UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Virtual Infrastructure Optimisation

Koulouzis, S.; Martin, P.; Zhao, Z.

DOI
10.1007/978-3-030-52829-4_11

Publication date
2020

Document Version
Final published version

Published in
Towards Interoperable Research Infrastructures for Environmental and Earth Sciences

License
CC BY

Link to publication

Citation for published version (APA):
Koulouzis, S., Martin, P., & Zhao, Z. (2020). Virtual Infrastructure Optimisation. In Z. Zhao, & M. Hellström (Eds.), *Towards Interoperable Research Infrastructures for Environmental and Earth Sciences: A Reference Model Guided Approach for Common Challenges* (pp. 192-207). (Lecture Notes in Computer Science; Vol. 12003). Springer. https://doi.org/10.1007/978-3-030-52829-4_11

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:11 Feb 2023

Virtual Infrastructure Optimisation

Spiros Koulouzis, Paul Martin, and Zhiming Zhao(✉)

Multiscale Networked Systems, University of Amsterdam, 1098XH Amsterdam, The Netherlands
{s.koulouzis,z.zhao}@uva.nl, pwmartin.research@gmail.com

Abstract. The increasing volumes of data being produced, curated and made available by research infrastructures in the environmental science domain require services able to optimise the delivery staging and process of data on behalf of researchers. Specialised data services for managing the data lifecycle, for creating and delivering data products, and for customised data processing and analysis, all play a crucial role in how these research infrastructures serve their communities, and many of these activities are time-critical needing to be carried out frequently within specific time windows. We describe our experiences identifying the time-critical requirements of environmental scientists making use of computational research support environments. We also present a microservice-based infrastructure optimisation suite, the Dynamic Real-time Infrastructure Planner, used for constructing virtual infrastructures for research applications on demand. This chapter is partially based on a recent paper presented in [1].

Keywords: Infrastructure optimization · Cloud computing

1 Introduction

The ENVRI community works together to provide shared technological and governance solutions for data-driven science, in particular defining common operations for environmental research infrastructures and identifying and adopting technologies that implement those operations. Addressing the need for interoperable services for such diverse topics as identification and citation, curation, provenance and cataloguing, the Data for Science theme of ENVRIplus brought together a cluster of environmental research infrastructures (RIs) and (Information and Communication Technologies) institutions to come up with practical solutions to long-standing problems in such diverse areas as identification and citation, curation, cataloguing, processing and provenance. One particular area of interest, however, was optimisation; particularly the optimisation of virtual infrastructure used to support scalable data workflows needed both by RIs as part of their own internal data pipelines, and by RI users as part of their data science applications. Therefore, it is necessary to provide sufficiently advanced computational networked infrastructure to manage both the transportation of large (distributed) datasets and the data-intensive processing of such datasets.

© The Author(s) 2020
Z. Zhao and M. Hellström (Eds.): Towards Interoperable Research Infrastructures for Environmental and Earth Sciences, LNCS 12003, pp. 192–207, 2020.
https://doi.org/10.1007/978-3-030-52829-4_11

Performance is a crucial factor for many scenarios involving research support environments, influencing the quality of experience factors such as responsiveness to requests, to more system-level concerns such as efficient load distribution across distributed nodes in a confederation of data services. An example of a performance-critical system involving environmental data would be an early warning system where real-time sensor data have to be analysed quickly enough to identify events and provide adequate time for response. Even in non-emergency contexts, there are many cases where RIs collect real-time data continuously from sensors for swift processing to provide "nearly real-time" services to researchers. The specific example used in this paper is that of a data subscription service whereby updates to tailored subsets of a dataset are pushed to subscribers within a requested deadline. Notably, these services often cut across research support environments; RIs provide the service but delegate the hosting and management of the data processing pipeline to an e-infrastructure, generally to take advantage of elastic infrastructure resources rather than provide dedicated infrastructure within their data centres (which often operate as loose confederations with limited budgets for services beyond data curation and publication). Virtual Research Environments (VREs) may also be involved as part of the interface with researchers: for example, to subscribe to RI services or retrieve (and process) the results from such services.

To deliver acceptable performance, time-critical applications thus rely not only on the infrastructure for parallel computing or fast communication between components but also on optimisation of system-level application behaviour [2, 3]. The customisation of the infrastructure must consider performance constraints on applications at run-time as well as the utilisation and cost of the underlying resources across applications [4, 5]. In this chapter, we present a smart infrastructure optimisation engine, called Dynamic Real-time Infrastructure Planner (DRIP), that has been developed to bridge the gap between application requirements and service delivery on the part of research support environments, to optimise the quality of service at all levels. DRIP can be used to deploy, control and manage the kinds of distributed data pipelines needed for advanced RI services on the Cloud-based infrastructures now being provided by e-infrastructures.

2 Requirements and State of the Art

In this section, we analyse the basic requirements for service performance optimisation for time-critical data services in research support environments, review the state-of-the-art in real-time systems that may bear an impact on the development or operation of such data services, and summarise the essential challenges for time-critical data services on modern e-infrastructures.

2.1 Requirements

When we refer to time-critical applications, we do not usually mean speed-critical applications in the sense of applications that simply need to minimise the completion time (i.e. must be run fast). True "real-time" applications are characterised by bounded response time constraints on inputs, with certain consequences upon failure to meet deadlines [6]. Based on those consequences, real-time applications are referred to as hard real-time

if any missed deadline leads to immediate failure of the application, soft real-time if missing deadlines merely leads to degradation of user experience, and firm real-time if failure is brought about by too many missed deadlines in succession. Nearly real-time applications meanwhile are those with an intrinsic yet bounded delay introduced by data processing or transmission. Note that this does not make all nearly real-time applications "soft"-such applications can still impose a hard requirement for processing to fall within the permitted bounds. We might consider most processes in research support environments to be soft insomuch as failure to meet deadlines is usually not immediately disastrous. However, processes that are continuously run in tandem with real-time data acquisition can be seen to be "firm" due to the cascading impact of repeated failure to process their inputs on time; similarly, any highly parallelised workflow with bottlenecks in the data pipeline can suffer a precipitous drop in general quality of service if delays in one parallel element impact a non-parallelised bottleneck downstream. When we refer to time-critical applications, we therefore generally mean real-time or nearly real-time applications that are "firm" (or harder) in terms of the consequences of failure to meet the quality of service requirements. True hard real-time constraints in research infrastructure are rare but may emerge in particular for safety-critical applications that depend on real-time observational data.

In practical terms, the "firmness" of a response time constraint dictates the degree of a limited resource that should be allocated to ensuring the constraint. Isolated failures do not have the same impact as failures that beget further failures. It may be possible (and desirable) in specific research support environments to be able to assign a metric to constraints based on firmness that can be translated into concrete resource level requirements or adaptation strategies so that this information can be passed to optimisation services that must prioritise particular services or metrics. The requirements for optimising performance in research support environments are mainly dominated by the requirements of the data-centric research activities (the simplest but most important being the retrieval of specific datasets on request) that demand high performance or responsiveness. Within RIs, services are often developed with time constraints imposed on the acquisition, processing and publishing of real-time observations, in scenarios such as disaster early warning [7]. For VREs and RIs, performance factors are strongly influenced by the time needed to customise the runtime environment and to schedule the workflow applications [8]. Steering of applications during complex experiments is also temporally bounded [9]. Computing tasks and services provided by e-infrastructures are managed and offered to clients based on service-level agreements (SLAs). Time constraints are also imposed on the scheduling and execution of tasks that require high performance or high throughput computing (HPC/HTC). The overhead introduced by the customisation, reservation and provisioning of suitable infrastructure, the monitoring of runtime behaviour for infrastructure, and the support for runtime control also needs to be reduced and maintained within minimum levels. Failure recovery for deployed services and applications in real-time is also important when supporting time-critical applications; time constraints are not only imposed on failure detection, but also on decision-making and recovery.

2.2 Related Work

Within the Cloud context, many approaches have been proposed to address the scheduling, scaling and execution of tasks with deadlines. The majority of these proposals, however, adopt the viewpoint of the Cloud provider, which is often concerned with optimal VM placement on physical machines [10, 11]. In other cases, complex scheduling algorithms consider only the planning phase and do not react at runtime to changes in performance or failures [12]. Moreover, the majority of these approaches consider either synthetic tasks and workloads, simulated Cloud environments or both [13, 14].

2.3 State of the Art

The fulfilment of most time-critical requirements for research support environments relies on optimal execution of tasks on e-infrastructures, as well as efficient movement of data across networks. We identify several categories of time-critical application.

Time-Critical Information Search and Query. Typical technologies for real-time data query and search model the search activities of users, and their projected needs, predicting future queries [15], optimising catalogues [16] or prioritising urgent tasks [17], as well as optimising the presentation of contextual information [18]. Information retrieval is a core part of many data services and may require the retrieval of multiple datasets to answer a given query or considerable internal processing of data files for document-oriented data.

Time-Critical Workflow Execution. Time-critical constraints on workflows are typically expressed as deadlines for completing (part of) the workflow, or for responding to invocations or events within a certain time window. Scheduling the execution of such workflows requires consideration of not only individual task deadlines but also cost and occupation of resources [19]. Algorithms based on partial critical paths can be used to solve such problems [20, 21], applying meta-heuristic approaches, e.g. particle swarm optimisation [22]. When customising virtual infrastructures, a common approach will 1) select suitable virtual machines (VMs) based on specific task-VM performance metrics, 2) minimise communication costs between tasks by grouping tasks needing frequent communication in the same VM, and 3) refine the selection based on the calculation of new critical paths. Most current work focuses on guaranteeing a single deadline encompassing the entire application, e.g. the Critical Path-based Iterative (CPI) [23] and Complete Critical paths (CPIS) [24] algorithms. All these technologies have been widely investigated for applications modelled as directed acyclic graphs (DAGs) as DAG-based methods are popular for building data-flows for data-intensive applications.

Real-Time Modelling and Simulation. In data science, coupling different simulation models of individual systems can be performed to study the behaviours of complex systems, e.g. combining species distribution models with weather models to study how diseases are distributed via insects and species migration at different times. Simulating physical systems does not always require the simulation to run at wall clock rates [25], but executing such simulations on distributed infrastructure does impose requirements on managing the simulation times of different sub-components, e.g. to control the relationships among events and time [26].

Real-Time Computational Steering. Real-time steering of a computing system requires monitoring of the runtime status of both application and infrastructure. Infrastructure-level monitoring takes place at the network level and on computing and storage nodes [27]. Monitoring service quality of Cloud environments allows providers or users to evaluate compliance with SLAs [28]. At application level monitoring often requires embedded probes within application components [29]. Logging and provenance subsystems often capture the runtime status of the overall system as well [30]. To visualise the runtime status and to allow a user to make correct decisions regarding system control, different kinds of monitoring information together with the context of the system execution have to be harmonised based on the timestamp. Semantic technologies are often used to integrate such information and to offer query interfaces to link them [29]. Runtime steering of computing systems can take the form of adaptations of application logic at specific control points where the system actively provides time windows for users to intercede, or else the system can be interrupted by the user during execution [31]. The controllability of infrastructures e.g. dynamically configuring or scaling nodes [32], or controlling network flows [33], offer applications opportunities to refine the system performance.

Real-Time Data Acquisition. Acquiring real-time observations is important for many RIs. The quality of communication between sensors and data processing units is crucial for timely acquisition. Software-defined sensor networks can be used to optimise communication between sensors [34], as can applying edge computing solutions to tightly coupled sensors with data processing [35]. To make sensor data available to users in near real-time, partially automating data quality control and annotation is important [36], but currently, most data quality control is performed manually. Standardising this process and exploiting scalable virtualised infrastructure are recurring requirements for environmental RI [37].

Real-Time Data Transfer. Real-time data transfer between components occurs frequently within e-infrastructures. At the network level, real-time data protocols [38], multi-path TCP and other protocols are used to optimise data streaming throughput. SDN [50] technologies are used to adapt network flows between data nodes dynamically, and traffic programming models such as co-flow [39] are used to reschedule runtime data transfer. At the transfer service level, dynamic schedule data transfer workers are used in the LOBCDER service to handle the balance of downloads [40].

Infrastructure Provisioning for Time-Critical Applications. Fast provisioning of virtualised infrastructure opens the possibility of runtime adaptation to meet time-critical requirements. Optimising VM image size [41], directly forking runtime images from memory [42], or parallelising the provision procedure [43]. Using P2P or SND technology to optimise image transfer among data centres is also possible. Zhou et al. describe a transparent networked virtual infrastructure graph partitioning and parallel provisioning approach to map infrastructure across data centres [44].

Real-Time Service Level Agreement. Real-time support of virtualised infrastructure has attracted increasing interest [45]. SLAs for real-time applications and their negotiation at runtime will be crucial for supporting real-time applications in Cloud. Most

approaches are based on graph mapping using key quality parameters such as execution time; improving the mapping procedure can be done by parallelising the search procedure for matching resources and applications [46], pre-processing the resource information by clustering the resource information based on the SLA request, and multi-objective optimisation for searching out alternative solutions [47]. Rich contextual information and semantic annotation is another key issue influence the search quality [48].

3 Challenges for Time-Critical Applications on e-Infrastructure

In data science, the research data lifecycle is considered to be of primary importance, but at each stage of that lifecycle, we must also consider the lifecycle of the data pipelines or data processing workflows that are needed to support each stage. Given the increasing availability and adoption of virtualised e-infrastructure and Cloud services targeted towards RIs and the general research community, we are particularly interested in the life cycle for applications on virtual infrastructure (i.e. configurations of networked VMs upon which data processing workflows are deployed on behalf of researchers either for specialised tasks or as part of the general data lifecycle managed by RIs).

For static infrastructures, the development and configuration of a particular application (e.g. a data processing pipeline or workflow) can be adapted with respect to the hardware and host architecture known to the developers. This may still require considerable technical expertise of course, but can nonetheless be considered to at least represent a single initial investment to providing an efficient, performant technical solution.

In contrast, deploying application workflows on virtual infrastructures allows RIs to make use of commodity e-infrastructure resources as and when needed, rather than requiring an investment in dedicated hardware, and in principle offers the additional advantages of scalability and seamless migration which can to some extent be managed almost entirely by the e-infrastructure provider. It is difficult, however, to optimise generic virtual infrastructure for specific applications, and so difficult to guarantee a certain level of performance, which is particularly of concern for time-critical applications.

Figure 1 illustrates the lifecycle of application workflows on virtual infrastructure. Five main phases are identified:

1. Virtual infrastructure planning. Regarding the scheduling of application workflows onto a topology of virtual machines that ensure the availability and suitability of virtual resources at all stages of the workflow.
2. Virtual infrastructure optimisation. Regarding the iterative refinement of an infrastructure plan to meet all (or a maximal subset of) requirements for performance, reliability, quality of service, etc.
3. Virtual infrastructure provisioning. Regarding the actual provisioning of planned infrastructure across one or more data centres or Clouds in such a way as to create a network of resources that meet the control and data-flow requirements of a distributed application.
4. Software platform deployment. Regarding the actual deployment of application elements onto the provisioned infrastructure, as well as the initialisation and control of such elements at runtime.

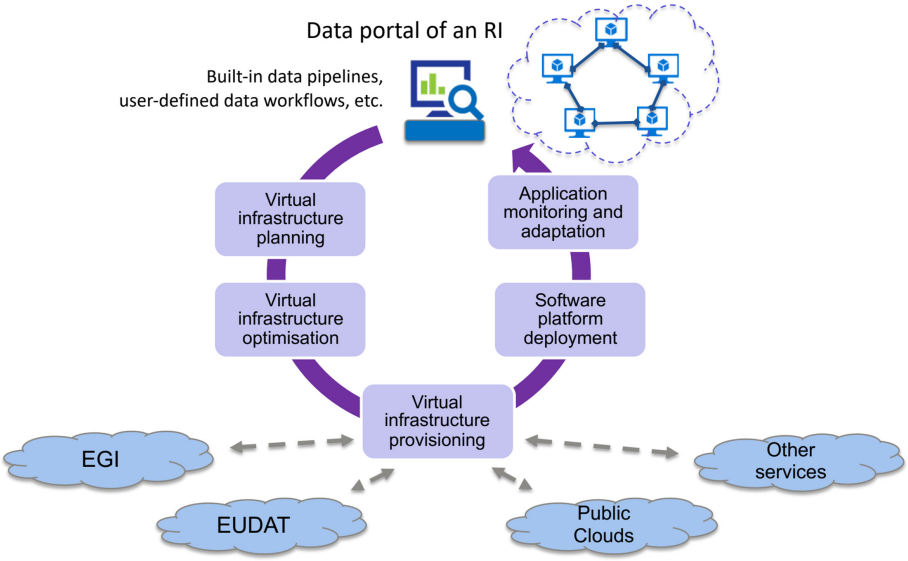

Fig. 1. The lifecycle of application workflows on virtual infrastructure.

5. Application monitoring and adaptation. Regarding the monitoring of a running application with respect to selected metrics necessary for evaluating the performance and liveness of the application, as well as the ability to intelligently take measures to improve and regain a desired quality of service, e.g. by automatically scaling or migrating application elements in the virtual infrastructure, or re-configuring components where practical.

While there exist a number of general solutions for managing each of these phases for the most common technologies for providing virtual infrastructure, or even subsets thereof, there is no single integrated solution for managing the entire lifecycle just described. Moreover, if we want to apply such a solution to time-critical applications, then it is necessary to address additional challenges:

- To meet time requirements for discovering and retrieving data from distributed access/storage services and virtual infrastructures provided by different RIs it is necessary to be able to define deadlines throughout the application deployment lifecycle both individually and collectively
- To develop a time-critical application, either the developer needs to be able to describe how constraints at application level propagate down to the level of restrictions on infrastructure and quality of service, or else the optimisation services developed for the infrastructure must be able to do that for the developer.
- During the execution of time-critical applications, data sources, software components, and the execution engines of some parts of the application may have to be handled by different underlying infrastructures, making it difficult to calculate and enforce quality of service constraints across the entire application/infrastructure stack.

To help address these concerns, we have developed the Dynamic Real-time Infrastructure Planner, which provides a set of services to optimise the automation from infrastructure customisation and provisioning to application deployment and runtime control.

4 Dynamic Real-Time Infrastructure Planner

The Dynamic Real-time Infrastructure Planner (DRIP) is a service suite for planning and provisioning networks of virtual machines and then deploying distributed applications across those networks, managing the virtual infrastructure during runtime based on certain time-critical constraints defined with the application workflow. The DRIP service provides an engine for automating all these procedures by making use of pluggable microservices for providing specific functionalities orchestrated via a single manager component behind a RESTful Web API for easy use and retrieval of results. This approach enables a more holistic approach to the optimisation of resources and meeting application-level constraints such as deadlines or SLAs. It also allows application developers to seamlessly plan a customised virtual infrastructure based on constraints on QoS, time-critical constraints or constraints on budget. Based on such a plan DRIP can provision a virtual infrastructure across several Cloud providers, and then be used for deploying application components, starting execution on-demand, and managing the runtime application deployment state. Therefore, DRIP is not bound to any particular application. Instead, it is flexible and can deploy a wide range of applications on top of a customised and heterogeneous virtual infrastructure composed of multiple Clouds providers to meet the application's constraints.

4.1 Architecture and Functional Components

The DRIP services include a number of components, interacting via an internal message brokering service orchestrated by a single manager. These components and their interaction are shown in Fig. 2.

All components of DRIP under the control of the DRIP manager are designed to be independently replaceable, to allow for improved or alternative implementations of e.g. the planner or the provisioner. Indeed, multiple versions of a component could coexist, allowing for greater flexibility or even simply to better balance the load of requests to DRIP. The types of service that can be included in DRIP, and their current implementations, are now detailed:

- **The DRIP manager** is a Web service that allows DRIP functions to be invoked by external clients. Each request is directed to the appropriate component by the manager, which coordinates the individual components and scales them up if necessary. Resource information, credentials, performance profiles and application workflows used by the manager and other components are all internally managed via the knowledge base as described below.
- **The planner** uses a partial critical path algorithm [1] optimised for workflows with multiple internal deadlines in order to produce efficient infrastructure topologies, selecting the most cost-effective virtual machines [20]. Multiple planner components

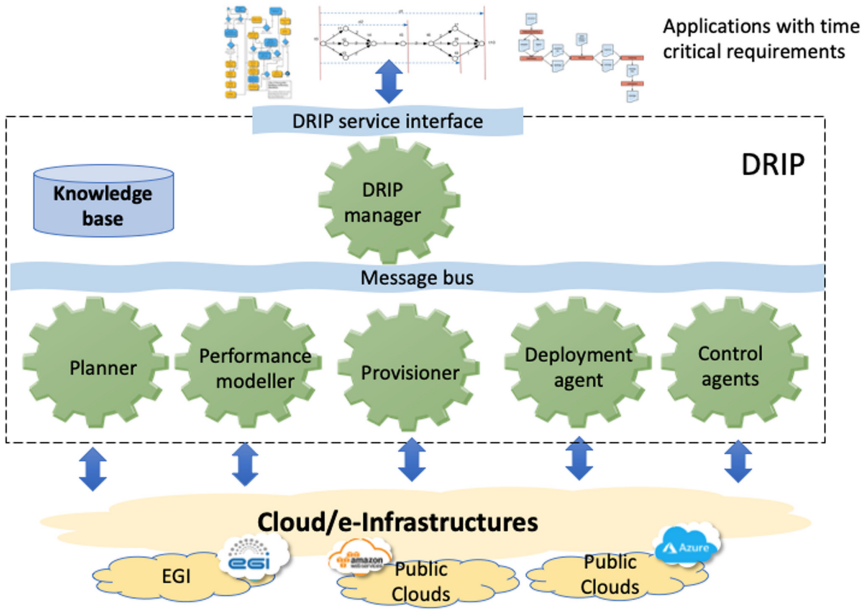

Fig. 2. How services provided by DRIP are invoked by RI services to provide downstream services to users.

can be attached to DRIP in order to manage different kinds of application workflow or infrastructure topology, taking of different technologies such as software-defined networking [49] to customise the network topology among VMs and optimally place network controllers for the networked VMs [50].

- **The performance modeler** is a tool which automates the execution of a given application on a virtual machine and collects the performance information of the application. In this way, it can profile the performance characteristics of specific applications. The output will be used by the planner to select virtual machines during its planning procedure.
- **The provisioner** is responsible for automating the provisioning of infrastructure plans produced by the planner(s) onto the underlying Cloud or e-infrastructure. The current provisioner can decompose the infrastructure description and provision it across multiple data centres (possibly from different providers) with transparent network configuration [44].
- **The deployment agent** installs application components onto provisioned infrastructure. The current deployment agent is able to schedule the deployment sequence based on network bottlenecks, and maximise the fulfilment of deployment deadlines for all the Cloud providers currently supported by the default DRIP provisioner [51].
- **Infrastructure control agents** provide sets of APIs that DRIP can then provide to applications to control the scaling of containers or VMs and for adapting network flows or to use itself in conjunction with a monitoring framework to automatically maintain the quality of service of the deployed application.

- **A DRIP knowledge base** is employed by DRIP for storing information about user credentials, the types of the resource offered by Clouds or e-infrastructure, and other useful data that the DRIP manager or any other component can retrieve or contribute to.
- **The message bus** connects all the components in the DRIP to enable the communication among them.
- **The service interface** provides a standardised API to the application developers, or software clients (e.g. data portal or workflow system) to invoke the function.

4.2 Implementation Details

DRIP was developed in the context of EU H2020 projects SWITCH[1] (as part of a workbench for time-critical, self-adaptive applications on Cloud) and ENVRIPLUS[2] (to provide e-infrastructure optimisation services for scientific workflows).

The prototype of DRIP adopts industrial and community standards. The infrastructure planner uses the TOSCA specification[3] to get descriptions of applications and their constraints. The infrastructure provisioner uses OCCI as its default provisioning interface, and currently supports the Amazon EC2, EGI FedCloud and ExoGen[4] Clouds. Since DRIP relies on multiple Cloud providers it offers a best-effort approach for the provision, stability and performance of the underlying virtual infrastructure. However, using performance and reliability models for each provider and each region, DRIP is able to provide an optimal, stable and responsive vitalised infrastructure for time-critical applications [52]. The deployment agent can deploy overlay Docker clusters such as Docker Swarm or Kubernetes. It may also deploy any type of customised application based on Ansible playbooks [53]. The infrastructure control agents are a set of APIs that DRIP provides to applications to allow for scaling of containers or VMs and for adapting network flows. The manager provides a RESTful interface to allow integrated interaction with all components and uses RabbitMQ as its internal message broker to direct requests appropriately. All DRIP software is available via open source repository[5] under the Apache-2.0 license.

4.3 How DRIP Works

Figure 3 demonstrates how DRIP works. We choose an example of disaster early warning, in which a legacy application needs to be migrated in Cloud environments using DRIP.

1. The application developer needs to identify the application components to be deployed in Cloud, describe the dependencies between components, and specify the time-critical constraints (the deadlines between specific application tasks), as shown in the step in Fig. 3.

[1] www.switchproject.eu.
[2] www.envriplus.eu.
[3] https://www.oasis-open.org/committees/tosca/.
[4] http://www.exogeni.net/.
[5] https://github.com/QCAPI-DRIP/.

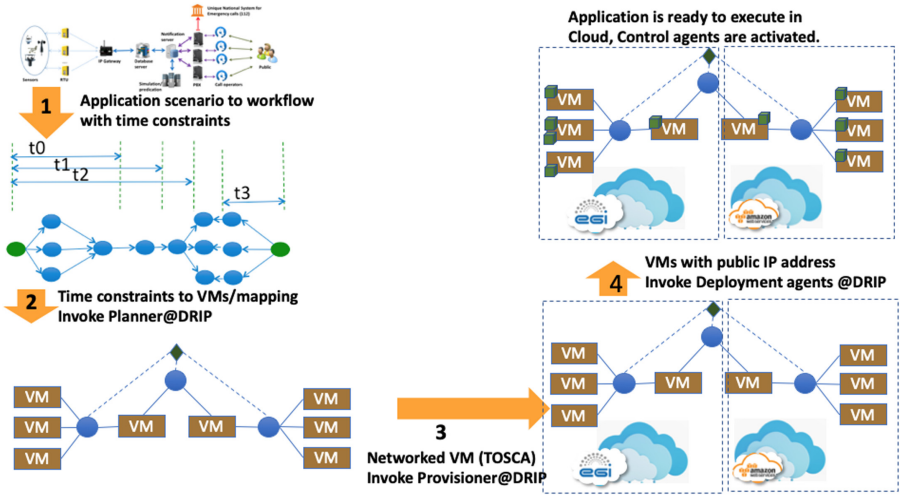

Fig. 3. A conceptual demonstrator of DRIP.

2. The DRIP planner will plan the virtual infrastructure for the application based on the description provided by step 1. Currently, the description structure is based on the template derived from TOSCA standard. The planned virtual infrastructure including a) a set of virtual machines (VM), with specific size of CPU, memory and storage, b) the network topology among VMs, and c) the controller for the network. The output of this step will use the same TOSCA format, but with concrete information of the virtual infrastructure.
3. The DRIP provisioner will continue with the step 2; it will select specific data centres or Cloud providers to provision the planned VM. The provisioner is able to parallelise the provisioning procedure and automate the network configuration among VMs. The step 3 modifies the TOSCA description from step 2 with concrete public IP address. After step 3, all the VM will be remotely accessible.
4. The DRIP deployment agent will use the provisioned virtual infrastructure to deploy the application components identified in step 1. After the fourth step, the application is ready to execute.

The application topology is currently described using TOSCA and must be part of the request made to DRIP. When a planning request comes, the manager will direct the request to the *infrastructure planner* to generate a plan, which can be sent back to the user for further confirmation. If the constraints cannot be satisfied the planner informs the user that a plan cannot be generated. The DRIP manager stores the necessary Cloud credentials on behalf of the user. The *provisioning agent* can provision the virtual infrastructure via interfaces offered by the Cloud providers. Once this has finished, the deployment agent will deploy all necessary components onto the provisioned infrastructure from designated repositories and set up the control interfaces needed for runtime control of application and infrastructure. Figure 4 shows a detailed sequence diagram.

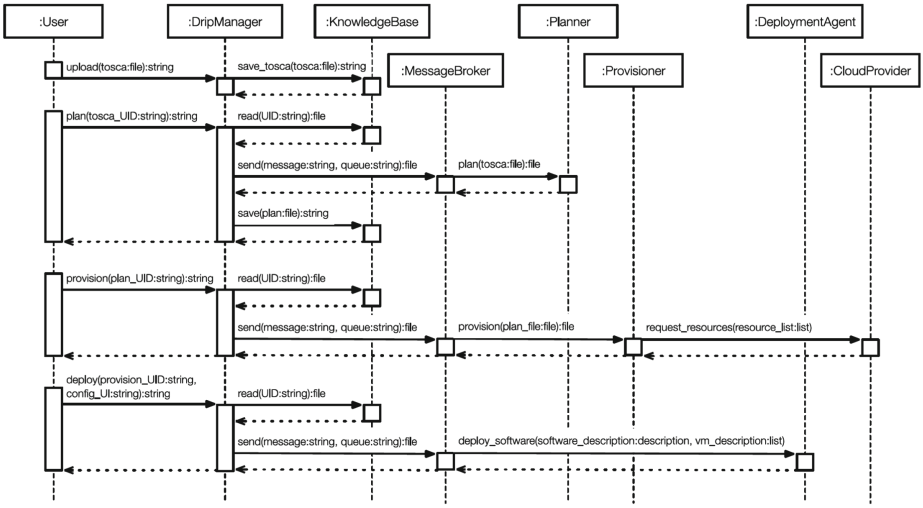

Fig. 4. A detailed sequence diagram of DRIP.

4.4 Future Work: Workflow Reproducibility

One of the key objectives in environmental research facilitated by the contextualisation of processes and actors in research infrastructure is that of workflow reproducibility, whereby the provenance of experiments conducted by scientists using the infrastructure is recorded so as to allow the experiments to be re-executed with minimal difficulty. Recording provenance in various workflow systems has been explored before, but more effort is needed to adopt standard provenance frameworks with standard vocabulary (e.g. based on PROV) that can be implemented independently by different workflow systems and data processing platforms. The use of ontologies for verification and validation of workflows has already been explored. For example, Miksa and Rauber [54] provide such ontologies and accompanying tool support for just this kind of activity.

Virtualisation, of both computing resources and the interstitial network, provide a good basis for fluid intelligent infrastructure, providing flexible logical networks of computing and data nodes on demand that can be optimised based on some co-extant knowledge fabric.

5 Summary

The ability to comprehend, facilitate and augment how researchers use research infrastructure to support data-driven science is crucial, but also extremely challenging to acquire given the proliferation of competing systems and standards in the world computational landscape. This problem is not limited to the logical aggregation of research data products for the purposes of cataloguing and access, but also extends to the use of information to promote the efficient operation of research infrastructure and the effective use of ‘e-infrastructure’ including the compute, data and network resources provided via initiatives such as EGI and PRACE. The seamless, easy access to distributed data and the

use of community computing platforms requires significant automation via brokering agents and other software services operating over the baseline infrastructure, which in turn requires substantial knowledge infrastructure to support the planning and optimal execution of a host of different concurrent operational and data-driven workflows.

Most scientific investigations follow a clear workflow, and for data-driven or otherwise computational workflows, different processes can be linked together into a single distributed application managed by a single system. There have been a number of scientific workflow management systems developed in order to address the manifold challenges raised by modern scientific computing in the last two decades, all with different characteristics and target applications and such systems have been made use of in many different scientific disciplines. The composition and execution of workflows require careful consideration of how to manage the communication between processing elements and maintain sufficient quality of service across the entire workflow.

Acknowledgements. This work was supported by the European Union's Horizon 2020 research and innovation programme via the ENVRIplus project under grant agreement No 654182. This work was also partially supported by the European Union's Horizon 2020 research and innovation programme via the SWITCH project under grant agreement No 643963, and ARTICONF project under grant agreement No 825134.

References

1. Koulouzis, S., et al.: Time-critical data management in clouds: challenges and a Dynamic Real Time Infrastructure Planner (DRIP) solution. Concurr. Comput. Pract. Exp. e5269 (2019). https://doi.org/10.1002/cpe.5269
2. Foster, I., Kesselman, C.: Scaling system-level science: scientific exploration and it implications. Computer **39**(11), 31–39 (2006)
3. Štefanič, P., et al.: SWITCH workbench: a novel approach for the development and deployment of time-critical microservice-based cloud-native applications. Future Gener. Comput. Syst. **99**, 197–212 (2019). https://doi.org/10.1016/j.future.2019.04.008
4. Koulouzis, S., et al.: Distributed data management service for 0 applications. IEEE Internet Comput. **20**(2), 34–41 (2015)
5. Zhao, Z., Belloum, A., De Laat, C., Adriaans, P., Hertzberger, B.: Using Jade agent framework to prototype an e-Science workflow bus. In: Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007), Rio de Janeiro, Brazil, pp. 655–660. IEEE (2007). https://doi.org/10.1109/CCGRID.2007.120
6. Laplante, P.A., Ovaska, S.J.: Real-Time Systems Design and Analysis: Tools for the Practitioner. Wiley, Hoboken (2011)
7. Poslad, S., Middleton, S.E., Chaves, F., Tao, R., Necmioglu, O., Bügel, U.: A semantic IoT early warning system for natural environment crisis management. IEEE Trans. Emerg. Top. Comput. **3**(2), 246–257 (2015)
8. Hu, Y., Zhou, H., de Laat, C., Zhao, Z.: Concurrent container scheduling on heterogeneous clusters with multi-resource constraints. Future Gener. Comput. Syst. **102**, 562–573 (2020). https://doi.org/10.1016/j.future.2019.08.025
9. Evans, K., et al.: Dynamically reconfigurable workflows for time-critical applications. In: Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science - WORKS 2015, Austin, Texas, pp. 1–10. ACM Press (2015). https://doi.org/10.1145/2822332.2822339

10. Dashti, S.E., Rahmani, A.M.: Dynamic VMS placement for energy efficiency by PSO in Cloud computing. J. Exp. Theor. Artif. Intell. **28**(1–2), 97–112 (2016)
11. Usmani, Z., Singh, S.: A survey of virtual machine placement techniques in a cloud data center. Procedia Comput. Sci. **78**, 491–498 (2016). https://doi.org/10.1016/j.procs.2016.02.093
12. Gao, Y., Wang, Y., Gupta, S.K., Pedram, M.: An energy and deadline aware resource provisioning, scheduling and optimization framework for Cloud systems. In: Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, p. 31 (2013)
13. Li, D., Chen, C., Guan, J., Zhang, Y., Zhu, J., Yu, R.: DCloud: deadline-aware resource allocation for Cloud computing jobs. IEEE Trans. Parallel Distrib. Syst. **27**(8), 2248–2260 (2016)
14. Shi, J., Luo, J., Dong, F., Zhang, J.: A budget and deadline aware scientific workflow resource provisioning and scheduling mechanism for Cloud. In: Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 672–677 (2014)
15. Downey, D., Dumais, S.T., Horvitz, E.: Models of searching and browsing: languages, studies, and application. IJCAI **7**, 2740–2747 (2007)
16. Agichtein, E., Brill, E., Dumais, S.: Improving web search ranking by incorporating user behavior information. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 19–26 (2006)
17. Mishra, N., White, R.W., Ieong, S., Horvitz, E.: Time-critical search. In: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 747–756 (2014)
18. Ingwersen, P., Järvelin, K.: The Turn: Integration of Information Seeking and Retrieval in Context, vol. 18. Springer, Heidelberg (2006). https://doi.org/10.1007/1-4020-3851-8
19. Yu, J., Buyya, R., Tham, C.K.: Cost-based scheduling of scientific workflow applications on utility grids. In: e-Science and Grid Computing (2005)
20. Wang, J., et al.: Planning virtual infrastructures for time critical applications with multiple deadline constraints. Future Gener. Comput. Syst. **75**, 365–375 (2017)
21. Abrishami, S., Naghibzadeh, M., Epema, D.H.: Deadline-constrained workflow scheduling algorithms for infrastructure as a service Clouds. Future Gener. Comput. Syst. **29**(1), 158–169 (2013)
22. Rodriguez, M.A., Buyya, R.: Deadline based resource provisioning and scheduling algorithm for scientific workflows on Clouds. IEEE Trans. Cloud Comput. **2**(2), 222–235 (2014)
23. Cai, Z., Li, X., Gupta, J.: Heuristics for provisioning services to workflows in XaaS Clouds. IEEE Trans. Serv. Comput. **9**(2), 250–263 (2016)
24. Taal, A., Wang, J., de Laat, C., Zhao, Z.: Profiling the scheduling decisions for handling critical paths in deadline-constrained cloud workflows. Future Gener. Comput. Syst. **100**, 237–249 (2019). https://doi.org/10.1016/j.future.2019.05.002
25. Alawneh, S., Dragt, R., Peters, D., Daley, C., Bruneau, S.: Hyper-real-time ice simulation and modeling using GPGPU. IEEE Trans. Comput. **64**(12), 3475–3487 (2015)
26. Zhao, Z., Albada, D.V., Sloot, P.: Agent-based flow control for HLA components. Simulation **81**(7), 487–501 (2005)
27. Bulut, A., Koudas, N., Meka, A., Singh, A.K., Srivastava, D.: Optimization techniques for reactive network monitoring. IEEE Trans. Knowl. Data Eng. **21**(9), 1343–1357 (2009)
28. Zhou, H., Ouyang, X., Su, J., Laat, C., Zhao, Z.: Enforcing trustworthy cloud SLA with witnesses: a game theory–based model using smart contracts. Concurr. Comput. Pract. Exp. (2019). https://doi.org/10.1002/cpe.5511
29. Taherizadeh, S., Jones, A.C., Taylor, I., Zhao, Z., Stankovski, V.: Monitoring self-adaptive applications within edge computing frameworks: a state-of-the-art review. J. Syst. Softw. **136**, 19–38 (2018). https://doi.org/10.1016/j.jss.2017.10.033

30. Chebotko, A., Lu, S., Chang, S., Fotouhi, F., Yang, P.: Secure abstraction views for scientific workflow provenance querying. IEEE Trans. Serv. Comput. **3**(4), 322–337 (2010)
31. Kritikakou, A., Pagetti, C., Baldellon, O., Roy, M., Rochange, C.: Run-time control to increase task parallelism in mixed-critical systems. Real-Time Syst. (ECRTS) **2014**(26), 119–128 (2014)
32. Serrano, N., Gallardo, G., Hernantes, J.: Infrastructure as a service and Cloud technol-ogies. IEEE Softw. **32**(2), 30–36 (2015)
33. Nussbaum, A., Choodamani, S.M.C., Schwan, K.: ObsCon: Integrated monitoring and control for parallel, real-time applications. Clust. Comput. (CLUSTER) **2015**, 474–477 (2015)
34. Anadiotis, A.C.G., Galluccio, L., Milardo, S., Morabito, G., Palazzo, S.: Towards a software-defined network operating system for the IoT. In: 2015 Internet of Things (WF-IoT), pp. 579–584 (2015)
35. Papageorgiou, A., Cheng, B., Kovacs, E.: Real-time data reduction at the network edge of Internet-of-Things systems. Netw. Serv. Manag. (CNSM) **2015**(11), 284–291 (2015)
36. Hu, S., et al.: Data acquisition for real-time decision-making under freshness constraints. In: 2015 IEEE Real-Time Systems Symposium, pp. 185–194. IEEE (2015)
37. Laranjeiro, N., Soydemir, S.N., Bernardino, J.: A survey on data quality: classifying poor data. Dependable Comput. (PRDC) **2015**, 179–188 (2015)
38. Shamani, M.J., Zhu, W., Naghshin, V.: TMPTCP: tailless multi-path TCP. In: Broadband and Wireless Computing, Communication and Applications (BWCCA), vol. 2015, no. 10, pp. 325–332 (2015)
39. Fu, Z., Song, T., Wang, S., Wang, F., Qi, Z.: Seagull–a real-time coflow scheduling system. Cyber Secur. Cloud Comput. (CSCloud) **2015**, 540–545 (2015)
40. Koulouzis, S., Belloum, A.S.Z., Bubak, M.T., Zhao, Z., Živković, M., de Laat, C.T.A.M.: SDN-aware federation of distributed data. Future Gener. Comput. Syst. **56**, 64–76 (2016). https://doi.org/10.1016/j.future.2015.09.032
41. Tang, C.: FVD: a high-performance virtual machine image format for Cloud. In: USENIX Annual Technical Conference, vol. 2 (2011)
42. Lagar-Cavilla, H.A., et al.: SnowFlock: rapid virtual machine cloning for Cloud computing. In: Proceedings of the 4th ACM European Conference on Computer Systems, pp. 1–12. ACM (2009)
43. Zhou, H., et al.: CloudsStorm: a framework for seamlessly programming and controlling virtual infrastructure functions during the DevOps lifecycle of cloud applications. Softw.: Pract. Exp. **49**, 1421–1447 (2019). https://doi.org/10.1002/spe.2741
44. Zhou, H., Hu, Y., Wang, J., Martin, P., Laat, C.D., Zhao, Z.: Fast and dynamic resource provisioning for quality critical Cloud applications. In: 2016 IEEE 19th International Symposium on Real-Time Distributed Computing (ISORC), pp. 92–99 (2016)
45. Wartel, R., et al.: Image distribution mechanisms in large scale Cloud providers. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science, pp. 112–117. IEEE (2010)
46. Müller, C., et al.: Comprehensive explanation of SLA violations at runtime. IEEE Trans. Serv. Comput. **7**(2), 168–183 (2013)
47. Casale, G., et al.: Current and future challenges of software engineering for services and applications. Procedia Comput. Sci. **97**, 34–42 (2016). https://doi.org/10.1016/j.procs.2016.08.278
48. Liao, X., Zhao, Z.: Unsupervised approaches for textual semantic annotation, a survey. ACM Comput. Surv. **52**, 1–45 (2019). https://doi.org/10.1145/3324473
49. Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)

50. Wang, J., de Laat, C., Zhao, Z.: QoS-aware virtual SDN network planning. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, pp. 644–647. IEEE (2017). https://doi.org/10.23919/INM.2017.7987350
51. Hu, Y., et al.: Deadline-aware deployment for time critical applications in clouds. In: Rivera, F.F., Pena, T.F., Cabaleiro, J.C. (eds.) Euro-Par 2017. LNCS, vol. 10417, pp. 345–357. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64203-1_25
52. Martin, P., et al.: Information modelling and semantic linking for a software workbench for interactive, time critical and self-adaptive Cloud applications. In: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 127–132. IEEE (2016)
53. Miller, M.A., Pfeiffer, W., Schwartz, T.: The CIPRES science gateway: enabling high-impact science for phylogenetics researchers with limited resources. In: Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond, p. 39 (2012)
54. Mayer, R., Miksa, T., Rauber, A.: Ontologies for describing the context of scientific experiment processes. In: 2014 IEEE 10th International Conference on e-Science, vol. 1, pp. 153–160. IEEE (2014)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.