Advances in Experimental Analysis Frameworks

Jan Fiete Grosse-Oetringhaus (CERN)
on behalf of the ALICE, ATLAS, CMS and LHCb collaborations

Large Hadron Collider Physics Conference
June 2021

Challenges, or:

Why change analysis frameworks?

- **Ongoing upgrades for Run 3+4**
- **Data-taking rate to tape : factor 100 up**
- **100 GB/s to storage**
- **Trigger-less readout in Pb-Pb**
- **AOD: ~5 PB/year (analysis input)**
- **Full software trigger / event selection in pp**

- **HL-LHC (Run 4+)**
- **Main upgrades for Run 4+**
- **5 times higher L_{inst}**
- **Pile up increases data volume per event**
- **Today's AOD format would lead to 49 PB/year data + 200 PB/year MC**
- **New reduced formats: PHYS / PHYS_LITE**
- **~15 PB/year as analysis input**

- **HL-LHC (Run 4+)**
- **Main upgrades for Run 4+**
- **5 times higher L_{inst}**
- **Pile up increases data volume per event**
- **Data formats based on existing MiniAOD and NanoAOD formats**

- **Ongoing upgrades for Run 3+**
- **Full software trigger and event selection**
- **10 GB/s to storage**
  - **68% turbo events (no re-reconstruction)**
- **~15 PB/year as analysis input**

WLCG Computing Resource Evolution

Can we just buy more computers?

- During Run 1 and 2 the growth (CPU, disk) was about 20%/year
- Flat budget scenario during Run 3+4: 15%/year
- Today, resources are saturated
- 15% over 10 years give factor 4.
  E.g. ALICE will take 100 times more data
- Opening additional analysis avenues requires more resources
- Computing power for analyzing additional data cannot be absorbed by growth of computing resources

Back of the envelope:
Need to gain factor 25 (disk and CPU) through smarter strategy and algorithms

Source

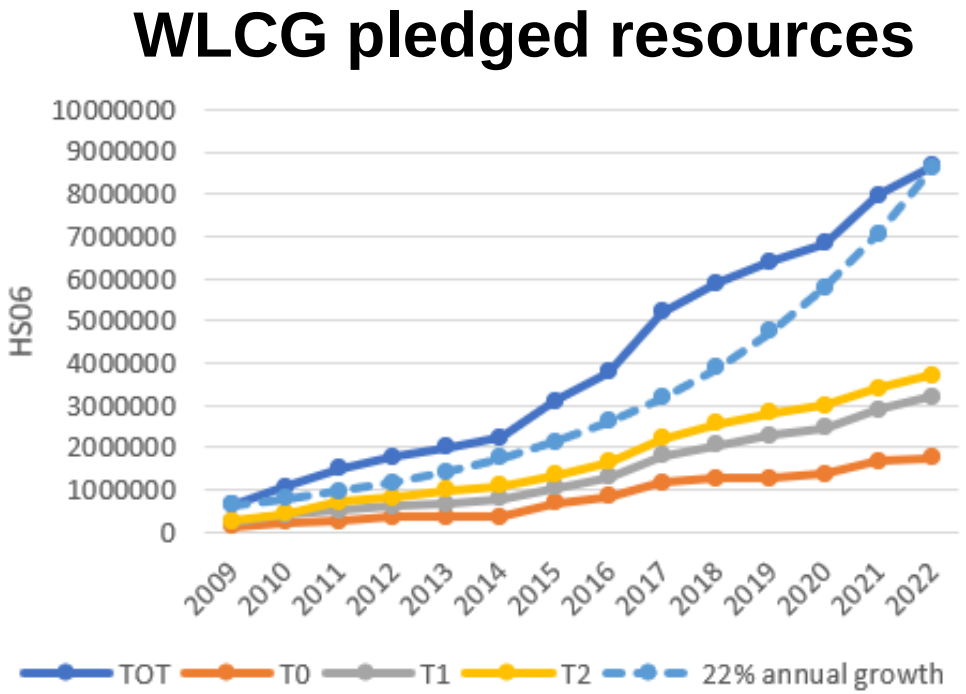


Figure: Stefano Piano, based on WLCG

10 HS06 ~ 1 core

Concepts and Trends

Reduce disk per recorded event ⟷ **opposite direction** ⟷ **Reduce CPU per analysed event**

In general: disk constraints drive the decisions
"what you cannot store, cannot be stored. Lack of CPU 'just' delays analysis"

- Smaller analysis input formats and skims
- Highly optimized compressed formats
  - CPU-expensive (but affordable) transition from disk to in-memory representation
- Store processed output instead of RAW
- Store only analysis input (AOD)
  - No re-reconstruction possible, but increase statistics for analysis

- Bulk processing
- Vectorization / parallelization
- Declarative analysis
- Modularity / multi-core
- Batch / Interactive
- Organize differently through analysis facilities

In the following, I will present some details of these concepts

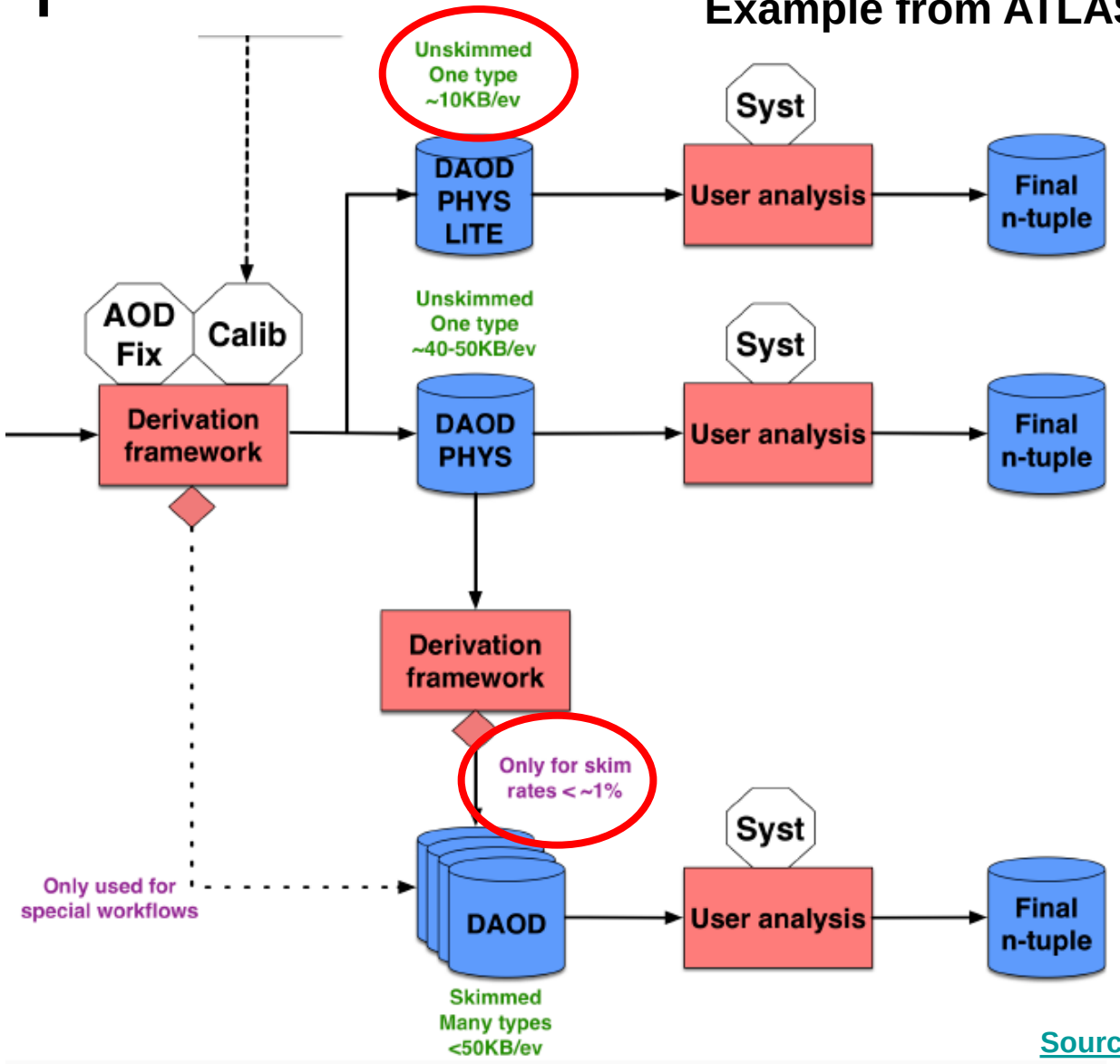
Analysis Input Formats

- Increased data volume requires smaller and more targeted analysis input formats
  - Individual formats only if filtering rate < 1%

ATLAS	AOD	DAOD PHYS	DAOD PHYSLITE
events / year	$7 \cdot 10^{10}$	$7 \cdot 10^{10}$	$7 \cdot 10^{10}$
size/event [kB]	700	50	10
disk [PB/year]	49.0	3.5	0.7

- Tape carousel for original reco output
- RAW data not stored for all events (all exp) ("Turbo stream", "Trigger-level analysis")
  - No re-reconstruction possible, but more statistics for analysis
  - "Dilemma": Discard event or RAW data

Example from ATLAS

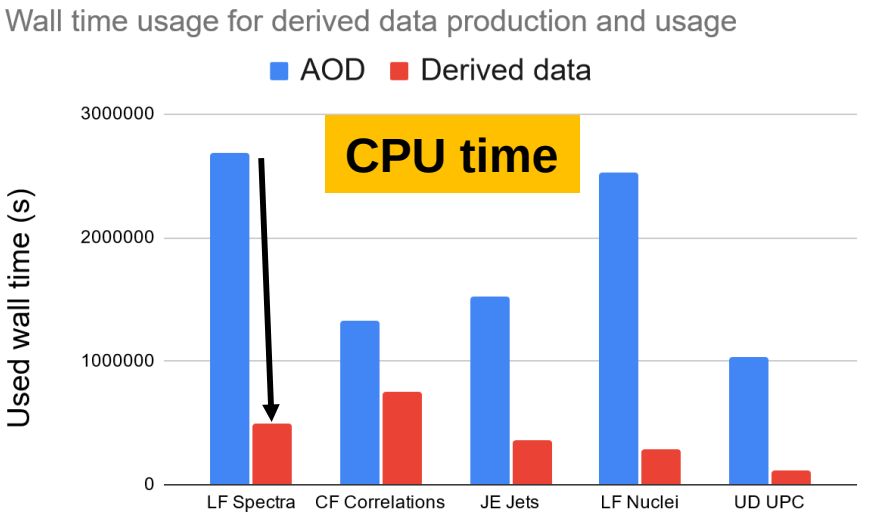


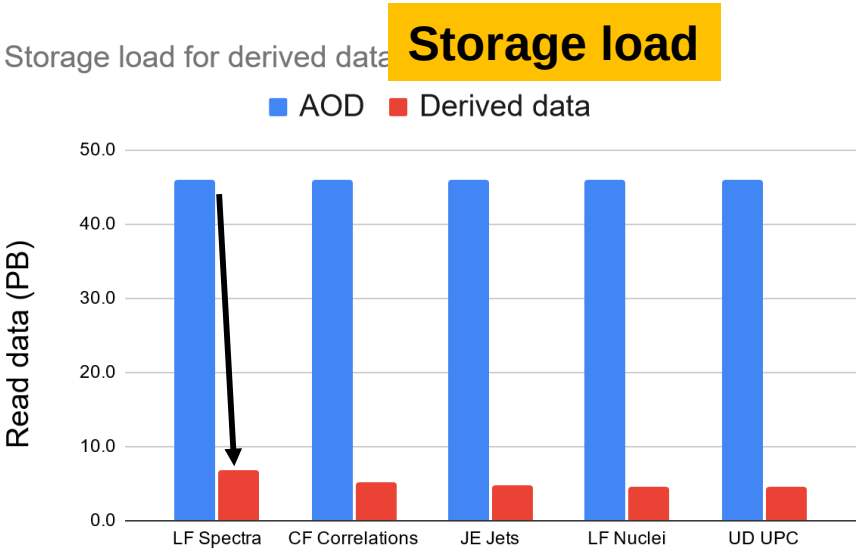
Source

Skimmed Data Formats

- Specific reduced formats serving an analysis groups
  - Already used in Run 1 and 2
  - Production supported by experimental frameworks
- Certain analysis completely removed from Grid
- Certain analysis moved to analysis facilities (see later)
- Significant reduction of
  - needed CPU to do the same analysis
  - load on the storage (less data read)

Plots from ALICE





Columnar Formats / Flat Tables

Run 1 and 2

```
Collision {
  float VertexPosX
  float VertexPosY
  …
  vector<Tracks> tracks
  vector<Jets> jets
}
```

```
Track {
  float eta
  float phi
  …
}
```

Nested objects
Collections
Referrals

New: Tables

Collisions

VertexPosX	VertexPosY	…

Tracks

CollisionID	eta	phi	…
0			
0			
1			
1			
1			
2			

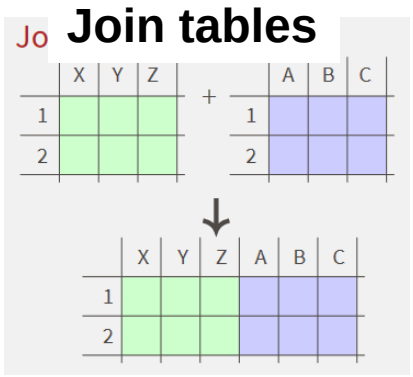
Flat tables (no nesting)
Similar to relational databases

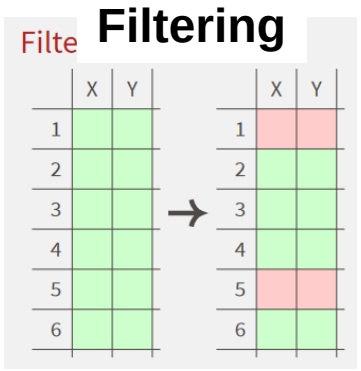
- **More disk space? → No. Compresses very well.**
- **Table operations have to be shielded from the user**

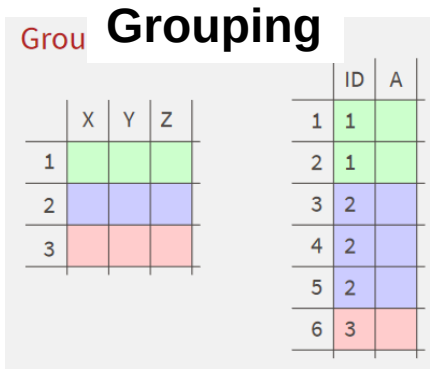
Operations are not applied per collision/event, but for large data blocks with 1000s of collisions (→ speed gains)

Real Example (ALICE)

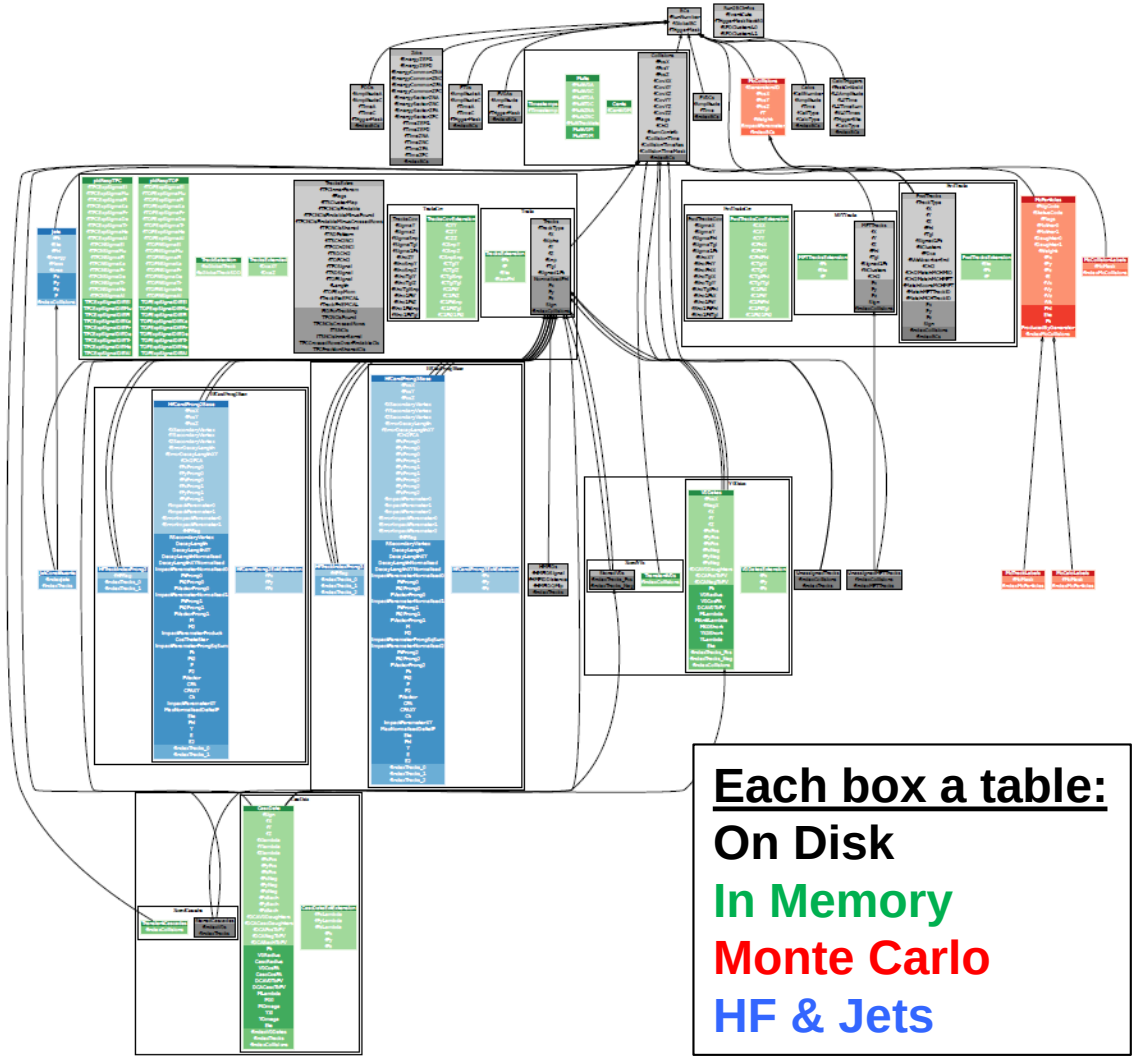
- Tables allow fast and bulk manipulation







- Complexity of representation is (and has to be) shielded from user
  - Object “look and feel”: track.pt()
  - In memory expansion of values:
    Stored: $\pm 1/p_T$ → Accessed: pt() and sign()
- Focus: least footprint on disk



Each box a table:
On Disk
In Memory
Monte Carlo
HF & Jets

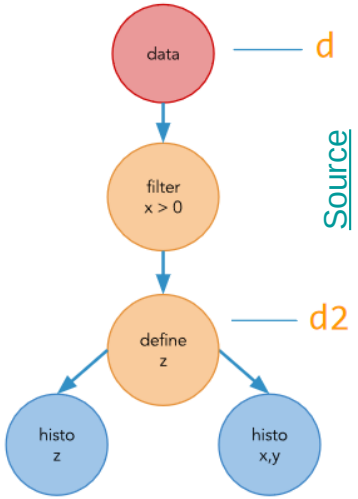
Focus on Compression

- Focus on optimal representation for compression
- Better compression algorithms
- Large baskets (block which is continuous in memory and disk)
- Optimize stored quantities for compression algorithm. Example:
  - Number of found clusters per track compresses badly (wide dynamic range)
  - Number of found-expected clusters compresses better (small dynamic range)
- Lossy compression
  - Reduce floating-point precision according to detector resolution

findable	found	findable - found
100	80	20
95	78	17
62	44	18
85	70	15

Smaller dynamic range
Better compression

Declarative Analysis



- Declarative analysis popular (e.g. RDataFrame, Coffea)

```
auto muons = df.Filter("dz < 0.1 && dxy < 0.01 && rel_Iso < 0.5");
auto histogram = muons.Filter("pt > 2").Histo1D("eta");
```

- User advantages: Clear compact definition of analysis cuts and flow
- Better resource utilization
  - Cut sequence can be optimized behind the scenes (restrictive cuts first)
  - Derived quantities calculated only once
  - Vectorization and parallel processing can be done by framework
- As usual there is no free lunch
  - Some conditions difficult to implement in a declarative way
  - "if detector condition A, and 3 muons, and highest p_T muon went into direction of detector module C, then apply an ad-hoc correction for the wrong HV setting"
  - Nested loops with certain conditions

Declarative + Imperative Analysis

- Not all analyses (or all parts of analysis) can be put in this schema
  - Risk: Two distinct classes of analysis (classic + declarative)
- ALICE framework combines declarative and imperative approach
  → “best of both worlds”

```
Filter vertexFilter = nabs(collision::posZ) < 7;
Filter ptFilter = track::pt > 0.5f;

void process(Collision const& collision,
                        Tracks const& tracks)
{
  // some complex event selection
  // which does not work declarative

  for (auto& track : tracks)
    hist.Fill(track.pt());
}
```

Declarative: filters, grouping

- **Could be optimized, vectorized, parallelized by fw**
- **Subscribes to single track, or complex collection**
- **Highly optimized vectorized framework code prepares collection**

Imperative user part

- **Completely flexible**
- **“Good old style”**
- **Limited implicit parallelization**

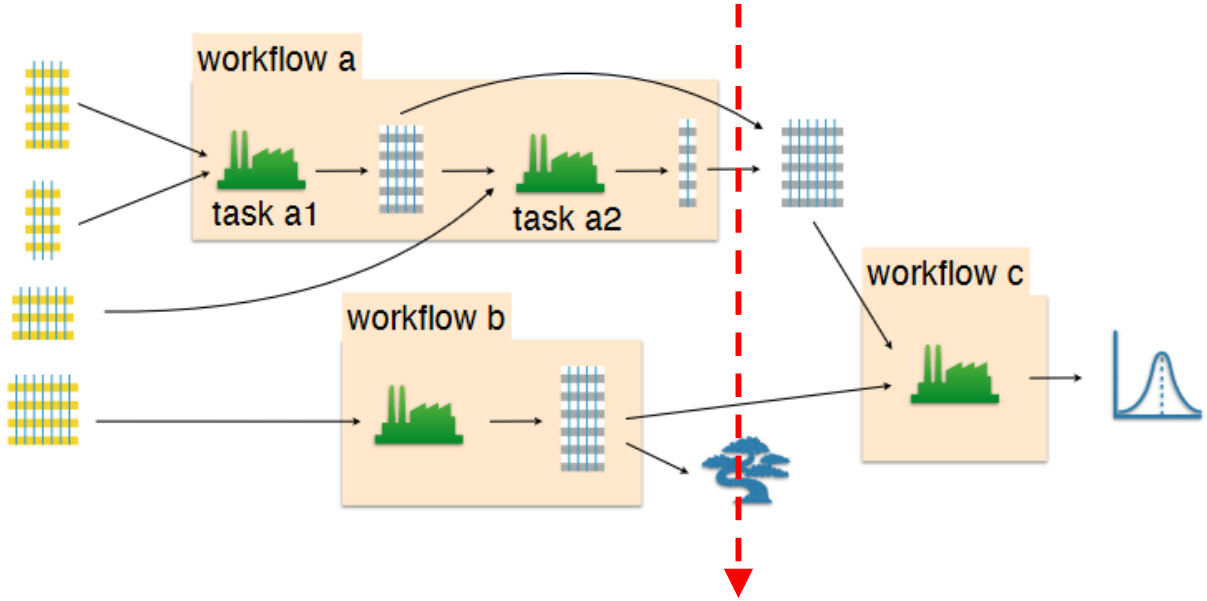
See vCHEP talk

Modularity and Multi-Core

- Analysis split into several blocks
- Each block consumes tables, produces histograms or tables

- Service tasks (event selection, track selection, secondary vertex finder, …) reused

- Information flow through tables between processes (zero copy, shared memory)
- Declarative parts only done once (if common)
- Multiple processes automatically exploit multi-core architecture



ALICE example, Source

- **Interrupt here and write to file**
- **Use as input for further processing**

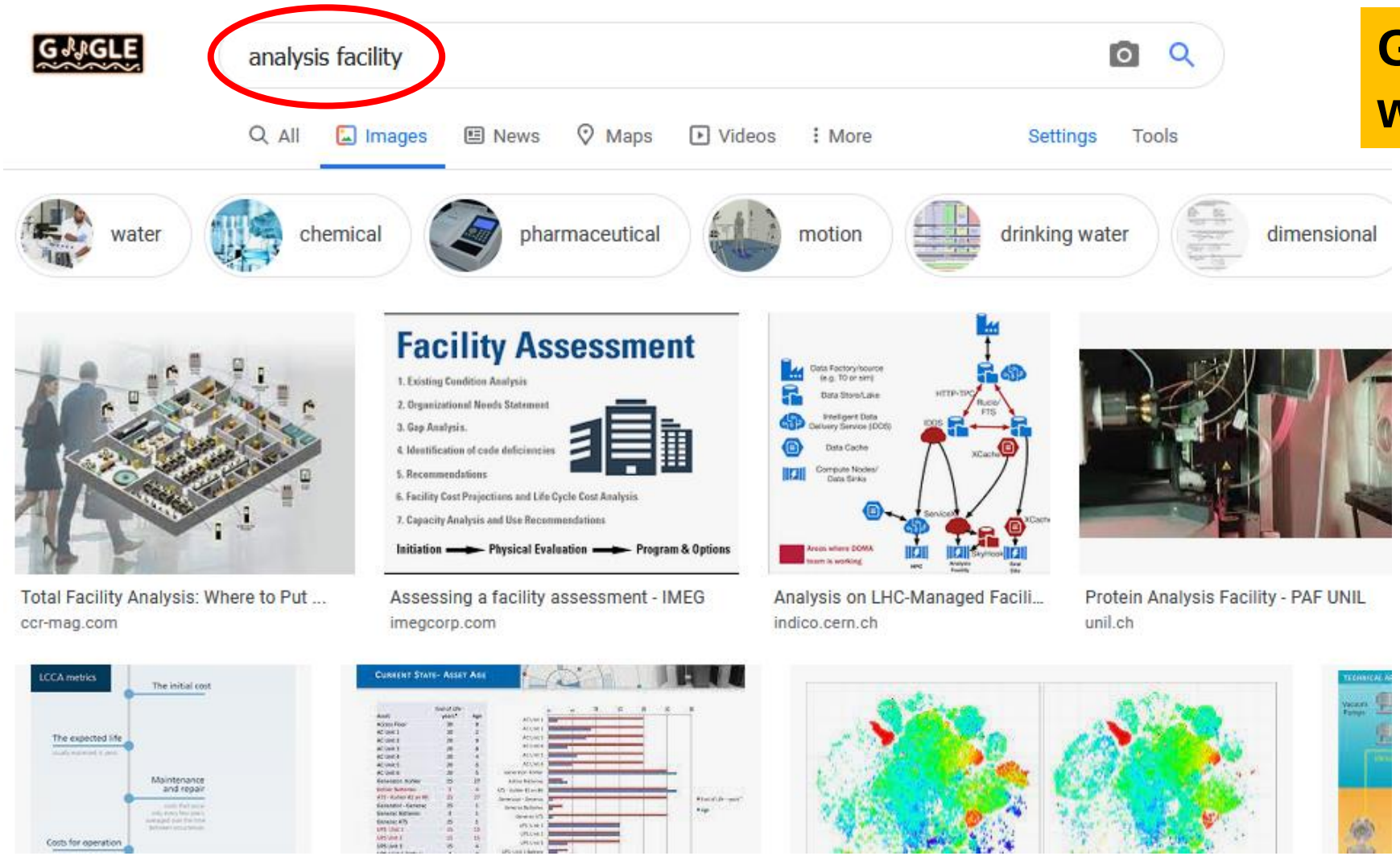
With Run 3 data model and analysis framework, ALICE demonstrated factor 15 smaller event size and factor 10 higher event throughput (without deep optimization)

Organized Analysis / Trains


Source

- Bundle several analysis or productions into a train
  - Read once, process for many different tasks
  - Reduces overall resource consumption
- Centralized production of derived data / skims (all experiments)
  - ALICE (→ derived data), ATLAS (→ DAOD), CMS (→ miniAOD/nanoAOD), LHCb “sprucing” (→ analysis productions)
- Bundling of user analysis (ALICE)
  - Adds bookkeeping, shielding users from Grid (submit, resubmit, merge)
  - Extensively used by ALICE in Run 1 and 2 (> 90% of all analysis activity)
  - Run 3+ : staged submissions (run on 1% of the data, then 10%, then 100%)
    - Integrated approval process
  - Derived data / skim production and processing on subsequent train
- Central activities can be integrated with tape carousel / on demand staging

Analysis Facilities



Google does not know what it is, so do we?

Analysis Facilities

Significant resources
E.g. ALICE: 5 PB, 5000 cores

- Analysis facilities in addition to Grid analysis
- Different ideas in different experiments
- LHCb: to investigate exploitation of GPU trigger farm for offline analysis
- CMS: prototyping of interactive facilities
  - "Coffea-casa" (see vCHEP) and portable analysis infrastructure (see vCHEP)
  - Facility allowing interactive parallelized python analysis
  - Data delivery service which produces user-specific flat tables "over night" (< 2kB/event)
  - Containerization on top of Kubernetes (can work also in commercial clouds)
- ATLAS: aim to provide interactive scale-out systems (C++ or Python)
  - to be able to hold full PHYSLITE dataset or efficient columnar data delivery/caching
- ALICE: dedicated Grid sites with sufficient storage for analysis input
  - Reconstruction output and user-defined skimmed data
  - Accessed through organized analysis trains (not interactive)

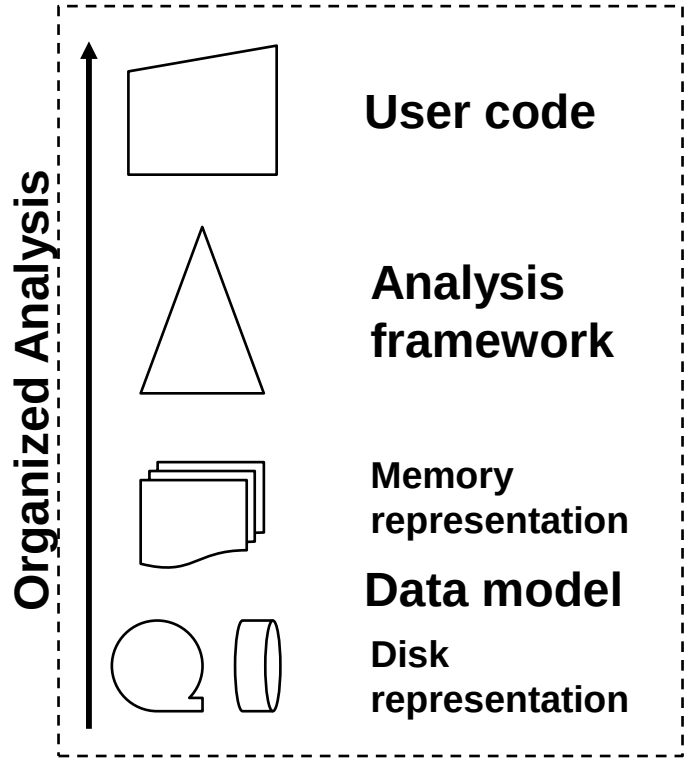
Increased Usage of Python

- Classical viewpoint: easier coding (Python) vs. optimized code (C++) leads to slower processing in Python → not true anymore
- Python present in all experiments
  - For parts of the analysis (plotting), or full analysis
- LHCb leading in Python usage for analysis
  - Application configuration and running of jobs entirely in Python
  - Already in 2018, 50% of analysis in Python
- CMS prototypes interactive Python-based analysis facilities
- Machine learning tools mostly in Python
- Growing number of toolkits and interactive solutions (Jupyter, SWAN platform, Scikit-HEP, Dask, …)
- Substantial influence by IRIS-HEP initiative

Summary

Questions during the coffee break in my zoom room

- LHC and detector performance grow faster than computing resources
  - Factor 100 more data but factor 4 more resources
  - Novel ideas and extension of physics programme beyond the usual paradigm (hardware trigger → record → analyze)
- Innovative rewriting and optimization of experimental frameworks
  - Columnar formats, in-memory expansion
  - Multi-processes, declarative analysis
- Analysis Facility concepts different in different experiments
  - Dedicated Grid sites vs. interactive solutions (often Python-based)
  - Room to learn from each other



I thank the ALICE, ATLAS, CMS and LHCb experts for input and discussion in the preparation of this talk