LHCb streaming mixed optimization

N. Kazeev[23] R. Neychev[12] A. Ustyuzhanin[123]

[1]MIPT [2]Yandex School of Data Analysis [3]HSE

May 16, 2017

9th LHCb computing workshop

Streams as a trade-off

- User jobs can only be launched on a whole stream and have to read the unneeded events
  - The smaller the streams are, the faster the jobs run
- If an event belongs to several streams, the information is duplicated
  - The fewer streams there are, the less space is occupied
- Number of streams should not be too large
- Boundary cases:
  - Best space — all lines in one stream
  - Best time — each line owns its personal stream

Problem statement

Goal

Decrease both occupied space and computation time with some trade-off with focus on space.

Conditions:

- Around $n_{\mathrm{DST}} = 400$ DST and $n_{\mathrm{mDST}} = 1500$ mDST lines
- Dataset: Stripping 28 Validation
- DST and mDST streams **must not** intersect
- Number of streams should be considered as the solution parameter

Due to the independence of DST and mDST lines their optimizations are threatened as independent sub-problems.

Previous and proposed solutions

Previous solution (presented on LHCb-week (March 2016))

- Based on lines clustering.
- Required using 93 streams — *too many streams*.

Suggested solution

- Based on time- and space metrics relaxation.
- Parametrized with number of streams.

The similar approach has already been applied to minimize reading time for **Turbostream**. It allowed to reduce the reading time by $20\% - 50\%$ with the same space usage.

Time and space models

Stream is read as many times as many lines are in it. Every line is assumed to be accessed with frequency p_{line}. Time of each reading is proportional to the number of events read. The time model (T score):

$$T = \sum_{\text{stream}} N_{\text{events in stream}} \cdot \left[\sum_{\text{lines}} p_{\text{line}} \right]$$

Occupied by stream space is proportional to the number of events stored. The storage model (S score):

$$S = \sum_{\text{stream}} N_{\text{events in stream}}$$

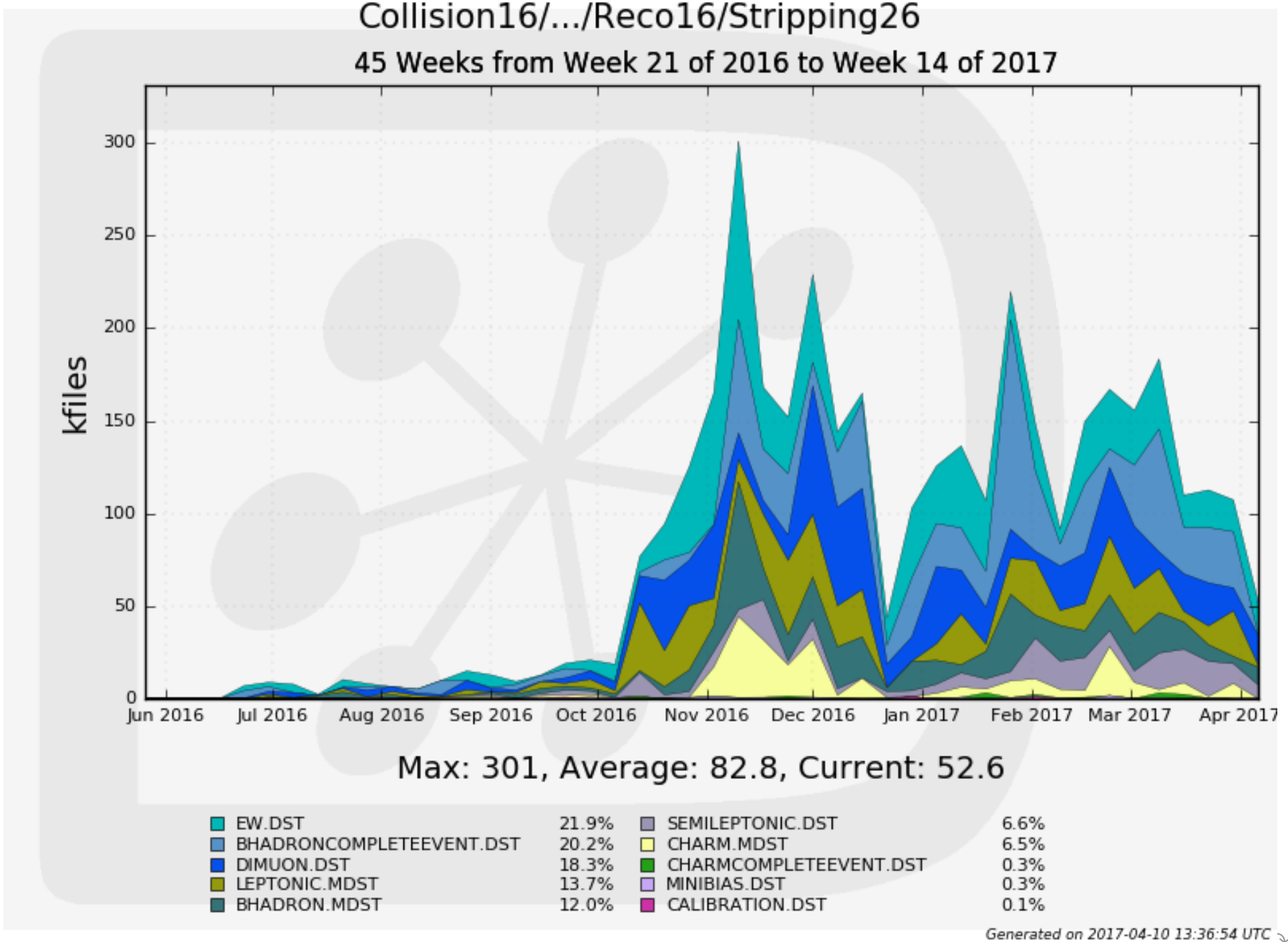
To decrease the occupied storage and keep the time complexity, the *mixed* optimization is used. The function to be optimized:

$$\text{Combined score} = \alpha \cdot T + (1 - \alpha) \cdot S$$

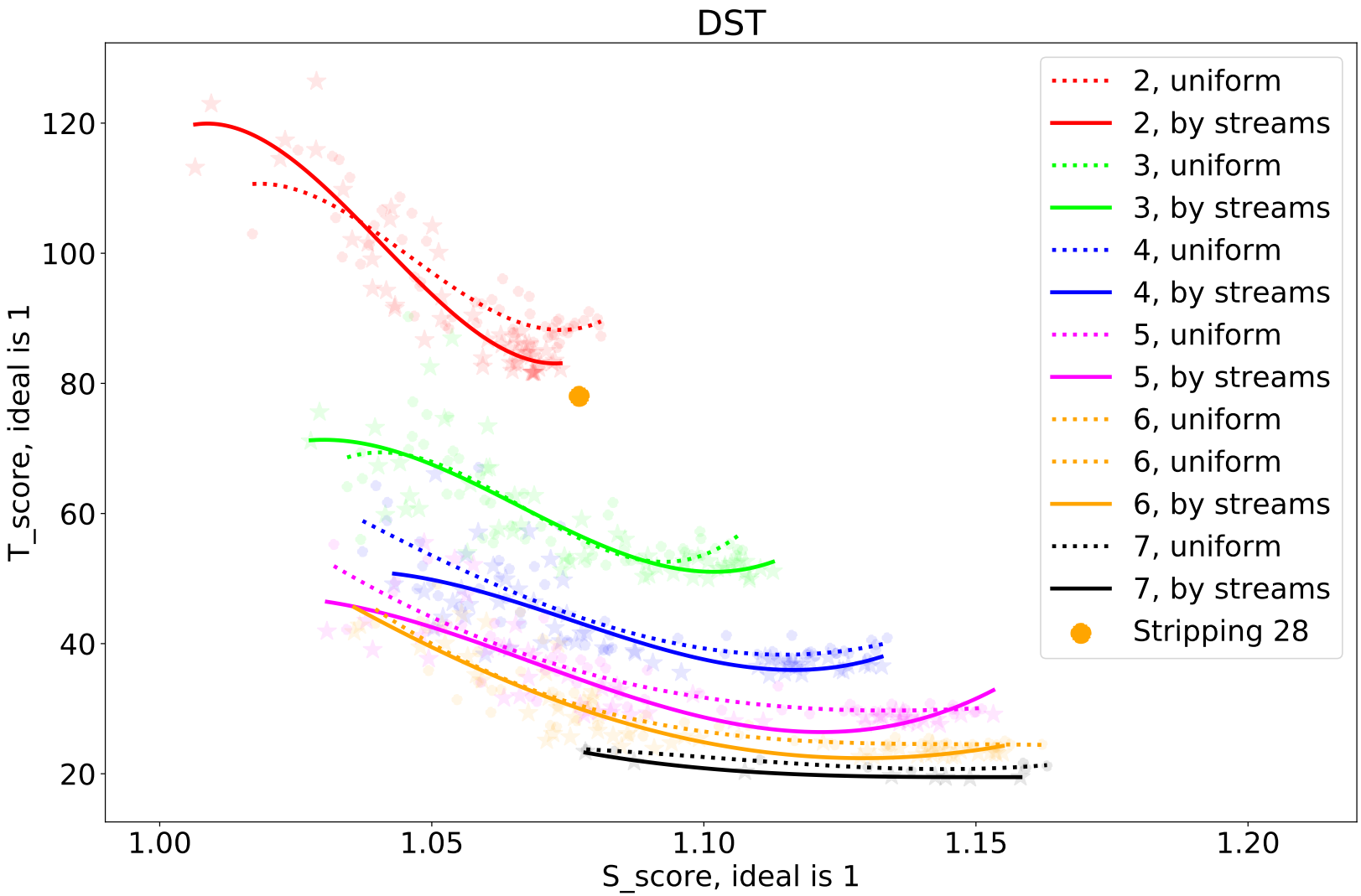
Lines popularity

Lines access frequency is estimated depending on lines popularity model. Currently, two models were used and compared:

- Uniform model.
  - All lines are accessed with the same frequency.
- Streams-corresponding model.
  - According to the statistics for Stripping 26, streams are accessed with different frequencies. All lines within stream are assumed to be accessed with the same frequency: $p_{\text{line}} = \frac{\text{stream access frequency}}{N \text{ lines in stream}}$.

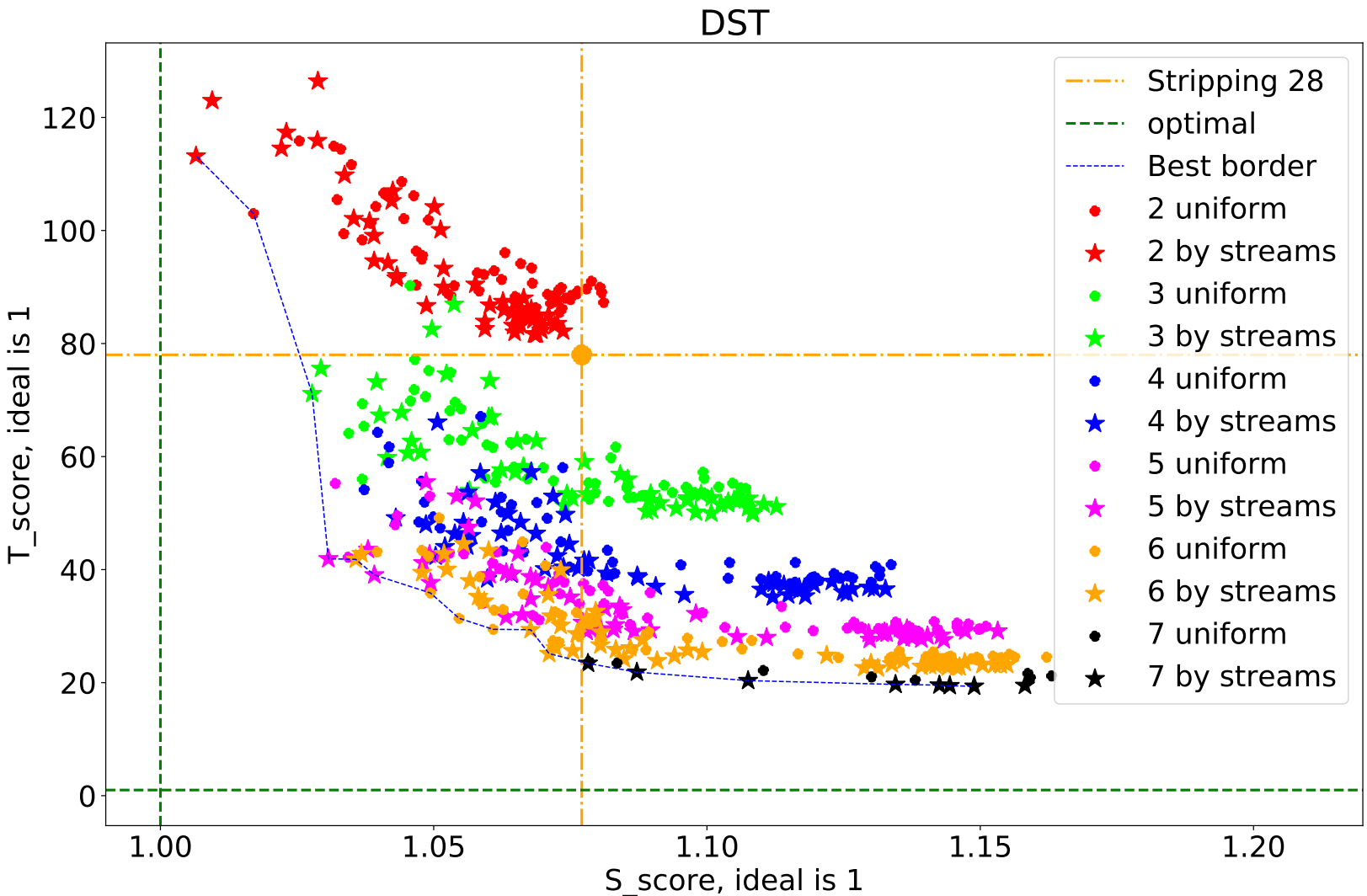

Collision16/.../Reco16/Stripping26
45 Weeks from Week 21 of 2016 to Week 14 of 2017
kfiles
300
250
200
150
100
50
0
Jun 2016
Jul 2016
Aug 2016
Sep 2016
Oct 2016
Nov 2016
Dec 2016
Jan 2017
Feb 2017
Mar 2017
Apr 2017
Max: 301, Average: 82.8, Current: 52.6
EW.DST 21.9%
BHADRONCOMPLETEEVENT.DST 20.2%
DIMUON.DST 18.3%
LEPTONIC.MDST 13.7%
BHADRON.MDST 12.0%
SEMILEPTONIC.DST 6.6%
CHARM.MDST 6.5%
CHARMCOMPLETEEVENT.DST 0.3%
MINIBIAS.DST 0.3%
CALIBRATION.DST 0.1%
Generated on 2017-04-10 13:36:54 UTC

Lines popularity models comparison

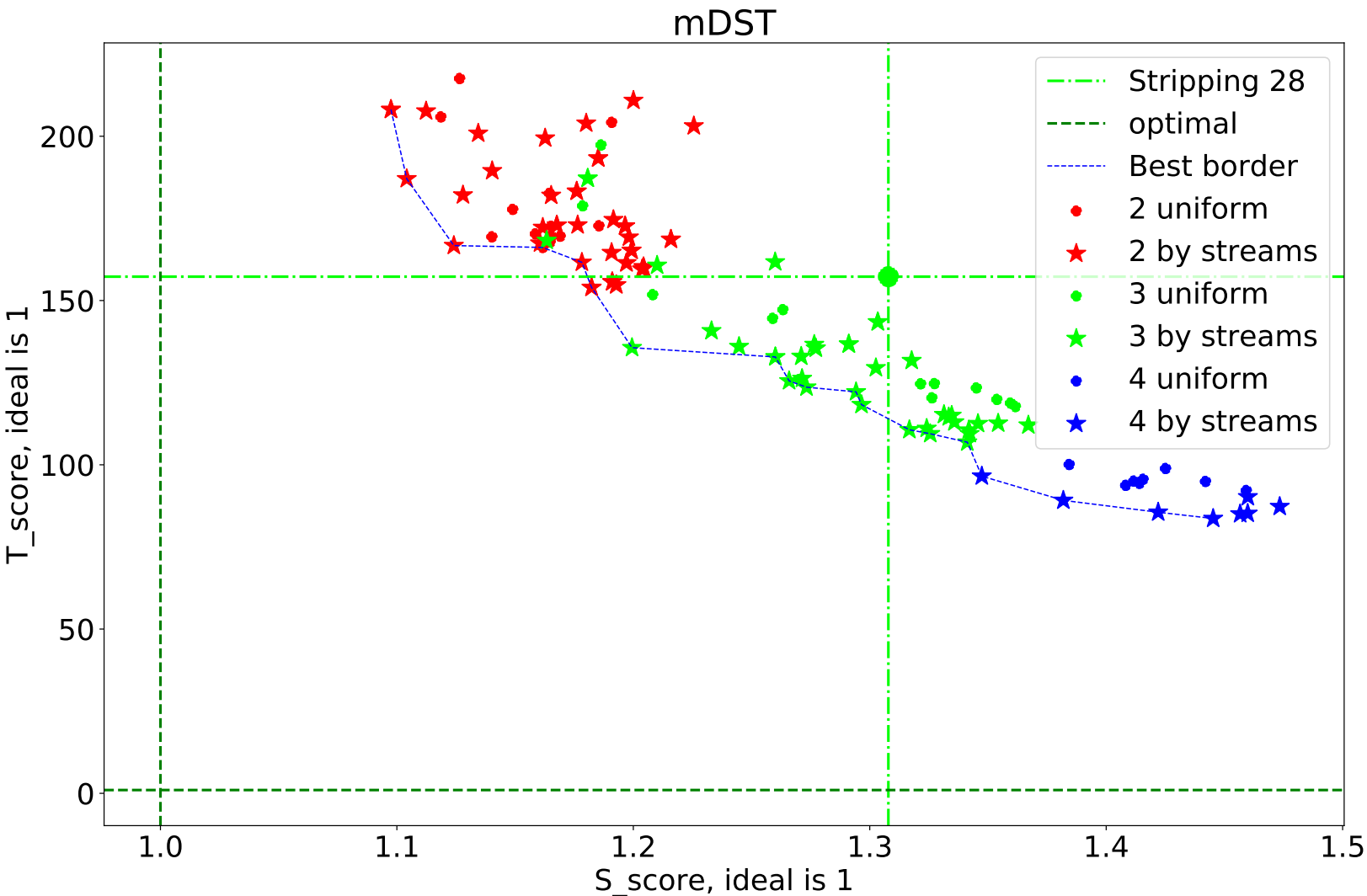


Lines shows mean T score for fixed S score; α increases from left to the right in range $[0.0005; 0.64]$

Optimization results on DST data



Optimization results on mDST data



N	T score	S score	Sim
2	113.177	1.007	0.81
2	102.983	1.017	0.773
3	71.124	1.028	0.566
5	41.913	1.031	0.439
6	41.816	1.036	0.326
5	39.058	1.039	0.354
6	35.866	1.049	0.29
6	31.389	1.055	0.321
6	29.436	1.061	0.292
6	29.37	1.068	0.267
6	25.131	1.071	0.319
6	78.006	1.077	1

Table: DST best results

N	T score	S score	Sim
2	208.134	1.097	0.522
2	187.046	1.104	0.556
2	166.726	1.124	0.615
2	166.157	1.162	0.768
2	161.614	1.178	0.671
2	153.983	1.182	0.694
3	135.647	1.199	0.388
3	132.821	1.26	0.48
3	125.535	1.266	0.412
3	123.621	1.273	0.696
3	122.198	1.294	0.477
3	118.307	1.297	0.435
3	157.259	1.308	1

Table: mDST best results

N is number of streams, Sim is similarity to Stripping 28 definitions. T and S scores are normalized by Stripping 28 corresponding scores. Last row is Stripping 28.

Summary

- An optimization method parametrized with number of streams was developed.
- Relaxation of both T-score and S-score allowed to decrease IO time and occupied space simultaneously.
  - DST data: up to 6.5% S score decrease, up to 67% T score decrease.
  - mDST data: up to 16% S score decrease, up to 24% T score decrease.
- Both considered line popularity models show comparable results.

Framework overview

The existing framework allows to optimize any relaxed functional using the provided data. Currently available:

- T-score
- S-score
- Combined score

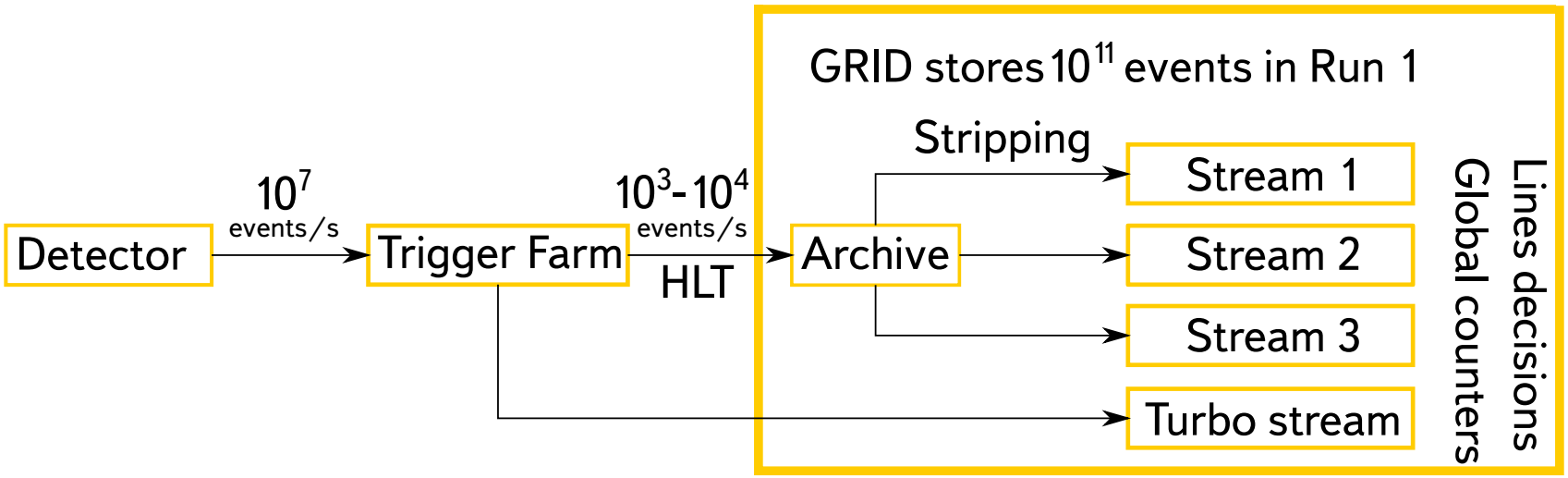
Useful features:

- Modules constrained optimization
- Prescale support
- Line popularities support

Available with demo at branch *demonstration_Stripping_28*
`https://gitlab.cern.ch/YSDA/streams-optimization`

Backup

LHCb processing pipeline



Time and space models

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:

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

Storage model

According to the current finginds, all lines correspond to *sum* G_S or *indicator* G_I groups. Event weight in stream is computed as follows:

$$w_{\text{event-stream}} = \sum_{g_I \in G_I} \text{Ind}(\text{event has line from } g_I) \cdot W_{g_I} + \sum_{g_S \in G_S} N_{\text{lines from } g_S \text{ than contain event}} \cdot W_{g_S}, \quad (1)$$

where W_g is a group "weight", Ind is an indicator function. Therefore, the general formula for storage:

$$S = \sum_{\text{stream}} \left(\sum_{\text{event} \in \text{stream}} w_{\text{event-stream}} \right)$$

Time model relaxation

Instead of assigning the lines to streams, 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:

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

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

The approximated T

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

$$= \sum_{s} \left[\sum_{l} L_{ls} \cdot \sum_{e} \left(1 - \prod_{l} (1 - \Delta_{el} L_{ls}) \right) \right] \tag{5}$$

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

Time and space models

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:

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

There are several *groups* of lines with different event weights W_g. So, the space can be estimate as:

$$S = \sum_{\text{stream}} \left[\sum_{\text{event} \in \text{stream}} \left(\sum_{g \in groups} \text{Ind}(\text{event has line from } g) \cdot W_g \right) \right]$$

Time and space models relaxation

Instead of assigning the lines to streams, 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 and $\Delta_{lg} \in \{0, 1\}$ likewise for line-group. Then:

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

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

$$\tilde{S} = \sum_{\mathrm{s}} \Big[\sum_{\text{event} \in \mathrm{s}} \Big(\sum_{g_l \in G_l} W_{g_l} \cdot \big(1 - \prod_l (1 - \Delta_{el} L_{ls} \Delta_{lg_l})\big) \Big) \Big]$$

Storage model relaxation

Let $\Delta_{lg} \in \{0, 1\}$ be the indicator whether line l is in group g. As for the time model:

$$\mathrm{P}(\text{event has line from } g_l) = 1 - \prod_l (1 - \Delta_{el} L_{ls} \Delta_{lg})$$

$$\mathrm{E}(N_{\text{lines from } g \text{ than contain event}}) = \sum_l (\Delta_{el} L_{ls} \Delta_{lg})$$

Therefore

$$\tilde{S} = \sum_{\text{s}} \Big[\sum_{\text{event} \in \text{s}} \Big(\sum_{g_l \in G_l} W_{g_l} \cdot (1 - \prod_l (1 - \Delta_{el} L_{ls} \Delta_{lg_l})) + \sum_{g_S \in G_S} W_{g_l} \cdot \sum_l (\Delta_{el} L_{ls} \Delta_{lg_S}) \Big) \Big]$$

Hereinafter call $\tilde{S}$ S-score.

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.

Mixed optimization

To decrease the occupied storage and keep the time complexity, the *mixed* optimization is used. The function to be optimized:

$$\text{Combined loss} = \alpha \cdot T + (1 - \alpha) \cdot S$$

Problems

- In $\tilde{T}$ and $\tilde{S}$ there is a sum over all the events $\sum_e$.
- Evaluating over all of them is too CPU consuming.

Solution: stochastic gradient optimization

- Choose random events batch of fixed size.
- Calculate gradient over the batch, make a descent step.
- Take the next batch, repeat until convergence.

Theano ad

`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