Meeting Future Software Challenges in High-Energy Physics

Graeme A Stewart, CERN EP-SFT

RAL PPD Seminar 2020-07-22

HL-LHC, the Intensity Frontier, and beyond

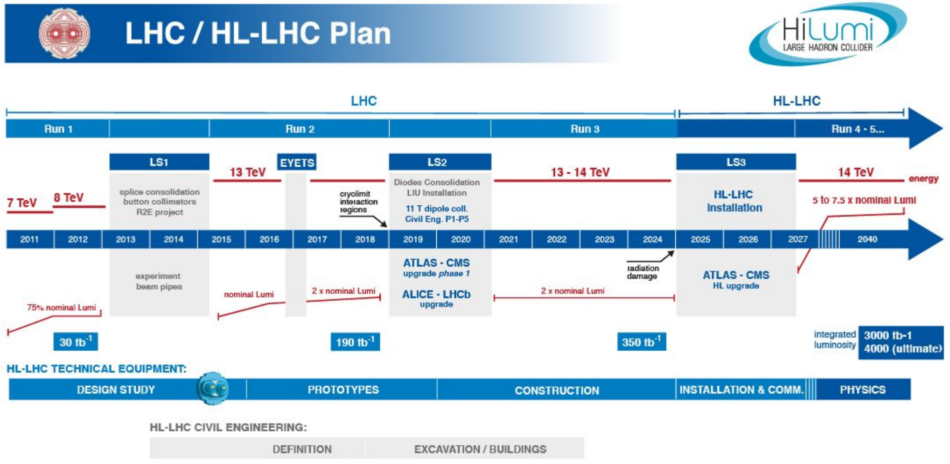
European Strategy Update

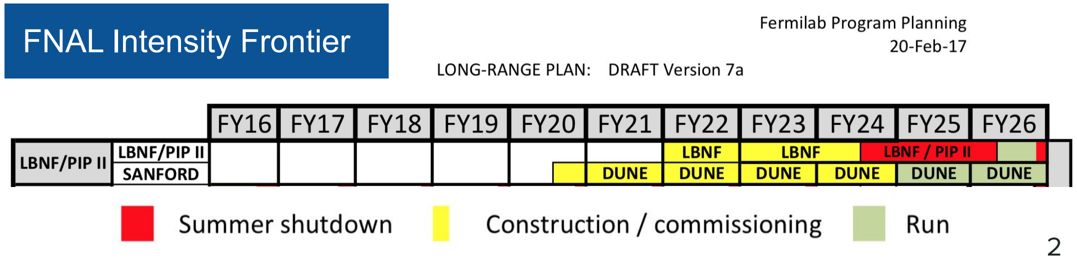
Our mission:

- Exploit the Higgs for SM and BSM physics
- b, c, tau physics to study BSM and matter/antimatter
- Dark matter
- QGP in heavy ion collisions
- Neutrino oscillations and mass
- Explore the unknown

Our Tools:

- (HL-)LHC, DUNE, Belle II
- ILC, FCC, CEPC, ~~CLIC~~





FNAL Intensity Frontier

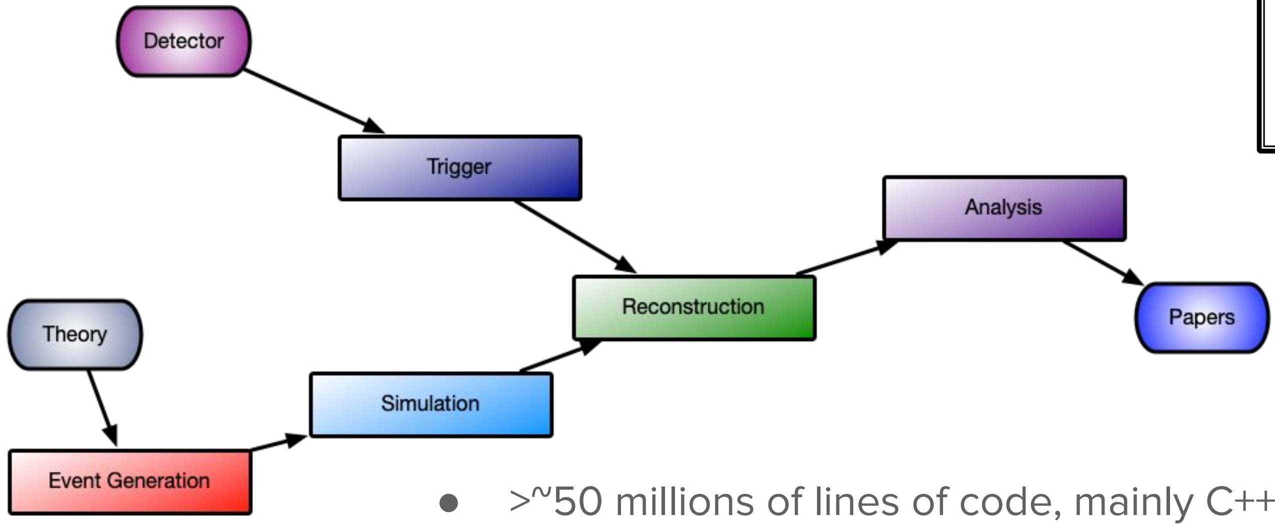
Fermilab Program Planning
20-Feb-17

LONG-RANGE PLAN: DRAFT Version 7a

		FY16	FY17	FY18	FY19	FY20	FY21	FY22	FY23	FY24	FY25	FY26
LBNF/PIP II	LBNF/PIP II							LBNF	LBNF	LBNF / PIP II		
	SANFORD						DUNE	DUNE	DUNE	DUNE	DUNE	DUNE

🟥 Summer shutdown 🟨 Construction / commissioning 🟩 Run

An Overview of HEP Software

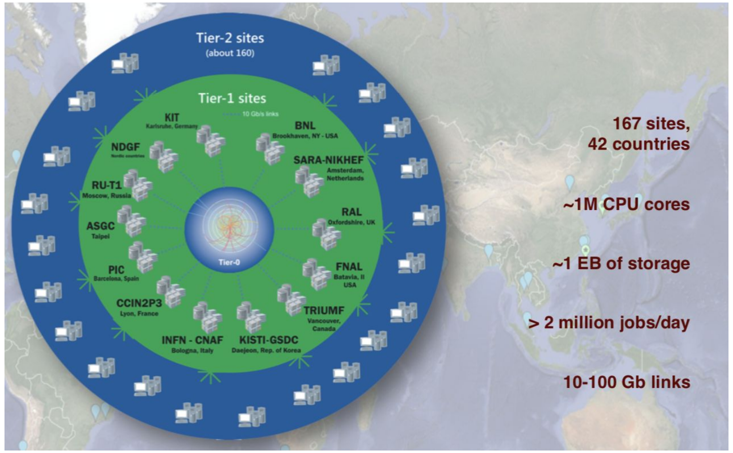


This is the “traditional” view and **how this changes** in the future is an important topic for our discussions

- >~50 millions of lines of code, mainly C++, a lot of Python
  - Commercial development cost ~500M CHF
- Critical part of our physics production pipeline, from triggering all the way to analysis and final plots as well as simulation
- Significant pieces of software are already shared by most experiments:
  - Event generators, Geant4, ROOT

HEP Computing

- Tasks broken into jobs by experiment production systems (levels of parallelism)
  - Tasks ➔ job ➔ events ➔ algorithms
- LHC experiments use
  - 1M CPU cores every hour of every day
  - Store 1000PB of data (600/400PB tape/disk split)
    - We are in the exabyte era already
  - 100PB of data transfers per year (10-100Gb links)
- This is a huge and ongoing cost in hardware and human effort
- With significant challenges ahead of us to support our ongoing physics programme



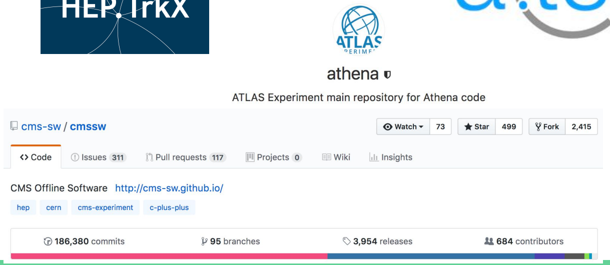
ROOT Data Analysis Framework

Geant 4

DD4hep

HEP.TrkX

acts


athena

ATLAS Experiment main repository for Athena code

cms-sw / cmssw Watch 73 Star 499 Fork 2,415

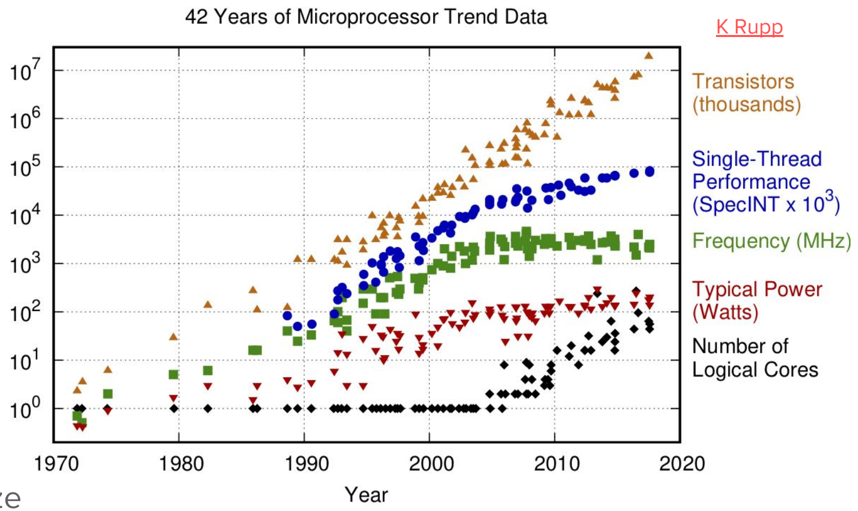
Code Issues 311 Pull requests 117 Projects 0 Wiki Insights

CMS Offline Software http://cms-sw.github.io/

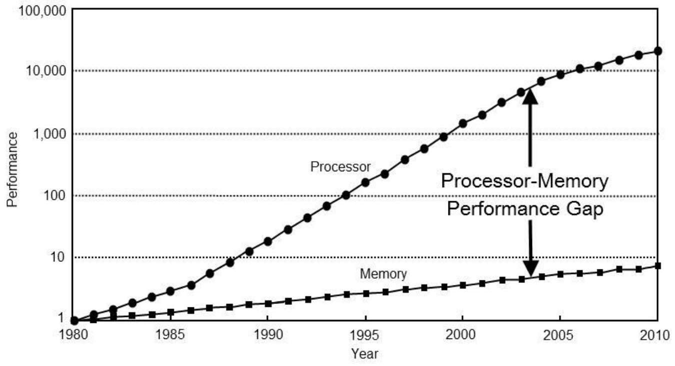
hep cern cms-experiment c-plus-plus

186,380 commits 95 branches 3,954 releases 684 contributors

Technology Evolution

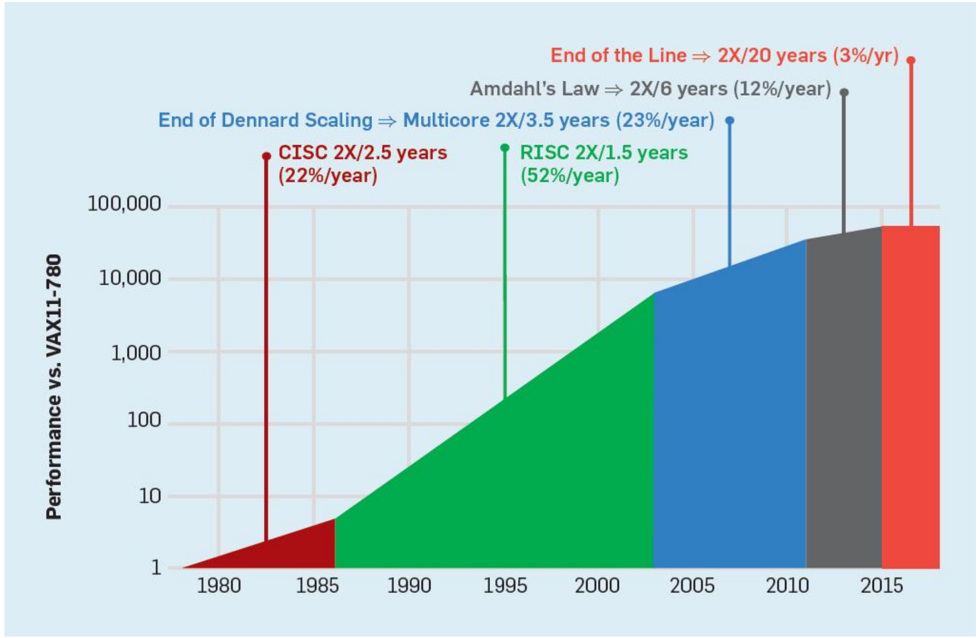


- Moore's Law continues to deliver increases in transistor density
  - But, doubling time is lengthening
- Clock speed scaling failed around 2006
  - No longer possible to ramp the clock speed as process size shrinks
  - Leak currents become important source of power consumption
- So we are basically stuck at ~3GHz clocks from the underlying Wm^{-2} limit
  - This is the *Power Wall*
  - Limits the capabilities of serial processing
- Memory access times are now ~100s of clock cycles
  - Poor data layouts are catastrophic for software performance

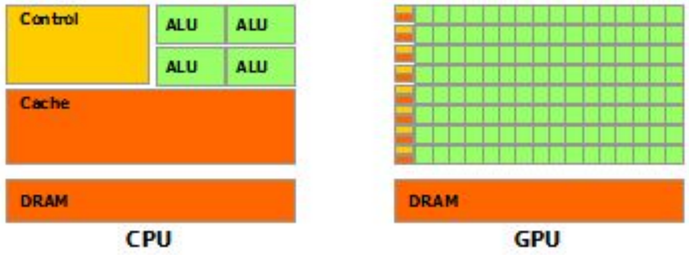


Decreasing Returns over Time

- ACM Conclusion: diversity of new architectures will only grow
- Best known example is of GPUs
  - Also FPGAs, TPUs, ...
- *A64FX ARM CPU recently took the #1 crown by fixing the memory latency issue*

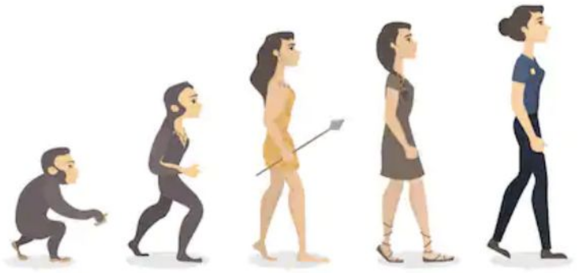


End of the Line ⇒ 2X/20 years (3%/yr)
Amdahl's Law ⇒ 2X/6 years (12%/year)
End of Dennard Scaling ⇒ Multicore 2X/3.5 years (23%/year)
CISC 2X/2.5 years (22%/year)
RISC 2X/1.5 years (52%/year)
Performance vs. VAX11-780
100,000
10,000
1,000
100
10
1
1980
1985
1990
1995
2000
2005
2010
2015



GPUs dedicate far more transistors to arithmetic

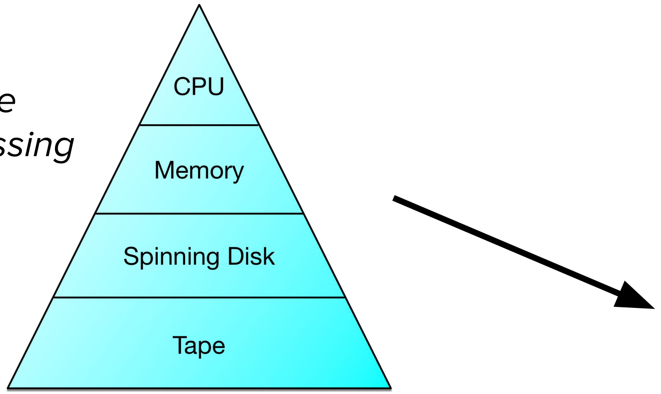
Drivers of Technology Evolution

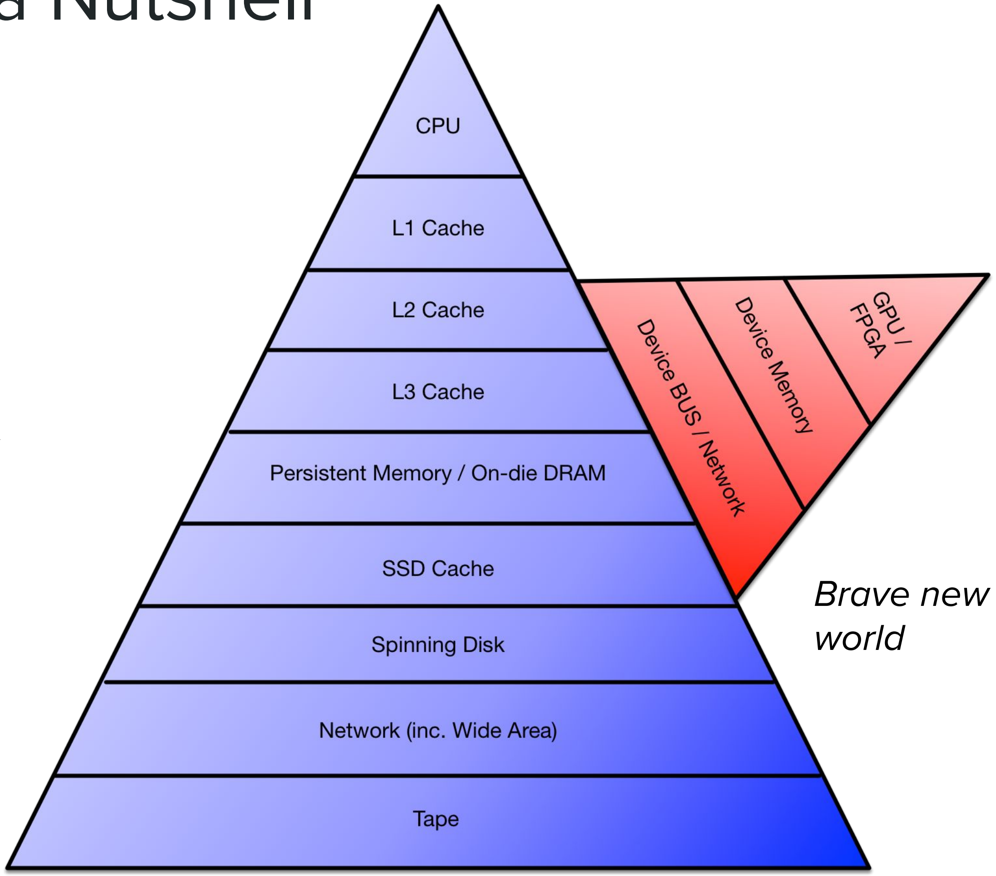


- Low power devices
  - Driven by mobile technology and Internet of Things
- Data centre processing
  - Extremely large clusters running fairly specialist applications
- Machine learning
  - New silicon devices specialised for training machine learning algorithms, particularly low precision calculations
- Exascale computing
  - Not in itself general purpose, but poses many technical problems whose solutions can be general - HEP pushed to use HPC centres, especially in US
    - New Top500 #1 Japanese Fugaku machine is really interesting…
- Energy efficiency is a driver for all of these developments
  - Specialist processors would be designed for very specific tasks
  - Chips would be unable to power all transistors at once: dark silicon is unlit when not used

Hardware Evolution in a Nutshell

Ye olde processing model





Brave new world

“We’re approaching the limits of computer power – we need new programmers now”
John Naughton, Guardian

Software Challenges and Opportunities

Concurrency

- The one overriding characteristic of modern processor hardware is concurrency
    - SIMD - Single Instruction Multiple Data (a.k.a. vectorisation)
        - Doing exactly the same operation on multiple data objects
    - MIMD - Multiple Instruction Multiple Data (a.k.a. multi-theading or multi-processing)
        - Performing different operations on different data objects, but at the same time
    - SIMT - Single Instruction Multiple Threads
        - GPU running a block of threads in instruction lock-step (masking allowed, more flexible than SIMD)
- Because of the inherently parallel nature of HEP processing a lot of concurrency can be exploited at rough granularity
    - Run many jobs from the same task in parallel; Run different events from the same job in parallel
- However, the push to highly parallel processing (1000s of GPU cores) requires **parallel algorithms**
    - This often requires completely rethinking problems that had sequential solutions previously, e.g. finding track seeds via cellular automata (TrickTrack library, CMS and FCC)

Heterogeneity

- There are a lot of possible parallel architectures on the market
    - CPUs with multiple cores and wide registers
        - SSE4.2, AVX, AVX2, AVX512, Neon, SVE, Altivec/VMX, VSX
    - GPUs with many cores; FPGAs
        - NVIDIA (many generations - often significantly different), AMD, Intel, …
- In addition there are ‘far out’ architectures proposed, like Intel’s Configurable Spatial Architecture
- Many options for coding, both generic and specific:
    - CUDA, HIP, TBB, HPX, OpenACC, OpenMP, OpenCL, SYCL, Alpaka, Kokkos, oneAPI, …
- Frustratingly no clear winner, mutually exclusive solutions and many niches
    - One option for now is to isolate the algorithmic code from a ‘wrapper’ that targets a particular device or architecture - approach of ALICE for their GPU/CPU code
    - Hiding details in a lower level library (e.g. VecCore) also helps insulate developers

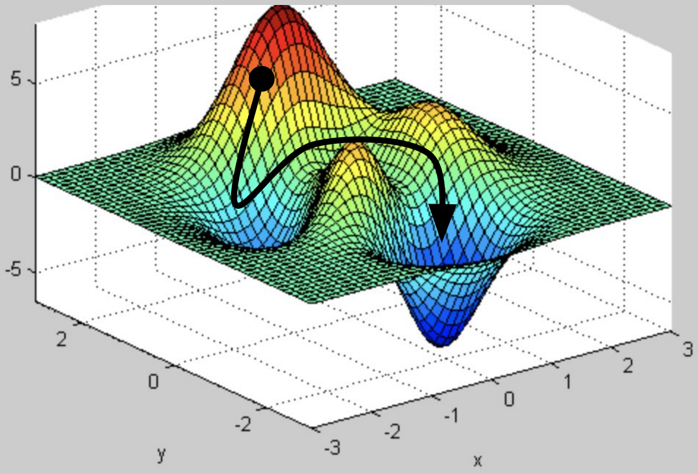
Data Layout and Throughput

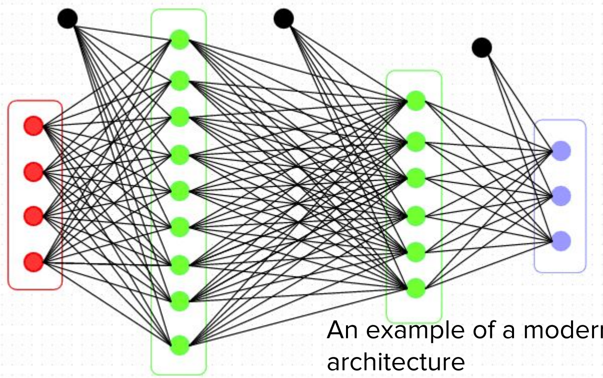
- Original HEP C++ Event Data Models were heavily inspired by the Object Oriented paradigm
  - Deep levels of inheritance
  - Access to data through various indirections
  - Scattered objects in memory
- Lacklustre performance was ~hidden by the CPU and we survived LHC start
- In-memory data layout has been improved since then (e.g. ATLAS xAOD)
  - But still hard for the compiler to really figure out what's going on
  - Function calls non-optimal
  - Extensive use of 'internal' EDMs in particular areas, e.g. tracking
- iLCSoft / LCIO also proved that common data models help a lot with common software development
- Want to be flexible re. device transfers and offer different persistency options
  - e.g. ALICE Run3 EDM optimised for message passing and the code generation approaches in FCC-hh PODIO EDM generator (being used at the implementation of common EDM4hep project)

Machine Learning

- Machine learning, or artificial intelligence, used for many years in HEP
    - Algorithms learn by example (training) how to perform tasks instead of being programmed
- Significant advances in the last years in ‘deep learning’
    - Deep means many neural network layers
    - Fast differentiability and use of GPUs have made this practical
- Rapid development driven by industry
    - Vibrant ecosystem of tools and techniques
    - *Highly optimised for modern, specialised hardware*

ML minimisation problem - do this minimisation with 10^6 variables...





An example of a modern ML architecture

Machine Learning in HEP

- Better discrimination
  - Important input for analysis (see improvements with Higgs)
  - Also used at HLT as inference can be fast (N.B. training can be slow!)
  - HEP analogies to image recognition or text processing
- Replace expensive calculations with trained output
  - E.g. calorimeter simulations and other complex physical processes
- There are significant opportunities here
  - Need to combine physics and data science knowledge
  - Field evolves rapidly and we need to deepen our expertise
  - New HSF initiative on differentiable computing
- Integration into our workflows is not at all settled
  - Resource provision, efficient use, heterogeneity and programming models pose problems
  - Training deep models may require *significant* resources
    - Especially when hyper-parameter scans are needed

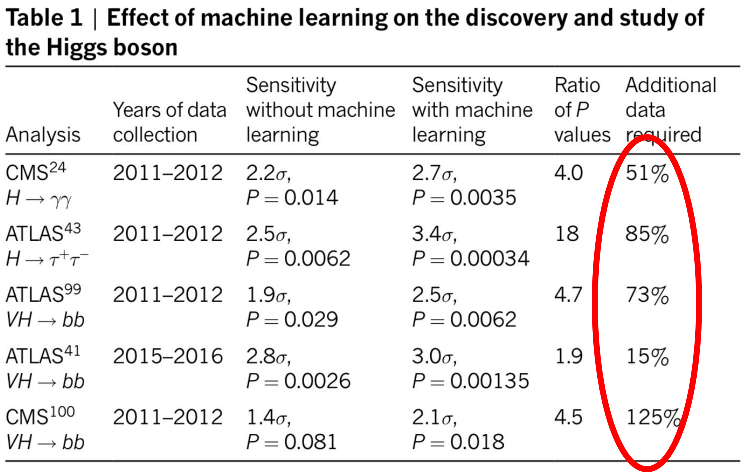
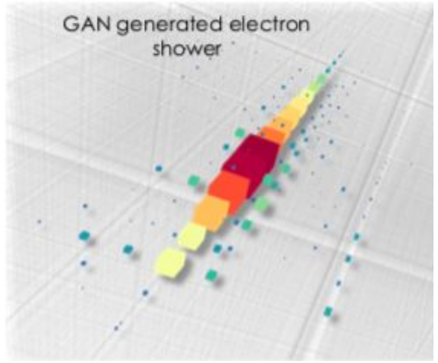


Table 1 | Effect of machine learning on the discovery and study of the Higgs boson

Analysis	Years of data collection	Sensitivity without machine learning	Sensitivity with machine learning	Ratio of P values	Additional data required
CMS[24] $H \rightarrow \gamma\gamma$	2011–2012	2.2σ, $P = 0.014$	2.7σ, $P = 0.0035$	4.0	51%
ATLAS[43] $H \rightarrow \tau^+\tau^-$	2011–2012	2.5σ, $P = 0.0062$	3.4σ, $P = 0.00034$	18	85%
ATLAS[99] $VH \rightarrow bb$	2011–2012	1.9σ, $P = 0.029$	2.5σ, $P = 0.0062$	4.7	73%
ATLAS[41] $VH \rightarrow bb$	2015–2016	2.8σ, $P = 0.0026$	3.0σ, $P = 0.00135$	1.9	15%
CMS[100] $VH \rightarrow bb$	2011–2012	1.4σ, $P = 0.081$	2.1σ, $P = 0.018$	4.5	125%

Machine learning at the energy and intensity frontiers of particle physics,

https://doi.org/10.1038/s41586-018-0361-2

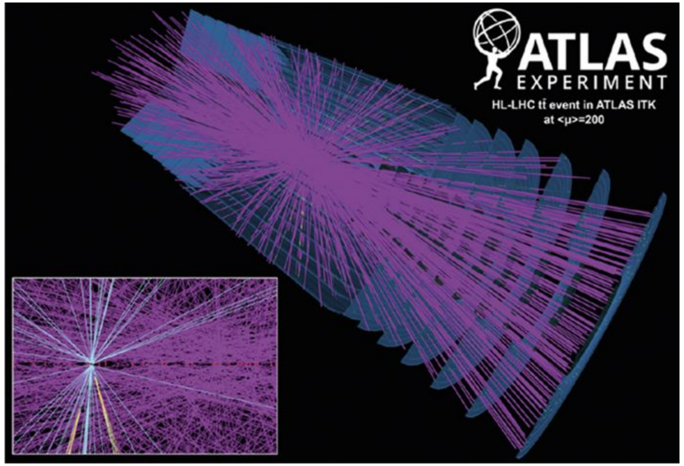


Use of Generative Adversarial Networks to simulate calorimeter showers, trained on G4 events (S. Vallacorsa)

HEP Software and Computing and the HSF Initiative

HEP Software Foundation

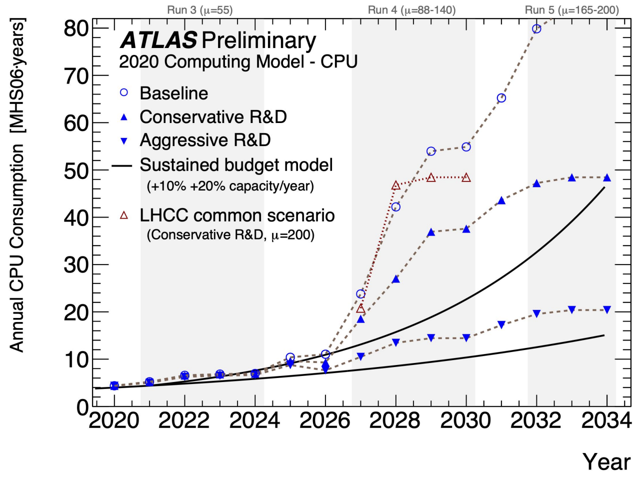
Software at the HL-LHC

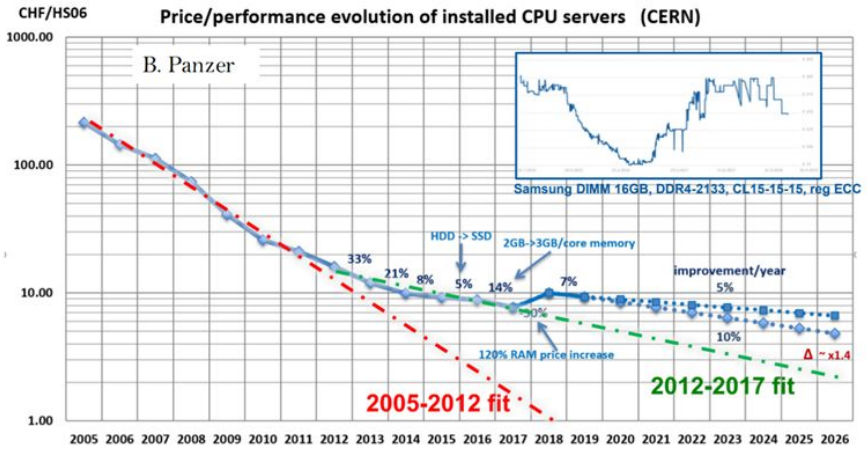


- Pile-up of ~200 ⇒ particularly a challenge for charged particle reconstruction
  - Inner trackers and high-granularity calorimeters
- Increase of event rates by up to x10 means the computing budget per event just went down by x10…
- Additional problem is how to store these events
  - Keeping today's event formats and volumes *will not work*
- Classical HEP software typically executes one instruction at a time (per thread)
  - Major SW re-engineering required (but rewriting everything is not an option)
  - Co-processors like GPUs *require* that this problem is solved
- Increased amount of data requires to revise/evolve our computing and data management approaches
  - We must be able to feed our applications with data efficiently
- *HL-LHC salvation will come from software improvements, not from hardware*

Challenges for the Next Decade

- HL-LHC brings a huge challenge to software and computing
  - Both rate and complexity rise
- No reasonable extrapolation of Run 2 software and computing
  - Resources needed would hugely exceed those from technology evolution alone
  - Considerable progress made from naive estimates, but still a large resource gap

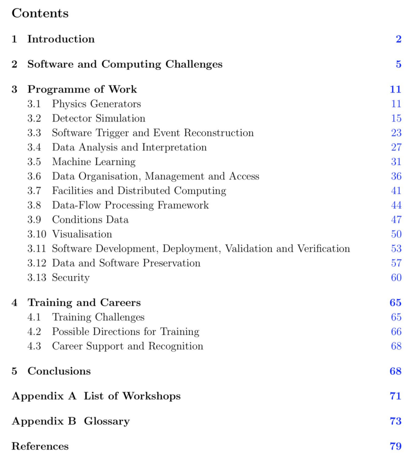




HEP Software Foundation (HSF)

- The LHC experiments, Belle II and DUNE face the same challenges
  - HEP software must evolve to meet these challenges
  - Need to exploit all the expertise available, inside and outside our community, for parallelisation
  - New approaches needed to overcome limitations in today’s code
- Cannot afford any more duplicated efforts
  - Each experiment has its own solution for almost everything (framework, reconstruction algorithms, ...)
  - New experiments should not be starting from scratch, but building on best-of-breed
- HSF started with a number of workshops and working groups on common topics (packaging, licensing)
- The goal of the HSF is to facilitate coordination and common efforts in software and computing across HEP in general
  - Our philosophy is bottom up, a.k.a. *do-ocracy*

Community White Paper

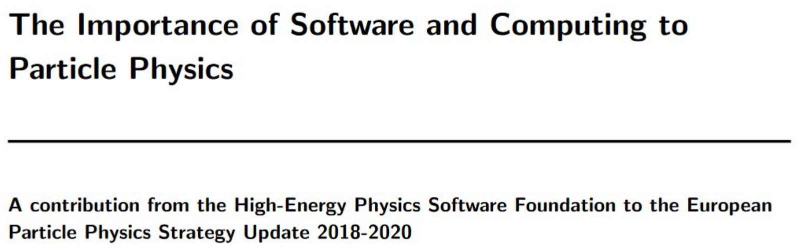


Contents

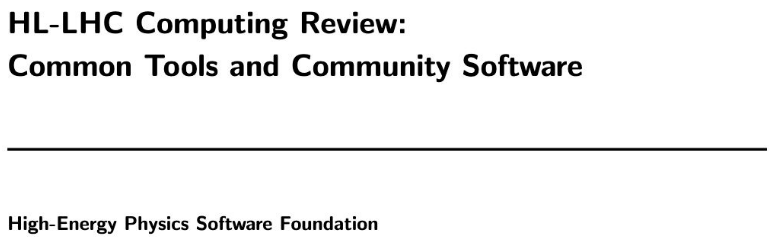
- We wanted to describe a **global vision for software and computing** for the HL-LHC era and HEP in the 2020s
- Formal charge from the WLCG in July 2016
  - Anticipate a "software upgrade" in preparation for HL-LHC
  - Identify and prioritize the software research and development investments
    1. to achieve improvements in software efficiency, scalability and performance and to make use of the advances in CPU, storage and network technologies
    2. to enable new approaches to computing and software that could radically extend the physics reach of the detectors
    3. to ensure the long term sustainability of the software through the lifetime of the HL-LHC
- Long process of 1 year, with many working groups and 2 major workshops
- Community engagement: 310 authors from 124 institutes, 14 chapters
- Published in *Computing and Software for Big Science*, https://doi.org/10.1007/s41781-018-0018-8 (and on arXiv)

Software Advocacy

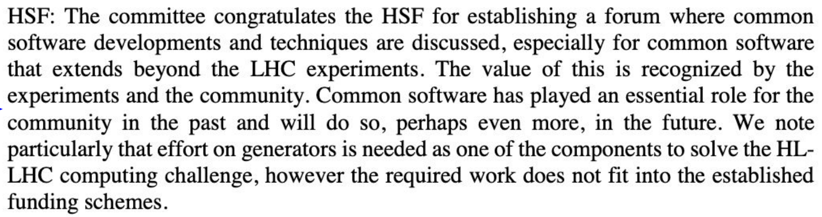
- HSF has continued to advocate for software at the highest levels of the field
- Paper submitted and talk at the European Strategy Process
- Paper on Common Tools and Community Software reviewed by the LHCC in May
    - Positive feedback in LHCC minutes

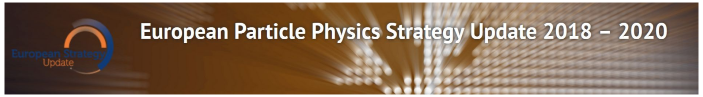

The Importance of Software and Computing to Particle Physics

A contribution from the High-Energy Physics Software Foundation to the European Particle Physics Strategy Update 2018-2020


HL-LHC Computing Review: Common Tools and Community Software

High-Energy Physics Software Foundation


HSF: The committee congratulates the HSF for establishing a forum where common software developments and techniques are discussed, especially for common software that extends beyond the LHC experiments. The value of this is recognized by the experiments and the community. Common software has played an essential role for the community in the past and will do so, perhaps even more, in the future. We note particularly that effort on generators is needed as one of the components to solve the HL-LHC computing challenge, however the required work does not fit into the established funding schemes.



European Strategy Update

2020 Strategy Statements

4. Other essential scientific activities for particle physics

Computing and software infrastructure

- There is a need for strong community-wide coordination for computing and software R&D activities, and for the development of common coordinating structures that will promote coherence in these activities, long-term planning and effective means of exploiting synergies with other disciplines and industry
- A significant role for artificial intelligence is emerging in detector design, detector operation, online data processing and data analysis
- Computing and software are profound R&D topics in their own right and are essential to sustain and enhance particle physics research capabilities
- More experts need to be trained to address the essential needs, especially with the increased data volume and complexity in the upcoming HL-LHC era, and will also help in experiments in adjacent fields.

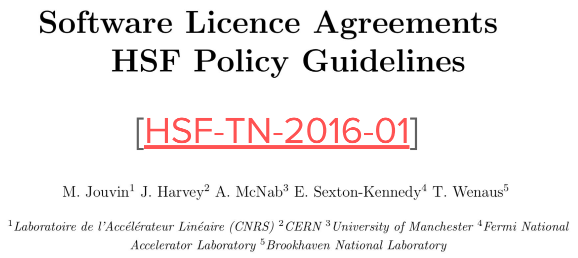
d) Large-scale data-intensive software and computing infrastructures are an essential ingredient to particle physics research programmes. The community faces major challenges in this area, notably with a view to the HL-LHC. As a result, the software and computing models used in particle physics research must evolve to meet the future needs of the field. *The community must vigorously pursue common, coordinated R&D efforts in collaboration with other fields of science and industry to develop software and computing infrastructures that exploit recent advances in information technology and data science. Further development of internal policies on open data and data preservation should be encouraged, and an adequate level of resources invested in their implementation.*

```
int main {
  cout << “write software” << endl;
  return 0;
}
```

HSF Working Groups

- The Roadmap established what challenges the community faced
    - But it did not spell out *how* to face them in detail
- HSF had working groups from its earliest days
    - These were open groups of people in the community, motivated enough to organise around a common topic, usually at their own initiative
- This model was a good one for moving forwards on the key topics
    - We setup new working groups for Detector Simulation, Reconstruction and Software Triggers, and Data Analysis
    - HSF Coordination group setup a search committee, whole community could nominate convenors
- The HSF's role here is one of an information conduit and meeting point
    - Report on interesting and common work being done
    - Forum for technical comments and discussion
    - Encourage cooperation across experiments and regions
- This model was a *real success* and was expanded last year to all working groups

Some important practical matters! Copyright and Licensing



Software Licence Agreements HSF Policy Guidelines

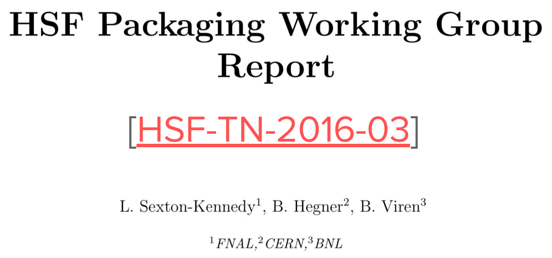
[HSF-TN-2016-01]

M. Jouvin[1] J. Harvey[2] A. McNab[3] E. Sexton-Kennedy[4] T. Wenaus[5]

[1]*Laboratoire de l'Accélérateur Linéaire (CNRS)* [2]*CERN* [3]*University of Manchester* [4]*Fermi National Accelerator Laboratory* [5]*Brookhaven National Laboratory*

- Long neglected inside collaborations
  - Code was arbitrarily licensed or unlicensed, copyright assigned to random authors and institutes
  - Yet this is essential to be able to
    - Open source our software properly
    - Combine with other open source projects and collaborate
- Copyright
  - Advice to keep this as low a number as practicable as copyright holders decide the licence
  - LHC experiments: © CERN for the benefit of collaboration X
- License
  - Favour liberal licenses for industry collaboration: LGPL, Apache, MIT
  - Definitely avoid GPL for libraries you want other people to use

Software Nuts and Bolts



HSF Packaging Working Group Report

[HSF-TN-2016-03]

L. Sexton-Kennedy[1], B. Hegner[2], B. Viren[3]

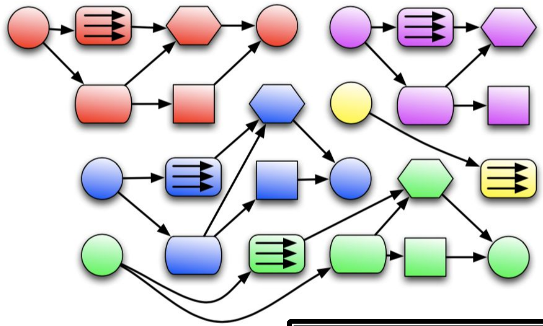
[1]FNAL,[2]CERN,[3]BNL

- Software Tools and Packaging WG
  - Active group promoting best practice for correctness and performance, developer productivity
  - There has been a revolution in adopting best open source practice in recent years
    - git, GitHub, GitLab, CMake, VS Code, merge requests, code review, …
  - HSF has an active group promoting best practice for correctness and performance
    - Profiling, static analysis, project build settings, …
  - Need a software stack, incorporating many components from the open source world and HEP community
    - This touches deeply on license and license combinations
  - Preference for tools that are not home grown and have a wider support base
  - Spack (LLNL) and Conda actively being prototyped, e.g. Key4hep project in EP R&D and AIDAinnova

Use Cases for Packaging and Deployment Tools

V1.0, 2018-03-21
10.5281/zenodo.3634722
HEP Software Foundation Packaging Group

Frameworks and Integration

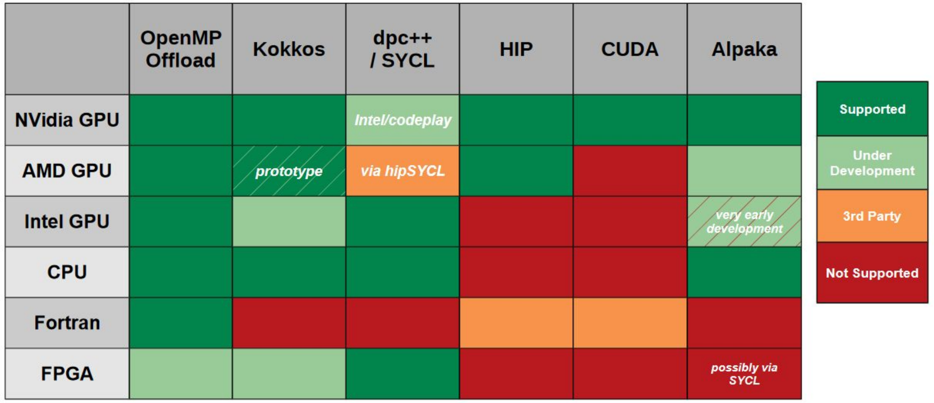


Cartoon of a single job, processing multiple events (colours) through different modules (shapes)

- Increasingly heterogeneous world requires advanced software support infrastructure
  - Software frameworks support use of different devices as well as insulate developers from many of the details of concurrency and threading models
    - Adapt to the new heterogeneous landscape
    - Latency hiding is critical to maintaining throughout
  - Framework development has traditionally been quite fragmented, but new experiments should offer a chance to increase convergence
    - Better to start off together than try to re-converge later (iLCSoft, LArSoft examples of success, albeit without concurrency; Gaudi for LHCb, ATLAS)
    - ALFA for ALICE and FAIR experiments
- New HSF working group established last year
  - Survey of the community and meetings on topical projects
  - Look at multi-threading
  - Now moving to heterogeneous resources

Issues for Heterogeneous Software

- Code Portability
  - Increasingly large number of possible non-CPU devices available
  - Clear that the community cannot support N codes for N platforms
  - Industry knows this too, hence proliferation of toolkits and projects
- How to assess the best?
  - This is an orthogonal question to redesigning code for at least one parallel architecture
- DOE HEP-CCE Project
  - *Portable Parallelization Strategies*
  - Assess metrics for toolkits on real HEP examples:
    - Patatrack (CMS),
    - FastCaloSim (ATLAS)
    - WireCell (Neutrino)
  - Will produce recommendations taking into account the nature of HEP workflows



	OpenMP Offload	Kokkos	dpc++ / SYCL	HIP	CUDA	Alpaka
NVidia GPU	🟩	🟩	▩ *Intel/codeplay*	🟩	🟩	🟩
AMD GPU	🟩	▨ *prototype*	🟧 *via hipSYCL*	🟩	🟥	▩
Intel GPU	🟩	▩	🟩	🟥	🟥	▨ *very early development*
CPU	🟩	🟩	🟩	🟥	🟥	🟩
Fortran	🟩	🟥	🟥	🟧	🟧	🟥
FPGA	▩	▩	🟩	🟥	🟥	🟥 *possibly via SYCL*

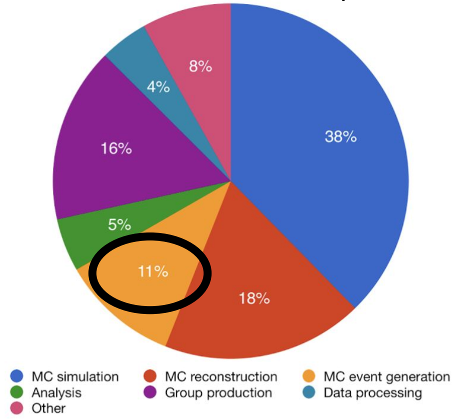
🟩	Supported
▩	Under Development
🟧	3rd Party
🟥	Not Supported

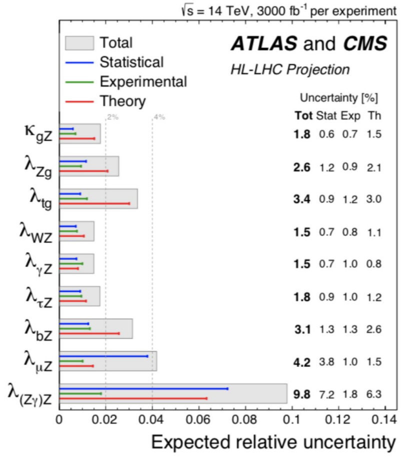
All this changes rapidly

Event Generators

- Event generators are the start of the simulation chain
  - At the LHC Run1 only leading order generators were used
  - Negligible CPU consumption compared with detector simulation
- However, with LHC upgrades coming higher order generators become much more important
  - These are inherently much more costly to run
  - Problems of *negative weights* can increase hugely the samples needed for weighted event samples
- In addition, the theory community, who develop these codes usually work in small teams
  - Recognition for technical improvements is limited/missing
- HSF/LPCC Workshop in 2018 helped spawned a new working group active in this area
- Additional paper submitted to LHCC Review [arXiv:2004.13687]

ATLAS 2018 CPU Report



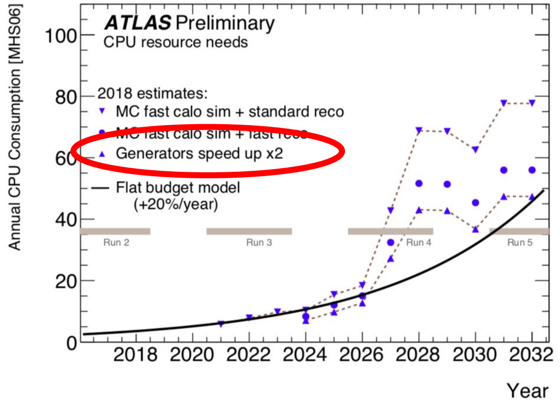


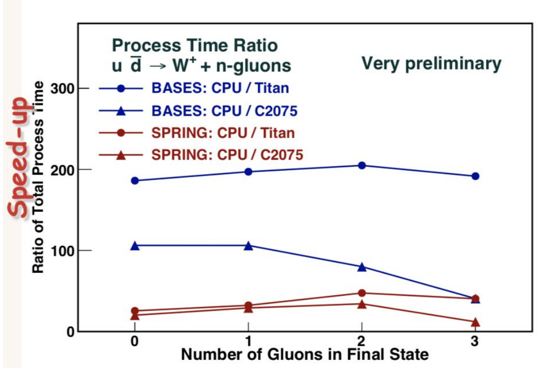
	Tot	Stat	Exp	Th
κ_{gZ}	1.8	0.6	0.7	1.5
λ_{Zg}	2.6	1.2	0.9	2.1
λ_{tg}	3.4	0.9	1.2	3.0
λ_{WZ}	1.5	0.7	0.8	1.1
$\lambda_{\gamma Z}$	1.5	0.7	1.0	0.8
$\lambda_{\tau Z}$	1.8	0.9	1.0	1.2
λ_{bZ}	3.1	1.3	1.3	2.6
$\lambda_{\mu Z}$	4.2	3.8	1.0	1.5
$\lambda_{(Z\gamma)Z}$	9.8	7.2	1.8	6.3

Many electroweak measurement errors dominated by theory (red). B. Hinemann

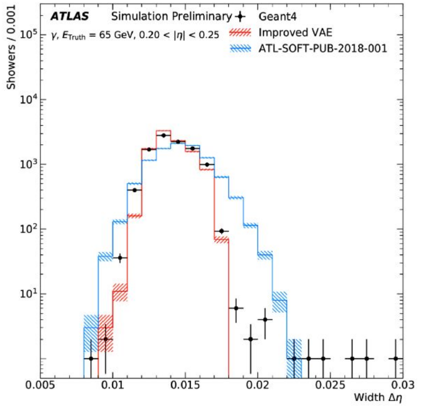
Event Generators - Technical Improvements and Porting

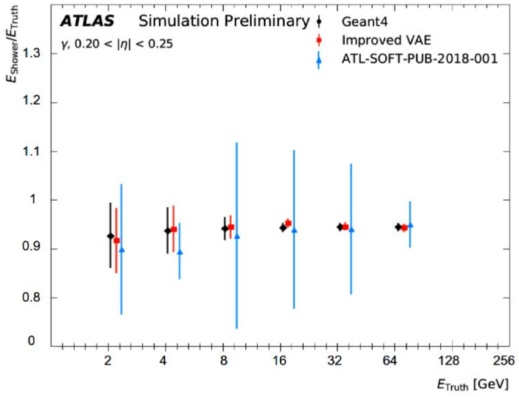
- Working group tackling technical challenges
  - Setting a baseline for further comparisons
  - Understanding how to run generators for best efficiency
  - Support for technical improvements (e.g. thread safety)
  - Porting to other architectures
    - Could be very suitable code to do this with (smaller, self contained code bases, numerically intensive)
- New Architectures
  - Original port of some elements of [MadGraph to GPUs]() by Japanese Group at KEK
  - Work reinvigorated and now pursued actively by CERN and UC Louvain
  - Interest also in Sherpa (heavily used by ATLAS)





Detector Simulation

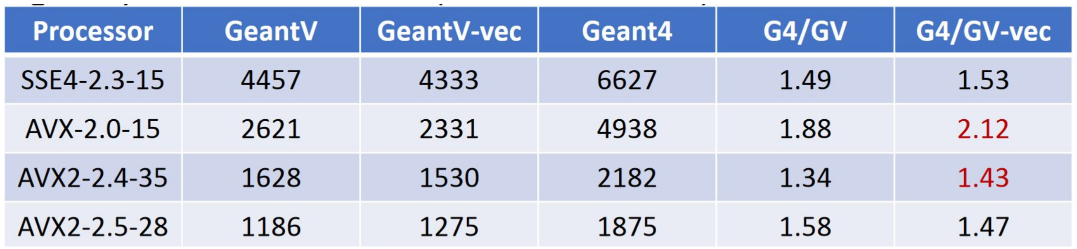




ATLAS VAE and GAN
cf. Geant4 simulation

- A major consumer of LHC grid resources today
  - Experiments with higher data rates will need to more simulation
- Faster simulation, with no or minimal loss of accuracy, is the goal
  - Range of techniques have been used for a long time (frozen showers, parametric response)
  - Key point is deciding when it's good enough for physics
- Machine learning lends itself to problems like this
  - Calorimeter simulations usually targeted
  - Variational Auto Encoders (VAEs) and Generative Adversarial Networks (GANs)
    - This is probably *not as easy as we thought* - traditional parametric approaches are hard to beat
  - R&D on lifecycle integration into Geant4 is starting

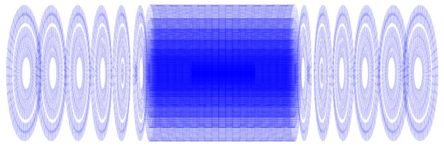
Detector Simulation



Processor	GeantV	GeantV-vec	Geant4	G4/GV	G4/GV-vec
SSE4-2.3-15	4457	4333	6627	1.49	1.53
AVX-2.0-15	2621	2331	4938	1.88	2.12
AVX2-2.4-35	1628	1530	2182	1.34	1.43
AVX2-2.5-28	1186	1275	1875	1.58	1.47

GeantV and Geant4 execution speeds for EM physics with and without vectorisation (Andrei Gheata)

- Technical improvement programme helps (and helps *everyone*)
- GeantV R&D [arXiv:2005.00949] modernises code and introduces vectorisation
  - Speed-ups observed (VecCore and VecGeom *backported* to Geant4)
  - Vectorisation introduces small gains, due to costs of "basketisation"
  - Code modernisation seems to help a lot
- HSF Simulation Working Group meetings on Accelerator R&D
  - Reports from successful projects that have used accelerators
    - MPEXS (medical); Opticks (JUNO Optical Photons); ExaSMR (reactor neutron transport); GATE (medical)
  - Round table of HEP R&D
    - Geometry on GPUs (VecGeom); EM physics (ExCALIBER) (use of native navigation and shaders); ML Accelerated Calculations (ATLAS EMEC); LHCb RICH Optical Photons using OptiX

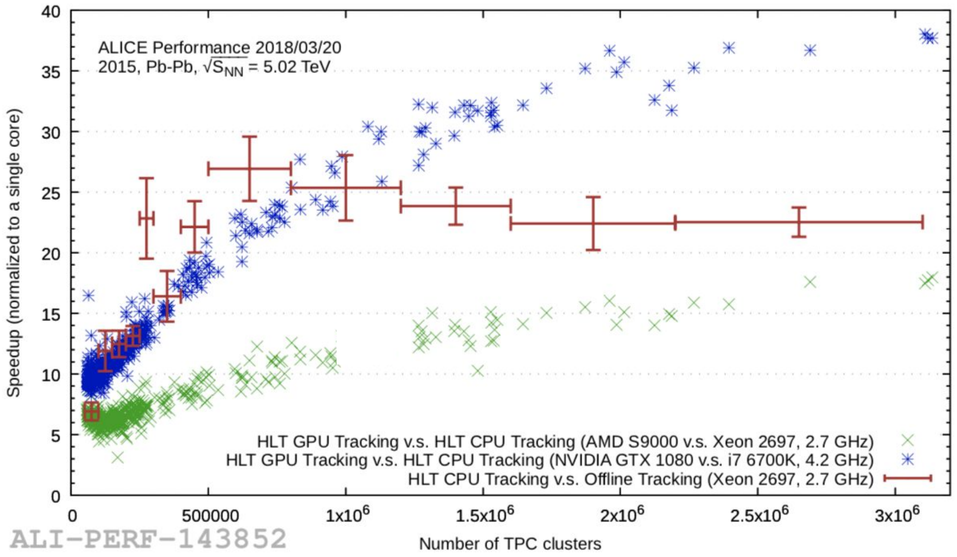


TrackML geometry rendered on GPU

Reconstruction and Software Triggers

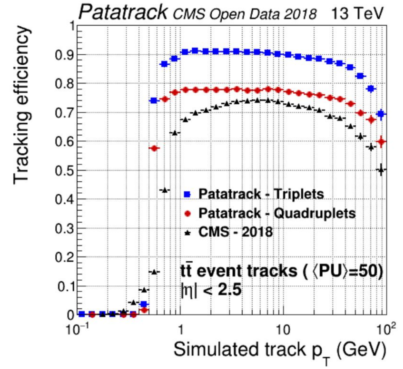
- Hardware triggers no longer sufficient for modern experiments in many cases
  - More and more initial reconstruction needs to happen in software
  - Especially in the high-signal regime
  - Here the pressure to break with legacy implementations is very high
- ALICE have used GPUs since Run 2
  - Lessons learned:
    - Keep data model simple
    - Asynchonous
    - Minimise data transfers
- Expansion to Run 3 rates requires additional Improvements
  - Better hardware helps modern GPUs)
  - Better algorithms are essential

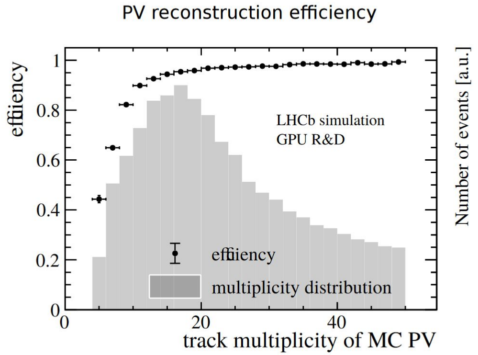
arXiv:1905.05515

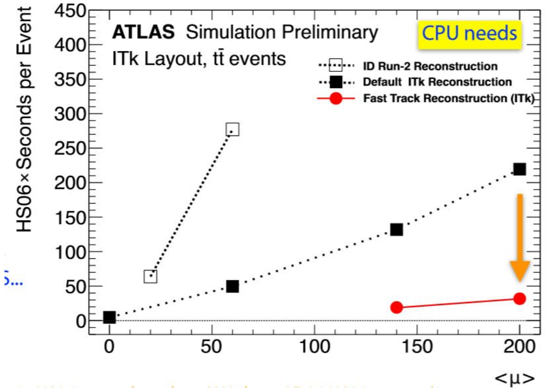


Additional LHC GPU Projects

- CMS Patatrack project developed track seeding algorithms to run on GPUs
  - Target 30% of trigger compute budget
  - Gain experience in the field and learn for Run 4
  - Physics was also improved!
- LHCb Allen project has demonstrated the entire HLT1 chain can run on GPUs
  - Great throughput and scalability
  - *Now chosen as baseline solution for Run-3*
- However, also design your detector taking software and computing into account
  - ATLAS ITk reconstruction is faster at 200 pile-up than current reconstruction at 60 (aka Don't Panic!) [ATL-PHYS-PUB-2019-041]





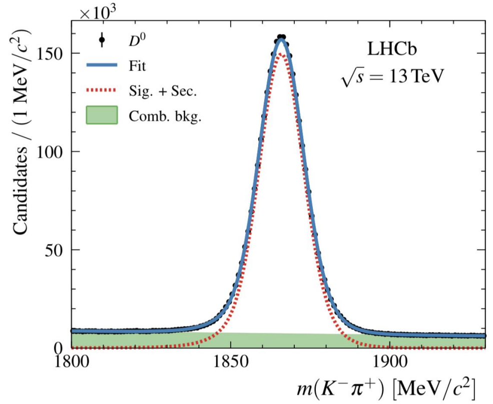


Real Time Analysis

- Design a system that can produce analysis useful outputs as part of the trigger decision
  - If this captures the most useful information from the event, can dispense with raw information
- This is a way to fit more physics into the budget
- LHCb Turbo Stream has been introduced in Run2 and will be dominant in Run3
- Whole ALICE data reduction scheme is based around keeping ‘useful’ parts of events (no more binary trigger)
  - O2 → Online/Offline Data Reduction Farm
- ATLAS and CMS have similar schemes for special handling of samples for which full raw data is unaffordable (aka. data scouting)

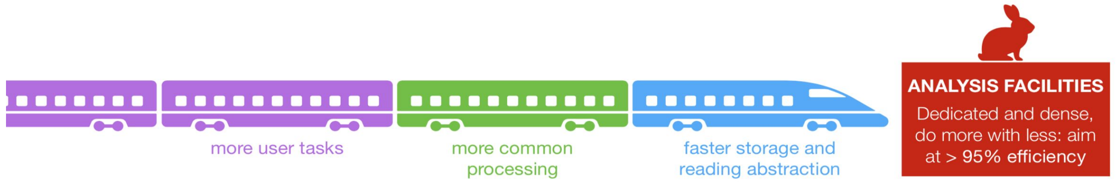
Persistence method	Average event size (kB)
Turbo	7
Selective persistence	16
Complete persistence	48
Raw event	69

LHCb Run2 Turbo took 25% of events for only 10% of bandwidth

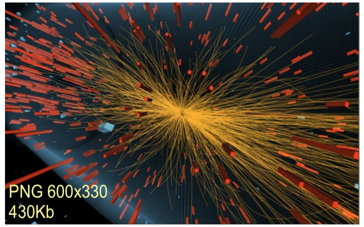


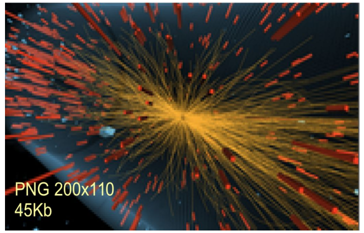
LHCb charm physics analysis using Turbo Stream (arXiv:1510.01707)

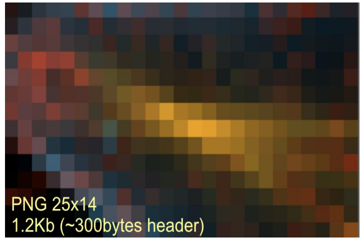
Analysis


more user tasks
more common processing
faster storage and reading abstraction
ANALYSIS FACILITIES
Dedicated and dense, do more with less: aim at > 95% efficiency

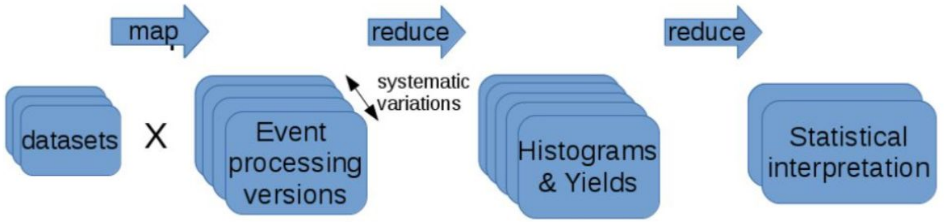
- Scaling for analysis level data also a huge challenge for all LHC experiments
- Efficient use of analysis data can come with combining many analyses as carriages in a train like model (pioneered by PHENIX and then ALICE)
    - Also goes well with techniques like tape carousels (ATLAS scheme for rotating primary AOD data from tape systems into a disk buffer)
    - Interest in *analysis clusters*, specialised for analysis operations over the generic grid resources (WLCG/HSF pre-CHEP workshop)
- Reducing volume of data needed helps hugely
    - CMS ~1kB nanoAOD makes a vast difference to analysis efficiency and “papers per petabyte”
    - Smaller EDM is easier to make efficient
    - Requires analyst agreement on corrections, scale factors, etc.
        - However the alternative is perhaps that your analysis never gets done





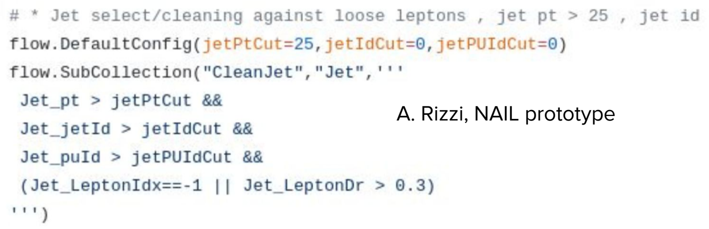


Analysis



- Improve analysis ergonomics - how the user interacts with the system to express their analysis
  - Streamline common tasks
    - Handle all input datasets; Corrections and systematics
    - Compute per event and accumulate; Statistical interpretations
  - Declarative models, building on ROOT's RDataFrame
    - Say *what*, not *how* and let the backend optimise
    - E.g. split and merge, GPU execution
- Notebook like interfaces gain ground, as do containers - lots of high level Python
- Interest in data science tools and machine learning is significant for this community

Many analysis frameworks, multiple per experiment, not well generalised

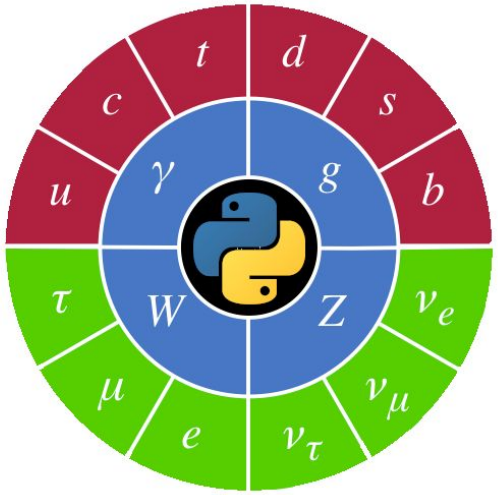


```
# * Jet select/cleaning against loose leptons , jet pt > 25 , jet id
flow.DefaultConfig(jetPtCut=25,jetIdCut=0,jetPUIdCut=0)
flow.SubCollection("CleanJet","Jet",'''
 Jet_pt > jetPtCut &&
 Jet_jetId > jetIdCut &&
 Jet_puId > jetPUIdCut &&
 (Jet_LeptonIdx==-1 || Jet_LeptonDr > 0.3)
''')
```

A. Rizzi, NAIL prototype

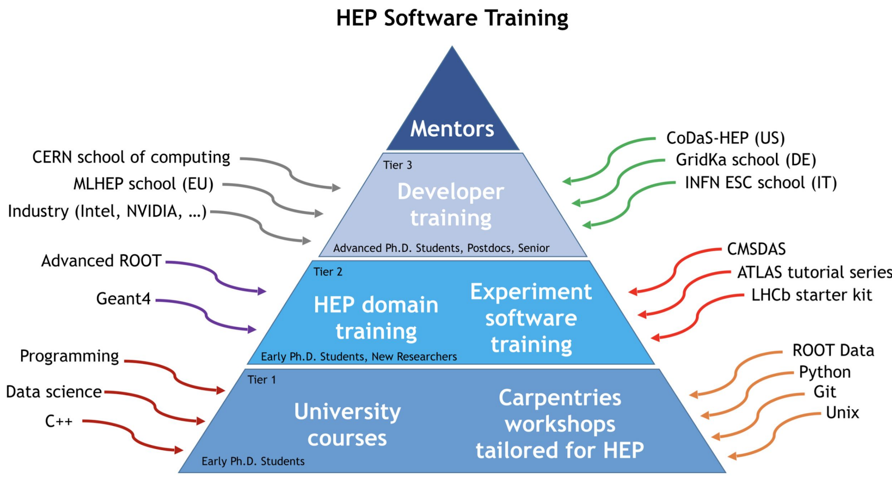
PyHEP ("Python in HEP") and New Approaches

- Python is ever more popular in Particle Physics
- Impressive developments of a Python scientific ecosystem for HEP in the last few years
- With strong links to the general scientific ecosystem
  - Interest in *data science* tools and *machine learning* is significant for this growing community
- Inspiring new approaches for data analysis
  - Exploiting modern approaches - declarative programming, heterogeneous resources, etc.
  - This is an ecosystem into which HEP can, and does, contribute
    - Fitting, histograms
  - Knowledge transfer goes both ways
  - Various projects under development, inter-communicating
- Yearly PyHEP workshops have been a success
  - This year's virtual PyHEP had 1000 people register!



Training and Careers

- Many new skills are needed for today's software developers and users
- Base has relatively uniform demands
  - Any common components help us
- LHCb StarterKit initiative taken up by several experiments, sharing training material
  - We ran a Software Carpentries tutorial at CERN last year
- New areas of challenge
  - Concurrency, accelerators, data science (upcoming: oneAPI training from openlab, Alpaka training from openlab/HSF, possible CUDA bootcamp via openlab)
  - Need to foster new C++ expertise (unlikely to be replaced soon as our core language, but needs to be modernised)
- Working hard to provide training templates for people developing material
  - Making training sustainable is important to maximise the time people invest here
- Careers area for HEP software experts is an area of great concern
  - Need a functioning career path that retains skills and rewards passing them on…



Directions for International Efforts

- Particle physics is in inherently international effort, with an excellent tradition of cooperation in many different domains
  - Detector R&D, Experiments, WLCG, Common Software
- But we have also had incoherent approaches and duplication
- HEP Software Foundation tries to foster much more the shared vision
  - This encourages diverse R&D!

There is clearly success in attracting funding to this area: IRIS-UK, ExCALIBER, SWIFT-HEP, IRIS-HEP, ErUM-DATA IDT, CERN EP R&D, AIDAinnova; links to other sciences and software engineers via SIDIS. Though overall we believe it is **not yet enough**!

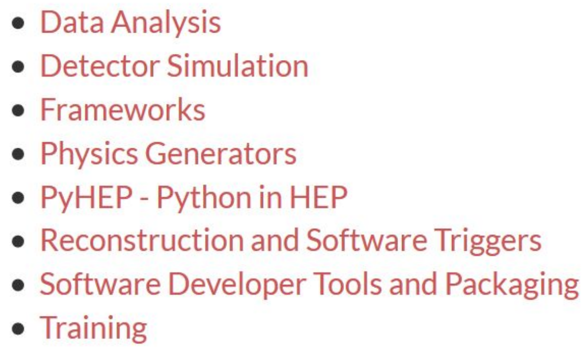
Conclusions

- We have a wide ranging and ambitious physics programme in HEP and in associated disciplines
  - Our experiments are highly data intensive and require high quality software and computing
- The landscape for software is becoming ever more challenging
  - Working together on common problems is not only the best use of our resources, our funding agencies will mandate it
- HSF is now established to help HEP achieve that goal and marshalls effort around the community
  - We had a very successful Virtual Workshop in May
  - Another planned now for November (19-20 + 23-24)

HL-LHC is a challenge and also a great opportunity to improve HEP software

HSF Getting Involved…

- Join the HSF Forum, hsf-forum@gmail.com
  - Few messages a week with updates, jobs, items of interest
  - Owned by the community - please just post items of relevance
- Join a working group, https://hepsoftwarefoundation.org/what_are_WGs.html
  - Follow the group's meetings and discussions
  - Suggest a meeting topic
- Annual meetings and Workshops
  - Established a tradition of a joint meeting with WLCG each Year (next short meeting pre-CHEP, November)
  - Now adapting for more virtual interactions
- Propose a new activity area
  - The HSF is there to help gather interest



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