GooFit - Fitting mit CUDA/OpenMP

- GooFit ist ein ToolKit für parallelisierte Maximum Likelihood Fits mit CUDA (OpenMP) in einem rooFit-artigen Aufbau. https://goofit.github.io/
- Die thrust Bibliothek, die parallelisierte Algorithmen zur Verfügung stellt und auf der STL aufbaut wird hier als interface von GooFit zu den GPUs und multikern CPUs verwendet.
- Für komplexe Datenanpassungen verringert sich die Prozesszeit durch die Verwendung von GPUs um Faktoren 200 bis 300.
  https://arxiv.org/pdf/1311.1753.pdf
- Installation
  - GooFit verwendet das lokal installierte ROOT → ROOT environment setzen
  - Installation von GooFit soll im directory `/cern/gooFit_build` als user root mit Hilfe von `cmake` durchgeführt werden:

```
xi0:#  cd /cern; mkdir  GooFit_build;
xi0:#  git clone -recursive https://github.com/GooFit/GooFit.git
xi0:#  cd GooFit_build
xi0:#  cmake ../GooFit -DGOOFIT_EXAMPLES=ON -DGOOFIT_TESTS=ON  /
       -DGOOFIT_PYTHON=ON  -DGOOFIT_DEVICE=CUDA                /
       -DGOOFIT_PACKAGES=ON -DGOOFIT_MAXPAR=1800
xi0:#  make -jN ; make install                  N ist die Anzahl der CPU cores
xi0:#  cd examples/simpleFit/; ./simpleFit      Fit von 3 PDFs
xi0:#  nvprof ./simpleFit                      profiling infos
```

thrust in CUDA

- CUDA
  - Vortrag E. Volkmann GPU computing mit CUDA
    Beispiele: intro.cu und closest_point.cu
  - Vorlesungsmaterial "Linux als Arbeitsplattform im wissenschaftlichen Umfeld"
- Mit der thrust Bibliothek sind die low level Operationen für Berechnungen mit einer GPU durch ein high-level C++ Interface verborgen. thrust ist vollständig mit CUDA-c benutzbar und als auf der STL basierenden template Bibliothek implementiert.
  - Vortrag S. Sontag Standard Template Library
- Die parallelen Programme/Algorithmen können aus Basiskomponenten mit geringem Aufwand codiert und mit `nvcc` kompiliert werden. thrust sucht die beste Implementierung für das Prozessing auf der GPU automatisch.
  - 2 vector container: host_vector und device_vector
  - parallele Algorithmen, z.B. reduce, transform, copy, plus, multiplies, fill, sequence, replace, sort, …..
  - Iteratoren und Device pointer

Einführungen finden sich hier: https://docs.nvidia.com/cuda/thrust/index.html und https://github.com/thrust/thrust/wiki/Quick-Start-Guide

thrust in CUDA

- thrust Beispiele
  - host_vector und device_vector, algorithms

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <algorithm>
#include <thrust/generate.h>
#include <thrust/sort.h>
#include <thrust/copy.h>
…...
// generates 1 048 576 random numbers
thrust::host_vector<int> h_vec(32 << 20);
std::generate(h_vec.begin(), h_vec.end(), rand);
// transfer data to the device
thrust::device_vector<int> d_vec = h_vec;
// sort data on the GPU and copy back
thrust::sort(d_vec.begin(), d_vec.end());
thrust::copy(d_vec.begin(), d_vec.end(),h_vec.begin());
…...
```

thrust_example.cu

  - algorithm

```
#include <thrust/transform.h>
// compute Y = -X
thrust::transform(X.begin(),X.end(),Y.begin(),thrust::negate<int>());
```

thrust in CUDA

- thrust Beispiele
  - algorithm

```
#include <thrust/count.h>
// count the 1s in the vector vec
int result = thrust::count(vec.begin(), vec.end(), 1);
// build the sum of array D using reduce
int sum=thrust::reduce(D.begin(),D.end(),(int)0, thrust::plus<int>());
// print
thrust::copy(Y.begin(),Y.end(),std::ostream_iterator<int>(std::cout, "\n"));
…...
```

thrust_sum.cu

  - algorithm Functor Beispiel

thrust_norm.cu

Beispiel für die Verwendung der CUDA thrust Bibliothek

Unsere Laptops haben eine CUDA Installation mit der die NVIDIA Graphikkarte für paralleles Datenprozessing benutzt werden kann.
Implementieren Sie unser CUDA saxpy.cu Beispiel mit Hilfe der thrust Bibliothek.

Arbeitsvorschlag:

- Stellen Sie das Sie `nvcc` aufrufen können, passen Sie gegebenenfalls die PATH variable des student users an.

- Kopieren Sie `x` und `y` in 2 device_vector Instanzen `X` und `Y`

- Schreiben Sie einen Functor, der die saxpy Operation Y <- A * X + Y in `thrust::transform` ausführt.

- Kopieren Sie den `device_vector` in einen `host_vector` und führen Sie den Test des Resultates wie in `saxpy.cu` durch.

- Verwenden Sie `nvprof` zum Vergleich beider Programme.

mysaxpy.cu