SCHOOL OF DATA ANALYSIS

LHCb trigger streams optimization

D. Derkach[1 2] N. Kazeev[2] R. Neychev[3 2] A. Panin[2] I. Trofimov[4] A. Ustyuzhanin[1 2 3] M. Vesterinen[5]

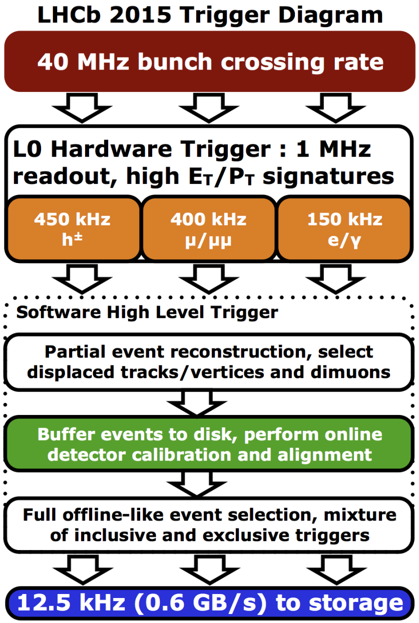
[1] National Research University Higher School of Economics (HSE) [2] Yandex School of Data Analysis [3] Moscow Institute of Physics and Technology [4] Yandex Data Factory [5] Ruprecht-Karls-Universitaet Physikalisches Institut

LHCb data processing

A planned streaming workflow:

- The entire trigger output is split up into different streams
- The streams feed the subsequent processing. In some cases this will be Stripping and in others (pure Turbo), it will be directly used in user jobs for NTuple making.
- Implementation for Turbo is in progress by the LHCb HLT team.

More on Turbo: https://arxiv.org/abs/1604.05596


LHCb 2015 Trigger Diagram
40 MHz bunch crossing rate
L0 Hardware Trigger : 1 MHz readout, high E_T/P_T signatures
450 kHz $h^\pm$
400 kHz $\mu/\mu\mu$
150 kHz e/γ
Software High Level Trigger
Partial event reconstruction, select displaced tracks/vertices and dimuons
Buffer events to disk, perform online detector calibration and alignment
Full offline-like event selection, mixture of inclusive and exclusive triggers
12.5 kHz (0.6 GB/s) to storage

Streams - the formal definition

- There is a number of lines looking for particular types of events
- An event is selected if it passes at least one line
- Lines are grouped into streams
- Each event (wholy or partially - depending on the lines) is copied to all the streams its lines belong

Streams composition matters

- User jobs can only be launched on a whole stream and have to the read the unneeded events
- If an event belongs to several streams, it's duplicated, degree depends on the lines
- There is a limit on the total number of streams from the file management
- The boundary cases are: all in one stream (best space) and each line in its own stream (best time)

User jobs execution time model

Assuming each stream would be read as many times as lines are in it and the time of each reading would be proportional to the number of events read, the total reading time would be proportional to:

$$T = \sum_{\text{stream}} N_{\text{events in stream}} \cdot N_{\text{lines in stream}} \tag{1}$$

Streams are clusters, aren't they?

- Tried clustering algorithms from scikit-learn with sparse input (KMeans, SpectralClustering, Birch, AffinityPropagation). Failed to beat the baseline for reasonable number of streams. See also my talk at ACAT-2016.
- Decided we can do better:
  - Optimize T directly instead of some cluster goodness function
  - Allow for different cost functions
  - Reasonable execution time (no brute force)

Continuous loss

During optimization instead of assigning the lines to streams, let's assume each line has a probability to be in each stream: L_{ls}, l-th line in the s-th stream.
Let $\Delta_{el} \in \{0, 1\}$ be the indicator whether event e was selected by line l. Then:

$$E\left[N_{\text{lines in stream}}\right] = \sum_{l} L_{ls} \tag{2}$$

$$E\left[N_{\text{events in stream } s}\right] = \sum_{e} \left(1 - \prod_{l} \left(1 - \Delta_{el} L_{ls}\right)\right) \tag{3}$$

After optimization we assign each line to the stream with highest probability to contain it.

The approximated T

$$\tilde{T} = \sum_s E\left[N_{\text{lines in stream } s}\right] \cdot E\left[N_{\text{events in stream } s}\right] \tag{4}$$

$$= \sum_s \left[\sum_l L_{ls} \cdot \sum_e \left(1 - \prod_l (1 - \Delta_{el} L_{ls})\right)\right] \tag{5}$$

In general, $\tilde{T} \neq E[T]$. However, if all the assignments are definite $L_{ls} \in \{0, 1\}$, $\tilde{T} = T$.

Solving the boundary conditions

L_{ls} are probabilities so

- $L_{ls} \in [0, 1]$
- A line must be on average assigned to a stream, so $\sum_s L_{ls} = 1$

Solving the boundary conditions

L_{ls} are probabilities so

- $L_{ls} \in [0, 1]$
- A line must be on average assigned to a stream, so $\sum_s L_{ls} = 1$

Let's parameterise L_{ls}:

$$L_{ls} = \frac{e^{A_{ls}}}{\sum_s e^{A_{ls}}}. \tag{6}$$

This way A_{ls} can have any value. This trick is from deep learning and is called softmax.

Optimizing [Theano advertisement]

`http://deeplearning.net/software/theano/`

> Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

- Easy-to-use Python interface
- Fast evaluation: the expressions are put into C code and compiled
- Transparent GPU support. Even faster evaluation.
- Symbolic differentiation — Theano does your derivatives

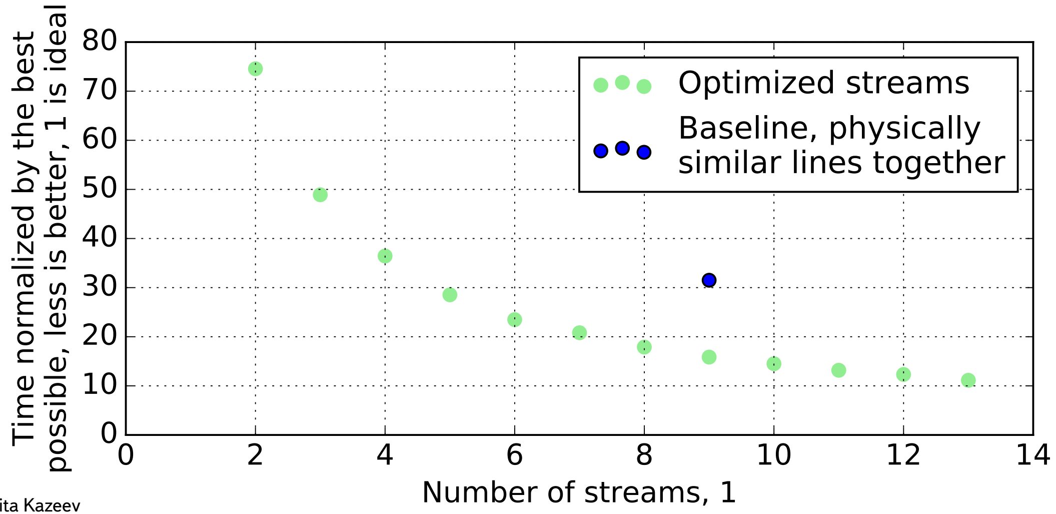
Stochastic gradient optimization

- In T there is a sum over all the events $\sum_e$.
- Evaluating over all of them is too CPU consuming.
- Solution: stochastic gradient optimization
  - Don't evaluate the sum over all the events
  - Split events into batches, calculate the gradient on it, make a descent step.
  - Take the next batch, repeat until convergence.

Experiment setup

- Using a sample of $2 \cdot 10^5$ LHCb events.
- Comparing the optimized streams to the baseline where the lines are grouped by physical similarity:
  `https://gitlab.cern.ch/lhcb-core/HLTRateScripts/blob/master/scripts/StreamDefs.py`

Optimization results

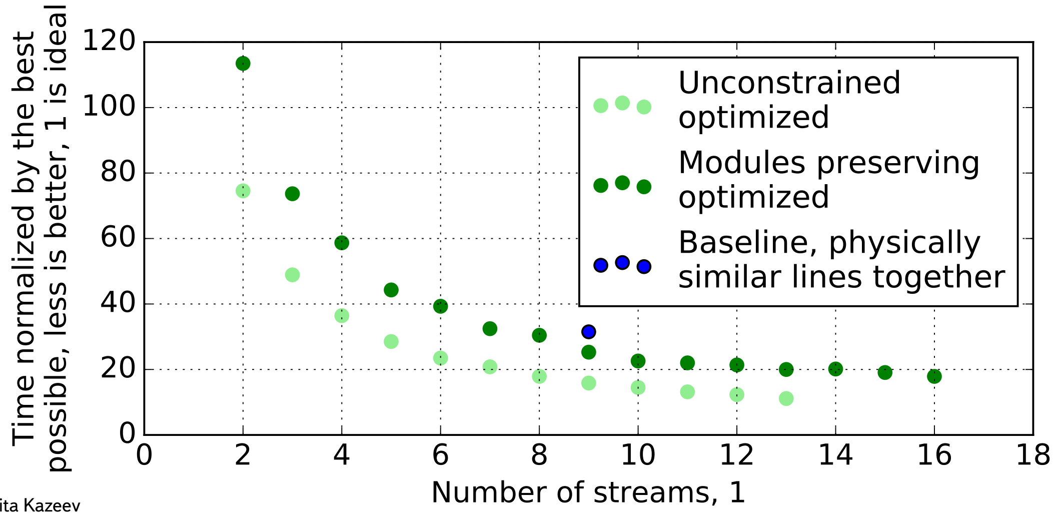


More constraints

Some lines should go together as they are often used together.
We took the HLT modules and adjusted Δ and L: $\tilde{\Delta}_{em}$ as indicator whether in module m any line has selected event e and $\tilde{L}_{ms}$ as the probability of module m to be in stream s and M_m as the number of lines in a module.

$$\tilde{\tilde{T}} = \sum_s \left[\sum_m \left(M_m \tilde{L}_{ms} \right) \cdot \sum_e \left(1 - \prod_m \left(1 - \tilde{\Delta}_{el} \tilde{L}_{ms} \right) \right) \right] \tag{7}$$

Constrained optimization result

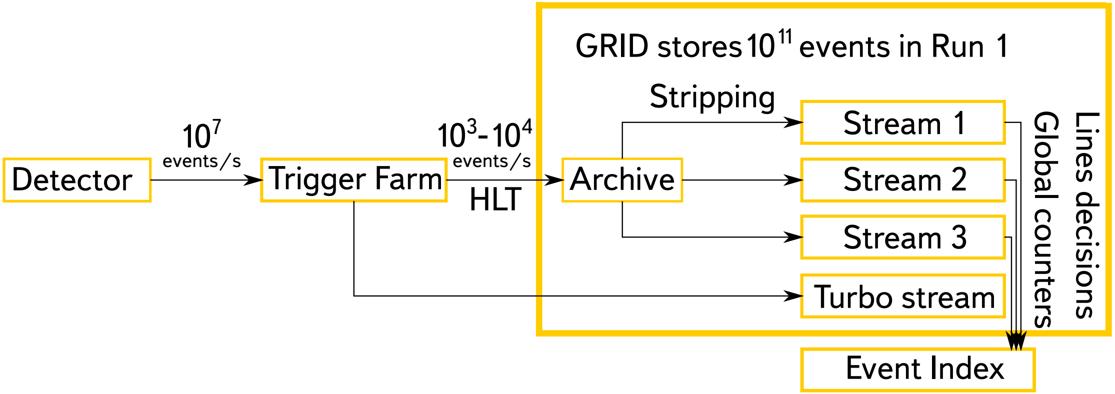


Summary

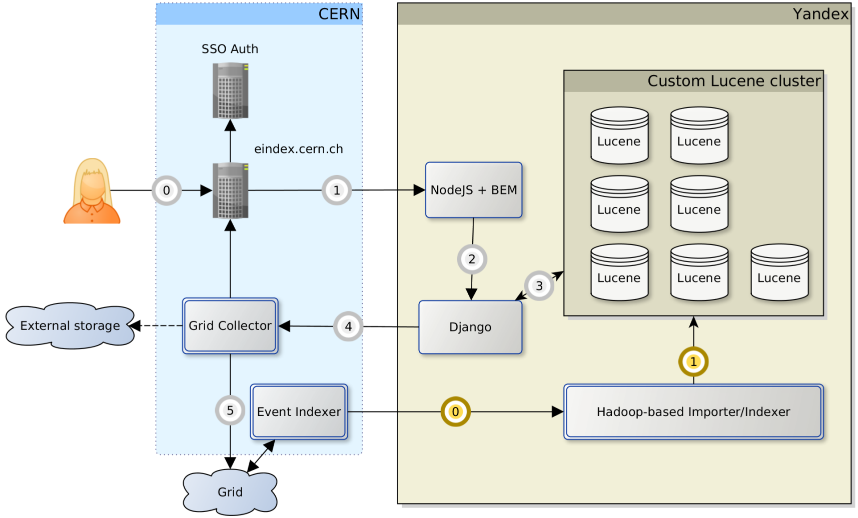
- A method was developed for finding the optimal streams composition. It is flexible and can be used for different cost functions and numbers of streams.
- For the data after the trigger, it is possible for the same number of streams to decrease the disk reading time of the analysis jobs by 20% while maintaining the lines groupings and by 50% while not.

Backup

LHCb pipeline



Event Index


CERN
SSO Auth
eindex.cern.ch
0
1
External storage
Grid Collector
4
5
Event Indexer
Grid
Yandex
Custom Lucene cluster
Lucene
Lucene
Lucene
Lucene
Lucene
Lucene
Lucene
NodeJS + BEM
2
3
Django
1
0
Hadoop-based Importer/Indexer