ENVRI^{PLUS} DELIVERABLE

D7.3

PERFORMANCE OPTIMISATION FOR ENVIRONMENTAL RI PROJECTS: SYSTEM DESIGN

WORK PACKAGE 7—DATA PROCESSING AND ANALYSIS

LEADING BENEFICIARY: UNIVERSITY OF AMSTERDAM

Author(s)	Beneficiary/Institution
Paul Martin, Zhiming Zhao	University of Amsterdam
Markus Stocker, Robert Huber	University of Bremen
Jani Heikkinen, Aleksi Kallio	CSC

Accepted by: Leonardo Candela

Deliverable type: REPORT

Dissemination level: PUBLIC

Deliverable due date: 31.12.2016/M20

Actual Date of Submission: 25.1.2017/M21

A document of the ENVRI^{PLUS} project—http://www.envriplus.eu
This document has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 654182.

Abstract

This is the first deliverable for **Task 7.2, "Performance optimisation for big data science"**, which addresses the optimisation strand of the ENVRIplus **Data for Science** theme. It provides an initial design vision and roadmap for the next 22 months; a successor deliverable **D7.4, "Performance optimisation services for environmental ESFRI projects: prototype"**, will be produced in 2018, detailing the actual technologies that will result from the task.

'Optimisation' covers a broad spectrum of possibilities. The focus of the optimisation task however is principally on the use of micro-services to optimise distributed data access, delivery and processing on e-infrastructures used by research infrastructures and their respective communities, allowing them to perform the kind of investigations of data that push the boundaries of what was previously achievable with the resources available.

This deliverable (**D7.3**) provides four main contributions. Firstly, it describes a general design vision for development of optimisation services, in particular noting how this vision relates to the various concurrent activities of the Data for Science theme. Secondly, it details some of the preliminary work that has already been performed or is in development by contributors to the task in order to provide the ENVRIplus community with an early indication of the kind of practical investigations being performed within this topic area. Thirdly, a provisional roadmap for the task is provided, sketching the shape of the task and the likely interaction points with different ENVRIplus activities up until the completion of the task in late 2018. Finally, a few recommendations are distilled from the work done so far, and presented in the summary section at the end of deliverable.

Project internal reviewer(s)

Project internal reviewer(s)	Beneficiary/Institution
Thierry Carval	Ifremer
Yin Chen	EGI

Document history

Date	Version
31/10/16	Outline of deliverable produced for approval by task members and theme leadership.
5/12/16	First draft of deliverable produced for internal review.
28/12/16	Revised draft based on internal review feedback.
23/1/17	Final approved draft.

Document amendment procedure

Amendments, comments and suggestions should be sent to Paul Martin (p.w.martin@uva.nl).

Project summary

ENVRIplus is a Horizon 2020 project bringing together Environmental and Earth System Research Infrastructures, projects and networks together with technical specialist partners to create a more coherent, interdisciplinary and interoperable cluster of Environmental Research Infrastructures across

Europe. It is driven by three overarching goals: 1) promoting cross-fertilization between infrastructures, 2) implementing innovative concepts and devices across RIs, and 3) facilitating research and innovation in the field of environment for an increasing number of users outside the RIs.

ENVRIplus aligns its activities to a core strategic plan where sharing multi-disciplinary expertise will be most effective. The project aims to improve Earth observation monitoring systems and strategies, including actions to improve harmonization and innovation, and generate common solutions to many shared information technology and data related challenges. It also seeks to harmonize policies for access and provide strategies for knowledge transfer amongst RIs. ENVRIplus develops guidelines to enhance transdisciplinary use of data and data-products supported by applied use-cases involving RIs from different domains. The project coordinates actions to improve communication and cooperation, addressing Environmental RIs at all levels, from management to end-users, implementing RI-staff exchange programs, generating material for RI personnel, and proposing common strategic developments and actions for enhancing services to users and evaluating the socio-economic impacts.

ENVRIplus is expected to facilitate structuration and improve quality of services offered both within single RIs and at the pan-RI level. It promotes efficient and multi-disciplinary research offering new opportunities to users, new tools to RI managers and new communication strategies for environmental RI communities. The resulting solutions, services and other project outcomes are made available to all environmental RI initiatives, thus contributing to the development of a coherent European RI ecosystem.

Table of contents

1 Introduction

The integration of data from a diversity of sources is a necessary component of system-level science [9]. The ESFRI projects[1] represented by ENVRIplus are intended to support such integration across the environmental and earth sciences. Beyond the larger consolidated data centres however, there is a great variety of long-tail data in small datasets scattered across many sites, and the velocity of data gathering (at all scales) continues to increase. Investigations of research data require not only access to those distributed data sources, but also reliable access to e-infrastructure onto which that data can be staged and processed. To ensure that this access is available to as broad a community of researchers as possible, there are a number of important issues to resolve, many of which are directly addressed by ENVRIplus, including administrative and legal issues. Here however we concern ourselves with some of the *technical* issues: for example how to efficiently transport large datasets over the network, how to efficiently provision (virtual) infrastructure to support complex workflows, how to steer data processing activities at runtime, and when best to preserve or cache intermediate data that might serve to reduce unnecessary process replication or data traffic.

One particular concern is how low-level configuration of e-infrastructure influences the quality of service or experience had by researchers engaging with research infrastructure, which in turn impacts their productivity and willingness to engage in ever more complex investigations of research data. Another concern is the process by which data-driven investigations and other research applications are specified, suitable host infrastructure is customised, and then the resulting application and infrastructure is steered during execution time.

This report is concerned with conceptual design of a system or methodology for addressing such concerns and others within the topic of 'optimisation' as the first deliverable of Task 7.2 of the ENVRIplus project.

1.1 Task overview

Within ENVRIplus, **Task 7.2 "Performance optimisation for big data sciences"** is concerned with characterising the availability and scope of data and computational resources provided by research infrastructures (RIs), and with the development of tools and services for optimising the performance of experiments and other data-driven interactions executed via those RIs. More precisely, the task's scope encompasses investigations into the optimisation of data, resource and service configuration on underlying e-infrastructure, and the scheduling of tasks on that e-infrastructure; part of this relates to the use of service-level agreements (SLAs) to establish the quality of service (QoS) that can be expected from any given e-infrastructure or federation thereof. This task is heavily embedded in the cross-cutting activities within the ENVRIplus "Data for Science" theme, especially those of Work Package 5 "Reference model guided RI design", as will be described in greater detail in section 2. It should also work in tandem with concurrent activities in other work packages in the theme, as well as its sister task, Task 7.1 "Interoperable Data Processing, Monitoring and Diagnosis".

The following specific actions are specifically ascribed to Task 7.2 by the ENVRIplus description of work:

- Provide an effective mapping between *application-level* performance and experience requirements, and *infrastructure-level* attributes for computing, storage and networking.
- Define test-bed requirements for any components produced, and identify the conditions under which tools and services can be deployed.
- Prototype components to assist with the optimisation of data movement and processing in ENVRIplus RIs such as EISCAT_3D, EMSO and EPOS.
- Investigate the use of tools for large-scale data analysis (such as Apache Hadoop, Spark and Storm) in the context of e-infrastructures provided by initiatives like EGI and EUDAT.

[1] http://ec.europa.eu/research/esfri/

The main role of this deliverable is to provide a design direction for Task 7.2 that will address both the overall vision of the task and eventually each of the specific actions described above, as well as provide a roadmap for how that direction will be followed through on in the next 22 months (the remaining lifespan of the task). In addition, a survey of initial development conducted within the task is also provided in order to showcase the work already done and provide the community with a perspective on the practical investigations being embarked upon.

1.2 Motivation

Data-centric approaches play an increasing role in scientific investigations. Large corpora of information, gathered 'in the field' or generated within laboratories, all requiring exploration and analysis, much of which must be stored, curated and made accessible to researchers. Increasingly important is the construction of specialist research infrastructure to manage the data and accompanying models, tools and services. As the complexity and diversity of research infrastructure grows however, there is a need for better tools to navigate and integrate the research assets provided—to provide investigators with a unified (virtual) research environment. In addition, the requirements of compute-intensive processing and the expense of moving large datasets across networks is leading to a greater focus on the underlying e-infrastructures provided by the investigator, the research infrastructure or by third parties. Initiatives such as EUDAT[2] (for generic data-oriented services such as storage and discovery) and EGI[3] (for common compute, storage and network infrastructure) push to make utility computing accessible to as broad a community as possible so as to promote collaboration and interdisciplinary research in Europe and beyond.

Optimisation of infrastructure depends on a certain degree of insight into the processes that are executed within an infrastructure. This insight is principally drawn from human expertise, but is generally realised via one of three methods:

- The designer responsible for implementing a particular tool or service within an infrastructure embeds their own understanding of how the infrastructure needs to operate in their implementations. This method is necessary for constructing intuitive and useful services; engineers and designers should absolutely apply their knowledge of the domain to create effective solutions, but must also consider the general applicability of their modifications and the resources needed to realise optimality under specific circumstances.
- The investigator using the infrastructure is given the tools to directly configure the infrastructure based on their own experience and knowledge of the task they are trying to perform. This can also be done by proxy, whereby the engineers responsible for maintaining the infrastructure configure it on their behalf. This method provides the greatest flexibility for users, but is also the most demanding. It is certainly possible and appropriate to provide a certain degree of configurability with data processing services, albeit with the caveat that casual users should not be confronted with too much technical detail. This method often curtails optimisation of resources for multiple applications that may compete for resources however.
- Experts encode their expertise as knowledge stored within or accessible to the infrastructure; autonomous systems within the infrastructure then use that knowledge to configure the infrastructure in accordance with the activities being currently performed. It is this method that is of most interest in the context of constructing interoperable architectures for environmental science research infrastructure solutions. The ability to assert domain-specific information explicitly in generic architecture and permitting the system to reconfigure itself based on the current global context is potentially very powerful. This method is however the most demanding in terms of fundamental design and the existence and adoption of standards for knowledge representation and reasoning.

In ENVRIplus, the goal is to provide shared services that can interact with the majority of RIs and e-infrastructure platforms to provide workable solutions to common problems. Key to this is the use of service catalogues that describe the services and research assets (data, tools, instruments, *etc.*)

[2] https://www.eudat.eu/

[3] https://www.egi.eu/

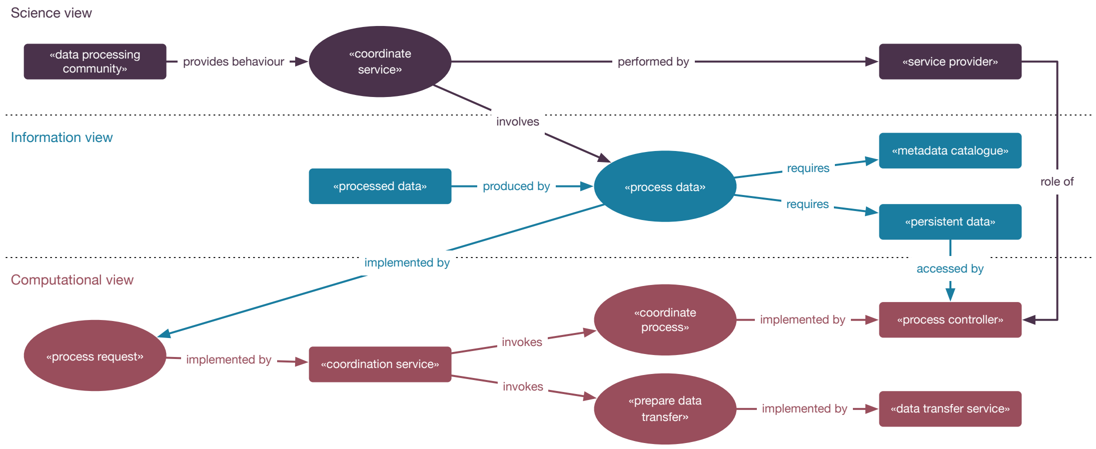

Figure 1: Sample of reference Model objects involved in process coordination.

available to researchers. Such catalogues, and the necessary use of standardised metadata to make them feasible, represent a particular form of information encoding into architecture. Alongside similar elements such as semantic descriptions and knowledge bases, the provision of catalogues produces the knowledge architecture needed to realise optimisation of processes using the third method above.

Task 7.2 is one of the spearhead activities in ENVRIplus for investigating how to build common, interoperable services for environmental science RIs. In particular, it exists to explore how to best make use of e-infrastructure to host data and orchestrate processes on behalf of RIs and their respective communities. This entails the development of a small number of prototype services that can be placed at specific strategic points in the computational research environment, but also some more experimental investigation of the use of knowledge embedded in the system to guide automated planning and reasoning about specific research application workflows.

The optimisation task is therefore intended to address a number of different questions, such as:

- How can we customise e-infrastructure for different kinds of computational research investigation?
- Can the customisation of e-infrastructure be partly or wholly automated?
- How can we best infer the requirements on e-infrastructure from a formal investigation design (*e.g.* expressed as a process workflow)?
- How can we use virtualised infrastructure to reduce the amount of unnecessary data transfer for data processing tasks?
- How can we make use of large-scale data processing tools while minimising the burden on investigators of configuring and managing their deployment?

To answer these questions, a number of kinds of optimisation *microservice* are being proposed and investigated. These services are intended to test techniques and technologies that may prove useful to future RI and e-infrastructure development, while being compatible with the broader architectural framework being promoted by the Data for Science theme in ENVRIplus in general. While a complete set of answers is unlikely to arise over the course of this project alone, some useful guidance, backed by operational prototypes, is expected to be produced for the benefit of the RIs in ENVRIplus.

As with all development tasks in the 'Data for Science' theme, Task 7.2 is *reference model guided*. The ENVRI Reference Model[4] clearly illustrates the complexity of the data handling pipeline, identifying a number of computational services required to realise the discovery, movement and processing of data. Such services include *coordination services* for oversight and *process controllers* for managing individual processes, which are needed for staging data and process coordination in general. The Reference Model also defines a number of roles and behaviours involved in data processing, such

[4] http://envri.eu/rm

as *service providers* needed for service coordination, and defines specific information objects that are required in a processing context, such as *persistent data* and *metadata catalogues*, as shown in Figure 1.

What the Reference Model does *not* define however is how best to realise these concepts and their interactions in practice. Within the theme, it is the role of the model architecture (produced by Task 5.4) to provide guidance as to how to realise these concepts in a manner that supports interoperability and shared common solutions. One of the objectives of Task 7.2 is to build upon the reference model and the model architecture to address specifically questions of 'optimal' implementation by investigating mechanisms for managing process coordination, data staging and e-infrastructure customisation that can then be repurposed as generic recommendations to the environmental science RIs in ENVRIplus and beyond.

In Section 2 we examine all of the above concerns in more detail, starting with the requirements raised by the community and moving on to their relationship with the topics addressed by the ENVRIplus project.

1.3 Layout

The remainder of this document is laid out as follows:

Section 2 "Vision and system design" is where we place the work of Task 7.2 within the large context of the ENVRIplus Data for Science theme, establishing the relationships that exist with the other concurrent tasks in Work Packages 5–9. In this section we draw upon the ENVRI reference model, semantic linking framework, and model architecture for producing interoperable solutions in order to provide a general depiction of how optimisation services fit into the ENVRIplus vision for environmental science RIs.

Section 3 "Current development" details some of the technical development and practical investigations that have already been carried out within ENVRIplus that contribute to Task 7.2. In particular, we examine the problem of planning and provisioning customised virtual infrastructures for specific research applications, a key challenge facing the practical integration of generic e-infrastructure into tailored research environments. We also look into certain useful technologies that can be deployed on e-infrastructure to drive large-scale data processing.

Section 4 "Roadmap" provides a provisional schedule for Task 7.2 activities, addressing the interaction points with other concurrent tasks, particularly the release of critical design documents to which our activities should comply and the release of key demonstrators to which our activities should contribute.

Section 5 "Summary" wraps up the deliverable, summarising its main points and contribution to the project, and providing a number of technical recommendations regarding future development in the optimisation area.

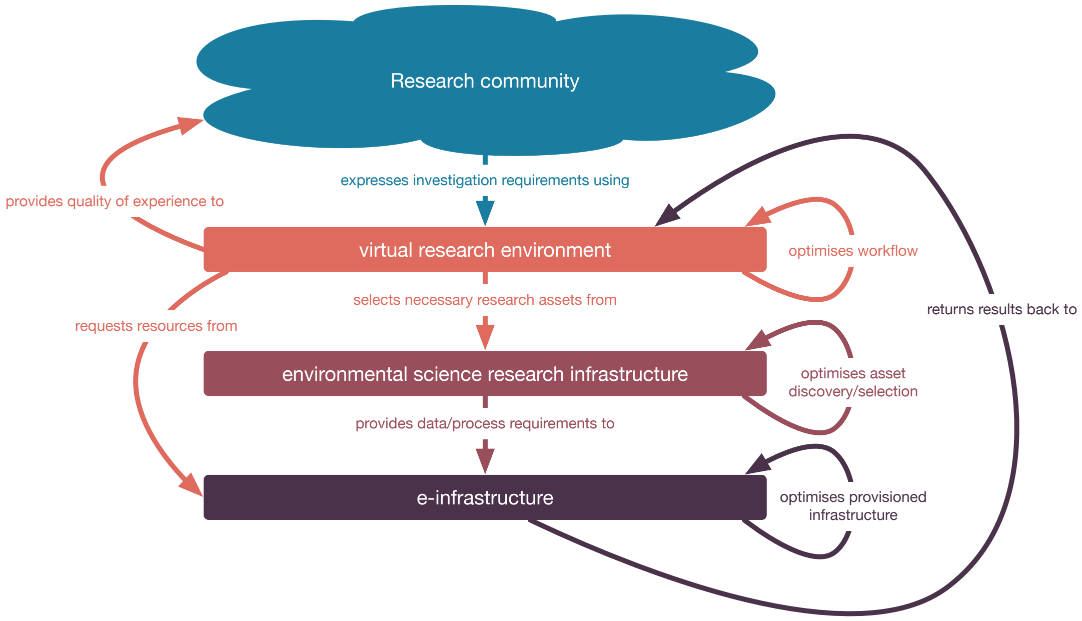
Research community
expresses investigation requirements using
provides quality of experience to
virtual research environment
optimises workflow
selects necessary research assets from
requests resources from
returns results back to
environmental science research infrastructure
optimises asset discovery/selection
provides data/process requirements to
e-infrastructure
optimises provisioned infrastructure

Figure 2: Different levels of computational infrastructure acquire and respond to different requirements, but all affect the 'optimality' of data-driven investigations.

2 Vision and system design

Environmental science now relies on the acquisition and analysis of great quantities of data gathered from a range of different sources. Data acquired might be consolidated into a few very large datasets, or dispersed across many smaller datasets; data may be ingested in batch or accumulated over a prolonged period of monitoring. Regardless of the acquisition methodology, in order to *use* this wealth of data effectively, it is important that the data is both efficiently distributed across a research infrastructure to support easy access and reliable availability, and is also carefully characterised to permit easy retrieval based on a range of parameters. It is also important that experiments conducted on the data can be easily compartmentalised so that individual processing tasks can be parallelised and executed close to the data itself, so as to optimise use of resources and provide swift results to investigators. Given the complexity of data curation, discovery, analysis and processing in general, it falls to a number of optimisation services to configure the underlying research environment to best meet the requirements of investigators both on an individual basis, but also as part of a community engaging with a shared infrastructure.

Fundamentally, research infrastructure should enhance and extend existing research networks, whether they be technical or social, explicit or tacit. Building and maintaining cohesive research networks enables greater collaboration, greater transparency, and faster normalisation of new practices, all of which can aid in producing high quality, high impact research that can be verified and validated by peers. This is one of the fundamental tenets of ENVRIplus, and the creation of a standard reference model, standard vocabulary and standard set of recommendations for service design are intended to guide the creation of services that embed themselves elegantly into these networks.

With regard to the optimisation strand of ENVRIplus, we need to consider the percolation of optimisation requirements throughout different 'levels' of research infrastructure, for example as shown in Figure 2. Within the architecture model promoted by Deliverable 5.4 ("A development plan for common operations and cross-cutting services based on a network of data managers and developers"), most computational research infrastructure is distributed across three main types of environment: *virtual research environments*, domain-specific *research infrastructures* and common *e-infrastructures* for providing computational resources. We revisit these types of environment in more detail in Section 2.3, but suffice for now to say that researcher requirements for optimisation must be passed through all of these kinds of environment and translated such that concrete requirements for the distribution and aggregation of data and processes across resources provided by e-infrastructure can be acquired. It is then necessary to provide the necessary tools and services to actually configure the available e-infrastructure accordingly.

2.1 Community requirements

The requirements of the ENVRIplus community for optimisation are numerous, but are difficult to separate from the requirements for individual services and use-cases. Nevertheless, Deliverable 5.1 "A consistent characterisation of existing and planned RIs" endeavoured to assess the requirements that were obtained and propose directions in which the environmental science community might proceed. The work of Task 7.2 should take cues from this assessment, tempered by the limitations of the task and the ability to deliver results within a short time frame.

Three questions were raised as being imperative to guiding optimisation efforts:

- What do we (the community) want optimised?
- What trade-offs to achieve that optimisation are we willing to accept?
- What measurable *cost functions* can we use to judge our answers to the first two questions?

Optimisation can be evaluated on an individual basis, or on a system-wide basis. While providing more efficient service to a specific user performing a specific data-driven investigation is generally desirable, it should not be at the cost of performance for everybody else using an RI. Approaches that optimise the use of resources across an entire RI, resulting in a modest improvement to overall productivity rather than boosting a single experiment significantly, should be considered. Certain cost functions such as energy consumption (*e.g.* by data centres) do not evidently impact individual user experiences, but are still important to research infrastructure as a whole.

The focus of Task 7.2 is on matters of data delivery and efficient large-scale data processing—to ensure that data can be quickly identified, retrieved and processed in a manner that accelerates the investigations of a wide research community. It is clear that we should not undermine the goals of the other tasks in ENVRIplus, which generally involve improving the accessibility and integration of RI resources and services. We *do* want to make better use of available common e-infrastructure, which does imply a certain trade-off on control of the data, albeit one that can be mitigated by proper attribution and provenance services. With regard to cost functions, it is difficult to measure optimality generically. Instead, different 'optimisation' services should each target a specific cost: execution time, communication time, monetary cost for leased resources, energy consumption, *etc.* In each case, we should be clear as to the benefit being offered to researchers, and allow the community to make their own judgements as to their applicability to their individual investigations.

Deliverable 5.1 also notes the importance of enlisting existing technologies supported by active communities, which can drive the evolution and ensure the robustness of a technological product. One of the strands of activity in the optimisation task is therefore to investigate data processing technologies that might be useful for communities if packaged within a service framework that would expedite configuration and deployment.

Finally, the impact of optimisation, even if performed at the e-infrastructure level, is felt at the user level, *e.g.* by researchers using an RI. As such, the runtime requirements on data processing and other live services have to be distilled from user-level requirements, however they are expressed, and the resultant performance of those services should be viewed in terms of how they impact user activity or productivity. This requires an understanding of the different operational layers of e-infrastructure, from physical resource provisioning up through virtualisation, data processing platforms and individual application processes, and the interaction that exists between such layers.

Use-case: data subscription

Consider the example of a data subscription service, where subscribers can register to receive periodic customised views of some data corpus. A technical use-case for just such a service, based on marine

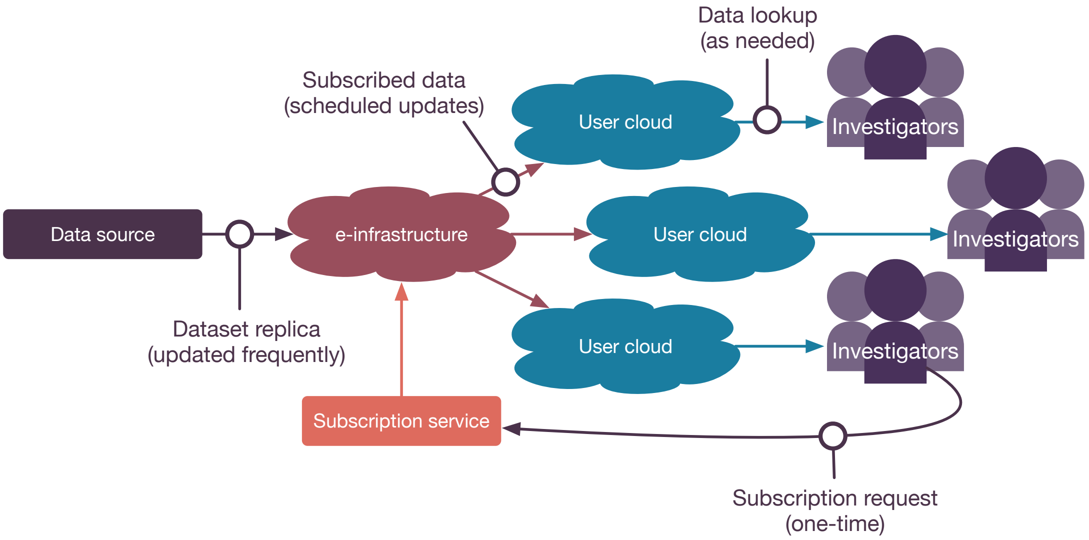

Figure 3: Data subscription: investigators subscribe to tailored updates of time series data provided by an RI.

data provided by the Euro-Argo RI and stored within the EGI FedCloud repository, is currently under investigation within ENVRIplus[5], with involvement of some of the participants in Task 7.2. Data drawn from the *Euro-Argo data service* replicated on a EUDAT-based *replica service*. Investigators subscribe to the data, but can tailor the extent and format of the data like a view on a database, as well as choose the frequency of updates. The tailored updated are then provided on schedule to investigators' cloud spaces, which they can access at their leisure.

Consider how this subscription service might scale for greater volumes of subscribers. Even given a fast engine for data processing such as Apache Spark (used in the Euro-Argo pilot as reported specifically in Section 3.4), too many different subscription requests for different views on the data may lead to a degradation of service quality, either on the part of the data processing system itself, or due to too much network traffic. Moreover, aside from having to provide the desired views on the updated dataset on demand, these views may involve joins of multiple datasets in more advanced cases, further increasing the processing burden service-side. Processing burden is often best alleviated by hosting additional replicas of the main dataset(s) in order to divide the burden of subscription processing (Figure 4, though this requires regular synchronisation of replicas. This can also help with data traffic, but there is another approach that can be taken to address data transport costs specifically; that is to identify most common requests (matching a given view on the data and a given frequency), and cache those requests either locally, or remotely based on geographic proximity (actually network distance) to given clusters of users (Figure 5).

Ultimately, what we want from 'optimisation' in the context of Task 7.2 is the best way to use the underlying e-infrastructure to provide services on behalf of environmental science RIs. While optimality can be considered on many levels (algorithmic, software, platform, *etc.*), we assume that by focusing on this area, we can generate tangible improvements to the quality of service provided by RIs to researchers by improving some of the most fundamental elements of data-driven investigation (*i.e.* the delivery and staging of data for processing) in a way that is complementary to the work being performed within other tasks in the ENVRIplus project. This entails the development of services to solve specific problems regarding the planning and provisioning of infrastructure, and the deployment of processes and tools. It entails the investigation of what tools are already available that can help in various scales of data processing problem, of interest to both RIs internally and researchers outside of the RIs. It also entails figuring out how best to gather information about the research context and the surrounding technical environment in order to make such services as clever as possible. One thing that is desired is the ability to automatically apply such optimisations without requiring manual

[5] https://confluence.egi.eu/display/EC/TC_2+EURO-Argo+Data+Subscription+Service

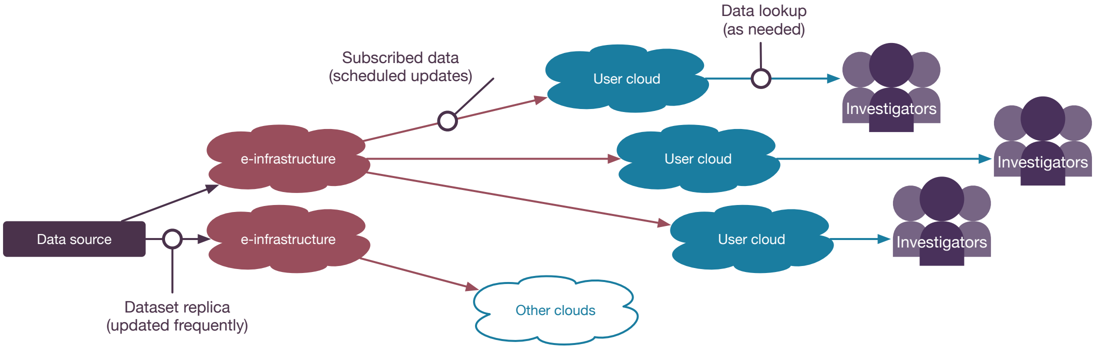

Figure 4: The burden on resources for the data subscription case may be alleviated by replicating the database on multiple e-infrastructures.

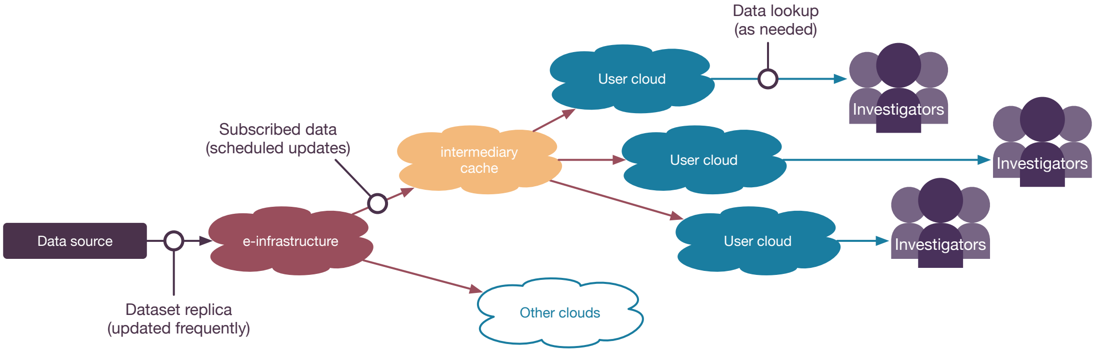

Figure 5: Caching common data requests by region (as defined by network distance) can also reduce data transmission costs close to the public e-infrastructure.

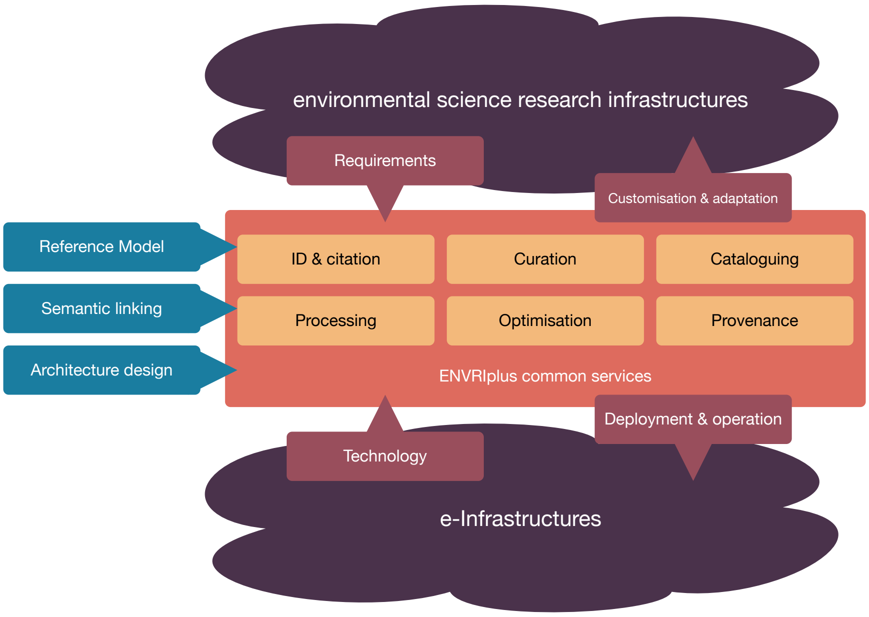

Figure 6: ENVRIplus will create common computational services for research infrastructures based on a standard reference model, semantic linking and architecture design.

configuration in every case. Given a plan for distributing data and processes across e-infrastructure, we want to be able to provision the necessary underlying resources automatically, applying the same technique and technology in multiple instances, and we want to be able to do it with the minimum of human intercession—for example, if the e-infrastructure stretches across multiple locales or domains, the provisioning agent should be able to provision the resources at all sites simultaneously and build the required overlay network (we look at just such a provisioner in Section 3.2).

Over the course of Task 7.2, a methodology for developing the kind of services for optimisation we are interested in will be refined, and a proof of concept will be developed by implementing a number of prototype services in the context of a number of technical use-cases provided via the RIs involved in ENVRIplus.

2.2 Optimisation in the 'Data for Science' theme

The ENVRIplus Data for Science theme addresses six topics of concern to environmental science RIs, interpreted via three 'cross-cutting' activities (see Figure 6). The six topics identified are *identification and citation*, *curation*, *cataloguing*, *processing*, *optimisation* and *provenance*. The three cross-cutting activities are *reference model design*, *semantic linking* and *architecture design*. Task 7.2 is the optimisation task, intended to provide technologies and recommendations to improve the use of e-infrastructure by RI communities to carry out data-driven investigations. It is closely linked to Task 7.1, for processing, which is concerned with providing an integrated common platform for data processing for the ENVRI community. The key difference between the two tasks is that whereas Task 7.1 provides a pre-packaged virtual research environment for the ENVRIplus RIs to use as they deem fit, Task 7.2 is more of an exploration of the use of microservices for e-infrastructure customisation on virtualised infrastructure platforms in general.

The cross-cutting activities are intended to inform design and guide interoperation. The ENVRI Reference Model provides a generic schema for RI architecture, and serves to establish a standard lexicon for describing RI components and activities, as well as the research data lifecycle. This lexicon is formalised as a controlled vocabulary by the Reference Model ontology, that serves as the core of the semantic linking framework, which tries to provide a means to link the different concept models used to describe data, models, architecture and infrastructure. The model architecture is intended to provide guidance as to how to actually implement interoperable common services in a fashion that will maximise the impact of those services and reduce the complexity of future RI development. With respect to the optimisation task, any development should be framed in terms of the reference model and carried out in compliance with the overall model architecture. Meanwhile, the semantic linking provides a basis for encoding information that might be used to guide optimisation services, *e.g.* to base resource selection and scheduling on application and infrastructure constraints.

2.2.1 Relationships with other topics

As illustrated in Figure 6, it is the environmental science RIs that provide the requirements which the technologies and recommendations produced by ENVRIplus are supposed to address, and it is the underlying e-infrastructure (whether provided by the RIs directly or via shared platforms) that determine the available existing technologies to build upon. The common services produced by ENVRIplus need to be customisable to meet the needs of specific RIs even as they adhere to a standard model, and they need to be deployable on the available e-infrastructure. Optimisation services should work principally on the e-infrastructure level, should work with data processing technologies of interest to the research community now, and should be guided by the requirements of *specific* research investigations, providing customisation based on those requirements at *deployment* time, or even during runtime. In that sense, the optimisation 'common service' is something more dynamic perhaps than the services produced by the other five topics. Nonetheless, we can identify where optimisation services might be of use in other topics or *vice versa*:

Identification and citation It is necessary to ensure availability of identification services, and it is useful to be able to direct users to the best replicas of a given dataset that would ensure the most effective use of the underlying network (*e.g.* by minimising network distance and maximising bandwidth). In general, the existence of persistent identifiers, whether global or localised to a particular e-infrastructure, is vital to the identification and selection of resources and research assets in any complex application. Thus the identification and citation service produced by ENVRIplus should be directly utilised by the optimisation services to locate assets and plan based on the associated metadata.

Curation Streamlining the acquisition of data from data providers is important to many RIs, both to maximise the range and timeliness of datasets then made available to researchers, and to increase data security (by ensuring that it is properly curated with minimal delay, reducing the risk of data corruption or loss). In general, the principal concerns of curation are ensuring the accessibility and availability of research assets (especially, but not exclusively, data). High availability in particular requires effective replication procedures across multiple sites. It would be expedient to minimise the cost of synchronising replicas and to anticipate where user demand (for retrieval) is likely to be so as to minimise network congestion.

Cataloguing Data catalogues are expected to be the main vector by which data is identified and requested by users, regardless of where that data is ultimately taken for processing and analysis. As such, the optimisation of both querying and data retrieval is of concern. Moreover, optimisation services should use catalogues as a basis for reasoning about the operating environment—*e.g.* what data is available and what facilities exist to process that data. Processing should be planned around minimising data movement and ensuring that the facilities used are sufficiently powerful to provide results in a timely fashion.

Processing Processing tools and services provide one of the key types of building block of an investigation workflow (the others being datasets and execution resources). Persistent services are generally integrated into pre-configured e-infrastructure; any authorised investigator can make use of them, sending any input data and retrieving the results. Otherwise, processing elements need to be retrieved from some repository and hosted on e-infrastructure provided by another source—for example deployed onto virtual resources provisioned on some Cloud. Optimisation with respect to processing is therefore most concerned with identifying the available processing elements or services, and either:

- Determining how best to stage in and out data in relation to existing process deployments (especially in the case of online services).

- Determining how best to deploy processes on available e-infrastructure based on processing requirements and existing data placement (especially for setting up customised processing pipelines for large datasets curated by e-RIs).

There two approaches are often both complementary and mutually dependent. Given data, an investigator can process them on their own compute resources (ranging from a laptop or desktop to a private compute cluster), transfer the data onto a dedicated resource (such as a supercomputer for which they have leased time and capacity, cloud infrastructure provisioned for the purpose, or an online web service), or direct processing of the data on-site (generally only possible where the investigator has authority over the site in question, and generally limited to standard analyses that are part of the aforementioned data pipeline). Each of these options incurs some kind of cost for data movement, data preparation, and process configuration. Any of these costs could be subject to optimisation; the provisioning of tools tailored to different kinds of (common) data processing, coupled with different cost functions (based on the priorities of the researcher or infrastructure) should be one of the outputs of the optimisation task.

Provenance Good provenance is fundamental to optimisation—in order to be able to anticipate how data will be used by the community, and what infrastructure elements should be able conscripted to provide access to and processing capability over those data, it is necessary to understand as much about the data (and the processes and models that work over the data) as possible. Provenance data is a key element of knowledge-augmented infrastructure, and provenance recording services are a major source and vector of the knowledge that needs to be disseminated throughout the infrastructure in order to realise this ideal. Ensuring that the various questions about data origins and processes can be asked and answered becomes more challenging the greater the heterogeneity of the data being handled by the RI, and so potential for runtime optimisation in particular will depend on the solutions provided by the provenance task (Task 8.3) witin ENVRIplus. As far as optimisation serving provenance in and of itself is concerned, it may also be possible to focus on the management of provenance data streams during data processing itself, being amenable to the real-time stream processing approaches being investigated in Section 3.

Actually realising these links with these topics in the context of the parallel tasks in ENVRIplus requires some degree of collaboration between task teams, possibly best implemented via instances of shared participation by partners in the project. The existence of specific use-cases[6] identified within the project to drive collaboration and focus efforts should also assist in this effort. Critically, the cross-cutting tasks identified earlier (conducted under the auspices of Tasks 5.2–4) should also provide a normalising factor, ensuring that the design of all topic services are 'reference model guided' as stated in the project's description of work.

2.2.2 Guidance from Reference Model

Data processing is considered to be one of the major phases of the research data lifecycle, according to the ENVRI Reference Model. Consequently, it identifies the data processing subsystem as a core part in (most) RIs, with its own dedicated community of agents and its own set of core behaviours. These behaviours can be decomposed into information actions (activities that transform or generate new data) which must be supported by computational services. Figure 7 illustrates some of the concepts associated with a generic *process data* action by the Reference Model, and in particular shows how significant the role of computation can be for something that is, in abstract, a relatively simple part of RI activity. The Reference Model identifies a number of computational objects and operations required simply to ensure access to research data, quite apart from the actual generation and storage of results. From Figure 7, it can be seen that:

- A processing environment has to be prepared before data can even be staged.
- Any results derived from the process should themselves be recorded, either on the infrastructure side or the client side.

[6] http://wiki.envri.eu, under 'use cases'

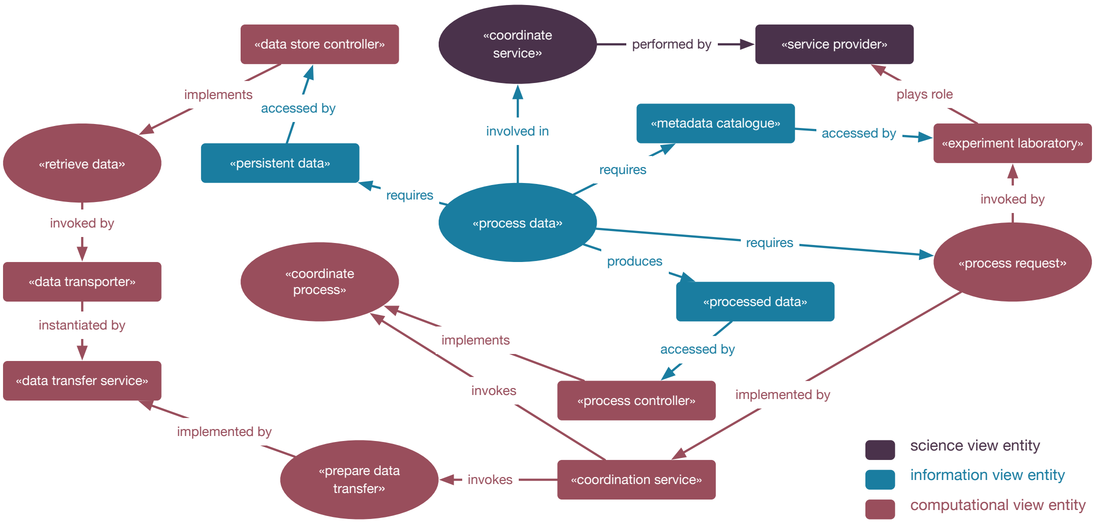

Figure 7: The ENVRI reference model identifies a number of requirements surrounding the act of processing data.

- A dedicated service is needed to handle coordination of processes, in order to ensure tasks are properly scheduled and processes on shared infrastructure do not interfere with one another.

Also evident in Figure 7 are the three main viewpoints used by the Reference Model to describe concepts. Each viewpoint has a different focus:

- The **science view** is concerned with identifying the main (human, organisational and artificial) actors in the RI ecosystem, the roles they assume and the behaviours they engage in. With regard to optimisation, the science view informs us of the data processing behaviours that an RI should be able to support, and who is responsible for carrying those behaviours out; hence, the roles played by optimisation services.

- The **information view** is concerned with the data objects associated with RIs and their operations, as well as their generation and transformation under various conditions. With regard to optimisation, the information view informs us of the data objects available within the RI at various stages in the research data lifecycle; hence, where optimisation services can extract and implant useful information.

- The **computational view** is concerned with the logical distribution of processes between computational elements, identifying necessary service objects and their interfaces. With regard to optimisation, the computational view informs us of the computational services involved in various operations; hence, which services are either subsumed by optimisation services, or augmented by them.

Examining the archetypes defined in the science view, optimisation of the sort we are concerned with here is primarily carried out by non-human actors, principally as part of the *data processing subsystem* of an RI. Most roles of import are members of the *data processing community*:

- *Data provider* and *Service consumer* represent data sources and either researchers or downstream workflow processes respectively. Optimal placement of data and services depends on the relative (network) locations of the providers and consumers.

- *Services* represent the processing blocks of a workflow, and either represent optimisation services themselves, or are the objects of optimisation activity in and of themselves.

- *Service registry* is a role fulfilled by the kinds of service catalogue needed for discovery of services and promoted by the model architecture work of Task 5.4.

- *Processing environment planner* represents the collective role of the kind of optimisation services that Task 7.2 seeks to produce; microservices for guiding the customisation of the environment (specifically the e-infrastructure on which processes are executed and data is staged).

In the information view, any instance of *persistent data* may be used in data processing, but optimisation services themselves will generally be guided by:

- *Specifications of investigation design* that define the characteristics of interactions with research infrastructure. The current Reference Model emphasises investigation design for data acquisition, but such formalisation is also relevant for any scientific investigation requiring data processing.
- *Conceptual models* that define the vocabulary for specifications of datasets, services, workflows, *etc.* Formal models are necessary for encoding the information needed by knowledge-driven services.
- *Metadata*, *metadata catalogues* and *data provenance* that provide contextual data, vital for optimisation.
- *Service descriptions* that identify the functionality of services, whether for optimisation or for use in an optimised application workflow.

Optimisation services may be applicable to any number of information actions, including to *store data*, *publish data*, *query data* and of course *process data*. Optimisation services rely on certain other actions such as *perform mapping* (to interpret between abstract investigation requirements and low-level infrastructure requirements) and *track provenance* (to acquire contextual information about the operating environment to guide services).

In the computational view, optimisation is triggered by the invocation of processing via some *virtual laboratory* (*i.e.* a virtual research environment or other scientific gateway), requiring the use of:

- A *coordination service* to manage a process or set of processes. A coordination service can invoke any number of optimisation microservices to help customise the underlying infrastructure and aid process orchestration.
- *Process controllers* to run individual processes or processing elements. The configuration of process controllers (and their underlying infrastructure) is something we want optimisation services to address.
- A *data transfer service* to coordinate the staging and recording of data via any number of *data store controllers*. Data movement remains a critical concern for data processing in general, in many cases posing a more significant obstacle than the requirements of the actual processing. Any optimisation must be carried out based on the limitations and costs of data transmission and staging.

Optimisation microservices themselves will likely collectively make up part of an RI's coordination service(s) and process controllers (the services themselves operating slightly below the level of detail defined in the current version of the Reference Model).

As with all the development tasks in ENVRIplus, while the reference model provides some guidance for the development and some standard conceptual vocabulary, there is still ample opportunity to deliver feedback to the reference model team (in Task 5.2) in order to expand and refine the model. In the case of Task 7.2, more detail can be imprinted into the computational view in particular to identify the specific interactions between computational objects that optimisation can (even should) target. There is also potential to influence the at present undeveloped engineering view, which pertains to the hosting of computational services across an RI; from the reference model perspective, the optimisation services described in Section 2.3 essentially focus on matching computational objects for dynamic applications to the best engineering objects.

2.2.3 Semantic modelling requirements

In order to model and characterise research applications and RI capabilities, it is intended to enlist the semantic linking framework of Task 5.3 for decision making and support. The semantic linking

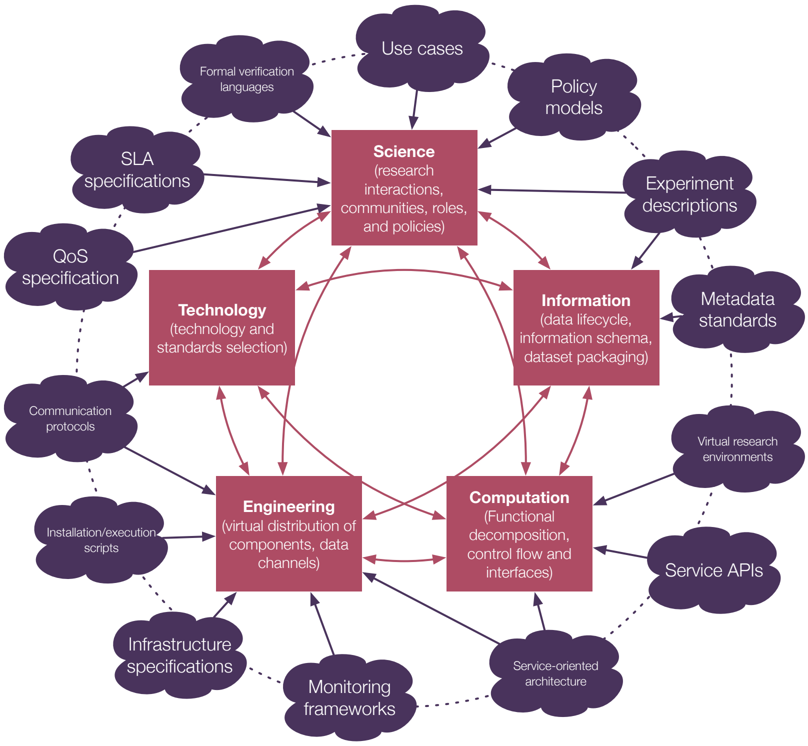
Use cases
Formal verification languages
Policy models
SLA specifications
Science (research interactions, communities, roles, and policies)
Experiment descriptions
QoS specification
Technology (technology and standards selection)
Information (data lifecycle, information schema, dataset packaging)
Metadata standards
Communication protocols
Virtual research environments
Engineering (virtual distribution of components, data channels)
Computation (Functional decomposition, control flow and interfaces)
Installation/execution scripts
Service APIs
Infrastructure specifications
Monitoring frameworks
Service-oriented architecture

Figure 8: The idea of the semantic linking activity is to use the ENVRI Reference Model as a hub for linking a wide range of standards, many of which describe entities required for smart optimisation services.

framework for ENVRIplus is part of *Open Information Linking for Environmental research infrastructures* (OIL-E)[7]. The semantic linking framework formally encodes the ENVRI reference model (RM) as an ontology, and is intended to provide a means to relate various research and technology standards applicable to environmental science research together in terms of that model, as well as provide guidance for mapping between standards. At a very basic level, it should be possible to describe many research activities in terms of behaviours and processes defined by the reference model, and it should be possible to make statements about the computational services needed by such activities and underlying (virtual) resources hosting those services. More generally however, OIL-E is supposed to develop links to a range of different controlled vocabularies addressing a number of facets of environmental science research, computation and data analysis, as illustrated in Figure 8. In the context of Task 7.2, we are interested in the controlled vocabularies necessary to describe data discovery, delivery and processing from the investigation level down to the infrastructure level. Formal descriptions of such form the essential basis for automated reasoning, and thus the (semi-)automated optimisation of research infrastructure in response to specific data-driven investigations.

One of the goals of the semantic linking work in ENVRIplus is to help realise a model of *application-infrastructure co-programming and control* for running data-driven investigations using research infrastructure. This model should bring together investigation design, e-infrastructure customisation, and runtime control of processes into one optimisation loop based on investigator requirements and the state of the research environment. In this model:

1. The application logic (*i.e.* the composition of processes needed to carry out a data-driven investigation) should be programmed with consideration for performance requirements together with the programmability and controllability of the underlying e-infrastructure environment. In that way, it should be possible to refine both the investigation and the virtual runtime environment for executing investigation processes together during the design phase of the investigation lifecycle.

[7] http://oil-e.net/

2. The virtual runtime environment can be customised to address application requirements, and can then be provisioned on an e-infrastructure environment with SLAs oriented towards specific investigation requirements.
3. It should be possible to monitor the state of the application at runtime.
4. The application should be able to autonomously adapt its own behaviour and that of the virtual runtime environment when performance drops during runtime, for example by scaling up or out virtual resources in response to workload.

Semantic linking helps to realise this model by providing a framework for describing investigations, data, tools, facilities, infrastructure and other elements of the research environment, and a methodology for efficiently translating requirements between such descriptions. It is not expected that mappings to and from every commonly used standard will be created within the context of Task 5.3, but it is expected that a certain proof-of-concept will be created and recommendations made. In particular, it is hoped that from Task 5.3, a means to translate quality of service requirements on research processes into constraints on e-infrastructure selection and customisation will be delivered that can be used as a means to generate contextual information required to guide optimisation microservices (and indeed data processing services in general).

At the core of the semantic linking model is the ENVRI Reference Model ontology, which can be used to generally classify all the main components of an RI and its research, data and computational elements. Also needed is a means to describe a research application workflow, its requirements, and a customised infrastructure. If application requirements can be translated into infrastructure constraints, then this information can be stored alongside application information in a suitable knowledge base and then directly used by microservices to optimise the use of e-infrastructure, *e.g.* to better select virtual resources, to better select their location, or to better configure the deployment of processes. We can break down some of the key knowledge products as follows:

- An **investigation design** is a specification of intent on the part of an investigator detailing the questions they are attempting to answer via their interaction with research infrastructure. At this time such investigation design is typically *not* formalised in any 'machine-readable' sense, or even provided to the research environment, the researcher instead immediately proceeding with defining the necessary task workflow or just interacting with the required services directly. Nevertheless, there are clear advantages to being able to specify (parts of) an investigation design within a virtual research environment—for example to allow the VRE to autonomously identify the resources necessary to answer given research questions without the investigator needing to be aware of exactly what is available, to guide discovery services, to identify requirements on services, or to partially automate the construction of an application workflow.
- An **application workflow** describes the configuration of services and resources that will be used to realise a particular data-driven investigation. The notion of workflow captures the dependencies between different parts of a research investigation and can be used to manage the transportation of data between services and the customisation of infrastructure to host data processing. In particular, the workflow is key to any automation of interaction between different research support environments; otherwise the investigator themselves must broker all such interactions.
- An **infrastructure specification** describes a specific customisation of e-infrastructure to serve a particular research investigation. In practice, this encompasses the conscription of virtual machines, HPC resources and online services to manage the workload of the investigation, and may be defined on an immediate or longer term basis. In advanced cases, the configuration of the underlying network may be specified (*e.g.* realised using software-defined networking), especially for larger scale investigations.
- A **control profile** describes the specific actions that can be taken against an active application workflow during its execution and their accompanying interfaces. In the basic case, this amounts to the ability to start and stop specific processes, but in more advanced cases there may be more elaborate options for computational steering, including the migration of processes and virtual machines, the calibration of runtime monitoring, and the modification of experiment parameters.

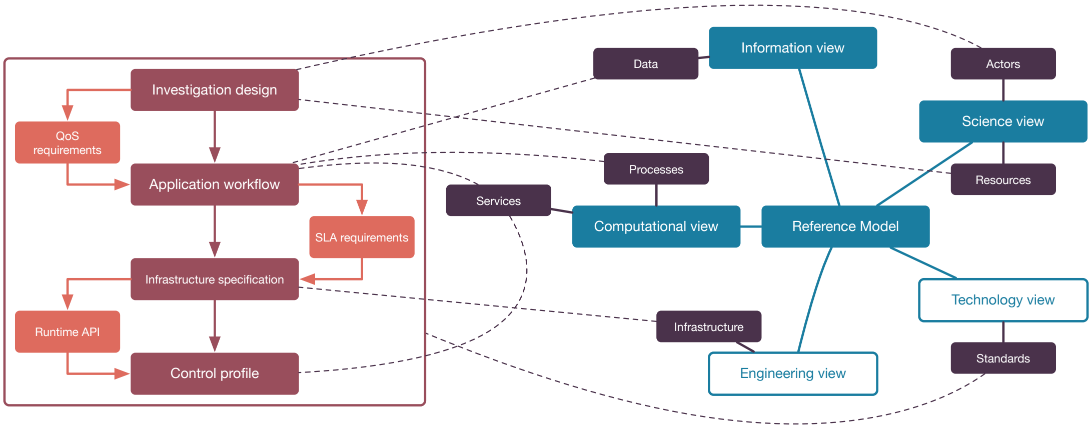

Figure 9: Vertical optimisation requires the ability to define investigations and percolate requirements down to infrastructure level—requiring semantic linking of various standards guided by the ENVRI Reference Model.

As illustrated in Figure 9, the Reference Model describes a number of the key concepts needed to describe these entities, but specific standards need to be linked to these concepts in order to actually describe them properly. Standards such as WS-BPEL[8] for describing (Web service) workflows, TOSCA[9] for describing applications on Cloud, INDL [10] for describing virtual infrastructure and OCCI[10] for describing APIs for interacting with such infrastructure are possible candidates. It is difficult however to identify existing standards that perfectly meet our needs, and a balance must be struck between re-using existing work, and adapting it to purpose (which carries a commensurate cost in development effort).

2.2.4 Real-time concerns

While speed is often a desirable quality of application execution, several research applications are subject to stricter or more subtle real-time constraints either directly attributed to the application itself (*e.g.* for event detection and forecasting) or indirectly attributed via the requirements of the supporting infrastructure (*e.g.* execution steering and failure recovery). *Timeliness* is a crucial factor in several application scenarios. For example, disaster warning systems rely on rapid event detection based on continuously updated forecast models. Data acquisition from sensor networks relies on real-time processing and quality checking of new data in order to provide frequent updates to investigators on schedule. The steering of applications at runtime requires agile infrastructure capable of responding to adaptation requests and performance variations at any time. Enabling time-critical applications on elastic infrastructure is still a challenge in many research environments however. The optimal assignment of (virtual) resources to individual tasks and processes, and the efficient scheduling of activity, pose a direct tension with the need to build in additional tolerances for performance fluctuations that otherwise risk violation of time-critical constraints.

A *time-critical application* is simply an application that requires the underlying infrastructure hosting the application to operate as a real-time system in order to fulfil some real-time constraints. Superficially most applications are real-time, but we are concerned with applications that specifically impose time constraints on their own execution, either due to deadlines on their output, or the cumulative effects of delays upon a processing pipeline that handles real-time data.

Real-time requirements on research infrastructure differ—*e.g.* parallelising computing tasks to minimise execution time versus optimising network communication to limit data latency versus simulating physical systems based on 'wall clock time'. Figure 10 identifies some of the main types of time-critical application. Of particular notice is the distinction between *real-time* and *speed-critical* applications,

[8] https://www.oasis-open.org/committees/wsbpel/
[9] https://www.oasis-open.org/committees/tosca/
[10] http://occi-wg.org/

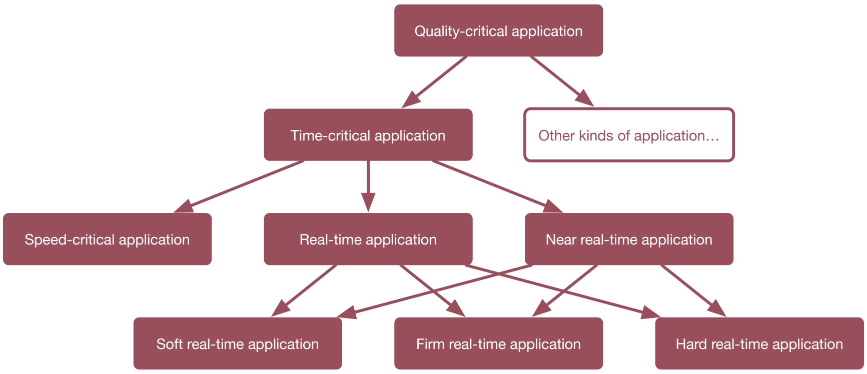
Quality-critical application
Time-critical application
Other kinds of application...
Speed-critical application
Real-time application
Near real-time application
Soft real-time application
Firm real-time application
Hard real-time application

Figure 10: Time-critical applications can be categorised based on the nature of their real-time requirements.

whereby the latter is concerned primarily with simply completing computation as fast as possible, while the former is concerned with responsiveness to events. Similarly, there is a distinction often upheld between pure real-time applications and *near real-time* applications; the main distinction being that near real-time applications have a minimum latency that should generally be upheld in order to properly 'pace' a larger processing pipeline (*i.e.* not overload interstitial buffers by completing tasks too fast). There is also a distinction between *hard*, *firm* and *soft* constraints on both real-time and near real-time applications [14]. Essentially, failure to meet a hard deadline constraint results in failure of the entire application, while failure to meet a soft constraint merely results in a degradation of the user experience. A firm deadline constraint represents a softer constraint that is acceptable to fail to meet occasionally, but will ultimately result in a failed system if failed too many times. In practice, most data processing pipelines have firm real-time constraints; the failure to meet a deadline is not instantly disastrous, but will place pressure on the entire real-time system due to the need to 'catch up' to prevent a cascading delay.

In general however, researchers do not explicitly frame their requirements of RI services in the manner just described. Instead, they are principally concerned with their own experience interacting with the RI from the outside. Similarly, even RI engineers are less concerned with the specific characteristics of particular applications, and more with the cumulative effect on the running of the infrastructure as a whole and the efficient use of (possibly quite expensive) underlying resources from data storage, transmission and processing. Therefore it is important to explore ways to model the abstract requirements on research activities in such a way as they can be translated into more specific real-time requirements on resources and processes as just described above. Hence why the semantic modelling of infrastructure by Task 5.3 is important to the optimisation task, so gaining attention in the Task 7.2 roadmap (in Section 4).

2.3 Vertical optimisation of research support environments

'Vertical optimisation' is concerned with how information flows between different technical layers so that each layer can be configured with regard to information provided by the other layers, rather than treating each layer as an isolated system. The ENVRI reference model identifies the 'customisation of processing environments' as a behaviour of an RI system, and this customisation requires knowledge of the kinds of investigation being carried out using RI resources, the location and state of different resources, and their relationship with the underlying e-infrastructure.

One of the most pressing concerns for large-scale data processing (and thus the customisation of processing environments) is the relative placement of data and processes. This concern is often articulated in the form of a dichotomy: *do we move the data to the process, or the process to the data?* This is a simplification of course; in a multi-stage research investigation it is necessary to configure the staging of interim datasets and processes to optimise the workflow as a whole, and it is also important to consider the set of all concurrent investigations in order to better optimise overall infrastructure performance.

Moving data to processes has advantages and disadvantages. For example:

- There is no need to provide and maintain additional processing capacity at the data source; the data provider can concentrate on ensuring availability, supporting discovery, and performing any necessary curation activities behind the scenes.
- Specialised facilities for high performance or high throughout computing can be conscripted for intensive processes, rather than limiting computation based on facilities local to the data source.
- For large datasets, the movement of data from source to computational platform can prove prohibitive; often network bandwidth becomes the primary bottleneck for research, rather than computing capacity or the degree of innovation exhibited by the research investigation at hand.

Conversely, moving processes to data offers its own advantages and disadvantages:

- Data transfer is minimised, allowing processes to be quickly and efficiently executed.
- Processing is limited within the capabilities offered by the local computing platform provided alongside the data.
- Processes must be limited or otherwise sandboxed to prevent destructive behaviours or simply to ensure compatibility with the technologies deployed at the source. The execution of arbitrary code for processing data is almost never acceptable to the data source.

Virtualisation provides a layer of insulation for processing that permits arbitrary deployments of tools and code to be executed on a common computing platform. The provision of cloud services for deploying virtual machines on which to run processes in principal allows for code to be brought closer to the data, and addresses some of the disadvantages of allowing investigators to 'bring their own code' and run it on a public facility. This does not remove the need to stage data however; it is unlikely that dedicated data providers in the environmental science domain will start acting as cloud providers in general. It is however likely that intermediary e-infrastructures with links to major research network backbones will provide such a service, and that the cost of moving data to those intermediaries will provide more efficient than moving entire datasets to an investigator's local machine. It should also be noted that increasingly such service providers will host replicas of major scientific datasets of interest, meaning that the data staging will already have been handled on behalf of a community of researchers in advance.

It is especially important that optimisation of e-infrastructure, both the provision of resources and the staging of data, be conducted from a *community* perspective rather than at the behest of individual researchers. It is possible, and feasible, that certain core datasets provided by RIs can be identified and staged on e-infrastructure services in anticipation of their use by a community, thus reducing the data delivery costs for a entire class of research investigation, and reducing congestion at the point of origin at the RIs' core facilities. The key facilities can then dedicate more of their resources to identification, curation, cataloguing and the handling of more specialised requests, and simply push the 'bulk common data' to the generic data centres and e-infrastructure providers exemplified by EUDAT and EGI.

Optimisation services can ensure that the e-infrastructure topologies used for specific investigations or applications best match the characteristics of the data and processes needed; this is especially doable for virtualised infrastructure, where virtual machines can be set up anywhere within a Cloud or Cloud federation. Ultimately though, it is necessary to improve the richness of metadata and other contextual information present within research environments to provide the inputs needed by such services to operate in more complex circumstances.

2.3.1 Research environments and the investigation lifecycle

We can identify three main classes of computational research environment that have drawn the attention of research communities in recent years: *research infrastructures* (RIs or e-RIs), *virtual research environments* (VREs) and *e-infrastructures*. The boundaries between these classes are not entirely distinct, but do nevertheless represent differing foci (and to a certain extent, an incremental evolution of the understanding of the needs of researchers): provision of research assets, development of better 'data-scopes' for navigating those assets, and providing the necessary resources to make full use of them. In more detail:

Research infrastructure Computational research infrastructures (e-RIs) are conglomerations of research infrastructure that are specifically concerned with curating and ensuring the availability of research data or other research assets that are of value to a research community. Basically, they provide the key elements of the workflows required to conduct a particular family of research investigation. In many cases, 'e-RI' and 'RI' (representing a research infrastructure as a discrete initiative or organisation) can be used (and is used) interchangeably. An RI in the general sense encompasses more than merely information technology related aspects however, whereas the use of e-RI specifically focuses on the computational, data and communication infrastructure within an RI. The use of research data infrastructure within the research environment can be brokered through virtual research environments, which act as aggregators and gatekeepers for access to research assets. e-RIs usually provide their own interfaces for interacting with their assets as well however, often integrated into a single scientific gateway, which itself can constitute a VRE. An e-RI itself is built upon technical services provided internally within an RI initiative, or enlisted from one or more generic e-infrastructure platforms.

Virtual research environments VREs act as gatekeepers for scientific exploration, providing the means to navigate research infrastructure. They do this by providing a unified 'data-scope' for researchers in the form of a portal or workbench for discovery and access of research assets (data, models, services and resources). They may also provide the means to translate an explicit specification of an investigator's goals and methodology into a concrete experimental workflow consisting of a series of actions to be taken manually or automatically. VREs typically provide graphical environments with which users can interact, access to workflow management systems, and access to data analytics.

e-Infrastructures e-Infrastructures focus on management of the service lifecycle of computing, storage and network resources provided for the hosting of RI assets and specific processes requested by investigators. A research infrastructure may provide its own internal e-infrastructure, but in practice typically relies on resources provided by research institutes and dedicated data centres. e-Infrastructures can also be presented as generic platforms for computational research, made available to the broader research community more directly. e-Infrastructures may provide services directly to RIs, for example to preserve a portion of their accumulated data or catalogues, or to provide compute for specific data processing tasks, which may or may not be brokered by RIs on behalf of investigators. e-Infrastructure resources are often provided subject to some kind of service-level agreement (SLA), often informally, though more formal agreements are likely to be needed as the computational research landscape develops further.

The association between these environments is illustrated in Figure 11. The use of e-RIs can be brokered through VREs, which acts as aggregators and gatekeepers for access to research assets. The research infrastructure itself is built upon technical services provided by underlying e-infrastructure platforms, which may be as simple as a few servers installed in a basement, or as complex as a distributed network of virtualised resources collectively provided by a federation of specialist data centres and public institutions.

The ENVRI Reference Model defines the *research data lifecycle*: five phases of data *acquisition*, *curation*, *publishing*, *processing* and *use*, representing how scientific data is handled and exploited at various times, ultimately leading to the gathering of fresh data to support new investigations. We can likewise define another lifecycle for *computational investigation* which should be supported by RIs: three phases of investigation *design*, *deployment* and *adaptation*, representing the construction and execution of such investigations on computational research infrastructure (see Figure 12).

- **Design** concerns the specification of an investigation design, typically as some kind of workflow of tasks or processes, that describes what an investigator is trying to do and how they intend to do it. While to a certain extent design remains in the mind of the investigator, VREs can potentially provide a context for formalising the stages of investigation and identifying their requirements in terms of resources and performance. e-RIs serve to provide the research assets

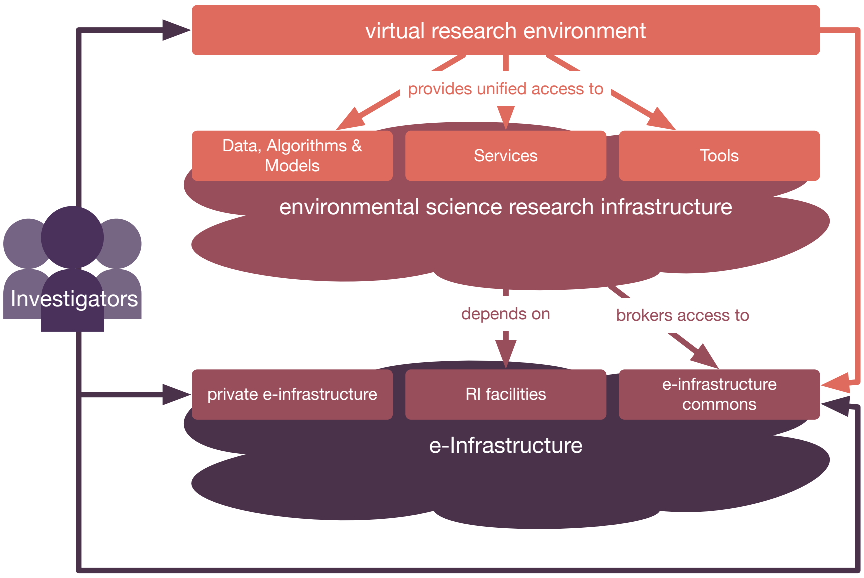

Figure 11: Researchers are supported by three main classes of research support environment: virtual research environments (VREs), computational research infrastructure (e-RIs) and e-infrastructures.

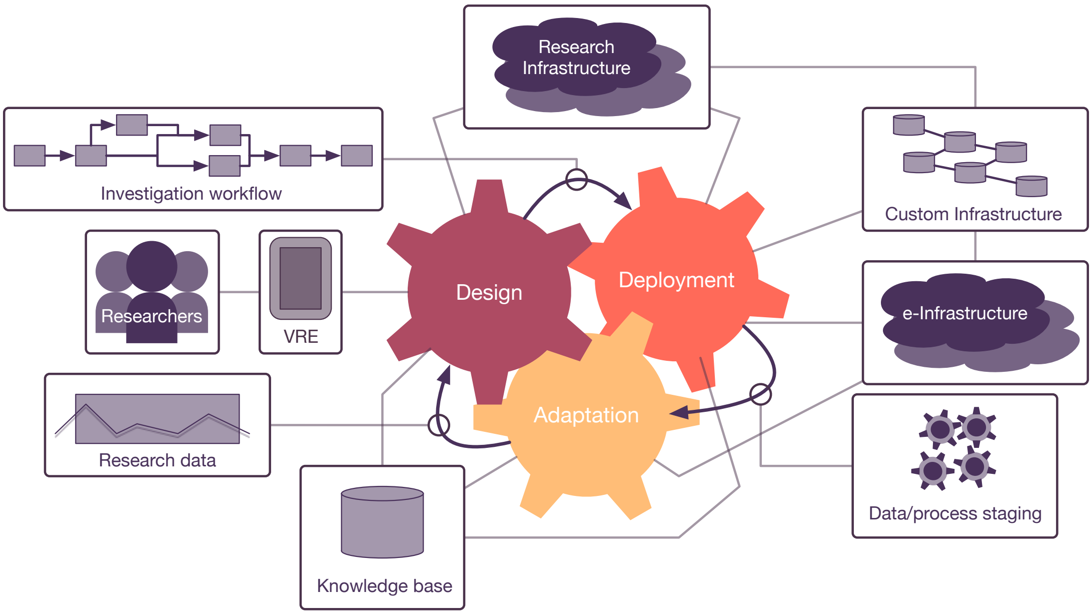

Figure 12: Computational investigation goes through a cycle of design, deployment and adaptation, involving a number of different actors and data entities throughout.

(data, models, tools, *etc.*) necessary to actually realise the investigation.

- **Deployment** concerns the provisioning of suitable infrastructure needed to carry out the stages of a computational investigation, the invocation of services and the installation of processes on infrastructure resources. Deployment depends on e-infrastructure, whether private, RI-provided or provided by an e-infrastructure 'commons'. e-RI services invoked by investigators directly are essentially backed by pre-deployed e-infrastructure, but are otherwise typically the sources of data that must be staged on other e-infrastructure in order to carry out more complex operations.
- **Adaptation** concerns the steering of an on-going investigation, both in terms of the investigator adapting their investigation to fresh results, and in terms of the adaptation of the infrastructure in response to environmental factors such as resource load and faults. The former kind of adaptation requires robust routes via which investigators can interact with the e-infrastructure hosting their applications, whether those be direct (*e.g.* by logging into virtual machines to make adjustments) or via some VRE. The latter kind of adaptation can be more easily contained within the e-infrastructure hosting an application, but should be logged for traceability.

Ultimately, the experiences of the investigator during execution of their investigation informs their future investigations, closing the cycle.

The focus of the optimisation services proposed by this deliverable is on deployment, and to a lesser extent adaptation (particularly the autonomous adaptation of infrastructure), but to achieve 'optimality' any services must ultimately be informed by design (specifically the formal performance requirements of researchers).

Specifically, we need to be able to capture and pass downstream the requirements of investigations; it is necessary to be able to be able to formally describe the investigation workflow, including components, dependencies and data-flows, and be able to express constraints on different compositions of elements in the workflow. These constraints need to be translated into terms comprehensible to optimisation services and allowed to trickle down to the optimisation services—and so it is necessary to look into the overall architecture supporting research investigation.

2.3.2 Optimisation architecture

Optimisation microservices of the kind described here can only be effectively exploited within the context of a well-defined supporting architecture. This supporting architecture must account for the variability of research infrastructure and be as unintrusive as possible to realise in practice. It should also be compliant with the architectural recommendations of Task 5.4 and the rest of the Data for Science theme.

As articulated in D5.4, there are a number of approaches that can be taken to modularise the components of research infrastructure architecture. The use of standard APIs and interchange formats for data and control directives makes it easier to set up independent modules for infrastructure at all levels of abstraction and publish their interfaces, however it is not a given that there will be a convergence of standards in the environment science domain. Nonetheless, there are various junctures where some degree of standardisation seems reachable.

The development and deployment of catalogues allows the range of data, tools, services and other research assets to be explored and indexed in various contexts. Standardised metadata and programmatic interfaces allow these catalogues to be drawn out of their immediate environments (*e.g.* a specific RI), and published more broadly, allowing for cross-RI service discovery and search. From the perspective of processing on e-infrastructure, standard interfaces also allow processing services and modules to be implemented in a more infrastructure-agnostic way, rendering it reasonable to host repositories of such tools that can be used in the context of multi-infrastructures.

The general 'raising' of tools and services out of specific e-infrastructures or e-RIs (*i.e.* providing generic tools and services based on standard APIs that exist outside any specific research environment) makes it easier to run cross-RI investigations, because those tools and services can then be invoked on a range of resources from more than one context, something that is impossible if they are firmly embedded into a specific environment. On the other hand, it must be recognised that cross-RI investigations are not yet a standard demand by research communities, and many communities are still struggling with the more fundamental problem of providing optimal access to the assets within a single RI. Thus we must be cautious about trying to be 'too general' and relying too heavily on unproven (or even non-existent) standards and technologies. Accounting for these concerns, the supporting architecture for optimisation services should 'fall inwards'; for each kind of catalogue, registry or repository made available via a VRE, we look for a service that exists outwith a specific e-RI or e-infrastructure, and in the absence of such a service, we look within the e-RI or e-infrastructure for an alternative service, sacrificing access to generic cross-RI resources in favour of ones specialised for a given context.

The approach taken with regard to optimisation is to provide small targeted 'microservices' that can perform specific operations over virtualised e-infrastructure to assist in the deployment and execution of processes. Such operations might include the planning of customised infrastructure for an application workflow, the provisioning of networked infrastructure across multiple sites, or the deployment of a Map-Reduce topology on a set of virtual machines. Four broad classes of microservice of optimising the deployment of applications on e-infrastructure have been identified:

- **Planners** infer what customised infrastructure best matches an investigation's requirements based on the research assets needed and current environmental factors (*e.g.* resource availability). A planner will generally plan using the information found in resource catalogues describing the characteristics (processing, memory, network, *etc.*) of various resources that can be used to host specific processes identified in the investigation workflow. Selection may be based on process requirements, deadline constraints, or other non-functional requirements such as resilience or security, depending on the characteristics of the investigation. A prototype planning service developed for a particular kind of time-critical application is described in Section 3.1.

- **Provisioners** actually use the infrastructure plans generated by planners in order to set up custom infrastructures. They may also be responsible for checking and negotiating SLAs for use of the provisioned infrastructure, should programmatic means of doing so exist. While planners may take responsibility for the interpretation of investigation requirements and the selection of resources, provisioners have to actually ensure that the resultant infrastructure is fully accessible and internally connected, which may pose a particular challenge for investigations that cross multiple sites. A prototype provisioner specialised towards provisioning virtual machines across multiple sites is described in Section 3.2.

- **Deployers** are responsible for installing and initiating processes on provisioned infrastructure. Deployers may be generic, installing pre-packaged components based on an application workflow specification, or may be specific to certain data processing frameworks. In Sections 3.3 and 3.4, we identify a few different systems for large-scale data processing for which custom deployers may be useful.

- **Runtime services** collectively describe services for deploying and coordinating runtime agents that can be used to monitor or control the execution of processes. Deployers may assume some runtime responsibilities, but many of these responsibilities go beyond the scope of what would be considered to be the role of a deployer service. While these runtime services may be provided generically on certain e-infrastructure platforms, it may also be possible to install these services alongside application processes as part of the investigation workflow, in which case they must be accounted for by instances of the above three service types.

Microservices may be specific to particular e-infrastructure platforms or specific types of investigation, or may be generic, particularly for platforms that use virtualisation and common APIs such as OCCI or standardised application components. There is a clear lineage of dependence between the classes of service described above, with provisioners and deployers being particularly dependent on particular implementations of planners in order to be able to function. It is also possible that microservices can be collectively implemented within a single service, though the use of standardised descriptions for application workflows, infrastructure plans and service APIs make this less necessary.

Figure 13 provides a basic depiction of how the microservices fit into the overall research support architecture. The research community engage with RIs and e-infrastructure via VREs; a well-featured

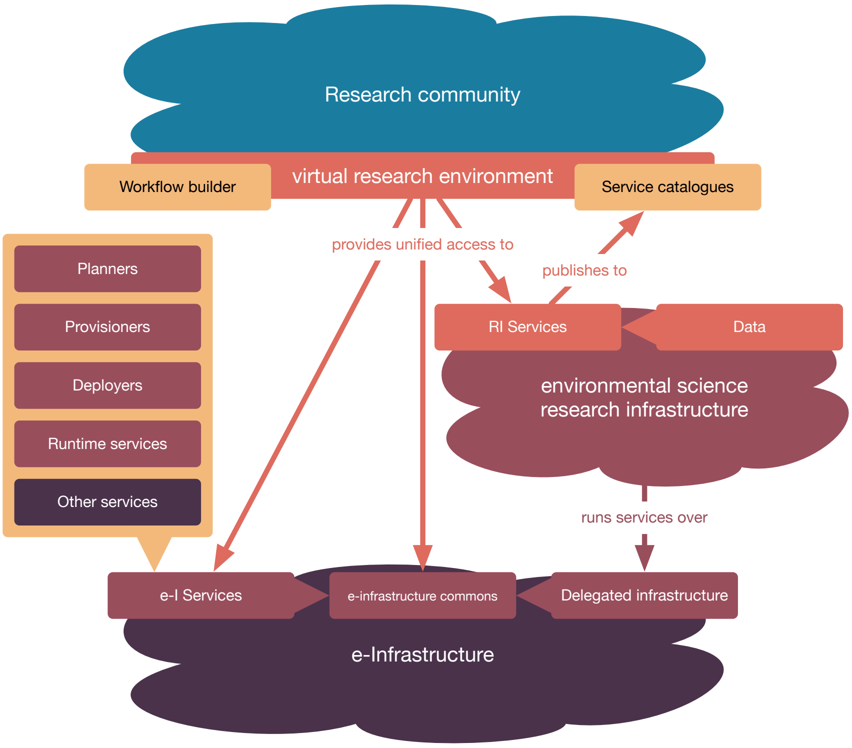

Figure 13: Optimisation microservices are provided via repository to e-infrastructure, and can be accessed by researchers via a VRE.

VRE provides the means to discover the research assets made available by RIs via a (possibly federated) service catalogue, and may also provide the means to compose an investigation workflow using those identified assets. Simple requests may be brokered directly through the e-RI, but more complex requests will require the customisation of e-infrastructure drawn from an e-infrastructure commons (such as EGI FedCloud). Such customisation can be managed by invocation of microservices hosted in some repository or marketplace linked to the e-infrastructure platform, and may be invoked via the VRE directly, or by the e-infrastructure via some internal manager.

Moving to a metadata-driven intelligent infrastructure requires the deployment of a number of different kinds of information service.

- **Metadata catalogues** identify and describe datasets and other research assets in a manner that allows them to be searched via their characteristic metadata. Application workflows are generally built from components discovered via these catalogues.
- **Service catalogues** are of a subset of metadata catalogue that describe and locate services in a range of contexts: VRE services for describing investigation workflows and navigating other service stacks; e-RI services for accessing and using research assets; and e-infrastructure services for acquiring and configuring compute resources. The main processing elements of workflows (as opposed to datasets and storage media) are drawn from this particular class of catalogue.
- **Knowledge bases** provide information about research infrastructure, resources or applications at different levels of abstraction, and provide a generic means for services to acquire knowledge about their operating environment. They also store the conceptual models used to define metadata schemas used by datasets and harvested for catalogues. The alternative to knowledge bases is for services to operate solely according to their internal logic based on their local environment, or on information directly passed to them as input. The use of separate knowledge bases allows services to operate with a global view of their operating environment, and benefit from collective information gathering. The determination of the specific information items that knowledge bases should gather, their distribution, and their placement, are non-trivial concerns

that must be addressed if intelligent information infrastructure is to be realised however. For example most e-infrastructure providers (for example EGI[11]) maintain some form of service catalogue that can be used by various meta-services for planning, scheduling, provisioning, *etc.*

- **Microservice repositories** actually host the optimisation microservices being discussed in this section. There are a number of locations in which such repositories can be placed in the architecture: they can be placed within an e-infrastructure platform (as part of its internal marketplace, most appropriate if services are specific to that e-infrastructure); they can placed within an e-RI (only appropriate if services are specific to e-RI services); they can be placed within a VRE (appropriate if services are more specific to a VRE's investigation design facilities than to the e-infrastructure on which they operate); or they can be hosted entirely externally (ideal if the optimisation services are wholly based on standard interfaces and specifications common to a range of both VRE and e-infrastructure services).

While optimisation microservices can be implemented to stand-alone to a certain extent, having these additional information services present, with standard APIs, would allow the microservices to be designed that can take advantage of an arbitrary degree of environmental knowledge. As the information architecture present in the research environment grows, so too does the potential for new and upgraded services.

[11] https://www.egi.eu/services/

3 Current development

The development focus of Task 7.2 is on the prototyping of microservices for e-infrastructure customisation, and for deployment of data processing environments. In particular, there has been an initial focus on time-critical applications—applications with deadline constraints on iterations of individual processes. There are two services that have already been prototyped and which are planned to be exposed to research infrastructures for demonstration and testing purposes in the near future:

- An **infrastructure planner** for selecting virtual resources to host applications with multiple internal deadlines.
- An **infrastructure provisioner** for provisioning an infrastructure topology across multiple sites in a transparent manner.

In addition to the development of services for planning and provisioning of infrastructure, another major focus of development is on specific large-scale data processing tools that can quickly process significant volumes of data on demand. The plan is to investigate some of these technologies in the context of selected ENVRIplus pilot studies, and then later to investigate how these technologies can be packaged so as to be more easily deployed for equivalent future investigations. The technologies so far examined are:

- Apache **Storm**, a framework for distributed real-time processing of streamed data.
- Apache **Cassandra**, a distributed, scalable NoSQL database system.
- Apache **Spark**, a distributed system for analysing large datasets in-memory.

These technologies have high potential in a range of different use-cases, and so could benefit from being provided to the wider ENVRIplus community as **deployment** services (as defined back in Section 2.3.2). In this section, we describe the aforementioned development studies in more detail.

3.1 Planning of virtual infrastructures for time-critical applications

A key problem when using virtualised infrastructure for experiments and other data-driven investigations is determining the best selection and configuration of virtual resources: what services to use, what VM images to install, how to ensure sufficient network connectivity between nodes, *etc.*

Typically this planning of custom infrastructure is performed manually as a collaboration between application developers and infrastructure engineers based on their combined experience. Automation of planning can be very useful; but different applications require different approaches based on the class of problem and their critical requirements.

Within Task 7.2, we want to prototype **planners for configuring e-infrastructure for specific classes of data-driven investigation**. In this section we discuss a particular planner prototype for a particular kind of application—a workflow of service invocations with multiple internal response deadline constraints. The major cost function are execution time (for service invocations), communication time (for data delivery) and resource cost (for leasing VMs; this can be monetary or some equivalent measure such as energy consumption). This does not necessarily represent all or even a majority of the applications of most interest in ENVRIplus, but at least provides an example of the kind of optimisation microservice that can be prototyped over the course of the project.

Problem specification

The selection of suitable resources to support data processing depends on a strong understanding of the type of processing needed. Some tasks are very compute-intensive, requiring access to powerful processors and large amounts of memory. Some tasks are more I/O-intensive, requiring fast access to storage or to other computing nodes. Some tasks are highly parallelisable, allowing the processing burden to be split across multiple concurrent processes all hosted on different machines—and some tasks simply need access to the most powerful machine available. There are many different kinds of

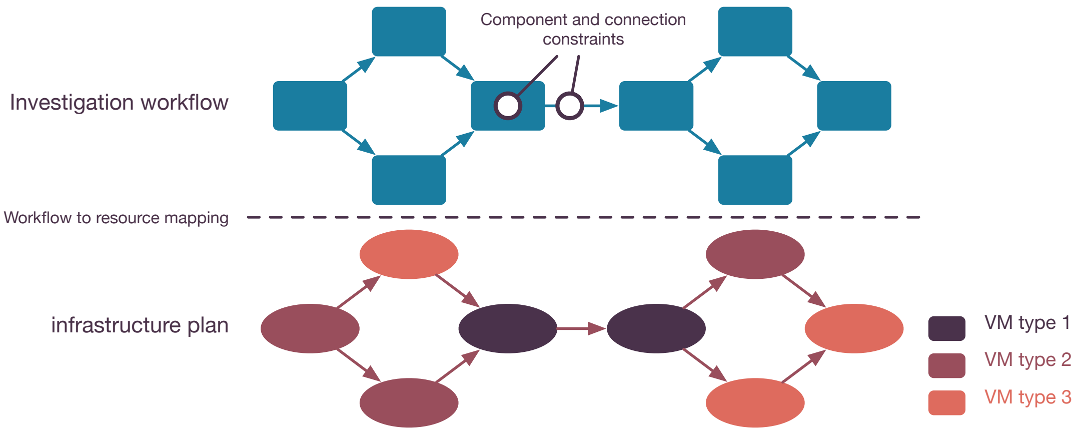

Figure 14: An infrastructure plan is composed for a given investigation workflow by selecting (virtual) resources based on the conditions and deadlines imposed on workflow elements.

process, and so correspondingly there are many types of resource offered by e-infrastructure providers. As datasets have grown larger and more difficult to transport, the networking between resources has also taken on an additional degree of importance. Thus the need for careful planning of infrastructure provided for specific applications.

The planner described here is one that is designed to build an infrastructure topology based on selection of virtual resources that should be able to meet multiple internal response time deadlines at runtime, while minimising some cost function, *e.g.* monetary cost or power consumption (see Figure 14). It assumes a persistent application workflow, where the constituent tasks are implemented as services that will continue to run for the entire investigation, and thus must continue to respond to new input under the same deadline constraints throughout that investigation's duration. We assume that the infrastructure resources selected exhibit a stable level of performance, with a stable 'price' for purchasing those resources that is maintained for the entire duration of an application's execution. Such a planner represents only one type of planner that may be of use for executing workflows generated to process data drawn from environmental science RIs.

Development

There exist a number of works that focus on optimal resource assignment on virtual infrastructure under different conditions and assumptions. Yu et al. [20] propose a method to minimise the execution cost of a workflow to satisfy a global deadline. Their method first clusters the sequential tasks that have only one parent and child together and assigns each task with a sub-deadline based on its minimum processing time and the sub-deadlines of its predecessor. Each task is then assigned to the least expensive virtual machine (VM) that can meet the deadline—however, the communication cost between tasks is not considered, nor the presence of multiple deadlines. The *Infrastructure-as-a-Service Cloud Partial Critical Paths* (IC-PCP) algorithm [1] calculates partial critical paths through the application workflow in order to schedule the deployment of tasks on the cloud in order to solve the same problem. IC-PCP can be combined with the approach taken by [20]; after finding a partial critical path, each task in the path is assigned a sub-deadline with the execution time in proportion to the whole partial critical path length. The tasks in the workflow are then assigned with the cheapest VMs that still meet these deadlines. Though originally formulated to meet a single global deadline, support for additional internal deadlines in IC-PCP can be easily added by overriding the generated sub-deadlines with pre-defined deadlines where the latter is more strict.

The planner we have prototyped is therefore based on the IC-PCP algorithm—however we make a number of assumptions different from those of [1]. For one, we assume that after one task transfers its results to all its successors, the VM where the task is deployed is not released, and instead the task will act as a persistent service waiting for more input—thus the deadline for a given task must be satisfied every time the task receives new input. We also make the assumption that every task in the workflow will be deployed on its own VM, partly for simplicity, and partly because sharing VMs impacts the performance of tasks [2] and our focus is on time-critical applications as described in Section 2.2.4. Most importantly however, we assume that workflows can have multiple internal deadlines on different component processes based on the requirements of users or downstream services.

Our multi-deadline workflow planning algorithm (MEPA) uses a 'compress-relax' method—VM types with best performance are assigned to tasks so that the makespan (the total execution time) is 'compressed' and all deadlines are met if possible, and then the assignment over the workflow is 'relaxed' by re-assigning to tasks less powerful VMs albeit with lower cost while preserving deadline satisfaction. Initially MEPA assigns each task in the workflow with the best performing VM to guarantee a basic solution; if not all deadlines can be met this way, then an alternative e-infrastructure will be needed, or else the QoS requirements of the application will need to be somehow relaxed. Based on the initial 'compressed' assignment, MEPA then calculates the earliest start time (EST), earliest finish time (EFT) and latest finish time (LFT) for each task based on the dependencies between tasks and necessary communication costs. MEPA then works backwards from the final tasks of the workflow to assign the internal deadlines; if the deadline on a task is stricter than the calculated LFT for that task, then the deadline simply replaces the LFT. If the EFT for a task exceeds its LFT, then the currently available resources cannot satisfy the time constraints on the workflow. Once the constraints on a critical path have been determined, it is then possible to determine the best assignment of VM type to each node on the path.

Actual assignment of different kinds of VM to different nodes in the same workflow can be based on brute-force calculations, or based on the use of heuristics. Convolbo and Chou [5] propose a heuristic approach which exploits the parallel properties of the workflow to minimise execution time. Rodriguez et al. [17] applies particle swarm optimisation, encoding within each particle a task-resource mapping. Heterogeneous Earliest Finish Time (HEFT) has been proved to perform better than other heuristics in robustness and schedule length [4], and Multi-Objective HEFT extends HEFT to optimise the trade-off between monetary cost and makespan of the workflow [7], though again the communication cost is not addressed. The critical path based iterative heuristic (CPI) [3] and multiple complete critical paths heuristic (CPIS) [2] are used in other algorithms for solving the cloud infrastructure planning problem within the bounds of a single deadline. Based on the calculated earliest finish time and latest finish time of individual tasks, CPI identifies a complete critical path through the application workflow from start to finish and assigns the tasks in the critical path to VM services. In CPIS, a graph labelling method is applied to construct complete critical paths of the kind generated by CPI.

In our case, VM types are assigned to the constructed partial critical path using a genetic algorithm and a matrix of execution costs per task per VM type (which can be based on historical observation or extrapolation). This genetic algorithm runs for a set number of generations to find the best combination of assignments to nodes on a critical path that fulfil all deadlines. After assignment, the tasks in the critical path are tagged as assigned and the EST, EFT and LFT of the other tasks in the workflow are updated accordingly. Assignment of the remaining tasks will the continue until all the tasks in the workflow are assigned.

Experimentation

To investigate the behaviour of the algorithm the graph generator GGen [6] was used to generate random workflow topologies with internal deadline constraints. Specifically, a 'fan-in/fan-out' procedure was used to generate directed acyclic graphs with different numbers of vertices, maximum in-degree per node and maximum out-degree per node. In order to test on different scales of graph, the number of vertices in the workflows ranged from 1 to 256. The in-degree and out-degree were used to generate different 'shapes' of workflow, with the in-degree and out-degree ranging from 1 to 5 and 1 to 4 respectively.

For every workflow there needs to be an execution profile describing the performance of tasks on different kinds of VM, as well as the projected communication cost between tasks. For our experiments, we first generated the execution cost of the task on the 'best' kind of VM, randomly selecting a response time between one and half of the communication cost upper bound; this ensured that the relative scale of execution and communication costs with respect to one another is varied across workflows. The execution costs of the task on the other 'lesser' services were generated iteratively by increasing the previously generated cost by a random amount, and a random value was assigned to the communication cost between workflow nodes.

The time constraints attached to a workflow were also randomly generated. The number of time constraints were set in proportion to the scale of the workflow—approximately 10% of the number of tasks in the workflow. Each deadline was then attached to a task in the workflow based on the critical path calculation performed during workflow generation, limiting each deadlines range based on best and worst performing VM services so as to ensure no 'impossible' deadlines are set.

Our implementation of MEPA was based on Python 2.7.10, using NetworkX (version 1.10)[12] to manage the workflow and PyDOT2 (version 1.0.33)[13] to parse the graphs generated by GGen. We used DEAP (Distributed Evolutionary Algorithms in Python) [8] as the underlying framework for implementing the genetic algorithm for assigning VM types to tasks on critical paths. Experiments were conducted on the Distributed ASCI Supercomputer 5 (DAS-5)[14].

A number of different performance experiments were carried out, including comparison of path assignment with IC-PCP and with our genetic algorithm; comparison of IC-PCP, CPI and MEPA to meet a single global deadline; and comparison of (a minimally modified) IC-PCP and MEPA to meet multiple deadlines. In the third experiment, IC-PCP was minimally augmented to support multiple deadlines, but retains the restriction of selecting a single VM type for all nodes, compared with MEPA's ability to mix VM types for different tasks. The full results have been submitted for publication in a forthcoming journal, but in essence show that MEPA consistently provides less expensive VM assignments than IC-PCP and produces results in line with CPI with greater scalability (due to lesser calculation complexity). Further experiments should focus on relaxing some of the assumptions made by MEPA in order to be applicable to wider range of time-critical application.

Integration into ENVRIplus

The planning algorithm detailed above shows some promise for a particular class of data-driven investigation—for example, the processing of data streams within a distributed pipeline that requires each major stage of processing to be carried out within a specific time window.

A RESTful service is being developed that can provide planning based on an e-infrastructure service catalogue for an abstract application topology with performance constraints. It is hoped that this kind of service can be deployed within e-infrastructures such as EGI for general use.

However, this planner is very limited in the range of applications it can support. To be truly useful to the ENVRIplus community, it will need to be either extended in its support for different kinds of constraint (time-critical or otherwise), or it should only be used as a starting point, with alternate planners provided exploring other classes of data processing problem.

3.2 Provisioning of virtual infrastructure across multiple sites

The manual provisioning of e-infrastructure resources for a given research application can be a difficult task for researchers, often requiring engineers knowledgeable about the target e-infrastructure to act on their behalf. The use of virtualised infrastructure, where the actual composition and configuration of physical resources is abstracted aside in favour of virtual machines (VMs) that users can configure freely, reduces this difficulty, but further automating the provisioning process can save even more time and effort, increasing the effective use of e-infrastructure by a research community accordingly.

Within Task 7.2, we want to examine **provisioners for setting up networked virtual infrastructure for hosting data processing workflows**. In this section we discuss a prototype provisioner for provisioning virtual machines in multiple locales in parallel that can ensure the network connectivity

[12] https://networkx.github.io/
[13] https://pypi.python.org/pypi/pydot2/1.0.33
[14] http://www.cs.vu.nl/das5/

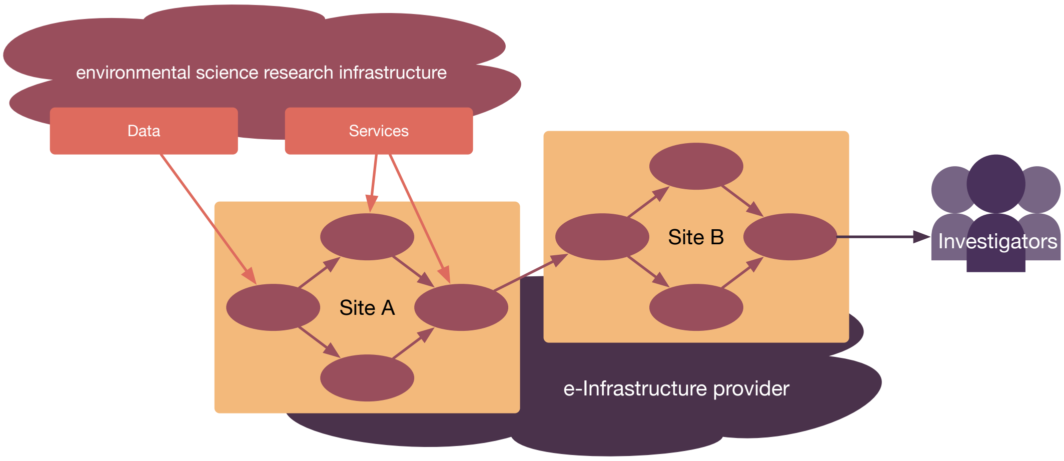
environmental science research infrastructure
Data
Services
Site A
Site B
Investigators
e-Infrastructure provider

Figure 15: Investigation workflows hosted on e-infrastructure may be provisioned across multiple sites in order to minimise data transfer costs and to take advantage of local compute facilities.

between components in multiple locales. This scenario is of interest in many cases, such as where there is need to cache datasets being distributed between different data centres or research infrastructures in such a way that data movement is minimised while ensuring that all data remains accessible, or where there is need for dedicated processing facilities only available at certain sites, but where those sites cannot host the entire application workflow.

Problem specification

Research applications can be deployed onto networked virtual infrastructure, but the layout of the network should be considered when planning such deployments, especially for applications that draw upon assets from more than one data centre or computing service. There may be a tension between different parts of the application as to where it is optimal to provision specific VMs, and hence it is important to consider how to 'co-provision' resources across multiple domains or data centres in order to minimise unnecessary data transfer and take advantage of the best available environments for data processing tasks (see Figure 15). Such co-provisioning involves three main steps:

1. Partitioning the original topology;
2. Establishing connections between partitioned parts;
3. Co-provisioning the partitioned parts simultaneously.

Moreover, co-provisioning can, in principle, make the provisioned infrastructure much more flexible and scalable by breaking complex infrastructure topologies into independent blocks that can be restored individually in the event of failures or faults.

A critical issue to address is how to create a network overlay that allows the partitioned parts to function and appear as a single topology. A co-provisioning mechanism that preserves the network topology across multiple sites has been proposed by Zhou et al. [24], based on cooperative provisioning (co-provisioning) on Clouds. An intermediary broker that exists between investigator and e-infrastructure provider performs the necessary modification of the resource plan without requiring extra attention from either party. This technology is now being adapted for the EGI FedCloud platform, as a service for investigators wishing to host tasks on the EGI e-infrastructure. A microservice encapsulating this technology may be of use however in a variety of contexts, as a means to accelerate the set up of experiments on a range of different virtualised e-infrastructure platforms.

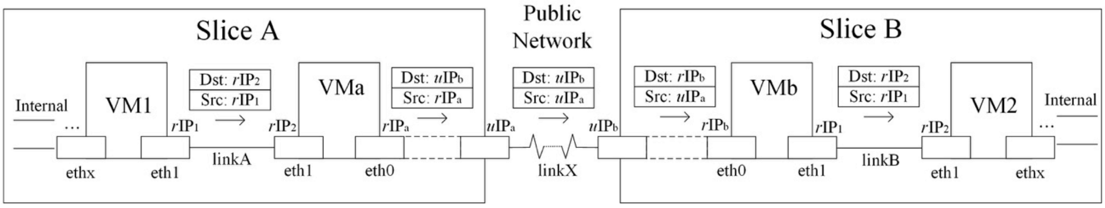

Figure 16: Establishing connectivity between partitions using proxy nodes.

Development

Many studies have been done on the topic of fast provisioning to accelerate the process of VM start-up in cloud environments, mostly focusing on the server (e-infrastructure) side. FVD [19] modifies the VM image format to make the start-up process shorter, but requires the hypervisor to be modified accordingly. SnowFlock [13] and Twinkle [25] adopt the method of directly forking from a running virtual machine to get rid of the start-up process. Zhang et al. introduce two solutions, VMThunder [23] and VMThunder+ [22], both of which can provision hundreds of VMs in a very short period. They provide further optimisation on top of prior peer-to-peer methods, but still require special configuration on the part of the e-infrastructure provider. Inter-cloud co-provisioning is another research direction; Grozev and Buyya [11] provide a detailed survey on inter-cloud architectures and provide a taxonomy to describe how co-provisioned resources collaborate with each other. Nelson and Uma [16] propose an Inter-cloud Resource Provisioning System (IRPS) to describe resources semantically and provision resources across clouds. They define a set of resource ontologies to work with, because management policies and descriptions about resources are different for different e-infrastructures. The target of their solutions are not applications that are particularly network-centric however, and resources are generally provisioned without considering network performance.

Fundamentally, the partitioning of a virtual infrastructure is based on characteristics of both application and infrastructure. Separate partitioning requests can be provisioned simultaneously, so reducing the total overhead time for provisioning. Two possible methods have been developed to settle the problem of connectivity between partitions. The first is to use proxy nodes specifically configured to manage cross-partition data transfer, the second is to use IP tunnelling for such transfers.

Figure 16 illustrates how one packet is sent through the public network between two partitions using the proxy node method. When using this method, every network link required by the infrastructure specification that crosses between two partitions requires the creation of two proxy nodes, one in each partition, to 'bridge' the gap between partitions. The role of a bridge node in a given partition is to act as recipient for any data packet destined for the other partition and to then redirect it across the public network to its partner node in that other partition, which will then redirect the packet to the 'real' node that the packet is destined for. When partitioning a specified infrastructure, it is necessary to determine the most efficient way to divide nodes such that the QoS requirements of the application can be met while at the same time minimising the number of proxy node pairs required to bridge across partitions. The necessary proxies are provisioned alongside the rest of the nodes in their respective partitions, and are treated within each partition as if they were the nodes on other partitions to which links are required. Upon provisioning of all partitions, these proxies are then provided with the information about their partner nodes in the other partitions that allow them to forward data packets across the network.

Another method to connect partitioned slices is to use IP tunnelling, as shown in Figure 17. With the IP tunnelling technique, a data packet provided by nodes destined for another partition can be wrapped in another packet via which the original packet can be transparently delivered over the public network. The advantage of this method is that it does not incur the extra overhead of establishing proxy nodes for every cross-site link, giving more flexibility to partition an infrastructure specification in accordance with QoS requirements. IP tunnelling however is not always available in all environments, and requires some minor re-configuration of provisioned nodes with regard to the original infrastructure specification. In that sense, it is not quite as 'transparent' to the client as the proxy node method.

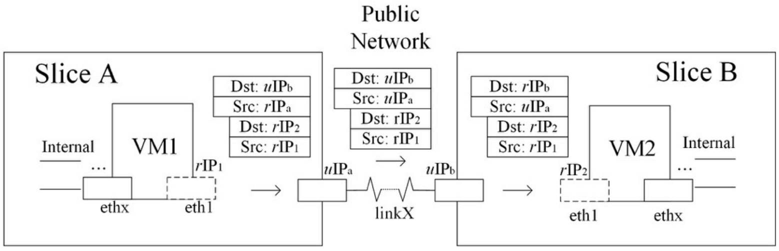

Figure 17: Establishing connectivity between partitions using IP tunnelling.

A key characteristic of this co-provisioning mechanism is that it does not require special configuration on the part of the infrastructure provider, and preserves the virtual topology of the original infrastructure plan, rendering it transparent both to the investigator (or the VRE acting on their behalf) and the e-infrastructure provider (which simply provides the required infrastructure resources for each partition as if they are separate requests). An initial implementation was tested on the ExoGENI platform[15], a networked infrastructure platform that permits configuration of the interstitial network between virtual machines.

Experimentation

Networked Infrastructure-as-a-Service (NIaaS) requires that the network topology between VMs is defined explicitly. Currently, most Infrastructure-as-a-Service platforms just provide VM nodes to clients without specifying how they are connected. The performance of many data-intensive applications however is constrained by the network, so consideration of the network links becomes important. ExoGENI uses the infrastructure and network description language (INDL) [10] to describe the topology of clients requests. The VM nodes and links between them are all described in one 'slice' (a virtual infrastructure provisioned for a specific application). In addition, ExoGENI permits clients to specify the location of their VMs, selected from a number of sites.

Measurements were performed on ExoGENI to better characterise the provisioning problem. As shown in [24], the provisioning overhead does not really depend on the VM type or topology as much as it does the number of nodes in the slice. A similar conclusion is drawn by Mao and Humphrey [15] based on measurements of public cloud providers. If we divide the slice into multiple smaller slices and provision them in different racks or locations in parallel, the provisioning overhead of every slice can be reduced. The strategy of the provisioner therefore is to provision cloud resources across multiple domains, or even multiple clouds, while ensuring that all resources can still operate together to provide a whole infrastructure as per the client's original design.

When adopting the proxy node method, every link across partitions brings in one more proxy node in each partition. We therefore want to minimise the need for extra nodes as much as possible given the QoS constraints presented and the topology of the infrastructure specification. When adopting the IP tunnelling method, the partitioning is slightly simpler due to not having to consider the impact of additional proxies. Regardless, in order to realise an effective partitioning algorithm, we define the weight of the VM nodes required by the infrastructure specification. As discussed, the provisioning overhead mainly depends on the type of VM nodes provisioned and the image installed on it, and the weight of each node should reflect this. For standardisation, we define the weight of the node with the most basic type and image available to the source e-infrastructure(s) as being of weight one. The weight of other nodes can be defined as multiples of this weight based on *a posteriori* measurements. To make provisioning as efficient as possible, the inferred weight of the partitions should be as close to one another as possible.

Some evaluation has been conducted in ExoGENI using the proxy node method to implement co-provisioning. This evaluation has mainly focused on the provisioning overhead. The experiment was designed using nodes on type 'XOMedium' (single core, 3 GB of RAM, 25 GB of storage).

[15] http://www.exogeni.net/

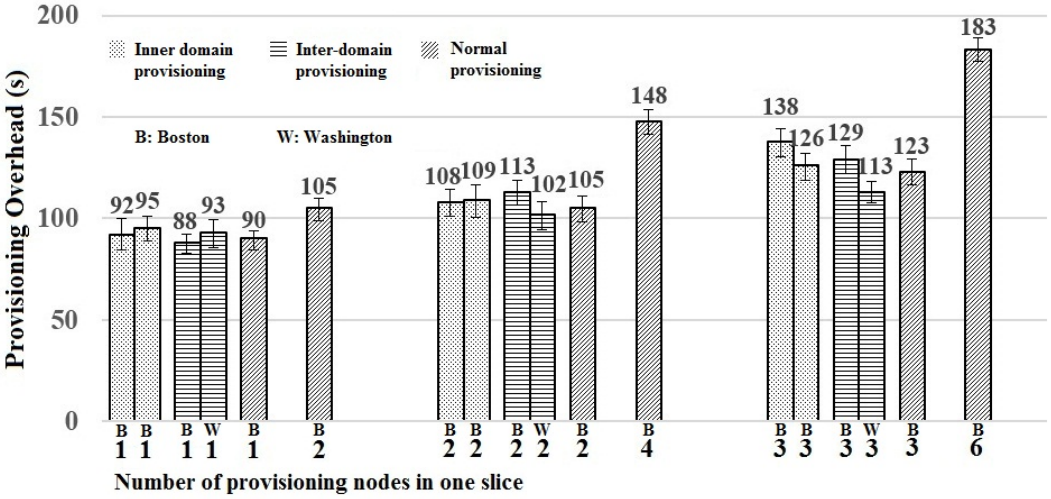

Figure 18: Measurements of provisioning overhead for different numbers of nodes provisioned together or in parallel partitions.

Three scenarios were investigated: 'normal' provisioning, *inner-domain* co-provisioning and *inter-domain* co-provisioning. Partitioning the infrastructure into just two equally-weighted parts reduces the provisioning overhead most. For inner-domain co-provisioning, we co-provision two partitions with equal numbers of nodes to servers hosted in Boston. For inter-domain co-provisioning, we co-provision the same two partitions, but with one hosted in Boston and the other in Washington. For normal (single partition) provisioning, we measure both the overhead when provisioning the partition with the same total number of nodes as used for the co-provisioning, and with just the number of nodes used in a single partition in each co-provisioning case as a control experiment. Measurements are repeated three times for every scenario and the mean average taken as the result. The results are shown in Figure 18 for infrastructure specifications consisting of in total two, four and six nodes in turn. The experimental results demonstrate that the overhead of the co-provisioning mechanism is close to that of the normal provisioning overhead for half the number of nodes. They show that parallel provisioning of partitions is faster than partitioning a single large infrastructure slice, much as expected, and that the benefit applies even when provisioning within the same locale, at least for ExoGENI. The results suggest that co-provisioning becomes more efficient as the number of nodes increases.

Integration into ENVRIplus

A RESTful service has been developed, packaging the above provisioning algorithm, using both methods of maintaining an overlay network across sites. It remains to provide versions that support the specific virtualised infrastructures that are being made available to RIs within the ENVRIplus project.

Identifying use-cases that would make good use of this provisioner remains necessary. Determining if there are other problems specific to infrastructure provisioning that can be solved by algorithms encapsulated by optimisation microservices is also important. There may be potential to customise the provisioning described here to assist with the deployment of distributed data processing environments such as those discussed in the next two subsections.

3.3 Using Storm and Cassandra for near real-time processing of time-series data

In this (and the next) section, we describe frameworks for large-scale data processing. These frameworks can potentially solve the problem of configuring and running data processing pipelines for a number of different application scenarios, whether for specific data-driven investigations, or as part

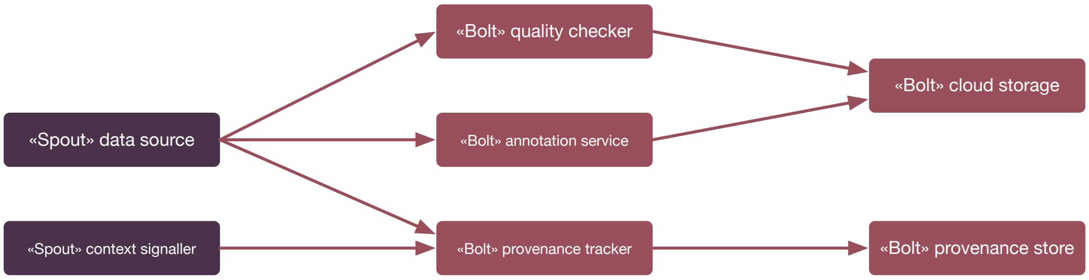

Figure 19: A storm topology defines a cluster of spouts and bolts for data emission and transformation respectively.

of the general data analysis cycle used by RIs to package and propagate their key research data products.

One of the things we wish to do in Task 7.2 is to build **deployment services for automatically configuring distributed data processing pipelines**. To do this, we need to identify existing technologies that are robust and easy to use, and then identify where we can package the deployment of those technologies as a service, minimising the need to build anew for every potential research data application. First, however, we need to analyse the potential of the candidate technologies, starting with Apache Storm and Cassandra.

Apache Storm

Apache Storm[16] is an interesting distributed computation system for real-time processing of streamed data of the sort curated by RIs. The free and open source Apache Software Foundation project is designed to "reliably process unbounded streams of data, doing for real-time processing what Hadoop did for batch processing". Apache Storm excels at real-time analytics, online machine learning, continuous computation and similar use cases.

The notion of a *topology* is a core Storm concept. A Storm application is a topology that runs forever, unless terminated. The structure of a Storm topology is a directed acyclic graph, consisting of vertices and edges as illustrated in Figure 19. Directed edges are streams, unbounded sequences of tuples. Tuples are data elements of primitive or user-defined data types. There exist two types of vertex: *spouts* and *bolts*. Spouts are sources of streams; they generate tuples. In contrast, Bolts transform streams; they perform operations on tuples.

In the context of ENVRIplus, Apache Storm is a promising tool for processing streamed data in real-time in an efficient manner. Data acquired by RIs from sensors, sensor networks, or observatories are delivered as streamed data, and RIs may want to perform computational operations on such streamed data prior to persistence. An example for such online computation may be transformations in data encoding and format, *e.g.* from binary encoded data to O&M-compliant encoding of observations [12]. Another example is the online annotation of observation streams designed to enrich observations with contextual metadata, *e.g.* information regarding time and space or other observed properties. A special case of such annotation is online quality control where streamed observations are annotated with a quality flag.

As sensing devices, and entire networks or observatories hosting such devices, generate observational data in a streamed fashion, *i.e.* until terminated, the Storm spout serves as a good abstraction for such infrastructure. Using this abstraction, instruments merely emit tuples that can be handled directly by a topology. A complex online real-time processing task can thus be broken down into simple tasks, each implemented by a Storm bolt. Bolts can then be assembled in a topology that implements the complete task.

[16] http://storm.apache.org/

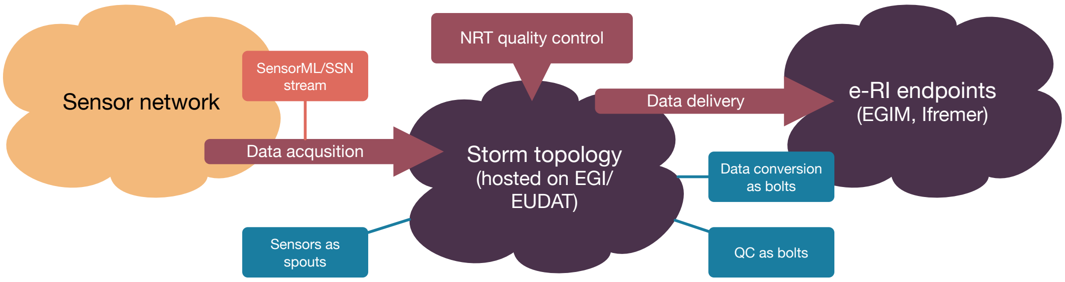

Figure 20: A Storm topology can be used to perform near real-time quality checking on sensor data streams.

Apache Storm can be configured to distribute the processing of streamed data on a cluster of machines. Collaboration has already been started with EGI to investigate how to deploy and execute Storm applications on EGI e-infrastructure. Within this context, EGI has been successful in setting up a Storm cluster consisting of three machines with an agreed access policy. As a next step, the execution of a simple Storm application will be tested. These steps are in preparation for the implementation, deployment and execution of a more complex Storm application designed to support the real-time acquisition and harmonisation of data from sensing devices, and the execution of real-time quality control routines on standardized observation data.

The details relevant for this Storm application are being developed as part of an ENVRIplus implementation case IC_14[17], illustrated by Figure 20. The aim of the implementation case is to investigate the possibility of shifting the data and metadata standardisation level closer to sensing devices so that transmitted data are encoded and formatted following international standards from the outset, using standards such as OGC Sensor Web Enablement[18] or the W3C Semantic Sensor Network (SSN) ontology[19]. Given such standardised transmission of sensor data and metadata, the implementation case aims at demonstrating the implementation of quality control routines that are common to ENVRIplus RIs. Hence, the IC attempts to demonstrate the re-use of quality control routine implementations across RIs. We expect that such re-use is facilitated by the standardisation of encoding and formatting of transmitted observational data.

Apache Cassandra

Apache Cassandra[20] is another potentially interesting system for persistence and retrieval of time-series data in RIs. The free and open source Apache Software Foundation project is a scalable, fault-tolerant, and decentralized NoSQL database system.

The Cassandra data model is interesting for sensor network based RIs as it can be designed to support the management of time-series data. Being a so-called "column store", Cassandra can not only order time-series data in rows but also in columns. Cassandra can hold up to 2 billion columns per row. It thus provides a compact representation of time-series data.

Persistence and retrieval of data in Cassandra relies on so-called row and column keys, *i.e.* references for the row and its column at which data is persisted or from which data is retrieved. A simple data model for sensor data may utilise the device identifier for row key and time for column key. Given time-ordered columns and a query for device identifier and a time interval, retrieving the series matching the query is thus a cheap operation in Cassandra.

Stocker et al. [18] have compared a system, called Emrooz, built on Apache Cassandra designed to persist and retrieve observational data represented according to the SSN ontology in RDF[21] with two other state-of-the-art RDF database systems. The study suggests that Emrooz outperforms

[17] https://confluence.egi.eu/x/a6N3es+of+GHGs
[18] http://www.opengeospatial.org/projects/groups/sensorwebdwg
[19] https://www.w3.org/TR/vocab-ssn/
[20] http://cassandra.apache.org/
[21] https://www.w3.org/RDF/

the compared systems in load performance of streamed observational data over time and has query performance independent of time-series length (size of data). The latter is a remarkable result, though easily explained by the fact that given the sensing attributes (*i.e.* sensing device, observed property and feature) and a time interval, Emrooz can efficiently retrieve the relevant data by addressing the corresponding Cassandra row and column keys. In contrast, RDF database systems need to evaluate the complex and expensive joins of a SSN observation query.

While the comparative study on RDF database systems is interesting, in ENVRIplus it may be useful to compare the load and query performance of Apache Cassandra with classical relational database management systems, such as PostgreSQL and MySQL. We plan to perform such an evaluation using data from the Euro-Argo RI provided by Ifremer.

Integration into ENVRIplus

Investigations proceed into both Storm and Cassandra, primarily driven within the project by the IC_14 implementation case. If the results of the use-case are positive, and it is deemed practical to standardise the process of deploying these technologies onto a standard e-infrastructure such as EGI, then there will be further investigation into the creation of dedicated microservice for (assisting with) deploying them onto that e-infrastructure.

3.4 Using Spark to query data as part of a data subscription service

Parallel to the investigation of Storm and Cassandra, experimentation has proceeded in using Apache Spark for handling fast data querying of a dataset.

Apache Spark

Similarly to various other scientific domains, the marine domain produces extensive amounts of research data that need to be stored and analysed. There are a number of technical platforms aiming to provide user-friendly access and functionality to the data throughout the whole data lifecycle. In particular, data analysis tools and platforms have been built that aim to provide scalable, efficient, and fault-tolerant operations over large distributed datasets.

Apache Spark[22] is an open-source, distributed system for analysing massive datasets through the use of in-memory computing. At its core is a scheduler/execution engine that organises operations across a number of worker nodes in a fault-tolerant and efficient way.

There are two types of operations: *transformations* and *actions*. Transformations define and create data structures known as Resilient Distributed Datasets (RDD) [21] and evaluate them lazily; only when an action operation is called is a possible chain of transformations executed and the action returns a value to the Spark application. There are several Spark modules which build on top of the core:

- The *SQL* module allows querying of structured data using the SQL query language.
- The *Streaming* module allows applications to process both streaming and historical data using the same execution engine.
- The *MLlib* module for machine learning provides various iterative machine learning algorithms.
- The *GraphX* module is a graph structured data analysis library.

One of the benefits of Apache Spark is that a variety of different analysis cases can be run through the same execution engine, which allows better performance and fault tolerance for example when graph data is combined with unstructured or tabular data.

There at least four aspects through which Apache Spark achieves fault tolerance and efficiency:

[22] http://spark.apache.org/

#	Nodes (+1 master)	Cores/ node	Memory/ node	Node type	Storage	Storage size
1	4 + 1	8	35 GB	I/O intensive, SDD	HDFS, 3 replicas; Parquet, 100 partitions	16 GB
2	8 + 1	16	78 GB	I/O intensive, SDD	HDFS, 3 replicas; Parquet, 100 partitions	5 TB

Table 1: Cluster configurations used for experimenting with Apache Spark.

Granularity of transformations The same low granularity transformations are applied many times on immutable data (in contrast to fine-grained updates to mutable states).

Data immutability Transformations are logged to provide data lineage, *e.g.* a lost partition of RDD can be derived from other RDDs by just recomputing that partition.

Data caching and partitioning in-memory Data access and movement costs are reduced by preserving transformations and keeping operations within memory rather than continuously touching storage.

Data locality Reducing data transmission and ensuring data availability.

Each Spark application can be run as a separate Hadoop YARN[23] or Apache Mesos[24] application, containing the master and workers, and each application shares resources with each other. As part of the investigation into a data subscription service for Euro-Argo data (one of the ENVRIplus technical use cases), two cluster configurations on top of OpenStack[25] were set up in order to investigate the potential of Apache Spark as the system for executing subscribed queries, as defined by Table 1. I/O intensive nodes were set up to give the best I/O performance on the virtual machine root and ephemeral disks, backed by local SSDs on the servers they ran on. The SSDs were configured in a RAID-0 configuration for maximal performance, which means an increased risk of loss of a node in case of hardware problems, but which may be compensated for by additional mirroring in the data subscription case. An anti-affinity policy was used to ensure that each node instance was run on a different host, again for performance reasons.

Two datasets were used in the experiments: first, a partial dataset representing June 2015 data; and second, a full dataset. The data was downloaded via FTP[26]. The Jun 2015 dataset size in Parquet[27] format was about 205 MB in size with 46 million entries, while the full data set in Parquet format was about 16 GB in size with around 4.2 billion entries.

For the first cluster configuration, the full dataset was queried; Table 2 shows the executed queries and the measured query time. Because only Spark versions above 2.0 include nested SQL query functionality and at the time of experimentation only a lesser version was available, more complex queries had to be implemented using additional look-ups, joining the results (necessary for query 8 in Table 2).

For the second cluster configuration, Table 3 shows the executed queries and measured query times. The second subset of queries (queries 2a–2d) were implemented through the use of Spark transformations and actions. The queries shown in this table were those demonstrated during the 3rd ENVRI week (November 2016).

The measurement of each query includes time to read data from HDFS/Parquet, to process the query, and to write result data to HDFS in CSV format. For the full dataset, each measurement was repeated 20 times, with the exception of query 1c, which was only run 19 times, and query 1a (extracting the entire dataset), which was not practical to run in this context. Average time, median time and standard deviation (σ) were derived for each measurement set. For each repetition, a new

[23] http://hadoop.apache.org/

[24] http://mesos.apache.org/

[25] https://www.openstack.org/software/

[26] ftp://ftp.ifremer.fr/ifremer/coriolis/co0547-bigdata-archive

[27] http://parquet.apache.org/

#	Query	Time (s)
1	SELECT station id FROM euroargo WHERE station id = 518167231	10
2	SELECT MAX(latitude), MIN(latitude), MAX(longitude), MIN(longitude) FROM euroargo	40
3	SELECT SUM(parameter value) FROM euroargo WHERE parameter code = 146	22
4	SELECT station id, SUM(parameter value) FROM euroargo WHERE parameter code = 146 GROUP BY station id	45
5	SELECT station date FROM euroargo WHERE station date > 2010-10-29 20:00:00 AND station date < 2010-10-29 22:30:00	235
6	SELECT SUM(parameter value) FROM euroargo WHERE parameter code = 9 AND station date > 2010-1-29 20:00:00 AND station date < 2010-12-29 22:30:00	42
7	SELECT MAX(parameter value), MIN(parameter value) FROM euroargo WHERE parameter code = 35 AND station date > 2010-1-10 20:00:00 AND station date < 2011-12-29 22:30:00	100
8	SELECT AVG(parameter value) FROM sealevel WHERE parameter code = 82	84

Table 2: Retrieval times for various queries on a subset of Euro-Argo data using the 1st cluster configuration.

#	Query	Time (s)				
		Jun 2015	Full dataset			
			Avg	Med	σ	runs
1a	SELECT * FROM DataSet	10	-	-	-	-
1b	SELECT * FROM DataSet WHERE measure type = 1	9	249	252	10	20
1c	SELECT * FROM DataSet WHERE ((latitude BETWEEN -5.61 AND 37) AND (longitude BETWEEN 28 AND 41)) OR ((latitude BETWEEN 0 AND 20) AND (longitude BETWEEN 41 AND 45.7))	5	106	107	5	19
1d	SELECT * FROM DataSet WHERE measure type = 1 AND ((latitude BETWEEN -5.61 AND 37) AND (longitude BETWEEN 28 AND 41)) OR ((latitude BETWEEN 0 AND 20) AND (longitude BETWEEN 41 AND 45.7))	5	104	106	5	20
2a	marel df = platform.filter(col("description").like("%Marel-Iroise%")); aa = df.join(marel df, df.platform code == marel df.id)	9	333	220	207	20
2b	dyfamed df = platform.filter(lower(col("name")) == "dyfamed"); bb = df.join(dyfamed df, df.platform code == dyfamed df.id)	9	343	219	178	20
2c	provor df = platform.filter(col("name") == "PROVOR III Profiling Float"); cc = df.join(provor df, df.platform code == provor df.id)	10	407	347	200	20
2d	pp df = platform.filter(col("name").like("%POURQUOI PAS?%")); dd = df.join(pp df, df.platform code == pp df.id).filter(col("measure type") == 13)	5	217	130	115	20

Table 3: Queries used to test Apache Spark on Euro-Argo data using the 2nd cluster configuration.

Spark application was initiated; the measured time includes only the running time of the application, with idle time due to YARN scheduling excluded. Table 3 shows that for the June 2015 dataset, query times on average were close to what is commonly understood as interactive use (less than 7 seconds). In addition, there is some variation in query times for the second query subset. The factor causing this was not yet detected and calls for enhanced experiment design. Nevertheless, if Apache Spark is utilized as a part of data subscription service, some variation in processing time is allowed due to the non-blocking interaction model of the service.

Integration into ENVRIplus

Investigations proceed into Spark within the context of the TC_2 (data subscription) technical case. Based on the positive results detailed above, there is a strong impetus for further investigation, including the creation of another microservice.

4 Roadmap

Task 7.2 is conducted in the context of the *Data for Science* theme of ENVRIplus, wherein each work package and task contributes to the overarching goals of the theme. In particular, the execution of the task, and the contributions of its participants, must be properly aligned with concurrent activities in other tasks, especially the provisioning of demonstrators and other prototype services.

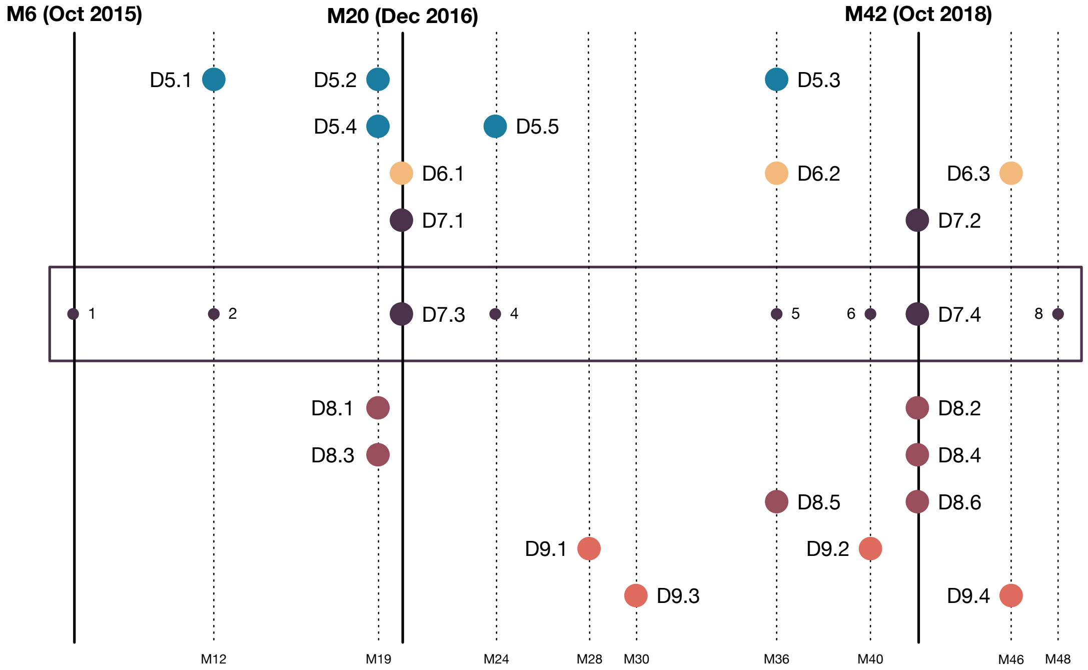

Figure 21: Timeline of *Data for Science* theme deliverables.

Figure 21 shows the timeline for deliverables in the theme, and identifies the key interface points with respect to Task 7.2 specifically:

1. **Beginning of Task 7.2.**
2. Delivery of gathered RI requirements.
3. **Delivery of D7.3.** By this time, design documents have been delivered for almost all major theme activities.
4. Delivery of a model architecture for guiding RI development.
5. Delivery of semantic linking model and design of provenance subsystem.
6. Delivery of demonstrator for service deployment on e-infrastructures.
7. **End of Task 7.2 and delivery of D7.4.** By this time, demonstrators have been produced for almost all major theme activities.
8. End of the ENVRIplus project.

Where possible, the activity in Task 7.2 should be aligned closely with concurrent activities in other tasks in the project. The activity should take advantage of results produced by the other tasks, and should also in turn influence their final outputs. In particular:

- The development conducted within Task 7.2 should comply with the reference model based architecture developed within Work Package 5.
- Where possible, an interface should be established to the interoperable data processing services provided by Task 7.1.

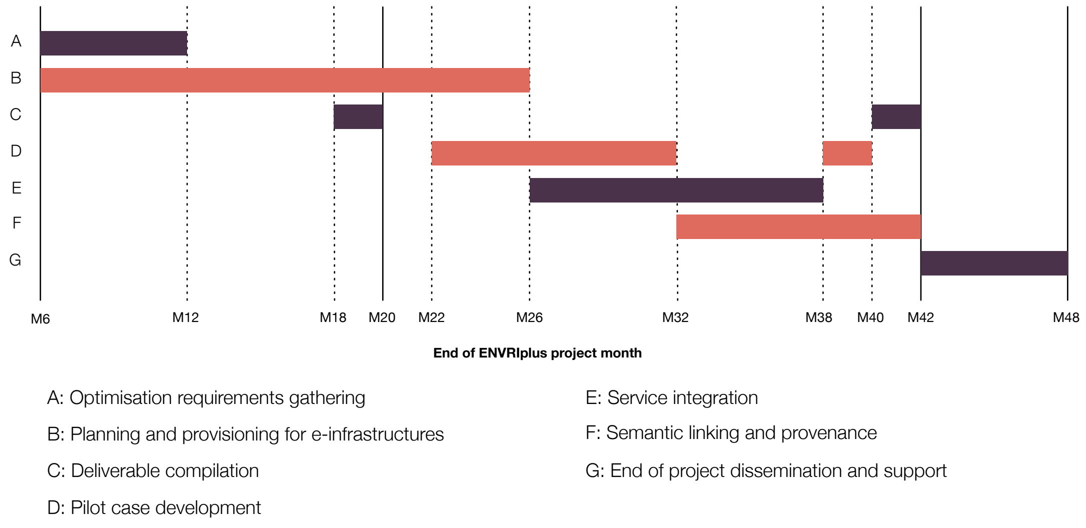

Figure 22: Projected roadmap for Task 7.2 within ENVRIplus.

- The technologies developed within Task 7.2 should be deployable on e-infrastructures as prescribed by Work Package 9.
- Technologies should be made available for specific use-cases identified within the theme to encourage uptake by research infrastructures.

In Section 1.1, four sub-tasks were explicitly identified as required by the ENVRIplus description of work. In addition, a number of opportunities have been identified based on the current and projected progress of the project.

There are 41 person-months associated with Task 7.2, divided among seven partners: UvA (18 PMs), EISCAT (6 PMs), UniHB (6 PMs), ETHZ (4 PMs), CSC (3 PMs), CINECA (2 PMs) and INGV (2 PMs). The spread of person-months dictates that not all partners can be active within the task throughout its entire lifespan. In order to best fulfil the requirements of the task, it is necessary to define several strands of activity that will be activated at different times in the project.

Seven main strands of activity have been identified in Figure 22, one of which has already been completed:

Optimisation requirements gathering Conducted under the auspices of Task 5.1 "Review of existing RIs" and reported in Deliverable 5.1 "A consistent characterisation of existing and planned RIs".

Completed April 2016.

Provisioning e-infrastructure for research applications Focuses on development of a few prototype services for planning and provisioning virtual infrastructure resources for distributed 'time-critical' applications as described in Section 3, as well as tools for the deployment of application components on the provisioned infrastructure.

Completion June 2017.

Deliverable compilation Production of this deliverable and later D7.4.

D7.3 completed December 2016; D7.4 completion October 2018.

Pilot case development Development of pilot cases for applying optimisation technologies to specific RI requirements, focusing (provisionally and not exclusively) on the EISCAT_3D, EMSO, Euro-Argo and EPOS infrastructures.

Initial development April 2017–December 2017; final integration June 2018–July 2018

Service integration Integration of optimisation technologies at VRE, RI and e-infrastructure levels. Also includes the design of deployers for selected large-scale data processing technologies on selected e-infrastructure.

November 2017–June 2018.

Semantic linking and provenance Prototyping of semantic linking framework for guiding optimisation of infrastructure, and integration of provenance recording based on recommendations of Task 8.3.

May 2018–October 2018

End of project dissemination and support Covers the gap between the end of Task 7.2 and the end of the project at large. No formal effort is scheduled for activities over this period, however some dissemination of optimisation technologies can still take place, as well as some support for their use in order to ensure some uptake that will outlast ENVRIplus itself.

November 2018–April 2019

Because of the interdependence with other tasks, and the standard risks associated with this kind of development work, the precise timeline for activities may change. Deliverable 5.4 "A development plan for common operations and cross-cutting services based on a network of data managers and developers" defines a basic schedule for the prototyping of common services across the entire Data for Science theme. According to this plan, there are four main phases of development: *familiarisation* (M19–M24), *development* (M19–M24), *deployment as prototype* (M30–M33) and *upgrading mechanism* (M30–M36). As can be seen, the roadmap described above broadly fits into this framework, with most development carried out by April 2016 (M26), and the deployment of tools in the context of specific RIs carried out by December 2017 (M32). Familarisation is treated as an activity concurrent to development, and has been on-going since the start of Task 7.2. 'Upgrading' can be seen as subsumed by the service integration strand of activity, which concerns the requirements of actual deployment onto e-infrastructure. Some additional work is carried out after April 2018 (M36); this does not adhere to the base plan, but is unavoidable due to the relative late delivery of outputs by tasks such as Task 5.3 (semantic linking) and 8.3 (provenance), which are considered important to Task 7.2's overall design.

5 Summary

The optimisation task (Task 7.2) of ENVRIplus is concerned with services for optimising the use of e-infrastructure by researchers who wish to make use of the data, tools and services offered by environmental science research infrastructures. Such optimisation relies on the existence of tools to customise e-infrastructure deployments, of technologies for large-scale data processing, and of mechanisms for translating user-oriented investigation requirements into system-oriented infrastructure requirements.

This deliverable provides a design vision for optimisation services, and describes how this vision fits into the greater design for common, interoperable services being promoted by the ENVRIplus 'Data for Science' theme. It also describes some of the current technical developments and investigations that are being carried out in the context of the task which might contribute to the final output of the ENVRIplus project. Finally, the deliverable provides a roadmap for the remaining 22 months of the task, based on the recommendations made by other project deliverables and future milestones for concurrent activities within the project.

Based on the analysis thus far, a number of technical recommendations can be made in this deliverable:

- Technical development within this task should focus on providing a small set of useful microservices for automating certain aspects of e-infrastructure customisation (including provisioning of virtual resources and automatic deployment of processes) and the configuration of large-scale data processing tools (such as Apache Storm and Spark) for use in data-driven investigations. Development should follow the recommendations of Task 5.4 wherever possible.
- These microservices should be made available on key e-infrastructure platforms and presented to RI communities as tools to be used as they wish. This requires collaboration with partners engaged in Work Package 9.
- The focus of development should be on providing useful services at the common e-infrastructure level, rather than only within specific RIs or researchers' private devices. These services should be able to interact with RI-specific services at a programmatic level, and should be invocable within suitable virtual research environments.
- Further investigation of the translation of application-level requirements into constraints on e-infrastructure (as a basis for guiding e-infrastructure customisation microservices) is required. The results of such investigation should be compatible with the semantic linking framework being developed in Task 5.3.
- The selection of specific development cases (for microservices) should be driven by practical use-cases provided by engagement with RI developers involved in ENVRIplus. A number of promising candidates have already emerged.

While the general use of e-infrastructure is becoming slowly more standardised, there are still significant differences between different e-infrastructure platforms, and the separation between VRE, e-RI and e-infrastructure is rarely as distinct as portrayed by abstract architecture in deliverables such as these. It remains therefore necessary to continue to monitor and study different e-infrastructure providers (EGI, EUDAT, DIRAC, *etc.*) and their relationships (for example, DIRAC runs some services over EGI), in order to determine how best to make optimisation services compatible with them and how best to present those services to their user communities.

These recommendations should now be fed back into the activities of the task, and where possible amalgamated with activities in other tasks where partners in Task 7.2 contribute concurrently.

References

[1] Saeid Abrishami, Mahmoud Naghibzadeh, and Dick HJ Epema. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Computer Systems*, 29(1):158–169, 2013.

[2] Zhicheng Cai, Xiaoping Li, and Jatinder N. D. Gupta. Heuristics for provisioning services to workflows in XaaS clouds. *IEEE Transactions on Services Computing*, 9(2):250–263, 2016.

[3] Zhicheng Cai, Xiaoping Li, and Jatinder N.D. Gupta. Critical path-based iterative heuristic for workflow scheduling in utility and cloud computing. In *International Conference on Service-Oriented Computing*, pages 207–221. Springer, 2013.

[4] Louis-Claude Canon, Emmanuel Jeannot, Rizos Sakellariou, and Wei Zheng. Comparative evaluation of the robustness of DAG scheduling heuristics. In *Grid Computing*, pages 73–84. Springer, 2008.

[5] Moïse W Convolbo and Jerry Chou. Cost-aware DAG scheduling algorithms for minimizing execution cost on cloud resources. *The Journal of Supercomputing*, 72(3):985–1012, 2016.

[6] Daniel Cordeiro, Grégory Mounié, Swann Perarnau, Denis Trystram, Jean-Marc Vincent, and Frédéric Wagner. Random graph generation for scheduling simulations. In *Proceedings of the 3rd international ICST conference on simulation tools and techniques*, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[7] Juan J Durillo and Radu Prodan. Multi-objective workflow scheduling in Amazon EC2. *Cluster Computing*, 17(2):169–189, 2014.

[8] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13(Jul):2171–2175, 2012.

[9] Ian Foster and Carl Kesselman. Scaling system-level science: Scientific exploration and it implications. *Computer*, 39(11):31–39, 2006.

[10] Mattijs Ghijsen, Jeroen Van Der Ham, Paola Grosso, Cosmin Dumitru, Hao Zhu, Zhiming Zhao, and Cees De Laat. A semantic-web approach for modeling computing infrastructures. *Computers & Electrical Engineering*, 39(8):2553–2565, 2013.

[11] Nikolay Grozev and Rajkumar Buyya. Inter-cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 44(3):369–390, 2014.

[12] ISO 19156:2011. Geographic information—Observations and measurements, 2011.

[13] Horacio Andres Lagar-Cavilla, Joseph Andrew Whitney, Adin Matthew Scannell, Philip Patchin, Stephen M Rumble, Eyal De Lara, Michael Brudno, and Mahadev Satyanarayanan. Snowflock: rapid virtual machine cloning for cloud computing. In *Proc of the 4th ACM European conference on Computer systems*, pages 1–12, 2009.

[14] Phillip A Laplante and Seppo J Ovaska. *Real-time systems design and analysis: tools for the practitioner*. John Wiley and Sons, 2011.

[15] Ming Mao and Marty Humphrey. A performance study on the VM startup time in the cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 423–430. IEEE, 2012.

[16] V Nelson and V Uma. Semantic based resource provisioning and scheduling in inter-cloud environment. In *Recent Trends in Information Technology (ICRTIT)*, pages 250–254. IEEE, 2012.

[17] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing*, 2(2):222–235, 2014.

[18] Markus Stocker, Narasinha Shurpali, Kerry Taylor, George Burba, Mauno Rönkkö, and Mikko Kolehmainen. Emrooz: A scalable database for ssn observations. In *First Joint International Workshop on Semantic Sensor Networks and Terra Cognita, Bethlehem, PA*, pages 11–12, 2015.

[19] Chunqiang Tang. Fvd: A high-performance virtual machine image format for cloud. In *USENIX Annual Technical Conference*, 2011.

[20] Jia Yu, Rajkumar Buyya, and Chen Khong Tham. Cost-based scheduling of scientific workflow applications on utility grids. In *First International Conference on e-Science and Grid Computing (e-Science'05)*. IEEE, 2005.

[21] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[22] Zhaoning Zhang, Dongsheng Li, and Kui Wu. Large-scale virtual machines provisioning in clouds: challenges and approaches. *Frontiers of Computer Science*, 10(1):2–18, 2016.

[23] Zhaoning Zhang, Ziyang Li, Kui Wu, Dongsheng Li, Huiba Li, Yuxing Peng, and Xicheng Lu. Vmthunder: fast provisioning of large-scale virtual machine clusters. *Parallel and Distributed Systems, IEEE Trans on*, 25:3328–3338, 2014.

[24] Huan Zhou, Yang Hu, Junchao Wang, Paul Martin, Cees De Laat, and Zhiming Zhao. Fast and dynamic resource provisioning for quality critical cloud applications. In *2016 IEEE 19th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 92–99. IEEE, 2016.

[25] Jun Zhu, Zhefu Jiang, and Zhen Xiao. Twinkle: A fast resource provisioning mechanism for internet services. In *INFOCOM, 2011 Proceedings IEEE*, pages 802–810. IEEE, 2011.