Disk storage management for LHCb based on Data Popularity estimator

P. Charpentier[1] M. Hushchyn[2] A. Ustyuzhanin[2,3]

[1]CERN
[2]Yandex School of Data Analysis, RU
[3]NRC "Kurchatov Institute", RU

Presentation outline.

1. Problem statement.
2. Data Popularity Estimation.
3. Data Intensity Prediction.
4. Data Placement Optimization.
5. Results for LHCb data.
6. Comparison with LRU algorithm.

Problem statement.

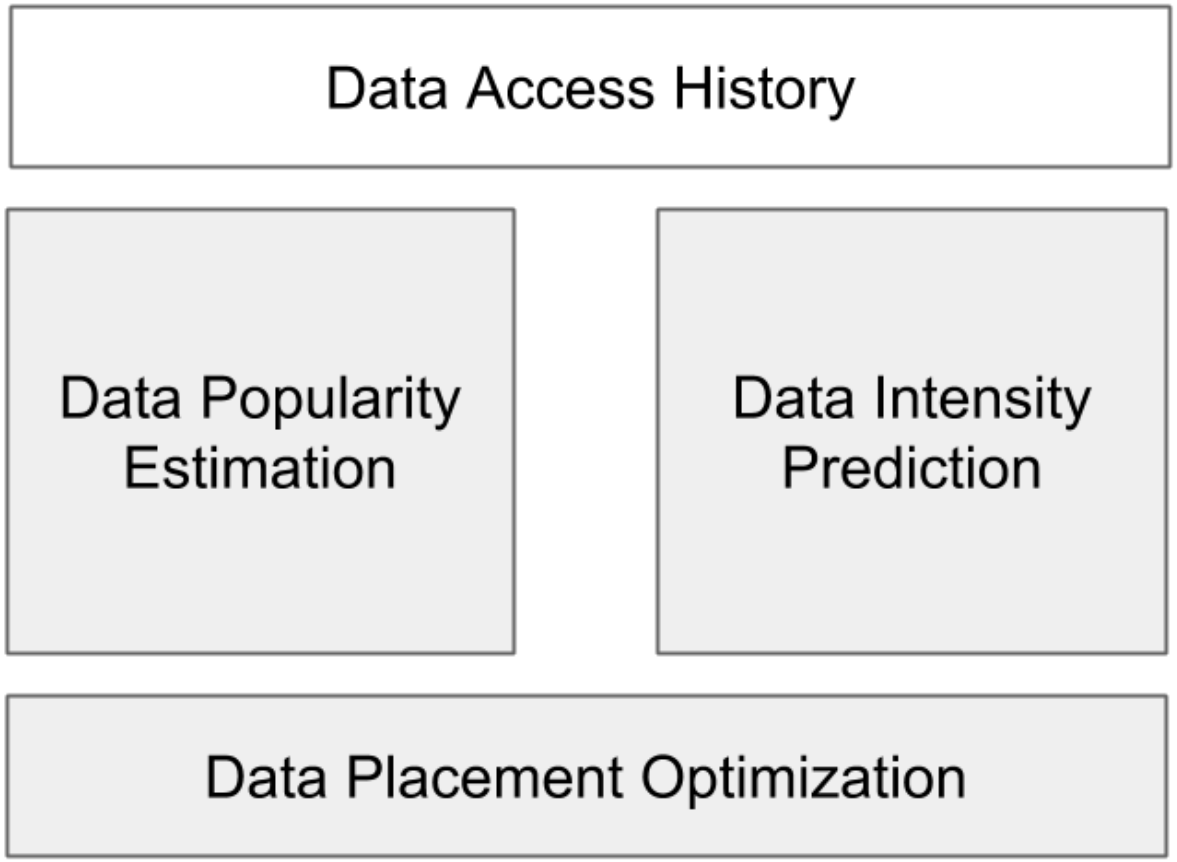
Optimization of the time access to the data

- Using access history of the data sets
- Using two type of storages (disks and types)
- Using costs of storage 1Gb data on storages

Optimization of disk space usage

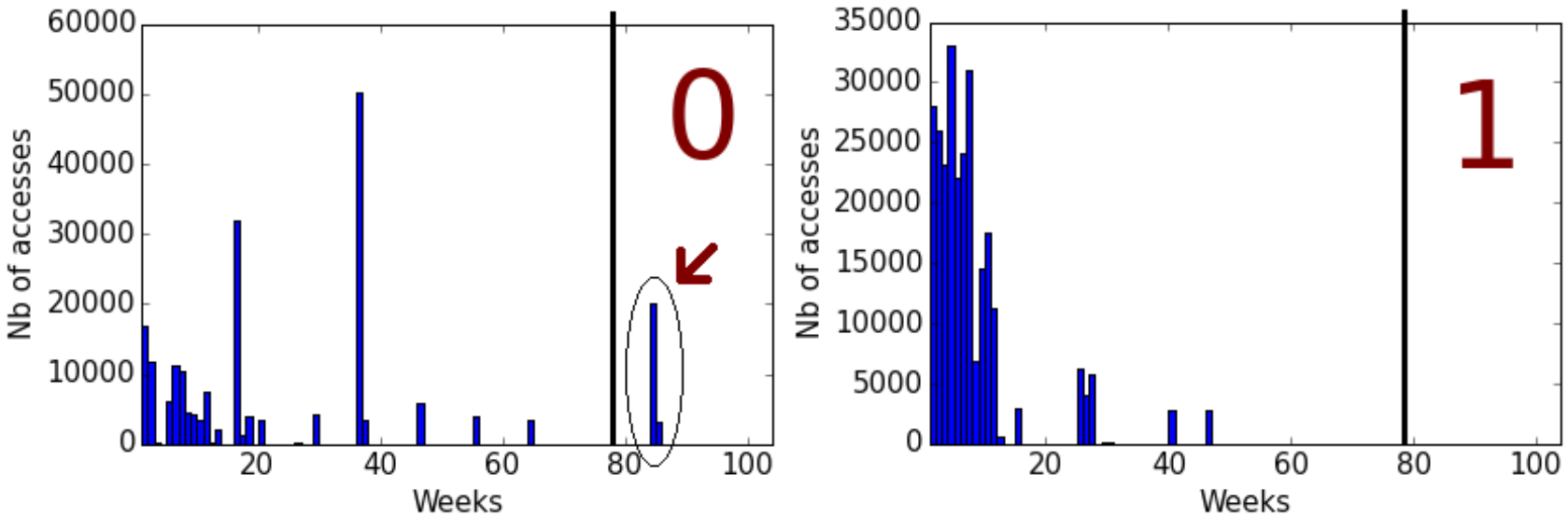
- Using access history of the data sets
- Using two type of storages (disks and types)
- Using number of replicas of the data sets
- Using costs of storage 1Gb data on storages

Stages of the Analysis.



Data Popularity Estimation.
Labels.

While time series of the data history usage are very sparse, last 26 weeks of access history are used to label data.



Data Popularity Estimation.
New features.

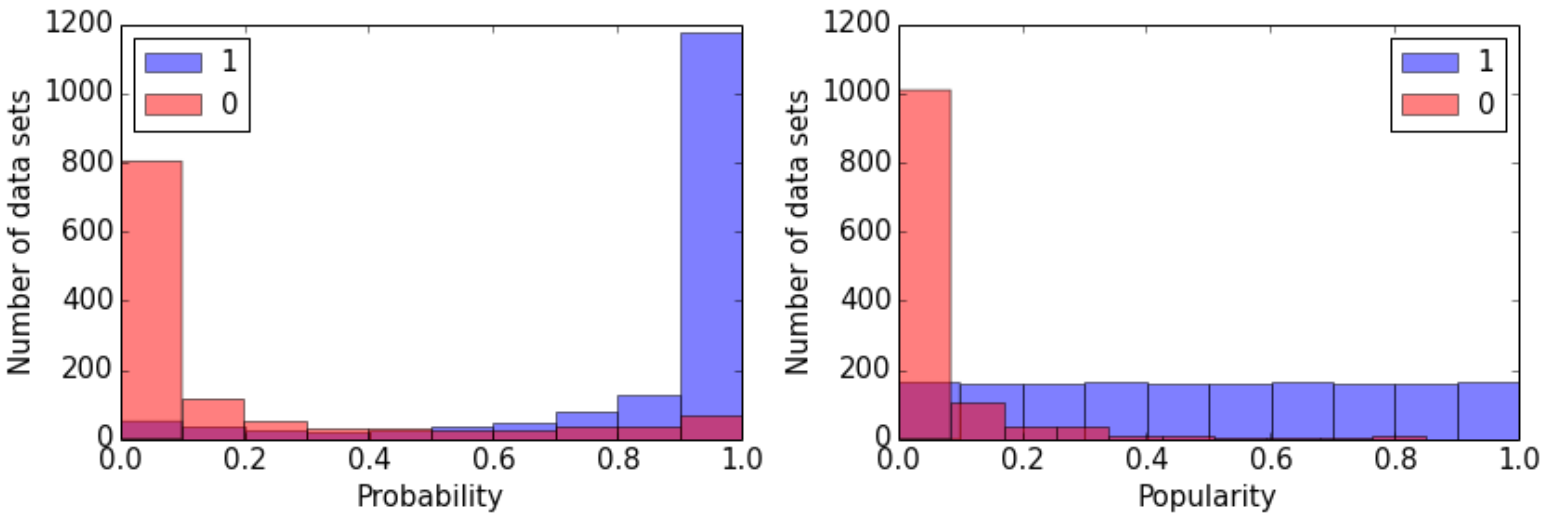
New features were used in analysis as well as existing ones:

- Nb_peaks is the number of week with non-zero number of accesses to a data set.
- Last_zeros is the number of last weeks with zero number of accesses to a data set.
- Inter_max, inter_mean and inter_std are max value, mean value and standart deviation of the number of weeks between neighboring weeks with non-zero number of accesses to a data set accordingly.
- Inter_rel is the ratio of the inter_std and inter_mean values.
- Mass_center is the mass center of a time serie for a data set, where 'mass' is the number of accesses to the data set and 'coordinate' is the week number.
- Mass_center_sqr, mass_moment and r_moment are simillar to mass_center, but 'mass' and 'coordinate' have different degrees.

These features are significantly increase quality of the data popularity estimator. Calculation the features relative creation_week time did not change the estimator's quality. For more details view the data popularity estimator code.

Data Popularity Estimation.
Classifier and Popularity.

- Labeled data with new features is used to train Gradient Boosting Classifier. All data was splited into two equal parts. The classifier was trained on one part of the data and then was used to predict probabilities to have label '1' for another part of the data.
- ROC curve was used to measure the classifier's quality.
- The probability was transformed to the popularity so, that popularity for data sets which have label '1' is uniform. Popularity closer to 1 the higher probability to be useless in future.



Data Intensity Prediction.
Kernel smoothing.

Let's points $(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)$ represent a time serie and $X^l = \{x_1, x_2, ..., x_l\}$. Then, Nadaraya-Watson equation for kernel smoothing:

$$\hat{y}_h(x; X^l) = \frac{\sum_{i=1}^{l} y_i K(\frac{\rho(x,x_i)}{h})}{\sum_{i=1}^{l} K(\frac{\rho(x,x_i)}{h})}, \textit{where} \tag{1}$$

$\hat{y}_h(x; X^l)$ - the time serie value at x after kernel smoothing,
$K(\frac{\rho(x,x_i)}{h}) = exp(-\frac{(x-x_i)^2}{2h^2})$ - RBF smoothing kernel,
h - smoothing window width.

For smoothing window width optimization "Leave-One-Out" method was applied:

$$LOO(h, X^l) = \sum_{i=1}^{l} (\hat{y}_h(x_i; X^l \setminus \{x_i\}) - y_i)^2 \mapsto \min_h \tag{2}$$

Data Intensity Prediction.
Rolling mean.

Let's points $(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)$ represent a time serie after the kernel smoothing. Then, rolling mean values were found:

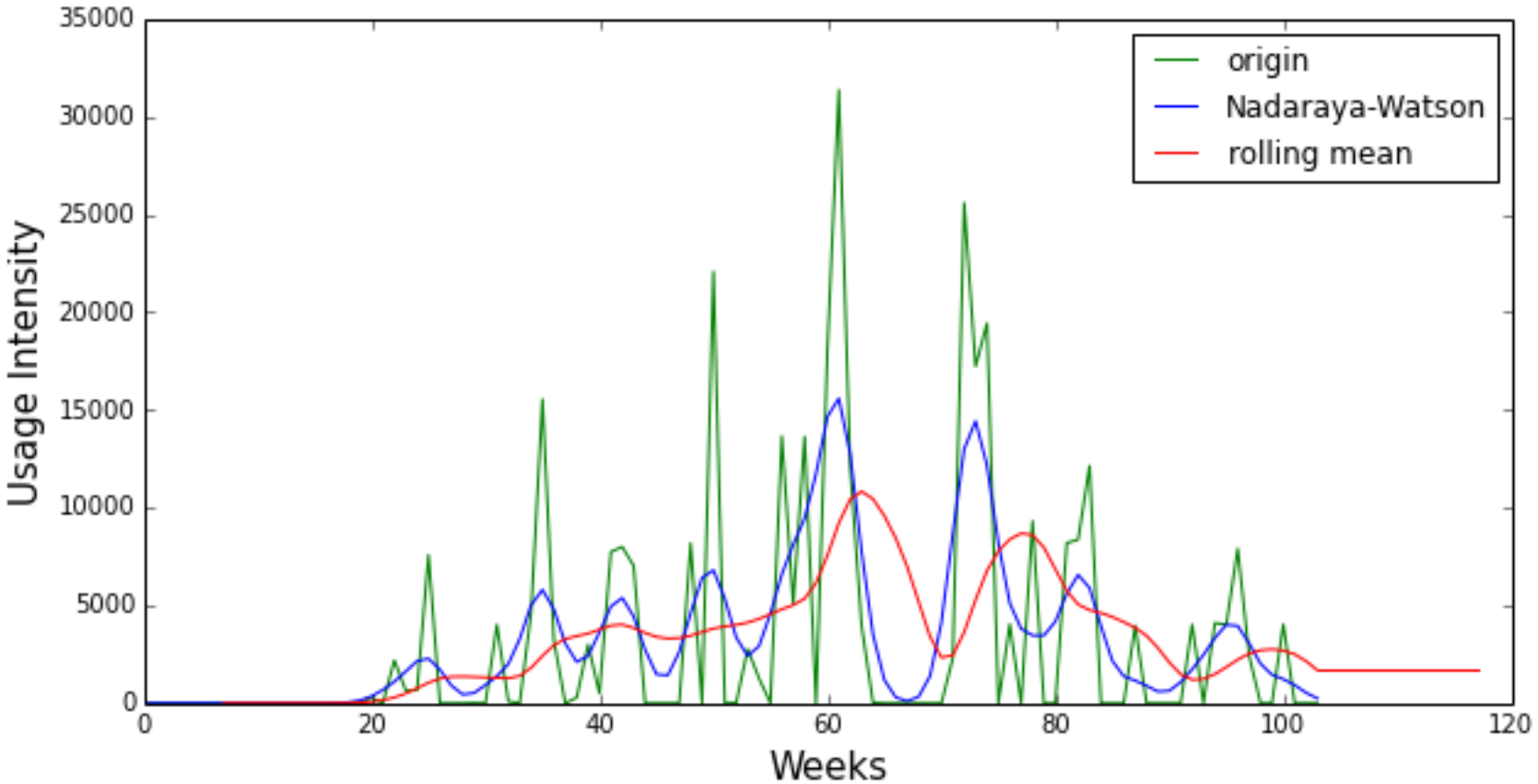
$$\hat{y}_k = \frac{\sum_{i=k-w}^{k} y_i}{w}, where \tag{3}$$

w - size of moving window.

Parameter w was chosen so, that 90% of all time series with equal *nb_peaks* values have *inter_max* values less than w.

Data Intensity Prediction.
Prediction.

When rolling mean values were calculated, prediction was done as constant by the last value of the smoothed time serie:



Data Placement Optimization.
Loss function.

Following loss function was used for data sets placement optimization:

$$L = C_{disk} \sum_{i}^{n} S_i (Rp_i + \alpha \frac{I_i}{Rp_i}) \delta_i + C_{tape} \sum_{i}^{n} S_i (1 - \delta_i) + C_{miss} \sum_{i}^{n} S_i m_i, \tag{4}$$

C_{disk} - cost of storage 1Gb data on disk,
C_{tape} - cost of storage 1Gb data on tape,
C_{miss} - cost of restoring 1Gb data from tape to disk,
α - fine for low number of replicas,
S_i - size of one replica of i^{th} data set,
Rp_i - number of replicas of i^{th} data set,
I_i - predicted usage intensity of i^{th} data set;
δ_i is equal 1 if i^{th} data set on disk, otherwise it is 0;
m_i is equal 1 if i^{th} data set was restored from tape to disk.

Data Placement Optimization. Number of replicas.

The first term of the loss function contains the following expression:

$$Rp_i + \alpha \frac{I_i}{Rp_i} \tag{5}$$

For i^{th} data set was chosen number of replicas $\{1, 2, 3, 4\}$, that minimize the expression above.

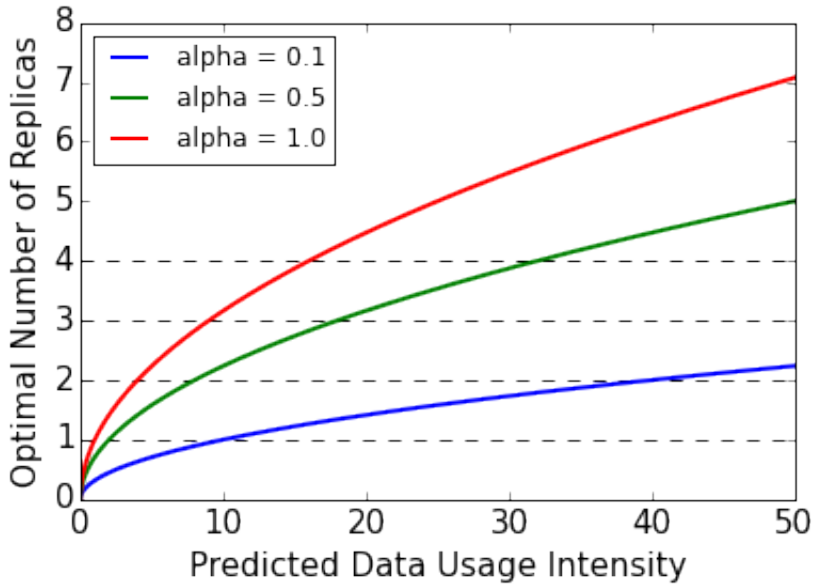
Because of the optimal number of replicas is

$$Rp_{i\_optimal} = \sqrt{\alpha I_i}, \tag{6}$$

the higher values of α means that larger number of data sets will have 2 and more replicas.

Data Placement Optimization.
Number of replicas.

The graph below shows how optimal number of replicas for a data set depends on the predicted data usage intensity and alpha value. Allowed numbers of replicas are $\{1, 2, 3, 4\}$.

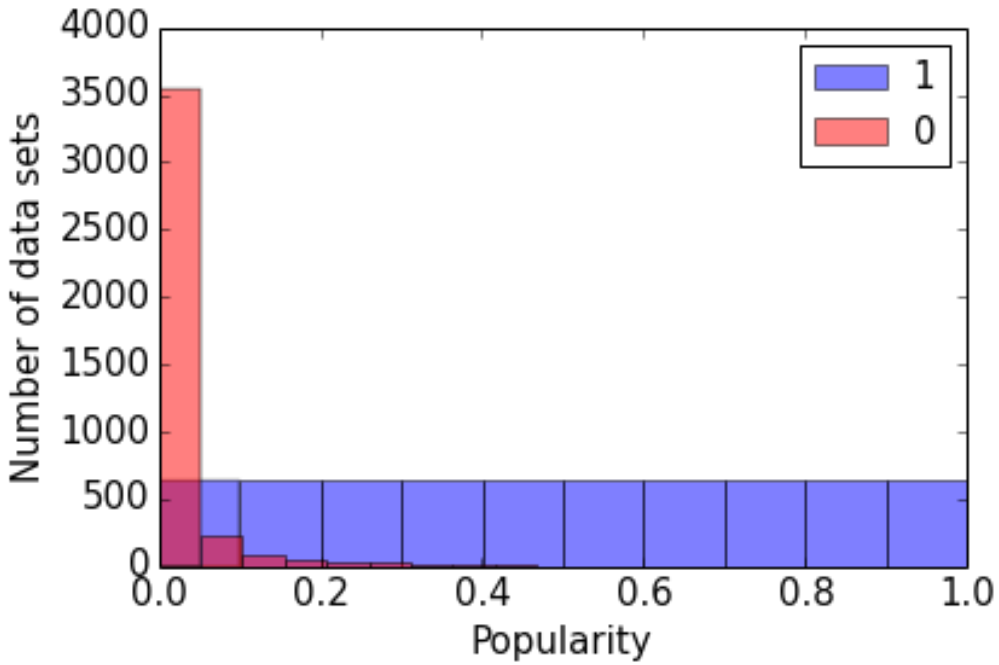


Example:

Lets predicted data intensity for a data set is $I = 10$ downloads per week and $\alpha = 0.5$. Then $Rp_{optimal} = \sqrt{\alpha I} = \sqrt{0.5 * 10} = 2.24 \approx 2$ replicas. This means that for the data set was assigned 2 replicas.

Results. (For LHCb data.)

Data access history of 10368 data sets were used in this analysis. Square under the ROC curve is 0.979. Data sets popularity distribution is shown below:



The higher data set's popularity, the higher probability that this data set will not be used in future.

Results. (For LHCb data.)

Original total size of the data sets on disk is about 10 322 Gb. About 77% of the data sets of disk have 3 replicas.

The following table represents how much disk space can be saved using the analysis above with parameters $C_{disk} = 100$, $C_{tape} = 1$, $C_{miss} = 2000$. (100% is all data sets on disk after removing some of them from disk.)

Alpha	Optimal popularity cut	Number of Replicas 1	2	3	4	Saving space
0	0,28	100 %	0 %	0 %	0 %	7 798 Gb
0,1	0,28	88,1 %	5,5 %	1,9 %	4,5 %	3 833 Gb
0,5	0,28	65 %	17,4 %	6,9 %	10,7 %	2 807 Gb
1	0,28	50,2 %	22,2 %	10,1 %	17,6 %	2 394 Gb
2	0,28	36,4 %	21,1 %	14 %	28,5 %	2 094 Gb
5	0,28	25,2 %	13,7 %	14 %	47,2 %	1 739 Gb

About 40% of all data sets were removed from disk.

Algorithms comparison.
Download time.

Following function was used to estimate time of downloading of all data sets by all users:

$$T = \sum_{i=1}^{n} I_i^* S_i t_{disk} \alpha(Rp_i)\delta_i + \sum_{i=1}^{n} (K_{tape} + S_i t_{tape}) m_i + \sum_{i=1}^{n} I_i^* S_i t_{disk} m_i \tag{7}$$

where $\alpha(Rp_i) = 0.05 + \frac{1}{Rp_i}$
t_{disk} - average time of downloading of 1Gb data from disk,
t_{tape} - average time of downloading of 1Gb data from tape to disk,
K_{tape} - constant time needed to restore data set from tape to disk,
I_i^* - average number of downloading of a data set per week,
S_i - size of one replica of i^{th} data set,
Rp_i - number of replicas of i^{th} data set,
δ_i is equal 1 if i^{th} data set on disk, otherwise it is 0,
m_i (misclassification) is equal 1 if i^{th} data set was restored from tape to disk.

Algorithms comparison.
Download time.

First 78 weeks of time series of data sets were used as algorithms inputs. Last 26 weeks were used to measure quality of the algorithms and estimating how many times the data sets were downloaded.

The first term of downloading time equation represents time of download of all data sets from disk by all users.

The second term represents time needed to restore data sets from tape which were removed from disk by an algorithm's mistake.

The third term represents time of download restored from tape to disk data sets by all users.

In this study the following values were used:
t_{disk}=0.1 hour/Gb, t_{tape}=3 hours/Gb, K_{tape} = 24 hours.

Algorithms comparison.
LRU (Last Recently Used) algorithm.

The LRU algorithm takes last observations of the data sets access history and decides which data sets should be removed from disk and which ones not. For the algorithm some assumptions were made:

- 1-78 weeks were used as the algorithm inputs. 79-104 weeks were used to measure quality of the algorithm.
- If a data set was not be used during last N weeks (from $78 - N^{th}$ to 78^{th} weeks), this data set was removed from disk.
- Number of data sets replicas were not changed.

Algorithms comparison. Results. (For LHCb data.)

Downloading time ratio is $T_{algorithm} / T_{no\_algorithm}$.

Our algorithm			
Alpha	Downloading time ratio	Saving space, %	Nb of wrong removings
0	3,35	71	9
0,01	0,99	46	9
0,05	0,96	34	9
0,1	0,96	30	9
0,5	0,96	23	9
1	0,96	19	9
2	0,96	16	9

LRU algorithm			
N	Downloading time ratio	Saving space, %	Nb of wrong removings
1	1,33	63	1973
2	1,28	58	1659
5	1,14	50	1357
10	1,11	44	966
15	1,07	38	635
20	1,03	33	370
25	1,02	30	193

For 7 maximum number of replicas:

Our algorithm			
Alpha	Downloading time ratio	Saving space, %	Nb of wrong removings
0	3,35	71	8
0,001	1,03	57	8
0,005	0,72	40	8
0,01	0,68	34	8
0,05	0,63	11	8
0,1	0,62	1	8

Conclusion.

- Data Popularity represents the data sets probability will not be used in future.
- The higher predicted data set's usage intensity, the higher the data set's number of replicas.
- The loss function optimization allows save up to 40% of disk space and decrease downloading time up to 30%.

Links

DataPopularity python module with presented results can be downloaded from here: https://github.com/hushchyn-mikhail/DataPopularity

Presented results with DataPopularity instructions is shown here.

Study is performed by means of Reproducible Experiment Platform.

Whole research can be found here.