Computing and Informatics, Vol. 39, 2020, 860–880, doi: 10.31577/cai_2020_4_860

TOWARDS EXASCALE COMPUTING ARCHITECTURE AND ITS PROTOTYPE: SERVICES AND INFRASTRUCTURE

Jan Meizner, Piotr Nowakowski

ACC Cyfronet, AGH University of Science and Technology
Krakow, Poland
&
Sano Centre for Computational Medicine
Krakow, Poland
e-mail: {j.meizner, p.nowakowski}@cyfronet.pl

Jan Kapala, Patryk Wojtowicz

ACC Cyfronet, AGH University of Science and Technology
Krakow, Poland
e-mail: {j.kapala, p.wojtowicz}@cyfronet.pl

Marian Bubak

ACC Cyfronet, AGH University of Science and Technology
Krakow, Poland
&
Sano Centre for Computational Medicine
Krakow, Poland
&
Department of Computer Science, AGH University of Science and Technology
Krakow, Poland
e-mail: bubak@agh.edu.pl

Viet Tran, Martin Bobák

Institute of Informatics, Slovak Academy of Sciences
Dúbravská cesta 9, 845 07 Bratislava, Slovakia
e-mail: {viet.tran, martin.bobak}@savba.sk

Maximilian Höb

Munich Network Management Team (MNM-Team)
Ludwig-Maximilians Universität, Munich, Germany
e-mail: hoeb@nm.ifi.lmu.de

Abstract. This paper presents the design and implementation of a scalable compute platform for processing large data sets in the scope of the EU H2020 project PROCESS. We are presenting requirements of the platform, related works, infrastructure with focus on the compute components and finally results of our work.

Keywords: Exascale computing, large data sets, HPC, cloud computing

1 INTRODUCTION

Despite continuous increases in the computing power of HPC systems, even the largest system on the latest edition of the TOP500 list [20] still provides less then 0.5 exaFLOP (in full precision). At the outset of the PROCESS project this figure was even lower (about 0.1 exaFLOP) [19]. It therefore seemed clear to us that – at least initially – reaching exascale computing capabilities would require a combination of multiple HPC systems. As we can see, this premise holds true to this day, and even when we finally break the exaFLOP barrier, the number of such systems will be highly limited – thus, combining the power of many sites would remain prudent in many cases. It is also important to notice that it is not always possible to quickly move data between computing sites; hence, the ability to bring computations to data remains an important issue.

Running tasks in such combined systems is fraught with multiple challenges. The basic one relates to the heterogeneity of the environment, as each system is operated by a different entity. This heterogeneity may involve access mechanisms (protocols, credential types) as well as software installed on clusters (OS type and version, queuing system). Additionally, each cluster usually runs its workloads in internal private LANs which do not allow inbound connections from the Internet (due to mechanisms such as NAT). This, in turn, places additional restrictions upon the designed infrastructure, such as the lack of direct P2P communication between jobs running on different clusters. One of the main goals of the project was to provide a mechanism which would allow running jobs on multiple sites and move data around freely.

In order to manage such heterogeneous systems, it is also crucial [14] to provide a highly scalable platform, able to process large quantities of data (although issues related to maintenance of such data have been described in a separate paper [23]).

Finally, we need to take into account that some sites in a distributed computational environment may be based on traditional HPC paradigms (such as the Prometheus cluster at Cyfronet, or CoolMUC and SuperMUC-NG at Leibniz-Rechenzentrum) while others may be Cloud-based (in the presented case, this includes the Institute of Informatics of the Slovak Academy of Sciences and other compatible private and community clouds such as those provided in the scope of the European Open Science Cloud; EOSC). This further elevates the level of heterogeneity [8].

2 A SHORT OVERVIEW OF RESEARCH ON EXASCALE SYSTEM

A systematic analysis of needs and profits of exascale computing systems was initiated by the U.S. DOE in a series of workshops on scientific grand challenges in 2007 [22]. Workshops were focused on the grand challenges of specific scientific domains and on the role for scientific computing in addressing those challenges. A summary of discussions during those scientific meeting is presented in [6]. It also gives an initial set of requirements for exascale systems, however, it is mostly concentrated on the computing aspect leaving aside big data aspects. Dongarra et al. in [9] analysed the approaches used to implement peta- and exa-scale compting pointing out that completely uncoordinated development model will not provide the software needed to support the unprecedented parallelism required for peta/exascale computation and presented the idea of the International Exascale Software Project. In this paper, the development of appropriate open-source software tools was clearly indicated as an important factor in quest for high performance and productivity. The focus was mostly on proper usage of new processors architectures of large parallel computers.

The need for considering together exascale computing and big data was presented by Reed and Dongarra in [18]; after overview of main area for exascale computing like biology, particle physics, climate science, cosmology, astrophysics, material science, they have analysed technical challenges in advanced computing including software elaboration, and overview a number of national and international projects. The ideas presented in this paper are reflected in the Big Data and Extreme Scale computing Project [1] and in the ECP – Exascale Computing Project [2]. On the basis of the achievements of these projects, an international group of scientists [5] elaborated a set of recommendations for successful approaches for software ecosystem convergence in big, logically centralized facilities that execute large-scale simulations and models and/or perform large-scale data analytics. In our research, we are going to take them into account having in mind that our solutions should be appropriate for exascale computing and data analytics on distributed systems.

The recently published paper [16] presents results of a study of possible convergence of big data coming from distributed scientific instruments and sensors and high performance computing appropriate for coming era of next-generation data centric computing. The study was performed in the framework of the EU project SAGE. The authors have elaborated an advanced storage system which has been

implemented and installed at the Jülich Supercomputing Center. It is a very interesting and important solution, however, it does not address usage of distributed computing resources.

3 REQUIREMENTS

The ultimate goal of PROCESS is to provide a versatile solution well suited for a wide range of use cases from multiple domains, both scientific and business-oriented. To achieve this goal we had to collect a set of requirements from each use case, and combine them into an integrated set applicable to all of them.

3.1 Hardware Requirements

The hardware resources are basis for any IT system. In this subsection we are presenting the set of hardware-related requirements such as providing sufficient access to: HPC resources, Cloud resources, accelerated computing resources (including the GPGPUS), external infrastructure reachable via API, data storage on the order of 1 PB (distributed).

3.2 Software Requirements

As hardware alone is not sufficient to provide the services sought by scientists, our platform also has to support a wide range of software tools. The common tools for machine learning and deep learning are essential for wide range of medical applications, but also others such as business oriented. Python development environment with support for Jupyter notebooks allows simplification of the Use Case codes developments. Apache Spark, Hadoop and HBase frameworks are necessary to process the big data sets. Support for containers such as Docker and Singularity allows packaging of the codes alongside required dependences for streamlined deployment. Secure access to and extraction from external data resources is of course also crucial for any use case. Finally, we need to provide appropriate support for programming languages and tools needed by the provided use cases' codes such as: Java environment, Grid support, Matlab environment, Large-scale modelling, Predictive analytic methods, Probabilistic risk calculation tools.

3.3 Execution Models

We have analyzed all these requirements and in our conclusions got a set of specific Execution Models, namely: Deep Learning, Exascale Data Management, Exascale Data Extraction, Probabilistic Analysis, Calibration and finally the Pre- and Post-Processing.

All those models had to be reflected in the the PROCESS system architecture – both general as well as Compute parts specific.

4 OVERVIEW OF COMPUTING COMPONENTS

While preparing such a complex system we have carefully taken into account the current state of the art in various respects, relevant to the integrated system and its computing components in particular. This analysis is presented in the following subsections.

4.1 Interactive Execution Environment

Several solutions already exist to support interactive execution of large-scale computations on hybrid underlying infrastructures. This subsection provides a basic description of the available mechanisms and tools which support creation and sharing of executable documents for data analysis. In this section we also provide a brief introduction to scripting notebooks, in particular their integration with HPC infrastructures in order to support building extreme large computing services as well as their extensions mechanisms needed to add support specific to the PROCESS project. We also present the results of comparison of their functionality. This section partially extends the work submitted to KUKDM 2018 conference [13].

During our studies we have analysed multiple notebook-based solutions, a summary of which is presented in Table 1.

Name	Large Data Sets	Infrastructure
R Notebook	Using custom libraries (e.g. for Apache SPARK)	Using custom libraries (e.g. communicating with HPC queuing systems)
DataBricks	The whole platform is based on Apache SPARK	Available only on AWS or Azure
Beaker	Using additional custom libraries	No specific support for HPC; docker version available
Jupyter	Using additional custom libraries	No mature solution for HPC; docker version available
Cloud Datalab	Support for Google data services (e.g. BigQuery, Cloud Machine Learning Engine, etc.)	Only GCP
Zeppelin	Native support for Apache Spark	Possible to run on HPC using connection to YARN cluster

Table 1. Interactive execution environment comparison

Although there exist many interactive execution environments that could be considered for extension to match PROCESS requirements, many also have important drawbacks. DataBricks and Cloud Datalab require to be run on specific cloud resources. Zeppelin and DataBricks are based on the Apache SPARK solution which potentially limits their usage to that platform. RNotebooks seems to be promising,

however some important features are only available with the commercial version of Rstudio. BeakerX (the successor to Beaker) and Cloud Data are actually based on the Jupyter solution, which appears to be the most popular base for building such environments.

The goal of the Interactive Execution Environment is to bridge the gap between users of computational services (who are not expected to be familiar with the complexity of developing and executing extreme large scale computational tasks with the use of modern HPC infrastructures) and the underlying hardware resources. Accordingly, the IEE is envisioned as an interface layer where applications can be accessed and their results browsed in a coherent manner by domain scientists taking part in the PROCESS project and beyond.

The following properties are regarded as particularly desirable:

- A means of implementing the "focus on services and forget about infrastructures" concept;
- Providing two ways of accessing the underlying computational resources: through a user-friendly GUI and programmatically, via a dedicated RESTful API;
- Embeddability in an external environment (such as Jupyter) via API integration;
- Interfacing computational clouds and traditional HPC (batch job submission) and public cloud access libraries, as appropriate.

The features which need to be provided by the environment are as follows:

- Deployment of computational tasks on the available resources,
- Infrastructure monitoring services,
- User-friendly access to PROCESS datasets,
- Security management (users, groups, roles),
- Administrative services (billing and logging),
- Integration with external tools via standardized APIs.

Following discussions with use case developers and the project's architecture team, the following tools, described further in this paper, have been identified as usable in the context of the PROCESS project – in addition to the previously discussed notebook solutions, which can function as an embedded feature in a comprehensive GUI.

4.2 EurValve Model Execution Environment

The EurValve [7] Model Execution Environment (shown in Figure 1) is an execution environment for data processing pipelines. Originally conceived in the context of the EurValve project, the goal was to develop a decision support system for procedures related to heart valve abnormalities, enabling clinicians to decide upon the optimal

course of action for any specific patient (i.e. choose between medication/surgery and advise on the possible strategies and outcomes of the latter). While the aforementioned DSS does not, by itself, involve large-scale processing, it is based on a knowledge base whose production is one of the principal goals of EurValve. Assembling this knowledge base calls for processing a vast array of patient data (for both prospective and retrospective patients) through the use of dedicated pipelines, consisting of multiple services and making use of various types of supercomputing infrastructures (both traditional batch HPC and cloud models).

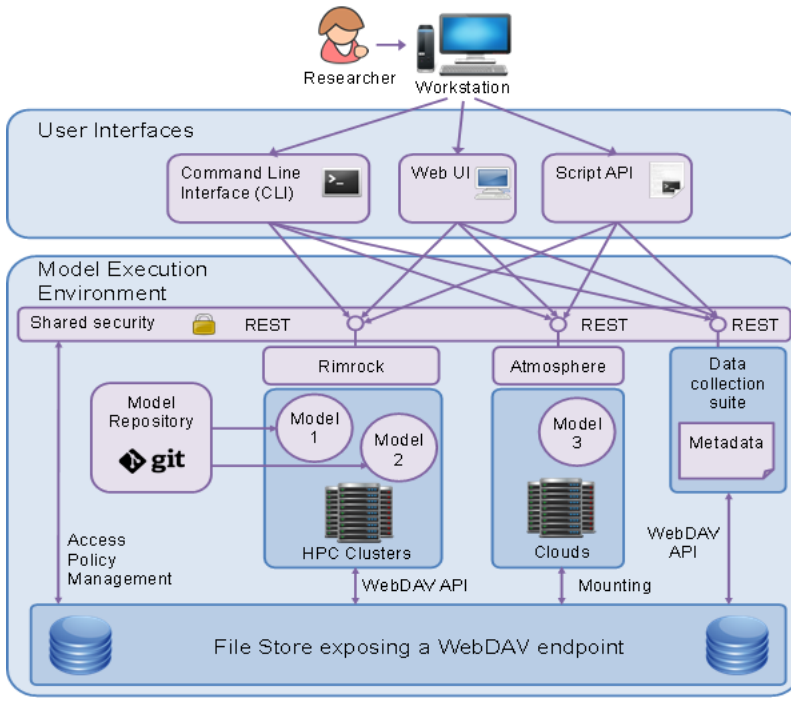

Figure 1. EurValve Model Execution Environment (MEE) architecture

4.3 Rimrock Execution Environment

Rimrock stands for Robust Remote Process Controller Controller [4] shown in Figure 2 is a service that simplifies interaction with remote HPC servers. It can execute applications in batch mode or start an interactive application, where output can be fetched online and new input sent using a simple REST interface. What is more, by using a dedicated REST interface users are able to submit new jobs to the infrastructure. This solution would support efficient creation and sharing of executable documents for analysis of heterogeneous research datasets. RIMROCK is

currently actively used in production in the Polish nationwide HPC infrastructure called PLGrid [17].

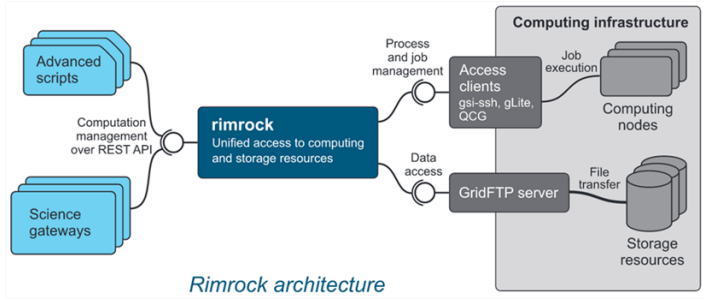

Figure 2. Rimrock architecture

4.4 Atmosphere Cloud Platform

Atmosphere [3], shown in Figure 3, is a hybrid cloud environment facilitating development and sharing of computational services wrapped as cloud virtual machines, with access to external data sources. Atmosphere supports the entire cloud service development lifecycle and provides a set of pluggable interfaces which can be included in portals and web applications. It is compatible with a wide range cloud middleware packages from open-source and commercial vendors, and provides interfaces to both public and private cloud resources in the form of a technology-agnostic UI and RESTful APIs.

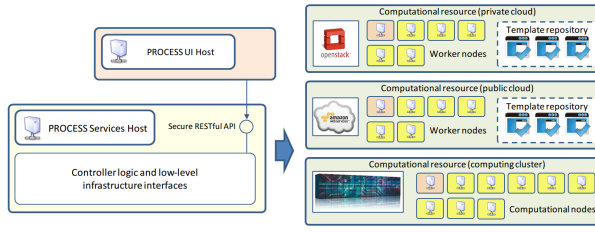

Figure 3. The architecture of atmosphere

4.5 Benchmarking and Monitoring Services

We have analyzed multiple monitoring solutions including Nagios, Zabbix, Icinga, Munin, Cacti, Ganglia, Collectd, Elastic Stack, Grafana, NFDUMP with NfSen and ntopng.

While detailed analysis is out of scope of this paper, we initially identified three candidates for this task: Zabbix, Nagios and Icinga, as they are all mature solutions and possess the required properties. Other available solutions, while usable, would only address a subset of the presented requirements; consequently, their usage would require integration of multiple distinct components, with its attendant impact on the stability of the integrated solution deployed in PROCESS.

Upon further analysis we finally chose Zabbix as the best solution (shown in Figure 4). Icinga 2 (current version of Icinga) was rejected due to not being as mature and well-established as either Zabbix or Nagios. As for Nagios Core, it offers great alerting capabilities, however it does not provide the same level of support for online resource monitoring, and its configuration is based on text files, requiring more complex integration. In contrast, Zabbix's configuration is based on an RDBMS (like MariaDB/MySQL) and could be modified via an API. With over 19 years of history and constant development/testing, as well as a wide range of users including large corporations representing a wide spectrum of domains such as banking/finance, health, IT and telecommunication, Zabbix is regarded as a mature solution.

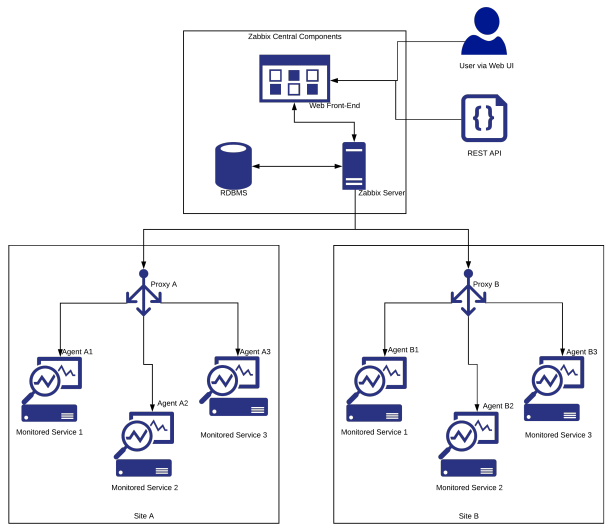

Figure 4. Monitoring architecture

5 CONCEPT OF COMPUTING ARCHITECTURE

The overall architecture of the PROCESS Platform [15] is presented in a separate paper [24]. In this section we focus solely on the computing components and glue code connecting them to one another as well as to external services.

While this aspect represents only part of the bigger platform, due to the inherent complexity of an exascale-capable system we had to design a solution composed of multiple layers, namely:

- API for third-party services (like external Portals),
- Web UI for domain scientists (to configure pipelines),
- Connectors for HPC and Cloud backend APIs,
- The aforementioned backend APIs themselves (Rimrock [4] and Cloudify [10]),
- Containers suitable for specific HPC/Cloud resources [12],
- Underlying HPC and Cloud e-Infrastructures.

6 BUILDING THE COMPUTING PLATFORM

6.1 IEE

The Interactive Execution Environment (IEE) is a platform which enables execution of HPC applications – including the PROCESS pilot use cases – in a heterogeneous infrastructure which comprises multiple computing sites based on various computing paradigms (such as "classic" batch-oriented HPC, interactive cloud computing and more). The basic architecture of the environment is schematically depicted in Figure 5. The IEE user interface serves as the entry point for the platform. Computations which rely on batch access to HPC sites are processed by a dedicated service called Rimrock, which can delegate operations to the underlying resources while exposing a RESTful API for IEE (and other tools) to use. In addition, IEE features integration with the Cloudify cloud platform, which enables scheduling cloud VMs from it.

In the context of validating the PROCESS architecture (and that of IEE in particular), we decided to focus on the full pipeline for the use case which involves processing the LOFAR dataset. This enables the means to select the HPC Site and then facilitate the process of staging in data from the LOFAR Long Term Archives and moving it to the relevant site, running relevant computation using site-specific settings (Queuing and Container systems) and finally staging out the results.

For this case we decided to utilize two separate HPC sites: the Prometheus supercomputing cluster at ACC Cyfronet AGH in Kraków (same as for the first prototype), as well as the SuperMUC-NG Cluster at LRZ in Garching bei München. In addition to those HPC sites, to showcase the full capabilities of the platform, we also included a demo of an additional application from another use case (Ancillary pricing for airline revenue management) deployed on the OpenStack cloud at UISAV in Bratislava via the Cloudify component.

Use case applications are organized as projects composed of multiple steps, each of which involves either processing (computations) or is related to data transfer (in particular, staging in the relevant data using the PROCESS data storage infrastructure, or accessing results of computational tasks for visualization and download).

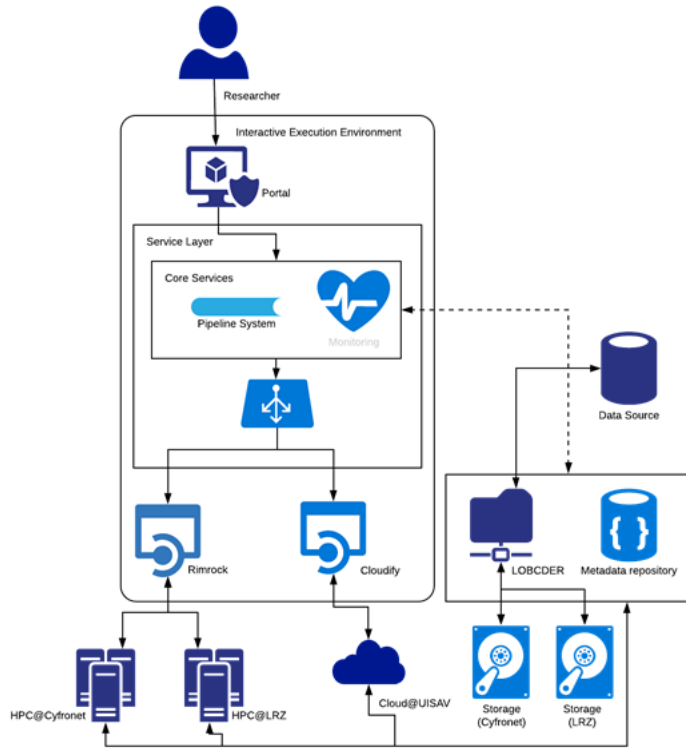

Figure 5. Schematic depiction of the IEE architecture, including its integration with HPC and data storage sites

6.2 Rimrock

The Rimrock component enables running batch scripts on an HPC infrastructure via a convenient REST API. In the scope of the PROCESS project we use it to spawn relevant use case computational components in the form of containers. The container technology is proper for each site, so in the case of Prometheus it is Singularity and in the case of SuperMUC-NG it is Charlie Cloud [21].

A schematic overview of the Rimrock architecture is presented in Figure 7. The service is invoked from the IEE component described in Section 6.1.

6.3 Cloudify Orchestration Service

Service orchestration is often understood as the process of automated configuration, deployment and other management activities of services and applications in the cloud. It can automate execution of different service workflows, including deploy-

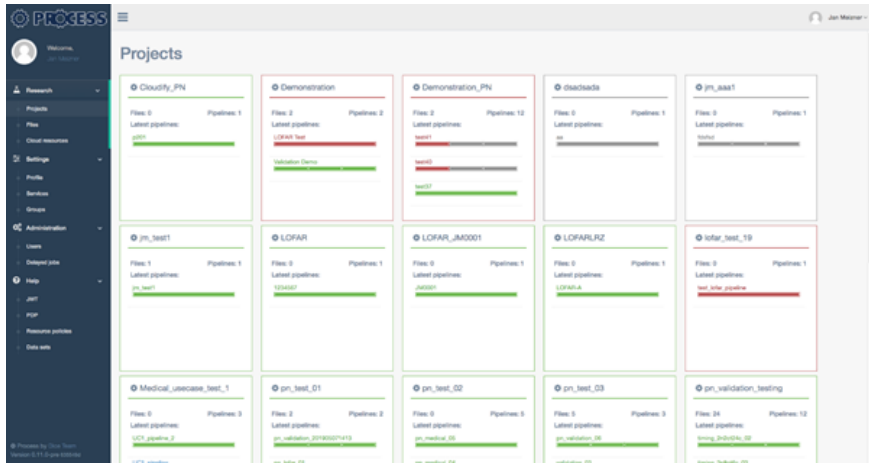

Figure 6. IEE user workbench

ment, initialization, start/stop, scaling, healing of services based on standardized descriptions of composed services, relations between components and their requirements. In the PROCESS project, we use the OASIS TOSCA standard for service description and Cloudify for orchestration.

Cloudify (`https://docs.cloudify.co/4.5.0/about/`) is an open source cloud orchestration platform, designed to automate the deployment, configuration and remediation of application and network services across hybrid cloud and stack environments. It uses OASIS TOSCA templates written in YAML (called blueprints in Cloudify) for defining applications, services and dependences among them. These blueprint files describe also the execution plans for the lifecycle of the application for installing, starting, terminating, orchestrating and monitoring the application stack. Cloudify uses the blueprint as input that describes the deployment plan and is responsible for executing it on the cloud environment. Figure 8 shows the architecture of the Cloudify orchestration service.

Cloudify has its own console, see Figure 9, but for integrating with other services, it is more comfortable using its REST API. Cloudify has completed REST API for all operations related to service orchestration. This REST API can be divided into several sections, the most important ones are the following:

Blueprint: management of TOSCA templates, e.g. upload, download, list, delete.

Deployment: deployment of services and management of already deployed services.

Execution: executing workflows defined in TOSCA templates on concrete deployment, e.g. install, restart, uninstall.

The details of REST API is described at `https://docs.cloudify.co/4.5.0/developer/apis/rest-service/`.

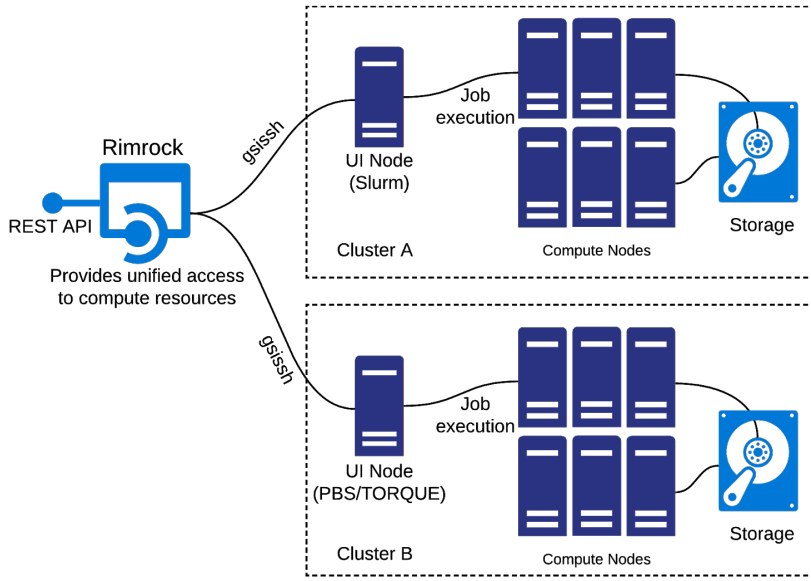

Figure 7. Schematic depiction of Rimrock, including Compute and Storage resources

The API is used by command-line clients (CLI) or scientific gateways (GUI) for deployment and management of services. The services need to be described in TOSCA templates (blueprints) and uploaded to the Cloudify server before deployment using blueprint API. After that, the users can deploy/undeploy instances of services described in the blueprints via deployment API or execute a specific workflow, e.g. restart the service running in the cloud.

Currently Cloudify is integrated with the data micro-infrastructure and IEE. A blueprint has been created for dynamic deployment of new nodes in the cloud and adding them to the Kubernetes cluster for the data micro-infrastructure. The use case of Ancillary pricing for airline revenue management runs in the Cloud using the Cloudify orchestration service. By communicating with the Cloudify API, IEE can manage the execution of the use case in the Cloud.

7 PLATFORM USAGE BY THE USE CASES

In this section we present the ways in which the PROCESS computing platform has been integrated with the use cases considered in the project.

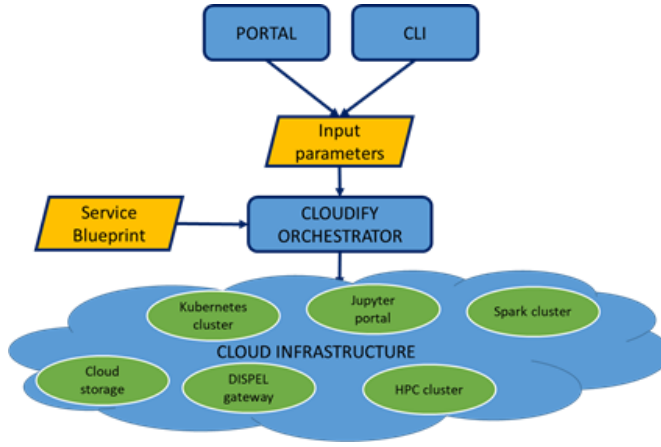

Figure 8. Architecture of Cloudify orchestration service

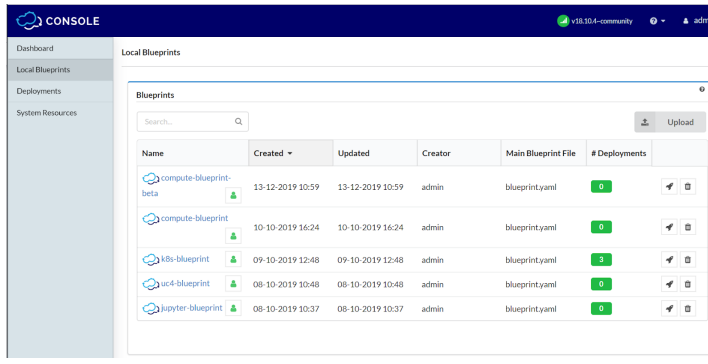

Figure 9. Cloudify console

7.1 Content-Based Search and Classification

The goal of this use case is to improve performance of AI-based medical image analysis using GPU-accelerated distributed computing backed by HPC resources. As the original code was prepared as a Docker container, part of our work was to port it to the Singularity format which is designed for multi-tenant environments.

The basic workflow used for this use case is presented in Figure 10.

7.2 Square Kilometre Array (SKA)

The SKA use case goal is to prepare the computational platform and domain codes for extreme challenges of the SKA radiotelescope when it is available, by using existing datasets procured using the LOFAR telescope.

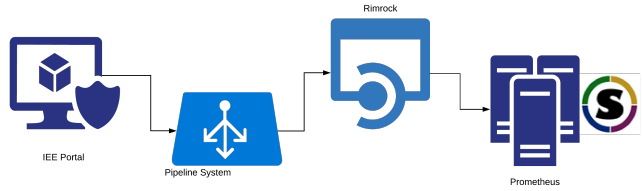

Figure 10. Content-based search and classification use case

Given that LOFAR produces large volumes of data even in its present state, and that this data is stored in multiple locations, this use case obviously presents a challenge for the data transfer subsystem, but at the same time sufficient computational resources must be provided to quickly process incoming data, as well as enable multi-site execution to bring computation nearer to the data.

Additionally, as the process of querying for the right data is complex and requires broad domain knowledge, we have decided to provide scientists who possess such knowledge with a familiar environment. To this end we have undertaken an effort to integrate the existing LOFAR Portal with IEE using a specialized REST API, as shown in Figure 11.

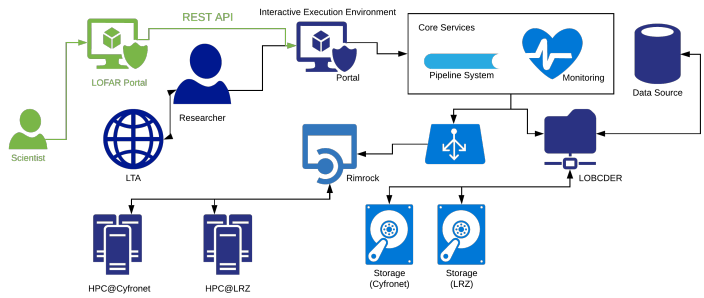

Figure 11. Square Kilometre Array (SKA) use case

7.3 Ancillary Pricing for Airline Revenue Management

This use case differs a bit from the others as it is a business-oriented one. This imposes additional constraints on the compute platform such as the requirement to build a platform that could be easily reproduced in a commercial environment.

To this end we have decided to provide a platform based on cloud resources. The service was deployed using the cloud infrastructure in Slovakia, however such

an environment can be adapted to work with other cloud providers to fulfill the said requirements. The infrastructure utilizes the Cloudify component, as shown in Figure 12.

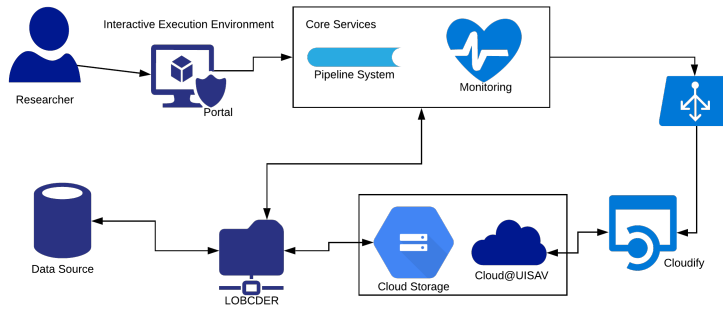

Figure 12. Ancillary pricing for airline revenue management use case

7.4 Agro-Copernicus

Finally, Agro-Copernicus is an example of a use case that features the use of a proprietary component called PROMET, which, due to licensing restrictions, cannot be directly accessed in a fashion similar to other use case codes. The only access mechanism is available via the provided REST API used to control computation, where the code itself is treated as a black box. This solution is shown in Figure 13.

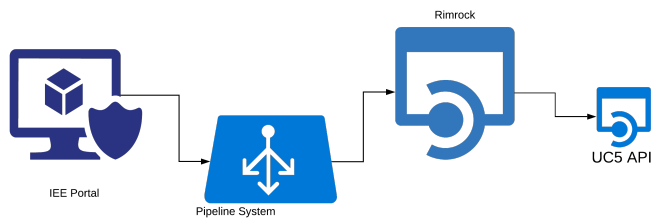

Figure 13. Agro-Copernicus use case

The mechanism can be reused for a wide range of software delivered in the Software as a Service model, as long as the proper API is available.

8 CONCLUSIONS AND FUTURE WORK

In this paper we have presented the work performed in the scope of the PROCESS Project from the beginning until close to its conclusion. This includes state of the art analysis, requirement gathering, platform design and implementation, and, finally, integration with use case codes.

The platform has already been validated using four different use cases, but it is also capable of providing support for other applications, whether based on traditional HPC resources, computing clouds or external services exposing proper APIs.

By the end of the project we aim to provide a unified and straightforward mechanism enabling deployment of all platform components to a containerized environment. In the future we will also seek to further extend the range of supporting cases, as well as make the platform ready for even more powerful upcoming infrastructures.

Within the project we will also continue to follow the containerization approach for scalable HPC applications and will integrate the EASEY framework described in [11], which enables also non-computing experts to easily deploy their applications inside a Charliecloud container through the PROCESS ecosystem.

Acknowledgements

This work is supported by the "PROviding Computing solutions for ExaScale ChallengeS" (PROCESS) project that received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 777533. This research was supported in part by PL-Grid Infrastructure. This work is supported by the project APVV-17-0619 (U-COMP) "Urgent Computing for Exascale Data" and by the VEGA project "New Methods and Approaches for Distributed Scalable Computing" No. 2/0125/20.

REFERENCES

[1] Big Data and Extreme Scale Computing Project. Accessed Oct 26, 2020, available at: `https://www.exascale.org/bdec/`.

[2] ECP – Exascale Computing Project. Accessed Oct 26, 2020, available at: `https://www.exascaleproject.org/reports/`.

[3] ACK Cyfronet AGH, DICE Team: Atmosphere. 2020, accessed Sep 14, 2020, available at: `http://dice.cyfronet.pl/products/atmosphere`.

[4] ACK Cyfronet AGH, DICE Team: Rimrock. 2020, accessed Sep 14, 2020, available at: `https://submit.plgrid.pl`.

[5] Asch, M.—Moore, T.—Badia, R. M.—Beck, M.—Beckman, P. H.—Bidot, T.—Bodin, F.—Cappello, F.—Choudhary, A. N.—de Supinski, B. R. et al.: Big Data and Extreme-Scale Computing: Pathways to Convergence – Toward a Shaping Strategy for a Future Software and Data Ecosystem for

Scientific Inquiry. International Journal of High Performance Computing Applications, Vol. 32, 2018, No. 4, pp. 435–479, doi: 10.1177/1094342018778123.

[6] Ashby, S.—Beckman, P.—Chen, J.—Colella, P.—Collins, B.—Crawford, D.—Dongarra, J.—Kothe, D.—Lusk, R.—Messina, P. et al.: The Opportunities and Challenges of Exascale Computing – Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee. U.S. Department of Energy, Office of Science, 2010.

[7] Bubak, M.—Bartyński, T.—Gubała, T.—Harężlak, D.—Kasztelnik, M.—Malawski, M.—Meizner, J.—Nowakowski, P.: EurValve Model Execution Environment in Operation. In: Turała, M., Bartyński, T., Wiatr, K. (Eds.): CGW Workshop '17, Proceedings. Academic Computer Centre CYFRONET AGH, 2017, pp. 65–66.

[8] Bubak, M.—Meizner, J.—Nowakowski, P.—Bobák, M.—Habala, O.—Hluchý, L.—Tran, V.—Belloum, A. S. Z.—Cushing, R.—Höb, M.—Kranzlmüller, D.—Schmidt, J.: A Hybrid HPC and Cloud Platform for Multidisciplinary Scientific Application. 2020, https://www.process-project.eu/wp-content/uploads/2020/03/SCFE2020-PROCESS-23-03-2020.pdf.

[9] Dongarra, J.—Beckman, P.—Moore, T.—Aerts, P.—Aloisio, G.—Andre, J.-C.—Barkai, D.—Berthou, J.-Y.—Boku, T.—Braunschweig, B. et al.: The International Exascale Software Project Roadmap. International Journal of High Performance Computing Applications, Vol. 25, 2011, No. 1, pp. 3–60, doi: 10.1177/1094342010391989.

[10] GigaSpaces Technologies, Inc.: Cloudify. 2020, accessed Oct 7, 2020, available at: https://cloudify.co/.

[11] Höb, M.—Kranzlmüller, D.: Enabling EASEY Deployment of Containerized Applications for Future HPC Systems. In: Krzhizhanovskaya, V. et al. (Eds.): Computational Science – ICCS 2020. Springer, Cham, Lecture Notes in Computer Science, Vol. 12137, 2020, pp. 206–219, doi: 10.1007/978-3-030-50371-0_15.

[12] Meizner, J.—Bubak, M.—Kapała, J.—Nowakowski, P.—Wójtowicz, P.: Use of the HPC Containers in the Way Towards Exascale. In: Wiatr, K., Bubak, M., Turała, M. (Eds.): CGW Workshop '18. Academic Computer Centre CYFRONET AGH, 2018, pp. 21–22.

[13] Rycerz, K.—Nowakowski, P.—Meizner, J.—Wilk, B.—Bujas, J.—Jarmocik, Ł.—Krok, M.—Kurc, P.—Lewicki, S.—Majcher, M.—Ociepka, P.—Petka, L.—Podsiadło, K.—Skalski, P.—Zagrajczuk, W.—Zygmunt, M.—Bubak, M.: A Survey of Interactive Execution Environments for Extreme Large-Scale Computations. KDM'18, Zakopane, Poland, 2018.

[14] Bobák, M.—Belloum, A. S. Z.—Nowakowski, P.—Meizner, J.—Bubak, M.—Heikkurinen, M.—Habala, O.—Hluchý, L.: Exascale Computing and Data Architectures for Brownfield Applications. 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Huangshan, China, IEEE, 2018, pp. 461–468, doi: 10.1109/FSKD.2018.8686900.

[15] Bobák, M.—Hluchý, L.—Belloum, A. S. Z.—Cushing, R.—Meizner, J.—Nowakowski, P.—Tran, V.—Habala, O.—Maassen, J.—Somosköi, B.

et al.: Reference Exascale Architecture. 2019 15th International Conference on eScience (eScience), San Diego, CA, USA, IEEE, 2019, pp. 479–487, doi: 10.1109/eScience.2019.00063.

[16] Narasimhamurthy, S.—Danilov, N.—Wu, S.—Umanesan, G.—Markidis, S.—Rivas-Gomez, S.—Peng, I. B.—Laure, E.—Pleiter, D.—de Witt, S.: SAGE: Percipient Storage for Exascale Data Centric Computing. Parallel Computing, Vol. 83, 2019, pp. 22–33, doi: 10.1016/j.parco.2018.03.002.

[17] PL-Grid Consortium. The PL-Grid National Computing Infrastructure. 2020, accessed Oct 7, 2020, available at: http://www.plgrid.pl/en.

[18] Reed, D. A.—Dongarra, J.: Exascale Computing and Big Data. Communications of the ACM, Vol. 58, 2015, No. 7, pp. 56–68, doi: 10.1145/2699414.

[19] TOP500.org. The TOP500 List (November 2017). 2017, accessed Sep 14, 2020, available at: https://www.top500.org/lists/top500/list/2017/11/.

[20] TOP500.org. The TOP500 List (June 2020). 2020, accessed Sep 14, 2020, available at: https://www.top500.org/lists/top500/list/2020/06/.

[21] Triad National Security, LLC. Charliecloud. 2020, accessed Oct 7, 2020, available at: https://hpc.github.io/charliecloud/.

[22] US Department of Energy, Office of Science. Advanced Scientific Computing Research (ASCR): Scientific Grand Challenges Workshop Series. Accessed Oct 26, 2020, available at: https://science.osti.gov/ascr/Community-Resources/Workshops-and-Conferences/Grand-Challenges.

[23] Cushing, R.—Valkering, O.—Belloum, A.—Souley, M.—Bobak, M.—Habala, O.—Tran, V.—Graziani, M.—Müller, H.: Process Data Infrastructure and Data Services. Computing and Informatics, Vol. 39, 2020, No. 4, pp. 724–756, doi: 10.31577/cai_2020_4_724.

[24] Bobák, M.—Hluchý, L.—Habala, O.—Tran, V.—Cushing, R.—Valkering, O.—Belloum, A.—Graziani, M.—Müller, H.—Madougou, S.—Maassen, J.: Reference Exascale Architecture (Extended Version). Computing and Informatics, Vol. 39, 2020, No. 4, pp. 644–677, doi: 10.31577/cai_2020_4_644.

Jan Meizner has graduated majoring in federated IT security systems. Since then he has been working at ACC Cyfronet AGH on many EU and national projects involving a wide range of subjects, including computational medicine. His work focuses on IT security, operations of cloud and HPC infrastructures, as well as building software for such infrastructures. Currently involved also in Sano Centre for Computational Medicine, focusing on the operations of IT systems, as well as a range of IT security tasks, including identity management and data security.

Piotr Nowakowski is Research Programmer at the Academic Computing Centre CYFRONET AGH and Senior Data Scientist at the Sano Centre for Computational Medicine. He specializes in design and development of distributed environments for computational science, and he has participated in a range of national and international research initiatives, including EU-funded projects – most recently VPH-Share, EurValve and PROCESS. He is the author or co-author of over 100 scientific publications.

Jan Kapala received his B.Sc. degree in computer science from AGH University of Technology, Krakow, Poland in 2020 and now he is pursuing the M.Sc. degree in the M.Sc. programme Computer Science and Intelligent Systems: Artificial Intelligence and Data Analysis. Both his B.Sc. thesis and ongoing M.Sc. thesis are focused on reinforcement learning agents. He is Software Engineer at Academic Computer Centre CYFRONET AGH. His main interest is artificial intelligence.

Patryk Wojtowicz received his B.Sc. degree in computer science from AGH University of Technology, Krakow, Poland in 2020 and now he is pursuing the M.Sc. degree. His B.Sc. thesis and ongoing M.Sc. thesis are both focused on intelligent reinforcement learning agents. He is involved in development of the interactive execution environment platform in the PROCESS project at Academic Computer Centre CYFRONET AGH. His deep interests are artificial intelligence and its ethical aspects, and data science.

Marian Bubak obtained his M.Sc. in technical physics and his Ph.D. in computer science from the AGH University of Science and Technology, Krakow, Poland. He is the Scientific Affairs Director and President of the Management Board of the Sano – Centre for Computational Personalised Medicine – International Research Foundation (`https://sano.science/`). He also leads the Laboratory of Information Methods in Medicine at ACC Cyfronet AGH, he is a staff member of the Department of Computer Science AGH, and the Professor of Distributed System Engineering (emeritus) at the Institute of Informatics of the University of Amsterdam. His research interests include parallel and distributed computing and quantum computing. He served key roles in about 15 EU-funded projects and authored about 230 papers. He is a member of editorial boards of FGCS, Bio-Algorithms and Med-Systems, and Computer Science Journal.

Viet Tran is Senior Researcher of the Institute of Informatics, Slovak Academy of Sciences (IISAS). His primary research fields are complex distributed information processing, grid and cloud computing, system deployment and security. He received his M.Sc. degree in informatics and information technology, Ph.D. degree in applied informatics from the Slovak University of Technology (STU) in Bratislava, Slovakia. He actively participates on preparations and solving a number of EU IST RTD 4^{th}, 5^{th}, 6^{th}, 7^{th} FP and EU H2020 projects such as PROCESS, DEEP-HybridDataCloud, EOSC-Hub and EOSC-Synergy. He is the author or co-author of over 100 scientific publications.

Martin Bobák is Scientist at the Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia, in the Department of Parallel and Distributed Information Processing. He started working at the institute in 2013, defended his dissertation thesis at the institute in 2017, became Member of the Scientific Board of the institute, and Guest Handling Editor in the CC Journal Computing and Informatics. His field of research is cloud computing and the architectures of distributed cloud-based applications. He is the author of numerous scientific publications and has participated in several European and Slovak R & D projects.

Maximilian Höb is Associate Scientist in the Munich Network Management Team at Ludwig-Maximilians-University Munich and Co-Coordinator of the PROCESS project, in which he also contributes to two Use Cases in the area of data management and agricultural simulation based on the Copernicus data sets. His research focuses on large scale system architectures and performance-aware containerization of HPC applications.