Data preservation through modernisation – the software of the H1 experiment at HERA

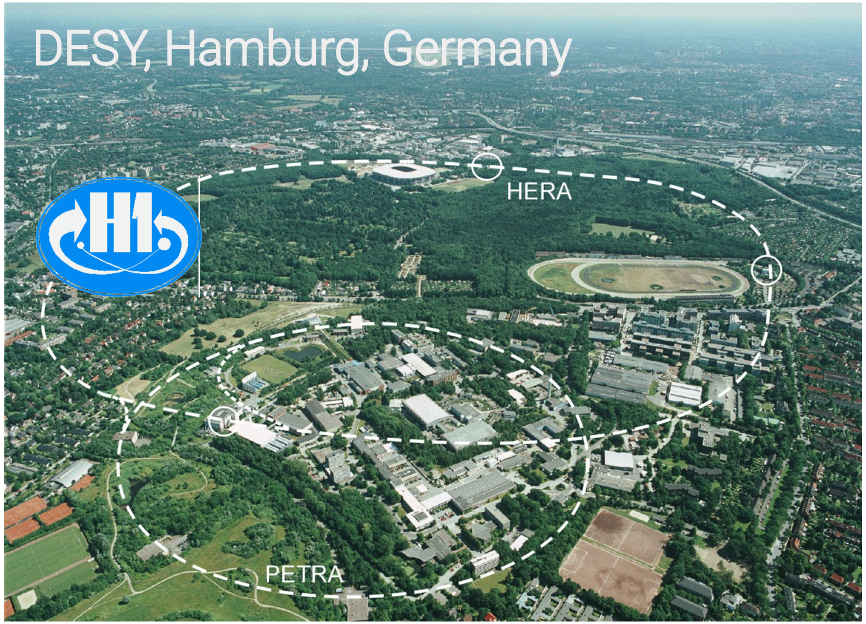
D. Britzger, S. Schmitt, S. Levonian, and D. South for the H1 Collaboration
Max-Planck-Institut für Physik München, Germany

virtual CHEP conference, 18.5.2021

MAX-PLANCK-INSTITUT
FÜR PHYSIK

The H1 experiment at HERA

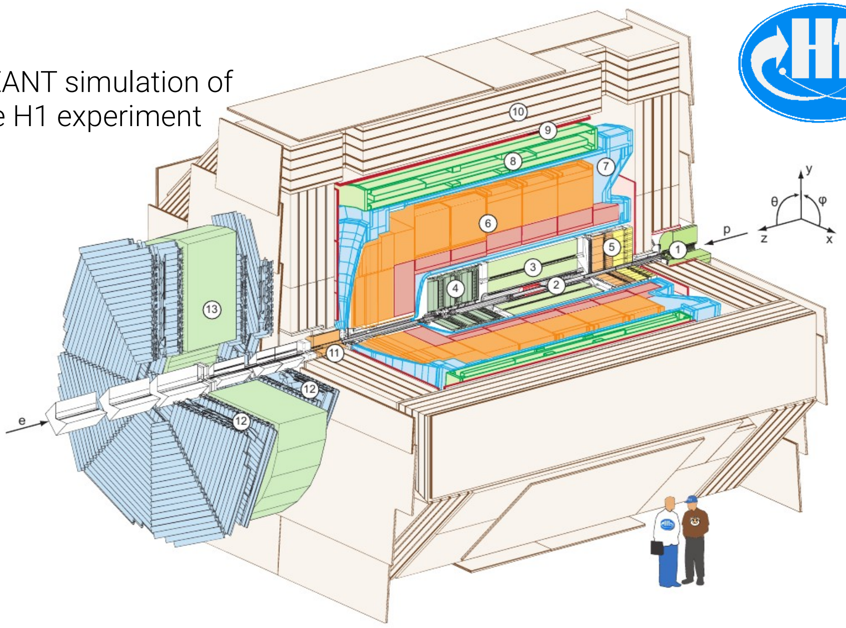
HERA electron-proton collider at DESY



- HERA I: 1994 – 2000
  HERA II: 2003 – 2007
- E_e=27.6 GeV, E_p=920GeV
  $\sqrt{s}$ = 300 or 319 GeV

H1 experiment at HERA

GEANT simulation of the H1 experiment

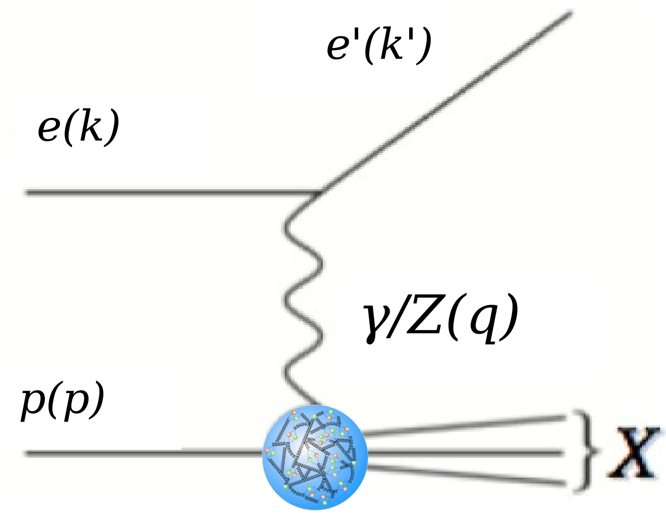


'multi-purpose' detector

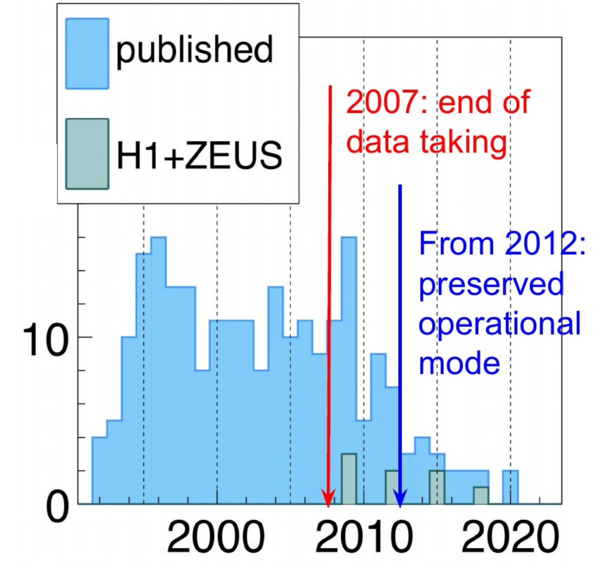
- Asymmetric design with trackers, calorimeter, solenoid, muon-chambers, forward & backward detectors, ...
- 270,000 readout channels

Physics motivation for H1 data preservation and analysis

Deep-inelastic scattering



Publications by H1 over time



Broad physics programme

- Proton structure, QCD, heavy flavors, electroweak physics, exclusive processes, diffraction, BSM, etc...

- Physics programme not yet finalized
  → Physics analysis ongoing or newly starting
- Many new physics ideas evolve
  → emerging interest in DIS because of new DIS-experiments (EIC@BNL, LHeC@CERN, EIC@China)

Data preservation of the H1 experiment

DPHEP Study Group for Data Preservation and Long Term Analysis in High Energy Physics

Data-preservation in HEP (DPHEP)

- H1 with significant involvement in DPHEP study group
  Six workshops in 2009–2012, continuous activity since then
  → community publication: CHEP2012

H1 adopted a 'level 4' preservation model

- Preserve not only analysis level data, but also reconstruction and simulation software as well as the basic level data
- Retain the full flexibility and potential of the experimental data

→ For full access to the data, the software must also be considered

Data preservation (the H1 data themselves)

- ~1 billion *ep* events, Total RAW data: 75TB; Final re-processed data (DST): 20TB; Analysis “H1oo”: 4TB, other special data, full set of MC samples,
  → total data volume about 0.5PB
- Data organised in a dedicated DPHEP storage at DESY (dCache) and a copy in Munich

The FORTRAN 'core' packages and H1oo

The 'core'-software packages (FORTRAN)

- H1 core software written in FORTRAN 77
  [NIM A A386 (1997) 310]
- first developed in 1988
  already with clear structure:
  highly modular, and based on BOS/FPACK
  [S. Egli 1990; V. Blobel 1990; V. Blobel CHEP1992]
- Programs for: Data storage and I/O, simulation (based on GEANT3), reconstruction and post-processing, visualisation (based on LOOK), data analysis, etc…
- MC generators
- External dependencies:
  CERNLIB, GEANT3, GKS, oracle-instant
- about 950k lines of code

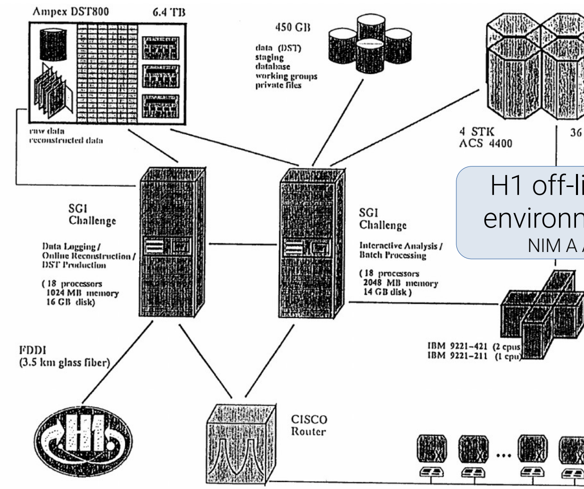
H1oo

- H1oo: object-oriented C++ common analysis framework based on ROOT
  [U. Berthon et al. CHEP2000; M. Peez et al. CHEP2003,
  J. Katzy et al. CHEP2005; M. Steder et al. CHEP2012]
- written in C++98, and until recently based on ROOT 5.34
- About 50 packages and 600 classes; analysis environment and data formats for analysis
- External dependencies:
  ROOT, fastjet, neurobayes-expert
- Standardised H1 data analysis and benefit from expert knowledge
- about 300k lines of code

H1 data and software preservation: 2012 – 2020

H1 computing environment

- Since 1988: DESY centered computing model (storage, workgroup servers + batch system (+grid)) [R. Gerhards et al. CHEP1994]
- Since 2015: only usage of common DESY-IT services (hardly any dedicated H1 resources) [NAF2.0, A. Haupt, Y. Kemp,F. Nowak, CHEP2013]



H1 off-line computing environment ~ 1988(?)
NIM A A386 (1997) 310

'DPHEP level-4' model includes recompilation of software and migration to newer OS

- OS used in 2012: SLD5 (32-bit Scientific Linux (DESY) 5, based on RHEL5)
- Until ~2015: Migration to 64-bit SLD5 (requiring detailed validation [V. Dodonov], sp-system [D. Ozerov and D. South, CHEP2013])
- 2015–2020: move to 64-bit SLD6, and now CentOS7 (possible due to initial 64-bit transition)

Software remained mainly static during this period

- FORTRAN codes were adapted to 64-bit
- H1oo effectively frozen in 2012 and with ROOT 5.34
- External dependencies reliant on H1 action (and experts) for updates

Modernisation of the H1 software

2020: Successful migration to CentOS7, but a few shortcomings now evident in the H1 software

- The programming languages (C++98) and standards are unattractive for new (young) people to learn
- Outdated dependencies, such as ROOT 5, complicate the usage of modern data analysis techniques
- New dependencies may be incompatible (different compilers standards or MC-generator formats)

→ Modern tools cannot or have not in general been introduced

- Relevant maintenance effort for external (outdated) dependencies

Component	Responsible	Maintained packages	Discontinued packages
H1 software	H1	H1 core software, H1OO	–
OS dependencies (continuous updates)	DESY–IT	Oracle, dCache, web–services, compilers, GNU utilities, gmake, system libraries	CVS
External dependencies (selected fixed releases)	H1	fastjet, neurobayes–expert, MC generators	CERNLIB, GKS, GEANT3, ROOT5, LHAPDF5, MC generators

2020: Restructuring the software

- Make use of 'modern' tools and dependencies, and recent releases of external packages

→ Introduction of dependence on the LCG package repository

- Previously: no externally maintained package repository: packages provided manually
- Two effects: reduction of H1 maintenance and bring in newer versions of existing software dependencies and compilers

Modernisation of the H1 software (cont'd)

All code repositories migrated to git (DESY-IT service)

- H1 used CMZ and CVS (H1 did not get to SVN)
- New build instructions for entire H1 s/w stack
  → Less reliance on historic development

Component	Responsible	Maintained packages	Discontinued packages
H1 software	H1	H1 core software, H1OO	–
OS dependencies (continuous updates)	DESY–IT	Oracle, dCache, web–services, GNU utilities, git, gmake, system libraries	–
External dependencies (selected fixed releases)	H1	–	CERNLIB, GKS, GEANT3 (selected) MC generators
External dependencies (selected regular updates)	LCG	LHADPF6, ROOT6, compilers, fastjet, neurobayes–expert, MC generators, (and as back up option: Oracle, dCache, git)	–

Using recent dependencies from LCG release (97a)

- Entire FORTRAN software stack was migrated
  (huge jump in GNU compiler collection 4.8 to 9.2)

H1oo analysis framework updated to ROOT 6 and C++17; CLING replaces CINT

- Original production of data and MC files remain compatible
- New C++ standard allowed s/w improvements, for example range-based for loops in H1Arrays
- Another benefit of ROOT 6 is PyROOT: Fully pythonic analysis of H1 data now possible, incl. interactive

Complete release of all H1 software now on /afs and /nfs at DESY (to be distributed on /cvmfs)

- H1 core packages were previously bound to the DESY−IT infrastructure; now can be relocated
- H1 s/w now runs (in principle) without problems e.g on CentOS7 lxplus at CERN

Bonus: SLD5, SLD6 container builds using Singularity as retrospective “DPHEP level 3” preservation

Examples with CentOS7 LCG_94a release

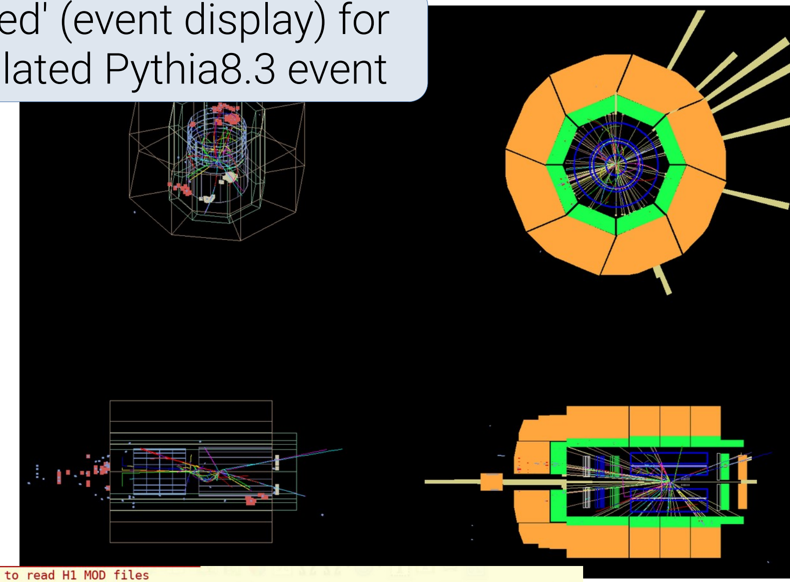
Entire software is relocatable and globally available (here: lxplus@CERN)

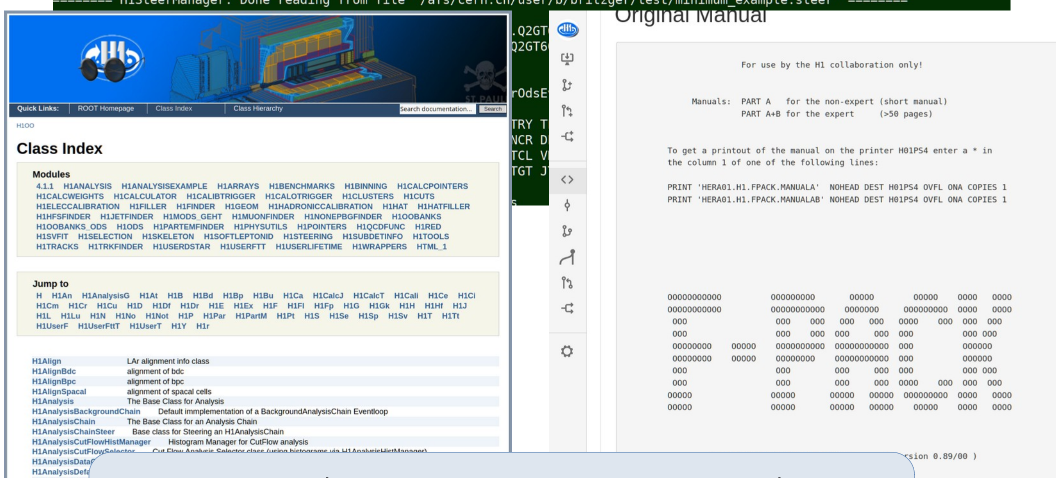
```
[lxplus750] ~/test $ minimum_example -f $PWD/minimum_example.steer

Info in <H1ErrorHandler::H1ErrorHandler>: Using H1ErrorHandler
<H1ErrorHandler::SetMaxCount> Errors printed 1 times
H1SteerManager: Searching for file /afs/cern.ch/user/b/britzger/test//afs/cern.ch/user/b/britzger/test/minimum_example.steer
H1SteerManager: Searching for file /afs/cern.ch/user/b/britzger/test/minimum_example.steer
H1SteerManager: File /afs/cern.ch/user/b/britzger/test/minimum_example.steer opened for reading

======= H1SteerManager: Reading from file '/afs/cern.ch/user/b/britzger/test/minimum_example.steer' ============
----- H1SteerTree(){
----- fHatFiles="hat.4.0.6.DJANGOH14.NC.EPLUSP.0607.RAD.Q2GT60.CTEQ6L.A.7091.S39200.R97800.ftt.DST.0000.root";
----- fModsFiles="mods.4.0.6.DJANGOH14.NC.EPLUSP.0607.RAD.Q2GT60.CTEQ6L.A.7091.S39200.R97800.ftt.DST.0000.root";
----- }
===== H1SteerManager: Accepted values from file '/afs/cern.ch/user/b/britzger/test/minimum_example.steer' ======
================================================================
Steering for H1SteerTree
================================================================
fModsFiles      = 1 entries
fHatFiles       = 1 entries
fTreeCacheSize  = 1000000
================================================================
======= H1SteerManager: Done reading from file '/afs/cern.ch/user/b/britzger/test/minimum_example.steer' ========
```

'H1Red' (event display) for simulated Pythia8.3 event





Extensive documentation: README's, Git, int-notes, PDF-manuals, etc...

```python
# A minimum example to read H1 MOD files
# with python and write the results into
# a ROOT.TH1D histogram and plot it
def minimum_example():
    tree = H1.H1Tree.Instance()
    tree.AddFile("/pnfs/desy.de/dphep/online/h1/mc2/oo-4.0/djangoh14/7091/hat.4.0.6.DJANGOH14.NC.EPL
    tree.AddFile("/pnfs/desy.de/dphep/online/h1/mc2/oo-4.0/djangoh14/7091/mods.4.0.6.DJANGOH14.NC.EP

    # direct access to MOD quantities using H1 Pointers
    Q2_ptr  = H1.H1FloatPtr("Q2e")          # virtuality
    Wgt_ptr = H1.H1FloatPtr("Weight1")      # Event weight
    PartCands_ptr = H1.H1PartCandArrayPtr()  # Array pointer to particle 'candidates' (can be static

    # book some ROOT-histograms
    hist_Q2 = ROOT.TH1F("Q2", "Q^{2} Detector level;Q^{2} [GeV^{2}];events",  40, 10, 1000);
    hist_Pt = ROOT.TH1F("pt", "P_{T} Particles;P_{T} of all PartCands [GeV];events",  40, 0, 10);
    hist_VtxZ = ROOT.TH1F("VtxZ", "Vertex z-position; Vertex z-position [cm];events",  40, -55, 55);
    hist_Empz = ROOT.TH1F("Empz", "Empz; E - P_{Z} of FS [GeV];events",  40, -0, 100);
    # --- H1Calculator, if requested
    gH1Calc = H1.H1Calculator.Instance()

    # --- event loop
    events = 0;
    while tree.Next() and events < 10000:

        # acces Q2, Wgt, etc...
        Q2  = Q2_ptr[0]
        Wgt = Wgt_ptr[0]
        # fill histogram
        hist_Q2.Fill(Q2,Wgt);
```

Full 'pythonic' H1 analysis possible

Summary and conclusion

The H1 experiment at HERA took a unique set electron-proton collision data

- All data preserved and software stack is continuously evolving

H1 data and software are kept in DPHEP mode 'level 4'

- Full offline and online documentation
- Full analysis capability: recompilation of software and continuous migrations to newer OS
- Since 2012: migrations from SLD5-32bit to SLD5-64bit, to SLD6 and to CentOS7
- Bonus: all previous releases can be executed within default Singularity images

Modernisation of H1 software architecture in 2020

- Introduction of LCG dependencies, and DESY-IT standards → reduction of maintenance for H1
- Latest dependencies (gcc9, ROOT6, C++20, Git, ...)
  → Modern analysis and computing environment → attractive for young physicists
- Data are actively analysed and new collaborators are welcomed and are joining