

Software Lifecycle at CERN and in the HSF

Lukas Heinrich for the HEP Software Foundation



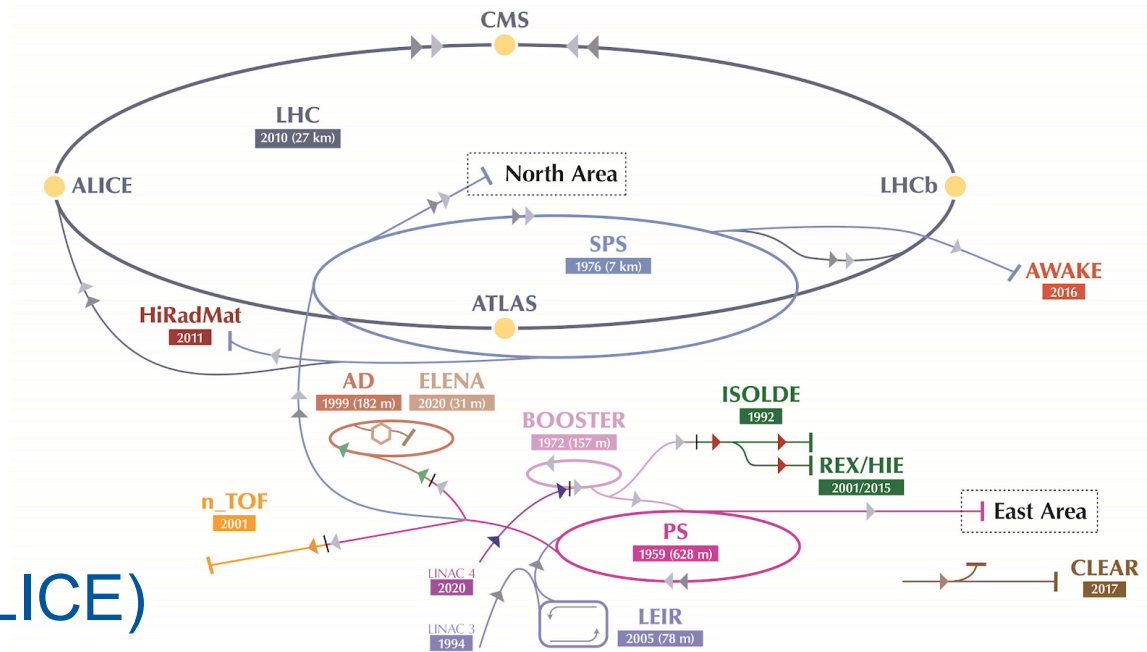
CERN:

Home of some of the biggest scientific experiments:

LHC + its experiments (ATLAS, CMS, LHCb, ALICE)

But also many others smaller ones (TOTEM, ISOLDE, ...)

The Lab is also a central Hub for **High Energy Physics** in general, coordinating many community-wide activities.





Besides massive Hardware...

Software is of course central to everything we do.

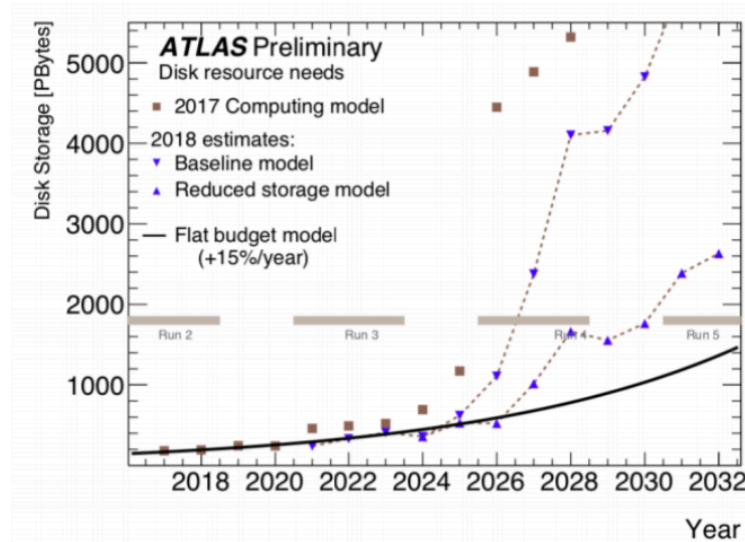
Wide range of software: from professional "products" (ROOT, Geant4, SHERPA,...) to collaboration specific million lines-of-code reconstruction frameworks (Athena, CMSSW) to one-off analysis code / shell scripts.



Wide ranging spectrum of "process" of writing and maintaining software.

Software is of course central to everything we do.

The big challenge ahead of us: HL-LHC



Software needs too rise to the occasion on all levels.
Community needs to be equipped with infrastructure and training to do so.

Software is of course central to everything we do.

HEP Software Foundation

- **founded in 2015 to tackle these challenges / to provide a forum for sharing ideas, experience and code between experiments**
 - **Encourage best practice for development Both at the algorithmic and tools level**
- Most of this work happening in the context of HSF Working Groups**

- Data Analysis
- Detector Simulation
- Frameworks
- Physics Generators
- PyHEP - Python in HEP
- Reconstruction and Software Triggers
- Software Developer Tools and Packaging
- Training



Software is of course central to everything we do.

Community White Paper process

- common document to lay out plans for the next 10 / 20 years in all areas of software & computing.

Community White Paper Reports

The roadmap summarised reports from fourteen working groups who studied the challenges in their sub-domains. All of the reports produced during the Community White Paper process are listed below. Working groups are in the process of **finalising** and **uploading** their work to ar

Paper	Report Number	Link
CWP Roadmap	HSF-CWP-2017-01	arXiv
Careers & Training	HSF-CWP-2017-02	arXiv
Conditions Data	HSF-CWP-2017-03	arXiv
Data Organisation, Management and Access	HSF-CWP-2017-04	arXiv
Data Analysis and Interpretation	HSF-CWP-2017-05	arXiv
Data and Software Preservation	HSF-CWP-2017-06	arXiv
Detector Simulation	HSF-CWP-2017-07	arXiv
Event/Data Processing Frameworks	HSF-CWP-2017-08	arXiv
Facilities and Distributed Computing	HSF-CWP-2017-09	Google Doc
Machine Learning	HSF-CWP-2017-10	arXiv
Physics Generators	-	No separate paper, see CWP Roadmap , section 3.1
Security	-	No separate paper, see CWP Roadmap , section 3.13
Software Development, Deployment and Validation	HSF-CWP-2017-13	arXiv
Software Trigger and Event Reconstruction	HSF-CWP-2017-14	arXiv - Executive Summary ; arXiv - full document
Visualisation	HSF-CWP-2017-15	arXiv

<https://hepsoftwarefoundation.org/organization/cwp.html>

HSF-CWP-2017-01
December 15, 2017

A Roadmap for HEP Software and Computing R&D for the 2020s

HEP Software Foundation¹

ABSTRACT: Particle physics has an ambitious and broad experimental programme for the coming decades. This programme requires large investments in detector hardware, either to build new facilities and experiments, or to upgrade existing ones. Similarly, it requires commensurate investment in the R&D of software to acquire, manage, process, and analyse the shear amounts of data to be recorded. In planning for the HL-LHC in particular, it is critical that all of the collaborating stakeholders agree on the software goals and priorities, and that the efforts complement each other. In this spirit, this white paper describes the R&D activities required to prepare for this software upgrade.

¹Authors are listed at the end of this report.

arXiv:1712.06982v5 [physics.comp-ph] 19 Dec 2018

Software is of course central to everything we do.

Community White Paper process

- common document to lay out plans for the next 10 / 20 years in all areas of software & computing.

Community White Paper Reports

The roadmap summarised reports from fourteen working groups who studied the challenges in their sub-domains. All of the reports produced during the Community White Paper process are listed below. Working groups are in the process of finalising and uploading their work to ar

Paper	Report Number	Link
CWP Roadmap	HSF-CWP-2017-01	arXiv
Careers & Training	HSF-CWP-2017-02	arXiv
Conditions Data	HSF-CWP-2017-03	arXiv
Data Organisation, Management and Access	HSF-CWP-2017-04	arXiv
Data Analysis and Interpretation	HSF-CWP-2017-05	arXiv
Data and Software Preservation	HSF-CWP-2017-06	arXiv
Detector Simulation	HSF-CWP-2017-07	arXiv
Event/Data Processing Frameworks	HSF-CWP-2017-08	arXiv
Facilities and Distributed Computing	HSF-CWP-2017-09	Google Doc
Machine Learning	HSF-CWP-2017-10	arXiv
Physics Generators	-	No separate paper
Security	-	No separate paper
Software Development, Deployment and Validation	HSF-CWP-2017-13	arXiv
Software Trigger and Event Reconstruction	HSF-CWP-2017-14	arXiv - Executive Summary ; arXiv - full document
Visualisation	HSF-CWP-2017-15	arXiv

- trend toward integration with standard industry tools
- make our own software more modular / inter-operable

HSF-CWP-2017-01
December 15, 2017

D

imental programme
stments in detector

hardware, either to build new facilities and experiments, or to upgrade existing ones. Similarly, it requires commensurate investment in the R&D of software to acquire, manage, process, and analyse the shear amounts of data to be recorded. In planning for the HL-LHC in particular, it is critical that all of the collaborating stakeholders agree on the software goals and priorities, and that the efforts complement each other. In this spirit, this white paper describes the R&D activities required to prepare for this software upgrade.

arXiv:1712.06982v5

<https://hepssoftwarefoundation.org/organization/cwp.html>

¹Authors are listed at the end of this report.

Preparing your software to be part of a larger community.

CODE

Traditionally has been undervalued in academic s/w (esp personal projects).

- It took a lot of work to correct this situation for the LHC experiments' code
- If it works, getting CERN (or your host lab) to hold copyright for the code works very well (a single copyright holder makes any relicensing easier)
- Collaborations or bodies like HSF cannot hold copyright
- Various Licenses Possible:
 - <https://opensource.org/licenses>
 - Good idea to choose such that your s/w can be easily reused / integrated into other software.

Community profile

Here's how this project compares to [recommended community standards](#).

Checklist

✓ Description	
✓ README	
✓ Code of conduct	
✓ Contributing	
✓ License	
✓ Issue templates	<input type="button" value="Edit"/>
✓ Pull request template	
✓ Repository admins accept content reports	

What is the community profile?

Preparing your software to be part of a larger community.

CONTRIBUTING

Having code "open source" in-name only without expectation / invitation to contribute generally seen as anti-pattern.

Should provide clear guidelines for new contributors no

- how to raise issues
- develop & test project code
- contribute code (pull requests)



Community profile

Here's how this project compares to [recommended community standards](#).

Checklist

- ✓ Description
- ✓ README
- ✓ Code of conduct
- ✓ Contributing
- ✓ License
- ✓ Issue templates
- ✓ Pull request template
- ✓ Repository admins accept

acts-project / acts

Example: ACTS Inter-experiment Tracking

<> Code Issues 56 Pull requests 15 Actions Security Insights

master acts / CONTRIBUTING.md

paulgessinger Contribution guide change (#256)

5 contributors

146 lines (104 sloc) | 13.6 KB

Contributing to Acts

Contributions to the Acts project are very welcome and fe

open for external contributors

Build Systems

A lot of our code is C++. Traditionally various versions of build systems used:

- in-house developments ("cmt", ..)
- autoconf tools (./configure; make...)

In the meantime CMake has become the dominant build system for C and C++ projects. Transitioned huge codebases to it.

Trainings for CMake from HSF/FIRST-HEP

This lesson is being piloted (Beta version)

Home Code of Conduct Setup Episodes Extras License Improve this page Search...

More Modern CMake

Welcome to the FIRST-HEP CMake tutorial! The aim of this tutorial is to cover the basics of using CMake. This tutorial is based on the online book [Modern CMake](#), with a focus on CMake 3.11+. This is almost what is called the "More Modern" era of CMake (which is 3.12+). We will cover the basics of making and building a project, and some details of design.

First taught at the [2019 USATLAS Computing Bootcamp at LBNL](#).

Prereqs

On your computer, you need to have:

- `git`
- `cmake` (Version 3.11 or newer). See the [instructions here](#).



Software Packaging:

Traditionally a lot of "source packaging" but big software projects provide e.g. conda recipes, RPMs, PyPI packages to install software with standard package managers.

```
conda install -c conda-forge root
yum install AnalysisBase
pip install uproot
```

CERN EP-SFT Group evaluated Spack as one of the most promising packaging tools for production use cases



A screenshot of the Scikit-HEP packaging guidelines page. The page has a light blue header with the Scikit-HEP logo and navigation links. The main content area is white with a blue sidebar on the left containing a table of contents. The main text discusses packaging standards, including a note on raw source code and a section on package structure (medium) that specifies the use of a src folder and the avoidance of installing from source. A section on PEP 517/518 support (high) is also visible at the bottom.

guidelines for python packages from Scikit-HEP

Software Distribution:

Traditional Approach (last 10 years or so):

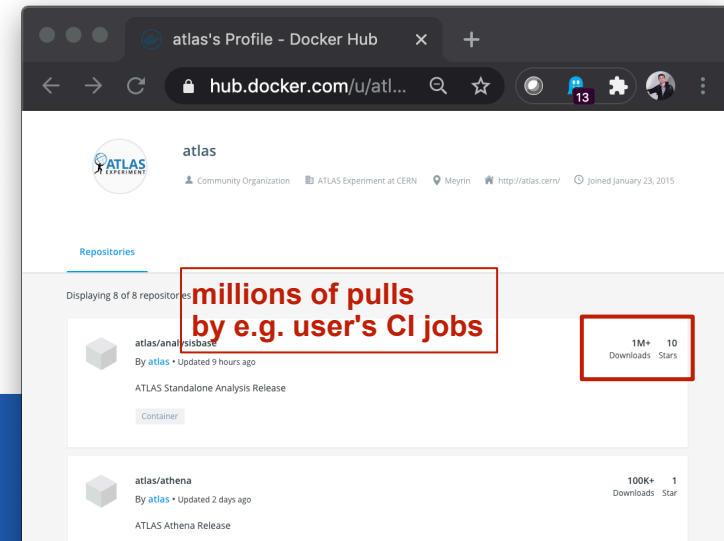
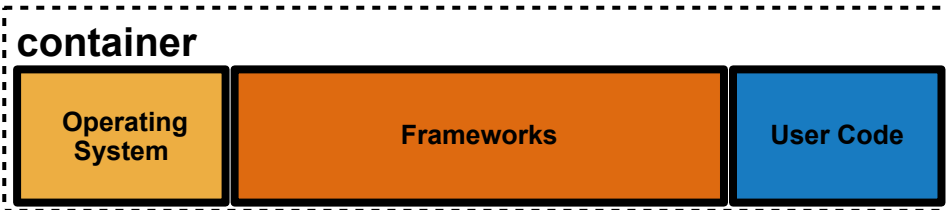
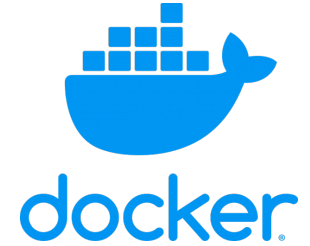


CVMFS: CERN VM Filesystem:

Model: read-only for most, write for few "publishers"

More recently: Everything is a Container.

- reproducible, portable standard unit
- easy integration into HPC, Cloud, Laptops...
- new development: efficient global distribution of images via cvmfs



Software Distribution:

Integration of Containers into workload mgmt systems

Allows developers increased flexibility to define their "own stack".

- Particularly useful for Machine Learning Application

16165474 task: user.lheinric.mlongriddemo.v7/															
Task ID	Jobset	Type	Working Group	User	Destination	Task status	Nevents used	HS06*sec Expected Total done failed	Ninputfiles finished failed	Average maxpss	Created	Modified	Cores	Priority	Parent
16165474	3728	analy		Lukas Alexander Heinrich		done	0 0 (%)	None 14016 14016	1 1 (100%)		2018-11-23 18:41:49	2018-11-23 19:17:29	1	1000	

Prodsys task parameters

architecture	
cliParams	<pre>prun --containerImage docker://lukasheinrich/atlasml:latest --exec "sh -c 'python /btagging/DL1_c_vs_b_slim.py %IN 10 gpu 500'" --inDS user.lheinric:mlongrid.v1 --outDS user.lheinric.mlongriddemo.v7 --noBuild --site ANALY_MANC_GPU_TEST --forceStaged</pre>

Software Citation:

CITATION

Software is often the research product itself. Should be treated as part of the scholarly record.

- cite software directly instead of "software papers" to attribute proper credit
- if you need a paper consider JOSS

CERN runs free service to mint DOI deposit code, datasets: ZENODO

JOSS
The Journal of Open Source Software

Journal of Open Source Software

The Basic Model Interface 2.0: A standard interface for coupling numerical models in the geosciences

Eric W.H. Hutton¹, Mark D. Piper¹, and Gregory E. Tucker^{1, 2, 3}

¹ Community Surface Dynamics Modeling System, University of Colorado Boulder, USA ² Cooperative Institute for Research in Environmental Sciences (CIRES), University of Colorado Boulder, USA ³ Department of Geological Sciences, University of Colorado Boulder, USA

DOI: 10.21105/joss.02317

Software

- Review [it](#)
- Repository [it](#)
- Archive [it](#)

Editor: Patrick Deligi [it](#)

Reviewers:

- @yanghu90
- @shuan

Submitted: 20 May 2020
Published: 23 July 2020

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Component modeling is a research technique in which new models are constructed by coupling the inputs and outputs of simpler existing models. Component modeling traces its roots to component-based software engineering, where a software system is constructed from a number of independent, reusable software components, each encapsulating a unit of functionality and exposing inputs and outputs through an interface. A tangible analogy is a bicycle. A bicycle is a system of reusable, replaceable components. Tires are one of the components. You can easily swap in a studded tire for icy winter streets, then swap it out again in the summer.

While there is a longer history of component modeling in fields such as climate modeling, with, for example, the Earth System Modeling Framework (Collins et al. 2002), component modeling is relatively new to the earth surface processes community. Recent examples include Rustill, Hutton, & Murray (2018), who show that a river model can be extended to can feed a delta model that distributes the sediment, and Hoch, Ettinger, & Gurnell (2019), Wisemanis (2019), who show that coupling hydrologic and hydrodynamic models can improve inundation estimates in flood modeling.

In component-based software engineering, components communicate through interfaces: named sets of functions with prescribed arguments and return values. The bicycle analogy above benefits from a standard interface for tire diameter and width. Likewise, component modeling can benefit from an interface for describing the inputs, outputs, and behaviors

maintaining BMI language specifications and examples. Prefix the repository name to obtain (<https://github.com/csdms/>)

Language	Specification	Example implementation
fortran	fortran	fortran
python	python	python
c	c	c
fortran	fortran	fortran
python	python	python

supports the four languages listed in Table 1, a BMI can be created for community-driven standard contributions that follow the contributor

[https://github.com/csdms/bmi-specification](#), and see [https://github.com/csdms/bmi-specification](#)

Extremely short "paper" main focus on code

Boost Histogram

for Python

zenodo

July 16, 2020

scikit-hep/boost-histogram: Version 0.10.0

Henry Schreiner; Hans Dembinski; Nino; Chanchal Kumar Maji; Doug Davis; Pierre Grimaud

This version was released during PyHEP 2020. Several improvements were made to usability when plotting and indexing.

User changes

- AxesTuple array now support operations via ArrayTuple #414
- Support `sum` and `bh.rebin` without slice #424
- Nicer error messages in some cases #415
- Made a few properties hidden for accumulators that were not public #418
- Boolean now supports reduction, faster compile #422
- AxesTuple now available publicly for subprojects #419

Bug fixes

- Histograms support operations with arrays, no longer take the first element only #417

DOI: 10.5281/zenodo.3948418

building docs passing build canceled

(source), a C++11 library. This is of the fast histogram object. See [what's new](#).

15

Example Projects:

generally can implement guidelines on all project scales:

Athena: ATLAS O(M) lines of core Reconstruction Code

<https://gitlab.cern.ch/atlas/athena>

- on-prem GitLab
- C++
- O(1000) contributors all from same collaboration
- Jenkins CI (moving to GitLab)
- RPM packages
- nightly tests
- docker images
- code linting
- citation available
- IDE integration
- Code Review
- ...

pyhf: statistics code

<https://github.com/scikit-hep/pyhf>

- GitHub
- 3 core developers with O(10) contributors
- code linting
- Github Actions CIs
- code auto-formatting (black)
- docs auto-generation in CI
- automated PyPI packaging
- O(1000) unit tests
- Test Coverage > 95%
- Code Review

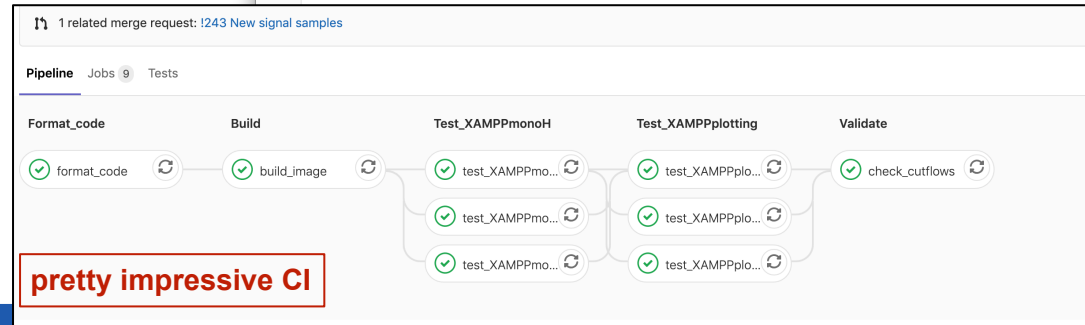
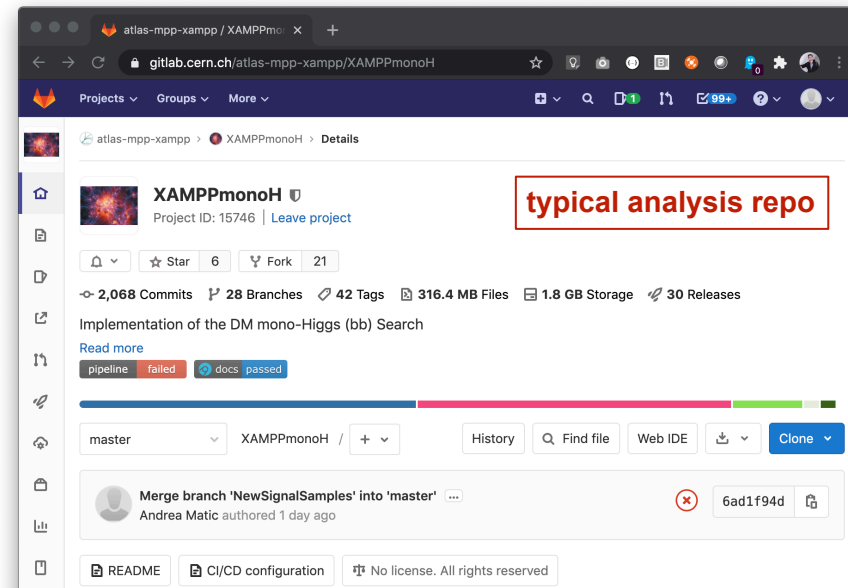
Role of Host Lab / (common entity / EU project) providing **infrastructure is crucial** to enable new developments.

Example 1 (Physical Infrastructure): **GitLab** introduced at CERN as new VC System. Analysis Teams required to have code in GitLab

Focus on integrated developer experience, on-prem deployment

Visible Change in attitudes from scientists to improve software creation cycle.

If it's easy to use & readily available, people become quite ambitious.



Role of Host Lab / (common entity / EU project) providing infrastructure is crucial to enable new developments.

Example 2 (Organizational Infrastructure): HSF/IRIS-HEP Training on Modern Software Development Tools

Topics: git, cmake,

Format:

- Software Carpentry
- Recorded Lessons (COVID)

Build up a library / catalogue of community-wide training material.

Again: if you offer it people will come: 200 sign-ups within days.

Time	Topic	Question
00:00	1. Introduction	
00:05	2. Exit (light) Codes	What is an exit code?
00:25	3. Understanding Yet Another Markup Language	What is YAML?
00:30	4. YAML and CI	What is the CI specification?
00:35	5. Coffee break	Get up, stretch out, take a short break.
00:50	6. Hello CI World	How do I run a simple CI job?
01:05	7. Adding CI to Your Existing Code	I have code already in GitLab, how can I add CI to it?
01:20	8. Eins Zwei DRY	How can we make job templates?
01:35	9. Coffee break	Get up, stretch out, take a short break.

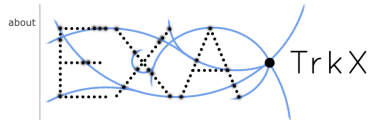
Role of Machine Learning:

Clear that ML will play increasing role in our software.

- we won't be driving the core software developments
- hardware increasingly targeted for ML

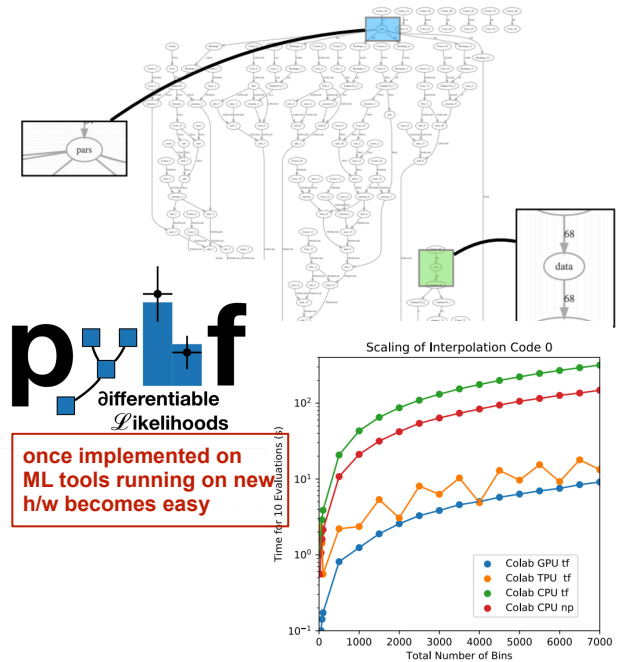
Strategies:

- cast existing problems as ML problems (e.g. tracking in Exa.TrkX)



HEP advanced tracking algorithms at the exascale (Project Exa.TrkX)

- instead of replacing our algorithms with ML, use ML foundation (highly vectorized computation + autodiff) for improved implementation (e.g. statistical fits)



Role of Quickly Changing Hardware

We're entering a new age of more dynamic hardware platforms.

- Not only GPU: FPGA, ASIC, Dataflow Engines,
- can we adapt / rethink our algorithms to match the h/w?
ML is one way, but not silver bullet.
 - SYCL, Alpaka, etc as underlying libraries for software portability
- even if we could, do we have the expertise to implement them? **Advanced Training of Experts is crucial**

Examples of Recent GPU initiatives:

- ALICE reco
- LHCb Allen (Trigger)
- CMS Patatrack

	OpenMP Offload	Kokkos	dpc++ / SYCL	HIP	CUDA	Alpaka	
NVidia GPU	Supported	Supported	Intel/codeplay	Supported	Supported	Supported	Supported
AMD GPU	Supported	prototype	via hipSYCL	Supported	Not Supported	Under Development	Under Development
Intel GPU	Supported	Supported	Supported	Not Supported	Not Supported	very early development	3rd Party
CPU	Supported	Supported	Supported	Not Supported	Not Supported	Supported	Not Supported
Fortran	Supported	Not Supported	Not Supported	3rd Party	3rd Party	Not Supported	Not Supported
FPGA	Supported	Supported	Supported	Not Supported	Not Supported	possibly via SYCL	Not Supported

HSF Frameworks and Reconstruction WGs gathering experience.

End User Software Re-use (Analysis Preservation)

For unique and large datasets such as those in HEP, the end-user software is often the only window to extract insight from a given dataset.

- software defined extraction of interesting data region
- If the software is gone the access to the "region" is gone as well: **need to preserve analyses**

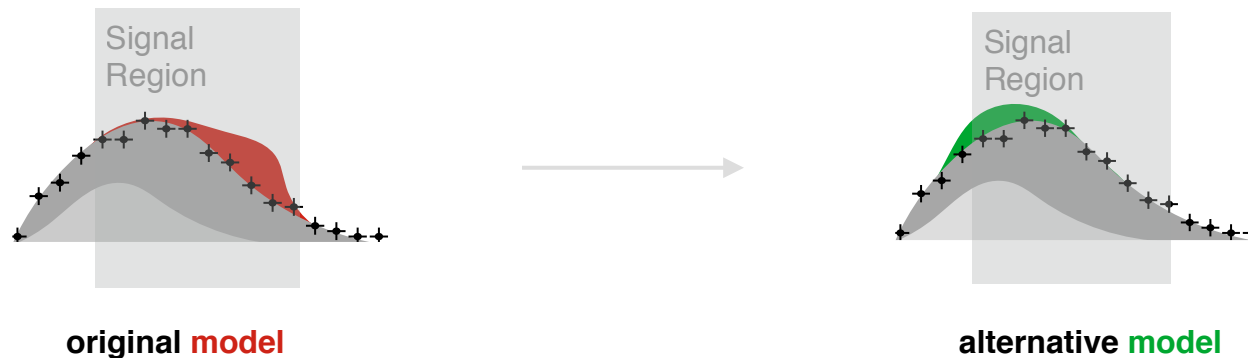
Ideally: continuously maintained code, but more realistically "fixed code" of final published analysis is all we can do

Marquee use-case: RECAST

Allow answer simple question: is a given theory already excluded by existing LHC analyses or do we need a dedicated study?

If analysis is preserved & functional: easy to answer

- simulate new model and pass through analysis to get a "reinterpreted result"



Simple Software Prereservation is not enough for Analysis Preservation: **need the full pipeline**

capture software

archive analysis code incl. dependencies

capture commands

what do with the captured software

capture workflow

order of individual steps

data assets

input data needed to run the analysis

Ingredients: Container Images, Workflow Languages

- similar trends in bioinformatics



CERN Working on Cyberinfrastructure to provide Archive of preserved analysis and compute resources to re-execute them (REANA)



CERN
Analysis Preservation

capture, preserve and reuse physics analyses

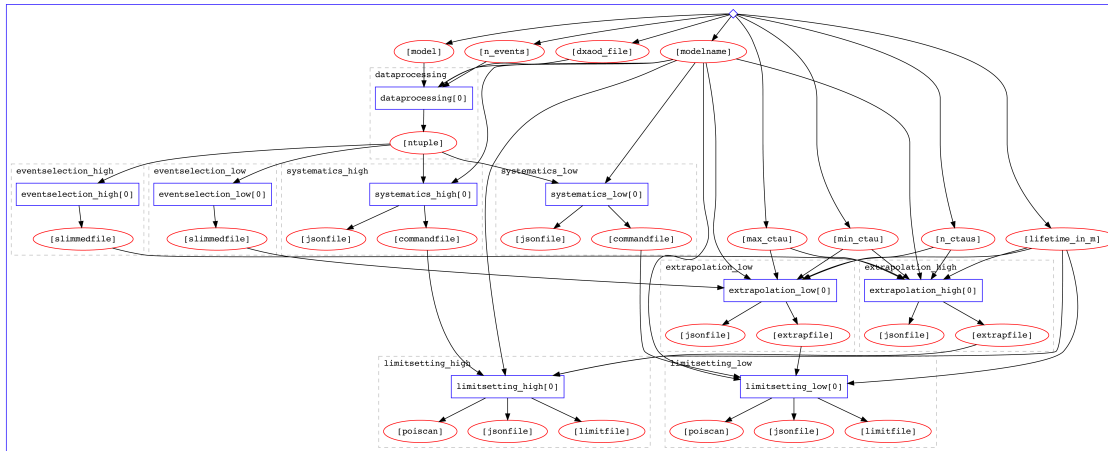


reana

Reproducible research data analysis platform

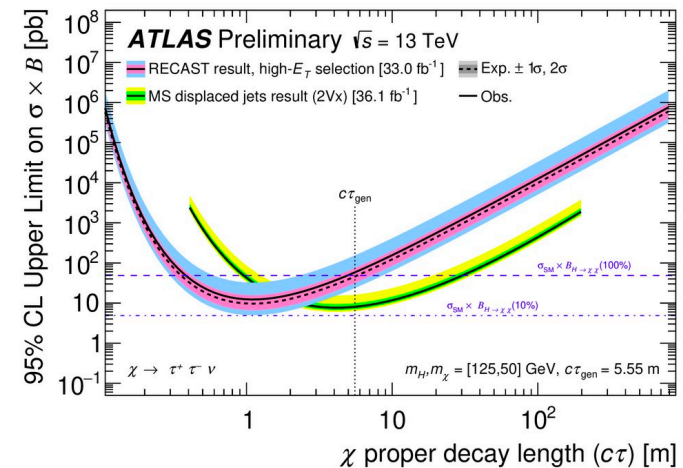
Success Stories: New Science out of Old Code :

- better scientific exploitation of data at low cost
- Examples from ATLAS: requirement on analysis teams to preserve analysis (docker images, workflows) leads to new results
- Potentially interesting ESCAPE test science case



Preserved Workflow (each node a Docker container workload)

ATL-PHYS-PUB-2020-007



New result!

Outlook

HSF founded to provide forum to discuss the big challenges in HEP software & computing (with many trends outside of our direct control: hardware, ML, data-science)

Provide / Develop Strategies and Tools for the full software lifecycle (authoring, build, package, distribute, preserve) to build a sustainable Software Ecosystem for HEP in a changing world.

Training is crucial at all levels: beginners (git, docker, cmake) to experts (accelerator programming, etc). HSF Provides

Given our unique data, the software itself becomes the product: increased attention to preservation, reuse, reproducibility

CERN plays a crucial role as the central Host Lab to provide infrastructure.