MadFlow: Towards hardware acceleration for MG5_aMC

Marco Zaro

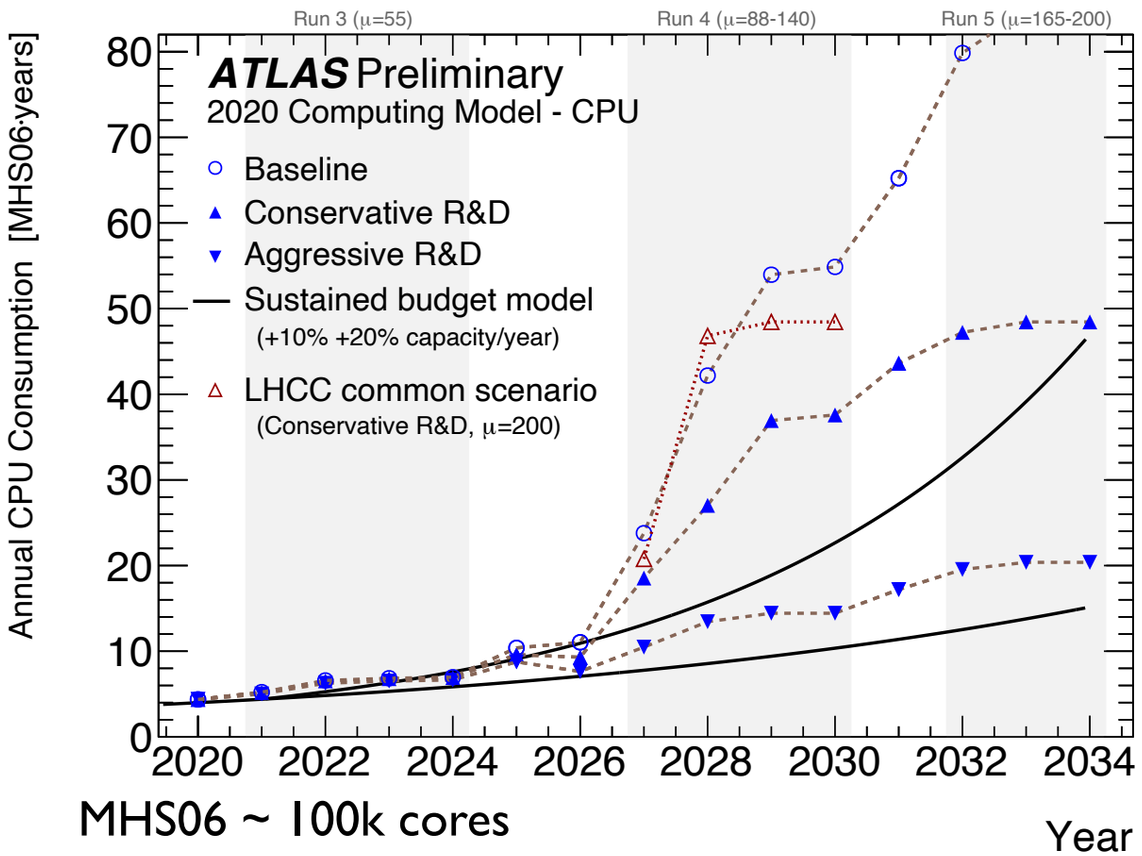
MG5_aMC meeting @Bonn

work in collaboration with: Stefano Carrazza, Juan Cruz-Martinez, Marco Rossi

Motivation

- Coming LHC runs will be extremely demanding from the computational point of view
- Most of the software used for theory simulations is still written using old paradigms (single job on CPU, possibly to be submitted on many cores/nodes)
- Technology has evolved a lot, not only increasing the speed of CPUs
  - Graphic processing units (GPUs) are extremely efficient for low-memory repetitive tasks
  - Field-programmable gate arrays (FPGAs) offer the possibility to customise the hardware according to one's needs

https://cds.cern.ch/record/2729668



Aim

- Develop an event generator that can run efficiently on various hardware platforms (CPU / GPU / FPGA) without being tied to any specific architecture
- Use MG5_aMC as matrix-element provider
- Use TensorFlow primitives for code vectorisation

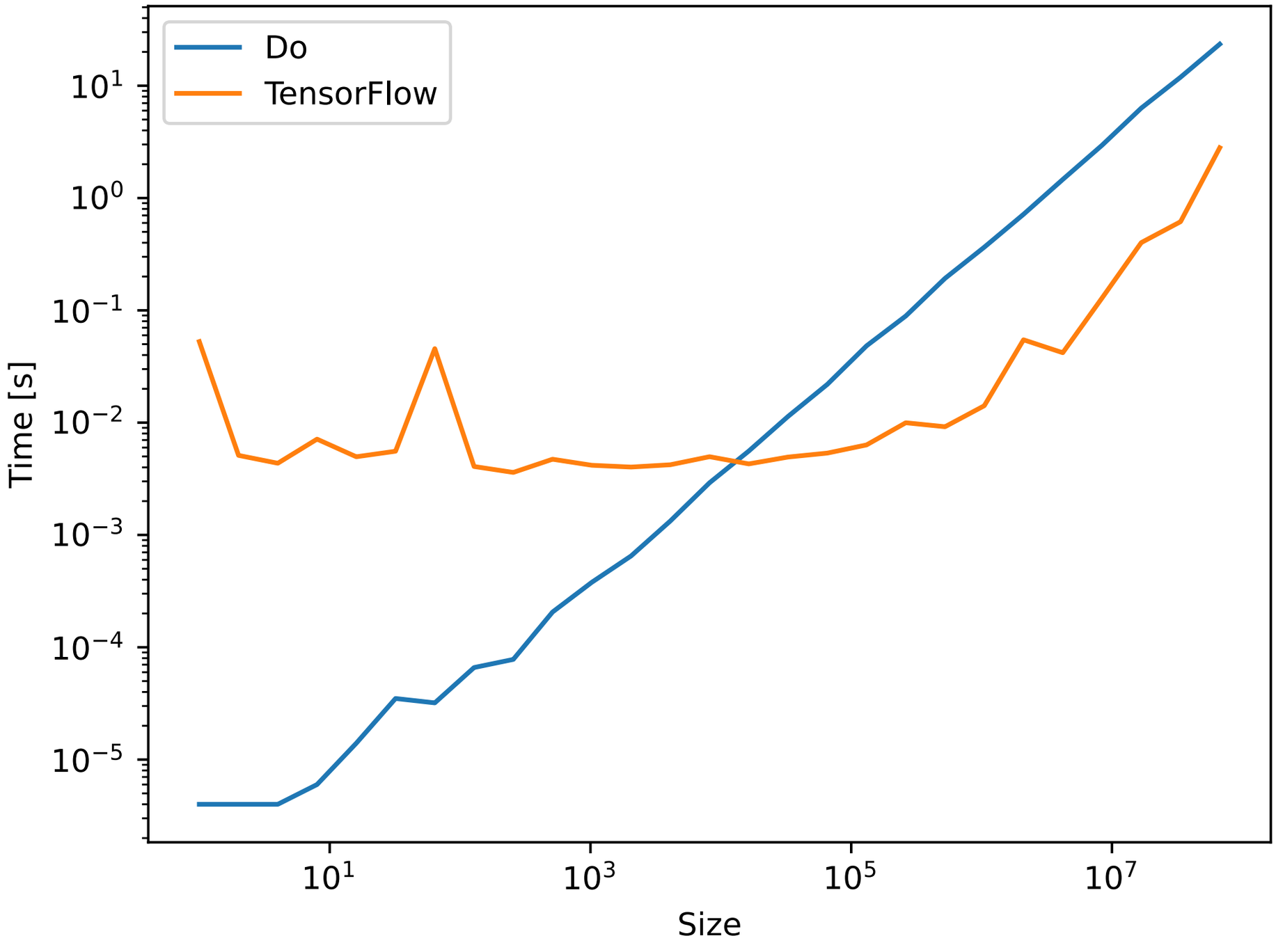
Code vectorisation

- Given a list of random numbers x, compute a*x+b
- With a for loop, it could be written as

```
res = []
for i in range(len(x)):
      res.append(a*x+b)
```

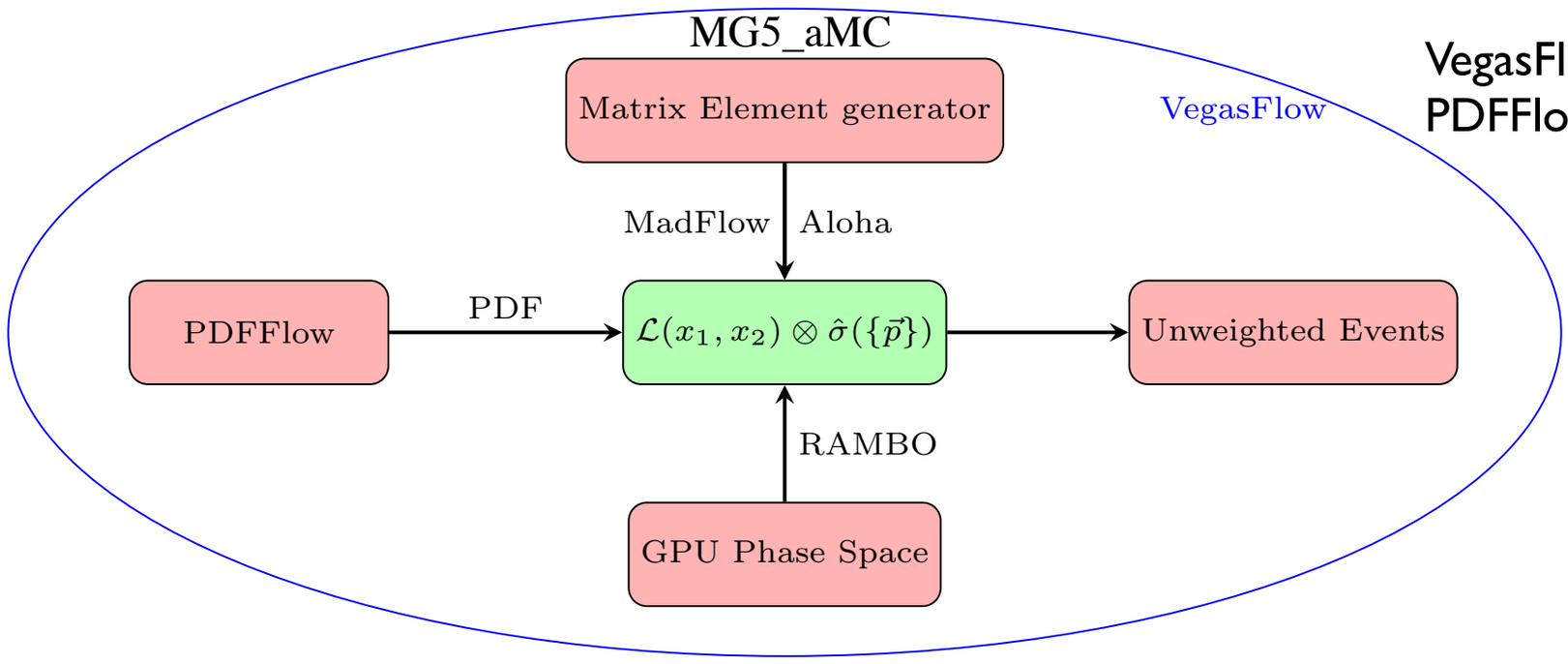
- Where all operations are executed sequentially, on a single core
- Treat the operation as involving vectors
  - Dedicated modules exist to speed up the computation (TensorFlow has many of them)
  - Each component is independent, and the computation is spread on available resources (CPU/GPU/…) in parallel
  - This works also for complicated functions, functions of functions, …

a*x+b: speed comparison



MadFlow

Carrazza, Cruz-Martinez, Rossi, Zaro 2105.10529 & 2106.10279
https://github.com/N3PDF/madflow



VegasFlow: 2002.12921
PDFFlow: 2009.06635

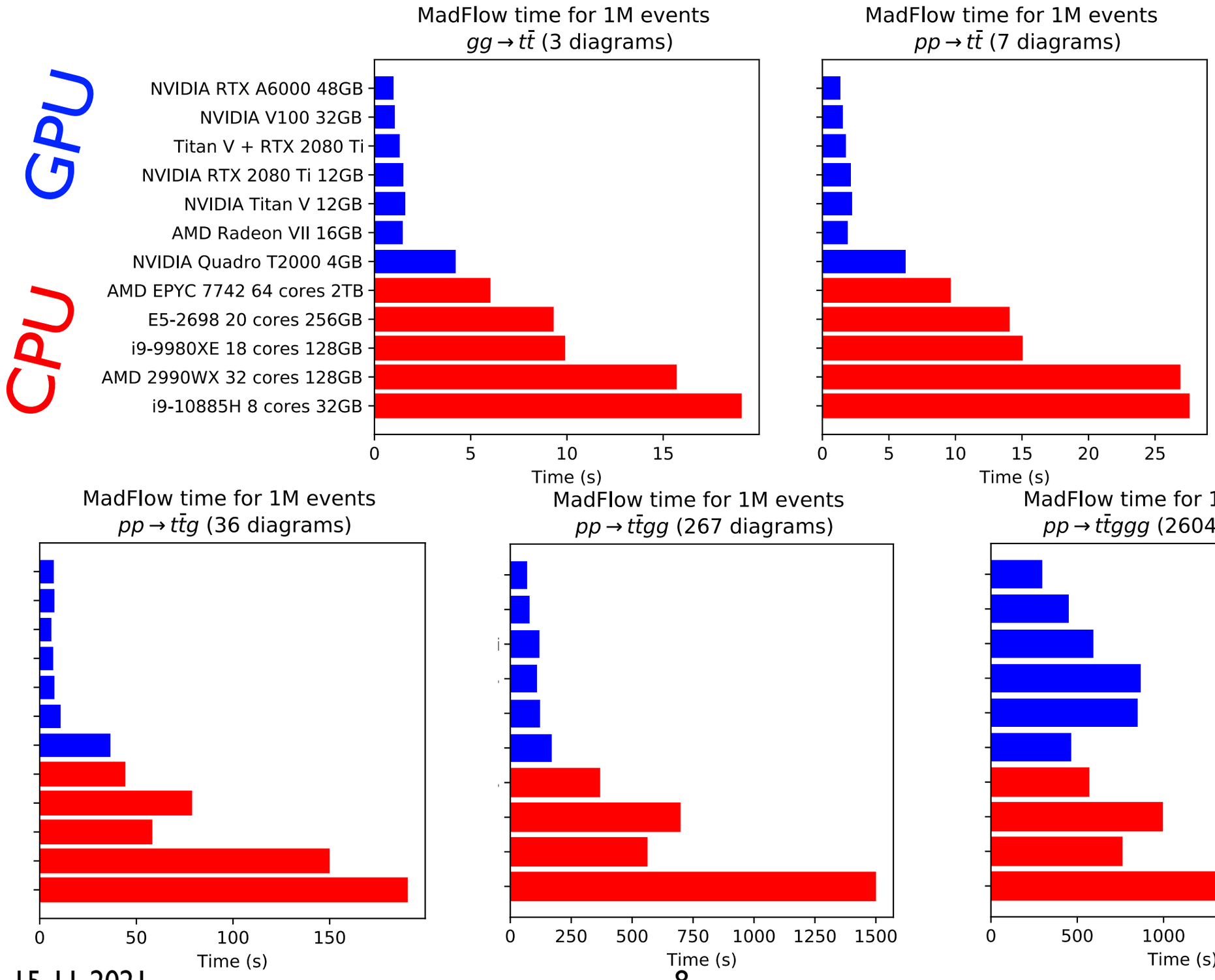
- Proof-of-principle of a modular and vectorised event generator
- Relies on VegasFlow for adaptive integration and PDFFlow for PDF evaluation
- Uses MG5_aMC as matrix-element provider
- MEs and ALOHA routines are exported in a dedicated, Python3/TensorFlow format

Usage

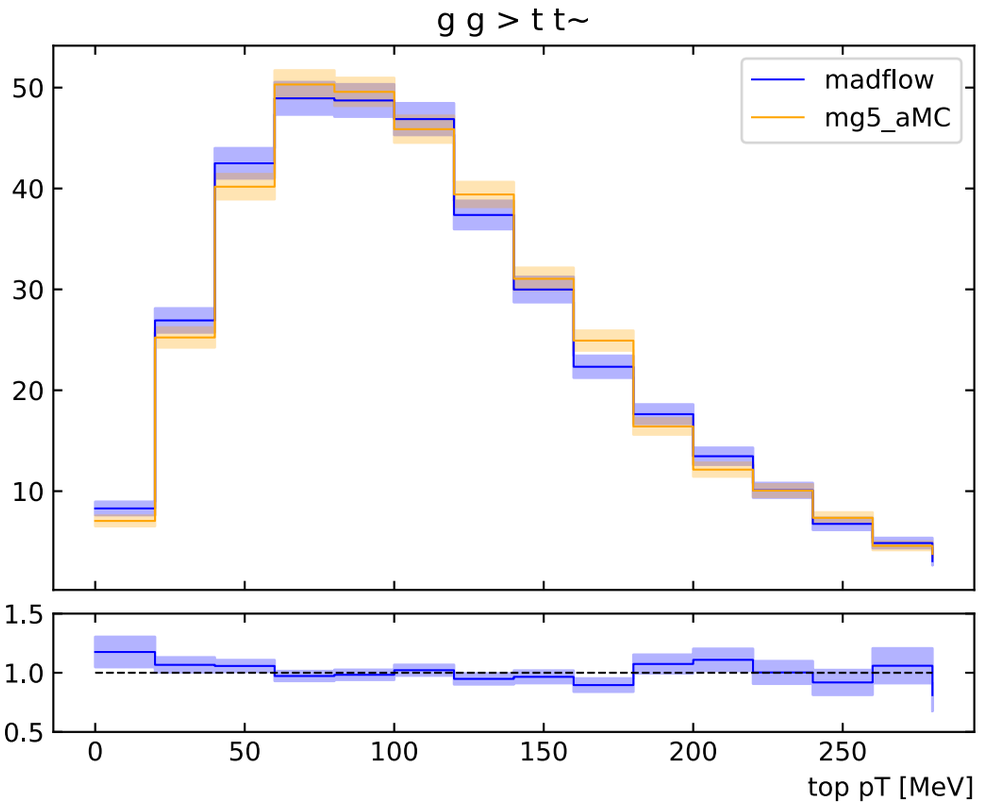
- Install from git repo (see readme.md)
- Run it

```
madflow --madgraph_process "p p > t t~"
[INFO] (<madflow>) Matrix files written to: /var/folders/p0/j7rq7fm5431bjdm1xxnlfvzw0000gn/T/mad_yuqm0lb1
[INFO] (pdfflow.pflow) Loading member 0 from NNPDF31_nnlo_as_0118
[INFO] (<madflow>) Set variable muF=muR=sum(mT)/2
[INFO] (<madflow>) Testing 1_uux_ttx: [0.73673989 0.78713886 0.63083346 0.58838473 0.82257544 0.65244094
 0.50578657 0.50560999 0.51154182 0.53350421]
[INFO] (<madflow>) Testing 1_gg_ttx: [1.61980536 2.34334231 0.78694774 0.60951565 3.19073427 0.92012759
 0.35567051 0.35742947 0.37301178 0.42844266]
[INFO] (<madflow>) Running 5 warm-up iterations of 1000000 events each
[INFO] Result for iteration 0: 450.5184 +/- 14.2488(took 53.16994 s)
[INFO] Result for iteration 1: 441.4085 +/- 2.2638(took 29.47527 s)
[INFO] Result for iteration 2: 441.3176 +/- 0.4585(took 33.38355 s)
[INFO] Result for iteration 3: 440.4455 +/- 0.2723(took 29.24159 s)
[INFO] Result for iteration 4: 440.8846 +/- 0.2667(took 31.80983 s)
[INFO]  > Final results: 440.77 +/- 0.17541
[INFO] (<madflow>) Running 5 iterations of 1000000 events each with the grid frozen
[INFO] Result for iteration 0: 441.1151 +/- 0.2830(took 30.61854 s)
[INFO] Result for iteration 1: 440.3910 +/- 0.2802(took 25.42976 s)
[INFO] Result for iteration 2: 440.8030 +/- 0.2681(took 32.79075 s)
[INFO] Result for iteration 3: 441.1218 +/- 0.2590(took 29.40579 s)
[INFO] Result for iteration 4: 441.0271 +/- 0.2567(took 28.18434 s)
[INFO]  > Final results: 440.901 +/- 0.120205
[INFO] (<madflow>) > Madflow took: 333.8s
```

Performances



Comparison with MG5_aMC and comments



Process	`MadFlow` CPU	`MadFlow` GPU	MG5_aMC
$gg \to t\bar{t}$	9.86 μs	1.56 μs	20.21 μs
$pp \to t\bar{t}$	14.99 μs	2.20 μs	45.74 μs
$pp \to t\bar{t}g$	57.84 μs	7.54 μs	93.23 μs
$pp \to t\bar{t}gg$	559.67 μs	121.05 μs	793.92 μs

time per PS point

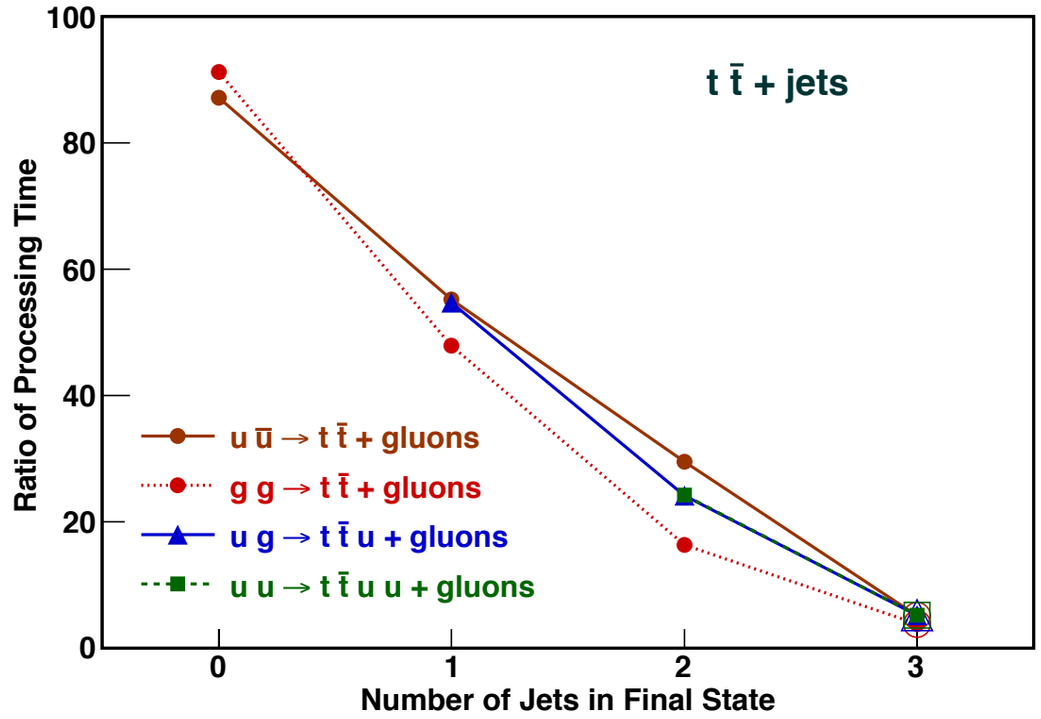
- Agreement found within statistical fluctuations
- On CPU, MadFlow is ~2x faster than MG5_aMC
- On GPU, MadFlow performances improve by a factor ~10
- For very complicated processes, memory is a limiting factor

Limitations

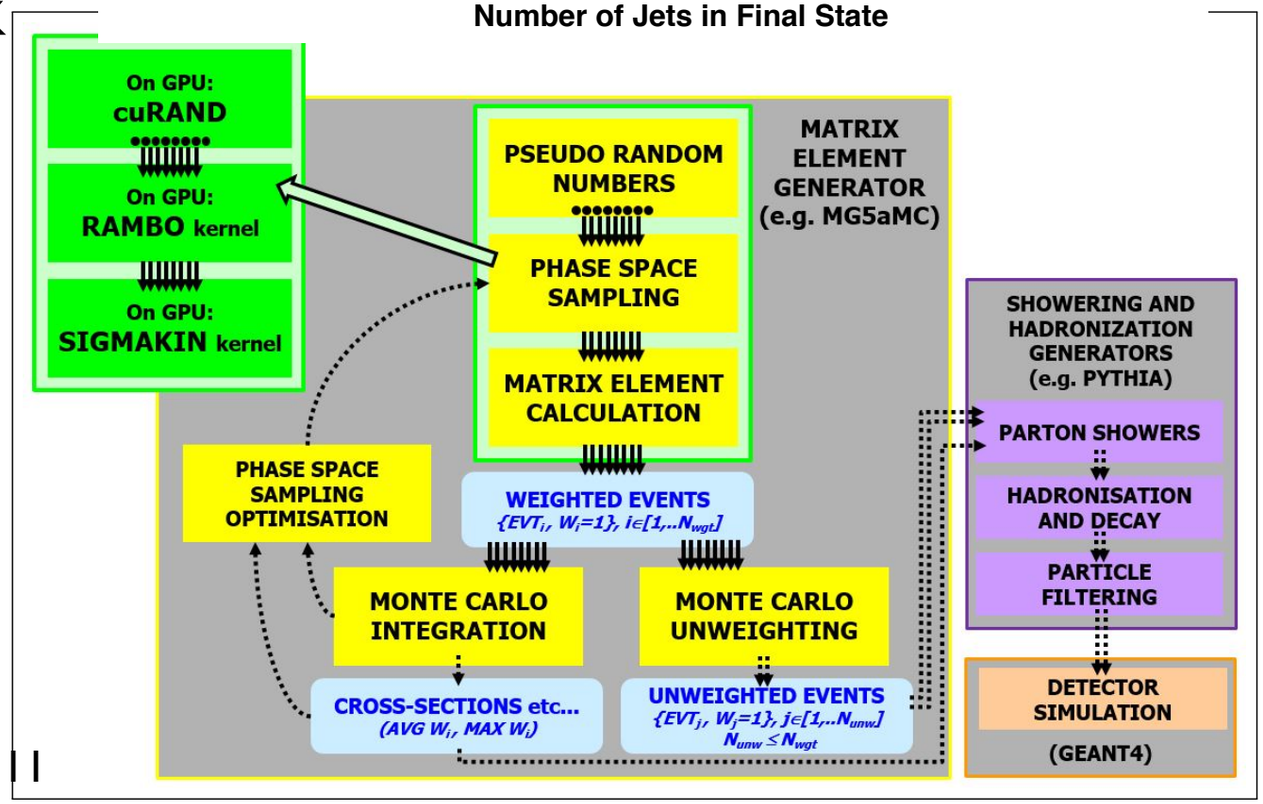
- PS generation relies on RAMBO: highly inefficient, in particular for complex processes
- Cuts module to be implemented
- Events lack color-flow information: cannot be showered
- LO only, of course ;-)

Other approaches

- MG2CUDA: pioneering work by K. Hagiwara et al (1305.0708), using CUDA



- MadGraph5 GPU: systematic work in progress by Olivier et al (2106.12631), using CUDA (extensions in other languages foreseen)



Outlook

- Efforts to improve computing efficiency of codes in HEP are needed!
- Proof of concept for TF-based version of MG5_aMC
  - Promising results in terms of speed and process reach
  - Not tied to a specific HW configuration
- Many limitations still to be addressed
- Possible further speed/memory improvements from translating the matrix element from Python to a TF custom operator (WIP)