



# **A tool for top-down performance analysis of GPU- accelerated applications**

Keren Zhou, Mark Krentel, John Mellor-Crummey

Rice University



- Problem:
  - Performance tools for GPU-accelerated programs typically lack a comprehensive profile view
- Rice University's HPCToolkit has the following capabilities:
  - It unwinds the CPU call stack to identify the **CPU calling context** for each GPU API invocation;
  - It employs **a novel and fast wait-free data structure** to correlate performance metrics for each asynchronous GPU API invocation with its CPU calling context;
  - It uses a novel technique to reconstruct an approximate **GPU calling context tree** for computations from flat GPU PC samples;
  - It derives a rich set of metrics from PC samples gathered during **a single execution**.

# HPCToolkit Profile View



NuclearData.cc

```

246 // Return the total cross section for this energy group
247 HOST_DEVICE
248 double NuclearData::getReactionCrossSection(
249     unsigned int reactIndex, unsigned int isotopeIndex, unsigned int group)
250 {
251     qs_assert(isotopeIndex < _isotopes.size());
252     qs_assert(reactIndex < _isotopes[isotopeIndex].species[0].reactions.size());
253     return isotopes[isotopeIndex].species[0].reactions[reactIndex].getCrossSection(group);
254 }
                
```

Top-down view Bottom-up view Flat view

Scope	GINS:Sum (l)	GINS:STL_ANY:Sum (l)
loop at main.cc: 159	1.07e+11 100 %	9.83e+10 100 %
loop at main.cc: 163	1.07e+11 100 %	9.83e+10 100 %
↳ 193: [I] CycleTrackingKernel(MonteCarlo*, int, ParticleVault*, ParticleVault*)	1.07e+11 100 %	9.83e+10 100 %
↳ 127: _device_stub_Z19CycleTrackingKernelP10MonteCarloiP13ParticleVaultS2_(MonteCarlo*, int, Parti	1.07e+11 100 %	9.83e+10 100 %
<b>CPU Calling Context</b> ↳ 14: [I] cudaLaunchKernel<char>	1.07e+11 100 %	9.83e+10 100 %
<b>GPU API Node</b> ↳ 209: <gpu kernel>	1.07e+11 100 %	9.83e+10 100 %
↳ 174: CycleTrackingKernel(MonteCarlo*, int, ParticleVault*, ParticleVault*)	1.07e+11 100 %	9.83e+10 100 %
↳ 132: CycleTrackingGuts(MonteCarlo*, int, ParticleVault*, ParticleVault*)	1.06e+11 100.0	9.82e+10 100.0
↳ loop at CycleTracking.cc: 118	8.90e+10 83.5%	8.08e+10 82.2%
↳ 63: CollisionEvent(MonteCarlo*, MC_Particle&, unsigned int)	4.99e+10 46.9%	4.49e+10 45.7%
[I] inlined from QS_Vector.hh: 94	3.76e+10 35.3%	3.34e+10 34.0%
↳ loop at QS_Vector.hh: 94	3.61e+10 33.9%	3.20e+10 32.5%
[I] inlined from CollisionEvent.cc: 71	3.58e+10 33.6%	3.17e+10 32.3%
↳ loop at CollisionEvent.cc: 71	3.42e+10 32.1%	3.03e+10 30.9%
<b>GPU Loops and Inline Functions</b> ↳ 73: macroscopicCrossSection(MonteCarlo*, int, int, int, int, int)	3.11e+10 29.2%	2.78e+10 28.3%
[I] inlined from MacroscopicCrossSection.cc: 45	2.57e+10 24.1%	2.31e+10 23.5%
↳ 41: NuclearData::getReactionCrossSection(unsigned int, unsigned	1.69e+10 15.9%	1.56e+10 15.9%
<b>GPU Calling Context</b> [I] inlined from NuclearData.cc: 194	9.36e+09 8.8%	8.70e+09 8.9%
<b>GPU Hotspot</b> NuclearData.cc: 253	9.06e+09 8.5%	8.44e+09 8.6%