

Optimizing the ATLAS code with different profilers

Sami Kama¹
on behalf of the ATLAS Collaboration²

¹Southern Methodist University, Texas

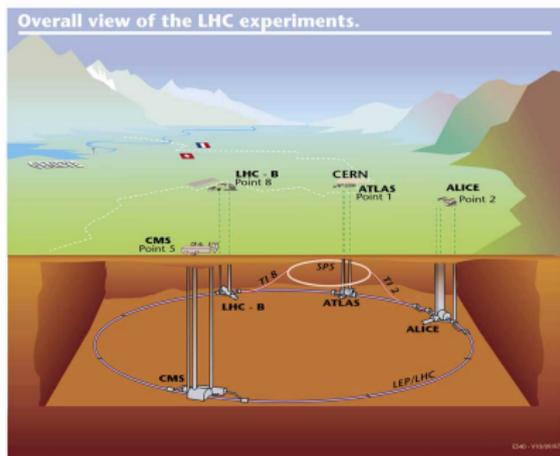
²<https://cdsweb.cern.ch/record/1386334>

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 ~ 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
- Written 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
- Written 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.



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



GOODA Example

Generic Optimization Data Analyzer **GOODA**

Reports

RecoHIPU1000EvtPinSingle Hotspots

Cycles Samples

process path module path unhalting_core_cycles wops_retired stall_cycles instruction_retired wops_retired any load_latency instruction_starvation bandwidth_saturated branch_misprediction store_res

process path	module path	unhalting_core_cycles	wops_retired	stall_cycles	instruction_retired	wops_retired any	load_latency	instruction_starvation	bandwidth_saturated	branch_misprediction	store_res
athena.py		26285302 (100%)	16722970 (63%)	26014389	33711057	9195152 (34%)	8095546 (30%)	499448 (1%)	946674 (3%)	456393 (1%)	
libc.so.6		25922896 (100%)	14847473 (57%)	25726701	33116688	8881359 (34%)	7785283 (29%)	483609 (1%)	933110 (3%)	443441 (1%)	
libm.so.2		315937 (100%)	1832252 (579%)	253488	536423	239668 (75%)	256911 (81%)	11355 (3%)	10895 (3%)	10917 (3%)	

function name offset length module process unhalting_core_cycles wops_retired stall_cycles instruction_retired wops_retired any load_latency instruction_starvation

Reports

function name	offset	length	module	process	unhalting_core_cycles	wops_retired	stall_cycles	instruction_retired	wops_retired any	load_latency	instruction_starvation
RungeKuttaPropagator...	0x258e0	0x1051	libTrkExRungeKuttaPropagator...	athena.py	1384230 (100%)	689204 (49%)	1919291	2632987	9211 (0%)	228579 (10%)	2932
operator new(unsigned lon...	0x13460	0x3da	libtcalloc_minimal.so	athena.py	652371 (100%)	279268 (42%)	889013	1133805	240962 (36%)	234114 (33%)	3264
operator new(unsigned lon...	0x2da	0x2da	libtcalloc_minimal.so	athena.py	479802 (100%)	203897 (42%)	658662	873682	147872 (30%)	112685 (23%)	3697
libS1SpacePointsSeedTool...	0xd74	0xd74	libS1SpacePointsSeedTool...	athena.py	622008 (100%)	293622 (47%)	887119	965745	227310 (36%)	503 (0%)	2164
libT_TRT_TrajectoryElem...	0xc180	0x809	libT_TRT_TrackExtensionTool...	athena.py	551277 (100%)	421609 (76%)	929595	382504	542788 (98%)	1641 (0%)	139933
libT_TRT_TrajectoryElem...	0x15680	0xc40	libT_TRT_TrackExtensionTool...	athena.py	287540 (100%)	147594 (51%)	379448	455976	182745 (63%)	99303 (34%)	5885
libT_TRT_TrajectoryElem...	0xbf80	0x40b	libT_TRT_TrackExtensionTool...	athena.py	307011 (100%)	189831 (61%)	235244	352133	210421 (68%)	112561 (36%)	2822
libT_TRT_TrajectoryElem...	0x5940	0x1c5	libT_TRT_TrackExtensionTool...	athena.py	449764 (100%)	287254 (63%)	313579	424356	127635 (28%)	105407 (23%)	3475
libT_TRT_TrajectoryElem...	0x5220	0xb93	libT_TRT_TrackExtensionTool...	athena.py	271153 (100%)	111591 (41%)	475779	654461	175 (0%)	107792 (39%)	1181
libT_TRT_TrajectoryElem...	0x2770	0xf55	libT_TRT_TrackExtensionTool...	athena.py	103548 (100%)	65400 (63%)	74474	180722	36798 (35%)	103723 (100%)	963
libT_TRT_TrajectoryElem...	0x64280	0x1ce	libT_TRT_TrackExtensionTool...	athena.py	496342 (100%)	309302 (62%)	398039	402833	131 (0%)	919 (0%)	197
libT_TRT_TrajectoryElem...	0x5220	0xb93	libT_TRT_TrackExtensionTool...	athena.py	271153 (100%)	111591 (41%)	475779	654461	175 (0%)	107792 (39%)	1181
libT_TRT_TrajectoryElem...	0x26140	0x573	libT_TRT_TrackExtensionTool...	athena.py	167081 (100%)	106231 (63%)	126570	173298	4551 (2%)	99610 (59%)	1406
libT_TRT_TrajectoryElem...	0xace0	0x108	libT_TRT_TrackExtensionTool...	athena.py	224926 (100%)	94451 (41%)	251920	382992	10042 (4%)	60273 (26%)	280
libT_TRT_TrajectoryElem...	0x6850	0x976	libz.so.1.2.3	athena.py	267347 (100%)	116870 (43%)	405506	449760	26997 (10%)	328 (0%)	88
libT_TRT_TrajectoryElem...	0x1e320	0x43e	libT_TRT_TrackExtensionTool...	athena.py	174563 (100%)	42079 (24%)	367583	420989	169072 (96%)	3388 (2%)	20193
libT_TRT_TrajectoryElem...	0xc1670	0x5ce	libT_TRT_TrackExtensionTool...	athena.py	117586 (100%)	56777 (48%)	153151	217375	14067 (11%)	40999 (34%)	1293



GOODA Example

Generic Optimization Data Analyzer **GOODA**

RecoHIPU1000EvtPinSingle Hotspots

Reports

- DQHistMerge1
- G4ExampleN03
- Sample
- MP4HLT
- MIG
- AthenaReco
- RecoCerebro
- RecoOUJoan
- CallGraphCerebr
- RecoCG
- RecoOUJoanPin
- RecoOUJoanPinP
- RecoEventNoPin
- RecoEventNoPin
- RecoEventPPS
- RecoNoMux
- RecoNoMuxPin
- RecoNoMuxPPS
- RecoNoMuxEver
- RecoNoMuxEver
- ParallelG4
- RecoNoMuxEver
- RecoNoMuxEver
- AtlasG4
- AtlasG4Pin
- RecoHIPU100Ev
- RecoHIPU100E
- AtlasG4Andreas
- HLTDummy

Enter search form

process path	module path	unhalted_core_cycles	wops_retired	stall_cycles	instruction_retired	wops_retired_any	load_latency	instruction_starvation	bandwidth_saturated	branch_misprediction	store_res
		26285302 (100%)	16722970 (63%)	26014389	33711057	9195152 (34%)	8095546 (30%)	499448 (1%)	946674 (3%)	456393 (1%)	
athena.py		25922896 (100%)	14847473 (57%)	25726701	33116688	8881359 (34%)	7785283 (29%)	483609 (1%)	933110 (3%)	443441 (1%)	
velinux		315937 (100%)	1832252 (579%)	253488	536423	239668 (75%)	256911 (81%)	11355 (3%)	10895 (3%)	10917 (3%)	

Processes

Enter search form

function	class	module	process	unhalted_core_cycles	wops_retired	stall_cycles	instruction_retired	wops_retired_any	load_latency	instruction_starvat	
Trk::RungeKuttaPropagator...	0x258e0	0x1051	libTrkExRungeKuttaPropagator...	athena.py	1384230 (100%)	689204 (49%)	1919291	2632987	9211 (0%)	228579 (16%)	2932
operator new(unsigned lon...	0x13460	0x3da	libtcalloc_minimal.so	athena.py	652571 (100%)	279268 (42%)	889013	1133805	240962 (36%)	234114 (33%)	3264
operator delete(void*)	0x12c10	0x2da	libtcalloc_minimal.so	athena.py	479802 (100%)	203897 (42%)	658662	873682	147872 (30%)	112685 (23%)	3697
InDet::S1SpacePointsSeedf...	0xb7b7d0	0xd74	libS1SpacePointsSeedTool_...	athena.py	622008 (100%)	293622 (47%)	887119	965745	227310 (36%)	503 (0%)	2164
InDet::TRT_TrajectoryElem...	0xc6180	0xb09	libTRT_TrackExtensionTool_...	athena.py	551277 (100%)	421609 (76%)	929595	382504	542788 (98%)	1641 (0%)	139933
bsolInterp...	0x15680	0xc40	libBFFieldCore.so	athena.py	287540 (100%)	147594 (51%)	379448	455976	182745 (63%)	99303 (34%)	5885
__dynamic_cast@CXXABI_1.3	0xb7200	0x11f	libstdc++.so.6.0.18	athena.py	307011 (100%)	189831 (61%)	235244	352133	210421 (60%)	112561 (36%)	2822
Trk::MagneticFieldMapSoc...	0x5940	0x1c5	libTrkMagFieldUtils.so	athena.py	449764 (100%)	287254 (63%)	315379	424356	127635 (28%)	105407 (23%)	3475
master_0_gbaggz_...	0xfb80	0x4a0b	libBFFieldStand.so	athena.py	360218 (100%)	166033 (46%)	537975	621755	38592 (10%)	30544 (0%)	416
Trk::RungeKuttaPropagator...	0x277f0	0xf55	libTrkExRungeKuttaPropagator...	athena.py	103548 (100%)	65400 (63%)	74474	180722	36798 (35%)	103723 (100%)	963
InDet::TRT_Trajectory_kk...	0x64280	0x1ce	libTRT_TrackExtensionTool_...	athena.py	496342 (100%)	309302 (62%)	398039	402833	131 (0%)	919 (0%)	197
Trk::PatternTrackParame...	0x5220	0xb93	libTrkPatternParameters.so	athena.py	271153 (100%)	111591 (41%)	475779	654461	175 (0%)	107792 (39%)	1181
Trk::RungeKuttaUtils::tra...	0x179f0	0x923	libTrkExUtils.so	athena.py	249429 (100%)	134726 (54%)	358266	453031	37970 (13%)	113371 (45%)	416
Trk::RungeKuttaPropagator...	0x26140	0x573	libTrkExRungeKuttaPropagator...	athena.py	167081 (100%)	106231 (63%)	126570	173298	4551 (2%)	99610 (59%)	1406
soleFittersSel...	0x8ace0	0x108c	libBFFieldCore.so	athena.py	224926 (100%)	94451 (41%)	251920	382992	10042 (4%)	60273 (26%)	280
deflate_slow	0x8650	0x976	libz.so.1.2.3	athena.py	267347 (100%)	116870 (43%)	405506	449760	26997 (10%)	328 (0%)	88
TTTrainedNetwork::calcul...	0x1e320	0x43e	libTrkNeuralNetworkUtilsLib...	athena.py	174563 (100%)	42079 (24%)	367583	420909	169072 (96%)	3388 (2%)	20193
__cxxabi_v1_class_t...	0xc1670	0x5ce	libstdc++.so.6.0.18	athena.py	117586 (100%)	56777 (48%)	153151	217375	14067 (11%)	40999 (34%)	1293

Help



GOODA Example

Generic Optimization Data Analyzer **GUJ**

RecoHIPU1000EvtPinSingle Hotspots

Reports

- DQHistMerge1
- G4ExampleN03
- Sample
- MP4HLT
- MIG
- AthenaReco
- RecoCerebro
- RecoOJoan
- CallGraphCerebr
- RecoCG
- RecoOJoanPin
- RecoOJoanPinP
- RecoEventNoPin
- RecoEventNoPin
- RecoEventPPS
- RecoNoMux
- RecoNoMuxPin
- RecoNoMuxPPS
- RecoNoMuxEver
- RecoNoMuxEver
- RecoNoMuxEver
- ParallelG4
- RecoNoMuxEver
- RecoNoMuxEver
- AtlasG4
- AtlasG4Pin
- RecoHIPU1000Ev
- RecoHIPU1000E
- AtlasG4Andreas
- HLTDummy

Enter search form

process path	module path	unhalted_core_cycles	wups_retired	stall_cycles	instruction_retired	wups_retired_any	load_latency	instruction_starvation	bandwidth_saturated	branch_misprediction	store_res
		26285302 (100%)	16722970 (63%)	26014389	33711057	9195152 (34%)	8095546 (30%)	499448 (1%)	946674 (3%)	456393 (1%)	
athena.py		25922896 (100%)	14847473 (57%)	25726701	33116688	8881359 (34%)	7785283 (29%)	483609 (1%)	933110 (3%)	443441 (1%)	
velinux		315937 (100%)	1832252 (579%)	253488	536423	239668 (75%)	256911 (81%)	11355 (3%)	10895 (3%)	10917 (3%)	

Enter search form

function name	offset	length	module	process	unhalted_core_cycles	wups_retired	stall_cycles	instruction_retired	wups_retired_any	load_latency	instruction_starvat
Trk::RungeKuttaPropagator...	0x158e0	0x1051	libTrkExRungeKuttaPropagator...	athena.py	1384230 (100%)	689204 (49%)	1919291	2632987	9211 (0%)	228579 (10%)	2932
operator new(unsigned lon...	0x13460	0x3da	libtcalloc_minimal.so	athena.py	652571 (100%)	279268 (42%)	889013	1133805	240962 (36%)	234114 (33%)	3264
operator delete(void*)	0x12c10	0x2da	libtcalloc_minimal.so	athena.py	479802 (100%)	203097 (42%)	658662	873682	147872 (30%)	112685 (23%)	3697
InDet::S1SpacePointsSeedF...	0xd7f40	0xd74	libS1SpacePointsSeedTool_...	athena.py	622008 (100%)	293622 (47%)	887119	965745	227310 (36%)	503 (0%)	2164
InDet::TRT_TrajectoryElem...	0x4180	0x809	libTRT_TrackExtensionTool_...	athena.py	551277 (100%)	421609 (76%)	929595	382504	542788 (98%)	1641 (0%)	139933
bsolInterp...	0x16880	0xc40	libBFFieldCore.so	athena.py	287540 (100%)	147594 (51%)	379448	455976	182745 (63%)	99303 (34%)	5885
__dynamic_cast@CXXABI_1.3	0x49200	0x11f	libstdc++.so.6.0.18	athena.py	307011 (100%)	189831 (61%)	235244	352133	210421 (60%)	112561 (36%)	2822
Trk::MagneticFieldMapSo...	0x1040	0x1c5	libTrkMagFieldUtils.so	athena.py	449764 (100%)	287254 (63%)	315379	424356	127635 (20%)	105407 (23%)	3475
master_0_gbagg_...	0x1680	0x10	libTrkFieldCore.so	athena.py	160318 (100%)	166033 (46%)	537975	621755	38592 (10%)	30544 (0%)	416
Trk::RungeKuttaPropagator...	0x277f0	0x148	libTrkExRungeKuttaPropagator...	athena.py	498148 (100%)	65400 (63%)	74474	180722	36798 (35%)	103723 (100%)	963
InDet::TRT_Trajectory_kk...	0x6c380	0x1ce	libTRT_TrackExtensionTool_...	athena.py	498142 (100%)	309302 (62%)	398039	402833	131 (0%)	919 (0%)	197
Trk::PatternTrackParame...	0x1220	0xb93	libTrkPatternParameters.so	athena.py	271153 (100%)	111591 (41%)	475779	654461	175 (0%)	107792 (39%)	1181
Trk::RungeKuttaUtils::tra...	0x179f0	0x923	libTrkExUtils.so	athena.py	249429 (100%)	134726 (54%)	358266	453031	37970 (13%)	113371 (45%)	416
Trk::RungeKuttaPropagator...	0x23440	0x573	libTrkExRungeKuttaPropagator...	athena.py	167081 (100%)	106231 (63%)	126570	173298	4551 (2%)	99610 (59%)	1406
soleFittersSel...	0x1ce0	0x10c8	libBFFieldCore.so	athena.py	224926 (100%)	94451 (41%)	251920	382992	10042 (4%)	60273 (26%)	2003
deflate_slow	0x1680	0x976	libz.so.1.2.3	athena.py	267347 (100%)	116870 (43%)	405506	449760	26997 (10%)	328 (0%)	88
TTTrainedNetwork::calcul...	0x13320	0x43e	libTrkNeuralNetworkUtilsLib...	athena.py	174563 (100%)	42079 (24%)	367583	420909	169072 (96%)	3388 (2%)	20193
__cxabi_vl1_class_t...	0x1670	0x5ce	libstdc++.so.6.0.18	athena.py	117586 (100%)	56777 (48%)	153151	217375	14067 (11%)	40999 (34%)	1293

Hotspot functions



GOODA Example

Generic Optimization Data Analyzer **GOODA**

Reports

RecoHIPU1000EvtPinSingleHotspots

Cycles Samples

process path module path unhalting_core_cycles wops_retired stall_cycles instruction_retired wops_retired_any load_latency instruction_starvation bandwidth_saturated branch_misprediction store_res

process path	module path	unhalting_core_cycles	wops_retired	stall_cycles	instruction_retired	wops_retired_any	load_latency	instruction_starvation	bandwidth_saturated	branch_misprediction	store_res
athena.py		26285302 (100%)	16722970 (63%)	26014389	33711057	9195152 (34%)	8095546 (30%)	499448 (1%)	946674 (3%)	456393 (1%)	
velinux		315937 (100%)	1832252 (579%)	253488	536423	239668 (75%)	256911 (81%)	11355 (3%)	10895 (3%)	10917 (3%)	

Cycles Samples

function name offset length module process unhalting_core_cycles wops_retired stall_cycles instruction_retired wops_retired_any load_latency instruction_starvation

function name	offset	length	module	process	unhalting_core_cycles	wops_retired	stall_cycles	instruction_retired	wops_retired_any	load_latency	instruction_starvation
Trk::RungeKuttaPropagator...	0x250e0	0x1051	libTrkExRungeKuttaPropagator...	athena.py	684230 (100%)	689204 (49%)	3919291	2632987	9211 (0%)	228179 (10%)	2031
operator new(unsigned lon...	0x13460	0x3da	libtcalloc_minimal.so	athena.py	652571 (100%)	279268 (42%)	889013	11105	240962 (30%)	234114 (13%)	3264
operator delete(void*)	0x12c10	0x2da	libtcalloc_minimal.so	athena.py	479802 (100%)	287540 (51%)	379448	455976	182745 (63%)	99303 (34%)	3697
InDet::S1SpacePointsSeed...	0xb7b7d0	0xd74	libS1SpacePointsSeedTool_...	athena.py	622008 (100%)	293013 (47%)	808113	842013	120000 (15%)	10000 (0%)	2164
InDet::TRT_TrajectoryElem...	0x6c180	0x809	libTRT_TrackExtensionTool_...	athena.py	551277 (100%)	421609 (76%)	929595	382504	542788 (98%)	1641 (0%)	139933
bsolInterp...	0x15680	0xc40	libBFFieldCore.so	athena.py	287540 (100%)	147594 (51%)	379448	455976	182745 (63%)	99303 (34%)	5885
__dynamic_cast@CXXABI_1.3	0xb7200	0x11f	libstdc++.so.6.0.18	athena.py	307011 (100%)	189831 (61%)	235244	352133	210421 (68%)	112561 (36%)	2822
Trk::MagneticFieldMapSoc...	0x5940	0x1c5	libTrkMagFieldUtils.so	athena.py	449764 (100%)	287254 (63%)	315379	424356	127635 (28%)	105407 (23%)	3475
master_0_gbagg_...	0xf80	0x4a0b	libBFFieldStand.so	athena.py	360218 (100%)	166033 (46%)	537975	621755	38592 (10%)	30544 (0%)	416
Trk::RungeKuttaPropagator...	0x277f0	0xf55	libTrkExRungeKuttaPropagator...	athena.py	103548 (100%)	65400 (63%)	74474	180722	36798 (35%)	103723 (100%)	963
InDet::TRT_Trajectory_kk...	0x64280	0x1ce	libTRT_TrackExtensionTool_...	athena.py	496342 (100%)	309302 (62%)	398039	402833	131 (0%)	919 (0%)	197
Trk::PatternTrackParame...	0x5220	0xb93	libTrkPatternParameters.so	athena.py	271153 (100%)	111591 (41%)	475779	654461	175 (0%)	107792 (39%)	1181
Trk::RungeKuttaUtils::tra...	0x179f0	0x923	libTrkExUtils.so	athena.py	249429 (100%)	134726 (54%)	358266	453031	37970 (13%)	113371 (45%)	416
Trk::RungeKuttaPropagator...	0x26140	0x573	libTrkExRungeKuttaPropagator...	athena.py	167081 (100%)	106231 (63%)	126570	173298	4551 (2%)	99610 (59%)	1406
soleFittersStl...	0xace0	0x108c	libBFFieldCore.so	athena.py	224926 (100%)	94451 (41%)	251920	382992	10042 (4%)	60273 (28%)	2063
deflate_slow	0x6850	0x976	libz.so.1.2.3	athena.py	267347 (100%)	116870 (43%)	405506	449760	26997 (10%)	328 (0%)	88
TTTrainedNetwork::calcul...	0x1e320	0x43e	libTrkNeuralNetworkUtilsLib...	athena.py	174563 (100%)	42079 (24%)	367583	420909	169072 (96%)	3388 (2%)	20193
__cxxabi1.1_vti_class_t...	0xc1670	0x5ce	libstdc++.so.6.0.18	athena.py	117506 (100%)	56777 (48%)	153151	217375	14067 (11%)	40999 (34%)	1293

PMU events, grouped

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



Magnetic field code details

Dissassembly window				Source window											
pc	retnc_30	principal_file	initial_file	dissassembly	line number	source	unretired_ops_cycles	ops_retired	stall_cycles	instruction_retired	ops_retired_any	load_latency	instruction_latency	branch_saturation	branch
91	IHagFieldSvc.h	95	BFieldCache.h	mulss %xmm0,%xmm1	82	// no. return defn.	103506891 (100%)	75992130 (73%)	71409388	83625478	95521 (0%)	95489 (0%)	307 (0%)	472767 (0%)	
91	IHagFieldSvc.h	95	BFieldCache.h	addss %xmm1,%xmm2	83	bxyz[0] = bxyz[1] -									
91	IHagFieldSvc.h	95	BFieldCache.h	mulss %xmm1,%xmm2	84	return;									
91	IHagFieldSvc.h	95	BFieldCache.h	addss %xmm1,%xmm2	85	}									
91	IHagFieldSvc.h	95	BFieldCache.h	add %eax,%ebx	86	// refresh cache									
91	IHagFieldSvc.h	83	BFieldCache.h	cmp %eax,%ebx	87	fillFieldCache(z, r,	131338 (100%)	59674 (45%)	190120	256662	2678 (2%)	161 (0%)	65 (0%)	32	
91	IHagFieldSvc.h	83	BFieldCache.h	jne %eax,%ebx	88	}									
91	IHagFieldSvc.h			Basic Block 22 <0x372d>	89										
91	IHagFieldSvc.h	99	BFieldCache.h	movss %xmm0,%xmm1	90	// do interpolation									
91	IHagFieldSvc.h	99	BFieldCache.h	null	91	m_cache.getB(xyz, r, ..	82971948 (100%)	59682740 (71%)	61724896	70339686	73327 (0%)	54164 (0%)	226 (0%)	15291	
91	IHagFieldSvc.h	102	BFieldCache.h	movss %xmm0,%xmm1	92										
91	IHagFieldSvc.h	102	BFieldCache.h	null	93	// add bnot savart con.									
91	IHagFieldSvc.h	99	BFieldCache.h	mulss %xmm15,%xmm3	94	if (m_cond) {	97 (100%)	125 (128%)		27	97 (100%)				
91	IHagFieldSvc.h	100	BFieldCache.h	movaps %xmm15,%xmm12	95	for (std::vector<BF	127620 (100%)	110944 (86%)	4279	54157	516 (0%)	290 (0%)			
91	IHagFieldSvc.h	100	BFieldCache.h	movaps %xmm15,%xmm12	96	it->addBnotSavartC									

- 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.



Magnetic field code details

pc	pcfile_38	pcfile_39	pcfile_40	initial file	disassembly	what	line number	source	unretired_ops_cycles	ops_retired_stall_cycles	ops_retired	load_latency	instruction_latency	branch		
91							103508091		103508091 (100%)	7592130 (73%)	71409300	83625478	95521 (0%)	95489 (0%)	307 (0%)	472767 (0%)
91	INagFieldSvc.h	95	BFieldCache.h				83	bxyz[0] = bxyz[i];								
91	INagFieldSvc.h	95	BFieldCache.h		addss %xmm1,%xmm2		84	return;								
91	INagFieldSvc.h	95	BFieldCache.h		mulss %xmm1,%xmm2		85	}								
91	INagFieldSvc.h	95	BFieldCache.h		movss %xmm2,(%r15,%eax)		86	// refresh cache								
91	INagFieldSvc.h	95	BFieldCache.h		add \$0x2,%rip		87	fillfdCache(z, r, ...	131338 (100%)	59674 (45%)	190120	256662	2678 (2%)	161 (0%)	65 (0%)	32
91	INagFieldSvc.h	83	BFieldCache.h		cmp \$0x0,%edx		88	}								
91	INagFieldSvc.h	83	BFieldCache.h		jpe \$0eef		89	// do interpolation								
91	INagFieldSvc.h	83	BFieldCache.h		jmp \$0eef		90	m_cache.getB(xyz, r, ...	82971948 (100%)	59682740 (71%)	61724896	70339606	73327 (0%)	54164 (0%)	226 (0%)	15291
91	INagFieldSvc.h	99	BFieldCache.h		movss 0x104(%rsp),%xmm...		91									
91	INagFieldSvc.h	99	BFieldCache.h		null		92									
91	INagFieldSvc.h	102	BFieldCache.h		movss 0x104(%rsp),%xmm...		93	// add bnot savart com...								
91	INagFieldSvc.h	102	BFieldCache.h		null		94	if (m_cond) {	97 (100%)	125 (128%)		27	97 (100%)			
91	INagFieldSvc.h	99	BFieldCache.h		mulss %xmm15,%xmm3		95	for (std::vector<BF...	127620 (100%)	110944 (86%)	4279	54157	516 (0%)	290 (0%)		
91	INagFieldSvc.h	100	BFieldCache.h		movaps %xmm15,%xmm12		96	it->addBnotSavartC...								

Hotspot basic block is highlighted

- Detailed view contains both disassembly 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.



Magnetic field code details

line number	source	unhalted_core_cycles	uops_retired_stall_cycles	instruction_retired	uops_retired_any	load_latency	instruction_starvation	branch_taken	branch
87	fillFieldCache(z, r...	131330 (100%)	59674 (45%)						
88	}								
89									
90	// do interpolation								
91	m_cache.getB(xyz, r, ...	82972948 (100%)	59682740 (71%)						
92									
93	// add biot savart com...								
94	if (m_cond) {	97 (100%)	125 (128%)						
95	for (std::vector<BF...	127620 (100%)	110944 (86%)						
96	it->addBiotSavart(

- Detailed view contains both disassembly 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] = sph*( 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 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]/r;
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  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] = sph*( 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 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]/r;
105 deriv[4] = ss*dBdr[1] + cs*dBdr[2] + cc*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 }

```

Lots of 1/r, replace with
inverse multiplication



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] = sph*( 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 ss = s*s;
101    deriv[0] = cc*dBdr[1] - cs*dBdr[2] + ss*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]/r;
105    deriv[4] = ss*dBdr[1] + cs*dBdr[2] + cc*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 }

```

Dot products of two vectors!

Lots of 1/r, replace with inverse multiplication



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] = sph*( 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 ss = s*s;
101    deriv[0] = cc*dBdr[1] - cs*dBdr[2] + ss*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]/r;
105    deriv[4] = ss*dBdr[1] + cs*dBdr[2] + cc*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 }

```

Dot products of two vectors!

Lots of 1/r, replace with inverse multiplication

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)



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



Hottest 10 CLHEP Functions

Calls	Instr	<instr>/call	Call rank	Function
1778523	6392431813	3594.24	1439	CLHEP::operator*(CLHEP::HepMatrix const&, 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)
15815930	3179001930	201.00	319	CLHEP::HepRotation::rotateAxes(CLHEP::Hep3Vector const&, 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)
51974716	1819115060	35.00	150	HepGeom::operator*(HepGeom::Transform3D const&, HepGeom::Point3D<double> const&)
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.



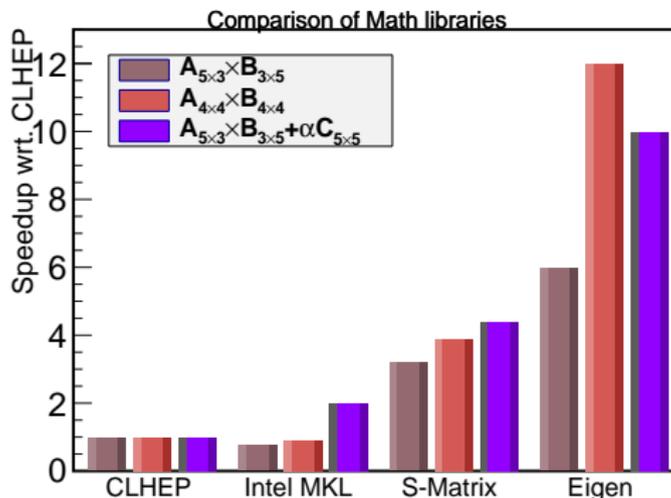
CLHEP Instrumentation

- 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 further offline processing

```
void InstFunc(ADDRINT addr, std::string& msg, par1 v1, par2 v2){
//do stuff
} //analysis code
VOID InstHOOK(RTN rtn,VOID *v){//called for each routine
RTN_Open(rtn);//read the routine from binary
std::string *msg=new std::string;//extra param for analysis func
if (RTN_Name(rtn).compare("<mangledName>") == 0) {
    RTN_InsertCall(rtn, IPOINT_BEFORE, (AFUNPTR)InstFunc,
        IARG_RETURN_IP,
        IARG_PTR, msg,
        IARG_FUNCARG_ENTRYPOINT_VALUE, 1,//func param1
        IARG_FUNCARG_ENTRYPOINT_VALUE, 2,//func param2
        IARG_END);//instrument <mangledName> function
}
}
```

Using pin results

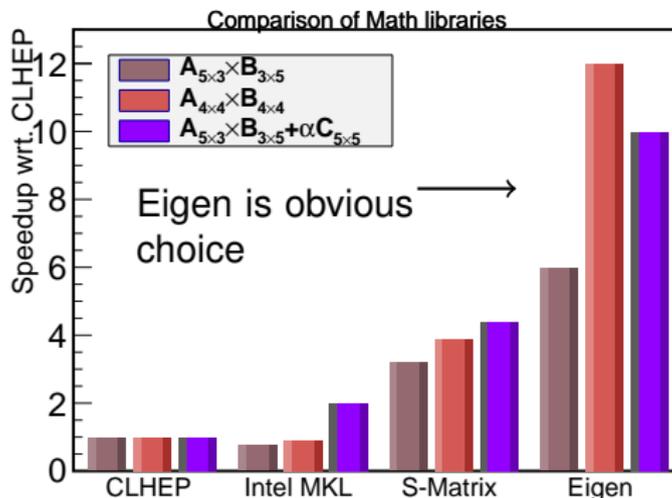
- From pin instrumentation we observed frequent use of 3x3, 3x5, 5x3 and 5x5 matrices.
- This information is used for testing different vector math libraries.
- 4x4 is 3D rotation(3x3) plus translation and vectorizes better
- Eigen performed best and is currently being implemented.





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

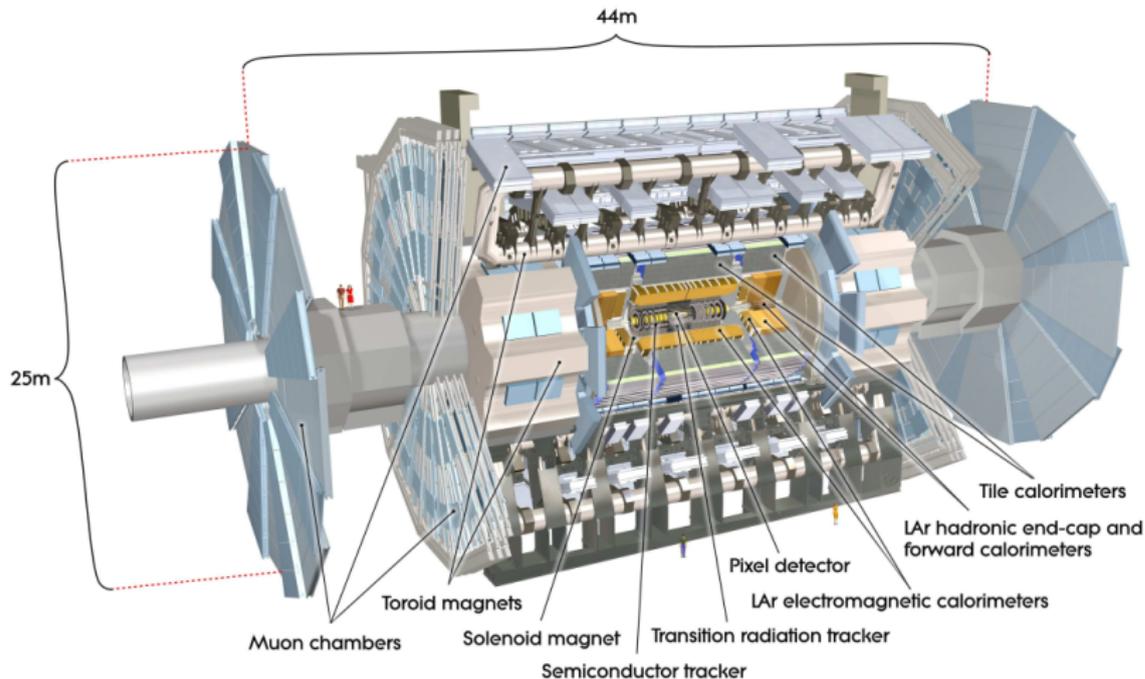
- Graeme A. Stewart
- Rolf Seuster
- Roberto A. Vitillo



References

- [GOODA home page](#)
- [Pin homepage](#)

ATLAS



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



```
VOID CLHEPHepSymMatrixCopyConst (ADDRINT addr,
                                  std::map<int, uint64_t> &counts,
                                  CLHEP::HepSymMatrix const& par1) {
    int hash=calcHash(par1.num_row(), par1.num_col());
    std::map<int, uint64_t>::iterator it;
    if ((it=counts.find(hash))==counts.end()) {
        counts[hash]=1;
    } else {
        it->second++;
    }
}
```