Software & Computing in modern HEP experiments

Alexey Zhemchugov
JINR
zhemchugov@jinr.ru

Physics Summer School at iThemba LABS
2 February 2023

Outline

- Software & Computing in HEP – brief recap
- Challenges in LHC era
- New trends
- An illustration: SPD computing

Many thanks to Graeme Stewart and Tommaso Boccali, whose talks at the Future Collider Software Workshop in Bologna in June 2019 were heavily (re-)used in this lecture

Online software

Trigger and Event Filter

DAQ

Slow control and monitoring

Offline software

Reconstruction and Calibration

Monte-Carlo simulation

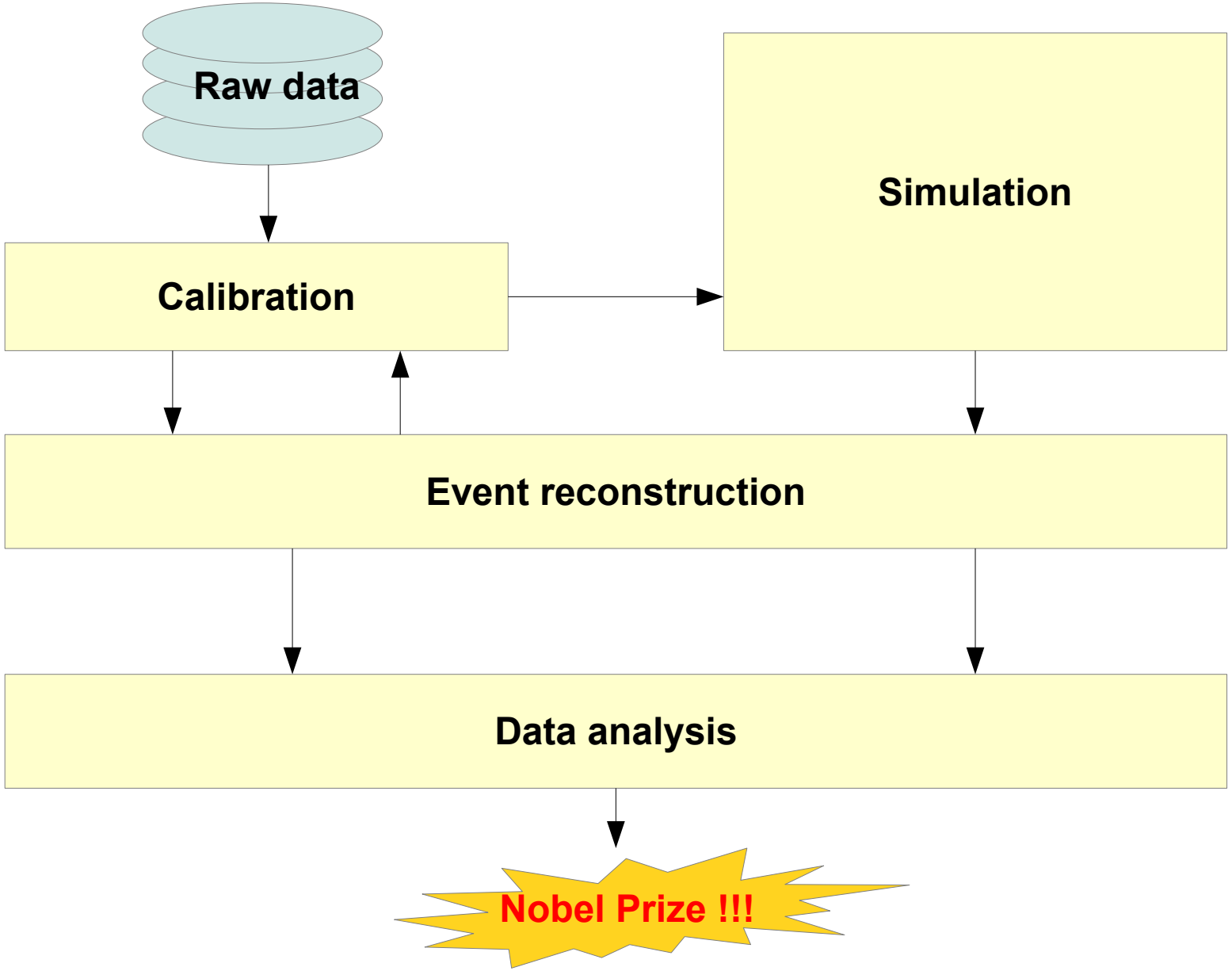
Analysis tools

Raw data

- Digitized detector response
  - electronics channel id
  - signal amplitude (charge, time)
  - signal shape (if Flash ADC is used)
  - scalers
  - …
- Rather simple, but the number of channels can be huge (~10^8 in ATLAS)
- Data stored as bytestream files
- Data format is determined by FE electronics and DAQ
- Design event rate: from 500 Hz (ATLAS) up to 30 kHz (BELLEII)
- Event size: from 10 kB (GlueX) up to 1.6 MB (ATLAS)

0000000 aaaa 1234 0008 0000 0002 0000 0001 0000
0000010 f26b 01c9 7149 0003 0000 0000 0000 0000
0000020 aabb 1234 0005 0000 4653 2d4f 2032 2020
0000030 0009 0000 6152 646e 6d6f 7254 2067 2020
0000040 bbbb 1234 0009 0000 5543 0000 0000 0000
0000050 0000 0000 0000 0000 0000 0000 0000 0000
0000060 0000 0000 cccc 1234 0004 0000 0001 0000
0000070 31ac 0000 34aa aa12 0c6b 0000 0012 0000
0000080 0000 0300 5015 0079 0001 0000 0000 0000
0000090 000a 0000 7c0d 4d45 0014 0000 5543 0000
00000a0 0014 0000 0000 0000 0000 0000 0000 8000
00000b0 0000 0000 0000 0100 0000 0000 34bb bb12
00000c0 0693 0000 0009 0000 0000 0300 5015 00a1
00000d0 0001 0000 0000 0000 0001 0000 0000 8000
00000e0 34cc cc12 043f 0000 000b 0000 0000 0300
00000f0 0030 00a1 0001 0000 0000 0000 0003 0000
0000100 5543 0000 0014 0000 0000 0000 34dd dd12
0000110 0035 0000 0008 0000 0000 0300 0010 00a1
0000120 0001 0000 0000 0000 0000 0000 34ee ee12
0000130 0009 0000 0000 0300 0010 00a1 0000 0000

Offline Data Processing



Calibration

Making correspondence between the detector response and physics quantities measured in the detector

- *Detector alignment*
- *Amplification gains*
- *Drift velocity measurement in gas chambers*
- *Timing and T0 calibration*
- *Energy scale in calorimeters*
- *….*

Depends on signal magnitude, temperature, electric and magnetic fields and many more factors.

Calibration is carried in continuously during the experiment

Reconstruction

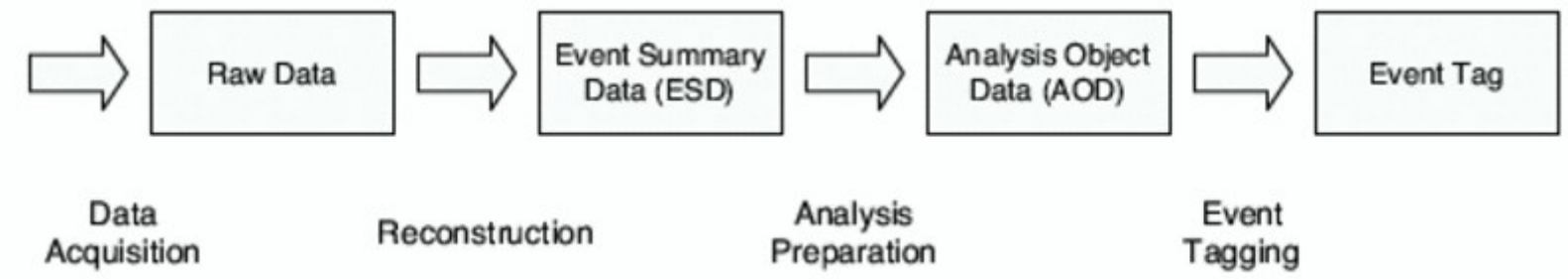
Move from detector response to particle physics

- Pattern recognition
- Track and vertex fitting
- Jet reconstruction
- Reconstruction of showers in calorimeters
- Measuring energy, momentum, time of flight
- Particle identification

Data analysis

- Event selection
- Background suppression
- Corrections for the energy losses, detector acceptance and efficiency, multiple scattering, secondary interactions etc.
- *Getting physics results*
- Systematic error study
- Interpretation and input to theorists

Data Flow



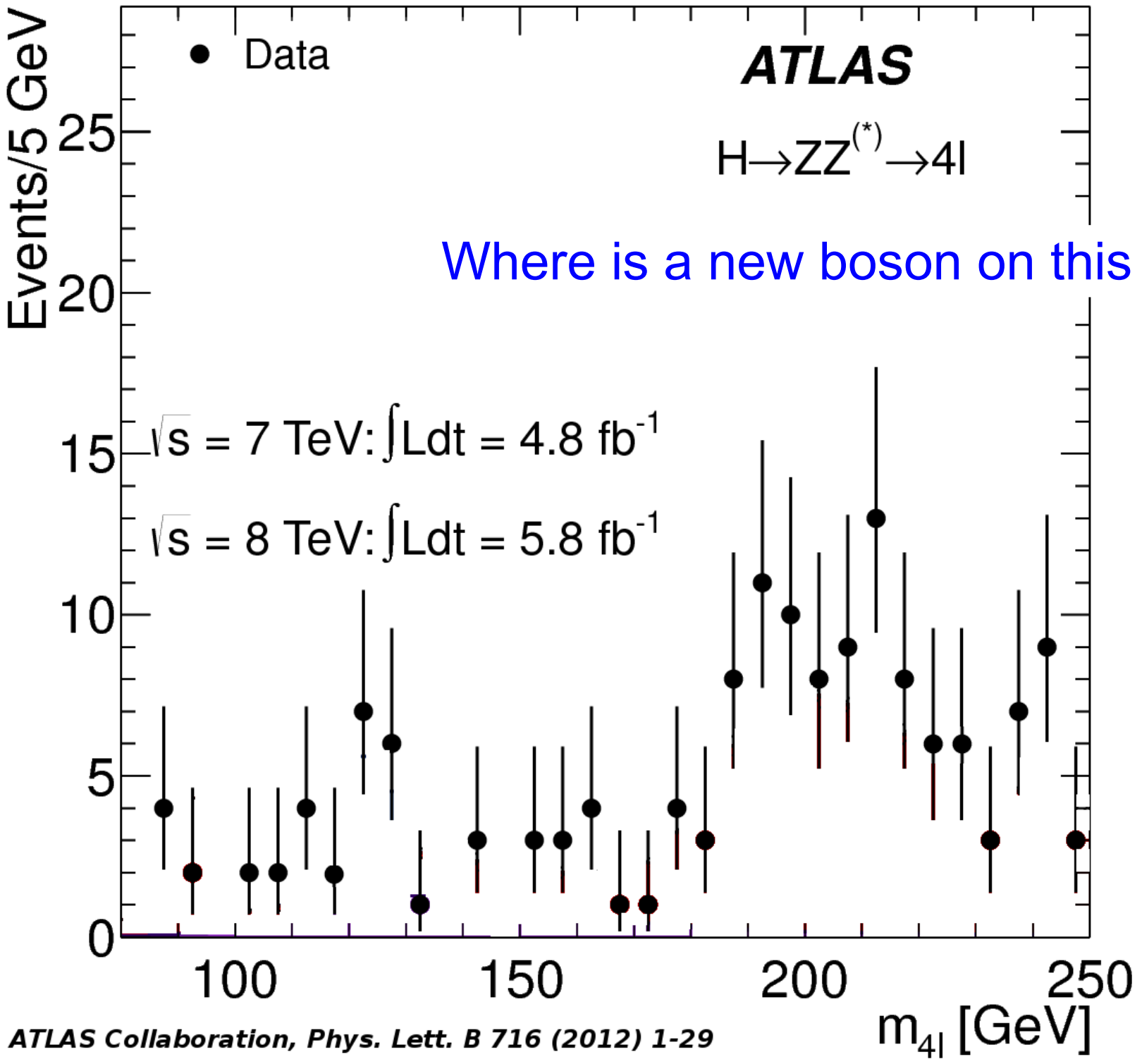
- Data formats
  - **RAW** bytestream from DAQ
  - **REC or ESD** (Event Summary Data): reconstructed physics objects + all input data. Allows to re-reconstruct all events
  - **DST or AOD** (Analysis Object Data): high level physics objects (tracks, jets, particles and so on) ~ 1/10 of ESD
  - **TAG** event summary to select interesting AODs
  - **D3PD** Derived Physics Data — ROOT trees, each Working group uses its own format

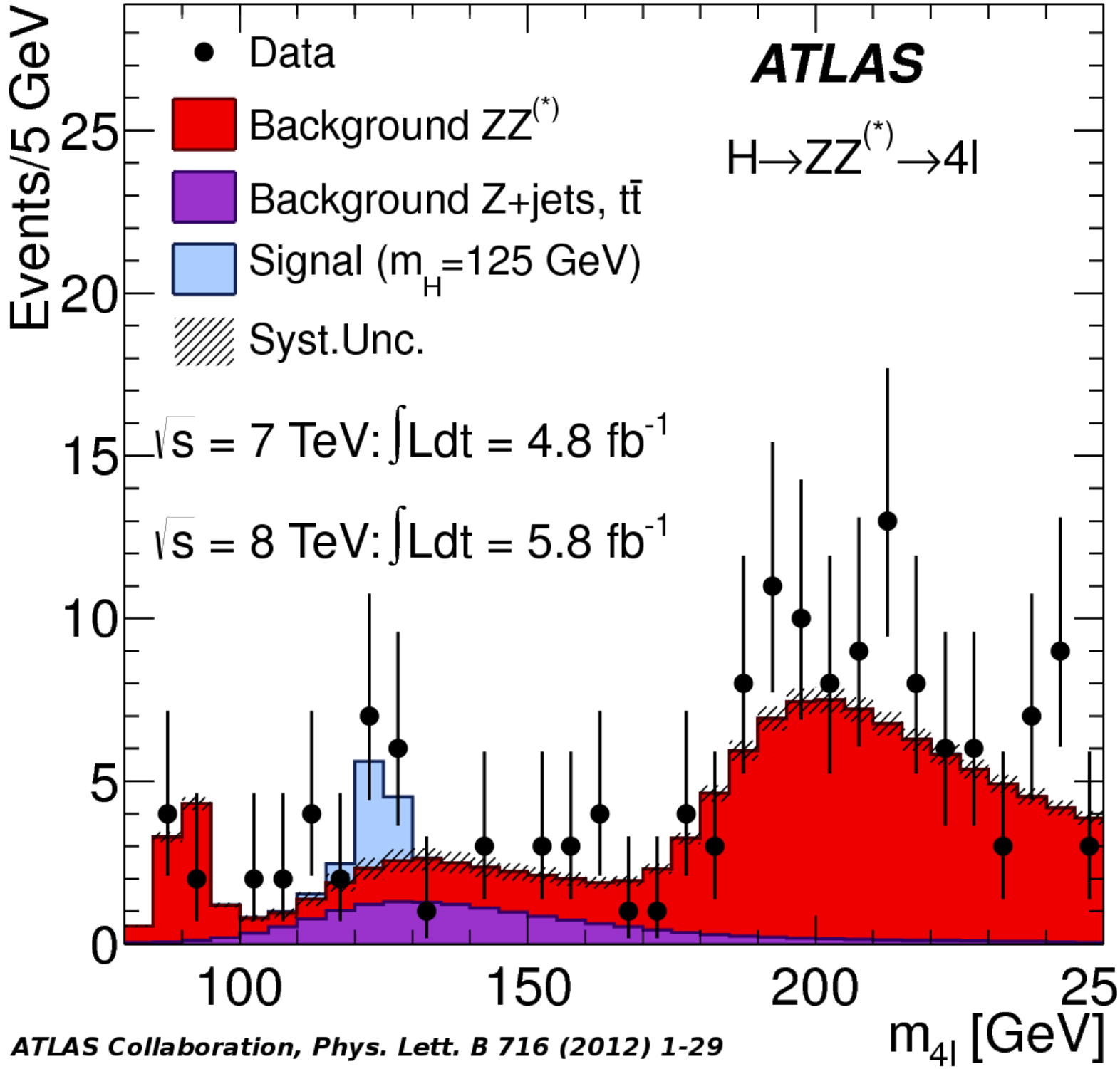
Simulation

Imitating processes in the detector and detector response on the base of known physics phenomena

Usually made by Monte-Carlo method and necessary for:

- *Detector optimization (during R&D and construction phases)*
- *Debugging and tuning the reconstruction software*
- *Data analysis*
  - event selection optimization
  - background contributions
  - study of systematic errors
  - comparison with theory predictions


Data
ATLAS
H→ZZ(*)→4l
Where is a new boson on this plot?
√s = 7 TeV: ∫Ldt = 4.8 fb-1
√s = 8 TeV: ∫Ldt = 5.8 fb-1
Events/5 GeV
0
5
10
15
20
25
100
150
200
250
m4l [GeV]
ATLAS Collaboration, Phys. Lett. B 716 (2012) 1-29



ATLAS Collaboration, Phys. Lett. B 716 (2012) 1-29

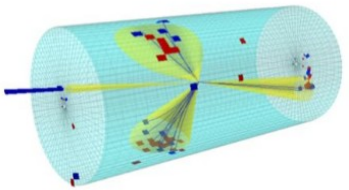
Software Components

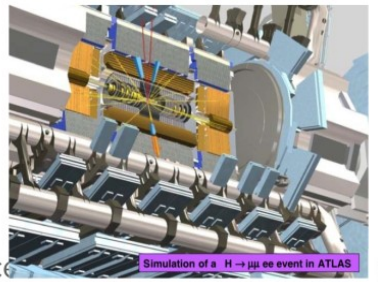
- Foundation Libraries
  - Basic types
  - Utility libraries
  - System isolation libraries
- Mathematical Libraries
  - Special functions
  - Minimization, Random Numbers
- Data Organization
  - Event Data
  - Event Metadata (Event collections)
  - Detector Description
  - Detector Conditions Data
- Data Management Tools
  - Object Persistency
  - Data Distribution and Replication
- Simulation Toolkits
  - Event generators
  - Detector simulation
- Statistical Analysis Tools
  - Histograms, N-tuples
  - Fitting
- Interactivity and User Interfaces
  - GUI
  - Scripting
  - Interactive analysis
- Data Visualization and Graphics
  - Event and Geometry displays
- Distributed Applications
  - Parallel processing (concurrency)
  - Distributed computing (grid/cloud/...)

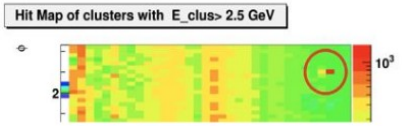
Experiment Software Lifecycle

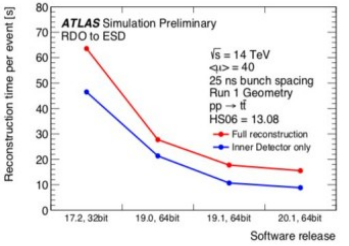
- First ideas and inspiration…
- Concepts
  - Very fast approximate methods, e.g. Delphes (smeared tracks, etc.)
- Design
  - Still need to be flexible to decide between alternatives
  - Ultimately need to pay a lot of attention to details for accurate performance evaluation
    - Accurate geometry, full simulation, realistic digitisation, …
- Production
  - Dealing with the real world - calibration, alignments, dead and noisy elements
  - Learn about the detector, need stability but also continual improvements
- Upgrade
  - Design better sub-detectors for the next version
- Preservation
  - How can I make sure we can look at the data in the future?

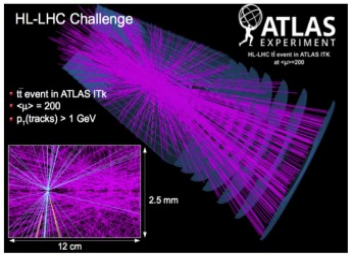
Not everything needs to, or should be, solved up-front, but forgetting about the next step entirely will cause problems down the line (**technical debt!**)









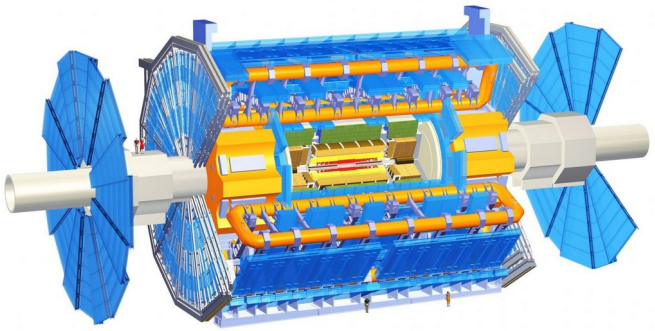



```
TOFGN=TOFG*1.E+9
WRITE(CHMAIL,1000)ITRA,ISTAK,NTMULT,(NAPART(I),I=1,5),TOFGN
CALL GMAIL(0,0)
WRITE(CHMAIL,1100)
CALL GMAIL(0,0)
IEVOLD=IEVENT
NTMOLD=NTMULT
```

Snippet from CERNLIB

Challenges in LHC era

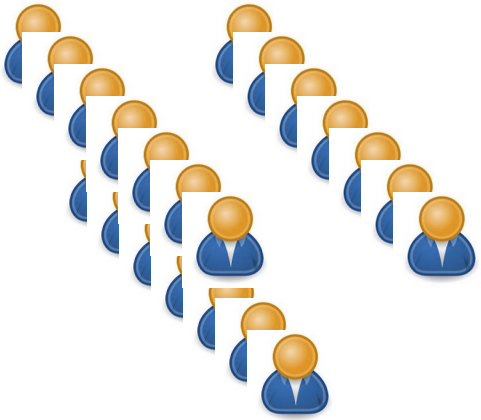
Typical experiment needs



Events to offline
~ GB/s

Process in sync

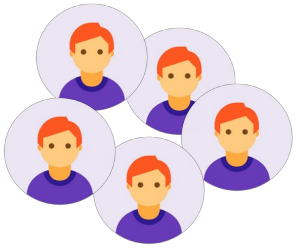
Events to GRID
some GB/s



MC Production teams

2x-3x the
data traffic

Some 500 users
per week
submitting analyses



users

WLCG Total Resource Deployment @ 2018

Tier	Pledge Type	SUM
Tier 0	CPU (HEP-SPEC06)	1,270,000
Tier 1	CPU (HEP-SPEC06)	2,302,398
Tier 2	CPU (HEP-SPEC06)	2,818,192
Tier 0	Disk (Tbytes)	96,700
Tier 1	Disk (Tbytes)	221,912
Tier 2	Disk (Tbytes)	210,615
Tier 0	Tape (Tbytes)	272,200
Tier 1	Tape (Tbytes)	499,899

~650k CPU cores

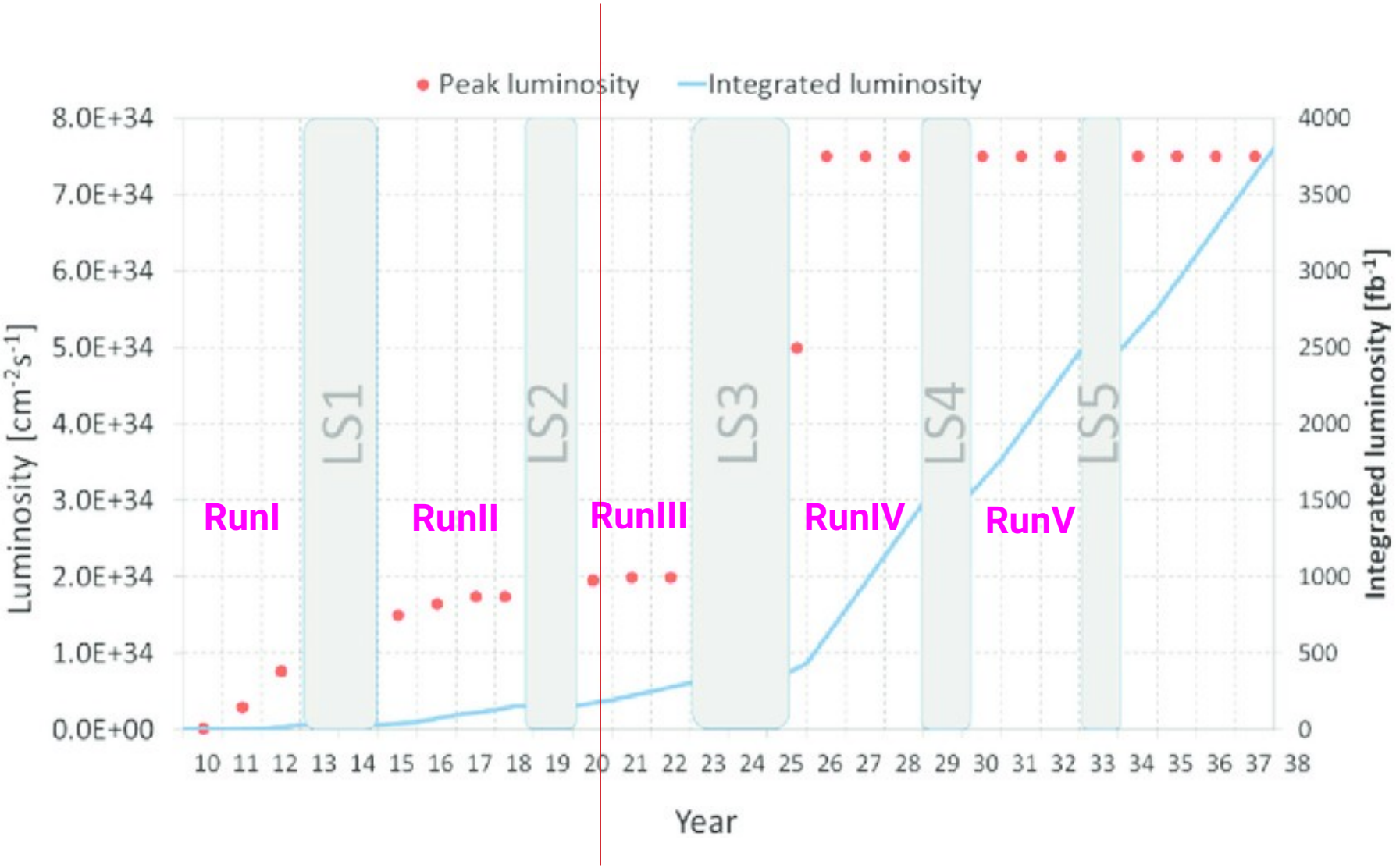
~530 PB disk

~770 PB disk

In reality more than this:

- Experiments report overpledges 1.5-2x
- HPC centers, filter farms, T3s, Commercial Clouds…

LHC prospects



Other experiments

- Well, LHC experiments are very big
- What about other experiments?
  - BELLEII

arXiv:1308.0672

Year	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Tape [PB]	2.8	2.8	2.8	2.8	19.24	54.43	103.55	153.89	204.64	255.39
Disk [PB]	4.00	4.00	5.00	8.00	27.98	79.17	115.68	153.10	190.82	228.55
CPU [kHepSPEC]	45.00	45.00	50.00	55.00	328.31	568.98	567.54	609.45	643.14	672.60

Table 1: Total required Belle II computing resources

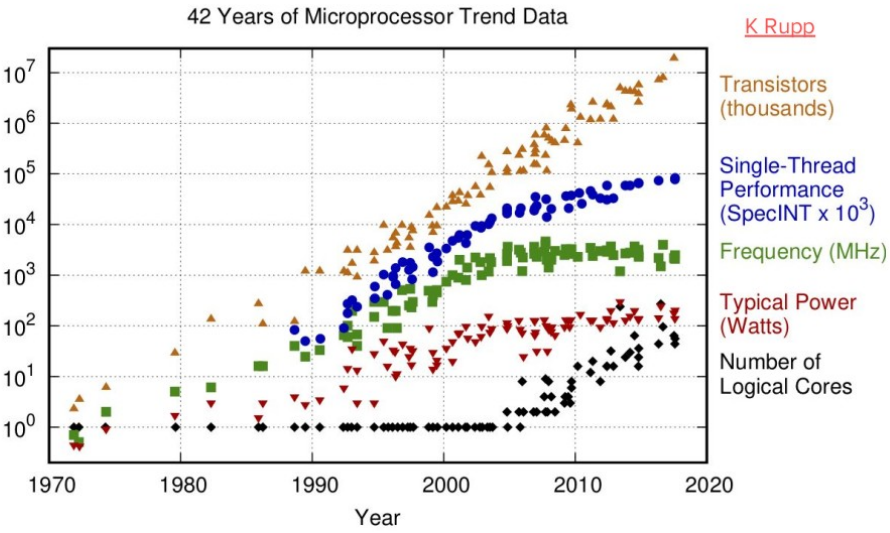
  - GlueX

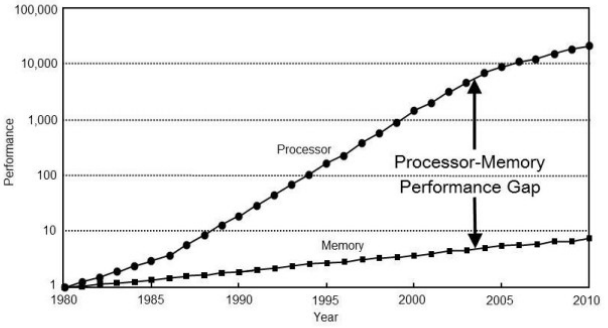
	2017 (low intensity GlueX)	2018 (low intensity GlueX)	2019 (PrimEx)	2019 (high intensity GlueX)
Real Data	1.2PB	6.3PB	1.3PB	3.1PB
MC Data	0.1PB	0.38PB	0.16PB	0.3PB
Total Data	**1.3PB**	**6.6PB**	**1.4PB**	**3.4PB**
Real Data CPU	21.3Mhr	67.2Mhr	6.4Mhr	39.6Mhr
MC CPU	3.0Mhr	11.3MHr	1.2Mhr	8.0Mhr
Total CPU	**24.3PB**	**78.4Mhr**	**7.6Mhr**	**47.5Mhr**

Data size of O(10 - 50) PB/year is getting routine

Technology Evolution

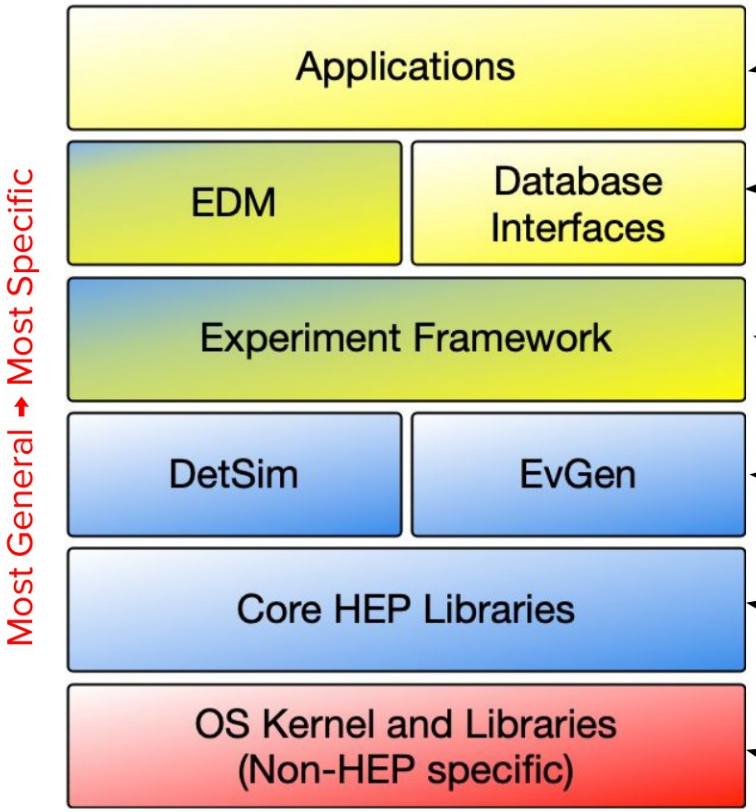
- Moore's Law continues to deliver increases in transistor density
  - But, doubling time is lengthening
- Clock speed scaling failed around 2006
  - No longer possible to ramp the clock speed as process size shrinks
- Basically stuck at ~3GHz clocks from the underlying Wm^{-2} limit
  - Limits the capabilities of serial processing
- Memory access times are now ~100s of clock cycles
  - Poor data layouts are catastrophic for software performance
- Conclusion is that diversity of new architectures will only grow
  - Best known example is of GPUs





New trends

HEP Software



Application layer of modules/algorithms/processors that perform physics tasks (some generic examples like FastJet and PandoraPFA)

Usually experiment specific libraries for data representation and access: xAOD (but LCIO!); also detector specific conditions data

Experiment core orchestration layer, where everything else plugs in: Gaudi, CMSSW, Marlin, FairRoot

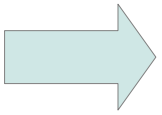
Specific components used by many experiments: Geant4, DELPHES, Pythia, ...

Provide core functionality widely used: ROOT, HepMC, HepPDT, DD4hep, ...

Many widely used non-HEP libraries: Boost, Python, Zlib, CMake, ...

EDM

- **Event Data Model**
- Simulation has been *de facto* standartized = Geant4
- Reconstruction hasn't been yet. Why?
  - all detectors are different
  - data model is determined by the detector construction and physics goals
- Still, all trackers, calorimeters, muon systems do the same job in all experiments
  - position
  - energy deposit
  - time
  - ...

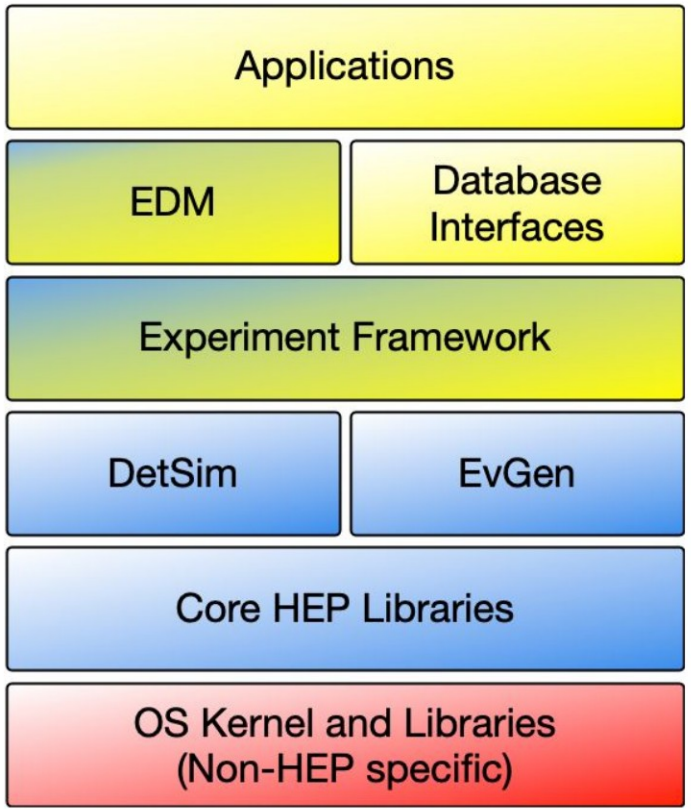


tracks
vertices
showers
PID
...

- Unified EDM is the first step to the standartized reconstruction (FastJet, PandoraPFA, GenFit, ACTS …)
- Good example of LCIO of ILC experiments
- EDM4HEP project: *https://github.com/key4hep/EDM4hep*

HEP Software

Building HEP Applications



- Each piece of software does not live in isolation
- There is an ecosystem of interacting pieces
- Compatibility between the elements doesn't usually come for free
  - Common standards do help a lot
- Building a consistent set of software for an experiment is a task in itself
  - But the software used to do it benefits from commonality
  - LCGCMake, Spack, etc.

Source Lines of Code

Table 6. SLOCCount measured lines of source code for ATLAS and CMS.

Experiment Type	Source Lines of code (SLOC)	Development effort (person-years)	Total estimated cost to develop
ATLAS	5.5M	1630	220 M$
CMS	4.8M	1490	200 M$

The Core Frameworks:

ATLAS/Gaudi: 115k SLOC, 29 FTEy, 4M$
CMSSW/FWCore and friends: 325k SLOC, 87 FTEy, 12M$

Just to set the scale:
Linux Kernel is: 15M sloc, 4800 FTEy, 650M$ (3x CMS)
Geant4 is: 1.2M sloc, 330 FTEy, 45 M$ (1/4x CMS)

Languages

- Community has standardised on two core languages: C++ and Python
- Other languages do circulate, but rarely seem to offer sufficient advantage over the two current dominant languages to really motivate a change
- C++
  - Extremely high performance when used correctly
  - Very large language that can do anything, certainly more than we need...
  - Training in correct modern use and good framework support essential
- Python
  - Fast to develop in, easy for prototyping
  - Good glue language to express concepts (configuration) or shim different pieces
  - Backed by an increasingly large ecosystem of high performance code (data analytics, machine learning)

Concurrency and Heterogeneity

- 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
- Because of the inherently parallel nature of HEP processing a lot of concurrency can be exploited at rough granularity
  - However, the push to highly parallel processing (1000s of GPU cores) requires **parallel algorithms**
- Developments to exploit heterogeneous architectures are happening now in the current experiment's frameworks
  - Gaudi, CMSSW, ALFA/O2 in particular
  - This requires advanced designs to hide latency and also careful study of appropriate data layouts
    - Data layouts for the LHC experiments are being optimised - it's clear that some level of abstraction for clients is important to ease that process

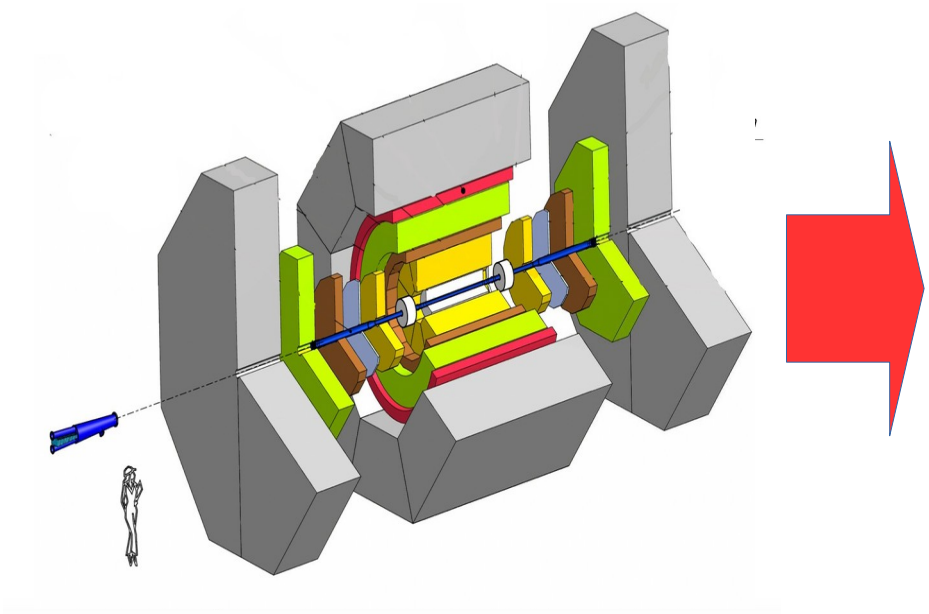
The Problem: Physicists 📠 Software Engineers

Machine learning

- Machine learning demonstrate a turbulent development during last decade
  - Standard tools exist: *tensorflow*, *pytorch*, *theano* etc.
  - Easy to run on CPU and GPU. Can be programmed into FPGA.
  - Hot topic everywhere. Present in any smartphone
- Limited use in particle physics:
  - requires excellent simulation
  - difficult to control systematics
  - application to the data analysis ends up with a big disappontment
- Reconstruction algorithms?
  - track finding, shower reconstruction, …
  - cross-check during data analysis like any reconstruction algorithm
- Trigger ? - already in use at HL-LHC
- The potential of ML in HEP is enormous (provided it is used properly)

An illustration: SPD computing

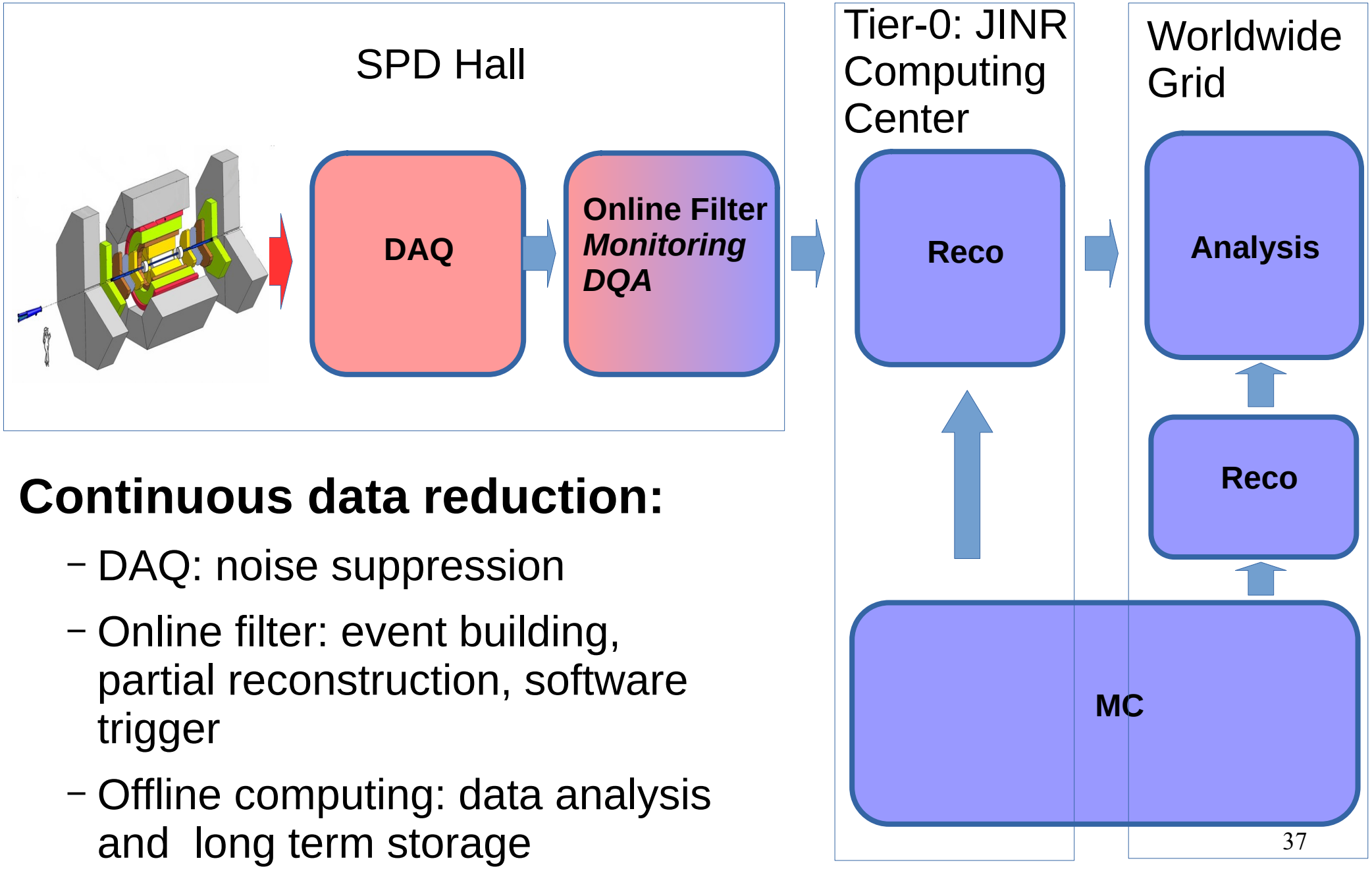
SPD as a data source



- Bunch crossing every 76.3 ns = crossing rate 13 MHz
- ~ 3 MHz event rate (at 10^{32} $cm^{-2}s^{-1}$ design luminosity)
- 20 GB/s (or 200 PB/year (raw data), $3*10^{13}$ events/year)
- Selection of physics signal requires momentum and vertex reconstruction → no simple trigger is possible

The SPD detector is a medium scale setup in size, but a large scale one in data rate!

Data workflow



Continuous data reduction:

- DAQ: noise suppression
- Online filter: event building, partial reconstruction, software trigger
- Offline computing: data analysis and long term storage

Online Data Filter

High-performance heterogeneous computing cluster

- Partial reconstruction
  - Fast tracking and vertex reconstruction
  - Fast ECAL clustering
- Event unscrambling
- Software trigger
  - several data streams

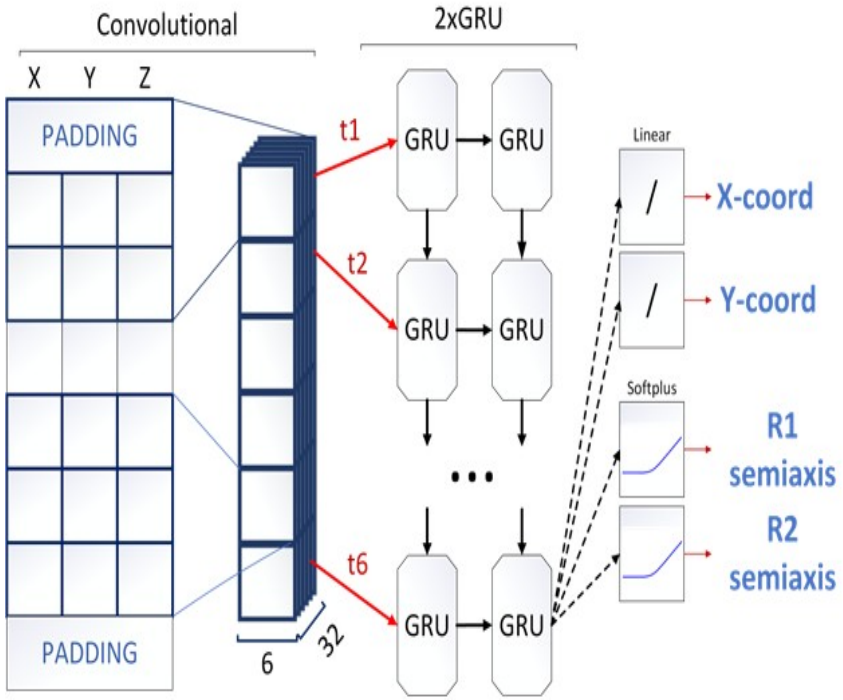
Machine learning is a key technology

Control of systematics?

- Monitoring and Data quality assessment
- Local polarimetry

Example: TrackNETv3 for track recognition

https://arxiv.org/abs/2210.00599

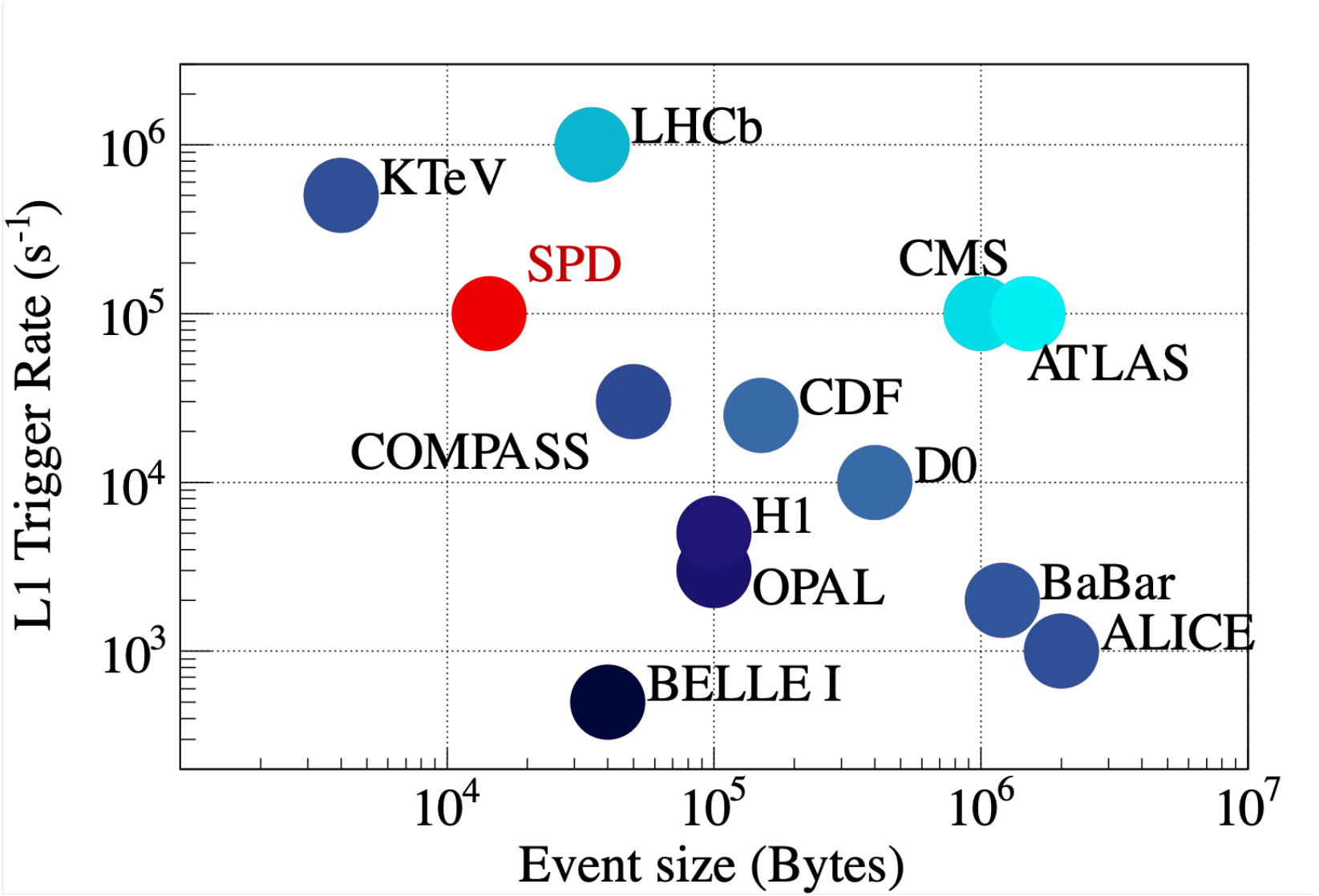


- Network predicts an area at the next detector layer where to search for the track continuation
- If continuation is found the hit is added to the track candidate and the procedure repeats again
- Essentially reproduces the idea of the Kalman filter: track parameters are predicted by synaptic weights determined by network training
- Generalization? Stability? Missing hits?

	Single events	Time slices of 40 events
Track efficiency (recall) (%)	99,62	96,78
Track purity (precision) (%)	99,52	88,02
Time slices / sec	48,70	43,52 (*40 = 1741,19)

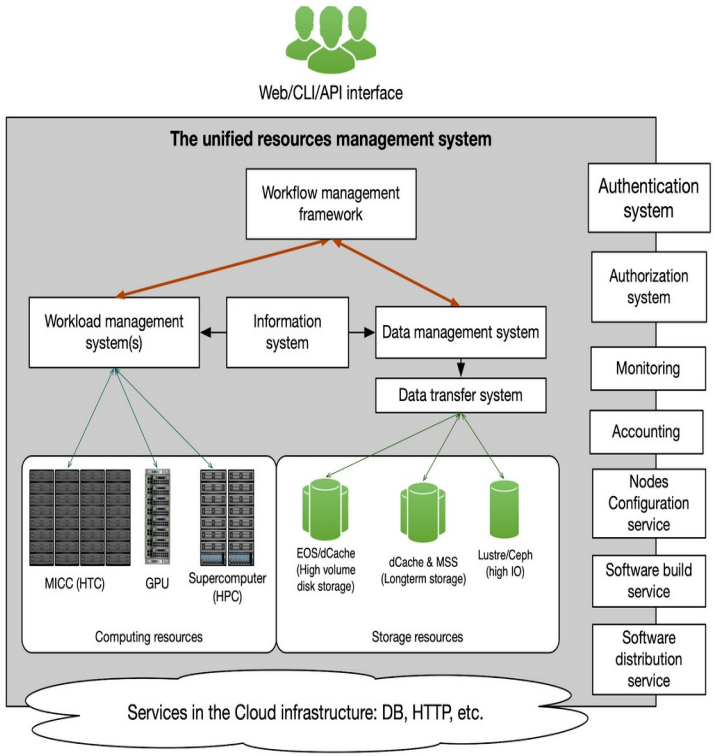
PRELIMINARY

After the online filter



Distributed computing system

By 2030:
- up to 30 PB of storage
- up to 1.5 Pflops of computing power



All basic components are already available from LHC experiments:

- *Workload management: likely PANDA*
- *Data management: RUCIO and FTS*
- *Software distribution: CVMFS*

Adaptation to operate with the SPD event model and offline software is needed

Summary

- Software and Computing remains, and even becomes more important component in a modern HEP experiment
- Large data rate and data volume demand highly parallel computing
  - multicore CPU, GPU, FPGA ...
  - The key problem: writing efficient code for heterogeneous computing platform is not easy for physicists
- Unification of the HEP software
  - Common simulation = Geant4
  - Common reconstruction bricks ??
  - Few universal frameworks in the market: Gaudi, FairRoot/ALFA, Key4HEP?
  - Unified distributed computing (PANDA, DIRAC, RUCIO)
- Machine Learning is a largely underestimated tool for HEP