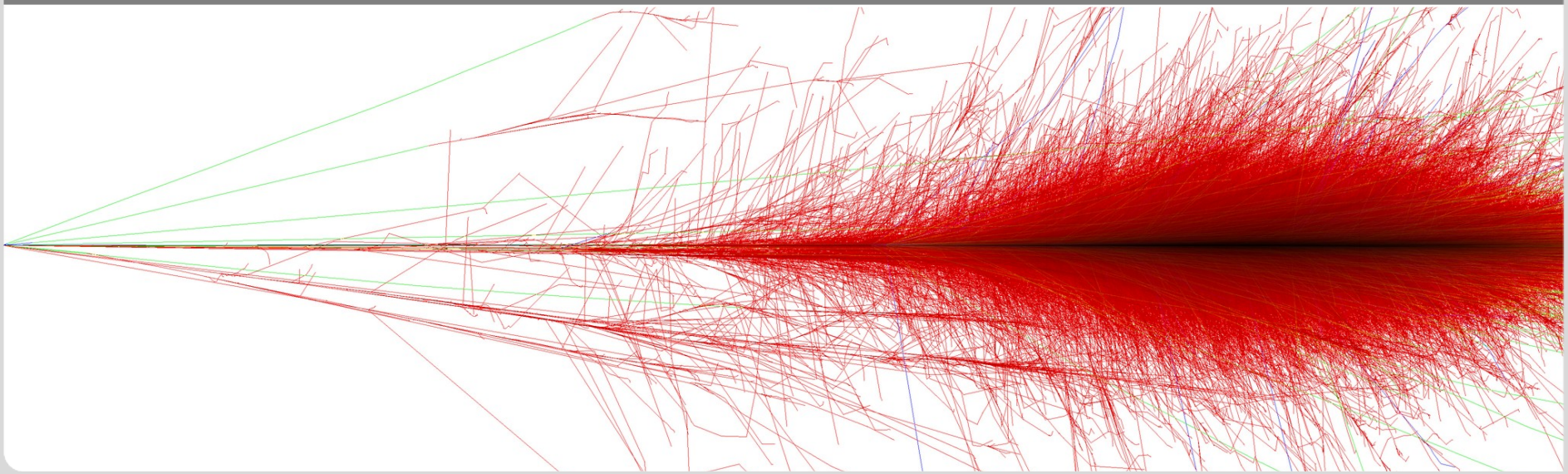
Karlsruher Institut für Technologie

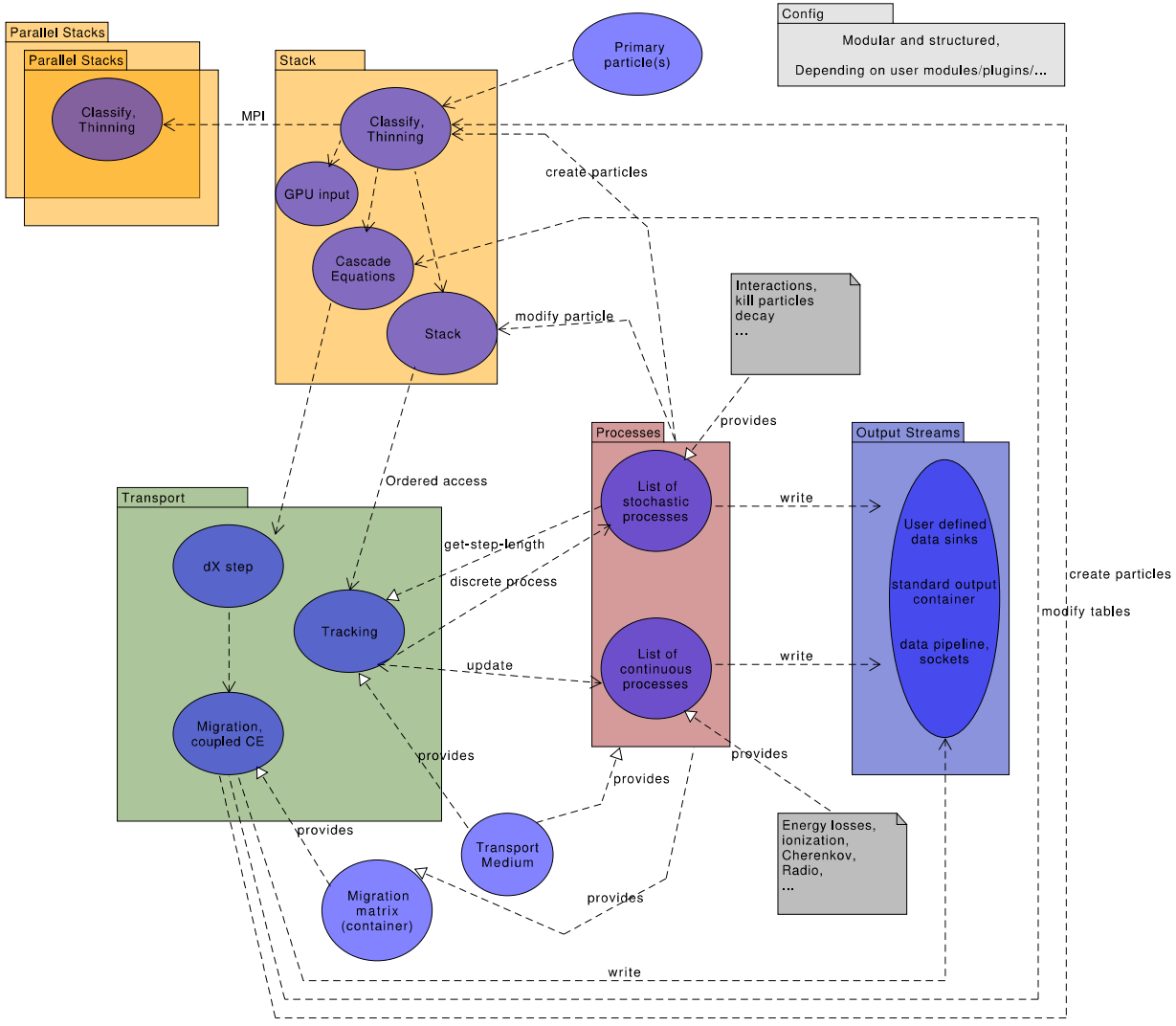
Introduction to first concepts

Maximilian Reininghaus

Institut für Kernphysik



Outline of the design



ngC is just like Karlsruhe...

Some construction sites

- Configuration / steering
- Stack
  - History
- Integration of interaction models
  - Hadronic
  - Electromagnetic
- Geometry (see Darko Veberič's talk)
- File output / storage
- Functionality for developers' convenience:
  - Support for units in code
  - Particle codes & properties

Configuration / steering

- Configuration files

  standardised format like *yaml*, *XML*
- Command-line arguments

  convenient for shell scripts

Unified interface
provided for modules

"Config.Get("PrimaryParticle/Energy")"

- Configuration should be saved in file output for self-documentation & reproducibility!

- Python interface wanted?

  dynamic change of parameters, setting up complicated scenarios

  downside: external depencendy

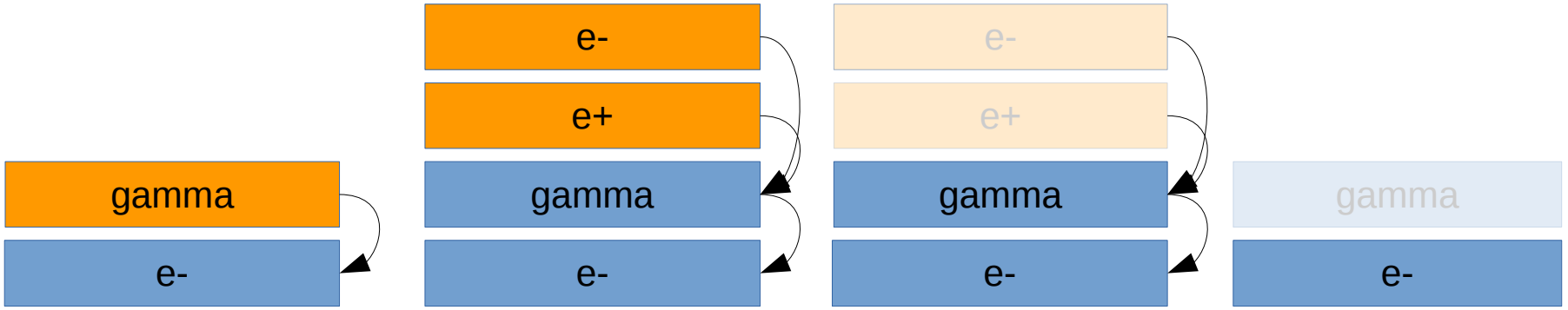
PyBind11 vs. sth. else?

Particles & the stack

- The heart of the simulation
- Crucial to be done *right*
- Stack and model of a particle defined together
- Flexibility
  - easily adding properties (weights, polarisation, history) needed
  - easy filtering (thinning, …), ordering, routing of particles by user routines

History option

- state of the art: *EHISTORY* [Heck, Engel, FZKA-7495 (2009)]
  - saves parent & grandparent particle
  - does not work well with thinning
- new proposal:
  - particles keep a reference to parent particle
  - particles are removed from stack after all children are processed & removed
  - full history is known
  - users have to decide what to save / analyse on-the-fly



Interaction models

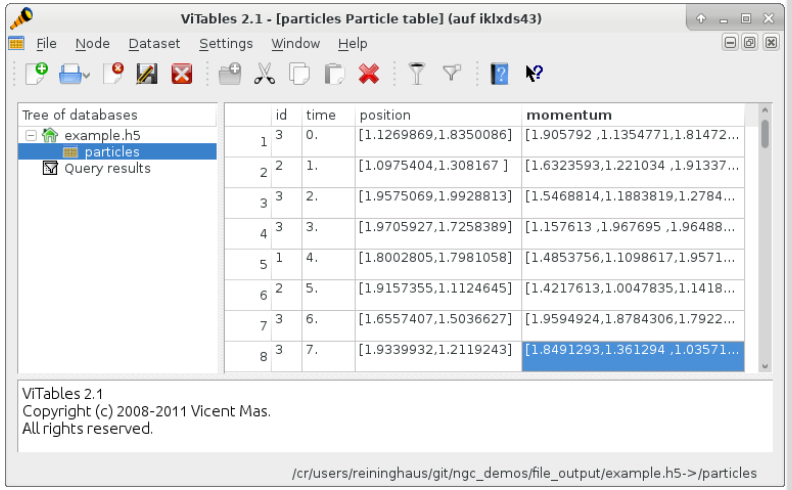
- General definition of interfaces needed, tied to stack interface
- simplistic models for testing: null model, Heitler model
- Hadronic interaction models
  - High energy: SIBYLL, QGSJet, EPOS, DPMJet
  - Low energy: UrQMD, Fluka

  work in progress @ KIT
- PYTHIA 8 for decays (work in progress)
- Electromagnetic interaction models
  - EGS4 from CORSIKA or CONEX, modified for high energies
  - EGS5 / EGSnrc: standalone codes but improvements comp. to EGS4
  - EM cascade code from H.E.S.S.
  - Still needs to be tackled!

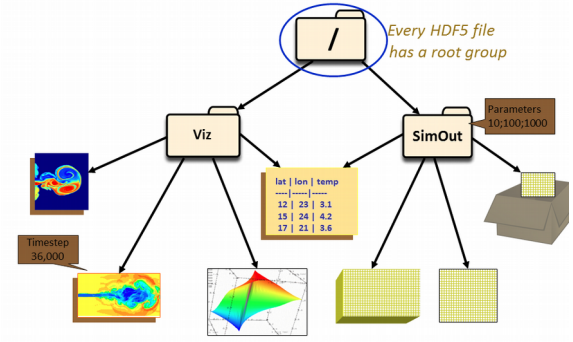
(File) output

Need interfaces for “general” output data

- Particles (with flexible properties)
- n-dim arrays, tables, histograms
- metadata, attributes



- Single file containing everything
- HDF5 / ROOT
  - “filesystem” within a file
  - self-describing
  - on-the-fly compression
  - parallel writing via MPI
  - readers available for your favorite language

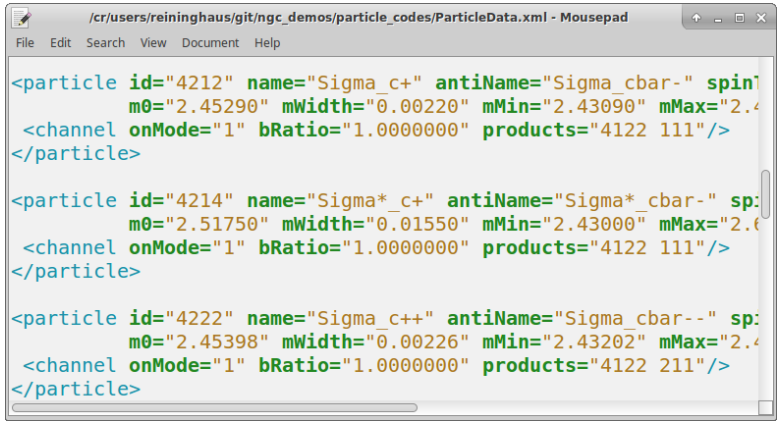


- Legacy DATxxxxxx still needed?
- Different needs / tuning parameters for different communities?

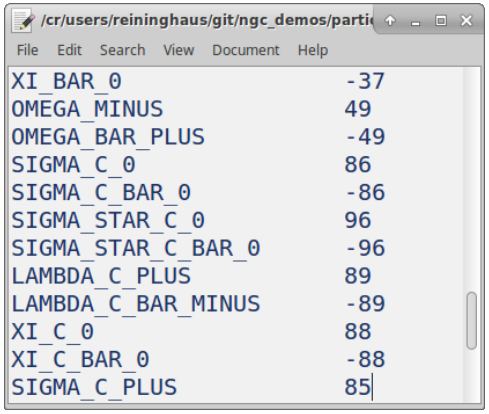
Particle codes & properties

- Particle codes:
  - every interaction model has its own definition
  - frequent conversions must be fast
  - PDG codes are sparse, not useful for lookup tables
  - should be invisible in code, no magic numbers
- Particle properties (mass, lifetime, …):
  - tables need to be maintained / updated
  - already available in machine-readable form
- automated generation of a “database”

Particle codes & properties



PYTHIA 8 ParticleData.xml



SIBYLL

...

enums, mass tables, code conversion

- `auto constexpr getMass(InternalParticleCode::RHO_0)`
- `auto constexpr getPDG(InternalParticleCode::SIGMA_BAR_PLUS)`

nuclei not yet included!

special particles (decayed μ) needed?

Units in code

- Legacy way:

```
double critical_energy = 0.087; // conv.: in GeV
```

- CLHEP (Geant4, Offline):

```
double MeV = 1.0e6; // simplified

double critical_energy = 87 * MeV;

double invalid = critical_energy + 30 * mm;
```

- Boost.Units / PhysUnits (https://github.com/martinmoene/PhysUnits-CT-Cpp11)

```
auto speed = 47_km / 32_min;

auto invalid = speed + 12_cm; // compiler error!
```

no runtime overhead!

Summary

There is a lot of work still to be done!

Comments welcome!