Performance of ATLAS G4 Simulation

(17th Geant4 Collaboration Week, Chartres)

Elmar Ritsch (Univ. Innsbruck, CERN)
on behalf of the ATLAS Collaboration

September 10, 2012

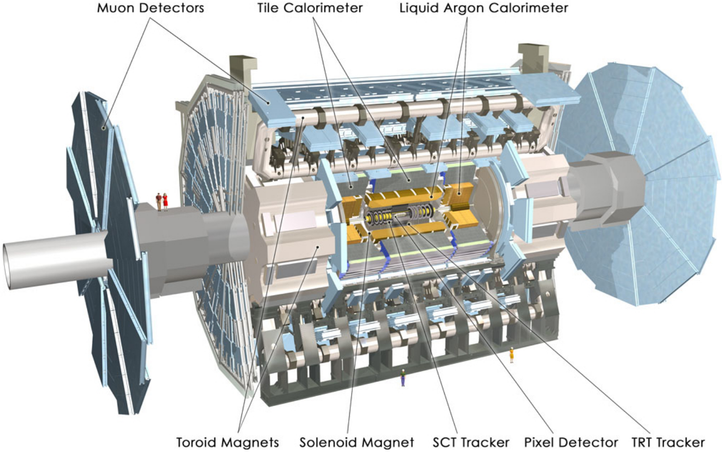
Overview

- **Introduction**
  use of Geant4 in ATLAS, Code Structure
- **Hotspots Analysis**
  UserSteppingAction, GPerfTools
- **Optimizations**
  math library, call graphs
- **The Integrated Simulation Framework**
  a novel approach to detector simulation

The ATLAS Detector

The ATLAS Experiment at the LHC

- **Inner Detector**: Tracker with solenoid magnetic field
- **Calorimeter**: electromagnetic and hadronic
- **Muon Spectrometer**: Muon Tracker with toroid magnetic field



The ATLAS Geant4 Framework

- **FADS**: Framework for ATLAS Detector Simulation
- Object Oriented, flexible and detector independent approach to setup and run detector simulation with Geant4:
  - detector geometry construction
  - magnetic field
  - sensitive detector elements
  - MC truth handling
  - simulation parameters (stepping, physics list, ...)
- dates back to 2001, before the current ATLAS software framework `athena` was put in place
- combination of C++ interfaces and implementation with Python configuration
- *"The ATLAS Simulation Infrastructure"* (doi: 10.1140/epjc/s10052-010-1429-9)

Hotspots Analysis with UserSteppingAction

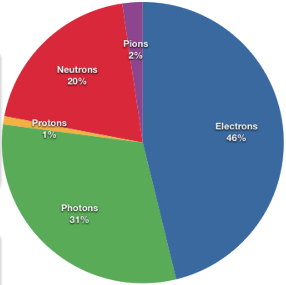
UserSteppingAction

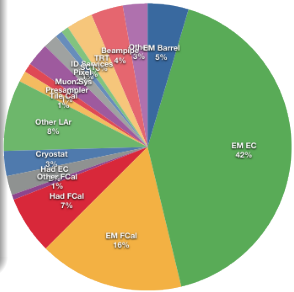
- using G4Timers to sum up CPU time spent
  - per detector volume
  - per particle type

Discoveries

- simulation of ttbar events
- break down per detector volume and per particle type
- CPU consumption dominated by:
  electrons, **photons**, **neutrons**
- hottest detector element:
  Electromagnetic end-cap calorimeter (EMEC)

→ deeper look into EMEC geometry





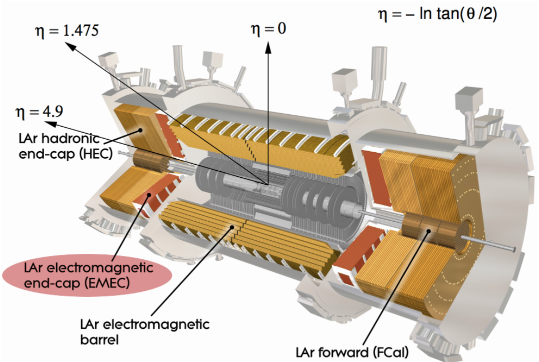
ATLAS LiquidArgon Calorimeter

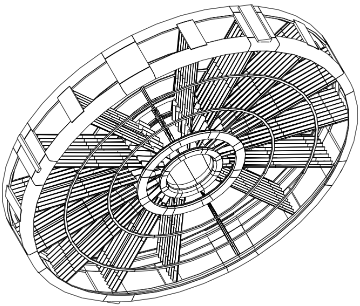
LAr Calorimeter

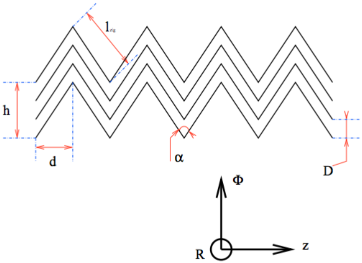
- three components:

  barrel (central)
  endcap (central-forward)
  forward

- EMEC *LAr Wheel* coverage:
  $1.475 < |\eta| < 4.9$



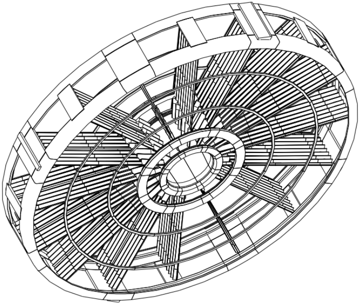




Electromagnetic endcap Calorimeter: "LAr Wheel"

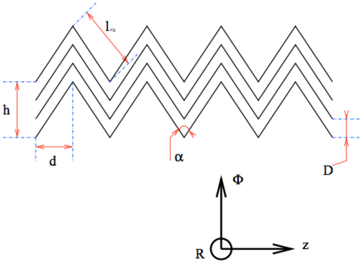
LAr Wheel Geometry

- very complex geometry
- zigzag shape in ϕ-z plane
- zigzag amplitude increases with r
- many of those zigzag shaped layers stacked on top of each other to form a wheel



ATLAS Implementation

- custom `G4VSolid` implementation
- *the* main CPU time consumer in ATLAS G4
- people working on improving the implementation
    - short term: code performance improvements
    - long term: new algorithmic approaches, ie dividing up into native Geant4 volumes



Hotspots Analysis with GPerfTools

GPerfTools and how we use it

- samples the stack trace of a program every couple of milliseconds (default: 10ms)
- hot functions will appear more often in the stack trace
- generate daily html tables with CPU timing information per function (tracked in DB)

Discoveries

- LArWheel appearing again (surprise!)
- unexpected: particle `Definition()` calls take $>$ 0.5% of total runtime
- magnetic field lookup `G4Field::GetFieldValue()` takes
  - $\sim$ 8% of ttbar simulation
  - $\sim$ 20% of single muon simulation

self hits	rel.	acc.	total hits	rel.	function name
5414	8.1%	8.1%	6010	9.0%	LArWheelCalculator::DistanceToTheNeutralFibre
2405	3.6%	11.8%	2736	4.1%	master.0.gbmagzsb_
1711	2.6%	14.3%	1978	3.0%	G4PolyconeSide::DistanceAway
1395	2.1%	16.4%	1439	2.2%	_init
1260	1.9%	18.3%	2078	3.1%	G4PhysicsVector::Value
1156	1.7%	20.1%	3253	4.9%	G4Navigator::LocateGlobalPointAndSetup
1129	1.7%	21.8%	1606	2.4%	G4PolyconeSide::PointOnCone
1104	1.7%	23.4%	1104	1.7%	atan2
1040	1.6%	25.0%	39198	58.9%	G4SteppingManager::DefinePhysicalStepLength
938	1.4%	26.4%	2287	3.4%	G4PolyconeSide::Inside
911	1.4%	27.8%	933	1.4%	G4IntersectingCone::LineHitsCone1
911	1.4%	29.1%	1740	2.6%	G4UniversalFluctuation::SampleFluctuations
859	1.3%	30.4%	7822	11.8%	G4VoxelNavigation::ComputeStep

function

id	selfcounts	totalcounts	fn_name	lib
307	0.03	0.43	G4ElectroNuclearCrossSection::IsIsoApplicable	GEANT4

calls

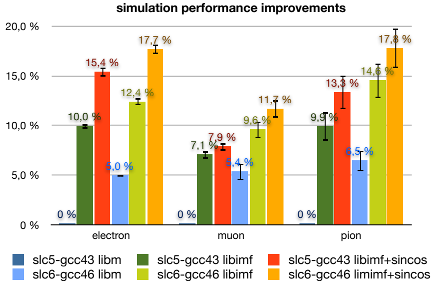
counts	acc	selfcounts	totalcounts	fn_name	lib
0.09	0.09	0.00	0.00	global constructors keyed to G4Electron.cc	unknown
0.07	0.16	2.19	2.26	_init	unknown
0.06	0.22	0.00	0.00	global constructors keyed to G4Positron.cc	unknown
0.06	0.28	0.14	0.14	G4Positron::Definition	GEANT4
0.05	0.33	0.16	0.16	G4Electron::Definition	GEANT4
0.04	0.37	0.00	0.04	G4Electron::Electron	GEANT4
0.03	0.40	0.00	0.03	G4Positron::Positron	GEANT4

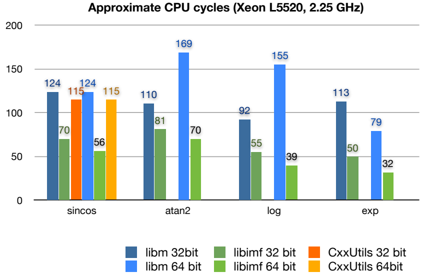
Observations

- switching compiler and SLC version from slc5-gcc43 to slc6-gcc46:
  $>5\%$ speedup
- preloading of Intel math library (*libimf*):
  7-10% speedup
- replacing `fsincos` CPU instruction by libimf function:
  3-5% speedup
- simulation output changes, due to different accuracy in the floating point functions

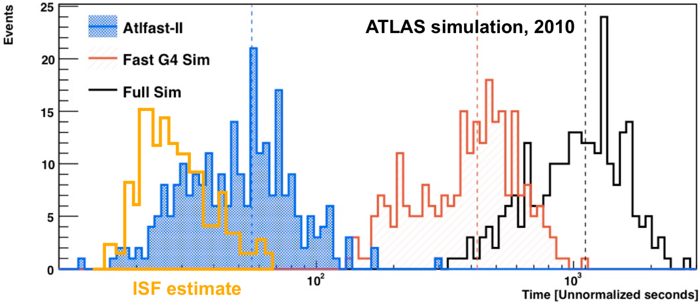
Notes

- only single particle jobs (difficult to get absolute numbers for complex events)
- question: will it be reproducible on different hardware?





Time for Something New



As a result of speeding up simulation

- number of ATLAS detector simulation engines increasing: **Geant4**, **FastCaloSim**, **Fatras**, **FrozenShowers**, **Parametrized Punch-Through**

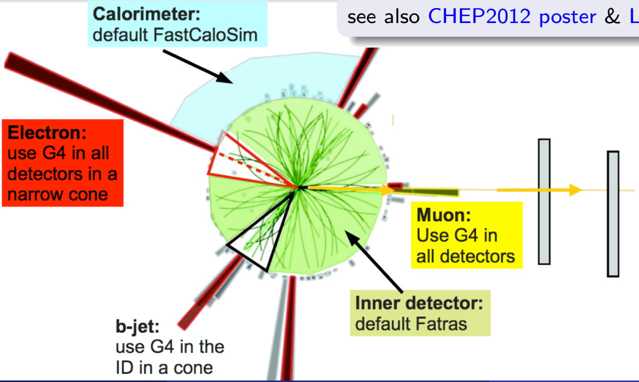
→ partly complex and incompatible setups

The Integrated Simulation Framework (ISF)

ISF Vision

- one framework for various simulation engines
  - core ISF responsibilities:
    ISF particle stack, particle routing, MC truth handling, barcode service
- allow for simulation engine selection on a particle level
  - speedup expected
  - modularity allows for parallelization approaches

see also CHEP2012 poster & Lightning Talk



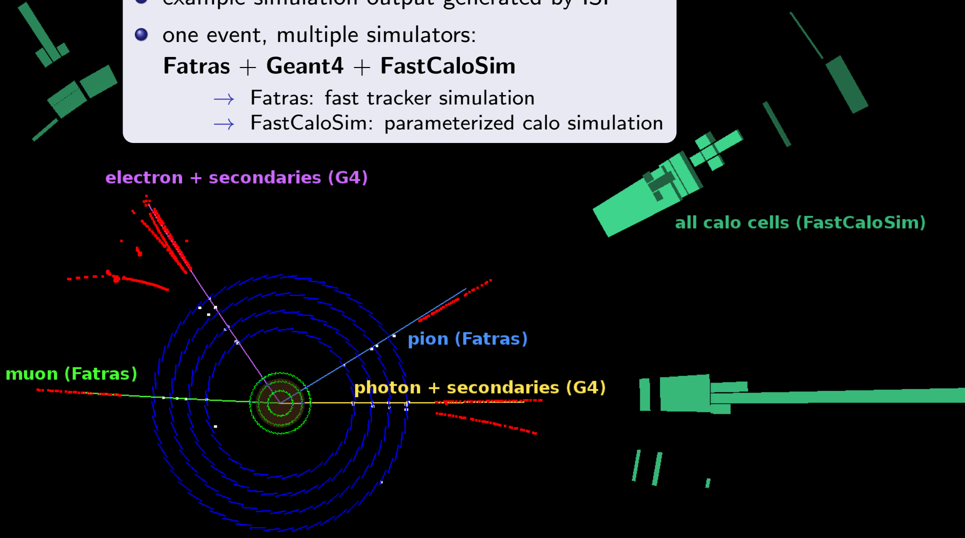
The first Multi-Simulator ISF Run

Event Display Screenshot

- example simulation output generated by ISF
- one event, multiple simulators:

  Fatras + Geant4 + FastCaloSim

  → Fatras: fast tracker simulation
  → FastCaloSim: parameterized calo simulation



ISF Implications on Geant4

ISF and Geant4

- ISF will become *the* future ATLAS simulation framework
- Geant4 is one of many possible simulation engines within the ISF framework
  - shared SD hit collections, common sub-detector boundary definitions
- ATLAS Event is now subdivided into multiple `G4Events`
  - new `G4Event` for each particle sent from ISF to Geant4
  - needed because G4 may not be the only simulation engine and it simplifies tracking of MC truth information
  - most significant change to ATLAS implementation of Geant4 has been in SensitiveDetector classes, which had to be modified to NOT finalize HitCollections at the end of every `G4Event`
- ongoing work for Geant4MT implementation within ISF
  - main objective: get it working in ATLAS framework
  - **sub-event level** parallelism with respect to ATLAS events
  - **event level** parallelism in a `G4Event` perspective
  - speedup studies to compare serial vs parallelized (eg: comparing against Geant4 alternative particle stacking approaches)

  → session on *Parallelization Efforts* this afternoon

Summary

Summary

- two main performance analysis tools: **UserSteppingAction**, **GPerfTools**
- simulation time (ttbar) dominated by: **electrons**, **photons** and **neutrons**
- hot-spot in EM end-cap calorimeter due to **complex geometry** (custom `G4VSolid`)
- 8-15% speedup due to pre-loading of **Intel MathLibrary** (*libimf*)
- simulation **output changes** with Intel MathLibrary, reproducibility on different hardware not clear yet
- $>$ 5% speedup with newer **compiler version 4.6 on SLC6**
- particle `Definition()` calls heavier than expected
- **Integrated Simulation Framework** (see previous slide)

Backup

The Performance Monitoring Tools

UserSteppingAction

- using G4Timers to sum up CPU time spent
  - in different detector volumes
  - per particle type

GPerfTools

- initially developed by Google
- samples the stack trace of a program every couple of milliseconds (default: 10ms)
- minor impact on job runtime
- hot functions will appear more often in the stack trace

Not mentioned in this talk

- commonly used: Valgrind
- first look: GOoDA, VTunes/Amplifier
- to be tested: Intel Performance Counter Monitor