MADGRAPH GPU AND VECTORISATION DEVELOPMENTS A PLAN FORWARD TO BECOME USABLE BY EXPERIMENTS

HSF GENERATORS MEETING, 6 MAY 2021

OLIVIER MATTELAER, STEFAN ROISER, ANDREA VALASSI

SPOILER ALERT

vCHEP 2021

Some info in these slides will be repeated in much more detail by Andrea during his

[vCHEP'21 plenary talk, Wed 19 May, 16:20 – 16:50 CEST]

to follow this talk you need to register at the conference (for free) at
https://indico.cern.ch/event/948465/registrations/63796/

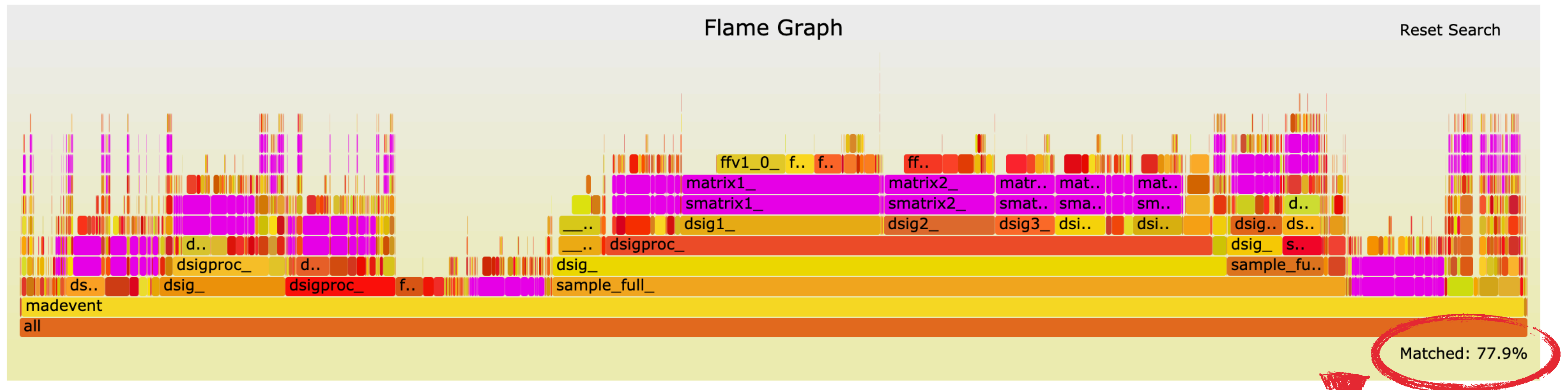
MADGRAPH_AMC@NLO DEVELOPMENT FOR GPUS AND VECTORIZED C++ CODE

- Activity started in Spring 2020, aiming to
  - speed up workflows by porting code to GPUs and improve C++ code
  - work on heterogeneous execution of workflows
  - investigate the use of compute accelerator abstraction layers (Alpaka, Kokkos, oneAPI, …)
  - port the developments upstream into the Madgraph5_aMC@NLO code generator
- Programming languages and tools currently in use: C++, Cuda, Alpaka, Kokkos, SYCL (oneAPI)

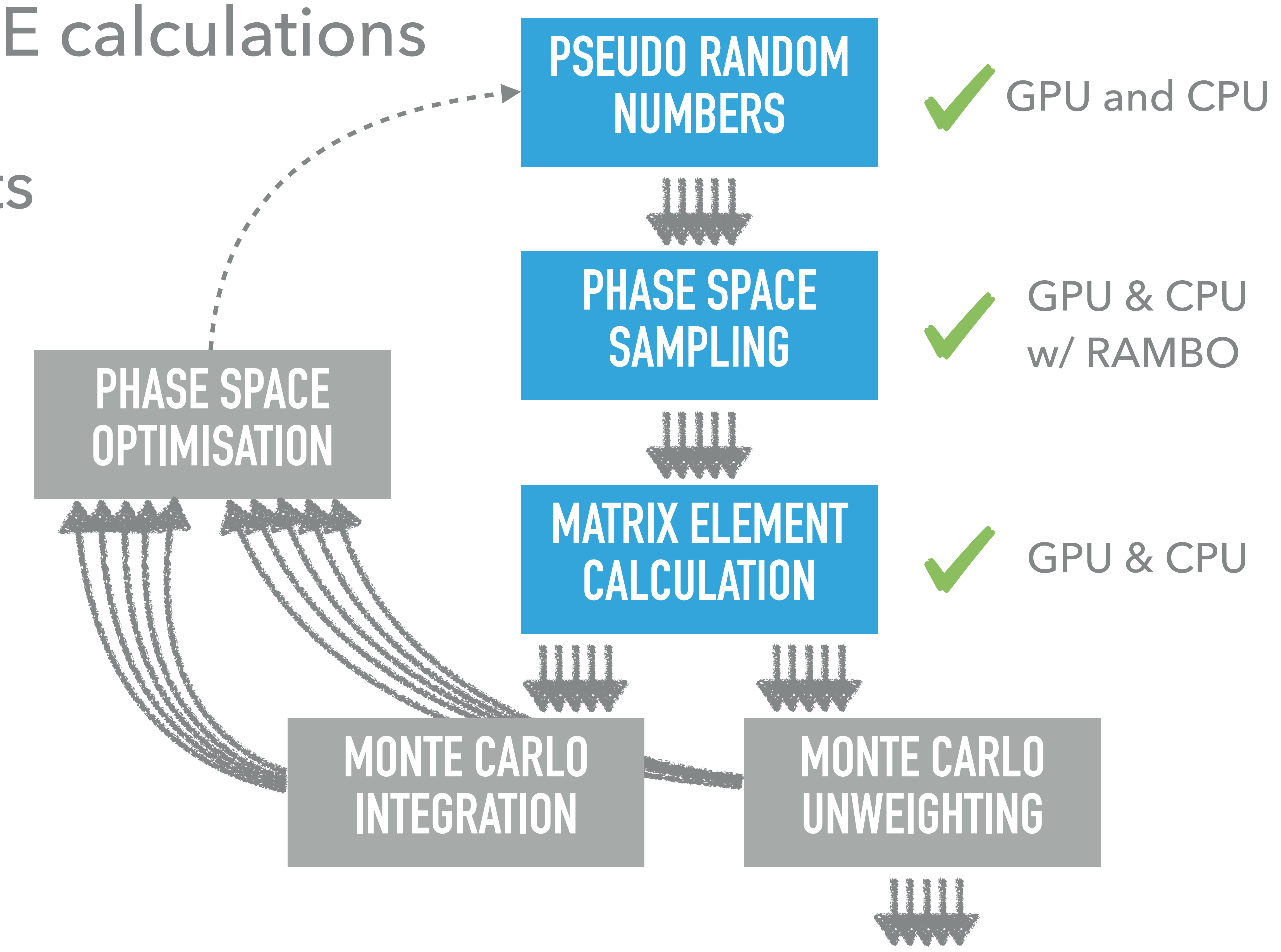
- Web page: https://madgraph5.github.io/
- Indico category (bi-weekly meetings): https://indico.cern.ch/category/12586/
- Mailing list: madgraph5-gpu-development@cern.ch

OVERALL DEVELOPMENT PLAN

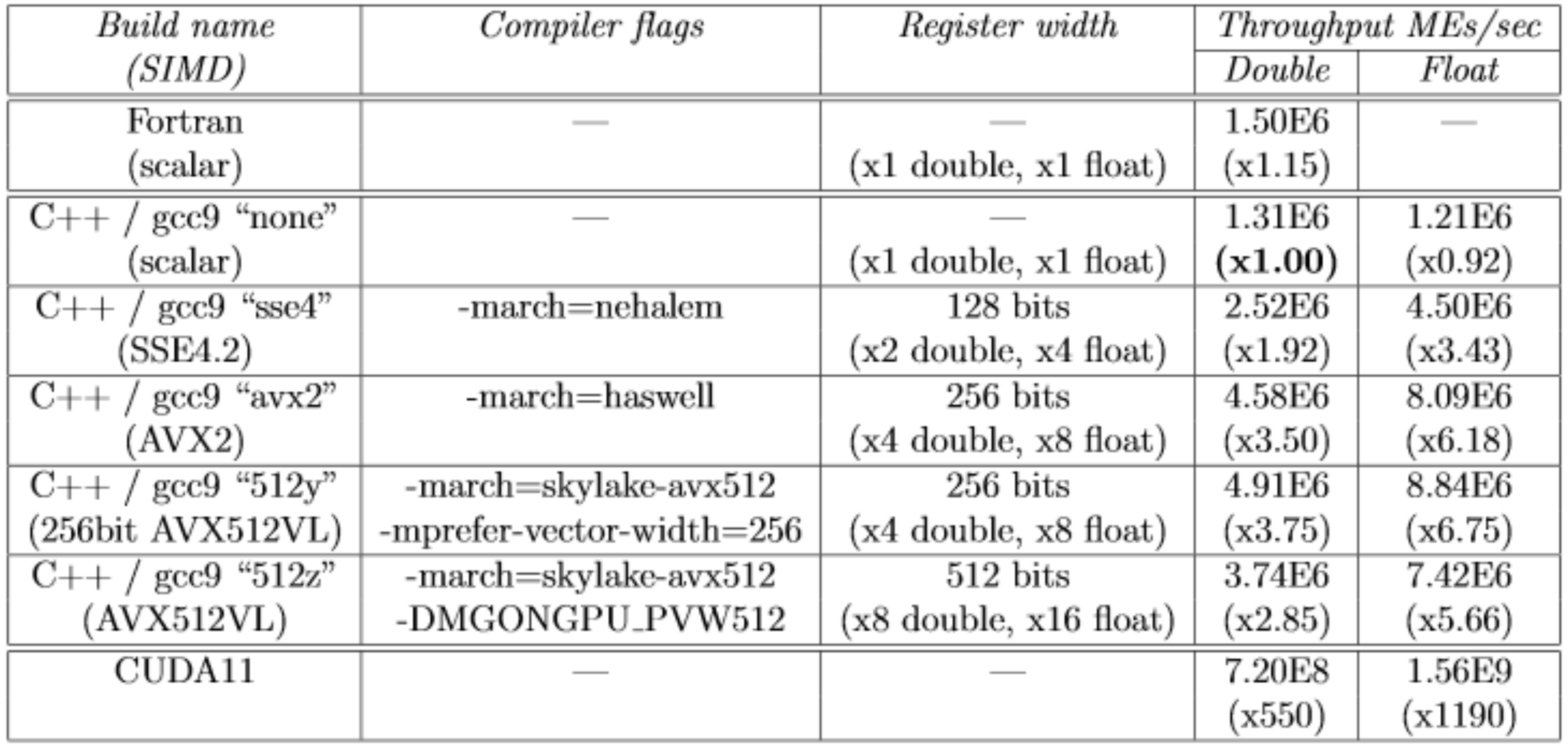
- Matrix element calculation is by far the highest CPU consumer in the workflow
  - –> Concentrate on porting & optimizing ME calculations
- Progressively also port & optimize other parts of the workflow
  - Working versions available for: Random Numbers, Phase space sampling (RAMBO)
  - Missing: MC integration, phase space optimisation, unweighting, PDFs, …



CMS example: p p –> l+ l- j j j j / h @0, 10^5 events



LATEST PERFORMANCE NUMBERS FOR CUDA AND C++ VECTORIZED EXECUTIONS OF MATRIX ELEMENT CALCULATIONS



Build name (SIMD)	*Compiler flags*	*Register width*	*Throughput MEs/sec*	
			Double	*Float*
Fortran (scalar)	—	— (x1 double, x1 float)	1.50E6 (x1.15)	—
C++ / gcc9 "none" (scalar)	—	— (x1 double, x1 float)	1.31E6 **(x1.00)**	1.21E6 (x0.92)
C++ / gcc9 "sse4" (SSE4.2)	-march=nehalem	128 bits (x2 double, x4 float)	2.52E6 (x1.92)	4.50E6 (x3.43)
C++ / gcc9 "avx2" (AVX2)	-march=haswell	256 bits (x4 double, x8 float)	4.58E6 (x3.50)	8.09E6 (x6.18)
C++ / gcc9 "512y" (256bit AVX512VL)	-march=skylake-avx512 -mprefer-vector-width=256	256 bits (x4 double, x8 float)	4.91E6 (x3.75)	8.84E6 (x6.75)
C++ / gcc9 "512z" (AVX512VL)	-march=skylake-avx512 -DMGONGPU_PVW512	512 bits (x8 double, x16 float)	3.74E6 (x2.85)	7.42E6 (x5.66)
CUDA11	—	—	7.20E8 (x550)	1.56E9 (x1190)

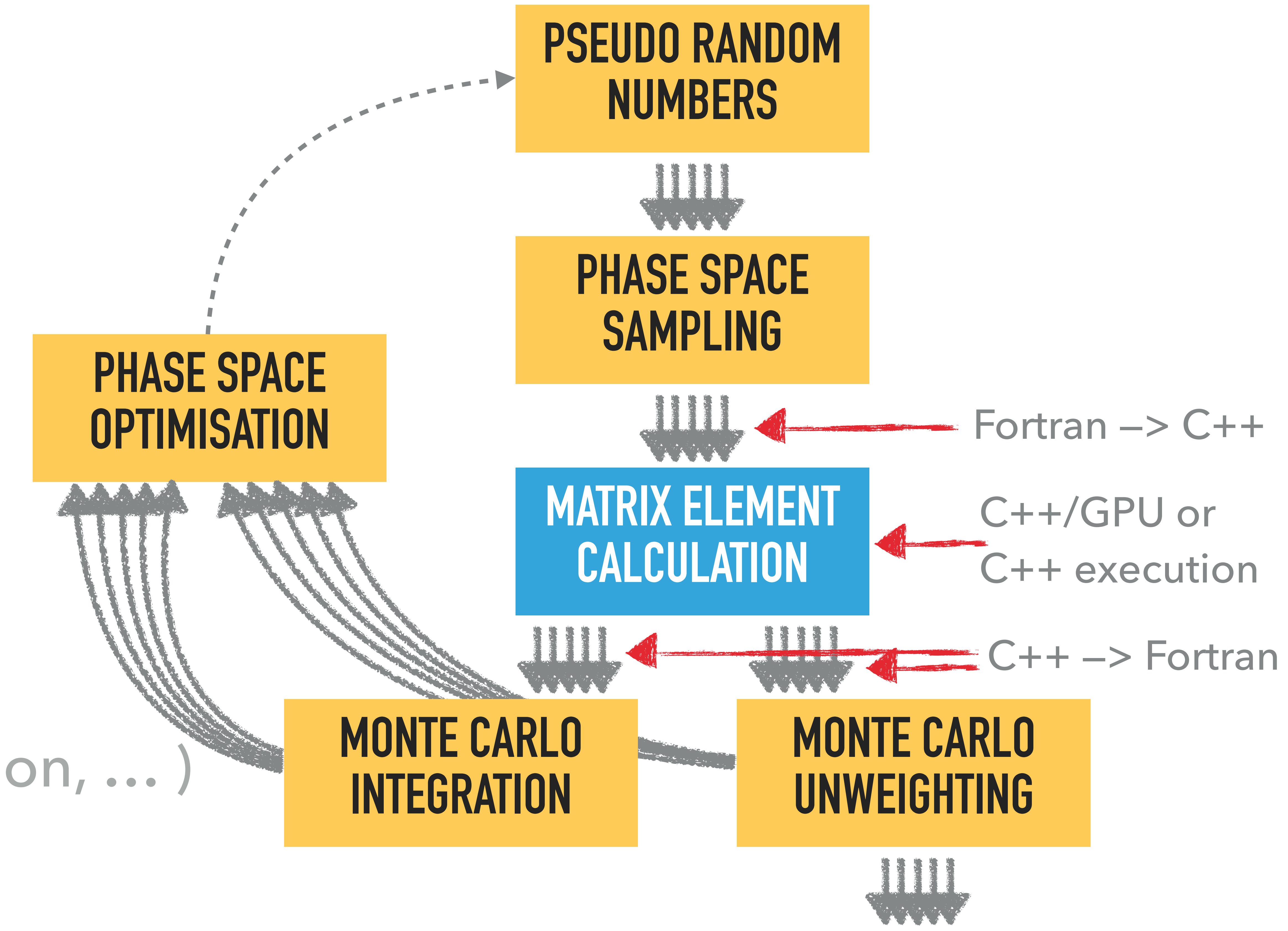
Table 1: Throughputs (matrix elements per second) for eemumu. For Fortran: estimates from MATRIX1 in MadEvent. For C++ and CUDA: measurements from the epoch1 standalone executables, over 12 iterations with 524k events (2048 blocks, 256 threads per block in CUDA), as of commit 51d7f52bf3 on May 04. All builds use "-O3" and "-ffast-math" or "-use_fast_math". Virtual machine itscrd70: skylake-avx512 CPU with 4 virtual cores, NVidia V100 GPU. Fortran and C++ throughputs use a single CPU core. CUDA throughputs include device-to-host copies of all matrix element values.

MORE INFO IN ANDREA'S VCHEP TALK …

POSSIBLE NEXT STEPS IN VIEW OF USEFUL WORKFLOW EXECUTION BY EXPERIMENTS

Can we produce a workflow which is usable by experiments on a ~ medium timescale?

- Use MADEVENT IN FORTRAN for the execution of the workflow outside of the MATRIX ELEMENT CALCULATION C++ / GPU CODE
  - We believe this can be rather soon achieved for LO calculations
  - NLO calculations may be more tricky …
- (Alternatively re-engineering Madevent in C++/GPU will take much more time because of engineering, physics validation, workflow integration, …)



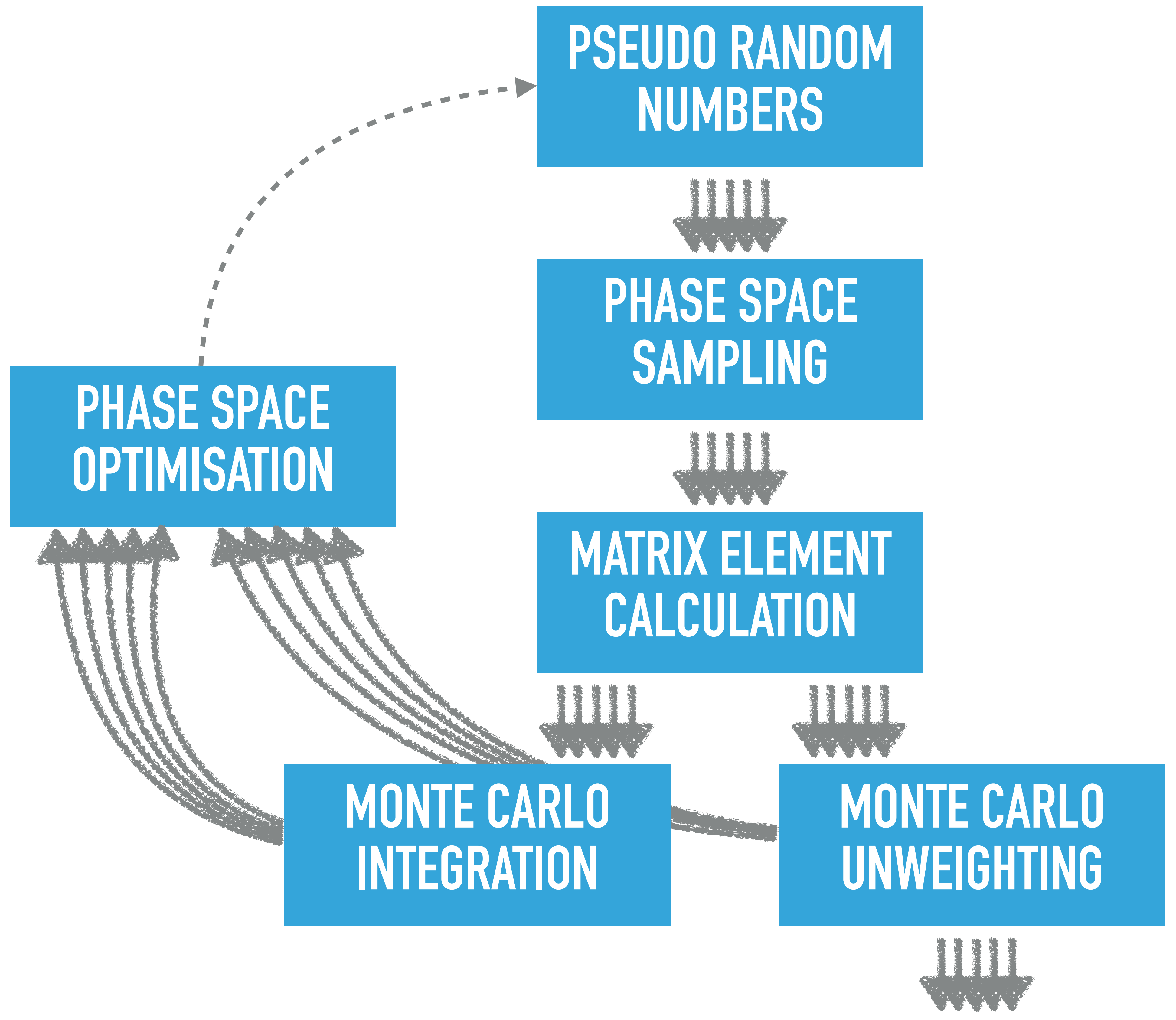
QUESTIONS (MAINLY) TO EXPERIMENTS

- Is a development of a mixed Fortran/C++/GPU workflow useful for experiments?
- Do experiments have access to data centers with GPUs where such a workflow could be tried?
- Alternatively, is the use of vectorized C++ enough motivation for experiments to try such a workflow?

BACKUP

SOME MORE NOTES

- The matrix element calculations can be done in double (default) or single precision floating point precision
- Madgraph currently can produce the Cuda code of the matrix element calculations
  - We are also thinking about other code production backends (Kokkos, …)
-


PSEUDO RANDOM NUMBERS
PHASE SPACE SAMPLING
MATRIX ELEMENT CALCULATION
PHASE SPACE OPTIMISATION
MONTE CARLO INTEGRATION
MONTE CARLO UNWEIGHTING