StoRM status and evolution plans

Enrico Vianello
INFN-CNAF

GDB, February 8th, 2023

enrico.vianello@cnaf.infn.it

StoRM - STOrage Resource Manager

StoRM is a storage resource manager for disk-based storage systems

Provides a "thin" management layer (SRM, WebDAV) over a POSIX FS

- Typically IBM GPFS or Lustre

Supports a tape system through integration with GEMSS, a tape library manager component also developed @ INFN

Provides flexible AuthN/Z support:

- VOMS & OAuth/OIDC (WebDAV & CDMI interfaces)
- File access control is enforced via POSIX ACLs

StoRM components

Name	Latest release
StoRM Backend	1.11.21
StoRM Frontend	1.8.15
StoRM WebDAV	1.4.1
StoRM Native Libs	1.0.6-2
StoRM Info Provider	1.8.2
StoRM SRM Client	1.6.1
StoRM GridFTP	1.2.4
StoRM XMLRPC-C	1.39.12
CDMI StoRM	0.1.1
StoRM Utils	1.0.0
StoRM Puppet module	3.4.0

StoRM main components are:

- StoRM Backend
- StoRM Frontend

SRM

- StoRM WebDAV
- StoRM GridFTP

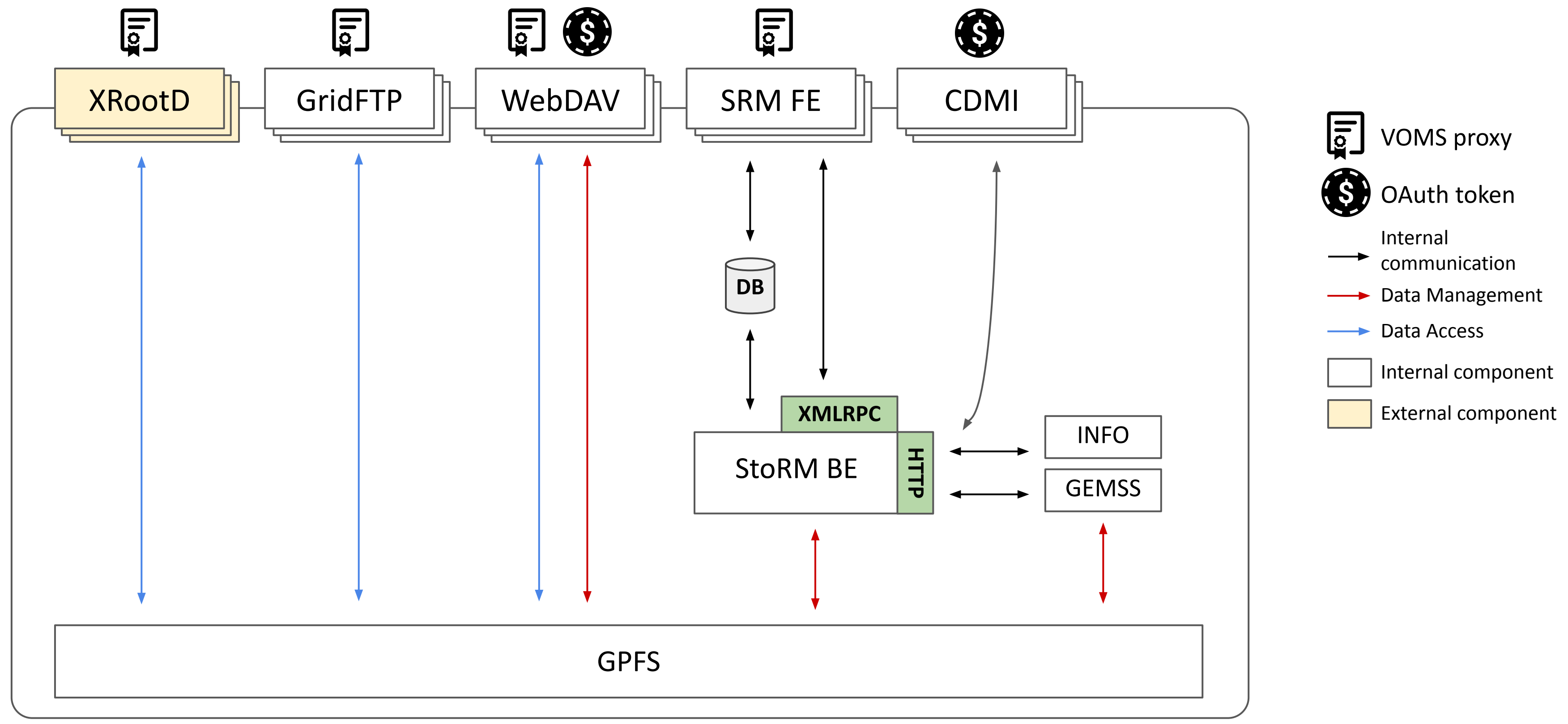
Platforms:

- CentOS 7
- AlmaLinux OS 9 (next)

Repo:

- StoRM stable yum repo
- UMD repositories

StoRM: a typical deployment architecture



Latest releases: StoRM 1.11.20 + StoRM 1.11.21

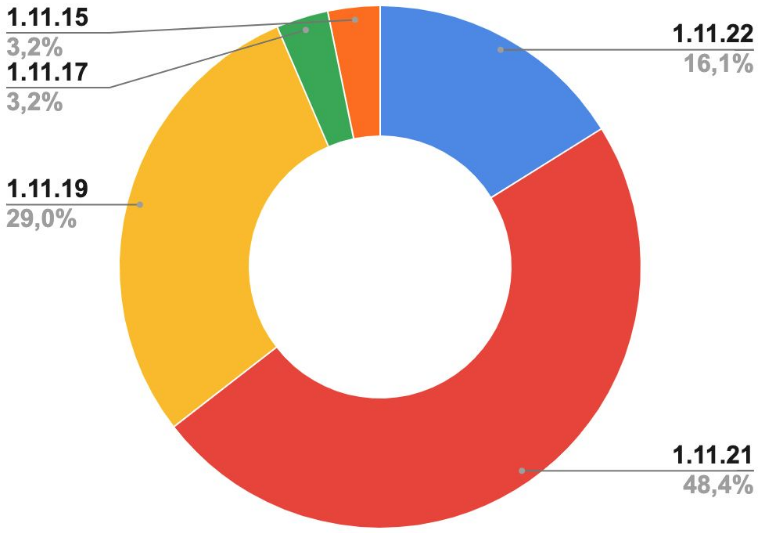
StoRM v1.11.20 release https://italiangrid.github.io/storm/release-notes/StoRM-v1.11.20.html

- requires and install Java 11 for all the Java components;
- adds **support for externalized session management for StoRM WebDAV**;
- includes user traceability information in StoRM WebDAV access log
- fixes several minor codebase issues on Frontend, some of them could cause a **memory leak**;
- adds the average time in the summary for round Frontend monitoring log;
- fixes some bugs about StoRM WebDAV: OIDC login button, ownership issue on logging directory.

StoRM v1.11.21 release https://italiangrid.github.io/storm/release-notes/StoRM-v1.11.21.html

- fixes known issue of StoRM v1.11.20 which could break connections with MariaDB
- fixes boot order ensuring that mariadb service is started before StoRM services;
- fixes the failed state shown on stop/restart of the Java services;
- provides a set of command line scripts that allows admins to edit storage area’s space info.

Distribution of StoRM versions



This is a beta version of StoRM v1.11.22 installed at CNAF-T1

21 sites
31 instances

Source egee-bdii.cnaf.infn.it

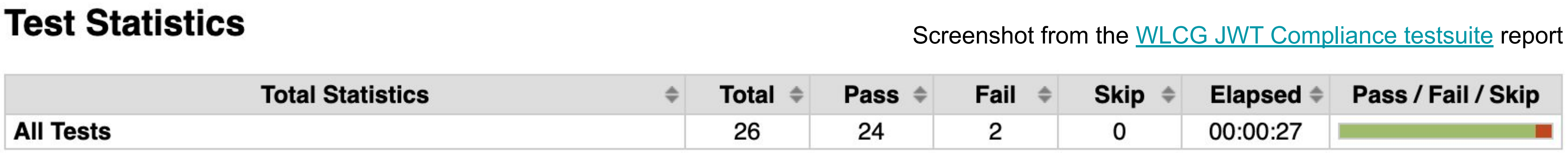
Sites	Instances
INFN PISA	1
KEK	3
UIIP NASB	1
INFN GENOVA	1
INFN TRIESTE	1
INFN LECCE	1
IFCA	1
IFIC	1
INFN ROMA3	1
INFN MILANO	1
UNI. SIEGEN	1
INFN FERRARA	1
UNI. MAINZ	1
IRB	1
QMUL	5
LIP	1
INFN-BARI	1
MILANO-BICOCCA	1
TEL-AVIV UNI.	1
INFN-CNAF	5
TECHNION HAIFA	1

Main recent developments

- proof-of-concept implementation of WLCG Tape REST API
  - more details later
- StoRM Backend database connection pool refactoring
- Upgrade of critical dependencies on StoRM WebDAV
  - spring boot
  - canl-java

Token-based AuthN/Z support

Screenshot from the WLCG JWT Compliance testsuite report



Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	26	24	2	0	00:00:27	🟩🟥

StoRM WebDAV fully supports token-based authN/Z

- Storage-issued token support (aka "Macaroons") ✔
- OpenID Connect authentication support ✔
- OAuth access token support ✔

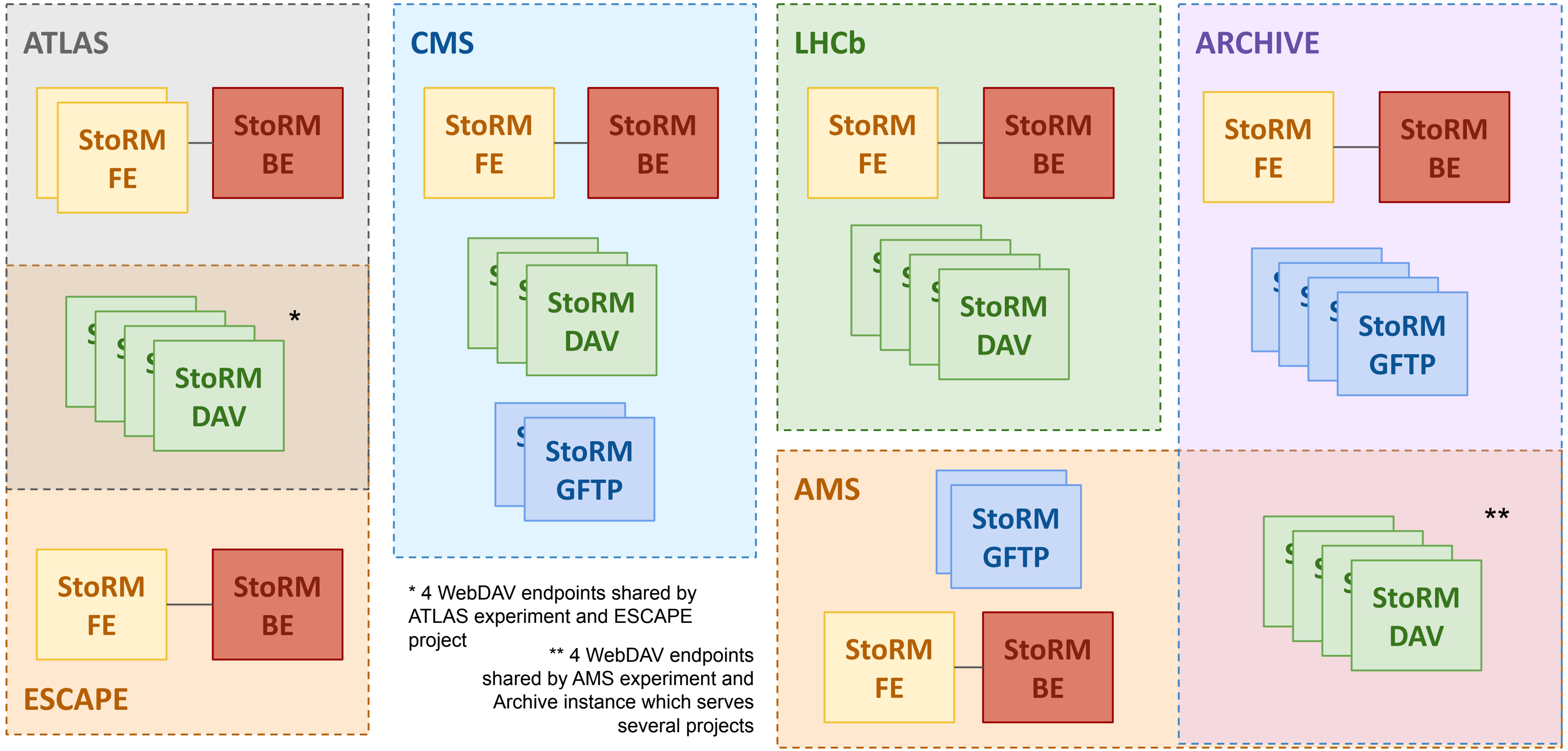
WLCG JWT profile compliance:

- audience enforcement support ✔
- capability-based authorization ✔
- group-based authorization ✔
- path-enforced-authz-checks ✘ (tag "not-critical")

StoRM @ CNAF-T1

Thanks to storage@infn-t1

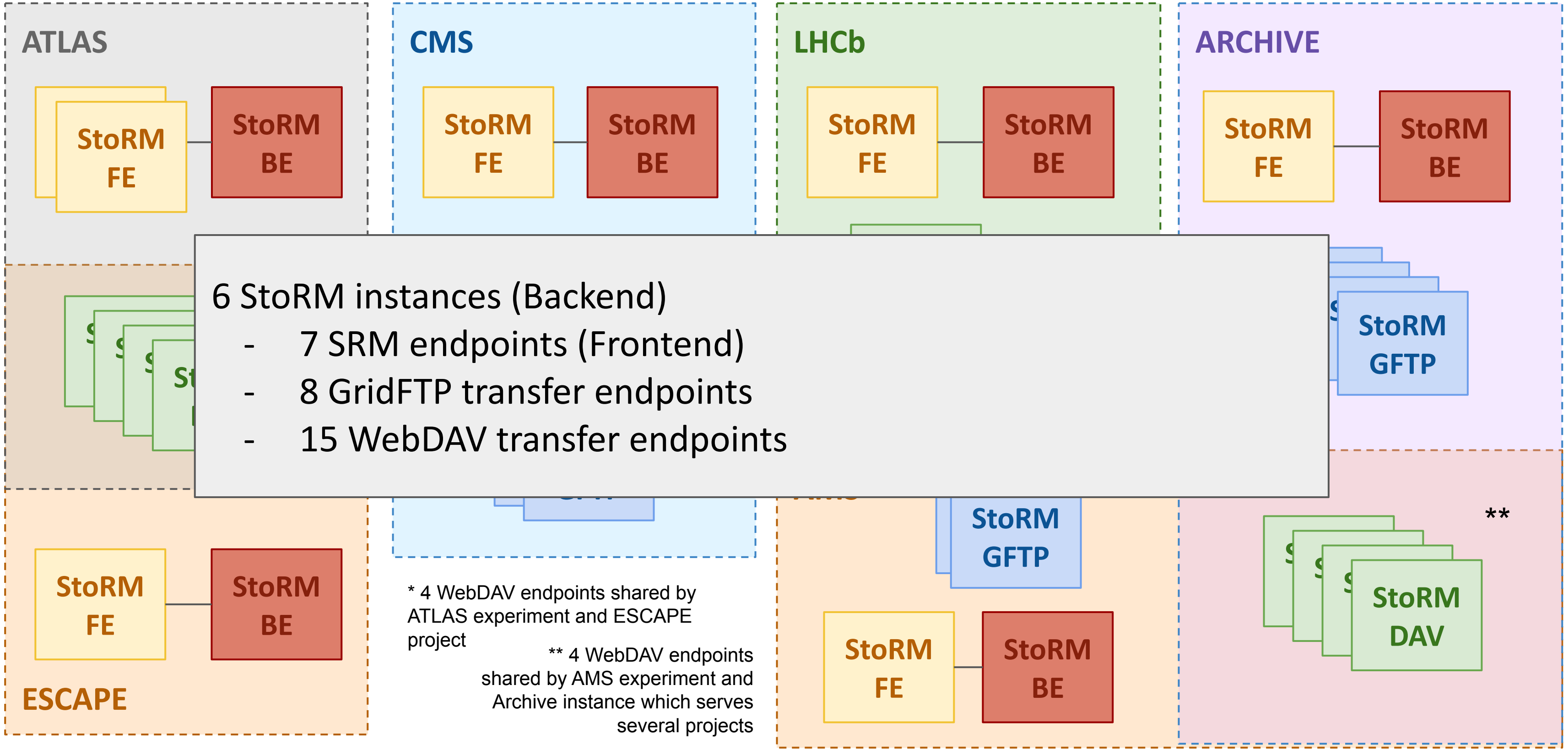
StoRM Deployments @ CNAF-T1 - Summary



* 4 WebDAV endpoints shared by ATLAS experiment and ESCAPE project

** 4 WebDAV endpoints shared by AMS experiment and Archive instance which serves several projects

StoRM Deployments @ CNAF-T1 - Summary



6 StoRM instances (Backend)

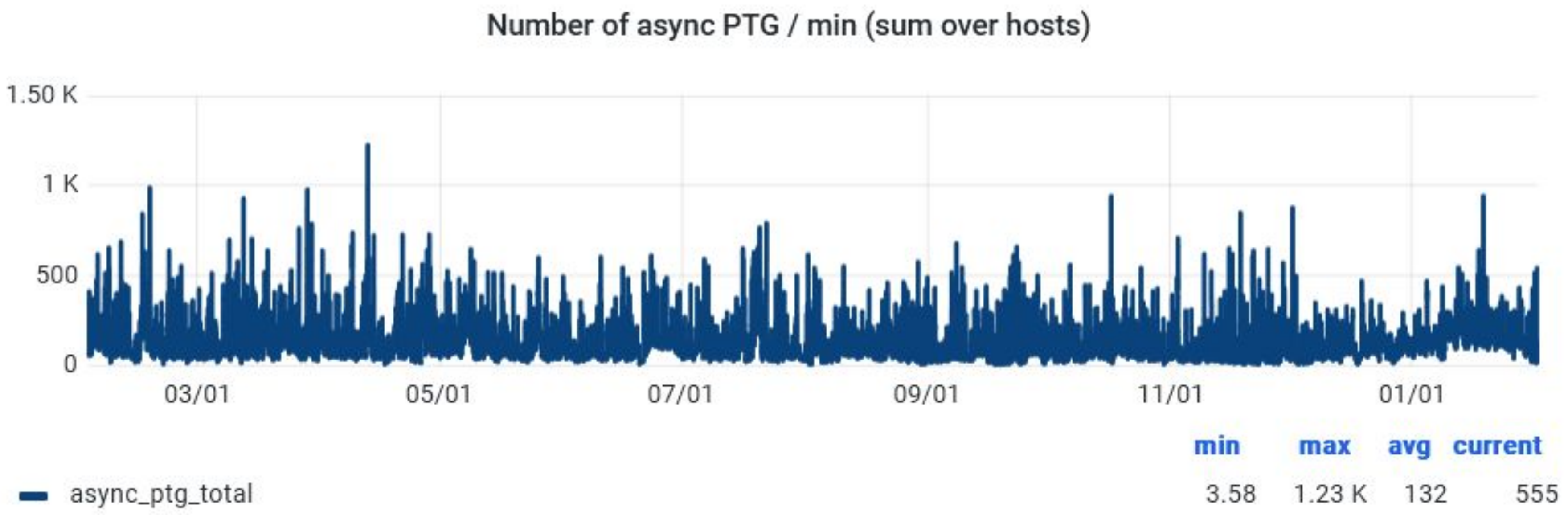
- 7 SRM endpoints (Frontend)
- 8 GridFTP transfer endpoints
- 15 WebDAV transfer endpoints

* 4 WebDAV endpoints shared by ATLAS experiment and ESCAPE project

** 4 WebDAV endpoints shared by AMS experiment and Archive instance which serves several projects

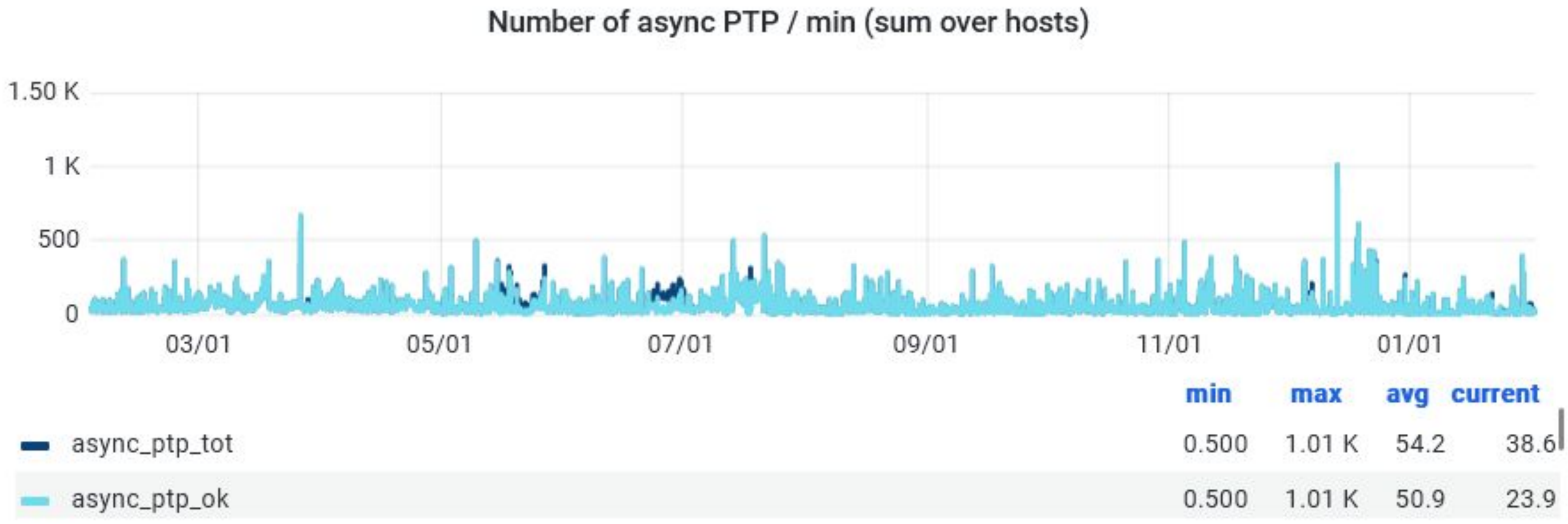
StoRM Deployments @ CNAF-T1 - Metrics

Some SRM specific metric of last 12 months



StoRM Deployments @ CNAF-T1 - Metrics

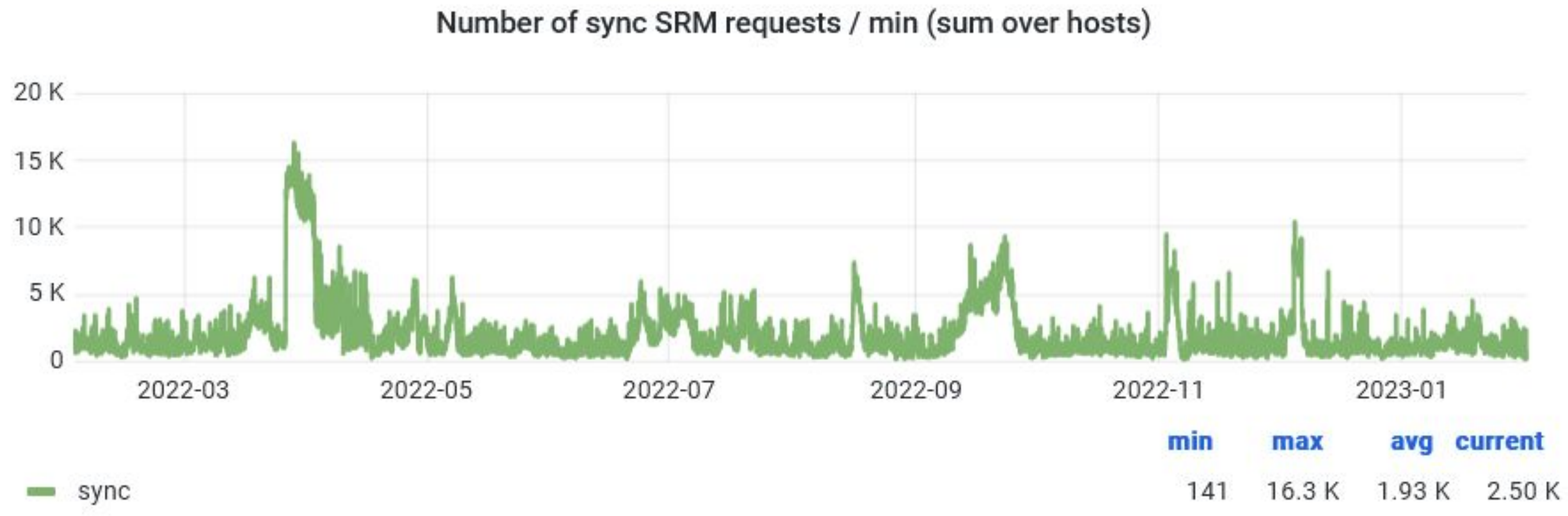
Some SRM specific metric of last 12 months



	min	max	avg	current
async_ptp_tot	0.500	1.01 K	54.2	38.6
async_ptp_ok	0.500	1.01 K	50.9	23.9

StoRM Deployments @ CNAF-T1 - Metrics

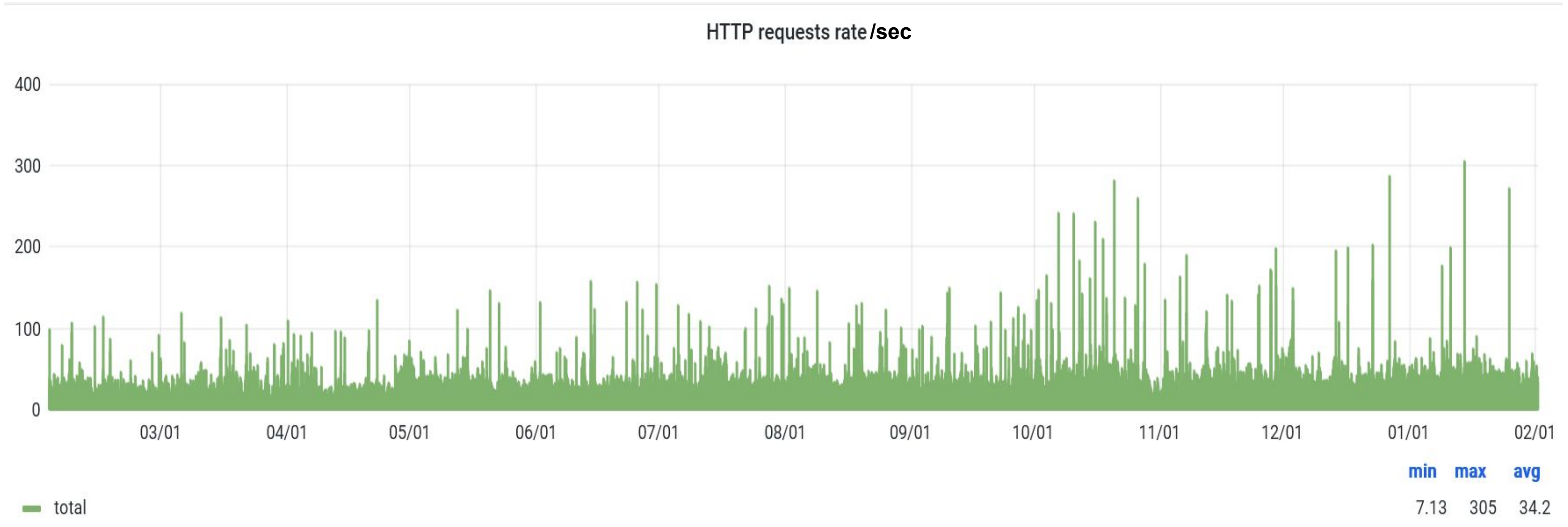
Some SRM specific metric of last 12 months



	min	max	avg	current
🟩 sync	141	16.3 K	1.93 K	2.50 K

StoRM Deployments @ CNAF-T1 - Metrics

Some HTTP/transfer specific metric of last 12 months

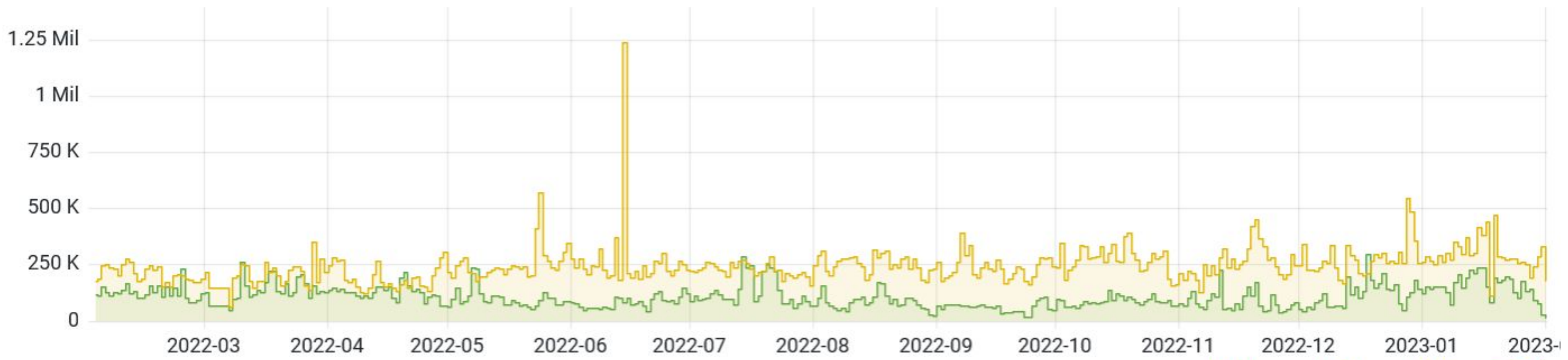


	min	max	avg
🟩 total	7.13	305	34.2

StoRM Deployments @ CNAF-T1 - Metrics

Some HTTP/transfer specific metric of last 12 months

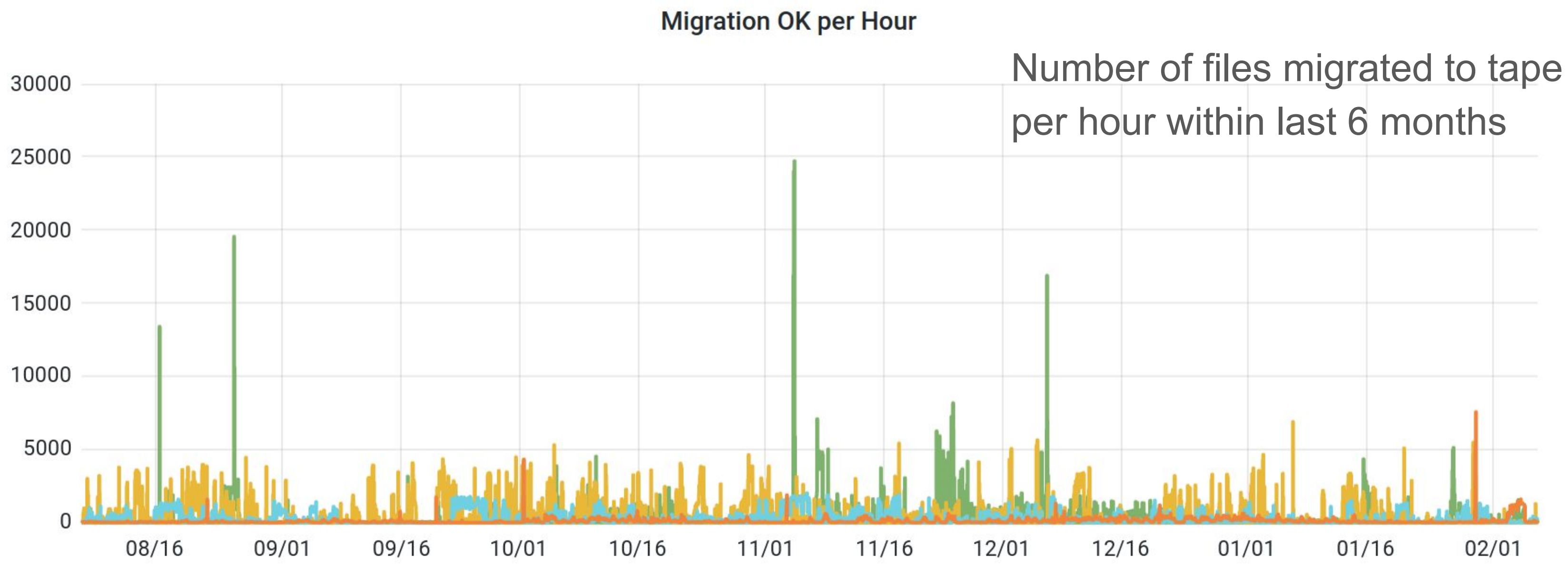
Number of transferred files **/day**



	Mean	Max	Min	Total
🟩 gridftp	133 K	449 K	12.1 K	69.4 Mil
🟨 webdav	239 K	1.24 Mil	48.0 K	125 Mil

StoRM Deployments @ CNAF-T1 - Metrics

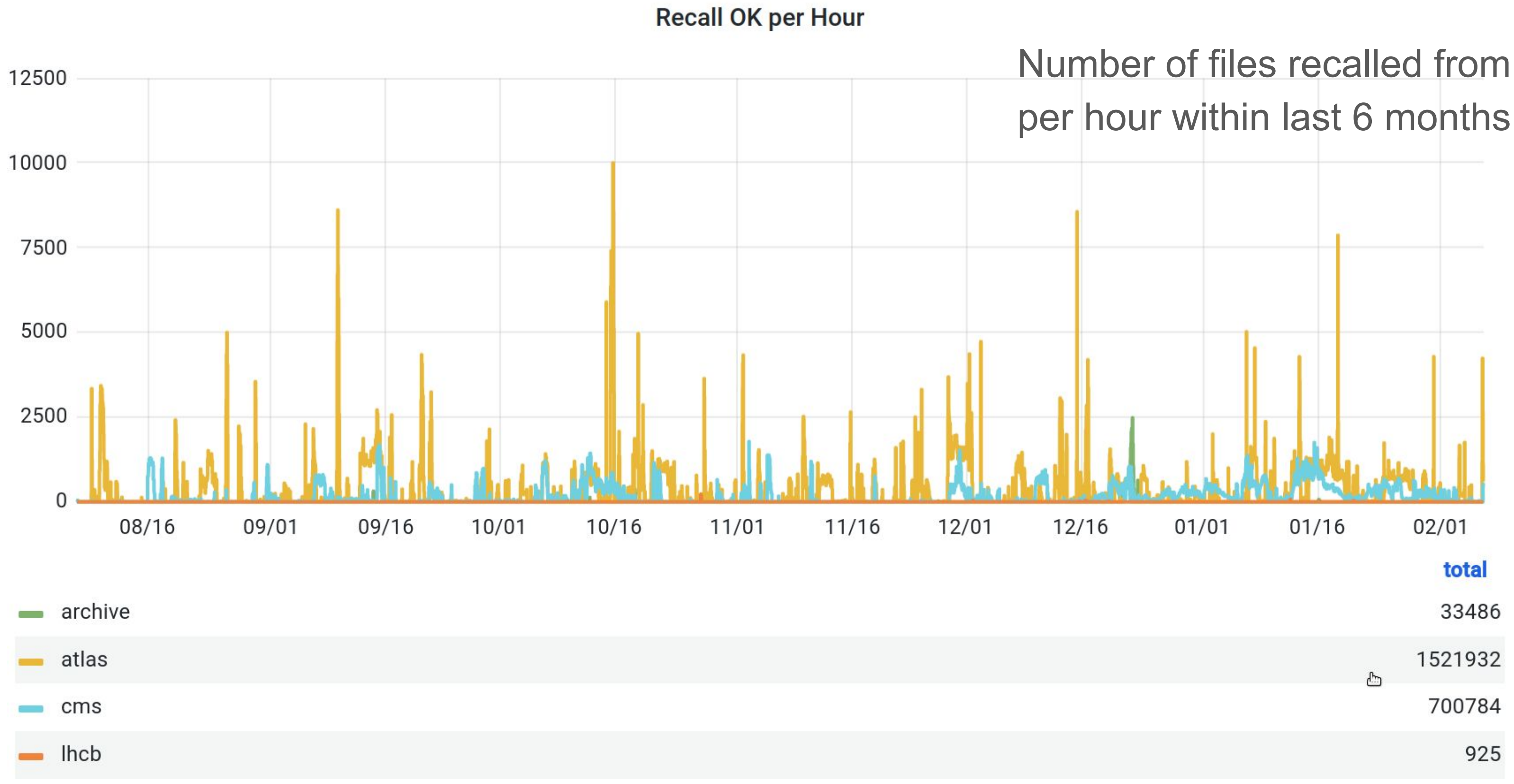
Number of files migrated to tape per hour within last 6 months



	total
🟩 archive	954264
🟨 atlas	2068857
🟦 cms	1198734
🟧 lhcb	515401

StoRM Deployments @ CNAF-T1 - Metrics

Number of files recalled from tape per hour within last 6 months



	total
🟩 archive	33486
🟨 atlas	1521932
🟦 cms	700784
🟧 lhcb	925

StoRM Deployments @ CNAF-T1 - Metrics

Summary of last year:

- avg ~132 SRM PtG/min (min 3.58, max 1.23 K)
- avg ~54.2 SRM PtP/min (min 0.5, max 1,01K, 94% success rate)
- avg ~1.93K Sync SRM requests/min (min 141, max 16.3K)
- avg ~34.2 HTTP requests/sec (min 7.13, max 16.3K)
- avg ~133K transferred files/day via GridFTP (min 12.1K, max 449K)
    - 69.4 Millions of files transferred within last 12 months
- avg ~239K transferred files/day via WebDAV (min 48, max 1.24 Millions)
    - 125 Millions of files transferred within last 12 months
- ~4.7 Millions of migrated files per hour (last 6 months)
- ~2.3 Millions of recalled files per hour (last 6 months)

StoRM support, maintenance and evolution

Support, maintenance & evolution statement

StoRM is the Grid/Cloud storage solution adopted by the INFN-CNAF data center and will be maintained and evolved by INFN for the foreseeable future

- This includes providing support (through GGUS tickets or mailing-lists) to other StoRM-based sites

StoRM: the development team

StoRM is mainly developed by the Software Development (SD) group at INFN-CNAF

We are currently five people working on middleware (mainly StoRM, INDIGO IAM, VOMS, Argus, ESACO):

- Francesco Giacomini (Lead)
- Enrico Vianello
- Federica Agostini
- Laura Cappelli
- Roberta Miccoli

Thanks to Tommaso Diotalevi for his contribution to the StoRM Tape REST API

We have to balance our effort on all these products

StoRM evolution: objectives

- Finalize new component StoRM Tape REST API
  - Last step to enable no-SRM deployments with tape
- Improve support for Cloud Storage providers by StoRM WebDAV
  - i.e. Google Cloud Storage, Amazon S3, ...
- Drop Globus dependency on StoRM Frontend, if/when needed
- Go beyond CentOS 7
  - build and release packages for AlmaLinux 9 platform
- Reduce maintenance and evolution costs
  - Current complexity mostly due to unused SRM “features”
- Simplify service operations and deployment
  - Service containerisation, K8S, …
- Improve observability
  - Consistent logging across services, metrics, tracing, …

To be released soon - ongoing developments

- StoRM Backend v1.11.22
  - Bug fixes + Pool of WebDAV endpoints support + DB Connection pool refactoring
- StoRM Native Libs v1.0.7 (already available into beta repo)
  - Bug fixes
- StoRM WebDAV v1.4.2
  - Deps upgrade + Bug fixes
- StoRM Info Provider 1.8.3
  - Bug fixes
- StoRM Tape REST API
  - alpha release

StoRM Tape REST API

The WLCG Tape REST API - Introduction

The WLCG tape REST API offers a common HTTP interface which allows clients to manage disk residency of tape-stored files and observe the progress of file transfer on disk.

In practice, this API realizes the HTTP alternative to SRM BringOnLine.
This API allows users to:

- stage bulk-request of tape-stored files, making them available on disk;
- track progress of a previously staged bulk-request;
- cancel a previously staged file replicas from disk;
- retrieve information about the progress of file's staging.

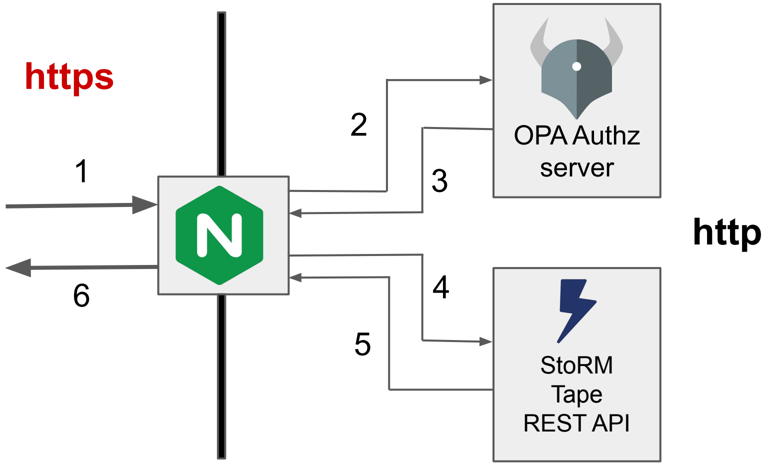
The API will be accessed via authentication mechanisms like X509 + VOMS (proxy-based) or token based (JWT).

StoRM Tape REST API - Introduction

- currently a proof-of-concept
  - but hopefully soon ready as a preview
- deployed as a standalone component
  - direct access from remote users
  - not deployed within StoRM WebDAV
- an opportunity for technology scouting
  - NGINX + NJS on the reverse proxy
  - Open Policy Agent as policy decision point
  - written in C++
  - containerized deployment

NGINX + OPA Authorization

1. The user submits an API request, which is VOMS/TLS terminated by NGINX
2. NGINX sends the request to Open Policy Agent (OPA)
3. OPA makes the authZ decision using its rules and data and sends it back to NGINX
   - In case of negative authZ, 403 is returned
4. In case of successful authZ, the request is forwarded to the upstream service
5. (and 6.) The response from the upstream service is relayed to the user



StoRM Tape REST API - Development status

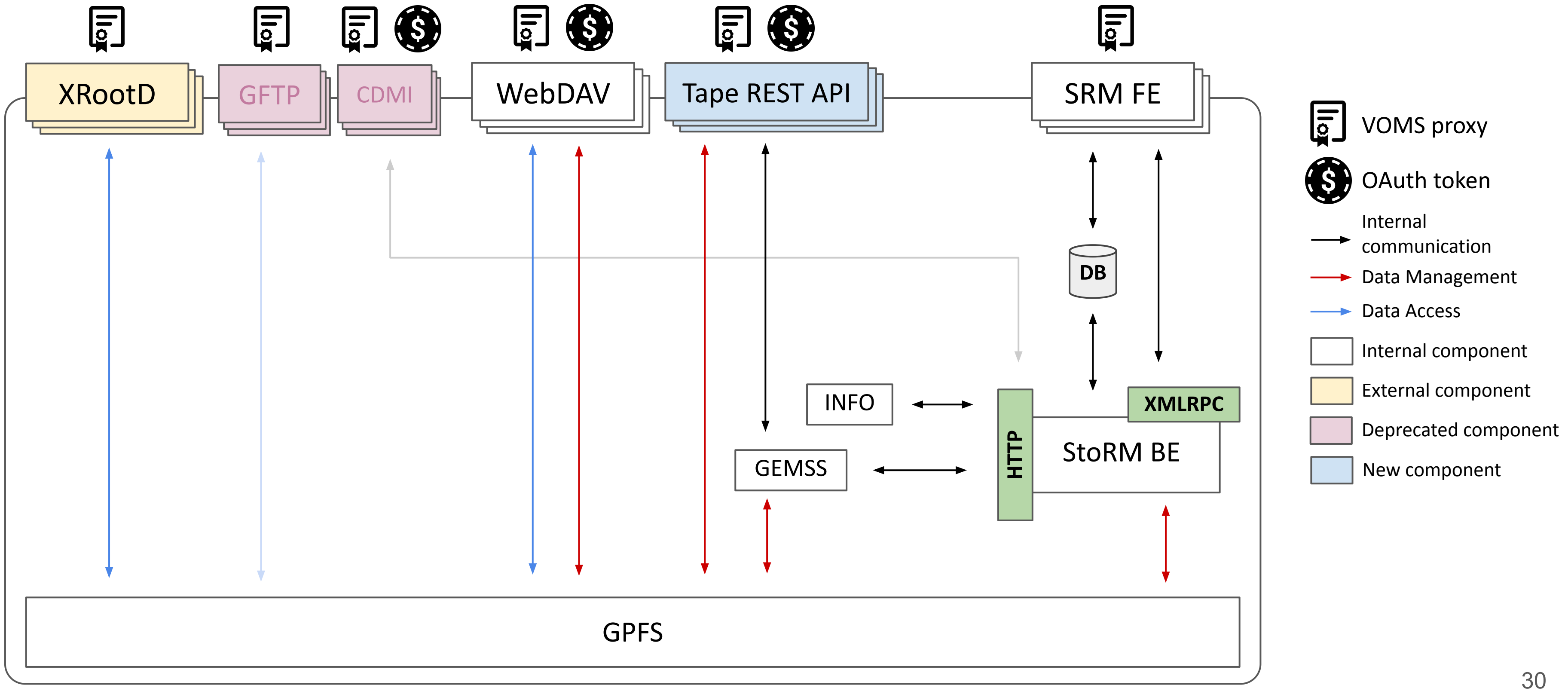
Done:

- Implemented HTTP request/response for all the API endpoints defined in the WLCG specification
- Packed the entire service in a Docker image
- Implemented the authZ workflow using NGINX and OPA

Certainly missing:

- Persistency
- Interaction with GEMSS for tape recall:
    - Provide an internal endpoint for GEMSS to replicate the current interaction with StoRM BE
- Finalize token/proxy management directly in NGINX

StoRM: Tape REST API deployment architecture



Towards new deployment architectures

Towards new deployment architectures

We’re moving towards new deployment scenarios

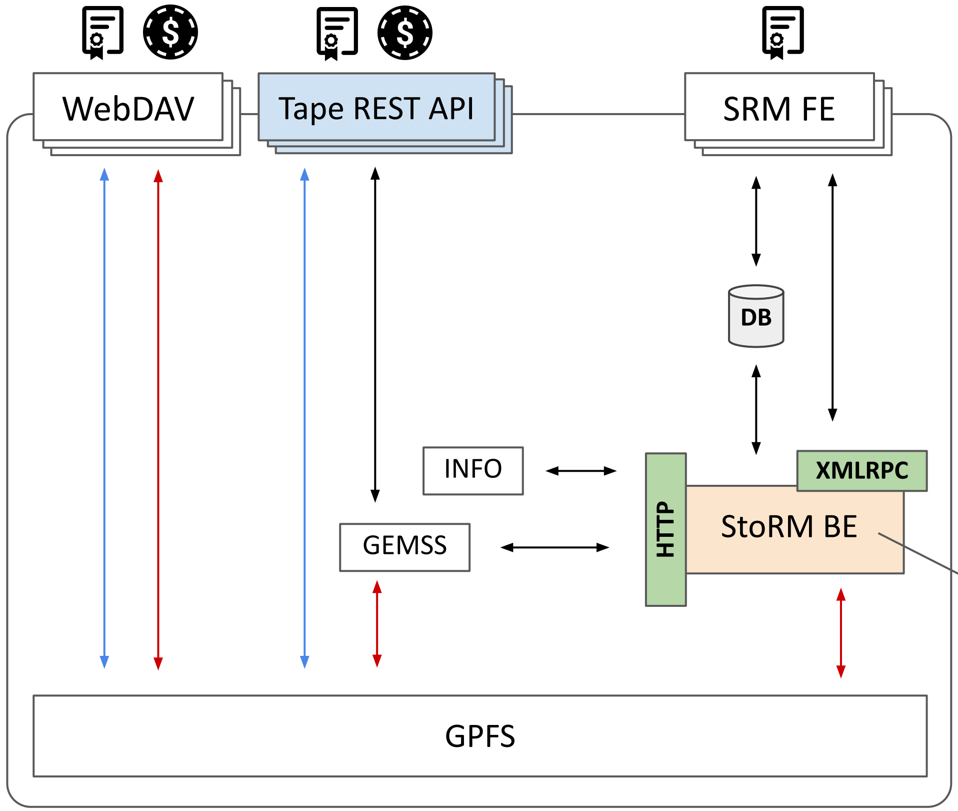
All Globus GridFTP will be turned off soon

The introduction of the StoRM Tape REST API will allow to have a valid alternative to SRM BringOnLine and will allow tape enabled no-SRM deployments

no-SRM deployments will mean that we will have no need for StoRM Frontend and also no need of a very big part of current StoRM Backend

StoRM: Future deployment architectures (1)

no-GridFTP scenario

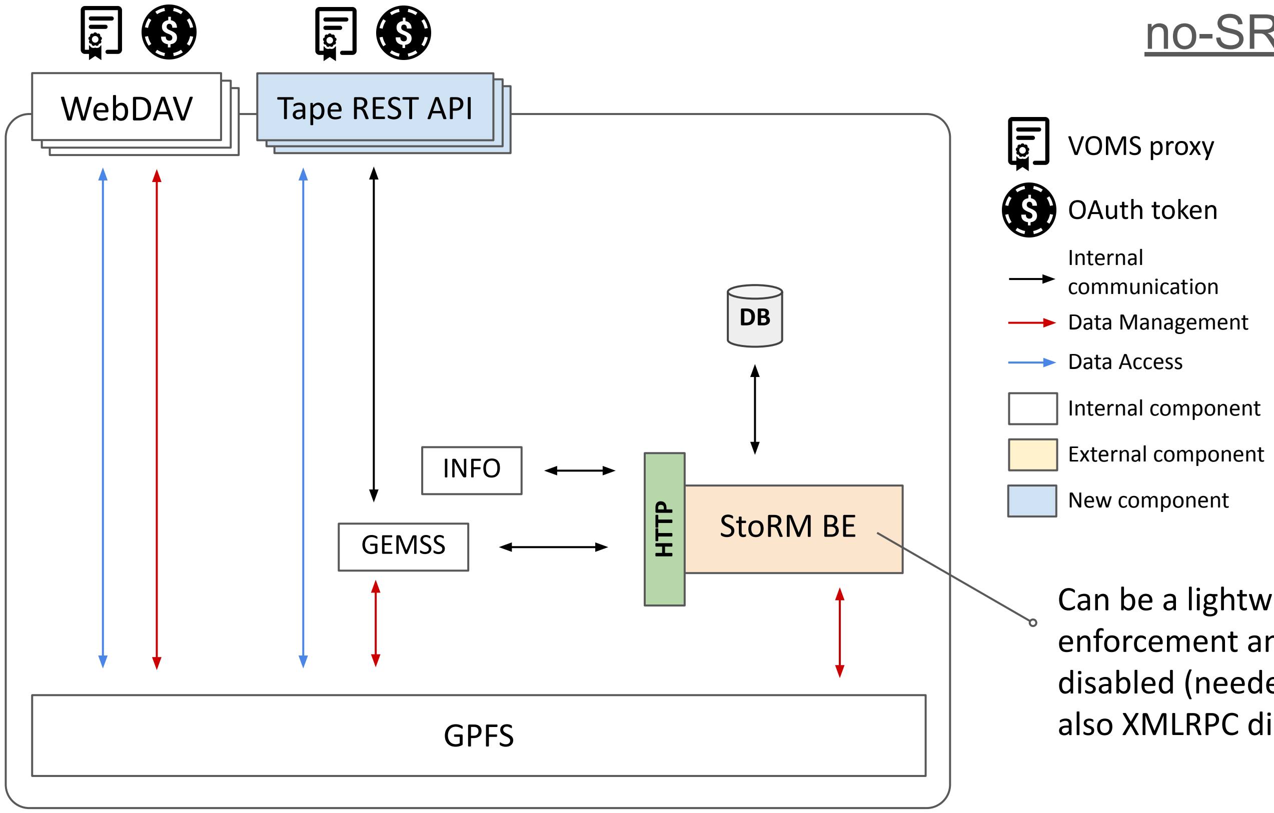


🗎	VOMS proxy
●	OAuth token
→	Internal communication
→	Data Management
→	Data Access
□	Internal component
🟨	External component
🟦	New component

Can be a lightweight BE with ACL enforcement and LCMAPS mapping disabled (needed only by GridFTP)

StoRM: Future deployment architectures (2)

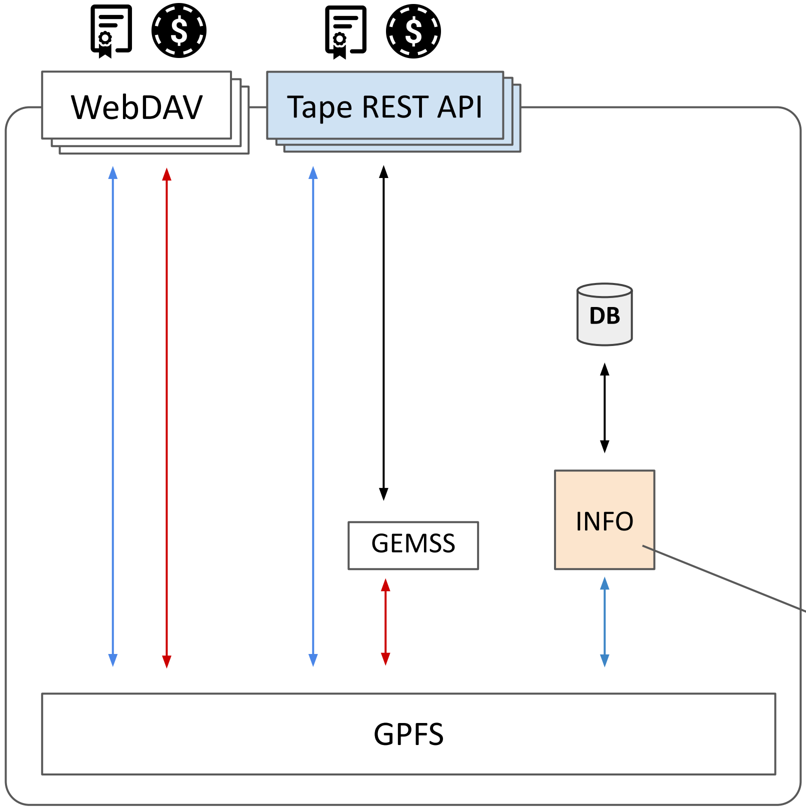
no-SRM scenario (1)



Can be a lightweight BE with ACL enforcement and LCMAPS mapping disabled (needed only by GridFTP) and also XMLRPC disabled (needed by FE)

StoRM: Future deployment architectures (3)

no-SRM scenario (2)



	VOMS proxy
	OAuth token
→	Internal communication
→	Data Management
→	Data Access
□	Internal component
🟨	External component
🟦	New component

A new component dedicated to manage storage info (used space, etc.) and publish info on BDII

Thanks! Questions?

Contacts and references

GitHub: https://github.com/italiangrid/storm

Documentation: http://italiangrid.github.io/storm/

Contacts:

storm-support@lists.infn.it and storm-users@lists.infn.it for users support

storm-devel@lists.infn.it for developers

Backup slides

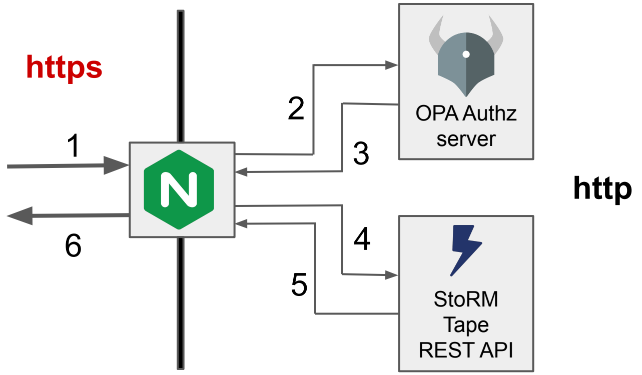
NGINX + OPA Authorization - Example

Example: a user of the WLCG experiment ALPHA wants to stage file `/storage/alpha/data` from tape to disk.

StoRM Tape REST API component is deployed at storm-tape.test.example and requires an access token of a user which is member of the group **wlcg/xfers**:

1. User sends a POST to the **stage** endpoint the providing the access token as a Bearer token:

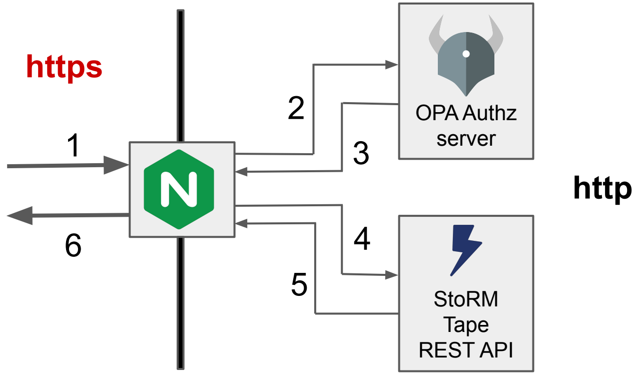
```
curl -d @stage_request.json -H 'Authorization: Bearer <access_token>'
https://storm-tape.test.example:8443/api/v1/stage
```



NGINX + OPA Authorization - Example

2. NGINX processes the request and forwards it to OPA authZ server
3. OPA evaluates its access policies and sends the authZ decision to NGINX.

Example of a simplified configuration:



```
{
    "roles" : {
        "/wlcg/xfers" : [
            "stage",
            "get_progress",
            "delete",
            "cancel"
        ],
        "/wlcg" : [
            "get_progress",
            "archiveinfo"
        ]
}
```

→ JSON with list of authorized operation per group

OPA policy file ←

```
allow {
    check_permission_jwt {...}
    check_time_valid
    check_signature
}
```

```
check_permission_jwt {
    data.roles[payload["wlcg.groups"][_]][_]
        == input.operation
}
```

NGINX + OPA Authorization - Example

```
allow {
    check_permission_jwt
    check_time_valid
    check_signature
}
```

Is the user allowed to perform operation? ✅

AND

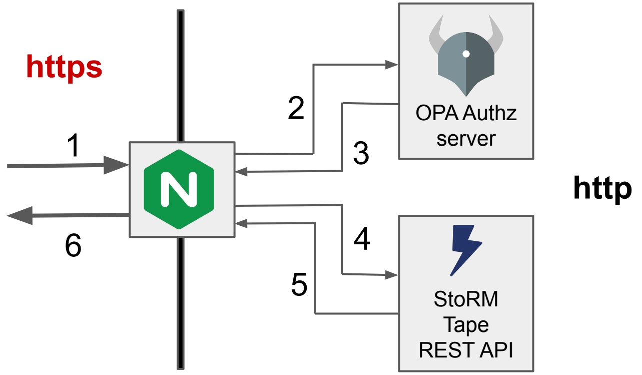
Is the token not expired? ✅

AND

Is the token authentic? ✅

||

`allow == true`



4. NGINX receives a response from OPA and processes it.
   Since we receive **allow == true**, the request is finally passed to the Tape REST API service.

NGINX + OPA Authorization - Example

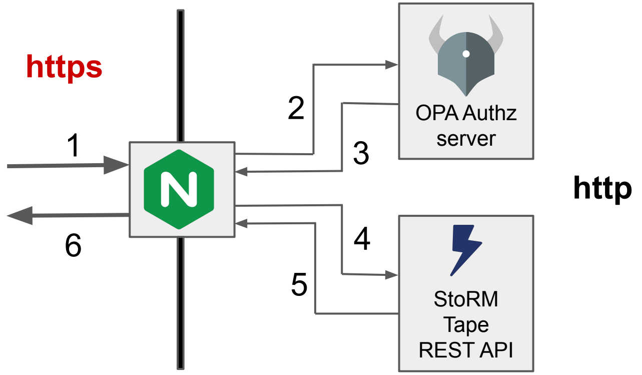
5. (and 6.) the response of the Tape REST API service is sent to the user

HTTP/1.1 201 Created

Location: https://storm-tape.test.example/api/v1/stage/**318640a8-424e-4071-adb8-abefad1bdbb3**

…

{"requestId":"**318640a8-424e-4071-adb8-abefad1bdbb3**"}



NGINX + OPA Authorization - Example

Missing OPA rego from the example proposed:

```
# Extract token from header
token := t {
   v := input.token
     startswith(v, "Bearer ") # Making sure it's the bearer token of the request
     t := substring(v, count("Bearer "), -1) # Take the token
}

# Extract the payload from the token
payload := p {
   [_, p, _] := io.jwt.decode(token) # Decode
}
```

ARE YOU SURE THIS IS HOW WE GET DATA INTO THE CLOUD?
IT