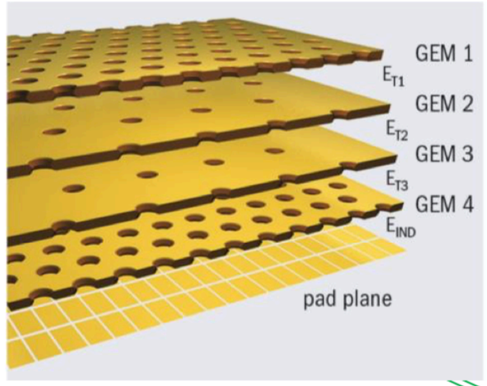
ALICE @ AF

Giulio Eulisse

ALICE IN RUN3

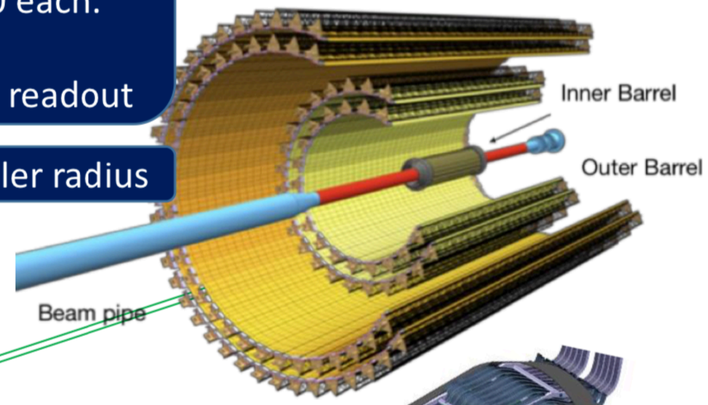
ALICE HW upgrades

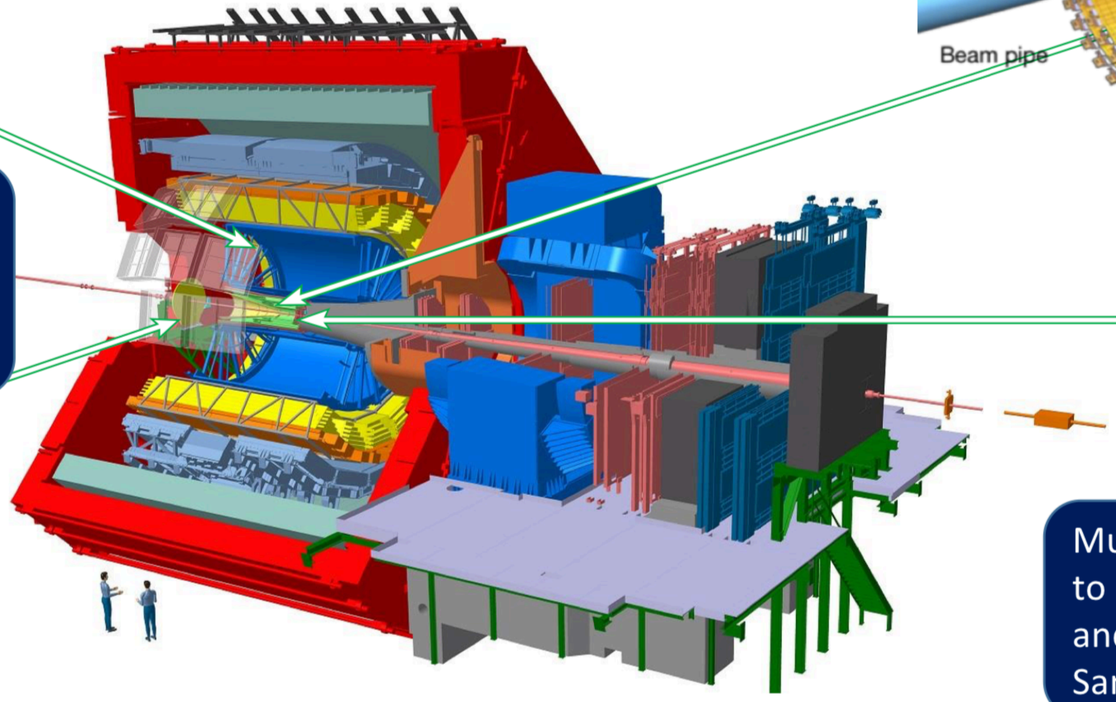


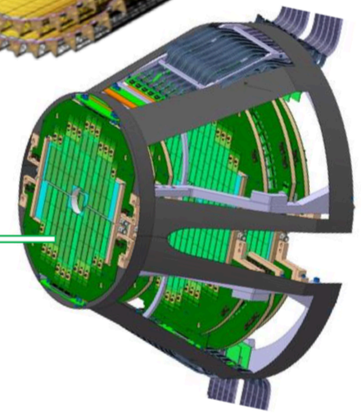
TPC MWPC readout → 4 layer GEM
(Intrinsic ion backflow ~99% blocking)
5MHz continuous sampling

New Si Inner Tracker: 10 m^2 of MAPS with 29x27µm^2 pixel size
3 inner layers ~0.3% X0 each.
Closer to the beam
50-500 kHz continuous readout

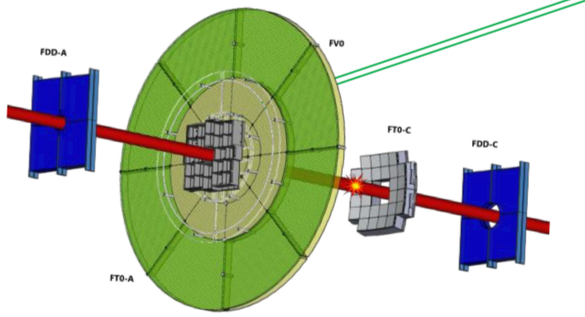
New beam pipe of smaller radius







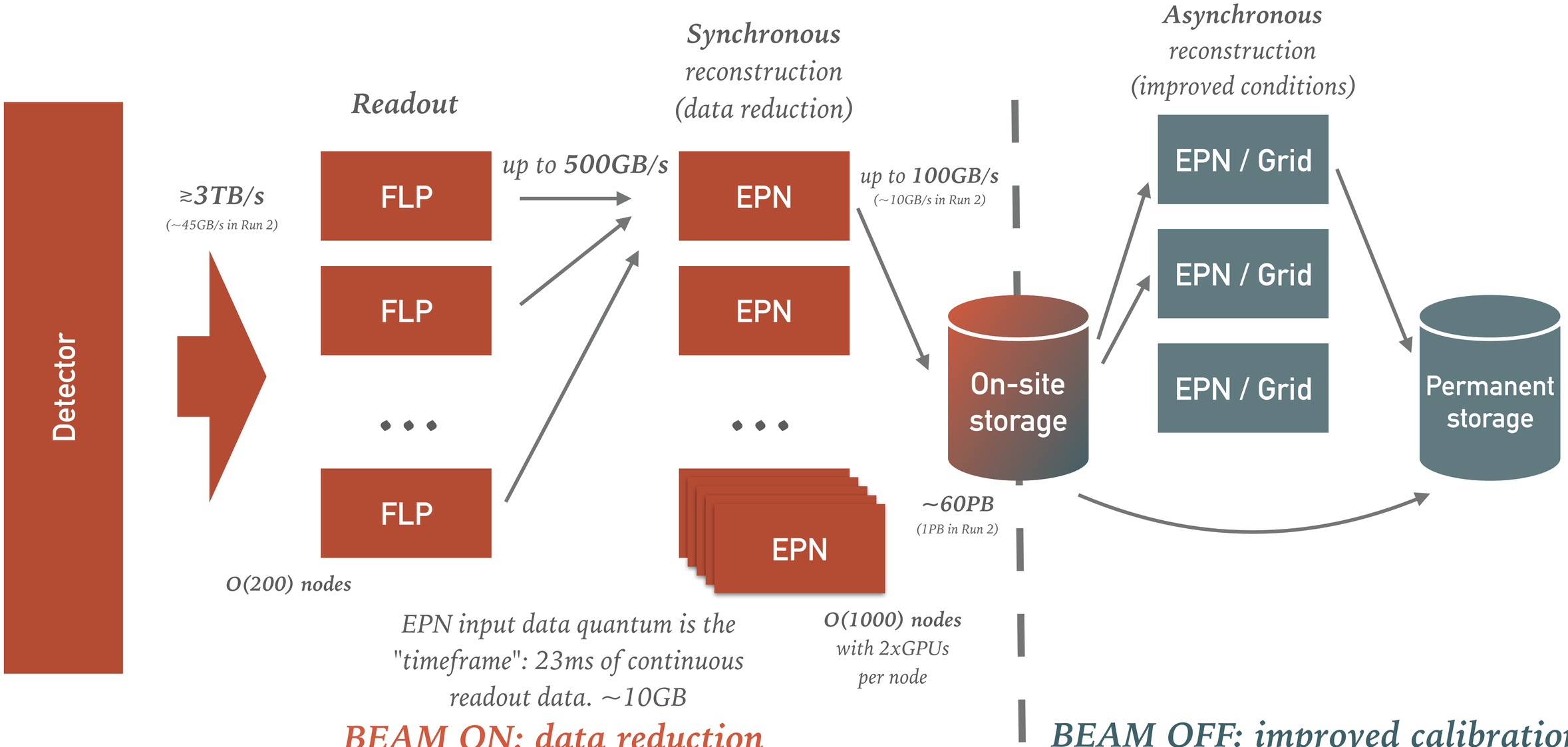
Fast Interaction Trigger (FIT) detector
Scintillator (FV0, FDD) + Cerenkov (FT0) detectors to provide Min.Bias trigger for detectors with triggered R/O



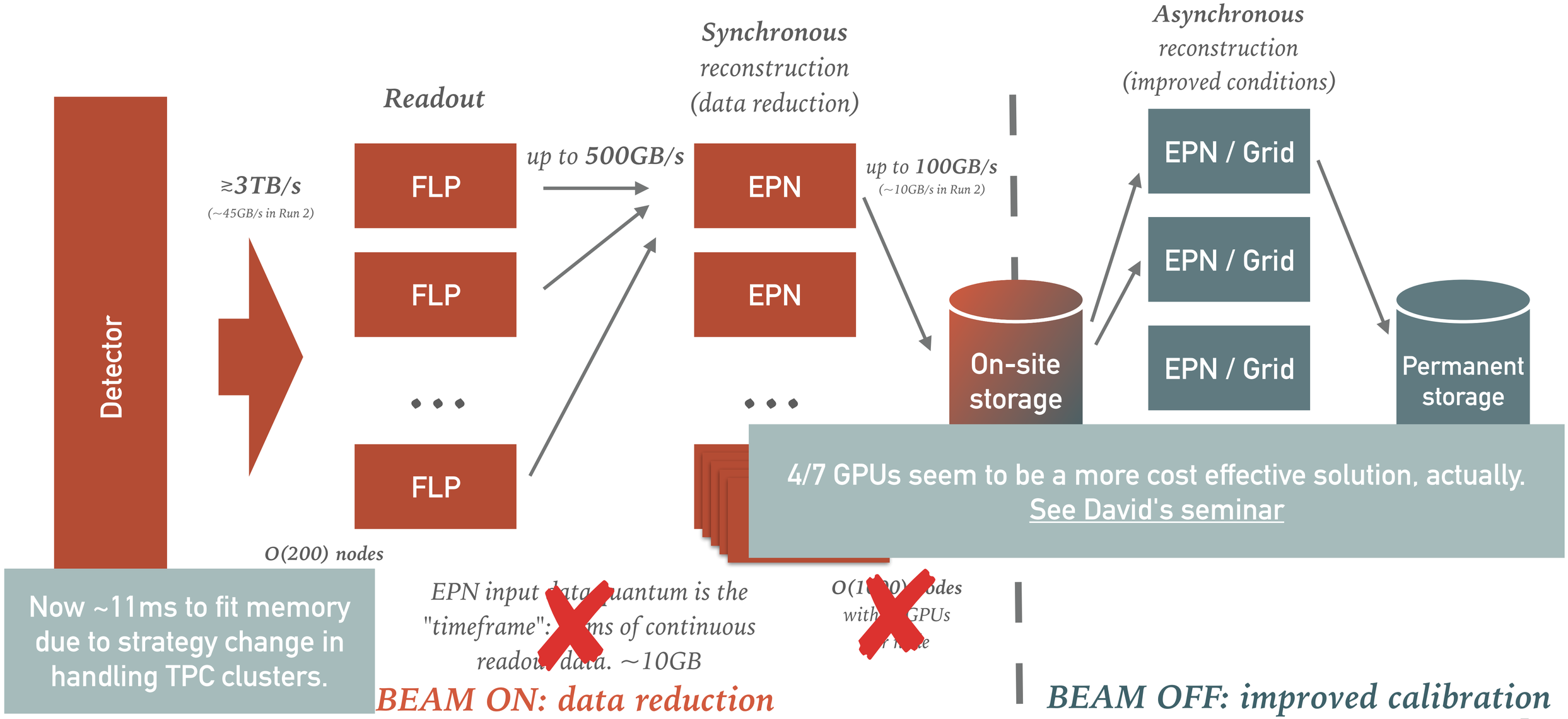
Muon Forward Tracker
to match muons before and after the absorber.
Same Si chips as new ITS

See CHEP2019 Plenary: ALICE continuous readout and data reduction strategy for Run3

ALICE IN RUN 3: POINT 2



ALICE IN RUN 3: POINT 2

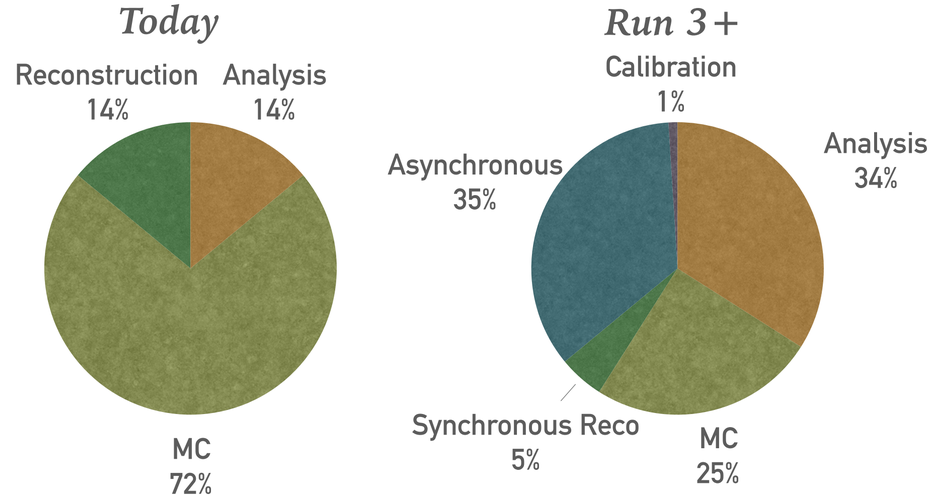


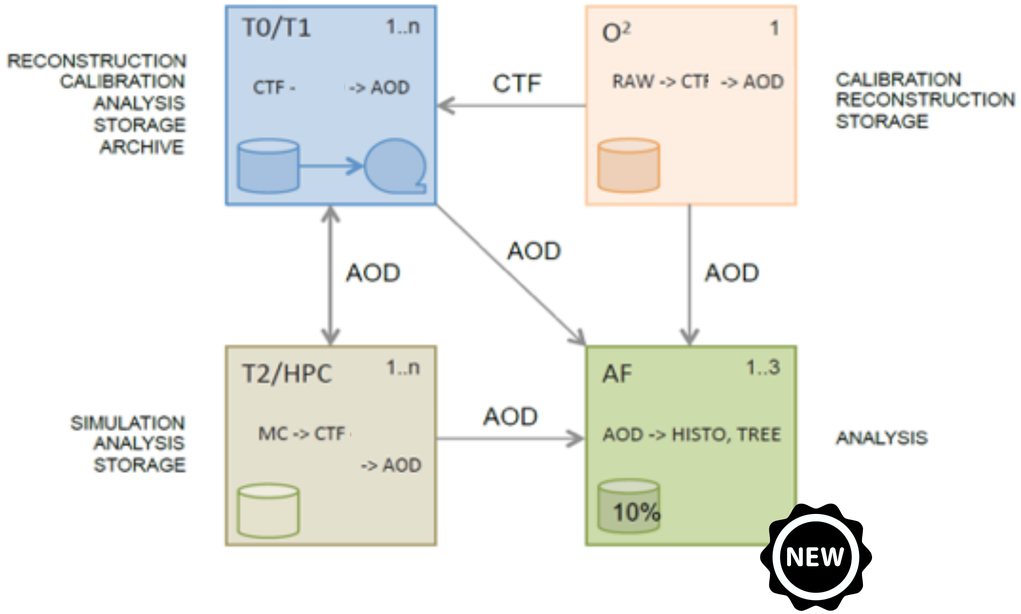
Shameless plug: Data Analysis Using ALICE Run3

Analysis Model: Run 3

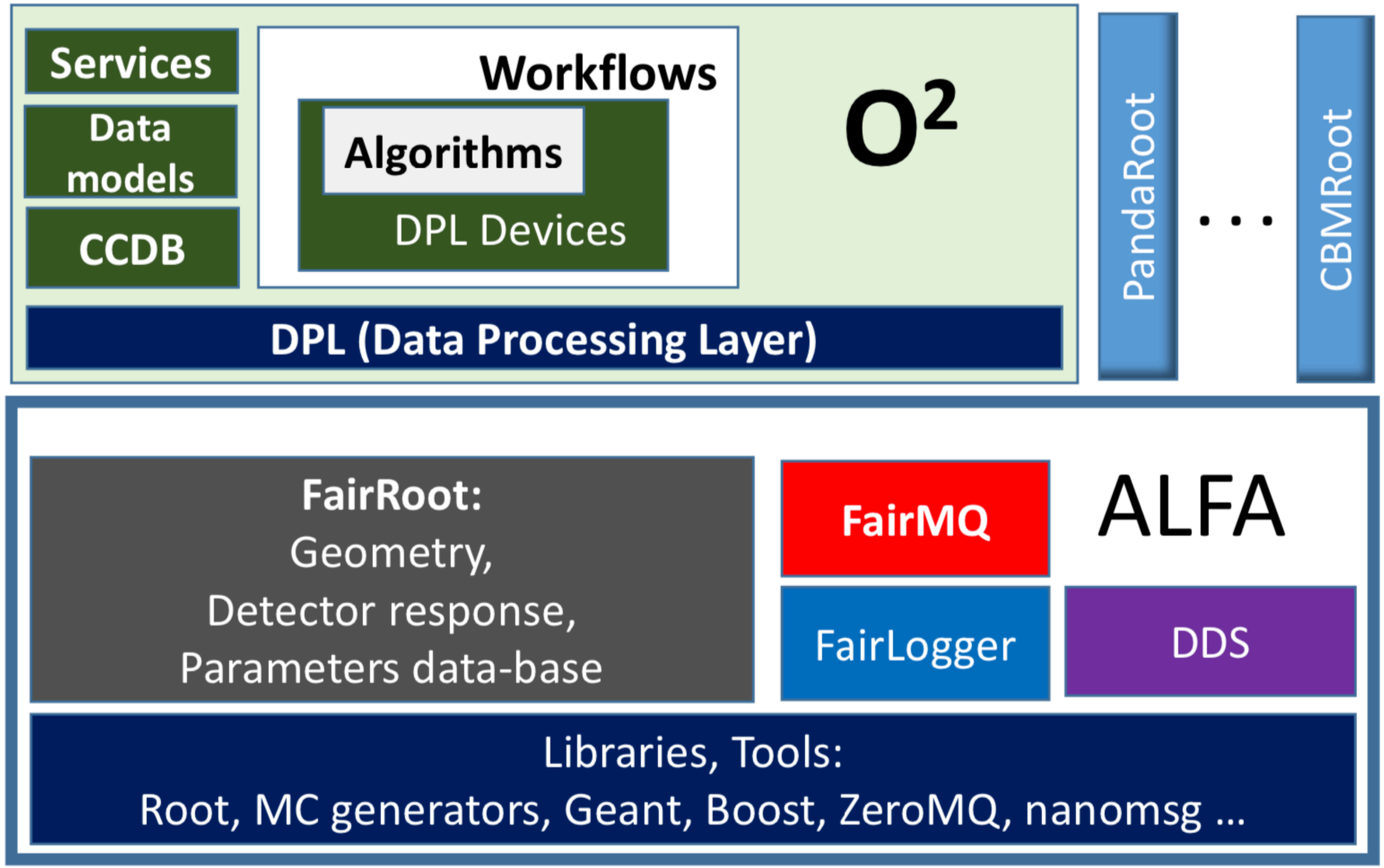
Solid foundations: *the idea of organised analysis (trains) will stay. Improve on the implementation.*

- ***x100** more collisions compared to present setup, **AOD only.***
- *Initial analysis of 10% of the data at fewer **Analysis Facilities,** highly performant in terms of data access.*
- *Full analysis of a validated set of wagons on the Grid*
  *⇒ **Prioritise processing according to physics needs**.*
- ***Streamline data model**, trade generality for speed, flatten data structures.*
- ***Recompute** quantities on the fly rather than storing them.*
  CPU cycles are cheap.
- ***Produce highly targeted ntuples** (in terms of information needed and selected events of interest) to reduce turnaround for some key analysis.*
- ***Goal is to have each Analysis Facility go through the equivalent of 5PB of AODs every 12 hours (~100GB/s).***

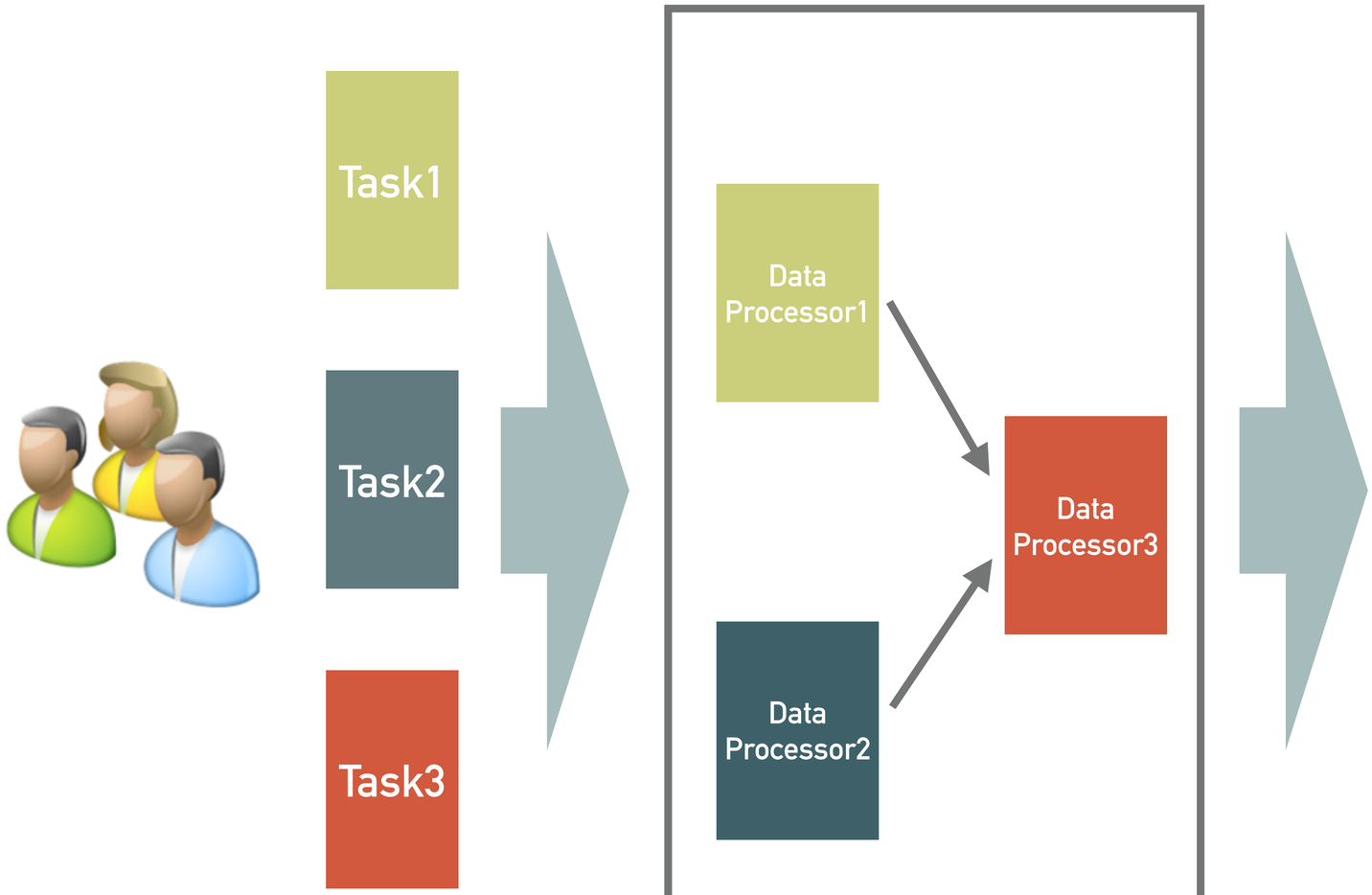




THE BIG PICTURE

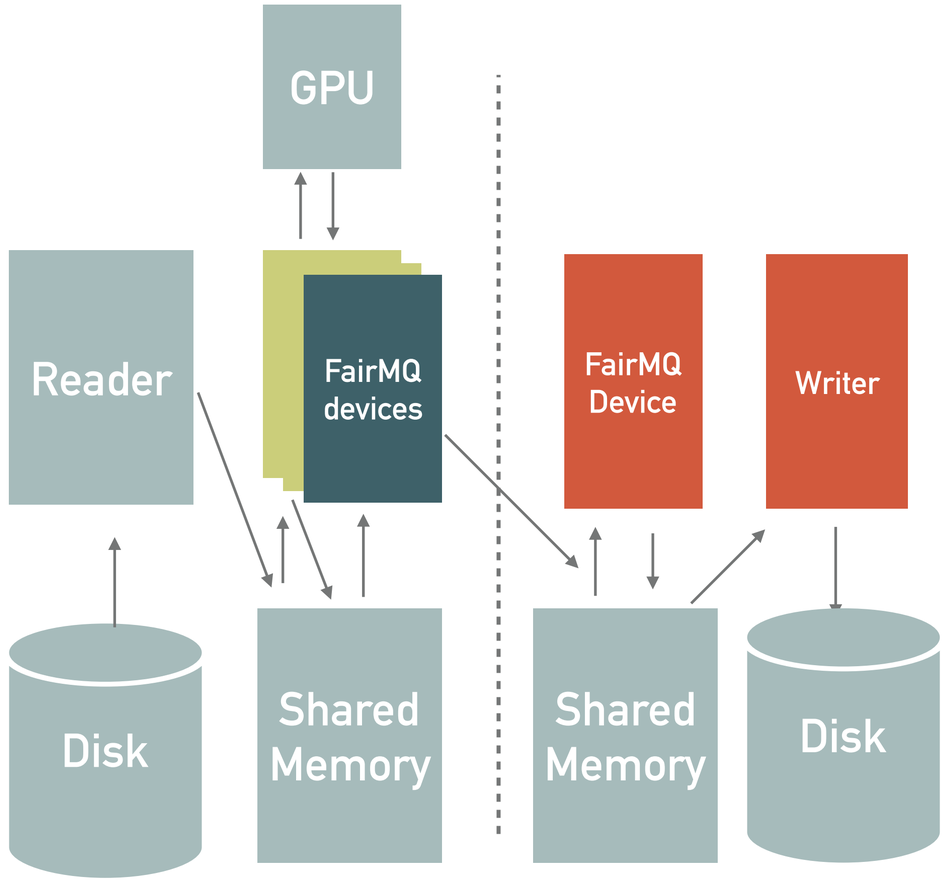


O2 FRAMEWORK ARCHITECTURE



Users define tasks and their dependencies

O2 DPL builds a workflow (DAG) out of the specified tasks



O2 DPL builds a topology of FairMQ devices and maps tasks on it into account available resources

FEATURES

- Asynchronous: nicely integrates offloading to GPUs.
- Resilient: failing tasks do not affect data-taking. Particularly important as Heavy Ion run is only a few weeks and downtime weights more in terms of the total fraction of data processed.
- Asymmetric / distributed: topology builder can optimise for asymmetric / distributed resources (e.g. NUMA domains, GPUs, Network).
- Trivially parallel: no need for all the components to be multithread safe in order to parallelise.
- Inherently distributed.

ALICE & AF

ROOT

Run 2
ROOT5, despite efforts to go away with it (due to memory issues).

Run 3

Brave new world: ROOT6 (as recent as possible).

- Run3 will be ROOT6. Baseline for persistency, analysis.
- Strong interest in RDataFrame / RNTuple for analysis.
- Strong desire to have seamless ROOT integration with Apache Arrow (backend for our shared memory message passing in the case of AODs).
- Strong desire to investigate common solutions for an SoA abstraction layer. We have our own, Arrow based. Overlapping features to what CMS and LHCb have.
- C++20 / modern Clang support before Run 3 would be highly appreciated.
- TMVA used for ML efforts.

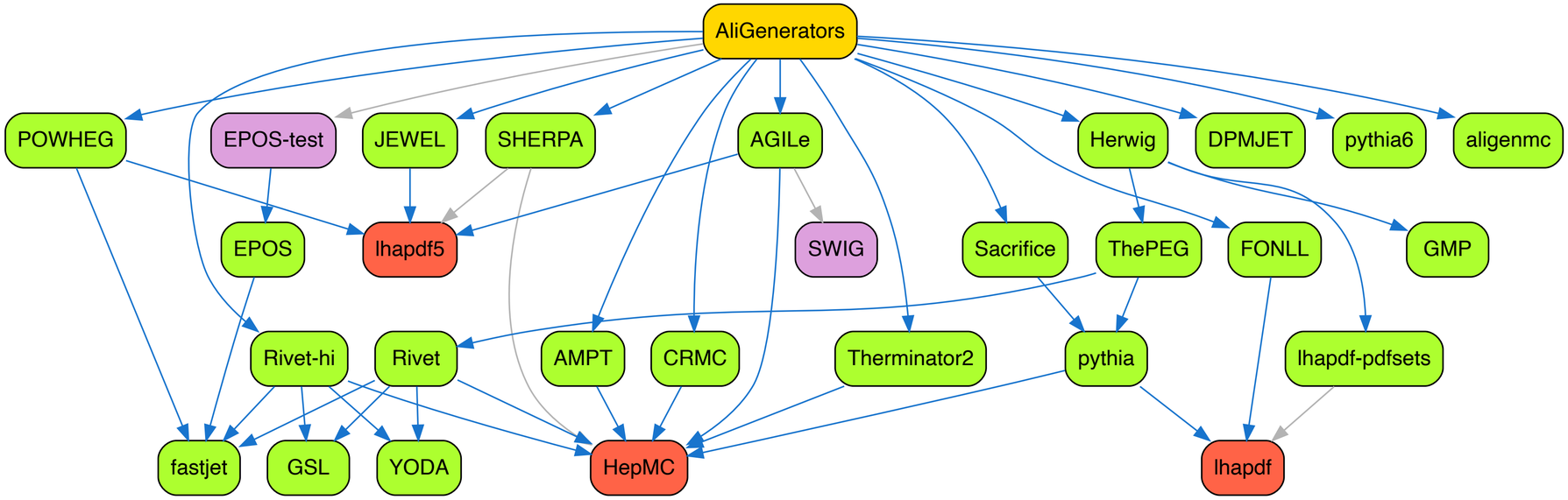
SIMULATION

GEANT3 / GEANT4:

- GEANT3 is still baseline for Run2. GEANT4 lost the train.
- Run 3 will move fully to GEANT4.
- VMC as abstraction layer.
- Multiprocess approach makes combining GEANT3 and GEANT4 (or others) possible.
- Investigating ability to have multiple engines within the same process, e.g. to plug in fast simulation.
- In event parallelism via message passing in order to scale on multiple cores / nodes and take advantage of short opportunistic time windows.
- Some in house debug tools e.g. VMCStepLogger others might be interested in.
- Interest for fast / GAN simulation support in GEANT4.

GENERATORS

- We mostly use Pythia6, Pythia8, Hijing, EPOS.
- We have our own "AliGenerators" distribution with a few extra HI specific packages and related software.
- We have some usecase requiring FLUKA.



COMPONENTS IN COLLABORATION WITH GSI

Extremely successful partnership with GSI common software foundations (ALFA).

Modular approach of the ALFA stack allows us to pick what we need, where we need it:

- FairRoot: common simulation and reconstruction components (e.g. magnetic field).
- FairMQ: actor model / message passing framework with multiple backends.
- asiofi: Boost.Asio language bindings for OFI libfabric.
- FairLogger: multithreaded, multibackend logging library.
- DDS: process topology deployment.

INTEGRATION WITH GPUS

A large fraction of ALICE Data Processing in Run 3 will happen on GPUs.

- See David's seminar for an in-depth description.
- We have our own library with multiple GPU / paradigm supported via preprocessor macros. Backend for NVIDIA CUDA, AMD HIP, OpenCL, and CPU (for consistency checking).

Some of the hurdles encountered include:

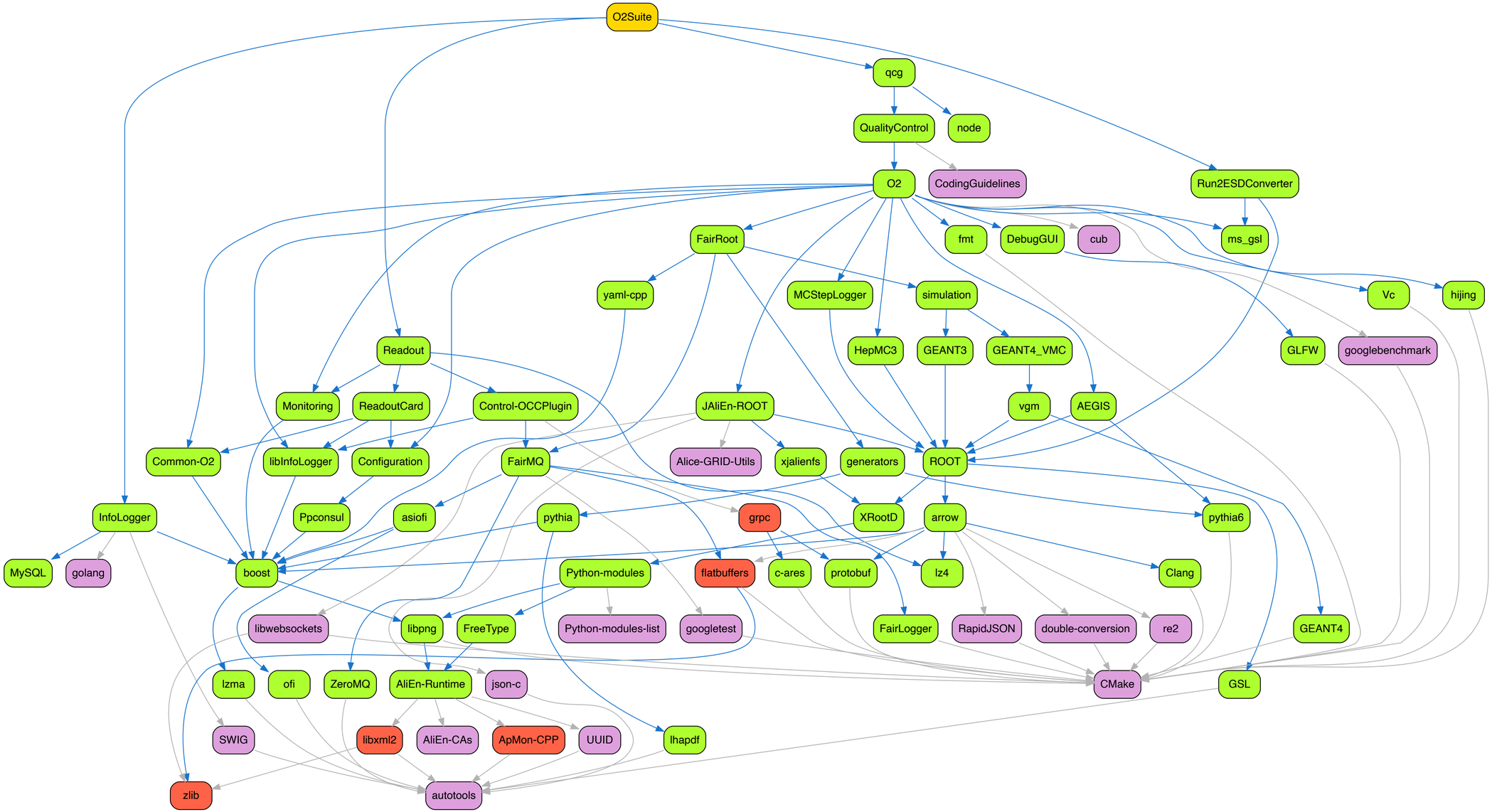
- Training and expertise building.
- Debugging of middleware issues to make it digest our code.

Major common point is making sure keep the information flowing and share experience and issues about our developments. CERN/EP seminar was a great example.

SOFTWARE COMPONENTS AND THEIR ORGANISATION

- Compared to Run1/2 we are moving to a **multi-repository / multi-project organisation** for both internal and external packages.
- Notable Boost and Python adoption.
- While C++ remains the most popular language for the reconstruction and simulation code, we see increasing adoption of Python (analysis), Java (AliEn), Go (AliECS - FLP Control System).
- Clang / LLVM is leaking in a number of places (Arrow query engine, GPU middleware, tooling).
- Challenge is of course to keep complexity under control.

FULL PICTURE (RECO + SIM ON MAC)



BUILD SYSTEM AND SOFTWARE DISTRIBUTION

- **aliBuild** served us well over the last 5 years, with a user base of ~300 people. Same tool for central installations and running laptops. We are more than open to move to **Spack**, but we must be sure use cases for such a large audience are well covered (SpackDev? RPMs? Automatic system dependencies? Recipe complexity under control?) and that the migration effort does not go beyond one month FTE.
- **CMake** remains the tool of choice for building the actual sub projects.
- **Github** has served us perfectly as a code hosting platform.
- **Jenkins + Apache Mesos** used for CI. We are looking into "**Github Actions**" and maybe Kubernetes to replace part / all of it. We would welcome some centralised effort to provision on-premise workers for Github repositories.
- **CVMFS** baseline for large scale deployments and to deploy Conditions to grid jobs.
- Build artifacts (tarballs, RPMS, logs) currently on single (mirrored) machine, in the process of consolidating to **CERN/IT S3.**

DEPLOYMENT

- Computing resources at P2 will be configured using **Ansible**.
- FLP control system is built around **Apache Mesos** with a custom written "framework" (in Go) which integrates into FairMQ plugin mechanism and aims to minimise downtime due to reconfiguration. Configuration is imported in **Consul**.
- EPN control system is built around GSI **DDS** as a baseline, which also integrates into FairMQ plugin mechanism. We are also looking into **PMIx** which has strong push from (supercomputing) industry. FairMQ plugin mechanism allows us to investigate multiple options.
- On the grid, business as usual: **AliEN** + single executable "DPL driver" which will take care of starting a given multiprocess workflow. One executable per workflow. Workflows can be merged to form larger ones. **Singularity & Docker** are the way forward for us for Grid Deployments and we plan to be 100% there by Run 3.

MISC

- CERNVM required by data preservation and outreach.
- Many users using Python ecosystem for ML: TensorFlow, PyTorch, XGBoost, Scikit-learn, Keras, Theano. TMVA also used.
- Jupyter being used for tutorials, outreach and "interactive analysis".
- Simulation tests on HPC facilities in conjunction with CNAF, Oak Ridge.