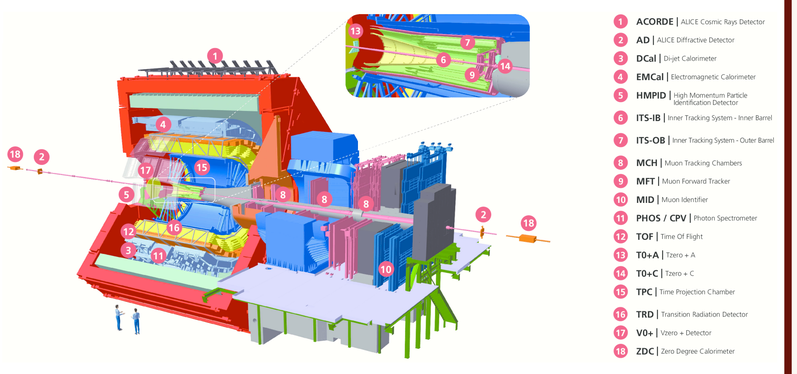
# Optimised Physics Simulation for the ALICE Upgrade

**Benedikt Völkel**, Heidelberg University & CERN

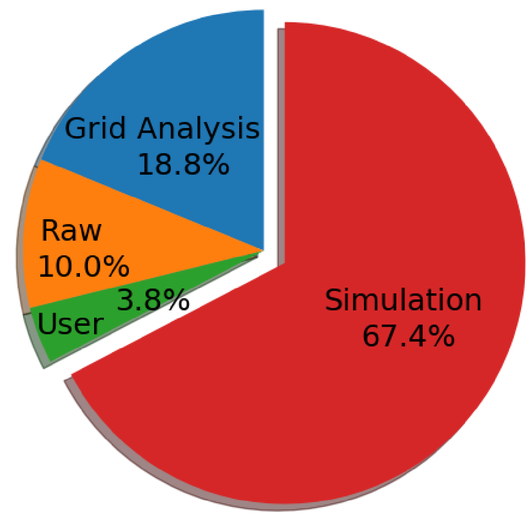
## The ALICE experiment




- optimised to study heavy-ion collisions at the Large Hadron Collider (LHC)
- optimised for particle identification and large particle multiplicities
- low material budget, optimised for particles with low momenta

$\Longrightarrow$ focus on QCD sector of Standard Model

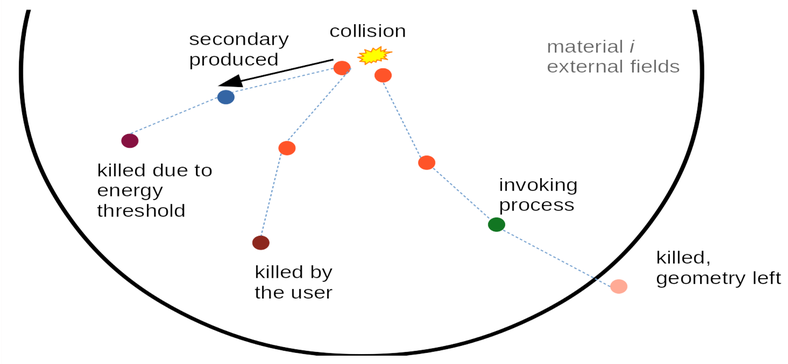
## Faster simulation required

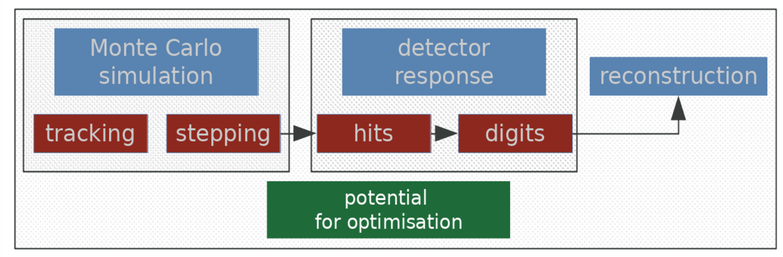



$\sim$ 10 (trigger) to $\sim$ 100 (min. bias) times more data expected in LHC Run 3

$\Longrightarrow$ **similar factor required in simulation to keep up with statistics in data**

## Current simulation approach

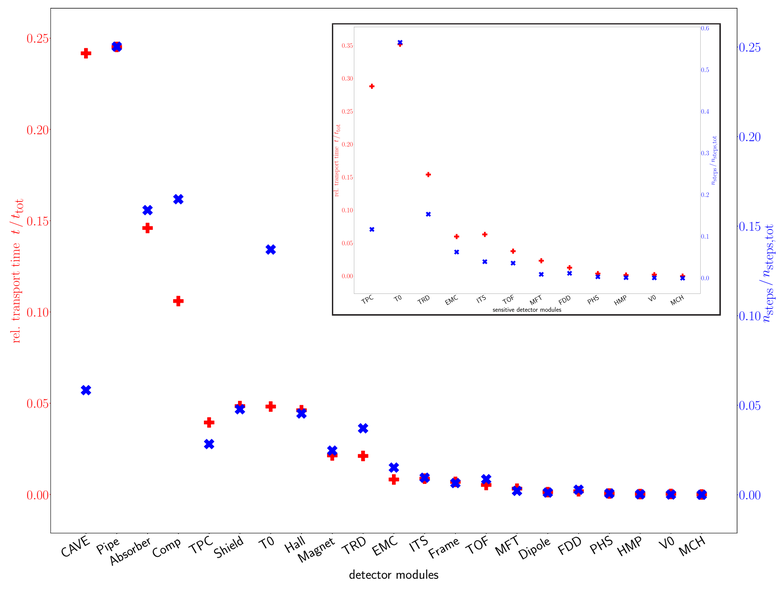



$\Longrightarrow$ **at each stage the potential for speed-up will be investigated**

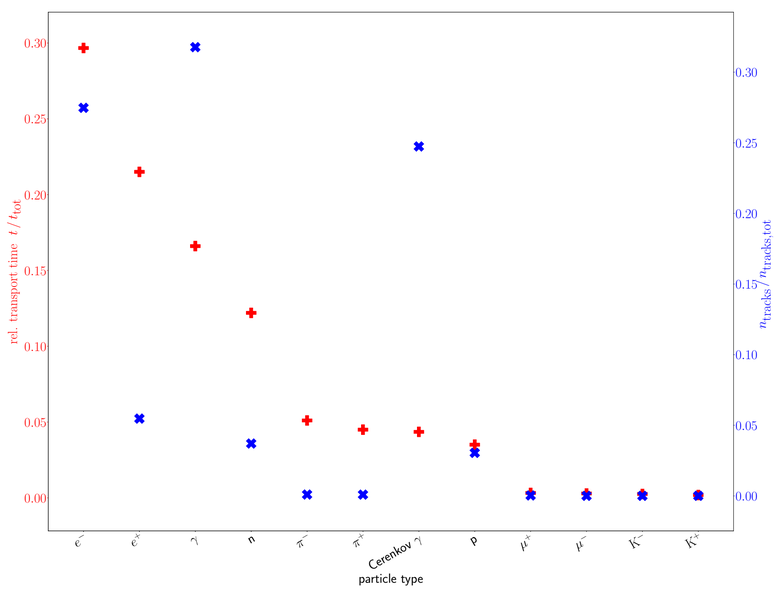
## Identify bottlenecks in full Monte Carlo simulation

### Time and steps per detector module



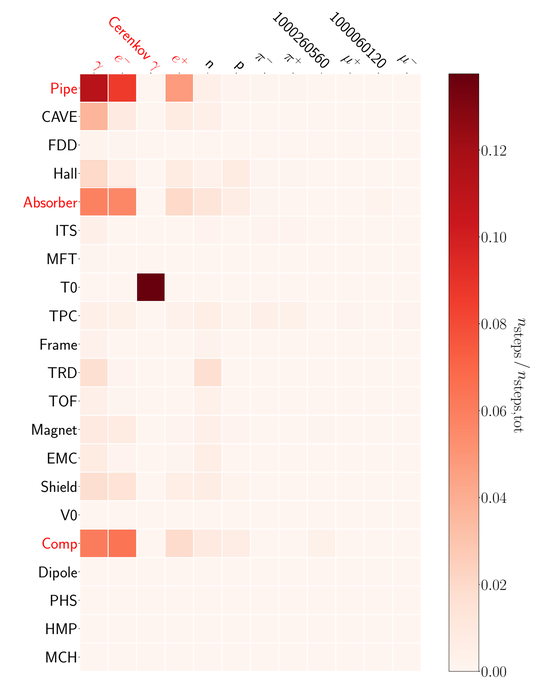

**Passive detector modules** are currently most resource demanding.

### Time and tracks per particle type




Most time consuming **particle types for tracks** are shown and can be clearly identified.

## Focus on passive modules first




## Optimisation strategy

1. transport
   (a) Monte Carlo parameter optimisation
   (b) parametrisation
   (c) simplified geometry
   (d) machine learning approaches
2. speed up digitisation and reconstruction
3. use embedding techniques

## Summary and outlook

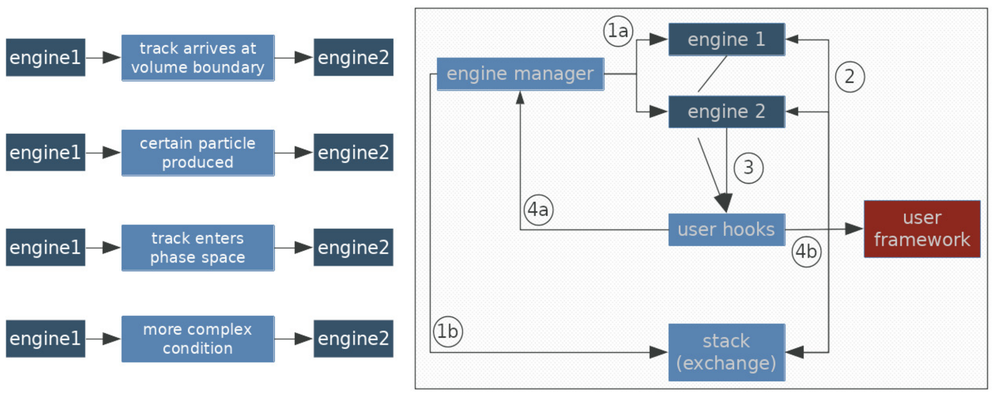
- expect $\sim$ 100 times more min. bias events $\Longrightarrow$ speed-up of simulation is crucial
- first optimise passive detector modules
- in-depth study for sensitive detector modules required
- use advantages of extended simulation workflow

## References

[1] ALICE collaboration. *The ALICE experiment at the CERN LHC*, JINST 3 (2008) S08002

[2] Z. Citron et al. *Future physics opportunities for high-density QCD at the LHC with heavy-ion and proton beams*, arXiv:1812.06772 [hep-ph]

## New extended (fast) simulation workflow




- first time that ALICE can mix multiple full and fast simulation engines
- providing vital new flexibility
- enable extended detector simulation studies
- fully automated workflow

## Assembling simulation engines

**MyFirstEngine**

```
#include <TVirtualMC.h>

class MyFirstEngine
: public TVirtualMC
{
  // implementations
};
```

**MySecondEngine**

```
#include <TVirtualMC.h>

class MySecondEngine
: public TVirtualMC
{
  // implementations
};
```

**VMCFastSim (virtual)**

```
#include <TVirtualMC.h>

template <class T>
class VMCFastSim
: public TVirtualMC
{
   //...
   virtual void process() = 0;
   //...
};
```

**user framework (simplified)**

```
// Get the TMCManager
auto manager = TMCManager::Instance();

// create user stack and notify manager
auto stack = new MyStack();
manager->SetUserStack(stack);

// create engines
//--------------------------------------
auto engine1 = new MyFirstEngine();
auto engine2 = new MySecondEngine();
auto fastSim = new MyFastSim();
//--------------------------------------

// specify condistions how to change
// and transfer tracks among engines

// init...
manager->Init();
// ...and run 42 events
manager->Run(42);
```

**MyFastSim**

```
#include <VMCFastSim.h>

class MyFastSim
: public VMCFastSim<MyFastSim>
{
  public:
    //...
    void process() override
    {
      // put implementation here
    }
};
```

FastSim1
FastSim2
FastSim3

- combine any full and fast simulation engine
- provide abstract class for fast simulation implementation
- fast simulation possible with full simulation engines which do not support this functionality natively

## Approach for passive detector modules

since there is no sensitive element, skip stepping inside passive detector modules to become faster