Yandex

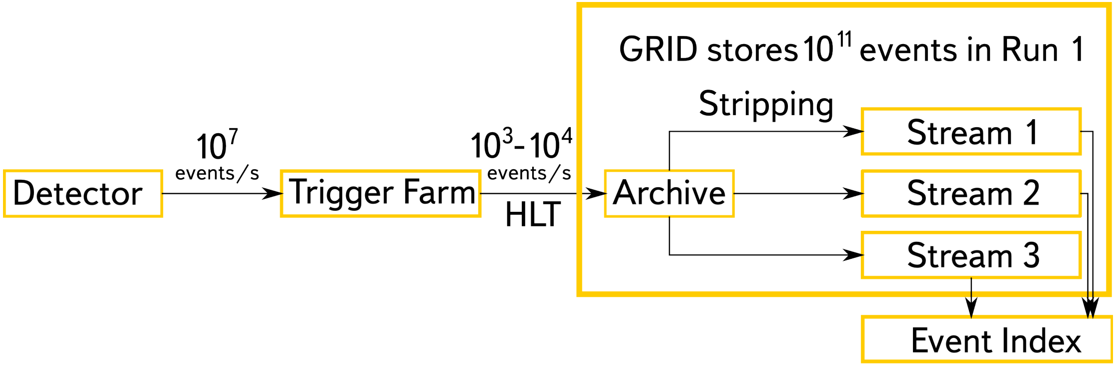
SCHOOL OF DATA ANALYSIS

LHCb data processing optimization using Event Index

A. Redkin[1] N. A. Kazeev[234] A. E. Ustyuzhanin[234] Ilya Trofimov[1]

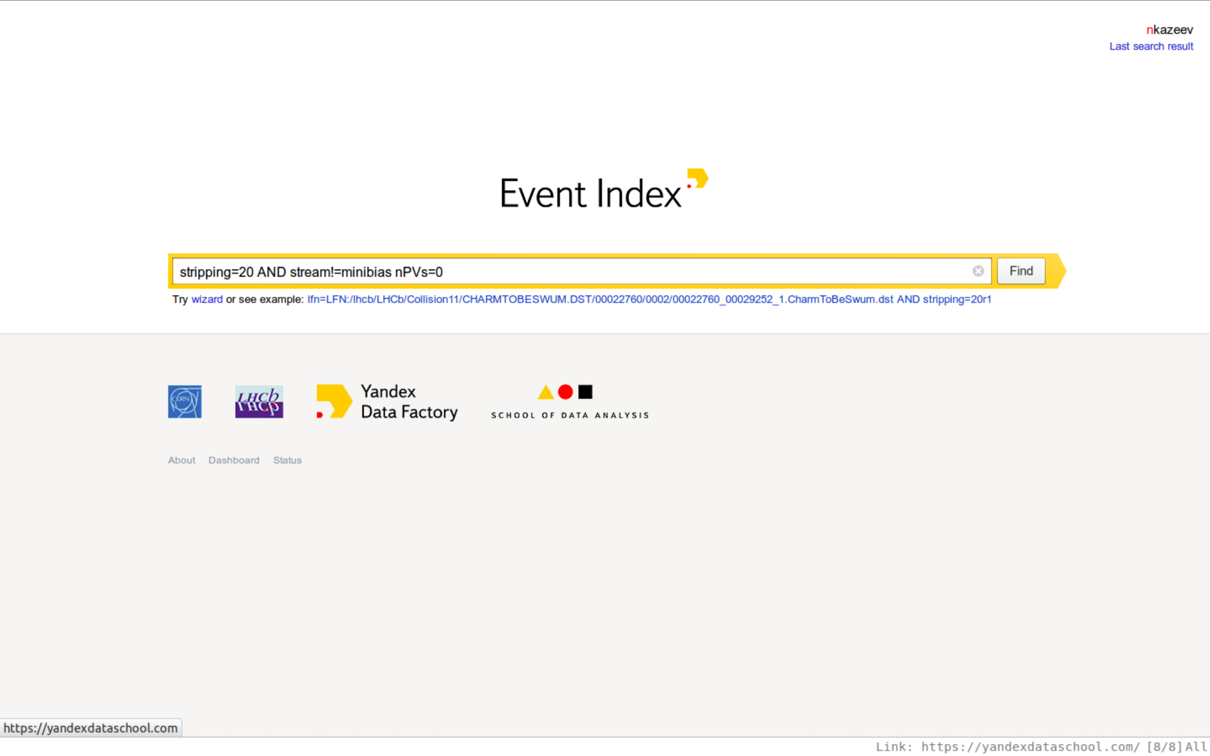
[1]Yandex Data Factory [2]MIPT [3]Yandex School of Data Analysis [4]Kurchatov Institute

LHCb processing pipeline



Streams

- There is number of programs looking for particular physical phenomena called stripping lines
- Event is selected if it passes at least one stripping line
- Stripping lines are grouped in streams
- Event is copied to all the streams its stripping lines belong



nkazeev
Last search result

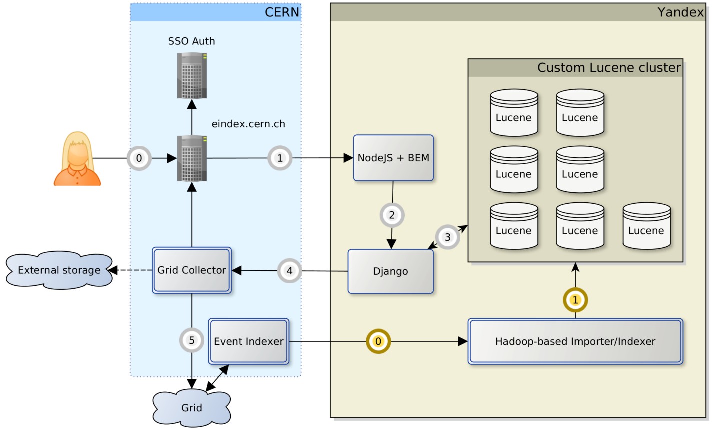
Event Index

stripping=20 AND stream!=minibias nPVs=0

Try wizard or see example: lfn=LFN:/lhcb/LHCb/Collision11/CHARMTOBESWUM.DST/00022760/0002/00022760_00029252_1.CharmToBeSwum.dst AND stripping=20r1

```
Link: https://yandexdataschool.com/ [8/8]All
```

Event Index architecture overview



Streams optimization

- Due to limitations of GRID analysis tasks can only be launched on the whole streams
- Streams are a trade-off between the analyses speed and storage space
- Event Index has access to the event-level stripping lines output and can use that to optimize the distribution of the lines between the streams

Problem setup

Each event would be duplicated as many times as many streams it belongs to. So that the total space would be:

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

Assuming the lines are equally popular, each stream would be read as many times as stripping lines are there and each reading would take $N_{\text{events in stream}}$. The total time would be:

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

Problem complexity

- There are around $n = 1600$ stripping lines and 10^{10} events.
- The total number of possible splits is $\sum_{k=1}^{n} \frac{k^n}{k!} \approx 10^{275}$
- The boundary cases are obvious - all in one stream (best space) and each line in its own stream (best time)

Lines similarity metric

- Compute a metric of similarity between the lines. We used modified pointwise mutual information

$$\text{mPMI} = N\frac{P(x,y)}{P(x)P(y)},$$

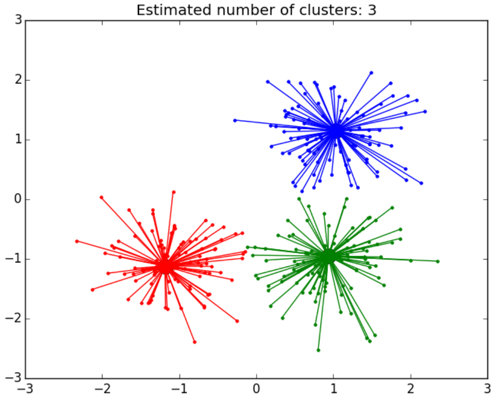
  where $P(x)$ is the probability that line x is present in an event and $P(x,y)$ is the probability that both lines x and y are present, N is the total number of events.
- Use a clustering algorithm to group alike lines together using mPMI estimated from frequency as affinity.

$$a(x,y) = \frac{N(x,y)}{N(x)N(y)},$$

  where $N(x,y)$ is the number of events with both lines x and y present, $N(x)$ is the number of events with line x present.

Clustering - affinity propagation

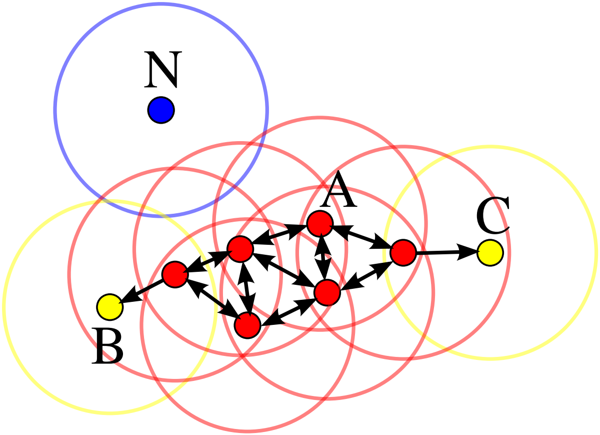
- Creates clusters by sending messages between pairs of samples until convergence
- A dataset is then described using a small number of exemplars, which are identified as those most representative of other samples.
- The messages sent between pairs represent the suitability for one sample to be the exemplar of the other, which is updated in response to the values from other pairs. This updating happens iteratively until convergence.



Brendan J. Frey; Delbert Dueck (2007). "Clustering by passing messages between data points". Science 315 (5814): 972–976. Text: http://scikit-learn.org

Clustering - DBSCAN

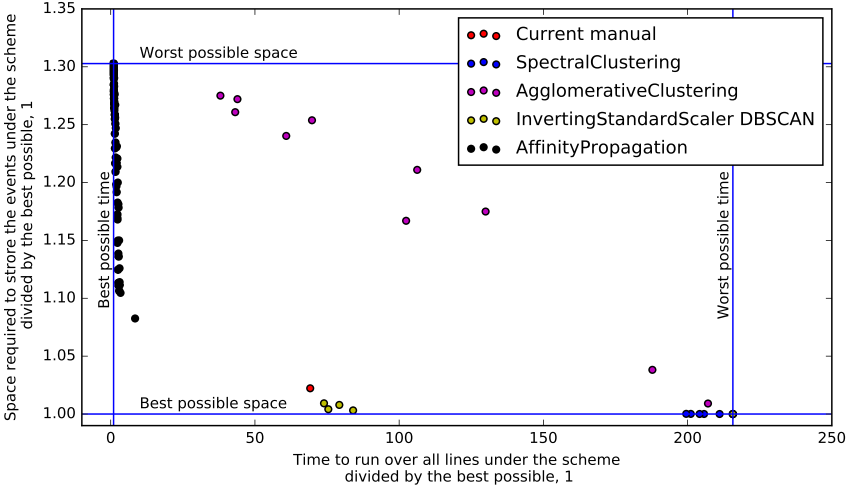
- A point p is a core point if at least minPts points are within distance ϵ of it, and those points are said to be directly reachable from p. No points are directly reachable from a non-core point.
- A point q is reachable from p if there is a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$, where each $p_i + 1$ is directly reachable from p_i.
- All points not reachable from any other point are outliers.



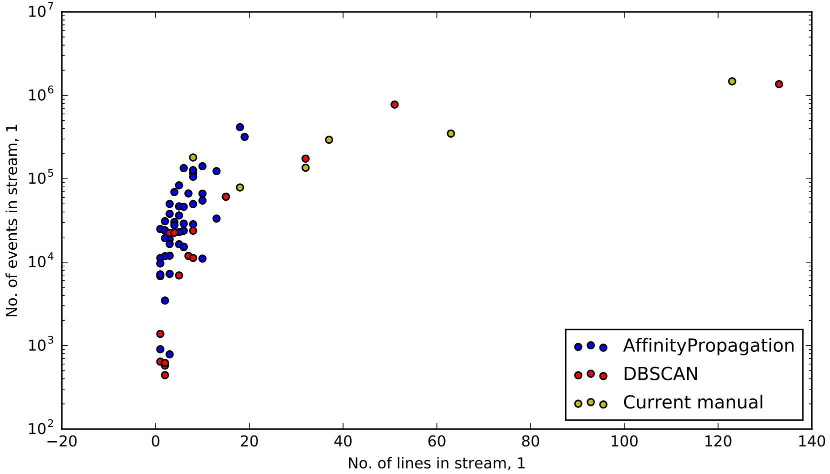
Trial setup

- Took a random sample of $2 \cdot 10^6$ events
- Discarded the lines with no events in the sample
- Computed mPMI affinities
- Computed clusters using different algorithms

Clustering performance



Clusters overview



Summary

- There is little to gain in terms of space (2%)
- The theoritcal maximum speedup is considerable (8000%)
- With clustering we were able to get 10x speedup for a 5% increase in space