LHCb HLT performance- and regeression-tests A hybrid deep learning approach to vertexing

Rui Fang¹ Henry Schreiner² Mike Sokoloff¹ <u>Marian Stahl</u>¹ Constantin Weisser³ Mike Williams³

¹ University of Cincinnati

² Princeton University

³ Massachusetts Institute of Technology



January, 29th 2020

Supported by:







- All tracking detectors are being replaced, PID detectors upgraded; hardware triggers will be removed and luminosity increased → there will be a **new experiment** at Point 8 after LS2!
- Build on success of offline-quality reconstruction, alignment and calibration in Run 2, LHCb is moving to a real-time analysis approach
- Running a software trigger at 30 MHz with limited resources poses major challenges
 rethink data structure, reconstruction and selection from the ground up; monitor and document every step in the development process!
- First part of this talk discusses a new configuration framework for the trigger, and the day-to-day quality assurance and monitoring of LHCb software



- Trigger codebase is C++, adapted to functional programming
- Configuration written from scratch in python
- Highlight features: implicit deduplication and dependency deduction; algorithms globally immutable, configurability from custom python framework
- Facilitate comprehension and debugging: tools to visualize control and data flow, enhanced documentation and tutorials



• In summer, we implemented the track reconstruction including it's documentation. Afterwards, we moved and adapted tests to the new framework

- The basis of LHCb's test driven development are Q(uality)M(anagment)Tests, e.g. run PV reconstruction on reference sample and compare (efficiency) counters
- Each (non-trivial) pull/merge request is tested against full software stack
- Testing done on demand or in nightly slots using webhooks in GitLab

	0001	Rob-gaudi-head - build: 2506 (020-01-20) Company with product build: Company with start #00 and 103,456 acceleration														Browse file C										
	🔹 1 💿									Brojact @	Version	x86_64-centos7- gcc8-op1 @ Completed at 05.01-69		x86_64-centos7- gcc9-dbg @ Compiled at \$7.24.89		x86_64-centos7- clang8-opt Completed at 0208.28		x86_64-centos7- clang8-dbg g		x86_64-centos7- goc9-do0 gg Domphetad at 80:3920		x86_64+avx2+fma- centos7-goc9-opt		skylake_avx512+vecwid250- centos7-gco9-opt		
A Link Pathware Obstantin Lungdrage										News	Accessed.		build.	Platesteat		PERCENT AT	Ave all all	PLANE AND	A	Parket at	Conjunte in our		Completion in term			
									DBASE	None																
 Started reference and integration test builds. Once done, check the comparison of build and test results. 											LCG	00h	0.00		0310		0.000		0180		0.010		0000			•
											Gaudi	HEAD	build	tests	build	tests	build	tests (4)	build	tests Hi	buld	tests	build	tests (3)	build	tests (20
											Online	HEAD						tests (1)						tests (1)		beats (25
												HEAD		teets										tests RI		tests (7)
	master vs. master+this MR+Rec!1869													lests								tests				
A Behanie														teets								tests		tests RI		bests (0)
ROD-INDEXEMPT - DURIL 227 (2020-01-10) + A EDU-INDEXEMPT - DURIL 330 (2020-01-10)											Rec	HEAD														bests (2)
Proje	м	Version	x80.84-centes7-gooth-opt		x86.04-centos7-pcc8-dbg		x86.64-centos7-cleng8-ept		x86,84-centos7-clang8-cbg		Brunel	HEAD		tests (1)								tests (10)		beats (13)		tests (12)
DBA	98	None	OK		OK		OK		OK		Phys	HEAD														
PARA	M	None	OK		OK		OK		OK		Moore	HEAD		14685 (Z)		14585 (2)		tests (2)		tests (3)		tests (6)		tests (4)		16515 (14)
Gau	di	monter	OK	249	OK	249	OK	249 (4)	OK	249 (4)	Analysis	HEAD										tests (1)				
Onlin	re	moster	OK	68 (2)	OK	68 (2)	15	68 (2)	37	68 (3)	Stripping	HEAD		10585 (1)		10585 (1)		tests (1)		tests (1)		10585 (1)		tests (1)		tests (1)
LHC	16	moster	OK	247	OK	247	OK	247	OK	247	DeWinci	HEAD		tests				bests		tests (2)		tests (2)		tests (1)		bests (1)
Lbee		moster	OK	1	OK	1	OK	1	OK	1	Panoramix	HEAD		tests		tests	build (6)	tests (0)		tests (0)		tests		tests		Sests
Rec	40	master	OK	14	OK		OW	14	OW	14	Bender	HEAD		Nets (3)		Nests (3)		tests (3)		Tests (4)		1658.93)		tests (3)		bear a
		manter	04	14	on	.4	01	14	OK	14	MooreOnline	HEAD		10685 (1)		\$868. (1)		tests (1)		tests (1)		tests (1)		tests (1)		
100	~		UK	33 .	UK	33	UK	33	UK	33	Panoptes	HEAD														SUSTS

- Several flavours of nightly builds (platform, git brances, compiler, e.g. clang sanitizer etc.)
- Dedicated build slots are picked up for performance- and regeression-tests

- Scheduled tests, e.g. resource consumption or pyhsics perf., in controlled conditions
- Important measure for the upgrade: throughputs of the whole trigger system
- Currently testing 5 different settings for HLT1, two for HLT2 on 3 slots each night



- Points on time evolution plot correspond to throughput tests, each with detailed info
- Right: summary plot of a *profiling* job running directly after the throughput test

- Results are vizualized in a flamegraph [ACM Queue, 14(2), 91]
 - A stack trace is represented as a column of boxes
 - Each box represents a function (a stack frame)
 - Width of each box shows frequency at which that function was present in the stack traces
 - The background color for each box is not significant
 - Flamegraphs are stored in interactive svg format
- Many colleagues now use the performance- and regression-scripts to test throughputs and do profiling offline



- Since Nov. '19 LHCb has two on-call responsibles with different roles for upgrade software maintainance
- The *maintainer* (on-call for 3 months) ensures the long term health, coherence, consistency, and quality of software, and supports shifters
- The *shifter* (on-call for 2 weeks) ensures quality and timely merging of merge requests by driving the review process



```
https://about.gitlab.com/stages-devops-lifecycle/
```

- In practice this means not only code review, but includes assignining nightly slots, testing, understanding the outcome and reporting back to the developers
- Much is communication and logistics: In my 2 week shift, we reviewed 107 merge requests, 50 of which were merged. There have been several days where we were at capacity with nightly slots and tests on demand
- The feedback from the collaboration has been very positive

- Main challenge for PV finding: number of visible PVs will increase from \sim 1.1 to 5.6 in upgraded LHCb
- Efficiency mainly driven by cluster search \rightsquigarrow use machine learning
- The project is standalone, and uses toy data and it's own proto-tracking. This is the workflow in a nutshell:



- Reduces sparce 3D data (41M pixels) to feature-rich 1D data kernel densities in z
- Start from (LHCb Velo) tracks, *i.e.* closest to beam (*x*, *y*, *z*) position, slopes (*t_x*, *t_y*), (covariance matrix)
- Goal: find kernel maximum in (x, y) in each of the 4000, 100 μm wide z-bins. Assign it as z kernel value
- Currently, this is done by a coarse manual search followed by a MINUIT minimization

• Kernel $\mathcal{K}(z) := \frac{\sum_{\text{tracks}} \mathcal{G}(IP_{x,y}|z)^2}{\sum_{\text{tracks}} \mathcal{G}(IP_{x,y}|z)} - \sum_{\text{tracks}} \mathcal{G}(IP_{x,y}|z)$, where \mathcal{G} is the product of Gaussian p.d.f.s in x and yevaluated at the (x, y) impact parameters to a hypothesized vertex.

 \mathcal{G} is centered around 0 and has a heuristic width estimating the IP_{x,y} uncertainties.



- Feed kernels to a convolutional neural network, implemented in PyTorch Further improvement possible using squared z-kernel and adding (x, y) perturbatively
- Cost function^{*} modified with asymmetry parameter^{**}, which serves as powerful control to balance efficiency to false-positive rate
 * inspired by cross entropy plus minimal offset *e* to *y* and ŷ





• Proof of principle established

Improved networks show increase in efficiency and \sim factor 2 decrease of FP rate

• Next step is benchmarking it's performance in the LHCb HLT

- PVFinder is standalone, written mostly in python and has own data structure
- Need C++/CUDA versions adapted to LHCb (with ability to write data for standalone PVFinder)
- Kernel generation, persistency and CPU inference-engine have now been deployed in HLT1
- It runs on output of a new Velo tracking algorithm [arXiv:1912.09901]
- Decided for TorchScript as C++ inference engine

frameworks such as Caffe2, Microsoft Cognitive Toolkit, and MXNet. ONNX is still supported and actively worked in PyTorch v1.x, but it appears that TorchScript is the preferred way for model exporting. See the "Further Reading" section for more details on ONNX if you're interested.

Programming PyTorch for Deep Learning, I. Pointer, ISBN: 9781492045342

WARNING

At the moment, the C++ API should be considered "beta" stability; we may make major breaking changes to the backend in order to improve the API, or in service of providing the Python interface to PyTorch, which is our most stable and best supported interface.

https://pytorch.org/cppdocs/

• We are currently collecting and evaluating results

Those from running a model trained with toy data on official LHCb MC look promising!

- Update concerns width of kernel-Gaussians \mathcal{G} . (Constant up to $\chi^2 = 6$, then a linear term was added)
- We now run on ouput of production Velo tracking → use measured covariance matrix. This results in a precise uncertainty estimate for the impact parameters and their correlation - G becomes a bivariate Gaussian
- New kernels look more pronounced in PV regions
- Overall kernel generation too slow
 ⇒ train another (C)NN that generates kernels





- outlook **Conclusions and**
- A new configuration framework for the trigger, the day-to-day quality assurance and monitoring of LHCb software has been presented

- A hybrid deep learning approach to vertexing PVFinder has been introduced
- PVFinder is on it's way into LHCb's HLT, and can now be trained using official LHCb MC
- Some final work needed to enable benchmarking the performance with standard LHCb tools
- We can then apply the lessons learned in the past to tune PVFinder
- Several ideas for improvement. Priorities need to be defined after benchmarking