


GPUs for Statistical Data Analysis in HEP : a performance study of *GooFit* on GPUs vs *RooFit* on CPUs



Alexis Pompili (on behalf of  Collaboration)



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



Outline

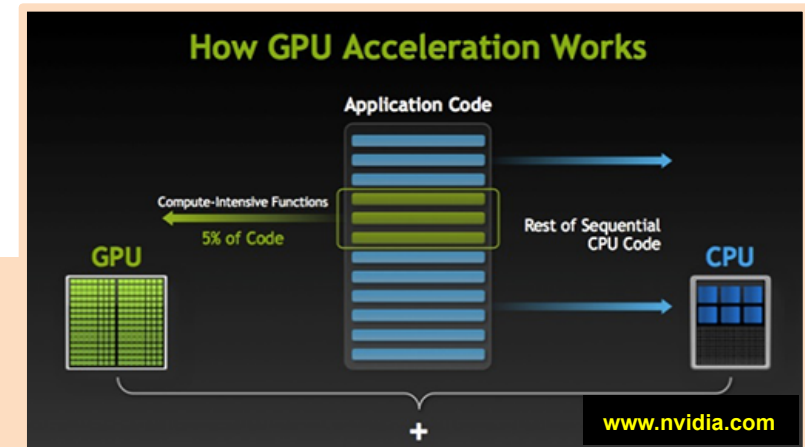
- Introduction to *GooFit* & its application to Unbinned Maximum Likelihood fitting
- Pseudo-experiments for p-value estimation : *GooFit* vs *RooFit* performance study
- Exploring the applicability limits of Wilks theorem
- Summary & Outlook

Introduction : *GooFit* on GPUs

Introduction to GPU-accelerated computing

➤ GPU-accelerated computing is the use of a Graphics Processing Unit to accelerate scientific applications (among other apps). Nowadays GPUs power efficient data-centers in labs, institutes and universities.

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



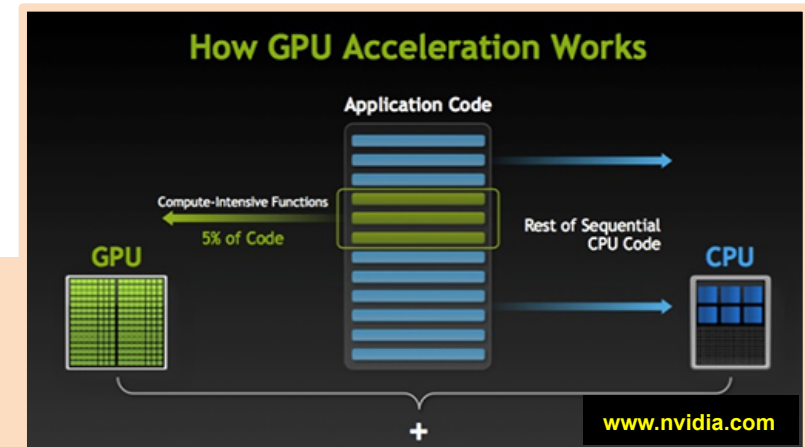
GPUs have been developed in the video recreational & 3D graphics sectors. They now can have **few thousands of cores** (having a **relatively low clock** $\sim 1\text{GHz}$) to process parallel workloads efficiently. Computing capabilities are enhanced once a sequence of elementary arithmetic operations need to be performed in parallel on an huge amount of data.



Introduction to GPU-accelerated computing

- GPU-accelerated computing is the use of a Graphics Processing Unit to accelerate scientific applications (among other apps). Nowadays GPUs power efficient data-centers in labs, institutes and universities.

Enhancement of application performances obtained by offloading compute-intensive portions to the GPU while the remaining code still runs on the CPUs.



GPUs have been developed in the video recreational & 3D graphics sectors. They now can have **few thousands of cores** (having a **relatively low clock $\sim 1\text{GHz}$**) to process parallel workloads efficiently. Computing capabilities are enhanced once a sequence of elementary arithmetic operations are performed in parallel on an huge amount of data.

- **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*.

What is *GooFit*? A **data analysis tool** for HEP analysts, that interfaces ROOT/RooFit to **CUDA** parallel computing platform on *nVidia* GPU. It also supports **OpenMP**.
It is an open source project, under development and funded by US NSF.

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 **P**robability **D**ensity **F**unction to be evaluated in parallel.



A preliminary example of *GooFit*/GPUs capabilities

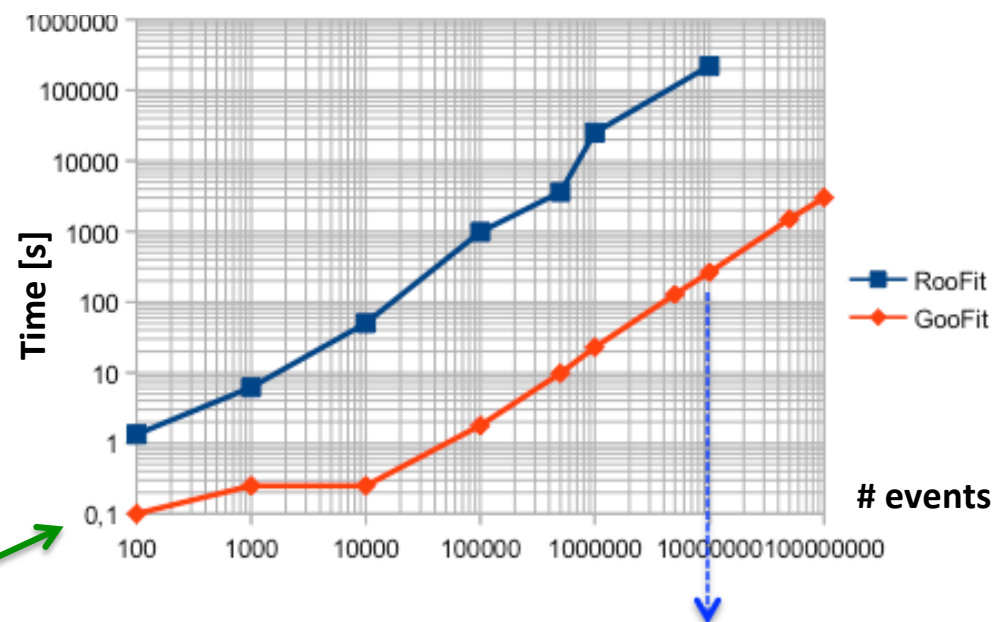
➤ Parameter estimation is a crucial part of many physics analyses.

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 **P**robability **D**ensity **F**unction 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**:



A preliminary example of *GooFit*/GPUs capabilities

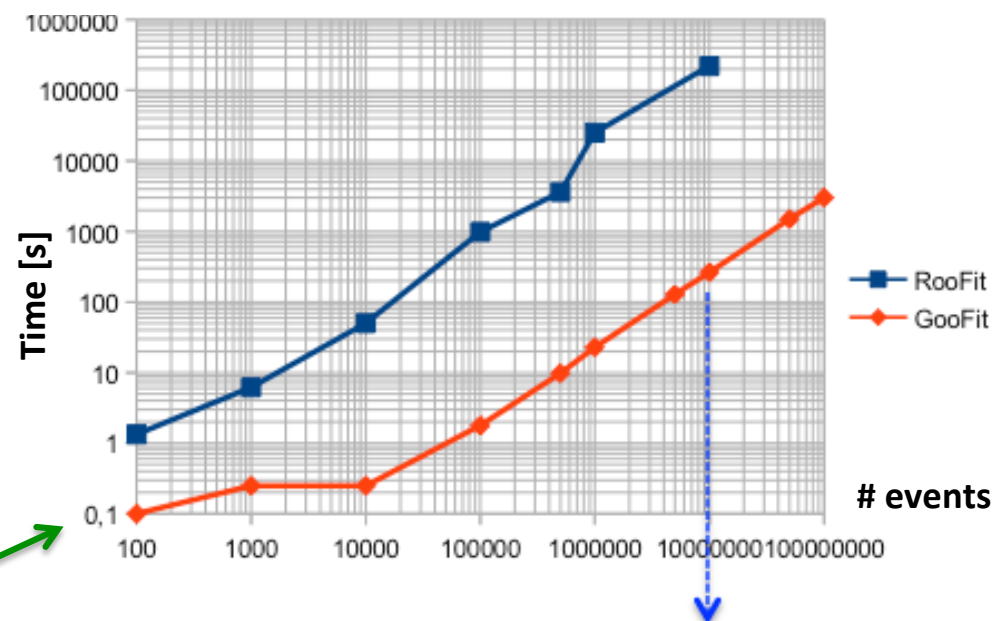
➤ Parameter estimation is a crucial part of many physics analyses.

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 **P**robability **D**ensity **F**unction 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!

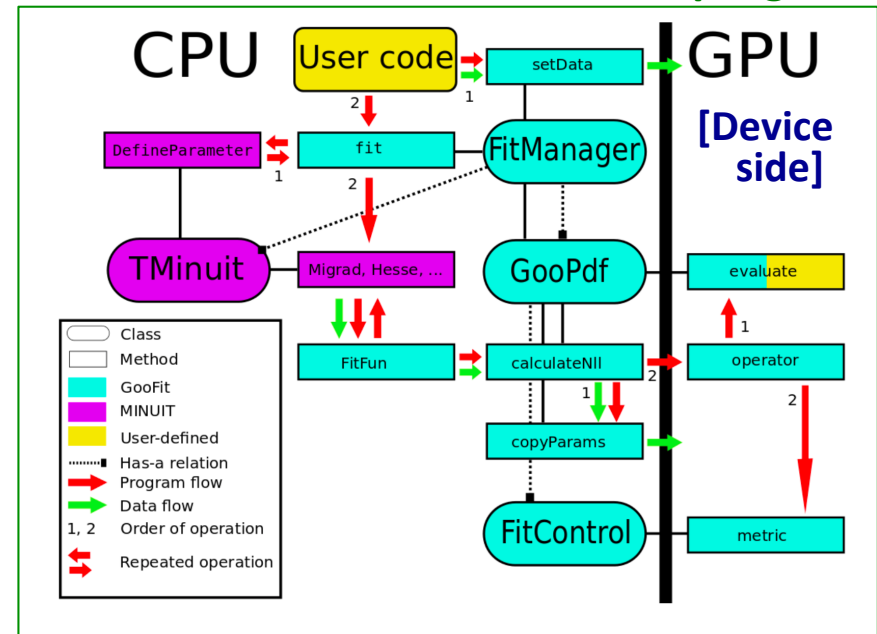


➤ As expected, for a **Binned ML fit**, the speed-up ranges from few units to few dozens (with #bins).

GooFit framework

- 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

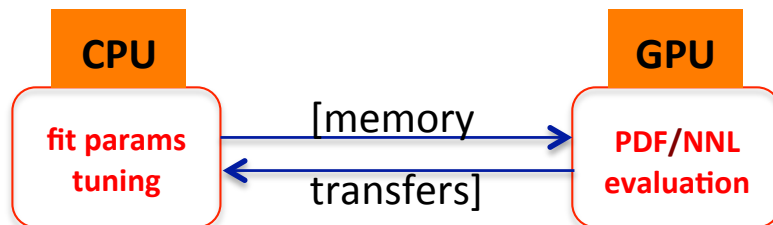
[few details in the backup]



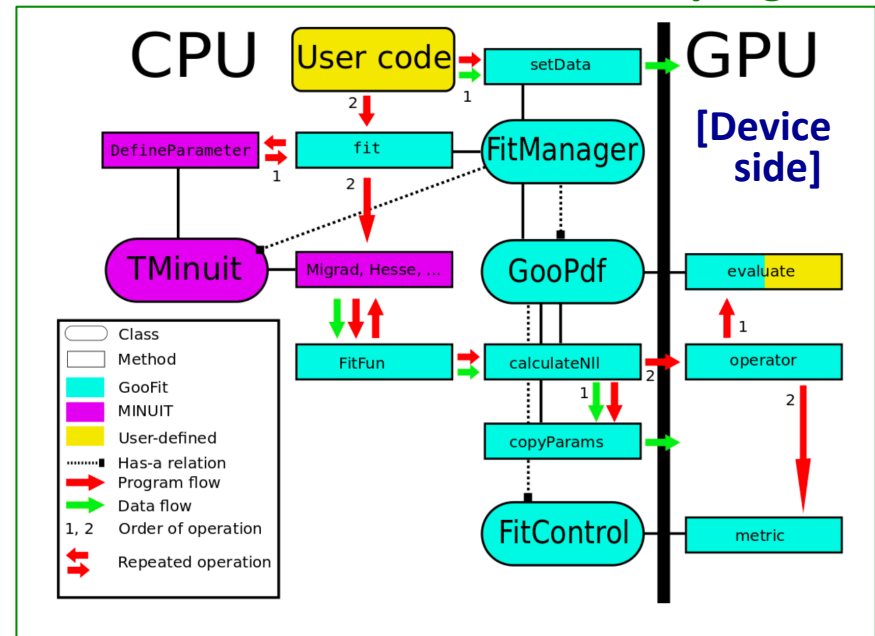
GooFit framework

- 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

[few details in the backup]

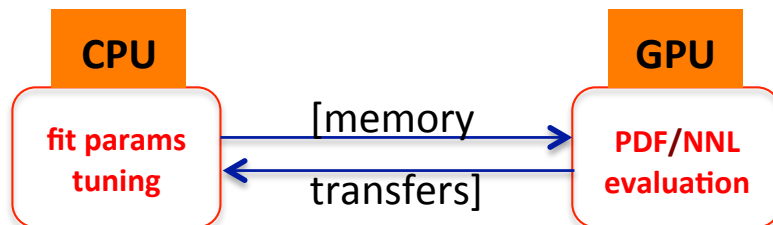
- This can be seen by analysing a cycle with the monitoring tool nVIDIA Visual Profiler [nvvp]



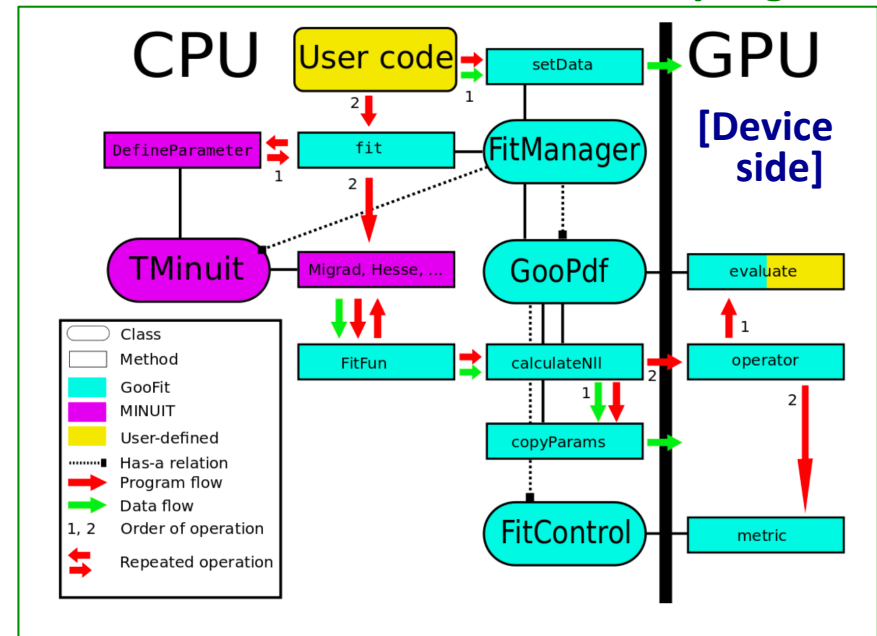
GooFit framework

- 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

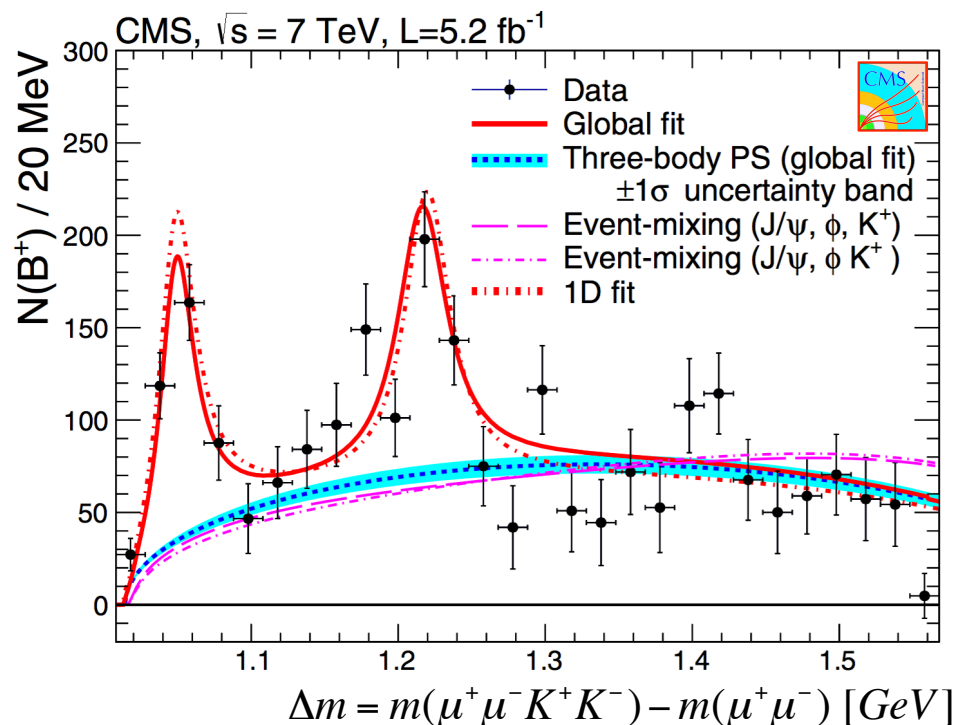
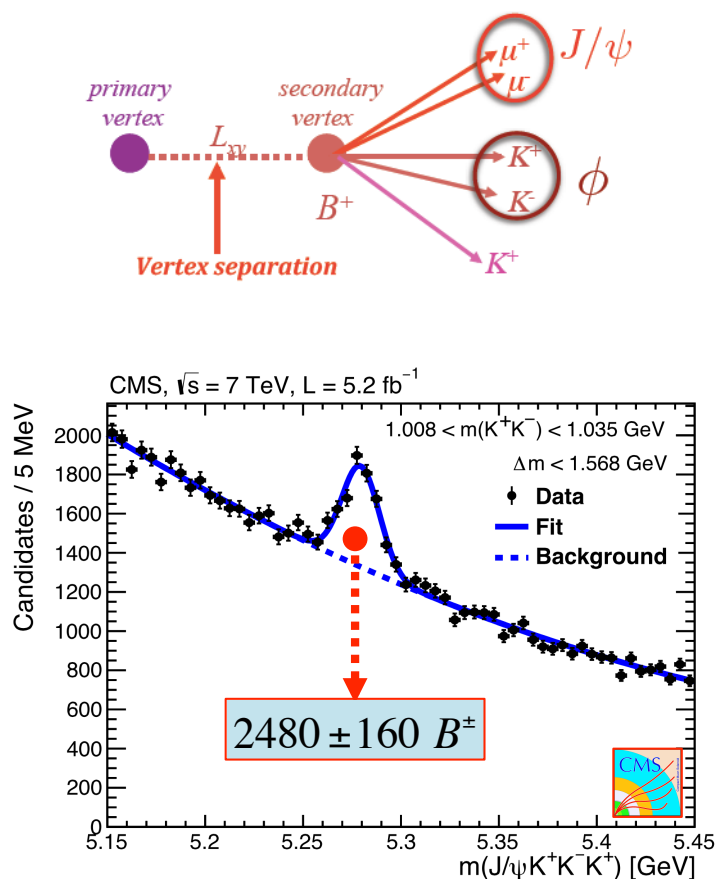
[few details in the backup]

- 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 & ML fits (either **unbinned** & **binned**).

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

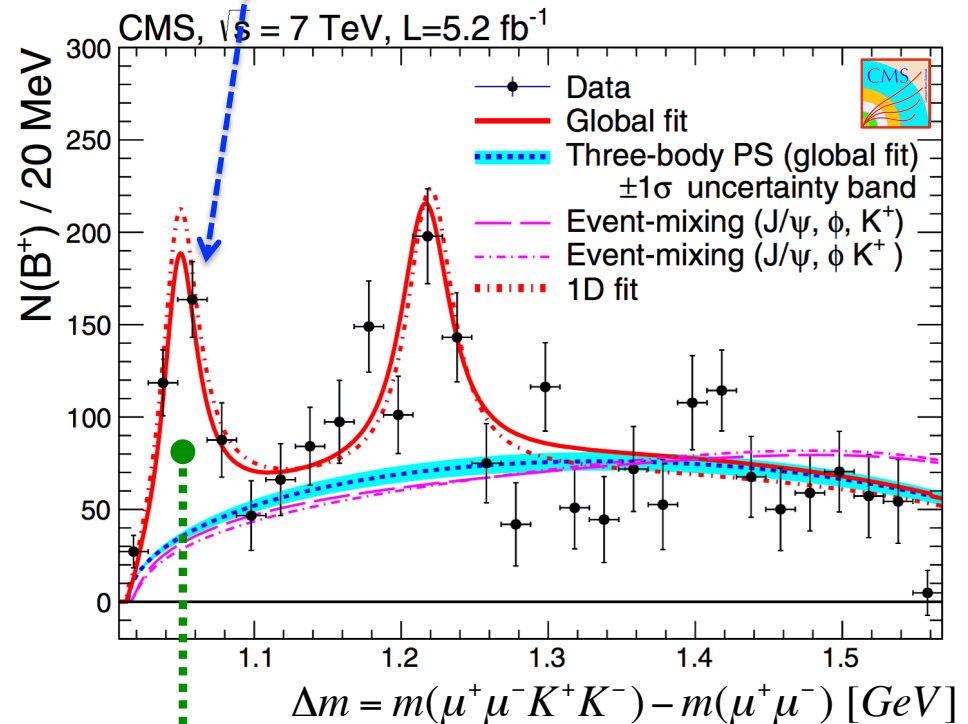
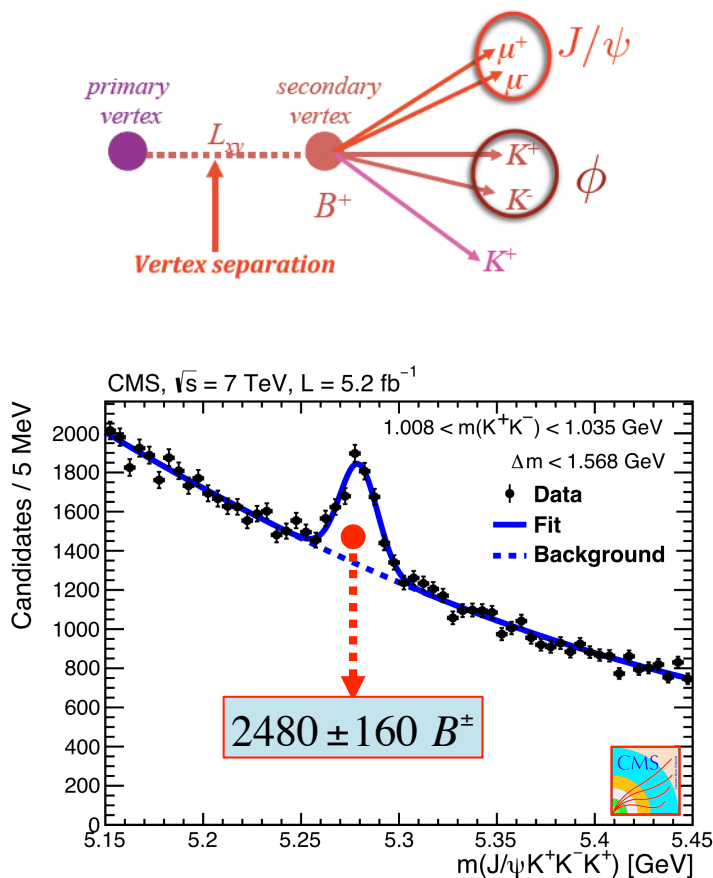
Test application : the Physics case

➤ 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 $B^+ \rightarrow J/\psi\phi K^+$ [PLB 734 (2014) 261]



Test application : the Physics case

➤ 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 $B^+ \rightarrow J/\psi\phi K^+$ [PLB 734 (2014) 261]



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

- $m = 4148.0 \pm 2.4(stat.) \pm 6.3(syst.)$ MeV
- $\Gamma = 28_{-11}^{+15}(stat.) \pm 19(syst.)$ 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 that in the data (ignoring Poisson fluctuations) ⇒ fits with not-extended ML]
- Null Hypothesis binned ML fit performed with the phase-space model only
-

Test application : the toy MC method

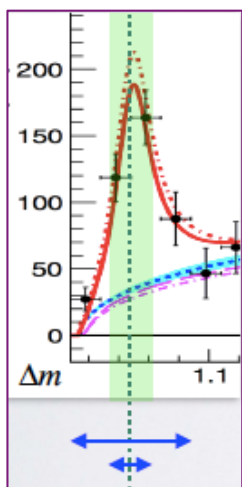
➤ **MC pseudo-experiments** are used to estimate the probability (*p-value*) that background fluctuations would - alone - give rise to a signal as much significant as that seen in the data.

Toy MC fit cycle (for each generated fluctuation)

- **Generation of fluctuated background binned distribution** (3-body phase-space model)
[total #entries fixed by that in the data (ignoring Poisson fluctuations) ⇒ 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

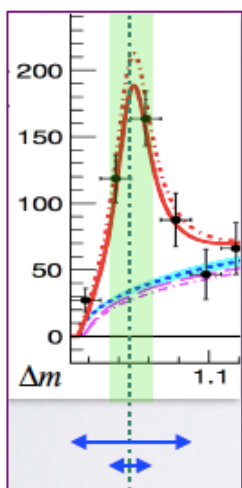
➤ **MC pseudo-experiments** are used to estimate the probability (p -value) that background fluctuations would - alone - give rise to a signal as much significant as that seen in the data.

Toy MC fit cycle (for each generated fluctuation)

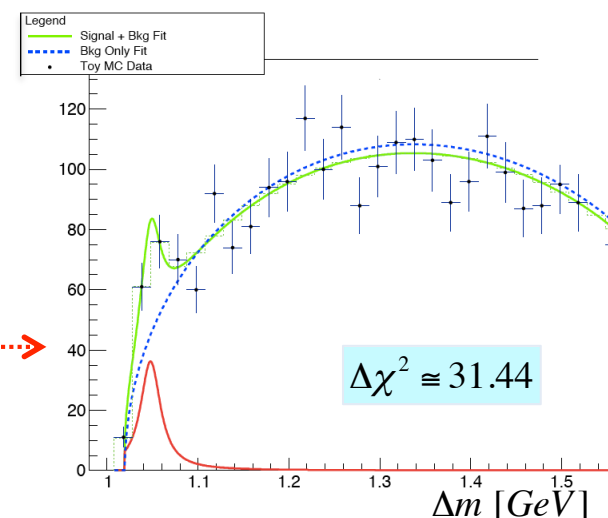
- **Generation of fluctuated background binned distribution** (3-body phase-space model)
[total #entries fixed by that in the data (ignoring Poisson fluctuations) \Rightarrow fits with not-extended ML]
- **Null Hypothesis binned ML fit** performed with the phase-space model only
- **Alternative Hypothesis 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**]

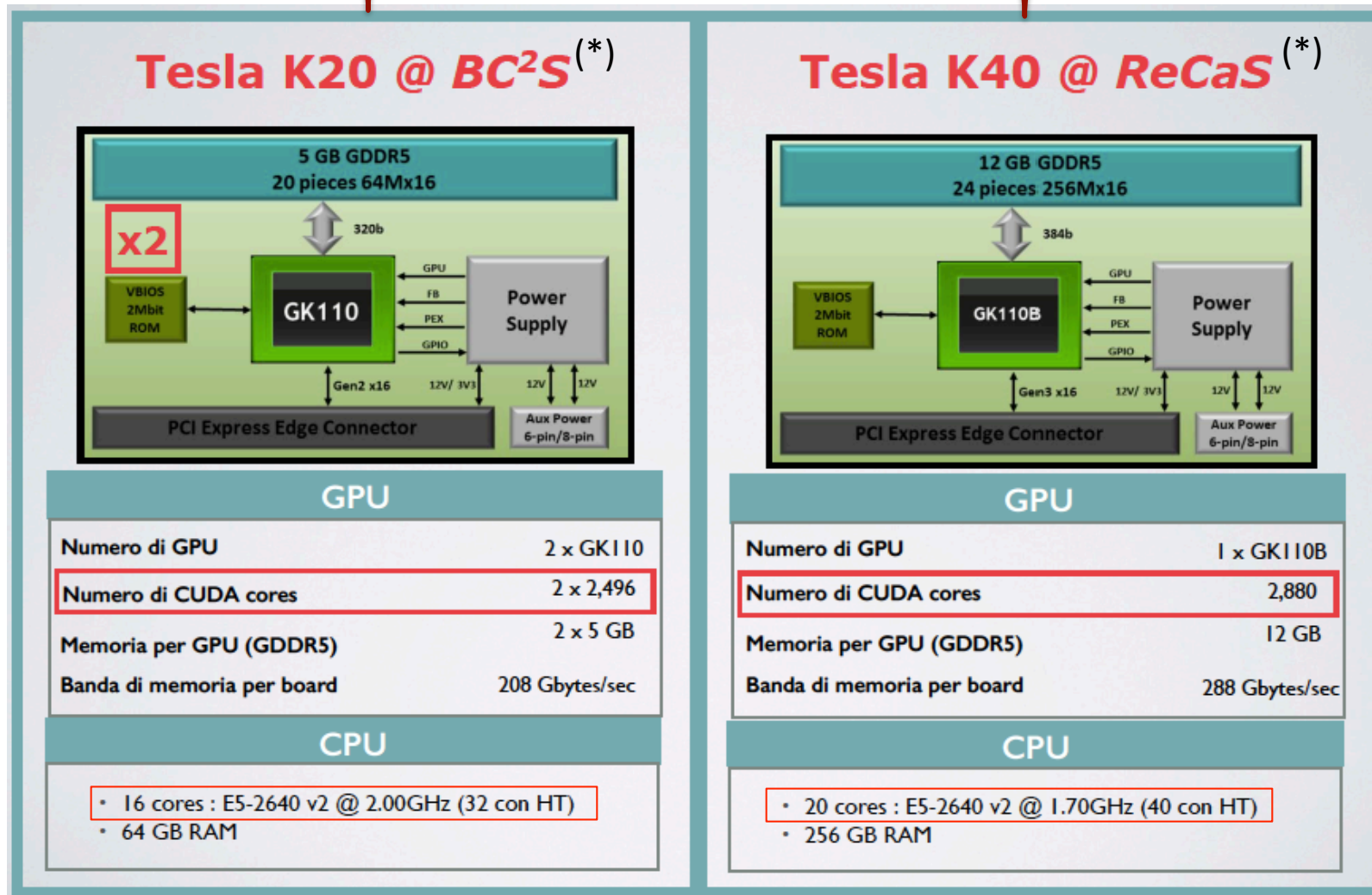


- Fit performed **8 times** within the region of interest (from CDF: **no LEE**) trying different starting values (2 masses & 4 widths).
- For each fit calculate a $\Delta\chi^2$ w.r.t. the Null Hypothesis fit; the best $\Delta\chi^2$ fit among the 8 alternative fits is chosen ! \rightarrow
- A $\Delta\chi^2$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



Hardware setup for this study

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



(*) <http://www.recas-bari.it/index.php/en/>

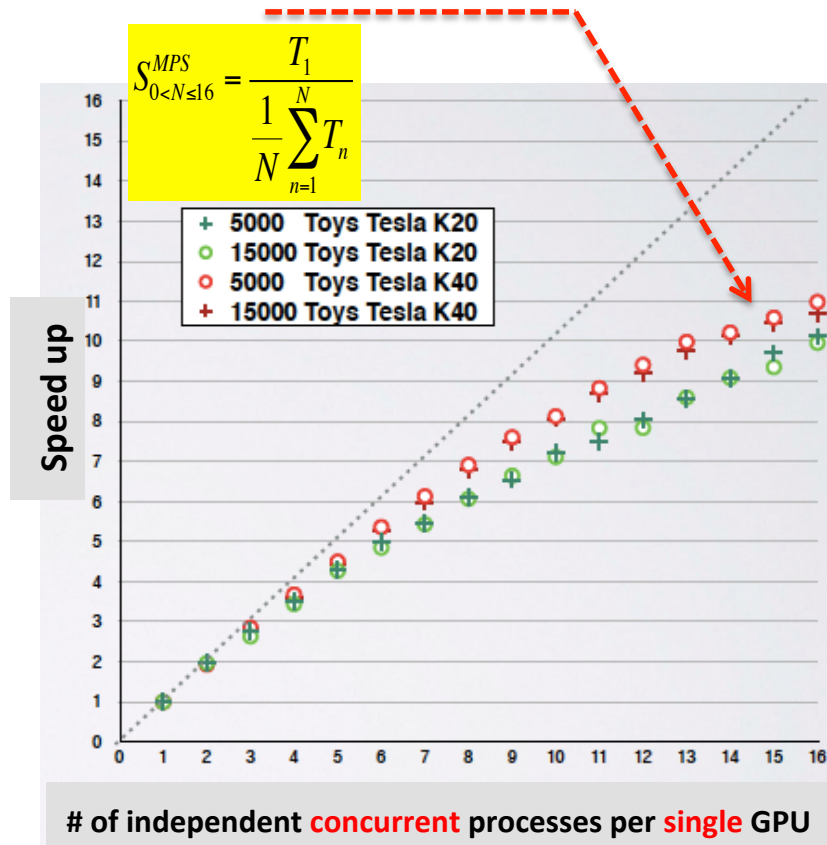
Performance of *GooFit* on nVIDIA Multi Process Server

➤ A single *GooFit* process does not exploit the whole computational power of a GPU (60%-70% usage).
The **Multi Process Server (MPS)** tool allows the execution of several (up to 16 max) simultaneous processes on the same GPU acting as a *scheduler* and allowing a balanced full use of GPU capabilities.

➤ Each process uses:

- 1(shared) GPU and 1(exclusively assigned) CPU

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



Performance of *GooFit* on nVIDIA Multi Process Server

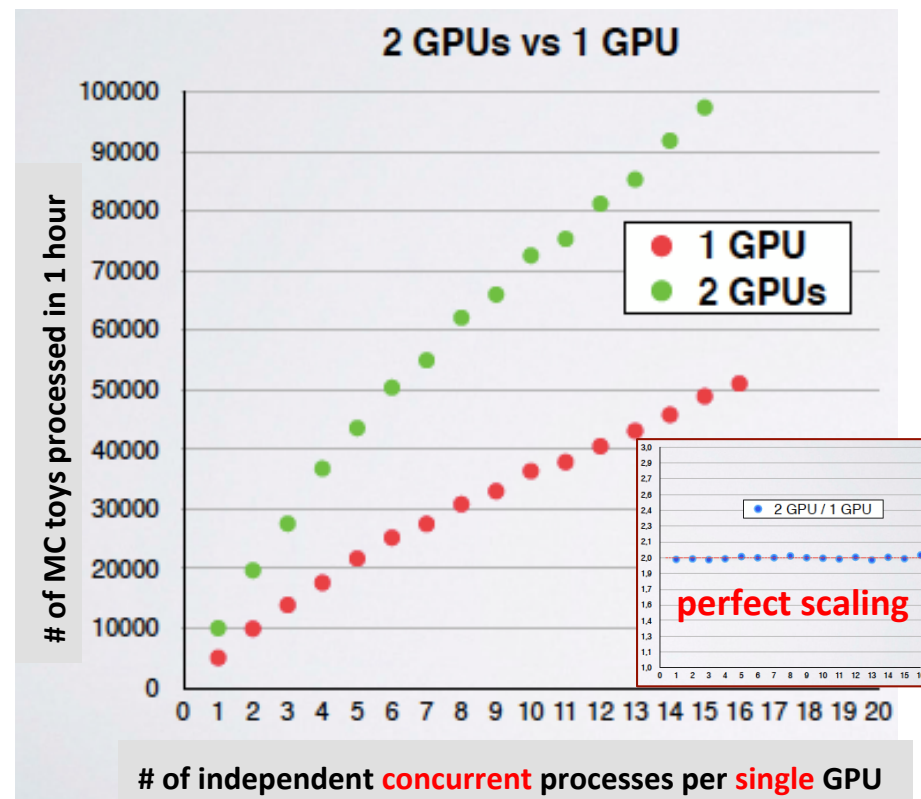
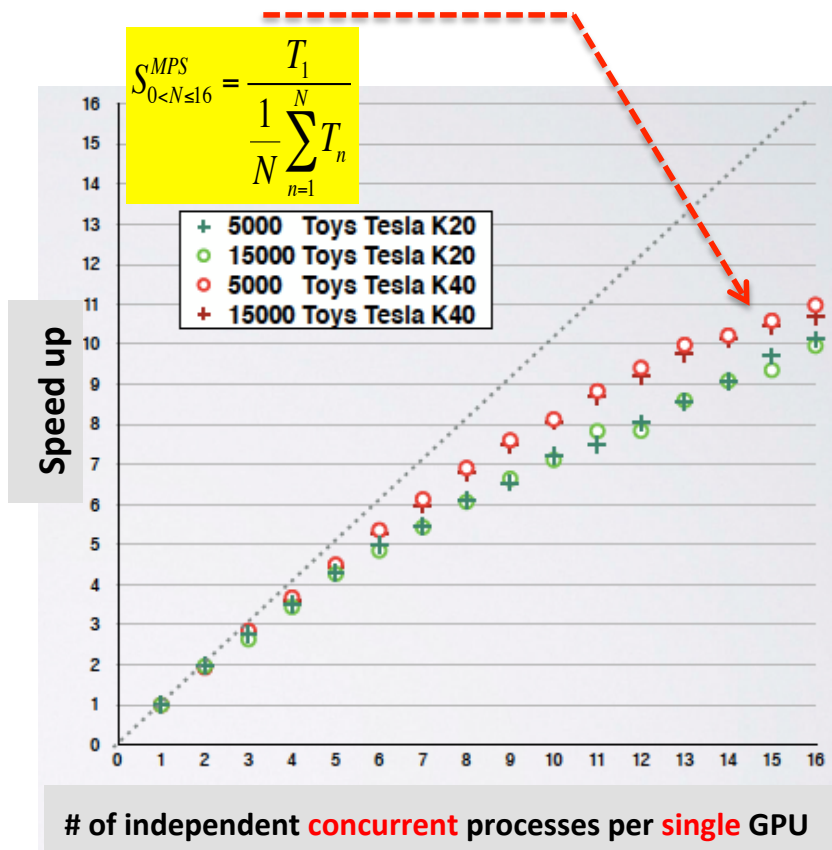
➤ A single *GooFit* process does not exploit the whole computational power of a GPU (50%-70% usage). The **Multi Process Server (MPS)** tool allows the execution of several (up to 16 max) simultaneous processes on the same GPU acting as a *scheduler* and allowing a balanced full use of GPU capabilities.

➤ 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 tails, 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 tails, unavoidable by running *RooFit* on a cluster in *Push approach* (the last job determines the total exec. time).

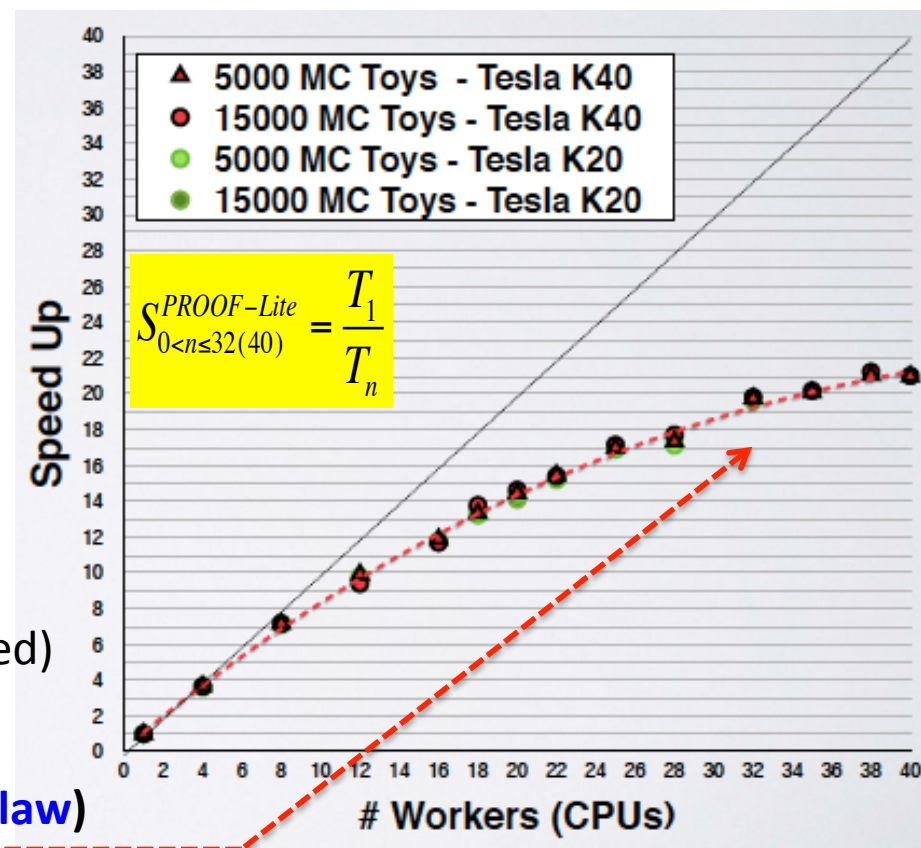
➤ Check **speed up performance** on 2 servers:

- 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 (Amdahl's law)**

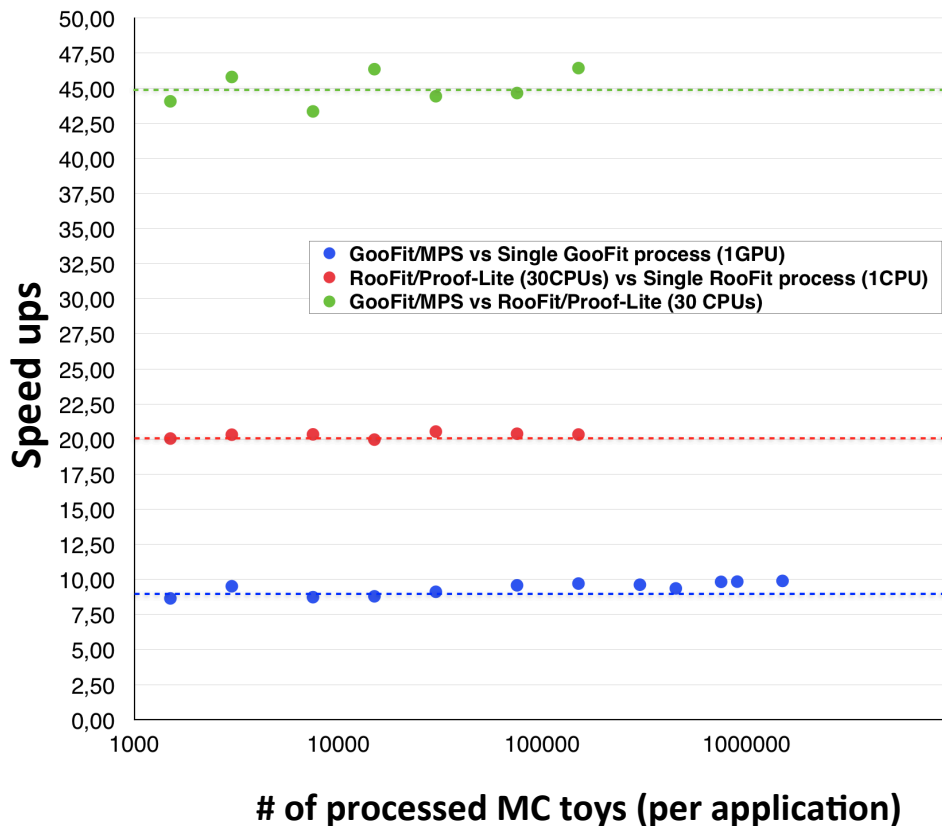


[*] G.Ganis et al., *PoS ACAT08 (2008) 007*; A.Pompili et al., *J. Phys.: Conf. Ser.* **396** 032043, CHEP12, 2012

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

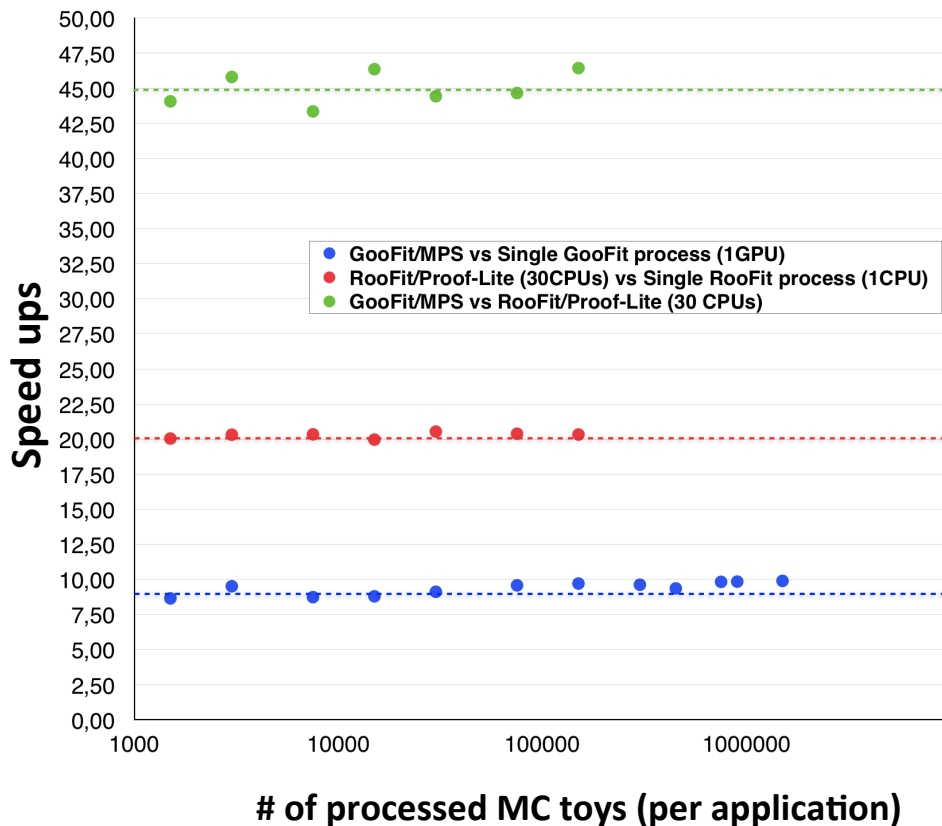
$$S_{n=30=N} \Big|_{2-TK20} = \frac{T_{n=30}^{RooFit}}{T_{N=30}^{GooFit} \Big|_{2-TK20}}$$



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

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

➤ Good scaling with extended # of MC toys :

$S_{n=30}^{PROOF-Lite}$
~20

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

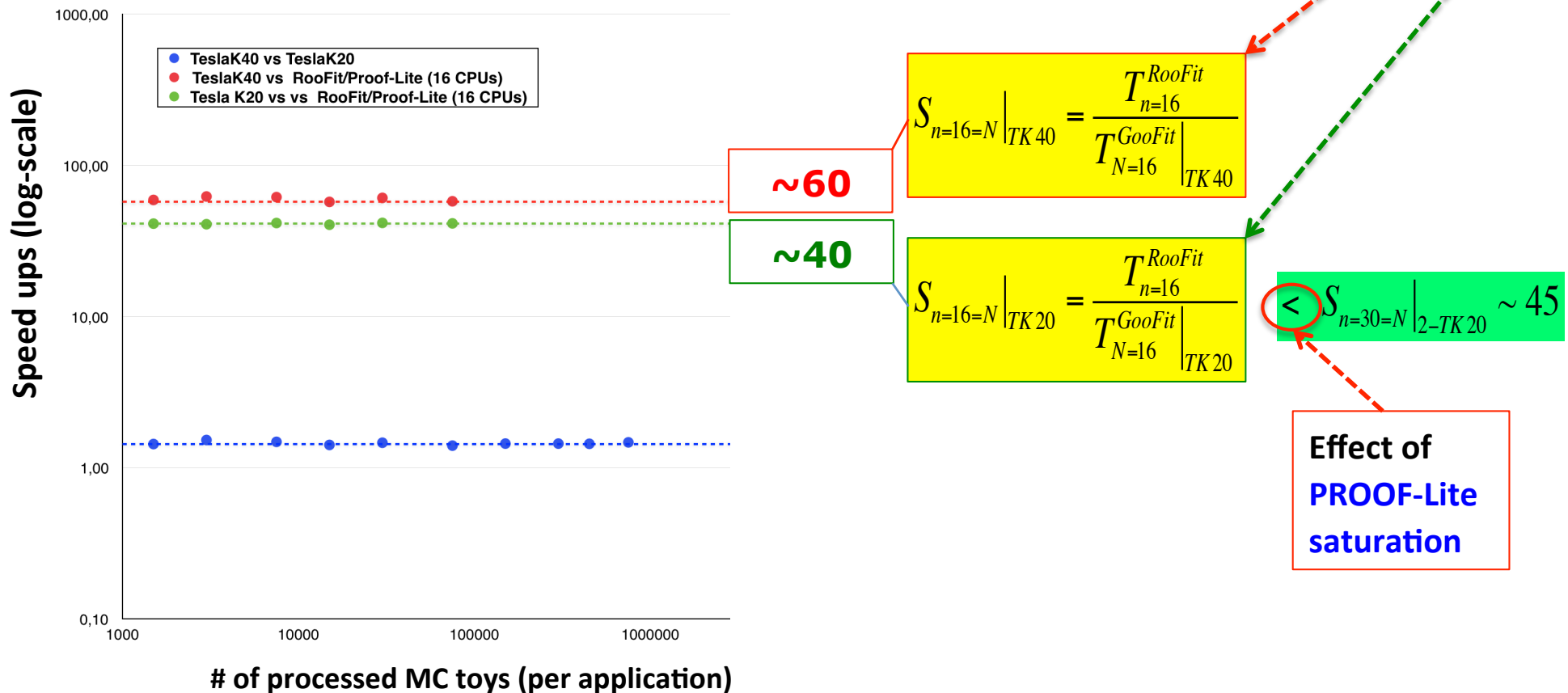
~9
 $S_{N=15}^{MPS-TK20}$

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

RooFit/PROOF-Lite vs GooFit/MPS performances - 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)

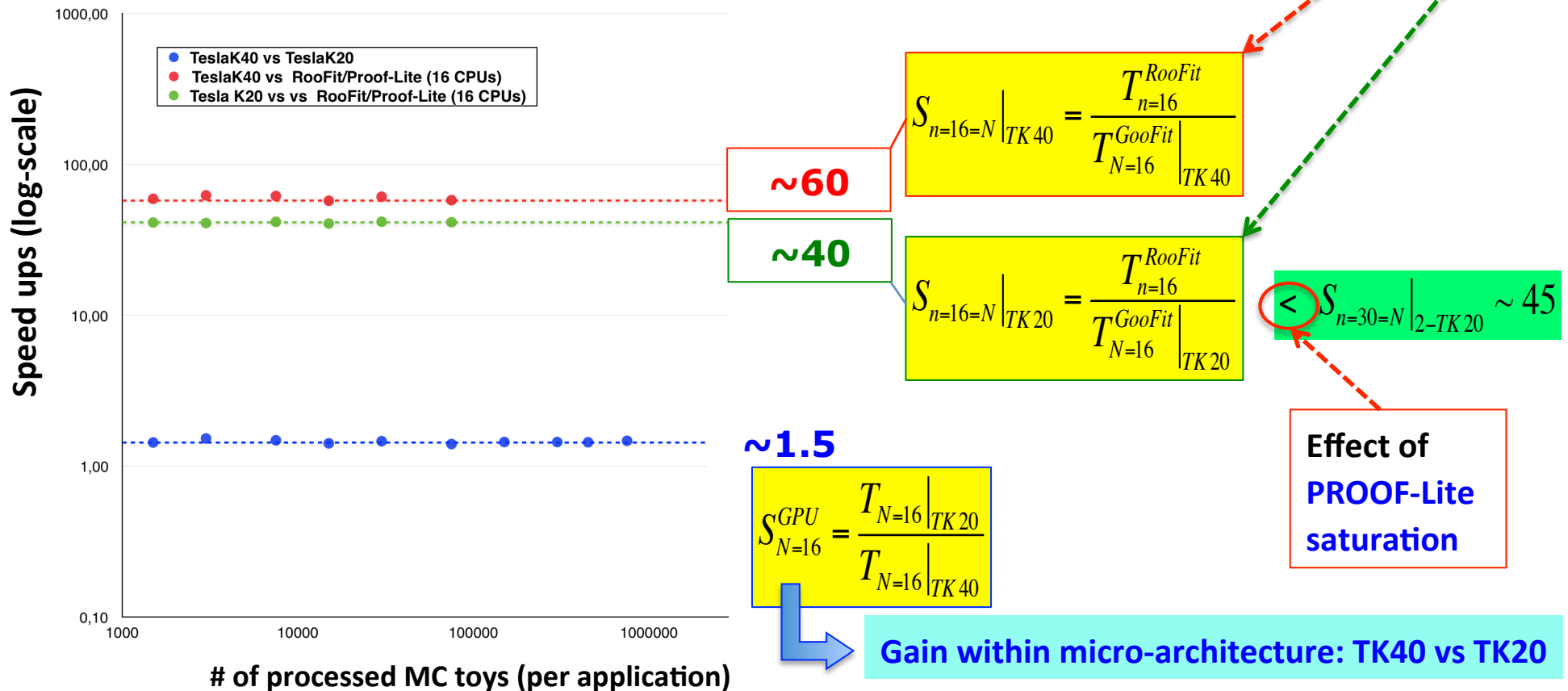
➤ We can compare : - 1 PROOF-Lite job using 16 workers (on 16 CPU cores)
with : - 1 GooFit/MPS job running 16 simultaneous processes on single TK40 / TK20



RooFit/PROOF-Lite vs GooFit/MPS performances - 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

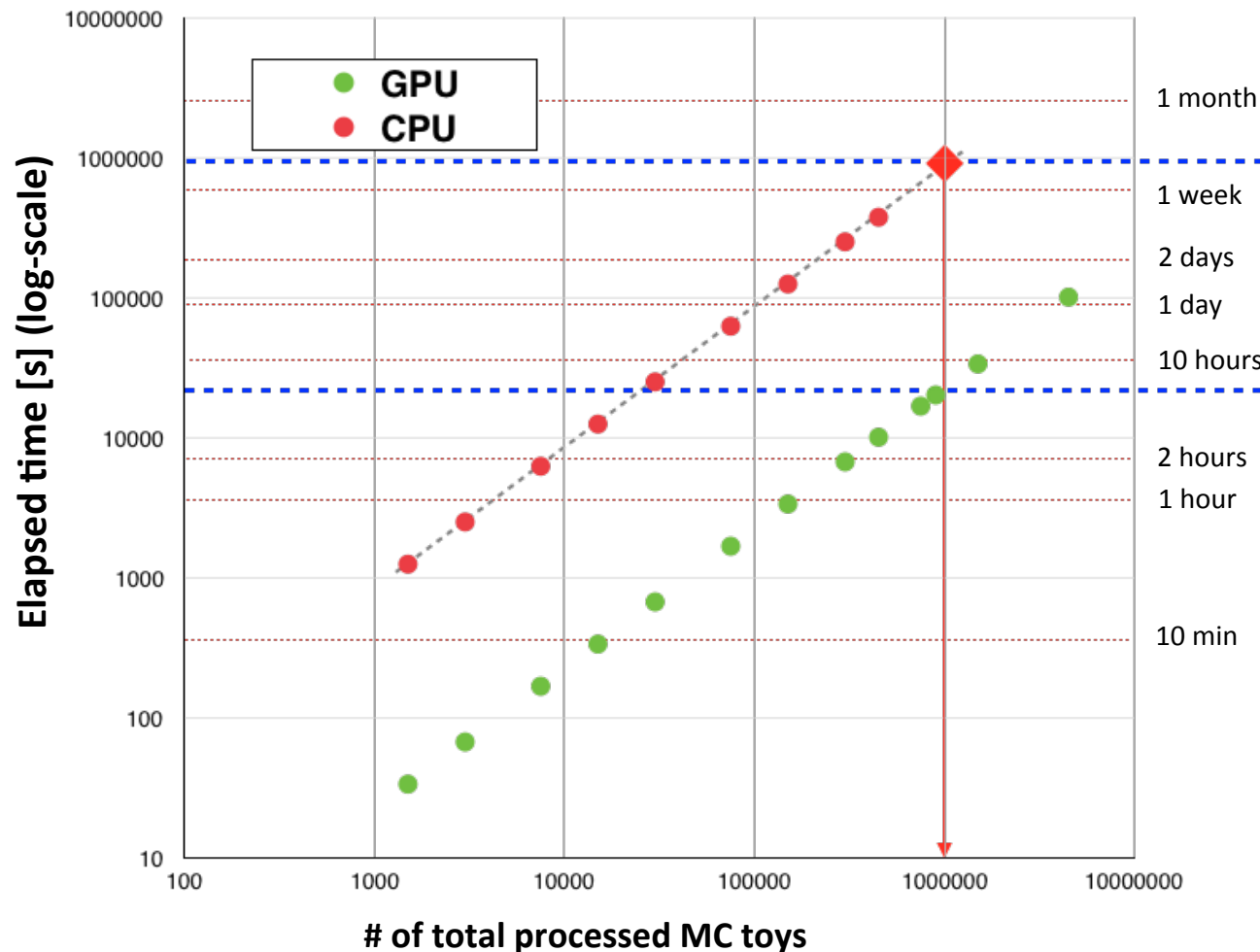


RooFit/PROOF-Lite vs GooFit/MPS performances - III

➤ 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).



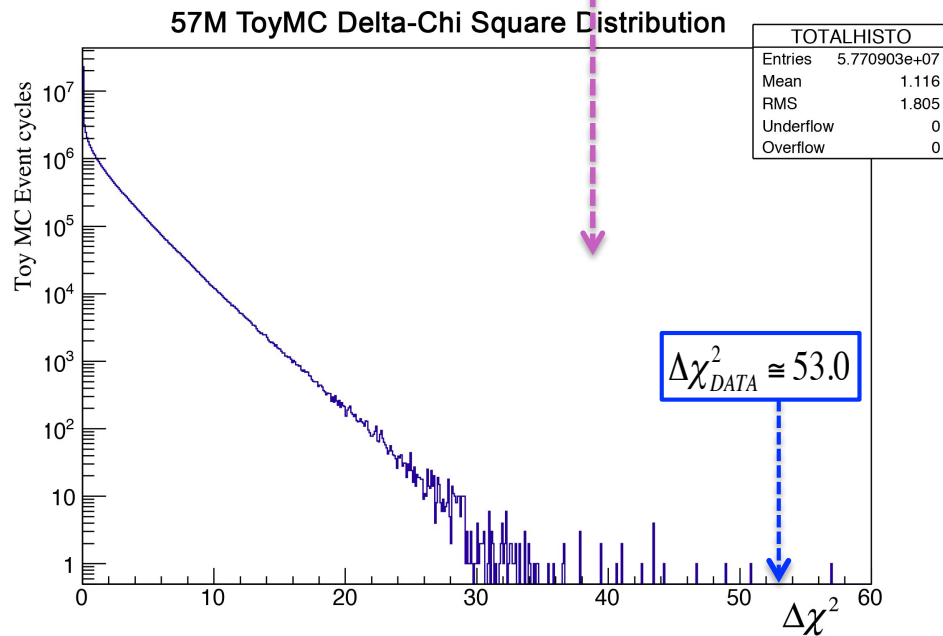
~ 11 days
x 1M Toys
~ 6 hours

To get a signal significance $>5\sigma$, a p -value $< 3 \times 10^{-7}$ is needed, namely at least 3.3M toys are needed.

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

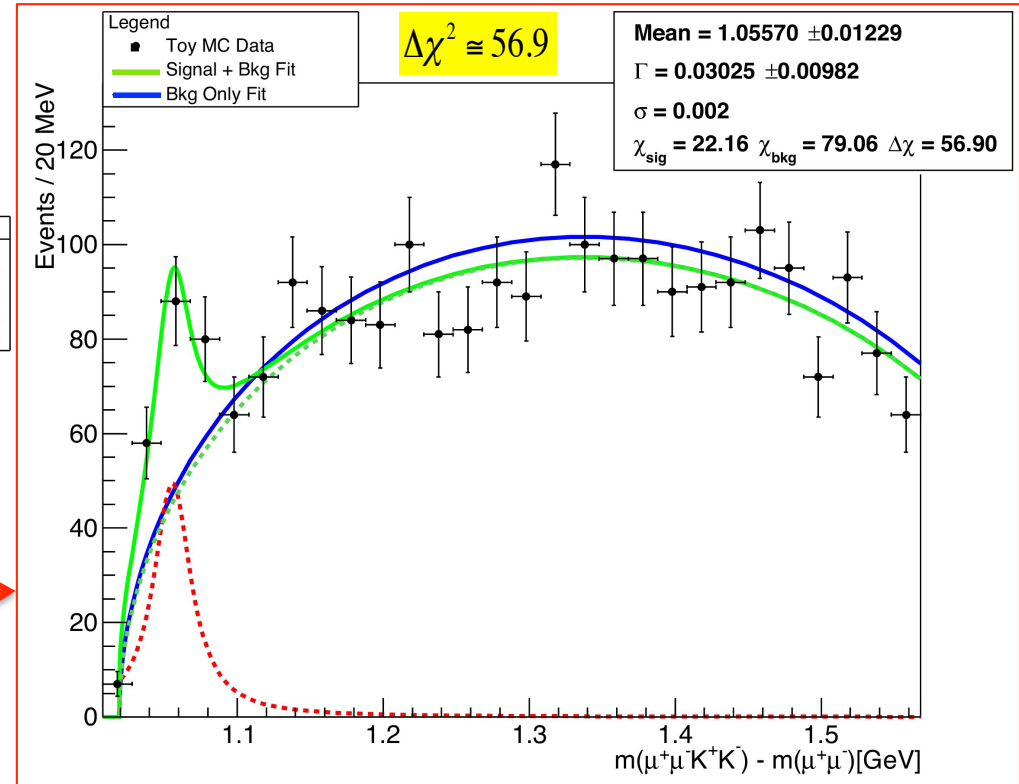
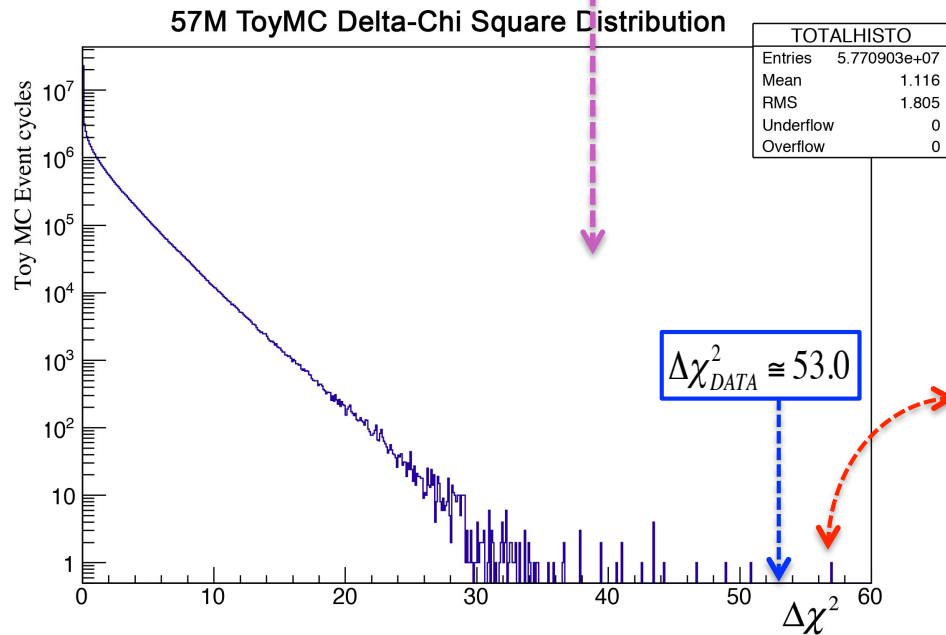
P-value & statistical significance estimation

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



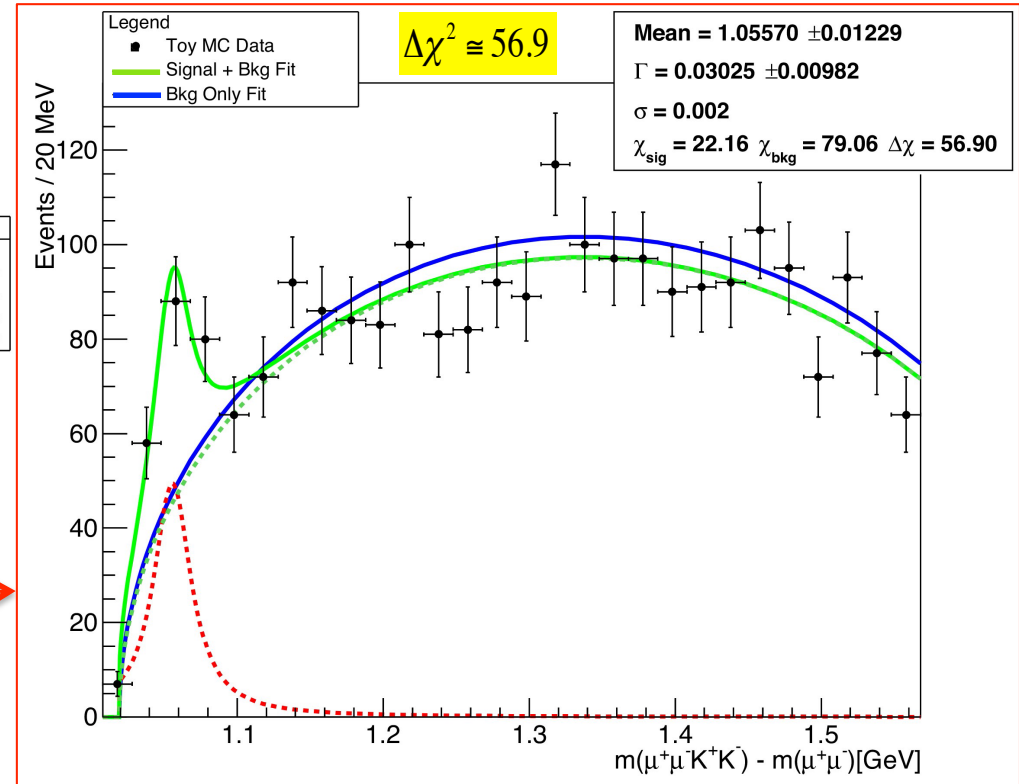
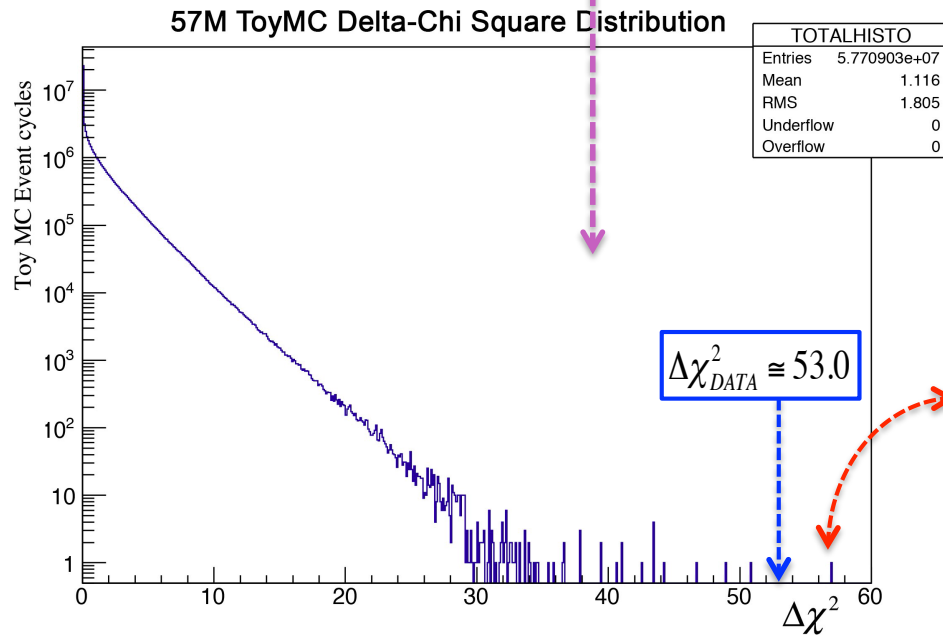
P-value & statistical significance estimation

➤ 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)



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:

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

Equivalent (gaussian) statistical significance:

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

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 *RootFit*).

Inverse function of the cumulative distribution of the standard gaussian

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 $\Delta\chi^2 = \chi_{H_0}^2 - \chi_{H_1}^2$],

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 $H_1 \rightarrow H_0$ 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 $\Delta \chi^2 = \chi_{H_0}^2 - \chi_{H_1}^2$],

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 $H_1 \rightarrow H_0$ 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 : $P = \int_{t_{obs}}^{\infty} \chi_{\nu_1 - \nu_0}^2(t) dt$

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!

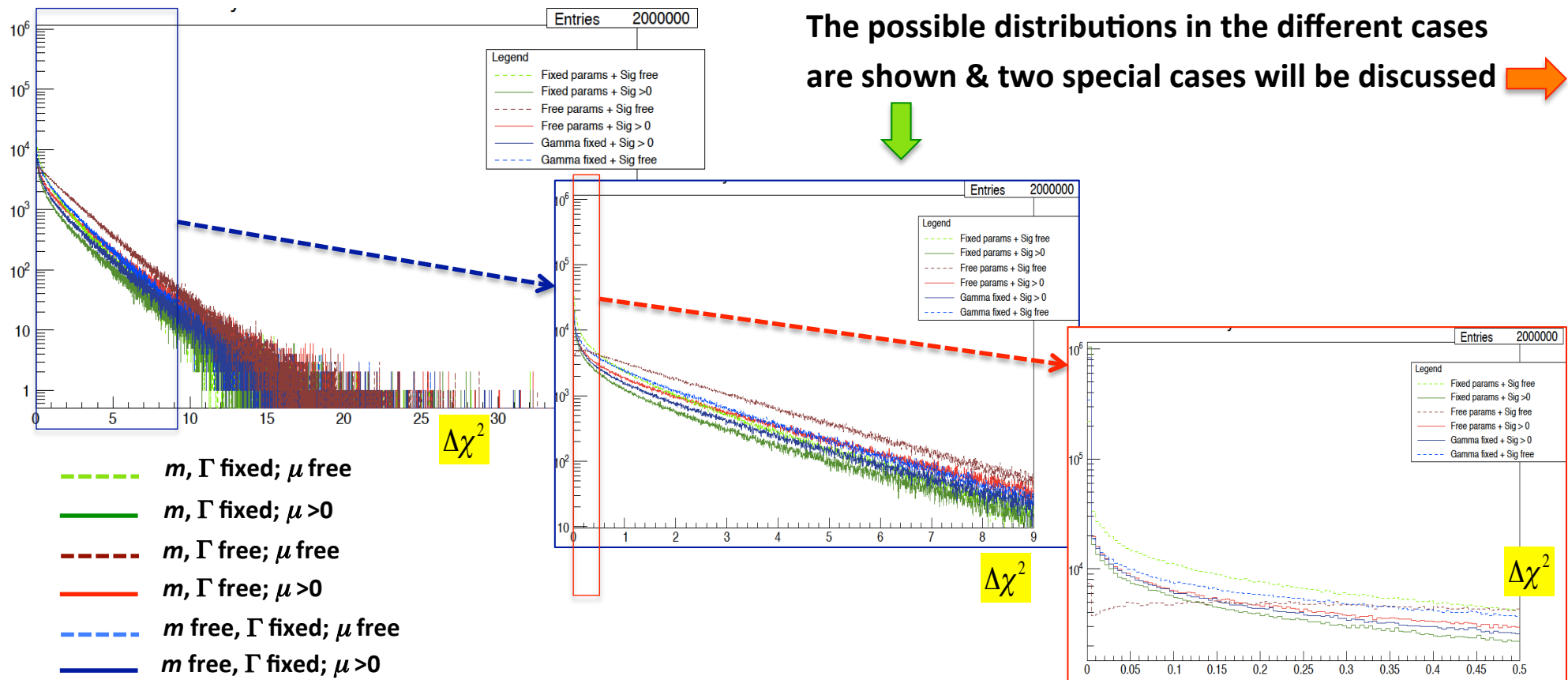
The signal parameters in the model of H_1 hypothesis are mass (m), width (Γ) and yield ($\mu \geq 0$).

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 $\mu = 0$ for purposes of **establishing the existence of a signal process**, or ... of $\mu \neq 0$ for purposes of **obtaining a confidence interval**.

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

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 $\mu = 0$ for purposes of **establishing the existence of a signal process**, or ... of $\mu \neq 0$ for purposes of **obtaining a confidence interval**.

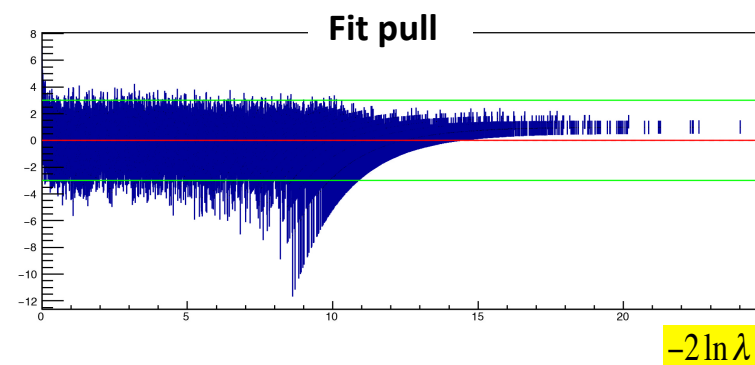
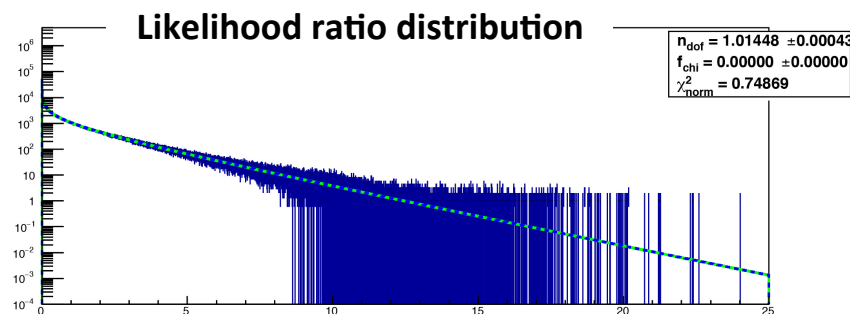
In the latter case, following Cowan *et al.* [*] the PDF of the test statistic approaches a **chi-square distribution for 1 d.o.f.** :
 [**in agreement with Wilks theorem !**]

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

- 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 :

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



[*] 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 $\mu = 0$ in a class of model where we assume $\mu \geq 0$. **Rejecting $\mu = 0$ (the null hypothesis) leads to the discovery of a new signal.**

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

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

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

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

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

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

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.** :

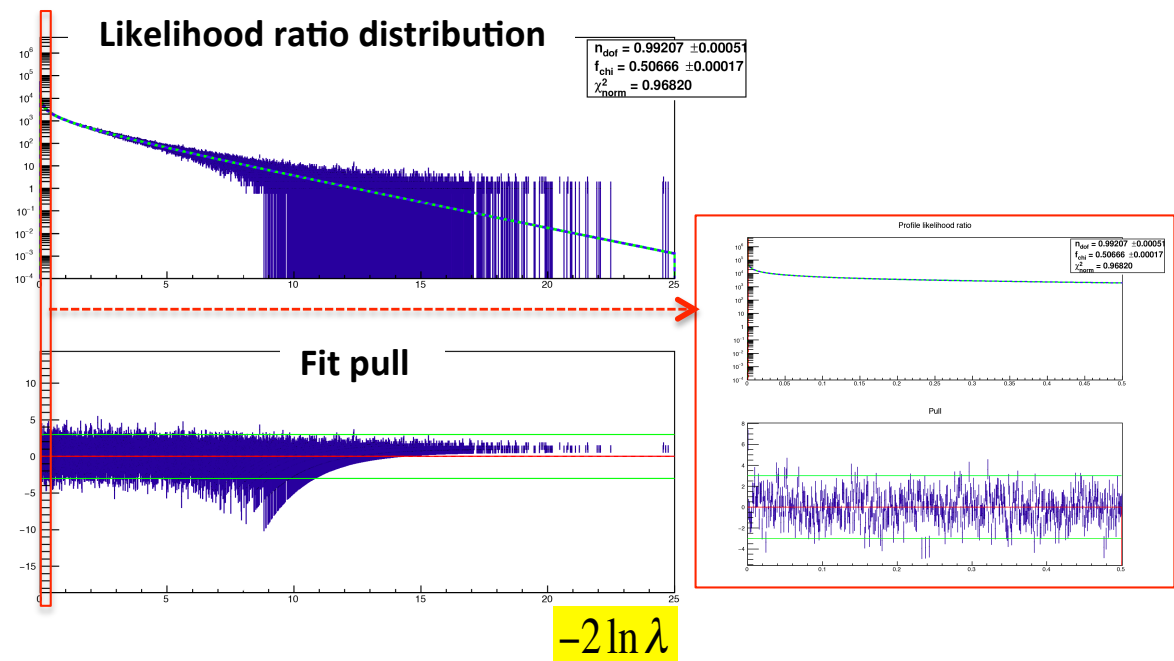
$$g(q_0 | \mu = 0) = \frac{1}{2} \delta(q_0) + \frac{1}{2} \left[\frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{q_0}} e^{-q_0/2} \right]$$

- 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

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 unreliable (the time needed unbearable) ,**
- **turning to GPUs would be mandatory ,**
- ***GooFit* would be the reliable & crucial tool .**

If you are interested to start learning & 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](#).

If you are interested to start learning & 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 collaborators of CMS-Bari: [Adriano Di Florio](#) & [Leonardo Cristella](#) (University of Bari)
& [Giacinto Donvito](#) (INFN-Bari, Tier2 manager)

& 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)

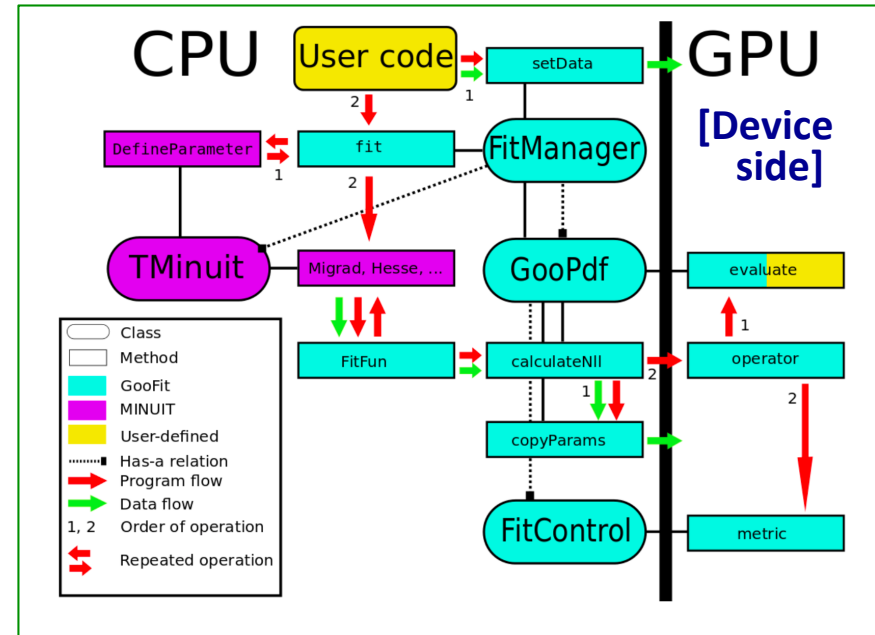
- [Brad Hittle](#) (Ohio Supercomputer Center) & [Tommaso Dorigo](#) (INFN-Padova)

Backup slides/Additional material

GooFit framework : few details

- A *GooFit* program has 4 main components :
 - a **GooPdf** object representing the PDF modelling the physical process
 - the **fit parameters** (represented by **Variables** contained in the **GooPdf**)
 - the **data** (gathered into a **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

The **fit** method of **FitManager** ...

- 1) sets-up the MINUIT fit ,
- 2) passes control to a MINUIT's method that will in turn call GooFit's evaluation method (**evaluate**) for different sets of parameters until it converges.

The **metric** method of **FitControl** allows switching between ML and χ^2 fits, or between UML and binned ML fits, without changing the PDF creation code.

GooFit analogy with RooFit

➤ Originally *GooFit* was meant to give access to the parallelizing power of CUDA (nVIDIA's programming language) without requiring to write CUDA code.

- Extension to OpenMP was obtained by using the Thrust library (thread code abstraction).
- A user needs to do CUDA coding **only if** a new class is needed.

***GooFit* has a code structure similar to *RooFit* framework; the overall fit set-up and running are familiar to *RooFit*.**

RooFit	GooFit
RooRealVar	Variable
RooDataSet	UnbinnedDataSet
RooDataHist	BinnedDataSet
RooAbsPdf	GooPdf
RooGaussian	GaussianPdf
RooArgSet	vector<Variable*>
RooPlot	None! Use ROOT plotting.
myPdf->plotOn(foo)	myPdf->getCompProbsAtDataPoints(points)
RooAbsTestStatistic	FitControl

➤ **Simple PDFs** : Exponential, Polynomial, Gaussian, Crystal Ball, Voigtian, Relativistic Breit-Wigner, Step function and others.

Composite PDFs : Sum (of PDFs), product, Composition (only 1D), Convolution, Map.

Specialized mixing PDFs :

Coherent amplitude sum, incoherent sum, 3-Gaussian resolution, and some others; for instance:

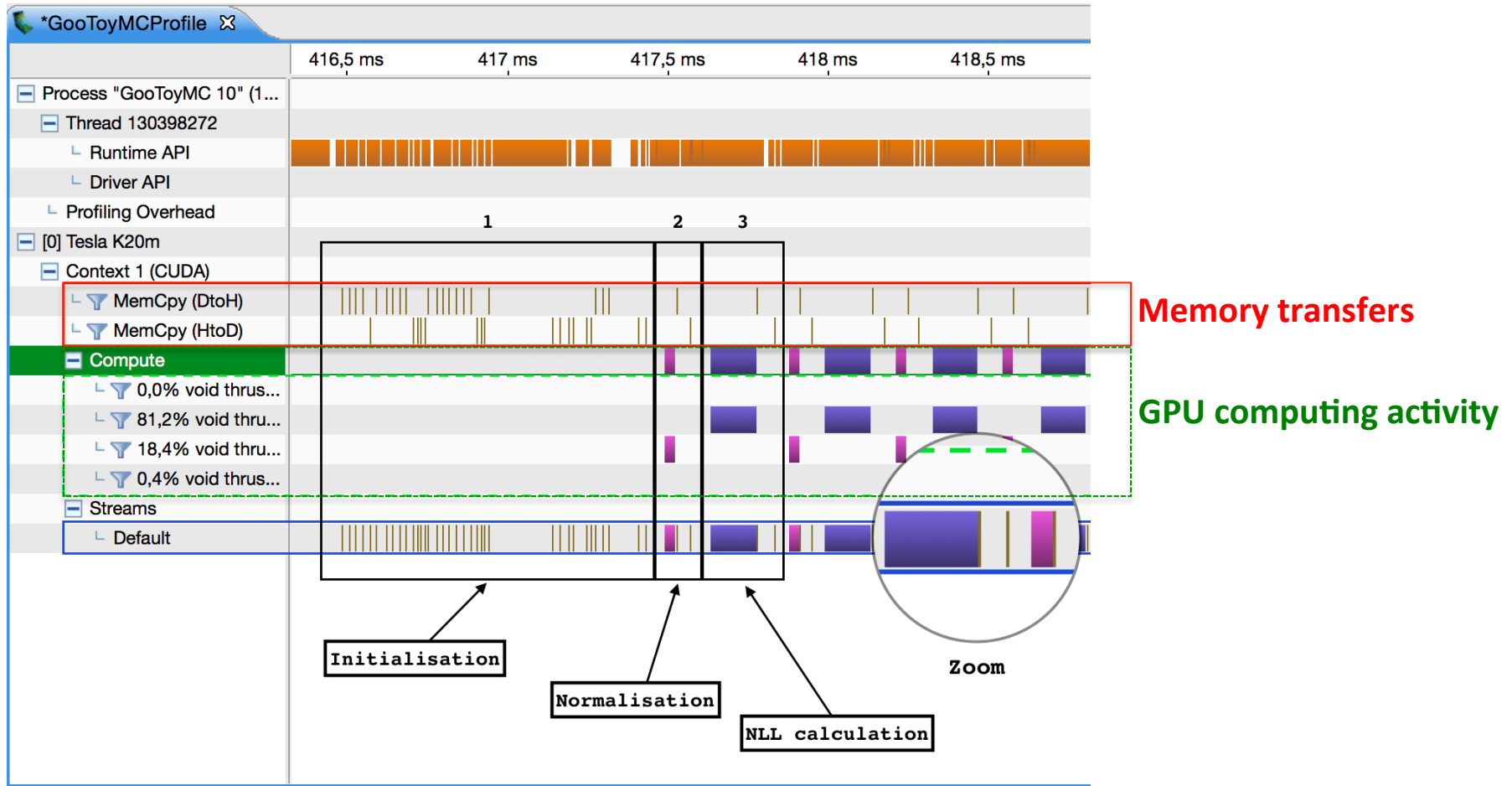
TddpPDF (DalitzPlotPdf) as the main engine for **time-dependent (-integrated)** Dalitz Plot fits, with a list of ResonancePdf objects as input to describe different (non-)resonance amplitudes.

➤ The code development mainly within **LHCb Cincinnati group** (M.Sokoloff, R.Andreassen, L.Sun, A.A.Alves Jr.)

➤ You can write your own PDFs with relative ease (there is some example PDF code)

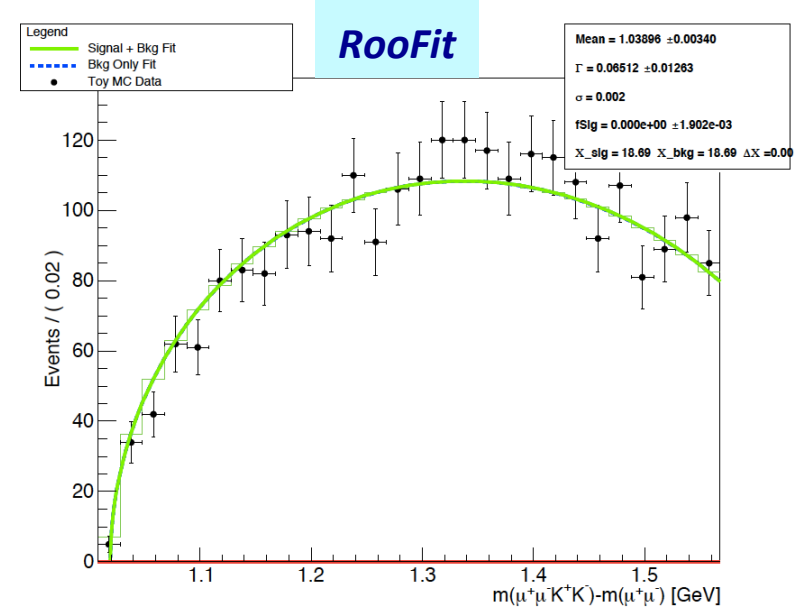
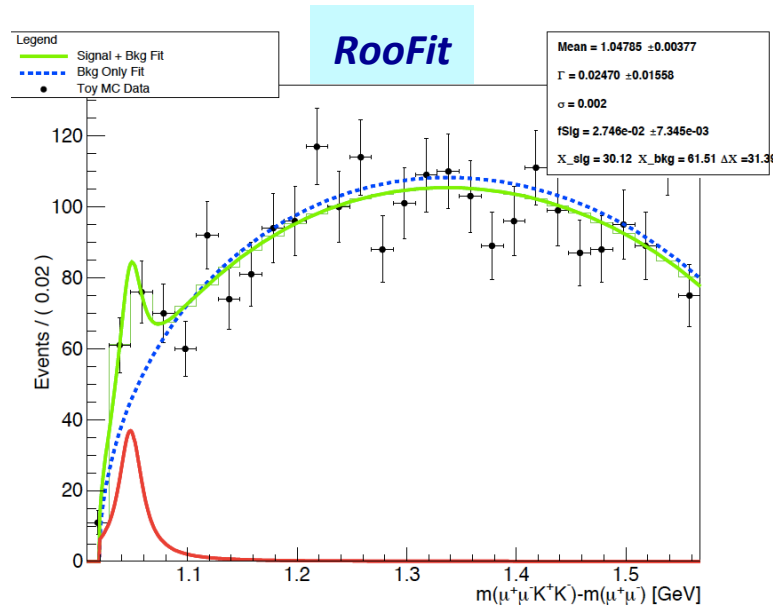
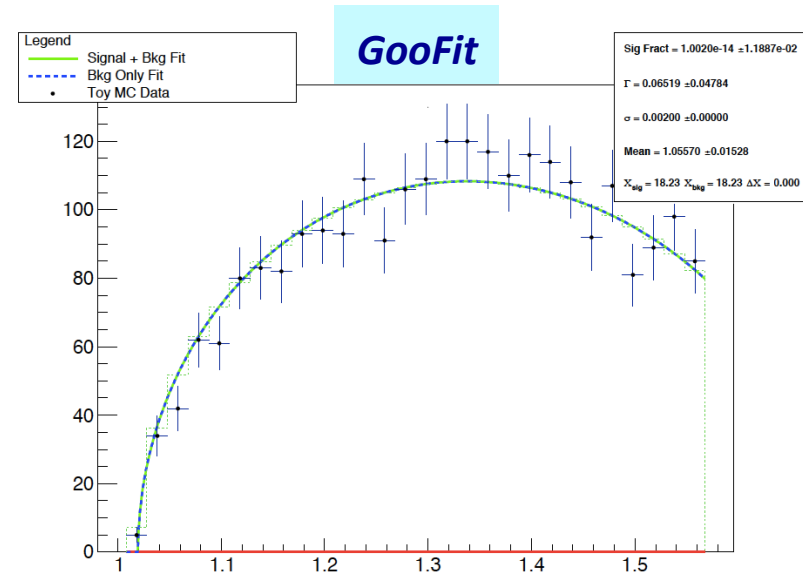
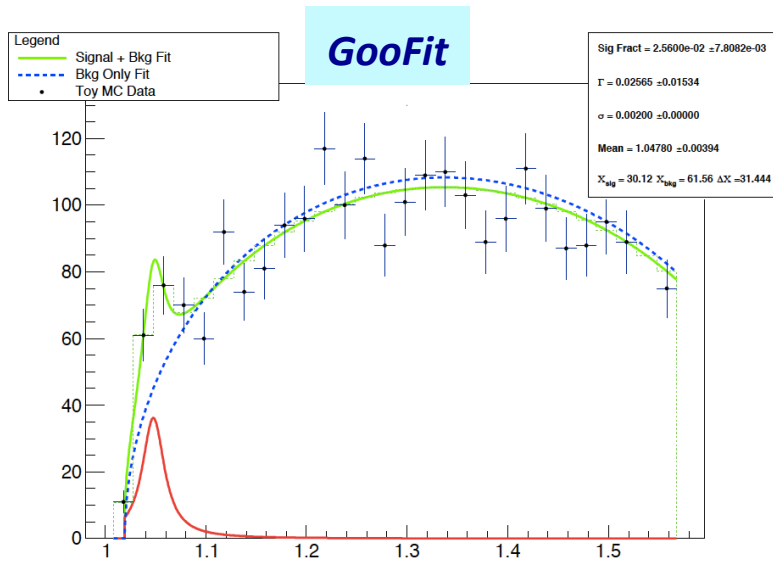
GooFit MC toy monitoring

➤ Fragment of a *GooFit* toy MC process taken by the nVIDIA Visual Profiler [nvvp]

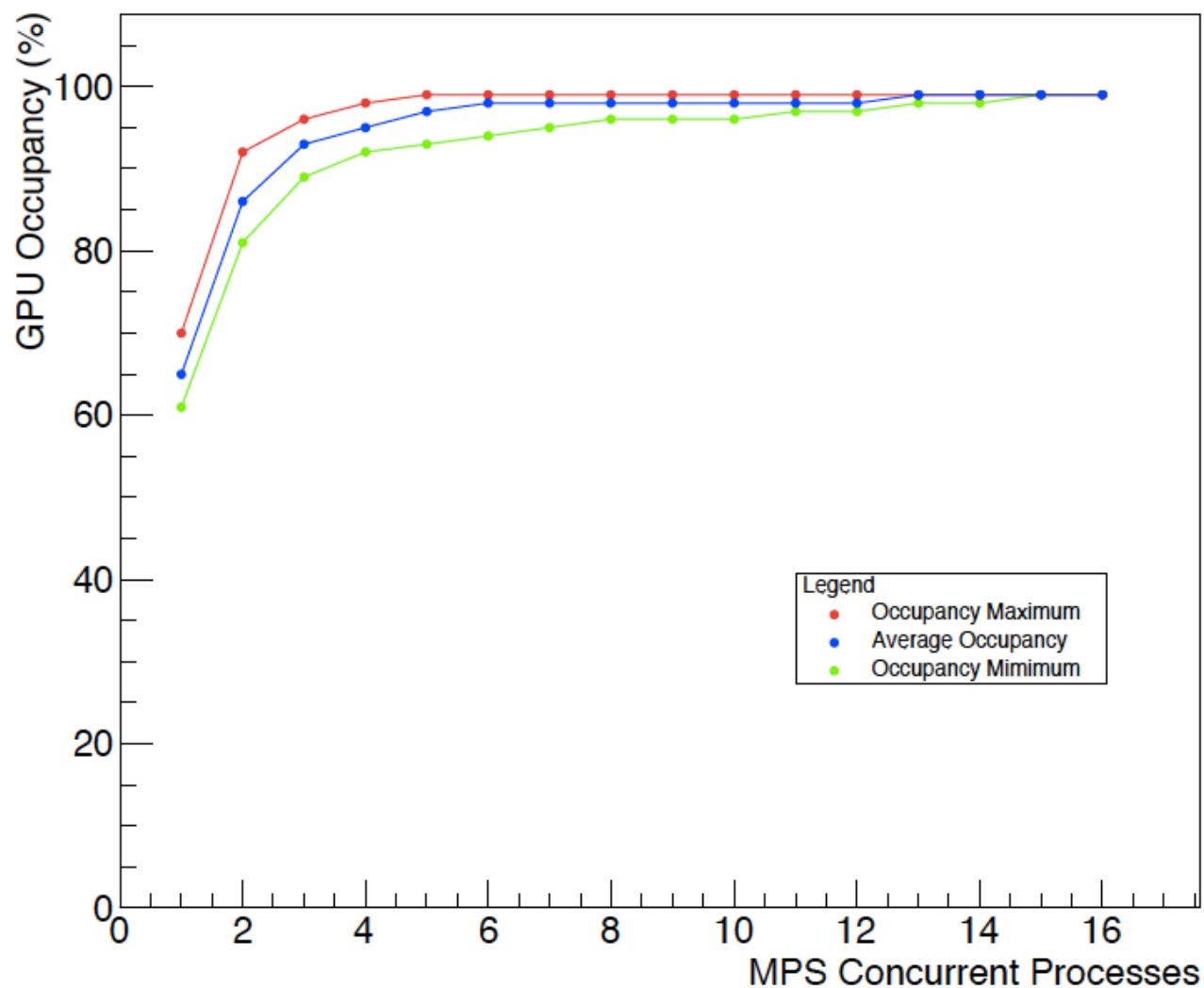


GooFit/RooFit fit examples

➤ Two examples of the **same fluctuation** as fitted both by *RooFit* and *GooFit* :

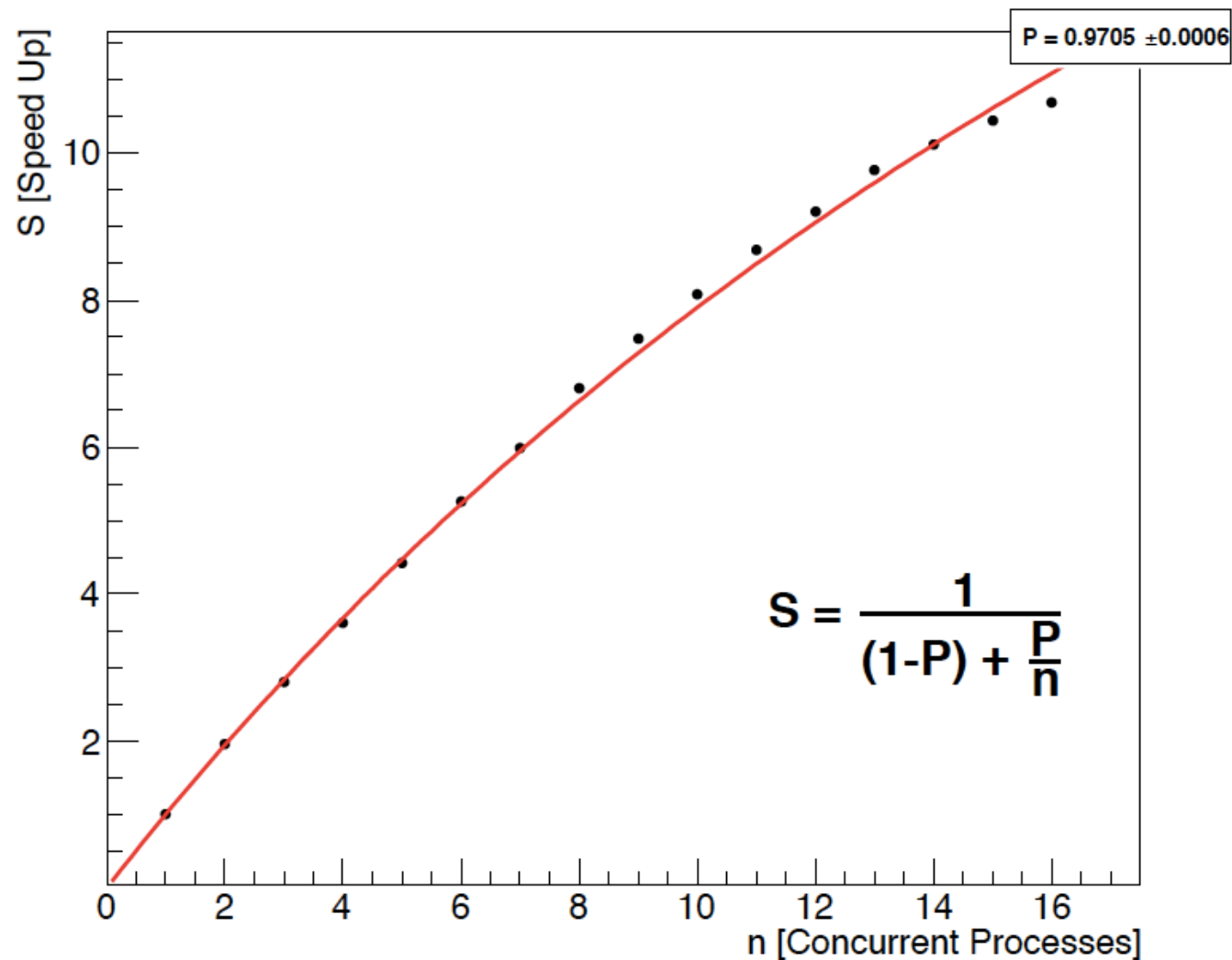


GPU occupancy with MPS



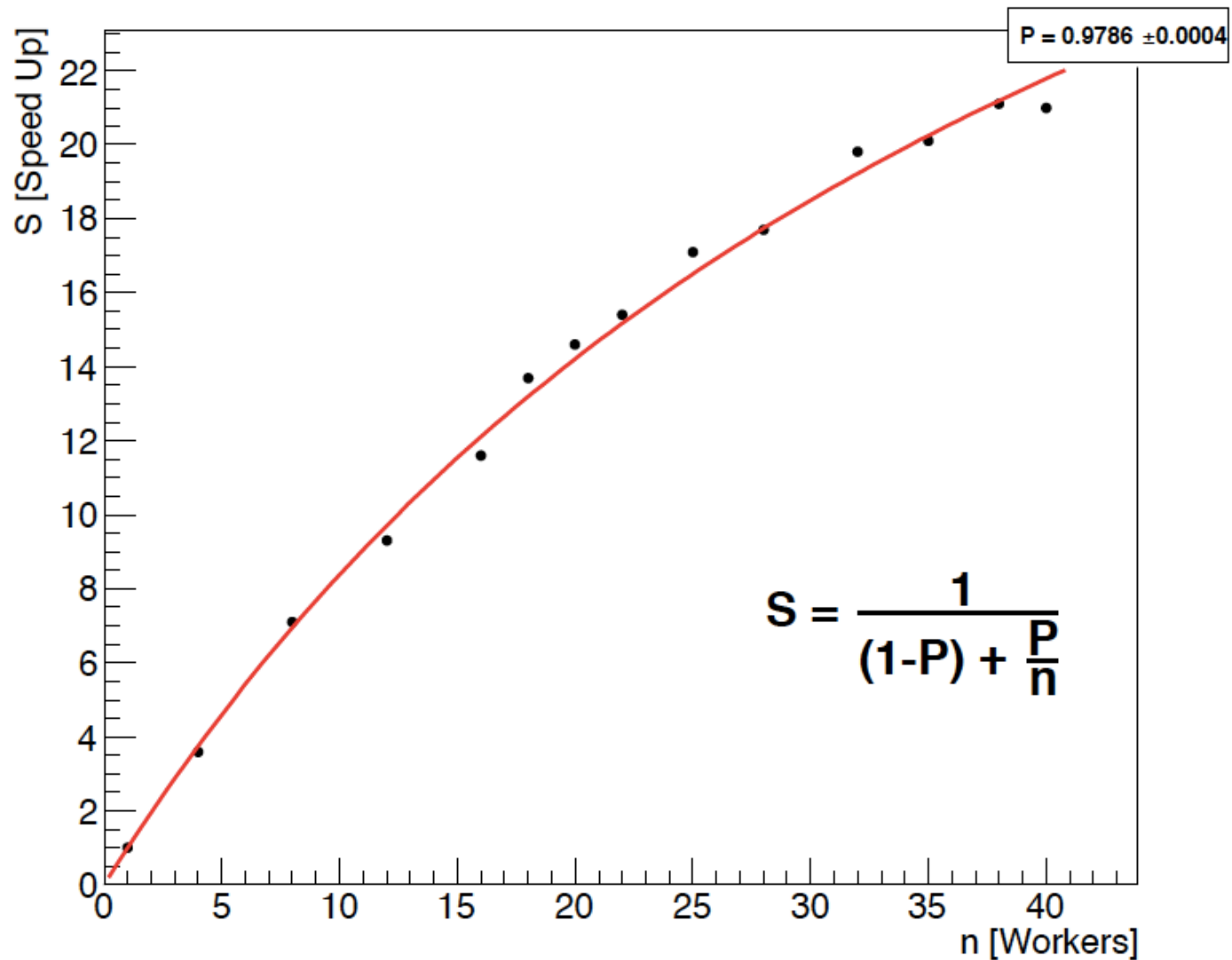
- GPU occupancy saturates with 4-5 simultaneous processes:
some development work and code profiling are needed to enhance the occupancy.

MPS speed up : Amdhal fit



➤ Serial overhead is $\sim 3\%$ of the application execution.

PROOF-Lite speed up : Amdhal fit



➤ Similar P (parallelizable fraction).