ATLAS Simulation Framework

Fast Detector Simulation in High Energy Physics, DESY Zeuthen

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

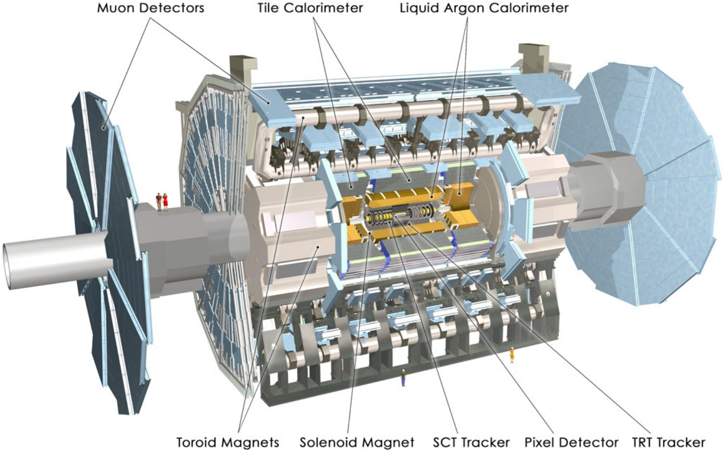
January 16, 2013

Overview

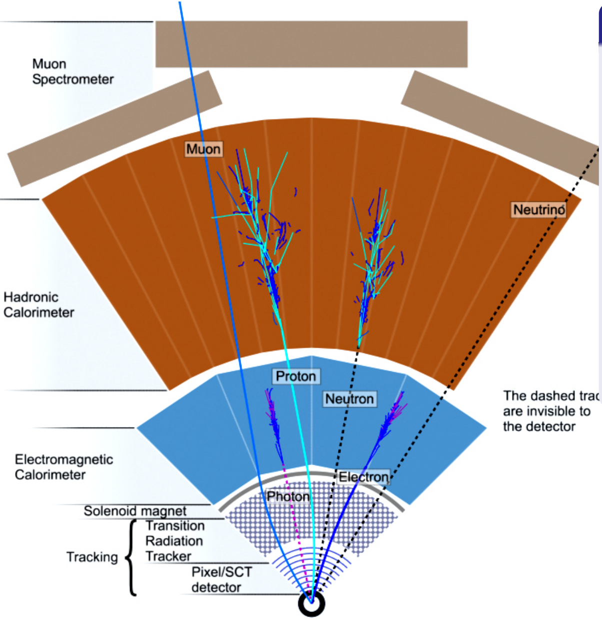
- **The ATLAS Experiment**
  the detector and its components
- **ATLAS Simulation Flow**
  MC Generators, Simulation, Digitization, Reconstruction
- **ATLAS Simulation Engines**
  fast and full simulation
- **The Integrated Simulation Framework**
  combining full and fast simulation in one event

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



ATLAS Detector Components



Detector Layers

- particle identification using different layers of ATLAS detector
- **Pixel and SCT Tracker**: q/p_T (charge over transverse momentum) measurement of charged particles
- **Calorimeter**: energy measurement of electromagnetically and hadronically interacting particles
- **Muon Spectrometer**: additional tracker for muons

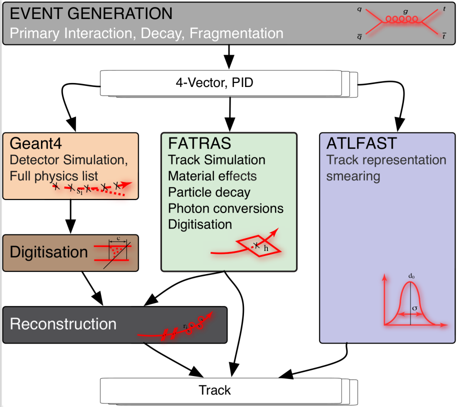
ATLAS Software Framework: Athena

- **Athena Algorithms**
  - a sequence of `AthAlgorithms` can be configured by the user
  - each defined `AthAlgorithm` will be called in every event, in sequence
- **Athena Services and Tools**
  - `AthAlgTools` and `AthServices` used by `AthAlgorithms`
  - usually responsible for computationally heavy duties
- **Transient Storage – Storgate**
  - accessed by the above to read and store various collections, eg. generator particles, SD hits, ...
  - can be configured to write some collections to persistent storage (output files)

ATLAS Simulation Flow

Typical Full Simulation Flow in ATLAS

1. **MC Event Generation**
   Monte Carlo Generators compute final state particles for collision events (Pythia, Herwig,...)
2. **Detector Simulation**
   full or fast simulation engines propagate particles though detector and generate hits on sensitive detector material
3. **Digitization**
   sensitive detector hits are converted into same format as is coming from the real detector
4. **Reconstruction**
   tracker and calorimeter reconstruction algorithms trying to find particle signatures
5. **Analysis**
   using reconstructed objects



Note: some fast simulation engines like Fatras and ALTFAST did skip certain parts of the full simulation flow in the past

ATLAS Simulation Engines

Geant4

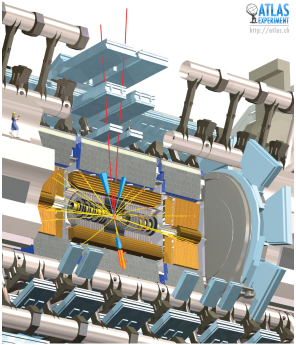
- the top dog, used by many HEP experiments, extensively used in ATLAS
- **high accuracy** simulation of particle-material interactions
- takes a huge amount of CPU resources

FastCaloSim

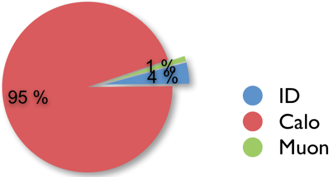
- **parameterized calorimeter** simulation
- much much faster calorimeter simulation compared to Geant4
- parameterized calorimeter punch-through module

Fatras

- **fast tracker simulation**
- based on a **simplified geometry** description and particle-material **interaction model**



Geant4 simulation time per subdetector:



ATLAS Simulation Engines

Geant4

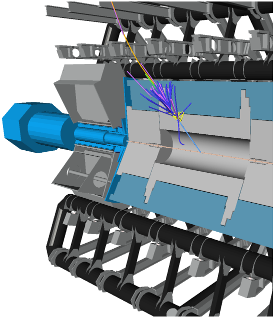
- the top dog, used by many HEP experiments, extensively used in ATLAS
- **high accuracy** simulation of particle-material interactions
- takes a huge amount of CPU resources

FastCaloSim

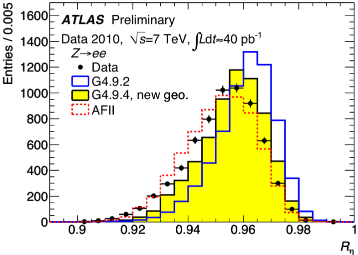
- **parameterized calorimeter** simulation
- much much faster calorimeter simulation compared to Geant4
- parameterized calorimeter punch-through module

Fatras

- **fast tracker simulation**
- based on a **simplified geometry** description and particle-material **interaction model**



FastCaloSim tuning:



ATLAS Simulation Engines

Geant4

- the top dog, used by many HEP experiments, extensively used in ATLAS
- **high accuracy** simulation of particle-material interactions
- takes a huge amount of CPU resources

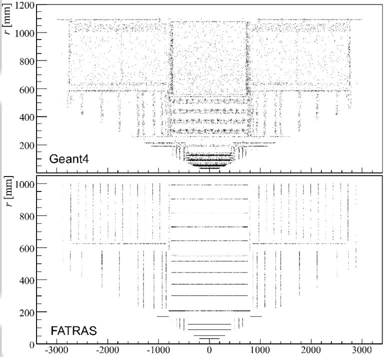
FastCaloSim

- **parameterized calorimeter** simulation
- much much faster calorimeter simulation compared to Geant4
- parameterized calorimeter punch-through module

Fatras

- **fast tracker simulation**
- based on a **simplified geometry** description and particle-material **interaction model**

photon conversion points:



Combining ATLAS Simulation Engines

Geant4

- the top dog, used by many HEP experiments, extensively used in ATLAS
- **high accuracy** simulation of particle-material interactions
- takes a huge amount of **CPU resources**

FastCaloSim

- **parameterized calorimeter** simulation
- much much faster calorimeter simulation compared to Geant4
- parameterized calorimeter punch-through module

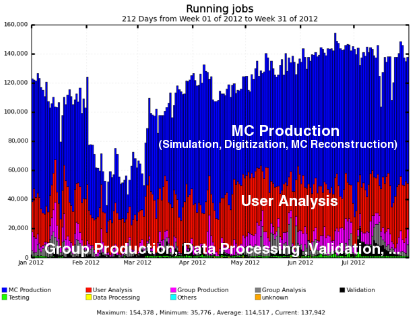
Fatras

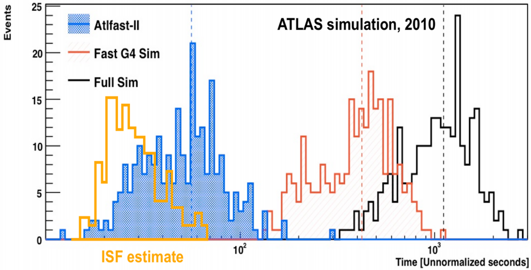
- fast tracker simulation
- based on a **simplified geometry** description and particle-material **interaction model**

ATLFASTII ∼10x speedup (Geant4 + FastCaloSim)

ATLFASTIIF ∼100x speedup (FastCaloSim + Fatras)

Simulation Timing





Monte Carlo Production

- Grid usage dominated by MC production (detector simulation)
- risking physics studies to be limited by MC data

→ need to speed up detector simulation

As a result of speeding up simulation

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

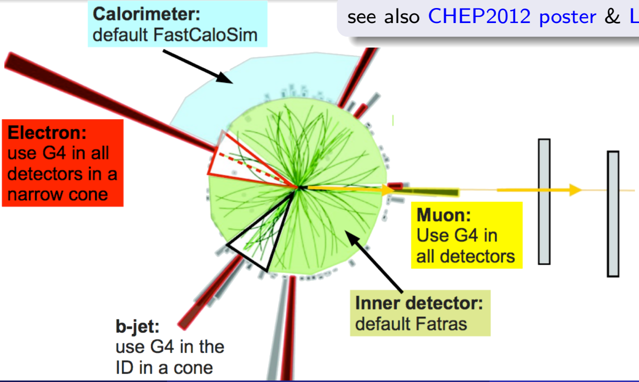
→ partly complex and incompatible setups

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 various parallelization approaches

see also CHEP2012 poster & Lightning Talk



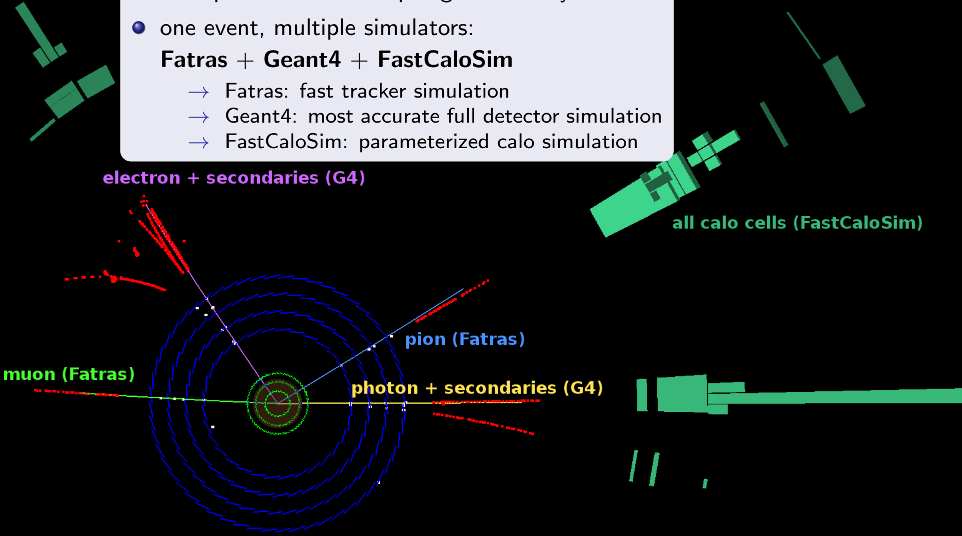
The First Multi-Simulator ISF Run

Event Display Screenshot

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

  Fatras + Geant4 + FastCaloSim

  → Fatras: fast tracker simulation
  → Geant4: most accurate full detector simulation
  → FastCaloSim: parameterized calo simulation



ISF Setups

Simulation Setups

- **Full Geant4** (MC12, ISF)

  G4 in all subdetectors

- **ATLFASTII** (MC12, ISF)

  ID: **Geant4**
  Calo: **Geant4** for muons, **FastCaloSim** for everything else
  MS: **Geant4** (only muons can reach MS, everything else gets *absorbed* by FCS)

- **ATLFASTIIF** (under development, ISF only)

  ID: **Fatras**
  Calo: **Fatras** for muons, **FastCaloSim** for everything else
  (optionally use **parameterized calo punch-through** simulation)
  MS: **Fatras**

- **FastGamma** (under development, ISF only)

  ID: **Fatras** for particles in cones around EvGen photons
  Calo: **FastCaloSim**

more to come...

ISF Simulation Time Measurements

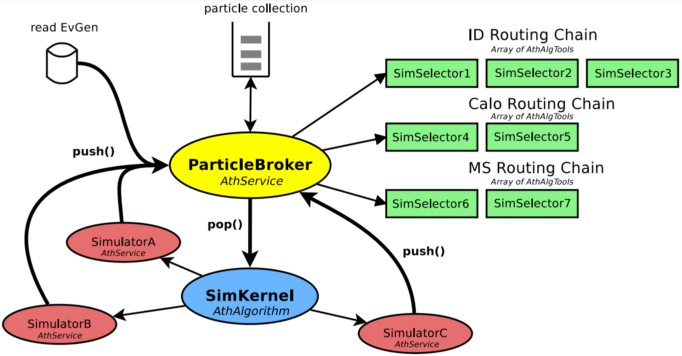
ggF Higgs $\rightarrow \gamma\ \gamma$ sample, no pileup

	ISF_Kernel::execute/evt	speed-up
$\mathcal{O}$ **Full Geant4** measured with 10 events	560 s	1
$\mathcal{O}$ **ATLFASTII** measured with 100 events	25 s	~25
$\mathcal{O}$ **ATLFASTIIF** measured with 1000 events	0.75 s	~750
measured with 1000 events $\mathcal{O}$ **FastGamma**	0.18 s	~3000

ISF Core Design

Main Components

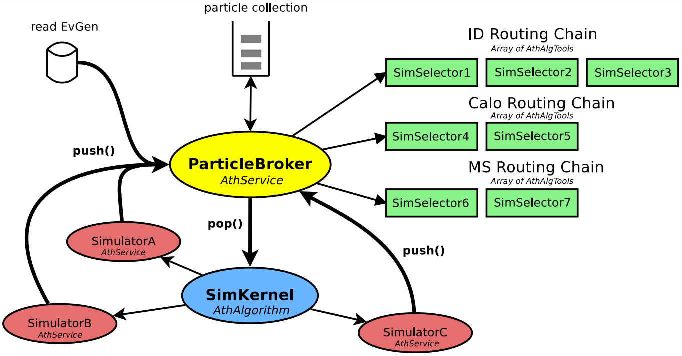
- **SimKernel**: responsible for sending particles to simulators
  - Athena Algorithm with the main particle loop
- **ParticleBroker**: stores particles and determines which simulator should be used for each particle
  - uses *RoutingChain* to determine appropriate simulator
  - separate RoutingChains for each sub-detector



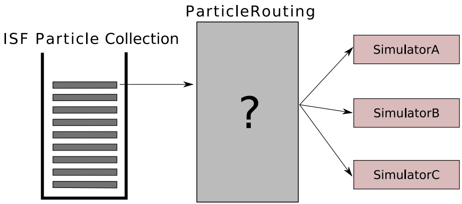
ISF Requirements to Simulators

Simulator Requirements

- **particle handling**: ISF internal particle collection ('*StackManager*')
- **MCTruth**: ISF MC truth manager responsible for truth + barcode recording
- **shared SD hits collections**: various simulators accessing hits for the same sensitive detectors
- **sub-detector boundaries**: simulators give particles back to ISF on boundaries → new routing decision required, due to varying technologies in different sub-detectors



Particle Routing



ISF main purpose is to route particles through sub-detectors and different simulators

- book-keeping necessary

Basic Router Requirements

- static routing rules (`SimulationSelectors`): e.g. using kinematic parameters or particle type
- dynamic `SimulationSelectors` which consider other particles in the event
- simple to configure
- intuitive, no deep knowlegde of the ISF should be needed

Static-only Router Examples

Two Examples

- Particle Type Selector: send all muons to *SimulatorA*
- Kinematic Particle Selector: send all high η particles to *SimulatorB*

Pros

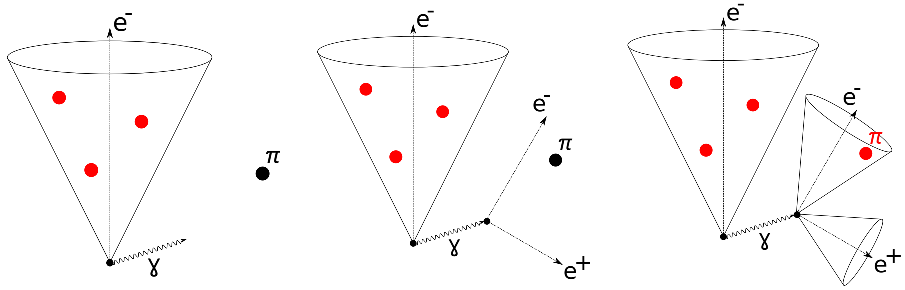
- **order independent**
- fully **consistent**: with the knowledge of particles after event simulation, the exact same decisions would have been made
- **intuitive** for the user
- **single pass** (each particle only simulated once)

Keep in Mind

- selector decisions may contradict each other

$\rightarrow$ selectors need to be defined in a priority list

Dynamic Router Example: Cone Selector



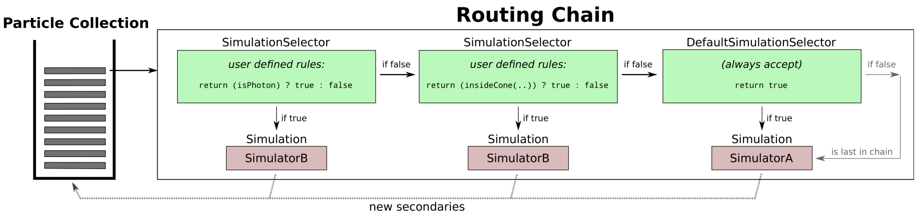
Dynamic Cone Selector

- the dynamic selector registers a cone for each new electron in the event
- all particles inside a cone are to be simulated in a certain simulation

Attention!

- decision on pion depends on the **simulation order**
- if π simulated before conversion: **inconsistent selector decision**

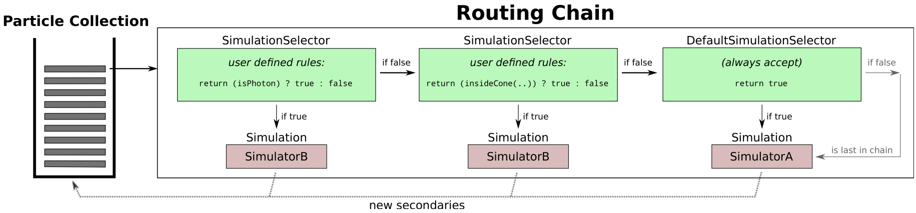
ISF Routing Chain: Functionality



How the Routing Chain works

1. a particle is taken from the particle collection
2. the `SimulationSelectors` are asked in a specific order whether they would select the particle
3. in case a `SimulationSelector` does not take the particle, it will be handed over to the next in the chain
4. the first `SimulationSelector` which returns `true` decides that the particle will be sent to the simulation attached to this `SimulationSelector`

ISF Routing Chain: Pros and Cons



Dynamic SimulationSelectors

- dynamic SimulationSelector rules will be updated only by the EvGen particles

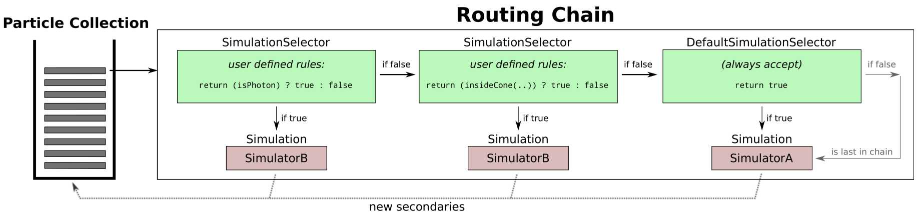
Pros

- **single pass**: each particle simulated only once
- **order independent**
- **intuitive** in its functionality

Cons

- does not support fully dynamic SimulationSelectors, eg. the cone example from before

The User and the Routing Chain



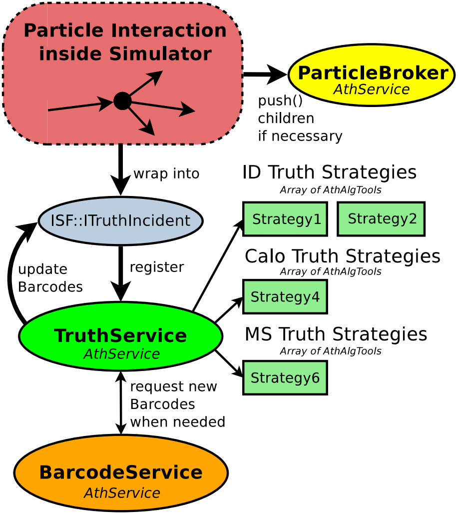
How the user interacts with the Routing Chain

- user implements `SimulationSelector(s)` which make yes/no decisions
- user defines one Simulator for each `SimulationSelector`
- user specifies the order in which the `SimulationSelector` will be used:

```
ISFRouter.SimSelectorID   = [ Selector1, Selector2, DefaultIDSelector ]
ISFRouter.SimSelectorCalo = [ Selector3, DefaultCaloSelector ]
ISFRouter.SimSelectorMS   = [ Selector4, Selector5, DefaultMSSelector ]
```

- no deep insight in ISF functionality required by the user

MC Truth and Barcodes in ISF I



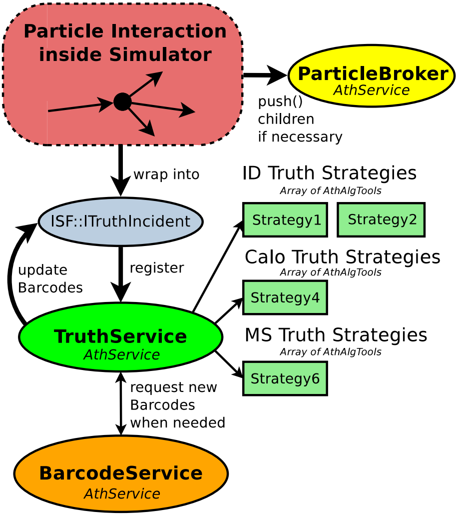
ITruthIncident Interface

- simulator independent interface
- basically a wrapper to allow the TruthService and TruthStrategies to access primary and secondary particle properties as well as the interaction type
- used by TruthService to pass updated particle barcodes back into the simulation

Simulator Requirements

- directly write to **shared hit collections** on StoreGate, eg: `PixelHits`, `LArHitFCAL`, `MDT_Hits`, ...
- **return all particles on sub-detector boundaries** to ISF ParticleBroker
- **implement ITruthIncident** for each simulator specific particle implementation, eg: `Geant4TruthIncident`, `ISFTruthIncident`

MC Truth and Barcodes in ISF II



TruthService

- one array of TruthStrategies per sub-detector
- TruthStrategies make a boolen decision based on information they can get from `ITruthIncident`: primary/secondary particle energy, type, number, interaction process, ...
- TruthIncident will be written to MCTruth only if at least one TruthStrategy in the corresponding array returned `true`

BarcodeService

- interchangeable Athena service
- no ISF dependency
- generates particle and vertex barcodes
- uses parent particle barcode and interaction type to generate secondary barcodes and update primary particle barcode
- current implementation reproduces MC12 behaviour, but ISF allows for way more:
- eg allows for *shared child particle barcodes* in case the truth incident is not recorded

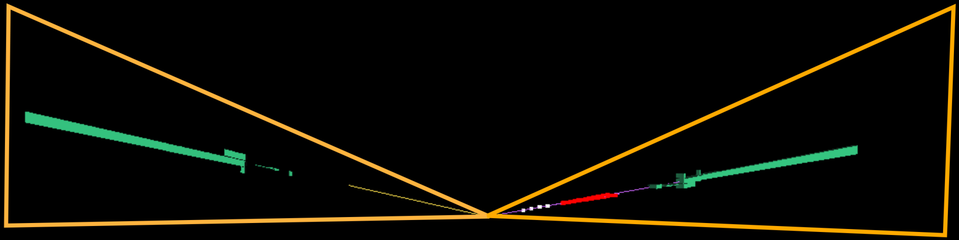
ISF FastGamma Simulation for $H \rightarrow \gamma\gamma$

Why ISF?

- need high statistics
- need accurate description of photons in ID and Calo

$\rightarrow$ simulate only **parts of the event** with ISF

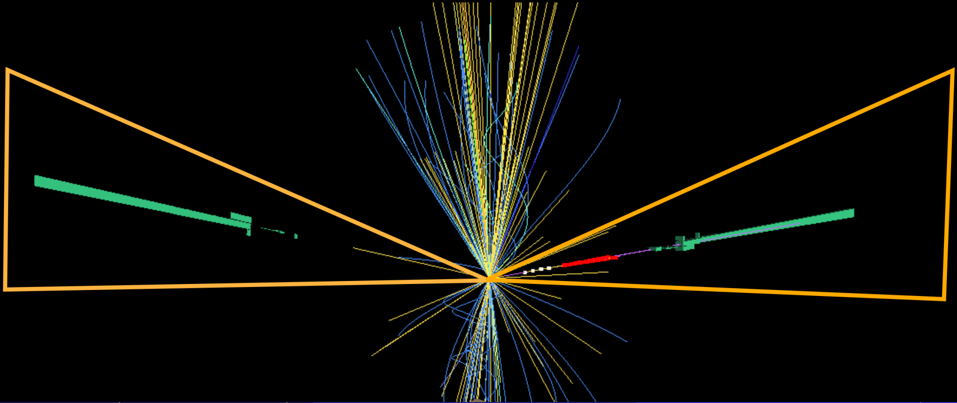
$\rightarrow$ region of interest is in cones around EvGen signal photons



ISF FastGamma Simulation: What you gain

What you gain

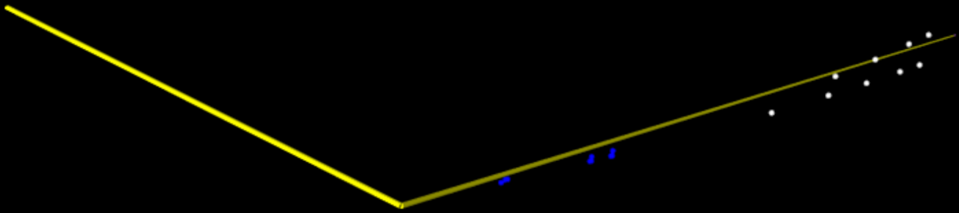
- spending **less (no) simulation time** on SimHits no one cares about
- **smaller** simulation output files
- **faster** Digitization
- **faster** Reconstruction



ISF FastGamma Simulation: inside cones we are..?

Unconverted photons

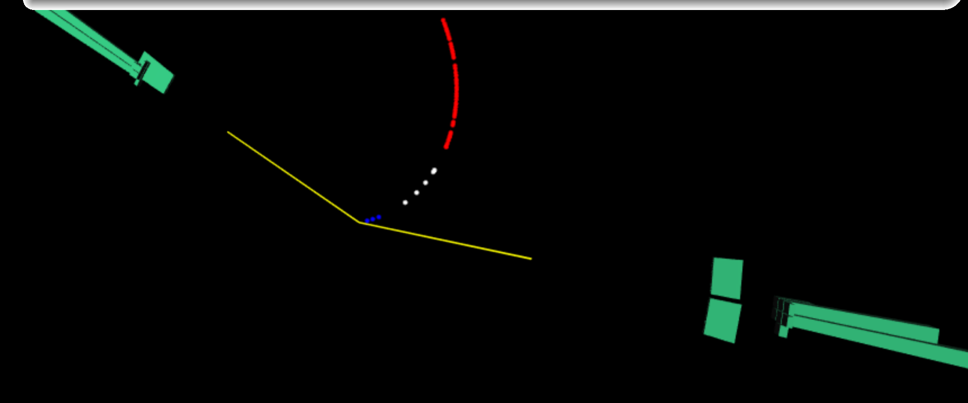
- two photons not converted
- some charged particles inside cones creating hits



ISF FastGamma Simulation: inside cones we are..?

Bending out particles I

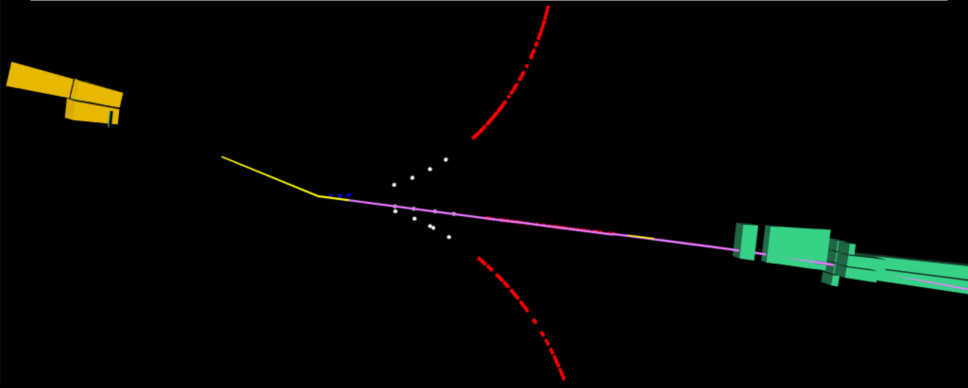
- two photons not converted
- charged particle initially inside cone bends out
- one electron from pair conversion bends out from the cone
  - → some CPU time spent on information that is not needed



ISF FastGamma Simulation: inside cones we are..?

Bending out particles II

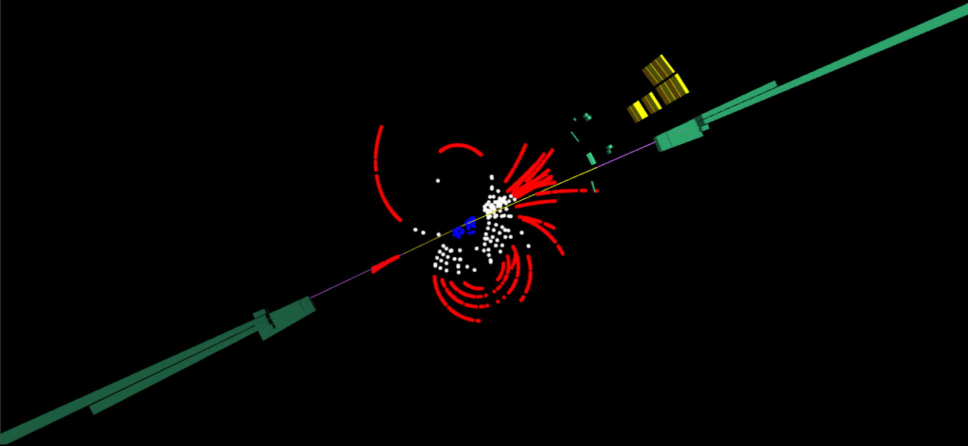
- two photons, one undergoes pair-conversion
- charged particle initially inside cone bends out
  - → some CPU time spent on information that is not needed



ISF FastGamma Simulation: inside cones we are..?

Particles bending out massively

- a lot of initial particles bending out of cones
  → quite a lot CPU time spent on information that is not needed



Summary and Outlook

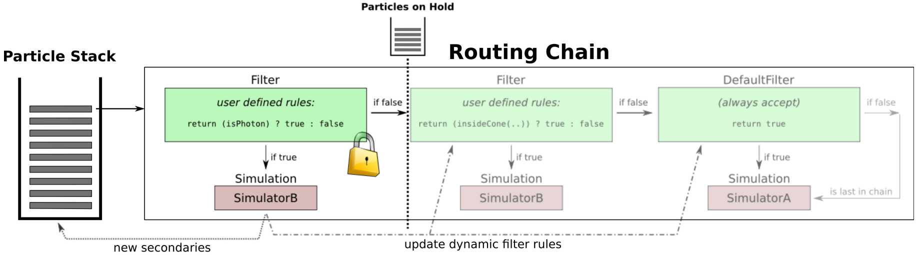
Summary

- ISF able to **reproduce all current detector simulation setups**
- ISF allows to combine simulation engines on a **particle level**
- **balance between accuracy and speed** on a particle level
- ISF can simulate only **parts of the event** (eg. signal only)
  - consequently saves diskspace, speeds up digitization and reconstruction
- working with physics groups to get **first usecases** into production
  - using ISF flexibility to mix Geant4, Fatras, FastCaloSim, Punch-Through
- speed-up by **factor 3000 measured** (relative to full Geant4)

Outlook

- developments ongoing for validation of fast simulation setup (with future production use)
- studies on various parallelization approaches: multithreading, vectorization, ...
- study behaviour of dynamic routing rules
- ISF will become *the* future ATLAS detector simulation framework

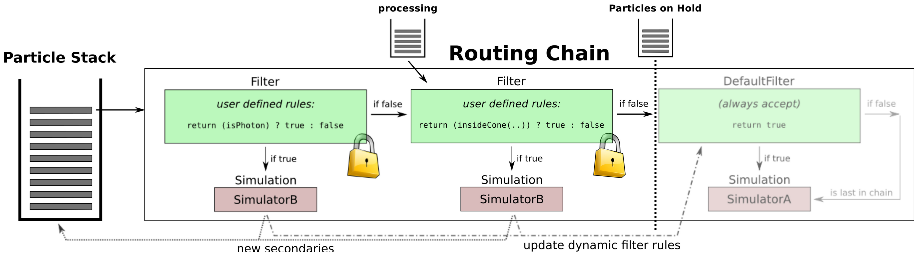
Backup



How the Routing Chain works

1. first filter will only be updated during read-in, afterwards: locked
2. simulate all particles selected by the first filter (*update all dynamic filters*)
3. simulate all child particles which are selected by the first filter until no more child particles of any generation are selected (*update filters*)
4. second filter will be locked
5. simulate all particles selected by the second filter (*update all dyn. filters down the chain*)
6. simulate all particles (and child particles) selected by the first two filters (*update filters down the chain*)

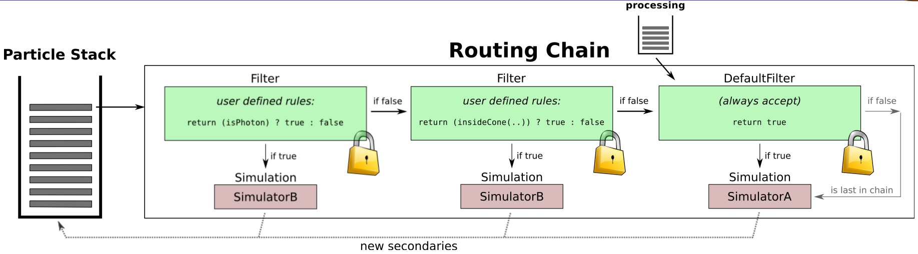
Approach 3: **Routing Chain with Incremental Locks**



How the Routing Chain works

1. first filter will only be updated during read-in, afterwards: locked
2. simulate all particles selected by the first filter (*update all dynamic filters*)
3. simulate all child particles which are selected by the first filter until no more child particles of any generation are selected (*update filters*)
4. second filter will be locked
5. simulate all particles selected by the second filter (*update all dyn. filters down the chain*)
6. simulate all particles (and child particles) selected by the first two filters (*update filters down the chain*)

Approach 3: **Routing Chain with Incremental Locks**

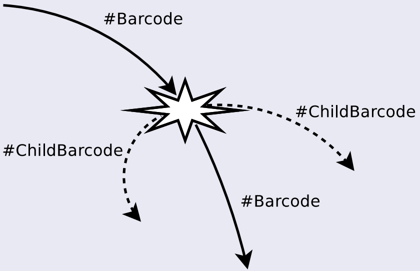


How the Routing Chain works

1. first filter will only be updated during read-in, afterwards: locked
2. simulate all particles selected by the first filter (*update all dynamic filters*)
3. simulate all child particles which are selected by the first filter until no more child particles of any generation are selected (*update filters*)
4. second filter will be locked
5. simulate all particles selected by the second filter (*update all dyn. filters down the chain*)
6. simulate all particles (and child particles) selected by the first two filters (*update filters down the chain*)

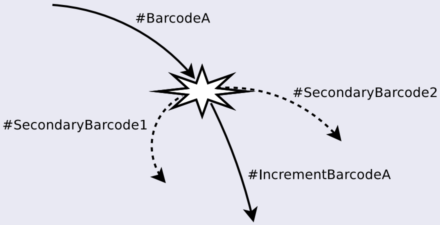
Particle Barcode Handling in ISF

Truth Incident not stored



- parent particle keeps same barcode
- all child particles will be assigned the same barcode
  - allows eg. SimHit to parent particle association
- nothing will be added to the HepMC TruthEvent on StoreGate

Truth Incident stored



- update parent barcode after vertex
- each child particle gets a unique barcode
- adding all particles to the HepMC TruthEvent on StoreGate

in both cases, the parent particle barcode and an interaction process identifier are available to generate the corresponding new child barcodes or updated barcodes

Calorimeter MC Truth and Barcode Service

Status in MC12

- TrackRecords at CaloEntry, MuonEntry and MuonExit surfaces
- in favour of CPU time and disk space (**big impact!**), much fewer truth strategies inside calo compared to ID
- only **muon Bremsstrahlung** in HepMC TruthEvent: $E_{kin,\mu} > 500\text{MeV}$ and $E_{kin,\gamma} > 100\text{MeV}$

Possibilities in ISF

- TrackRecords still there
- CPU time and disk space restrictions still apply
- **Flexible Barcode Service:**
  - possible to encode information about parent particle in all child particles
  - eg. would allow to to trace back particles at MuonEntry to initial particle CaloEntry, by using the barcode only
  - ISF-independent `AthService` which could be used for barcode encoding and decoding