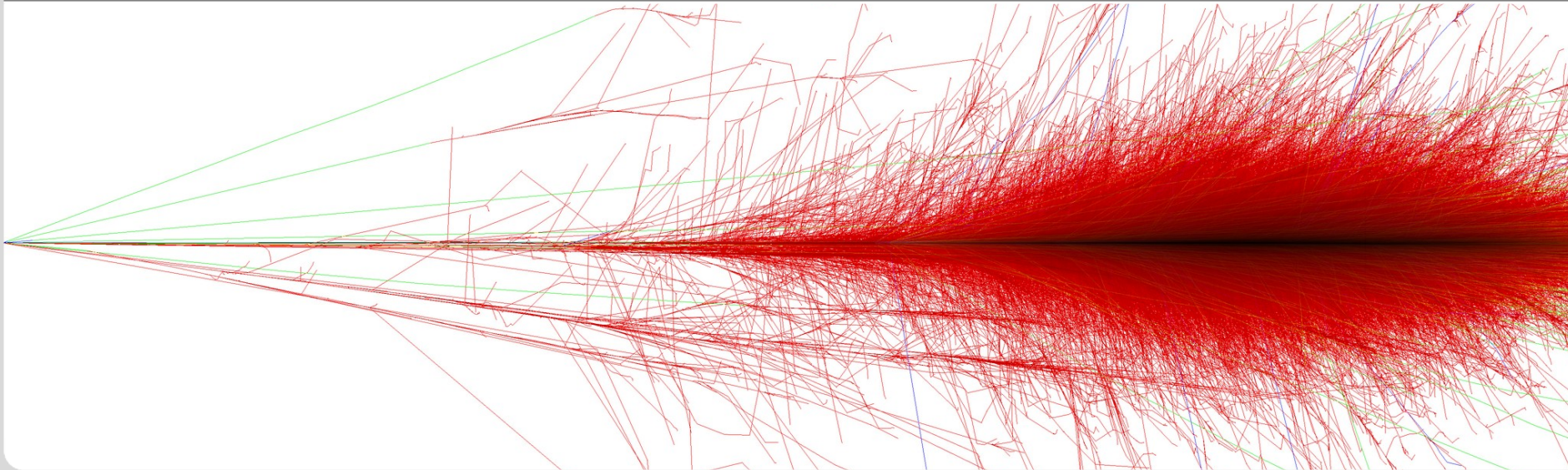




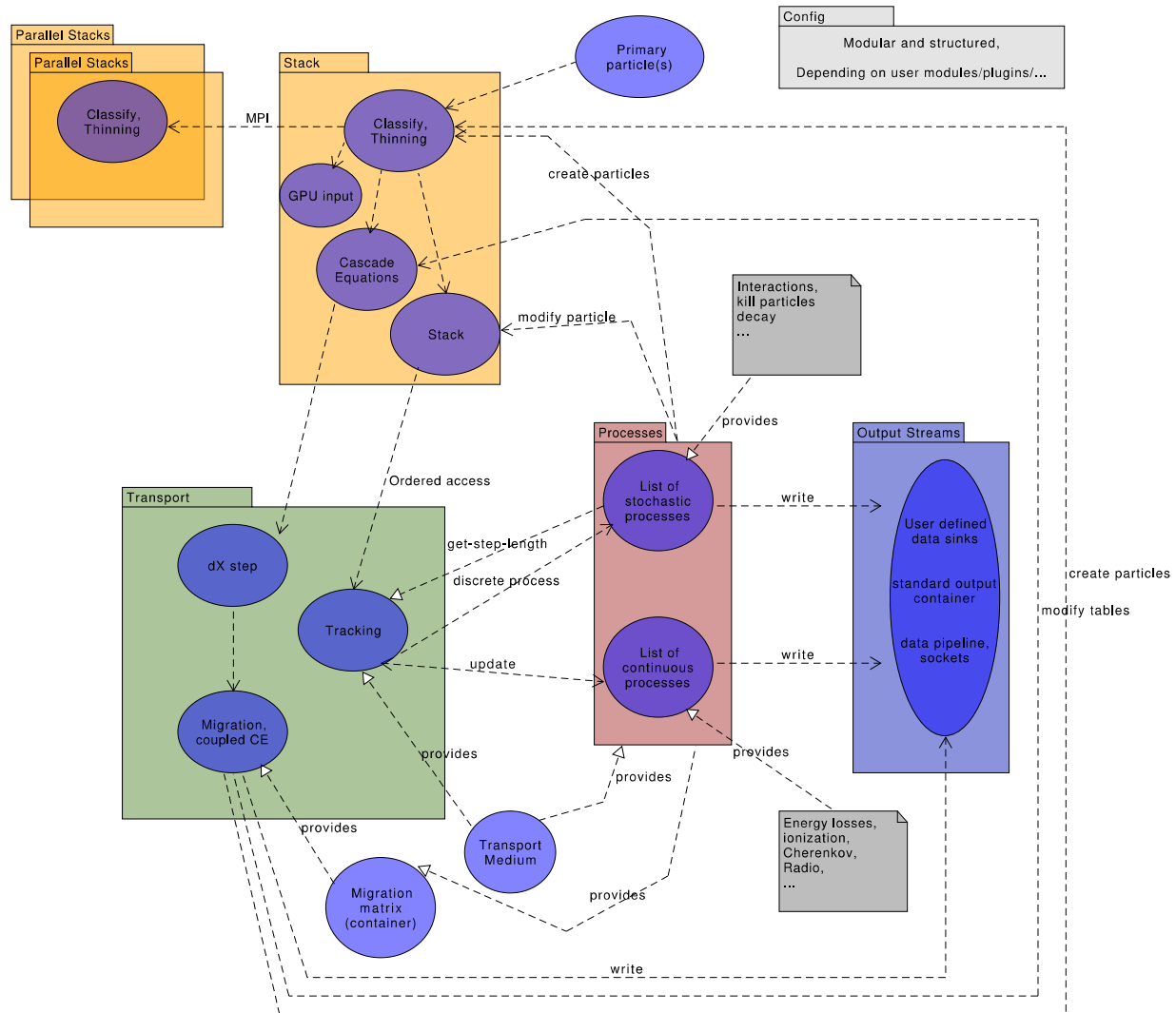
Introduction to first concepts

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 dependendy

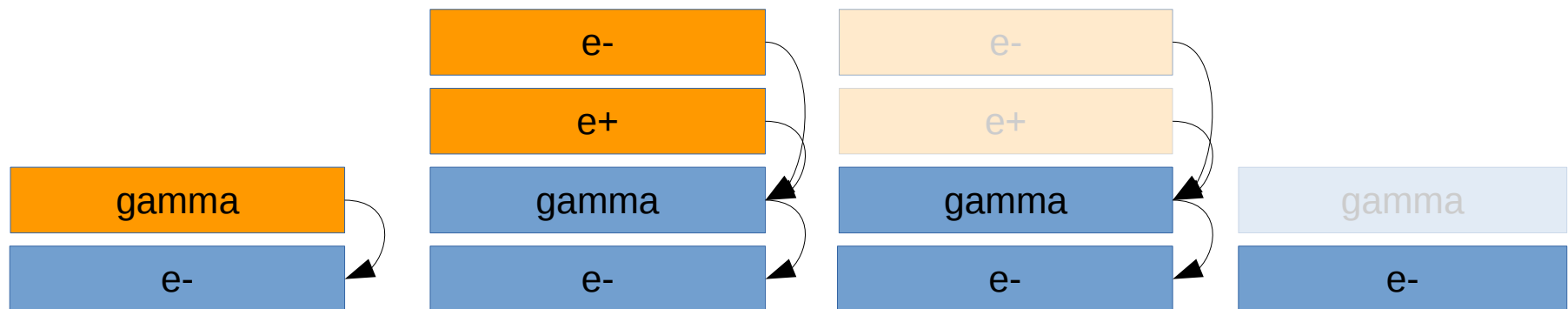
PyBind11 vs. sth. else?

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

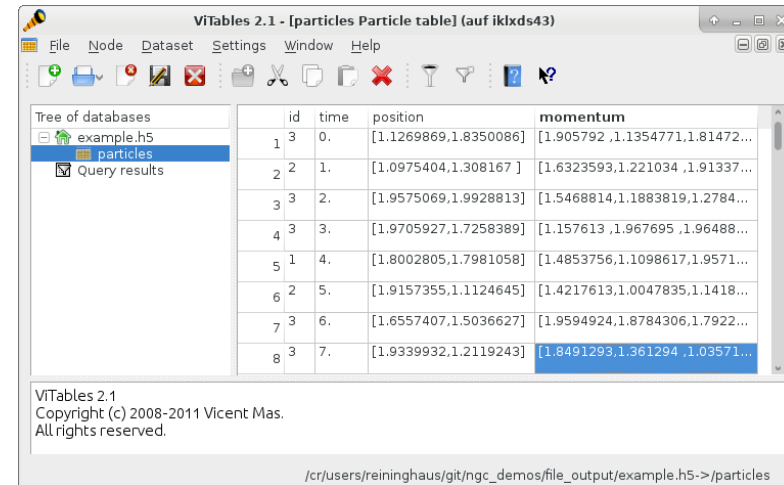
work in progress @ KIT

- 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?

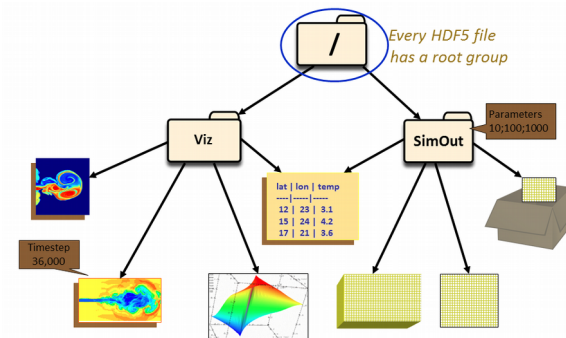


ViTables 2.1 - [particles Particle table] (auf ikfids43)

	id	time	position	momentum
1	3	0.	[1.1269869,1.8350086]	[1.905792 ,1.1354771,1.81472...
2	2	1.	[1.0975404,1.308167]	[1.6323593,1.221034 ,1.91337...
3	3	2.	[1.9575069,1.9928813]	[1.5468814,1.1883819,1.2784...
4	3	3.	[1.9705927,1.7258389]	[1.157613 ,1.967695 ,1.96488...
5	1	4.	[1.8002805,1.7981058]	[1.4853756,1.1098617,1.9571...
6	2	5.	[1.9157355,1.1124645]	[1.4217613,1.0047835,1.1418...
7	3	6.	[1.6557407,1.5036627]	[1.9594924,1.8784306,1.7922...
8	3	7.	[1.9339932,1.2119243]	[1.8491293,1.361294 ,1.03571...

ViTables 2.1
Copyright (c) 2008-2011 Vicent Mas.
All rights reserved.

/cr/users/reininghaus/git/ngc_demos/file_output/example.h5->/particles



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

```

/cr/users/reininghaus/git/ngc_demos/particle_codes/ParticleData.xml - Mousepad
File Edit Search View Document Help

<particle id="4212" name="Sigma_c+" antiName="Sigma_cbar-" spin
m0="2.45290" mWidth="0.00220" mMin="2.43090" mMax="2.4
<channel onMode="1" bRatio="1.0000000" products="4122 111"/>
</particle>

<particle id="4214" name="Sigma*_c+" antiName="Sigma*_cbar-" sp
m0="2.51750" mWidth="0.01550" mMin="2.43000" mMax="2.6
<channel onMode="1" bRatio="1.0000000" products="4122 111"/>
</particle>

<particle id="4222" name="Sigma_c++" antiName="Sigma_cbar--" sp
m0="2.45398" mWidth="0.00226" mMin="2.43202" mMax="2.4
<channel onMode="1" bRatio="1.0000000" products="4122 211"/>
</particle>
  
```

PYTHIA 8 ParticleData.xml

```

/cr/users/reininghaus/git/ngc_demos/parti
File Edit Search View Document Help

XI BAR_0 -37
OMEGA_MINUS 49
OMEGA_BAR_PLUS -49
SIGMA_C_0 86
SIGMA_C_BAR_0 -86
SIGMA_STAR_C_0 96
SIGMA_STAR_C_BAR_0 -96
LAMBDA_C_PLUS 89
LAMBDA_C_BAR_MINUS -89
XI_C_0 88
XI_C_BAR_0 -88
SIGMA_C_PLUS 85
  
```

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!

There is a lot of work still to be done!

Comments welcome!