GPU-Based Monte Carlo Simulation of Light Transport in Tissue

A Chapel Implementation

Engin Kayraklioglu
engin@hpe.com
Hewlett Packard Enterprise

Hui Wang
hui.wang@miamioh.edu
Hewlett Packard Enterprise, Miami University

What is Chapel?

A modern parallel programming language:

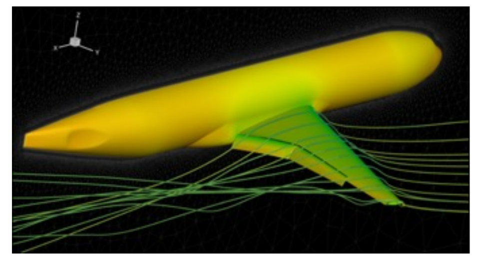
- portable & scalable, and
- open-source & collaborative

chapel-lang.org

Goals:

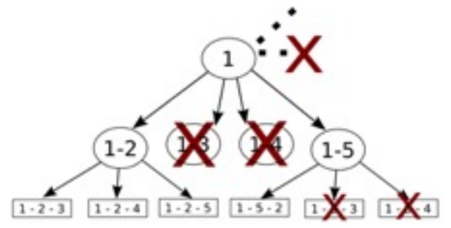
- Support general parallel programming
- Make parallel programming at scale far more productive

Applications using Chapel's GPU Support



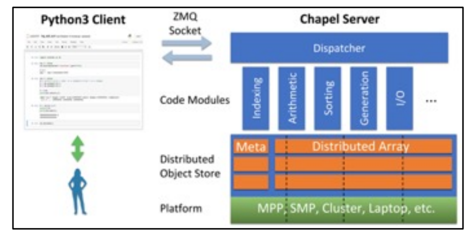
CHAMPS: Chapel Multiphysics Simulation
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal

Computational fluid dynamics for aerospace simulations



ChOP: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.

Combinatorial optimization

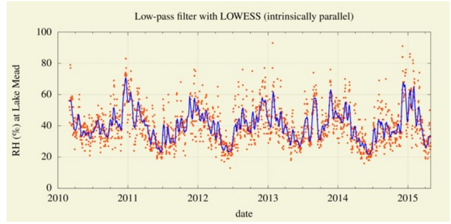


Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD

Exploratory data analytics

RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance

Satellite image analysis for coral reef biodiversity

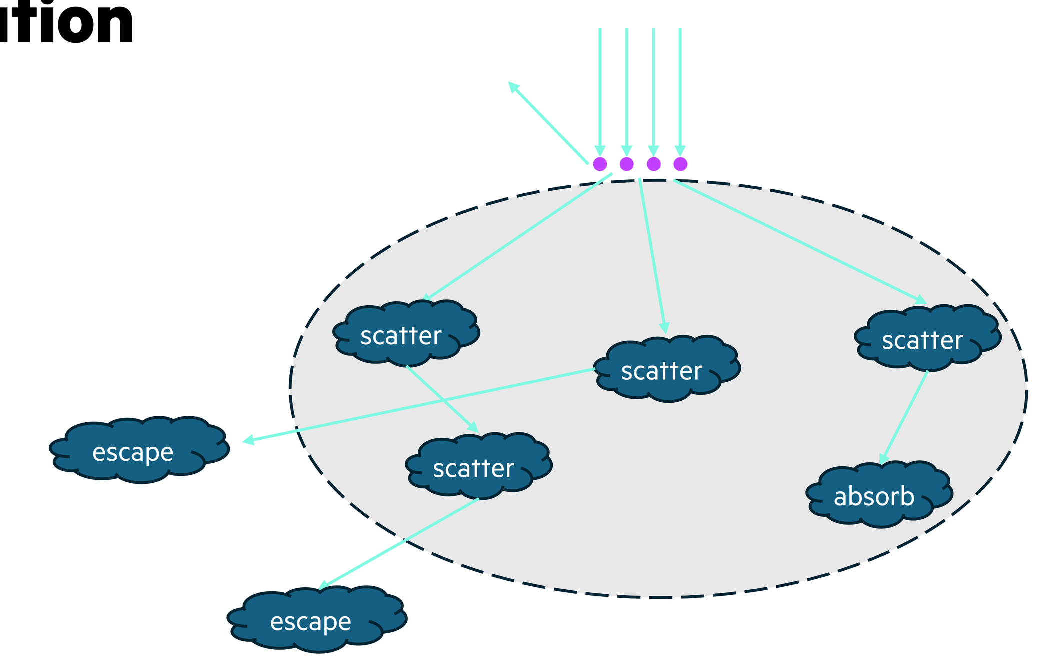


Desk.chpl
Nelson Luis Dias
The Federal University of Paraná, Brazil

Utilities for environmental engineering

(images provided by their respective teams and used with permission)

Problem & Solution



Problem

Optical imaging is key in the diagnosis of many disorders. Developing better optical imaging technologies requires improving our understanding of the behavior of photons propagating through tissue. An accurate simulation of this behavior relies on the ability to **simulate a massive number of photons with stochastic behavior**.

Simulation in a Nutshell [1]

- Photons enter the tissue,
  - They get **scattered** and **absorbed** during their interactions with it
  - Each event (scattering or absorption) is decided randomly
- This simulates the inherently stochastic behavior of photons
- Photons propagate independent of one another: *a parallelizable operation*

How Does Chapel Help?

- The parallel-first design provides intuitive features for any kind of parallelism, including GPUs
  - The `on` statement moves the execution and allocation to the GPU sublocale
  - The order-independent `foreach` loop is automatically launched as a kernel
  - Same code runs on both GPUs and CPUs
- Each photon is propagated as long as its alive
  - Random events use the `curand_kernel` interface for RNG
  - We are planning to incorporate `curand_kernel` into Chapel's Random module
  - Currently, the implementation relies on Chapel's strong C interoperability

Main part of the implementation is concise and intuitive:

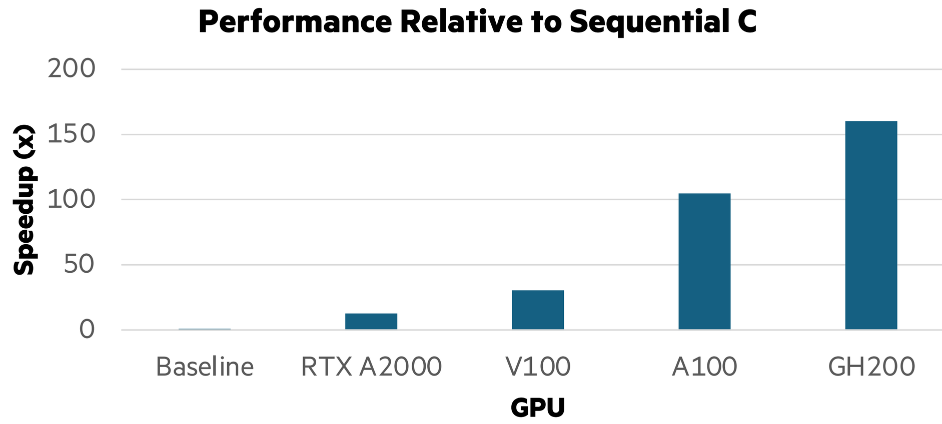
```
on if useGpu then here.gpus[0] else here {
  var Csph, Ccyl, Cpla: [0..100] real;   // photon concentration

  foreach thread in 0..<numGpuThreads {  // this will be a kernel
    var rng = new RNG(thread);     // curand_kernel is used under the hood

    for i_photon in 0..<NphotonsPerGpu {
      var p = new photon(rng);
      do {   // simulate a photon as long as it is alive
        p.hop(rng);
        p.drop();

        gpuAtomicAdd(Csph[p.spherical()], p.absorb);
        gpuAtomicAdd(Ccyl[p.cylindrical()], p.absorb);
        gpuAtomicAdd(Cpla[p.planar()], p.absorb);

        p.spin(rng);
        p.update(rng);
      } while (p.photon_status == ALIVE);
    }
  }
}
```

https://github.com/e-kayrakli/mc321.chpl/releases/tag/nvidia-gtc

How does it perform?

- Performance is measured on different NVIDIA GPUs using Chapel 2.2
- 1 billion photons are simulated using 1 million GPU threads
- 4 different NVIDIA GPUs deliver varying levels of speedup
  - ranging from **12.5x** to **160.1x**

	Time (s)	Throughput (Mphotons/s)	Relative Performance
Sequential C	83.57	11.97	**1.00**
RTX A2000 w/Chapel	6.67	149.94	**12.53**
V100 w/Chapel	2.74	365.55	**30.54**
A100 w/Chapel	0.80	1254.21	**104.78**
GH200 w/Chapel	0.52	1916.49	**160.11**



Chapel delivers significant performance improvement on NVIDIA GPUs with intuitive code.

Conclusion

NVIDIA GPUs and CUDA stack make up the backbone

- can deliver more than 2 orders of speedup
- curand_kernel is key in delivering this speedup

Chapel provides an easy way of using NVIDIA GPUs

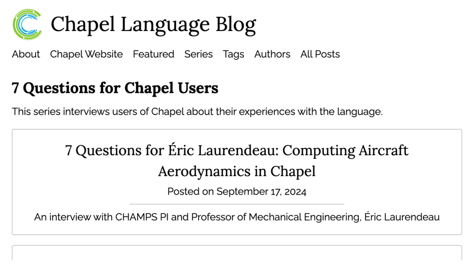
- easy to develop and maintain the code
- same set of features can be used for
  - multiple-GPU, and
  - multiple-node parallelism

What is next?

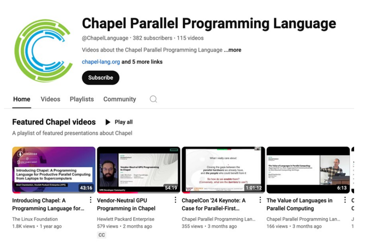
- Track the photon traces for further analysis
- Simulate
  - more realistic situations like Gaussian beam
  - computation intensive situations such as electronic field MC
- Multi-node / multi-GPU performance analysis and optimization

More on Chapel

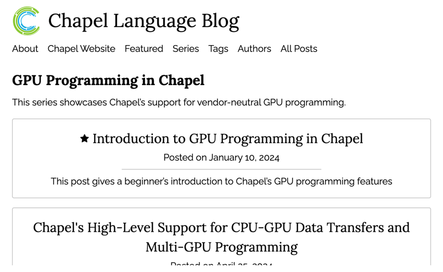
7 Questions with Chapel Users



Tutorials, Demos and Talks on YouTube



GPU Programming Blog Series



Social Media

- @ChapelLanguage
- @ChapelProgrammingLanguage
- @ChapelLanguage
- @ChapelLanguage
- r/chapel

Acknowledgement & References

The authors would like to thank Jade Abraham for their assistance in collecting performance numbers on GH200.

[1] The implementation is based on https://omlc.org/software/mc/mc321/index.html

Contact Info