Bergen pCT Software - Structure discussion

Collecting the information on a discussion page on the wiki

https://wiki.uib.no/pct/index.php/Software_tructure#Software_Modules

Two main points to clarify:

1. Content
   - Software packages in use
   - Data to be exchanged, interface descriptions
   - I/O formats and data adaptors
2. Usage
   - Repositories
   - Build tool

Build tool/ Packaging system

What the build tool must do:

- Definition of software modules
- Dependency definition, internal and external dependencies
- Build instructions
- Separation of tool and definitions

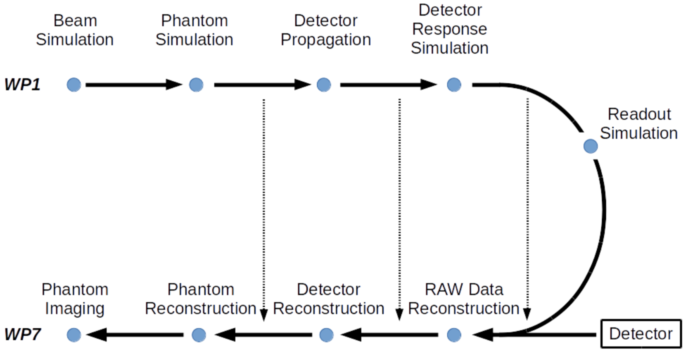
What the build tool can support:

- Building of subset of the software stack
- Mixing of compiled and pre-compiled packages

Examples:

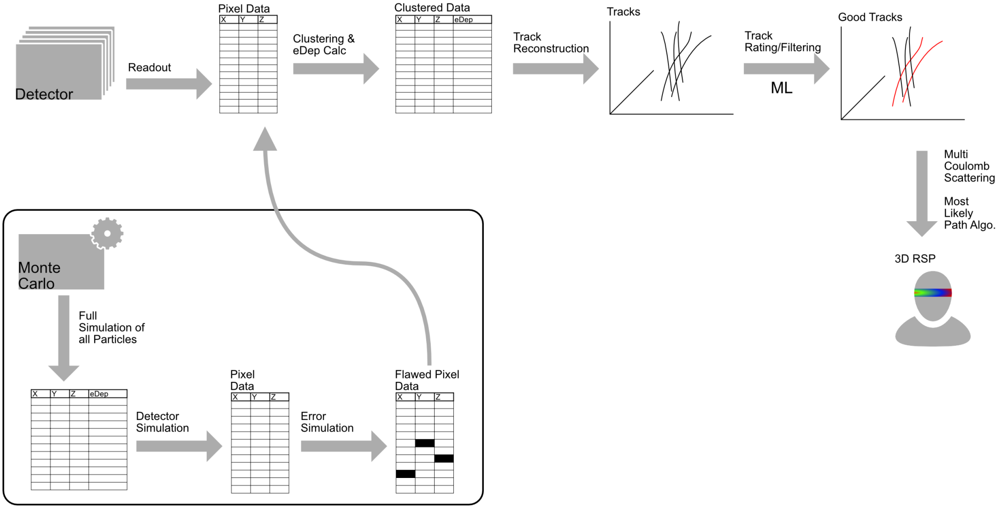
- Spack https://spack.io/
- Specific and very comprehensive build tool for Alice: https://github.com/alisw/alibuild

Bergen pCT Software - Task sequence



- Task sequence to be split into separate packages
- A data model ensures exchange between packages
- Shortcuts of the cycle can be used to test tasks individually

Bergen pCT Software - Different perspective



Detector

Readout

Pixel Data

X Y Z

Clustering & eDep Calc

Clustered Data

X Y Z eDep

Track Reconstruction

Tracks

Track Rating/Filtering

ML

Good Tracks

Multi Coulomb Scattering

Most Likely Path Algo.

3D RSP

Monte Carlo

Full Simulation of all Particles

X Y Z eDep

Detector Simulation

Pixel Data

X Y Z

Error Simulation

Flawed Pixel Data

X Y Z

Software modules

- Simulation
  - Beam Simulation
  - Phantom Simulation
  - Detector Propagation
- Detector Response Simulation
- Readout Simulation
- Readout and Raw Data reconstruction
  - pct-online
- Detector Reconstruction
  - Clustering and Tracking
- Phantom reconstruction
- Imaging

Software packages in use

- Gate setups
- Baylar RSP code
- DTC Tracking https://github.com/HelgeEgil/DigitalTrackingCalorimeterToolkit
- Jarle https://github.com/JarleSoelie

Towards a common data model

Guidelines

- Simple structures to avoid serialization
- Meta data descibing the data
- Usable from both C++ and Python and other scripting languages
- Mostly header definitions with a few helpers, e.g. std I/O operators

Metadata

- Description of data

Integrating CoDiPack

Some preliminary performance measurements, using the simple test program calculating x^2

```
--------------------------------------------------
Benchmark           Time             CPU Iterations
--------------------------------------------------
BM_double           3 ns           3 ns  254732697
BM_forward        137 ns         137 ns    5056067
BM_reverse     199352 ns      199370 ns      10000

--------------------------------------------------
Benchmark           Time             CPU Iterations
--------------------------------------------------
BM_double           3 ns           3 ns  268545786
BM_forward        140 ns         140 ns    5079251
BM_reverse        350 ns         350 ns    2011288
```

If we switch on compiler opimization:

```
--------------------------------------------------
Benchmark           Time             CPU Iterations
--------------------------------------------------
BM_double           0 ns           0 ns 1000000000
BM_forward          0 ns           0 ns 1000000000
BM_reverse         45 ns          45 ns   15517705
```

Integrating CoDiPack

Some preliminary performance measurements:

```
--------------------------------------------------    100000 double operations
Benchmark           Time             CPU Iterations
--------------------------------------------------
BM_double           0 ns            0 ns 1000000000
BM_forward       2718 ns         2717 ns     257242
BM_reverse      20826 ns        20821 ns      33672

--------------------------------------------------   1000000 double operations
Benchmark           Time             CPU Iterations
--------------------------------------------------
BM_double           0 ns            0 ns 1000000000
BM_forward      27240 ns        27236 ns      25579
BM_reverse     198284 ns       198257 ns       3483

--------------------------------------------------  10000000 double operations
Benchmark           Time             CPU Iterations
--------------------------------------------------
BM_double           0 ns            0 ns 1000000000
BM_forward    2714844 ns      2714417 ns        254
BM_reverse   19741054 ns     19737783 ns         35
```

Integrating CoDiPack - what next?

- Maybe I'm doing something wrong with the evaluation of performance impact
- The current numbers make it impossible to apply the technique to a full Geant4/Gate simulation
- We should focus on a smaller code module we have full control over, e.g. the tracking algorithm
- All pct code can only run optionally with CoDiPack

Integrating CoDiPack - support in the data model

- Define a data type to be used within data model library which can transparently map to bare types or CoDiPack types
- Some glue code to avoid tape initialization of every variable
- A CoDiPack sandbox as separate module to run pct code and frees the code from setting up tapes and analysis
- Everything chosen on the level of build system of the data model library, all other modules should be agnostic of CoDiPack as long as they use the pct data model library
- Deploy static code analysis to avoid that the fundamental types are accidentally used in code