



ALICE Simulation for Run3

“Towards Faster Simulation”: **Ideas** and Developments

Sandro Wenzel (CERN) for WP12



Run3 Simulation Software

1. Detector Setup

(Geometry, Material,
Magnetic Field)

2. Modeling Hits

(Model interaction of tracks inside
actual sensitive detector regions)

3. Description of readout processes

(How energy deposit is translated
into hardware signal)

O2 Common Simulation Layer

Data format, Magnetic Field, Management (Run,
Detector, etc), Geometry tools, Passive
structures, Configuration,

FairRoot

ALFA

Major rewrite of simulation within the new
common online/offline project (O2)

Major Novelties:

- **Continuous readout of detector**
- Detector data is aggregated into **"timeframes"** which are further processed in a novel **data-flow computing framework** based on collaborating devices.

Run3 Simulation Software

1. Detector Setup

(Geometry, Material, Magnetic Field)

2. Modeling Hits

(Model interaction of tracks inside actual sensitive detector regions)

3. Description of readout processes

(How energy deposit is translated into hardware signal)

O2 Common Simulation Layer

Data format, Magnetic Field, Management (Run, Detector, etc), Geometry tools, Passive structures, Configuration,

FairRoot

ALFA

Major rewrite of simulation within the new **common online/offline project (O2)**

Major Novelties:

- **Continuous readout of detector**
- Detector data is aggregated into **“timeframes”** which are further processed in a novel **data-flow computing framework** based on collaborating devices.

Based on common (simulation) software building blocks and a partnership with many other experiments

[See talk by T. Kollegger](#)

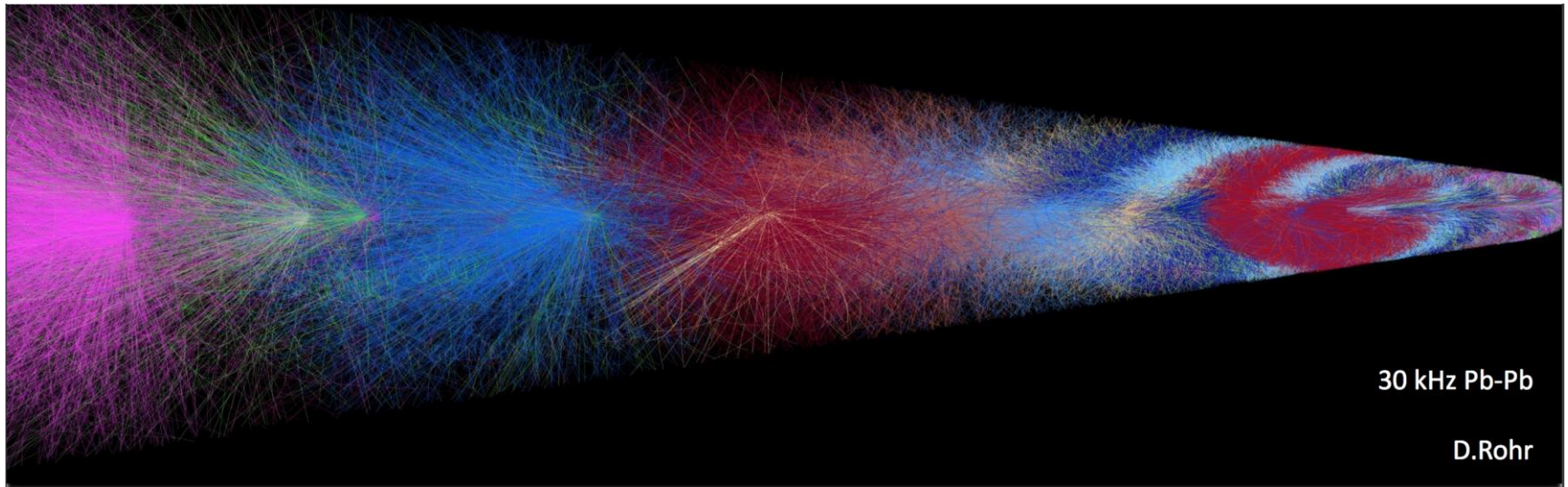
Actual simulation (“hits”) will stay **Virtual Monte Carlo (VMC)** based and be quite similar to Run2

- Able to use different engines (G4, G3, Fluka) based on one geometry description (TGeo) and abstract scoring code

Main “Challenge” Run3/Run4: Continuous Readout

Overlapping events in the TPC

Biggest conceptual changes required in digitization and reconstruction



Scale of the simulation problem : Focus on hit creation

The occupancy of Pb-Pb events can be quite large ($O(100k)$ primaries)

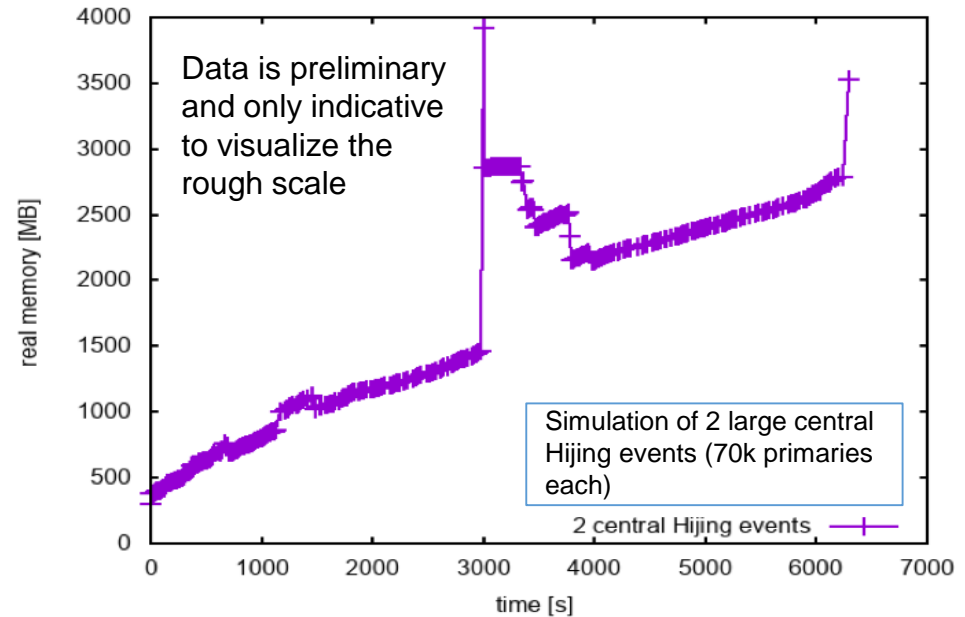
- **Substantial CPU times (+ pressure on memory)** if full event is treated as the most basic unit
- Worked hard to achieve Run3 simulation on any laptop ... but

1. **Processing time** for one event can be $O(h)$
2. **Memory** may be on the order of **GB(s)** per event

Not ideal for scheduling jobs when given $O(h)$ slots

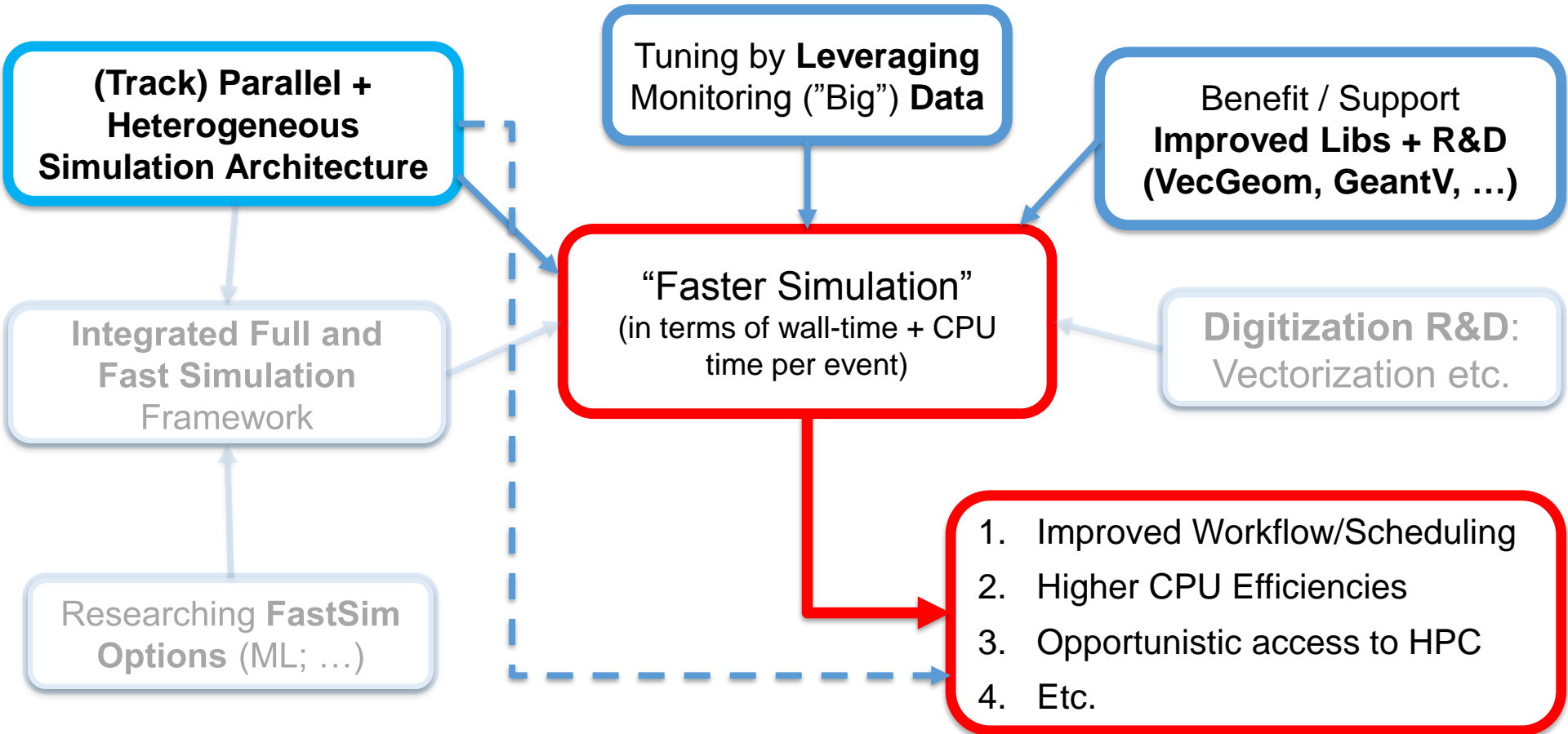
1. Difficult to access to opportunistic resources on HPC machines
2. On grid usually not able to optimally fill 24h allocations

Setting the rough scale (taking similar cuts + settings used in Run2)

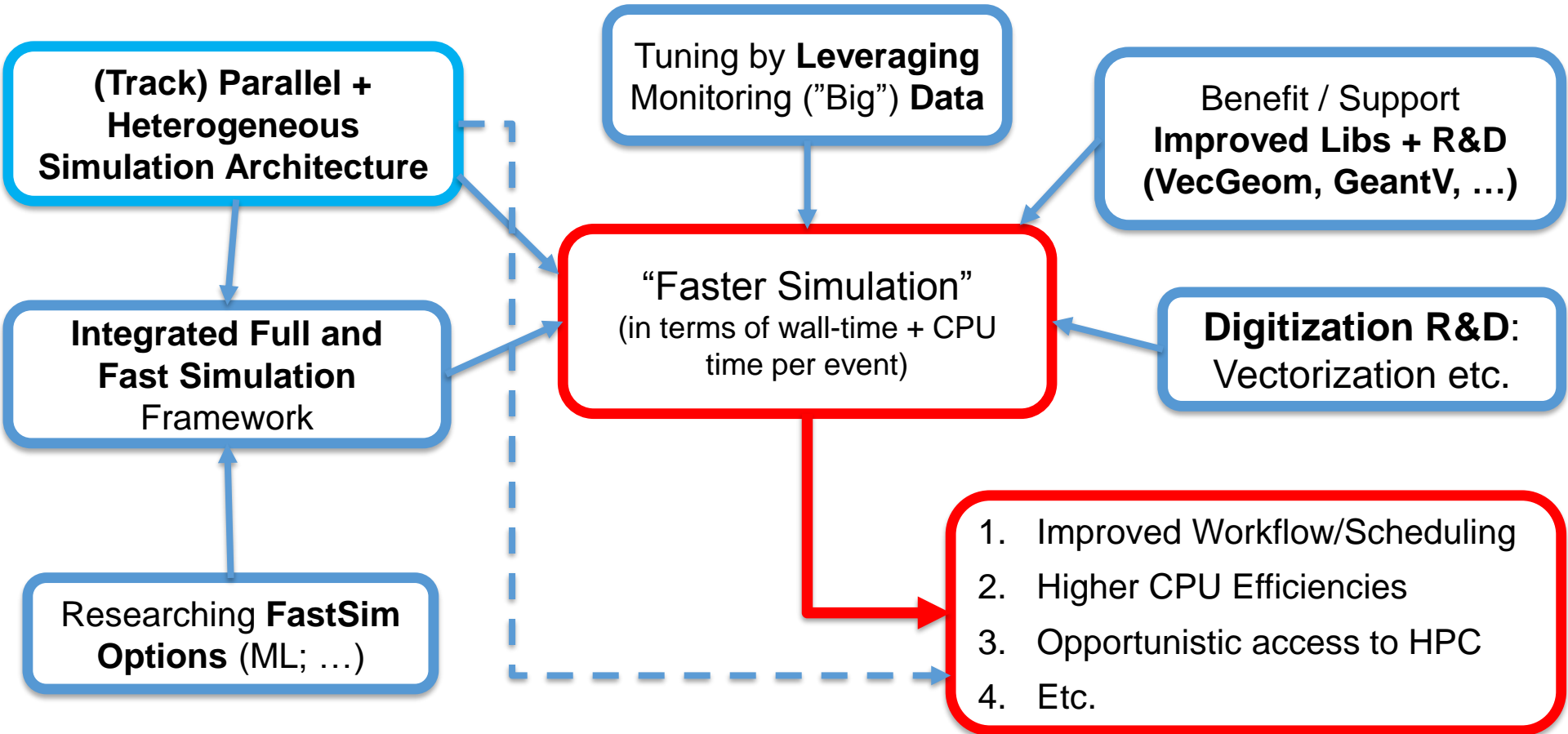


Starting point for activities mentioned in this talk

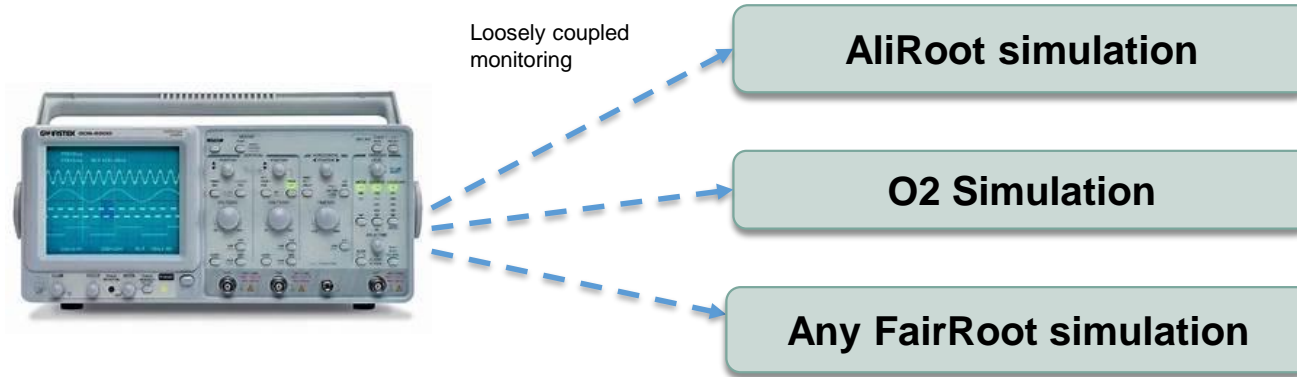
Faster Simulation Ideas: The big picture



Faster Simulation Ideas: The big picture



Leveraging Monitoring “Data”: Simulation tuning



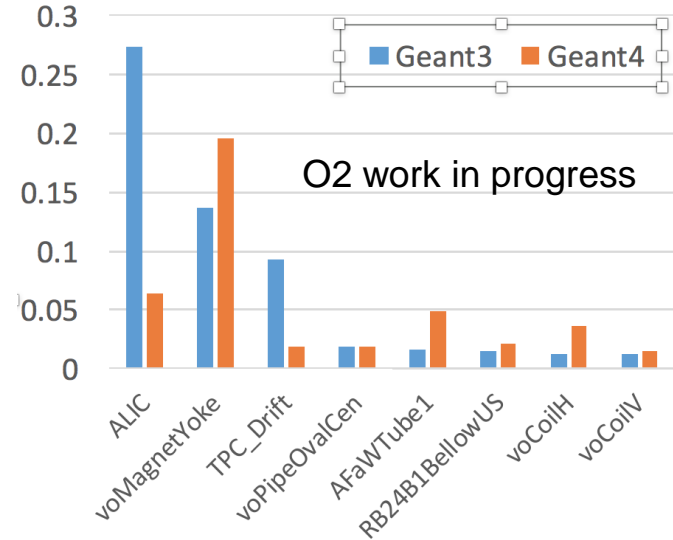
- Invested work and develop(ed) loosely coupled + “platform independent” monitoring mechanism to be able to inspect the simulation at step level
- **Key to**
 - Understand details of simulation
 - Compare AliRoot (Run2) to O2 (Run3)
 - Compare different simulation engines (Geant4, Geant3)
 - **Simulation tuning and optimization**

First insights from monitoring

- `LD_PRELOAD=libMCStepLogger.so o2sim`
- Allows us obtain detailed costs of various detector components on simulation CPU budget
- **See where to focus with tuning/optimization**

Preliminary non-trivial observations (also Run2):

- Found frequent queries to the magnetic field in detector volumes which are fully outside of the field region



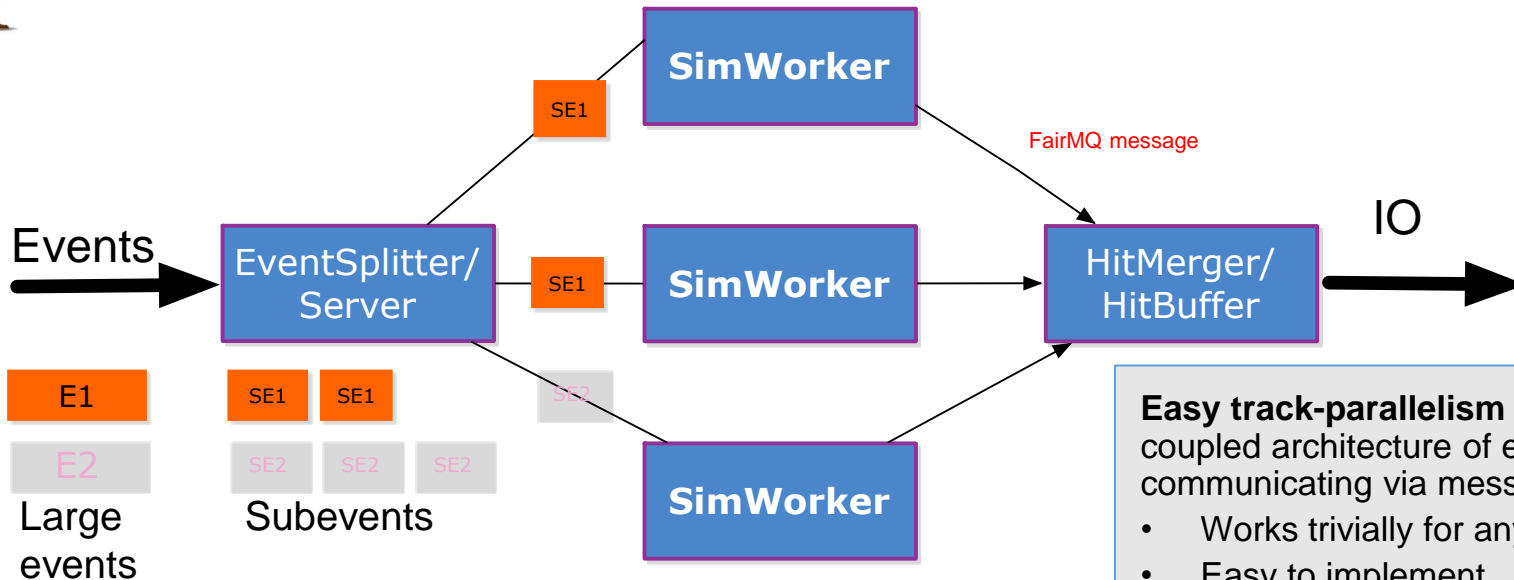
We will be able to improve simulation for Run3.

See strong opportunity for common tools here!

Easy Track Parallelism through FairMQ-based architecture

Remember: Basic simulation entities in terms of full events are (too) large and static

From full event to track-parallelism on the primary level to speed up simulation (time to return)



Easy track-parallelism by loosely coupled architecture of entities communicating via messages.

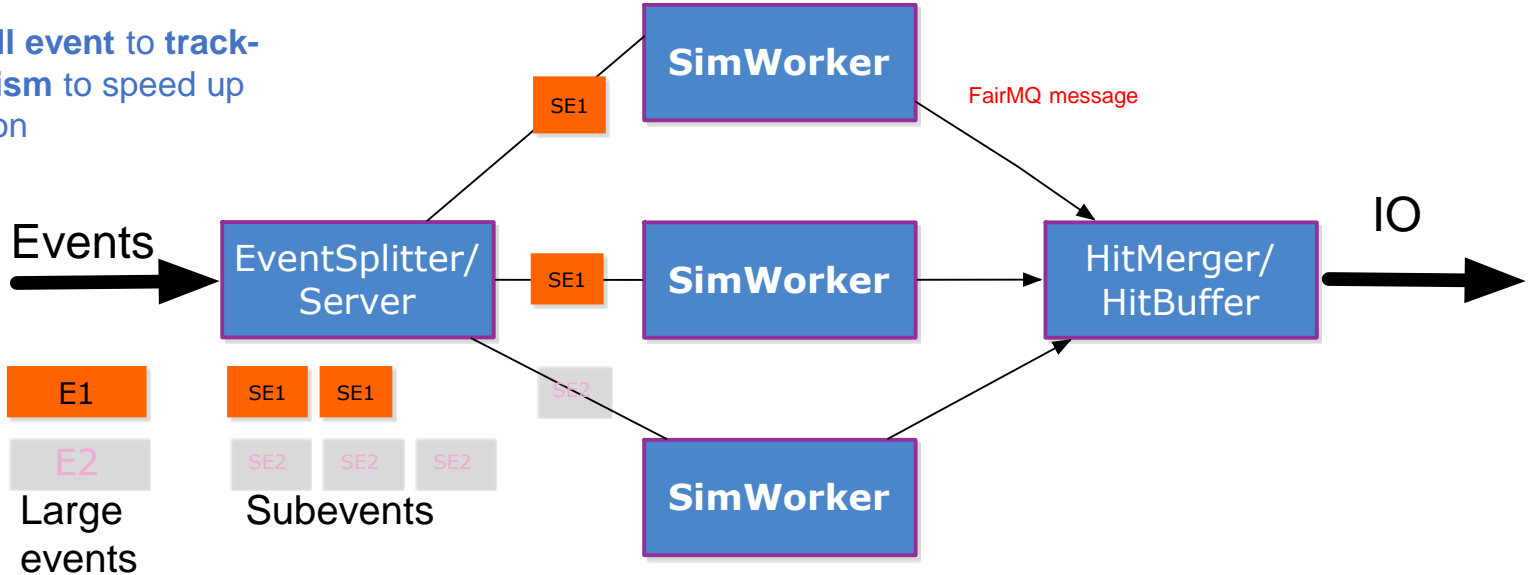
- Works trivially for any VMC worker
- Easy to implement
- Combinable with G4-MT
- Scalable across node
- Memory sharing by copy-on-write

Similar to ideas implemented by ATLAS

Embrace ALFA: FairMQ-based parallel simulation



From full event to track-parallelism to speed up simulation



“Event - based serial FairRoot simulation”

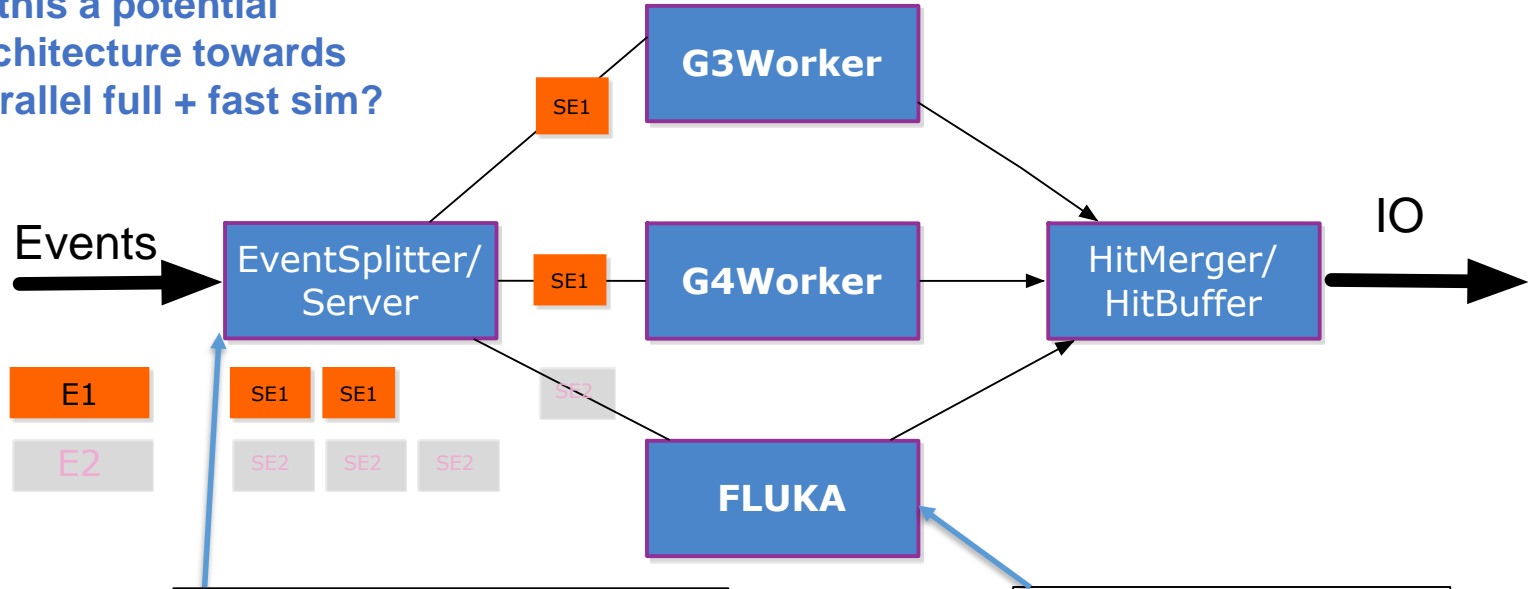
~45min one event (1core)

“Collaborative ALFA simulation treating subevents in parallel”

~6min for one event (6cores + HT);
<< 2GB per process

Towards fast-sim with heterogeneous workers

Is this a potential architecture towards parallel full + fast sim?



Step 1:

Different worker queues based on **primary properties** (function on type, energy, Pt, ...) and some heuristics

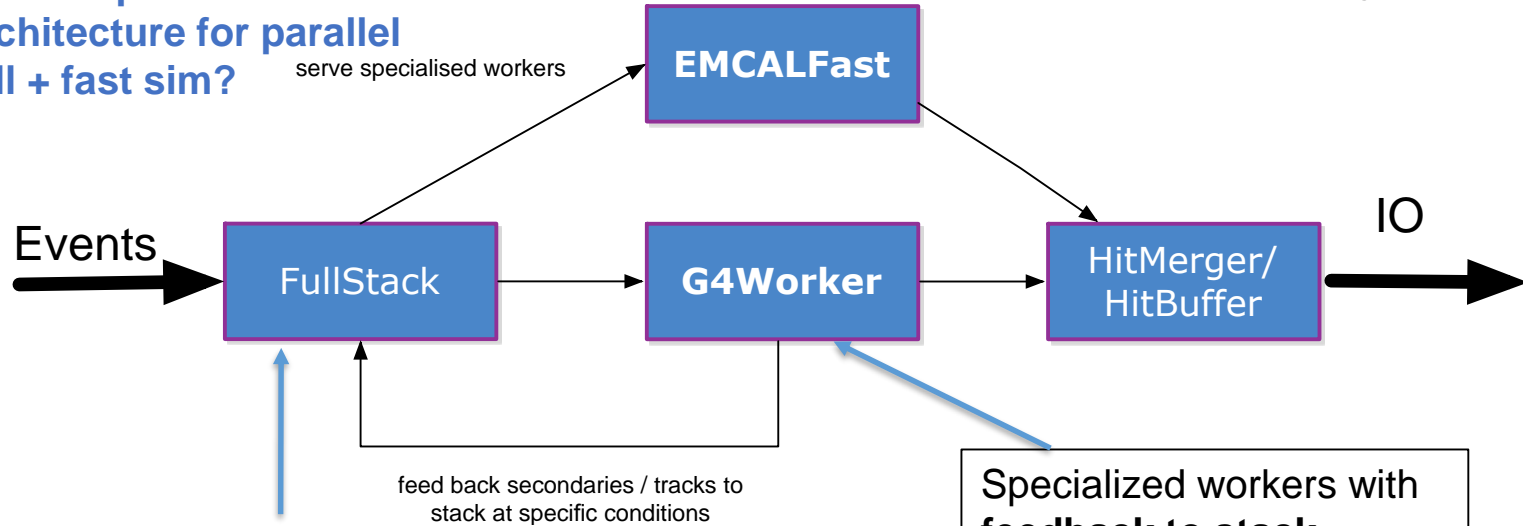
heterogeneous “specialized” workers

[Sharing ideas with “Heterogeneous Computing” presented by Ch. Leggett yesterday](#)

Ideas for a Parallel Full-Fast Sim Framework

O2 “thought” in progress

Is this a potential architecture for parallel full + fast sim?



serve specialised workers

feed back secondaries / tracks to stack at specific conditions

Possible Step 2???:

Worker selection based on **primary or secondary** (function on type, energy, Pt, **location**)

Specialized workers with **feedback to stack**

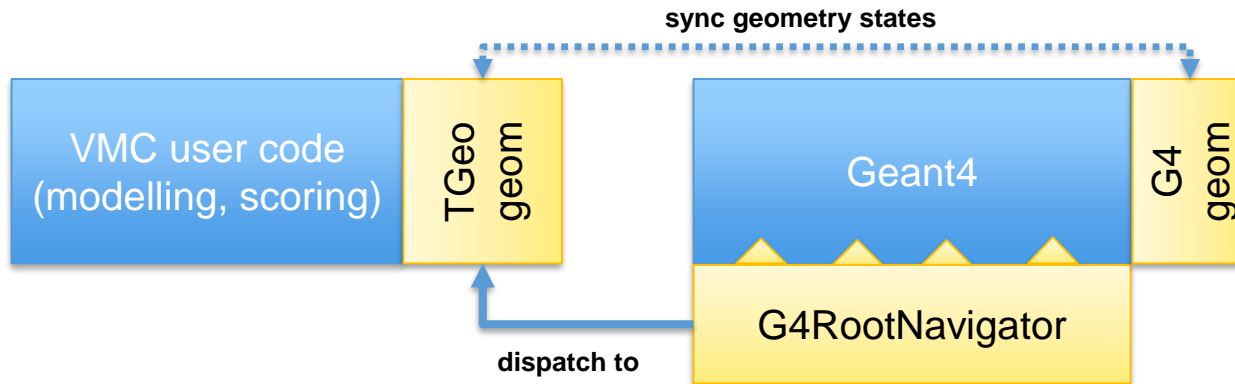
- Can be combined or alternatively done at VMC level inside each worker (see [Andreas Morsch](#))
- Need some prototyping
- **Finally, hope to raise incentive for physics community**

Improvements through Accelerated Geometry

Transport engines (G4) spend a considerable CPU amount in geometry related queries

Particularity for ALICE and other VMC users is to use **TGeo both** as

- **Descriptive** (modelling) language
- **Tracking geometry** in VMC (Geant3, Geant4, Fluka)



Improvements through Accelerated Geometry

Transport engines (G4) spend a considerable CPU amount in geometry related queries

Particularity for ALICE and other VMC users is to use **TGeo both** as

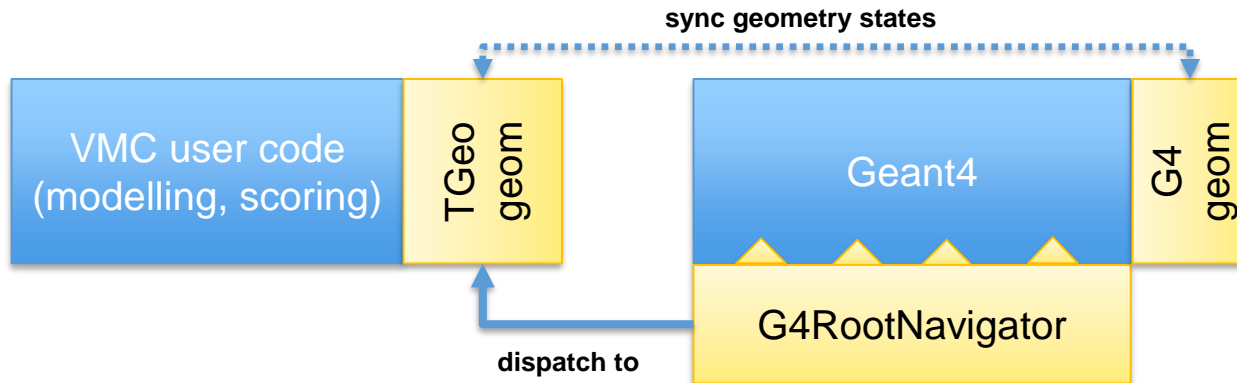
- **Descriptive** (modelling) language
- **Tracking geometry** in VMC (Geant3, Geant4, Fluka)

We like to profit from VecGeom

- Algorithms globally ~2-3x faster than TGeo

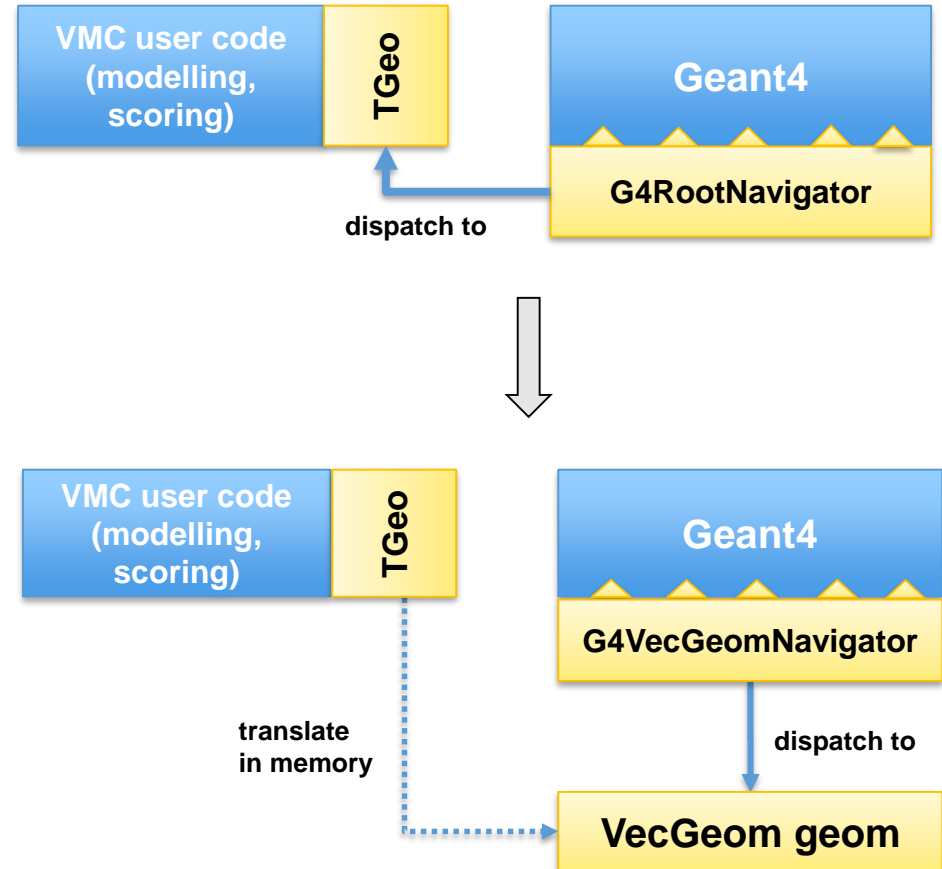
(How) can we profit from it?

- CMS [good reports](#) based on a first integration into G4... but not directly applicable to us
- We actually need integration of the VecGeom SIMD navigation engine



Improvements through Accelerated Geometry

- **ALICE is actively pushing VecGeom-G4 integration on the navigator level and contributing to the development**
- **As of 3/18, prototype available**
 - First promising results
 - No impact for user scoring code!
Benefit will come automatically!
- **Target integration into Geant4 to share this with other experiments**



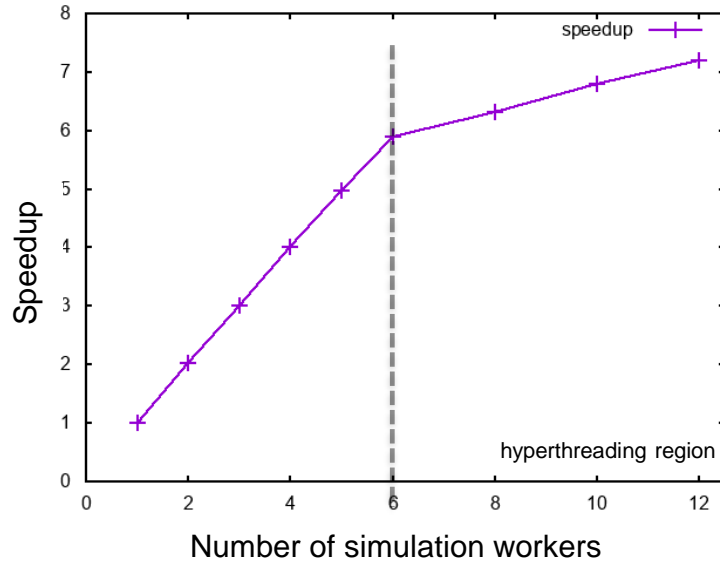
Executive summary

- ALICE is currently undergoing major changes in the software stack developed for Run3/Run4
- Changes/challenges for simulation include
 - Continuous Readout Simulation
 - New software framework and usage of common components based on FairRoot/ALFA
 - We are limited by long wall-clock times per event as well as memory
- ALICE is targeting/investigating “faster simulation”, e.g., through
 - A dedicated optimization program leveraging **monitoring data** and **tools**
 - A heterogeneous simulation architecture targeting **track-level parallelism (in VMC)**
 - A dedicated push to connect **VecGeom navigation** to G4 replacing TGeo navigation
 - Investigating fast simulation components (within an integrated full and fast simulation framework)
 - **Improved workflow** on grid and **enhanced scheduling capabilities** due to more fine-grained workloads
 - Support simulation R&D, e.g., the GeantV engine, and target future integration



BACKUP

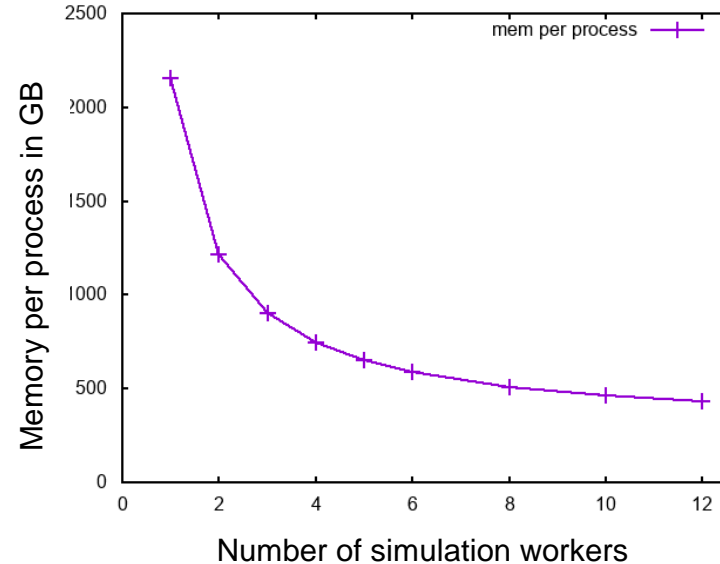
Preliminary performance study of device-based simulation



Perfect parallelization up to number of physical CPU cores;
Further gain in hyperthreading region

**~45min -> ~6min
to finish one event**

(on my desktop PC)



Better memory footprint per worker
as number of workers increases

Well below 2GB limit per worker

Subevent size 500primaries