



Computing and Software Challenges

Graeme A Stewart, CERN EP-SFT



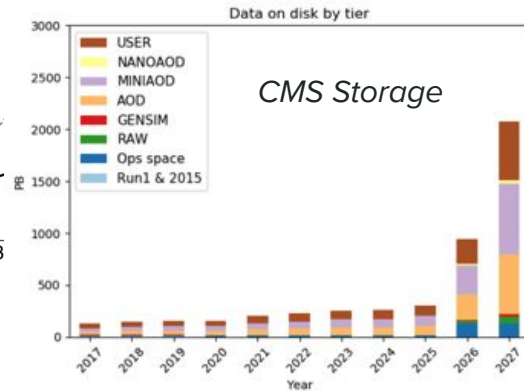
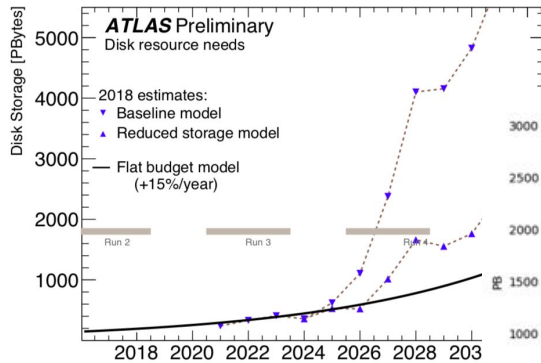
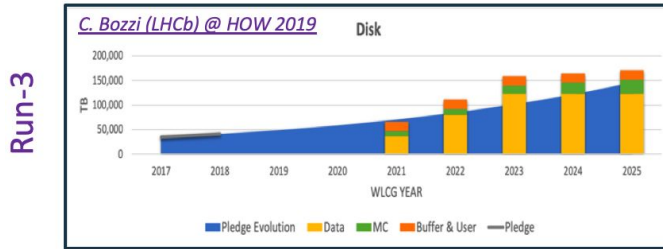
Acknowledgement

- Preparing this talk would not have been possible without the many interesting submissions to the EPPSU
- In particular I drew heavily on talks at the [Granada Workshop](#) from Simone Campana, Ian Bird, Roger Jones, Matthias Kasemann, Maria Girone and Brigitte Vachon

Thank you!

Of course, I take responsibility for any mistakes and misunderstandings and it was my choice as to which work, in particular, to highlight

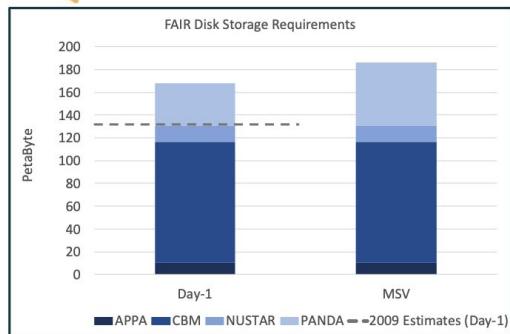
LHC and HL-LHC Challenges



- ALICE and LHCb will have a very large increase in rate for LHC Run-3
 - This puts pressure on both CPU resources and storage
- Move to model of data reduction and software triggers
 - Maximise physics within available resources
- HL-LHC factor 4 in instantaneous luminosity for ATLAS and CMS (7.5×10^{34})
- Trigger rates of 7.5-10kHz
 - Challenge of *rate x complexity*
- Current plots already represent significant improvements over the estimates in the [HSF Community White Paper](#) from 2017

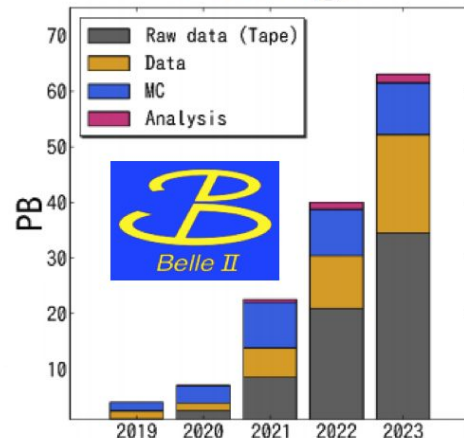
Non-LHC Experiments

- HENP
 - DUNE - Foresee 70PB/year by mid-2020s
 - FAIR - LHC data volumes
 - Belle II - 10PB/year RAW
- Non-HEP
 - SKA
 - LSST
- We will not be alone as a science at the exabyte scale
- This is a both a *threat* and an *opportunity*

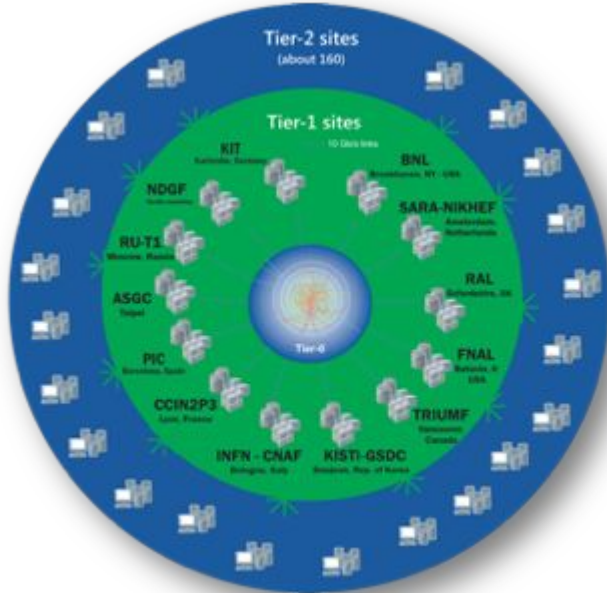


[J. Eschke @ ESCAPE kick-off](#)

Storage



The Scale of HEP Computing



- **WLCG**: an international collaboration to distribute and analyse LHC data
 - Born of the need to scale up our computing to the challenge of the LHC
 - Integrates computer centres worldwide
 - Provide resources as a single infrastructure accessible by all LHC physicists
 - LHC is about 95% of total HEP resources
- 167 sites in 42 countries
- ~1 million CPU cores (100€ each)
- ~1 exabyte of storage (10-100€ per TB)
- >2 million jobs per day
- 10-100Gb network links

The Scale of HEP Software

- At least 50 million lines of code
 - Each LHC experiment has about 6M lines each
- Mostly C++, a lot of Python
- *This would cost at least €500M to develop commercially*
- A lot of significant common software
 - Event Generators
 - Detector Simulation
 - ROOT, foundational toolkit and analysis framework
- A lot of experiment specific software
 - Even when a common solution would have been credible



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

hep cern cms-experiment c-plus-plus

201,818 commits

61 branches



athena

Project ID: 53790

LICENSE 40,776 Commits 36 Branches 2,280 Tags 1.1 GB Files

The ATLAS Experiment's main offline software repository

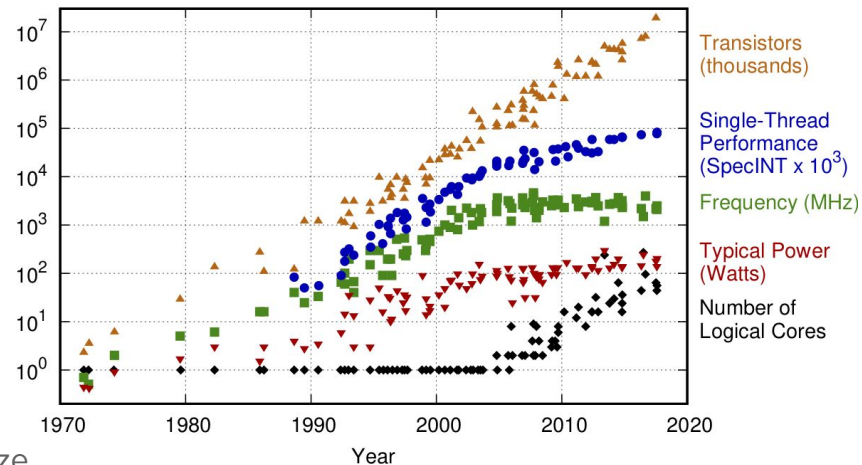


Technology Evolution

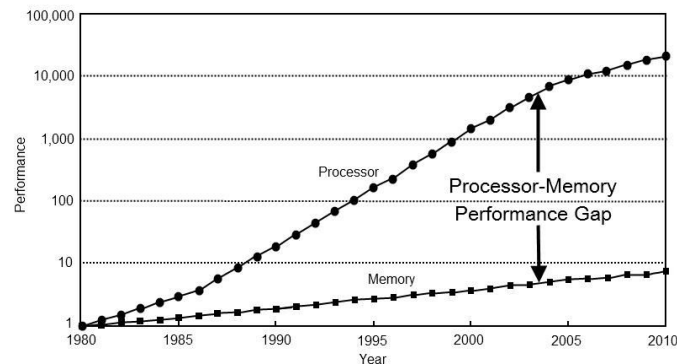
Technology Evolution

- 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 $\sim 3\text{GHz}$ clocks from the underlying Wm^{-2} limit
 - This is the *Power Wall*
 - Limits the capabilities of serial processing
- Memory access times are now $\sim 100\text{s}$ of clock cycles
 - Poor data layouts are catastrophic for software performance

42 Years of Microprocessor Trend Data

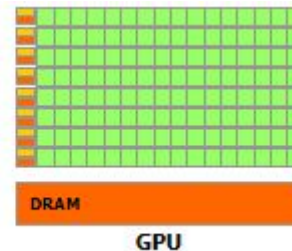
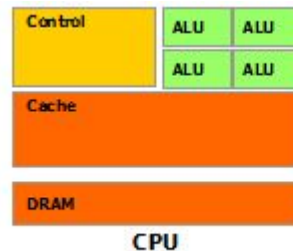
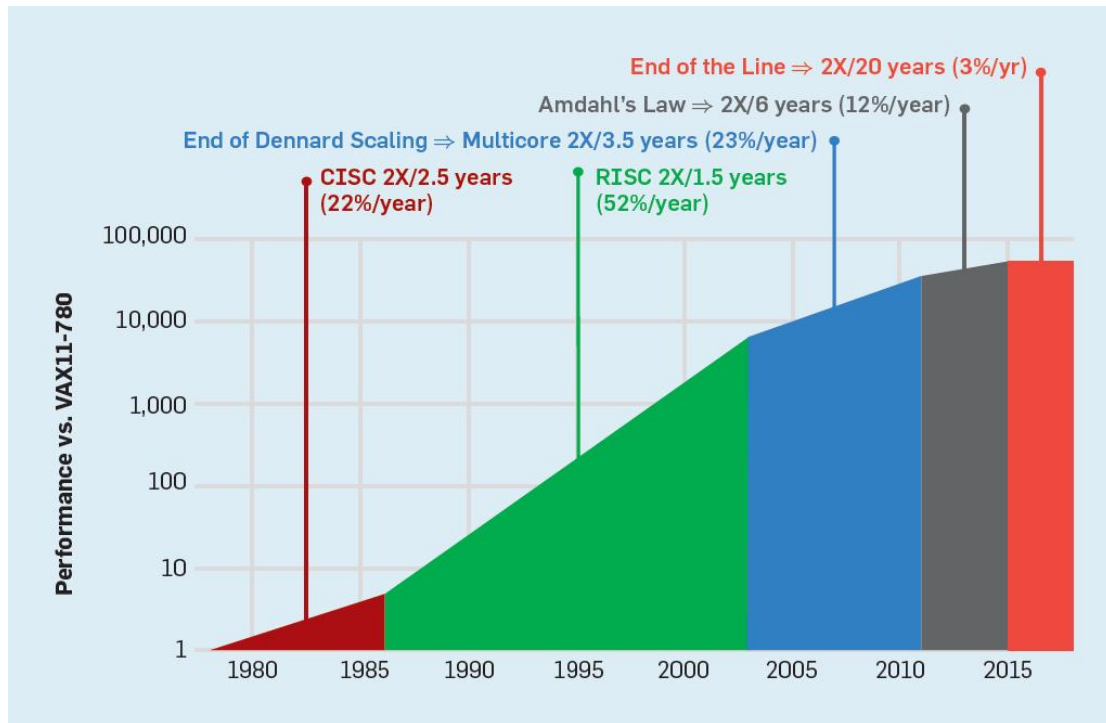


K Rupp



Decreasing Returns over Time

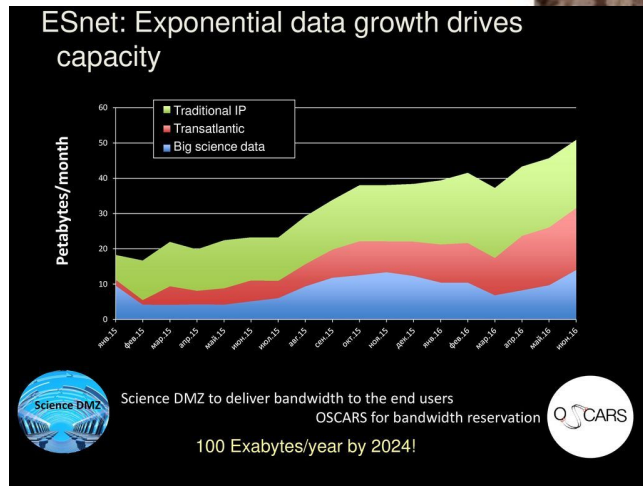
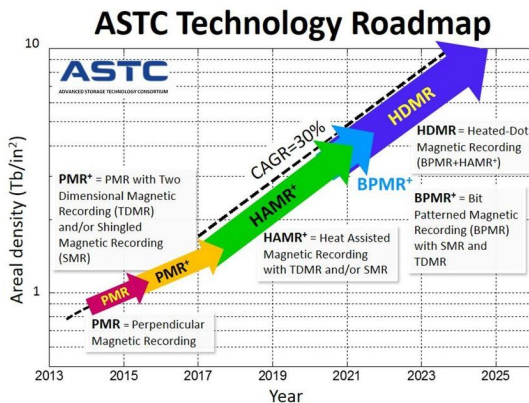
- Conclusion is that diversity of new architectures will only grow
 - We don't know, specifically, what processors will look like in a decade
- Best known example is of GPUs
- But FPGAs and TPUs (Tensor Processing Units) are also used



[ref]

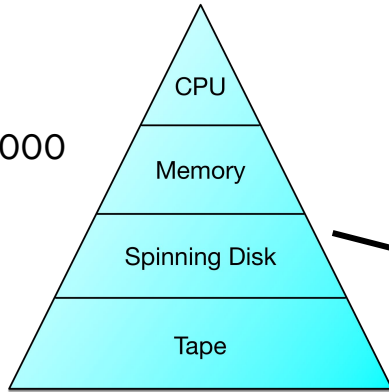
Disk, Tape, Network

- Tape market now dominated by a single manufacturer
 - No serious technological obstacles
 - Non-tape archival storage options are not competitive right now
- Hard disk sizes do still grow
 - 100TB by HL-LHC
 - Time to read a disk's worth of data increases
- Network technology keeps improving
 - Foresee continued increases in available bandwidth and increasing capabilities (SDNs)

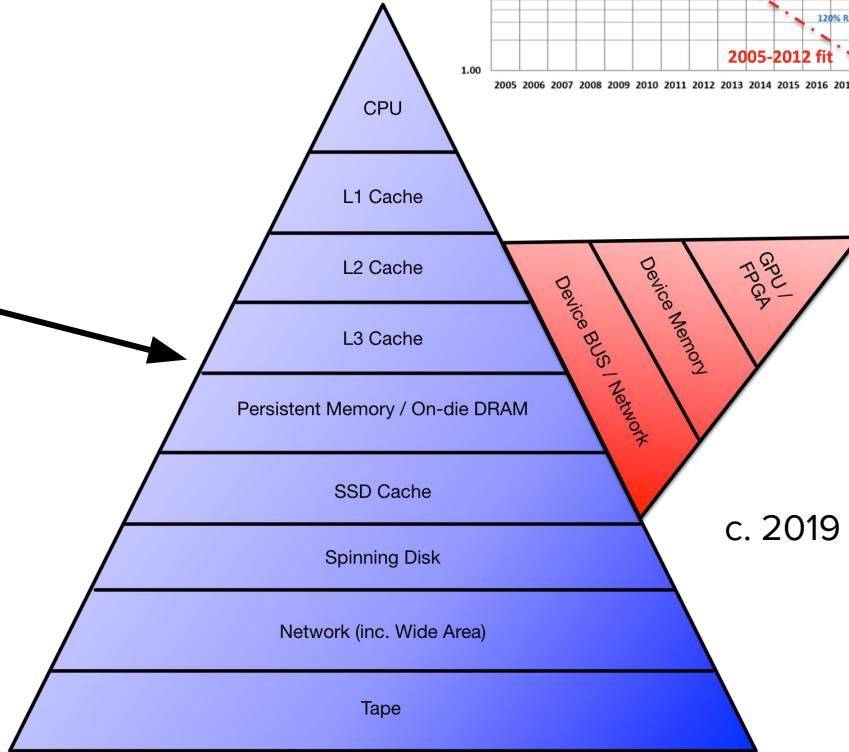


Hardware Evolution in a Nutshell

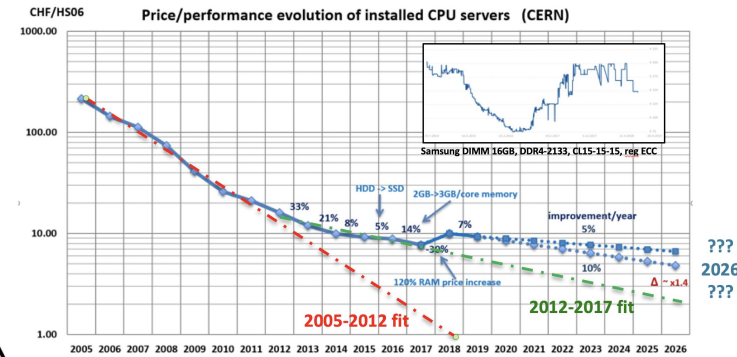
c. 2000



- More complex future
- Rising uncertainties over technology and prices



c. 2019



Challenges and Opportunities

Concurrency and Heterogeneity

- The one overriding characteristic of modern processor hardware is concurrency
 - Doing more than one thing at a time (SIMD, a.k.a. Vectorisation; MIMD, a.k.a. multi-threading)
- Because of the inherently parallel nature of HEP processing a lot of concurrency can be exploited at rough granularity
 - Task and job parallelism served us well for many years
- 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
- There are a lot of possible parallel architectures on the market
 - Different CPU and GPU variants, no real common API to access them
 - To avoid lock-in need to use a wrapper (isolate the main algorithm) or a low level library

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 for message passing and the code generation approaches in FCC-hh
PODIO EDM generator

Machine Learning

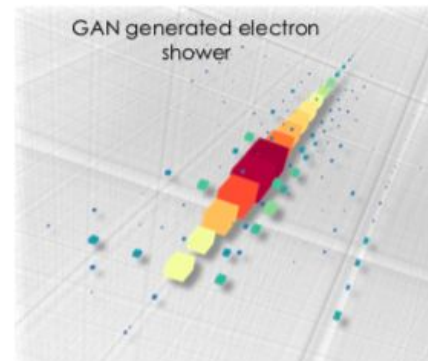
- Machine learning, or artificial intelligence, used for many years in HEP
- Significant advances in the last years in ‘deep learning’
- Rapid development driven by industry
 - Vibrant ecosystem of tools and techniques
 - *Highly optimised for modern, specialised hardware*
- For HEP offers
 - Better discrimination, already used widely
 - Replace slow calculations with trained outputs
 - In extreme cases skip entire processing steps
- Challenge to fully exploit these techniques and to integrate them into workflows

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,

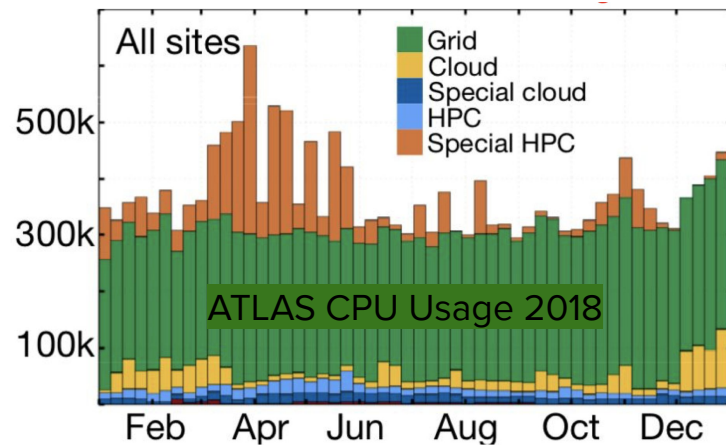
<https://doi.org/10.1038/s41586-018-0361-2>







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

Facilities

- 25% of compute used by LHC experiments already comes from non-grid resources
 - Cloud Computing
 - HPC Centres
 - HLT Farms
- These resources will likely become more important in the future
- **Exascale HPCs** planned around compute accelerators
- Key challenge is their *efficient use*
 - How to utilise their GPUs
 - End to end problem to optimise total throughput
 - Overcome access peculiarities per site



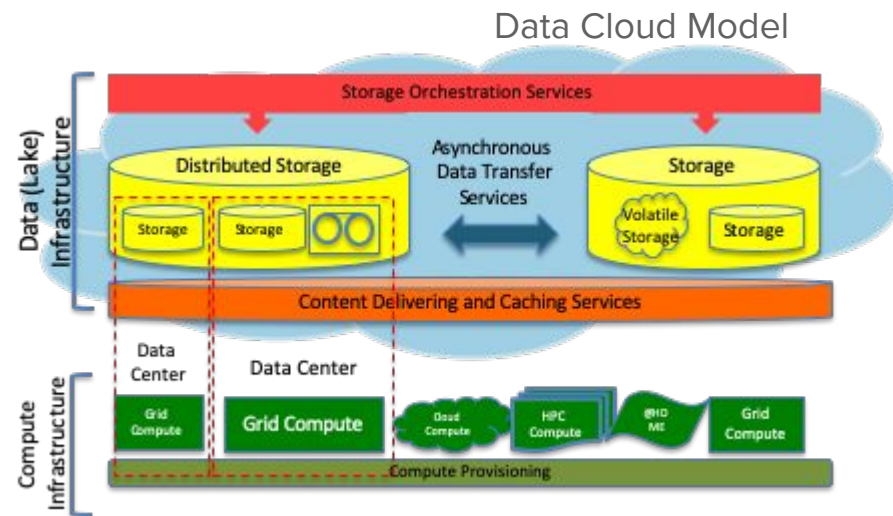
<p>U.S. </p> <p>Sustained ES*: 2022-2023 Peak ES: 2021 Vendors: U.S. Processors: U.S. (some ARM?) Initiatives: NSCI/ECP Cost: \$600M per system, plus heavy R&D investments</p>	<p>EU </p> <p>PEAK ES: 2023-2024 Pre-ES: 2021-2022 Vendors: Likely European Processors: Likely ARM or RISC-V Initiatives: EuroHPC Cost: Over \$350M per system, plus heavy R&D investments</p>
<p>China </p> <p>Sustained ES*: 2021-2022 Peak ES: 2020 Vendors: Chinese (multiple sites) Processors: Chinese (plus U.S.?) 13th 5-Year Plan Cost: \$350-\$500M per system, plus heavy R&D</p>	<p>Japan </p> <p>Sustained ES*: ~2022 Peak ES: Likely as a AI/ML/DL system Vendors: Japanese Processors: Japanese Cost: \$800M-\$1B, this includes both 1 system and the R&D costs They will also do many smaller size systems</p>

ES, EU, Japan and China all planning for exascale machines

HEP Evolution and R&D

Storage and Data Management

- Storage of HEP data is the main challenge in the next decade
 - Data is our main asset, and our main cost
 - No opportunistic storage
 - Petabyte level storage facilities are hard to operate
- *We have massive experience in this area*
- Active R&D into Data Organisation, Management and Access (DOMA)
 - Modernised network protocols
 - Use caches to hide latency, support CPU only sites
 - Data carousels to increase tape use with scheduled access
 - Quality of storage interfaces



Horizon 2020 funding of exabyte scale science infrastructure

Future Shared Infrastructure

- There is an opportunity to leverage commonality across HEP and beyond.
 - This is happening already - compromise between experiment specific and common solutions
 - Sustainability is very important
- Most of the facilities supporting HEP and other science projects are the same.
 - The Funding Agencies do not want to deploy several computing infrastructures
- The idea to generalize the infrastructure related aspects of WLCG and open them to more scientific communities is well received
 - Prototyped with DUNE and Belle-2



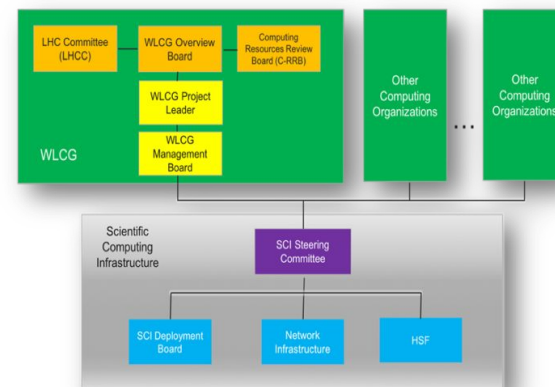
[Users Workshop](#)

[Users Workshop](#)



**CERN VM
File System**

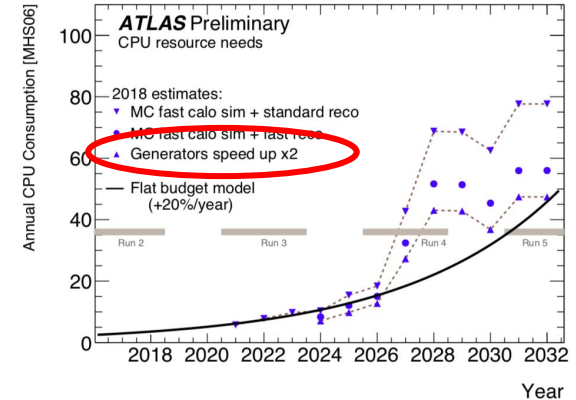
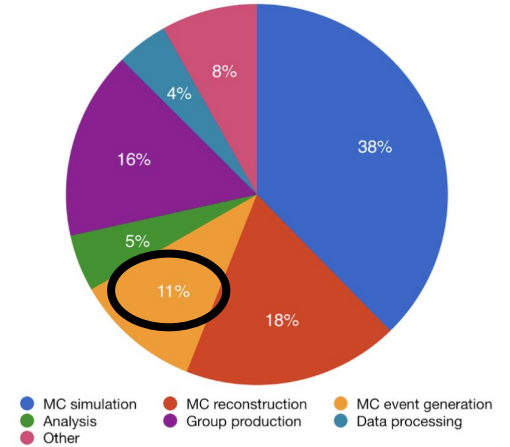
[Users Workshop](#)



Event Generation

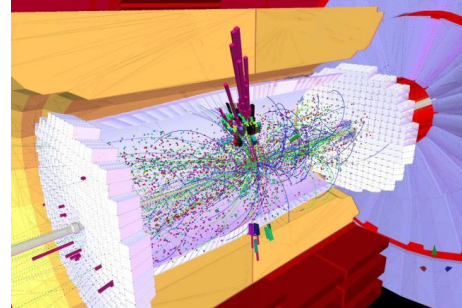
- Starting the simulated events chain from theory
 - Previously was very small part of LHC computing budget (cf. detector simulation), no pressure to optimise
- Increasing use of higher precision to drive down errors (NLO, NNLO, ...); negative weights are a serious problem
 - Greatly increases the CPU budget fraction given over to event generation
 - Possibility of sharing matrix element calculations between experiments being explored ([HSF WG coordinating](#))
- Theory community not rewarded for providing generators to experiments
 - Lack of expertise and incentives to adapt to modern CPU architectures
- From the technical point of view, these codes are a good target for optimisation
 - Might even be suitable for GPUs

ATLAS 2018 CPU Report



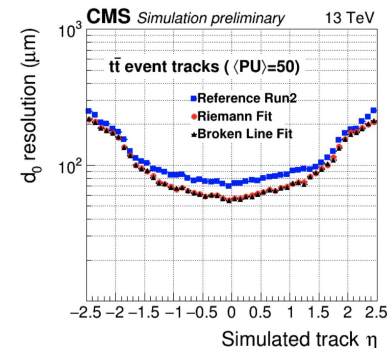
Simulation

- A major consumer of LHC grid resources today
 - Experiments with higher data rates will need to more simulation
- At the same time flat budget scenarios don't give a lot more cycles
 - So need faster simulation
- Technical improvement programme helps (and helps *everyone*)
 - GeantV R&D modernises code and introduces vectorisation; serious studies of GPU porting are starting (US Exascale Computing Project), *but the problem is seriously hard*
- Even this will probably not be sufficient to meet future needs
 - Will need to trade off accuracy for speed with approximate and hybrid simulation approaches
 - Combine full particle transport with faster techniques for non-core pieces of the event
- Machine learning techniques are gaining ground, but yet to be really proven
 - Need to decide when they are good enough cf. Geant4
 - Integrating these into the lifecycle of simulation software and developing toolkits for training and inference is needed - this is a software and a computing problem



Reconstruction and Software Triggers

- Hardware triggers no longer sufficient for modern experiments (LHCb, ALICE)
 - More and more initial reconstruction needs to happen in software
- Close to the machine, need to deal with tremendous rates and get sufficient discrimination
 - Pressure to break with legacy code is high
 - Lots of work in rewriting code for GPUs
- Best practice essential - data layout optimised, concurrent, async
- Even the physics performance can improve when revisiting code
- Real Time Analysis (HEP Version)
 - 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*



(a) d_0 resolution vs η

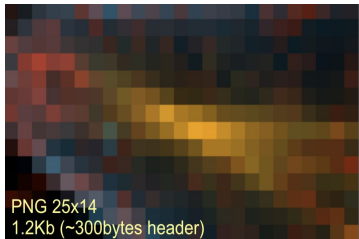
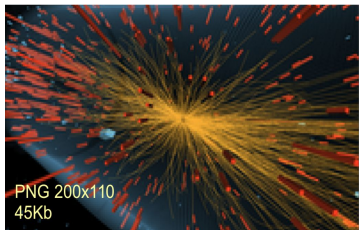
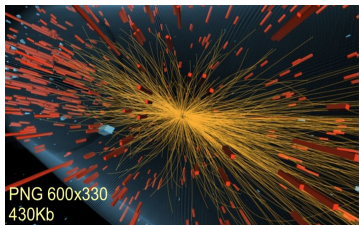
Persistence method	Average event size (kB)
Turbo	7
Selective persistence	16
Complete persistence	48
Raw event	69

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

Analysis



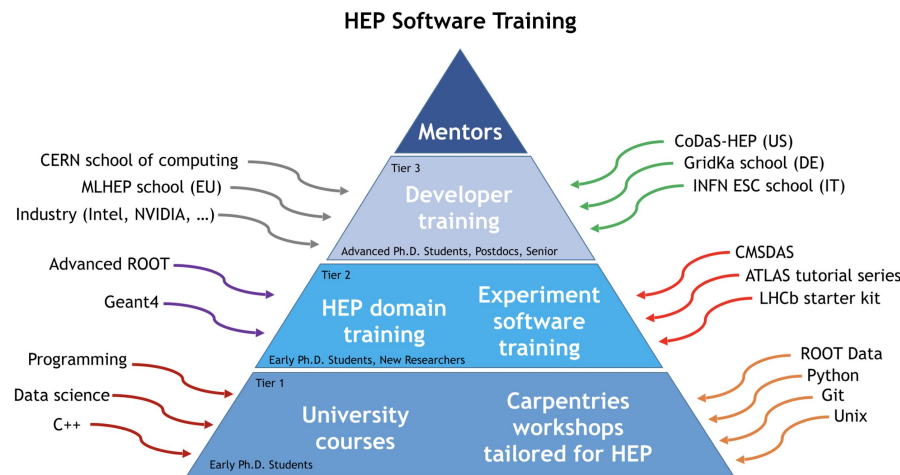
- Scaling for analysis level data also a huge challenge
- Efficient use of analysis data can come with combining many analyses as carriages in a train like model (pioneered by PHENIX and ALICE)
 - Also goes well with techniques like tape carousels
- Reducing volume of data needed also helps hugely
 - CMS ~1kB nanoAOD - a vast difference to analysis efficiency and “papers per petabyte”
- Improve analysis ergonomics - how the user interacts
 - Declarative models (ROOT’s RDataFrame)
 - Say what, not how and let the backend optimise
 - Containers gain ground; notebook interfaces used for training and may scale further
 - Cluster power, laptop convenience - analysis clusters (interactive ROOT on HPCs)
- Interest in data science tools and machine learning is significant for this community - inspiring new approaches (e.g. Coffea)
 - This is an ecosystem into which HEP can, and does, contribute - knowledge transfer goes both ways



Facing the Challenges

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
 - Links to 'Carpentries' being remade (US funded training projects)
- New areas of challenge
 - Concurrency, accelerators, data science
 - Need to foster new C++ expertise (unlikely to be replaced soon as our core language, but needs to be modernised)
- 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
 - Recognition that software is a key part of HEP now



Organising for the Future



- HSF
 - Overarching umbrella organisation, at the international level (strongest in Europe and North America)
 - Builds community efforts, very inclusive; defined the [Community White Paper Roadmap](#)
- Software Institutes
 - IRIS-HEP in US
 - NSF funded at US\$25M over 5 years
 - Machine Learning, DOMA, Innovative Advanced Algorithms, Analysis
 - Should Europe do more here?
 - Traditionally labs (CERN, DESY) have played this role, but time to break out beyond HEP?
 - A lot of shared problems - critical architecture changes, new techniques affect us all
 - Value of the institute is in breaking boundaries (experiment, region, science)
 - Linking to *academic experts in software engineering* could be mutually very beneficial
 - Also helps us to tackle the training problem (pass on skills) and careers (better defined path) and solve practical software problems

Summary



- The landscape has shifted significantly in the last decade
 - *Concurrency, Accelerators, High-Speed Networks, Exascale, ...*
- We are constantly adapting and evolving our software and computing
 - Challenges are not just for current experiments, but R&D for future detectors
- Adopting a more radical approach involves committing **a lot of human effort**
 - It really pays off - *improved software improves our physics*
 - Poor and underfunded software is resource costly or cuts into physics
 - Efficient use of heterogeneous resources needs a critical mass of software
- Pyramid of skills and expertise
 - Need a lot of software engineering and physics talent
 - Address training needs
 - Long term career prospects for HEP software experts need to improve
- Huge opportunities for software to improve *that we have to grasp*
 - Organise around this goal - continue to reach out to industry, software engineers, other sciences

Backup

Optimal Software - The Golden Roles

- Orienting the design around the data (with optimal layouts) is critical
- Bulk data together and exploit concurrency where ever possible
- Be as asynchronous as possible
 - Framework should hide latency
 - Storage systems should help
- Transfers between host and device are expensive
 - Port blocks of algorithms, even ones where gain is small
- The physics performance can improve when revisiting code!
 - We have a lot of legacy; revisiting the code oriented to the primary goal simplifies and improves maintainability

```
StatusCode AuxStoreWrapper::execute() {
    // The StoreGate calls execute() before the object wrapping;
    ATH_MSG_VERBOSE( "Execute before wrapping: %s" << evtStoreC->dump()
                    << "\n" );
    // If we don't have a list of keys, we should get one now:
    const std::vector<string> keys = m_keysSet;
    // Ask StoreGate for a list of keys that it's holding on to:
    const std::vector<string> keys = m_keysSet;
    // Apply them to the objects that it's holding on to:
    for( const SG::Data* > proxies =
        m_keysSet.find( proxy->name() ) == m_keysSet.end() ) {
            // This SG key was not mentioned in the jobOptions;
            ATH_MSG_VERBOSE( "Evaluating object with name \"%s\" << proxy->name()
                            << "\n" and CLID " << proxy->clID() );
            // Check if we need to worry about this object:
            if( m_keysSet.size() ) {
                if( m_keysSet.find( proxy->name() ) == m_keysSet.end() ) {
                    // This SG key was not mentioned in the jobOptions;
                    ATH_MSG_VERBOSE( "Conversion for this key was not
                                    requested" );
                    // But if this is a proxy for an interface container, then let's remember it.
                    // we wrap the auxiliary container, then let's remember it.
                    if( m_keysSet.find( proxy->name() ) == m_keysSet.end() ) {
                        std::string typeName;
                        ATH_CHECK( m_clidSvc->getTypeName( proxy->clID(),
                                                         proxy->name() ) );
                        if( typeName.substr( 0, 6 ) == "xAOD::" ) {
                            if( m_clids[ proxy->name() ] = proxy->clID();
                                continue;
                            }
                            ATH_MSG_VERBOSE( "The wrapping of this object was requested" );
                        }
                    }
                }
            }
            // Remember the CLID of this type, if it's an xAOD type;
            std::string typeName;
            ATH_CHECK( m_clidSvc->getTypeName( proxy->clID(),
                                             proxy->name() ) );
            if( ( typeName.substr( 0, 6 ) == "xAOD::" ) &&
                ( typeName != "xAOD::ByteStreamAuxContainer_v1" ) ) {
                m_clids[ proxy->name() ] = proxy->clID();
            }
        }
    }
}
```

Summary of EPPSU Inputs

- The EPPSU inputs that made mention of software and computing are summarised here:

https://docs.google.com/spreadsheets/d/1mjN6AaS0UUFY-r_HxkKvV4E4f2cgPkEaLc_hEFIHm0LxA/edit?usp=sharing