Accelerating Event Generation with GPUs

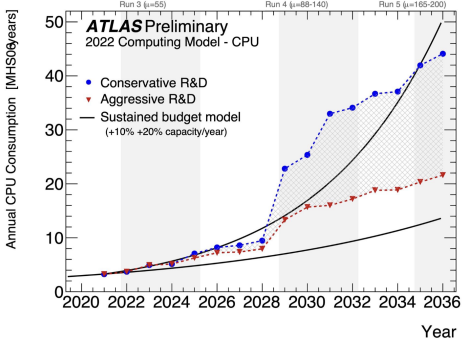
Siddharth Sule (Sid)
Supervised by Mike Seymour

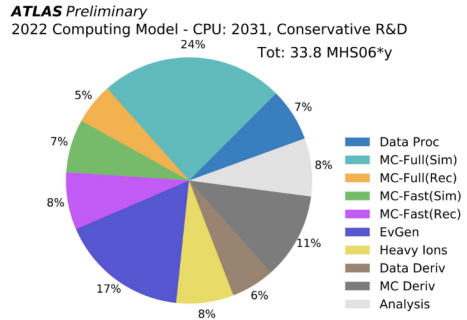
UK HEP Forum, 26th November 2024

Science and Technology Facilities Council

Event Generation is Expensive

CPU usage to surpass sustainable budget. Majority is MC, of which EvGen contributes 17% [CERN-LHCC-2022-005].





Event Generation is Expensive

Two ways to reduce CPU usage:

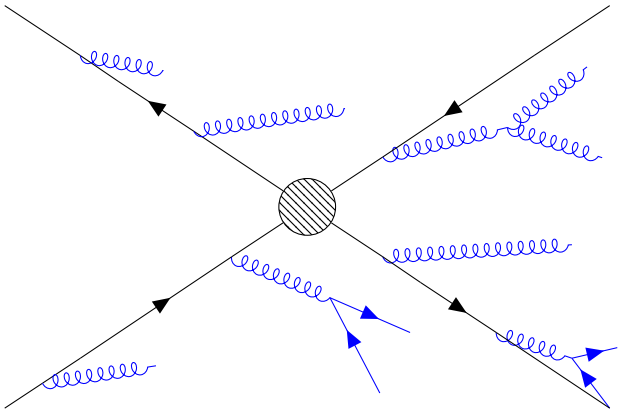
1. Profiling and treating bottlenecks [2209.00843]
2. Parallelising tasks to reduce execution time and cost

We can achieve this parallelisation by using **GPUs**:

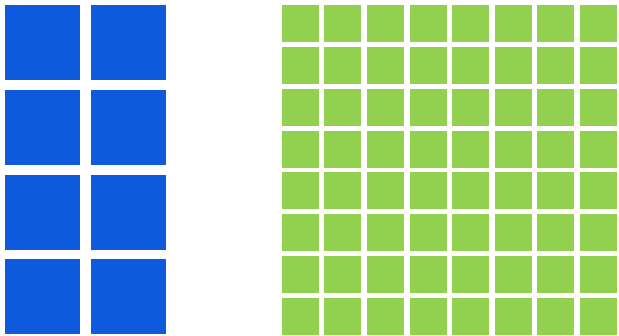
- Matrix Element + Phase Space: PEPPER [2311.06198], MadGraph4GPU [2303.18244]
- PDF Evaluations: LHAPDF [1412.7420, 2311.06198] PDFFlow [2009.06635]

Event Generation is Expensive

BUT Parton Shower, each event behaves uniquely - *How do we do this on a GPU?*



Preliminaries - GPUs and SIMT



Preliminaries - GPUs and SIMT

SIMT = Single Instruction Multiple Threads

CPU: For Loop

```
for i in range(len(a)):
    b[i] = 3*a[i] + 2
```

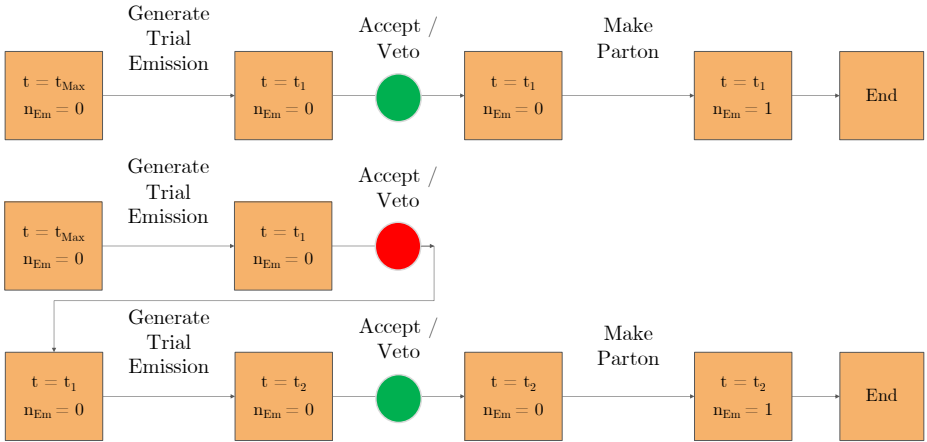
GPU: Threads in a Kernel

```
if thread_i < len(a):
    b[thread_i] = 3*a[thread_i]
        + 2
```

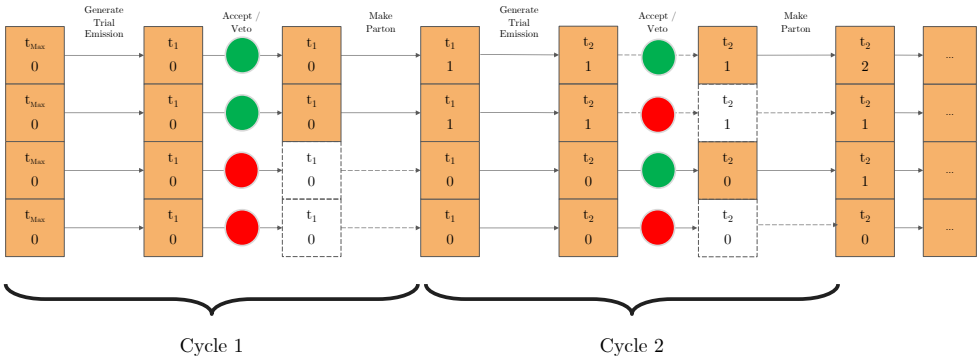
\+ GPUs have $\sim$ 1000 cores, so you can parallelise more tasks than a CPU cluster

– GPU cores aren't as sophisticated as CPU cores; tasks have to be fairly simple

The Parton Shower Veto Algorithm



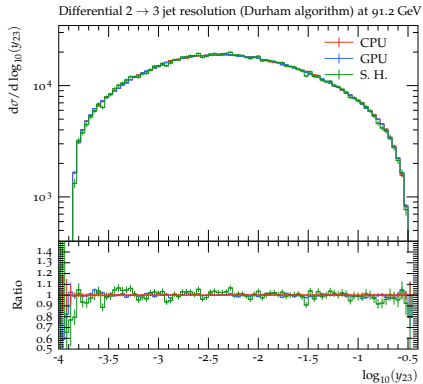
The Parallelised Veto Algorithm



Implementation Overview

LO ME + Final State Parton Shower

- Based on S. Höche's Tutorial [1411.4085]
- Process $e^+e^- \to q\bar{q}$ at 91.2 GeV
- CPU: C++, GPU: CUDA
- Both Codes Validated with Tutorial



Plots made with Rivet [1912.05451]

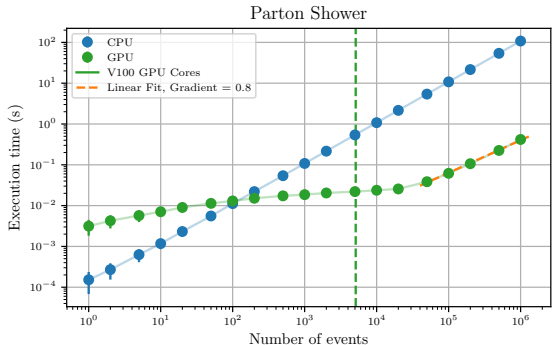
Comparison of Execution Times

We compare **1 CPU Core** against **1 CPU Core + 1 GPU**

- To study if GPUs can replace a CPU Cluster running individual EvGens

Parton Shower, 10^6 events

- GPU: 0.4s, CPU: 148s
- Speedup: 295x



Context - Power Consumption

Type	Name	Cores	Power
CPU	Intel Xeon	16	85 W (5 W per core)
GPU	NVIDIA V100	5000	250 W

Around 19 CPUs (= 304 Cores) needed to match 1 CPU Core + 1 GPU:

- 19 CPUs $\rightarrow$ 1615 W
- 1 CPU Core + 1 GPU $\rightarrow$ 255

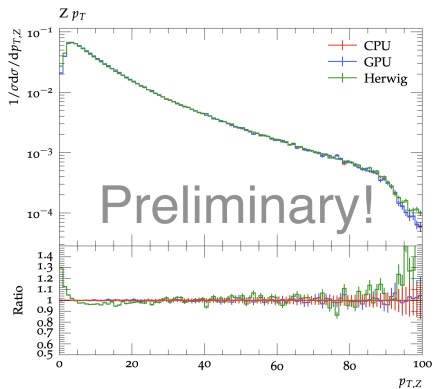
Using GPUs leads to 6x less power consumption!

Up Next: LHC at 13 TeV

Combined with LHAPDF on GPU
(Thanks to M. Knobbe)

Shower speedup at 1,000,000 events: 80x
(45s vs 1h)

Still, need to fine tune the physics!



Plots made with Rivet [1912.05451]

...And That's All!

This talk is based on:
M. H. Seymour and S. Sule, An Algorithm to Parallelise Parton Showers on a GPU, SciPost Physics Codebases 33, 2024 [2403.08692]

Code: gitlab:siddharthsule/gaps (here, v1.1.0)

My details:

- siddharth.sule@manchester.ac.uk
- Office: Room 6.08, Schuster Building, Uni of Manchester, M13 9PL

Backup Slides

For a longer version of this talk...

See Talk: *S. Sule, Parton Showers on GPU with GAPS, Parton Showers and Resummation 2024 (PSR24) Universität Graz, 2nd to 4th July 2024.* [Link]

Example Code Snippet

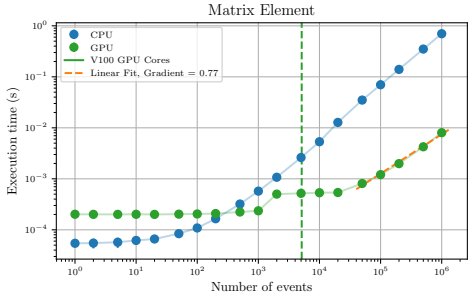
Left: C++, Right: CUDA

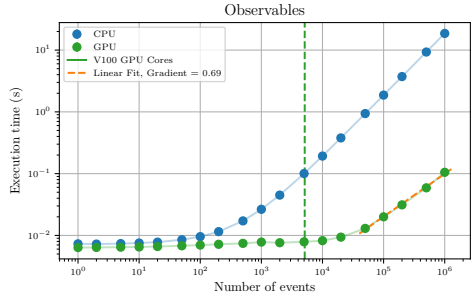
```cpp
double rand = dis(gen);

// Generate z
double zp = ev.GetWinParam(0);
double z = sfGenerateZ(1 - zp, zp, rand, sf);

double y = ev.GetShowerT() / ev.GetWinParam(1) / z / (1. - z);

double f = 0.;
double g = 0.;
double value = 0.;
double estimate = 0.;

// CS Kernel: y can't be 1
if (y < 1.) {
  value = sfValue(z, y, sf);
  estimate = sfEstimate(z, sf);

  f = (1. - y) * as(ev.GetShowerT()) * value;
  g = asmax * estimate;

  if (dis(gen) < f / g) {
    ev.SetShowerZ(z);
    ev.SetShowerY(y);

    double phi = 2. * M_PI * dis(gen);
```

```cpp
double rand = curand_uniform(&state);
states[idx] = state;

// Generate z
double zp = ev.GetWinParam(0);
double z = sfGenerateZ(1 - zp, zp, rand, sf);

double y = ev.GetShowerT() / ev.GetWinParam(1) / z / (1. - z);

double f = 0.;
double g = 0.;
double value = 0.;
double estimate = 0.;

// CS Kernel: y can't be 1
if (y < 1.) {
  value = sfValue(z, y, sf);
  estimate = sfEstimate(z, sf);

  f = (1. - y) * ev.GetAsVeto() * value;
  g = asmax * estimate;

  if (curand_uniform(&state) < f / g) {
    ev.SetAcceptEmission(true);
    ev.SetShowerZ(z);
    ev.SetShowerY(y);
```

Execution Time - ME Observables

Maximum speed up: 87x for ME and 177x for Observables at 1,000,000 events





Maximum Speedup of ME + PS + Ob: 239x for 1,000,000 events (0.5s vs 120s)