Monitoring of the SPD data management system

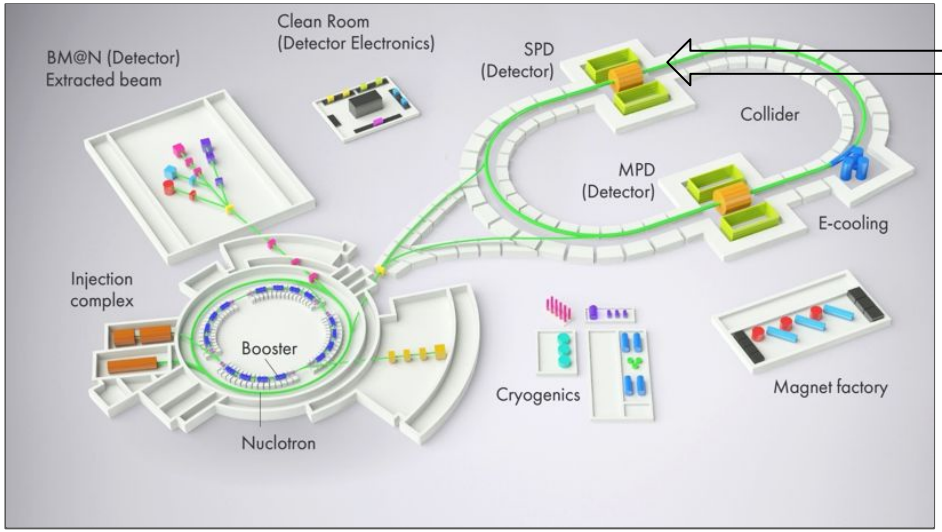
Alexey Konak
JINR MLIT
konak@jinr.ru

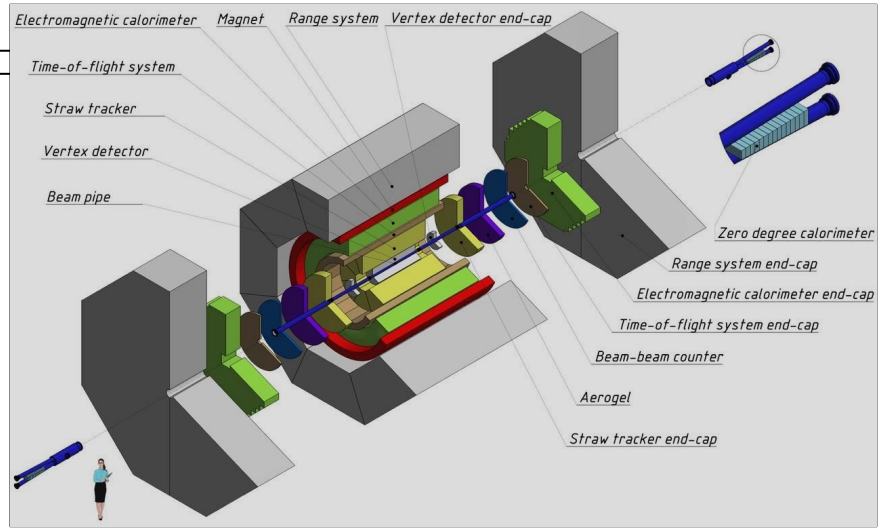
29th International Scientific Conference of Young Scientists and Specialists (AYSS-2025)
MLIT, JINR, Dubna
30.10.2025

Spin Physics Detector (SPD)

The spin structure of the **nucleon** is one of the fundamental properties of matter. The spin of a nucleon is distributed between its components — **quarks** and **gluons**, and their mutual movement.

The EMC, HERMES, and COMPASS experiments have made it possible to study in detail the contribution of **quarks** to spin. However, the role of **gluons** remains poorly understood and requires further research.



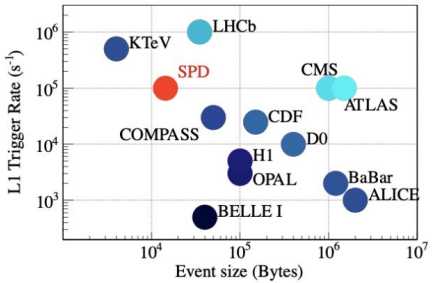


The SPD facility is being created for a more accurate study of the contribution of **gluons** to the spin of the **nucleon**.

SPD as data source

The expected event rate of the SPD experiment is about 3 MHz (pp collisions at $\sqrt{s}$ = 27 GeV and 10^{32} cm^{-2}s^{-1} design luminosity). This is equivalent to a raw data rate of 20 GB/s or 200 PB/year, assuming a detector duty cycle is 0.3, while the signal-to-background ratio is expected to be on the order of 10^{-5}. Taking into account the bunch-crossing rate of 12.5 MHz, one may conclude that pile-up probability cannot be neglected.

- SPD TDR



The goal of the online filter is at least to decrease the data rate by a factor of 20, so that the annual growth of data, including the simulated samples, stays within 10 PB. Then, data are transferred to the Tier-1 facility, where a full reconstruction takes place and the data is stored permanently. The data analysis and Monte-Carlo simulation will likely run at the remote computing centres (Tier-2s). Given the large data volume, a thorough optimization of the event model and performance of the reconstruction and simulation algorithms are necessary.

- Data from the detector – 20 GB/s (or 200 PB/year "raw" data, ~3*10^13 events/year)
- Simulation results – ??? (the exact volume is unknown, but there will be no less of them than the data from the detector.)
- Data of various intermediate formats along the way from “raw” to ready for analysis by physical groups – ??? (there will be a lot of them...)

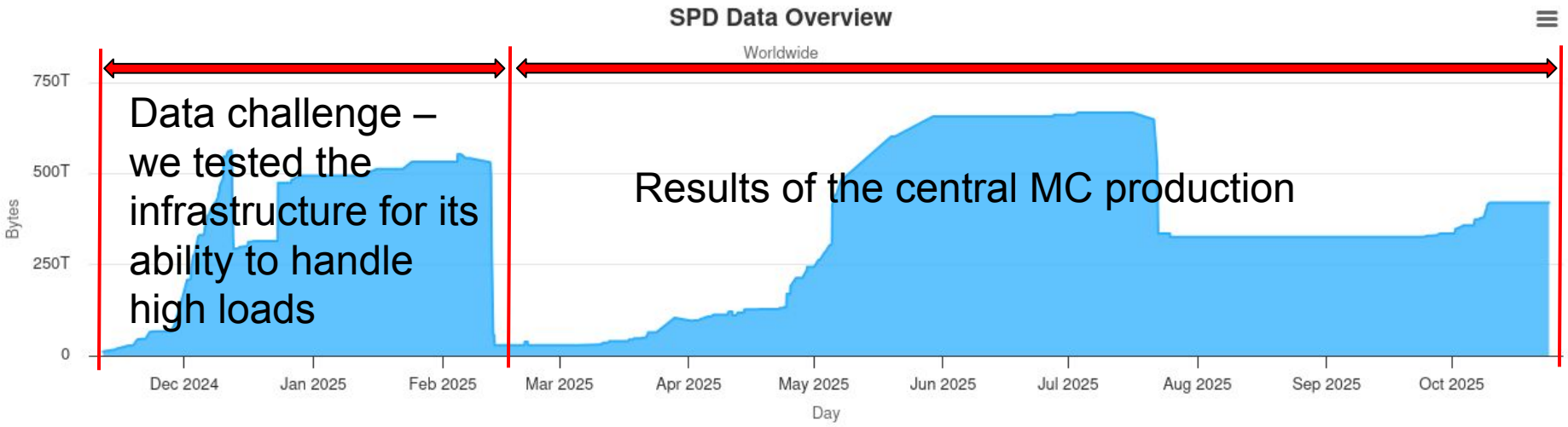
About Rucio

Rucio is an open-source software framework that provides functionality for data management and access in a distributed storage environment.

Currently, the Rucio system can be used to:

- data catalogue;
- organize data in a hierarchical structure for easy navigation and management;
- storage of any types of experimental data;
- storage and management of metadata;
- data lifecycle management;
- unified interaction of a heterogeneous network and storage infrastructure;
- adaptive data replication and recovery;
- automated data transfer between storages;
- providing monitoring metrics: data usage, system performance, services health.

Data Management



- Rucio in stable operation mode
- We already have significant data
- Replication of new data across all out storage facilities is automated

It’s necessary to monitor system performance, services health etc…

Development of Monitoring System

Rucio collects a bunch of metrics that can be used to monitor different activities, but does not provide a ready-made monitoring solution. It was necessary to:

- **design** a monitoring system;
- **deploy** all the necessary infrastructure;
- **configure** each component;
- **study** information provided by Rucio;
- **build** visualization panels.

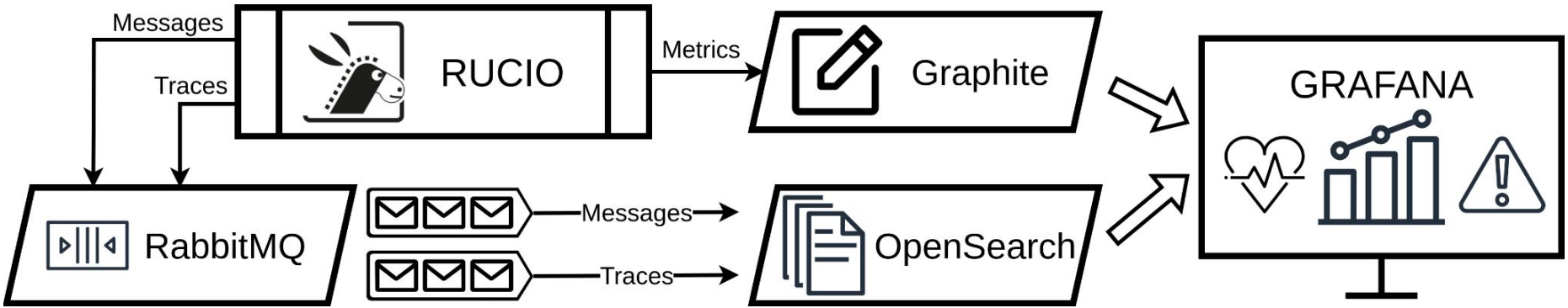
Types of information for monitoring:

- **Metrics** provided by Rucio-server and Rucio-daemons: a measurable numerical characteristic that shows how the system works.
- **Messages**: information about replication and deletion operations, Scope, name, size, status (successful/failure), duration, etc.
- **Traces**: information about the execution of a single transaction.
  GET, PUT – from PanDA;
  UPLOAD, DOWNLOAD, TOUCH – from users.

Monitoring System

Components of Monitoring System:

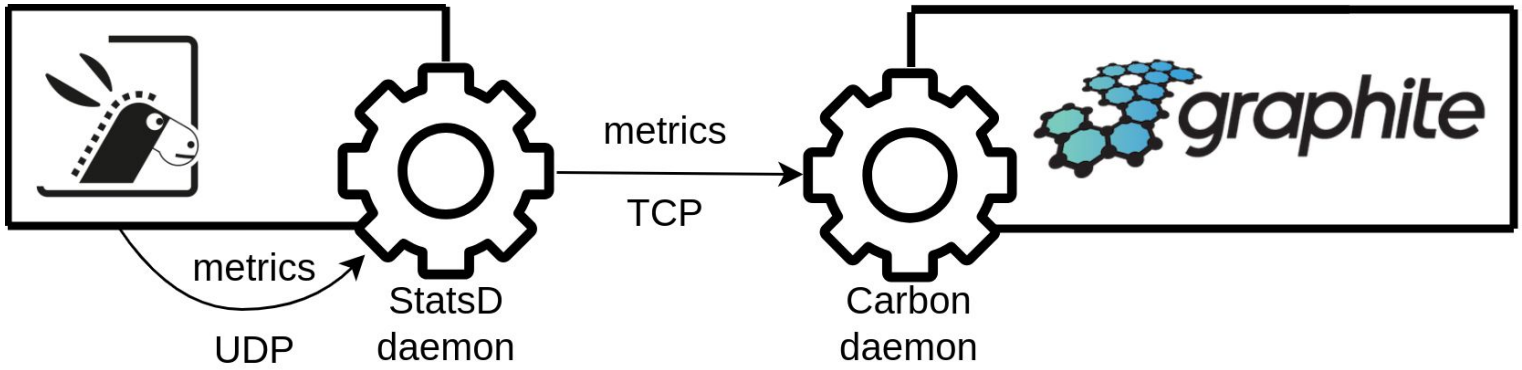
- **Graphite**: collects metrics from Rucio server.
- **RabbitMQ**: queues with messages and traces.
- **OpenSearch**: documents with messages and traces (long term storage).
- **Grafana**: visualizations.



Metrics delivery

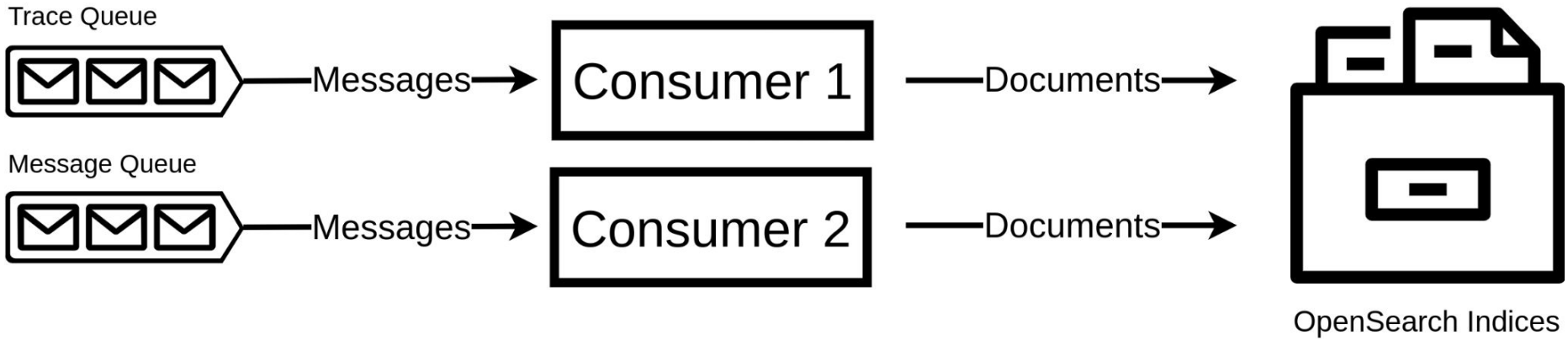
Metric ia a measurable numerical characteristic that shows how the system works. **Metrics** sended to Graphite automatically.

1) Metrics generation – the **Rucio** code is instrumented using a library for collecting metrics. The metrics are collected in a memory buffer and sent in batches to the **StatsD daemon** (on Rucio side).
2) Aggregation and forwarding – The **StatsD daemon** pre-aggregates the received metrics and converts them for transmission to Graphite (buffering is possible if Graphite is not available).
3) Receiving and storage – **Carbon daemon** (on Graphite side) receives metrics and writes them to disk in an efficient format. (Whisper is a fixed-size, time-series database format for storing numerical data over time.)



Messages/Traces delivery

Queues in RabbitMQ are used for temporary storage of Rucio messages and traces. One entry in the queue is called a **message**.

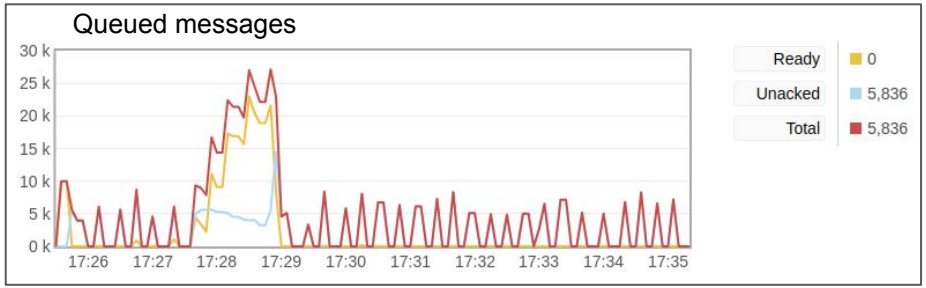


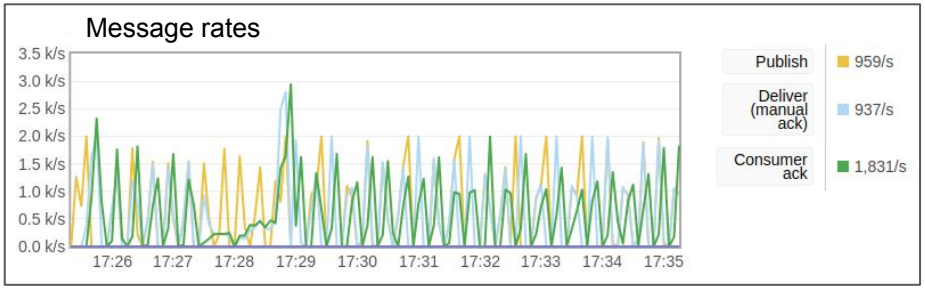
Special consumers are used to deliver messages in OpenSearch. It also preprocesses documents and distributes them to different indices (by event_type).

One unit of information in OpenSearch – **document**. Documents are stored in Indices. **Index** is a logical space for similar documents in OpenSearch.

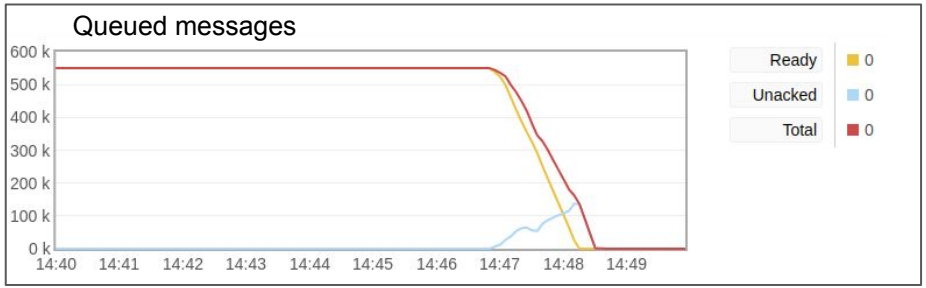
Messages/Traces delivery

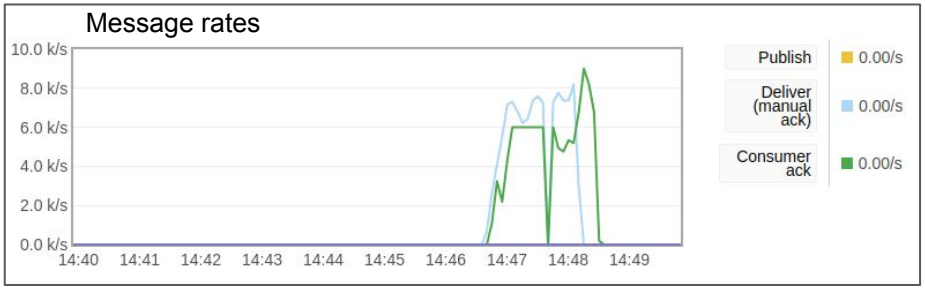
A special Rucio-daemon sends **messages** to the RabbitMQ queue (messages are taken from the Rucio-DB and sent in batches of 10k).





Traces are automatically sent to the RabbitMQ queue after the Rucio-server receives them.





The consumer test from the queue with traces turned out to be artificial. I haven't been able to test it online yet.

OpenSearch indices

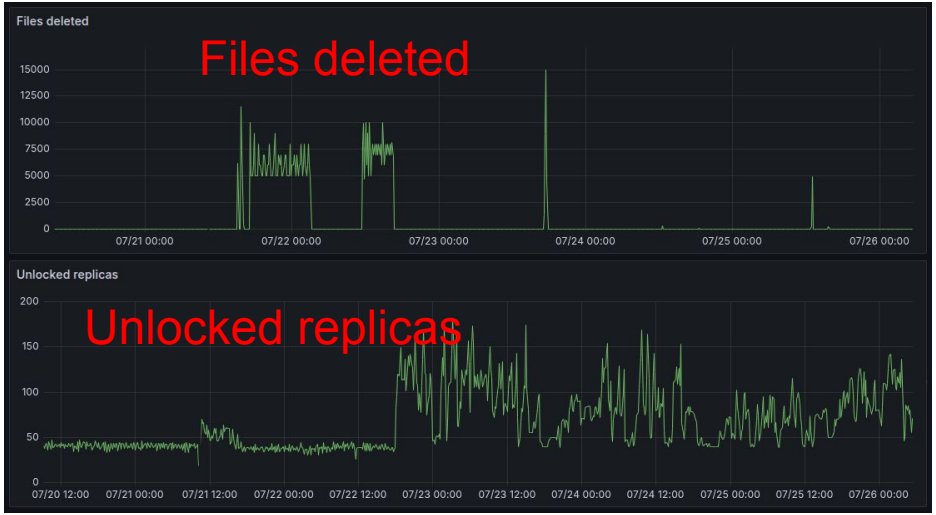
health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
yellow	open	rucio_trace_upload	WEQKdZF0TM2MkAHzzaZzBA	1	1	133	0	135.8kb	135.8kb
yellow	open	rucio_transfer_queued	mCXJo6WjRHa8-zBo0bKo7Q	1	1	1592610	0	385.8mb	385.8mb
yellow	open	rucio_deletion_failed	HcmZfy8ITaunD5rHpjbm9g	1	1	841369	0	90.7mb	90.7mb
yellow	open	rucio_trace_put	PdZLv9zBS8WiTcRR3XiVgg	1	1	733710	0	366.3mb	366.3mb
yellow	open	rucio_transfer_submitted	O7YfPGd0TQ2ZadA7os5IRA	1	1	1286321	0	371mb	371mb
yellow	open	rucio_trace_touch	9I6J7UEGTy-25zjAvl3bDA	1	1	0	0	208b	208b
yellow	open	rucio_deletion_done	kezjdZa9SX2GpK6XU8luTw	1	1	585189	0	108.1mb	108.1mb
green	open	.opensearch-observability	MnInmQADRn2W36RcsHgFjA	1	0	0	0	208b	208b
yellow	open	rucio_transfer_done	aXNNF80HQj--Livqjddr6A	1	1	781302	0	405.8mb	405.8mb
yellow	open	rucio_transfer_submission_failed	fqSz1208R9uZ4BBN5lB62w	1	1	1387101	0	356.7mb	356.7mb
yellow	open	rucio_transfer_failed	0lgFhz5XQyOwm7cfOicOMA	1	1	563410	0	295.5mb	295.5mb
yellow	open	rucio_trace_get	E6CwiYg1SYG3BV0Re9wIXA	1	1	226467	0	109.1mb	109.1mb
yellow	open	rucio_deletion_not_found	xKsiLJ4yTGWMPiL1ykEKdQ	1	1	86715	0	12.6mb	12.6mb
yellow	open	scopes	8LLkv18iQK-7OdPb6ijCtA	1	1	120	0	18kb	18kb
yellow	open	rucio_trace_download	IKo1MFGVQNecIFIc5D5dWw	1	1	748	0	300.5kb	300.5kb

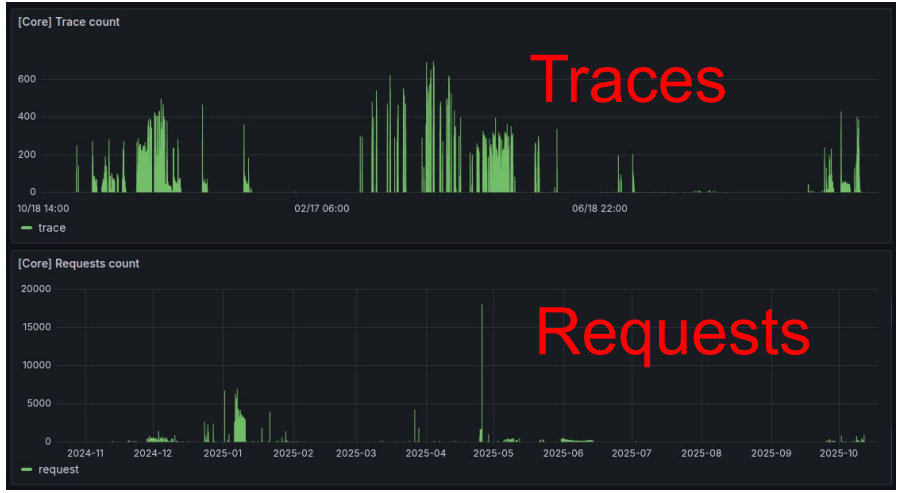
There are already a lot of documents in OpenSearch for monitoring (~ 8kk documents).
Each production generates many traces (1 job = 1 trace = 1 document).
Each transfer/deletion generates a message –> 1 document.

Monitoring System (from metrics) [1/2]

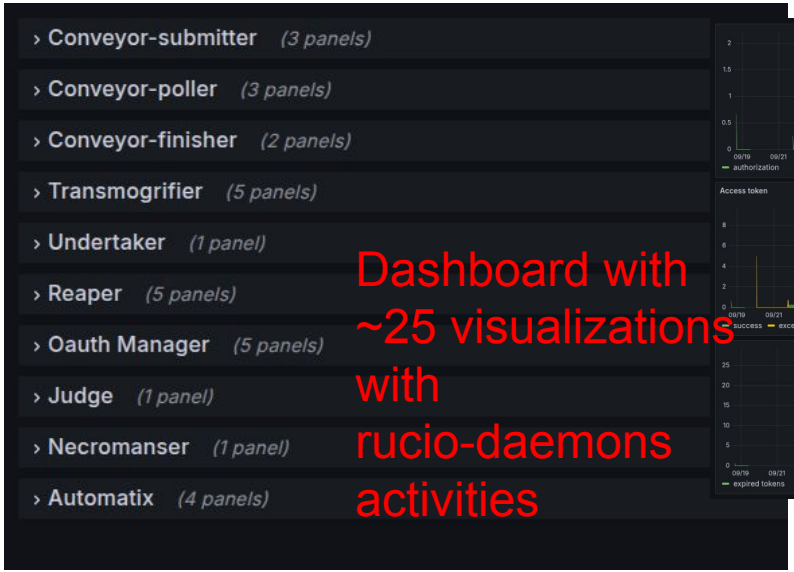
We can monitor:

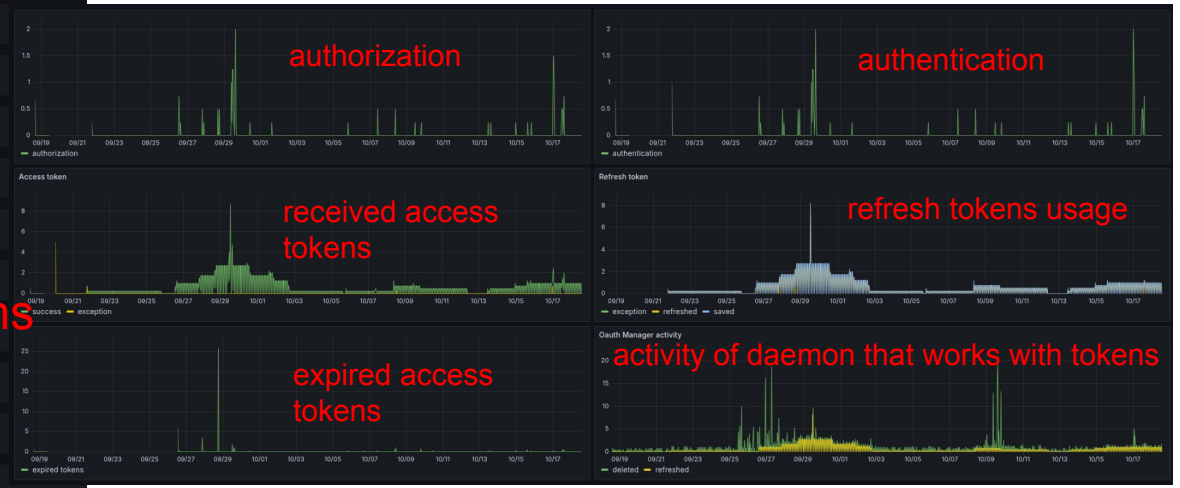
- counts of traces and requests;
- daemons activity and theirs errors;
- authentication/authorization via IAM;
- deletion process;
- transferring process;





Monitoring System (from metrics) [2/2]

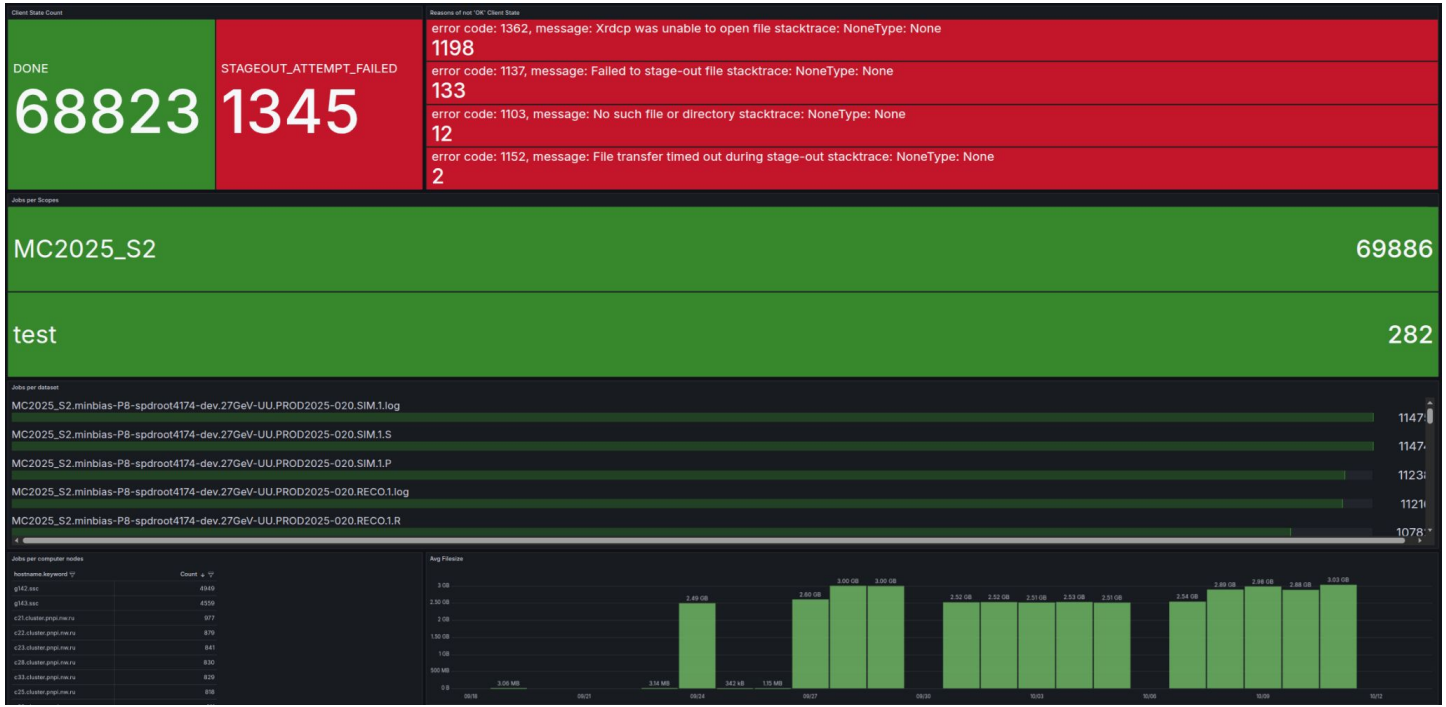




Trace Monitoring (put) [1/2]

- number of traces (documents);
- number of jobs per Panda Queue/Remote Site;
- number of jobs by task ID;

Trace Monitoring (put) [2/2]



- number and types of clientState (it's ok or error);
- number and types of errors;
- number of jobs per Scope/Dataset/computer node;
- average file size;

Message Monitoring (transfer submission)

- count successful/failure transfer submission;
- count of successful/failure submissions per Scope/Dataset;

Message Monitoring (transfers) [1/2]

- count success/failure transfer;
- sum of transferred bytes total and per Scope/Dataset;

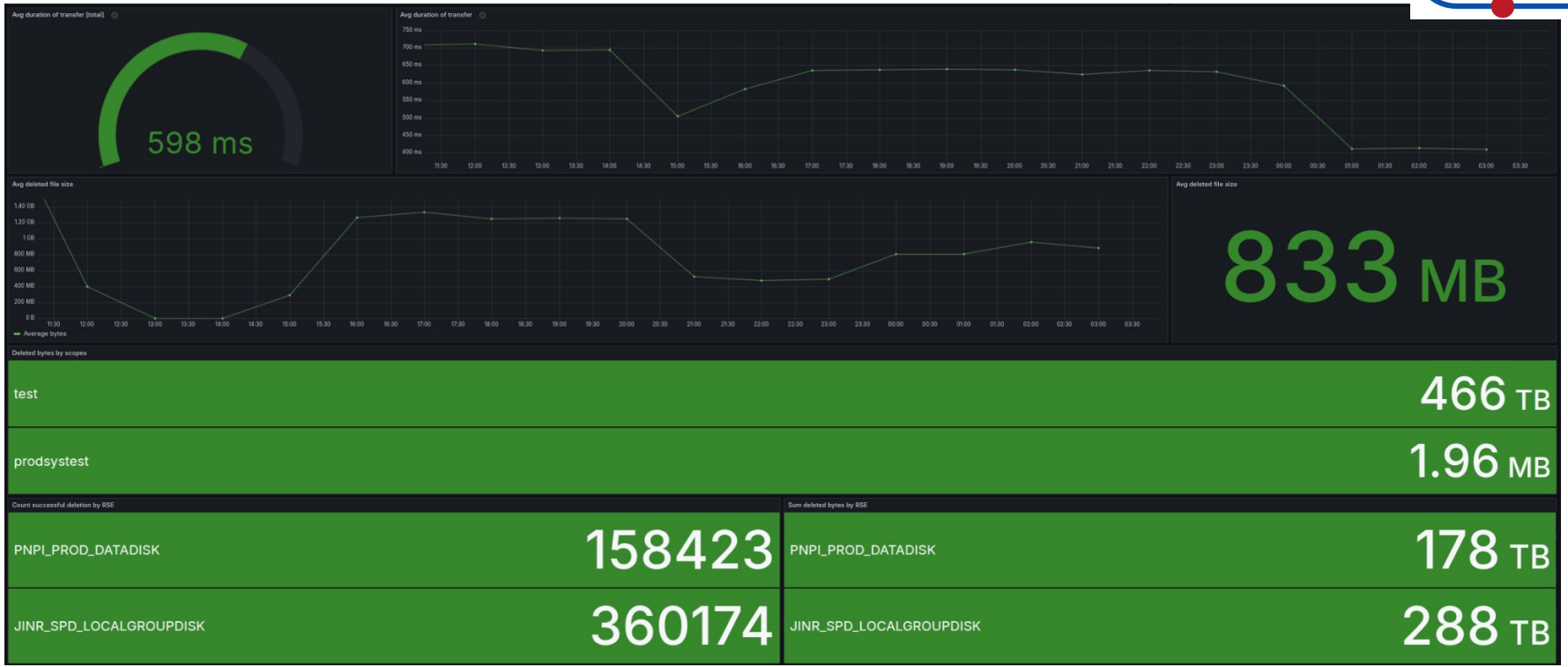
Message Monitoring (transfers) [2/2]

- avg duration of transfer;
- avg transferred file size;
- count successful transfers and sum transferred bytes between RSEs;

Message Monitoring (deletion) [1/2]

- count success/not_found/failure deletions;
- sum of deleted bytes total and per Scope;

Message Monitoring (deletion) [2/2]



- avg duration of deletion;
- avg deleted file size;
- count files/sum bytes deleted by RSEs;

Summary

- A monitoring system based on the collection and presentation of Rucio metrics has been deployed.
- The system allows us to monitor the status of Rucio-server, Rucio-daemons and various activities from the services and users.
- This allows us to detect typical errors and fix them faster.

Future plans

- Possible changes/additions in current monitoring.
- Monitoring of user activity – needs to be completed.
- The development of monitoring of data volumes by scopes – already begun.

Thank you for your attention!

Trace Monitoring

We have the following visualizations of GET/PUT traces:

- number of traces (documents);
- number of jobs per Panda Queue/Remote Site/Local Site;
- number of jobs by task ID;
- number and types of clientState (it's ok or error);
- number and types of errors;
- number of jobs per Scope/Dataset/computer node;
- average file size;

UPLOAD/DOWNLOAD traces:

- number of traces (documents);
- user activity;
- number and types of clientState (it's ok or error);
- number and types of errors;
- average file size;
- number of activities per protocol;

Message Monitoring

We have the following visualizations of messages:

- count successful/failure transfer submission;
- count of successful/failure submissions per Scope/Dataset;
- count success/failure transfer;
- sum of transferred bytes total and per Scope/Dataset;
- avg duration of transfer;
- avg transferred file size;
- count successful transfers and sum transferred bytes between RSEs;
- count success/not_found/failure deletions;
- sum of deleted bytes total and per Scope;
- avg duration of deletion;
- avg deleted file size;
- count files/sum bytes deleted by RSEs;