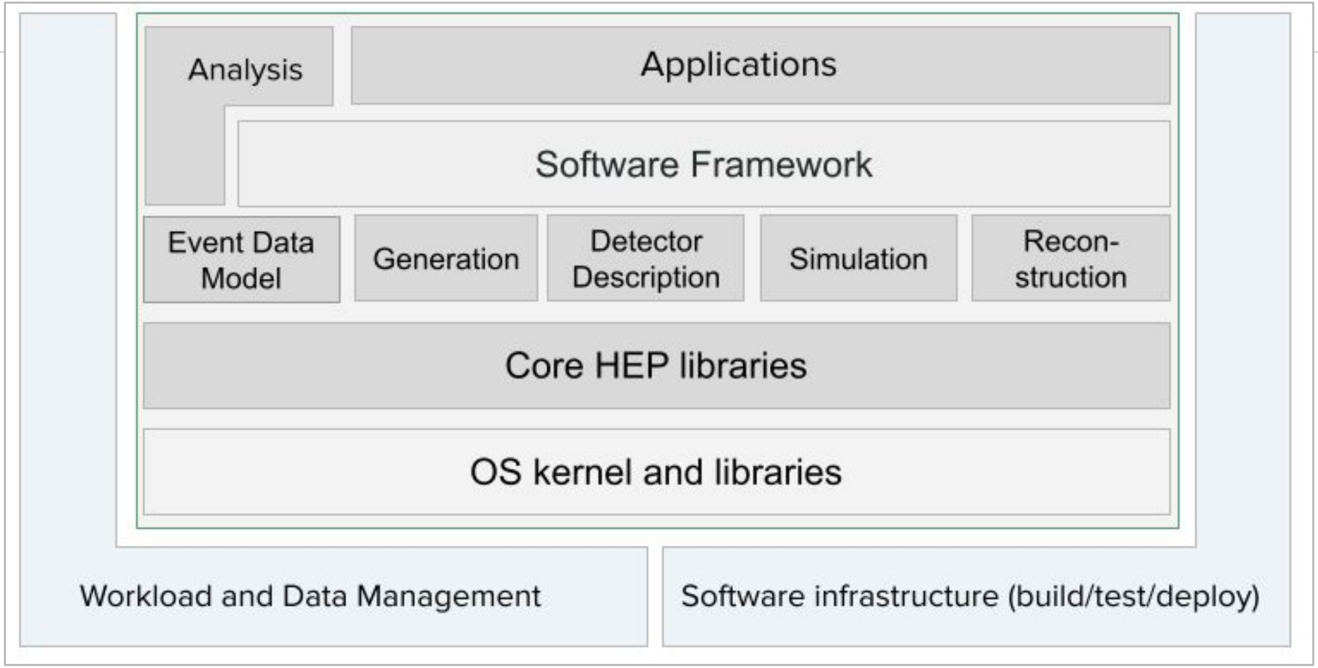
Software Ecosystem

PED-Higgs kick-off meeting

June 18, 2021
G Ganis, CERN-EP

HEP Software Ecosystem

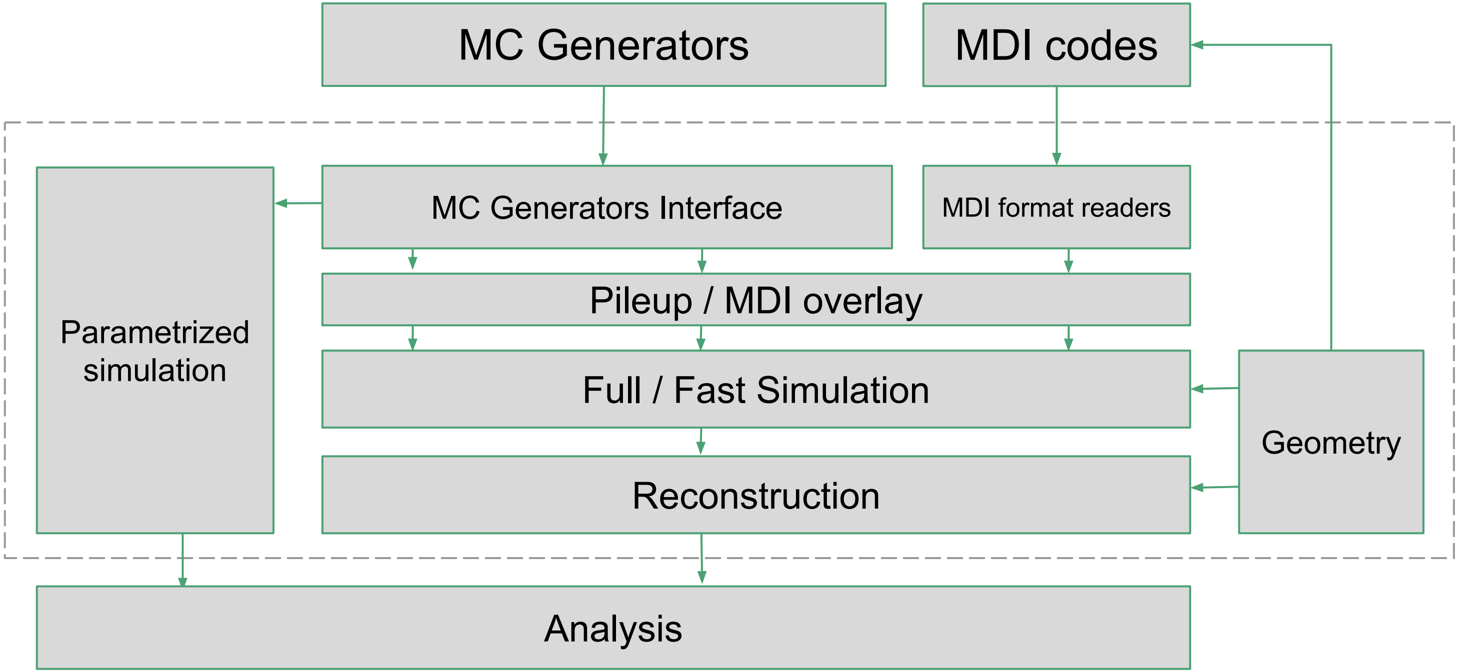
Seamless integration and optimization between various networks of devices, software and services aimed to facilitate data processing for High Energy Physics experiments



Scope of this presentation: future EW/Higgs/Top factories

- Two families of e+e- collider experiments: linear or circular
- Similar detection context, different experimental conditions
  - Center of mass energies, luminosities
  - Interaction Region / Machine Detector Interface
- Similar software needs: support for physics and detector studies
  - Flexible detector description, open to evolution
    - Easy switch / replace sub-detectors, change dimensions, layout, …
  - Completeness: include all major aspects
    - Generation, {Parametrized, fast, full} simulation, reconstruction, analysis, MDI support, …
  - Ease of use: low usability threshold and fast learning curve
    - Extensive documentation, regular training
- Similar needs for software tools to manage computing
  - Tools to facilitate effective access to CPU / storage resources

Typical workflows to support



Levels of interoperability

- Level 0 - *Common Data Formats*
  - Maximal interoperability, even on different hardware
- Level 1 - *Callable Interfaces*
  - Defined for one or more programming languages
  - Implementation quality of interfaced components important
  - Required to define plugins
- Level 2 - *Introspection Capabilities*
  - Software elements to facilitate the interaction of objects in a generic manner such as Dictionaries and Scripting interfaces
  - Language bindings, e.g. PyROOT
- Level 3 - *Component Level*
  - Software components are part of a *common framework*, optimal interplay
  - Common configuration, log and error reporting, plug-in management, ...

The role of the framework

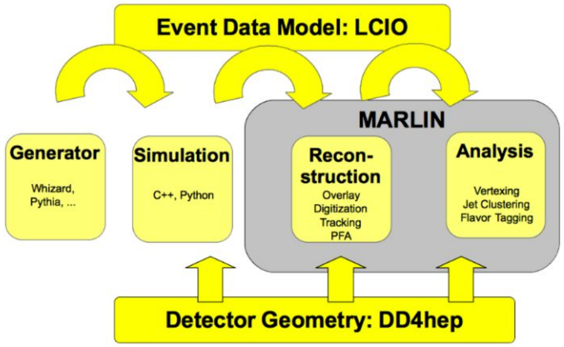
- Provide uniform view on the components
  - Common configuration, log and error reporting, plug-in management, ...
- A good framework adapts to varying landscape to always provide optimal interoperability and use of resources. Today this means:
  - Solid multi-thread support, ability to cope with heterogeneous resources
    - Both in terms of different hardware (GPU, FPGA), segregation level (limited network connection), cloud protocols
- HEP experiments have a tendency to start from scratch
  - Typically mutuating concepts from previous experiences
- Adopting a demonstrated solution matching the (projected) needs buys in the experience and future evolution

ILCSoft approach

F. Gaede, Bologna 2019

Brief history of iLCSoft

- in 2002 there where three LC projects in the world: Tesla, JLC and NLC
  - and about four or five different detector concepts and software frameworks
  - using C++, Java and F77 (no Python yet)
- decided to provide the basis for collaboration and common development by defining the common language, i.e. the event data model: **LCIO**
- adding **Marlin** already provided the basis for iLCSoft
- last major evolution: develop and incorporate the **DD4hep** geometry toolkit

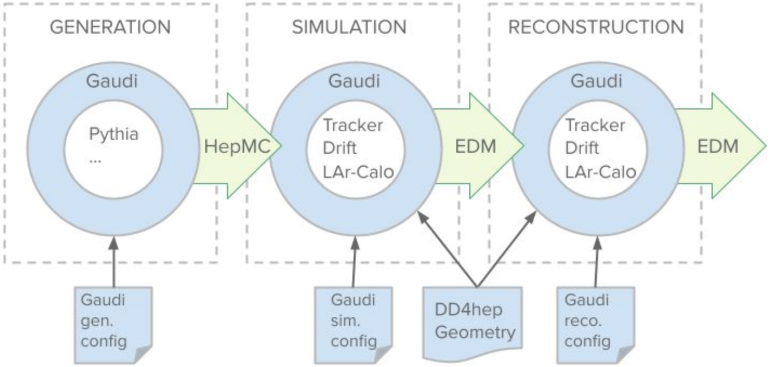


iLCsoft schema today

Design goals: keep it simple, modular, flexible, using and developing generic HEP tools whenever possible

FCCSW approach

- Started in 2014
- Driving considerations
  - One software stack to support all the cases (hh,ee,eh), all the detector concepts
  - Need to support physics and detector studies
    - Parametrised, fast and full simulation (and mixture of the three)
  - Modularity: allow for evolution
    - Component parts can be improved separately
  - Allow multi-paradigm for analysis
    - C++ and Python at the same level
- Adopted Strategy
  - Adapt existing solutions from LHC
  - Look at ongoing common R&D projects (AIDA)
  - Invest in streamlining of event data model
- Focus on FCCee after CDR (2019)



LHC as a reference?

- Requirements of running experiments, in particular LHC, in terms of software and computing, are unprecedented. Working solutions exist for:
  - Frameworks
  - Reconstruction techniques / algorithms
  - ML techniques development / deployment
  - MT, heterogenous (GPU, FPGA, …)
  - Workload and Data Management
  - Software build / packaging / test / deployment
  - Analysis tools
  - ...

New developments should focus on what is uncovered / specific to EW/Higgs factories e.g. e+e- MC, flexible geometry, specific detector technologies, reconstruction/analysis tools, ... possibly generic and re-usable.

AIDA, AIDA2020, AIDAInnova

- Joint European effort for detector R&D
- Successful software packages (WP2@AIDA, WP3@AIDA2020)
    - Core software, Simulation
        - VecGeom: vectorized geometry
        - DD4hep: geometry description, conditions data, alignement + extensions
            - Gateway to Geant4 (DDG4), interface to reconstruction (DDrec)
        - PoDIO: EDM toolkit
    - Advanced reconstruction
        - Tracking (converged to ACTS)
        - Particle Flow (PandoraPFA)
- Approved follow-up: AIDA-Innova 2021-2025
    - Fast Simulation, Track Reconstruction (ACTS), Particle Flow (PFA), Turnkey Software Stack
    - Focus on: parallelisation, acceleration, machine learning

The common software vision

Create a software ecosystem integrating in optimal way various software components to provide a ready-to-use full-fledged solution for data processing of HEP experiments

Complete set of tools for

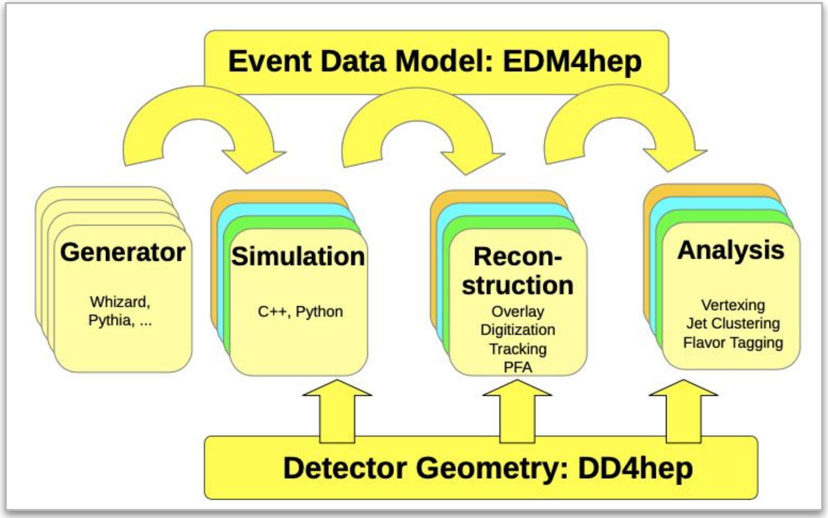
- Generation, simulation, reconstruction, analysis
- Build, package, test, deploy, run

Core Ingredients of current key4hep

- PoDIO for EDM4hep, based on LCIO and FCC-edm
- Gaudi framework, devel/used for (HL-)LHC
- DD4hep for geometry, adopted at LHC
- Spack package manager, lot of interest from LHC

Community project, unifying efforts

- Contributions from CLIC, ILC, FCC, CEPC

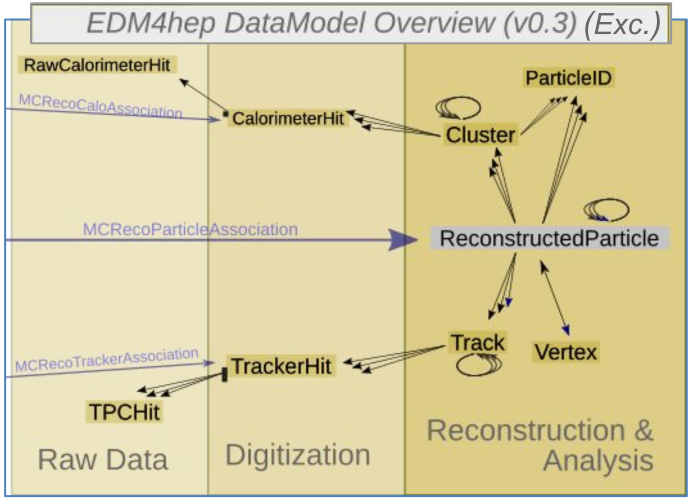


Kick-off meetings in [Bologna](), [Hong Kong]()

The common event data model: the challenges

EDM provides common language for exchange among framework components

- Challenge 1: efficient support different collision environments (e+e-, pp, ...)
  - Positive first experiences with FCC-hh components
- Challenge 2: keep I/O efficient
  - PoDIO: separate definition from implementation, facilitate optimal adaptation to backend
    - POD layer designed for efficient I/O, simple memory layout
    - Flat data support (RNTuple) will provide insight
- Challenge 3: efficient support for schema evolution
  - Requires schema evolution in PoDIO, planned
- Challenge 4: efficient support for detector needs
  - Interaction w/ detector teams from the start
    - Eg. cluster counting for IDEA Drift Chamber
    - See also P. Roloff's talk



Key4hep adoption plans

- ILC/CLIC
  - Keep existing software chains / samples available for on-going studies
    - Enabled by k4MarlinWrapper and k4LCIOReader components
  - Full reconstruction chain through the wrapper (and EDM4hep) conversion part of the AIDA Innova work plan milestones
    - Study overheads, eventually port algorithms to Gaudi
- FCC
  - Already Gaudi based, move FCC-edm to EDM4hep
  - Re-arrangement and modernization into components considered for migration to common project
    - Generation, simulation, reconstruction, …

MC Generators

More details in W. Kilian's talk

Needs

- High energy ($\sqrt{s}$ > hZ threshold) e+e- generators
- Generators at Z peak, WW
- Heavy Flavour decays, including taus

Examples of heavly used codes for LC

- Whizard, MadGraph5_aMC, PhysSim, Pythia6, ...

Areas of work

- Recovery of LEP generators, but still state of art for Z peak, WW
  - KKMC family, BHLUMI, BHWIDE, Babayaga, … interfacing work in progress
- Hadronization “tunes” for e+e- (Pythia, Herwig, …)
  - Eg. cannot import Pythia6 tune (from LEP) to Pythia8
- Interfaces with up-to-date decay codes (EvtGen, …)

Beam and MDI-related backgrounds

Required level of understanding needs integration in experiment software

- Several processes and codes, including
  - (In)coherent pair creation — GuineaPig
  - Synchtron Radiation — MDISim, SynRad, Sync_Bkg
  - Radiative bhabhas — GuineaPig, BBBrem, BHLUMI, Whizard
  - $\gamma\gamma \rightarrow$hadrons — GuineaPig + Pythia

Requirements depend on experimental context and therefore community dependent:

LC: mostly w/ GuineaPig
- Standalone; level-0 interplay with Pythia and Whizard; and w/ the simulation (DDG4)

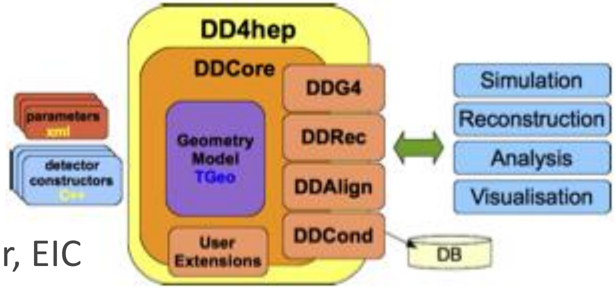
FCC: mixture of codes
- for CDR, standalone occupancy estimations, CLD occupancies with iLCSoft

Codes not always in public repositories, outputs in different, non-standard formats
Framework integration in key4hep could unify/simplify access of each relevant codes

Geometry Description - DD4hep

More details in [D. Jeans' talk]

- Addresses the needs of HEP
  - Precision: possibility to precisely describe the smallest element
  - Flexibility, modularity: facilitate composition of basic elements
  - Universality: single source for all needs
- De facto a standard
  - AIDA2020 project for ILC/CLIC, adopted by FCC, CEPC, Muon Collider, EIC
    - Chosen by CMS and LHCb (run 3)
  - Geometry description with C++ detector ctors and XML (*compact*) files
  - Component-based architecture, interfaces for alignment and conditions data, python bindings



iLCSoft
- lcgeo: models repository
- ddsim: full simulation with DDG4
- DDrec: reconstruction interfaces

FCC
- FCCDetectors: models repository
- GeoSvc: DD4hep Gaudi service, translation to Geant4 geometry, constructions of sensitive detectors
- Framework integration: simulation, reconstruction

Detector palettes

LC: [lcgeo]

- CLICdp, ILD, SiD, CLD, test beam setups

FCC: [FCCDetectors]

- FCChh baseline, CLD, simplified IDEA tracker (DC+vertex), IDEA LAr, ...

Areas of work

- Complete detector concepts: IDEA DC and vertex, Dual Readout calorimeter, muon detectors, …
- Establish/consolidate dynamics for implementation of new concepts in DD4hep
- Key4hep unified repository for detectors?

Parametrized Simulation

More details in D. Jeans' talk

Fast parametrization of detector response w/o transport; O(ms/evt)

LC approach:

- SGV (Simulation Grande Vitesse) used for ILC (standalone);
  - Tracks w/covariance, rest typically parametrized
- Delphes (CLIC)

FCC approach:

- Delphes, w/ EDM4hep output (standalone)
- Latest versions includes tracks w/covariance, rest typically parametrized

Areas of work

- Consolidate framework integration of Delphes
- Validate Delphes cards with full simulation (or test beam results)

Full Simulation

More details in D. Jeans' talk

LC approach: DDG4 through ddsim (DD4hep)

- Includes also implementations for sensitive detectors, MC-truth linking, magnetic field maps, ...

FCC approach: Framework integration à la LHCb (Gauss, Gaussino)

- *Fast* simulation: full transport + parametrized response; dedicated ‘physics’ process in Geant4; parametrization tuned on full simulation

Areas of Work

- Evaluate possibility of unified approach with maximal re-use of existing code
  - For example digitisation and MC linking from iLCSoft / DDG4
- Integration of existing standalone implementations interesting for the community
  - E.g. IDEA DC and DR calorimeter

Reconstruction

More details in [P. Roloff's talk]

LC: DDrec (DD4hep)
- Generic interface for tracking: pattern recognition, fitters, surface definition, …
- PandoraPFA: generic framework for pattern recognition in calorimeters

FCC: Framework integration
- Tracking and calorimetric algorithms for baseline FCChh, little specific to FCCee
  - Full sim studies for FCCee not really started (using conversion to LCIO for tests)

Areas of work
- Consolidate LCIO to/from EDM4hep on the fly converter
  - Enables access to LCIO-based algorithms in key4hep
- Integration of existing algorithms interesting for the community, e.g. IDEA DC and DR calo
- Framework integration of general purpose tools such as ACTS, PandoraPFA, CLUE/TICL, ...

Analysis

More details in P. Roloff's talk

LC approach

- Several analysis algorithms (jet clustering, vertexing, ID, …) available for LCIO
- Run through Marlin, output can be read in ROOT, Julia, Python

FCC approach

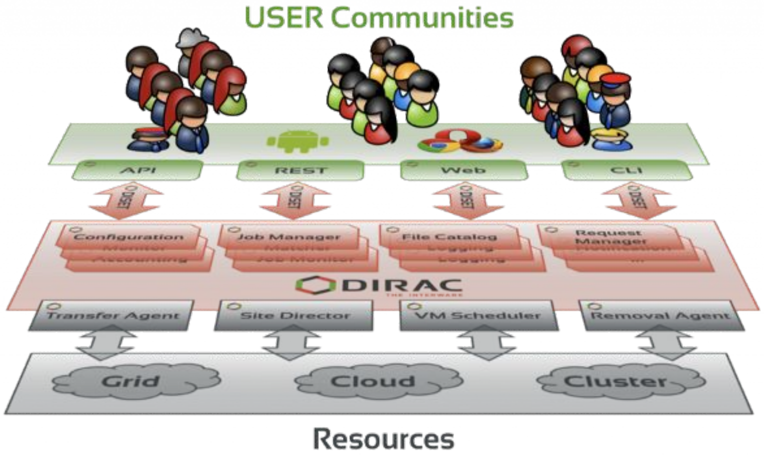
- FCCAnalysis: declarative analysis framework for EDM4hep based on RDataFrame with high level python interface (PyROOT), access to advanced python tools (e.g. akward array)
- Library of C++ algorithms (thrust, clustering, vertex, …) being populated

Comments

- Generic framework working on EDM4hep would be good for the community
  - Consolidate algorithm offer, avoid duplication of efforts
- Need to be able to preserve access to existing tools, e.g. w/ LCIO↔EDM4hep conversion

Workload Management: DIRAC

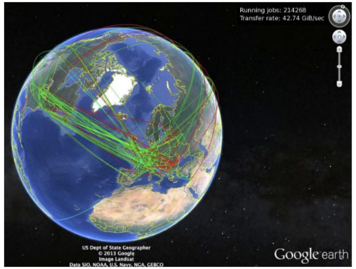
- Lots of developments in LHC to optimally exploit distributed resources
  - Each experiment its own tool: BigPanda (ATLAS), GlideinWMS (CMS), …
- DIRAC: software framework for distributed computing
  - Complete solution for user community(ies) / VOs
    - Workload management, File catalogue
  - Started by LHCb, developed into community project (DIRAC consortium)
    - Used also by Belle II, BES III, JUNO, … ILC/CLIC

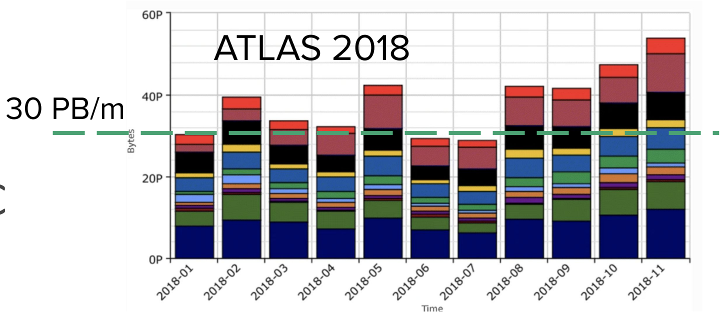


LC community instance, iLCDirac, serves also CALICE and now FCC: good example of re-use of generic solution

Data Management

- Lots of developments and experience in LHC community to manage big amounts of data to optimally exploit networks and distributed storage
  - E.g. CMS transferring actively at 3 GB/s all around the year; DM system automatically deletes O(40PB/month) of «least used data»
- One common solution emerging: RUCIO
  - Developed by ATLAS, adopted by CMS
  - De-facto a standard
  - DIRAC being interfaced with RUCIO
- WLCG DOMA project addressing the needs of HL-LHC





Needs of future projects well covered for now

Build / packaging / testing / deploying

Lots of tools / technologies acquired in HEP during the years

- Build, Packaging
  - Converging on Spack for build recipe definitions, compilation, installation
  - Originating from HPC world though HSF, key4hep community pioneering *production* use
- Deployment
  - CernVM-FS, de-facto standard solution for HEP (origine: LHC)
    - Key4hep is using a HSF dedicated repository: /cvmfs/sw.hsf.org/key4hep
- Continuous Integration, Testing
  - Essential given the size of stacks! GitLab, GitHub in-repo CI support, Jenkins used widely
    - Key4hep uses GitHub actions, for pull request checks, unit-testing (catch2)
- Controlling runtime environment w/ light-weight container technology
  - E.g. singularity (all in user space)

Lots of useful experience to build upon

A few take away messages

- Software is essential during any phase of a project
    - No quality CDR/TDR without a robust and flexible framework
- Designing with long term vision and awareness of existing solutions provides stability and sustainability, and preserves / optimises use of knowledge
    - Documentation and training for both users and developers is fundamental
- HEP is ready for a leap towards a common software ecosystem
    - Now done by the community w/ Key4hep
    - Common tools always existed (Cernlib, PAW, ROOT, …), new general purpose tools go further and enter deep in the running experiments systems (DD4hep, Gaudi, …)
    - Common R&D, such as the AIDAs, CERN EP’s, ECFA’s ... have an essential role in this
- Next generation experiments should try to go that way
    - Also beneficial data preservation

Thank you!

- Key4hep GitHub Project
  https://github.com/key4hep
- Main documentation page
  https://key4hep.github.io/key4hep-doc/
- Doxygen available., e.g. for EDM4hep
  https://edm4hep.web.cern.ch/