Università degli Studi di Milano

INFN Istituto Nazionale di Fisica Nucleare

Issues in the porting of HEP algorithms on GPUs

Alessandro Vicini
University of Milano, INFN Milano

CERN, November 14th 2023

Outline of the talk

This presentation illustrates the problems found in the study of two computational problems relevant in HEP

The topics have been considered in the framework of two undergraduate thesis at the University of Milano
(Stefano Schmidt, Isacco Beretta)

The goal was to accelerate the execution of a sequential algorithm, discussing how to restructure it for porting on a GPU

The problems have been analysed, and the studies offer useful indications for future attempts

- Introductory remarks on the GPU structure and on parallelisation of a sequential algorithm
- Reconstruction of rational functions as a tool to speed up the solution of very large linear systems
  (e.g. relevant in writing NNLO-EW amplitudes)
- Parton showering
  (LL shower already very fast, future NLL showers might benefit from a)

Basic comments about GPUs

- Commercial GPUs offer a large number of cores $O(10^3)$, with reasonable amount of RAM O(10 GB), but with a small working cache O(100 kB) per microprocessors strip

- A GPU works as a "single instruction multiple threads" processing device
  acceleration is achieved if all the logical threads executed on the physical cores do exactly the same sequence of operations

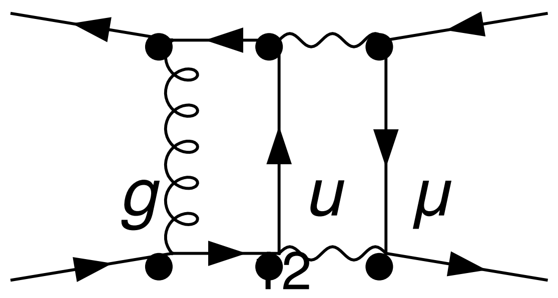
- The cache limitation sets a bound on the amount of data and code which can be processed with high efficiency

- The estimate of the speed-up factor:
  1) depends on the hardware
  2) should be done for fixed amount of power consumption (the only parameter common to servers with CPU / GPU)

Problem 1: reconstruction of a rational function, the physical background

- Scattering amplitudes are the mathematical tool used to compute the scattering probability of elementary particles

  The presence of "virtual exchanges", i.e. the presence of particles forming closed internal loops in the Feynman diagrams



  leads to a structure like

$$\mathscr{M} = \sum_{j=1}^{N} c_j(s,t,u,\ldots)\; I_j(s,t,u,\ldots)$$

  where the integrals I_j account for the impact of the closed loops,

  while the c_j are rational functions of the kinematical invariants of the process and of the particle masses

- The actual number of terms *N* in this sum can be sizeably reduced by exploiting many linear relations among the I_j

  As a drawback, the symbolic solution of the linear systems of equations expressing these relations carries along the presence of huge expressions of the determinants of these systems O(GB)

The fact that the c_j are rational functions can be exploited to determine them via a reconstruction procedure

→ the goal is to avoid the intermediate huge expressions and directly get the compact final form of the result

The solutions must be obtained in terms of rational numbers, floating point approximations are not considered

Problem 1: reconstruction of a rational function, the mathematical background (Mastrolia, Peraro, Huber)

- the numerical evaluation of the "cumbersome function f(x) " must be available
- one variable problem

$$f(x) = \frac{\sum_{r=0}^{R} p_r x^r}{\sum_{r'=0}^{R'} q_{r'} x^{r'}}$$

$$f(x) = a_0 + \cfrac{x - x_0}{a_1 + \cfrac{x - x_1}{a_2 + \cfrac{x - x_2}{\cdots + \cfrac{x - x_{N-1}}{a_N}}}}$$

Thiele representation

$$f_0(x) := f(x), \qquad f_{n+1}(x) := \left(\frac{f_n(x) - a_n}{x - x_n}\right)^{-1}$$

$$\Rightarrow a_i = f_i(x_i)$$

recursive solution
N ≧ R+R' evaluations are needed
→ intrinsically sequential

- Analogous recursive relations to reconstruct polynomials of one or several variables

- multivariate problem

$$f(\vec{z}) = \frac{\sum_{r=0}^{R} p_r(\vec{z})}{\sum_{r'=0}^{R'} q_{r'}(\vec{z})}$$

$$\vec{z} = (z_1, \ldots, z_n)$$

$$h(t, \vec{z}) = f(t\,\vec{z})$$

$$h(t, \vec{z}) = \frac{\sum_{r=0}^{R} p_r(\vec{z})\, t^r}{\sum_{r'=0}^{R'} q_{r'}(\vec{z})\, t^{r'}}$$

$p_r(\vec{z})$ and $q_{r'}(\vec{z})$ are homogeneous polynomials, for fixed $\vec{z}$ they are numerical coefficients of the polynomial in t

they can be evaluated several times, for different choices of $\vec{z}$

exploiting these determinations, $p_r(\vec{z})$ and $q_{r'}(\vec{z})$ can be reconstructed using an algorithm for the reconstruction of polynomials

Problem 1: reconstruction of a rational function, GPU implementations

- porting on a GPU

the recursive solution to reconstruct the rational function of a single variable can NOT be parallelized

but

in the realistic multivariate case,

we need a large number of evaluations of $p_r(\vec{z})$ and $q_{r'}(\vec{z})$, $\binom{n-1+R}{R}$, millions in typical cases (R~60)

→ the GPU can be effective at this point

- perform, in parallel, on the GPU all those evaluations
- assign to a single thread the whole recursive procedure to evaluate $p_r(\vec{z})$ ($q_{r'}(\vec{z})$) for a given r (r')
  we have O(R) distinct independent polynomials $p_r(\vec{z})$ and $q_{r'}(\vec{z})$ → exploit parallelism

  a substantial speed up is apparent when the number of evaluations is large

Problem 1: reconstruction of a rational function, GPU implementations

- finite fields and number representation

  - the polynomials $p_r(\vec{z})$ and $q_{r'}(\vec{z})$ have rational coefficients
  - the integers involved in the reconstruction have a huge number of digits (hundreds!)

Problem 1: reconstruction of a rational function, GPU implementations

In[562]:= `g[t_] := (4 t^6 + 3 t^2 + 1) / (4 t^4 + 1)`

In[563]:= `MonoThiele[g, 1] // StandardForm`

Out[563]//StandardForm=

$$1+\cfrac{t}{\frac{5}{3}+\cfrac{-1+t}{-\frac{34}{35}+\cfrac{-2+t}{-\frac{3955}{2269}+\cfrac{-3+t}{\frac{50\,151\,707}{1\,728\,580}+\cfrac{-4+t}{\frac{13\,366\,491\,790}{155\,634\,383\,511}+\cfrac{-5+t}{-\frac{425\,127\,287\,797\,812}{542\,269\,806\,785}+\cfrac{-6+t}{-\frac{3\,099\,279\,815\,136\,890}{326\,826\,014\,773\,710\,393}+\cfrac{-7+t}{-\frac{87\,852\,110\,901\,846\,165\,957}{136\,525\,679\,820\,040}+\cfrac{-8+t}{-\frac{217\,999\,210\,988}{22\,078\,675\,100\,625\,165}+\cfrac{-9+t}{\frac{11\,635\,824\,426\,435}{777\,142}+\frac{4\,633\,695}{4}(-10+t)}}}}}}}}}}$$

In[564]:= `MonoThieleCoeff[g, 1]`

$\{1, 0\}$

$\left\{\frac{5}{3}, 1\right\}$

$\left\{-\frac{34}{35}, 2\right\}$

$\left\{-\frac{3955}{2269}, 3\right\}$

$\left\{\frac{50\,151\,707}{1\,728\,580}, 4\right\}$

$\left\{\frac{13\,366\,491\,790}{155\,634\,383\,511}, 5\right\}$

$\left\{-\frac{425\,127\,287\,797\,812}{542\,269\,806\,785}, 6\right\}$

$\left\{-\frac{3\,099\,279\,815\,136\,890}{326\,826\,014\,773\,710\,393}, 7\right\}$

$\left\{-\frac{87\,852\,110\,901\,846\,165\,957}{136\,525\,679\,820\,040}, 8\right\}$

$\left\{-\frac{217\,999\,210\,988}{22\,078\,675\,100\,625\,165}, 9\right\}$

$\left\{\frac{11\,635\,824\,426\,435}{777\,142}, 10\right\}$

$\left\{\frac{4}{4\,633\,695}, 11\right\}$

Problem 1: reconstruction of a rational function, GPU implementations

- finite fields and number representation

- the polynomials $p_r(\vec{z})$ and $q_{r'}(\vec{z})$ have rational coefficients
- the integers involved in the reconstruction have a huge number of digits (hundreds!)
  → usage of finite fields to handle the problem
- the accuracy of the inverse map $\phi^{-1} : \mathbb{Z}_p \to \mathbb{Q}$ depends on the value of p
  → the larger p the higher the probability to end up on the correct rational number

- the usage of one single field (the lazy solution) suffers of the size of the largest available p
  → #bits > $\log_{10}(2)$ * degreemax^2 if degreemax>29 then 256 bits could be insufficient to represent an integer

- alternative solution: reconstruct over several finite fields $\{\mathbb{Z}_{p_1}, \ldots, \mathbb{Z}_{p_n}\}$ and use the chinese remainder theorem
  → the GPU can be effective also at this point, because we repeat the previous long procedure several times

Problem 1: reconstruction of a rational function, comments

the reconstruction of the rational functions multiplying the Master Integrals in 2-loop scattering amplitudes is a demanding problem
the GPU can deal with the large number of evaluations needed to sample and reconstruct the coefficients of the polynomials

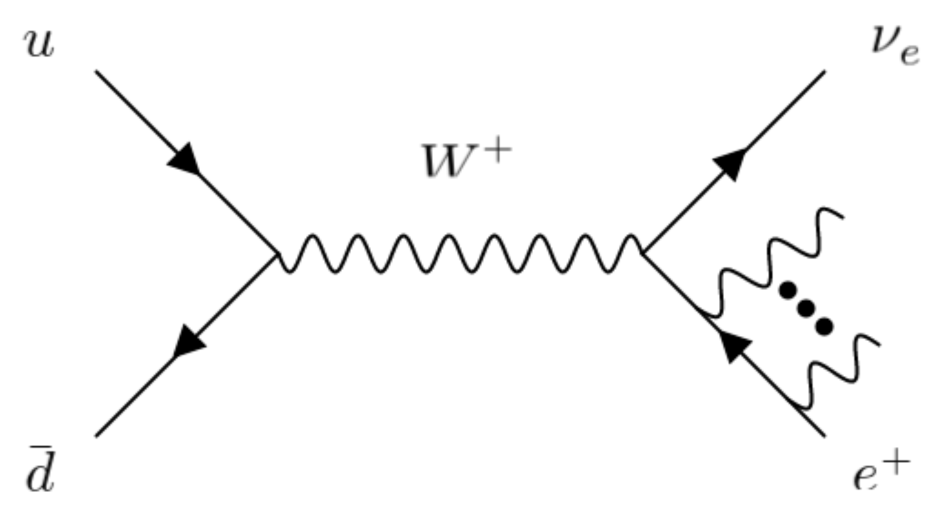
the problem has been fully solved on Mathematica and in C++ (Boost Multiprecision)

the GPU implementation found a bottleneck with one single finite field because the small set of available integers
→ in this study, the parallel version of the code was limited to polynomials of low degree → low gain

prospects:
the usage of multiple finite fields must be considered (optimisation of the algorithm flow needed)
and tested on known coefficients of NNLO QCD-EW or NNLO-EW corrections

Problem 2: parton showering, physical background

the exact calculation of the amplitude describing a final state $f + n\gamma$ is extremely difficult, for large n



soft/collinear phase space regions:

the γ emission probability is higher

the approximated energy and angular emission spectra are known

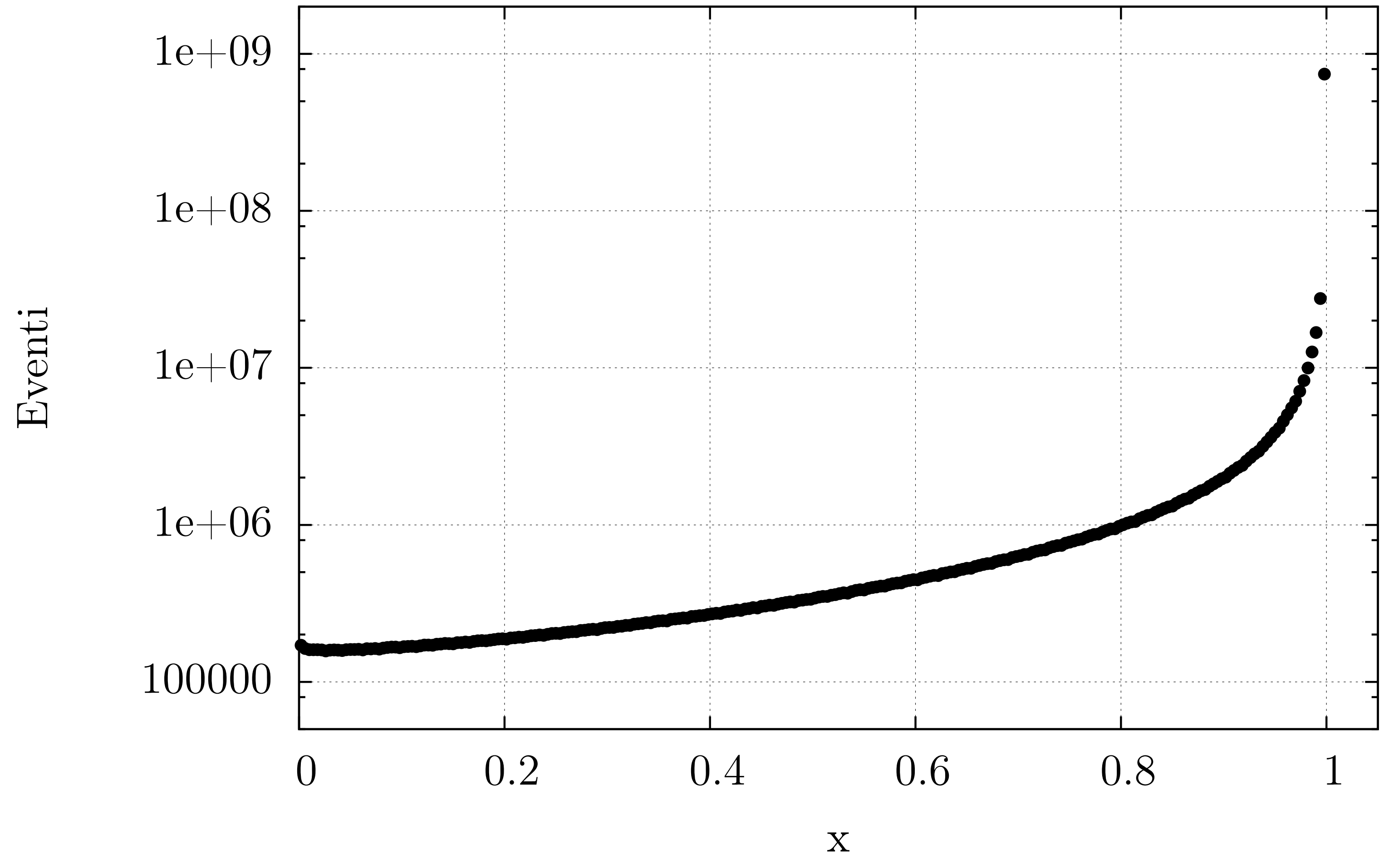
thanks to factorisation properties, each emission can be described independently of all the others,

the photon parameters depend only on the event kinematics before its emission

→ a probabilistic algorithm (a Markov chain) can be formulated to dress an event with $n\gamma$

The energy spectrum of an electron radiating a photon is given by the structure function $D(x, Q^2)$

solution of the DGLAP equations



A central element to simulate $D(x, Q^2)$ is the Sudakov form factor

$$\Pi(s_f, s_i) = \exp\left[-\frac{\alpha}{2\pi}\log\left(\frac{s_f}{s_i}\right) I_+(\varepsilon)\right]$$

which expresses the probability of an electron changing its virtuality from s_i to s_f

without emitting photons with energy larger than ε

Problem 2: parton showering, the algorithm

A HEP simulation typically has two stages:

a) the generation of N events

b) the simulation of additional radiation via Parton Shower on top of each individual event

The QED soft/collinear factorisation guarantees that the convolution of the two procedures yields events distributed according to the SM predictions

Large bunches of tree-level (LO) events can be easily generated on a GPU, simultaneously, because the squared matrix elements are rational functions of kinematical invariants and masses

The Parton Shower algorithm:

1) the Sudakov form factor allows to decide if an additional photon has to be emitted, with an hit-or-miss procedure

2a) the photon is not emitted → exit

2b) the photon is emitted → photon and electron kinematics are generated (the electron gets a virtuality)

3) the electron variables are updated and the emission of another photon is attempted again (go to step 1) unless the virtuality of the electron exceeds a max value (→ exit)

🙁 the final number of photons emitted, before the algorithm exits, can not be predicted (probabilistic process)

for each emission, the calculation of photon/electron kinematics has a variable number of steps

The structure of the Parton Shower algorithm breaks the parallelism of the GPU, because of the different number of photons emitted, event by event

Problem 2: parton showering, sequential *vs* parallel executions

A toy Monte Carlo event generator has been implemented, with CPU and GPU versions.
The simulation describes the process $u\bar{d} \to e^+\nu_e$ and adds FSR radiation. Use kinematical distributions as benchmarks.

The gain is the ratio of exec time between GPU Nvidia GeForce480 and a single core of a CPU Intel i5-3550 @ 3.3 GHz

- The first phase, the generation of tree-level events (no shower) on the GPU shows a gain factor 25 (one event per thread).

- If the complete showering of one event is assigned to a thread, then the gain factor of the whole simulation is only 3.
  this value signals that - the execution of each thread is sequential (the OS can not execute the threads of a warp simultaneously)
  - the GPU has a lower clock

The parallelism loss is due to the several branching points of the procedure:
- in the hit-or-miss step to decide about the emission → reorganising these two steps
- in the functions to generate the values of the photon kinematical values is a key point

Problem 2: parton showering, parallel generation of kinematical variables

A significant (~50%) fraction of the execution time is spent in the generation of the kinematics.

The emitted photon is described by 3 variables: $(E_\gamma, \theta_{e\gamma}, \phi)$

$E_\gamma = z\, E_e$ and z is distributed according to $P(z) = (1+z^2)/(1-z)$,

with primitive $$I_+(\varepsilon) = \int_0^{1-\varepsilon} dz\, P(z) = -\,2\log(\varepsilon) - \frac{1}{2}(1-\varepsilon)^2 - 1 + \varepsilon$$

we determine z solving $\xi = I_+(1-z)$ with a bisection routine

$\theta_{e\gamma}$ is distributed according to $$P(\theta) = \frac{1}{1-\beta\cos\theta} \qquad \beta = \sqrt{1-\left(m_e/E\right)^2}$$

we determine $\theta_{e\gamma}$ with hit-or-miss

These procedures executed from inside a Parton Shower thread are highly inefficient (parallelism breaking)

Problem 2: parton showering, parallel generation of kinematical variables

Ideally, 32 threads in a warp should be executed simultaneously

- If the hit-or-miss for the emission decision fails for some threads, they wait until the other threads finish → inefficiency
- If the routines to construct the kinematics diverge, the active threads are executed in a sequential manner → divergence

The loss of efficiency in the first hit-or-miss is unavoidable and moderate, compared to the divergence in the second step

Tentative solution

→ we generate a long array of E_γ and $\theta_{e\gamma}$ values, distributed as above (dedicated run, before the showering phase)
we fully exploit the GPU parallelism (for this phase only speed-up by a factor ~40)

→ we run the showering procedure
in the threads where an emission takes place, the photon kinematics routine picks the values from the above lists
the computation of the photon momentum components is a fixed set of equations (it preserves the parallelism)

This choice brings the overall gain from 3 to 11

Problem 2: parton showering, perspectives

Restructuring the Parton Shower algorithm requires:

- efficient separate generation of the kinematical variables (done, feasible)
    (in QED no angular ordering is needed, this trick must be modified in QCD)

- understand the possibility of a generation of samples of events with one-photon only, two-photons only, etc.
    technically feasible with good preservation of parallelism
    non trivial recombination of the events to obtain physically meaningful results

Conclusions

The GPU is effective when a specific very demanding task is isolated

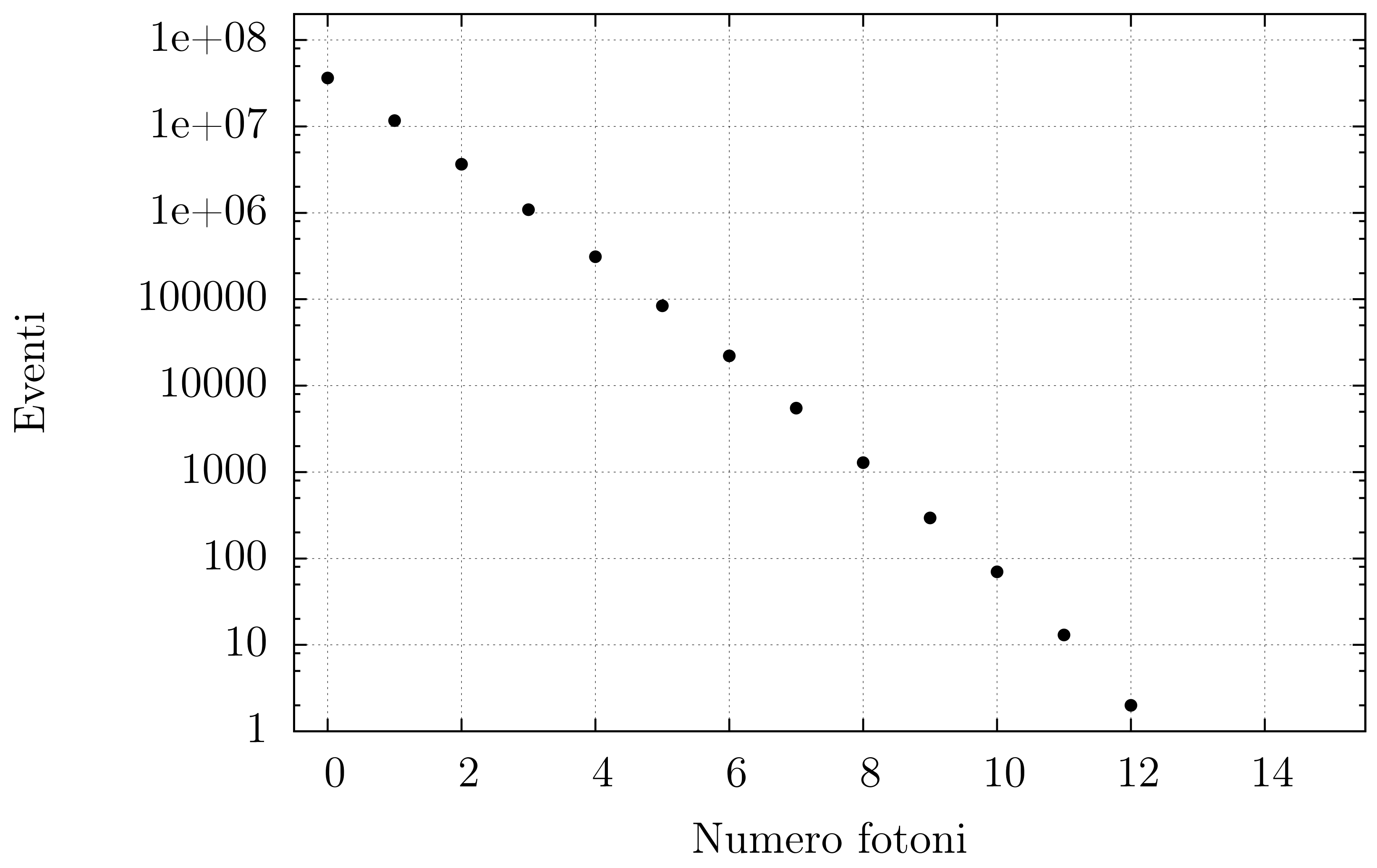
More flexible data structures (longer integers) would be beneficial
In polynomial reconstruction, GPU are potentially very relevant for fast acquisition of "readable" information about the analytical structure of the amplitude

In Parton Showers, a reorganisation of the sequence of operations has proved to be beneficial

Backup

preliminary results

photon multiplicities

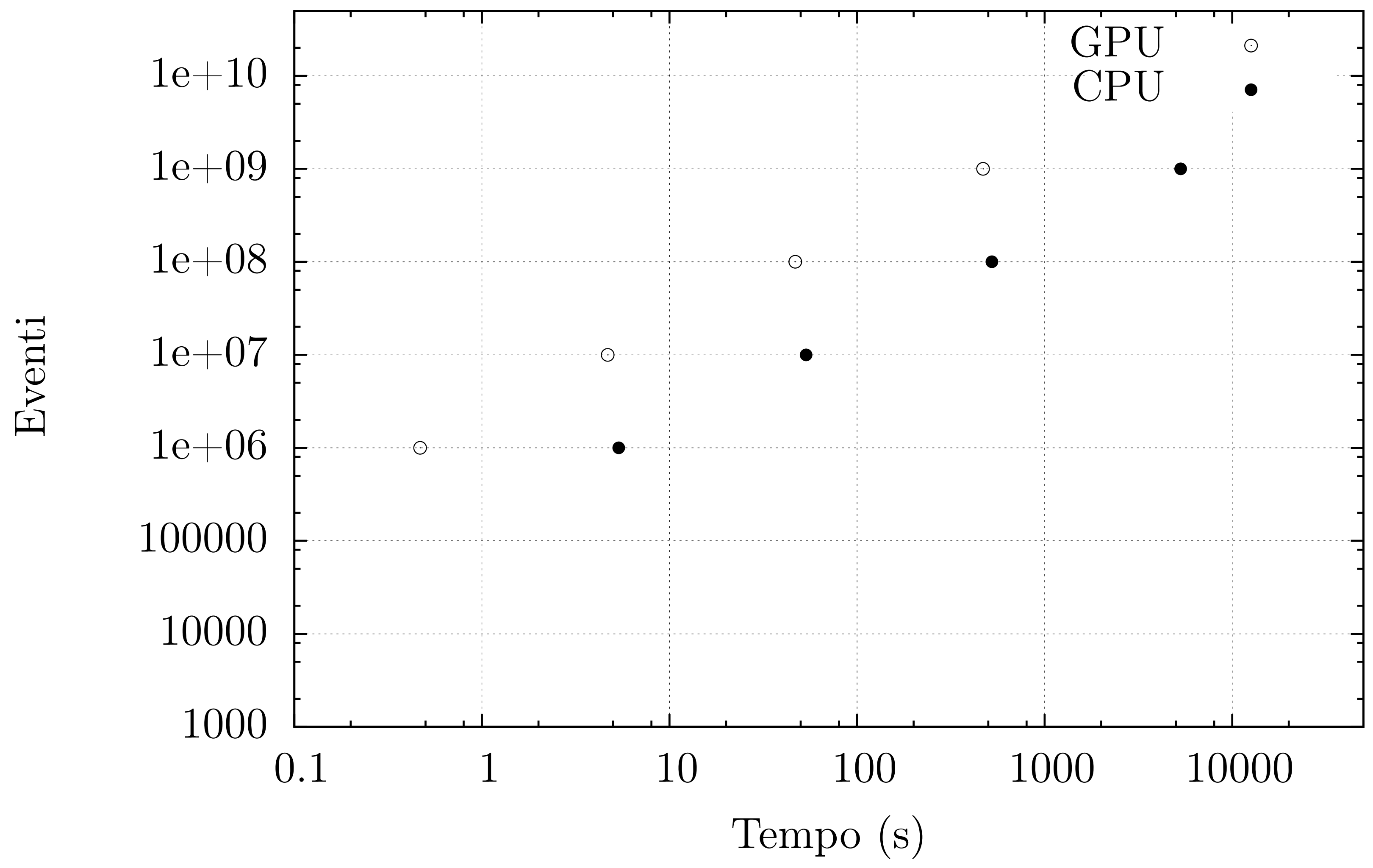


breakdown of execution time

GPU=10^7 events, CPU=10^6 events

Procedura	**Tempo di GPU (s)**		**Tempo di CPU (s)**		***Gain***
Simulazione completa	9.375	100 %	9.257	100%	11
Generazione stati finali di scattering	0.935	10 %	2.009	22%	25
Calcolo delle distribuzioni	0.218	2 %	0.046	0.5%	3
Algoritmo di parton shower	8.222	88 %	7.202	77.5%	10
Estrazione della viariabile z	0.195	2 %	1.042	11%	63
Estrazione della variabile θ	2.545	27 %	3.922	42.5%	18
Ricostruzione cinematica di un fotone	3.619	39 %	0.656	7%	2

execution time



CUDA profiling

N_THREADS \ N_BLOCKS	1	8	32	256	512	1024	2048
1	10103.20	1295.87	342.70	135.72	116.32	106.94	104.92
8	1641.48	207.55	56.03	24.07	20.63	19.19	18.73
32	547.32	69.09	18.85	8.44	7.33	6.83	6.68
256	99.98	13.00	7.31	4.79	4.68	4.64	4.62
512	70.28	9.06	6.91	5.20	5.08	5.01	5.00

the Parton Shower algorithm

1. set the initial conditions $K^2 = m^2$ and $x = 1$ for the electron virtuality and its energy fraction;

2. extract uniformly a random number ξ between 0 and 1;

3. compare ξ with the probability $\Pi(s, K^2)$:

   - if $\xi \leq \Pi(s, K^2)$, then the algorithm stops;
   - else, if $\xi > \Pi(s, K^2)$, a photon emission has occurred. Get the virtuality K'^2 at which the emission occurred by inverting the equation $\xi = \Pi(K'^2, K^2)$. Go to the next step;

4. extract the z energy fraction remaining to the electron according to the Altarelli-Parisi vertex $P(z)$, in the range between 0 and $x_+ = 1 - \varepsilon$;

5. update the virtuality and the energy fraction: substitute K'^2 to K^2 and zx to x. Go back to the step 2.