Statistical significance estimation of a signal within the *GooFit* framework on GPUs

Leonardo Cristella
on behalf of the CMS Collaboration

UNIVERSITA' DEGLI STUDI DI BARI "ALDO MORO" & I.N.F.N. SEZIONE DI BARI

29 Aug - 3 Sep 2016 XII Quark Confinement and the Hadron Spectrum Thessaloniki

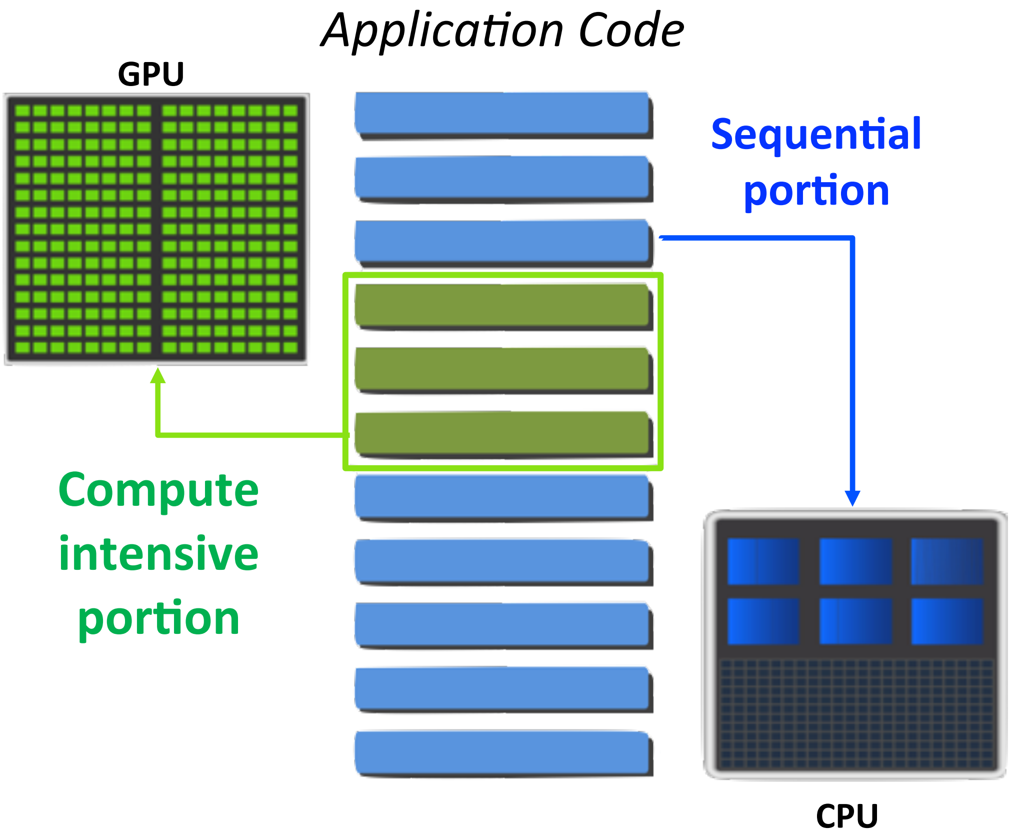
Outline

- Introduction to *GPU computing* & *GooFit*
- Pseudo-experiments for p-value estimation: *GooFit* vs *RooFit* performance study
- Exploring the applicability limits of Wilks theorem
- Summary & Outlook

Introduction: *GPU* computing & *GooFit*

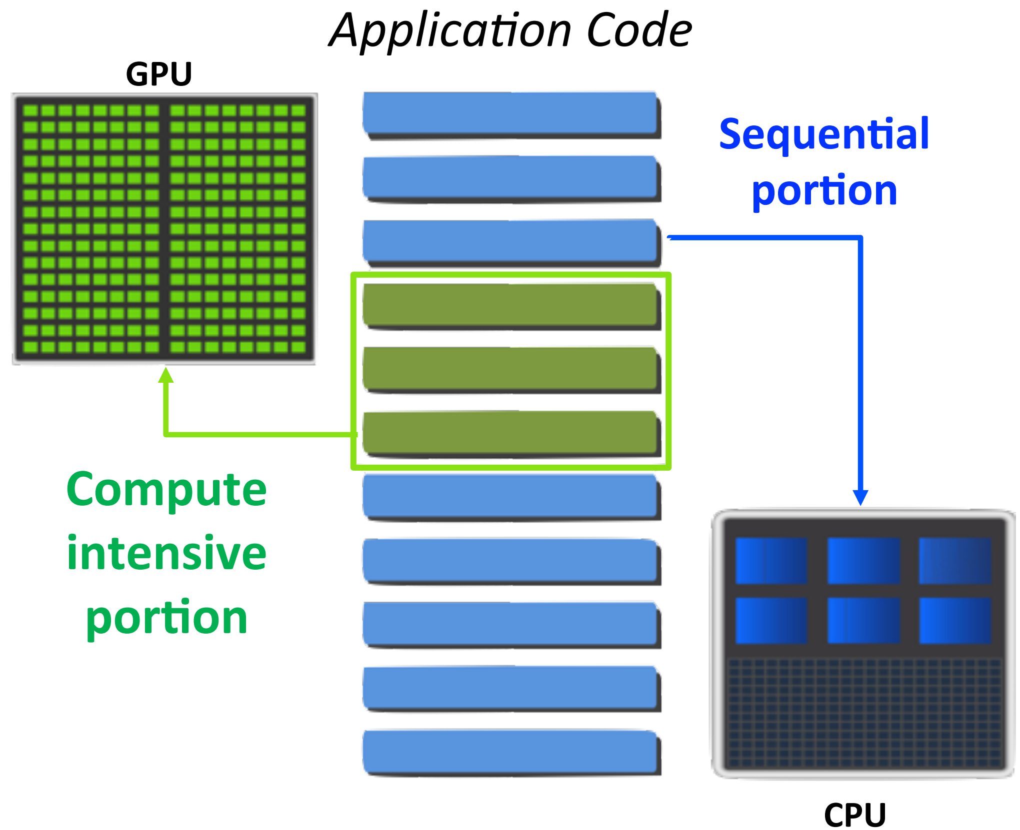
- **Hetherogeneous GPU-acccelerated computing** is the use of a **G**raphics **P**rocessing **U**nit **to accelerate scientific applications** (among other apps).

Enhancement of application performance obtained by offloading compute-intensive portions to the GPU (*the device*) while the remainder of the code still runs on the CPUs *(the host).*



- **Hetherogeneous GPU-acccelerated computing** is the use of a **G**raphics **P**rocessing **U**nit **to accelerate scientific applications** (among other apps).

  Enhancement of application performance obtained by offloading compute-intensive portions to the GPU (*the device*) while the remainder of the code still runs on the CPUs *(the host).*

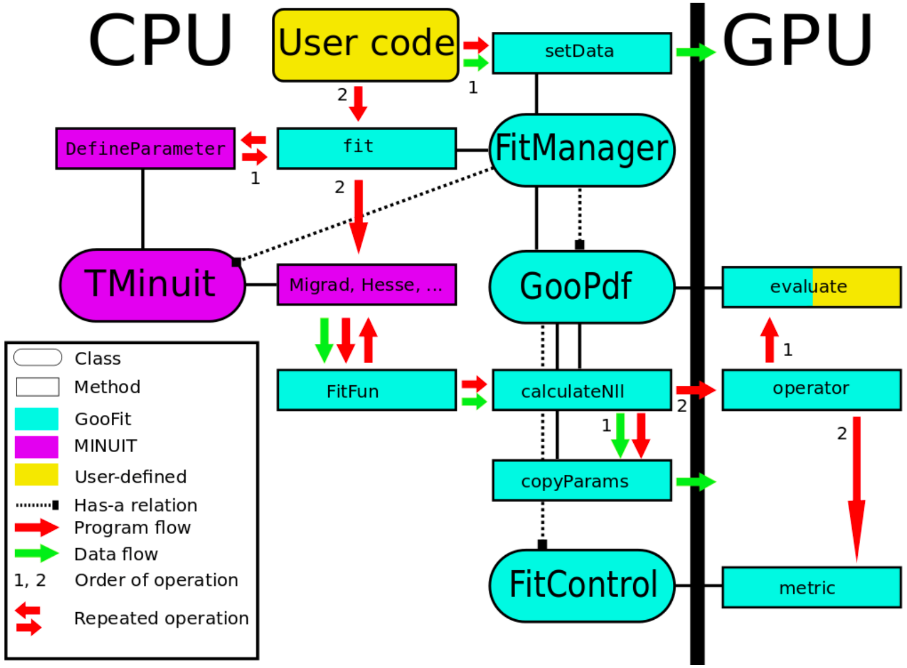


- **From the user's perspective? Applications simply run significantly faster!**
  How much faster? It depends - of course - on the application…
  We want to explore it in the context of the 'end-user HEP analyses' by using *GooFit*.

GooFit framework

- *GooFit* is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.
- The `FitManager` object forms the interface between MINUIT (running on CPU) and a GPU which allows a PDF representing the physical model (`GooPdf` object) to be evaluated in parallel.

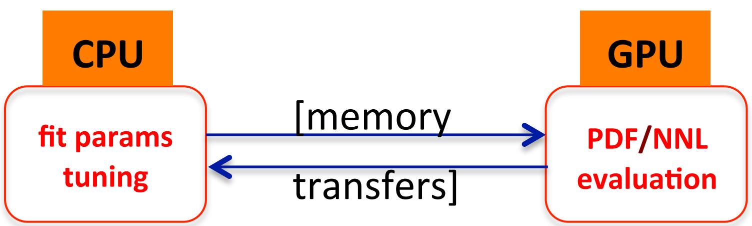
Control & Data Flow of a GooFit program



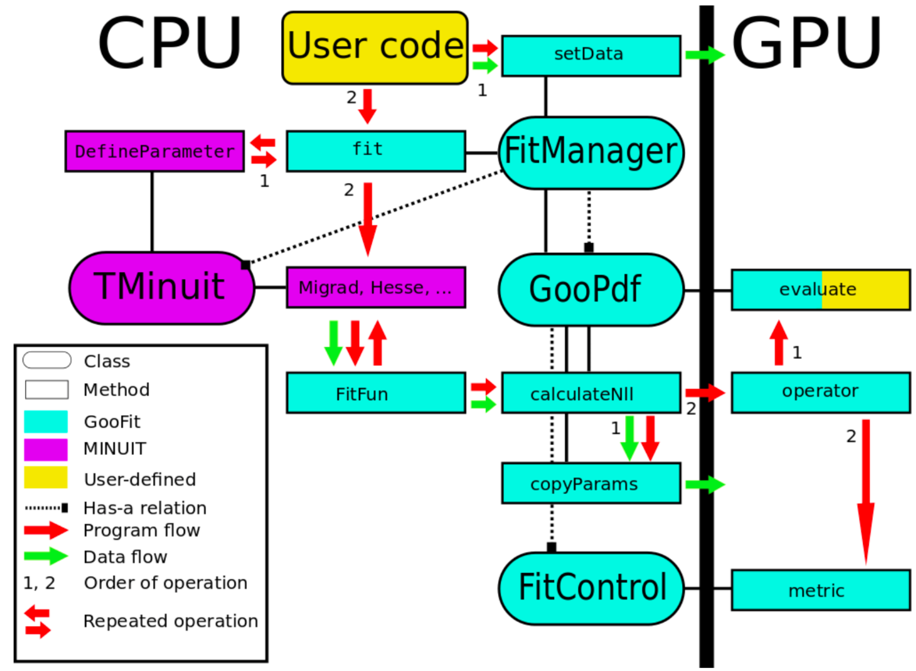
GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

- *GooFit* is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.
- The `FitManager` object forms the interface between MINUIT (running on CPU) and a GPU which allows a PDF representing the physical model (`GooPdf` object) to be evaluated in parallel.

Fit parameters are estimated at each NegLogLikelihood minimization step on the *host side* (CPU) while the PDF/NLL is evaluated on the *device side* (GPU) [all that until convergence]:



Control & Data Flow of a GooFit program



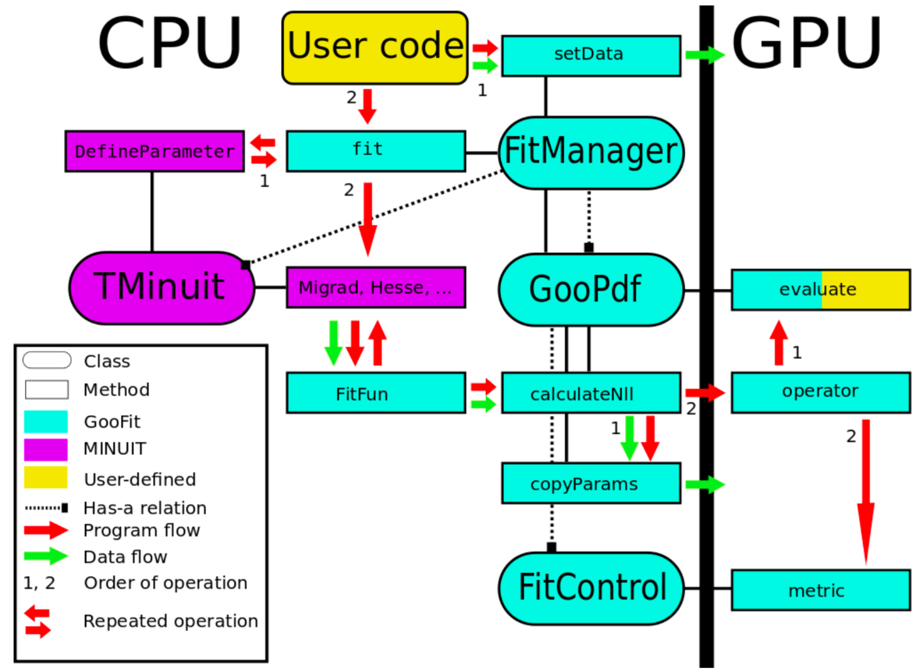
GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

GooFit framework

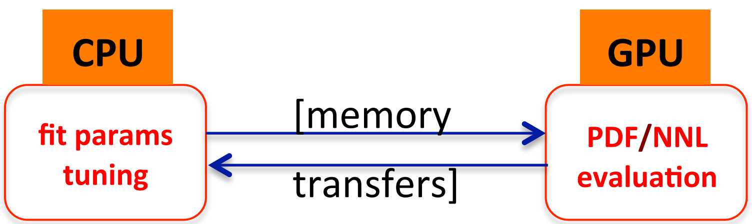
- *GooFit* is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.

- The `FitManager` object forms the interface between MINUIT (running on CPU) and a GPU which allows a PDF representing the physical model (`GooPdf` object) to be evaluated in parallel.

Control & Data Flow of a GooFit program



Fit parameters are estimated at each NegLogLikelihood minimization step on the *host side* (CPU) while the PDF/NLL is evaluated on the *device side* (GPU) [all that until convergence]:



GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

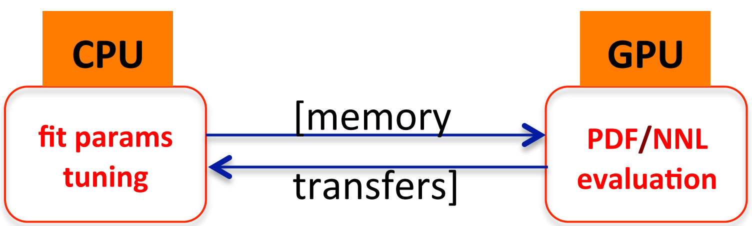
- This can be seen by analysing a cycle with the monitoring tool nVIDIA Visual Profiler [`nvvp`]

GooFit framework

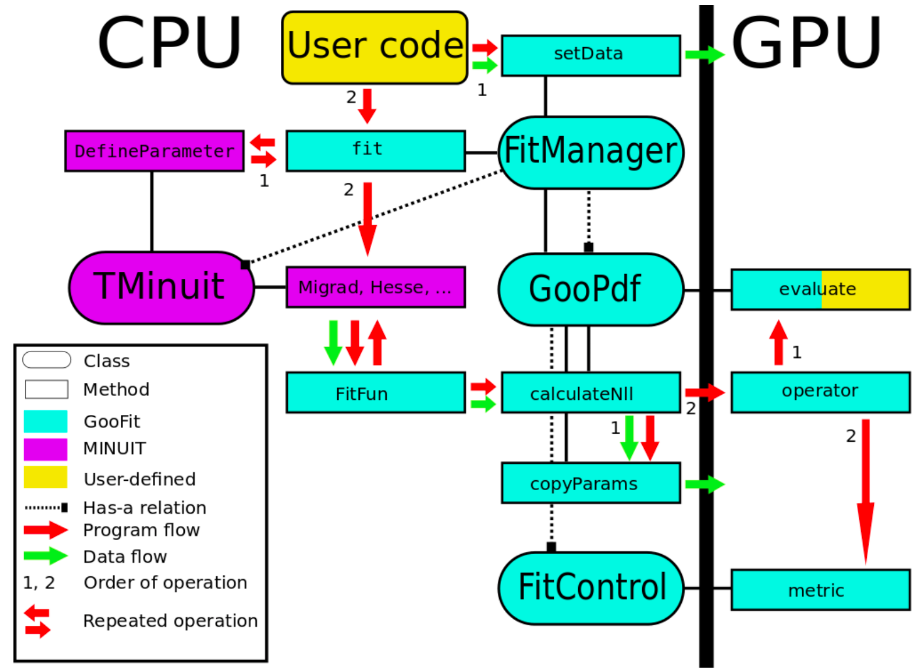
- ***GooFit*** **is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.**

- **The `FitManager` object forms the interface between MINUIT (running on CPU) and a GPU which allows a PDF representing the physical model (`GooPdf` object) to be evaluated in parallel.**

Fit parameters are estimated at each NegLogLikelihood minimization step on the *host side* (CPU) while the PDF/NLL is evaluated on the *device side* (GPU) [all that until convergence]:



Control & Data Flow of a GooFit program



GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

- **This can be seen by analysing a cycle with the monitoring tool nVIDIA Visual Profiler [`nvvp`]**

- **The `FitControl` object allows to switch between χ^2 fits and ML fits (either unbinned and binned).**

A preliminary example of *GooFit/GPUs* capabilities

- **Parameter estimation is a crucial part of many physics analyses.**

PDF evaluation on large datasets is usually the bottleneck in the MINUIT algorithm.

GooFit **acts as an interface between the MINUIT minimization algorithm and a parallel processor which allows a Probability Density Function to be evaluated in parallel.**

A preliminary example of *GooFit/GPUs* capabilities

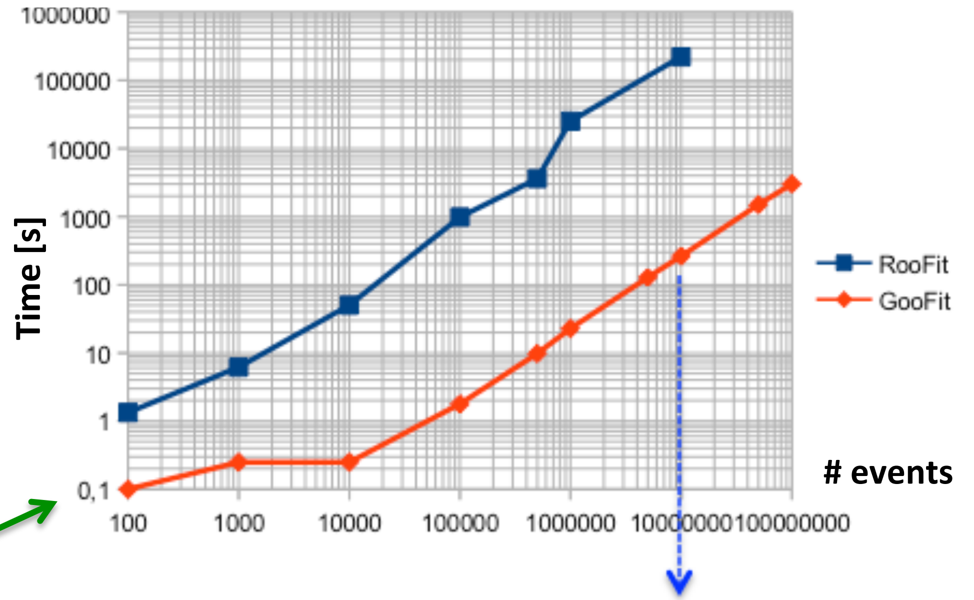
- **Parameter estimation is a crucial part of many physics analyses.**

PDF evaluation on large datasets is usually the bottleneck in the MINUIT algorithm.

GooFit **acts as an interface between the MINUIT minimization algorithm and a parallel processor which allows a Probability Density Function to be evaluated in parallel.**

- **A preliminary test was done with an Unbinned ML fit either by using a single CPU and by using an additional GPU** (an nVIDIA Tesla C2070 hosted @ Bari T2).

  Events according to a Voigtian model (convolution is CPU-intensive) are generated & fitted. The time needed (the negligible generation time is not included) **is studied as a function of the #events:**



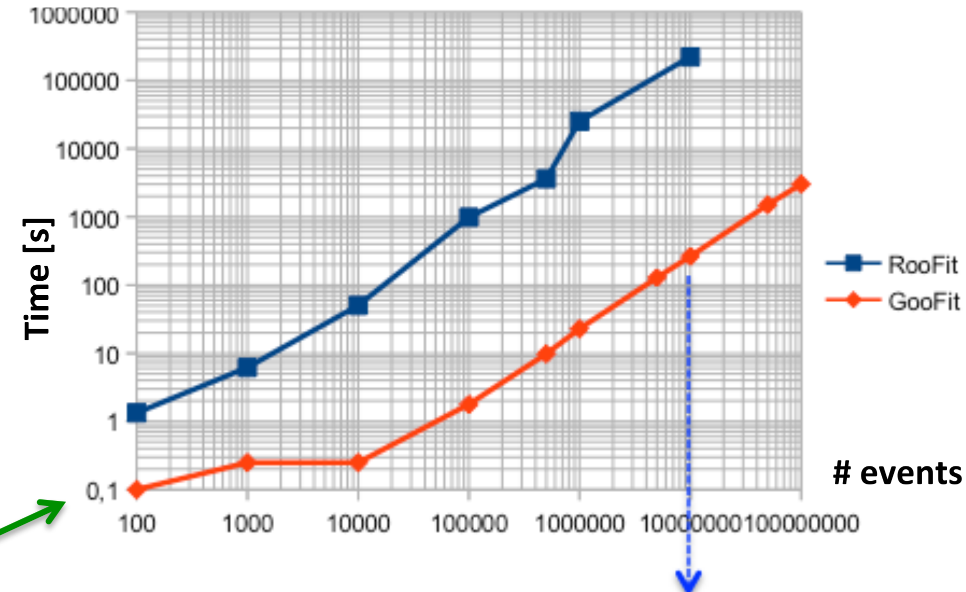
- **Parameter estimation is a crucial part of many physics analyses.**

PDF evaluation on large datasets is usually the bottleneck in the MINUIT algorithm.

***GooFit* acts as an interface between the MINUIT minimization algorithm and a parallel processor which allows a Probability Density Function to be evaluated in parallel.**

- **A preliminary test was done with an Unbinned ML fit either by using a single CPU and by using an additional GPU** (an nVIDIA Tesla C2070 hosted @ Bari T2).

Events according to a Voigtian model (convolution is CPU-intensive) are generated & fitted. The time needed (the negligible generation time is not included) **is studied as a function of the #events:**



For 10M events *RooFit* needs 61h+23m & *GooFit* takes 4m+39s: speed-up ~ 750

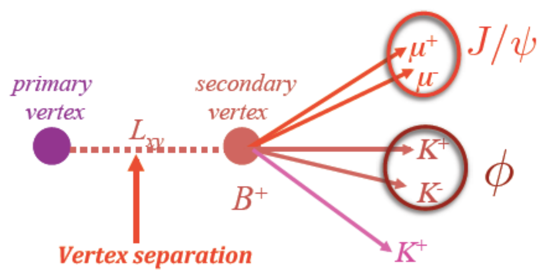
For 1M fitted events with *RooFit* … you need to wait overnight,

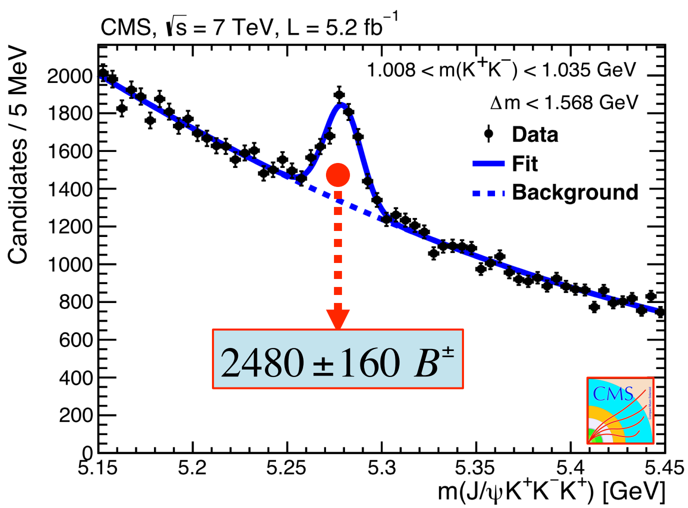
For 10M fitted events with *GooFit* … you need to take an espresso!

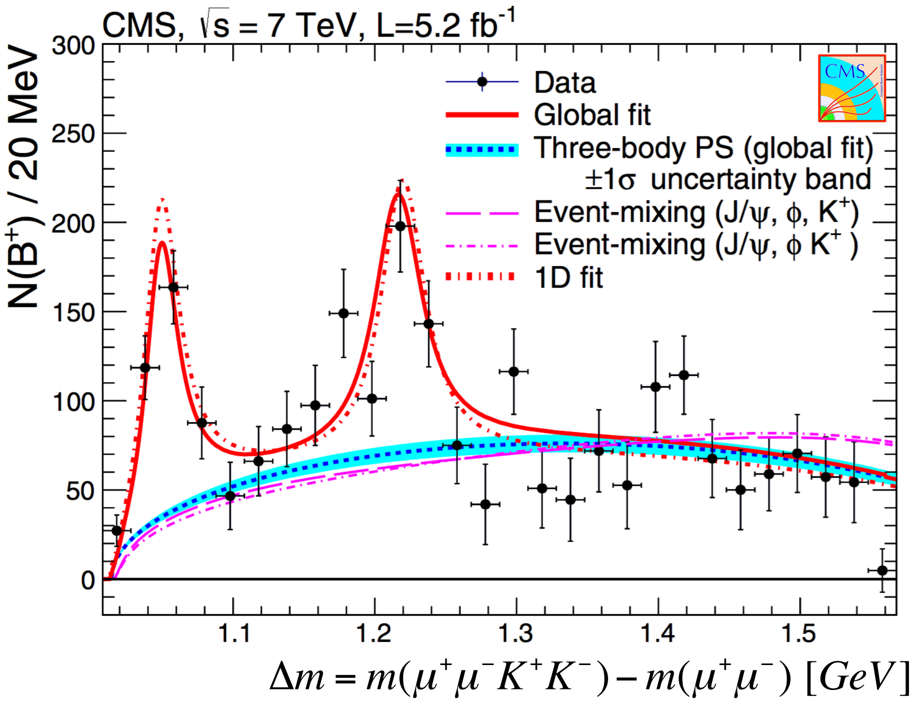
- **As expected, for a Binned ML fit, the speed-up ranges from few units to few dozens (with #bins)**

MC toys for p-value estimation: *GooFit* vs *RooFit*

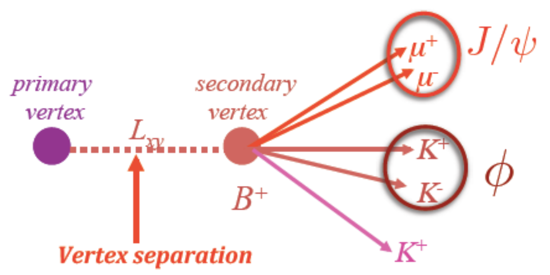
- **To test the computing capabilities of GPUs with respect to CPU cores: a high-statistics toy Monte Carlo technique has been implemented both in *ROOT/RooFit* and *GooFit* frameworks with the aim to estimate the (local) statistical significance of the structure observed by CMS close to the kinematical boundary of the $J/\psi\phi$ invariant mass in the 3-body decay $B^+ \to J/\psi\phi K^+$ [PLB 734 (2014) 261]**

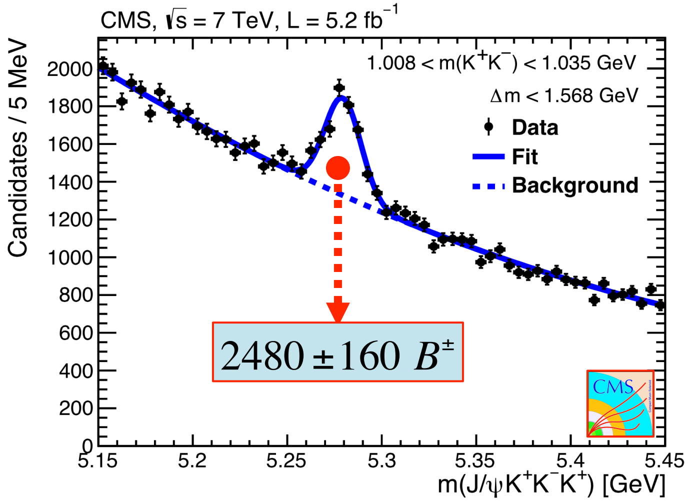


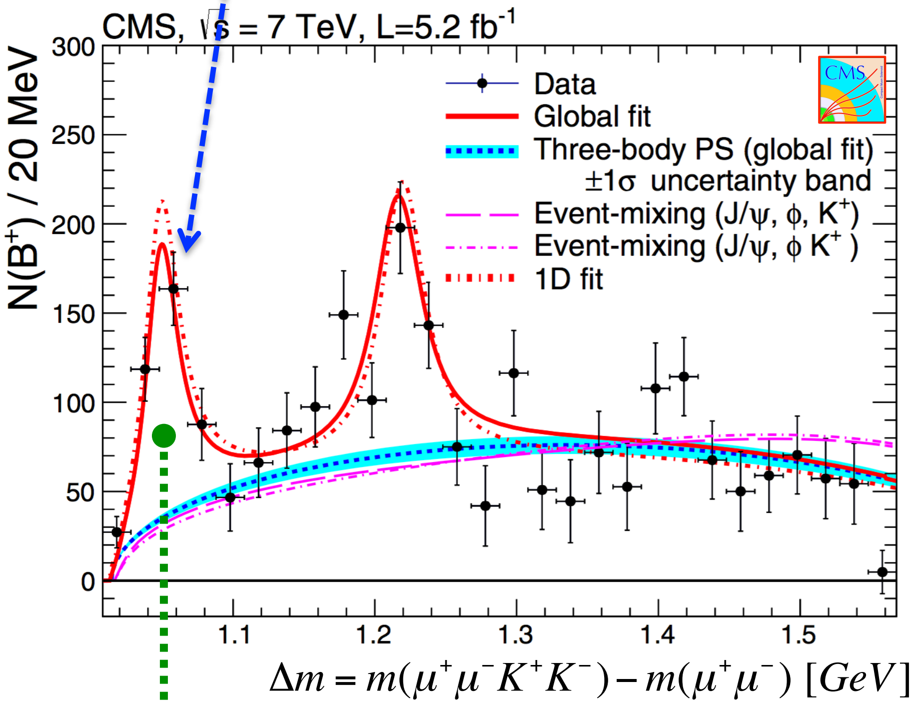




To test the computing capabilities of GPUs with respect to CPU cores: **a high-statistics toy Monte Carlo technique** has been **implemented both in *ROOT/RooFit* and *GooFit* frameworks** with the aim **to estimate the (local) statistical significance** of the structure observed by CMS close to the kinematical boundary of the $J/\psi\phi$ invariant mass in the 3-body decay $B^+ \to J/\psi\phi K^+$ [PLB 734 (2014) 261]







Structure parameters [compatible with Y(4140) by CDF]:

- $m = 4148.0 \pm 2.4(stat.) \pm 6.3(syst.)\ \mathrm{MeV}$
- $\Gamma = 28^{+15}_{-11}(stat.) \pm 19(syst.)\ \mathrm{MeV}$

Test application: the toy MC method

- **MC pseudo-experiments are used to estimate the probability (*p-value*) that background fluctuations would - alone - give rise to a signal as much significant as that seen in the data.**

Toy MC fit cycle (for each generated fluctuation)

- Generation of fluctuated background binned distribution (3-body phase-space model) [total #entries fixed by data ⇨ fits with not-extended ML]
- Null Hypothesis binned ML fit performed with the phase-space model only
-

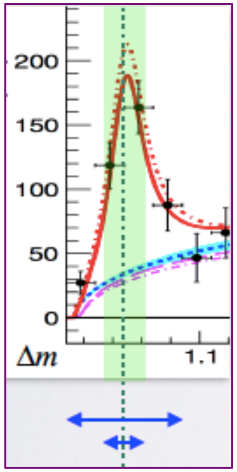
Test application: the toy MC method

MC pseudo-experiments are used to estimate the probability (*p-value*) that background fluctuations would - alone - give rise to a signal as much significant as that seen in the data.

Toy MC fit cycle (for each generated fluctuation)

- Generation of fluctuated background binned distribution (3-body phase-space model)
  [total #entries fixed by data ⇨ fits with not-extended ML]
- Null Hypothesis binned ML fit performed with the phase-space model only
- Alternative Hypothesis binned ML fit performed with the phase-space model + Voigtian PDF
  [the latter is truncated to correctly account for the kinematical threshold; the Gaussian resolution function has width fixed @ 2MeV]. Signal yield constrained > 0.

Note: for each bin, the PDF value is estimated by ROOT integration over the bin [*time-consuming* but needed: steep signal w.r.t. bin size]



- Fit performed 8 times within the region of interest (from CDF: no LEE) trying different starting values (2 masses & 4 widths).

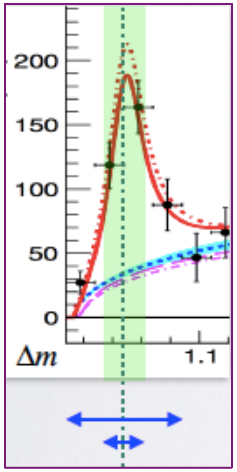
Test application: the toy MC method

- **MC pseudo-experiments are used to estimate the probability (*p-value*) that background fluctuations would - alone - give rise to a signal as much significant as that seen in the data.**

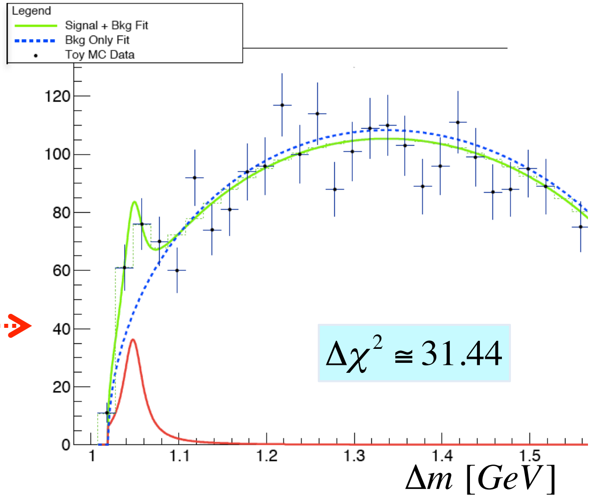
Toy MC fit cycle (for each generated fluctuation)

- Generation of fluctuated background binned distribution (3-body phase-space model)
  [total #entries fixed by data ⇨ fits with not-extended ML]
- Null Hypothesis binned ML fit performed with the phase-space model only
- Alternative Hypothesis binned ML fit performed with the phase-space model + Voigtian PDF
  [the latter is truncated to correctly account for the kinematical threshold; the Gaussian resolution function has width fixed @ 2MeV]. Signal yield constrained > 0.

<u>Note</u>: for each bin, the PDF value is estimated by ROOT integration over the bin [*time-consuming* but needed: steep signal w.r.t. bin size]



- Fit performed 8 times within the region of interest (from CDF: no LEE) trying different starting values (2 masses & 4 widths).
- For each fit calculate a $\Delta\chi^2$ w.r.t. the Null Hypothesis fit; the best $\Delta\chi^2$ fit among the 8 alternative fits is chosen !
- A $\Delta\chi^2$ (our test statistic) distribution is obtained over the sample of MC toys.

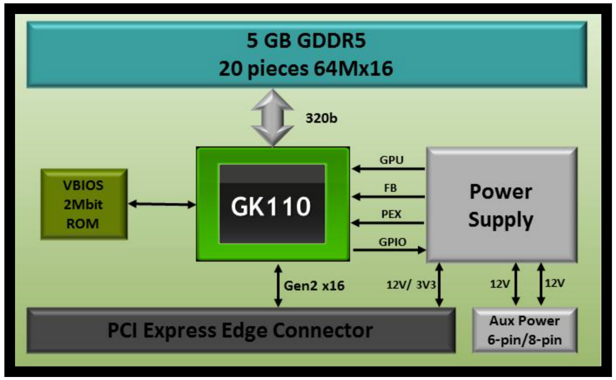


Hardware set-up

» Used: **1 server** hosting **2** nVIDIA **TeslaK20** & **1 server** hosting **1** nVIDIA **TeslaK40**

NVIDIA

Tesla K20 @ *BC²S* (*)



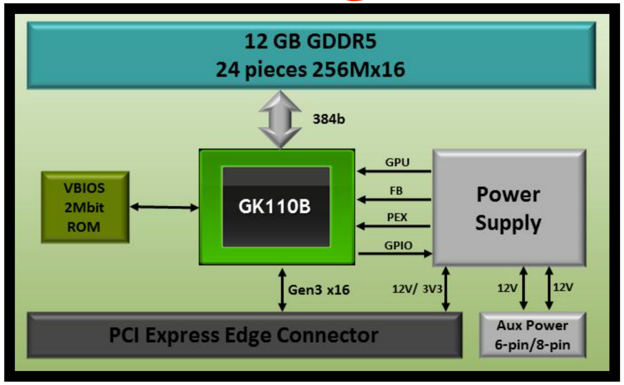
GPU

Numero of GPU	2 x GK110
Number of CUDA cores	**2 x 2,496**
Memory per GPU (GDDR5)	2 x 5 GB
Memory bandwidth per board	208 Gbytes/sec

CPU

- 16 cores: E5-2640 v2 @ 2.00GHz (32 with HT)
- 64 GB RAM

Tesla K40 @ *ReCaS* (*)



GPU

Numero of GPU	1 x GK110B
Number of CUDA cores	**2,880**
Memory per GPU (GDDR5)	12 GB
Memory bandwidth per board	288 Gbytes/sec

CPU

- 20 cores: E5-2640 v2 @ 1.70GHz (40 with HT)
- 256 GB RAM

(*) http://www.recas-bari.it

Performance of *GooFit* vs *ROOT/RooFit:* a preliminary result

- A **first result** obtained is simple comparison between the MC Toys procedure running on a **single GPU** via *GooFit* and on a **single CPU** . The speed ups:

S = 62 (TeslaK40) **S = 48 (TeslaK20)**

For 15k MC Toys produced (Highly time consuming for ROOT: ***~6 hours***)

Performance of *GooFit* vs *ROOT/RooFit:* a preliminary result

- A **first result** obtained is simple comparison between the MC Toys procedure running on a **single GPU** via *GooFit* and on a **single CPU** . The speed ups:

S = 62 (TeslaK40) **S = 48 (TeslaK20)**

For 15k MC Toys produced (Highly time consuming for ROOT: ***~6 hours***)

This kind of application (*binned fit* & *few parameters*) **doesn't exploit** the whole GPU computational capability.

```
+------------------------------------------------------+
| NVIDIA-SMI 340.29     Driver Version: 340.29         |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K20m          Off  | 0000:02:00.0     Off |                    0 |
| N/A   29C    P0    51W / 225W |     82MiB /  4799MiB |     66%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K20m          Off  | 0000:84:00.0     Off |                    0 |
| N/A   27C    P8    25W / 225W |     12MiB /  4799MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Compute processes:                                               GPU Memory |
|  GPU       PID  Process name                                     Usage      |
|=============================================================================|
|    0     31180  GooToyMC                                              67MiB |
+-----------------------------------------------------------------------------+
```

66%

Example snapshot of `nvidia-smi` (nvidia monitoring tool – top) for a single process.

Performance of *GooFit* vs *ROOT/RooFit:* a preliminary result

- A **first result** obtained is simple comparison between the MC Toys procedure running on a **single GPU** via *GooFit* and on a **single CPU** . The speed ups:

S = 62 (TeslaK40) **S = 48 (TeslaK20)**

For 15k MC Toys produced (Highly time consuming for ROOT: ***~6 hours***)

This kind of application (*binned fit* & *few parameters*) **doesn't exploit** the whole GPU computational capability.

```
+------------------------------------------------------+
| NVIDIA-SMI 340.29     Driver Version: 340.29         |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K20m          Off  | 0000:02:00.0     Off |                    0 |
| N/A   29C    P0    51W / 225W |     82MiB /  4799MiB |     66%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K20m          Off  | 0000:84:00.0     Off |                    0 |
| N/A   27C    P8    25W / 225W |     12MiB /  4799MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Compute processes:                                               GPU Memory |
|  GPU       PID  Process name                                     Usage      |
|=============================================================================|
|    0     31180  GooToyMC                                              67MiB |
+-----------------------------------------------------------------------------+
```

66%

Example snapshot of `nvidia-smi` (nvidia monitoring tool – top) for a single process.

How to exploit the full computational power of a GPU?

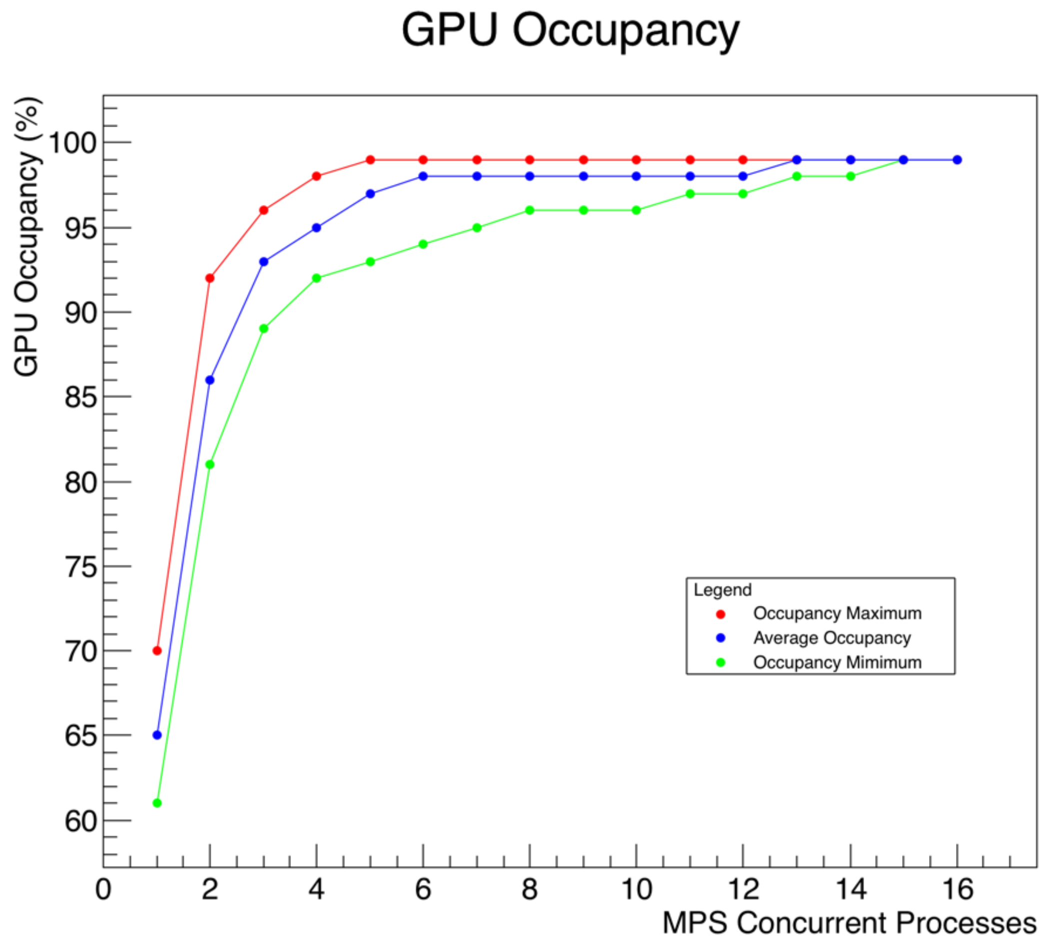
nVidia Multi Process Server

- The nVidia **Multi Process Server (MPS)** is a tool developed by nVidia that allows to execute multiple processes (up to 16) on the same GPU chip. It acts as a **scheduler**: manages the access to **memory** and **CUDA cores.**

  Here is an example of how it affects the occupancy of a TeslaK40 GPU:

- The nVidia **Multi Process Server (MPS)** is a tool developed by nVidia that allows to execute multiple processes (up to 16) on the same GPU chip. It acts as a **scheduler**: manages the access to **memory** and **CUDA cores.**

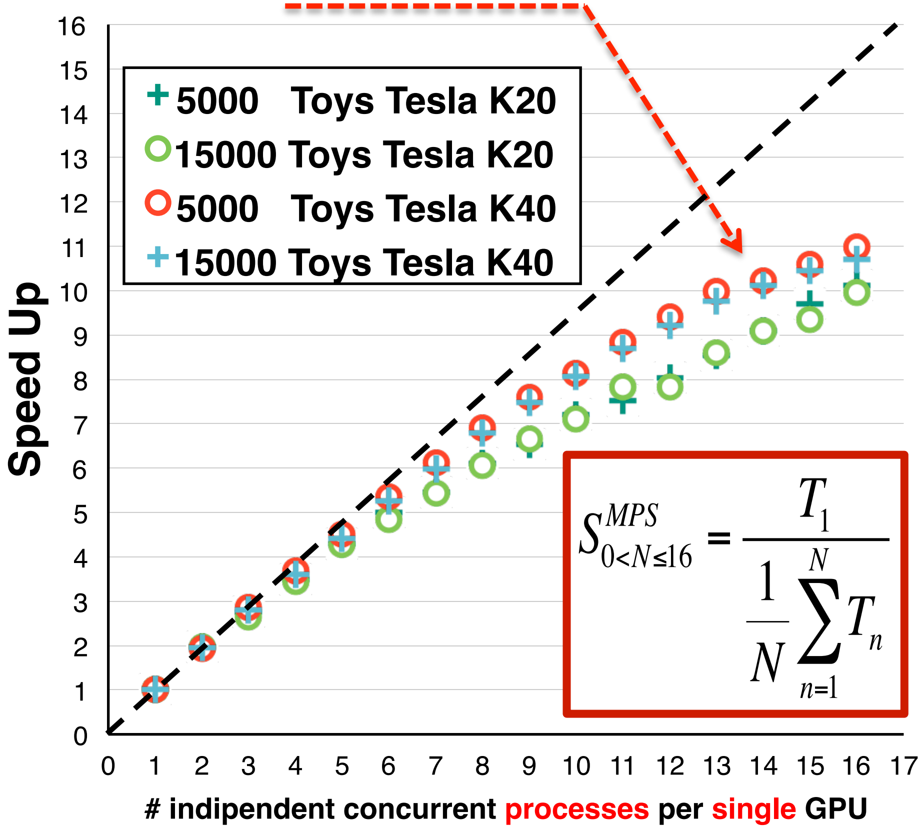
Here is an example of how it affects the occupancy of a TeslaK40 GPU:



- The nVidia **Multi Process Server (MPS)** is a tool developed by nVidia that allows to execute multiple processes (up to 16) on the same GPU chip. It acts as a **scheduler**: manages the access to **memory** and **CUDA cores.**

- **Each** process uses:
  - 1(**shared**) GPU and 1(**exclusively assigned**) CPU

  There is a **saturation effect (Amdhal's law)**

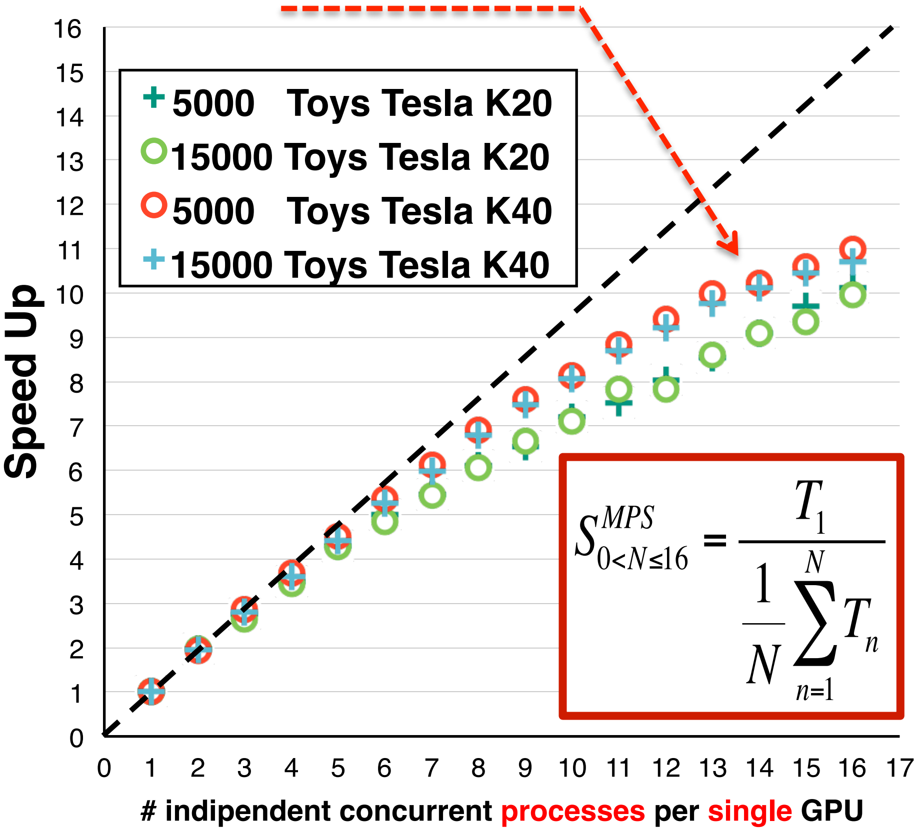


$$S^{MPS}_{0<N\leq 16} = \frac{T_1}{\frac{1}{N}\sum_{n=1}^{N} T_n}$$

- The nVidia **Multi Process Server (MPS)** is a tool developed by nVidia that allows to execute multiple processes (up to 16) on the same GPU chip. It acts as a **scheduler**: manages the access to **memory** and **CUDA cores.**

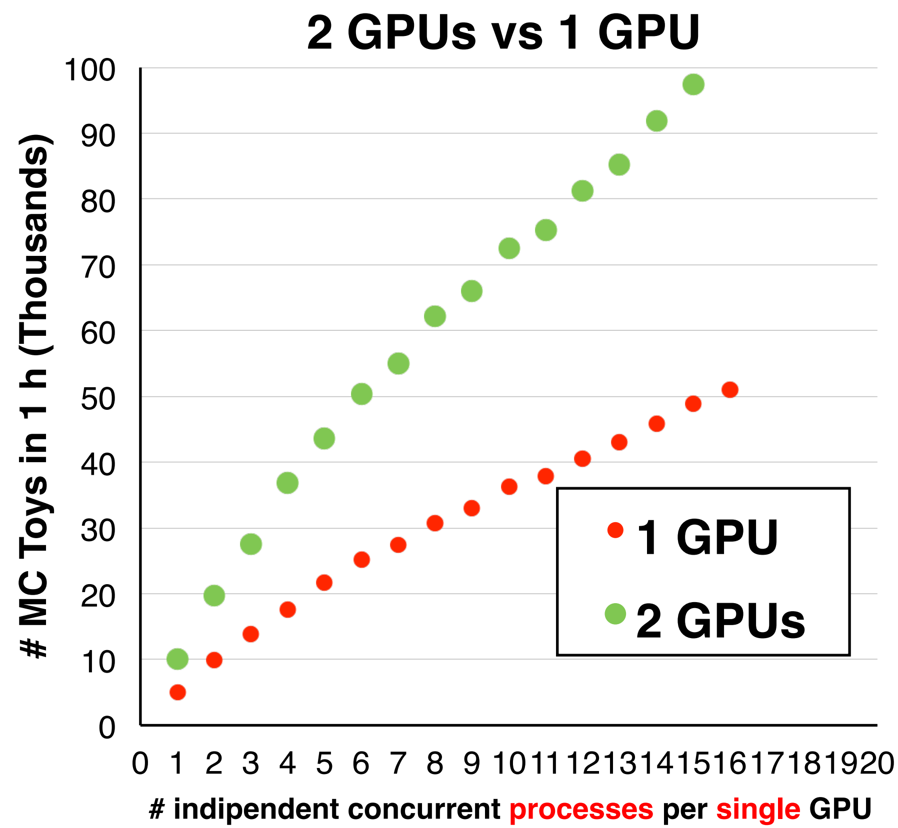
- **Each** process uses:
  - 1(**shared**) GPU and 1(**exclusively assigned**) CPU

  There is a **saturation effect (Amdhal's law)**



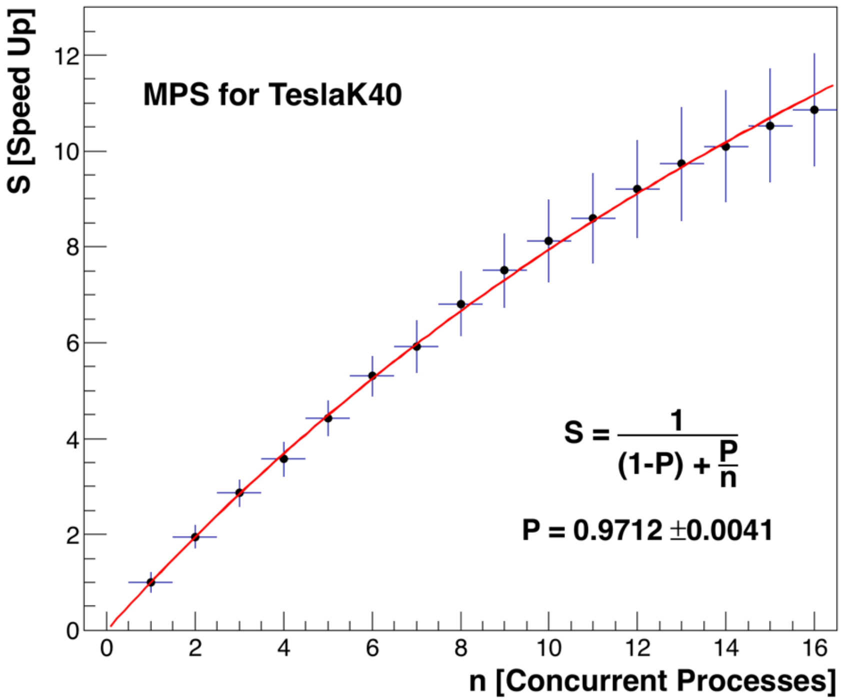
$$S^{MPS}_{0<N\leq16} = \frac{T_1}{\frac{1}{N}\sum_{n=1}^{N} T_n}$$

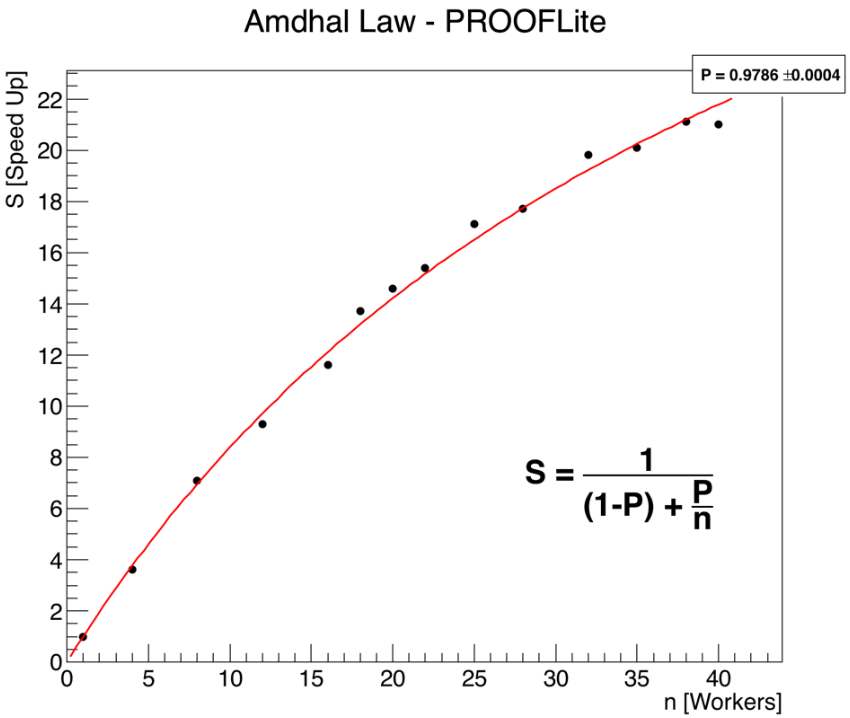
- 1st(2nd) group of $0 < N \leq 16$ processes assigned to… …1st(2nd) GPU (the 2 GPUs TK20 on the same server hosting 32 CPUs via HyperThreading)



Amdhal's Law

In computer architecture, Amdahl's law gives **the theoretical speedup** when using multiple processors as a function of the fraction (**P**) of the code that can be parellilised and of the number of multiprocessors (**n**) used.



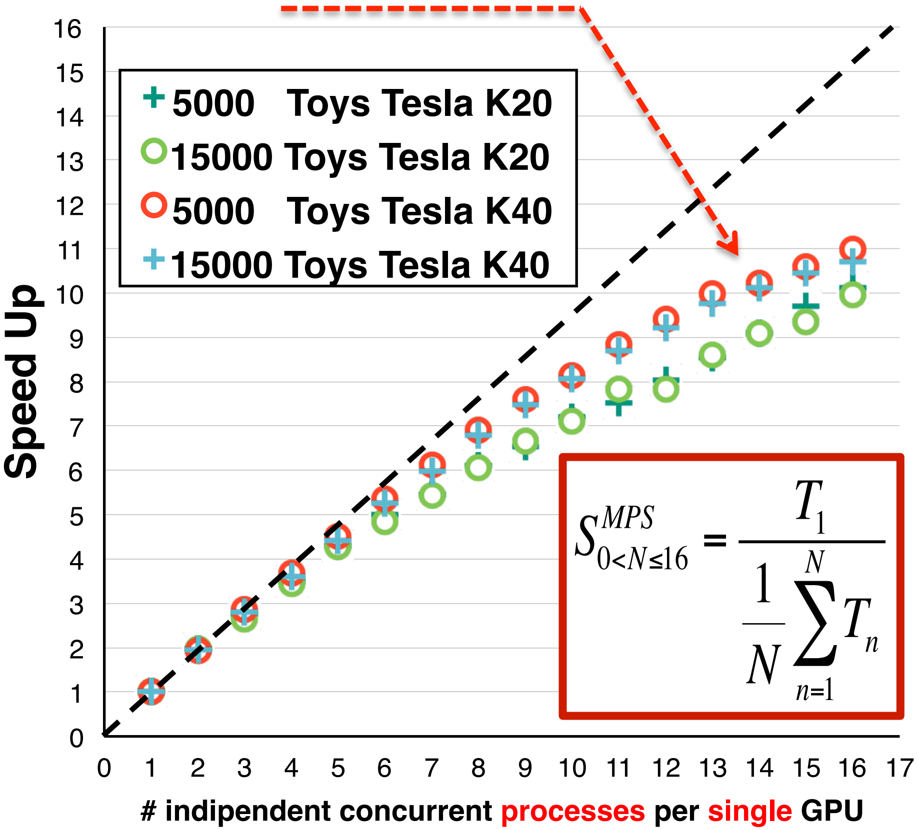


Performance of *GooFit* on nVIDIA Multi Process Server

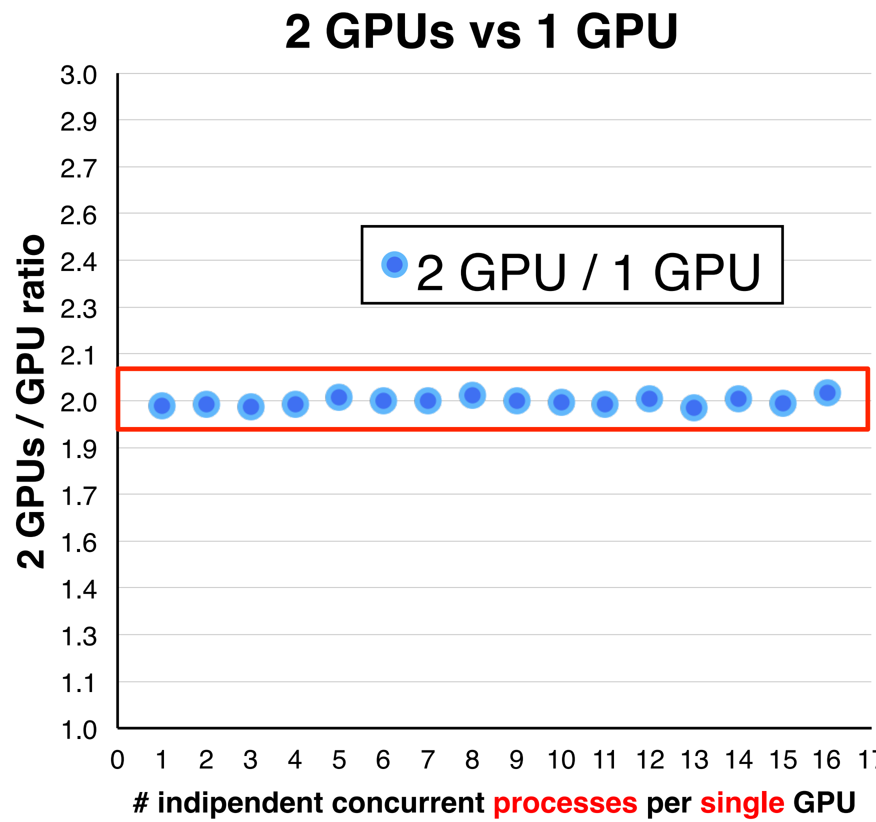
- The nVidia **Multi Process Server (MPS)** is a tool developed by nVidia that allows to execute multiple processes (up to 16) on the same GPU chip. It acts as a **scheduler**: manages the access to **memory** and **CUDA cores.**

- **Each** process uses:
  - 1(**shared**) GPU and 1(**exclusively assigned**) CPU

  There is a **saturation effect (Amdhal's law)**



- 1st(2nd) group of $0 < N \leq 16$ processes assigned to… …1st(2nd) GPU (the 2 GPUs TK20 on the same server hosting 32 CPUs via HyperThreading)



Performance of *RooFit* on CPUs with PROOF-Lite

- **To efficiently run *RooFit* MC toys in parallel on the 72 CPUs available on the 2 servers hosting the GPUs, we use PROOF-Lite that is a dedicated version of PROOF optimized for single multi-core machines** [*].

This ROOT/*RooFit* extension implements a 2-Tier architecture with the master merged into the client, controlling directly the workers (workers are processes not threads).

PROOF has a ***Pull architecture***: **all workers end at the same time** avoiding long queues, unavoidable by running *RooFit* on a cluster in *Push approach* (the last job determines the total exec. time).

[*] G.Ganis et al., *PoS ACAT08 (2008) 007*; A.Pompili et al., *J. Phys.: Conf. Ser.* ***396*** *032043, CHEP12, 2012*

Performance of *RooFit* on CPUs with PROOF-Lite

- **To efficiently run *RooFit* MC toys in parallel on the 72 CPUs available on the 2 servers hosting the GPUs, we use PROOF-Lite that is a dedicated version of PROOF optimized for single multi-core machines** [*].

This ROOT/*RooFit* extension implements a 2-Tier architecture with the master merged into the client, controlling directly the workers (workers are processes not threads).

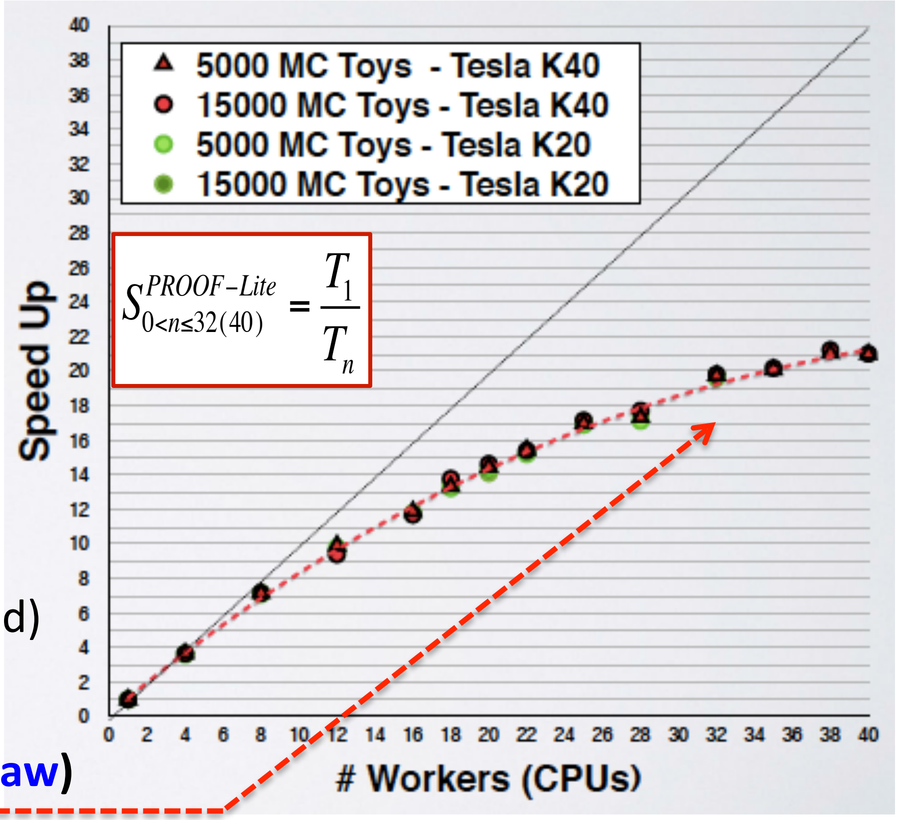
PROOF has a ***Pull architecture***: **all workers end at the same time** avoiding long queues, unavoidable by running *RooFit* on a cluster in *Push approach* (the last job determines the total exec. time).

- Check **speed up performance** on 2 servers:
  - server hosting TK20 has 32 CPUs
  - server hosting TK40 has 40 CPUs

Good scaling with # of MC toys

No difference between 2 servers (as expected)

Speed up perfectly scaling till ~8 workers; then there is a saturation effect (Amdhal's law)



[*] G.Ganis et al., *PoS ACAT08 (2008) 007*; A.Pompili et al., *J. Phys.: Conf. Ser. **396** 032043, CHEP12, 2012*

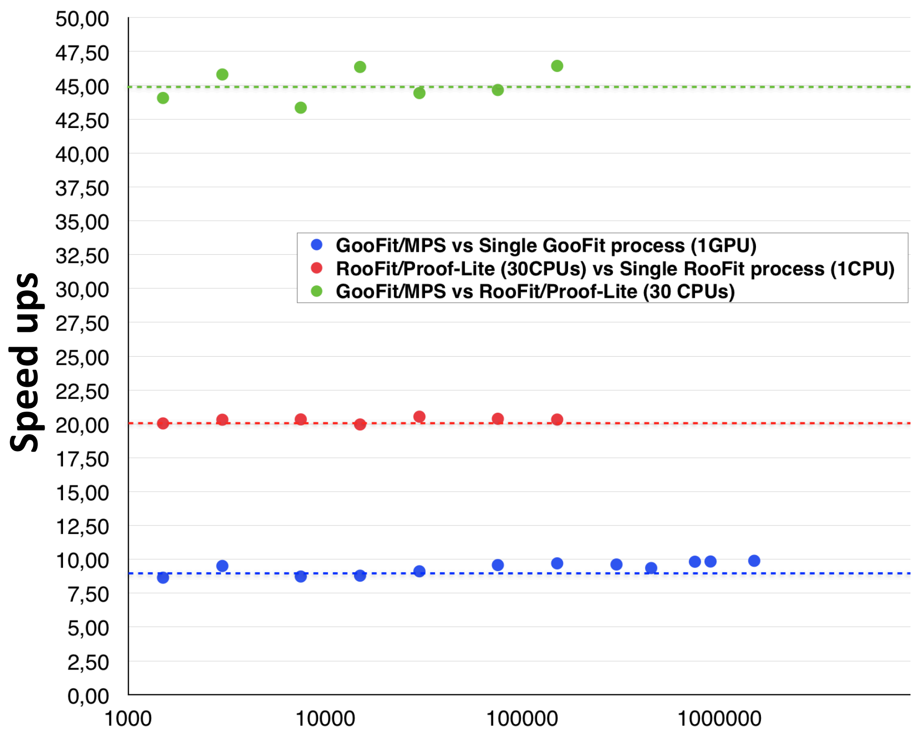
Performance comparison: *RooFit/PROOF-Lite* vs GooFit/MPS - I

- **A first performances' comparison can be carried out on the server hosting 32 CPUs and 2 GPUs TK20 as a function of the # of pseudo-experiments produced.**

- **We can compare: - 1 PROOF-Lite job using 30 workers (on 30 CPU cores)**
  with: - 2 GooFit/MPS jobs (each one running 15 simultaneous processes)

~45

$$S_{n=30=N}\Big|_{2-TK20} = \frac{T_{n=30}^{RooFit}}{T_{N=30}^{GooFit}\Big|_{2-TK20}}$$



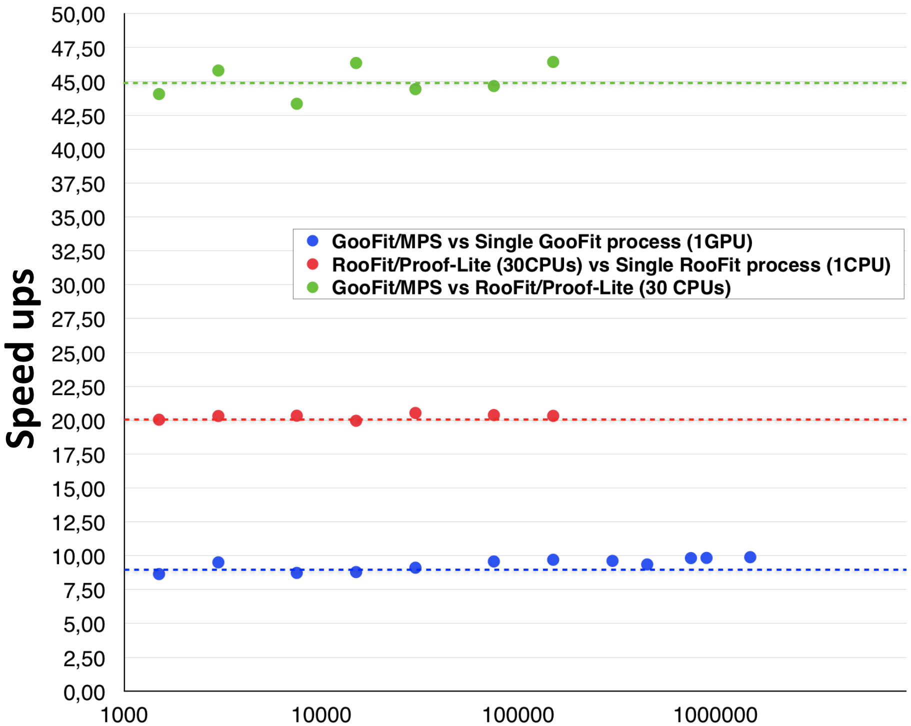
# of processed MC toys (per application)

A first performances' comparison can be carried out on the server hosting 32 CPUs and 2 GPUs TK20 as a function of the # of pseudo-experiments produced.

We can compare: - 1 PROOF-Lite job using 30 workers (on 30 CPU cores)
with: - 2 GooFit/MPS jobs (each one running 15 simultaneous processes)

~45

$$S_{n=30=N}\Big|_{2-TK20} = \frac{T^{RooFit}_{n=30}}{T^{GooFit}_{N=30}\Big|_{2-TK20}}$$



# of processed MC toys (per application)

Good scaling with extended # of MC toys:

$S^{PROOF-Lite}_{n=30}$ **~20**

1 *PROOF-Lite* job using 30 workers
VS
1 *RooFit* job using 1 CPU

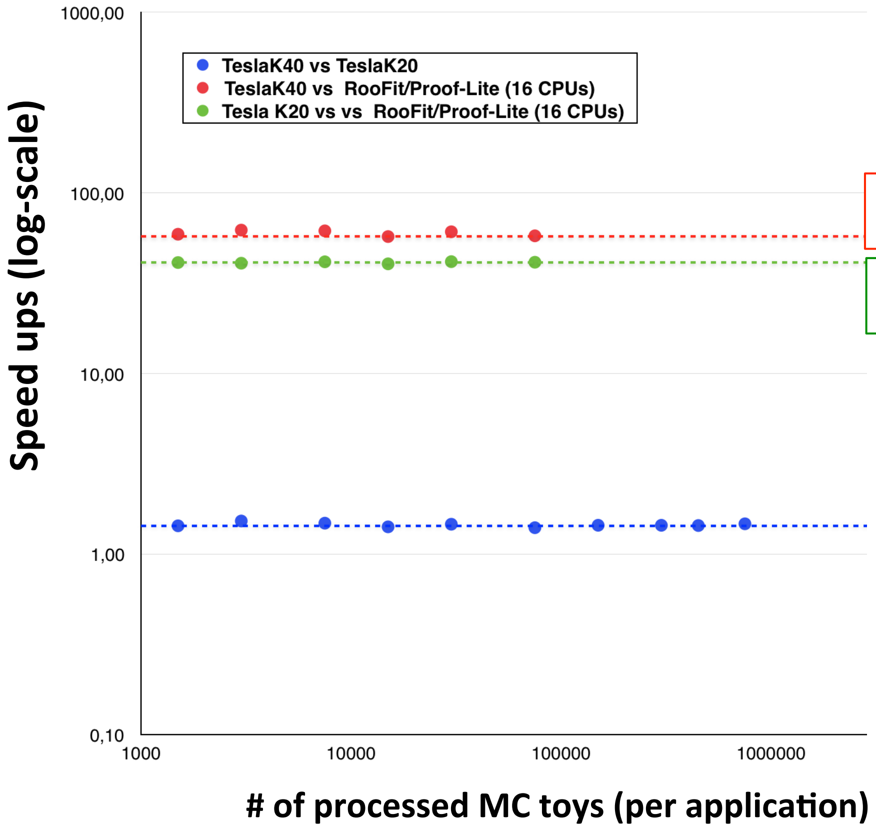
~9 $S^{MPS-TK20}_{N=15}$

1 *GooFit*/MPS job
(running 15 simultaneous processes)
VS
1 *GooFit* job

Performance comparison: *RooFit/PROOF-Lite* vs GooFit/MPS - II

- A **second performances' comparison** can be carried out on both the servers hosting both type of GPUs (TK20 & TK40) as a function of the # of pseudo-experiments produced.
  Here we limit the comparison to **16 independent processes** (due to MPS limit for the single TK40)

- We can compare: - 1 PROOF-Lite job using 16 workers (on 16 CPU cores)
  with: - 1 GooFit/MPS job running 16 simultaneous processes on single TK40 / TK20



~60

$$S_{n=16=N}\Big|_{TK40} = \frac{T^{RooFit}_{n=16}}{T^{GooFit}_{N=16}\Big|_{TK40}}$$

~40

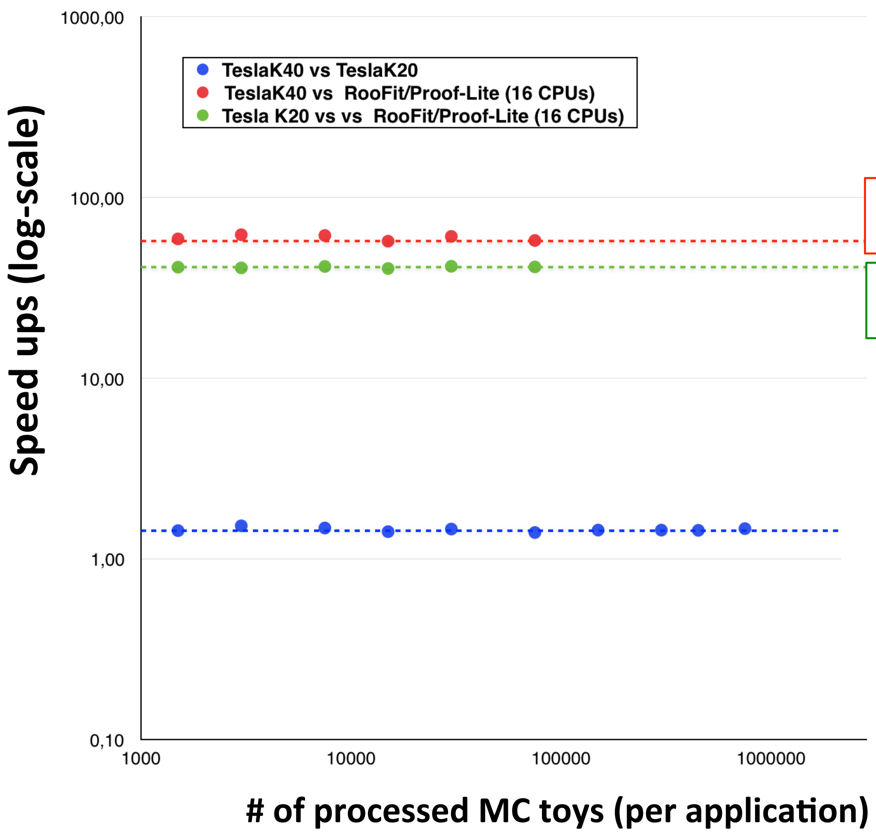
$$S_{n=16=N}\Big|_{TK20} = \frac{T^{RooFit}_{n=16}}{T^{GooFit}_{N=16}\Big|_{TK20}}$$

$$< S_{n=30=N}\Big|_{2-TK20} \sim 45$$

Effect of **PROOF-Lite saturation**

- A **second performances' comparison** can be carried out on both the servers hosting both type of GPUs (TK20 & TK40) as a function of the # of pseudo-experiments produced. Here we limit the comparison to **16 independent processes** (due to MPS limit for the single TK40)

  - We can compare: - 1 PROOF-Lite job using 16 workers (on 16 CPU cores)
    with: - 1 GooFit/MPS job running 16 simultaneous processes on single TK40 / TK20



~60

$$S_{n=16=N}\Big|_{TK40} = \frac{T^{RooFit}_{n=16}}{T^{GooFit}_{N=16}\Big|_{TK40}}$$

~40

$$S_{n=16=N}\Big|_{TK20} = \frac{T^{RooFit}_{n=16}}{T^{GooFit}_{N=16}\Big|_{TK20}}$$

$$< S_{n=30=N}\Big|_{2-TK20} \sim 45$$

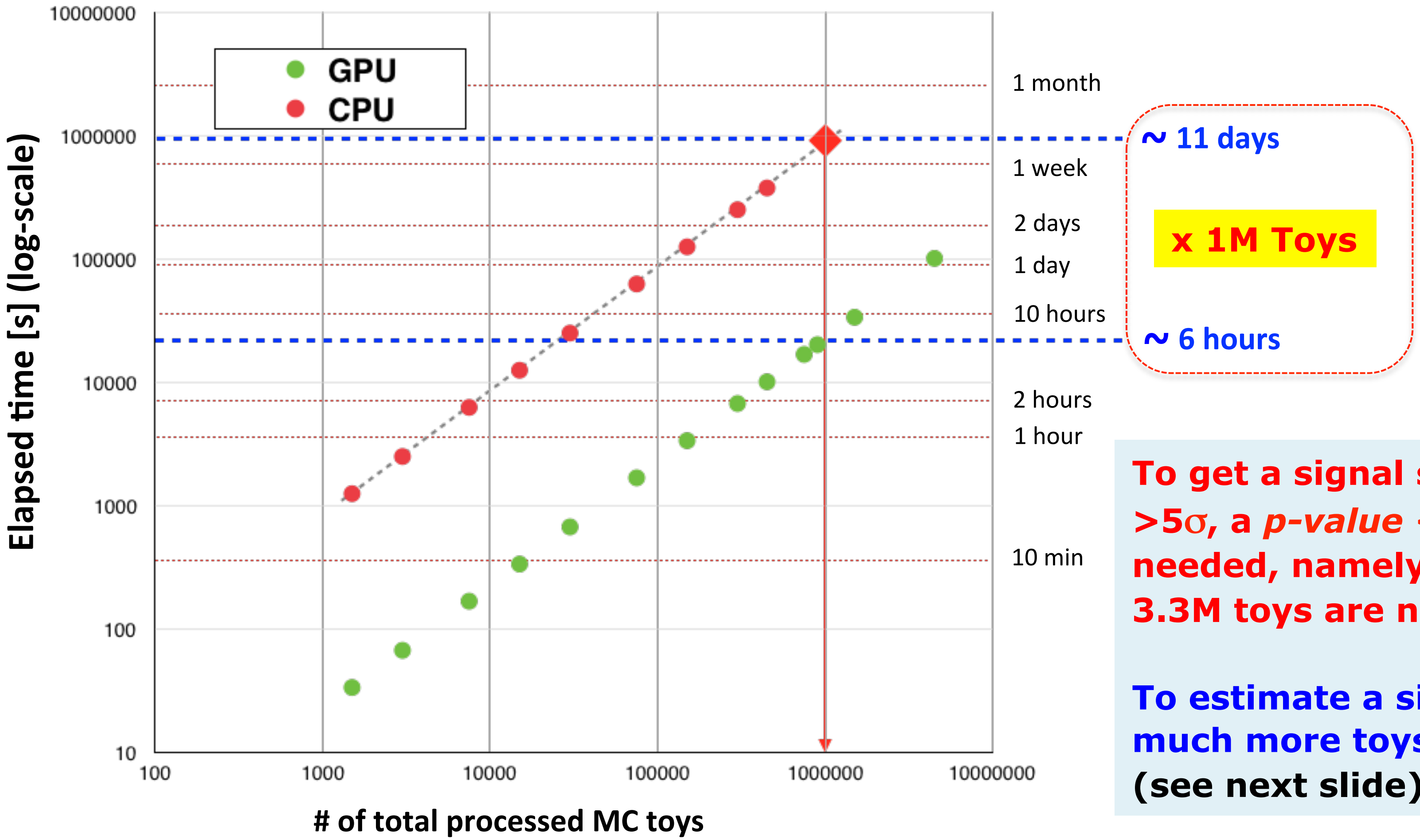
Effect of **PROOF-Lite saturation**

~1.5

$$S^{GPU}_{N=16} = \frac{T_{N=16}\Big|_{TK20}}{T_{N=16}\Big|_{TK40}}$$

Gain within micro-architecture: TK40 vs TK20

- A third performances' comparison can be done from the point of view of the end-user/analyst and the time needed to deliver the pseudo-experiments' task.
Let us assume he has at his own disposal the full computational power used in these studies:
2 servers equipped with 3 GPUs (2 TK20 & 1 TK40) and 72 CPU cores (36 physical cores + HyperThr).



TOTAL SPEED UP

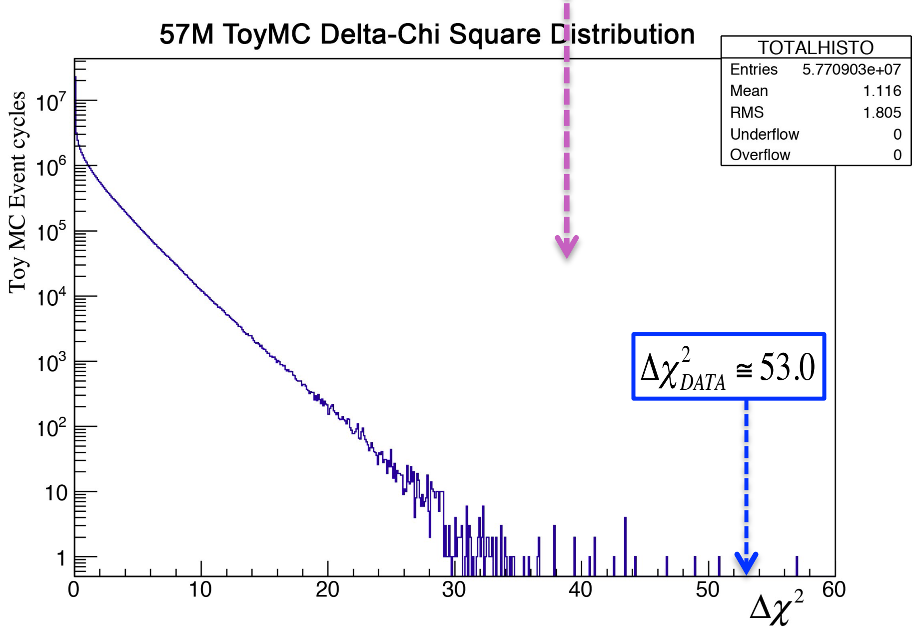
$S \sim 45$

To get a signal significance >5σ, a *p-value < 3x10^-7^* is needed, namely at least 3.3M toys are needed.

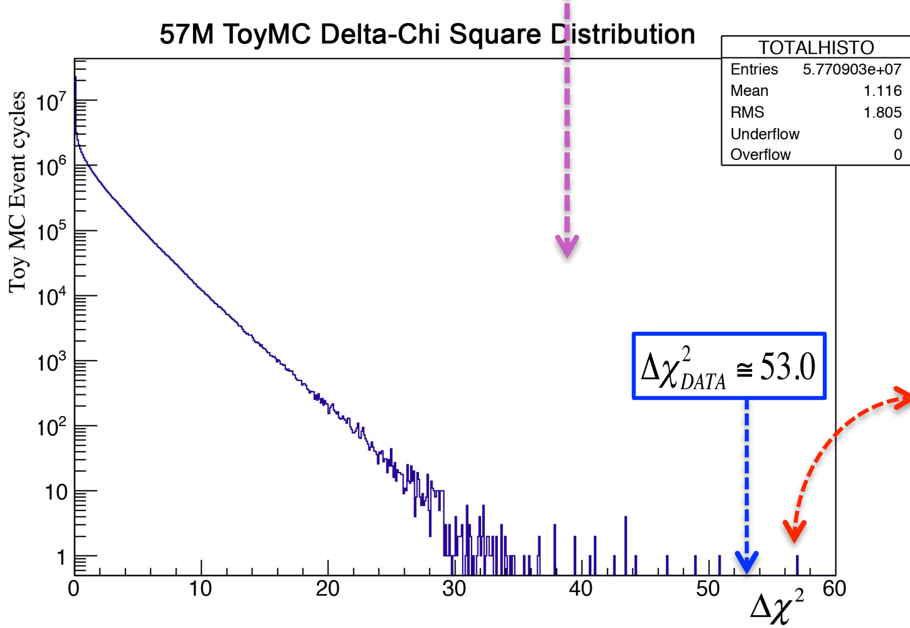
To estimate a signal signif. much more toys are needed (see next slide)

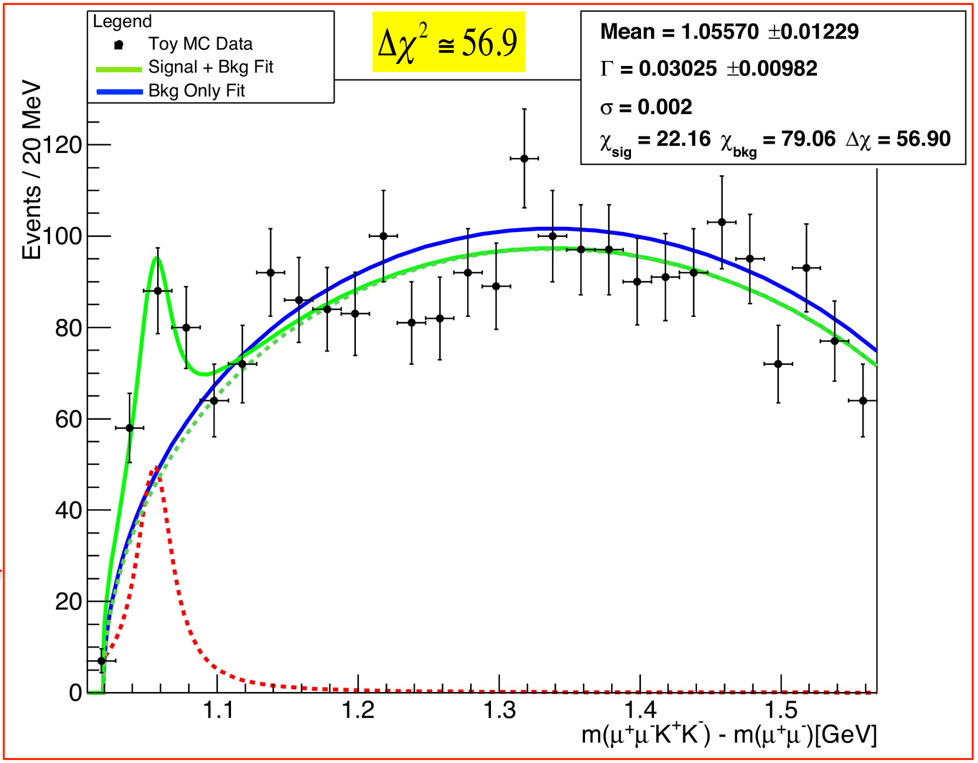
P-Value & statistical significance estimation

- The **final obtained** $\Delta\chi^2$ **distribution** (MC toys production was stopped once a fluctuation with $\Delta\chi^2 > \Delta\chi^2_{DATA}$ was found)



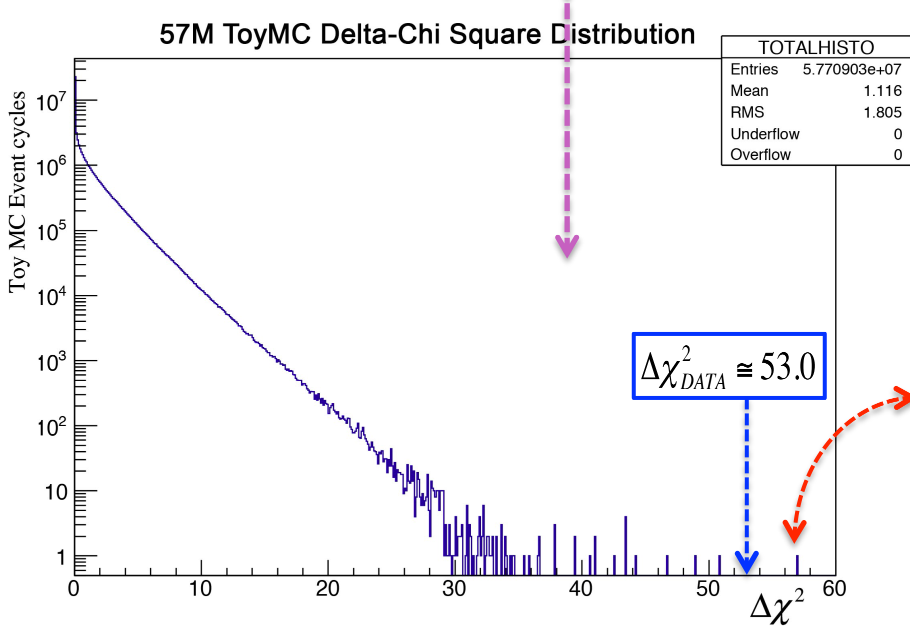
- The **final obtained $\Delta\chi^2$ distribution** (MC toys production was stopped once a fluctuation with $\Delta\chi^2 > \Delta\chi^2_{DATA}$ was found)

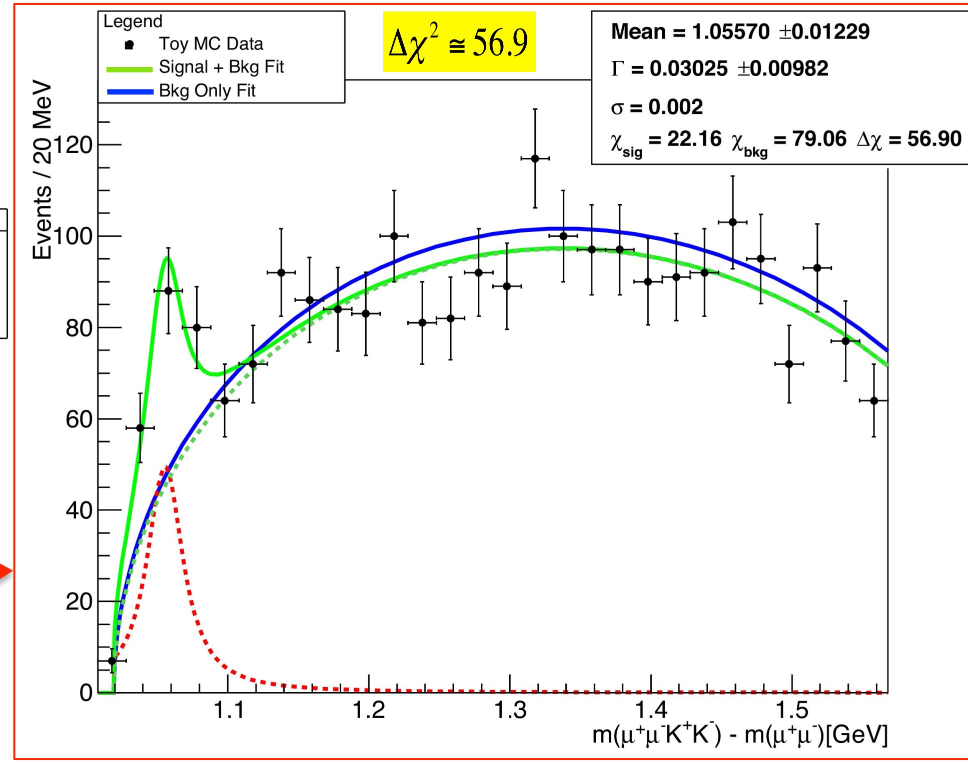




P-Value & statistical significance estimation

- The **final obtained** $\Delta\chi^2$ **distribution** (MC toys production was stopped once a fluctuation with $\Delta\chi^2 > \Delta\chi^2_{DATA}$ was found)





- The **p-value estimation is straightforward:**

$$p-value\ :\ P = \int_{\Delta\chi^2_{DATA}}^{+\infty} \Delta\chi^2 \approx \frac{1}{57.7\cdot 10^6} \cong 1.73\cdot 10^{-8}$$

Equivalent (gaussian) statistical significance:

$$Z\sigma = \Phi^{-1}(1-P)\sigma \cong 5.52\sigma$$

Inverse function of the cumulative distribution of the standard gaussian

Compatible with the lower limit of *5σ* for the statistical significance quoted in the CMS paper **PLB 734 (2014) 261** on the basis of 50.5 millions of MC toys (by *RooFit*).

Exploring the applicability limits of Wilks theorem

Wilks theorem & the need of MC toys - I

[*] S.S.Wilks, *Ann.Math.Stat.* **9** (1938) 60-62

- The **Wilks[*] theorem** is often used to estimate the p-value associated to a new/unexpected signal:

  Given two hypotheses:
  - **Null hypotheses** H_0 with ν_0 d.o.f.
  - **Alternative hypotheses** H_1 with ν_1 d.o.f.

  … **any test statistic** t, defined as a likelihood ratio $-2\ln\lambda = -2\ln\left(\frac{L_{H_0}}{L_{H_1}}\right)$

  [or similarly (in the asymptotic limit) as a $\Delta\chi^2 = \chi^2_{H_0} - \chi^2_{H_1}$],

  approaches a χ^2 distribution with $\nu = \nu_1 - \nu_0$ d.o.f., **provided that these regularity conditions hold:**

  - H_0 and H_1 are nested (H_1 "includes" H_0)
  - while $H_1 \to H_0$ the H_1 parameters are well behaving (defined and not approaching some limit)
  - asymptotic limit (of a large data sample)

-

-

Wilks theorem & the need of MC toys - I

[*] S.S.Wilks, *Ann.Math.Stat.* **9** (1938) 60-62

- **The Wilks[*] theorem is often used to estimate the p-value associated to a new/unexpected signal:**

  Given two hypotheses:
  - **Null hypotheses H_0 with ν_0 d.o.f.**
  - **Alternative hypotheses H_1 with ν_1 d.o.f.**

  ... any test statistic t, defined as a likelihood ratio $-2\ln\lambda = -2\ln\left(\frac{L_{H_0}}{L_{H_1}}\right)$

  [or similarly (in the asymptotic limit) as a $\Delta\chi^2 = \chi^2_{H_0} - \chi^2_{H_1}$],

  approaches a χ^2 distribution with $\nu = \nu_1 - \nu_0$ d.o.f., provided that these regularity conditions hold:
  - **H_0 and H_1 are nested (H_1 "includes" H_0)**
  - **while $H_1 \to H_0$ the H_1 parameters are well behaving (defined and not approaching some limit)**
  - **asymptotic limit (of a large data sample)**

- **Once this theorem holds, the p-value associated to the signal is given by:** $P = \int_{t_{obs}}^{\infty} \chi^2_{\nu_1-\nu_0}(t)\,dt$

  The use of pseudo-experiments to estimate the p-value is not needed

  (but still suggested)

- **When null hypothesis is background-only and the alternative is background+signal, often the above regularity conditions are not all satisfied, and MC toys are mandatory !**

Wilks theorem & the need of MC toys - II

- Indeed this is the case we are dealing with, here!

  The signal parameters in the model of H_1 hypothesis are mass (m), width (Γ) and yield ($\mu \geq 0$).

  When $H_1 \rightarrow H_0$ the problem is that: 1) m and Γ are not well defined, 2) μ tend to the null limit.

  This explains why we have used pseudo-experiments.

- The distributions of test statistic are in general nonpredictable and can be extracted from MC toys!

Wilks theorem & the need of MC toys - II

- **Indeed this is the case we are dealing with, here!**

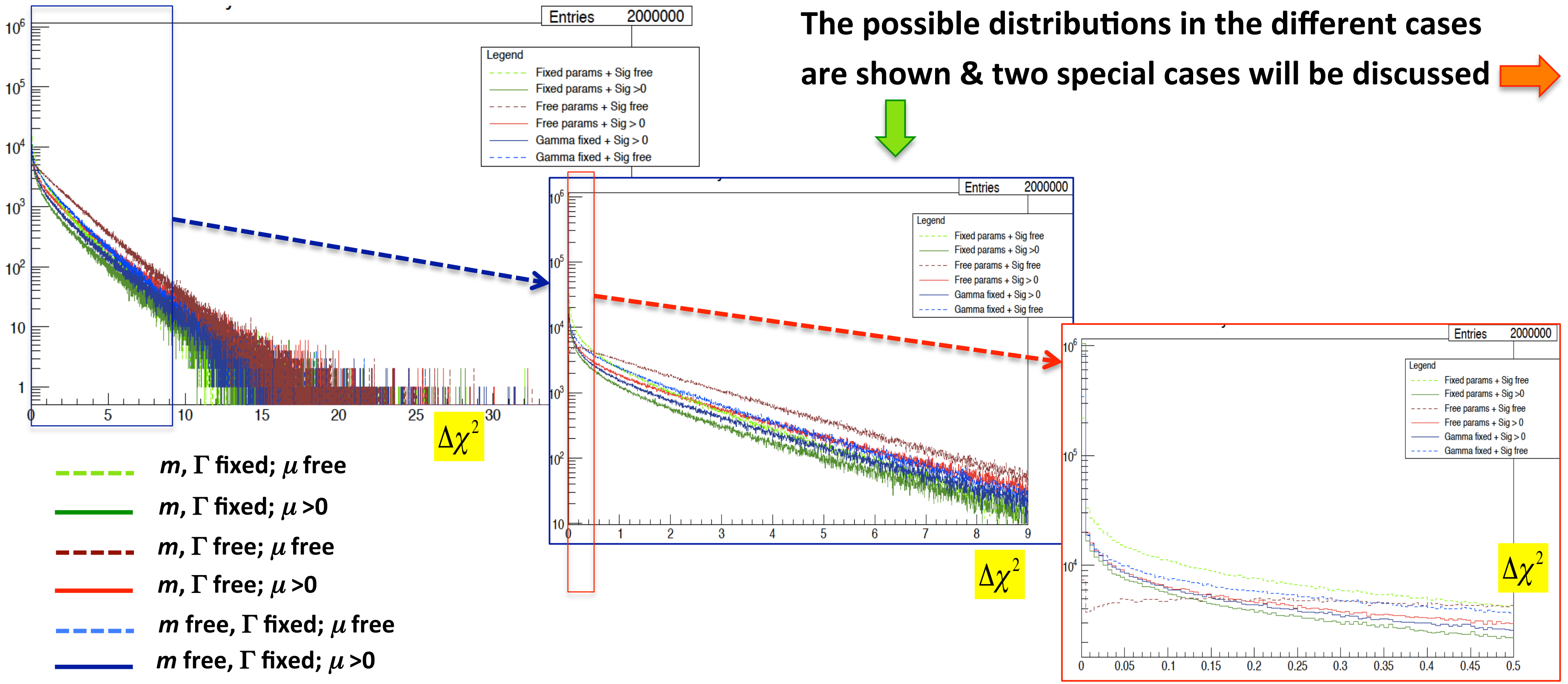
  The signal parameters in the model of H_1 hypothesis are mass (m), width (Γ) and yield ($\mu \geq 0$).

  When $H_1 \rightarrow H_0$ the problem is that: 1) m and Γ are not well defined, 2) μ tend to the null limit.

  This explains why we have used pseudo-experiments.

- **The distributions of test statistic are in general nonpredictable and can be extracted from MC toys!**

The possible distributions in the different cases are shown & two special cases will be discussed



Special case in which Wilks theorem holds

- Consider the test statistic $t_\mu = -2\ln\lambda(\mu)$ [μ: *strength parameter*] as the basis of the statistical test. This could be a test of μ=0 for purposes of establishing the existence of a signal process (no constrain on μ).

In the latter case, following Cowan *et al*. [*] the PDF of the test statistic approaches a chi-square distribution for 1 d.o.f.: [in agreement with Wilks theorem!]

$$f(t_\mu|\mu) = \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{t_\mu}}e^{-t_\mu/2}$$

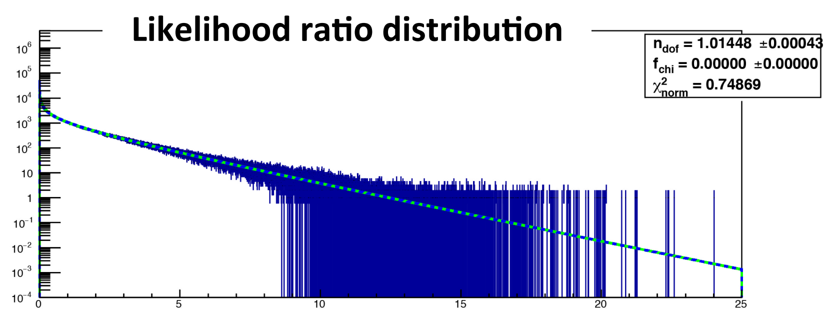
[*] Cowan *et al.*, EPJ C71 (2011) 1554

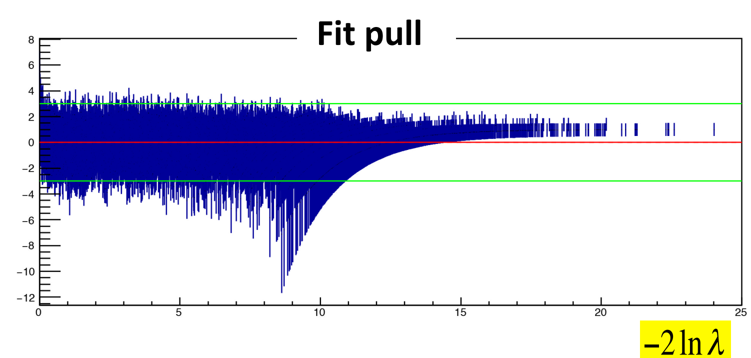
- Consider the test statistic $t_\mu = -2\ln\lambda(\mu)$ [μ: *strength parameter*] as the basis of the statistical test. This could be a test of μ=0 for purposes of establishing the existence of a signal process (no constrain on μ).

In the latter case, following Cowan *et al.* [*] the PDF of the test statistic approaches a chi-square distribution for 1 d.o.f.: [in agreement with Wilks theorem!]

$$f(t_\mu|\mu) = \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{t_\mu}}e^{-t_\mu/2}$$

- Let us fix the m & Γ parameters, (to the CMS estimates from the fit to data) while leaving μ free in our ML fits (μ is not properly a signal yield).





By fitting our likelihood ratio distrib. we indeed get:

$$\text{d.o.f.} \approx 1.014 \pm 0.001$$

$$\chi^2_{norm} = 1.009 \quad P(fit) = 0.118$$

[*] Cowan *et al.*, EPJ C71 (2011) 1554

Special case: asymptotic formula by Cowan *et al.* [*] holds

- **Consider the special case of the test statistic t_μ with the purpose to test μ=0 in a class of model where we assume μ>0. Rejecting μ=0 (the null hypothesis) leads to the discovery of a new signal.**

  In this case following Cowan *et al*. the test statistic is:
  $$q_0 = \begin{cases} -2\ln\lambda(0) \\ 0 \end{cases} \quad with \begin{cases} \hat{\mu} \geq 0 \\ \hat{\mu} < 0 \end{cases}$$

  Cowan *et al*. derive analitically that the PDF of q_0 is an equal mixture of a delta function at 0 & a chi-square distribution for 1 d.o.f.:
  $$g(q_0|\mu=0) = \frac{1}{2}\delta(q_0) + \frac{1}{2}\left[\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{q_0}}e^{-q_0/2}\right]$$

-

[*] Cowan *et al.*, EPJ C71 (2011) 1554

- Consider the special case of the test statistic t_μ with the purpose to test μ=0 in a class of model where we assume μ>0. Rejecting μ=0 (the null hypothesis) leads to the discovery of a new signal.

In this case following Cowan *et al*. the test statistic is:

$$q_0 = \begin{cases} -2\ln\lambda(0) \\ 0 \end{cases} with \begin{cases} \hat{\mu} \geq 0 \\ \hat{\mu} < 0 \end{cases}$$

Cowan *et al*. derive analitically that the PDF of q_0 is an equal mixture of a delta function at 0 & a chi-square distribution for 1 d.o.f.:

$$g(q_0|\mu=0) = \frac{1}{2}\delta(q_0) + \frac{1}{2}\left[\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{q_0}}e^{-q_0/2}\right]$$

- Let us fix the m & Γ parameters (to the CMS estimates from fit to data) while constraining $\mu \geq 0$ in our ML fits (μ represents a signal yield here).

By fitting our likelihood ratio distrib. we indeed get:

$$\text{d.o.f.} \approx 0.992 \pm 0.001$$

$$\text{weight } C_{\chi^2} \approx 0.507 \pm 0.01$$

[*] Cowan *et al.*, EPJ C71 (2011) 1554

Special case: asymptotic formula by Cowan *et al.* [*] holds

- **Consider the special case of the test statistic t_μ with the purpose to test μ=0 in a class of model where we assume μ>0. Rejecting μ=0 (the null hypothesis) leads to the discovery of a new signal.**

In this case following Cowan *et al*. the test statistic is:

$$q_0 = \begin{cases} -2\ln\lambda(0) \\ 0 \end{cases} with \begin{cases} \hat{\mu} \geq 0 \\ \hat{\mu} < 0 \end{cases}$$

Cowan *et al*. derive analitically that the PDF of q_0 is an equal mixture of a delta function at 0 & a chi-square distribution for 1 d.o.f.:

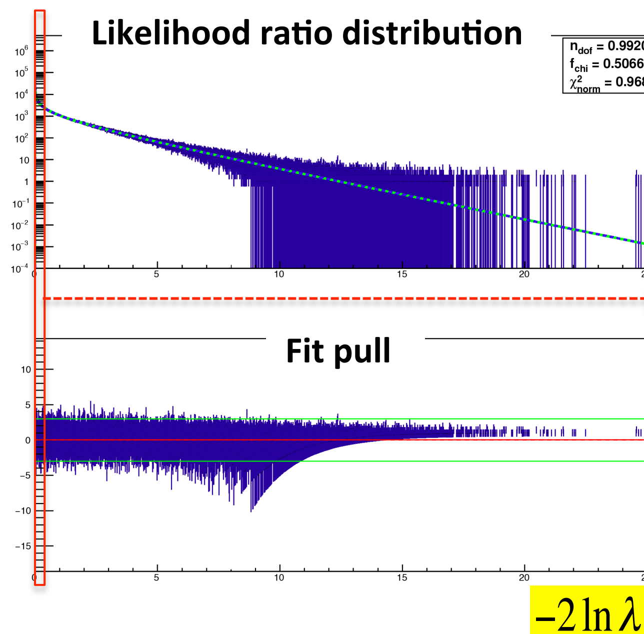
$$g(q_0|\mu=0) = \frac{1}{2}\delta(q_0) + \frac{1}{2}\left[\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{q_0}}e^{-q_0/2}\right]$$

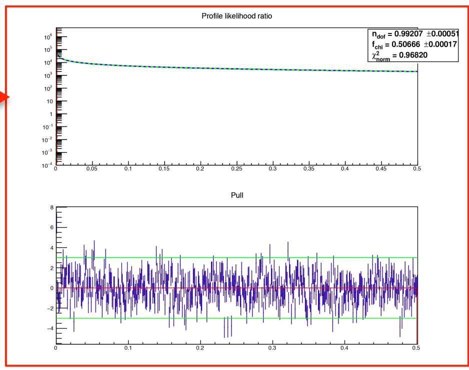
- **Let us fix the m & Γ parameters (to the CMS estimates from fit to data) while constraining $\mu \geq 0$ in our ML fits (μ represents a signal yield here).**

By fitting our likelihood ratio distrib. we indeed get:

$$\text{d.o.f.} \approx 0.992 \pm 0.001$$

$$\text{weight } C_{\chi^2} \approx 0.507 \pm 0.01$$

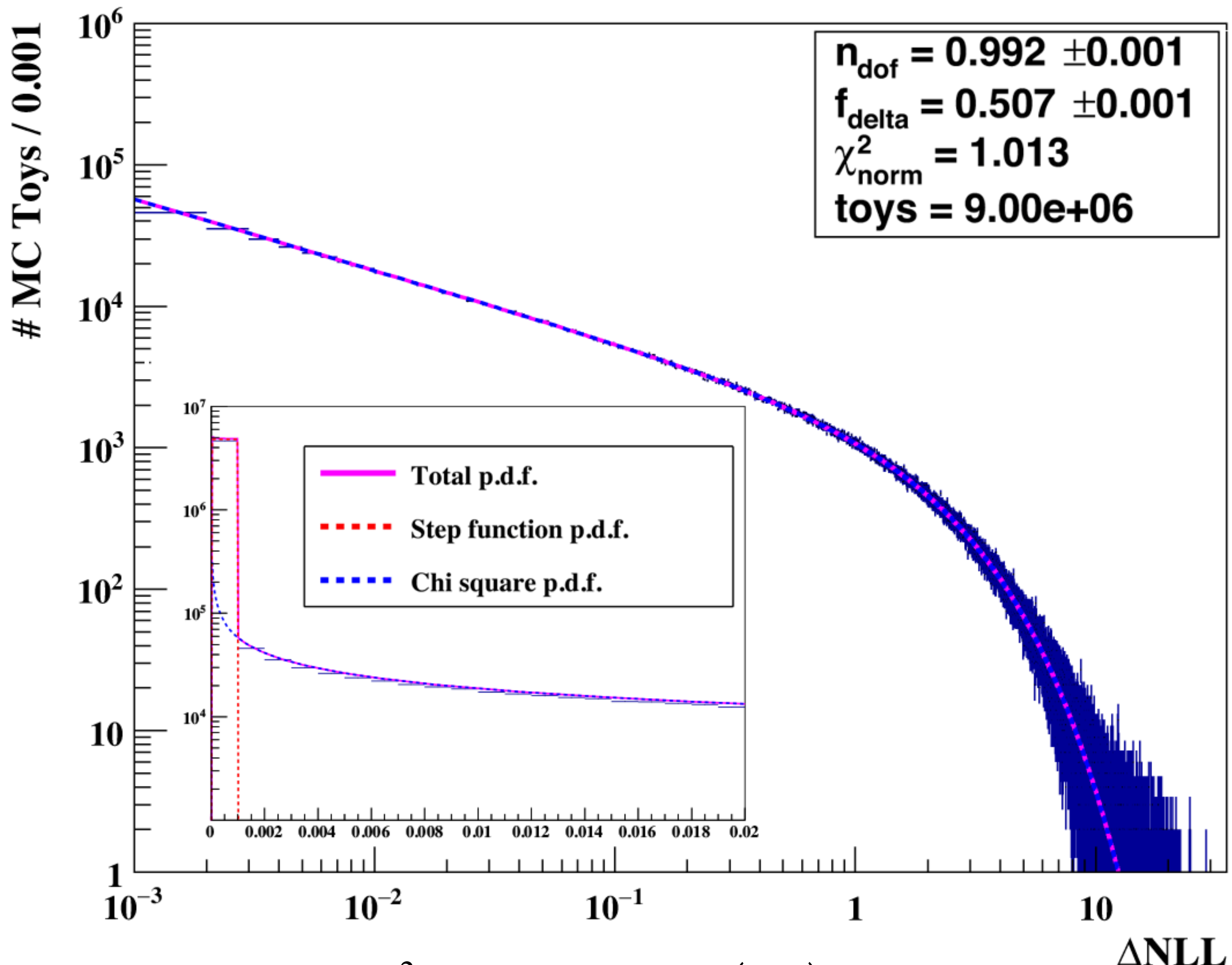




[*] Cowan *et al.*, EPJ C71 (2011) 1554

Special case: asymptotic formula by Cowan *et al.* [*] holds

- The quality of the previous fit (with a χ^2 pdf + a very narrow step function at 0) is **good enough**:



$$\chi^2_{norm} = 1.013 \quad P(fit) = 0.035$$

Summary & Outlook

Summary

- In order to test the **computing capabilities of GPUs** with respect to traditional CPU cores, a high-statistics toy Monte Carlo technique has been implemented both in *ROOT/RooFit* and *GooFit* frameworks with the purpose to estimate the **local statistical significance** of a - possibly exotic charmonium-like - signal recently confirmed by CMS (it was firstly observed by CDF).

The optimized *GooFit* applications running, by means of the MPS, on GPUs, hosted by the servers used in the presented test, provides a striking speed-up performance with respect to the *RooFit* application parallelized on multiple CPUs by means of *PROOF-Lite*.

-

Summary

- In order to test the **computing capabilities of GPUs** with respect to traditional CPU cores, a high-statistics toy Monte Carlo technique has been implemented both in *ROOT/RooFit* and *GooFit* frameworks with the purpose to estimate the **local statistical significance** of a - possibly exotic charmonium-like - signal recently confirmed by CMS (it was firstly observed by CDF).

The optimized *GooFit* applications running, by means of the MPS, on GPUs, hosted by the servers used in the presented test, provides a striking speed-up performance with respect to the *RooFit* application parallelized on multiple CPUs by means of *PROOF-Lite*.

- By means of *GooFit* it has also been easier to explore the (asymptotic) behaviour of a likelihood ratio test statistic in different situations in which **the Wilks Theorem may apply or does not apply because its regularity conditions are not satisfied**.

Outlook

- **The presented method can be extended to situations with a new unexpected signal, where a global statistical significance must be estimated.**
  To include properly the Look-Elsewhere-Effect a sort of scanning technique of the relevant mass spectra needs to be implemented.

Outlook

- **The presented method can be extended to situations with a new unexpected signal, where a global statistical significance must be estimated.**
To include properly the Look-Elsewhere-Effect a sort of scanning technique of the relevant mass spectra needs to be implemented.

This can certainly either …
- increase the execution time of the fits to be performed on the single fluctuation, and…
- require to try different scan models (and repeat the whole procedure) in order to evaluate the associated systematic uncertainty.

In this situation:
- the *RooFit* approach would be unbearable (highly time-consuming!),
- turning to GPUs would be mandatory,
- *GooFit* would be the reliable & crucial tool.

If you are interested to start learning and working with *GooFit* …

1) you can take the tutorial by R.Andreassen: http://indico.cern.ch/conferenceDisplay.py?confId=235992

2) *GooFit* source code lives in a GitHub repository: https://github.com/GooFit

3) you may want to exchange useful feedbacks on the *GooFit* Google Group.

Thank you for your attention

Let me thank in particular:

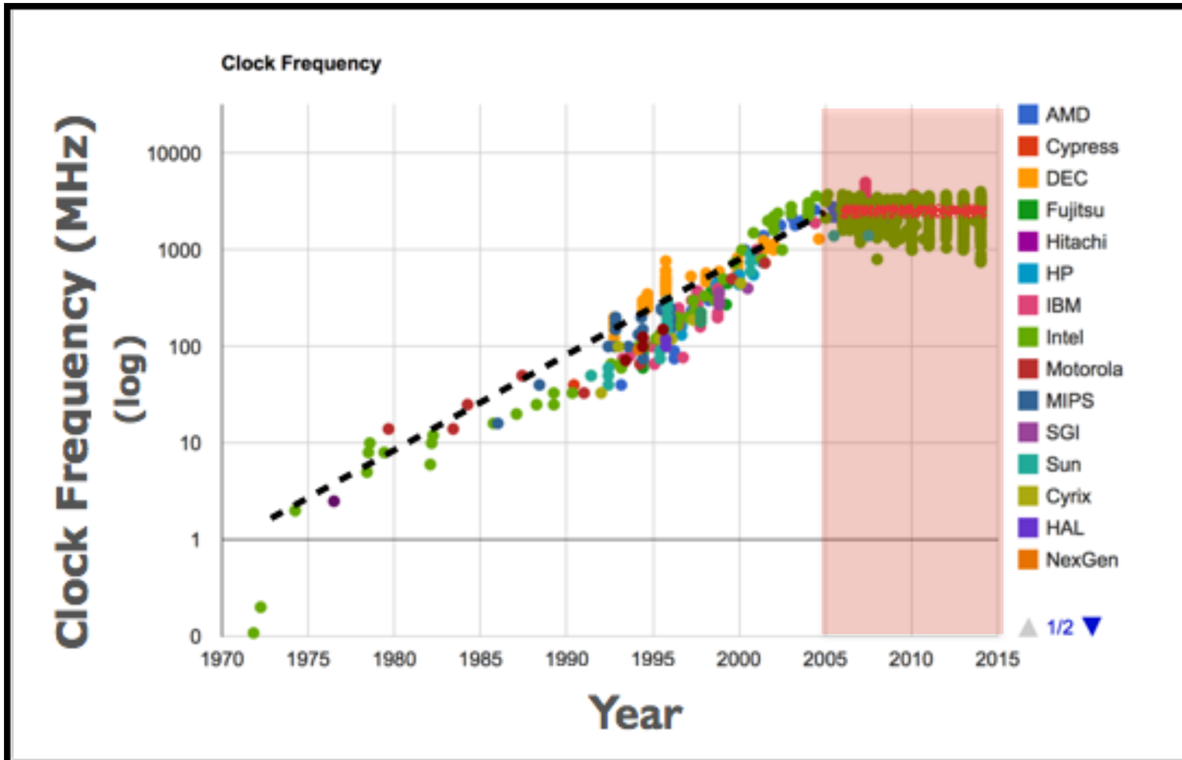
- **my supervisor of CMS-Bari Alexis Pompili (University of Bari & INFN), Adriano Di Florio (University of Bari & INFN), Giacinto Donvito (INFN-Bari, Tier2 manager) and the support by Italian Project *20108T4XTM* - MIUR PRIN 2010-2011 - *STOA-LHC***

- **Mike Sokoloff (University of Cincinnati) coordinator of the *GooFit* project funded by NSF** (NSF-1414736 Enabling HEP at the Information Frontier Using GPUs and Other Many/Multi-Core Architectures)

- **Brad Hittle (Ohio Supercomputer Center) and Tommaso Dorigo (INFN-Padova)**

BACKUP

***Moore's Law** :"the number of transistors per unit area would double approximately every two years"*



Stanford CPU Database - *cpudb.stanford.edu*

Physical limit: **heat dissipation**

$$P = C \times V \times f^2$$

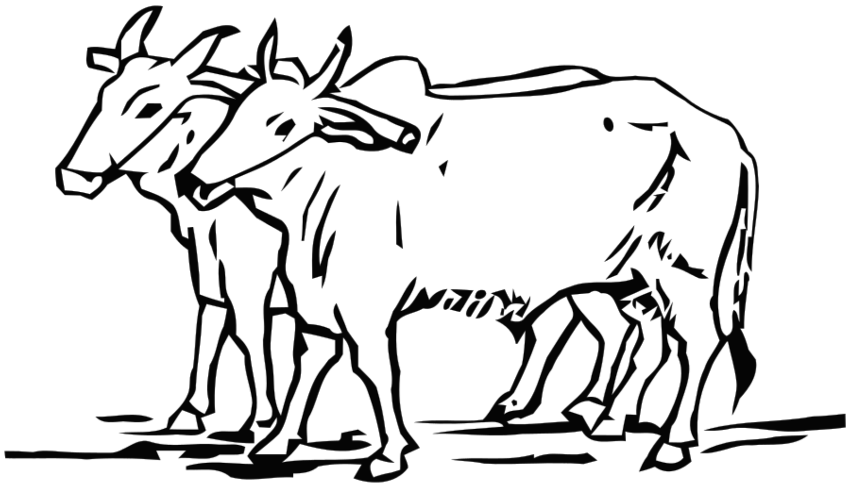
V – working tension
C – capacity
f – clock frequency

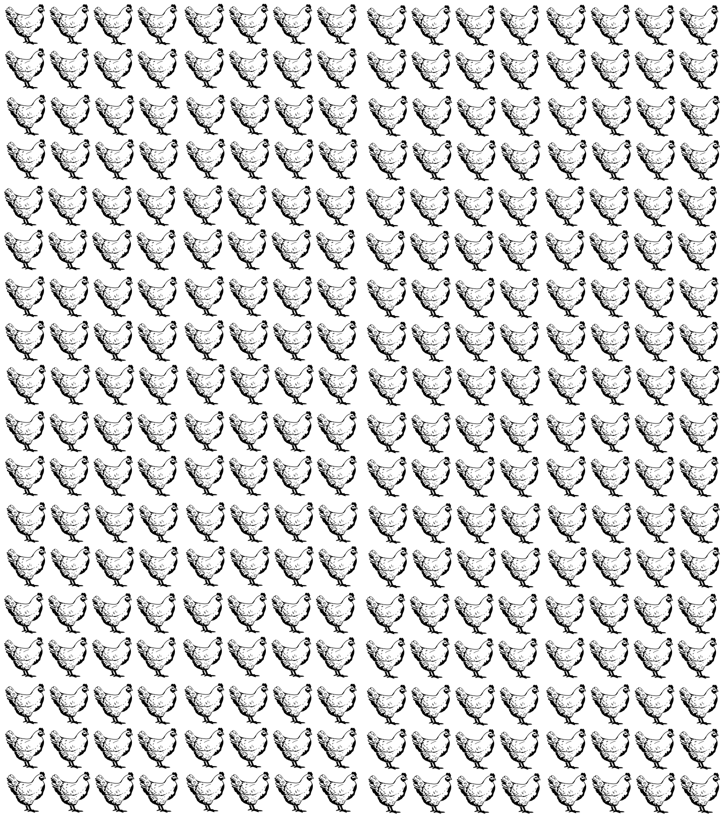
Future developments **cannot** rely anymore on an **exponential growth of frequency**

A new approach is needed: a possible solution is ***GPU-computing*.**

"If you were plowing a field, which would you rather use:
Two strong oxen *or 1024 chickens?"*

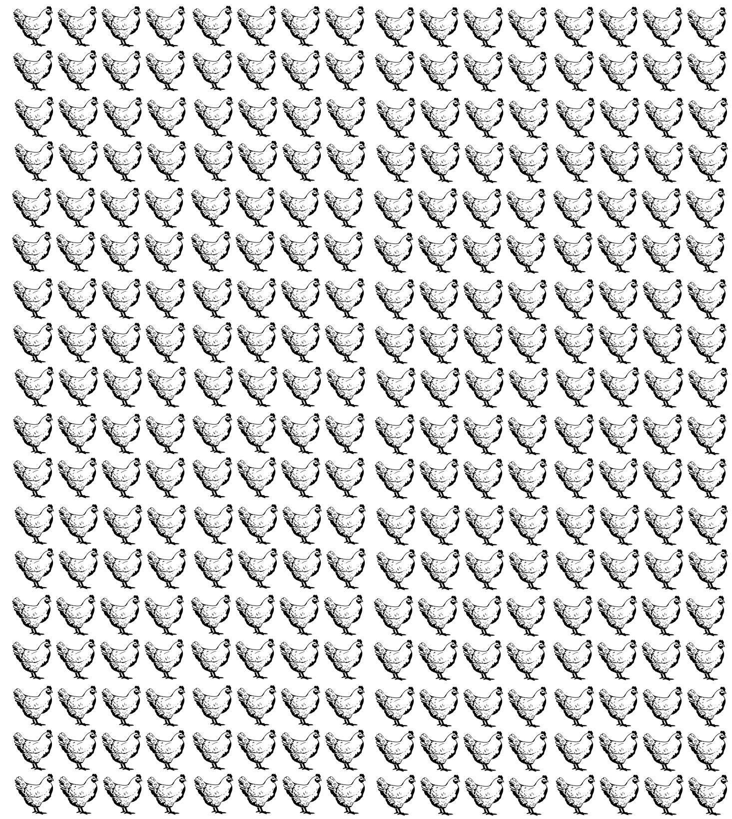
Seymour Cray





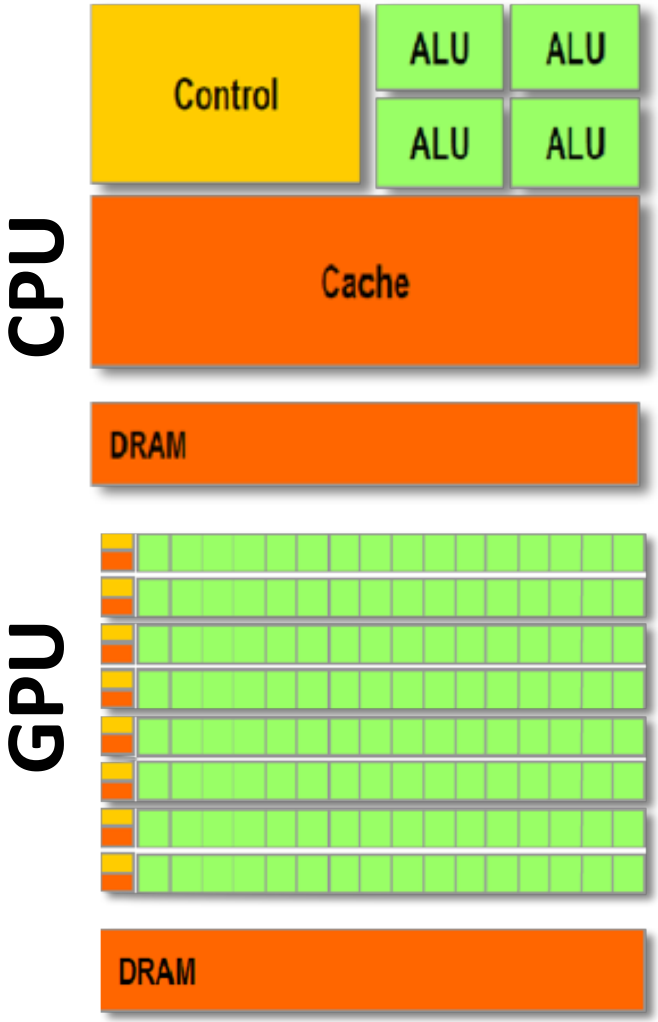
"If you were plowing a field, which would you rather use:
Two strong oxen *or 1024 chickens?"*

Seymour Cray

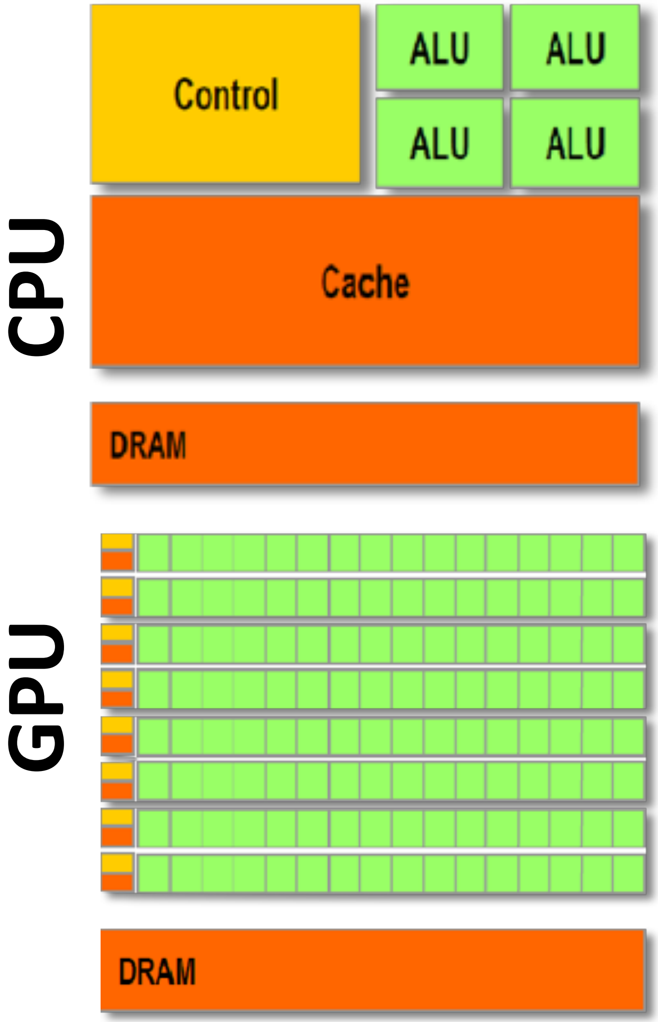


We definetely choose the chickens

What is a GPU? *Graphic Processing Unit*



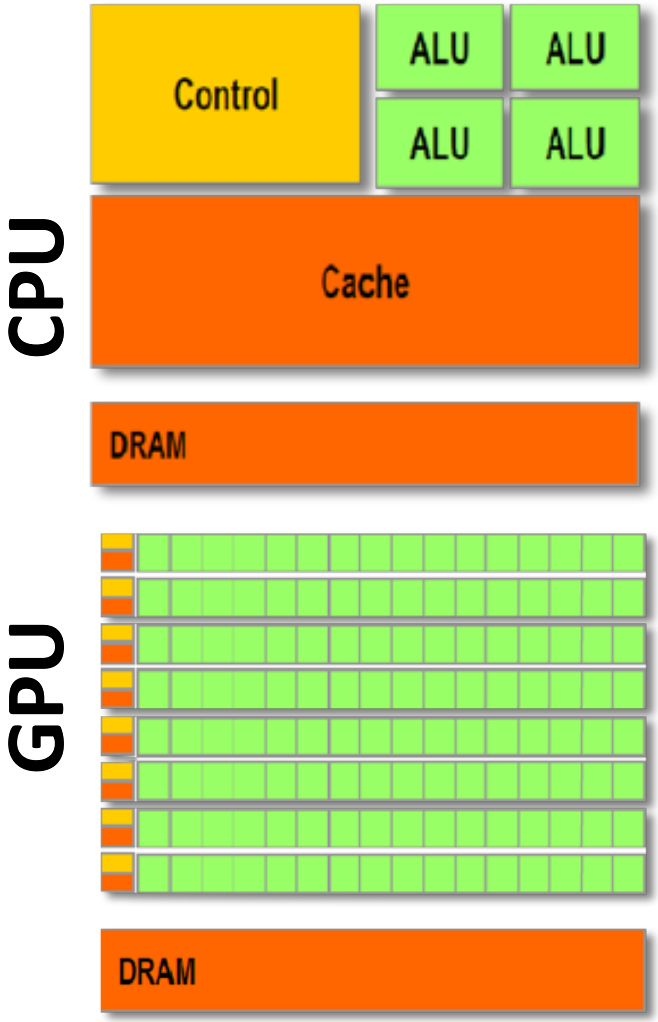
What is a GPU? *Graphic Processing Unit*



1970s: first graphical user interface produced requiring dedicated microchips

Video games and **3D graphics**: strong economic stimulus for GPU development

What is a GPU? *Graphic Processing Unit*



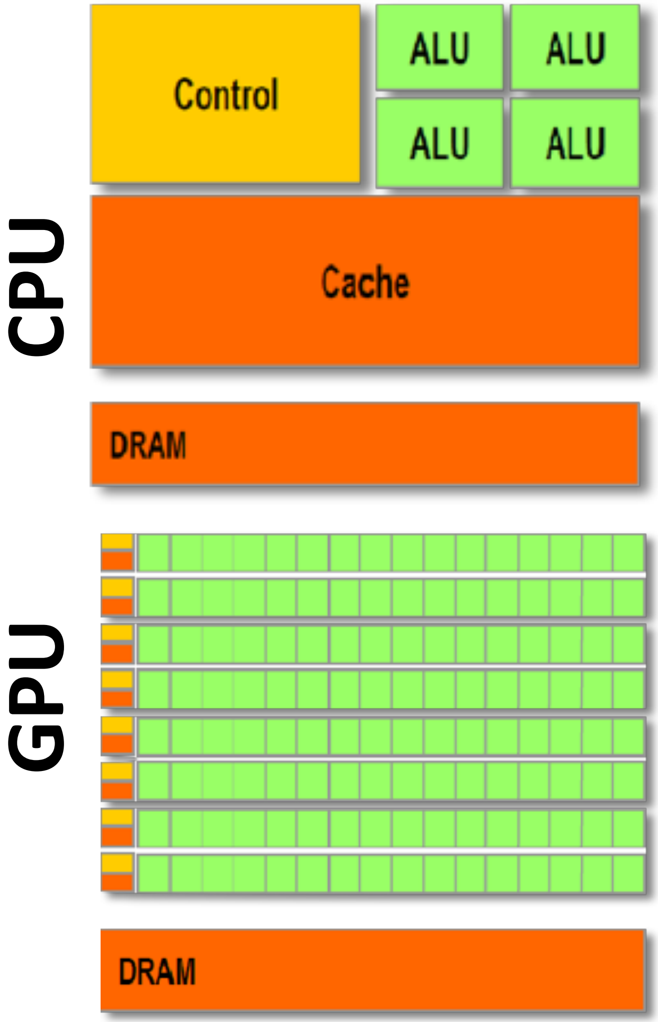
1970s: first graphical user interface produced requiring dedicated microchips

Video games and **3D graphics**: strong economic stimulus for GPU development

Consequences on GPU architecture:

Thousands of cores

What is a GPU? *Graphic Processing Unit*



1970s: first graphical user interface produced requiring dedicated microchips

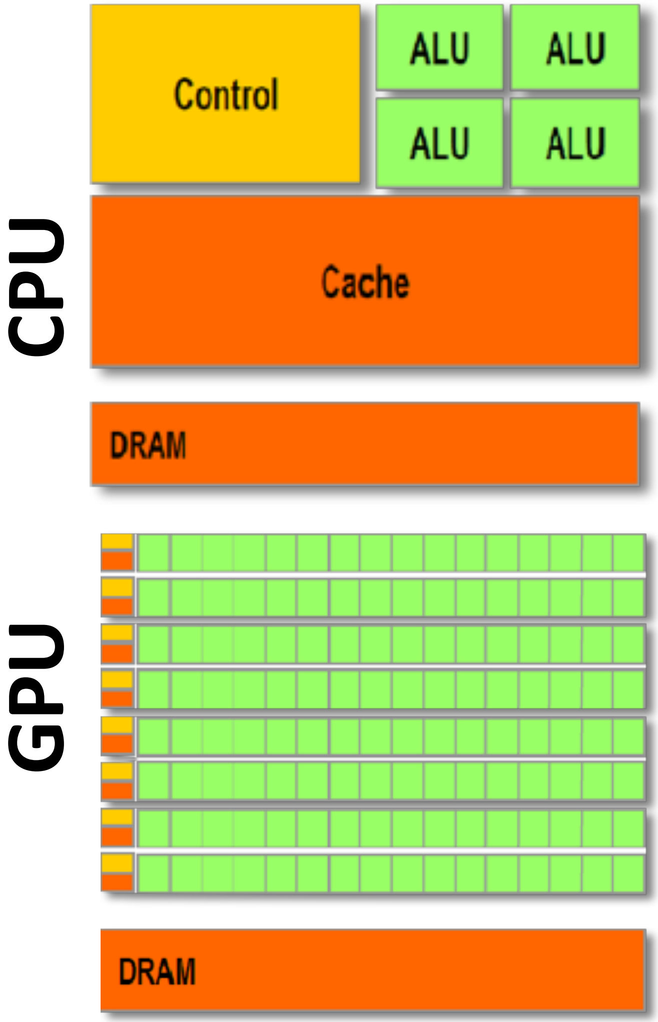
Video games and **3D graphics**: strong economic stimulus for GPU development

Consequences on GPU architecture:

Thousands of cores

Big loads of data

What is a GPU? *Graphic Processing Unit*



1970s: first graphical user interface produced requiring dedicated microchips

Video games and **3D graphics**: strong economic stimulus for GPU development

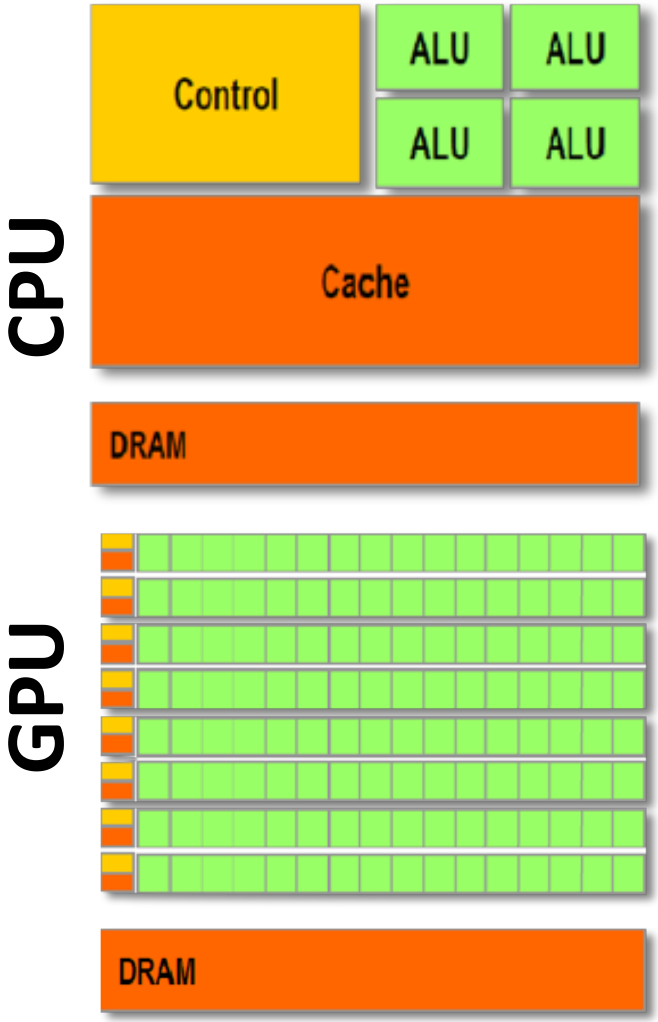
Consequences on GPU architecture:

Thousands of cores

Big loads of data

Low frequency clock (~1GHz)

What is a GPU? *Graphic Processing Unit*



1970s: first graphical user interface produced requiring dedicated microchips

Video games and **3D graphics**: strong economic stimulus for GPU development

Consequences on GPU architecture:

Thousands of cores

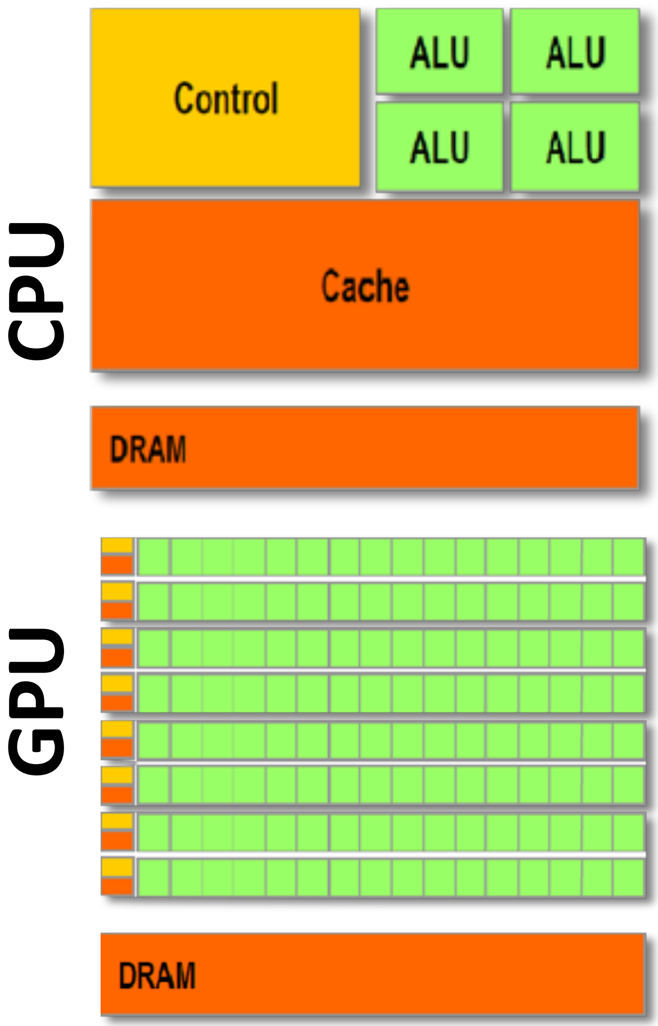
Big loads of data

Low frequency clock (~1GHz)

Arithmetical operations in a single clock cycle (*sin,cos,sqrt,1/x, …)*

GPUs' architecture

What is a GPU? *Graphic Processing Unit*



1970s: first graphical user interface produced requiring dedicated microchips

Video games and **3D graphics**: strong economic stimulus for GPU development

Consequences on GPU architecture:

Thousands of cores

Big loads of data

Low frequency clock (~1GHz)

Arithmetical operations in a single clock cycle (*sin,cos,sqrt,1/x, …*)

x2

GPU

Numero di GPU

Numero di CUDA cores

Memoria per GPU (GDDR5)

Banda di memoria per board

CPU

- 16 cores : E5-2640 v2 @ 2.00GHz (32 con HT)
- 64 GB RAM

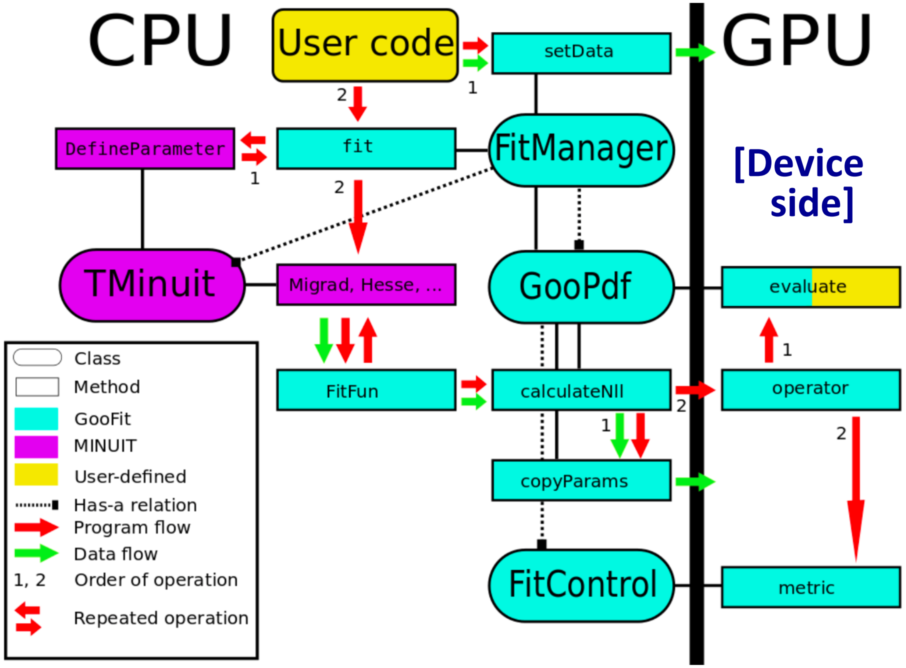
Memoria per GPU (GDDR5)

Banda di memoria per board

GooFit framework

GooFit is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.

Control & Data Flow of a GooFit program



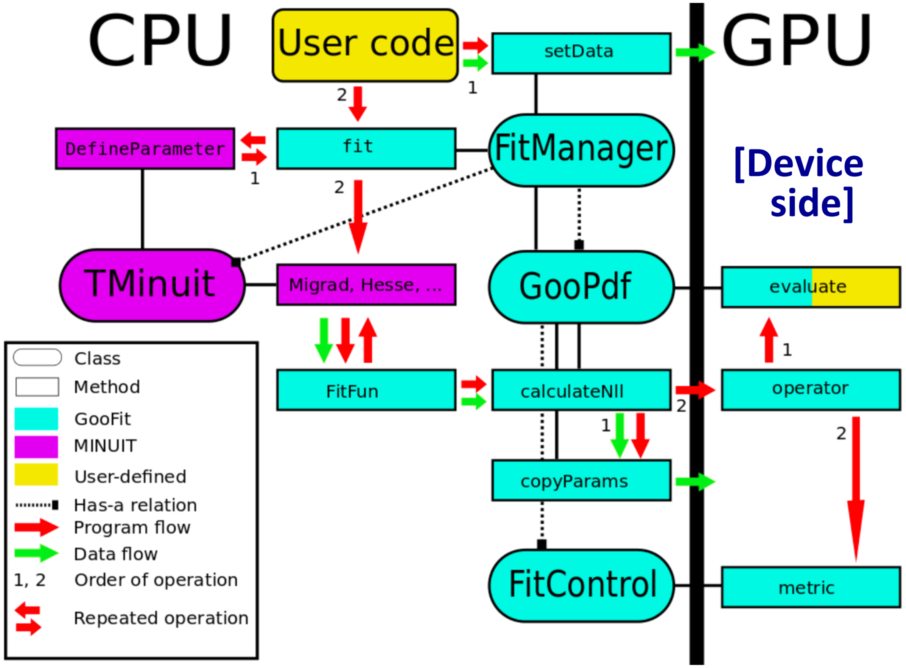
GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

It is an open source project, under development and funded by US NSF.

GooFit **is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.**

- **A *GooFit* program has 4 main components:**

Control & Data Flow of a GooFit program



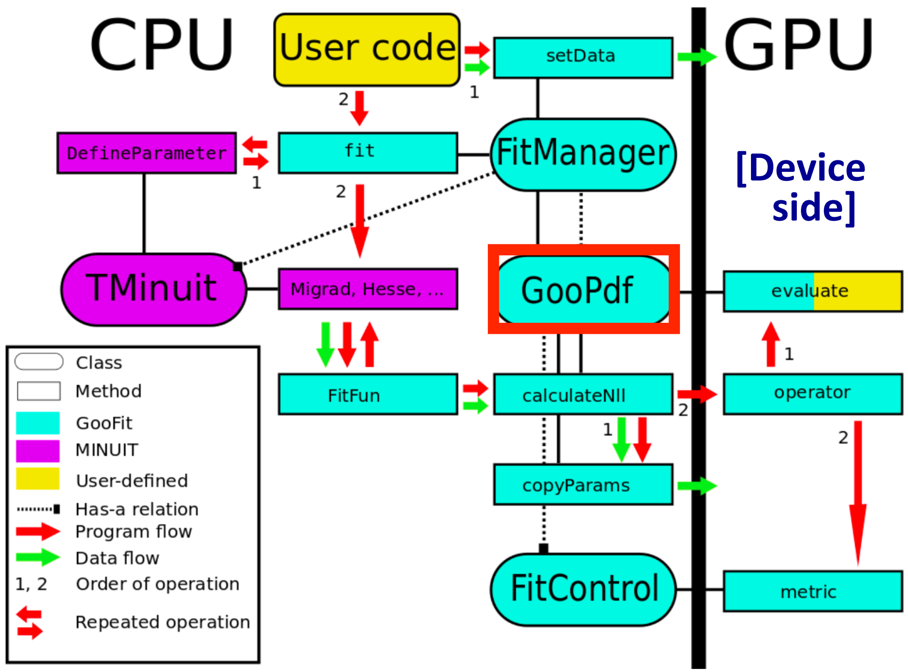
GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

It is an open source project, under development and funded by US NSF.

GooFit **is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.**

- **A *GooFit* program has 4 main components:**
  - **a `GooPdf` object representing the PDF modelling the physical process**

Control & Data Flow of a GooFit program



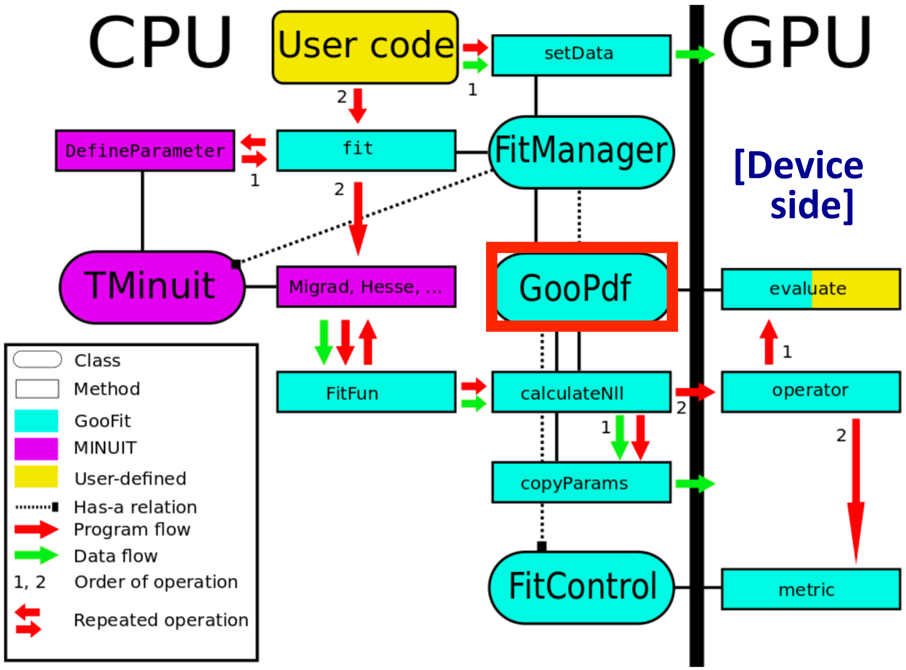
GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

It is an open source project, under development and funded by US NSF.

GooFit **is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.**

- **A *GooFit* program has 4 main components:**
  - **a `GooPdf` object representing the PDF modelling the physical process**
  - **the fit parameters (`Variables` objects contained in the `GooPdf`)**
  - **the data (`DataSet` object)**

Control & Data Flow of a GooFit program



GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

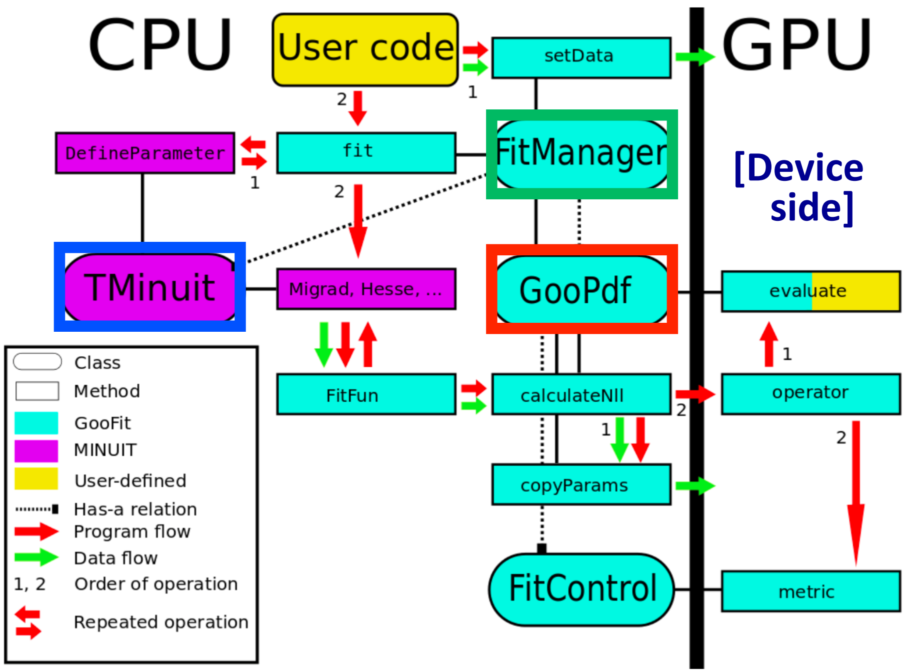
It is an open source project, under development and funded by US NSF.

GooFit framework

GooFit is a data analysis tool for HEP, that interfaces ROOT/RooFit to CUDA parallel computing platform on *nVidia* GPU. It also supports OpenMP.

- A *GooFit* program has 4 main components:
  - a **`GooPdf`** object representing the PDF modelling the physical process
  - the fit parameters (**`Variables`** objects contained in the **`GooPdf`**)
  - the data (**`DataSet`** object)
  - a **`FitManager`** object forming the interface between MINUIT and the **`GooPdf`**

Control & Data Flow of a GooFit program

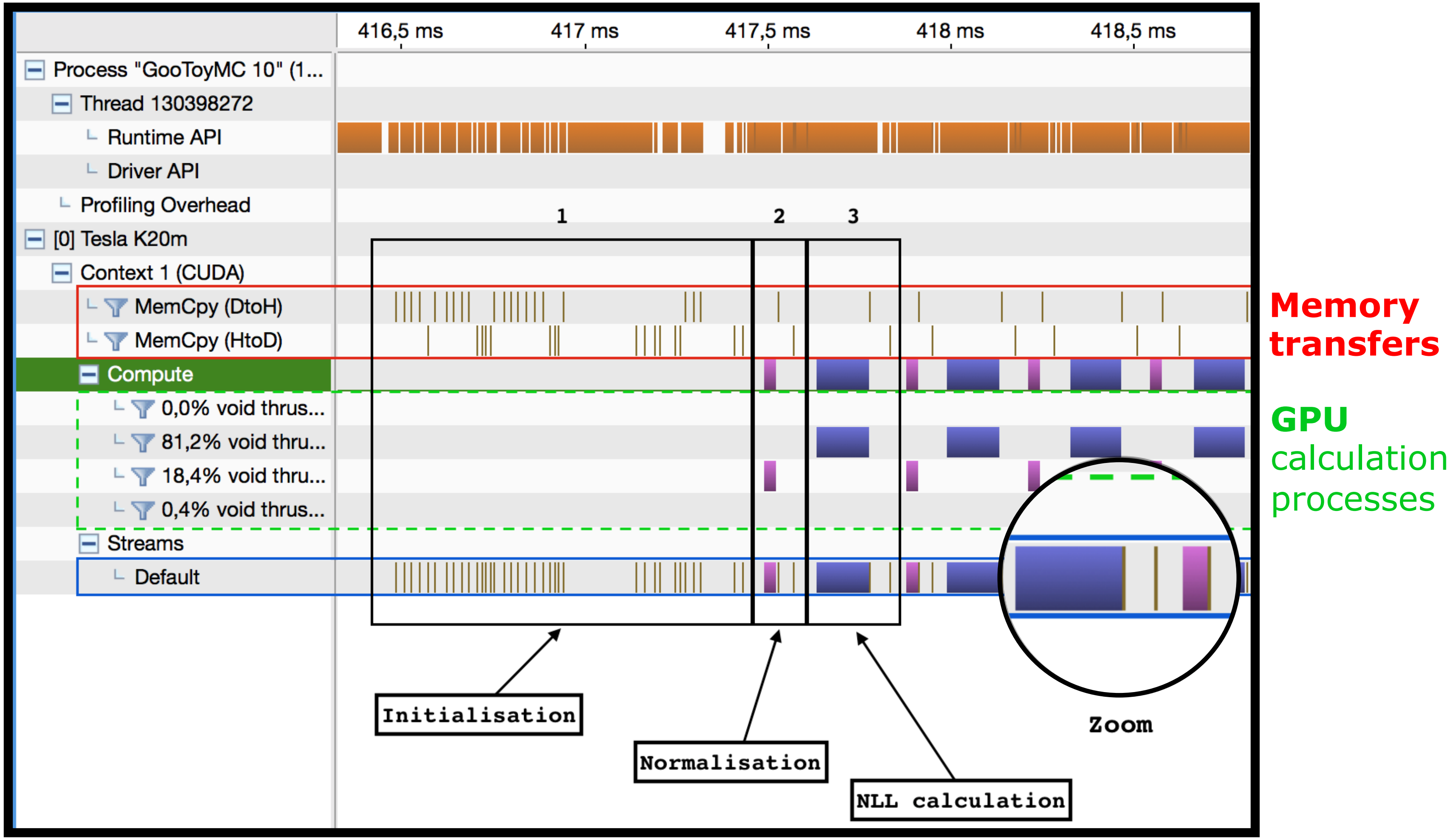


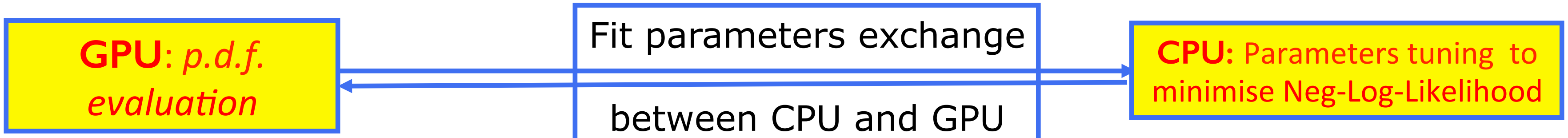
GooFit: a library for massively parallelising maximum-likelihood fits
R.Andreassen et al., *J.Phys.:Conf.Ser.* 513 (2014) 052003

It is an open source project, under development and funded by US NSF.

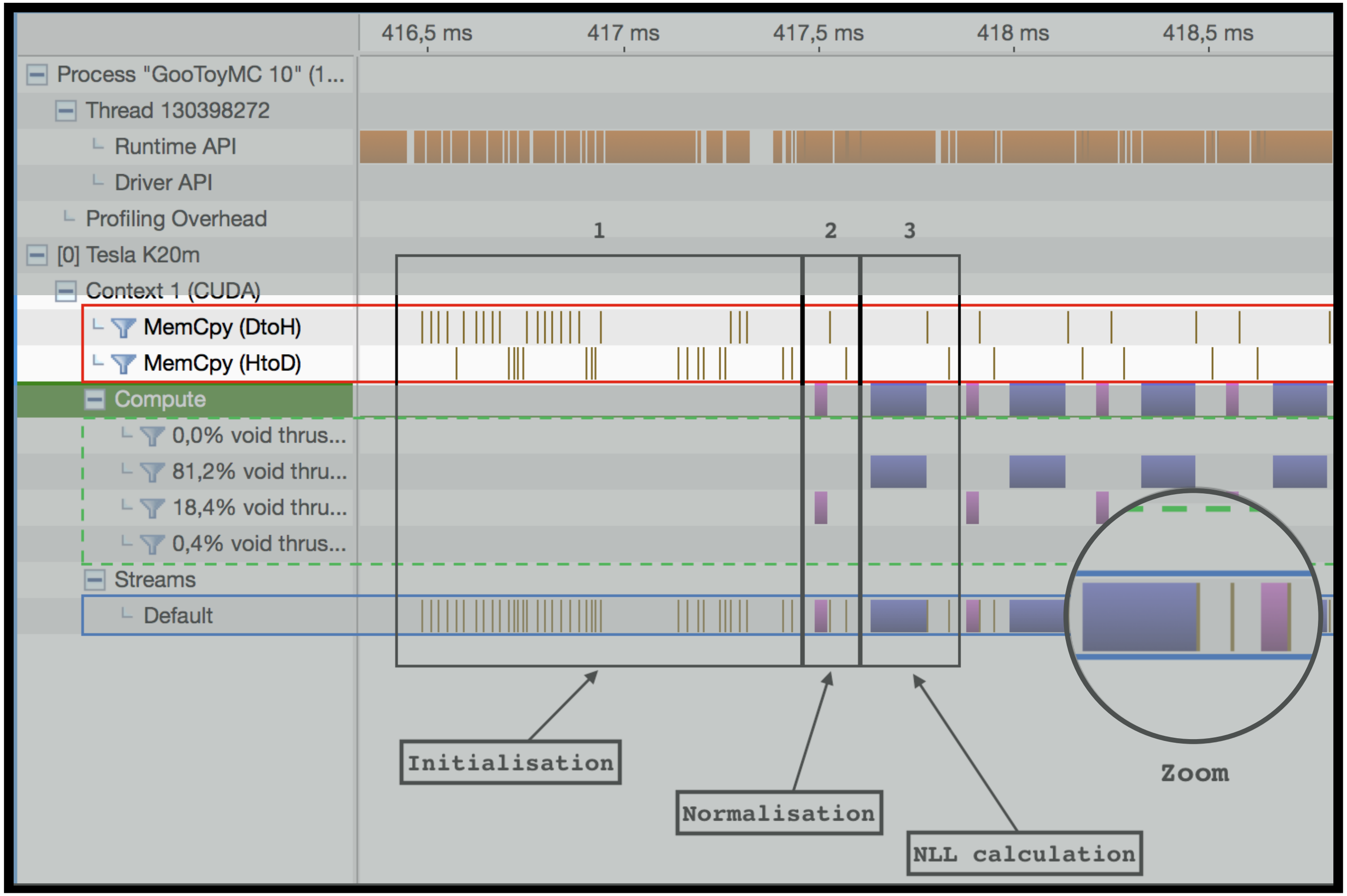
GooFit profiling

Example of a snapshot of the profile of a GooFit process provided by Nvidia Visual Profiler:





Example of a snapshot of the profile of a GooFit process provided by Nvidia Visual Profiler:



Memory transfers

GPU calculation processes

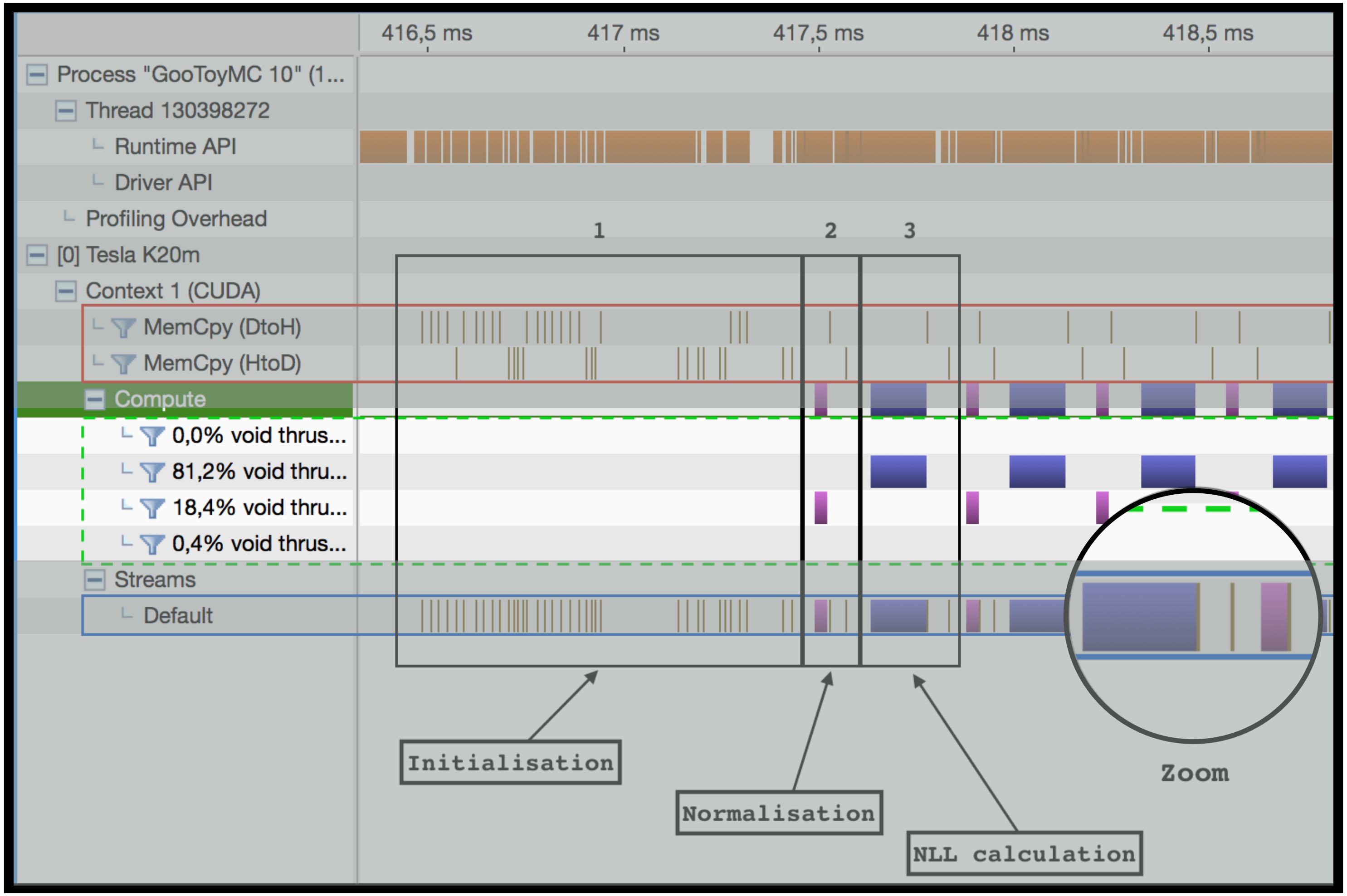
GPU: *p.d.f. evaluation*

Fit parameters exchange between CPU and GPU

CPU: Parameters tuning to minimise Neg-Log-Likelihood

GooFit profiling

Example of a snapshot of the profile of a GooFit process provided by Nvidia Visual Profiler:



GPU: *p.d.f. evaluation*

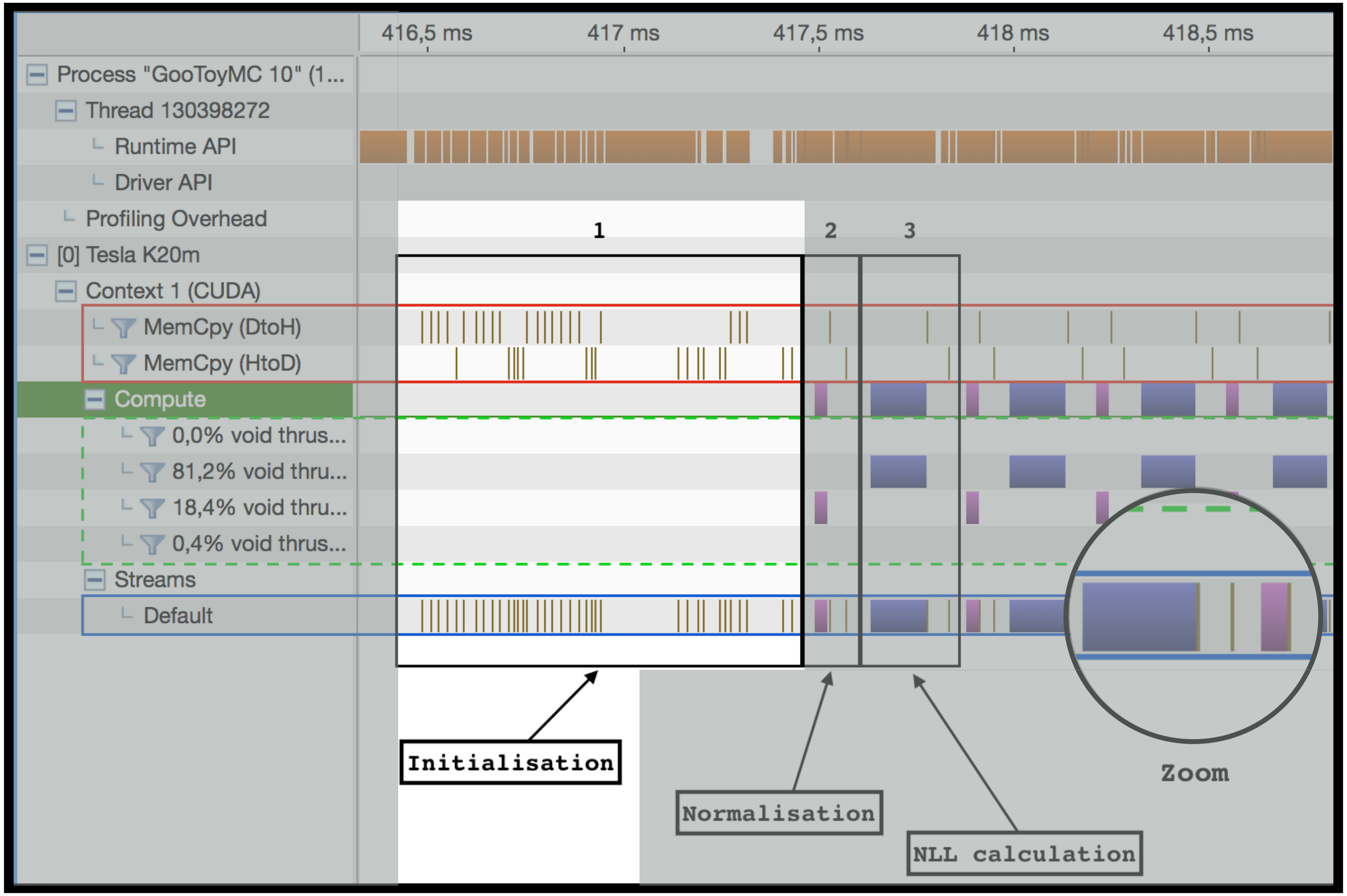
Fit parameters exchange

between CPU and GPU

CPU: Parameters tuning to minimise Neg-Log-Likelihood

GooFit profiling

Example of a snapshot of the profile of a GooFit process provided by Nvidia Visual Profiler:



Memory transfers

GPU calculation processes

GPU: *p.d.f. evaluation*

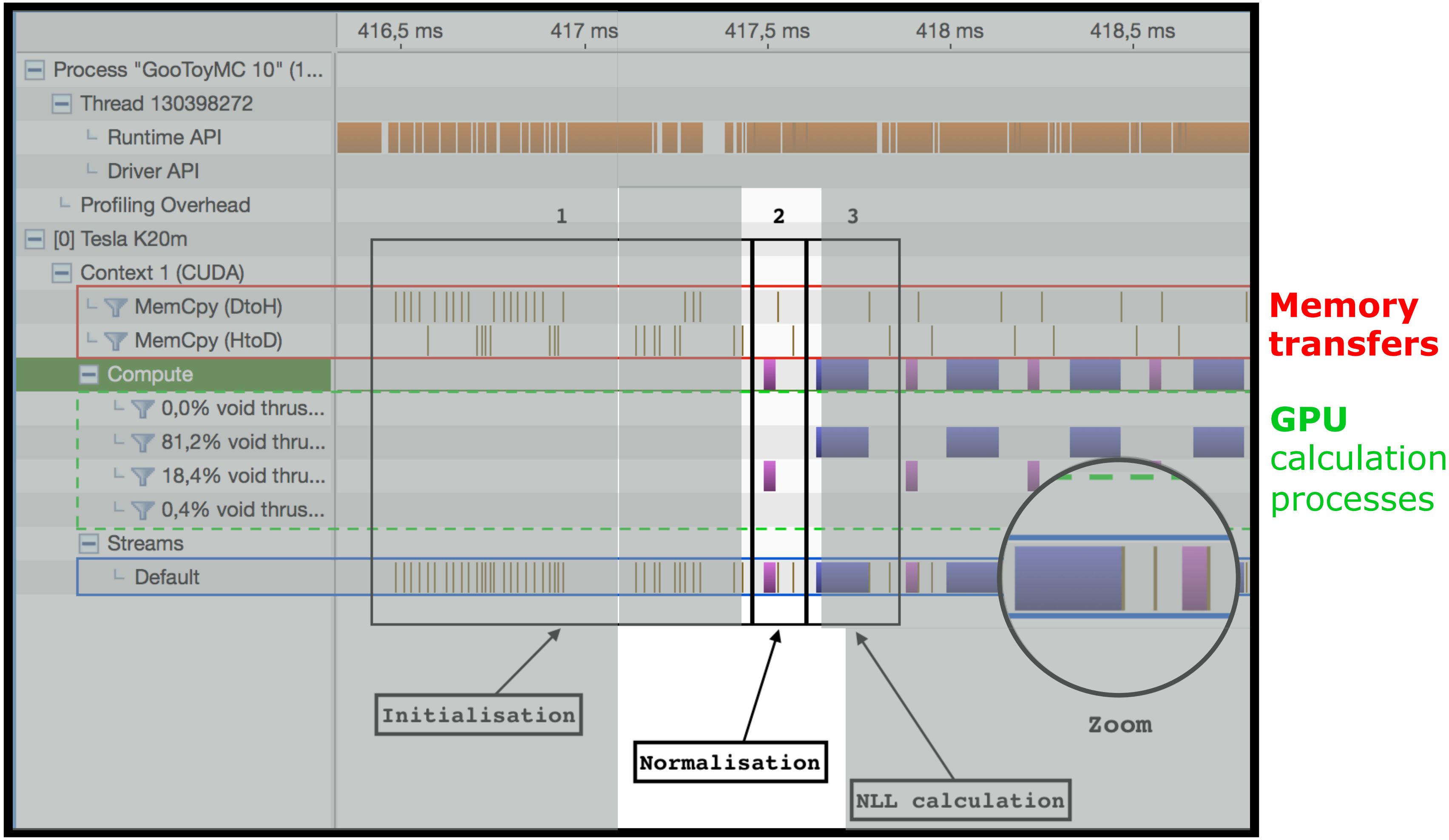
Fit parameters exchange

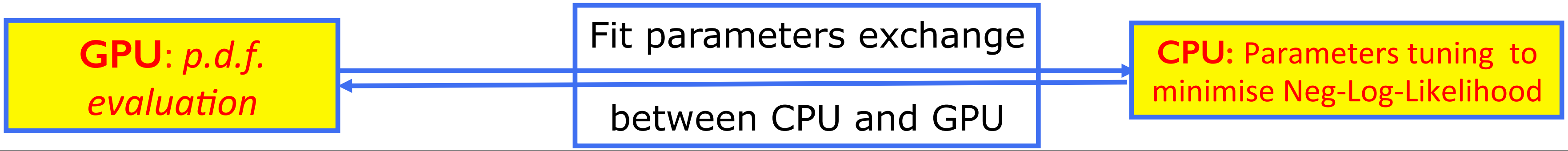
between CPU and GPU

CPU: Parameters tuning to minimise Neg-Log-Likelihood

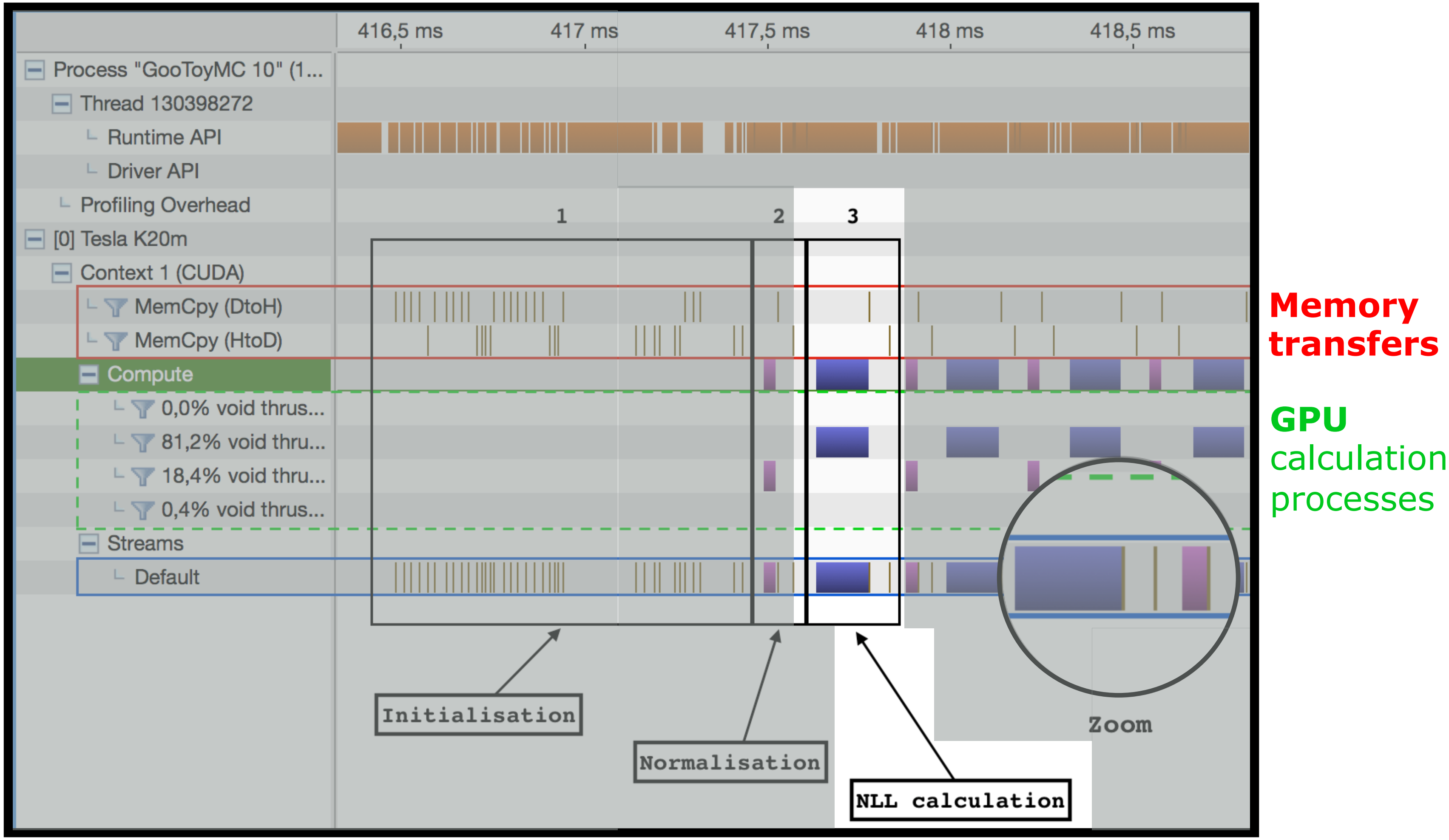
GooFit profiling

Example of a snapshot of the profile of a GooFit process provided by Nvidia Visual Profiler:





Example of a snapshot of the profile of a GooFit process provided by Nvidia Visual Profiler:



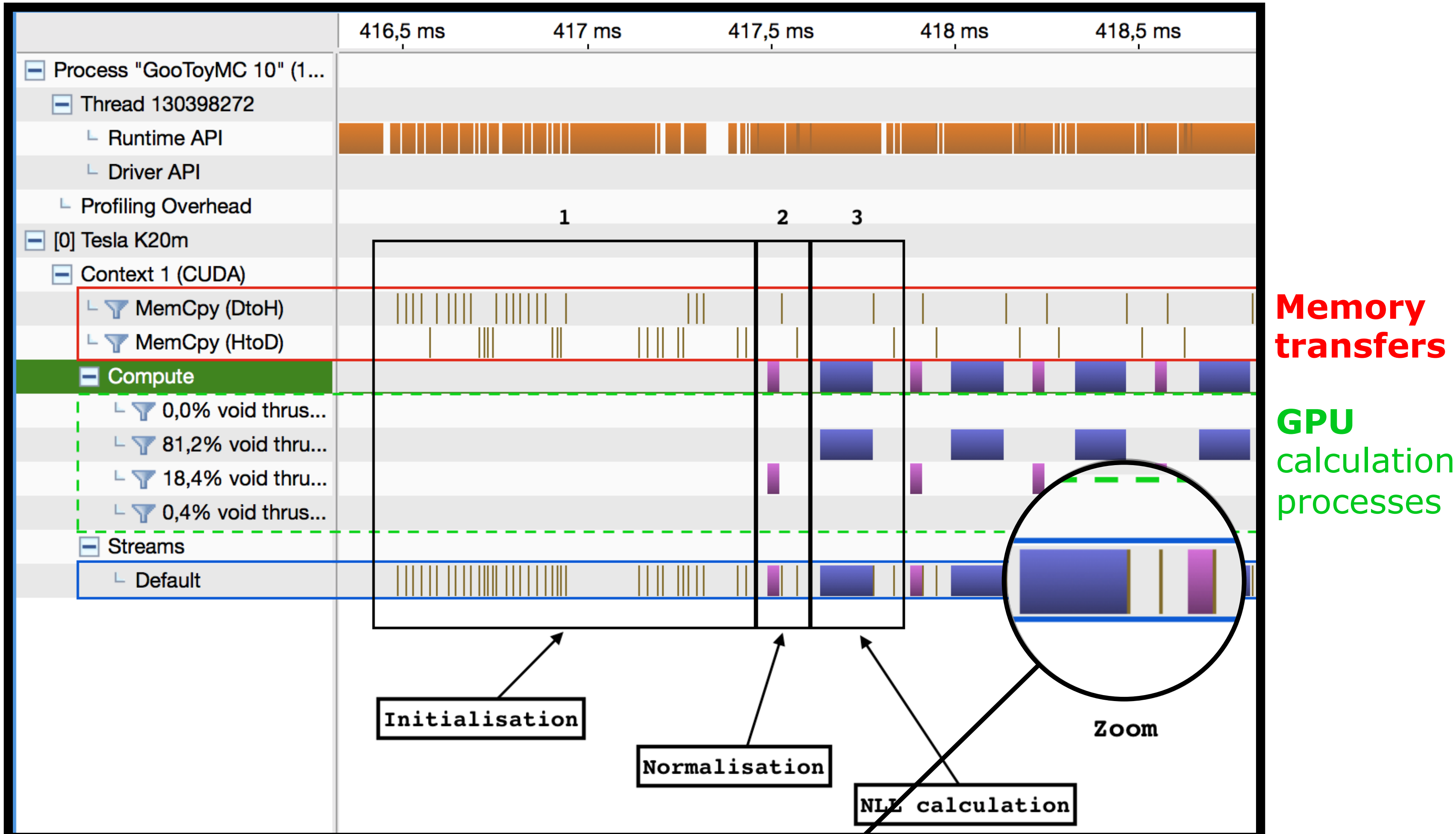
GPU: *p.d.f. evaluation*

Fit parameters exchange between CPU and GPU

CPU: Parameters tuning to minimise Neg-Log-Likelihood

GooFit profiling

Example of a snapshot of the profile of a GooFit process provided by Nvidia Visual Profiler:



GPU: *p.d.f. evaluation*

Fit parameters exchange

between CPU and GPU

CPU: Parameters tuning to minimise Neg-Log-Likelihood