

Status of the micro framework development (infrastructure, CI, fast reconstruction)

V. Cicero for the SAND software WG

Meeting annuale della collaborazione DUNE-Italia

11/11/2025



- **framework with minimal features as a bridge between sandreco and DUNE reconstruction framework**
- **C++ application engine with plugin algorithms**
 - Load only the modules you use
 - Simpler building, dependencies are somewhat isolated
 - .json configuration to decide what to run and how to configure
- **motivations:**
 - Sandreco code is almost monolithic : proper testing and development difficult, unfriendly for beginners
 - Current data structures are suboptimal: poor match to info for filling CAFs, some unfriendly to full spill or truth matching

μfw design

- **Focus on moving complexity away from algorithms and configuration scripts**
 - Worst of the C++ nasty stuff hidden from people developing algorithms, who tend to be junior/less experienced
 - Hiding most complexity in the framework engine and, in part, data structures (it has to go somewhere...)
- **Adopt some more modern C++ (17 minimum required, would like to go to 23)**
 - More moving, less copies, more in-place data operations, ...
- **Streamline and/or hide ROOT I/O**
 - Automatic dictionaries and trees. Users only see their structs
- **More details and instructions** on how to implement modules here:
<https://baltig.infn.it/dune/ufw/-/wikis/home> and a registered tutorial session here:
<https://indico.fnal.gov/event/70441/>

μfw configuration

- A json file with:
 - Some global parameters
 - Some meta info on the spills to be processed
 - A sequence of algorithms and I/O operations to run
 - Their configuration

```
{
  "ufw" : {
    "ufw-loglevel" : "debug",
    "ufw-basepath" : "/usr/local/share/sandeco/data",
    "ufw-ldpath" : ["/usr/local/lib64"]
  },
  "globals" : {
    "sand::geomanager" : { "geometry" : "test/grain.edepsim.root" },
    "sand::grain::geant_run_manager" : {}
  },
  "contexts" : {
    "keys" : 100,
    "locals" : {
      "sand::edep_reader" : {"uri" : "test/grain.edepsim.root"}
    }
  },
  "run" : [
    {
      "sand::grain::optical_simulation" : {
        "geometry" : "geometries/grain/masks-3mm/main.gdml",
        "energy_split_threshold" : 100
      },
      "reqs" : {},
      "prods" : {"hits" : "grain_hits"}
    },
    {
      "sand::grain::detector_response_fast" : {
        "sipm_active_size": 3.0,
        "sipm_border": 0.1,
        "pde": 0.2,
        "reqs" : {"hits" : "grain_hits"},
        "prods" : {"digi": "grain_digits"}
      },
    },
    {
      "sand::grain::spill_slicer" : {
        "slice_times": [0.0, 20000.0]
      },
      "reqs" : {"digi" : "grain_digits"},
      "prods" : {"images": "grain_images"}
    },
    {
      "sand::root::tree_streamer" : {
        "uri" : "test/images.root",
        "tree" : "cameras"
      },
      "write" : { "grain_images" : "sand::grain::images" }
    }
  ]
}
```

sandreco status

- sandreco with ufw is now sandreco-experimental

- The “old” one will be *sandreco-legacy*

- We started adding/porting modules

- started with GRAIN code:

- Missing entirely from sandreco-legacy
 - More complex requirements: Geant4 as dependency, GPU code with OpenCL, memory use... (~100GB system matrix)
 - If we can fully port GRAIN code, the rest *should be* easier (famous last words...)

dune / sandreco-experimental / Issues

Open 40 Closed 14 All 54

sandreco status

Modules:

- Geometry managers for **GRAIN**, **Drift**, **STT**,
- **GRAIN** :
 - optical simulation
 - Detector response (fast)
 - Spill slicer
 - OpenCL test module with GPUs (needed for coded aperture)
- **Tracker**:
 - Digitization
- Reader for **edep-sim** and **Genie** files

completed

to be tested/reviewed

CI pipeline

<https://baltig.infn.it/dune/sand-ci>

- Deployed on INFN's gitlab
- Based on a set of docker images:

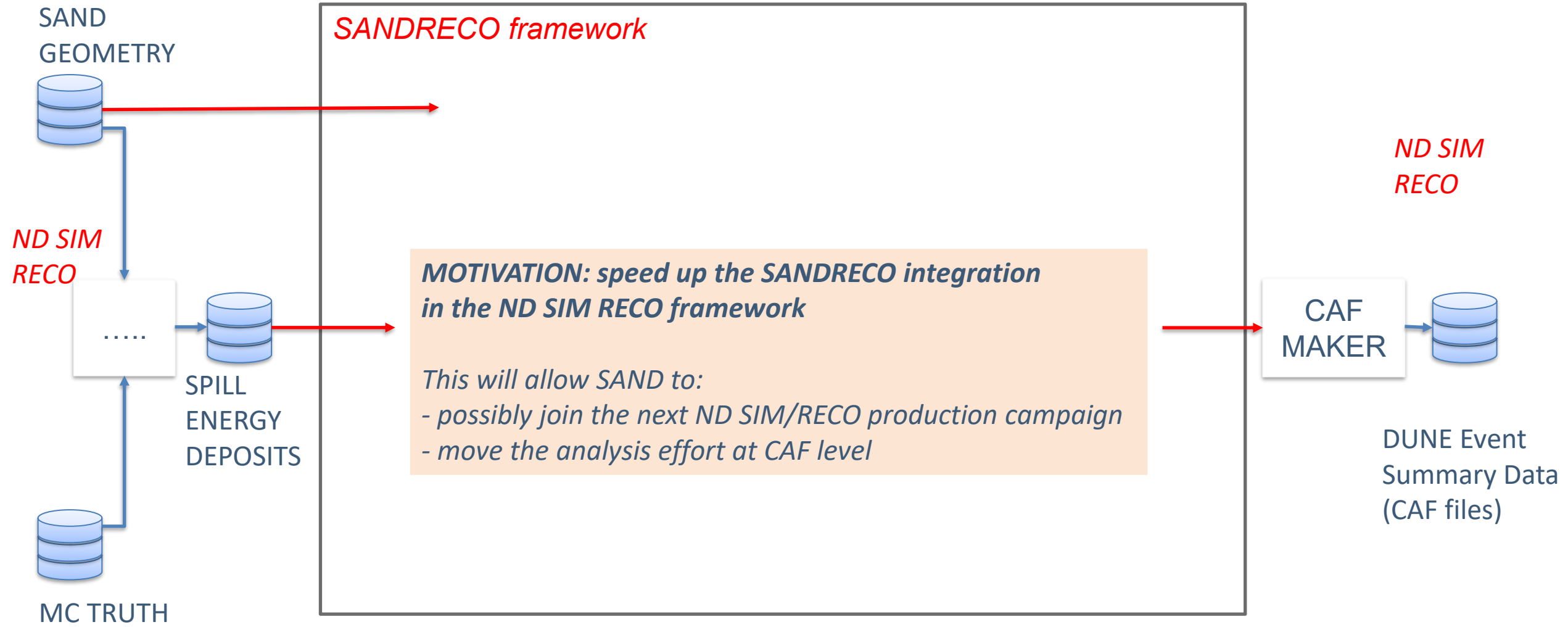
```
base --> physics-sw --> sandreco-ci -> sand-prod -> sand-dev
      |               |
      +-->   ufw-ci   +-+
```

- **base:** This is a base image with system tools: almalinux9 plus cmake, gcc, etc...
- **physics-sw:** This is the set of dependencies for physics. It is built in stages, starting from the system root rpm: 6.34.06.el9, installing from source geant4 and edep-sim, built with system compiler (gcc 11.5), both installed in /usr/local
- **ufw-ci:** This image is meant to be used for the CI pipeline of ufw itself. It can also be used as a build environment for its development.
- **sandreco-ci:** This image is meant to be used for the CI pipeline of sandreco. It can also be used as a build environment for local development.
- **At every pushed commit in sandreco-experimental, the sandreco-ci image builds and runs tests**
 - tests can added by placing their coinfiguration .json file in tests/framework folder

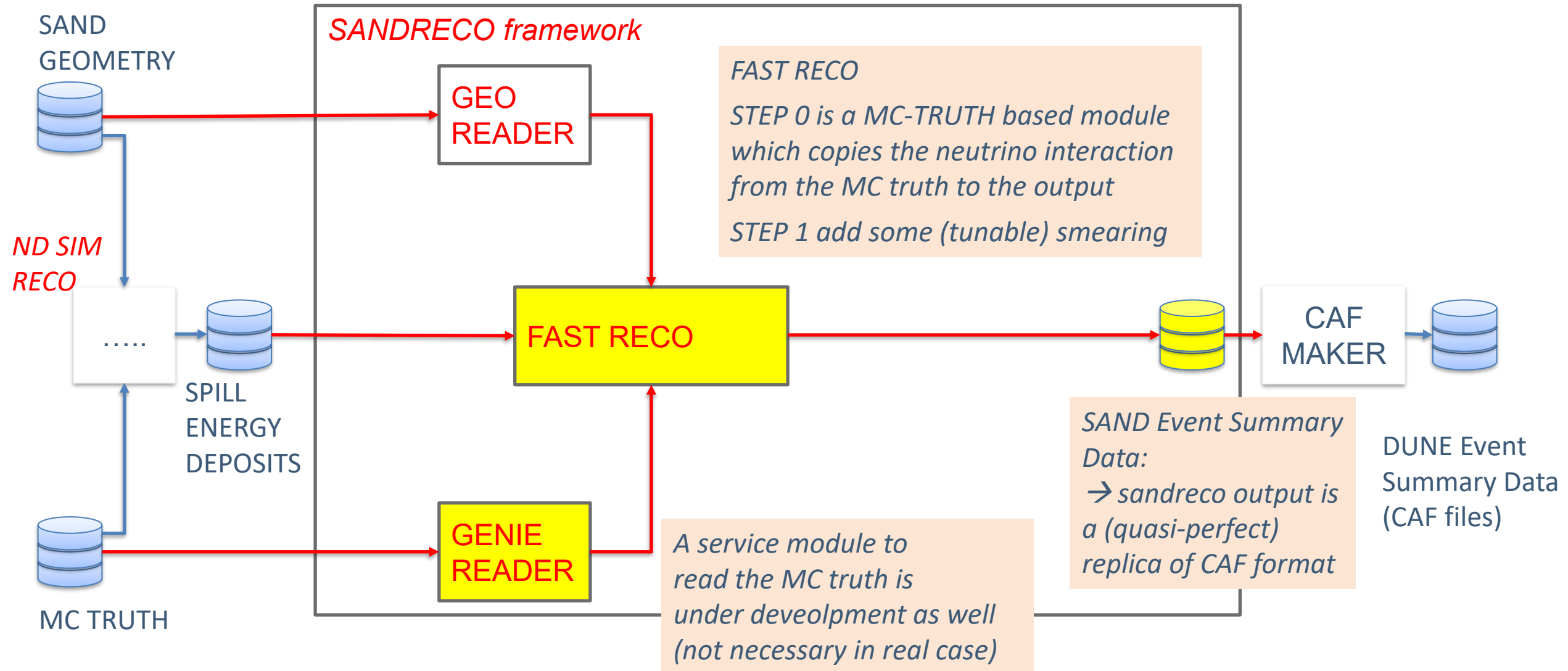
Development and production docker images

- **sand-prod-[cpu/nv/amd]**: Images for production environment. Comes in three versions: **cpu** (CPU-only), **nv** (NVIDIA GPUs), **amd** (AMD GPUs).
- **sand-dev-[cpu/nv/amd]**: Images for development environment that match the production images version. Come with all software installed in /usr/local, including a debug build of ufw and sandreco (main branches). See below for usage.
- GRAIN software (with coded aperture masks) uses GPUs different images for usage with NVIDIA or AMD or CPU-only
- Tested with docker/podman and singularity/apptainer (on CNAF HTC and HPC cluster)
- Instructions for using the dev images are here: <https://baltig.infn.it/dune/sand-ci>

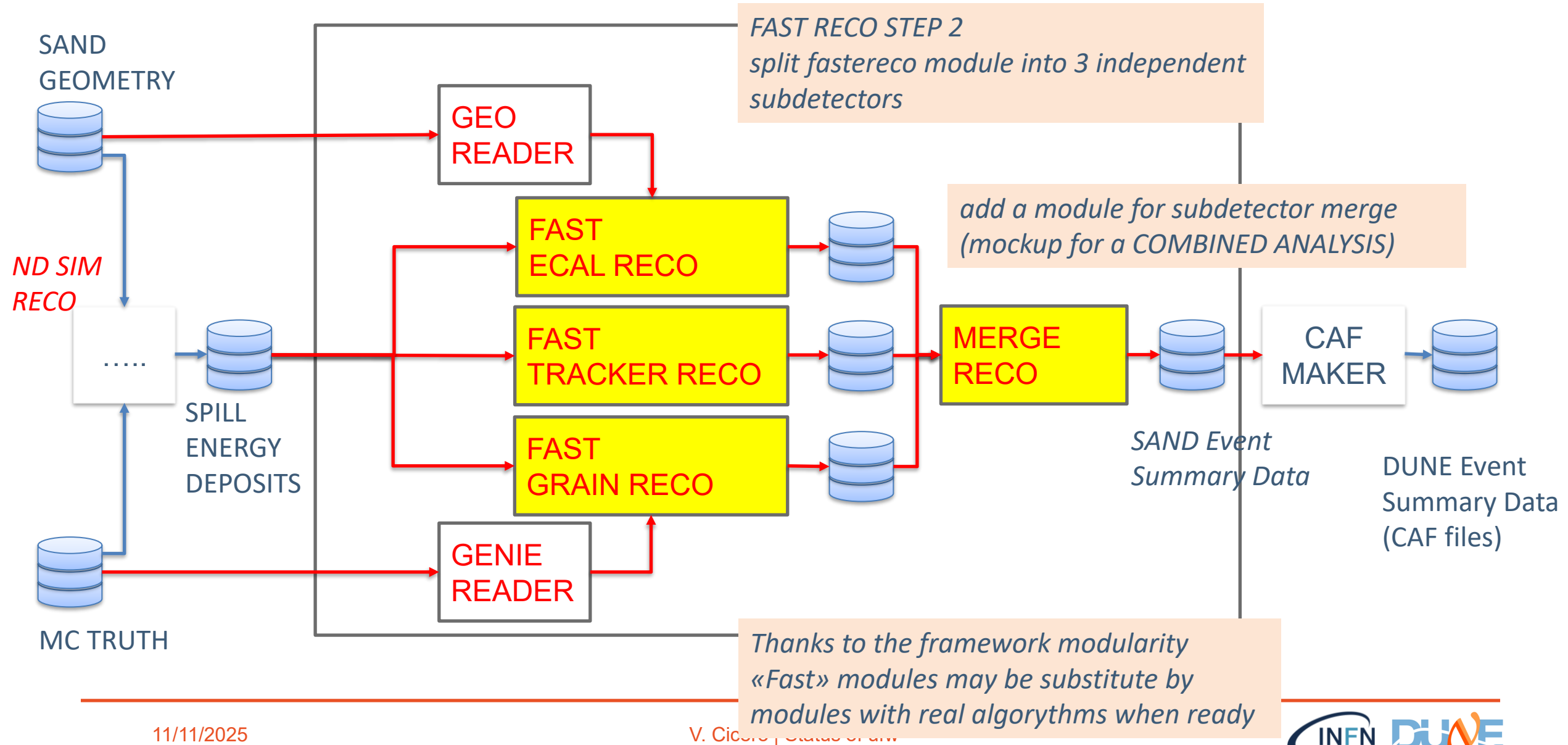
FastReco Module



FastReco Module



FastReco Module



Conclusions

- The software WG decided to adopt a minimal framework while waiting for the DUNE one
- This is a chance to restructure and modularize sandreco
- Some modules are already implemented in ufw, mostly for GRAIN detector
- The implementation of a fast-reco module has high priority to move the physics analysis efforts to CAF level