

The eXtreme-DataCloud project



Data Management for extreme scale computing



Daniele Cesini

With contribution by: Patrick Fuhrmann, Lukasz Dutka,
Cristina Duma, Alessandro Costantini, Oliver Keeble,
Fernando Aguilar, Giacinto Donvito

info@extreme-datacloud.eu



eXtreme DataCloud is co-funded by the Horizon2020
Framework Program – Grant Agreement 777367
Copyright © Members of the XDC Collaboration, 2017-2020

XDC Objectives

- ✘ The eXtreme DataCloud is a software development and integration project
- ✘ Develops **scalable** technologies for federating storage resources and managing data in highly distributed computing environments
 - ☛ Focus efficient, policy driven and Quality of Service based DM
- ✘ The targeted platforms are the current and next generation e-Infrastructures deployed in Europe
 - ☛ European Open Science Cloud (EOSC)
 - ☛ The e-infrastructures used by the represented communities

XDC Foundations

✘ XDC take the move from

- ☛→ the INDIGO Data management activity
- ☛→ the experience of the project partners on data-management

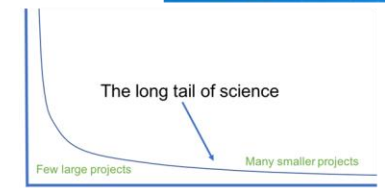
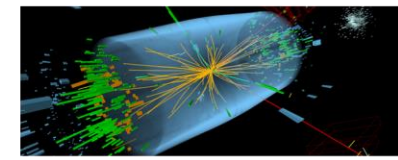
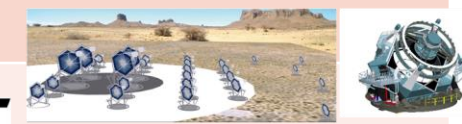
✘ Improve already existing, production quality, Federated Data Management services

- ☛→ By adding **missing functionalities** requested by research communities
- ☛→ Must be coherently harmonized in the European e-Infrastructures
- ☛→ **TRL 6+ → TRL8** (as requested by the H2020 call)

XDC Consortium



ID	Partner	Country	Represented Community	Tools and system
1	INFN (Lead)	IT	HEP/WLCG	INDIGO-Orchestrator
2	DESY	DE	Research with Photons (XFEL)	dCache
3	CERN	CH	HEP/WLCG	EOS, DYNAFED, FTS
4	AGH	PL		ONEDATA
5	ECRIN	[ERIC]	Medical data	
6	UC	ES	Lifewatch	
7	CNRS	FR	Astro [CTA and LSST]	
8	EGL.eu	NL	EGL communities	

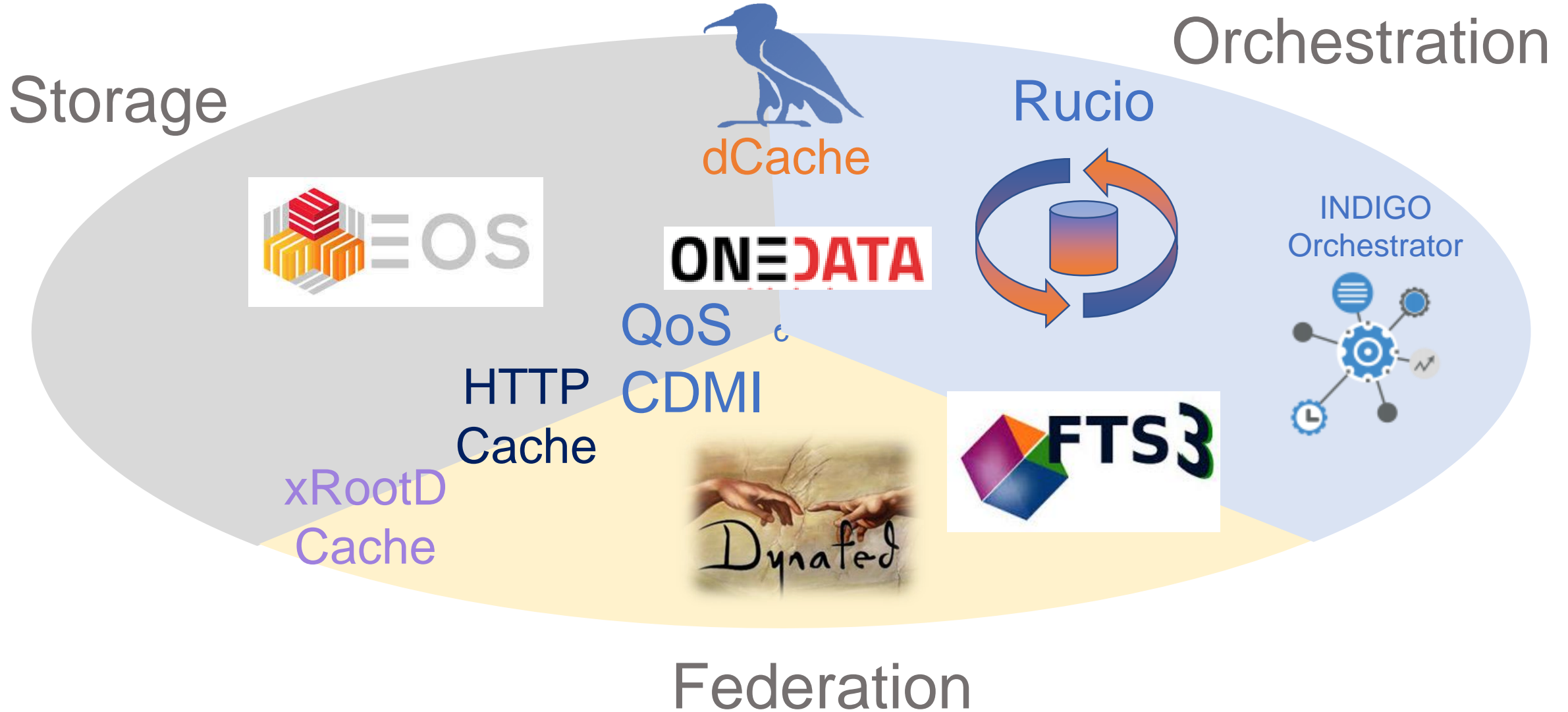


- ✗ 8 partners, 7 countries
- ✗ 7 research communities represented + EGI
- ✗ XDC Total Budget: 3.07Meuros
- ✗ XDC started on Nov 1st 2017 – will run for 27 months until Jan 31st 2020

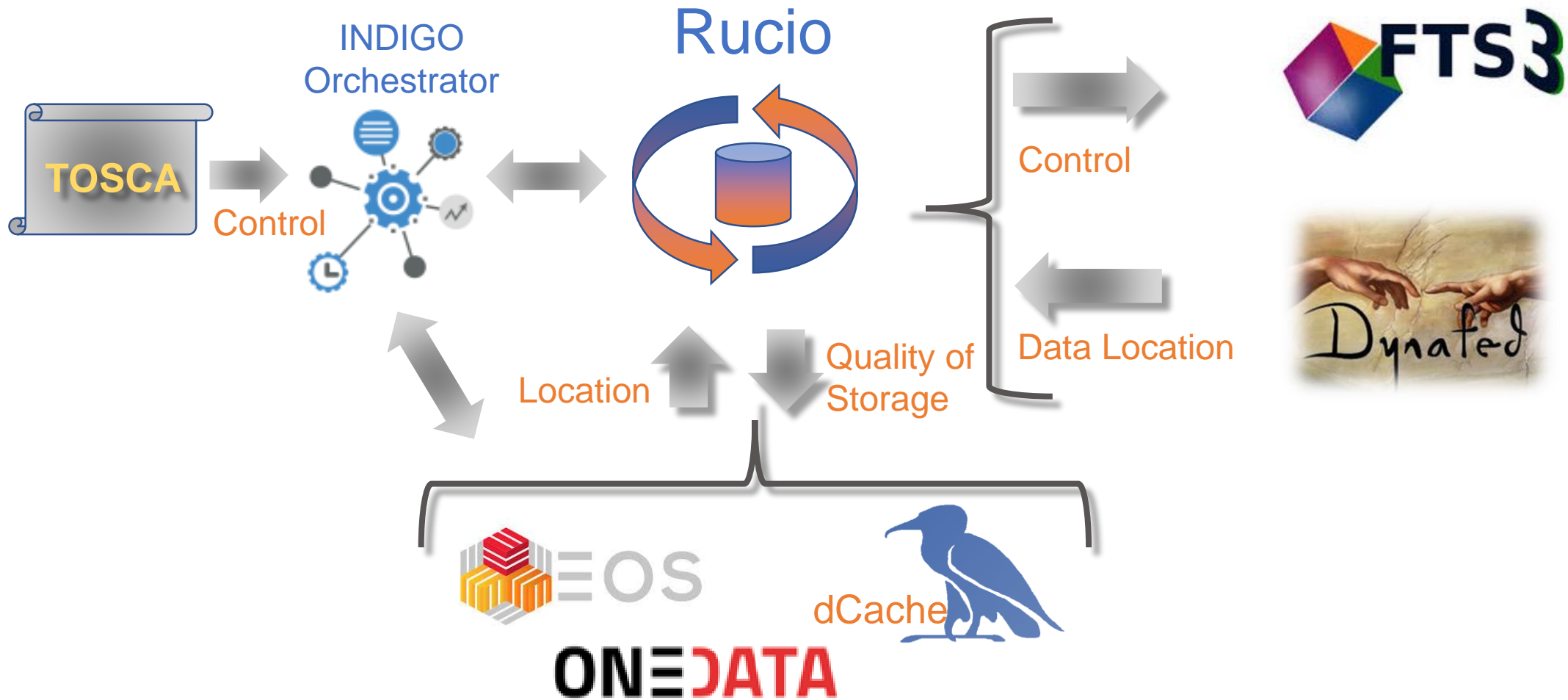
XDC Technical Topics

- ✘ Intelligent & Automated Dataset Distribution
 - ⋯→ Orchestration to realize a policy-driven data management
 - ⋯→ Data distribution policies based on Quality of Service (i.e. disks vs tape vs SSD) supporting geographical distributed resources (cross-sites)
 - ⋯→ Data lifecycle management
- ✘ Data pre-processing during ingestion
- ✘ Smart caching
 - ⋯→ Transparent access to remote data without the need of a-priori copy
- ✘ Metadata management
- ✘ Data management based on access patterns
 - ⋯→ Move to 'glacier-like' storage unused data, move to fast storage "hot" data
 - ⋯→ at infrastructure level
- ✘ Sensitive data handling
 - ⋯→ secure storage and encryption

XDC Toolbox

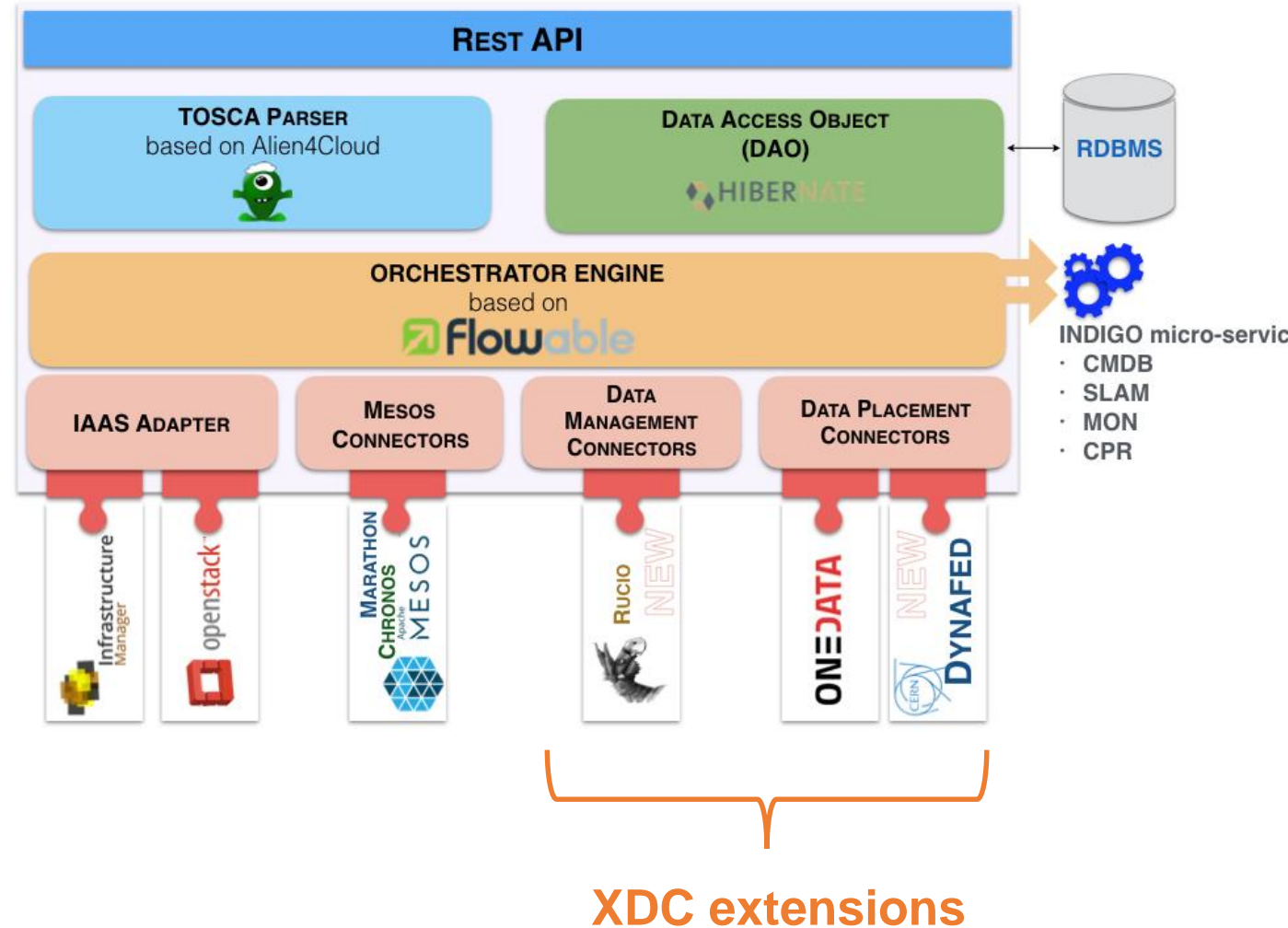


Expected Control Flow for Orchestration



INDIGO Orchestrator Overview

- ✗ The Orchestrator accepts TOSCA requests
- ✗ It can run three main workflows in the PaaS:
 - ➡ Deploy
 - ➡ Undeploy
 - ➡ Update
- ✗ The Orchestration layer is being extended in order to address the new requirements:
 - ➡ move data between distributed storages
 - ➡ specify different QoS for replicas
 - ➡ launch and monitor user defined processing jobs at ingestion time



Quality of service in storage








- ✘ Modern Storage offers go beyond the WLCG model of disk and tape
 - ☛→ Often the actual cloud storage technology is not even known to the end user or might change over time
 - ☛→ Mostly only quality attributes are known, like Glacier:
 - ☛→ Durability (Retention Policy): 99.999999999%
 - ☛→ Access Latency : a) 5 minutes, b) 5 hours, c) 12 hours

- ✘ XDC aims to provide
 - ☛→ help to standardize the different attributes
 - ☛→ a prototype of rendering those attributes through a network protocol
 - ☛→ a set of reference implementations to be used by the Orchestration system

- ✘ Work is based on INDIGO-DataCloud prerequisites

Work in progress for QoS

- ✗ Definition currently based on a RDA working group
- ✗ "Cloud Data Management Interface" chosen as control protocol.
 - ➡ Defined by SNIA.
 - ➡ INDIGO acknowledged by SNIA as contributor to the reference implementation.
- ✗ Implementing the defined API into GFAL, dCache, EOS and StoRM

	Name	Access Latency [ms]	Number of Copies	Storage Lifetime	Location	Storage type	Available Transitions
	disk	100	1		DE	Processing	tape, disk+tape
	disk+tape	100	2		DE	Processing	tape
	DiskAndTape	50	3	20 years	DE	Processing	
	DiskAndTape	50	2		IT	Processing	
	DiskOnly	50	3	20 years	DE	Processing	
	DiskOnly	50	1		IT	Processing	
	profile1	10	3	20 years	DE	Processing	

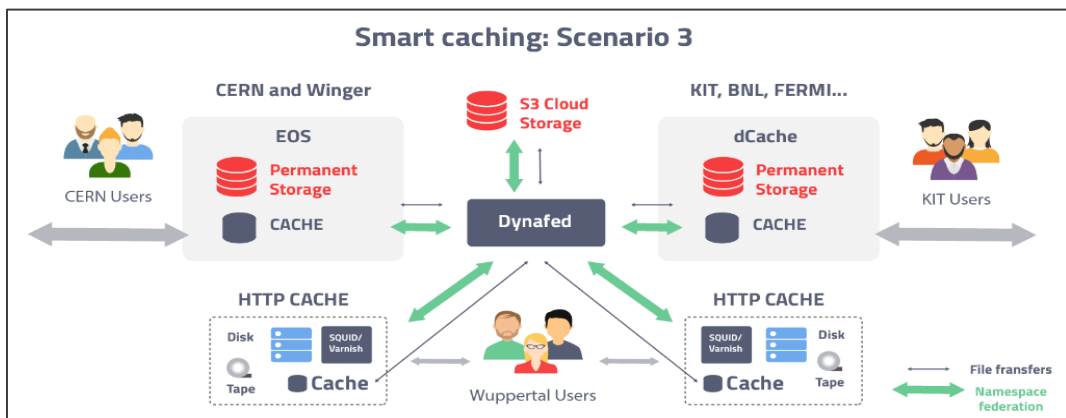
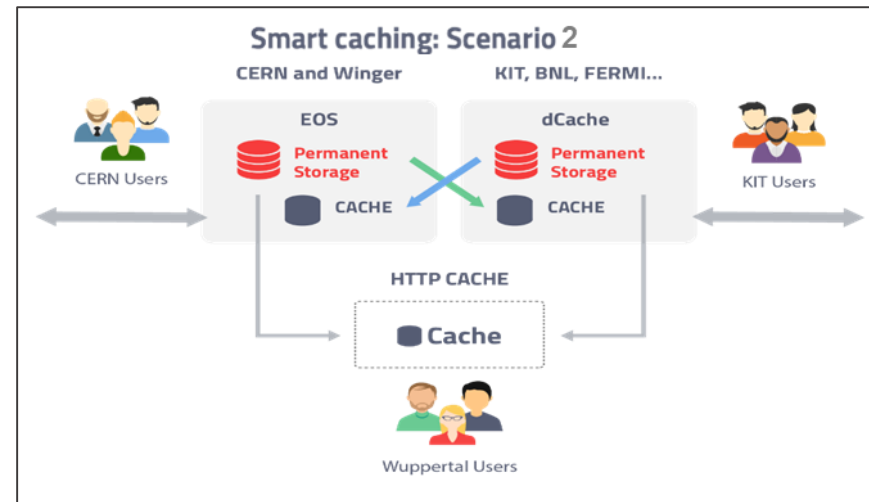
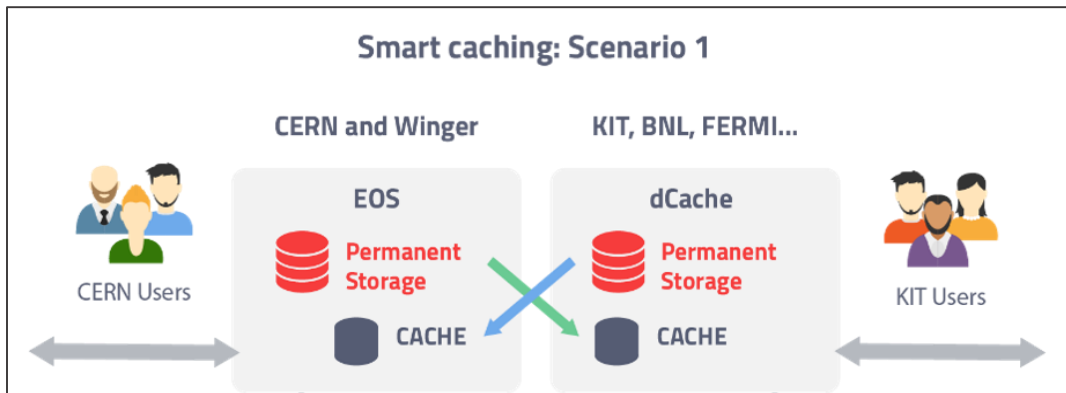
From INDIGO-DataCloud

- Access Latency [ms]
- Number of Copies
- Storage Lifetime
- Location
- Available Transitions

Smart caching

Smart caching

- Develop a global caching infrastructure supporting the following building blocks:
 - dynamic **integration of satellite sites** by existing data centres
 - creation of standalone caches modelled on existing **http and xrootd** solutions
 - federation of the above to create a large scale, **regional caching infrastructure**

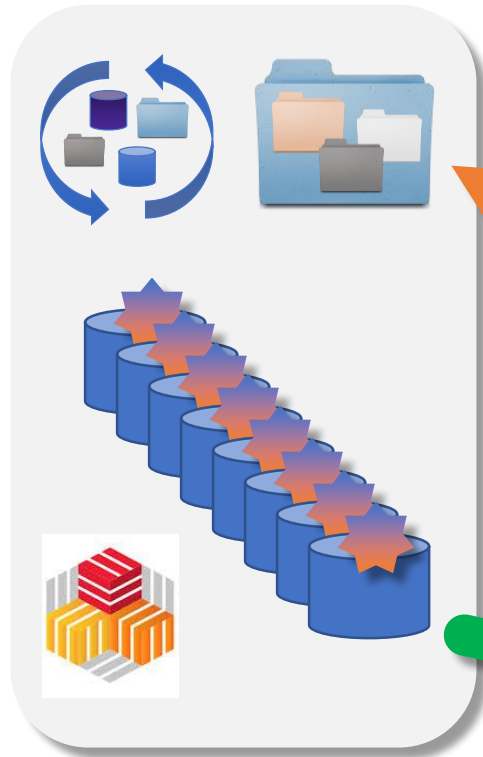


For an use case implementing XDC results on Scenario2 using xcache in collaboration with CMS and HNSciCloud see D.Spiga presentation on Monday: «Exploiting private and commercial clouds to generate on-demand CMS computing facilities with DODAS»

<https://indico.cern.ch/event/587955/contributions/2937198/attachments/1682105/2702791/CHEP-2018-Spiga.pdf>

EOS-dCache integration with cached access

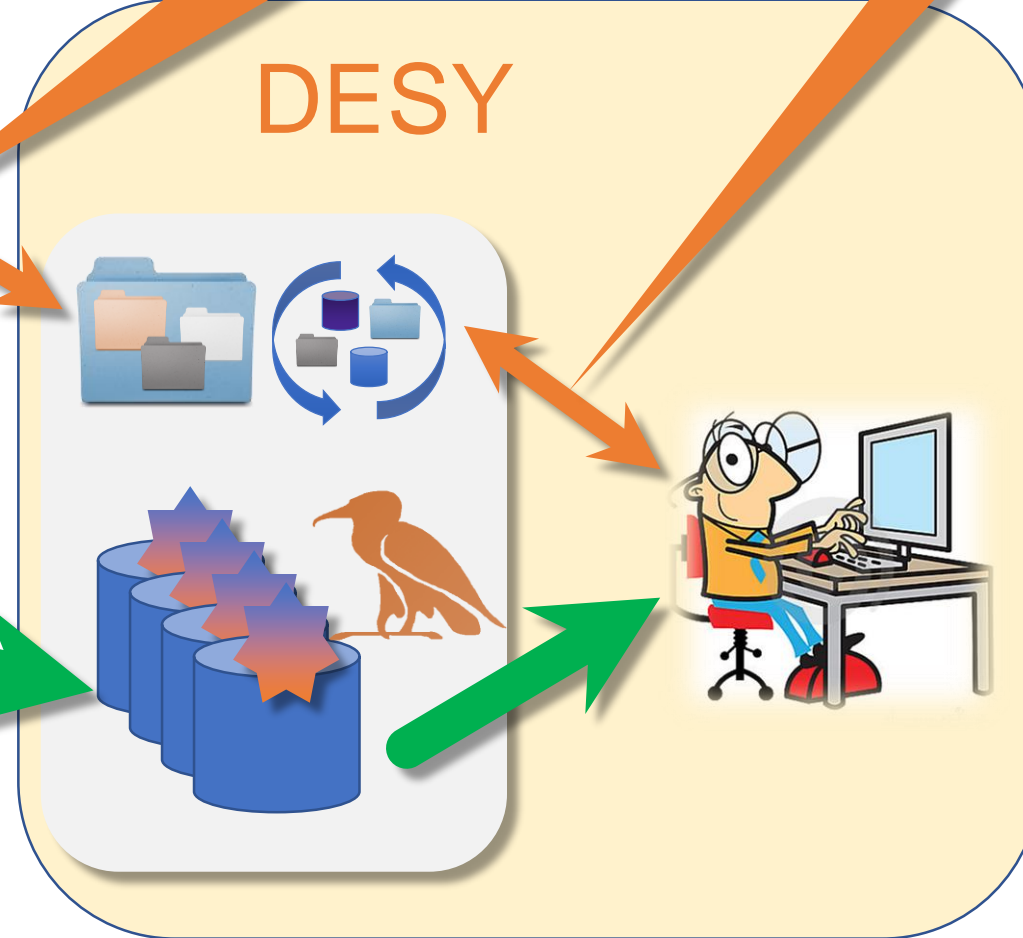
CERN



Sync Namespace

Request

DESY



Advantages

- ✗ Same software stack as we currently have at the sites.
- ✗ After the data has been transferred to the local storage system, a name space entry has been created locally and the data is available at the local site independently of the remote network link and the availability of the central service.

Data in Hybrid Cloud Environments: Onedata

✘ ONEDATA is a storage federator that allows to use resources backed by providers worldwide

→ Providers deploy *Oneprovider* services near physical storage resources

→ Users use *Onezone* web interfaces

→ APIs available

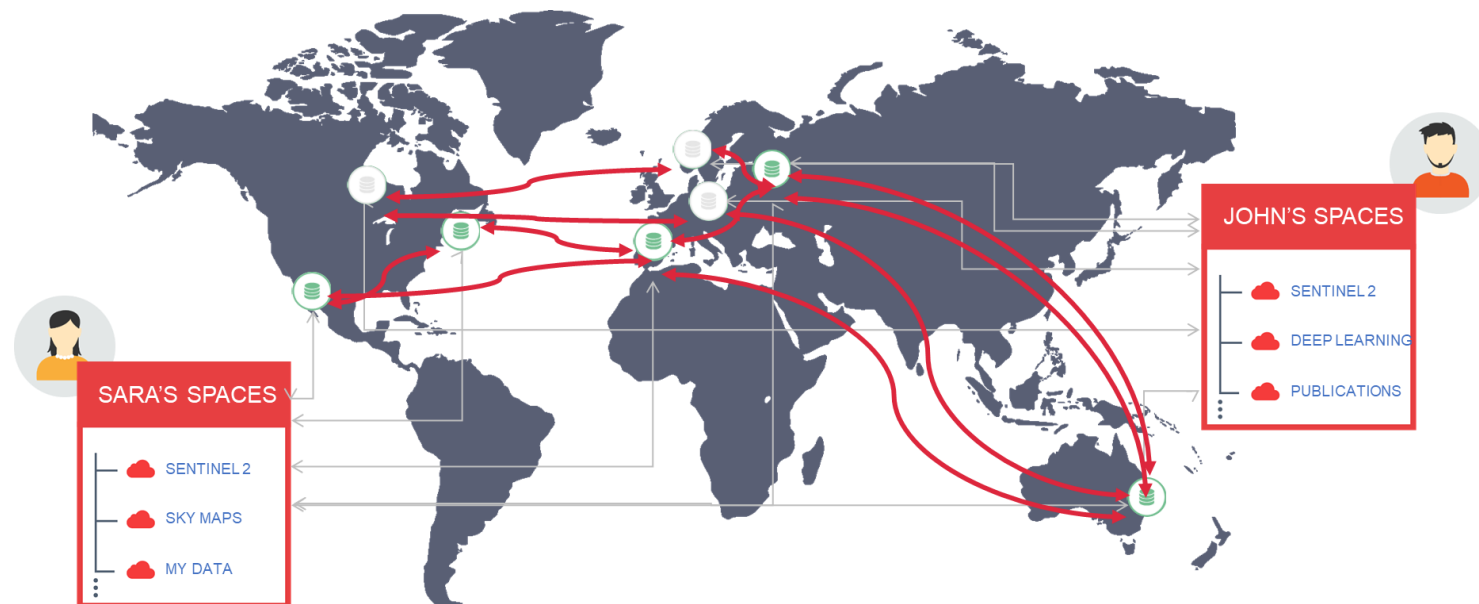
→ Local mounting on users machines available

→ Storage is organized into **Zones**

→ federations of providers

→ enable the creation of closed or interconnected communities

<https://onedata.org>

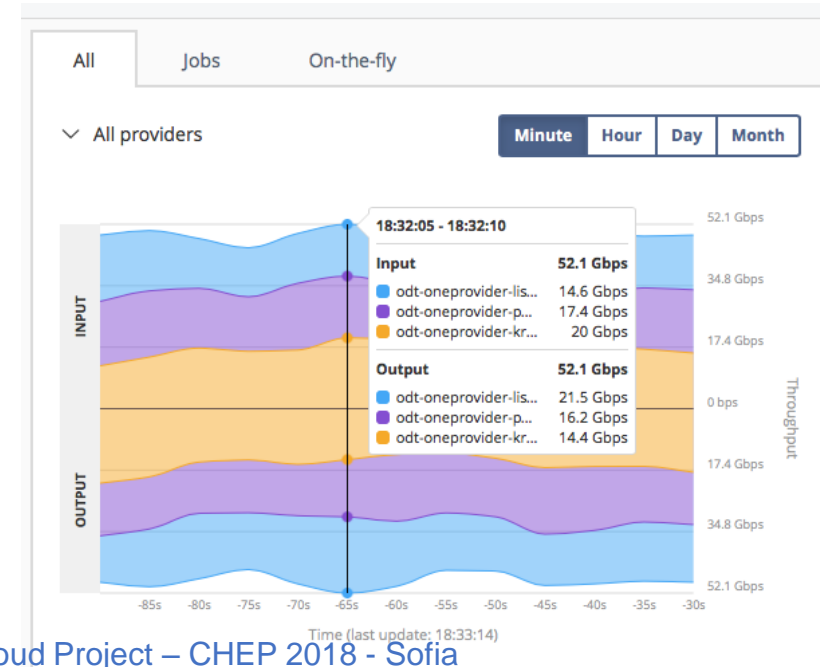
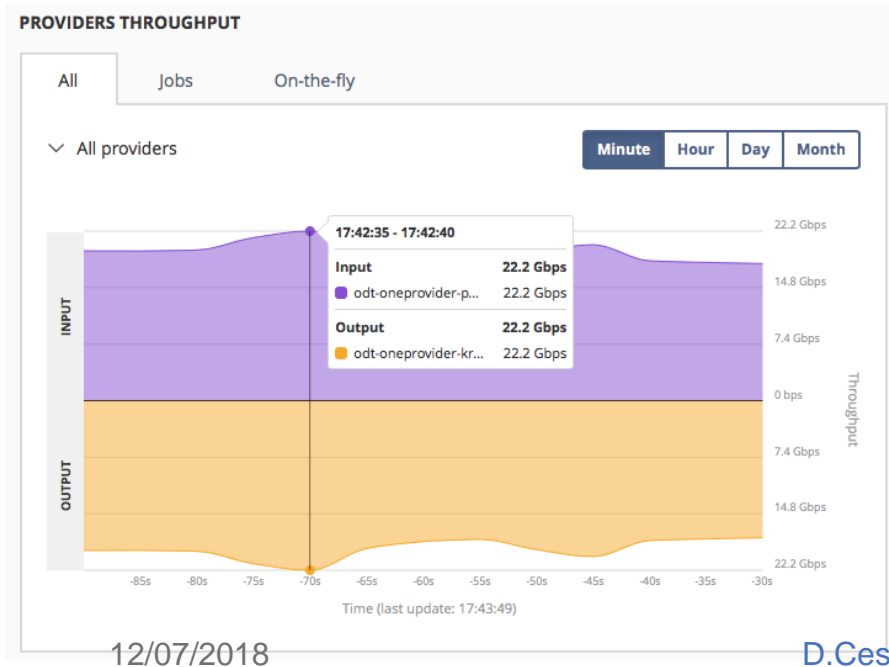


Onedata XDC Goals

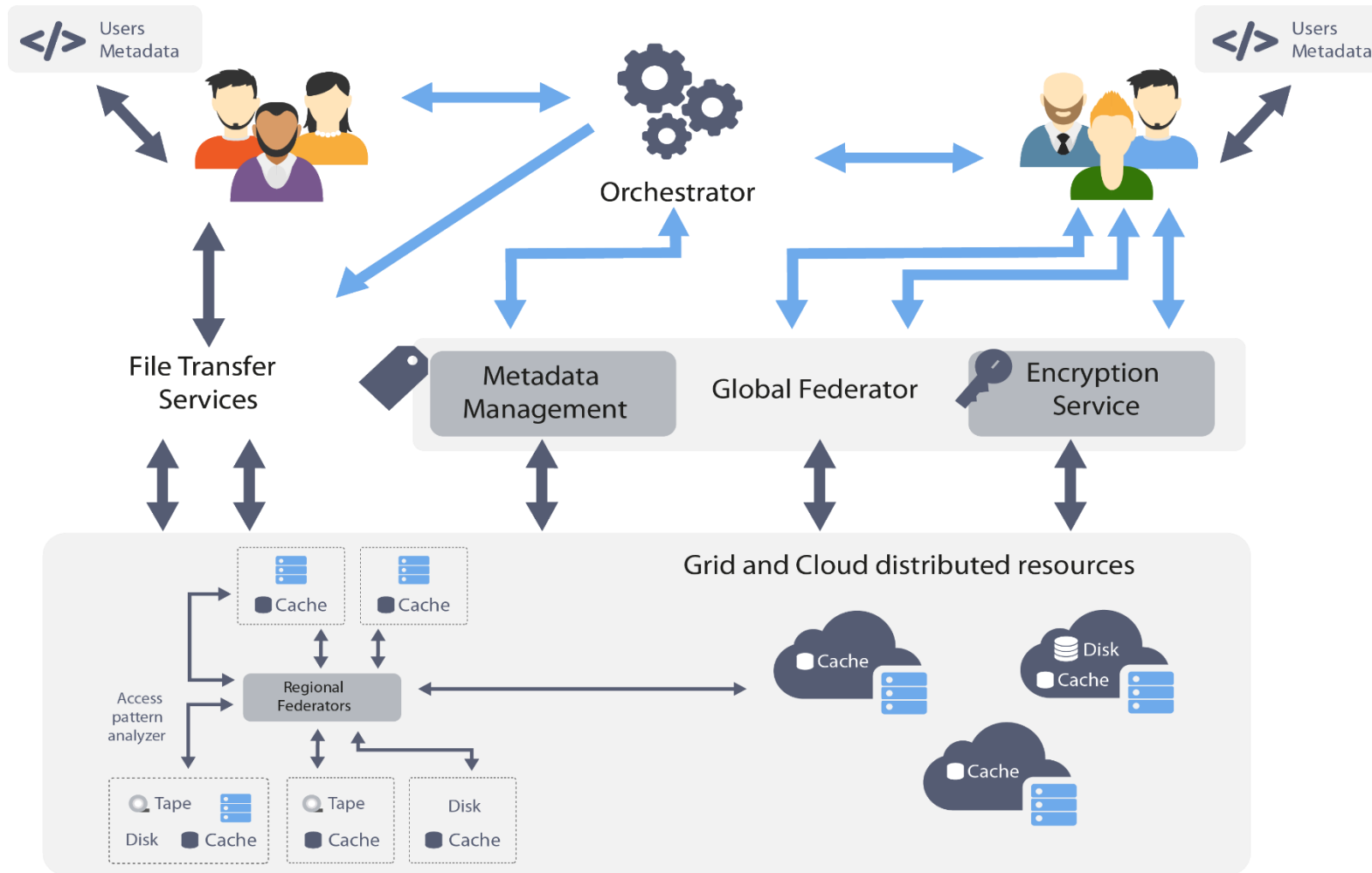
- ✘ Improve scalability and stability
 - ☛ Provide unified Exascale data access and management platform for processing on multi-cloud infrastructures
- ✘ Next Generation Replica Management Engine
 - ☛ Multi level priority queues and Adaptive buffering and TCP/IP connection management
 - ☛ Weighted source selection in case of many possible sources
- ✘ Enable advanced metadata mechanisms supporting legacy metadata solutions and open data publishing
 - ☛ Use cases from ECRIN, CTA, LIFEWATCH
- ✘ Ensure secure data storage and provisioning for sensitive data sets
- ✘ Smart Cache Management and Advanced file popularity index
- ✘ QoS management at federation and data centre level

Next Generation Replica Management Engine Released

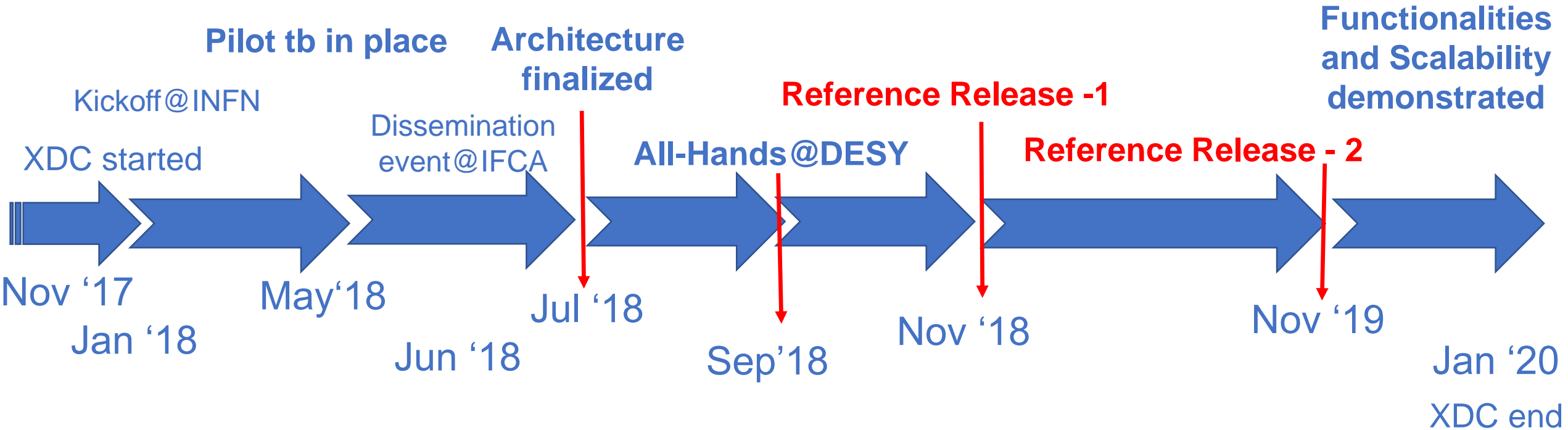
- ✗ External tests were done in OTC infrastructure under Helix Nebula Science Cloud
 - Heavily tested on real infrastructures
 - Very close to iperf3 – multistream throughput ~95-120% tested on several long distance real infrastructures
- ✗ Current results shows we are able to transfer data between two providers at the speed of ~2Gbps/core. The communication and data transfers are encrypted.



XDC high level architecture



The Plan for the Next Months



XDC Contacts

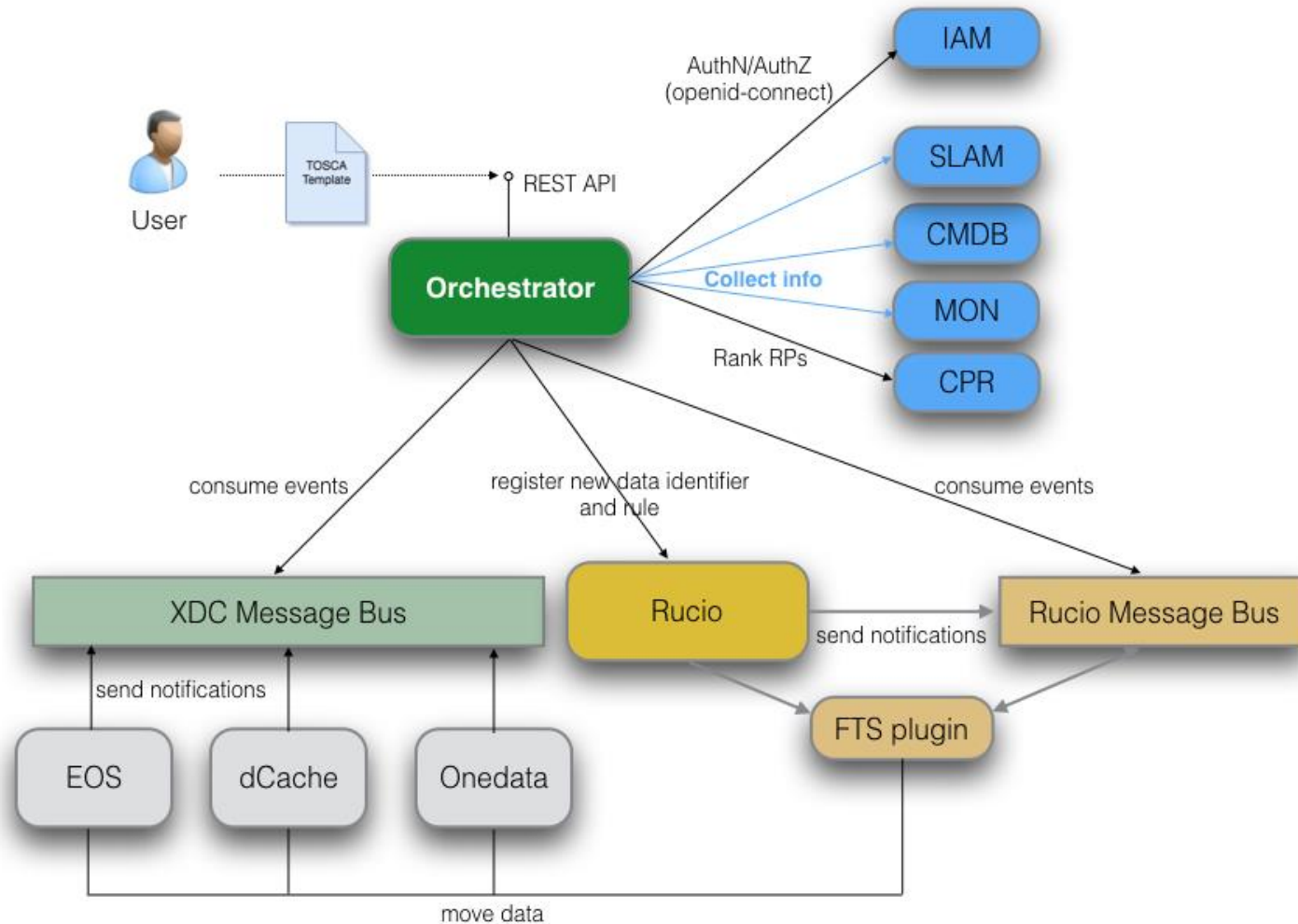
✂ Website: www.extreme-datacloud.eu

✂ [@XtremeDataCloud](https://twitter.com/XtremeDataCloud) on Twitter

✂ Mailing list: [info<at>extreme-datacloud.eu](mailto:info@extreme-datacloud.eu)

Backup slides

Orchestration System draft architecture



Reference workflow I: DLC management

1. The user submits the DLC policy request to the Orchestrator, specifying:
 - Either the data identifier, in case the file is already known to Rucio or the physical location of the data if not. In the latter case, the Orchestrator has to register the data into Rucio before it proceeds.
 - The DLC policies, which are the initial and time dependent attributes of the data.
2. The Orchestrator calls Rucio on behalf of the user and creates the replication rule using the information provided by the user.
3. Rucio orchestrates the data movement using the input information provided by the Orchestrator to create the data transfer requests (e.g. to copy the data to some specified endpoint).
4. The transfer tool (e.g. FTS) initiates and monitors the requested transfers. Depending on the set up, the transfer tool may choose from a set of source locations
 - by using its transfer history to find the best performing data source
 - and by evaluating the network topology.
5. Rucio continuously compares its rules against reality until the rule is manually deleted or becomes obsolete based on its time dependencies.
6. At any time after the request was submitted to the Orchestrator, the user can query the Orchestrator for:
 - A list of rules submitted.
 - Activity details, including the status, on a specific rule.

Reference workflow II: Preprocessing on data ingestion

1. The user submits his workflow request to the Orchestrator including the following information:
 - the space to watch for incoming data
 - the application to be run on incoming data
 - the replication rule to be enforced on the incoming or preprocessed data.
2. The storage system, holding the watched storage, notifies the presence of new data by sending a message to the XDC message queue.
3. The INDIGO Orchestrator receives the notification and registers the ingested data file into Rucio, including the replica policy, specified by the user.
4. The Orchestrator selects the best compute site to perform the requested processing. To do so, it might collect information from different sources: CMDDB, SLAM, Monitoring, Storage endpoints.
5. The Orchestrator triggers a data movement through Rucio in order to copy the data to the selected compute center.
6. The Orchestrator gets notified on the completion of the data transfer, by listening to the Rucio Message Queue.
7. The Orchestrator triggers the processing Job by submitting the request to those computing clusters (Mesos/Kubernetes) available at the site.
8. As soon as the job output is produced, its availability is notified to interested parties, in particular Rucio, via the XDC message bus.
9. The data, generated by the processing step, is automatically registered into Rucio through the Orchestrator.
10. Rucio takes care that the policies requested for the raw and preprocessed data are applied.

Federation, Orchestration

CERN Dynafed



- ✘ Federates endpoints to a virtual global namespace.
- ✘ Supports http, WebDAV, S3

CERN FTS



- ✘ Orchestrates reliable wide area data transfers
- ✘ Supports multiple protocols, GridFTP, http, xrootd
- ✘ Can select most appropriate source
- ✘ Has knowledge on network topology and health

The INDIGO PaaS Orchestrator



- ✘ The INDIGO PaaS Orchestrator is a component of the PaaS layer that allows to **instantiate resources on Cloud Management Frameworks** (like **OpenStack** and **OpenNebula**) and **Mesos** clusters.
- ✘ It takes the deployment requests, expressed through **templates written in TOSCA** YAML Profile 1.0 and deploys them on the best cloud site available. In order to do that
 - it gathers SLAs, monitoring info and other data from other platform services,
 - it asks to the cloud provider ranker for a list of the best cloud sites.
- ✘ Exposes **REST APIs**

Stolen from the INDIGO PaaS Orchestrator GitHub

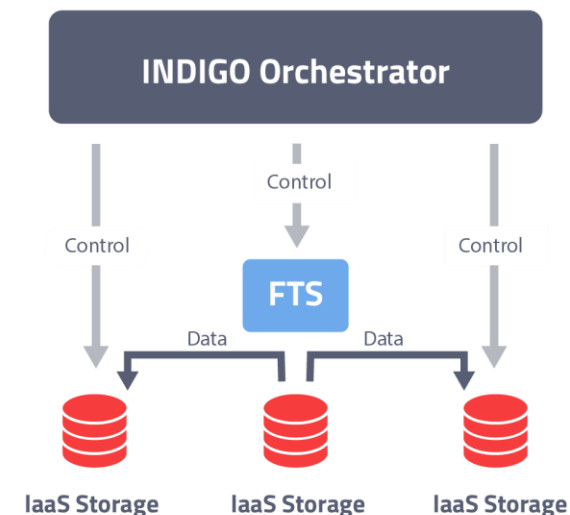
Policy driven Data Management

✘ Intelligent & Automated Dataset Distribution

☛ A typical workflow

- ☛ Initially the data will be stored on low latency devices for fast access
- ☛ To ensure data safety, the data will be replicated to a second storage device and will be migrated to custodial systems, which might be tape or S3 appliances
- ☛ Eligible users will get permission to restore archived data if necessary
- ☛ After a grace period, Access Control will be changed from “private” to “open access”

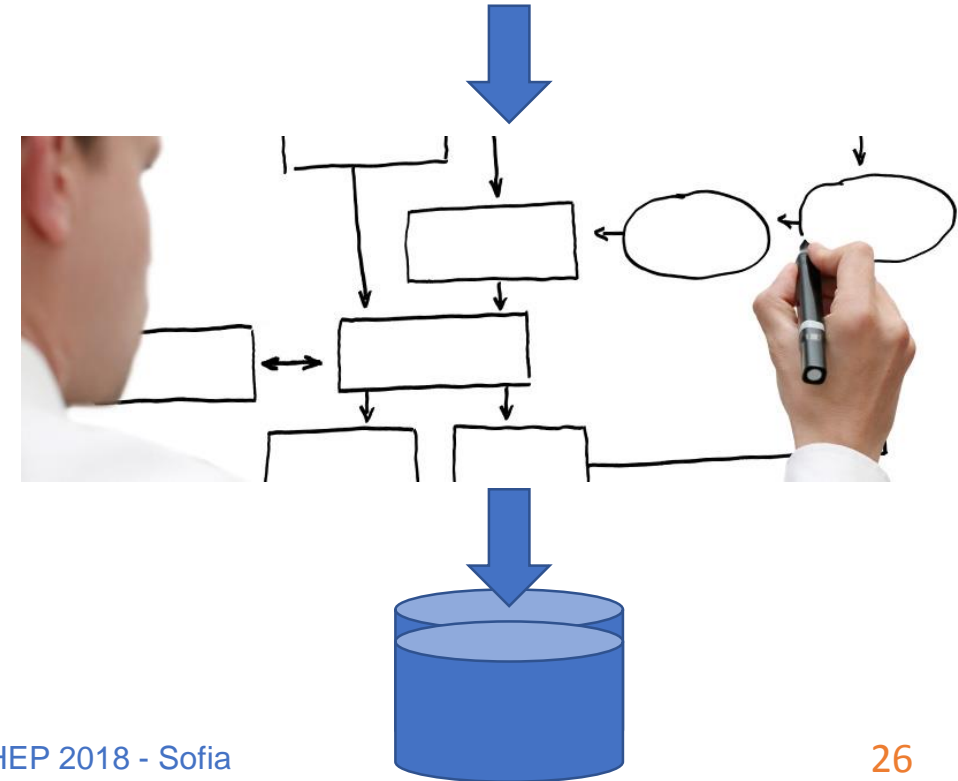
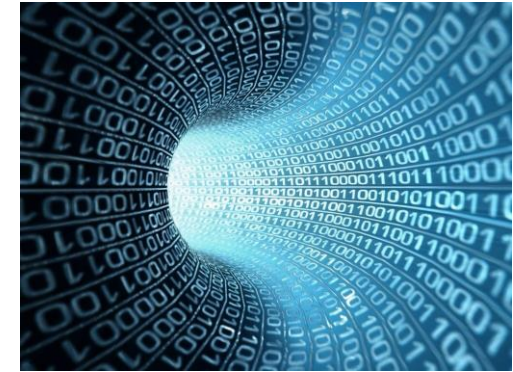
☛ Data management based on access pattern



Data pre-processing

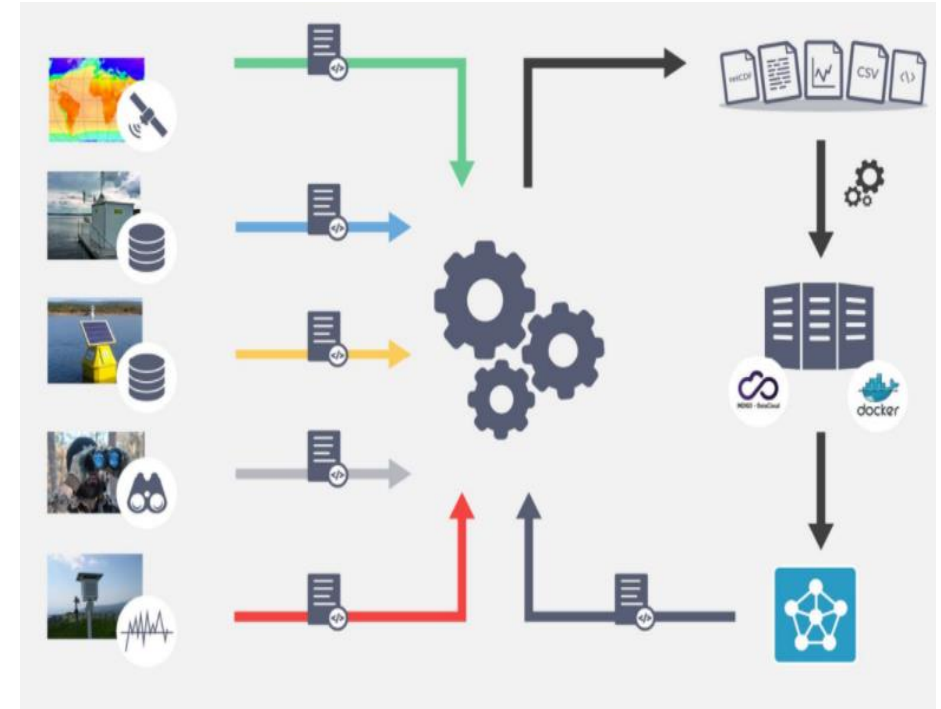
✘ Data pre-processing during ingestion

- ☛ Automatically run user defined applications and workflows when data are uploaded
 - ☛ i.e. for Skimming, indexing, metadata extraction, consistency checks
- ☛ Implement a solution to discover new data at specific locations
- ☛ Create the functions to request the INDIGO PaaS Orchestrator to execute specific applications on the computing resources on the Infrastructure
- ☛ Implement a high-level workflow engine, that will execute applications defined by the users
- ☛ Implement the data mover to store the elaborated data in the final destination



LifeWatch Use Case

- ✘ **Problem:** Life Cycle Management of data related to **Water Quality** involving **heterogeneous data sources**
 - Satellite, Real-time monitoring, meteorological stations.
- ✘ **Goal:** Integrate data sources and different types of modelling tools to simulate freshwater masses in a FAIR data environment
 - Use of standards like EML (Ecological Metadata Language)
- ✘ **XDC Solution:**
 - Onedata
 - Metadata management and discovery, Digital Identifier minting, storage
 - PaaS Orchestrator
 - automatic preprocessing for data harmonization and model deployment



CTA Use Case

✘ **Problem:** Complex and Big Data management in a distributed environment. Data quality Assurance

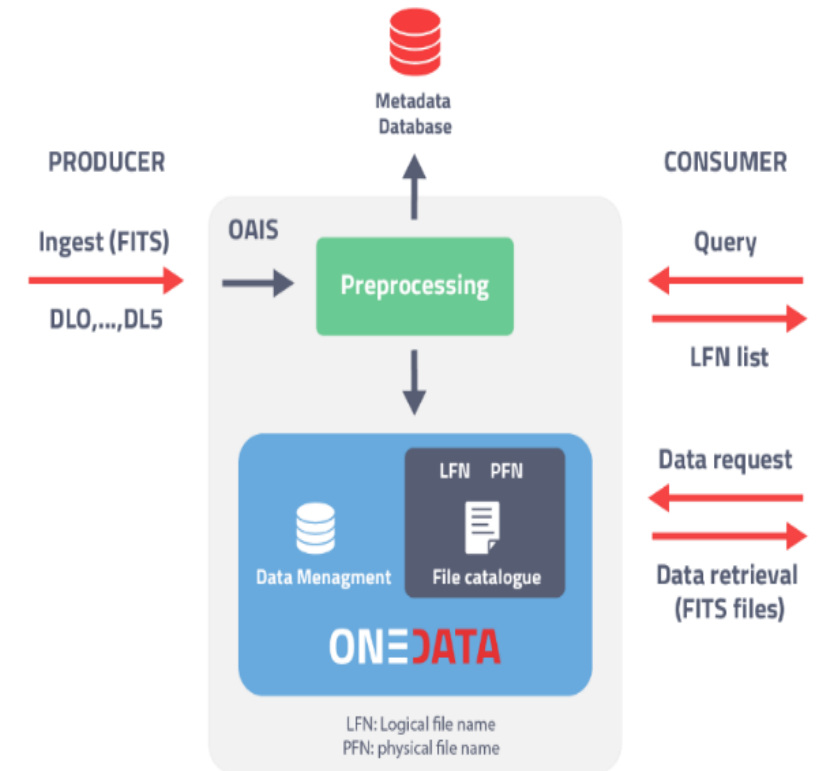
- ➡ The CTA distributed archive lies on the « Open Archival Information System » (OAIS) ISO standard.
- ➡ Event data are in files (FITS format) containing all metadata.

✘ **Goal:** Metadata are extracted from the ingested files, with an automatic filling of the metadata database.

- ➡ Metadata will be used for querying of archive.
- ➡ The system should be able to **manage replicas**, tapes, disks, etc, with data from low-level to high-level

✘ **XDC Solutions**

- ➡ Onedata
 - ➡ Metadata management and discoverability
- ➡ PaaS Orchestrator + QoS



ECRIN Use Case

- ✘ **Problem:** Distributed files and data objects across different repositories. Metadata heterogeneity. Sensitive Data
- ✘ **Goal:** Single environment to make clinical trial data objects available for sharing with others. Sources are spread over
 - a variety of access mechanisms
 - several different locations
 - growing number of general and specialised data repositories
 - trial registries
 - Publications
 - the original researchers' institutions
- ✘ **XDC Solution:** Onedata
 - Metadata management and discovery
 - Secure Storage

Next Generation Replica Management Engine Released

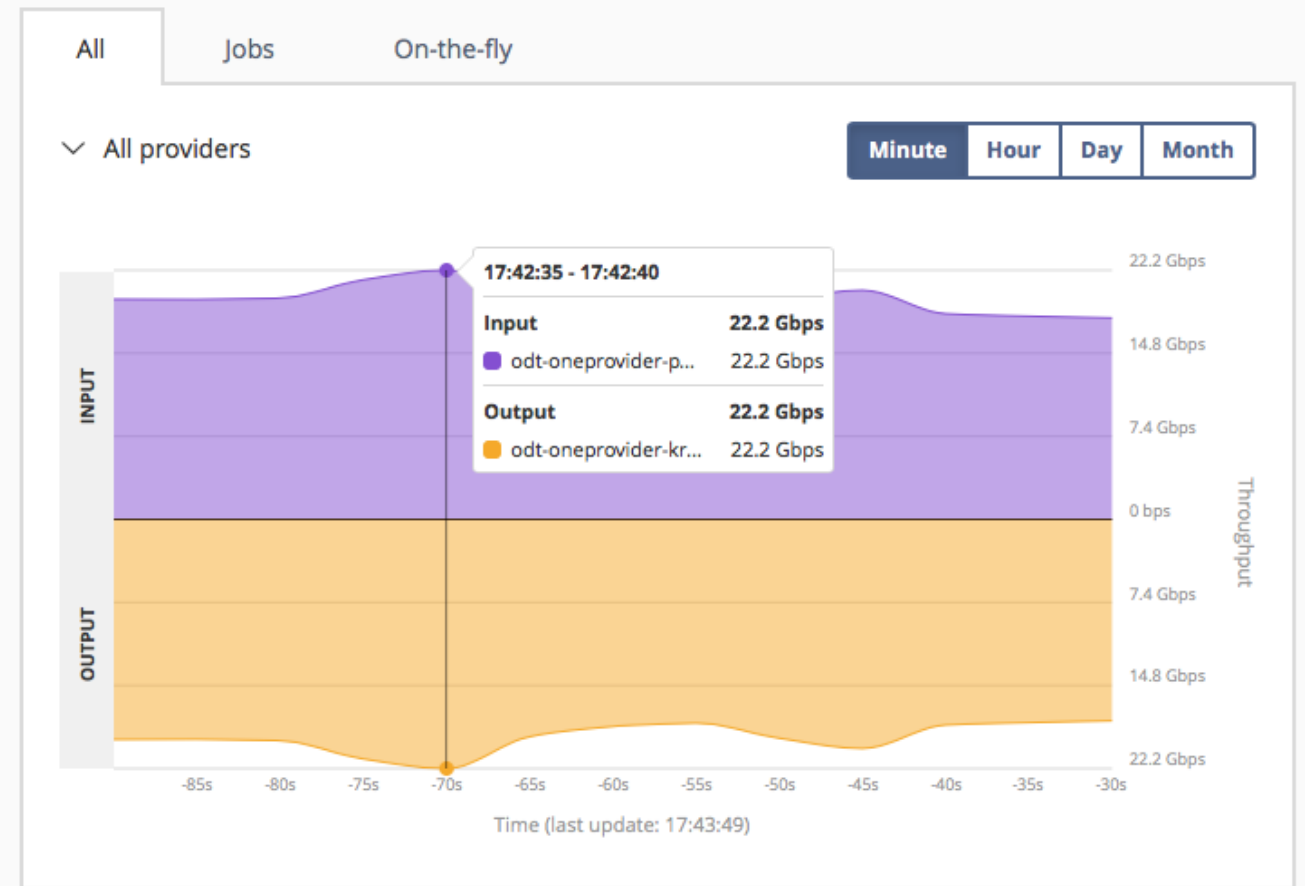


- ✘ External tests were done in OTC infrastructure under Helix Nebula Science Cloud
 - Heavily tested on real infrastructures
 - Very close to iperf3 – multistream throughput ~95-120% tested on several long distance real infrastructures
- ✘ Current results shows we are able to transfer data between two providers at the speed of ~2Gbps/core. The communication and data transfers are encrypted.
- ✘ QoS on the line, the RTRansfer system is now able to prioritize traffic on the existing tunnel and make fair network resources distribution amongst different requests.
- ✘ The combined overhead in terms of the latency is ~5ms on top of network connection latency – mainly due to scheduling algorithms for QoS
- ✘ The source and the target protection algorithm to now overload the resources and automatically adopts the request pace

DEDICATED CIRCUIT TESTS

- ✗ Two oneproviders
- ✗ Data is stored in one and replicated to the second one.
- ✗ Each running 12 cores
- ✗ Results thanks OTC devoted resources for testing.
- ✗ iperf3 -P 32 showed average throughput range 21 Gbps

PROVIDERS THROUGHPUT



DEDICATED CIRCUIT MESH TESTS

- ✘ Three oneproviders
- ✘ Each running 12 cores
- ✘ Data collection is distributed in three places
- ✘ The migration job is making replica of all files cached in all three providers
- ✘ Each provider needs to deliver data to two others
- ✘ Each provider needs to download data from two others

