Performing PDF fits on GPU: a preliminary search for the best solution

Emilio Villa

Supervisor: Dr. Stefano Carrazza
Co-supervisor: Dr. Juan Cruz-Martinez

University of Milan

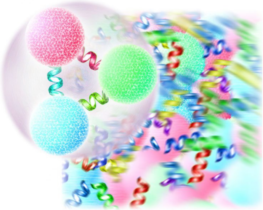
3 December 2019

Structure of the presentation

- Parton Distribution Functions (PDFs)
- Neural-Network PDF (NNPDF) collaboration
- Speeding up the fits: `APPLgrid`, `APFEL` and `APFELgrid`
- Going beyond the Fast Kernel method: convolution on GPU
- Performance analysis
- Conclusions and outlook

Parton Distribution Functions

- **QCD** is the theory of strong interactions between *quarks* and *gluons* (*partons*), the elementary constituents of *hadrons*.
- The **factorization theorem** allows to determine the hadronic cross sections in terms of a convolution between the partonic cross sections and the PDFs.
- PDFs are the probability densities of finding a parton with a certain momentum fraction x inside the parent hadron at the energy scale Q^2 of the scattering process.



Parton Distribution Functions

The expression of a typical cross section $pp \to X$ reads:

$$\sigma_{pp\to X} = \sum_s \sum_k \int dx_1 dx_2 \;\; \hat{\sigma}^{(k)(s)} \alpha_s^{k+k_{LO}} \left(Q^2\right) F^{(s)} \left(x_1, x_2, Q^2\right)$$

$F^{(s)}$ represents the partonic density of the s-th subprocess:

$$F^{(s)} \left(x_1, x_2, Q^2\right) = \sum_{i,j} C_{ij}^{(s)} f_i \left(x_1, Q^2\right) f_j \left(x_2, Q^2\right)$$

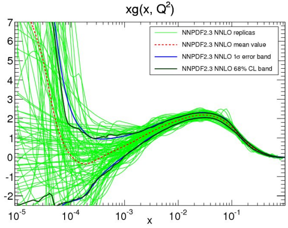
f_i and f_j are the **PDFs**

$C_{ij}^{(s)}$ counts the PDFs combinations that contribute to the s-th subprocess.

Parton Distribution Functions

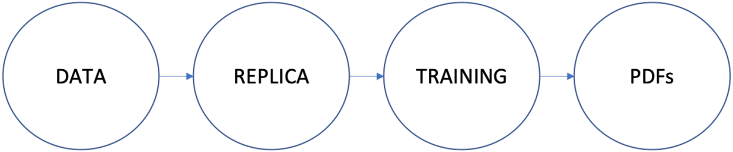
PDFs are **non perturbative** objects and, at the moment, they cannot be determined from first principles. However, it is possible to determine them by **fit** to data.

- data → hadronic cross sections;
- theory → partonic cross sections;
- fit → PDFs.



NNPDF Collaboration

Over the years, the NNPDF collaboration has developed a fitting methodology based on **Monte Carlo methods** and the usage of **neural networks** as interpolating functions.



- huge number of convolutions $\rightarrow$ **40 [h/PDF replica]**
- need to speed up the convolutions:
  - **reduction** of the number of operations required during the fit
  - increase the performances at **code level**.

APPLgrid and APFEL

Packages such as `APPLgrid` and `APFEL`, by means of interpolating techniques, allow to **lower** significantly the number of operations needed at the time of fitting.

APPLgrid

$$W^{(p)(s)}_{\alpha\beta,\tau} = \mathcal{I}_\tau\left(Q^2\right) \int dx_1 dx_2 \hat{\sigma}^{(p)(s)} \mathcal{I}_\alpha\left(x_1\right) \mathcal{I}_\beta\left(x_2\right) \quad (1)$$

$$\sigma_{pp\to X} = \sum_s \sum_p \sum_{\alpha\beta\tau} \alpha_s^{p+p_{LO}}\left(Q^2_\tau\right) W^{(p)(s)}_{\alpha\beta,\tau} F^{(s)}_{\alpha\beta,\tau} \quad (2)$$

APFEL

$$f_i\left(x_\alpha, Q^2_\tau\right) = \sum_k \sum_\beta A^\tau_{\alpha\beta,ik} f_k\left(x_\beta, Q^2_0\right) \quad (3)$$

$$F^{(s)}_{\alpha\beta,\tau} = \sum_{i,j} \sum_{k,l} \sum_{\delta,\gamma} C^{(s)}_{ij}\left[A^\tau_{\alpha\delta ik} f_k\left(x_\delta, Q^2_0\right) A^\tau_{\beta\gamma jl} f_l\left(x_\gamma, Q^2_0\right)\right] \quad (4)$$

FK table

APFELgrid simplifies further the convolution:

$$\sigma_{pp\to X} = \sum_{k,l}\sum_{\delta,\gamma} \mathrm{FK}_{kl,\delta\gamma} f_k\left(x_\delta, Q_0^2\right) f_l\left(x_\gamma, Q_0^2\right)$$

$\mathrm{FK}_{kl,\delta\gamma}$ is referred to as **Fast Kernel table**

Pros:
- number of operations lowered;
- PDFs at the initial scale;
- dot product $\rightarrow$ multi threading.

Cons:
- theory embedded in FK.

FK table

At the end, the convolution is reduced to a simple **matrix** $\times$ **matrix** product:

$$\sigma_i = \sum_{j=1}^{N} \mathrm{FK}_{ij}\mathrm{PDF}_{jn}$$

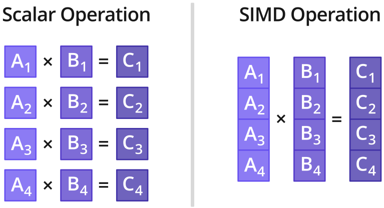
i $\rightarrow$ data, j $\rightarrow$ kl$\delta\gamma$, n $\rightarrow$ PDFs

- Ndata $\rightarrow$ data (σ at different Q^2);
- Npdf $\rightarrow$ PDFs convolved with the FK;
- N $\rightarrow$ $N_k N_l N_\delta N_\gamma$, i.e. product of active partons and x-grid points.

Next step is to **speed up** the evaluation of such product.

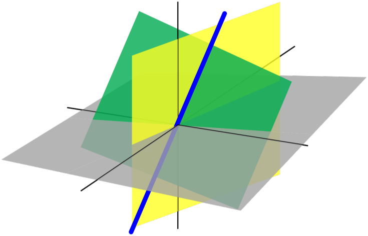
Libraries

- **SSE3** (Streaming SIMD Extensions 3);
- **AVX** (Advanced Vector Extension);
- **Eigen**;
- **OpenBLAS**;
- **GSL** (GNU Scientific Library);
- **MKL** (Math Kernel Library);
- **OpenCL** (Open Computing Language);
- **TensorFlow**.



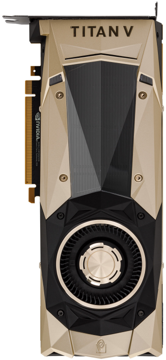
Libraries

- **SSE3** (Streaming SIMD Extensions 3);
- **AVX** (Advanced Vector Extension);
- **Eigen**;
- **OpenBLAS**
- **GSL** (GNU Scientific Library);
- **MKL** (Math Kernel Library);
- **OpenCL** (Open Computing Language);
- **TensorFlow**.



Libraries

- **SSE3** (Streaming SIMD Extensions 3);
- **AVX** (Advanced Vector Extension);
- **Eigen**;
- **OpenBLAS** (Open Basic Linear Algebra Subprograms);
- **GSL** (GNU Scientific Library);
- **MKL** (Math Kernel Library);
- **OpenCL** (Open Computing Language);
- **TensorFlow**.



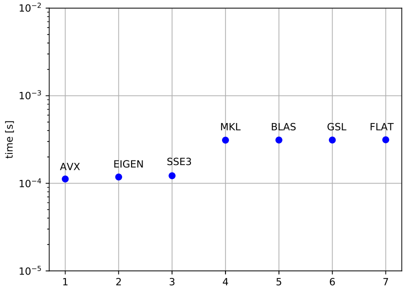
OpenCL vs CUDA

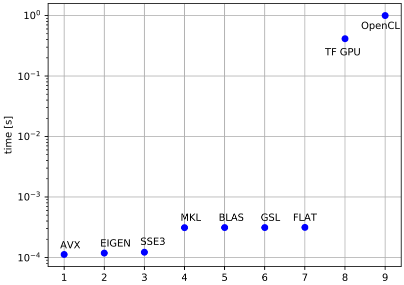
CUDA and OpenCL are very similar to each other, however:

- OpenCL can be executed over a **variety** of platforms, including CPUs, GPUs and other type of processors.
- CUDA executes **only** on NVIDIA GPUs.

First time benchmark

Convolution between an FK 8×35721 (`atlas-Z0-rapidity.root`) and a single PDF
$$(\text{Ndata} = 8\ , \quad \text{Npdf} = 1\ , \quad \text{N} = 35721)$$





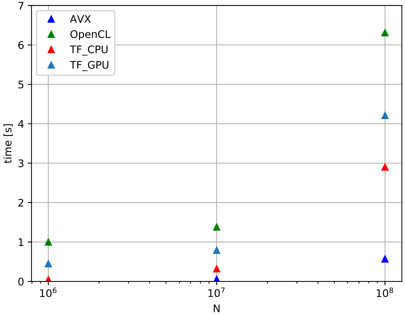
First time benchmark

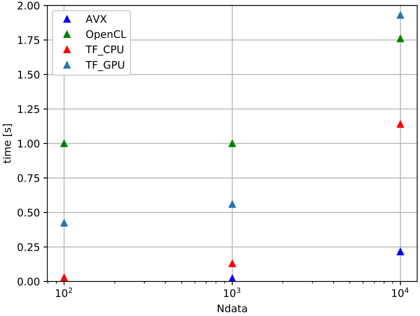
- OpenCL, TF GPU $\rightarrow$ **slowest** methods;
- need to develop a **different approach**;
- convolution evaluated for **arbitrary** dimensions of FK and PDF matrices;
- test AVX and TensorFlow on **CPU**, OpenCL and TensorFlow on **GPU**.

$$\begin{bmatrix} \mathrm{FK}_{11} & \cdots & \mathrm{FK}_{1,N} \\ \mathrm{FK}_{21} & \cdots & \mathrm{FK}_{2,N} \\ \vdots & \ddots & \vdots \\ \mathrm{FK}_{Ndata,1} & \cdots & \mathrm{FK}_{Ndata,N} \end{bmatrix} \times \begin{bmatrix} \mathrm{PDF}_{11} & \cdots & \mathrm{PDF}_{1,Npdf} \\ \mathrm{PDF}_{21} & \cdots & \mathrm{PDF}_{2,Npdf} \\ \vdots & \ddots & \vdots \\ \mathrm{PDF}_{N,1} & \cdots & \mathrm{PDF}_{N,Npdf} \end{bmatrix}$$

N, Ndata

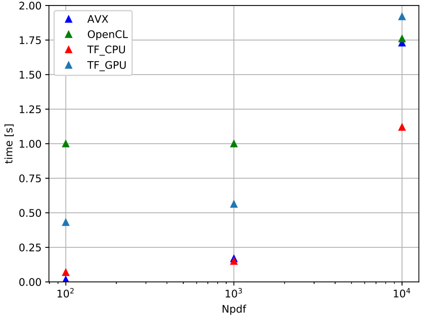
Results obtained varying respectively N and Ndata

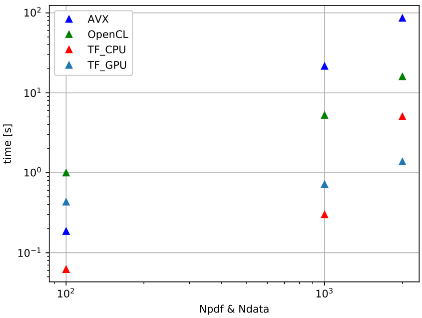




Npdf, Npdf & Ndata

Results obtained varying respectively Npdf and Npdf & Ndata





Summary

Size of the PDF	Size of the FK Table	TensorFlow CPU [s]	AVX [s]	TensorFlow GPU [s]	OpenCL [s]
35721×1	8×35721	$1.10 \cdot 10^{-2}$	$1.57 \cdot 10^{-4}$	$4.14 \cdot 10^{-1}$	~ 1
$10^6 \times 1$	8×10^6	$4.70 \cdot 10^{-2}$	$5.00 \cdot 10^{-3}$	$4.49 \cdot 10^{-1}$	~ 1
$10^7 \times 1$	8×10^7	$3.20 \cdot 10^{-1}$	$5.70 \cdot 10^{-2}$	$7.90 \cdot 10^{-1}$	1.38
$10^8 \times 1$	8×10^8	2.90	$5.70 \cdot 10^{-1}$	4.21	6.31
35721×10^2	8×35721	$6.90 \cdot 10^{-2}$	$1.60 \cdot 10^{-2}$	$4.31 \cdot 10^{-1}$	~ 1
35721×10^3	8×35721	$1.50 \cdot 10^{-1}$	$1.69 \cdot 10^{-1}$	$5.63 \cdot 10^{-1}$	~ 1
35721×10^4	8×35721	1.12	1.73	1.92	1.76
35721×1	$10^2 \times 35721$	$2.80 \cdot 10^{-2}$	$2.43 \cdot 10^{-3}$	$4.24 \cdot 10^{-1}$	~ 1
35721×1	$10^3 \times 35721$	$1.30 \cdot 10^{-1}$	$2.14 \cdot 10^{-2}$	$5.60 \cdot 10^{-1}$	~ 1
35721×1	$10^4 \times 35721$	1.14	$2.16 \cdot 10^{-1}$	1.93	1.76
35721×10^2	$10^2 \times 35721$	$6.20 \cdot 10^{-2}$	$1.86 \cdot 10^{-1}$	$4.32 \cdot 10^{-1}$	~ 1
35721×10^3	$10^3 \times 35721$	$3.00 \cdot 10^{-1}$	21.61	$7.19 \cdot 10^{-1}$	5.25
$35721 \times 2 \cdot 10^3$	$2 \cdot 10^3 \times 35721$	5.06	86.13	1.38	15.97

Conclusions and outlook

Conclusions[1]:

- PDF fits can benefit from **hardware accelerators** (i.e. GPUs and OpenCL compatible devices) thanks to the possibility of offloading the most time-consuming tasks to the accelerator;
- However, in order to achieve performance improvements some precautions are required by defining the sizes of FK tables and the number of PDFs that should be convoluted simultaneously.

Outlook:

- Solutions proposed in this work should be tested in a real PDF fit. This will be possible thanks to the future extension of the `n3fit` framework to support GPU hardware.

[1] S. Carrazza, J. Cruz-Martinez, J. Elizari, E. Villa, *Towards hardware acceleration for parton densities estimation*, Proceeding of PHOTON 2019, preprint arXiv:1909.10547

CPU and GPU specs

Intel(R) Core(TM) i9-9980XE CPU 3.00GHz:

- # of cores: 18
- processor base frequency: 3.00 GHz

GPU Nvidia Titan V:

- # of streaming multiprocessors: 80
- # of CUDA cores: 5120
- base clock: 1.2 GHz
- total memory: 12288 MB
- memory clock: 850 MHz
- total memory bandwidth: 652.8 GB/s