A High-Performance Persistent Identification Concept

Fatih Berber*, Philipp Wieder*, Ramin Yahyapour*‡
*Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen (GWDG), Germany
‡University of Göttingen
{fatih.berber, philipp.wieder, ramin.yahyapour}@gwdg.de

***Abstract*—The immense research dataset growth requires new strategies and concepts for an appropriate handling at the corresponding research data repositories. This is especially true for the concept of Persistent Identifiers (PIDs), which is designed to provide a persistent identification layer on top of the address-based resource retrieval methodology of the current Internet. For research datasets, which are referenced, further processed and transferred, such a persistent identification concept is highly crucial for ensuring a long-term scientific exchange.**
Often, each individual research dataset stored in a research data repository, is registered at a particular PID registration agency in order to be assigned a globally unique PID. However, for the explosive growth of research datasets this concept of registering each individual research dataset is in terms of the performance highly inappropriate. Therefore, the focus of this work is on a concept for enabling a high-performance persistent identification of research datasets.
Recent research data repositories often are equipped with a built-in naming component for assigning immutable and internally unique identifiers for their incoming research datasets. Thus, the core idea in this work is to enable these internal identifiers to be directly resolvable at the well-known global PID resolution systems. This work will therefore provide insight into the implementation of this idea into the well-known Handle System. Finally, in this work, we will provide an experimental evaluation of our proposed concept.

I. Introduction

In recent years there is an explosive growth of research datasets. This growth has been led the architecture of research data repositories to become increasingly sophisticated. To strengthen the scientific exchange each incoming research dataset is assigned a globally unique Persistent Identifier (PID). For research datasets, which are referenced, further processed and transformed a sustainable access is highly important. PIDs are therefore an important concept for the discipline of research data management. However, for the immense research data growth, the concept of registering each individual research dataset at a particular PID registration agency in order to get a globally unique and persistent identifier, is highly inappropriate. This is due to the fact that this concept often causes a significant performance loss at the ingest workflow into the corresponding research data repository.
The focus of this work is therefore on a high-performance persistent identification concept, which is appropriate for huge amounts of research datasets stored in research data repositories.
Recent research data repositories usually a have built-in naming component, which already assigns immutable and internally unique identifiers for the incoming research datasets. Our basic approach is to enable these internal identifiers to be directly resolvable at the well-known global PID resolution systems without being registered at a particular PID registration agency. Such an approach will eliminate the overhead caused by the PID assignment process. As a consequence, such an approach will improve the performance of the ingest workflow into the research data repository. In addition, we will provide insight into the implementation of this concept into the established Handle System, which is one of the most important PID systems.
The remainder of this paper is structured as follows. Section II gives an overview of the related work. In Section III, we will discuss the core idea behind the concept of PIDs. In addition, we will reveal the fundamental performance problem caused by PID systems. Finally, we will provide a novel concept for enabling high-performance persistent identification of research datasets. In section IV, we will provide two different solutions for the implementation of our proposed concept into the established Handle System. In order to evaluate our approach, in section V, we will experimentally analyze the effect of the implementation on the performance of the resolution process. Finally, section VI presents our conclusions and gives hints for future work.

II. Related Work

The concept of persistent identification is not new, however, its importance has been significantly increased due to the explosive growth of research datasets. Various different PID systems have been developed in order to provide a persistent identification layer on top of the address-based retrieval methodology of the current Internet.
Among them, the Handle System [1] is the most sophisticated one. It has been developed to serve as a general-purpose naming system, which can be included in a research data repository. Because of its elaborated concept, many other important PID systems are based on the Handle System. The well-known DOI system [2] is the most prominent of such PID systems. The ePIC system [3] is another PID system based on the Handle System. Since the Handle Protocol does not support a native HTTP-interface, the common goal of these PID systems is to provide a HTTP interface on top of the underlying Handle System.

978-1-5090-3315-7/16/$31.00 ©2016 IEEE

A much more lightweight PID concept is provided by the Archival Resource Key (ARK) identifiers [4]. In contrast to the Handle System, the ARK concept is in principle composed of a naming scheme and URL-based conventions for defining resolution results. In the case of the Handle System, the resolution of an individual *Handle* always results in the corresponding Handle Record, which is comparable to the resolution of a Domain Name in the DNS system.
Comparable to the ARK identifier scheme is the Uniform Resource Name (URN) [5], which is purely a scheme for providing various namespaces. One of the most important naming schemes is the URN:NBN namespace, which is used to uniquely identify resources in the area of libraries. The main problem of the URN concept is that there is no central resolution system capable of resolving all namespaces.
The research efforts related to persistent identifiers so far, do not substantially focus on the performance aspects of the persistent identification concept. However, since science has discovered the Web as a platform for scientific exchange, there is a massive growth of research datasets stored in research data repositories. The focus of this work is therefore on an appropriate persistent identification concept for huge amounts of research datasets stored in such research data repositories.
A first substantial work for PID system implementation is done in [6]. The authors introduce a set of existing PID systems and describe them. In addition, they provide high level guidelines and recommendations about an appropriate deployment in terms of research data management.
An introductionary work is presented in [7]. The author emphasizes the usage of PIDs for various research datasets, by providing guidelines for the granularity issue of research datasets.
A comparison of several PID systems is done in [8], whereas the focus is to provide an overview about their functionalities. Another functional PID system comparison is provided by [9].
The work in [10] provides a hierarchical architecture, which reflects the architecture of the DNS system. Their core objective is to overcome a centralized system in order to avoid a single point of failure.
An interoperability framework for PID systems is introduced by [11]. Their basic approach is an ontological refinement of metadata sets contained in PID-records, which enables a common information set for the various PID systems.
A similar approach is provided by [12]. Their focus is the integration of common abstract datatypes into PID-records, which can be consumed by machine actors. In contrast to [11], their core emphasize is to provide appropriate semantical information about the identified dataset itself.
The concept of PIDs is actually intended for the identification of research datasets, but currently their usage is also discussed in different fields. We assume that their application fields will even more increase in the coming years.
In the field of scientist identification, the authors in [13] introduce the concept of persistent identification for scientists. In their concept, an individual PID-record corresponds to a single scientist, which also contains the respective bibliography.
In the work [14], the authors are discussing the usage of PIDs in the field of Named Data Network (NDN). Their focus is to use existing PID concepts within NDN environment for delivering big datasets. A similar approach is done in [15]. The authors emphasize that legacy data should be still operational in the future NDN. In contrast to [14], they focus on the Handle System and consider all aspects for operating the Handle System within NDN environments. Therefore, they introduce concepts for enabling the transition from the Handle data model into the NDN data model. In addition, they provide a performance comparison of the PID management between the traditional network and the future NDN.
An abstract framework for a distributed research data repository is provided by [16]. In this framework, the Handle System is proposed to be the central component for enabling the distributed components to act as a single system.

All the research efforts listed above, do neither provide a concept for persistent identification of huge amounts of research datasets, nor they substantially address the performance aspect at all.

III. High-Performance Persistent Identification

A. Basic Idea of Persistent Identifiers

The basic idea behind PIDs is to provide a persistent identification layer on top of the current address-based methodology for accessing research datasets. This is an important concept in research data management for providing sustainable access to research datasets stored in research data repositories. In principle, a PID is composed of an opaque identifier string, which does not contain any information about the actual content of the identified research dataset itself.
In their functionality PIDs are very much comparable with Domain Names, which are a human-friendly representation of IP-addresses corresponding to computer hostnames. Hence, the resolution of a Domain Name results in the IP-address of the corresponding computer host. In contrast to that, the resolution of a PID reveals the current Internet address of the corresponding research dataset. Both, PIDs and Domain Names are composed of a specific hierarchical naming scheme, which enables global uniqueness and resolvability. Usually, the resolution request is processed in an delegated manner. In the case of Domain Names, the resolution request is delegated to the responsible *Nameserver*, whereas in the case of PIDs, it is delegated to the responsible naming authority service. Also common for both systems is that a PID and/or a Domain Name is registered once and resolved many times.
The fundamental difference in both systems lies in the workload at the registration process. From the registration agency point of view, the common goal is to process as much as possible requests in a short time. However, from registrant point of view, especially for the PID assignment it is highly crucial to enable a fast registration of the incoming research datasets. This is even more crucial for research data repositories subjected to an explosive research datasets growth. Thus, the respective software agents steadily issue PID registration agencies for registering the incoming research datasets. Hence,

the concept of registering of each individual research dataset at a particular PID system is not appropriate for the massive amount of research datasets.
In the following subsections, we will therefore introduce a novel concept for persistent identification of research datasets contained in huge research data repositories.

B. Fundamental Performance Problem

The fundamental problem caused by current PID systems is that they require a research data repository to implement a particular protocol and/or data model in order to be integrable into the respective global resolution infrastructure.
The Handle System for instance, requires the implementation of the Handle Protocol [17], which enables the integration into the corresponding global resolution system. In fact, this system can be considered as an imitation of the DNS system for research dataset identification. A resolution in the DNS system results in a DNS-record containing the corresponding IP-address. In contrast to that, in the Handle System it results in a Handle Record, which contains the corresponding current Internet address of the identified research dataset. However, the native Handle Protocol lacks in the support of a REST interface. In the present time, where the main platform used for scientific exchange is provided by the Web, recent research data repositories are build as Web applications. Hence, for interoperability, a REST interface is also highly important for PID systems.
A REST module in the Handle System software package [18] is only available since version 8, which was recently released. Thus, various PID service providers have been emerged, whereas the underlying Handle System PID system is extended by an additional REST interface module. In addition, these service providers take over the requirement of implementing the Handle Protocol.
It should be noted that persistency is not limited to a technical solution, in fact, it is purely a matter of service. The role of technical solutions are solely limited to the support of appropriate service commitments, which are promised by these PID service providers.
The DOI system is the most prominent of these PID service providers, in addition to a REST interface, it also provides a search index for the registered research datasets. At the registration process, for each research datasets a set of mandatory metadata records has to be provided [19]. The focus of DOI is on research datasets, which already are in finalized state of research data lifecycle.
The ePIC system is another PID service provider based on the Handle System. By providing a REST interface on top of the underlying Handle System, it is very similar to the DOI system, however in ePIC, the metadata model is much more flexible. Each user can define its own metadata model that is imposed into the individual Handle Records. In contrast to the DOI system, ePIC PIDs can be assigned to any research dataset at any state of the research data lifecycle.
Also an established PID system is the ARK identifier [4], which in comparison to the Handle System is much more lightweight. The ARK identifier is in principle a naming scheme for which a set of URL-based conventions is defined. These conventions form the so called THUMP protocol [20]. Therefore, to be integrated into the global resolution infrastructure of the ARK identifier system, each research data repository is compelled to implement the THUMP protocol.
A PID service provider based on the ARK identifier system is provided by the EZID system [21]. Similar to the service providers based on the Handle System, this system also provides a REST interface for the registration of research datasets. Also the implementation of the THUMP protocol is covered by the EZID system.
To summarize this discussion, to assign PIDs for research datasets, research data repository system designers are compelled to choose between one of the following two options:

a) Setting up an **internal naming authority service**, which complies with the protocol and/or data model of a particular PID system.
b) Making use of an **external naming authority service**, which is hosted by an external PID service provider.

Option a), usually involves in an considerable implementation effort. In addition, this option requires an appropriate maintenance of the service. If the internal naming authority service is implemented afterwards as an additional component into the research data repository, this could cause a significant performance loss of the ingest workflow. This is caused by the overhead of the additional registration at the internal naming service for assigning PIDs for the incoming research datasets. Since in option b), the naming service is already offered by an external service provider, there is no implementation and maintenance effort that has to be taken into account. However, because of the network latency, the performance loss at the ingest workflow usually is even more significant than in option a).
To overcome the problems posed by both options, in the following, we will propose a novel persistent identification concept. In the next subsection, we will therefore first provide an abstract view on recent research data repositories to get a better understanding of their essential functionality. This is needed to deduce a novel persistent identification concept, which suited for research data repositories subjected to an immense dataset growth.

C. Abstract View on Research Data Repository

An abstract view on research data repositories is illustrated in Figure 1. As can be seen from that Figure, a research data repository in principle is a research data silo, which offers the following five basic operations: Create, Read, Update, Delete, and Search (CRUDS). Originally, the CRUD paradigm [22] is established in the world of databases. In addition to the CRUD operations, many research data repositories also offer a search operation, which provides an aggregation of different datasets containing a common information pattern.
The architecture of data silos for storing research datasets is often composed of various components, whereas each of which provides a specific functionality. Usually, many of

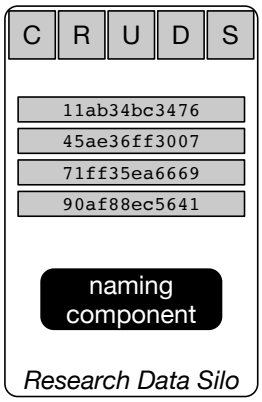

Fig. 1. Abstract view on a research data respository

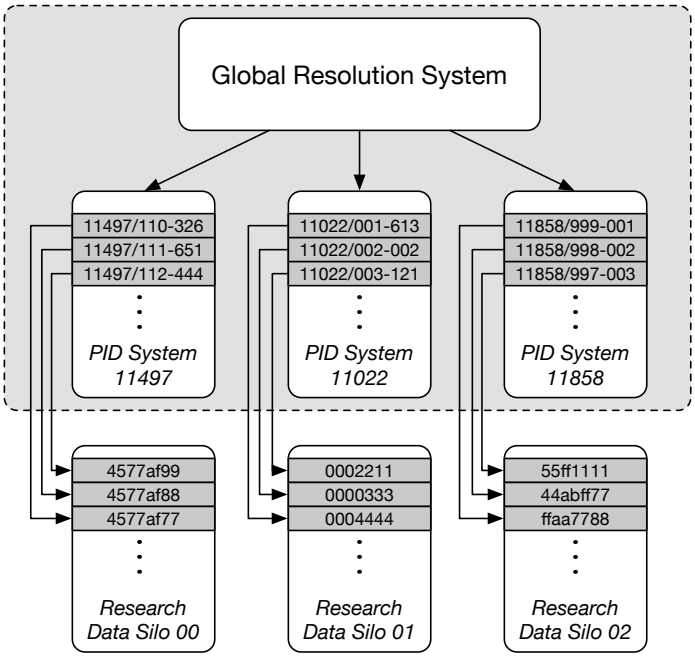

Fig. 2. Research data silo registering each of its datasets at assigned PID system

recent research data silos have a built-in naming component for generating opaque and immutable identifiers for their incoming datasets. Therefore, PIDs can also be considered as official (public) identifiers. However, as mentioned above, this additional (public) identifier assignment often causes a significant performance loss at the ingest workflow. Since the public identifier is assigned by a particular naming authority, which is integrated into a specific global resolution system, it enables a sustainable access to the corresponding research dataset stored in a research data silo. This is depicted in Figure 2.

To overcome the performance loss of the additional (public) identifier assignment, for such research data silos it would be much more efficient to enable the internal identifiers to be globally resolvable. This would eliminate the whole overhead of the additional identifier assignment.

In the next subsection, we will therefore focus on the approach of enabling these internal identifiers to be globally resolvable, without an additional registration at a globally connected naming authority.

D. Global Resolvability of Internal Identifiers

The global resolvability of a PID is in principle enabled by the imposed hierarchy in the identifier string itself. Commonly, a single PID is composed of a prefix and a suffix. The prefix uniquely identifies the responsible naming authority to which the corresponding resolution requests are delegated to. The suffix for instance, is a locally unique identifier assigned by the naming authority to an individual research dataset. Hence, by prefixing each locally unique suffix becomes globally unique. If we assume, that each research data silo is assigned a dedicated prefix, by prefixing the internal identifiers with the dedicated prefix, these internal identifiers therefore become globally unique. We will call a prefix, which is exclusively dedicated to an individual research data silo a *Research Data Silo Identifier* (DSID). For the global resolvability of these internal identifiers prefixed with a DSID, the responsible naming authority has to implement a special functionality. This special functionality is necessary for enabling the internal identifiers to be resolvable without the requirement of registering each individual research dataset.

In the following subsection, we will provide insight into this *special functionality*.

1) Access Request Syntax Registration: Usually, an individual naming authority is responsible for several prefixes. To respond a resolution request, a naming authority service has to lookup into its database for the entry associated with the requested PID string. In principle, each entry contains the following three information entities:

a) The current Internet address of the research data silo, where the identified research dataset is stored.
b) The current access request syntax, which is necessary for retrieving datasets from that research data silo.
c) The internal identifier of the research dataset stored in the research data silo.

Hence, for all PIDs prefixed with the same DSID, the entities a) and b) usually are the same for all the associated entries. Therefore, to enable the global resolvability of the internal identifiers, for a research data silo assigned a DSID it is sufficient to register its current offered access request syntax at the naming authority. Thus, the *special functionality* of this naming authority is to dynamically extend the stored individual current access syntax with the incoming PID string.

Instead of registering each individual research dataset, with this concept there is only a one-off registration for each research data silo required. The concept of *Access Request Syntax Registration* is therefore appropriate for research data repositories subjected to an immense dataset growth.

Another benefit of this concept is that each change in a research data silo has only to be tracked in the single database

entry of the corresponding naming authority service. Otherwise, all entries associated with research datasets of that research data silo have to be updated in the naming authority service.
However, individual changes in research datasets, which affect the Internet address are not covered with this concept. In such a case, the naming authority service, could be used again to track the individual Internet address of the corresponding research dataset.
Another issue is that in addition to a persistent identification layer, recent PID systems do also provide the retrieval of additional information about the identified research dataset. However, often research data silos do not resolve an internal identifier directly to the actual research dataset itself, instead they usually resolve to a so called "landing page". In addition to the current Internet address, often such a "landing page" do also provide descriptive information about the identified research dataset. Hence, the corresponding PID do actually resolve to that "landing page" instead to the actual research dataset itself. For DOIs it is even mandatory to register the "landing page" of a research dataset instead of the actual Internet address of the research dataset itself.
Ultimately this means, that the feature of enabling the retrieval of additional descriptive information is already offered by many recent research data silos. Hence, this again confirms the sufficiency of registering the current offered access request syntax for many recent research data silos.

2) Search Request Syntax Registration: Since science has discovered the Web as platform for scientific exchange, various research data silos have been emerged. The real challenge is therefore to enable the exploitation of the full potential from all these research data silos. Since persistent identifier systems could be considered as central research data silo registries, these PID systems could be used as search engines on top of the registered research data silos.
Therefore, a naming authority service could be even equipped with a further *special functionality*, namely a search interface. Such a naming authority service would then act as a federated search engine on top of all registered data silos. A search query submitted to that naming authority service would then be delegated to all the registered research data silos.
Technically, this could be realized by the specification of the individual search interface at the one-off registration of the corresponding research data silo. However, this requires further research to develop appropriate filtering and mapping strategies at the naming authority service.
Since the focus of this work is on a high-performance persistent identification concept, a detailed discussion of the search functionality would go beyond the scope of this paper.

E. Summary

In this section we have revealed that the root cause for the performance problems caused by current PID systems is originated in the protocols and/or data models, which they require to implement. Since many recent research data repositories already have a built-in naming component for generating immutable and locally unique identifiers for their research datasets, we have proposed to enable these internal identifiers to be globally resolvable. Since the resolution of a PID is enabled because of the imposed hierarchy in the identifier string itself, we have proposed to prefix the internal identifiers with a *Research Data Silo Identifiers* (DSID). A DSID is basically a prefix, which is exclusively dedicated to all research datasets of an individual research data silo. The resolution request for all these internal identifiers prefixed with a DSID is then delegated to a naming authority service, which is equipped with a *special functionality*: Instead of looking up the corresponding database entry for each individual PID, the incoming PID is used to extend the registered current offered *Access Request Syntax*, which is associated with that DSID.
This approach would eliminate the whole overhead, that occurs at the registration process of each individual research dataset at a particular naming authority service. As a consequence, the concept of *Access Request Syntax Registration* would be suitable for persistently identifying huge amounts of research datasets.

IV. Implementation

In this section, we will discuss the possibilities of implementing the concept of *Access Request Syntax Registration* in the Handle System.

A. Handle System

Among the current existing PID systems, the Handle System is the most elaborated and established one. It stands out because of its compactness and widespread use. However, despite its sophisticated conception, the Handle System also weakens in the aspect of high-performance persistent identification for a vast amount of research datasets. This is confirmed by the measurements in [15]. In their setup, the average response time for a single registration request was around 40 milliseconds, which is fairy slow for a huge amount of research datasets. In addition, this is also confirmed by the fact that the Handle Protocol [17] do not offer a native batch operation. The current offered batch operation provides only a session for sequentially registering a large group of research datasets. Otherwise, each registration request has to be individually authenticated before processing with the actual registration process.
In order to enable the Handle System to meet the high-performance requirements, in this section we will consider the implementation of the concept of *Access Request Syntax Registration* into the Handle System. Therefore, in the following we will provide a brief insight into the components of the Handle System:

Handle System Namespace: An identifier assigned in the Handle System to an individual research dataset is called a *Handle*. A *Handle* consists of two parts: The prefix, used to uniquely identify its naming authority, followed by a unique local name under the naming authority, called the suffix. Both parts are separated by the "/" character: PREFIX / SUFFIX. The prefix is assigned and controlled by the CNRI[23], which assures that each naming authority is uniquely allotted a

prefix. In addition, the prefix registration enables the naming authority service to be integrated into the global resolution infrastructure.

Handle Data Model: Each *Handle* is associated with a Handle Record. A Handle Record in turn, is composed of one or many Handle Values. Each Handle-Value consists of a "<type>" field, used to specify the syntax and semantics of the corresponding data in its "<data>" field. In principle, a Handle-Value could have any arbitrary type, however, the Handle System also pre-defines a set of types required to carry out the global resolution system.

Handle System Protocol: The Handle Protocol defines a complete set of operations for administering of Handle Records in any registered naming authority service. It also enables a distributed server infrastructure for a high-available resolvability of the *Handles*. For detailed insight, we refer to the Handle System specification [17].

Handle System Resolution Infrastructure: As shown in Figure 3, the global resolution infrastructure in principle consists of three core entities:

a) The Global Handle Record (GHR): The GHR is under the control of CNRI and contains all the registered prefix Handle Records. A prefix Handle Record for instance, primarily contains a list of IP addresses corresponding to the servers forming the Local Handle Service, which is responsible for Handles under a particular prefix.
b) Local Handle Services (LHS): A Local Handle Service is composed of at least one primary server and none or many mirror servers. In principle, a single LHS holds all the Handle Records of its assigned prefixes. Each of the servers in the LHS implements the Handle Protocol in order to be integrable into the global resolution infrastructure.
c) The Global Proxy Resolver: The Global Proxy Resolver (hdl.handle.net), is the central system for resolving the *Handles*. As can be seen in Figure 3, the resolution process of an individual *Handle* covers six steps and results in the revealing of the current Internet address of the corresponding research dataset. In addition, this Figure also shows a brief description of each step of the resolution process. The resolution request is followed by an access request (steps 7 and 8) to the research repository storing that particular research dataset.

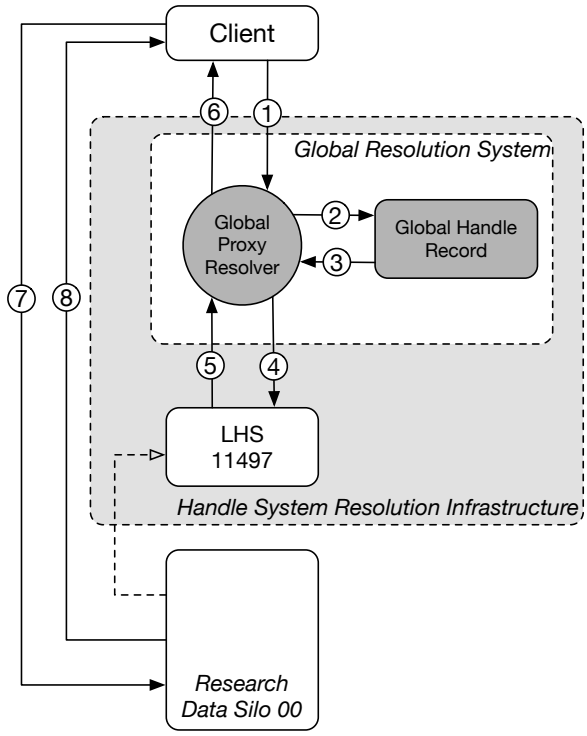

Fig. 3. The resolution process in the Handle Resolution Infrastructure.

B. Performance Problem caused by Handle System

In the previous section we have showed the fundamental performance problem caused by current PID systems to be originated in the respective protocols and/or data models, which they require to implement. In the particular case of the Handle System this means that a research data repository has to implement the Handle Protocol in order to be integrable into the global Handle Resolution Infrastructure. Usually, instead of implementing the Handle Protocol natively into the research data repository, each incoming research dataset has to be additionally registered at a particular LHS. This LHS could either be installed as an additional component into the research data repository architecture or it could be hosted by a specific PID service provider. In any case, this means that each research dataset, which is already assigned an internal identifier, additionally is assigned a public identifier, called a *Handle*. However, this additional identifier assignment often causes a significant performance loss at the ingest workflow into the corresponding research data repository. Due to the network latency, the performance at the ingest workflow will be even more critical in the case of a LHS hosted by a PID service provider.

The internal identifier is used within the research data repository to manage the corresponding research dataset. In contrast to that, the assigned *Handle* is in principle used to provide a sustainable public access to the corresponding research dataset. To eliminate the overhead caused by this additional identifier assignment, in the previous section, we have therefore proposed the concept of *Access Request Syntax Registration*. In the case of the Handle System, this essentially means that an individual research dataset's internal identifier, prefixed with a specific DSID, becomes the public *Handle* without being registered at a particular LHS. It should be noted, that it is not

necessary to prefix the internal identifiers within the research data repository.
In the following, we will therefore analyze the possibilities in the Handle System for implementing the concept of *Access Request Syntax Registration*.

C. Implementing into Handle System

The basic functionality for implementing the concept of *Access Request Syntax Registration* is to extend the registered current offered access syntax of the individual research data repository by the incoming PID string. To perform such a request extension, in the case of the Handle System, there are in principle the following two solutions:

Solution A: Setting up a special LHS, which is responsible for all the DSIDs of the registered research data repositories. Hence, each resolution request is delegated in the usual manner to that special LHS. In contrast to an ordinary LHS, this special LHS dynamically extends the registered current offered access request syntax by the incoming *Handle* identifier. The individual access request syntax could be specified by the pre-defined "HS_NAMESPACE" typed Handle-Value in the corresponding prefix Handle Record. Originally, the pre-defined "HS_NAMESPACE" type is intended for identifying subparts of a single research dataset. A "HS_NAMESPACE" typed Handle-Value allows the definition extension rules, which can be applied onto a base *Handle*. An example would be a large video clip, which is assigned a base *Handle*. Possible parameters added to the base *Handle* at the resolution request could then be used to extend the corresponding Handle Record for retrieving subparts of that video clip.
In the usual resolution processing, each LHS tries to lookup the corresponding database entry for a requested PID. In the case, when there is no corresponding entry found, the LHS tries to lookup a "HS_NAMESPACE" definition, which can be used to generate an appropriate response.
In principle, the "HS_NAMESPACE"-typed Handle-Value, could also be used to specify the current offered access request syntax of the corresponding research data repository. However, the fundamental problem of this solution is that there are additional commitments and strategies necessary for ensuring a reliable and sustainable operation of this special LHS.

Solution B: Instead of making use of the "HS_NAMESPACE" type, another approach would be as follows: Extend the pref-defined set of data types by a special data type for specifying the individual currently offered access request syntax, which is then directly processed in the global resolution system without delegating to a special LHS.
In contrast to **solution A**, since there is no need for a special LHS, the sustainability and reliability of this solution is directly covered by the global resolution system. Another important result of this solution is that the resolution performance could also be essentially improved. Since the resolution is directly processed in the global resolution system, the resolution process would not contain the steps 4 and 5 in Figure 3. Instead, the resolution process would be reduced from six to four steps, which would therefore also improve the performance of the resolution process.
However, this solution requires a minor modification of the global resolution system and the corresponding Handle Protocol specification.

Because of the promising benefits of solution **solution B**, in the following we will provide insight into the essential aspects of the corresponding implementation.

TABLE I
EXAMPLE HANDLE-VALUE CONTAINING THE CURRENT ACCESS REQUEST SYNTAX DEFINITION

<index>:	1
<type>:	HS_RDS_URL
<data>:	http://rdsilo-00.net/datasets/#{DSID}/#{suffix}
<TTL>:	{Relative: 24 hours}
<permissions>:	PUBLIC_READ, ADMIN_WRITE
<timestamp>:	927314334000
<reference>:	{empty}

D. Implementing the Special Data Type

The implementation of **solution B** essentially covers the following:

Access Syntax Data Type: For the specification of the individual currently offered access request syntax we propose to extend the pre-defined set of data types by the "HS_RDS_URL" type. The "HS" part indicates that this data type belongs to the pre-defined set of types for carrying out the Handle Resolution System. This is followed by the "RDS_URL" part, whereas "RDS" is an abbreviation for research data silo. The data field of a Handle-Value with such a type, in principle specifies the current offered access request syntax of the corresponding research data silo. Since recent recent research data silos usually offer a HTTP interface, the access request syntax would be an URL containing placeholders for inserting the incoming *Handle* identifier. The resulting Handle-Value in turn, has to be added to the corresponding prefix Handle Record at the GHR.
An example of such a Handle-Value can be seen in Table I.

Implementing into core Handle Software Package: All the components in the Handle Resolution Infrastructure contain a common core software package [18], which is a reference implementation of the Handle Protocol. Therefore, in order to enable the global resolution system to process the "HS_RDS_URL" type, an appropriate algorithm has to be implemented into this core software package.
The core function of that algorithm is to extend the individual access request syntax with the incoming *Handle* identifier. This extension has to be performed only when there is found a "HS_RDS_URL" typed Handle-Value at step 3 of resolution process (see Figure 3). In that case, the extended access request syntax has to be sent back to the requesting client without involving the steps 4 and 5.

For the case, when there is no "HS_RDS_URL" typed Handle-Value found, the usual resolution process is performed instead.

E. Summary

In this section, we have provided two different solutions for implementing the concept of *Access Request Syntax Registration*. The first solution involves the setup of a responsible special LHS and making use of the pref-defined "HS_NAMESPACE" data type. This data type is actually intended for identifying subsets of a large research dataset, which is assigned a base *Handle*. This base *Handle* is then extended by rules specified in the "HS_NAMESPACE" typed Handle-Value. However, the fundamental problem of this solution is the requirement for additional commitments and strategies for providing a reliable and sustainable operation of this special LHS.
In contrast to that, the second solution involves in the extension of the pref-defined set of data types and in the implementation of an appropriate algorithm into the core software package of the Handle System.
Since the overhead of an additional (public) identifier assignment is eliminated by both solutions, the primary goal of a high-performance identification concept is provided by both solutions. However, since the second solution could additionally lead to an improvement of the resolution performance, this solution is even better for providing high-performance.

V. Evaluation

In this section, we will provide an experimental evaluation of the solutions provided in the previous section. For the evaluation, we have implemented the functionality described in section IV into the publicly available Handle System software package. The core parts of our implementation can be viewed in [24]. It should be noted, that this only contains the Java class files of the Handle System software package, where a modification was required.
With the resulting software package we then developed an imitation of the Global Proxy Resolver.
The productive Global Proxy Resolver is composed of a distribution of four servers, whereas one of them is hosted at GWDG. In total, there are up to 40 million resolution requests per month. Due to this huge amount of resolution requests, caching at the Global Proxy Resolver is only enabled for the prefix Handle Records. The caching of prefix Handle Records provides an accelerated determination of the responsible LHS to which the resolution request of the particular *Handle* is delegated to. The caching of individual Handle Records in turn, is enabled at the responsible LHS for providing a fast resolution process. Therefore, for our evaluation we have implemented the same caching policy as in the productive setup.
For each measurement process 1000 consecutive resolution requests for different *Handles* under the same prefix were issued against our Global Proxy Resolver imitation. For each request the elapsed time in milliseconds was recorded.
For the evaluation we established two different setups, whereas each setup was composed of three components:

a) Global Proxy Resolver Imitation: As mentioned above, this component was based on our implementation [24] of the concept of *Access Request Syntax Registration* into the Handle System software package.
b) Global Handle Record (GHR): The productive GHR was used for storing the appropriate "HS_RDS_URL" and "HS_NAMESPACE" typed Handle-Values into a prefix Handle Record, which is owned by GWDG.
c) Local Handle Service (LHS): This was a LHS responsible for the specific prefix owned by GWDG.

In the first setup, the LHS was installed on a different host than our Global Proxy Resolver imitation. In this setup, for each solution a dedicated measuring process was performed. In contrast to that, in the second setup, the LHS was installed on the same host as our Global Proxy Resolver imitation. In this setup, only for **solution A** the measuring process was launched. Although, the first setup is suitable for reflecting the real world situation, the second setup was needed to examine the effect of network latency at the resolution process.
Figure 4 depicts the elapsed time for each resolution request of the three measurements processes. Hence, these measurements allow the following interpretation:

- In the first setup, the resolution process for **solution A** (blue points with "$\star$" marker) was always significantly slower than for **solution B**.
- The longer resolution time in **solution A** was caused by the overhead of the delegation process to the responsible LHS.
- Due to caching at the LHS, this delegation overhead was mostly caused by the network latency between the Global Proxy Resolver and the LHS.
- This is confirmed by the measurements from the second setup (red points with "▼" marker) for **solution A**. Because of the reduced network latency, the resolution process for **solution A** is only slightly slower than for **solution B**.
- The average resolution process time for **solution A** in the first setup is $\overline{A}\_1 \approx 36ms$, whereas in the second setup it is $\overline{A}\_2 \approx 21ms$.
- In contrast to that, the average resolution time for **solution B** is $\overline{B} \approx 18ms$.

VI. Conclusion and Future Work

In this work, we have revealed that the performance problems caused by current existing and established Persistent Identifier systems is originated in the protocols and/or data models, which their require to implement. The implementation of these protocol and/or data models is necessary to be integrable into the particular global resolution system. Hence, research data repository designers often are compelled to either install an additional naming component, which implements a specific required protocol and/or data model or to make use of a PID service provider. In any case, for repositories

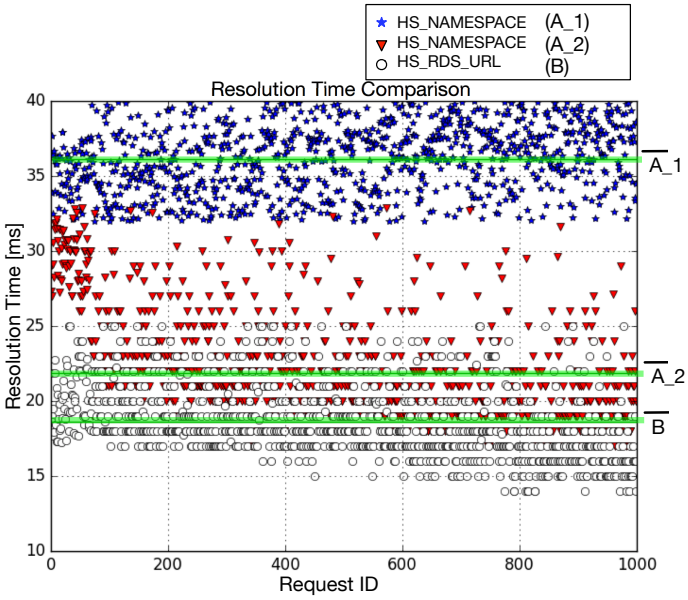

Fig. 4. Measurements of the resolution process for both solutions

subjected to an immense research dataset growth the additional identifier assignment causes a considerable performance loss at the ingest workflow. Due to the network latency, this is even more critical for the case of making use of a PID service provider.

Since recent research data repositories often have a built-in naming component, which already assigns immutable and internally unique identifiers, we have proposed to enable these internal identifiers to be globally resolvable. This could be realized by first prefixing the internal identifiers with a prefix, which is dedicated exclusively to the corresponding research data repository and secondly, by implementing an appropriate functionality for the resolution process.

This functionality is based on the fact that, the PID-records corresponding to a single research data repository in principle do contain the same information entity: The current offered access request for retrieving research datasets from the individual research data repository. Hence, instead of registering each research dataset at a naming authority service, we have proposed to register the current offered *Access Request Syntax* once for each individual research data repository. This *Access Request Syntax* can be used to dynamically generate the appropriate response for all resolution requests, which are targeted for research datasets in an individual research data repository.

Ultimately, this concept enables a high-performance persistent identification of the vast amount of research datasets in an individual research data repository.

Furthermore, we have investigated the implementation of this concept into the established Handle System. We have provided two solutions for the implementation of the concept of *Access Request Syntax Registration* into the Handle System. By both solutions, the overhead of additional identifier assignment is eliminated. However, the second solution also enables an improvement of the resolution process. For the first solution an additional special naming authority service is required for holding the individual interface specifications of the registered research data repositories. However, for a sustainable and reliable operation of such a special naming authority service there are additional commitments and strategies necessary, which is not the case fo the second solution. The disadvantage of the second solution is the necessity of minor changes in the Handle Protocol specification and of an appropriate implementation into the global resolver.

Moreover, in an experimental evaluation we have compared the performance of the resolution process of the two proposed solutions. This evaluation confirmed that the second solution also leads to an improved resolution process.

Since currently the Web is the primary platform for the scientific exchange, our solutions are focused on the specification of HTTP interfaces. Therefore, further research is required for the specification of more general interfaces, which are directly based on UDP or TCP.

Another functionality, which PID systems could provide, is a federated search engine on top of all the registered research data repositories. Hence, search queries submitted to PID systems could be delegated to these research data repositories. However, further research is necessary, for developing appropriate filtering and mapping concepts in order to enable a fast federated search and result presentation.

REFERENCES

[1] S. Sun, L. Lannom, and B. Boesch, "Handle system overview," Tech. Rep., 2003.

[2] N. Paskin, "Digital object identifier (doi) system," *Encyclopedia of library and information sciences*, vol. 3, pp. 1586–1592, 2008.

[3] "European persistent identifier consortium," http://www.pidconsortium.eu, accessed: 2016-04-18.

[4] "The ark persistent identifier scheme," https://tools.ietf.org/html/draft-kunze-ark-14, accessed: 2016-04-18.

[5] "URN Syntax," RFC, 1997. [Online]. Available: https://tools.ietf.org/html/rfc2141

[6] H.-W. Hilse and J. Kothe, *Implementing persistent identifiers*. Consortium of European Reasearch Libraries, 2006.

[7] R. Kevin Richards, *A beginners guide to persistent identifiers*. Gbif. [Online]. Available: https://books.google.de/books?id=Q5TkjgRll9gC

[8] J. Hakala *et al.*, "Persistent identifiers: an overview," *KIM Technology Watch Report*, 2010.

[9] E. Tonkin, "Persistent identifiers: considering the options," *Ariadne*, no. 56, p. 8, 2008.

[10] E. Bellini, C. Cirinnà, M. Lunghi, E. Damiani, and C. Fugazza, "Persistent identifiers distributed system for cultural heritage digital objects," *iPRES 2008*, p. 242, 2008.

[11] E. Bellini, C. Luddi, C. Cirinnà, M. Lunghi, A. Felicetti, B. Bazzanella, and P. Bouquet, "Interoperability knowledge base for persistent identifiers interoperability framework," in *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*. IEEE, 2012, pp. 868–875.

[12] T. Weigel, S. Kindermann, and M. Lautenschlager, "Actionable persistent identifier collections," *Data Science Journal*, vol. 12, no. 0, pp. 191–206, 2014.

[13] A. E. Evrard, C. Erdmann, J. Holmquist, J. Damon, and D. Dietrich, "Persistent, global identity for scientists via orcid," *arXiv preprint arXiv:1502.06274*, 2015.

[14] A. Karakannas and Z. Zhao, "Information centric networking for delivering big data with persistent identifiers," 2014.

[15] O. Schmitt, T. A. Majchrzak, and S. Bingert, “Experimental realization of a persistent identifier infrastructure stack for named data networking,” in *Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 33–38.
[16] R. Kahn and R. Wilensky, “A framework for distributed digital object services,” *Int. J. Digit. Libr.*, vol. 6, no. 2, pp. 115–123, Apr. 2006. [Online]. Available: http://dx.doi.org/10.1007/s00799-005-0128-x
[17] “Handle System Protocol Specification,” RFC, 2003. [Online]. Available: https://www.ietf.org/rfc/rfc3652.txt
[18] “Handle software package,” http://www.handle.net/download_hnr.html, accessed: 2016-04-18.
[19] “Doi handbook data model,” https://www.doi.org/doi_handbook/4_Data_Model.html, accessed: 2016-04-18.
[20] “Thump protocol specification,” https://tools.ietf.org/html/draft-kunze-thump-02, accessed: 2016-04-18.
[21] “Ezid,” http://ezid.cdlib.org, accessed: 2016-04-18.
[22] J. Martin, *Managing the Data Base Environment*, ser. A James Martin book. Pearson Education, Limited, 1983. [Online]. Available: https://books.google.de/books?id=nMgmAAAAMAAJ
[23] “Corporation for national research initiatives,” http://www.cnri.reston.va.us, accessed: 2016-04-18.
[24] “Implementation of our concept,” http://hdl.handle.net/11022/0000-0000-9B7C-7, accessed: 2016-04-18.