Fast Simulations at LHCb

Adam Davis
ECHEP Workshop

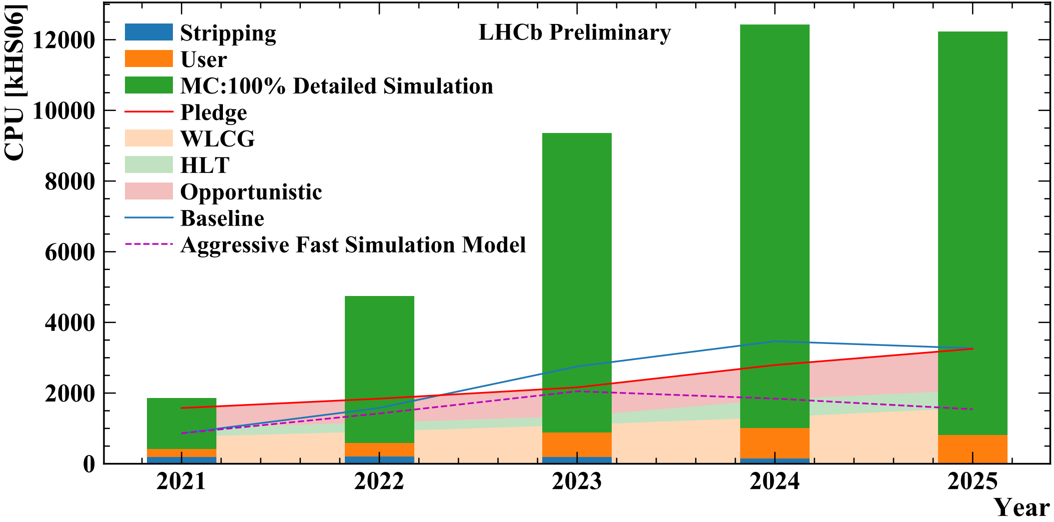
Feb 17, 2020

With many thanks to P. Ilten, M. Kreps

The University of Manchester

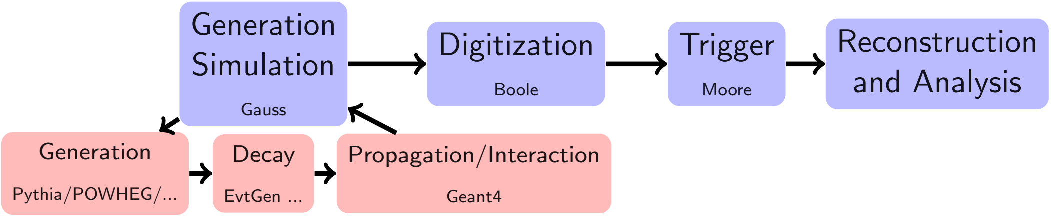
Future Simulation Needs

[LHCb-FIGURE-2019-018]



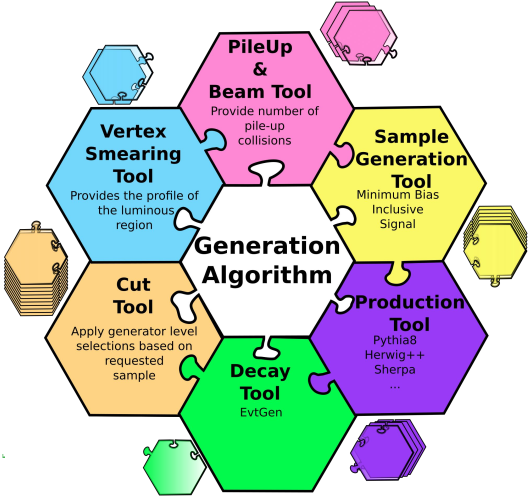
Simulation Strategy

- Projected CPU usage is too high for Upgrade and Upgrade II era
- Must actively pursue other simulation options
- Reminder: LHCb simulation sequence



- Efforts to speed up all portions ongoing

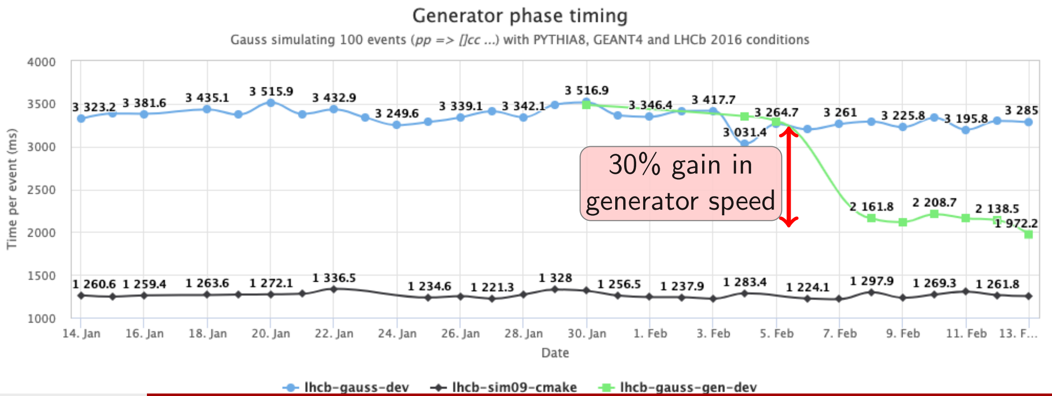
Generation Phase



- Highly modular generation phase → support the large physics programme of LHCb
- Incoporated in LHCb software, which is built around Gaudi Algorithms

Opportunity: Generation Speedups

- D. Konstantinov - run Valgrind on Pythia → discovered many issues
- Many `lexical_cast` calls to PDF sets
- Elimination of these + other improvements give large gain in speed for LHCb Simulation



Opportunity: Forced Hadronization (With thanks to P. Ilten)

- At LHCb, most common method is to generate min bias events until signal decay is found
- Look at all existing signal files, ask how many signal per minimum bias event
- Using forced hadronization, would get 1/multiplicity speedup
- On average, could gain 2.7×10^4 in timing

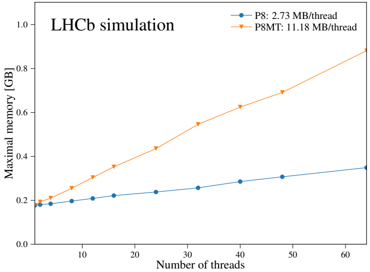
PDG ID	name	mult	speedup	decfiles
4224	Sigma_c*++	8.9e-04	1.1e+03	1
4114	Sigma_c*0	8.6e-04	1.2e+03	1
4112	Sigma_c0	5.3e-04	1.9e+03	2
10431	D_s0*+	7.3e-05	1.4e+04	5
10441	chi_c0	5.3e-03	1.9e+02	5
100443	psi(2S)	1.7e-04	6.0e+03	6
4232	Xi_c+	1.3e-03	7.6e+02	7
...				
553	Upsilon(1S)	6.8e-05	1.5e+04	7
3222	Sigma+	4.2e-01	2.4e+00	7
435	D_s2*+	4.0e-05	2.5e+04	8
445	chi_c2	6.5e-03	1.5e+02	8
200553	Upsilon(3S)	1.6e-05	6.2e+04	8
5232	Xi_b0	8.5e-05	1.2e+04	32
310	K_S0	3.2e+00		35
...				
5132	Xi_b-	9.0e-05	1.1e+04	69
411	D+	7.9e-02	1.3e+01	100
431	D_s+	2.8e-02	3.6e+01	106
413	D*(2010)+	5.4e-02	1.8e+01	126
541	B_c+	2.1e-05	4.8e+04	415
5122	Lambda_b0	6.8e-04	1.5e+03	435
531	B_s0	1.8e-03	5.4e+02	676
511	B0	8.0e-03	1.2e+02	856
521	B+	7.9e-03	1.3e+02	984

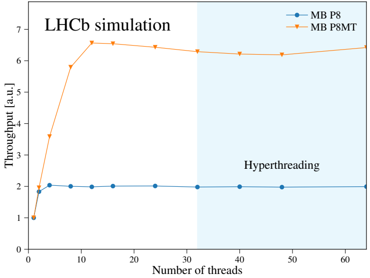
current <time>/event [s]	1.7e+02
nominal <time>/event [s]	6.4e-03
<speedup>	2.7e+04

Opportunity: Multithreading with Gaussino

LHCb-PROC-2019-010

- Future of LHCb simulation: Gauss on Gaussino
- Core principles of Gaussino
  - LHCb independent core framework
  - Build on modularity of Gauss
  - Incorporate task-based parallelism of Gaudi
  - Interface to Geant4 and Pythia8
- First developments show promising results
  - 2016 LHCb Conditions before previous slide's improvements
  - Blue: Shared Pythia 8 configuration
  - Orange: Thread local Pythia8 configuration

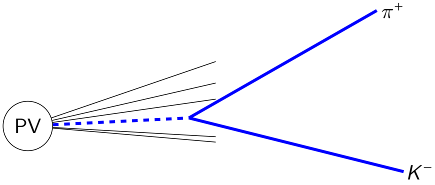




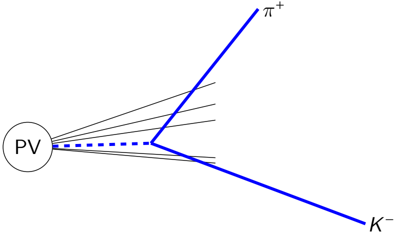
ReDecay

Eur. Phys. J. C (2018) 78:1009

- Goal: use more efficiently CPU used per event
- Method: split particles into two groups: those involved in signal process and those from the rest of the event
  - Generate MC event, store signal origin and momentum
  - Remove signal and decay products, pass rest of event through the simulation framework
  - Generate signal decay, merge with rest of the event
  - Repeat previous step N_{ReDecay} times
- Spend $\mathcal{O}(90\%)$ time simulating signal
- Independent of Generator
- Note: Only useful for generating specific signals

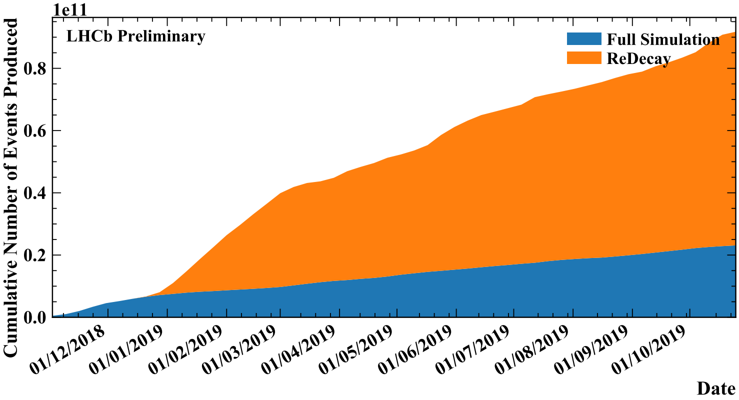


- Goal: use more efficiently CPU used per event
- Method: split particles into two groups: those involved in signal process and those from the rest of the event
  - Generate MC event, store signal origin and momentum
  - Remove signal and decay products, pass rest of event through the simulation framework
  - Generate signal decay, merge with rest of the event
  - Repeat previous step N_{ReDecay} times
- Spend $\mathcal{O}(90\%)$ time simulating signal
- Independent of Generator
- Note: Only useful for generating specific signals



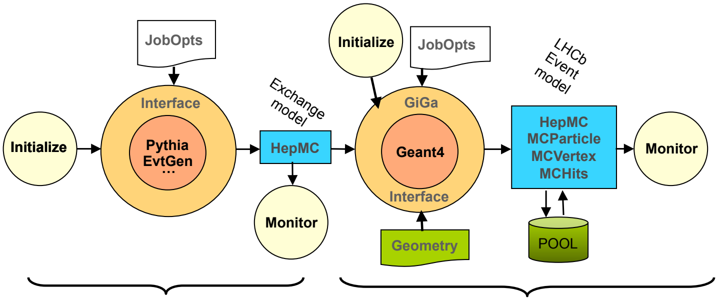
The Adoption of ReDecay

LHCb-FIGURE-2019-017



- Over the past year, ReDecay has been validated and adopted by Physics WGs
- 10-50$\times$ faster $\rightarrow$ able to generate MC for analyses requiring high statistics samples

Simulation Phase



Event Generation

primary event generator
specialized decay package
pile-up generation

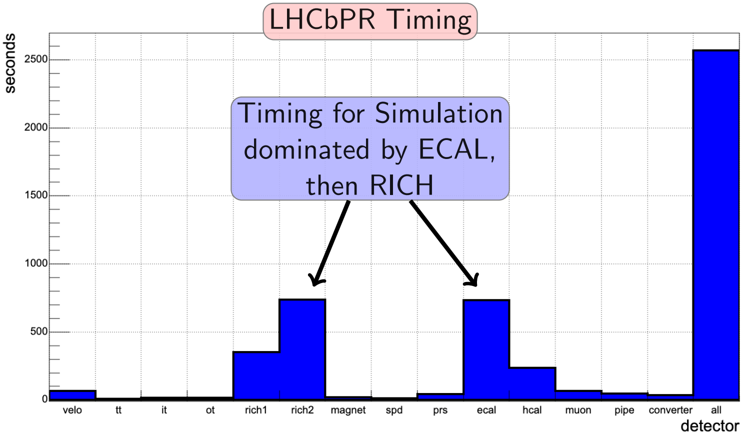
Detector Simulation

geometry of the detector (LHCb → Geant4)
tracking through materials (Geant4)
hit creation and MC truth information (Geant4 → LHCb)

- After simulation phase, events are passed in the same sequence to the propagation through material

Timing of Detector Simulation

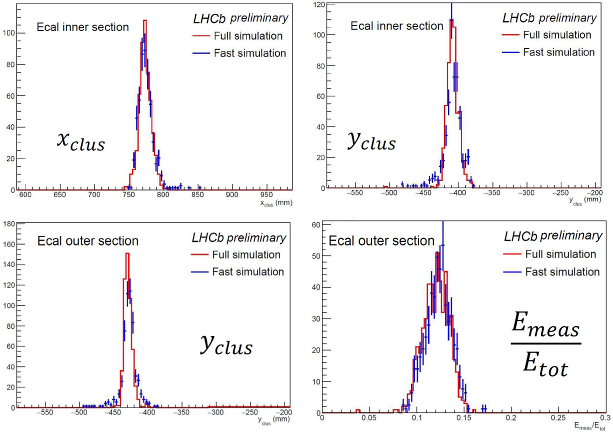
- Use LHCbPR framework to assess the timing per event for detailed simulation



Calorimeter fast simulation: Point Library

EPJ Web Conf. 214 (2019) 02040

- Idea: Provide a fast simulation option which automatically replaces Geant4 hits with a hit collection from library, as a function of discrete E, θ, ϕ "nodes"
- Not a collection of cell images, but rather energy deposit in smaller points, hence called "point library"
- Timing for lookup and transformation of points negligible



Calorimeter GAN

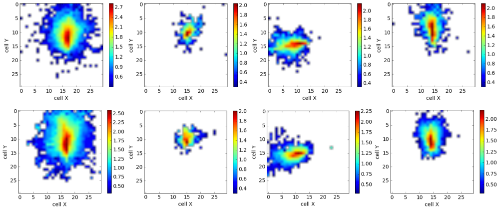
F. Ratnikov,CHEP2019

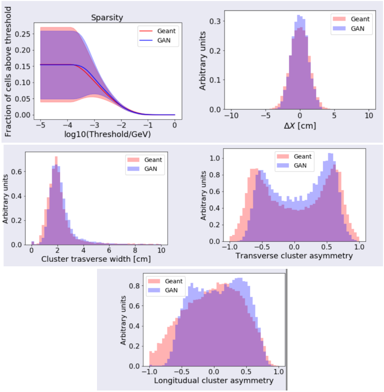
- Use GANs to generate shower image on ECAL face

GEANT Simulated

$\log_{10}$(cell energy)

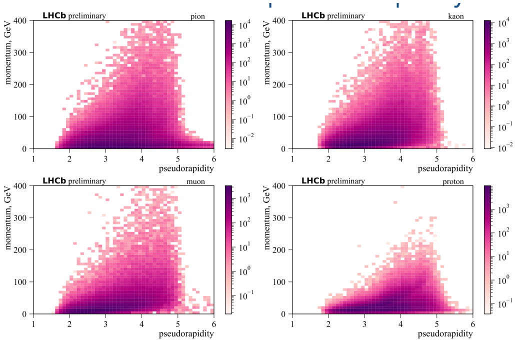
GAN Generated

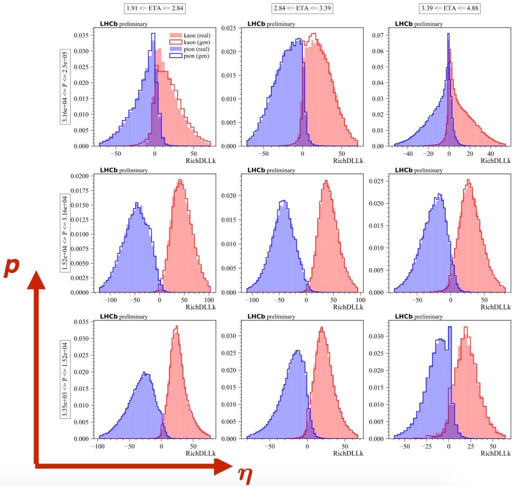




- While input distributions are well reproduced, higher level variables are not necessarily → Model needs to know about these

- Learn PID response given only particle type and kinematics
- Based on Cramer GAN, trained on calibration data

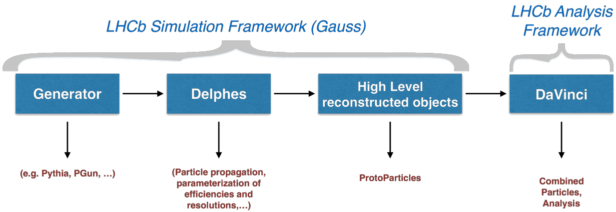




Ultra-fast parametrized simulation

A.D. CHEP2019

- "Aggressive" simulation model necessitates 20% ultra-fast parametrized simulation
- Existing solution: Delphes
  - LHCb geometry not naively supported → implemented dipole magnet instead of torroid
  - Calorimeter segmentation not cartesian → implemented new calorimeter segmentation
  - Interfacing within Gauss required extra steps



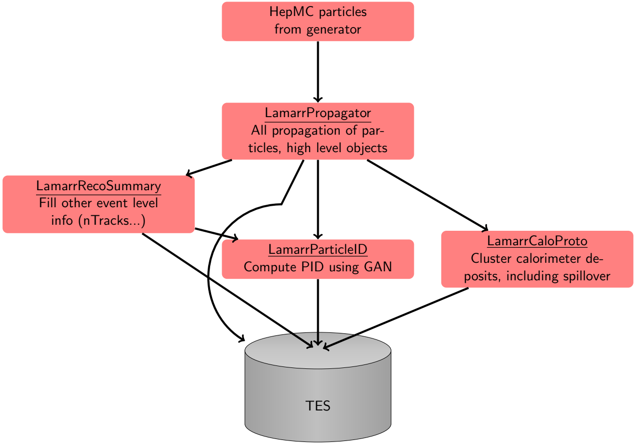
Delphes: the review

- During internal review of the adoption of Delphes, we noted that there were many duplications and complications of event processing frameworks
- The cost/benefit analysis for using Delphes within Gaudi was considered too high compared to using simple parameterization tools (see next slides)
  - Non-trivial interfacing
  - External constraints from both sides for data preparation and timing
- We therefore switch to a fully in-house implementation of parameterizations $\rightarrow$ Lamarr

Lamarr Strategy: Tracking and Clusterization

A.D. CHEP2019

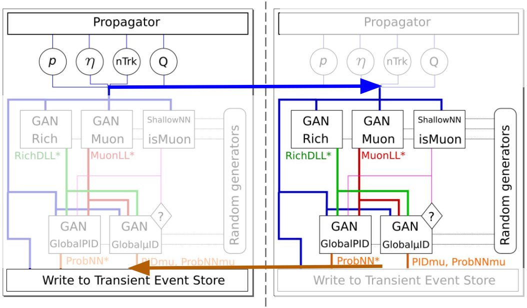
- Propagator redesigned: Propagate MC particle first to all points of interest, then smear and apply efficiencies.
- Use Inverse Cumulative Method to sample track info (χ^2_{track}, C_{ij}^{track}, fake track probability...) $\rightarrow$ mitigates binning dependence of parameterization, and large gain in speed
- Calorimeter parameterization ported from Delphes Card to simple python lists



Lamarr Stragegy: Charged Particle PID

A.D. CHEP2019

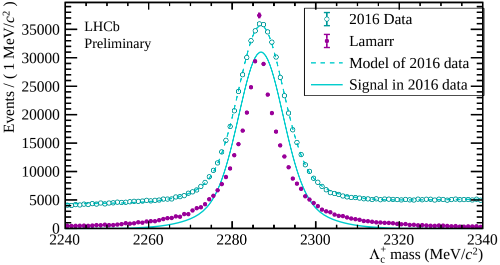
- Charged particle PID variables in LHCb can vary with occupancy and other event level variables not available at ultra-fast simulation level
- Solution: sample non-signal variables (e.g. nTracks) with random input
- Once input defined, use stacked GANs evaluated in TensorFlow to form PID information without calorimeter inputs
- Limits calls to TensorFlow to once per event

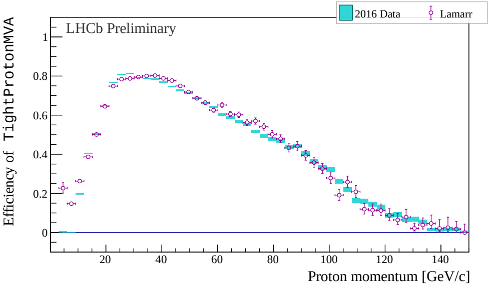


Physics Validation

[A.D. CHEP2019](), [LHCb-FIGURE-2019-017]()

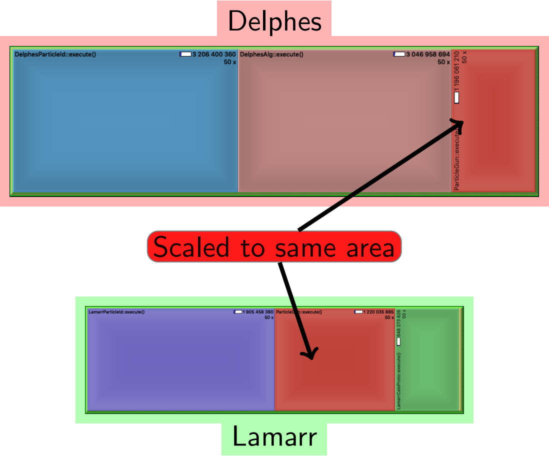
- Use 2016 Data to validate
- Example: $\Lambda_b^0 \to \Lambda_c^+ \mu^- \overline{\nu}_\mu$, $\Lambda_c \to pK^-\pi^+$



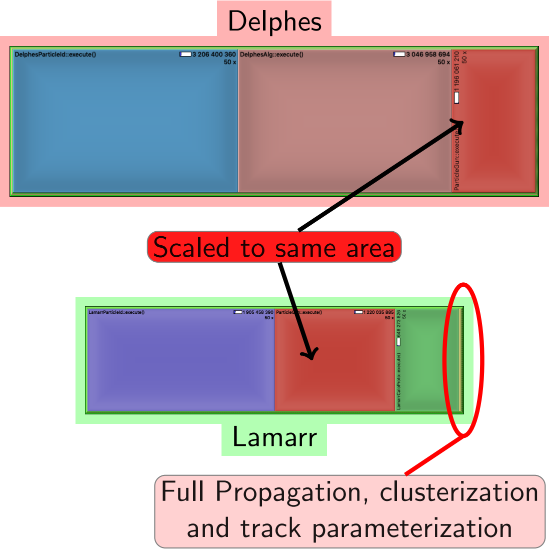


Performance

- Run Valgrind with Cachegrind on 50 $B^0 \to K^+K^-\pi^0(\to \gamma\gamma)$ events with both Delphes and Lamarr setup
- Scale call graph such that ParticleGun is the same
- With improvements, Propagation and high level particle making is now the sliver on the right hand side of the lowest graph
- Future improvements focus on:
    - Internal TensorFlow memory management
    - Faster random number generators for Calorimeter clusterization



- Run Valgrind with Cachegrind on 50 $B^0 \to K^+K^-\pi^0(\to\gamma\gamma)$ events with both Delphes and Lamarr setup
- Scale call graph such that ParticleGun is the same
- With improvements, Propagation and high level particle making is now the sliver on the right hand side of the lowest graph
- Future improvements focus on:
    - Internal TensorFlow memory management
    - Faster random number generators for Calorimeter clusterization

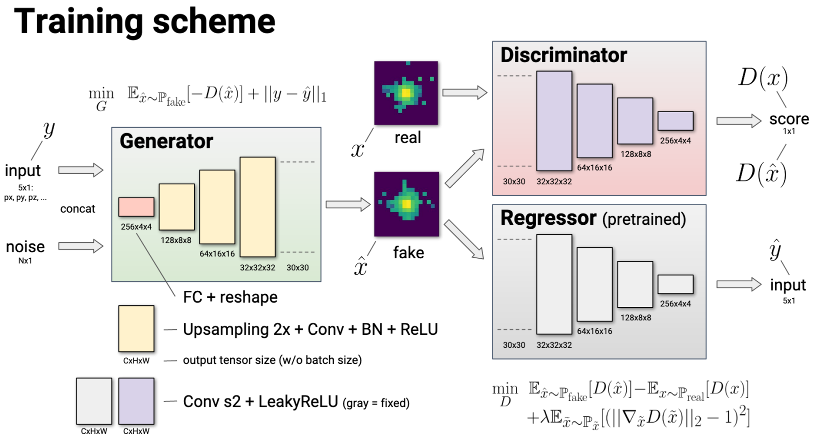


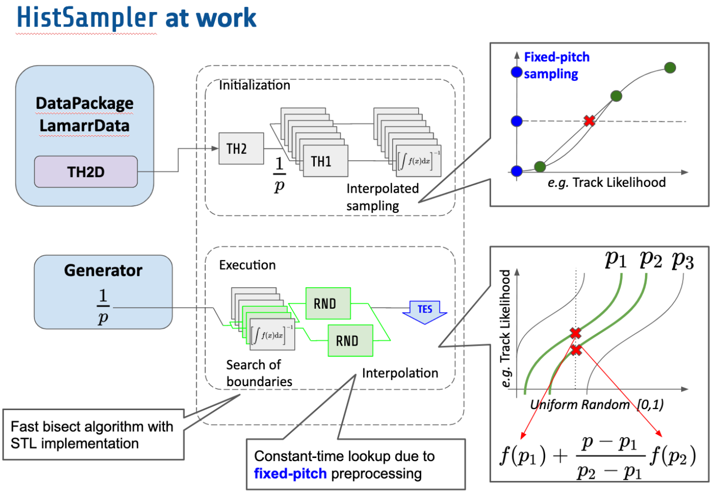
Synergies and Conclusions

- Simulation is a necessity to physics experiments
- The future computing needs of LHCb necessitate fast simulation framework options
- While LHCb simulation is specialized, synergies exist and are important
  - Generator level improvements → everyone benefits
  - Geant4 → mixing of detailed detector simulation with parametric simulation and ML techniques
  - Ultra-fast simulation is not an HL-LHC problem for LHCb → experience being gained now
  - Mix and match solutions should exist
- Exploiting synergies will benefit all

Backup Slides

CALO Gan Training




HistSampler at work
DataPackage
LamarrData
TH2D
Initialization
TH2
$\frac{1}{p}$
TH1
$\left[\int f(x)dx\right]^{-1}$
Interpolated sampling
Fixed-pitch sampling
e.g. Track Likelihood
Generator
$\frac{1}{p}$
Execution
RND
RND
TES
Search of boundaries
Interpolation
p_1 p_2 p_3
e.g. Track Likelihood
Uniform Random [0,1)
$f(p_1) + \frac{p - p_1}{p_2 - p_1} f(p_2)$
Fast bisect algorithm with STL implementation
Constant-time lookup due to fixed-pitch preprocessing

What LHCb Simulates

- For LHCb searches, need to generate specific decays for specific searches
- Example, $R(D^*)$ needs to generate not only the signal, but also specific backgrounds
- For some analyses, MC stats is the limiting systematic uncertainty

$R(D^*), \tau \to \pi\pi\pi(\pi^0)$

Phys. Rev. Lett.120(2018) 171802

Source	$\delta R(D^{*-})/R(D^{*-})[\%]$
Simulated sample size	4.7
Empty bins in templates	1.3
Signal decay model	1.8
$D^{**}\ \tau\ \nu$ and $D_s^{**}\ \tau\ \nu$ feeddowns	2.7
$D_s^+ \to 3\pi X$ decay model	2.5
$B \to D^{*-}D_s^+X$, $B \to D^{*-}D^+X$, $B \to D^{*-}D^0X$ backgrounds	3.9
Combinatorial background	0.7
$B \to D^{*-}3\pi X$ background	2.8
Efficiency ratio	3.9
Normalization channel efficiency (modeling of $B^0 \to D^{*-}3\pi$)	2.0
Total uncertainty	9.1