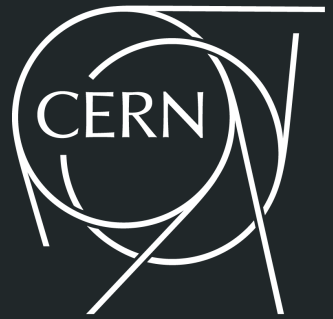


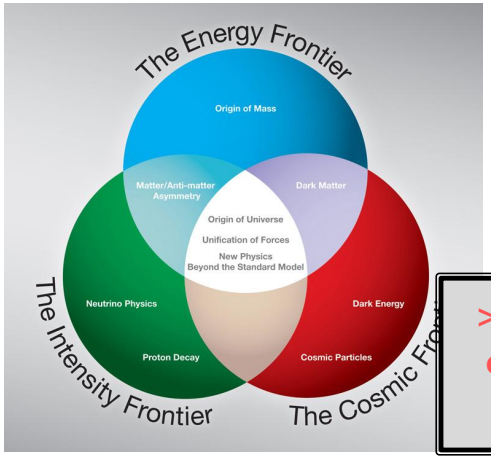
Future opportunities and challenges for software in HEP

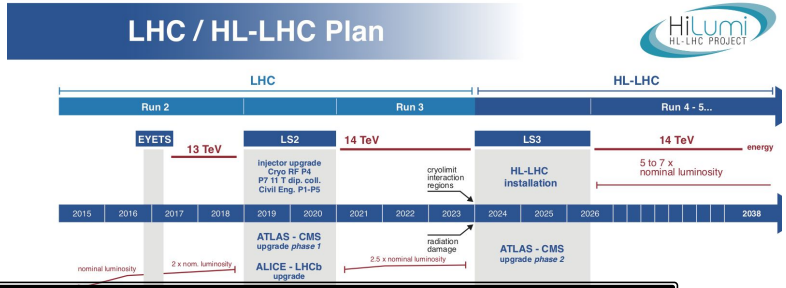
(with an analysis slant)

Graeme Stewart, CERN EP-SFT

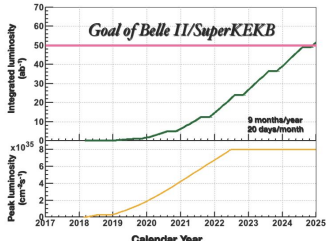
ECFA 102nd Plenary Meeting, ALBA Synchrotron, 20 July 2018

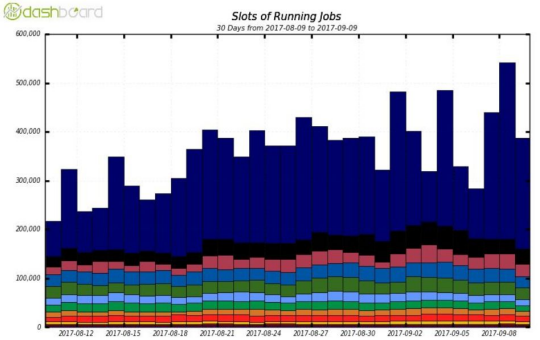
HEP: Landscape and Frontiers





SuperKEKB luminosity projection





>50M Lines of code that exist in multiple experiments and packages; persistence across generations of experiments

ROOT Data Analysis Framework

HSF HEP Software Foundation

Geant 4

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

FNAL Intensity Frontier

HEP.TrkX

ATLAS

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

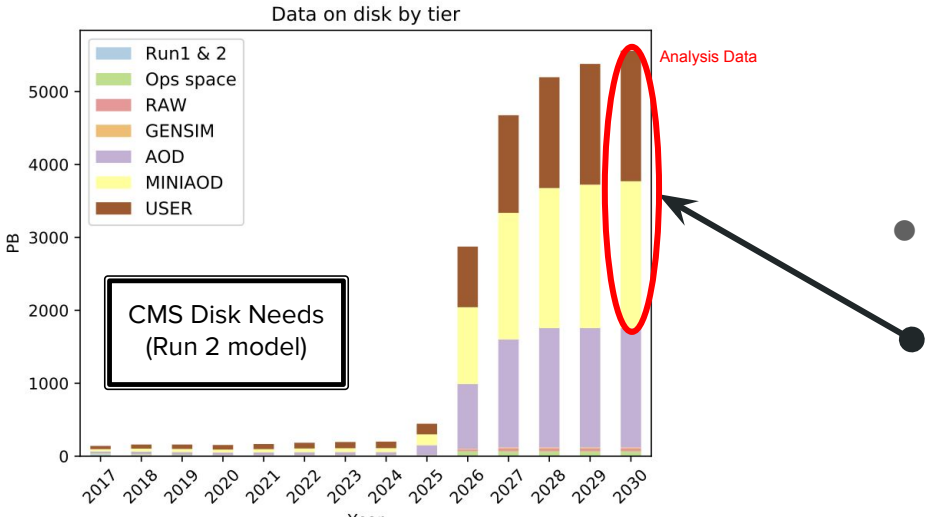
CMS Offline Software http://cms-sw.github.io/

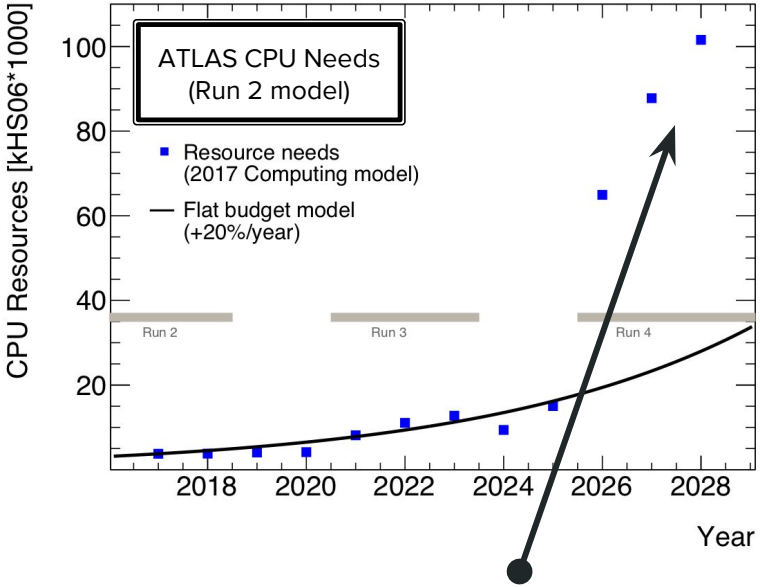
hep cern cms-experiment c-plus-plus

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

High Luminosity LHC

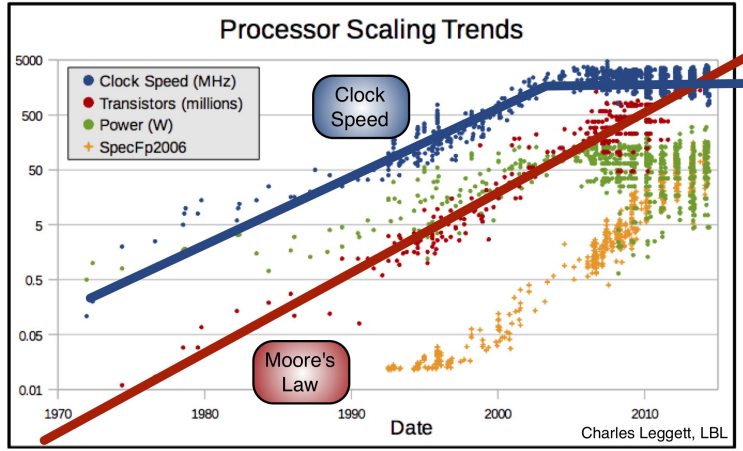
- Large rise in rate (~10kHz) and complexity (mu~200) : Run 2 SW & computing will not scale

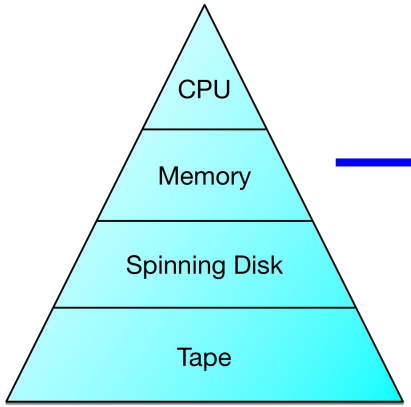




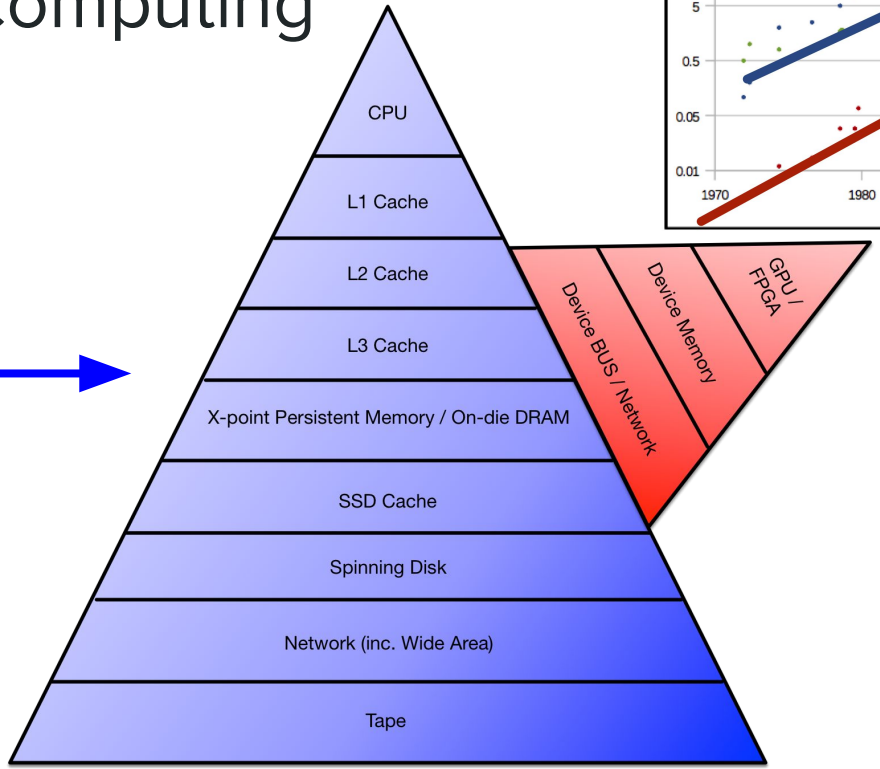
- Resources needed would hugely exceed those from technology evolution alone alone with a flat budget (close to Run 2+3 evolution)

Shifting landscape for end-to-end computing





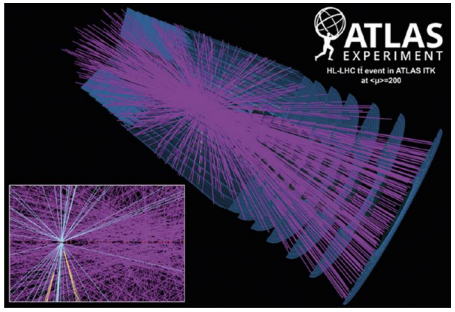
The Good Old Days



The Brave New World

Graeme Stewart, CERN

Software Challenges for HL-LHC



- Pile-up of ~200 ⇒ particularly a challenge for charged particle reconstruction (superlinear scaling, ~x30-50)
- With a flat budget, improvements from hardware of ~x6 (Moore's Law) are the real maximum we can expect
- Increased amount of data requires us to revise/evolve our computing and data management approaches
  - We must be able to feed our applications with data efficiently at scale (end-to-end computing)
  - For analysis sheer volume of event data is a major factor - I/O bound workload
- HEP software typically executes 1 instruction at a time (per thread)
  - Major re-engineering required to benefit from modern CPUs (can do 8 in theory, more like 2-4 for 'real' code)
  - Accelerators like GPUs are even more challenging
- HL-LHC salvation will come from software improvements, not from hardware

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

HSF-CWP-2017-01
December 15, 2017

- HSF established in 2015 to facilitate *coordination* and *common efforts* in software and computing across HEP in general
- Charged by WLCG to address R&D for the next decade
- 70 page document on arXiv (1712.06982)
- **13 topical sections** summarising R&D in a variety of technical areas for HEP Software and Computing
  - Backed by topical papers with more details also (e.g. 50-page detailed review about Detector Simulation)
- **1 section on Training and Careers**
- **310 authors** (signers) from 124 HEP-related institutions
- Feature article in CERN Courier
- More details on the HSF web site

Contents

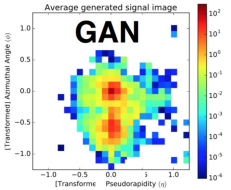
Guiding Strategy for the Future

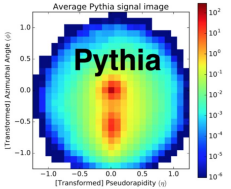
- HEP faced many challenges before other communities and has developed over the decades a lot of community-specific solutions
  - Mainly for good reasons!
  - Several HEP-tools adopted by some other communities, e.g. GEANT4 and ROOT, and WLCG itself is a model/driver for large-scale computing adopted by some other disciplines
- But the world changed: other scientific communities and industry facing some similar challenges and HEP must be able to benefit from them
- Does not mean that we have drop-in replacements for our solutions
  - Challenge: find the proper integration between our community tools and the available technologies outside, maintain the necessary backward compatibility/continuity and long-term sustainability
  - This means we need HEP domain experts who are also well versed in new techniques
- We face an end-to-end optimisation problem and we need to tackle issues from event generation right through to final histograms

Simulating Physics and Detectors

- Physics event generation starts our simulation chain
  - At Next-to-Leading Order (NLO) precision used today, CPU consumption can become significant
  - Study of rare processes at the HL-LHC will require the more demanding NNLO for more analyses
- Generators are written by the theory community
  - Need expert help to achieve code optimisation
  - Even basic multi-thread safety is problematic for many older, but still heavily used, generators

- Simulating our detectors consumes huge resources
- Improved physics models for higher precision at higher energies
- Adapting to new computing architectures
  - Vectorised transport engine tested in a realistic prototype - GeantV early releases
  - Evolution and re-integration into Geant4
- Faster simulation - develop a common toolkit for tuning and validation of fast simulation
  - How can we best use Machine Learning profitably here?
  - Multi-level approach, from processes to entire events

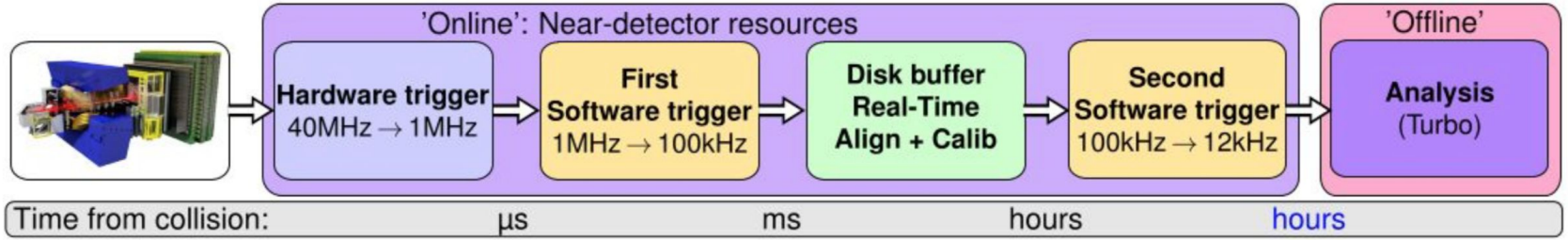




Machine learning simulated calorimiter

Software Triggers and Real Time Analysis

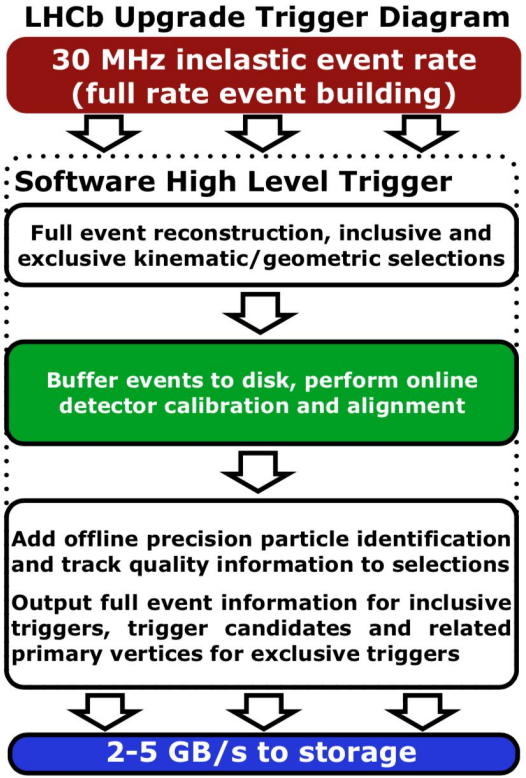
- Physics programs for LHCb and ALICE become very signal rich in Run 3
- Classic binary triggers cut too much into physics when many events are interesting
- Use a full software trigger to be able to extract analysis quality outputs from collisions
  - 30MHz pp collisions for LHCb
  - 50kHz HI collisions for ALICE
- Challenge is to keep data volumes under control
  - The only way is to drop the RAW data and keep only the reconstructed outputs
  - This is a paradigm shift to ‘lossy’ compression of events



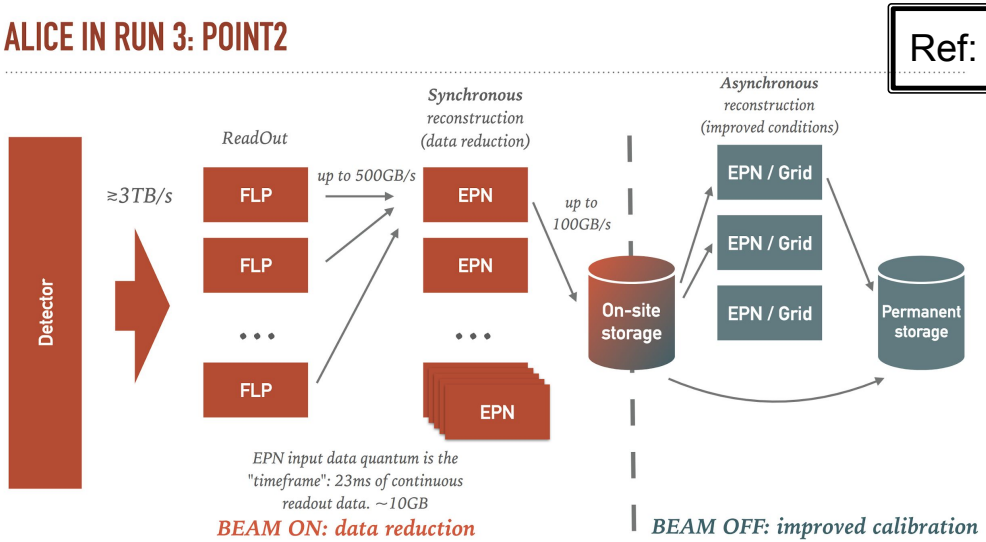
Refs: 1, 2

LHCb Turbo Stream

- If RAW is not to be saved long term, reconstruction needs to be final analysis quality from HLT
- ‘Real time’ alignment and calibration done in ~hours
- HLT 2 does a high quality properly calibrated reconstruction
  - Reduced turbo format stored long term (flexible content)
  - RAW data deleted
- Run 2 turbo is 25% of trigger, but only 10% of bandwidth
- Run 3 will extend this, with no hardware trigger and HLT 1 running at full rate



ALICE in Run 3

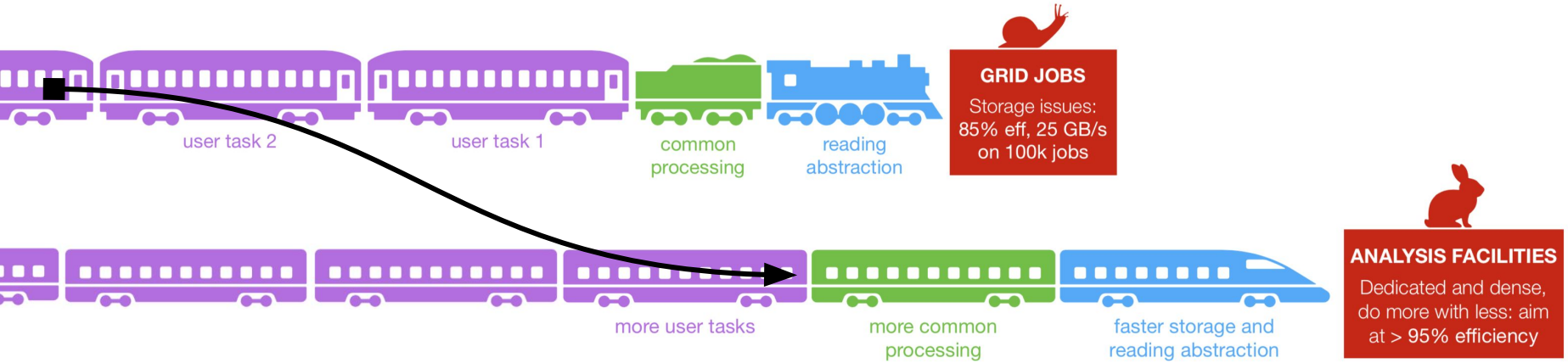


Ref: 1

- Data reduction scheme very similar in spirit to LHCb
- Innovative message passing framework
- Big data chunks based on timeframes of ~1000 bunch crossings
- Pioneered the use of analysis trains
  - Train model is to read analysis inputs only once (the locomotive)
  - But to run many groups' analysis code on the data (the carriages)
  - Amortises the costs of reading large input data sets
- Current problem is that the grid is not very well setup for I/O heavy analysis tasks - generic compute clusters doing simulation and reconstruction as well

Ref: 1

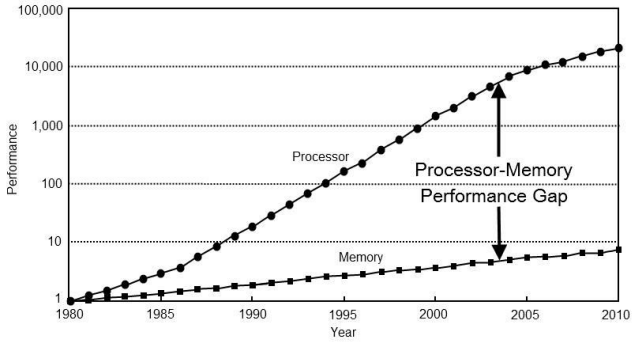
Analysis Clusters

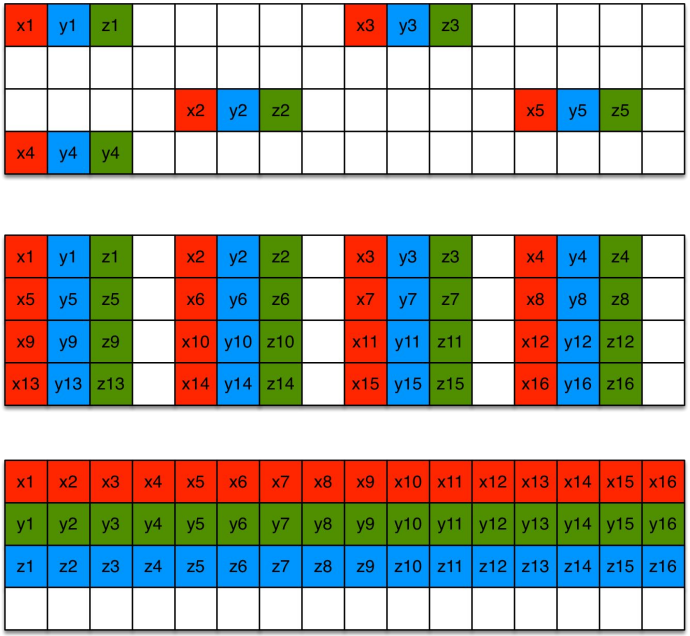


- Dedicated clusters can provide the I/O needed for analysis
- Better compression algorithms and parallelisation
- Improve greatly the data model to ease loading the data into memory
  - Flat data structures, cross references with offsets, no scattered memory

Aside: Data Layout

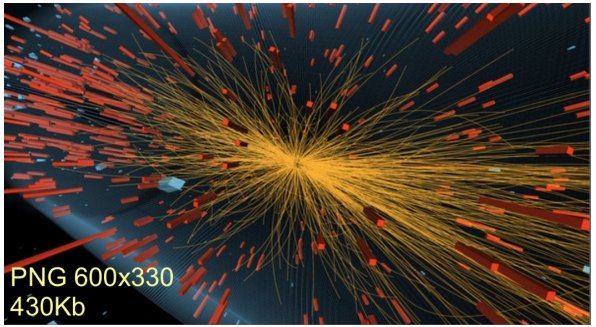
- Modern CPUs run much faster than memory
- Memory cache misses are hugely expensive
  - Many times more loss than gains from, e.g., vectorisation
- Critical to layout data in a friendly way for the CPU
  - Vectorisation friendly
  - Prerequisite to using GPUs
- But present an interface to physicists that looks more natural
  - ATLAS xAOD, LHCb SOAContainer

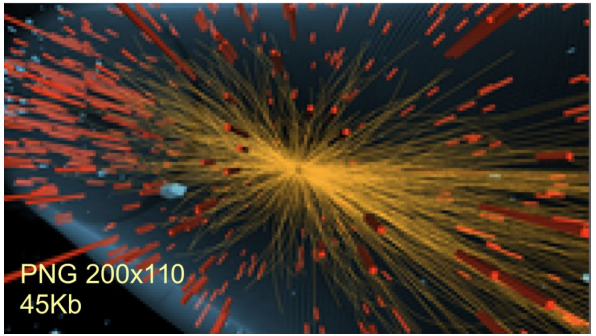




Analysis Data Reduction

- CMS full AOD weighs in at 450kB/evt (on disk)
  - But how much is really needed for analysis?
  - 95% of Run 2 analysis on MiniAOD, 45kB/evt
- Up front decisions made as to what analysis will need
  - This cannot work unless the detector is well understood and the reconstruction robust
- nanoAOD aims to cover 50% with a format that is O(1kB/evt)
  - No tracks or individual particle candidates
  - No detector details
  - Precomputed object IDs
  - No systematics (compute as needed)
  - Reduced precision (not even 32bit floats)
- *Caveat Emptor*: Not yet physics validated





Ref: 1

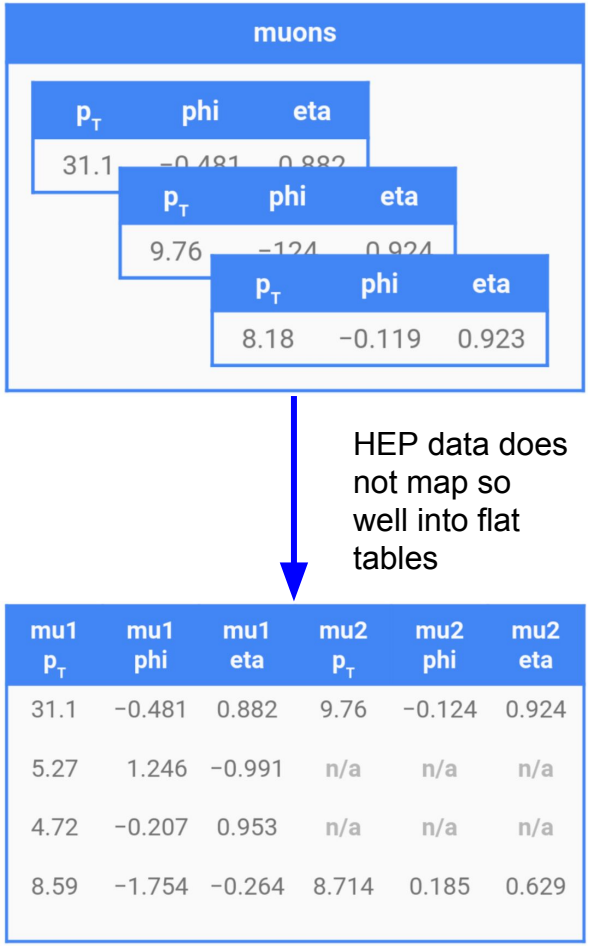
Next Generation Analysis Clusters

- Even with improvements to input data size and formats the process of skimming analysis data is heavy and quite slow
- Industry does not analyse their data like HEP
  - Traditionally used SQL databases
  - Now facilities like Apache Spark clusters or Google BigQuery are now common
  - Underlying structure is not based on files or filesystems now, but “objects”
- Allows data to be addressed more directly at column level
  - Filtering, computing derived data from selections supported
  - Workload is usually split our onto many processing nodes all looking at the same object store
  - Database-like (but of the *NoSQL* variety)

Big Query

For HEP data?

- ...but HEP data isn't flat
  - events naturally have different content and is analysed in sophisticated ways
- For this reason HEP invented its own columnar data format
  - It's a ROOT TTree - we know this is highly efficient and works very well for our data
- Other options
  - Use HDF5 (Hierarchical Data Format)
    - Doesn't perform as well for our data
  - Flatten data in novel ways, spread on event across multiple rows (such as the AwkwardArray library)
- A lot of R&D in this area (FNAL Spark Cluster), but potential benefits would be large



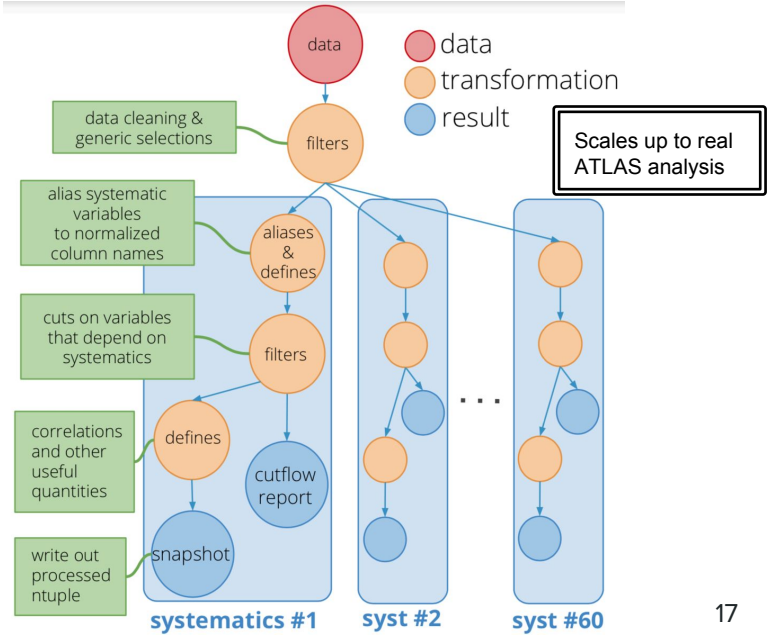
Ref 1, 2

Declarative Analysis

- Notable trend from industry is that there is *no event loop*
- User describes *what* they want to do, *not how* to do it
  - This is actually a big advantage - at the moment analysts need to learn too much boilerplate to run jobs
  - Strive for a simple programming model, easy to use
- Backend system then free to optimise
  - Scaling to 100 threads demonstrated
  - Future proofed for future hardware

```
ROOT::EnableImplicitMT();
ROOT::RDataFrame df(dataset);
auto df2 = df.Filter("x > 0")
             .Define("r2", "x*x + y*y");
auto rHist = df2.Histo1D("r2");
df2.Snapshot("newtree", "out.root");
```

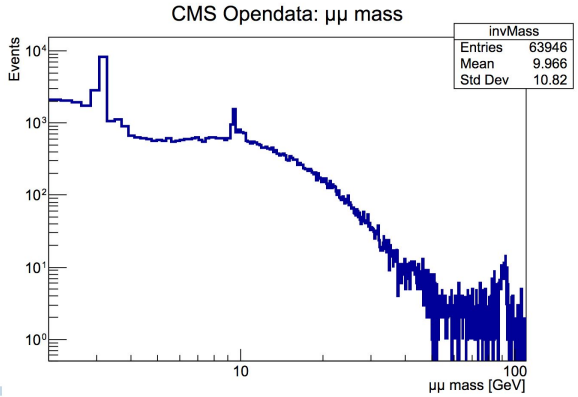
This is for ROOT, but also pure python examples



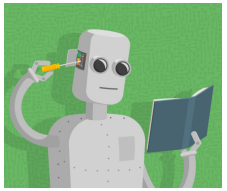
Juypter Notebooks

- Web based technology for running interactive scripts
- Better for training and reproducibility (also reinterpretation)
- Can be used as a portal to large scale resources
  - E.g., Using CERN SWAN service to send jobs to an Apache Spark cluster
- Can allow ‘interactive’ parts of analysis to scale up significantly over laptop or workstation resource
  - But has to offer the same user experience

```
In [7]: invMass = ROOT.TH1F("invMass","CMS Opendata: #mu#mu mass;#mu#mu mass [GeV];Events",512, 2, 110)
        invMassFormula = "sqrt((E1 + E2)^2 - ((px1 + px2)^2 + (py1 + py2)^2 + (pz1 + pz2)^2))"
        cut = "Q1*Q2==-1"
        c = ROOT.TCanvas()
        dimuons.Draw(invMassFormula + " >> invMass",cut,"hist")
        c.SetLogx()
        c.SetLogy()
        c.Draw()
```



https://swan.cern.ch/



Machine Learning

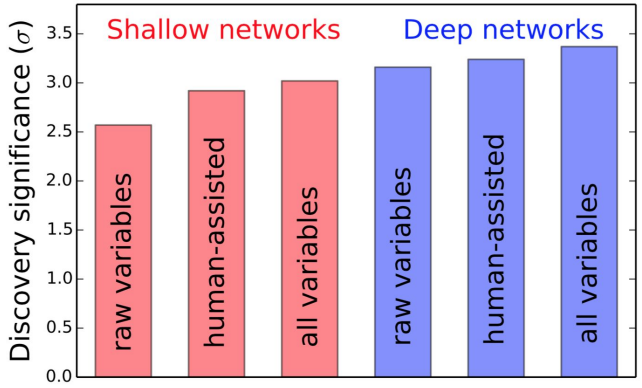
- Probably the hottest topic in IT these days
  - AlphaGo, Self Driving Cars, Language Processing, ...
- Deep Neural Networks are enormous non-linear functions, with huge numbers of free parameters
  - Breakthrough is in being able to *efficiently train* these networks to give a useful response
- Toolkits to are generally very friendly to modern hardware

Driving image classification

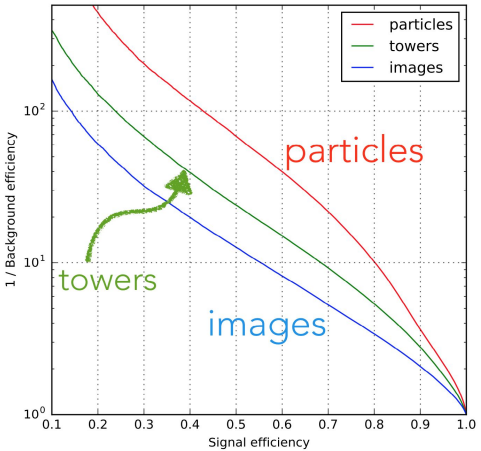
Chihuahua or blueberry muffin?

Machine Learning in HEP

- Techniques clearly work for our field
  - Classifiers improving analysis today (~+50% discovery power)
  - Can reconstruct physics objects even ‘unsupervised’
  - Generative models very interesting for simulation
    - Even simulation straight to analysis output
- Moving beyond ‘naive’ applications to folding in physics knowledge as field matures
  - Needs HEP experts in ML
- Training the network is the significant part of the computing burden
  - Inference is usually fast
    - But can run on accelerated devices, like FPGAs
  - HEP software has to incorporate many networks - memory consumption is a problem



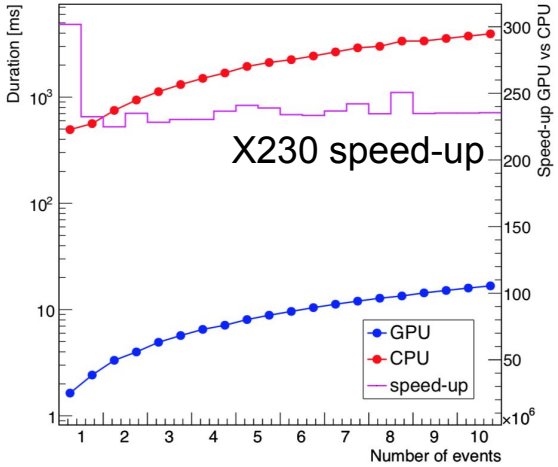
$H \rightarrow \tau^+\tau^-$ significance with different NN setups and with/out ‘high-level’ variables (1410.3469)



Adding physics knowledge to ML W-jet reconstruction improves results (1609.00607)

Accelerated Computing

- GPUs superb at delivering floating point operations
  - Often x10-20 higher than CPUs
  - But difficult to program against in many cases
    - Don’t deal well with branchy code
  - GPGPU cards not cheap, not easy to measure efficiency of use
- Excel at *training* deep learning neural networks
- Data ingestion can be limiting factor for other uses
  - Particularly when few calculations need done on the data
  - E.g., cuts, filters, derived variables
- However, there are some cases where they can help analysis a lot
  - Goofit and Hydra minimiser, very much applicable to analysis with large numbers of toy models



Phase space generator speed-up with Hydra

Conclusions

- Major challenges for software and computing come in the future
  - Run 3 is almost upon us for ALICE and LHCb, HL-LHC for ATLAS and CMS
- Analysis requires software tooling that will deal with a huge increase in events, driven by physics
- How to succeed:
  - Reduce to the data you really need
  - Optimal layout for fast ingestion and processing
  - Declarative syntax for clarity, reproducibility and optimisation (concurrency and parallelisation)
    - Make the backend smart
  - Suitable infrastructure
    - Are dedicated facilities the future here?
  - Take advantage of industry advances, adapted to our problems
    - Modern CPUs and GPUs are everyone's concern here, Machine Learning is a game changer
  - Cooperation and recognition matter a lot

References

- LHCb Turbo Stream: http://iopscience.iop.org/article/10.1088/1742-6596/664/8/082004 https://cds.cern.ch/record/2630473
- ALICE Analysis for Run 3: https://indico.cern.ch/event/587955/contributions/2938126/attachments/1678944/2705295/20180709_-_CHEP2018_-_The_ALICE_Analysis_Framework_for_LHC_Run_3.pdf, https://indico.cern.ch/event/587955/contributions/2938144/attachments/1675256/2705832/2018-7-chep-framework.pdf
- CMS nanoAOD: https://indico.cern.ch/event/587955/contributions/2937531/attachments/1683536/2706024/rizzi-nanoaod-CHEP.pdf
- RDataFrame: https://indico.cern.ch/event/587955/contributions/2937534/attachments/1683046/2704767/RDataframe__CHEP.pdf
- F.A.S.T. Analysis Framwork: https://indico.cern.ch/event/587955/contributions/2937579/attachments/1681008/2707182/BKrikler_CHEP_FAST-BinnedDataframes_16to9.pdf
- Parsl Complete Analysis in a Notebook: https://indico.cern.ch/event/587955/contributions/2937563/attachments/1683321/2707679/2018-07-09_CHEP_Parsl.pdf
- Good Overviews of ML in HEP: https://indico.cern.ch/event/666278/contributions/2830616/attachments/1579293/2495102/Skeikampen-Physics-AI.pdf, https://indico.cern.ch/event/587955/contributions/3012266/attachments/1683944/2706920/tr1807_davidRousseau_CHEP2018_HEPML_final.pdf
- Machine learning on FPGAs: https://indico.cern.ch/event/587955/contributions/2937529/attachments/1683932/2706842/HLS4ML_CHEP2018_Ngadiuba.pdf

Acknowledgements

Many thanks to:

Jakob Blomer, Tommaso Boccali, Pere Mato, Axel Naumann, Elizabeth Sexton-Kennedy,
Andrea Valassi

And to the whole community for their excellent work in HEP Software and Computing