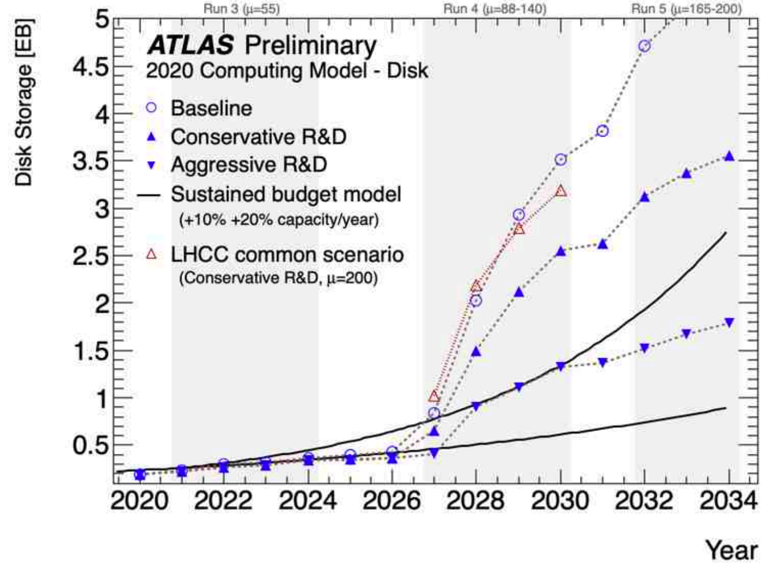
User Analysis for HL-LHC

Lukas Heinrich, CERN

Analysis at the HL-LHC scale will be challenging
- **firmly in the PB-regime for analysis**



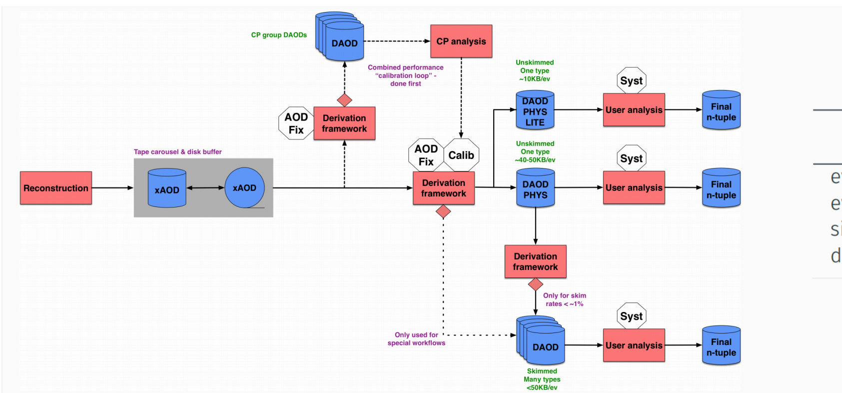
Need to ensure that analyzers have the tools and infrastructure in place to:

- **access data quickly**
- **deploy and use latest analysis methods**
- **extract the most physics of the HL-LHC dataset**

General Trend:
Infrastructure should become smarter to handle the complexity on behalf of the user.

Data Reduction, do more physics centrally

- reduce tasks for analyzers
- reduce amount of data required for end-user analysis
  → small data formats for HL-LHC O(10kb/event)



	MC			Data			Sum
	AOD	DAOD	DAOD PHYSLITE	AOD	DAOD	DAOD PHYSLITE	
events (25-28)	$6.4 \cdot 10^{11}$			$1.5 \cdot 10^{11}$			
events / year	$2.13 \cdot 10^{11}$	$1.07 \cdot 10^{12}$	$2.13 \cdot 10^{11}$	$5.0 \cdot 10^{10}$	$2.5 \cdot 10^{11}$	$5.0 \cdot 10^{10}$	
size/event [kB]	1000	100	10	700	50	10	
disk [PB/year]	213.3	106.7	2.1	35.0	12.5	0.5	369.6

[slides]

Trends:

- push object preparation / calibration upstream
- push analysis itself upstream
  (cf. real time analysis / analysis train efforts by LHC experiments)

Columnar Data Analysis / DataFrame based analysis

Arguably what we've always done, but new set of tools is emerging.

https://github.com/CoffeaTeam/coffea

Deep Scientific Ecosystem developed outside of HEP
- pandas, numpy, matplotlib, Dask, HDF5, zarr, scikit-learn

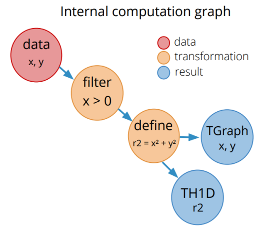
Increasing Effort to develop HEP specific toolkits based on that ecosystem (PyHEP)

uproot https://github.com/scikit-hep/uproot

Awkward Array https://github.com/scikit-hep/awkward-1.0

ROOT Ecosystem is evolving as well:
- RNtuple - the new TTree. [slides]
- RDataFrame, PyRDF

```
from ROOT import RDataFrame
df  = RDataFrame(dataset);
df2 = df.Filter("x > 0")
        .Define("r2", "x*x + y*y");
rHist = df2.Histo1D("r2");
g = df2.Graph("x","y")
```

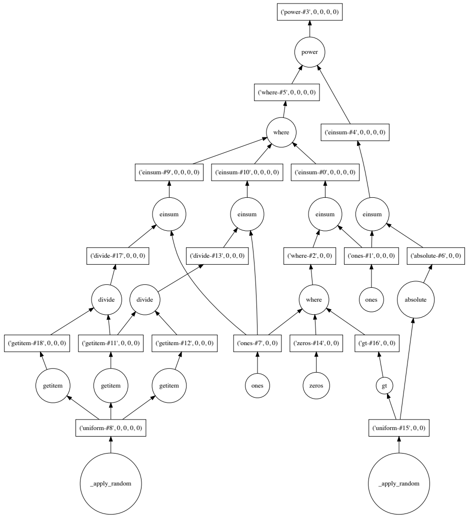


Maps well to analysis model, in which complex details (object calibration, etc..) that tend to be imperative are are pushed upstream.
→ fulfilled by small HL-LHC formats
Remaining User analysis can be highly declarative

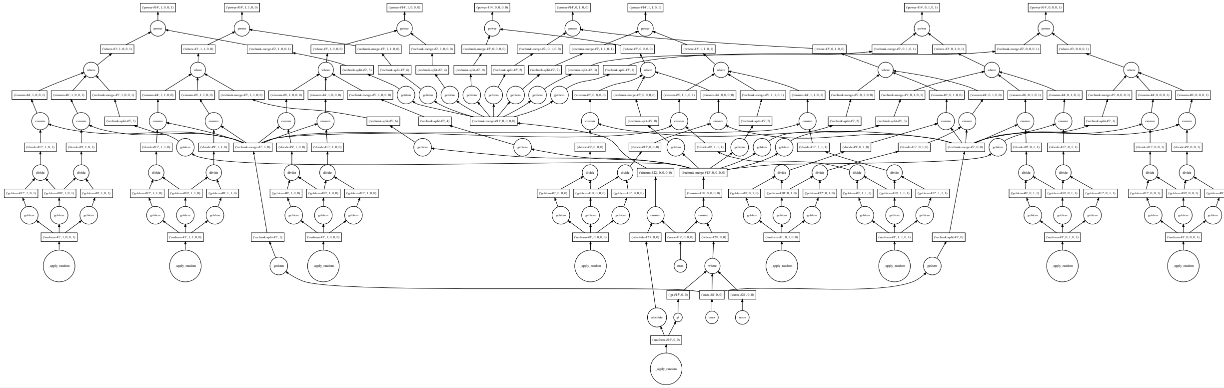
CERN

Scalable Dataframes:

Declarative Analysis: allows backend implementation to optimize, scale as it wishes. Great for distributed analysis.



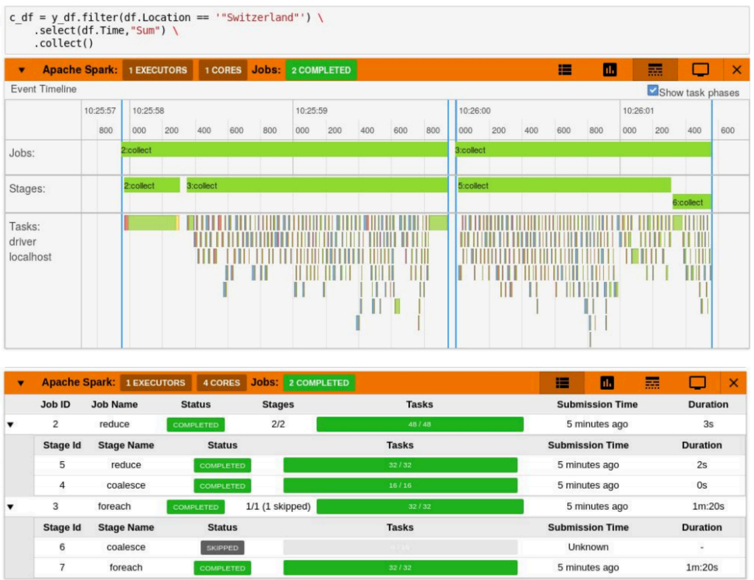
scale same "logical computation"
into two concret realizations
(depending on resources)



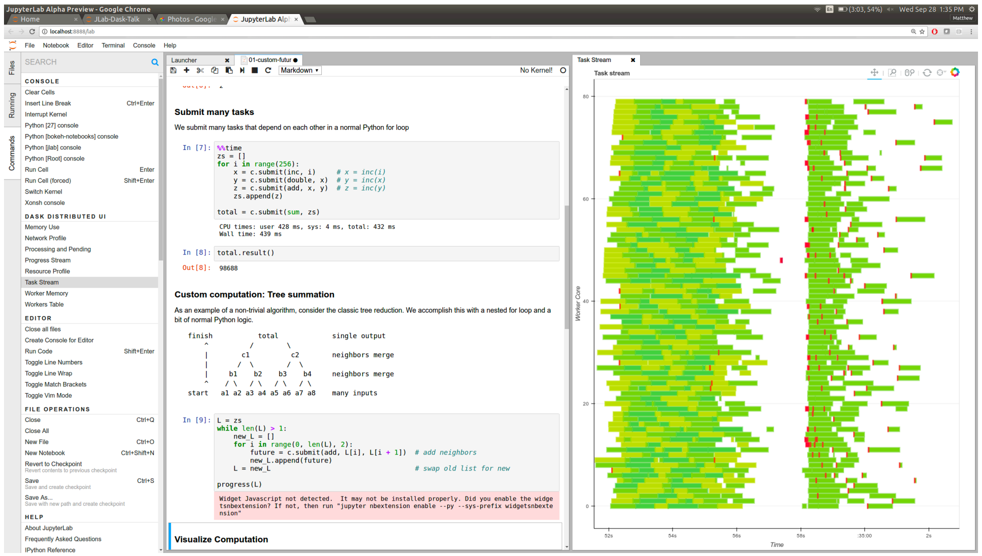
Scalable Dataframes:

Declarative Analysis: allows backend implementation to optimize, scale as it wishes. Great for distributed analysis.

PyRDF + Spark   E. Tejedor [slides]



Dask   M Rocklin [slides]



For HL-LHC, need efficient data management and delivery.

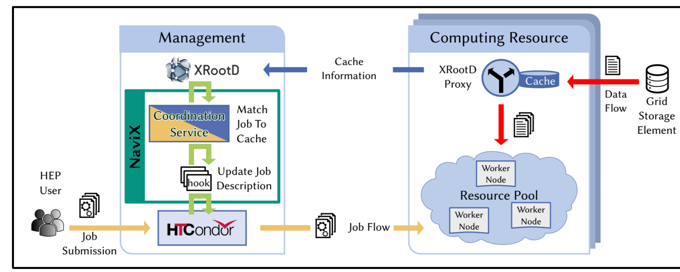
Macro-scale: Data-Lakes, optimize data placement on Tape vs Disk, load balancing: Data Carousels, etc.

- Rucio successful in growing a wide community across domains (ATLAS, CMS, BelleII, Dune, ..) most recent project: Folding@Home

RUCIO
SCIENTIFIC DATA MANAGEMENT

Micro-scale:

- Smart Data Caches within computing resources (NaviX [slides])
  - Job Placement on batch based on Local Cache Status



- Columnar Data Delivery (ServiceX [slides])
  - REST API to stream from columnar data into Kafka Topics
  - in-flight transformation C++ from OO EDM to columnar data

RUCIO
SCIENTIFIC DATA MANAGEMENT

kafka


```
{
"did": "mc15_13TeV:mc15_13TeV.361106.PowhegPythia8EvtGen_AZNLOC
"columns": "Electrons.pt(), Electrons.eta(), Muons.eta(), Muons
"image": "sslhep/servicex-transformer:v0.1",
```

UI Evolution: From Shell + Batch to Jupyter + Scale out Systems

More HEP software integrates nicely w/ notebooks

- PyHEP stack
- ROOT notebooks

TOOLBOX · 30 OCTOBER 2018

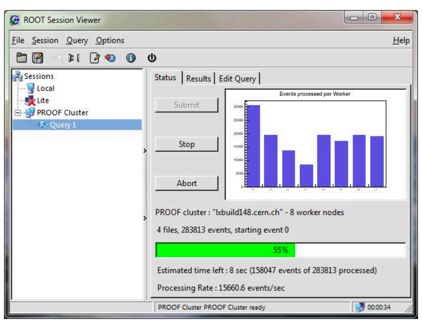
Why Jupyter is data scientists' computational notebook of choice

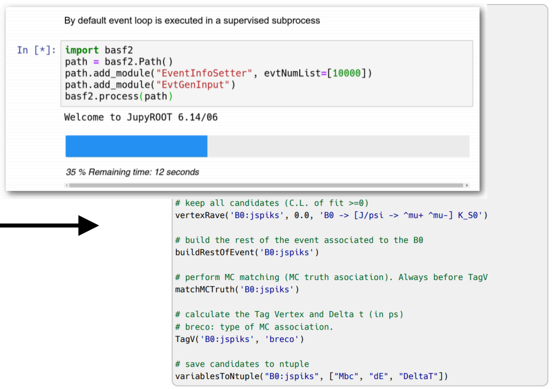
An improved architecture and enthusiastic user base are driving uptake of the open-source web tool.

Jeffrey M. Perkel

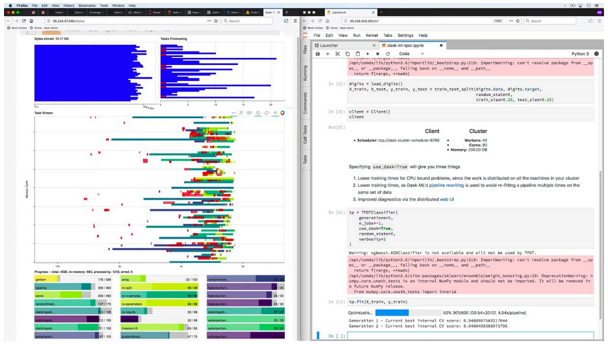
[link]

Provides widely recognizable, browser-based interactive UI, where in the past custom UIs / local applications were built



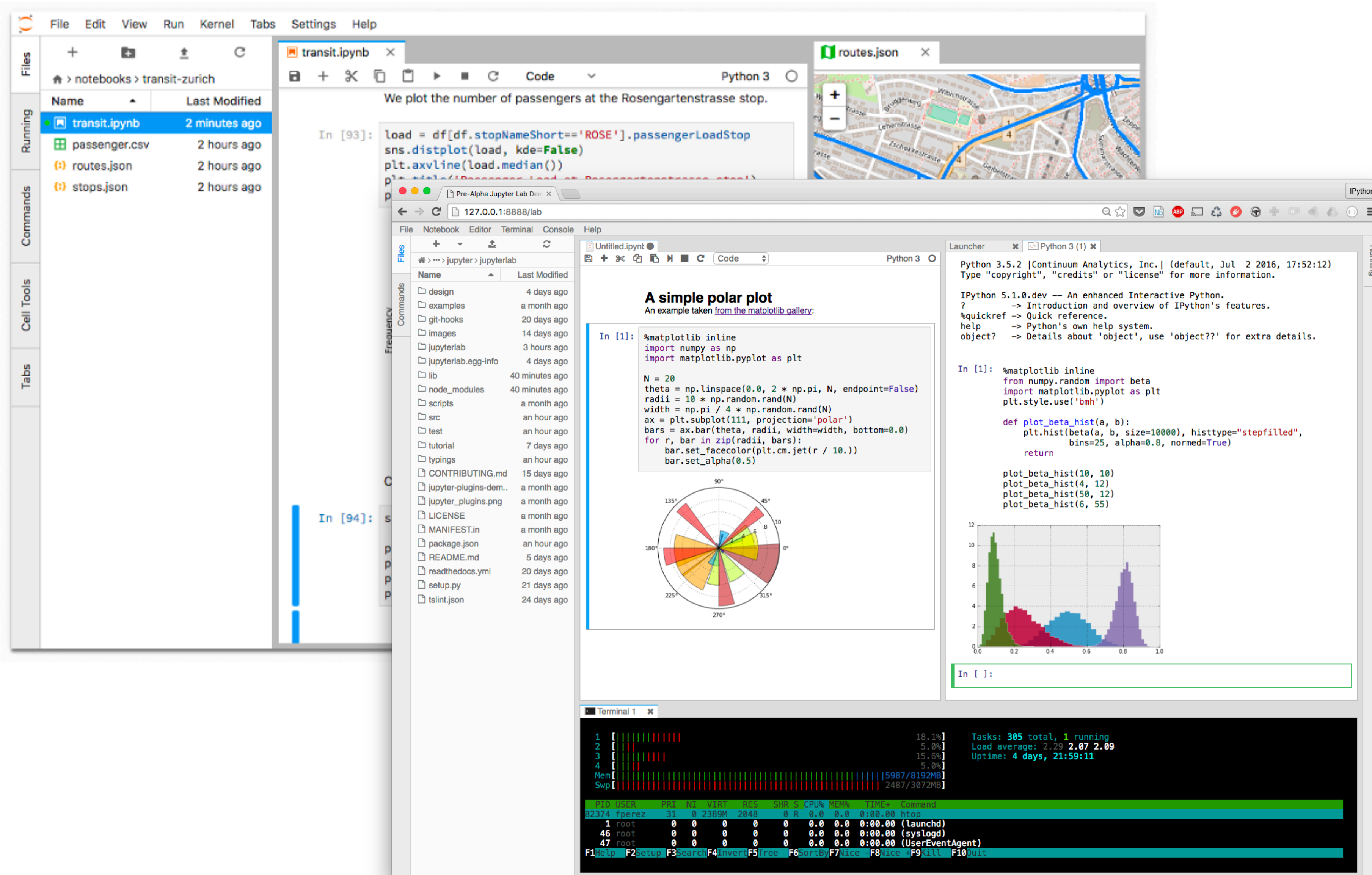


Belle II Jupyter [poster]

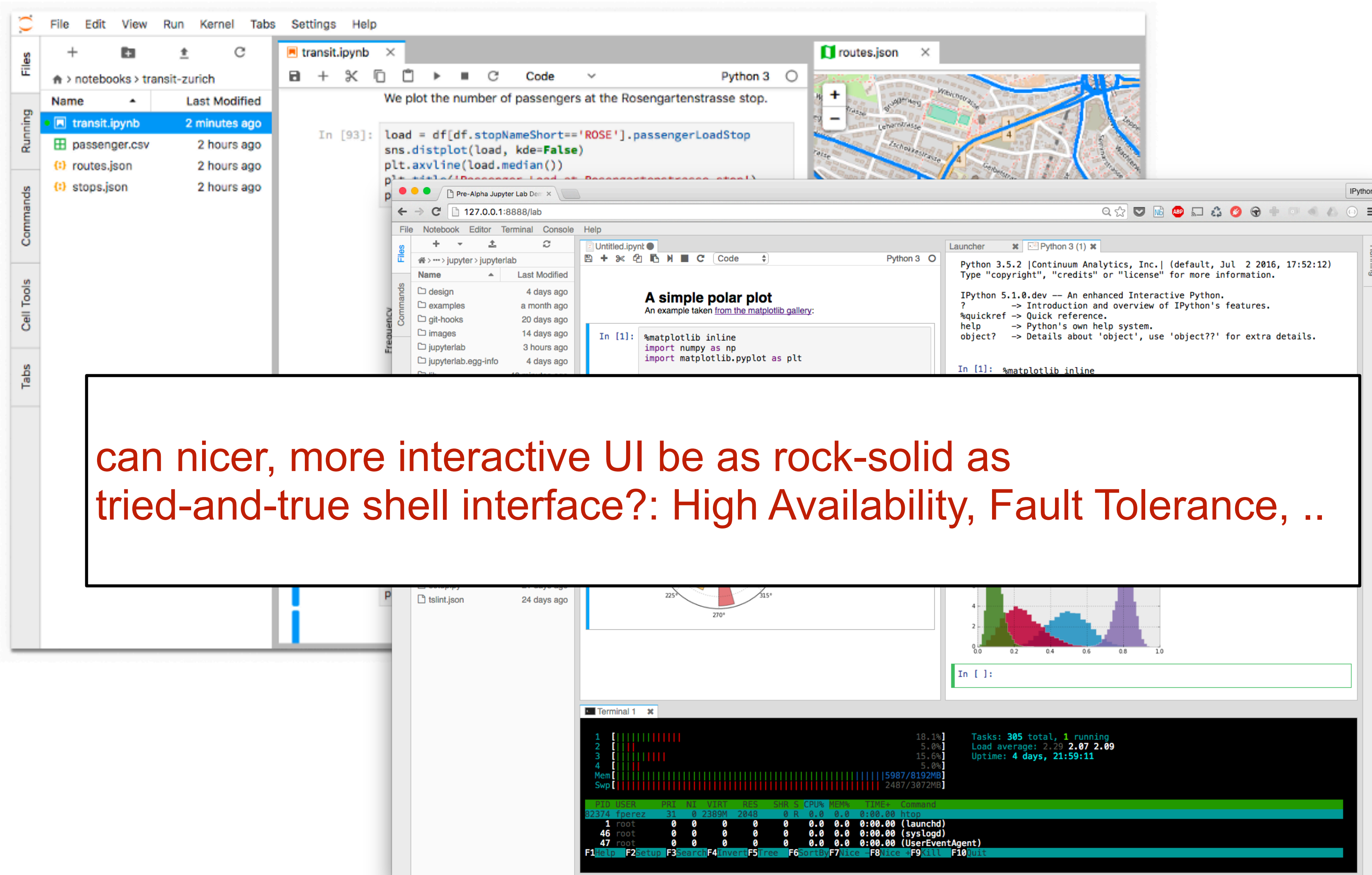


Dask in Notebooks

It's not about notebooks, many different UIs possible



It's not about notebooks, many different UIs possible

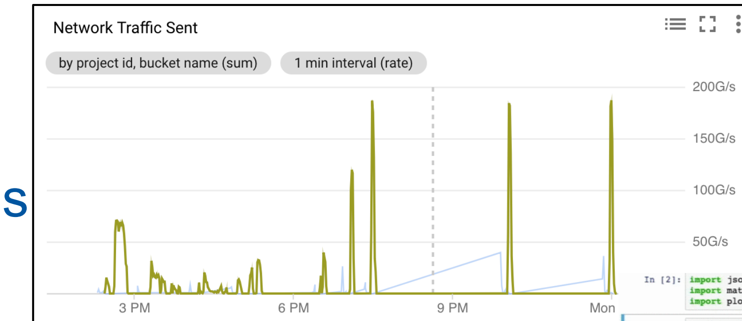


can nicer, more interactive UI be as rock-solid as
tried-and-true shell interface?: High Availability, Fault Tolerance, ..

Is interactive Analysis on PB-scale possible
It seems like it could be at the HL-LHC timeline

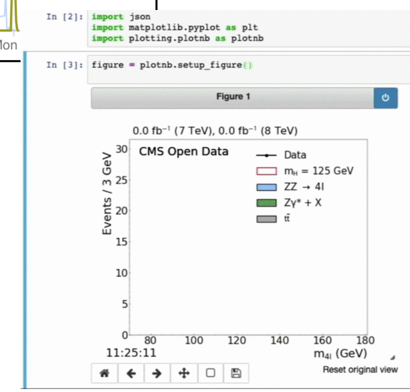
Examples:

Simplified Open Data Higgs Analysis
- 70TB in ~4 min
- old software stack (2010), but on new infrastructure (Google)
- demo designed to show scalability of cloud workflows (interactive control of 25k k8s jobs)



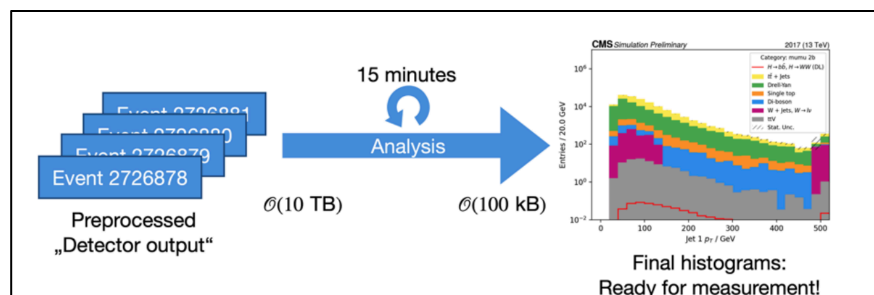
CERN openlab

Google Cloud



KubeCon 2019 Keynote [link]
R. Rocha, LH

Dataframe-based analysis
- more modern stack (coffea + dask)
- can get good throughput with much fewer resources (200 CPU)



VISPA, P Fackeldey [slides]

Analysis Facilities:

Tension:
more complex infrastructure → centralization
- **data caches**
- **interactive scale-out analysis (Notebook services, containers, ...)**
- **heterogeneous hardware**

But community relies on distributed / federated resources
- **need to scale out to where resources are**

Traditional Grid: solved for non-interactive usecase
- **common technology: batch systems + Linux + VO auth**

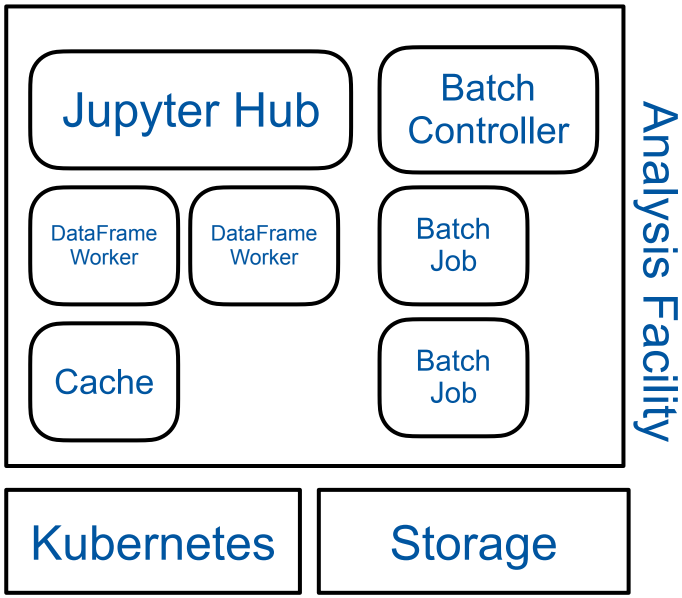
Can we do something similar for more complex interactive, distributed analysis @ HL-LHC?

Analysis Facilities:

Choice of common infrastructure substrate helps add resources on demand and development of common tools across institutions.
→ compose building blocks into analysis facilities
→ deploy transparently everywhere

Kubernetes / Cloud-Native stack suitable technology as common "substrate layer?

- native support for both interactive & batch
- notions of load balancing, fault tolerance federation built-in.



A lot of applications already target k8s natively
- Rucio (data mangement) ServiceX
- (UI frontends) JupyterHub, Binder
- out-of-core dataframes: Dask-K8s, Ray, ...
- Distributed ML (Ray, TorchElastic)
- Batch Systems, (Condor, Volcano)

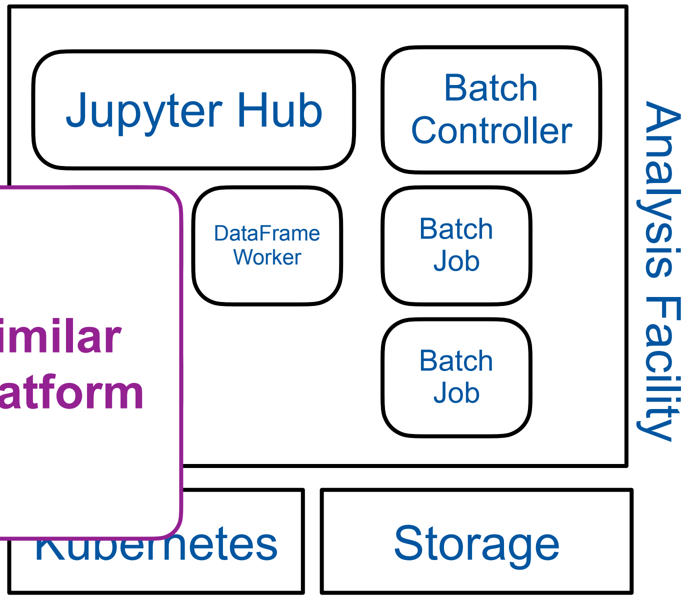
Analysis Facilities:

Choice of common infrastructure substrate helps add resources on demand and development of common tools across institutions.
→ compose building blocks into analysis facilities
→ deploy transparently everywhere

Kubernetes / Cloud-Native stack suitable technology as common "substrate layer?

- native su
- notions o
  federatio

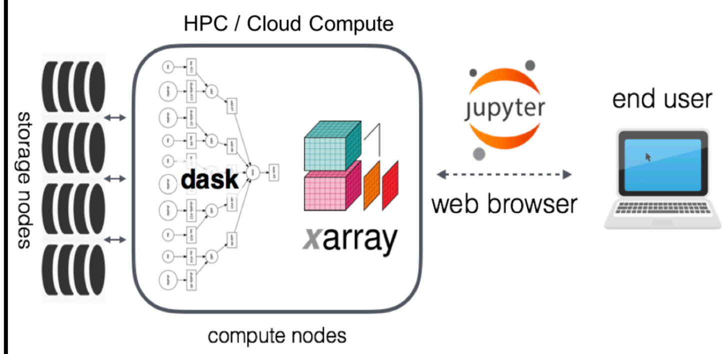
k8s: "distributed linux kernel" choosing it as common base similar to choice of using Linux x86 platform for HEP in the past.



A lot of app
- Rucio (data mangement) ServiceX
- (UI frontends) JupyterHub, Binder
- out-of-core dataframes: Dask-K8s,
- Distributed ML (Ray, TorchElastic)
- Batch Systems, (Condor, Volcano)

Examples from Geo-Science:

Pangeo: curated package of <u>existing</u> components:
- **Binder , Jupyter Hub, Dask**
- **Cloud Storage, xarray DataFrames**



A simple schematic describing how Pangeo envisions a data proximate science platform using Jupyter as the user interface.

Deployable using Kubernetes + Helm anywhere: "a portable facility"

Example from Astro:

LSST Science Platform
- **storage**
- **data catalogue**
- **Jupyter Notebooks**

(I6.1) Why is the LSST Science Platform built on Kubernetes?

Christine Banek,[1] Adam Thornton,[1] Frossie Economou,[1] Angelo Fausti,[1] K. Simon Krughoff,[1] and Jonathan Sick[1]

[1]*AURA/LSST, Tucson, AZ, USA;* cbanek@lsst.org

Abstract. LSST has chosen Kubernetes as the platform for deploying and operating the LSST Science Platform. We first present the background reasoning behind this decision, including both instrument-agnostic as well as LSST-specific requirements. We then discuss the basic principles of Kubernetes and Helm, and how they are used as the deployment base for the LSST Science Platform. Furthermore, we provide an example of how an external group may use these publicly available software resources to deploy their own instance of the LSST Science Platform, and customize it to their needs. Finally, we discuss how more astronomy software can follow these patterns to gain similar benefits.

https://arxiv.org/abs/1911.06404

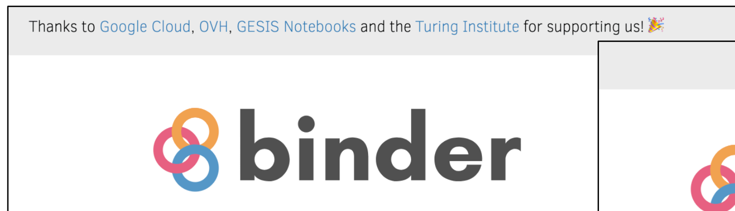
(CERN's EOS/SWAN ScienceBox roughly similar idea, less composition of existing community tools, more internal tooling)

Fedaration of Facilities:

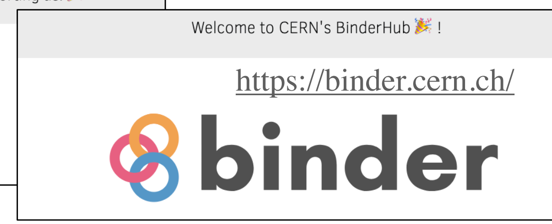
- **crucial component: similar deployments, common authentication (as with batch, user should not care too much where their notebooks run, cold optimize for data locality, etc..)**

Local Resource vs Analysis Facility not necesarily a dichotomy.

Example: Binder already fedarates across mix of commercial and academic resources



https://mybinder.org/



Resources

- common WLCG Kubernetes Working Group
- CNCF Research User Group [link]

wlcg-k8s@cern.ch

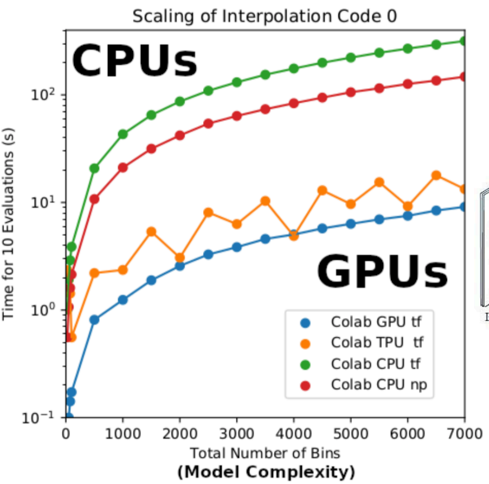
share experience with

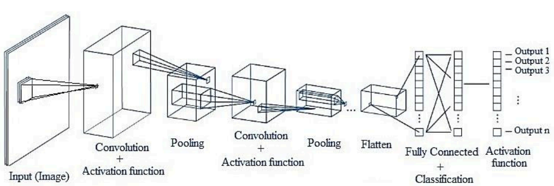
- deployment
- packaging
- cluster management

New analysis methods can benefit from GPU

- **columnar analysis and vectorized evaluation are a natural combination. Fits well w/ e.g. GPUs**
  - **true for standard HEP operations**
  - **truer for ML-based analysis**
- **But transport cost to and from hardware, requires efficient pipeline, large-batch calculation**

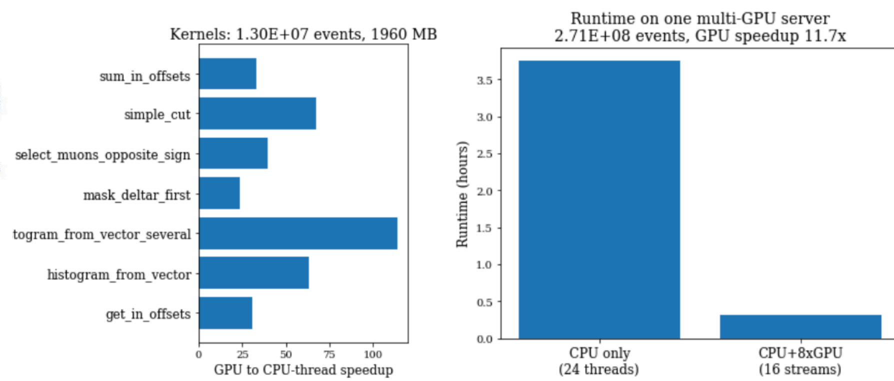
HEP Stats (HistFactory)





ML Evaluation in Analysis

HEP Event Selection (coffea + hepaccelerate)



https://github.com/pyhf

https://github.com/hepaccelerate/hepaccelerate

Differentiable Analysis Workflows:

Emerging Paradigm generalizing from Deep Learning:

neural networks → "differentiable programs"

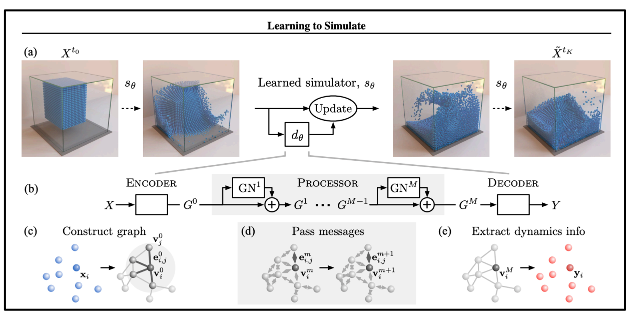
Synthesis of power of neural networks and desire to impose more structure on ML algorithms ("inductive bias")

- systematics awareness
- explainability
- inclusion of domain knowledge

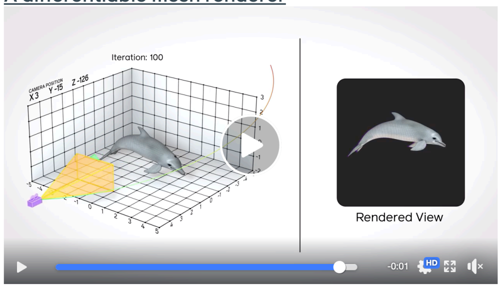
Yann LeCun · January 5, 2018 · Follow — Head of FB AI

OK, Deep Learning has outlived its usefulness as a buzz-phrase.
Deep Learning est mort. Vive Differentiable Programming!

An increasingly large number of people are defining the networks procedurally in a data-dependent way (with loops and conditionals), allowing them to change dynamically as a function of the input data fed to them. It's really very much like a regular progam, except it's parameterized, automatically differentiated, and trainable/optimizable.



A differentiable mesh renderer



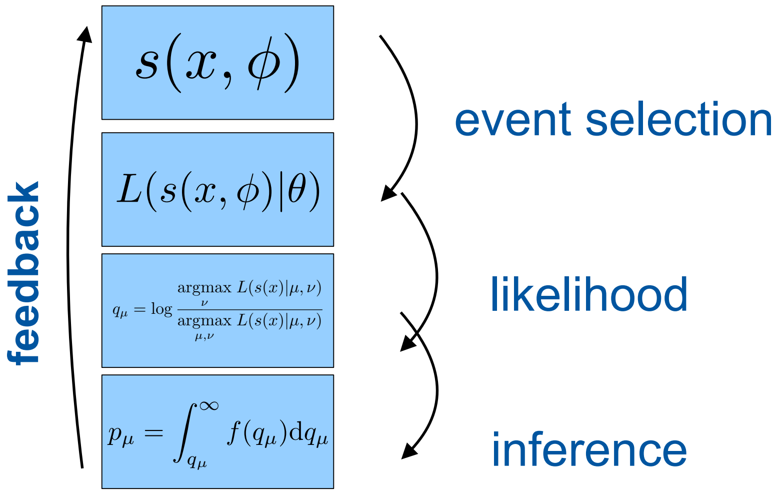
JAX, M.D.
End-to-End Differentiable, Hardware Accelerated, Molecular Dynamics in Pure Python

Samuel S. Schoenholz
Google Brain
schsam@google.com

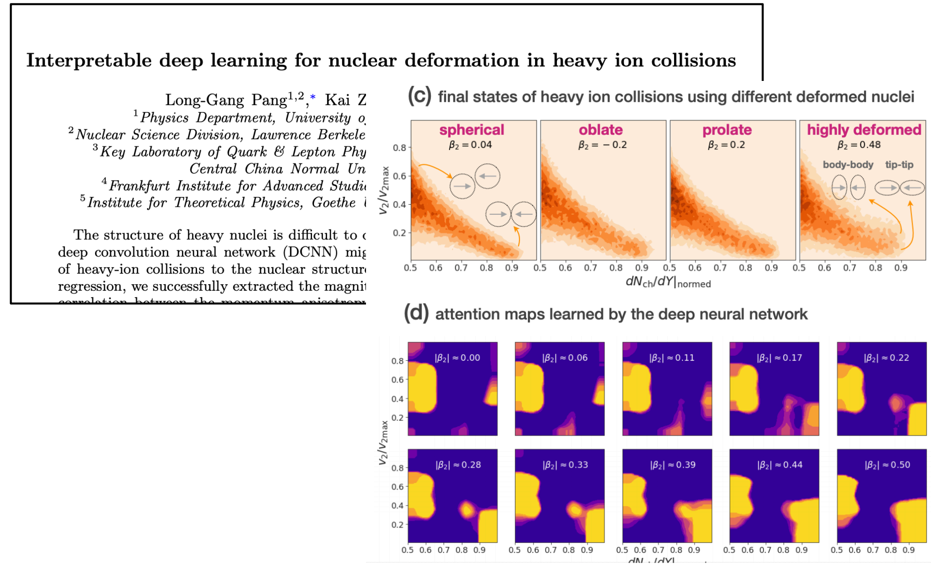
Ekin D. Cubuk
Google Brain
cubuk@google.com

Abstract

Also arriving in HEP, NP



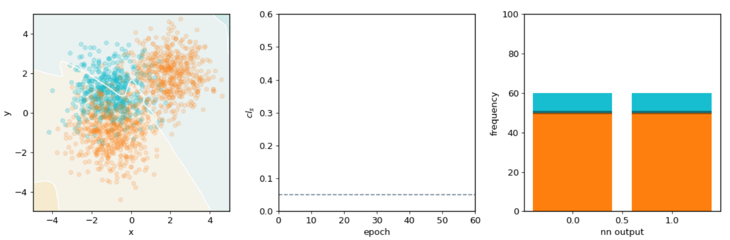
https://arxiv.org/pdf/1906.06429.pdf


Interpretable deep learning for nuclear deformation in heavy ion collisions

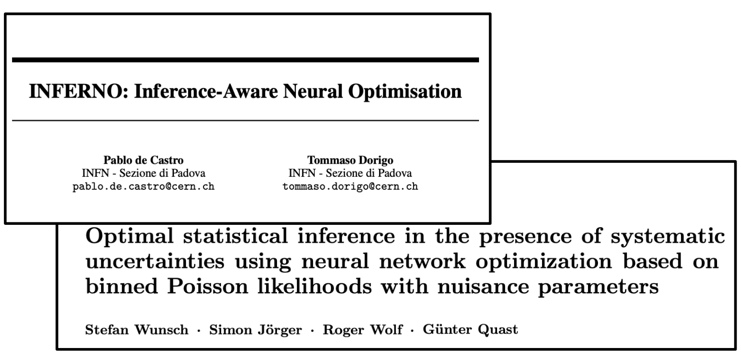
Long-Gang Pang[1,2],* Kai Z
[1] Physics Department, University o
[2] Nuclear Science Division, Lawrence Berkele
[3] Key Laboratory of Quark & Lepton Phy
Central China Normal Un
[4] Frankfurt Institute for Advanced Studi
[5] Institute for Theoretical Physics, Goethe

The structure of heavy nuclei is difficult to deep convolution neural network (DCNN) mig of heavy-ion collisions to the nuclear structur regression, we successfully extracted the magni

increase s/b AND minimize systematic uncertainties



https://github.com/pyhf/neos


INFERNO: Inference-Aware Neural Optimisation

Pablo de Castro
INFN - Sezione di Padova
pablo.de.castro@cern.ch

Tommaso Dorigo
INFN - Sezione di Padova
tommaso.dorigo@cern.ch

Optimal statistical inference in the presence of systematic uncertainties using neural network optimization based on binned Poisson likelihoods with nuisance parameters

Stefan Wunsch · Simon Jörger · Roger Wolf · Günter Quast

Infrastructure Needs:

support execution of large, distributed end-to-end pipelines with intact gradients (e.g. ray, torch elastic)

Software Needs:

easiest: build HEP software on top of popular autodiff platforms (jax, torch, tensorflow,...)

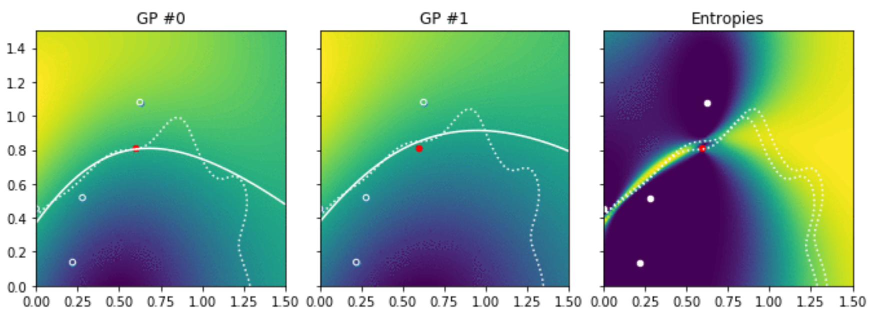
harder, but possible: integrate autodiff into existing software stack (e.g. HEP C++ fwks)

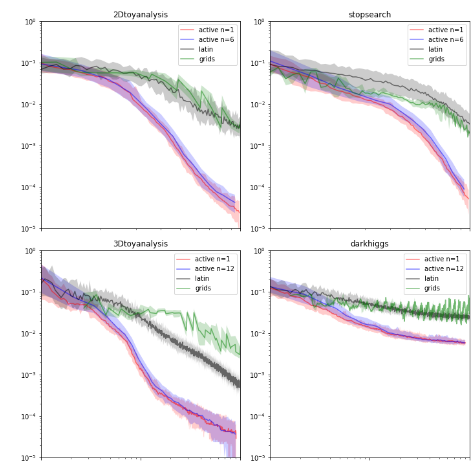
ML not inside the analysis but in the outer loop that manages the overall workflow (jobs to submit, decision-making, ...)

Examples:

Bayesian Optimization w/ Gaussian Processes:

ML-driven decision which Monte Carlo Samples to produces based on analysis results.

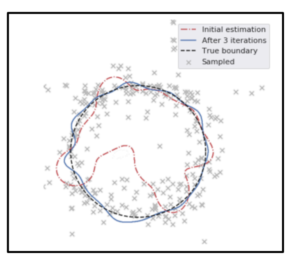


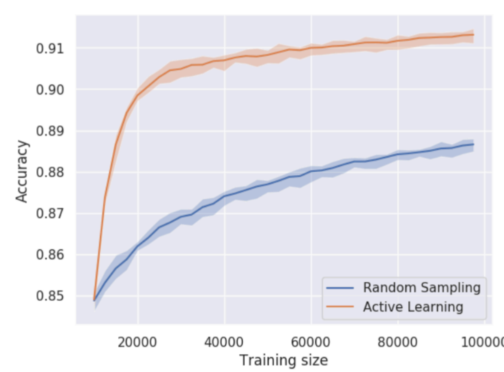


https://github.com/diana-hep/excursion

Active Learning (Query-By-Committee)

https://arxiv.org/pdf/1905.08628.pdf

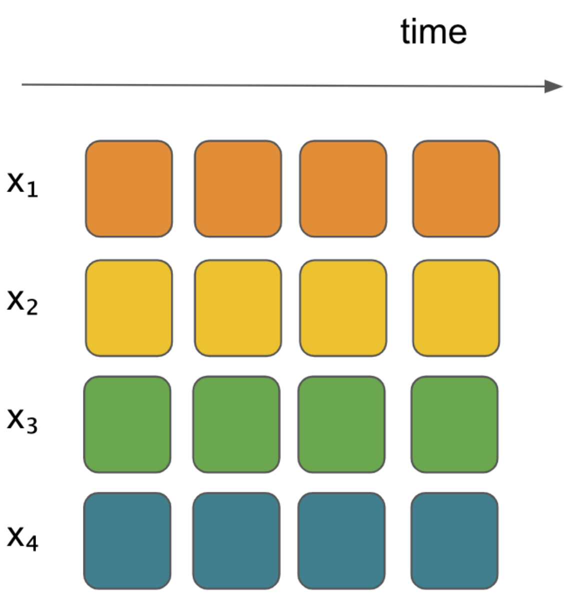




ML not inside the analysis but in the outer loop that manages the overall workflow (jobs to submit, decision-making, ...)

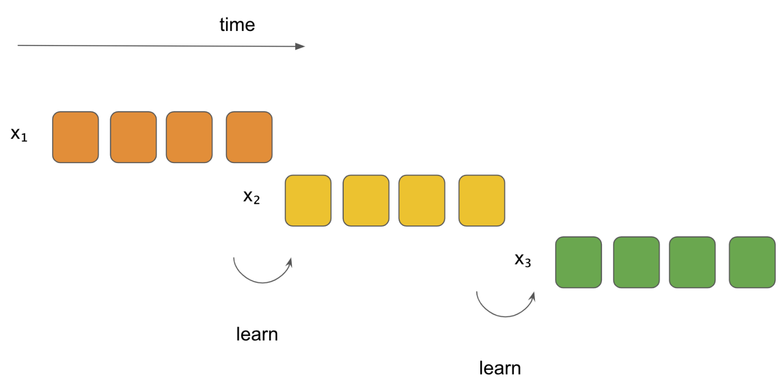
Infrastructure Need:

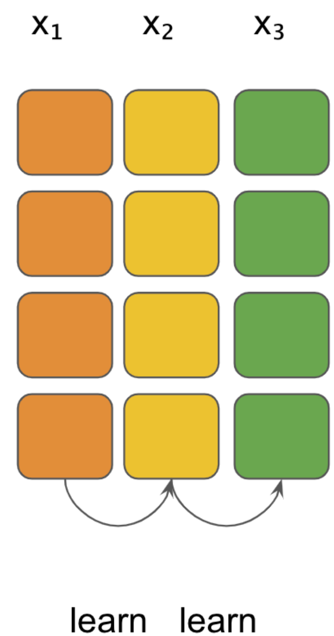
- dynamic workflow management (ML decision engine)
- high level of parallelism: not attractive if wall-clock time explodes
    - need good distributed computing



waste CPU but fast

smart use of CPU, but slow, due to serialization



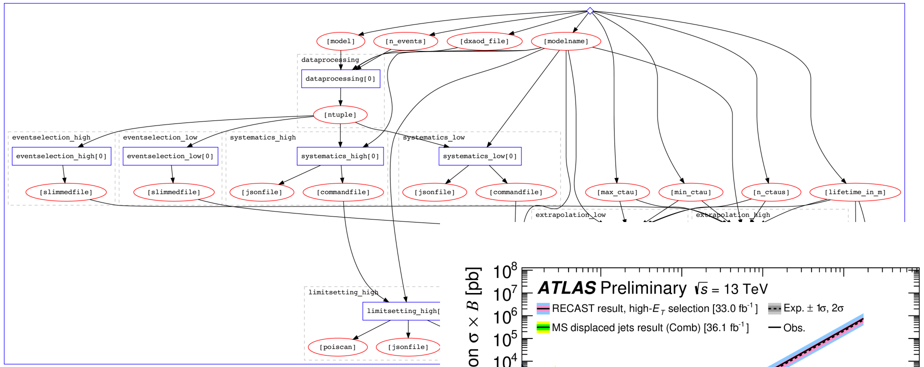


smart use of CPU, and shorter wall-clock due to parallelism

Declarative Workflows:

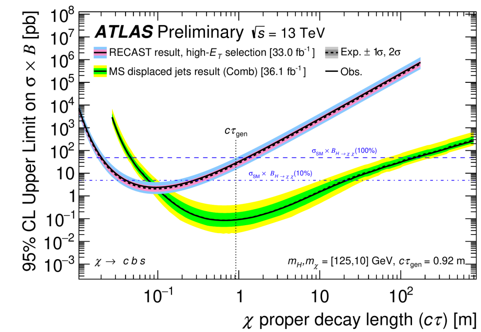
exploitation of full HL-LHC dataset physics potential analysis reusability is important. Example Use-case: Reinterpretation.

Idea of declarative workflows that process DAG of jobs important:
- heavily used in bio-informatics, genomics (CWL, nextflow, [Open]WDL)
- use in HEP: yadage, law (luigi-based)



Effort by CERN to provide workflow- as-service component: REANA
- **potentially part of analysis facilities.**

A. Morris (ATLAS) [paper]



M. Rieger (CMS) [slides]

Reproducible research data analysis platform

Conclusions

- Smart Data Handling
  - preprocess as much as possible for user
  - caches, smart data delivery
- Enable Declarative Analysis
  - DataFrames, Columnar Analysis, Workflow Languages
- Federatable Analysis Facilities (avoid silos)
  - use common technology widely used outside of HEP (Kubernetes, Containers,Jupyter Hub, ...)
  - compose existing tools rather than create new ones
- Native ML Integration
  - Hardware Acceleration
  - Differentiable Analysis Workflows