Smart Policy Driven Data Management and Data Federations

Patrick Fuhrmann[*]
DESY
E-mail: patrick.fuhrmann@desy.de

Marica Antonacci
INFN/Bari
E-mail: marica.antonacci@ba.infn.it

Giacinto Donvito
INFN/Bari
E-mail: giacinto.donvito@ba.infn.it

Oliver Keeble
CERN
E-mail: oliver.keeble@cern.ch

Paul Millar
DESY
E-mail: paul.millar@desy.de

The core activity within the newly created “eXtreme DataCloud” project will be the policy-driven orchestration of federated data management for data intensive sciences like High Energy Physics, Astronomy, Photon and Life Science. Well-known experts in this field will work on combining already established data management and orchestration tools to provide a highly scalable solution supporting the entire European Scientific Landscape. The work will cover “Data Life Cycle Management” as well as smart data placement on meta data, including storage availability, network bandwidth and data access patterns. Mechanisms will be put in place to trigger computational resources based on data ingestion and data movements. This paper presents the first architecture of this endeavour.

International Symposium on Grids and Clouds (ISGC) 2018 in conjunction with Frontiers in Computational Drug Discovery
16-23 March 2018
Academia Sinica, Taipei, Taiwan

[*]Speaker.

1. Introduction

In November 2017, the "eXtreme DataCloud" (XDC) project was launched within the framework of the EU Horizon 2020 framework, targeting the development of scalable technologies for federating storage resources and the management of data in highly distributed computing environments. The project will last for 27 months and combines the expertise of 8 large European organizations. The targeted platforms are the current and next generation e-Infrastructures deployed in Europe, such as the European Open Science Cloud (EOSC)[20], EGI, the Worldwide LHC Computing Grid (WLCG) and the computing infrastructures that will be funded by the upcoming calls.

The solutions provided by XDC will be based on already well established data management components as there are dCache[11], EOS[13], FTS[14] and the INDIGO Orchestrator[18], Onedata[21] and many more. The targeted scientific communities, represented within the project itself, are from a variety of domains, like astronomy (CTA)[24], Photon Science (European X-FEL)[9], High Energy Physics (LHC)[25], Life Science (LifeWatch)[26] and others.

The work will cover "Data Life Cycle Management" and smart data placement on meta data, including storage availability, network bandwidth and data access patterns. The inevitable problem of network latency is planned to be tackled by smart caching mechanisms or, if time allows, using deep learning algorithms to avoid data transfers in the first place. Furthermore, data ingestion and data movement events, reported to the centralized INDIGO orchestration engine can trigger automated compute processes, starting from meta-data extraction tools to sophisticated work flows, e.g. to pre-analyze images from Photon Science or Astronomy detectors.

2. Target communities and high level objectives

According to the project's work program[1], the high-level objective of this work package is the orchestrated placement of scientific data in the area of Exabytes on the site (IaaS) as well as on the federated storage level. In the context of this work, placement may either refer to the media the data is stored on, to guarantee a requested Quality of Service, or the geographical location, to move data as close to the compute facility as possible to overcome latency issues in geographically distributed infrastructures. In the latter case, data might either be permanently moved, or temporarily cached. It is envisioned that data migration can either be enforced by static policies at the local or at the federated level or as a result of pattern recognition technologies, including utilizing information on metadata or short term and historical access patterns. Long term policy driven data movements are referred to as"Data Life Cycle Management". Data Life Cycle may not only concern data location and data storage quality control but similarly policy driven changes of low level meta data, like "Access Control Lists" to make data publicly available after a predefined "grace period". Finally, this work package will enable the orchestration of compute activities based on events produced as a result of data ingestion and on automatic or condition driven data movements.

In order to work towards realistic scenarios, XDC is linked to existing high throughput scientific infrastructures. In the case of this particular work package, this is primaray the Europen XFEL[9] and the LHC Computing Grid (WLCG[8]. However, in anticipation of upcoming new challanges, e.g. the Square Kilometer Array, SKA[7], those scientific demonstrators should be seen as role models for new endeavours in the Exabyte data range. Furthermore, as the XDC funding period is limited to 27 months, it is quite evident that new functional improvements provided by XDC can only be implemented within the framework of already existing software stacks. As WLCG is undoubtedly the most advanced distributed computing infrastructure in Europe, it is not surprising that this particular work within XDC, to a large extend, is based on a selection of WLCG software components. See section 3 ”Available components” for details.

2.1 Gap analysis and advanced features

E-Infrastructures or experiment frameworks have to combine a variety of distributed resources to successfully manage their work flows. Those resources can be the actual data sources, possible HPC cluster systems for fast analysis, large low latency storage systems and long term archiving endpoints. Currently those entities are provided by different resource provides with their proprietary interfaces. The orchestration of those resources is performed within each individual experiment framework as indicated in Figure 1. Although this approach is expensive, it is generally not an issue for large scientific infrastructures like WLCG or others. However, smaller groups might not be able to afford such a professional setup. The alternative XDC is envisioning, is to interface the resources at the European infrastructure level and make them available, including high level services, like data transfer and data life cycle orchestration, through standard interfaces to the scientific communities. This not only significantly reduces the software stack of the communities but also simplifies the work of the storage providers as they only have to handle a single proxy framework, being shared by all their clients. (See Figure 2)

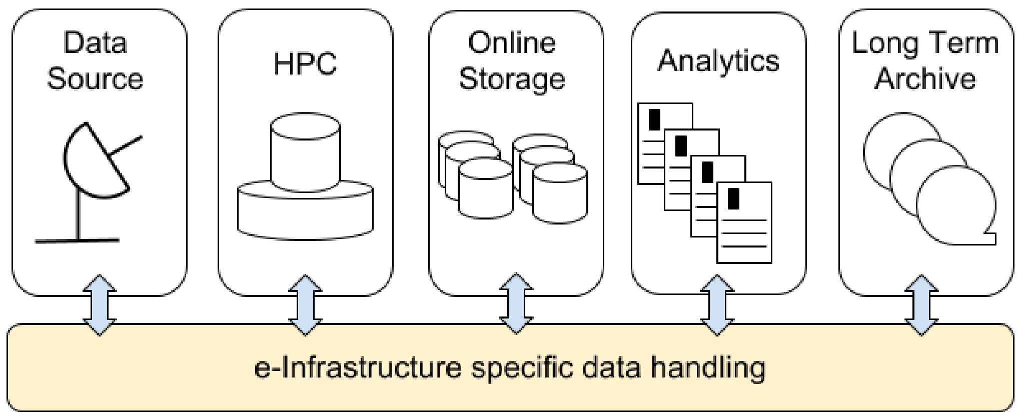

Figure 1: Current infrastructure management

Moving from this high level goal to the actual technical implementation, we found three major areas we would have to work on:

- **Quality of Service** in storage: When following the life cycle of data, we found that, depending on the processing step performed, different capabilities of the data storage space are required. They range from high throughput, low latency storage to long term archive. Consequently, if the data life cycle process needs to be abstracted, those capabilities have to be defined with a common vocabulary and storage endpoints need to be able to understand and acknowledge storage requests with those capabilities.

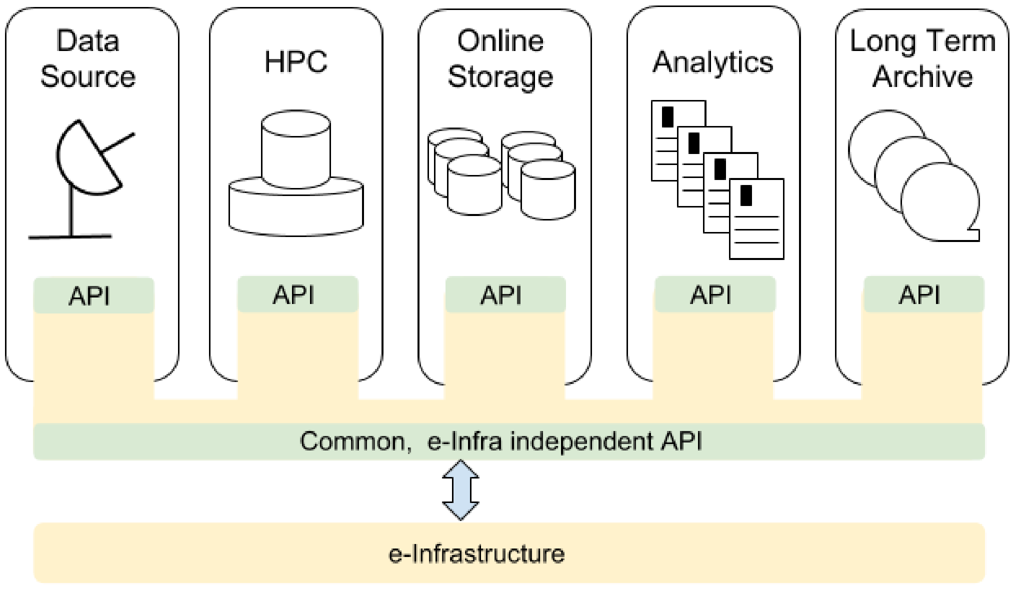

Figure 2: Current infrastructure management

- **Caching** Distributed infrastructures require additional precautions by the remote clients or services, as data links can be slow, or interrupted and suffer from inevitable latencies. One way of coping with those drawbacks is to provide caching, close to the data sinks.
- **Orchestration** In order to allow an arbitrary work flow to be applied to data, certain interfaces have to be implemented. In particular when data is ingested into a global system, the system must be informed and based on a common language, actions, like preprocessing or data replication may have to be performed. Moreover, interfaces to the compute world are required to started processes or functions on grids or clouds on events happing within the system infrastructure.

After the next section, describing the components we have chosen for XDC, we will outline the architecture we are envisioning to implement the three areas described above.

3. Available components

As described initially, neither the time nor our funding capacity of XDC will allow us to implement a system from scratch, providing all features we listed in the sections above. However, almost all components we need for our endeavour are already available and most importantly are in production for more than a decade. This chapter is briefly describing those components as they are needed to understand subsequent chapters on the XDC architecture.

3.1 The file transfers service, FTS

FTS[14] is a data movement service developed at CERN that reliably copies data from one Storage URL to another. It is based on 3rd party copy to achieve this and implements mechanisms to ensure reliability of the copies and an internal scheduler to ensure an optimal usage of the network channels and bandwidth maintaining a high availability of the service. It is a mature product extensively used in production by many Virtual Organizations, including the LHC experiments as a fundamental component of their data management system to move data from one Grid site to another.

3.2 Storage Federations Technologies, Dynafed

The Dynamic Federations system, Dynafed[16] provides a fast dynamic name space that it exposes via HTTP and WebDAV. This is built on-the-fly by merging and caching (in memory) meta data items taken from a number of (remote) endpoints. Dynafed is developed to be performant, scalable and resilient to endpoints that become unavailable. HTTP and WebDAV clients can browse the Dynamic Federation as if it were a unique partially cached name space. It will redirect clients to the best available host when they ask for a file replica. Dynafed also supports writing. Distributed data are presented as a unified repository. Dynafed natively supports HTTP, WebDAV, S3[27] and MS Azure[28] endpoints. The preferred data management choice for Dynafed is to work with algorithmic, prefix-based file path translations across sites. This gives the best flexibility to the data management model, together with very quick user interaction and high performance. If needed, Dynafed can also contact an external service of file name/path translation.

3.3 The storage backends: dCache, EOS and StoRM

dCache[11] is a Data Storage and Management system developed by dCache.org (DESY, Fermilab, NDGF). dCache is in production at more than 60 sites around the world, including 8 WLCG Tier 1 centers and for over a decade it is managing more than 150 PBytes in total. Some installations are spanning cities and even countries. dCache already provides basic mechanisms for QoS, data orchestration and remote caching. EOS[13] : Data Storage and Management system developed by the CERN IT Department, scalable to many tens of petabytes and supporting geographically distributed deployment. EOS is managing the data of the distributed WLCG Tier 0 center (CERN and Wigner in Budapest). StoRM[17] : StoRM is a Storage Resource Manager that relies on a parallel file system or a standard Posix file system backend. StoRM provides an SRM interface to GPFS and other parallel filesystem technologies, which can be seen as a prerequisite to the XDC QoS in storage approach.

3.4 INDIGO Orchestrator and TOSCA, the system wide orchestration engine

The INDIGO PaaS Orchestrator[18] is a component of the PaaS layer that allows to instantiate resources on Cloud Management Frameworks (like OpenStack and OpenNebula) and Mesos clusters. It takes the deployment requests, expressed through templates written in TOSCA YAML Profile 1.0 [29] and deploys them on the best cloud site available. In order to do that it gathers SLAs, monitoring info and other data from other platform services, it asks to the cloud provider ranker for a list of the best cloud sites.

3.5 Rucio, the data management orchestration subsystem

The Rucio project[19] is the new version of ATLAS Distributed Data Management (DDM)[15] system services, allowing the ATLAS collaboration to manage large volumes of data, both taken by the detector as well as generated or derived, in the ATLAS distributed computing system. Rucio manages accounts, files, datasets and distributed storage systems.

3.6 The INDIGO CDMI reference implementation for QoS

The INDIGO CDMI Reference Implementation[3] is the INDIGO solution to implement CDMI (SNIA Cloud Data Management Interface) specifications aimed at improving Quality of Service capabilities of distributed storage resources. This tool will underpin the implemented solution for an effective policy driven data management system. It provided a flexible plug-in mechanism to call out to a variety of storage solutions.

4. The envisioned overall architecture

As described in Section 2 the main focus of our work is the extension of the current infrastructure in three areas. The application of services qualities in storage, smart caching mechanisms and the overall storage orchestration in terms of location and data transfer and for the implementation of data life cycle policies. It is clear that those three areas are deeply interlinked and influence each other in the overall architecture, which is still in discussion within the project. The final version of the architecture will be published with deliverables XDC-D4.1 and XDC-D1.6[30]. However, in order to provide at least some insight into our work, we describe those areas independently.

4.1 Quality of Service in storage

The service used to store scientific data is important for scientists responsible for the data management of their communities: it must satisfies the demands of the scientists, both in terms of performance and reliability. Research communities often operate on a fixed budget; therefore there is a natural desire to provide the necessary storage without unnecessary expense. To achieve this, the scientists and storage providers must have a common understanding of what is required and what the different storage services can offer.

The idea to provide scientific communities or individuals with the ability to specify a particular quality of service when storing data, e.g. the maximum access latency or minimum retention policy, was introduced within the INDIGO-DataCloud[2] [12] project. In INDIGO, QoS in storage was split into 3 interlinked activities:

- The definition of a common vocabulary within the framework of the Research Data Alliance, RDA[5] when describing storage qualities
- The rendering of those attributes by a standard network protocol or API
- A reference implementation of the selected standard, covering different storage back ends.

In INDIGO, the network protocol and the implementation was based on an extension of the SNIA Cloud Data Management Protocol, CDMI[4]. Within XDC, the QoS concept is envisioned to consistently compliment all data related activities. In other words, whenever storage space is requested, either manually by a user or programmatically by a framework, the quality of that space can be negotiated between the requesting entity and the storage provider.

4.1.1 QoS definitions with the Research Data Alliance

Storage providers, such as research institutes and universities acquire storage through some procurement process. This procurement process may involve buying physical hardware, but increasingly this involves some company provisioning cloud resources tailored to the specific needs. In either case, the success of this process depends on the storage provider and the company having a common understanding of what is required.

While this can be achieved through direct meetings and discussions, the process is greatly simplified if there is a shared, common vocabulary that allows a description of what a storage service should provide.

In order to achieve a minimum of commonality between the different stakeholders within this ecosystem, the XDC project is following up on an initiative, started with the INDIGO-DataCloud project, to define a basic storage quality vocabulary within the framework of the Research Data Allicance, RDA. XDC is in the process of setting up a working group called "WG Storage Service Definitions" [6], developing a common vocabulary that allows storage services to be described. This language will be machine readable and will be sufficient to describe both in-house storage services in addition to various cloud storage offerings.

4.1.2 The Cloud Data Management Interface, CDMI

To communicate the vocabulary of the different QoS's defined with RDA over the wire between clients and storage services, the Cloud Data Management Interface, CDMI[4] has been chosen. Although it only provides very limited functionality to express the storage capabilites and processes we developed, it is the only known standard in this area and it allows to define extensions to the basic protocol definitions. INIDGO-DataCloud as well as XDC are partnering with the Storage Network Industry Association, SNIA[10], in charge of guarding the protocols and its extensions. The extensions suggested by INDIGO-DataCloud and XDC have been accepted and have been implemented in the SNIA CDMI reference implementation (see next section). However, as the XDC requirements are even more challenging we envision to continue the process of enhancing the CDMI extensions according to our needs.

4.1.3 The CDMI Reference Implementation and its plug-ins

To allow organizations implementing the CDMI protocol to test their software stack against a well defined reference, SNIA is providing a protocol java skeleton in GitHub. With the permission of SNIA, INDIGO-DataCloud created a second branch of this reference implementation using the java Sevice Provider Interface, SPI, simplifying the provisioning of plug-ins for a variety of different storage back ends. Figure 3 sketches the basic idea of that approach. While the common way of communicating QoS for XDC products will be CDMI, different storage system endpoints generally provide this functionality through proprietary protocols or API's. EOS and dCache, for instance, expose QoS through a RESTful interface. While dCache already offers a QoS plug-in for the SNIA reference implementation, due to its involvement in INDIGO-DataCloud, EOS will have to implement one for it's own interface. Another alternative would be to unify the dCache and EOS RESTful interfaces and only provide one plug-in for the CDMI reference implementation.

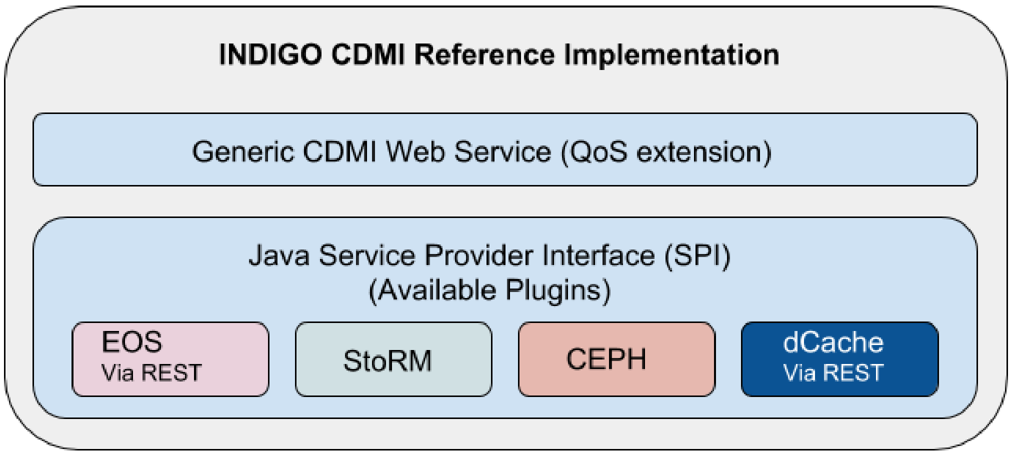
INDIGO CDMI Reference Implementation
Generic CDMI Web Service (QoS extension)
Java Service Provider Interface (SPI)
(Available Plugins)
EOS
Via REST
StoRM
CEPH
dCache
Via REST

Figure 3: The SNIA-INDIGO-DataCloud CDMI reference implementation architecture.

As almost all WLCG client applications communicate with storage services through the common GFAL library, GFAL will be extended to support the CDMI client functionality. As a consequence the File Transfer Service, FTS would automatically be enabled to define QoS for data transferred to remote storage services. Figure 4 illustrates the relationship between the different components, from the Orchestration to the final QoS enabled storage services. The architectures should be seen as a tool box. Not all components need to present, only those needed to achieve the requested features.

4.2 Data Flow Orchestration

Orchestration of data flows within storage management systems is already available in various products, like iRODS[22], Onedata[21], IBM GPFS[23] and others. However, those systems almost always only interact with data within their systems themselfves. In contrast, the objective of the *XDC Data Flow Orchestration* is to interface to a variaty of storage systems and storage management components, already installed in data intensive e-Infrastructures and to use a well defined language, e.g. TOSCA[29] to configure those data flows. In particular, orchestration of data in XDC is covering a variety of aspects. They range from enforcing QoS on the global level, as described in the chapter before, granting access to private data after a well define Grace Period. Grace Periods are granted by funding agencies to permit keeping scientific data confidential for a well defined interval, allowing the scientific communities to prepare their findings or finalize their publications, before the data has to become public. Ideally the XDC orchestration system is supposed to apply the Data Management Plan of a e-Infrastructure or a scientific experiment to the data generated. To illustrate the potential of this functionality we are presenting a typical use case, the *data preprocessing on data ingestion*, as shown in Figure 5.

In this use case, briefly sketched in Figure 5, a user submits a work-flow request to the INDIGO Orchestrator, instructing it to watch a particular part of an attached storage space. The request defines that data, newly ingested into the observed storage space should be

- preprocessed by a user-specified application or algorithm and

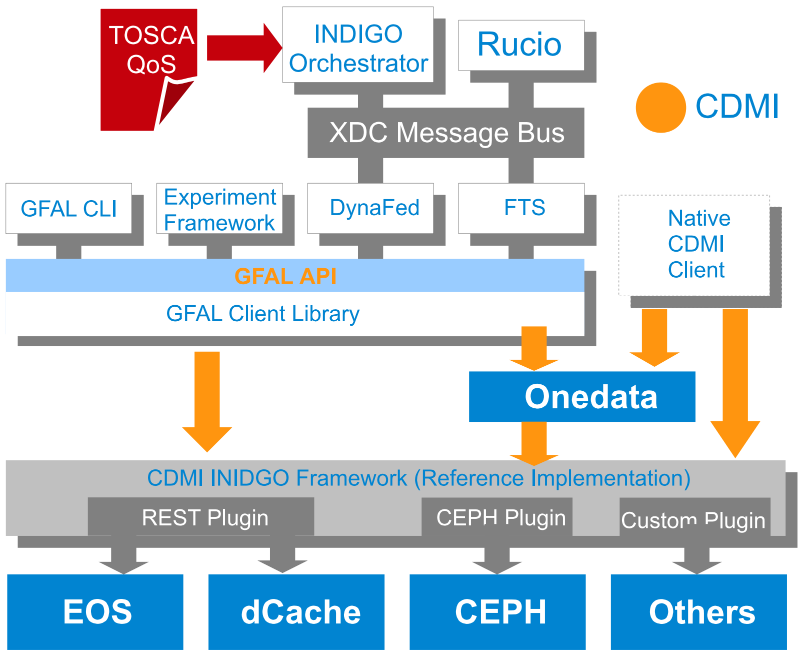

Figure 4: Relationship of components composing "Orchestrating Quality of Service" in storage. At the time of this writing, the component denoted as *XDC Message Bus* should be regarded as a place holder for any kind of possible communication between the INDIGO Orchestrator, Rucio and FTS or DynaFed. As described in figure 6 we might even decide to utilize the Rucio internal bus and expose it to other XDC components.

- subsequently replicated to other storage spaces.

The preprocessing algorithm could be a data quality check, a data skimming process, metadata extraction, indexing, etc. The replication might just ensure data availability or safety or might be the preparation for further analysis steps. XDC components involved, are the INDIGO-Orchestrator and Rucio, performing the orchestration, FTS for performing the data transfers as well as the different storage systems for the distributed storage spaces. Please refer to section "Available Components" for details. Figure 5 depicts the steps performed on each ingestion of data into the system.

1. The user submits to the Orchestrator including the following information:
   (a) the space to watch for incoming data
   (b) the application to be run on incoming data
   (c) the replication rule to be enforced on the incoming or preprocessed data.
2. The storage system, holding the watched storage, notifies the presence of new data by sending a message to the XDC message queue.
3. The INDIGO Orchestrator receives the notification and registers the ingested data file into Rucio, including the replica policy, specified by the user.

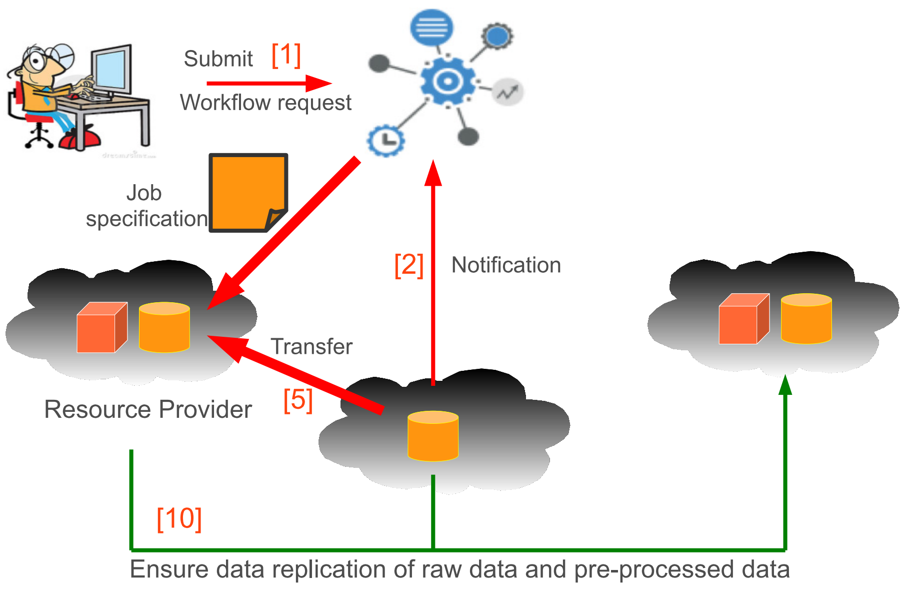

Figure 5: Data preprocessing on data ingest.

4. The Orchestrator selects the best compute site to perform the requested processing. To do so, it might collect information from different sources: Configuration Management DB, Software configuration Lean Agile Management, Monitoring, Storage endpoints.

5. The Orchestrator triggers a data movement through Rucio in order to copy the data to the selected compute center.

6. The Orchestrator gets notified on the completion of the data transfer, by listening to the XDC Message Bus.

7. The Orchestrator triggers the processing Job by submitting the request to those computing clusters available at the site.

8. As soon as the job output is produced, its availability is notified to interested parties, in particular Rucio, via the XDC message bus.

9. The data, generated by the processing step, is automatically registered into Rucio though the Orchestrator, using storage events.

10. Rucio takes care that the policies requested for the raw and preprocessed data are applied.

Figure 6 provides a high level overview of the orchestration architecture. Executing the predefined rules is thus performed by the INDIGO Orchestrator, which is also responsible for handling attached compute resources like cloud or batch systems. However, the orchestrator is delegating all storage resource operations to the Rucio framework as it already provides the necessary functionality and interfaces to data transfer, federation and storage endpoints.

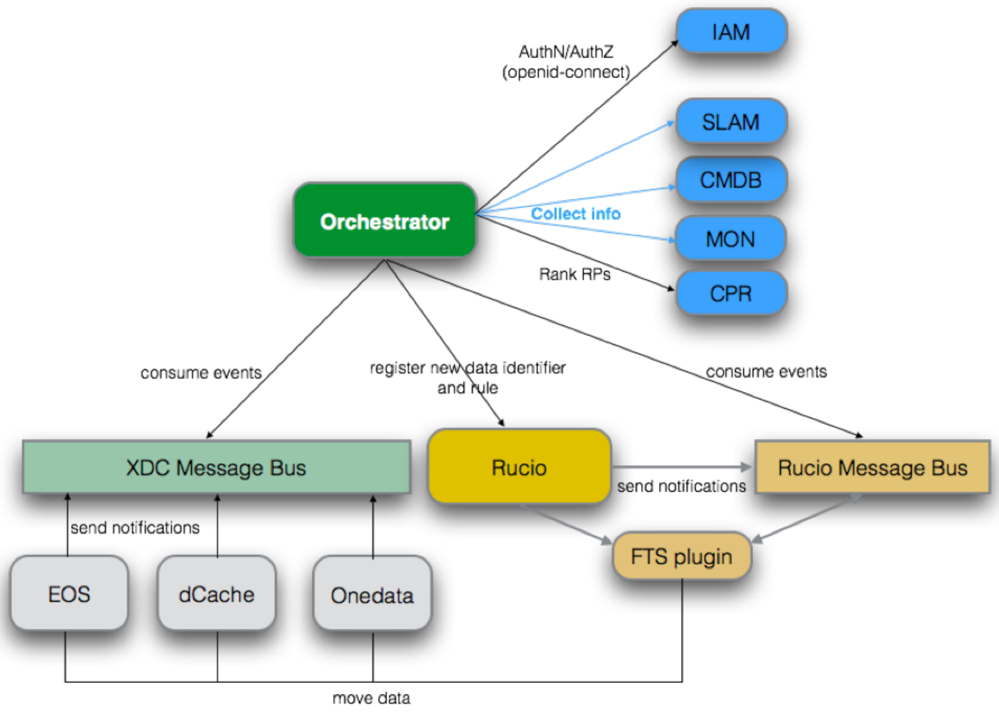

Figure 6: Components and communication for event driven orchestration of data and compute resources. See caption of figure 4 for details on the XDC respectively Rucio message bus.

4.3 Advanced Caching

Highly distributed data infrastructures require additional precautionary measures to overcome the limitations of wide area connections, as there are data link failures and congestions or the inevitable message latencies. Traditionally those issues are overcome by introducing data caching layers close to the data sinks. Those caches can either be filled by the orchestration engine in advance or automatically on request. The *Advanced orchestration* is already covered in the 'Orchestration' chapter 4.2. Consequently, this section focuses on the local caching *on demand* functionality. Although there is a variety of possible ways to cache data, we will concentrate on three cases:

- *Data-Only* cache nodes
- *Name Space Enabled* cache nodes
- *Managed* cache nodes

4.3.1 *Data-Only* cache nodes

The *Data-Only* cache approach is illustrated in figure7. Within an area *logically close* to the central storage system, clients are served by that system directly. Although *logically close* in general refers to the network latency of the connection, the concept can be generalizing and can be interpreted as everything which gives *logically close* an advantage over *far*, e.g. the network bandwidth or the price of the connection. Areas outside of *logically close* are served by Data-Only nodes. Those nodes are only storing the data, while all name space operations are still performed by the central system. When the location manager of the central system determines that a client exceeds a certain distance as defined above, it transfers the data to the closest location, seen from the client, and instructs the client to redirect the request to that node, as depicted in figure 8. Naturally, the central system needs to keep track of the contents of all remote nodes, so that a second request for already transferred data from a similar location can be redirected right away. There are some issues to consider when implementing this approach:

- This system only provides mechanisms to overcome wide area latencies. A wide area network breakdown will cause all remote nodes not to function any more, as each file request is directed to the central system first.
- In case of a failure of the remote *Data-Only* cache node, the central system must be informed via a back channel, to avoid permanent failures of clients close to that node.
- Some advantages of distributed systems are not taken advantage of in this scenario, as
  - the central name space database is growing with the number of remote *Data-Only* cache nodes and its file content
  - the central system has to handle all name space operations even for those files which are already close the requesting clients.
- As there is always communication ongoing between the client and the central system, the latency problem will still accur for each file, an effect which may become dominant, particularly for small files.

However the advantage is an expected low maintenance cost of the *Data-Only* cache nodes. Therefore the deployment of Data-Only cache nodes is preferably envisioned at sites focusing on providing compute resources instead of data storage.

4.3.2 *Name Space Enabled* cache nodes

To overcome most of the issues of the *Data-Only* cache system we are investigating *Name Space Enabled* cache nodes, using a Squid-like approach. In this cache the initial request for a file is always directed to the local cache node first. In case of a cache miss, the local system would take care of fetching the requested data from the closest remote location before it delivers the data to the client. With this approach, not only the data access latency would be minimized but also the latency of name space operations. Moreover, the system would be agnostic to wide area failures for data already present locally. Here as well some issues need further consideration:

- The client needs to know the address of the local endpoint. Although this should not an issue, a failure of the cache for that area is difficult to reconfigure, unless dynamic DNS entries are used.

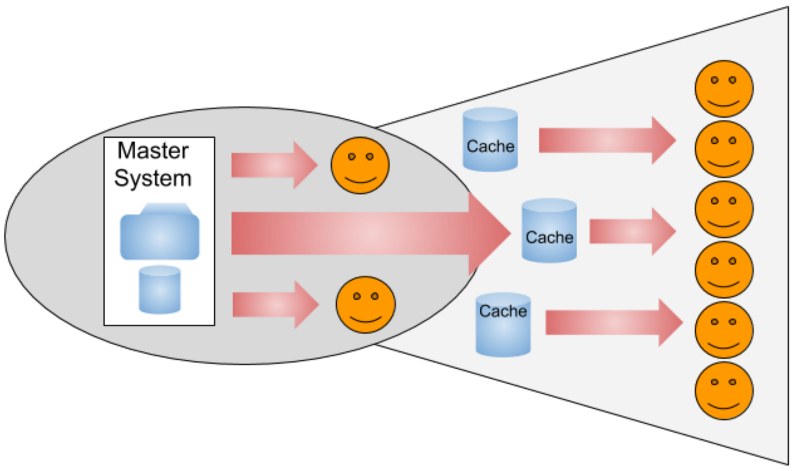

Figure 7: An infrastructure based on *data-only* caching nodes. The red arrows are representing the data flow. The bigger arrow indicates that the network bandwidth between the master system and the *data-only* caches likely is higher than to the end users.

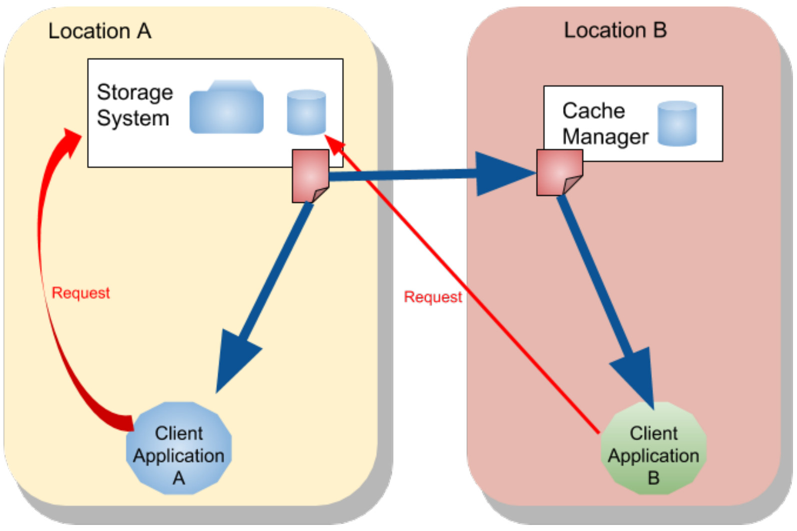

Figure 8: Request flow for a *data-only* caching node. The blue arrows are indicating the data movement between the various components.

4.3.3 *Managed* cache nodes

With the above two cache approaches, the assumption is that the caching site is not interested or not capable of hosting managed storage. This is not necessarily true for sites already running large storage installations. Nevertheless, even those sites might want to enable transient storage of data without the need to install a new software stack that supports acting as caching storage for a remote system. Therefore we are investigating a more complex cache scenario where the name space entry and possibly the access control information is transferred together with the data to the system closer to the client that is requesting the data. Figure 9 depicts such an infrastructure. It is essentially composed of two central systems, virtually connected via a control channel to communicate meta

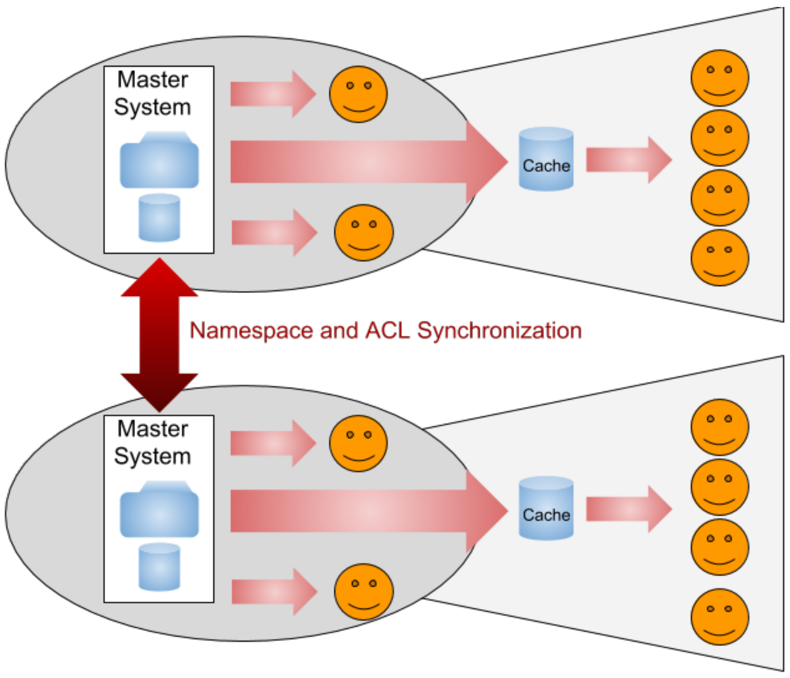

Figure 9: An infrastructure based on *managed* caching nodes.

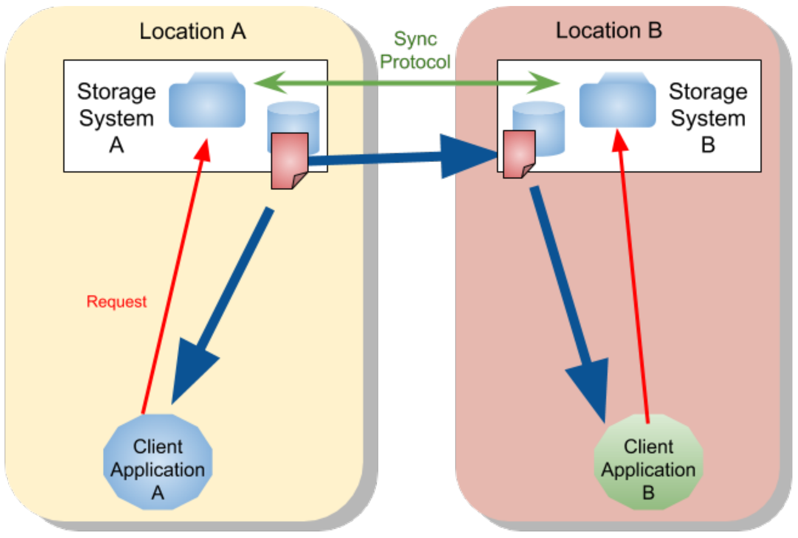

Figure 10: Request flow for *managed* caching nodes.

data information that is accompanying the actual data transfers. Figure 10 illustrates the process in more detail. Client B is requesting data from system B. As this data is currently only available at system A, B is fetching the data from A, including primary meta-data information like the name entry and the access control information. This name entry is added to the name space of system B. Within B, the data is declared *cached* and will be removed according to the cache cleaning policies of B. There is no need to inform system A in case the file is removed from the cache-part of B. The data is just re-fetched if needed. There are additional options possible for this scenario which we didn't further investigate yet but might become important:

- A cached copy of a file in system B could be declared 'permanent' and would then appear in the permanent part of the name space.
- The *Managed* cache nodes approach allows to stack more than one system. At the end of the chain there could be a *Name Space Enabled* cache node or a *Data-Only* cache node.
- Cache System B could be connected to more than one upstream node. Theoretically, the approach would allow for a meshed setup.

5. Summary and outlook

The high-level objective of this work is the automated placement of scientific data in the Exabyte region on the site (IaaS) as well as on the federated storage level, including the realization of necessary cache layers and the negotiation of required versus available Quality of Service classes in storage. We are working in three strongly interlinked tasks to achieve these goals. These are the implementation of Quality of Service in available storage systems, the advanced caching to overcome known problems of highly distributed systems, and the orchestration of data to cope with experiment work flows or predefined data life cycle policies. As the project just started we are still working on finalizing parts of the overall architecture which will be published as part of the XDC deliverable D1.6[30] before November 2018. Besides the technical goals, the intention is to be able to improve the existing infrastructure provided by WLCG, EGI or EUDAT without significant changes in the deployed software stacks, as intrusive changes would not be accepted by those infrastructures, based on their risk assessment or imposed costs.

References

[1] XDC Grant Agreement: 777367 eXtreme DataCloud H2020 EINFRA 2017, *https://ec.europa.eu/research/participants/portal* ; *Project ID =777367; Program ID=31045243*

[2] INDIGO-DataCloud Grant Agreement: 653549 H2020 EINFRA 2017, *https://ec.europa.eu/research/participants/portal* ; *Project ID =653549; Program ID=31045243*

[3] https://github.com/SNIA/CDMI/tree/indigo-dc

[4] ISO/IEC 17826:2016 Information Technology *Cloud Data Managment Interface, CDMI*

[5] Research Data Alliance *https://www.rd-alliance.org*

[6] *https://rd-alliance.org/group/qos-datalc-definitions-wg/case-statement/qos-datalc-definitions-wg-case-statement*

[7] Square Kilometer array *https://www.skatelescope.org/*

[8] World Wide LHC Computing Grid *http://wlcg.web.cern.ch/*

[9] European X-FEL *www.xfel.eu*

[10] The Storage Networking Industry Association *https://www.snia.org/*

[11] AP Millar, T Baranova, G Behrmann, C Bernardt, P Fuhrmann, DO Litvintsev, T Mkrtchyan, A Petersen, A Rossi, A and K Schwank, *Cache, agile adoption of storage technology*, IOP Publishing, Journal of Physics: Conference Series 2012, 3966,3,32077–32087

[12] P Fuhrmann et al. *Storage quality-of-service in cloud based scientifc environments: a standardization approach*, October 2017Journal of Physics Conference Series 898(6):062043

[13] AJ Peters et al. *EOS Development*, October 2017 Journal of Physics Conference Series 898(6):062032,DOI10.1088/1742-6596/898/6/062032

[14] A. Kiryanov, M Salichos, AA AyllÃşn and O Keeble *FTS3 - A File Transfer Service for Grids, HPCs and Clouds*, Conference: International Symposium on Grids and Clouds 2015, DOI10.22323/1.239.0028

[15] V. Garonne, G A Stewart, M Lassnig, A Molfetas, M Barisits, T Beermann, A Nairz, L Goossens, F B Megino and C Serfon *The ATLAS Distributed Data Management project: Past and Future*, Journal of Physics: Conference Series, Volume 396, Part 3

[16] The Dynamic Federation project *http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project*

[17] StoRM Web Site *https://italiangrid.github.io/storm/*

[18] The INDIGO DataCloud PaaS Orchestrator *https://www.indigo-datacloud.eu/paas-orchestrator*

[19] Scientific Data Management, Rucio *https://rucio.cern.ch/*

[20] The European Open Science Cloud. *https://www.egi.eu/about/newsletters/what-is-the-european-open-science-cloud*

[21] The Onedata distributed storage technology. *https://onedata.org/*

[22] The Integrated Rule-Oriented Data System, iRODS. *https://irods.org/*

[23] The IBM General Parallel Filesystem, GPFS, *https://searchstorage.techtarget.com/definition/IBM-General-Parallel-File-System-IBM-GPFS*

[24] The Cherenkov Telescope Array. *https://www.cta-observatory.org/*

[25] The Large Hadron Collider. *https://home.cern/topics/large-hadron-collider*

[26] The LifeWatch ERIC. *https://www.lifewatch.eu/*

[27] The Amazon S3 Storage Service. *https://aws.amazon.com/s3/*

[28] The Microsoft Azur Storage Service.*https://azure.microsoft.com/en-us/*

[29] The TOSCA Simple Profile in YAML Version 1.0 *http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/TOSCA-Simple-Profile-YAML-v1.0.html*

[30] eXtreme-DataCloud deliverable D1.6 (General Architecture) and D4.1 (WP4 Architecture) (will be published at *http://www.extreme-datacloud.eu/deliverables/*)