Reference exascale architecture

Martin Bobák, Ladislav Hluchý
Institute of Informatics
Slovak Academy of Sciences
Bratislava, Slovakia
{martin.bobak, ladislav.hluchy}@savba.sk

Adam S. Z. Belloum, Reginald Cushing
Informatics Institute
University of Amsterdam
Amsterdam, Netherlands
{a.s.z.belloum, r.s.cushing}@uva.nl

Jan Meizner, Piotr Nowakowski
ACC Cyfronet AGH
AGH University of Science and Technology
Krakow, Poland
{j.meizner, p.nowakowski}@cyfronet.pl

Viet Tran, Ondrej Habala
Institute of Informatics
Slovak Academy of Sciences
Bratislava, Slovakia
{viet.tran, ondrej.habala}@savba.sk

Jason Maassen
Netherlands eScience Center
Amsterdam, Netherlands
j.maassen@esciencecenter.nl

Balázs Somosköi
Lufthansa Systems
Berlin, Germany
balazs.somoskoi@lhsystems.com

Mara Graziani
University of Applied Sciences
Western Switzerland (HES-SO)
Sierre, Switzerland
mara.graziani@hevs.ch

Matti Heikkurinen
University of Applied Sciences
Western Switzerland (HES-SO)
Sierre, Switzerland
Ludwig-Maximilians Universität
Munich, Germany
matti.heikkurinen@iki.fi

Maximilian Höb, Jan Schmidt
Munich Network Management Team (MNM-Team)
Ludwig-Maximilians Universität
Munich, Germany
{hoeb, schmidtja}@nm.ifi.lmu.de

***Abstract*—While political commitments for building exascale systems have been made, turning these systems into platforms for a wide range of exascale applications faces several technical, organisational and skills-related challenges. The key technical challenges are related to the availability of data. While the first exascale machines are likely to be built within a single site, the input data is in many cases impossible to store within a single site. Alongside handling of extreme-large amount of data, the exascale system has to process data from different sources, support accelerated computing, handle high volume of requests per day, minimize the size of data flows, and be extensible in terms of continuously increasing data as well as increase in parallel requests being sent. These technical challenges are addressed by the general reference exascale architecture. It is divided into three main blocks: virtualization layer, distributed virtual file system, and manager of computing resources. Its main property is modularity which is achieved by containerization at two levels: 1) application containers – containerization of scientific workflows, 2) micro-infrastructure – containerization of extreme-large data service-oriented infrastructure. The paper also presents an instantiation of the reference architecture - the architecture of the PROCESS project (PROviding Computing solutions for ExaScale ChallengeS) and discuss its relation to the reference exascale architecture. The PROCESS architecture has been used as an exascale platform within various exascale pilot applications. This work will present the requirements and the derived architecture as well as the 5 use cases pilots that it made possible.**

***Index Terms*—exascale computing; architecture; functional design; qualitative analysis**

I. Introduction

New scientific instruments (e.g. distributed radio telescopes such as LOw-Frequency ARray – LOFAR, Square Kilometre Array – SKA, space telescopes such as Copernicus sentinels, etc.) are producing data at an accelerating pace. LOFAR observations are stored in the long term archive (LTA) which is distributed over Amsterdam, Jülich and Poznan. It currently contains around 30 PB of data and grows with 5 to 7 PB/year. SKA represents an even bigger challenge. It is expected that a raw data will grow by zettabytes/year which will produce 130 to 300PB/year of correlated data. Copernicus sentinels also present an exascale challenge. They produce approximately 7.5 PB of raw data each month.

This work is supported by "PROviding Computing solutions for ExaScale ChallengeS" (PROCESS) project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777533, project APVV-17-0619 (U-COMP) "Urgent Computing for Exascale Data", and project VEGA 2/0167/16 "Methods and algorithms for the semantic processing of Big Data in distributed computing environment".

978-1-7281-2451-3/19/$31.00 ©2019 IEEE
DOI 10.1109/eScience.2019.00063

Another significant amount of data is generated by branches that are digitized. A typical example is a medical science. The final report of the High Level Expert Group on Scientific Data [1] describes it as follows: "In 2010, about 2.5 petabytes – more than a million, billion data units – are stored away each year for mammograms in the US alone. World-wide, some estimate, medical images of all kinds will soon amount to 30% of all data storage."

With the rapid growth of data [2], [3] it is often required to migrate data to a remote computation location [4]. Often times the data structures are very complex and are stored in a (geographically) distributed infrastructure. Those features are so significant, that new approaches and methods need to be investigated. This paper presents an architectural blueprint that allows the whole data and compute infrastructure to draw maximal benefits from the emerging exascale capacities.

The main aim of this paper is to describe the reference architecture for exascale systems which starts from the gathering of requirements to its application in real world use cases. In the context of our work and of this paper, an exascale system is one that uses exabytes of data or exaflops of computational power. The design of the reference architecture is driven by the requirements analysis of the various use cases which come from diverse scientific communities as well as the industry. Through the generalisation of them, the reference architecture is proposed.

This paper has the following structure:

- Section II represents the requirements analysis of the various exascale-related use cases
- Section III presents the reference exascale architecture
- Section IV describes the updated technology-based architecture of the PROCESS Project.

II. Analysis of the Use Cases on the Requirements of the Exascale-ready Data Platform

There are many research communities reaching the exascale threshold. This paper investigates requirements coming from the following communities[1]: (a) medical science, (b) astronomy, (c) behavioural modelling and analysis, and (d) harvest prediction [5]. The requirements coming from the communities can be divided into two groups: (i) computational requirements, and (ii) accessing and processing of data sets (exceeding petabytes). The two requirements categories are not completely isolated. Contrariwise, both of them are tangled which also creates new additional requirements. One of them is distributed and/or parallel processing of data which is the consequence of data amount generated by various simulations, and observations conducted by the above mentioned exascale research communities coming from distributed data sources and/or laboratories.

A. Exascale learning on medical image data

The medical use case focuses on automated cancer diagnostics and treatment planning. Its aim is to study cancer detection, localisation and stage classification. Cancer diagnostics is based on the automatic analysis of a biopsy or surgical tissue specimens, which are captured by a high resolution scanner and stored in a multi-resolution pyramid structure. The amount of the data set is huge (up to PBs), since it also includes tissue that is not relevant for cancer diagnosis (e.g. background, stroma, healthy tissue etc.). The whole processing workflow is depicted in Figure 1.

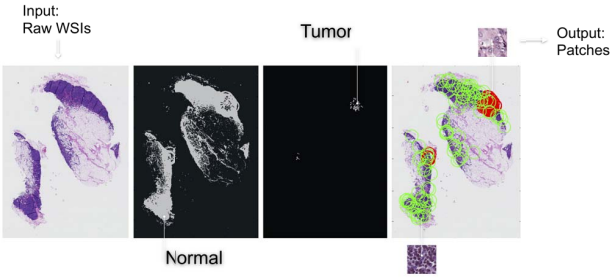

Fig. 1. Medical use case processing pipeline [5].

The key components of this use case are focused on pattern recognition, statistical modelling and deep learning. However, the approach could be relevant in modelling problems where the amount of data to be analysed is the key challenge. The typical requirement coming from this kind of data processing tasks is huge amount of computing power [7], [8]. Extremely large datasets might be difficult to download [9], and hospitals might require a high level of confidentiality. As a generalised solution, the "Evaluation as a Service" (EaaS) could be seen as a "Clean slate" approach to deal with very large datasets, especially ones that require sophisticated access control e.g. due to privacy issues. In EaaS the data remains in a central infrastructure and does not need to be moved. For example, data might remain internal at hospitals, or be made available in a closed cloud environment. The VISCERAL project[2] represents an example where the data is kept in a central space and the algorithms are moved to the data. The main benefits involve the possibility of working with the data in situ and therefore allowing hospitals not to reveal sensible data. Moreover, the use of the computational resources is reserved to the training of the models, which has the highest computational demands.

Cancer diagnostics is generally time consuming and reports high disagreement rates between pathologists, since tumour stages do not have defined boundaries. Through the visualisation and interpretation of the network decisions, the level of objectivity in the diagnostics can be increased, consequently reducing the analysis time and disagreement rates. The mole of data to process to train models that can statistically generalize to new patient cases is often of the size of several Terabytes. The extraction of images of tissue from the gigapixel histopathology images of breast cancer, can easily grow to more than 60 thousand patches for the single record. Moreover,

[1]They are part of the PROCESS project, http://process-project.eu/

[2]Visual Concept Extraction Challenge in Radiology., http://www.visceral.eu/

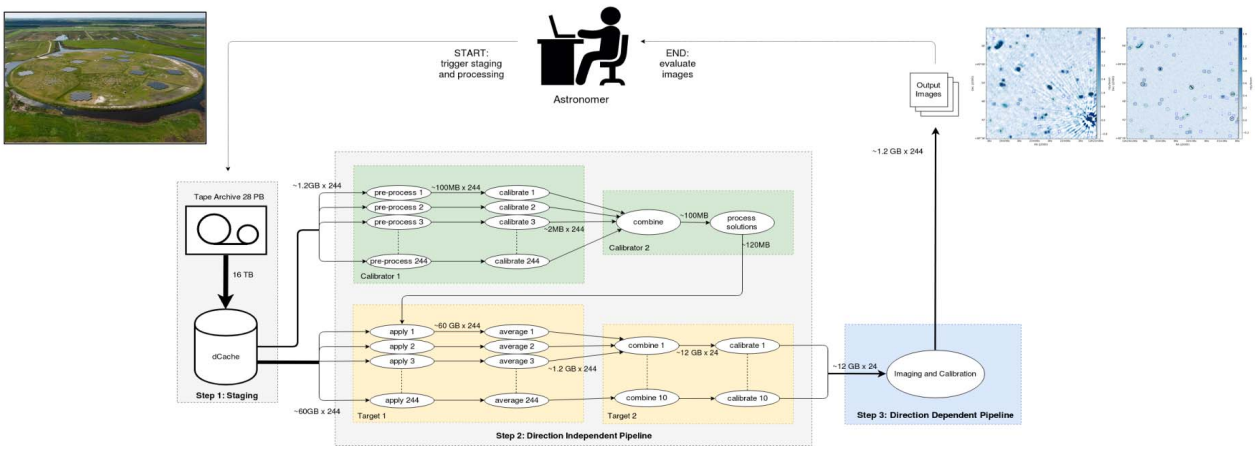

Fig. 2. LOFAR use case processing pipeline [5], [6].

more than one record (between 5 and 20) is kept per each patient. If state-of-the-art model for image classification are used, the number of operations needed for training is of the order of magnitude of 1017, with at least 3.6 GFLOPS required per each full scan of the training data (i.e. epoch). If the depth of the network is increased and more than one epoch of training is considered, the computations required scale up to the order of magnitude of ExaFlops and several days on a single dedicated GPU. Moreover, such a large dataset (each resolution level occupies on average 50 to 60 GB) calls for a dedicated infrastructure to handle the storage requirements and data processing.

The requirements analysis pointed out the need for Docker or Singularity containers in the software environment. Singularity containers are generally better suited to High Performance Computing (HPC) clusters, as they provide essential features such as improved security and portability, as well as the ability to operate in userspace without the need for root access on the HPC machine itself (which is typically not provided to standard users). While Docker containers are more flexible and often encountered in medical research applications, Singularity containers can be used to package entire scientific workflows, software and libraries. Semiautomatic conversion between Docker and Singularity containers also remains an option.

As a consequence of these conflicting requirements, the need of a flexible conversion from Docker containers to Singularity arises to guarantee software integration. Furthermore, access to GPUs has to be ensured even in the Singularity framework, as it is essential for the medical use case. The Environments Manager has to guarantee a flexible building, deployment and management of multiple running applications. Training monitoring is one of the essential requirements for exascale training [10]. The distribution of the training across the different computing resources needs to be monitored constantly.

A data storage system needs to take into account the different image formats and resolutions across the datasets and ensures flexibility and adaptation to a variety of datatypes. For instance, WSIs are generally saved as BIGTIFF files, and paired with annotation files that might be either XMLs or CSVs of even TXTs. Moreover, datasets such as PubMed central require the possibility to handle and process JPEG, MPEG and more common image compression formats with the same degree of elasticity. As exascale requirements, we need the distribution of the datasets across the different computational centres, ensuring fast connectivity between the local storage and the computational resources to reduce internal latency.

Support of performing dense linear algebra on distributed-memory HPC systems. Multiple GPU computation libraries should be used to merge multiple CUDA kernels. Furthermore, top-level development of the deep models should be performed using the most common machine learning and deep learning frameworks (e.g. Tensorflow 1.4.0, Keras 2.1.2, TFLearn, Caffe, Microsoft CNTK, Nvidia DIGITS, Theano, etc.).

The distribution of training runs across different centres requires automated detection of the optimal model parameters and efficient scheduling of processes to the available resources.

Data acquired in conjunction with hospitals needs to be pseudonymised, thus retaining a level of detail in the replaced data that should allow tracking back of the data to its original state. In this way the ethical constraints related to the usage of patient data will also be addressed. However, the use of sensible data sets the need for specific security requirements:

1) Access control. Access to the data should be restricted to a subset of authorised users.
2) Traceability. It should be possible to trace back access history to the user, place and date/time when the access was performed.
3) The Evaluation-as-a-Service approach will address the privacy constraints concerning the use of patient data in research. Algorithms used to detect ROIs and visual similarities in the data can be designed by using large training datasets from a variety of sources, such as

cohort studies. In such a scenario, pattern detection and classification are performed in a closed environment, namely virtual machines or Docker containers. This approach supports an open development of models and algorithms, and avoids data protection issues by processing information that should not leave the hospitals "in situ".

B. LOFAR use case

Low Frequency Array (LOFAR) is a state-of-the art radio telescope capable of wide field imaging at low frequencies. It has been ingesting data into a long-term archive (the LOFAR LTA) since 2012 and its volume is now expanding at a rate of approximately 5-7PB/year. Its current volume is about 28 PB. This consists mostly of "Measurement Sets", i.e. visibilities - correlated signals from LOFAR stations. The LOFAR use case is depicted in Figure 2.

The core requirement is the provision of a mechanism to run containerized workflows, thereby improving the portability and easy of use. Analysing the massive volumes of data stored in the archive is an acute problem. The environment for selecting the data and workflows has to be user-friendly , and it has to support launching the workflows, monitoring the results and downloading outputs. The whole workflow needs to be containerized by Docker or Singularity containers as workflow steps to allow each step to use different analysis tools and dependencies.

The platform must have a mechanism to run the workflows on suitable processing hardware. While some parts of the workflow may run in parallel on relatively simple compute nodes (24 cores, 8 GB memory, 100GB scratch storage), other parts currently run sequentially on a fat node with significant memory (256 GB or more, 3 TB scratch storage). The data management system has to be capable of efficiently transporting the Measurement Sets from the archive locations in Amsterdam, Juelich and Poznan to the processing locations.

The capability to horizontally scale to a significant number of compute resources to in order to run a large number of (independent) workflows at the same time. Since processing the entire archive for a single science case already requires a significant amount of core hours O(47M), handling multiple science cases simultaneously will require up to exascale resources.

C. Ancillary pricing for airline revenue management

The ancillary pricing use case concentrates on an analysis of current ancillary sales and hidden sales pattern. This aim is tackled by machine learning approaches (e.g. random forest and neural networks) for pricing of offered ancillaries. The overview of the use case is shown in Figure 3.

The core requirement is a platform that is capable of storing the incoming ancillary data in a way that allows easy exploitation for airlines. On the one hand, the platform should provide libraries for machine learning and quick processing for the model learning, while on the other hand storing the models in an efficient way such that several hundred million ancillary pricing requests a day can be answered. The platform needs to be capable to deal with the large passenger data sets that airlines generate. It has to be based on a scalable architecture which has to foster the following features:

- handle large amount of data
- handle data from different sources
- handle high volume of requests per day
- provide quick response times
- be extensible in terms of continuously increasing data as well as increase in parallel requests being sent.

An average airline may handle approximately 100 million passengers per year (the largest airlines carry up to twice as many passengers), each of whom will buy on average 5 ancillaries. Each ancillary record can be several kilobytes in size, and several years of data need to be processed. During processing, the size of the data is further increased due to the specifics of the used algorithms.

The data do not only need to be stored, but have to provide efficient algorithmic usage which means on the one hand the update of the model parameters within a reasonable timeframe (e.g. within a nightly time slot). On the other hand, this implies real-time responses with revenue-optimal prices upon customer request. Tool-stack of the platform has to support Lambda Architecture principles especially for historical data and further statistical analyses (e.g. applying mathematical and statistical algorithms on consolidated data structure to identify an optimal reference model, or applying variables of incoming requests on the optimal reference model to compute probability, estimates and forecast). The platform has to also support a processing of ongoing data streams to keep the consolidated data structure up-to-date (i.e. learning new data behaviour into reference data). Also distributed computing fundamentals has to be one of the core features (e.g. support of the Hadoop ecosystem).

Ancillary data are personal data. However, they do not carry the strictest privacy requirements, since the data do not contain names, addresses or credit card information. However they may contain data that directly connect to a person such as frequent traveller information. If using real data this will be considered as confidential information provided by the involved airlines. This results in the following requirements:

The software is supposed to be run at airlines in the European Union. Therefore it has to comply with the "EU General Data Protection Regulation", if applicable. The software needs to be deployable on-site at the customer's cloud service, for example Microsoft Azure.

D. Agricultural analysis based on Copernicus data

The main focus is on Multi-Model Simulations which are seamlessly linked with observational data (including also sources like social media and local sensor networks to complement satellite observations) to improve accuracy, relevance and efficiency beyond what would be possible to achieve using either simulation or observational data on their own.

The key inputs for the analysis are the Copernicus datasets, consisting of data from several satellites (Sentinel family) that

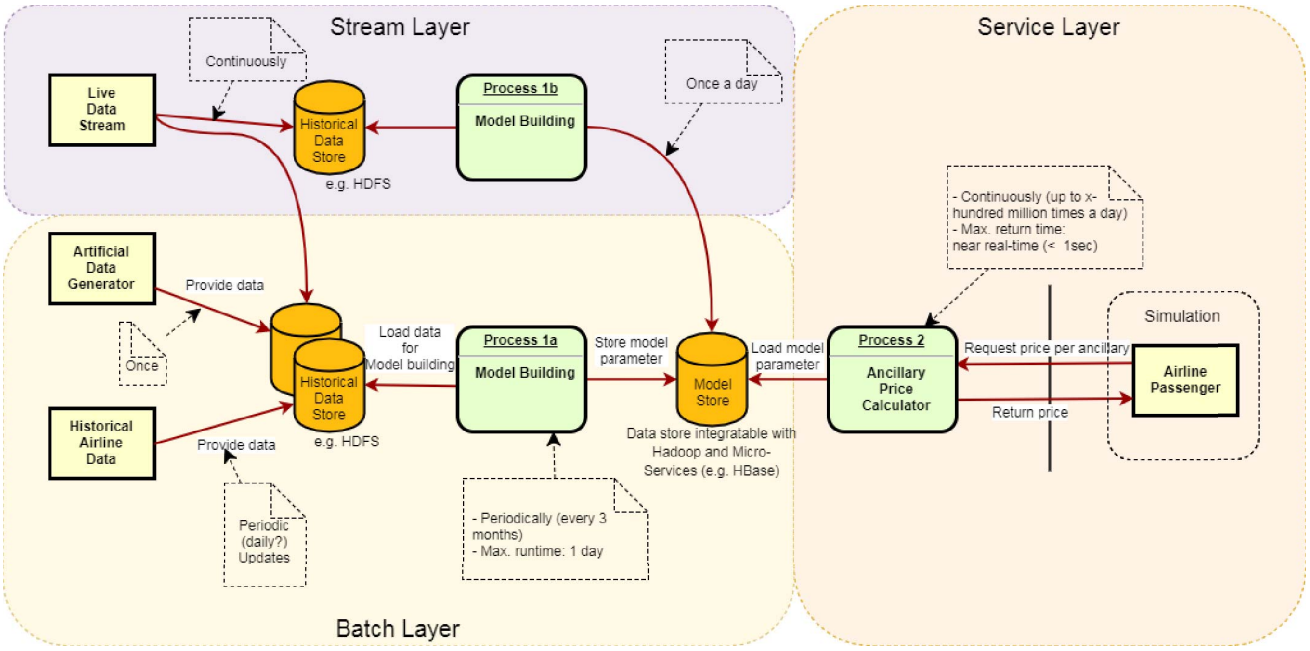

Fig. 3. The workflow of the ancillary pricing use case [5].

produce radar and visible light data with up to 10m resolution. The use case uses is a tightly coupled with a modelling framework PROMET (Processes of Mass and Energy Transfer) that combines modelling the macro-scale carbon cycle, water and energy inputs with the behaviour of water and key nutrients in the soil with analysis of plant metabolism and even human decision-making.

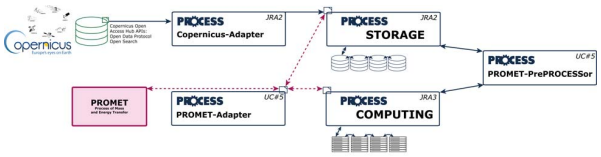

Fig. 4. The workflow of the Copernicus use case [5].

The use case requires linking and comparing modelling and simulation results with the actual observational data, supporting formation of hypotheses and quality assessment of the simulation tools. Providing this functionality requires:

- Extraction of relevant time series from the Sentinel data (up to 7.5PB per month)
- Storing the simulated data and associated metadata in a way that it can be linked with the Sentinel time series
- Providing generic API for using Copernicus data with any modelling framework

E. Supporting of next exascale communities

According to the previous use cases, the PROCESS platform offers a generic environment capable to deploy fully containerized scientific workflows. Through the distributed virtual file system, the containers are able to access data within the storage federation by pointers. The platform also provides methods for metadata management and effective data transferring on any hierarchical datasystem-based data source.

Supporting innovation based on global disaster risk data is an use case which context is to show that it is possible to use exascale tools to deploy and provide services to user that may some day become the “long tail of exascale”. For now, we have made available some of the datas ets behind the 2015 GAR and 2017 Risk Atlas, and are in the process of adding/cross-linking with complementary hazard-specific data sets. However, as the the 2019 GAR report indicates[3], the risk assessment community processes are undergoing a paradigm shift. Now only are the hazard/exposure/vulnerability data sets becoming more fine-grained and comprehensive (increasing the amount of data), but the integrated, systems-based approach require analysis that takes the interactions and feedback loops between different hazards into account. Simply analysing the hydrologic impact of an earthquake and potential damages caused by chemical spills triggered by the floods that have been caused by landslides can quite easily turn probabilistic hazard modelling into an exascale problem.

III. Reference Exascale Architecture

After reviewing of all requirements coming from different user communities (such as medicine, radio astronomy,

[3]The new UNISDR Global Assessment Report was just published (https://gar.unisdr.org/), in the recommendations (page 420) they recommend that member states adapt and approach called “integrated risk governance assessment” that takes “into account multiple hazards (man-made, natural and mixed) and related risks, the way hazards, vulnerability and economic activity interacts with the environment and with each other within and among complex systems, and the need to adapt policy and implementation to enable systems-based approaches to risk reduction.”

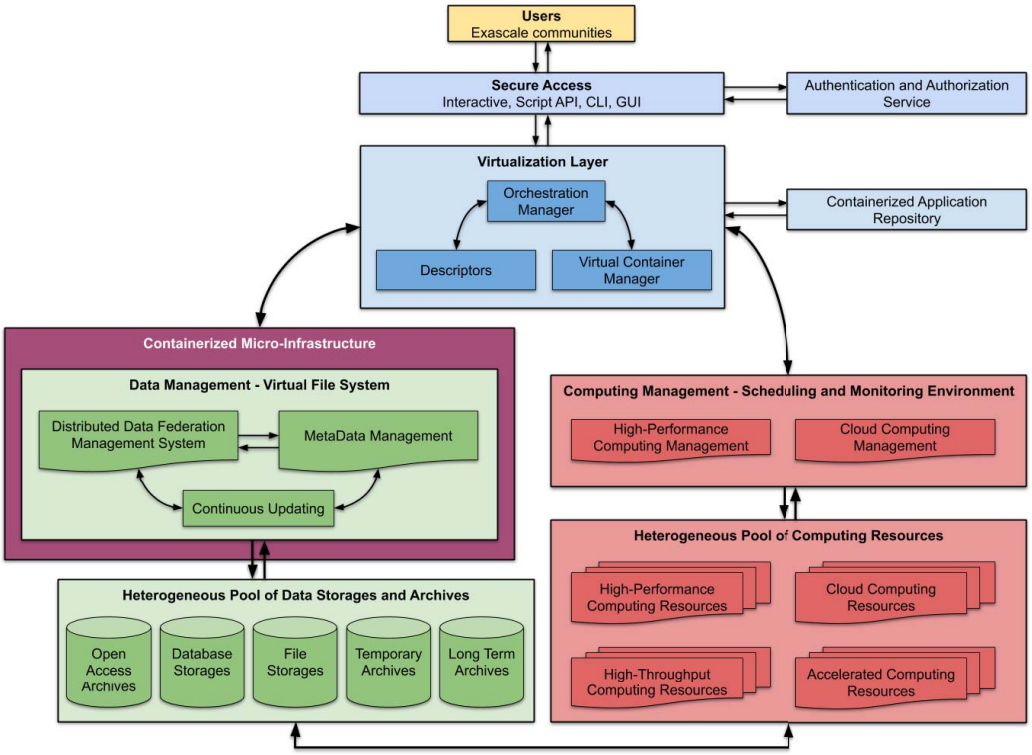

Fig. 5. Reference exascale architecture.

airline revenue management, etc.), the main challenge was to propose an architecture that is suitable for all of them. Their requirements can be divided into three main groups: 1) virtualization requirements, 2) data requirements, and 3) computing requirements.

Virtualization requirements are very straightforwardly derived from application platforms of our user communities – support of containers which offers a lightweight virtualization approach which is similar to application packages. Its advantages include easy deployment and maintenance, flexibility, reliability, scalability, etc. Since the users' applications need to be deployed on various computing infrastructures, portability and interoperability are very important features of every exascale-capable platform.

The core data requirement is handling of exascale data sets or extreme data flows which is not possible to manipulate and manage by a single data center nowadays. It brings a very demanding and ambitious request – a data federation across multiple data centers. It is also tangled with metadata management (processing of such data is impossible without their description – the metadata). The main challenge is communication or data transmission in terms of data services. The exascale platform has to support huge data transfers across the whole infrastructure.

The main computing requirement is supporting high-performance computing as well as cloud computing which is able to offer accelerated computing also – a computing environment for the application containers. The other significant requirement is performance optimization. The current trends in the scientific applications are the following: high distribution across different research computing centers or nodes, and a degree of parallelism and concurrency also increased. Those challenges need to be taken into account during designing of computing management.

The proposed architecture is driven by modularity and scalability. These two approaches are the most suitable for an environment in which the core features are high distribution and massive parallelism. The modularity also enables to extend and adjust the platform, according to the needs of new user communities. It gives flexibility in using its sub-modules in a way which exploits the heterogeneous resources of exascale systems the most efficiently.

The aim of the proposed reference architecture is to characterize key attributes and properties that have to be handled by every scientific application using exascale data and computations. From altogether viewpoint, the reference exascale architecture (see Figure 5) is divided into the following parts (from top to bottom):

- **Users of the exascale scientific applications** (in yellow) - the exascale system has to support functionalities required by its user communities. That also means to support legacy applications in some cases (see the Copernicus use case). According to the initial and up-

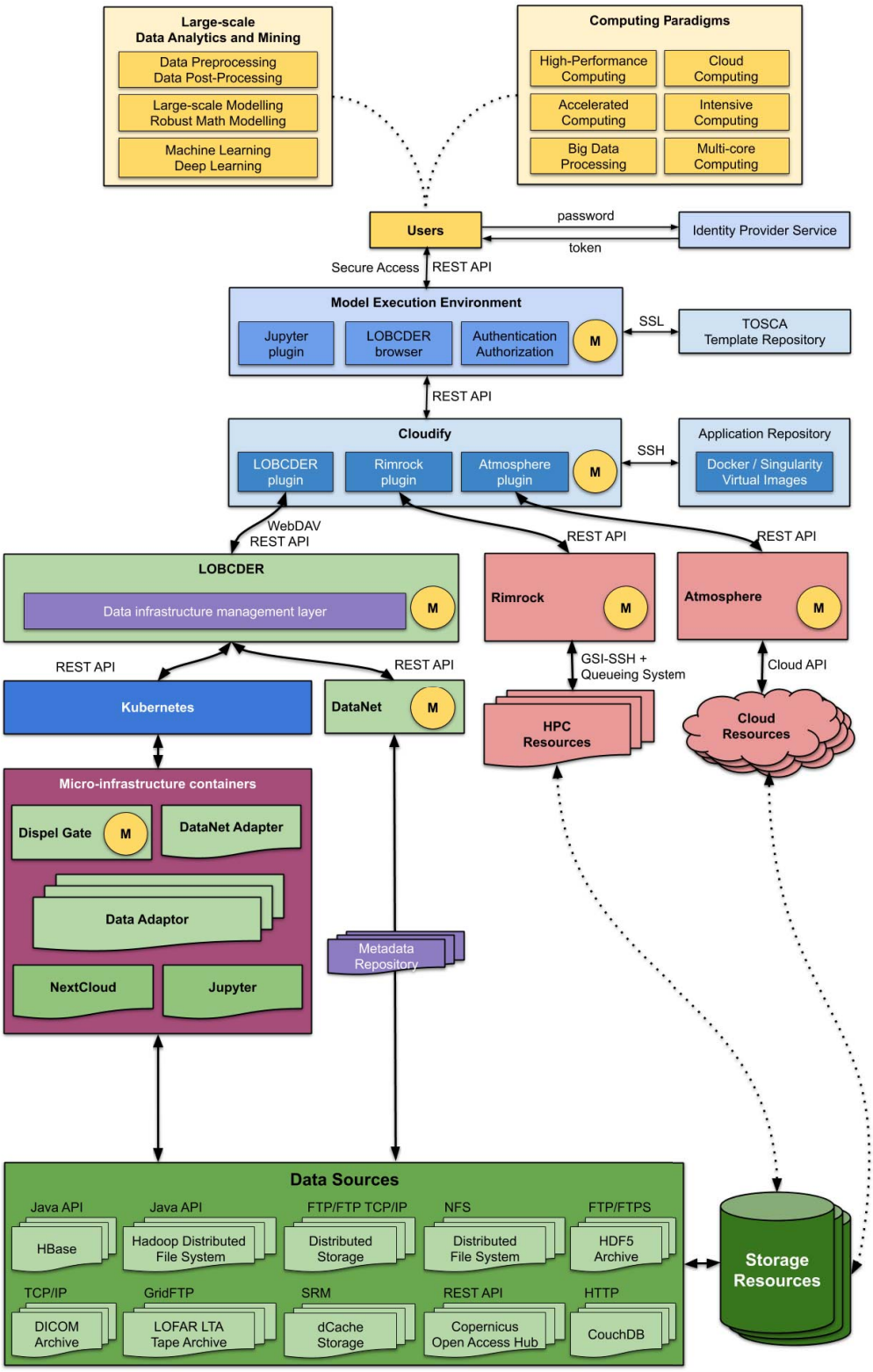

Fig. 6. PROCESS architecture (the yellow "M" token represents connection with a monitoring system. The PROCESS platform uses Zabbix as a monitoring agent.).

dated requirements analysis, the best way is to build it on containerization. All of the applications are stored in a containerized repository which is available to use communities. The users are accessing the exascale platform through virtualized scientific portals which are also containers providing a user friendly graphical interface. The proposed GUI is easily templated and so modified according to the needs of its user community. The containerization approach is flexible, scalable, reusable and ready to use. Moreover, it does not require any special technical skills (especially, related to integration - exascale data processing is often contingent on complex software tools involving expert knowledge about its management) to make it run on the resource infrastructure (see the LOFAR use case).

- **Virtualization layer** (in blue) - is situated between the containerized application repository and platform infrastructure managers. Interoperability of data and computing infrastructure is the key and critical requirement of the exascale systems. To use both infrastructures in the most efficient way, we propose the exascale reference architecture based on containerisation instead of virtual machines. The performance of the infrastructure as the whole is utilized in a better way. It is caused by minimization of overheads (e.g. software duplications). Thus virtualization layer based on containerization approach exploits the infrastructure resources in the most optimal way and also it supports the requirements from our user communities. The main requirement coming from the users is supporting of various application containers. According to a type of computing resources, they can be divided into two groups: 1) HPC containers, and 2) cloud containers. Thus the virtualization layer has to be capable to handle them in cooperation with lower layers (data management and computing management).
- **Data management** (in green) - requirements coming from the exascale scientific applications could be divided into two main groups: distributed data federation, and metadata. It is very common that the exascale scientific applications have highly complicated datasets, that need to handled and processed by relevant systems. For that purpose, its file systems must have a module capable to work with metadata. The metadata module has to be federated and distributed as well as the management system for the data infrastructure itself. At this level of the infrastructure, the system architect has to be careful whether the component will be containerized, or not. On the one hand, the exascale system has to avoid overhead and latency (according to our experiments, it is caused by needless duplication of software) thus we prefer containers to virtual machines. However, on the other hand, all infrastructural services do not need to be virtualized. For example, virtualization of HBase (through containers, or virtual machines) is not necessary because it leads to dataset duplication in the worst case or a performance overhead in the best case. Thus a better approach is to support infrastructural services ecosystem through micro-services. Micro-services serve as adapters and connectors to infrastructural services. They are integrated into a containerized micro-infrastructure, which is customized according to requirements coming from a use case and connecting them to a distributed virtual file system. The micro-infrastructure allows for application-defined infrastructures with the main advantages being threefold: First, services can be customized for the application; e.g., data staging service. Second, minimizing global state management (a major scaling issue); e.g., instead of having one global index for all files for all applications, have micro-infrastructures manage their own local indices and states. Third, micro-infrastructures are isolated from each other, which increases security between users of different applications. The PROCESS distributed file system layer needs to be virtualized because it has to run on top of multiple file systems. Also, it is crucial that access to a data storage federation is unified. Thus the virtual file system is distributed.
- **Computing Management** (in red) - this part of the infrastructure is related to scheduling and monitoring computing resources. The infrastructure has to be loaded as balanced as possible. Two kinds of resources was recognized as suitable for exascale scientific applications by our user communities, namely: high performance computing (HPC) resources, and cloud resources. HPC manager is based on a queuing approach. Manager of cloud resources is based on REST API. Both types of resources are often enriched by support from high-throughput resources or accelerated resources. For example, GPU utilization within machine learning and deep learning application is very commonly required by those user communities, however, the requirement is still quite hard to satisfy. Since clusters build on CPUs can be used by every community, big clusters with strong GPUs are not very common nowadays.

IV. PROCESS ARCHITECTURE – AN EXAMPLE OF AN EXASCALE ARCHITECTURE

The PROCESS project is one of exascale research projects funded by the European Union's Horizon 2020 research and innovation programme. One of its main outputs will be a modular software platform which will be capable to handle exascale challenges coming from both scientific communities as well as industry.

The PROCESS architecture shows how the exascale platforms look in the real world. It was introduced in [5] and extended by a micro-services approach which is described in the following text. Micro-infrastructure [4] is a very specialized and autonomous set of services and adaptors which interact across the extreme large data service-oriented infrastructure. Alongside the efficiency mentioned above, the approach supports scalability, high adaptability, modularity, and straightforward

[4]The set of integrated micro-services.

integration with the virtual layer. Since each use case has its own requirements and dependencies, modularity together with high adaptability are very important and useful properties of every exascale environment.

The data services expose interaction points through a REST management API where users can manage their private micro-infrastructure and a set of external infrastructure endpoints such as WebDAV. The authorization to the web services is ensured through tokens. Generation of the access tokens is achieved through a global access and authorization service.

Another typical characteristic of the exascale environment is handling of different elements for processing, distribution, and management, which requires specific hardware, or nodes. These requests are possible to satisfy by the micro-infrastructure composed of dedicated nodes, or services addressing a particular request. Since the requirements are handled by virtualization typically (abstracting details of the hardware infrastructure, or the software stack), and so the micro-infrastructure offers a natural solution.

Figure 6 depicts the changes of the initial PROCESS architecture [5], [11] needed to involve the micro-infrastructure approach into the initial architecture. All the changes are highlighted in magenta. The main change is a new way of accessing data sources (through data adapters). The described approach also simplifies it. The new version has one "branch" instead of two "branches" (one dedicated to pre/post processing tools; e.g., DISPEL, and the other dedicated to pure data access through the distributed virtual file system; e.g., LOBCDER). It also influences IEE (Jupyter is a part of micro-infrastructure, thus the IEE needs only a plugin for it), and LOBCDER (the data infrastructure management layer responsible for integration of lower adjacent tools was added).

The PROCESS architecture is also a result of applying the reference exascale architecture which represents the common features of the PROCESS platform (as well as every exascale-related platform, or application). On its top users are interacting with the platform through a secure access. IEE represents the environment for users, however, security is out of the project scope. Therefore, this aspect is not investigated anymore. The whole resource infrastructure is orchestrated by the virtual layer. The technology responsible for it is Cloudify. Below that layer is situated a virtual file system alongside HPC and Cloud managers. The virtual file system is containerized through micro-infrastructure. The main reason behind this decision is that the use cases have different requirements (e.g. need to access various data sources). Micro-infrastructure containers are managed by Kubernetes. Last but not least, HPC and Cloud managers. Both of them have to be scheduled and users have to have information coming from monitoring tools about their task as well as raw hardware infrastructure. Rimrock is used as a unified environment for managing HPC resources, and Atmosphere for managing of cloud resources.

V. Conclusion

This paper presents the requirements analysis of four use cases from the PROCESS project. The medical use case represents a computationally intensive application requiring accelerated computing. The LOFAR use case requires highly effective exascale workflows for exascale datasets. The ancillary pricing use case embodies exascale throughput requirements. The Copernicus use case requests for exascale data extraction methods.

According to the requirements analysis, the reference exascale architecture is proposed. The reference architecture combines several design approaches (e.g. modularity, service-oriented architectures) with computational paradigms (e.g. high-performance computing, cloud computing, accelerated computing). Consequently, the reference architecture is used within the PROCESS project as a blueprint for the PROCESS architecture.

References

[1] P. Wittenburg, H. Van de Sompel, J. Vigen, A. Bachem, L. Romary, M. Marinucci, T. Andersson, F. Genova, C. Best, W. Los *et al.*, "Riding the wave: How europe can gain from the rising tide of scientific data," 2010.

[2] J.-G. Lee and M. Kang, "Geospatial big data: challenges and opportunities," *Big Data Research*, vol. 2, no. 2, pp. 74–81, feb 2015.

[3] R. Kune, P. K. Konugurthi, A. Agarwal, R. R. Chillarige, and R. Buyya, "The anatomy of big data computing," *Software: Practice and Experience*, vol. 46, no. 1, pp. 79–105, oct 2016.

[4] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina *et al.*, "The opportunities and challenges of exascale computing–summary report of the advanced scientific computing advisory committee (ascac) subcommittee," *US Department of Energy Office of Science*, 2010.

[5] L. Hluchý, M. Bobák, H. Müller, M. Graziani, J. Maassen, H. Spreeuw, M. Heikkurinen, J. Pancake-Steeg, S. Spahr, N. O. vor dem Gentschen Felde, M. Höb, J. Schmidt, A. S. Z. Belloum, R. Cushing, P. Nowakowski, J. Meizner, K. Rycerz, B. Wilk, M. Bubak, O. Habala, M. Šeleng, S. Dlugolinský, V. Tran, and G. Nguyen, "Heterogeneous exascale computing," in *Recent Advances in Intelligent Engineering*, L. Kovács, T. Haidegger, and A. Szakál, Eds. Cham, Switzerland: Springer Nature Switzerland AG, 2020, ch. 5, pp. 81–110.

[6] T. Shimwell, H. Röttgering, P. N. Best, W. Williams, T. Dijkema, F. De Gasperin, M. Hardcastle, G. Heald, D. Hoang, A. Horneffer *et al.*, "The lofar two-metre sky survey-i. survey description and preliminary data release," *Astronomy & Astrophysics*, vol. 598, p. A104, 2017.

[7] O. Jimenez-del Toro, S. Otálora, M. Andersson, K. Eurén, M. Hedlund, M. Rousson, H. Müller, and M. Atzori, "Analysis of histopathology images: From traditional machine learning to deep learning," in *Biomedical Texture Analysis*. Elsevier, 2018, pp. 281–314.

[8] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2017.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[10] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2755–2763.

[11] M. Bobák, A. S. Z. Belloum, P. Nowakowski, J. Meizner, M. Bubak, M. Heikkurinen, O. Habala, and L. Hluchý, "Exascale computing and data architectures for brownfield applications," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2018 14th International Conference on*. IEEE, 2018, pp. 461 – 468.