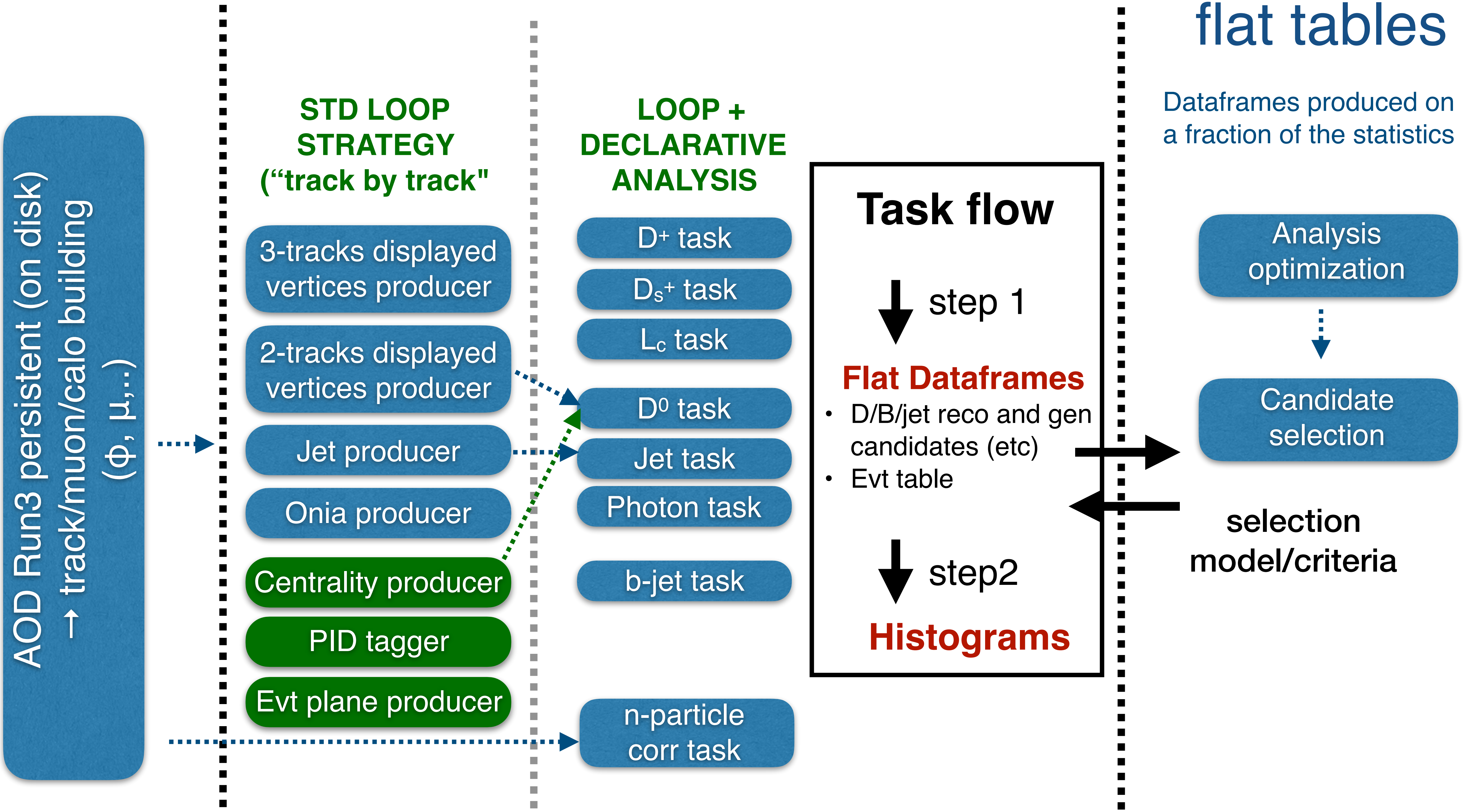
Brainstorming for Run3 analysis software

P. Hristov, A. Morsch, G. Eulisse,
J. F. Grosse-Oetringhaus, G.M. Innocenti

AOD

Lego Trains with DPL

Analysis on flat tables



AOD Run3 persistent (on disk) → track/muon/calo building (ϕ, μ,..)

STD LOOP STRATEGY (“track by track"

3-tracks displayed vertices producer

2-tracks displayed vertices producer

Jet producer

Onia producer

Centrality producer

PID tagger

Evt plane producer

LOOP + DECLARATIVE ANALYSIS

D^+ task

D_s^+ task

L_c task

D^0 task

Jet task

Photon task

b-jet task

n-particle corr task

Task flow

step 1

Flat Dataframes

- D/B/jet reco and gen candidates (etc)
- Evt table

step2

Histograms

Dataframes produced on a fraction of the statistics

Analysis optimization

Candidate selection

selection model/criteria

- Optimized Kernels

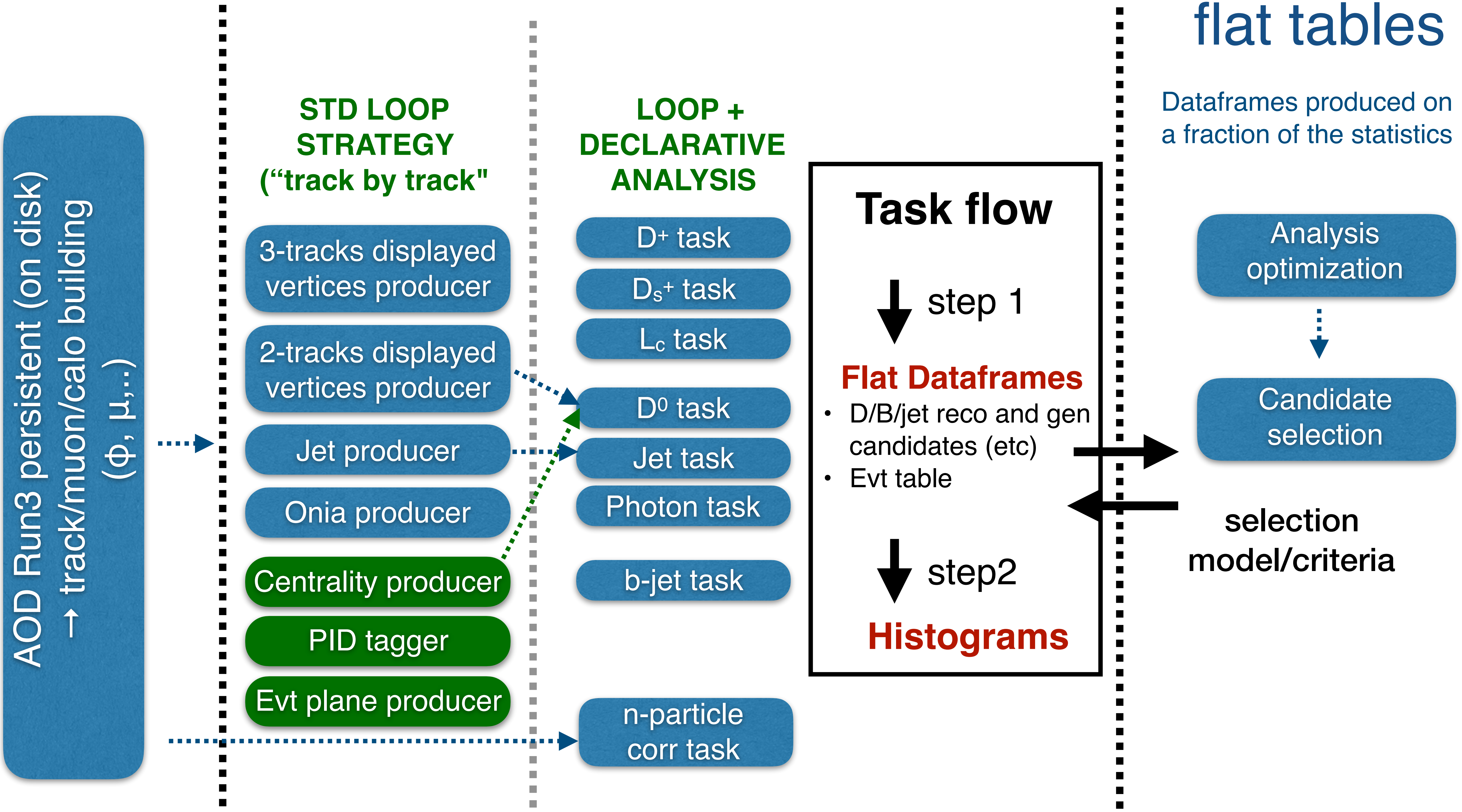
- Maximize number of quantities produced “centrally”
- Objects shared via message passing
- Nothing written on tape during intermediate steps

- Run on the Grid or in clusters of HPC servers
- GPUs for optimization and fast processing

AOD

Lego Trains with DPL

Analysis on flat tables



AOD Run3 persistent (on disk) → track/muon/calo building (φ, μ,..)

STD LOOP STRATEGY (“track by track"

- 3-tracks displayed vertices producer
- 2-tracks displayed vertices producer
- Jet producer
- Onia producer
- Centrality producer
- PID tagger
- Evt plane producer

LOOP + DECLARATIVE ANALYSIS

- D^+ task
- D_s^+ task
- L_c task
- D^0 task
- Jet task
- Photon task
- b-jet task
- n-particle corr task

Task flow

step 1

Flat Dataframes

- D/B/jet reco and gen candidates (etc)
- Evt table

step2

Histograms

Dataframes produced on a fraction of the statistics

Analysis optimization

Candidate selection

selection model/criteria

- Optimized Kernels

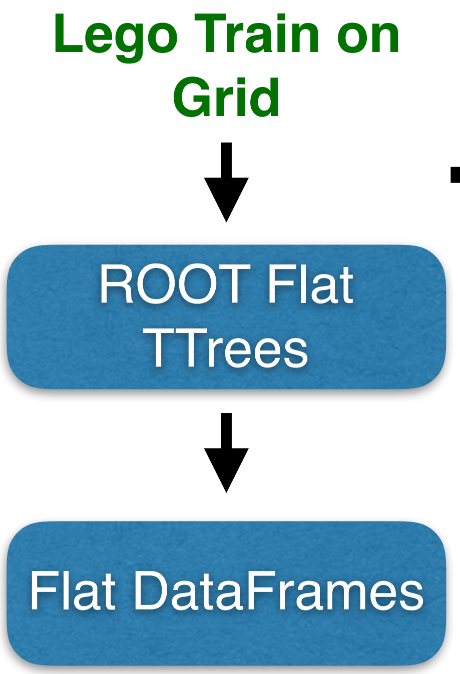
Differences with standard strategy:

- Currently Global event objects created on the fly by each task and displaced track vertices written on disk

- Run on the Grid or in clusters of HPC servers
- GPUs for optimization and fast processing

A working prototype of fast HF analysis on TTrees with 2018 data

Flat table producer



Lego Train on Grid

ROOT Flat TTrees

Flat DataFrames

Ds/Lc reco tables:

- kinem/selection variables
- track/PID variables

Ds/Lc gen tables:

- kinematic variables

Event tables:

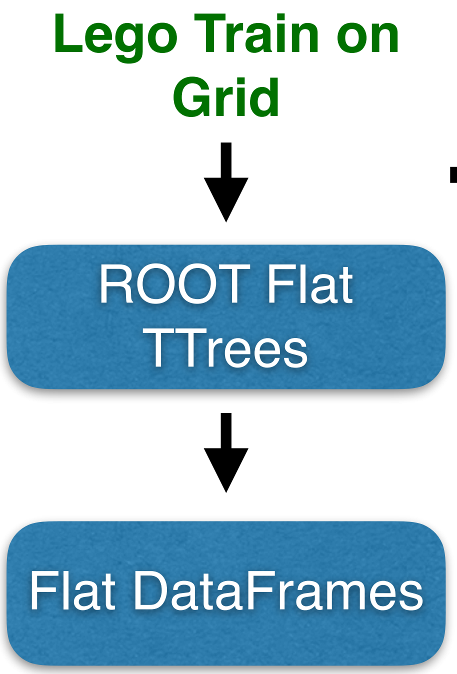
- event selection

LHC18r 0-10%

- AOD ~ 0.9 PBytes
- Trees (Λ_c + Evt) ~ 1 TB with **very** loose cuts and no PID

Samples downloaded on a local servers on SSD disks

Flat table producer



Lego Train on Grid

ROOT Flat TTrees

Flat DataFrames

Ds/Lc reco tables:

- kinem/selection variables
- track/PID variables

Ds/Lc gen tables:

- kinematic variables

Event tables:

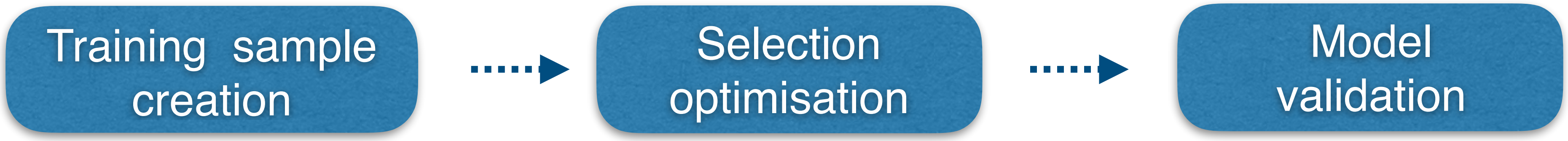
- event selection

LHC18r 0-10%

- AOD ~ 0.9 PBytes
- Trees (Λ_c + Evt) ~ 1 TB with **very** loose cuts and no PID

Samples downloaded on a local servers on SSD disks

Selection optimisation



Training sample creation

- Partial merging
- Signal from MC
- Bkg from data

Selection optimisation

- “Rectangular” optimization
- ML techniques:
  - Binary Trees / Boosting
  - Deep networks
- Grid search:

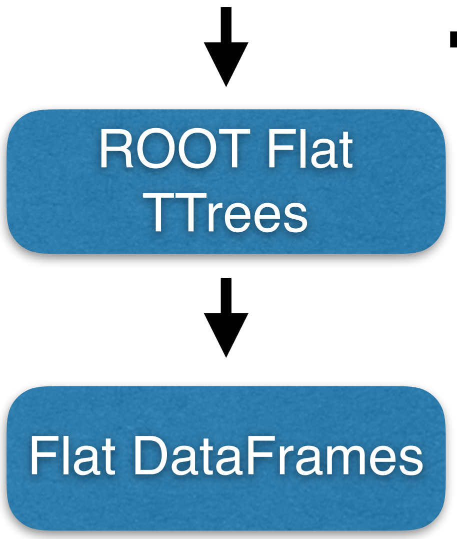
Model validation

- ROC
- Cross validation
- ….
- significance optimisation

- **time training** ~ 10 minutes/p_T bin with GPUs for XGboost or TensorFlow

Flat table producer

Lego Train on Grid



ROOT Flat TTrees

Flat DataFrames

Ds/Lc reco tables:

- kinem/selection variables
- track/PID variables

Ds/Lc gen tables:

- kinematic variables

Event tables:

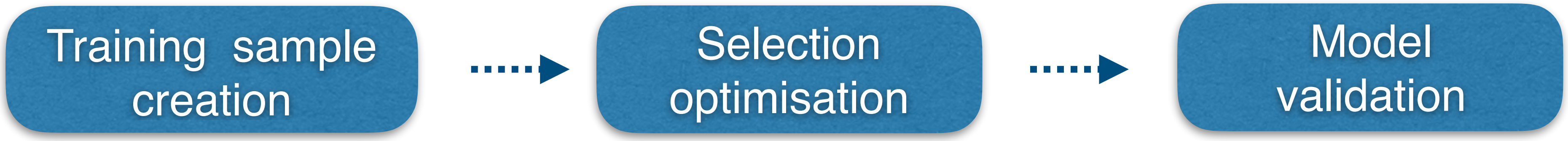
- event selection

LHC18r 0-10%

- AOD ~ 0.9 PBytes
- Trees (Λ_c + Evt) ~ 1 TB with **very** loose cuts and no PID

Samples downloaded on a local servers on SSD disks

Selection optimisation



Training sample creation

Selection optimisation

Model validation

- Partial merging
- Signal from MC
- Bkg from data

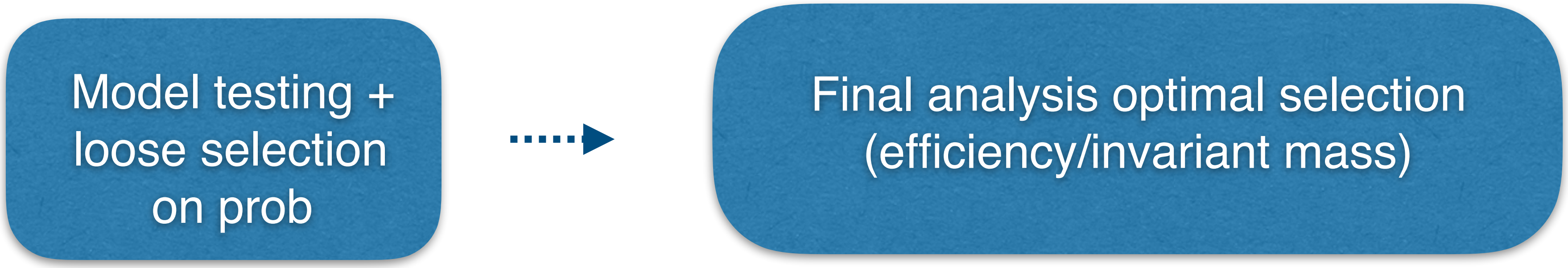
- “Rectangular” optimization
- ML techniques:
  - Binary Trees / Boosting
  - Deep networks
- Grid search:

- ROC
- Cross validation
- ….
- significance optimisation

- **time training** ~ 10 minutes/p_T bin with GPUs for XGboost or TensorFlow

Parallelized analysis

selection model/criteria



Model testing + loose selection on prob

Final analysis optimal selection (efficiency/invariant mass)

- **input** ~ 1 TB
- **time** ~ 10min/pt bin
- multiprocessing on unmerged files

- **size input** ~ 10 GB
- **time** ~ few s/pt bin
- multiprocessing on unmerged files or single core on merged file
- output saved as root files as for old analysis tasks

Software and hardware:

- **analysis package**: https://github.com/ginnocen/MachineLearningHEP written in python + Pandas + XGBoost/Keras with Numba functions for fast processing
- **Server** : 32 cores, 16 TB SDD (+16 coming soon), 350 GB Ram, 1 GPU TESLA V100

Discussion

AOD Run3 content:

- https://docs.google.com/spreadsheets/d/120fJK5vfhyvIKZ94-xEOaDwIN2H1Mu4iVPUOwEgCTyo/edit#gid=0
- **PWGs agree with the current content? Feedback needed.**

Traditional loop strategy:

```
for track in tracks:
 if trackpt>1:
     histo->Fill(trackpt)
```

- More flexible
- Very similar to current strategy

Declarative strategy:

```
.Filter(“trackpt>1).Histo1D(“trackpt)
```

- Compact
- Optimised for parallization and multi-threading

Timing and performances:

- Critical to benchmark the timing performance of the analysis structure with DPL in a real case analysis
    - compression?
    - parallelization and concurrency strategy?

Ongoing activities

Perform a complete analysis starting with the Run3 framework:
→ test the AOD format
→ test the arrow/message passing strategy
→ first look at timing/performance, compression and concurrency organization

- Run2 → Run3 AOD conversion (DONE)

- Secondary vertex reconstruction with the Table double/triple looping strategy

- Global observable object creation (e.g. simplified centrality)

- Create a first template of **L_c/D_0** and **2-particle correlation task** using declarative analysis
  - including both candidate TTree creation and histograms

Timeline:

→ A first working “complete” flow will be ready by November 2019
→ In November, a working system will be shared with PWGs for more estensive validation with more use cases