Software for future BINP HEP experiments

Dmitry Maksimov, Kirill Chilikin, Yury Maslov,
Andrey Suharev, Anastasia Zhadan, Daniil Zhadan

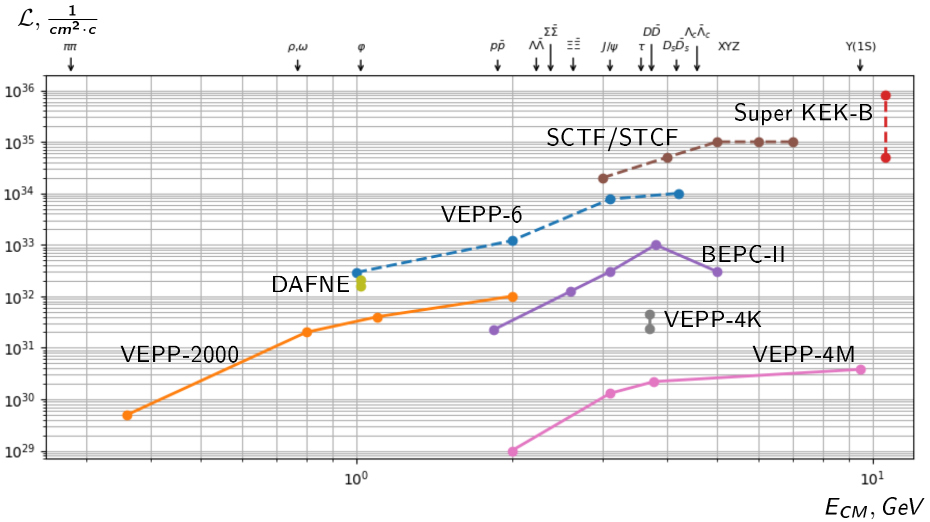
11th International Conference "Distributed Computing and Grid Technologies in Science and Education" (GRID'2025)

Budker Institute of Nuclear Physics, Novosibirsk, Russia

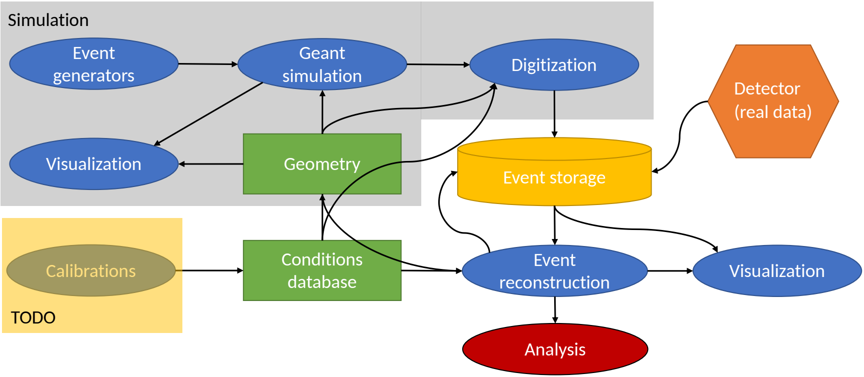
8 July 2025

BINP

Electron-positron colliders world wide



Framework components and data flows



Framework also has:

- build & configuration system,
- release management software.

The Aurora framework features

Initially developped for SCTF project, now in a process of separation between framework itself and experiments specific parts.

- based on Gaudi with influence of FCCSW, Athena and BASF2,
- uses DD4Hep for detector geometry description,
- has its own Fast Simulation (DC, Calorimeter, PID, Mu),
- uses Geant4 for full simulation,
- data persistency based on ROOT and PODIO,
- single source of particle properties,
- random numbers generation infrastructure independent on algorithms execution order,
- conditions database prototype,
- powerful data analysis package.

The Aurora framework management

- build & configuration system inspired by ATLAS Athena,
- `lcgcmake` system to build external packages,
- current computing environment is AlmaLinux 8, gcc13 (C++17), Python3
- next probable computing environment is AlmaLinux 9, gcc14 (C++20), Python3
- software is partially distributed via CVMFS

Event Generators

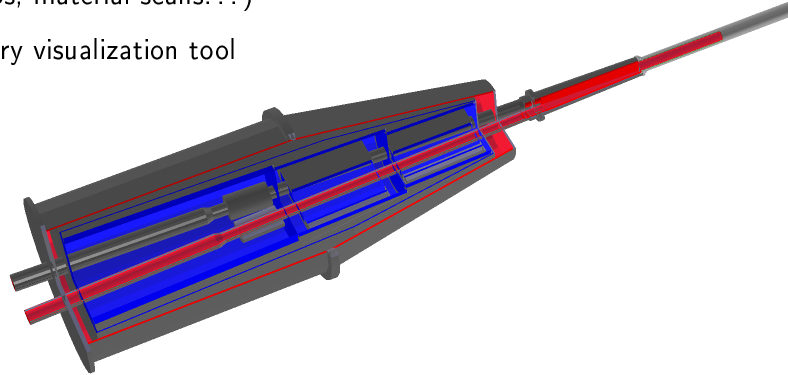
The conventional set of event generators available:

- Exclusive decays of hadrons and tau lepton
  - EvtGen, Tauola, PHOTOS, Pythia
- Inclusive generators for $e^+e^- \to$ hadrons *or* leptons
  - Solution based on KKMC, Pythia and EvtGen
- Technical generators
  - Particle gun, etc

List of generators is extended on demand

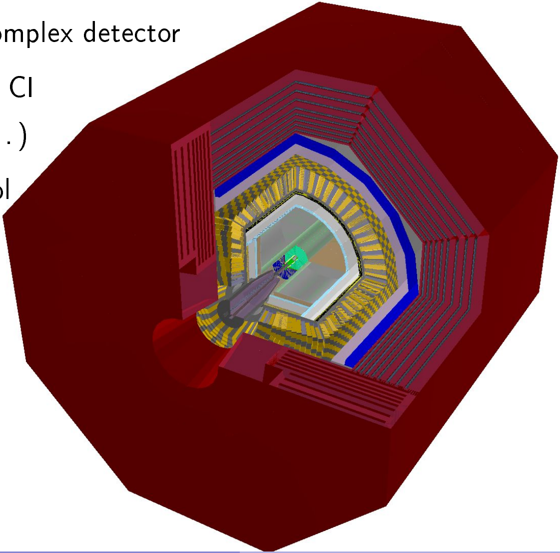
Geometry tools in Aurora

- DD4Hep stuff and some custom volumes implemented
- Enough to describe any complex detector
- Geometry testing tools for CI
  (overlaps, material scans. . .)
- Geometry visualization tool



Geometry tools in Aurora

- DD4Hep stuff and some custom volumes implemented
- Enough to describe any complex detector
- Geometry testing tools for CI (overlaps, material scans. . .)
- Geometry visualization tool

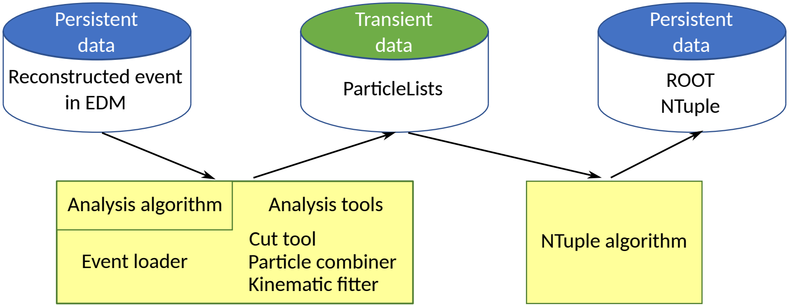


Data storage

- Based on ROOT and PODIO.
- Uses stable PODIO (since 1.0) storage container called `Frame`.
- `Frame` is an universal container capable store EDM and simple non-EDM data, also it is capable to contain another `Frame`.
- `Frame` container owns data stored into it, leads to conflicts with Gaudi storage (owns also), required additional step to resolve this.

Data Analysis package

- Following Belle II recipes and solutions for analysis



- final particles processing
- kinematic cuts
- reconstructuion of decayed particle

- kinematic parameters calculation
- particle selection
- kinematic fit

- save results

Data analysis usage example

Loading services

First of all it is necessary to configure data services:

```
# Particle data.
particle_data = ParticleDataService('ParticleDataService')

# Conditions database.
conditions = ConditionsService('ConditionsSvc',
    testingPayloadLocations=['local_database/database.txt',
                             AuroraConditions.databasePath()]
)
```

- ParticleDataService — is new unified service providing particle properties information based in EvtGen data format.
- ConditionsService — provides experimental environment: beam parameters and magnetic field.

Data analysis usage example

Input data, analysis algorithm

Creating analysis algorithm:

```
# Analysis algorithm.
analysis = AnalysisAlgorithm(
    'Analysis',
    eventLoader=event_loader,
    analysisTools=[cut1, combiner1, extrainfo1, fit1, combiner2, extrainfo2,
                   fit2]
)
```

Analysis algorithm sequentially calls analysis tools, event loader is called before all others.

Data analysis usage example

Event loading and selection

Event loader is configured by:

```
# Event loader.
event_loader = EventLoaderTool(
    'EventLoader',
    particleLists=['pi+', 'K+', 'gamma']
)
```

particleLists property defines lists of particles to be created (loaded). All particles including antiparticles created at this stage without any selection. Selection rules are applied by CutTool later.

```
# Apply selection criteria for kaons.
cut1 = CutTool(
    'Cut1',
    outputParticleList='K+:sel',
    inputParticleLists=['K+'],
    cut='pt␣>␣0.1'
)
```

New list K+:sel contains Kaons from original list K+, which satisfy condition $p_t > 0.1$ ГэВ$/c$.

Data analysis usage example

Particles combinations

Compound particles are created by `ParticleCombinerTool` from earlier created lists of daughter partcles.

```
# Reconstruct pi0 -> gamma gamma.
combiner1 = ParticleCombinerTool(
    'Combiner1',
    decayString='pi0 -> gamma gamma',
    cut='0.120 < M < 0.150'
)
```

Created particles are subject to specified selection criteria.
J/ψ example:

```
# Reconstruct J/psi -> K+ K- pi0.
combiner2 = ParticleCombinerTool(
    'Combiner2',
    decayString='J/psi:channel1 -> K+:sel K-:sel pi0',
    cut='2.9 < M < 3.3 and p < 0.6'
)
```

Data analysis usage example

ExtraInfo and kinematic fits

ExtraInfoTool saves specified variables of particles list to extra info area, for example combined particle mass before fit could be saved.

```
# Save pi0 mass before fit.
extrainfo1 = ExtraInfoTool(
    'ExtraInfo1',
    particleList='pi0',
    extraInfo=[['M', 'M']]
)
```

First argument is ExtraInfo name, second is variable name.

KinematicFitterTool makes a fit of mass of particles from specified list.

```
# Perform mass fit for pi0.
fit1 = KinematicFitterTool(
    'Fit1',
    particleList='pi0'
)
```

Data analysis usage example

Save results

Particles after analysis stage could be saved to file by `NtupleAlg`. Variables to save for each particle could be specified.

```
# Define aliases.
aliases = [['M_before_fit', 'extraInfo(M)'],
           ['p_before_fit', 'extraInfo(p)']]

# Create ntuple.
ntuple1 = NtupleAlg('NtupleAlg1',
    listName='J/psi:channel1',
    fileName='scttuple/jpsi1',
    tupleTitle='',
    vars=[['M_before_fit', 'p_before_fit', 'M', ''],
          ['E', 'px', 'py', 'pz', 'px_mc', 'py_mc', 'pz_mc', 'pidkpi',
           'J/psi -> ^K+ ^K- pi0'],
          ['E', 'px', 'py', 'pz', 'M_before_fit', 'M', 'J/psi -> K+ K- ^pi0'],
          ['px', 'py', 'pz', 'px_mc', 'py_mc', 'pz_mc',
           'J/psi -> K+ K- [ pi0 -> ^gamma ^gamma ]']],
    aliases=aliases
)
NTupleSvc(Output = [
    "scttuple DATAFILE='jpsi_kpkmpiz_reconstructed.root' OPT='NEW' TYP='ROOT'"
])
```

It is possible to use several `NtupleAlgs` for different reconstructed particles.

Conclusions

The Aurora framework contains minimally required set of components for detector development:

- set of primary event generators,
- parameterized simulation,
- detector geometry description tools,
- full Geant4-based simulation,
- analysis and job configuration tools,
- test and service tools.

All described software is targeted to Aurora 3.0.0 release

Thank you for your attention

Backup

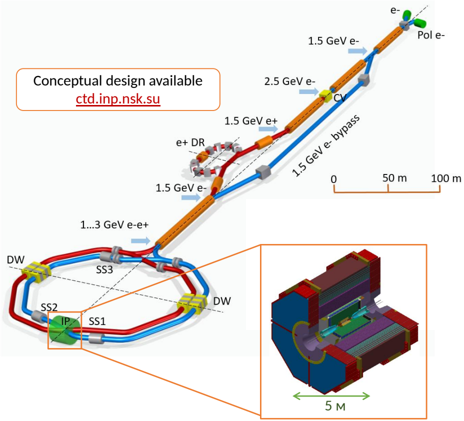
Further software development

The nearest goals for the software development are:

- implementation of digitization modules for all subsystems
- further reconstruction improvements, including adoption of some high-level tools, i. e. track finding,
- improvement of detector and event visualization tools. The underlying DDEve has been not actively developed recently, so this is an area of possible backward contribution to DD4Hep
- distribution of the software via CvmFS

SCT Experiment overview

- Precision experiments with tau lepton and charmed hadrons, and search for BSM phenomena
- Electron-positron collider
  - Beam energy varying between 1.5 and 3.5 GeV
  - Luminosity $\mathcal{L} = 10^{35} cm{-}2s{-}1$ @ 2 GeV
  - Longitudinal polarization of the e^- beams
- Universal particle detector
  - Tracking system
  - Crystal electromagnetic calorimeter
  - Particle identification system



Detector overview

Requirements:

- Trigger rate up to 300 kHz
- $10^4 cm^{-2}s^{-1}$ tracks at $R \leq 20$ cm
- $\sigma_p/p \leq 0.4\%$ at 1 GeV/c
- Good π^0/γ separation, $E_\gamma = 10 - 3000$ MeV, $\sigma_E \leq 1.8\%$ at 1 GeV
- Dedicated PID system
  - $\frac{dE}{dx} < 7\%$,
  - μ/π separation up to 1.5 GeV/c,
  - π/K separation up to 3.0 GeV/c.
- Minimal CP detection asymmetry