



NextGen
Next Generation Triggers

Data parallelism in MadGraph: hardware acceleration with GPUs and SIMD CPUs

Daniele Massaro

CERN

*On behalf of the MG5AMC
CUDA CPP Development Team*

Durham HPC days

6th June 2025



This talk would have not been possible without the help and the material I borrowed from my team-mates: Olivier Mattelaer, Stefan Roiser, Andrea Valassi, Zenny Wettersten.

The teams

CUDACPP plugin core development (NVIDIA and AMD GPUs, vectorised C++ on CPUs)

Olivier Mattelaer
(UCLouvain)

Daniele Massaro
Stefan Roiser
Zenny Wettersten
Jorgen Teig (2023)
(CERN IT-FTI-PSE)

Andrea Valassi
(CERN EP-LBD)

Stefan Hageboeck
(CERN EP-SFT)

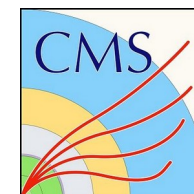


CMS integration tests

Jin Choi
(Seoul National University)

Saptaparna Bhattacharya
(Wayne State University)

Robert Schoefbeck
(HEPHY Vienna)



SYCL plugin (also Intel GPUs)
WIP: integration into CUDACPP

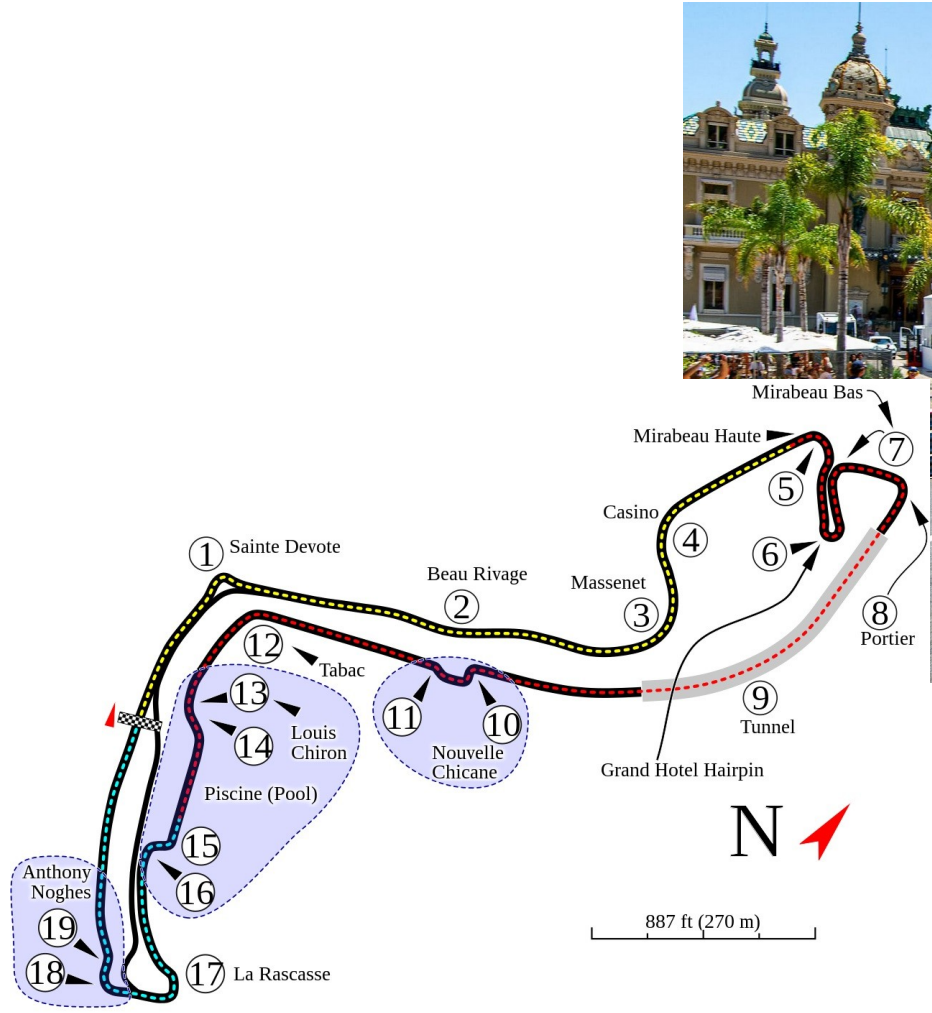
Taylor Childers
Nathan Nichols

(ANL)



What is Monte Carlo?

What is Monte Carlo?



© Scuderia Ferrari Press Office (2023)

© Formula 1

What is Monte Carlo?

Monte Carlo Casino



© Scuderia Ferrari Press Office (2023)

[A] Monte Carlo Simulation is a mathematical technique that is used to estimate the possible outcomes of an uncertain event.

[<https://www.ibm.com/topics/monte-carlo-simulation>]

What is Monte Carlo?

- In pragmatic terms, and in the realm of High-Energy-Physics (HEP):

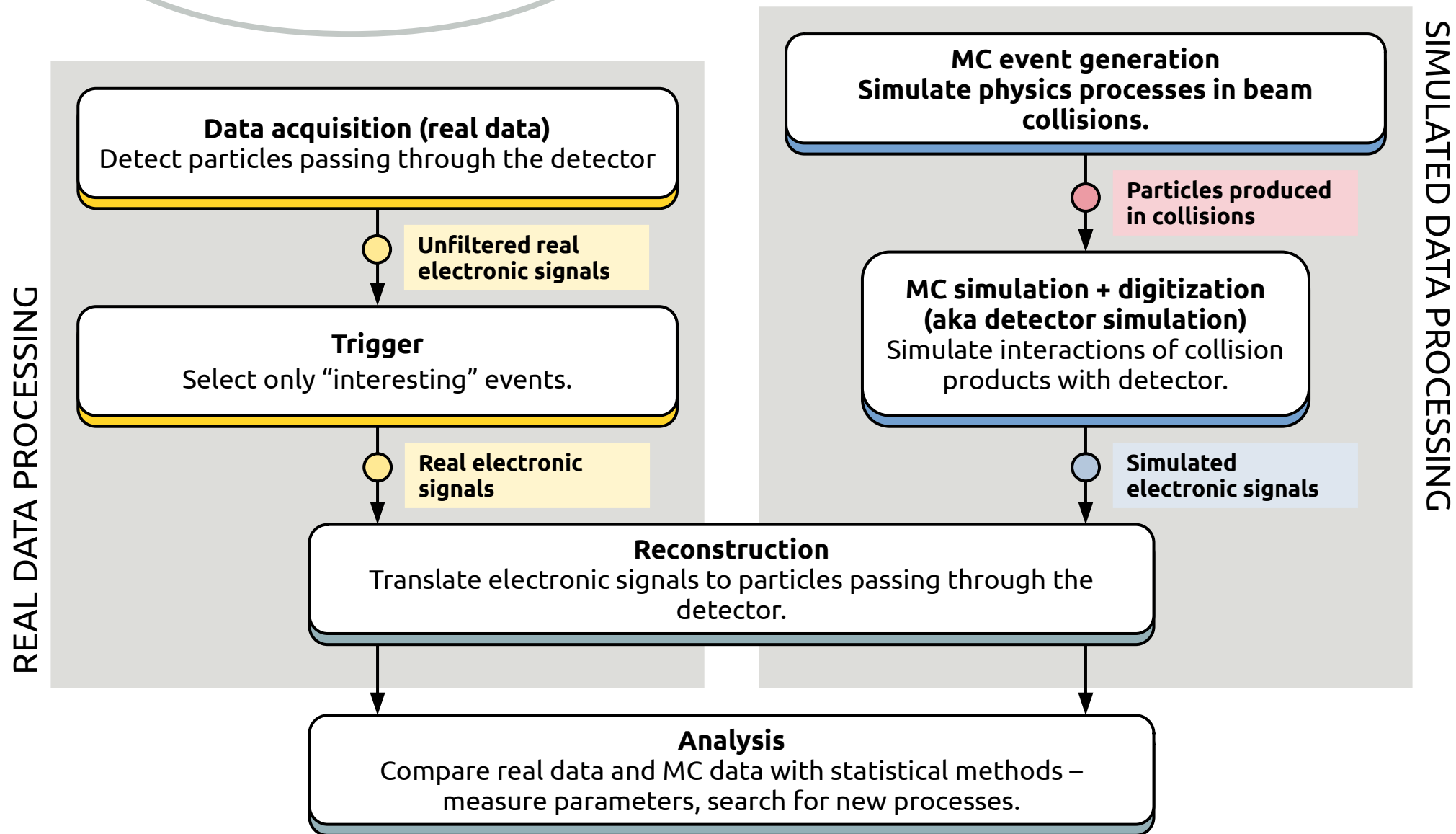
a Monte Carlo simulation is the process from which an event generator goes from (pseudo-)random numbers to physical events, that can be further processed and used for analysis purposes.

« God does ~~not~~ play dice » (Einstein)



Software libraries that generate simulated high-energy particle physics events.

Event generators

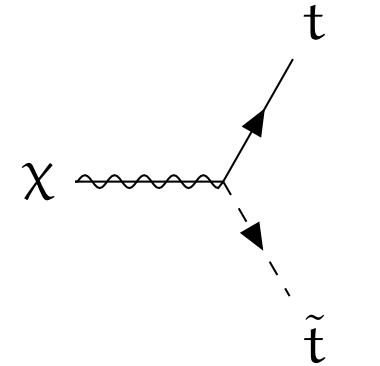


In summary

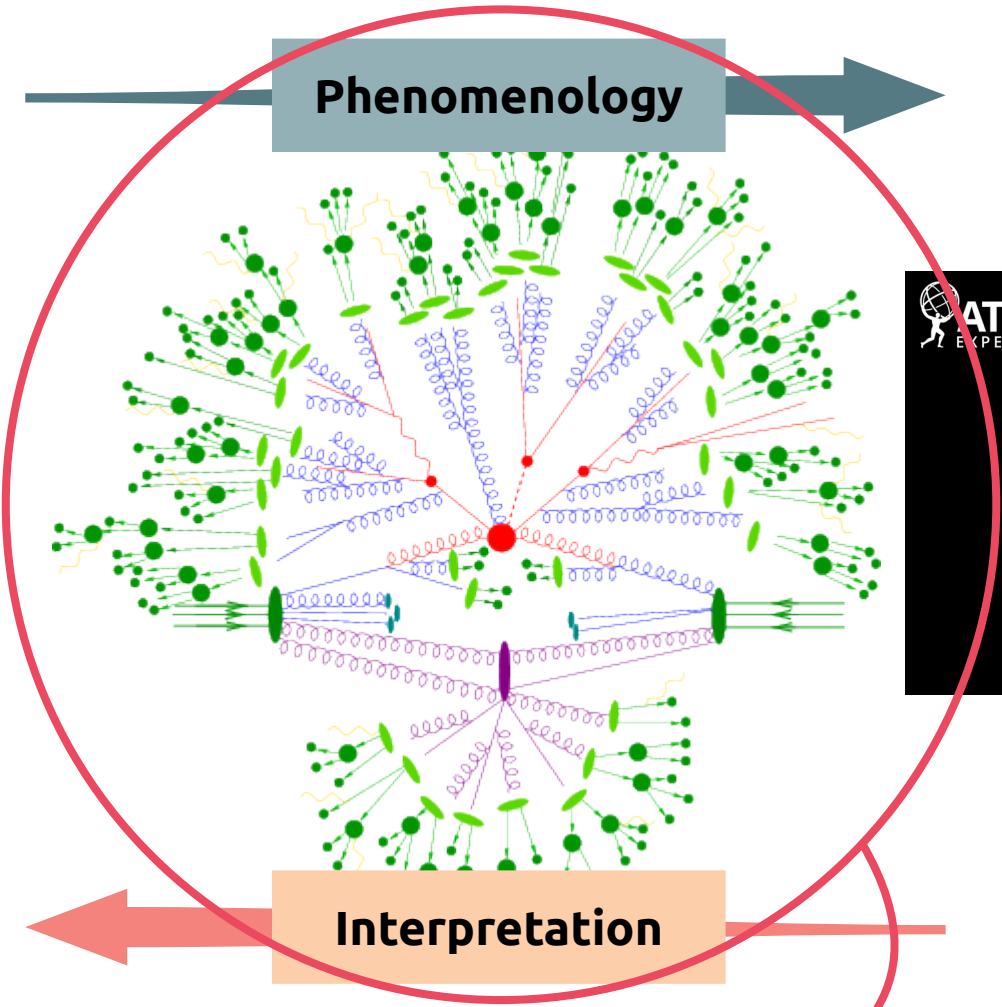
Theory

Phenomenology

Experiment



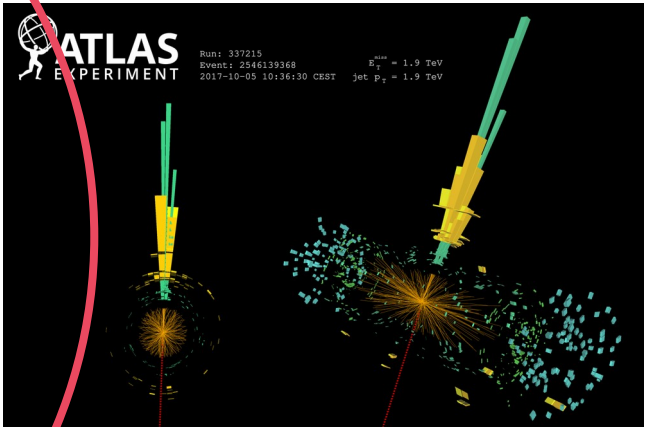
Fundamental parameters, field content, symmetries.



Interpretation

Monte Carlo event generators

© ATLAS collaboration, CERN



Observables, triggers, acceptance, background, calibration.

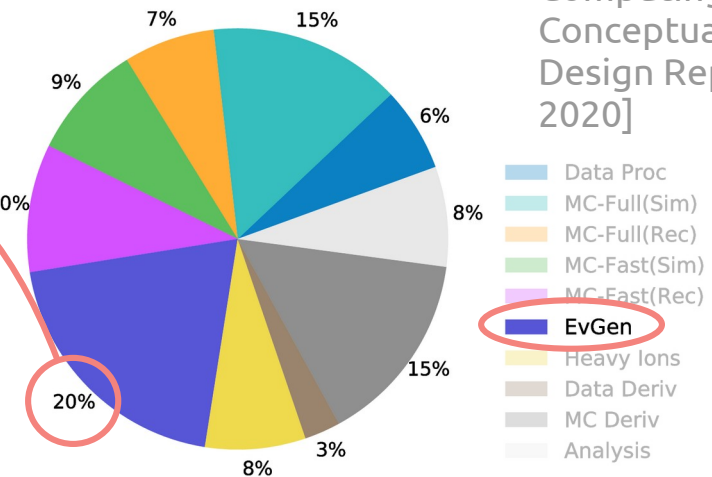
[P. Skands]
[SHERPA]

Looking ahead

- Monte-Carlo event generation for HL-LHC is estimated to be $\approx 20\%$ of the total computing resources.

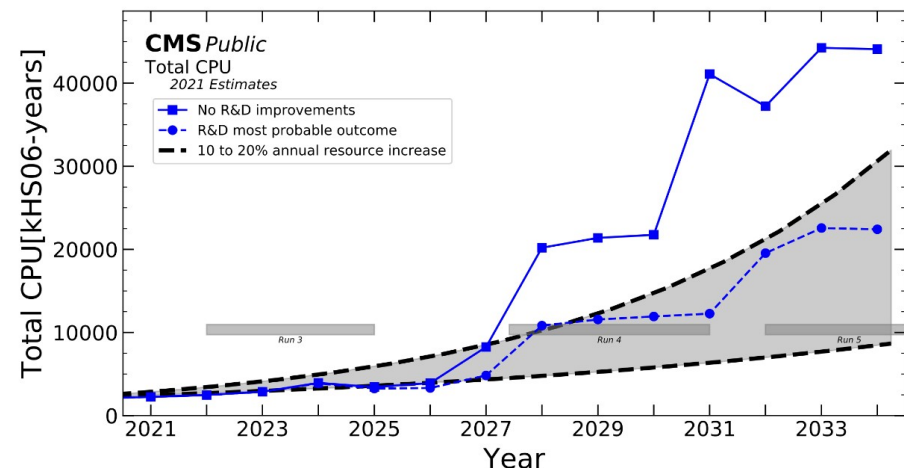
ATLAS Preliminary

2020 Computing Model -CPU: 2030: Baseline



[ATLAS HL-LHC Computing Conceptual Design Report, 2020]

- Predicted CPU resources are not enough to cover experiments' needs.



[CMS Phase-2 Computing Model: Update Document, 2022]

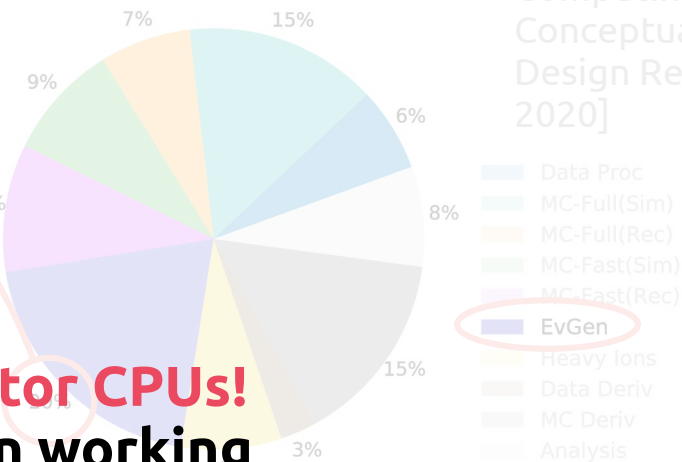
Daniele Massaro

Looking ahead

- Monte-Carlo event generation for HL-LHC is estimated to be $\approx 20\%$ of the total computing resources.

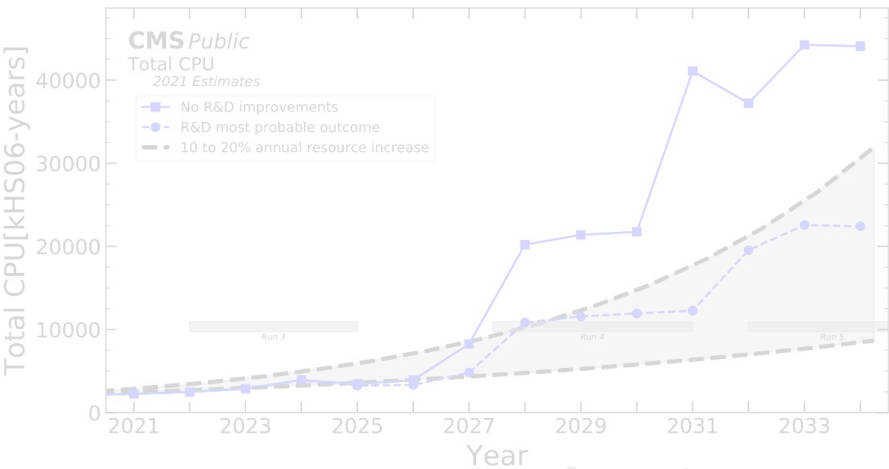
Solution: use GPUs/vector CPUs!
CERN IT et al. have been working for few years on this topic.

ATLAS Preliminary
2020 Computing Model -CPU: 2030: Baseline



[ATLAS HL-LHC Computing Conceptual Design Report, 2020]

- Predicted CPU resources are not enough to cover experiments' needs.



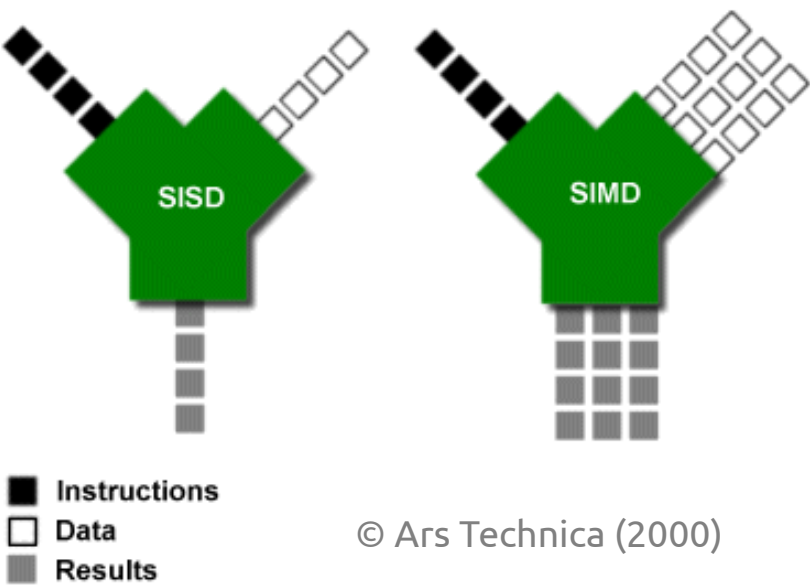
[CMS Phase-2 Computing Model: Update Document, 2022]

Daniele Massaro

Data parallelisation

Sequential processing

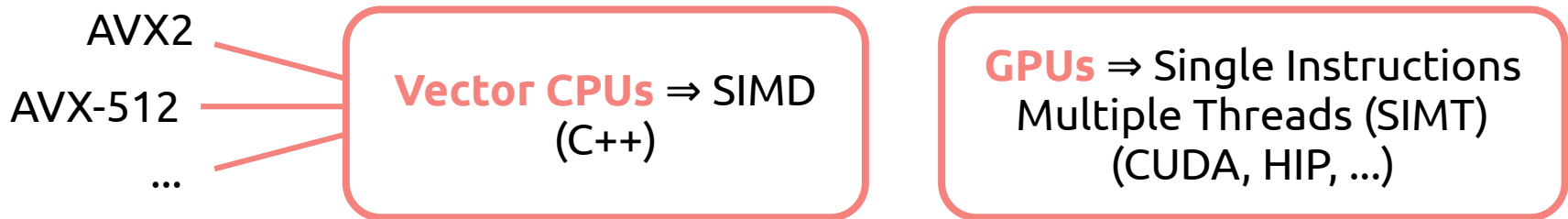
**Single Instruction
Single Data:**
1 input and 1
output per cycle.



Data-parallel processing
(lockstep processing)

E.g.: Single Instruction Multiple Data (SIMD):
N inputs and N
outputs per cycle

© Ars Technica (2000)



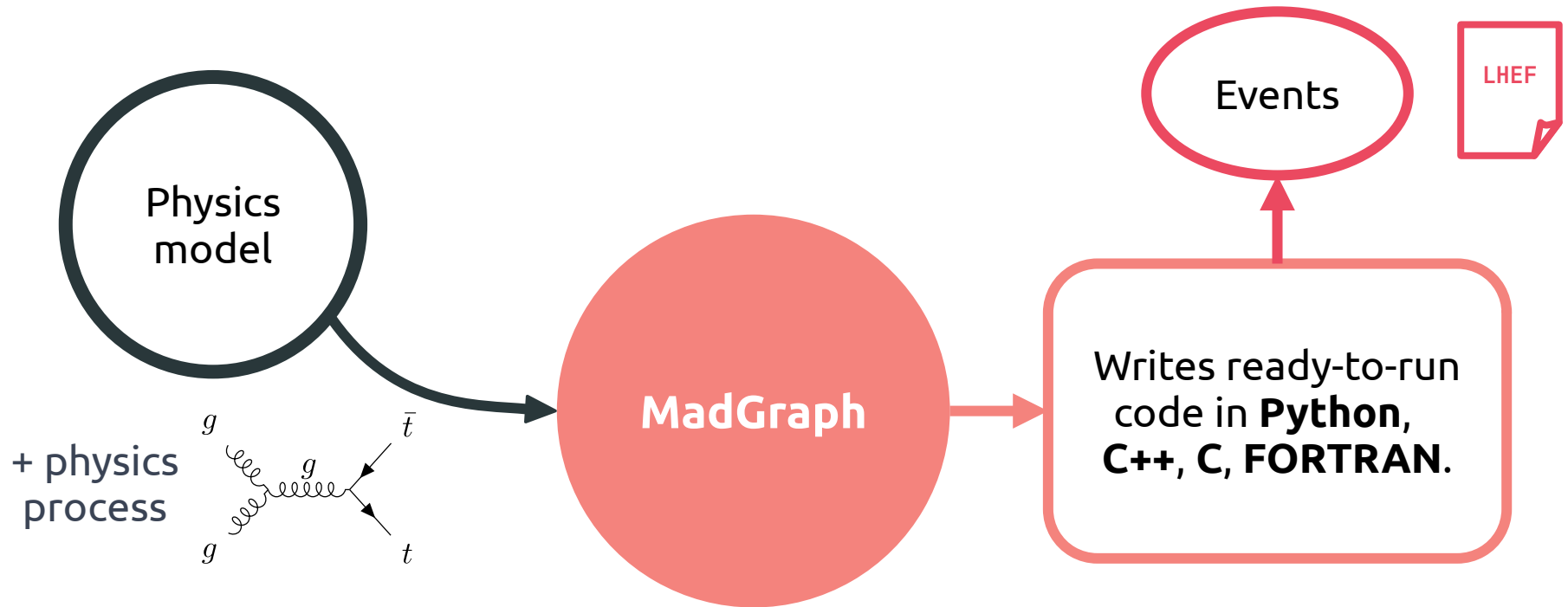
2 main strategies to go from sequential to data-parallel,
but **that's not enough!**

CPU vectorisation and GPUs

- Vector CPUs and GPUs are now widely available to HEP:
 - Most CPUs support at least AVX2;
 - GPUs are becoming more and more available (see HPC centres).
- **However**, they are difficult to exploit in HEP software:
 - e.g. MC detector simulations rely on multiple stochastic branching (makes lockstep processing difficult).
- **But**, **matrix element event generators** are ideal software workflows for SIMD and GPUs
 - MC sampling of many data point.

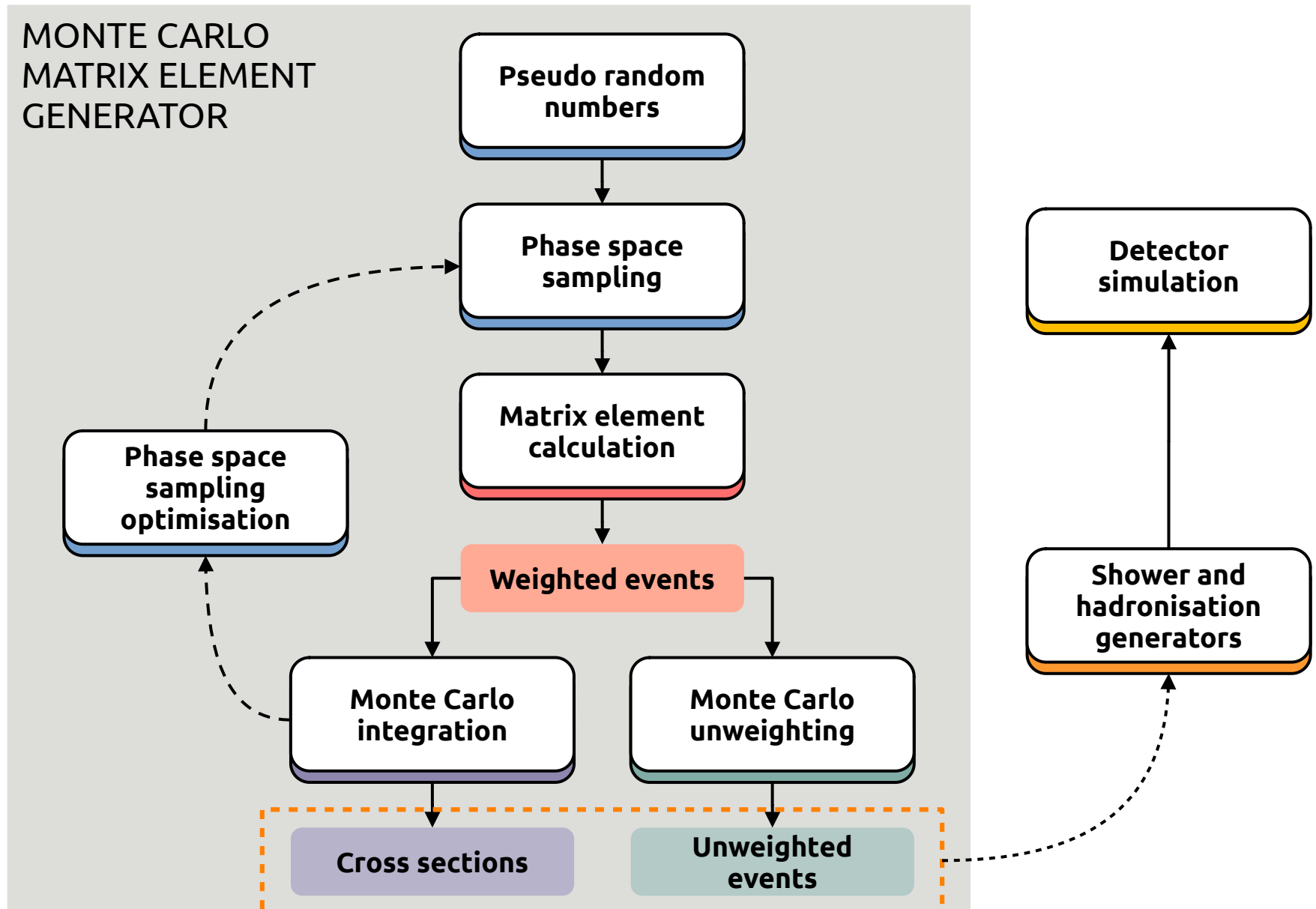


MadGraph



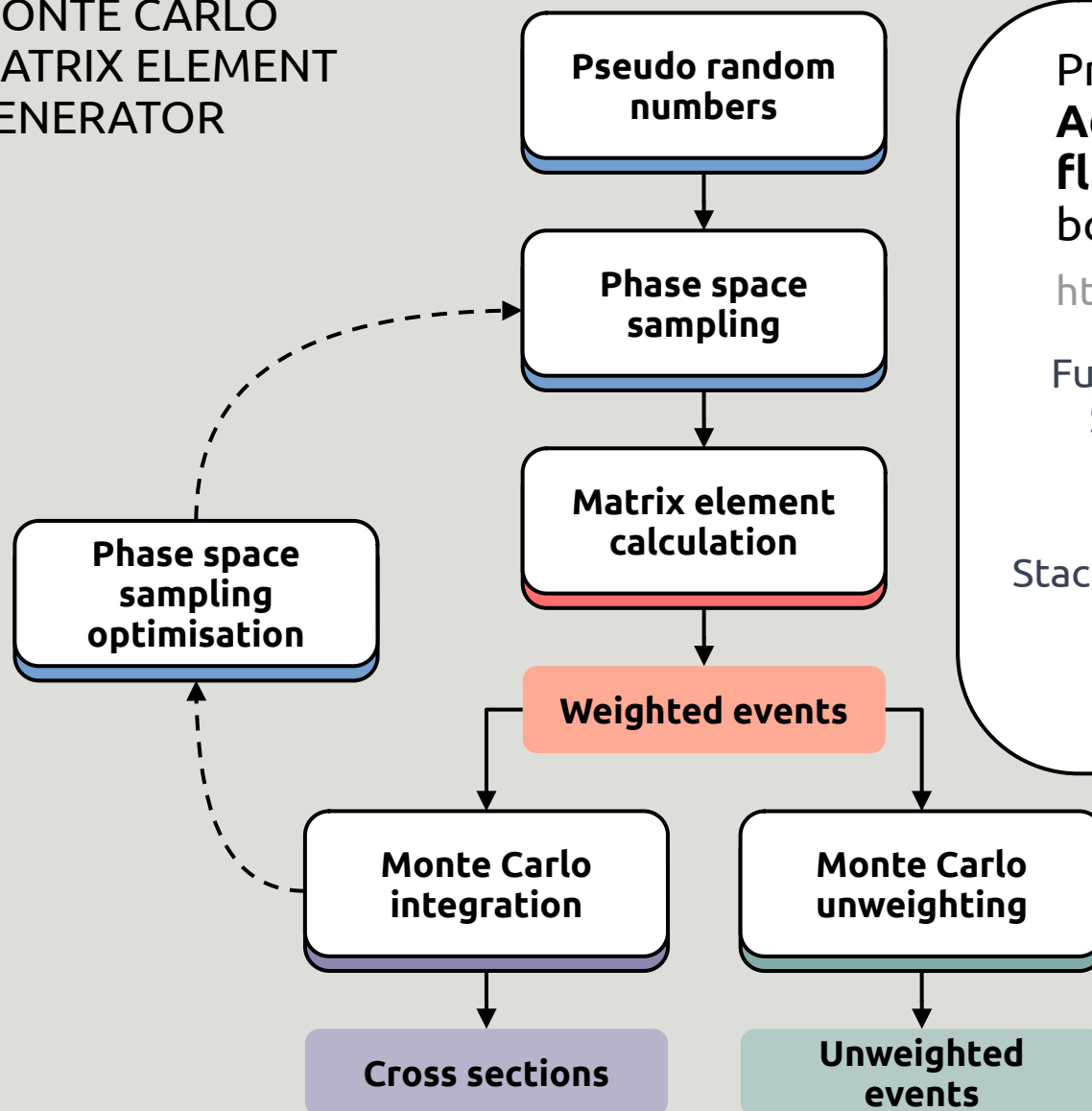
But what exactly does this code do?

Anatomy of an event generator



Anatomy of an event generator

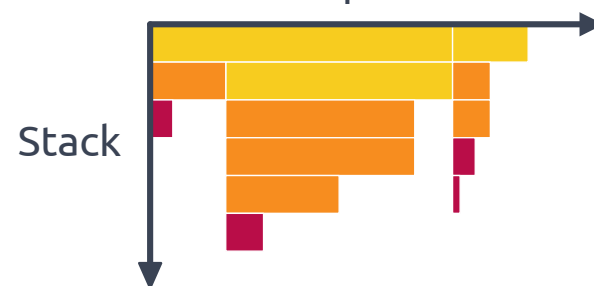
MONTE CARLO
MATRIX ELEMENT
GENERATOR



Profile the code using **Adaptyst** and produce **flamegraphs** to find bottlenecks.

<https://adaptyst.web.cern.ch/>

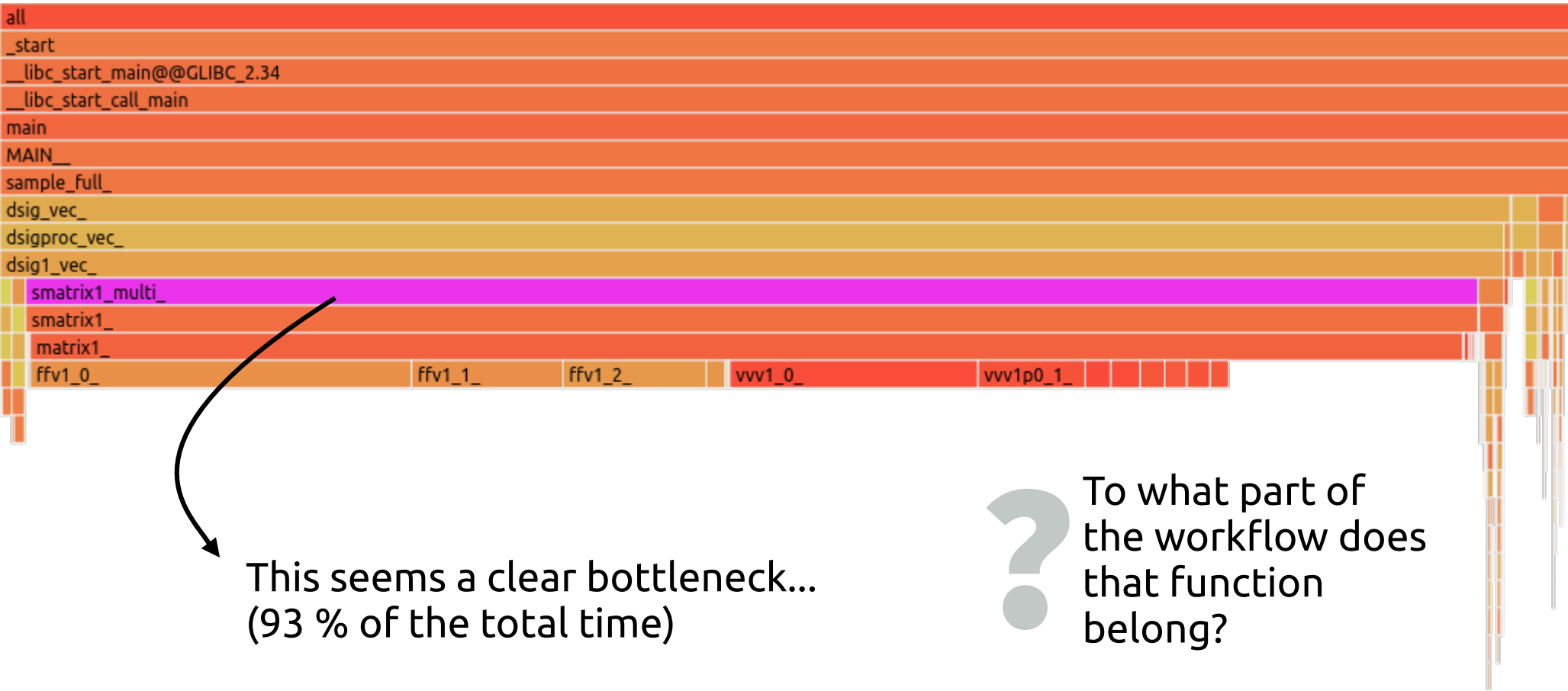
Functions in alphabetical order
Size \propto time spent wrt total



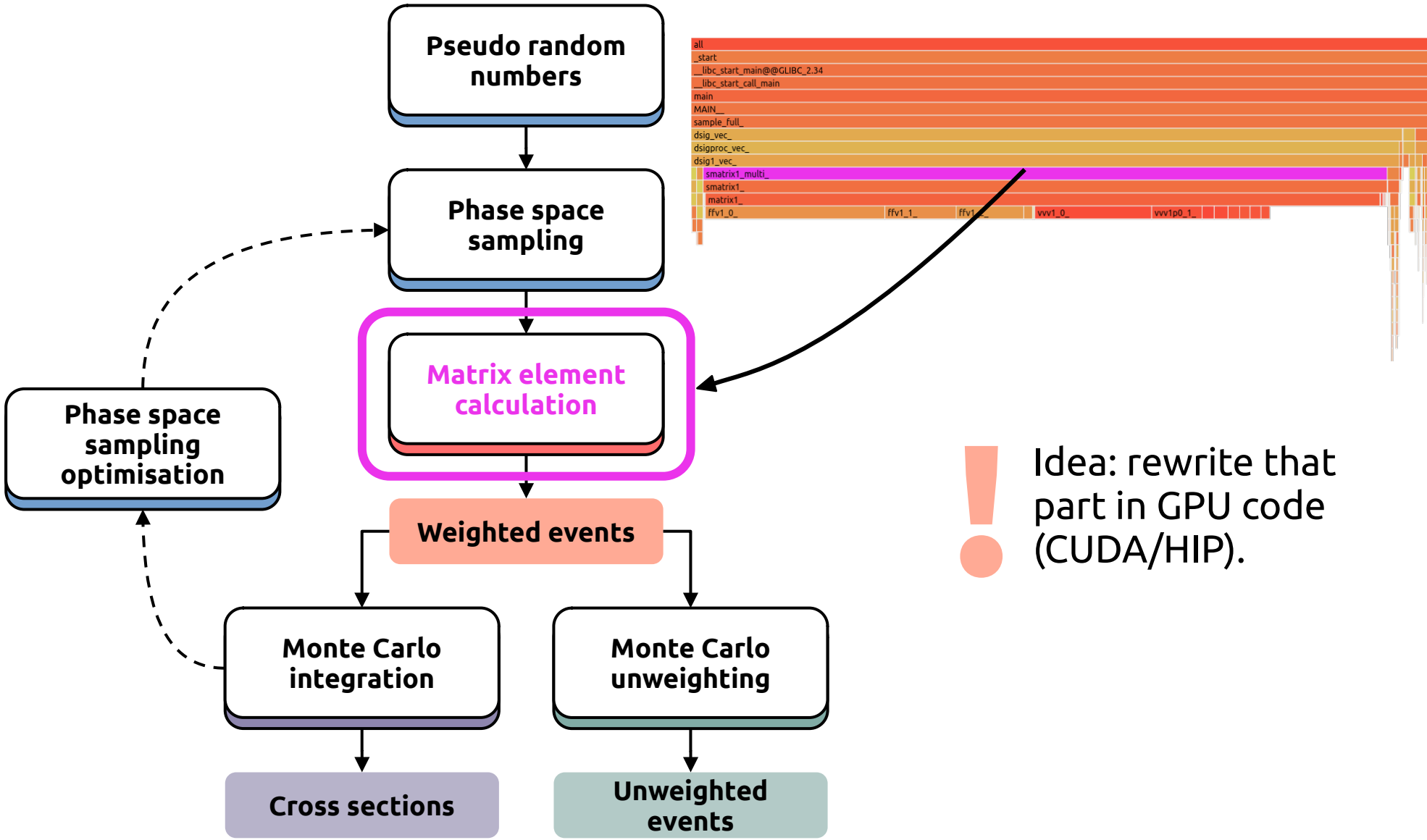
Profiling MadGraph

Details: $g g \rightarrow t \bar{t} g g$ with 10^6 events.
All the code is FORTRAN single thread.

Total execution time: **134.83 s**



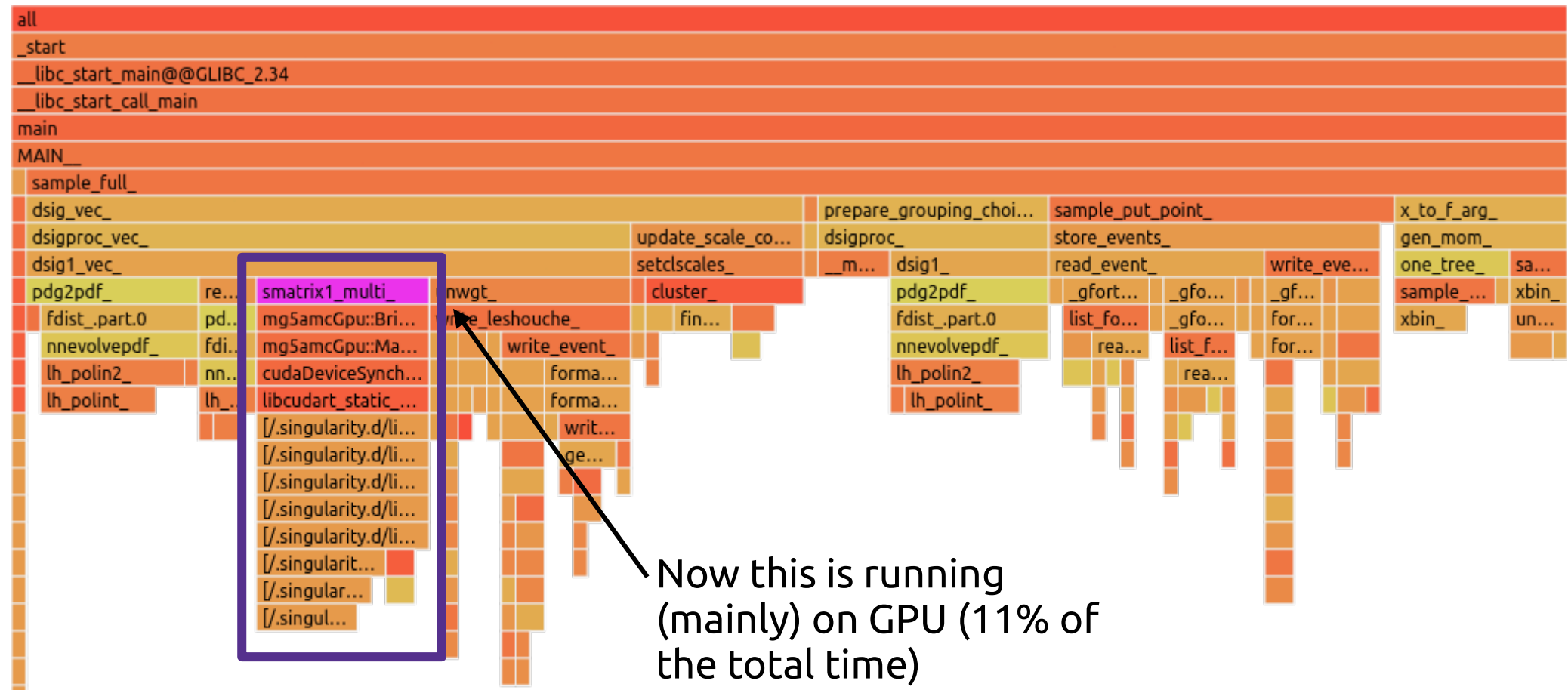
Profiling MadGraph



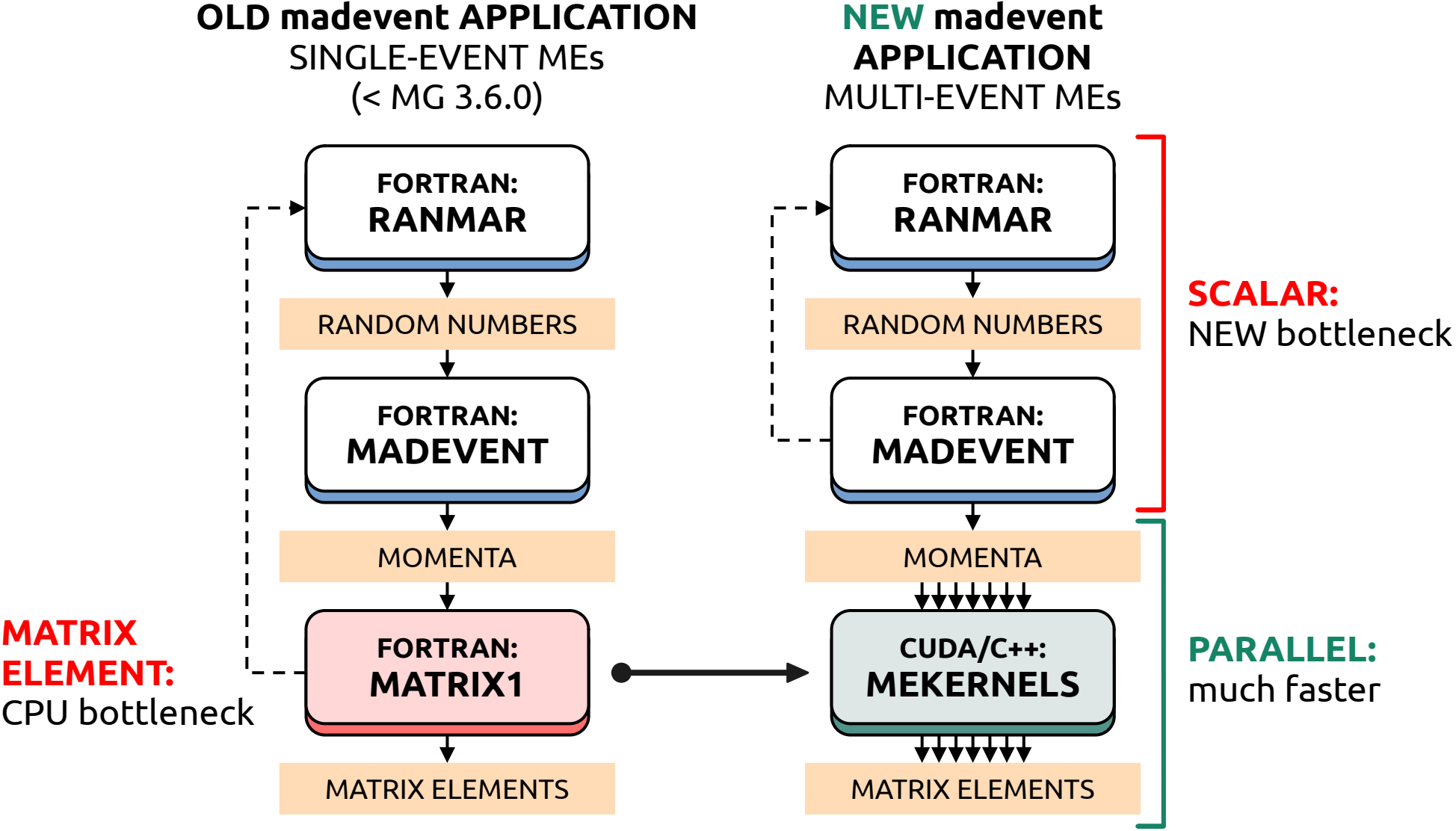
Profiling MadGraph

Details: $g g \rightarrow t \bar{t} g g$ with 10^6 events.
Matrix element computation has been offloaded to GPU (in this case NVIDIA V100).

Total execution time: **10.94 s**



The new architecture



Design choices

- **Re-entrant functions:**

- With **well-defined inputs and outputs**.
- **Stateless** with respect to shared data (static, common blocks).

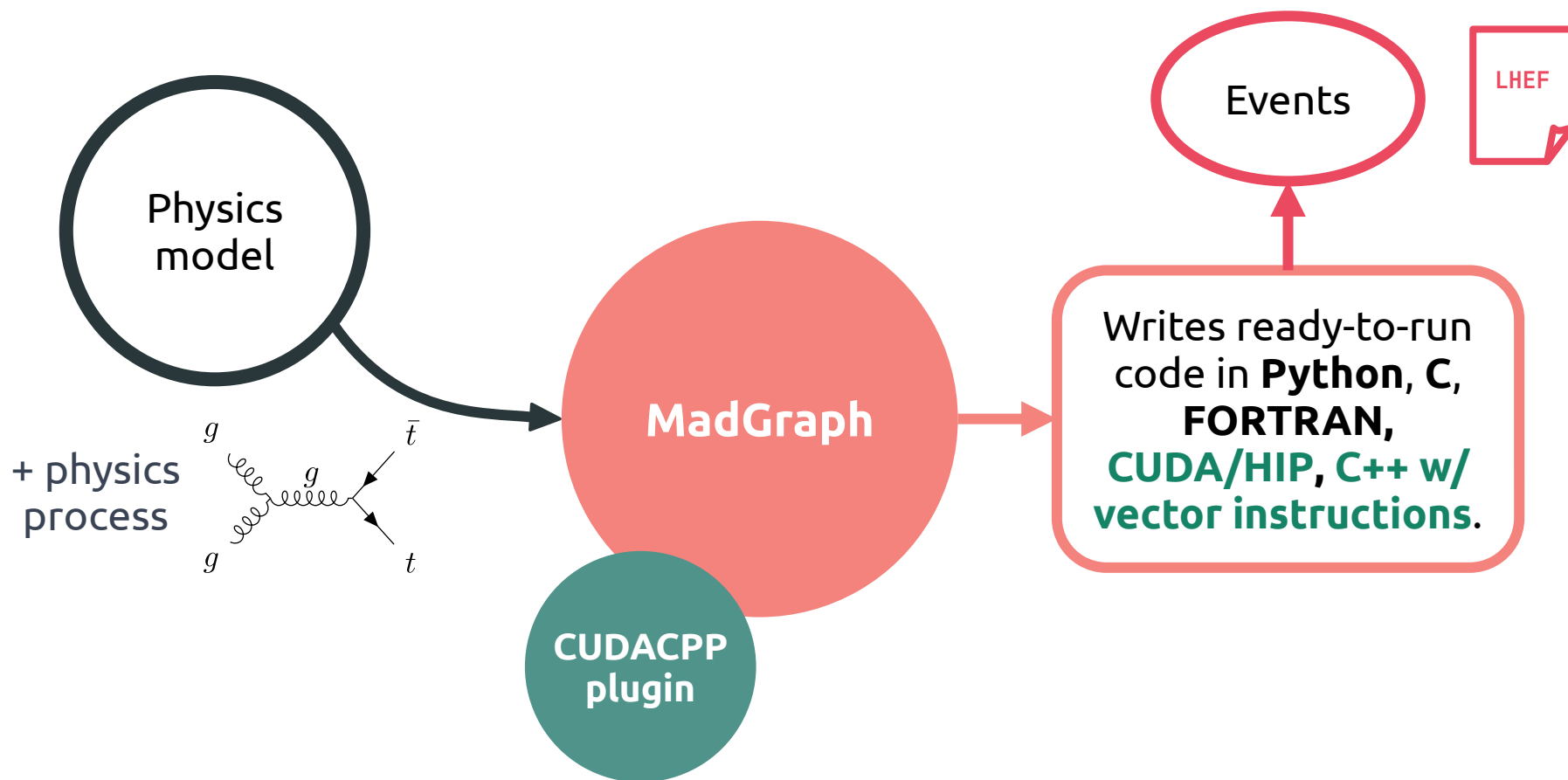


- **Data parallelism in mind from the start:**

- Move from single-event APIs to **multi-event APIs** (well-defined input arrays and well-defined output arrays).
- Structure-of-Arrays (**SoA**) memory layout is crucial for inputs and outputs.



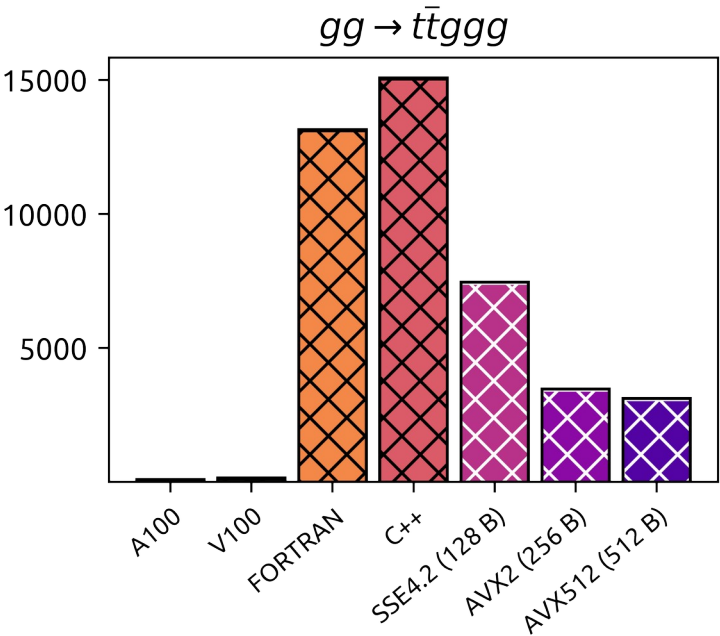
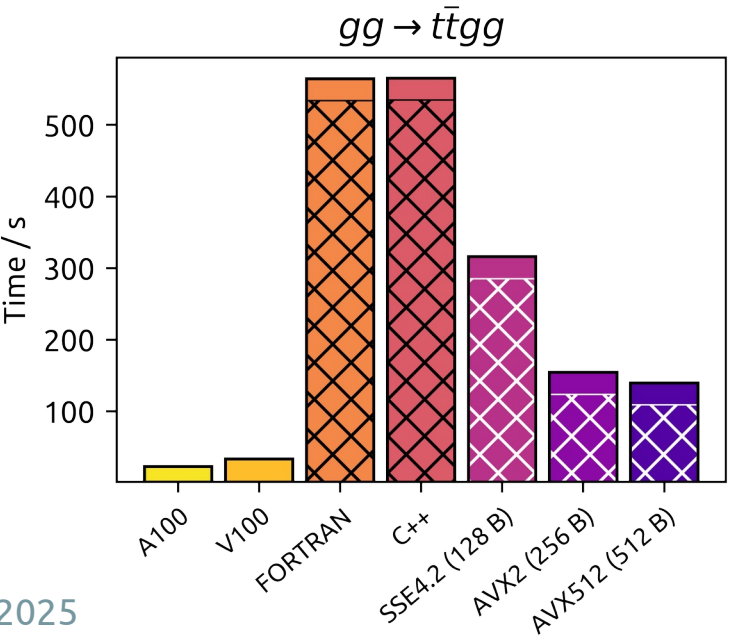
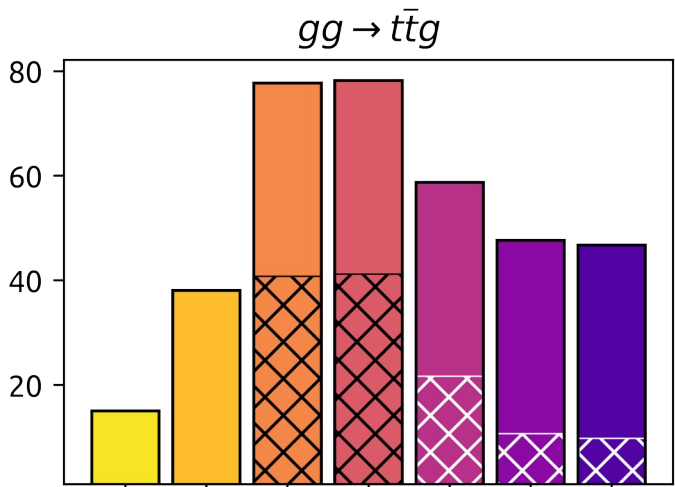
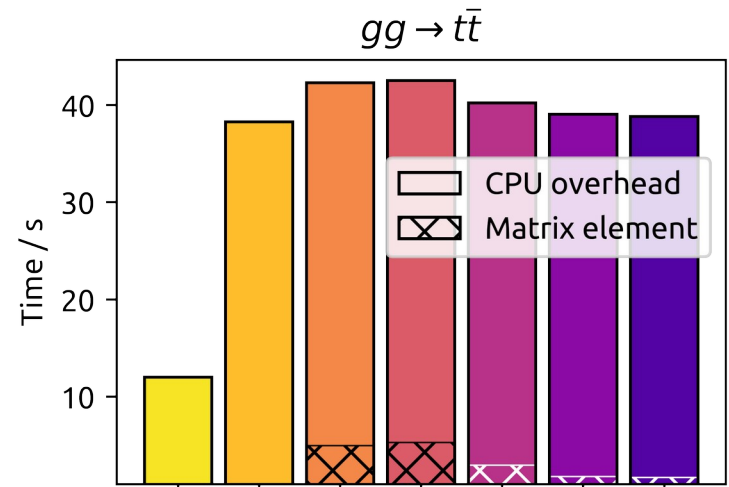
MadGraph w/ CUDACPP plugin



First release of the **CUDACPP** MadGraph plugin available:
github.com/madgraph5/madgraph4gpu (requires MG5_aMC@NLO v3.6.0).

Performance comparisons

Madevent single thread execution, 10⁶ events

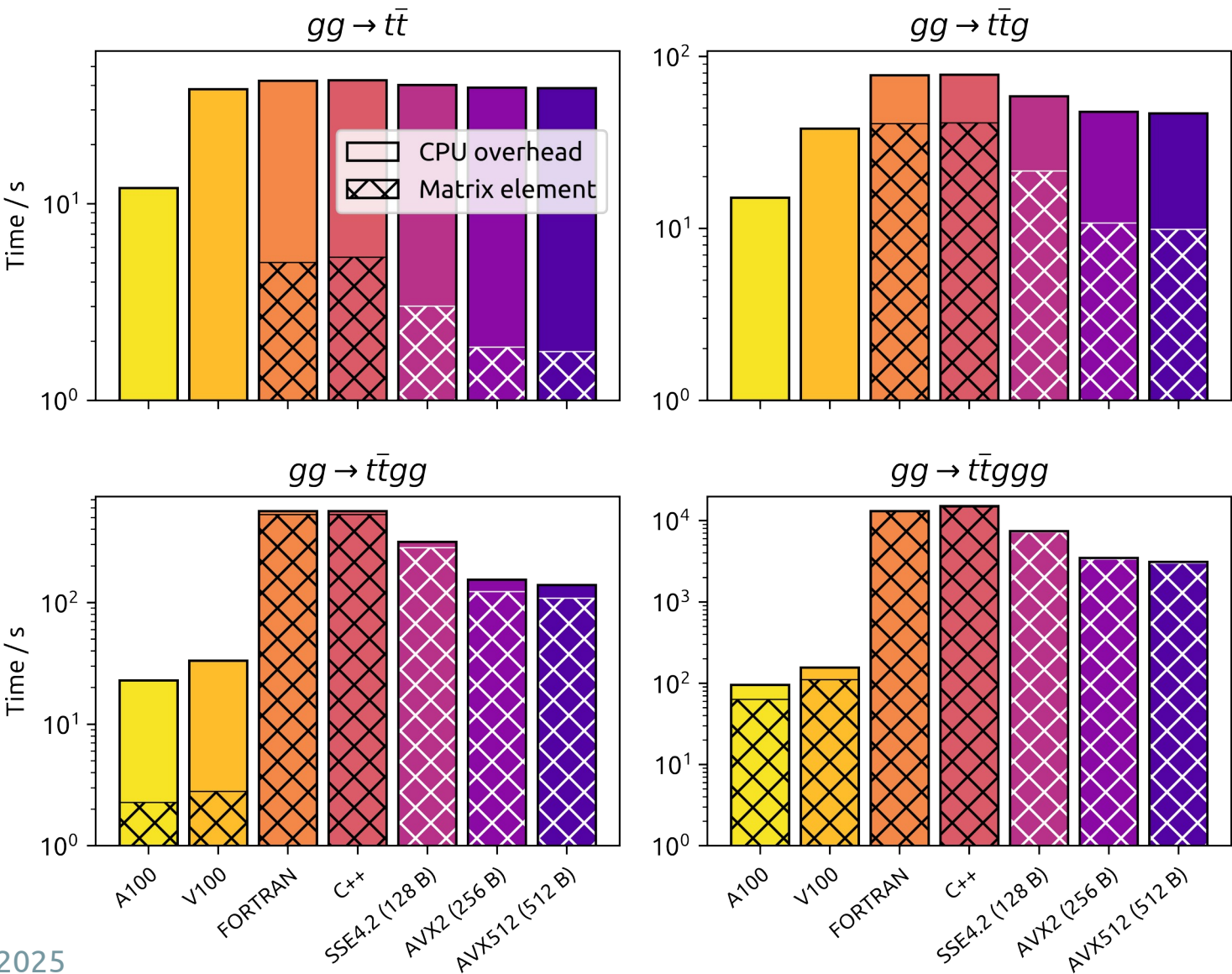


Tip for non-experts: the more the “g”s (or the “j”s) at the end of the process name, the more resource-intensive the job is.

- **A100:** AMD EPYC 7313
- **Others:** Intel(R) Xeon(R) Silver 4216

Performance comparisons

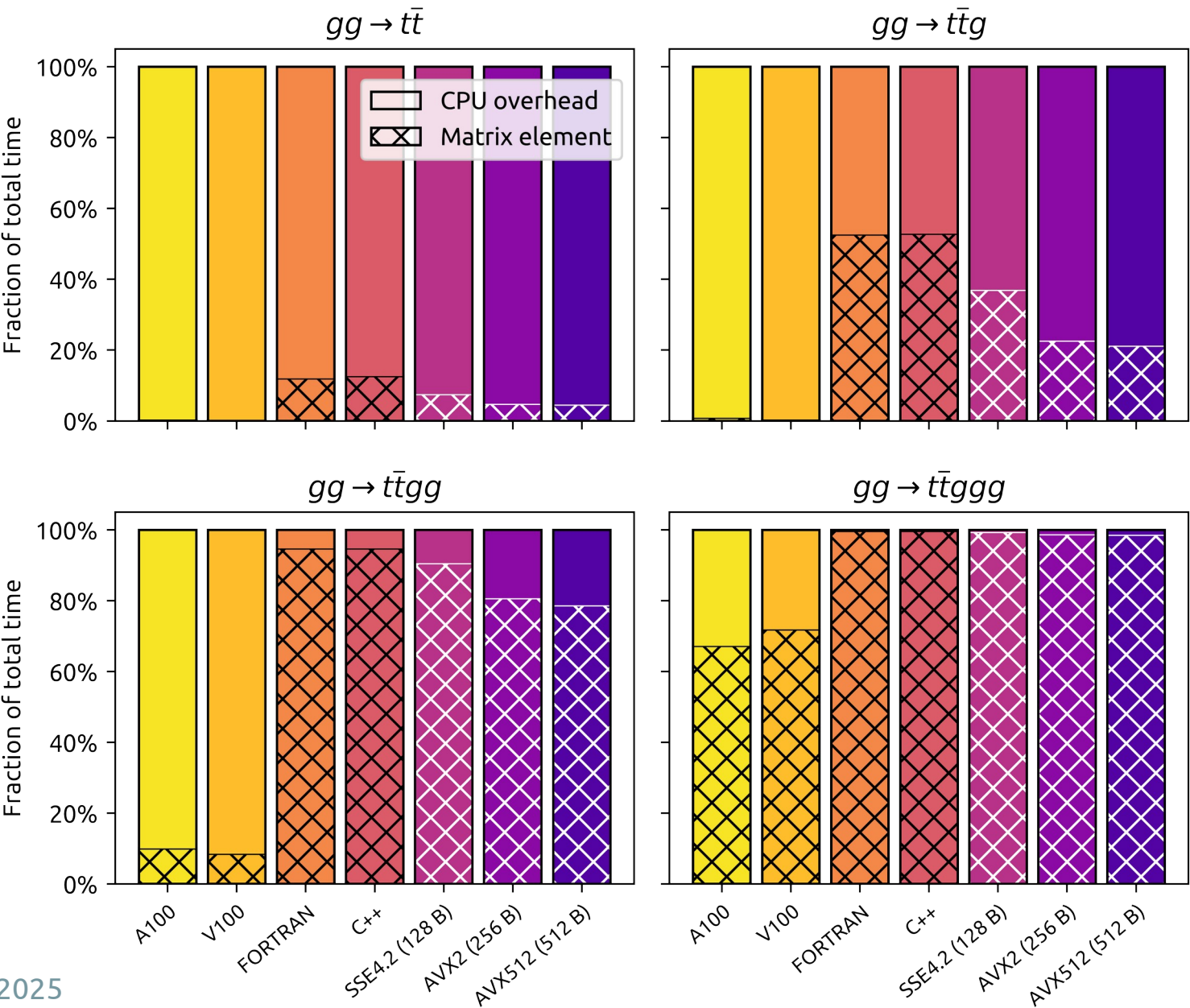
Madevent single thread execution, 10^6 events



Tip for non-experts: the more the “g”s (or the “j”s) at the end of the process name, the more resource-intensive the job is.

- **A100:** AMD EPYC 7313
- **Others:** Intel(R) Xeon(R) Silver 4216

Performance comparisons

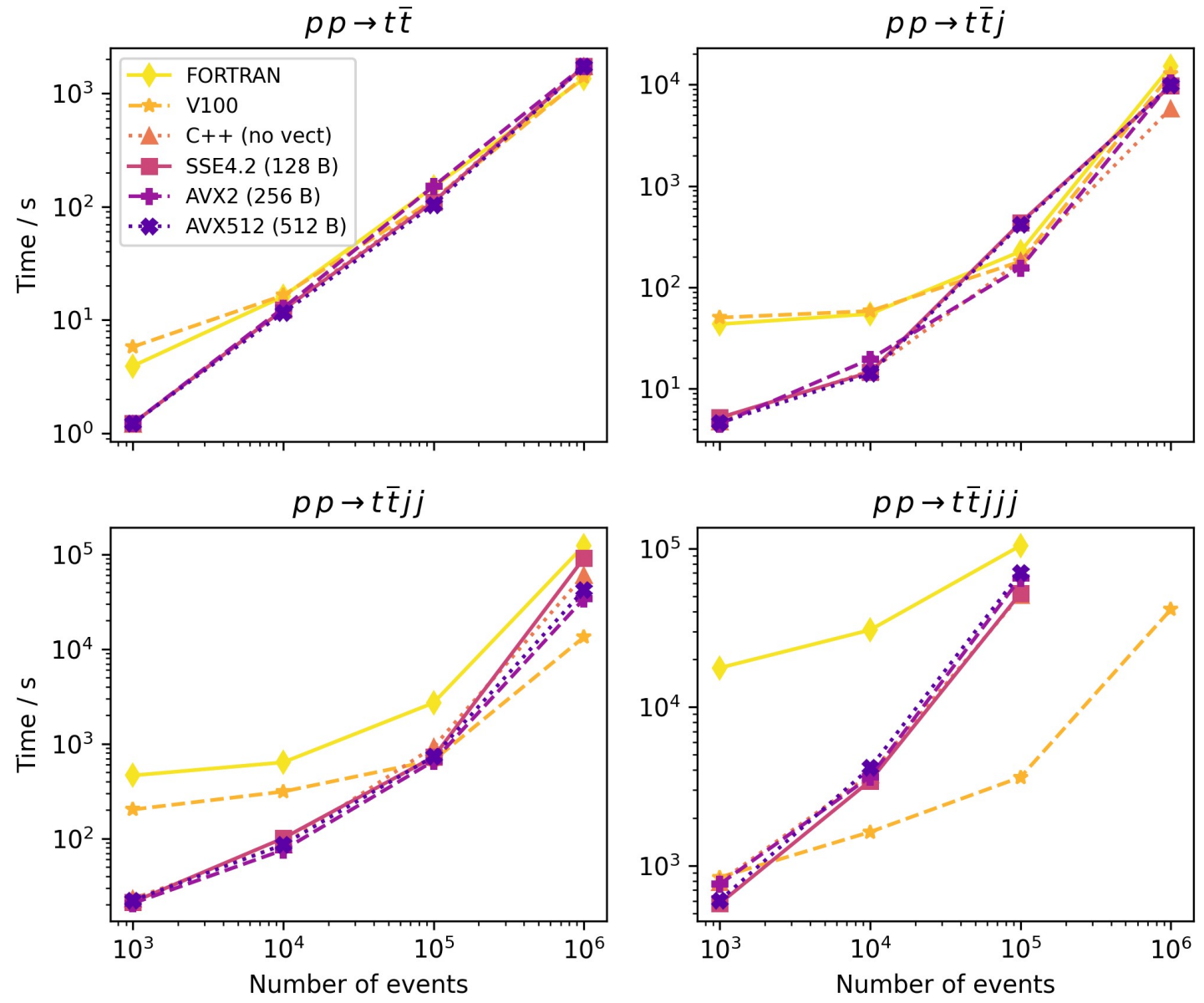


Tip for non-experts: the more the “g”s (or the “j”s) at the end of the process name, the more resource-intensive the job is.

- **A100:** AMD EPYC 7313
- **Others:** Intel(R) Xeon(R) Silver 4216

PRELIMINARY

Performance comparisons



Tip for non-experts: the more the “g”s (or the “j”s) at the end of the process name, the more resource-intensive the job is.

➤ **CPU:**
Intel(R) Xeon(R)
Silver 4216

Floating point precision

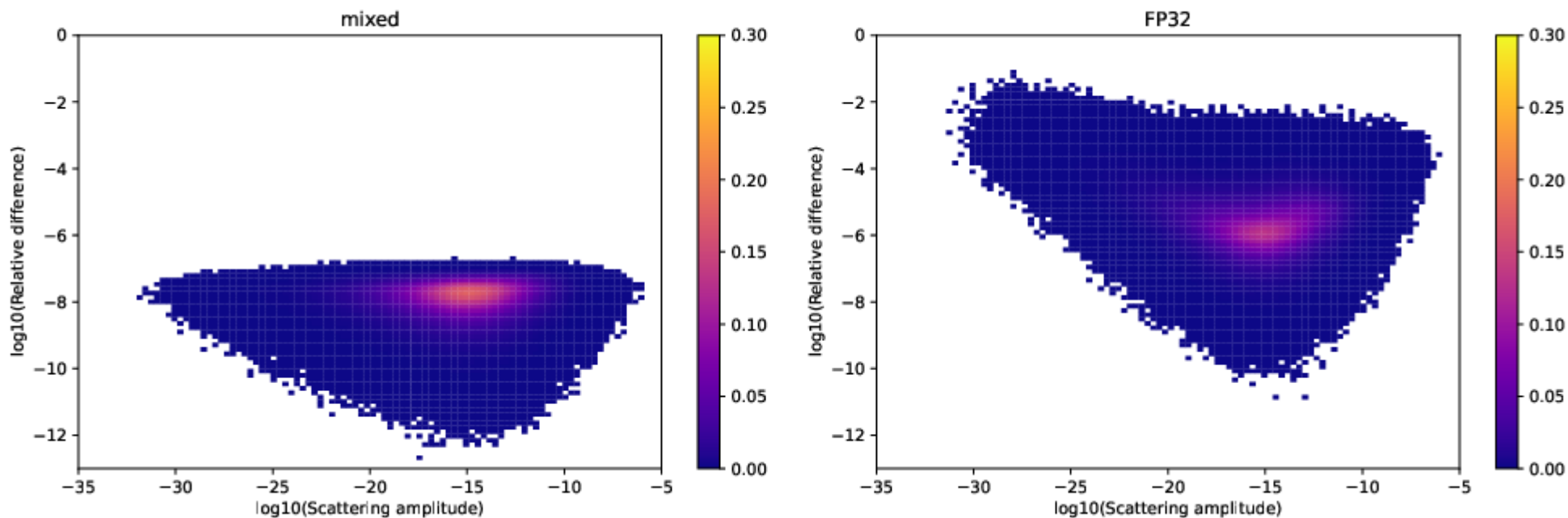
- In the previous plots, we used **double precision** for the matrix element calculation.
- What if we used single precision?
 - Both vectorized C++ and CUDA (V100) would be 2x faster.
- **However**, we need double precision for these computations (helicity amplitudes).
 - Cannot use consumer grade GPUs.
 - For future GPUs (like NVIDIA Blackwell)...you sure get more FP64, but you pay the price of many FP4 tensor cores.
- **Solution** (for now...): mixed-precision:
 - double precision when strictly necessary;
 - single precision anywhere else.

	Arch.	NVIDIA Blackwell	NVIDIA Hopper	NVIDIA Ampere	NVIDIA Volta
	Year	2024	2022	2020	2017
	GPU	GB200	H100	A100	V100
Tensor cores	FP64	90	34	9.7	7.8
	FP32	180	67	19.5	15.7
	FP16	N/A	134	78	-
	FP64	90	67	19.5	-
	TF32	5 000	989	312	-
	FP16	10 000	1979	624	125
	FP8	20 000	3958	-	-
	FP4	40 000	-	-	-
	Numbers in TFLOPS				

Floating point precision

- Relative differences between the mixed/single and double precision, on the basis of the value of the amplitude.
- **Process:** Vector Boson Fusion $p p \rightarrow h j j j j$ with 10^6 events.

Relative difference across phase space for mixed and FP32 ($u u > h 4j$)



- **Mixed precision:** max error around 6-7 digits: it's within float significant digits.
- **Single precision:** we lose significant digits in most of the cases.

Floating point precision

NGT - Openlab "Optimising Floating Point Precision" Workshop

Jul 1 – 2, 2025
CERN
Europe/Zurich timezone



Overview

Timetable

Contribution List

Registration

Participant List

Videoconference

Practical Information

How to reach CERN

Wi-fi Connection

Health insurance, VISA

Privacy information

Contact

✉ [alex.lasa.lamarca@cern...](mailto:alex.lasa.lamarca@cern.ch)

✉ mariana.velho@cern.ch

✉ vasiliki.batsari@cern.ch



Scientific applications in high energy physics depend in many areas on floating point operations in single, double or even higher precision.

**FYI next month
@ CERN!**
Dedicated
workshop on
floating point
precision.

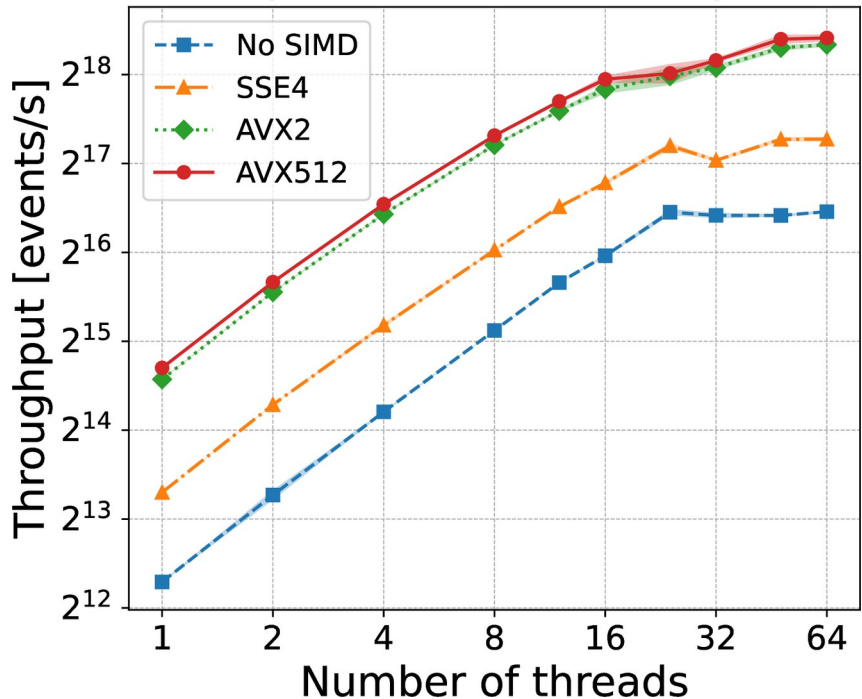


<https://indico.cern.ch/event/1538409/>

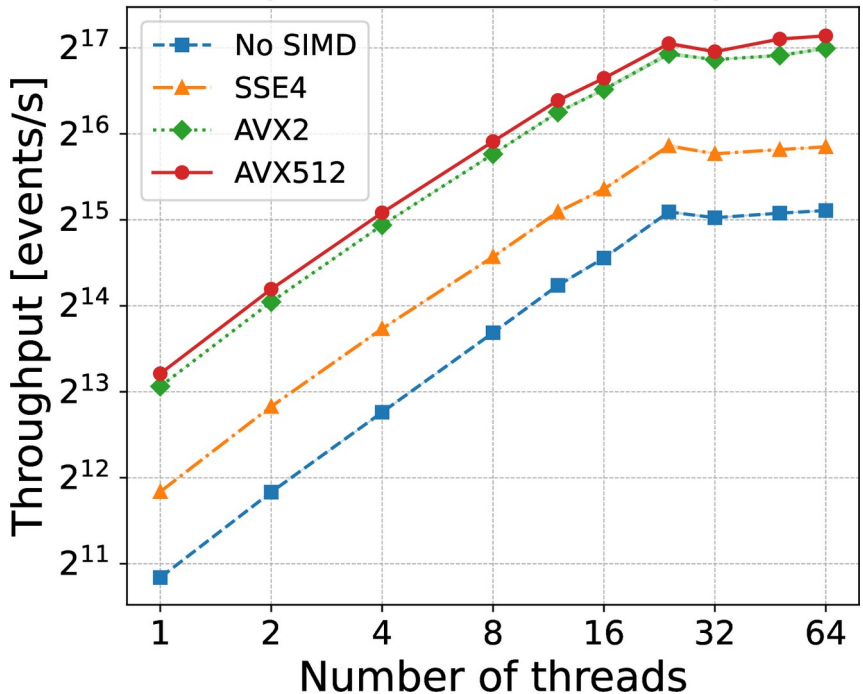
PRELIMINARY

Scaling with threads and instruction set

Throughput by instruction set,
process $u\bar{u} \rightarrow e^+ e^- ggg$
(Intel Gold Xeon 5118)



Throughput by instruction set,
process $gg \rightarrow t\bar{t} gg$
(Intel Gold Xeon 5118)



Conclusions

- **CUDACPP plugin:** MadGraph on GPU & vector CPUs:
 - Hardware acceleration of the matrix element calculation (CPU bottleneck).
 - **Speed-ups** by several orders of magnitudes (process-dependent):
10x to 30x for $p p \rightarrow t \bar{t} \{2,3\}j$ (V100 GPU wrt FORTRAN).
- Other parts of the generation pipeline can be offloaded in the future (phase space integration, parton distribution functions evaluation).
- **Double precision** (FP64) needed for the matrix element, but:
 - Encouraging results when using single precision everywhere else (**mixed precision**).
 - Latest GPUs will have the same (if not worse) FP64/FP32 ratio, while having a lot of FP16/FP8/FP4 cores.
 - For Next-to-Leading Order (NLO) calculation we would probably need quadruple precision.
- Set up the **basis for hardware acceleration** of event generators: can be “recycled” for future efforts, namely NLO computations.