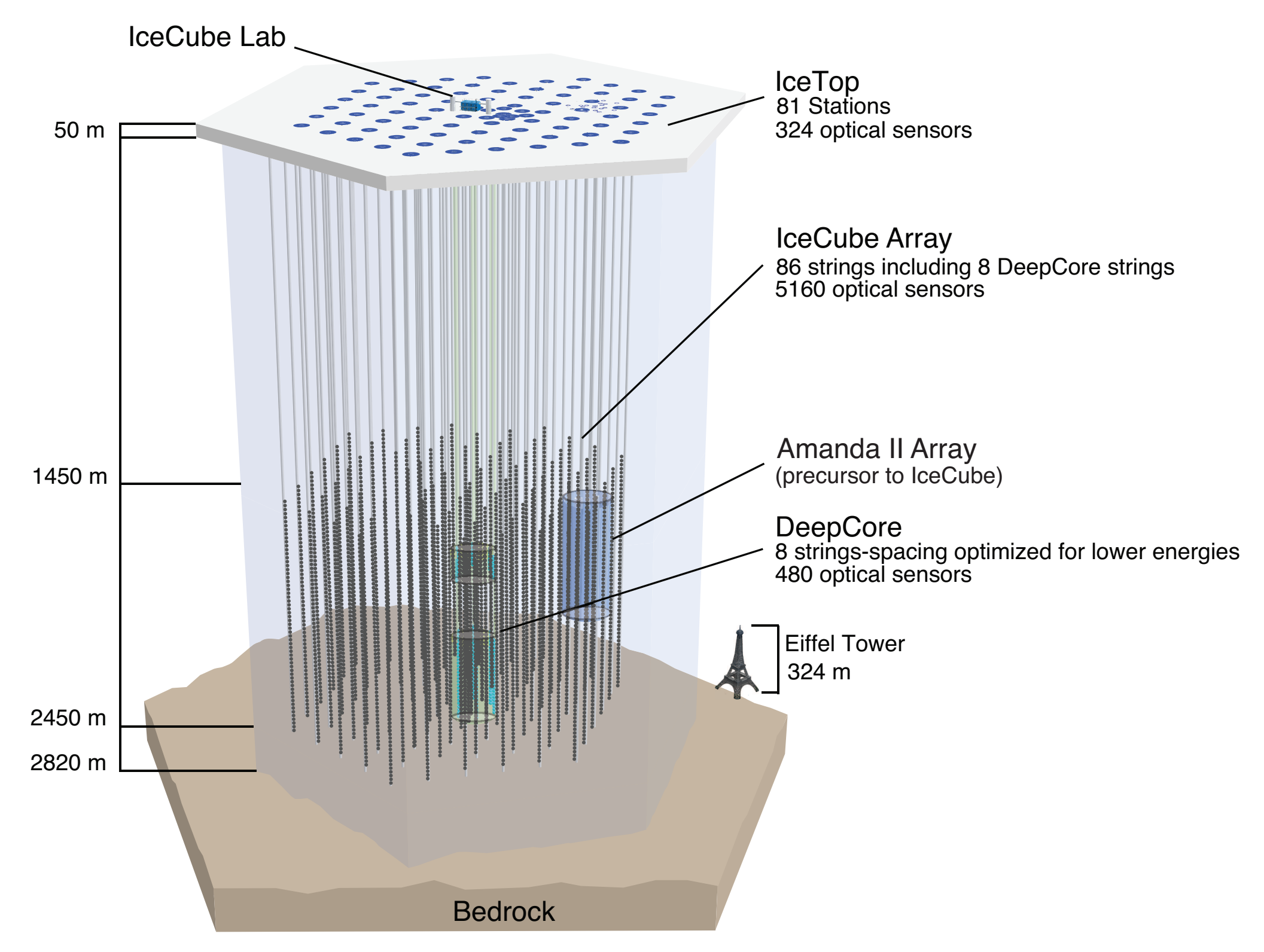
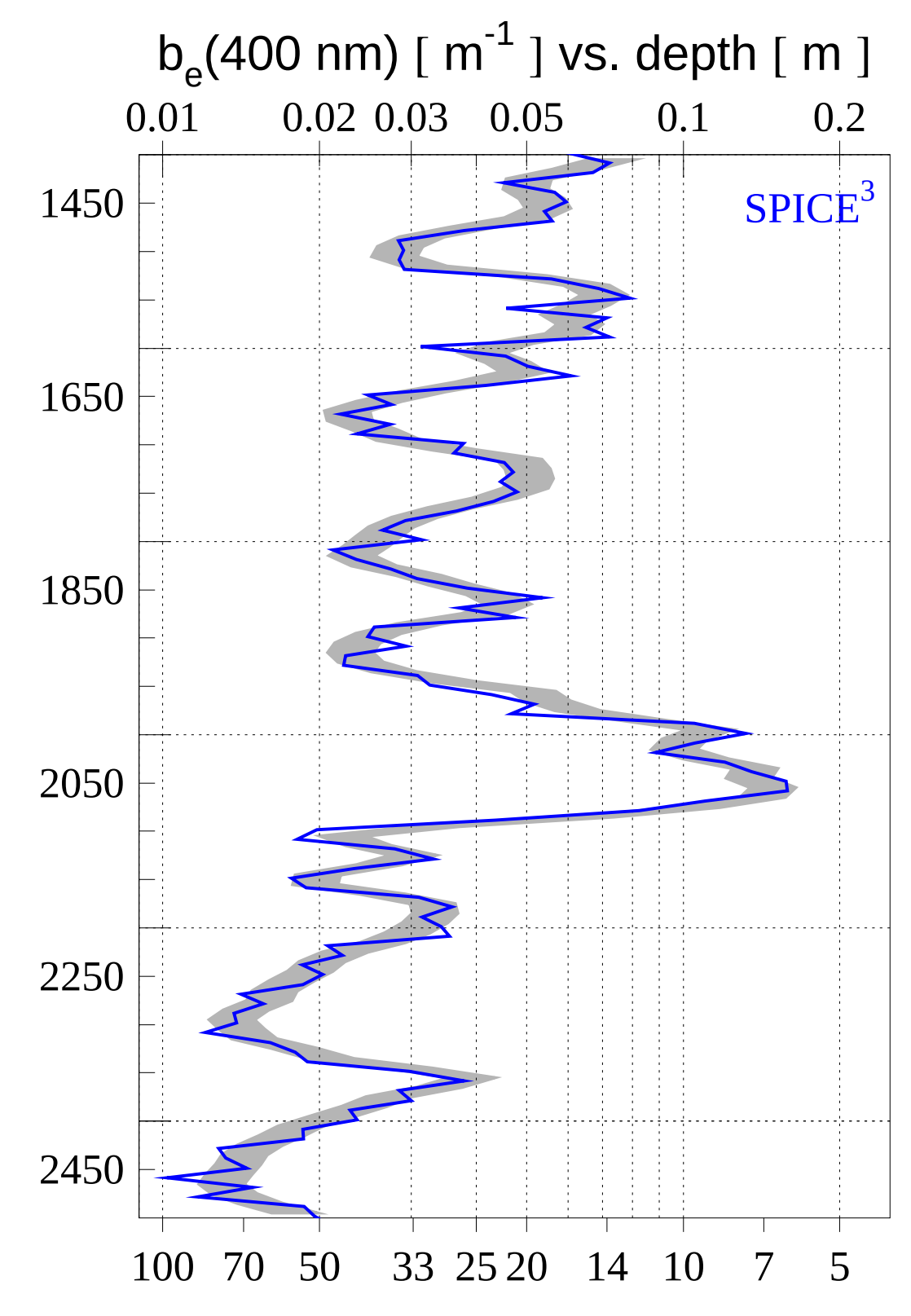
Photon Propagation with GPUs in IceCube

The IceCube collaboration
Dmitry Chirkin (dima@icecube.wisc.edu) University of Wisconsin, Madison, U.S.A.

Introduction

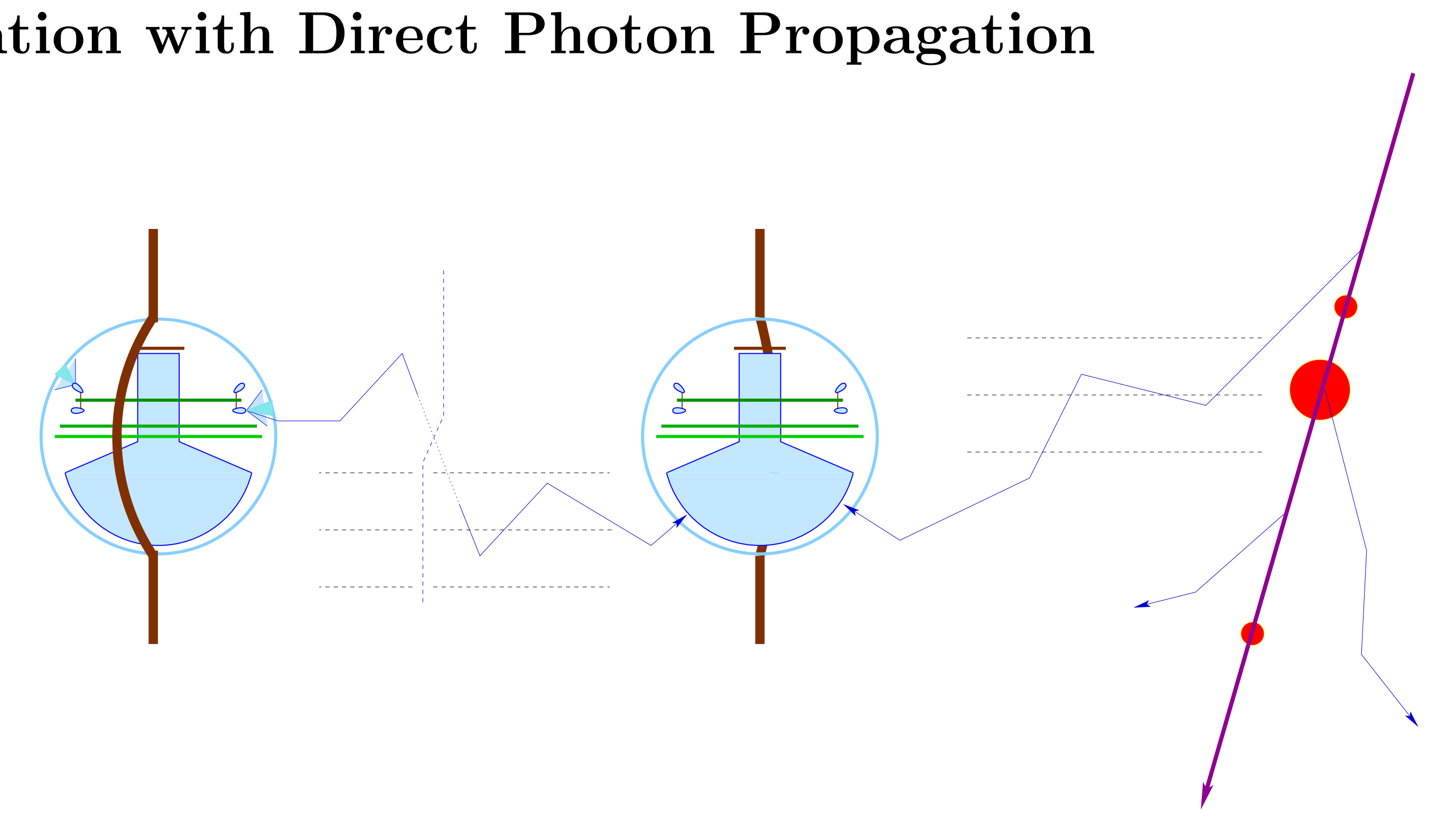
Describing propagation of a large number of photons in a transparent medium is a computational problem of a highly parallel nature. All of the simulated photons go through the same stages: they are emitted, they may scatter a few times, and they get absorbed. These steps, when performed in parallel on a large number of photons, can be done very efficiently on a GPU. The IceCube collaboration uses parallelized code that runs on both GPUs and CPUs to simulate photon propagation in a variety of settings, with significant gains in precision and, in many cases, speed of the simulation compared to the table lookup-based code. The same code is also used for the detector medium calibration and as a part of an event reconstruction tool. I will describe the code and discuss some if its applications within our collaboration.





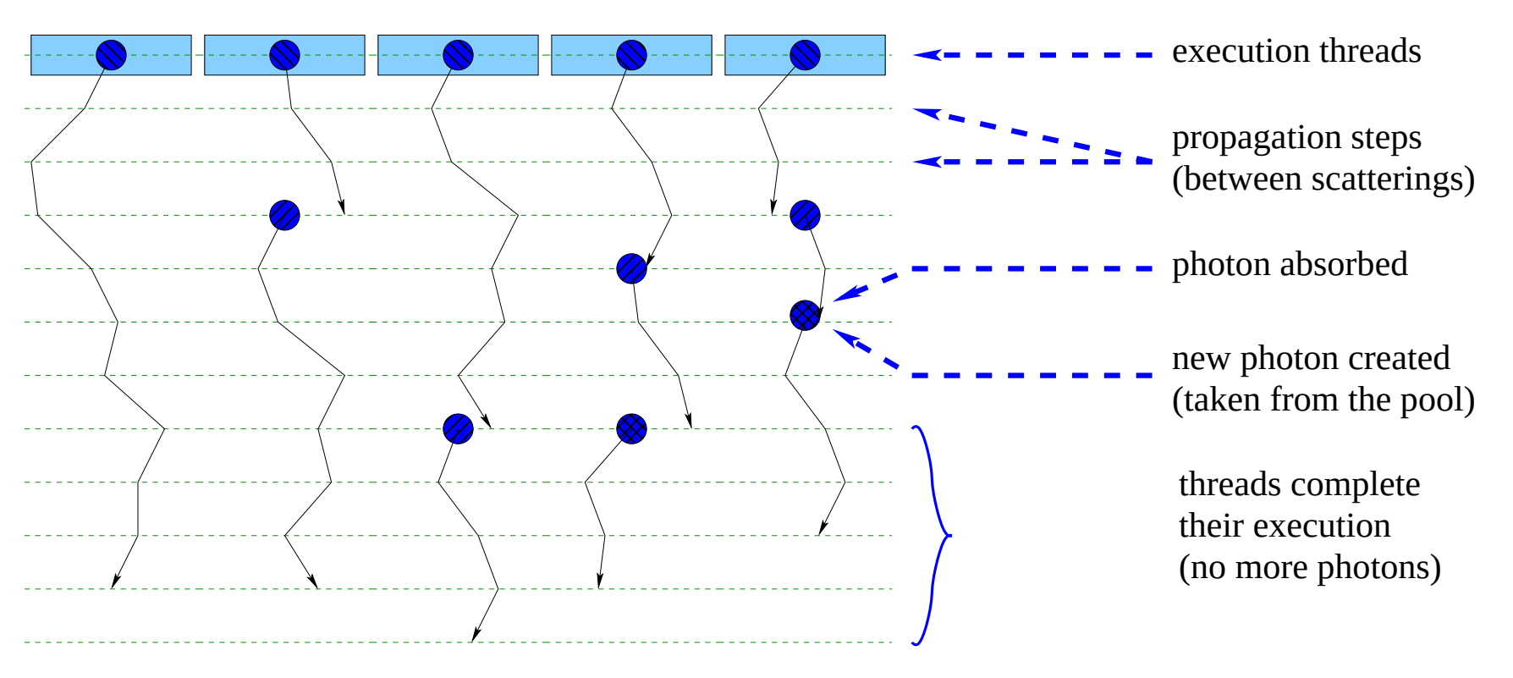
IceCube is a cubic-kilometer-scale high-energy neutrino observatory built at the geographic South Pole. IceCube uses the 2.8 km thick glacial ice sheet as a medium for producing Cerenkov light emitted by charged particles created when neutrinos interact in the ice or nearby rock. Ice transparency in the active volume of the detector is highly variable: plot on the right shows the effective scattering length, varying from 8 to 80 m.

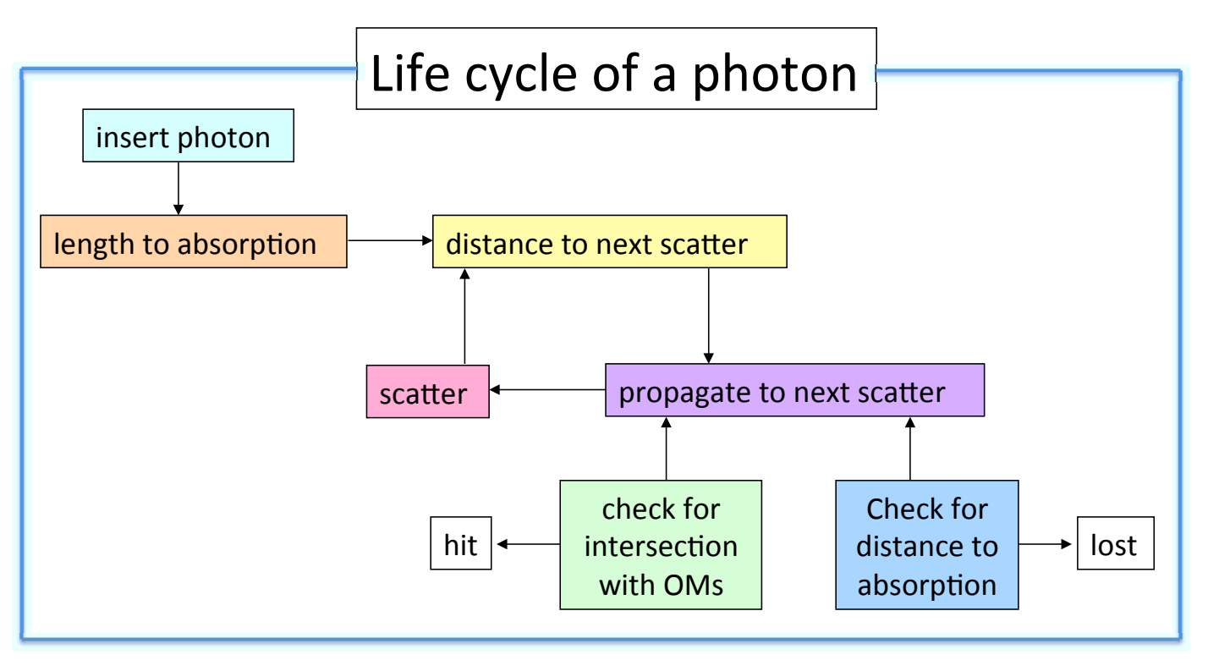
Simulation with Direct Photon Propagation



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).

Direct Photon Propagation: implementation details

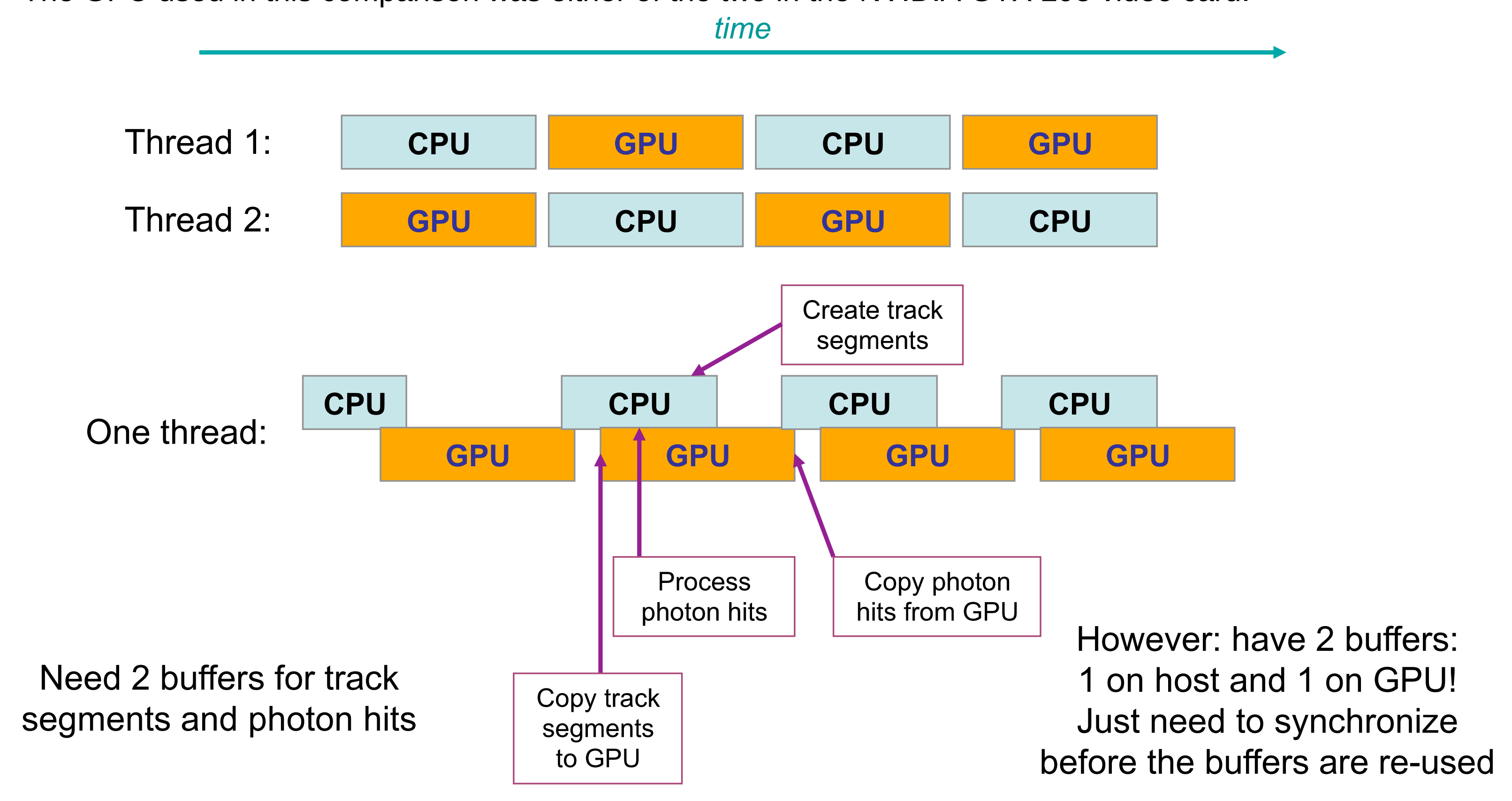




Parallel nature of the photon propagation simulation: tracking of photons entails the computationally identical steps: propagation to the next scatter, calculation of the new direction after scatter, and evaluation of intersection points of the photon track segment with the detector array. These same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

	C++	Assembly	GTX 295 GPU
flasher	1.00	1.25	147
muon	1.00	1.37	157

Speedup factor of different implementations of Photon Propagation Code (PPC) compared to the C++ version. The GPU used in this comparison was either of the two in the NVIDIA GTX 295 video card.

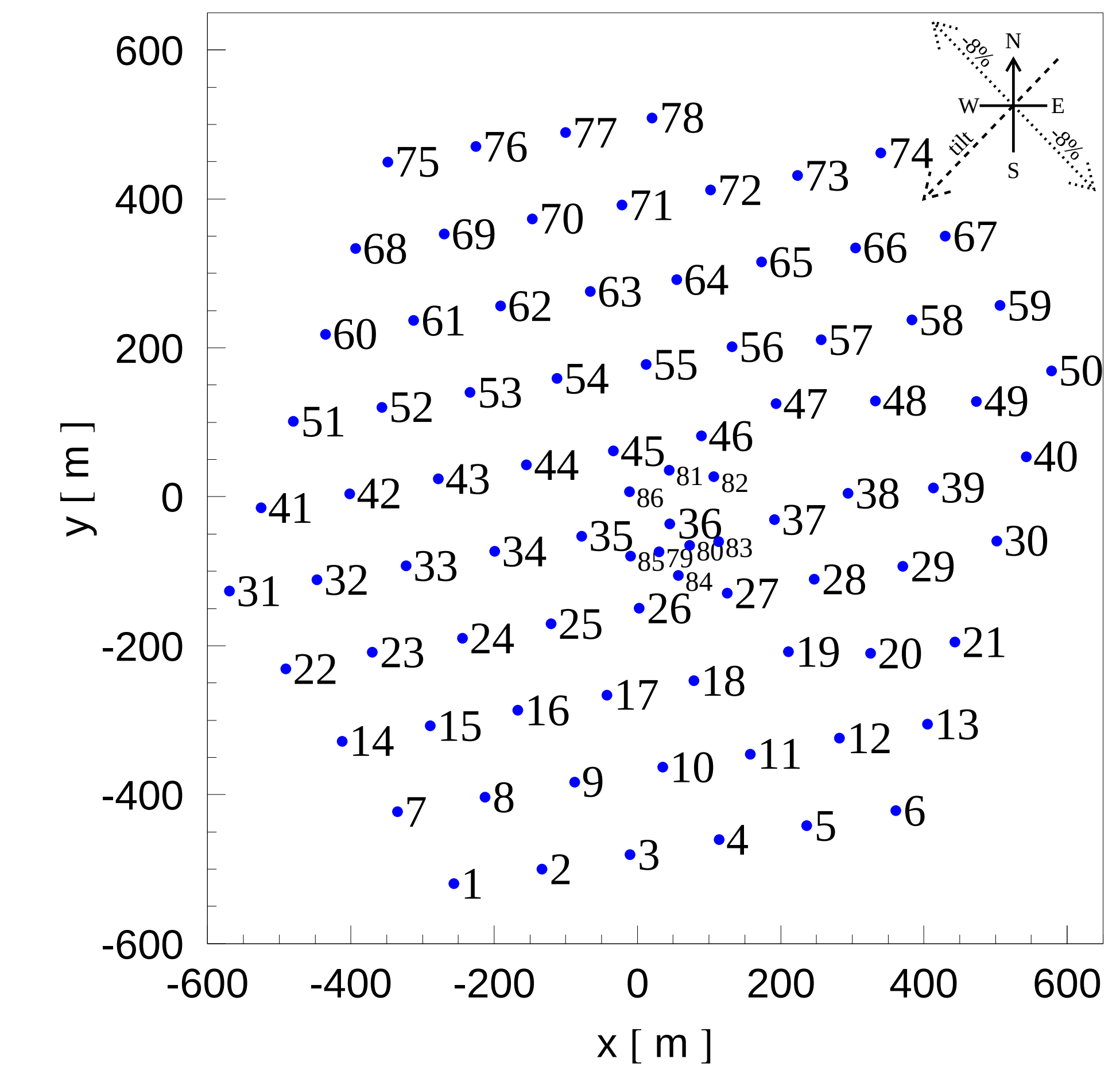

time
Thread 1:
CPU
GPU
CPU
GPU
Thread 2:
GPU
CPU
GPU
CPU
One thread:
CPU
GPU
Create track segments
Process photon hits
Copy photon hits from GPU
Copy track segments to GPU
Need 2 buffers for track segments and photon hits
However: have 2 buffers: 1 on host and 1 on GPU! Just need to synchronize before the buffers are re-used

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.

Faulty cards (CUDA)

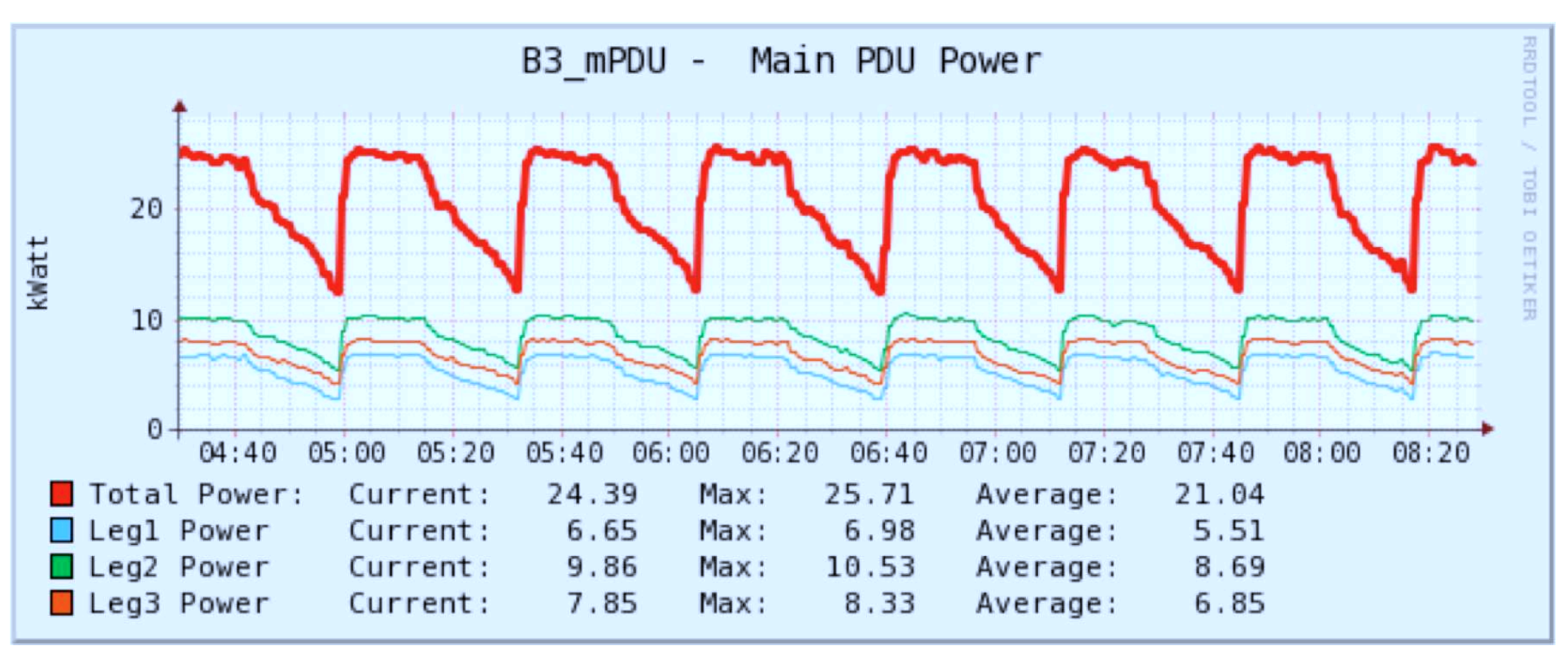
Unfortunately we experienced problems with some of the GPUs (jobs exiting with errors, GPUs or even entire computer locking up and requiring rebooting). With a small bit of in-line assembly we found that only threads running on specific hardware multiprocessors (MPs) on each of the faulty GPUs are affected (producing NANs). By checking within threads during execution whether they are running on the faulty MP and immediately stopping those threads it was possible to significantly prolong the use the faulty GPUs albeit at sligtly reduced capacity.

Only with Direct Photon Propagation



Geometry of IceCube when viewed from above. Effects that are only taken into account with simulation described here are:

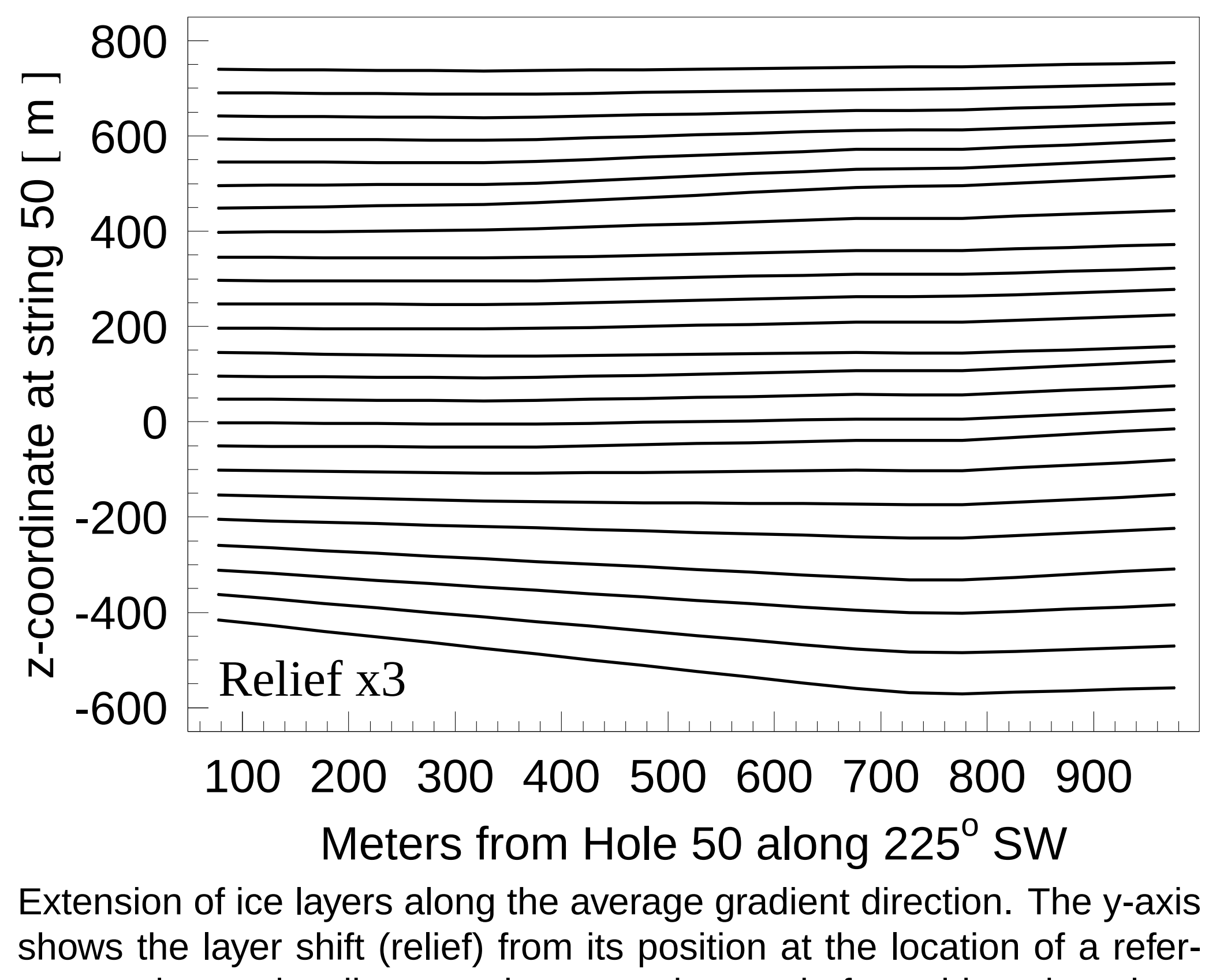
- *ice layer tilt*: a change in ice layer (layer of approximately the same transparency) elevation
- *ice anisotropy*: effective scattering is 8% smaller in one preferred direction, and up to 4% greater in other directions



```
■ Total Power:   Current:   24.39   Max:   25.71   Average:   21.04
■ Leg1 Power     Current:    6.65   Max:    6.98   Average:    5.51
■ Leg2 Power     Current:    9.86   Max:   10.53   Average:    8.69
■ Leg3 Power     Current:    7.85   Max:    8.33   Average:    6.85
```

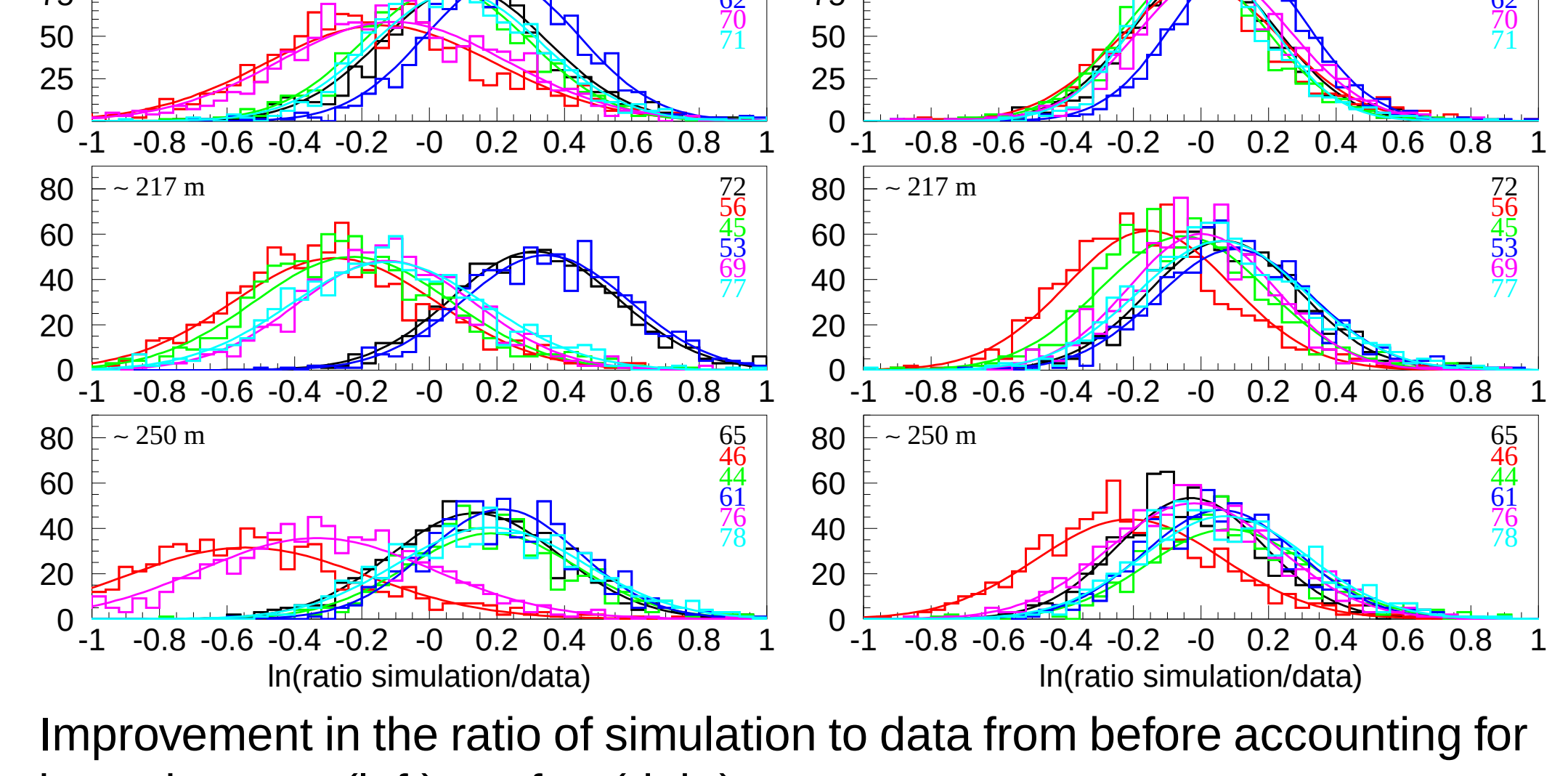
Power usage by the GPU cluster (about 90 gpu nodes) while fitting ice properties to in-situ light calibration data.

Ice Layer Tilt



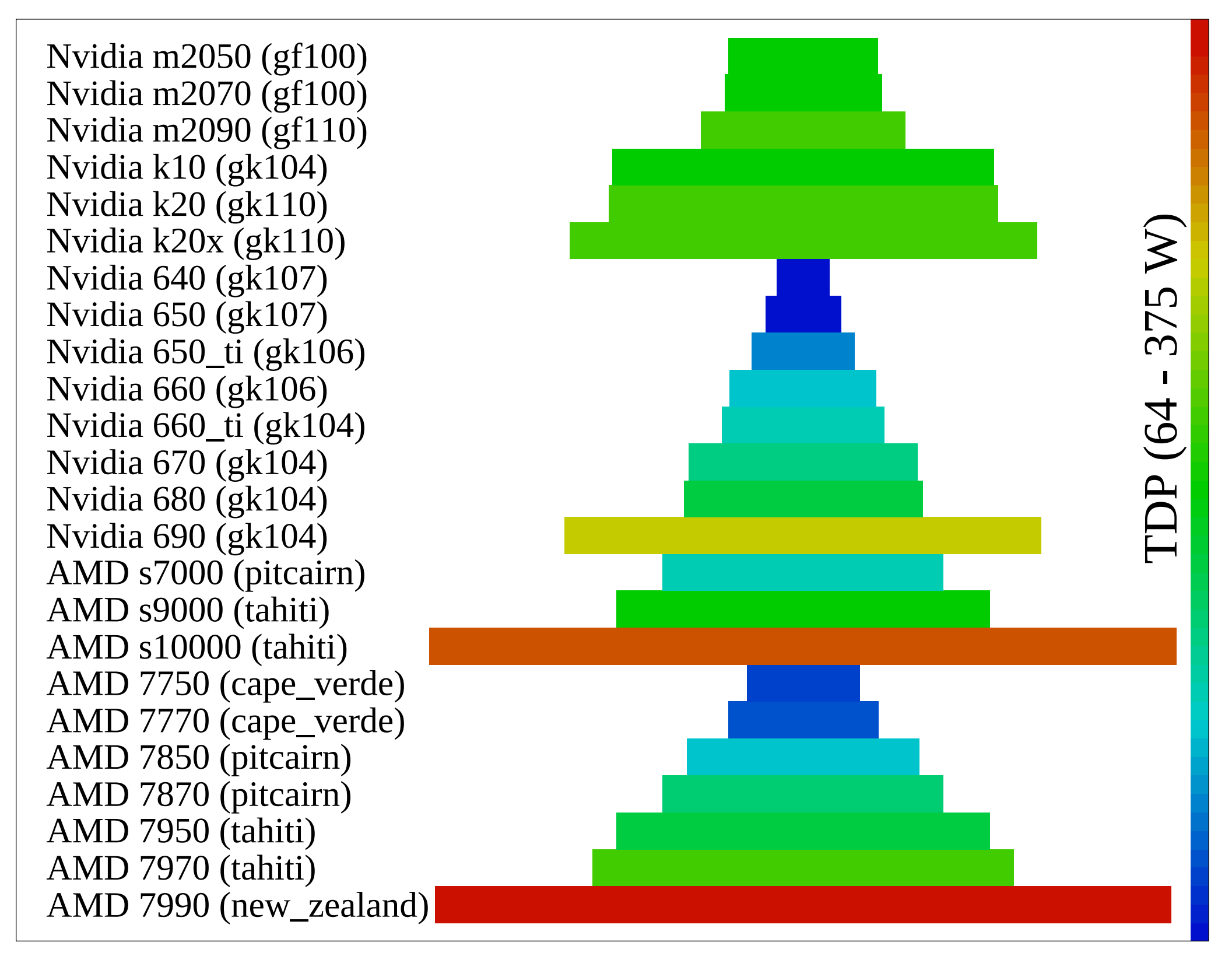
Extension of ice layers along the average gradient direction. The y-axis shows the layer shift (relief) from its position at the location of a reference string at the distance shown on the x-axis from this string along the average gradient direction (225 degrees SW). The relief shown is amplified by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

Ice Anisotropy



Improvement in the ratio of simulation to data from before accounting for ice anisotropy (left) to after (right).

Relative performance of GPU cards



Relative performance of tested GPU cards: longer bars indicated better performance. Also shown in color is the TDP (manufacturer specified maximum power consumption).

References

[1] Measurement of South Pole ice transparency with the IceCube LED calibration system, NIM-A (2013) pp. 73-89, arXiv:1301.5361

[2] Study of South Pole ice transparency with IceCube flashers, 32nd ICRC, Beijing, China, arXiv:1111.2731

[3] Photon tracking with GPUs in IceCube, NIM-A, as part of VLVNT proceedings, DOI: 10.1016/j.nima.2012.11.170

[4] Evidence of optical anisotropy of the South Pole ice, 33rd ICRC, Rio de Janeiro (arXiv:1309.7010)

[5] The IceProd Framework: Distributed Data Processing for the IceCube Neutrino Observatory, Journal of Parallel and Distributed Computing (arXiv:1311.5904)