Sustainable Software in HEP and the HSF

Graeme A Stewart, CERN EP-SFT
for HSF Coordinators and WG Convenors

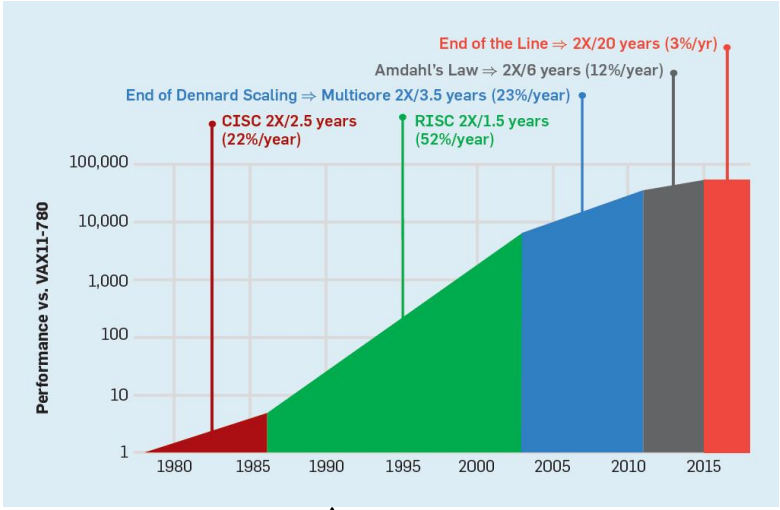
IRIS-HEP Blueprint Meeting, 2020-07-22

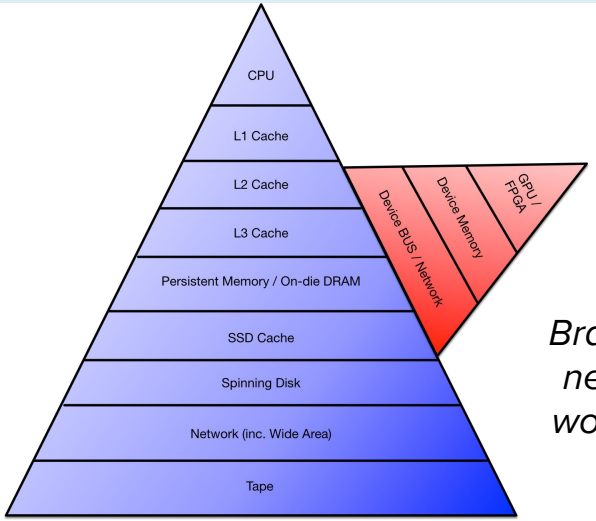
The HEP Software Foundation

- Organisation founded in 2015 to tackle common problems of software development and evolution in high-energy physics
- Challenges faced are well known:
  - LHC Run-3 data rates for ALICE and LHCb
  - HL-LHC programme starts at Run-4 for ATLAS and CMS
    - Large increases and event complexity and rate
  - Neutrino programme, particularly DUNE and the advent of massive TPC volumes
- HSF tries 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

Technological Challenges

- *Sustainability* of HEP software means also being able to evolve and adapt to new experimental conditions
  - This includes the constraints from the evolution of computing hardware
  - The advent of more and more ‘accelerated’ computing is diversifying hardware platforms beyond traditional CPUs
  - This, in turn, is bringing a plethora of SDKs and APIs for different platforms
- This adds significant complexity to the problem of sustainable code
  - Code not in standard C++ anymore





Brave new world

Software Tools and Best Practice

- It's much better to start off on the right foot than to try to correct a badly structured project later
- HSF has provided a project template, mainly aimed at C++ projects, that will get a lot of the structure right
  - License
  - Setup CMake correctly
  - Adding tests
  - Documentation hooks
- This has been used for a few projects, but could itself, benefit from some TLC...
- The HSF Best Practice Guide also describes setting up CI, release building, etc.
  - Also see the nice SciKit-HEP developer guide for similar (targeting Python)
- A good development environment for beginners can work wonders for productivity

Copyright and Licensing

- First of all, *copyright and licensing matter*, and we encourage everyone to sort these things out from day 1
  - 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)
  - It is in CERN's mission to disseminate knowledge as widely as possible
  - N.B. Collaborations or bodies like HSF cannot hold copyright
- There are many choices of open source license and authors should think about what they want to achieve with the license when deciding
  - Be aware that the GPL removes choice from people who might use your code and may inhibit reuse and collaboration
  - Otherwise, not so much practical difference between LGPL, MPL, Apache, MIT, BSD-3

CMake Use and Setup

- CMake has become the de-facto standard for C and C++ projects
- It helps a lot with package interoperability *if used correctly*
- Ben Morgan (HSF packaging and tools WG) provided an excellent guide to using modern CMake correctly
  - See the hilarious How To Package Managers Cry video for why this is important
  - Getting relocation correct
  - Allowing downstream packages to use your software
- Should also mention the very useful Introduction to Modern CMake authored by Henry Schreiner (IRIS-HEP)
  - https://cliutils.gitlab.io/modern-cmake/
  - And, oh look, here is the document source: https://gitlab.com/CLIUtils/modern-cmake
    - So please improve things with a PR... which is a good example of how to make things sustainable
  - And carpentry-style training material: https://henryiii.github.io/cmake_workshop/

Building Stacks

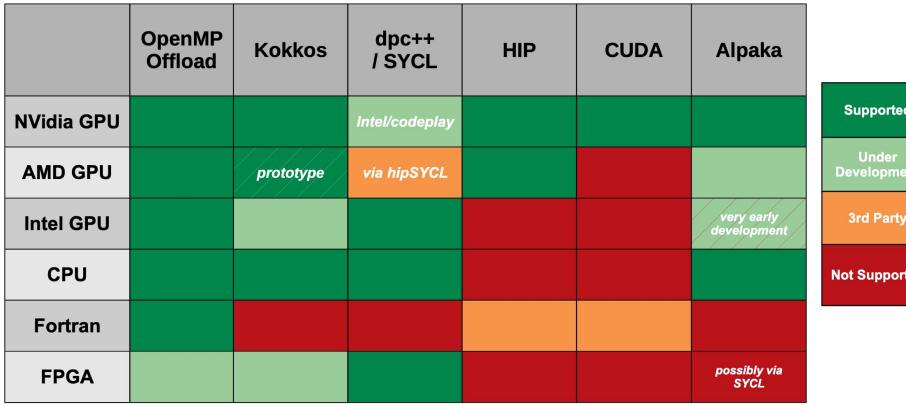
- We don’t just build single packages these days
  - Many dependent packages and a lot of utilities farmed out - *wide stack*
  - Performance and control usually push us to build from quite a low level - *deep stack*
- HSF Packaging Group spent a lot of time looking at build orchestration tools
- CERN EP-SFT group has the Software Process Integration project working on a transition to Spack (selected by HSF as one of the best tools available)
  - LLNL orchestration tool, originally for HPCs
  - Has been adapted for use by HEP, with *contributions from HEP*
    - Different use case, CVMFS installation for WLCG, e.g. relocatability
  - Benefit greatly from build recipes for generic software components coming from outwith HEP
- Key4hep project building robust flexible software for future detectors
  - Common language for algorithms: EDM4hep

HEP Reusable Software

- Many common problems faced by different experiments
  - Event generation
  - Detector simulation
  - Reconstruction
  - Analysis
- Common software that can sustain a larger user base will usually have a better chance of being sustained over time
  - ROOT and Geant4 are examples of code bases that are common for multiple experiments and have hugely helped HEP at the LHC and beyond
- Recent trend towards more use of modular components that can form more of a ‘toolkit’ ecosystem, e.g.,
  - DD4hep (detector description), Acts (tracking), SciKit-HEP (Python ecosystem for analysis)
  - Can see the direct impact of funding from, e.g., AIDA, IRIS-HEP, EP R&D

Sustainability on Uncertain Hardware

- HEP pushed to really maximise code performance to support physics programme by running on GPUs
  - ALICE reconstruction
  - LHCb HLT1 Allen project
  - CMS Patatrack
- More and more use of different accelerators to do this
- Coupled with latest generations of HPCs
  - But fundamentally driven by the underlying technology

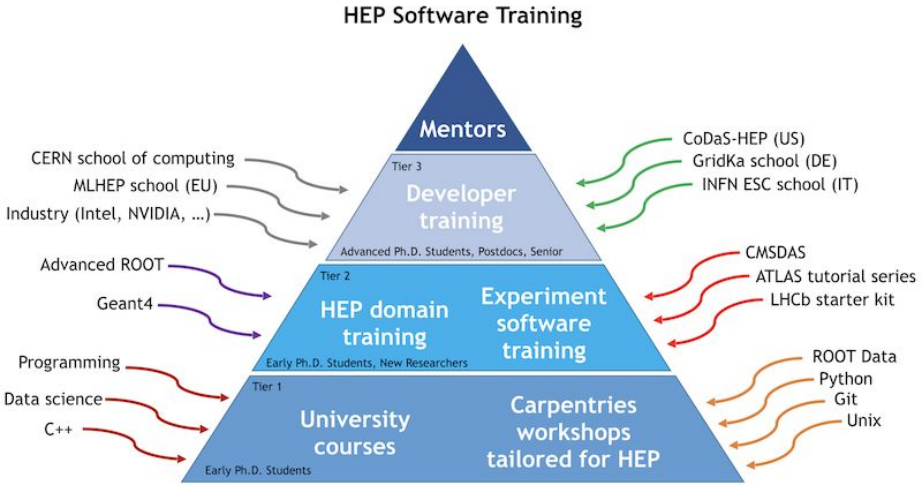


Matrix of support for different heterogeneous toolkits

- Hardest thing is rethinking algorithms!
  - But code portability and sustainability is a long term consideration
- DOE HEP-CCE project on Portable Parallelisation Strategies looking at this
- HSF Frameworks and Reconstruction WGs gathering experience and exchanging ideas

Training

- Software needs people
- So having cohorts of good programmers in our field is key to underpinning any successful software sustainability strategy
- Range of training encapsulated as a pyramid
  - Broad and common base
  - Rising up through more specialist levels
- Training WG has a programme to directly organise some training and support and sustain other efforts
  - Working with The Carpentries (non-HEP) as well as IRIS-HEP and FIRST-HEP
  - Adopting lesson templates
  - Up to advanced training (Alpaka heterogeneous toolkit, co-organised with CASUS)



From David Lange (IRIS-HEP, FIRST-HEP)

Outlook

- Software sustainability is a multifaceted issue
  - From hardware platforms, through software to the human factor
  - *As well as some (open) data to run on!*
- HSF is helping on many fronts, in concert with other projects: IRIS-HEP, ExCALIBUR, EP R&D, …
  - HSF Working Groups cover a large spectrum of the key software areas for HEP
  - We hope to be a ‘multiplier’ where effort exists; and to help to attract new effort as well
- Correct tooling makes life much easier
  - Support materials for this kind of sustainability do exist
  - Trying to encourage people to commit time to them can be hard
- Training is a key issue
  - We want to establish a broad portfolio of courses where best practice and sustainability are built-in
  - Key goal is to make training itself sustainable (reusable material and course templates)
- …but also long term career prospects