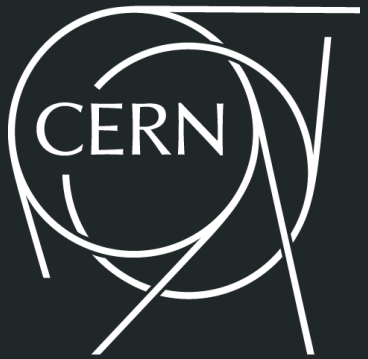
Advancing ~~Advances~~ in HEP Software and Computing

Graeme Stewart, CERN EP-SFT

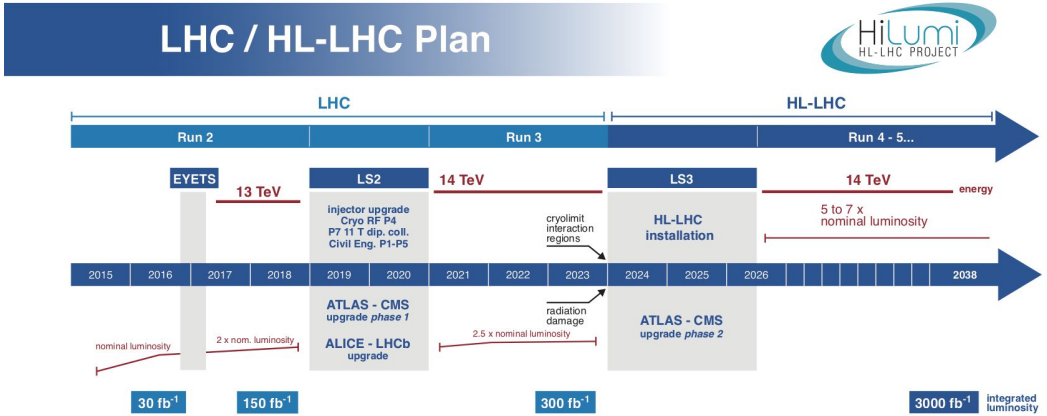


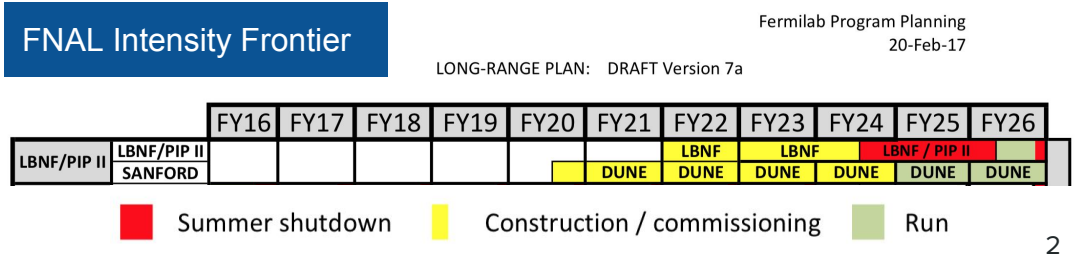
Spåtind 2018 - Nordic Conference on Particle Physics, 2018-01-05

Experimental Particle HL-LHC and Intensity Frontier

Our mission:

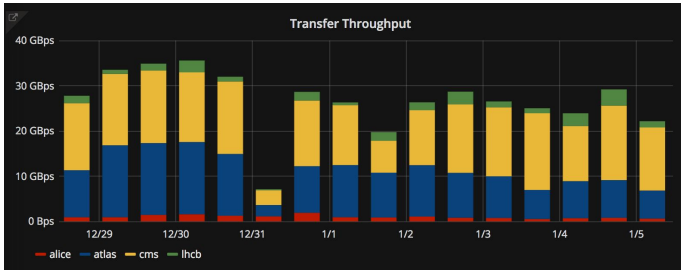
- Exploit the Higgs for SM and BSM physics
- b, c, tau physics to study BSM and matter/anti-matter
- Dark matter
- Neutrino oscillations and mass
- QGP in heavy ion collisions
- Explore the unknown

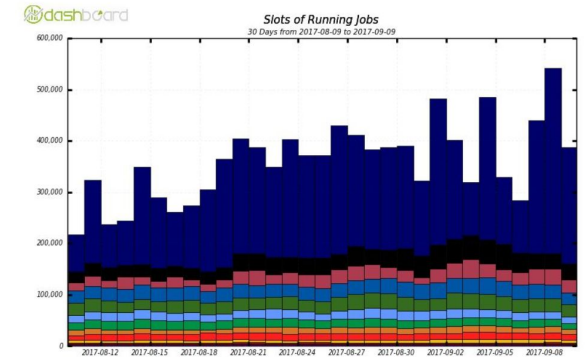


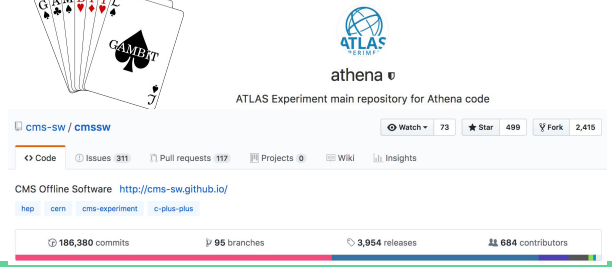


HEP Software and Computing

- High Energy Physics has a vast investment in software
  - Estimated to be around 50M lines of C++
  - Which would cost more than 500M$ to develop commercially
- It is a critical part of our physics production pipeline, from triggering all the way to analysis and final plots as well as simulation
- LHC experiments use about 600k CPU cores every hour of every day and have around 400PB of data stored on disk and 600PB on tape
  - We are in the exabyte era already
- This is a huge and *ongoing* cost in hardware and human effort
- With significant challenges ahead of us to support our ongoing physics programme

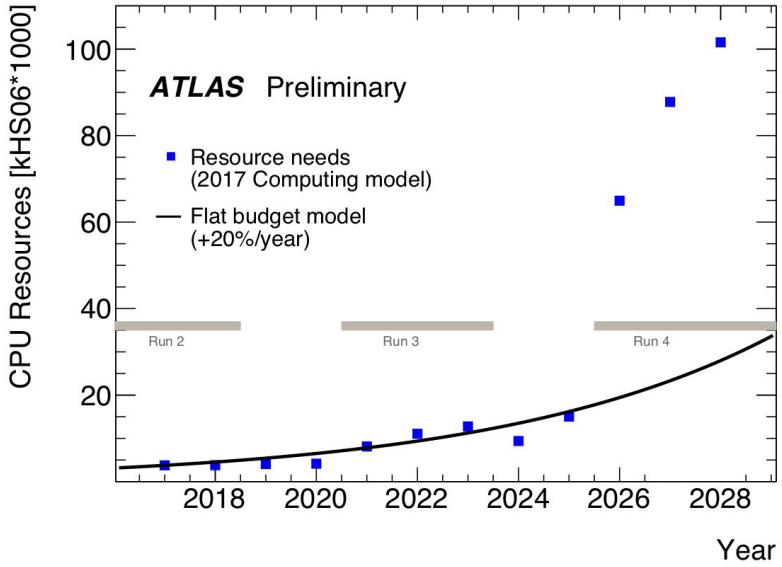


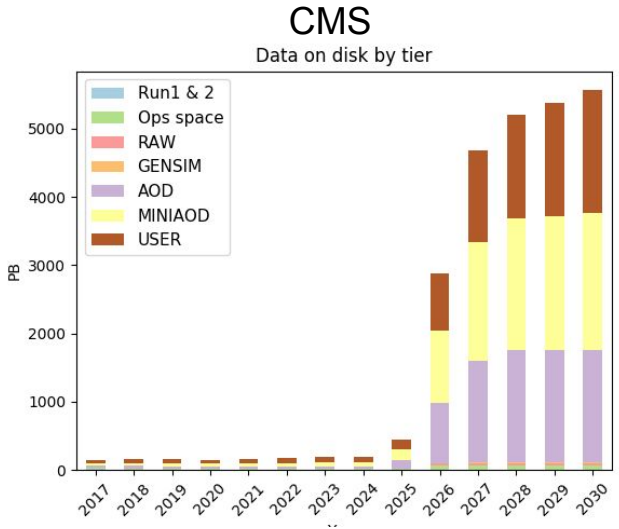




Challenges for the Next Decade

- HL-LHC brings a huge challenge to software and computing
  - Both rate and complexity rise

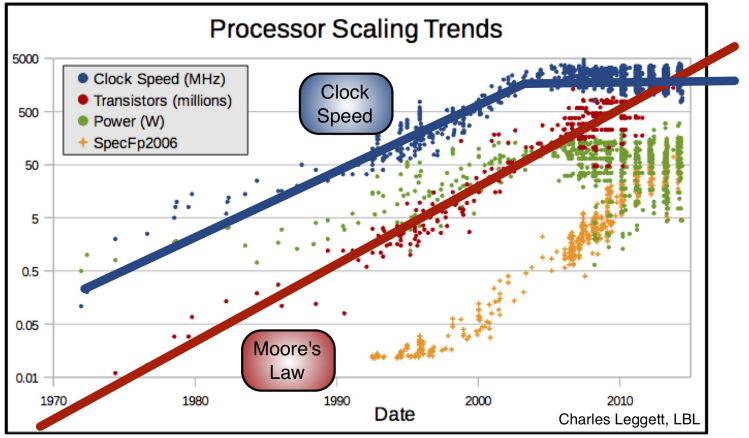




- Not just a simple extrapolation of Run 2 software and computing
  - Resources needed would hugely exceed those from technology evolution alone

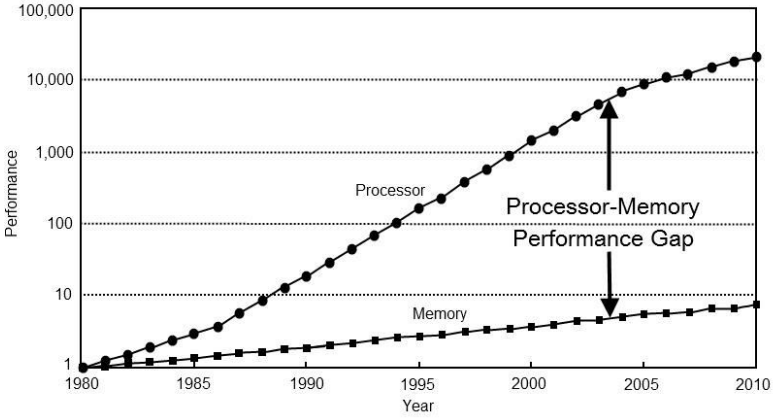
Technology Evolution

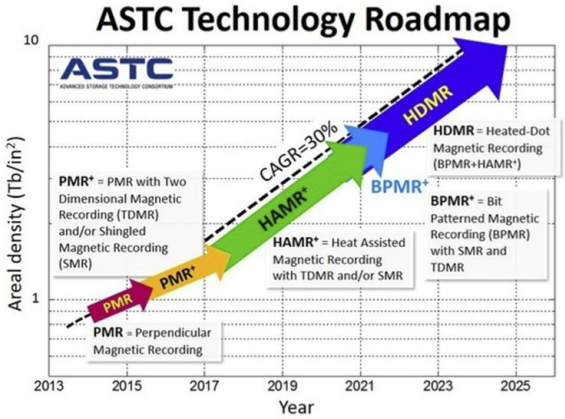
- Moore's Law continues to deliver increases in transistor density
  - Doubling time is lengthening
  - IBM recently demonstrated 5nm wafer fabrication
- Clock speed scaling failed around 2006
  - No longer possible to ramp the clock speed as process size shrinks
  - Leak currents become important source 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



Technology Evolution

- Many/multi core systems are the norm
  - Serial or multi-process processing is under severe memory pressure
- Co-processors now commonplace
  - GPUs, FPGAs - greater throughput, far more challenging programming model
- Wide vector registers
  - Up to 512 bit
- Feeding data from main memory is a serious problem
  - Deeper cache hierarchy
- Storage capacity climbing
  - 100TB disks possible by HL-LHC, but little I/O improvement expected
- Network capacity keeps growing

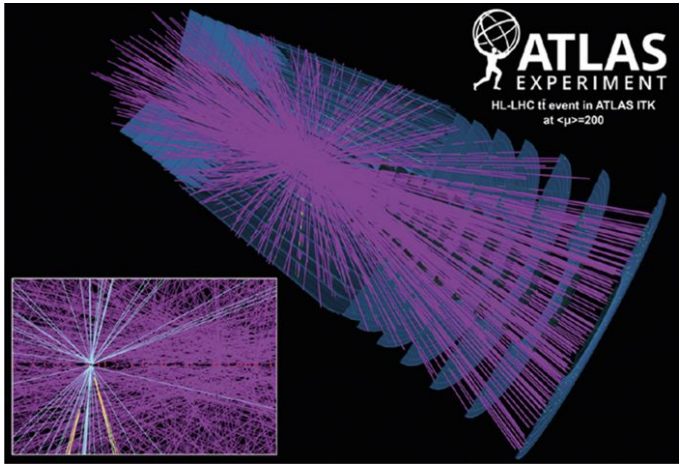




Drivers of Technology Evolution

- 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
- 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

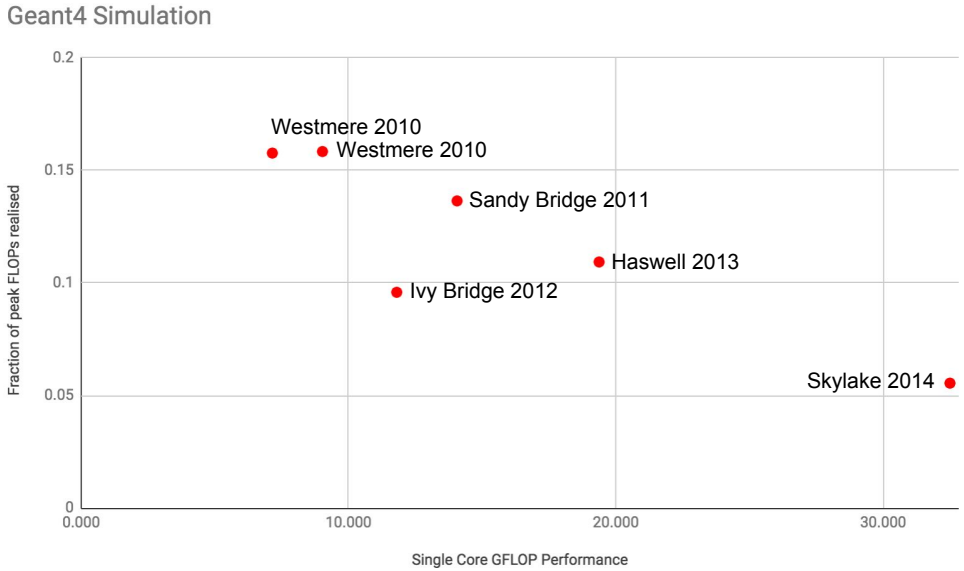
Software Challenges for HL-LHC



- Pile-up of ~200 ⇒ particularly a challenge for charged particle reconstruction
- With a flat budget, Moore's lawish improvements are the real maximum we can expect on the HW side
- HEP software typically executes one instruction at a time (per thread)
  - Since ~2013 CPU (core) performance increase is due to more internal parallelism
  - x10 with the same HW only achievable if using the full potential of processors
    - major SW re-engineering required (but rewriting everything is not an option)
  - Co-processors like GPUs are of little use until the problem has been 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

How is our code doing? Simulation on 5 years of Intel CPUs

- Fraction of the potential floating point performance we use has been dropping over time
- CPU manufacturers add wider vectors that we do not take advantage of, or deep pipelines where cache misses are very costly
- Confirms what we have long suspected about the growing performance gap on modern architectures



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, ...)
- HSF already 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 Inception

- From Spring 2016 discussions, idea started to crystallise at the May 2016 HSF Meeting at LAL
  - *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
    i. to achieve improvements in software efficiency, scalability and performance and to make use of the advances in CPU, storage and network technologies
    ii. to enable new approaches to computing and software that could radically extend the physics reach of the detectors
    iii. to ensure the long term sustainability of the software through the lifetime of the HL-LHC

CWP process

- Kick-off workshop 23-26 January 2017, San Diego
- Groups held workshops and meetings in the subsequent months
  - Broadening the range of participation, often with non-HEP experts participated
- Workshop in Annecy 26-30 June started to draw the process to a close
  - 13 Working Groups had made good progress on their chapters
- Both workshops involved ~100 people, mainly US and EU
  - Total number of people involved in the writing process was about 250
  - Many others commenting

CWP - Making a roadmap for the future

- Editorial Board was set up, with the aim of encompassing the breadth of our community
  - Wide regional and experimental representation
- First draft released 20 October
- Second draft released 17 November
- These drafts elicited a *substantial response* from the community, leading to many improvements
- Final version of the document published arXiv: 1712.06982 on 20 December
- In addition there are many individual working group chapters giving significant detail on their area of expertise

- Predrag Buncic (CERN) - ALICE contact
- Simone Campana (CERN) - ATLAS contact
- Peter Elmer (Princeton)
- John Harvey (CERN)
- Benedikt Hegner (CERN)
- Frank Gaede (DESY) - Linear Collider contact
- Maria Girone (CERN Openlab)
- Roger Jones (Lancaster University) - UK contact
- Michel Jouvin (LAL Orsay)
- Rob Kutschke (FNAL) - FNAL experiments contact
- David Lange (Princeton)
- Dario Menasce (INFN-Milano) - INFN contact
- Mark Neubauer (U.Illinois Urbana-Champaign)
- Eduardo Rodrigues (University of Cincinnati)
- Stefan Roiser (CERN) - LHCb contact
- Liz Sexton-Kennedy (FNAL) - CMS contact
- Mike Sokoloff (University of Cincinnati)
- Graeme Stewart (CERN, HSF)
- Jean-Roch Vlimant (Caltech)

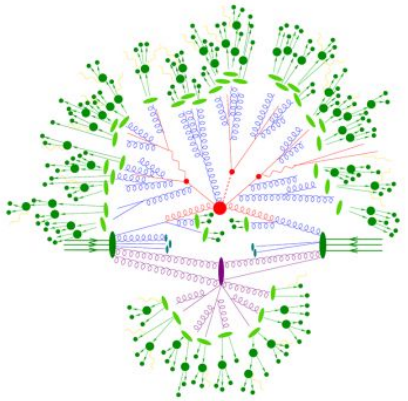
A Roadmap for HEP Software and Computing R&D for the 2020s

HSF-CWP-2017-01
December 15, 2017

- 70 page document
- 13 sections summarising R&D in a variety of technical areas for HEP Software and Computing
  - Almost all major domains of HEP Software and Computing are covered
- 1 section on Training and Careers
- 274 authors from 117 institutions
- Signing policy: sign the document if you agree with the main observations and conclusions
  - hsf-cwp-ghost-writers@googlegroups.com
- *We really actively encourage you to do this as your name indicates the breadth of support in the community*

Contents

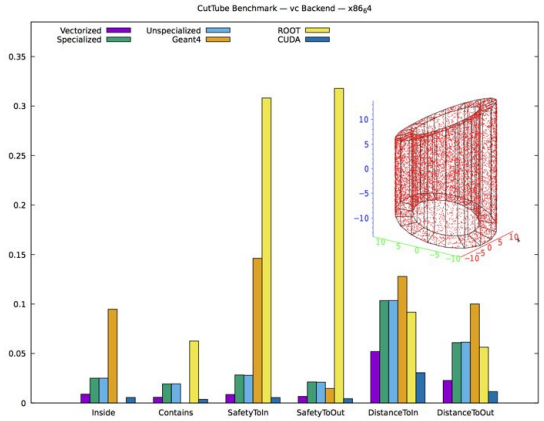
Physics Event Generators



- Physics event generation starts our simulation chain to enable comparisons with detector events
  - At Leading Order the CPU consumption of event generation is modest
  - At Next-to-leading Order CPU consumption can become important
  - To get a proper handle on rare processes at the HL-LHC Next-to-next-to-leading order, NNLO, will be required for more analyses
- Generators are written by the theory community
  - Need expert help and long term associations to achieve code optimisation
  - Even basic multi-thread safety is problematic for many older, but still heavily used, generators
  - Ongoing maintenance of tools like HepMC, LHAPDF, Rivet is required and needs rewarded
  - Projects such as scalable VEGAS-style integrator and reweighting tools are foreseen

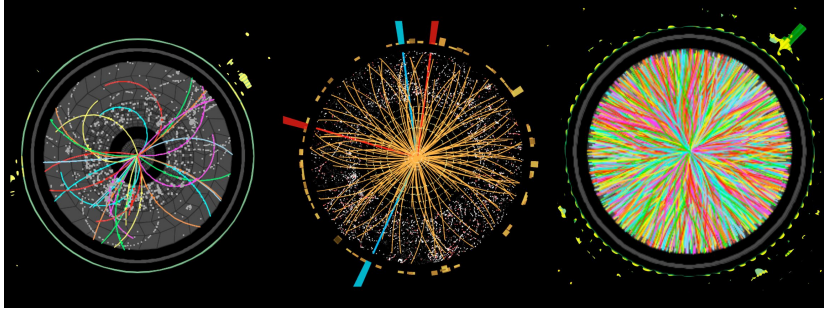
Detector Simulation

- **Simulating our detectors consumes huge resources today**
  - Remains a vital area for HL-LHC and intensity frontier experiments in particular
- **Main R&D topics**
  - Improved physics models for higher precision at higher energies (HL-LHC and then FCC)
    - Hadronic physics in LAr TPCs needs to be redeveloped
  - Adapting to new computing architectures
    - Can a vectorised transport engine be demonstrated to work in a realistic prototype? How painful would evolution be?
  - Fast simulation - develop a common toolkit for tuning and validation
    - Can we use Machine Learning profitably here?
  - Geometry modeling
    - Easier modelling of complex detectors, targeting new computing architectures



Vectorised geometry speed-ups, Mihaela Gheata

Software Trigger and Event Reconstruction



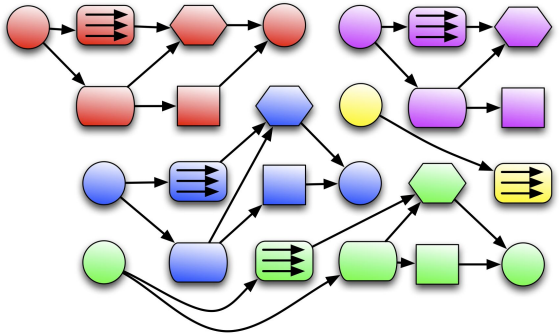
- **Move to software triggers is already a key part of the program for LHCb and ALICE already in Run 3**
  - ‘Real time analysis’ increases signal rates and can make computing much more efficient (storage and CPU)
- **Main R&D topics**
  - Controlling charged particle tracking resource consumption and maintaining performance
    - Do current algorithms’ physics output hold up at pile-up of 200 (or 1000)
    - Can tracking maintain low p_T sensitivity within budget?
  - Detector design itself has a big impact (e.g., timing detectors, track triggers)
  - Improved use of new computing architectures
    - Multi-threaded and vectorised CPU code
    - Extending use of GPGPUs and possibly FPGAs
  - Robust validation techniques when information will be discarded
    - Using modern continuous integration, tackling multiple architectures with reasonable turnaround times

See Andy’s talk next

Data Analysis and Interpretation

- **Today we are dominated by many cycles of data reduction**
  - Aim is to reduce the input to an analysis down to a manageable quantity that can be cycled over quickly on ~laptop scale resources
  - Key metric is ‘time to insight’
- **Main R&D topics**
  - How to use the latest techniques in data analysis that come from outside HEP?
    - Particularly from the Machine Learning and Data Science domains
    - Need ways to seamlessly interoperate between their data formats and ROOT
      - Python is emerging as the *lingua franca* here, thus guaranteeing our python/C++ bindings is critical
  - New Analysis Facilities
    - Skimming/slimming cycles consume large resources and can be inefficient
    - Can interactive data analysis clusters be set up?
  - Data and analysis preservation is important

Data Processing Frameworks



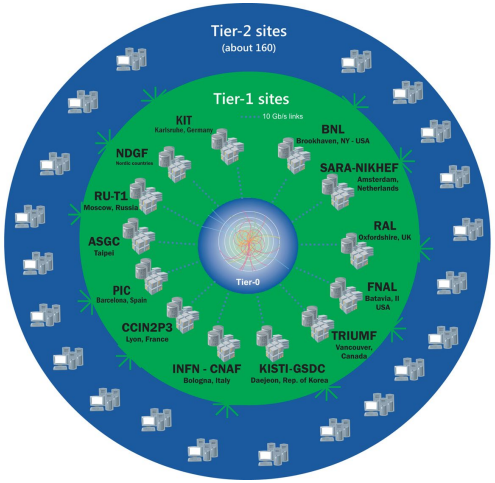
Event processing framework schematic (colours ar events, boxes algorithms)

- **Experiment software frameworks provide the scaffolding for algorithmic code**
  - Currently there are many implementations of frameworks, with some sharing between experiments
  - All of these frameworks are evolving to support concurrency
    - Protect physicists from details and dangers of parallelisation
- **Main R&D topics**
  - Adaption to new hardware, optimising efficiency and throughput
    - We need the best libraries for this and these will change over time
  - Incorporation of external (co)processing resources, such as GPGPUs
  - Interface with workload management system to deal with the inhomogeneity of processing resources
    - From volunteer computing to HPC job slots with 1000s of nodes
  - **Which components can actually be shared and how is that evolution achieved?**

Data Management and Organisation

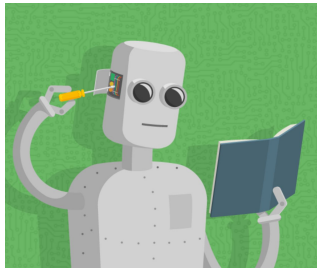
- **Data storage costs are a major driver for LHC physics today**
  - HL-LHC will bring a step change in the quantity of data being acquired by ATLAS and CMS
- **Main R&D topics**
  - Adapt to new needs driven by changing algorithms and data processing needs, e.g,
    - The need for fast access to training datasets for Machine Learning
    - Supporting high granularity access to event data
      - Needed to effectively exploit backfill or opportunistic resources
    - Rapid high throughput access for a future analysis facility
    - Processing sites with small amounts of cache storage
  - Do this profiting from the advances in industry standards and implementations, such as Apache Spark-like clusters (area of continued rapid evolution)
    - Of course what we do is not exactly like what they do… structured access to complex data
  - Consolidate storage access interfaces and protocols
  - Support efficient hierarchical access to data, from high latency tape and medium latency network

Facilities and distributed computing



- **Storage and computing today are provided overwhelmingly from WLCG resources**
  - Expected to continue for HL-LHC, but to be strongly influenced by developments in commodity infrastructure as a service (IaaS, commercially this is usually Cloud Computing)
- **Main R&D topics**
  - Understand far better the effective costs involved in delivering computing for HEP
    - This needs to be sensitive to regional variations in funding and direct and indirect costs
      - e.g., smaller sites frequently contribute ‘beyond the pledge’, power and human resources
    - Full model is unfeasible, but providing a reasonable gradient analysis for future investment should be possible
      - Should we invest in better network connectivity or in more storage?
      - Does a data lake make sense for us? Concentrated storage at fewer sites.
  - How to take better advantage of new network and storage technologies (software defined networks, object stores or content addressable networks)
  - Strengthen links to other big data sciences (SKA) and computing science; how to share network resources

Machine learning



- Neural networks and Boosted Decision Trees have been used in HEP for a long time
  - e.g., particle identification algorithms
- More recently the field has been significantly enhanced by new techniques (Deep Neural Networks) and enhanced training methods
  - Very good at dealing with noisy data and huge parameter spaces
  - A lot of interest from our community in these new techniques, in multiple fields
- Main R&D topics
  - Speeding up computationally intensive pieces of our workflows (fast simulation, tracking)
  - Enhancing physics reach by classifying better than our current techniques
  - Improving data compression by learning and retaining only salient features
  - Anomaly detection for detector and computing operations
- Significant efforts will be required to make effective use of these techniques
  - Good links with the broader Machine Learning and Data Science communities required

See Kyle's talk Thursday

Data, software and analysis preservation

- We seem to be doing well compared to other fields
- Challenge is both to physically preserve bits and to preserve knowledge
  - DPHEP has looked into both
- Knowledge preservation is very challenging
  - Experiment production workflows vary in significant details
  - Variety of different steps are undertaken at the analysis stage, even within experiments
- Need a workflow that can capture this complexity
  - Technology developments that can help are, e.g., containers
- CERN Analysis Preservation Portal forms a good basis for further work
  - Needs to have a low barrier for entry for analysts
  - Can provide an immediate benefit in knowledge transmission within an experiment

Other technical areas of work

Conditions Data

- Growth of alignment and calibration data is usually linear in time
  - Per se, this does not represent a major problem for the HL-LHC
- Opportunities to use modern distributed techniques to solve this problem efficiently and scalably
  - Cacheable blobs accessed via REST
  - CVMFS + Files
  - Git

Visualisation

- Many software products developed for event visualisation
  - Part of the framework, with full access to event and geometry data
  - Standalone as a lightweight solution
- New technologies for rendering displays exist, e.g., WebGL from within a browser

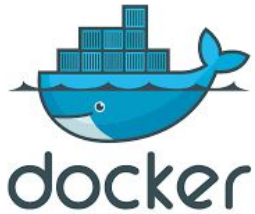
Security

- We have a large infrastructure that is an important resource for us
- Evolution of authorisation and authentication away from X.509 towards simpler mechanisms
  - OAuth used widely now
- Enhancing effective security incident response teams

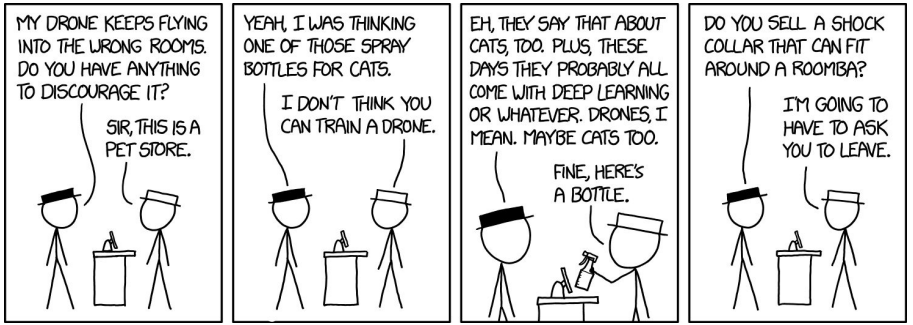
- These areas are examples of where we can refocus current effort towards common software solutions
- This should improve quality, economise overall effort and help us to adapt to new circumstances

Software Development - Process and Tools

- **Experiments have modernised their software development models a lot recently**
  - Source code has migrated, by and large, to git
    - Far more developer independence and flexibility than before
  - Social coding sites like github and gitlab amplify these advantages considerably
    - Pull/Merge request workflows help a great deal with code quality
    - Continuous integration and code review become natural and much easier than before
    - Enables widespread collaboration across many boundaries of geography and
  - CMake is becoming a pretty standardised way to build
  - This is a suite of common tools that facilitate collaboration and knowledge sharing
- **Additional tools would benefit the community:** Static analysis of code, refactoring code, performance measures
- As well as a more regular generic development forum



Training and Careers



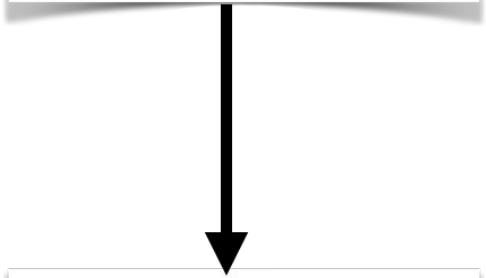
- **Using new tools requires investing in training for the community**
  - The more commonality in the tools and techniques, the more training we can share
    - ALICE and LHCb recently did this in practice using the StarterKit material
  - This provides preservation and propagation of knowledge
  - A lot of the training we need to do is generic, but some is quite specific (to HEP or experiment)
    - We should also encourage appropriate training at the undergraduate and graduate school level
- **Our environment is becoming more complex; we require input from physicists *whose concerns are not primarily in software***
  - Sustainability of these contributions is extremely important
- **Recognition of the contribution of our specialists in their careers is extremely important**
  - There needs to b an appropriate career path for software experts as much as any other technical discipline on which our success depends
  - We should also improve our publication and citation record (weak in some areas) and explore new avenues, e.g., Zenodoo

Parallelisation - Promising Directions

- Describe *what* you want to do, not *how* to do it
  - Leave that for the backend
  - Can parallelise much more easily and reorder calculations for maximum efficiency
  - Lazy evaluation helps optimise the workflow graph and will cache results
  - Approach extends more easily to new backends and facilities
    - Inspired by, e.g., Apache Spark
- Use different parallelisation models
  - Message passing models are conceptually easier and have less thread safety problems
    - ALICE O2 approach or the new Gaudi Functional model
    - Better adaptation to offloading to different devices
  - *ML toolkits are generally good at this* (e.g., TensorFlow)

```
TTreeReader data(tree);
TTreeReaderValue<A> x(data, "x");
TTreeReaderValue<B> y(data, "y");
TTreeReaderValue<C> z(data, "z");

while (reader.Next()) {
   if (IsGoodEvent(x, y, z))
      DoStuff(x, y, z);
}
```



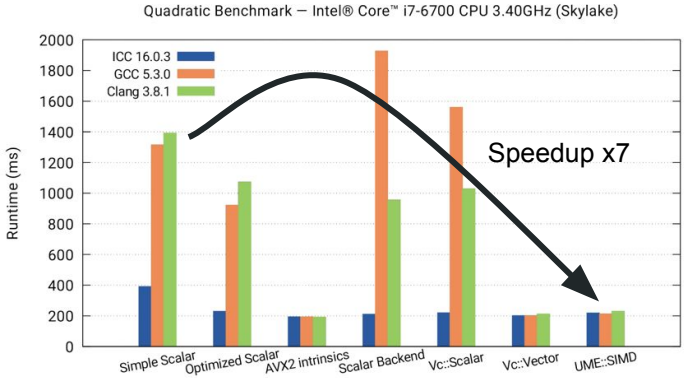
```
ROOT::EnableImplicitMT();
TDataFrame data(tree, {"x","y","z"});


data.Filter(IsGoodEvent)
    .Foreach(DoStuff);
```

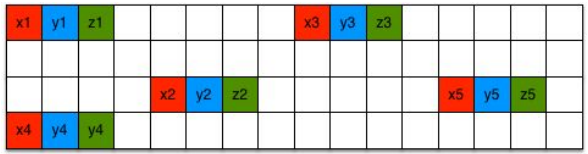
ROOT TDataFrame approach

Vectorisation and Cache Hierarchy - Promising Directions

- Exploiting vector registers is frustrating
  - Many different vectorisation models (SSE4.2, AVX, AVX2, AVX512, NEON, SVE, AltiVec)
  - Auto-vectorisation from the compiler is generally fragile and hard to use
  - Best to use an abstraction library: VecCore
  - Add fundamental vector types to C++
  - Optimise code at compile time for different backends
- Memory Layout
  - Efficient use of vectors requires also efficient load and store
  - Underlying storage of ATLAS xAOD is vector friendly
  - AIDA PODIO does this same for a simple EDM
  - Even C++ evolution, like `range_v3`, can support views of complex vector types



Guilherme Amadio et al.



Bad

Better

Best

Advancing from here

- Community White Paper process has been a success
  - Engaged more than 250 people and produced more than 300 pages of detailed description in many areas
- Summary roadmap lays out a path forward and identifies the main areas we need to invest in for the future for our software upgrade
  - Supporting the HL-LHC Computing TDRs and NSF S2I2 strategic plan
  - You can still sign :-)
- HEP Software Foundation has proved its worth in delivering this CWP Roadmap
  - Achieving a *useful* community consensus is not an easy process
  - Sign up to our forum to keep in touch and get involved (hep-sf-forum@googlegroups.com)
- We now need to marshal the R&D efforts in the community, refocusing our current effort and helping to attract new investment in critical areas
  - The challenges are formidable, working together will be the most efficacious way to succeed
  - HSF will play a vital role in spreading knowledge of new initiatives, encouraging collaboration and monitoring progress
  - Next HSF workshop in March, shared with WLCG, should start to put our ideas into practice:
    - C++ Concurrency, Workload Management and Frameworks, Facilities Evolution, Analysis Facilities, Training, ...