

Generic Metadata Handling in Scientific Data Life Cycles

Dissertation

zur Erlangung des akademischen Grades Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Diplom-Informatiker Richard Grunzke
geboren am 14. Januar 1983 in Hagenow

Gutachter:

Prof. Dr. rer. nat. Wolfgang E. Nagel, Technische Universität Dresden
Prof. Dr. rer. nat. Achim Streit, Karlsruher Institut für Technologie

Tag der Einreichung:

16. Dezember 2015

Tag der Verteidigung:

12. April 2016

Acknowledgments

I hereby want to express my deep gratitude to all persons that supported me in creating this thesis.

First and foremost I would like to thank my doctoral supervisor Prof. Dr. Wolfgang E. Nagel for his continued support, trust, and advise. I'm thankful to my advisor Dr. René Jäkel for being available when needed to constructively discuss various aspects of the thesis and for generally being very critical which helped to ensure a quality thesis. Then, I want to thank my department head Dr. Ralph Müller-Pfefferkorn for letting me carve out time to pursue my own research and projects. I want to thank my colleagues at ZIH for the very nice work environment. Distinct thanks go to Olaf Krzikalla for highly motivating mobile discussions about doctoral research and Daniel Molka for discussing motivational aspects. I'm also thankful to the subsequent organizers of the ZIH doctoral Seminar, Dr. Andreas Knüpfer, Prof. Dr. Florina Ciorba, and Dr. Matthias Lieber, for enabling this stimulating and thought provoking event series that helped me stay on track and get a broader context.

I want to thank the people in the very productive and nice MoSGrid collaboration. Special mentions go to Dr. Jens Krüger, Prof. Dr. Sandra Gesing, and Prof. Dr. Sonja Herres-Pawlis for being highly motivating and inspiring. Besides MoSGrid, the second cornerstone of my doctoral research was the LSDMA project. Here, I want to especially thank Prof. Dr. Achim Streit for his support and for enabling the freedom to pursue my research. Thanks also go to Parinaz Ameri for organizing the LSDMA doctoral seminar 2014.

I'm highly appreciative of the funding agencies for granting the projects that supported me in pursuing my doctoral research; the BMBF for MoSGrid, the EU for ER-flow, the Helmholtz association for LSDMA and the DFG for MASi. I'm thankful to anonymous reviewer 2 of our IEEE eScience 2015 paper [GKG⁺15] for taking the time to be extraordinarily thorough in reviewing the manuscript and raising interesting thoughts, aspects, and interconnections that led to significant improvements of the manuscript and subsequently of aspects of Chapter 2 of this thesis.

I want to thank Dr. René Jäkel, Dr. Jens Krüger, Prof. Dr. Sandra Gesing, Dr. Ralph Müller-Pfefferkorn, and Dr. Andreas Knüpfer for proofreading and suggesting many improvements. Special thanks in this department go to my advisor Dr. René Jäkel for being extremely thorough and leaving no sentence unturned in the small scale while at the same time being highly mindful of the big picture. His proofreading and suggestions led to substantial improvements of the dissertation.

I'm thanking my parents Roswita and Wolfram for their trust in my decisions and the generous support of my academic studies which gave me considerable freedom of choice and lay the groundwork for my doctoral research and dissertation.

Finally, I'm especially thanking my partner Linda for her appreciation, acceptance, and trust.

Contents

1	Introduction	5
1.1	Context	5
1.2	Challenges	5
1.3	Contributions and Impact	6
1.4	Publications	7
1.5	Thesis Structure	9
2	Managing Complexity in Scientific Data Life Cycles	11
2.1	The Scientific Data Deluge	11
2.1.1	Data Sources	12
2.1.2	Data Sinks	14
2.2	Handling Scientific Data	15
2.2.1	Storage	16
2.2.2	Data Management	17
2.2.3	Metadata Management	18
2.3	Applying Computing in Data Life Cycles	26
2.3.1	Computing	27
2.3.2	Computing Management	27
2.3.3	Workflow Management	28
2.4	Security	30
2.4.1	Authentication	30
2.4.2	Authorization	31
2.4.3	Single Sign-on	32
2.5	Data Life Cycle Utilization	32
2.5.1	API-based	35
2.5.2	Commandline-based	37
2.5.3	Workbench-based	37
2.5.4	Web-based	38
2.5.5	Visualization	39
2.6	A Molecular Simulation Data Life Cycle	39
2.6.1	Data Sources and Sinks	40
2.6.2	Data Management	41
2.6.3	Computing and Workflow Management	41
2.6.4	Authorization and Authentication Infrastructure	43
2.6.5	Utilization and Visualization	45

2.6.6	Application Domains	48
2.7	Further Data Life Cycles	53
2.7.1	Worldwide Large Hadron Collider Computing Grid	53
2.7.2	Human Brain Project	54
2.7.3	Pierre Auger Observatory	55
2.7.4	IceCube	55
2.7.5	Virtual Earthquake and Seismology Research Community in Europe e-Science Environment	56
2.7.6	Climate-G Testbed	56
2.7.7	PolarGrid Portal	56
2.7.8	Distributed Research Infrastructure for Hydro-Meteorology Community	57
3	A Novel and Generic Metadata Handling Concept	59
3.1	Limits in the Current Situation	59
3.1.1	Important Challenges out of Focus	59
3.1.2	The Challenge	61
3.2	Improving the Situation - Focus and Scope	62
3.2.1	The Overall Approach	62
3.2.2	Alternative Approaches	62
3.3	Characteristics of Generic Metadata Handling	64
3.3.1	Abstraction from Technologies	64
3.3.2	Generic Handling of Metadata and Data Formats	66
3.3.3	Seamless Data Life Cycle Integration	67
3.4	A Design Guide to Metadata in Data Life Cycles	68
3.4.1	Overall Data Life Cycle Design	68
3.4.2	Metadata Management Integration	71
3.4.3	Technology Recommendations	72
4	Implementation within a Complex Data Life Cycle	77
4.1	The Molecular Simulation Markup Language as Information Hub	77
4.1.1	Molecular Simulation Markup Language	77
4.1.2	Integration with Data Life Cycle	78
4.2	Metadata Extraction, Annotation, and Indexing	82
4.2.1	Extraction and Annotation	82
4.2.2	Indexing	83
4.3	Metadata Service Integration and Result Utilization	84
4.3.1	Search Interface and Metadata Service Access	84
4.3.2	Search Result Utilization	84
4.4	Overall Integration and Example Usage Scenario	85
4.4.1	MoSGrid Data Life Cycle Integration	85
4.4.2	Example Usage Scenario	86

5	Evaluation	89
5.1	Adaptability	89
5.1.1	Concept Adaptability	89
5.1.2	Adaptation Outlook for a High-Throughput Big Data Microscopy Use Case	90
5.2	Performance	91
5.2.1	Various Aspects	91
5.2.2	Metadata Extraction and Annotation	92
5.2.3	Metadata Indexing	95
5.3	Sustainability	97
5.4	Resilience	98
5.5	Efficiency of Use	99
6	Conclusion and Outlook	101
6.1	Conclusion	101
6.2	Outlook	102
A	Publication List	105
	Bibliography	111

1 Introduction

This chapter introduces the dissertation by describing its context, the identified challenges, how the chosen challenge was met, the achieved impact, relevant publications, and how the thesis is structured.

1.1 Context

In science data is the essential focal point in today's computational and quantitative approaches to scientific knowledge gain. Computational simulations enable far reaching explorations of modeled realities while quantitative methods gather data to improve the understanding of observed phenomena. These methods are increasingly viable only via high-end storage and large-scale High Performance Computing resources with individual requirements dramatically rising. Data throughputs involve gigabytes per second continuously, volumes are of petabyte magnitude, continuous files per second rates are in the double-digit range, and a vast universe of complex data representations exists. The great potential of such data is evident by the current trend of Big Data in science that aims at large-scale information extraction to foster scientific discoveries. This is fundamentally enabled by intelligently handling data and by combining a large variety of information technology methods to so-called data life cycles. In principle, these consist of data sources, systems to manage data as well as compute resources, methods for access rights management, utilization interfaces and data sinks. Scientists are naturally focused on their particular research. Thus, metadata is an essential step forward in the efficiency of use as it enables managing data based on its content instead of location. Via specific data life cycles scientists are freed from the necessity to extensively deal with IT infrastructures while still utilizing them to drive their research by handling their extensive data and computing demands. In this complex technological environment, a plethora of significant challenges presents itself that hinders the advancement of the state-of-the-art in data-driven knowledge gain.

1.2 Challenges

Vital challenges in managing data life cycles are manifold. Federated authentication and authorization infrastructures need to be integrated while being mindful of the overall resilience of increasingly complex data life cycles. The increasing numbers of files and data amounts need to be managed by Big Data systems. These in turn need to be efficiently integrated with High Performance Computing resources for analysis which signifies the need for advanced interoperability. Besides automated pre- and postprocessing, the user-friendly creation, and execution of workflows to encapsulate complex analysis procedures need to be supported. Integrated scientific environments need to be provided that hide the underlying complexity while enabling that use. Essential is also the building of trust that an infrastructure delivers

what it promises. Closely connected is moving from a fixed-term build up phase to a sustainable operation phase. As these goals are partly opposing to each other, a effective balance between them needs to be developed for each data life cycle. The dissertation focuses on the major challenge of the organization of large numbers of files in the million range using information about data, so-called metadata. Currently, solutions are often either use case specific or lacking completely, thus, preventing easy access and re-use. Without metadata, users have to remember where an individual file is located. With a large number of files this is inefficient if not impossible. This especially holds true for Big Data use cases with a large number of files with complex content and stored in distributed locations. Currently, significant efforts need to be made to implement even narrowly applicable and pragmatic metadata handling solutions for every new scientific experiment.

1.3 Contributions and Impact

The contributions described in this thesis to meet the major challenge and the resulting impact are described in the following. In all cited references the author was active as main author or co-author.

First, to facilitate a thorough understanding of the context and challenges within the highly complex situation of managing distributed data life cycles, a comprehensive and overarching analysis and classification of all major data life cycle elements was created [GKG⁺15]. This novel and extensive analysis encompasses the complete data life cycle from creation over data handling and processing to archiving (see Sections 2.1 to 2.5). MoSGrid, the basis for the concept example implementation, is an advanced data life cycle for enabling complex and compute-intensive molecular simulations by integrating data resources with HPC and workflow management in a secure and efficiently usable way (see Section 2.6) [KGG⁺14, GBB⁺12, GKG⁺14]. Further data life cycles are described to give a broad context (see Section 2.7). Non-existent or highly use case specific metadata management approaches were identified, besides others, as major challenges in data life cycles across scientific communities (see Section 3.1). The necessity of generic overarching metadata approaches is elucidated via an evaluation of important metadata approaches [GHS⁺14] and the need for higher abstraction levels towards exascale [JMPK⁺15].

Second, improving upon this situation, a generic and comprehensive metadata concept was designed (see Section 3.2.1). The concept is widely applicable yet enables scalability and efficiency of use [GGJN14, GBG⁺14]. With respect to other approaches (see Section 3.2.2), it improves upon the state-of-the-art. On the one hand, the concept provides the following characteristics of generic and well-balanced metadata handling (see Section 3.3). Abstraction from various kinds of technologies is important in order to enable efficient integration and adaptability to new high performance Big Data life cycles. Metadata and data formats need to be generically handled from the source data format over storing to being searchable in the end. This enables users to efficiently and transparently organize their data. Steps such as metadata extraction, annotation, and indexing must be fully automated for management of large numbers of files. Metadata management needs to be seamlessly integrated with underlying infrastructures including systems from data, computing, security, and utilization categories. Together with the possibility to directly execute computing tasks based on metadata search results this facilitates a high usability. On the other hand, the concept provides a design guide to metadata in data life cycles (see Section 3.4). As scientific

data life cycles are highly complex, overall design aspects are described in detail to facilitate an appreciation and understanding of this complexity [GDP⁺15]. Then, metadata design aspects are thoroughly discussed. For example, re-creating of metadata capabilities are avoided by facilitating the integration of standard components. This enables quick adaptations to different types of use cases and overall integration paths into new scientific data life cycles. Technology recommendations include an analysis of proven technologies that directly enable metadata management and beyond in data life cycles. The concept advances the state-of-the-art in data life cycle management. When implemented, it enables scientists to focus on their core research while utilizing Big Data and High Performance Computing resources.

Third, based on the generic concept an example implementation for the MoSGrid data life cycle was created within the MoSGrid collaboration (see Section 4). On the one hand, MoSGrid concept implementation adopts the concept characteristics by utilizing abstraction layers. It transparently integrates with the complex underlying High Performance Computing and data infrastructures and with the single sign-on concept and implementation [GGK⁺12]. Based on the MoSGrid data description format, the extraction, annotation and indexing is fully automatic [GBG⁺14]. The seamless integration of the metadata capabilities are the basis for the implemented and integrated search interface. It facilitates the finding of data and enables the seamless use of results for further workflows. On the other hand, the metadata aspects of the concept's design guide (see Section 3.4.2) played a central role in the MoSGrid implementation as well as the technology recommendations (see Section 3.4.3) that were closely followed. The implementation results in a major advancement of the MoSGrid data life cycle by extending its capabilities in handling large amounts of complex data.

Fourth, the generic metadata management approach and its example implementation for the MoSGrid data life cycle were evaluated [GKJ⁺ed, GBG⁺14] on the basis of criteria that permeate the concept (see Section 5). The generic metadata approach is enabling important synergy effects in supporting new data life cycles (see Section 5.1). One is the ability to quickly integrate metadata capabilities. Another is the increased efficiency based on enabling an easy and seamless integration with High Performance Computing and Big Data infrastructures (see Section 5.5). A performance evaluation of the extraction, annotation, and indexing key components shows favorable characteristics (see Section 5.2). Furthermore, sustainability (see Section 5.3) and resiliency aspects (see Section 5.4) are evaluated.

This thesis is a step towards widely enabling metadata management across scientific disciplines. The uptake of the concept and its implementation in the MASi research infrastructure (see Section 6.2) as well as its utilization within the MoSGrid data life cycle ensures a broad and lasting impact.

1.4 Publications

The results of the dissertation and intermediary connected research were published in various journals, book chapters, and proceedings of conferences and workshops. The following publications are closely connected to the doctoral research and are referenced in Section 1.3. The complete list of the author's publications is attached as appendix A.

[GBB⁺12] **Richard Grunzke**, Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Sandra Gesing, Sonja Herres-Pawlis, Oliver Kohlbacher, Jens Krüger, Martin Kruse, Ralph

- Müller-Pfefferkorn, Patrick Schäfer, Bernd Schuller, Thomas Steinke, and Andreas Zink. A Data Driven Science Gateway for Computational Workflows. In UNICORE Summit 2012 Proceedings, volume 15 of IAS Series, pages 35–49, 2012.
- [GBG⁺14] **Richard Grunzke**, Sebastian Breuers, Sandra Gesing, Sonja Herres-Pawlis, Martin Kruse, Dirk Blunk, Luis de la Garza, Lars Packschies, Patrick Schäfer, Charlotta Schärfe, Tobias Schlemmer, Thomas Steinke, Bernd Schuller, Ralph Müller-Pfefferkorn, René Jäkel, Wolfgang E. Nagel, Malcolm Atkinson, and Jens Krüger. Standards-based Metadata Management for Molecular Simulations. *Concurrency and Computation: Practice and Experience*, 26(10):1744–1759, 2014.
- [GDP⁺15] Sandra Gesing, Rion Dooley, Marlon Pierce, Jens Krüger, **Richard Grunzke**, Sonja Herres-Pawlis and Alexander Hoffmann. Science Gateways - Leveraging Modeling and Simulations in HPC Infrastructures via Increased Usability. *High Performance Computing Simulation (HPCS)*, 2015 International Conference on, 2015, 19-26.
- [GGJN14] **Richard Grunzke**, Sandra Gesing, René Jäkel, and Wolfgang E. Nagel. Towards Generic Metadata Management in Distributed Science Gateway Infrastructures. In *IEEE/ACM CC-Grid 2014 (14th International Symposium on Cluster, Cloud and Grid Computing)*, pages 566–570, Chicago, IL, US, May 2014.
- [GGK⁺12] Sandra Gesing*, **Richard Grunzke***, Jens Krüger, Georg Birkenheuer, Martin Wewior, Patrick Schäfer, Bernd Schuller, Johannes Schuster, Sonja Herres-Pawlis, Sebastian Breuers, Ákos Balaskó, Miklos Kozlovsky, Anna Szikszay Fabri, Lars Packschies, Peter Kacsuk, Dirk Blunk, Thomas Steinke, Andre Brinkmann, Gregor Fels, Ralph Müller-Pfefferkorn, René Jäkel, and Oliver Kohlbacher. A Single Sign-On Infrastructure for Science Gateways on a Use Case for Structural Bioinformatics. *Journal of Grid Computing*, 10(4):769–790, 2012.
- [GHS⁺14] **Richard Grunzke**, Jürgen Hesser, Jürgen Starek, Nick Kepper, Sandra Gesing, Marcus Hardt, Volker Hartmann, Stephan Kindermann, Jan Potthoff, Michael Hausmann, Ralph Müller-Pfefferkorn, and René Jäkel. Device-driven Metadata Management Solutions for Scientific Big Data Use Cases. In *22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2014)*, February 2014.
- [GKG⁺14] Sandra Gesing, Jens Krüger, **Richard Grunzke**, Luis de la Garza, Sonja Herres-Pawlis, and Alexander Hoffmann. Molecular Simulation Grid (MosGrid): A Science Gateway Tailored to the Molecular Simulation Community. In *Science Gateways for Distributed Computing Infrastructures*, pages 151–165. Springer International Publishing, 2014.
- [GKG⁺15] **Richard Grunzke**, Jens Krüger, Sandra Gesing, Sonja Herres-Pawlis, Alexander Hoffmann, Alvaro Aguilera, and Wolfgang E. Nagel. Managing Complexity in Distributed Data Life Cycles Enhancing Scientific Discovery. In *e-Science (e-Science)*, 2015 IEEE 11th International Conference on, pages 371–380, August 2015.
- [GKJ⁺ed] **Richard Grunzke**, Jens Krüger, René Jäkel, Wolfgang E. Nagel, Sonja Herres-Pawlis, and Alexander Hoffmann. Metadata Management in the MoSGrid Science Gateway for Quan-

tum Chemistry. Journal of Grid Computing, accepted.

[JMPK⁺15] René Jäkel, Ralph Müller-Pfefferkorn, Michael Kluge, **Richard Grunzke**, and Wolfgang E. Nagel. Architectural Implications for Exascale based on Big Data Workflow Requirements. In Big Data and High Performance Computing, volume 26 of Advances in Parallel Computing, pages 101 – 113. IOS Press, 2015.

[KGG⁺14] Jens Krüger*, **Richard Grunzke***, Sandra Gesing*, Sebastian Breuers, André Brinkmann, Luis de la Garza, Oliver Kohlbacher, Martin Kruse, Wolfgang E. Nagel, Lars Packschies, Ralph Müller-Pfefferkorn, Patrick Schäfer, Charlotta Schärfe, Thomas Steinke, Tobias Schlemmer, Klaus Dieter Warzecha, Andreas Zink, and Sonja Herres-Pawlis. The MoSGrid Science Gateway - A Complete Solution for Molecular Simulations. Journal of Chemical Theory and Computation, 10(6):2232–2245, 2014.

* These authors contributed equally to the respective work.

1.5 Thesis Structure

Chapter 2 of the thesis introduces the highly complex field of scientific data exploitation. Data life cycle management components are filed in categories such as data sources and sinks, storage and computing, data and computing management, metadata and workflow management, and security and utilization. Relevant systems and approaches in this context are discussed. The chapter continues with a detailed description of the MoSGrid data life cycle in which the example implementation was performed. It is rounded out with an overview description of various further data life cycles.

Chapter 3 starts with presenting overall challenges in the complex data life cycle situation and follows with the central metadata challenge of too specific or completely lacking solutions. The contribution of a generic and overarching metadata concept is described in detail. First, it describes key characteristics of abstraction, generic format handling, and seamlessness that are essential for advantageous metadata management implementations. Second, it includes a design guide for developing an understanding of the general data life cycle complexity and how to integrate metadata management. Finally, technology recommendations are given.

Chapter 4 describes the example implementation of the generic concept following the characteristics, design guide and technology recommendations. It extends the complex MoSGrid data life cycle with generic metadata management in a fully automatic and seamlessly integrated way.

Chapter 5 evaluates the generic concept theoretically and in practice along the MoSGrid example implementation. The criteria are adaptability, performance, sustainability, resilience, and efficiency of use. The concept and implementation are shown to have favorable properties.

Chapter 6 gives a detailed conclusion and looks ahead to future work and research directions.

2 Managing Complexity in Scientific Data Life Cycles

In this day and age, data is seen as a fundamental resource to gain knowledge and insights. This manifests itself in the trend of and concepts behind "Big Data" with data challenges in various dimensions. With exascale computing at the horizon the computing capabilities are rapidly increasing with core numbers in the order of millions. To support this, work is ongoing in the field of exascale hardware [KBB⁺08] and software [DBM⁺11, JMPK⁺15] to lay the ground for the efficient and resilient operations of such systems. The rate of data creation is continuously and rapidly growing while the raw data processing capabilities are even growing faster. This situation leads to an increasing focus on data as its transfer and management becomes the bottleneck.

Scientific data life cycles are highly complex due to high and increasing capability requirements from scientists. In order to manage them various technologies exist which are categorized and described in detail in this chapter. In Figure 2.1 the categories are depicted with data sources and sinks at the left and right sides respectively. A conceptional view is taken with commonly used systems being discussed as examples. The field of scientific data with a multitude of functions and components adds to the complexity of data life cycles. Ways to access and utilize data life cycles are presented as well as security mechanisms. The focus is on approaches to make complex data life cycles manageable and usable. The whole data life cycle with the essential management, analysis and utilization of data is considered [JGG⁺14]. A major characteristic of many data life cycles is the quickly increasing complexity with international collaborations and highly distributed resources [BF07]. The intelligent handling of data, its analysis and access by users and systems is increasingly the essential requirement to enable cutting-edge science. Thus, the overall goal is to enable scientists to generate new scientific knowledge based on data. This data exploration can be seen as the 4th paradigm of science [HTT09] following the empirical, theoretical, and computational approaches. A paper presenting the systematic and extensive data life cycle background within this chapter was recently published [GKG⁺15].

The MoSGrid data life cycle is described in detail as it provides the context in which the example implementation of the dissertation concept was performed. Further data life cycles are compiled to give a glimpse into the existing variety and complexity. This chapter facilitates a thorough understanding of the management of scientific data life cycles to give context to the contributions in Section 3.

2.1 The Scientific Data Deluge

The importance of data is thoroughly described in [BHS09] where data-intensive science is seen as the fourth paradigm of science with data as its center. In an important strategy paper by the German research council, data is described as the fundamental basis for science and beyond [Wis12]. In this chapter data sources and sinks are described (see Figure 2.2 for an overall classification within data

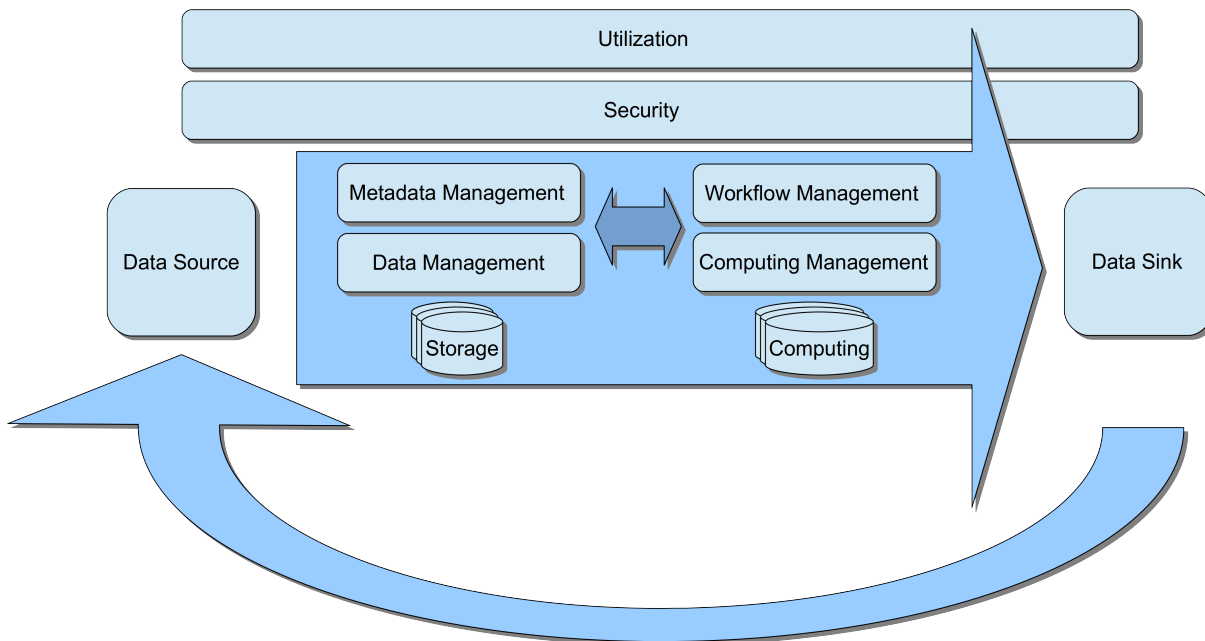


Figure 2.1: Principal data life cycle management component categories are depicted with data source, storage and computing pillars, data sink, and layers for security and utilization. Each category consists of a multitude of technologies to manage the inherent complexity.

life cycles). Instruments and simulations may act as sources and various data management systems as sinks. Archiving data is a huge challenge [BHS09] as it needs to be usable, efficient, cost-effective, and sustainable at the same time. The computer power is increasing faster compared to the increase of storage capacity and transfer rates. This trend tends to result in computing being performed near the data instead of moving data to computing resources.

2.1.1 Data Sources

There is a multitude of data sources with increasingly complex instruments and computing resources [HTT09]. The data rates, amounts, and complexity are continuously growing and data life cycles need the capabilities to manage this situation. This aligns itself with the trend of Big Data [BHS09, Eur10].

Instruments

One kind of data sources are instruments, defined here as hardware focused on a specialized set of functions. Particle accelerators [LBS62, AAA⁺08] are significant examples in this category with the Large Hadron Collider (LHC) [Bir11] as famous instance creating about 1 PB/s of raw event data per detector. Depending on the detector and use case, the data is significantly filtered to only retain data that is of further interest. For the LHCb experiment it is reduced to about 50 MB/s while for the ALICE experiment it can only be reduced to 1250 MB/s for certain experiments. Another example are instruments such as high-throughput microscopes in biology [CSB⁺10]. These can produce millions of files with reaching over 1 GB/s per instrument such as the Selective Plane Illumination Microscopy (SPIM) [HSDB⁺04, HS09, WMH13]. In its basic state one kind, for example, produces data with a

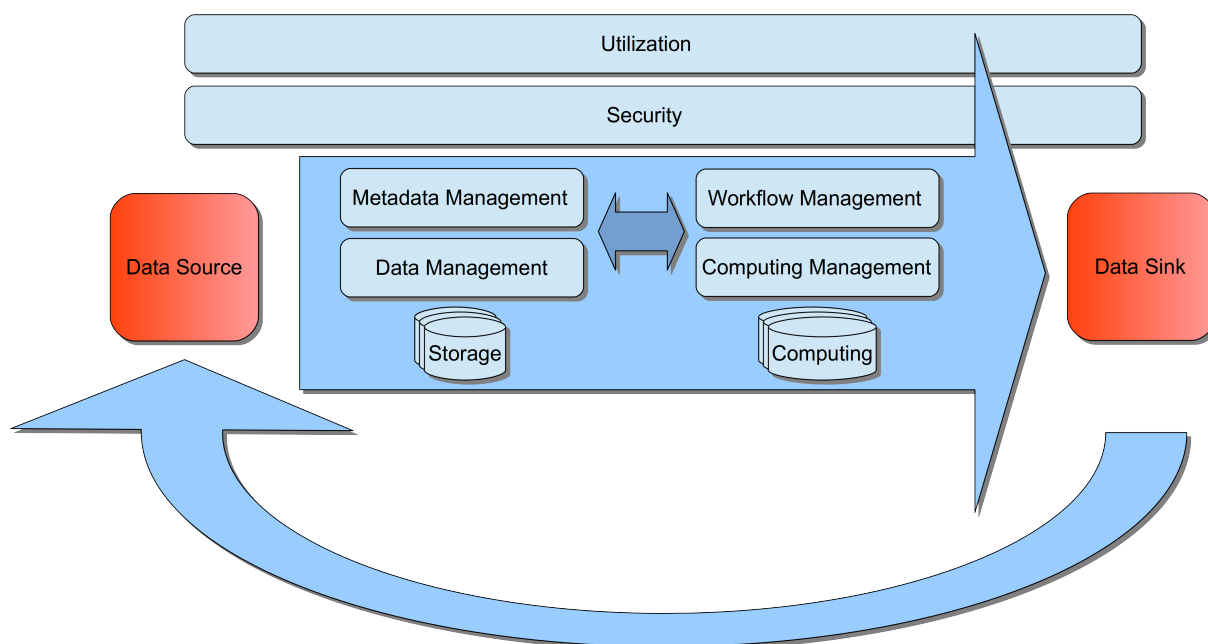


Figure 2.2: The data source and data sink categories include a classification of data creation and storing methods as principal input and outputs of a data life cycle.

rate of 0.85 GB/s and roughly 10 files/s. In continuous mode, which is the clear goal, 2 petabyte and 26 million files per month for each microscope in every institute that utilizes them will be created. Exact data and file rates depend on specific microscope configurations. Big Data in biology [Mar13] is a vast field with a multitude of use cases. Another category of instruments are sensors. In the VAVID project [VAV15, AGM⁺15], sensors are distributed and small and they measure a limited amount of monitoring data but for a huge array of wind power stations. With the station number of the order of thousands, the monitoring data amounts to almost one petabyte per year.

Computing Resources

Another kind of data sources are computing resources such as supercomputers and clusters. Under the consumption of time and conversion of energy these resources convert input data into output data of higher value. Supercomputers can produce highly dimensional and complex data at petabyte-scale e.g., highly dimensional plasma physics simulations [BBC⁺13]. Alternatively, data on a small scale but with a high-throughput are created with data analysis or extraction tasks. A high-throughput computing example is the analysis of the previously mentioned microscopy data with tasks such as registration, segmentation, and information extraction [CSB⁺10]. The author led collaborations to facilitate the efficient use of High Performance Computing (HPC) resources for such high-throughput image analysis use cases [GS10, GMPMM11]. Another high-throughput computing example are complex docking workflows within the MoSGrid data life cycle (see Figure 2.23).

2.1.2 Data Sinks

The term data sink is defined here as a storage location where temporary and result data of a data life cycle is ultimately kept for future use. The kind of storage location is chosen according to the re-use probability and safety requirements of the data. As systems configurations and policies vary the following list is intended to give a general impression of the situation within High Performance Computing environments. A scratch file system is often deployed system-wide. It is rather large, has a high bandwidth, has usually no backup, and is intended for temporary result data. A home file system is usually limited in size and bandwidth but is backed up, thus, it is intended to be used for short-term storing of final result data. An archive is designated for keeping final result data for the mid- and long-term. It is usually large, fully backed up, has a high bandwidth but high latency since often tape technology is used. An emerging trend is that further services are built on top of archives, which are called research data repositories [Wit08]. The data is usually annotated with metadata and can be referenced via a unique identifier. A common schema is the Digital Object Identifier (DOI) [Pas05] with the DataCite initiative [Bra09] working towards the goal of establishing research datasets as "as independent, citeable, unique scientific objects"[Bra09]. Important further aspects are access rights and licenses as not all data is aimed to be made open data immediately.

Policy and technological challenges in this area are manifold [Ber08]. For example, when data is archived it needs to be clear who is owning and who is allowed to access the data. It needs to be determined who is responsible for the data even when the original owner leaves the institute. This might be the institute head, but it might be questionable if he is willing to automatically assume responsibility as incorrect or fraudulent data might also be included. When the responsibilities are clear, such a transfer of responsibility should be as automatic as possible enabled by the technologies in use. Also, users might retain access rights even in case they leave the institute. This case is a challenge as an institute-internal account is not applicable anymore. Optimally, an institute-independent and worldwide ID system such as ORCID [HFP⁺12, KPWW10] is used to identify person independent of institutes.

One essential aspect is that with more affordable and, thus, increasingly available and larger storage devices, more data tends to be created and kept. This trend is part of technological developments that enable more extensive studies using scientific data as part of the Big Data trend. The UNESCO recognizes this implications in its *Charter on the Preservation of Digital Heritage* [UNE03]. Digital data in all forms is an increasing part of the worlds heritage. Thus, it is essential to sustainability save and make data available to following generations. It follows that universal interfaces to long-term archives are important to enable the easy storing and accessing of this data. Metadata and unique identifiers are fundamental in enabling that data can be searched for and worked with in a meaningful way, even in the ages to come. As storage for currently handled data acquisitions rates is limited, data has to be selected for long-term storage based on its relevance. Data is also seen as the natural resource of the information age and therefore needs stable, efficient, cost-effective, usable and sustainable digital infrastructures [Ber08]. In this context data preservation is essential to ensure the availability of data in the future, whether short- or long-term. The key challenge of sustainability is the continuous maintenance of system operation long after the time it was built up. With data archives it is of special importance as it is their base requirement to keep data for the long-term.

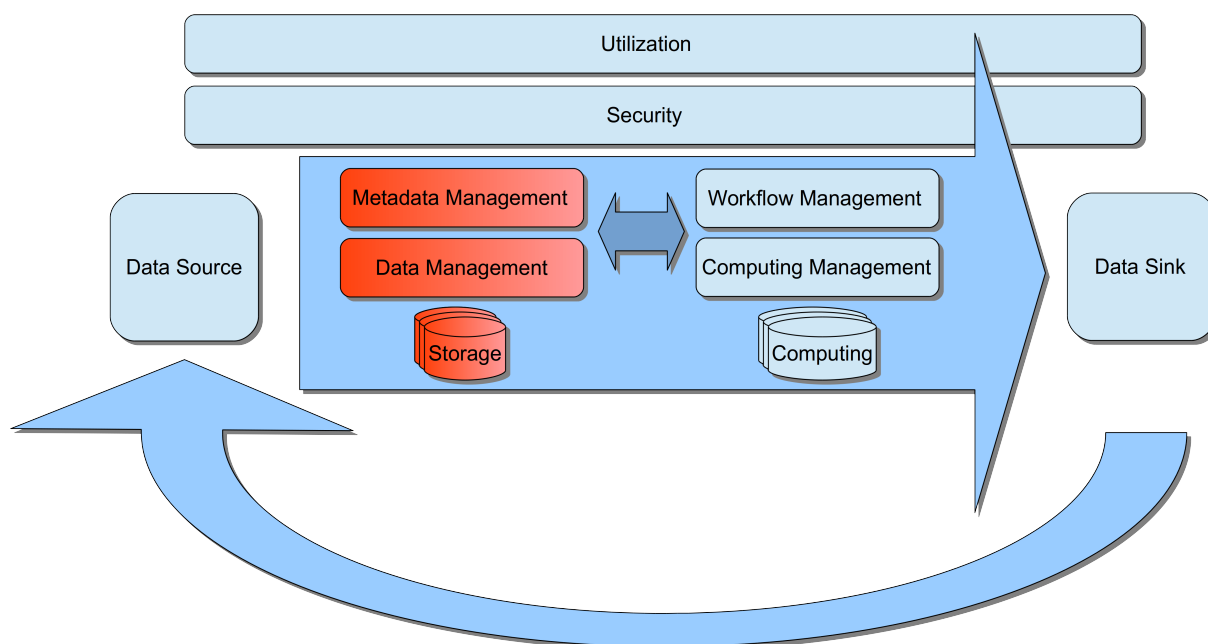


Figure 2.3: The data pillar in data life cycle management is depicted with categories of increasing levels of abstraction; storage, data, and metadata management.

2.2 Handling Scientific Data

Data in scientific life cycles (see Figure 2.3) need to be safely managed and stored [CFK⁺00] in order to be accessed by user and by diverse systems with high performance requirements [ABB⁺02]. Data sets can have large numbers of files and/or large file sizes, depending on the data source and how it is stored and processed. Nowadays, data sharing is becoming increasingly important [TAD⁺11, FZ11] with interdisciplinary collaboration on the rise as a way to create synergy effects and solve complex societal challenges [FZ11]. This positive impact is only possible when diverse scientific fields cooperate and when sharing is explicitly independent of the initial purpose for which the data was originally created. Sharing is fostered by the notion that publicly funded research endeavors should release its data back as open access data to the public that originally funded it. In case embargo periods are required by public research or in commercial research settings an integration with authentication and authorization infrastructures is essential. Section 2.4 will elucidate that security is a highly complex task.

The German Science Counsel published a strategy paper on the further development of research information infrastructures in Germany up to 2020 [Wis12]. It stresses the importance of information infrastructures, including digital forms, as the fundamental basis for science and beyond. Digital research data has to be made accessible, archived and, thus, kept for the long term.

According to the overall concept for the information infrastructure in Germany [dI11], sustainable data management with open access is a fundamental requirement to save, exploit, provide, and utilize data in the long-term. Research data is of cultural importance and must be permanently kept and provided to enable future re-use. Thus, operational resources have to be maintained to fulfill this goal. These resources include storage, computing capacities, documentation, safekeeping, archiving, and curation. Data management policies have to enforce the creation of data management plans and include respon-

sibilities and privacy considerations. Clear responsibilities have to exist to enable system maintenance, data curation, sharing, access control, data extension, and deletion. Responsible for the safe-keeping should be libraries and data centers. Best practice guides, standards, and process models are important to re-use once build-up expertise. Development resources include the automatic acquisition of data and metadata, persistent identifiers, access control, linking, and standards.

Raw data storage devices are storing data in file form and can have various levels of bandwidths, latencies, safety, and sizes (see Section 2.2.1). The data management category contains systems that enable the sophisticated organization of data based on distributed raw storage devices with additional functionality such as a unified name space, various access mechanisms, replication management, and policy enforcement. Details are provided in Section 2.2.2.

As data sets increasingly consist of numbers of files in the million range, information about data itself needs to be present and available, called metadata. It is used to find and access data without the need for users to remember what exactly is in which file and where the data is located. Automatic extraction, annotation, and indexing of metadata is essential to deal with the inherent complexity. As this topic is the core of the dissertation at hand, its background is dealt with in detail in Section 2.2.3, the overarching and generic metadata concept is presented in Chapter 3, an example implementation presented in Chapter 4 and evaluated in Chapter 5. Furthermore, authentication and authorization infrastructures including single sign-on (see Section 2.4) need to be naturally integrated with systems of the data pillar (see Figure 2.3).

2.2.1 Storage

This section introduces concepts and technologies regarding storage [TE03] at a suitable level of detail for this thesis. Here, storage is considered as a concept that is available on a server offering space to keep data. Depending on the use case and its requirements, different kinds of basic storage technologies are utilized, namely ramdisks, solid state disks, hard disk drives, and tapes. These technologies form the storage hierarchy whose component characteristics range from small, high-bandwidth, low latency, and expensive to huge, high-bandwidth, high latency, and cheap with mixes in between.

Storage can be connected to a server in various ways. It is available on such a server via a file system that is an abstraction layer on top of block storage devices [TE03]. One kind is the concept of Direct Attached Storage (DAS) where local block storages devices are attached via standards such as SCSI (Small Computer System Interface) to a single server. It initiates and manages the file system as an interface for applications to store files. A Storage Area Network (SAN) is a concept that includes dedicated local storage networks. These are optimized for latency and flexible in terms of configuration. The provided storage is also block oriented and common examples are FibreChannel and iSCSI. Network Attached Storage (NAS) is a client-server concept. Servers expose their storage capacity to other systems on a file basis and clients consume storage by integrating it into their local filesystem tree. Examples are NFS [SEN10] and CIFS [Her04] with parallel and distributed files systems with additional functionality and focus being described in Section 2.2.2.

2.2.2 Data Management

Data management systems are concerned with the organization of data mostly based on files and directory structures. Such systems need to be able to safely store and manage large quantities of data with a size range of petabytes as well as high number of files in the double-digit million range.

Parallel file systems

Parallel file systems usually provide functionality to combine various storage devices to a unified overall system with a common name space including an overall directory structure. An essential requirement is that systems in this category are able to scale with the amount of data. This scalability enables them to serve High Performance Computing environments and a large number of distributed accesses at the same time. In the following several widely used examples of parallel file systems are introduced.

The General Parallel File System (GPFS) [SH02] is a parallel file system featuring a POSIX (Portable Operating System Interface for Unix) interface and parallel access to both files and its metadata as well as administrative functions. GPFS' shared-disk architecture enables well scalable installations. It is mounted on the nodes of a cluster that are connected via a switching fabric to the shared disks. These disks store the segmented data which can subsequently be accessed by all nodes equally. This schema enables load balancing for a high-throughput and fault tolerance. To ensure file systems consistency during parallel read and write accesses, GPFS implements distributed locking. GPFS is installed on some of the largest supercomputers in the world [SH02].

Lustre [Sch03, B⁺04] is an object-based parallel file system. It is client-server based and features one or more metadata servers (MDSes) and targets (MDTs), object storage servers (OSSes), and targets (OSTs) and clients. Clients can access the unified name space of a Lustre installation via its POSIX interface. MDSes, integrated with MDTs, provide name space information including file and directory names, access rights, and keep the information where files on OSSes are stored. These servers mount a number of OSTs via its local file system. Data objects are potentially stored accross many OSTs and are referenced via a unique identifier. A high overall bandwidth is achieved by the direct transfer of data from OSTs to clients while consistency is guaranteed via a similar strategy as with GPFS. Lustre is also widely used on large-scale supercomputers [Sch03, B⁺04].

Parallel NFS (pNFS) [GWGC04] is an optional extension of the NFS v4.1 standard. It improves NFS' capabilities with a scalable approach of separating data and metadata. As with other parallel file systems, the metadata server provides information about the unified name space layout, access rights, and location of the actual data on OSS'. The distributed architecture provides scalability by enabling clients to directly access data while the metadata server enables consistency.

Distributed Data Management Systems

Besides a unified name space and scalability, distributed data management systems include advanced features such as various access mechanisms, advanced authorization and authentication support, and

policy enforcement via replication and rule features. In the following some important example will be introduced.

The integrated Rule-Oriented Data System (iRODS) [RMH⁺10, CCQ⁺11] is an open source distributed data management system. Its three distinct services are the iRODS Metadata Catalog (iCAT), the iCAT-Enabled Server (IES), and resource servers. iCAT is a relational database such as PostgreSQL. It handles the logical name space including the mapping to physical locations, metadata about files, user management, and iRODS deployment information. The IES is the main server of an iRODS installation, a so-called zone, that features the iCAT. Resource servers are used to integrate storage space while they can also increase the performance, safety, and resilience of a zone. The Rule Engine is an advanced feature that evaluates predefined rules at certain trigger events. A basic example is an automatic format conversion when a file is uploaded to an iRODS installation. Entire data workflows can be designed in order to automatically enforce potentially complex data policies. The iCAT enables to associate metadata triplets (key, value, unit) to files and collections to enable the association of meta information.

dCache [Fuh04, FG06] is a storage middleware that was initially developed in the context of the LHC computing system. Today, it handles a significant part of the worldwide LHC data [WLC15]. dCache provides a unified name space with the physical disk space being located on potentially many so-called pool servers. Tape archives can be transparently integrated and data transfers between all storage elements are done automatically. This way dCache transparently supports load balancing, replication, and is inherently resilient against hardware failures. A large set of protocols is supported for accessing data within dCache, e.g., NFS and WebDAV. It is nowadays used around the world [dCa15].

As a distributed file system, XtremFS [HCK⁺08, SBR12] was engineered with scalability in mind by separating raw data and the index that organizes the data. Object Storage Devices (OSDs) are installed on server and make attached storage capacity available for raw data storage. Metadata and Replica Catalogues (MRCs) manage the file name space and the mapping on which OSD specific data is located. The Directory Service (DIR) is a registry to enable service loop-ups by clients. As for clients, the FUSE client, for instance, provides POSIX compatibility [SKH⁺08]. Accesses via clients are translated to remote procedure calls that access functions on XtremFS backend services.

Within the computing middleware UNICORE [SEL⁺05, SBBR⁺10, BSP⁺ed], the Storage Management Service (SMS) abstracts from underlying storage systems such as POSIX or HDFS. The distributed Storage Management Service (dSMS) [RBB11] goes one step further and provides a single point of access to possibly many SMS instances. Thus, dSMS enables a common view and way of access to many independent storage elements with UNICORE.

Further examples include the Google File System [GGL03], Hadoop Distributed File System (HDFS) [SKRC10], Amazon Simple Storage Service (S3) [Sers3], and Windows Azure Storage [CWO⁺11].

2.2.3 Metadata Management

Metadata management entails the handling of data based on metadata, meaning information about the data, to enable easy access for future use. An easy discovery is essential for the usefulness of data. As metadata is usually closely linked with the data itself, a file reference is commonly attached. To create

metadata several steps are required. As first step, relevant information from data needs to be extracted. Second, metadata annotated or attached to the data itself. Third, in order to use metadata for search, it needs to be indexed. An high degree of automation in these steps is essential to enable usability and error resilience. The steps are described in the following paragraphs, depending on the kind of system that is used.

Through these steps, metadata facilitates the sustainability of data in the sense that it can be found again, even when the original creator becomes unavailable or forgets about the data. This findability is especially important when the data set is just one in a large institutional collection of data sets with possibly tens of millions of individual files. Furthermore, there is the major challenge of metadata standards with a huge variety existing as is detailed in the next paragraph. A general goal is having metadata management as integrated as possible [Sen04] with the specific overall data life cycle. This avoids the need to manually cross system boundaries and, thus, facilitates a seamless usability.

Metadata Standards

Metadata standards is a highly complex field with a large number existing [Lan11, Bal09] as can be seen in Figure 2.4 and 2.5. These standards differ in complexity, adaptability, domain specific concreteness, community, function, purpose, and strictness [Lan11]. Often, standards are either highly use case focused and easy-to-apply or very generic and complex to apply in use cases they were not designed for. A compromise is to use a common set of core metadata such as Dublin Core [WKLW98, MSF⁺10, EUD15b] together with metadata that is specific to the current use case. Various standards are presented in a detailed study [Lan11].

Metadata Extraction

Metadata extraction toolkits exist to extract specific information from a file with a specific format to be stored according to the metadata system in use.

Apache Tika [MZ11] is an open source Java metadata detecting and extraction toolkit as a top-level project under the umbrella of the Apache Software Foundation. Tika supports over a thousand different file types. The main component of Tika is its parser interface. It abstracts from supported formats and libraries and is easy-to-use yet powerful. Another major component of Tika is its content detection interface that provides through the common "detect" method. It takes a stream and metadata object with available information as input and returns a guess about the type in form of a MediaType object. Various kinds of detection such as mime magic, resource name based, known content type, default mime types, container aware, and the default one are available.

JHOVE (JSTOR/Harvard Object Validation Environment) [LPU15] is an open source Java framework that enables format validation, identification, and file characterization. Validation means that a file is given and checked if its specified format follows the format specifications. Conversely, identification takes a file as input and determines its format. Characterization returns properties of the file content. JHOVE features a GUI and commandline client and an API for integration with other applications.

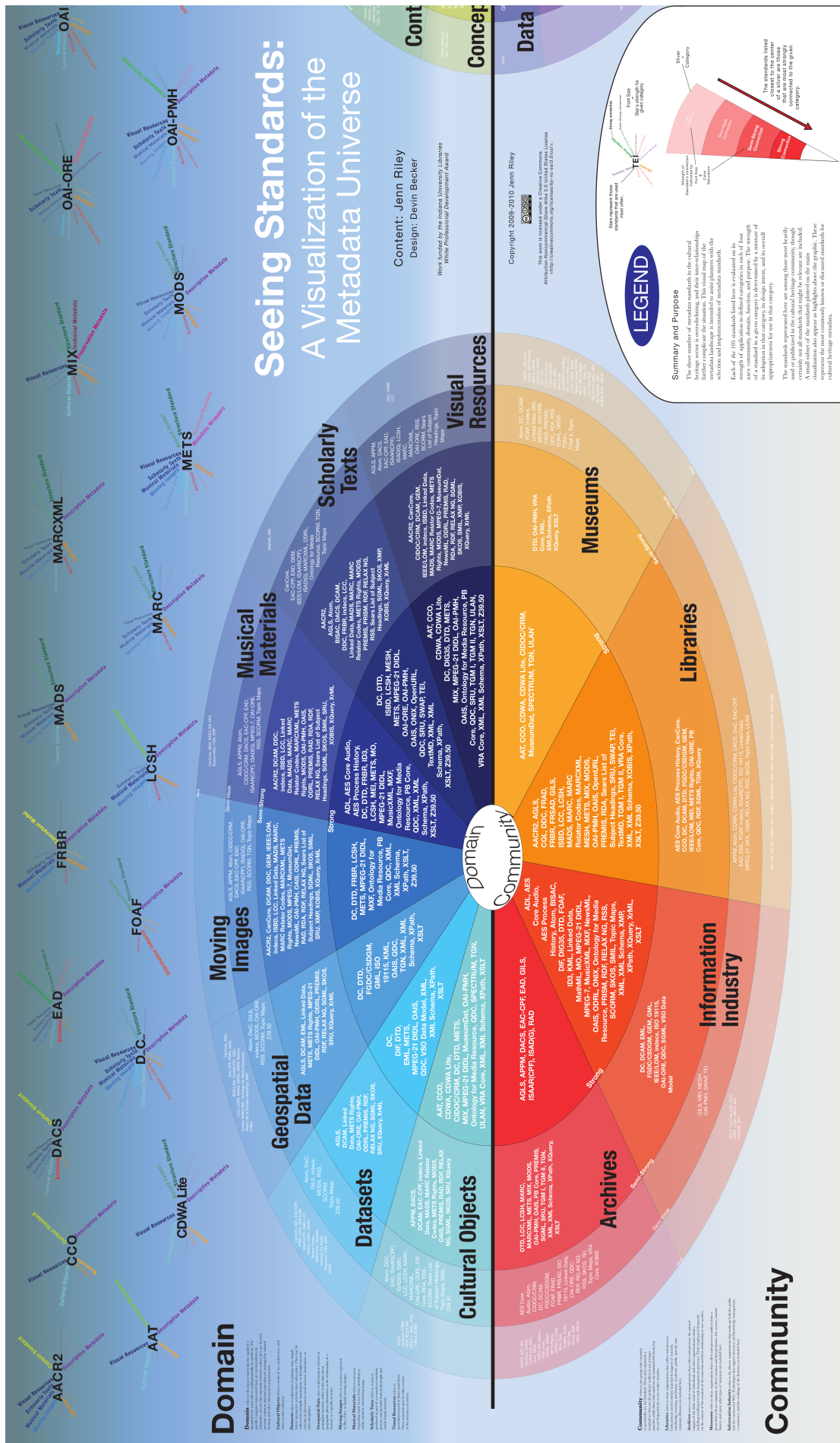


Figure 2.4: A visualization of complex metadata standards landscape is shown (left side) according to the categories of domain and community and the property of closeness to the respective category as distance from the center [Lan11].

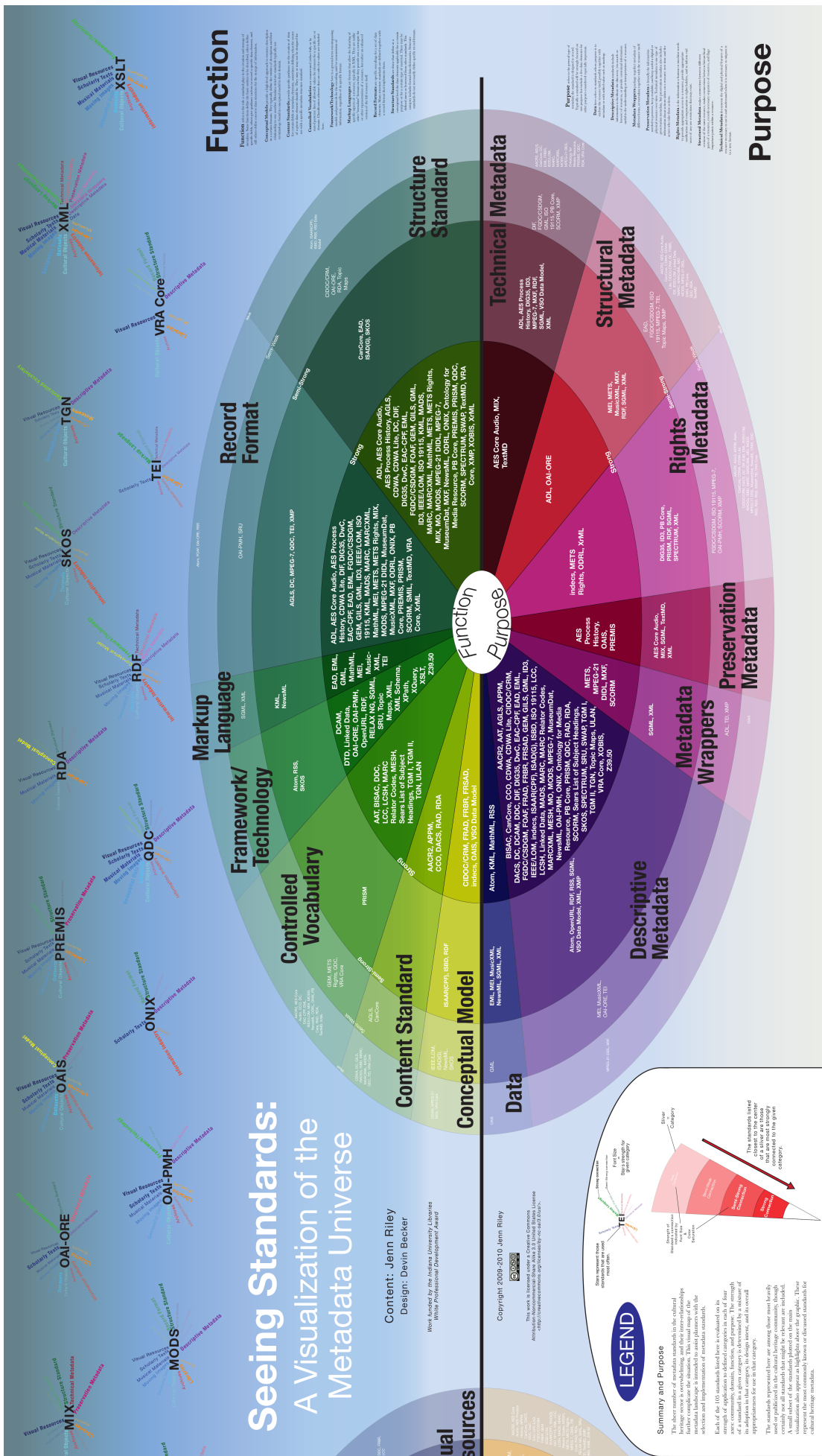


Figure 2.5: A visualization of complex metadata standards landscape is shown (right side) according to the categories of function and purpose and the property of closeness to the respective category as distance to the center [Lan11].

Supported formats include AIFF, ASCII, Bytestream, GIF, HTML, JPEG, JPEG 2000, PDF, TIFF, UTF-8, WAV, and XML. It is used by digital library applications and repositories. A completely rewritten successor called JHOVE2 [AMC09] is available. It addresses a number of feature requests based on user feedback. The requests involve improved performance, refactoring of the architecture and an API. JHOVE2 extends the functionality with respect to metadata extraction and includes modules for structured processes related to preservation workflows. It was funded by the US Library of Congress, the California Digital Library, Portico, and Stanford University.

Further general examples are ExifTool [Har15], NLNZ Metadata Extraction Tool [oNZ15], and C3Grid metadata tools [SBD13]. Specialized extraction tools are for example Taglib [Tag15] for audio formats and the Bio-Formats Importer plugin of Fiji [SACF⁺12]. This plugin enables to convert various proprietary formats in the microscopy domain to the OME [GAB⁺05] data model which includes metadata.

Centralized Metadata Catalogues

Centralized metadata catalogs are systems that store metadata and a search index together in a single system with references to files located somewhere else. This approach implicitly offers a consistent and uniform view on data with seamless search capabilities. One disadvantage are potential bottlenecks when all metadata capabilities are concentrated in one system. The approach limits the resiliency as it represents a single point of failure. As metadata is stored separately from data, another downside is that archiving is challenging. Data without metadata tends to be useless. Thus, metadata in centralized systems needs to be archived as well. When the referenced data is archived, it might be possible to continuously maintain a metadata system. But this is unfeasible as it requires a lot of effort in the long term. Furthermore, with several metadata systems this approach would significantly increase the required backup effort even more.

A solution to the maintainability challenge would be to choose one metadata system in the expectation that it is widely enough used for a high chance of it being maintained by the developers in the long-run. Examples of such systems are DSpace [SBB⁺03] and Fedora Commons [Fed16]. These systems called repositories and integrate a advanced functionality and are used for long-term archiving of data with metadata as the central functionality for data organization. Fedora Commons provides common core services to be integrated with existing environments while DSpace offers a system with a complete set of features to be deployed as a whole. Fedora and DSpace joined their organizations in 2009 and formed the none-profit DuraSpace [Dur13] organization to manage and facilitate the development of both and realize synergy effects.

iRODS [RMH⁺10, CCQ⁺11] fits mainly in the data management category as described in Section 2.2.2. It provides some metadata capabilities with attribute-value-unit triplets being stored in the central iRODS database and search capabilities. The AMGA Metadata Service [KSP08] is a specialized development within the WLCG [Shi07, BF07] context. ICAT [MSF⁺10] with its central database featuring the Core Scientific Metadata Model [MSF⁺10] aims at providing an interface to access photon and neutron physics facility specific life cycles. ISOcat [KSWWW08] provides a data category registry for linguistic resources with a web interface and various APIs being available.

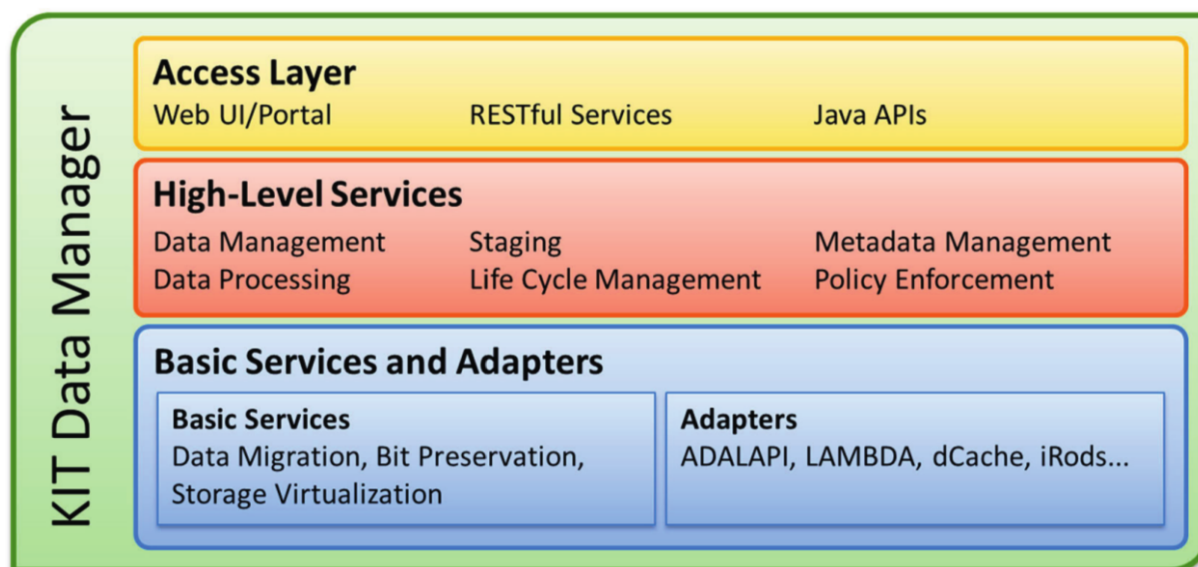


Figure 2.6: An overall view [JVK⁺14] of the KIT data manager architecture is depicted with metadata management being a core service.

The KIT Data Manager (KDM) [JVK⁺14, PSJ⁺ed] is a research data repository with metadata management at its core to deal with digital data objects and their life cycle. Use cases include managing synchrotron tomography images that can potentially range into petabyte scale and handling a large variety of images in the digital humanities domain. The KIT Data Manager consists of a set of basic and high-level services that seamlessly integrate with each other (see Figure 2.6 for an architecture overview). Access is offered via REST and Java APIs for developing solutions and a Web Portal for serving the user communities. Basic services are for example for data migration, bit preservation, and storage visualization. Data management, staging, and policy enforcement are examples for high-level services. A search interface can be utilized to find annotated digital data objects. Execution computing tasks is supported via the LSDF Execution Framework for Data Intensive Applications (LAMBDA) [GBH⁺11, JHS⁺12]. An aim of the MASi project (see Section 6.2) is to extend KDM with a generic and widely applicable High Performance Computing integration via UNICORE. Another aim is to add support for federated identities.

EUDAT [LWE⁺13, Mal14] is a large European data management project. It aims at creating a generic infrastructure to access and preserve research data (see Figure 2.7 for an EUDAT overview). The metadata related services B2SHARE and B2FIND are of particular interest here. B2SHARE aims to be a service to store and share small research data sets via a web portal that is also the sole mean to upload data. Metadata fields have to be filled in manually. Basic fields are the same across all domains while specific fields are offered for individual domains or projects. B2FIND is a service for finding research data sets via its metadata. It also allows to comment on metadata and resources. Via the OAI-PMH [VdSNLW04] protocol metadata is regularly gathered from integrated metadata providers to be made available for search. The respective communities decide which metadata is to be published. A new service for semantic annotation called B2NOTE is planned. It aims to enable automatic metadata annotation of ingested data [EUD15a].

The Agave science gateway API [DVS⁺12] supports metadata capabilities. Metadata can be automat-

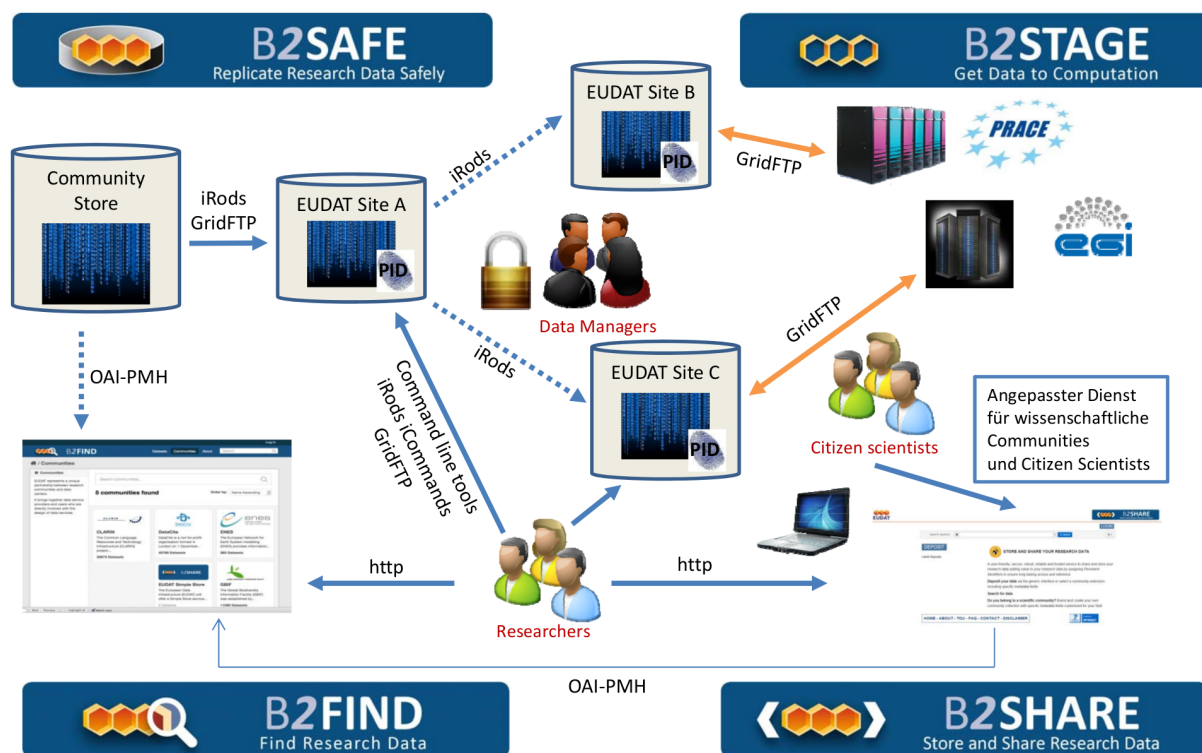


Figure 2.7: A general overview [Mal14] of the EUDAT architecture is shown, including the B2SHARE and B2FIND services that involve generic but limited metadata capabilities.

ically extracted by employed external extraction tools via its rule-based processing feature. The metadata itself is managed via an integrated unstructured column-oriented NoSQL database. Agave supports metadata schemas as XML or JSON descriptions [NPRI09]. The schemas may form a series of tuples or highly structured descriptions. The system can be adapted to specific use cases by defining and inter-linked schemas via a registry. The science gateway API supports global search capabilities.

DIRAC as a computing middleware (see Section 2.3.2) offers the Dirac File Catalogue (DFC) [TP12] as a way to organize files and their metadata. DIRAC is based on experiences of the LHCb collaboration. Metadata can be associated to files and directories as key-value pairs. Metadata is inherited to sub-directories. Search and access operations are possible via a Python API, commandline client, and web portal.

Other examples are LOCKSS [MRG⁺05], MyCoRe [MyC13], Eprints [Epr13], UrMEL [UrM13], OPUS [OPU13, SK09], Kopal [Kop13], and the Rosetta Project [Pro13].

Systems with Metadata in Close Proximity to Data

The opposite approach to the centralized one in the previous Section 2.2.3 is that of storing the metadata in close proximity to the data itself. Metadata is kept in either separate files, with the example of JSON [Cro06], or kept in the same files besides the original data with the examples of HDF5 [FCY99, CYCA06, FHK⁺11] or (p)NetCDF [RD90, LLC⁺03].

The Hierarchical Data Format (HDF) in its fifth version is a structured container format. It aims at

providing access to scientific data in an architecture independent, structured, and high-performant way at a large scale with support for a large variety of data types. Within HDF5, data is stored in objects and objects are organized in a hierarchical structure. Objects can be arbitrary datasets with types such as images, arrays, and others. Objects can have properties associated to it such as size, content or lists of datasets. The hierarchy is organized top-down starting with a root element. Objects can also include links to other objects or HDF5 files. An HDF5 library is available to access specific parts of files and enable various other operations. Examples use cases are in the geosciences and next-generation DNA sequencing. HDF5 has a structure that is well-defined within the file itself. This information can be extracted and utilized as metadata.

The Network Common Data Format (NetCDF) is a self-describing data format for multidimensional array-oriented scientific data. Data is stored in coordinate systems with multiple dimensions. These coordinate systems are accessed via variables and annotated with properties. The number of dimensions is the same across variables while the properties can be different. When the file structure is honored, data can be appended and read later via the serial NetCDF interface. This interface can randomly access any part of a data set stored in a NetCDF file. Common use cases are within the climate and oceanography research domains. The extension parallel NetCDF (pNetCDF) enables random parallel access to data sets in NetCDF files to avoid bottlenecks in parallel applications while providing backward compatibility.

A main advantage of this close proximity approach is that potential inconsistencies can be more easily avoided. Such inconsistencies are dangling references when metadata exists but the data is already deleted and dark data where the data exists but the metadata with the references is already gone. This approach is inherently more distributed and is, thus, more failure-resistant and scalable. From the point of view of the annotated data, it is independent from remote systems. The approach is also potentially more universal since metadata can be stored in arbitrary formats. Archiving data together with the metadata is straight forward as the same archiving operations and procedures can be applied for both and are independent of external services. The disadvantage is that a search index still needs to be build and provided by a central component. Also, only with a central search index a uniform data view can be provided and the number of stored files might increase dramatically when data and metadata are kept in separate files.

Systems with a Combined Proximity Approach

Both previous approaches in Section 2.2.3 and 2.2.3 are mixed in this combined proximity approach. Metadata is stored in close proximity to the data either in a separate file or in the original file as described in the previous Section 2.2.3. At the same time, capabilities are integrated for the creation of an index to enable the search for data via metadata. The advantages of both approaches are combined. The disadvantages being that the consistency between the distributed metadata and the central index needs to be managed to provide a suitable level of currentness of the index. Also, this approach is more complex.

An example system is the metadata service of the UNICORE computing middleware [NS10] that enables to organize data via metadata. Its metadata service (see Figure 2.8) is integrated with the UNICORE Storage Management service (SMS) and access available via the UNICORE commandline client, workbench, portal, and Java API. Metadata can be extracted from arbitrary data formats by the Apache Tika

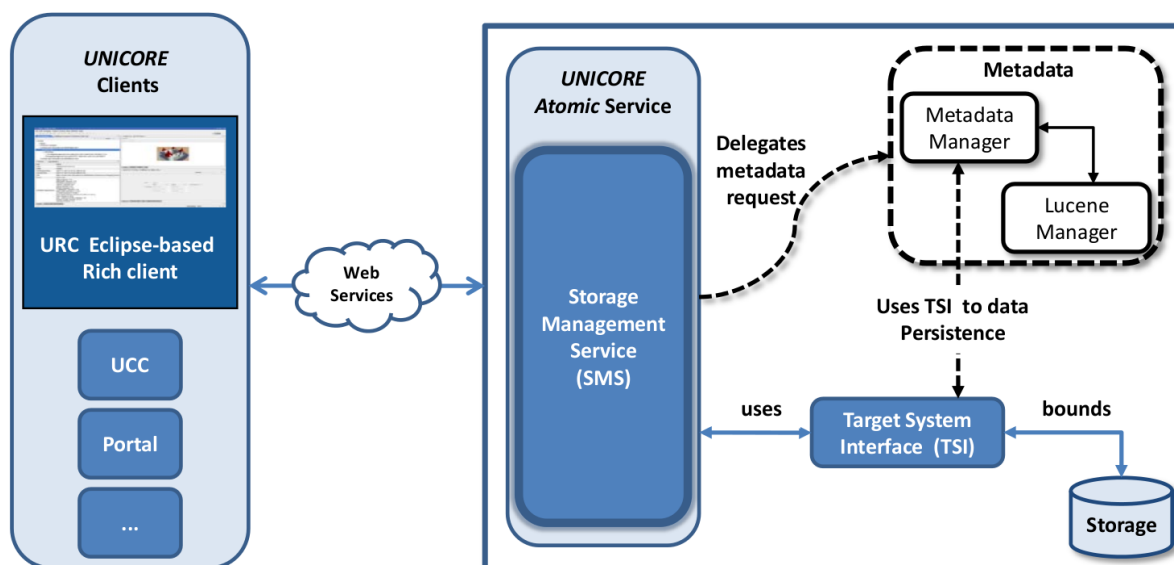


Figure 2.8: The architecture of the UNICORE metadata service is shown [NS10].

framework (see Section 2.2.3) and it is stored besides the data in JSON files. The data itself is stored in local or remote directories or in data management systems that are available via the UNICORE Storage Management Service. The Apache Lucene framework is utilized to automatically build an index and provide powerful search features within the UNICORE ecosystem. This approach was chosen for its high adaptability, it being open source and part of UNICORE to be part of the dissertation research presented here.

2.3 Applying Computing in Data Life Cycles

A central aspect in a life cycle of digital data is to apply computing to it (see Figure 2.9 for an overall classification). Data is transformed to data of higher value or information is extracted for knowledge gain with e.g., simulations that aim at transforming a model and input parameters to insightful results or image analysis that applies an algorithm with parameters to images to extract information. Other examples are the processing of data in order to prepare it for visualization or to extract information for human-readable statistics. The complexity of computing resources is introduced in Section 2.3.1. To manage this complexity, middlewares abstract from computing resources [HTT09, TTH11]. Computing middlewares are utilized to abstract from raw computing devices and hide their complexity (see Section 2.3.2). Workflow middlewares further abstract to automate both tedious manual and highly complex tasks in order to enable high usability and error resistance (see Section 2.3.3). These systems need to be tightly integrated with authentication and authorization infrastructures to enable seamless usage of all components of data life cycles. As coming exascale systems are predicted to be even more complex [JMPK⁺15, KBB⁺08], it is even more important to hide their complexity in order to enable end users to make efficient use of them. Displaying specifics of underlying systems should be strictly optional.

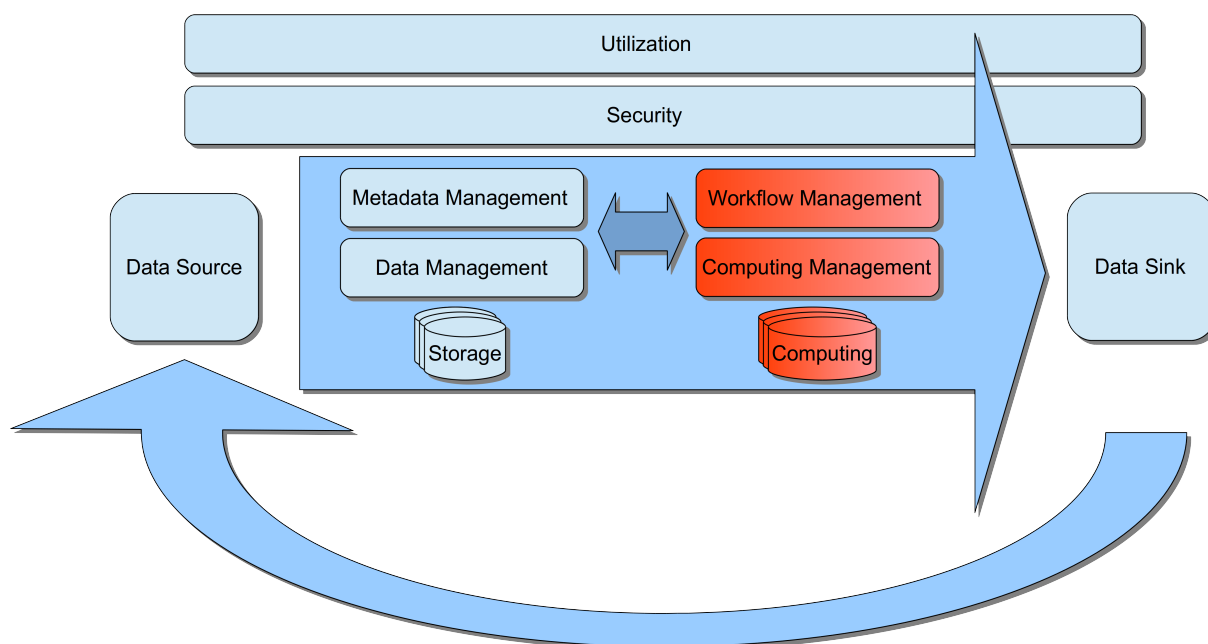


Figure 2.9: Data life cycle management component categories for executing computing tasks (computing pillar) with three layers of abstraction are shown; computing, computing management, and workflow management.

2.3.1 Computing

Computing devices in this context are defined as High Performance Computing and cluster resources for arbitrary computing task execution. These resources are highly complex in themselves with components such as CPUs, RAM, storage, operating systems, racks, nodes, and interconnects. This complexity is partly managed via batchsystems which local users utilize to distribute their tasks to the underlying compute resources of which they only have to know overall information and not infrastructure specifics anymore. Batch systems in themselves are a highly diverse and a complex class of systems. Widely used examples are SLURM [YJG03], LSF [Zho92], Oracle Grid Engine [Gen01], CCS [KR98], PBS [Hen95], and TORQUE [Sta06]. A new and advanced example is Apache Mesos [HKZ⁺11].

2.3.2 Computing Management

Batchsystems are designed for being used locally with remote access usually only available by extra means such as SSH. To mitigate these restrictions and further facilitate usability, computing middlewares are utilized to form an abstraction layer on top of batchsystems. These middlewares hide specifics of different batchsystems, provide access to virtualized local and distributed computing resources and provides staging mechanisms for local and remote data. Besides hiding various High Performance Computing and cluster specific details, integration capabilities with authentication and authorization infrastructures are provided for flexible user access. Access possibilities are manifold and described in Section 2.5. A unique method of abstraction exists in the UNICORE computing middleware. Its Incarnation Database (IDB) is a way to abstract from applications to hide cluster specific details such as installation paths and required modules. From a UNICORE-external point of view only the application name and its version

has to be known. By just stating this information in a job description, UNICORE is enabled to automatically decide where to execute a received task. In the following, computing middlewares are categorized with respect to how computing tasks are initiated.

Task-driven

Technologies that enable the initiation of individual tasks directly by users or components are termed task-driven here. Examples are included in UNICORE [SEL⁺05, SBBR⁺10, BSP⁺ed], Globus Toolkit [FK97, Fos05], gLite [LEP⁺06], ARC [EGK⁺07], and DIRAC [CGP⁺10]. Also Cloud and Big Data technologies are relevant here such as the Amazon Elastic Compute Cloud (EC2) [Serc2], the Windows Azure Platform [Jen10], OpenStack [WGL⁺12], OpenNebula [WGL⁺12], Apache Hadoop [Whi12], Apache Spark [ZCF⁺10], and Apache Hive [TSJ⁺10].

Workflow-driven

In this category computing tasks are organized in workflows. When initiated, workflows are enacted by workflow engines. These in turn submit the individual workflow tasks to computing resources. This topic, including examples, is presented in Section 2.3.3 in detail.

Data-driven

Data-driven systems are characterized by the capability that actions are triggered in reaction to data management events based on pre-defined rules. For example, the action might be the format conversion from jpeg to png and the trigger might be the appearance of a file in a specific location. Further functionality, depending of the system, might include the execution of local script, remote tasks, and even complex workflows. The iRODS [RMH⁺10, CCQ⁺11] data management middleware supports this approach. Rules and actions, also called microservices, can be defined in a flexible way with various trigger points being available. The downside is that the execution of actions is limited to the storage server itself. The same applies to the rule-based processing capabilities of the Agave science gateway API [DVS⁺12]. Both do not natively support outsourcing of computing tasks to external computing resources such as High Performance Computing resources and clusters. In contrast, UNICORE with its data oriented processing approach [SGG13], co-designed by the author, as a computing middleware natively supports most computing resources. Actions are inherently scalable as they can be executed on High Performance Computing resources as highly parallel tasks. In contrast to iRODS, a minor downside is that the trigger events defined in rules are limited to files appearing in an observed directory. This data-driven approach significantly advances the respective solutions by enabling new data processing paradigms.

2.3.3 Workflow Management

Workflows are a way to formalize expert knowledge and make it available to solve complex scientific challenges and support reproducible science. They can be visualized as graphs of tasks. These are usu-

ally represented as nodes and directed edges symbolize control and data flows from one task to another. Tasks may be executed sequentially or in parallel with respect to each other. Individual tasks themselves may also be single core jobs or highly parallel applications using potentially a large number of cores. The individual tasks of such workflows are enacted via workflow engines on computing resources. Workflow management systems are a further abstraction layer on top of computing management middlewares providing a higher-level functionality. Highly complex and tedious manual tasks can be automated and executed as a whole.

Workflow management capabilities are provided via the UNICORE Workflow Engine [SDM⁺08], the workflow engine of Grid and Cloud User Support Environment (gUSE) [FK11], Pegasus [DVJ⁺14], KNIME [BCD⁺08, BMW⁺13], Galaxy [GNT⁺10], Swift [WAW⁺13, MRK⁺13], Taverna [WHF⁺13], and Kepler [LAB⁺06].

UNICORE provides advanced workflow management features in its workbench and web-based portal. The workflow management is natively integrated with UNICORE-enabled infrastructures. The fundamental two services are the workflow engine and the service orchestrator. The workflow engine accepts complete workflows as input, processes them and forwards individual workflow tasks to the service orchestrator. Which in turn submits these tasks to appropriate target systems, monitors their execution and sends log messages back to the workflow engine. The workflow editor provides constructs such as if-then-statements to steer the control flow and loops for repetition of individual workflow parts. The workflow engine also enables parameter and file sweeps inside workflows.

gUSE incorporates an abstract and concrete workflow concept. The abstract concept includes a yet to be configured workflow and is represented as a directed graph that is created and edited via the gUSE graphical workflow editor. Nodes can be added to a workflow and input and output ports attached. These ports can be connected to each other in order to represent the data flows. In order to add information about jobs and data, a further step is required. The concrete workflow needs to be created by configuring the nodes with information about target computing system, job, and input data. Previously these were completely separate steps, now they were unified by a collaboration initiated by the author for much higher usability and efficiency [MAG⁺15].

As gUSE does, Pegasus enables integrated access to various computing and data infrastructures. It offers a prototype web-based user interface based on Triana [TSWH07]. KNIME is a workbench-based workflow environment. It is widely used in pharmaceutical and biological research settings. Its approach is convenient and intuitive. Pre-defined nodes serve very specific functions that can be selected and connected with each other. Nodes can be newly developed in the same user interface as the workflows are formed. Parts of workflows can be easily tested including data and dependencies. A downside is that the KNIME workbench is hardly HPC-enabled. An High Performance Computing integration is possible in a narrow way with the KNIME server. It is limited to the Oracle Grid Engine as batchsystem, user logins have to be the same across all involved systems and a common distributed data management system has to be present. A further approach to integrate KNIME with HPC is based on the conversion of KNIME to gUSE workflows [dIGKS⁺13]. To create a generic and transparent High Performance Computing integration, a collaboration led by the author, develops a highly intuitive way to execute KNIME workflows on HPC resources. Users just export a workflow into a specific directory that is

monitored by UNICORE via its data oriented processing feature (see Section 2.3.2). UNICORE then transparently spawns multiple workflow instances and distributes them to various cluster nodes to be executed in parallel.

Via its web-based interface, Galaxy [GNT⁺10] offers a toolbox for users to create and execute workflows. It is widely used in the biomedical context. Galaxy's downside of limited data management capabilities can be mitigated via Swift as a parallelized enactment engine. Taverna, also widely applied in the biomedical community, is a workbench with the possibility to share workflows via the social network and workflow repository myExperiment [GBA⁺10]. Kepler offers feature-rich workflow management capabilities via its workbench. The web-based graphical interface can be used in order to upload, modify, and execute workflows. These can be nested, executed on a diverse set of computing resources and data gathered from distinct sources.

The Dispel workflow language [ALG⁺12] has streaming data processing as its target. The core of Dispel are processing elements (PEs). These logical units define computation that is applied to data streams and can be organized in packages. PEs can be in abstract or concrete form and also individually combined into packages. A Dispel workflow is a logical representation of data streams passing through PEs. These modify streams to varying degrees. Such workflows are parsed and enacted by processing services called Dispel gateways with OGSA-DAI [AAB⁺05] being an example. Dispel allows to design workflows in a data focused approach in contrast to the computing focused approaches mentioned earlier. Currently, Dispel is focused on local computing resources and not High Performance Computing resources. Though research is done to integrate Dispel with High Performance Computing resources [CKL⁺13].

Other examples include the Grid Workflow Execution Service (GWES) [Hoh06, DGST09], the Business Information Systems in Grid (BIS-Grid) [HSG09], the Meta Scheduling Service (MSS) [EWW⁺07], and the Workflow Scheduling Service (WSS) [GP09].

2.4 Security

In this chapter, security aspects in data life cycles are introduced along the topics of authentication, authorization, and single sign-on (see Figure 2.10). These three aspects are supported by the novel identity management service Unity (UNified identiTY management) [Uni15]. Unity is utilized by the national grid infrastructures of Germany (NGI-DE) and Poland (PLGrid), the European data infrastructure EUDAT, and the European Human Brain Project, beside others.

2.4.1 Authentication

A basic requirement in data life cycles is to ensure a person requesting access is really the person he indicates to be, a principle called authentication. A current approach is to utilize personal X.509 certificates [SHF02]. These are strongly linked to a person and, thus, should not leave the computer of the owner. Around the world, users can apply for X.509 certificates at recognized institutions called certification authorities, such as DFN [DFN15b] and GridKa [Gri15] in Germany. Usually, the possibility to get a certificate is limited to members of research institutions of the respective country. An online

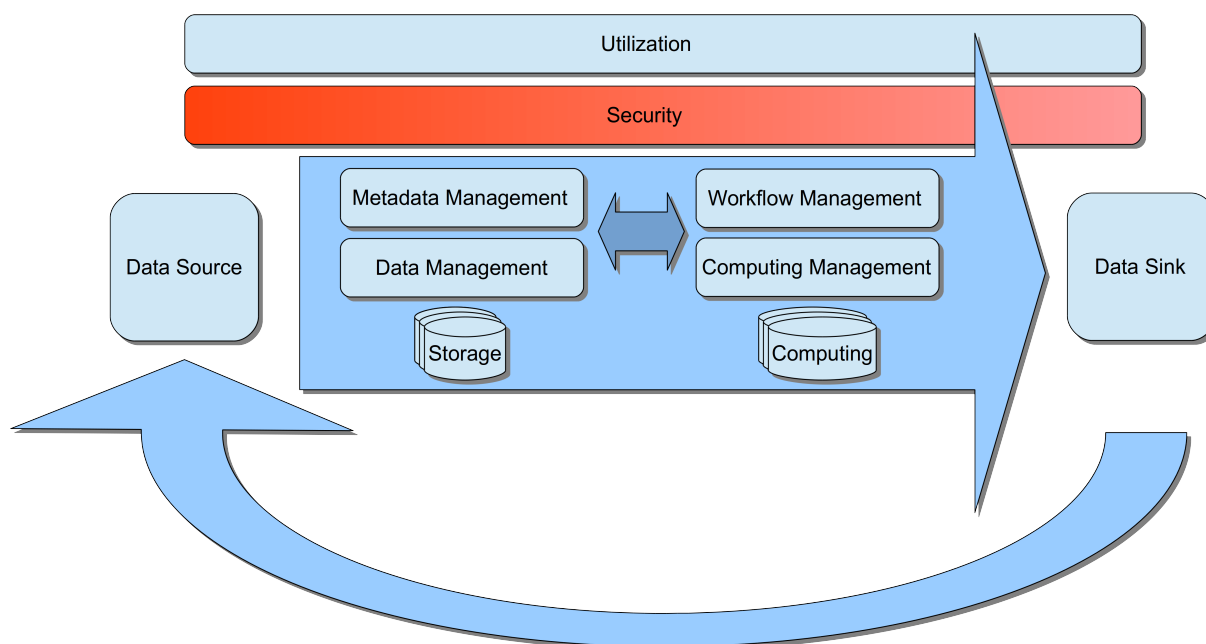


Figure 2.10: Security components enable the security and seamless integration of the data life cycle parts.

form needs to be filled, printed, signed, and brought in person to an official registration authority. The registration authority checks and certifies the users' ID or passport as prove of identity. Subsequently, the application is approved and enables the user to get a certificate.

Here, a current research and development direction is to create overarching identity management systems [LOP04]. These utilize trusted identity sources with varying degrees of trustworthiness. On one end of the spectrum, there are, for example, universities where students have to identify themselves via their passport or ID card. This closely connects the real identity and the digital one. On the other end of the spectrum are systems that might only require a valid e-mail address to register. DFN-AAI [DFN15a] is a German academic federation of Shibboleth based identity providers with a high trustworthiness. The European eduGAIN [Pro15] service aims to federate individual national federations such as DFN-AAI. The Unity service is able to act as an identity proxy between arbitrary identity providers with technologies such as LDAP [HSG03], Active Directory [LND00], Shibboleth [MCC⁺04], OpenID [RR06], and OAuth [Har12]. Services are enabled to consume information from Unity via interfaces such as SAML POST, SAML SOAP, and OpenID with further ones being planned. Commercial identity providers with less trustworthiness are Facebook [Fac15] and LinkedIn [Lin15]. They offer to use user accounts to authenticate within other services.

2.4.2 Authorization

After the identity of a user is established, it needs to be specified what a user is allowed to do, also called authorization.

Virtual organizations (VOs) are a flexible way for providers to manage usage of their resources with respect to remote users. A VO is a collection of resource providers and users with the common goal of advancing a specific scientific field. A VO solves the challenge of resource providers having to

deal with potentially thousands of users and users with potentially dozens of providers. Both users and providers simply join a VO which goal they support in order to consume or provide resources respectively via a VO as intermediary entity. A common solution is the Virtual Organization Membership Service (VOMS) [ACC⁺04]. It allows for users to register by using certificates to become member. Resource providers also become members and they can subsequently download information about all users to enables these to use the resources. This way both users and providers only have to interact with the VOMS server. For example, Unity can provide authorization information via its group capabilities. In turn, UNICORE installations on High Performance Computing resources are configured to utilize the Unity service as a source of authorization information.

2.4.3 Single Sign-on

Single sign-on (SSO) is the essential concept that a user only needs to authenticate once in order to access complex infrastructures with potentially various underlying systems. Without single sign-on, for every access to further systems in a complex distributed infrastructure the user would have to re-authenticate which is hardly feasible. Single sign-on is enabled by the so-called trust delegation, where services access other services impersonating the original user in a controlled and secure way. Thus, the intermediate system has the same access rights on the target system than the user itself would have if he would directly access the target system. This trust delegation is enabled by short-lived authentication token. With this concept the users avoid manual and tedious re-authentications while accessing underlying systems. Important examples are SAML (Security Assertion Markup Language) [RHP⁺08] trust delegation assertions and GSI (Grid Security Infrastructure) proxy certificates [FKTT98].

The basic idea of a SAML assertion is to specify who (subject) is allowed to do something in the name of the user and on which resource (target). Both, the distinguished names (DNs) of the subject and target have to be specified together with a validity period. The information is encoded in the assertion and cryptographically secured via public key cryptography [FS03]. The advantage of such a SAML assertion is that it can be tightly controlled. An entity that received trust from a user, is only able to perform the exactly specified activity. This limits potential security issues in case an assertion is lost. For example, a SAML assertion can not be used to create a further assertion. The Unity service directly supports issuing SAML assertion based on already existing user accounts.

Another commonly used technology are GSI proxy certificates. These may be created by using a X.509 public key certificate in order to delegate trust to another entity. This delegation can be done dynamically without access to a third party service. Such a proxy certificate can be utilized the same way as a end user certificate would be used. Proxy certificates can be utilized to create further proxy certificates which limits their security properties. Numerous computing and data middlewares support proxy certificates.

2.5 Data Life Cycle Utilization

Overarching the underlying layers is the utilization of and within data life cycles (Figure 2.11). As this layer also represents the utilization of users with data life cycles and their components, they are the

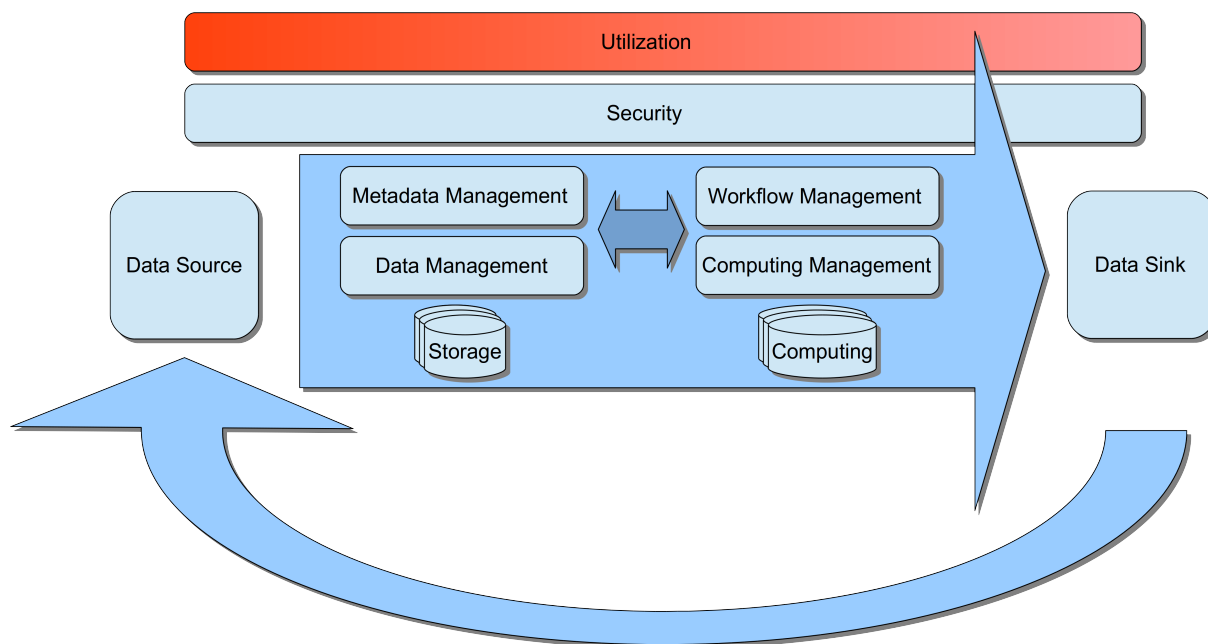


Figure 2.11: On top of the underlying layers the utilization is the essential point of entry to enable user and systems to interact with data life cycles.

most important ones from the users' perspective. They determine how productive and satisfied users are going to be while utilizing the data life cycles for their research challenges. Data life cycles are and will continuously become more complex [KBB⁺08, dI11]. This trend results from the fact that they are built upon underlying components that will in turn be increasingly complex. In order to maintain and increase the usability of data life cycles, this complexity needs to be abstracted from and, thus, hidden from the users [GKG⁺15, HTT09, TTH11]. The overall goal is to provide environments that are highly integrated to enable scientists to easily perform common tasks such as ingesting, access, downloading, processing, and archiving data. To enable a hassle-free traversal through the layers, the whole infrastructure optimally has to provide single sign-on capabilities (see Section 2.4.3). The goal is to have existing institute logins enabled to be re-used instead of user certificates that are highly complicated for users. Then, users only need to authenticate once and are then enabled to use the underlying services without re-authentication.

Flexibility versus usability is a goal of conflicts that the components in this chapter inherently possess. Either user environments are generic to a high degree which supports re-usability or they are highly usable but very use case specific. Optimal is the middle way of a generic framework providing a lot of common functionality that can be adapted to further use cases with limited effort. Also important is a high degree of seamlessness to avoid hurdles of users having to manually overcome system boundaries. Tolle et al [TTH11] describe their vision of NUI (Natural User Interfaces) as the way to advance science in general by making computing infrastructures and computers in general completely transparent by integrating them into the user's working environment to be not a hurdle but purely an enabling force. They illustrate the vision by well chosen examples in a future working environment, the multitude uses of oceanic data and the distributed effort in finding a cure for Alzheimer's disease.

In the report *Riding the Wave* [Eur10] to the European Commission the *High Level Expert Group on Scientific Data* describes in detail the essential criteria for dealing with Big Data and how to enable its

use in science. One is the necessity for seamless access to, use of, and trust in data in a way that the infrastructure becomes invisible and the data itself the asset. Big Data is changing the way science in general is done with entirely new possibilities. This seamlessness is to be enabled by fostering virtual research environments. These are to enable the integrated finding, accessing, and processing of individually relevant data with a transparent degree of trust in the methods and infrastructure. Another criteria is that the arising concepts need to be generic in nature to be adaptable to new technologies. The goal is to make transitions of underlying technology invisible to users. Furthermore, a key aspect is interoperability between systems. Thereby, easy availability of data and services reduces barriers of use and significantly fosters scientific innovation.

The strategy paper *Gesamtkonzept für die Informationsinfrastruktur in Deutschland* [dI11] deals with the overall direction of information infrastructures with a special focus on the user point of view. Generic solutions, standards, and best practices are especially important as well as easy-to-use access. For these characteristics to be facilitated, interfaces should be open to maintain interoperability and enable a high sustainability. Intelligent search and visualization mechanisms have to be provided in concert with easy-to-use interfaces between data formats used in research data repositories. Virtual research environments (VRE) are seen as a central field of activity. The core statements are; VREs should be as simple and easy-to-use as possible while at the same time providing sophisticated access to various information types in distributed systems. The environments should also be close to research and developed by IT experts in close cooperation with domain specialists. In order to facilitate a high usability, existing systems should be re-used as much as possible. VREs should be universally applicable and are envisioned to be widely used by 2020 in all research disciplines while maintaining research field diversity. The sustainability should be ensured by organizational and technical means. Whole data life cycles, from data creation over analysis and processing to finally publication, teaching, and archiving should be supported by modular systems, flexible configurations, standardized interfaces, and possibilities for seamless information exchanges. It is of highest importance to enable frictionless access to every part of the life cycle.

VREs are an important component for any compute center to provide services for the full data life cycle. These services contain manual SSH access as basic and VREs as advanced mean of access. The integration of VREs with currently used environments utilized by advanced users are of the utmost importance as well as graphical access for all offered services for non-IT-affine communities. Collaborative service such as Wikis, mailing list, data exchange systems, forums, event organization tools and source code management systems should be directly offered in an integrated way. Thereby, user groups with varied ranges of expertise can be supported and attended upon. The tight integration with federated identity infrastructures is the center piece in order to enable a high usability as then already existing and trusted login credentials can be transparently used. Generic metadata and workflow management services as high level functionality should be offered for the highest possible efficiency while working with data and related computing tasks. The importance of VREs is emphasized by the fact that in the large US XSEDE research infrastructure more users are utilizing science gateway than command line clients [LZWD⁺15].

In this section, utilization components are put into five categories with an increasing focus on users and a decreasing one on system integration. As data life cycles are usually comprised of many components, the first section about APIs (2.5.1) and partly the second one about commandline clients (2.5.2) will cover major aspects of building up such complex data life cycles. APIs are fully and commandline clients

partly meant to be used for integration with other systems. The commandline clients in Section 2.5.2 may also be used by IT-affine users. In sections 2.5.3 (workbenches), 2.5.4 (science gateways) and 2.5.5 (visualization methods) the components are of lower flexibility but higher usability and, thus, meant for all users, including users with a low level of IT-affinity.

2.5.1 API-based

In this context, Application programming interfaces (APIs) and libraries are mainly utilized to either build up the data life cycle itself or to access it by external systems. These can, for example, be existing systems that users utilize for their research. Naturally, APIs are made use of by developers and at most by advanced users that may wish to extend their individual research environment [GDP⁺15]. Example frameworks are Apache Airavata [MGH⁺11, PMG⁺14] and Agave API [DVS⁺12]. Both can be utilized to develop complete science gateways from scratch in various programming languages with minimal developer effort due to the fact that common functionalities are provided.

Apache Airavata is a free, open source, and an open community project offering workflow management capabilities for a variety of computing resources. It provides pluggable components such as a service registry, a job orchestrator, a workflow interpreter, a component for connections to remote resources, for a credential store [KMBP14] and a messaging system (see Figure 2.10 for the architecture). Airavata is developed with extensibility in mind in order to enable a high flexibility to facilitate the integration of novel approaches. It can be deployed either as a single instance or in an environment where various components are running on distributed resources in order to provide scalability and resiliency. Airavata is designed to support workflows, to be freely integrated into potentially existing environments and to expose all features via its API.

Agave is an open source solution providing RESTful web services to build science gateways. It provides the following features and support for technologies:

- advanced authentication and authorization infrastructures,
- application discovery and monitoring,
- data and computing management systems,
- metadata and workflow management,
- policy-based data handling and analytics,
- reporting and publication.

Agave is developed in projects such as the US iPlant Cyberinfrastructure [GVM⁺11] and the Canadian iReceptor [iRe16]. In mid 2015 Agave is utilized by a number of science gateways around the world serving multiple disciplines. To enable service deployment Docker images are provided.

Another example is the VineToolkit as a modular and extensible Java library to build distributed applications. Vine supports a number of middlewares and is used in several projects besides the polish national PL-Grid infrastructure. Furthermore, the existing Java and new REST API [SBR14] of the important

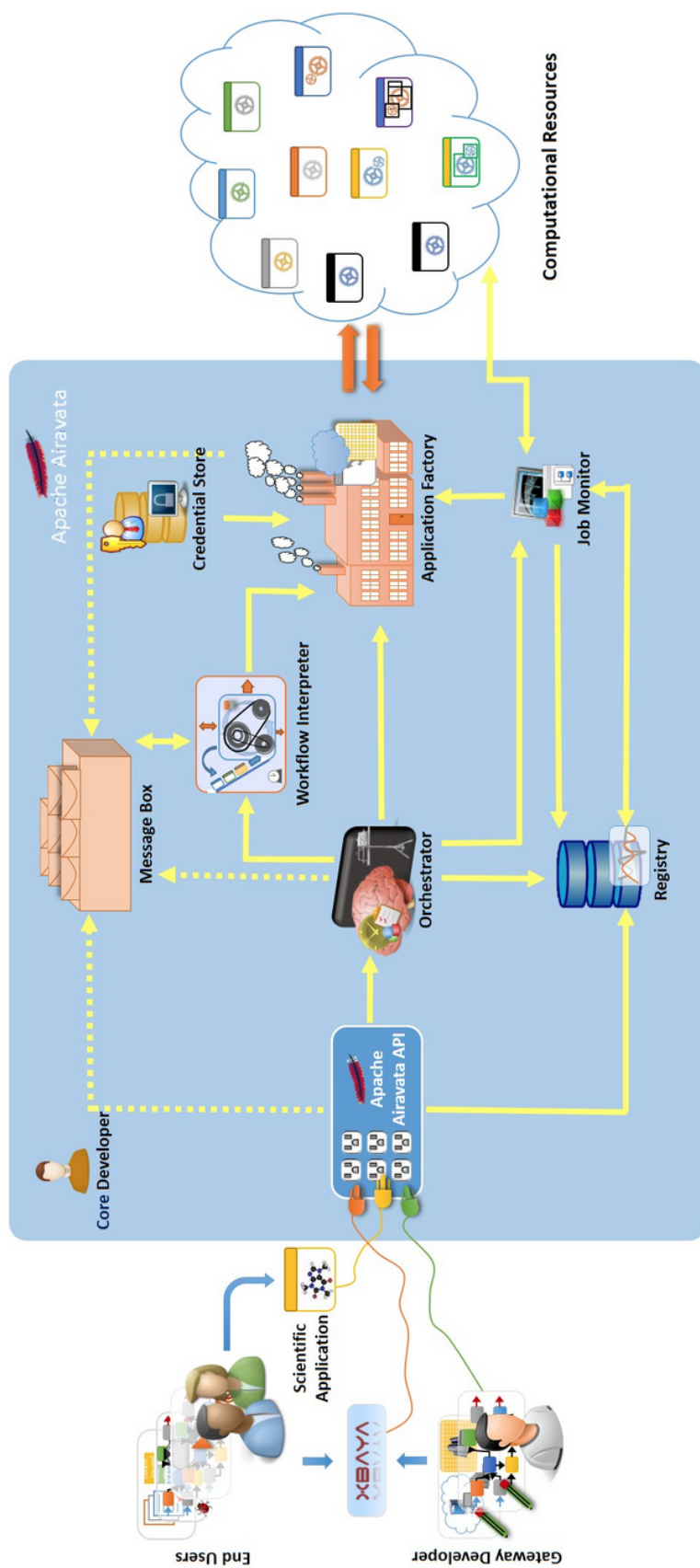
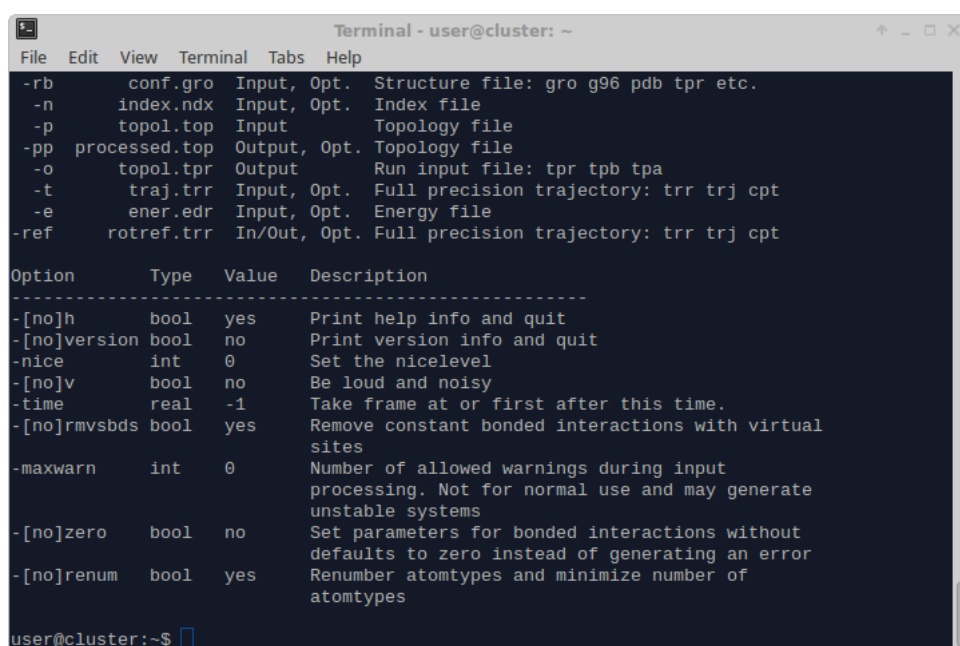


Figure 2.12: The architecture of the Apache Airavata science gateway API is depicted [GDP⁺ 15]

and widely used UNICORE computing middleware offer flexible means to integrate UNICORE into a variety of systems.

2.5.2 Commandline-based

Commandline tools are a highly flexible way to access features of whole data cycles or parts of it. Such clients are utilized by both developers to build data life cycle or connect them with other systems. Alternatively, commandline tools are used by advanced users that are familiar and content with such an interface (see Figure 2.13 for an example of its complexity). Examples in this category are offered by the Advanced Resource Connector [EGK⁺07], gLite [LEP⁺06], UNICORE [Sch15], and the Globus Toolkit [FK97, Fos05].



```
Terminal - user@cluster: ~
File Edit View Terminal Tabs Help
-rb conf.gro Input, Opt. Structure file: gro g96 pdb tpr etc.
-n index.ndx Input, Opt. Index file
-p topol.top Input Topology file
-pp processed.top Output, Opt. Topology file
-o topol.tpr Output Run input file: tpr tpb tpa
-t traj.trr Input, Opt. Full precision trajectory: trr trj cpt
-e ener.edr Input, Opt. Energy file
-ref rotref.trr In/Out, Opt. Full precision trajectory: trr trj cpt

Option      Type  Value  Description
-----
-[no]h      bool  yes    Print help info and quit
-[no]version bool  no     Print version info and quit
-nice       int   0      Set the nicelevel
-[no]v      bool  no     Be loud and noisy
-time       real  -1     Take frame at or first after this time.
-[no]rmvsbds bool  yes    Remove constant bonded interactions with virtual sites
-maxwarn    int   0      Number of allowed warnings during input processing. Not for normal use and may generate unstable systems
-[no]zero   bool  no     Set parameters for bonded interactions without defaults to zero instead of generating an error
-[no]renum  bool  yes    Renumber atomtypes and minimize number of atomtypes

user@cluster:~$
```

Figure 2.13: Example output of the grompp application within the Gromacs molecular dynamics application suite is shown to illustrate the complexity of commandline based tools.

2.5.3 Workbench-based

This category includes graphical workbenches where users install an application on their workstation. Workbenches usually bring more advanced usability in contrast to tools introduced in the previous Section 2.5.2. One downside is that workbenches have to be installed and maintained on the client side by either users or administrators. This leads to an increased management effort. Additionally, the development effort is increased as compared to web-based systems as developers have to maintain compatibility between different versions of the workbenches and underlying services. Examples include the UNICORE Rich Client [DSH⁺10], KNIME [BCD⁺08], the Taverna Workbench [BCD⁺08], and the Kepler Workbench [LAB⁺06]. Workbenches include a full set of workflow management functionality where workflows can be graphically created, modified, and submitted to connected computing resources.

Except for KNIME, which is designed to operate locally with only a limited HPC integration (see Section 2.3.3), they can also access remote data resources for input and result data. All are able to execute a large variety of workflows, scripts, and executables. The UNICORE Rich Client is focused on executing tasks on UNICORE-enabled remote High Performance Computing resources.

2.5.4 Web-based

Web-based science gateways [WDGK⁺08, WD07, LZWD⁺15] or virtual research environments [VP09] serve as a single point of entry to underlying data and computing infrastructures and, thus, make such infrastructures more easily available [GDP⁺15]. Users can directly access and utilize scientific life cycles from anywhere in the world with just a web browser. As science gateways are server-based, from the point of view of the user they are always up-to-date. Thus, users are relieved of having to install and maintain software by themselves.

The content management systems Drupal [Gra06], Joomla [Nor11], TYPO3 [FHA05], and the higher-level framework Django [HKM09] are widely used to enable science gateways. As these systems do not offer built-in support for access to distributed data and computing, such integration has to be developed from scratch or external libraries. This downside also afflicts portal frameworks such as Liferay [?] and Pluto [Fou15], but these offer the advantage of being JSR168/JSR286 [AH03, H⁺05] standard-compliant. Thus, portlets as user-facing components can be easily re-used in every portal framework also supporting JSR168/JSR286. Liferay is the most widely used portal framework supporting the standards JSR168/JSR286 in the distributed computing community, evident in portal like gUSE/WS-PGRADE [KFK⁺12, gUS15], EnginFrame [BLRR⁺10], VineToolkit [RDG⁺08]. Liferay offers advanced layout features, roles, support for various authentication and authorization mechanisms, layout support, and various general plugins.

On top of these technologies science gateway frameworks support advanced features such as workflow management and distributed computing access. gUSE/WS-PGRADE and Galaxy are examples that offer the ability to utilize remotely installed applications or web services. The ability to upload arbitrary scripts or applications is excluded from Galaxy as it is designed as a toolbox configured and maintained by an administrator. As gUSE-based science gateways are based on Liferay they are easily extensible by use case specific interfaces. As Galaxy is not based on such a portal framework, it does not directly support use-case specific extensions, but it is possible in general as Galaxy is open source.

There is a number of commonly used technologies such as WS-PGRADE, Hubzero, OGCE, and the Vine Toolkit. WS-PGRADE provides its own workflow engines besides being able to execute workflows from systems such as Kepler, Taverna, Galaxy, Triana, KNIME, and UNICORE. It is used in several science gateways such as MoSGrid (see Section 2.6). HUBzero [MK10] offers a middleware to graphically access remote clusters and tools. It is used in various science gateways in the US [MCD⁺15]. The Open Gateway Computing Environment (OGCE) [ACF⁺07] is a standards-based 3-layer architecture to create science gateways. It is based on Globus toolkit OGSA-DAI technologies, supports workflows, and is used in multiple projects.

Further examples of web based utilization methods are the UNICORE Portal [PHD⁺13], the Distributed

Application Runtime Environment (DARE) [MKEKJ12], Genius [ABF⁺03, BAD⁺11], EnginFrame, and GridSpace2 Virtual Laboratory [CNK⁺12]. Also worth mentioning is the Globus Online Data Management service [ABC⁺11] that enables usable and web-based data transfer scheduling between distributed storage devices by use of installed daemons and various supported data management systems.

2.5.5 Visualization

Visualization is an integral part of utilizing scientific data life cycles as it facilitates to visually grasp the meaning of data. Visualization ranges from creating 2D plots over graphical data representations to immersive 3D environments such as CAVE [OK07]. As the overall topic of this dissertation is a different one, the focus here is on visualization components that can be integrated with web-based utilization approaches such as science gateways. Dygraphs [Van06], for example, is a JavaScript plugin for displaying 2D plots based on text files. The ChemDoodle Web components [Tod14] offer 3D interactive molecule visualization capabilities. JMOL [Han10] is a molecular visualization toolbox that can be used via its Java API to enable molecular visualization in desktop applications. The JMOL visualization applet can directly display molecules within a science gateway based on chemical structure files such as SHELX, PDB, CIF, and mmCIF. The Web Graphics Library (WebGL) [Web15] is a JavaScript API able to render 2D and 3D graphics in compatible browsers. An example use is that within the Visible Patient Project [CSK⁺11]. Background on scientific visualization in general can be found in [NHM97]. These components enable advanced visualization within science gateways.

2.6 A Molecular Simulation Data Life Cycle

MoSGrid [KGG⁺14, GBB⁺12, GKG⁺14a], short for "Molecular Simulation Grid", is a data life cycle designed and developed for the computational chemistry community. The author took a significant part in its creation. MoSGrid supports complex HPC- and workflow-enabled simulations in three major chemical application domains. These simulations are highly sophisticated. An overview of its architecture (see Figure 2.14) is given about the data sources and sinks (see Section 2.6.1), how the data is handled (see Section 2.6.2) and how computing resources are integrated and utilized (see Section 2.6.3) to solve molecular questions in the three supported application domains Quantum Chemistry, Molecular Dynamics, and Docking (see Section 2.6.6). MoSGrid supports a complete single sign-on infrastructure through all layers (see Section 2.6.4). It enables an efficient usability of the functionality (see Section 2.6.5) of the complex infrastructure.

MoSGrid was extended by the author by creating a specific metadata management implementation (see Chapter 4) of the generic metadata management concept of this dissertation (see Chapter 3). Previously, users had to exactly remember where specific files are in a complex set of data and manually browse through a complicated directory structure in order to access them. Now, users are enabled to easily search for and utilize these files. This functionality is seamlessly integrated in the MoSGrid data life cycle.

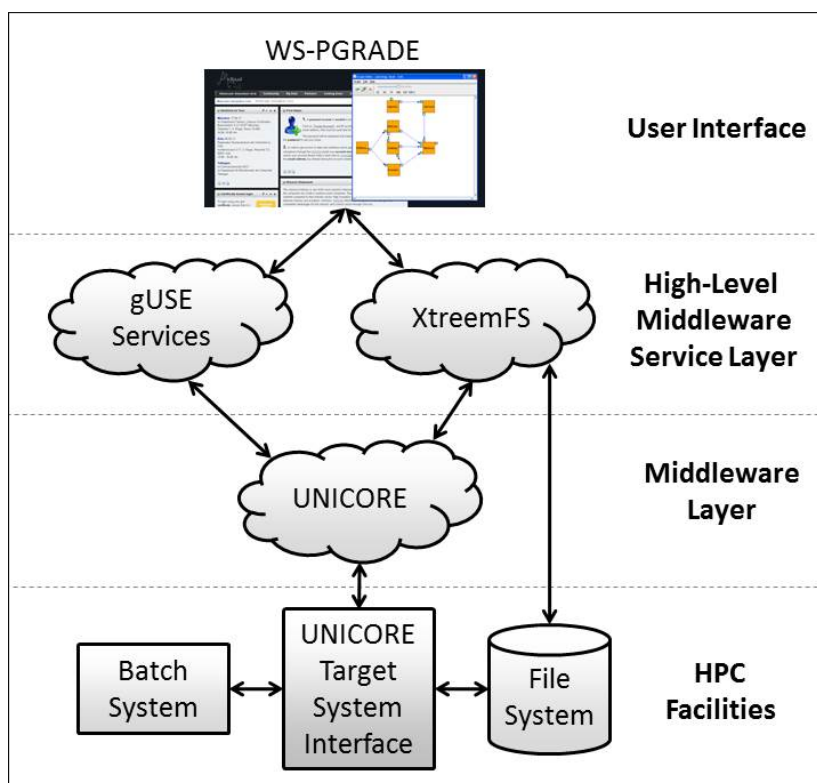


Figure 2.14: The underlying MoSGrid architecture [GGK⁺12] with the user interface based on Liferay and WS-PGRADE is shown. gUSE and UNICORE are integrated for workflow and computing management and XtreamFS for data management.

2.6.1 Data Sources and Sinks

The sources of data in MoSGrid are of a high variety. Their kind depends on the specific research question, professional preference, and application domain (see Section 2.6.6).

In the Quantum Chemistry domain there is the widely used Cambridge Structural Database [All02] and the Protein Data Bank [BWF⁺00]. Then there is the possibility for scientists to digitally draw the structures themselves. Also, input data might come from scientific measurement instruments such as resonance raman spectroscopy [HCB⁺13]. It can be used to experimentally gather measurements and compare these to results of theoretically simulated chemical processes. Then, the UV/Vis spectroscopy is a common method to characterize molecules and to observe chemical reaction processes. The comparison of this data with theoretical predictions, which necessitates efficient metadata management, is highly important [HCB⁺13, HHP14] in order to obtain multifaceted information about the respective electron structure. In the Docking domain a common research question done in MoSGrid is the benchmarking of new algorithms and methods in order to evaluate their performance and correctness. For this evaluation, thoroughly tested benchmarking data sets are used as a common benchmarking basis. A challenging example is the dataset "for the tyrosine-protein kinase ABL1 (PDB code 2HZI), containing 295 known active ligands and 10.885 inactive ones [MCIS12]."[KGHP⁺14] Another example is the DEKOIS [BIVB13] dataset. In Molecular Dynamics there is Casp [MBJ00] to evaluate protein folding methods. For production input data an important part is played by commercial vendor databases that are highly vetted.

Currently, the final destination of result data within MoSGrid is the data management system XtreamFS (see Section 2.6.2). It draws on backuped and professionally maintained storage capacity at Center for Information Services and High Performance Computing, Technische Universität Dresden. It is also easily possible for users to download results to their local workstation.

2.6.2 Data Management

The MoSGrid application domains (see Section 2.6.6) are highly complex. They have common requirements from an information technology point of view and specifically with respect to data handling.

XtreamFS (see Section 2.2.2) is utilized as the basis for the MoSGrid data management. A central installation at Technische Universität Dresden, complete with Object Storage Devices (OSDs), Object Storage Devices (OSDs), and Directory Service (DIR), is utilized for the MoSGrid data life cycle. XtreamFS is seamlessly integrated with existing technologies utilized by MoSGrid in the following ways. On each cluster that is connected to MoSGrid via UNICORE the XtreamFS FUSE client is installed. The client exposes the content of XtreamFS via the POSIX file system interface. Thus, the UNICORE Target System Interface (TSI) is enabled to access files within the MoSGrid XtreamFS installation.

In UNICORE job definitions are specified in the JSDL (Job Submission Description Language) standard [ADF⁺] and data staging is defined for input and result files. In the former case, input files are downloaded from a possibly remote location to the temporary UNICORE working directory, called USPACE, of the job. In the latter case, result data is uploaded from the USPACE to another location for safe-keeping. Two challenges existed with respect to the integration of distributed data management systems such as XtreamFS. First, inefficiency could occur when always UNICORE mechanisms have to be used. Second, when defining a workflow, distributed data sources had to be explicitly declared which limits the efficiency and portability of such a workflow.

A concept was designed to significantly mitigate these challenges that was consequently implemented and applied in MoSGrid [GBB⁺12]. The concept allows to use location independent file references. It is described in detail in Section 4.3.2. Another integration point with XtreamFS is via its Java API and utilized within the Liferay/WS-PGRADE domain specific portlets via the Portlet-API, described in Section 2.6.5 below. Furthermore, the Java API was utilized to create a file browsing portlet to graphically access files within XtreamFS.

2.6.3 Computing and Workflow Management

The computing pillar of the MoSGrid data life cycle is deeply rooted in the demands of the users. The central basis in all three supported chemical domains (see Section 2.6.6) are workflows (see Section 2.3.3). A large number of these workflows was and is continuously developed in order to provide use cases with fitting workflows. As creating workflows is a challenging task, the concept in MoSGrid is that either MoSGrid experts in concert with domain scientists or advanced scientists themselves create the workflows. During workflow enactment individual workflow tasks are submitted via UNICORE to clusters with potentially various batchsystems. The HPC systems that are currently connected are situ-

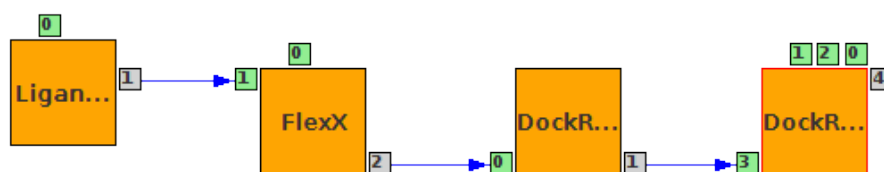


Figure 2.15: A docking workflow based on FlexX including pre- and postpressing is shown.

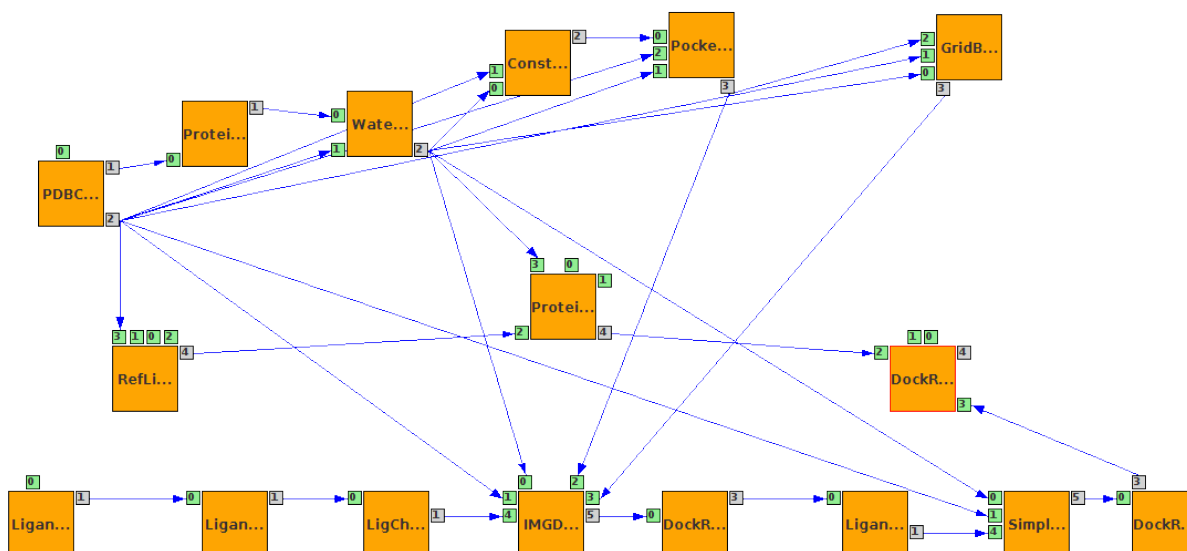


Figure 2.16: A complex workflow that automates a complete molecular docking procedure is shown.

ated in Dresden, Tübingen, Paderborn, and Cologne. The range of workflows in MoSGrid starts with basic ones that simulate phenomena via the execution of a single standard tool such as NWChem. Such a basic workflow provides compatibility were users already have existing NWChem input and configuration files. A slightly more advanced workflow (see image 2.15) includes the FlexX [RKLK96] docking application with pre- and postprocessing. It is worth noting that even these basis pre- and postprocessing steps are highly valuable to users. These workflows already automate previously manual and tedious steps. Without workflows user would have been required to manually do these steps themselves.

A highly complex workflow (see Figure 2.16) automates a complete docking protocol. It is a costly task to design and create such workflows. In return, these provide methods that are impossible for novice and advanced users to do manually. Even for expert users performing such advance scientific protocols manually would be both challenging and error-prone. It is important to note that from the perspective of the user details of a workflow are hidden. It is basically the same whether a workflow contains one node with a requirement of one core or a workflow with many nodes and a requirement of thousands of cores. Either way, in the MoSGrid science gateway just a few clicks are needed for the execution on distributed and large-scale High Performance Computing resources.

Highly complex workflows were developed within MoSGrid [HPBB⁺12, HPHGP13, HPBB⁺13]. As gUSE allows workflows to be nested, this concept was heavily utilized within MoSGrid [HHPG⁺13, HPHG⁺14, HPHB⁺15, HPHR⁺15]. The author actively participated in these activities.

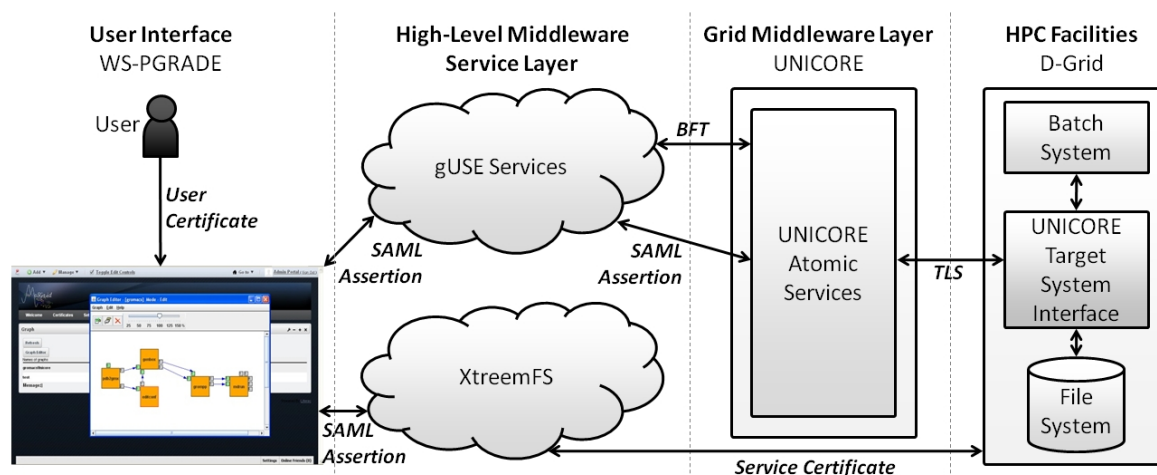


Figure 2.17: An overall view of the MoSGrid security and single sign-on architecture is depicted [GGK⁺12].

Generation of new assertion:

User Certificate (.p12):

Validity (in days):

Figure 2.18: The Java applet for creating and uploading a SAML assertion in the user's browser is shown.

2.6.4 Authorization and Authentication Infrastructure

The MoSGrid data life cycle features an elaborate security and single sign-on concept that is published [GGK⁺12, GGB⁺11] with the author being a lead author. The following section summarizes the overall concept and how individual components are working together to ensure a high level of security while at the same time being convenient to use. The concept is illustrated in Figure 2.17.

With the intention to register at the MoSGrid portal a user accesses the MoSGrid certificate tutorial that concisely details every step in order to gain full access. First, he needs to register at the science gateway and write a mail to MoSGrid in order to become an activated user. A role with the same name is associated with the new user. In parallel he is required to apply at his home institution for a certificate (see Section 2.4 for details). When both steps are approved, he can login at the science gateway. Via a special signed Java applet, extended under the supervision of the author, the user is enabled to create a SAML trust delegation assertion (see Section 2.4 for a SAML introduction and reference). The applet solution (see Figure 2.18) has the important feature that the personal certificate does not leave the computer of the user and, thus, is not stored or processed on a remote server. The applet was also improved in order to only display needed input fields and to automatically upload the generated assertion.

Both, the "activated user" role and the SAML assertion enables the new user to use the full functionality

of MoSGrid, namely to access the domain portlets and run molecular simulations. The role concept includes various roles in order to enable the separation of users and functionality, if needed. There is for example a section for advanced users and one reserved to MoSGrid developers and administrators. Under the supervision of the author, a further feature was implemented [SGG⁺12]. It allows for the automatic login into the science gateway with a single click while avoiding to enter his username-password combination. It automatically utilizes the certificate being embedded in his browser.

The created SAML assertion is stored in the directory of the user on the science gateway server. The assertion is used via the MyData portlet or the domain specific portlets to access the data management system XtreamFS. Access to the gUSE workflow services is automatically enabled when the user is logged in. When a workflow is executed by the user, gUSE utilizes the assertion to submit the individual jobs to the underlying UNICORE middleware. UNICORE interfaces with cluster resources via its target system interface (TSI). Individual users have their identity in UNICORE based on the DN from their certificate. A mapping exists for all users that maps their DN to specific user logins for each cluster. Whenever UNICORE acts in the name of the user via his SAML assertion to submit a job to a batchsystem of a specific cluster, such commands are executed as his specific local user. This way UNICORE transparently thigs in with the individual cluster security measures.

Distributed computing environments are highly complex and pose a challenge to ensure high availability. Furthermore, from a user's point of view it is optimal to keep things he has to deal with as easy and as few as possible. To work towards these goals, a separate VO service managing the MoSGrid VO was obsoleted. Instead, existing infrastructure that is already present in the MoSGrid science gateway is utilized and slightly extended. The information is used that the user is member of the MoSGrid VO, which is implicitly already present in the science gateway. Thus, modules were designed and implemented [GKG⁺14b] by the author. The modules manage this information, enrich it with necessary ID and securely make it available to relevant clusters in order to allow user access via UNICORE and their respective certificate. A cluster needs a mapping from the user's unique DN to the local login on the cluster in order for UNICORE to execute jobs in his name. This mapping is created in two phases. First, a generic mapping is created on the science gateway. It contains the DN of the MoSGrid VO members and a unique identifier, the VO name itself, and the access rights. The information that a user is activated in the science gateway is extracted from the science gateway database and for these users their DN, extracted from their SAML assertions, are stored in a flat file database together with the unique VO identifier, access rights, and the database entry status. Through this status it is prevented that user identifies are used twice. Based on this flat file database a generic mapping file is created and offered as download to the clusters. This is secured via encrypted HTTPS download and a cluster specific password using the Tomcat [Tom15] WebApp method. The second phase involves a module running on the cluster that is periodically executed to download the generic mapping. In order to make it specific for the local cluster, the module extends the list in order to include cluster specific information such as the cluster name and a login prefix. Then, it is ingested into the UNICORE user database for the mappings to be activated. By that, new users are activated and enabled to execute workflows via the science gateway on underlying High Performance Computing resources. The concept is similar to authentication federation where the user can use already existing login/password information, from for example the user's home institute, to login into other services. This concept lowers the hurdle to use MoSGrid as one manual registration

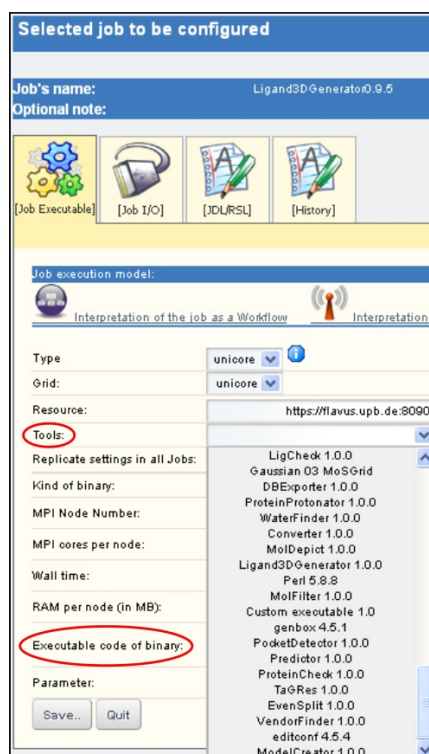


Figure 2.19: This screenshot shows configuration options a workflow developer might see during workflow creation [GGK⁺12]. The options to choose a pre-installed tool on the cluster and to upload an executable are highlighted.

steps by users was made obsolete and it improves the reliability and maintainability of the infrastructure as one less service needs to be maintained. Both increase the sustainability.

The author is leading the effort to design an integration with the DFN-AAI identity federation using the Unity service (see Section 2.4) to enable the use of pre-existing institute logins within MoSGrid. This will significantly further lower the hurdle of entry and use.

2.6.5 Utilization and Visualization

The MoSGrid science gateway [KGG⁺14, GBB⁺12, GKG⁺14a] is an advanced integration point and web-based graphical user interface to the MoSGrid data life cycle. It is run and maintained at the Eberhard-Karls-Universität Tübingen. The overall goal is to enable scientists to utilize powerful High Performance Computing and distributed data infrastructure to help solve their research challenges while at the same time being as easy-to-use, seamless, and supportive as possible. Liferay as a portal frame enables the overall structure of the web-based portal, its user and role management, and support for JSR168/JSR286 [AH03, H⁺05] portlets. These software components are pluggable and can be implemented to provide any kind of service. By default, Liferay provides the widely-used portlet framework Vaadin [Grö11]. It enables developers to create visually pleasing and responsive portlets with a large collection of standard components, including client-server communication, being available.

WS-PGRADE is the graphical user interface of gUSE and provides interfaces to the workflow graph edi-

tor, the workflow configuration editor 2.19 and the workflow repository. The gUSE workflow engine and UNICORE connector are used to enact workflows and send their individual tasks to UNICORE resources by transparently using the SAML assertions mentioned in Section 2.6.4. WS-PGRADE is lacking in usability with respect to workflow execution and it is unable to provide customized and advanced user interface that are tailored to the specific needs of a target community. Therefore, the so-called Portlet-API was developed in order to encapsulate complex features and provide them in a common way. This enables developers to quickly create user-friendly domain specific portlets while providing advanced features such as workflow management, data management, and Molecular Simulation Markup Language (see Section 4.1) handling.

The supported chemical domains are highly diverse in their research aims and therefore require their individual portlet for the execution of molecular simulations (see Section 2.6.6 for a detailed description). From an information technology point of view though, they are quite similar. Therefore, the Portlet-API was designed to provide a common portlet structure that is the same across the different domains. Additionally, it is possible [GBG⁺14] to customize portlet to suite specific requirements. The three main parts of the Portlet-API from a user's point of view are described in the following.

The "Import" view enables a user to load a workflow into his local workflow repository from the global MoSGrid repository. It stores all available workflow including default settings. MSML (see Section 4.1) [GBG⁺14] templates define the mapping from workflows to specific domains. A set of workflows belongs to a domain and each has a MSML template associated to it store a reference to the respective workflow in the global gUSE workflow repository. When a domain portlet is accessed the Portlet-API loads all available templates and creates a list of workflow from which a user can chose a workflow. Each imported workflow has a specific name. A default one is created that can be manually modified by users. Usually a template is also imbued with a workflow description and a figure which are automatically displayed. This information enables the user to quickly decide whether the selected workflow is correct for the task he intends. When a workflow was chosen the user has to click the "Import" button in order to create a concrete workflow representation. Once loaded, the concrete workflow can be selected in the next view.

In the "Submission" view, the user parametrizes the loaded workflow to enable its execution. He can select input data, for example molecule files, from either his local workstation or from the remote XtreamFS and choose parameters that configure the workflow at hand. As a novel approach, the "Submission" view is created on-the-fly by the Portlet-API based on the MSML template of the workflow that was chosen. This on-the-fly approach is highly flexible and advantageous to both users and administrators of the science gateway. Users are not overloaded with input fields that are not necessary. Administrators have to deal with less support requests as with less input fields less errors are made by users. The on-the-fly mechanism depends on the MSML dictionaries (see Section 4.1). Information in the MSML templates are stored in a list and used to determine what input is required for each workflow task while the dictionaries store information about the parameters of the applications themselves. Each template list entry, representing an input of a workflow task, can contain a default value and a link to a dictionary that corresponds to the application utilized in the specific workflow task. The information in the linked dictionary is then used to define data types, value upper and lower bounds, and may provide a help text about the meaning of the parameter. Furthermore, such configuration input fields may be defined as optional or

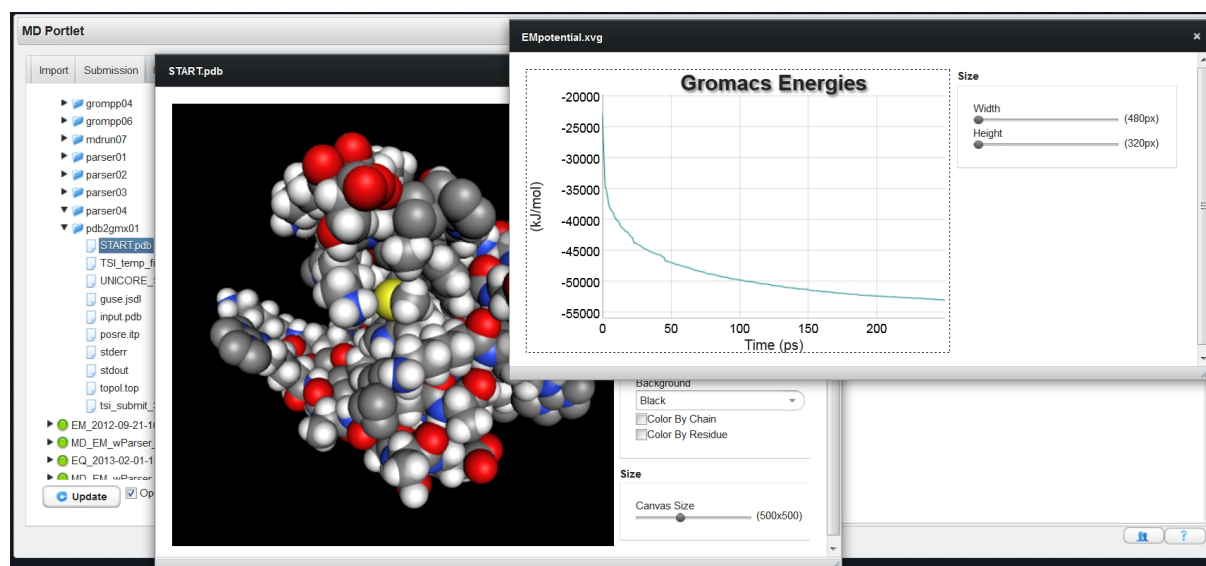


Figure 2.20: "Visualization of molecular structure using ChemDoodle is shown, as well as the dynamic generation of plots using Dygraphs. In this way simulation results from all application domains can be analyzed directly within the portal, without immediate need for download or external postprocessing." [KGG⁺14]

mandatory and the input may be chosen from a list of default values. Parameters may also be set in an on/off manner via a checkboxes. Also, on-the-fly integrated preprocessing of input files such as in PDB format is supported. Workflows may be linked to more than one MSML template in order to allow for workflows that differentiate between novice and expert users. This differentiation is done by hiding or presenting advanced parameters for fine-tuning. Once the input mask is filled in and the user click the "Submit" button, all entered parameters are checked via a validation routine. When the validation process is successful, adapters convert the input into the corresponding format needed for actually executing the workflow tasks. This format can be a string representing commandline parameters or complex input files. The MSML template file will be filled with the user's input and subsequently with results during workflow execution on a cluster. The workflow is submitted via the gUSE workflow engine that in turn submits the individual tasks via UNICORE to the underlying High Performance Computing resources. The running workflow is now available for inspection in the "Monitoring" view.

Besides monitoring the status of a submitted workflow, this view enables to explore running and finished workflow tasks. Intermediate and final results can be browsed and visualized in various ways. One is a basic text mode. Then, ChemDoodle Web components [Tod14] were integrated as a 3D molecule viewer with features such as changing direction of the molecule and highlighting properties (see Figure 2.20). Finally, Dygraphs [Van06] enables to interactively display 2D plots. The latter two are based on JavaScript and, thus, avoid the requirement of a Java browser plugin. In case intermediate results are deemed faulty, the workflow can be aborted and re-started with improved parameters. As this visualization approach is highly flexible, the integration of further plugins such as JMOL [Han10] or Vaadin Charts [Vaa15] is planned for the future by the MoSGrid collaboration.

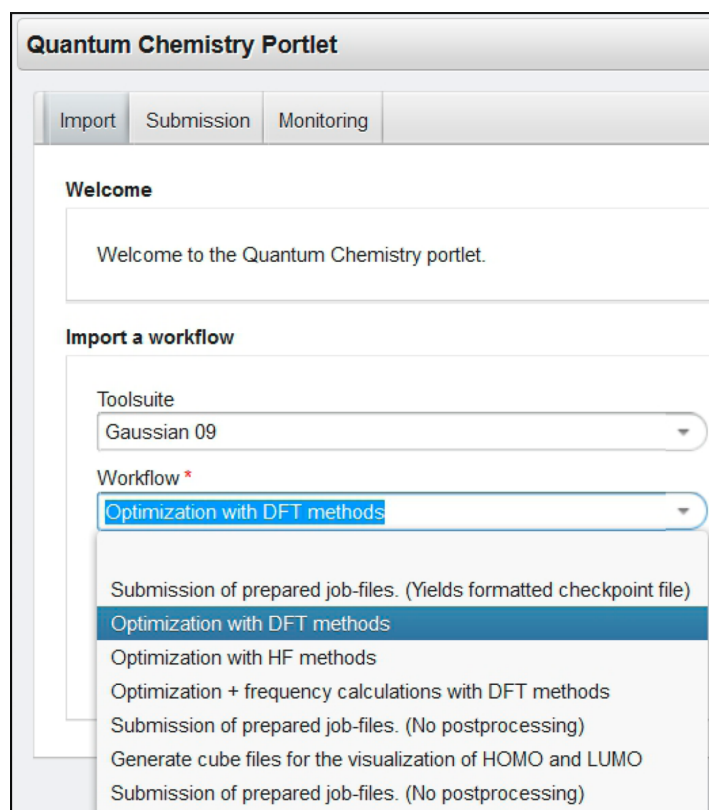


Figure 2.21: In the import view within the Quantum Chemistry portlet a variety of available workflows to choose from are shown [KGG⁺14].

2.6.6 Application Domains

In this section, common use cases in the three MoSGrid application domains are described in detail in order to enable an understanding how chemists utilize MoSGrid, what calculations they perform and how these are facilitated by the MoSGrid data life cycle. The individual sections are reproduced from the main MoSGrid overview publication [KGG⁺14] where the author is a lead author.

Quantum Chemistry Simulations

Quantum chemistry simulations (QC) "deal with the electronic structure of molecules. The job definition is always quite similar representing an ideal playground for the use of workflows. In a rather simple workflow, a given geometry can be calculated with a given set of methods, functional, and basis sets. The key geometric parameters are parsed and collected in tables afterward. Another use case would be the study of a complex potential hypersurface by varying one or several geometric parameter. Then, a set of similar jobs has to be submitted into the grid with the same methodical setup and varying coordinates. A suited workflow (e.g., a parameter sweep) combines the predefined coordinates with the chosen method and facilitates the whole process of submitting and collecting data. Both types of workflows are independent of the quantum chemical code. Further postprocessing can cover the addition of a solvent model, calculation of charges and frequencies, formatting of checkpoint files, and definition of new job files for subsequent time-dependent DFT calculations.

With respect to requests from the community, the geometry optimization workflows were primarily implemented for Gaussian [FTS⁺09], the well-renowned quantum chemical suite. Gaussian compute jobs are typically driven by a single input file, which includes a machine-specific section with the so-called “link0” commands, stating the number of cores and the amount of memory required, a job-specific section called route card, which describes the type of quantum chemical calculation to be performed, and finally the representation of the molecule in question, i.e., charge, spin multiplicity, and atomic coordinates. These input files are uncomfortable for scientists to work with and are inflexible in the represented structure, e.g., blank lines at certain positions are required. With the workflows for geometry optimizations, made available through the quantum chemistry portlet, users are relieved from manually constructing such an input file. Instead, the required file is generated with the help of an adapter from the user input to textboxes and listboxes with reasonable default values, taking advantage of MSML. In order to reduce the complexity of the input masks, geometry optimizations using Hartree-Fock (HF) methods or Density functional theory (DFT) are realized through two separate workflows (see Figure 2.21). While the HF workflow allows to explicitly set the type of HF theory, for example ROHF or UHF for open shell species, the DFT workflow requires the DFT functional to be set. Machine-specific parameters common to both workflows, such as the number of cores, the wall time, and the required memory, are accessible through input fields. For the convenience of the users, these are preset to reasonable default values. A single listbox is populated from an MSML dictionary entry containing frequently used combinations with the Pople basis set 6-31G as the default. Unless a charged or open-shell species is to be examined, the default values for charge and spin multiplicity can be left unchanged as well. User interaction is required for the input of the molecular structure. With respect to the needs of less experienced users, the corresponding textbox supports an easily understandable four column data format, consisting of an element symbol and the Cartesian coordinates of the respective atom. When the workflow is submitted, an adapter generates the necessary input file. Once successfully calculated, the geometry optimization can be followed by several postprocessing steps. From the machine-dependent binary checkpoint file of the calculation, a new Gaussian-type input file with the optimized geometry is created. A subsequent task extracts fundamental job data, such as the type of calculation (route card), the number of orbitals, etc., to an ASCII file; another one extracts vibrational frequency data. Since the binary checkpoint file cannot be used on any other cluster, it is converted to its machine-independent and human-readable, formatted counterpart. From this formatted checkpoint file, a PDB file is generated. As a result, the user is enabled to visualize the optimized molecular geometry in the monitoring tab of the portlet through ChemDoodle.

While these workflows primarily address users needing guidance, experienced users may prefer the expert workflow, which allows to directly submit an input file previously created by the user. A manifold of molecular properties, however, is not reflected in the molecular geometry itself. Therefore, different workflows have been devised to generate volumetric representations of their spatial distribution in the form of cube files, a file format also common to other quantum chemical packages. Currently Gaussian and NWChem [VBG⁺10] workflows are available through the MoSGrid science gateway." [KGG⁺14]

Molecular Dynamics Simulations

Molecular Dynamics (MD) simulations "enable the users to study properties of larger molecules such as proteins and nucleic acids. They deal with the impact of force fields between atoms and molecules on their physical movements. The huge number of interacting atoms as well as the required small time discretization makes it a computationally challenging task, which cannot be solved analytically for any reasonable sized molecular ensemble. Besides geometric aspects (as for example the opening and closure of a protein binding pocket ⁴³), also kinetic and thermodynamic quantities can be derived, e.g., rate constants for protein folding [SKL07, SPHvdS05, VDSPS07]. Corresponding workflows can be considered as rooted graphs starting from the molecular structure provided by the user. The user input undergoes some filtering and conversion to MSML to ensure clean and error free input for the simulations. The resulting structure needs to be processed with various tools in order to create a valid simulation system including. Here the modular character of Gromacs tools [BvdSvD95, HKVDSL08] is advantageous as an individual task can be assigned to a node within a workflow handling the execution of a specific tool. Although multiple different simulations are possible, all of them have at least one node pointing to the main MD executable. This program calculates the actual movement of each atom in the system and its contribution to the energy. Within the Gromacs suite this application is mdrun, which expects a binary combination of topology and simulation parameters as input and provides as output a trajectory, i.e. positions with respect to time, an energy file, and logs comprising simulation output. Further simulation steps can be built on this output along with various analysis tools can use it to generate scientifically relevant plots and charts. This information is directly visualized within the MD portlet. To give an example, the user may directly check output conformations using ChemDoodle or inspect root-mean-square deviation plots using Dygraphs. All relevant simulation results are annotated with metadata and stored in the MoSGrid repository.

Within MoSGrid, currently MD simulations with Gromacs are supported. Popular programs like Desmond [BCX⁺06], NAMD [PBW⁺05], and AMBER [SFCW13] shall follow soon to support the user community with their research. Independent of the simulation code used, all MD programs have some general steps in common, which can be easily represented by workflows. It is noteworthy that MD workflows are mostly rather complex, and a suited workflow represents a significant help to the user who tries to overcome technical hurdles. The molecular simulation system has to be put into a periodic box, solvated with water and counterions, followed by an energy minimization. The single steps for a typical equilibration are described in detail in the following paragraph.

The first task is to identify all atoms and bonds in order to generate a correct description of the atomistic interactions, a so-called topology (topol.top). This is accomplished by using the Gromacs tool pdb2gmx. All protein hydrogen atoms are removed before being replaced according to the definitions of the force field (e.g., Gromos [OVMVG04], OPLS-AA [JTR88], Charmm [MBB⁺98], or Amber [PC03]) used for the later simulations. A reasonable protonation state is estimated based on the default capabilities of pdb2gmx. The resulting structure with force field consistent atom names is put into a simulation box using editconf. The size of the box is chosen large enough to avoid periodicity artifacts. This is achieved by setting the distance between the solute and the box 20 percent larger than the largest cutoff. Afterward the box is filled with water using genbox. Here, different water models are possible (e.g., the Simple Point

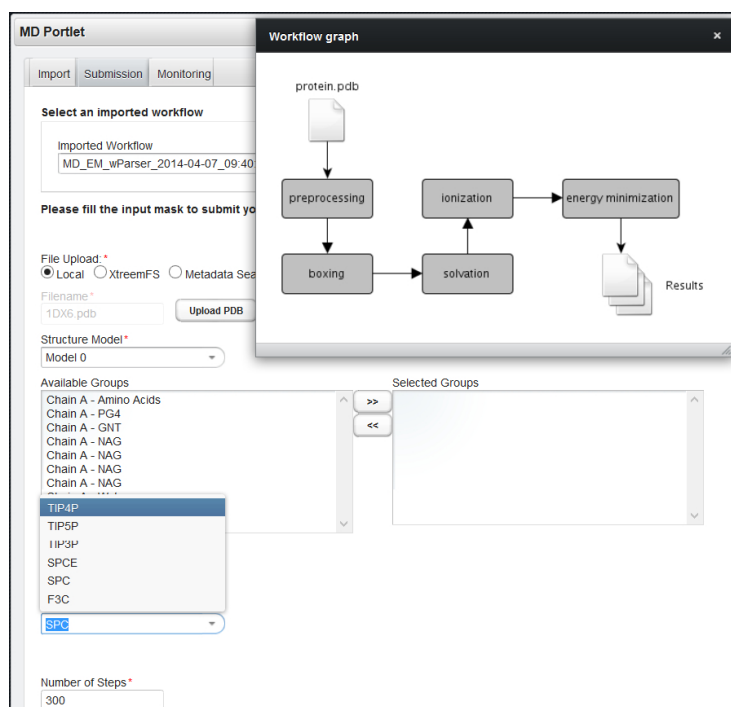


Figure 2.22: "Part of the configuration of a Molecular Dynamics workflow is shown. In order to ensure meaningful simulations the users are obliged to select appropriate groups of the uploaded PDB file. Furthermore, details such as the force field, corresponding water model, and number of steps for the energy minimization have to be specified. For inexperienced users carefully chosen default values are present." [KGG⁺14]

Charge model (SPC) [BPvGH81] or the series of different Transferable Intermolecular Potential functions (TIP) [JCM⁺83]) (see Figure 2.22). The calculation of the explicit solvent water is computationally intensive, as it increases the number of interacting particles, but is usually desired to enable natively like conditions. Implicit solvent models may be used through the submission of self-prepared job descriptions. The topology file is updated accordingly in the next step. The precompiler grompp is executed in conjunction with a simulation description yielding the binary file `topol.tpr` including all coordinates, bonds, forces, and further information to carry out an energy minimization. This calculation is carried out by calling an MPI-parallel version of `mdrun`. The output of the minimization is directly taken as a starting point for a brief position restrained equilibration run. The precompiler grompp is executed once again but using a different simulation description. After the restrained MD equilibration simulation the last frame of the simulation is written to a PDB file, serving as a starting point for production runs, docking studies, and/or further scientific analysis. Usually it is futile to relax the simulation system further by carrying out a nonrestrained equilibration. For variants of this simulation protocol multiple workflows are provided, reducing the tiresome system preparation to a few mouse clicks. By adding further options to the input masks that guide the user through the simulation setup process the (advanced) user will gain more control to adapt the workflow to his/her specific problem. By creating standardized workflows, in which only a small subset of parameters are modifiable by the user, different simulations of one workflow become comparable. Additionally, since all information to set up the simulation is stored inside the workflow and is inherent part of the sustainable stored file and directory structure, the simulation experiment is readily reproducible.

With a large variety of possible MD simulations, it is not possible to provide prebuilt workflows for all use cases. Therefore, users may also upload and simulate self-prepared job descriptions (Gromacs-topol.tpr), taking advantage of powerful grid resources." [KGG⁺14]

Docking Simulations

"Docking deals with the calculation of the pose of a ligand within a receptor's binding site and the estimate of the binding energy of the resulting complex. For the purpose of docking, we used the open source docking tools of CADDSuite (Computer Aided Drug Design Suite) as framework to pre- and postprocess the molecular structures. We here take advantage of several grid-enabled tools that divide docking into simple tasks. 57,58 The central docking and scoring steps can be carried out with CADDSuite [Koh12], Autodock [GMO96] or FlexX [RKLK96].

The docking portlet provides several workflows with varying detail of user input required, ranging from the unprepared protein plus a flat file screening library to fully prepared receptor and ligand files. The complete workflow consists of four major steps: 1) target preparation, i.e., separation of the receptor structure from its ligand in the PDB file, protein protonation, elimination of irrelevant water molecules, and pocket detection; 2) ligand preparation, i.e., generation of 3D coordinates for the screening library at a specified pH; 3) docking, i.e., grid calculation and docking; and 4) rescoring of the best scoring compounds. This workflow only requires two input files: a PDB file containing a protein with a reference ligand and a SDF file containing the molecules to be docked into the protein. In addition to that, variants of the workflow are offered via the portlet that leave out parts of the preparation steps in case the user has already prepared their files using other software and only wants to perform the docking simulation on the grid. Although the user interface is kept quite clean and simple, it allows the setup for a wide range of parameters for the initial docking step and for the refinement procedures if desired (see Figure 2.23).

The compute time, necessary to prepare a high-level ligand database, can easily exceed the actual time needed for docking itself. Once a ligand library was properly prepared, e.g., assigning the correct protonation state and generating meaningful conformers, it may be reused anytime. As it is stored within the MoSGrid repository it easily can be selected for further simulations. It is also possible to share such a library with co-workers via the integrated user rights management.

When it comes to performing docking using libraries with several thousands of ligands, the biggest challenge is the dynamic utilization of grid resources. The workflow management service gUSE provides a simple way to parallelize jobs via generator and collector ports. Combining the use of these ports with tools in the CADDSuite, the workflow takes the ligand input SDF file and splits it into multiple smaller files. Each generated file contains a small portion of the given ligand library. Next, docking is performed on each of these single files. The workflow service will generate as many parallel jobs as files are provided. In the end, the results of each parallel docking job are merged into a file for further processing. This parallelization will provide results much faster than a pure sequential approach and consequently saves a lot of the scientist's valuable time. It is also possible to rescore docking results. Following a similar approach as for the docking step, intermediate docking results can be re-evaluated employing a different scoring function.

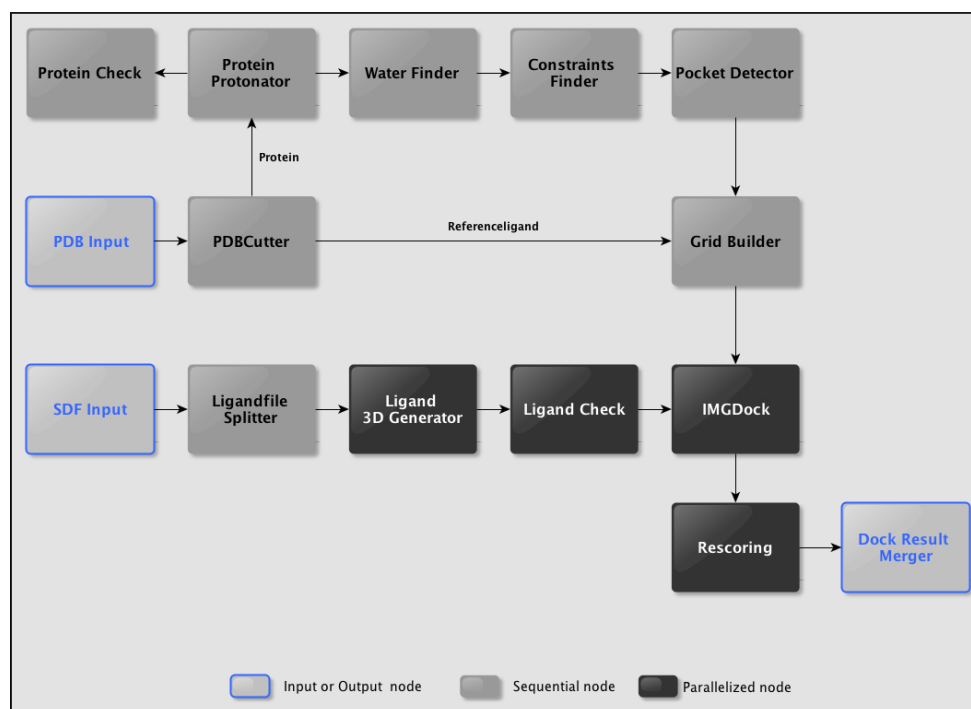


Figure 2.23: "A typical docking workflow is shown, based on CADDSSuite and its tools. The workflow handles the complete preparation and docking simulation for an input set of a target structure complex and a screening library. Workflows for other docking applications, such as FlexX and Autodock, follow the same principle and structure, although the distinct placement algorithm is used." [KGG⁺14]

A major advantage for the end-users utilizing MoSGrid's docking portlet is found in the ready-to-use state of the portlet, thus preventing them from creating their own scripts to run a docking simulation. By automatizing the process of protein and ligand preparation relying on MSML, the reproducibility is guaranteed and the risks of avoidable errors minimized. More complex crystal structures of larger protein multimers may require a deeper insight into the structure of the PDB file itself. However, MoSGrid's docking portlet makes an educated guess within the docking workflow. Furthermore, the problems occurring when converting one molecular file format into another, e.g. PDB to PDBQT, can be nearly completely eliminated due to the usage of MSML as uniform storage format, ensuring consistency of all data." [KGG⁺14]

2.7 Further Data Life Cycles

This section introduces several further data life cycles in a compressed form to give a general impression about the variety of data life cycles integrating data and compute capabilities.

2.7.1 Worldwide Large Hadron Collider Computing Grid

The Large Hadron Collider (LHC) [Bir11] is a 7.5 billion Euro particle accelerator at CERN, Switzerland. It took 20 years of preparations and construction before its operation started in 2010. The aim of the LHC

is to revolutionize the understanding of the nature of the universe. In this line, a significant discovery was the Higgs particle in 2013 that gives some particles its mass. The particle was postulated in 1964 and the Nobel Prize in Physics was awarded in 2013 for its discovery. Each of the LHC detectors (ATLAS, CMS, ALICE, and LHCb) record the products of collisions with a rate of 1 petabyte per second. Dedicated electronics and a large cluster decide in real-time what data is of further interest. This reduction reduces the data rates ranging from 50 MB/s to 1250 MB/s, depending of the detector. The reduced data is subsequently stored in the archives at CERN Computer Center. The Worldwide LHC Computing Grid (WLCG) manages and processes the data utilizing resources around the world and it consists of 5 tiers. Tier 0 at CERN is the central data facility. It does a part of the processing and archives all data with a replica distributed to the 11 tier 1 centers. These also provide clusters for (re)processing of the data and provide a large number of tier 2 centers with data for analysis. At the tier 3 institutional sites, data is cached and analyzed by the end users, whose workstations constitute the 4th tier. The data distribution across all sites is done by GridFTP. The computing services handle job submission and monitoring to the clusters while the workflow management matches the jobs to fitting resources. File catalogs and database are running at tier 0 and tier 1 sites to manage the location and metadata of files. Further services exist for virtual organization and information management and application software is to be regularly updated across all sites. The interoperability is ensured via grid projects under the hood of major grid infrastructures such as EGI (European Grid Infrastructure) [EGI15] in Europe and OSG (OpenScience Grid) [AAB⁺11] in the United States. The WLCG infrastructure has been evaluated in 2014 for the second phase of LHC operations. The integrated metadata capabilities are highly use case specific.

2.7.2 Human Brain Project

The Human Brain Project (HBP) [HBP15a] is a FET Flagship project of the European Commission with the aim to significantly further the understanding of the human brain and to ultimately emulate it. The 10 year project started in 2013 with about 80 partners from 25 countries with the aim of 200 partners by 2018. The budget is about 1 billion Euro including 50% direct funding from the European Commission. The HBP HPC Platform [HBP15b, Sch14] aims to provide services that integrate supercomputing, Big Data, and Cloud capabilities at exascale level. The core of the HPC Platform is the UNICORE middleware (see Section 2.3.2) for the federation of major High Performance Computing capabilities located in Jülich (Germany), Barcelona (Spain), Bologna (Italy), and Lugano (Switzerland) with the large-scale Cloud storage located at Karlsruhe (Germany). A newly developed Unified Portal will provide the web-based graphical interface for all users. The portal will interact with UNICORE via its new REST API. A central LDAP server will manage all users including their access rights. The user information will be exposed for authentication via an OpenID Connect (OIDC) server. The Unity federated identity management system (see Section 2.4) will integrate OIDC with the UNICORE security capabilities. The UNICORE user database will be synced with the LDAP server for authorization purposes. An architecture overview can be seen in Figure 2.24. Metadata capabilities are planned to be integrated in the future.

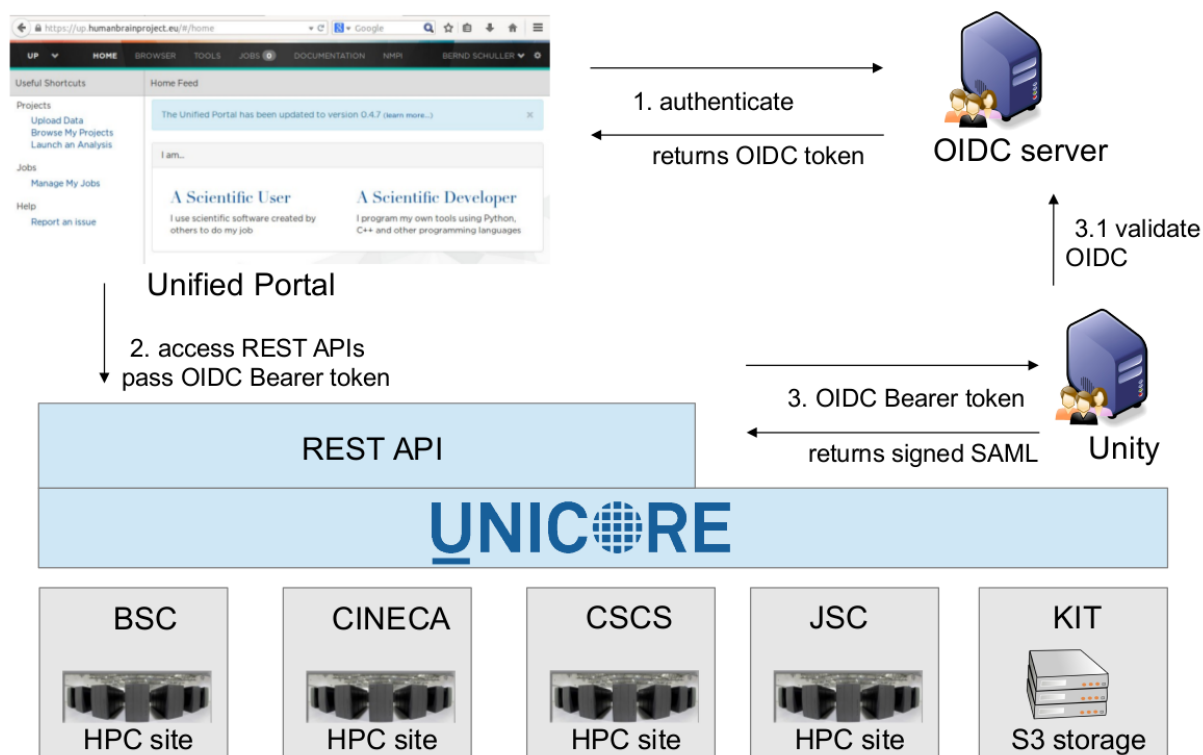


Figure 2.24: An overview of the architecture of the HPC Platform overview of the Human Brain Project is shown. Computing and data sites from all over Europe are integrated via the UNICORE middleware with authentication being handled by both an OIDC and Unity service [Sch14].

2.7.3 Pierre Auger Observatory

The Pierre Auger Observatory [Bah12] aims at detecting rare ultra-high energy cosmic rays and to determine their source and composition. The computational aim is to compare the recorded data with theoretical models using Monte Carlo simulations. These require significant computing capabilities. Tasks are submitted to EGI computing resources by means of the gLite middleware by execution scripts and by defining input files needed for the simulation or a list of files of measured data to be analyzed. Result files are stored on storage elements integrated by gLite. Management scripts keep track of the job status and re-submit in case of an error and write status information to a database. This information is utilized within a PHP webpage to make relevant information available as needed. Metadata capabilities do not seem to be existing.

2.7.4 IceCube

IceCube [AAA⁺09, AAA⁺15] is a cosmic neutrino detector below the ice at the geographic south pole. Its aim is to facilitate the study of high-energy neutrinos and dark matter. The combination of processing the acquired data and running corresponding simulations is requiring significant processing power. IceProd [AAA⁺15] is a computing management system that is lightweight, coordinates data movement and distributes processing tasks to grid resources. A central database is the coordination point and a set of daemons are responsible for job management and submission to underlying resources via plug-

ins. These are available for batchsystems such as HTCondor, PBS, and SGE and middlewares such as Globus, gLite, EDG, CREAM, and ARC. About 1 TB of data is produced daily that is reduced on a 400 core cluster at the south pole by an order of magnitude before transmission to the central storage facility. The data is further processed at 43 research institutes around the world. The central database is storing basic provenance information such as software version and job parameters.

2.7.5 Virtual Earthquake and Seismology Research Community in Europe e-Science Environment

VERCE (Virtual Earthquake and Seismology Research Community in Europe e-Science Environment) [CKL⁺13] is a research project that conducts seismological research in order to simulate earthquakes and predict their consequences. The VERCE platform centers around the Dispel data streaming workflow language (see Section 2.3.3). VERCE nodes enact Dispel workflows which may access a library of workflow components and scientific catalogs. A special Dispel gateway was implemented that de-serializes data and submits computing tasks with such data as input to either Globus or UNICORE resources. The resulting data is re-serialized to be stored in the VERCE data archive from which it may be accessed by users. While integrating metadata capabilities was outside the scope of the VERCE project, it is evaluating to utilize iRODS together with the MonetDB database [NK12].

2.7.6 Climate-G Testbed

The Climate-G Testbed [FAF⁺11] is a science gateway and underlying infrastructure to enable climate change research. It features the data management technology OPeNDAP [GPW⁺06] that provides data access, subsetting and aggregation. THREDDS [DCD⁺06] is employed as a middleware for connecting data providers and users. Both are specific the earth science focused technologies. The gateway supports proxy certificates for authentication which are created via a JNLP Java Webstart application. GReIC [FNA11] allows to manage databases for gLite and Globus middlewares and supports proxy certificates and virtual organizations. GReIC is utilized in the Climate-G Testbed to provide use case specific metadata management.

2.7.7 PolarGrid Portal

The PolarGrid Portal [GSP09] follows a different approach than over infrastructures. It integrates functionality of third-party commercial collaborative services such as Facebook, Google, Gadgets, and Twitter. This approach is described as allowing for less development effort and leading to results in a shorter time. The heavy reliance on external services makes the sustainability questionable and has privacy implications. Metadata capabilities do not seem to exist.

2.7.8 Distributed Research Infrastructure for Hydro-Meteorology Community

DRIHM (Distributed Research Infrastructure for Hydro-Meteorology community) [DCG⁺ 14] is a hydro-meteorology research community. It utilizes computational methods to improve the prediction of severe and increasingly common hydro-meteorological events such as storms and floods. This constitutes an interdisciplinary challenge involving meteorology, hydrology, and earth sciences. Its data life cycle consists of three main stages; workflow design, definition, and execution, task execution on resources and the interpretation of results. The design, definition and execution is based on a science gateway build with the gUSE/WS-PGRADE framework (see Section 2.5.4). Depending on the data and workflow, the individual workflow tasks are executing on HPC, HTC or Cloud resources. The DRIHM binary repository hold relevant applications while the static data and temporary result data repositories hold input and result data respectively. Based on this data, the results are interpreted by remote visualization or downloading for local visualization, statistic or 3D visualization methods. No metadata capabilities are integrated.

3 A Novel and Generic Metadata Handling Concept

This chapter thoroughly describes the novel and generic metadata handling concept as a way for including metadata capabilities into scientific data life cycles. First, limits in the current situation of life cycle management are presented, followed by a description of the main challenge the thesis is tackling. Second, besides an introduction to alternative approaches, the solution concept is described. On the one hand, the concept consists of an elaboration of essential characteristics required to enable advanced metadata management in a favorable way. On the other hand, it consists of a design guide for implementing metadata management in complex data life cycles including technology recommendations. By embracing the characteristics and developing the answers to the design guide questions, a computer scientist can expect to acquire a deep and broad understanding of the complex situation of data life cycles in general and of metadata specifically. This understanding will significantly facilitate the creation of an effective metadata management architecture in a specific data life cycle.

3.1 Limits in the Current Situation

3.1.1 Important Challenges out of Focus

In the vast complexity of data life cycle management, a multitude of challenges exists that are described in the following. This section leads up to the challenge that this dissertation tackles.

Scientists tend to be in their field for the reason, they want to focus on it. Many will naturally only learn about information technology tools that are necessary to further their research and then often only the bare minimum. From another point of view, overly complex technologies are significantly hindering scientists in advancing their fields. At the same time, bleeding edge science increasingly requires the most powerful computing and data systems [Hub12, BHS09]. These systems tend to be increasingly complex. A major challenge is to provide virtual environments that are intuitive and seamless to enable scientists to be more efficient in producing scientific insights (see Section 2.11 for context). At the same time, such environments need to be integrated with powerful High Performance Computing and Big Data infrastructures. This integration is required to enable the speedy production of simulation and analysis results that are now of a previously impossible to handle magnitude. To enable this increase, highly complex simulation and analysis procedures have to be encapsulated as scientific workflows (see Section 2.3.3). These can be run with just a few steps instead of the need to manually execute each job by itself and manually convert input and output formats between jobs. Workflows are an essential concept to simulate phenomena and analyze data. They enable novice users to execute complex scientific procedures and they significantly foster reproducibility of scientific results. Scientists that wish to independently reconstruct results are much more easily able to do so via workflows. To offer the maximum usability,

all data life cycle components have to be integrated via a workflow engine. At the same time, as many details have to be hidden as possible from users. This abstraction enables workflows that are easy to create and execute in order to provide the most benefit for the user with the least hindrance.

As data amounts are ever rising (see Section 2.1.1), it will increasingly be difficult for more and more use cases to store and process all raw data. Preprocessing methods to quickly decide what data needs to be kept and compression methods to further reduce the size will become significantly more important. These methods have to be as transparent as possible in order to avoid errors and additional effort by users. Necessary user interactions needs to be minimized by automation as far as possible. Postprocessing will also have to be increasingly applied to transform data to a state that is more immediately useful to users. To efficiently enable postprocessing, all data life cycle components have to be efficiently integrated with each other, to avoid bandwidth and server capacity waste.

With the increasing data life cycle complexity [JMPK⁺15] comes the increasing tendency towards errors as more and more systems depend on each other. To counteract this tendency, resilience, and error handling need to be major design factors on all levels. Systems should be able to continue to function to a certain degree even with failing components or at least fail gracefully. Erroneous states should be automatically recognized as early as possible. Resulting error messages should be propagated where and on what specific knowledge levels they are needed. For example, an administrator is often able to digest the highest detail of infrastructure information while per default end users should not be exposed to such detailed information.

A challenge that affects all components is to enable effective and efficient security measures (see Section 2.4). These involve the integration of federated identity management systems to enable efficient authentication, authorization, and single sign-on mechanisms. Existing institute accounts should be used wherever possible to ensure a high convenience and low barrier of entry. Furthermore, involved systems need to support a chosen standard which is a significant challenge as a high variety of systems exist.

Sustainability is a challenge on all levels. Funding schemes often only enable computer scientists to build up infrastructures, but seldom do they include funds for operation and maintenance of the developed infrastructures. Long-term funds would increase the reliance of users in such systems.

Another challenge is that of building and maintaining trust;

- How can a user know that the infrastructure does exactly what it promises?
- How does he know a workflow does exactly what the descriptions says?
- Are the produced results of high scientific quality?
- Is the infrastructure well designed?
- Are his results secure from competing researchers and save with respect to infrastructure failures?

A challenge overlaying all others is finding a good balance between opposing goals such as performance, feature-richness, usability, security, resilience, and sustainability (see Section 3.4.1 for overall design aspects). While information technology systems tend to advance in general, use case specific balances continuously need to be found.

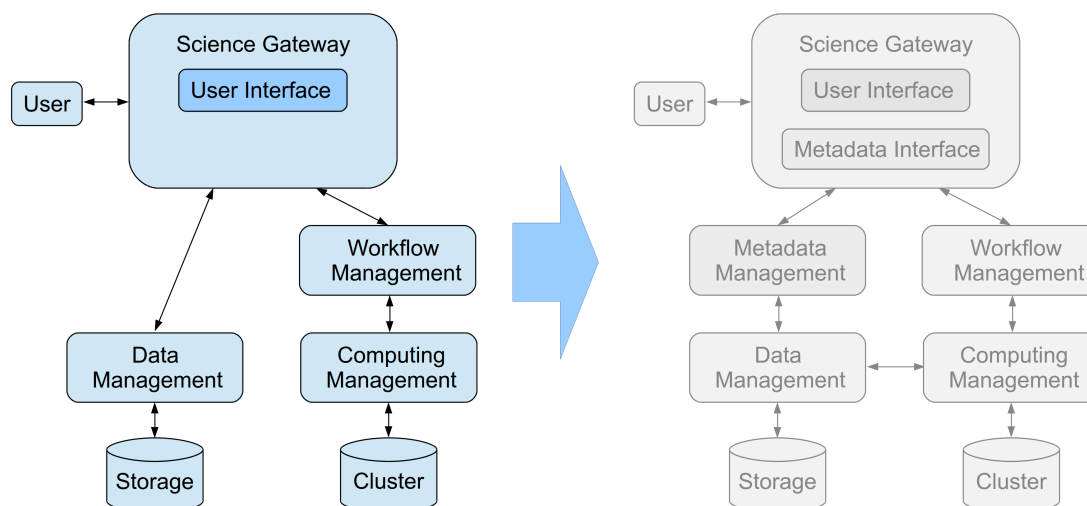


Figure 3.1: The left side shows a science gateway centric abstract view of data life cycles without means to semantically organize data.

3.1.2 The Challenge

The major challenge this dissertation focuses on is handling data in increasing amounts and complexity. Knowledge about the data; where it is, what it contains, what it means, is often only in the minds of the scientists, in lab books on paper, or at most, hard coded into thousand directory and millions of file names. These situations make important knowledge very volatile and accessible only to a very limited number of people. When these people become unavailable, data that only they know in detail becomes basically useless as nobody else might have a sense of the data anymore. Without knowledge about data content it is increasingly difficult to access data in a meaningful way.

Metadata management systems within data life cycles can be used to organize such highly complex data with possibly tens of millions of files. They enable easy access via capabilities for searching based on data content. However, such systems are often highly use case specific or completely missing (see Figure 3.1 for an illustration). Meaning that either deployed solutions are hardly applicable for other use cases and, thus, are hard to maintain in the long term. Or, metadata for the benefit of the user is missing completely in a data life cycle. An essential aspect of this challenge is the efficient and frictionless integration of metadata systems with all layers of a data life cycle. This seamlessness enables users to focus on advancing their science, instead of having to learn a lot of information technology specifics.

Legacy techniques are not suitable anymore as these require significant efforts to integrate metadata capabilities in limited ways for every new data life cycle.

3.2 Improving the Situation - Focus and Scope

3.2.1 The Overall Approach

The significant open challenge of missing or narrow use case specific metadata management was met by designing a novel and generic metadata handling concept for integrating metadata capabilities in scientific data life cycles [GBG⁺14, GGJN14] (see Figure 3.2). Previously, to the best of my knowledge, no overarching and generic concept existed. The concept consists of characteristics of generic metadata handling and a design guide to well-balanced metadata handling within highly complex scientific data life cycles.

The first part of the concept describes required characteristics for generic metadata handling (see Section 3.3). A main property is the abstraction from technologies on various levels such as data, metadata, computing, workflow, security, and utilization technologies. This enables metadata systems to be generically integrated and accessed in a uniform way as unnecessary specifics of underlying systems are hidden. Automation is also facilitated by encapsulating various functionalities as it makes them easy to trigger automatically. Another characteristic is the generic handling of metadata and data formats and more specifically the data-to-search chain of components. This chain includes the extraction, annotation, and indexing of metadata and results in meta information about data being searchable by users. The chain needs to be automated as much as possible. Then there is the requirement of seamless data life cycle integration of metadata capabilities. A part is the integration with systems for data, computing, workflows, security, and utilization. This tight integration enables the metadata management to be highly usable as scientists can utilize the data life cycle as one whole system instead of distinct individual parts. At the same time of potentially vast capabilities of HPC and Big Data resources can be directly used.

Second, a design guide to metadata in data life cycles (see Section 3.4) was created in order to significantly ease the integration of metadata management in scientific data life cycles. First, overall data life cycle design aspects are discussed in detail. This enables a thorough appreciation and understanding of the complexity of scientific data life cycles and subsequently their design. Then, the focus is on the metadata handling integration. A plethora of points is raised and discussed that ranges from the scale of data over fixed cornerstones to enabling search and utilizing of results. Technology recommendations are described that cover all aspects of data life cycles such as data, metadata, computing, workflow, security, and utilization components. The technologies are generic, have favorable characteristics, and are proven in existing data life cycles.

The complete implementation of the concept for the MoSGrid data life cycle is described in detail in Chapter 4. An adaption outlook for a microscopy data life cycle is described in Section 5.1.2.

3.2.2 Alternative Approaches

Alternative technology approaches exist that are also independent of specific use cases. These are described here and put in contrast to the dissertation concept.

The KIT Data Manager (KDM) is a research data repository (see Section 2.2.3 for details). It aims at

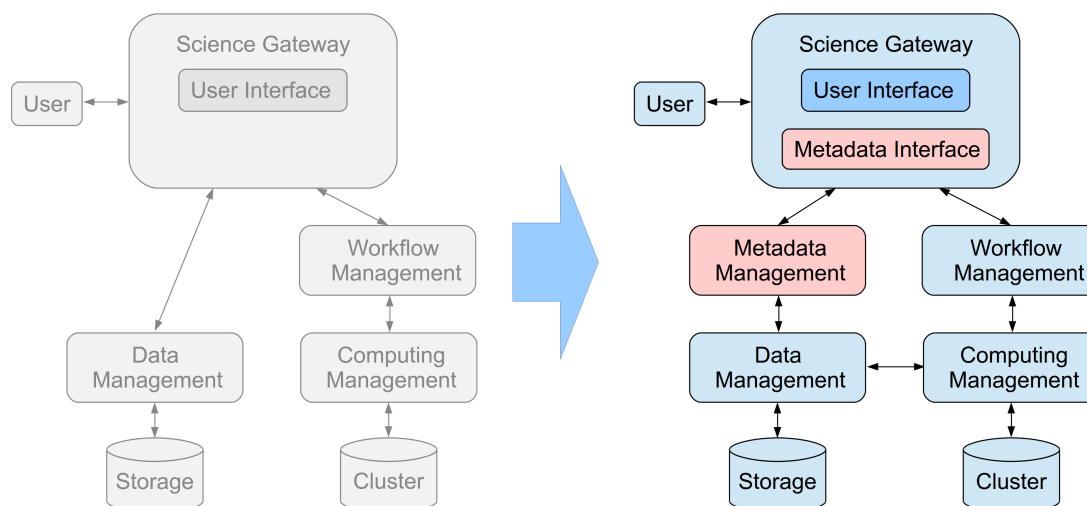


Figure 3.2: The right side shows an abstract data life cycle with metadata capabilities that are both seamlessly integrated and generic. The capabilities are accessible in a usable and efficient way to transparently organize and utilize data based on its content.

petabyte scale data management by utilizing elaborate metadata management. KDM features a data-centric view and advanced archiving services such as bit preservation, data migration and storage visualization as well as automatic policy enforcement. KDM is able to abstract from data management systems such as dCache and iRODS via the Abstract Data Access Layer API (ADALAPI) [SHJ⁺11]. As utilization mechanisms, REST, and Java APIs are offered as well as a web interface. Furthermore, services are offered for data and metadata management, staging, and sharing, authorization and discovery of data based on its metadata. Integration with specific data life cycles can be performed by the creation of individualized commandline clients. KDM supports cluster-based executions of computing tasks via the LSDF Execution Framework for Data Intensive Applications (LAMBDA) [JHS⁺12]. KDM is missing a generic and widely applicable HPC and workflow integration. Thus, computing is a priority of secondary importance. To mitigate this, KDM is currently extended within the MASi research infrastructure project (see Section 6.2). In contrast, the approach presented in this thesis facilitates the native integration with High Performance Computing, Big Data frameworks, and workflow management systems (see sections 3.3.3 and 3.4.3). Thus, the dissertation approach can support data life cycles that are heavily compute intensive and diverse in their computing requirements.

The EUDAT services B2SHARE and B2FIND offer metadata capabilities that are generic but limited in scope (see Section 2.2.3 for details). The aim is on small research data that is ingested via a web portal with manual metadata annotation. As such manual steps are tedious and error prone, the applicability is severely limited. Basic metadata is the same across communities while a profile can be chosen to manually enter use case specific metadata. Metadata profiles for new use cases can be added. The planned B2NOTE service aims to add more elaborate metadata capabilities such as automatic extraction. Further metadata can be included in B2FIND with the OAI-PMH interface. EUDAT is a centralized system. As such, user data and metadata can be outside of the direct control of the user's local data center. These characteristics can have negative safety, privacy, and sustainability implications. In contrast to the dissertation approach, EUDAT aims at providing basic metadata services that are only loosely integrated

with further infrastructures and, thus, difficult to seamlessly integrate with the working environment of users. In the dissertation approach the metadata is stored in the same directory of the data, independent of the physical data location. Thus, if the data is backed up or replicated, the metadata is automatically treated so also. EUDAT is fixed on the iRODS data management system while the dissertation approach is agnostic with respect to data management systems, besides other categories. Metadata in EUDAT is accessed via a centralized web portal. Seamlessly integrated computing capabilities are missing, while manual data staging to remote High Performance Computing resources is possible.

iRODS is mainly a data management system (see Section 2.2.2). Its metadata capabilities are limited to the iRODS relational database storing attribute-value-unit triples. These may be used for basic and advanced search queries which can be executed via different methods. Existing functionality for metadata extraction and annotation are missing. These have to be implemented using the iRODS rules engine. By virtue of the central iRODS data database, its metadata approach is completely centralized. Thus, data and metadata are stored in different locations with the related downsides described in Section 2.2.3.

Agave as a science gateway API (see Section 2.5.1) supports metadata capabilities in the form of an unstructured NoSQL database. Metadata schemas are supported as well as automated metadata extraction and a global search. In contrast to the dissertation approach, the metadata is stored in a central database.

The main use case of the DIRAC computing middleware (see Sections 2.3.2 and 2.2.3) is within physics environments while it is not limited to it. Via the Dirac File Catalogue, metadata can be associated as key-value pairs to files and directories. Certain keys can be declared for indexing and, thus, used for search operations. Complex structures can be build as directories, subdirectories and files subsequently inherent metadata of their precursor. The access and search functionality is offered via Python API, commandline interface and web interface. In contrast to the dissertation approach, DIRAC is centralized and has no extraction. Annotation capabilities seem to be available and the data is completely separated from its metadata in a central database.

3.3 Characteristics of Generic Metadata Handling

Metadata handling in complex data life cycles is required to have certain characteristics in order to be generally applicable. One characteristic is that of abstraction from underlying layers to hide unnecessary specifics. Another key aspect is the application of extraction, annotation, and indexing of metadata. The final aspect is the necessity of being able to seamlessly integrate metadata management with surrounding technologies.

3.3.1 Abstraction from Technologies

Abstraction, hiding specifics of lower layers, of various technologies is vital for this concept to be both generic and manage complexity at the same time. Abstraction is fundamental in all of computer science and ranges to levels with as less abstraction as the physical principles.

In the specific field of High Performance Computing systems, Big Data, and their application within

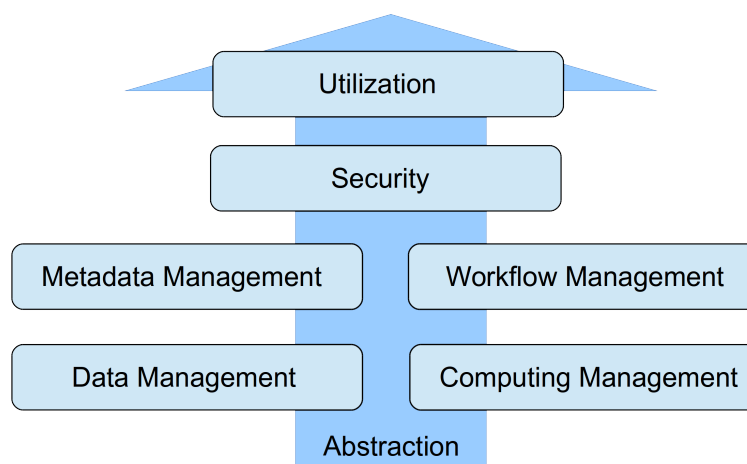


Figure 3.3: Abstraction from metadata systems and various other levels of abstraction from data and computing management up to the security and utilization layers are shown.

scientific data life cycles, example abstractions are the following. Frameworks and libraries enable the re-use of code created to encapsulate and automate lower level functionality and to hide its complexity. Models abstract from reality itself by generalizing and focusing on certain aspects in order to enable to simulate reality based on these models. Such model-based simulations enable the thorough investigation of phenomena with specific levels of detail, optimized for the size of the computing resource at hand. The implementation of such highly compute-intensive algorithms is fundamentally facilitated by automated parallelization and error detection methods. Developers are aided in graphical ways by performance analysis and optimization tools. The essential point is to enable the transparent use of complex underlying methods without the need to know all their specific details. Abstraction from specific systems directly facilitates a general applicability.

The fundamental concept of abstraction is applied to metadata and data management systems and access standards to provide uniform ways to access such underlying systems. In principle the dissertation concept supports many concrete technologies. Meaning that it is applicable for any data life cycles employing these data management system. As the further abstraction from data and metadata formats is central within the concept, it is dealt with in detail in Section 3.3.2. Another aspect is that besides this related interwoven systems for computing, security, and utilization (see Figure 3.3) should be abstracted from as well. A suitable metadata management needs to fit with such layers of abstraction. Abstraction from computing resources such as supercomputers and their batchsystems (see Section 2.3.1) is an important part of the concept. Computing management systems can be further abstracted from via workflow management in which various workflow engines exist. By cluster and computing management abstraction, the concept is naturally HPC-enabled and also enables the re-use of metadata search results as input in compute tasks. Various general modes of access exist. For example commandline clients, web clients, workbenches, and APIs (see Section 2.11). With flexible utilization mechanism, usable data life cycles can be more easily created and, thus, enable scientists to fully focus on their core research instead of IT technologies. Example security mechanisms are certificates and federated identity management systems such as LDAP, Active Directory, Shibboleth, and OpenID among others. To enable single sign-on via trust delegation, SAML assertions and proxy certificates are included (see Section 2.10).

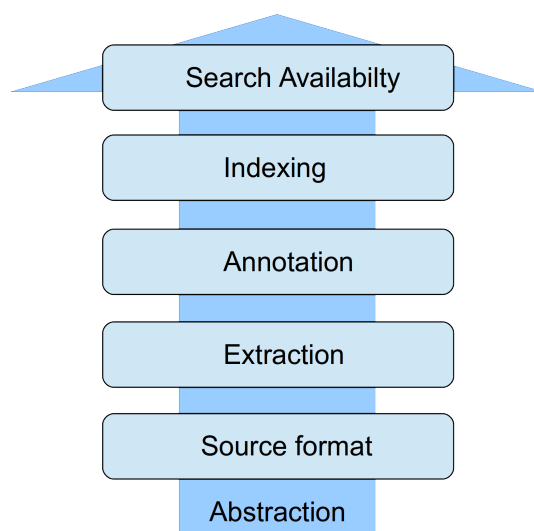


Figure 3.4: The figure depicts various abstraction layers reaching from data and metadata formats at the base to search availability in ascending layers.

The various levels of abstraction mentioned in this section directly and natively enable automation as the underlying complex methods and functions can be more directly be triggered via a simple interface. Furthermore, the need for implementing these methods by oneself in a data life cycle tends to be avoided by adapting or integrating existing technologies.

3.3.2 Generic Handling of Metadata and Data Formats

This section describes the generic handling of metadata and data formats by utilizing the combined proximity metadata approach (see Section 2.2.3) to enable the exposure to search queries. For this, the required tasks are extraction of information from formats, annotation of original data with metadata and the indexing of metadata for searchability (see Figure 3.4). This flexible approach facilitates a high degree of automation to enable the management of large numbers of files. The approach also facilitates the avoidance of re-creating metadata capabilities by the integration of standard components to enable quick adaptations to different types of use cases.

The concept includes generic metadata extraction frameworks (see Section 2.2.3) for support of a wide range of file types. In principle, any such framework can be integrated. Meta information can be extracted from supported formats in order to be further utilized to fuel search capabilities. Besides such generic frameworks, specific extraction applications exist within scientific communities. This may be necessary when the target format is too specific or complex to be included in an extraction framework or the required mode of extraction is not supported. To enable a high degree of flexibility, it is important that support for such custom methods is possible in metadata management.

Commonly, metadata extraction is performed on a central server. This can be a problem in the case the extraction step is very compute intensive or many are performed. The central server might be overloaded. Thus, it can be advisable for performance as well as resilience reasons to apply customized distributed metadata extraction methods. In such a case, the extraction tasks shall run close to the data alongside

the distributed computing tasks within workflows running on computing resources instead of the central server. The relation of extraction to compute task depends on the situation and workflow at hand. A sensible strategy is to attach an extraction task to every compute task within a workflow whose result is of interest to search for and, thus, metadata needs to be extracted from it. Several intermediate workflow task results may be of interest or only those of a final workflow job. This decentralized extraction approach tends to make good use of compute resources, if the extraction should prove compute-intensive. In case of the MoSGrid implementation, within individual workflow tasks only a minor compute overhead is imposed as is shown in Section 5.2.2.

Additionally, this execution schema increases the resilience as the metadata extraction and annotation are independent of a central metadata service. Thus, if this service fails, a job or workflow can still finish on the High Performance Computing system and extract and annotate the metadata. This ensures the usefulness of the data, with metadata in a consistent state and annotated to it. When the metadata service is available again, the metadata can subsequently be indexed to become available for search. The annotation happens when the created metadata is stored in the metadata file besides the actual data.

In the dissertation approach, a unified index to enable search queries over all metadata information is build utilizing the metadata in this intermediate format. This way, only this specific format needs to be supported by the indexing engine. The necessary many-to-one metadata conversions from specific source data formats to the intermediary metadata format are performed by dedicated and specialized tools. This division of labor enables both kinds of tools to be highly optimized for their respective goal. The mode of access and search architecture depends on the utilized search software. Using such an intermediate format as introduced above, the metadata is stored in a schema free way which facilitates indexing and fosters adoption in further use cases. By avoiding a schema the flexibility and independence of data and metadata formats is facilitated.

The metadata extraction, annotation, and indexing shall be automated as much as possible. Depending on the data life cycle technologies, the triggering of these steps can be integrated in the workflow engine at the end of a workflow, via a commandline client, an API or whenever it is sure that relevant data is written. This full automation is essential for an efficient usability.

3.3.3 Seamless Data Life Cycle Integration

Seamless interoperability is a fundamental requirement of generic metadata handling as data life cycles are only well functioning when users recognize them as one whole system and not as many individual parts.

Metadata Management needs to be seamlessly integrated with data, compute and workflow management. This enables the direct use of metadata search results in further analysis tasks. With workflow management integration, the results can be utilized as input for any task in highly complex encapsulated analytical procedures. In data life cycles that integrate such concepts, HPC and Big Data technologies can in principal also be applied in jobs and workflows while utilizing search results. The more a user is able to subsequently do with the found data, the more useful metadata and the whole data life cycle is. Optimally, the underlying metadata management technologies should be interchangeable by a

programming interface that hides their specifics and offers general metadata capabilities such as extraction, annotation, and indexing. This way the overall metadata management is more adaptable to given surrounding technologies.

The integration of metadata management with authentication and authorization as well as single sign-on capabilities is essential to avoid additional required user actions which would severely limit the seamlessness within complex data life cycles. When every data life cycle part is well integrated the whole data life cycle becomes more as the sum of its parts. For example, the construction of higher level functionality such as user interfaces in science gateway environments is facilitated. This can be done in a way that all underlying functionality is provided in a consistent way without the cumbersome crossing of system boundaries and, thus, provides a prerequisite for optimal efficiency of use. This facilitates efficient and transparent access for users to organize their data. It raises the acceptance among users and subsequently allows more users to benefit from metadata management. User interfaces within science gateways should be implemented using standards such as JSR168/JSR286 [AH03, H⁺05] to foster cross science gateway interchangeability. The possibility to use commandline clients and APIs increases the flexibility inside a data life cycle and to the outside in regard to utilization and access capabilities.

3.4 A Design Guide to Metadata in Data Life Cycles

A deep integration of metadata methods with a data life cycle is essential for a high efficiency and frictionlessness. Metadata management integration is highly complex as it is associated with most technologies in a data life cycle. Thus, to adequately discuss integrating the previously described metadata aspects (see Section 3.3) in a data life cycle, first, design aspects of creating a complete data life cycle are discussed (see Section 3.4.1). Then, aspects to integrate metadata management capabilities are presented (see Section 3.4.2). The chapter is concluded by discussing technological recommendations (see Section 3.4.3).

3.4.1 Overall Data Life Cycle Design

Designing scientific data life cycles is a highly complex task. It combines being in close collaboration with the target user community to understand their specific requirements while understanding a wide range of technologies and especially their interoperability characteristics [GDP⁺15]. Overall design aspects are discussed in this section while metadata and subsequently data aspects are discussed in the next section. The scientific principle of utilizing existing knowledge and technologies and to apply and extend these should be applied. This enables life cycles to be created much more quickly. At the same time they are more advanced than it usually would be possible if they would be developed entirely from scratch.

A thorough requirement analysis has to be carried out by the target community in cooperation with IT experts. The goal is to list the features that the community requires and get an understanding of the specific situation. This is usually a highly challenging task as users have a deep knowledge and understanding mainly in their specific research domain and usually not within the information technology

domain. This tends to lead to highly specific views and languages in different research domains. These differing languages constitute a fundamental challenge in interdisciplinary science and collaborations. This challenge includes that domain researchers naturally have a different focus than computer scientists. It leads to the unawareness of domain scientists of specific software, computational tools, security demands, the management of data and computing, and concepts such as metadata and workflows.

To bridge this gap, first, a common language and understanding has to be developed in order to effectively communicate and cooperatively design the new data life cycle. Second, the focus should be for both IT experts and future scientific users to avoid the expectation that the other group has the time and inclination to deeply understand the other field. Instead, both should abstain from delving too deeply into explanations of their specific methods and focus of the parts that are immediately relevant for both sides in order to efficiently design the new data life cycle.

The planing should at least address the following aspects related to the scientific domain in order to develop an understanding for many important aspects [GDP⁺15]:

- What is the overall goal of the data life cycle?
- What are the characteristics of its future users and domain likely to be?
- What distributed resources have to be integrated?
- What are the computing requirements (see Section 2.3.1)?
- What are the user interface requirements (see Section 2.5)?
- What are the relevant user roles and the level of experience with using computing and data resources?
- What are currently utilized systems, applications, and technologies and how do they need to be integrated?
- Are these technologies commandline-based or web services?
- What new components have to be developed?
- Which features are required, nice-to-have or optional?
- What are the requirements with respect to security, safety, and privacy?
- What specific workflows have to be developed?
- How much effort will this require and how are the domain experts involved in the creation process?
- What are the visualization requirements?
- What collaboration tools need to be integrated?
- How is the user community spatially distributed?
- Is it local, national or international?

Besides domain-related aspects, technical, and developmental ones need to be considered as well [GDP⁺ 15]:

- What is the level of experience of involved developers with frameworks and programming languages to be chosen from?
- Where, how, and under what conditions will the service be hosted?
- How will the new data life cycle be integrated with various local, national or international resources for authentication, authorization, data, and computing?
- Do the capacities of these security resources sufficiently scale?
- How well supported and sustainable are the involved technologies?
- How well are the technologies going to scale at expected usage levels?
- How will the effort be distributed between extending current technologies and developing new ones?
- What are potential synergy effects with related data life cycles and how can these be utilized?

The more of these aspects are discussed and analyzed the more soundly the architecture tends to be and the less it has to be re-iterated during development and at later stages. The described aspects significantly influence the choice of technologies.

Based on the analysis of the situation and the requirements, it is the task of the IT experts to design the data life cycle as a whole. As existing technologies should be re-used as much as possible, the evaluation of existing technologies should be done in parallel. This way, design requirements influence the technology choices and vice versa. An important aspect of choosing a technology is its sustainability so that there is a good chance it is actively maintained and further extended. Free and open source software is an important aspect so developed extensions can be fed back. When an open source project is used widely enough, there is a reasonable chance that the software is important enough to be further maintained. The risk is principally avoided that in case of a commercial product, that if the company goes bankrupt or loses interest in it, the software completely disappears.

With increasingly advanced existing technologies and concepts, life cycles can be designed and developed much more efficiently nowadays. This leads to even more advanced and feature-rich data life cycles. During build-up, the graphical, and architectural design is often re-iterated between developers and future users due to changes in the underlying technology and further feature requests from users. Experience shows that the reasons for this re-iteration process can be minimized with careful planning and close collaboration especially during the requirement analysis, design, and continuous evaluation phase along. It is important to regard a data life cycle from multiple view points. This helps to cover various topics, reach sensible design decisions, and facilitate a smooth development process. As each new data life cycle is different, there is no one technology that is applicable to all situations [GDP⁺ 15]. An array of technologies exists that needs to be chosen from and combined with each other.

3.4.2 Metadata Management Integration

The integration of metadata management is tightly coupled with the overall data life cycle design. Thus, it has to be considered in close interaction with the overall data life cycle design aspects from the previous section. Besides the following aspects, especially the technical and developmental ones from the previous section are highly relevant as well.

- What are the data sources (see Section 2.2)?
- What is the scale of the data in terms of size and file count?
- What data formats are going to be involved?
- Are these openly standardized?
- Are metadata extraction tools available for these?
- What does a user need to search for and what metadata is needed for that?
- Is the defined metadata useful to users?
- Is the metadata sufficient to enable the utilization of the data within that use case?
- Is the targeted metadata system efficiently usable?
- At what point should steps such as extraction, annotation, and indexing be automatically triggered?
- Are there existing sources of metadata such as standard metadata out of scientific devices?
- By whom is data envisioned to be re-used and under which conditions (see Section 2.2)?
- Will electronic lab book functionality be required?
- With what security aspects will the metadata management have to fit in?
- Is a new data life cycle integrated with metadata management or an existing one extended?
- Are there fixed underlying conditions such as specific technologies that metadata management has to deal with?
- What are the data sinks (see Section 2.2)?

The inclusiveness of the definition of metadata highly depends on the use case at hand. For example, a computing task is finished and the result created. The result might already be considered metadata. It was created based on the data and, thus, it is information that describes the data. Here, practicality comes into focus. When the result is a limited number of values, it might be feasible to extract and annotate these as metadata. But already when there is a large number of these values or result files, it might become impractical to extract, annotate, and index all values. This especially holds true when the result data is in the terabyte or even petabyte range. Existing metadata standard should be used if possible and not a new one invented as there are already a high number of standard available (see Figure 2.4 and 2.5 for an illustration). Widely used standards also constitute an integration-enabling characteristic, as supporting tools may already exist that can be re-used.

It needs to be determined if there are potential scalability challenges of the extraction methods in terms of computational effort. If so, the distributed approach of extracting metadata along side computing jobs is advisable. The combined proximity metadata approach directly enables this by storing the metadata alongside the data. In this case resilience characteristics are also improved as the metadata extraction and annotation is independent of a central service instance. In extraction, annotation, and indexing metadata, the actual data should be moved as little as possible to avoid bottlenecks.

Metadata extraction is usually the parsing of data or metadata files and the storing of relevant information in a designated format in other files or a database. The extraction is mostly specific to the file formats at hand as these tend to be all different. Indexing exposes the decentralized metadata, stored in files besides the actual data, to be available for search. The search base can be extensive when, for example, the full content of a text file is extracted and indexed. Or, the search base can be small when just the most important information is extracted as metadata. A balance has to be found for each specific use case between a large search base that might yield more search results but which also negatively affects the size and, thus, performance of the index and vice versa.

Metadata capabilities need to be integrated with the authentication, authorization and trust delegation mechanisms of the data life cycles at hand. This enables user-friendly search interfaces to discover files based on their content. This is essential as well as the ability to directly further utilize search results. The metadata integration has to be as transparent and usable as possible in order to avoid an increase of complexity from the view point of the user and, thus, facilitate acceptance. Crossing system boundaries, for example when different systems and user interfaces for data and metadata systems are used, have to be acutely avoided.

3.4.3 Technology Recommendations

Various technologies may be required to fulfill the requirements for a new data life cycle. The life cycle architect has to choose technologies based on the situations and requirements of the specific situation which was explored for data life cycles in general in Section 3.4.1 and specifically for metadata in Section 3.4.2. Besides the focus on metadata technologies, general ones are recommended here as well as no technology can be seen in isolation. Figure 3.5 elucidates specific technologies that are supported or abstracted from by the following recommended technologies. They are categorized by the central data life cycle categories from Chapter 2. That chapter also lists further technologies, besides the following, that could be worthwhile to consider. The technologies that are described here were chosen as they are highly generic and show favorable characteristics. They are utilized in the MoSGrid implementation (see Section 4) and in various other data life cycles. They are evaluated alongside the implementation in Chapter 5 and are considered for a further implementation in Section 5.1.2.

Data Management

UNICORE is a prime example for a generic and mature, yet continuously evolving technology that is widely utilized in, besides others, large infrastructures such as PRACE [PRA15], XSEDE [XSE15], and the EU flagship Human Brain Project [HBP15a]. It is a central technology in the MoSGrid data life

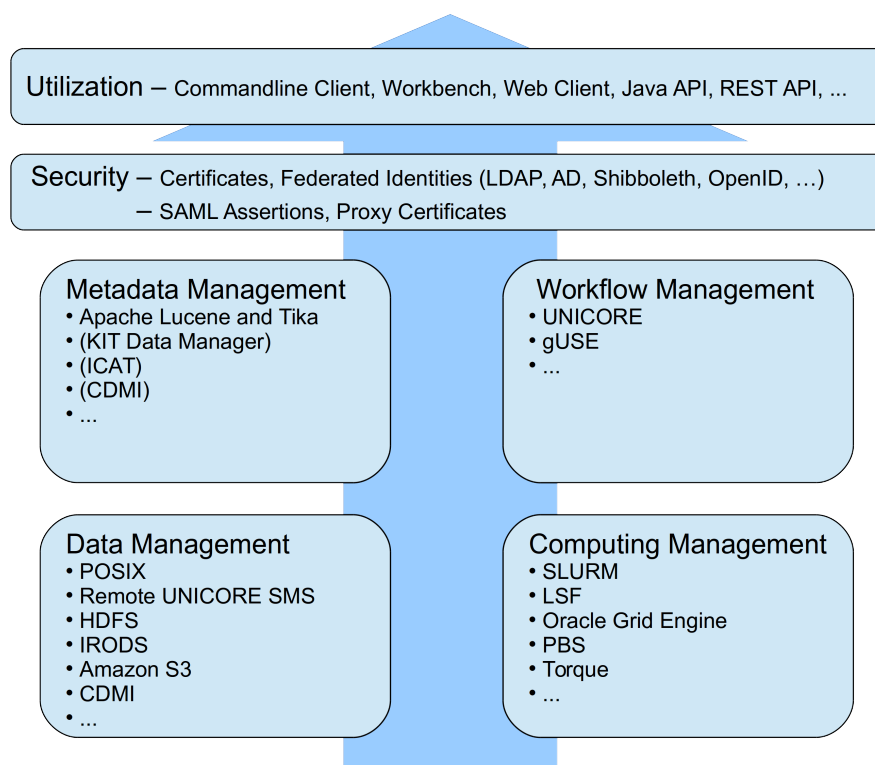


Figure 3.5: Abstraction from metadata systems and various other levels of abstraction from data and computing management up to the security and utilization layers are shown. Specific example technologies that the concept supports via the UNICORE middleware are noted.

cycle (see Section 2.6) which is the basis for the concept implementation (see Chapter 4). In terms of data management (see Section 2.2.2), UNICORE supports, abstracts from and provides access to the following data management systems and access standards.

- Local POSIX storage in form of a directory,
- UNICORE storages on remote UNICORE deployments,
- iRODS data management systems,
- Hadoop distributed filesystems,
- the Amazon S3 storage interface, and
- the CDMI interface.

Those can subsequently be integrated in a data life cycle. Via these capabilities an important requirement is fulfilled as it makes UNICORE generally applicable in regard to data resources.

Metadata Management

In the following, specific technologies are mentioned to perform the common steps that enable the search for metadata that was extracted from source formats (see Figure 3.6).

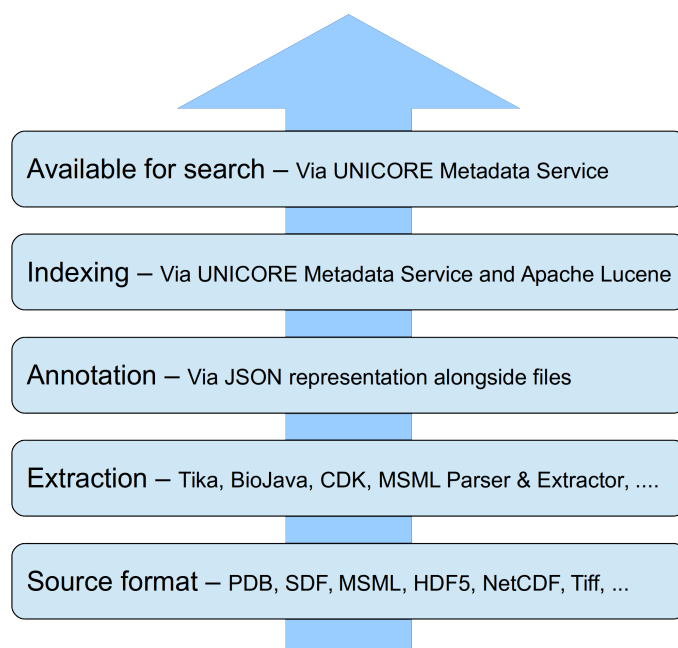


Figure 3.6: The figure depicts various abstraction layers reaching from data and metadata formats at the base to search availability in ascending layers. Example formats and technologies are noted.

Especially the UNICORE metadata service [NS10] is recommended to be employed as a core technology. It follows the combined metadata proximity approach (see Section 2.2.3) and supports metadata abstraction while at the same time enabling an advanced level of performance and resilience. The metadata service abstracts from metadata systems and access standards. A high number of source formats can be handled by Apache Tika, employed by the UNICORE metadata service. It is a highly generic framework for metadata extraction and abstracts from over a thousand supported file types ranging from markup languages over proprietary formats to media and scientific formats. Scientific format examples are listed in the following [Tik15]:

- Directory Interchange Format (DIF)
- Geospatial Data Abstraction Library (GDAL)
- Hierarchical Data Format (HDF)
- Network Common Data Format (NetCDF)
- Matrix Laboratory (MATLAB)

Tika offers its vast extraction capabilities via a straight forward yet powerful parser interface. Its single parse method enables streamed input parsing for efficiency, including extra information in the extracted content, returning extracted metadata, and fine-tuning the parser process. JHOVE, as another metadata extraction framework (see Section 2.2.3), supported by large libraries, might be worth considering for inclusion. Examples for further, use case specific extraction applications, are listed in the following:

- BioJava [HDP⁺08],
- Chemistry Development Kit (CDK) [SHK⁺03] and

- MoSGrid specific components as described in Section 4.2

Extraction via Apache Tika within the UNICORE metadata service is executed on the server where the service is running. It may be worthwhile for performance and resilience reasons to execute extraction methods in a distributed way on HPC nodes instead of a central server. This aspect is conceptually discussed in detail in Section 3.3.

Annotation with the UNICORE metadata service is done via a JSON files that is situated besides the source file itself. It is named the same as the file it references but with the distinction that its name additionally starts with a "." and ends with ".metadata". JSON (JavaScript Object Notation) [Cro06] is an open and human-readable data exchange format. In contrast to XML it is a low-overhead format due to its simplicity. Information is expressed as attribute-value pairs. Values can be optionally nested in order to represent complex information structures.

For indexing, the UNICORE metadata service provides a programming interface that can be implemented for underlying technologies. Currently, UNICORE's default implementation employs Apache Lucene [MHG10] as indexing and search technology that provides a high performance [MHG10]. Together with the metadata in JSON format, Lucene enables a schema-free way of handling metadata. Though not a recommendation, but a plan that shows UNICORE's extensibility; another implementation of the interface for the KIT Data manager repository system (see Section 2.2.3 and 3.2.2) is currently under design within the MASi research infrastructure (see Section 6.2).

Furthermore, advanced search capabilities are provided via the UNICORE metadata service. These capabilities include search for basic metadata such as filename and creation data, full text search, and the possibility to use boolean queries, wild characters and range and fuzzy queries. Search queries can be issued via a commandline client, workbench, web client, Java, and REST API.

This whole chain from source format to searchability can be automated by being triggered depending on the use case, for example, via a bash script that initiates the respective UNICORE commandline client command. Or, it can be the UNICORE submitter of gUSE using the UNICORE Java API to be triggered once a workflow is finished. Also, the data oriented processing feature of UNICORE (see Section 2.3.2) can be utilized. Here, a rule can be defined that automatically triggers the metadata extraction, annotation, and possibly indexing when a file appears in an observed directory.

Computing and Workflow Management

High Performance Computing resources can be accessed by transparently making use of underlying batchsystems while the utilization of Cloud and Big Data frameworks is also possible. Computing resource abstraction is enabled via the UNICORE middleware that directly interface with batch systems via a component running on the cluster's login node. Examples of supported batchsystems on cluster and supercomputers are the following:

- SLURM,
- LSF,
- Oracle Grid Engine,

- CCS,
- PBS, and
- TORQUE

Besides the batchsystem category, support for Big Data frameworks such as Hadoop [Whi12] via YARN [VMD⁺13] are currently being released in UNICORE and will subsequently be available within implementations of the dissertation concept when UNICORE is utilized. Computing and data resources can be combined via workflows for the benefit of the user. Recommended workflow engines (see Section 2.3.3) are those of the gUSE/WS-PGRADE science gateway framework and of UNICORE. The gUSE/WS-PGRADE engine integrates well as it is an integral part of the science gateway framework mentioned below. The UNICORE internal workflow engine has the advantage of being natively integrated with the UNICORE middleware.

Security and Utilization

With UNICORE as recommended technology, as deployed in the MoSGrid data life cycle (see Section 4) and the Human Brain Project HPC platform, examples of the available security mechanisms are the following. Authentication can be handled via trusted certificates or by the Unity identity management system via trusted federated identities. Authorization within UNICORE is covered via its XUADB user database. Its information may come from a VOMS server, Unity or may be manually inserted. Or, an external authorization source such as LDAP as in the case of the Human Brain Project, or a science gateway as within the MoSGrid data life cycle, may be used. Single sign-on capabilities are available via SAML trust delegation assertions and proxy certificates.

The main recommendation for a utilization technology is the gUSE/WS-PGRADE science gateway framework. It can be utilized as part of building feature-rich data life cycles as in the case of the MoSGrid concept implementation. Custom portlets can be created to provide specialized user interfaces to make available the underlying gUSE/WS-PGRADE and UNICORE services in a user-friendly way. Further clients can be the UNICORE commandline, rich, and portal clients as well as the Java and REST APIs.

4 Implementation within a Complex Data Life Cycle

An implementation of the generic metadata approach was carried out within the complex and distributed MoSGrid data life cycle showing the feasibility and applicability of such an implementation. The implementation incorporates the overall concept characteristics (see Section 3.3) by following the design guide (see Section 3.4.2) and making use of the technology recommendations (see Section 3.4.3). First, the central MoSGrid data format, the Molecular Simulation Markup Language (MSML) is described in detail in Section 4.1. The description includes the conversions to and from MSML. Then, the specific metadata extraction, annotation, and indexing within MoSGrid is described in Section 4.2. The integration of the metadata service is described in Section 4.3 and how the immediate and seamless utilization of search results is enabled. The fully automated metadata integration, that importantly includes the security and single sign-on infrastructure, and an example usage scenario are presented in Section 4.4. MSML, its parser and adapter were co-designed and tested by the author. The specific metadata extraction, annotation, indexing, access, and search were designed, implemented, and integrated by the author. The result utilization scheme was designed, integrated, and tested by the author. This implementation adds metadata capabilities to the MoSGrid data life cycle and is essential as now data can be found by its content and seamlessly used as input in workflows.

4.1 The Molecular Simulation Markup Language as Information Hub

4.1.1 Molecular Simulation Markup Language

The MoSGrid data life cycle supports a variety of applications and workflows to simulate chemical phenomena. Most of these workflows produce different output formats while also requiring input in specific formats. This situation includes many formats and makes creating large workflows (see Figure 2.16) highly complex. Numerous data conversions between individual applications of chemical simulation suite would have to be made in a potentially pairwise way. To avoid the need for such an inefficient conversion situation, a common hierarchical XML-based data format called MSML (Molecular Simulation Markup Language) [GBG⁺14, KGG⁺14] was designed and introduced to the MoSGrid data life cycle. With MSML, a full description, independent from the specific chemical simulation applications and their results, became possible. MSML serves as the central information hub and data format for workflows within MoSGrid. Furthermore, instead of many-to-many format transformations between every individual application formats, now, parsers and adapters just from and to MSML are utilized as described in the next section. These enable one-to-many format conversions and, thus, saves significant development efforts. Their integration is depicted in Figure 4.1.

MSML is a dialect of the XML-based description language CML (Chemical Markup Language) [MRR99] to meet the needs of the computational chemistry community. MSML is fully compatible with CML. MSML is composed of three main components, namely schema, conventions, and dictionaries. The possible combinations of XML elements and attributes are described in the schema. XML elements may contain components. The relationship of elements and components is defined in the conventions with the categories mandatory, optional, and forbidden. For example, a condition is defined in the compchem convention; If the attribute dictRef in the module element is set to jobList, then at least one module element with dictRef set to job must be present. As a jobList in a module element describes a workflow, this conditions means that every workflow has to have at least one job. The semantics of elements in MSML are described using dictionaries. For example, different simulation applications require an input parameter denoting the number of iterations for a specific task. However, one tool uses values such as "few" or "many" while another uses numerical values. These mappings are described in application specific dictionaries. Such a dictionary has to be set up for each application containing entries for identifier, iterations, and the definition of possible values.

An MSML template scaffold, representing a workflow, is shown in Figure 4.2. The element "cml" is the root element and contains one "joblist" which details the individual jobs and may contain a parser-Configuration element. Each of these tasks has three sections. The "environment" section details the job requirements with respect to nodes, cores, memory, and walltime. The "initialization" section describes chemical structures such as small or large molecules in the form of an parameterlist with parameters of the types matrix, array or scalar. The section may also contain the adapterConfiguration and the parserConfiguration, each in the form of a parameterlist. Each job has a "finalization" section with a propertylist of the results of a job.

4.1.2 Integration with Data Life Cycle

MSML is utilized as an intermediate format by MoSGrid tools such as the Portlet-API, Templatedesigner, and the Generic Parser. MSML importantly also acts as the source format for metadata extraction (see Section 4.2). Figure 4.1 depicts the relationship between these individual components.

MSML templates are created by MoSGrid developers for each new workflows. The templates define the characteristics of a molecular simulation such as the number of iterations, basis set or the required main memory. Some values may be mandatory and some optional for a workflow execution. Default values are also possible. These are extensively utilized in order to further simplify the initiation of workflows. Due to the complexity and modular structure of MSML templates, common XML editors are unreliable as they can not take dictionaries and conventions into account. Therefore, to support the creation of new MSML template the so-called Templatedesigner was implemented within MoSGrid. It helps in creating templates by checking for semantic and syntactic inconsistencies and reduces the risks of errors significantly.

Concrete workflows, loaded from the central MoSGrid workflow repository, are described in MSML templates that reference dictionaries. Using these templates, the Portlet-API (see Section 2.6.5) creates workflow parameter input masks on the fly. This, for example, includes allowed parameter ranges for numerical values or specific keywords for entries denoting task iterations. For every application supported

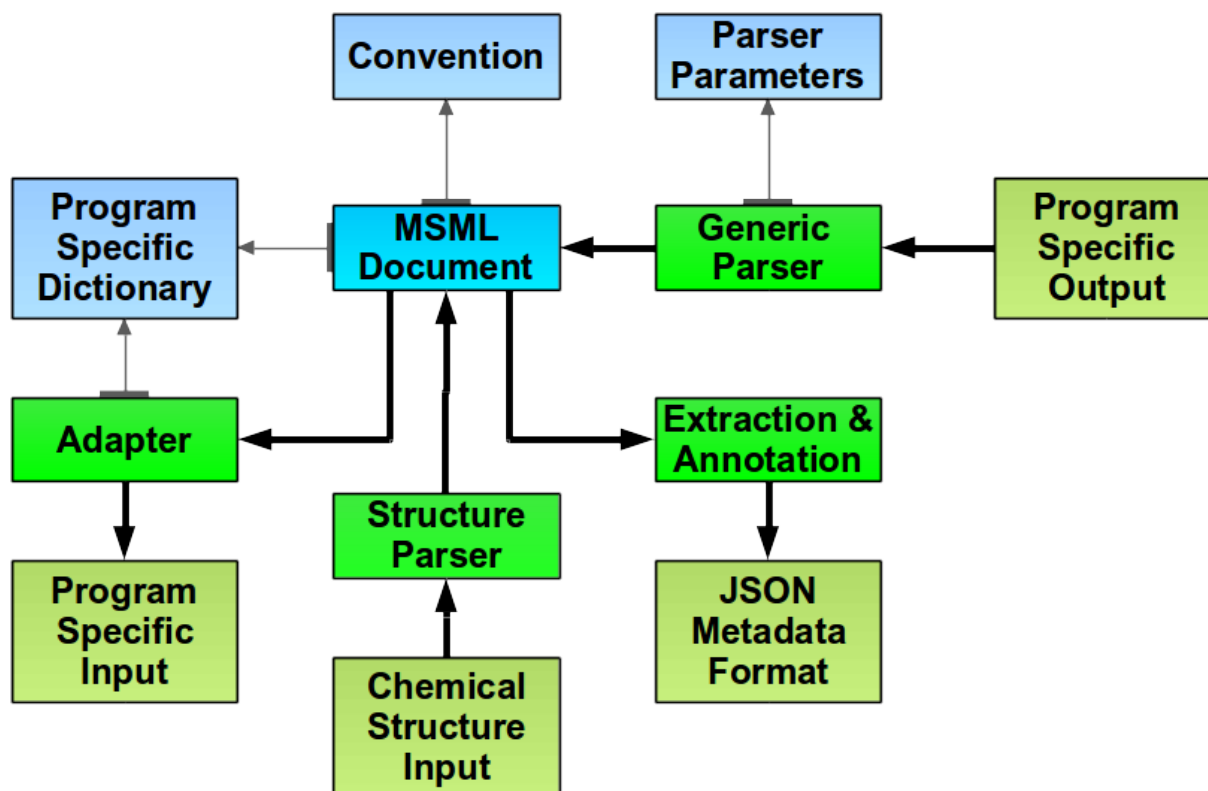


Figure 4.1: MSML as the information hub of the MoSGrid data life cycle is at the center and components in the periphery of this graphic. These components enable the integration of MSML within the MoSGrid data life cycle.

```

<?xml version="1.0" encoding="UTF-8"?>
<cml convention="convention:compchem">
  <module dictRef="compchem:jobList">
    <cmlx:parserConfiguration/>
    <module dictRef="compchem:job" id="Job1">
      <module dictRef="compchem:environment">
        <propertyList/>
      </module>
      <module dictRef="compchem:initialization">
        <parameterList/>
        <cmlx:adapterConfiguration/>
        <cmlx:parserConfiguration/>
      </module>
      <module dictRef="compchem:finalization">
        <propertylist/>
      </module>
    </module>
    <module dictRef="compchem:job" id="Job2"/>
    ...
  </module>
</cml>

```

Figure 4.2: The structure of a simplified example MSML template document is shown. [KGG⁺14].

```

<?xml version="1.0" encoding="UTF-8"?>
<cml "omitted for reasons of space" ...>
  <module dictRef="compchem:jobList" id="nwchem_generic_final_2014-03-20-143337"
    cmlx:displayName="NWChem" title="Submission of prepared job-files. (No postprocessing)">
    <description cmlx:plainText="With this workflow you may upload a nw-file used as input for NWChem-6.3."/>
    <module dictRef="compchem:job" id="nwchem">
      <module dictRef="compchem:environment">
        <propertyList>
          <property dictRef="env:medium.nodes" cmlx:editable="true">
            <scalar dataType="xsd:integer" units="si:none">1</scalar> </property>
          <property dictRef="env:medium.cores" cmlx:editable="true">
            <scalar dataType="xsd:integer" units="si:none">4</scalar> </property>
          <property dictRef="env:medium.walltime" cmlx:editable="true">
            <scalar dataType="xsd:integer" units="nonsi:min">240</scalar> </property>
          <property dictRef="env:mediummem.memory" cmlx:editable="true">
            <scalar dataType="xsd:integer" units="si:none">2000</scalar> </property>
        </propertyList> </module>
      <module dictRef="compchem:initialization">
        <parameterList>
          <parameter dictRef="nw3:nomolecule">
            <scalar dataType="xsd:boolean" units="si:none">true</scalar> </parameter>
        </parameterList>
        <cmlx:parserConfiguration>
          <parameter dictRef="parser:parserConfig">
            <scalar dataType="xsd:string" units="si:none">\Qqc/nwchem.xml\E</scalar> </parameter>
        </cmlx:parserConfiguration>
        <cmlx:uploadList> <cmlx:jobInputUpload fileType="nw" port="input.nw"/> </cmlx:uploadList>
      </module>
      <description cmlx:plainText="Please provide a proper nw-file for NWChem-6.3."/>
    </module>
  </module>
</cml>

```

Figure 4.3: This figure displays a specific NWChem MSML scaffold [?]. Default configuration values are shown that define the computational requirements of a task within a workflow.

in the MoSGrid data life cycle a dictionary was created (see Figure 4.1).

MSML documents may include "finalization" sections for finished jobs. Such a section represents the results of the job. As such it is directly relevant for the extraction of metadata described below. The section is generated after the job is finished by the Generic Parser utilizing the parserConfiguration in a MSML document. This configuration defines rules on how to extract the relevant information from the application output format. Based on such MSML files, a so-called adapter is able to generate input files that are specific to chemical applications and the respective parameters. In the adapter configuration, information about methods for transforming parameters to application specific input are specified. This configuration is contained within the parameter list of the initialization section. For new application formats, parser configurations need to be created for format conversions from and to MSML.

MSML supports the three major chemical domains, namely Quantum Chemistry, Molecular Dynamics, and Docking (see Section 2.6.6). A detailed MSML template example for the Quantum Chemistry NWChem application is shown in Figure 4.3. It includes default values for the required nodes, cores, walltime, and memory requirements. The Generic Parser configuration is also included as well as the workflow requirement of an NWChem input file. During the runtime of the workflow the generic parser appends the module tag "compchem:finalization" and files in information from the NWChem output file.

A main function of MSML is being the central MoSGrid information hub. All aspects of a specific computational chemistry simulation are covered by MSML. Therefore, it strongly fosters reproducibility. For example, for a molecular dynamics simulation, MSML stores the following information;

- what molecules were simulated,

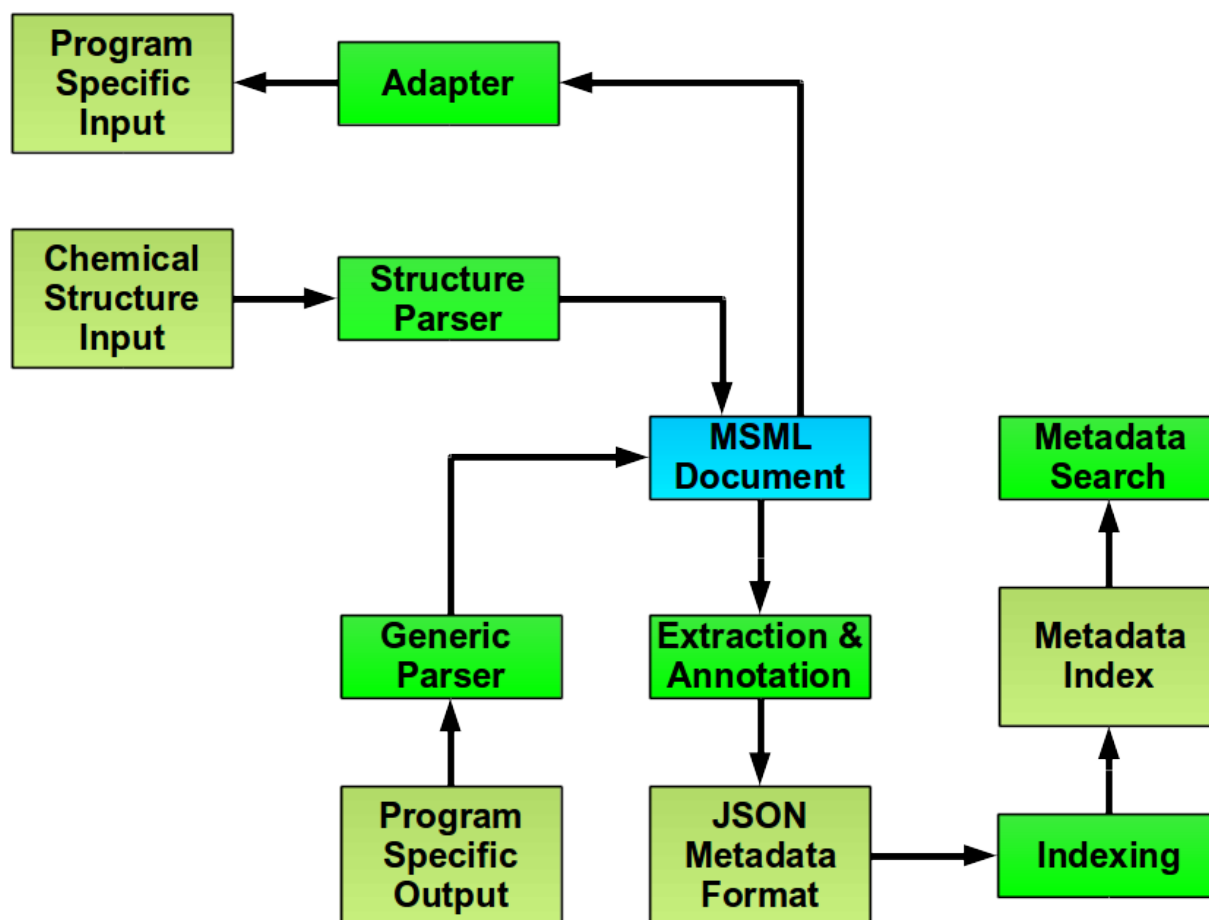


Figure 4.4: As part of the concept implementation within the complex MoSGrid data life cycle, the complete component chain for extracting, annotating, and indexing metadata based on the MSML format, the MoSGrid information hub, is depicted.

- the utilized solvents and force fields,
- the settings for electrostatics and van der Waals,
- temperature and pressure,
- the utilized applications for preparation and simulation,
- the employed compute resources such as core number and memory,
- what output structures were generated, and
- the characteristics of the simulated potential energy.

Filled MSML templates are stored in the distributed data management system at the end of a workflow. A user can search for information to find relevant data. How this process is enabled is described in the following sections.

4.2 Metadata Extraction, Annotation, and Indexing

4.2.1 Extraction and Annotation

Chemical structures such as molecules are represented in corresponding file formats. A parser was developed that converts such formats to MSML. The widely used PDB and SDF files are currently supported. The BioJava [HDP⁺08] framework is utilized for PDB support and the SDF support is based on the Chemistry Development Kit (CDK) [SHK⁺03]. Chemical structures are the basis of chemical workflows. At the beginning of a workflow, a user provides such structures with PDB or SDF files. The parser is automatically executed on these files in order to extract the structure information and store it in the MSML template (see Figure 4.4).

Output file formats of molecular simulation applications have a high degree of variety. During the runtime of a workflow the so-called generic parser is utilized to extract relevant information from such formats. The extracted information is continuously included in the workflow's MSML template. Regular expressions are employed as a highly flexible mean to specify separators between elements, replacement operations and extractions of specific strings. The regular expressions are stored in the above mentioned parser configuration files to be specifically referenced in the corresponding MSML section of a workflow job. When the generic parser is automatically executed, it loads the parser configuration file and extracts the information. This information is added as a finalization section to the job's section within the workflow's MSML template. The parsing is executed on High Performance Computing resources as it may be compute intensive, especially with a large number of parallel extraction tasks. A performance evaluation is described in Section 5.2.2.

At the end of the successful workflow, relevant information from the workflow's MSML document is extracted as metadata (see Figure 4.4). In order for this process to be as generic as possible, the relevancy of information was not judged as it is highly dependent of the use case in question. All information is extracted that can be meaningfully used for search queries. This specifically excludes chemical structure information as these can hardly be meaningfully searched for. The Java SAX parser (`javax.xml.parsers.SAXParser`) was chosen for the extraction process due to performance reasons within the MoSGrid data life cycle. SAX stands for "Simple API for XML". In contrast to a DOM (Document Object Model) parser, a SAX parser is faster and less memory-intensive as it avoids fully loading the XML document into memory. It subsequently avoids the need to manage a document representation in memory. Instead, a SAX parser traverses through a XML document and utilizes callback methods in order to return notifications about the structure of the document. The following methods are to be overwritten in order to implement a concrete SAX parser.

- `startDocument()`,
- `endDocument()`,
- `startElement()`,
- `endElement()`, and
- `characters()`

```
{
  "md-input:min.stepsize": "0.1",
  "title": "minimization",
  "md-input:ESA.outerRadius": "1.0",
  "md-input:NSA.updateFrequency": "1",
  "md-input:NSA.radius": "1.0",
  "md-input:MinimizerAlgorithm": "Steepest descent",
  "md-input:VdWA.outerRadius": "1.0",
  "md-input:ElectrostaticsAlgorithm": "PME",
  "md-input:min.tolerance": "1.0",
  "md-input:NumberOfSteps": "500",
  "md-input:NSA.PBC": "XYZ",
  "md-input:NeighbourSearchingAlgorithm": "Grid"}
}
```

Figure 4.5: A specific example of extracted metadata in JSON format is shown.

`startDocument()` and `endDocument()` are called at the beginning and end of a document respectively. At the beginning and end of each XML element the `startElement()` and `endElement()` methods are called. The `characters()` method is called when text content is registered between the start and end tags of an element. For the specific MSML metadata extraction, these methods were implemented to reflect the above mentioned relevant information. At the end of the workflow, the concrete MSML SAX parser is executed and consequently, step by step, extracts all relevant metadata from the MSML file. The parser adds this metadata as key-value pairs to a hash map structure of two string values. At the end of the extraction process, a new file is created with the naming scheme that adds a leading "." and an ending ".metadata" to the name of the workflow's MSML file. For example, "gromacs_min_steep.cml" becomes ".gromacs_min_steep.cml.metadata". The created hash map containing the extracted metadata is converted to a concrete "JSONObject". This object is then converted to JSON text and finally written into the new file (see Figure 4.5 for an example). This file is stored next to the actual data. By virtue of the naming scheme the MSML document is annotated by the JSON metadata file (see Figure 4.4).

4.2.2 Indexing

Indexing the annotated metadata in the form of JSON files exposes it for search requests by users. At the end of each workflow, the gUSE UNICORE adapter executes the `startMetadataExtraction` method of the `MetadataClient` object within the UNICORE Java API. This method instructs the UNICORE metadata service to index new JSON files. This indexing is narrowed down to the user's home directory for performance reasons and widened to all connected UNICORE metadata services in order to expose the new metadata for search everywhere. The `startMetadataExtraction` method actually triggers the metadata extraction via the default extraction framework Apache Tika as well, but since the corresponding metadata files were already created during the workflow's runtime, Apache Tika is not triggered for those. By the indexing, the metadata becomes automatically available in the index of the underlying Apache Lucene and, thus, becomes discoverable. The performance of the indexing was evaluated and is described in Section 5.2.3.

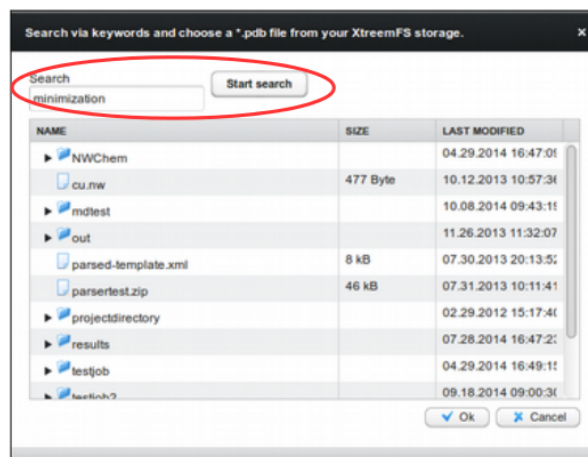


Figure 4.6: The metadata search interface within the MoSGrid data browser is shown.

4.3 Metadata Service Integration and Result Utilization

4.3.1 Search Interface and Metadata Service Access

A search interface was implemented and integrated. It enables an easily usable method to search for data via metadata stored in Apache Lucene via the UNICORE metadata service. The MoSGrid data browser was extended by search field and button (see Figure 4.6). The click on the button passes the entered search term, the available SAML assertion and the user's ID to the UNICORE interface libraries. A metadata search command is transparently issued to the connected UNICORE metadata service which in turn passes the search term to the underlying Apache Lucene. The corresponding result set is returned and used to narrow down the view in the data browser in order to only display files that correspond to the search query. The UNICORE interface libraries were extended in order to enable an efficient integration. A public "get" method was defined to enable direct access to the search result set. This method avoids the need of additionally parsing output strings. As a search example, a user might look for a specific chemical "basis set" that was utilized in a molecular simulation. He just enters a term such as "6-31G" and links to fitting simulations, including input and output files, are returned.

4.3.2 Search Result Utilization

An important feature of the implementation of the concept is the possibility to seamlessly use search results as input in further workflows. The search interface (see previous Section 4.3.1) is integrated directly into the submission view of each domain portlet via the Portlet-API (see Figure 4.7). A user can directly utilize the search capability to narrow down the data view to choose an input file. This file is then uploaded to the portal for preprocessing while the file reference is used as input in the current workflow. This integration was made possible by the design and implementation of a feature for using location independent file references. This "xtremfs://" schema describes the relative location of a file within XtremFS. This specification includes the location from where to download input data and upload result data. The mechanism enables the separation between specifying the XtremFS instance to be

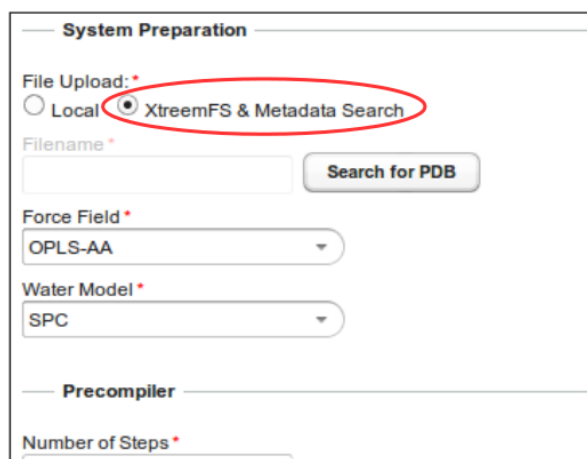


Figure 4.7: The data browser that includes the search interface can be chosen in order to find input data based on its content.

used and the relative path within XtreemFS. In the `uas.config` configuration file of each UNICORE/X server component, either a local mount point (i.e. `/mnt/mosgrid/`), on the server where the TSI runs, is specified via the `"coreServices.xtreemfs.mountpoint"` parameter. This enables the use of the simple and efficient local copy command (`"cp"`) to copy files to or from the UNICORE job working directory. Or, `"coreServices.xtreemfs.url"` is specified in order to indicate that a remote UNICORE storage should be used. This method is applicable when a local XtreemFS client is unavailable. Now, relative file references such as the metadata search returns can be utilized as input in workflows. Additionally, the `"xtreemfs://"` schema mechanism automatically enables advanced efficiency and portability of workflows.

4.4 Overall Integration and Example Usage Scenario

4.4.1 MoSGrid Data Life Cycle Integration

The overall integration into the MoSGrid data life cycles is as follows. Templates of the chemical format MSML define and reference workflows and utilize dictionaries for application parameter characterization. MSML is stored in the data management system of MoSGrid, XtreemFS. A specific MSML document is created when the workflow is initiated and it is enriched with information throughout the runtime of the workflow. The actual metadata is stored besides the data in JSON files in the data management system. The UNICORE metadata service indexes these files via Apache Lucene at the end of each workflow to provide discoverability based on search terms. This implementation of the dissertation approach constitutes a combined metadata proximity approach. Location independent access to data in XtreemFS, from within UNICORE, is provided by the `"xtreemfs://"` schema. It enables the seamless re-use of search results. The generic concept and its implementation are completely integrated with the underlying single sign-on implementation.

4.4.2 Example Usage Scenario

A complete usage scenario is depicted in Figure 4.8 and described in the following. First, the user chooses the portlet of the domain (see Section 2.6.6) he is interested in. The user is immediately in the "Import" view while the corresponding workflows were automatically loaded in the background by the Portlet-API (see Section 2.6.5). Then, the user has to choose a workflow (Figure 4.8 upper left). The list can be narrowed down by choosing a tool suite while a modifiable default name is given. After clicking on the "Import" button, the "Submission" view becomes visible while in the background a concrete MSML template is created. In order to configure a workflow for submission, the user has to upload one or more files as input. Besides the possibility to upload a local file, the "XtreemFS & Metadata Search" option can be chosen (Figure 4.8 upper right). After clicking on "Search for PDB" an overlay window appears (Figure 4.8 middle left) showing the content of the data management system XtreemFS that is available to the user. The user can search for data by its content via the search field on the top (Figure 4.8 middle left). After a search term was entered and the button clicked, only those files remain in the view that correspond to the search terms (Figure 4.8 middle right). In the background the Apache Lucene index was queried via the UNICORE metadata service and the result file references were incorporated with the data view. When a file was chosen (Figure 4.8 lower left) and confirmed, it is loaded for preprocessing. This is visualized by displaying the file name in gray (Figure 4.8 lower left). After the remaining parameters values are entered (Figure 4.8 lower middle), the submit button has to be clicked for submission of the now configured workflow (Figure 4.8 lower right). In the background the individual workflow tasks are submitted to fitting resources. During the runtime of the workflow, information is continuously added to the MSML template reflecting the progress of the workflow. Once the workflow is finished, the metadata is automatically extracted, annotation, and indexed. Now, when a new workflow is to be initiated, the previously created data can be immediately found and seamlessly utilized as input.

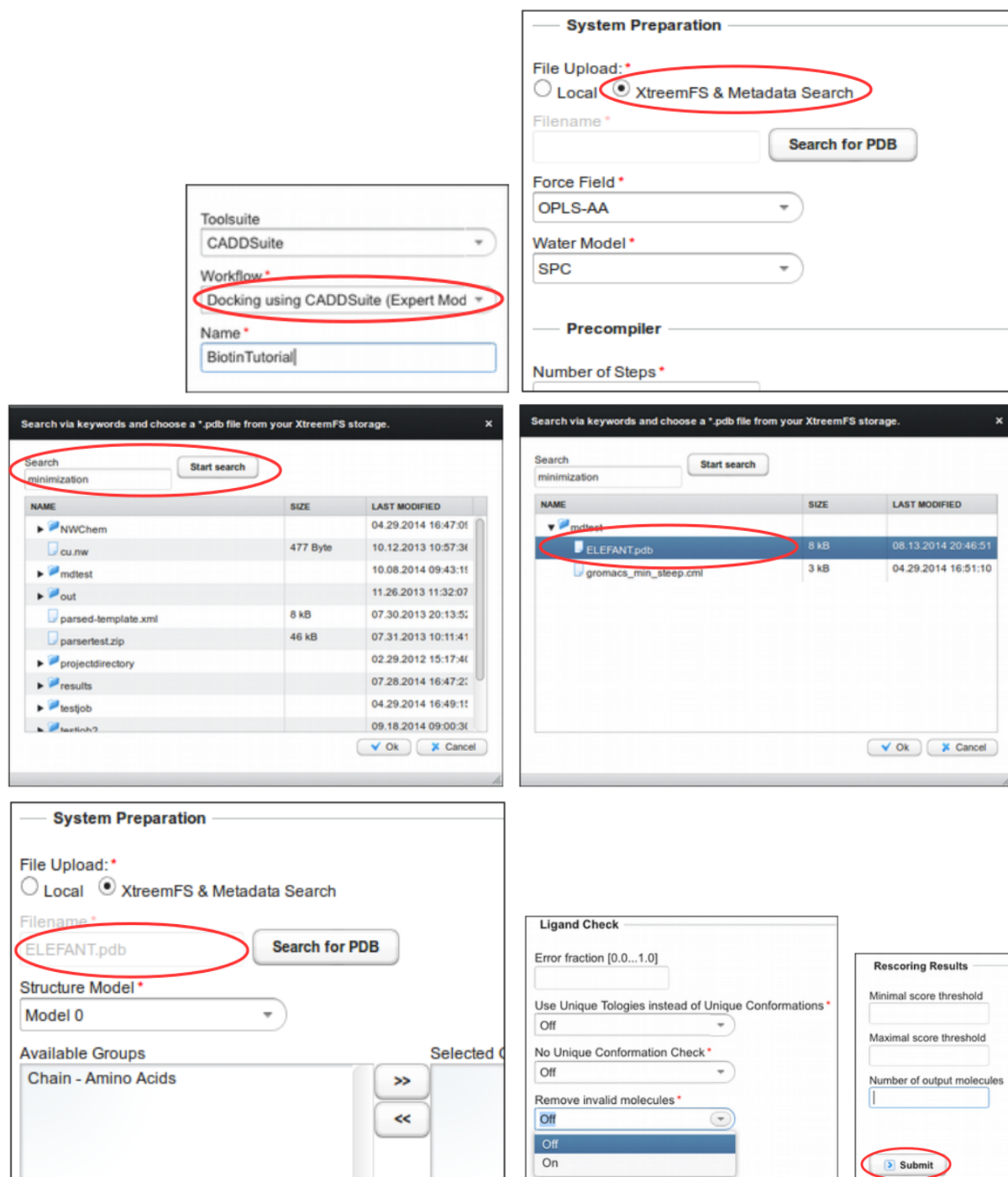


Figure 4.8: A sequence of actions is shown ranging from choosing a workflow, seamlessly searching for input files via metadata, adjusting parameters, and finally submitting the workflows.

5 Evaluation

This chapter evaluates both theoretically and practically the generic concept and its implementation within the MoSGrid data life cycle. A major point is that both concept and implementation were integrated in a wider context of technologies such as UNICORE, Apache Lucene, Apache Tika, Liferay, and gUSE/WS-PGRADE. This way, besides metadata management, also surrounding data life cycle capabilities are available. The concept and implementation is put in contrast with alternative approaches in Section 3.2.2. The favorable adaptability characteristics are described in Section 5.1 along with a concrete further adaptation outlook. In Section 5.2, besides general consideration, the performance of the implementation is evaluated and shown to scale well. Sustainability and resilience aspects of the concept and its implementation are described in Section 5.3 and 5.4 respectively. Finally, the efficiency of usage is shown in Section 5.5.

5.1 Adaptability

5.1.1 Concept Adaptability

The generic concept significantly fosters the adaptability of metadata management in scientific data life cycles. The concept is widely agnostic with respect to data management systems. A requirement is that such a system either offers a POSIX interface or one that the UNICORE middleware directly supports such as the ones for iRODS, Hadoop distributed filesystem and Amazon S3, while support for the CDMI [CDM13] interface is currently being released. As UNICORE is under the free and open source BSD license, support for further data management protocols can be implemented via its Storage Management Service (SMS) interface. Another requirement is that such a system stores data in the form of files to allow for the combined proximity approach.

Currently, the UNICORE-native metadata system based on Apache Tika and Lucene is supported. The integration with further systems can be implemented using UNICORE's metadata interface. It was designed with adaptability as clear design goal. The existing metadata service is completely generic with respect to data and metadata formats. All relevant metadata is extracted and stored in as JSON format as key-value pairs next to the actual files. This way, the necessity of schema management is avoided. Per default the Apache Tika framework, which supports over a thousand formats, is utilized for extraction. The extracted metadata is stored in form of JSON files which are subsequently indexed to be available for search. Further formats can be supported by extending Tika, by integrating another extraction framework or by implementing custom extraction methods for scalability reasons. By using a custom method, the extraction processes can run in a distributed manner within workflows on High Performance Computing resources. This was done for the MSML metadata format of the MoSGrid data life cycle which by virtue

of the generic concept. This way, native support in MoSGrid for the three major chemical domains exists. These are the highly diverse domains of molecular dynamics, quantum chemistry and docking.

Regarding supercomputers and batchsystems, the concept is highly generic by virtue of the recommended UNICORE middleware. UNICORE is deployed on many supercomputers in the world with all major batchsystems being supported. Any kind of application, from small preprocessing tools to large scale simulations, and various CPU, GPU, and interconnect architectures are naturally supported. Workflows are supported by the UNICORE workflow system as well, as is any workflow management systems that utilizes UNICORE as backend technology. For example, the workflow engine within the gUSE science gateway framework can utilize UNICORE as backend technology to interface with supercomputers.

The concept generically integrates well with security measures throughout a data life cycle. Metadata in JSON format is secured via underlying access rights management methods of the utilized data management system, which methods are natively recognized. By default, the access to the central index is restricted to the user owning it. For authentication and authorization infrastructures including trust delegation to enable single sign-on, SAML assertions are supported. For example, MoSGrid aims to implement support for federated identity management via the Unity service and the distributed DFN-AAI infrastructure. Even with such a major upgrade, changes to the metadata management can be largely avoided. The mode of integration with other systems such as science gateways is highly adaptable as well. Access via a commandline client, Java, and Rest API is supported. The implemented search interface resides within a JSR168/JSR286 [AH03, H⁺05] portlet. These standards make the concept generic with respect to the deployed portal technology, assuming the underlying services are available in such a case as well.

Summarized, the metadata concept is generic in multiple dimensions and, thus, highly adaptable while providing an advanced level of security.

5.1.2 Adaptation Outlook for a High-Throughput Big Data Microscopy Use Case

At the MPI-CBG (Max Planck Institute of Molecular Cell Biology and Genetics) Selective Plane Illumination Microscopy (SPIM) [HSDB⁺04, HS09, WMH13] is being developed and utilized. In a common configuration, SPIM devices can produce 0.85 GB/s and about 10 files/s. Continuous operations are the goal which will produce about 2 petabyte and 26 million files monthly. Various microscopes already exist with more being planned.

Based on the generic concept, the technology recommendation UNICORE, the experience gained by the implementation within the MoSGrid data life cycle and together with CBG information technologists, an architecture is being designed to preprocess, manage, analyze and safely store SPIM data. The microscopes are controlled via workstations running Windows operation systems. Currently, the immediate data is stored on local SSD raids with the full speed. When the SSD raid is full, the microscope stops, and the data is manually staged to the larger HDD raid. Furthermore, its is transfered to remote cluster storages for analysis. This is planned to be automatized.

The data needs to be reduced, compressed, and preprocessed close to the microscope in an automatic way. This is because the data is too large to be stored indefinitely. The data-driven computing concept (see

Section 2.3.2), which is supported by UNICORE and iRODS, is a perfect fit for this scenario. UNICORE is currently being evaluated for this capability. It can be naturally integrated with vast High Performance Computing resources and it is built to deal with large amounts of data as for example within the Human Brain Project. Loss-less compression is a more obvious choice in contrast to lossy reduction as scientists are extremely hesitant to throw away data. An example of preprocessing is the alignment of the same object across various images. As data management system, dCache and iRODS are two choices that are widely used, mature, with advanced features and access methods. Both offer a high performance in storing petabytes of data.

Metadata management is an immediate and central challenge to organize SPIM data based on its content. It is especially challenging as SPIM data is continuously growing with increasing data rates and number of microscopes. Currently, the data is stored on various large and professionally administrated storages but they are only loosely coupled. This makes it increasingly more challenging to organize the data properly and for scientists to find it again. Data generated by microscopes is often already annotated with metadata, either in OME (Open Microscopy Environment) [GAB⁺05] format, in a proprietary one or metadata is completely missing. Specific extraction tools for biology formats such as BioJava or the Fiji Plugin Bio-Format exist. These tools are able to convert various formats to the standard OME data format that includes metadata. The UNICORE metadata service can support this scenario by being completely flexible with respect to data and metadata formats. The service is able to handle large amounts of data and provides transparent metadata indexing and exposure for search via Apache Lucene.

With respect to computing management, data accessed via metadata can be processed transparently via UNICORE in its capacity as a computing middleware. Jobs that require an arbitrary number of cores can be issued. This also holds true for workflows that are a graph of jobs. Workflows are utilized to encapsulate complex analysis protocols to make them easy to modify and run on large HPC resources with just a few clicks. Such workflows enable advanced users and data maintainers to perform previously inefficient pre- and postprocessing of large amounts of data. At the same time end users are supported via such encapsulated analysis protocols and via easy access to High Performance Computing resources. Utilizing workflows is less error prone than manually doing the steps via SSH and the batch systems.

The advanced automation features via workflows can be integrated into the working environments of the users. Examples are KNIME, Galaxy or custom analysis tools such as Motion Tracking. A commandline client, the data oriented processing, the Java or Rest API can be used for the integration. Additionally, a graphical client as well as a web portal exists. The concept depicted in this dissertation is a natural fit for implementation within this SPIM use case. It is currently explored within the MASi research infrastructure project (see Section 6.2).

5.2 Performance

5.2.1 Various Aspects

Various aspects play into the overall performance of a data life cycle. The MoSGrid implementation of the generic metadata concept seamlessly blends into the overall MoSGrid life cycle. For example, the

central search index using Apache Lucene provides a high search performance [MHG10] over metadata from potentially highly distributed data. Bottlenecks during a high number of search queries can be avoided by the possibility of utilizing several instances of the search indexes by employing a number of metadata service.

One downside of working within the UNICORE environment is a noticeable latency while interacting with a UNICORE service. This latency is caused by the fact that UNICORE was implemented following the Web Services Resource Framework (WSRF) specifications [Ban06]. WSRF provides the advantages of being strongly typed, including the possibility of message validation, advanced SOAP (Simple Object Access protocol) message mechanisms and well established web service security with SAML for trust delegation [Sch14]. Besides the downside of being highly complex, it is also CPU intensive to a high degree due to XML processing [Sch14]. This causes a latency overhead which is aimed to be significantly reduced within the Human Brain Project 2.7.2. There, UNICORE is currently extended to provide a Representational State Transfer (REST) [BB08] API. The advantages are that REST is only weakly coupled, inherent HTTP benefits are used, multiple authentication options exist, various representations of messages and resources are supported and clients can be implemented in basically all languages [Sch14]. A downside is that currently no standard solutions for trust delegation exists [Sch14]. Due to the much lower computational overhead the request latency is reduced [SBR14].

Another performance aspect is the avoidance of potential bottlenecks in extracting metadata from application output files and MSML. This is done by performing the extraction steps in a distributed way within workflows. Section 5.2.2 shows that the overhead is quite small within a workflow. When a high number of workflows is executed, performing the extraction steps on a central metadata service server would potentially lead to bottleneck situations. This scenario is completely avoided by the distributed nature of the metadata extraction. The metadata indexing, on the other hand, imposes a very small overhead, as shown in Section 5.2.3. Performing the indexing on a central server, even with parallel indexing processes, is safe for the foreseeable future.

5.2.2 Metadata Extraction and Annotation

The Molecular Simulation Markup Language (MSML) is described in Section 4.1. In this section a performance evaluation is presented that covers the extraction from application specific output to MSML and from MSML to metadata in JSON format including the annotation. Extracting and annotating metadata are crucial steps that are abstractly discussed in Section 3 and their implementation and integration described in Section 4. In this section the goal is to investigate the performance properties of these central steps for enriching the MSML template with metadata during workflow runtime and possibly uncovering bottlenecks. A manuscript presenting these measurements was accepted for publication [?].

During the execution of each individual workflow task on a High Performance Computing resource, the workflow's MSML document is continuously extended with relevant metadata. It is extracted from a specifically formatted output file of a molecular simulation tools which was executed within a particular workflow. The extraction is done by the generic parser component (see Section 4.1). It converts the application specific output to XML which is subsequently inserted into the "finalization" section of the corresponding application section in the MSML document. This MSML document represents the whole

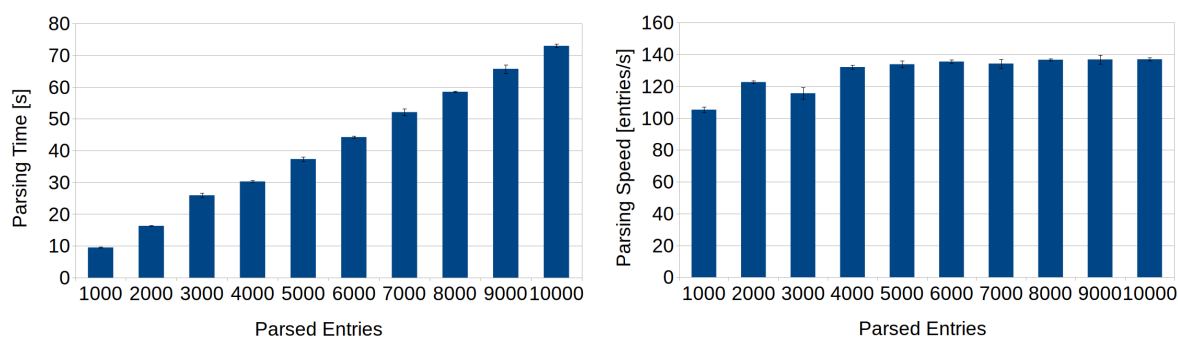


Figure 5.1: A performance evaluation of the extraction and annotation step from the central MoSGrid information hub MSML to the JSON metadata format is presented. Depicted is the parsing time in seconds on the left side and the corresponding parsing speed in processed entries per second on the right side. 10 datasets with 1000 to 10000 entries in steps of 1000 are shown. Each measurement was repeated 10 times and the respective average and error bars are shown. At this large benchmarking scale, that will probably not be reached in production use cases in the foreseeable future, the imposed overhead by the metadata extraction and annotation is insignificant in contrast to typical workflow runtimes of hours or even days.

workflow. In current use cases the number of extracted metadata entries tends to be in the low double-digit number range. The following parameters are utilized by the generic parser for these extraction and annotation steps;

- The workflow's MSML document,
- the identifier of the current job for which the MSML "finalization" section will be created,
- the reference in the MSML document to the parser configuration which is included in the generic parser itself,
- the regular expression that extracts relevant information from the output and which is stored in the parser configuration, and
- the dictionary corresponding to the current application that is listed in the parser configuration.

The following paragraph will focus on extracting information from NWChem specific output. The described steps are independent of specific applications and can be applied for further use cases as well.

Exactly one NWChem output file (nwchem.out) is processed during each measurement. To extract sample information from the output file, a rule in the form of a regular expression, was developed. The information is then inserted into the corresponding MSML template of the workflow. During these measurements, the regular expression rule was utilized for a varying number of executions of the generic parser. One measurement set contains executions ranging from 1k to 10k in 1k steps in between (see Figure 5.1). Another measurement ranges in 10k steps from 10k to 100k executions (see Figure 5.2). To enable the measurements, nineteen parser configurations were build. Each contains duplicated regular expression rules. Their number matches the times of executions (1k to 100k). Each extracted metadata is immediately inserted into the MSML document within the finalization segment of the corresponding job in an individual property tag. A separate MSML document was created as input for each measurement

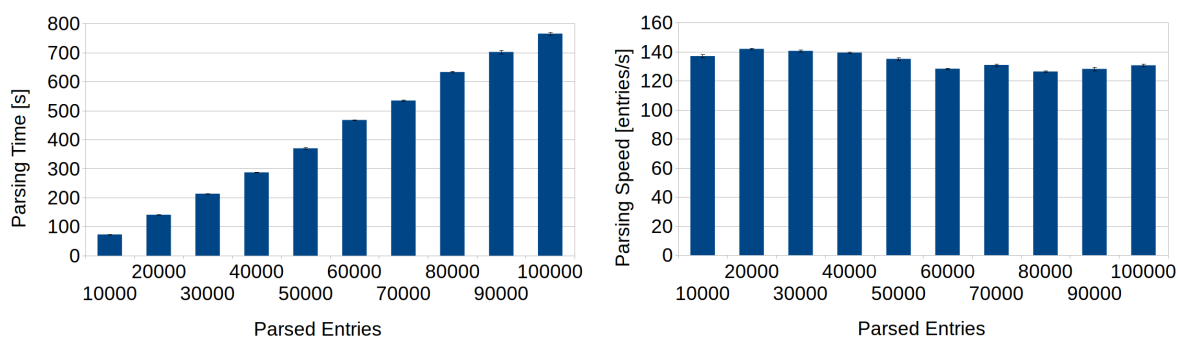


Figure 5.2: A performance evaluation of the extraction and annotation step from the central MoSGrid information hub MSML to the JSON metadata format is presented. Depicted is the parsing time in seconds on the left side and the corresponding parsing speed in processed entries per second on the right side. 10 datasets with 10000 to 100000 entries in steps of 10000 are shown. Each measurement was repeated 10 times and the respective average and error bars are shown. Even at this extreme benchmarking scale, that will not be reached in production use cases in the foreseeable future, the imposed overhead by the metadata extraction and annotation is insignificant in contrast to typical workflow runtimes of hours or even days.

with an individual number of parser executions. The document contains a reference to the respective parser configuration. Additionally, individual property tags, storing the extracted values, were defined in nineteen new dictionaries. For example, 10k property value tags were created in the corresponding dictionary for the 10k run. During each of the nineteen measurements (1k to 10k and 10k to 100k) the walltime was recorded. Each measurement was repeated ten times and the calculated average utilized to create the plots in the Figures 5.1 and 5.2. An example of how to execute a parser with an individual configuration is: `"java -jar genparser.jar -h -f msml-document.xml -o extended-msml-document.xml -j Job-ID"`. The HPC node that the measurements was executed on has 16 cores (2x Intel E5-2670@2,60GHz) and 64 GB of main memory.

To sum up, in measurements between 1k and 100k repetitions, metadata is extracted from a NWChem specific output file via regular expression rules. Subsequently, the metadata is inserted into the corresponding MSML document. This step also includes the extraction from the metadata in MSML to JSON format in order to annotate it to the data. The utilized numbers of regular expression rules are fitting in order to evaluate the performance of extracting metadata from output files. The upper bound number of 100k was chosen to represent a level of complexity that is unattained in current use cases. Even when a large list of atoms in an application output file is processed and inserted in a MSML document, the upper bound value of 100k is by far high enough. All realistic use cases can be considered to be covered for the foreseeable future.

The measurement results are shown as plots in the Figures 5.1 and 5.2. In the Figure 5.1 the measurements from 1k to 10k in steps of 1k are shown. The left plot displays the overall time a measurement took which ranges from 9 seconds to 73 seconds, representing a linear rise. In the right plot of Figure 5.1, the rate with which entries are processed is plotted in entries per second. The number ranges from 105 entries per second to 137 entries per second. In both measurements the standard deviation is very small. The plots in Figure 5.2 show the measurements between 10k and 100k in 10k steps. In the left plot, the

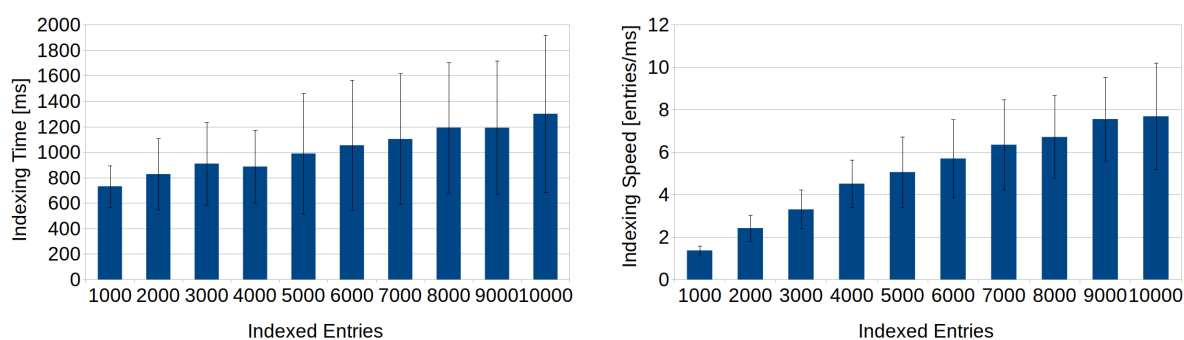


Figure 5.3: Here, the performance of the central step that indexes the metadata in JSON format in order to expose it to search queries by users is evaluated. Depicted is the indexing time in seconds on the left side and the corresponding indexing speed in processed entries per second on the right side. 10 datasets with 1k to 10k entries in steps of 1k are shown. Each measurement was repeated 10 times and the respective average and error bars are shown. The overhead of about 1 second for 10k indexed entries, a large number that will probably not be reached in production use cases in the foreseeable future, is completely insignificant.

overall time linearly rises from 73 seconds to 765 seconds. In the right plot, the measurements show a processing rate ranging from 126 entries per second to 142 entries per second. At 40k the rate is slightly decreasing due to the increasing size of dictionaries and parsing rules that need to be evaluated and executed. By starting the Java virtual machine for executing the generic parser, a base overhead was introduced. The maximum additional time that a workflow execution takes is 760 seconds by executing 100k rules. This increase of the workflow runtime is by orders of magnitude smaller than the typical workflow executions times of hours or even days. Thus, even in such a benchmarking scenario with a huge number of metadata entries, the added overhead is insignificant compared to the value added by metadata management.

5.2.3 Metadata Indexing

The indexing of metadata, that was extracted from a MSML document, is another crucial step in the extraction-annotation-indexing component chain. It is performed via the UNICORE metadata service that integrates Apache Lucene as the indexing engine. This step exposes the extracted and annotated metadata in JSON format to search requests by users. In a concrete data life cycle such as MoSGrid, this indexing step is automatically triggered at the end of each workflow. The following evaluation of the performance aims at finding potential scalability limits to ensure the implementation is applicable for future use cases. Current metadata entry number is in the low double-digit range. This investigation is included in the recently accepted manuscript mentioned above [?].

For one set of measurements, 19 JSON metadata files in JSON format were created that contain 1k to 10k entries in 1k steps (Figure 5.3) and another measurement set ranges from 10k to 100k entries in 10k steps (Figure 5.4). To ensure constant measurement conditions, both the Lucene metadata index and directory containing the JSON file is purged before each measurement. Then, the JSON metadata file is transferred to a directory managed by UNICORE. Then, to start the indexing process, the "start-extract"

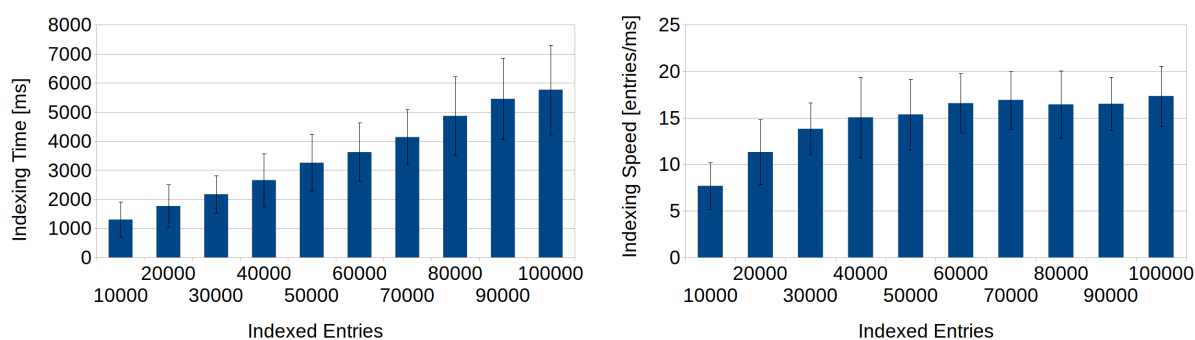


Figure 5.4: Here, the performance of the central step that indexes the metadata in JSON format in order to expose it to search queries by users is evaluated. Depicted is the indexing time in seconds on the left side and the corresponding indexing speed in processed entries per second on the right side. 10 datasets with 10k to 100k entries in steps of 10k are shown. Each measurement was repeated 10 times and the respective average and error bars are shown. Even for the 100k entries, an extreme number that will not be reached in production use cases in the foreseeable future, the overall overhead is only about 6 seconds and, thus, insignificant in contrast to common workflow runtimes in the range of hours to days.

command is triggered on this directory. Specifying just this directory ensures that exactly one JSON file is indexed while the walltime is measured. After the indexing is completed, the UNICORE metadata service writes the indexing time to its log file (uas.log). Ten repetitions were performed for each measurement and the average for used to create the Figures 5.3 and 5.4. A virtual machine with four cores (AMD 6272@2,1GHz) and 8 GB of main memory was utilized for the measurements. Summarized, 1k to 100k is the number of metadata entries in a JSON file which was created and indexed. Corresponding to the extraction and annotation steps, 100k is by far a high enough limit that even with complete lists of atoms all use cases in the foreseeable future are covered.

The measurement results for 1k to 10k entries are shown in Figure 5.3. The averaged indexing times over ten repetitions basically rises linearly from 731 milliseconds to 1300 milliseconds (left plot). The indexing rate in the right plots shows how many JSON entries per millisecond are indexed. The rate ranges from 1.3 to 7.7 entries per millisecond. A significant base overhead can be seen in the left plot. Due to this, the rate is rising as the overhead plays a smaller role with growing numbers of entries. Figure 5.4 shows the measurement results for 10k to 100k with the indexing time on the left linearly rising from 1300 milliseconds to 5765 milliseconds. The indexing rate ranges from 7.7 entries per millisecond to 17.4 entries per millisecond and starts to flatten at about 30k entries because the imposed base overhead becomes increasingly less significant with a growing number of indexed entries. In these measurements the standard deviation is noticeable because the results are in the single-digit second order for the time measurements and in the single-digit to lower double-digit entries/milliseconds order for the speed measurements. At these low scales concurrently running processes from other applications or from the operation system play a significant role. Even with a large metadata file of 100k entries the indexing step only an overhead of about 6 seconds on a workflow. In the face of workflow runtimes in the order of hours to days, the imposed overhead is completely negligible, even for such a scalability measurement scenario. Thus, it is well worth the added capabilities of metadata management.

5.3 Sustainability

Sustainability of utilized metadata and related technologies and infrastructures is of the utmost importance. It directly lowers the maintenance effort as such systems tend to be more mature and better supported through developers and an active user community.

An important point is that software is published under a free and open source license. In this case, risks are significantly lowered. For example, the bankruptcy of a supporting company would only have limited negative effects. In contrast, proprietary software would have a high chance of being completely gone forever. This is also often the case when a competitor buys the company to get rid of a competing product. The recommended technologies UNICORE and all its underlying libraries, Apache Lucene and Tika, Liferay, and gUSE/WS-PGRADE are free and open source software.

The use of standards is an essential enabler for sustainability. Technologies become more interoperable by employing standards. Metadata components become easier interchangeable which further increases the long-term viability. Example standards that the concept employs are JSON, JSR168/JSR286, the XML-based MSML, WSRF, SAML, and more. Also, the more widely used a technology is, the more sustainable it is as more people care and depend on the technology. For example, UNICORE is used on some of the worlds largest supercomputers via the European PRACE and the US-wide XSEDE research infrastructures. It is also the core of the one billion Euro Human Brain Project. Apache Lucene and Tika are top level Apache projects, indicating both a decent maturity and prevalence. Furthermore, gUSE/WS-PGRADE is utilized as the basis for many science gateways in Europe [gUS15]. When these solutions become standard technologies, that are routinely deployed at data centers, the sustainability is significantly fostered.

Due to the central focus on research, developments of some involved technologies can be ensured for a longer term via an institute's core funding. This increases the long-term viability as in the cases of UNICORE and gUSE/WS-PGRADE. Additionally, the more wide spread and mature a system is, the more related third party funding through research projects it tends to attract. It includes the implementation within new use case and of extensions. An example here is the integration and extension of UNICORE within the EU Human Brain Project.

An important factor are active research collaborations. Synergy effects can be realized by saving development efforts and by exchanging ideas and, thus, getting to know novel ideas earlier. Outreach and dissemination through the utilization of web technologies, by giving presentations and publishing the results are contributing factors to the sustainability as well. It fosters by how many people and how well the research is known. This is actively done for the metadata concept within the MASi project (see Section 6.2) and for the MoSGrid implementation in the chemistry community and beyond. In the MASi research infrastructure project, the generic concept and its implementation are incorporated.

Successful community building is also essential for the sustainability. Users have to be well informed at a level that fits their expertise and expectations. This may be done via newsletters and community events. On such events tutorial sessions shall be given in order to create a greater familiarity with the technology. Community building is also important in a wider context through networking with developers and new potential users. Furthermore, a vital aspect is the user-friendliness. The higher it is, the more users will

use the system and the more sustainable it tends to be as a consequence.

5.4 Resilience

Resilience is highly important in ever more complex scientific data life cycles. Various distributed systems are involved and depend on each other. Thus, inevitable errors immediately have a negative impact. Though beside the focus of this dissertation, resilience has been kept in mind during the design of the concept and its MoSGrid implementation.

The generic concept and its implementation are resilient to an advanced degree due to several characteristics. The specific metadata components within the MoSGrid data life cycle are depicted in Figure 5.5. The rows show components that are relevant before (top row), during (middle row), and after (bottom row) the runtime of a workflow. The left column represents components running on distributed resources within a workflow with the files being stored in distributed data management systems. The middle column shows components running via central services but several of these services can be used. The right column depicts components that are running on a central component once for every data life cycle.

A main characteristic is the aspect of combined metadata proximity (see Section 2.2.3). The metadata is kept near the data (see left cell in Figure 5.5) and a central component (see components in the middle and lower middle cells in Figure 5.5) is only used for building an index for searchability over the metadata (see lower right cell in Figure 5.5). This characteristic provides the advantage that the metadata can be handled in the same way as the data. For example, the metadata can be archived in the same way as the data. Archiving would be a big problem with a purely central metadata components. A downside of the combined proximity approach is that the file count is significantly increased when the metadata is stored in one extra file for each original file. Per default, this is the case within the UNICORE metadata service. A significant mitigation would be to extend the metadata service to only keep one metadata file for each directory. This file could include the metadata of all files in the directory.

When the loosely coupled central search index (middle and lower middle cells in Figure 5.5) fails and is possibly lost, the metadata is still save besides the actual files. The index can simply be rebuilt. A completely centralized systems would loose the metadata. The data would basically become useless until the database backup is restored, if a backup exists and if it is sufficiently sophisticated to have survived the cause of the original failure. The dissertation approach avoids this negative possibility.

The extraction and indexing components are loosely coupled as well. Meaning, if the metadata service temporarily fails, the metadata can still be extracted and annotated on the workflow level on the High Performance Computing resources (middle left cell in Figure 5.5). The metadata can later effortlessly be indexed when the service is operational again. Running the extraction and annotation on a central component would be more fragile with respect to error situations. This is avoided in the dissertation approach.

Additionally, more than one metadata service can be employed for indexing and exposure of the index for searchability. If one index fails, others can be transparently used. This redundancy comes at the cost of a higher complexity and overhead. But the indexing overhead is completely insignificant, as was

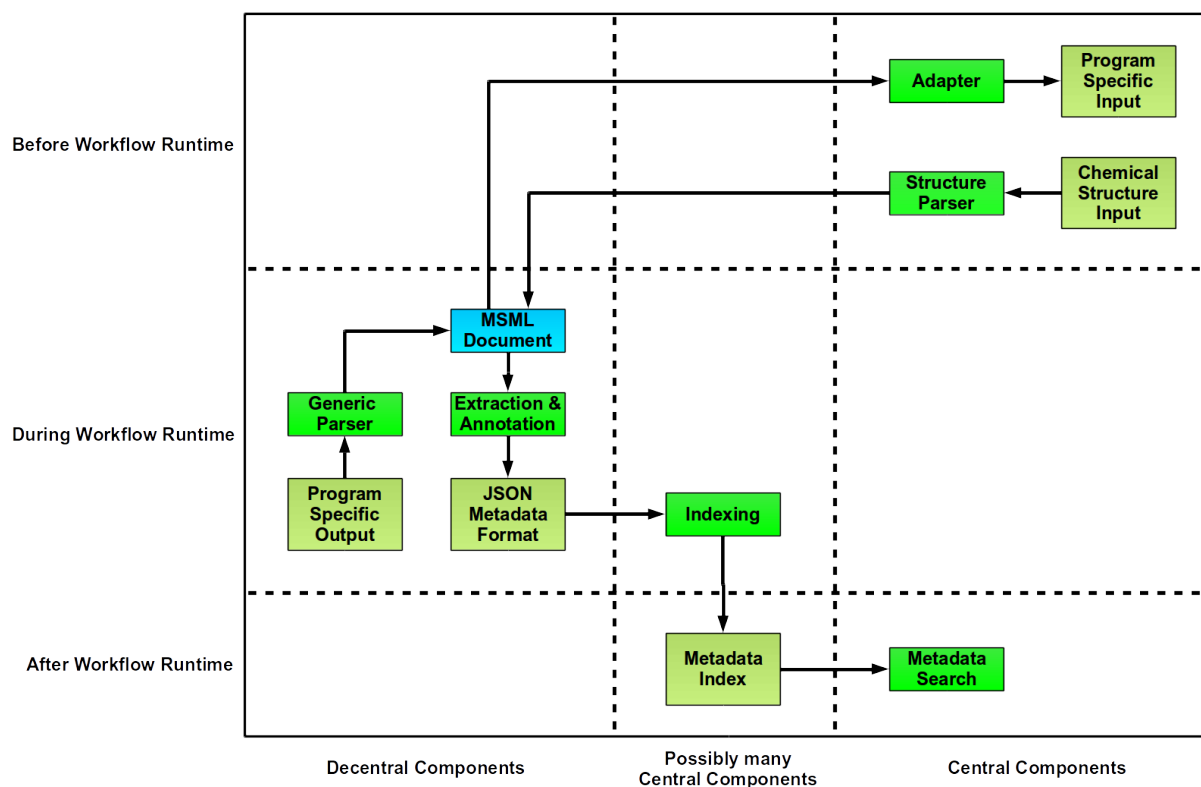


Figure 5.5: Different resilience classes are depicted. From top to bottom, the rows represent before, during and after a workflow is running. From left to right, the columns represent components that run decentralized, central but possibly many and completely centralized. The figure shows that during workflow runtime (middle left) within the MoSGrid data life cycle only at the end the central indexing component (middle) is used, but which may be duplicated for greater resilience. The created index (lower middle) can exist multiple times and is accessed via the central search interface of MoSGrid (lower right). Favorable resilience characteristics exist due to the principle of distributed when possible and central where necessary.

shown in Section 5.2.3. The added complexity is invisible and, thus, goes unnoticed by the user. At the end of the workflow, all connected metadata services are triggered to search for new metadata to be indexed (middle cell in Figure 5.5) in the specific directory of the user.

5.5 Efficiency of Use

In this chapter the question is investigated how the generic metadata concept and its implementation facilitate usage efficiency. It is an important point as usability directly leads to a high acceptance among the target audience of applying scientists.

The central design criterion of the concept and subsequently its implementation was a seamless integration with complex infrastructures. This was fully achieved. Users notice the added metadata capabilities while the required added complexity, on the other hand, is completely hidden beneath the surface of the user interfaces. This seamlessness greatly fosters usability as even for complete novice users an increase of the hurdle of use is avoided while greatly advancing data management capabilities.

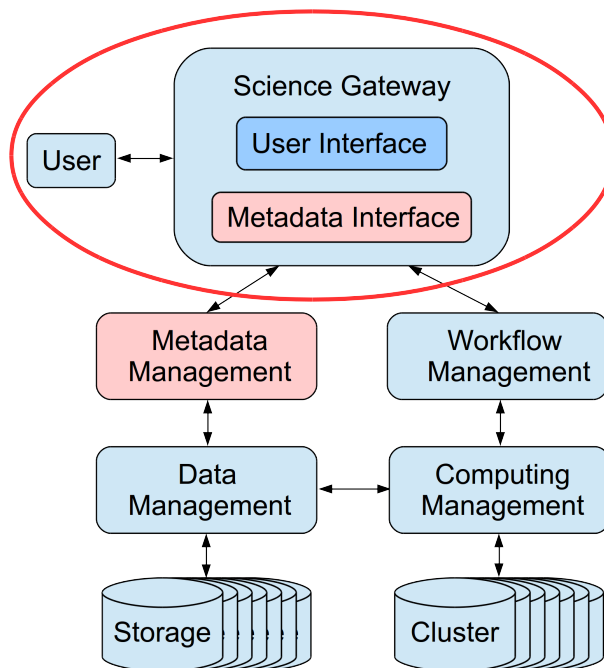


Figure 5.6: The goals are to fully abstract from and seamlessly integrate with the underlying highly complex MoSGrid data life cycle. These goals were achieved in an automated and seamless way. Access to potentially millions of files spread of numerous file server is enabled based on their content. Thus, a high degree of usability is provided.

Instead of manually having to remember what content each file has and where it is located, the organization is enabled based on a file content. Otherwise, millions of files spread over numerous file server would be impossible to manage. The search interface allows to specify information about the content. The files matching these criteria are returned as results and can seamlessly be utilized as input for further workflows. As the metadata was defined in close cooperation with the end users, it is ensured the metadata search results are immediately useful for the scientists.

An essential part enabling the high usability is full automation. Every underlying metadata feature is automated. This includes metadata extraction, annotation, and indexing. Other systems, such as EUDAT, still require the manual annotation of metadata. For users, manual steps are highly annoying and tedious and constitute a reason to avoid the system. In the metadata concept and implementation at hand, full automation is incorporated.

Via the abstraction from specific technologies the usability is further increased. When an underlying metadata system is exchanged by another, this goes unnoticed from the point of users and how they interact with the system. Through the overall ease-of-use of the metadata capabilities (see Figure 5.6), scientists are enabled to focus on their science while making transparent use of complex High Performance Computing and Big Data infrastructures that fundamentally enable their research.

6 Conclusion and Outlook

This chapter concludes the thesis and looks ahead.

6.1 Conclusion

The thesis presents an overarching and generic metadata handling concept for scientific data. Utilizing metadata, the approach facilitates the move towards the next generation of data management within scientific data life cycles. Metadata management in general enables the organization of large amounts of data with file number in the millions. Instead of only being accessible via names in directory structures, files can be accessed by their content. Despite the inherent complexity of data contents, the data can be seamlessly accessed via simple search queries and directly further utilized. The high complexity and large magnitude of scientific data life cycles is motivated. This subsequently necessitates sophisticated and integrated technologies for their management [GKG⁺15]. Based on such technologies, scientists are enabled to advance their respective state-of-the-art with the combined support of High Performance Computing and Big Data resources. A multitude of open challenges in this broad data life cycle context was identified. Within the challenges, a major one is that of completely missing or only narrowly applicable metadata management approaches.

The metadata concept [GGJN14, GBG⁺14] first specifies multiple advantageous characteristics. The abstraction of various technologies is heavily utilized to handle the high data life cycle complexity. Data and metadata formats are integrated in a generic way to enable advanced search capabilities for the efficient usage by scientists. The concept is inherently scalable within HPC-enabled and Big Data life cycles. Full automation is provided for extraction, annotation, and indexing of metadata. Second, the concept provides a design guide that facilitates an understanding of overall data life cycle design aspects [GDP⁺15] as well as aspects specific to metadata management. The guide includes recommendations of proven technologies. The overall concept is generic in the sense that it enables metadata management to be more quickly and efficiently integrated in concrete data life cycles. The direct usage of metadata search results is enabled while underlying Big Data and High Performance Computing resources are seamlessly integrated. Users gain from new capabilities while the added necessary complexity is hidden.

The concept was implemented within the MoSGrid data life cycle [KGG⁺14, GBB⁺12, GKG⁺14]. MoSGrid enables highly complex molecular simulations within the three major computational chemistry domains. The MoSGrid implementation is HPC and workflow enabled, offers advanced data management capabilities with a sophisticated single sign-on architecture throughout all layers [GGK⁺12]. The implementation based on the generic approach was seamlessly integrated with this complex data life cycle. The metadata extraction, annotation, and indexing are performed in a fully automatic way [GBG⁺14].

A search interface enables finding data based on its content. Furthermore, search results can immediately be re-used as input within further workflows.

A thorough evaluation of the concept and its implementation was performed [GKJ⁺ed, GBG⁺14] with respect to adaptability, performance, sustainability, resilience, and efficiency of use. The existence of favorable properties was shown.

On a theoretical level, data life cycle management is advanced by facilitating higher level abstraction with metadata management. A practical impact is achieved by the implementation within the MoSGrid data life cycle and the uptake of the concept within the MASi research infrastructure.

6.2 Outlook

The dissertation contributes to the state-of-the-art in facilitating the handling of large amounts of complex data via metadata. Further advancements include the following.

Based on the contributions and the doctoral research collaborations, the author initiated and coordinated the proposal of the MASi (Metadata Management for Applied Sciences) research infrastructure project submitted to the German Research Foundation (DFG). MASi is now funded and led by the author. It is creating a distributed, generic, and sustainable service for the integrated management of large scale scientific data based on metadata. The service will be loosely coupled between data centers where the components are sustainably run locally for the respective communities. MASi aims at long-term sustainability via its involvement in the Research Data Alliance (RDA), the Large Scale Data Management and Analysis (LSDMA) project, and the Competence Center for Scalable Data Services and Solutions (ScaDS Dresden/Leipzig). MASi aims at supporting various advanced concepts as detailed in the following.

Multiple data and metadata sources are being integrated with MASi, so users can stage in data from external data sources and use it in a seamless way. Support for workflow provenance shall be offered as it is increasingly important to foster scientific reproducibility. The same holds true for persistent identifiers (PIDs) so data can be uniquely referenced worldwide. Different underlying metadata management systems will be transparently supported by MASi. This will be enabled by the CDMI (Cloud Data Management Interface) and OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) access standards as well as with support for the iRODS and ICAT systems. For example, the widely used dCache distributed management systems plans to natively support metadata management and data access via CDMI.

MASi establishes metadata as the central source of information within a data life cycle. This needs to be established as a general design principle in data life cycles in general. With increasing data amounts, the computing power required for extraction is increasing as well. Metadata extraction methods have to be integrated in a scalable way as close to the data as possible. The dissertation approach includes this and it is exemplary implemented for the MoSGrid extraction methods. General extraction frameworks such as Apache Tika need to support scalable execution as well. In addition, pre-existing metadata, from whatever source available, needs to be seamlessly utilized. Future research also necessitates the

integration of authentication, authorization, and single sign-on methods via nation-level federations such as DFN-AAI or on the European level via EduGain. This is essential to further lower barriers of entry by enabling the re-use of already existing login credentials.

As a central part of advanced data life cycles, metadata management should be resilient and fault tolerant to a high degree. Usually many distributed components depend on each other. Single points of failure have to be avoided by, for example, utilizing a combined metadata proximity approach as in the dissertation approach. On the one hand, it is decentralized as much as possible with loose coupling. On the other hand, the approach is centralized as much as necessary. Such an emphasis on resiliency enables a data life cycle to tolerate certain kinds of errors. When errors occur, users should get error messages in an understandable form while administrators should get all the details. In MASi, fault tolerance will additionally be supported by replicating metadata across participating data centers.

A further research and development direction is to encapsulate even more functionality within scientific workflows. An example is to define a metadata search term as a workflow parameter. Then, during workflow enactment, the search query is executed and all files referenced via search results are utilized as input. A vital factor to enable such an approach is the interoperability and integration of High Performance Computing and Big Data systems with metadata management.

The Max Planck Institute of Molecular Cell Biology and Genetics in Dresden develops, builds, and utilizes high-throughput microscopes with common data rates of 0.85 GB/s and 10 files/s. These vast amounts of data alone are a significant challenge. An adaption estimation of the dissertation metadata approach and surrounding technologies for this use case is presented in Section 5.1.2. The author currently cooperates with the institute towards its implementation.

Exascale is on the horizon in both the computing and data domain. A multitude of challenges arises on various abstraction levels. These challenges necessitate advanced approaches in data life cycle management with metadata management as a vital component [JMPK⁺15]. The dissertation approach is a step in paving the way for the efficient management of increasingly large quantities of data towards exascale.

A Publication List

During his time as doctorand, the author contributed to the wider field of data life cycle management. The resulting publications are listed in the following and many are referenced throughout the dissertation. Publications closely connected to the doctoral research are also listed in Section 1.4 and referenced in Section 1.3.

* These authors contributed equally to the respective work.

41. Krzysztof Benedyczak, Bernd Schuller, Maria Petrova, Jędrzej Rybicki and **Richard Grunzke**. UNICORE 7 - Middleware Services for Distributed and Federated Computing. Future Generation Computer Systems, Elsevier, submitted.
40. **Richard Grunzke**, Jens Krüger, René Jäkel, Wolfgang E. Nagel, Sonja Herres-Pawlis and Alexander Hoffmann. Metadata Management in the MoSGrid Science Gateway for Quantum Chemistry. Journal of Grid Computing, accepted.
39. Alvaro Aguilera, **Richard Grunzke**, Ulf Markwardt, Dirk Habich, Dirk Schollbach and Jochen Garcke. Towards an Industry Data Gateway: An Integrated Platform for the Analysis of Wind Turbine Data. Science Gateways (IWSG), 2015 7th International Workshop on, 2015, 62-66.
38. Sandra Gesing, Rion Dooley, Marlon Pierce, Jens Krüger, **Richard Grunzke**, Sonja Herres-Pawlis and Alexander Hoffmann. Science Gateways - Leveraging Modeling and Simulations in HPC Infrastructures via Increased Usability. High Performance Computing Simulation (HPCS), 2015 International Conference on, 2015, 19-26.
37. Sandra Gesing, Jens Krüger, **Richard Grunzke**, Sonja Herres-Pawlis and Alexander Hoffmann. Challenges and Modifications for Creating a MoSGrid Science Gateway for US and European Infrastructures. Science Gateways (IWSG), 2015 7th International Workshop on, 2015, 73-79.
36. **Richard Grunzke**, Jens Krüger, Sandra Gesing, Sonja Herres-Pawlis, Alexander Hoffmann, Alvaro Aguilera and Wolfgang E. Nagel. Managing Complexity in Distributed Data Life Cycles Enhancing Scientific Discovery. e-Science (e-Science), 2015 IEEE 11th International Conference on, 2015, 371-380.
35. Sonja Herres-Pawlis, Alexander Hoffmann, Ákos Balaskó, Peter Kacsuk, Georg Birkenheuer, André Brinkmann, Luis de la Garza, Jens Krüger, Sandra Gesing, **Richard Grunzke**, Gabor Terstyansky and Noam Weingarten. Quantum Chemical Meta-Workflows in MoSGrid. Concurrency and Computation: Practice and Experience, 2015, 27, 344-357.
34. Sonja Herres-Pawlis, Alexander Hoffmann, Thomas Rosener, Jens Krüger, **Richard Grunzke** and Sandra Gesing. Multi-layer Meta-metaworkflows for the Evaluation of Solvent and Dispersion

- Effects in Transition Metal Systems Using the MoSGrid Science Gateways. Science Gateways (IWSG), 2015 7th International Workshop on, 2015, 47-52.
33. René Jäkel, Ralph Müller-Pfefferkorn, Michael Kluge, **Richard Grunzke** and Wolfgang E. Nagel. Architectural Implications for Exascale based on Big Data Workflow Requirements. Big Data and High Performance Computing, IOS Press, 2015, 26, 101 - 113.
 32. Gary A. McGilvary, Malcolm Atkinson, Sandra Gesing, Alvaro Aguilera, **Richard Grunzke** and Eva Sciacca. Enhanced Usability of Managing Workflows in an Industrial Data Gateway. Interoperable Infrastructures for Interdisciplinary Big Data Sciences (IT4RIs 15), 2015, 495-502.
 31. Sandra Gesing, Jens Krüger, **Richard Grunzke**, Luis de la Garza, Sonja Herres-Pawlis and Alexander Hoffmann. Molecular Simulation Grid (MosGrid): A Science Gateway Tailored to the Molecular Simulation Community. Science Gateways for Distributed Computing Infrastructures, Springer International Publishing, 2014, 151-165.
 30. **Richard Grunzke**, Sebastian Breuers, Sandra Gesing, Sonja Herres-Pawlis, Martin Kruse, Dirk Blunk, Luis de la Garza, Lars Packschies, Patrick Schäfer, Charlotta Schärfe, Tobias Schlemmer, Thomas Steinke, Bernd Schuller, Ralph Müller-Pfefferkorn, René Jäkel, Wolfgang E. Nagel, Malcolm Atkinson and Jens Krüger. Standards-based Metadata Management for Molecular Simulations. Concurrency and Computation: Practice and Experience, 2014, 26(10), 1744–1759.
 29. **Richard Grunzke**, Sandra Gesing, René Jäkel and Wolfgang E. Nagel. Towards Generic Metadata Management in Distributed Science Gateway Infrastructures. IEEE/ACM CCGrid 2014 (14th International Symposium on Cluster, Cloud and Grid Computing), 2014, 566-570.
 28. **Richard Grunzke**, Jürgen Hesser, Jürgen Starek, Nick Kepper, Sandra Gesing, Marcus Hardt, Volker Hartmann, Stephan. Kindermann, Jan Potthoff, Michael Hausmann, Ralph Müller-Pfefferkorn and René Jäkel. Device-driven Metadata Management Solutions for Scientific Big Data Use Cases. 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2014), 2014.
 27. **Richard Grunzke**, Jens Krüger, Sandra Gesing, Sonja Herres-Pawlis, Alexander Hoffmann and Luis de la Garza. Improved Resilience and Usability for Science Gateway Infrastructures via Integrated Virtual Organizations. EGI Community Forum 2014, 2014.
 26. **Richard Grunzke** and Ralph Müller-Pfefferkorn. Certificate-free User-friendly HPC Access with UNICORE. UNICORE Summit 2014 Proceedings, 2014, 26, 23-30.
 25. Sonja Herres-Pawlis, Alexander Hoffmann, Luis De La Garza, Jens Krüger, Sandra Gesing, **Richard Grunzke**, Wolfgang E. Nagel, Gabor Terstyansky and Noam Weingarten. Meta-Metaworkflows for Combining Quantum Chemistry and Molecular Dynamics in the MoSGrid Science Gateway. Science Gateways (IWSG), 2014 6th International Workshop on, 2014, 73-78.
 24. Sonja Herres-Pawlis, Alexander Hoffmann, Luis De La Garza, Jens Krüger and **Richard Grunzke**. Expansion of Quantum Chemical Metadata for Workflows in the MoSGrid Science Gateway. Science Gateways (IWSG), 2014 6th International Workshop on, 2014, 67-72.

-
23. Alexander Hoffmann, **Richard Grunzke** and Sonja Herres-Pawlis. Insights into the Influence of Dispersion Correction in the Theoretical Treatment of Guanidine-Quinoline Copper(I) Complexes. *Journal of Computational Chemistry*, 2014, 35, 1943-1950.
 22. Jens Krüger*, **Richard Grunzke***, Sandra Gesing*, Sebastian Breuers, André Brinkmann, Luis de la Garza, Oliver Kohlbacher, Martin Kruse, Wolfgang E. Nagel, Lars Packschies, Ralph Müller-Pfefferkorn, Patrick Schäfer, Charlotta Schärfe, Thomas Steinke, Tobias Schlemmer, Klaus Dieter Warzecha, Andreas Zink and Sonja Herres-Pawlis. The MoSGrid Science Gateway - A Complete Solution for Molecular Simulations. *Journal of Chemical Theory and Computation*, 2014, 10(6), 2232-2245.
 21. Jens Krüger, **Richard Grunzke**, Sonja Herres-Pawlis, Alexander Hoffmann, Luis de la Garza, Oliver Kohlbacher, Wolfgang E. Nagel and Sandra Gesing. Performance Studies on Distributed Virtual Screening. *BioMed Research International*, 2014.
 20. Sonja Herres-Pawlis, Alexander Hoffmann, Luis De La Garza, Jens Krüger and **Richard Grunzke**. Expansion of Quantum Chemical Metadata for Workflows in the MoSGrid Science Gateway. *Science Gateways (IWSG)*, 2014 6th International Workshop on, 2014, 67-72.
 19. Sonja Herres-Pawlis, Alexander Hoffmann, **Richard Grunzke** and Lars Packschies. Orbital Analysis of Oxo and Peroxo Dicopper Complexes via Quantum Chemical Workflows in MoSGrid. *Proceedings of the International Workshop on Scientific Gateways 2013 (IWSG)*, 2013.
 18. Alexander Hoffmann, Sonja Herres-Pawlis, Sandra Gesing, Luis de la Garza, Jens Krüger and **Richard Grunzke**. User-friendly Metaworkflows in Quantum Chemistry. *Cluster Computing (CLUSTER)*, 2013 IEEE International Conference on, 2013, 1-3.
 17. Lars Packschies, Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Luis de la Garza, Ines Dos Santos Vieira, Gregor Fels, Sandra Gesing, **Richard Grunzke**, Sonja Herres-Pawlis, Oliver Kohlbacher, Jens Krüger, Martin Kruse, Ulrich Lang, Ralph Müller-Pfefferkorn, Patrick Schäfer, Tobias Schlemmer, Charlotta Schärfe, Thomas Steinke, Klaus-Dieter Warzecha and Andreas Zink. The MoSGrid e-Science Gateway: Molecular Simulations in a Distributed Computing Environment. *Journal of Cheminformatics*, 2013, 5(Suppl 1):O3.
 16. Bernd Schuller, **Richard Grunzke** and Andre Giesler. Data Oriented Processing in UNICORE. *UNICORE Summit 2013 Proceedings*, 2013, 21, 1-6.
 15. Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Ines Dos Santos Vieira, Gregor Fels, Sandra Gesing, **Richard Grunzke**, Sonja Herres-Pawlis, Oliver Kohlbacher, Jens Krüger, Ulrich Lang, Lars Packschies, Ralph Müller-Pfefferkorn, Patrick Schäfer, Thomas Steinke, Klaus-Dieter Warzecha and Martin Wewior. MoSGrid: Efficient Data Management and a Standardized Data Exchange Format for Molecular Simulations in a Grid Environment. *Journal of Cheminformatics*, 2012, 4(Suppl 1):P21.
 14. Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Gregor Fels, Sandra Gesing, **Richard Grunzke**, Sonja Herres-Pawlis, Oliver Kohlbacher, Jens Krüger, Ulrich Lang, Lars Packschies, Ralph Müller-Pfefferkorn, Patrick Schäfer, Johannes Schuster, Thomas Steinke, Klaus-

- Dieter Warzecha and Martin Wewior. MoSGrid: Progress of Workflow driven Chemical Simulations. Proceedings of the Grid Workflow Workshop (GWW), CEUR Workshop Proceedings, Cologne, Germany, 2012, 826.
13. Sandra Gesing*, **Richard Grunzke***, Jens Krüger, Georg Birkenheuer, Martin Wewior, Patrick Schäfer, Bernd Schuller, Johannes Schuster, Sonja Herres-Pawlis, Sebastian Breuers, Ákos Balaskó, Miklos Kozlovsky, Anna Szikszay Fabri, Lars Packschies, Peter Kacsuk, Dirk Blunk, Thomas Steinke, Andre Brinkmann, Gregor Fels, Ralph Müller-Pfefferkorn, René Jäkel and Oliver Kohlbacher. A Single Sign-On Infrastructure for Science Gateways on a Use Case for Structural Bioinformatics. *Journal of Grid Computing*, 2012, 10, 769-790.
 12. Sandra Gesing, Sonja Herres-Pawlis, Georg Birkenheuer, Andre Brinkmann, **Richard Grunzke**, Peter Kacsuk, Oliver Kohlbacher, Miklos Kozlovsky, Jens Krüger, Ralph Müller-Pfefferkorn, Patrick Schäfer and Thomas Steinke. The MoSGrid Community – From National to International Scale. *EGI Community Forum 2012*, 2012.
 11. Sandra Gesing, Sonja Herres-Pawlis, Georg Birkenheuer, Andre Brinkmann, **Richard Grunzke**, Peter Kacsuk, Oliver Kohlbacher, Miklos Kozlovsky, Jens Krüger, Ralph Müller-Pfefferkorn, Patrick Schäfer and Thomas Steinke. A Science Gateway Getting Ready for Serving the International Molecular Simulation Community. *Proceedings of Science*, 2012, PoS(EGICF12-EMITC2)050.
 10. **Richard Grunzke**, Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Sandra Gesing, Sonja Herres-Pawlis, Oliver Kohlbacher, Jens Krüger, Martin Kruse, Ralph Müller-Pfefferkorn, Patrick Schäfer, Bernd Schuller, Thomas Steinke and Andreas Zink. A Data Driven Science Gateway for Computational Workflows. *UNICORE Summit 2012 Proceedings*, 2012, 15, 35-49.
 9. Sonja Herres-Pawlis, Georg Birkenheuer, Andre Brinkmann, Sandra Gesing, **Richard Grunzke**, René Jäkel, Oliver Kohlbacher, Jens Krüger and Ines Dos Santos Vieira. Workflow-enhanced Conformational Analysis of Guanidine Zinc Complexes via a Science Gateway. *Studies in Health Technology and Informatics*, 175:142-151, IOS Press, 2012.
 8. Tobias Schlemmer*, **Richard Grunzke***, Sandra Gesing*, Jens Krüger, Georg Birkenheuer, Ralph Müller-Pfefferkorn and Oliver Kohlbacher. Generic User Management for Science Gateways via Virtual Organizations. *EGI Technical Forum 2012*, 2012.
 7. Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Ines Dos Santos Vieira, Gregor Fels, Sandra Gesing, **Richard Grunzke**, Sonja Herres-Pawlis, Oliver Kohlbacher, Jens Krüger, Ulrich Lang, Lars Packschies, Ralph Müller-Pfefferkorn, Patrick Schäfer, Hans-Günther Schmalz, Thomas Steinke, Klaus-Dieter Warzecha and Martin Wewior. A Molecular Simulation Grid as new tool for Computational Chemistry, Biology and Material Science. *Journal of Cheminformatics*, 2011, 3(Suppl 1):P14.
 6. Sandra Gesing*, **Richard Grunzke***, Ákos Balaskó, Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Gregor Fels, Sonja Herres-Pawlis, Peter Kacsuk, Miklos Kozlovsky, Jens Krüger, Lars Packschies, Patrick Schäfer, Bernd Schuller, Johannes Schuster, Thomas Steinke,

-
- Anna Szikszay Fabri, Martin Wewior, Ralph Müller-Pfefferkorn and Oliver Kohlbacher. Granular Security for a Science Gateway in Structural Bioinformatics. Proceedings of 3rd Workshop IWSG-Life 2011, Proceedings of the 3rd International Workshop on Science Gateways for Life Sciences, 2011.
5. Sandra Gesing, Peter Kacsuk, Miklos Kozlovsky, Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Gregor Fels, **Richard Grunzke**, Sonja Herres-Pawlis, Jens Krüger, Lars Packschies, Ralph Müller-Pfefferkorn, Patrick Schäfer, Thomas Steinke, Anna Szikszay Fabri, Klaus-Dieter Warzecha, Martin Wewior and Oliver Kohlbacher. A Science Gateway for Molecular Simulations. EGI User Forum, Book of Abstracts, 2011, 94-95.
 4. **Richard Grunzke**, Ralph Müller-Pfefferkorn, Ulf Markwardt and Matthias Müller. Advancing Cutting-Edge Biological Research with a High-Throughput UNICORE Workflow. UNICORE Summit 2011 Proceedings, 2011, 09, 35-43.
 3. Jens Krüger, Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Gregor Fels, Sandra Gesing, **Richard Grunzke**, Oliver Kohlbacher, N. Kruber, Ulrich Lang, Lars Packschies, Ralph Müller-Pfefferkorn, Patrick Schäfer, Hans-Günther Schmalz, Thomas Steinke, Klaus-Dieter Warzecha and Martin Wewior. Molecular Simulation Grid. Journal of Cheminformatics, 2011, 3(Suppl 1):O17.
 2. Sandra Gesing, Istvan Márton, Georg Birkenheuer, Bernd Schuller, **Richard Grunzke**, Jens Krüger, Sebastian Breuers, Dirk Blunk, Gregor Fels, Lars Packschies, Andre Brinkmann, Oliver Kohlbacher and Miklos Kozlovsky. Workflow Interoperability in a Grid Portal for Molecular Simulations. Proceedings of the International Workshop on Scientific Gateways 2010 (IWSG), 2010, 44-48.
 1. **Richard Grunzke** and Bernd Schuller. Secure High-Throughput Computing Using UNICORE XML Spaces. UNICORE Summit 2010 Proceedings, 2010, 5, 27-35.

Bibliography

- [AAA⁺08] Georges Aad, E Abat, J Abdallah, AA Abdelalim, A Abdesselam, O Abdinov, BA Abi, M Abolins, H Abramowicz, E Acerbi, et al. The ATLAS Experiment at the CERN Large Hadron Collider. *Journal of Instrumentation*, 3(08):S08003, 2008.
- [AAA⁺09] Rasha Abbasi, Markus Ackermann, John Adams, Markus Ahlers, J Ahrens, K Andeen, Jan Auffenberg, Xinhua Bai, Michael Baker, SW Barwick, et al. The IceCube data acquisition system: Signal capture, digitization, and timestamping. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 601(3):294–316, 2009.
- [AAA⁺15] MG Aartsen, R Abbasi, M Ackermann, J Adams, JA Aguilar, M Ahlers, D Altmann, C Argüelles, J Auffenberg, X Bai, et al. The IceProd Framework: Distributed Data Processing for the IceCube Neutrino Observatory. *Journal of Parallel and Distributed Computing*, 75:198–211, 2015.
- [AAB⁺05] Mario Antonioletti, Malcolm Atkinson, Rob Baxter, Andrew Borley, Neil P Chue Hong, Brian Collins, Neil Hardman, Alastair C Hume, Alan Knox, Mike Jackson, et al. The Design and Implementation of Grid Database Services in OGSA-DAI. *Concurrency and Computation: Practice & Experience*, 17(2):357–376, 2005.
- [AAB⁺11] Mine Altunay, Paul Avery, Kent Blackburn, Brian Bockelman, Michael Ernst, Dan Fraser, Robert Quick, Robert Gardner, Sebastien Goasguen, Tanya Levshina, et al. A Science Driven Production Cyberinfrastructure - The Open Science Grid. *Journal of Grid Computing*, 9(2):201–218, 2011.
- [ABB⁺02] Bill Allcock, Joe Bester, John Bresnahan, Ann L Chervenak, Ian Foster, Carl Kesselman, Sam Meder, Veronika Nefedova, Darcy Quesnel, and Steven Tuecke. Data Management and Transfer in High-Performance Computational Grid Environments. *Parallel Computing*, 28(5):749–771, 2002.
- [ABC⁺11] Bryce Allen, John Bresnahan, Lisa Childers, Ian Foster, Gopi Kandaswamy, Raj Ketimuthu, Jack Kordas, Mike Link, Stuart Martin, Karl Pickett, et al. Globus Online: Radical Simplification of Data Movement via SAAS. *Preprint CI-PP-5-0611, Computation Institute, The University of Chicago*, 2011.
- [ABF⁺03] A Andronico, R Barbera, A Falzone, G Lo Re, A Pulvirenti, and A Rodolico. GENIUS: A Web Portal for the Grid. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 502(2):433–436, 2003.

- [ACC⁺04] Roberto Alfieri, Roberto Cecchini, Vincenzo Ciaschini, Luca dell’Agnello, Akos Frohner, Alberto Gianoli, Karoly Lorentey, and Fabio Spataro. VOMS, An Authorization System for Virtual Organizations. In *Grid Computing*, pages 33–40. Springer, 2004.
- [ACF⁺07] Jay Alameda, Marcus Christie, Geoffrey Fox, Joe Futrelle, Dennis Gannon, Mihael Hategan, Gopi Kandaswamy, Gregor von Laszewski, Mehmet A Nacar, Marlon Pierce, et al. The Open Grid Computing Environments Collaboration: Portlets and Services for Science Gateways. *Concurrency and Computation: Practice and Experience*, 19(6):921–942, 2007.
- [ADF⁺] A. Anjomshoaa, M. Drescher, D.I Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) Specification, Version 1.0. <http://www.gridforum.org/documents/GFD.56.pdf>.
- [AGM⁺15] Alvaro Aguilera, Richard Grunzke, Ulf Markwardt, Dirk Habich, Dirk Schollbach, and Jochen Garcke. Towards an Industry Data Gateway: An Integrated Platform for the Analysis of Wind Turbine Data. In *Science Gateways (IWSG), 2015 7th International Workshop on*, pages 62–66, June 2015. doi:10.1109/IWSG.2015.8.
- [AH03] Alejandro Abdelnur and Stefan Hepper. JSR 168: Portlet Specification. *Java Specification Requests, Java Community Process, Sun Microsystems and IBM*, 15, 2003.
- [ALG⁺12] Malcolm Atkinson, Chee Sun Liew, Michelle Galea, Paul Martin, Amrey Krause, Adrian Mouat, Oscar Corcho, and David Snelling. Data-intensive Architecture for Scientific Knowledge Discovery. *Distributed and Parallel Databases*, 30(5-6):307–324, 2012.
- [All02] Frank H Allen. The Cambridge Structural Database: A Quarter of a Million Crystal Structures and Rising. *Acta Crystallographica Section B: Structural Science*, 58(3):380–388, 2002.
- [AMC09] Stephen Abrams, Sheila Morrissey, and Tom Cramer. “What? So What”: The Next-Generation JHOVE2 Architecture for Format-Aware Characterization. *International Journal of Digital Curation*, 4(3):123–136, 2009.
- [B⁺04] Peter J Braam et al. The Lustre Storage Architecture, 2004.
- [BAD⁺11] R Barbera, G. Andronico, G. Donvito, A Falzone, J. J. Keijser, G. La Rocca, L. Milanese, G. P. Maggi, and S. Vicario. A Grid Portal with Robot Certificates for Bioinformatics Phylogenetic Analyses. *Concurrency and Computation: Practice and Experience*, 23(3):246—255, March 2011.
- [Bah12] Julio Lozano Bahilo. Production of Simulated Extensive Air Showers for the Pierre Auger Collaboration using Grid Technology. In *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pages 545–550. IEEE, 2012.
- [Bal09] Alexander Ball. Scientific Data Application Profile Scoping Study Report, 2009. URL: <http://www.ukoln.ac.uk/projects/sdapss/>.

- [Ban06] Tim Banks. Web Services Resource Framework (WSRF) - Primer v1.2. *OASIS Committee Draft*, 2006.
- [BB08] Robert Battle and Edward Benson. Bridging the Semantic Web and Web 2.0 with Representational State Transfer (REST). *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):61–69, 2008.
- [BBC⁺13] Michael Bussmann, Heiko Burau, Thomas E Cowan, Alexander Debus, Alex Huebl, Guido Juckeland, Thomas Kluge, Wolfgang E Nagel, Richard Pausch, Felix Schmitt, et al. Radiative Signatures of the Relativistic Kelvin-Helmholtz Instability. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 5. ACM, 2013.
- [BCD⁺08] Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. *KNIME: The Konstanz Information Miner*. Springer, 2008.
- [BCX⁺06] Kevin J Bowers, Edmond Chow, Huageng Xu, Ron O Dror, Michael P Eastwood, Brent A Gregersen, John L Klepeis, Istvan Kolossvary, Mark A Moraes, Federico D Sacerdoti, et al. Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pages 43–43. IEEE, 2006.
- [Ber08] Francine Berman. Got Data?: A Guide to Data Preservation in the Information Age. *Communications of the ACM*, 51(12):50–56, 2008. URL: <https://dl.acm.org/citation.cfm?id=1409376>, doi:10.1145/1409360.1409376.
- [BF07] Daniele Bonacorsi and Tiziana Ferrari. WLCG Service Challenges and Tiered Architecture in the LHC Era. In *IFAE 2006*, pages 365–368. Springer, 2007.
- [BHS09] Gordon Bell, Tony Hey, and Alex Szalay. Beyond the Data Deluge. *Science*, 323(5919):1297–1298, 2009.
- [Bir11] Ian Bird. Computing for the Large Hadron Collider. *Annual Review of Nuclear and Particle Science*, 61:99–118, 2011.
- [BIVB13] Matthias R Bauer, Tamer M Ibrahim, Simon M Vogel, and Frank M Boeckler. Evaluation and Optimization of Virtual Screening Workflows with DEKOIS 2.0 - A Public Library of Challenging Docking Benchmark Sets. *Journal of chemical information and modeling*, 53(6):1447–62, June 2013. doi:10.1021/ci400115b.
- [BLRR⁺10] R Barbera, G La Rocca, R Rotondo, A Falzone, P Maggi, and N Venuti. Conjugating Science Gateways and Grid Portals into e-Collaboration Environments: The Liferay and GENIUS/EnginFrame Use Case. In *Proceedings of the 2010 TeraGrid Conference*, page 1. ACM, 2010.
- [BMW⁺13] Stephan Beisken, Thorsten Meinl, Bernd Wiswedel, Luis de Figueiredo, Michael Berthold, and Christoph Steinbeck. KNIME-CDK: Workflow-driven Cheminformatics. *BMC Bioinformatics*, 14(1):257, 2013. doi:10.1186/1471-2105-14-257.

- [BPvGH81] Herman JC Berendsen, James PM Postma, Wilfred F van Gunsteren, and Jan Hermans. Interaction Models for Water in Relation to Protein Hydration. In *Intermolecular forces*, pages 331–342. Springer, 1981.
- [Bra09] Jan Brase. DataCite - A Global Registration Agency for Research Data. In *Cooperation and Promotion of Information Resources in Science and Technology, 2009. COINFO'09. Fourth International Conference on*, pages 257–261. IEEE, 2009.
- [BSP⁺ed] Krzysztof Benedyczak, Bernd Schuller, Maria Petrova, Jędrzej Rybicki, and Richard Grunzke. UNICORE 7 - Middleware Services for Distributed and Federated Computing. In *International Conference on High Performance Computing Simulation (HPCS)*, submitted.
- [BvdSvD95] Herman JC Berendsen, David van der Spoel, and Rudi van Drunen. GROMACS: A Message-Passing Parallel Molecular Dynamics Implementation. *Computer Physics Communications*, 91(1):43–56, 1995.
- [BWF⁺00] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, TN Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The Protein Data Bank. *Nucleic acids research*, 28(1):235–242, 2000.
- [CCQ⁺11] Gen-Tao Chiang, Peter Clapham, Guoying Qi, Kevin Sale, and Guy Coates. Implementing a Genomic Data Management System using iRODS in the Wellcome Trust Sanger Institute. *BMC bioinformatics*, 12(1):361, 2011.
- [CDM13] CDMI. Cloud Data Management Interface (CDMI), 2013. URL: http://www.snia.org/tech_activities/standards/curr_standards/cdmi/.
- [CFK⁺00] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *Journal of network and computer applications*, 23(3):187–200, 2000.
- [CGP⁺10] Adrian Casajus, Ricardo Graciani, Stuart Paterson, Andrei Tsaregorodtsev, et al. DIRAC pilot framework and the DIRAC Workload Management System. In *Journal of Physics: Conference Series*, volume 219. IOP Publishing, 2010.
- [CKL⁺13] Michele Carpené, Iraklis A Klampanos, Siew Hoon Leong, Emanuele Casarotti, Peter Danecek, Graziella Ferini, André Gemünd, Amrey Krause, Lion Krischer, Federica Magnoni, et al. Towards Addressing CPU-intensive Seismological Applications in Europe. In *Supercomputing*, pages 55–66. Springer, 2013.
- [CNK⁺12] Eryk Ciepiela, Piotr Nowakowski, Joanna Kocot, Daniel Hareźlak, Tomasz Gubała, Jan Meizner, Marek Kasztelnik, Tomasz Bartyński, Maciej Malawski, and Marian Bubak. Managing Entire Lifecycles of e-Science Applications in the GridSpace2 Virtual Laboratory - From Motivation through Idea to Operable Web-accessible Environment Built on Top of PL-Grid e-Infrastructure. In *Building a National Distributed e-Infrastructure-PL-Grid*, pages 228–239. Springer, 2012.

- [Cro06] Douglas Crockford. The Application/JSON Media Type for Javascript Object Notation (JSON). 2006.
- [CSB⁺10] Claudio Collinet, Martin Stoter, Charles R. Bradshaw, Nikolay Samusik, Jochen C. Rink, Denise Kenski, Bianca Habermann, Frank Buchholz, Robert Henschel, Matthias S. Mueller, Wolfgang E. Nagel, Eugenio Fava, Yannis Kalaidzidis, and Marino Zerrial. Systems Survey of Endocytosis by Multiparametric Image Analysis. *Nature*, 464(7286):243–249, March 2010.
- [CSK⁺11] John Congote, Alvaro Segura, Luis Kabongo, Aitor Moreno, Jorge Posada, and Oscar Ruiz. Interactive Visualization of Volumetric Data with WebGL in Real-time. In *Proceedings of the 16th International Conference on 3D Web Technology*, pages 137–146. ACM, 2011.
- [CWO⁺11] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, et al. Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 143–157. ACM, 2011.
- [CYCA06] Christian M Chilan, M Yang, Albert Cheng, and Leon Arber. Parallel I/O performance study with HDF5, A Scientific Data Package. *TeraGrid 2006: Advancing Scientific Discovery*, 2006.
- [DBM⁺11] Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, Franck Cappello, Barbara Chapman, Xuebin Chi, Alok Choudhary, Sudip Dosanjh, Thom Dunning, Sandro Fiore, Al Geist, Bill Gropp, Robert Harrison, Mark Hereld, Michael Heroux, Adolfo Hoisie, Koh Hotta, Zhong Jin, Yutaka Ishikawa, Fred Johnson, Sanjay Kale, Richard Kenway, David Keyes, Bill Kramer, Jesus Labarta, Alain Lichnewsky, Thomas Lippert, Bob Lucas, Barney Maccabe, Satoshi Matsuoka, Paul Messina, Peter Michielse, Bernd Mohr, Matthias S. Mueller, Wolfgang E. Nagel, Hiroshi Nakashima, Michael E Papka, Dan Reed, Mitsuhisa Sato, Ed Seidel, John Shalf, David Skinner, Marc Snir, Thomas Sterling, Rick Stevens, Fred Streitz, Bob Sugar, Shinji Sumimoto, William Tang, John Taylor, Rajeev Thakur, Anne Trefethen, Mateo Valero, Aad Van Der Steen, Jeffrey Vetter, Peg Williams, Robert Wisniewski, and Kathy Yelick. The International Exascale Software Project Roadmap. *Int. J. High Perform. Comput. Appl.*, 25(1):3–60, February 2011. doi:10.1177/1094342010391989.
- [dCa15] dCache. Installations around the World, 2015. URL: <https://www.dcache.org/dcache-map.html>.
- [DCD⁺06] Ben Domenico, John Caron, Ethan Davis, Robb Kambic, and Stefano Nativi. Thematic Real-time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL. *Journal of Digital Information*, 2(4), 2006.
- [DCG⁺14] Daniele Dagostino, Andrea Clematis, Antonella Galizia, Alfonso Quarati, Emanuele

- Danovaro, Luca Roverelli, Gabriele Zereik, Dieter Kranzlmuller, Michael Schiffers, Nils Gentschen Felde, et al. The DRIHM Project: A Flexible Approach to Ontegrate HPC, Grid and Cloud Resources for Hydro-meteorological Research. In *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*, pages 536–546. IEEE, 2014.
- [DFN15a] DFN. DFN-Verein - Authentication and Authorization Infrastructure (AAI). 2015. URL: <https://www.aai.dfn.de/>.
- [DFN15b] DFN. German National Research and Education Network, 2015. URL: <https://www.dfn.de/en/>.
- [DGST09] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-Science: An Overview of Workflow System Features and Capabilities. *Future Gener. Comput. Syst.*, 25(5):528–540, May 2009. doi:10.1016/j.future.2008.06.012.
- [dI11] Kommission Zukunft der Informationsinfrastruktur. Gesamtkonzept für die Informationsinfrastruktur in Deutschland, 2011. URL: <http://www.leibniz-gemeinschaft.de/infrastrukturen/kii/>.
- [dIGKS⁺13] Luis de la Garza, Jens Krüger, Charlotta Schärfe, Marc Röttig, Stephan Aiche, Knut Reinert, and Oliver Kohlbacher. From the Desktop to the Grid: Conversion of KNIME Workflows to gUSE. In *Proceedings of the International Workshop on Scientific Gateways 2013 (IWSG)*, 2013.
- [DSH⁺10] Bastian Demuth, Bernd Schuller, Sonja Holl, Jason Daivandy, Andre Giesler, Valentina Huber, and Sulev Sild. The UNICORE Rich Client: Facilitating the Automated Execution of Scientific Workflows. In *e-Science (e-Science), 2010 IEEE Sixth International Conference on*, pages 238–245. IEEE, 2010.
- [Dur13] DuraSpace Organization. Duraspace, 2013. URL: <http://duraspace.org/>.
- [DVJ⁺14] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, and Kent Wenger. Pegasus, A Workflow Management System for Science Automation. *Future Gener. Comput. Syst.*, 2014. doi:10.1016/j.future.2014.10.008.
- [DVS⁺12] Rion Dooley, Matthew Vaughn, Dan Stanzione, Steve Terry, and Edwin Skidmore. Software-as-a-service: The iPlant Foundation API. In *5th IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*. Citeseer, 2012.
- [EGI15] EGI. European Grid Infrastructure, 2015. URL: <https://www.egi.eu>.
- [EGK⁺07] Mattias Ellert, Michael Grønager, Aleksandr Konstantinov, Balázs Kónya, Jonas Lindemann, Ilja Livenson, Jakob Langgaard Nielsen, Marko Niinimäki, Oxana Smirnova, and Anders Wäänänen. Advanced Resource Connector Middleware for Lightweight Computational Grids. *Future Generation computer systems*, 23(2):219–240, 2007.

- [Epr13] Eprints. Eprints - Digital Repository Software, 2013. URL: <http://www.eprints.org/>.
- [EUD15a] EUDAT. EUDAT Semantics Working Group, 2015. URL: <http://eudat.eu/semantics>.
- [EUD15b] EUDAT. Metadata, 2015.
- [Eur10] EuropeanUnion. Riding the Wave - How Europe can Gain from the Rising Tide of Scientific Data, 2010. URL: <http://cordis.europa.eu/fp7/ict/e-infrastructure/docs/hlg-sdi-report.pdf>.
- [EWW⁺07] Thomas Eickermann, Lidia Westphal, Oliver Wäldrich, Wolfgang Ziegler, Christoph Barz, and Markus Pilz. Co-Allocating Compute and Network Resources. In *Towards Next Generation Grids*, pages 193–202. Springer, 2007.
- [Fac15] Facebook. Connect with Friends and the World Around you on Facebook., 2015. URL: <http://www.facebook.com/>.
- [FAF⁺11] Sandro Fiore, Giovanni Aloisio, Peter Fox, Monique Petitdidier, Horst Schwichtenberg, Sebastien Denvil, Jonathan D. Blower, and Antonio Cofino. The Climate-G Testbed: Towards Large Scale Distributed Data Management for Climate Change. *Procedia Computer Science*, 4(0):567 – 576, 2011. Proceedings of the International Conference on Computational Science, ICCS 2011. URL: <http://www.sciencedirect.com/science/article/pii/S1877050911001177>, doi:10.1016/j.procs.2011.04.059.
- [FCY99] Mike Folk, Albert Cheng, and Kim Yates. HDF5: A File Format and I/O Library for High Performance Computing Applications. In *Proceedings of Supercomputing*, 1999.
- [Fed16] FedoraCommons. Fedora Commons Repository Software. 2016. URL: <http://fedora-commons.org/>.
- [FG06] Patrick Fuhrmann and Volker Güllow. dCache, Storage System for the Future. In *Euro-Par 2006, Parallel Processing, 12th International Euro-Par Conference, Dresden, Germany, August 28 - September 1, 2006, Proceedings*, volume 4128 of *Lecture Notes in Computer Science*, pages 1106–1113. Springer, 2006. doi:10.1007/11823285_116.
- [FHA05] Rene Fritz, Daniel Hinderink, and Werner Altmann. TYPO3: Enterprise Content Management. *PACKT PUB*, 2005.
- [FHK⁺11] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An Overview of the HDF5 Technology Suite and its Applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, pages 36–47. ACM, 2011.
- [FK97] Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications*, 11(2):115–128, 1997.

- [FK11] Zoltan Farkas and Peter Kacsuk. P-GRADE Portal: A Generic Workflow System to Support User Communities. *Future Generation Computer Systems*, 27(5):454–465, 2011.
- [FKTT98] Ian Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A Security Architecture for Computational Grids. In *Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92. ACM, 1998.
- [FNA11] Sandro Fiore, Alessandro Negro, and Giovanni Aloisio. The Data Access Layer in the GRelC System Architecture. *Future Generation Computer Systems*, 27(3):334–340, 2011.
- [Fos05] Ian Foster. Globus Toolkit Version 4: Software for Service-oriented Systems. In *Network and parallel computing*, pages 2–13. Springer, 2005.
- [Fou15] Apache Software Foundation. Pluto, 2015. URL: <https://portals.apache.org/pluto/>.
- [FS03] Niels Ferguson and Bruce Schneier. *Practical Cryptography*, volume 23. Wiley New York, 2003.
- [FTS⁺09] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, Jr. Montgomery, J. A., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Ö. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J. Fox. Gaussian 09, Revision A. 02, Gaussian. Inc., Wallingford, CT, 200, 2009.
- [Fuh04] Patrick Fuhrmann. dCache, The Overview. *White paper*, 2004.
- [FZ11] Ixchel M Faniel and Ann Zimmerman. Beyond the Data Deluge: A Research Agenda for Large-Scale Data Sharing and Reuse. *International Journal of Digital Curation*, 6(1):58–69, 2011. doi:10.2218/ijdc.v6i1.172.
- [GAB⁺05] Ilya G Goldberg, Chris Allan, Jean-Marie Burel, Doug Creager, Andrea Falconi, Harry Hochheiser, Josiah Johnston, Jeff Mellen, Peter K Sorger, and Jason R Swedlow. The Open Microscopy Environment (OME) Data Model and XML File: Open Tools for Informatics and Quantitative Analysis in Biological Imaging. *Genome biology*, 6(5):R47, 2005.
- [GBA⁺10] Carole A Goble, Jiten Bhagat, Sergejs Aleksejevs, Don Cruickshank, Danus Michaelides, David Newman, Mark Borkum, Sean Bechhofer, Marco Roos, Peter Li,

- et al. myExperiment: A Repository and Social Network for the Sharing of Bioinformatics Workflows. *Nucleic acids research*, 38(suppl 2):W677–W682, 2010.
- [GBB⁺12] Richard Grunzke, Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Sandra Gesing, Sonja Herres-Pawlis, Oliver Kohlbacher, Jens Krüger, Martin Kruse, Ralph Müller-Pfefferkorn, Patrick Schäfer, Bernd Schuller, Thomas Steinke, and Andreas Zink. A Data Driven Science Gateway for Computational Workflows. In *UNICORE Summit 2012 Proceedings*, volume 15 of *IAS Series*, pages 35–49, 2012. URL: <http://hdl.handle.net/2128/4705>.
- [GBG⁺14] Richard Grunzke, Sebastian Breuers, Sandra Gesing, Sonja Herres-Pawlis, Martin Kruse, Dirk Blunk, Luis de la Garza, Lars Packschies, Patrick Schäfer, Charlotta Schärfe, Tobias Schlemmer, Thomas Steinke, Bernd Schuller, Ralph Müller-Pfefferkorn, René Jäkel, Wolfgang E. Nagel, Malcolm Atkinson, and Jens Krüger. Standards-based Metadata Management for Molecular Simulations. *Concurrency and Computation: Practice and Experience*, 26(10):1744–1759, 2014. doi:10.1002/cpe.3116.
- [GBH⁺11] A. O. Garcia, S. Bourov, A. Hammad, V. Hartmann, T. Jejkal, J. C. Otte, S. Pfeiffer, T. Schenker, C. Schmidt, P. Neuberger, R. Stotzka, J. Van Wezel, B. Neumair, and A. Streit. Data-Intensive Analysis for Scientific Experiments at the Large Scale Data Facility. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 125–126, 2011. doi:10.1109/ldav.2011.6092331.
- [GDP⁺15] Sandra Gesing, Rion Dooley, Marlon Pierce, Jens Krüger, Richard Grunzke, Sonja Herres-Pawlis, and Alexander Hoffmann. Science Gateways - Leveraging Modeling and Simulations in HPC Infrastructures via Increased Usability. In *International Conference on High Performance Computing Simulation (HPCS)*, pages 19–26, July 2015. doi:10.1109/HPCSim.2015.7237017.
- [Gen01] Wolfgang Gentsch. Sun Grid Engine: Towards Creating a Compute Power Grid. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pages 35–36. IEEE, 2001.
- [GGB⁺11] Sandra Gesing, Richard Grunzke, Ákos Balaskó, Georg Birkenheuer, Dirk Blunk, Sebastian Breuers, Andre Brinkmann, Gregor Fels, Sonja Herres-Pawlis, Peter Kacsuk, Miklos Kozlovsky, Jens Krüger, Lars Packschies, Patrick Schäfer, Bernd Schuller, Johannes Schuster, Thomas Steinke, Anna Szikszay Fabri, Martin Wewior, Ralph Müller-Pfefferkorn, and Oliver Kohlbacher. Granular Security for a Science Gateway in Structural Bioinformatics. In *Proceedings of the 3rd International Workshop on Science Gateways for Life Sciences*, CEUR Workshop Proceedings, 2011. URL: <http://ceur-ws.org/Vol-819/paper8.pdf>.
- [GGJN14] Richard Grunzke, Sandra Gesing, René Jäkel, and Wolfgang E. Nagel. Towards Generic Metadata Management in Distributed Science Gateway Infrastructures. In *IEEE/ACM CCGrid 2014 (14th International Symposium on Cluster, Cloud and Grid Computing)*, pages 566–570, Chicago, IL, US, May 2014. doi:10.1109/CCGrid.2014.98.

- [GGK⁺12] Sandra Gesing, Richard Grunzke, Jens Krüger, Georg Birkenheuer, Martin Wewior, Patrick Schäfer, Bernd Schuller, Johannes Schuster, Sonja Herres-Pawlis, Sebastian Breuers, Ákos Balaskó, Miklos Kozlovsky, Anna Szikszay Fabri, Lars Packschies, Peter Kacsuk, Dirk Blunk, Thomas Steinke, Andre Brinkmann, Gregor Fels, Ralph Müller-Pfefferkorn, René Jäkel, and Oliver Kohlbacher. A Single Sign-On Infrastructure for Science Gateways on a Use Case for Structural Bioinformatics. *Journal of Grid Computing*, 10(4):769–790, 2012. URL: <http://link.springer.com/article/10.1007%2Fs10723-012-9247-y>, doi:10.1007/s10723-012-9247-y.
- [GGL03] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 29–43. ACM, 2003.
- [GKG⁺14a] Sandra Gesing, Jens Krüger, Richard Grunzke, Luis de la Garza, Sonja Herres-Pawlis, and Alexander Hoffmann. Molecular Simulation Grid (MosGrid): A Science Gateway Tailored to the Molecular Simulation Community. In *Science Gateways for Distributed Computing Infrastructures*, pages 151–165. Springer International Publishing, 2014. doi:10.1007/978-3-319-11268-8_11.
- [GKG⁺14b] Richard Grunzke, Jens Krüger, Sandra Gesing, Sonja Herres-Pawlis, Alexander Hoffmann, and Luis de la Garza. Improved Resilience and Usability for Science Gateway Infrastructures via Integrated Virtual Organizations. In *EGI Community Forum 2014*, 2014.
- [GKG⁺15] Richard Grunzke, Jens Krüger, Sandra Gesing, Sonja Herres-Pawlis, Alexander Hoffmann, Alvaro Aguilera, and Wolfgang E. Nagel. Managing Complexity in Distributed Data Life Cycles Enhancing Scientific Discovery. In *IEEE 11th International Conference on e-Science*, pages 371–380, August 2015. doi:10.1109/eScience.2015.72.
- [GMO96] David S Goodsell, Garrett M Morris, and Arthur J Olson. Automated Docking of Flexible Ligands: Applications of AutoDock. *Journal of Molecular Recognition*, 9(1):1–5, 1996.
- [GMPMM11] Richard Grunzke, Ralph Müller-Pfefferkorn, Ulf Markwardt, and Matthias Müller. Advancing Cutting-Edge Biological Research with a High-Throughput UNICORE Workflow. In *UNICORE Summit 2011 Proceedings*, volume 09 of *IAS Series*, pages 35–43, 2011. URL: <http://hdl.handle.net/2128/4518>.
- [GNT⁺10] Jeremy Goecks, Anton Nekrutenko, James Taylor, T Galaxy Team, et al. Galaxy: A Comprehensive Approach for Supporting Accessible, Reproducible, and Transparent Computational Research in the Life Sciences. *Genome Biol*, 11(8):R86, 2010.
- [GP09] Christian Grimme and Alexander Papaspyrou. Cooperative Negotiation and Scheduling of Scientific Workflows in the Collaborative Climate Community Data and Processing Grid. *Future Generation Computer Systems*, 25(3):301–307, 2009.
- [GPW⁺06] James Gallagher, Nathan Potter, Patrick West, Jose Garcia, and Peter Fox. OPeNDAP’s Server4: Building a high performance data server for the DAP using existing software. In *AGU meeting in San Francisco*, 2006.

- [Gra06] Hagen Graf. *Drupal: Community-Sites entwickeln und verwalten mit dem Open-Source-CMS*. Addison-Wesley, 2006.
- [Gri15] GridKa. Grid Computing Centre Karlsruhe, 2015. URL: <http://www.gridka.de>.
- [Grö11] Marko Grönroos. *Book of Vaadin*. Lulu.com, 2011.
- [GS10] Richard Grunzke and Bernd Schuller. Secure High-Throughput Computing Using UNICORE XML Spaces. In *UNICORE Summit 2010 Proceedings*, volume 5 of IAS, pages 27–35, 2010. URL: <http://hdl.handle.net/2128/3812>.
- [GSP09] Zhenhua Guo, Raminderjeet Singh, and Marlon Pierce. Building the PolarGrid Portal Using Web 2.0 and OpenSocial. In *Proceedings of the 5th Grid Computing Environments Workshop, GCE '09*, pages 5:1–5:8, New York, NY, USA, 2009. ACM. doi:10.1145/1658260.1658267.
- [gUS15] gUSE. Available Science Gateways. <http://guse.hu/portals/sg>, 2015. URL: <http://guse.hu/portals/sg>, doi:<http://guse.hu/portals/sg>.
- [GVM⁺11] Stephen A Goff, Matthew Vaughn, Sheldon McKay, Eric Lyons, Ann E Stapleton, Damian Gessler, Naim Matasci, Liya Wang, Matthew Hanlon, Andrew Lenards, et al. The iPlant Collaborative: Cyberinfrastructure for Plant Biology. *Frontiers in Plant Science*, 2, 2011.
- [GWGC04] G Gibson, B Welch, G Goodson, and P Corbett. Parallel NFS Requirements and Design Considerations. *IETF Draft*, 2004.
- [H⁺05] Stefan Hepper et al. JSR 286: Portlet Specification 2.0. *Website Java Community Process*, 2005.
- [Han10] Robert M Hanson. JMOL - A Paradigm Shift in Crystallographic Visualization. *Journal of Applied Crystallography*, 43(5):1250–1260, 2010.
- [Har12] Dick Hardt. The OAuth 2.0 authorization framework. 2012.
- [Har15] Phil Harvey. ExifTool: Read, Write and Edit Meta Information, 2015. URL: <http://www.sno.phy.queensu.ca/~phil/exiftool/>.
- [HBP15a] HBP. The Human Brain Project, 2015. URL: <https://www.humanbrainproject.eu>.
- [HBP15b] HBP. The Human Brain Project - High Performance Computing Platform, 2015. URL: <https://www.humanbrainproject.eu/high-performance-computing-platform1>.
- [HCB⁺13] Alexander Hoffmann, Cooper Citek, Stephan Binder, Arne Goos, Michael Rübhausen, Oliver Troeppner, Ivana Ivanović-Burmazović, Erik C. Wasinger, T. Daniel P. Stack, and Sonja Herres-Pawlis. Catalytic Phenol Hydroxylation with Dioxygen: Extension of the Tyrosinase Mechanism Beyond the Protein Matrix. *Angewandte Chemie International Edition*, 52(20):5398–5401, 2013. doi:10.1002/anie.201301249.

- [HCK⁺08] Felix Hupfeld, Toni Cortes, Björn Kolbeck, Jan Stender, Erich Focht, Matthias Hess, Jesus Malo, Jonathan Marti, and Eugenio Cesario. The XtremFS Architecture - A Case for Object-based File Systems in Grids. *Concurrency and Computation: Practice and Experience*, 20(17):2049–2060, 2008. doi:10.1002/cpe.1304.
- [HDP⁺08] Richard CG Holland, Thomas A Down, Matthew Pocock, Andreas Prlić, David Huen, Keith James, Sylvain Foisy, Andreas Dräger, Andy Yates, Michael Heuer, et al. Bio-Java: An Open-Source Framework for Bioinformatics. *Bioinformatics*, 24(18):2096–2097, 2008.
- [Hen95] Robert L Henderson. Job Scheduling Under the Portable Batch System. In *Job scheduling strategies for parallel processing*, pages 279–294. Springer, 1995.
- [Her04] Christopher R Hertel. *Implementing CIFS: The Common Internet File System*. Prentice Hall Professional, 2004.
- [HFP⁺12] Laurel L Haak, Martin Fenner, Laura Paglione, Ed Pentz, and Howard Ratner. ORCID: A System to Uniquely Identify Researchers. *Learned Publishing*, 25(4):259–264, 2012.
- [HHP14] Alexander Hoffmann and Sonja Herres-Pawlis. Hiking on the Potential Energy Surface of a Functional Tyrosinase Model - Implications of Singlet, Broken-Symmetry and Triplet Description. *Chem. Commun.*, 50:403–405, 2014. doi:10.1039/C3CC46893C.
- [HHPG⁺13] Alexander Hoffmann, Sonja Herres-Pawlis, Sandra Gesing, Luis de la Garza, Jens Krüger, and Richard Grunzke. User-friendly Metaworkflows in Quantum Chemistry. In *IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–3. IEEE, 2013. doi:10.1109/CLUSTER.2013.6702700.
- [HKM09] Adrian Holovaty and Jacob Kaplan-Moss. *The Definitive Guide to Django: Web Development Done Right*. Apress, 2009.
- [HKVDSL08] Berk Hess, Carsten Kutzner, David Van Der Spoel, and Erik Lindahl. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *Journal of chemical theory and computation*, 4(3):435–447, 2008.
- [HKZ⁺11] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy H Katz, Scott Shenker, and Ion Stoica. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In *NSDI*, volume 11, pages 22–22, 2011.
- [Hoh06] Andreas Hoheisel. Grid Workflow Execution Service-Dynamic and Interactive Execution and Visualization of Distributed Workflows. In *Proceedings of the Cracow Grid Workshop*, volume 2, pages 13–24. Citeseer, 2006.
- [HPBB⁺12] Sonja Herres-Pawlis, Georg Birkenheuer, Andre Brinkmann, Sandra Gesing, Richard Grunzke, René Jäkel, Oliver Kohlbacher, Jens Krüger, and Ines Dos Santos Vieira. Workflow-enhanced Conformational Analysis of Guanidine Zinc Complexes via a Science Gateway. In *Studies in Health Technology and Informatics*, 175:142-151, IOS Press, 2012. doi:10.3233/978-1-61499-054-3-142.

- [HPBB⁺13] Sonja Herres-Pawlis, Ákos Balaskó, Georg Birkenheuer, Andre Brinkmann, Sandra Gesing, Richard Grunzke, Alexander Hoffmann, Peter Kacsuk, Jens Krüger, Lars Packschies, Gabor Terstyansky, and Noam Weingarten. User-Friendly Workflows in Quantum Chemistry. In *Proceedings of the International Workshop on Scientific Gateways 2013 (IWSG)*, 2013. URL: <http://ceur-ws.org/Vol-993/paper14.pdf>.
- [HPHB⁺15] Sonja Herres-Pawlis, Alexander Hoffmann, Ákos Balaskó, Peter Kacsuk, Georg Birkenheuer, André Brinkmann, Luis de la Garza, Jens Krüger, Sandra Gesing, Richard Grunzke, Gabor Terstyansky, and Noam Weingarten. Quantum Chemical Meta-Workflows in MoSGrid. *Concurrency and Computation: Practice and Experience*, 27(2):344–357, 2015. doi:10.1002/cpe.3292.
- [HPHG⁺14] Sonja Herres-Pawlis, Alexander Hoffmann, Luis De La Garza, Jens Kruger, Sandra Gesing, Richard Grunzke, Wolfgang E. Nagel, Gabor Terstyansky, and Noam Weingarten. Meta-Metaworkflows for Combining Quantum Chemistry and Molecular Dynamics in the MoSGrid Science Gateway. In *6th International Workshop on Science Gateways (IWSG)*, pages 73–78, June 2014. doi:10.1109/IWSG.2014.20.
- [HPHGP13] Sonja Herres-Pawlis, Alexander Hoffmann, Richard Grunzke, and Lars Packschies. Orbital Analysis of Oxo and Peroxo Dicopper Complexes via Quantum Chemical Workflows in MoSGrid. In *Proceedings of the International Workshop on Scientific Gateways 2013 (IWSG)*, 2013. URL: <http://ceur-ws.org/Vol-993/paper3.pdf>.
- [HPHR⁺15] Sonja Herres-Pawlis, Alexander Hoffmann, Thomas Rosener, Jens Kruger, Richard Grunzke, and Sandra Gesing. Multi-layer Meta-metaworkflows for the Evaluation of Solvent and Dispersion Effects in Transition Metal Systems Using the MoSGrid Science Gateways. In *Science Gateways (IWSG), 2015 7th International Workshop on*, pages 47–52, June 2015. doi:10.1109/IWSG.2015.13.
- [HS09] Jan Huisken and Didier YR Stainier. Selective Plane Illumination Microscopy Techniques in Developmental Biology. *Development*, 136(12):1963–1975, 2009.
- [HSDB⁺04] Jan Huisken, Jim Swoger, Filippo Del Bene, Joachim Wittbrodt, and Ernst HK Stelzer. Optical Sectioning Deep Inside Live Embryos by Selective Plane Illumination Microscopy. *Science*, 305(5686):1007–1009, 2004. doi:10.1126/science.1100035.
- [HSG03] Timothy A Howes, Mark C Smith, and Gordon S Good. *Understanding and Deploying LDAP Directory Services*. Addison-Wesley Longman Publishing Co., Inc., 2003.
- [HSG09] André Höing, Guido Scherp, and Stefan Gudenkauf. The BIS-Grid Engine: An Orchestration as a Service Infrastructure. *International Journal of Computing*, 8(3):96–104, 2009.
- [HTT09] Anthony JG Hey, Stewart Tansley, and Kristin Michele Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. 2009.

- [Hub12] Bernardo A Huberman. Sociology of Science: Big Data Deserves a Bigger Audience. *Nature*, 482(7385):308–308, 2012.
- [iRe16] iReceptor. iReceptor Data Management System and Science Gateway. <http://ireceptor.irmacs.sfu.ca/>, 2016.
- [JCM⁺83] William L Jorgensen, Jayaraman Chandrasekhar, Jeffrey D Madura, Roger W Impey, and Michael L Klein. Comparison of Simple Potential Functions for Simulating Liquid Water. *The Journal of chemical physics*, 79(2):926–935, 1983.
- [Jen10] Roger Jennings. *Cloud Computing with the Windows Azure Platform*. John Wiley & Sons, 2010.
- [JGG⁺14] Christopher Jung, Martin Gasthuber, Andre Giesler, Marcus Hardt, Jörg Meyer, Fabian Rigoll, Kilian Schwarz, Rainer Stotzka, and Achim Streit. Optimization of Data Life Cycles. *Journal of Physics: Conference Series*, 513(3):032047, 2014. doi:10.1088/1742-6596/513/3/032047.
- [JHS⁺12] Thomas Jejkal, Volker Hartmann, Rainer Stotzka, Jens Otte, Ariel Garcia, Jos van Wezel, and Achim Streit. LAMBDA – The LSDF Execution Framework for Data Intensive Applications. In *Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on*, pages 213–220. IEEE, 2012.
- [JMPK⁺15] René Jäkel, Ralph Müller-Pfefferkorn, Michael Kluge, Richard Grunzke, and Wolfgang E. Nagel. Architectural Implications for Exascale based on Big Data Workflow Requirements. In *Big Data and High Performance Computing*, volume 26 of *Advances in Parallel Computing*, pages 101 – 113. IOS Press, 2015. doi:10.3233/978-1-61499-583-8-101.
- [JTR88] William L Jorgensen and Julian Tirado-Rives. The OPLS [Optimized Potentials for Liquid Simulations] Potential Functions for Proteins, Energy Minimizations for Crystals of Cyclic Peptides and Crambin. *Journal of the American Chemical Society*, 110(6):1657–1666, 1988.
- [JVK⁺14] Thomas Jejkal, Alexander Vondrous, Andreas Kopmann, Rainer Stotzka, and Volker Hartmann. KIT Data Manager: The Repository Architecture Enabling Cross-Disciplinary Research. In *Large-Scale Data Management and Analysis - Big Data in Science - 1st Edition*, 2014. URL: <http://digbib.ubka.uni-karlsruhe.de/volltexte/1000043270>.
- [KBB⁺08] Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, W Carson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, et al. Exascale Computing Study: Technology Challenges in Achieving Exascale Systems. 2008.
- [KFK⁺12] Peter Kacsuk, Zoltan Farkas, Miklos Kozlovsky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai, and Istvan Marton. WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities. *Journal of Grid Computing*, 10(4):601–630, 2012. doi:10.1007/s10723-012-9240-5.

- [KGG⁺14] Jens Krüger, Richard Grunzke, Sandra Gesing, Sebastian Breuers, André Brinkmann, Luis de la Garza, Oliver Kohlbacher, Martin Kruse, Wolfgang E. Nagel, Lars Packschies, Ralph Müller-Pfefferkorn, Patrick Schäfer, Charlotta Schärfe, Thomas Steinke, Tobias Schlemmer, Klaus Dieter Warzecha, Andreas Zink, and Sonja Herres-Pawlis. The MoSGrid Science Gateway - A Complete Solution for Molecular Simulations. *Journal of Chemical Theory and Computation*, 10(6):2232–2245, 2014. doi:10.1021/ct500159h.
- [KGHP⁺14] Jens Krüger, Richard Grunzke, Sonja Herres-Pawlis, Alexander Hoffmann, Luis de la Garza, Oliver Kohlbacher, Wolfgang E. Nagel, and Sandra Gesing. Performance Studies on Distributed Virtual Screening. *BioMed Research International*, 2014. doi:10.1155/2014/624024.
- [KMBP14] Thejaka Amila Kanewala, Suresh Marru, Jim Basney, and Marlon Pierce. A Credential Store for Multi-Tenant Science Gateways. In *Cluster, Cloud and Grid Computing (CC-Grid), 2014 14th IEEE/ACM International Symposium on*, pages 445–454. IEEE, 2014.
- [Koh12] Oliver Kohlbacher. CADDSuite - A Workflow-Enabled Suite of Open-Source Tools for Drug Discovery. *Journal of Cheminformatics*, 4(Suppl 1):O2, 2012. doi:10.1186/1758-2946-4-s1-o2.
- [Kop13] Kopal. Kopal Langzeitarchiv - Daten für die Zukunft, 2013. URL: <http://kopal.langzeitarchivierung.de/>.
- [KPWW10] Robert Koopman, William Powers, Zhi Wang, and Shang-Jin Wei. Give Credit where Credit is Due: Tracing Value Added in Global Production Chains. Technical report, National Bureau of Economic Research, 2010.
- [KR98] Axel Keller and Alexander Reinefeld. CCS Resource Management in Networked HPC Systems. In *Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh*, pages 44–56. IEEE, 1998.
- [KSP08] Birger Koblit, Nuno Santos, and V Pose. The AMGA Metadata Service. *Journal of Grid Computing*, 6(1):61–76, 2008.
- [KSWWW08] Marc Kemps-Snijders, Menzo Windhouwer, Peter Wittenburg, and Sue Ellen Wright. ISOcat: Corraling Data Categories in the Wild. In *LREC*, 2008.
- [LAB⁺06] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006. doi:10.1002/cpe.994.
- [Lan11] Betty Landesman. Seeing Standards: A Visualization of the Metadata Universe. *Technical Services Quarterly*, 28(4):459–460, 2011. doi:10.1080/07317131.2011.598072.
- [LBS62] M Stanley Livingston, John P Blewett, and EE Stickley. Particle Accelerators. *American Journal of Physics*, 30(12):940–941, 1962.

- [LEP⁺06] Erwin Laure, A Edlund, F Pacini, P Buncic, M Barroso, A Di Meglio, F Prelz, A Frohner, O Mulmo, A Krenek, et al. Programming the Grid with gLite. Technical report, 2006.
- [Lin15] LinkedIn. World's Largest Professional Network, 2015. URL: <https://www.linkedin.com/>.
- [LLC⁺03] Jianwei Li, Wei-keng Liao, Alok Choudhary, Robert Ross, Rajeev Thakur, William Gropp, Robert Latham, Andrew Siegel, Brad Gallagher, and Michael Zingale. Parallel netCDF: A High-Performance Scientific I/O Interface. In *Supercomputing, 2003 ACM/IEEE Conference*, pages 39–39. IEEE, 2003.
- [LND00] Alistair G Lowe-Norris and Robert Denn. *Windows 2000 Active Directory*. O'Reilly & Associates, Inc., 2000.
- [LOP04] Javier Lopez, Rolf Oppliger, and Günther Pernul. Authentication and Authorization Infrastructures (AAIs): A Comparative Survey. *Computers & Security*, 23(7):578–590, 2004.
- [LPU15] California Digital Library, Portico, and Stanford University. JHOVE2 Project, 2015. URL: <http://www.jhove2.org/>.
- [LWE⁺13] Damien Lecarpentier, Peter Wittenburg, Willem Elbers, Alberto Michelini, Riam Kanso, Peter Coveney, and Rob Baxter. EUDAT: A New Cross-Disciplinary Data Infrastructure for Science. *International Journal of Digital Curation*, 8(1):279–287, 2013.
- [LZWD⁺15] Katherine A. Lawrence, Michael Zentner, Nancy Wilkins-Diehr, Julie A. Wernert, Marlon Pierce, Suresh Marru, and Scott Michael. Science Gateways Today and Tomorrow: Positive Perspectives of Nearly 5000 Members of the Research Community. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a, 2015. doi:10.1002/cpe.3526.
- [MAG⁺15] Gary A. McGilvary, Malcolm Atkinson, Sandra Gesing, Alvaro Aguilera, Richard Grunzke, and Eva Sciacca. Enhanced Usability of Managing Workflows in an Industrial Data Gateway. In *Interoperable Infrastructures for Interdisciplinary Big Data Sciences (IT4RIs 15)*, pages 495–502, Aug 2015. doi:10.1109/eScience.2015.62.
- [Mal14] Daniel Mallmann. EUDAT-Dienste für eine kollaborative Dateninfrastruktur. In *DFN-Forum*. Jülich Supercomputing Center, 2014.
- [Mar13] Vivien Marx. Biology: The Big Challenges of Big Data. *Nature*, 498(7453):255–260, 2013. doi:10.1038/498255a.
- [MBB⁺98] Alexander D MacKerell, Donald Bashford, MLDR Bellott, RL Dunbrack, Jeffrey D Evanseck, Martin J Field, Stefan Fischer, Jiali Gao, H Guo, Sookhee Ha, et al. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *The Journal of Physical Chemistry B*, 102(18):3586–3616, 1998.
- [MBJ00] Liam J McGuffin, Kevin Bryson, and David T Jones. The PSIPRED Protein Structure Prediction Server. *Bioinformatics*, 16(4):404–405, 2000.

- [MCC⁺04] RL Morgan, Scott Cantor, Steven Carmody, Walter Hoehn, and Ken Klingenstein. Federated Security: The Shibboleth Approach. *Educause Quarterly*, 27(4):12–17, 2004.
- [MCD⁺15] Michael McLennan, Steven Clark, Ewa Deelman, Mats Rynge, Karan Vahi, Frank McKenna, Derrick Kearney, and Carol Song. HUBzero and Pegasus: Integrating Scientific Workflows into Science Gateways. *Concurrency and Computation: Practice and Experience*, 27(2):328–343, 2015.
- [MCIS12] Michael M Mysinger, Michael Carchia, John J Irwin, and Brian K Shoichet. Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking. *Journal of medicinal chemistry*, 55(14):6582–94, July 2012. doi:10.1021/jm300687e.
- [MGH⁺11] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, Ross Gardler, et al. Apache Airavata: A Framework for Distributed Applications and Computational Workflows. In *Proceedings of the 2011 ACM workshop on Gateway computing environments*, pages 21–28. ACM, 2011.
- [MHG10] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010.
- [MK10] Michael McLennan and Rick Kennell. HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering. *Computing in Science & Engineering*, 12(2):48–53, 2010.
- [MKEKJ12] Sharath Maddineni, Joohyun Kim, Yaakoub El-Khamra, and Shantenu Jha. Distributed Application Runtime Environment (DARE): A Standards-based Middleware Framework for Science-Gateways. *Journal of Grid Computing*, 10(4):647–664, 2012.
- [MRG⁺05] Petros Maniatis, Mema Roussopoulos, T. J. Giuli, David S. H. Rosenthal, and Mary Baker. The LOCKSS Peer-to-Peer Digital Preservation System. *ACM Trans. Comput. Syst.*, 23(1):2–50, February 2005. doi:10.1145/1047915.1047917.
- [MRK⁺13] Ketan Maheshwari, Alex Rodriguez, David Kelly, Ravi Madduri, Justin Wozniak, Michael Wilde, and Ian Foster. Enabling Multi-task Computation on Galaxy-based Gateways Using Swift. In *CLUSTER 2013*, pages 1–3, Sept 2013. doi:10.1109/CLUSTER.2013.6702701.
- [MRR99] Peter Murray-Rust and Henry S Rzepa. Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles. *Journal of Chemical Information and Computer Sciences*, 39(6):928–942, 1999.
- [MSF⁺10] Brian Matthews, Shoaib Sufi, Damian Flannery, Laurent Lerusse, Tom Griffin, Michael Gleaves, and Kerstin Kleese. Using a Core Scientific Metadata Model in Large-Scale Facilities. *International Journal of Digital Curation*, 5(1):106–118, 2010. URL: <http://ijdc.net/index.php/ijdc/article/view/149>, doi:10.2218/ijdc.v5i1.146.

- [MyC13] MyCoRe. MyCoRe - Dein Repository-Framework, 2013. URL: <http://www.mycore.de/>.
- [MZ11] Chris Mattmann and Jukka Zitting. *Tika in Action*. Manning Publications Co., 2011.
- [NHM97] Gregory M. Nielson, Hans Hagen, and Heinrich Müller. Scientific Visualization, Overviews, Methodologies, and Techniques. *IEEE Computer Society*, pages 299–323, 1997.
- [NK12] Stratos Idreos Fabian Groffen Niels Nes and Stefan Manegold Sjoerd Mullender Martin Kersten. MonetDB: Two Decades of Research in Column-Oriented Database Architectures. *Data Engineering*, page 40, 2012.
- [Nor11] Barrie M North. *Joomla! 1.6: A User's Guide: Building a Successful Joomla! Powered Website*. Pearson Education, 2011.
- [NPRI09] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, and Clemente Izurieta. Comparison of JSON and XML Data Interchange Formats: A Case Study. *Caine*, 9:157–162, 2009.
- [NS10] Waquas Noor and Bernd Schuller. MMF: A Flexible Framework for Metadata Management in UNICORE. In *UNICORE Summit 2010 Proceedings*, volume 5, pages 51–60, 2010. URL: <http://hdl.handle.net/2128/3812>.
- [OK07] Nobuaki Ohno and Akira Kageyama. Scientific Visualization of Geophysical Simulation Data by the CAVE VR System with Volume Rendering. *Physics of the Earth and Planetary Interiors*, 163(1):305–311, 2007.
- [oNZ15] National Library of New Zealand. NLNZ Metadata Extraction Tool, 2015. URL: <http://www.natlib.govt.nz/services/get-advice/digital-libraries/metadata-extraction-tool>.
- [OPU13] OPUS. Online Publikationssystem Universität Stuttgart (OPUS), 2013. URL: <http://www.kobv.de/opus4/>.
- [OVMVG04] Chris Oostenbrink, Alessandra Villa, Alan E Mark, and Wilfred F Van Gunsteren. A Biomolecular Force Field based on the Free Enthalpy of Hydration and Solvation: The GROMOS Force-Field Parameter Sets 53A5 and 53A6. *Journal of computational chemistry*, 25(13):1656–1676, 2004.
- [Pas05] Norman Paskin. Digital Object Identifiers for Scientific Data. *Data Science Journal*, 4:12–20, 2005.
- [PBW⁺05] James C Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D Skeel, Laxmikant Kale, and Klaus Schulten. Scalable Molecular Dynamics with NAMD. *Journal of computational chemistry*, 26(16):1781–1802, 2005.
- [PC03] Jay W Ponder and David A Case. Force fields for protein simulations. *Advances in protein chemistry*, 66:27–85, 2003.

- [PHD⁺13] Mariya Petrova, Valentina Huber, Bastian Demuth, Krzysztof Benedyczak, and Bernd Schuller. The UNICORE Portal. In *UNICORE Summit 2013 Proceedings*, volume 21 of *IAS Series*, 2013.
- [PMG⁺14] Marlon Pierce, Suresh Marru, Lahiru Gunathilake, Thejaka Amila Kanewala, Ramin-der Singh, Saminda Wijeratne, Chathuri Wimalasena, Chathura Herath, Eran Chinthaka, Chris Mattmann, et al. Apache Airavata: Design and Directions of a Science Gateway Framework. In *Science Gateways (IWSG), 2014 6th International Workshop on*, pages 48–54. IEEE, 2014.
- [PRA15] PRACE. PRACE Research Infrastructure, 2015. URL: <http://www.prace-ri.eu/>.
- [Pro13] The Rosetta Project. Rosetta Digital Library, 2013. URL: <http://rosettaproject.org/>.
- [Pro15] Geant Project. eduGAIN - Interconnecting Federations to Link Services and Users Worldwide. 2015. URL: <http://www.geant.net/service/eduGAIN/Pages/home.aspx>.
- [PSJ⁺ed] Ajinkya Prabhune, Rainer Stotzka, Thomas Jejkal, Volker Hartmann, Margund Bach, Eberhard Schmitt, Michael Hausmann, and Juergen Hesser. An Optimized Generic Client Service API for Managing Large Datasets within a Data Repository. In *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, accepted.
- [RBB11] Tomasz Rekawek, Piotr Bała, and Krzysztof Benedyczak. Distributed Storage Management Service in UNICORE. In *UNICORE Summit 2011 Proceedings*, pages 125–133, 2011. URL: <http://hdl.handle.net/2128/4518>.
- [RD90] Russ Rew and Glenn Davis. NetCDF: An Interface for Scientific Data Access. *Computer Graphics and Applications, IEEE*, 10(4):76–82, 1990.
- [RDG⁺08] Michael Russell, Piotr Dziubecki, Piotr Grabowski, Michal Krysiński, Tomasz Kuczyński, Dawid Szjenfeld, Dominik Tarnawczyk, Gosia Wolniewicz, and Jaroslaw Nabrzyski. The Vine Toolkit: A Java Framework for Developing Grid Applications. In Roman Wyrzykowski, Jack Dongarra, Konrad Karczewski, and Jerzy Wasniewski, editors, *Parallel Processing and Applied Mathematics*, volume 4967 of *Lecture Notes in Computer Science*, pages 331–340. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-68111-3_35.
- [RHP⁺08] Nick Ragouzis, John Hughes, Rob Philpott, Eve Maler, Paul Madsen, and Tom Scavo. Security Assertion Markup Language (SAML) v2.0 Technical Overview. *OASIS Committee Draft*, 2, 2008.
- [RKLK96] M Rarey, B Kramer, T Lengauer, and G Klebe. A Fast Flexible Docking Method Using an Incremental Construction Algorithm. *Journal of Molecular Biology*, 261:470–489, 1996.

- [RMH⁺10] Arcot Rajasekar, Reagan Moore, Chien-yi Hou, Christopher A Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, et al. iRODS primer: Integrated Rule-Oriented Data System. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1):1–143, 2010.
- [RR06] David Recordon and Drummond Reed. OpenID 2.0: A Platform for User-Centric Identity Management. In *Proceedings of the second ACM workshop on Digital identity management*, pages 11–16. ACM, 2006.
- [SACF⁺12] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. Fiji: An Open-Source Platform for Biological Image Analysis. *Nature methods*, 9(7):676–682, 2012.
- [SBB⁺03] MacKenzie Smith, Mary Barton, Mick Bass, Margret Branschofsky, Greg McClellan, Dave Stuve, Robert Tansley, and Julie Harford Walker. DSpace: An Open Source Dynamic Digital Repository. 2003. URL: hdl.handle.net/1721.1/29465.
- [SBBR⁺10] Achim Streit, Piotr Bala, Alexander Beck-Ratzka, Krzysztof Benedyczak, Sandra Bergmann, Rebecca Breu, Jason Milad Daivandy, Bastian Demuth, Anastasia Eifer, André Giesler, et al. UNICORE 6 - Recent and Future Advancements. *Annals of Telecommunications-Annales des Télécommunications*, 65(11-12):757–762, 2010.
- [SBD13] Uwe Schindler, Benny Bräuer, and Michael Diepenbroek. Harvesting of Metadata with Open Access Tools. In *Earth System Modelling-Volume 6*, pages 13–20. Springer, 2013.
- [SBR12] Jan Stender, Michael Berlin, and Alexander Reinefeld. XtreamFS - A File System for the Cloud. In *Data intensive storage services for cloud environments*, pages 267–285. IGI Global Press, 2012.
- [SBR14] Bernd Schuller, Krzysztof Benedyczak, and Jędrzej Rybicki. High-Performance Computing on the Web: Extending UNICORE with RESTful Interfaces. In *The Sixth International Conference on Advances in Future Internet*. Jülich Supercomputing Center, 2014.
- [Sch03] Philip Schwan. Lustre: Building a File System for 1000-Node Clusters. In *Proceedings of the 2003 Linux Symposium*, volume 2003, 2003.
- [Sch14] Bernd Schuller. Keynote: Federated Access to High-Performance Computing and Big Data Resources. In *The Sixth International Conference on Advances in Future Internet (AFIN)*, 2014.
- [Sch15] Bernd Schuller. UNICORE Commandline Client, 2015. URL: http://www.unicore.eu/unicore/architecture/client-layer.php#anchor_ucc.
- [SDM⁺08] Bernd Schuller, Bastian Demuth, Hartmut Mix, Katharina Rasch, Mathilde Romberg, Sulev Sild, Uko Maran, Piotr Bała, Enrico Del Grosso, Mosé Casalegno, et al. ChemoMentum - UNICORE 6 based Infrastructure for Complex Applications in Science and

- Technology. In *Euro-Par 2007 Workshops: Parallel Processing*, pages 82–93. Springer, 2008.
- [SEL⁺05] Achim Streit, D Erwin, Th Lippert, Daniel Mallmann, Roger Menday, Michael Rambadt, Morris Riedel, Mathilde Romberg, Bernd Schuller, and Ph Wieder. UNICORE - From Project Results to Production Grids. *Advances in Parallel Computing*, 14:357–376, 2005.
- [Sen04] Arun Sen. Metadata Management: Past, Present and Future. *Decision Support Systems*, 37(1):151 – 173, 2004. URL: <http://www.sciencedirect.com/science/article/pii/S0167923602002087>, doi:10.1016/S0167-9236(02)00208-7.
- [SEN10] S Shepler, M Eisler, and D Noveck. Network File System (NFS) Version 4 Minor Version 1 Protocol. *IETF(online)*, <http://www.ietf.org/rfc/rfc5661.txt>, 2010.
- [Serc2] Amazon Web Services. Amazon Elastic Compute Cloud (Amazon EC2). <http://aws.amazon.com/ec2/>.
- [Sers3] Amazon Web Services. Amazon Simple Storage Service (Amazon S3). <http://aws.amazon.com/s3/>.
- [SFCW13] Romelia Salomon-Ferrer, David A Case, and Ross C Walker. An Overview of the Amber Biomolecular Simulation Package. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 3(2):198–210, 2013.
- [SGG⁺12] Tobias Schlemmer, Richard Grunzke, Sandra Gesing, Jens Krüger, Georg Birkenheuer, Ralph Müller-Pfefferkorn, and Oliver Kohlbacher. Generic User Management for Science Gateways via Virtual Organizations. In *EGI Technical Forum 2012*, 2012. URL: <https://indico.egi.eu/indico/contributionDisplay.py?contribId=125&confId=1019>.
- [SGG13] Bernd Schuller, Richard Grunzke, and Andre Giesler. Data Oriented Processing in UNICORE. In *UNICORE Summit 2013 Proceedings*, volume 21 of *IAS Series*, pages 1–6, 2013.
- [SH02] Frank B Schmuck and Roger L Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. In *FAST*, volume 2, page 19, 2002.
- [SHF02] David Solo, Russell Housley, and Warwick Ford. Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. 2002.
- [Shi07] Jamie Shiers. The worldwide LHC computing grid (worldwide LCG). *Computer physics communications*, 177(1):219–223, 2007.
- [SHJ⁺11] Rainer Stotzka, Volker Hartmann, Thomas Jejkal, Michael Sutter, Jan Van Wezel, Ariel Garcia, Marcus Hardt, Rainer Kupsch, and Serguei Bourov. Perspective of the Large Scale Data Facility (LSDF) Supporting Nuclear Fusion Applications. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, pages 373–379, 2011. doi:10.1109/pdp.2011.59.

- [SHK⁺03] Christoph Steinbeck, Yongquan Han, Stefan Kuhn, Oliver Horlacher, Edgar Luttmann, and Egon Willighagen. The Chemistry Development Kit (CDK): An open-source Java library for chemo- and bioinformatics. *Journal of chemical information and computer sciences*, 43(2):493–500, 2003.
- [SK09] Matthias Schulze and Eike Kleiner. OPUS 4: Einblick und Ausblick. 2009.
- [SKH⁺08] Jan Stender, Björn Kolbeck, Felix Hupfeld, Eugenio Cesario, Erich Focht, Matthias Hess, Jesús Malo, and Jonathan Martí. Striping without Sacrifices: Maintaining POSIX Semantics in a Parallel File System. In *LASCO*, 2008.
- [SKL07] Harold A Scheraga, Mey Khalili, and Adam Liwo. Protein-folding Dynamics: Overview of Molecular Simulation Techniques. *Annu. Rev. Phys. Chem.*, 58:57–83, 2007.
- [SKRC10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop Distributed File System. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.
- [SPHvdS05] M Marvin Seibert, Alexandra Patriksson, Berk Hess, and David van der Spoel. Reproducible Polypeptide Folding and Structure Prediction using Molecular Dynamics Simulations. *Journal of molecular biology*, 354(1):173–183, 2005.
- [Sta06] Garrick Staples. TORQUE resource manager. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 8. ACM, 2006.
- [TAD⁺11] Carol Tenopir, Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, and Mike Frame. Data Sharing by Scientists: Practices and Perceptions. *PloS one*, 6(6):e21101, 2011.
- [Tag15] TagLib. TagLib Audio Meta-Data Library, 2015. URL: <https://taglib.github.io/>.
- [TE03] Ulf Troppens and Rainer Erkens. *Speichernetze - Grundlagen und Einsatz von Fibre Channel SAN, NAS, iSCSI und InfiniBand*. dpunkt.verlag GmbH, 2003.
- [Tik15] Apache Tika. Supported Scientific Document Formats, 2015. URL: https://tika.apache.org/1.9/formats.html#Scientific_formats.
- [Tod14] William L Todsén. ChemDoodle 6.0. *Journal of chemical information and modeling*, 54(8):2391–2393, 2014.
- [Tom15] Apache Tomcat. An Open Source Software Implementation of the Java Servlet and JavaServer Pages Technologies, 2015. URL: <http://tomcat.apache.org/>.
- [TP12] A Tsaregorodtsev and S Poss. DIRAC File Replica and Metadata Catalog. In *Journal of Physics: Conference Series*, page 032108. IOP Publishing, 2012.
- [TSJ⁺10] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy. Hive—a petabyte scale data

- warehouse using hadoop. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 996–1005. IEEE, 2010.
- [TSWH07] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. The Triana Workflow Environment: Architecture and Applications. pages 320–339. Springer London, 2007. doi:10.1007/978-1-84628-757-2{_}20.
- [TTH11] Kristin M Tolle, D Stewart W Tansley, and Anthony JG Hey. The Fourth Paradigm: Data-Intensive Scientific Discovery [Point of View]. *Proceedings of the IEEE*, 99(8):1334–1337, Aug 2011. doi:10.1109/JPROC.2011.2155130.
- [UNE03] UNESCO. Charter on the Preservation of Digital Heritage, 2003. URL: http://portal.unesco.org/en/ev.php-URL_ID=17721&URL_DO=DO_TOPIC&URL_SECTION=201.html.
- [Uni15] Unity. Unity - Cloud Identity and Federation Management, 2015. URL: <http://unity-idm.eu>.
- [UrM13] UrMEL. Universal Multimedia Electronic Library (UrMEL), 2013. URL: <http://www.urmel-dl.de/>.
- [Vaa15] Vaadin. Vaadin Charts, 2015. URL: <https://vaadin.com/add-ons/charts>.
- [Van06] Dan Vanderkam. Dygraphs Javascript Charting Library, 2006.
- [VAV15] VAVID. VAVID Project - Comparative Analysis of Engineering relevant Measurement and Simulation Data, 2015. URL: <http://vavid.de/>.
- [VBG⁺10] Marat Valiev, Eric J Bylaska, Niranjana Govind, Karol Kowalski, Tjerk P Straatsma, Hubertus JJ Van Dam, Dunyou Wang, Jarek Nieplocha, Edoardo Apra, Theresa L Windus, et al. NWChem: A Comprehensive and Scalable Open-Source Solution for Large Scale Molecular Simulations. *Computer Physics Communications*, 181(9):1477–1489, 2010.
- [VdSNLW04] Herbert Van de Sompel, Michael L Nelson, Carl Lagoze, and Simeon Warner. Resource Harvesting within the OAI-PMH Framework. *D-lib magazine*, 10(12):1082–9873, 2004. URL: <http://www.dlib.org/dlib/december04/vandesompel/12vandesompel.html>.
- [VDSPS07] David Van Der Spoel, Alexandra Patriksson, and M Marvin Seibert. Protein folding properties from molecular dynamics simulations. In *Applied Parallel Computing. State of the Art in Scientific Computing*, pages 109–115. Springer, 2007.
- [VMD⁺13] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache Hadoop Yarn: Yet Another Resource Negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.
- [VP09] Alexander Voss and Rob Procter. Virtual Research Environments in Scholarly Work and Communications. *Library Hi Tech*, 27(2):174–190, 2009.

- [WAW⁺13] Justin M Wozniak, Timothy G Armstrong, Mark Wilde, Daniel S Katz, Ewing Lusk, and Ian T Foster. Swift/T: Large-Scale Application Composition via Distributed-Memory Dataflow Processing. In *Proc. IEEE/ACM CCGRID '13*, pages 95–102, May 2013. doi : 10.1109/CCGrid.2013.99.
- [WD07] Nancy Wilkins-Diehr. Special Issue: Science Gateways - Common Community Interfaces to Grid Resources. *Concurrency and Computation: Practice and Experience*, 19(6):743–749, 2007.
- [WDGK⁺08] Nancy Wilkins-Diehr, Dennis Gannon, Gerhard Klimeck, Scott Oster, and Sudhakar Pamidighantam. TeraGrid Science Gateways and their Impact on Science. *Computer*, 41(11):32–41, 2008.
- [Web15] WebGL. WebGL, 2015. URL: <https://www.khronos.org/news/press/khronos-releases-final-webgl-1.0-specification>.
- [WGL⁺12] Xiaolong Wen, Genqiang Gu, Qingchun Li, Yun Gao, and Xuejie Zhang. Comparison of open-source cloud management platforms: OpenStack and OpenNebula. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2457–2461. IEEE, 2012.
- [WHF⁺13] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalgo, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble. The Taverna Workflow Suite: Designing and Executing Workflows of Web Services on the Desktop, Web or in the Cloud. *Nucleic Acids Research*, 41(W1):W557–W561, 2013. URL: <http://nar.oxfordjournals.org/content/41/W1/W557.abstract>, doi:10.1093/nar/gkt328.
- [Whi12] Tom White. *Hadoop: The Definitive Guide*. " O'Reilly Media, Inc.", 2012.
- [Wis12] Wissenschaftsrat. Empfehlungen zur Weiterentwicklung der wissenschaftlichen Informationsinfrastrukturen in Deutschland bis 2020, July 2012. URL: <http://www.wissenschaftsrat.de/download/archiv/2359-12.pdf>.
- [Wit08] Michael Witt. Institutional Repositories and Research Data Curation in a Distributed Environment. *Library Trends*, 57(2):191–201, 2008.
- [WKLW98] Stuart Weibel, John Kunze, Carl Lagoze, and Misha Wolf. Dublin Core Metadata for Resource Discovery. *Internet Engineering Task Force RFC*, 2413:222, 1998.
- [WLC15] WLCG. Storage Deployment, 2015. URL: <https://twiki.cern.ch/twiki/bin/view/LCG/WLCGT0T1GridServices>.
- [WMH13] Michael Weber, Michaela Mickoleit, and Jan Huisken. Light Sheet Microscopy. *Methods in cell biology*, 123:193–215, 2013.

-
- [XSE15] XSEDE. Extreme Science and Engineering Discovery Environment, 2015. URL: <https://www.xsede.org>.
- [YJG03] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple Linux Utility for Resource Management. In *Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.
- [ZCF⁺10] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2nd USENIX conference on Hot topics in Cloud Computing*, pages 10–10, 2010.
- [Zho92] Songnian Zhou. LSF: Load Sharing in Large Heterogeneous Distributed Systems. In *Workshop on Cluster Computing*, 1992.

