





traccc

Track Reconstruction on GPUs in Acts

Attila Krasznahorkay on behalf of **a lot of** people...

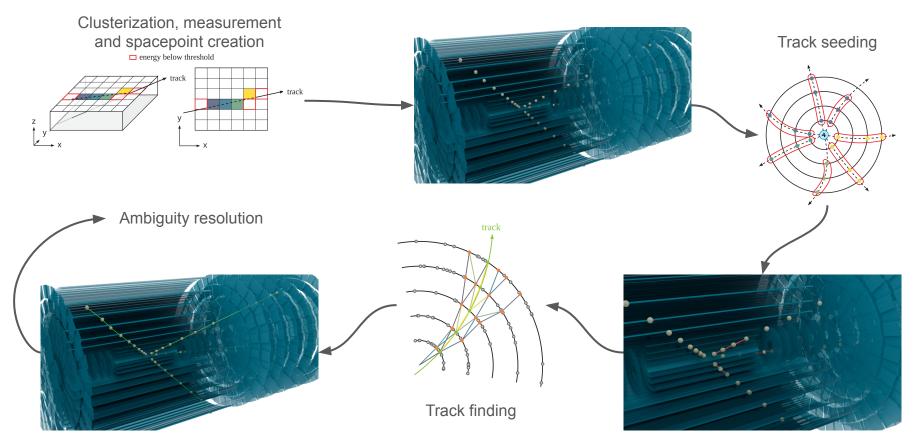




HEP Software Foundation

(Classical) Track Finding 101

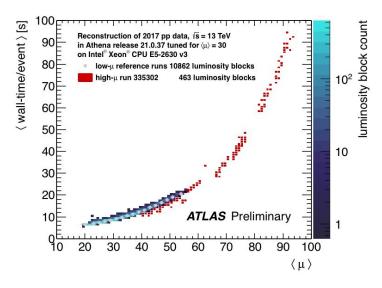


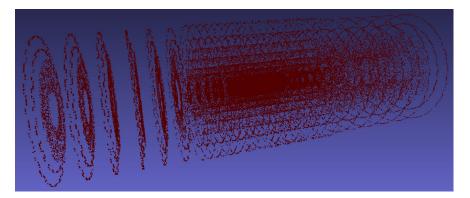


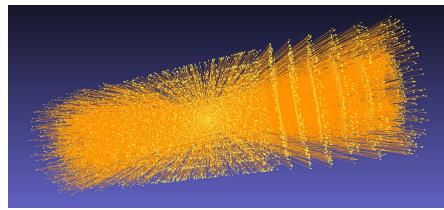
The Need For GPUs



- The sort of events that we will need to reconstruct during the HL-LHC, are the ones shown here
 - On which the combinatorics of our algorithms explode

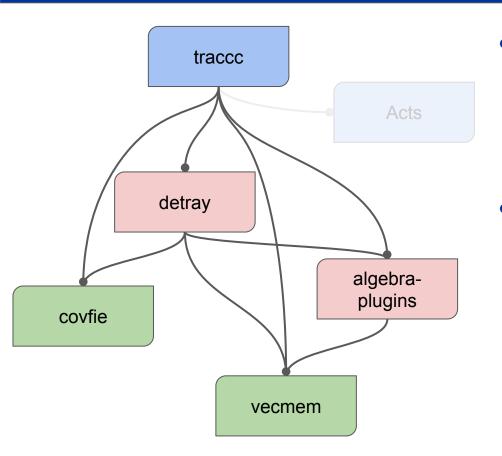






tt event in the ODD at μ = 200

The Acts Parallelization R&D



- To explore fundamentally new ways for reconstructing particle tracks, we created a set of "standalone" projects
 - With the top reconstruction algorithms sitting in traccc, and all other projects serving various purposes for making that happen
- The overall goal is to demonstrate that we could run track reconstruction on GPUs without any shortcuts in reconstruction / physics quality
 - Using the same (type of) combinatorial Kalman filtering used by <u>Acts</u>, with detector geometry and magnetic field modeled at the same level of accuracy

Base Projects

ecme	em / core / include / vecmem / edm / 口
۲	krasznaa Added a constant data type to vecmem::edm::container.
Nam	e
	details
	impl
ß	buffer.hpp
Ľ	container.hpp
Ľ	data.hpp
Ľ	device.hpp
Ľ	host.hpp
Ľ	schema.hpp
Ľ	view.hpp

- Good technical work has happened in <u>vecmem</u>, <u>algebra-plugins</u> and <u>covfie</u>
 - But those are not the main things for today..
- vecmem introduced basic support for SoA containers
 - But they did not make their way into traccc yet
- <u>algebra-plugins</u> improved its vectorization support in host code
 - Both for auto- and explicit-vectorization

✓	
191 ALGEBRA_HOST_DEVICE	191 ALGEBRA_HOST_DEVICE
<pre>192 inline bool operator==(const transform3 &rhs) const {</pre>	<pre>192 inline bool operator==(const transform3 &rhs) const {</pre>
194 - for (size_type i = 0; i < 4; i++) {	194 + for (size_type j = 0; j < 4; j++) {
195 - for (size_type j = 0; j < 4; j++) {	195 + for (size_type i = 0; i < 4; i++) {
196 if (matrix_actor().element(_data, i, j) !=	196 if (matrix_actor().element(_data, i, j) !=
197 matrix_actor().element(rhsdata, i, j)) {	<pre>197 matrix_actor().element(rhsdata, i, j)) {</pre>

Current contributors:

Joana Niermann, Beomki Yeo, Stephen Swatman

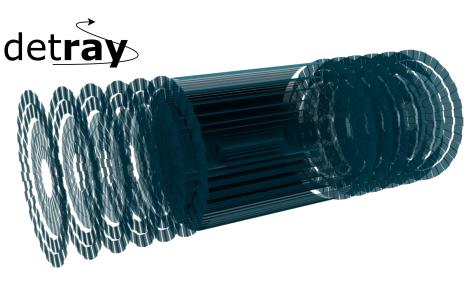
detray



- Is maybe our most ambitious project
- It provides a surface based geometry for tracking, with efficient navigation / propagation support between the surfaces
 - Including the management of surface material and magnetic field during the navigation

All implemented without using "GPU hostile" programming methods

• Virtual inheritance, dynamic memory allocation, etc.



Current contributors:

Joana Niermann, Beomki Yeo, Andreas Salzburger, Frederik Verdoner Barba, Eleni Xochelli, Stephen Swatman

Latest Developments



• After updates in Acts and <u>ODD</u>, created JSON descriptions of the ODD for Detray

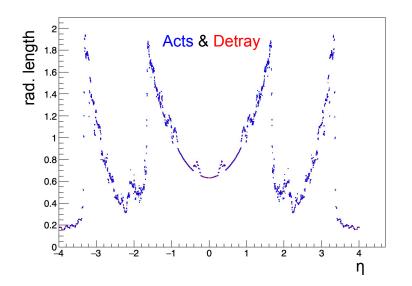
- Including the properly defined "surface grids" and "material maps"
- Still a little manually for these tests, but will make it a lot more automatic soon. Making it possible to convert any "Acts geometry" to a Detray one.

Can now exactly reproduce the behaviour of Acts's existing tracking geometry code

- Material mapping comparisons on device to some soon, current comparisons all done in host code.
- Tons of technical developments done to make it all happen...



ODD surfaces and grids



traccc

Category	Algorithms	CPU	CUDA	SYCL	Alpaka	Kokkos	Futhark
Clusterization	CCL / FastSv / etc.						~
	Measurement creation						Image: A start of the start
Seeding	Spacepoint formation				•		
	Spacepoint binning				 Image: A set of the set of the		
	Seed finding				Image: A start of the start		
	Track param estimation		v				
Track finding	Combinatorial KF			•			
Track fitting	KF						
Ambiguity resolution	Greedy resolver						

✓: exists, ●: work started, ●: work not started yet

Current contributors:

Beomki Yeo, Joana Niermann, Ryan Joseph Cross, Stewart Martin-Haugh, Shima Shimizu, Sylvain Joube, Stephen Swatman

- It is our main repository, combining the capabilities of all of the other ones
 - GPU code development initially happens in <u>CUDA</u> most of the time, then generalising it to work with <u>SYCL</u>, <u>Alpaka</u>, etc. as well.
- As was the original goal, significant code sharing is achieved between the host and device, and the different device implementations
 - Technically in all cases happening through shared, inlined functions (working on "GPU friendly" data types)



Reconstruction Algorithm Status

devico > alpa > com

∼ cuda ∼ inc

> > fi > se

> u > src

M CM

> kokł ~ svcl

> √ incl > cl

> > ✓ fil
> > G+ I
> > > se

> u

M CM

CERN

- The full ODD reconstruction chain now works on the host and with CUDA!
 - Without ambiguity resolution... For that we still only have an algorithm for the host.
 - Technically the "full CUDA chain" can fit on a single screen!
- Geant4 simulation files for its input can now be produced using Acts's main branch
 - See: <u>acts-project/acts#3169</u>

	136	full_chain_algorithm::output_type full_chain_algorithm::operator()(
e	137	<pre>const cell_collection_types::host& cells,</pre>
ka	138	<pre>const cell_module_collection_types::host& modules) const {</pre>
ka	139	
mon	140	
3	141	cell_collection_types::buffer cells_buffer(cells.size(),
- 	142	<pre>*m_cached_device_mr);</pre>
lude/traccc/cuda	143 144	<pre>m_copy(vecmem::get_data(cells), cells_buffer)->ignore(); cell ==dule_pellection_turney.buffer_medules_buffer(cedules_pirs())</pre>
lusterization	144	cell_module_collection_types::buffer modules_buffer(modules.size(), *m_cached_device_m
nding	146	m_ccopy(vecmem::get_data(modules), modules_buffer)->ignore();
	147	
finding_algorithm.hpp	148	
tting	149	<pre>const clusterization_algorithm::output_type measurements =</pre>
eeding	150	<pre>m_clusterization(cells_buffer, modules_buffer);</pre>
	151	<pre>m_measurement_sorting(measurements);</pre>
tils	152	
	153	// Run the seed-finding (asynchronously).
lakeLists.txt	154 155	<pre>const spacepoint_formation_algorithm::output_type spacepoints = m_spacepoint_formation(measurements, modules_buffer);</pre>
	155	<pre>const track_params_estimation::output_type track_params =</pre>
ark	157	<pre>m_track_parameter_estimation(spacepoints, m_seeding(spacepoints),</pre>
kos	158	m_field_vec);
	159	
	160	
lude / traccc / sycl	161	<pre>if (m_detector != nullptr) {</pre>
lusterization	162	
	163	// Create the buffer needed by track finding and fitting.
tting	164	<pre>auto navigation_buffer = detray::create_candidates_buffer(to detector</pre>
fitting_algorithm.hpp	165 166	*m_detector, m_finding_config.navigation_buffer_size_scaler *
eedina	167	<pre>m_copy.get_size(track_params),</pre>
	168	<pre>*m_cached_device_mr, &m_host_mr);</pre>
tils	169	
	170	
1akeLists.txt	171	<pre>const finding_algorithm::output_type track_candidates =</pre>
IakeLists.tkt	172	m_finding(m_device_detector_view, m_field, navigation_buffer,
	173	<pre>measurements, track_params);</pre>
	174	
	175 176	// Run the track fitting (asynchronously).
	176	<pre>const fitting_algorithm::output_type track_states = m_fitting(m_device_detector_view, m_field, navigation_buffer,</pre>
	178	track_candidates);
	179	
	180	
	181	output_type result{&m_host_mr};
	182	<pre>m_copy(track_states.headers, result)->wait();</pre>
	100	roturn rocultu

Host <- > Device Agreement(?)



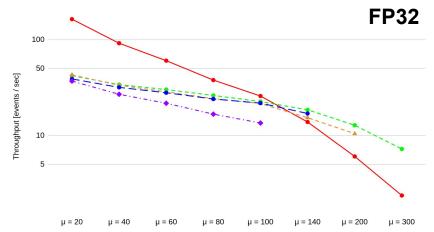


- Number of measurements: 637 (host), 637 (device) 100% at 0.01% uncertainty **FP64** 100% at 0.1% uncertainty 100% at 1% uncertainty - 100% at 5% uncertainty Number of spacepoints: 637 (host), 637 (device) 100% at 0.01% uncertainty 100% at 0.1% uncertainty - 100% at 1% uncertainty 100% at 5% uncertainty Number of seeds: 96 (host), 96 (device) 100% at 0.01% uncertainty - 100% at 0.1% uncertainty - 100% at 1% uncertainty 100% at 5% uncertainty Number of track parameters: 96 (host), 96 (device) 100% at 0.01% uncertainty 100% at 0.1% uncertainty 100% at 1% uncertainty 100% at 5% uncertainty lumber of track candidates (header): 108 (host), 108 (device) 100% at 0.01% uncertainty - 100% at 0.1% uncertainty 100% at 1% uncertainty - 100% at 5% uncertainty Track candidates (item) matching rate: 100% Number of track states: 108 (host), 108 (device) Matching rate(s): 100% at 0.01% uncertainty - 100% at 0.1% uncertainty - 100% at 1% uncertainty
- 100% at 5% uncertainty

- Our main "development applications" are ones executing the algorithms one by one, checking their outputs at every step
 - Allowing us to measure the "physics" Ο performance" of the code, and to compare results between different implementations of the same algorithm
- At FP32/single precision, agreement between the host and GPU is not perfect. But it's also not terrible.
 - While at FP64/double precision the GPU 0 code finds the exact same tracks, with the exact same properties. 10

ODD Reconstruction Compute Performance

AMD EPYC 7413 (48 CPU threads)
 NVIDIA RTX A5000 (2 CPU threads)
 NVIDIA RTX A4000 (2 CPU threads)
 NVIDIA RTX3080 (2 CPU threads)
 NVIDIA RTX2060 (2 CPU threads)

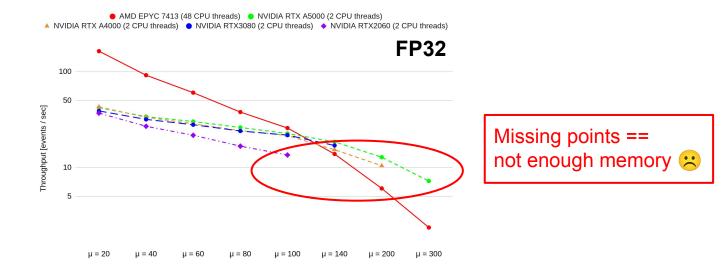


• We also have tests that load N events into (host) memory, and process them over- and over again to test the throughput of our algorithms

• Just copying stuff back to the host at the end, but not analyzing the output of the reconstruction

• Even with the so far hardly optimized algorithms, we can beat a single "decent" CPU with a single "workstation" GPU at HL-LHC luminosities

ODD Reconstruction Compute Performance



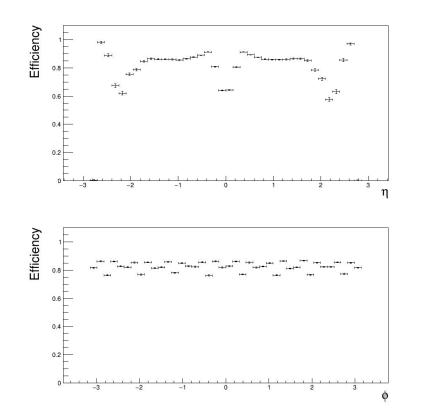
 We also have tests that load N events into (host) memory, and process them over- and over again to test the throughput of our algorithms

• Just copying stuff back to the host at the end, but not analyzing the output of the reconstruction

• Even with the so far hardly optimized algorithms, we can beat a single "decent" CPU with a single "workstation" GPU at HL-LHC luminosities

ODD Reconstruction Physics Performance

- Makes it very clear that all compute performance numbers are to be taken with some salt
 - These efficiencies (for high- p_T muons) should be ~100%. We will make sure that they would be.
- With this in mind, such efficiencies without any ODD specific settings for our code, are not a terrible starting point





The Bugs / Next Steps



- With the full chain only starting to work a few weeks ago, and only running on larger simulation samples now (this week) for the first time, we are finding a lot of errors still...
 - I'm not too worried about this though
- We will need to demonstrate that the algorithms can find tracks in the ODD efficiently
 - Already identified a few places where our default algorithm configurations don't seem to work well
 - Making proper use of material maps during reconstruction will also help
- Will need to make the code work with ATLAS's HL-LHC inner detector geometry (<u>ITk</u>)
 - With the infrastructure developed with the ODD geometry, this should be a finite amount of effort
- We will switch to a fully-SoA Event Data Model from the current, naive AoS one
- Implement the missing algorithms with <u>CUDA</u>, <u>SYCL</u>, <u>Alpaka</u>, etc.
- Integrate everything into Acts!
 - With a unified UI with all the existing / CPU tools

Summary



- I believe the future is bright for Act's GPU capabilities!
 - The very first version of the code that works on the ODD (v0.10.0), has a lot to improve still
 - However the performance, as is, makes me very hopeful already!
- Much of the current code is held together by sellotape, spit and blind luck...
 - But we have a plan for making it all a lot more robust, and (hopefully) significantly faster
- A lot of work already done, and a lot of good work still ahead of us!





ODD Reconstruction Compute Performance

CER	NN
N	X

Device	ttbar event processing rate [events / sec]									
Device	μ = 20	μ = 40	μ = 60	μ = 80	μ = 100	μ = 140	μ = 200	μ = 300		
AMD EPYC 7413 (48 CPU threads)	163.71	91.8513	60.359	37.8601	25.8034	13.8167	6.03643	2.35974		
NVIDIA RTX A5000 (2 CPU threads)	42.0662	33.9328	30.1514	26.1469	22.6047	18.5172	12.7826	7.21733		
NVIDIA RTX A4000 (2 CPU threads)	42.8472	33.4305	28.555	24.2146	21.5356	15.314	10.4362			
NVIDIA RTX3080 (2 CPU threads)	38.9144	31.7598	27.9324	24.0226	21.7591	16.9548				
NVIDIA RTX2060 (2 CPU threads)	36.941	26.9102	21.679	16.6888	13.4879					

Throughput Measurement Profile

NVDA Najsht Systems 2023.4.4 _ D ×										
te Awa Tony Rep										
odu _ministeriore										
E Timeline View + @ Options										🖼 🔍 👘 tx 🔥 2 manings, 19 messages
11s • +545ms +550ms +555ms	s +560m	rs +565	Sms +	570ms	+575ms	+580m	s +58	35ms	+590ms	+595ms +600ms +605ms +610ms +615ms +620ms +622ms +622ms +632ms +635ms +640ms +645ms +655ms +655ms +655ms +660ms +665m*
+ CPU (48)										
- CUDA HW (0000.01:00.0 - NVIDIA RTX A5000)	-									
Memory usage	45									
Static memory usage										
Local Memory Pool										
• 97.8% Kernels • 71.3% propogate to next surface	void tracec.o	uda kernels prop	pegate_to_n_	-void tracco.cui	da :kernels :pro	opogate_to_next	surfacevi	old tracec.cud	la kernela propaga	L. evid trace: code. You't rece: code. You't rece: code. Xennet: Service: Annuel: Service:
20.4% fr: 20.4% fr:	WHO EDICOLO	ada, Merricia, proc	Conduction (and a master cost	au Activeto pro	pagaretophen	Cacillater (M	000000000000	a wenes propage	
> 2.3% find_tracks				1			-			
+ 1.1% ccl_kernel							-		A	
24 kernel groups hidden+	u								1	it it is a second se
~ 2.2% Memory	L									<u></u>
3.2% Memoet				_	-					
				. E				_		
Clusterization, mea	asur	rem	ien	t 🗏				Tr:	ack	finding Track fitting
	2001			۲ –						inding
creation, seedir		oto								
	ıy, t	ະເບ.		_						Event mocessing for 201 al
					cudaMomep	pyAsync		cudaM	iemepyAsync	
Profiler overhead				_						
8 threads builden -+										
				_						
Stats System View V										
CUDA API Summary	Time + To	tal Time In	istances A	wg M	ted 1	Min I	viax :	StdDev	Category	Operation *
CUDA API Trace CUDA GPU Kernel Summary	69.8%	6.497 s	2108	3.082 ms 8	855.908 µs	48.993 µs	16.055 ms	4.154 ms	CUDA_KERNEL	
CUDA GPU Kernel/Grid/Block Summary CUDA GPU MemOos Summary (by Size)	19.9%	1.857 s			16.967 ms	13.783 ms	21.109 ms	1.692 ms	CUDA_KERNEL	void traccc:cuda:kemels:fit-traccc:kalman.fiter-detray:rk.stepper-covide:field_view-covife_backend_constant=covife_vector:vector_d=float, (unsigned long)3+, covife_vector:vector_d=float, (unsigned long)3+, detray:counter-float-,
CUDA GPU MemOps Summary (by Time)		205.813 ms	2108		41.824 µs	6.688 µs	348.993 µs 545.570 µs	91.491 µs	CUDA_KERNEL MEMORY_OPER	vial trace.cc.udi.kmmlsf.rhd.tacks.comti detray-detector-detray-detector-detray-detaut_metadata_detray-topena_extens_types-vecmem_device_vector, detray-tapite_detray-daray-vector_detray-daray-ktracec_finding_config=float-v[T2, T1 = inter R. Didda mensero winds to Deviced
CUDA GPU Trace CUDA Kernel Launch & Exec Time Summary		103.931 ma			37.904 µs 184.180 µs	417 ms	1.695 ms			(2004) memory host-solverse) taccoccude transfer of kernels col kernels where the solverse is a solverse transfer of kernels and kernels and kernels col kernel
CUDA Kernel Launch & Exec Time Trace		87.242 ms		41.385 µs	5.392 µs	1.120 µs	241.313 µs		CUDA_KERNEL	tacc: cuda kernels add, links, for holes/vecmem: data vector, view-const unsigned inty, vecmem: data vector, view-const defray; bound, tack, parameters-defray; cmath-floats->, vecmem: data vector, view-const tracc: candidate_link>, vecmem: data
CUDA Summary (APUKernels/MemOps) DX11 PIX Range Summary	0.9%	86.766 ms	110	788.785 µs 8	807.060 µs	538.179 µs	1.024 ms	128.972 µs	CUDA_KERNEL	trace: cuda:kernels:count_triplets(trace: seedinder_config_detay:const_grid2_view-detray:grid2vdetay:attach_populator, detray:asis2.vicular, detray:asis2-regular, detray:serializer2, detray: detray: detray: detray:detray detray: detray:detray.
DX12 GPU Command List PIX Ranges Summary DX12 PIX Range Summary		55.866 ms					626.883 µs		CUDA_KERNEL	tracce: cuda: kernels: find_doublets[tracce: seedfindler_config_detray:const_gnid2_veev-detray: attach_populator, detray: axis2_circulat, detray: axis2_regulat, detray: axis2_regulat, detray: avertation: detray: degred_vector, detray:
MPI Event Trace NVTX GPU Projection Summary		42.631 ms		eer eer pe	420.242 µs	correcto po	482.882 µs	a	CUDA_KERNEL	void fraces: suda: kernels: build_tracks-traces: finding.config=float>>(T), vecemem:.data: yeetor, view-const traces: measurement>, vecmem:.data: vector, view-const detray bound, track, parameters-detray: cmath=float>>, vecemem:.data: jagged_vector, v
NVTX GPU Projection Trace		35.270 ms	12190	2.893 µs	960 ns 39.040 us	927 ns 31.616 us	245.249 µs	19.272 µs	MEMORY_OPER	[0/DB memory Device-H-Hot] visit dub UD_2010 520 NE-DeviceMenaSorMenaKemeH-flool0 cub UB 20100 520 NE-DeviceMenaSorMena
NVTX Push/Pop Range Summary NVTX Push/Pop Range Trace		34.360 ms				P	48.832 µs			ven die Dut 2010 20 July 15 Develowingsontweigskennen (ool), dut Dut die 2010 2010 5: Develowingsontherspektrese) (ook) de zuroppolitiken (ook) de zur
NVTX Range Kernel Summary NVTX Range Summary		16.151 ms					174.977 µs		CUDA_KERNEL	and countering of the second counterplace countering of the second county of the counterplace of the second counterplace of the s
NVTX Start/End Range Summary Network Devices Congestion		15.977 ms	110	145.249 µs 1	144.689 µs	102.177 µs	201.441 µs	26.240 µs	CUDA_KERNEL	taccc: cuda kernels grave backs(inccc: container view-const detay bound; track_parameters/detay: small+float>, const traccc: messarement; werment; data:wetor_view-const unsigned ins; traccc: container_view-detay_bound; track_parameters
NvVideo API Summary		15.001 ms	2108	7.116 µs	6.784 µs	3.392 µs	15.424 µs	2.893 µs	CUDA_KERNEL	trace: cuda:kernels:count_masurements/vecmem_data:vector_view=const detray:bound_track_parameters=detray:comatk=float>>>,vecmem_data:vector_view=const detray:goometry_barcode>,vecmem_data:vector_view=const unsigned int>, unsign=
OS Runtime Summary OpenACC Summary		12.424 ms	2108	5.893 µs	3.841 µs	2.624 µs	16.384 µs		CUDA_KERNEL	void traces:usda:kemelia.apply_interaction=const.detray-detector-detray-detector-detray-detector-detray-container_types=vecmem:dev/ce_vector_detray-tuple, detray-darray-vecmem:appled_dev/ce_vector_detray-tuple, detray-darray-vecmem:appled_dev/ce_vector_detray-tuple, detray-darray-vecmem:appled_dev/ce_vector_detray-tuple, detray-darray-vecmem:appled_dev/ce_vector_detray-tuple, detray-darray-vecmem:appled_dev/ce_vector_detray-tuple, detray-darray-vecmem:appled_dev/ce_vector_detray-tuple, detray-darray-vecmem:appled_dev/ce_vector_detray-tuple, detray-darray-vector_detray-tuple, detray-tuple, detray-tuple, detray-tuple, detray-tuple, detray-tuple, detray-tuple, detray-tuple, detray-tuple, detray-tuple, detray-t
OpenGL KHR, debug GPU Range Summary OpenGL KHR, debug Range Summary	0.1%	6.927 ms	110			59.456 µs 53.024 µs	68.641 µs		CUDA_KERNEL	tacce: cudickernels form, spacepoints/vecmem: data: vector_view=const tracce: measurement, vecmem: data: vector_view=const tracce: cell_module_uragenel int, vecmem: data: vector_view=tracce: spacepoints/
OpenMP Summary Unified Memory Analysis Summary	0.1%	6.679 ms	110	60.714 µs	61.248 µs	53.024 µs	72.001 µs		CUDA_KERNEL	vel c.b. CUB_200106_2539, NE-beiceMergl6078lbckSortkennel/Loop)0, db. CUB_200100_2529, NE- DeviceMergl6071bickpristocc: massurement *-pbic/pd00, tracc: massurement *_pbic/pd00, tracce: massurement *_pbic/pd00, tracce: massurement *_pbic/pd00, tracce: mas
Unified Memory CPU Page Faults Summary Unified Memory Totals Summary	0.1%	6.475 ma	7864	823 ns	448 ns	415 ns	3.104 µs		MEMORY_OPER	
United Memory Lotas summary Vulkan API Summary	0.1%	5.806 ms	2108	2.754 µs	2.080 µs	1.215 µs	6.560 µs		CUDA_KERNEL	void frust sub-core. Jemel, agent-thrust cuda, cub:parallel.for:Para
Vulkan GPU Range Summary	0.1%	5.356 ms	2108	2.540 µs	2.592 µs	1.600 µs	4.161 µs	631 ns	CUDA_KERNEL	void cub:CUB_200100_520_NS:DeviceScanKemel-cub:CUB_200100_520_
Wilkan Range Summary WDDM Queue Utilization Summary	0.0%	4.486 ms	4216	1.064 µs	1.024 µs	959 ns	1.696 µs	112 ns		void thrust.cuda.cub.com:.kernel.agent-thrust.cuda.cub:_parallel.for:ParallelForAgent-thrust.cuda.cub:_fil:functor+unsigned int *, unsigned inte, long=YT2, T3)
and a second	0.0%	3.524 ms	1998	1.763 µs	1.760 µs	1.216 µs	2.624 µs			void thrust cuda, cub-core. kernel, agent-thrust cuda, cub:parallel.for:ParallelForAgent-thrust.cuda, cub:transform.unary,transform.lunary,transform.lunary,transform.turary
	0.0%	3.154 ms			28.304 µs	22.400 µs 15.968 µs	38.720 µs 32.672 µs	3.316 µs	CUDA_KERNEL CUDA KERNEL	traccc: coda:kenels:select_seeds(traccc:seed)(ter_config:vecmem.data:vector_viewcomst traccc:spacepoints_detray:cont_ght2_viewdetray=ght2/detray=atach_populato; detray=asis2-circula; detray=asis2-regulate; detray=seinter2, detray=context_period.eteray=teration_form_torat_period.eteray=teration_form_torat_period.eteray=teration_form_torat_period.eteray=teration_form_torat_period.eteray=teration_form_torat_period.eteray=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=setara=teration_form_torat=
	0.0%	2.937 ms	1804	26.098 µs	1.696 µs	1.088 µs	2.816 µs	4.665 µs 317 ns		You must could up concerning approximate could up and on a manifestion and could up and the
	0.0%	2.322 ms	110		21.488 µs	17.600 µs	25.984 µs	2.068 µs		trace: cuda: kernels-restmate_track, paramit/vecmem: data:vector_view-const trace: spacepoint> vecmem: data:vector_view-const trace: seeds, atd.amayefloat (unsigned long)3>, std.amayefloat (unsigned long)3>, st
CLI command: nsvs stats - r cuda.gpu_sum "/home/krssznaa/ATLAS/projects/traccc/cuda_ttbar_mu200.sqlite"	0.0%	2.266 ms	2108	1.074 µs	1.024 µs	991 ns	1.472 µs	107 ns	CUDA_KERNEL	void cub: CUB_200100_520_NS:DeviceScantIntKernel-cub::CUB_200100_520_NS:ScanTi#State-unsigned int, (loco)11v/T1, int)
vela erere a nover Wint anu nucuel krietstraet vir thesholdscreatestic chorage "Uppy" with 500 solities.										



http://home.cern