DD4hep Status

HEP detector description supporting the full experiment life cycle

M. Frank, F. Gaede, M. Petric, A. Sailer

April 26th, 2018

AIDA 2020 Annual Meeting 2018, Bologna

Markus Frank/CERN
• Motivation and Goals
  => Introduction / Reminders

• Simulation

• Conditions and Alignment

• Conditions use case study

• Miscellaneous

• Summary
Motivation and Goal

- **Develop a detector description**
  - For the full experiment life cycle
    - detector concept development, optimization
    - detector construction and operation
    - “Anticipate the unforeseen”
  - **Consistent description, with single source, which supports**
    - simulation, reconstruction, analysis
  - **Full description, including**
    - Geometry, readout, alignment, calibration etc.
What is Detector Description?

- Description of a tree-like hierarchy of “detector elements”
  - Subdetectors or parts of subdetectors

- Detector Element describes
  - Geometry
  - Environmental conditions
  - Properties required to process event data
  - Optionally: experiment, sub-detector or activity specific data
DD4Hep - The Big Picture
Saga in 5 Episodes: Sub-packages

- **DD4hep** – basics/core \(^{(1)}\)
- **DDG4** – Simulation using Geant4 \(^{(1)}\)
- **DDRec** – Reconstruction supp. \(^{(2)}\)
- **DDCond** – Detector conditions \(^{(1,3)}\)
- **DDAlign** – Alignment support \(^{(1,3)}\)

\(^{(1)}\) Bug-fixes and maintenance
\(^{(2)}\) See presentation of F. Gaede (WP3, Task 3.6)
\(^{(3)}\) Work since start of AIDA\(^{2020}\)
• Motivation and Goals
  => Introduction / Reminders

• Simulation

• Conditions and Alignment

• Conditions use case study

• Miscellaneous

• Summary
Simulation: DDG4

- Simulation = Geometry + Detector response + Physics
- Mature status
  - Eventual bug fixes, smaller improvements
  - Phase of constant re-validation
- CLICdp implementation was presented at CHEP 2017 (M. Petric)
- Deliverable 3.6 Submitted in time March 2018
DDG4 in Production

• Fully deployed for CLICdp in DIRAC
  – For every detector study (now ~14) centrally generated:
    ≈ 200k $Z^0 \rightarrow qq$ (uds)
    ≈ 20k Signal events ($Z^0 \rightarrow tt$, $Z^0 \rightarrow H+x$, etc.)

  + private productions by team members
  – Used as input for ~ 20 physics analyses and detector studies

• Eagerly awaiting ILC mass production
  – Should have started beginning of the year
  – Client confirmation that concept works at large scale
- Motivation and Goals
- Simulation
- **Conditions and Alignments**
  => DDCond, DDAign
- Conditions use Case Study
- Miscellaneous
- Summary
DDCond: Conditions Data

- Time dependent data necessary to process the detector response [of particle collisions]
  - slowly changing: every run $O(1\text{h})$, lumi section $O(10\text{min})$
  - multiple conditions change in batches: require discipline
  - conditions may be the result of computation(s)

- DDCond deals with the management of these data
  - Efficient and fast, if used according to design ideas
  - Manage resources
  - Supports multi threading by design
    Well define locking points
  - Cache where necessary but no more

- But all this got presented already last time ...
DDCond: The Data Cache

ConditionsManager

IOV type "fill"

<table>
<thead>
<tr>
<th>Cond 1</th>
<th>Cond 2</th>
<th>Cond N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill 1000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Door keeper

Manage different data types

IOV type "run"

<table>
<thead>
<tr>
<th>Cond 1</th>
<th>Cond 2</th>
<th>Cond N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 127895</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IOV type "lumi-section"

<table>
<thead>
<tr>
<th>Cond 1</th>
<th>Cond 2</th>
<th>Cond N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 127896</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IOV type "YEAR"

<table>
<thead>
<tr>
<th>Cond 1</th>
<th>Cond 2</th>
<th>Cond N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run NNNN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data Stores:

-- Organized by IOV
-- Provide and manage data
-- Cache for multiple configurations
-- Relatively static
-- Structure hidden

=> Only data cache
=> Once loaded data stay unless explicitly dropped
DDCond: The Data Cache

- Conditions sub-pools are read-only (gentleman's agreement)
- Only when sub-pools enter or leave the scene the container must be locked
- MT Detail: sub-pools referenced by shared_ptr: => usable beyond registration
DDCond: IOV Data Projection

Condition pool managing each one unique IOV
-- ALL conditions in one pool have the identical IOV
-- IOV Type manage all pools with the same type:
  run, fill, epoch, year, ...

Client

1) creates

ConditionsSlice

2) inserts

Dependency Rules

3) inserts

Load Addresses

4) requests load / computation (slice, run=127896)

UserPool

5) Access consistent conditions

Run 127895

Run 127896

IOV type "run"

Cond 1

Cond 2

Cond N

Cond 1

Cond 2

Cond N

ConditionsManager

Slice management is up to client:
- Share slices between threads
- Populate new slices independently
DDCond: Status

- Described functionality is implemented
  - Tested with LHCb conditions data

- Accomplished implementation deliverable D3.2, Submitted February 2018
  - Includes alignment support to handle geometry imperfections

- Local Alignments are derived conditions
  - Convert Δ parameters (translation, rotation, pivot-point) to transformations to world or reference point
DDAlign: Global and Local Alignments

- **Global alignment corrections**
  - Physically alters geometry with intrinsic support by ROOT
  - Transition between geometries by construction not multi-threaded
  - Possibility to simulate misaligned geometries

- **Local alignment corrections**
  - Geometry stays intact (either ideal or globally aligned)
  - Multi-threading supported, multiple versions
  - Local alignment corrections are conditions
  - Provide matrices from ideal geometry to world e.g. to adjust hit positions

- **Both supported and implemented**
Local Alignment: Apply $\Delta$ - Parameters

- Trickle-up the hierarchy and compute the matrices the most effective way with re-use of intermediate results
- Math verified (and corrected) by C. Burr
• Motivation and Goals

• Simulation

• Conditions and Alignments

• Conditions Use Case Study
  
  VELO (upgrade) detector elements

• Miscellaneous

• Summary
Get Fingers Dirty: LHCb Velo Detector

- People want to see “Detector elements”
  - Fully functional description of parts of the detector
    - Long term valid stuff (structure)
    - Short lived quantities (temperature, alignment, …)
- A “natural” aggregation would be similar to:

- Intuitive, but not good: violates multi-threading.
  (DetElements exist only in one instance)
Real World Use Case: LHCb Velo Detector

- **Chosen solution:**
  - Use IOV dependent projection for event processing
    - This is our new “detector element”
    - Keeps reference to the not changing properties
  - Dress with facade to provide required functionality(ies)
Real World Use Case: LHCb Velo Detector

- Structures are build using the derived conditions callback mechanism
  - Static part: once only
  - IOV dependent part: when not in pools
    Also fills link to static information

- Since conditions in existing pools still can be shared while preparing new IOV depending conditions
  - No locking strategy necessary

- Alignment computation incorporated
  - Reminder: alignments must be computed ‘en block’ for an efficient computation

- Common activity with WP3 Task 3.3 (C.Burr et al.)
- Motivation and Goals
- Simulation
- Conditions and Alignment
- Conditions use case study
- Miscellaneous
  
  stake holders, documentation
- Summary
Toolkit Users

Increasing interest in the HEP community

- ILC       F. Gaede et al.
- CLICdp    A. Sailer et al.
- SiD       W. Armstrong
- FCC-eh    P. Kostka et al.
- FCC-hh    A. Salzburger et al.
- FCC-ee    O. Viazlo (CLD design), N. Alipour, G. Voutsinas
- CMS       Evaluation for upgrade started (202x) (I.Osborne et al.)
- LHCb      Evaluation for upgrade started (2019) (B.Couturier et al.)
- CALICE    Calorimeter R&D, started
- EIC       Evaluation considered/started
Multiple Input Sources (Update)

Compact description \( \text{xml} \) \rightarrow Detector constructors \( \text{c++} \)

Conditions DB \rightarrow DDDB converter \( \text{c++} \)

CMS XML geometry \rightarrow DDCMS converter \( \text{c++} \)

CAD drawing \rightarrow CAD converter \( \text{c++} \)

Generic Detector Description Model

Based on ROOT TGeo

(Future)
PR-Plot: CMS Trackers
PR Plot: LHCb Detector of RunI and RunII
PR Plot: FCC Design Study
Miscellaneous

- **New comprehensive web-site**
  - [http://dd4hep.cern.ch](http://dd4hep.cern.ch)
  - Doxygen updates bound to git mergers: doc up to date

- **User Manuals improved but not perfect**
  - DDCond manual was updated
  - DDAAlign user manual still to be largely rewritten

- **Things are not entirely stable and in the past for sure were not**
  - It is difficult to document a moving target
  - Concrete deployment examples only arise
    Documentation with concrete running use cases is simpler
• Motivation and Goals
• Simulation
• Conditions support
• Alignment Support
• Miscellaneous
• Summary
Summary

- **The DD4hep core the sub-packages are at a quite mature state**
  - Increasing interest from the community

- **Support for alignment handling**
  In collaboration with WP3 / Task 3.3, C.Burr et al.

- **It is now time to think about guidelines for concrete deployment strategies based on the toolkit**
  - **Example: LHCb VELO**

- **Documentation is better, but still needs improvement**
  - There is hope. Documentation target getting more stable
Questions and Answers
Backup
Implementation: Geometry

Subdetector Hierarchy (Tree)

- Detectors
  - DetectorElement
    - children 1..n
    - detector: 1
    - placements: 0…1

- PlacedVolume [TGeoNode]
  - [TGeoMatrix]
    - transform: 1

- LogicalVolume
  - Envelope [TGeoShape]
    - [TGeoBox] [TGeoCone] ... [TGeoTube]

Subdetector status (conditions)

- Alignment
- Conditions
- Readout
- Visualization
- Segmentation

GDML content

Geometry
**Views & Extensions: Users Customize Functionality**

DD4hep is based on handles to data

- Clients only use the handles
- Possibility of many views based on the same DE data
  - Associate different behavior to the same data
  - Views consistent by construction
  - User data according to needs
- Be prudent: blessing or curse
  - User data: common knowledge
  - No one fits it all solution
  - Freedom is also to not do everything what somehow looks possible
Example of a DDG4 Action Sequence: Event Overlay with Features

- Combine simple and reusable modules
- Input module
- Any data format
- Primary vertex smearing
- Primary vertex boost
- Common: initialization, final merge

Start Detector Simulation