Standards-based Models and Architectures to Automate Scalable and Distributed Data Processing and Analysis

Mohammad Shahbaz Memon

Faculty of Industrial Engineering,
Mechanical Engineering and Computer Science
University of Iceland
2019

Standards-based Models and Architectures to Automate Scalable and Distributed Data Processing and Analysis

Mohammad Shahbaz Memon

Dissertation submitted in partial fulfilment of a
Philosophiae Doctor degree in Computer Science

Advisor
Morris Riedel

PhD Committee
Morris Riedel
Helmut Neukirchen
Matthias Book

Opponents
Ramin Yahyapour
Róbert Lovas

Faculty of Industrial Engineering,
Mechanical Engineering and Computer Science
School of Engineering and Natural Sciences
University of Iceland
Reykjavik, September 2019

Standards-based Models and Architectures to Automate Scalable and Distributed Data Processing and Analysis

Dissertation submitted in partial fulfilment of a *Philosophiae Doctor* degree in Computer Science

Faculty of Industrial Engineering,
Mechanical Engineering and Computer Science
School of Engineering and Natural Sciences
University of Iceland
Dunhagi 5
107 Reykjavik
Iceland

Telephone: 525-4000

Bibliographic information:
Mohammad Shahbaz Memon, 2019, *Standards-based Models and Architectures to Automate Scalable and Distributed Data Processing and Analysis*, PhD dissertation, Faculty of Industrial Engineering,
Mechanical Engineering and Computer Science, University of Iceland, 102 pp.

ISBN 978-9935-9473-4-5

Printing: Háskólaprent
Reykjavik, Iceland, September 2019

Abstract

Scientific communities engaging in big data analysis face numerous challenges in managing complex computations and the related data on emerging and distributed computing infrastructures. Large-scale data analysis requires applications with simplified access to multiple resource management systems. Several generic or domain-specific technologies have been developed to exploit diversified computing environments, but due to the heterogeneity of computing and data architectures they are not capable of enabling real science cases. Scientific gateways and workflows are one such example which requires the management of jobs on multiple kinds of batch systems using heterogeneous supercomputing architectures and access to advanced distributed file systems. To support these requirements, a unified architectural framework is presented in this dissertation that coalesces the right combination of standards and adequate middleware realisation. This framework manages concurrent access for diversified user communities through consistent and robust computing and data interfaces oriented to current application and infrastructure demands.

The investigations reported in this dissertation were mainly motivated by physical and machine-learning models, represented by two scientific case studies: biophysics and Earth sciences. In the field of biophysics, the UltraScan scientific gateway is enhanced to enable the processing of domain-specific data through standards-based job and data management interfaces in *High-Performance Computing* (HPC) environments. The second domain deals with Earth sciences and automates the processing of machine-learning algorithms (e.g. classification of remote sensing images) using scalable and parallel implementations. As proof of concept, both the case studies are supported through open source implementations, in the form of middleware realisation, client APIs and their integration with state-of-the-art science gateway frameworks.

Keywords— High-Performance Computing, Scientific Workflows, Distributed Computing, Open Standards, Job Execution, Data Analysis

Útdráttur

Vísindasamfélög sem vinna með stórtæk gögn kljást við margskonar áskoranir í sambandi við meðhöndlun flókinna útreikninga, og gögnum þeim tengdum, á komandi og dreifðum kerfum. Stórtæk gagnagreining kallar á lausnir með einfölduðu aðgengi að margvíslegum tölvurekstrarkerfum. Margar almennar og sértækar aðferðir hafa verið þróaðar til að nota síbreytileg reiknikerfi, en vegna ólíkra reikniaðferða og þeirra gagnaskipan geta þær ekki framkvæmt alvöru vísindarannsóknir. Vísindalegar gagnagáttir og vinnuferli eru dæmi um slíkt sem þarfnast verkmeðhöndlunar á margvíslegum bunkakerfum á ólíkum ofurtölvuhögum og aðgengi að háþróuðum dreifðum skráarkerfum. Til að styðja þessar kröfur er í þessari doktorsritgerð kynntur högunarrammi sem sameinar réttu samsetninguna af stöðlum og uppsetningu fullnægjandi millibúnaðar. Þessi rammi meðhöndlar samhliða aðgang fyrir fjölbreytta notandahópa í gegnum öflug og áreiðanleg reikni- og gagnasnið sem eru sniðin að þörfum forrita og tölvukerfainnviðum.

Rannsóknaniðurstöðurnar sem eru kynntar í þessari doktorsritgerð eru aðalega rökstuddar með raun- og vélarnámsmódelum frá tveimur dæmum frá jafnmörgum fræðasviðum: lífeðlisfræði og jarvísindum. Fyrir lífeðlisfræði er UltraScan vísindagáttin betrumbætt til þess að gera henni kleift að meðhöndla sértæk gögn í gegnum stöðluð verkumsjónar- og gagnastjórnunarsnið í háhraða tölvukerfum (HPC). Seinna fræðisviðið er jarðvísindi og gerir meðhöndlun vélarnámsaðferða sjálfvirka (t.d. greiningu fjarkönnunarmyndefnis) með stigvaxand útfærslum sem hægt er að keyra samhliða. Dæmin frá bæðum fræðisviðum eru studd með opnum hugbúnaði í formi millibúnaðarútfærslna, biðlaraforritaskil með bestu gáttarömmum sem fyrirfinnast í dag, til þess að sanna gildi þeirra.

Lykilorð— Ofurtölvureikningar, Vísindalegt vinnuflæði, Dreifð útreikningar, Opinn staðall, Verkmeðhöndlun, Gagnagreining

Table of Contents

List of Figures

List of Tables

List of Publications

Paper I: **M. S. Memon**, M. Riedel, A. S. Memon, C. Koeritz, A. Grimshaw, and H. Neukirchen. Enabling Scalable Data Processing and Management through Standards-based Job Execution and the Global Federated File System, *Journal of Scalable Computing: Practice and Experience*, Vol 17 No 2, 115–128 (2016). DOI: 10.12694/scpe.v17i2.1160

Paper II: **M. S. Memon**, M. Riedel, F. Janetzko, B. Demeler, G. Gorbet, S. Marru, A. Grimshaw, L. Gunathilake, R. Singh, N. Attig, and T. Lippert. Advancements of the UltraScan Scientific Gateway for Open Standards-based Cyberinfrastructures, *Journal of Concurrency and Computation*, Vol 26 Issue 13, 2280–2291 (2014). DOI: 10.1002/cpe.3251

Paper III: S. Holl, **M. S. Memon**, B. Schuller, M. Riedel, Y. Mohammed, M. Palmblad, and A. Grimshaw. Enhanced Resource Management enabling Standard Parameter Sweep Jobs for Scientific Applications, *International Conference on Parallel Processing – The 42nd Annual Conference ICPP 2013 Ninth International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems (SRMPDS)*, Lyon, France, 1 October 2013, IEEE, pp. 783–790 (2013). DOI: 10.1109/ICPP.2013.92

Paper IV: **M. S. Memon**, S. Holl, M. Riedel, B. Schuller, and A. Grimshaw. Enhancing the Performance of Scientific Workflow Execution in e-Science Environments by harnessing the Standards based Parameter Sweep Model, *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (XSEDE'13)*, San Diego, California, USA, July 22–25 2013, ACM Press, pp. 56:1–56:7 (2013). DOI: 10.1145/2484762.2484820

Paper V: **M. S. Memon**, G. Cavallaro, M. Riedel, and H. Neukirchen. Facilitating Efficient Data Analysis of Remotely Sensed Images using Standards-based Parameter Sweep Models, in *Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2017),* Fort Worth, Texas, USA, 23 July–28 July 2017, IEEE, pp. 3680–3683 (2017). DOI: 10.1109/IGARSS.2017.8127797

Additional Papers

Paper VI: **M. S. Memon**, D. Vallot, T. Zwinger, J. Åström, H. Neukirchen, M. Riedel and M. Book. Scientific Workflows Applied to the Coupling of a Continuum (Elmer v6.3) and a Discrete Element (HiDEM v1.0) Ice Dynamic Model. *Journal of Geoscientific Model Development*, Vol No 12, 3001–3015 (2019). DOI: 10.5194/gmd-12-3001-2019

Paper VII: **M. S. Memon**, G. Cavallaro, B. Hagemeier, M. Riedel and H. Neukirchen. Automated Analysis of Remotely Sensed Images using the UNICORE Workflow Management System. *In Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2018)*, Valencia, Spain, 22–27 July 2018, IEEE, pp. 1128–1131 (2018). DOI: 10.1109/IGARSS.2018.8519364

Paper VIII: **M. S. Memon**, M. Riedel, H. Neukirchen, M. Book, A. Grimshaw, D. Dougherty, P. Kaschuk, M. István, and Á. Hajnal. Enabling scientific workflow and gateways using the standards-based XSEDE architecture. *2017 International Conference on Information and Communication Technologies (ICICT)*, Karachi, Pakistan, 30–31 December 2017, IEEE, pp. 97–105 (2017). DOI: 10.1109/ICICT.2017.8320171

Paper IX: T. K. Samuel, S. Wan, P. V. Coveney, M. Riedel, **M. S. Memon**, S. Gesing, and N. Wilkins-Diehr. Overview of XSEDE-PRACE collaborative projects in 2014. *In Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure (XSEDE '15)*, St. Louis, Missouri, USA, 26–30 July 2015, ACM, pp. 24:1–24:8 (2015), DOI: 10.1145/2792745.2792769

Paper X: **M. S. Memon**, N. Attig, G. Gorbet, L. Gunathilake, M. Riedel, T. Lippert, S. Marru, A. Grimshaw, F. Janetzko, B. Demeler, R. Singh, and M. Riedel. Improvements of the UltraScan scientific gateway to enable computational jobs on large-scale and open-standards based cyberinfrastructures. *In Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery* (XSEDE '13), San Diego, California, USA, 22–25 July 2013, ACM, pp. 39:1–39:7 (2013), DOI: 10.1145/2484762.2484800

Acknowledgements

I owe my supervisor Morris Riedel a debt of gratitude during the whole process of doctoral research and guidance. His continuous encouragement inspired me to diligently pursue my doctoral studies. In addition to being my supervisor, he went above and beyond to support me in the technical and formal process of carrying out my research work at the University of Iceland and Forschungszentrum Jülich GmbH, Germany. It also gives me great pleasure to thank my co-supervisor Helmut Neukirchen, who guided me through the whole doctoral process. I am particularly indebted to his analytical skills and thorough knowledge of distributed systems which helped me to undertake my PhD degree. I am also grateful to him for providing me with the opportunity to take part in the eSTICC project, which enabled me to explore the area of scientific workflows and their applications in detail. This acknowledgement would be incomplete without mentioning my co-supervisor Matthias Book, to whom I am grateful for his valuable input and constructive ideas in streamlining the research work and doctoral dissertation. My sincere thanks are also due to Kristján Jónasson for his tremendous support in arranging the office location and logistics during my stay at the University of Iceland. I am also grateful to the Faculty of Industrial Engineering, Mechanical Engineering and Computer Science for supporting my research and providing a platform to collaborate with their brilliant team of academics from diversified disciplines.

Sincere thanks are due to the Federated Systems and Data Division of Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich GmbH for financially supporting my doctoral studies. At JSC, I am particularly indebted to Thomas Lippert, Norbert Attig and Daniel Mallmann for helping me with the resources needed for my doctoral research. I would also like to express my gratitude to Gabriele Cavallaro and Ernir Erlingsson for interesting and useful discussions. I am also grateful to the Open Grid Forum for providing me with a medium for taking part in its standard working groups' sessions.

I would have not sustained my studies and research successfully without the generous support of my parents and family. My father and mother have always been a source of inspiration and have always been there for me motivating my pursuit of my professional and personal goals. Their encouragement and advice helped me to remain persistent and steadfast in completing this arduous but exciting and memorable journey. I am very grateful to my wife for her patience during my long working hours, and also for taking good care of our children in my absence. Now, I have to mention the most important part, my children, Mishaal and Alyaan, to whom I am very grateful for bringing endless happiness and joy to my life.

1 Introduction

1.1 Motivation

In the present business and industrial environment, there is an increasing dependence on data and related services. Therefore, data-oriented services play an important role in devising meaningful models and results. For example, social networks or sophisticated search engines are all based on data-intensive analysis. To this end, scientific initiatives are progressing towards the exploitation of data for discovering and extracting useful information. A few large experiments can be mentioned as examples: the *Sloan Digital Sky Survey* (SDSS) – central data for spectroscopic survey [37]; *In-service Aircraft for a Global Observing System* (IAGOS) – a data infrastructure for climate change and air quality [9]; and PANGAEA, one of the largest data collections for Earth sciences [44]. With such a flood of data from scientific experiments which are potential data sources, it is very challenging to assimilate and interpret meaningful insights. This emerging trend is gradually raising the bar for service and technological ecosystems enabling the processing of data on distributed computing infrastructures. It is therefore critical to offer a working functional basis for unified, user-friendly, portable and robust data analysis and management platforms or interfaces that manage the processing of large data sets on disparate and distributed computing clusters.

As the computing clusters supporting the data-intensive scientific applications often consist of *High-Performance Computing* (HPC) platforms and services, it is imperative for the data analysis layer to consider the HPC environment as the target platform. There are several publicly funded initiatives that have led to the development of diversified HPC infrastructures, for example, PRACE [106], DEISA [56] and XSEDE [105]. These projects aim to provide supercomputing resources to multiple, more versatile and science-agnostic user communities. Considering the heterogeneity of HPC infrastructural capabilities, the serial and manual analysis becomes unmanageable because it requires automated and unified mechanisms to scale with the increasing magnitude of data sets and resource requirements. Managing multiple computations and their input and output data poses the following challenges. Firstly, there is a lack of convenience in accessing the complex combination of geographically dispersed data, computing services, and reliable data processing tools, which is considered to be a hindrance to scientists. Secondly, the wide variety of parallel and distributed systems, such as computing clusters with the emerging hybrid architectures, and the techniques and infrastructures available to scientists require more intuitive forms of scientific analysis to perform remote processing of data (e.g., entire data sets) which contrasts with traditional sampling. Last but not least, parallel and scalable implementations supporting machine learning applications are not advanced enough to allow users to undertake the

semi-automatic tuning and identification of parameters.

To lower the barrier between scientists and HPC-oriented data analysis, community or application-oriented science gateways are being developed which relieve users of the system's technical intricacies and enable them to focus on science through Web- and desktop-based interfaces for accessing HPC and *High-Throughput Computing* (HTC) platforms. For this reason, many science gateways have emerged in the past decade to provide scientific communities with usable, interactive, reproducible and domain-specific portals. A few examples are, UltraScan [43], Chem Compute [92], CyberGIS [107] and CIPRES [81]. One major shortcoming in the existing scientific gateways is that they cannot be used with different resource management systems or data management services. By chance, if some can be used in this way, then they are statically confined to specific target computing services by separate technology specific connectors. In this approach, one gateway must maintain a separate client implementation of each job or type of data management service. As a consequence, it becomes more impractical to maintain many connectors (so-called adaptors) in a longer run. To provide these gateways with unified access for consistent user experience in heterogeneous HPC environments, it is indispensable to have a common layer of abstraction that exposes a generic set of job and data management models and interfaces. This layer should not only be transparent enough to expose real computing resource and data management capabilities but should also cater for user requirements coming through a variety of user interface methods, science gateways, desktop clients or client APIs. This dissertation defines an architectural model that identifies, combines and extends multiple standards with the goal of forming a common mechanism for encouraging multi-domain scientific application users to manage their computations on HPC infrastructures. This addresses not only the above mentioned issues but also offers extensibility of future capabilities (e.g., access to other standards-based infrastructures). In this dissertation, a science gateway use case is employed as a motivation for in-depth analysis by eliciting key requirements and then offering useful realisations.

Learning models represent the wide area of data mining and machine learning algorithms. These models serve as the basis for this dissertation to automate the complex workflow and life cycle of the data analysis. Several challenges are encountered in data analysis phases. One of them is to execute machine learning algorithms multiple times but with a different set of application parameters. In terms of implementation, for the job submission, each combination of the parameter instance and the algorithm needs a separate job request to the batch system. In addition to the generated multiple job submission requests steered by the parameters, there is also an overhead once they are successfully submitted. That is the management and monitoring of the running, parameter-generated jobs. In this process, after all the computations have been completed, the next phase is essentially data aggregation, in which the job results of all the parametric computations have to be structured and gathered for further analysis. Evidently, this becomes non-trivial if all the steps of parameter sweep are performed manually as there is no generic mechanism available that automates the iterative compute and data-oriented executions in HPC environments. To support the parametric scenarios abstractly, in this dissertation open and standards-based models are selected and implemented which ensure the management of parametric, parallel and data-intensive computations and their large data sets in a robust, intuitive and automated manner.

In order to enable and automate scientific models through scientific gateways and parametric interfaces, this dissertation analyses the three following case studies belonging to two broad scientific areas of data analysis: domain-specific simulation science models and learning models. Domain specific simulations refer to the generation and interpretation of models based on established, well-defined theories and algorithms, i.e. using numerical methods with known physical laws. On the other hand, general learning models are produced through algorithms devised in the field of statistical data mining, machine learning and computer vision. These models are adopted in many domain-specific applications and include in many cases some form of optimisation method. The challenges and problems described above and the briefly presented methods for solutions are motivated by the following three scientific use cases:

1. Facilitate the processing of hydrodynamic data from ultracentrifugation experiments by using the specialised UltraScan science gateway. This science gateway uses the parallel UltraScan application for analysing high sedimentation velocity data through its HPC modules, specifically, 2-dimensional spectral analysis, genetic algorithms, and Monte-Carlo analysis.
2. Perform semi-automated remote sensing data analysis including different preprocessing methods such as *Principal Component Analysis* (PCA) using various *Support Vector Machine* (SVM) kernel parameters in HPC settings.
3. Enable real-time semi-automatic outlier detection methods using different algorithms such as *Density-based spatial clustering of applications with noise* (DBSCAN) with parameters (e.g. algorithm grid search or execution environment configuration, etc.) using a distributed computing architecture.

1.2 Thesis Objectives

The main *Thesis Objective*s (TOs) of this research project are to enable scientific communities to lower barriers for complex and data-intensive scientific workflows on distributed HPC infrastructures through a unique set of interfaces and their respective implementations. The TOs are achieved through the use of open and widely known computing and data standards, and their implementations by using state-of-the-art HPC middleware platforms. The use of HPC middleware is essential here as it defines a layer of abstraction that provide users with a unified interface to run parallel computations by encapsulating the heterogeneity of the underlying scientific application, batch system, network, operating system, and hardware. One of the HPC middleware platforms is chosen as a basis for selected reference implementations of the thesis findings.

The TOs with associated papers are summarised below:

- **TO 1:** Elicit the requirements of two fundamentally different data analysis approaches, physical and learning models.
- **TO 2:** Compare and contrast requirements from the two different (learning and physical) models, and identify a set of primary functions required to support the envisioned architectural elements.

- **TO 3:** Develop an architectural design based on primary functional components, which form the building blocks of a unified solution to enable a variety of scientific applications with similar set of requirements.
- **TO 4:** Implement the software prototypes that realise the architectural design elements, thereby forming a technical basis for the scientific case studies to access multiple computing and data services.
- **TO 5:** Adopt the software prototypes in the form of distinct case studies, primarily aimed at enabling different scientific communities using distributed and heterogeneous, computing and data infrastructures.

Figure 1.1 shows the research methodology in *Business Process Model and Notation* (BPMN 2.0) [13], which outlines a clear research path taken to fulfil the highlighted thesis objectives of this doctoral dissertation. As a very first step, the research commenced with a survey of existing technologies and then proceeded to the use case analysis of two major learning approaches on distributed HPC infrastructures. The next phase compared and contrasted the requirements, which resulted in a functional analysis. This functional analysis was further taken as a basis for developing a core architectural design. The work done in the architectural design has been contributed to several standard documents as cross-interoperability excursions and experience documents. Following the architecture, the implementation phase included the scientific gateway, support for parametric applications and integration of HPC-based job execution services with HTC-based storage systems. Each of these aspects resulted in several middleware enhancements and client *Application Programming Interface*s (APIs). The final phase delivers the designed solution to the relevant scientific communities by providing them with prototypical and service deployments on production computing and data infrastructures.

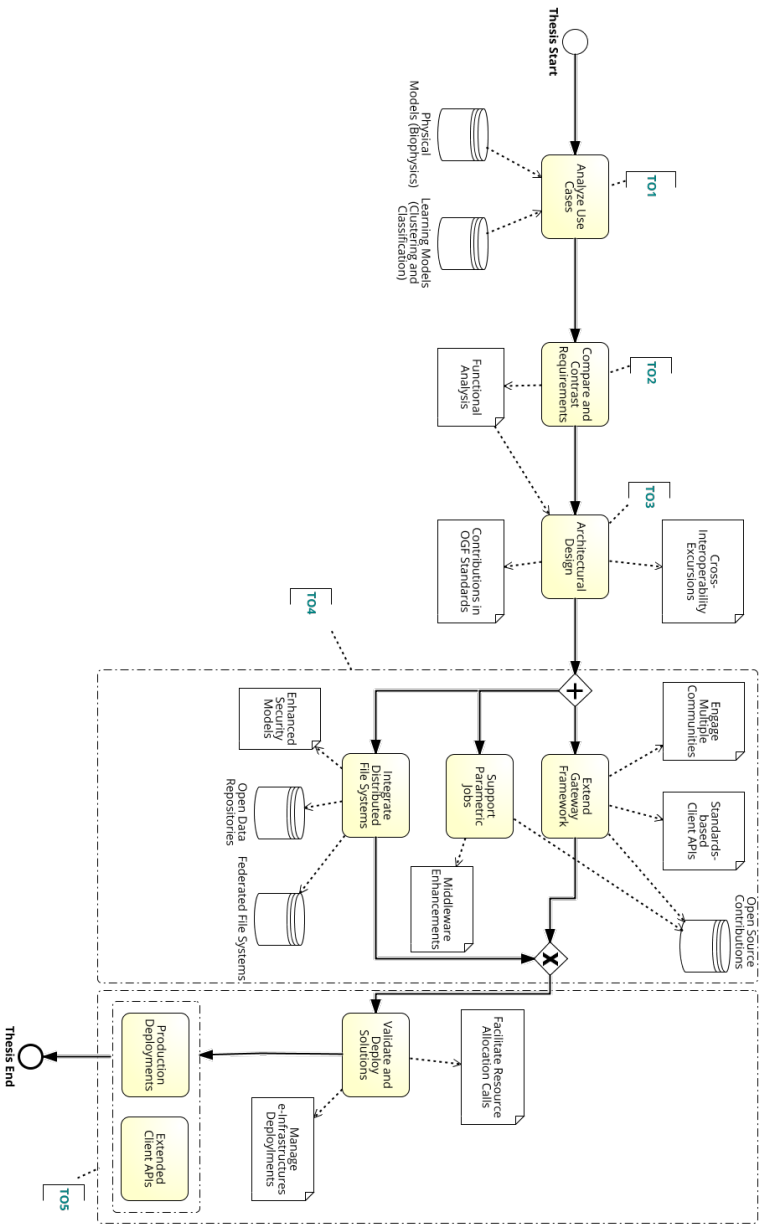


Figure 1.1. A BPMN 2.0 depiction of the research methodological process along the lines of the thesis objectives.

1.3 Contributions

The main contributions of this doctoral dissertation are divided into four areas. They are i) analysis of scientific case studies from different learning models, ii) enhancing scientific gateways to access multi-resource and cross-infrastructural services through standards-based client APIs, iii) bridging computing and data services from two different (HPC and HTC) resource architectures, and iv) enabling and automating parallel, scalable and iterative machine learning models. The first contribution is the analysis of scientific case studies from physical and learning models, specifically referring to Earth sciences and biophysics. In this contribution, the established scientific communities are involved, from whom the application requirements were initially gathered and then followed by the architecture design. This takes the *TO1, TO2* and *TO3* objectives into account and offers the associated implementation. Along these lines, the second contribution focuses on scientific gateways. Its realisation commenced with the concrete design, which is based on open source gateway middleware, and then provides the corresponding implementation in the form of generic and domain-agnostic client *Application Programming Interface*s (APIs). Since the client APIs are based on standards-based models and can be used generically, they can easily be integrated with multiple scientific portals and gateway frameworks. Enabling scientific gateways caters for *TO3, TO4* and *TO5*. The third contribution fulfils *TO3, TO4* and *TO5* by providing an integration base-line for integrating cross-architectural HTC and HPC case studies through a shared, federated and distributed file system. That in turn is a unique contribution as it supports the realisation of complex scientific workflows bridging two different (HTC and HPC) infrastructure paradigms simultaneously. This effort also partly contributes to the area of security, which is cross-cutting with respect to the architectural layer that facilitates users to access distributed computing resources while interacting with remote jobs and data and the services managing them. It particularly develops the provision of multi-identity authorisation access models realising complicated scenarios that allow end users who hold multiple secure identities and perform inter-organisational data analysis. The fourth contribution enables scalable machine learning models of parametric nature by leveraging the open and standards-based parameter sweep models in HPC environments. In this contribution two parallel machine learning algorithms are targeted: *Density-based spatial clustering of applications with noise* (DBSCAN) and *Support Vector Machine*s (SVMs). As most of the machine learning models require iterative computations with varying parameters at some point in the data analysis, this contribution is very useful in effectively managing long running parallel and parametric computations. This contribution facilitates the implementation of *TO4* and *TO5*.

1.4 Outline

This dissertation is compiled in a cumulative form with a collection of research papers that are included as an appendix. The main thesis contributions have been published in several peer-reviewed journals and conference proceedings. The following chapters provide a summary of all the case studies, their requirements, and corresponding realisations.

1.4.1 Structure

Chapter 1 - Introduction outlines the motivation for the research project and defines the overall thesis objectives.
Chapter 2 - Background provides a comprehensive but essential background on the data analysis approaches, case studies, supporting standards and distributed computing middleware technologies used.
Chapter 3 - Automated and Standards-based Data Processing in HPC derives an open standards-based, distributed computing architecture for supporting multidisciplinary data analysis applications on distributed and heterogeneous high performance computing resource environments. This chapter presents related work, a summary of requirements, with functional component analysis that summarises the contributions presented in the List of Publications.
Chapter 4 - Summary of Publications presents a summary of appended papers and shows that each of them is aligned with the respective thesis objective.
Chapter 5 - Conclusions ends the thesis and present prospects for the relevant research topics.

1.4.2 Relation of Publications and Thesis Objectives (TO)

As the dissertation primarily focuses on the automated processing of two scientific models, which are presented in detail in the publications shown in Table 1.1. *Paper I*, *Paper III*, *Paper IV* and *Paper V* focus on the learning models. *Paper I* also discusses a reference implementation to support HPC and HTC workflows through the integration of UNICORE and the *Global Federated File System* (GFFS), and thus contributes to the realisation of multi-tier identity authorisation scenarios.

Table 1.1. Association matrix of learning / physical models and scientific publications.

Model / Paper	Paper I	Paper II	Paper III	Paper IV	Paper V
Learning Model	X		X	X	X
Physical Model		X			

Paper III, *Paper IV* and *Paper V* present more insights into the adoption of standards-based parameter sweep models and show how they can be used to automate big data analysis scenarios based on massively parallel *Density-based spatial clustering of applications with noise* (DBSCAN) and *Support Vector Machine* (SVM) methods. *Paper II* highlights the remote processing of physical models. It also enables them to be realised through science gateway frameworks in combination with the platform-agnostic and standards-based client APIs.

The building blocks presented in this thesis are not only confined to the learning and physical models presented here, but they can also be applied to more generalised and domain-agnostic scientific models. Table 1.2 summarises the association of the thesis objectives and the publications.

The research carried out for this dissertation resulted in a contribution to several open source software libraries and scientific publications. Table 1.3 depicts the relationship

Table 1.2. Association matrix of thesis objectives and scientific publications.

TO / Paper	Paper I	Paper II	Paper III	Paper IV	Paper V
TO 1	X	X		X	X
TO 2		X	X		
TO 3		X		X	X
TO 4	X		X		X
TO 5	X	X		X	X

of the developed software artefacts and related scientific publications. The software contributions are most notably distributed as production versions and are actively supported by multiple organisations through open source collaborative platforms, such as SourceForge[1] and Apache Software Foundation[2].

Table 1.3. Association matrix of software contributions and scientific publications.

Software / Paper	Paper I	Paper II	Paper III	Paper IV	Paper V
UNICORE-BES	X	X	X	X	X
unicore-client-wrapper		X			X
JSDL-Sweep	X		X	X	X
Airavata-GFAC-BES		X			
UNICORE-GFFS	X				

[1] https://sourceforge.net
[2] http://apache.org

2 Background

A wide range of literature is available on the data analysis approaches involved, which includes a cohort of algorithmic variations and methods, as well as various data processing and management middleware technologies that enable those methods on distributed infrastructures. Therefore, it is not feasible to describe all of them. For the sake of brevity, this chapter only highlights the relevant methods, computing and data standards, and middleware technologies that primarily support the motivation behind the case studies analysed in this dissertation. This chapter begins with the data analysis approaches and later examines the case studies where these approaches are applied. Further, it also describes how these approaches are realised through the adoption of standards and middleware technologies.

2.1 Data Analysis Approaches: Physical and Learning Models

Data analysis provides an overall methodology and platform using a myriad of tools and technologies [66, 64] for processing large and complex data sets. This methodology applies to the complete life cycle including different processing phases to derive meaningful information to support useful scientific conclusions. This chapter describes potential use cases from two different areas of modelling – physical and learning models. Specifically, it explores the algorithms and tools from the domains of biophysics and Earth sciences.

2.1.1 Analysis of Physical Models

Computer simulations are based on physical laws and are produced by running physical models based on parallel programming models often employing numerical methods. Pragmatically, these simulations are depend on data from various experiments and observations. There are many examples, but to name a few: molecular modelling, nuclear incident modelling, car crash simulation, human brain modelling and climate change patterns.

To support applications and complex simulations on distributed computing infrastructures, scientific computing provides two system architectures, *High-Performance Computing* (HPC) and *High-Throughput Computing* (HTC). HPC uses a cluster-based architecture in which computing cores are tightly integrated through fast network interconnects. It is suitable for scientific applications where performance is a critical factor. On the other hand, HTC architecture [83] consists of computing cores that are coupled

loosely, including low-cost commodity clusters connected with non-high-performance network interconnects. It has proved to be more suitable for applications with less demanding run-time performance.

Several programming models are available to harness the capabilities of the above-mentioned processing infrastructures. Prominent examples are, *Message Passing Interface* (MPI) [80], *Open Multi-Processing* (OpenMP) [89] and OpenACC [28]. They enable computing processors to work in a cooperative manner using different programming models which are divided into two methods: *Single Program Multiple Data* (SPMD) [38] refers to data parallelism whereby multiple processors work on different parts of the data; whereas *Multiple Program Multiple Data* (MPMD) [29] allows each processor to perform execution using a different code and data. Applications developed on these programming models are placed locally at the resource or site level. This implies that if a scientist intends to access and run applications remotely, a layer of software and interfaces is required that allows the management and monitoring of the computations in-progress and their data.

2.1.2 Data Intelligence: Data Mining and Machine Learning

Statistical data mining is an area which encompasses a wide spectrum of algorithms and methods to find interesting patterns from data. In some cases, this field is interchangeably referred to as machine learning, although the two fields have minor distinctions in terms of overall objective. The term machine learning is defined by Tom Mitchell as [82], *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*

In broad terms, machine learning problems are categorised into three groups,

- Supervised learning: In this method, a given data set already has output assigned, so that the input and output association is learned in a straightforward way. Supervised learning deals with two kind of problems: regression and classification. Regression problems involve predicting through continuous function, which means that the output is a real number. Classification problems discover results from a discrete function. There are several classification methods that enable supervised learning problems to deal with, such as, decision trees [95], SVM [33], k-Nearest Neighbour (kNN) [94] and Neural Networks [74].

- Unsupervised learning: Unsupervised learning tackle problems without labelled response, which means that an input and output relation has to be created. This contrasts with supervised learning problems where we have a data set that is already labelled and which to some extent guides the process of data understanding. Unsupervised learning problems use several clustering approaches to form the association of data attributes. Several unsupervised learning algorithms have been developed in the last few decades. For instance, K-means [69], hierarchical clustering [65] and DBSCAN [47].

- Reinforcement learning: Reinforcement learning tackles problems by using a graded response for each given input in a particular learning context. It is often

applied in game theory and is not a major part of this thesis. It is listed here for the sake for completeness.

2.2 Case Studies

2.2.1 UltraScan: Analysis of Ultracentrifugation (AUC) Experiments

UltraScan [6, 41] is a data analysis application suite which processes the data generated from high-resolution analytical ultracentrifugation (AUC) [32] experiments using sedimentation velocity centrifugation methods. This application provides useful insights in order to understand biological macromolecules and synthetic polymers, which help in investigating complex relationships among biological systems such as different kind of tumors. Scientists can perform noise analysis by using 2-dimensional spectral analysis (2-DSA) [23], genetic algorithm optimisation [24] and Monte Carlo analysis methods [42]. Most of the functionality offered by this application is accessible through its Web-based interface called the UltraScan science gateway (US3-LIMS) [6]. As US3-LIMS only supports a limited type of resource interfaces, it cannot be seamlessly invoked on different computing sites with various resource management systems. To address this concern, the US3-LIMS gateway was enhanced as part of this doctoral thesis to perform cross-organisational, seamless and resource-management-agnostic computations through standards-based interfaces and models.

2.2.2 Outlier Detection on Point Cloud Data

The identification of outliers on a 3-D point cloud data set motivated the research carried out in this thesis. Specifically, the point cloud data [71] of the city of Bremen, Germany, is taken as a use case. The underlying outlier analysis is performed using the DBSCAN [47] algorithm which groups densely populated data points that are in proximity to each other. The algorithm finds the number of clusters automatically. Of these the low-density clusters are grouped as noise, and they are then reserved for application-specific analysis.

The outlier analysis method used in this dissertation is realised through HPDBSCAN [59], a parallel HPC DBSCAN-based implementation. It uses the combination of MPI and OpenMP (hybrid approach) to attain parallel processing. This thesis takes the implementation described in [59] as a basis for performing standards-based remote outlier analysis. Further details are elaborated in *Paper I*.

2.2.3 Classification of Remotely Sensed Images

The *Support Vector Machine* (SVM) [33] is one of the most intuitive machine learning methods available today. It belongs to the class of algorithms applicable for supervised learning, i.e., for classification and regression analysis. The main idea of SVMs is to form a maximum margin boundary which separates training samples from different classes. Initially, SVMs are used for linear decision boundaries, that is to say binary classification, which is not useful for most real scientific case studies as they generally

have any number of classes. This situation led to the development of kernel functions [98]. One drawback of SVM is the algorithm's sensitivity to the choice of the kernel function and its cost parameters.

piSVM [25] is a libSVM-based [30] parallel implementation of SVMs, particularly supported for HPC architectures. The first version was published at pisvm.sourceforge.net. Due to its shortcomings with respect to performance and storage access, this thesis preferred to use a more enhanced implementation presented in [27], which efficiently utilises parallel compute resources, and handles large data sets by the state-of-the-art parallel I/O methods. This implementation is based on MPI and *Hierarchical Data Format* (HDF5) [2] in optimising multiple data analysis life cycle phases.

2.3 Standards

2.3.1 Job Request Model: Job Submission and Description Language (JSDL)

Job Submission and Description Language (JSDL) [8] is a comprehensive specification published by the *Open Grid Forum* (OGF) [4]. It comes with a rich data model as XMLSchema [22], covering a wide spectrum of concepts that are used to express application, compute and data management requirements. The model structure consists of four major elements, job identification, application, resources and data staging. Each of these elements further contains hierarchically more data structures.

The JSDL specification mentioned above represents a core set of information, but still lacks integration with the emerging requirements of science. Therefore it offers a number of profiles. The profile embodies auxiliary but useful data models that include additional structure and constraints based on JSDL specifying, for instance, different data access protocols or parametric job requirements in a single job request instance.

This dissertation has significantly contributed to three out of the many available JSDL profiles by providing input from the selected use cases. The profiles are briefly introduced here as follows:

- HPC FileStaging Profile [109]: This specification models extensions for representing HPC-oriented concepts for data transfer and access. For instance, it lets the client pass credential information while submitting job requests. The specification is useful for constructing job requests with different data transfer protocols (e.g., *File Transfer Protocol* (FTP), *Secure Copy Protocol* (SCP)) and their corresponding required security credentials.

- JSDL-SPMD (Single-Program-Multiple-Data) [97]: This specification is an extension based on JSDL for expressing job requests with the parallel application and run-time environment elements as part of the JSDL instance. It contains a variety of concepts used while running MPI- or OpenMP-based jobs. For instance, it gives clients the option of specifying what kind of parallel run-time environment (e.g., OpenMPI or MPICH) and what number of processes are required. The JSDL-SPMD extension overrides the application element of the JSDL specification. Therefore all the extension elements are placed inside this structure.

- *JSDL-Parameter Sweep* (JSDL-PS) [45]: The JSDL-PS extension captures the job submission requests of a parametric nature. Many scientific applications require such functionality to manage many jobs intuitively as part of one simulation. This specification allows parameter sweeps to be formulated within the main job request in which a set of parameters or parametric constraints are specified. The sweep model defines an XML instance that includes which parts of the job request are parametric and which are not. This is defined through the X-PATH [85] query language. The JSDL-PS specification supports two fundamental types of sweeps: document and file sweep. Document sweep is particularly suitable for the parametrisation of application arguments and environment variables enclosed in the job request. File sweep provides a more invasive approach that parametrises the content of the input files used by the submitted job.

2.3.2 Job Management and Monitoring Interfaces: OGSA-Basic Execution Service

In order to provide several scientific disciplines with a convenient and unified computing experience, common, convenient and unified interfaces and models are required. To serve this purpose, the job management and monitoring standards deliver interoperable access to different distributed computing middleware clients and services. The *OGSA-Basic Execution Service* (OGSA-BES) [51] specification is one of the emerging job submission and management standards considered in this dissertation. It provides a web services-based interfaces to manage and monitor distributed and remote computations. OGSA-BES stands on a set of SOAP [35]-based specifications, such as *Web Services Interoperability* (WS-I) [5], *Web Services Addressing* (WS-A) [36] and *Web Service Resource Framework* (WSRF) [18]. It includes separate interfaces for job submission, management, and monitoring concerns. The service interfaces are briefly described below:

- BESFactory: This provides an interface to create and monitor sets of activities, and also allows capabilities of an HPC resource management system to be exported.
- BESActivity: This interface logically represents a single job instance. In this manner, it exports a set of functions to control individual jobs, such as abort, pause, and resume.
- BESManagement: This is an administrative interface for managing BESFactory endpoints, which instructs the BESFactory interface whether to accept or reject any new job submission requests from the user.

Apart from the interfaces, the OGSA-BES specification contains the data models that significantly represent the complex concepts and associations of HPC architectures. This includes the computing site's deployed resource management system, storage capabilities and managed jobs. In addition to this information, the middleware service information can also be represented. The specification further provides a state model for capturing the complete job life cycle. It is similar to a state machine. The information

model is part of the BESFactory interfaces to represent back-end resource management attributes such as the total jobs being managed, the total number of cores, and number of nodes, etc.

2.3.3 Resource Capabilities: GLUE2 Information Model

The GLUE specification [15] models compute and store entities in the natural language representation. It is an abstract model with theoretical representation. However, it has several machine-readable renderings in XML Schema [16], *Lightweight Directory Access Protocol* (LDAP) Schema [14] and *JavaScript Object Notation* (JSON) [103]. In this dissertation, the XML Schema rendering is used as an extension to the BESFactory attributes. It is used to accommodate more sophisticated types of hardware and software resource capabilities. A typical scenario is a logical conceptualisation of a computing cluster that offers an OGSA-BES-compliant service instance, application and runtime environment capabilities also including storage and hardware configurations.

2.4 UNICORE: HPC Grid Middleware

HPC middleware is a software abstraction that encapsulates the complexity of underlying layers of a batch system, data distribution, and file management capabilities. According to [34] the middleware is defined as:

"The term middleware applies to a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems and programming languages."

UNICORE is distributed computing middleware which provides a layer of encapsulation over multiple kinds of batch systems, platform-agnostic deployment, data management, and transfer mechanisms. It supports a variety of use cases which are most common while performing distributed job submissions on HPC resources. It works not only for single resources, but can be used for a set of computing clusters, grouped in an infrastructure. Examples of such computing infrastructures are XSEDE [105] and PRACE [106].

UNICORE is based on a client-server architectural pattern [34] and follows a *Service Oriented Architecture* (SOA) style of software design. In this respect, it provides all of the major functionalities, such as job management, data transfer, storage management, resource brokering, and workflow execution and enactment in the form of web services. In a service-oriented design, the client-side applications can be independent of server-side implementations or services. Therefore, the clients that are based on other middleware stacks, though not directly developed for UNICORE, can be written to interact with UNICORE services, for example by following the client-proxy design pattern [55].

Figure 2.2 depicts the UNICORE architecture from a broader perspective with two major layers of client and server-side components. On the client side, it provides a thick client called *UNICORE Rich Client* (URC), with the capability to manage complex scenarios. It is developed with the Eclipse *Rich Client Platform* (RCP) [73] framework

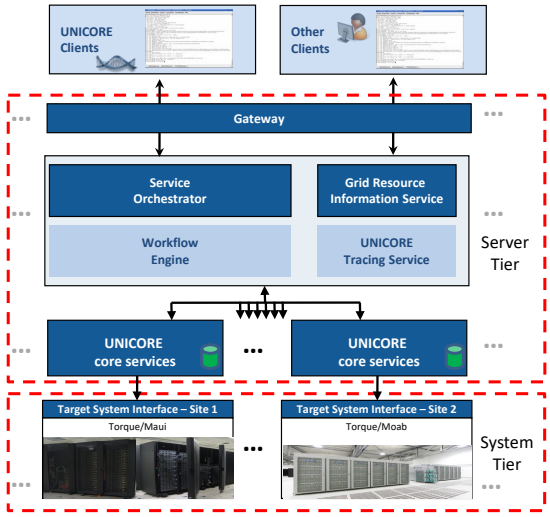


Figure 2.2. An architectural view of the UNICORE middleware

and comes with an interface for users ranging from novice to advance system expertise through which they can construct and edit complex scientific workflows. This not only allows a user to interact with different computing sites but also provides a rich interface to visually set up workflows through a canvas and interactively submit and monitor remote jobs. The *UNICORE Command-line Client* (UCC) is a command-line interface for more advanced users willing to interact with a server tier.

When a user interacts with a UNICORE instance, the very first component that receives the request is the UNICORE Gateway. This ensures a mutual client-side authentication based on the SSL protocol [53]. The components beneath are responsible for performing authorisation and managing jobs and data. Immediately below the server tier, there is the lower level resource management layer called the system tier on which the *Target System Interface* (TSI) resides. The TSI allows incoming user requests through the job management service to be translated into the target batch system specific commands. In this manner, the TSI layer provides the request and response translation of several batch systems, e.g., SLURM, LoadLeveler, and *Load Sharing Facility* (LSF).

As far as the enabling of scientific applications is concerned, the server tier provides multiple applications in a declarative manner, i.e., through an extendable configuration database on which multiple scientific applications, run-time environments and their pre- and post-execution requirements are specified. UNICORE has a well-supported framework of user-facing components to enable science requirements. Nevertheless, it lacks standards-based interfaces to support science communities using their own client applications. In the research covered by this thesis, UNICORE is considered as a frame-of-reference for providing reference implementations of the abstract concepts addressed in this thesis. It is used for introducing standards so that other scientific clients can leverage the remote management of jobs and data for *High-Performance Computing* (HPC) deployments. Details on what standards are used, how they are

applied and integrated into UNICORE are given in the context of the corresponding research later in this thesis.

2.5 The Global Federated File System (GFFS)

The *Global Federated File System* (GFFS) [61] is a distributed file system to federate distributed remote compute and data resources in a secure and standardised manner. All the interfaces of the GFFS are based on an SOA design, thus exporting all the functionalities as a set of web services. In particular, the core of the service interfaces is built upon the OGF's *Resource Naming Service* (RNS) [84] standard.

As part of this dissertation, the GFFS is extended and integrated with the job execution service of UNICORE. This is to provide the HPC-based jobs with access to the GFFS-managed storage pools, because the GFFS was primarily designed to support HTC infrastructures. With this extension, cross-HPC and -HTC resources can be simultaneously used. Specifically, this is comprehensively discussed in *Paper I*, which covers the outlier analysis of data in the GFFS space running through HPC-oriented UNICORE job management and execution services.

3 Automated and Standards-based Data Processing in HPC

This chapter covers the main research contributions which are comprehensively described in the List of Publications. It begins with the work related to the research undertaken as part of this dissertation. The chapter then highlights significant requirements and subsequently explores how these requirements are realised through the functional component analysis. The identified functional components are the main building blocks, each forming a group of similar functions which are discovered during the design and implementation phase of the tools supporting the case studies.

3.1 Related Work

Extensive research has been performed in the area of distributed computing and data infrastructures. However, little research has been published that provides a design integrating the elements of, application-centred scientific gateways, cross–infrastructural and distributed file systems, and automated and scalable parametric computations, combined in one coherent, standards-based architectural framework supporting different case studies. In view of this, the following section examines the initiatives which are closely related to the research work presented in this dissertation.

Science gateways have benefited a larger set of scientific communities than the community-specific portals which are limited to use by individual applications, server-side protocols or computing clusters. It is evident that the integration of scientific gateways is founded upon client APIs. This concept is also used in the architectural design of this thesis, while adhering to established software engineering methods. Hence, several client APIs have been developed to support different communities. Examples include the Vine toolkit [67, 96], which is a JAVA-based API that provides a client-side layer containing a separate sub-API for each middleware. Initially developed for Gridsphere [86] it was subsequently released as an independent library for other portals. It provides separate connectors for UNICORE and gLite, but in a proprietary format. Thus, they cannot be used by scientific applications that are standards-compliant. The *Simple API for Grid Applications* (SAGA) [58] is a state-of-the-art API specification developed by the *Open Grid Forum* (OGF). It supports multiple programming models (e.g. map-reduce, replication and parametric executions, very similar to the Apache Airavata framework. In this way, the *Simple API for Grid Applications* (SAGA) framework [58] and it's implementation bindings [79] in combination with the client API developed in this dissertation can offer more value for both Airavata and SAGA-based scientific gate-

ways and portals. The P-Grade portal [63] is yet another closely related development concerning scientific gateways. This portal bridges multiple middleware clients through different adaptors. It provides a separate component for standard interfaces. However, it comes with a limited support of standards, which may not be sufficient for enabling the scientific case studies presented in this dissertation which use multiple and distributed HPC infrastructures.

In this research project, the machine learning models use the DBSCAN [47] algorithm and its parallel implementation [91], automated through the combination of UNICORE-BES and the *Global Federated File System* (GFFS) thus enabling remote management of job executions in heterogeneous resource environments. Similar to this endeavour, GridFTP [72], one of the data transfer mechanisms used by many open source and commercial vendors, is integrated with UNICORE by following the same pattern of UNICORE and GFFS realisation. In contrast to the GFFS integration, the GridFTP extension comes with a separate security model that encapsulates user credentials in the job request model and also the extension of the native execution framework of UNICORE with a dedicated file transfer component which facilitates the actual data transfer and management. While submitting UNICORE jobs with GridFTP data staging a user has to include a chain of X.509 proxy certificates, as part of the message payload. In this approach, the complete credential hierarchy is exposed as plain text, although the message payload is encrypted, which still might not be intuitive and secure as administrators can see this information. On the other hand, this is not what the GFFS integration does. It transports an encrypted set of credentials as *Security Assertion Markup Language* (SAML) assertions – even in the client request payload, see *Paper I* for more details. With GridFTP, there is a web-based data transfer service known as GlobusOnline [49]. This mainly uses GridFTP for providing one-to-one and third-party data transfers. Comparing GlobusOnline and the *Global Federated File System* (GFFS), in terms of concept and functionality they share many functionalities. However, one major benefit of integrating UNICORE-BES with the GFFS is to let users access and manage, distributed and cross-site HPC (UNICORE) and HTC (GenesisII) workflows. Similar to the approach followed in this dissertation, ARC [46] integrates with the data management platforms, dCache [54] and *Distributed Data Management* (DDM) [20]. This integrated solution is oriented towards the *Large Hadron Collider* (LHC) particle physics community known as ATLAS [104]. Functionality-wise, the dCache and GFFS platforms offer the same kinds of services, but they are based on different sets of standards. The GFFS adopts RNS and ByteIO standards, whereas dCache is based on the *Storage Resource Management* (SRM) [10] interface. The integration of the GFFS with the WS-PGRADE / gUSE (grid and cloud user support environment) [63] science gateway framework is closely related to the research on which thesis is based. The WS-PGRADE / gUSE framework connects with the GFFS [7] and allows jobs to be submitted to UNICORE and ARC services through the OGSA-BES interface.

In addition to the enabling of standards-based job executions and data management, parametric computations are imperative for automating learning models used in different scientific domains. For that reason, the research work for this dissertation produced the JSDL-Sweep library, a lightweight, standards-based, technology and platform-agnostic API implementation of the JSDL-PS specification. While analysing the related work, there are a few initiatives that are closely related to the research presented in this thesis

in the context of enabling semi-automatic processing of jobs in HPC environments. gEclipse [57] is one such related research project that is based on Eclipse framework [73]. It provides individual client adaptors for several types of computing middleware, such as gLite, ARC and UNICORE. gEclipse is the first implementation of the JSDL-PS specification, and it also provides an automatic generation of JSDL-PS requests through Eclipse's *Graphical User Interface* (GUI). That means users are relieved of the complex XML representation of their parametric job requests. It is encouraging in terms of usability. However, gEclipse is tightly integrated with the underlying Eclipse libraries, which hinders third-party scientific clients, gateways or *Workflow Management System* (WMS) from adopting it in their environments. In the area of parametric executions, gLite's Workload Management System (WMS) is a meta-scheduler and comes with the provisioning of parametric computations. This implementation builds upon its own proprietary specification called *Job Description Language* (JDL). In contrast to the JSDL-PS specification, the JDL structure has two major short comings. Firstly, it is very specific to the gLite clients, thus other non-gLite clients cannot be integrated in a straightforward manner. Secondly, the JDL's structure of parametric requests only handles the sweeps of a single parameter within the scope of the whole job request. This contrasts with the approach followed in this doctoral dissertation which relies on the adoption of standards. The research led to the JSDL-Sweep implementation that is based on the rich structure of JSDL-PS for representing the parallel, multi-variant and complex hierarchy of parametric job request structures.

3.2 Requirement Analysis

This section summarises the requirements for realising the learning and physical models. The requirements were gathered through personal interviews, passively observing user interactions, and studying the characteristics of the applications with respect to compute and data access patterns. During this phase, the emphasis was placed on allowing users to focus on science rather than dealing with the complex system access intricacies, such as distributed batch system interaction, data management and security. The requirement elicitation and analysis presented in this section implements thesis objectives *TO1* and *TO2*.

R1-Remote job submission: This requirement comprises a scenario in which a user may submit jobs to remote execution service. This scenario should be performed through standards-based protocols because scientific applications often require a complex set of input and output parameters to be qualified for successful computation. In this respect, the standards-based structure plays an important role as it captures a variety of application and data representations. Most notably these include, application, compute and data resource requirements. To support this functionality, there should be a standard for the job submission interface and model that conceptualises the user's job execution request and response protocol. The standard should also provide a set of methods representing the overall job submission phase.

R2-Remote job management and monitoring: Job management and monitoring realise remote execution and monitoring scenarios. Job submission to the resource

management system only captures the scenario of channelling the user's job request to the batch system or queue while the job management and monitoring scenario envisions the interaction of users with the running remote job, such as pause, abort and resume, push-based data staging and status monitoring.

R3-Resource discovery and registration: Several computing sites are involved within a distributed computing infrastructure. Each of these computing sites can deploy a different variation of execution services. Resource discovery is essential for scientific clients and computing sites so that the existence and availability of desired execution sites should be known.

R4-Security: This is one of the fundamental requirements in order to support remote job and data management scenarios. It allows scientific clients to communicate securely with the target execution or data service. With this requirement, the target service should be capable of authenticating and authorising the identity of the client requesting the job submission. In the case of distributed file system integration, after passing through various layers of different services (e.g., identity provider or meta-scheduler), scientific clients present multiple identities to the job management service. To vet these complex requests, a secure validation and delegation model is required to parse and verify multi-assertion and nested authorisation tokens, known as assertions. Furthermore, integration with MyProxy [87] services should also be realised for supporting the computing resources deployed across XSEDE [105]-managed services.

R5-Enabling science gateways: Science gateways provide domain-specific user interfaces. Mostly they are available in the form of web portals. Science gateways accessing resources and running compute and data-intensive jobs require components to understand remote system interfaces. Therefore, the gateways should be extended to include client-side components that can communicate the target resources and execution interfaces on the users' behalf.

R6-Parametric executions: Several scientific applications and complex simulations require recurring executions with different parameters. In parametric computations, the basic run-time environment specification of the application remains constant while the application parameters are variable. The job management services should provide a data model and appropriate implementation to accommodate and monitor parametric job executions.

R7-Job provenance: Job submission through middleware may go through several tiers until the final output is produced. To diagnose any errors or inconsistencies during the job execution life cycle phases, typically pre-processing, processing and post-processing phases, adequate information is required for analysing any unexpected job execution failures. Therefore, a data model should be available that provides an ample provenance record on various job execution phases. This may lead to a data structure enabling scientific applications to project and search for interesting information events for debugging job failures.

R8-Federated job and data management: This requirement complements the elicited requirements by allowing scientific clients to manage multiple jobs and their data on distributed computing infrastructures. Running cross-infrastru-ctural computations sometimes requires accessing both HPC and HTC sites and has a tremendous potential for supporting a wide range of scientific case studies, specially for managing distributed scientific workflows consisting of many tasks connected to each other forming compu-

tational graphs. *Paper I* particularly covers the scenario in which data processing such as clustering using *Highly Parallel DBSCAN* (HPDBSCAN) as well as access via the GFFS is based on two different infrastructures.

Requirements R1-R8 presented above lead to the analysis of the functional components that provide a unified and standards-based framework for distributed and HPC-oriented scientific applications. These requirements are summarised in *Paper I, Paper II, Paper IV, and Paper V*.

3.3 Functional Component Analysis

This section describes the functional components which are motivated by the requirements presented above and the scientific case studies originating from physical and learning models. The design of the functional components ensures that the users accessing scientific applications should have seamless management of remote data processing and management interfaces. The analysis of functional components fulfils thesis objectives *TO3, TO4* and *TO5*. Furthermore, in combination these components form a generic architectural framework following open standards. Thus, interoperability among other standards-based tools is achieved. The development of the functional components also contributed the experiences and lessons learned for inclusion in the standard specifications. From the implementation perspective, this framework provides technology-agnostic abstractions for heterogeneous, data-intensive and distributed HPC infrastructures.

3.3.1 Remote Data Processing

Remote data processing refers to a set of scenarios capturing job management and monitoring in distributed computing environments.

As part of this research project, one objective was to facilitate resource-management-agnostic submission and monitoring of jobs on remote sites. In order to support multiple resource management systems, standards-based interfaces play a significant role in enabling multiple computing sites. Job submission and management is realised through the *OGSA-Basic Execution Service* (OGSA-BES) specification.

The OGSA-BES interfaces described in the first version of the specification lacked support for expressing adequate resource capabilities, for example nodes, cores-per-node, and network-type. In pursuit of a standards-based job management service exporting state-of-the-art batch systems, the GLUE2 specification model is incorporated as an extension to the OGSA-BES's BESFactory interface. The resulting BESFactory implementation is deployed as a server-side component to expose the computing capabilities of the target site. With the GLUE2 extensions, this would require client-side tools or APIs to adhere to the GLUE2 specification for communicating with the server-side endpoints. The related client tools and APIs have been enhanced with the provision and understanding of GLUE2 elements, which are then used while expressing service request and response payloads.

The first OGSA-BES implementation was realised as part of the thesis: UNICORE-

BES. The initial results on this implementation have been published by the author of this thesis in [76]. UNICORE middleware is used as one of the building blocks, as its web services environment provides intrinsic support for stateful services [50] through WSRF-based [18] standards. One reason for using a stateful services environment was that the OGSA-BES specification mainly focuses on use cases referring to the stateful nature, such as job execution services and managed job related information. To implement a stateful service is not a mandatory requirement, since it can also be developed through a stateless service framework. For instance [46, 68, 75] are built upon the established WS-I [5] stack of Web service interfaces.

3.3.2 Remote Data Transfer and Access

Data transfer and management is an essential element to support job execution and management systems. During job execution, data management is either accomplished through an automated middleware abstraction or by invoking interactive SSH sessions. In both cases the data should be made available a priori to the job execution phase and normally requires transfer to or from different kinds of data sources, such as remote data repositories [17] or specialised storage management systems [54, 61].

In this respect, different data transfer protocols such as SCP [112], UFTP [101], GridFTP [72], SFTP [70] and HTTP (S) [21] are extensively used in scientific computing. Specifically, GridFTP provides support for secure and parallel streams [12] and therefore appear to be more commonly used among scientific communities. As part of the present research work, this is considered to be one of the mainstream requirements to enable communities from multi-disciplinary backgrounds given that appropriate data models have been developed and integrated with standards-based interfaces. UNICORE-BES has been enhanced by the author of this thesis to support job submissions that access GridFTP-based data sources or sinks. The extension follows JSDL's HPC-FSP [109], HPC profile [97] and WSRF basic profile [52].

The implementation is deployed and used by multiple computing centres across the computing sites participating in the XSEDE project [105]. The UNICORE-BES implementation with GridFTP access has successfully undergone several interoperability tests in comparison with other similar implementations, for instance GenesisII [1]. These interoperability tests ensure that the implementations are compliant with the mandatory specification elements, particularly envisioning the combined use of JSDL and OGSA-BES specifications and their HPC related profiles. The realisation of this functional component is demonstrated in the form of an UltraScan gateway implementation that bridges UNICORE-BES and GridFTP in order to run computations on the Jülich Supercomputing Centre's JURECA [3] cluster accessing data from XSEDE deployed file systems.

Bridging HPC and HTC infrastructures is one of the critical aspects in executing cross-site scientific workflows. In this approach, the *Global Federated File System* (GFFS) is integrated with the UNICORE-BES implementation. The GFFS is a distributed file system that provides a set of components to connect many different data resources across geographical boundaries into one logical data space known as the GFFS storage pool. As the GFFS embraces open standards-based interfaces, ByteIO and *Resource Naming Service* (RNS), and is used by scientific applications running on

HTC infrastructures, it is an optimal option to promote scenarios in which HPC-oriented computing jobs analyse data placed on HTC-managed storage. With this integration of UNICORE-BES and the GFFS, the users running computations on HPC can import or export data from the GFFS storage space. The main idea was to enable the scientific communities to perform some part of their computations on HTC and then use the results produced as input to massively parallel computations on other distributed HPC sites. One good example is presented in *Paper I*, in which a use case of unsupervised learning (HPDBSCAN) is showcased. In this approach, the data-oriented workflow is realised through UNICORE-BES and GFFS integration. The first pre-processing job runs on an HTC site managed by the GenesisII execution interface, whereas the second task of data clustering runs on an HPC cluster managed through UNICORE-BES. For this workflow, one common, shared GFFS storage space is used, where all the input, intermediate and output data is stored.

3.3.3 Standard Interfaces and Data Models

As part of this research, standards played a significant role in bringing together multidisciplinary scientific communities, various tools and infrastructures. The standards mentioned in Section 2.3, directly and indirectly, contributed to the development and validation of those specifications. They are primarily the experience documents of ByteIO [31], the *High Performance Computing Basic Profile* (HPCBP) [108] and the *EMI-Execution Service* (EMI-ES) [99] for harnessing the capabilities of emerging HPC environments. Furthermore, this thesis made tangible contributions to the modelling of job provenance, tracking and check-pointing on distributed computing infrastructures, in the form of the Activity Instance Document [111] specification published by the OGF.

3.3.4 Parameter Sweep

Many complex simulations and data analysis tasks running on distributed infrastructures are often parametric. The Parametric jobs require the same application to be executed multiple times but with different input parameters, for example application arguments, environment variables, varying input file names and file contents. Providing such a model for distributed and parallel applications is a challenging task. To this end, the OGF has therefore published the *JSDL-Parameter Sweep* (JSDL-PS) [45] specification to model parameter sweep use cases.

As part of this thesis, a standalone, platform-independent and open source implementation of the JSDL-PS specification, known as the JSDL-Sweep library [45] has been developed. The JSDL-Sweep library implementation was inspired by an initial version produced by the GenesisII middleware team. To implement the thesis objectives, it was then refactored and improved with the aim of making it more abstract, reusable and generalised for different middleware technologies. As proof of integration, the UNICORE's XNJS (eXtended Network Job Superviser) [100] is considered as the first candidate to be integrated with the newly developed JSDL-Sweep library. The capabilities of XNJS are extended in two dimensions: i) for accepting and understanding the parametric job request with respect to the JSDL-PS compliance and ii) for managing

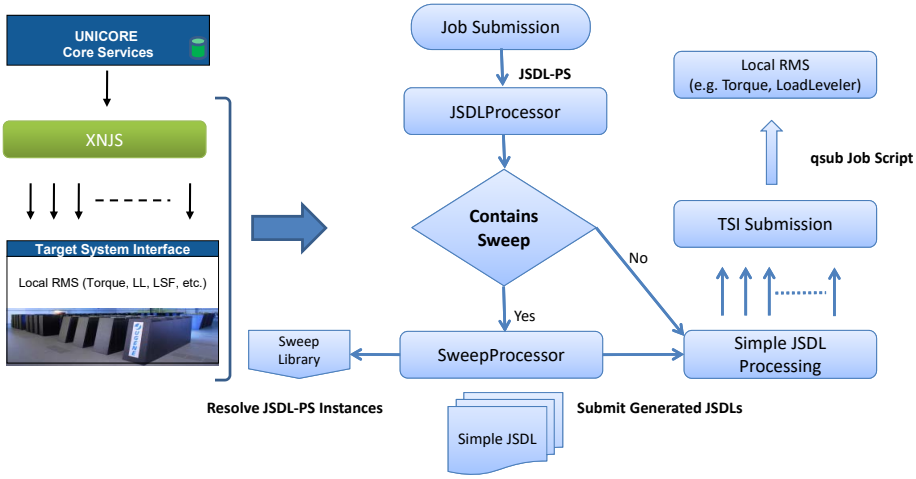


Figure 3.3. UNICORE integration with the independent JSDL-Sweep library and the workflow of processing parametric jobs in the UNICORE execution layer.

the execution of the individual resulting sweep job requests. Figure 3.3 presents an integrated architecture showing how JSDL-Sweep is connected to the UNICORE's middleware services. It also depicts how the sweep-based jobs are executed in the case of parametric or composite computations. The JSDL-Sweep library processes the request by first validating it and then generating multiple sweeps according to the user's parameter configuration. These sweeps are later transformed into the actual incarnation and execution of multiple normal JSDL-based job requests. Each JSDL request generated is considered as a sub-job and lies within the scope of a composite sweep job. During the execution of the sweep job, data redundancy is one of the issues while maintaining sub-job's working or session directories. This situation is eliminated by using the master-slave pattern. Full implementation details are given in *Paper III*.

Furthermore, UNICORE integration with the JSDL-Sweep library is demonstrated in the form of automated supervised and unsupervised learning models. The detailed use case analysis and results are published in, *Paper III*, *Paper IV* and *Paper V*.

3.3.5 Reusable and Secure Client APIs

User-facing job submission and management services can be more adaptable if they provide consistent and unified client-side APIs. Client APIs expose server side functionality to interact with job management and monitoring, data transfer and access interfaces through convenient abstractions. In this research project, the standards-based API known as the *unicore–client–wrapper* [77] was implemented to support gateway access and integration. *Paper II* deals with the API usage in the light of science gateways and web portals. It shows the benefits of the Airavata framework by using the *unicore–client–wrapper* API to support multiple heterogeneous resource infrastructures through standards-based models and interfaces.

3.3.6 Gateway Access and Integration

Many scientific communities running data-intensive workflows and computational simulations require a convenient and user-friendly interface that would dispense with the need to interact with low-level details of the resources (resource management system or file system) where these applications are deployed and running. Science gateways or science portals emerged primarily to support this objective. They express application-specific and usable front-end services that provide a rich web-based interface to manage job and data management tasks. Despite the convenient interfaces, there are still challenges with these gateways, specifically connecting end users with the resources providing computing and data capabilities. The R5-Enabling science gateways requirement represents this in more detail, for instance in a scenario where *n* scientific communities with corresponding *n* gateways access computing resources through *m* services, then *n* x *m* client adaptors will be required on the gateway side. This thus leads to a bottleneck for multi-disciplinary communities or users accessing multiple resources for one use case. To fulfil that requirement, gateway middleware frameworks are being developed to reduce the overhead of providing a generic bridge for heterogeneous science gateways in order to integrate multiple kinds of computing and data resources. Since the UltraScan gateway concerns the area of scientific gateways, the Apache Airavata framework [93] is used as the gateway middleware. The author of this thesis contributed to the Airavata framework's GFAC component, which is specifically a separate sub-component following the proxy-based [55] approach for communicating multiple HPC- and HTC-based execution management systems. The implemented extension integrates the client API with the support of standards-based (OGSA-BES and JSDL) job submission and management interfaces. Figure 3.4 depicts an architectural sketch of multiple kinds of job execution services, showing both the proprietary and the implemented extension based on standards. This architecture portrays how the standards play a bridging role while seamlessly integrating multiple job execution service, i.e. standards compliant deployments overcoming different organisational and technological boundaries. More comprehensive details are presented in *Paper II*.

As the scientific applications or workflows are mostly accessed through gateways and gateway frameworks, the Airavata's GFAC extension developed in this thesis can enable many gateway users to manage their remote jobs and data on a variety of HPC and HTC infrastructures. This extension is not only capable of supporting UltraScan gateway users but is also useful for other Airavata-enabled gateways, e.g. CIPRES [81] and Chemcompute [92].

3.3.7 Secure Access

Security plays an important role in protecting compute and data resources. Two major contributions have been published on the provision of secure access; *Paper I* and *Paper II* furnish more details. In this area, *Paper I* contributes to the conceptual design and implementation of authentication and authorisation in scenarios where the user or subject holds multiple identities, for instance, when one user has access to multiple infrastructures, i.e. a separate identity for each of them. For this security model, the UNICORE services framework has been extended to integrate with the GFFS server and

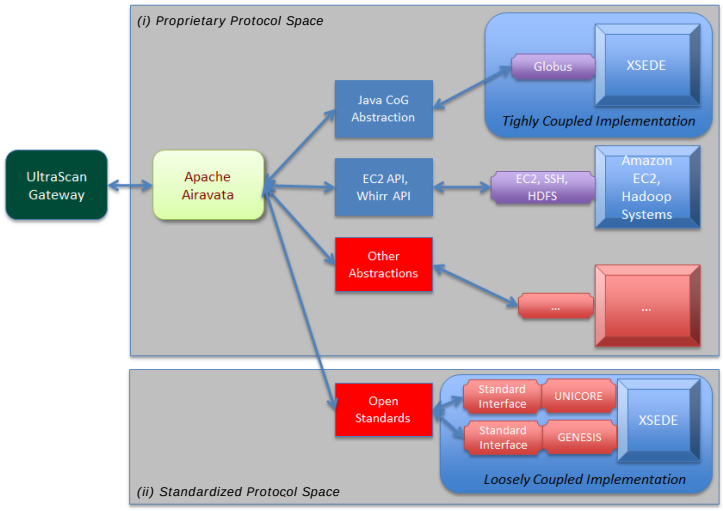


Figure 3.4. UltraScan Gateway access to multiple execution services through standards-based and proprietary interfaces. ©John Wiley & Sons, Inc.

client components. One result of this research has made contribution to the XSEDE's Architecture document (also called security profile) [48], purely based on the *Security Assertion Markup Language* (SAML) specification. The scenario presented in *Paper I* is considered to be a vital contribution to the middleware community as there is no initiative yet available that supports the standards-based authentication and authorisation of handling multi-hop user requests. The software artifacts thus produced are deployed on XSEDE's service provider computing centres. Further, *Paper II* contributes to the scientific gateways that extend a security model of the Airavata framework to integrate with the services offered by standards-compliant HPC and HTC middlewares. The overall architecture implements the scenario of a science gateway user submitting remote massively parallel jobs: the request includes several sequences of steps implementing the authentication and validation of the job submission request, and then forwards it to the target HPC job execution services.

4 Summary of Publications

This section summarises the papers that contributed to this doctoral dissertation. These papers provide more detail on the requirements and functional analysis of the scientific case studies outlined in the previous chapters.

4.1 Enabling Scalable Data Processing and Management through Standards-based Job Execution and the Global Federated File System

M. S. Memon, M. Riedel, A. S. Memon, C. Koeritz, A. Grimshaw, H. Neukirchen. Enabling Scalable Data Processing and Management through Standards-based Job Execution and the Global Federated File System, *Journal of Scalable Computing: Practice and Experience*, Vol 17 No 2, 115–128 (2016).

This publication partially serves the first thesis objective – TO1 – that involves the requirements' analysis of the learning models. In the fundamental design of the system architecture this paper also contributes to TO4 and TO5 i.e. software implementation and adoption of standards-based models.

This paper focuses on the elicitation of requirements in the course of securely integrating a distributed file system, termed the *Global Federated File System* (GFFS), with the standards-based job submission and management middleware, UNICORE, through the OGSA-BES standard known as UNICORE-BES. As part of the architectural implementation, a security model has been designed and implemented as an extended package in addition to the OGSA-BES interface for accessing large data sets placed on a GFFS-managed storage space. This scenario is motivated by a machine learning use case in which the computations need to be managed on both HPC and HTC resources, in a sequential manner. The paper presents a solution in which the standards-based interfaces of GFFS, GenesisII and UNICORE are integrated in a secure and seamless manner. The data set required for the computation is pre-processed on a GenesisII instance in an HTC manner, whereas processing occurs on a HPC site through UNICORE-BES. Figure 4.5 provides a simplistic view of the realisation by showing a single user running a learning model workflow using the GFFS client instance and performing computing on both HPC and HTC clusters.

The implementation produced by this research is successfully demonstrated by a use case from the Earth science domain. The scenario uses the 3D point cloud data

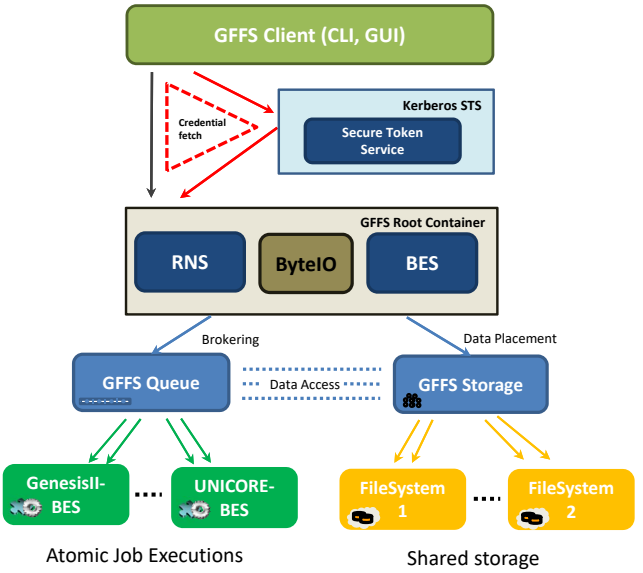


Figure 4.5. A holistic view of the integrated Global Federated File System *(GFFS) and UNICORE architecture. ©SCPE.*

set that is gathered in the form of a large number of points, which are a set of groups representing different object surfaces (i.e. the inner city of Bremen in Germany). The HPDBSCAN [59] application is used underneath to perform anomaly detection using the DBSCAN [47] algorithm in an HPC environment.

4.2 Advancements of the UltraScan Scientific Gateway for Open Standards-based Cyberinfrastructures

M. S. Memon, M. Riedel, F. Janetzko, B. Demeler, G. Gorbet, S. Marru, A. Grimshaw, L. Gunathilake, R. Singh, N. Attig, T. Lippert. Advancements of the UltraScan Scientific Gateway for Open Standards-based Cyberinfrastructures, *Journal of Concurrency and Computation*, Vol 26 Issue 13, 2280–2291 (2014)

This publication analyses the requirements of physical models and shows the corresponding software artefacts developed in the context of supporting scientific gateway frameworks. The research conducted for this publication was motivated by the UltraScan science gateway and fulfils the objectives: TO1, TO2, TO3 and TO5.

UltraScan is a state-of-the-art application package for processing data produced by *Analytical Ultracentrifugation* (AUC) experimentation. The UltraScan user community

employs a scientific gateway called the *UltraScan Laboratory Information Management System* (USLIMS) to run and manage complex simulations. USLIMS uses the Airavata framework [93] to enable computing and data management on different kind of resources in a proprietary manner. As part of the research, the Airavata framework was designed and implemented to access execution services and manage remote jobs and data transfers through standards-based protocols.

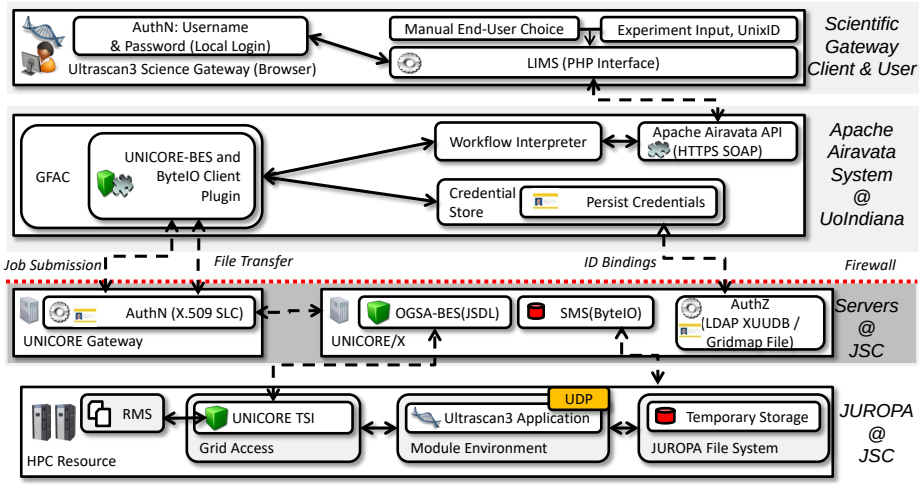


Figure 4.6. Architectural realisation of the UltraScan gateway, the Apache Airavata framework and its integration with the open standards-based job and data management remote services through client APIs. ©John Wiley & Sons, Inc.

The first part of the publication analyses the existing state of the UltraScan gateway, and identifies its limitations. One of the main contributions of this research is enabling the UltraScan gateway to access different kind of HPC resources and middlewares. As mentioned above, this is supported by introducing the security model and integrating the standards-based client API through which an abstract Airavata extension is realised that helps to manage distributed and remote job executions and their corresponding data.

The second part of the manuscript focuses on the architectural design and implementation aspects. The implementation is deployed on production computing sites at Jülich Supercomputing Centre, Germany. Figure 4.6 depicts an architectural view of the different software components involved.

The underlying research resulted in a client API that was published to the Apache Software Foundation (ASF) repository. Due to active contributions, the author has been granted formal Apache developer rights to make source code contributions. The resultant client API is also part of the stable distribution of the Airavata framework [93].

4.3 Enhanced Resource Management Enabling Standard Parameter Sweep Jobs for Scientific Applications

S. Holl, **M. S. Memon**, B. Schuller, M. Riedel, Y. Mohammed, M. Palmblad, A. Grimshaw. Enhanced Resource Management enabling Standard Parameter Sweep Jobs for Scientific Applications, *International Conference on Parallel Processing – The 42nd Annual Conference ICPP 2013 Ninth International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems (SRMPDS)*, Lyon, France, 1 October 2013, IEEE, pp. 783–790 (2013)

This publication contributes to objectives TO2 and TO4, that provides a technological basis for enabling massively parallel scientific applications, in particular of parametric nature. Pragmatically, computing learning models are mostly parametric and require multiple iterations until some threshold is achieved. This work demonstrated the automation of supervised and unsupervised learning approaches.

This paper focuses on the core design and realisation of a standards-based parameter sweep model. The implementation is based upon the OGSA-BES specification and UNICORE's execution management framework. Although the OGSA-BES interface comes with a request and response protocol for job management and monitoring, it does not define the contents of the job submission request contents. The *JSDL-Parameter Sweep* (JSDL-PS) [45] specification is an extension of JSDL and provides an abstract model definition for parametric jobs. As part of this dissertation, the research presented in this paper also contributed to the interoperability tests with other computing middleware providers. Conventionally, a parametric job incarnates to an ensemble of jobs generated on certain criteria, such as a list of application arguments, file names or environment variables. In order to form this structure a mechanism should be in place to resolve the composite nature of job submission and management requests. Thus, in this dissertation a multi-job management pattern is derived and implemented that orchestrates multiple parametric jobs in reply to a single user request. The master-worker pattern helps to curb challenges of parametric jobs spawned due to more complex nested structures (e.g. lists or sets). The adoption of a master-worker pattern here supports the implementation in two ways, i) structuring and incarnating before submission ii) managing the job session data with complex and nested parameter combinations. As proof of implementation, UNICORE's execution framework, XNJS [100] is extended. The output of this research endeavour resulted in the standards-based, stable and platform-agnostic parameter sweep library known as JSDL-Sweep [78].

Furthermore, this paper gives more insight into the implementation details of the JSDL-Sweep library, which is compliant with the *JSDL-Parameter Sweep* (JSDL-PS) specification. Finally, the paper compares the performance of running a single parametric job request with a separate statically generated job requests. It demonstrates a significant performance gain by reducing the number of web service invocations and having a smaller data foot print of the overall composite job.

4.4 Enhancing the Performance of Scientific Workflow Execution in e-Science Environments by Harnessing the Standards based Parameter Sweep Model

M. S. Memon, S. Holl, M. Riedel, B. Schuller, A. Grimshaw. Enhancing the Performance of Scientific Workflow Execution in e-Science Environments by Harnessing the Standards based Parameter Sweep Model, *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (XSEDE'13)*, San Diego, California, USA, July 22–25 2013, ACM Press, pp. 56, 56:1–56:7 (2013)

This publication enables a scientific workflow with a focus on achieving parameter optimisation through standards-based parameter sweep models. In this work, the Taverna workflow system is extended to run parametric workflow jobs on different HPC sites through the UNICORE middleware. This contribution serves objectives TO1, TO3 and TO5.

The research work underlying this paper was also motivated by learning models and enables a scientific workflow through an automated parameter sweep mechanism. The underlying research focuses on a scientific workflow that uses the Taverna [88] workflow management system at the user end and UNICORE on the server side. The use case aims to perform parameter optimisation using genetic algorithms. For this purpose, a specialised parameter sweep plugin for Taverna is implemented. This plugin uses the JSDL Sweep library [78] along with the native UNICORE client libraries in order to submit scientific workflows on UNICORE managed execution service instances.

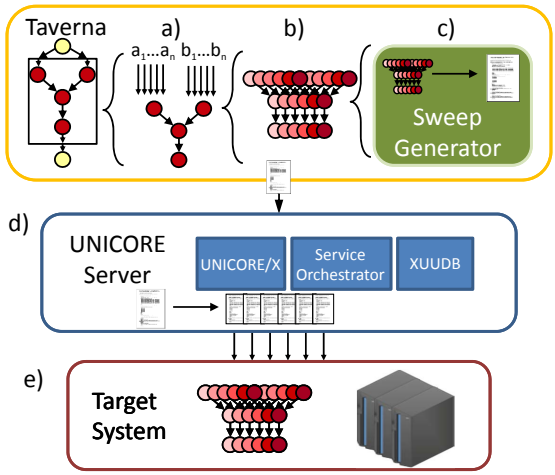


Figure 4.7. The figure shows the integration of the Taverna workflow engine with the JSDL-Sweep library and the UNICORE middleware. ©ACM 2013.

Figure 4.7 depicts the software integration of Taverna and UNICORE in conjunction with the sweep functionality. It also shows the flow of a parametric sweep job request. The figure presents a sequence that starts with i) the definition of two parameters ax and bx, ii) Taverna creates several parallel workflow instances, iii) the sweep generator takes them and creates a JSDL-PS job request iv) that is sent as a remote job request to a UNICORE Server instance and (v) then the request is finally processed at the Target System layer. See *Paper IV* for more details. This paper describes significant performance improvements while running multiple sub-workflows concurrently.

4.5 Facilitating Efficient Data Analysis of Remotely Sensed Images using Standards-based Parameter Sweep Model

M. S. Memon, G. Cavallaro, M. Riedel, H. Neukirchen. Facilitating efficient data analysis of remotely sensed images using standards-based parameter sweep model, *Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2017)*, Fort Worth, Texas, USA, 23 July–28 July 2017, IEEE, pp. 3680-3683 (2017).

The research that is described in this publication enables an automated classification of remotely sensed images (learning models) using the standards-based job submission and parameter sweep models. Image classification is performed through a parallel version of a Support Vector Machine *(SVM), termed PiSVM. Classifying data is a workflow of multiple steps, and cross-validation is one of its essential steps. In this research endeavour, the critical phase of cross-validation is automated for model generation and testing. The JSDL-Sweep library is used for automating the cross-validation phase and UNICORE-BES is applied for the remote execution of HPC jobs. This publication achieves the objectives of TO1, TO3, TO4 and TO5.*

The research in the paper focused on the classification of the Indian Pines data set by using *Support Vector Machine* (SVM)s. Indian Pines [19] is a well known data set used in most of the remote sensing research papers. It was acquired in 1992 by the AVRIS sensor recording information from fields and a variety of crops. To extract any meaningful information from Indian Pines is critical as it consists of 30 features and 1417 x 617 pixels (with a spatial resolution of 20m).

The research presented in this paper focuses on automating massively parallel machine learning workflows, i.e. to optimise the identification of hyper plane parameters C and G, through the grid-search method. This process is called the cross-validation phase, which then leads to model generation. The research prototype presented is based on the *JSDL-Parameter Sweep* (JSDL-PS) standard in combination with *OGSA-Basic Execution Service* (OGSA-BES). In particular, the implementation uses UNICORE-BES and supporting Python and Bash scripts. The realisation of the automated classification of the Indian Pines data set enables parallel cross-validation on HPC in a single step. The only requirement is to prepare the corresponding parameter look-up files, known as

the data dictionary, with possible enumeration values for the grid-search. The implemented software further performs all the processing and produces one pair of C and G parameters.

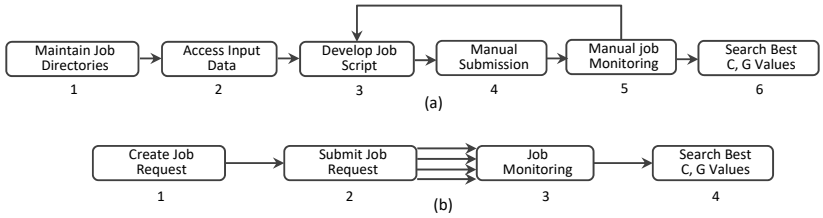


Figure 4.8. The figure compares the (a) manual and the (b) automated process of cross-validation. ©2017 IEEE.

Figure 4.8 compares two workflows with i) manual analysis and ii) automatic parameter identification. The experiment with the automated mechanism shows a significant impact with respect to usability and overall workflow runtime. The paper not only points the way to the robust classification life cycle of the given use case, but also provides an avenue for other parameter-oriented and parallel machine learning methods through a standards-based mechanism.

5 Conclusions

5.1 Summary

This thesis was motivated by multiple scientific case studies from different disciplines. It comprised four main *Thesis Objectives*: *TO1* – assess the requirements of two fundamentally different data analysis approaches; *TO2* – compare and contrast requirements from two different (learning and physical) models; *TO3* – develop an architectural design based on primary functional components; *TO4* – provide technical realisation of the software prototypes; *TO5* – adoption of the produced software prototypes in the form of distinct case study implementations. The thesis objectives *TO1* and *TO2* are fulfilled by exploiting the case studies in order to elicit the requirements, while *TO3*, *TO4* and *TO5* are achieved in the form of the concrete design and implementation of solutions for enabling physical and machine learning models on distributed computing infrastructures. This doctoral dissertation produced several contributions in the form of design solutions and their implementations which have become part of the production middleware platforms. These contributions are devised to form a generic standards-based framework to enable and facilitate the semi-automated remote processing of complex applications and access to distributed data sets. The standards-based implementation mitigates major obstacles while running computations in distributed, multi-resource management environments. Different data analysis concepts are evaluated and compared for their suitability for addressing selected scientific case study needs. As part of the work, architectural designs were produced to support multi-disciplinary scientific scenarios. A strong focus was placed on enabling convenient access for end users of HPC and HTC infrastructures and scientific gateways, most notably in the form of automated data analysis, which mainly covers a generic mechanism for reusability by other scientific and engineering applications. This thesis has also directly and indirectly contributed to several OGF[4] standards, e.g. EMI-ES [99], Activity Instance Document [111], and to the following standard interoperability experience documents, e.g. ByteIO [31] and *High Performance Computing Basic Profile* (HPCBP) [108].

The job submission scenario enabled users to manage and monitor their jobs running on disparate and remote execution services. The sequence of this scenario starts with a user request that is processed and accepted by the remote execution service after a response is returned. This scenario and the request and response data structure concerned are realised through the OGSA-BES specification. The implementation was performed as part of this thesis: UNICORE-BES [76] which is compliant with standards-based job submission and management interfaces. *Paper I-V* are based upon UNICORE-BES with respect to the technical and application adoption scenarios. The benchmark analysis of the preliminary UNICORE-BES implementation is stated in [62]. The

OGSA-BES specification provides a generic interface and model for job management. Nevertheless, it has still some deficiencies while providing user-centred interactions, such as resource capability advertisement, job chaining or dependency representation, and server- or client-initiated data transfers. In response to these requirements, primary extensions necessitated an upgrading of the existing model of the specifications, which has been contributed by the author of this thesis to the *Open Grid Forum* (OGF) standard publication as *EMI-Execution Service* (EMI-ES) [99].

The GLUE2 [15] specification, particularly its XML Schema-based rendering [16] is used to identify the sites and resource capabilities through end-user clients and resource brokers. As part of this doctoral dissertation, the GLUE2 model was carefully analysed while integrating the OGSA-BES standard with the subset of the extensive GLUE2 model. The outcome of this work is useful in cross-infrastructural workflows spanning HPC and HTC sites. Also, the GFFS and UNICORE-BES implementations are integrated with the purpose of bridging heterogeneous services. The authentication and authorisation model has been developed to support the provisioning of multi-identity assertions. More details are provided in *Paper I*. The research prototypes produced are now included in the UNICORE and GenesisII's software repositories [77, 78, 60].

The client API plays a vital role in emerging scientific gateway communities by providing essential utilities and functions required by gateway frameworks and end-user portals. The research presented here identified and engaged real scientific communities by running their computations on distributed HPC infrastructures. In particular, the UltraScan gateway is considered as a motivational use case, providing concrete user and scientific requirements, and the standards-based client API and the Airavata GFAC extension were developed on this basis. More detailed results are presented in *Paper II*. Furthermore, the client APIs contributed to the Activity Instance Document [111] standard that provides additional support for debugging and error analysis purposes.

In order to support workflows related to the learning models, the standards-based *JSDL-Parameter Sweep* (JSDL-PS) model has been adopted to support the semi-automatic processing of a typical machine learning life cycle, e.g. classification and clustering. The data model presented in JSDL-PS is particularly used to automate the submission and execution of multiple jobs for identifying the optimal hyper parameter space. For instance, when using the *Support Vector Machine* (SVM) method together with the *Radial Basis Function* (RBF) [26] kernel configuration (during the cross-validation phase), the identification of the best single parameter combination is obtained to produce a model with the highest accuracy. The development and integration of the platform-independent and open-source JSDL-Sweep library aims to offer a general purpose utility for diversified scientific workflows, including machine learning and data mining. *Paper IV and V* describe the use case and present the implementation details.

5.2 Future Work

In the emerging and dynamic technological environment of computing architectures where distributed programming models are continuously evolving, it is essential for data models to address the hardware and software systems deployed. In view of this

concern, the learning and physical models presented may pose multiple challenges when accessing current resource management systems and computing architectures. One prominent scenario is the case of a single computing cluster with heterogeneous and modular hardware architectures grouped in mini-clusters with different CPUs and memory configurations. The EU-funded DEEP–EST project [40] is one such example that develops multi-purpose and modular supercomputing architectures catering for different kinds of workloads and applications by providing different node groups with the most suitable resource capabilities in terms of network, processors and storage. In addition to multiple hardware specifications, the DEEP–EST managed cluster is laden with various parallel programming runtime environments. As future work, a relevant information and data model is required to conceptualise such set-ups with adequate access abstractions. This scenario will eventually affect the OGSA-BES and GLUE2-based interfaces and will require an extension to the existing research prototypes. Embracing this change at the standards level will impact the range of services starting from the core job execution or middleware layer up to and including to the libraries for the end users.

Learning models are attracting increasing attention as data-oriented computing frameworks have evolved. The open source Hadoop framework [110] is one of such major enabler providing an ecosystem of tools, execution frameworks, programming models [39] and distributed file systems [102]. Another example is the more recent Apache Spark [113] framework that provides APIs and tools for processing relational- and graph-based data structures on general-purpose cluster computing systems. Both Hadoop and Spark are not only used by commercial applications, but have also been adopted by scientific communities. Integrating HPC and Hadoop- or Spark-based services can support cross-site data and compute-intensive workflows through common interfaces and models. To widen the scope of the work presented in this dissertation, the above-mentioned standards and models and their implementations, OGSA-BES, JSDL and GLUE2, and respective client APIs have to be adjusted to accommodate functional capabilities through a common set of APIs and implementations.

In facilitating the fast-paced evolution of learning models and the respective methodologies such as the application of deep neural networks for processing complex image and video data sets, it becomes imperative to align the middleware abstraction models accordingly. The state-of-the-art frameworks offering distributed deep learning implementations are Tensorflow [11] and PyTorch [90]. As a future direction, the workflow of automating deep learning scenarios on HPC architectures can be realised through the functional components presented in this dissertation. Specifically, this effort envisions the processing of large data sets obtained through remote sensing devices. The initial approach is published in *Paper VII*.

Apart from the job management and monitoring representations, the underlying research focused on the use and impact of the iterative execution of learning models through standards-based parametric data structures. However, the current *JSDL-Parameter Sweep* (JSDL-PS) specification lacks parametric operators to represent new sophisticated user-defined functions for evolving learning models. It is therefore necessary to have enhanced functional operators permitting new operations. For this endeavour, the development will be integrated with the existing model of the JSDL-PS specification, and also implemented in the JSDL-Sweep library [78].

Paper I

Enabling Scalable Data Processing and Management through Standards-based Job Execution and the Global Federated File System

M. S. Memon, M. Riedel, A. S. Memon, C. Koeritz, A. Grimshaw, and H. Neukirchen.

Journal of Scalable Computing: Practice and Experience, Vol 17 No 2, 115–128 (2016).

M. S. Memon has actively participated in the main analysis, design and implementation of the UNICORE and GFFS integration. He is the main author of this publication. He outlined the paper structure and wrote majority of the paper contents.

ENABLING SCALABLE DATA PROCESSING AND MANAGEMENT THROUGH STANDARDS-BASED JOB EXECUTION AND THE GLOBAL FEDERATED FILE SYSTEM

SHAHBAZ MEMON[†§], MORRIS RIEDEL[†§], SHIRAZ MEMON[†§], CHRIS KOERITZ[‡], ANDREW GRIMSHAW[‡], AND HELMUT NEUKIRCHEN[§]

Abstract.
Emerging challenges for scientific communities are to efficiently process big data obtained by experimentation and computational simulations. Supercomputing architectures are available to support scalable and high performant processing environment, but many of the existing algorithm implementations are still unable to cope with its architectural complexity. One approach is to have innovative technologies that effectively use these resources and also deal with geographically dispersed large datasets. Those technologies should be accessible in a way that data scientists who are running data intensive computations do not have to deal with technical intricacies of the underling execution system. Our work primarily focuses on providing data scientists with transparent access to these resources in order to easily analyze data. Impact of our work is given by describing how we enabled access to multiple high performance computing resources through an open standards-based middleware that takes advantage of a unified data management provided by the the Global Federated File System. Our architectural design and its associated implementation is validated by a usecase that requires massivley parallel DBSCAN outlier detection on a 3D point clouds dataset.

Key words. UNICORE, Genesis II, statistical data mining, data processing, distributed file system, security, standards, parallel processing

1. Introduction. An ever increasing number of datasets from scientific experimentation such as earth observatories or computational simulations generate an enormous amount of information for discovering useful knowledge. In order to analyze data, the area of statistical data mining provides useful methods and tools to extract and explore useful patterns or prediction models. The field of statistical data mining comes with intuitive methods to learn from data, using a wide variety of algorithms for clustering, classification and regression. Several implementations are available, for example, Matlab, R, Octave [3], or scikit-learn. Mostly, these tools offer serial implementation of the algorithms, which is quite challenging (i.e. insufficient memory, extremely long running times, etc.) for processing the volume of data having terabytes or petabytes of magnitude. Considering that amount, the resources running the data processing tools require large number of processors, as well as much more primary and secondary storage. Therefore, parallel tools and platforms such as Hadoop [15] implementing the map reduce paradigm [18] and selected massively parallel algorithm developments based on the MPI and OpenMP environments are commonly used.

We observe mainly tools for (High Performance Computing) HPC and High Throuput Computing (HTC) paradigms evolving concurrently, but each supporting their own set of requirements. Scientific communities, either from biology, physics and medicine adopt more conservative approaches in order to retain their focus on scientific findings and as such traditional HPC environment still play a major role in the relatively new realm of 'big data'. Given the stability of HPC environments and its benefits using locally parallel filesystems with parallel I/O techniques motivates our work to enable straightforward job executions managed by HPC sites that seamlessly

[†]Juelich Supercomputing Centre, Forschungszentrum Juelich GmbH Juelich, Germany
[‡]Department of Computer Science,University of Virginia Charlottesville, USA
[§]School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland

access data from a distributed file system service which has not been traditionally supported in HPC-based execution services.

We validate our approach with a use case from earth science using 3D points cloud obtained from devices that measure a large number of points of an object surface (i.e. in our case the inner city of Bremen). The data analysis of this dataset has the goal to cluster special data points and identify any noise elements. In this paper we describe the architectural design and implementation necessary to run data analysis jobs on HPC resources through standards-based UNICORE middleware [10] and uses data from the Global Federated File System [28] that we derive from the architecture of the Extreme Science and Engineering Discovery Environment (XSEDE) [24]. UNICORE is a HPC middleware and deployed on production on XSEDE supercomputing sites, whereas the GFFS is a distributed network file system which is an integral component of the Genesis II platform, that is also consider to be a middleware element in XSEDE. Our architectural design overcomes the limit that the data is hosted by the GFFS cannot be easily made available to the job executions that perform data clustering over the points cloud data set.

The paper is structured as follows. Section 2 describe the basic background of the technologies and standards used as part of our research. Section 3 lists a detailed requirement analysis we obtained during the course of the integration effort. Section 4 describes the security model and the implementation we derived for supporting the requirements from Section 3. Section 5 offers detailed insights on our architectural design and its realization that enable the UNICORE and GFFS integration while addressing the selected requirements. Section 6 takes a massively parallel data analysis application in order to validate our work based on a real world use case. Section 7 provides a brief overview of the related work, and the paper concludes in Section 8.

This article is a joint and extended version of [34] and [40].

2. Background. This section gives a brief background of the technologies, algorithms and standards that supported our work.

2.1. UNICORE. UNICORE is an HPC middleware which is built upon the priniciples of Service Oriented Architecture (SOA). It realizes compute, information and data functions through a set of stateful web services [42]. These services are designed in such a way that they enable seamless access to heterogeneous high performance computing resources. In this sense, the middleware layer to these clusters provides access and location transparency to compute and and thus offers scientists a unique environment hiding low level technical complexities (i.e. avoiding writing and submitting error-prone scheduler dependent job scripts). The compute access transparency enables an abstraction of different flavors of resource management systems (sometimes also referred to as schedulers), such as SLURM [45] or Torque [6], and more notably through a unified and standard interface. Figure 2.1 depicts the basic UNICORE architecture that is composed of layers with distinct functionality, including Client, Services and Target System Interface.

The client layer provides API and end user interface, which include interfaces for constructing and sending client requests to remotely deployed services. The client side API is useful for scientific communities which are not necessarily using the UNICORE's povided interface, but instead their own clients such as their application specific science portals or gateways (e.g. UltraScan Scientific Gateway [33]. Hence, all important functionality of the middleware services can be invoked through the direct client API interaction. The end user interface offers a rich client interface called UNICORE Rich Client (URC) [19], with advanced user controls to compose and or-

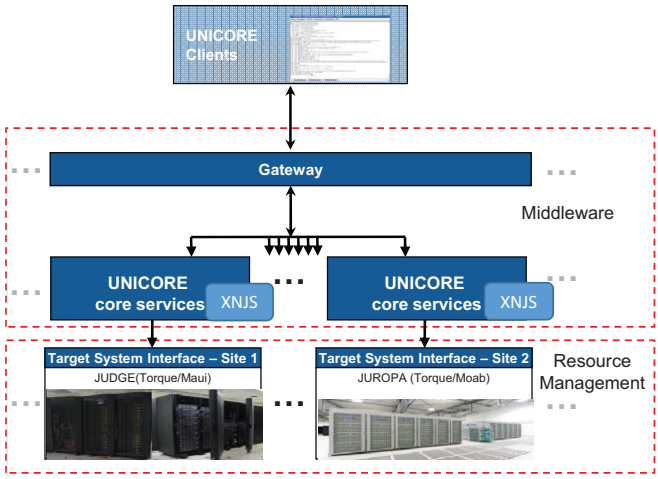


FIG. 2.1. *Basic UNICORE architecture with example deployments of two HPC systems in Juelich (JUDGE and JUROPA).*

chestrate scientific workflows. The second variant is a command line client called UNICORE Command Line (UCC), which provides an interface for advanced users who know the low level details of writing job request scripts to be executed jointly on the batch system. For a more detailed architecture explanation we refer to [10].

The Services layer plays a vital role in enabling job execution and data management by means of SOAP [16] over XML based web services. Not only the Services layer implement the core functionalities, but also the hosting environment which can host and deploy stateless and stateful web services, for instance, WS-I (Web Services Interoperability) [5] and WS-RF (web Services Resources Framework) [42]. Job management functionality implements a complete life cycle through which job passes, and that includes submission, monitoring and data staging. The job management functionality is supported by UNICORE's embedded scheduling and execution framework, called XNJS (Extended Network Job Supervisor) [43]. It manages the incoming middleware requests against the hosted application and resource capabilities (for example number of available nodes, processors per node etc.). The Services layer gives a configuration based interface to expose underlying cluster resource and environment, so that XNJS can perform resource match making upon the client initiated job requests. After validating the job request, the XNJS component formats the job to the generic UNICORE protocol, and then sends it to the resource manager specific implementation of Target System Interface. This is the layer where the generic UNICORE script gets translated to the request formatted according to the batch system.

As a summary, a simple job execution sequence comprises of, client job submission to the Services layer, then the request is forwarded to XNJS, and then it is communicated to the Target System tier. This tier in turn directly interacts with the low level batch system, and fetch job statuses, and manage underlying running file transfers during the job's execution life cycle.

2.2. The Global Federated File System. The Genesis II Global Federated File System (GFFS) [28] is a distributed file system that provides researchers with tools for securely managing and sharing their scientific data. The GFFS offers a set of interfaces that manage jobs and provide access to the required scientific data. This is achieved through the GFFS-Queue component, which is also based on SOA wherein standards-based interfaces are adopted for storing and accessing remote compute and data resources. In order to support federation across different organizational entities, the GFFS provides a hierarchical file system structure with standard namespace locations for storing user profiles, groups, directories, and service elements such as meta-scheduling queues and Basic Execution Service endpoints (BES) [29] for processing jobs. Figure 2.2 provides an overview of the integrated architecture with Genesis II and UNICORE Basic Execution Service (BES) endpoints interacting with the GFFSs root container.

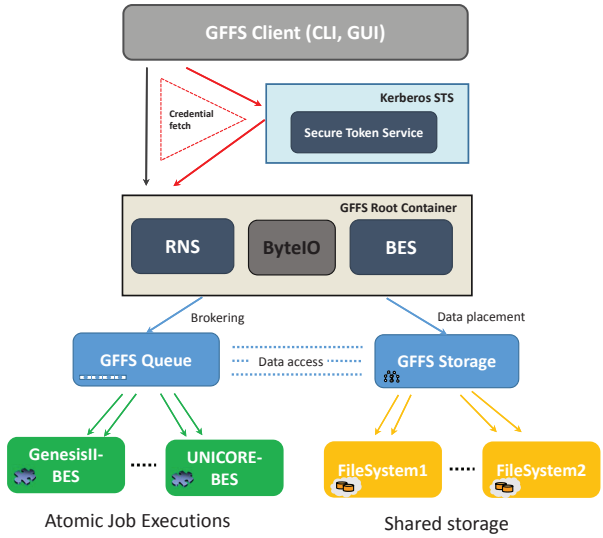


FIG. 2.2. *The Global Federated File System (GFFS) architecture.*

The GFFS implements many of the standard Unix commands (such as `cp` and `mv`) in a console mode through the so called grid shell. There is also a GUI view of the GFFS, which supports rich drag and drop file management. The GFFS also provides a FUSE file system interface [2] that allow users to mount the GFFS on a Unix directory and operate on files in the GFFS as if a user is interacting with her local file system. The GFFS has an export feature like NFSv4 that allows users to share part of their own file system visible within the GFFS, and to other users part of the broader federated infrastructure. The GFFS Queue is a metascheduler that supports submission of multiple jobs for subsequent distribution to the execution service endpoints connected to the queue. The GFFS Queue provides researchers with a mechanism for managing and controlling their computations via a GUI as well as with familiar command line tools such as "qstat" and "qkill". Jobs will be distributed to BES resources automatically by the GFFS Queue, but can also be rescheduled as needed. The XSEDE project benefits from the GFFS by giving researchers a way to

securely share data with their colleagues and by providing a high level view of the computational resources available at the XSEDE infrastructure through the GFFS Queue.

2.3. Standards. RNS (Resource Namespace Service) [36] is an Open Grid Forum initiative that standardizes the naming of distributed resources that form an infrastructure. It has a simple set of operations for managing grid and cluster services mapped as file system operations such as `rm`, `mkdir` or `cp`. The RNS specification provides client applications to couple WS-Addressing [22] based endpoints with human readable notations. For instance, a job execution service managing multiple jobs can be represented as a parent directory and individuals jobs are child directories which may further contain the contents of their working directories.

Considering the stateful service based endpoints, wherein web service resources can have a nested heirarchical structure, the RNS representation is very helpful in providing an access and location transparency to underlying resources. The GFFS mentioned earlier implements the core RNS and its Web Services Resource Framework (WSRF) [37] rendering to access service endpoints. The statefulness gives individual access which is very much analogous to the domain of distributed remote objects. As far as the data transfer is concerned the GFFS primarily uses the ByteIO [35] standard. ByteIO provides a set of interfaces to interact with bulk data sources and sinks. The ByteIO standard enables large amount of data in an efficient way. It has two interfaces, RandomByteIO and StreamableByteIO. RandomByteIO provides an interface to access bulk data in a stateless and random manner. This interface is normally being called when a client transfer large files. StreamableByteIO interface allows data transfer data in a stateful manner. It is normally used for accessing short files, in most cases standard outputs of managed jobs.

The Job Submission and Description Language (JSDL) [8] is an XML-based comprehensive data model for specifying computational job requirements consisting of application, resources and data concepts. UNICORE and Genesis II clients specify job requirements in the JSDL format. UNICORE server side implements most of the JSDL, and also its related profiles and extensions.

The JSDL specification has a generic model for representing multiple type of resource settings, such as HPC and HTC. The JSDL model further provides a set of profiles which imposes constraints on requirements according to the type of resource architecture. These requirements may include, file staging modalities, parallel execution environments and parametric jobs. This paper mainly targets HPC resource types which normally use parallel execution environments and a resource management system. HPC resource profile [12] enable users to specify more internal HPC architecture specific requirements in combination with restrictions on the execution service. The HPC file staging profile [44] captures data movement specific elements to be used within heterogeneous cluster environments. This profile impose constraints on using HPC specific data staging attributes as part of the job submission request. For instance, the request may contain FTP user name and password credentials for the BES instance to carry out third-party file transfers on user's behalf.

The related technologies presented above provide a base for providing a seamless and robust middleware platform to tackle big data challenges. One of the common big data processing machinery requirement is to have an iterative execution for discovering optimal set of algorithm parameters. Specifically, data clustering algorithms such as K-Means or DBSCAN require certain parameters before their processing. In the case of iterative execution multiple runs with a varying set of parameters are required.

If we combine these runs into a single composite job then this kind of job is called parametric. UNICORE as our base execution middleware supports parametric jobs through the JSDL Parameter Sweep extension [23]. This specification provides an intuitive model to parametrize multiple parts of job request per se. It may allow client application to sweep over a list of arguments, file transfer locations and environment variables specified under the JSDL request. The sweep model can represent different kind of iterations, either be it a element-wise iteration on a set or a counter with configurable stepping factor. There are two major kind of sweeps the specification provides, Document sweep and File sweep. The Document sweep provides a model to modify a requested JSDL instance. The File sweep is an advanced model which presents a data structure to modify the contents of the files imported before the job execution phase. This kind of sweep is applicable to text based files. UNICORE middleware implements both kind of sweeps through its client API and command line client (UCC). This specification is used to automate the iterative data clustering on the points cloud data set we use in this paper.

OGSA-Basic Execution Service (BES) [8] is an Open Grid Forum (OGF) initiative which provides a web services-based interface for managing and monitoring computational jobs in HPC and HTC environments. For a job submission use case BES interface accepts a JSDL instance and its related profiles as a parameter and then runs it on back-end resource. The focus of this paper is based on a scenario in which UNICORE server expose it computing capabilities via BES model, and Genesis II client use this interface to invoke remote calls on the UNICORE endpoint.

2.4. Unsupervised Learning. While the overall architectural design in this paper is applicable to many learning models, our work is using parallel version of the Density Based Spatial Clustering for Application with Noise (DBSCAN) algorithm [21]. The focus is rather on the access of the algorithm implementation through the UNICORE middleware and the GFFS based file system. Therefore, this section briefly introduces the DBSCAN method. Goetz et al. describes more details on the algorithm implementation in [27], which gives more detail on what parallelization strategies are used to achieve scalability and high performance while analyzing large data sets. DBSCAN [21] is an unsupervised density based clustering algorithm. The cluster based on density is represented by a number of points MinPoints within a specified radius Epsilon. These are the important user defined parameters of the algorithm to identify the clusters.

i) Core point: A central point in a dense region, it has more than a specified number of minimum points MinPoints within its neighborhood (or radius) Epsilon.

ii) Border point: A point that lies on the border of the dense region, it fewer than minimum points MinPoints within its neighborhood (or radius) Eps

iii) Noise point: A point that is neither a core point nor a border point

DBSCAN intrinsically enables maximizing the local point density recursively. It sets apart from other clustering algorithms as it detects the clusters of arbitrary shapes and sizes. Notably, it is resistant to noise and suitable for finding anomalies or filter specific noise signals from the data. We use the parameter-based DBSCAN learning algorithm as a specific example of how our architectural design and its implementation can be generically used by a wide variety of learning algorithms in this paper.

3. Requirement Analysis. During the course of transparent integration for bridging both technologies, we identified multiple requirements which not only aims at superficially combining them, but also some extensions in the UNICORE services layer which will help to tackle "big data" challenges from machine learning and data

#	Requirements	
	Description	*Environment*
R1	Secure Trust Delegation	GFFS and UNICORE
R2	Openness	OGSA-BES, JSDL
R3	Data transport	RNS, BYTEIO
R4	Infrastructure integration	XSEDE, MYPROXY
R5	Transparency	UNICORE Server
R6	Parameter sweep	JSDL Parameter Sweep
R7	Extensible resource and job model	GFFS and UNICORE

TABLE 3.1
Summary of Requirements

mining. The integration has taken XSEDE infrastructure as an example, but the implementation is applicable to any distributed computing and storage infrastructure. The major integration requirements from both the technologies' perspective are summarized together with relevant technology environments in Table 3. They lie in the following areas. R1) Secure Trust Delegation: As in a distributed service interaction a user interacts with a portal or meta scheduler which then forwards the request to a service that takes care of the job execution and also calls upon data management services to pull and fetch data. While the user is participating in the very beginning, then the following phases are to be done by other services, require some kind of trust delegation which the user entity assigns to the target job execution and data management services to act on her behalf. In our scenario a user communicates a data oriented job request with a set of input data staging elements, therefore trust delegation has to be implemented by the UNICORE platform to understand the GFFS user requests. R2) Openness: In any kind of communication between a user and the GFFS or UNICORE, it should support standards-based protocols, so that users or services from different middleware backgrounds can easily interact through UNICORE and the GFFS client-side APIs. R3) Data transport: As UNICORE jobs are intended to use data from the GFFS, the running jobs should be able to upload and download data from the file system space. R4) Infrastructure integration: XSEDE-based identity management should be understandable to both layers of job submission and data management middlewares. R5) Transparency: The jobs managed by UNICORE in an HPC environment which accesses data from the GFFS data should not know the physical location and also on how the data is structured across data nodes within the file system space. R6) Parameter sweep: This requirement is very specific to jobs which require re-running the same application but with different parameters:these are called composite or parametric jobs. In a parametric job, the execution middleware iterates through a set of parameters provided by a user job submission request and creates a separate job internally for each parameter combination. In this case UNICORE middleware should be capable of interpreting and incarnating parametric jobs. R7) Extensible resource and job model: As supercomputing architectures are evolving to support data and network intensive applications, the hosting middleware should be adaptive to new changes and thus possess an extensible model for users specifying sophisticated requirements. For instance number of GPGPUS or use of execution environment.

4. Interoperable Security Model. The GFFS security model is based on the Security Assertion Markup Language (SAML) [41] standard, and takes the UNICORE SAML profile [14] as a reference implementation. The GFFS extends the UNICORE's SAML profile to represent the trust delegation chains of the Genesis II security model. A delegation chain encompasses that a user has delegated some level of trust to a service in the GFFS in order to achieve a task, such as processing a job. Mostly, delegation operations include three entities, (1) a grid user (for our example, called U, (2) a TLS connection identified by an X.509 certificate (called C), and (3) a grid resource (called R). Longer delegation chains usually contain all three of these types of entities as the first links in the chain.

In the GFFS, the first entity U is always a user defined as a grid STS (Secure Token Service) object. This entity is the prime mover for any operations that are performed in the GFFS. The user's rights within the grid's access control list permission system dictates what that user can and cannot do with regard to every grid resource.

The client software must authenticate as the grid user U to obtain services from a GFFS container. This is where the second security entity C comes in; it is the connection by the client software at the behest of the user. Initially, the client credentials only contain C, as one makes the connection before STS authentication occurs. In the XSEDE login process, this connection is always based on X.509 credentials obtained from a certificate authority service, the so called XSEDE MyProxy server. Thus, it can be a well-known identity within both Genesis (via a Kerberos-based STS) and UNICORE (via the grid-mapfile). In the example, the client software first authenticates to MyProxy by using user name and password to obtain the certificate C. Then, the client authenticates to the Kerberos STS in the GFFS to obtain grid user U.

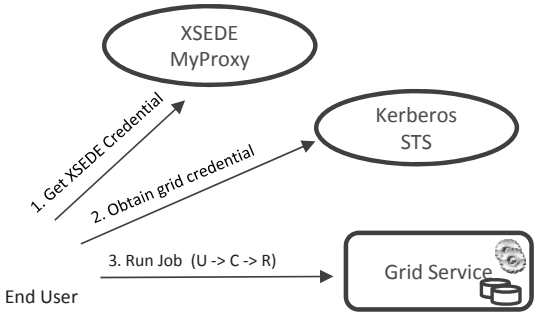


FIG. 4.1. *End User interaction with security and grid services.*

After the user authenticates, the client software's credential wallet will contain the first delegation of trust, U→C. This states that the grid user U trusts the TLS connection C to act on its behalf. Afterward, all of the actions taken by the TLS connection C are understood to be U's actions. Any access that provides U with permissions will also be granted to C. That may include submitting a job to a BES named R. This is the second point where trust is delegated; the certificate for TLS connection C signs a new trust delegation that expresses "C trusts R" to perform a job execution. This extends the length of the delegation chain by one, so that it now has all three entities involved in two trust delegation objects. This can be represented with delegation arrows such as:

- First delegation: U→C (The grid user trusts the TLS connection)

- Second delegation: C→R (The TLS connection trusts the resource)
- Full Chain: U→C→R (The user U trusts the connection C which trusts R to run a job)

This new delegation chain can be presented by resource R when it needs to do further actions on the grid user's behalf. Moreover, actions might include storing file staging results back to RNS space (U→C→R→D, where D is a Data folder or possibly submitting the job to another BES for final processing. The important point is that entity D is just another resource to which trust can be delegated, and the chain can be continued in that manner for as long as its delegation depth limit allows. Figure 4.1 depicts a typical interaction of an end user with the security services and a target grid (execution or data management) service.

One challenge while interoperating between the GFFS and UNICORE integration arose due to a difference in interpretation of the SAML assertions. The delegation chains in the GFFS are tightly-coupled, and do not allow mixing and matching of individual entities. This is not directly provided by the UNICORE SAML implementation, which permits the receiver to mix and match any delegations provided in a message (U→C→R is considered to be two separable delegations U→C and C→R). In the GFFS model, a delegation chain must be used in its entirety or not at all.

To address this difference, the GFFS implementation of SAML adds a unique identifier to each SAML assertion. A chain such as U→C→R is built by embedding the identifier of the U→C assertion in the C→R assertion. To make this cryptographically secure, the signature of the U→C assertion is also embedded in the C→R assertion before C→R itself is signed. This enforces the connections between the GFFS delegation chains while still leveraging the UNICORE's Security Assertion Markup Language (SAML) implementation. Upon reception, the chains are reassembled and any assertions that are referenced by a longer delegation chain are removed from the pool of available assertions.

The Genesis delegation chain model supports having multiple chains in a credential wallet. This supports the user possessing multiple different types of identity and authorization on resources. The users will always have their own identity as a credential, which allows them access to resources where the user has been given explicit permission. The user will also usually have at least one group credential, which allows them access to portions of the grid file system. Additional group credentials may convey access to different BES or queue resources within the grid. Thus the credential wallet approach supports a flexible authorization appropriate to the variety of grid resources, possibly across multiple administrative domains, that may be required for the user's work.

The signing of credentials ensures that it is computationally infeasible to create a fraudulent credential chain where a new identity is inserted into the credential chain. Each credential records the signature of the prior element in the chain, along with its unique identifier. Thus an attacker would have to compute a valid XML digital signature inside a valid trust delegation object, where the unique id is also properly signed by that signature.

To ensure that the credential wallet cannot be easily compromised and used for playback attacks (where the valid credentials of a user are stolen and used by a different user), all credentials must be "anchored" with the current TLS session credential of the grid client. At least one link in the credential chain must be identical to the TLS session certificate. This ensures that playback is very difficult indeed, since the stolen credentials must be based on the TLS session key that the user was employing

at the time the valid credentials were minted. This mitigates attacks upon the server, where the container database is compromised. Attacks using an entire set of stolen client credentials are also somewhat mitigated, since the TLS session certificate is based on a short-lived key pair.

With respect to the requirements mentioned in Table 3, the given security model implementation covers R1-Secure Trust Delegation. XSEDE's MyProxy access is provided to allow XSEDE users run job with their infrastructure credentials. This feature relates to R4-XSEDE MyProxy integration.

5. Integrated Architecture and Implementation. We have extended UNICORE's server tier to accept the incoming requests incoming from Genesis II remote clients. The remote clients here implies the GFFS's GFFS-Queue component which is an entry point for a user to submit job. The GFFS-Queue acts as a meta scheduler that schedules the user's request based on its resource requirements on a set of available BES-based computing endpoints. Even though UNICORE understands BES protocol, but still the execution service should know how to interpret, authenticate, and authorize the incoming GFFS Queue requests. A separate UNICORE server extension is implemented that is invoked when server finds a security token containing GFFS–related information in the incoming client request. The extension validates the SAML chain by looking into every element of the chain. These elements are entities (described in the previous section) which contain every stakeholder including end user or service through which the request was passed. The standards-based access and the validation of incoming requests required to trigger the data transfers serve the requirements R2-Openness and R3-Data transport. The user doesn't need to provide the actual physical location of the GFFS hosted data, instead she uses the symbolic RNS qualified hierarchical paths. This feature is inclined to support R5-Transparency.

Before a Genesis II client is able to send jobs to a UNICORE BES endpoint, a Genesis II container should recognize and link the UNICORE BES instance into the RNS space. The linking is achieved through the Endpoint Reference Minting process. An EPR (Endpoint Reference) is the basic component of the RNS. Every location in the GFFS namespace has an EPR that identifies (1) where the resource lives and (2) the X.509 certificate that represents the resource. Minting an EPR is the process of creating a new EPR as an XML document that represents an external resource, such as a UNICORE BES instance. The process of minting an EPR combines the URI where the resource is located with the X.509 certificate expected as the resource's identity (which it would report over a TLS connection). Once an EPR is minted, the EPR's XML document can be stored locally as a file or added as a new link to the grid namespace. When linked into the grid, a user with appropriate credentials can see the entry in the GFFS files system and can use it to obtain whatever services the resource provides.

The user's XSEDE identity is extracted from the delegation chain, and the retrieved identity is validated against the authorization store of the UNICORE server deployment. If the user is found under the authorization store then the required user context is created for carrying out GFFS data staging invocations. After the context creation phase, the server extension releases its control and job moves to the next phase of execution. In the beginning of the execution phase the request is processed further to carry out the GFFS-based data stagings of jobs. Figure 5.1 shows the job request encoded in JSDL containing application requirements and data staging elements pointing to the GFFS space. Note that the job will be executed on the UNICORE site, therefore Genesis II-BESes are not involved in this sequence.

The GFFS-based data transfer is realized through an XNJS extension. The GFFS download component prepares a command to pull data from the GFFS. The command (such as '`grid cp remote-source job-working-directory`') will be forwarded to the Target System Interface (TSI) that invokes the Genesis II '`grid`' command gracefully and downloads the data to the job's working directory. In a likewise manner the GFFS upload takes care of uploading output files to the remote GFFS location. Any failure will make UNICORE fail the job and abandon any further processing related to it. For every job which is sent by the Genesis II client UNICORE server extension maintains an additional folder in each of the job's working directory that contain user's contextual information.

The pictorial representation of a simple job submission sequence is shown in Figure 5.1. In the first step, the Genesis II GUI client asks a user to log-in through an XSEDE provided credentials. After the authentication phase (step 2), she uploads data to the GFFS folder (step 3). In step 4 the user then submits a job request with application details, and location of the data to be downloaded. In a similar manner, output data paths are also specified. The user then selects a UNICORE endpoint and run the job. Steps 5 and 7 shows a submission of request to the TSI and the target batch system. As soon as the job has been submitted to the execution service, the client continuously monitors the job until it reaches to a terminal state. Once the job is finished successfully the output is fetched back from the remote job working directory which is located on cluster's file system to the GFFS space. Steps 6 and 8 depicts the TSI and the GFFS interaction. For the sake of brevity only a sequence of major steps are being highlighted.

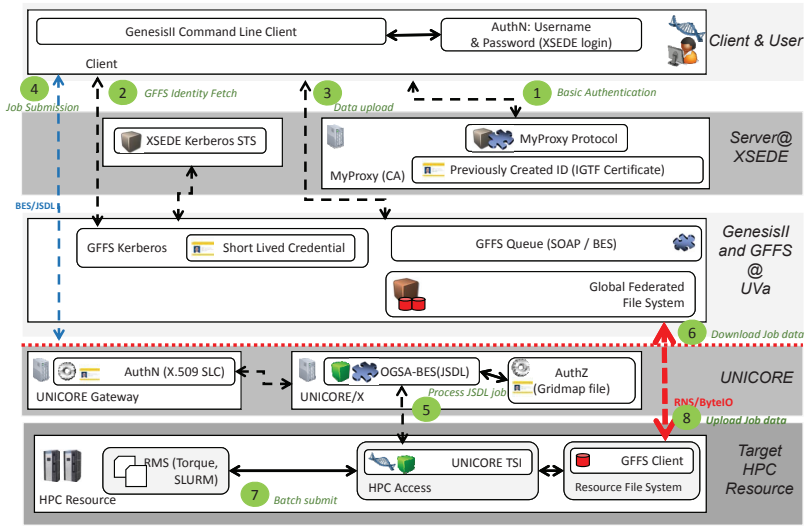


FIG. 5.1. *The UNICORE and the GFFS integration showing a job submission sequence with data staging.*

Another building block for supporting semi-automated data analytics is to allow user running jobs of parametric nature. Specially, the use case presented in the next section needs multiple runs required to identify optimal set of application parame-

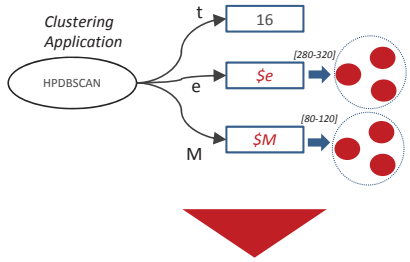
ters. By using the JSDL Parameter Sweep extension implementation of UNICORE [32] users can run multiple jobs as a single request. This is a very useful feature that has a positive impact on the overall data analysis life cycle. Considering, if manually running many jobs and compiling resultant outputs by hand, it will take user's considerable time just for book keeping the previous and next set of runs. We experimented by sending parametric jobs via Genesis II client and UNICORE BES implementation, and compared this model with manual SSH based submissions. By following this approach the user's administrative and usability overhead has significantly reduced. Figure 6.1 shows the snapshot of the used JSDL Parameter Sweep instance for the application.

6. Usecase: Point Cloud Anomaly Detection. Anomaly or outlier detection algorithms primarily identify a set of data points that appear to be different than the remaining data. There are different data clustering approaches which help data scientists to discover anomalies and a meaningful set of clusters from data. Several methods exist, for instance K-Means and Agglomerative clustering, have been used in commercial and scientific domains.

In this paper we place our focus on the DBSCAN [21]] algorithm which allows to reduce noise factor of the 3D point cloud dataset. A point cloud dataset captures objects in three dimensional space representing the external surface of objects by a point cloud. In our case, we use a data set that contains a point cloud for landscape elements, such as different kind of buildings, monuments or bridges, of the city of Bremen, Germany. This points cloud has approximately 81 million data points. We use DBSCAN to detect outliers in particular noise artefacts produced by the 3D scanner when recording the 3D point cloud. In practice if the dataset is processed using serial algorithm, it may take a couple of days. Therefore, it is imperative to have a parallel implementation of DBSCAN, which not only improves the performance, but also uses storage and memory requirements in an efficient manner. Another requirement is to have an implementation that adequately exploits execution environments of HPC. In order to support the application, HPDBSCAN [27] implementation is used. It is an initiative of Juelich that provides parallel implementation of DBSCAN. For efficient data storage and access, it uses the HDF5 data format.

We deployed this application on XSEDE infrastructure. We specifically used the BlackLight cluster that is deployed at Pittsburgh Supercomputing Center (PSC). A UNICORE service instance has been deployed and linked with the XSEDE-wide GFFS. Before the job execution, the dataset is placed on the GFFS node at the Indiana University's Mason cluster. The data staging was done by using the UNICORE's GFFS extension that copies data from the file system space to the local job's working directory.

The identification of anomalies from the point cloud dataset is the main objective of the clustering application. This requires to find an optimal set of application arguments: MinPoints M, and epsilon (also called radius) e which influence the clustering of new point cloud instances or different variants of the same data, respectively. In terms of data mining this phase is called post-processing. The discovery of optimal arguments is achieved by analyzing each of the completed job's output which contains the cluster distribution and noise factor. Within the output, the criterion is to select the job configuration containing the minimum noise factor combined with the best cluster distribution. The whole process of optimization requires multiple manual runs of the same application but with different M and e values. In order to avoid that users need to run these multiple jobs manually, the extended JSDL Parame-




```
<sweep:Sweep>
    <sweep:DocumentNode> ..
    <sweep:Match>//jsdl-posix:Argument[$e]</sweep:Match> </sweep:DocumentNode>
    <sweepfunc:LoopInteger start = "280" end="320" step="10">
    <sweep:Sweep> ..
     <sweep:DocumentNode> ..
      <sweep:Match>//jsdl-posix:Argument[$M]</sweep:Match> </sweep:DocumentNode>
      <sweepfunc:LoopInteger start = "80" end="120" step="10">
    </sweep:Sweep>
  </sweep:Sweep>
```

FIG. 6.1. *HPDBSCAN representation in the JSDL Parameter Sweep format depicting application with arguments: epsilon (e) and MinPoints (M) sweeping through a range of values.*

Access Mode / Execution Phase	Data Transfer	Data Processing	Post Processing
Manual	SCP, GridFTP, ByteIO, FTP	Job script for every different resource and batch system	Create script manually for every variation
Middleware (UNICORE & the GFFS)	Automated through the supported data transfer protocols	One JSDL instance for all kind of backends	Single JSDL-PS template for the specified parametric variations

TABLE 6.1
The user perspective of the Data analysis lifecycle using manual and automated mechanisms.

ter Sweep implementation which is provided by UNICORE's execution back-end was used. This allows using just a single job request which is not only more convenient for the user, but also faster, reproducible and less error-prone. The parameter sweep implementation serves the requirement R6-Parameter Sweep. Even though the user submits only one job, multiple child jobs are automatically generated according to the number of parameter iterations and nested sweeps. Figure 6.1 shows the sample HPDBSCAN JSDL job description making use of the parameter. The sweep factor of epsilon (e) and MinPoints (M) shown in Figure 6.1 will spawn 25 jobs in total with each generating a separate output.

Table 6 summarize the steps user need to perform data analysis in manual (script-based) and middleware-hosted environment. It is also evident from the illustration that the use of JSDL and JSDL-PS is more intuitive and avoids a need to write custom job requests for each flavor of the target resource management system. Furthermore, the data transfer event here applies to the pre-execution and fetch outputs phase.

7. Related Work. In this section we present the related job execution middleware technologies which are integrated with distributed file systems as well as work related to DBSCAN.

GridFTP [31] is one of the major data transfer protocols used in today's scientific and commercial data infrastructures. Specifically, GlobusOnline [25] data transfer service is mainly using this protocol to move data across widely distributed end-

points. UNICORE's GridFTP extension [7] helps scientists to submit job executions on UNICORE and using GridFTP-based data endpoints for data stagings. From the implementation perspective, both the GridFTP and GFFS extensions are integrated following the same approach, that is by using the XNJS programmatic interfaces. In the case of GridFTP integration, the clients requiring UNICORE servers to perform data staging on user's behalf, need to send at least X.509 proxy certificate chain along with the job submission request. For the GFFS the entities communicating the job request to the server must send a set of SAML assertions.

The GlobusOnline [25] service is a web portal to help end users perform GridFTP based high performance data transfers across different data endpoints. From the data management aspect, GlobusOnline and the GFFS are sharing a common set of features. By bridging the data access and processing (i.e. the job submission and execution, and execution service mount-point) services simultaneously distinguish the GFFS from GlobusOnline. The processing part is capable of attaching high performance (GenesisII) and high throughput computing (UNICORE) in a standard way. A very positive aspect of GlobusOnline is usability as it offers a ready to use data transfer service through a common web brower, whereas the GFFS user interaction is native desktop-based, which is not very intuitive and responsive as compare to browser-based applications.

ARC [20] is a middleware suite used by high throughput computing communities. ARC's integration with [26] and DDM (Distributed Data Management) [13] solutions are mostly used by the ATLAS [13] particle physics community at the Large Hadron Collider. dCache and DDM are distributed data management platforms providing storage and retrieval of huge amounts of data. dCache and the GFFS share mostly the same set of scenarios, but the major difference is that the GFFS expose its interface via RNS and ByteIO, whereas dCache is accessed through the SRM [9] interface.

WS-PGRADE / gUSE (grid and cloud user support environment) [30] is an open source scientific gateway framework that allows access to heterogeneous grid and cloud resources. gUSE provides a client extension in the form of DCI bridge [1] to the GFFS by invoking Genesis II clients. It is much similar to the way UNICORE integrates the GFFS. The frame- work provides access to UNICORE and ARC job submission services through the OGSA-BES interface.

In the context of worfklow (e.g. Taverna [38], Kepler [11], etc.) enabling data mining methods on distributed computing infrastructures. Da Silva et al. [17] describe workflows with serial implementation of DBSCAN. According to our understanding their approach is not using the parallel DBSCAN implementation and in contrast to our approach that is intended for production usage in a high performance computing environment, the paper rather describes a research project than a production implementation.

PDSDBSCAN-D [39] is an implementation of DBSCAN, based on the MPI and OpenMP frameworks. According to [27] the HPDBSCAN application is more performant on various earth science data sets, among which the points cloud data is one. It performs better due to efficient pre-processing of spatial cells and use of density-based chunking to balance the local computation load on each node. Furthermore, HPDBSCAN uses the HDF5 [4] data format to store data and uses its library for achieving better parallel input and output performance.

8. Conclusion. In this paper, we have derived and implemented an integrated architecture which covers a set of requirements for providing transparent, secure and interoperable data processing tasks. Also provide these tasks access to the datasets

managed by the Genesis II's Global Federated File System (GFFS). This is mainly achieved by the technical integration of UNICORE and the GFFS. The most important requirements are: R1 expresses a need for a secure trust delegation model, but should be standards-based and extensible (R2). As part of the integration the GFFS uses the SAML-based profile provided by the UNICORE's execution service. On the other hand, we extended UNICORE's identity validation that understand the GFFS-based requests containing multi-chain delegation assertions. R2 is also fulfilled by having the standards-based job execution and data management interfaces through the OGSA-BES and RNS specifications, respectively. The ByteIO standard is used to manage the data transport, thus the functionality implements R3.

While jobs are managed by UNICORE execution services, its internal service implementation is taking care of any status update delays through time out based probes against the target resource management system. In addition to that, the execution service also handles gracefully if the parallel file system on which the job's working data is stored becomes temporarily unresponsive, quite normal in production environment. The requirement R5 is served in this case.

For the Extensible resource and job model requirement R7, the resource model of the OGSA-Basic Execution Service (BES) and Job Submission and Description Language (JSDL) standards are extensible. But it will be only helpful if the compliant implementations are with minimal effort supporting the standard-allowed extensions. The technologies in our focus, UNICORE and the GFFS, are providing server and client side APIs to easily extend the resource model. This feature will be much more useful for community specific science gateways and next generation infrastructures with varying requirements. The XSEDE infrastructure has been used to demonstrate our implementation and data analysis excursion. This would require any technology and users entering the domain of an infrastructure should abide by its security model and its policies. With the GFFS client and UNICORE-based server, we used XSEDE-provided credentials to execute data processing jobs on a production deployment.

In our observation, most of the machine learning and data mining job submissions are parametric in nature, thus they need to be running multiple times. UNICORE's standard-based parameter sweep implementation helps to support our point cloud data clustering tasks. If we are able to represent the HPDBSCAN application requirement through JSDL and its parameter sweep extension, then any other data mining application can easily be supported. For the sake of implementation validity, we are analysing other methods of data mining, for example classification algorithms. One usability issue with the UNICORE's parametric sweep implementation is to produce a single job output based on some user specified criteria, which is currently not supported. The realization of this feature will reduce an overhead for data scientists to manually sort and merge the resultant job outputs. We intend to support this feature through a rule-based convergence of all the results from different parametric jobs into a single meaningful output. Another useful aspect is to avoid submitting multiple jobs to the batch system and rather use its internal feature of chaining multiple jobs. By enabling this feature the management and monitoring of complex job composites can be much more intuitive and usable.

REFERENCES

[1] *DCI Bridge Manual - v3.7.4*. http://sourceforge.net/projects/guse/files/3.7.4/Documentation/. [Online; accessed 29-Dec-2015].

[2] *FUSE (Filesystem in userspace)*. https://github.com/libfuse/libfuse. [Online; accessed 29-Dec-2015].
[3] *GNU Octave*. https://www.gnu.org/software/octave/. [Online; accessed 31-Dec-2015].
[4] *Hierarchical data format version 5*. http://www.hdfgroup.org/HDF5. [Online; accessed 29-Dec-2015].
[5] *The OASIS Web Services Interoperability (WS-I)*. http://www.ws-i.org/. [Online; accessed 29-Dec-2015].
[6] *TORQUE Resource Manager*. http://www.adaptivecomputing.com/products/open-source/torque-resource-manager/. [Online; accessed 31-Dec-2015].
[7] *UNICORE/X Manual*. https://www.unicore.eu/documentation/manuals/unicore6/files/unicorex/unicorex-manual.html. [Online; accessed 29-Dec-2015].
[8] A. Anjomshoaa et al., *Job Submission Description Language (JSDL) Specification, Version 1.0*. Open Grid Forum, GFD-R.136, July 2008.
[9] A. Sim et al., *The Storage Resource Manager Interface Specification, Version 2.2*. Open Grid Forum, GFD-R-P.129, May 2008.
[10] A. Streit et al., *Unicore 6 recent and future advancements*, Annals of Telecommunications, 65 (2010), pp. 757–762.
[11] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, *Kepler: an extensible system for design and execution of scientific workflows*, in Proceedings. 16th International Conference on Scientific and Statistical Database Management., June 2004, pp. 423–424.
[12] B. Dillaway et al., *HPC Basic Profile, Version 1.0*. Open Grid Forum, GFD-R-P.114, Aug 2007.
[13] G. Behrmann, D. Cameron, M. Ellert, J. Kleist, and A. Taga, *ATLAS DDM integration in ARC*, Journal of Physics: Conference Series, 119 (2008).
[14] K. Benedyczak, P. Bała, S. van den Berghe, R. Menday, and B. Schuller, *Key aspects of the UNICORE 6 security model*, Future Generation Computer Systems, 27 (2011), pp. 195 – 201.
[15] A. Bialecki, M. Cafarella, D. Cutting, and O. O'MALLEY, *Hadoop: a framework for running applications on large clusters built of commodity hardware*. http://hadoop.apache.org/. [Online; accessed 29-Dec-2015].
[16] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/. [Online; accessed 29-Dec-2015].
[17] R. F. da Silva, G. Juve, E. Deelman, T. Glatard, F. Desprez, D. Thain, B. Tovar, and M. Livny, *Toward fine-grained online task characteristics estimation in scientific workflows*, in Proceedings of the 8th Workshop on Workflows in Support of Large-Scale Science, WORKS '13, New York, NY, USA, 2013, ACM, pp. 58–67.
[18] J. Dean and S. Ghemawat, *Mapreduce: Simplified data processing on large clusters*, Commun. ACM, 51 (2008), pp. 107–113.
[19] B. Demuth, B. Schuller, S. Holl, J. Daivandy, A. Giesler, V. Huber, and S. Sild, *The unicore rich client: Facilitating the automated execution of scientific workflows*, 2013 IEEE 9th International Conference on e-Science, 0 (2010), pp. 238–245.
[20] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. L. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen, *Advanced resource connector middleware for lightweight computational grids*, Future Gener. Comput. Syst., 23 (2007), pp. 219–240.
[21] M. Ester, H. peter Kriegel, J. Sander, and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, AAAI Press, 1996, pp. 226–231.
[22] D. B. et al., *Web Services Addressing (WS-Addressing)*. http://www.w3.org/Submission/ws-addressing/. [Online; accessed 29-Dec-2015].
[23] M. D. et al., *JSDL Parameter Sweep Job Extension*. Open Grid Forum, GFD-R-P.149, May 2009.
[24] F. Bachmann et al., *XSEDE Architecture Level 3 Decomposition*, Dec 2012.
[25] I. Foster, *Globus online: Accelerating and democratizing science through cloud-based services*, IEEE Internet Computing, 15 (2011), pp. 70–73.
[26] P. Fuhrmann and V. Gülzow, *dCache, Storage System for the Future*, in Euro-Par 2006 Parallel Processing, W. Nagel, W. Walter, and W. Lehner, eds., vol. 4128 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 1106–1113.
[27] M. Götz, M. Richerzhagen, C. Bodenstein, G. Cavallaro, P. Glock, M. Riedel, and J. A. Benediktsson, *On Scalable Data Mining Techniques for Earth Science*, Procedia Computer Science, 51 (2015), pp. 2188–2197.

[28] A. Grimshaw, M. Morgan, and A. Kalyanaraman, *GFFS - The XSEDE Global Federated File System*, Parallel Processing Letters, 23 (2013), p. 1340005.
[29] I. Foster et al., *OGSA Basic Execution Service (BES), Version 1.0*, Nov 2008.
[30] P. Kacsuk, Z. Farkas, M. Kozlovszky, G. Hermann, A. Balasko, K. Karoczkai, and I. Marton, *WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities*, Journal of Grid Computing, 10 (2012), pp. 601–630.
[31] I. Mandrichenko, W. Allcock, and T. Perelmutov, *GridFTP v2 Protocol Description*. Open Grid Forum, GFD-R-P.047, May 2005.
[32] S. Memon, S. Holl, B. Schuller, M. Riedel, and A. Grimshaw, *Enhancing the performance of scientific workflow execution in e-science environments by harnessing the standards based parameter sweep model*, in Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery, XSEDE '13, New York, NY, USA, 2013, ACM, pp. 56:1–56:7.
[33] S. Memon, M. Riedel, F. Janetzko, B. Demeler, G. Gorbet, S. Marru, A. Grimshaw, L. Gunathilake, R. Singh, N. Attig, and T. Lippert, *Advancements of the ultrascan scientific gateway for open standards-based cyberinfrastructures*, Concurrency and Computation: Practice and Experience, 26 (2014), pp. 2280–2291.
[34] S. Memon, M. Riedel, C. Koeritz, and A. Grimshaw, *Interoperable job execution and data access through unicore and the global federated file system*, in Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on, May 2015, pp. 269–274.
[35] M. Morgan, *ByteIO Specification 1.0*. Open Grid Forum, GFD-R-P.87, Oct 2006.
[36] M. Morgan, A. Grimshaw, and O. Tatebe, *RNS Specification 1.1*. Open Grid Forum, GFD-R-P.171, December 2010.
[37] M. Morgan and O. Tatebe, *RNS 1.1 OGSA WSRF Basic Profile Rendering 1.0*. Open Grid Forum, GWD-R.172, December 2010.
[38] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, *Taverna: lessons in creating a workflow environment for the life sciences*, Concurrency and Computation: Practice and Experience, 18 (2006), pp. 1067–1100.
[39] M. A. Patwary, D. Palsetia, A. Agrawal, W.-k. Liao, F. Manne, and A. Choudhary, *A New Scalable Parallel DBSCAN Algorithm Using the Disjoint-set Data Structure*, in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, Los Alamitos, CA, USA, 2012, IEEE Computer Society Press, pp. 62:1–62:11.
[40] M. Riedel, M. Goetz, M. Richerzhagen, P. Glock, C. Bodenstein, A. Memon, and M. Memon, *Scalable and parallel machine learning algorithms for statistical data mining - practice amp; experience*, in Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on, May 2015, pp. 204–209.
[41] S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS Standard saml-core-2.0-os, March 2005.
[42] S. Graham et al., *Web Services Resource 1.2 (WS-Resource)*, April 2006.
[43] B. Schuller, R. Menday, and A. Streit, *A versatile execution management system for next-generation unicore grids*, in Euro-Par Workshops, 2006, pp. 195–204.
[44] G. Wasson and M. Humphrey, *HPC File Staging Profile, Version 1.0*. Open Grid Forum, GFD-R-P.135, Jun 2008.
[45] A. B. Yoo, M. A. Jette, and M. Grondona, *Slurm: Simple linux utility for resource management*, in Job Scheduling Strategies for Parallel Processing, Springer, 2003, pp. 44–60.

Paper II

Advancements of the UltraScan Scientific Gateway for Open Standards-based Cyberinfrastructures

M. S. Memon, M. Riedel, F. Janetzko, B. Demeler, G. Gorbet, S. Marru, A. Grimshaw, L. Gunathilake, R. Singh, N. Attig, and T. Lippert.

Journal of Concurrency and Computation, Vol 26 Issue 13, 2280–2291 (2014).

M. S. Memon has actively participated in bridging the UltraScan science gateway, UNICORE and Airavata to one architectural landscape. He is the main author of this publication. He outlined the paper structure and produced majority of the sections of this publication.

CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2014; **26**:2280–2291
Published online 13 March 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3251

SPECIAL ISSUE PAPER

Advancements of the UltraScan scientific gateway for open standards-based cyberinfrastructures

Shahbaz Memon[1,5,*,†], Morris Riedel[1,5], Florian Janetzko[1], Borries Demeler[2], Gary Gorbet[2], Suresh Marru[3], Andrew Grimshaw[4], Lahiru Gunathilake[3], Raminder Singh[3], Norbert Attig[1] and Thomas Lippert[1]

[1]*Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany*
[2]*The University of Texas Health, Science Center at San Antonio, San Antonio, Texas, USA*
[3]*Indiana University Bloomington, Bloomington, Indiana, USA*
[4]*Department of Computer Science, University of Virginia, Virginia USA*
[5]*School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland*

SUMMARY

The UltraScan data analysis application is a software package that is able to take advantage of computational resources in order to support the interpretation of analytical ultracentrifugation experiments. Since 2006, the UltraScan scientific gateway has been used with Web browsers in TeraGrid by scientists studying the solution properties of biological and synthetic molecules. UltraScan supports its users with a scientific gateway in order to leverage the power of supercomputing. In this contribution, we will focus on several advancements of the UltraScan scientific gateway architecture with a standardized job management while retaining its lightweight design and end user interaction experience. This paper also presents insights into a production deployment of UltraScan in Europe. The approach is based on open standards with respect to job management and submissions to the Extreme Science and Engineering Discovery Environment in the USA and to similar infrastructures in Europe such as the European Grid Infrastructure or the Partnership for Advanced Computing in Europe (PRACE). Our implementation takes advantage of the Apache Airavata framework for scientific gateways that lays the foundation for easy integration into several other scientific gateways. Copyright © 2014 John Wiley & Sons, Ltd.

Received 10 November 2013; Revised 31 January 2014; Accepted 4 February 2014

KEY WORDS: scientific gateways; UltraScan; UNICORE; Apache Airavata; standards

1. INTRODUCTION

Scientific gateways have emerged as a lightweight access layer on top of complex middleware setups that typically include different security paradigms and heterogeneous computational resources. They provide simple access to remote resources and permit scientists to focus on their research questions instead of technical low-level configuration details. Scientific gateways therefore have become an indispensable tool for domain-specific scientists and enable various benefits. Examples of these benefits include minimizing the amount of errors from manual inputs by checking input parameters, or by just offering correct combinations of input configuration parameters and their relationships that actually make sense from the science perspective as convenient options in the GUI. As a consequence, scientific gateways have been implemented in a broad range of scientific disciplines, and their use is even expected to grow in various scientific domains. The focus of this

*Correspondence to: Shahbaz Memon, Forschungszentrum Jülich, 52425 Jülich, Germany.
†E-mail: m.memon@fz-juelich.de

contribution is a scientific gateway known as the UltraScan Laboratory Information Management System (US-LIMS), which is used worldwide, primarily in biochemistry, material science, and polymer science. Its underlying UltraScan data analysis application [1, 2] is a software package that enables the use of computational resources in order to support the interpretation of analytical ultracentrifugation (AUC) experiments. The UltraScan scientific gateway was used in TeraGrid, the predecessor of Extreme Science and Engineering Discovery Environment (XSEDE), by scientists studying the solution properties of biological and synthetic molecules. The job management was traditionally limited to proprietary protocols based on one specific underlying middleware known as Globus [3]. At the same time, we observe an increased usage of middleware systems that offer open standard interfaces (e.g., UNICORE [4], GENESIS [5], and GridSAM [6]), because of two major reasons. First, open standard interfaces avoid vendor-locks, thus offering resource providers the possibility to change their systems, if needed, without requiring a change in the end user open standard-based client setup. Second, the open standards in the distributed computing domain have become extremely stable in the last couple of years after already having been intensively used in various scientific domains, which increases the trust in adopting them in different technologies.

In this paper, we provide insights about advancements of the UltraScan scientific gateway architecture to leverage open standard-based interface to supercomputing for extracting very high-resolution information from the experimental data. This contribution will focus on several improvements of the UltraScan scientific gateway that enable a standardized job management while retaining its lightweight design in order to not disturb the established user experience of researchers that use UltraScan in daily science. The current US-LIMS scientific gateway, which is used to access remote computational resources with these interfaces, has been in constant use since 2006 by several hundred users worldwide, with computational resources available from TeraGrid, XSEDE, Universities in the USA and Australia, as well as one commercial site in the USA. This system is stable and mature and undergoes only very minor changes in the user interface that contributes to its maturity.

The paper is structured as follows. After the introduction in Section 1, the rationale behind our work is given in Section 2, explaining the scientific case underlying UltraScan, and providing the context for the relevance of scientific gateways. Section 3 lists the identified limitations of the UltraScan scientific gateway. Section 4 then describes how we overcome the identified limitations of the framework with various architectural improvements. One concrete deployment of our proposed integrated architecture is presented in Section 5 with insights into its use in a production setup. Section 6 will bring out significant experiences acquired during the course of gateway integration and application enabling. After presenting related work in Section 7, this paper ends with some concluding remarks.

2. RESEARCHERS USING ULTRASCAN

Researchers wishing to perform computationally intensive analysis of ultracentrifugation data on high performance computing (HPC) resources have long been able to use a gateway that is highly integrated with the UltraScan application and databases. This section gives an overview on the UltraScan application and further reveals an interaction between the gateway and researchers.

Since the early 1990s, digital data acquisition from the analytical ultracentrifuge laid the foundation to analyze sedimentation data using powerful computational resources. The UltraScan data analysis application [1, 2] became a well-known multi-platform software package that is able to take advantage of HPC resources. It supports the interpretation of complex, high-resolution analytical ultracentrifugation (AUC) experiments in various application domains. These include biophysics, biomedicine, material science, nanotechnology, pharmaceutics, and other industrial applications. UltraScan not only provides sophisticated parallelized optimization algorithms but also offers guidance with the design of sedimentation experiments, addresses data management challenges that arise from large data sets, and provides visualization routines to assist with result interpretation. UltraScan is organized into five modules: (1) a parallelized application using the message passing interface (MPI) for solving optimization problems on massively parallel compute clusters. This application includes a finite element solver for the Lamm equation [7] and implements

DOI: 10.1002/cpe

three fundamental optimization algorithms, the two-dimensional spectrum analysis [8] for finding the size and anisotropy distributions, genetic algorithms for parsimonious regularization [9], and a Monte Carlo implementation to identify confidence regions and provide parameter statistics [10]. Other modules include (2) a relational database for data management and result storage and (3) a laboratory information management system (LIMS), which is available through the XSEDE Science Gateways [11]. It serves the user with a common interface for the collaboration with other users, formulates all HPC analysis job requests, and sends them to a remote cluster for analysis [12]. Module (4) integrates grid middleware functionality for job abstraction that is based on Airavata [13] and manages submitted computational UltraScan jobs. The final module (5) provides a multi-platform GUI to assist with interpretation and visualization of analysis results. The UltraScan science gateway hides the arcane tedium of submitting supercomputer jobs from the user, and thereby makes the UltraScan software package user-friendly, and avails its data analysis methods to a broad community. As a result of these designs, the utility of the software was continously improved and was more accessible to novice users. The UltraScan science gateway is used worldwide, serving approximately 800 academic and industrial users today.

In more detail, a unique component of the UltraScan software is a PHP web interface that allows the user to interact with their data in the MySQL database and to submit data for analysis to a remote cluster. The LIMS system also offers an extensive reporting system that dynamically generates analysis reports based on available datasets in the database and allows the user to share their data with colleagues over the Web to facilitate collaborations. Job submissions can be conveniently tracked through the LIMS queue viewer that updates the user on the status of all active UltraScan jobs. Multiple analysis methods can be selected by researchers for submission to a remote compute resource, and individual parameters for each analysis method (e.g., fitting ranges, number of Monte Carlo iterations, genetic algorithm parameters, and others) can be conveniently adjusted through the Web interface. An administrator function is available for tracking job submissions and reviewing standard error and output files from each job, and providing experimental project information, and uploading ancillary and supporting data needed for the analysis. The US-LIMS system takes advantage of the Apache Airavata system, which is further described in the succeeding text, but largely responsible for the job staging on the requested computational resources. The US-LIMS system and its interfaces are illustrated in Figure 1.

Researchers use the tool by logging into their LIMS Web instance (using the underlying Apache Web server) and selecting one or more datasets from the MySQL database for analysis. After selecting the analysis method and setting up the parameters for the analysis, the researcher selects a computing resource and submits the job. At this point, a unique identifier is created in a second

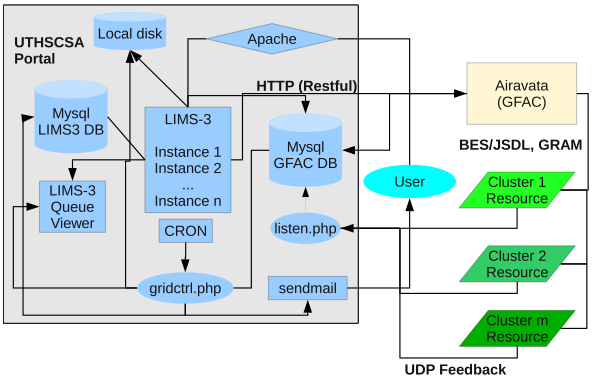


Figure 1. The UltraScan-LIMS science gateway architecture with its entities and relationships to computing resources.

Concurrency Computat.: Pract. Exper. 2014; **26**:2280–2291
DOI: 10.1002/cpe

database called Generic Application Factory (GFAC) that tracks all active jobs. The data are extracted from the database, and an XML file containing the job parameters is written to a local disk. A tar archive containing the experimental data is also created and placed into the GFAC database, and then the job is forwarded to the Apache Airavata system. The job request is issued for the targeted resource, using the abstractions provided through GFAC. The GFAC database is now continually being updated with information from the Airavata software. A grid daemon (gridctrl.php), controlled by crontab, running on a server at UTHSCSA monitors every job sent from any LIMS instance by periodically querying the GFAC database and updating the information in the LIMS queue viewer. Any cluster can also send messages over UDP to a specified port on the UTHSCSA server, which is read by the listen.php daemon running on the UTHSCSA server. This UDP stream contains information about the job status known only to the MPI master process in the UltraScan parallel application. The information includes details about iteration numbers and parameter positions. All such information are recorded in the GFAC database and also updated in the queue viewer by the gridctrl.php daemon. When the job is finished, the status in the database is updated to signal that the job has completed and all results have been transferred to the GFAC database. At this point, gridctrl.php updates the queue viewer, retrieves all job information from the GFAC database, and places it into the LIMS/MySQL database. Finally, an email is generated with the result code from the job, and the researcher is notified. The aforementioned process is using entities illustrated in Figure 1.

3. IDENTIFIED LIMITATIONS

In order to understand the two identified key limitations of the approach described in the previous section, we explain several entities of Figure 1. The latest US-LIMS gateway uses the Apache Airavata [13] software framework to access a variety of computational resources. As shown in Figure 1, the GFAC component within Airavata wraps command-line driven science applications and turns them into robust grid middleware-agnostic services. Airavata has a pluggable architecture that supports submission to different middleware systems using their corresponding proprietary protocols (e.g., using Globus [3]). We identify this direct vendor dependence of Airavata as limitation A inherent in the previous design that also affects the end user by being limited to those infrastructures that offer the corresponding specific proprietary interface as access method. The framework also provides access through Secure Shell (SSH) to remote machines, Amazon EC2, or clusters using Apache Hadoop. All of these access methods are mostly based on proprietary protocols to a broad set of different technologies or are very basic (e.g., access via SSH) requiring manual intervention by researchers. A deeper analysis of the Apache Airavata software reveals that it is designed on service-oriented architecture principles, which allow abstraction, extension, and component encapsulation. As illustrated in Figure 2 (i), different plugins (e.g., JGlobus, EC2 API, etc.) can be added to Airavata to authenticate, authorize, move data, submit jobs, and monitor progress. All of them need to be maintained separately. We identify this maintenance overhead of n technologies as limitation B, that is, especially problematic when you consider m different versions of the same technology deployed across infrastructures over time (i.e., maintenance of up to nxm different technologies). Apache Airavata is comprised of GFAC and several other components [13] to provide science gateway capabilities. GFAC provides a generic framework to wrap an application as a service interface in Java. This service layer separates the invocation from the communication layer supporting multiple protocols like SOAP, REST, or JSON. The application provider first describes the application through inputs, outputs, deployment information, temporary working directories, and remote access mechanisms for file transfers and job submissions, and registers this information with a registry service. Once applications are registered, the GFAC's distributed application management handles the file staging, job submission, and security protocols associated with executions. During execution, the application schedule is determined, and the input data files specified by input parameters are staged to the computational resource. The underlying application is then executed using a job submission mechanism. The framework monitors the status of the remote application and periodically publishes activity information to a so-called event bus. Once the invocation is complete, the

DOI: 10.1002/cpe

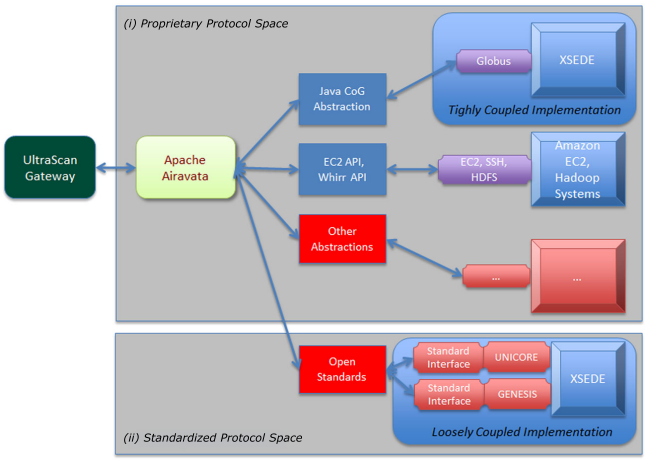


Figure 2. Overview of limitations by providing proprietary protocols (top) instead of open standards (bottom).

application service tries to access the results of the application invocation by searching the standard output for user-defined patterns or by listing a prespecified location for generated data.

Based on the aforementioned details, we are able to identify the limitations in Figure 2 (i). A direct comparison with identified solutions to overcome the limitations is shown in Figure 2 (ii). Our standard-based client integration into Apache Airavata offer a mature and stable resource access interface and overcomes the limitation A as standard interfaces evolve slowly. As part of this contribution, we integrated a client that enables standard-based grid middleware (e.g., UNICORE [4] or GENESIS [5]). This solution also overcomes the limitation B by avoiding vendor-locks and is thus not limited to use only infrastructures with Airavata deployment having one particular proprietary interface installed. To sum up, apart from saving maintenance efforts (identified as limitation A) by just providing one open standards adapter, the adoption of the open standards protocols also provides access to a diverse range of computational resources offered by multiple infrastructures worldwide.

4. ARCHITECTURAL ADVANCES

This section provides an introduction to the relevant open standard protocols used to improve the architecture analyzed in the previous section, gives an overview on UNICORE middleware, and describes our architectural advances.

4.1. Relevant open standard protocols

There are three relevant open standards in the field that are of relevance to the Apache Airavata's standard-based extensions presented in Section 5. These are named as the Job Submission and Description Language (JSDL) [14], ByteIO [15], and the Open Grid Services Architecture Basic Execution Service (OGSA-BES) [16] specification (often named BES for simplicity). These standards are well defined and although being very basic in nature, they are extremely mature and stable.

Basic Execution Service provides a Web services based interface and a state model and information schema to represent job management requirements. The major interface operations generally include create, monitor, and terminate actions for computational jobs. The job state machine defined

DOI: 10.1002/cpe

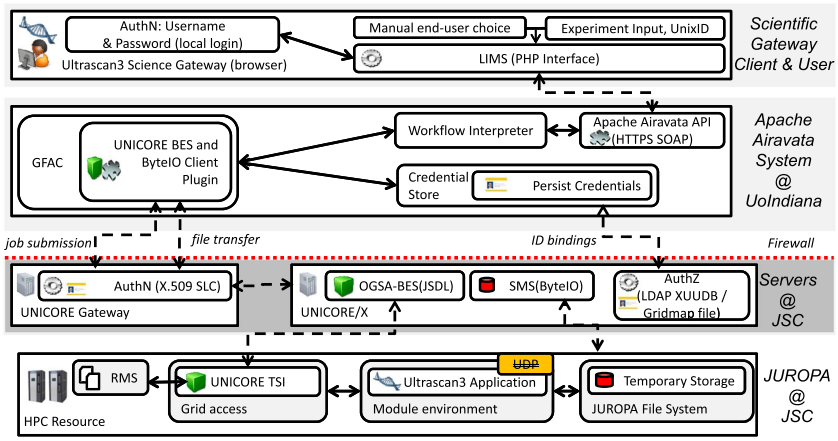


Figure 3. An adoption of open standards in the overall UltraScan integrated architecture.

in BES is an extensible model depicting complete life cycle of the job. The computing resource capabilities can be expressed through the information schema.

Job Submission and Description Language is an Open Grid Forum standard that defines clear syntax and semantics for describing the requirements of scientific applications that are intended to execute via a middleware abstraction on computational platforms. Applications invoked via the aforementioned BES interface must be specified in JSDL. The JSDL specification comes with several profiles such as the single-program-multiple-data (SPMD) extension [17] and the JSDL parameter sweep extension [18] that offer standard abstractions projecting common functionality required by researchers on higher-levels (e.g., parameter sweeps). In the context of this contribution, the JSDL SPMD is used by an UltraScan client to express job submission requests. This extension allows the client to specify a number of processes and processes per host required by the job. Section 5.1 further elaborates the usage of JSDL SPMD for representing the UltraScan application requirements. The deployment presented in Figure 3 shows that data stagings to job locations are performed through the ByteIO standard. This protocol is already adopted in several middleware implementations such as within UNICORE, GenesisII [5], and OGSA-DAI [19]. The ByteIO specification consist of Web services based interfaces allowing data read and write in a stateful and stateless fashion.

4.2. Open standards-based middleware

In this section, we demonstrate the advantages of our approach by describing one middleware system (UNICORE) that adopts the relevant standards described in the previous section. UNICORE is an open source, platform independent middleware abstraction for most of the production HPC systems. It is a service-oriented architecture providing most of its functionality through Web services. Many of these services are compliant with open computing, data, and security standards.

In more detail, UNICORE consists of a server containing multiple Web services and multiple clients, such as UNICORE Rich Client, a graphical client, and the UNICORE Command-line Client. The server tier adopts the OGSA-BES interface to enable standard-based job submission and management. The backend execution of the BES interface is performed by the Extended Network Job Supervisor (XNJS), which is an execution manager that processes an incoming job request. Through XNJS, the request is verified and validated against the JSDL compliance, and the resource requirements are matched against the computing site's resource capabilities, such as the number of compute nodes or processors per node. Once the request is successfully validated, it is then prepared

DOI: 10.1002/cpe

for submission to a remote target batch system, often also named as resource management system (e.g., Torque, Loadleveler, etc.). At the batch system tier, there is a separate component called Target System Interface (TSI). It is a tiny Perl based daemon process deployed on the login node of a batch system that takes the abstract submission request from XNJS and converts it to a batch system specific job command script.

4.3. Integrated architecture

Figure 3 illustrates the adoption of the aforementioned open standard protocols in the integrated architecture that includes the UltraScan scientific gateway and the Apache Airavata framework. It illustrates the standardized access via grid middleware to computational resources in production systems that are similar to those in infrastructures such as XSEDE or PRACE. The integrated architecture overcomes the limitations identified in the previous sections by stable standard protocols that do not change often.

The benefits of this integrated architecture are twofold. First, the BES and JSDL Client Classes Plugin of GFAC enable the standardized job submission to any standard-compliant middleware that adopts these standards. Examples of grid middleware that offer an OGSA-BES (including JSDL) compliant access include UNICORE, GENESIS, and GridSAM [6]. They ensure a stable interface for job management, independent of the underlying grid middleware services, which can be exchanged, when needed, while the Airavata framework implementation stays the same. More benefits in this context can be found in our previous work on open standard reference architectures [20].

The second benefit for the Airavata framework as a whole is the lowered maintenance overhead by just managing one standard-based plugin instead of multiple proprietary plugins. This is particularly crucial in e-Science production infrastructure deployments where separate plugin is desired for different middleware versions. There is certainly more administrative and community overhead when new version comes out which might require client and server update at the infrastructure scale.

5. CONCRETE DEPLOYMENT

This section describes a concrete production deployment of our proposed architecture advancements.

5.1. Hardware and software environment

For the application setup and job runs the Juelich Supercomputing Centre's JUROPA cluster [21] had been used. The UltraScan software package was installed on the JUROPA system. This application is MPI parallelized and uses the Qt4 library. In order to ease the usage of UltraScan and to avoid the necessity to manually set the correct environment, it is accessible as a Linux environment module. The UNICORE server is configured to trust UltraScan users and also exports the UltraScan application details, which includes executable name and the environment variables required during the runtime of the application. Figure 4 refers to the related JSDL-SPMD compliant instance used as a job execution request token to the UNICORE deployment on JUROPA. *us_mpi_analysis* is the target executable of the job run. Because the application is MPI-based, the SPMD describes this information by leveraging the SPMDVariation element, which indicates what MPI runtime flavor should be used.

5.2. Job management advancements

In the previous section, we described our general approach of the integrated architecture, whereas this section focuses on one specific job submission and management flow to illustrate the feasibility and benefits of using open standards. Prior to job submission on JUROPA, it is assumed that a user is known to a UNICORE deployment and has enough allocation to acquire compute time on the cluster. In the first step, the end user needs to bind his personal identity with the credential service within the Apache Airavata system using the X.509 certificate of the end user. Hence, during daily

DOI: 10.1002/cpe

```
<spmd:SPMDApplication>
 <spmd:NumberOfProcesses>256</spmd:NumberOfProcesses>
 <posix:Executable>us_mpi_analysis</posix:Executable>
 <posix:Output>stdout</posix:Output>
 <posix:Error>stderr</posix:Error>
 <posix:Argument>hpcinput.tar</posix:Argument>
 <spmd:SPMDVariation>
    http://www.ogf.org/jsdl/2007/02/jsdl-spmd/MPI
 </spmd:SPMDVariation>
</spmd:SPMDApplication>
```

Figure 4. UltraScan job as a Job Submission and Description Language (JSDL) single-program-multiple-data (SPMD) instance.

operation, the end user will start by presenting her username and password to authenticate at the UltraScan3 scientific gateway (local login). After successful authentication, the end user configures the corresponding computational job with a choice of instrument data inputs and user information associated with a grant and analysis parameters in the US-LIMS. The Apache Airavata API communicates the job to the framework in which it is specifically received by the workflow interpreter that analyzes the job request and forwards it to the GFAC component.

Before the job request is prepared for the target middleware service (i.e., UNICORE in our case), GFAC generates the right credential from the user information, which was communicated previously through US-LIMS. These credentials are persisted to the Airavata's Credential Store service. Next, the request can proceed job submission. At this stage, we take advantage of the solutions proposed in Section 3. We implemented an open standard compliant plugin within Apache Airavata using BES and JSDL. Although UNICORE Client Classes have been used for the implementation, in principle any BES/JSDL client could have been used (e.g., GENESIS BES client or JSAGA, for example).

As part of the submission process, the job's temporary directory is created on the JUROPA's shared file system. The directory is a temporary location where the job input and output files are copied from the job's actual working directory. After the directory is created, any input files are uploaded to the input directory. ByteIO is used as a data transfer mechanism for transferring the job's input files. Once GFAC knows all the files are staged-in to the temporary location, the JSDL instance generation will take place by combining the UltraScan application parameters, executable name, and environment variables in a single JSDL instance. The generated instance will also contain names of the required input and output files.

The UNICORE BES/JSDL client plugin will then take this JSDL instance and fire job submission request to JUROPA's BES endpoint. After the successful submission takes place, the job status is continously monitored, unless it reports a finished state. The finished state implies that the JUROPA BES instance has copied all output files to the temporary directory. Once they are copied, the client plugin will be able to download all the output files to the location where GFAC is deployed. After the job outputs are fetched, GFAC uploads them to the gateway's desired location.

At the server side, UNICORE/X forwards the job request to the TSI [4]. The TSI interacts with the resource management system that is Moab/Torque in our case.

6. GATHERED EXPERIENCES

The section provides details on the benchmarks captured during the UltraScan job execution runs and also compiles a list of issues faced during the process of enabling the application on Airavata and UNICORE layers.

6.1. Job submission benchmarks

In order to illustrate our new open standard-based approach, we compare it with existing middleware deployments of UltraScan on XSEDE infrastructure. The observation is made on the basis of

DOI: 10.1002/cpe

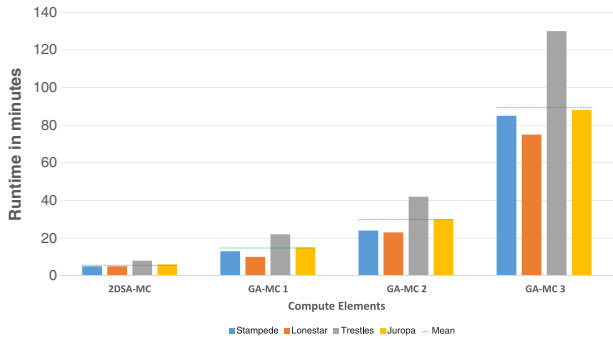


Figure 5. A graphical representation of the job runtimes captured from different production computing sites.

time-to-solution. Both general experience and a specific set of benchmarks show that throughput for Airavata initiated jobs for UNICORE middleware is comparable to that of the GRAM middleware. Figure 5 summarizes execution times for various job types on XSEDE's Stampede, Lonestar, Trestles, and Juelich's JUROPA supercomputing. All the jobs shown in this figure ran UltraScan with the same application version and runtime parameters. Figure 5 depicts job statistics showing the genetic algorithm - Monte Carlo (GA-MC) jobs average 80 min (1 h and 20 min). These jobs typically use 256 processors. The chart shows run times for four different jobs run on four different clusters. Three of the clusters (Stampede, Lonestar, and Trestles) are in the XSEDE system. The final one (JUROPA) is part of PRACE. The XSEDE clusters used GRAM, whereas the PRACE cluster used UNICORE as compute middleware.

The run times are greatly affected by the size of the data and by parameterization. The tests on different clusters used exactly the same data and parameterization. The relative run times are a fair comparison of processing speeds on the clusters. Further, the run times primarily reflect processor speed and memory available on each cluster rather than the middleware. The chart does illustrate that the JUROPA cluster has a speed that is near the average of all clusters and that its results (using Airavata and UNICORE) is comparable to the others. The four jobs run are all Monte Carlo runs of two types: 2DSA (two-dimensional spectrum analysis) and genetic algorithm (GA). 2DSA-MC typically run very quickly. A GA-MC job can take a long time. GA-MC1, GA-MC2, and GA-MC3 correspond to the same application but with different runtime application parameters, one such runtime parameter is a variation in the number of Monte Carlo iterations.

6.2. Integration experiences

Scientific gateways provide a seamless access to different computing insfrastructures via convenient user interfaces. They are mostly Web browser based interfaces that aggregate application requirements in response to user interaction. Users must be registered and properly authenticated before using any functionality of the gateway. The security here is enforced by a gateway portal rather than a target end resource used for computing. In this manner, the complexity of computing middleware access and site security intricacies are hidden behind the gateway portal. The scientific gateway as a community is solely responsible for arranging credentials on behalf of the user and carries out job submission. The US-based scientific UltraScan gateway is configured with a single community credential to function on behalf of multiple users. Because of legal and policy constraints, this was not possible on a European cluster, that is, JUROPA. In order to adhere to the policy requirements, the security model of UltraScan/Airavata and UNICORE integration has been revisited, and a special field in the US-LIMS Web interface is introduced that captures a user login (unixID on JUROPA) for every user who intends to target JUROPA.

Concurrency Computat.: Pract. Exper. 2014; **26**:2280–2291
DOI: 10.1002/cpe

In our experience, the file system availability appears to be a critical factor when hundreds of jobs are simultaneously reading and writing to a shared file system. JUROPA has a Lustre file system. We observed that 30% of jobs failed because UNICORE receives a delay in file system response. The response time varies from a few seconds to a minute. In order to have job submission be resilient to the file system delays, a time-out grace factor was introduced into the UNICORE server, which prevents job failure until a certain time interval has passed. After this configuration, we have experienced a negligible number of job failures because of file system delays.

As described in Section 4.3, TSI acts as a thin layer for submitting and monitoring UNICORE managed jobs to the MOAB scheduler. One observation shows during peak loads on JUROPA the underlying batch system does not return any status of the submitted job. This occurs even when the job is not completed. In this case, the UNICORE server reports those jobs for which the status is not shown as finished, because the middleware tier assumes the job is completed, even though it is still running. In several cases, a finished job status was incorrectly issued. To overcome this issue, we introduced a status update grace interval configuration into UNICORE, which enables it to wait until the actual job status can be retrieved. Introduction of this property has greatly reduced the incidence of false status reports in production UNICORE deployments.

7. RELATED WORK

There is a wide variety of related work but we focus on those that are of direct relevance to our approach. First, the Vine Toolkit [22] provides Java based adapters for various middleware technologies and represents a high-level API for managing jobs on specific target sites. It is optimized to run in browser-based setups that act as the client tool for the GridSphere portal. Vine supports not only gLite and UNICORE (e.g., SSL key generation, and job-based management, etc.) but also Virtual Organization Membership Service (VOMS) (e.g., proxies, registering and un-registering users, etc.). In order to send jobs to a middleware, the Vine Toolkit has dedicated client classes for each middleware. The Uniform Resource Identifier (URI) of this container and the corresponding gateway portal is typically at a fixed location that end users can use to perform their daily scientific application runs. In contrast to our approach, its several features include the use of proprietary UNICORE interface elements for job management.

The 'Simple API for Grid Applications' (SAGA) [23] is another concrete example of related work. SAGA has two notable characteristics. First, SAGA is an open standard [24] to promote 'Grid interoperability on the application level' developed by the Open Grid Forum. Second, the SAGA framework is actively developed by a team that adopts the SAGA standard as described in [23]. This framework is similar to the Airavata framework and enables a high-level programming abstraction, which significantly facilitates the development of scientific applications. As such, it provides a lot of application patterns (e.g., map-reduce, replication, parameter studies, etc. [25]) that are useful for end users. Apart from being a high-level application framework, [25] reveals that it can work on top of the open standards used in this contribution (i.e., OGSA-BES, JSDL, etc.). Despite being previously bounded to proprietary interfaces of middleware systems as described in [23] and [24], the SAGA framework has now been extended towards OGSA-BES. Hence, SAGA is not only a standard but also an application framework that provides many features that could be beneficial to the Apache Airavata framework. In the context of our work, this option might be a candidate for future integration once the BES in SAGA is properly tested and used by various stakeholders.

Closely related to scientific gateways is the P-Grade Portal [26]. This portal can also be considered as a building block for various scientific gateways and consists of many proprietary middleware adapters. P-Grade was not used, because UltraScan already takes advantage of the Apache Airavata framework for accessing middleware and we wanted to keep the changes at minimum to ensure stability.

8. CONCLUSIONS

The results of our pilot activities of enabling scientific gateways with open standard interfaces lead to a couple of interesting conclusions. We can conclude that using an abstraction framework within

DOI: 10.1002/cpe

the UltraScan scientific gateway is extremely useful for two reasons. First, it enables the UltraScan gateway to focus on its domain-specific functionality (i.e., US-LIMS, etc.). In other words, the source-code to access the wide variety of heterogeneous computational resources and middleware systems does not need to be directly included in the scientific gateway. Second, the Apache Airavata plugin for open standards is available through the Apache open community. As it is developed through an open source process, it can be reused within a wide variety of other scientific gateways. This lowers the maintenance overhead required by potentially many scientific gateways that all need to perform job submissions and management activities. In summary, our work holds the potential that the standard-based plugin within Apache Airavata will be reused many times by being integrated in various scientific gateways we already have started to analyze for possible integration (cf. to XSEDE scientific gateways). A major conclusion is the benefit of using open standard interfaces to access computational resources (e.g., such as JUROPA [21]). We have shown one particular detailed deployment in Europe based on the UNICORE middleware, but the approach can be applied to a wide variety of other resources that use standard-compliant middleware such as GENESIS and GridSAM. For example, any GENESIS installation on the FutureGrid testbed can be easily used by just changing the URI within the BES/JSDL client classes. This requires neither redevelopment of the plugin within Apache Airavata nor a tuning of the GENESIS BES/JSDL implementation. Also, the XSEDE architectural process includes the use of open standards such as BES, JSDL, and ByteIO while these interfaces already can be used to access European EGI and PRACE resources if needed. This represents a key benefit of using a standardized and stable interface approach. Furthermore, even if some middleware system will not be available in the future (perhaps because of lack of funding), the use of standardized interfaces will enable an easy switch to another middleware system that adopts the standard interface.

The enhancements discussed in this paper are integrated with production versions of the UltraScan gateway. Since September 2013, European users of the UltraScan gateway have been using our contributions and computational resources in Juelich. As a future work, we intend to use the integration effort as a basis for the UltraScan users of the XSEDE infrastructure. Once we have more than one UltraScan and BES deployments, we will perform interoperability tests. In summary, we conclude that our approach lowers risk and increases trust because it is based on standard interfaces that do not change as often as proprietary interfaces. Especially, for large-scale infrastructures such as XSEDE, PRACE, or EGI, it is important that the interface elements do not change. In the past, many proprietary interfaces changed their protocols often, which resulted in loss of trust in the grid middleware systems. Also, switching from one middleware version to another version is a very slow process in several infrastructures. That means that old versions of the proprietary interfaces within middleware systems are still supported in order to support end users that are not able to change their client setup or that are faced with resource providers that also do not want to frequently reinstall their middleware software. The use of stable standards reduce this change as BES or JSDL did not change in the last couple of years.

ACKNOWLEDGEMENTS

This work is partially funded by the National Science Foundation through the awards: OCI-1053575 (Extreme Science and Engineering Discovery (XSEDE)) and ACI-1032742 (SDCI NMI Improvement: Open Gateway Computing Environments - Tools for Cyberinfrastructure-Enabled Science and Education). The authors gratefully acknowledge the computing time granted by the John von Neumann Institute for Computing (NIC) and provided on the supercomputer JUROPA at Juelich Supercomputing Centre (JSC).

REFERENCES

1. Demeler B. UltraScan A Comprehensive Data Analysis Software Package for Analytical Ultracentrifugation Experiments. In *Modern Analytical Ultracentrifugation: Techniques and Methods*, Scott DJ, Harding SE, Rowe AJ (eds). The Royal Society of Chemistry: Cambridge CB4OWF, UK, 2005; 210–230. (Available from: http://dx.doi.org/10.1039/978184755).
2. UltraScan-III version 2.0: a comprehensive data analysis software package for analytical ultracentrifugation experiments. (Available from: http://www.utrascan3.uthscsa.edu/) [Accessed on 12 September 2013].

3. Foster I, Kesselman C. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications* 1996; **11**:115–128.
4. Streit A, Bala P, Beck-Ratzka A, Benedyczak K, Bergmann S, Breu R, Daivandy JM, Demuth B, Eifer A, Giesler A, Hagemeier B, Holl S, Huber V, Lamla N, Mallmann D, Memon AS, Memon MS, Rambadt M, Riedel M, Romberg M, Schuller B, Schlauch T, Schreiber A, Soddemann T, Ziegler W. UNICORE 6-recent and future advancements. *Annals of Telecommunications* 2010; **65**:757 –762.
5. Morgan MM, Grimshaw AS. Genesis II - standards based grid computing. *Seventh IEEE International Symposium on Cluster Computing and the Grid, 2007*, Rio de Janeiro - Brazil, 2007; 611–618.
6. Lee W, Mcgough AS, Darlington J. Performance Evaluation of the GridSAM Job Submission and Monitoring System. *In UK e-Science All Hands Meeting*, Nottingham UK, 2005; 915–922.
7. Cao W, Demeler B. Modeling analytical ultracentrifugation experiments with an adaptive space-time finite element solution of the Lamm equation. *Biophysical Journal* 2005; **89**(3):1589 –1602.
8. Brookes E, Cao W, Demeler B. A two-dimensional spectrum analysis for sedimentation velocity experiments of mixtures with heterogeneity in molecular weight and shape. *European Biophysics Journal* 2010; **39**(3):405–414.
9. Brookes EH, Demeler B. Parsimonious Regularization using Genetic Algorithms Applied to the Analysis of Analytical Ultracentrifugation Experiments. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation,* GECCO '07, New York, NY, USA, 2007; 361–368. ACM.
10. Demeler B, Brookes E. Monte Carlo analysis of sedimentation experiments. *Colloid and Polymer Science* 2008; **286**(2):129–137.
11. XSEDE Science Gateway Listing. (Available from: https://www.xsede.org/web/guest/gateways-listing) [Accessed on 15 September 2013].
12. Demeler B, Singh R, Pierce M, Brookes EH, Marru S, Dubbs B. UltraScan Gateway Enhancements: In Collaboration with TeraGrid Advanced User Support. *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery,* TG '11, New York, NY, USA, 2011; 34:1–34:8. ACM.
13. Marru S, Gunathilake L, Herath C, Tangchaisin P, Pierce M, Mattmann C, Singh R, Gunarathne T, Chinthaka E, Gardler R, Slominski A, Douma A, Perera S, Weerawarana S. Apache Airavata: a framework for distributed applications and computational workflows. *Proceedings of the 2011 ACM Workshop on Gateway Computing Environments,* GCE '11, New York, NY, USA, 2011; 21–28. ACM.
14. Anjomshoaa A, Brisard F, Drescher M, Fellows D, Ly A, McGough S, Pulsipher D, Savva A. Job Submission Description Language (JSDL), Version 1.0, July 2008.
15. Morgan M. ByteIO Specification 1.0, October 2006.
16. Foster I, Grimshaw A, Lane P, Lee W, Morgan M, Newhouse S, Pickles S, Pulsipher D, Smith C, Theimer M. OGSA Basic Execution Service (BES), Version 1.0, November 2008.
17. Savva A. JSDL SPMD Application Extension, 2007.
18. Drescher M, Anjomshoaa A, Williams G, Meredith D. JSDL Parameter Sweep Extension, 2009.
19. Jackson M, Antonioletti M, Dobrzelecki B, Hong N. Distributed data management with OGSA–DAI 2011:63–86.
20. Riedel M, Wolf F, Kranzlmüller D, Streit A, Lippert T. Research advances by using interoperable e-science infrastructures. *Cluster Computing* 2009; **12**(4):357–372.
21. *JUROPA*. (Available from: http://tinyurl.com/lf2r9e8) [Accessed on 30 January 2014].
22. Russell M, Dziubecki P, Grabowski P, Krysiński M, Kuczyński T, Szjenfeld D, Tarnawczyk D, Wolniewicz G, Nabrzyski J. The vine toolkit: A Java framework for developing grid applications. In *Parallel Processing and Applied Mathematics*, vol. 4967, Lecture Notes in Computer Science. Springer: Berlin Heidelberg, 2008.
23. Jha S, Kaiser H, Merzky A, Weidner O. Grid interoperability at the application level using SAGA. *IEEE International Conference on e-Science and Grid Computing*, Bangalore, India, 2007; 584–591.
24. Goodale T, Jha S, Kaiser H, Kielmann T, Kleijer P, Merzky M, Shalf J, Smith C. *A Simple API for Grid Applications (SAGA). Open Grid Forum*, November 2008.
25. Smith C, Kielmann T, Newhouse S, Humphrey M. The HPC basic profile and SAGA: standardizing compute grid access in the Open Grid Forum. *Concurrency and Computation:Practice and Experience* 2009; **21**(8):1053–1068.
26. Kacsuk P, Sipos G. Multi-grid, multi-user workflows in the p-grade grid portal. *Journal of Grid Computing* 2005; **3**(3–4):221–238.

Concurrency Computat.: Pract. Exper. 2014; **26**:2280–2291
DOI: 10.1002/cpe

Paper III

Enhanced Resource Management enabling Standard Parameter Sweep Jobs for Scientific Applications

S. Holl, **M. S. Memon**, B. Schuller, M. Riedel, Y. Mohammed, M. Palmblad, and A. Grimshaw.

International Conference on Parallel Processing – The 42nd Annual Conference ICPP 2013 Ninth International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems (SRMPDS), Lyon, France, 1 October 2013, IEEE, pp. 783–790 (2013).

M. S. Memon is one of the co-authors of this publication. He has actively contributed to the implementation of the parameter sweep functionality, specifically focusing on the UNICORE end. In this publication he produced Sections I, II, IV, V and VII. He was also involved in developing the initial structure of the paper.

Enhanced Resource Management enabling Standard Parameter Sweep Jobs for Scientific Applications

Sonja Holl[1], Shahbaz Memon[1], Bernd Schuller[1], Morris Riedel[1, 2], Yassene Mohammed[3], Magnus Palmblad[3], Andrew Grimshaw[4]

[1]Juelich Supercomputing Centre, Forschungszentrum Juelich GmbH, Juelich, Germany, s.holl@fz-juelich.de

[2]School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland, Department of Computer Science

[3]Center for Proteomics and Metabolomics, Leiden University Medical Center, The Netherlands

[4]University of Virginia, Charlottesville, Virginia, USA

***Abstract* — Parameter sweeps are used by researchers with scientific domain-specific tools or workflows to submit a large collection of computational jobs whereby each single job of it only varies in certain parts. They require a more fine-grained distribution of jobs across resources, which also raise a significant challenge for efficient resource management in middleware environments that have been not specifically designed to perform parameter sweeps. This paper offers insights into parameter sweep solutions that support multi-disciplinary science environments via abstraction from resource management complexities using middleware. The solutions are based on use case requirements, enable efficient submission, enhanced usability, and standard compliance. We also apply a use case taken from the life science domain to demonstrate usefulness and efficiency of the solutions.**

***Index Terms*—Parameter Sweep Jobs, HPC, UNICORE, Standards, Taverna, Proteomics**

I. Introduction

For scientific experimentation and analysis the efficient job and data management is considered to be the most significant functionality. In most of the cases the applications are being accessed or executed through a layer of middleware abstraction. This abstraction can have a potential to interface with multiple computing and data platforms. The computing platforms here could be either high performance computing (HPC) or high throughput computing (HTC) resources. We have witnessed in different scientific domains that the use of middleware abstraction over the mentioned resource management systems is becoming useful.

There are certain scientific use cases which might need to have applications running with different parameter sets. This could happen for instance in the field of simulation sciences, where applications are executed in parallel within pipelines i.e. scientific workflows. It is often observed that the simulation is regenerated by sending the same application with different parameter and data sets. In a likewise manner we can easily identify multiple e-Science use cases which possess this requirement. One such use case is [20], in which the workflow implementation is re-reading same data multiple times, thus degrading workflow performance. In scientific computing this problem has been tackled through a concept of parameter sweep. The computational jobs that are required to re-run with different parameters are named parametric jobs. The parameters could be any of application arguments, environment variables and file staging parameters.

While considering the parametric job enactment as an important functionality in supporting scientific user communities, the UNICORE [1] consortium initiated an effort to take this as a potential requirement. As a result of this initiative UNICORE's execution tier is extended to generate and execute parametric jobs. The implementation is built on the open standards based Job Submission Description Language (JSDL) [16] Parameter Sweep extension [17]. The execution tier is a server side functionality, theoretically it could benefit all UNICORE user clients or client side API adopters. The impact of API users is wider as it may benefit a diversified set of scientific tools that are not UNICORE per se, but are willing to use UNICORE based services. One of the examples is the Taverna-UNICORE [5] integration, which is using the UNICORE client side API to submit and monitor jobs on UNICORE servers through the Taverna workbench [6].

The remainder of the paper is structured as follows. Section I introduces the problem space of parameter sweeps in scientific computing. After introducing basics of resource management and middleware in Section II, Section III models the problem space and offers a requirement analysis. Section IV describes solutions to overcome known limitations that are then validated with scientific applications in Section V. A survey of related work is in Section VI, while the paper ends with some concluding remarks.

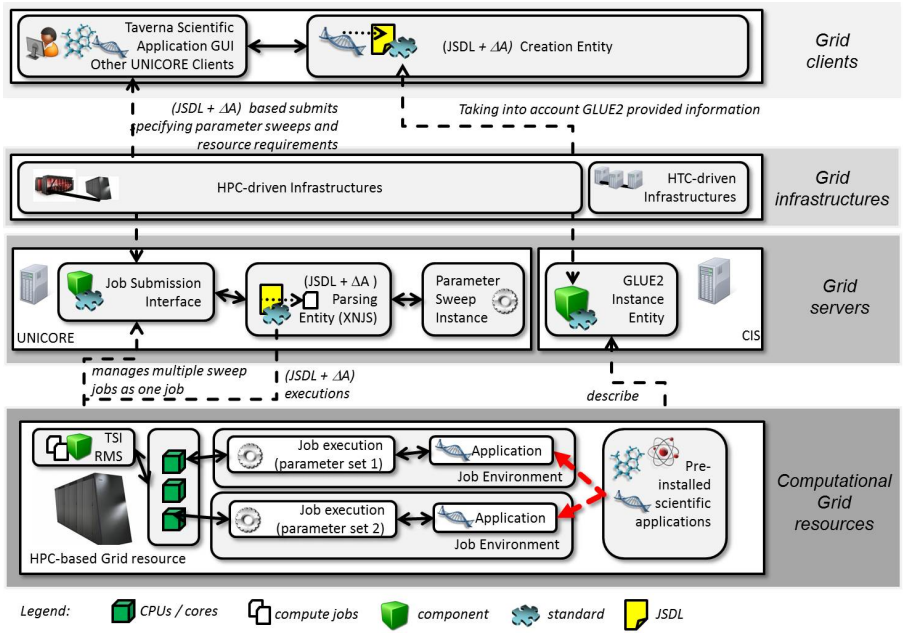


Figure 1 Standard UNICORE Resource Management and overview of parameter sweep adoption in context

II. Standard Resource Management with UNICORE

UNICORE is a middleware that provides a seamless computing and data abstraction to a wide range of computational resources such as high performance computing (HPC) supercomputers or high throughput computing (HTC) oriented geographically distributed resources. The overall architecture follows a standard three-tier architecture as shown in Figure 1, including the 'virtual' Grid infrastructures tier representing on which part of Grid infrastructures the deployment belongs to.

A. CLIENTS

UNICORE Clients offer a user facing tier to interact with server side functionality. UNICORE Rich Client (URC) [18] is a platform agnostic desktop client implemented on the Eclipse framework. It offers a complete set of functionality, which can be provided through the web services. They include managing jobs, data, and accessing remote file systems through a convenient desktop interface. In case of job submission, the URC generates a JSDL instance based on user request and submits it to the client selected UNICORE target system.

UNICORE Command-line Client (UCC) is a platform independent software that provides a command line based access to the server tier, as shown in Figure 1. For the job submission phase UCC can accept job request as a JSDL instance in the form of XML or JSON, it validates the request and submits it to a remote job management service. Once the job is submitted the UCC client tracks the job status and finally stages output files to the user machine.

Apart from user facing interfaces, UNICORE provides client side APIs, which can be used separately for non-UNICORE based user interfaces or portals.

B. SERVICES

UNICORE is based on the principles of service oriented architecture, thus exporting its major capabilities such as job execution and data management via a set of web services. Figure 1 depicts an overview of the UNICORE architecture. By following the architecture in an optimal environment UNICORE clients prepare a job request which is then sent to a remotely deployed web service. The UNICORE core services shown in Figure 1 contain job and data management web services. Once the request is validated, the job management web service forwards the request to a backend execution

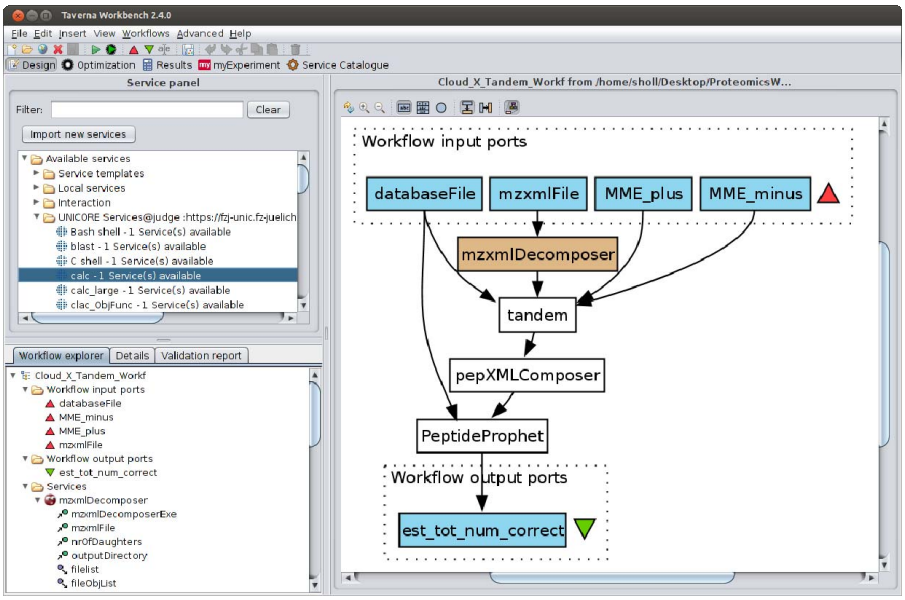


Figure 2 The Taverna workbench with UNICORE plugin and an example scientific workflow

system named as Extended Network Job Supervisor (XNJS). XNJS is an integral part of UNICORE's job and data management framework. It vets incoming job requests by checking them against JSDL compliance. It also checks the incoming request with compute and data capabilities available on backend resources, they include for instance, application support (whether it is serial or parallel), number of compute nodes, number of cores per node, total physical memory and wall time etc. XNJS also implements a support for performing file transfers required for job data stagings.

Its major design decisions are based on open standards to enable a mature and stable access to computing resources and thus it can be exchanged with any other standard compliant technology in the field of middleware.

C. TSI and Ressource Management:

The UNICORE middleware abstraction provides a layer above commercial and open source batch and scheduling systems such as Torque, Load Leveler, SLURM, LSF, just to list a few out of the supported batch systems. UNICORE is used by a wide range of scientific disciplines such as computational biology, physics, or chemistry. While supporting scientific communities, the resource management capabilities of UNICORE focused on providing a simple yet powerful interface keeping many low-level details out of sight for scientific end-uses. It provides extensible access interfaces for managing and monitoring computations and any associated data which is typically required as the standard resource management functions to perform scientific experimentation and analysis with middleware.

UNICORE has been used by Fusion and Virtual Physiological Human communities through its standards based interfaces. From usage and deployment perspective UNICORE has been deployed on production e-Science infrastructures such as DEISA [2] and PRACE [3]. At the time of writing it is being considered as a HPC middleware service platform deployed on supercomputing centers taking part in the XSEDE project [4].

III. Scientific Use Case Environment Requirements

In the life science domain, the Taverna workflow management system [6] has been widely used since many years. Taverna's components are implemented in Java and available as open source. Figure 2 shows the so-called *'Taverna workbench'* that provides a GUI client for the design and monitoring of scientific workflows. Their execution is managed by the Taverna workflow engine. This engine comes along with the client or standalone as Taverna server. Taverna is the common working environment for life scientists and as

such this should not be changed to any other form of clients. Hence, the use of the UNICORE Clients such as the Rich Client or UCC is not possible that represents the first requirement added to Table 1. In terms of scientific applications, Taverna provides to the user a considerable number of *'bio-informatic activities'* representing Web Services, Java classes, or local scripts.

Services can be easily added and activity types can be extended within Taverna. Accordingly, some of the authors extended Taverna with a UNICORE activity [5] to enable simple job submission to a UNICORE environment in order to take advantage of its strength in resource management capabilities. Hence, in earlier work, we already created a plugin to enable the job submission via JSDL to UNICORE. In particular, the support of the wide range of resource management systems was a motivational factor and its adoption of open standards. Hence, also the standard-based job submission in particular with JSDL is another key requirement in order to enable also the job submission to other technologies than UNICORE if needed. We added this as requirement to Table 1.

Taverna offers an easy to use parallelization mechanism for activities, by referring to the depth of an input port and the incoming data object. If the depth of the input data is higher than the input port can consume – for example the input port can consume only one data item, results in depth 0, but the input consists of an array of data items, results in depth 1 – Taverna creates a set of parallel executing activities each consuming one of the data items. The original UNICORE plugin managed this behavior of parallel executed jobs as a single job instance for each activity, which means one JSDL per job instance. In order to increase the performance of parallel job execution of UNICORE jobs in Taverna we extended the existing UNICORE plugin with the parameter sweep approach, further described in [7]. The extension handles the parallel activity execution by creating a parameter sweep JSDL. It identifies global and individual input values and uses the individual inputs as parameter sweep values. The outcome of this is that globally used files only have to be staged in once.

#	Requirements	
	Description	***Technology***
(1)	*Use Taverna Client as working environment*	*Taverna Client plugin*
(2)	*Open standard-based job submission*	*Taverna Client plugin, UNICORE*
(3)	*Increase performance of parallel job submission*	*Taverna Client plugin, UNICORE*

Table 1 Simple Requirements Summary

This enables researchers to focus on their scientific problems rather than 'thinking technically' how these parallel jobs can be separated across different target systems. In addition, the server-side handling of the generation of potentially tens, hundreds or thousands of JSDLs lowers the load of Taverna clients. This also paves the way for specifying and modeling sophisticated science workflows on mobile devices while the server remains to be the powerful part in the workflow execution. We summarized this requirement as the increase of performance with parameter sweeps in Table 1.

IV. Enhanced Resource Management Concepts

For several computational simulations we observed that for certain use cases the applications need to be executed with different set of parameter sets in order to obtain its results. Normally this is achieved by running multiple computational jobs for each iteration of the parameters. These parameters could be for instance application input arguments or environment variables and depend normally very much on underlying scientific code or research application.

A. Overview of the enhanced support with parameter sweeps

The accurate description of the computational resources (i.e. HPC or HTC resource capabilities such as CPU/cores, memory, etc.) is typically provided by an information service. As shown in Figure 1, one example of an information service that works with UNICORE is the common information service (CIS) [8]. This service based on the GLUE2 open standard information model [9]. This also includes the description of the available scientific applications with possible sets of parameters.

As described in Section III, we faced several scientific communities that raise the demand for applications with such parameter sweeps. This section therefore described how we enhanced the standard resource management architecture of UNICORE as described in Section II in order to enable parametric jobs for a wide range of scientific applications. This also has been done without losing sight of the importance of using open standards within UNICORE. Hence, we adopted the JSDL parameter sweep extension (JSDL + ΔA) [7] that represents an open standard for the description of parameter sweep jobs.

Figure 1, provides an overview of this enhancement whereby in each case of parameter sweeps, the computational job requests are represented as a single instance. But this single instance in turn represents a standardized abstraction of various application and data parameters which leads to multiple invocations of the job on different resources for described parameter sets (i.e. 1, 2, … n).

In more detail, OGF produced a mature set of job management and modeling standards. The JSDL specification is the most adopted standard among science middleware communities today. JSDL comes with a core specification followed by a set of profiles which are extensions to the core. Examples include the High Performance Computing Application Extensions [10] or the Single Program Multiple Data (SPMD) extension [11].

In the context of this work, the JSDL parameter sweep extension is implemented, which encapsulates an XML

representation of parametric job requests. This specification has two types of parameters that are Document and File sweeps explained briefly in the next paragraphs.

The *Document sweep* is a function to manipulate the original JSDL instance that was submitted. For instance, if there is an application and it has an argument which is to be substituted by a set of values in the existing JSDL request. This can easily be modeled through the JSDL parameter sweep schema extensions.

The *File sweep* type provides a capability to manipulate the contents of a file that is to be staged-in before the job execution starts. This functionality only provides a way to search and replace set of tokens within the file. Both of the above sweep types use an iteration function such as numeric loops or a discrete set of values if needed. It is possible that these two sweep types mentioned above may not be sufficient for all scientific use cases, therefore the specification is providing extension points for introducing additional kind of sweeps and iteration functions. This enables an extension to the most possible degree without breaking the standard compliant job submission encapsulated as JSDL documents. In this paper our focus is mainly showing a usage of Document sweep types, although the File sweep was also implemented in parts.

B. Details on resource management and JSDL parsing

More details on the UNICORE implementation with respect to resource management enhancements are as follows. The *UNICORE XNJS* in particular was enhanced to support parametric jobs through standards based JSDL parameter sweep extensions. By adopting this standard, UNICORE job management services are capable of accepting user requests as fully JSDL parameter sweep compliant instances that can describe multiple computational jobs to be executed each on the computational resource. Figure 4 describes how the parametric jobs are being handled in a step wise manner in order to understand the handling within the JSDL parsing entity that is part of the XNJS.

The UNICORE XNJS receives a job submission request from a Web service interface that is not the focus of this paper and thus is only abstractly described. The submission request is a JSDL instance including possible different extensions it is bringing along. XNJS delegates this request to the *JSDLProcesssor* component that is responsible of parsing and validation checks of the job descriptions. In turn, the *JSDLProcessor* parses the corresponding job description instance with standard and possible many extensible JSDL elements. If the request contains the JSDL parameter sweep extension elements, it is forwarded to the so called *SweepProcessor*. This component is responsible for validating and resolving parametric requests by analyzing its sweep type (i.e. Document or File type) and the applied parametric function.

For the subsequent resolution phase the *SweepLibrary* is being invoked which checks if the number of generated JSDL instances are exceeding the middleware threshold (pre-defined parameter to engage in denial-of-service attacks with millions of small trivial jobs for example). The middle ware threshold is

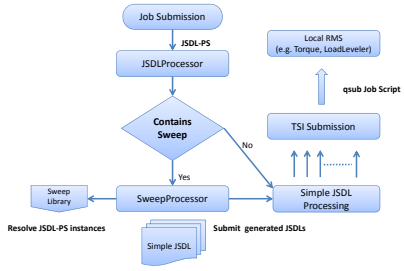


Figure 4 Sequence of steps during JSDL +ΔA parsing.

a server side configuration property that is used as a constraint on the allowable number of JSDL instances generated per sweep request. After performing initial checks, plain JSDL instances are generated, and then forwarded to the execution engine which takes care of translating commands to the UNICORE TSI. If there is no sweep element found in the job request, the XNJS doesn't contact the *SweepProcessor* instead it prepares submission to the target batch system through the TSI process directly as shown in Figure 4.

The TSI component is not part of the XNJS, it is a PERL based component deployed on a login node of a resource management system as shown in Figure 1. Its main job is to translate XNJS commands into the batch system specific job scripts. It also carries out the submission of the job script using batch system submission commands. In a similar manner it performs job management and monitoring after the job is successfully submitted. Typically, there are multiple plain JSDL instances generated after a sweep request and the XNJS submits those as different jobs to the underlying batch system but keep track of them via the so-called *'parent job'*.

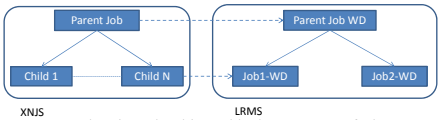


Figure 5 showing the hierarchical structure of the XNJS managed composite and simple jobs, and their relationship with working directories created on target batch system. In order to perform job monitoring, XNJS only exports the parent job's status. Its status is estimated by calculating the overall children jobs' status. Specifically, if all the child jobs are done, the parent job is set to be completed, irrespective of any child job's failure or success.

In more detail, the UNICORE XNJS considers each sweep job (i.e. JSDL + ΔA) as a composite of many simple jobs, thereby it creates a parent job for the composite which serves only as access point for a set of child jobs. As far as the management of job working directories is concerned, XNJS manages a separate directory for every job, either it is parent or

child. But a sweep job from a user perspective is only a single entity, which is accessed through a Web service interface as shown in Figure 1.

Behind this interface there could be hundreds or thousands of child jobs but the user will see only the parent jobs as single job. XNJS creates a directory for a composite sweep job and therein creates a child folder for each of the generated simple jobs. For data-stagings, XNJS download files for generated jobs only once into a parent working directory. Afterwards typical soft links are created within each of the child working directories. It implies that we avoid performing duplicate data transfers. However, this approach has one caveat, if a job request is parametrically sweeping hostnames with same filenames, then XNJS will end up overriding already downloaded files.

C. Differences to the UNICORE workflow engine

The adoption of the parameter sweep job extension in JSDL sounds very similar to the UNICORE workflow engine concept, but we describe in this section briefly some differences in order to avoid confusions. UNICORE workflows are defined as directed acyclic graphs (DAGs) and relationships are specified between different jobs (e.g. loops) or input and output data for subsequent dependent jobs. Each computational job within a DAG is described via its own JSDL element and the whole workflow is submitted to the UNICORE workflow engine for handling the submission process.

In principle, one could model the same functions we have implemented with the parameter sweep job extension, but this would require a tremendous amount of configurations within the definition process of the workflow DAG, especially when huge parameter sets are used (e.g. iteration of one value from one to 100) as specific parameter of one workflow element.

In addition, the workflow functionality of UNICORE enables that parts of the workflow can be submitted to any UNICORE resource available to the end-user that involve potentially geographically distributed resources. Instead, the parameter sweep extension is particularly optimized for submission on one specific computational resource, but using varying parameters and arguments for the job itself. Nevertheless, as shown in the Taverna requirements in Section III, also workflow engines could model one node as JSDL parameter sweep.

V. USE CASE VALIDATION

In this paper we used an existing workflow [12] taken from the proteomics domain to verify the execution performance of parallel jobs via parameter sweeps. The workflow shown in Figure 2 performs a typical task in mass spectrometry (MS)-based proteomics [13], where tandem mass spectra are matched to peptide sequences using X!Tandem (*tandem*) and subsequently validated using PeptideProphet (*PeptideProphet*), both described in [19]. Scientific background: a common approach in MS-based proteomics is the identification of proteins “bottom-up” via enzymatic digestion of the proteins into peptides. The peptides are then separated by liquid chromatography coupled to mass spectrometry. In liquid chromatography, the peptides are separated by their physicochemical properties, such as size, charge and hydrophobicity. In the mass spectrometer, the peptides are separated with very high resolving power based on their mass-to-charge ratios. Individual peptides are then selected and fragmented using collisions with neutral gas or ion-electron reactions. The output is a set of tens of thousands of peptide masses and associated fragmentation spectra. These spectra are compared against those predicted from the genome by one or more algorithms – here X!Tandem – by comparing measured spectra with calculated spectra for all peptides of similar mass that could possibly be generated by the enzyme used from the biological species. After this step, PeptideProphet estimates the probability of each peptide-spectrum match assigned by X!Tandem by a mixture model of the X!Tandem score distribution, assuming there will be some correctly and some incorrectly identified spectra.

To speed up the calculation of *X!Tandem*, Mohammed et. al. [12] developed an mzXMLDecomposer and pepXMLComposer to split up the input file into several small input files and after execution join the results. Using these parts as inputs for the tandem activity in Taverna, several tandem instances are being created, each of which consuming one of the input file parts. This parallel execution was performed on the one hand by the traditional plugin, sending one JSDL for each parallel execution and on the other hand by the extended plugin, using UNICORE's JSDL parameter sweep extension sending only one single JSDL job for all executions.

For the evaluation of the extended parallel sweep execution the mass spectrometry workflow was used as described in the previous section. As input set a human dataset was used with a human fasta library. The library was 37.4 MB and the input data set was 113.8 MB. To speed up the execution, the input file was split into 32 sub files (between file size 3.0 MB and 4.2 MB); following represent an individual input for the sweep JSDL whereas the database file is used by each execution and needs to be staged-in during sweep execution only once.

The non sweep classic solution submits for each of those sub-files one job to the Grid (in total 32) each including the library and one sub file (41 MB upload per job). These 32 jobs are monitored by sending every 2 seconds a web service call to receive the job status. The developed sweep extension executes one job and uploads the library once and all sub-files together (151.2 MB). The single job is monitored by sending a web service call every 2 seconds.

Table 2 shows the analysis in detail; the extended plugin performs during the single job execution and the overall workflow execution faster than the common submission mechanism. This is mainly based on the avoided stage-ins for job execution. The CPU load was monitored for Taverna during the executions by using the top command, showing that the load could be shifted to the server component, as the client now only has to monitor one job instead of several job instances. The total packets and transferred bytes were captured by using the tool Whireshark

(http://www.wireshark.org) using a filter for the target registry and corresponding port.

	Common job submission	*Job submission via sweep extension*
Single tandem job runtime (in minutes)	*5.5*	*4.2*
Overall workflow runtime (in minutes)	*10.3*	*8.4*
Total web service calls	*4032*	*126*
Average CPU load on the client in %	*43.7*	*14.1*
Total packets	*458,353*	*15,244*
Totally transferred bytes	*1,370,513,098*	*190,571,550*
Total file upload in MB	*32 x 44MB + 32 x 4.5MB ≈ 1555MB*	*1 x 44MB + 32 x 4.5MB ≈ 191MB*

Table 2 showing the performance for job execution and data stage-in of the common UNICORE plugin and the extended parameter sweep

VI. Related Work

This paper integrates work concepts from several different fields of related work that can be categorized roughly in workflow engine support and distributed processing of jobs.

Starting with surveying the field of workflow management systems and middleware developments there has been a tremendous amount of research to exploit the concept of parameter sweep for enhancing the performance of scientific applications. But in this section we relate to those developments which are very much close to existing middleware implementations in the standards space, or highlighting a proprietary approach.

gEclipse [14] is an Eclipse based framework that provides an integrated development environment for researchers. It allows an abstraction layer to plugin different grid middleware stacks. gEclipse supports JSDL and many of its extensions. Most notably it is the first implementation of the JSDL parameter sweep specification that is named as *'JSDL-Param library'*. It provides a graphical editor to let users configure JSDL based job requests without being exposed to the internals of JSDL's XML format. gEclipse generates JSDL instances from JSDL parameter sweeps at the client side and forwards them to the intended execution site.

In contrast to our approach, gEclipse doesn't send JSDL parameter sweep job instances directly to the target middleware, because at the time when this extension was implemented no middleware implementation was able to support this type of JSDL extension. But the JSDL-Param library is rather tightly integrated with the Eclipse platform. Thus, it will be challenging for other clients, or middleware providers which may use this library independent of the extensive Eclipse platform dependencies.

Another workflow management system in context is the gLite Workload Management System (WMS) [15] that offers an interface and meta-scheduler that provides an interface for describing parametric jobs. In contrast to our approach, within the WMS client, a job's parametric space is represented through the JDL (Job Description Language) which is a proprietary language to describe job submission requests. Within JDL, different parameter functions can be specified such as start, end, and step, similar to JSDL standard extension with parameter sweeps. In addition, the parametric iterations can be applied to more than one program arguments in one JDL instance, with a flexibility to specify them in a linear range, or a discrete set of values. But to our observation, a single instance of JDL is limited to only single parametric modification functions. The JSDL parameter sweep instance instead enables to specify a set of parallel, and nested sweep modification functions as part of the XML description. Thus our implementation supports that feature and can handle a complex hierarchy of parallel parameter sweeps. From an interoperability perspective, JDL is a proprietary language, thus it is not trivial for other scientific workflow or portal clients to adopt them.

VII. Conclusions

We have shown in this paper a further specialized support for a specific set of scientific use cases requirements based on earlier work that generally enables UNICORE with Taverna. In order to overcome the requirement (1) in Table 1, Taverna was still used but enhanced with the client support of specifying JSDL parameter sweep jobs. Hence, the working environment of life scientists does not need to change and the load of the Taverna client remains low.

Requirement (2) of Table 1 raised the demand of using open standards rather than proprietary job description schemas for parameter sweeps (cf. to WMS proprietary language in Section V). JSDL itself is an open standard adopted by many middleware systems and the JSDL normative extension for parameter sweep jobs is another standard profile where several adoptions are emerging (e.g. within GENESIS). Using these standards, it is possible to exchange UNICORE with any other standard compliant middleware technology.

Table 1 also lists requirement (3) that represent the functionality of the sweep jobs in itself. For this not only the Taverna client plugin for UNICORE was enhanced, also UNICORE itself needed to be enhanced with an approach to support parameter job extensions to JSDL. Section IV explains in detail the taken approach that is seamlessly integrated into the UNICORE middleware without major changes to its components. In other words, the same submission interface can still be used and is not changed while the JSDL schema is just extended with additional XML elements that are exactly

defined in the OGF JSDL parameter sweep extension specification.

We can also conclude that the support for parameter sweeps is highly relevant for scientific workflows, but is typically not covered by workflow engines to this level of granularity. We described in Section IV some differences, but also outline the use of parametric jobs as parts of a larger scientific workflow such as those defined using the UNICORE workflow or Taverna workflows.

Future works remains on the full support of the File sweep parameter sweep extension as specified in the JSDL specification. Also, we noted that if all the child jobs are done, the parent job is set to be completed, irrespective of any child job's failure or success. Future work includes more sophisticated approaches of reliability and fault tolerance potentially taking advantage of the dynamic configuration of networks of HPC machines that arise in some of the more recent HPC architectures and network providers (e.g. Mellanox and JUROPA3 system in Juelich). If an error occurs, the HPC machine in itself may reconfigure and redistribute the job and thus we need to synchronize this status with executions performed with or without parameter sweeps in UNICORE.

ACKNOWLEDGEMENTS

The work in this paper is partially supported by the Extreme Science and Discovery Project (XSEDE) of the National Science Foundation grant number OCI-1053575. The authors also want to thank the Taverna team for their support.

REFERENCES

[1] Streit, Achim, et al. "UNICORE 6—recent and future advancements" annals of telecommunications-annales des télécommunications 65.11-12 (2010): 757-762

[2] Gentzsch, Wolfgang, et al. "DEISA—Distributed European Infrastructure for Supercomputing Applications" Journal of Grid Computing 9.2 (2011): 259-277

[3] http://www.prace-project.eu/, Accessed May, 2013

[4] https://www.xsede.org/, Accessed May, 2013

[5] Holl, Sonja, Olav Zimmermann, Martin Hofmann-Apitius, "A UNICORE Plugin for HPC-Enabled Scientific Workflows in Taverna 2.2." *2011 IEEE World Congress on Services (SERVICES),* IEEE, 2011

[6] Wolstencroft, Katherine, et al. "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud" *Nucleic acids research* (2013)

[7] Memon, S., Holl, S., Schuller, B., Riedel, M., Grimshaw, A. "Enhancing the performance of workflow execution in e-Science environments by using the standards based Parameter Sweep Model", In Proceedings of XSEDE13, 2013, to be published

[8] Memon, Ahmed S., et al. "CIS: An information service based on the Common Information Model", IEEE International Conference on e-Science and Grid Computing, IEEE, 2007

[9] Carenini, Alessio, et al. "Glue2: a web service discovery engine with non-functional properties" on Web Services, ECOWS'08, IEEE Sixth European Conference, 2008

[10] Dillaway, B., Humphrey, M., Theimer, M., Wasson, G. HPC Basic profile, Version 1.0 http://www.ogf.org/documents/GFD.114.pdf

[11] Savva A., "JSDL Single Program Multiple Data Application Extensions, OGF draft" , 2007 http://www.ogf.org/Public_Comment_Docs/Documents/May-2007/draft-ogf-jsdl-spmd-008.pdf

[12] Mohammed, Yassene, et al. "Cloud Parallel Processing of Tandem Mass Spectrometry Based Proteomics Data" *Journal of Proteome Research* 11.10 (2012): 5101-5108

[13] Aebersold, R., Mann, M., Mass spectrometry-based proteomics, Nature 422 (2003), 198-207

[14] Wolniewicz, Paweł, et al. "Accessing Grid computing resources with g-Eclipse platform" Computational Methods in Science and Technologie 13.2 (2007): 131-141

[15] Andreetto, Paolo, et al. "The gLite workload management system." Journal of Physics: Conference Series. 119.6 IOP Publishing, 2008

[16] Anjomshoaa, A., et al.," Job Submission Description Language (JSDL) Specification, Version 1.0", 2005, Global Grid Forum (GGF), p. 72

[17] Drescher, M., Anjomshoaa, A., Williams, G., Meredith, D. "JSDL Parameter Sweep Job Extension" http://www.ogf.org/documents/GFD.149.pdf

[18] Demuth, B., et al. "The unicore rich client: Facilitating the automated execution of scientific workflows" 2010 IEEE Sixth International Conference on e-Science (e-Science), IEEE, 2010

[19] Deutsch, Eric W., et al. "A guided tour of the Trans - Proteomic Pipeline" Proteomics 10.6 (2010): 1150-1159

[20] Gideon Juve, et al, "Characterizing and profiling scientific workflows", Future Generation Computer Systems 29.3 (2013): 682-692

Paper IV

Enhancing the performance of scientific workflow execution in e-Science environments by harnessing the standards based Parameter Sweep Model

M. S. Memon, S. Holl, M. Riedel, B. Schuller, and A. Grimshaw.

Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (XSEDE'13), San Diego, California, USA, July 22–25 2013, ACM Press, pp. 56:1–56:7 (2013).

M. S. Memon is the main author of this publication. He has actively contributed to the implementation of the parameter sweep functionality at the UNICORE end and also helped in enabling the case study. In this publication he produced Sections I, II, some portions of IV, VI and VII. He was also involved in developing the structure of the paper.

Enhancing the performance of scientific workflow execution in e-Science environments by harnessing the standards based Parameter Sweep Model

Shahbaz Memon
Juelich Supercomputing Centre (JSC)
Forschungszentrum Juelich
52425 Juelich, Germany
m.memon@fz-juelich.de

Sonja Holl
Juelich Supercomputing Centre (JSC)
Forschungszentrum Juelich
52425 Juelich, Germany
s.holl@fz-juelich.de

Bernd Schuller
Juelich Supercomputing Centre (JSC)
Forschungszentrum Juelich
52425 Juelich, Germany
b.schuller@fz-juelich.de

Morris Riedel
Juelich Supercomputing Centre (JSC)
Forschungszentrum Juelich
52425 Juelich, Germany
m.riedel@fz-juelich.de

Morris Riedel
School of Engineering and Natural Sciences
University of Iceland
Reykjavik, Iceland
m.riedel@fz-juelich.de

Andrew Grimshaw
Department of Computer Science
University of Virginia
Charlottesville, Virginia, USA
grimshaw@cs.virginia.edu

ABSTRACT

Certain scientific use cases possess complex requirements to have Grid jobs executed in collections where the jobs' request contains only some variation in different parts. These scenarios can easily be tackled by a single job request which abstract this variation and can represent the same collection. The Open Grid Forum (OGF) standards community modeled this requirement through the Job Submission and Description Language (JSDL) Parameter Sweep specification, which takes a modular approach to handle different variations of parameter sweeps (e.g. document and file sweep). In this paper we present the UNICORE server environment implementing this specification build upon its existing JSDL implementation. We also demonstrate the application of UNICORE's parameter sweep extension for optimizing job executions, which are submitted as a sub-activity of a Taverna based scientific workflow. Further we validate our approach by analyzing performance of the workflow with and without using the parameter sweep extension.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; D.2.12 [**Software Engineering**]: Interoperability—*Data mapping*

General Terms

Management, Design, Performance, Standardization, Human Factors

Keywords

UNICORE, parameter sweep, standards, Taverna, optimization

1. INTRODUCTION

In e-Science, job submission and management is one of the significant functionalities to manage scientific simulations. Most scientific disciplines such as, physics, chemistry, and biology, use computational simulation as a significant tool to perform simulations based on numerical methods or known physical laws, that also includes data analysis in a more effective and precise manner. However, for executing several tasks within such an experiment, scientific workflows have successfully being used in e-Science [25]. scientific workflows are widely used to ease complex computational simulations in most scientific disciplines.

Computing science is continuously in the pursuit of technological [21, 15] and infrastructural advancements [20, 26, 1]. From the technical perspective there is an extensive set of open standards from OGF [2]. Many of these standards are actually an outcome of lessons learned from the production Grid infrastructures, therefore most of them are easily adoptable in real scientific environments. JSDL [4], for example, is a basic building block for representing job requests.

Certain scientific use cases have a requirement that several identical jobs are executed with the same executable but with varying a magnitude of parameter values. As a result user facing clients need to submit hundreds and thousands of sweep generated jobs to a target resource. Therefore the execution service receiving these requests see multiple submissions with each may have an overhead to perform separate, and most likely duplicated data stagings. Considering a situation in which the client that submitted multiple jobs need to monitor each of them, the server will be busy in replying to the numerous incoming monitoring requests. Apparently, the client generated jobs are not optimal to data oriented applications where huge amount of data is fetched to the

remote computing elements.

The open standards community introduced the JSDL extension that specifically addresses this requirement through the JSDL Parameter Sweep specification (JSDL-PS) [7]. This extension is capable of representing multiple job requests in a single abstract job where parameter type and range of values to be iterated are defined. Middleware can easily adopt this specification to enhance the overall performance of jobs having parametric behavior.

The application of parameter sweep has emerged as an essential tool for scientific workflow management systems. As an example we have identified a Taverna [17] based scientific application. Taverna is a widely used workflow management system that is able to perform the job executions in parallel. While taking a scientific use case into account, we presented in [13], how in one step Taverna is performing 120 parallel Grid job executions. This parallel submission can become a bottleneck as 120 jobs not only have to be sent and monitored during the execution phase but also the input files have to be uploaded for each job in a repetitive manner. As a result, we have a high cpu load on the client machine [11] and many unnecessary replicated file uploads. Hence, the workflow step that is required to generate multiple jobs for analysis, can be enhanced through the use of parameter sweep that is the key contribution in this paper.

After the introduction, Section 2 will describe the realization of JSDL-PS in the UNICORE middleware. The integration of this method into the client layer is outlined in Section 3. In Section 4, we show our use case exploiting the parameter sweep implementation. The performance analysis of this use case is given in Section 5. Related work is presented in Section 6 and finally the paper ends with concluding remarks and future work in Section 7.

2. PARAMETER SWEEP IMPLEMENTATION

UNICORE [24] is an open source, standards based middleware, allowing a transparent, secure, and seamless access to distributed high performance computing (HPC) resources. It adopts several distributed computing standards often required by end users of e-Science infrastructures and thus for the over all scientific computing community. UNICORE supports several OGF standards, such as JSDL, OGSA-BES [8], and ByteIO [19]. In order to carry out job submissions, it offer a convenient, and extensible middleware platform to process the JSDL compliant requests. UNICORE is implemented on the basis of a service oriented architecture. The vital functionalities such as job submission and management, and data management are realized in the form of SOAP [5] web services.

While implementing the JSDL-PS into the UNICORE service environment, it was not trivial to decide that, at what level of the UNICORE architecture, which is shown in Figure 1, the resolution of parametric jobs should be implemented. To our analysis there are two alternatives. First option is to have this process at the client tier that can generate the resultant sweep jobs, and subsequently start submitting each of them. This tier typically may consist of client APIs, scientific gateways, and workflow management systems. As mentioned before, this approach inevitably exposes compute site with multiple remote calls, therefore it could possibly waste significant amount of infrastructure re-

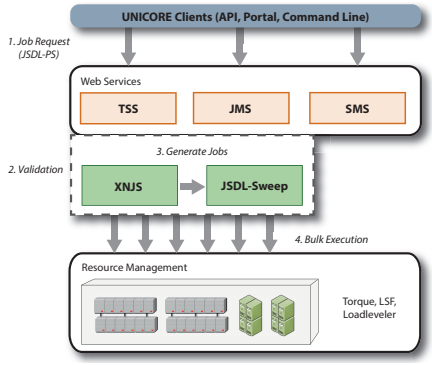


Figure 1: UNICORE Architecture

sources. Another alternative is to have the parametric resolution at the execution layer - the entity responsible for job execution. In this case the client sends a single job request compliant with the JSDL-PS format to the remote execution site. Following to that the job request will be analyzed, and according to the parameter range and type the compute site will initiate the consequent job runs. This approach relieves UNICORE site from staging application data files multiple times, thus it saves the amount of network bandwidth, and also efficiently utilize the amount of storage space allocated to a user submitting that job. Considering a fact that scientists normally perform simulation analysis and simulated experimentation on public funded computing infrastructures where compute and data resources are limited. The second approach appears to be more feasible to UNICORE implementation.

Figure 1 gives an architectural view of UNICORE, and also describe the interactions happened during the job submission and execution phase. In the first step Client interacts with the Target System Service (TSS) web service to create a job. As a response of this request Client recieves a WS-Adressing [6] compliant Endpoint Reference (EPR) of the Job Management Service (JMS). JMS is a WS-Resources [10] compliant web service that provides an interface to represent and manage individual job instances. The Client starts a job and waits until it is completed. Once the job is successfully finished Client can easily fetch the output. During the job submission phase, the request flows through the JMS interface to the back-end execution system called Extended Network Job Supervisor (XNJS) [22]. The XNJS manages the life cycle of each job managed within the UNICORE services container. The job lifecycle is generally consist of creation, incarnation, and finished states. This component is responsible for carying out the basic JSDL parsing, validation, and most notably translating the XML formatted JSDL instance to the target batch system specific scripts. The XNJS handles the parsing of JSDL named as JSDLProcessor. In order to support any extensions of JSDL, XNJS is required to have an extended processor that understands the incoming job's extension elements. By complying to this de-

sign principle, it has been extended to support the JSDL-PS specification. It is implemented as a subentity to JSDLProcessor, and it is called SweepProcessor. The SweepProcessor is activated on-demand, that occurs when an incoming job request contains the parameter sweep elements, otherwise it is not even instantiated. Figure 3 is one such instance of the JSDL-PS compliant job request. The SweepProcessor performs the validation in two consecutive phases. Firstly it validates the incoming request against the allowable limit of future generated jobs to be executed by XNJS. This limit is a constant and can easily be configured within XNJS to set the maximum threshold of the jobs allowed per sweep request. After successfully passing through the validation phase it finally resolves the sweep parameters into a list of plain JSDL instances. The SweepProcessor performs these both functions using the JSDL-Sweep library [18] which is part of the GenesisII middleware [18]. The UNICORE team extended this library to be compatible with XNJS. As soon as the JSDL instances are generated the JSDLProcessor executes them on a target resource management system (e.g. Torque, PBS).

As it is mentioned above, a parametric job is a single JSDL-PS instance, which may lead to hundreds and thousands of separate jobs. To have an individual JMS service instance for each of the resultant sweep job is not scalable, and therefore affect server's responsiveness if it recieves multiple parametric jobs at the same time. It is also pragmatically unfeasible to let Clients communicate with such number of JMS instances spawned from a single sweep request. We tackled this problem by introducing a single parent job, that is linked to the JMS instance that was created at the time of Client's initial job request. At the XNJS level the parent job is supervising and tracking all the compute jobs generated in response to the sweep request. The parent is not executed per se, rather it provides a proxy to the Client through which the status of the child jobs can be monitored. While executing the generated jobs our implementation ensures that for the common set of files the data staging is performed only once. But if the input data set is unique for each child, the files are staged-in separately to the respective job's working directory. As far as monitoring is concerned, the status of the generated jobs is mapped to the status of the parent job. It means when all the generated jobs are done with executing, the status of the parent is set to completed, and this is irrespective to the status of the children whether they are finished successfully or failed.

3. TAVERNA'S EXTENSION FOR SUPPORTING PARAMETER SWEEP

The Taverna workflow management system [17] has been widely used in the field of life science informatics in recent years. Taverna's components are implemented in Java and are available as open source. The Taverna workbench provides a graphical client for the design and monitoring of workflows. Workflow execution is managed by the Taverna workflow engine, which comes along with the client or standalone as Taverna server. Taverna provides to the user a considerable number of so-called bioinformatic activities that can be invoked via Web Services, Java classes, local scripts or other special services. Services can be easily added and activity types can be extended. Accordingly, we extended Taverna with a UNICORE activity [13] to enable job submission to a UNICORE environment.

All those services can be used to design a workflow in Taverna. A workflow consists of several user defined activities connected via its input and output ports. Thereby the dataflow between activities and the overall workflow structure is defined. A very useful feature offered by Taverna, is the parallel execution of activities. As Taverna has a dataflow-centric workflow language, an activity is invoked if all required data items have reached their corresponding input ports. Input ports have a user defined depth, whereas depth 0 represents one item, depth 1 represents an array of items, and so on. If an input with a higher depth than the port can consume reaches the input port, one individual service is executed for each data item. As an example, for each item of an array with size x that arrives at a port with depth 0 the service is invoked x times separately. If several high dimensional inputs reaching input ports with a lower depth, the cross or dot product of all input values is consumed typically parallel by the service. In designing such a workflow, our extended UNICORE activity can also be executed in parallel in Taverna. The type of parallelism and number of parallel jobs can be set by the user. The parallel execution mechanism in Taverna was extended especially for UNICORE activities with the new UNICORE parameter sweep mechanism, as shown in Figure 2.

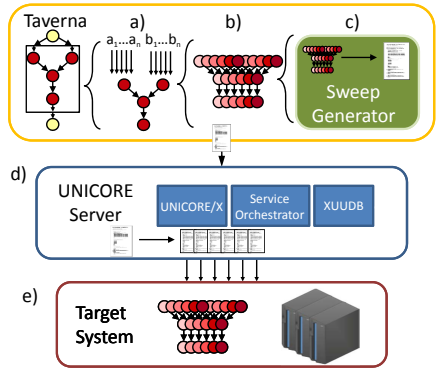


Figure 2: The figure shows the mechanism in Taverna to create a sweep job. (a) Two value sets arrive at the workflow input ports. (b) Corresponding to the parallel execution in Taverna several parallel workflows are instantiated. (c) The sweep generator collects all instantiations and creates one JSDL sweep job for submission to UNICORE. (d) The UNICORE server splits up the JSDL script into the original number of executions. (e) The workflows are executed on the target system.

During the parallel invocation of a UNICORE activity in a Taverna workflow, several activity instances are created. We have implemented a sweep generator in our Taverna plugin to collect all these instances. This sweep generator gains information about the maximum number of allowed parallel services and can collect as many allowed activities. When

the maximum number is reached or all instantiated activities have been selected, the sweep generator creates a parameter sweep job for submission. Therefore, identical and different inputs are identified. Identical inputs are set as global job inputs, the differing inputs are defined as parameter sweeps. Each differing value defines one sweep function value in the parameter sweep job description. In doing so, globally used input files are only uploaded once. An example JSDL file created by the sweep generator is shown in Figure 3. The *POSIXApplication* node describes the application specific parameter, including output and input parameter values. The *Sweep* node hosts all the elements for a parameter sweep. The sweep *Match* element defines the element that should be swept during execution, in this example we are sweeping over *jsdl:posix:Evironment* element number 3. You can find this element in the *POSIXApplication* node, having the value 253. During the sweep this value is replaced for each job by one of the values given in the *fun:Value elements*.

Figure 3: Example XML structure for a parameter sweep job.

The sweep generator also stores the mapping of Taverna jobs and instantiated sweep values. After the job submission and execution on the target system, the sweep generator keeps track about the outputs files. Each output is downloaded as string, file or logical address and mapped to its specific Taverna job. The output values are staged out to the activities output ports for further usage in the workflow.

4. USECASE: ADVANCED WORKFLOW OPTIMIZATION

The set up of a scientific workflow can be very challenging for domain scientists with less computational background. A reason is the enormous choice of similar applications, each of which coming along with a considerable number of parameters. Scientists typically tend to use the already known applications and their default parameters. However, default parameters are often not the best suitable ones and may lead to weak scientific results. Thus, we have implemented a plugin [12] for the Taverna workflow management system to enable a semi-automated parameter optimization of scientific workflows. The plugin uses a genetic algorithm [14] to sample the parameters search space. Additionally, ranges and dependencies between parameters can be defined to narrow the search space. The parameters are encoded as genomes in the genetic algorithm. As typically only parts of a workflow need to be optimized, a sub-workflow can be defined for the optimization. During optimization, several instances of this sub-workflow are executed in parallel. Each instance is set up with one parameter sample from the genetic algorithm. After the parallel execution, the result values are taken as fitness and a new generation of sub-workflows is instantiated.

As the parallel execution of several sub-workflows might lead to a high cpu load on the client machine, we have extended this plugin by workflow Grid submission [11]. This extension allows to submit a bunch of sub-workflows to a UNICORE infrastructure, which executes the sub-workflows on a Taverna server on the target system. The submissions of the parallel sub-workflows were performed by single jobs. Our developed UNICORE parameter sweep plugin can be used to perform these submissions in one single job. In doing so, we use the extended parameter sweep mechanism as described in Section 3 during the submission. During the invocation of one generation, several sub-workflows are instantiated for execution. The sweep manager again collects these sub-workflows and only submits one sweep job definition for the bunch of sub-workflow executions. The sweep job contains as sweep values the different parameter sets. Again, identical input files have to be uploaded only once, such as the sub-workflow file itself. After execution of the sub-workflows, the outputs are taken as fitness values for creation of the next generation.

5. PERFORMANCE ANALYSIS

We have designed an example workflow in Taverna in order to measure the performance of the developed parameter sweep extension. The workflow execution was performed by the two different submission mechanisms. The first one is the conventional submission as described in [11]. This submission mechanism creates for each job an individual JSDL, uploads all input files individually for each job and monitors each jobs after submission. The other developed sweep generator method collects all invocations, creates one JSDL, uploads identical files only once and monitors a single job after submission.

For the performance measurement we set up two different use cases. The use cases have in common, that they perform a parameter optimization with selected generation size of 20 and 5 generations, means 100 executions in total. The first workflow has two string inputs. The second use case has next to the same two string additional two file inputs (each of size 17MB). The file inputs remain the same for all executions whereas the string inputs are changeable parameters and will thus be subject of the optimization process. We have executed both workflows once with the conventional submission method and once with the developed sweep generator method. In both cases, 100 executions are performed:

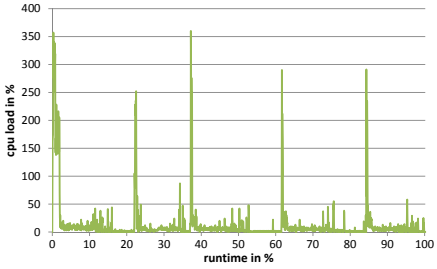


Figure 4: The cpu load on the client during execution of the conventional submission mechanism.

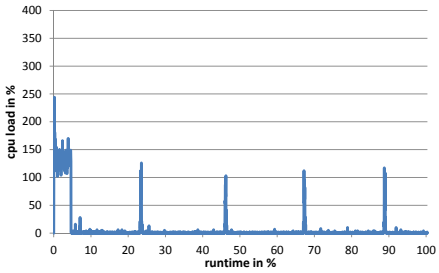


Figure 5: The cpu load on the client during execution of the sweep generator.

20 parallel workflow invocations are performed five times in a series.

Figure 4 shows the recording of the cpu load on a client machine with the conventional submission method (i.e. without using our parameter sweep extensions) and Figure 5 shows results with the new sweep generator submission method. As the two different use cases do not major differ in the recordings, Figure 5 shows a small workflow example. The chart shows that the initialization phase of the newly developed method takes a little longer but consumes in the average less than the conventional method. The other peaks indicate the submissions of the jobs during the optimization steps. It becomes apparent that the conventional method requires in the average nearly twice as much cpu than the newly developed method. Additionally, the noise between the submissions, which represent the monitoring of the jobs, is in the average more than fourth times higher in the conventional method. This results from the higher number of jobs being send and monitored by the conventional method that includes a lot of polling WS calls for job status. Table 5 demonstrates statistics about the cpu load. Additionally, it shows the more significant measurement namely the upload of input files. Whereas the sweep generator identifies similar file inputs and uploads these only once, the conventional method has no mechanism for this and have to upload each file for each job repeatedly. In the average case, this is the

	conventional submission	sweep generator
File upload in kB (small example)	400	20
File upload in MB (large example)	700	35
Average CPU load on the client in %	12.89	7.79
Max. CPU load on the client in %	360	244

Table 1: Statistics of the submission via the conventional submission mechanism and the developed sweep generator.

number of parallel executions multiplied by the file size for each input file. This might mainly affect the time required for the uploading step. We haven't measured the time itself here, because it critically depends on the used protocol for stage-ins. We can recommend to use UFTP [23], which is a high performance data transfer for UNICORE. Taken together, the performance measurement shows that the newly method perform better considering the cpu load and moreover perform much better regarding the upload rate.

6. RELATED WORK

In the landscape of workflow management systems, and middleware developments there has been a tremendous amou of research done to exploit the concept of parameter sweep for enhancing the performance of e-Science applications. One example is the parameter exploration mechanism implemente in Kepler via Nimrod Directors [3]. It enables scientists to perform various parameter sweeps over different parameter without using any specific looping mechanism. Such as the most worklfow management systems, the described implementation is integrated into the workflow engine on the client machine. Hence, in this section we relate to those developments which are especially very much close to existing middleware implementations in the standards space, or a proprietary approach.

gEclipse [9] is a Eclipse based framework that provides an integrated development environment for researchers. It allows an abstraction layer to plugin different grid middleware stacks. gEclipse supports JSDL and its most of the extensions. Most notably it is the very first implementation of the JSDL-PS specification that is named as JSDL-Param library. JSDL-Param provides a graphical editor to let users configure JSDL based job requests without being exposed to the internals of JSDL's XML format. gEclipse generate JSDL instances from JSDL-PS at the client side and forward them to the intended execution site. gEclipse doesn't send JSDL-PS instance directly to target middleware, it is because at the time when this extension was implemented no middleware implementation was able to support JSDL-PS compliant job requests. Nevertheless, the JSDL-Param library is appeared to be integrated with the Eclipse platform. Thus, it will be challenging for other clients, or middleware providers which may use this library independent of the extensive Eclipse platform dependencies.

gLite' Workload Management System (WMS) [16] is a key entry service and meta-scheduler that provides an interface for describing parametric jobs. For a WMS client, a

job's parametric space is represented through the JDL (Job Description Language) which is a propereitary language to describe job submission requests. Within JDL, different parameter functions can be specified such as start, end, and step, similar to JSDL-PS. Moreover, parameteric iterations can be applied to more than one program arguments in one JDL instance, with a flexibility to specify them in a linear range, or a discrete set of values. But to our observation, a single instance of JDL is limited to only single parametric modification functions, whereas JSDL-PS allows you to specify a set of parallel, and nested sweep modification functions. Thus our implementation supports that feature can handle complex heirarchy of parallel parameter sweeps. From a interoperability perspective, JDL is a proprietary language, thus it is not trivial for other scientific workflow or portal clients to adopt them unless middleware specific APIs are not used.

7. CONCLUSIONS AND FUTURE WORK

In this paper we demonstrated the realization of the JSDL parameter sweep specification in UNICORE service environments. This feature enabled UNICORE clients to increase the performance of job runs where requested applications intended to execute on definite parametric space. We developed not only the solution ready to use, but also identified an existing scientific application that was explicitly submitting jobs for each parameter iteration to production UNICORE sites. This use case performed a workflow optimization and executed 20 sub-workflows in parallel and was used to prove our implementation. The performance analysis showed that the parameter sweep library ensured the reduction of the cpu load on the client machine and most notably a higher performance for file uploads.

At the time of writing, the parameter sweep implementation has its first release part of the UNICORE main distribution, thus we anticipate a couple of improvements, such as the support for File type parameter sweeps. This kind of sweep is intended for parametric iterations on textual file contents. Furthermore, the usability of information exposed through the master job can be enhanced in such a way that the clients can have fine grained control and view of the child jobs.

8. ACKNOWLEDGMENTS

The work in this paper is partially supported by the Extreme Science and Discovery Project of the National Science Foundation grant number OCI-1053575. The authors also want to thank the Taverna team for their support.

9. REFERENCES

[1] Extreme Science and Engineering Discovery Environment. `http://www.xsede.org`. [Online; accessed 14-March-2013].

[2] Open Grid Forum. `http://www.ogf.org`. [Online; accessed 12-March-2013].

[3] D. Abramson, B. Bethwaite, C. Enticott, S. Garic, and T. Peachey. Parameter space exploration using scientific workflows. In *Computational Science–ICCS 2009*, pages 104–113. Springer, 2009.

[4] A. Anjomshoaa and et al. Job Submission Description Language (JSDL), Version 1.0, July 2008.

[5] D. Box and et al. Simple Object Access Protocol (SOAP) 1.1. `http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`. [Online; accessed 12-March-2013].

[6] D. Box and et al. Web Services Addressing (WS-Addressing). `http://www.w3.org/Submission/ws-addressing/`. [Online; accessed 12-March-2013].

[7] M. Drescher and et al. JSDL Parameter Sweep Extension, May 2009.

[8] I. Foster and et al. OGSA Basic Execution Service (BES), Version 1.0, Nov 2008.

[9] H. Gjermundrod, M. D. Dikaiakos, M. Stumpert, P. Wolniewicz, and H. Kornmayer. g-eclipse - an integrated framework to access and maintain grid resources. In *Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing*, GRID '08, pages 57–64, Washington, DC, USA, 2008. IEEE Computer Society.

[10] S. Graham and et al. Web Services Resource 1.2 (WS-Resource), April 2006.

[11] S. Holl, O. Zimmermann, B. Demuth, B. Schuller, and M. Hofmann-Apitius. *Secure Multi-Level Parallel Execution of Scientific Workflows on HPC Grid Resources by Combining Taverna and UNICORE Services*, volume 15 of *IAS Series*, pages 27–34. Forschungszentrum Jülich, Jülich, May 2012.

[12] S. Holl, O. Zimmermann, and M. Hofmann-Apitius. A new optimization phase for scientific workflow management systems. *2012 IEEE 8th International Conference on E-Science*, 0:1–8, 2012.

[13] S. Holl, O. Zimmermann, and M. Hofmann-Apitius. A unicore plugin for hpc-enabled scientific workflows in taverna 2.2. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 220–223, July.

[14] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[15] P. Kacsuk, J. Kovacs, Z. Farkas, A. C. Marosi, and Z. Balaton. Towards a powerful european dci based on desktop grids. *J. Grid Comput.*, 9(2):219–239, June 2011.

[16] C. Marco and et al. The glite workload management system. *Journal of Physics: Conference Series*, 219(6):062039, 2010.

[17] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble. Taverna, reloaded. pages 471–481, 2010.

[18] M. Morgan. GeniiJSDL Library. `http://www.cs.virginia.edu/~mmm2a/Software/GeniiJSDL/GeniiJSDLLibrary.pdf`. [Online; accessed 12-March-2013].

[19] M. Morgan. ByteIO Specification 1.0, Oct 2006.

[20] S. Newhouse. European Grid Infrastructure - an Integrated Sustainable Pan-European Infrastructure for Researchers in Europe, April 2011.

[21] M. Riedel, J. Rybicki, and A. Di Meglio. *Software for Distributed Systems - The EMI Product Portfolio*, page 7 p. Number PoS(EGICF12-EMITC2)082. 03/26/2012 - 03/30/2012 2012.

[22] B. Schuller, R. Menday, and A. Streit. A versatile execution management system for next-generation unicore grids. In *Euro-Par Workshops*, pages 195–204, 2006.

[23] B. Schuller and T. Pohlmann. Uftp: high-performance data transfer for unicore. *Proceedings of UNICORE Summit*, pages 135–142, 2011.

[24] A. Streit and et al. Unicore 6 - recent and future advancements. *Annals of telecommunications*, 65:757 – 762, 2010.

[25] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields. *Workflows for e-Science: Scientific Workflows for Grids*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[26] A. Turunen, , and et al. PRACE: Partnership for Advanced Computing in Europe Preparation of a Petascale Supercomputing Infrastructure for European Scientists. In N. V. Pogorelov, E. Audit, and G. P. Zank, editors, *Numerical Modeling of Space Plasma Flows, Astronum-2009*, volume 429 of *Astronomical Society of the Pacific Conference Series*, page 317, Sept. 2010.

Paper V

Facilitating Efficient Data Analysis of Remotely Sensed Images using Standards-based Parameter Sweep Models

M. S. Memon, G. Cavallario, M. Riedel, and H. Neukirchen.

Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2017), Fort Worth, Texas, USA, 23 July–28 July 2017, IEEE, pp. 3680–3683 (2017)

M. S. Memon is the main author of this publication. He has actively contributed to the implementation of the remote sensing case study. In this publication he produced Sections 1 (some paragraphs), 2.2, 3.3 and 4. He was also involved in developing the structure of the paper.

FACILITATING EFFICIENT DATA ANALYSIS OF REMOTELY SENSED IMAGES USING STANDARDS-BASED PARAMETER SWEEP MODELS

Shahbaz Memon[1,2]*, Gabriele Cavallaro*[1]*, Morris Riedel*[1,2]*,Helmut Neukirchen*[2]

[1] Jülich Supercomputing Centre, Forschungszentrum Jülich, Jülich, Germany
[2] School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland

ABSTRACT

Classification of remote sensing images often use Support Vector Machines (SVMs) that require an n-fold cross-validation phase in order to do model selection. This phase is characterized by sweeping through a wide set of parameter combinations of SVM kernel and cost parameters. As a consequence this process is computationally expensive but represents a principled way of tuning a model for better accuracy and to prevent overfitting together with regularization that is in SVMs inherently solved in the optimization. Since the cross-validation technique is done in a principled way also known as 'gridsearch', we aim at supporting remote sensing scientists in two ways. Firstly by reducing the time-to-solution of the cross-validation by applying state-of-the-art parallel processing methods because the sweep of parameters and cross-validation runs itself can be nicely parallelized. Secondly by reducing manual labour by automating the parallel submission processes since manually performing cross-validation is very time consuming, unintuitive, and error-prone especially in large-scale cluster or supercomputing environments (e.g., batch job scripts, node/core/task parameters, etc.).

Index Terms— Remote sensing, Support Vector Machine (SVM), cross-validation, High-Performance Computing (HPC), Parameter Sweep, Middleware.

1. INTRODUCTION

Remote sensing image datasets are an important source of information for many interdisciplinary applications addressing specific topics such as global and local climate change studies, ecological and environmental monitoring, or urban planning. These datasets can be complex (i.e., with high spectral, spatial, radiometric and temporal resolutions) and not reliable (e.g., equipment failure, noise), thus they can not be directly used by the applications. A powerful and automatic processing scheme for extracting reliable and valuable information must usually include feature engineering approaches (e.g., spatial information enhancement [1]) and data mining methods (e.g., classification including validation and regularization techniques). The classification of remote sensing images is the essential technique [2] used for extracting information. A relevant example is the separation of different types of land-cover classes. But the implicit complexity and dimensionality of sensed images are responsible for extensive limitations in classification. For instance problems arise when the classification methods require fast and highly scalable solutions for real-time applications (e.g., earthquake scenarios or glacial surges). Selected developments in High-Performance Computing (HPC) allow the classification algorithms to scale to large datasets [3]) while yielding high accuracy and results in a reasonable time.

Among the widely used remote sensing classifiers, Support Vector Machines (SVMs) [4] have often been found to be more effective in terms of classification accuracies and stability of parameter settings. However, SVMs are very demanding with respect to the processing time, e.g., in tuning the hyperplane parameters with cross-validation in order to perform model selection. The cross-validation phase is laborious if done manually by remote sensing scientists, as it requires re-runs of SVM optimization corresponding to a wide set of parameter combinations. Without any automation tool this phase takes a considerable amount of time and is also error-prone especially when performed on HPC machines with even more low-level technical parameters (e.g. number of cores, number of tasks per core, number of nodes, memory) that are typically machine-specific. To simplify the enhancement and usability of a parallelized cross-validation phase, we are proposing the adoption of a standards-based HPC middleware that handles an automated parameter sweep model which may consist of complex parametric representations, in a single n-fold cross-validation computational job.

2. BACKGROUND AND RELATED WORK

We shortly introduce the two basic concepts SVM and middleware that have been combined in this paper.

Thanks to Jülich Supercomputing Centre (JSC) for funding. This work was partly supported by NordForsk as part of the Nordic Center of Excellence (NCoE) eSTICC (eScience Tools for Investigating Climate Change at High Northern Latitudes). Correspondence: m.memon@fz-juelich.de

2.1. Support Vector Machines and piSVM

SVMs are one of the most powerful classification techniques today. The general idea of SVMs lies in separating training samples which belong to different classes by tracing maximum margin hyperplanes in the space where the samples are mapped [4]. Hence, SVMs only demand training samples close to the class boundary, and it is thus capable of handling high dimensional data even if only a small number of training samples is available. SVMs were originally introduced to solve linear classification problems. In order to generalize them to non-linear decision functions, i.e., more complex classes that are not linearly separable in the original feature space, the so-called *kernel trick* can be applied [5]. The sensitivity to the choice of the kernel and the cost parameters can be considered as the most important disadvantages of SVM.

We surveyed related work in [3] and have shown that despite the availability of many SVM parallelization strategies, only a very limited set of stable and scalable implementations is available as open source software. We improved a version of piSVM 1.2 [6] that was identified in [3] as a stable implementation since it is based on the libSVM library. We optimized it using better parallel processing techniques such as collective operations of the mature HPC standard Message Passing Interface (MPI). This implementation offers significant speed-ups for the cross validation, training and testing steps while maintaining the same accuracy as achieved when performing the classification with serial algorithms.

2.2. Standards-based Middleware and UNICORE

We use the middleware approach to abstract from low-level HPC machine details to make it easier for non-experts to submit and monitor parallel remote sensing classification jobs. To avoid vendor-locks, we rely on middleware based on standards for parallel job management and monitoring as well as data transfer and management functions [7] such as OGSA-BES [8], JSDL [9] and its extensions [10, 11]. The elements of the SVM cross-validation step can be captured through JSDL's parameter sweep extension [10] for building job executions of parametric nature. JSDL allows any part of job request to be parametrized, in particular application arguments. UNICORE [12] is a middleware which offers a seamless layer of abstraction to access different kinds of HPC environments and implements those standards. UNICORE supports this extension on its client and server tiers [13].

Required parameter sweep functionality is also implemented by gLite and gEclipse. But gLite's Workload Management System uses a proprietary approach called JDL to allow parametric job requests. gEclipse is a client-side application, though it supports the JSDL standard but not the full suite of extensions for HPC environments. We therefore have chosen the UNICORE middleware.

3. EXPERIMENTAL ANALYSIS

We validate our approach by measuring the performance of our parallel SVM implementation and the parameter sweep functionality using UNICORE with a remote sensing dataset.

3.1. Remote Sensing Dataset

The Indian Pines hyperspectral dataset [14] was acquired in June 1992 by the AVIRIS sensor over an agricultural site composed of fields with regular geometry and with a variety of crops. This data set represents a very challenging land-cover classification problem dominated by similar spectral classes and mixed pixels. The scene is made up of 1417×617 pixels (with spatial resolution 20 m) and 30 features, which were obtained with the methods described in [3].

3.2. Experimental Setup

For evaluating the performance of our parallel piSVM implementation, we compared it to the serial SVM implementation in MATLAB running on a laptop computer having one Intel Core i7-4710HQ 2.5 GHz CPU and 16 GB of RAM. The piSVM was executed on the JURECA [15] cluster where each compute node has two Intel Xeon E5-2680 v3 Haswell 12 core processors with 2.5 GHz and 128 to 512 GB of RAM. We deployed the piSVM application and the UNICORE server on the JURECA HPC cluster.

The evaluation was performed in two modes, with and without UNICORE middleware adoption. Fig. 1 depicts the steps required in the manual and in our automated UNICORE workflow methodology.

For evaluation of our application of parameter sweeps for the automated cross-validation, we have implemented a workflow shown in Fig. 1(b) while Fig. 1(a) shows the manual labour process by scientists when running a parallel cross-validation job. The workflow runs tThe whole cross-validation process as a single parametric job on piSVM application parameters and performs model selection by picking the best parameters according to estimated accuracies. The concerned application parameters for our case study are C and G with C being the cost as regularization parameter, and G the parameter of the choosen RBF kernel.

The manual workflow shown in Fig. 1(a) is rather low-level and thus advanced HPC system knowledge is required by a user, e.g. to access and monitor jobs. In the course of job management, the user prepares the environment on the cluster to create a job directory for each cross-validation parameter combination (Step 1). The required data then has to be supplied explicitly to the job environment (Step 2). The user then creates a job script which is hard-coded to the respective HPC machine batch system (Step 3) that is SLURM in our case. Once the job script is prepared, the user submits individual jobs separately or by means of a wrapper script (Step 4) that in turn requires sound UNIX knowledge. All

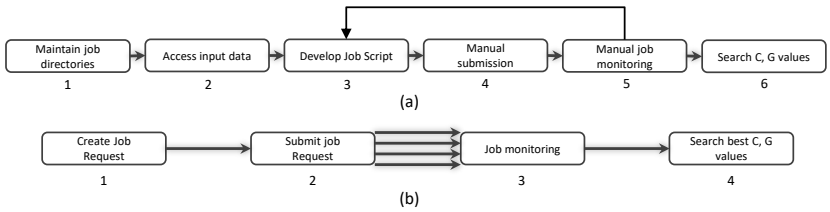


Fig. 1. Flowchart of the (a) manual and the (b) automatic method.

Table 1. Serial 10-fold cross-validation (MATLAB)

G / C	1	10	100	1000	10000
2	48.90 (18.8)	65.01 (19.6)	73.21 (20.1)	75.6 (22.5)	74.42 (21.2)
4	57.53 (16.8)	70.74 (13.9)	75.94 (13.5)	76.04 (14.0)	74.06 (15.6)
8	64.18 (18.3)	74.45 (15.0)	**77.00 (14.4)**	75.78 (14.7)	74.58 (14.9)
16	68.37 (23.2)	76.20 (21.9)	76.51 (20.7)	75.32 (19.6)	74.72 (19.7)
32	70.17 (34.5)	75.48 (34.8)	74.88 (34.1)	74.08 (34.0)	73.84 (38.8)

Table 2. Parallel 10-fold cross-validation (piSVM, 24 cores)

G / C	1	10	100	1000	10000
2	49.02 (7.3)	65.12 (8.6)	73.17 (13.5)	75.76 (22.5)	74.44 (33.0)
4	57.59 (7.4)	70.88 (8.9)	75.87 (11.6)	76.02 (14.7)	74.06 (17.9)
8	64.17 (7.9)	74.53 (9.3)	**77.02 (10.4)**	75.79 (11.3)	74.42 (12.2)
16	68.58 (9.8)	76.07 (10.6)	76.4 (10.9)	75.26 (11.2)	74.53 (11.3)
32	70.12 (13.9)	75.38 (14.3)	74.69 (14.6)	73.91 (14.6)	73.73 (14.6)

the jobs have to be monitored individually (Step 5). After the jobs are finished, then the user searches for the best C and G according to the accuracy somewhere in job output logs (Step 6). As part of cross-validation, many parameter combinations need to be tried, hence Steps 3, 4 and 5 are iterative; if there is no sophisticated script to automatically deal with parameter combinations, then individual job scripts and jobs have to be created for each parameter setting. All in all this process is error-prone and time consuming.

In the automated workflow, depicted in Fig. 1(b), the user creates a job request which conforms to the JSDL [9] [11] and Parameter Sweep [10] specification (Step 1). The job request is formulated as an XML instance in which the user can specify what parameters shall be iterated together with a pointer to the input data source. Once the job request is formalized, the next step is to execute the workflow. In this step, a remote request will be sent to a target server which interfaces the backend cluster, which is in our case JURECA (Step 2). The server validates the job requirements and then performs the resolution of the parameters to be processed before execution. During execution, the server will generate the required number of jobs; in our case, the parameter sweep equals the cartesian product of five C and five G values (=25 jobs). The input data set will also get transparently downloaded from the data source. Furthermore, the server monitors all the generated sweep jobs (Step 3), but these sweep jobs are hidden from the user, only one job is visible to her: the master job representing the sweep. From the result, the best C and G values can be obtained (Step 4).

3.3. Experimental Results

Tables 1 and 2 show the accuracies and the computation times (in minutes shown as value in parentheses) of cross-validation for the serial and the parallel SVM implementation, respectively. When comparing the tables, a significant speed-up was obtained for all parameter combinations while maintaining the accuracies. The best accuracy is marked in bold and indicates the optimal C and G parameter combination which is used in the training phase.

As can be seen, the cross-validation in the serial case is computationally intensive. The reason is that the training-validation is performed 10 times for each of the 25 combinations of the C and G parameters. The total processing time is 534.6 minutes. Because each partition set is independent, the cross-validation performed in parallel can achieve a significant speed up, thus reducing the overall processing time to 322.25 minutes using 24 cores. The biggest impact is shown when performing parallel and scalable cross-validation over the so-called 'gridsearch' as each step in the grid can be also performed in parallel. As a consequence, we implement a two-level parallelization of the cross-validation phase compared to serial MATLAB runs.

It should be noted that Table 2 shows the performance of all parameter combinations without any overhead of UNICORE middleware. In our experience, the use of UNICORE brings additional delay of approximately 2 minutes for completing the whole sweep. Thus, for every single sweep iteration, a delay of few seconds is introduced, which we think is not critical when keeping in mind that otherwise the whole process of cross-validation would involve multiple and time consuming and and error-prone manual user interactions which are now avoided.

In the manual sequence, the user has to engage in multiple steps, e.g. creating batch system specific job scripts (for all the parameters combinations), data management and transfer, and job monitoring. Debugging of failed sweeps may be very cumbersome. The automatic sequence is more high-level and prevents user from manually interacting with individual runs, except for creating and submitting the initial job request.

4. CONCLUSIONS

We conclude that one can obtain significant speed-ups of an automated cross-validation phase used in classification of remote sensing images by applying a parallel and scalable SVM approach. We further conclude that using a standard-based middleware that implements the concept of parameter sweeps for cross-validation runs significantly increases the productivity of scientists when using HPC machines for the analysis. The middleware approach allows not only reduces error-prone and time-consuming and tedious manual labour but also supports the re-use of the workflow on different HPC machines using other standards-based middleware.

In order to automate also other data analysis steps, we intend to extend our cross-validation workflow to include model generation and prediction phases which will directly use best parameters resulted from it.

The described approach has applications in many other remote sensing application areas. For our work, it is promising to apply it to determine calving fronts of glaciers [16] where we are already applying a UNICORE workflow to couple a continuum ice sheet model and a discrete element calving model [17].

5. REFERENCES

[1] G. Cavallaro, M. Dalla Mura, J. A. Benediktsson, and A. Plaza, "Remote Sensing Image Classification Using Attribute Filters Defined over the Tree of Shapes," *IEEE Transactions on Geoscience and Remote Sensing*, 2016.

[2] A. K. Maini and V. Agrawal, *Satellite Technology: Principles and Applications*, John Wiley & Sons, 3rd edition, 2014.

[3] G. Cavallaro, M. Riedel, M. Richerzhagen, J.A. Benediktsson, and A. Plaza, "On understanding big data impacts in remotely sensed image classification using support vector machine methods," *IEEE journal of selected topics in applied earth observations and remote sensing*, vol. 8, no. 10, pp. 4634–4646, 2015.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20(3), pp. 273–297, 1995.

[5] B. Schölkopf and A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.

[6] D. Brugger, "piSVM," website, 2014, `http://pisvm.sourceforge.net`.

[7] M. Shahbaz Memon, A. Shiraz Memon, M. Riedel, B. Schuller, D. Mallmann, B. Tweddell, A. Streit, S. van de Berghe, D. Snelling, V. Li, M. Marzolla, and P. Andreetto, "Enhanced resource management capabilities using standardized job management and data access interfaces within UNICORE Grids," in *International Conference on Parallel and Distributed Systems*. 2007, IEEE.

[8] I. Foster et al., "OGSA Basic Execution Service (BES), Version 1.0," Open Grid Forum GFD-R.108, Nov 2008.

[9] A. Anjomshoaa et al., "Job Submission Description Language (JSDL) Specification, Version 1.0," Open Grid Forum GFD-R.136, July 2008.

[10] M. Drescher et al., "JSDL Parameter Sweep Job Extension," Open Grid Forum GFD-R-P.149, May 2009.

[11] A. Savva, "JSDL SPMD Application Extension, Version 1.0," Open Grid Forum GFD-R-P.115, August 2007.

[12] A. Streit et al., "Unicore 6 recent and future advancements," *Annals of Telecommunications*, vol. 65, no. 11-12, pp. 757–762, 2010.

[13] Shahbaz Memon, S. Holl, B. Schuller, M. Riedel, and A. Grimshaw, "Enhancing the Performance of Scientific Workflow Execution in e-Science Environments by Harnessing the Standards Based Parameter Sweep Model," in *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (XSEDE '13)*. 2013, ACM.

[14] M.F. Baumgardner, L.L. Biehl, and D.A. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3," 2015, DOI:10.4231/R7RX991C.

[15] Jülich Supercomputing Centre, "JURECA," website, 2016, `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/JURECA_node.html`.

[16] J. A. Åström, D. Vallot, M. Schäfer, E.Z. Welty, S. O'Neel, T.C. Bartholomaus, Yan Liu, T.I. Riikilä, T. Zwinger, J. Timonen, and J.C. Moore, "Termini of calving glaciers as self-organized critical systems," *Nature Geoscience*, vol. 7, pp. 874–878, 2014.

[17] Shahbaz Memon, D. Vallot, T. Zwinger, and H. Neukirchen, "Coupling of a continuum ice sheet model and a discrete element calving model using a scientific workflow system," in *Geophysical Research Abstracts, Volume 19 – EGU General Assembly 2017*. 2017, Copernicus Publications, submitted.

References

[1] Genesis II. URL:http://genesis2.virginia.edu/wiki/Main/HomePage. [Online; Accessed 12 March 2013].

[2] Hierarchical Data Format version 5. http://www.hdfgroup.org/HDF5. [Online; Accessed 29 October 2018].

[3] JURECA. URL:http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/JURECA_node.html. [Online; Accessed 13 November 2018].

[4] Open Grid Forum. URL:http://ogf.org. [Online; Accessed 26 November 2018].

[5] The OASIS Web Services Interoperability (WS-I). http://www.ws-i.org/. [Online; Accessed 2 November 2018].

[6] UltraScan Laboratory Information Management System (USLIMS). URL:http://www.uslims.aucsolutions.com/. XSEDE Science Gateway. [Online; Accessed 29 October 2018].

[7] DCI Bridge Administrator Manual, Version 3.7.4. http://sourceforge.net/projects/guse/files/3.7.4/Documentation/, Nov 2015. MTA SZAKI. [Online; Accessed 29 December 2015].

[8] A. Anjomshoaa et al. Job Submission Description Language (JSDL) Specification, Version 1.0. Open Grid Forum, GFD-R.136, July 2008.

[9] A. Petzold et al. Global-scale atmosphere monitoring by in-service aircraft – current achievements and future prospects of the European Research Infrastructure IAGOS. *Tellus B: Chemical and Physical Meteorology*, 67(1):28452, 2015.

[10] A. Sim et al. The Storage Resource Manager Interface Specification, Version 2.2. Open Grid Forum, GFD-R-P.129, May 2008.

[11] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[12] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster. The Globus Striped GridFTP Framework and Server. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, pages 54–54, Washington, DC, USA, 2005. IEEE Computer Society. DOI:10.1109/SC.2005.72.

[13] T. Allweyer. *BPMN 2.0: : Introduction to the Standard for Business Process Modeling*. Books on Demand, 1st edition, May 2016.

[14] S. Andreozzi, S. Burke, F. Ehm, L. Field, G. Galang, D. Horat, B. Konya,

M. Litmaath, S. Memon, P. Millar, J. Navarro, and F. Paganelli. GLUE v.2.0 - Reference Realisation to LDAP Schema. Open Grid Forum, GFD-R-P.218, October 2013.

[15] S. Andreozzi, S. Burke, F. Ehm, L. Field, G. Galang, B. Konya, M. Litmaath, P. Millar, and J. Navarro. GLUE Specification v.2.0. Open Grid Forum, GFD-R-P.147, March 2009.

[16] S. Andreozzi, S. Burke, F. Ehm, L. Field, B. Konya, S. Memon, D. Meredith, J. Navarro, F. Paganelli, and W. Smith. GLUE v.2.0 - Reference Realization to XML Schema. Open Grid Forum, GFD-R-P.209, October 2013.

[17] S. B. Ardestani, C. J. Håkansson, E. Laure, I. Livenson, P. Stranák, E. Dima, D. Blommesteijn, and M. v. d. Sanden. B2share: An open escience data sharing platform. In *2015 IEEE 11th International Conference on e-Science*, pages 448–453, August 2015.

[18] T. Banks. Web Services Resource Framework (WSRF) - Primer v1.2. `URL:http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf`. OASIS 2006. [Online; Accessed 25 April 2019].

[19] M. Baumgardner, L. Biehl, and D. Landgrebe. 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3, 2015. DOI:10.4231/R7RX991C.

[20] G. Behrmann, D. Cameron, M. Ellert, J. Kleist, and A. Taga. ATLAS DDM integration in ARC. *Journal of Physics: Conference Series*, 119(6), 2008.

[21] M. Belshe, R. Peon, and M. Thompson. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, IETF, July 1995.

[22] C. Binstock, D. Peterson, M. Smith, M. Wooding, C. Dix, and C. Galtenberg. *The XML Schema Complete Reference*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[23] E. Brookes, W. Cao, and B. Demeler. A two-dimensional spectrum analysis for sedimentation velocity experiments of mixtures with heterogeneity in molecular weight and shape. *European Biophysics Journal*, 39(3):405–414, 2010.

[24] E. Brookes and B. Demeler. Genetic algorithm optimization for obtaining accurate molecular weight distributions from sedimentation velocity experiments. In C. Wandrey and H. Cölfen, editors, *Analytical Ultracentrifugation VIII*, volume 131 of *Progress in Colloid and Polymer Science*, pages 33–40. Springer Berlin Heidelberg, 2006.

[25] D. Brugger. piSVM. `URL:http://pisvm.sourceforge.net`, 2014. [Online; Accessed November, 8th 2018].

[26] M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA, 2003.

[27] G. Cavallaro, M. Dalla Mura, J. A. Benediktsson, and A. Plaza. Remote Sensing Image Classification Using Attribute Filters Defined over the Tree of Shapes. *IEEE Transactions on Geoscience and Remote Sensing*, 2016.

[28] S. Chandrasekaran and G. Juckeland. *OpenACC for Programmers: Concepts and Strategies*. Addison-Wesley Professional, 1st edition, 2017.

[29] C. Chang, G. Czajkowski, T. von Eicken, and C. Kesselman. Evaluating the Performance Limitations of MPMD Communication. In *SC '97: Proceedings of the 1997 ACM/IEEE Conference on Supercomputing*, pages 11–11, November

1997.

[30] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at URL:http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[31] N. P. Chue Hong, M. Drescher, A. Krause, M. S. Memon, and M. Morgan. OGSA® ByteIO Implementations – Experiences Document. Open Grid Forum, GFD-E.146, March 2009.

[32] J. L. Cole, J. W. Lary, T. P. Moody, and T. M. Laue. Analytical ultracentrifugation: Sedimentation velocity and sedimentation equilibrium. In *Biophysical Tools for Biologists, Volume One: In Vitro Techniques*, volume 84 of *Methods in Cell Biology*, pages 143 – 179. Academic Press, 2008.

[33] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995. DOI:10.1007/BF00994018.

[34] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design (International Computer Science)*. Addison-Wesley Longman, Amsterdam, 4th rev. ed. edition, 2005.

[35] D. Box et al. Simple Object Access Protocol (SOAP) 1.1. URL:http://www.w3.org/TR/2000/NOTE-SOAP-20000508/. [Online; Accessed December, 29th 2015].

[36] D. Box et al. Web Services Addressing (WS-Addressing). URL:http://www.w3.org/Submission/ws-addressing/. [Online; Accessed 29 December 2015].

[37] D. J. Eisenstein et al. SDSS-III: Massive Spectroscopic Surveys of the Distant Universe, the Milky Way, and Extra-solar Planetary Systems. *The Astronomical Journal*, 142(3):72, August 2011. DOI:10.1088/0004-6256/142/3/72.

[38] F. Darema. The spmd model: Past, present and future. In Y. Cotronis and J. Dongarra, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 1–1, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[39] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[40] DEEP Consortium. DEEP-EST - Towards a modular supercomputing architecture. URL:https://cordis.europa.eu/project/rcn/210094_en.html. [Online; Accessed 8 November 2018].

[41] B. Demeler. Ultrascan - a comprehensive data analysis software package for analytical ultracentrifugation experiments. In D. J. Scott, S. E. Harding, and A. J. Rowe, editors, *Analytical Ultracentrifugation: Techniques and Methods*, pages 210–230p. The Royal Society of Chemistry, 2005.

[42] B. Demeler and E. Brookes. Monte carlo analysis of sedimentation experiments. *Colloid and Polymer Science*, 286(2):129–137, 2008.

[43] B. Demeler, R. Singh, M. Pierce, E. H Brookes, S. Marru, and B. Dubbs. Ultrascan gateway enhancements: in collaboration with TeraGrid advanced user support. In *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, TG '11, pages 34:1–34:8, New York, NY, USA, 2011. ACM.

[44] M. Diepenbroek, H. Grobe, M. Reinke, U. Schindler, R. Schlitzer, R. Sieger, and G. Wefer. PANGAEA - an information system for environmental sciences. *Computers & Geosciences*, 28(10):1201–1210, 2002.

[45] M. Drescher, A. Anjomshoaa, G. Williams, and D. Meredith. JSDL Parameter Sweep Job Extension. Open Grid Forum, GFD-R-P.149, May 2009.

[46] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. L. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen. Advanced Resource Connector Middleware for Lightweight Computational Grids. *Future Gener. Comput. Syst.*, 23(2):219–240, 2007. DOI:10.1016/j.cam.2006.05.008.

[47] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[48] F. Bachmann et al. XSEDE Architecture Level 3 Decomposition. `URL:http://hdl.handle.net/2142/45117`, Dec 2012. [Online; Accessed 10 January 2018].

[49] I. Foster. Globus online: Accelerating and democratizing science through cloud-based services. *IEEE Internet Computing*, 15(3):70–73, 2011.

[50] I. Foster, K. Czajkowski, D. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke. Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF. *Proceedings of the IEEE*, 93(3):604–612, March 2005. 10.1109/JPROC.2004.842766.

[51] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, and M. Theimer. OGSA Basic Execution Service, Version 1.0. GWD-R.108, Open Grid Forum, November 2008.

[52] I. Foster, M. Terry, and D. Snelling. OGSA WSRF Basic Profile 1.0. Open Grid Forum, GFD-R-P.072, May 2006.

[53] A. Freier, P. Karlton, and P. Kocher. The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101, IETF, January 2011.

[54] P. Fuhrmann and V. Gülzow. dCache, Storage System for the Future. In W. Nagel, W. Walter, and W. Lehner, editors, *Euro-Par 2006 Parallel Processing*, volume 4128 of *Lecture Notes in Computer Science*, pages 1106–1113. Springer Berlin Heidelberg, 2006.

[55] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[56] W. Gentzsch, D. Girou, A. Kennedy, H. Lederer, J. Reetz, M. Riedel, A. Schott, A. Vanni, M. Vazquez, and J. Wolfrat. DEISA—Distributed European Infrastructure for Supercomputing Applications. *Journal of Grid Computing*, 9(2):259–277, June 2011. DOI:10.1007/s10723-011-9183-2.

[57] H. Gjermundrod, M. D. Dikaiakos, M. Stumpert, P. Wolniewicz, and H. Kornmayer. g-Eclipse - an integrated framework to access and maintain Grid resources. In *Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing*, GRID '08, pages 57–64, Washington, DC, USA, 2008. IEEE Computer Society. DOI:10.1109/GRID.2008.4662783.

[58] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, A. Merzky, J. Shalf, and

C. Smith. A Simple API for Grid Applications (SAGA). Open Grid Forum. Open Grid Forum, GFD-R.90, November 2008.

[59] M. Götz, C. Bodenstein, and M. Riedel. HPDBSCAN: Highly Parallel DBSCAN. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, MLHPC '15, pages 2:1–2:10, New York, NY, USA, 2015. ACM. DOI:10.1145/2834892.2834894.

[60] A. Grimshaw, C. Koeritz, B. Demuth, and S. Memon. GenesisII Libraries. `URL:svn://svn.xcg.virginia.edu:9002/GENREPO/GenesisII/trunk/libraries`, 2016. [Online; Accessed 30 April 2019].

[61] A. Grimshaw, M. Morgan, and A. Kalyanaraman. GFFS - The XSEDE Global Federated File System. *Parallel Processing Letters*, 23(02):1340005, 2013.

[62] F. Hedman, M. Riedel, P. Mucci, G. Netzer, A. Gholami, M. S. Memon, A. S. Memon, and Z. A. Shah. Benchmarking of integrated OGSA-BES with the grid middleware. In *Euro-Par 2008 Workshops - Parallel Processing*, pages 113–122, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[63] P. Kacsuk and G. Sipos. Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal. *Journal of Grid Computing*, 3(3-4):221–238, 2005. DOI:10.1007/s10723-005-9012-6.

[64] K. Kambatla, G. Kollias, V. Kumar, and A. Grama. Trends in big data analytics. *Journal of Parallel and Distributed Computing*, 74(7):2561 – 2573, 2014. Special Issue on Perspectives on Parallel and Distributed Processing.

[65] K. W. Kintigh and A. J. Ammerman. Heuristic Approaches to Spatial Analysis in Archaeology. *American Antiquity*, 47(1):31–63, 1982.

[66] R. Kune, P. K. Konugurthi, A. Agarwal, R. R. Chillarige, and R. Buyya. The Anatomy of Big Data Computing. *Softw. Pract. Exper.*, 46(1):79–105, 2016. DOI:10.1002/spe.2374.

[67] K. Kurowski, P. Dziubecki, P. Grabowski, M. Krysiński, T. Piontek, and D. Szejnfeld. Easy Development and Integration of Science Gateways with Vine Toolkit. In *eScience on Distributed Computing Infrastructure - Volume 8500*, pages 147–163, New York, NY, USA, 2014. Springer-Verlag New York, Inc. DOI:10.1007/978-3-319-10894-0_11.

[68] E. Laure, S. M. Fisher, A. Frohner, C. Grandi, P. Z. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, F. Hemmer, A. Di Meglio, and A. Edlund. Programming the Grid with gLite. Technical Report EGEE-TR-2006-001, CERN, Geneva, March 2006.

[69] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. DOI:10.1109/TIT.1982.1056489.

[70] M. Lottor. Simple File Transfer Protocol. RFC 913, IETF, July 1984.

[71] M. Götz et al. HPDBSCAN Benchmark test files. `URL:https://b2share.eudat.eu/records/7f0c22ba9a5a44ca83cdf4fb304ce44e`, 2015. DOI:10.23728/b2share.7f0c22ba9a5a44ca83cdf4fb304ce44e.

[72] I. Mandrichenko, W. Allcock, and T. Perelmutov. GridFTP v2 Protocol Description. Open Grid Forum, GFD-R-P.047, May 2005.

[73] J. McAffer and J.-M. Lemieux. *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java(TM) Applications*. Addison-Wesley Professional, 2005.

[74] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in ner-

vous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, December 1943. DOI:10.1007/BF02478259.

[75] S. McGough, W. Lee, and S. Das. A standards based approach to enabling legacy applications on the grid. *Future Generation Computer Systems*, 24(7):731 – 743, 2008.

[76] M. S. Memon, A. S. Memon, M. Riedel, B. Schuller, D. Mallmann, B. Tweddell, A. Streit, S. van de Berghe, D. Snelling, V. Li, M. Marzolla, and P. Andreetto. Enhanced resource management capabilities using standardized job management and data access interfaces within UNICORE Grids. In *2007 International Conference on Parallel and Distributed Systems*, pages 1–6, December 2007. DOI:10.1109/ICPADS.2007.4447834.

[77] S. Memon. UNICORE Client Wrapper. `URL:https://mvnrepository.com/artifact/eu.unicore`, 2014. [Online; Accessed 10 December 2018].

[78] S. Memon. JSDL Sweep Library. `URL:https://sourceforge.net/p/unicore/svn/HEAD/tree/contributions/jsdl-sweep/tags/jsdl-sweep-1.1.0/`, 2015. [Online; Accessed 13 September 2018].

[79] A. Merzky, M. Santcroos, S. Fischer, and O. Weidner. SAGA API Bindings: Python. Open Grid Forum, GFD-R-P.211, November 2013.

[80] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard Version 3.1, June 2015.

[81] M. A. Miller, W. Pfeiffer, and T. Schwartz. The CIPRES Science Gateway: Enabling High-impact Science for Phylogenetics Researchers with Limited Resources. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond*, XSEDE '12, pages 39:1–39:8, New York, NY, USA, 2012. ACM. DOI:s10.1145/2335755.2335836.

[82] T. M. Mitchell. Machine learning and data mining. *Commun. ACM*, 42(11):30–36, 1999. DOI:10.1145/319382.319388.

[83] M. Morgan and A. Grimshaw. Chapter 8 - High-Throughput Computing in the Sciences. In M. L. Johnson and L. Brand, editors, *Computer Methods Part B*, volume 467 of *Methods in Enzymology*, pages 197 – 227. Academic Press, 2009. DOI:10.1016/S0076-6879(09)67008-7.

[84] M. Morgan, A. Grimshaw, and O. Tatebe. RNS Specification 1.1. Open Grid Forum, GFD-R-P.171, December 2010.

[85] J. Natarajan, R. Bruchez, S. Shaw, and M. Coles. *XQuery and XPath*, pages 355–398. Apress, Berkeley, CA, 2012.

[86] J. Novotny, M. Russell, and O. Wehrens. Gridsphere: A portal framework for building collaborations: Research articles. *Concurr. Comput. : Pract. Exper.*, 16(5):503–513, 2004. DOI:10.1002/cpe.v16:5.

[87] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the Grid: MyProxy. In *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on Intelligent Control*, pages 104–111, 2001. DOI:10.1109/HPDC.2001.945181.

[88] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Sen-

ger, R. Stevens, A. Wipat, and C. Wroe. Taverna: Lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006. DOI:10.1002/cpe.993.

[89] OpenMP Architecture Review Board. OpenMP Application Program Interface Version 4.5, 2015.

[90] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017.

[91] M. A. Patwary, D. Palsetia, A. Agrawal, W.-k. Liao, F. Manne, and A. Choudhary. A New Scalable Parallel DBSCAN Algorithm Using the Disjoint-set Data Structure. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 62:1–62:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[92] M. J. Perri and S. H. Weber. Web-Based Job Submission Interface for the GAMESS Computational Chemistry Program. *Journal of Chemical Education*, 91(12):2206–2208, 2014. DOI:10.1021/ed5004228.

[93] M. Pierce, S. Marru, L. Gunathilake, T. A. Kanewala, R. Singh, S. Wijeratne, C. Wimalasena, C. Herath, E. Chinthaka, C. Mattmann, A. Slominski, and P. Tangchaisin. Apache Airavata: Design and Directions of a Science Gateway Framework. In *2014 6th International Workshop on Science Gateways*, pages 48–54, June 2014. DOI:10.1109/IWSG.2014.15.

[94] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD Rec.*, 29(2):427–438, 2000. DOI:10.1145/335191.335437.

[95] L. Rokach and O. Maimon. *Data Mining With Decision Trees: Theory and Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2nd edition, 2014.

[96] M. Russell, P. Dziubecki, P. Grabowski, M. Krysiński, T. Kuczyński, D. Szjenfeld, D. Tarnawczyk, G. Wolniewicz, and J. Nabrzyski. The Vine Toolkit: A Java Framework for Developing Grid Applications. In *Parallel Processing and Applied Mathematics*, pages 331–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. DOI:10.1007/978-3-540-68111-3_35.

[97] A. Savva. JSDL SPMD Application Extension, Version 1.0. GWD-R-P.115, Open Grid Forum, August 2007.

[98] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[99] B. Schuller, B. Konya, O. Smirnova, A. Konstantinov, M. S. Andersen, M. Riedel, S. Memon, S. Memon, L. Zangrando, M. Sgaravatto, and E. Frizziero. European Middleware Initiative Execution Service Version 2.0. GFD-I.210, Open Grid Forum, March 2013.

[100] B. Schuller, R. Menday, and A. Streit. A Versatile Execution Management System for Next-Generation UNICORE Grids. In *Euro-Par Workshops*, pages 195–204, 2006. DOI:10.1007/978-3-540-72337-0_19.

[101] B. Schuller and T. Pohlmann. UFTP: high-performance data transfer for UNICORE. *Proceedings of UNICORE Summit*, pages 135–142, 2011.

[102] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage*

Systems and Technologies (MSST), MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[103] W. Smith, D. Meredith, S. Memon, and J. Navarro. GLUE v.2.0 - Reference Realization to JSON Schema. Open Grid Forum, GFD-R-P.219, May 2016.

[104] The ATLAS Collaboration and et al. The ATLAS Experiment at the CERN Large Hadron Collider. *Journal of Instrumentation*, 3(08):S08003, 2008.

[105] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. Xsede: Accelerating scientific discovery. *Computing in Science Engineering*, 16(5):62–74, September 2014. DOI:10.1109/MCSE.2014.80.

[106] A. Turunen, J.-P. Nominé, F. Robin, D. Erwin, H. Huber, A. Berg, R. Murri, and A. Simpson. PRACE: Partnership for Advanced Computing in Europe Preparation of a Petascale Supercomputing Infrastructure for European Scientists. In N. V. Pogorelov, E. Audit, and G. P. Zank, editors, *Numerical Modeling of Space Plasma Flows, Astronum-2009*, volume 429 of *Astronomical Society of the Pacific Conference Series*, page 317, 2010.

[107] S. Wang. A CyberGIS Framework for the Synthesis of Cyberinfrastructure, GIS, and Spatial Analysis. *Annals of the Association of American Geographers*, 100(3):535–557, 2010. DOI:10.1080/00045601003791243.

[108] G. Wasson. Interoperability Experiences with the High Performance Computing Basic Profile (HPCBP), Version 1.0. Open Grid Forum, GFD-E.146, March 2008.

[109] G. Wasson and M. Humphrey. HPC File Staging Profile, Version 1.0. Open Grid Forum, GFD-R-P.135, June 2008.

[110] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 4th edition, 2015.

[111] P. Wieder, A. Papaspyrou, S. Andreas, D. Fellows, and S. Memon. Activity Instance Container Specification Version 1.0. GWD-R-P.203, Open Grid Forum, May 2013.

[112] T. Ylonen and C. Lonvick. Simple File Transfer Protocol. RFC 4251, IETF, January 2006.

[113] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, 2016. DOI:10.1145/2934664.