### Optimizing the ATLAS code with different profilers

#### Sami Kama<sup>1</sup> on behalf of the ATLAS Collaboration<sup>2</sup>

<sup>1</sup>Southern Methodist University, Texas

<sup>2</sup>https://cdsweb.cern.ch/record/1386334

Advanced Computing and Analysis Techniques in Physics, Beijing/China, May 2013





#### Large Hadron Collider

- Located at CERN near Geneva
- 27 km circumference and ~100 m below surface
- It is operational since 2010.
- There are 4 detectors located on it, ATLAS is one of two large general purpose detectors
- It is shutdown for two years for upgrades and maintenance on March 2013
- It will operate at higher beam energy and higher luminosity after shutdown





#### **ATLAS**



- ATLAS is composed of different co-centric cylindrical detectors of ~150 M readout channels
- It has a three-level trigger system
  - Level-1 trigger is hardware based and located in the detector. It reduces 40(20) MHz input rate to 75 kHz
  - Level 2 and Level 3 are software based triggers running on  ${\sim}16k$  core pc farm, reducing final event rate to 300(600) Hz at  ${\sim}1.6~MB/ev$
  - Trigger will be upgraded to 1 kHz output in 2015
- Selected events are stored and processed offline in more detail.
- Both offline processing and online selection is done with the same software using a different configuration
- So far ATLAS stored and processed ~22 PB of raw data.
- With the increase in LHC energy, collision rate, event complexity and trigger output ATLAS software needs to speedup considerably

#### ATLAS Software (ATHENA)



- Comprised of more than 6 million lines of C++ and Python code with a small amount of FORTRAN code
- Spread over ~2000 packages
- Producing 4k+ libraries of various sizes
- Evolving for more than 10 years
- Writen by people with various levels of programming knowledge, some experts, some first timers
- Detailed knowledge of packages is frequently lost due to authors changing topics, institutes or leaving the field.
- Configuration is done in Python
- 64-bit application consumes ~4 GB memory
  - big challenge for many profilers

### ATLAS Software (ATHENA)



- Comprised of more than 6 million lines of C++ and Python code with a small amount of FORTRAN code
- Spread over ~2000 packages
- Producing 4k+ libraries of various sizes
- Evolving for more than 10 years
- Writen by people with various levels of programming knowledge, some experts, some first timers
- Detailed knowledge of packages is frequently lost due to authors changing topics, institutes or leaving the field.
- Configuration is done in Python
- 64-bit application consumes ~4 GB memory
  - big challenge for many profilers

## Need tools to point out problematic code!

### Profilers commonly used in ATLAS



ATLAS uses various tools to profile and monitor ATHENA

- PerfMon to collect coarse level resource utilization information from ATHENA instrumentation.
- Valgrind suite to check leaks, extract callgraphs and detailed CPU utilization
- GOODA to investigate most detailed CPU utilization
- Pin Tools to do detailed code instrumentation to study parameter ranges
- Other tools such as Intel Vtune, PAPI, igprof from CMS, Google perf tools etc.

#### GOODA

### Google Data Analyzer (GOODA)



- Open source, developed by a collaboration between ATLAS and Google
- Uses Linux perf tool to configure and collect detailed performance monitoring unit (PMU) information from hardware monitoring units inside CPUs
- Analyzes the monitoring data and creates spreadsheets that can be displayed in web browsers.
- Gives detailed information about performance bottlenecks

Profiling

#### GOODA

Generic Optimization Data Analyzer GU

#### **GOODA Example**

					~	
				Ę	R	A T L A S
		Enter	search	term		
_starvati	on nówidtr	saturated br	anch_a'	spredictio	n .ore_res	
499448	(15)	946674 ®	(3%)	e <sup>456393</sup>	(1%)	
483609	(1%)	933110	(3%)	443441	(15	
11355	(3%)	10895	(3%)	10917	(3%	

Reports	Reci	oHIPU1000EvtPinSingle H	otspots																		
DQHistMerge1		Cycles Samples																	er search	term	
G4ExampleN03	$\mathbf{\Lambda}$									11.58	145					watio	n	- **	.0		100
Sample		incess path			e path		nalted_co	le che.	retire	distation of the	ction_reti	etired: any	d laten	-9	etruction	starte	dwidth.	saturat	ranch_B	spredice	wore res
MP4HLT		p		80.0		26285382	(100%)	167229	0 (63%)	26014389	33711057	9195152	(345)	8005546	(30%)	499448	(15)	946674	(3%)	456393	(15)
MIG		theme and				35033806	(104)	148474		35736701	22114449	0	12485	7742382	(2004)	492600	1181	022110	(20)		(18)
AthenaReco		triena, py				23922090	(1008)	100074	0 (074)	20720701		0001777	(348)	2000200	(274)	103003	(18)	10000	(34)	10017	
RecoCerebro	ा	incrinox			_	31373/	(100%)	10322	2 (373%)	233400	536423	239000	(7 3%)	200911	(01.0)	11333	(36)	10095	(3%)	10917	(30 ~
RecoOLIoan	12	Outlan Complex			_																
CallGraphCerebr	-	cycles samples																		cerm	
RecoCG		atten na	9.0									core_cycles	- reti	red:stall_	iction_re	tired retired:	ers .	atency		wiction	Lstarvati
RecoOLIoanPin		function		offse	lens	L0	RODULE		proc	ar ar ar a	unharco	46733070	ops	11501	uop		1080_		10	(Carrow)	P2
RecoOLIoanPinP										262853	02 (100%)	) 16/229/8	(63%)	26814389	33/1105	0 919515	2 (3	(4%) 81	985546	(30%)	499448 @
RecoEventNoPin		Trk::RungeKuttaPropaga1	tor 6	0x250e0	0x1051	libTrkExRungel	uttaProp	agator	athena.py	13842	30 (100%)	689204	(49%)	1919291	263298	17 921	1 (	(0%)	228579	(16%)	2933
RecoEventPin		operator new(unsigned ]	Lon C	0×134b0	0x3da	libtcmalloc_m	nimal.so		athena.py	6525	71 (100%	279268	(42%)	889013	113380	5 24096	2 (1	36%)	234114	(35%)	3260
RecoEventPPS		*********	or	ts -	0x2da	libtcmalloc_m	nimal.so		athena.py	4798	02 (100%)	203097	(42%)	658602	87368	12 14787	2 (3	30%)	112605	(23%)	3697
RecoNoMux		InDec: : Si SpacePoints ee	rdN. C	x857d0	0xd74	libSiSpacePoir	tsSeedTo	io1_xk	athena.py	6220	08 (100%	293622	(47%)	887119	96574	5 22731	0 (3	36%)	503	(0%)	2168
RecoNoMuxPin		InDet::TRT_TrajectoryE1	Len., c	0x6c180	0x8d9	libTRT_TrackE	tensionT	001_xk	athena.py	5512	77 (100%	421609	(76%)	329595	38250	4 54278	8 (9	98%)	1641	(0%)	139833
RecoNoMuxPPS	۰	solinterp_	6	0×15680	0xc40	libBFieldCore	50		athena.py	2875	40 (100%)	147594	(51%)	379448	45597	6 18274	5 (6	53%)	99303	(34%)	5885
RecoNoMuxEver	.,⊞	_dynamic_cast@@CXXABI_	1.3 6	9xbf200	0x11f	libstdc++.so.	.8.18		athena.py	3070	11 (100%)	189831	(61%)	235244	35213	3 21042	1 (6	58%)	112561	(36%)	2823
RecoNoMuxEver		rk::MagneticFieldMapSc	ole	0×5940	0×1c5	libTrkMagFiel	Utils.so		athena.py	4497	64 (100%	287254	(63%)	315379	42431	6 12763	5 G	20%)	L05407	(23%)	3475
RecoNoMuxEver	۰.	aster.0.gbmagz_		0×fb80	0x4a0b	libBFieldStan	1.50		athena.py	3602	18 (100%	166033	(46%)	537975	62175	5 3859	2 (1	L0%)	30454	(8%)	410
ParalleiG4		rk::RungeKuttaPropagat	tor e	9x277f0	0xff5	libTrkExRunge	uttaProp	agator	athena.py	1035	48 (100%	65400	(63%)	74474	10073	2 3679	8 (3	35%)	L03723	(100%)	963
Reconomusever	Ħ	nDet::TRT_Trajectory_>	dk: 6	9×64200	0x1ce	libTRT_TrackE:	tensionT	001_xk	athena.py	4963	42 (100%	309302	(62%)	398839	48283	13 13	1 (	(0%)	919	(0%)	197
Aller C 4		rk::PatternTrackParame	rte	0x5220	0xb93	libTrkPattern	arameter	5.50	athena.py	2711	53 (100%)	111591	(41%)	475779	65446	1 17	5	(0%)	107792	(39%)	1181
AllasC 40 in		rk::RungeKuttaUtils::1	tra. 0	0×179†0	0x923	libTrkExUtils	50		athena.py	2494	29 (100%	134726	(54%)	358266	45383	1 3787	0 (1	15%)	113371	(45%)	410
Resolution		rk::RungeKuttaPropagat	tor 6	9x26140	0x573	libTrkExRunge	uttaProp	agator	athena.py	1670	81 (100%)	106231	(63%)	126570	17325	8 455	1	(2%)	99610	(59%)	1498
BecoHiPU1000F		olefittorsiml_		0xace0	0x10c8	libBFieldCore	50		athena.py	2249	26 (100%	94451	(41%)	251920	38299	2 1004	2 (	(4%)	60273	(26%)	
AtlasG4Andreas		deflate_slow		0×6850	0x976	libz.so.1.2.3			athena.py	2673	47 (100%)	116870	(43%)	405506	44976	0 2699	7 (1	1.0%)	328	(0%)	82
HITDummy		TTrainedNetwork::calcul	lat. 0	0×1e320	0x43e	libTrkNeuralNe	tworkUti	1sL1b	athena.py	1745	63 (100%)	42079	(24%)	367583	42090	9 16907	2 (5	96%)	3588	(2%)	20193
		_cxxabiv1::_vmi_class	_t. 6	0xc1670	0x5ce	libstdc++.so.	.8.18		athena.py	1175	06 (100%	56777	(48%)	153151	21737	5 1406	7 (1	1%)	48999	(34%)	1291
Help	3																				

#### **GOODA Example**

					Generi	c Optim	ization Da	ta Analyz	er 💷										
Reports	RecoHIPU1000EvtPinSingle Hotspot	s																	
DQHistMerge1	🗄 Cycles Samples															Enter	search tr	erm	
G4ExampleN03						-vele		-tall_cy	185	red				taryatic	sn.	mated			100
Sample	process path	=pdul	e path		naited_c	ore	upps_retire	d.stra	ction_ret.	retired.on	d_laten	c7 50	struction.	_5(#) 	nowidth	atur-	ranch_#15	prest	store_res
MP4HLT				26285302	(100%)	1672297	0 (63%)	26014389	33711057	9195152	(34%)	8005546	(30%)	499448	(15)	946674	(3%)	456393	(15)
MIG	E athena ov			25922896	(1005)	1484747	3 (57%)	25726701	33116688	8881359	(345)	7763283	(295)	483609	(15)	933110	(35)	443441	(15
AthenaReco	Florel trace			215927	(1005)	102225	2 (576%)	252499	\$26422	220050	(75%)	256911	(01%)	11355	(26)	10005	(26)	10917	(25
RecoCerebro	N III				(2004)														>
RecoOLIoan	Cortes Samples	$\sim$	_		_											Enter			
CallGraphCerebr	Droo	č								145			vc1es	ad.					-17
RecoCG	FIUC	625	ses	2	4116				insted.	core_cycles	ret!	red:stall_	uction_ret	retired.	ans -4.15	atency		truction	starvat
RecoOLJoanPin			1500		\$000.		provi	161953	07 (100%)	16733970	(6 3W)	110-51	22711057	2 010515	1000-	N 000	110	(200)	400440
RecoOLIoanPinP								101033	01 (100%)		(03.6)	10014505	3371103	6	. (24	a) 6		(2004)	A
RecoEventNoPin	El Trk::RungeKuttaPropagator	8x250e0	0x1051	LibTrkExKungel	uttaProp	bagator	athena.py	13842	30 (100%)	689204	(49%)	1919291	263298	7 921	.1 (8	ns) 223	8579	(16%)	2932
RecoEventPin	El operator new(unsigned lon	0×13460	0x3da	libtcmalloc_m	nimal.s	>	athena.py	6525	71 (100%)	279268	(42%)	889013	113380	5 24090	32 (36	76) 23	.4114	(35%)	3260
RecoEventPPS	H operator delete(void*)	0×12c10	0x2da	libtcmalloc_m	nimal.s	>	athena.py	4798	02 (100%)	203097	(42%)	658602	87368	2 14787	/2 (30	n6) 11	.2605	(23%)	3697
RecoNoMux	■ InDet::SiSpacePointsSeedR	0×8b7d0	0xd74	libSiSpacePoir	tsSeedTo	201_xk	athena.py	6220	08 (100%)	293622	(47%)	887119	96574	5 22731	.0 (36	(16)	503	(0%)	2160
RecoNoMuxPin	■ InDet::TRT_Trajectoryflem	0×6c180	0x8d9	libTRT_TrackE	tension	fool_xk	athena.py	5512	77 (100%)	421609	(76%)	329595	382584	4 54278	18 (98	1%)	1641	(0%)	139833
RecoNoMuxPPS	■ bsolinterp_	0×15680	0xc40	libBFieldCore	50		athena.py	2875	40 (100%)	147594	(51%)	379448	45597	6 18274	45 (63	J%) 91	9303	(34%)	5885
RecoNoMuxEver	□dynamic_cast@@CXXABI_1.3	8xbf288	0x11f	libstdc++.so.	.8.18		athena.py	3070	11 (100%)	189831	(61%)	235244	35213	3 21042	21 (68	7%) 11	.2561	(36%)	2823
ReconomuxEver	■ Trk::MagneticFieldMapSole	0×5940	0x1c5	libTrkMagFiel	Utils.s	,	athena.py	4497	64 (100%)	287254	(63%)	315379	42435	6 12763	35 (28	/%) 10	15407	(23%)	3475
Reconomuxever	🗄 master.0.gbmagz_	0×fb80	0x4a0b	libBFieldStan	1.50		athena.py	3602	18 (100%)	166033	(46%)	537975	62175	5 3859	92 (10	/%) 3	10454	(0%)	410
ParalleiG4	Trk::RungeKuttaPropagator	0×277†0	0xff5	libTrkExRunge	uttaProp	agator	athena.py	1035	48 (100%)	65400	(63%)	74474	10072	2 3679	98 (35	3%) 10	13723 (	(100%)	963
Reconcentuxever	Ⅲ InDet::TRT_Trajectory_xk:	0×64200	0x1ce	libTRT_TrackE:	tension	foo1_xk	athena.py	4963	42 (100%)	309302	(62%)	398839	48283	3 13	31 (0	1%)	919	(0%)	197
Aller C4	I Trk::PatternTrackParamete	0×5220	0xb93	libTrkPattern	arameter	rs.so	athena.py	2711	53 (100%)	111591	(41%)	475779	65446	1 13	75 (8	<b>7%)</b> 10	17792	(39%)	1181
Allas G40 in	Trk::RungeKuttaUtils::tra	0×179†0	0x923	libTrkExUtils	50		athena.py	2494	29 (100%)	134726	(54%)	358266	45383	1 3787	70 (15	5%) 11	3371	(45%)	410
Receiver 11	H Trk::RungeKuttaPropagator	0×26140	0x573	libTrkExRunge	uttaProp	agator	athena.py	1670	81 (100%)	106231	(63%)	126570	173291	8 455	51 (2	2%) 9*	9610	(59%)	1496
RecohiPU1000F	H solefittorsiml_	0xace0	0x10c8	libBFieldCore	50		athena.py	2249	26 (100%)	94451	(41%)	251920	38299	2 1004	42 (4	(%) 64	0273	(26%)	
AtlasG4Andreas	⊞ deflate_slow	0×6850	0x976	libz.so.1.2.3			athena.py	2673	47 (100%)	116870	(43%)	405506	449761	0 2699	37 (18	9%)	328	(0%)	82
HITDummy	H TTrainedNetwork::calculat.	0×1e320	0x43e	libTrkNeuralNe	tworkUt	1sL1b	athena.py	1745	63 (100%)	42079	(24%)	367583	42090	9 16907	72 (96	5%)	3588	(2%)	20193
v v	■ _cxxabfv1::_vmf_class_t.	0xc1670	0x5ce	libstdc++.so.	.0.10		athena.py	1175	06 (100%)	56777	(48%)	153151	21737	5 1406	57 (11	4	10999	(34%)	1291
Help	۲. II			••••••						· · · · · · ·									>



#### **GOODA Example**



			Generic Optin	nization Da	ata Analyze	er GUI									
Reports	RecoHIPU1000EvtPinSingle Hotspots														
DQHistMerge1	E Cycles Samples														
G4ExampleN03			in sych	6	a stall_cycl	es retir	ed				starvatio	n	urated	- redic	61.011
Sample	process path	sodule path	unhalted_color	uops_retire	instruc	uops_r	etires. load	Listenc	.) 10	struction.	680	dwidth_50	branch.	p1501-	store_res
MP4HLT			26285302 (100%) 167229	70 (63%)	26014389	3711057	9195152	(34%)	8005546	(35%)	499448	(15) 94	6674 (3%	45639	3 (15)
AlbertRese	🗄 athena.py		25922896 (100%) 148474	73 (57%)	25726701	3116688	8881359	(34%)	7703283	(29%)	483609	(1%) 93	3110 (39	) 44344;	L (15
RecoCerebro	H wmlfnux		315937 (100%) 18322	52 (579%)	253488	536423	239868	(75%)	256911	(81%)	11355	(3%)	.0895 (3%	) 1091	r (3% -
BecoOiloan	(< II														>
CallGraphCerebr	Cycles Samples												Enter searc	h term	
RecoCG							ore_cycles		ad stall_c	ycles ret	ared	ri i			starvati
RecoOLloanPin	function name	offset length	scoule	proc	155	unhalted	u01	ps_retu	instr	uction_	retires	load_lat		nstructio	102
RecoOLIoanPinP					2628530	2 (100%)	16722978	(63%)	26014389	33711057	0,919515	2 (34%)	0,8005546	(30%)	499448 @
RecoEventNoPin	H Trk::RungeKuttaPropagator	03250e0 0x1051 l	ibTrkExRungeKuttaPropagator	athena.py	138423	0 (100%)	689204	(49%)	1919291	2632987	921	1 (0%	228579	(16%)	2932
RecoEventPin	■ operator new(unsigned lon	0×154b0 0×3da l'	ibtcmalloc_minimal.so	athena.py	65257	1 (100%)	279268	(42%)	889013	1133801	5 24096	2 (36%	234114	(35%)	3260
RecoEventPPS	■ operator delete(void*)	0x12c10 0x2da l	ibtcmalloc_minimal.so	athena.py	47980	2 (100%)	203097	(42%)	658602	873681	2 14787	2 (30%	112605	(23%)	3697
RecoNoMux	Ⅲ InDet::SiSpacePointsSeedM	0×807d0 0xd74 1	ibSiSpacePointsSeedTool_xk	athena.py	62200	8 (100%)	293622	(47%)	887119	965745	5 22731	0 (36%	503	(0%)	2166
RecoNoMuxPin	InDet::TRT_TrajectoryElem.	0×6:180 0×8d9 l	ibTRT_TrackExtensionTool_xk	athena py	55127	7 (100%)	421609	(76%)	329595	382584	54278	8 (98%	1641	(0%)	139833
RecoNoMuxPPS	⊞ bsolinterp_	0×15680 0×c40 1	1bBFieldCore.so	athena.py	28754	0 (100%)	147594	(51%)	379448	455976	5 18274	5 (63%	99303	(34%)	5885
RecoNoMuxEver		8xb/200 0x11f l	ibstdc++.so.6.8.18	athena.py	30701	1 (100%)	189831	(61%)	235244	352133	3 21042	1 (68%	112561	(36%)	2823
RecoNoMuxEver	H Trk::HagneticFieldHapSole	0x5940 0x1c5 1	ibTrkMagFieldUtils.so	athena.py	44976	4 (100%)	287254	(63%)	315379	424356	5 12763	5 (28%	105407	(23%)	3475
RecoNoMuxEver	H master.0.gbmagz_	0x1080 01450 1	thereidStand so f.	athens by	36021	8 (100%)	166033	(46%)	537975	621751	5 3859	2 (10%	30454	(8%)	410
Paralle/G4	I Trk::RungeKuttaPropagator	0x2770	ispot iur	CIIO	15.34	8 (100%)	65400	(63%)	74474	100723	2 3679	8 (35%)	103723	(100%)	963
RecoNoMuxEver	H InDet::TRT_Trajectory_xk:	0x6 200 0x1ce 1	ibTRT_TrackExtensionTool_xk.	athena.py	49634	2 (100%)	309302	(62%)	398839	482833	3 13	1 (0%	919	(0%)	193
Aller C 4	H Trk::PatternTrackParamete	0x5220 0xb93 1	ibTrkPatternParameters.so	athena.py	27115	3 (100%)	111591	(41%)	475779	654461	1 17	5 (0%)	107792	(39%)	1181
Attarcation	⊞ Trk::RungeKuttaUtfls::tra	0×17910 0×923 1	ibTrkExUtils.so	athena.py	24942	9 (100%)	134726	(54%)	358266	453833	L 3787	0 (15%	113371	(45%)	410
BecoHiPU100Ev	■ Trk::RungeKuttaPropagator	0x25140 0x573 l	ibTrkExRungeKuttaPropagator	athena.py	16708	1 (100%)	106231	(63%)	126570	173298	455	1 (2%	99610	(59%)	1496
RecoHiPU1000E	⊞ solefittorsiml_	0xpce0 0x10c8 l	ib8FieldCore.so	athena.py	22492	6 (100%)	94451	(41%)	251920	382992	2 1004	2 (4%	60273	(26%)	
AtlasG4Andreas	H deflate_slow	0x5850 0x976 1	ibz.so.1.2.3	athena py	26734	7 (100%)	116870	(43%)	405506	449760	2699	7 (10%	328	(0%)	88
HLTDummy	H TTrainedNetwork::calculat.	0x1e320 0x43e l	ibTrkNeuralNetworkUtilsLib	athena.py	17456	3 (100%)	42079	(24%)	367583	428985	9 16907	2 (96%	3588	(2%)	20193
	□	0×1670 0×5ce l	ibstdc++.so.6.8.18	athena.py	11750	6 (100%)	56777	(48%)	153151	217375	5 1406	7 (11%	40999	(34%)	1291
Help	×	/													>

#### **GOODA** Example

Reports

MIG

Help





#### Hottest functions



- Tracking code, particularly Runge-Kutta methods
  - suffering from instruction starvation
  - mostly composed of vector and matrix operations
  - Vectorization helps, up to 2.5x speedup from manual vectorization in certain points is achieved, need vectorized vector math libs
- Memory allocation and de-allocation
  - too many new() and deletes
  - Event Data Model (EDM) change is underway
- Magnetic field code
  - suffering from load latency and instruction latency
  - was written in FORTRAN code, several calls deep
  - re-written in C++, already was about 2x faster than fortran implementation
  - C++ code profiled again to optimize further

### Improving new Magnetic Field



#### Profiled a special test code that queries the magnetic field code randomly

Description         Description <thdescription< th=""> <thdescription< th=""></thdescription<></thdescription<>	🗄 Cycles	Samples																								
A Constant Co		function uses		halled_ci	,re_9/c145	ops_retir	ed stall_cy	ites Iction_retires	etred: any	94_1ate	pet 1	astract	ion_stat	vation andwin	th_satura	ead march.J	also realics	ian core_t	esources_satu	ITATES	~latency	x c e p e i i	n handline mhall	uops_	e_crcles Lecaded:sta uops_	L_cycles
Drag-field Ung-fields. Ung-fields. (1981) 353218 (2081) 353218 (2081) 353218 (2081) 3542 (	٩		116795606	(100%)	86739168	(74%)	84707079	101187857	1750191	(1%)	207425	(0%)	33679	(0%)	368439	(0%)	287951	(0%)	63182335	(54%)	162073	(0%)	106231587	91366677	87217441	86995635
Bing/state         Attus/state         532244         (108)         1440276         (108)         93226         (108)         52226         (108)         5202         5202         5202         520	⊟ MagField	::IMagFieldSvc:	103506891	(100%)	75992130	(73%)	71409380	83625478	95521	(0%)	95489	(0%)	387	(0%)	472767	(0%)	53906	(0%)	56925933	(54%)	48906	(0%)	93845367	79918686	75534086	76435337
BinggFishid InggFishidTexts. 2103730 (1985) 3118945 (033) 3665167 4165965 697 (96) 642 (96) 7742 (96) 127252 (198) 2997 (96) 1946533 1872139 1700973 111	⊞ Magfield	::AtlasFieldSvc.	5217442	(100%)	3454370	(66%)	4639374	5882879	693876	(13%)	35228	(0%)	290	(05)	50938	(0%)	156525	(3%)	2398129	(45%)	3323	(05)	4757519	3537121	3701766	3550893
	⊞ MagField	::MagFieldTestb.	2151793	(100%)	1118041	(51%)	3665167	4163691	6897	(8%)	6452	(0%)			7742	(8%)			1078252	(50%)	2897	(6%)	1946513	1873359	1708075	1112767

### Improving new Magnetic Field



#### Profiled a special test code that queries the magnetic field code randomly

Cycles S	Samples																					
	touction uses	æ	halled_ci	re_cycles	ops_retir	6.stall_cy	tes petion_retires upps_ce	d stired: and	187 aTal	ned inserve	tion_starvatio	th_satur	red ranch_miss	rediction store_	esources_sat	urated	LIATENCY ext	ception.)	and line, Vedau	ced_reteren	ce_cycles	11_CY 15948
		116795606	(100%)	86739168	(74%)	84707079	101187857	1750191	(1%)	707425 (0%)	33679 (0%)	368439	(0%) 28	7951 (0%)	63182335	(54%)	162073	(0%) 10	6231587	9136667	8721744	869
MagField:	:IMagFieldSvc:	103506891	(100%)	75992130	(73%)	71409380	83625478	95521	(0%)	95489 (0%)	387 (0%)	472767	(04) 5	3906 (0%)	56925933	(54%)	48906	(0%) 9	3845367	7991868	5 7553408	7643
Magfield:	AtlasFieldSvc.	5217442	(100%)	3454370	(66%)	4639374	5882879	693876	(13%)	35228 (0%)	200 (0%)	50938	(0%) 15	6525 (3%)	2398129	(45%)	3323	(0%)	4757515	353712	370176	355
MagField:	HagFieldTestb.	2151793	(100%)	1118041	(51%)	3665167	4163691	6897	(9%)	0452 (0%)	_	7742	(8%)		1878252	(50%)	2897	(6%)	1946513	187535	1708075	111
							_	_			Expan	ding	j insi	truction	on late	ency					$\sim$	
								-t101		- 53	acu.		11	3		4.6	V Du-					
							ispredi	ction	res	ources_si	turetruc	tion_	laten	cy	th: cyc	les_di	v_bus	, div		reptio	n_hand	
							28795	store	_res	6318233	instruc 5 (54	tion_	1aten (	ar ₩ar 422	(54%)	1es_di 3931	arith	1620	e×'	ceptio (0%)	n_h <sup>and</sup> 10623	
							ispredi 28795:	store	_res %) {	ource5_51	instruc 5 (54	(100-	1 <sup>aten</sup>	• <b>₽</b> ₽ <sup>¶</sup> 422	(54%)	3931	arith	1620	e×'	ceptio (0%)	n_hand 10623	
							15predi 28795: 5390	ction store 1 (0 6 (0	5 res %)	ource <sup>s_s;</sup> 6318233: 5692593:	in <sup>struc</sup> 5 (54 3 (54	(tion) (%) ( (%) 5	1a <sup>ten</sup> 3182	ar ∳ar 422 806	(54%) (54%)	3931 3424	arit 173	1620 489	ex 973 (	(0%)	n_h <sup>ans</sup> 10623 9384	
							28795 28795 5390 15652	ction store 1 (0 6 (0 5 (3	_res %) ( %) : %) :	ourc <sup>e5_51</sup> 6318233 5692593 239812	in <sup>struc</sup> 5 (54 3 (54 9 (45	tion- (%) 6 (%) 5 (%)	1aten 3182- 6926 2398	₩ 422 806 148	(54%) (54%) (54%) (45%)	3931 3424 142	arit 173 795	1620 489 33	ex 973 0	(0%)	میر م 10623 9384 475	
							i≤predi 28795: 5390 15652:	ction store 1 (0 6 (0 5 (3	5_res %) ( %) : %)	ource5_51 63182333 5692593 239812 107825	instruc 5 (54 3 (54 9 (45 2 (50	(510 (%) (%) (%) (%) (%)	1a <sup>ten</sup> 3182- 6926 2398 1078	₩×* 422 906 148 256	(54%) (54%) (45%) (50%)	142 96	173 795 693	1620 489 33 20	ex 973 906 923	(0%) (0%) (0%) (0%)	n مع <sup>مر م</sup> 10623 9384 475 194	

### Improving new Magnetic Field



#### Profiled a special test code that queries the magnetic field code randomly

: cycles samples																					
function pase	a	halled_c	re_cycles	OFS_FEELT	instri	ites Iction_retired UDFS_re	terred: any	N_latency	Instruct	ion_starvatio	ath_satur	ared aranch_mi	spredictio	in pre_resources-	instruction	on_latency	xcepcic	ادی درس	tred_reters	nce_crcles s_tecaded.st uops	11_cy _15.946
	116795606	(100%)	86739168	(74%)	84707079	101187857	1750191	(1%) 70	97425 (0%)	33679 (0%)	368439	(0%)	287951	(0%) 631823	15 (54%)	162073	(0%)	10623158	7 913666	7 8721744	869
MagField::IMagFieldSvc:	103506891	(100%)	75992130	(73%)	71409380	83625478	95521	(0%) 9	15489 (0%)	387 (0%)	472767		53906	(0%) 569259	3 (54%)	48906	(0%)	9584536	7 7991861	6 7553408	5 7643
MagField::AtlasFieldSvc.	5217442	(100%)	3454370	(66%)	4639374	5882879	693876	(135) 3	5228 (0%)	200 (0%)	50938	(8%)	156525	(3%) 23981	9 (45%)	3323	(05)	475751	35371	1 370176	5 35
HagField: HagFieldTestb.	2151793	(100%)	1118041	(51%)	3665167	4163691	6897	(9%)	0452 (0%)	_	7742	(8%)		18782	2 (50%)	2897	(6%)	194651	3 18753	9 170807	11
						-	_			Expan	ding	g ins	struc	tion la	tency	/				$\sim$	
																				_	
						ispredi	store	resou	11C62-20	instruc	tion.	late		arith: 9	cles_d	arit	h: d	(V ()	ceptio	n_hand	
						28795:	store. 1 (09	_resou	182335	instruc (54	tion.	1 <sup>ate</sup>	2422	arith: 9 (54%	cles_d	par <sup>11</sup> 1173	16	e1 2073	(0%)	10623	
						28795: 53906	store. 1 (09 6 (09	_r <sup>esov</sup> 6) §3 6) 56	182335 925933	instruc i (54	(100) (%) (%)	1 <sup>ate</sup> 5318	2422	arith: 9 (54% (54%	cles_d ) 393 ) 342	4795	16: 4:	2073 8906	(0%)	10623 9384	
						زيه و و مار	store. 1 (09 5 (09	_resov 6) 63 6) 56 6) 2	182335 5925933 2398129	in <sup>struc</sup> 5 (54 8 (54 9 (45	(%)	1ate 5318: 5692 239	2422 6006 8148	ar <sup>ith: Q</sup> (54% (54% (45%	<ol> <li>cles_d</li> <li>393</li> <li>342</li> <li>14</li> </ol>	4795 2693	16: 4:	2073 8906 3323	(0%) (0%)	10623 9384 475	

High instruction latency originating from division operations. Can drill down (double-click) for details

GOODA

#### Magnetic field code details

ss pri	nc_18 *principal	eile 1910	it.18 pintrial	es.54	disasseably	الاين		ine waber	under 1	Ited_C	are_Q'Cles	ps_retif	ed:stall_cr	iction_reti	erites:	load_lat	encl .	nstrac	eren starvat	ijan dth_satorat
9						103506891	c		103506891	(100%)	75992130	(73%)	71409380	83625478	85521	(0%)	85489	(0%)	387 (0%)	472767
91	IMagF1eldSvc.h	95	BFieldCache.h	mulss	Sxee9, Sxee1	1		32 // no. return defa.												
91	IMagFieldSvc.h	95	SFieldCache b	a dds s	Sxen1, Sxnn2	17840	8	<pre>B3 bxyz[0] = bxyz[1] .</pre>				$\sim$								
91	IlensieldSvc.h	95	BFieldCache	nutss	Sxm11,Sxm1	899274	8	84 return;				- T								
91	UISS	a	semb	V	windov	N 13 62106	8	35 }	S S	SO	irce	a v	vin	dov	V					
91	INagFieldSvc.h	95	BFieldCache.h	add	\$8x2,Srbx	92392	8	36 // refresh cache				•••	• • • • •		•					
91	IMagF1eldSvc.h	83	BF1eldCache.h	cmp	\$8x6,%rbx	32	8	<pre>####################################</pre>	131330	(100%)	59674	(45%)	198128	256662	2678	(2%)	161	(0%)	65 (0%)	32
91	IMagFieldSvc.h	83	8FieldCache.h	jne	36ee1	24421	8	38 }												
91				🕀 Basin	: Block 22 <8x372d.	35964257	8	39												
91	IMagFieldSvc.h	99	BFieldCache.h	movss	0x104(%rsp),%xem.	100586	9	0 // do interpolation												
91	IMagFieldSvc.h	99	BFieldCache.h	null			9	<pre># #_cache.getB( xyz, r, .</pre>	82972948	(100%)	59682740	(715)	61724896	70339606	73327	(8%)	54164	(05)	226 (0%)	15291
91	IMagF1eldSvc.h	102	BFieldCache.h	moves	0x104(%rsp),%xem.	30260	9	92												
91	IMagFieldSvc.h	102	8fieldCache.h	null			9	33 // add biot savart com.												
91	IMagFieldSvc.h	99	BFieldCache.h	mulss	Sxee15,Sxee3	97	9	04 if ( m_cond ) {	97	(100%)	125	(128%)		27	97	(100%)				
91	IMagF1eldSvc.h	100	BF1eldCache.h	novap	Sxee15,Sxee12	9936	9	for ( std::vector <bf.< td=""><td>127620</td><td>(100%)</td><td>110944</td><td>(86%)</td><td>4279</td><td>54157</td><td>516</td><td>(0%)</td><td>290</td><td>(0%)</td><td></td><td></td></bf.<>	127620	(100%)	110944	(86%)	4279	54157	516	(0%)	290	(0%)		
							0	is subdivision for and												

 Detailed view contains both dissassembly and source code if debug symbols and source files are available

- Events are displayed at instruction level and hottest basic block is automatically highlighted
- Debug symbols in optimized builds are skewed, instruction latency is coming from another file.



Profiling

GOODA

#### Magnetic field code details



- Detailed view contains both dissassembly and source code if debug symbols and source files are available
- Events are displayed at instruction level and hottest basic block is automatically highlighted
- Debug symbols in optimized builds are skewed, instruction latency is coming from another file.

GOODA

#### Magnetic field code details

ss priv	cle principal rile	- jai	eial eile	ass amb by	تەرابى	line ounker	50 <sup>13 FCR</sup>		قرارين ا	Ited_core_cycl	us usp <sup>5</sup> - <sup>rati</sup>	red stall, c	uction_ret	ratired:any	Lateocy in	struction_starvat	ion ath_satorated bran
9				18	33506891 (			10	3506891 (	100%) 75992	30 (73%	7140938	8362547	8 85521 (6	%) <u>§</u> 5489	(0%) 387 (0%)	472767 (
					🖻	82 //	so return defa						_				
9	1 IMagFieldSvc.h	83	BFieldCache.h	cmp	\$0x6,%rbx		32		87	mi	IF 1 e L d C	acne( z	, r.,	131330	(100%)	59674	(45%)
9	1 IMagFieldSvc.h	83	BFieldCache.h	jne	36ee1		24421	=	88	}							
9	1			🗄 Basic	Block 22	< 0 x 3 7 2 d	35964257		89								
9	1 IMagFieldSvc.h	99	BFieldCache.h	movss	0x104(%rs	p),%xnn	100586		90	// do	interp	olation					
9	1 IMagFieldSvc.h	99	BFieldCache.h	null					91	m_cac	e.getB	(xyz,	r.,	82972948	(100%)	59682740	(71%)
9	1 IMagFieldSvc.h	102	BFieldCache.h	movss	0x104(%rs	p),%×nn	30260		92								
9	1 IMagFieldSvc.h	102	BFieldCache.h	null					93	// ad	d biot	savart	com				
9	1 IMagFieldSvc.h	99	BFieldCache.h	mulss	%×nn15,%×	:nn3	97		94	if ()	_cond	) (		97	(100%)	125	(128%)
9	1 IMagFieldSvc.h	100	BFieldCache.h	movaps	%×nn15,%×	nn12	9936		95	for	( std:	vector	<bf< td=""><td>127620</td><td>(100%)</td><td>110944</td><td>(86%)</td></bf<>	127620	(100%)	110944	(86%)
									96	1	->addB	iotSava	rt(				

- Detailed view contains both dissassembly and source code if debug symbols and source files are available
- Events are displayed at instruction level and hottest basic block is automatically highlighted
- Debug symbols in optimized builds are skewed, instruction latency is coming from another file.

#### Fixing the problem

82		float dBdz[3], dBdr[3], dBdphi[3];
83		for ( int j = 0; j < 3; j++ ) { // Bz, Br, Bphi components
84		dBdz[j] = sz*( gr*( gphi*(m_field[4][j]-m_field[0][j]) +
85		fphi*(m_field[5][j]-m_field[1][j])
86		fr*( gphi*(m_field[6][j]-m_field[2][j]) +
87		fphi*(m_field[7][j]-m_field[3][j])
88		dBdr[j] = sr*( gz*( gphi*(m_field[2][j]-m_field[0][j]) +
89		fphi*(m_field[3][j]-m_field[1][j])
90		fz*( gphi*(m_field[6][j]-m_field[4][j]) +
91		fphi*(m_field[7][j]-m_field[5][j])
92		dBdphi[j] = sphi*( gz*( gr*(m_field[1][j]-m_field[0][j]) +
93		fr*(m_field[3][j]-m_field[2][j]) ) +
94		fz*( gr*(m_field[5][j]-m_field[4][j]) +
95		fr*(m_field[7][j]-m_field[6][j]) ) );
96		}
97		// convert to cartesian coordinates
98		float cc = c*c;
99		float cs = c*s;
100		float <mark>ss = s*s</mark> ;
101		<pre>deriv[0] = cc*dBdr[1] - cs*dBdr[2] - cs*dBdphi[1]/r + ss*dBdphi[2]/r + s*B[1]/r;</pre>
102		deriv[1] = cs*dBdr[1] - ss*dBdr[2] + cc*dBdphi[1]/r - cs*dBdphi[2]/r - c*B[1]/r;
103		deriv[2] = c*dBdz[1] - s*dBdz[2];
104		<pre>deriv[3] = cs*dBdr[1] + cc*dBdr[2] - ss*dBdphi[1]/r - cs*dBdphi[2]/r - s*B[0]/r;</pre>
105		deriv[4] = ss*dBdr[1] + cs*dBdr[2] + cs*dBdphi[1]/r + cc*dBdphi[2]/r + c*B[0]/r;
106		deriv[5] = s*dBdz[1] + c*dBdz[2];
107		deriv[6] = c*dBdr[0] - s*dBdphi[0]/r;
108		deriv[7] = s*dBdr[1] + c*dBdphi[0]/r;
109		deriv[8] = dBdz[0];
110	}	
111 }		



### Fixing the problem

82	<pre>float dBdz[3], dBdr[3], dBdphi[3];</pre>
83	for ( int j = 0; j < 3; j++ ) { // Bz, Br, Bphi components
84	dBdz[j] = sz*( gr*( gphi*(m_field[4][j]-m_field[0][j]) +
85	fphi*(m_field[5][j]-m_field[1][j]) ) +
86	fr*( gphi*(m_field[6][j]-m_field[2][j]) +
87	fphi*(m_field[7][j]-m_field[3][j]) ) );
88	dBdr[j] = sr*( gz*( gphi*(m_field[2][j]-m_field[0][j]) +
89	fphi*(m_field[3][j]-m_field[1][j])
90	fz*( gphi*(m_field[6][j]-m_field[4][j]) +
91	fphi*(m_field[7][j]-m_field[5][j]) ) );
92	dBdphi[j] = sphi*( gz*( gr*(m_field[1][j]-m_field[0][j]) +
93	fr*(m_field[3][j]-m_field[2][j]) ) +
94	fz*( gr*(m_field[5][j]-m_field[4][j]) +
95	fr*(m_field[/ dfs-öff#e// 6/eblace with
96	
97	// convert to cartesian coordinates Inverse multiplication
98	float cc = c*c;
99	tloat cs = c*s;
100	float ss = s*s;
101	$deriv[0] = cc^* dBdr[1] - cs^* dBdr[2] - cs^* dBdph1[1]/r + ss^* dBdph1[2]/r + s^* B[1] $
102	deriv[1] = C\$*dBdr[1] - S\$*dBdr[2]/+ CC*dBdph1[1]/F - C\$*dBdph1[2]/F - C*B[1]/F
103	$deriv[2] = C^{*} dB d2[1] - S^{*} dB d2[2];$
104	$deriv[3] = cs^* dBdr[1] + cc^* dBdr[2]$ $ss^* dBdpn1[1]/r - cs^* dBdpn1[2]/r - s^* B[0]/s$
105	$deriv[4] = SS^*dBdr[1] + CS^*dBdr[2] + CS^*dBdpn1[1]/r + CC^*dBdpn1[2]/r + C*Bbp]/r$
100	$derIV[5] = S^*dDdz[1] + C^*dDdz[2];$
100	deriv[0] = C (der [0] - S (der der der le [0]))
100	deriv[7] = S'ubu[1] + C'ubup[1][7],
110	l derivioj - ubuzioj,
111	
***	1



### Fixing the problem

82	<pre>float dBdz[3], dBdr[3], dBdphi[3];</pre>
83	for ( int j = 0; j < 3; j++ ) { // Bz, Br, Bphi components
84	dBdz[j] = sz*( gr*( gphi*(m_field[4][j]-m_field[0][j]) +
85	fphi*(m_field[5][j]-m_field[1][j])
86	fr*( gphi*(m_field[6][j]-m_field[2][j]) +
87	fphi*(m_field[7][j]-m_field[3][j]) ) );
88	dBdr[j] = sr*( gz*( gphi*(m_field[2][j]-m_field[0][j]) +
89	
90	fz*( gphi*(m_field[6][j]-m_field[4][j]) +
91	fphi*(m_field[7][j]-m_field[5][j]) ) );
92	dBdph1[j] = sphi*( gz*( gr*(m_field[1][j]-m_field[0][j]) +
93	fr*(m_field[3][j]-m_field[2][j]) ) +
94	fz*( gr*(m_field[5][j]-m_field[4][j]) +
95	Dot products of two fr*(m_field [] offs off hereblace with
96	vectors
97	inverse multiplication
98	float cc = c*c;
99	float cs = c*s;
100	float ss = s*s;
101	deriv[0] = cc*dBdr[1] - cs*dBdr[2] / cs*dBdphi[1]/r + ss*dBdphi[2]/r + s*B[1]/r;
102	deriv[1] = cs*dBdr[1] - ss*dBdr[2] + cc*dBdph1[1]/r - cs*dBdph1[2]/r - c*B[1]/r
103	$deriv[2] = C^* dBdz[1] - S^* dBdz[2];$
104	$\operatorname{aeriv}[3] = \operatorname{Cs^*aBar}[1] + \operatorname{Cc^*aBar}[2]  \operatorname{ss^*dBdph1}[1]/r - \operatorname{Cs^*dBdph1}[2]/r - \operatorname{Ss^*B}[0]/r;$
105	$deriv[4] = Ss^* dBdr[1] + Cs^* dBdr[2] + Cs^* dBdpn1[1]/r + Cc^* dBdpn1[2]/r + C*BL0]/r;$
100	$der[V[5]] = S^* dBdz[1] + C^* dBdz[2];$
100	deriv[0] = C"dDdr[0] - S"dbdphi[0]/r;
100	deriv[/] = S*dBdr[i] + C*dBdpn1[0]/r;
110	deliv[o] = dbdz[o];
111	3
TTT	3



#### Fixing the problem

82	float dBdz[3], dBdr[3], dBdphi[3];
83	for ( int j = 0; j < 3; j++ ) { // Bz, Br, Bphi components
84	dBdz[j] = sz*( gr*( gphi*(m field[4][j]-m field[0][j]) +
85	fphi*(m field[5][j]-m field[1][j]) ) +
86	fr*( gphi*(m <sup>-</sup> field[6][j]-m <sup>-</sup> field[2][j]) +
87	fphi*(m <sup>-</sup> field[7][j]-m <sup>-</sup> field[3][j]) ) );
88	dBdr[j] = sr*( gz*( gphi*(m_field[2][j]-m_field[0][j]) +
89	<pre>fphi*(m_field[3][j]-m_field[1][j]) ) +</pre>
90	fz*( gphi*(m field[6][j]-m field[4][j]) +
91	fphi*(m_field[7][j]-m_field[5][j]) ) );
92	dBdph1[j] = sphi*( gz*( gr*(m_field[1][j]-m_field[0][j]) +
93	fr*(m_field[3][j]-m_field[2][j]) ) +
94	fz*( gr*(m_field[5][j]-m_field[4][j]) +
95	Dot próducts of two fr*(m_field[/ dis affiel// Greblace with
96	
97	Inverse multiplication
98	float cc = c*c;
99	float cs = c*s;
100	float ss = s*s;
101	deriv[0] = cc*dBdr[1] - cs*dBdr[2] / cs*dBdphi[1]/r + ss*dBdphi[2]/r + s*B[1]/r;
102	deriv[1] = cs*dBdr[1] - ss*dBdr[2] + cc*dBdphi[1]/r - cs*dBdphi[2]/r - c*B[1]/r
103	deriv[2] = c*dBdz[1] - s*dBdz[2];
104	deriv[3] = cs*dBdr[1] + cc*dBdr[2] ss*dBdphi[1]/r - cs*dBdphi[2]/r - s*B[0]/;
105	deriv[4] = ss*dBdr[1] + cs*dBdr[2] + cs*dBdphi[1]/r + cc*dBdphi[2]/r + c*B[0]/r;
106	deriv[5] = s*dBdz[1] + c*dBdz[2];
10/	deriv[6] = c*dBdr[0] - s*dBdph1[0]/r;
108	aeriv[/] = s*aBar[i] + c*aBapn1[⊎]/r;
109	aerīv[8] = amaz[0];
110	
	b contract of the second se

40% more speedup after replacement, 5% - 20% global speedup with new code. Vectorization is yet to come.



#### **Pin Tools**



Pin is a dynamic binary instrumentation framework from Intel

- instrumentation is done on binary at run-time, eliminates need to modify or recompile the code
- can instrument from instruction level to function level, supports dynamically generated code
- can access function parameters and register contents
- can work with threaded programs
- has limited access to symbol and debug information
- creates a copy of the binary, inspects applications instructions and inserts calls to analysis functions
- used in computer architecture, security, emulation and parallel program analysis tools such as Intel's Parallel Inspector, Parallel Amplifier, Trace Analyzer and Collector, CMP\$im and many others
- Great documentation and user community (PinHeads)

#### Pin Tools

#### Improving Tracking Code



- Tracking is mostly based on vector and matrix operations
- CLHEP library is used for matrix and vector representations and operations
  - CLHEP is not performance optimized and does not vectorize well
  - It is hard to know from inspecting the code the ratios of different operations and matrix/vector sizes which happen when processing real events
- Another, vectorized vector math library is required
  - There are many libraries, which is the best?
- Pin is used to instrument CLHEP classes and operations to extract information on most commonly used objects

Pin Tools

#### Hottest 10 CLHEP Functions



Calls	Instr	<instr>/call</instr>	Call rank	Function
1778523	6392431813	3594 24	1439	CLHEP::operator*( CLHEP::HepMatrix const&,
1110020	0002401010	0004.24	1400	CLHEP::HepSymMatrix const&)
671676353	5988139520	8.92	9	CLHEP::Hep2Vector::operator()(int) const
232093102	5956556656	25.66	27	HepGeom::Transform3D::operator()(int, int) const
285282108	3709057782	13.00	21	CLHEP::Hep3Vector::operator()(int)
15015020	2170001020	201.00	210	CLHEP::HepRotation::rotateAxes( CLHEP::Hep3Vector const&,
10010930	31/9001930	201.00	319	CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const&)
20529818	2422518524	118.00	267	HepGeom::Transform3D::inverse() const
31612743	2212258670	69.98	200	CLHEP::HepSymMatrix::HepSymMatrix(CLHEP::HepSymMatrix const&)
28914115	1929106393	66.72	214	CLHEP::HepVector::HepVector(int, int)
51074716	1910115060	25.00	150	HepGeom::operator*( HepGeom::Transform3D const&,
515/4/10	1019113000	33.00	150	HepGeom::Point3D <double> const&amp;)</double>
27652274	1506352669	54.47	219	CLHEP::HepVector::HepVector( CLHEP::HepVector const&)

In order to determine a suitable replacement, these routines are instrumented with Pin and function properties are queried.

#### **CLHEP** Instrumentation



- Each hot function is instrumented by its respective analysis routine
- These functions analyzed the call parameters for hottest functions for each call and produced output to futher offline processing

#### Profiling

#### Using pin results



- From pin instrumentation we observed frequent use of 3x3, 3x5, 5x3 and 5x5 matrices.
- This information is used for testing different vector math libraries.
- 4x4 is 3D rotation(3x3) plus translation and vectorizes better
- Eigen performed best and is currently being implemented.



#### Profiling

#### Using pin results



- From pin instrumentation we observed frequent use of 3x3, 3x5, 5x3 and 5x5 matrices.
- This information is used for testing different vector math libraries.
- 4x4 is 3D rotation(3x3) plus translation and vectorizes better
- Eigen performed best and is currently being implemented.



#### Summary



- ATLAS has successfully identified points to improve in its huge codebase using profilers such as GOODA and Pin tools
- GOODA is an open source, performance profiling tool giving valuable insight about bottlenecks in the program
- Pin is very good at finding out the details of actually executed code. It makes analysis of large code bases easier
- Information gained from Pin enabled us to choose optimal vector library
- Some results are already implemented and improved performance up to 20%
- Studies are ongoing, many more improvements to come

# Thank you for your attention

Thanks to

- Graeme A. Stewart
- Rolf Seuster
- Roberto A. Vitillo

#### Backup

#### References

- GOODA home page
- Pin homepage



Backup

**ATLAS** 





# Instrumenting CLHEP::HepSymMatrix(const CLHEP::HepSymMatrix&)

