ATLAS Fast Framework

2nd Fast Monte Carlo Workshop in HEP, DESY Zeuthen

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

January 15, 2014

Overview

- $\mathcal{O}$ **Framework Vision**

  one flexible and common framework

- $\mathcal{O}$ **Framework Requirements**

  reproducibility, predictability

- $\mathcal{O}$ **Simulator Requirements**

  many simulators talking to one common framework

- $\mathcal{O}$ **Challenges**

  and lessons learned

ATLAS Simulation Engines (recap)

Geant4 + Frozen Showers

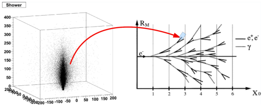
- the top dog, used by many HEP experiments, extensively used in ATLAS
- **high accuracy** simulation of particle-material interactions
- takes a huge amount of CPU resources

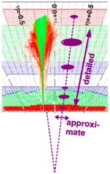
FastCaloSim

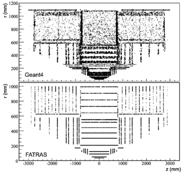
- **parameterized calorimeter** simulation
- much much faster calorimeter simulation compared to Geant4
- parameterized calorimeter punch-through module

Fatras

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







Consequence

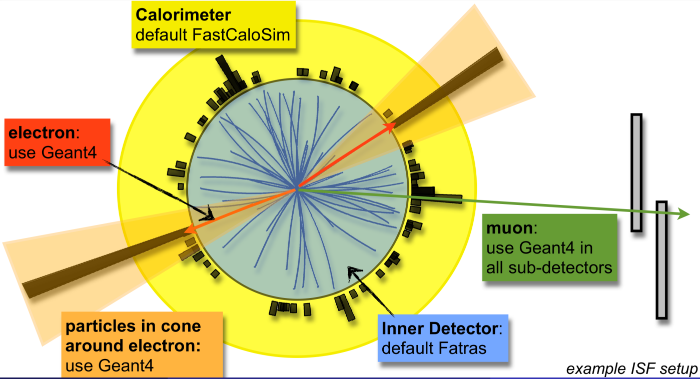
- increasing number of ATLAS detector simulation engines

$\rightarrow$ 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 to **select simulation engine on a particle level**
  - speedup expected
  - modularity allows for various parallelization approaches



example ISF setup

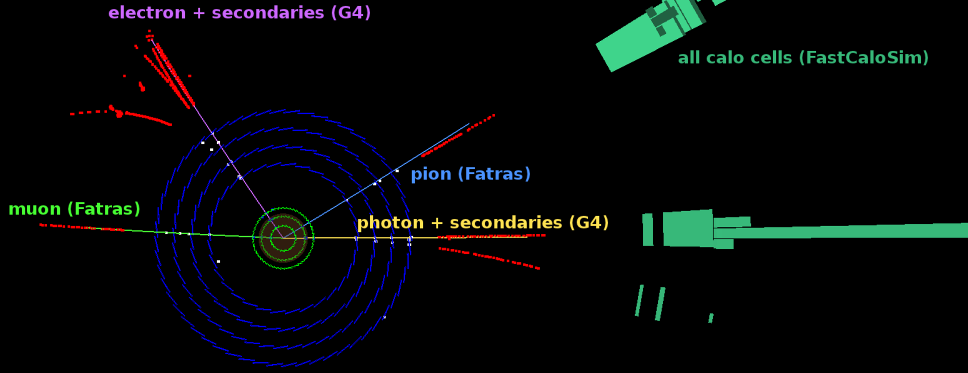
The First Multi-Simulator ISF Run

Event Display Screenshot

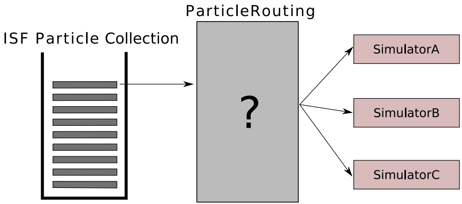
- example simulation output generated by ISF
- one event, multiple simulators:

  Fatras + Geant4 + FastCaloSim

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



Particle Routing



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

- book-keeping necessary

Basic Router Requirements

- static routing rules: e.g. using kinematic parameters or particle type
- dynamic routing which considers 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

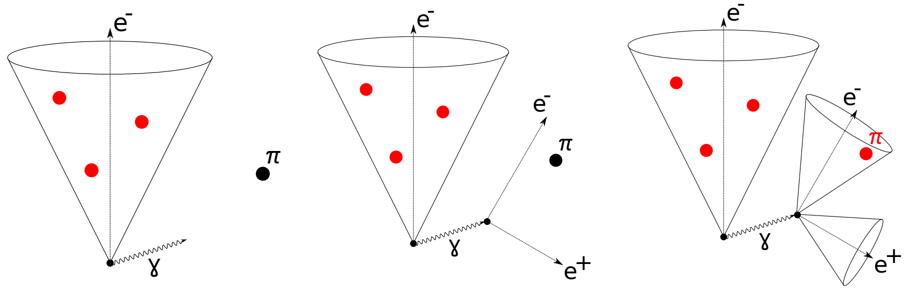
- **order independent**
- fully **consistent**: with the knowledge of particles after event simulation, the exact same decisions would have been made
- **intuitive** for the user
- **single pass** (each particle only simulated once)

Keep in Mind

- selector decisions may contradict each other

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

Dynamic Router Example: Cone Selector



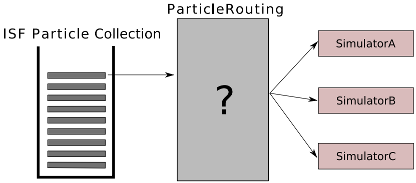
Dynamic Cone Selector

- 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 Router Requirements



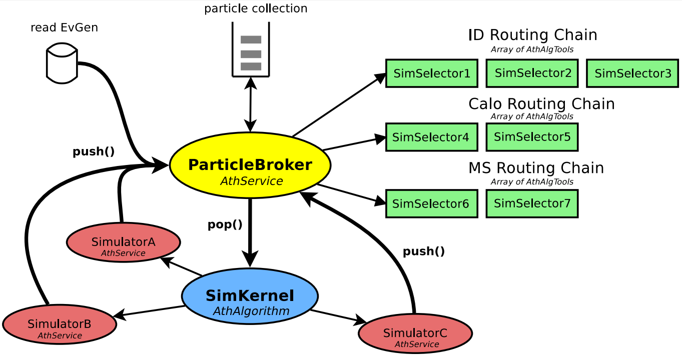
Router Requirements

- **reproducible** results
- **order independence**: important for concurrent processing
- **arbitrary (dynamic) routing rules** which may consider other particles in the event
- **single pass**: no re-simulation of the same particle (due to changing filter decisions)
- **event consistency**: with the knowledge of particles after event simulation, the same router decisions would have been made
- **intuitive** ISF routing logic and configuration

ISF Core Design

Main Components

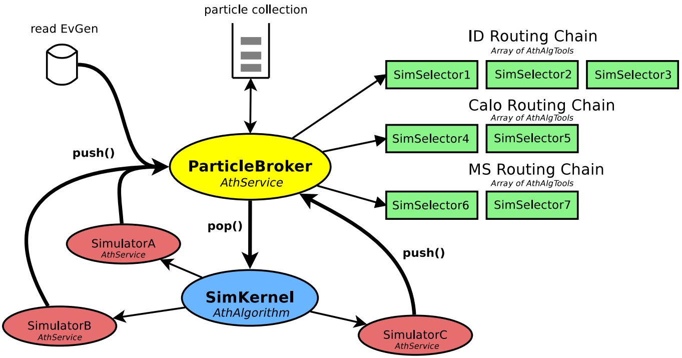
- **SimKernel**: responsible for sending particles to simulators
  - Athena Algorithm with the main particle loop
- **ParticleBroker**: stores particles and determines which simulator should be used for each particle
  - uses *RoutingChain* to determine appropriate simulator
  - separate RoutingChains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



ISF Requirements to Simulators

Simulator Requirements

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



ISF Setups in use / under validation

Classical setups

→ static routing rules only

- full Geant4 for all sub-detectors
- ATLFASTII: G4 for InDet and muons, FastCaloSim for calorimeter
- ATLFASTIIF: Fatras for InDet and muons, FastCaloSim for calorimeter

Dynamic particle routing ($Z \to ee$, $H \to 4\ell$)

→ dynamic, only at generator event-level

- **signal decay products** in Geant4, rest in Fatras/FastCaloSim
- **cones around signal** decay products in Geant4, rest in Fatras/FastCaloSim
  - inside/outside cones checked at generator event-level
  - optionally re-check if particle still inside cone at InnerDet/Calo boundary

Challenges

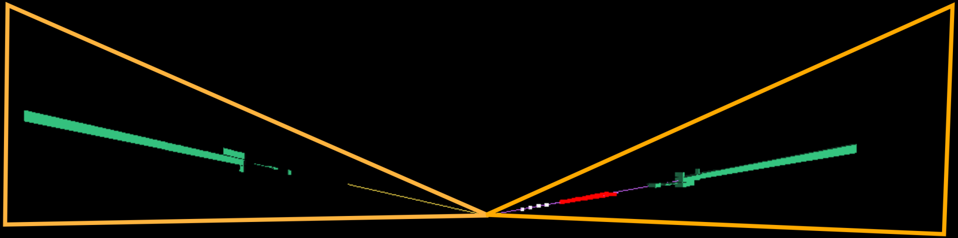
Partial Event Simulation for $H \to \gamma\gamma$

Why ISF?

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

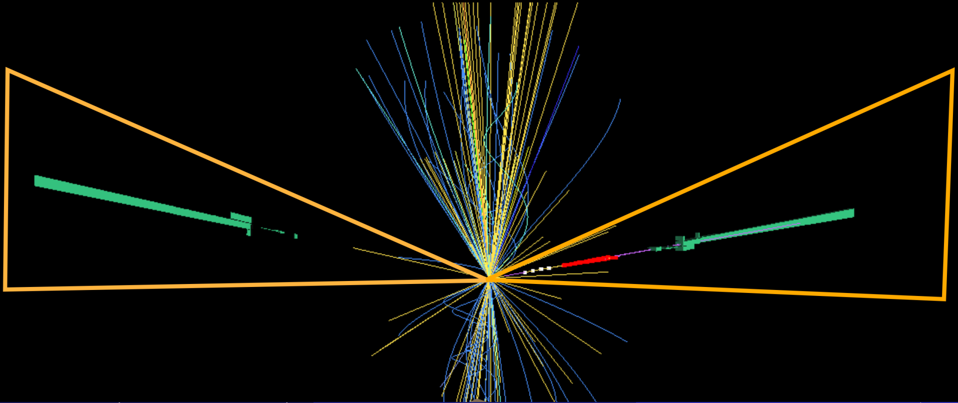
$\rightarrow$ simulate only **parts of the event** with ISF

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



What you gain

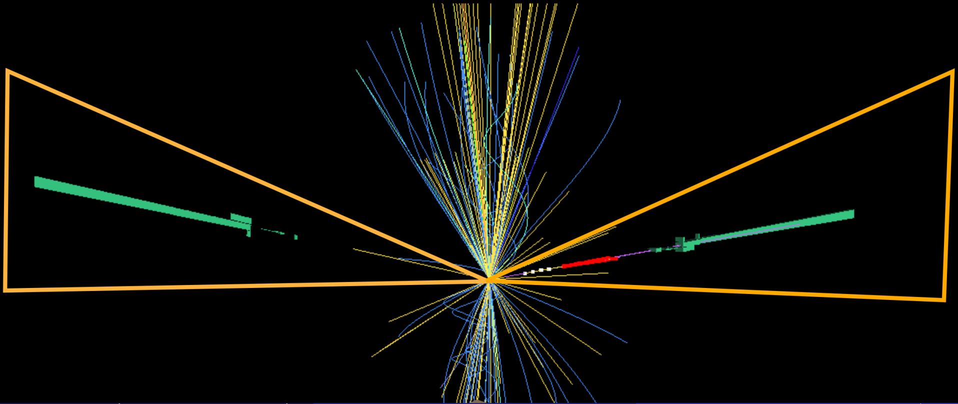
- spending **less (no) simulation time** on SimHits no one cares about
- **smaller** simulation output files
- **faster** Digitization
- **faster** Reconstruction



Where you lose

- global event variables gone ($\sum E_T$, missing E_T)

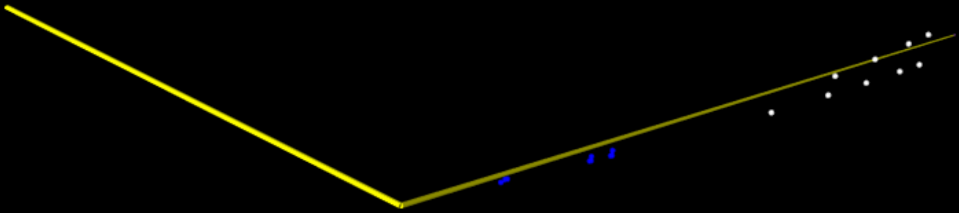
→ analysis side covered in Andi's talk



Partial Event Simulation: inside cones we are..?

Unconverted photons

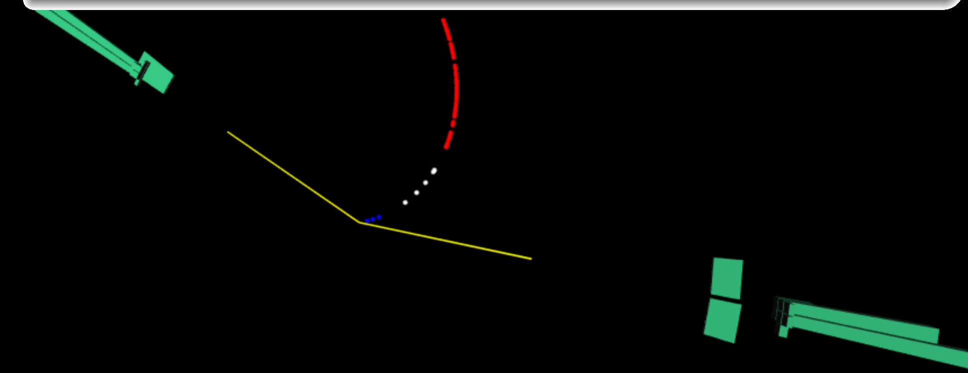
- two photons not converted
- some charged particles inside cones creating hits



Partial Event Simulation: inside cones we are..?

Bending out particles I

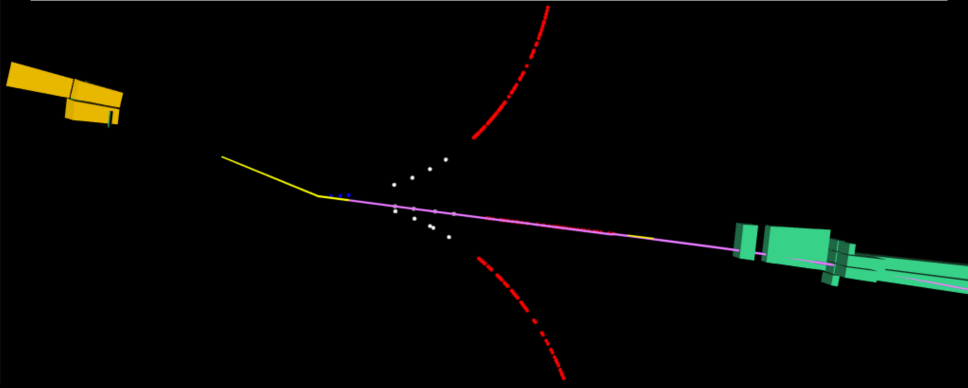
- two photons not converted
- charged particle initially inside cone bends out
- one electron from pair conversion bends out from the cone
  - $\rightarrow$ some CPU time spent on information that is not needed



Partial Event Simulation: inside cones we are..?

Bending out particles II

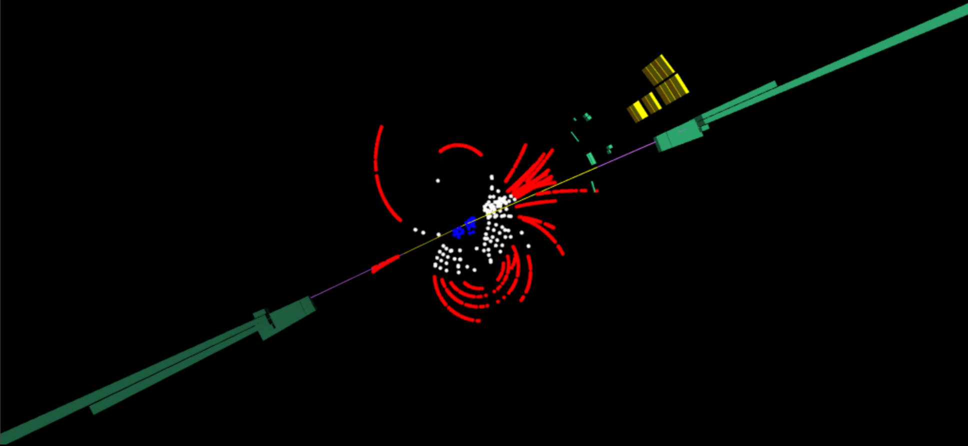
- two photons, one undergoes pair-conversion
- charged particle initially inside cone bends out
  - → some CPU time spent on information that is not needed



Partial Event 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

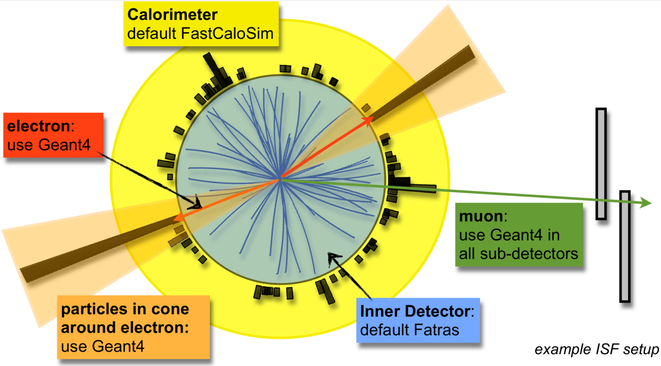


Bookkeeping

- a number of simulators now creating hits for the same detectors
- simulators may need to know a particle's previous simulator type
- bookkeeping neccessary → which particles and hits created by which simulator

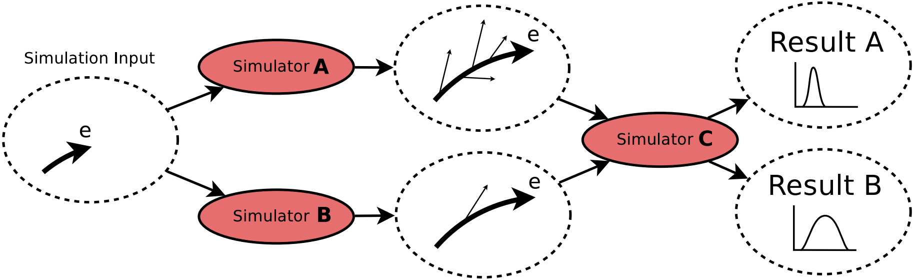
→ encode simulator identifier into truth particle barcode

→ barcode is stored in MC-truth representation and sensitive detector hits



example ISF setup

Simulator Consistency



Simulator Consistency

- output of *simulator A* and *simulator B* is input for *simulator C*
- simulators A and B have different tunings, energy cuts, ...

$\rightarrow$ simulation output for exactly the same generator particle **will be different** between *simulator A* and *simulator B*

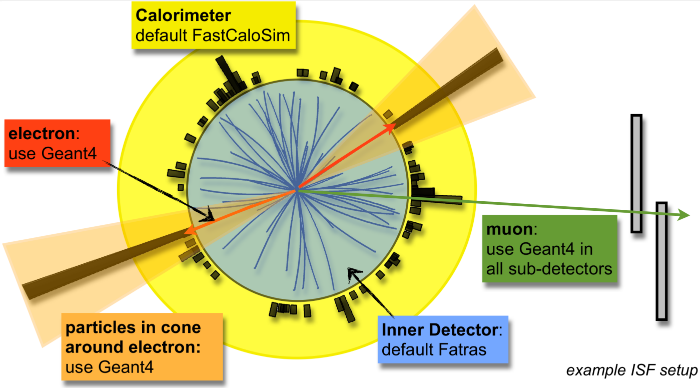
$\Rightarrow$ consequently *simulator C* output will be different

$\rightarrow$ *simulator C* may need to take into account the originating simulator of a particle

Simulator Consistency – Scenario

Example Scenario

- signal electrons simulated with Geant4 (produces many low-E secondaries)
- rest of ID simulated with Fatras (much higher secondary threshold)
- FastCaloSim takes any of the secondaries for calo simulation
  - $\rightarrow$ does not distinguish between Fatras or G4 secondaries
  - $\rightarrow$ will be **over/under-estimating the energy** in the calorimeter

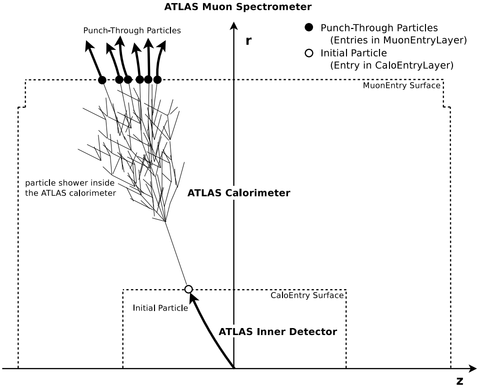


example ISF setup

Consistency – Parametrized Punch-Through

Parametrized Simulation

- fast punch-through simulation parametrized with Geant4 input/output
- needs to be parametrized together with FastCaloSim (highly correlated)



Summary and Outlook

Summary

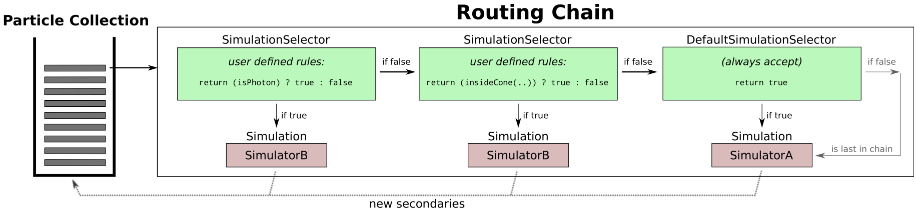
- ISF is becoming *the* future ATLAS detector simulation framework
- ISF able to **reproduce all current detector simulation setups**
- ISF allows to combine simulation engines on a **particle level**
- **balance between accuracy and speed** on a particle level
- ISF can simulate only **parts of the event** (eg. signal only)
  - consequently saves diskspace, speeds up digitization and reconstruction
- **validating first usecases** of mixed full/fast simulation setups
  - do we need to correlate simulators?

Outlook

- studies on various parallelization approaches: multithreading, vectorization, ...
- more ISF usecases to come

Backup

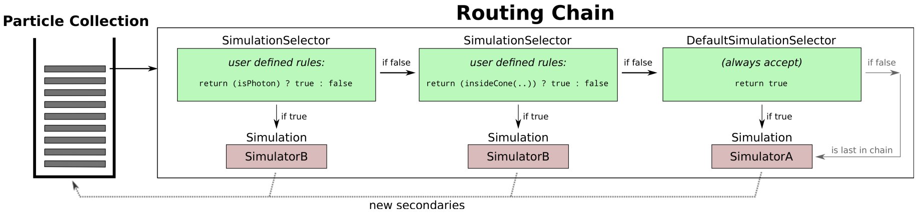
ISF Routing Chain: Functionality



How the Routing Chain works

1. a particle is taken from the particle collection
2. `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

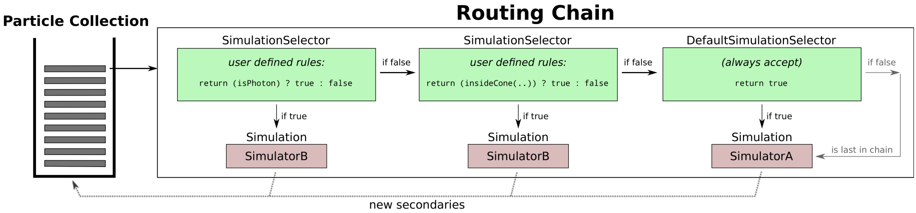
Pros

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

Cons

- does not support fully dynamic SimulationSelectors, eg. the cones around every electron in the simulated event

The User and the Routing Chain



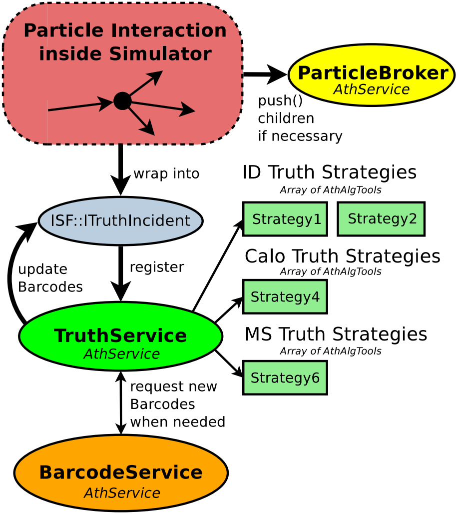
How the user interacts with the Routing Chain

- 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



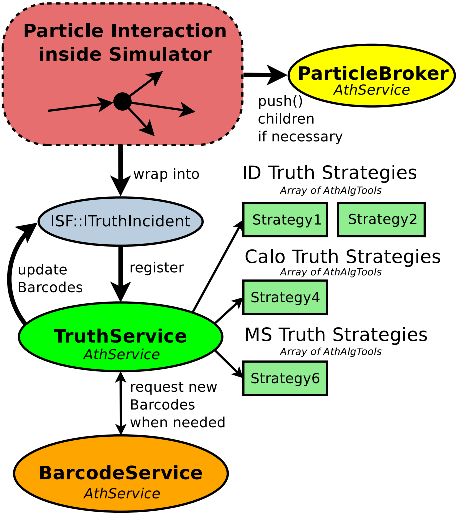
ITruthIncident Interface

- 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



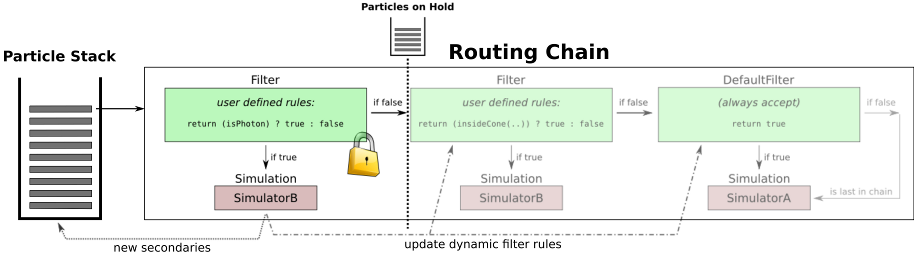
TruthService

- 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

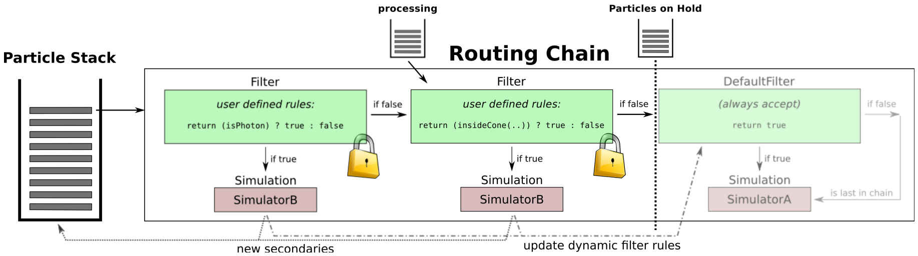
Approach 3: **Routing Chain with Incremental Locks**



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*)

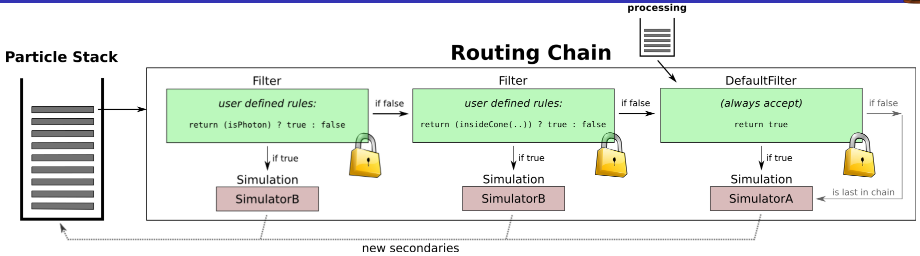
Approach 3: **Routing Chain with Incremental Locks**



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*)

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

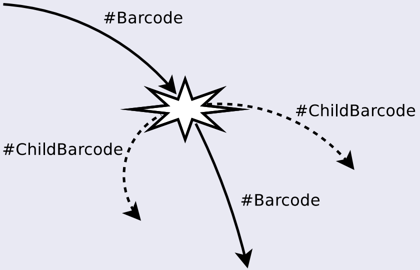


How the Routing Chain works

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*)

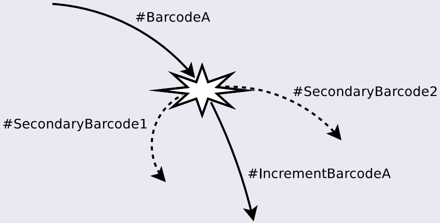
Particle Barcode Handling in ISF

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