RESEARCH ARTICLE

Data management for distributed computational workflows: An iRODS-based setup and its performance

Mohamad Hayek[1], **Martin Golasowski**[2], **Stephan Hachinger**[1]*, **Rubén J. García-Hernández**[1,3¤], **Johannes Munke**[1], **Gabriel Lindner**[3], **Kateřina Slaninová**[2], **Philipp Tunka**[1], **Vít Vondrák**[2], **Dieter Kranzlmüller**[1,3], **Jan Martinovič**[2]

1 Leibniz Supercomputing Centre (LRZ), Bavarian Academy of Sciences and Humanities, Garching near Munich, Germany, **2** IT4Innovations National Supercomputing Center (IT4I), VŠB - Technical University of Ostrava, Ostrava, Czech Republic, **3** MNM-Team, Ludwig-Maximilians-Universität (LMU) München, Munich, Germany

¤ Current address: CGI Deutschland B.V. & Co. KG, Munich, Germany
* hachinger@lrz.de

Abstract

Modern data-management frameworks promise a flexible and efficient management of data and metadata across storage backends. However, such claims need to be put to a meaningful test in daily practice. We conjecture that such frameworks should be fit to construct a data backend for workflows which use geographically distributed high-performance and cloud computing systems. Cross-site data transfers within such a backend should largely saturate network bandwidth, in particular when parameters such as buffer sizes are optimized. To explore this further, we evaluate the "integrated Rule-Oriented Data System" iRODS with EUDAT's B2SAFE module as data backend for the "Distributed Data Infrastructure" within the LEXIS Platform for complex computing workflow orchestration and distributed data management. The focus of our study is on testing our conjectures—i.e., on construction and assessment of the data infrastructure and on measurements of data-transfer performance over the wide-area network between two selected supercomputing sites connected to LEXIS. We analyze limitations and identify optimization opportunities. Efficient utilization of the available network bandwidth is possible and depends on suitable client configuration and file size. Our work shows that systems such as iRODS nowadays fit the requirements for integration in federated computing infrastructures involving web-based authentication flows with OpenID Connect and rich on-line services. We are continuing to exploit these properties in the EXA4MIND project, where we aim at optimizing data-heavy workflows, integrating various systems for managing structured and unstructured data.

OPEN ACCESS

Citation: Hayek M, Golasowski M, Hachinger S, García-Hernández RJ, Munke J, Lindner G, et al. (2026) Data management for distributed computational workflows: An iRODS-based setup and its performance. PLoS One 21(1): e0340757. https://doi.org/10.1371/journal.pone.0340757

Editor: Muhammad E. S. Elrabaa, King Fahd University of Petroleum & Minerals, SAUDI ARABIA

Received: December 12, 2024

Accepted: December 26, 2025

Published: January 12, 2026

Data availability statement: All relevant data are within the manuscript and its Supporting information files.

Funding: This work has been co-funded by the EU's Horizon 2020 Research and Innovation Programme (2014-2020) under grant agreement N° 825532 (Project LEXIS – "Large-scale EXecution for Industry and Society"). The support for the work was in the form of a grant awarded to IT4Innovations, LRZ and partners, from which MH, MG, SH, RGH, JMu, KS, VV and JMa were co-funded in terms of salary. Furthermore, this work received support from the EXA4MIND project ("EXtreme Analytics for MINing Data spaces"), funded by the European Union's Horizon Europe Research and Innovation Programme, under Grant Agreement N° 101092944. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them. The support for the work was in the form of a grant awarded to IT4Innovations, LRZ and partners, from which MH, MG, KS and JMa were co-funded in terms of salary. This work has also received significant support from the project grant ICBxBCI for Czech-Bavarian collaboration and researcher mobility of the Bavarian State Chancellery (Bayerische Staatskanzlei). The support was in the form of a travel grant awarded to LRZ and used for this work by MH and SH. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254). From this program, MG, JM and VV were co-funded in terms of salary. The state of Bavaria, Ministry for Science and the Arts, has co-funded SH and funded PT and DK in terms of salary. The study with all its authors has been co-funded by LEXIS, by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254), and by the state of Bavaria, Ministry for Science and the Arts in terms of infrastructure support. The specific roles of the authors are articulated in the 'author contributions' section. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. There was no additional external funding received for this study.

Introduction

Distributed computing in science and engineering has been a great technical and organizational challenge since the appearance of large-scale grid-computing [1] infrastructures such as TeraGrid [2], WLCG [3], and EGI [4]. Many solutions have failed the test of large-scale adoption. However, the idea of using best-suited computing resources across geographical sites for workflows keeps playing a very important role [5]. In line with this, the development of frameworks and infrastructures for distributed data management continues. The EUDAT Collaborative Data Infrastructure has become a major player in data management for European research with its services such as B2SAFE and B2HANDLE [6–8]. The European Open Science Cloud EOSC [9] will then integrate the most promising data-management approaches. Furthermore, European and International Data Spaces [10] are expected to help shape the distributed-data landscape in the coming years.

In this setting, the IT4Innovations National Supercomputing Center (IT4I, Czech Republic) and partners across Europe have developed the LEXIS Platform for distributed computing [11–13]. LEXIS orchestrates workflows across systems at supercomputing centers, including the Leibniz Supercomputing Centre (LRZ, Germany). It leverages heterogeneous computing backends, such as Infrastructure-as-a-Service clouds and classical High-Performance-Computing (HPC) clusters, depending on the use case. The original development project involved three demanding pilot applications, focused on data-intensive simulations and big data analytics in the sectors of weather & climate [14], aeronautics [15] and earthquakes & tsunamis [16].

Computational workflows across different locations need an efficient mechanism for data management, facilitating uniform data access and automated data transfer. For this purpose, a "Distributed Data Infrastructure" (DDI) as a component of the LEXIS Platform was to be set up [17], suitable for generic data from multiple research disciplines. We conjectured *(I)* that an open-source data-transfer and/or distributed data management framework could deliver the needed functionality with limited need for adaptation; *(II)* that our non-functional requirements could thus be satisfied, and in particular that large parts of the available network bandwidth could be utilized for typical data transfers over the wide-area network; and *(III)* that, to this end, performance limitations could be sufficiently understood to mitigate or overcome them with straightforward approaches, for example optimization of buffer size settings or reasonable sanitation of usage patterns.

The construction of the DDI thus involved three challenges: choosing a data management framework, delivering a setup concept, and putting the conjectures *(I)-(III)* given above to a test – which is the focus of this work. As a data-management framework, the DDI leverages the "integrated Rule-Oriented Data System" iRODS [18] with the B2SAFE [7] module. Based on this, it fulfils our functional and non-functional requirements, including basic prerequisites for compliance with the FAIR (Findable, Accessible, Interoperable, Reusable) principles [19] of modern research data management. We present and analyze performance measurements within our large-scale iRODS federation and discuss our learnings from tests and operation. We thus hope

Competing Interests: Financial: This work has been co-funded by the EU's Horizon 2020 Research and Innovation Programme (2014–2020) under grant agreement N° 825532 (Project LEXIS – "Large-scale EXecution for Industry and Society"). Furthermore, this work received support from the EXA4MIND project ("EXtreme Analytics for MINing Data spaces"), funded by the European Union's Horizon Europe Research and Innovation Programme, under Grant Agreement N° 101092944. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them. This work has also received significant support from the project grant ICBxBCI for Czech-Bavarian collaboration and researcher mobility of the Bavarian State Chancellery (Bayerische Staatskanzlei). This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the eINFRA CZ (ID:90254). Non-Financial: IT4Innovations is part of the iRODS Consortium and of the EUDAT CDI Council. The decision towards these participations has been influenced by the use of these frameworks in LEXIS, not the other way around. The measurements presented have been conducted before the iRODS Consortium membership. All this does not alter our adherence to PLOS ONE policies on sharing data and materials.

to give some guidance for similar applications and to shed further light on the conditions under which iRODS performs well. As LEXIS technology will help to build and operate the EuroHPC Federation Platform [20], our findings are likely of relevance for future European distributed-data and distributed-computing applications.

Below, we first give an overview of related work. We then discuss and illustrate requirements on a DDI supporting workflows as in LEXIS, and lay out how iRODS with the B2SAFE module helps to fulfill these. We explain how we used this framework to build a reliable, federated distributed-data system for the first version of the LEXIS Platform. The article then extensively discusses our experience with the DDI with respect to the conjectures given above. It addresses operational aspects, performance benchmarks, limitations, and directions beyond the original LEXIS setup, as they reflect in our current projects.

Related work

Distributed data management solutions can roughly be categorized into *i)* low-level file-system and data-management frameworks for storage clusters, *ii)* middleware for unified access to heterogeneous storage systems, and *iii)* solutions loosely federating data systems at the metadata-catalog or data-transfer level. We give an overview and argue that solutions from category *(ii)* lend themselves as a backend for distributed computational workflows. Our exploration focuses on sustainable open-source solutions which were available at the time the first version of the LEXIS Platform was built. We also discuss other data federations of a scale similar to our DDI and work on benchmarks of iRODS systems as relevant to the later sections of our article.

File-system and management solutions for storage clusters

Ceph [21], EOS [22], GlusterFS [23], HDFS [24], XtreemFS [25] and similar systems manage data distributed over a number of server and disk systems in a data center. The user sees the system as block or object storage – which can be mounted or, alternatively, addressed via interfaces such as S3-compatible [26] endpoints. The solutions are usually strong in guaranteeing storage redundancy, failure resilience and load balancing. Block-wise data distribution or reduplication across physical systems are often used to enhance performance and data safety. With exception of XtreemFS, the systems mostly target a relatively tight integration of the storage elements rather than a federation over wide-area networks between regions and countries. Also, they do not focus on handling metadata or persistent identifiers to support FAIR [19] research data management.

Middleware frameworks for unified access to heterogeneous storage

Storage middleware frameworks for integration or federation of heterogeneous storage systems include dCache [27], iRODS [18], Onedata [28], Rucio [29] and XRootD [30]. Data on different backends is thus presented through a unified namespace, and cross-system or cross-site data retrieval is facilitated by efficient internal data-transfer mechanisms. With respect to file-system-level solutions, unavailability of sites or higher network latencies can be handled more gracefully. To this end, a typical federation consists of several (remote) storage sites and data is distributed on object, file or dataset level. The systems often support high-level metadata and policy management features, enabling storage tiering or sophisticated data-access control, for example. All these capabilities make middleware frameworks ideal for data federations across computing centers.

Federation solutions with mere focus on metadata catalogs or data transfer

On a still higher level, data storage can be indexed or federated via repository software or data catalogs. Systems implementing this strategy include Fairspace [31], COLID [32] or Gen3 [33]. Also, Data Spaces [10] – as standardized by the International Data Spaces Association [34] – rely on a federated architecture with metadata catalogs. Other data-management frameworks, such as DataLad [35] or GLOBUS [36] have strongly focused on facilitating the exchange of data or metadata.

In the systems discussed, data transfer is typically triggered by explicit pull or push requests. In contrast, we had envisaged transparent and efficient cross-system access on data through a tighter integration. Therefore, we focused on middleware frameworks.

Relevant work on large-scale data federations and benchmarking

Large-scale data federations based on middleware often have a background in high-energy physics [37,38]. Such federations typically make use of dCache, XRootD or Rucio [27,29,30]. These solutions show competitive performance in a variety of demanding contexts [39,40], however with data predominantly from one field of science. Thus, the resulting platforms may have a tendency towards higher performance and lower flexibility than our solution discussed here, but can be considered similar overall.

In other fields of research, the value of data federations beyond the facilitation of data transfer [36,41] has been recognized as well [42]. iRODS together with the data replication features of B2SAFE has been used in the CompBioMed project series to set up a data federation across European supercomputing centers [43]. Our DDI for LEXIS follows the same basic principle, but is not domain-specific, and has a focus on elevated availability according to the “High Availability iRODS System” concept [44]. Onedata has been chosen to set up data handling for the cross-site Indigo DataCloud [45], and is used within the EGI Federated Cloud [46]. Although these infrastructures mostly cover cloud-computing applications, the data-federation capabilities are comparable to what we present. Recently, the concept of Data Spaces and the Dataspace protocol [34] have been increasingly adopted [47] as a higher-level federation approach complementary to our work. The European Open Science Cloud [9] is expected to interoperate with European Data Spaces [10].

A number of authors have aimed at assessing the features [48–50] or the performance [40,51,52] of distributed data management frameworks. Due to the heterogeneity and rapid development of such systems, meaningful and precise comparisons are challenging. In our work, we lay out requirements on a data federation in our setting and assess the match with the five different middleware-based systems discussed above. We motivate and discuss our solution, evaluating its performance. Performance benchmarks on iRODS have previously been published by several authors [51,53,54], covering different aspects of the system. However, these works used earlier iRODS versions than we do, and did not focus on testing data transfer performance in a large-scale international federation.

Basic requirements on a DDI for computational workflows across computing centers

Platforms such as LEXIS [12,13] aim at simplifying the efficient usage of heterogeneous computing systems for science and enterprise applications. They help to automate typical computational workflows, which are sequences or directed acyclic graphs of computing tasks; a simple example would be the sequence: preprocessing – weather simulation – post-processing – output data analytics – visualization. Through orchestration across computing centers, an optimal computing resource can be used for each task. In the LEXIS case, the user supplies a workflow description, necessary applications and input data via the LEXIS Portal [12]. After workflow launch, an orchestration system addresses the infrastructure involved. Input, intermediate and output data are stored on the DDI to ensure data access from all sites and to keep results accessible in the mid to long term. Furthermore, the orchestrator triggers data staging between the DDI and computing systems where computing systems have no direct or efficient access to the DDI (e.g., because of firewalling).

This setting is illustrated in Fig 1. The upper part of the figure shows an urgent-computing [55] workflow for simulating stent placement inspired by an earlier demonstration [56]. The example uses input/output data replication in order to attempt to run the simulation on three HPC systems for resilience and to collect the results. The lower part shows a workflow for weather and wildfire-risk simulation, roughly following an example implemented earlier as well [13]. It uses one HPC/IaaS-Cloud site for actual computing, while another site/system is used to regularly download input data.

The setting determines the functional/non-functional requirements (FR/NR) on the DDI as further described below and indicated in the figure. After the requirements collection, we lay out how the requirements motivate our choice of iRODS with B2SAFE module for our first test versions of the DDI (and – owing to the success – for its continued operation). The discussion follows our earlier work [17], but adds performance requirements and an assessment of all middleware systems discussed above.

Functional requirements

FR1: Unified cross-location access to data. To avoid complications for the orchestration system and for users, the DDI shall implement a unified view on all data in the platform, wherever they may be stored. At the same time, the orchestration system must easily retain awareness of the hosting site for a given dataset to allow for optimized workflow execution.

FR2: Implementation of data replication and further storage policies. A distributed computing platform lends itself [14,16,56] to urgent-computing use cases (see also Fig 1). To avoid unnecessary delays in such a scenario, certain input datasets have to be available on-site at multiple computing centers in the federation.

A DDI accommodating such workloads thus has to automate data replication between the participating computing/data centers, keeping record of the copies stored. This also helps to fulfill expectations on data integrity and availability of academic and commercial users. Furthermore, the DDI should support the implementation of a rich set of storage policies, e.g. for management of tiered storage.

FR3: Support for FAIR research data management. Basic support for the FAIR principles [19] is mandatory in modern research data management, and consequently funding agencies can require implementation of FAIR in research projects using the DDI. Technical aspects of FAIR prominently include the assignment of metadata and persistent identifiers to datasets [19], which the DDI has to support.

FR4: Application programming interfaces (APIs) and bindings. The DDI is to be used via well-defined REST [57] APIs, for example for data up-/download, and staging and replication requests. To implement such APIs [17], the data management system must offer sufficiently rich programming interfaces.

FR5: Authentication and authorization infrastructure support. The DDI system has to interoperate with a common identity and access management (based on OpenID Connect tokens [58] and Keycloak [59] in LEXIS), which helps to implement state-of-the-art security concepts [13], based, e.g., on the zero-trust paradigm.

Non-functional requirements

NR1: Reliability. The DDI shall be built with redundancy of critical components to offer reliable data management. This helps to create trust in the platform and to avoid a waste of computing time through failed workflows.

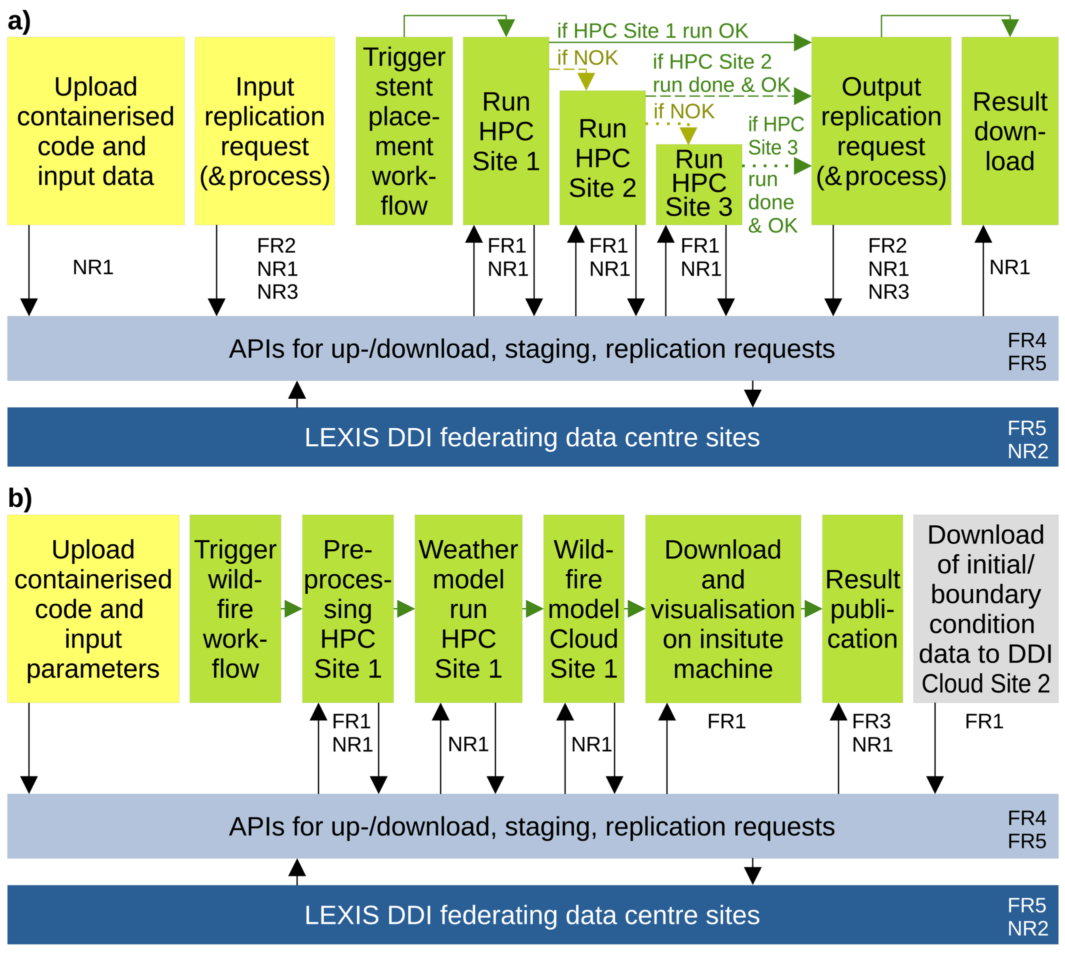
a)
Upload containerised code and input data
Input replication request (&process)
Trigger stent place-ment work-flow
Run HPC Site 1
if HPC Site 1 run OK
if NOK
Run HPC Site 2
if HPC Site 2 run done & OK
if NOK
Run HPC Site 3
if HPC Site 3 run done & OK
Output replication request (&process)
Result down-load
NR1
FR2 NR1 NR3
FR1 NR1
FR1 NR1
FR1 NR1
FR2 NR1 NR3
NR1
APIs for up-/download, staging, replication requests
FR4 FR5
LEXIS DDI federating data centre sites
FR5 NR2
b)
Upload containerised code and input parameters
Trigger wild-fire work-flow
Pre-proces-sing HPC Site 1
Weather model run HPC Site 1
Wild-fire model Cloud Site 1
Download and visualisation on insitute machine
Result publi-cation
Download of initial/ boundary condition data to DDI Cloud Site 2
FR1 NR1
NR1
NR1
FR1
FR3 NR1
FR1
APIs for up-/download, staging, replication requests
FR4 FR5
LEXIS DDI federating data centre sites
FR5 NR2

Fig 1. **Two example workflows (simplified, read from left to right) that a cross-center DDI as in LEXIS should accommodate. a) stent-placement simulation [56]; b) regular weather and wildfire-risk predictions [13].** Yellow and grey boxes indicate interactive and regularly-executed actions, respectively, while the actual workflow steps are marked green. In case a), runs are attempted on three HPC sites to warrant success in an urgent-computing [55] setting. Functional and non-functional requirements (FR/NR, see text) are related to example functionalities of the workflows. NR2, FR4 and FR5 (see blue boxes) make sure that the DDI offers uniform APIs and authentication across all relevant backends. NR1 and NR3 warrant reliability and performance, most importantly in the urgent-computing case a). FR1 warrants cross-center functionality in both workflows. FR2 facilitates automated cross-center data replication (case a)) and FR3 warrants FAIR Research Data Management (case b)).

https://doi.org/10.1371/journal.pone.0340757.g001

NR2: Flexibility in backend system usage. The DDI has to integrate heterogeneous storage backends typically used in production at various computing centers. Integration with both commercial and open solutions should be possible, including, e.g., IBM Spectrum Scale (formerly GPFS [60]) or Ceph [21].

NR3: Performance requirements. Data staging and replication actions in typical applications such as the original LEXIS Project pilots (see also FR2) typically involve datasets with sizes from 10 GB to 10 TB. The time spent for data transfers in workflows should, as a rule of thumb, remain below the time spent for computing. In urgent-computing use cases [14,16], data-transfer times must usually remain in the order of 100 seconds. In use cases involving long simulations on HPC systems [14,15], a data transfer time of one day may be acceptable. In particular, these limits have to be respected for cross-site workflows, while on-site data transfers will typically be faster.

Choice of system

For the LEXIS DDI as a concrete implementation, we have chosen iRODS with the B2SAFE module. We motivate this choice by assessing the suitability of different middleware frameworks, and by discussing how exactly the solution fulfils our requirements.

Suitability of distributed-data middleware. Table 1 illustrates how well the middleware solutions discussed above match our requirements.

While iRODS closely matches our requirements, it remains clear that other systems, e.g. OneData, provide similar functionality. Strong points of iRODS are certainly its long and sustained history of development and applications [64], and its architecture federating storage sites without any federation-management server or strong functional interdependencies [18]. Furthermore, in the setting of LEXIS, the vision of embedding the DDI in European data ecosystems has been important. The positioning of B2SAFE with iRODS as a building block of a general-purpose European data infrastructure matches here. We note, however, that other frameworks see a usage within European data-infrastructure projects as well [46,65,66].

How iRODS with B2SAFE module fulfills our requirements. iRODS fulfils FR1 by addressing distributed storage systems through a unified structure of directories and files – dubbed “collections” and “data objects” in iRODS terminology [18]. The top level collections refer to “iRODS Zones” – units in the data federation that can function independently [18], typically corresponding to hosting locations. The framework can react to file-system events (e.g. file creation) by executing freely definable “iRODS Rules” [18], realized via scripts which help to implement policies or functionalities. The B2SAFE [7] module implements iRODS Rules for automatic data replication (cf. FR2) and for the acquisition of persistent identifiers for data through the B2HANDLE [8] service. FR3 is thus met, as iRODS can store user-defined metadata for files and collections. REST APIs and various clients are offered, fulfilling FR4. Via the iRODS-OpenID plugin [67] and some customization work (see section “Access management” below), requirement FR5 has been fulfilled.

Elevated availability (cf. NR1) can be reached with iRODS through well-documented configurations [44]. In addition, we rely on multiple iRODS Zones. The resulting system allows for data mirroring and minimizes the probabilities of data loss or unavailability. iRODS can address a wealth of storage backends, and thus complies with NR2. With our benchmarks, we will test whether sufficient performance is offered to fulfill NR3.

Table 1. Requirements on a DDI as in LEXIS vs. characteristics of middleware solutions. “Y” indicates that the solution fulfils the requirement, “N” that it does not. “C” indicates conditional or partial fulfilment.

Requirement	dCache	Onedata	iRODS (+B2SAFE)	Rucio	XRootD
FR1 (unified access)	Y	Y	Y	Y	Y
FR2 (replication)	Y	Y	Y	Y	Y
FR3 (FAIR)	C[1,2]	Y	Y	C[2]	N[3]
FR4 (APIs/bindings)	Y	Y	Y	Y	Y
FR5 (auth. token support)	Y	Y	Y	Y	C[4]
NR1 (reliability)	Y	Y	Y	Y	Y
NR2 (flexible backend use)	Y	Y	Y	C[5]	Y
NR3 (performance)	needs benchmarking[6]				

[1]Handling of user-defined metadata was added in 2020 (DCache 6.2 – extended attributes) when the design decision for using iRODS in LEXIS had already been made.
[2]We have, to the best of our knowledge, no indication that these systems facilitate the acquisition of external persistent identifiers such as DOIs or B2HANDLEs [8,61].
[3]XRootD concentrates on pure file access and handling.
[4]OpenID Connect support was added in or after 2020 [62].
[5]Rucio typically accesses storage through higher-level protocols [63], using XRootD as backend, for example.
[6]Performance must be verified by measurements as in our work; however for all these systems, we had no evidence for critical limitations from the literature.

https://doi.org/10.1371/journal.pone.0340757.t001

A DDI concept for LEXIS

We built the LEXIS DDI as an iRODS federation with one iRODS Zone per LEXIS data center. The requirement NR1 motivated us to set up particularly reliable infrastructure within each iRODS Zone. Below, we summarize our concept [17] to facilitate a basic understanding of the system, as relevant for its evaluation.

High-availability setup for one iRODS Zone

Each of our iRODS Zones has been set up largely following earlier works on the "High Availability iRODS System" concept [44,68,69]. The most critical component of an iRODS Zone is its "provider server" or "iCAT-enabled iRODS server" and the iCAT database connected to it. Via the provider server, virtually all information about an iRODS Zone, including file-system and user metadata, is stored in the iCAT database, for which we use PostgreSQL [70]. We have installed two provider servers and two database servers per iRODS Zone to maximize resilience. The resulting blueprint installation is illustrated in Fig 2.

Our design for the redundant database backend for the iCAT (Fig 2, upper part) uses repmgr and Pgpool-II – see [71] and references therein. repmgr handles live replication between the PostgreSQL instances. Pgpool-II, as ingress on a separate machine, handles request forwarding and manages failovers. Pgpool-II furthermore acts as a load-balancer, using the secondary backend for parallel reading. If the primary backend becomes unavailable, Pgpool-II detects this and performs failover, triggering repmgr to register the secondary database as a new primary. An outage of the new primary in this state would make the database backend unavailable as a whole, as Pgpool-II would have no failover options left. It is thus crucial to bring a failed primary up again in due time, re-synchronizing it using repmgr and re-registering it as a

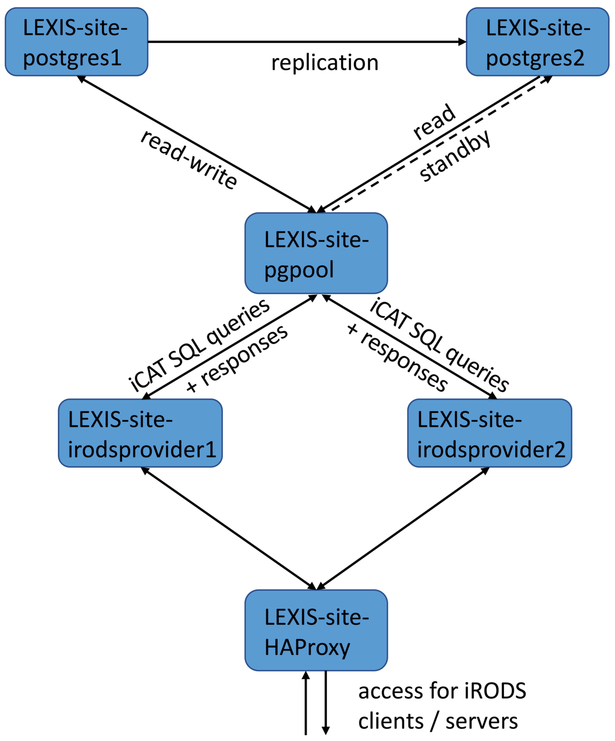
LEXIS-site-postgres1
replication
LEXIS-site-postgres2
read-write
read
standby
LEXIS-site-pgpool
iCAT SQL queries + responses
iCAT SQL queries + responses
LEXIS-site-irodsprovider1
LEXIS-site-irodsprovider2
LEXIS-site-HAProxy
access for iRODS clients / servers

Fig 2. Blueprint iRODS Zone setup. Both iRODS provider servers and database servers as potential points of failure are redundantly provisioned, following [44].

https://doi.org/10.1371/journal.pone.0340757.g002

new secondary with Pgpool-II. Within LEXIS, this has been done manually on monitoring alerts. Thus, the original system status has been restored, albeit with database roles switched.

Our redundant iRODS provider servers (Fig 2, lower part) address the Pgpool-II server and are exposed themselves via an HAProxy [72] server as load balancer and ingress. Requests from one IP address to HAProxy have been configured to be redirected to the same default backend to avoid problems with the authentication flow (cf. section "Access management" below).

Pgpool-II and HAProxy are well-proven software products; however they have to be deployed on infrastructure with sufficient availability in order to not constitute a probable point of failure. Managed-server infrastructures within computing centers offer virtual machines with automatic reinstantiation or redundant deployment for such purposes.

Federation between data centers

The iRODS Zone federation [18] mechanism has been used to implement cross-site data access in LEXIS, while retaining local data-system scopes. Remote iRODS Zone data appears in the collection structure simply under a separate top-level collection, for example `/center1Zone` or `/center2Zone`.

Data-staging actions to and from the DDI are handled via a staging REST API, with a backend built on top of iRODS's interfaces [17], whereas transfer between iRODS Zones can be transparently handled by iRODS on file access through the federation.

Access management

The LEXIS DDI has been integrated with the LEXIS authentication and authorization infrastructure based on Keycloak [59], an identity and access management solution.

iRODS provides its own authentication mechanism based on username and password, but encourages the use of its plugin architecture [18] for leveraging other mechanisms. The iRODS OpenID authentication plugin [67], in particular, enables the use of OpenID Connect [58] within iRODS. In this approach, the authentication token has been passed in the username field to iRODS, where the field has 1024 bytes in iRODS version 4.2.8. Keycloak, however, uses JSON Web Tokens [73] with a larger size.

We use a combination of pre-authentication and hashing [74] to overcome the size limit. In the architecture of the authentication plugin, the token is sent to a broker service, which validates tokens and stores them in an authentication database. In our solution, the user first pre-authenticates directly with the broker, so that the token is validated and stored. We then modified the Python iRODS Client [75] to hash the token if it is larger than the permitted length. The broker, on the other hand, has been changed to pre-calculate and store hashes of validated tokens and to test received authentication data against both validated tokens and hashes. We have also wrapped the iRODS command line interface (iCommands) to ensure the token remains hashed in the `.irodsA` file. In the three-way handshake between the client, iRODS and the broker, the iRODS provider server addressed within our redundant system must remain the same. This can be guaranteed in our setup by configuring HAProxy accordingly. In order to have a consistent approach with respect to availability, the broker and its database were set up redundantly as well. We have used setups similar to those described above – in particular a Pgpool-II and repmgr-based setup for the PostgreSQL database of the broker.

LEXIS/Keycloak user identities are mirrored in the iRODS system using the `iadmin aua` (add user authentication-name) command, and roles are reflected in appropriate group memberships [13,17] using a periodically executed synchronization script.

Usage of B2SAFE and B2HANDLE

The B2SAFE module [7] is an add-on for iRODS, providing enhanced replication and replica-tracking capabilities. The add-on implements iRODS Rules to put data policies into effect, in particular rules that can be executed for replication of datasets over different backends in an iRODS federation.

When run on a collection, the relevant B2SAFE rule will create a replica on the target destination. It will register persistent identifiers for both the source and the replica via B2HANDLE [8], and reflect the relation of the replicas in their iRODS (user-definable) metadata. iRODS can execute rules as a response to events or upon external trigger. In LEXIS, B2SAFE replication is triggered upon request on the LEXIS Portal [12]. For our data-transfer benchmarks, it is run via the command line. The B2SAFE rule execution process and functionality is visualized in Fig 3.

PID assignment via B2HANDLE is used in the LEXIS DDI for datasets wherever needed, in particular in the context of FAIR data handling. B2HANDLE can conveniently be addressed via the `EUDATCreatePID` function included in B2SAFE, also when not using the replication functionality.

Performance, limitations and directions for development

Over three years of operation of the DDI enabled us to evaluate the system in depth. Below, we first discuss ease of operation and reliability. We then present benchmark measurements of data-transfer rates and analyze their dependency on dataset size as well as transfer buffer size with the Python iRODS Client [75]. Table 2 shows an overview of these performance tests. Our evaluation refers to the period up to mid 2022, where the original configuration was maintained within the first version of the LEXIS Platform. Besides encouraging results, we also discuss limitations of our concept.

After 2022, the system underwent some re-design and is currently complemented by database management systems in order to run extremely data-heavy workflows within the EXA4MIND Horizon Europe project ("Extreme Analytics for MINing Data spaces"). We give a glimpse on these developments in the concluding part of this section.

Experience with the iRODS blueprint setup in operation

In operation, the setup as described delivered the expected functionality, confirming our conjecture *(I)* given in the introduction.

Indeed, the concept prevented unexpected outages of the LEXIS DDI. We are not aware of any user complaints nor of operators noticing major outages while running the DDI as described. However, the relatively complicated concept with six servers for the core functionality of each iRODS Zone has its price.

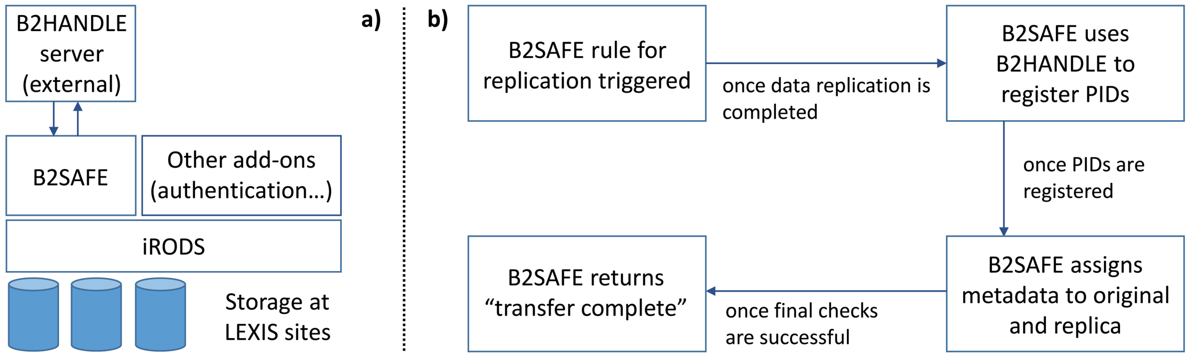
B2HANDLE server (external)
B2SAFE
Other add-ons (authentication…)
iRODS
Storage at LEXIS sites
a)
b)
B2SAFE rule for replication triggered
once data replication is completed
B2SAFE uses B2HANDLE to register PIDs
once PIDs are registered
B2SAFE assigns metadata to original and replica
once final checks are successful
B2SAFE returns "transfer complete"

Fig 3. The role of B2SAFE and B2HANDLE for data replication in the LEXIS DDI. The left part (a) of the figure sketches the system, where B2HANDLE is run on a server separate from the DDI ("external"). Part (b) illustrates the replication and replica-registration process.

https://doi.org/10.1371/journal.pone.0340757.g003

Table 2. **Overview of performance tests, grouped by figures where the results are shown (second column).** Details on the tests and further explanations can be found in the respective sections in the text.

Test	Fig	File Sizes	Number of Streams	Client Buf. Size
LRZ → IT4I: `icp`[1]	4	5/100/1 000 MB	auto[2] (1 at 5 MB)	no-client transfer[3]
IT4I → LRZ: `icp`	4	5/100/1 000 MB	auto[2] (1 at 5 MB)	no-client transfer[3]
LRZ → IT4I: `icp`[1]	5	5/100/1 000 MB	auto[2] (1 at 5 MB)	no-client transfer[3]
LRZ → IT4I: B2SAFE	5	5/100/1 000 MB	auto[2] (1 at 5 MB)	no-client transfer[3]
LRZ → IT4I: Python[4]	5	5/100/1 000 MB	1	2 MiB
LRZ (Test System) → IT4I (Test System): Python, Buffer 1 MB	6	10/100/1 000 MB	1	1 MB
LRZ (Test System) → IT4I (Test System): Python, Buffer 10 MB	6	10/100/1 000 MB	1	10 MB
LRZ (Test System) → IT4I (Test System): Python, Buffer 100 MB	6	10/100/1 000 MB	1	100 MB

[1]Identical test.
[2]iRODS server and client default to one stream below file sizes of 32 MB and to auto-negotiation above, where the auto-negotiation is likely to yield three streams [76].
[3]`icp` and B2SAFE "client" applications trigger transfers between resources in the respective iRODS zones; however, in the transfers themselves only the servers are involved.
[4]The Python iRODS Client has been used here to push data ("put" function, see respective section) from LRZ to the IT4I iRODS server.

https://doi.org/10.1371/journal.pone.0340757.t002

The redundant PostgreSQL backend showed outstanding stability, but generated substantial system-administration overhead when machines were rebooted for urgent kernel security upgrades. In a typical managed-server infrastructure, for instance at LRZ, such reboots are automated and may happen a few times per year. Once the primary PostgreSQL server reboots, Pgpool-II detects its outage and triggers a failover with de-registration of the instance. As a result, the administrator will find the system with only one PostgreSQL server active, and has to re-synchronize and re-register the other instance, as described in the discussion of our setup. This analogously applies to the authentication-broker system as discussed above.

The duplicate iRODS provider servers in front of the database backend have been set up replicating the internal host-name [69]. To our knowledge, this is necessary as iRODS provider servers are generally aware about their own hostname via the iCAT database, which is shared here. The solution generally worked in our case, but is certainly not best practice from a network and system configuration point of view.

Benchmarking in real-life configuration

We measured file-transfer performance between the LEXIS sites IT4I and LRZ. Thus, we tested whether iRODS could use the network bandwidth well enough to fulfill our performance requirements (NR3).

We describe the test setup and discuss selected benchmark results below. Further data are included in the Supporting Information. The iRODS version in all tests was 4.2.8; tests involving B2SAFE or the Python iRODS Client were based on the versions 4.3.0 and 0.8.6 of these products, respectively.

Test setup. Files of 5 MB, 100 MB, and 1 000 MB were created on production machines within the LEXIS DDI at IT4I and LRZ. The files were stored on sufficiently fast local storage (e.g. solid-state or RAM drive) for evaluating actual iRODS transfer performance without any influence from file system speeds. What is therefore actually measured is the speed of the wide-area network minus *(i)* performance losses caused by firewall/gateway infrastructure or concurrent network usage, *(ii)* overheads related to the client program and to calls to the iRODS system, *(iii)* overheads due to metadata generation in iRODS data object creation, and *(iv)* overheads of the iRODS protocol itself.

First measurements were taken using `icp` to copy files. Afterwards, we compared performance measurements using `icp`, B2SAFE and the Python iRODS Client [75]. Each test was repeated 20 times, and average results are reported and discussed below (the raw data, also for individual repetitions, is delivered with the Supporting Information). Within the series of 20 repetitions, no consistent trends of speedup or slowdown were noted. This agrees with our assumption that read/write processes on our file systems only consume minor amounts of time, which also excludes influences from caching.

Results and discussion. Fig 4 shows data-transfer rates for files of 5, 100 and 1 000 MB of size from LRZ to IT4I and vice versa. The copy process has been initiated at LRZ side by issuing the command `icp` and addressing source and target files in the different iRODS Zones. We expect that issuing the command on the IT4I side would not give a qualitatively different picture. We measure a slight direction-related asymmetry in the transfer rates of 10% - 20%. It is statistically within the sample standard deviation for each file size, but consistent across file sizes. This may be due to various reasons, including the implementation of the actual copy process in iRODS.

For files of ~1 GB of size, decent transfer rates of >100 MB/s are obtained. Within the LEXIS context, these allow for the transfer of typical intermediate-result datasets (10 GB - 1 TB) from one system to the next within a workflow. In particular, smaller datasets can be transferred within ~100 s for urgent-computing cases, and requirement NR3 is fulfilled. Huge input datasets for computations, such as gridded input data for simulation ensembles, are often re-used within several workflows. Such datasets can thus be transferred over a longer, but acceptable period of time, with an effective rate of roughly 10 TB a day.

An important methodical take-home message from the measurements in Fig 4 is the significant speed gain with file size. To explain this speed gain, we take into account typical durations of the file transfers executed: 1 s for 5 MB; 2 s for 100 MB; 10 s for 1 GB. The pure transfer time for 5 MB should be a minor fraction of a second only. This suggests that in the LEXIS DDI ~1 s of time is spent on connecting, writing metadata into the catalog of the target iRODS Zone, and initiating the transfer. Above file sizes of 32 MB, `icp` is expected to show performance improvements due to the usage of parallel streams [77], so that the times measured for 100 MB and 1 GB reflect maximum reachable throughputs plus initial latencies. Assuming ~100 MB/s to be the throughput (found for the 1 GB test), a duration of 2 s for 100 MB indicates that latencies as found for the 5 MB tests (~1 s) apply to all transfers. This also implies that the transfer rates measured in the 1 GB tests have saturated, i.e. reflect maximum reachable rates, to within ~10%.

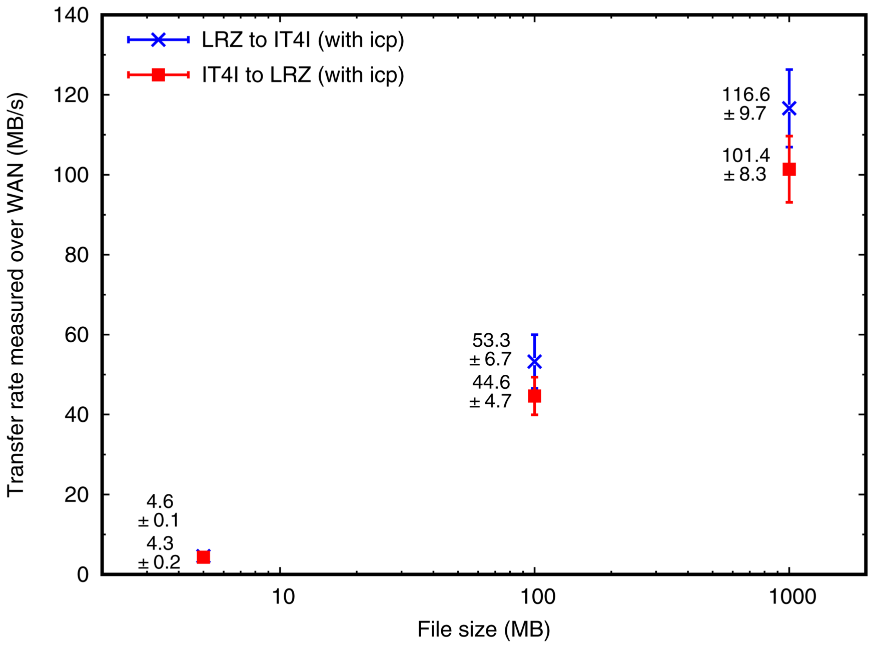

Fig 4. Data transfer rates in the LEXIS iRODS federation (between LRZ and IT4I) with `icp`. The figure shows average transfer rates and sample standard deviations (from 20 transfer tests per data point shown) in MB/s, using `icp` on files of various sizes.

https://doi.org/10.1371/journal.pone.0340757.g004

Our findings are consistent with those of Hünich & Müller-Pfefferkorn [53], who stated that the `iput` time for a negligible-size file into an iRODS system can indeed vary between hundredths of a second and a few seconds. Configuration peculiarities in our setup, as our redundancy provisions, firewalls and the cross-country federation, may introduce additional latency with respect to their setup. Bandwidth usage efficiency in iRODS transfers has strongly improved since the measurements of Hünich & Müller-Pfefferkorn in 2010 [53] and Depardon, Le Mahec & Ségui in 2013 [51]. In particular, Russell, Coposky & Keller [54] found that almost full network speed is reached for negligible network latencies, and 50%-80% are retained for network latencies of 50 ms as we typically measure between LRZ and IT4I LEXIS infrastructure. Bandwidth tests between the LEXIS sub-networks at the centers, using `iperf3`, yielded results of 1.1-1.4 Gbit/s (i.e. 0.14-0.18 GB/s) at the time of our iRODS experiments [78]. This is consistent with the bandwidth through the virtualized project firewall at LRZ as a probable limitation. The >100 MB/s we find for 1 GB files correspond to a ~70% utilization of the network bandwidth, consistent with what one would predict from [54]. The results confirm our conjecture *(II)* given in the introduction, with data rates almost as high as the GridFTP and UFTP mass-data transfer protocols yield them [79,80] over wide- and local-area networks (70%-90%).

Fig 5 compares the speed of transfers initiated with B2SAFE and with the `put` function of the Python iRODS Client to `icp` speed in the direction from LRZ to IT4I.

The B2SAFE replication mechanism is based on calling the iRODS-internal `msiDataObjRsync` and `msiCollRsync` functions. This presumably yields a similar performance as the `irsync` command. Relative to the `icp` transfers, the B2SAFE transfers/replications show a speed of 21%, 30% and 51% with rising file size (5, 100, 1 000 MB). This points towards an additional latency per transfer, probably due to the metadata and persistent-identifier handling of B2SAFE, triggered on each replication. Checks by `irsync` on target data as part of the synchronization functionality may also contribute to the delay. We have performed additional tests with 10 GB files from LRZ to IT4I (see Supporting Information) and measured an average rate of 130 ± 7 MB/s. This confirms both our assumptions of an additional per-transfer latency

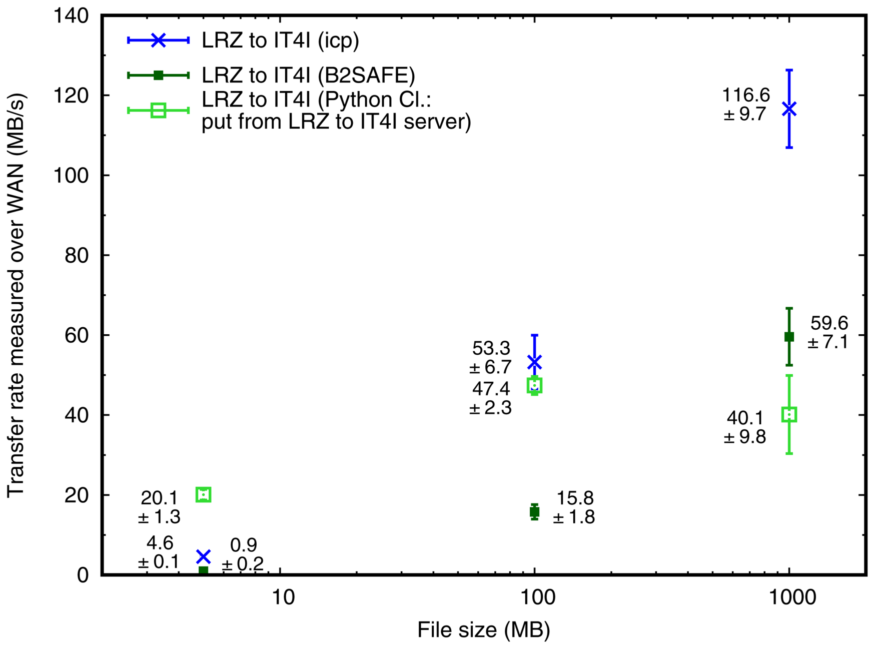

Fig 5. Data transfer rates in the LEXIS iRODS federation (from LRZ to IT4I) with various methods. The figure shows average transfer rates and sample standard deviations (from 20 transfer tests per data point shown) in MB/s, using `icp`, B2SAFE or the Python iRODS Client on files of various sizes.

https://doi.org/10.1371/journal.pone.0340757.g005

in B2SAFE compared to `icp` (which becomes irrelevant when transferring a 10 GB file) and of speed saturation with `icp` at a file size of 1 GB (as mentioned above).

The Python iRODS Client shows a completely different behavior. It implements its own `put` function, which introduces a write buffer that can be manually adjusted. We used a buffer size of 2 MiB in our tests here as a first guess (similar to upstream development). The client, run on a LRZ server machine, then writes out data to the remote (IT4I) iRODS server and target collection in write-buffer chunks as specified. In the version tested, only one parallel transfer stream to the server is used, which certainly limits the speed somewhat with respect to the `icp` and B2SAFE methods. The results (see Fig 5) indicate a lower overhead at 5 MB file size, but inferior performance for large files. In particular, going from 100 MB to 1 000 MB in file size, performance saturates (or even decreases) instead of an expected increase. Obviously, the network is not fully utilized. This behavior is further investigated below.

Dependency of performance on buffer size in the Python iRODS Client

Data transfer within an iRODS federation is subject to buffering in all client and server components involved, suggesting potential for speed improvements by optimization or matching of buffer size. Parts of this topic have been discussed by the iRODS team in an extensive whitepaper [54], possibly as a response to earlier claims of a relatively low iRODS performance over networks [51]. Native file transfers between iRODS Zones have – according to Russell, Coposky & Keller [54] – long been optimized in terms of buffer sizes and of negotiation of the number of transfer streams.

In this context, it is illustrative to measure the influence of the internal buffer/chunk size on the performance of the `put` function of the Python iRODS Client. This aspect has been of practical importance for us, as LEXIS users and the LEXIS Platform are often employing the Python iRODS Client for data access. Below, we again first describe our tests – which have been executed in the scope of a Bachelor's thesis [52] – and then discuss the results, following the findings in that thesis.

Test setup. A test iRODS system without redundancy was set up between IT4I and LRZ, comprising a few virtual machines in the OpenStack [81] infrastructures at both sites. Performance was then measured as in the benchmarks above, albeit no single-file tests with 5 MB files were made (10 MB were used instead). Tests were continued up to file sizes of 10 GB. A modified version of the `put` routine of the Python iRODS Client version 0.8.6 was used, where the write buffer size was adjusted to be 1 MB, 10 MB or 100 MB. Transfer speeds measured on virtual machines within an OpenStack infrastructure clearly depend on the infrastructure load. Yet, from runs with different buffer sizes one after another, and consistency checks at different times of the day, a stable overall picture emerged.

Results and discussion. The results are shown in Fig 6. The rates here are lower than above, owing to the test infrastructure used. It is evident that only a large enough buffer guarantees optimum transfer speeds. This finding led us to suggest a Python iRODS Client buffer size of at least 100 MB for LEXIS. With this modification, we have roughly doubled transfer speeds for most applications with respect to the original MB-range buffer sizes, confirming conjecture *(III)* from the beginning of this work.

We observe (Fig 6, rightmost data points) that transfer rates with the largest-size files (10 GB) are equal or lower in comparison to those with 1 GB files even when providing a buffer of 100 MB. We may investigate this problematic behavior of the Python iRODS Client, which is somewhat analogous to the observations in the production DDI, further in the future. Possibly, the buffering strategy in the Python code interferes with iRODS-internal or infrastructure-related buffers.

Limitations of our approach

While our results have been encouraging, a DDI constructed to follow our requirements certainly has limitations inherent to the concept. These limitations are mostly related to the usage of data-management middleware.

Flexibility in federation and data-policy implementation and the extended metadata handling capabilities usually imply performance penalties. These are particularly evident when many small files are transferred (cf. previous sections), as

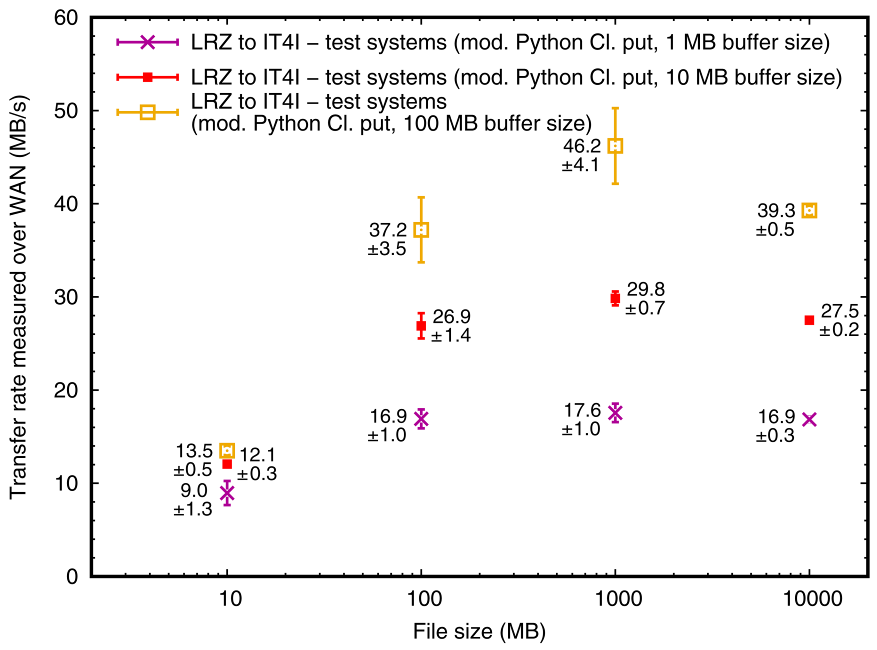

Fig 6. Data transfer rates from LRZ to IT4I in a test iRODS federation, using the Python iRODS Client `put` routine and various buffer/chunk sizes. The figure shows average transfer rates for various file sizes and sample standard deviations (from 20 transfer tests per data point shown) in MB/s, depending on the buffer size setting in the modified `put` routine. For this experiment, we have used virtual machines in OpenStack infrastructure on both sides.

https://doi.org/10.1371/journal.pone.0340757.g006

a large part of the middleware-based overheads expresses as waiting times per file-handling action. Furthermore, direct usage of a middleware-based DDI from HPC clusters faces limits by firewalling; for example, no outgoing connection to iRODS ports is allowed on the largest HPC clusters of LRZ. Performance is normally lower than with direct file-system access. This necessitates staging actions to HPC file systems before code runs, where methods have to be optimized and automated within computational workflows.

Directions for development of the LEXIS DDI

With our experience and benchmarks at hand, and requirements from new use cases, we devise directions for development. The scopes of this are *(i)* the continued maintenance and improvement of the LEXIS Platform beyond the initial project, and *(ii)* projects to extend or complement the DDI functionality.

Platform improvements: Ease of DDI maintenance and transfer rate optimization. As already stated above, the redundant PostgreSQL system generates maintenance work after failovers on a relatively regular basis, and the redundant provider-server setup is not optimum from an engineering perspective. We have thus moved the provider on a single virtual server with sufficient availability, and aim at a redundant PostgreSQL setup with less need for manual intervention. An automated recovery may be achieved via enhanced configuration or using alternative products, such as Patroni on Kubernetes [82,83].

Furthermore, the iRODS consortium has released a new interface for iRODS exposing a well defined HTTP-based API [84] with native support of OpenID Connect. As part of our efforts in the EXA4MIND project, we have updated the LEXIS DDI to use this iRODS HTTP API instead of the native iRODS protocol. We will thus be able to avoid usage of the iRODS OpenID authentication plugin and related middleware, further simplifying maintenance.

To enhance data-transfer rates, a simple, but important measure is to avoid usage patterns involving many small files. This had already been addressed in the LEXIS project, where data compression via a REST API trigger was introduced [17]. We will continue to analyze the structure of our APIs and of iRODS/B2SAFE in order to optimize the usage of the iRODS system from the client side. At the same time, we are pinpointing network-infrastructure limitations (e.g. overloaded switches or firewalls). A bottleneck affecting speeds and maximum dataset sizes in transfers between the DDI and computing systems has been related to temporary storage: Shared data-staging areas have often been used within LEXIS workflows to overcome connectivity restrictions. Enabling direct transfers here, together with the other measures, should consistently enable cross-system and cross-site transfers of the order of 100 TB within a day. We have been working on this to optimally support application cases of the EXA4MIND project, which use the LEXIS Platform with huge datasets.

Functionality enhancements. We plan to enhance and complement the existing functionalities of the LEXIS DDI in a series of subsequent projects. In particular, the EXA4MIND project aims at an integrated management of structured and unstructured data within a so-called “Extreme Data Database”. Instances of the Extreme Data Database comprise database management systems for structured data, and iRODS- and S3-based object stores. These can be flexibly deployed depending on the application, as EXA4MIND automates installation of data stores as well as data staging between these systems. Therefore, we also have the opportunity to index files on an iRODS or S3 system using a database. Fast search or data subsetting mechanisms can thus be implemented, supported by extended metadata extracted from the datasets. Such functionality is developed within the EXA4MIND “Advanced Query and Indexing System”.

EXA4MIND will furthermore enhance connectivity from LEXIS to European data ecosystems. In particular, we target European Data Spaces [10]. Connectivity to Data Spaces involves the installation of a connector [85] supporting the Dataspace protocol [34]. This may require extensions or mappings of the LEXIS DDI standard metadata set, currently adhering to the DataCite standard [86]. All this will help us to extend the FAIR data capabilities of the DDI as well as its interoperability with enterprise data ecosystems.

Conclusions and outlook

In this work, we have investigated the setup of a distributed data management backend for computational workflows across geographically-distributed computing centers. We conjectured that we could fulfill typical requirements for such a setting with an open-source product and limited customization, that network speeds could largely be utilized, and that we could understand the system well enough for basic performance optimization. This was confirmed by our experience setting up and evaluating the Distributed Data Infrastructure for LEXIS [12], a versatile cloud-HPC-big data workflow platform federating supercomputing centers.

Our data backend for LEXIS is based on iRODS and on B2SAFE [6,7,18]. The choice of the system has been motivated by a requirements analysis and a focus on an optimum immersion into the European data landscape. Our setup follows earlier work on the “High Availability iRODS System” concept [44,68,69]. It warrants elevated availability through redundant deployment of the central server components of each iRODS Zone. LEXIS-specific challenges during the setup have included the integration with Keycloak [74] for authentication, and the federation of multiple computing centers over the wide-area network. Holding metadata, our system works towards compliance with the FAIR principles of research data management [19]. B2HANDLE [8] is used to add persistent identifiers to datasets. B2SAFE enables users to request data replication across sites where appropriate.

Performance benchmarks of the system have yielded sustained data-transfer rates beyond $\sim$100 MB/s between our sites in Czech Republic and Germany. Our measurements complement and confirm earlier work [51,53,54], which has used older iRODS versions and has not focused on the setting of a large-scale federation. A typical waiting time of one second beyond the mere transfer time is observed for cross-site data transfers in the DDI. We have uncovered significant differences between data transfer speeds with `icp`, B2SAFE and the Python iRODS Client. B2SAFE showed more strongly decreased rates at small transfer sizes. This pointed towards additional latencies of about a second, caused by a more elaborate management of metadata, persistent identifiers and synchronization. The rates achieved with the Python iRODS Client remained relatively low at large dataset sizes, a problem which needs further investigation. We have, however, already been able to roughly double these rates, optimizing the buffer-size settings within the client. Our results for large files and optimum conditions comply with earlier work [54], where it was shown that iRODS transfers can utilize a large fraction of the network bandwidth when correctly configured.

The first years of operating the DDI have met our expectations; nevertheless, limitations and directions for further improvement were identified. The transfer of many small files arguably challenges complex cross-site data systems by construction. This has been addressed by implementing automated compression mechanisms. To reduce workload on the operators, we aim at an enhanced redundancy setup with less need for manual intervention. We intend our work to give a general guidance for setup and utilization of geographically distributed data stores, conveying a feeling for possibilities and optimization strategies.

Within the EXA4MIND EU project on Extreme Data analytics, and LEXIS Platform operation, we continuously improve our systems. Staging mechanisms will be offered for the DDI to interoperate with local databases and object stores, adapted to applications with structured and unstructured data. In the DDI itself, we are eliminating data-transfer bottlenecks and enabling direct data streams between our computing and data-management systems. The resulting data ecosystem will remain interoperable with EUDAT. In the mid to long term, we aim at offering enhanced connectivity to European Data Spaces as well as other European and International data platforms.

Supporting information

S1 Dataset. Measured data from our speed tests.

This paper is distributed with data from our speed measurements in csv-table format, packed in a zip file (`S1_dataset.zip`), as supplementary information. The supplementary data go beyond what has been shown in the figures, with details and further explorative measurements. The general formatting paradigm of the csv tables is as follows:

- the column separator is a single tabulator sign;
- the first row contains column headings;
- the first column contains run ids or experiment descriptors (where average values and sample standard deviations, where calculated, are denoted as “Average” and “Stddev”, respectively);
- strings are not quoted and do not contain spaces; and
- all cells but those of the first column and row contain values in MB/s; the decimal sign is a point.

The data in the eight files contained in `S1_dataset.zip` is as follows:

(A) `Dataset_hayek25_TestA_cf_fig23_icp_LRZ_to_IT4I.csv`
Contains the speed-test results corresponding to the “LRZ to IT4I (with icp)” part of Fig 4 (note: these data/plot values are repeated in Fig 5). The data columns contain measured values for a file size of 5, 100 and 1 000 MB, respectively. Before the file provides the mere aggregated data for the figures (average, sample standard deviation) in the last two rows, it provides the data from all 20 runs (as executed to obtain average and sample standard deviation) in the first 20 data rows.

(B) `Dataset_hayek25_TestB_cf_fig2_icp_IT4I_to_LRZ.csv`
Is structurally equal to file A), but contains the data for the opposite transfer direction (IT4I to LRZ), used in Fig 4.

(C) `Dataset_hayek25_TestC_cf_fig3_B2SAFE_LRZ_to_IT4I.csv`

Contains the speed-test results corresponding to the “LRZ to IT4I (B2SAFE)” part of Fig 5. While the file is structurally similar to file A), note that it contains one more measurement row for a file size of 10 000 MB to verify the maximum speed and speed saturation behavior as described in the main text.

(D) `Dataset_hayek25_TestD_B2SAFE_IT4I_to_LRZ.csv`
Contains speed-test results corresponding to those in file C), but in the opposite direction (IT4I to LRZ with B2SAFE – not shown in the figure) and only extending to file sizes of 1 000 MB as in the tests A) and B).

(E) `Dataset_hayek25_TestE_cf_fig3_PyPut_LRZ_to_IT4I.csv`
Contains the speed-test results corresponding to the “LRZ to IT4I (Python Cl. ...)” part of Fig 5. The file is equal to file C) in structure.

(F) `Dataset_hayek25_TestF_PyGet_IT4I_from_LRZ.csv`
Contains speed-test results corresponding to those in file E), but in the opposite direction (IT4I to LRZ – not shown in figure). Note that these results, for practical reasons, were obtained by executing the get routine on LRZ side (i.e. we pulled data from IT4I side instead of pushing as in all other tests). Thus, the results are methodically not strictly comparable to those of file E).

(G) `Dataset_hayek25_TestG_cf_fig4_PyModPut_LRZ_to_IT4I.csv`
Contains all speed-test results corresponding to Fig 6, based on our modified version of the `put` routine of the Python iRODS Client. The data columns contain measured values for a file size of 10, 100, 1 000 and 10 000 MB, respectively. We have included no single-run values here, as the rows are reflecting the buffer-size dependency (average and sample standard deviation for a buffer size of 1, 10 and 100 MB).

(H) `Dataset_hayek25_TestH_PyModPut_IT4I_to_LRZ.csv`
Contains speed-test results corresponding to those in file G), but in the opposite direction (IT4I to LRZ – not shown in figure). The file is equal to file G) in structure.

(ZIP)

Acknowledgments

The authors would like to sincerely thank the colleagues from EUDAT for good collaboration. We thank the LRZ and IT4I for providing cloud-computing and other infrastructure essential for this publication. We are indebted to S. Gordon from LRZ for proofreading and wording corrections. Furthermore, we sincerely thank the reviewers for their remarks, which helped to considerably improve the paper.

Author contributions

Conceptualization: Mohamad Hayek, Martin Golasowski, Stephan Hachinger, Jan Martinovič.

Data curation: Stephan Hachinger.

Formal analysis: Mohamad Hayek, Gabriel Lindner.

Funding acquisition: Stephan Hachinger, Kateřina Slaninová, Vít Vondrák, Dieter Kranzlmüller, Jan Martinovič.

Investigation: Mohamad Hayek, Martin Golasowski, Stephan Hachinger, Rubén J. García-Hernández, Johannes Munke, Gabriel Lindner, Philipp Tunka.

Methodology: Mohamad Hayek, Martin Golasowski, Stephan Hachinger, Rubén J. García-Hernández, Johannes Munke, Gabriel Lindner.

Project administration: Martin Golasowski, Stephan Hachinger, Kateřina Slaninová, Dieter Kranzlmüller, Jan Martinovič.

Resources: Martin Golasowski, Philipp Tunka, Vít Vondrák, Dieter Kranzlmüller, Jan Martinovič.

Software: Mohamad Hayek, Martin Golasowski, Rubén J. García-Hernández, Gabriel Lindner.

Supervision: Mohamad Hayek, Stephan Hachinger, Rubén J. García-Hernández, Kateřina Slaninová, Vít Vondrák, Dieter Kranzlmüller, Jan Martinovič.

Validation: Mohamad Hayek, Martin Golasowski, Gabriel Lindner.

Visualization: Mohamad Hayek, Stephan Hachinger.

Writing – original draft: Mohamad Hayek, Martin Golasowski, Stephan Hachinger, Rubén J. García-Hernández, Johannes Munke.

Writing – review & editing: Mohamad Hayek, Martin Golasowski, Stephan Hachinger, Rubén J. García-Hernández, Johannes Munke, Gabriel Lindner, Kateřina Slaninová, Philipp Tunka, Vít Vondrák, Dieter Kranzlmüller, Jan Martinovič.

References

1. Foster I, Kesselman C. The grid 2: blueprint for a new computing infrastructure. San Francisco, CA: Morgan Kaufmann; 2003.
2. Pennington R. Terascale clusters and the TeraGrid. In: Proceedings of the 6th International Conference on High-Performance Computing in Asia-Pacific Region (HPC Asia 2002), Bangalore, India. 2002. p. 407–13.
3. Shiers J. The worldwide LHC computing grid (worldwide LCG). Computer Physics Communications. 2007;177(1):219–23.
4. Kranzlmüller D, de Lucas JM, Öster P. The European Grid Initiative (EGI). Remote instrumentation and virtual laboratories. Springer US; 2010. p. 61–6. https://doi.org/10.1007/978-1-4419-5597-5_6
5. Barreiro Magino F, Cameron D, Di Girolamo A, Filipcic A, Glushkov I, Legger F, et al. Operation of the ATLAS distributed computing. EPJ Web Conf. 2019;214:03049.https://doi.org/10.1051/epjconf/201921403049
6. Lecarpentier D, Wittenburg P, Elbers W, Michelini A, Kanso R, Coveney P, et al. EUDAT: a new cross-disciplinary data infrastructure for science. IJDC. 2013;8(1):279–87. https://doi.org/10.2218/ijdc.v8i1.260
7. EUDAT Collaborative Data Infrastructure. B2SAFE-EUDAT. EUDAT Collaborative Data Infrastructure. 2023. [cited 4 July 2025]. https://www.eudat.eu/services/b2safe
8. EUDAT Collaborative Data Infrastructure. B2HANDLE-EUDAT. In: [Internet]. EUDAT CDI; 2023 [cited 4 July 2025]. https://www.eudat.eu/services/b2handle
9. Almeida AV d, Borges MM, Roque L. The European open science cloud: a new challenge for Europe. In: Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality. 2017.
10. EU Commission. A European strategy for data. 2020. [cited 3 Mar 2024]. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020DC0066
11. Scionti A. HPC, cloud and big-data convergent architectures: the LEXIS approach. In: Barolli L, Hussain F, Ikeda M, editors. CISIS 2019. Advances in Intelligent Systems and Computing. Cham: Springer; 2020. p. 200–12.
12. Golasowski M, Martinovic J, Krenek J, Slaninova K, Levrier M, Harsh P. The LEXIS platform for distributed workflow execution and data management. In: Terzo O, Martinovic J, editors. HPC, Big Data, and AI convergence towards exascale. Boca Raton (FL): CRC Press; 2022. p. 17–35.
13. Hachinger S, Golasowski M, Martinovic J, Hayek M, Garcıa-Hernandez RJ, Slaninova K. Leveraging high-performance computing and cloud computing with unified big-data workflows: the LEXIS project. In: Curry E, Auer S, Berre AJ, Metzger A, Perez MS, Zillner S, editors. Technologies and applications for big data value. Cham: Springer; 2022. p. 159–80.
14. Parodi A, et al. LEXIS weather and climate large-scale pilot. In: Barolli L, Poniszewska-Maranda A, Enokido T, editors. CISIS 2020. Advances in Intelligent Systems and Computing. Cham: Springer; 2021. p. 267–77.
15. Magarielli D, Spano E, Fondelli T, Pampaloni D, Andreini A, Savio P. Advanced engineering platform supporting CFD simulations of aeronautical engine critical parts. In: Terzo O, Martinović J, editors. HPC, big data, and AI convergence towards exascale. Boca Raton (FL): CRC Press; 2022. p. 105–29.
16. Goubier T, et al. Earthquake and tsunami workflow leveraging the modern HPC/cloud environment in the LEXIS project. In: Barolli L, Nishino H, Enokido T, Takizawa M, editors. Advances in networked-based information systems. Cham: Springer; 2020. p. 223–34.
17. Munke J, Hayek M, Golasowski M, García-Hernández RJ, Donnat F, Koch-Hofer C. Data system and data management in a federation of HPC/cloud centers. In: Terzo O, Martinović J, editors. HPC, big data, and AI convergence towards exascale. Boca Raton (FL): CRC Press; 2022. p. 59–77.

18. Xu H, Russell T, Coposky J, Rajasekar A, Moore R, de Torcy A. iRODS primer 2: integrated rule-oriented data system. Williston, VT: Morgan & Claypool Publishers; 2017.
19. Wilkinson MD, Dumontier M, Aalbersberg IJJ, Appleton G, Axton M, Baak A, et al. The FAIR guiding principles for scientific data management and stewardship. Sci Data. 2016;3:160018. https://doi.org/10.1038/sdata.2016.18 PMID: 26978244
20. IT4Innovations National Supercomputing Center. LEXIS Platform 2 components as a part of the EuroHPC Federation Platform - LEXIS Platform 2. 2025. [cited 11 June 2025]. https://www.lexis.tech/lexis-platform-2-components-part-of-the-federation-platform
21. Weil SA, Brandt SA, Miller EL, Long DDE, Maltzahn C. Ceph: a scalable, high-performance distributed file system. In: Proceedings of the 7th Symposium on Operating Systems Design and Implementation. 2006. p. 307–20.
22. Adde G, Chan B, Duellmann D, Espinal X, Fiorot A, Iven J, et al. Latest evolution of EOS filesystem. J Phys: Conf Ser. 2015;608:012009. https://doi.org/10.1088/1742-6596/608/1/012009
23. Boyer EB, Broomfield MC, Perrotti TA. GlusterFS one storage server to rule them all. In: Computing and Information Technology Student Mini-Showcase. 2012.
24. Venner J, Wadkar S, Siddalingaiah M. Pro Apache Hadoop. 2nd ed. Berkeley, CA: Apress; 2014.
25. Hupfeld F, Cortes T, Kolbeck B, Stender J, Focht E, Hess M, et al. The XtreemFS architecture—a case for object￾based file systems in Grids. Concurrency and Computation. 2008;20(17):2049–60. https://doi.org/10.1002/cpe.1304
26. Amazon Web Services, Inc. Cloud Object Storage – Amazon S3 – Amazon Web Services. 2023. [cited 30 June 2025]. https://aws.amazon.com/de/s3/
27. Fuhrmann P, Antonacci M, Donvito G, Keeble O, Millar P. Smart policy driven data management and data federations. In: Proceedings of the International Symposium on Grids and Clouds (ISGC) 2018 in conjunction with Frontiers in Computational Drug Discovery – PoS. 2018. 001.
28. Wrzeszcz M, Trzepla K, Slota R, Zemek K, Lichon T, Opiola L, et al. Metadata organization and management for globalization of data access with Onedata. In: Wyrzykowski R, Deelman E, Dongarra JJ, Karczewski K, Kitowski J, Wiatr K, editors. Parallel Processing and Applied Mathematics - 11th International Conference, PPAM 2015, Krakow, Poland, September 6-9, 2015. Revised Selected Papers, Part I. Springer; 2015. p. 312–21.
29. Barisits M, Beermann T, Berghaus F, Bockelman B, Bogado J, Cameron D, et al. Rucio: scientific data management. Comput Softw Big Sci. 2019;3(1):11. https://doi.org/10.1007/s41781-019-0026-3
30. Poenaru R, Simon M. Declarative API - implementation overview within the XRootD framework. In: 19th RoEduNet Conference: Networking in Education and Research (RoEduNet). 2020. p. 1–8.
31. The Hyve BV. Fairspace | The Hyve. The Hyve BV. 2025. [cited 16 June 2025]. https://www.thehyve.nl/services/fairspace
32. Bayer Group, Contributors. COLID tech docs. Bayer Group/GitHub. 2025. [cited 16 June 2025]. https://bayer-group.github.io/COLID-Documentation
33. Gen3 Community and Steering Committee. Gen3 - Technical Intro. In: [Internet]. Gen3; 2025 [cited 16 June 2025]. https://gen3.org/resources/developer
34. International Data Spaces Association. Dataspace Protocol 2024-1 | IDS Knowledge base. In: [Internet]. IDSA; 2024 [cited 16 June 2025]. https://docs.internationaldataspaces.org/ids-knowledgebase/dataspace-protocol
35. Halchenko YO, Meyer K, Poldrack B, Solanky DS, Wagner AS, Gors J, et al. DataLad: distributed system for joint management of code, data, and their relationship. J Open Source Softw. 2021;6(63):3262. https://doi.org/10.21105/joss.03262 PMID: 39469147
36. Foster I. Globus online: accelerating and democratizing science through cloud-based services. IEEE Internet Comput. 2011;15(3):70–3. https://doi.org/10.1109/mic.2011.64
37. Molina JF, Forti A, Girone M, Sciaba A. Operating the worldwide LHC computing grid: current and future challenges. J Phys: Conf Ser. 2014;513(6):062044. https://doi.org/10.1088/1742-6596/513/6/062044
38. Bolton R, Campana S, Ceccanti A, Espinal X, Fkiaras A, Fuhrmann P, et al. ESCAPE prototypes a data infrastructure for open science. EPJ Web Conf. 2020;245:04019. https://doi.org/10.1051/epjconf/202024504019
39. ESCAPE Consortium. ESCAPE Deliverable D2.2 – Assessment and analysis of performance of the first pilot data lake. In: [Internet]. ESCAPE Consortium; 2021 [cited 16 June 2025]. https://projectescape.eu/sites/default/files/ESCAPE-D2.2-v1.0.pdf
40. Chudoba J, Svatos M, Mikula A, Voka´˘c P, Chudoba M. Extension of local dCache instance capacity using national e-infrastructure. In: Proceedings of International Symposium on Grids and Clouds (ISGC) 2023 – PoS. 2023. 005.
41. Allcock W, Bresnahan J, Kettimuthu R, Link M. The globus striped GridFTP framework and server. In: SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. 2005. p. 54–54.
42. Spinuso A, et al. The VERCE science gateway: enabling user friendly seismic waves simulations across European HPC infrastructures. In: EGU General Assembly Conference Abstracts. 2014. p. 6141.
43. CompBioMed Consortium. CompBioMed Deliverable D5.5 – Preparing Data Infrastructures for Large-Scale Resources: Report on the Optimisation Activities. 2021. [cited 16 June 2025]. https://www.compbiomed.eu/wp-content/uploads/2021/02/D5.5_Preparing-Data-Infrastructures-for-largescale-resources_V2.0.pdf
44. Kawai Y, Hasan A. High-Availability iRODS System (HAIRS). In: Proceedings of the iRODS User Group Meeting 2010: Policy-Based Data Management, Sharing and Preservation, Chapel Hill, NC. 2010.

45. Salomoni D, et al. Indigo-datacloud: foundations and architectural description of a platform as a service oriented to scientific computing. CoRR. 2016. https://doi.org/10.48550/arXiv.1603.09536

46. Sipos G, La Rocca G, Antonio F, Elia D, Nassisi P, Fiore S, et al. Scientific data spaces - experiences from the EGI-ACE project. Open Res Eur. 2024;4:136. https://doi.org/10.12688/openreseurope.17418.1 PMID: 39219788

47. Pretzsch S, Drees H, Rittershaus L. Mobility data space: a secure data space for the sovereign and cross-platform utilization of mobility data. In: Otto B, ten Hompel M, Wrobel S, editors. Designing data spaces: The ecosystem approach to competitive advantage. Springer; 2022. p. 343–61.

48. van der Horst E. Open-source tools for research data management: a brief overview and evaluation; 2022. Zenodo. https://doi.org/10.5281/zenodo.7404626

49. PROCESS Consortium. Design of data infrastructure for extreme-large data sets. 2018. https://www.process-project.eu/wp-content/uploads/2020/02/PUBLIC_PROCESS_D5.1_Design_of_data_infrastructure_v1.0.pdf

50. Borrow J, Plante PL, Aguirre J, Williams PKG. Making research data flow with python. In: Proceedings of the Python in Science Conference. 2024. p. 236–46. https://doi.org/10.25080/hwga5253

51. Depardon B, Le Mahec G, S é guin C. Analysis of six distributed file systems. INRIA. 2013. https://hal.inria.fr/hal-00789086

52. Lindner G. Benchmarking of a large-scale distributed data management platform. Ludwig Maximilian University of Munich; 2022.

53. Hunich D, Muller-Pfefferkorn R. Managing large datasets with iRODS—a performance analysis. In: Proceedings of the International Multiconference on Computer Science and Information Technology. 2010. p. 647–54. https://doi.org/10.1109/imcsit.2010.5679862

54. Russell TG, Coposky JM, Keller B. iRODS Networking Performance. 2016. https://irods.org/uploads/2016/09/russell2016-iRODS-networking-419.pdf

55. Beckman P, Nadella S, Trebon N, Beschastnikh I. Spruce: a system for supporting urgent high-performance computing. In: Gaffney P, Pool J, editors. Grid-Based Problem Solving Environments. IFIP International Federation for Information Processing; 2007. p. 295–311.

56. Pringle G, Hachinger S, Hayek M, Kˇrenek J, Za´vodszky G, Martinoviˇc J, et al. A resilient workflow to control a biomedical HPC simulation in an urgent computing setting. Zenodo. 2024.

57. Fielding RT. Architectural styles and the design of network-based software architectures. University of California, Irvine; 2000.

58. Sakimura N, Bradley J, Jones MB, de Medeiros B, Mortimore C. OpenID Connect Core 1.0 Incorporating Errata Set 1. OpenID Foundation. 2014. https://openid.net/specs/openid-connect-core-1.0.html

59. Keycloak Authors. Keycloak. In: [Internet]. Keycloak Authors/Linux Foundation; 2025 [cited 22 June 2025]. https://www.keycloak.org

60. Schmuck FB, Haskin RL. GPFS: a shared-disk file system for large computing clusters. In: Proceedings of the Conference on File and Storage Technologies. 2002. p. 231–44.

61. DOI Foundation. Home Page. In: [Internet]. DOI Foundation; 2022 [cited 23 June 2025]. https://www.doi.org

62. GSI Helmholtzzentrum für Schwerionenforschung GmbH. Virtual XRootD Workshop (14-15. Dezember 2020) – GSI Indico. 2020. [cited 16 June 2025]. https://indico.gsi.de/event/11858

63. Pansanel J. La gestion des donnees distribuees: etude comparative de Rucio et iRODS. In: [Internet]. Data-Terra; 2022 [obtained 02 Apr 2025 via request to author]. 2022. https://www.gaia-data.org/evenement/atelier-2

64. Hedges M, Hasan A, Blanke T. Curation and preservation of research data in an iRODS data grid. In: Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007); 2007. p. 457–64.

65. Meneses R, Mari M. CS3MESH4EOSC Deliverable D5.3 – Final report on dissemination and exploitation of project results. Zenodo. 2023.

66. Calatrava A, Asorey H, Astalos J, Azevedo A, Benincasa F, Blanquer I, et al. A survey of the European Open Science Cloud services for expanding the capacity and capabilities of multidisciplinary scientific applications. Computer Science Review. 2023;49:100571. https://doi.org/10.1016/j.cosrev.2023.100571

67. Ferriter K. OpenID Connect Authentification in iRODS. In: [Internet]. iRODS Consortium; 2018 [cited 20 Feb 2024]. https://irods.org/uploads/2018/Ferriter-RENCI-OpenID_Connect_Authentication_in_iRODS-slides.pdf

68. James J. Configuring iRODS for High Availability. In: [Internet]. iRODS Consortium; 2015 [cited 20 Feb 2024]. https://irods.org/2015/07/configuring-irods-for-high-availability/

69. James J. UGM 2016 - High Availability with iRODS. In: [Internet]. Slides.com; 2016 [cited 20 Feb 2024]. https://slides.com/justinjames-1/ugm2016-high-availability

70. Stonebraker M, Rowe LA. The design of POSTGRES. SIGMOD Rec. 1986;15(2):340–55. https://doi.org/10.1145/16856.16888

71. Depuydt J. Setup a redundant PostgreSQL database with repmgr and pgpool — Jensd's I/O buffer. In: [Internet]. J. Depuydt; 2015 [cited 7 July 2025]. http://jensd.be/591/linux/setup-a-redundant-postgresql-database-with-repmgr-and-pgpool

72. Tarreau W. The History of HAProxy. In: [Internet]. HAProxy Technologies, LLC; 2019 [cited 22 June 2025]. https://www.haproxy.com/blog/the-history-of-haproxy

73. Jones M, Campbell B, Mortimore C. JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants. RFC Editor. 2015. https://doi.org/10.17487/rfc7523

74. García-Hernández RJ, Golasowski M. Supporting Keycloak in iRODS systems with OpenID authentication. In: [Internet]. CS3 2020 - Workshop on Cloud Storage Synchronization & Sharing Services; 2020. https://indico.cern.ch/event/854707/contributions/3681126

75. LaRose, Chris and others. GitHub - irods/python-irodsclient: A Python API for iRODS. In: [Internet]. GitHub; 2022 [cited 20 Feb 2024]. https://github.com/irods/python-irodsclient

76. Russell TG. iput numThreads (Issue 6447, comment 1163530154). In: [Internet]. GitHub; 2022 [cited 06 Dec 2025]. https://github.com/irods/irods/issues/6447#issuecomment-1163530154

77. iRODS Consortium. iRODS – Frequently Asked Questions. 2024. [cited 11 Mar 2024]. https://irods.org/faq

78. LEXIS Consortium. LEXIS Deliverable D2.6 – Infrastructure validation and assessment report. LEXIS Consortium. 2022. https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5e6894d86&appId=PPGMS

79. Ellert M, Konstantinov A, Konya B, Smirnova O, Waananen A. Performance evaluation of the GridFTP within the NorduGrid project. arXiv preprints. 2002. https://doi.org/10.48550/arXiv.cs/0205023

80. Schuller BT, Pohlmann T. UFTP: high-performance data transfer for UNICORE. In: Proceedings of the 2011 UNICORE Summit, Torun, Poland. 2011. p. 135–42.

81. Rosado T, Bernardino J. An overview of openstack architecture. In: Proceedings of the 18th International Database Engineering & Applications Symposium. 2014. p. 366–7.

82. Thomas SM. PostgreSQL High Availability Cookbook. Birmingham, UK: Packt Publishing Ltd.; 2017.

83. Brewer EA. Kubernetes and the path to cloud native. In: Proceedings of the Sixth ACM Symposium on Cloud Computing. SoCC '15. New York, NY, USA: Association for Computing Machinery; 2015. p. 167.

84. James J. New iRODS APIs: Presenting as HTTP and S3. In: [Internet]. CS3 2024 - Workshop on Cloud Storage Synchronization & Sharing Services; 2024 [cited 25 Dec 2025]. https://indico.cern.ch/event/1332413/contributions/5749455

85. Dam T, Klausner LD, Neumaier S, Priebe T. A survey of dataspace connector implementations. In: CEUR Workshop Proceedings, Naples, Italy. 2023. https://ceur-ws.org/Vol-3606/paper41.pdf

86. DataCite Metadata Working Group. DataCite Metadata Schema Documentation for the Publication and Citation of Research Data and Other Research Outputs. DataCite; 2024 [cited 4 July 2025]. https://doi.org/10.14454/g8e5-6293