

Technical
University
of Munich

TUM



HL-LHC and Beyond Computing Challenges

Vangelis Kourlitis

on behalf of the HEP Software Foundation

11th Edition of the Large Hadron Collider Physics Conference - 23 May 2023

Synopsis



HL-LHC Computing & HSF

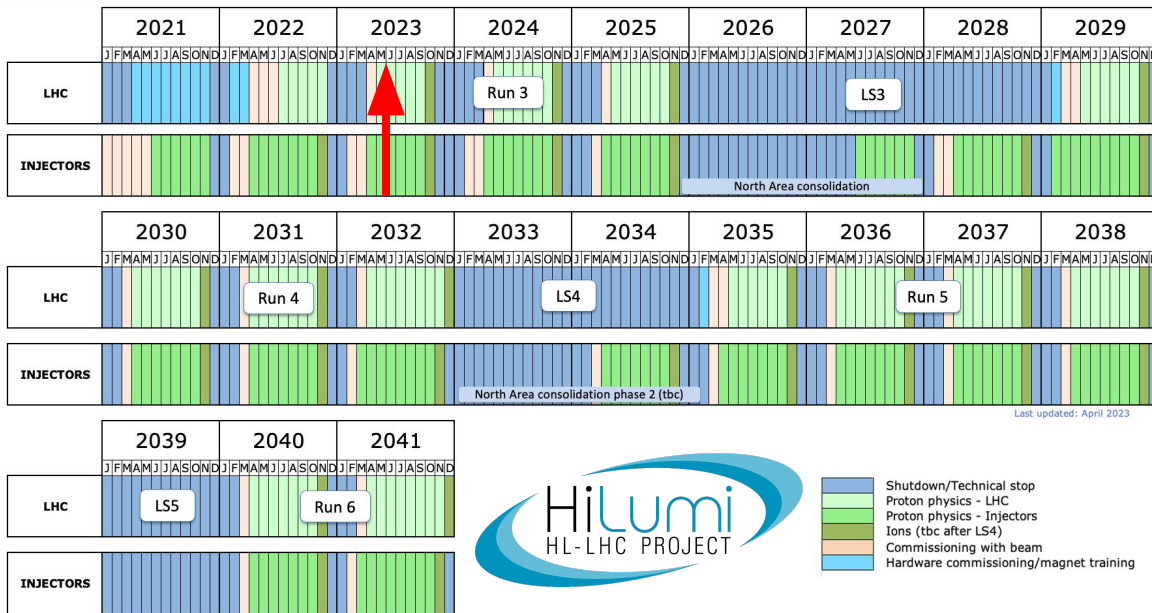
Timeline & Strategy
Resources Evolution

Energy Frontier Computing Challenges

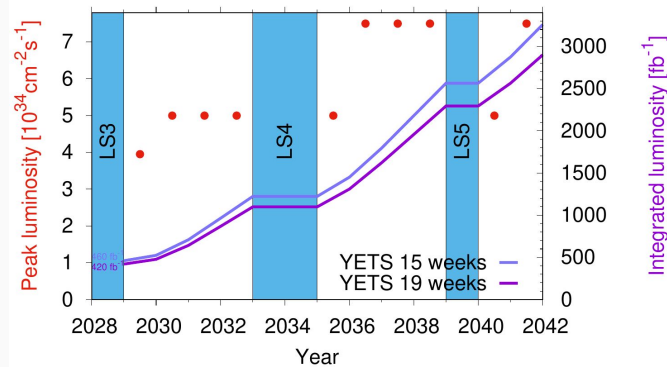
Trigger
Event Generation
Detector Simulation
Event Reconstruction
Data Analysis

HL-LHC Timeline

Longer term LHC schedule



LHC luminosity forecasts



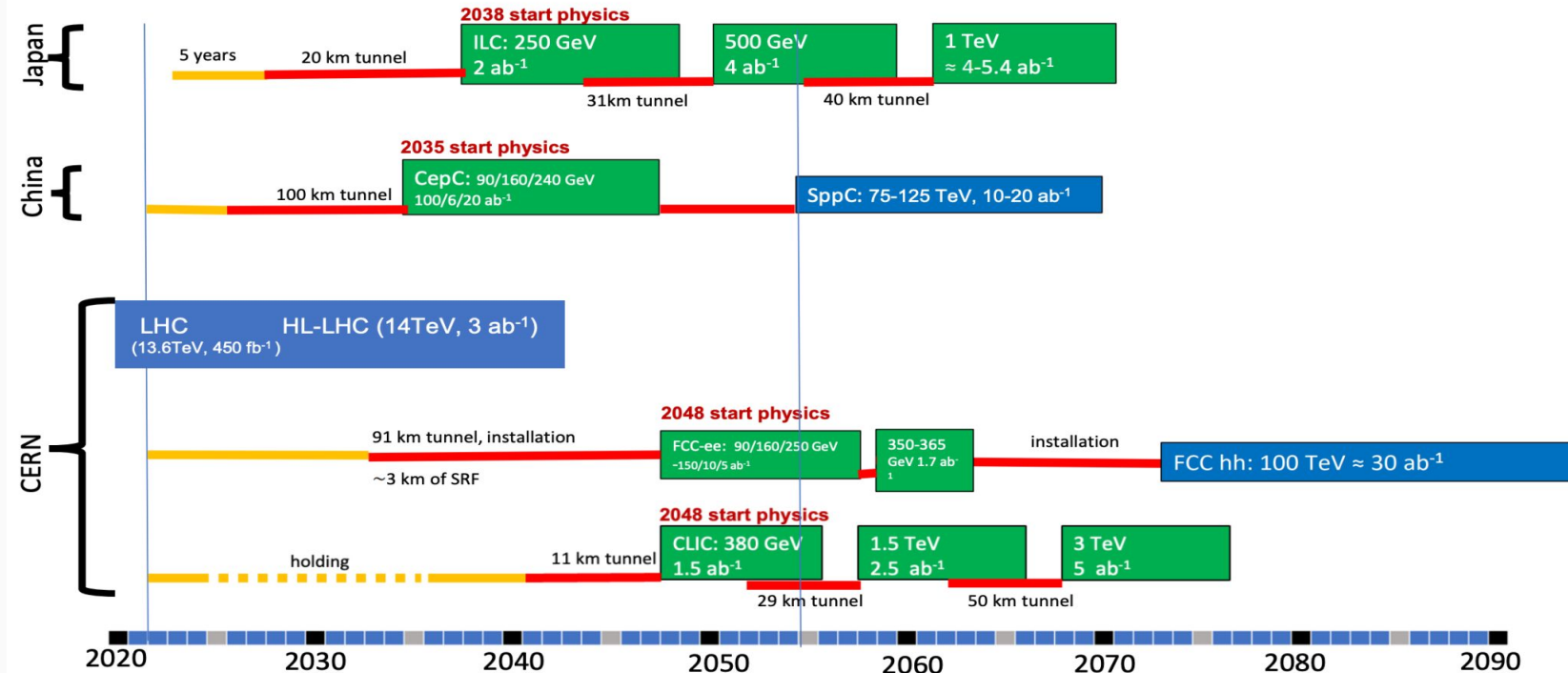
**Targeting ~3000 fb⁻¹ of data
or 180 million Higgs bosons**

Physics goals: SM Higgs boson properties and EWSB, BSM Higgs, EW multiboson production, SM parameters ($\sin^2\theta_{\text{eff}}$, $m_{W'}$, m_{top}), CKM unitarity, bottom/top FCNCs, LFV via τ decays, exotic hadronic states, improved PDF precision, Supersymmetry, Dark Matter, BSM resonances, LLPs, QGP, ...

and Beyond

Indicative scenarios of future colliders [considered by ESG]

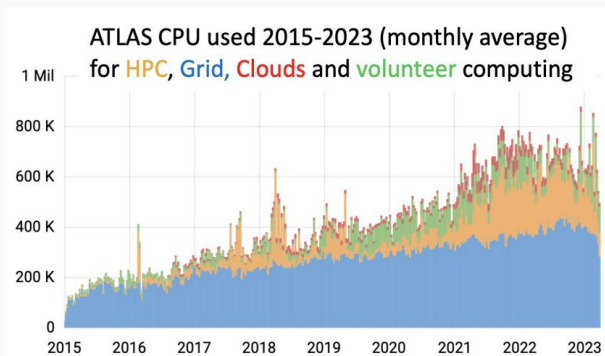
- Proton collider
- Electron collider
- Muon collider
- Construction/Transformation
- Preparation / R&D



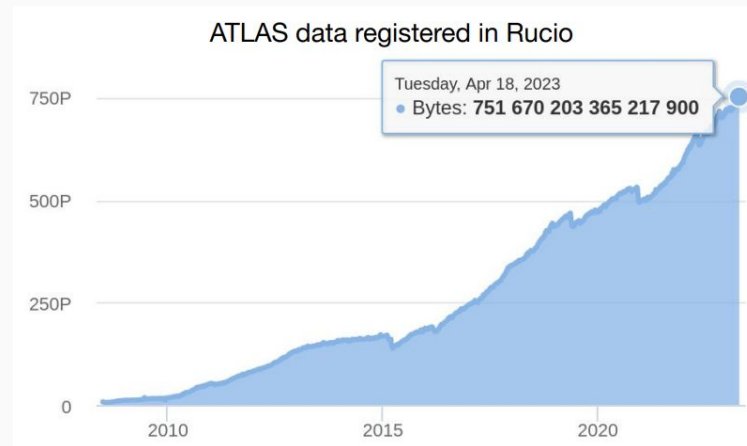
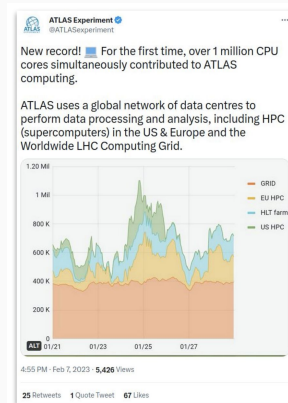
LHC Computing Resources by the ATLAS Experiment

High Energy Physics heavily invests in software

~50M lines of C++ that would cost >\$0.5B to develop commercially



[A. Klimentov - Future Data-Intensive Experiment Computing Models](#)



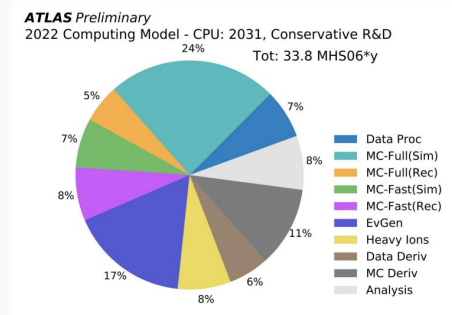
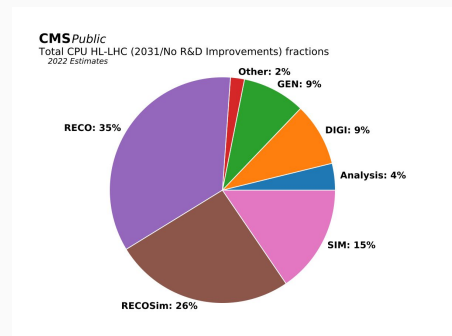
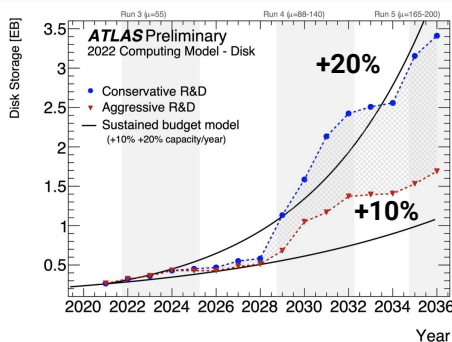
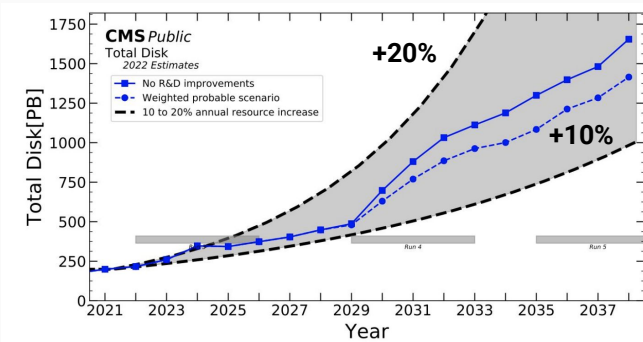
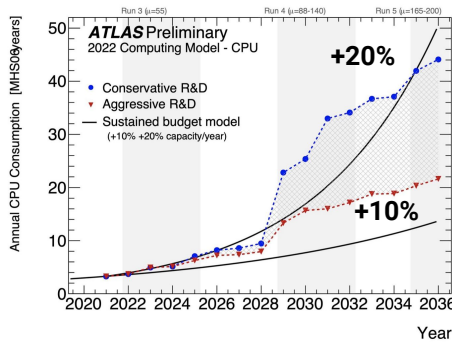
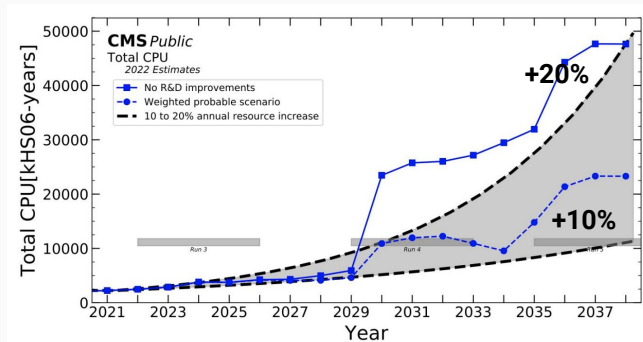
[D. South - ATLAS Distributed Computing Evolution](#)

- Record of >1M simultaneous CPU cores from supercomputers & grids
- Heterogeneity of computing resources increased dramatically after/during Run 2
- WLCG stably provides about ~50% of CPU slots

- 1B+ files, 750+ PB of data, 400+ Hz interaction
- 120 data centres, 5 HPCs, 3 clouds, 1000+ users
- 2.7 Exabytes/year uploaded & downloaded
- 1.2 Exabytes/year transferred


HL-LHC Projections

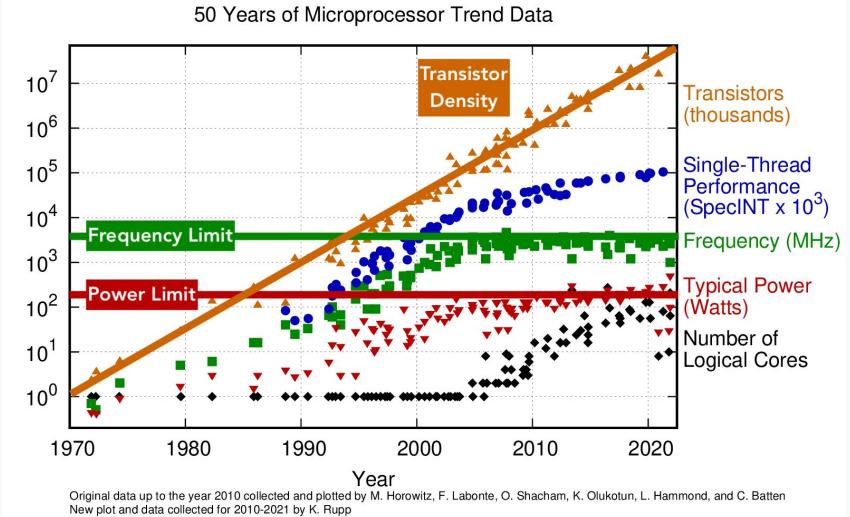
In general, resources will fall short unless significant R&D occurs



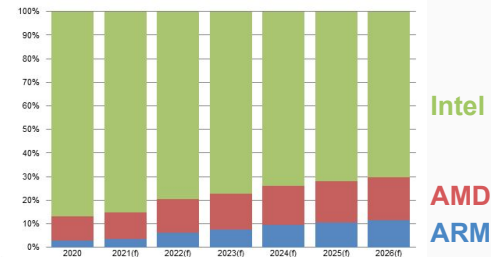
charts assume pessimistic R&D scenario

Hardware Evolution

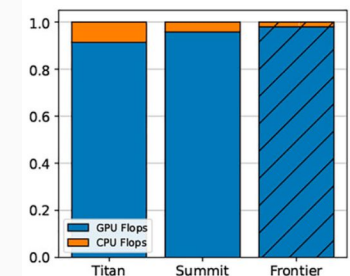
- **There is a shift in the way microprocessors deliver performance**
 - Clock frequency stalls but transistor (and core) counts increase
 - Necessitates programming model based on **concurrency** (e.g. [AthenaMT](#))
- non-x86 CPU architectures start getting a market share
 - ARM at least 30% more energy efficient on HEP workloads [1] 
 - Current **#2 HPC FUGAKU** is based on ARM
- Performance of modern systems is primarily delivered by GPUs – **significantly increasing energy efficiency**



CPU shipment share [2]



OLHFs HPC performance share [3]



Challenges are shared: LHC experiments, Belle II, DUNE, ...

- “Software upgrade” is needed to run in parallel with the hardware upgrades
 - Developing and deploying *sustainable software* is both a technical and a social challenge
-
- The [HEP Software Foundation](#) was established in 2014 in an attempt to encourage the community to work together in an efficient and effective way
 - Facilitate **coordination and common efforts** in HEP software and computing **internationally** and to provide **ground to set goals** and priorities for the future
 - Various working groups were established and compiled [A Roadmap for HEP Software and Computing R&D for the 2020s](#) signed by >300 authors from 124 institutes

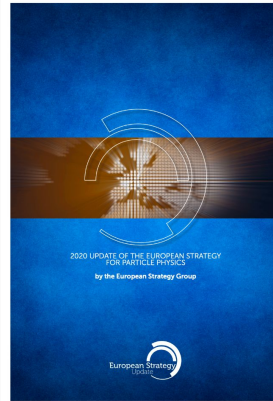


EU & US Strategies

4

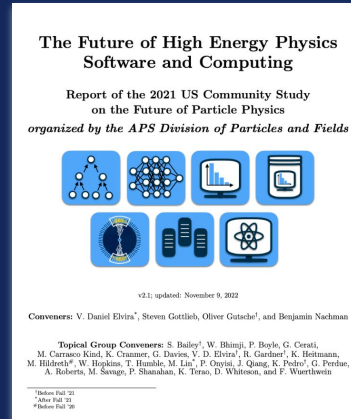


Other essential scientific activities for particle physics



[10.17181/CERN.JSC6.W89E](https://cds.cern.ch/record/10.17181/CERN.JSC6.W89E)

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



[arXiv:2210.05822](https://arxiv.org/abs/2210.05822)

“Computing is a global endeavor and addressing the above items should include coordination with worldwide partners”

1. Long-term development, maintenance, and user support of essential software packages with targeted investment.
2. R&D efforts cutting across project or discipline boundaries should be supported from proof of concept to prototype to production.
3. Support for professionals to conduct code re-engineering and adaptation
4. Strong investment in career development

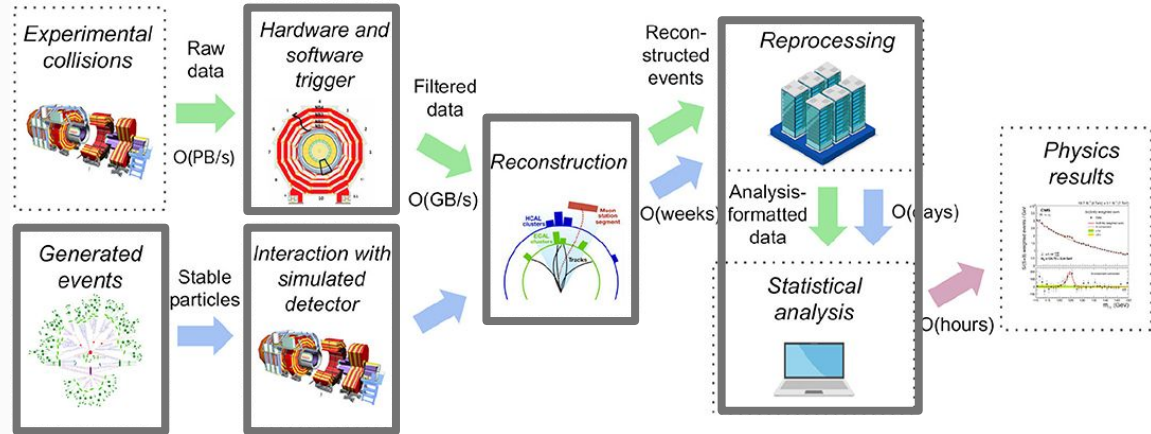
Challenges Landscape

- Unprecedented data volume
- Workflow complexity increase
- Heterogeneity of computing resources
- Physics analysis model evolution

►► re-engineering
scientific software

Computing stages in HEP

affected and discussed below



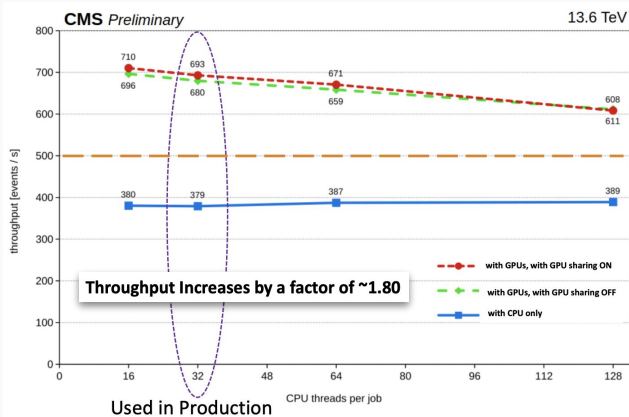
[10.3389/fdata.2021.661501](https://doi.org/10.3389/fdata.2021.661501)

Software Based Triggering

HL-LHC data challenge: Luminosity: $2 \mapsto 7.5 \cdot 10^{34} \text{ cm}^{-2}\text{s}^{-1}$, Pileup: $60 \mapsto 200$, collision rate: 40 MHz

Hardware triggers deliver a data rate of $O(0.1 - 1) \text{ MHz}$ for CMS & ATLAS

- **Heterogeneous** computing farms reconstruct events and further reduce data rate to $O(1) \text{ kHz}$
- Increased R&D on **accelerators** (GPUs and FPGAs) usage
 - Running fast-reconstruction and/or ML algorithms
- Commodity hardware reduces cost

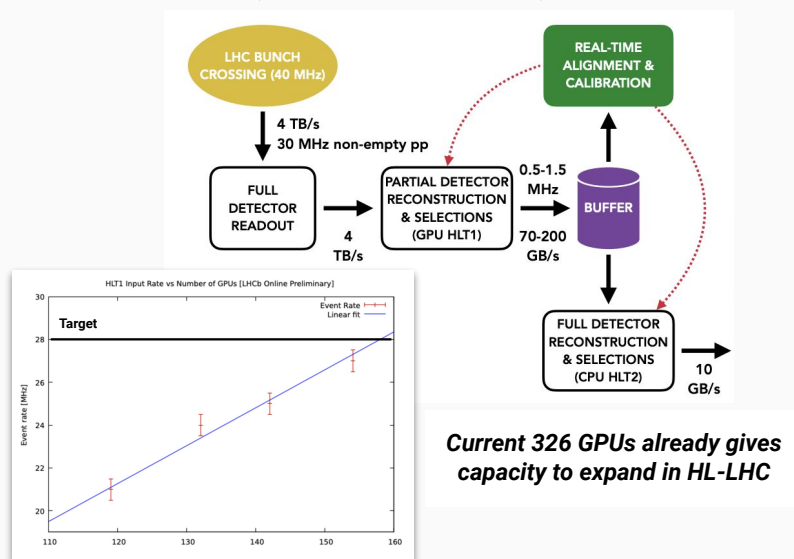


+50% more events per kWh [1]

LHCb takes an alternative approach

first introduced by ALICE [2, 3]

completely removes hardware trigger and reconstruct events on **GPUs** (+ R&D on FPGAs [4]) at collision rate



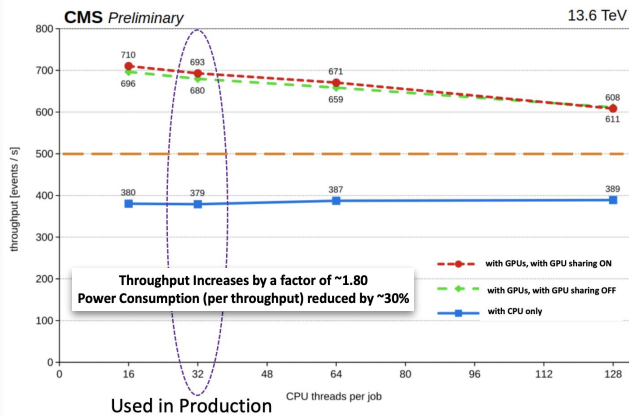
Current 326 GPUs already gives capacity to expand in HL-LHC

Software Based Triggering

HL-LHC data challenge: Luminosity: $2 \mapsto 7.5 \cdot 10^{34} \text{ cm}^{-2}\text{s}^{-1}$, Pileup: $60 \mapsto 200$, collision rate: 40 MHz

Hardware triggers reduce the data rate to $O(0.1 - 1)$ MHz for CMS & ATLAS

- **Heterogeneous** computing farms reconstruct events and further reduce data rate to $O(1)$ kHz
- Increased R&D on **accelerators** (GPUs and FPGAs) usage
 - Running fast-reconstruction and/or ML algorithms
- Commodity hardware reduces cost



+50% more events per kWh [1]

LHCb takes an alternative approach

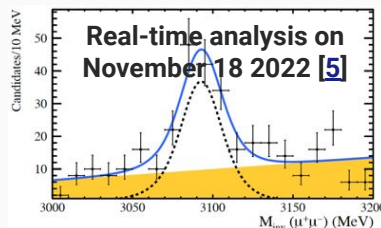
first introduced by ALICE [2, 3]

completely removes hardware trigger and reconstruct events on **GPUs** (+ R&D on FPGAs [4]) at collision rate

Bonus: Real-time analysis

Data rate \sim event size * event rate

compress the raw detector data in real-time



- Requires real-time alignment and calibration
- Only the data of the signal candidate are saved – rest of event data are discarded
- Flexibility on that selection – both the most interesting events and the most interesting pieces of each event

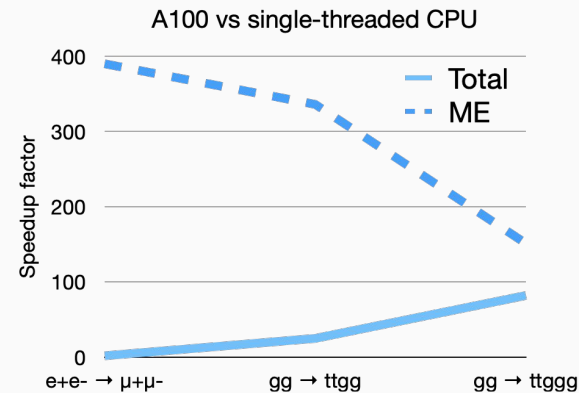
[M. Vesterinen - Real-time analysis in Run 3 with the LHCb experiment](#)

Event Generation

Event generation is projected to use 10-20% of the HL-LHC computing resources

HSF Physics Event Generator WG major R&D focus [1, 2]:

- Gain a detailed understanding of current CPU costs
- Understand the best way to move to GPUs and vectorized implementations
 - ME computation on GPUs and SIMD CPUs
- Optimize phase space sampling and integration algorithms, including the usage of ML
 - [MadJax](#) provides differentiable per-event MEs
 - [Active Learning](#) to sample critical regions of the phase space
- Reduce the cost associated with negative weight events, using new theoretical or experimental approaches
 - New Sherpa configuration by ATLAS achieves 50% reduction of negative weights [[J. High Energ. Phys. 2022, 89 \(2022\)](#)]
- Promote relevant collaboration, training, funding and career opportunities



[SYCL](#) to employ various HPC vendors/architectures

gg → t \bar{t} gg	MEs precision	madevent		
		t _{TOT} = t _{Mad} + t _{MEs} [sec]	N _{events} /t _{TOT} [events/sec]	N _{events} /t _{MEs} [MEs/sec]
Fortran(scalar)	double	37.3 = 1.7 + 35.6	2.20E3 (=1.0)	2.30E3 (=1.0)
C++/none(scalar)	double	37.8 = 1.7 + 36.0	2.17E3 (x1.0)	2.28E3 (x1.0)
C++/sse4(128-bit)	double	19.4 = 1.7 + 17.8	4.22E3 (x1.9)	4.62E3 (x2.0)
C++/avx2(256-bit)	double	9.5 = 1.7 + 7.8	8.63E3 (x3.9)	1.05E4 (x4.6)
C++/512y(256-bit)	double	8.9 = 1.8 + 7.1	9.29E3 (x4.2)	1.16E4 (x5.0)
C++/512z(512-bit)	double	6.1 = 1.8 + 4.3	1.35E4 (x6.1)	1.91E4 (x8.3)

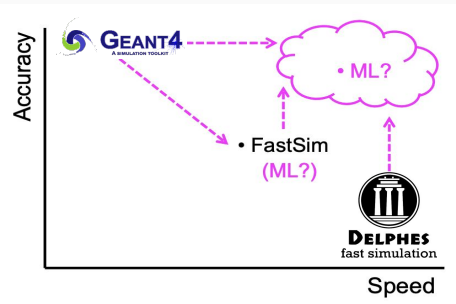
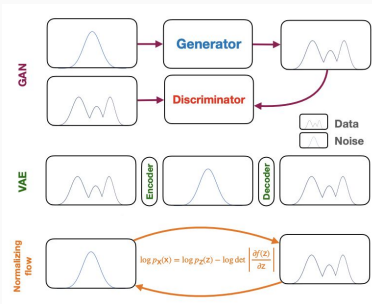
S. Hageboeck - Madgraph5_aMC@NLO on GPUs and vector CPUs



Detector Simulation (Fast)

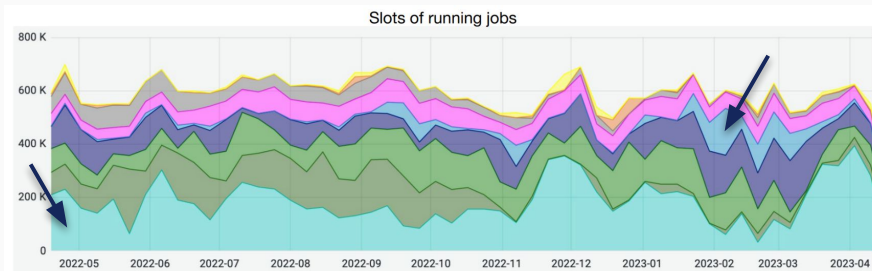
Detector simulation is the largest CPU consumer at the moment (WLCG & HPCs) and needs are only going to increase

LHC experiments try to alleviate using **fast simulation techniques** primarily for calorimeter simulation



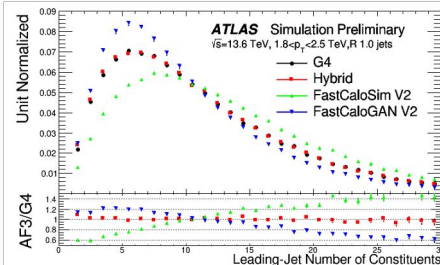
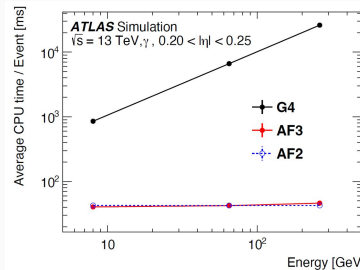
[arXiv: 2203.08806](https://arxiv.org/abs/2203.08806)

- Fast simulation methods are traditionally based on parameterizations of detector response
- Last years there is an explosion of (generative) ML techniques [1]:
 - a. Replace or augment (part or all of) Geant4
 - b. Replace or augment (part or all of) FastSim



[D. South - ATLAS Distributed Computing Evolution](#)

Probably a **hybrid à la divide-and-conquer** approach will pave the way to HL-LHC



[M. Faucci Giannelli - The AtFast3, the ATLAS fast simulation tool in Run3](#)

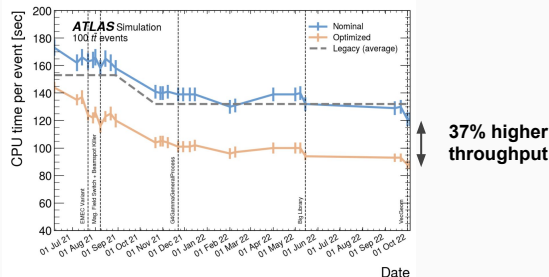
Detector Simulation (Full)

Highly accurate simulations are still needed

1. Detector performance and calibration studies
2. Reference for training fast-simulation algorithms on

Two main avenues to reach HL-LHC demands

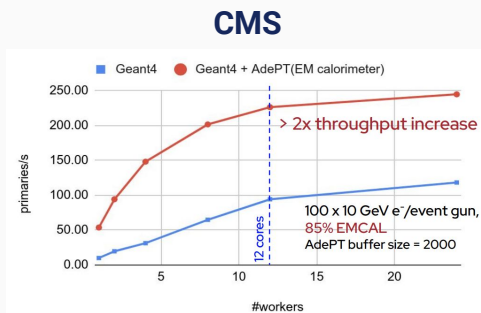
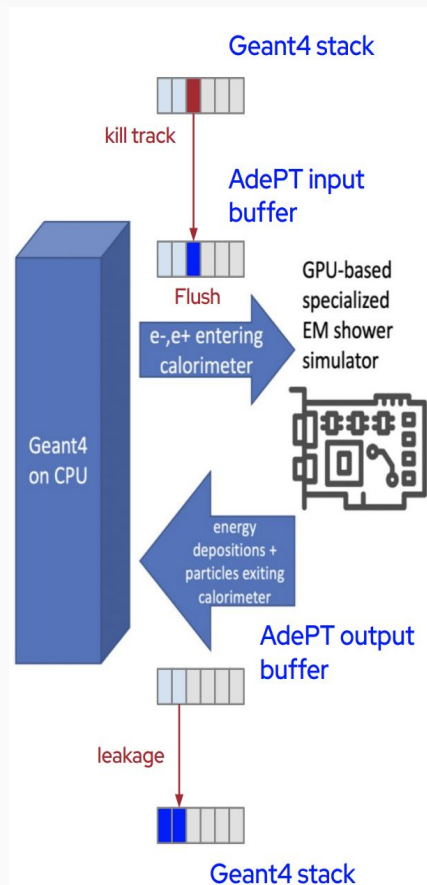
A. Geant4 Optimization



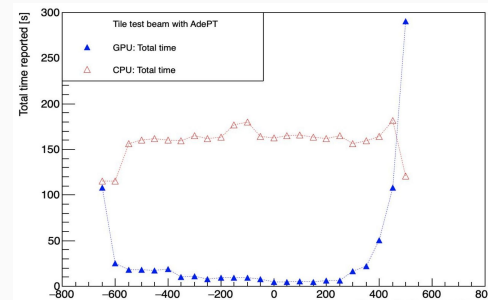
[V. Kourlitis - Optimizing the ATLAS Geant4 detector simulation software](#)

B. GPUs Utilization

- Two R&D projects developing prototypes focusing on EM Physics:
 - [AdePT](#) (CERN)
 - [Celeritas](#) (DOE)
- Knowledge exchange & [community workshop](#)



ATLAS TileCal (test-beam)



[A. Gheata - Accelerated demonstrator of electromagnetic Particle Transport \(AdePT\) status and plans](#)

- Traditionally, reconstruction software is very experiment-dependent
 - detector geometry, calibrations, etc.
- Recent hardware evolution dictate common solutions to tackle the HL-LHC challenges
- [A Common Tracking Software Project](#) (ACTS) develops an **experiment-independent set of track reconstruction tools**

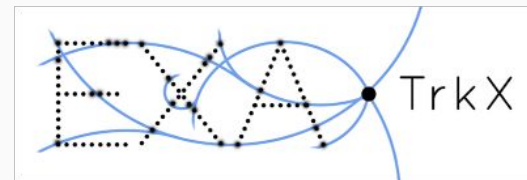
tools

- Provide high-level modules that can be used for any tracking detector
- Highly performant, yet largely customizable implementations
- Users from LHC, FCC, SPHENIX, EIC, ...

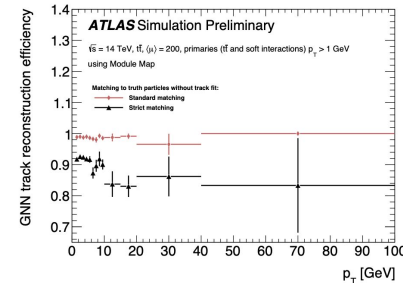
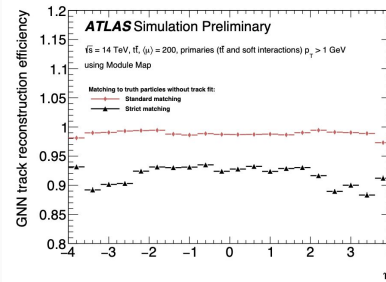
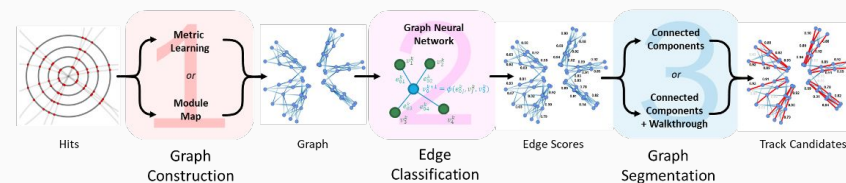
Key features:

1. Tracking geometry description
2. Simple event data model
3. Common algorithms for:
 - Track propagation and fitting
 - Seed finding
 - Vertexing

Testbed for HEP Geometric Deep Learning



ATLAS GNN4ITK

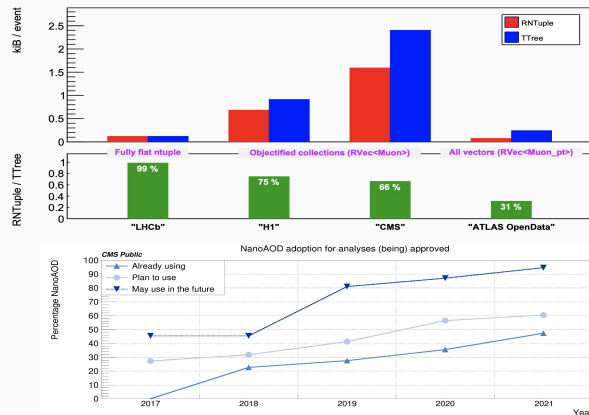


Track candidate not matched to any particle = fake track
 found to be $O(10^{-3})$

Data Analysis

Analysing the vast data of HL-LHC is it on itself a challenge as current workflows do not scale

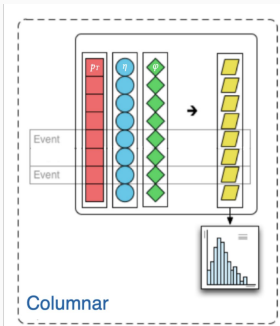
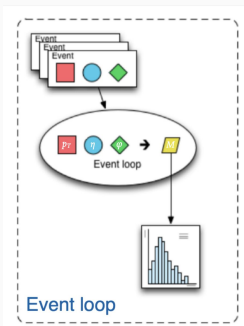
- Disk/Tape are limited/expensive and analysis data are projected to occupy ~30% (ATLAS model). **Adoption is needed.**
 - Evolved encodings: ROOT's [RNTuple](#)
 - Reduce data copies and intermediate formats
 - More compact data formats (e.g. NanoAOD and [PHYSLITE](#))
- **Target is ~10kB per event for super-fast lightweight analysis**



[J. Blomer - ROOT's RNTuple I/O Subsystem: The Path to Production](#)

[CMS Offline and Computing Public Results](#)

Analysis workflow evolution: **Columnar Analysis** *array programming solution for quick insights delivery*



Bringing HEP analysis closer to Python ecosystem and industrial standards!



Declarative analysis description language coupled with dask task graphs

```
# "array" operations only describe what is to be done
cut = (events.MET.pt < 100.) & (events.Electron.pt > 30.)
hist.fill(eta=events.Electron.eta[cut].flatten())
# in order to render a result, we ask for it
hist.compute()
```

lazy evaluation on arbitrary compute resources

[L. Gray - Fine-Grained HEP Analysis Task Graph Optimization with Coffea and Dask](#)

Community Training

Need for a **unified, scalable and sustainable** software training framework powered by the entire community

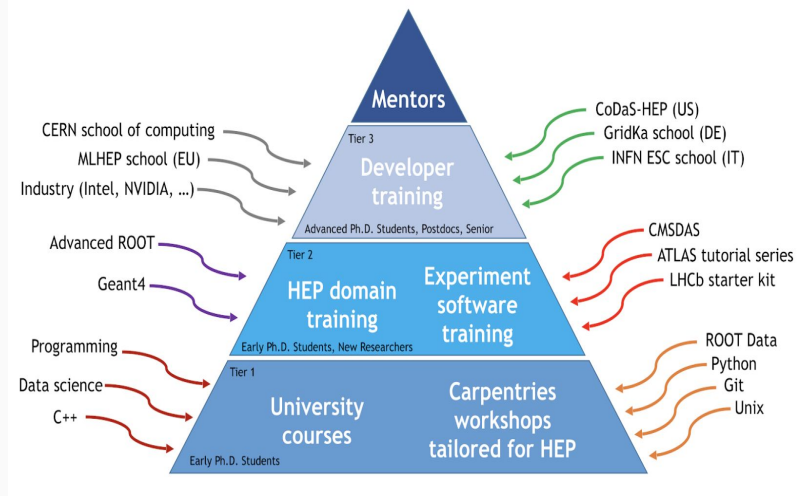
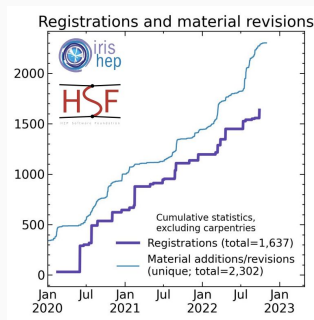
HSF [Training Working Group](#) (along with [IRIS-HEP](#)) is building a community around these principles

A unified [Training Center](#) for HEP currently lists 21 training modules embracing [The Carpentries](#) framework

Workshops and training courses annually

Highlights:

1. Software Carpentry
2. C++ Training Courses



Conclusions



**While preparing to collect the unprecedented data of the HL-LHC,
the computing landscape undergoes significant evolution**

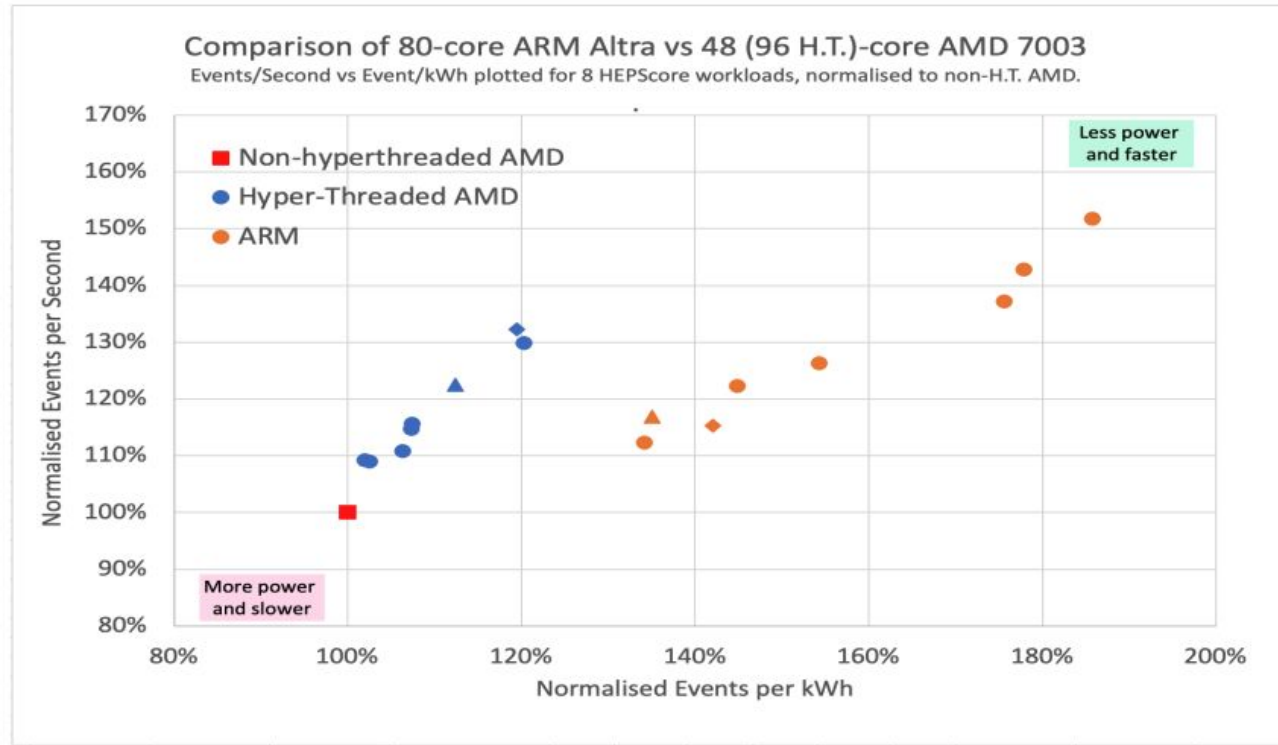
Common, coordinated effort is necessary across the community for a *software upgrade* to occur

The [HEP Software Foundation](#) plays a vital role in fostering scientific computing collaborations to address specific challenges

Join the several [forums](#) and [newsletters](#)

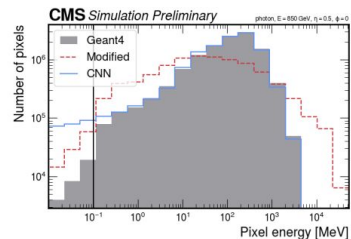
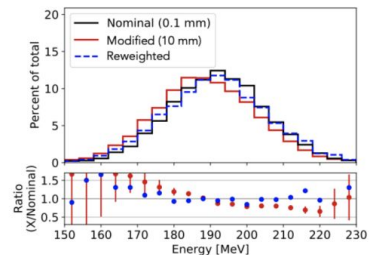
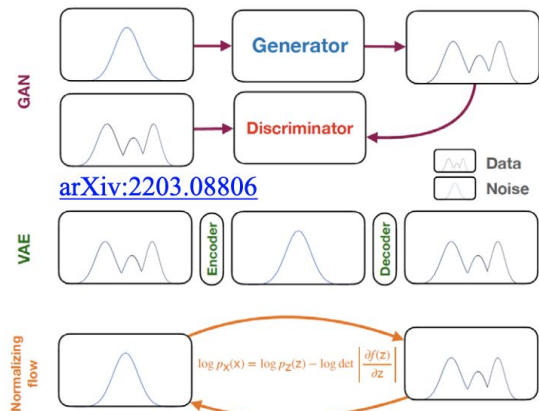
backup

ARM_64 vs AMD(x86) results



Taxonomy

- Generative models (“replace”):
 - Usually *stochastic*
 - Generative Adversarial Networks (GANs)
 - Variational Autoencoders (VAEs)
 - Normalizing Flows (NFs)
- Refinement techniques (“augment”):
 - Usually *deterministic*
 - Classification-based (reweighting)
 - Regression-based (correcting)



PHYSLITE by numbers

File sizes ([kB per event], using the current Run 3 prototype):

Actual size	Run 2 MC $t\bar{t}$	Run 3 MC $t\bar{t}$	data17	Target size	MC	Data
PHYS	34.2	40.7	21.7	PHYS	50	30
PHYSLITE	13.6	16.3	6.2	PHYSLITE	12	10

Work is ongoing to further reduce PHYSLITE size

Estimate for the total Run 3 dataset, assuming targeted PHYS/LITE sizes are met:

	#Events [10^9]	Size [kB/event]	#Replicas	#Versions	Total [PB]
PHYS Data	19	30	4	2	4.6
PHYS MC	24	50	4	2	9.6
PHYSLITE Data	19	10	4	2	1.5
PHYSLITE MC	24	12	4	2	2.3

Collections

(create task graphs)

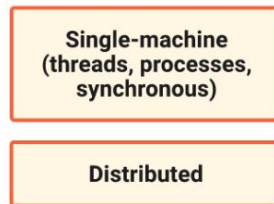
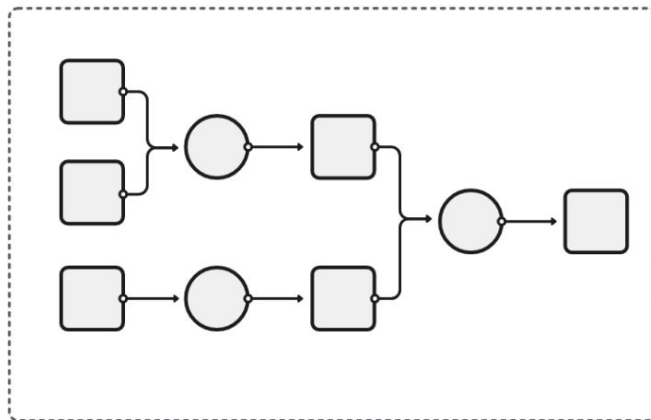


Task Graph



Schedulers

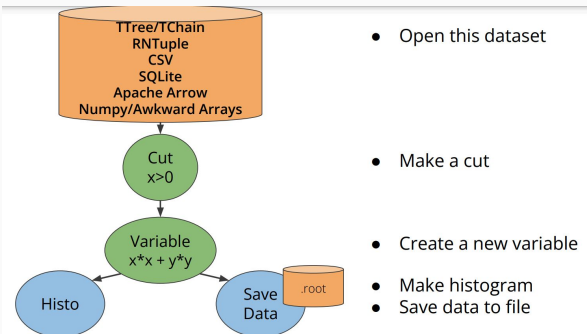
(execute task graphs)



- Dask provides an interface for specifying/locating input data and then describing manipulations on that data are organized into a task graph
 - This task graph can then be executed on local compute or on a cluster
- Dask Array and Dask Dataframe deal well with rectangular data
 - Provide a scalable interface to describe manipulations of data that may not fit into system memory by mapping transformations onto partitions of the data that fit in memory

Declarative interface for data analysis

current R&D [1]



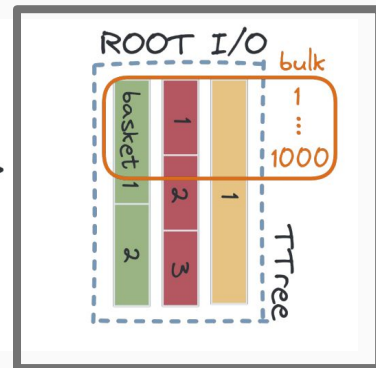
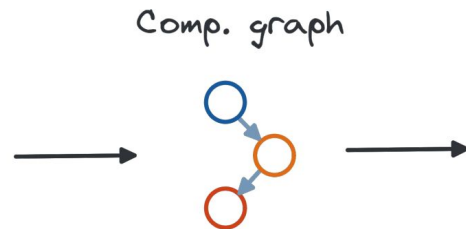
- Open this dataset
- Make a cut
- Create a new variable
- Make histogram
- Save data to file

```
df = ROOT.RDataFrame(dataset)

df = df.Filter("x > 0")\
        .Define("r2", "x*x + y*y")

rHist = df.Histo1D("r2")

df.Snapshot("newtree", "out.root")
```



TTree → RDataFrame → Numpy → RDataFrame → TTree

```
df = ROOT.RDataFrame("Events", "file.root")
np_arrs = df.Filter("x > 10").AsNumpy(["x", "y"])
data = {"x": np.array(...), "y": np.array(...), ...}
df = ROOT.RDF.FromNumpy(data)
```

Awkward arrays → RDataFrame → Awkward Arrays

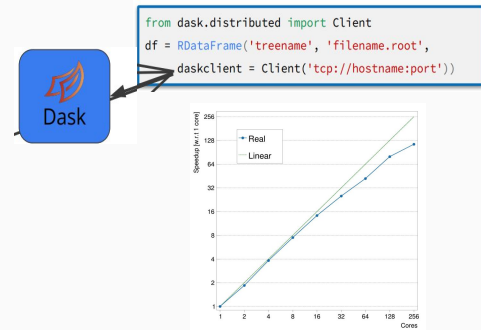
```
df = ak.to_rdataframe(
    {"x": ak.Array(), "y": ak.Array(), "z": ak.Array()})
array = ak.from_rdataframe(
    df, columns=("Dimuon_mass"))
```

```
nominal_hx = df.Vary("pt", "RVecD[pt*0.9, pt*1.1]", ["down", "up"])
               .Filter("pt > k")
               .Define("x", someFunc, ["pt"])
               .Histo1D("x")
)
hx = ROOT.RDF.VariationsFor(nominal_hx)
hx["nominal"].Draw()
hx["pt:down"].Draw("SAME")
```

proceed as usual, as if working with nominal values only

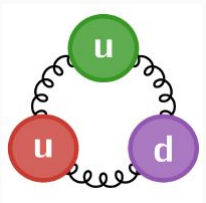
obtain all variations

Quantities automatically propagate to selections, derived quantities and results. Only needed quantities are re-computed, all in one event loop.



Julia & HEP = <3

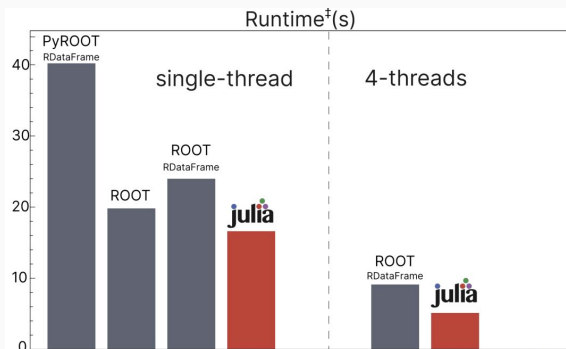
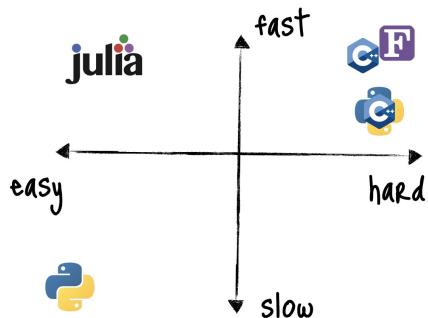
JuliaHEP



"Julia: A language that walks like Python, runs like C"

-- K. S. Kuppusamy

```
using UnROOT *not code from benchmark
tree = LazyTree("./data.root", "Events")
for evt in tree
    muon_HT = sum(evt.Muon_pt)
    if muon_HT < 200
        continue
    end
    #...
end
```



UpROOT.jl



Les Houches Event File Format



LHE.jl

LCIO



LCIO.jl



Arrow.jl



HDF5.jl

- An honorable mention for a **fully wrapped** HEP software
- **Geant4.jl** (fully wrapped using CxxWrap.jl)

