GPU DEVELOPMENT IN THE CONTEXT OF THE MADGRAPH5_AMC@NLO EVENT GENERATOR

HSF WLCG WORKSHOP, 20 NOV 2020

STEFAN ROISER

MADGRAPH_AMC@NLO (IN THE CONTEXT OF THIS ENGINEERING WORK)

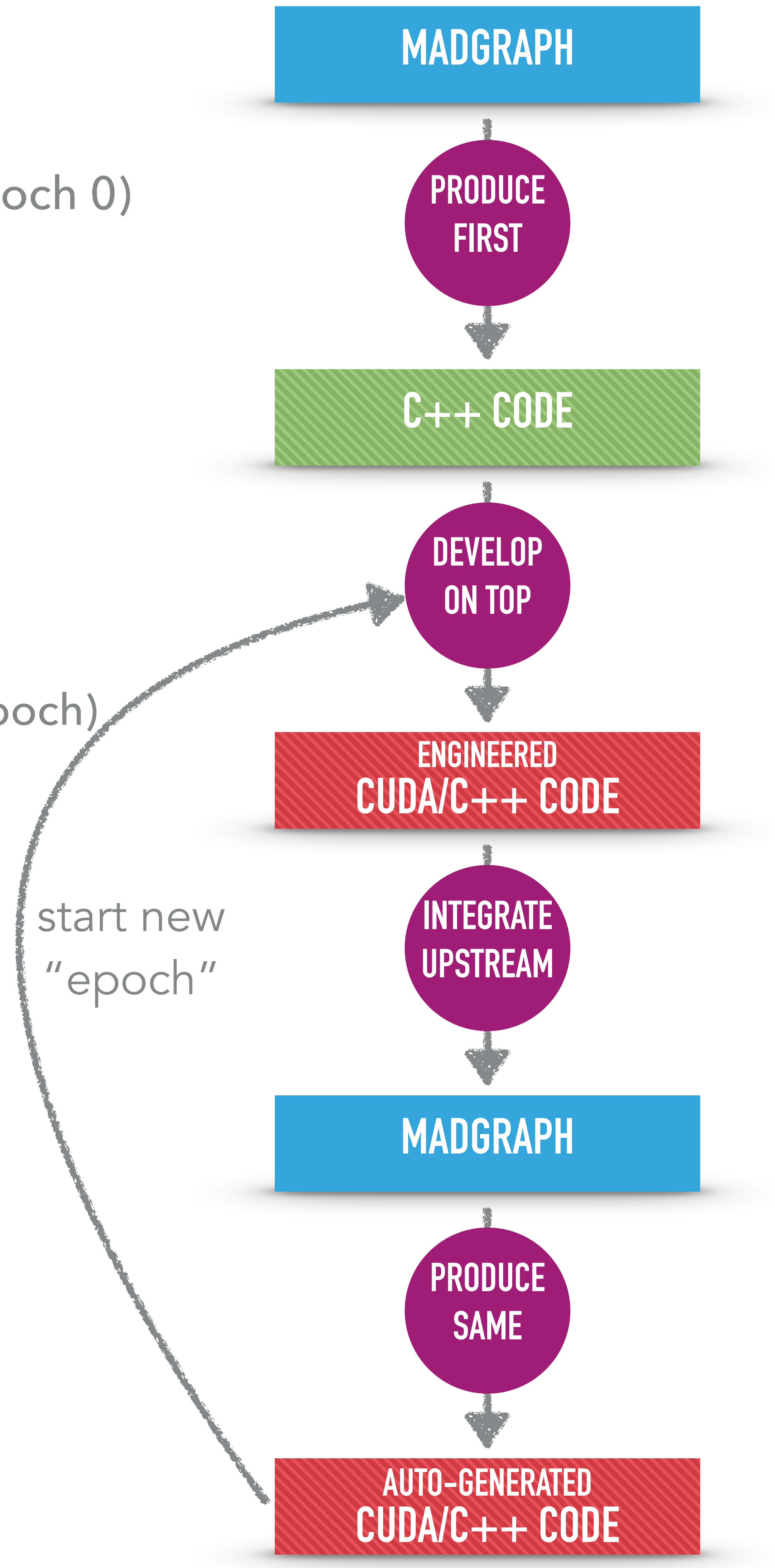
- Madgraph_aMC@NLO is a
  - Monte Carlo event generator used by HEP experiments (e.g. ATLAS & CMS)
  - Code generator, written in Python, to produce source code in multiple languages (Fortran, C, C++, MPI) for the calculation of physics processes
  - Framework that integrates additional functionality needed for the execution of the overall workflow (random number generation, parton distribution function, phase space sampling/integration, unweighted event generation, cross section calculation, …)

WHO IS WORKING WITHIN THIS ACTIVITY

- Established a very active connection to Olivier Mattelaer, a core Madgraph developer
- Very good mix of people with expertise in physics theory, applied physics and software engineering
  - Cuda development: Andrea Valassi, OM, Stephan Hageboeck, Taran Singhania, SR
  - Abstraction layers & profiling: David Smith, Laurence Field, Smita Darmora, Taylor Childers, Tyler Burch, Walter Hopkins
- Bi-weekly developers meeting

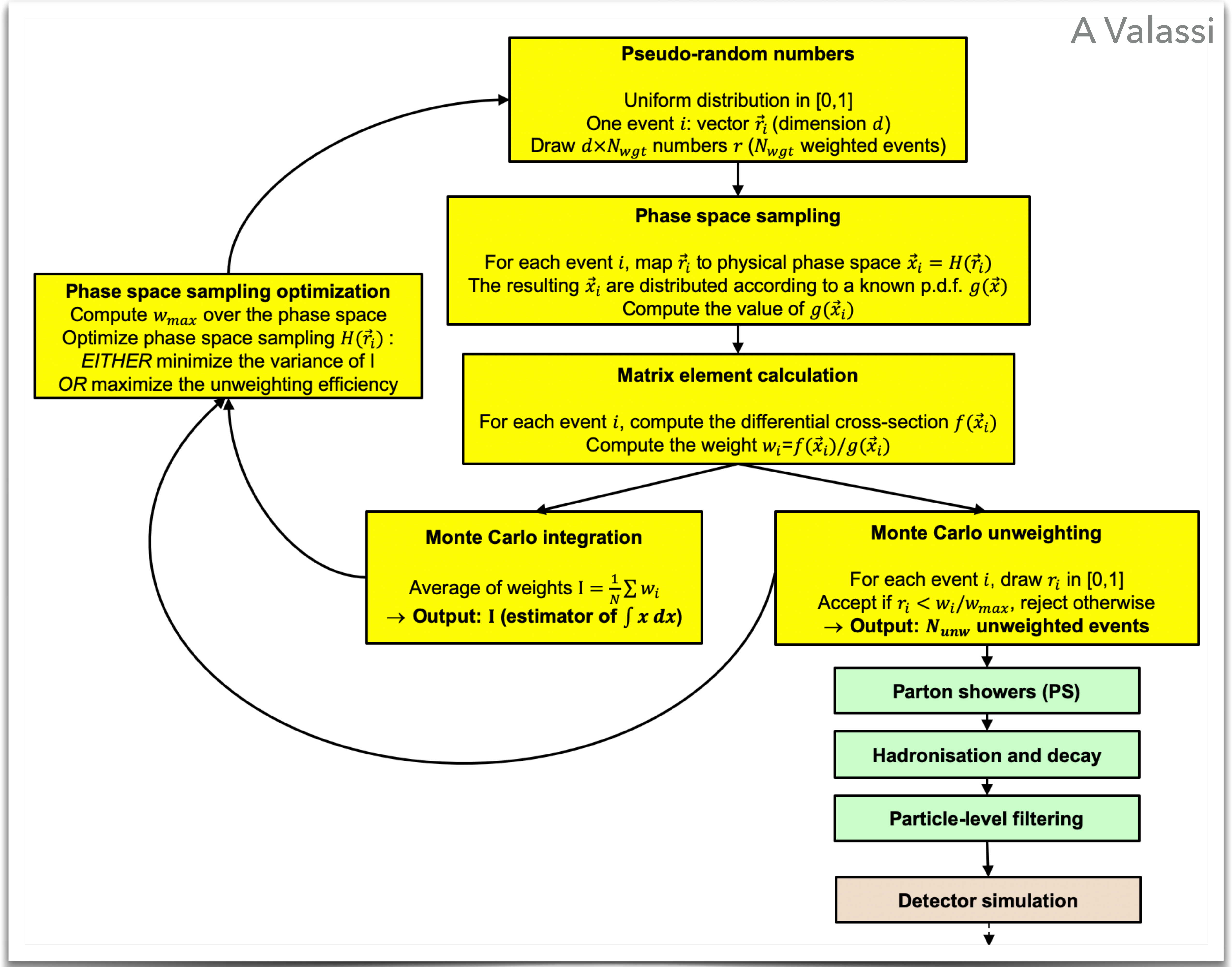
THE SOFTWARE ENGINEERING PROCESS

1. Let Madgraph produce the C++ code for a simple first physics process (epoch 0)
2. Develop Cuda on top of this first version for one part of the workflow
3. while (true)
    3.1. Integrate changes upstream into the Madgraph code generator
    3.2. Let Madgraph (re)produce same and more complex processes (new epoch)
    3.3. Develop on top of the newly generated code to
        - optimise physics processes for accelerator execution
        - test new cuda features
        - port further parts of the workflow
        - work on and test more complex physics processes
        - use as baseline for hw abstraction code



THE MONTE CARLO EVENT GENERATION WORKFLOW

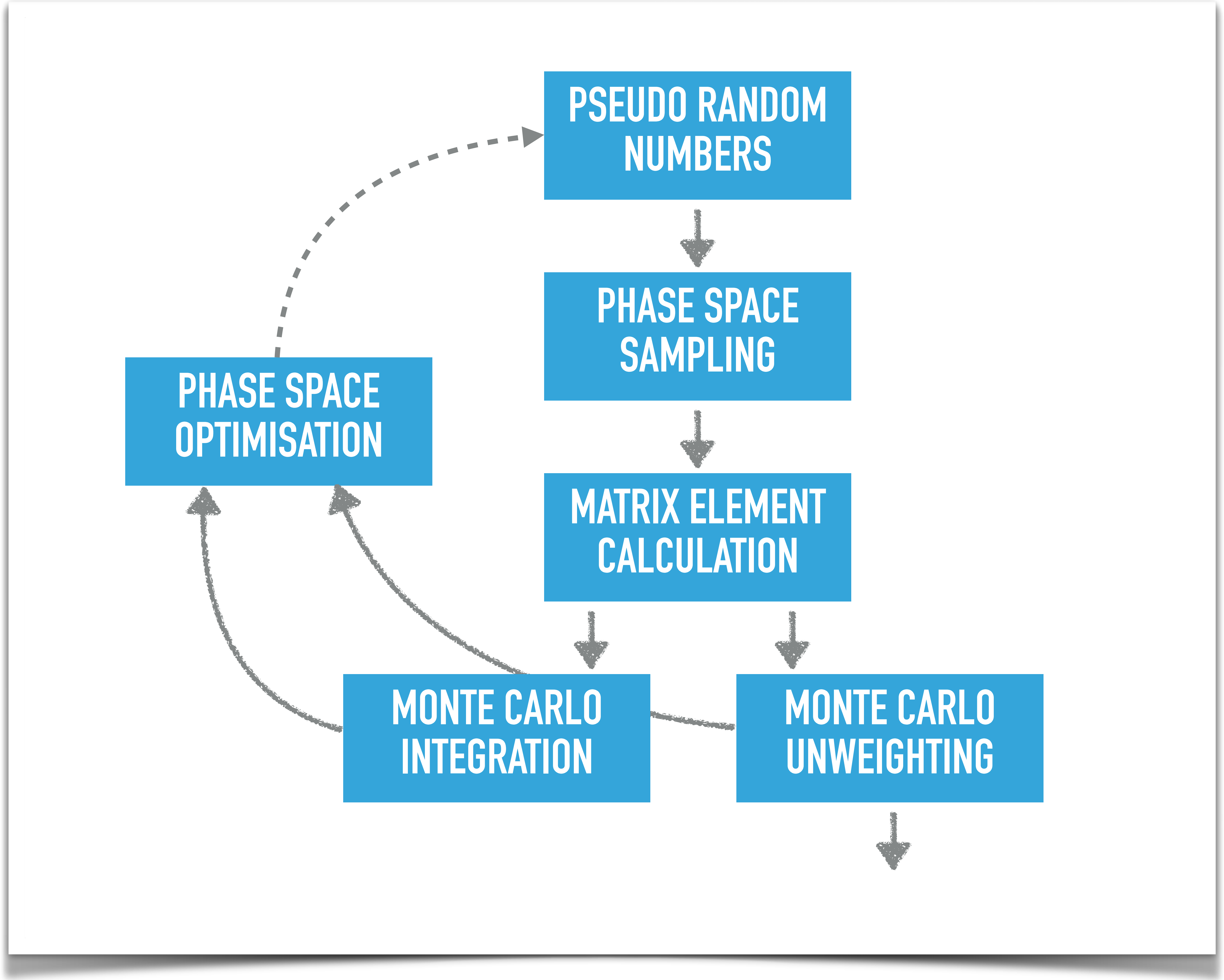
- Start with random numbers and map those to 4-momenta
- Calculate Feynman diagrams describing the physics processes
- Optimise the workflow once for the phase space of the physics process
- Mass production of events



THE MONTE CARLO EVENT GENERATION WORKFLOW, SIMPLIFIED VIEW

Specifics of the workflow, especially relevant for porting on accelerators:

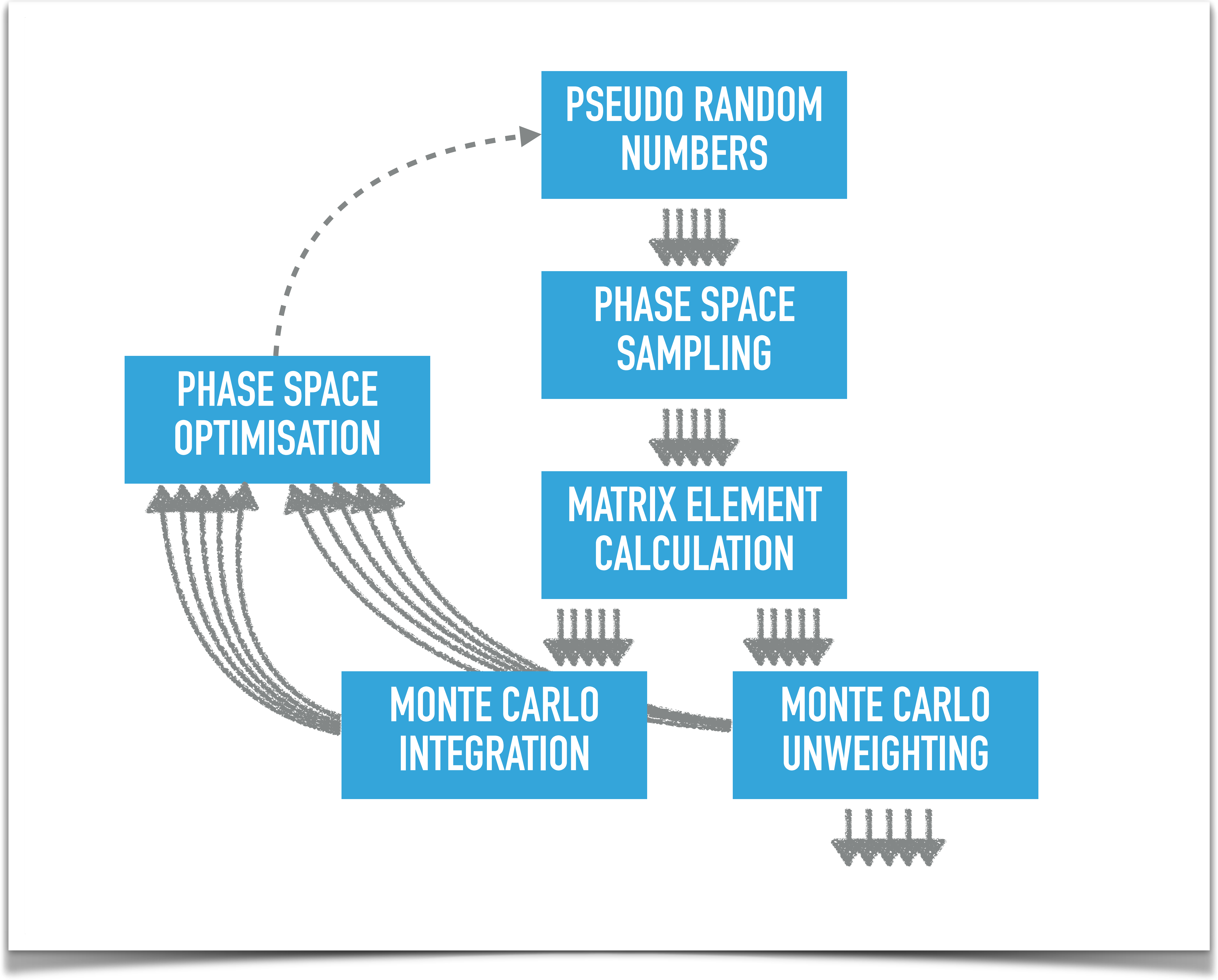
- Random numbers generated on device
  - ☑ Workflow has no input data



THE MONTE CARLO EVENT GENERATION WORKFLOW, SIMPLIFIED VIEW

Specifics of the workflow, especially relevant for porting on accelerators:

- Random numbers generated on device
  - ☑ Workflow has no input data
- Parallelisation on "event level"
  - ☑ Every part can be split into smaller ones if too complex / heavy
- All cores follow same code execution path
  - ☑ no thread divergence



WHERE TO START?

- E.g. real world CMS example: p p –> l+ l- j j j j / h @0
  - Madgraph/MadEvent (Fortran), 10^5 events

http://sroiser.web.cern.ch/sroiser/madgraph5/profiling/profiling.html

WHERE TO START?

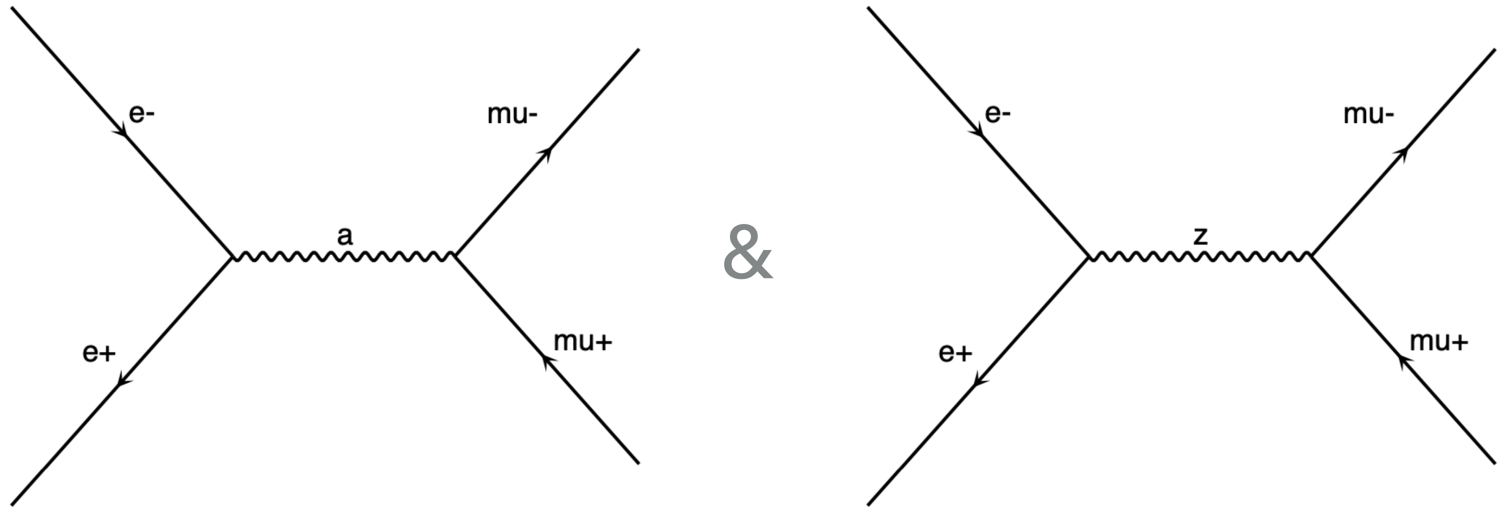
- E.g. real world CMS example: p p –> l+ l- j j j j / h @0
  - Madgraph/MadEvent (Fortran), 10^5 events

http://sroiser.web.cern.ch/sroiser/madgraph5/profiling/profiling.html

- Matrix element calculations use majority of CPU time

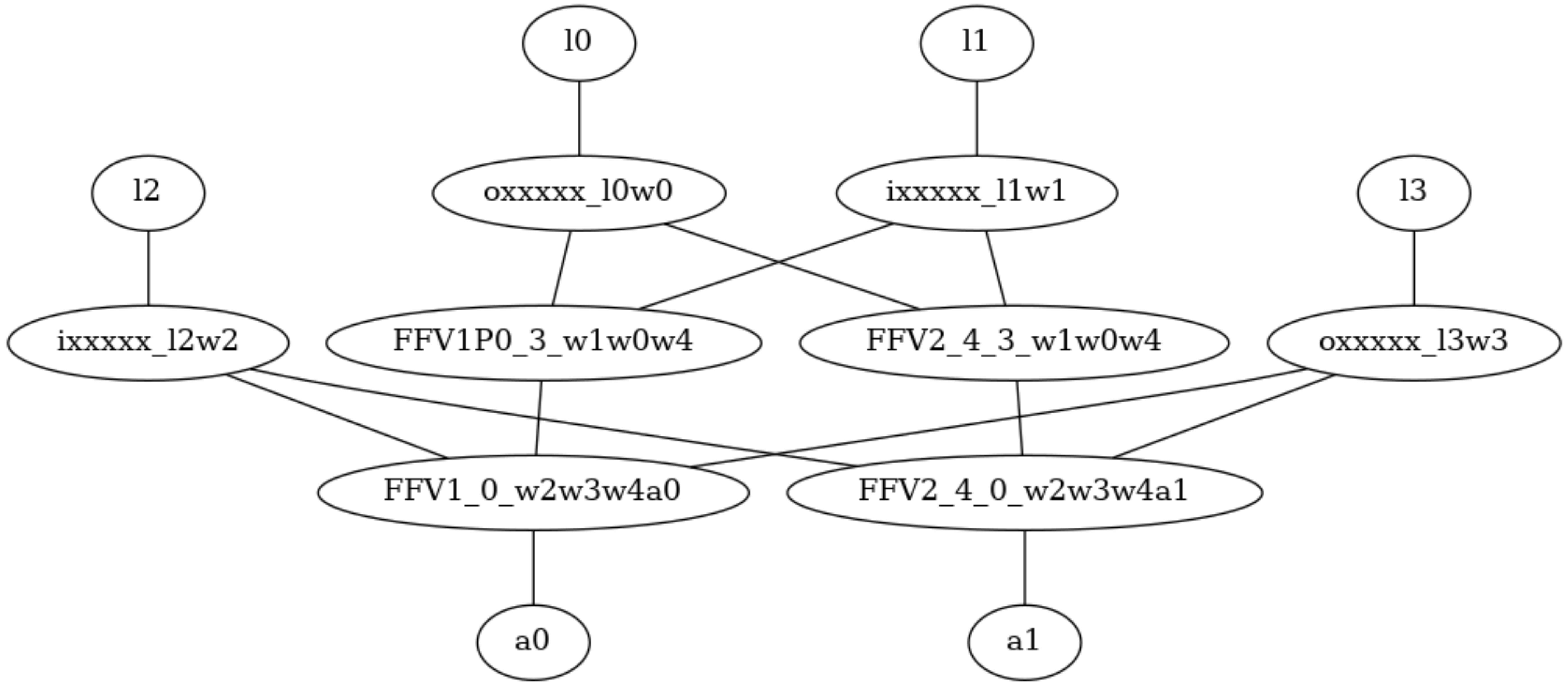
MATRIX ELEMENT CALCULATION

WHERE TO START?



- Produce C++ version of simple(st) process: $e^+e^- \rightarrow \mu^+\mu^-$
  - two diagrams, no PDF, 499 lines of code for matrix element calculations
- Start porting the matrix element calculations of this process to Cuda (aka “epoch 0”)

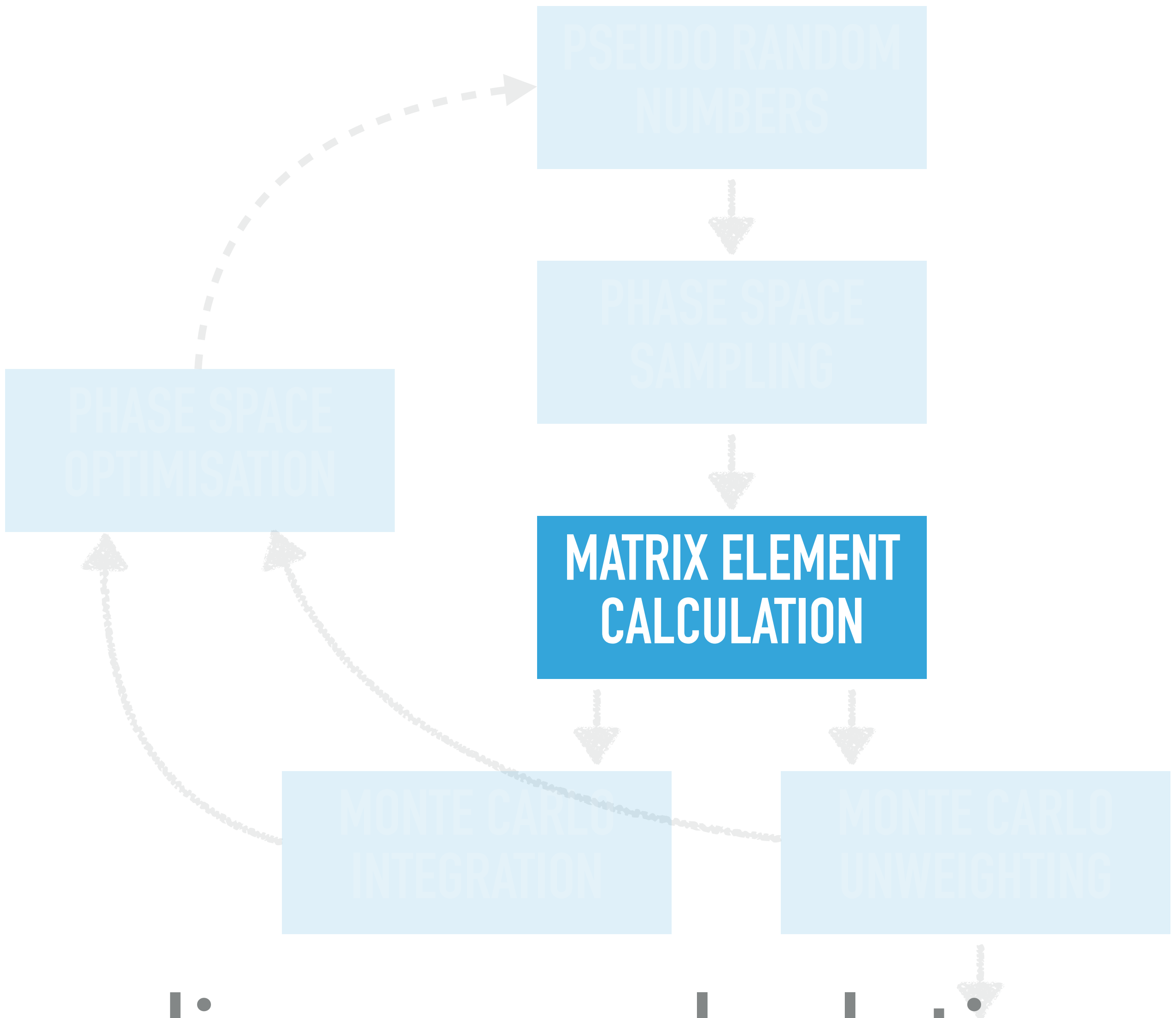
```
void CPPProcess::sigmaKin(bool ppar) {
  // read  parameters
  for (int ihel = 0; ihel < ncomb; ihel++) {
    calculate_wavefunctions(perm, helicities[ihel]);
    t[0] = matrix_1_epem_mupmum();
  }
}

void CPPProcess::calculate_wavefunctions(/* ... */) {
  oxxxxx(p[0], mME[0], hel[0], -1, w[0]);
  ixxxxx(p[1], mME[1], hel[1], +1, w[1]);
  ixxxxx(p[2], mME[2], hel[2], -1, w[2]);
  oxxxxx(p[3], mME[3], hel[3], +1, w[3]);
  FFV1P0_3(w[1], w[0], par3, w[4]);
  FFV2_4_3(w[1], w[0], -par51, par59, w[5]);
  // Calculate all amplitudes
  FFV1_0(w[2], w[3], w[4], par3, amp[0]);
  FFV2_4_0(w[2], w[3], w[5], par51, par59, amp[1]);
}

```



EPOCH 0

(spring 2020)

SOFTWARE ENGINEERING DURING EPOCH 0 – MATRIX ELEMENT CALCULATIONS



- Memory allocations
  - Initially used specialised cuda 2D and 3D structures
    reworked later allocate flat memory blocks and organise structs within
  - Data organised in SOA
- Call one cuda kernel (sigmaKin) and call the remaining feynman diagram calculations as separate functions within

☑ First implementation of matrix element calculations done at the end of epoch0
(~ x 200 performance over single threaded CPU process)

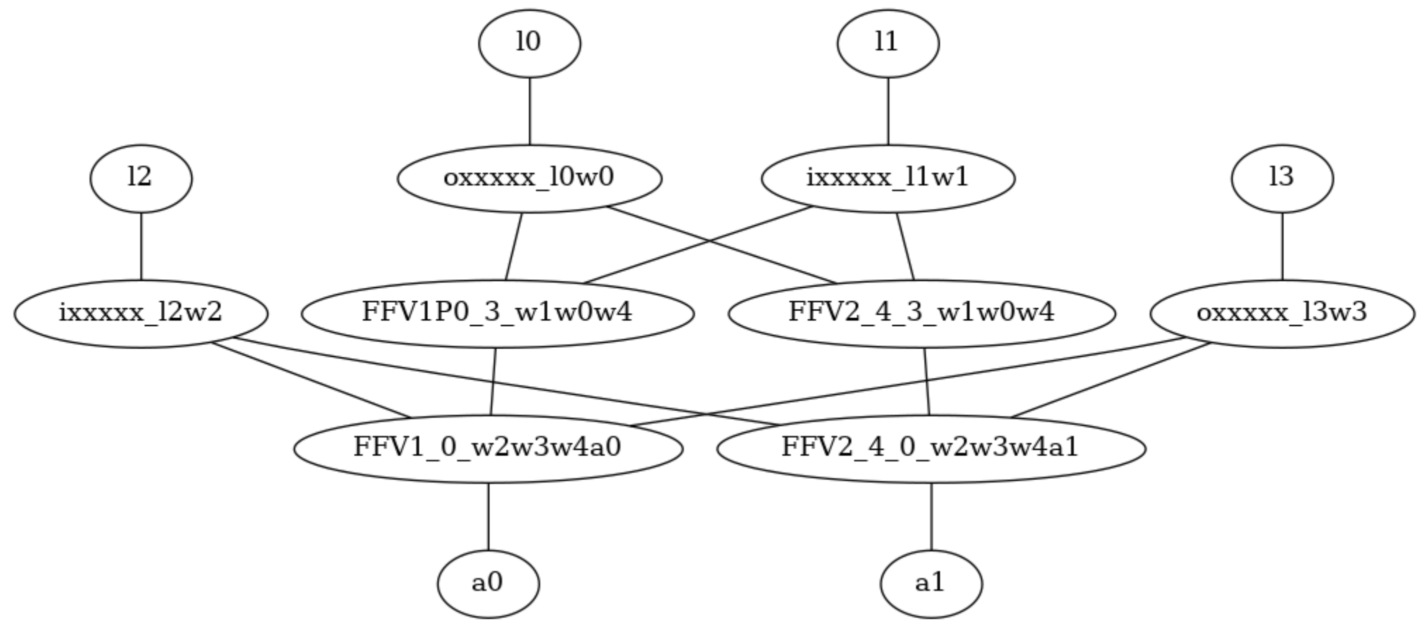
☑ Code changes ported upstream into the Madgraph code generator

EPOCH 1

(summer 2020)

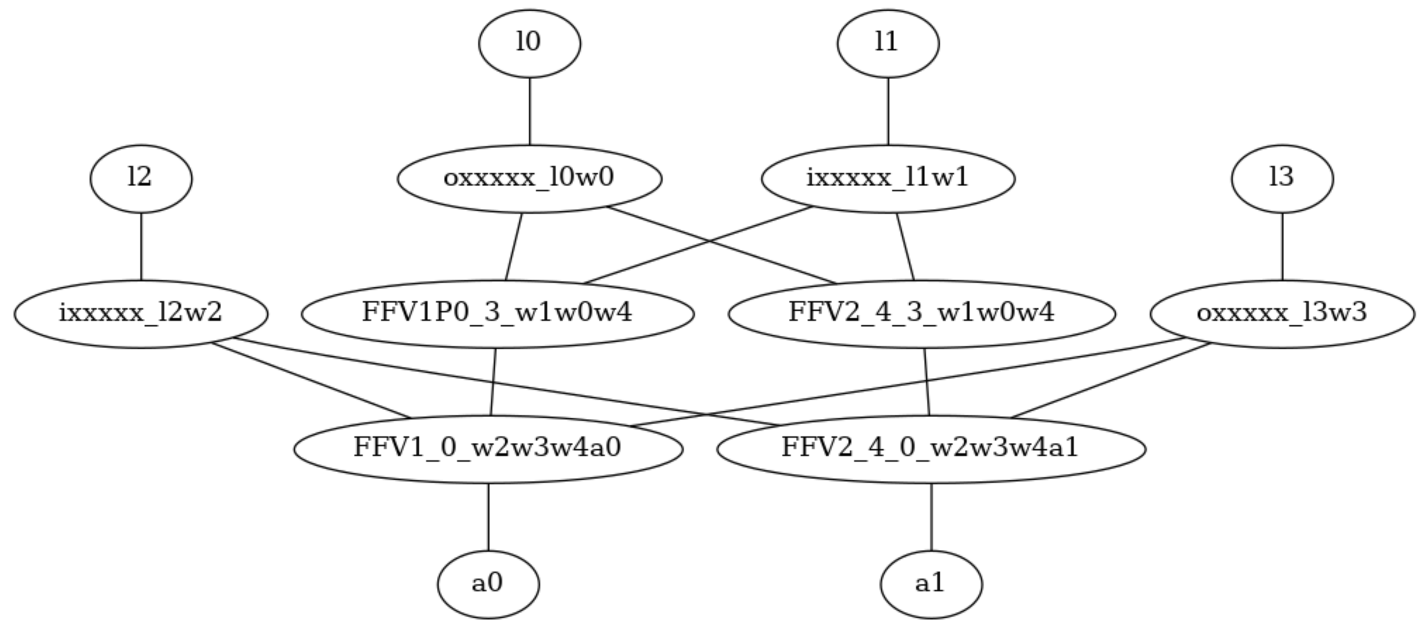
EPOCH 1

- Auto-generating $e^+e^- \rightarrow \mu^+\mu^-$

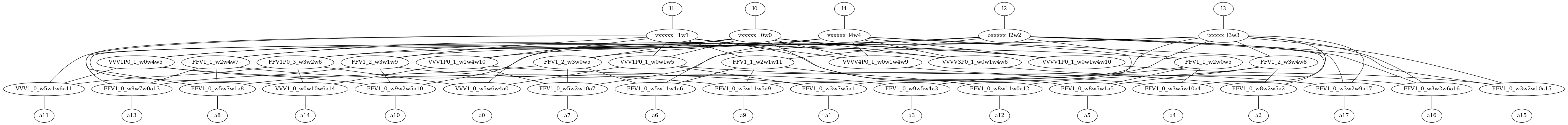


EPOCH 1

- Auto-generating $e^+e^- \rightarrow \mu^+\mu^-$

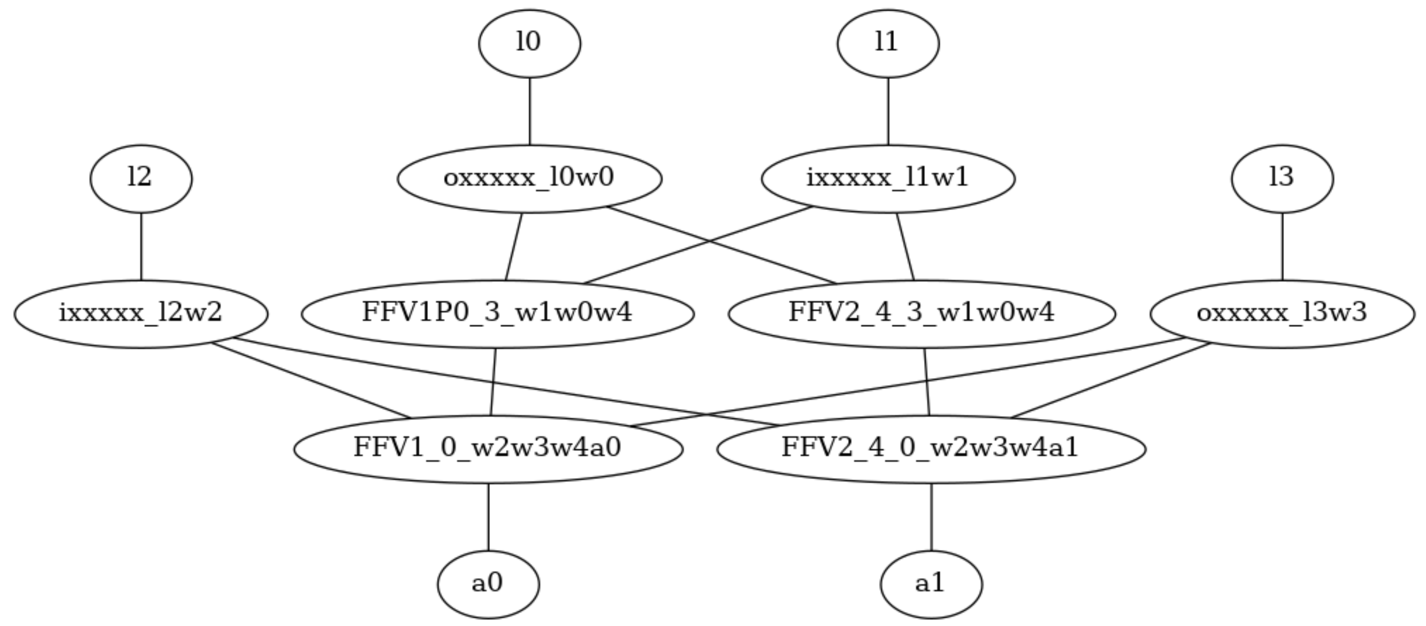


- … and additional more complex processes, e.g. gg → $t\bar{t}g$

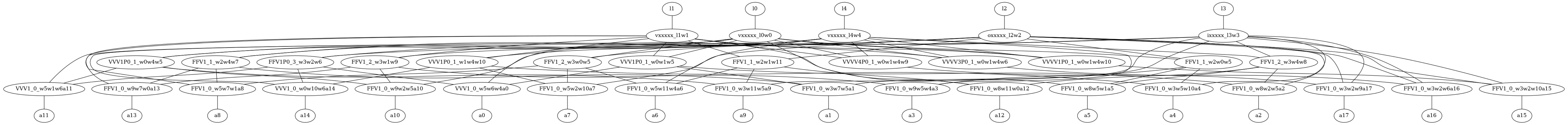


EPOCH 1

- Auto-generating $e^+e^- \rightarrow \mu^+\mu^-$

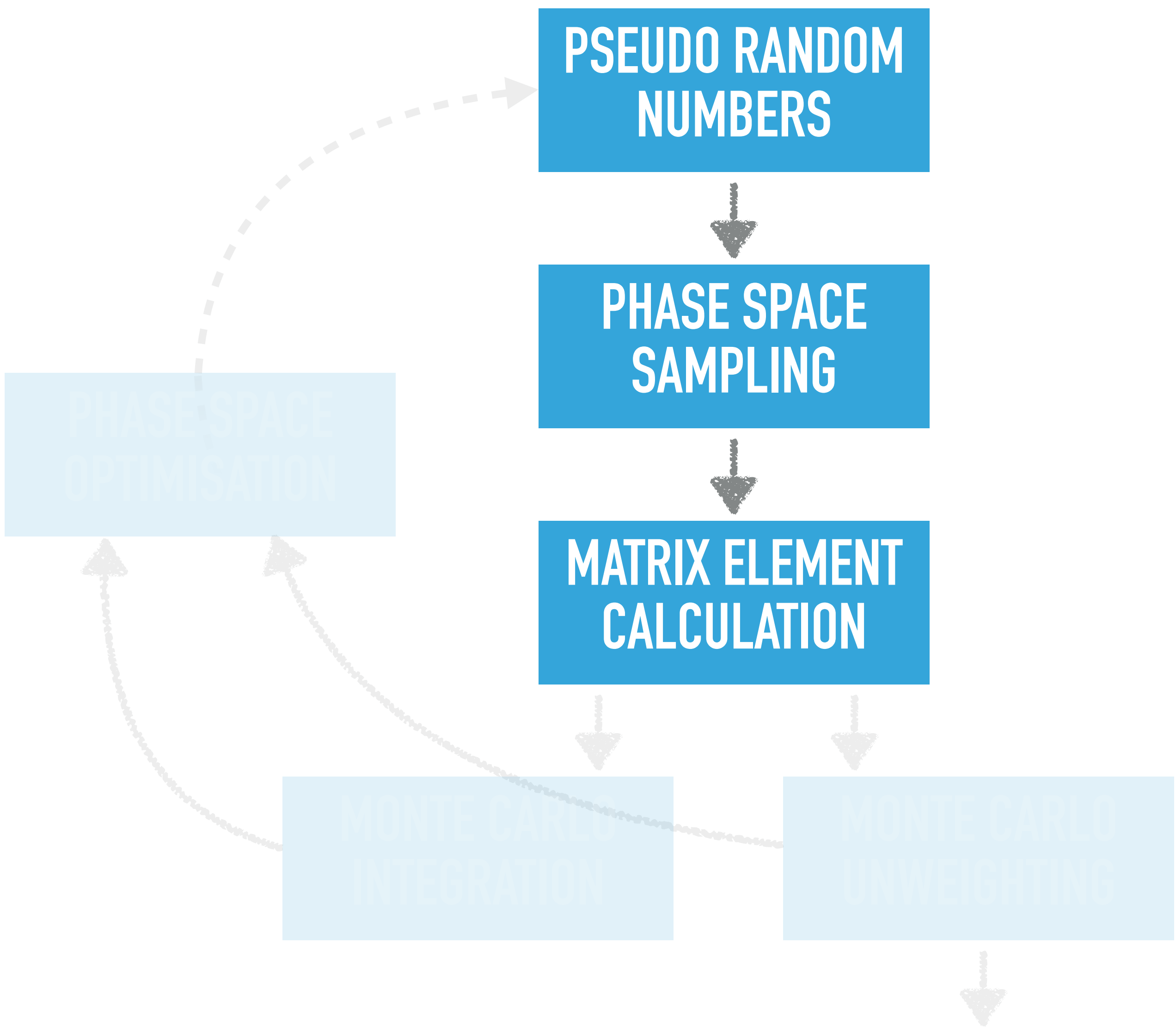


- … and additional more complex processes, e.g. $gg \rightarrow t\bar{t}g$



- … or $gg \rightarrow t\bar{t}gg$

SOFTWARE ENGINEERING DURING EPOCH 1



- Move random number generation to GPU
- Port (simple) phase space sampling ("Rambo") to GPU
- Removal of STL structures on interfaces
  - Nice side-effect –> resulted also in CPU performance improvements
- Introduce switches (C++ macros) to change workflow characteristics
  - Double / single precision for complex arithmetics
  - AOS, SOA, AOSOA data structures
  - Complex arithmetics implementations (cuComplex, thrust::complex, self-woven)
  - Execution of same codebase on host (C++) or device (Cuda), yielding exactly the same results
- Improvements for performance measurements and profiling (Cuda NVTX, json output, …)

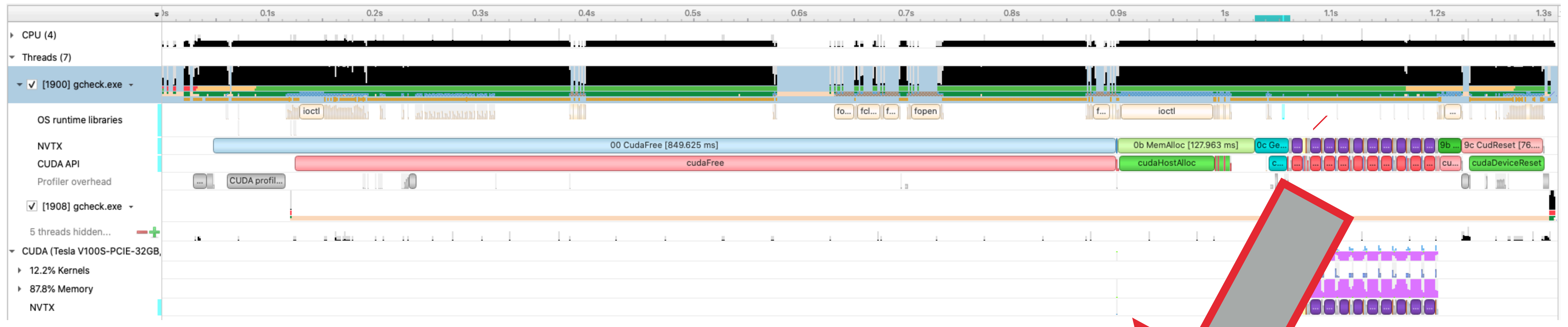
PROFILING

$\underline{e^+e^- \to \mu^+\mu^-}$

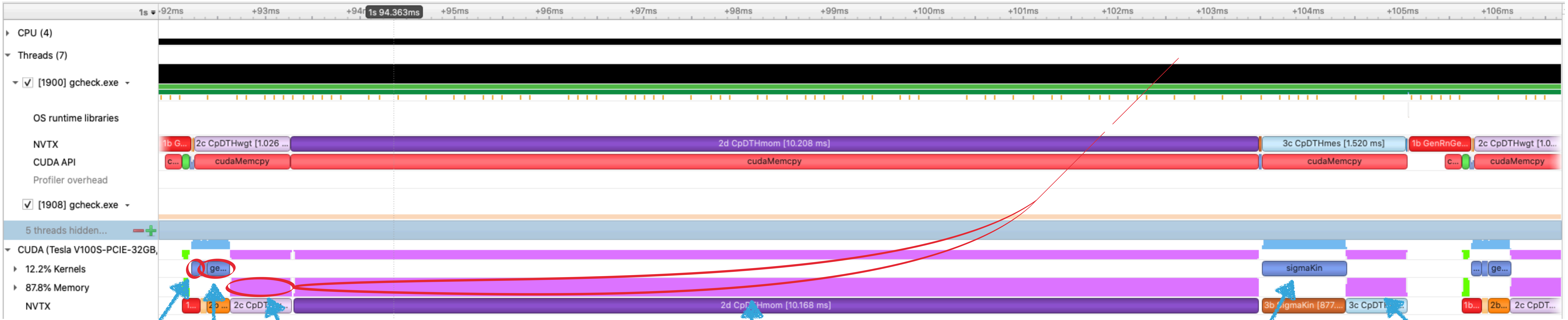
4096 blocks / grid

256 threads / block

10 iterations



$\underline{e^+e^- \to \mu^+\mu^-}$, 4096 blocks / grid, 256 threads / block, *1 iteration*



Random Numbers

Phase Space Sampling

Copy Weights Device –> Host

Copy 4-Momenta Device –>Host

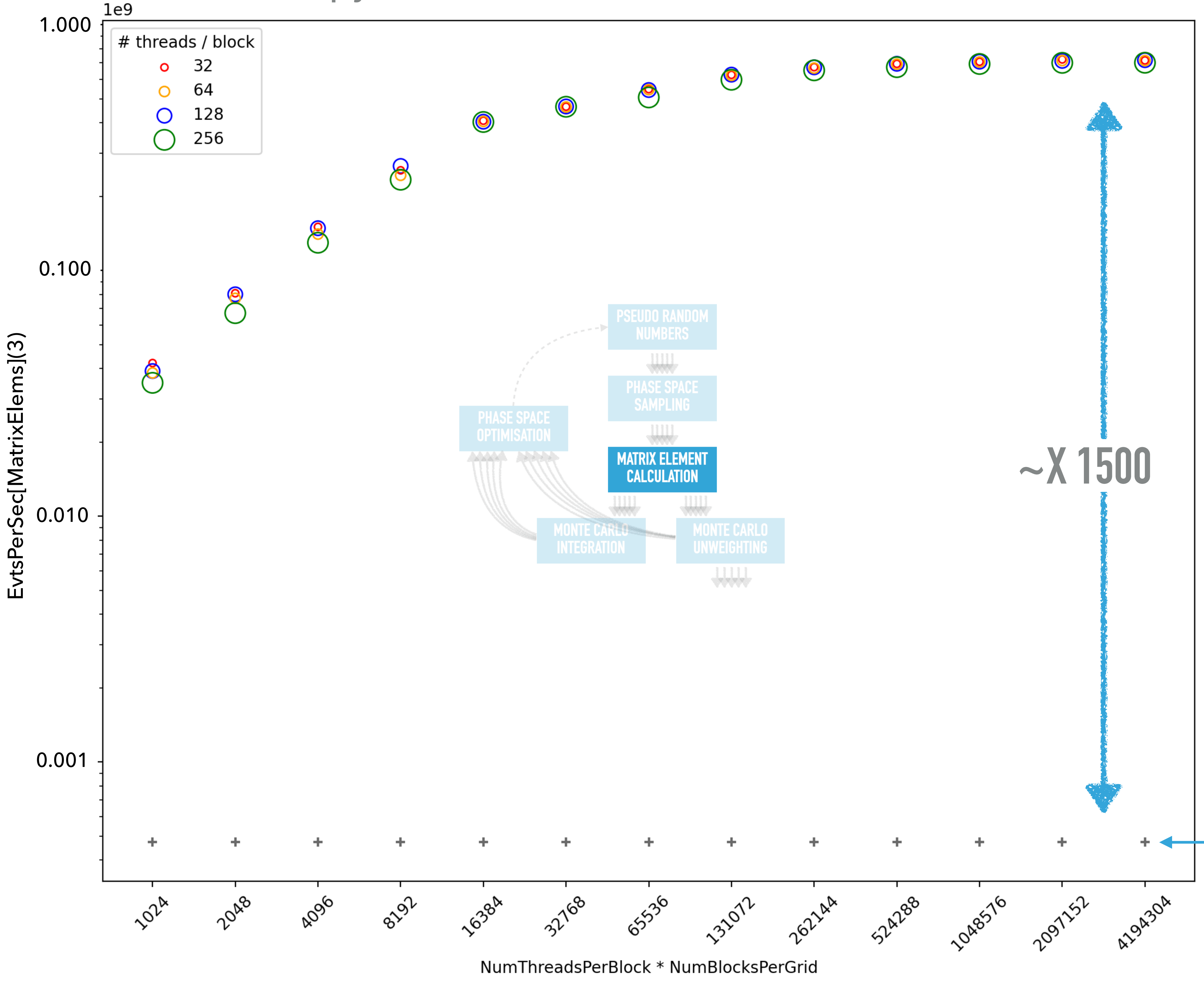
Matrix Element Calculation

Copy ME values Device –> Host

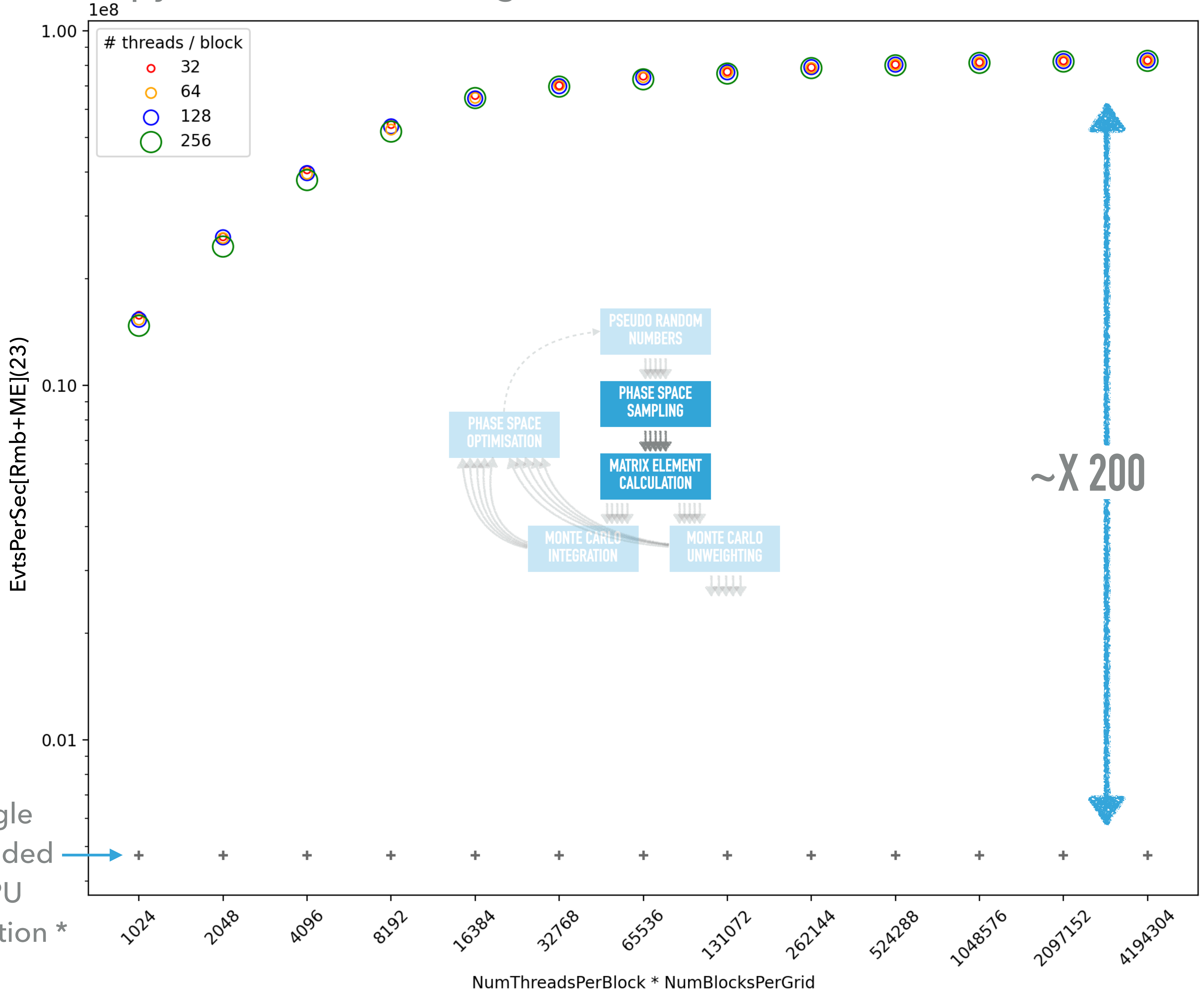
Most time is currently spent in data copy. Conclusion: $e^+e^- \to \mu^+\mu^-$ calculations are too simple

PRELIMINARY PERFORMANCE NUMBERS ($e^+e^- \rightarrow \mu^+\mu^-$)

(Matrix Element Calculations + Copy Device2Host 'ME values') / second



(Phase space sampling + Matrix Element Calulations + Copy Device2Host 'Weigths', '4-Momenta', 'ME values') / second



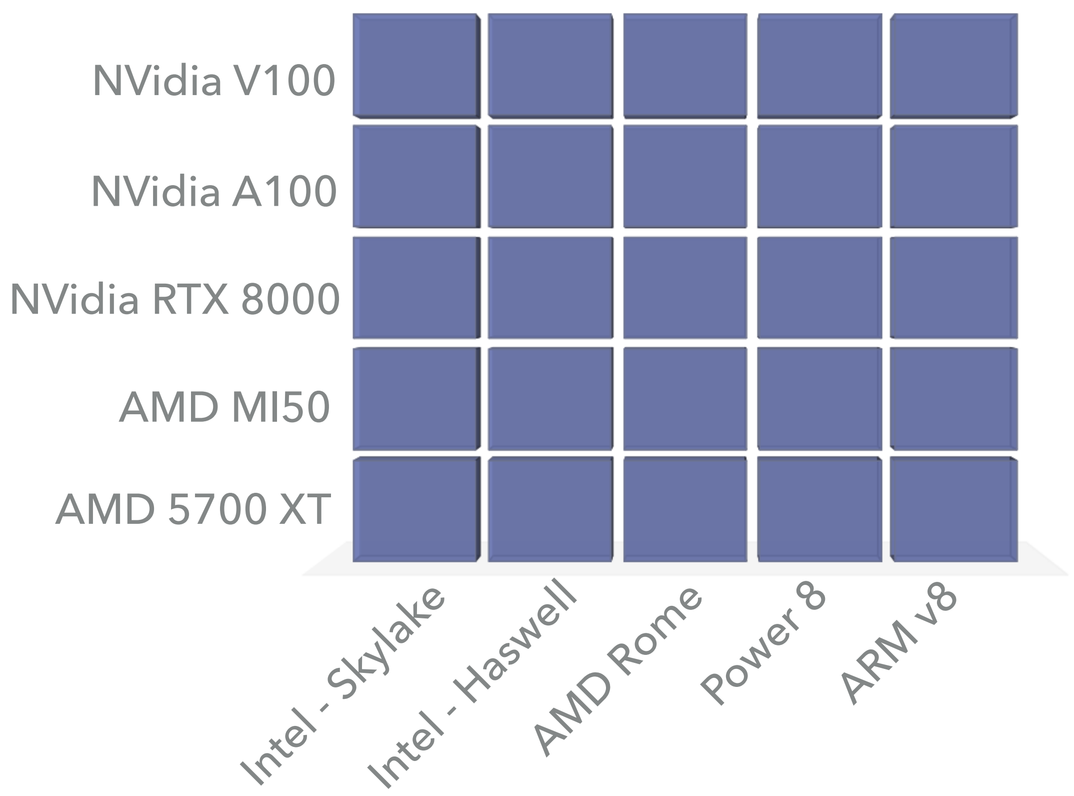
single threaded CPU execution *

GPU: GV100GL, Tesla V100S, PCIe 32 GB

CPU: Intel Xeon E5-2630 v3 @ 2.40GHz (Haswell), 32 core, 64 GB RAM

* CPU code itself has improved by ~X 1.6

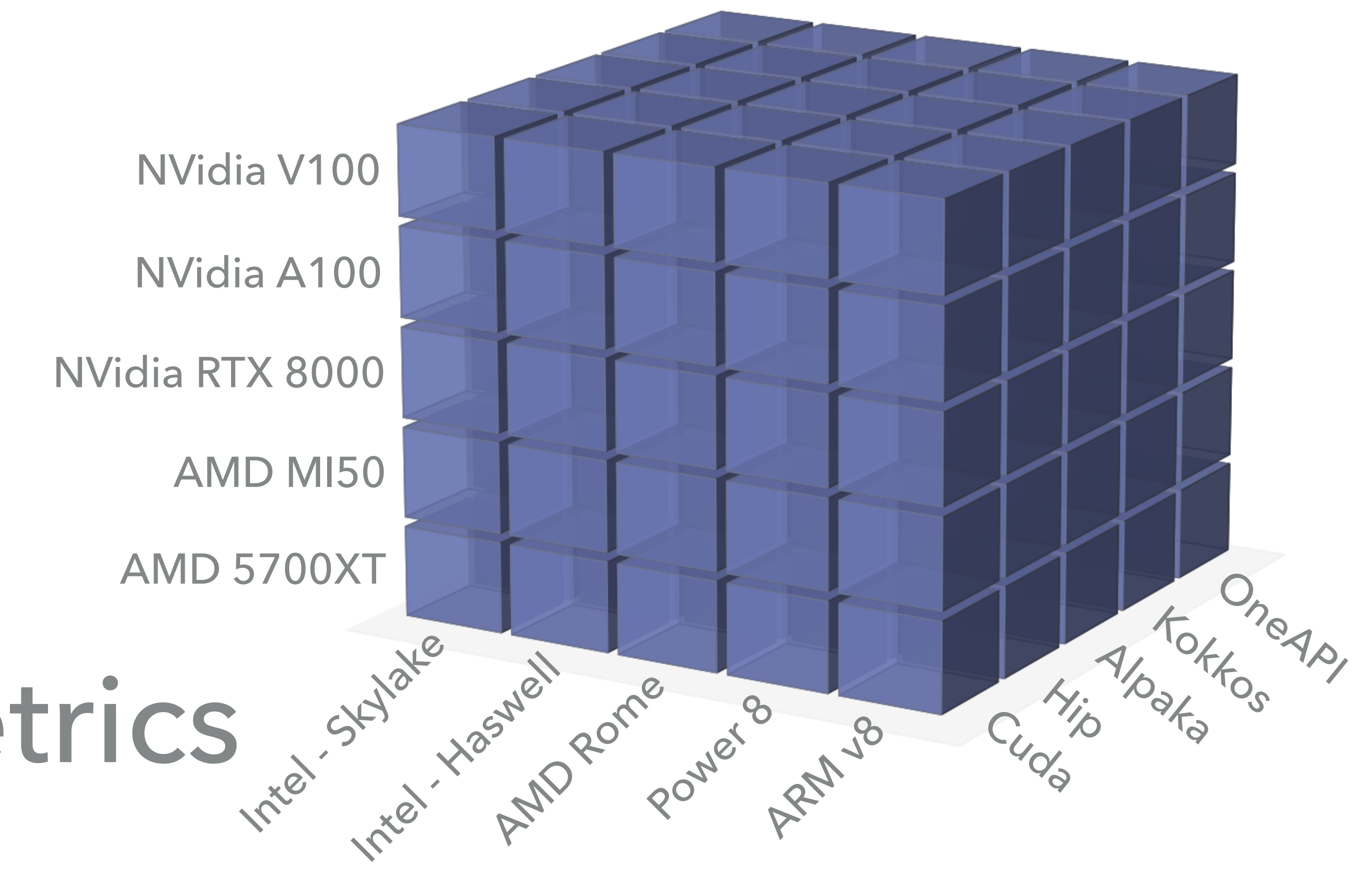
TESTING THE WORKFLOWS ON VARIOUS HARDWARE PLATFORMS

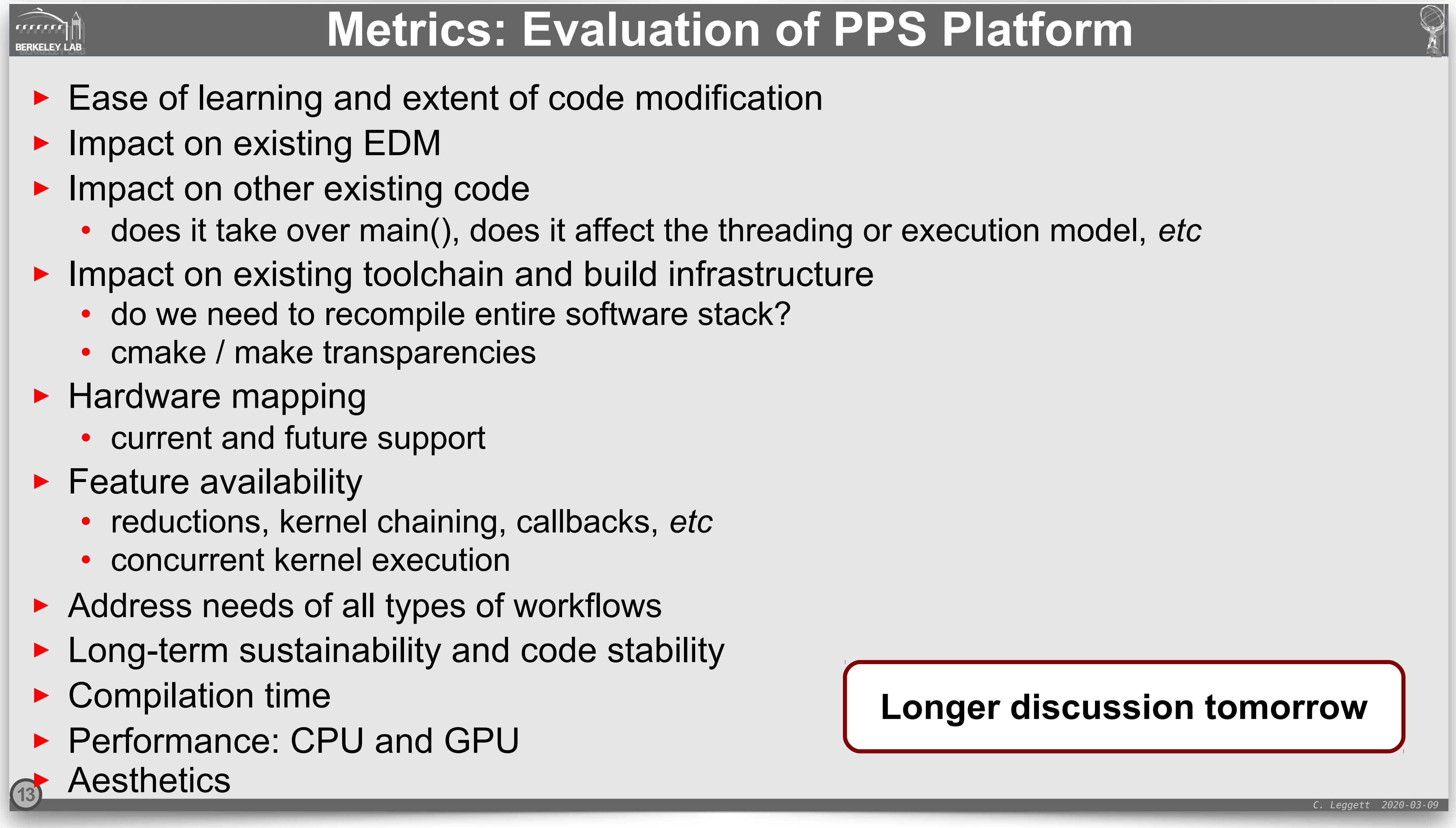


- Planning for near future to establish a software package to build the software and run performance tests on any hardware combination
  - Many combinations of CPUs/GPUs currently coming up in various HPC centers
  - Once the software builds and runs should be less work to maintain the builds
    - In summer we built successfully e.g. on Power8 but access to platform was lost
  - The WLCG benchmarking team is working on a similar goal for their GPU benchmarking suite
    - Currently discussing a collaboration of the two teams

TESTING THE WORKFLOWS ON VARIOUS HARDWARE AND SOFTWARE PLATFORMS

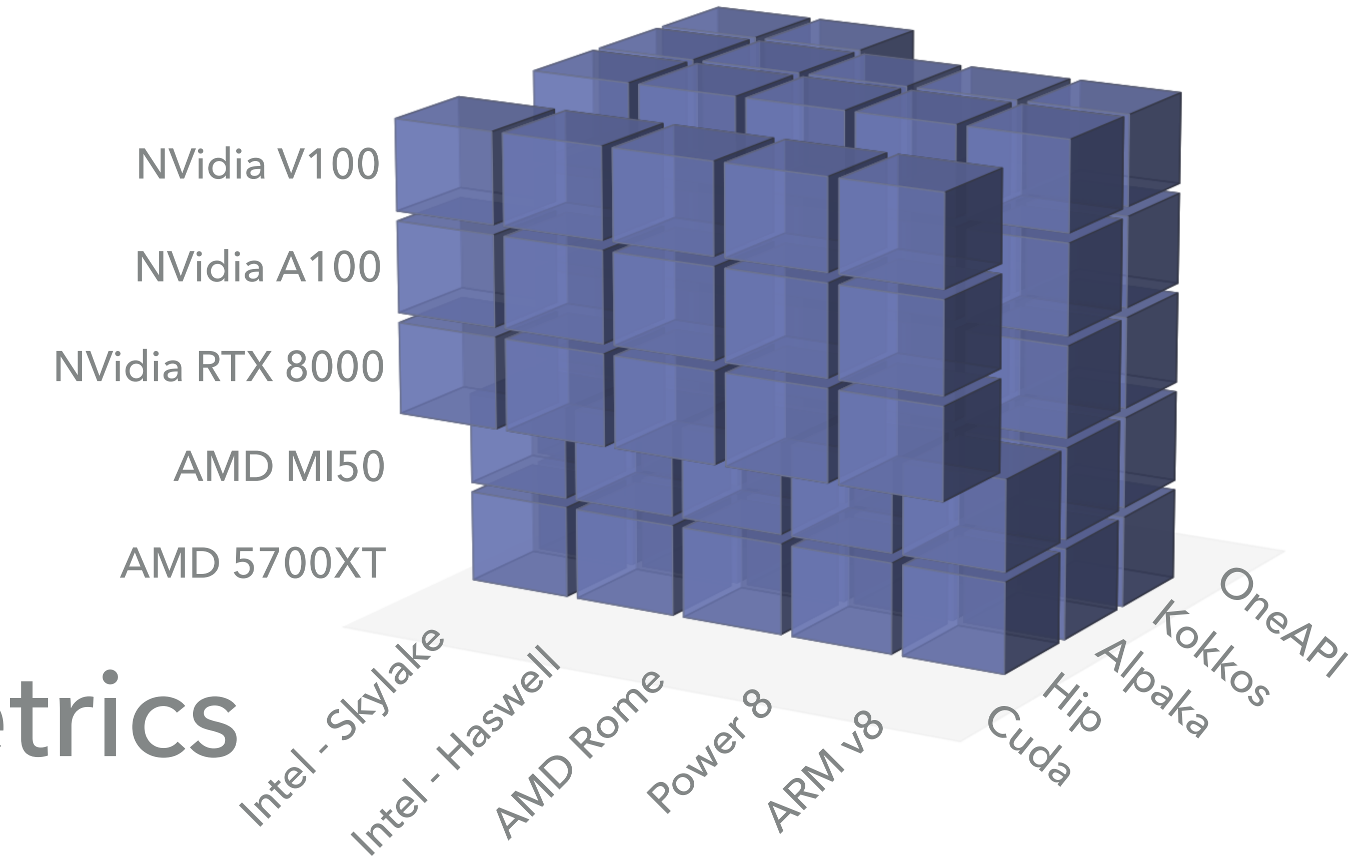
- Third dimension are implementations and abstraction layers
  - Compare those against a native implementation (e.g. Cuda)
    - In addition to performance need to also compare other metrics





Charles Leggett, CCE Kickoff, March 2020

TESTING THE WORKFLOWS ON VARIOUS HARDWARE AND SOFTWARE PLATFORMS



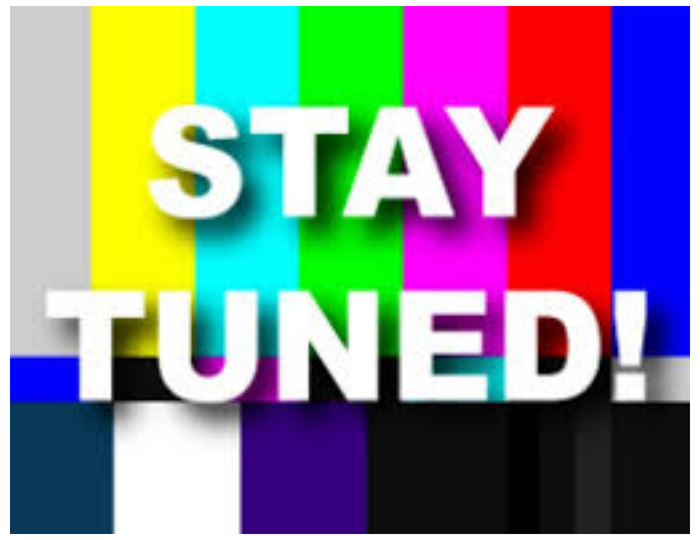
- Third dimension are implementations and abstraction layers
  - Compare those against a native implementation (e.g. Cuda)
    - In addition to performance need to also compare other metrics
- Within the team we are working on ports of (currently $e^+e^- \rightarrow \mu^+\mu^-$) to Alpaka, Kokkos, Sycl & OneAPI
  - First implementations available in the repository

EPOCH 2

(winter 2020)

EPOCH 2

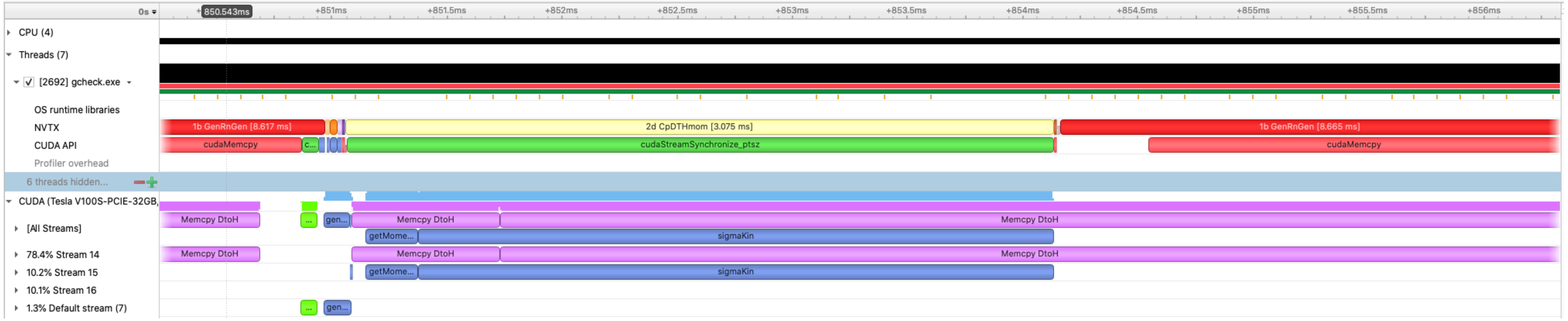
- Epoch1 engineering is currently ported upstream into the Madgraph code generator
- More news to come, …



ONGOING SOFTWARE ENGINEERING ACTIVITIES, CANDIDATES FOR EPOCH 3

PRELIMINARY

- [WIP] Cuda streams – hide latency of output data when copying back to host



- [WIP] Cuda graphs – combine kernels for faster execution
    - First promising results obtained

MORE IDEAS ON OPTIMSING THE APPLICATION

- Register pressure
  - Most calculations are in complex<double>, i.e. use 4 registers / number
  - Quickly using up available registers per streaming multiprocessor
- Move to single precision, at least for parts of the workflow
  - Current estimate for all single precision calculations is ~ x 2.4
- Balance work between host and device
- Profit from SOA/AOSOA data structures for code vectorisation also in CPU execution
- How to deal with hardware abstraction

WHAT WE ARE MISSING FOR LHC WORKFLOW EXECUTION

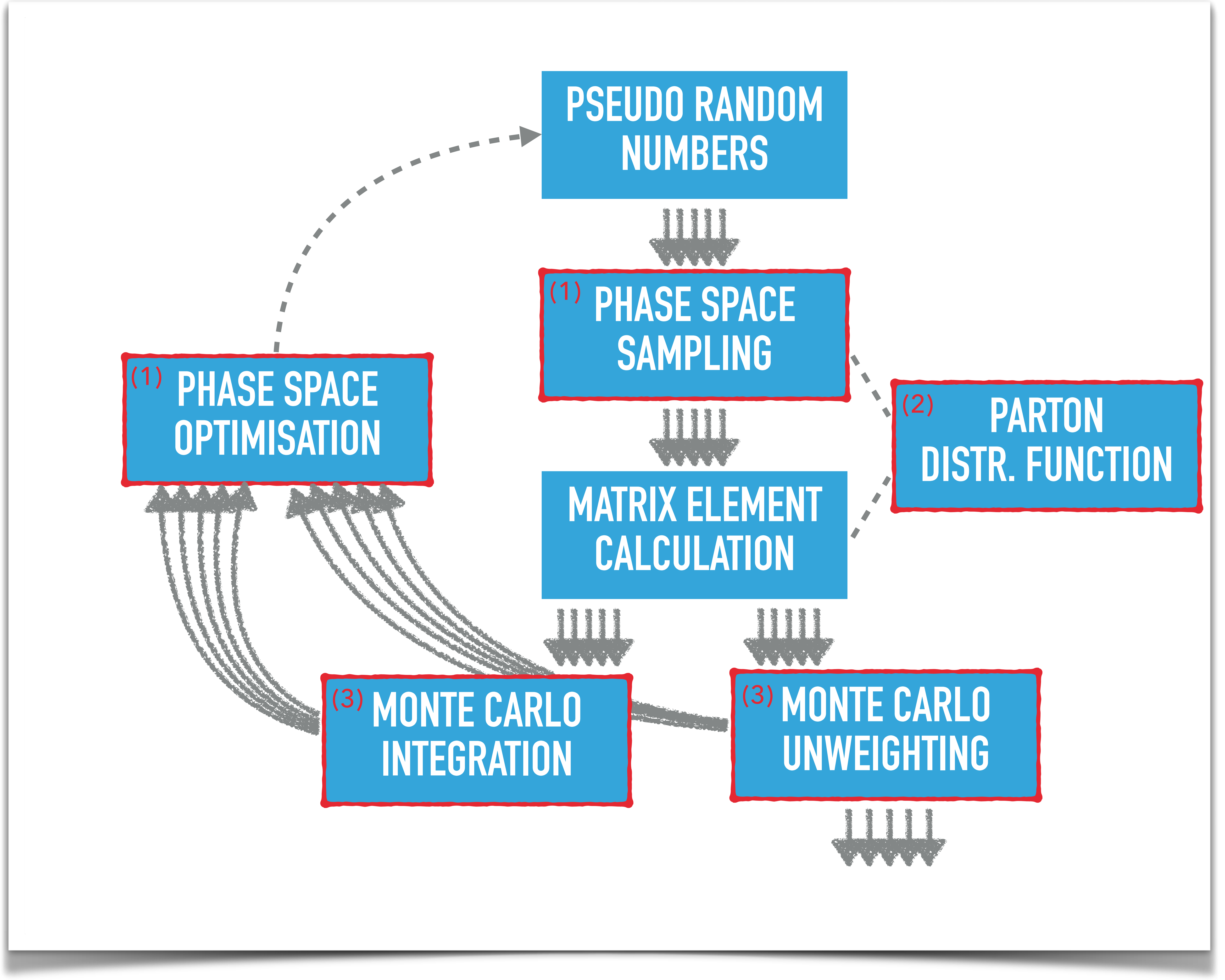
(1) Use better phase space sampler/integrator (MadEvent, gVegas, VegasFlow)

- Current sampling is too simplistic
- Work on gVegas was started

(2) Use parton distribution function (lhapdf, PDFflow)

(3) Produce cross sections and unweighted events

- Currently "physics validation" is an average of the ME values (has no physics meaning)
- Currently being worked on

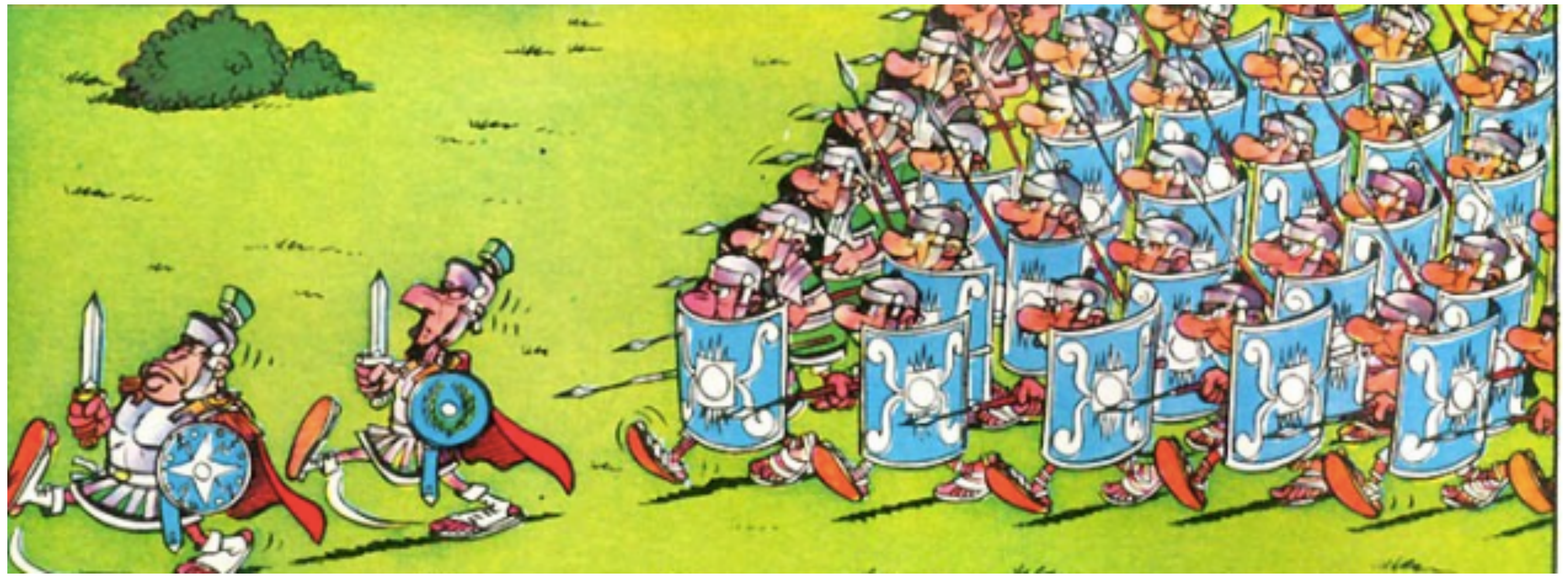


MORE THINGS WE LEARNED ON THE WAY AND OTHER THINGS THAT NEED TO BE DONE

- Attending the gpuhackathons.org Sheffield event in July was extremely helpful
- Other teams starting with GPU programming run into very similar software engineering issues
  - "Compute Accelerator Forum[1]" discusses fundamental aspects of accelerator programming
- With more collaborators joining the Madgraph effort we need to get better organised
  - moved to our own github organisation
  - reflect "epoch" structure in repo, work in forks & PRs, …
  - introduce continuous integration and test suite
  - …

[1] https://indico.cern.ch/category/12741/

SUMMARY



SUMMARY

- Started porting of individual components of Madgraph MC event generation processes onto Cuda during this year
- Currently random number generation, a simple phase space sampling and matrix element calculations have been ported
  - Speedup factors for $e^+e^- \rightarrow \mu^+\mu^-$ range from 200 to 1500 (vs single threaded CPU). Dominated by data copy due and simplicity of the underlying physics process
  - For the execution of a complete workflow, which is also representative for LHC experiment usage, more components need to be ported
- All developments are being integrated upstream in the Madgraph event generator Cuda code generation backend

MANY THANKS TO

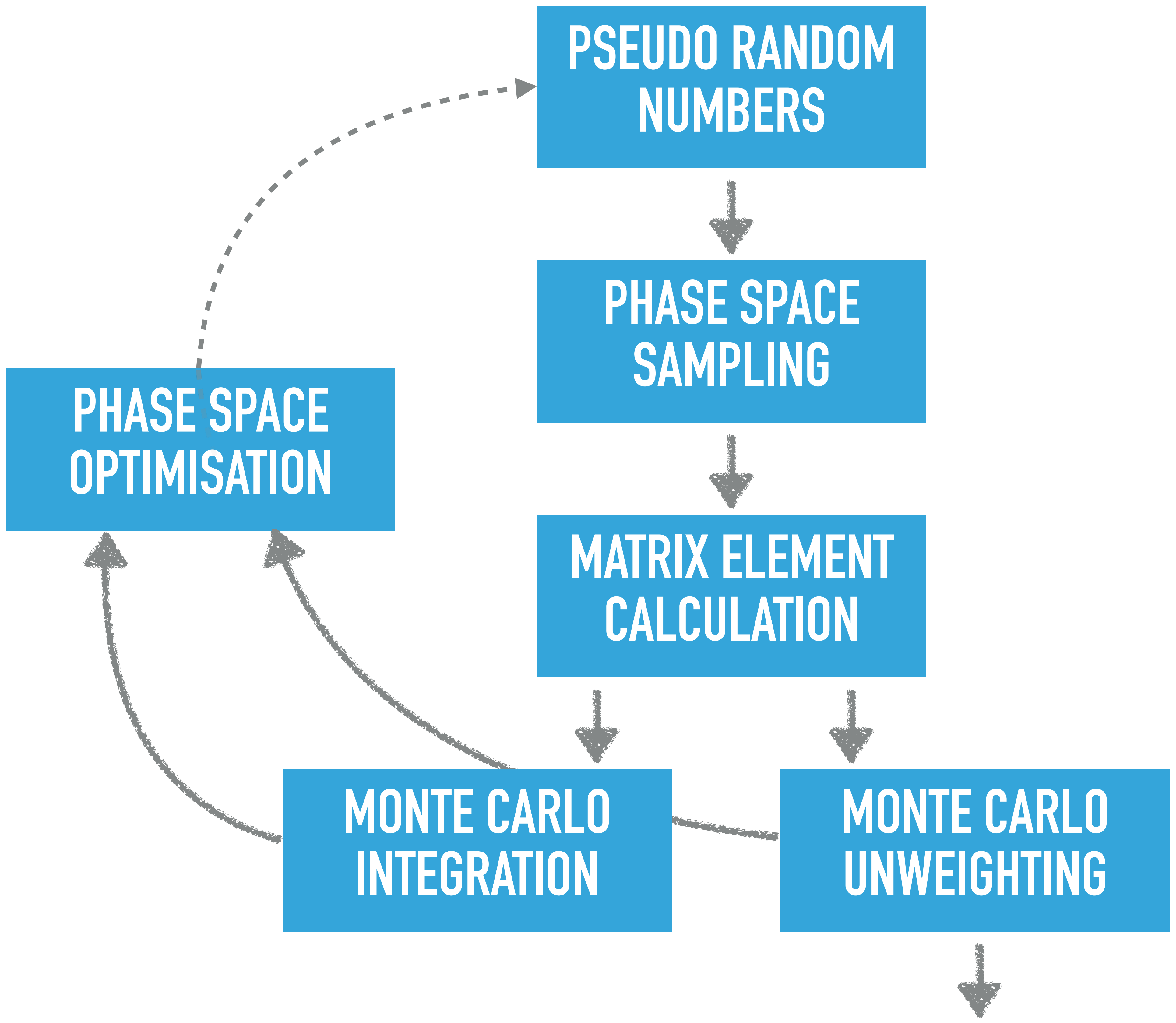
- Our mentors during the Sheffield GPU hackathon
  - Andreas Herten (Jülich), Mateusz Malenta (U Manchester), Peter Heywood (U Sheffield)
- Ricardo Rocha and CERN IT-CM
  - Providing a GPU development infrastructure via CERN/Openstack
- Domenico Giordano (CERN) and the WLCG benchmarking working group
  - Discussion and exchange of information on GPU benchmark testing
- ALICE & LHCb Online teams
  - Offering hardware platforms for system and performance testing
- Maria Girone and CERN/Openlab
  - Offering hardware and future collaboration possibilities

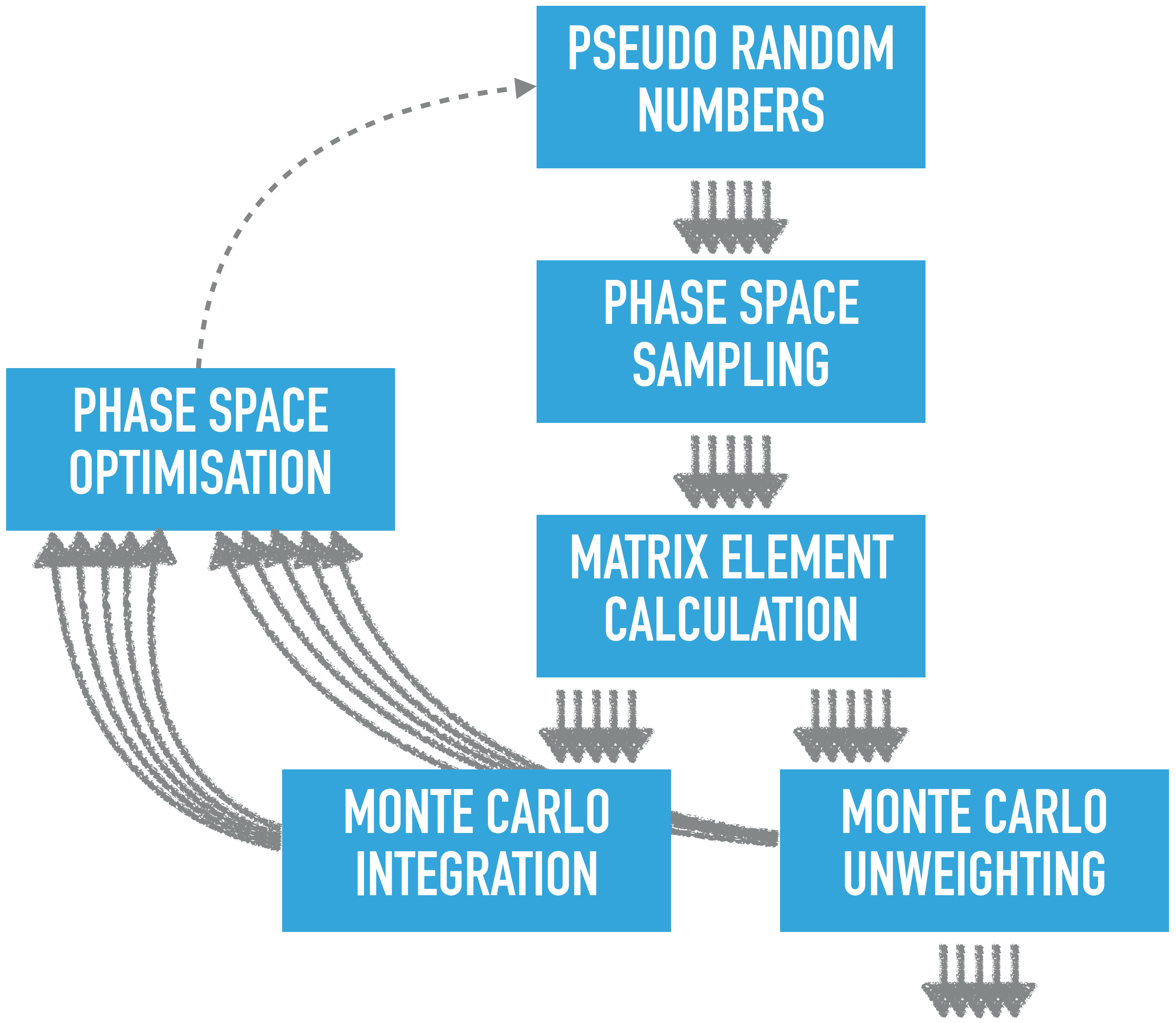
MORE DETAILS AND INFOS ON EVENT GENERATORS

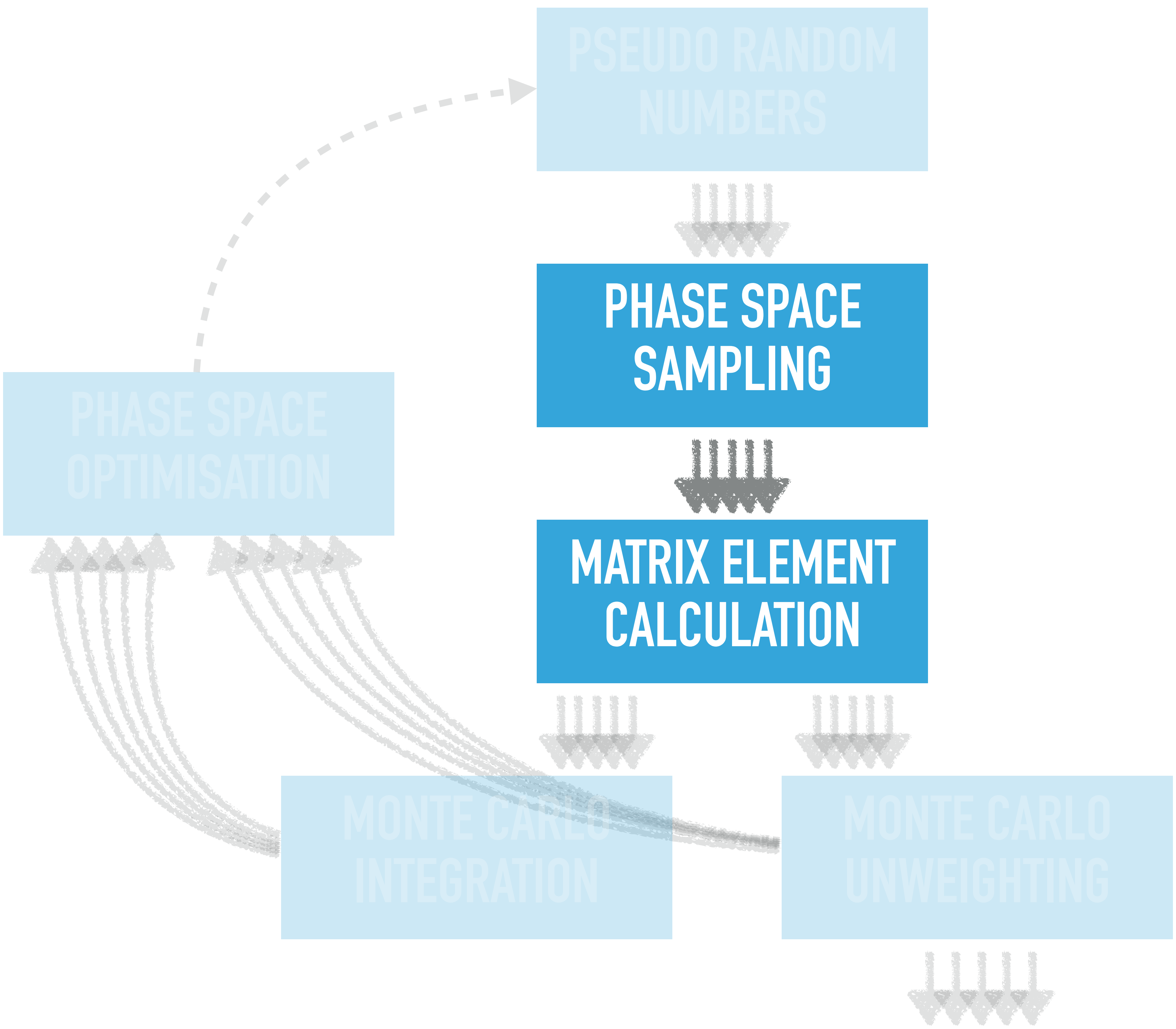
- Madgraph GPU development info
  - https://indico.cern.ch/category/12586/
  - https://github.com/madgraph5/madgraph4gpu
  - madgraph5-gpu-development@cern.ch
- HSF Monte Carlo Working Group
  - https://hepsoftwarefoundation.org/workinggroups/generators.html
- Relevant and recent papers:
  - MC generator overview paper: arXiv:2004.13687
  - HSF input to HL-LHC review: HSF-DOC-2020-01, doi:10.5281/zenodo.3779250

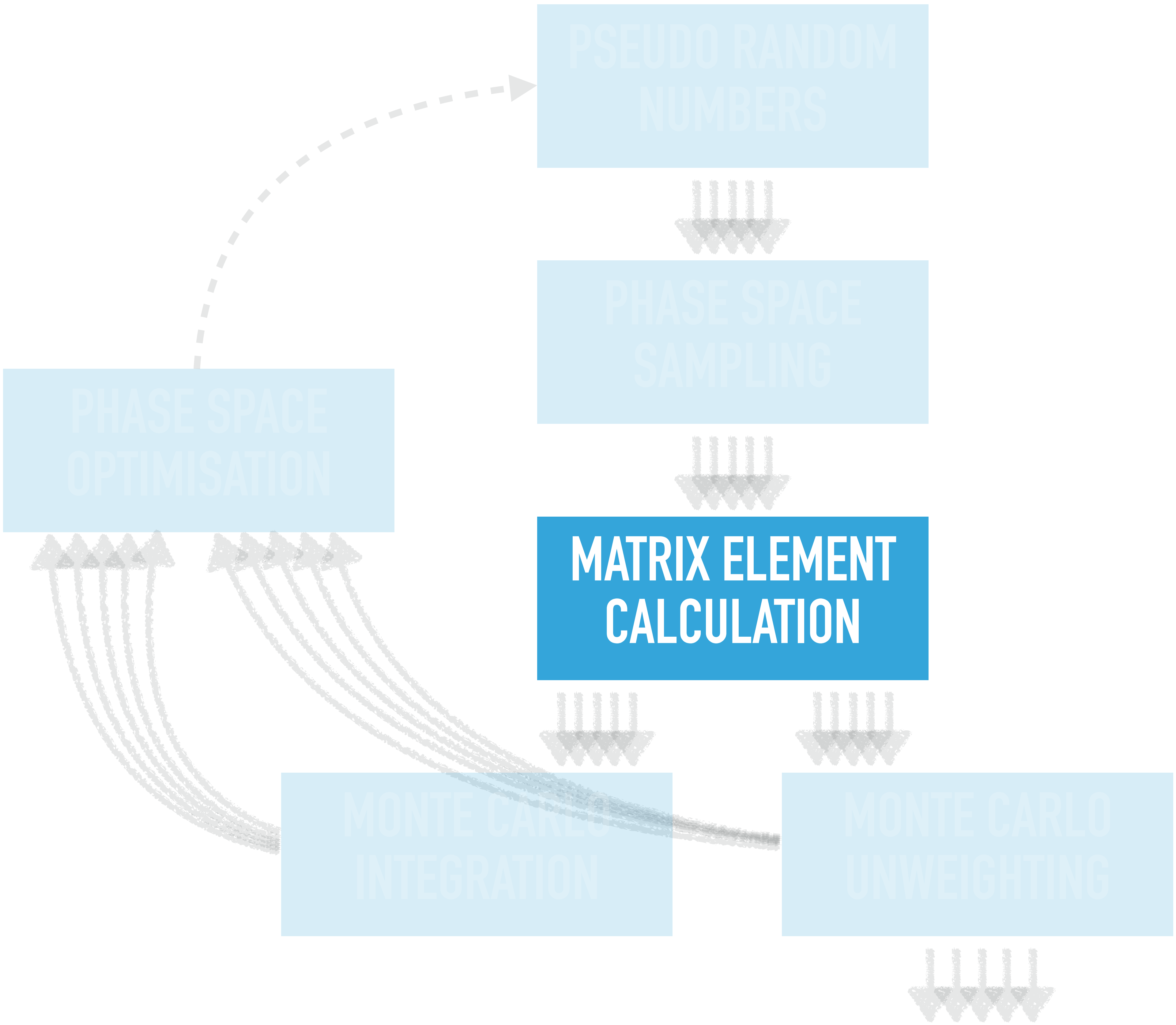
THANK YOU!

BACKUP


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

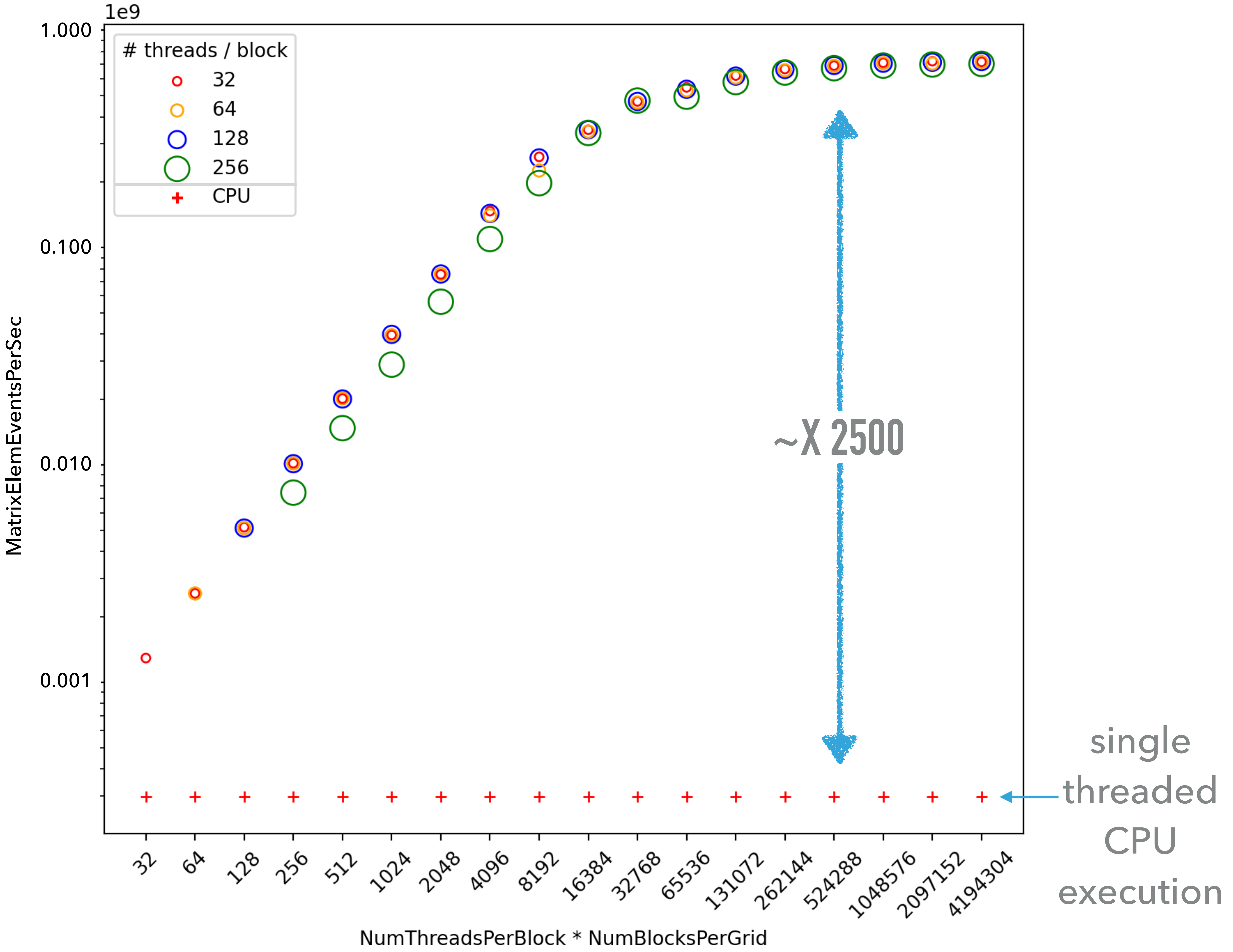

PSEUDO RANDOM NUMBERS
PHASE SPACE SAMPLING
MATRIX ELEMENT CALCULATION
PHASE SPACE OPTIMISATION
MONTE CARLO INTEGRATION
MONTE CARLO UNWEIGHTING


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

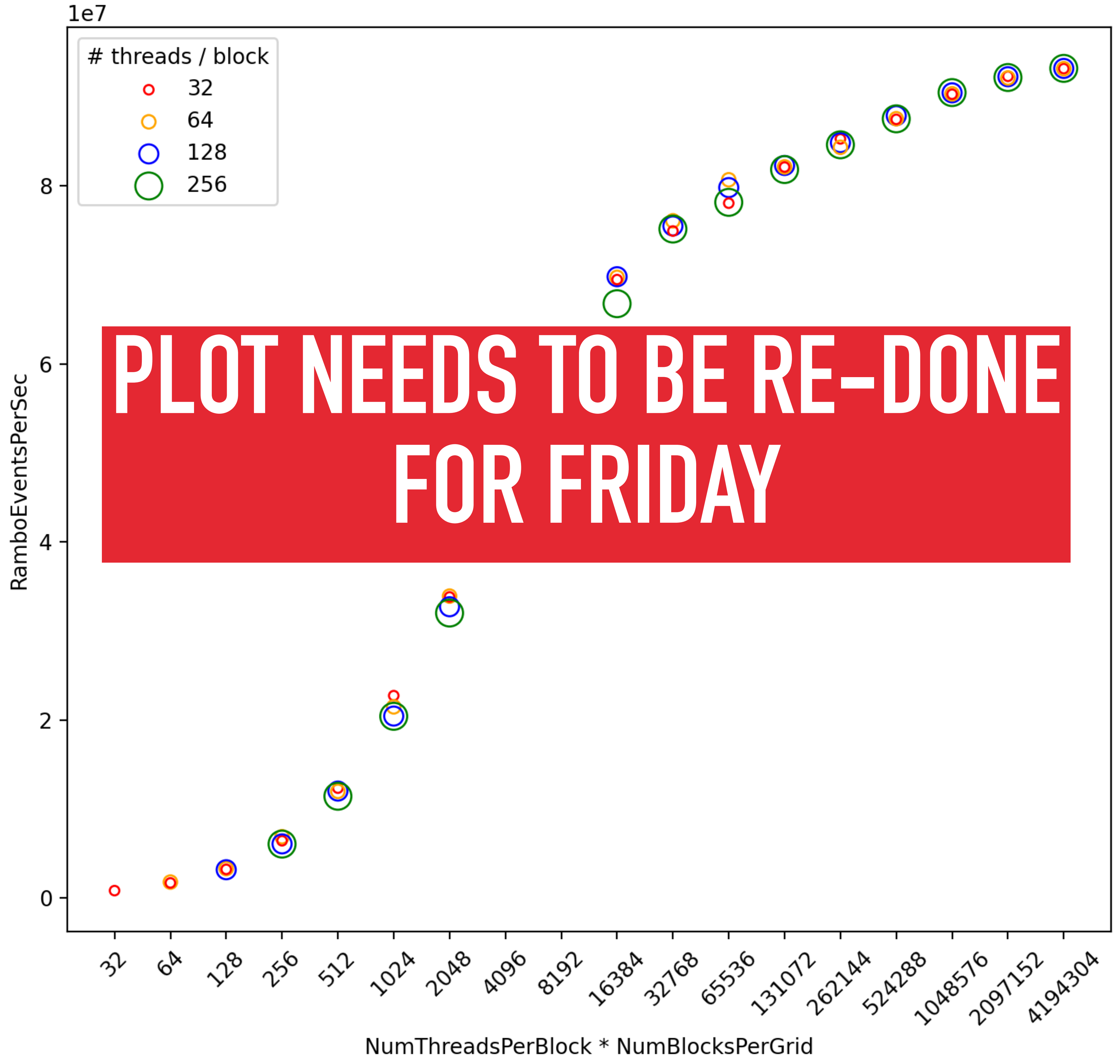

PSEUDO RANDOM NUMBERS
PHASE SPACE SAMPLING
MATRIX ELEMENT CALCULATION
PHASE SPACE OPTIMISATION
MONTE CARLO INTEGRATION
MONTE CARLO UNWEIGHTING

PRELIMINARY PERFORMANCE NUMBERS ($e^+e^- \rightarrow \mu^+\mu^-$)

Matrix Element Calculations / second



Overall workflow events / second



comparison in plot vs CPU pending, ~ x 250

(see also https://github.com/madgraph5/madgraph4gpu/issues/22)

GPU: GV100GL, Tesla V100S, PCIe 32 GB
CPU: Intel Xeon E5-2630 v3 @ 2.40GHz (Haswell), 32 core, 64 GB RAM