Today's analysis ecosystem landscape and toolset

at HSF Analysis Ecosystem Workshop

Conveners: A Rizzi, A Krasznahorkay, PJ Laycock

Session preparation guidelines

- Survey of the HEP analysis ecosystems
- Look at analysis models of the experiments as they relate to a common analysis ecosystem
- Identifying commonalities, existing and new opportunities
- How to improve sustainability challenges of ROOT and other key components
- How can contributions large and small to the ecosystem be assimilated
- ROOT as the analysis ecosystem hub
  - Modularity, interfaces, packaging, discreteness of components
  - Using and integrating ROOT in C++, in Python, …
- ROOT 2017 workplan: input/discussion from a developers viewpoint

in view of Run 3 and Run 4 ...

Session plan

- P Mato
  ROOT's Place and its Capabilities in the HEP Analysis Ecosystem
- J Blomer
  Data Formats in HEP Analyses
- G Stewart
  Data Analysis in ATLAS
- G Cerminara
  Data Analysis in CMS
- P Canal
  Data Analysis at the Intensity Frontier
- E Rodrigues
  Data Analysis in LHCb
- M Ritter
  Data Analysis in Belle II
- S R Schramm
  The Machine Learning Landscape
- A Gilbert
  Last Steps of the Analysis
- Round Table
  (Final Questions)

Data formats in HEP analysis (J Blomer)

Comparison of several data formats (ROOT, protobuf, SQLite, HDF5, Parquet, Avro) for the LHCb OpenData analysis

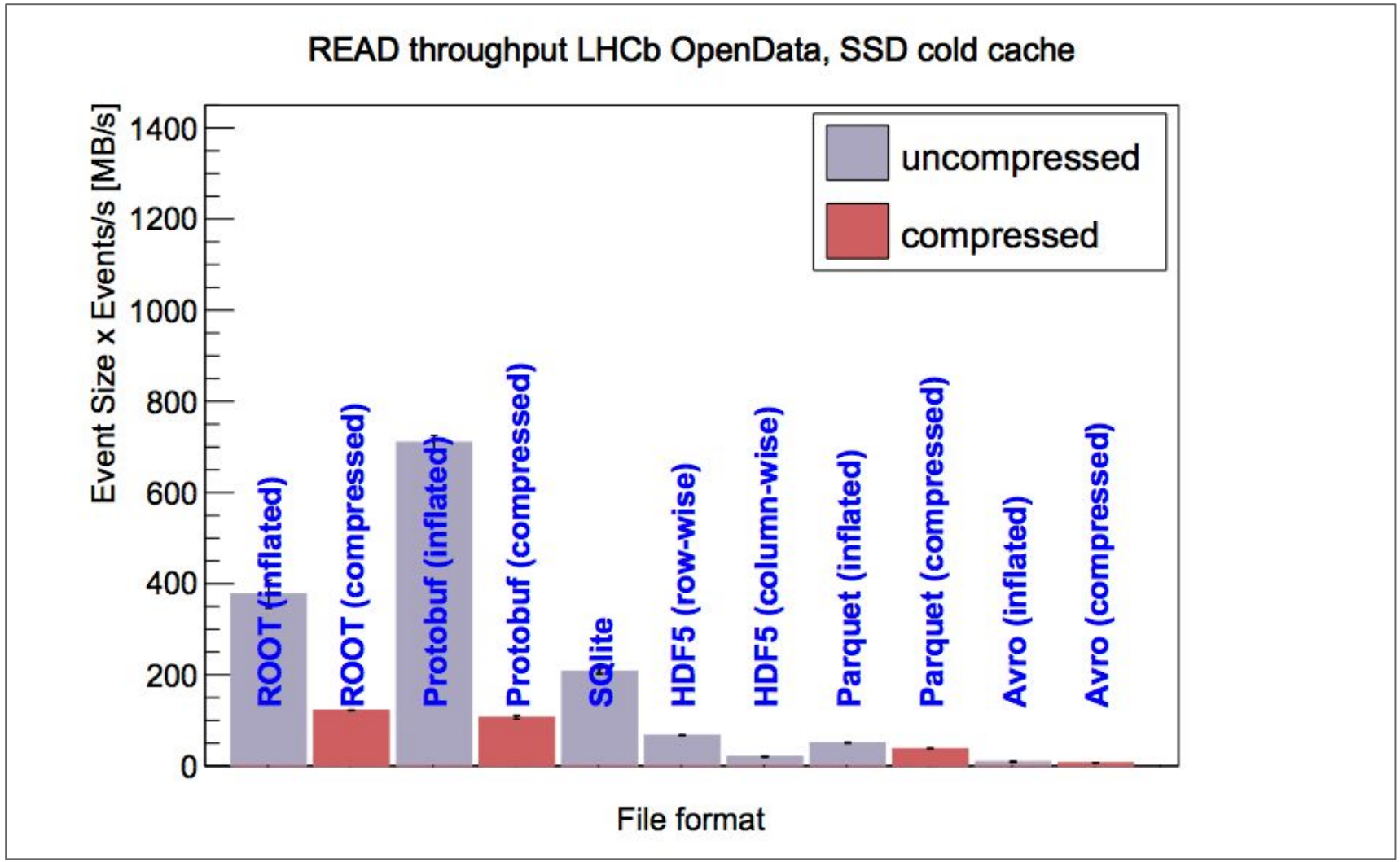
- ROOT technology still well placed
  - On the case studied (LHCb analysis) only **Google protobuf** does better when reading the full event
    - But protobuf is not columnar and has no schema evolution
- On fast storage: performance is dominated by deserialization, compression default may be reconsidered

Related talk on data formats by Jim Pivarski (Diana)

- ROOT:
  - *can do anything with the right settings*
  - *is the best way to access petabytes of HEP data and use tools developed in HEP*

Read throughput - SSD

(J Blomer)



The Machine Learning Landscape (SR Schramm)

- The HEP-ML landscape is currently undergoing a substantial change
  - A few years ago, TMVA dominated the market
  - Now, external tools are increasingly replacing TMVA
- Deep learning is increasingly prevalent in HEP-ML
  - Usage is growing fast and is now ~even with BDTs
  - This is a large part of the shift towards external tools
  - This is also creating a need for GPUs
- Dominant community request: ROOT integration with external tools

Last Steps of the Analysis (A Gilbert)

- For statistical analysis and combination within and between the experiments
  - **RooFit + RooStats** *recommended*
    - For the flexibility in defining models
    - Ease of likelihood level combinations
    - Persistence via RooWorkSpace
- Reinterpretation of results
  - No one-size-fits-all solution
    - Tools such as Rivet, HEPData help standardizing formats
    - Would benefit from common software framework to run fits and create signal parametrizations

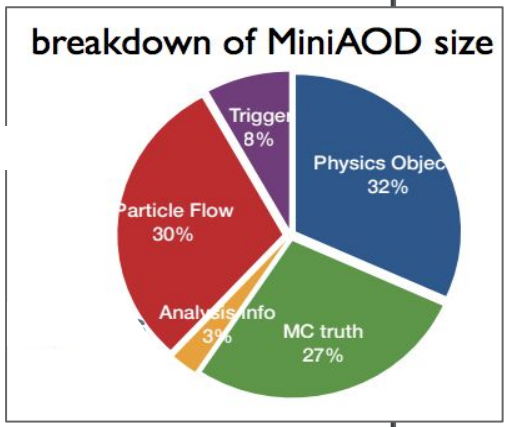
Experiments' reports

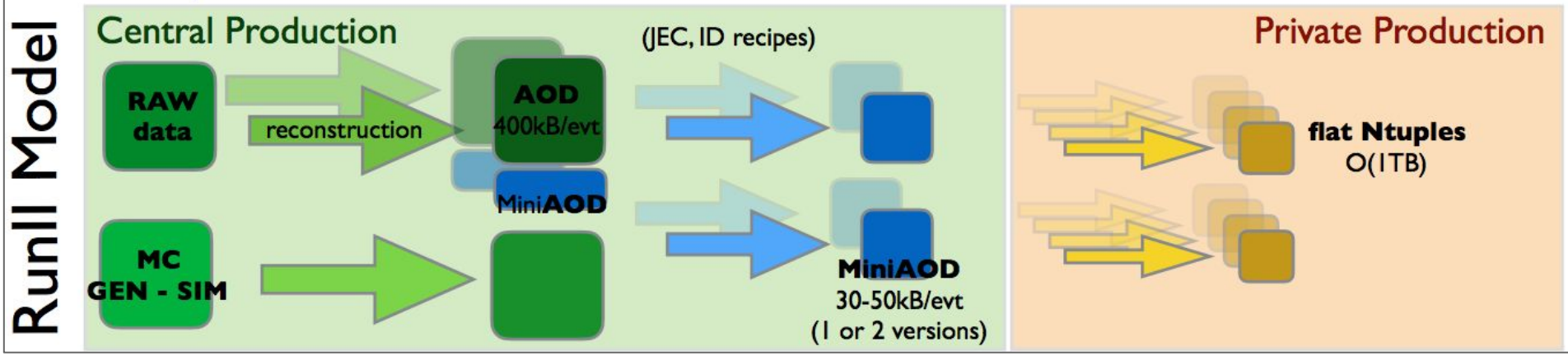
- Need to improve resource utilisation:
  a. Central provision of slim and flexible analysis data formats
  b. Increase level and quality of parallelism
     i. GPU, vectorization, multi-thread, multi-process
  c. Move analysis parts to previous levels, e.g. HLT (LHCb)

CMS Run II lightweight data-model: MiniAOD

(G Cerminara)

- fast to produce (~1s/event from AOD) & fast to analyze
- small event footprint: ~10% of AOD size (~40kb/event)
  - typical needs of big analysis (30% of all data for 1y) fit 40-50TB
- common processing AOD ⟶ MiniAOD for all analysis
  - analysis level corrections & recipes for Object ID applied centrally (e.g lepton ID, b-tagging, mitigation of unexpected issues with real data, jet energy corrections)
  - can be re-produced from AOD to include newer high-level calibrations and improvements





Experiments' reports

- Need to improve resource utilisation:
  a. Central provision of slim and flexible analysis data formats
  b. Increase level and quality of parallelism
     i. GPU, vectorization, multi-thread, multi-process
  c. Move analysis parts to previous levels, e.g. HLT (LHCb)

- Common interface for condition (non-event) data at analysis level
  a. E.g. Belle II REST service

Belle II conditions data

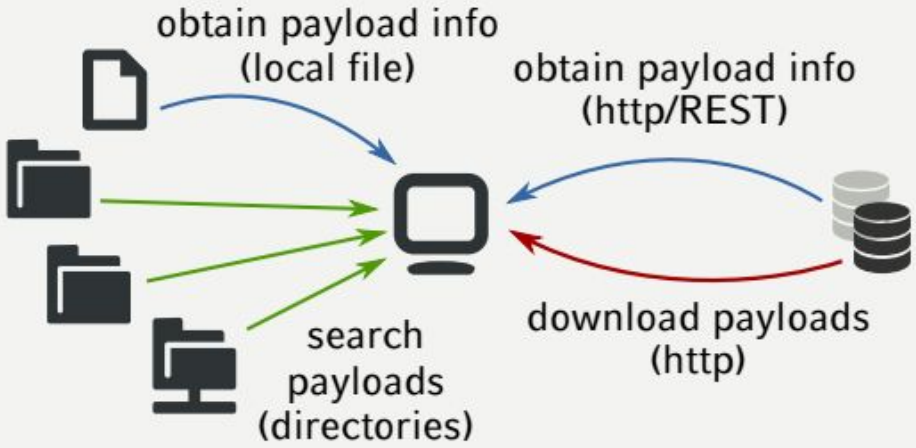
(M Ritter)

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Conditions Data

Belle II uses REST-based Conditions database:

- payload: named file valid for certain interval
- distributed server architecture (hazelcast)
- payloads usually ROOT files
- command line tool to manage payloads
- software can run with downloaded snapshot
- requires only http(s)



Flexible Payload distribution

- cascade of lookup directories for local copies
- download on demand if not present
- reuse between executions if possible

Experiments' reports

- Need to improve resource utilisation:
  a. Central provision of slim and flexible analysis data formats
  b. Increase level and quality of parallelism
     i. GPU, vectorization, multi-thread, multi-process
  c. Move analysis parts to previous levels, e.g. HLT (LHCb)

- Common interface for condition (non-event) data at analysis level
  a. E.g. Belle II REST service

- Python as a first level language
  a. Offload performance-critical code to C++ libraries

LHCb software ecosystem

(E Rodrigues)

Purpose	Software	Language of use	HEP ?
Data manipulation	ROOT	C++ & Python	Yes
	numpy, pandas, bcolz	Python	No
	root_numpy, root_pandas	Python	Yes
Machine learning (classification, regression)	TMVA	C++ & Python	Yes
	scikit-learn	Python	No
	NeuroBayes	C++	No
Plotting	ROOT	C++ & Python	Yes
	matplotlib, seaborn, bokeh	Python	No
Fitting	RooFit	C++ & Python	Yes
	<Institute/user packages>	C++	Yes
Statistics	CLs	Python	Yes
	RooStats	C++ & Python	Yes
Reweighting	hep_ml	Python	Yes & no
Error propagation	uncertainties, mcerp	Python	No

Other packages some analysts use

- Docker for the runtime environment, Snakemake for defining the analysis pipeline
- jug for submitting jobs to the batch system
- Note: MC programs not listed

Note: not claiming it to be a comprehensive list.

Experiments' reports

- Need to improve resource utilisation:
  a. Central provision of slim and flexible analysis data formats
  b. Increase level and quality of parallelism
     i. GPU, vectorization, multi-thread, multi-process
  c. Move analysis parts to previous levels, e.g. HLT (LHCb)

- Common interface for condition (non-event) data at analysis level
  a. E.g. Belle II REST service

- Python as a first level language
  a. Offload performance-critical code to C++ libraries

- Adoption of GIT + automatic integration services helps code quality (and archiving)

Additional points of discussion

- I/O performance on complex (non-ntuple) format
  a. Suffering CMS (for sociological reasons), ATLAS (unknown reasons ...)
- Availability of new Machine Learning developments in ROOT
  a. TMVA as interface-to-all or integration of external toolkits?
- Approaches to parallelization, including GPU
  a. Level of details to be exposed to people
- Role of functional programming
  a. Suitable to express full-featured use-cases of HEP community?
  b. E.g. TDataFrame and non-event data