Status of CEPC Software Framework

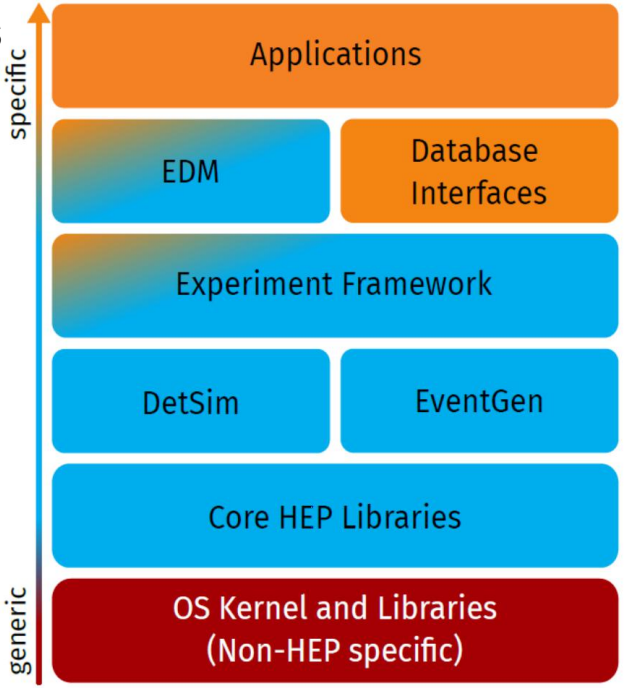
Teng Li
on behalf of the CEPC software framework working group
2022.5.25

Outline

- Introduction
- Overview of CEPCSW
- CEPCSW core components
- Progress and plans
- Summary

Key4hep

- HEP software usually consist of lots of applications
  - Application layer of modules / algorithms /processors performing physics task (*PandoraPFA, FastJet, ACTS,...*)
  - Data access and representation layer including EDM
  - Experiment core orchestration layer (*Gaudi, Marlin, ...*)
  - Specific components reused by many experiments (*DD4hep, Delphes, Pythia,...*)
  - Commonly used HEP core libraries (*ROOT, Geant4, CLHEP, ...*)
  - Commonly used tools and libraries (*Python, CMake, boost, ...*)

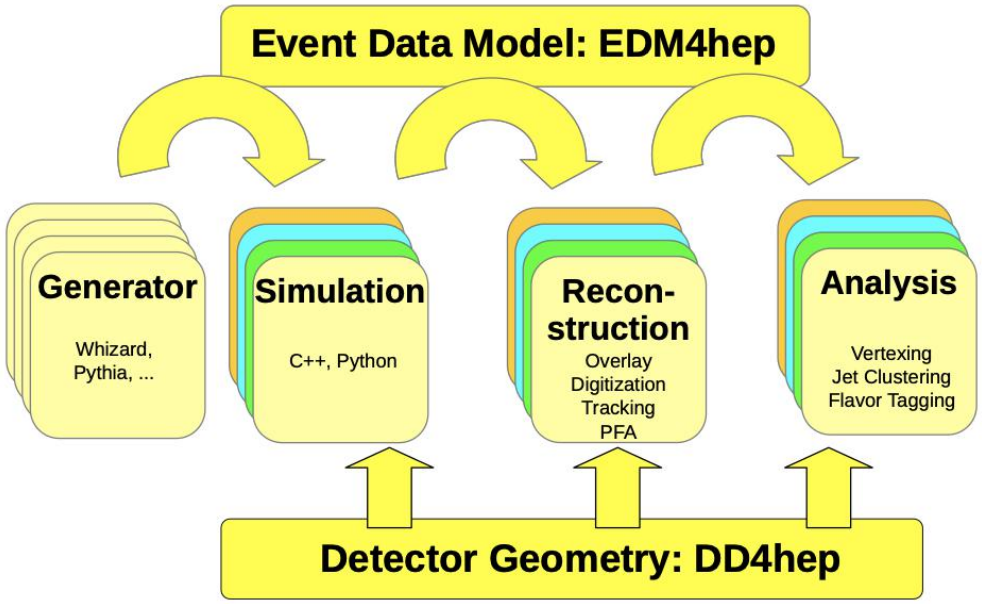


Thomas Madlener, Epiphany Conference 2021

- Turnkey software for future colliders
  - An agreement at Future Collider Software Workshop (Bologna, June 2019)
  - Software components sharing between different experiments
    - CPEC, CLIC, FCC, ILC, STCF

Status of Key4hep

- https://github.com/key4hep
- Main ingredients
  - Event Data Model: EDM4hep, based on LCIO and FCC-EDM; podio
  - Geometry Information: DD4hep
  - Framework: Gaudi
  - Packaging and deployment: Spack
- Key4hep is under rapid development



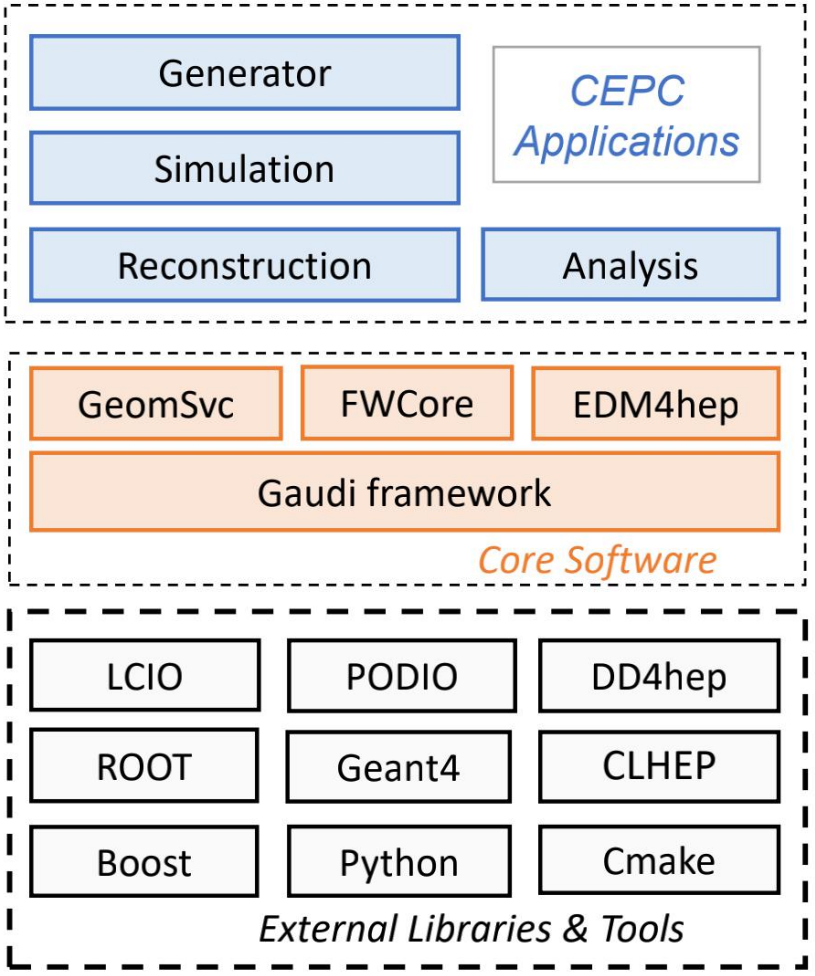
André Sailer, etc. , CHEP2021

k4FWCore	EDM4hep	key4hep-doc	key4hep-spack	EDM4hep-utils
k4MarlinWrapper	k4Clue	k4LCIOReader	k4SimDelphes	k4Analysis
k4EDM4hep2LcioConv	key4hep-validation	k4ActsTracking	key4DCMTSim	k4Pandora

- Weekly meetings: https://indico.cern.ch/category/11461

CEPCSW

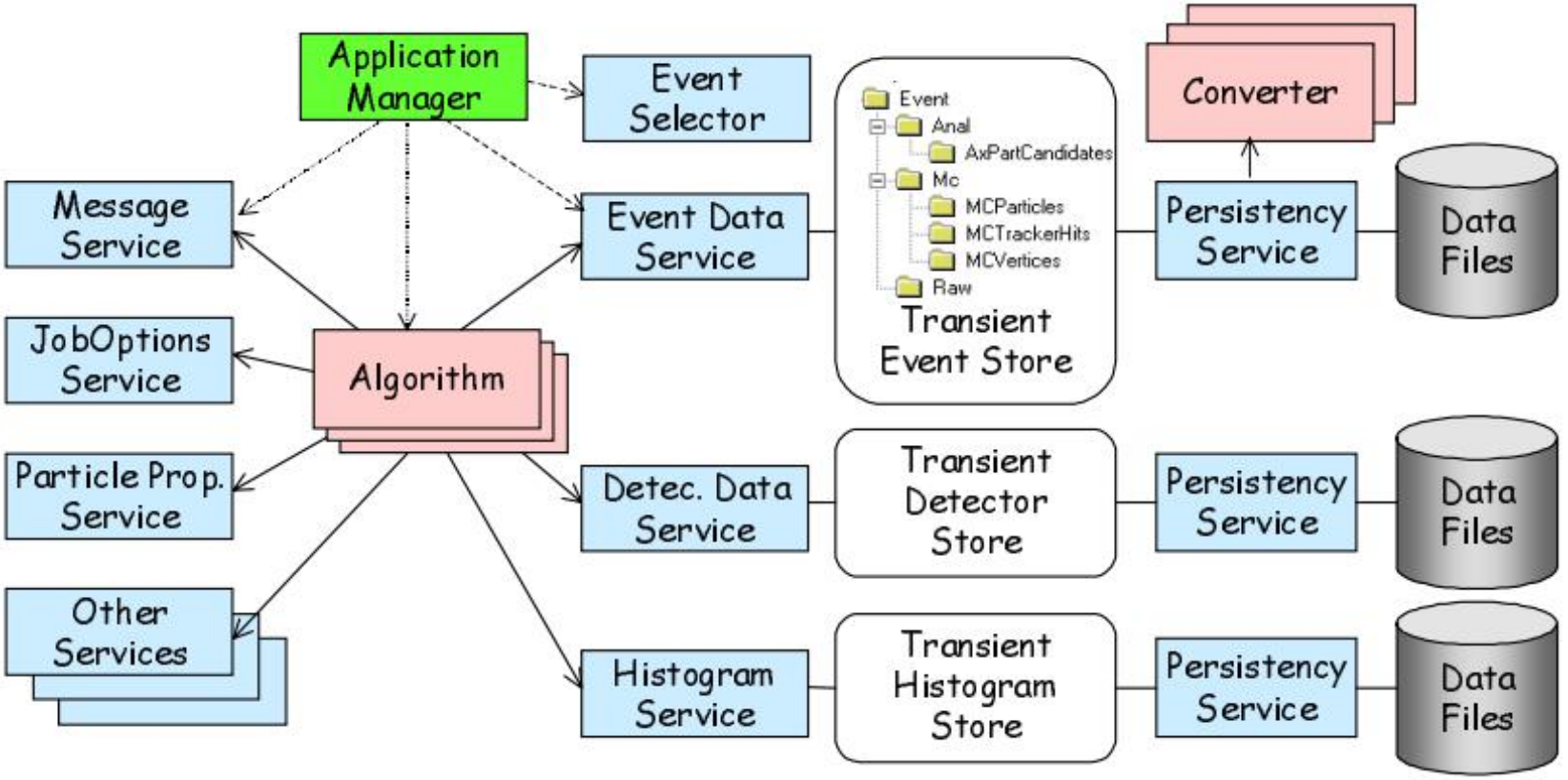
- Based on Key4HEP
- Reuse existing components
  - Gaudi, EDM4hep, DD4hep, ...
- Implement the specific components for CEPC
  - Geometry, generator, simulation and reconstruction algorithms, etc.
- Provide ready-to-work environment to algorithm developers and physicists
  - Porting algorithms from iLCSoft to CEPCSW
  - Integrate/develop more algorithms and features



https://github.com/cepc/CEPCSW

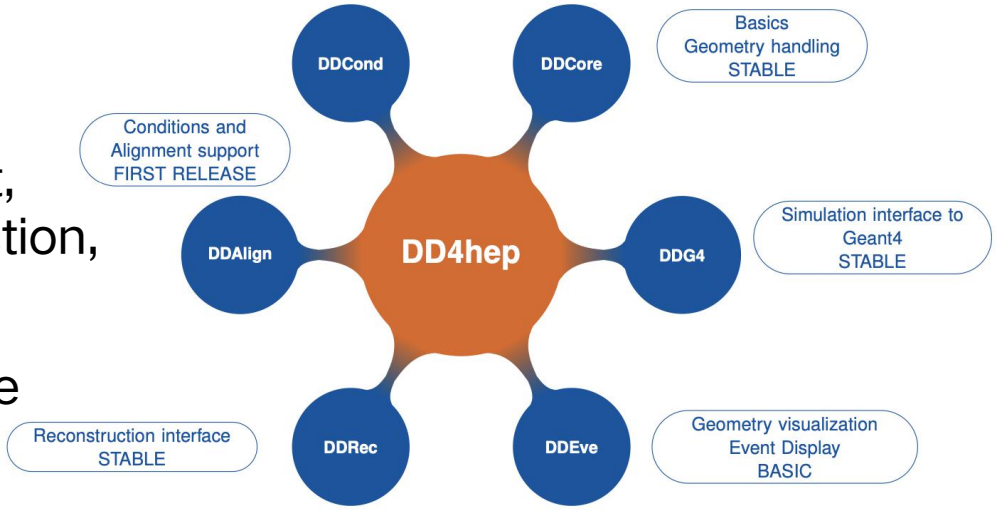
Data Processing Framework: Gaudi

- Modular design and well defined component interfaces
- Three basic categories of data characterized by their lifetime
- Seperation of algorithm, transient and persistent data in the event loop



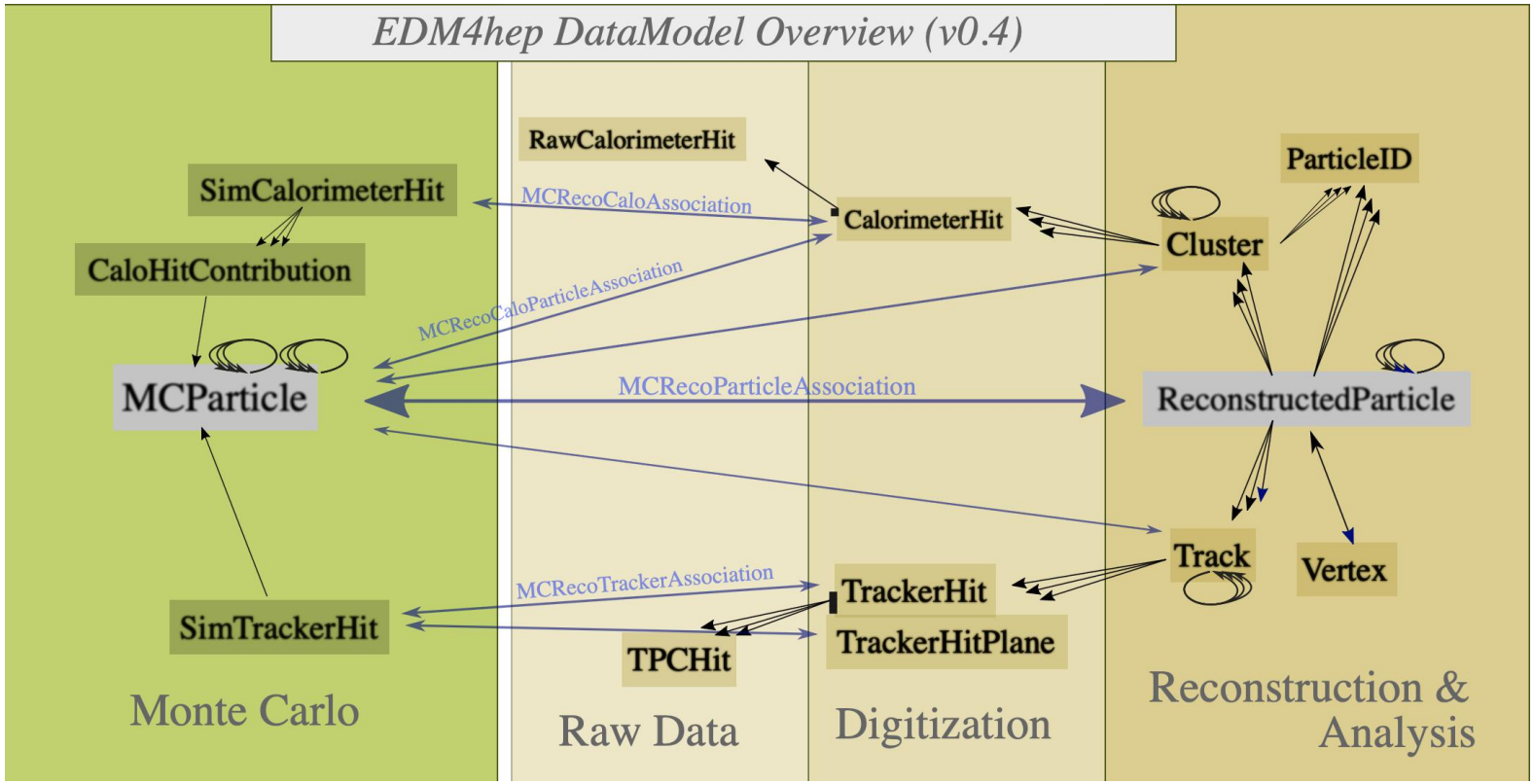
Geometry Description Toolkit: DD4hep

- DD4hep provides complete detector description in CEPCSW
  - Provides geometry, materials, visualization, readout, calibration...
- Single source of information to ensure consistent description
  - In simulation, reconstruction, analysis
- Supports full experiment life cycle
  - Detector concept development, detector optimization, construction, operation
  - Facile transition from one stage to the next
- See latest developments:
  https://github.com/AIDASoft/DD4hep



Common Event Data Model: EDM4hep

- EDM4hep is the common EDM that can be used by all communities in the Key4Hep project: ILC, CLIC, FCC-ee & FCC-hh, CEPC, ...
- Efficiently implemented, support multi-threading and potentially heterogeneous computing
- Use podio to generate thread safe code starting from yaml description



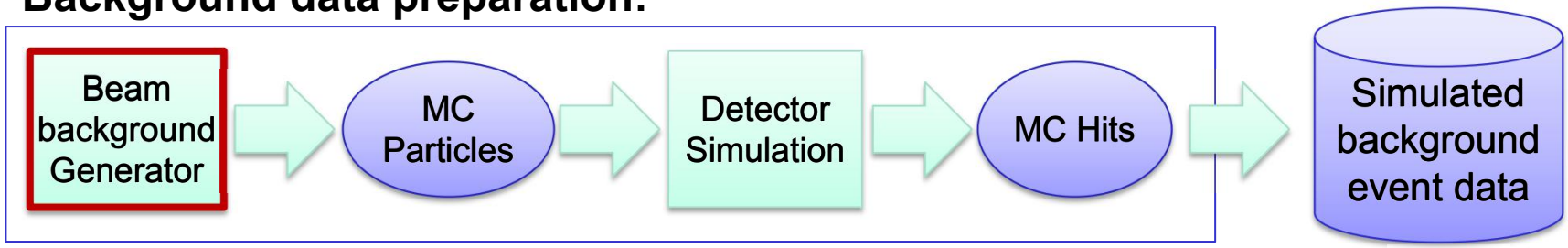
Recent Developments of EDM4hep and podio

- The support of schema evolution is being developed
  - A critical function to guarantee backward compatibility of data
- Support of podio::Frame
  - Act as a container that aggregates and owns all relevant data (collections and metadata)
- EDM4hep extensions under discussion
  - Split EDM4hep into a 'core EDM4hep' and a few extensions
- Other enhancements and improvements
  - Adding edm4hep::Quantity in Track to support dN/dx and dE/dx etc. #137
  - Support of fixed width variables #186
  - A lot more...

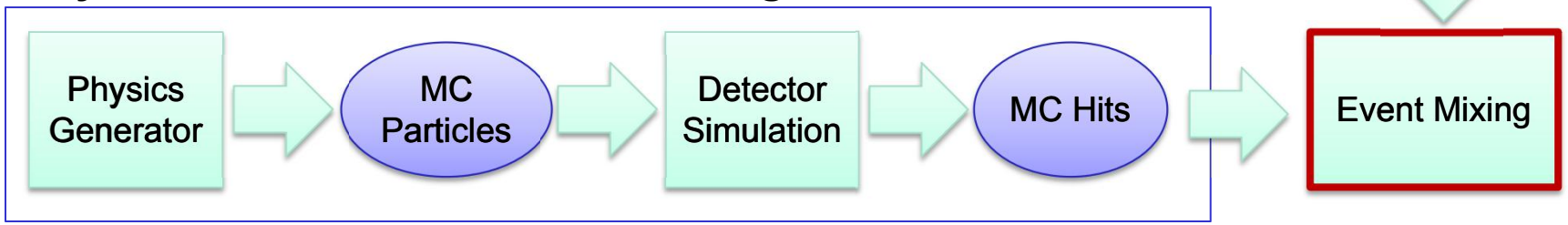
Simulation Framework (1)

- Hit-level background mixing
  - Event pre-mixing strategy is adopted to generate the simulated background data.
  - A GenTool called GtBeamBackgroundTool has been developed to simulate the beam background data.

Background data preparation:

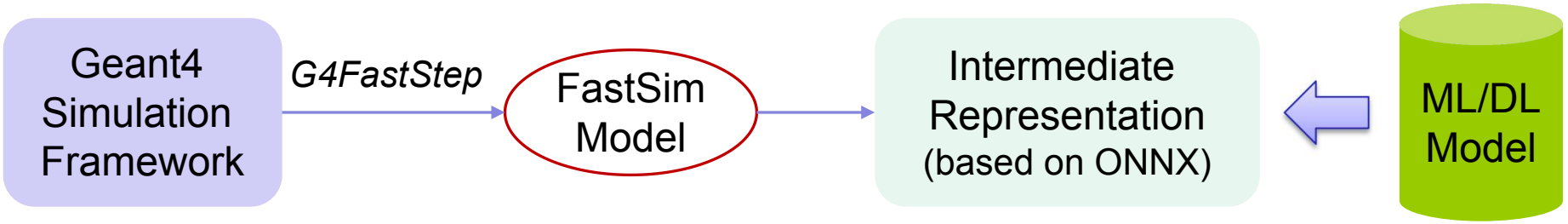


Physics simulation and event mixing:



Simulation Framework (2)

- Integration with ML based simulation
  - ONNX: Open Neural Network Exchange
    - Open standard for machine learning interoperability
  - Inference based on ONNX and ONNX runtime
  - ONNX Runtime:
    - Supporting Models from Tensorflow, PyTorch, scikit-learn, XGBoost…



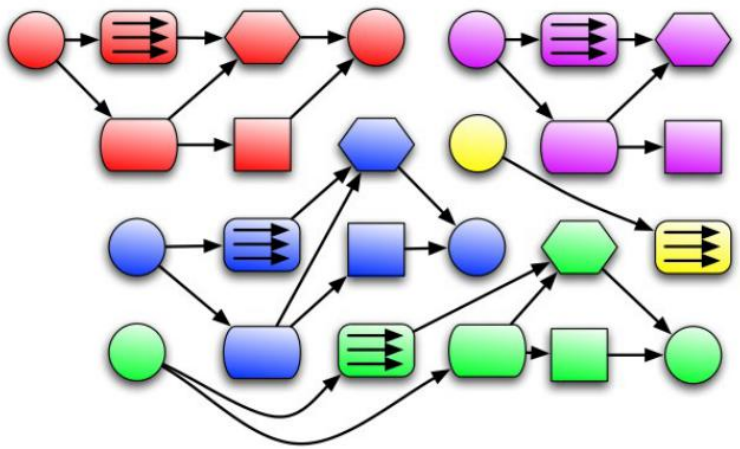
Anna Zaborowska, AIDAInnova WP 12.3 Fast Simulation CERN contribution

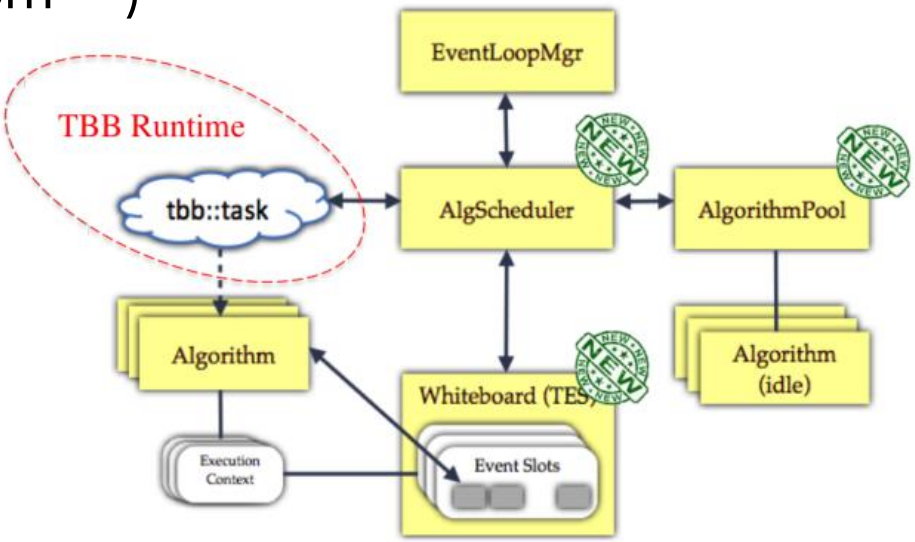
Software Building and Management

- Common tools
  - CMake: Build & deployment
    - Gaudi cmake macros
  - Git: version control
    - https://github.com/cepc/cepcsw
    - A Gitlab mirror is planned
  - CVMFS: software distribution
    - CEPC specific: /cvmfs/cepcsw.ihep.ac.cn/prototype
- In current software release
  - Build by ourselves: Gaudi, k4FWCore, EDM4hep, GEAR, GenFit, podio, LCIO, k4LCIOReader, Pandora
  - Re-use the LCG software stack for other external libraries

Parallel Computing

- GaudiHive
  - A component in Gaudi now
  - Concurrency in event & alg level
- Work to be done in Key4HEP
  - Thread-safe I/O components
  - Other core services (DB, Geom …)
  - Performance testing and optimization
- Challenges
  - Thread-safe algorithms
  - Suitable job types

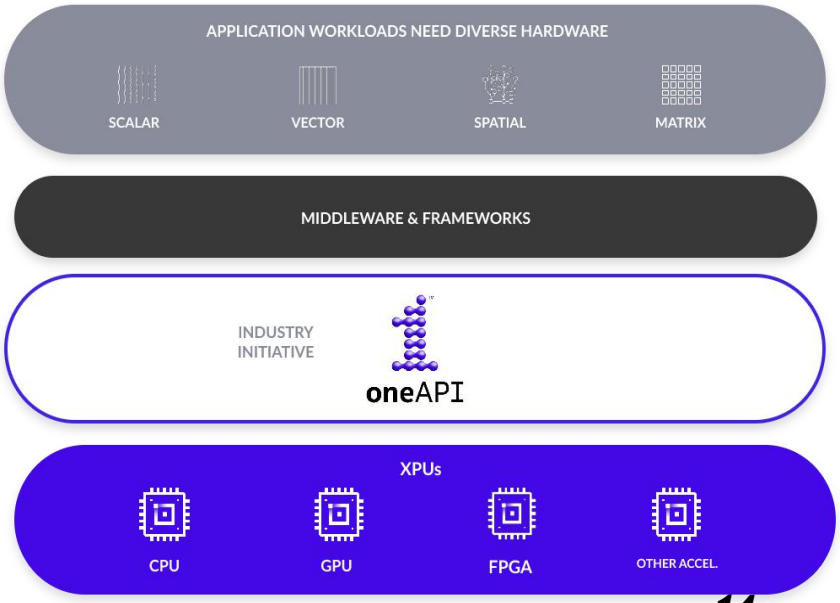




From: http://iopscience.iop.org/1742-6596/513/5/052028

Heterogeneous Computing (1)

- Motivation
  - Computational accelerators are becoming increasingly prevalent.
  - LHC experiments, including ATLAS and CMS, launched heterogeneous computing projects a number of years ago.
  - With an abstract software layer, it can be achieved for one copy of source code to be run on different hardware devices.
- SYCL and DPC++
  - SYCL enables the definition of data parallel functions, which can be offloaded to CPU/GPU/FPGA devices, by providing required APIs and runtime libraries.
  - The oneAPI DPC++ is an extension on top of SYCL. So it can provide a unified programming model across multiple hardware architectures.



Heterogeneous Computing (2)

- Traccc is successfully built with DPC++ from Intel oneAPI and the Intel/llvm compiler with CUDA extension.

Config	Hardware	OS	Compiler	SYCL backend	Bulid traccc	Run traccc
1	Intel CPU (IHEP login node)	CentOS 7.8	LCG 101 (GCC 10.3 + clang 12) + oneAPI DPC++	CPU	OK	OK
2	Intel CPU + NVIDIA RTX 8000 (workstation)	CentOS 7.9	LCG 101 (GCC 11.1) + intel/llvm (2021-12)	CUDA 11.2	OK	OK

- The different devices could be used automatically with SYCL.

```
[lint@lxslc705]$ export TRACCC_TEST_DATA_DIR=$(pwd)/data
[lint@lxslc705]$ ./build_sycl/bin/traccc_seeding_example_sycl --detector_file=tml_detector/track
ml-detector.csv --hit_directory=tml_full/ttbar_mu300/ --events=10
Running ./build_sycl/bin/traccc_seeding_example_sycl tml_detector/trackml-detector.csv tml_full/
ttbar_mu300/ 10
Running on device: Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz
==> Statistics ...
- read    1386759 spacepoints from 0 modules
- created (cpu)  0 seeds
- created (sycl) 264832 seeds
==> Elpased time ...
wall time           35.7607
hit reading (cpu)   26.0041
seeding_time (cpu)  1.0116e-05
seeding_time (sycl) 8.70652
tr_par_esti_time (cpu)    4.398e-06
tr_par_esti_time (sycl)   0.592434
```

Config 1: more than 1 CPUs are used.

```
[tao@localhost traccc]$ export TRACCC_TEST_DATA_DIR=$(pwd)/data
[tao@localhost traccc]$ ./build_clang_sycl_cuda/bin/traccc_seeding_example_sycl --detector_file=
tml_detector/trackml-detector.csv --hit_directory=tml_full/ttbar_mu300/ --events=10
Running ./build_clang_sycl_cuda/bin/traccc_seeding_example_sycl tml_detector/trackml-detector.cs
v tml_full/ttbar_mu300/ 10
Running on device: Quadro RTX 8000
==> Statistics ...
- read    1386759 spacepoints from 0 modules
- created (cpu)  0 seeds
- created (sycl) 277930 seeds
==> Elpased time ...
wall time           32.7898
hit reading (cpu)   29.1174
seeding_time (cpu)  8.491e-06
seeding_time (sycl) 1.91882
tr_par_esti_time (cpu)    3.481e-06
tr_par_esti_time (sycl)   0.0675415
```

Config 2: NVIDIA GPU is used.

Heterogeneous Computing (3)

- Progress
  - With dpcpp and Intel LLVM, track seed finding software of ACTS was successfully compiled and run on CPU and GPU servers respectively.
  - The DPC++ code for track seed finding with pixel detector is being studied.
  - Collaboration with Intel (China) was established and zoom meetings were organized for technical discussions.
- Short-term plan
  - With the help of profiling tools, the performance of track seed finding will be studied on both Intel and NVIDIA GPUs.
  - The performance of EDM4hep/PODIO will also be investigated in the heterogeneous computing environment.
  - Implementation of asynchronous offloading in the framework

Development of Analysis toolkit based on RDataFrame

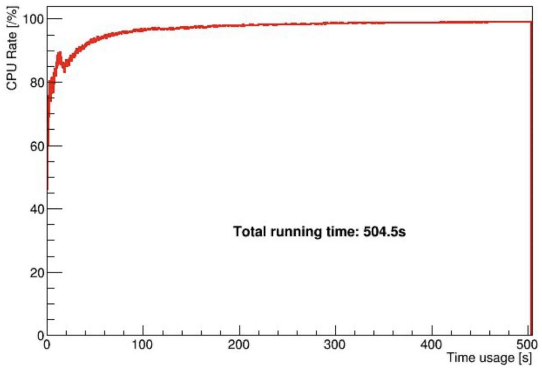
- RDataFrame became official part of ROOT since v6.14
  - Support declarative programming
    - allow users to write efficient physics analyses more easily
  - Fastest way to analyze data
    - supports local multi-threaded parallelization with TBB
    - also supports distributed data analysis with python+Spark without additional changes to the analysis code
  - Full support analysis in both Python and C++
    - Automatically generated Python bindings with PyROOT
  - Already support reading EDM4hep root files
  - Actively used by FCC-ee for flavour, higgs and top physics analysis

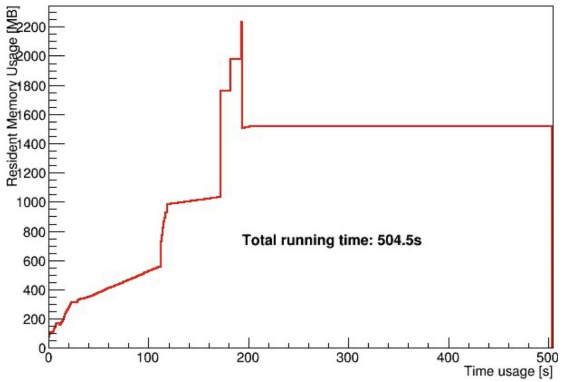
Development of Analysis tool based on RDataFrame

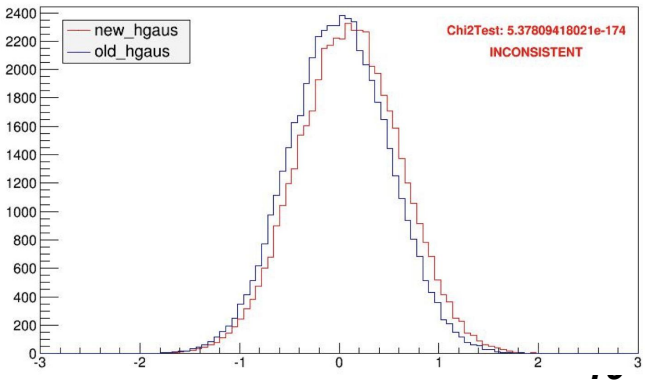
- Plan for development analysis tool for CEPC
  - Start from a inclusive analysis: Higgs recoil analysis and Higgs width measurement in e+e- -> Z(mumu)H process
    - https://github.com/zeusmail/higgsrecoil (from Gang Li)
  - Use CEPCSW to convert the LCIO data produced with Marlin to EDM4hep
  - Develop the common components (functions) for analyzing EDM4hep data (partly from FCC-ee)
    - Analysis function code in C++
      - event selection, filtering, vertexing, PID, Jet clustering, producing ROOT ntuples,…
    - Python for configuration
      - define analysis functions, output variables, input samples, etc…
  - Testing the multi-threading performance

Automated Validation System

- An automated validation system is being developed for software validation at different levels
  - Unit test, integrated test, performance test, physical validation etc.
- A toolkit is developed for building software validation workflow
  - Provide interfaces to define and run unit tests
  - Support performance profiling
  - Support results validation based on statistical methods
- Automated physical validation system based on massive data production is being developed

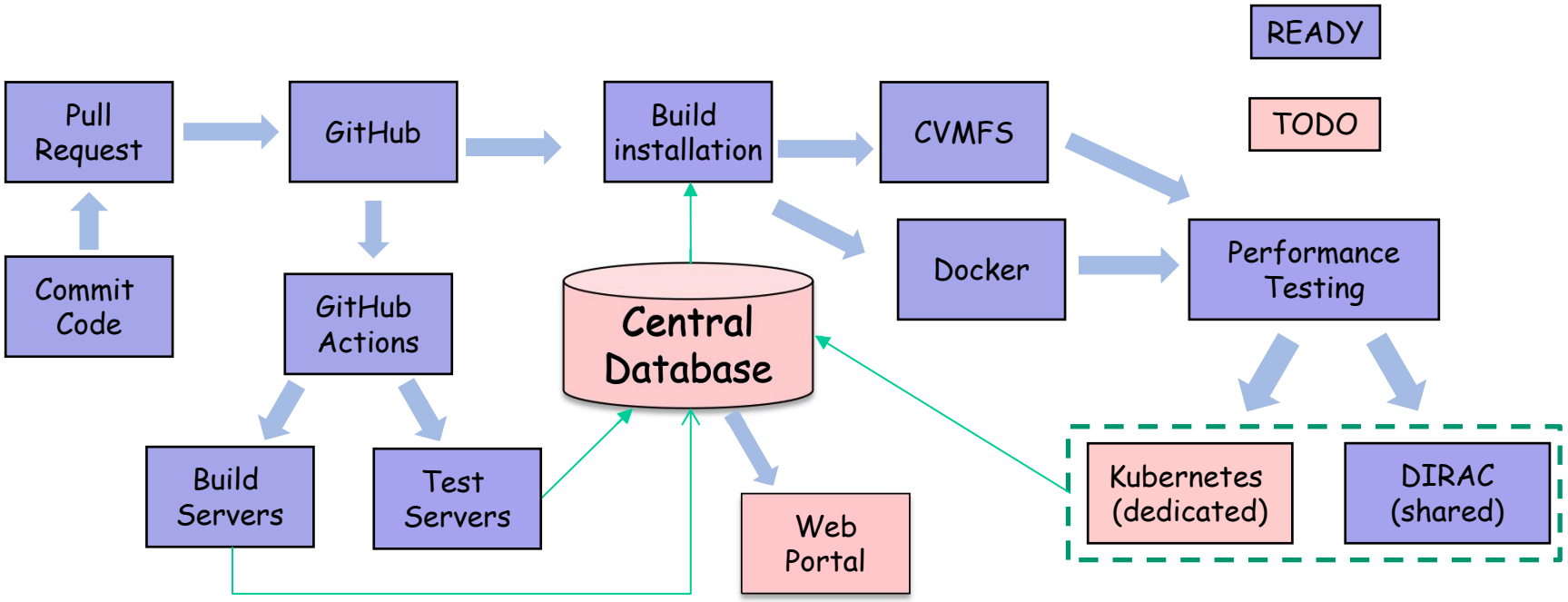






Automated Validation System

- The validation system is being integrated with the Github Action system
  - Full validation workflow can be triggered by commit/merge-request
  - A web-based monitoring dashboard is also being developed
- ~ O(200) cores are now available for running validation jobs



Summary

- CEPCSW framework is being developed in collaboration with the Key4hep project
- Most CEPCSW core software components are in place and function well to support detector simulation and reconstruction studies
- Latest developments are focused on:
  - Parallel computing
  - Heterogeneous computing
  - RDataFrame based Analysis framework
  - Automated validation system

Welcomed to joining CEPCSW!
We hope to work together with developers in the community.
https://github.com/cepc/cepcsw

Backup

Data Input/Output

- The default EDM4hep data format: ROOT
- k4FWCore: the default data I/O components
  - PodioDataSvc: read/write podio data collections in ROOT
  - DataHandler: register data collections to Gaudi
- k4LCIOReader: read LCIO data generated by Marlin
  - Data converters