Data Analysis: Trends in Technology

Danilo Piparo (CERN) for the HSF DAWG Conveners

This Talk

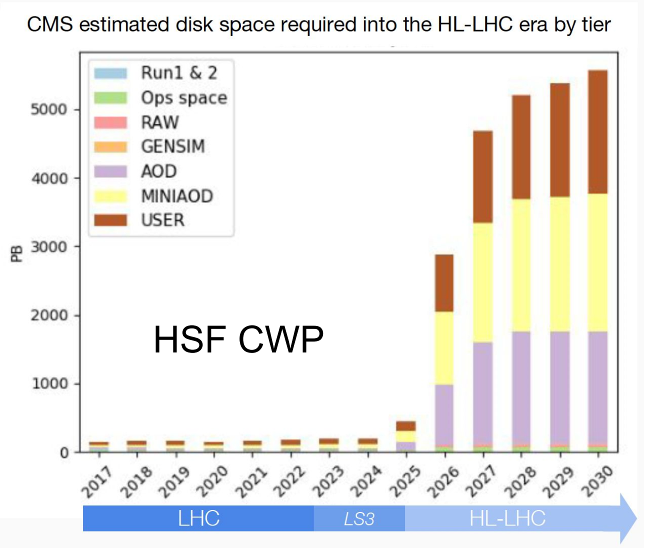
- Also elaborates input of first 2 DAWG events
  - DAWG Analysis Requirements Jamboree
  - 1st DAWG Technology and Innovation meeting
  - Thanks again to all attendees and 20 speakers!
    - Diversity: students, postdocs, senior scientists, LHC, e+e-, heavy ions, physicists, software specialists, universities, laboratories.
- Objective: ***summarise trends and themes*** identified in the contributions and discussions
  - Trigger discussions
- Not the final word about future analysis technologies
  - Rather the opposite: *another milestone of our journey*

Content

- Challenges posed by future datasets sizes
- Simplicity and programming model
  - Declarative analysis, analysis description, programming languages
- Parallelism and performance
- New interfaces
- Potential next steps

Future Dataset Sizes

- The Community is aware of the forthcoming challenges
- Not only a hurdle for LHC Run3 and Run4
  - Belle2: projected size of Phase3 dataset is 60PB
  - CMS W mass *Run2* precision study: O(1B) events needed already today
- Looking for the “right” set of tools
  - And analysis procedures
- At least three areas to invest in:
  - Programming model: simplicity
  - Performance and parallelism (in all forms)
  - Infrastructure: data management, analysis facilities



Strive for Simplicity

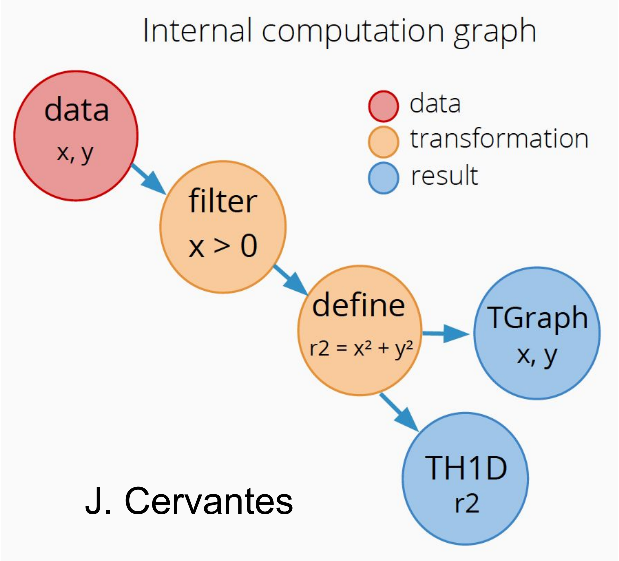
- An objective of many, very present in all contributions
  - Cost of dealing with complexity is high and does not scale linearly
- Different meanings of simplicity in different contexts
- Analysis specific frameworks
  - Handling of datasets, systematics, automatic bookkeping of results
  - “Easy to do the right thing” e.g. pick the right calibrations, workarounds
  - Rely often on very flexible configuration systems
- Programming model
  - Declarative approach, same code for local and distributed execution, high level description of data transformation and actions.

T.J. Khoo

Declarative Analysis

- Established approach: >40y old
  - Specify what you want and not how you want to do it
- Possible **frontend/backend separation** opens new possibilities:
  - Caching of intermediate results
  - Optimisations targeted to exploit hardware features
  - Analysis description languages, also important for Preservation
  - Transparently distributed computations
- A lot already achieved, potential hurdles ahead:
  - R&D needed, e.g. for efficient caching, state hashing
  - Paradigm shift wrt today's imperative approach, e.g. no explicit loops
  - Express functional approach with non-functional languages (Can we stick always to a functional programming style?)

Declarative Approach - Graphs and Queries




```
ele = electrons[(electrons.p4.pt > 20) &
                (np.abs(electrons.p4.eta) < 2.5) &
                (electrons.cutBased >= 4)]

mu = muons[(muons.p4.pt > 20) &
           (np.abs(muons.p4.eta) < 2.4) &
           (muons.tightId > 0)]
```

N. Smith

```
events
    .Select(e => e.Data.eventWeight)
    .FuturePlot("event_weights", "Sample EventWeights",
        100, 0.0, 1000.0)
    .Save(hdir);
```

What we want to plot

1D Histogram Declaration

Save the plot in a file

G. Watts

Languages

- The C++-Python duo is the reference
  - Functionality-, performance- and programming model-wise
- Clear trend: propose Python to physicists and accelerate it with C++/Python jitting and bindings to compiled libraries
- An example of C# (+LINQ)
  - Can we re-propose the useful concepts discussed w/o imposing the language itself?
- No in-depth discussion about this but the idea of an Analysis Description Language is in the air.

Parallelism and Performance 1/2

- We'll need efficient backends
- **Physicists cannot and will not always write optimised analysis code**
- Can we improve providing high quality trainings complementing universities' curricula?

Parallelism and Performance 2/2

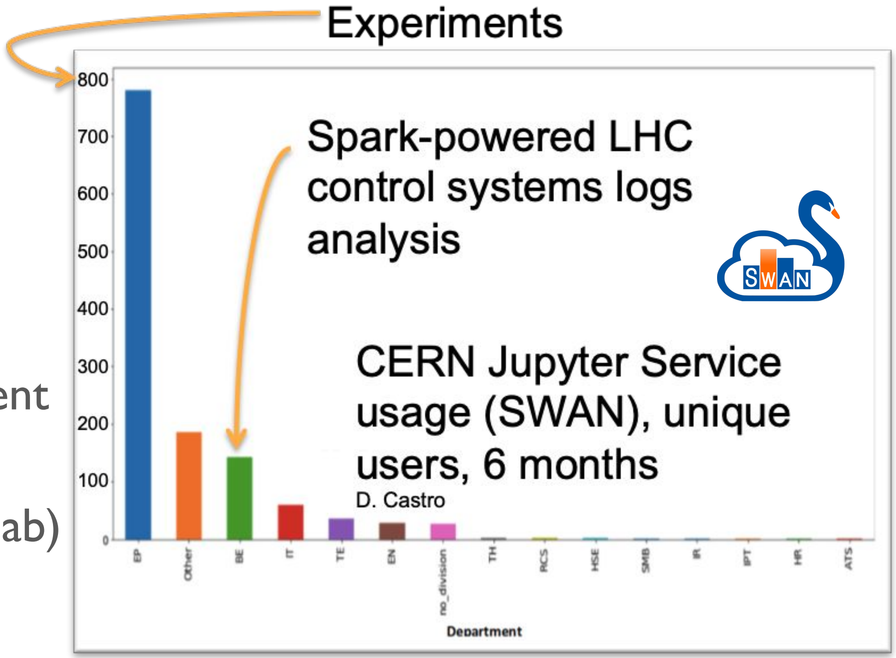
- Parallelism is a prerequisite for future analysis, in all forms
  - Data parallelism: accessed transparently via array syntax, backed by SIMD. Can we think to accelerators? Is it worth and under what circumstances?
  - Multithreading: become recently accessible with an acceptable programming model only with a declarative approach
  - Multiprocessing: needed for Python only frameworks, potentially legacy C++ code
  - Batch jobs: presently used extensively, HEP has extensive experience, not orthogonal with other ways of expressing parallelism (MT, MP, vectorisation)
  - Interactive distributed analysis: not an entirely new approach (PROOF). Revived thanks to tools such as Apache Spark.

Single Server and Distributed Parallelism

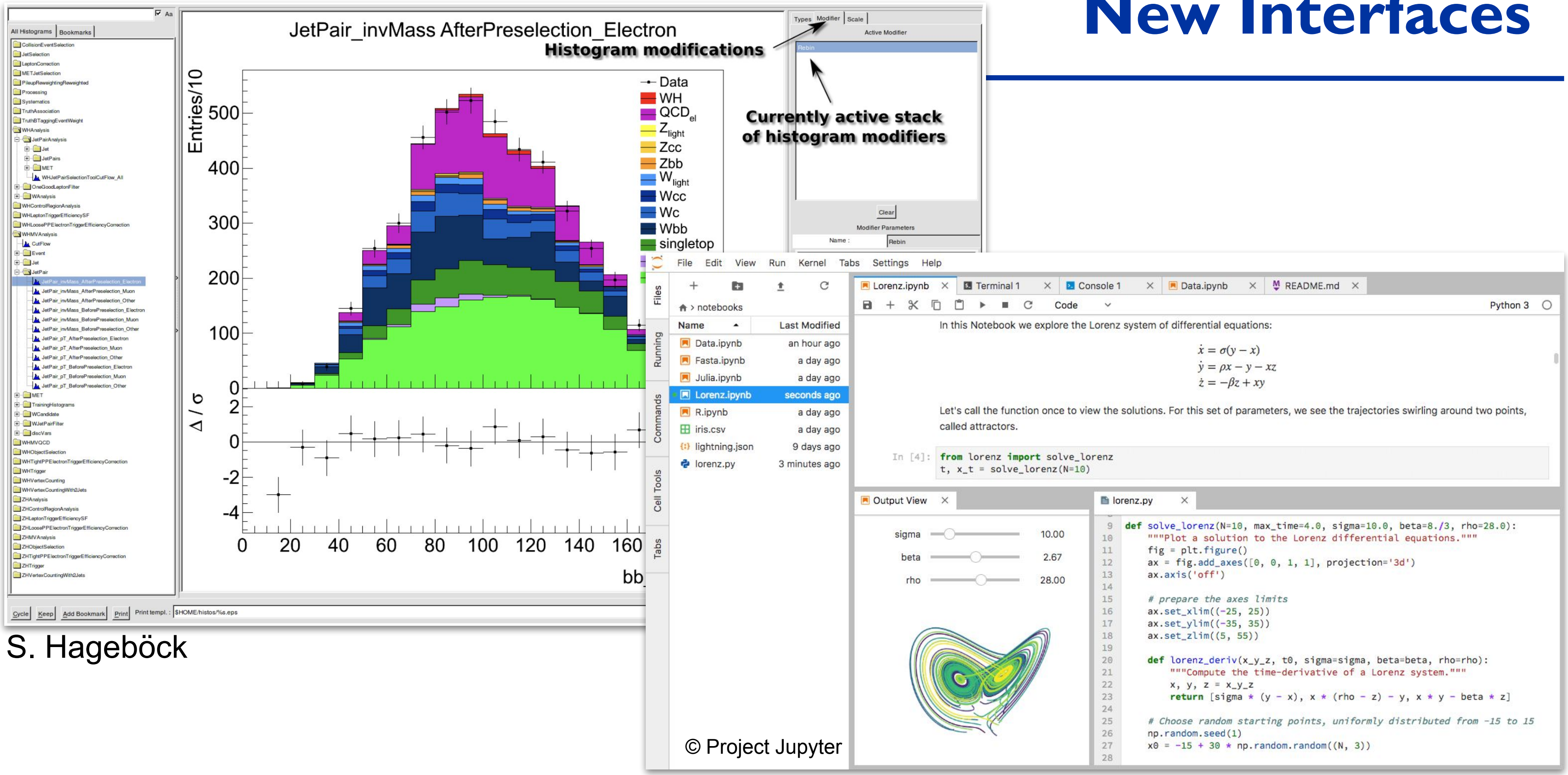
- We'll need to count on distributed (interactive/batch) computations
  - Manage clusters, e.g. in clouds
- We'll be able to count on O(100) cores individual servers
  - Available already now at some universities and labs
  - E.g. fast turn-around, checks required by analysis reviews
- Our software must be flexible enough to get the maximum of both kind of resources: parallelism is key.
- Can we identify an analysis use case for which the use of GPUs represents a game changer?

New Interfaces

- Not only compiled code + shell invocations
  - Interpreted code (C++ and Python)
  - Interactive exploration
  - Graphical User Interfaces
- Jupyter, not only notebooks
  - Results+explanations in the same document (Notebook)
  - Fully fledged web-based desktop (Jupyterlab)
  - Develop code, document and share
  - An ingredient for Preservation?



New Interfaces



S. Hageböck

Analysis Facilities

- Resources dedicated to analysis. Objectives:
  - Reduce as much as possible Δt between dataset arrival and results produced
  - Increase quality/quantity of scientific results within same resource envelope
- R&D needed, e.g. :
  - Data management beyond file based approach
  - Creation of columnar datasets `à la carte` (no slimming)
  - User interface: interactive, web based, explorative analysis
  - Optimisations, e.g. caching: user specific, common to all users
  - Partition of resources
  - User storage space, e.g. sync'n'shared

Wrap up and potential Future Steps

- Were all prototypes and production tools relevant for future HEP data analysis reviewed?
- Can concepts common to all HEP experiments be identified?
- Can a classification of the aforementioned tools help?
  - Table: row is the technology/tool, columns supported features
- Create a set of common benchmarks of increasing complexity
  - Compare and improve ergonomy of interfaces, performance
  - Can this be useful for testing/procurement of new hardware too?
- Next few months: organise WG meetings about benchmarks, ongoing efforts and collaboration strategies